EVONTO
Joint Evolution of Ontologies and Semantic Annotations

Anis Tissaoui, Nathalie Aussenac-Gilles, Nathalie Hernandez
IRIT, Université Paul Sabatier, 118, route de Narbonne, Toulouse, France

Philippe Laublet
STIH, Maison de la recherche, Université de Paris Sorbonne, 28 Rue Serpente, 75006 Paris, France

Keywords: Ontology and terminology evolution, Semantic annotation, Protégé Plug-in.

Abstract: In dynamic contexts, ontologies and their lexical component (termino-ontologies or TOR) have to frequently adapt to domain evolutions, new uses and new user needs. Among all depending data, ontology-based semantic annotations also are regularly updated to annotate new documents or to reflect new points of view. Within the TextViz ontology-based annotation framework, we propose the EvOnto tool and method that supports a coherent joint change management of termino-ontologies and semantic annotations as well as quality criteria to evaluate automatic text annotations and to detect lacks in the ontology.

1 INTRODUCTION

Early definitions of ontologies insisted on their consensual and stable content that would guarantee a sharable conceptualization of domain concepts and knowledge. In the Semantic Web, semantic annotations take the form of indexes based on ontology concepts and/or relations. In this scope, ontologies play a key role: they provide the domain vocabulary used to tag or describe the content of unstructured web pages.

Ontology based semantic annotations require that they are rich enough to describe document content; that concepts have a large variety of labels to anticipate lexical variations of terms. Then ontologies have to be frequently adapted to the new data that they describe, to the new applications that use them and to domain evolutions (Maedche, 2002). Changes on an ontology aim at making it more appropriate to model domain knowledge and its uses (Flouris et al., 2006). This process may be complex and challenging, even more for large ontologies or in dynamic contexts. It raises practical issues, like how to appreciate the right changes that are needed, how they impact the ontology uses and what are the consequences of a change on other ontology elements. Research on ontology evolution produced a taxonomy of ontology changes, identified the main stages of this process, proposed versioning tools, the logical consequences of change on the overall ontology (Stojanovic, 2004) (Klein, 2004). Still it has to take into account a large variety of reasons that lead to update ontology content.

Our research follows this trend but we focus on some specific features. First, we manage termino-ontologies, i.e. ontologies with a lexical component, where terms appear as entities in the ontology in addition to conceptual classes. Second, ontology change is driven by the annotation needs, when new data (textual documents for us) require annotation. Third, we want to reduce the negative impact of each evolution (on the ontology or the annotation) by showing these impacts at each step. For instance, negative impacts could be to run a useless new annotation or to produce lower quality annotations from a new semantic structure.

Managing terms and semantic annotations in the ontology evolution process extends the classical issues of change management: how can we ensure the consistency of the ontology and the anotations when one of the two is modified? which is the optimal consequence of a given change and how to guide the ontology engineer in its selection? How to reduce the impact of a change on the TOR and the annotations? how to detect evolution needs from

Tissaoui A., Aussenac-Gilles N., Hernandez N. and Laublet P..
EVONTO - Joint Evolution of Ontologies and Semantic Annotations.
DOI: 10.5220/0003658602260231
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2011), pages 226-231
ISBN: 978-989-8425-80-5

new uses of the ontology? All these questions are the basis for defining the EvOnto (Evolution of Ontology) method and tool. EvOnto enriches the TextViz annotation platform (Reymonet et al., 2009), a Protégé plug-in dedicated to the semantic annotation of domain specific text collections.

In this paper, we motivate and detail the EvOnto method. We first present the Dynamo project and some specific constrains on annotations and on the TOR model. Then we report a short state of the art about ontology evolution. The core of the paper is dedicated to the presentation of the EvOnto method and support tool, illustrating the two processes: from new annotations to ontology changes and back, from ontology evolution to annotation update.

2 THE DYNAMO PROJECT

The overall purpose of the DYNAMO project (DYNAMic Ontologies for information retrieval, http://www.irit.fr/DYNAMO) is to design a method and to implement a set of tools that manage domain ontology evolution and its use for semantic annotation and search in dynamic context. Changes in the context may mean new domain knowledge to be taken into account, new documents to be added to the searched collection or new user needs or queries. One of the innovations in DYNAMO is the joint specification of two modules, one for ontology evolution and one for semantic annotation and search. The goal is twofold: on the one hand, be able to take into account the document collection evolution to adapt the ontology, and, on the other hand, to manage the annotation update in keeping with any ontology change. A key feature of DYNAMO is to involve three application domains proposed by 2 companies and a research lab: (i) research in archaeology of techniques, (ii) diagnosis and fault repair of car electronic components and (iii) diagnosis and management of software errors.

DYNAMO is concerned with using and managing the evolution of enriched ontologies with a lexical component – or terminology -, that we will call from now on *termino-ontological resources* (TORs). TORs contain representations of domain concepts, relations and properties as well as terms that designate these concepts. Terms are considered both as linguistic formulations of concepts and as means to keep track of concepts or concept instances in documents. In DYNAMO, the annotation process relies on mapping terms with the language used in text. Annotations are graphs connecting term and concept instances, terms being anchored in text at precise locations.

To benefit of the web standards, the DYNAMO TOR meta-model relies on OWL-full (Reymonet *et al.*, 2009). `obir:Term` and `obir:Domain Class` are two meta-classes that specialize `owl:class`. The `denote` relation from `Term` to `Domain Class` may connect one or several `terms` to one or several meanings. Each `obir:Term` instance is a term occurrence that `designates` a concept instance. In Textviz, Protégé interfaces have been adapted to manage `obir:term` in addition to standard OWL classes and properties.

3 ONTOLOGY EVOLUTION

Ontology evolution appears in the literature in the scope of ontology maintenance as part of more global methods like KAON (Maedche et al., 2003). Several findings propose guidelines and tools (i) to identify change needs (Cimiano et al., 2005) or to detect new knowledge that leeds to some change in the ontology (Klein, 2004), (Plessers et al., 2005) (ii) tools that implement change application (Stojanovic, 2004) (Flouris, 2006); (iii) checking rules to maintain the ontology consistency (Stojanovic, 2004) (Djedidi, 2009) or (iv) versioning (Klein, 2004). Other studies defined an overall evolution process that includes both the analysis of the ontology evolution impacts and the propagation of any change on all the applications and artefacts depending on the modified ontology (Stojanovic, 2004) (Klein, 2004) (Luong, 2007). (Flouris 2006) identifies the features that differentiate evolution support tools from version management tools. In EvOnto we focus on assisting ontology evolution, not versioning.

Some tools are particularly interesting for us: KAON (http://kaon.semanticweb.org) is one of the pioneer ontology edition and engineering platform from text. It integrates an ontology evolution support (Maedche et al., 2003). Several change types are defined as subtypes of the ChangeLog class: addEntity, deleteEntity, ModifyEntity, which bear on a single structure. KAON does not deal with complex changes like splitting or merging concepts.
ECCO offers a collaborative and contextual environment to build ontologies. It is one of the blocks of the CoSWEM (Corporate Semantic Web Evolution Management) evolution management tool (Luong et al., 2007). CoSWEM manages the way vocabularies can be enriched from text when these

vocabularies are used for text semantic annotation. A history log file keeps track of the process followed to build or update an ontology thanks to RDF meta-data. More recently, the EVOLVA plug-in of the NEON tool-kit for ontology engineering provides facilities to enrich domain ontologies. EVOLVA reuses texts and parts of reusable resources like existing ontologies and databases (Zablith., 2009). EVOLVA extracts terms from text corpora, which leads to define new concepts.

4 THE EvOnto METHOD

EVOLVA and CoSWEM share similar goals as ours, but none of then is able to both manage TORs and keep an ontology consistent with annotations. This is why we have proposed the EvOnto tool and method for ontology and annotation evolution.

EvOnto is intended to be used by a single person (the knowledge engineer in the following) in charge of a coherent change management of termino-ontologies and semantic annotations. Text collections have a reasonable size (from 1000 up to 5000 texts in each of the three case studies of the DYNAMO project). They are very homogeneous (each document has the same structure and the same type of content). Depending on the case study, either the whole document or only some paragraphs are annotated.

EvOnto relies on three original features: it pays special attention to terms that contribute to define annotations; it defines quality criteria to evaluate the result of automatic text annotation and to detect lacks in the ontology; it assumes that the ontology quality is evaluated through its use for annotation. EvOnto leads to define "minimal" and "document-driven" domain ontologies: they are detailed enough to provide an optimal annotation of the text collection, but they do not pretend to fully describe a domain.

The method defines a cyclic process: after the annotation of new documents, EvOnto proposes to check their quality and indentify needs for the ontology evolution. Various TOR change operators are proposed and, for each of them, EvOnto allows to tune the consequences of this change within the TOR and back to the annotations.

The first process is data driven and deals with the impact of new annotations on the TOR following three steps.

Firstly, documents are added to or withdrawn from the text collection. Document withdrawal can lead to no longer use some concepts and terms for annotations. Up to now, no special device is provided to automatically check for unused items. But EvOnto makes it possible to "view the uses" of any a selected term or concept: it displays the list of all the documents annotated by this item.

Secondly, after adding new documents to the collection, the knowledge engineer can launch the annotation module. The originality of EvOnto is to propose to define quality criteria for the annotations, and to evaluate new annotations according to these criteria. A criterion gathers a set of concepts and/or relations that are expected to be found in each text annotation. An annotation will not be valid unless at least an instance of these concepts/relations or one of their sub-concepts is used for annotation. In general, these concepts are very high level classes in the ontology. Checking if annotations are compliant with the criteria results in a score for each document that reflects how well it is described by the annotations.

Thirdly, the knowledge engineer can browse the document list, starting with those with lower scores. Although this process is manual, it is efficiently guided because the missing expected concepts as well as the part of text without annotation are well identified. These lacks may lead to changes in the ontology, in general to the addition of new terms to existing concepts or the addition of new sub-classes to the expected ones in the annotation criteria.

We have presented here how new annotations may drive ontology evolution in EvOnto. We will detail the steps of the reverse process, when modifications in the ontology require updating annotations.

Ontology evolution is driven by the knowledge engineer. We specified the EvOnto method so that it could help him to be aware of the impact of any change both on the TOR and the annotations, and to let him adapt the change consequences case by case if needed. The process takes place in 4 stages. EvOnto supports (1) the selection of the entity to be modified and the selection of a type of change selection, (2) the decision making about the consequences or selection of an evolution strategy and (3) the impact adaptation. Then (4) it propagates the change on the annotation in a way such as it avoids an overall new annotation and it reduces side effects.

To express an evolution need, we have identified all the possible *types of changes* to be made on a TOR, and made explicit their meaning (Tissaoui, 2009). This new typology of change extends the one proposed by Stojanovic in 2004 in a way that makes it suitable for Reymonet's TOR meta-model (2009).

In particular, changes may occur on terms or on term-concept relations if the domain vocabulary evolves. Changes may be either elementary (bearing on one type of meta-model structure, like DeleteConcept or CreateTerm) or composite if several structures are involved (like merge or split).

Each type of change may lead to various options when deciding what to do with related structures in the TOR We call an *evolution strategy* a coherent way to manage the impact of a change on all the related structures with the one(s) that is (are) being modified. For instance, a possible strategy when deleting a concept is to delete all its sub-classes and all related terms. We have defined strategies that extend the one proposed by Luong (2007) and Stojanovic (2004) in order to deal with the terminological component of a TOR.

For each change and each evolution strategy, the corresponding consequences of this change can be shown to the knowledge engineer before these consequences are actually performed. In EvOnto we have defined this stage as a decision process. A first innovation at this stage is that consequences on annotations are also taken in to account. EvOnto lists all the document tagged with the modified item (*direct consequences on annotations*) as well as the documents tagged by related structures that will be modified as a consequence of the initial evolution (we call these *un-direct consequences on annotation*). A second innovative option here is to let the knowledge engineer adapt all the consequences that are not appropriate according to him. Strategies offer a global and rapid solution that can be adapted more precisely according to the evolution semantics.

5 EvOnto TOOL

The method is implemented in a support tool, the EvOnto plug-in, which includes the TextViz annotation and TOR management tool. In the following, we will illustrate the two processes presented above on one of the three case-studies of the DYNAMO project. The document collection is made of bug-tracking reports, and the TOR represents the main concepts of the software maintenance domain. This TOR includes its high level concepts (`Trigger_Event`, `Default`, `Component`), some of the associated terms (`T_Default`, `T_Problem` and `T_Bug` for instance) and semantic relations like `Concerns_Default`, `Affects-Component`, `Causes`, `Located_in`.

To ensure an efficient sematic search, each bug report file must be annotated with at least the following data:

- an instance of each one of the two concepts: `Default` and `Component`, or of their sub-classes;
- An `Affects_Component` relation between these instances.

The EvOnto interface allows to capture annotation quality criteria. The user can select concepts. It means that he expects that each document should contain one or several instances of these concepts or one of their sub-classes. He can also select the expected relations between these concepts. Once a set of criteria has been defined, it is matched to the current annotations. Results are displayed and the annotations not fulfilling the criteria are indicated. For each document where one or several concepts are missing, the knowledge engineer can display its content and select words that have not been annotated yet to define new concepts or to add terms to the ontology.

The `DeleteConcept` is an elementary change operation. EvOnto opens a new window that displays all the consequences of this change on the TOR and the annotation according to the default strategy for `DeleteConcept`, which is to attach the concept subclasses and related terms to the father concept of the deleted one. These consequences are defined so that they keep a high consistency between the TOR and the document annotations. The knowledge engineer can select another strategy and check its consequences, or he may decide to give up the modification.

Whatever the selected change, concept and strategy, the information is distributed in 4 areas: name of the current change and the modified concept (Changed Concept); information about the selected strategy (or the default one); situation of this concept in the concept hierarchy (concept Browser) and its related terms (classes starting with T_ in the Term Browser); all the change consequences (Lexical and Conceptual Information), with the upper part dedicated to consequences on the TOR, and the lower part that lists the text with annotations that could have to be changed.

In the running example, three strategies are possible for `DeleteConcept`:

- Attach the subClasses to the superClasses (default strategy),
- Attach the subClasses to the DomainThing,
- Delete the subClasses.

Then either the knowledge engineer validates

this strategy and all the previous changes, or he selects another strategy. Whatever the strategy, he can decide to modify only a part of the consequences. The ADJUST CONSEQUENCES option is dedicated to support such fine grain changes. The process is almost the same for a composite change. The major changes are the proposed strategies and the adjustment interface.

At the time being, ten change operations have been implemented with the corresponding strategies and the adequate window to adapt the consequences of the change.

6 PROPAGATION ON ANNOTATIONS

The consecutive evolution of annotations takes place in two stages:

- **Detection of Inconsistent Annotations** after the ontology has evolved; these annotations are those referring to one of the modified terms, concepts or relations;
- **Modification of Inconsistent Annotations** in keeping with the new ontology content.

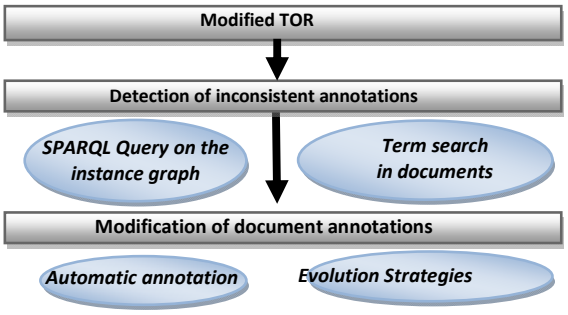

Figure 1: Propagation of TOR evolutions on annotations.

Two methods have been identified for the **first stage.** The first one requires to *browse the annotation graphs* and to look in these graphs for modified items in the ontology. For instance, any RDF triple in which appears a deleted concept is no longer valid. To search for this type of data, we defined SPARQL queries that search the set of all annotation graphs.

The second method consists in browsing the documents themselves instead of the annotations. The ideas would be to look for the terms denoting any of the changed entities in the TOR.

The **second stage** of the propagation process aims at fixing the annotations once the inconsistencies have been identified. If the second method is used for checking the documents that have inconsistent annotations, the only possible way to fix the annotations is to run again the automatic annotation algorithm with the new TOR. The computation time is short enough to make this an easy solution. Nevertheless, many annotations can be improved manually, in particular the conceptual relations in the annotation graph. Because the implemented relation extraction solution is very basic, some of the found relations are trivial and manually modified. In that case, running automatic annotations may lead to lose all the manual work. It would also require a new manual checking of all the documents and their annotations, which is quite time consuming.

The first method (detection of inconsistent annotations) makes it possible, in most cases, to locally modify the graphs in a “surgical way”. We apply here annotation evolution strategies (AS).

Each strategy is mapped to one of the TOR evolution strategies. The goal of a strategy is to fix the inconsistencies due to changes in semantic annotations. Each change operation is associated a set of rules that will modify the annotations.

We have started to evaluate how helpful EvOnto is when evolving an ontology and a document collection. The tool makes it possible to take better justified decisions, but we have to check whether these decisions are more accurate than without EvOnto. Indeed, it is quite complex to evaluate an interactive tool like EvOnto. We have not yet carried out a full evaluation of the overall process, but we have tested several measures on the results obtained in two of the DYNAMO case-studies: one in the domain of software bug tracking and the other one in the domain of car electronic fault diagnosis.

We have carried out several SplitClass modifications and compared the time required to make such changes in EvOnto and in Protégé without any evolution tool. We have not yet considered the impact on the annotations because we could not have made such modifications with Protégé. The time required to perform a change and its consequences in the TOR can be estimated to about 1 minute using EvOnto, 2 minutes using Protégé. The analysis of the change log obtained in both cases shows that EvOnto takes into account all the the types of changes and their consequences, where as Protégé only manages 3 types of changes: C0 is deleted and new1 and newC2 are created.

In addition, EvOnto provides a qualitative improvement of TOR and annotation evolution. The tool anticipates all the consequences of a change on the TOR and on the annotations, which avoids

forgetting some of the impacts of a change. For instance, in Protégé, it is up to the user to check all the ObjectProperties in which a modified concept is involved as domain or range, on to anticipate by moving sub-classes before deleting their super class. Moreover, in Protégé, taking into account annotations is completely set apart. In TextViz, it is very easy to perform a new annotation after each evolution, or to carry out local modifications by propagating changes.

7 CONCLUSIONS

We have proposed EvOnto, a method and tool for ontology evolution that takes into account its use for semantic annotation. EvOnto implements several principals for a consistent evolution of ontologies and semantic annotations. Our study brings several innovations compared with previous works. First, we are interested in ontologies with a lexical component (TORs), defining according to a meta-model where terms are represented as classes. Second, EvOnto assists the evolution process by providing the knowledge engineer with information on the consequences of a change before it is implemented. These consequences take into account the structures linked to the one modified in the TOR as well as the semantic annotations using this structure. This information supports decision making and avoid costly trial and attempts.

We go on improving EvOnto by adding new change operations and their corresponding strategies to manage the consequences in the TOR and on annotations. Most of our effort now is dedicated to the evaluation of EvOnto. This evaluation raises issues related to the time required to build an ontology, to the difficulty to judge the quality of an ontology and even more of semantic annotations. Thanks to the three case studies of the DYNAMO project, we have data, ontologies and domain experts to carry out several evaluation experiments.

REFERENCES

Cimiano, P., Völker, J. (2005). Text2Onto - a framework for ontology learning and data-driven change Discovery. In *LNCS: Vol. 3513. Natural Language Processing and Information Systems*. Berlin: Springer, 227-238.

Djedidi, R. (2009). Approche d'évolution d'ontologie guidée par des patrons de gestion de changement. Thèse de doctorat, université Paris-Sud XI Orsay. 2009.

Flouris G., Plexousakis D. and Antoniou G. (2006), A classification of ontology change, *Proc. of the 3rd Italian Semantic Web Workshop Scuola Normale Superiore, Pisa, Italy*, 18-20 December, 2006. CEUR-WS201.

Flouris, G. (2006). On belief change and ontology evolution. Ph.D. Thesis, *University of Crete, Department of Computer Science*, Heraklion, Greece.

Klein, M. (2004). Change management for distributed ontologies. Ph.D. Thesis, *Dutch Graduate School for Information and Knowledge Systems*. Germany.

Klein, M., Fensel, D., Kiryakov, A., & Ognyanov, D. (2002). Ontology versioning and change detection on the web. LNCS: Vol. 2473. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web* (pp. 197-212). Berlin: Springer.

Luong, P. H. (2007). Gestion de l'évolution d'un web sémantique d'entreprise. Thèse de doctorat, *école de Mines de Paris*.

Mäedche, A. (2002). Ontology learning for the Semantic Web, vol. 665, *Kluwer Academic Publisher*.

Mäedche, A., Motik B., Stojanovic L. (2003). Managing multiple and distributed ontologies in the Semantic Web. *VLDB Journal*, 12(4), 286–300.

Plessers, P., De Troyer, O. (2005). Ontology change detection using a version log, In Y.Gil, E. Motta, V.R. Benjamins, & M. Musen (Eds.), LNCS: Vol. 3729. *The Semantic Web – ISWC 2005*. Berlin, Germany: Springer-Verlag, 578-592.

Reymonet, A., Thomas J. & Aussenac-Gilles N. (2009). Ontology Based Information retrieval: an application to automotive diagnosis. *International Workshop on Principles of Diagnosis (DX 2009)*. Stockholm M. Nyberg, E. Frisk, M. Krisander, J. Aslund (Eds.), Linköping University, *Institut of Technology*, pp 9-14.

Roux, V., Aussenac-Gilles, N. (2010). Knowledge Bases and Query Tools for a Better Cumulativity in the Field of Archaeology: The Arkeotek Project. *Computer Applications and Quantitative Methods in Archaeology*, *Granada (Spain)*, 07-09/04/2010, UCMSS, p. 1-5.

Stojanovic, L. (2004). Methods and Tools for Ontology Evolution, Ph.D. Thesis, *Karlsruhe University*. Germany.

Tissaoui, A. (2009). Typologie de changements et leurs effets sur l'évolution de Ressources Termino-Ontologiques (Poster) 20e *Journées Francophones d'Ingénierie des Connaissances IC2009*, Hammamet (Tunisie) http://ic2009.inria.fr/docs/posters/Tissaoui_Poster_IC2009.pdf

Zablith, F. (2009). Evolva: A Comprehensive Approach to Ontology Evolution. *ESWC 2009*: 944-948.