Computer-aided Ontology Development: an integrated environment

Manuel Fiorelli, Maria Teresa Pazienza, Steve Petruzza, Armando Stellato, Andrea Turbati

ART Research Group, Dept. of Computer Science,
Systems and Production (DISP) University of Rome, Tor Vergata
Via del Politecnico 1, 00133 Rome, Italy
{pazienza, stellato, turbati}@info.uniroma2.it
{manuel.fiorelli, steve.petruzza}@gmail.com

Abstract

In this paper we introduce CODA (Computer-aided Ontology Development Architecture), an Architecture and a Framework for semi-automatic development of ontologies through analysis of heterogeneous information sources. We have been motivated in its design by observing that several fields of research provided interesting contributions towards the objective of augmenting/enriching ontology content, but that they lack a common perspective and a systematic approach.
While in the context of Natural Language Processing specific architectures and frameworks have been defined, time is not yet completely mature for systems able to reuse extracted information for ontology enrichment purposes: several examples do exist, though they do not comply with any model nor architecture. Objective of CODA is to acknowledge and improve existing frameworks to cover these gaps, by providing: a conceptual systematization of data extracted from unstructured information to enrich ontology content, an architecture defining the components which take part in such scenario, and a framework supporting all of the above.
This paper provides an overview of the whole picture, and introduces UIMAST, an extension for the Knowledge Management and Acquisition Platform *Semantic Turkey*, that implements CODA principles by allowing reuse of components developed inside UIMA framework to drive semi-automatic Acquisition of Knowledge from Web Content.

1. Introduction

A number of tasks focused on ontology development as well as on augmentation or refinement of their content through reuse of external information has been defined in the last decade. The nature of these tasks is manifold: from the automation of ontology development processes to their facilitation through innovative and effective solutions for human-computer interaction. In some cases their assessment has produced a plethora of (often contrasting) methodologies and approaches (as in the case of *ontology and lexicon integration* (Buitelaar, et al., 2006; Cimiano, Haase, Herold, Mantel, & Buitelaar, 2007; Pazienza & Stellato, 2006; Pazienza & Stellato, Linguistic Enrichment of Ontologies: a methodological framework, 2006)); in other ones, such as *ontology learning*, it has lead to founding entire new branches of research (Cimiano, 2006)
The "external information" we are interested in, mostly refers to diverse forms of "narrative information sources", such as text documents (or other kind of media, such as audio and video) or to more structured knowledge content, like the one provided by machine readable linguistic resources. These latter comprise lexical resources (e.g. rich lexical databases such as WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1993), bilingual translation dictionaries or domain thesauri), text corpora (from pure domain-oriented text collections to annotated corpora of documents), or other kind of structured or semi-structured information sources, such as frame-based resources (Baker, Fillmore, & Lowe, 1998; Shi & Mihalcea, 2005).
With the intent of providing a definition covering all of the previously cited tasks and addressing the interaction they have with the above resources, we coined the expression COD (Computer-aided Ontology Development), with this acronym covering all processes for enriching ontology content through exploitation of external resources, by using (semi)automatic approaches.
In this paper, we lay the basis for an architecture (CODA: COD Architecture), supporting Computer-aided Ontology Development, then introduce UIMAST, an extension for the Knowledge Management and Acquisition Platform *Semantic Turkey*, implementing CODA principles by allowing reuse of components developed inside the UIMA framework to drive semi-automatic Acquisition of Knowledge from Web Content.

2. State-of-the-art and Motivation

Motivations and ideas for supporting fulfillment of the above tasks' objectives, have been often supported through proof-of-concept systems, tools and in some cases open platforms (Cimiano & Völker, 2005) developed inside the research community, laying the path and showing the way for future industrial follow-up.
Until now basic architectural definitions and interaction modalities have been defined in detail fulfilling industry-standard level for processes such as:

- *ontology development* with most recent ontology development tools following the path laid by Protégé (Gennari, et al., 2003)
- *text analysis* starting from the TIPSTER architecture (Harman, 1992), its most notable implementation GATE (Cunningham, Maynard, Bontcheva, & Tablan, 2002) and the recently approved OASIS standard UIMA (Ferrucci & Lally, 2004).

On the contrary a comprehensive study and synthesis of an architecture for supporting ontology development driven by knowledge acquired from external resources, has not been formalized until now.
What lacks in all current approaches is an overall perspective on the task and a proposal for an architecture providing instruments for supporting the entire flow of

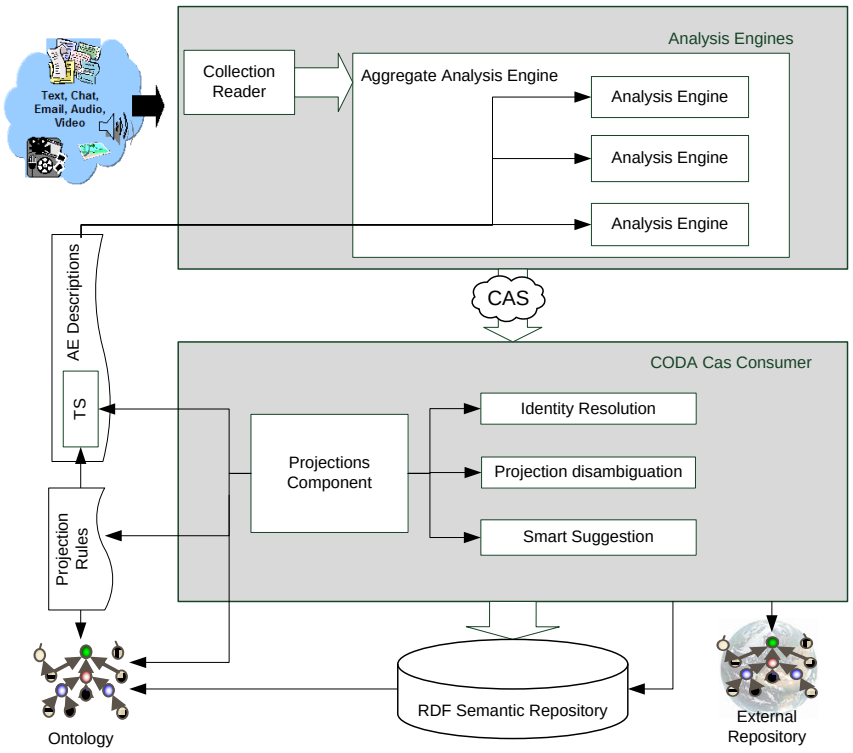

Figure 1. CODA Architecture (overview of components related to tasks 1 and 2)

information (from acquisition of knowledge from external resources to its exploitation) to enrich and augment ontology content. Just scoping to ontology learning, OntoLearn (Velardi, Navigli, Cucchiarelli, & Neri, 2005) provides a methodology, algorithms and a system for performing different ontology learning tasks, OntoLT (Buitelaar, Olejnik, & Sintek, 2004) provides a ready-to-use Protégé plugin for adding new ontology resources extracted from text, while the sole Text2Onto (Cimiano & Völker, 2005) embodies a first attempt to realize an open architecture for management of ontology learning processes.

If we consider ontology-lexicon integration, previous studies dealt with how to represent this integrated information (Peters, Montiel-Ponsoda, Aguado de Cea, & Gómez-Pérez, 2007; Buitelaar, et al., 2006; Cimiano, Haase, Herold, Mantel, & Buitelaar, 2007), other have shown useful applications exploiting onto-lexical resources (Basili, Vindigni, & Zanzotto, 2003; Peter, Sack, & Beckstein, 2006) though only few works (Pazienza, Stellato, & Turbati, 2008) dealt with comprehensive framework for classifying, supporting, testing and evaluating processes for integration of content from lexical resources with ontological knowledge.

3. Objectives

Considering these expectations, we worked with the objective of acknowledging and improving existing frameworks for Unstructured Information Management, thus providing:

- a *conceptual systematization* of the tasks covering reuse of data extracted from unstructured information to improve ontology content
- an *architecture* defining the components which take part in such a scenario
- a *framework* supporting all of the above through standard implementations

We provide here requirements and objectives which characterize COD tasks, the COD Architecture, and a CODA Framework

3.1. COD Tasks

Given the definition of COD provided at the beginning, we sketch here major related tasks:

1. **(Traditional) Ontology Learning tasks**, devoted to augmentation of ontology content through discovery of new resources and axioms. They include discovery of new concepts, concept inheritance relations, concept instantiation, properties/relations, domain and range restrictions, mereological relations or equivalence relations etc…
2. **Population of ontologies with new data:** a rib of the above, this focuses on the extraction of new ground data for a given (ontology) model (or even for specific concepts belonging to it)
3. **Linguistic enrichment of ontologies**: enrichment of ontological content with linguistic information coming from external resources (eg. text, linguistic resources etc…)

3.2. CODA Architecture

COD Architecture (CODA, from now on) defines the components (together with their interaction) which are needed to support tasks above. This architecture builds on top of existing standard for Unstructured Information Management UIMA (UIM Architecture) (tasks 1&2) and, for task 3, on the Linguistic Watermark (Pazienza, Stellato, & Turbati, 2008) suite of ontology vocabularies and software libraries for describing linguistic resources

and the linguistic aspects of ontologies. Figure 1 depicts the part of the architecture supporting tasks 1 and 2. Tiny arrows represent the *use/depends on* relationship, so that the Semantic Repository owl:imports the reference ontologies, the projection component *invokes* services from the other three components in the CODA CAS Consumer as well as *is driven* by the projection document and TS and reference ontology. Large arrows represent instead the flow of information.

While UIMA already foresees the presence of CAS Consumers[1] for projecting collected data over any kind of repository (ontologies, databases, indices etc…), COD Architecture expands this concept by providing ground anchors for engineering ontology enrichment tasks, decoupling the several processing steps which characterize development and evolution of ontologies. This is our main original contribution to the framework. Here follows a description of the presented components.

Projection Component

This is the main component which realizes the projection of information extracted through traditional UIM components (i.e. UIMA *Annotations*).

The Unstructured Information Management (UIM) standard foresees data structures stored in a CAS (Common Analysis System). CAS data comprises a *type system*, i.e. a description – represented through feature structures (Carpenter, 1992) – of the kind of entities that can be manipulated in the CAS, and the *data* (modeled after the above type system) which is produced over processed information stream.

This component thus takes as input:

- A *Type System* (TS)
- A reference *ontology* (we assume the ontology to be written in the RDFS or OWL W3C standard)
- A projection document containing projection rules from the TS to the ontology
- A CAS containing annotation data represented according to the above TS

and uses all the above in order to project UIMA annotations as data over a given Ontology Repository.

The language for defining projections allows for:

- *Projecting CAS feature structures (FS) as instances of a given class*. FeaturePaths can be used to project arbitrary feature values as instance names
- *Projecting FSs as values of datatype properties*. Note that this requires ontology instances to be elected as subjects for each occurrence of this property annotation. The domain class which will be used to look for instance can be specified in the projection rule. Note that, by default, the domain of the property is inherited from the ontology, though it may be further restricted for the specific rule. So, for example, if property date has owl:Thing as its domain (i.e. no domain restriction), the outcome of a specific Analysis Engine, which is able to capture dates for conference events (or which is being used in a given setting for this purpose), can be restricted in the projection rule to automatically search for instances of the restricted domain. The use that is made of the above information is partially demanded to the application context, in order to properly select the right instances to be associated to the valued property.
- *Projecting complex FSs as custom graph patterns*. Some TS provide complex extraction patters which contains much more than plain text annotations; they possibly provide *facts* with explicit semantics which only need to be properly imported into the ontology. In this case, custom RDF graph patterns can be defined to create new complex relations inside the ontology. GRAPH Patterns are sets of RDF triples, in this case enriched by the presence of bindings to TS elements (again, in the form of FeaturePaths). When this projection is being applied, the feature path bindings are resolved and the ground pattern is used in a SPARQL CONSTRUCT query to generate new RDF triples in the Semantic Repository).

The Projection Component can be used in different scenarios (from massively automated ontology learning/population scenarios, to support in human centered processes for ontology modeling/data entry) and its projection processes can be supported by the following components.

Identity Resolution Component

Whenever an annotation is projected towards ontology data, the services of this component are invoked to identify potential matches between the annotated info which is being reified into the semantic repository, and previously recognized resources already present inside it.

If the Identity Resolution (IR) component discovers a match, then the new entry is merged into the existing one; that is, any new data is added to the resource description while duplicated information (probably the one which helped in finding the match) is discarded.

The IR component may look up on the same repository which is being fed by CODA though also external repositories of LOD (linked open data) can be accessed. Eventually, entity naming resolution provided by external services – such as the Entity Naming System (ENS) OKKAM (Bouquet, Stoermer, & Bazzanella, 2008) – may be combined with internal lookup on the local repository.

Input for this component are:

- External RDF repositories (providing at least indexed approximate search over their resources)
- Entity Naming Systems access methods
- Other parameters needed by specific implementation of the component

Projection Disambiguation Component(s)

These components may be invoked by the Projection Component to disambiguate between different possible projections. Projection documents may in fact describe more than one projection rule which can be applied to given types in the TS. These components are thus, by definition, associated to entries in Projection Documents and are automatically invoked when more than a rule is matched on the incoming CAS data.

[1] UIMA terminology is widely adopted along the paper: though some explanations are provided here, we refer non-proficient readers to the *UIMA Glossary* inside the *UIMA Overview & SDK Setup* document, which is available at: http://incubator.apache.org/uima/documentation.html

This component has access by default to the current Semantic Repository (and any reference ontology for the Projection rules), to obtain a picture of the ongoing process which can contribute to the disambiguation process.

Smart Suggestion Component(s)

These components help in proposing suggestions on how to fill empty slots in projection rules (such as subject instances in datatype property projections or free variables in complex FS to graph-pattern projections). As for Disambiguation Components, these components can be written for specific Projection Documents and associated to the rules described inside them, as supporting computational objects.

3.3. CODA Framework Objectives

CODA Framework is an effort to facilitate development of systems implementing the COD Architecture, by providing a core platform and highly reusable components for realization of COD tasks.
Main objectives of this architectural framework are:

1. Orchestration of all processes supporting COD tasks
2. Interface-driven development of COD components
3. Maximizing reuse of components and code
4. Tight integration with available environments, such as UIMA for management of unstructured information from external resources (e.g. text documents) and Linguistic Watermark (Pazienza, Stellato, & Turbati, 2008) for management of linguistic resources
5. Minimizing required LOCs (lines of code) and effort for specific COD component development, by providing high level languages for matching/mapping components I/O specifications instead of developing software adapters for their interconnection
6. Providing standard implementations for components realizing typical support steps for COD tasks, such as management of corpora, user interaction, validation, evaluation, production of reference data (oracles, gold standards) for evaluation, identity discoverers etc…

In the specific, with respect to components described in section 3.2, CODA Framework will provide the main Projection Component (and its associated projection language), a basic implementation of an Identity Resolution Component, and all the required business logic to fulfill COD tasks through orchestration of COD components.

4. Possible application scenarios

Willing to fulfill these objectives, we envision several application scenarios for CODA. We provide here a description of a few of them.

Fast Integration of existing UIMA components for ontology population

By providing projections from CAS type systems to ontology vocabularies, one could easily embed standard UIMA AEs (Analysis Engines) and make them able to populate ontology concepts pointed by the projections, without requiring developing any new software component. These projections, which are part of objective 5 above, will be modeled through a dedicated language which will be part of the CODA framework. Moreover (objective 6 above), standard or customized identity discoverers will try to suggest potential matches between entities annotated by the AE and already existing resources in the target ontology, to keep identity of individual resources and add further description to them. In this scenario, given an ontology and a AE, only the projection from the CAS type system of the AE to the ontology is needed (and optionally, a customized identity discoverer). Everything else is assumed to be automatically embedded and coordinated by the framework.

Rapid prototyping of Ontology Learning Algorithms

This is the opposite situation of the scenario above. CODA, by reusing the same chaining of UIMA components, ontologies, CAS-to-Ontology projections, identity discoverers etc… , will provide:

- a preconfigured CAS type system (Ontology Learning CAS Type System) for representing information to be extracted under the scope of standard ontology learning tasks (i.e. the ones discussed in section 3.1)
- preconfigured projections from above CAS type system to learned ontology triples
- extended interface definitions for UIMA analysis engines dedicated to ontology learning tasks: available abstract adapter classes will implement the standard UIMA AnalysisComponent interface, interacting with the above Ontology Learning CAS type system and exposing specific interface methods for the different learning tasks

In this scenario, developers willing to rapidly deploy prototypes for new ontology learning algorithms, will be able to focus on algorithm implementation and benefit of the whole framework, disburdening them from corpora management and generation of ontology data. This level of abstraction far overtakes the *Modeling Primitive Library* of Text2Onto (i.e. a set of generic modeling primitives abstracting from specific ontology model adopted and being based on the assumption that the ontology exposes at least a traditional object oriented design, such as that of OKBC (Chaudhri, Farquhar, Fikes, Karp, & Rice, 1998)). In fact CODA does not even leaves to the developer the task of generating new ontology data, while just asks for specific objects to be associated and thus produced for given ontology learning tasks. For example, pairs of terms could be produced by taxonomy learners, which need then to be projected as IS-A or type-of relationships by the framework.

Plugging of algorithms for automatic linguistic enrichment of ontologies

In such a scenario, the user is interested in enriching ontologies with linguistic content originated from external lexical resources. The Linguistic Watermark library - which is already been used in tools for (multilingual) linguistic enrichment of ontologies (Pazienza, Stellato, & Turbati, 2010) and which constitutes a fundamental module of CODA - supports uniform access to heterogeneous resources wrapped upon a common model for lexical resource definition, allows for their integration

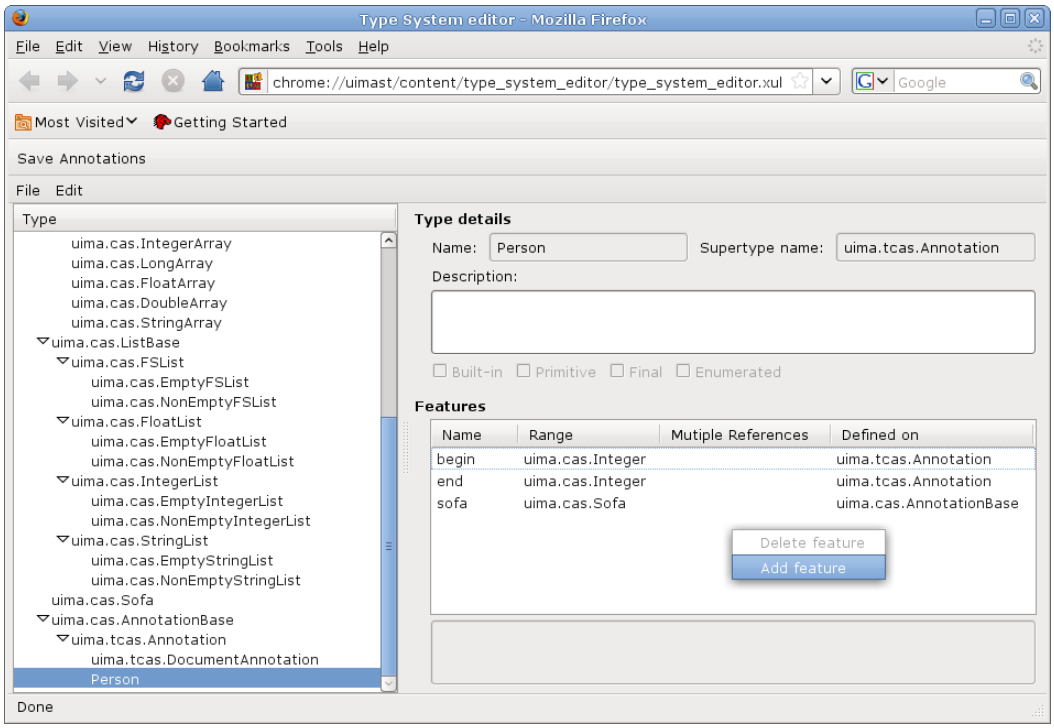

Figure 2. UIMAST Type System Editor

with ontologies and for evaluation of the acquired information. Once more, the objective is to relieve developers from technical details such as resource access, ontology interaction and update, by providing standard facilities associated to tasks for ontology-lexicon integration/enrichment, and thus leaving up to them the sole objective of implementing enrichment algorithms.

User Interaction for Knowledge Acquisition and Validation

User interaction is a fundamental aspect when dealing with decision-support systems. Prompting the user with compact and easy-to-analyze reports on the application of automated processes, and putting at his hands instruments for validating choices made by the system can dramatically improve the outcome of processes for knowledge acquisition as well as support supervised training of these same processes. CODA front-end tools should thus provide CODA specific applications supporting training of learning-based COD components, automatic acquisition of information from web pages visualized through the browser (or management of info previously extracted from entire corpora of documents) and editing of main CODA data structures (such as UIMA CAS types, projection documents and, obviously, ontologies). Interactive tools should support iterative refinement of massive production of ontology data as well as human-centered process for ontology development/evolution.

This last important environment is a further very relevant objective, and motivated us to define and develop UIMAST, an extension for Semantic Turkey (Griesi, Pazienza, & Stellato, 2007; Pazienza, Scarpato, Stellato, & Turbati, 2008), - a Semantic Web Knowledge Acquisition and Management platform[2] hosted on the Firefox Web Browser - to act as a CODA front-end for doing interactive knowledge acquisition from web pages.

5. UIMAST: A CODA-based tool supporting dynamic ontology population

The UIMAST Project[3] originated in late 2008, with the intent of realizing a system for bringing UIMA support to Semantic Turkey's functionalities for Knowledge Acquisition. The project has been organized around two main milestones:

- Supporting manual production of UIMA CAS compliant annotations
- Reuse UIMA annotators to automatically extract information from web pages and project them over the edited ontology

Milestone 1 has been reached in early 2009, with the first release of UIMAST. This release features:

1. *A UIMA Type System Editor* (figure 2 above), more intuitive to use than the Eclipse-based one bundled with UIMA, in that it provides a taxonomical view of edited Feature Structures, showing explicit and inherited attributes for each Type.
2. *Interactive UIMA annotator*: Semantic Annotations taken through Semantic Turkey can be projected as

[2] https://addons.mozilla.org/it/firefox/addon/8880 is the official page on Firefox add-ons site addressing Semantic Turkey extension, while http://semanticturkey.uniroma2.it/ provides an inside view about Semantic Turkey project, with updated downloads, user manuals, developers support and access to ST extensions.

[3] http://semanticturkey.uniroma2.it/extensions/uimast/. The idea for the project has been awarded with IBM UIMA Innovation Award http://download.boulder.ibm.com/ibmdl/pub/software/dw/university/innovation/2007_uima_recipients.pdf

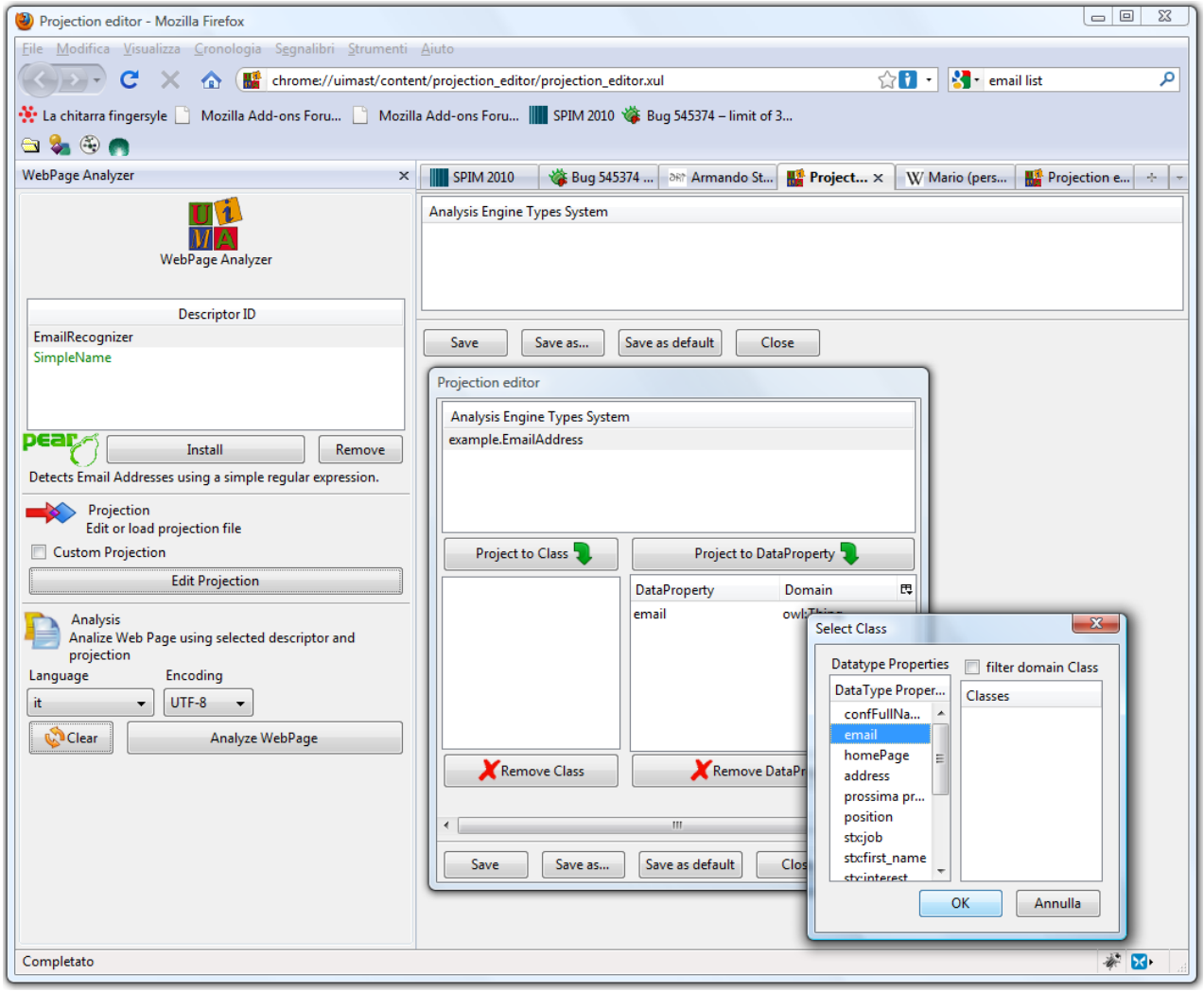

Figure 3. Editing Projections in UIMAST: from simple TS feature EmailAddress to ontology Datatype property email

UIMA Annotations. A xml based projection language[4]allows to project standard annotations taken against any domain ontology with respect to a given Type System. Currently there is no system supporting manual production of UIMA annotations from Web Pages. Annotations taken by human annotators can be reused to train machine-learning based AEs as well as to evaluate the output of AEs by producing golden-standard annotated documents.

Annotations taken through feature 2 can be exported in different formats, providing that their content can be projected according to *begin*/*end* attributes of UIMA AnnotationBase feature. By default, UIMAST exploits x-pointer annotations taken through the RangeAnnotator[5] extension of Semantic Turkey.

During Milestone 2, we produced a cross-SOFA[6] annotator which is able to parse content of specific document formats (such as HTML, PDF etc…) and produce cross-annotations setting links between pure raw-text surrogates of analyzed documents and their original source formats. An HTML version of this annotator thus accepts HTML documents, stores their content in a dedicated HTML SOFA, then runs an HTML SAX parser erupting raw-text content which is stored in a dedicated SOFA and cross-linked with the tag elements of the former one.

X-pointer annotations taken over the HTML page can thus be easily aligned with annotations taken over raw-text. This alignment allows to produce standard char-offset annotations starting from those manually taken with the interactive UIMA annotator, as well as to project automatically generated annotations produced by UIMA AEs (which usually work over raw text content) over X-Pointer references; as a consequence they can be visualized inside the same web page under analysis (which is the objective of milestone 2).

Currently, the new release of UIMAST provides:

1. A projection editor (figure 3: supporting only simple Class and Property projections)
2. A UIMA pear installer, able to load UIMA pear packages
3. The Visual Knowledge Acquisition Tool (KA Tool or simply KAT).

KAT provides visual anchors for users willing to semi-automatically import textual information present inside

[4] http://semanticturkey.uniroma2.it/extensions/uimast/schemas/projection-20081117.xsd

[5] http://semanticturkey.uniroma2.it/extensions/rangeannotator/

[6] SOFA: Subject OF Analysis, a perspective over a (multimodal) artifact, see UIMA User guide

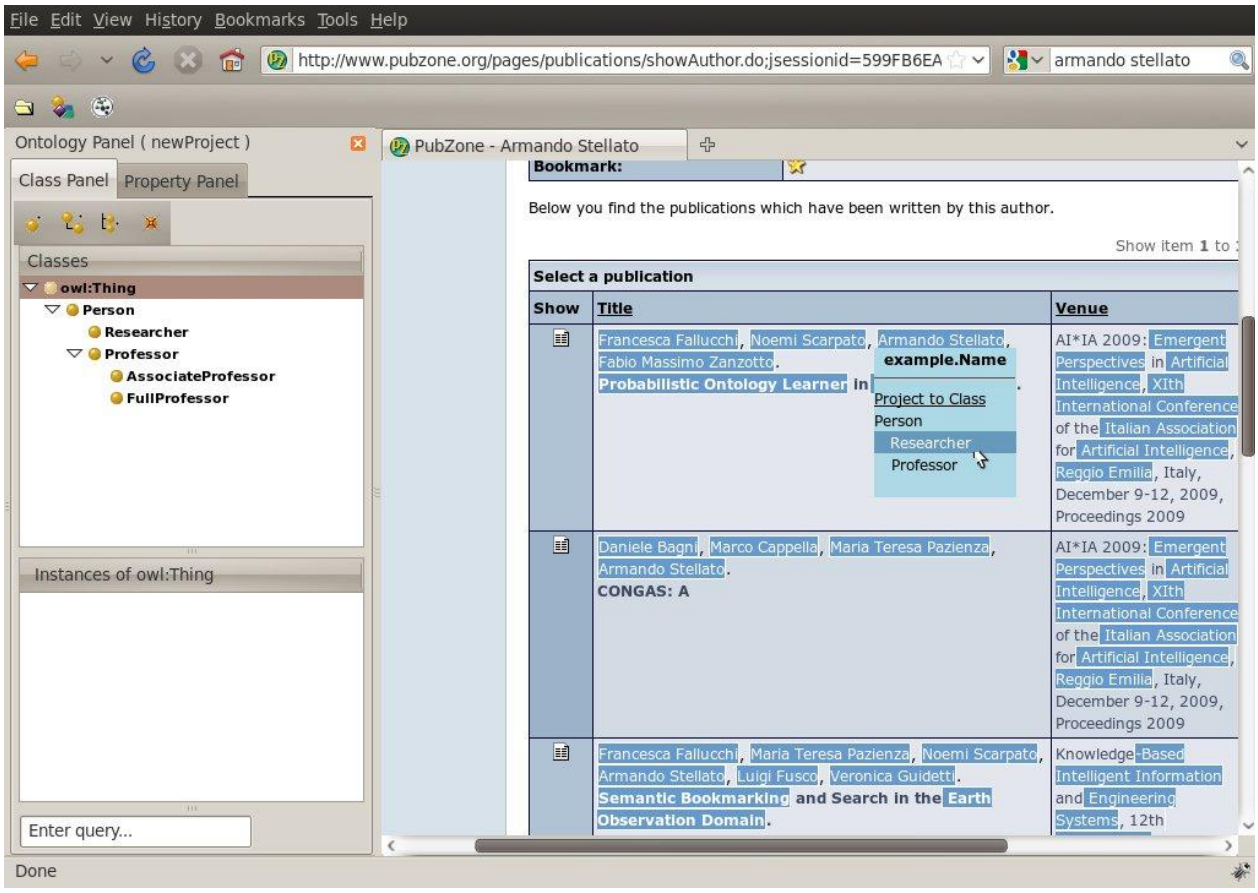

Figure 4. Knowledge Acquisition with UIMAST

web pages into the current working ontology of Semantic Turkey. While knowledge acquisition in standard Semantic Turkey requires the user to perform manual work (discovery of useful info and annotation) and decision making (produce data from annotated elements), UIMAST KAT heavily exploits the background knowledge available from the Type System of the loaded AEs and the projection document, as well as beneficiating of support coming from CODA components in the form of smart suggestions, resolved identities etc…) thus speeding up the acquisition process in the direction of automatizing the task.

In an ordinary KA session, the user starts by defining the tool setup: this implies loading one or more UIMA pears[7] through the *pear installer* and then loading a projection document associated to the currently edited ontology and the imported pears (all of the above may be stored as default settings for the ontology project being edited so that this process will not need to be repeated each time).

After tool setup, the user can immediately inspect web pages containing interesting data which can be extracted by the loaded AEs. The KAT then highlights all the text sections of the web page which have been annotated by the AEs. Each of these dynamically added highlights is not a purely visual alteration of the underlying HTML, but an active HTML component providing fast-to-click acceptance of proposed data acquisitions as well as more in-depth decision making procedures.

As an example of integrated process involving different resources in a user defined application, see figure 4 where a simple Named Entity Recognizer (the one bundled with UIMA sample AEs) has been projected towards ontology class Person. The AE has been launched and named entities discovered over the page have been highlighted. When the user passes with the mouse over one of these highlighted textual occurrences, the operation available from the projection doc is shown, and the user can right away either authorize its execution, or modify its details. In the example in the figure, the user has been prompted with the subtree rooted in the projected ontology class, and the user chooses to associate selected name to class Researcher instead of the more general Person. Should an identity resolution component discover that the given text may correspond to an existing resource (from the same edited ontology or from an external ENS), then he may choose to associate the taken annotation to it or reject it and create a new one.

6. Conclusion

The engineering of complex processes involving manipulation, elaboration and transformation of data and synthesis of knowledge is a recognized and widely accepted need, which lead in these years to the reformulation of tasks in terms of processing blocks other than (more than?) resolution steps. While traditional research fields such as Natural Language Processing and Knowledge Representation/Management have now found their standards, cross-boundary disciplines between the two need to find their way towards real applicability of approaches and proposed solutions. CODA aims at filling this gap by providing on the one hand a common

[7] A UIMA components package

environment for ontology development through knowledge acquisition, and on the other one by reusing the many solutions and technologies which years of research on these fields made easily accessible .

We hope that the ongoing realization of CODA will lead to a more mature support for research in the fields of both ontology learning and ontology/lexicon interfaces, .

7. References

Baker, C., Fillmore, C., & Lowe, J. (1998). The Berkeley FrameNet project. *COLING-ACL*. Montreal, Canada.

Basili, R., Vindigni, M., & Zanzotto, F. (2003). Integrating Ontological and Linguistic Knowledge for Conceptual Information Extraction. *IEEE/WIC International Conference on Web Intelligence*. Washington, DC, USA.

Bouquet, P., Stoermer, H., & Bazzanella, B. (2008). An Entity Naming System for the Semantic Web. *In Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*. Springer Verlag.

Buitelaar, P., Declerck, T., Frank, A., Racioppa, S., Kiesel, M., Sintek, M., et al. (2006). LingInfo: Design and Applications of a Model for the Integration of Linguistic Information in Ontologies. *OntoLex06*. Genoa, Italy.

Buitelaar, P., Olejnik, D., & Sintek, M. (2004). A Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. *Proceedings of the 1st European Semantic Web Symposium (ESWS)*. Heraklion, Greece.

Carpenter, B. (1992). *The Logic of Typed Feature Structures. Cambridge Tracts in Theoretical Computer Science* ((hardback) ed., Vol. 32). Cambridge University Press.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P., & Rice, J. P. (1998). OKBC: A programmatic foundation for knowledge base interoperability. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)* (pp. 600-607). Madison, Wisconsin, USA: MIT Press.

Cimiano, P. (2006). *Ontology Learning and Population from Text Algorithms, Evaluation and Applications* (Vol. XXVIII). Springer.

Cimiano, P., & Völker, J. (2005). Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems*, (pp. 227-238). Alicante.

Cimiano, P., Haase, P., Herold, M., Mantel, M., & Buitelaar, P. (2007). LexOnto: A Model for Ontology Lexicons for Ontology-based NLP. *In Proceedings of the OntoLex07 Workshop (held in conjunction with ISWC'07)*.

Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia.

Ferrucci, D., & Lally, A. (2004). Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng. , 10* (3-4), 327-348.

Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubézy, M., Eriksson, H., et al. (2003). The evolution of Protégé-2000: An environment for knowledge-based systems development,. *International Journal of Human-Computer Studies , 58* (1), 89–123.

Griesi, D., Pazienza, M., & Stellato, A. (2007). Semantic Turkey - a Semantic Bookmarking tool (System Description). In E. Franconi, M. Kifer, & W. May (A cura di), *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. 4519*, p. 779-788. Springer.

Harman, D. (1992). The DARPA TIPSTER project. *SIGIR Forum , 26* (2), 26-28.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1993). *Introduction to WordNet: An On-line Lexical Database.*

Pazienza, M. T., & Stellato, A. (2006). Exploiting Linguistic Resources for building linguistically motivated ontologies in the Semantic Web. *Second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006), held jointly with LREC2006.* Magazzini del Cotone Conference Center, Genoa, Italy.

Pazienza, M. T., Stellato, A., & Turbati, A. (2010). A Suite of Semantic Web Tools Supporting Development of Multilingual Ontologies. In G. Armano, M. de Gemmis, G. Semeraro, & E. Vargiu (Eds.), *Intelligent Information Access. Studies in Computational Intelligence Series.* Springer-Verlag.

Pazienza, M., & Stellato, A. (2006). Linguistic Enrichment of Ontologies: a methodological framework. *Second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006)*. Genoa, Italy.

Pazienza, M., Scarpato, N., Stellato, A., & Turbati, A. (2008). Din din! The (Semantic) Turkey is served! *Semantic Web Applications and Perspectives.* Rome, Italy.

Pazienza, M., Stellato, A., & Turbati, A. (2008). Linguistic Watermark 3.0: an RDF framework and a software library for bridging language and ontologies in the Semantic Web. *Semantic Web Applications and Perspectives, 5th Italian Semantic Web Workshop (SWAP2008)*. FAO-UN, Rome, Italy.

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

Peters, W., Montiel-Ponsoda, E., Aguado de Cea, G., & Gómez-Pérez, A. (2007). Localizing Ontologies in OWL. *In Proceedings of the OntoLex07 Workshop (held in conjunction with ISWC'07).*

Shi, L., & Mihalcea, R. (2005). Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. *CICLing 2005*, (pp. 100-111). Mexico.

Velardi, P., Navigli, R., Cucchiarelli, A., & Neri, F. (2005). Evaluation of ontolearn, a methodology for automatic population of domain ontologie. In *Ontology Learning from Text: Methods, Applications and Evaluation.* IOS Press.