Evaluating the Impact of Ontology Evolution Patterns on the Effectiveness of Resources Retrieval

Mauro Dragoni and Chiara Ghidini

FBK–IRST, Trento, Italy
dragoni|*ghidini*@fbk.eu

Abstract. The usage of ontologies for annotating resources significantly growth in the last years. However, ontologies are dynamics artifacts that may change during time due to the change of the conceptual representation of the domain they describe or to the necessity of modifying the definition of some entities. The usage of repositories containing annotated resources available for users, like in the web environment, leads to the necessity of managing the changes carried on the ontology in order to avoid both the decrease of effectiveness of the search system and the change, by the users, of how they have organized their knowledge. In this work, we have investigated how ontology evolution operations impact on the effectiveness of search systems. We have performed the evaluation by implementing an ontology versioning approach that permits to fully track the changes carried out on the ontology and by performing an information retrieval evaluation by using two different document collections, the MuchMore collection and the TREC 2004 Genomic Track collection.

1 Introduction

Ontologies are dynamic entities that evolve over time because they need to reflect changes in the domain they describe, in their conceptualization, or in their specification. As stated in [1], ontology evolution can be defined as "the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artifacts.".

Starting from a high level perspective, we can look at the ontology evolution field from two main points of view:

- the first one is the evolution of the ontology "per se". This activity is usually performed by the domain experts in order to adapt the ontology to the arisen changes of the domain they want to describe, in its conceptualization or in its specification. Supporting this activity requires to address a number of challenges and has originated a number different research efforts and techniques, ranging from the adequate control of ontology changes to the administration of ontology versions, to knowledge reconciliation in collaborative ontology editing, to the (in)consistency management, and so on.
- the second one is to look at the evolution of the ontology from the point of view of users and tools that exploit the ontology for different tasks. Here, the main requirement is that the evolution of the ontology is transparent to them in order to

avoid both the loss of effectiveness of the tools and the necessity, by the users, of adapting their knowledge to the changes of the ontology.

In this paper we look at ontology evolution from this latter point of view, in particular from the point of view of third-party tools which use ontology concepts for resource annotation or for supporting information retrieval. In such scenarios, the evolution of an ontology may have an impact on the effectiveness of the resource retrieval and the third party tools have to be able to understand what are the changes performed in the ontology, and how they have been represented in order to:

- update the set of concepts available for annotation, and the relationships between these concepts; and
- extend the search environment with a mechanism for navigating through the different versions of the ontology in order to maintain the effectiveness of the information retrieval system.

The reason why we look at the problem from the point of view of applications is that while in the literature several approaches have been proposed for supporting the evolution and versioning challenges of an ontology "per se" (see Section 2 for a brief overview of some of the approaches), much less effort has been devoted to investigate the impact of the ontology evolution and versioning mechanisms from the usage side, and on developing techniques to manage them in an effective manner. In this paper we take a first step towards this, by investigating the impact of ontology evolution on the effectiveness information retrieval systems where queries are performed by using concepts defined in an ontology that evolves through the time. More in detail, we start from a scenario in which a collection of documents is annotated with the version "t" of an ontology, and where the ontology is evolved (one or more times) by domain experts using editing operations such as concept deletion, replace, move, addition and so on. All these activities lead to the deployment of n further versions "$t + x_1, \ldots t + x_n$" of the ontology which are more adequate to express an up-to-date knowledge of the domain, but are likely to be "disaligned" with the terminology used to classify documents. In fact, reclassification is an expensive task and usually the annotation of resources is not updated when an ontology evolves. This is, e.g., the case of the MeSH light-ontology[1] which is updated twice a year, and for which specific, different, versions have been used to annotate big collections of documents such as MuchMore and TREC 2004. Thus, while it would be nice, from a (human or software) user perspective to be free to use an arbitrary version of the ontology for information retrieval, it may be the case that the choice of the ontology version has an impact on the performances of the retrieval itself, both in terms of precision and recall. In this paper we take inspiration from scenarios such as the one of MeSH, and aim at evaluate the impact of ontology changes on the effectiveness information retrieval systems where queries are performed by using concepts defined in an ontology that evolves through the time. In order to do this, we first specify the list of transformations that we take into account to describe ontology change. This is based on the usage of well known ontology transformation patterns that we recall in Section 3, which we use to track the full history of change

[1] http://www.nlm.nior

of the ontology and therefore to manage the rewriting of the queries performed on the different versions of the ontology. Then, in Section 4 we evaluate the performance of information retrieval tasks on different versions of the ontology, analyze the limits of ontology versioning approaches, and provide hints of possible solutions.

2 Related Work

The dynamism of the ontologies and knowledge bases have been widely treated in the literature. In particular, the aspects that have been mainly taken into account are the evolution and the versioning of an ontology, while an emerging discipline is the construction of temporal ontologies in which timing annotations are used for injecting temporal information into the ontologies [2].

The evolution of an ontology aims to change the schema of the ontology without loss of data and by maintaining the consistency of the ontology. Moreover, the accessibility to both old and new data through the new ontology schema should be maintained.

The problem of the schema evolution has been faced by considering the different phases of the evolution: from the planning of the changes in which an analysis for estimating the cost and the effort required to implement the requested change [3] [4] to their implementations. Concerning the latter point, the approaches presented in the literature may be categorized in four different categories: immediate conversion (or coercion) [5], deferred conversion (or screening) [6] [7], explicit deletion [8], and filtering [6] [7].

The renovation of an ontology, may introduce inconsistencies into the ontology schema. There are two different schools of thought related to the management of the generated inconsistencies: the first one is the "consistency maintenance", that it is a conservative approach in which the system is kept consistent at all costs, for example, by disallowing some kind of changes [1] and [9]. While, the second strategy simply consists in the "inconsistency management", in which inconsistencies are considered inevitable and, therefore, it is necessary to maintain them. The main studies in the literature about inconsistency management are related to the localization of inconsistencies, for example, in [10], it is presented an approach based on the use of sub-ontologies to identify inconsistencies, while in [11] the authors present a logic-based method to detect some kinds of inconsistencies.

On the other side, the management of the versioning of an ontology aims to maintain the ability of accessing all the data (both old and new) through all versions of the ontology. A version is a reference that labels a quiet point in the definition of a schema. Therefore, all resources have to be retrievable by using coherent concepts and by using every historic definition of the ontology schema;

In [12] the authors discuss about the minimal requirements that an ontology versioning system must take into account for being considered reliable with respect to the maintenance of the ontology itself and for avoiding the issues caused by the evolution of the schema.

Two main strategies might be adopted to manage different versions of an ontology: "state-based" and "change-based". A "state-based" versioning approach consists in considering the state of the ontology at certain moment in time; a new state is created

each time a change (or a set of changes) is applied to the ontology schema. Examples of systems that supports such a versioning strategy are described in [13] and [14]

The second way to manage schema versioning is a "change-based" approach (or "operation-based") consisting in storing information about the precise changes or explicit operations that are performed on the ontology. The advantages of this approach, with respect to the previous one, is that it is simpler to compare different versions of an ontology and to implement undo/redo mechanisms. Example of systems implementing a change-based approach are proposed in [15] and [16].

3 Patterns of Change

We aim at characterizing and evaluating the impact of ontology evolution on information retrieval tasks by investigating how much the different operations used to modify an ontology can have an impact on it. Thus before performing the experiments we need to (1) provide a characterization of the change operations we consider and (2) to provide a mechanism that enriches the representation contained in an ontology with information that permits to track the changes that have been carried out from version "t" to version "$t + x_i$".

Our approach is based on the idea of transformation patterns introduced in [17]. These patterns provide a mechanism for describing the compatibility between two different ontologies in terms of how the first ontology can evolve into the second one. For the sake of our work, we have considered the case of light ontologies and we have focused on the transformation patterns that can modify the *isA* hierarchy (taxonomy). They are:

RENAME. It consists in the change of the label used for identifying the concept;
DELETE. It consists in the removal of a concept from the ontology;
MOVE. It consists in the change of the position of the concept in the ontology. Actually, this operation is composed by the two atomic operations DELETE and ADD, but the MOVE operation has a different impact on the versioning of the ontology.

The rational of choosing these operations is that they are the basic operations that support the evolution of almost all ontologies. We didn't include the **ADD** operation as new concepts appearing in version "$t + x_i$" cannot be rewritten in terms of concepts appearing in version "t". The analysis of this and of more complex case is one of the objectives of our future work. As an additional note, we remark that our claim is not to propose a new approach for facing the evolution and versioning problem, but to use methods that characterize ontology change to study the impact of the ontology evolution on real-world information retrieval scenarios.

3.1 RENAME

The rename of a concept is an action performed by a user which consists in the modification of the definition with which a concept is identified in the ontology. Our example shows the case in which the concept definition is modified by renaming its label from "A" to "B".

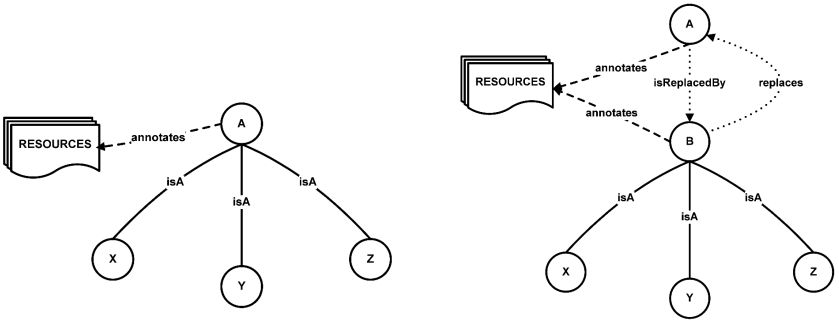

Fig. 1. Concept update by modifying its definition.

In this context, it is desirable that the old definition of the concept "A" is maintained in the ontology, because it is necessary that the resources stored in the repository, that has been annotated with the concept "A", still remain retrievable when users look for resources annotated with the concept "A". Figure 1 presents the scenario in which a concept is renamed. The left part of the image shows the ontology situation before the rename, while the right parts show how the ontology evolves after the rename. The starting point is the scenario in which a concept subsumes ("isA" relationship) other concepts, (the same strategies may also be applied for different type of relations), and it is used to annotated a set of resources stored in the repository.

When a concept is renamed (Figure 1), the ontology is modified by inserting a new concept that replaces the old one; however, the old one is maintained in the ontology. The associations between the old concept and its subsumed concepts are moved, and they are placed as subsumptions of the new concept. Two relations are created between the old concept and the new one: "isReplacedBy" and "replaces". The relations "isReplacedBy" and "replaces" are used for retrieval purposes because they permit to navigate through the old concept definitions that are maintained in the ontology in order to preserve the retrieval effectiveness. In fact, the resources annotated with the definition of the old concepts still remain retrievable when users perform queries containing the definition of the old concepts.

3.2 DELETE

Differently from the rename operation, the management of the concept deletion is based on the relationships between the deleted concept and the other ones. Two different scenarios are expected: (i) the concept is a middle node of the ontology; and, (ii) the concept is a leaf of the ontology. The case in which the concept is at the top of the ontology is not managed because it is supposed that the top concept is never been deleted.

Figures 2 and 3 show the two scenarios respectively when the deleted concept is a middle node or a leaf of the ontology.

In the first case, it is supposed that the concept "B" (middle node) is deleted, while in the second case, it is supposed that the deleted concept is "Y" (leaf node). It is desirable

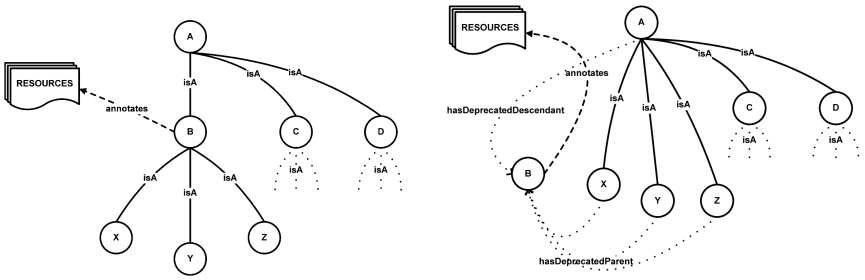

Fig. 2. Deletion of a middle-node concept.

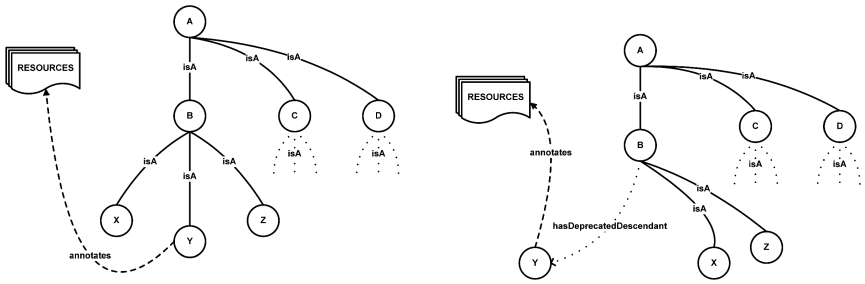

Fig. 3. Deletion of a leaf concept.

that in both cases the concepts “B” and “Y” are somehow maintained in the ontology, because, as explained earlier, it is necessary that the resources stored in the repository, that has been annotated with the concept “B” (or “Y”), still remain retrievable when users look for resources annotated with the concept “B” (or “Y”).

The left parts of the images show the ontology situation before the deletion, while the right parts show how the ontology evolves after the two possible concept deletions. In both cases, the relation used to associated the concepts is the “isA” relationship, however, the same strategies may also be applied for different type of relations. It is also supposed that the resources are annotated by using only the deleted concept.

When the deleted concept is a middle node (the concept “B” in Figure 2), the subsumed concepts (in this case the concepts “X”, “Y”, and “Z”) are directly associated with the parent concepts of “B” in order to preserve the consistency of the taxonomy. However, there is the possibility that a set of resources is annotated by using only the concept “B”. To make these resources retrievable it has been decided to use the “hasDeprecatedParent” and “hasDeprecatedDescendant” relationships. The “hasDeprecatedParent” relationship associates the concepts that was descendants of the deleted concept, with the deleted concept itself. This way, when users perform queries containing the concept “X” and/or “Y” and/or “Z”, the platform will also looks for resources that are annotated by using the concept “B”. In the same way, when a user performs queries containing the concept “A”, the “hasDeprecatedDescendant” relationship is exploited to retrieve the resources that have been annotated by using the concept “B”

In the second case (Figure 3), it is supposed that the deleted concept “Y” is a leaf node of the ontology. In this case, the evolution of the ontology is more simpler than the previous case. The “hasDeprecatedDescendant” relationship is created between the concepts “B” and “Y”, and this relationship is exploited in the same way explained for the previous case. Therefore, when users look for resources annotated with the concept “Y”, these resources still remain retrievable, while, when users look for resources annotated with the concept “B”, the resources annotated with the concept “Y” will be retrieved too.

3.3 MOVE

The move operation consists in the movement of a concept from one parent to another one as shown in Figure 2. As we have already introduced above, this operation may be split in the two atomic action “REMOVE” and “ADD”. However, we want to maintain the reference with the old parent in order to reconstruct the movement operation.

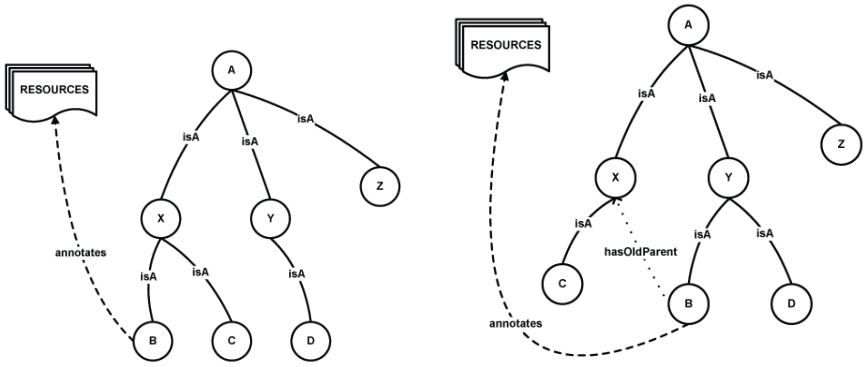

Fig. 4. Move of a concept.

Figure 4 presents the scenario in which a concept is moved from a branch of the ontology to another one. A further assumption is that the distance, in terms of graph edges in the ontology, between the old concept position and the new one, is quite low; otherwise, the change in the ontology reflects a more heavy change in the conceptualization of the domain model.

The left parts of the images show the ontology situation before the move, while the right parts show how the ontology evolves when the movement operation has been performed. When a concept is moved (Figure 1), the ontology is modified by changing the parent of the moved concept, but the old relationships have been maintained. Two relations are created between the concept and the old parent: “hasOldParent” and “hasOldDescendant”. These relations are used for retrieval purposes because, when users perform a query containing a moved concept, it is possible to expand the query by including information about its movement in order to inject into the query, information

that may be helpful for retrieving relevant documents annotated with an old version of the ontology.

4 Experimental evaluation

In this section, we present the evaluation of the impact of ontology evolution on the retrieval capabilities of search engine. We used transformation patterns for linking the changes carried out on each concept and we have exploited them for expanding the queries in order to map the concepts of each query on the different versions of the adopted ontology.

The experimental evaluation has been performed by using two different document collections annotated by using the MeSH light-ontology[2]:

- the MuchMore collection consisting in 7,823 abstracts of medical papers and 25 queries with their relevance judgments. The documents have been annotated with the 2001 version of the ontology, while the queries have been annotated with the 2002 version of the MeSH ontology. We have used this set of queries for performing the evaluation.
- the TREC 2004 Genomics Track collection consisting in 4,591,008 abstracts of medical papers annotated with the 2004 version of the MeSH ontology. This collection contain also 50 queries but we have not used them for the evaluation, as it is explained later.

The rational behind the use of two different collections is to replicate the following scenario:

- the user has its own knowledge related to a particular step of the ontology evolution process, this is represented by the queries annotated with concepts coming from one version of the ontology;
- the relevant documents contained in the repository are annotated with a different version of the ontology known by the user;
- the repository contains documents that have been annotated with a further different version of the ontology. The characteristic of this subset is that it contains only non-relevant documents; this way, the query process is "perturbed" by a huge number of documents that may be retrieved when queries are excessively expanded.

For creating the index we have adopted the indexing approach described in [18]. Briefly, this approach permits to create a conceptual representation of each document starting from the set of concepts used for representing (or annotating) it.

4.1 Evaluation of the System Effectiveness

The experiments have been performed as follows:

- we have built the transformation patterns for navigating through the history of each concept from the first version of the ontology to the last one;

[2] http://www.nlm.nih.gov/mesh/meshhome.html

- for each version of the ontology, we have exploited the transformation patterns for mapping each concept contained in the query to the version of the ontology taken into account;
- for each version of the ontology, we have performed the mapped queries on the repository and we have evaluated the effectiveness of the system.

The evaluation method follows the TREC protocol [19]. For each query, the first 1,000 documents have been retrieved and the precision of the system has been calculated at different points: 5, 10, 15, and 30 documents retrieved. Moreover, the Mean Average Precision (MAP) of the system has been calculated.

Table 1 shows the results obtained by performing all queries. The first column contains the correspondent year of the ontology, columns form the second to the fifth show the Precision at 5, 10, 15, and 30 respectively, the sixth column shows the MAP, while the last column shows the number of operations applied in each version of the ontology.

By observing the results, we can notice two different phase in the behavior of the system effectiveness: in a first phase from the 2002 to the 2007 ontology versions, the MAP values are very similar (with peaks of the MAP values in the 2006 and 2007 versions); this means the system is able to manage the evolution of the ontology. Instead, in a second phase, from the 2008 to the 2012 ontology version, the MAP values constantly decreases by highlighting an evident loss of precision in the query results. By analyzing the correlation between the effectiveness of the system and the operations carried out on the ontology during the different steps of its evolution, it is possible to infer that some operations probably has a more relevant impact on the performance of the system with respect to other ones. For instance, it is possible to notice that when the number of MOVE operations increases, especially in the versions from 2008 to 2012, both the MAP and the Precision@X values significantly decreases. On the contrary, the DELETE operations, that are concentrated in the first revisions of the ontology from 2003 to 2006, do not significantly affect the effectiveness of the system; indeed, both the MAP and the Precision@X values does not suffer of significant changes. A similar deduction may be done for the RENAME operation, that is the most frequent one; however, in this case it is not possible to conclude, by observing the general results, how much this kind of operation affects the results.

In order to analyze more in depth the effect of each kind of operations, we have performed a separated experiment by focusing on the analysis of the effectiveness of the queries in which the RENAME or the MOVE operations was more frequent. In Tables 2 and 3, we shown the results. For each table, we have considered the four queries containing the highest numbers of the RENAME and MOVE operations, respectively. Before analyzing the results, it is important to premise that the starting differences of the results obtained by the two sets of queries (for instance the Prec@5 values of 0.600 and 0.457 for the 2002 ontology version) depends exclusively by the fact that the two sets contain different queries.

By observing the results in Table 2, we can notice that the RENAME operations have not a significant impact on the effectiveness of the system. Indeed, the MAP and Precision@X values remain almost the same during the entire evolution process. Only two small changes may be reported: the first one is a very small decrease of the MAP, while the second one is the decrease of the Prec@15 value followed by a small incre-

ment of the Prec@30 value. These small changes have been caused by the RENAME operations performed on the 2004 and 2005 versions of the ontology. These operations caused a small shift of the relevant documents placed between the 10th and the 15th positions, and, this shift, has affected also the MAP value that registers a small decrease.

A different scenario is depicted by observing the results reported in Table 3, where the effects of the MOVE operations significantly impact on the effectiveness of the system. By analyzing the results, when the number of MOVE operations is small (as in the first versions of the ontology), the effectiveness of the system remain quite stable. On the contrary, when the number of MOVE operations increases (since the 2007 version of the ontology) both the MAP and the Precision@X values significantly decreases. The reasons of this loss of effectiveness may be caused by the fact that when MOVE operations are carried out on the ontology, the changes in the concepts hierarchy introduce the necessity of reconsidering how the documents have been annotated in order to decide if an update of their classification is needed. Indeed, by moving a concept "X" from a branch of the ontology to another one, all documents annotated with "X" should be analyzed for deciding if the current annotation should be maintained or if the annotation of the document has to be enriched by using, for example, the concepts located in the branch where "X" has been moved.

By analyzing the obtained results, it is possible to notice that, independently of how much the versioning mechanism is able to track the changes carried out on the ontology, there is a break point that marks the limit after that a system is not able to manage the effects of these changes. For instance, a system focused on the retrieval of annotated resources. In order to avoid this issue, two possible solutions might be adopted in the context of the considered scenario:

- a re-classification procedure (manual or automatic) has to be performed on the resources in order to update the annotations for avoiding the loss of effectiveness that may affect the retrieval system;
- the versioning mechanism may enrich information related to the changes carried out on the ontology (for instance by describing the consequences of a MOVE operation) by foreseeing the possibility that the ontology might be used from third-party users and systems.

5 Conclusions

In this paper, we have evaluated the impact of ontology evolution on the effectiveness of an information retrieval system. We have implemented an ontology versioning approach based on the use of transformation patterns that is able to track the full set of changes carried out on an ontology. These patterns have been exploited for injecting, into the OWL code used for describing the ontology, information that have been used for mapping the queries performed on the document collection on the different versions of the ontology.

The experiments have been performed by using a repository containing two different document collection annotated with different versions of the MeSH light ontology. We have considered three kinds of operations generally applied to concepts: DELETE,

Year	Prec@5	Prec@10	Prec@15	Prec@30	Mean Average Precision	Operations Rename - Move - Delete
2002	0.624	0.592	0.544	0.413	0.2996	0 - 0 - 0
2003	0.624	0.576	0.533	0.418	0.2995	17 - 8 - 26
2004	0.624	0.576	0.538	0.424	0.2988	25 - 7 - 6
2005	0.624	0.576	0.538	0.421	0.2993	18 - 9 - 12
2006	0.640	0.592	0.538	0.421	0.3001	19 - 10 - 11
2007	0.640	0.584	0.528	0.421	0.3000	22 - 8 - 5
2008	0.624	0.576	0.517	0.413	0.2981	11 - 10 - 2
2009	0.588	0.552	0.517	0.416	0.2869	20 - 14 - 0
2010	0.560	0.524	0.512	0.413	0.2834	11 - 15 - 0
2011	0.546	0.484	0.442	0.383	0.2793	10 - 19 - 0
2012	0.528	0.464	0.406	0.348	0.2750	8 - 23 - 1

Table 1. Results obtained by applying the transformation patterns for expanding the queries.

Year	Prec@5	Prec@10	Prec@15	Prec@30	Mean Average Precision	Operations Q_5 - Q_7 - Q_{14} - Q_{20}
2002	0.600	0.650	0.600	0.416	0.3976	0 - 0 - 0 - 0
2003	0.600	0.550	0.566	0.416	0.3964	0 - 0 - 1 - 0
2004	0.600	0.550	0.566	0.416	0.3962	3 - 1 - 1 - 2
2005	0.600	0.550	0.566	0.416	0.3962	0 - 1 - 1 - 0
2006	0.600	0.550	0.566	0.432	0.3962	0 - 1 - 1 - 0
2007	0.600	0.550	0.532	0.432	0.3960	2 - 0 - 0 - 0
2008	0.600	0.550	0.532	0.432	0.3960	2 - 2 - 2 - 1
2009	0.600	0.550	0.532	0.432	0.3960	0 - 0 - 1 - 2
2010	0.600	0.550	0.532	0.432	0.3960	1 - 0 - 0 - 1
2011	0.600	0.550	0.532	0.432	0.3960	1 - 3 - 0 - 1
2012	0.600	0.550	0.532	0.432	0.3960	0 - 0 - 1 - 2
					Total Operations:	9 - 8 - 8 - 9

Table 2. Results obtained on queries affected by RENAME operations.

RENAME, and MOVE. Our results shown how different operations have a different impact on the effectiveness of the system, and that, even if the versioning system is able to fully track the changes carried out on an ontology, further activities are necessary if the ontology is exploited by third-party tools or users.

Acknowledgments. Organic.Lingua is funded under the ICT Support Program of the EU Commission (Grant Agreement Number 270999).

References

1. Stojanovic, L.: Methods and Tools for Ontology Evolution. PhD thesis, University of Karlshrue, Karlshrue, Germany (2004)
2. Batsakis, S., Petrakis, E.: Sowl: spatio-temporal representation, reasoning and querying over the semantic web. In I-SEMANTICS. ACM International Conference Proceeding Series, ACM (2010)
3. Plessers, P.: An Approach to Web-Based Ontology Evolution. PhD thesis, Department of Computer Science, Vrije Universiteit Brussel, Brussel, Belgium (2006)
4. Simperl, E., Popov, I., Bürger, T.: Ontocom revisited: Towards accurate cost predictions for ontology development projects. In ESWC.Volume 5554 of Lecture Notes in Computer Science., Springer (2009) 248–262

Year	Prec@5	Prec@10	Prec@15	Prec@30	Mean Average Precision	Operations Q_2 - Q_{12} - Q_{19} - Q_{22}
2002	0.457	0.442	0.419	0.295	0.2515	0 - 0 - 0 - 0
2003	0.457	0.428	0.409	0.309	0.2515	0 - 0 - 0 - 0
2004	0.457	0.428	0.419	0.314	0.2517	1 - 0 - 1 - 0
2005	0.457	0.428	0.419	0.309	0.2517	1 - 0 - 0 - 1
2006	0.465	0.388	0.349	0.284	0.2550	0 - 0 - 0 - 0
2007	0.437	0.368	0.309	0.254	0.2437	1 - 2 - 2 - 2
2008	0.395	0.314	0.280	0.230	0.2398	0 - 2 - 1 - 1
2009	0.377	0.308	0.250	0.200	0.2358	0 - 1 - 1 - 1
2010	0.356	0.264	0.200	0.165	0.2314	2 - 1 - 0 - 0
2011	0.336	0.236	0.170	0.135	0.2248	1 - 1 - 1 - 2
2012	0.317	0.208	0.140	0.105	0.2195	1 - 2 - 1 - 1
					Total Operations:	7 - 9 - 7 - 8

Table 3. Results obtained on queries affected by MOVE operations.

5. Lerner, B., Habermann, A.: Beyond schema evolution to database reorganization. In: OOPSLA/ECOOP. (1990) 67–76
6. Andany, J., Léonard, M., Palisser, C.: Management of schema evolution in databases. In VLDB, Morgan Kaufmann (1991) 161–170
7. Ra, Y.G., Rundensteiner, E.: A transparent schema-evolution system based on object-oriented view technology. IEEE Trans. Knowl. Data Eng. **9**(4) (1997) 600–624
8. Banerjee, J., Kim, W., Kim, H.J., Korth, H.: Semantics and implementation of schema evolution in object-oriented databases. In SIGMOD Conference, ACM Press (1987) 311–322
9. Leenheer, P.D., de Moor, A., Meersman, R.: Context dependency management in ontology engineering: A formal approach. J. Data Semantics **8** (2007) 26–56
10. Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In ESWC. Volume 3532 of Lecture Notes in Computer Science., Springer (2005) 182–197
11. Plessers, P., Troyer, O.D.: Resolving inconsistencies in evolving ontologies. In ESWC. Volume 4011 of Lecture Notes in Computer Science., Springer (2006) 200–214
12. Klein, M., Fensel, D.: Ontology versioning on the semantic web. In SWWS. (2001)
13. Klein, M., Fensel, D., Kiryakov, A., Ognyanov, D.: Ontology versioning and change detection on the web. In EKAW. Volume 2473 of Lecture Notes in Computer Science., Springer (2002) 197–212
14. Pittet, P., Cruz, C., Nicolle, C.: Towards dynamic ontology. In: 4th International Conference on Advances in Semantic Programming (SEMAPRO 2010). (2010)
15. Maedche, A., Motik, B., Stojanovic, L.: Managing multiple and distributed ontologies on the semantic web. VLDB J. **12**(4) (2003) 286–302
16. Bozsak, E., et all: Kaon - towards a large scale semantic web. In EC-Web. Volume 2455 of Lecture Notes in Computer Science., Springer (2002) 304–313
17. Sváb-Zamazal, O., Svátek, V., Iannone, L.: Pattern-based ontology transformation service exploiting oppl and owl-api. In EKAW. Volume 6317 of Lecture Notes in Computer Science., Springer (2010) 105–119
18. Dragoni, M., da Costa Pereira, C., Tettamanzi, A.: A conceptual representation of documents and queries for information retrieval systems by using light ontologies. Expert Systems With Applications **39**(12) (2012) 1037610388
19. Voorhees, E., Harman, D.: Overview of the sixth text retrieval conference (trec-6). In: TREC. (1997) 1–24