A Suite of Semantic Web Tools Supporting Development of Multilingual Ontologies

Maria Teresa Pazienza, Armando Stellato, Andrea Turbati

ART Group, Dept. of Computer Science, Systems and Production

University of Rome, Tor Vergata

Via del Politecnico 1, 00133 Rome, Italy

{pazienza, stellato, turbati}@info.uniroma2.it

Abstract The multilingual aspects which characterize the (Semantic) Web and the constant demand for more understandable and easy-to-share forms of knowledge representation, push for a more "linguistically aware" approach to ontology development and foresees an environment where formal semantics could coexist with natural language, contributing to improve "shareability" of the content they describe. As a consequence ontologies should be enriched to both cover formally expressed conceptual knowledge as well as to expose content in a linguistically motivated fashion. In this paper we present a suite of tools, libraries and ontologies, ranging from ontology development to language resources access and management, supporting the development of multilingual ontologies. The contribution of this work, going beyond mere tool presentation, is two-fold: the presented tools implicitly embody a new way (methodology?) of rethinking the development of ontologies in terms of making their content easy reusable and comprehensible; moreover, they represent living proofs of software engineering principles associated to software reuse, documentation, modularity, interaction analysis, applied to the domain of Knowledge Management Software

Introduction

Semantic Web ontologies represent the shared vocabularies through which machines can read and access content from the Web, or even communicate between them, to exchange information or cooperate for achieving some goal. This definition implicitly assumes that in an heterogeneous scenario like the whole WWW, the same concepts will be represented by the same ontologies and that, therefore, ontological models of data will be consistent; conversely, sensible effort will be put in trying to match these "not-so-shared" vocabularies. If that general assumption may hold true for reduced-size, very specific and data-oriented ontologies (e.g. the WGS84 Geo Positioning RDF vocabulary[1], which contains only a few properties for describing latitude, longitude and point-in-space concepts), for larger domain descriptions, requiring different levels of abstraction and different perspectives depending on local needs, we expect to see several, different ontologies arise from independent organizations, often addressing overlapping domains.

Two issues then urge to be solved: first, facilitating people and automated systems in performing alignments between ontologies where they represent the same concepts and, secondly, make their vocabularies more explicit to humans, so that they can be re-used consistently in different scenarios and by different actors; in this sense, logical consistency may only help in restricting the range of possible interpretations which may be assigned to logical symbols, while common-sense human reasoning using these vocabularies may beneficiate a lot by the presence of clear and exhaustive documentation. Extensive use of Natural Language contents, providing free descriptions, synonymical expressions and translations in different idioms of the intended meaning of a vocabulary, appears thus as the most intuitive kind of documentation for data structures such as ontologies, dealing with representation of domains. Several efforts have been undertaken to cover different aspects of this problem, motivating the adoption of linguistic resources for enriching ontology vocabularies with natural language contents [25,32,34,31,15], showing useful applications exploiting these combined resources [2,30,6], providing standards for representing this enrichment/integration, like in SKOS[2] (Simple Knowledge Organization Systems) and in [4], and promoting the development of techniques for automating this task [26].

Objective of our research work, which moves in between the Ontology Engineering and Natural Language Processing areas, is to strongly integrate conceptual and linguistic knowledge to reduce the everlasting gap which exists between these two forms of knowledge representation, breaking down the barrier between what is known as the "world model" of intelligent systems, and what is the "world outside there", characterized by real documents written in natural language.

In this context, a suite of tools, libraries and ontologies dedicated to the development of multilingual ontologies will be presented. First, the Linguistic Watermark Ontology Suite and Java Library: a suite of ontologies for describing both linguistic resources and software interfaces for accessing their content, other than representing (multi)lingual information inside ontologies, and a java extensible library providing interfaces (and a few implementations) for covering all of the above tasks. Then, the OntoLing Framework will be showed: a portable extension for ontology development tools supporting manual and semi-automatic annotation of ontological data with information from different, heterogeneous linguistic resources. Lastly, we describe Semantic Turkey, a Web Browser extension for Knowledge Management and Acquisition of Semantic Web data, and introduce for the first time OntoLing-ST, an implementation of the recent OntoLing 4.0 which, thanks to its high portability across different platforms and ontology stan-

[1] http://www.w3.org/2003/01/geo/wgs84_pos

[2] http://www.w3.org/TR/swbp-skos-core-guide/

dards, has been easily integrated in the Semantic Turkey environment. Before introducing the above tools, section 2 will discuss state-of-the-art on language representation and linguistic resource modeling, while section 3 exposes our desiderata in reconsidering the process of ontology development, and details requirements for building applications for multilingual ontology development.

State of the Art and Standards for Linguistic Resources and Language Representation

"The term linguistic resources refers to (usually large) sets of language data and descriptions in machine readable form, to be used in building, improving, or evaluating natural language (NL) and speech algorithms or systems"[8]. Examples of linguistic resources are written and spoken corpora, lexical databases, grammars, treebanks and field notes. In particular, this definition includes lexical databases, bilingual dictionaries and terminologies (which can all be indicated as lexical resources), which may reveal to be necessary in the context of a more linguistic-aware approach to KR. In past years, several lexical resources were developed and made accessible (a few for free), and a wide range of resources is now available, ranging from simple word lists to complex MRDs and thesauruses. These resources largely differentiate between the explicit linguistic information they expose, which may vary in format, content granularity and motivation (linguistic theories, task or system-oriented scope etc…).

Multiple efforts have been spent in the past towards the achievement of consensus among different theoretical perspectives and systems design approaches. The Text Encoding Initiative (www.tei-c.org) and the LRE-EAGLES (Expert Advisory Group on Linguistic Engineering Standards) project [5] are just a few, bearing the objective of making possible the reuse of existing (partial) linguistic resources, promoting the development of new linguistic resources for those languages and domains where they are still not available, and creating a cooperative infrastructure to collect, maintain, and disseminate linguistic resources on behalf of the research and development community.

A more recent effort is given by the Lexical Markup Framework [11] – which is now pursuing ISO standardization – a UML-based model for the description of Lexical Resources. However, at the present time, a definitive standard is not available. Often, even a local agreement on the model adopted to describe a given (a series of) resource does not prevent from an incorrect formulation of its content. This is due to the fact that many resources have been initially conceived for humans and not for machines. As an example, in existing available dictionaries, the definitions of words and synonyms are not always managed the same way: in some cases synonyms are clustered upon the senses which are related to the particular term being examined (among others, Babylon, www.babylon.com, and Dict, www.dict.org/bin/Dict dictionaries, where the senses are separated by a ";"

symbol), other simply report flat lists of terms without even identifying their different meanings (as in Freelang dictionaries: www.freelang.com). In several dictionaries, synonyms are mixed with extended definitions (glosses) in an unpredictable way and it is not possible to automatically distinguish them. Terms reported as synonyms may sometimes not be truly synonyms of the selected term, but may represent more specific or general concepts (this is the case of the Microsoft Word synonymn prompter). Of course, the ones mentioned above represent mere dictionaries not adhering to any particular linguistic model, though they may represent valuable resources on their own.

A much stronger model is offered by WordNet [21,10], which, being a structured lexical database, presents a neat distinction between words, senses and glosses, and is characterized by diverse semantic relations like hypernymy/hyponymy, antonymy etc… Though not being originally realized for computational uses, and being built upon a model for the mental lexicon, WordNet has become a valuable resource in the human language technology and artificial intelligence. Due to its vast coverage of English words, WordNet provides general lexico-semantic information on which open-domain text processing is based. Furthermore, the development of WordNets in several other languages [37,33,35] extends this capability to trans-lingual applications, enabling text mining across languages.

Linguistic Enrichment of Ontologies: motivation and desiderata

Ontology Development is a task requiring considerable human involvement and effort, at a large extent with the objective of providing a shareable perspective over domain related knowledge. What “shareable” means, depends on the nature of the task(s) the ontology is thought for. The scenario offered by the Semantic Web is in fact characterized by distributed services which must both realize and rely on a proper connection of machine-accessible formal semantics and more traditional Web content.

For this connection to be true, a complete Ontology Development process should consider the formal aspects of conceptual knowledge representation, as well as guarantee that the same knowledge be recognizable amongst its multiple expressions which are available on real data: that is language.

To achieve such an objective, we should reconsider the process of Ontology Development to include the enrichment of semantic content with proper lexical expressions in natural language. Ontology Development tools should reflect this need, supporting users with dedicated interfaces for browsing linguistic resources: these are to be integrated with classic views over knowledge data such as class trees, slot and instance lists, offering a set of functionalities for linguistically enriching concepts and, possibly, for building new ontological knowledge starting from linguistic one.

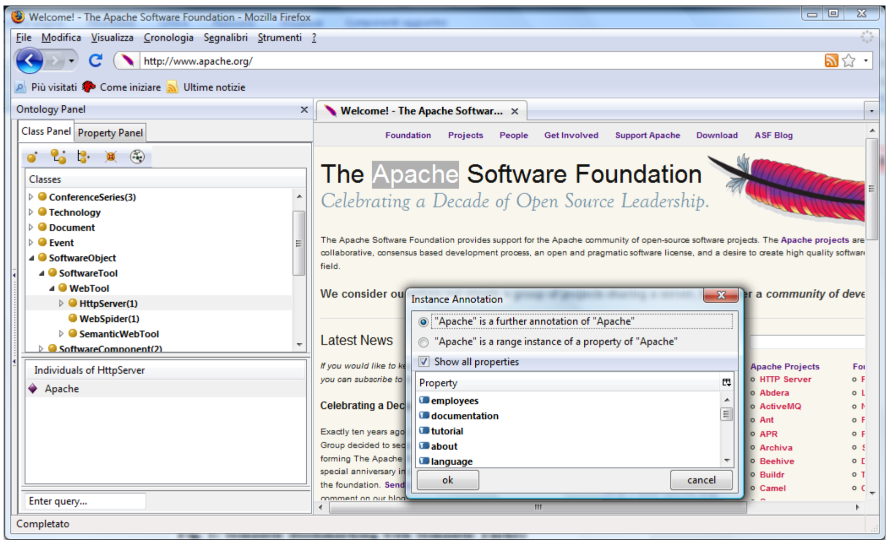

Welcome! - The Apache Software Foundation - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti ?
http://www.apache.org/
Più visitati Come iniziare Ultime notizie
Ontology Panel
Class Panel Property Panel
Classes
ConferenceSeries(3)
Technology
Document
Event
SoftwareObject
SoftwareTool
WebTool
HttpServer(1)
WebSpider(1)
SemanticWebTool
SoftwareComponent(2)
Individuals of HttpServer
Apache
Enter query...
Welcome! - The Apache Softwar...
Foundation Projects People Get Involved Support Apache Download ASF Blog
The Apache Software Foundation
Celebrating a Decade of Open Source Leadership.
Latest News
Instance Annotation
[✓ "Apache" is a further annotation of "Apache"]
"Apache" is a range instance of a property of "Apache"
[✓ Show all properties]
Property
employees
documentation
tutorial
about
language
ok
cancel
Apache Projects
HTTP Server
Abdera
ActiveMQ
Ant
APR
Archiva
Beehive
Buildr
Camel
Completato

Fig. 1: Semantic Bookmarking with Semantic Turkey

By considering some of our past experiences [1,29,27] with knowledge based applications dealing with concepts and their lexicalizations, a few basic functionalities for browsing linguistic resources (from now on, LRs) emerged to be mandatory:

- *Search term definitions (glosses)*
- *Ask for synonyms*
- *Separate different sense of the same term*
- *Explore genus and differentia*
- *Explore resource-specific semantic relations*

as well as some others for ontology editing:

- *Add synonyms (or translations, for bilingual resources) as additional labels for identifying concepts*
- *Add glosses to concepts description (documentation)*
- *Use notions from linguistic resources to create new concepts*

While ontologies have undergone a process of standardization which culminated, in 2004, with the promotion of OWL as the official ontology language for the semantic web, linguistic resources still maintain heterogeneous formats and follow different models, which make tricky the development of such an interface.

In the next sections we present our suite of tools for multilingual ontology development, starting by first through our ontology development and knowledge acquisition framework Semantic Turkey, and then presenting the suite of ontologies, software libraries and tools supporting multilingual enrichment of ontologies

Semantic Turkey

Semantic Turkey [14] was born inside a national project – funded by the FILAS agency (Finanziaria Laziale di Sviluppo) under contract C5748-2005 – focused on innovative solutions for browsing the web and for collecting and organizing the information observed during navigation (Fig. 1).

The prototype for the project immediately took the form of a Web Browser extension allowing users to annotate information from visited web sites and organize it according to a personally defined domain model: Semantic Turkey paradigmatic innovation was in fact to "obtain a clear separation between (acquired) knowledge data (the WHAT) and web links (the WHERE)" pointing to it. That is, to be able, through very easy-to-use drag'n'drop gestures, to *select* textual information from web pages, *create* objects in a given domain and *annotate* their presence in the web by keeping track of the selected text and of its provenience (web page *url*, *title* etc…). We coined the expression "semantic bookmarking" for this kind of activity.

Due to its proverbial extendibility, the Firefox platform[3] had been chosen as the hosting browser for our application, while Semantic Web standards and technologies were the natural candidate for representing its knowledge model.

Standing on top of mature results from research on Semantic Web technologies, like Sesame [3] and OWLim [18] as well as on a robust platform such as the Firefox web browser, ST (Semantic Turkey) differentiates from other existing approaches which are more specifically tailored respectively towards knowledge management and editing[13], semantic mashup and browsing [9,16] and pure semantic annotation [7,17], by introducing a new dimension which is unique to the process of building new knowledge while exploring the web to acquire it.

By focusing on this aspect, which has been further investigated in the two years of finalization leading to the current release, we went beyond the original concept of Semantic Bookmarking and tried to amplify the potential of a new Knowledge Management and Acquisition System: we thus aimed at reducing the impedance mismatch between domain experts and knowledge investigators on the one side, and knowledge engineers on the other, providing them with a unifying platform for acquiring, building up, reorganizing and refining knowledge.

Semantic Turkey Architecture

The architecture (Fig. 2) of Semantic Turkey follows a three layered design, with the presentation layer embodying the true Firefox extension and the other two layers built around java technologies for administering the business logic and data access.

[3] http://www.mozilla.com/en-US/firefox/

Everything relating user interaction is directly managed by the Firefox extension, thanks to a solution directly integrated in the browser. This approach has two main advantages: total reuse of the functionalities of a well assessed, stable and complete software for web browsing, and a non invasive offer for the user, who can still use the web browser he has been acquainted with.

The second layer, the service layer, is realized through a collection of Java Web Services, published through the Web Server "Jetty"[4]. Jetty is implemented entirely in Java, and the architecture foresees its use as an embedded component. This means that the Web Server and the Web Application run in the same process, without interconnection overheads and other sort of complications.

The following sections describe more in detail the three layers which constitute the architecture of Semantic Turkey

Presentation Layer. The User Interface has been created through a combined use of the XML User Interface Language XUL[5], XBL[6] and Javascript language.

The UI physically appears as a set of Firefox sidebar, representing ontological information. User requests are handled through the Ajax [12] paradigm: the data – in XML format – is thus mainly exchanged between the two layers in an asynchronous way, to preserve good performance and to not penalize the activity of the browser.

Javascript XPCOM[7] components have been developed and the Simile Java Firefox Extension[8] has been adopted for linking the chrome part and the Java part to start the Jetty embedded java server.

Middle Layer. This layer offers services which may be invoked through http requests submitted according to the Ajax paradigm, thus enabling communication between the client (Firefox extension) and the server. The server receives the requests coming from the client by GET or POST http calls, carries out the operations associated to these calls, and in case replies with an XML response. If a call implies the return of a XHTML page, an XSLT transformation is being performed, in order to decouple the data model with its manifestation in the presentation layer.

The majority of invocations to the server are being completed in an asynchronous way, so that, independently from the workload that is subjected the server, the browser can continue to respond to the user. This is a crucial issue for the usability of the application: expensive computations blocking normal behavior of the browser would otherwise not be tolerated by the user.

Besides supporting the communication with the client, the middle layer provides the functionalities for definition, management and treatment of the data. Several objects are described through an ontological model (see next section), to

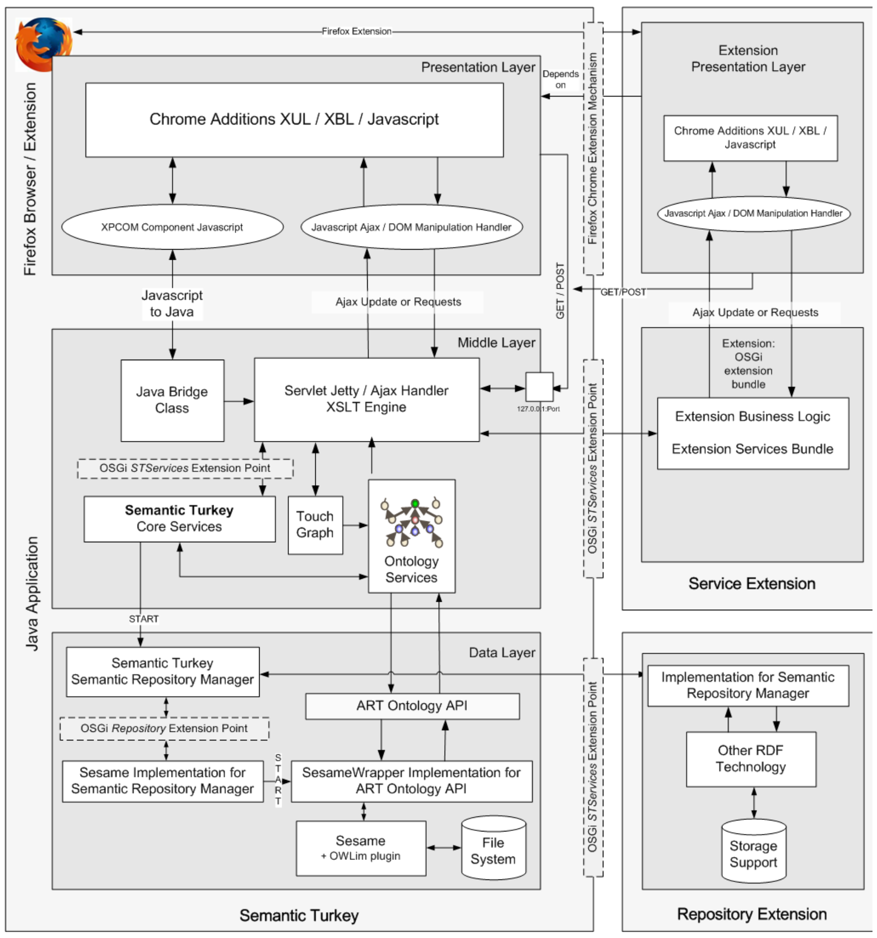
[4] http://jetty.mortbay.org/jetty/

[5] http://www.mozilla.org/projects/xul/

[6] http://www.mozilla.org/projects/xbl/xbl.html

[7] http://www.mozilla.org/projects/xpcom/

[8] http://simile.mit.edu/java-firefox-extension/



Firefox Extension
Firefox Browser / Extension
Presentation Layer
Depends on
Chrome Additions XUL / XBL / Javascript
XPCOM Component Javascript
Javascript Ajax / DOM Manipulation Handler
Firefox Chrome Extension Mechanism
Extension Presentation Layer
Chrome Additions XUL / XBL / Javascript
Javascript Ajax / DOM Manipulation Handler
GET / POST
GET/POST
Javascript to Java
Ajax Update or Requests
Ajax Update or Requests
Middle Layer
Java Bridge Class
Servlet Jetty / Ajax Handler XSLT Engine
127.0.0.1:Port
OSGi STServices Extension Point
Extension: OSGi extension bundle
Extension Business Logic
Extension Services Bundle
OSGi STServices Extension Point
Semantic Turkey Core Services
Touch Graph
Ontology Services
Java Application
Service Extension
START
Data Layer
Semantic Turkey Semantic Repository Manager
ART Ontology API
OSGi Repository Extension Point
Implementation for Semantic Repository Manager
Other RDF Technology
START
Sesame Implementation for Semantic Repository Manager
SesameWrapper Implementation for ART Ontology API
Sesame + OWLim plugin
File System
OSGi STServices Extension Point
Storage Support
Semantic Turkey
Repository Extension

Fig. 2 Architecture of Semantic Turkey and of its extensions

represent both pure conceptual knowledge as well as application required information.

Data layer. It is mainly constituted by the component for managing the ontology. This has recently been rewritten as a series of dedicated API for accessing ontological data: these offer both RDF triple-level access methods as well as more object oriented facilities, which have been appreciated in RDF libraries like Jena [20]. Semantic Turkey API constitute an interface which can be implemented by building wrappers for existing ontology libraries, so that we could easily select those which best fit the needs of a given situation (like working with small or large repositories, on a local or collaborative environment etc…) without having to modify the whole application. The first implementation of these API has been developed as a wrapper for Sesame [3] and the OWLIM plugin [18], which has been added for reasoning over OWL [38] data.

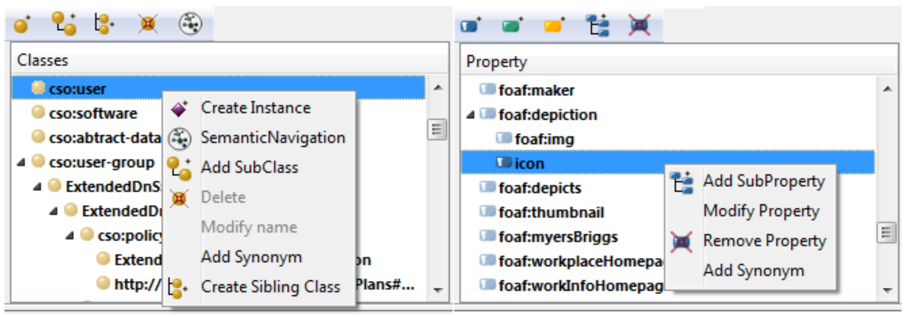


Fig. 3 Class and Property panels in Semantic Turkey

Semantic Turkey also features an extension mechanism supporting both technologies belonging to the Front End and the Business and Data Layers.

The whole extension mechanism is obtained by a proper combination of the Mozilla extension framework (which is used to extend the user interface, drive user interaction and add/modify browser functionalities of ST) and the OSGi java extension framework [23] (providing extensions capabilities for the service and data layers of the architecture). OSGi compliance is obtained through the OSGi implementation developed inside the Apache Software Foundation, called Felix (felix.apache.org/).

Two main extension points have been introduced: a Service extension and a Repository Extension. The first one allows for the development of arbitrary services which can be added dynamically to the system. Extensions of this type typically need to realize both a client extension through Mozilla technology, by adding new functionalities (and hooks for them in the user interface) to the system, as well the corresponding Service which is added dynamically through OSGi.

The second kind of extension provides openness to different triples store technologies; Semantic Turkey is in fact no more strictly based on the Sesame + OWLim libraries for RDF management, but features proprietary APIs for querying the managed ontologies. These API are defined through a set of interfaces, which can be implemented to adopt different triple stores. This can be of particular interest in specific scenarios where the target user has to connect to a specific triplestore, or where a service extension is being built by annexing an existing application, and in either case, these are based on a different triple store technologies.

Both kind of extensions are deployable as an xpi (cross-platform installers) packages which, once installed inside Firefox, are handled by Semantic Turkey extension discovery system, which extracts OSGi bundles and installs them in the main application. This assures easy installation for the user, which can install ST extensions as any other Firefox one, by dragging the xpi over Firefox and restarting the browser.

User Interaction

Semantic Turkey offers editing operations for populating the *personal ontology* with annotations from visited web sites, as well as search and navigation functionalities which facilitate the recovery of already acquired knowledge.

Main functionalities The user may interact with the ontology panel to modify its personal ontology, through a series of operations, which we describe here, organized into categories.

Interaction with the browser. These mainly include drag&drop operations which allow to annotate information from the visited sites:

- Drag and drop of a selection of a text from an html document displayed in the browser, on the icon that represents a class, in order to create an individual of that class. The selection will become the local name of the new individual, which will be shown inside the instances panel
- Drag and drop of a selection of text from an html document, on the icon that represents an individual, in order to add a further bookmark for that individual, or to characterize a property which that individual owns. A specific window will open, prompting the user to choose the appropriate functionality. In the first case, a new semantic annotation is taken for the individual, with a new webpage as a bookmark for it and the new textual occurrence of that individual in the observed page. In the other case, the user can choose a property for enriching the description of the chosen individual through the selected text. If the selected property is an *owl:ObjectProperty*, the selection will become the name of a new individual created as an instance of the range class of the chosen property, or a further annotation for an existing individual. In both cases, the two individuals are bound through the selected property. In case of an *owl:Datatype* or *owl:AnnotationProperty*, a new value will be added.
- Drag and drop of a selection of text from an html document, on the icon that represents an individual, in order to define a further lexicalization for that individual. The user can choose, from the same panel described before, if the selection characterizes a range of a property or a new *lexicalization*

These functionalities have been conceived to speed up typical series of operations which characterize both the worlds of ontology development and semantic annotation. For example, the second one which has been described above performs, in case of an object property, the creation of a new instance, its annotation with the current web page and the assertion of a relationship between the new individual and the selected one, at the cost of just a drag&drop and a selection.

Direct Ontology Editing. These functionalities operate exclusively on the ontologies, as it should be important for the user to integrate the knowledge acquired through semantic bookmarking with information he could get through other media. All typical ontology editing operations (Fig. 3) are carried out through buttons and context menus associated to the nodes of the tree, in a way much similar to

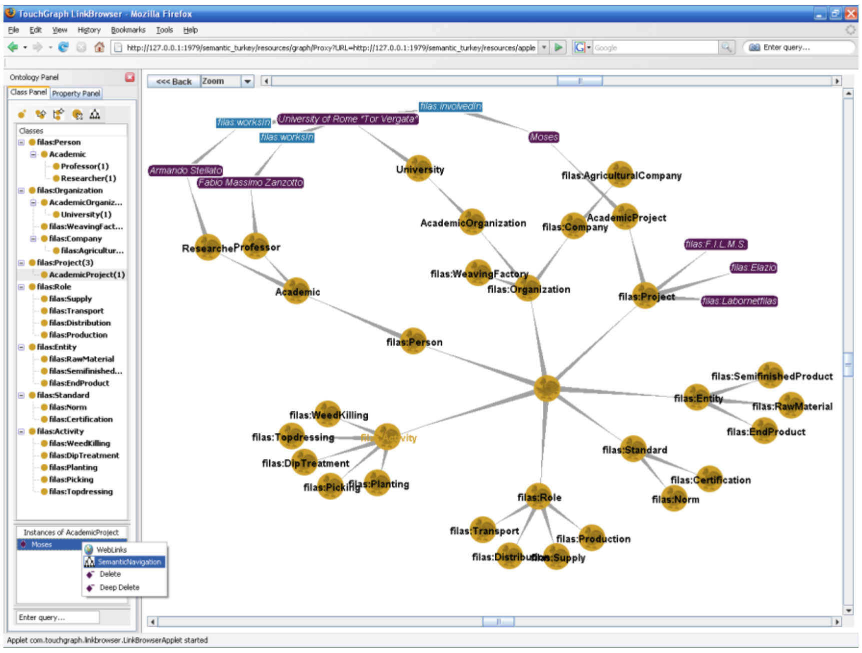


Fig. 4 Class and Property panels in Semantic Turkey

traditional ontology editing tools, like Protégé [13] or TopBraid Composer[9]. By offering complete interaction with the ontology via the XUL interface (instead of an HTML interface, like in Piggy-Bank), the user is not diverted from his current navigation (i.e. the main browser panel is still focused on the visited web page, which would otherwise be replaced by the HTML UI) and may, at the same time, maintain its attention over the observed web page. Extended support for *natural language descriptions* of ontology objects is also present in the system, allowing for explicit representations of the same objects through different synonimical expressions, or translation for different idioms, thus accounting for multilinguism. This is a further aspect to be distinguished from keeping track of the several ways in which ontology objects have been annotated over web pages, since this last is thought for addressing other phenomena, like acronyms, misspells and other idiosyncratic expressions.

Semantic Browsing As an additional feature, the user may graphically explore the ontology, thanks to the *SemanticNavigation* component: a customized version of the TouchGraph library[10] allowing for a graph-like exploration of ontology nodes. A Java applet will be loaded on a new tab of the browser, displaying the graph view of the ontology, allowing the user to navigate its content. The nodes of the

[9] http://simile.mit.edu/java-firefox-extension/

[10] http://touchgraph.sourceforge.net/

graph will be displayed in different manners, according to the nature of the onto-logical entity: classes, properties or individuals. By dragging the mouse pointer on a node that represents an individual, it is possible to open a popup window, which contains the URLs of the pages where that instance has been annotated.

The Linguistic Watermark

The Linguistic Watermark [28] is an ontological and software framework for describing, referring and managing heterogeneous linguistic resources and for using their content to enrich and document ontological objects. It articulates into two results: first, a set of coordinated RDF vocabularies providing descriptors for representing linguistic resources (ranging from lexical to frame-based ones) and their software counterparts (data structures, access libraries etc…), as well as offering metadata for describing the linguistic enrichment of ontologies, both on quantitative and qualitative grounds. The second result is a software library for evaluating the quality of automatic linguistic enrichment tools, through comparison of enriched ontologies compiled against the above vocabularies.

The Linguistic Watermark Ontology Suite

The Linguistic Watermark suite of RDF vocabularies is composed of three ontologies:

– The *Linguistic Watermark (LW)* vocabulary, describing linguistic resources through their purposes and structure organization
– *The Ontological Linguistic Watermark (OLW)* vocabulary: a set of metadata descriptors for characterizing the linguistic expressivity of ontologies
– *The LW Linguistic Interfaces vocabulary (LWLI)*, providing concepts for describing software libraries which grant access to specific (or ranges of) linguistic resources

The Linguistic Watermark (LW) Vocabulary

While the Linguistic Watermark vocabulary partially covers general linguistic concepts like term, word, lexical/semantic relation, frame, agent etc... its main objective is to provide descriptors or characterizing the purpose and structure of linguistic resources: whether they represent translation vocabularies, synonyms collections, lexicons, frame based resources or terminologies, if they are organized around some kind of semantic structure or merely <entry, description> pairs etc..

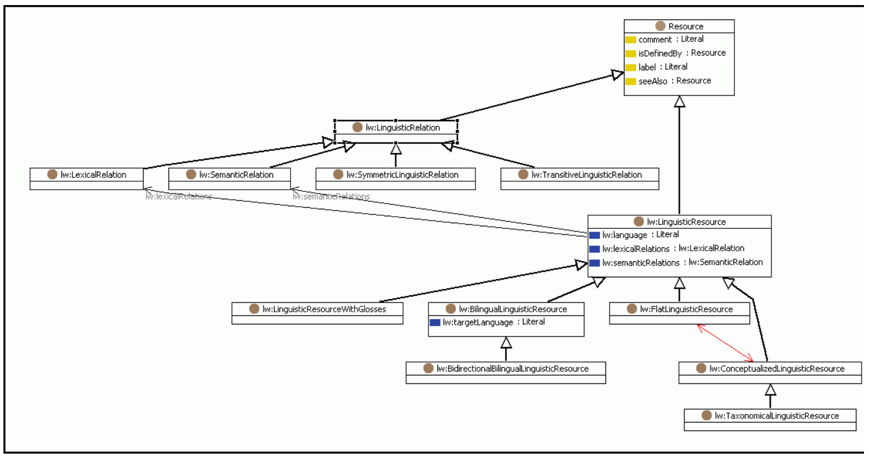

Resource
comment : Literal
isDefinedBy : Resource
label : Literal
seeAlso : Resource
lw:LinguisticRelation
lw:LexicalRelation
lw:SemanticRelation
lw:SymmetricLinguisticRelation
lw:TransitiveLinguisticRelation
lw:lexicalRelations
lw:semanticRelations
lw:LinguisticResource
lw:language : Literal
lw:lexicalRelations : lw:LexicalRelation
lw:semanticRelations : lw:SemanticRelation
lw:LinguisticResourceWithGlosses
lw:BilingualLinguisticResource
lw:targetLanguage : Literal
lw:FlatLinguisticResource
lw:BidirectionalBilingualLinguisticResource
lw:ConceptualizedLinguisticResource
lw:TaxonomicalLinguisticResource

Fig. 5 An excerpt (focused on main descriptors for Linguistic Resources) from the Linguistic Watermark vocabulary

Though originally conceived to cover any kind of Linguistic Resource, the first version of the Linguistic Watermark (Fig. 5) was limited to represent only lexical resources: by proper combination of its LW ontological descriptors, one could be able to represent very different linguistic resources, from simple synonym dictionaries, to complex resources such as WordNet[21]. This provided a shared and homogeneous vocabulary upon which multilingual (and multi-resource) applications could be defined.

In this work we have extended le LW vocabulary into two main directions:

– *RDF Porting*: now the LW model can be expressed as an RDF vocabulary
– *Instantiation*: now the vocabulary is not only used to describe linguistic resources, but even to predicate over their content (see section 4.2.2 for details)

Frames description: covering frame/class based linguistic resources, such as FrameNet and VerbNet (see [22] for further details).

The Ontological Linguistic Watermark (OLW)

The characterization given by the OLW is expressed in terms of the linguistic content of the described ontology and with respect to the resources which have been adopted for enriching its concepts. As stated in [30], where its adoption has been considered in a scenario involving Semantic Coordination of FIPA agents, its metadata assume great significance in all the contexts where ontologies sharing a common domain, but no explicit semantic bridging between their respective vocabularies, need to be automatically aligned or merged. Resource-based algorithms for ontology alignment and semantic coordination agents can in fact inspect the OLW data of the ontologies to be compared and configure at best the resources and facilities to be used for matching their content. This is an aspect which has often been underestimated in literature: setting up the resources to be adopted in a realistic scenario, while being not a trivial task, influences dramatically the outcome and performances of any mediation activity.

The LWLI takes its roots from the first version of the Linguistic Watermark software library[11] – developed by the University of Rome, Tor Vergata – a component providing uniform access to different and heterogeneous linguistic resources, which has been used in several resource-based tools, such as the OntoLing Protégé plug-in [25]. The LW presented in that work, was just a class diagram offering several interfaces and abstract classes whose combination could be used to describe the main aspects of a linguistic resource: implementing the proper subset of those (software) interfaces would result in the definition of a linguistic wrapper for accessing a particular linguistic resource. The LW library thus offered a combination of descriptive (with regard to the resources to be wrapped) and operative aspects (delineating the operations which the required wrapper had to implement). Later on, the exigencies which brought to developing the OLW, required a formal ontological representation, merely focused on resource description, to be extracted from the original class diagram, which led to the LW.

Now, it was time to close the circle, and with the LWLI we recovered the original intent of the LW library.

The LW Linguistic Interfaces vocabulary (LWLI)

LWLI contains concepts describing parameters needed by software libraries for setting up access to their target linguistic resources. This third ontology completely migrates the original framework to RDF, thus providing a complete vocabulary at the hand of Semantic Web tools which rely on the use of linguistic resources or are even expressly dedicated to the integration of ontologies with linguistic resources.

The LWLI includes concepts like:

– *LinguisticInterface*: for describing a specific implementation of a wrapper for a linguistic resource
– *LinguisticInterfaceConfiguration*: representing instances of basic runtime configurations for a given *LinguisticInterface*.
– *LinguisticInterfaceInstanceConfiguration*: each instance of this class provides data for completing a single runtime configuration for accessing a specific linguistic resource, basing on partial configuration from a given *LinguisticInterfaceConfiguration*.

and properties for specifying these configuration settings, among which, we list the following ones:

[11] http://art.uniroma2.it/software/LinguisticWatermark/

- *configuredInterface*: this property tells which *LinguisticInterface* is being configured through the described configuration
- *interfaceableResource*: tells which linguistic resources are made accessible through the described *Linguistic Interface*
- *ConfigurationProperty*: a property defining configuration parameters for accessing a linguistic resource through a dedicated linguistic interface. This property is never instantiated, though it has a few relevant subproperties for telling whether a given configuration parameter points to the file system, if a property is relevant for configuring a linguistic interface (InterfaceProperty) as a whole, or just for accessing specific resources (InstanceProperty) etc..

As for the LW, even this vocabulary provides an upper ontology which, though extensible in principle to match the specification of each represented software library, already contains all the required descriptors for automatically driving different linguistic resources under a shared knowledge model.

The Linguistic Watermark library

Following the recent improvements on the LW suite, we are releasing a new version of the Linguistic Watermark library (LW 3.0), which offers java API for accessing linguistic resources through dedicated Linguistic Interfaces, both entities being defined according to the LW and LWLI vocabularies. In particular, a mapping between the above ontologies and newly added java interfaces allows implemented java wrappers for linguistic resources to declare themselves as new instances of the LinguisticInterface class and accept strongly typed configuration parameters, thus enabling data consistency checks and providing hooks for automatic generation of configuration user interfaces for hosting applications.

To implement this mechanism we adopted and OSGi compliant java extension framework: Apache Felix (felix.apache.org/). Each OSGi bundle (the OSGi name given to the extension packages) contains a class that extends the abstract class LI-Factory (see class diagram in Fig. 6), which is in charge of generating objects implementing the LinguisticInterface interface. Each class that implements the LinguisticInterface interface has some of its fields representing specific InterfaceProperty and InstanceProperty properties (they are automatically identified through *java annotations*). InterfaceProperties share their value among all the instances, so they are declared as static fields, while InstanceProperties have values specific to each object (identifying a specific linguistic resource present in the host). LIFactories release new instances of LinguisticInterface by getting their needed configuration (i.e. InterfaceProperties and InstanceProperties values), which is stored in a LinguisticResource object, from a loaded LW LingModel. We implemented two serializations (and related loaders/writers) of the LingModel: one compact xml represention (handled my LingModelXMLIO) and an RDF representation which follows the LW RDF Vocabulary (LingModelRDFIO).

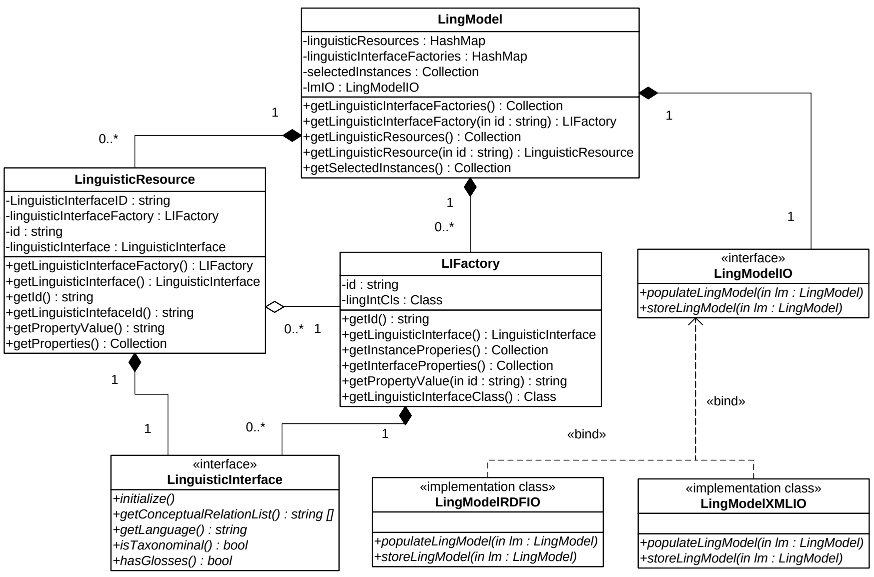

LingModel
-linguisticResources : HashMap
-linguisticInterfaceFactories : HashMap
-selectedInstances : Collection
-lmIO : LingModelIO
+getLinguisticInterfaceFactories() : Collection
+getLinguisticInterfaceFactory(in id : string) : LIFactory
+getLinguisticResources() : Collection
+getLinguisticResource(in id : string) : LinguisticResource
+getSelectedInstances() : Collection
LinguisticResource
-LinguisticInterfaceID : string
-linguisticInterfaceFactory : LIFactory
-id : string
-linguisticInterface : LinguisticInterface
+getLinguisticInterfaceFactory() : LIFactory
+getLinguisticInterface() : LinguisticInterface
+getId() : string
+getLinguisticIntefaceId() : string
+getPropertyValue() : string
+getProperties() : Collection
LIFactory
-id : string
-lingIntCls : Class
+getId() : string
+getLinguisticInterface() : LinguisticInterface
+getInstanceProperies() : Collection
+getInterfaceProperties() : Collection
+getPropertyValue(in id : string) : string
+getLinguisticInterfaceClass() : Class
«interface»
LingModelIO
+populateLingModel(in lm : LingModel)
+storeLingModel(in lm : LingModel)
«interface»
LinguisticInterface
+initialize()
+getConceptualRelationList() : string []
+getLanguage() : string
+isTaxonominal() : bool
+hasGlosses() : bool
«implementation class»
LingModelRDFIO
+populateLingModel(in lm : LingModel)
+storeLingModel(in lm : LingModel)
«implementation class»
LingModelXMLIO
+populateLingModel(in lm : LingModel)
+storeLingModel(in lm : LingModel)
«bind»
«bind»
0..*
1
1
1
0..*
1
0..*
1
1
1
0..*
1

Fig. 6 Class diagram of the main components of LW model

While there should be exactly one LinguisticInterface which is responsible for providing access to a specific loaded resource, proper handling of the LIFactory/LinguisticInterface pair can hide implementation issues related to wrapping and reusing existing foreign libraries with different architectures into this framework.

As an example, one existing library for a particular kind of resource – let us call it *LRESLIB* – could adopt one singleton object (*ResManager*) for managing different linguistic resources of the same type (different versions or for different languages). In this case, the *LRESLIB* library can be easily wrapped in the LW framework by initializing, storing and hiding *ResManager* inside its built LIFactory implementation, while the associated LinguisticInterface implementation will represent simple objects retaining reference to their LIFactory and invoking *ResManager* methods (with parameters customized for their specific resource) through delegation.

This approach guarantees reuse of existing libraries and tools for accessing linguistic resources while porting their provided content inside an extensible framework with well defined model, vocabulary and operations.

The OLW library and OLW vocabulary improvements

With the specific aim of obtaining a stable range of instruments for enriching ontologies with lexical content, and of formalizing the model and associated format for representing this information, we have developed a dedicated component which, together with the LW library, can be embedded in ontology based tools and applications needing to incorporate linguistic content.

The OLW Integration Model

In modeling our framework for the integration of ontological and linguistic content, we have taken into consideration the following requisites, which should allow for:

- Reporting quantitative and qualitative information on the overall process of enriching an ontology with content from a linguistic resource (this was the primary objective of the OLW metadata ontology)
- Keeping track (at least maintain the possibility to do that) of the source used for enriching the content
- Being able to properly map different kind of linguistic entities (words, linguistic/semantic relations etc…) with (structures of) ontological objects
- Giving the user the possibility of adopting resources' specific objects (e.g. FrameNet frames or WordNet synsets) for enriching an ontology
- Embedding existing models for integration of ontologies and linguistic entities, still respecting the above priorities
- Assessing reliable links between ontological and linguistic objects as well as taking into account for probabilistic matches produced by automatic enrichment tools (which could also be used for evaluation purposes)

The first requisite has been satisfied by defining a set of meta-descriptors – represented through object properties with domain set to owl:Ontology – for providing an overview of the "linguistic expressiveness" of ontologies. These properties may prove to be helpful for services/agents which, having to map/merge/align/mediate different ontologies, may be willing to invoke the proper linguistic resources for supporting this task. These mediators can thus beneficiate of the overall statistical information provided by the OWL metadata, without inspecting the entire ontologies' content. This part of the OLW has already been described in details in [30].

The second, third and fourth requisites have been accomplished by extending the LW; in its first incarnation, which served solely as a conceptual driver for the software library, the LW was able to express descriptions of linguistic resources, without predicating about their specific content. Now it has been extended to make possible the instantiation of objects from the described resources. The example in Fig. 7 shows fragments originating from three different ontologies: the first fragment is a description of WordNet synset 100001740 originating from the WordNet-RDF vocabulary developed by the WordNet task force of the W3C (http://www.w3.org/TR/wordnet-rdf/); the second one is the binding of concept wn20schema:Synset to the lw:SemanticIndex, through a rdfs:subClassOf relationship. Finally, a certain Noun concept coming from a fictitious ontology is enriched with the meaning expressed by the above synset, through the

```
<wn20schema:NounSynset rdf:about="wn20instances:synset-entity-noun-1" rdfs:label="entity">
            <wn20schema:synsetId>100001740</wn20schema:synsetId>
</wn20schema:NounSynset>

<rdf:Description rdf:about="wn20schema:Synset">
     <rdfs:subClassOf rdf:resource="lw:SemanticIndex"/>
</rdf:Description>

<someOntology:Noun>
            <olw:semanticDescriptor rdf:resource="wn20instances:synset-entity-noun-1">
</someOntology:Noun>
```

Fig. 7 an example of resource wrapping: binding WordNet-RDF synsets to a class concept

owl:semanticDescriptor property. With this extensible pattern, the LW+OLW offer reusable vocabularies for describing linguistic resources which drive the behavior of software applications serving the same task, while specific extensions (both in terms of ontologies and software components) can be added to describe specific lexical and semantic objects from new resources, without requiring modifications to the core vocabulary nor to the original application

Compatibility with existing (proposed) models As previously mentioned, several formats exists or have been proposed for integrating ontological content with linguistic information

While we did not intend to propose a new one, we tried to obtain cross-compatibility with available standards and proposed models, by gearing our software library with a OntoLinguisticModel interface, consisting of a series of enrichment/retrieval operations defined upon abstract "slots" for representing linguistic information. These slots can be then implemented according to a specific onto-linguistic representation model, by specifying the properties and concepts used to map/integrate linguistic information with ontological one.

Obviously, it is impossible to foresee in advance all the characteristics of each model/interface-implementation which could be integrated in the future, thus we provided a specific *project/decode* feature for projecting the linguistic information extracted from linguistic resources according to the LW ontology, towards the (possibly more fine-grained) adopted ontolinguistic model. For evaluative (see next section) and comparative purpose in general, we demand to each specific implementation the specifications of equivalence between the locally defined linguistic objects.

Implementations of OntoLinguisticModel have been developed for the traditionally adopted RDFS annotation properties (rdfs:label and rdfs:comment), for the base SKOS vocabulary (by extending the above with skos:prefLabel and skos:altLabel), for SKOS +SKOS-Mapping[12] vocabularies (thus including skos:broader/skos:narrower and skos:related, to map ontology concepts with in-

[12] http://www.w3.org/2004/02/skos/mapping/spec/

stances of lw:SemanticIndex from the LW ontology) and, finally, for the LingInfo model, by wrapping the linginfo:linginfo property and linginfo:LingInfo class. The above integration model satisfied our fifth requirement, while the resolution of the sixth one is part of the discussion presented in the next section.

The evaluation framework

The newly developed OLW Library provides a framework for evaluating the quality of algorithms for Linguistic Enrichment of ontologies with respect to previously defined reference standards, by using standard *precision&recall* metrics [36].

The OLW library can accept pairs of linguistic enrichment documents (that is: ontologies with integrated linguistic content), where one is the Oracle and the other one is the result to be tested, providing that the following extensions are included in the library and properly configured:

– *Enrichment Model* and related software extension
– *Resource(s) description* (and their wrapper implementation) used for enrichment
– *Match Specification and Evaluation (MSE)* extension, if different enrichment entries differ from simple links between ontological and linguistic objects

With the ones above, the library is able to seek the enrichment properties (at least, those which need to be considered) in the ontology documents (first extension) and to properly identify the elements used for the enrichment (second extension). The third one is an extension needed for those cases where an algorithm produces any kind of probabilistic/quantitative result, so that the enrichment links in the tested document cannot be evaluated just in terms of correct/wrong matches versus those in the Oracle. Inter-annotator agreement can as well be measured against two enrichment documents compiled by human annotators, with no further requirement apart from above.

OntoLing

OntoLing [24] is, in its last incarnation (OntoLing 4.0), a generic architecture for extending Ontology Development tools with functionalities for enriching ontological knowledge with linguistic content. The architecture of OntoLing will be implementable through realization and composition of different components:

By first a core component exposing the following characteristics:

– can be *interfaced* with the Linguistic Watermark software library to access linguistic resources, and with different enrichment algorithms and models (see Linguistic Watermark description in previous section) for enriching the content of ontologies with information gathered from loaded resources

- *knowledge* of the main functionalities and user interfaces characteristics exposed by common ontology development tools and of the extensions which should be brought by the OntoLing framework
- *high portability*: the core component has a module called `UIReasoner` (*User Interface Reasoner*) which is able to describe – according to an abstract representation formalism – the way the UI should appear to the user (which depends on the characteristics of the loaded linguistic resource) as well as describe actions and events which happen inside it. This way, a concrete implementation of this component could be easily ported and reused across different development environments. Moreover, if the abstraction layer is sufficiently expressive, changes to the core component should not require (heavy) modifications on each of its multiple implementations available for current ontology development tools

Second, trivially: the Linguistic Watermark library

Third: a set of linguistic resources (and wrappers for them, compatible with Linguistic Watermark API)

Fourth: an ontology development tool

Fifth (and last), an adapter between OntoLing core component and the ontology development tool, which directly wraps its API and provides concrete implementations for OntoLing User Interface extensions.

OntoLing Core Application

The core component of the architecture is responsible for interpreting the Watermark of linguistic resources and for exposing those functionalities which suit to their profile. Moreover, the behavior of the whole application is dependent on the nature of the loaded resource and is thus defined at run-time. Several methods for querying LRs and for exposing results have been encapsulated into objects inside a dedicated library of behaviors: when a given LR is loaded, the core module parses its Linguistic Watermark and assigns specific method-objects to each GUI event.

With such an approach, the user is provided with a uniform view over diverse and heterogeneous linguistic resources, as they are described in the Linguistic Watermark ontology, and easily learns how to interact with them (thus familiarizing with their peculiarities) by following a policy which is managed by the system.

For example, with a flat resource, a search on a given term will immediately result in a list of (potential) synonyms inside a dedicated box in the GUI; instead, with a conceptualized resource, a list of word senses will appear in a results table at first, then it will be browsed to access synonymical expressions related to the

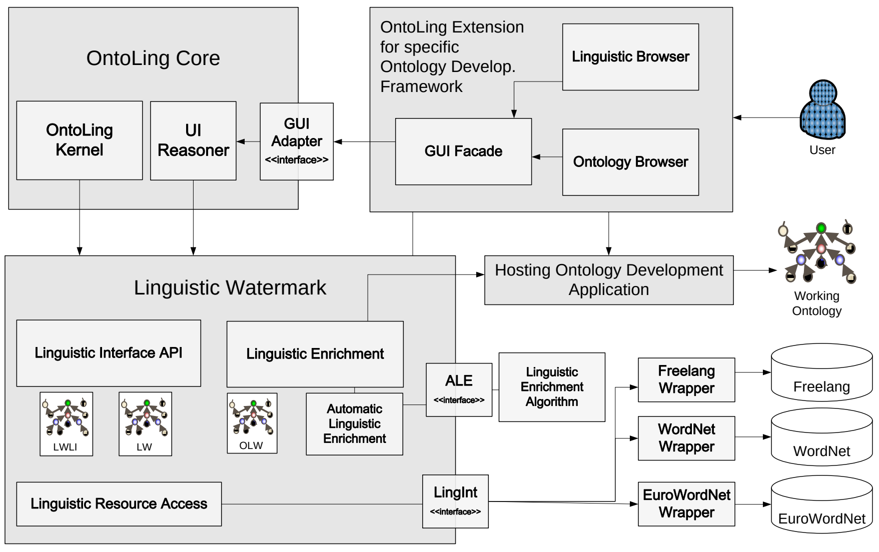


Fig. 8 OntoLing Framework Architecture

selected sense. Analogous adaptive approaches have been followed for many other aspects of the Linguistic Watermark (mono or bidirectional Bilingual Translators, presence of glosses, Taxonomical structures and so on…) sometimes exploding with combinatorial growth.

Future development of Ontoling will go in the direction of considering supervised techniques for automatic ontology enrichment; selecting and modeling the right strategies for the adopted LRs is another task the core module is in charge for.

OntoLing User Interface

Once activated, the plug-in displays two main panels, the Linguistic Browser on the left side, and the Ontology Panel on the right side (see Fig. 9).
The Linguistic Browser is responsible for letting the user explore the loaded linguistic resource. Fields and tables for searching the LR and for viewing the results, according to the modalities decided by the core component, are made available. The menu boxes on the left of the Linguistic Browser are filled at run time with the methods for exploring LR specific Lexical and Conceptual relations.
The Ontology Panel, on the right, offers a perspective over ontological data in the classic Protégé style. By right-clicking on a frame (class, slot or instance), the typical editing menu appears, with some further options provided by OntoLing to:

1. search the LR by using resources names as keys

2. change the name of the selected resource by using a term selected from the Linguistic Browser
3. add terms selected from the Linguistic Browser as additional labels for the selected resource
4. add glosses as a description (*rdfs:comment*) for the selected resource
5. add IDs of senses selected from the linguistic browser as additional labels for the resources
6. create a new resource with a term selected from the Linguistic Browser as resource name
7. only in class and property browser: if the LR is a `TaxonomicalLR`, explore hyponyms (up to a chosen level) of the concept selected on the Linguistic Browser and reproduce the tree on the resource browser, starting from the selected resource, if available

These functionalities allow not only for linguistic enrichment of ontologies, but can be helpful for Ontologists and Knowledge Engineers in creating new ontologies or in improving/modifying existing ones.

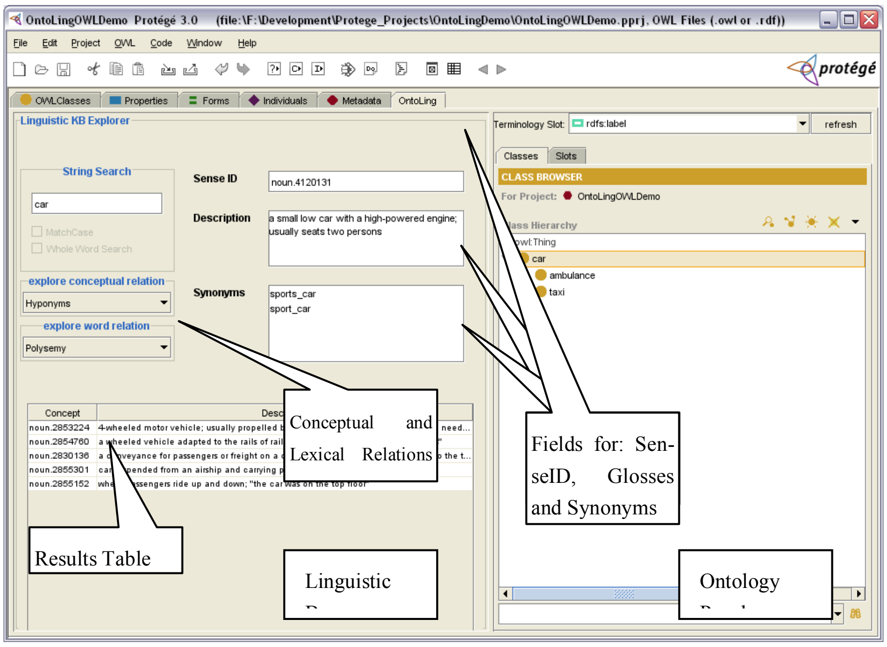
In OntoLing-Protégé, how terms and glosses are added to the description of ontologies concepts, depends on the ontology model which is being adopted and is explained in detail in the following section.

Using OntoLing with Protégé and Protégé OWL

The first version of OntoLing was developed expressly as an extension for the Protégé Ontology Editor. All of the work which radically modified its backing architecture has not changed much the way OntoLing appears to its users. In this section we describe choices and history of this first extension.

When a frame-based approach was first adopted in Protégé as a knowledge model for representing ontologies and knowledge bases, no explicit effort was dedicated to the representation of possible alternate labels (synonyms) for concepts neither to support the idea of multilingualism in Ontologies. Frame names were almost as equivalent as IDs, and people were only encouraged, as it is common practice in computer programming when addressing variable names, to adopt "meaningful and expressive names" to denote these IDs. The Protégé model was indeed quite strong and expressive, so that every ontology developer could deal with his linguistic needs at a meta-ontological level and find the right place for them, though no official agreement was yet established.

Later on, with the advent of OWL as a KR standard for the Semantic Web, and with the official release of the Protégé OWL plug-in [19], things started to converge towards a minimal agreement for the use of language inside ontologies. When we first started working on OntoLing, the OWL plug-in had just been released, and the majority of users continued to use Protégé in the usual way, so we had to find a solution that was quite easy (for the user) to make do with this lack in the standard Protégé model.


OntoLingOWLDemo Protégé 3.0 (file:\F:\Development\Protege_Projects\OntoLingDemo\OntoLingOWLDemo.pprj, OWL Files (.owl or .rdf))
File Edit Project OWL Code Window Help
protégé
OWLClasses Properties Forms Individuals Metadata OntoLing
Linguistic KB Explorer
String Search
MatchCase
Whole Word Search
explore conceptual relation
explore word relation
Terminology Slot:
refresh
Classes Slots
CLASS BROWSER
For Project: OntoLingOWLDemo
Class Hierarchy
owl:Thing
car
ambulance
taxi
Conceptual and Lexical Relations
Fields for: SenseID, Glosses and Synonyms
Results Table
Linguistic
Ontology

String Search: car

explore conceptual relation: Hyponyms

explore word relation: Polysemy

Sense ID: noun.4120131

Description: a small low car with a high-powered engine; usually seats two persons

Synonyms: sports_car, sport_car

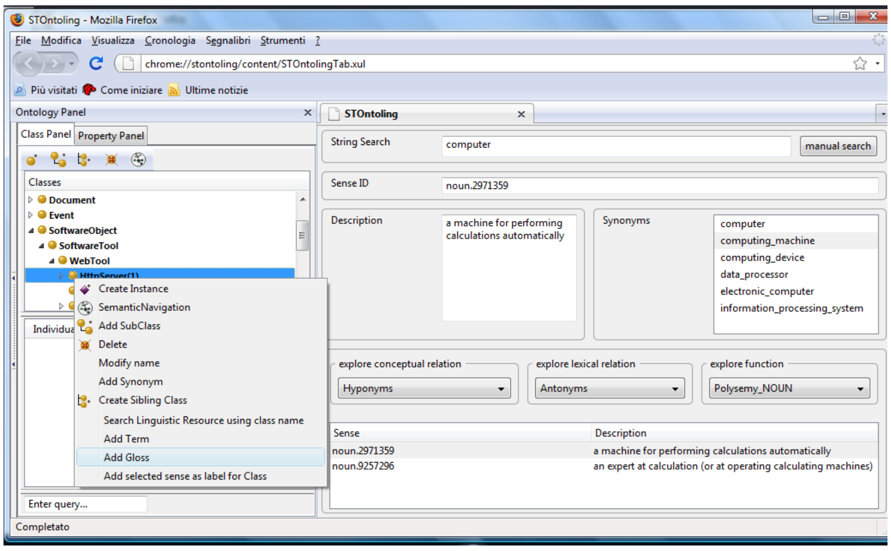
Terminology Slot: rdfs:label

Concept	Desc
noun.2853224	4-wheeled motor vehicle; usually propelled b… need...
noun.2854760	a wheeled vehicle adapted to the rails of rail… "
noun.2830136	a conveyance for passengers or freight on a … the t...
noun.2855301	car suspended from an airship and carrying p…
noun.2855152	whe… passengers ride up and down; "the car was on the top floor"

Fig. 9 OntoLing Screeshot (Protégé version)

To this end, we defined the notion of terminological slot, as a slot which is elected by the user to contain different linguistic expressions for concepts. Any string-typed slot with cardinality set to multiple, can potentially be selected as a terminological slot, and, for easiness of use, OntoLing prompts the user only with this class of slots. This way, to use Ontoling with standard Protégé, a user only needs to define a proper metaclass and metaslot, containing the elected terminological slot; naturally, the same slot can be dedicated to instances at class level. Multilingual ontologies can also be supported by creating different slots and selecting each of them as terminological slots during separate sessions of Linguistic Enrichment, with diverse LRs dedicated to the different chosen languages. Concerning glosses, these can be added to the common "documentation" slot which is part of every frame by default.

Conversely, Linguistic Enrichment of OWL Ontologies follows a more predictable path, thanks to OWL's language dedicated Annotation Properties, such as *rdfs:label* and *owl:comment*. When Ontoling recognizes a loaded ontology as expressed in the OWL language, the terminological slot is set by default (though modifiable) to *rdfs:label*. In this case the *xml:lang* attribute of the label property is automatically filled with the language declared by the Linguistic Interface.


STOntoling - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti ?
chrome://stontoling/content/STOntolingTab.xul
Più visitati Come iniziare Ultime notizie
Ontology Panel
Class Panel Property Panel
Classes
Document
Event
SoftwareObject
SoftwareTool
WebTool
HttpServer(1)
Individua
Create Instance
SemanticNavigation
Add SubClass
Delete
Modify name
Add Synonym
Create Sibling Class
Search Linguistic Resource using class name
Add Term
Add Gloss
Add selected sense as label for Class
Enter query...
Completato
STOntoling
manual search

String Search: computer

Sense ID: noun.2971359

Description: a machine for performing calculations automatically

Synonyms: computer, computing_machine, computing_device, data_processor, electronic_computer, information_processing_system

explore conceptual relation: Hyponyms

explore lexical relation: Antonyms

explore function: Polysemy_NOUN

Sense	Description
noun.2971359	a machine for performing calculations automatically
noun.9257296	an expert at calculation (or at operating calculating machines)

Fig. 10 Ontoling in Semantic Turkey

OntoLing as an extension to Semantic Turkey

While recent changes to the architecture of OntoLing have not produced (they were not meant do that) sensible impact on interaction with the user, they surely allowed for more flexible development of new functionalities as well as fast-to-produce porting over different applications.

Our experience in porting the new version of OntoLing on the Semantic Turkey architecture revealed that we were able to keep down realization costs by more than two thirds of the whole development effort, since we had to:

- realize its user interface
- realize a ST service extension which includes the OntoLing Core component
- serialize abstract UI actions produced by OntoLing Core component as XML messages sent from ST server
- develop handlers for UI actions sent by the server, realizing necessary handling of requested actions over the Firefox UI of OntoLing

Of the above, only the first part required sensible effort, due to the completely different UI technology adopted by Firefox with respect to traditional Java Swing adopted by Protégé, thus preventing even minimal reuse of code. On the other hand, this aspect is a necessary step for any porting attempt, while we totally beneficiated of the complex UI management (depending on the ling. watermark of the loaded resource) which has been completely demanded to the included core component. Also, apart from the effort, this approach is not requiring deep knowledge of the framework nor of its inner logic, since most relevant and critical aspects are concentrated inside the core component and need not to be re-implemented: this lowers requirements in terms of development personnel and eases even more the porting process.

Though we focused in obtaining a portable and completely replicable multilingual extension for Ontology Development systems, we plan to obtain the best from the combination of OntoLing with the possibilities of our ontology development environment, deriving from its inherent connection with the Web and, as a consequence of that, with the many different information sources (Wikipedia, online dictionaries etc…) which can be explored in such an open environment.

Conclusion

In this paper we presented a collection of software libraries, tools and ontologies for supporting multilingual development of Semantic Web ontologies. The presented work is the result of different research efforts which we tried to converge towards a common goal, though this can be seen just as "end of the beginning" of this exploration.

We expect that our work, through its tangible proofs-of-concepts, may give a contribution or at least motivate the standardization of models, methodologies and tools for the effective integration of ontologies and linguistic resources: something which is much felt as a need for the future of Web 3.0 – which on the one side foresees a web of data made accessible by machines, and on the other one expects this data to be self-explanatory and human-comprehensible on a multicultural and multilingual ground – but which is until now demanded to specific efforts and arbitrary solutions.

References

[1] Paolo Atzeni et al., "Ontology-based question answering in a federation of university sites: the MOSES case study," in *9th International Conference on Applications of Natural Language to Information Systems (NLDB'04)*, Manchester (United Kingdom), 2004, Month: June.

[2] Roberto Basili, Michele Vindigni, and Fabio Massimo Zanzotto, "Integrating Ontological and Linguistic Knowledge for Conceptual Information Extraction," in *IEEE/WIC International Conference on Web Intelligence*, Washington, DC, USA, 2003.

[3] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," in *The Semantic Web - ISWC 2002: First International Semantic Web Conference*, Sardinia, Italy, 2002, pp. 54-68, June 9-12.

[4] Paul Buitelaar et al., "LingInfo: Design and Applications of a Model for the Integration of Linguistic Information in Ontologies," in *OntoLex06*, Genoa, Italy, 2006.

[5] Nicoletta Calzolari, John McNaught, and Antonio Zampolli, "EAGLES Final Report:

EAGLES Editors Introduction," Pisa, Italy, EAG-EB-EI, 1996.

[6] A. Cappelli, E. Giovannetti, and P. Michelassi, "Ontological Knowledge and Language in Modelling Classical Architectonic Structures," in *Ontology and Lexical Resources - OntoLex 2004*, Lisboa, Portugal, 2004.

[7] Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks, "User-system cooperation in document annotation based on information extraction.," in *13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*, 2002.

[8] R. A. Cole et al., Eds., *Survey of the State of the Art in Human Language Technology*. Cambridge, UK: Cambridge University Press, 1997.

[9] Martin Dzbor, John Domingue, and Enrico Motta, "Magpie: Towards a Semantic Web Browser," in *2nd International Semantic Web Conference (ISWC03)*, Florida, USA, 2003.

[10] Christiane Fellbaum, *WordNet: An Electronic Lexical Database*. Cambridge, MA: WordNet Pointers, MIT Press, 1998.

[11] Gil Francopoulo et al., "Lexical Markup Framework (LMF)," in *LREC2006*, Genoa, Italy, 2006.

[12] J.J. Garrett. (2005, February) Ajax: A New Approach to Web Applications. [Online]. http://www.adaptivepath.com/publications/essays/archives/000385.php

[13] John Gennari et al., "The evolution of Protégé-2000: An environment for knowledge-based systems development, ," *International Journal of Human-Computer Studies*, vol. 58, no. 1, p. 89–123, 2003.

[14] Donato Griesi, Maria Teresa Pazienza, and Armando Stellato, "Semantic Turkey - a Semantic Bookmarking tool (System Description)," in *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007, Proceedings. Lecture Notes in Computer Science*, vol. 4519, 2007, pp. 779-788, Innsbruck, Austria, June 3-7.

[15] C. Huang, "Sinica BOW: Integrating bilingual WordNet and SUMO Ontology. Ontology and Lexical Resources - OntoLex 2004. Lisboa, Portugal,".

[16] D. Huynh, Stefano Mazzocchi, and D.R. Karger, "Piggy Bank: Experience the Semantic Web Inside Your Web Browser," in *Fourth International Semantic Web Conference (ISWC05)*, Galway, Ireland, November, 2005, pp. 413-430.

[17] José Kahan and Marja-Ritta Koivunen, "Annotea: an open RDF infrastructure for shared Web annotations," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*, Hong Kong, Hong Kong, 2001, pp. 623-632.

[18] A. Kiryakov, Damyan Ognyanov, and D. Manov, "OWLIM – a Pragmatic Semantic Repository for OWL," in *Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE 2005*, New York City, USA, 2005, 20 November.

[19] Holger Knublauch, Ray W. Fergerson, Natasha Friedman Noy, and Mark, A. Musen, "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications," in *Third International Semantic Web Conference - ISWC 2004*, Hiroshima, Japan, 2004.

[20] Brian McBride, "Jena: Implementing the RDF Model and Syntax Specification," in *Semantic Web Workshop, WWW2001*, 2001.

[21] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller, *Introduction to WordNet: An On-line Lexical Database*., 1993.

[22] Alessandro Oltramari and Armando Stellato, "Enriching Ontologies with Linguistic Content: an Evaluation Framework," in *The role of ontolex resources in building the infrastructure of Web 3.0: vision and practice (OntoLex 2008)*, Marrakech, Morocco,

2008, May, 31.

[23] (2005) OSGi RFC0112. [Online]. http://www2.osgi.org/Download/File?url=/download/rfc-0112_BundleRepository.pdf

[24] Maria Teresa Pazienza and Armando Stellato, "An open and scalable framework for enriching ontologies with natural language content," in *Advances in Applied Artificial Intelligence, 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006, Annecy, France, June 27-30, 2006, Proceedings. Lecture Notes in Computer Science*, vol. 4031, Annecy, France, 2006, pp. 990-999, June 27-30, special session on Ontology & Text.

[25] Maria Teresa Pazienza and Armando Stellato, "Exploiting Linguistic Resources for building linguistically motivated ontologies in the Semantic Web," in *Second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006)*, Genoa, Italy, 2006.

[26] Maria Teresa Pazienza and Armando Stellato, "Linguistic Enrichment of Ontologies: a methodological framework," in *Second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006)*, Genoa, Italy, 2006.

[27] Maria Teresa Pazienza, Armando Stellato, Lina Enriksen, Patrizia Paggio, and Fabio Massimo Zanzotto, "Ontology Mapping to support ontology-based question answering," in *Second MEANING workshop*, Trento, Italy, 2005, Month: February.

[BO] Maria Teresa Pazienza, Armando Stellato, and Andrea Turbati, "Linguistic Watermark 3.0: an RDF framework and a software library for bridging language and ontologies in the Semantic Web," in *Semantic Web Applications and Perspectives, 5th Italian Semantic Web Workshop (SWAP2008)*, FAO-UN, Rome, Italy, 2008, 15-17 December.

[29] Maria Teresa Pazienza, Armando Stellato, Michele Vindigni, Alexandros Valarakos, and Vangelis Karkaletsis, "Ontology integration in a multilingual e-retail system," in *HCI International 2003*, Crete, Greece, 2003.

[30] H. Peter, H Sack, and C. Beckstein, "SMARTINDEXER – Amalgamating Ontologies and Lexical Resources for Document Indexing," in *Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006)*, Genoa, Italy, 2006.

[31] A. Philpot, Edward Hovy, and Patrick Pantel, "The Omega Ontology," in *Ontology and Lexical Resources (OntoLex2005)*, Jeju Island, South Korea, 2005.

[32] Laurent Prevot, S. Borgo, and Alessandro Oltramari, "Interfacing Ontologies and Lexical Resources," in *OntoLex2005 - Ontologies and Lexical Resources*, Jeju Island, South Korea, 2005.

[33] Adriana Roventini et al., "ItalWordNet: A Large Semantic Database for the Automatic Treatment of the Italian Language," in *First International WordNet Conference*, Mysore, India, January 2002.

[34] J. Scheffczyk, C. F. Baker, and S. Narayanan, "Ontology-based Reasoning about Lexical Resources," in *Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006)*, Genoa, Italy, 2006.

[35] S. Stamou et al., "BALKANET: A Multilingual Semantic Network for the Balkan Languages.," in *First International Wordnet Conference*, Mysore, India, January 2002, pp. 12-14.

[36] C. J. Van Rijsbergen, *Information Retrieval*. London, United Kingdom: Butterworths, 1975.

[37] Piek Vossen, *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Dordrecht: Kluwer Academic Publishers, 1998.

[38] W3C. [Online]. http://www.w3.org/TR/owl-features/