

Recommendations for Better Quality Ontology Matching Evaluations¹

Aliaksandr Autayeu and Vincenzo Maltese and Pierre Andrews²

Abstract. Evaluating and comparing different ontology matching techniques is a complex multifaceted problem. Currently, diverse golden standards and various practices are used for evaluations. In this paper we show that, by following certain rules, the quality of the evaluations can be significantly improved, particularly in regard to the accuracy of precision and recall measures obtained.

1 Introduction

In the recent years, several matching tools have been proposed as solutions to the semantic heterogeneity problem. We focus on the problem of evaluating semantic matching techniques [10] which find rich mappings, that is, mappings which contain *disjointness*, *equivalence*, *less general* and *more general* links.

Most of the tools for semantic matching identify only equivalence, some identify less and more generality, but only a few include explicit disjointness [23]. Reflecting this, an overwhelming majority of available golden standards are targeted at evaluating mappings with equivalence links only. In this paper we explain why evaluating a rich mapping using such a golden standard makes results imprecise and propose some recommendations to make evaluations and comparisons between different tools fairer and more accurate.

In addition, we address important issues concerning the coverage of a golden standard and the presence of redundant links, namely links which can be logically inferred from the others in the mapping, and how they can influence the quality measures.

The rest of the paper is organized as follows. In Section 2 we present state of the art approaches. Section 3 shows how to conduct the evaluation according to the characteristics of the golden standard used. Section 4 focuses on the importance of maximizing both the mapping and the golden standard for an optimal evaluation. Finally, Section 5 concludes the paper and outlines future work.

2 Related Work

With matching techniques being the main focus of the ontology matching field, a few initiatives³ pay attention to evaluation. On the one hand, general [22, 6] and domain-specific [18, 17] evaluation experiments are reported, without discussing the evaluation methodology. On the other hand, considerable attention has been paid to appropriateness and quality of the measures [8, 9, 4].

Attention has also been brought to the mapping itself. In [20] the authors propose to complement the precision and recall with new

measures to take into account possible mapping incoherence, thus addressing the issues of internal logical problems of the mapping and the lack of reference mappings. In [24] two evaluation techniques are proposed. The first is practice-oriented and evaluates the behaviour of the mapping in use. The second focuses on the manual evaluation of a mapping sample and the generalization of the results.

Closer to our work, Sabou and Gracia [23] raise the issue of evaluating non-equivalence links, pointing out that several systems also produce subsumption and disjointness links. In particular, they discuss the issue of evaluating a mapping that contains redundant links, that is, links that can be logically derived from the others in the mapping. They compute precision both for the original set and the set from which the redundant links are removed. We extend and improve their conclusions.

3 Computing Precision and Recall

Golden standards are fundamental for computing the well-known precision and recall measures [4]. Typically, hand-made positive (GS^+) and negative (GS^-) golden standards contain links considered correct and incorrect, respectively. Ideally, GS^- complements GS^+ , leading to a precise evaluation. Yet, annotating all links in big datasets (with thousands or millions of links) is impossible and therefore the golden standard is often composed of three sets:

GS^+ the set of links considered correct;

GS^- the set of links considered incorrect;

Unk the pairs of nodes for which the semantic relation is unknown.

If we denote the result of the matcher (the mapping) with Res , precision and recall can be computed as follows [14]:

$$Precision = \frac{TP}{TP + FP} = \frac{|Res \cap GS^+|}{|Res \cap GS^+| + |Res \cap GS^-|} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} = \frac{|Res \cap GS^+|}{|GS^+|}, \quad (2)$$

where:

TP (True Positives) is the set of links found by the algorithm that hold;

FP (False Positives) is the set of links found by the algorithm that do not hold;

FN (False Negatives) is the set of links that hold, but which were not found by the algorithm.

For the cases in which GS^- is not available, precision can be approximated as follows:

$$Precision = \frac{|Res \cap GS^+|}{|Res|} \quad (3)$$

¹ This is an extended version of the TechRep DISI-09-046, also presented as a poster at the ISWC Ontology Matching Workshop (OM 2009), 25th October 2009, Washington DC, USA

² University of Trento, Italy, email: surname@disi.unitn.it

³ <http://oaei.ontologymatching.org/>

These sets are illustrated in Figure 1. For example, if for sake of simplicity we use numbers to indicate links, we could have:

$$\begin{aligned} Res &= \{1, 2, 3, 4\} & Unk &= \{\} \\ GS^- &= \{3, 4, 6, 8\} & GS^+ &= \{1, 2, 5, 7, 9, 10\} \\ Precision &= \frac{2}{(2+2)} = 0.5 & Recall &= \frac{2}{6} = 0.33 \end{aligned} \quad (4)$$

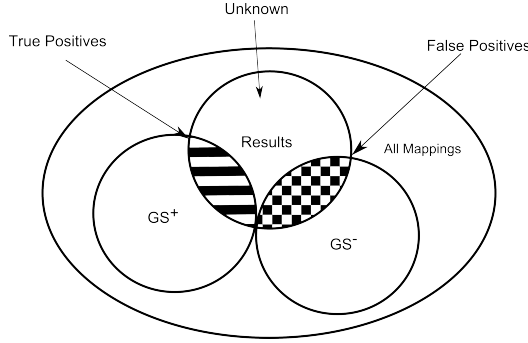


Figure 1. True Positives, False Positives and Golden Standards

The precision gives an indication of the amount of noise that is retrieved by the matching algorithm (how many correct links it returns) while the recall is a measure of the coverage of the algorithm (how many links the algorithm found and missed).

3.1 Coverage of the Golden Standard

Given two ontologies of size n and m , the size of a mapping and the golden standards range in $[0, n \times m]$. To enable precise computation of precision and recall, one should inspect all $n \times m$ combinations of nodes and consider all possible semantic relations that can hold between them. For large ontologies this is practically impossible. The huge effort required for their construction is the main reason why only a few golden standards are available and evaluation campaigns tend to use very small ontologies, risking a loss of statistical significance of the results and biasing towards one algorithm or the other.

When setting up exhaustive GS^+ and GS^- is not possible, the common practice is to inspect only a subset of the $n \times m$ node pairs [2, 14]. Partial coverage leads to an approximated evaluation. In particular, we cannot say anything about the subset $Res \cap Unk$ of the links. However, if GS^+ and GS^- are sampled properly, the precision and recall should still be evaluated in a statistically significant manner. For example, we could have reduced coverage compared to the previous example, as follows:

$$\begin{aligned} Res &= \{1, 2, 3, 4\} & GS^+ &= \{1, 2, 7\} & GS^- &= \{3, 6, 8\} \\ Unk &= \{4, 5, 9, 10\} & Res \cap Unk &= \{4\} \\ Precision &= \frac{2}{(2+1)} = 0.66 & Recall &= \frac{2}{3} = 0.66 \end{aligned}$$

As it turns out (compare the measures with those in (4)), such evaluations may be very different from the real values, therefore:

Recommendation 1. Use large golden standards. Include GS^- for a good approximation of the precision and recall. To be statistically significant, cover in GS^+ and GS^- an adequate portion of all node pairs.

In a sampled GS, reliability of results depends on: (a) the portion of the pairs covered by the golden standard; (b) the ratio between the size of the GS^+ and the GS^- and (c) their quality (see Section 4).

3.2 Comparing Semantic Relations

Current state of the art tools output different kinds of relations. While most of the matching tools, such as Similarity Flooding [21], Cupid [19] and COMA [7] only produce *equivalence*, some tools, such as AROMA [5] also produce *less general* and *more general* relations. At the best of our knowledge only ctxMatch [3], S-Match [12], Min S-Match [11] and Spider [23] also produce explicit *disjointness*.

Currently, different tools are usually compared without distinguishing among the different semantic relations produced and only the presence or absence of a relation between a pair of nodes is evaluated. This means, for instance, that subsumption and equivalence are considered as the same. This approach can be used to compare heterogeneous mappings, but leads to imprecise results.

A particular discourse has to be made for *disjointness* relations. Typically disjointness links are seen as a negative result, that is, a clear indication of two completely unrelated nodes. Thus, the majority of matching tools do not consider them interesting to the users. As a consequence they do not compute them at all, but corresponding node pairs are rather put in the GS^- . Moreover, they are often confused with *overlap*.

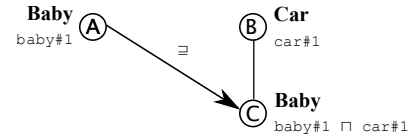


Figure 2. Overlap between nodes A and B. Natural language labels are in bold with a corresponding DL formulas under them.

Consider the example in Figure 2. The link $\langle A, C, \sqsupseteq \rangle$ is a correct result and as such should be part of the GS^+ . In fact, given the semantics of lightweight ontologies [16], the meaning of the node C includes the meaning of the node B above it. What about the relation between A and B then? They are not disjoint as they share C. The relation is rather an *overlap* (namely $A \cap B \neq \emptyset$).

Discriminating the two cases above is fundamental both to conclude the right relations between the nodes and to correctly evaluate precision and recall of disjointness relations when they are explicitly computed by the matching tool. In fact, the main problem is that negative golden standards (when available) typically contain undifferentiated links. For instance, the authors of [14] make no difference between *disjointness* and *overlap* relations. To the best of our knowledge, no evaluations take *disjointness* and *overlap* relations into account when measuring precision and recall. To summarize:

Recommendation 2. When presenting evaluation results, specify whether and how the evaluation takes into account the kinds of the semantic relations.

4 Maximized and Minimized Golden Standards

We use the notion of minimal mapping [11] to judge the quality of a golden standard. The basic idea is that among all possible links between two ontologies there are some redundant ones, which can be logically inferred from the others. The minimal mapping is defined

as the minimal set of (non-redundant) links such that all the other (redundant) links can be logically inferred from the non-redundant ones [11, 13]. We use the **Min(mapping)** function to remove the redundant links from the mapping (producing the *minimized mapping*) and the **Max(mapping)** function to add all the redundant links (producing the *maximized mapping*).

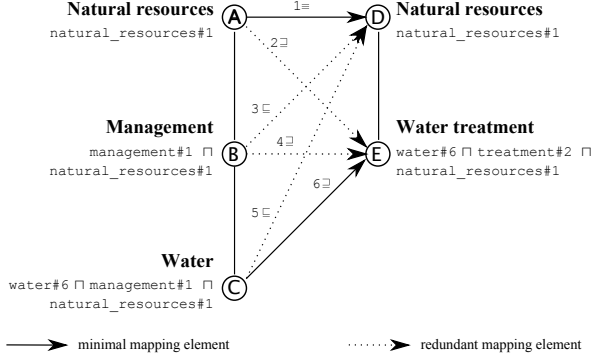


Figure 3. The mapping between two lightweight ontologies. Original natural language labels are in bold.

Consider the example in Figure 3 taken from [13]. It shows the minimal mapping (the solid arrows) and the mapping of maximum size (including the maximum number of redundant links, represented as dashed arrows) computed between two lightweight ontologies.

In the following we provide three observations.

The first observation is that following [11] and staying within lightweight ontologies guarantees that the maximized mapping is always finite and thus corresponding precision and recall can always be computed.

The second observation is that, in contrast with [23], we argue (and show with an example) that comparing the minimized versions of the mapping and the golden standards is not informative. The reason is that minimization can significantly reduce the amount of links in their intersection. In other words, they can share a few non-redundant links still generating a significant amount of redundant links in common. Notice that different non-redundant links can generate the same redundant links.

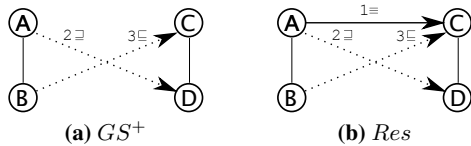


Figure 4. Minimization changing precision and recall

Consider the examples in the Figure 4. Suppose that all the displayed links are correct. Notice that links 2 and 3 follow from link 1. Suppose that our golden standard (Figure 4a), as it often happens with large datasets, is incomplete (it contains only the links 2 and 3, while link 1 is unknown) and thus we use precision formula (3) which returns an approximated value. Suppose that the matcher, being good enough, finds all displayed links (Figure 4b). By computing the precision and recall figures first on the original and then on the

minimized versions of the mapping and the golden standard we obtain:

$$\begin{aligned} GS^+ &= \{2, 3\} \quad Res = \{1, 2, 3\} \\ Precision &= 0.66 \quad Recall = 1 \\ Min(GS^+) &= \{2, 3\} \quad Min(Res) = \{1\} \\ Precision &= 0 \quad Recall = 0 \end{aligned} \quad (5)$$

Compare the normal situation (5) with (6) that shows the situation when minimized sets are used to calculate precision and recall figures. From this example we see that precision and recall figures computed on the minimized versions are far from the real values and are unreliable.

Our last observation is that using maximized sets gives no preference to redundant or non-redundant links and leads to more accurate results. In particular, recall figure better shows the amount of information actually found by the system. If we maximize the sets we also decrease the number of unknown links and therefore we obtain a more accurate result.

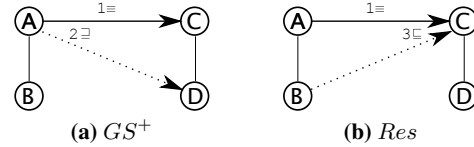


Figure 5. Maximization changing precision and recall

Consider now the example in Figure 5. The precision and recall figures are given in (7) for the original sets and in (8) for the maximized ones.

$$\begin{aligned} GS^+ &= \{1, 2\} \quad Res = \{1, 3\} \\ Precision &= 0.5 \quad Recall = 0.5 \\ Max(GS^+) &= \{1, 2, 3\} \quad Max(Res) = \{1, 2, 3\} \\ Precision &= 1 \quad Recall = 1 \end{aligned} \quad (7)$$

Maximizing a golden standard can also reveal some unexpected problems and inconsistencies. For instance, we can discover that even if GS^+ and GS^- are disjoint, $Max(GS^+)$ and $Max(GS^-)$ are not, namely, $Max(GS^+) \cap Max(GS^-) \neq \emptyset$. During our experiments with the TaxME2 golden standard [14], we discovered that there are two links in the intersection of GS^+ and GS^- and 2187 in the intersection of their maximized versions.

We conducted several experiments to study the differences between precision and recall measures when comparing the minimized and maximized versions of the golden standards with the minimized and maximized versions of the mapping returned by S-Match [15].

We used three different golden standards [1] already used in several evaluations. The first two datasets come from OAEI; they describe publications, contain few nodes and corresponding golden standard is exhaustive. It only contains equivalence links. The second two come from the arts domain and the golden standard is crafted by experts manually. The third two datasets have been extracted from the Looksmart, Google and Yahoo! web directories. The golden standard is part of the TaxME2 and is extensively described in [14]. Unfortunately, all these golden standards suffer to a certain degree from

Table 1. Precision and Recall for minimized, normal, and maximized sets

Dataset pair	Precision, %			Recall, %		
	min	res	max	min	res	max
101/304	32.47	9.75	69.67	86.21	93.10	92.79
Topia/Icon	16.87	4.86	45.42	10.73	20.00	42.11
Source/Target	74.88	52.03	48.40	10.35	40.74	53.30

the problems described in the previous sections, thus the measures obtained must be considered as indicative.

Table 1 contains precision and recall figures calculated using standard precision and recall formulas (1) and (2). For the cases where no GS^- is provided, (3) is used instead of (1). In particular, these figures are the result of the comparison of the minimized mapping with the minimized golden standards (min), the original mapping with the original golden standards (res) and the maximized mapping with the maximized golden standards (max) respectively. As it can be noted from the measures obtained comparing the maximized versions with the original versions, the performance of the algorithm is on average better than expected. To summarize:

Recommendation 3. To obtain accurate measures it is fundamental to maximize both the golden standard and the matching result.

5 Conclusions and future work

In this paper, we proposed some recommendations to follow when building golden standards and to effectively use them to evaluate matching algorithms. Following these recommendations will make the evaluation and comparison of different algorithms more accurate.

We also discussed the issue of evaluating mappings with disjointness. In the current state of the art, no golden standard is available that explicitly provides true disjointness links. Moreover, disjointness is often confused with overlap. Thus, it is currently impossible to evaluate the performance of such algorithms.

In the future we will explore how the size of the golden standard influences the evaluation and how large should be the part covered by GS^+ and GS^- to be statistically significant as well as describe the methodology for evaluating rich mappings by supporting our recommendations with further experimental results.

6 Acknowledgments

This work has been partially funded by the FP7 Living Knowledge FET IP European Project.

REFERENCES

- [1] Aliaksandr Autayeu, Vincenzo Maltese, and Pierre Andrews, ‘Best practices for ontology matching tools evaluation’, Technical report, University of Trento, DISI, (2009).
- [2] Paolo Avesani, Fausto Giunchiglia, and Mikalai Yatskevich, ‘A large scale taxonomy mapping evaluation’, in *Proc. of ISWC*, pp. 67–81, (2005).
- [3] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini, ‘Semantic coordination: A new approach and an application’, in *Proc. of ISWC*, pp. 130–145, (2003).
- [4] Jérôme David and Jérôme Euzenat, ‘On fixing semantic alignment evaluation measures’, in *Proc. of the Third International Workshop on Ontology Matching*, (2008).
- [5] Jérôme David, Fabrice Guillet, and Henri Briand, ‘Association rule ontology matching approach’, *International Journal on Semantic Web and Information Systems*, **3**(2), 27–49, (2007).
- [6] Hong Hai Do, Sergey Melnik, and Erhard Rahm, ‘Comparison of schema matching evaluations’, in *Proc. of the 2nd International Workshop on Web Databases*, pp. 221–237, (2002).
- [7] Hong Hai Do and Erhard Rahm, ‘Coma — a system for flexible combination of schema matching approaches’, in *VLDB*, pp. 610–621, (2002).
- [8] Marc Ehrig and Jérôme Euzenat, ‘Relaxed precision and recall for ontology matching’, in *Proc. of Integrating Ontologies Workshop*, (2005).
- [9] Jérôme Euzenat, ‘Semantic precision and recall for ontology alignment evaluation’, in *Proc. of IJCAI*, pp. 348–353, (2007).
- [10] Jérôme Euzenat and Pavel Shvaiko, *Ontology Matching*, Springer-Verlag, 2007.
- [11] Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu, ‘Computing minimal mappings’, Technical report, University of Trento, DISI, (2008).
- [12] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich, ‘Semantic schema matching’, in *Proc. of CoopIS*, pp. 347–365, (2005).
- [13] Fausto Giunchiglia, Dagobert Soergel, Vincenzo Maltese, and Alessandro Bertacco, ‘Mapping large-scale knowledge organization systems’, in *Proc. of ICSD*, (2009).
- [14] Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko, ‘A large dataset for the evaluation of ontology matching systems’, *The Knowledge Engineering Review Journal*, **24**, 137–157, (2008).
- [15] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko, ‘Semantic matching: algorithms and implementation’, in *Journal on Data Semantics*, **IX**, (2007).
- [16] Fausto Giunchiglia and Ilya Zaihrayeu, ‘Lightweight ontologies’, in *Encyclopedia of Database Systems*, 1613–1619, (2009).
- [17] Antoine Isaac, Shenghui Wang, Claus Zinn, Henk Mattheizing, Lourens van der Meij, and Stefan Schlobach, ‘Evaluating thesaurus alignments for semantic interoperability in the library domain’, *IEEE Intelligent Systems*, **24**(2), 76–86, (2009).
- [18] Siddharth Kaza and Hsinchun Chen, ‘Evaluating ontology mapping techniques: An experiment in public safety information sharing’, *Decision Support Systems*, **45**(4), 714–728, (2008).
- [19] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm, ‘Generic schema matching with cupid’, in *VLDB*, pp. 49–58, (2001).
- [20] Christian Meilicke and Heiner Stuckenschmidt, ‘Incoherence as a basis for measuring the quality of ontology mappings’, in *Proc. of the 3rd International Workshop on Ontology Matching*, (2008).
- [21] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm, ‘Similarity flooding: A versatile graph matching algorithm and its application to schema matching’, in *Proc. of ICDE*, (2002).
- [22] Natalya Fridman Noy and Mark A. Musen, ‘Evaluating ontology-mapping tools: Requirements and experience’, in *Proc. of OntoWeb-SIG3 Workshop*, pp. 1–14, (2002).
- [23] Marta Sabou and Jorge Gracia, ‘Spider: Bringing non-equivalence mappings to OAEI’, in *Proc. of the Third International Workshop on Ontology Matching*, (2008).
- [24] Willem Robert van Hage, Antoine Isaac, and Zharko Aleksovski, ‘Sample evaluation of ontology-matching systems’, in *Proc. of EON*, pp. 41–50, (2007).

Exploring Ontology Merging with examples

Liwei Deng¹

Abstract. We review some recent work on ontology merging using Category Theory and explore some simple problematic examples.

1 INTRODUCTION

We review some recent work on ontology merging; use the techniques presented in these to merge non-trivial ontologies and in the process analyse both the methods and the ontologies themselves.

We are interested in ontology management using a Category Theoretic framework; the origin of such an approach can be traced back to [5], which at the end outlines a work program for its realization. Progress on this will be reviewed here.

Before that, it will be useful to have a review of all the main ontology operation terms used by the papers cited in this report, since sometimes the same term is used by different authors to mean different things, and some authors give more precise definitions than others.

The techniques mentioned above will enable us to compute the merge of non-trivial ontologies, to give us an idea of how they work in practice. One of the merges will be from [5] and [3]. It seems to have some problems (pointed out in [3]) and is analysed further in an attempt to understand the cause of the problem, and possibly find remedies.

Another example, from a completely different source (a talk by Kai-Uwe Kuehnberger), besides being a good test for merging, can also be easily extended and refined, thus allowing for more rigorous tests of current techniques and tests of more powerful methods in the future.

2 DEFINITIONS OF SOME ONTOLOGY OPERATIONS

A survey of ontology mapping [1], gives definitions to many ontology operation terms. However at least two of the papers cited in this report, [6] and [2], are published after [1]; they used old terms with similar meanings while introducing new, mostly related ones.

This ever growing collection of related ontology operations terms is on the threshold of becoming a non-trivial ontology in itself. They include mapping, translation, alignment, articulation, merging, connection, integration and refinement. It is useful, for both understanding what follows in this report and generally, to review these terms and explain their relations.

3 ONTOLOGY MERGING, V-ALIGNMENT, HETS

In [5], an approach to ontology merging using Category Theory was presented; it is argued that this approach does not depend on a par-

ticular specification, therefore achieving a greater level of generality.

To illustrate this approach, it gave a definition of the merge of two ontologies as the result of a pushout operation.

This is generalized and extended in [6] with the notion of a V-alignment, with a set of well defined operations for ontology merging and alignment operations.

The V-alignment is placed in a yet more general framework in [2] (though in this case not totally rigorously), through which it can be practically implemented in the Heterogeneous Tool Set (HETS) (see [4]).

4 ONTOLOGY MERGING EXAMPLES

4.1 Ontology merging in literature

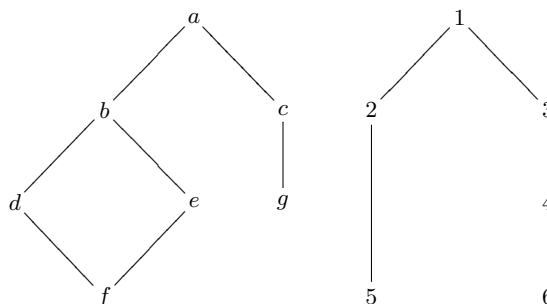
In the literature there seems to be relatively few examples of merging of non-trivial ontologies using the framework above (particularly the Category Theoretic one).

[6] gives a very simple example illustrating problems of using the V-alignment; [2] gives the specification of a W-alignment (see [6]), which is made out of V-alignments and is more difficult to automate, in the appendix.

It seems natural to try out the more basic construction first, which is what we will do later. As already mentioned in the previous section, we will implement these examples in HETS, along the lines set out in [2].

4.2 A problematic example

[5] gave an example with the following two partial orders, with an ontology mapping of $\{(a, 1), (b, 2), (c, 4), (f, 5), (g, 3)\}$. They are then merged with a pushout operation.

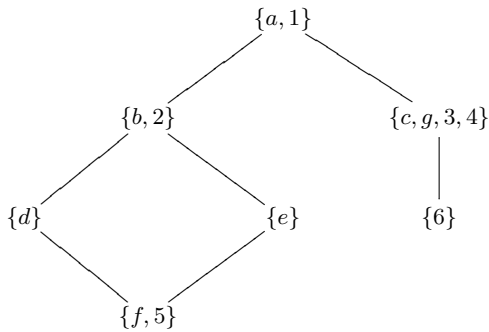


In [3], the elements in the example of [5] were instantiated by the author, which then obtained a merged ontology with the same structure. We are unhappy with the instantiation since it is of no help in

¹ University of Edinburgh, UK, email L.Deng-2@sms.ed.ac.uk

illustrating the point to come, but rather confuses the issue by introducing problems with interpretation of the terms used; therefore they are not included here.

The merged ontology in [5] is shown below.



In the merged ontology there is an element, $\{c, g, 3, 4\}$ that contained four elements of the original ontology (two from each); whereas intuition would suggest elements of the merged ontology should contain an equivalent pair of elements or singleton elements without equivalence, from the original ontologies.

This is pointed out in [3], where it's taken as evidence that this approach is deficient. We do not believe the problem lies with the approach, but rather the failure indicates a problem with either the original ontologies or the ontology mapping used to compute the merge. (This is in fact hinted at in [5] at the end of the 4th step of the 'How to put our approach into practice' section, even though where the example was presented the authors didn't seem to think it was problematic.) This belief is investigated, with a view to fixing any errors detected.

4.3 A more complicated example

Finally, we finish with a more complicated example (this is from a talk by Kai-Uwe Kuehnberger at University of Edinburgh). This example has the advantages of being natural and from a third party not directly concerned about ontology alignments and merging.

We note that there is an obvious analogy between an atom and the solar system (numerous small bodies orbiting a large core); this is formalized in Kuehnberger. At a crude level, the analogy works well and we expect ontologies formulated at this level to be very similar, and so the merge will be very similar to the originals.

But if we refine or extend the ontologies we will get big differences, with the intuitive merge more dissimilar to the original ontologies; so this provides a tougher test of the categorical techniques, and further techniques developed in the future.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their comments.

REFERENCES

[1] Yannis Kalfoglou and Marco Schorlemmer., 'Ontology mapping: The state of the art', *Yannis Kalfoglou, Marco Schorlemmer, Amit Sheth, Stephen Staab, and Michael Uschold, editors, Semantic Interoperability and Integration, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl*, (2005).

[2] Lucke Dominik Kutz, Oliver and Till. Mossakowski, 'Heterogeneously structured ontologies: Integration, connection and refinement', *Knowledge Representation Ontology Workshop, KR-08*, (2008).
 [3] Roko. Mijic, *Representing knowledge about the real world: probabilistic, logical and categorical techniques*, Master's thesis, School of Informatics, University of Edinburgh, 2009.
 [4] Maeder Christian Mossakowski, Till and Klaus Luttich, *Lecture Notes in Computer Science*, volume 4424, chapter The Heterogeneous Tool Set, 519–522, Springer-Verlag Heidelberg, 2007.
 [5] Marc Ehrig Pascal Hitzler, Markus Kr?tzsch and York Sure., 'What is ontology merging? - a category theoretic perspective using pushouts', *Proc. First International Workshop on Contexts and Ontologies: Theory, Practice and Applications (CO)*, 104–107, (July 2005).
 [6] Krotzsch M. Euzenat J. Zimmermann, A. and P. Hitzler, 'Formalizing ontology alignment and its operations with category theory', *Proceeding of the 2006 Conference on Formal Ontology in Information Systems*, 277–288, (2006).

Automatizing the Evaluation of Model Matching Systems

Kelly Garces and Wolfgang Kling and Frédéric Jouault¹

Abstract. Model-Driven Engineering (MDE) and the Semantic Web are valuable paradigms. This paper sketches how to use a scientific and technical result around MDE in the Semantic Web. The work proposes a mechanism to automatize the evaluation of model matching algorithms. The mechanism involves *megamodeling* and a Domain Specific Language named AML (AtlanMod Matching Language). AML allows to implement matching algorithms in a straightforward way. We present how to adapt the mechanism to the ontology context, for example, to the Ontology Alignment Evaluation Initiative (OAEI).

1 Introduction

Over the last decade, whereas the software engineering community has advanced the Model-Driven Engineering (MDE) paradigm, the Web, AI, and database communities have promoted the Semantic Web. MDE suggests to develop a model of the system under study, which is then transformed into an executable software entity [12]. The Semantic Web, in turn, aims to express Web information in a precise, machine-interpretable form, ready for software agents to process [12].

Even though MDE has foundations and applications different from the Semantic Web's, researches have raised the question: how the scientific and technical results around MDE can be used productively in the Semantic Web, and vice versa? [21] elucidates the potential of ontologies technologies in MDE. Given a classical MDE transformation scenario, [21] proposes further transformations into the ontology working context. The idea is to enable model checking, logics-based model analysis, and reasoning. [12], in turn, details how to use MDE for ontology development on the Semantic Web. The authors suggest MDE to automate the semantic markup of Web resources which has been done more or less manually.

Just as [12], we investigate how MDE can contribute to ontology development. In particular, our work is about matching systems evaluation. The Semantic Web defines *matching* like an operation computing alignments between ontologies. Alignments are the more longstanding solution to the interoperability problem. Because the number of matching systems is rapidly increasing, it is necessary to pursue efforts on an extensive evaluation [9].

To illustrate what a matching system evaluation consists of, let us consider to the Ontology Alignment Evaluation Initiative (OAEI). Every year since 2004, the OAEI develops a campaign that evaluates ontology matching systems presented by participants. Fig. 1 illustrates a classical evaluation. The matching system f takes as input two ontologies (o and o'), an initial alignment (A), parameters (p), and resources (r), and yields as output a new alignment (A'). R is a reference alignment (or *gold standard*), the one that participants must

find. The evaluation mechanism calculates matching metrics (M) by comparing R to each A' . The OAEI uses such metrics to figure out strengths and weaknesses of proposed matching systems.

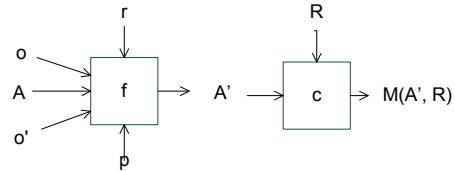


Figure 1. Ontology matching systems evaluation (Adapted from [19])

Each campaign involves three phases: preparation, execution, and evaluation. During the preparation phase, the OAEI defines a set of test cases to compare matching systems and algorithms on the same basis. A given test case includes ontologies (coming from different domains) and their corresponding alignments. In the execution phase, the participants use their algorithms to match all test cases. Finally, the OAEI organizers check the results obtained in the evaluation phase.

Inspired from the OAEI framework, we have developed a mechanism that automatizes the evaluation of model matching systems. Instead of ontologies, our mechanism involves systems that match metamodels (i.e., formalisms to represent models). In addition, the mechanism itself is based on MDE.

In a nutshell, our approach tackles two issues of matching algorithm evaluation.

1. **Not enough test cases.** Whereas it is relatively easy to pairs of metamodels, the availability of reference alignments is restricted. Instead of asking experts for reference alignments, our mechanism automatically extracts them from model transformations.
2. **Low evaluation efficiency.** Evaluation is a time-consuming task. By using a *megamodel* (i.e., a map), our mechanism executes matching algorithms over test cases in an automatic way. Such automatization can reduce the time spent during evaluations, and increase the confidence on results.

We believe that these problems are relevant to ontology matching systems evaluation too. The aim of this paper is to present the whole mechanism and draw out how the Semantic Web community can take advantage of it. Besides megamodels, our approach uses AML programs. AML (AtlanMod Matching Language) is a Domain Specific Language (DSL) to implement matching algorithms. In an experimentation, our mechanism evaluates an AML program over 30 test cases in 40 minutes.

The paper structure is as follows: Section 2 presents an overview of the mechanism. The subsequent three sections describe the mechanism in detail. Each of them briefly outlines how to integrate our

¹École des Mines de Nantes Email: kelly.garces, wolfgang.kling, frederic.jouault@emn.fr

approach to the ontology context. Section 6 shows and discusses the results of applying the approach. Section 8 concludes the paper.

2 Mechanism Overview

Fig. 2 shows the artifacts manipulated by our mechanism along the evaluation phases. The artifacts are stored in a model repository.

- **Preparation.** The mechanism constitutes a given test case using two metamodels, m and m' , and a transformation $tm2m'$ written in terms of them. A reference alignment R is extracted from $tm2m'$ (block e in Fig. 2).
- **Execution.** We develop a matching algorithm f with AML.
- **Evaluation.** The mechanism executes f which computes a candidate alignment A' from m , m' , A , r , and p . Finally, the block c compares A' to R and derives matching metrics $M(A', R)$.

A megamodel is a kind of map of a model repository. Our approach uses a megamodel to constitute many test cases, launch a set of matching algorithms, and compute metrics. Next sections describe our mechanism in detail.

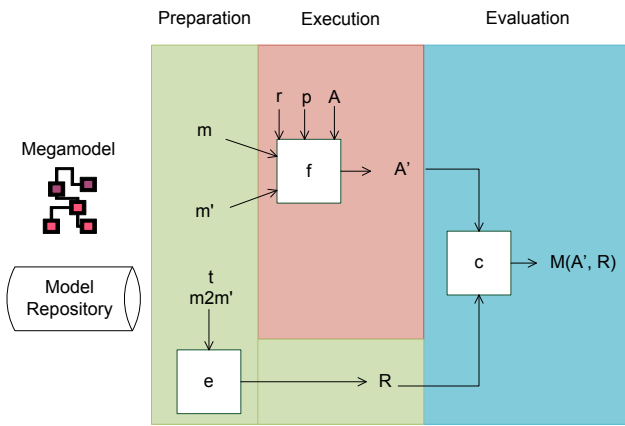


Figure 2. A mechanism to automatize matching system evaluation

3 Preparation: Getting Test cases from Model repositories

In this phase, our mechanism prepares a set of test cases. Each test case consists of a pair of metamodels and a reference alignment.

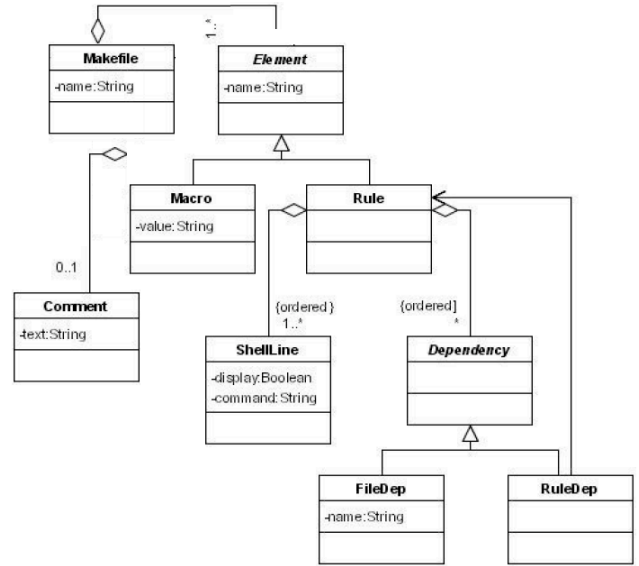
3.1 Discovering test cases

The megamodel is the cornerstone of our approach, it provides a roadmap of all the modeling artifacts stored in a repository, as well as, the relations between them [23]. Note that in MDE everything is a model (metamodels, transformations, etc). Before working with the megamodel it has to be populated. Since a model repository is continuously growing, we have implemented a strategy to automatically populate the megamodel. The strategy consists of the following steps:

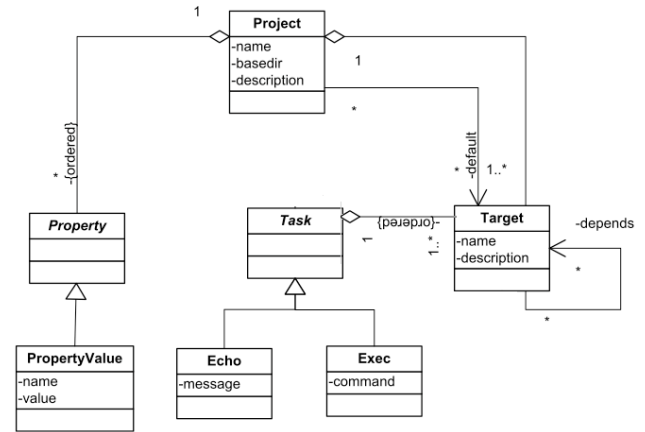
1. A program parses a set of files containing metadata. The metadata describes, for example, in terms of what metamodels a transformation is written. The output of the parsing is a text file.
2. A textual syntax tool [6] translates the text file into a megamodel.

3. Because the megamodel refers to more models than the ones we are interested in, a program refines the megamodel to keep only the records related to transformations and their corresponding metamodels.

Fig. 3 shows the metamodels of an extracted test case. In general, the elements of a metamodel are classes and structural features (i.e., attributes, and references). For instance, the Ant metamodel contains the class `Project`, the attribute `message`, and the reference `default`.



(a) Make metamodel



(b) Ant metamodel

Figure 3. Metamodels of a test case

3.2 Extracting reference alignments from transformations

Having the megamodel, the mechanism extracts reference alignments from transformations. A transformation is a program that translates models, conforming to source metamodels, into models, conforming to target metamodels. A transformation may be available

in various flavors, i.e., being developed by programmers geographically distributed, or using model transformation languages different each other. Thus, we should define the extraction scope by answering the questions: 1) in what languages the available transformations are developed?, and 2) what patterns are embedded in them?

For example, we have found the following patterns in ATL² transformations (Listing. 1 shows a concrete example):

- Each transformation contains a set of rules (lines 1-9).
- Each rule consists of a pair of patterns, i.e., `inPattern` and `outPattern`. The `inPattern` matches a class of m (line 3), and the `outPattern` a class (or set of classes) of m' (line 5).
- Each `outPattern` is composed of a set of bindings. A binding initializes a `rightClass` property using a structural feature of `leftClass`. We have characterized four types of bindings:
 1. `rightStructuralFeature` $\leftarrow$ a `leftStructuralFeature` (line 6).
 2. `rightStructuralFeature` $\leftarrow$ an OCL [20] iteration expression which includes a `leftStructuralFeature` (line 7).
 3. `rightStructuralFeature` $\leftarrow$ an operation (i.e., a helper expression) involving a `leftStructuralFeature`.
 4. `rightStructuralFeature` $\leftarrow$ a variable referring to another `rightClass` created in the `outPattern`.

Note that the more we figure patterns out, the better is the quality of reference alignments. The quality can be further increased by extracting alignments not only from one transformation but also from a set of transformations $tm2m'$ if they are available. Alignments conform to one extension of the AMW core metamodel [2]. AMW (AtlanMod Model Weaver) is a tool to manipulate model-based alignments.

Listing 1. Excerpt of the transformation `Make2Ant`

```

1 rule Makefile2Project{
2   from
3     m : Make!Makefile
4   to
5     a : Ant!Project {
6       name      <- m.name,
7       targets   <- m.elements -> select(c | c.
8         oclIsKindOf(Make!Rule))
9     }

```

There are ways to bring the gap between modeling artifacts and ontologies. Tools like EMFTriple [13] can translate metamodels to ontologies. A transformation moreover can transform our alignments to a more standard alignment format (e.g., the Alignment API [8]).

4 Execution: Implementing and Testing Matching Algorithms with AML

The execution phase typically involves implementation, testing, and deployment³ of matching algorithms. In our approach, the execution only embraces development and testing. We drop the deployment to the evaluation phase.

Instead of GPLs, we propose a DSL to implement and test matching algorithms. In general, DSLs improve programmers productivity and software quality. Often, this is achieved by reducing the cost of initial software development as well as maintenance costs. The

²The AtlanMod Transformation Language [14].

³We call deployment the stage that executes algorithms in order to produce the final matching results.

improvements - programs being easier to write and maintain - materialize as a result of domain-specific guarantees, analysis, testing techniques, verification techniques, and optimizations [18].

According to [22], the matching operation has been investigated from the early 1980s. We promote the use of a DSL gathering all the expertise gained over the last twenty years. The DSL has to capture a significant portion of the repetitive tasks that an expert needs to perform in order to produce an executable matching algorithm.

A first approximation to such DSL is the AtlanMod Matching Language (AML) which is based on the MDE paradigm [11]. AML instruments the fundamental behind the most recent matching algorithms: to assemble multiple components each one employing a particular matching technique [15]. Internally, an AML algorithm is a chain of model transformations, each one implementing a matching technique. Input/output flow along the chain as models.

Roughly speaking, an AML program is a text file containing three sections [11]:

1. **Import section** enables reusing of pre-existing AML algorithms.
2. **Matching rule declaration** allows to specify a concrete matching technique. This conforms to one of 5 kinds of techniques:
 - **Creation** establishes an alignment between the element a (in metamodel m) and the element b (in metamodel m') when these elements satisfy a condition.
 - **Similarity** computes a similarity value for each alignment prepared by the creation techniques. One function establishes the similarity values by comparing particular metamodel aspects: labels, structures, and/or data instances.
 - **Aggregation** combines similarity values by means of one expression. An expression often involves:
 - n , the number of matching techniques providing alignments.
 - $\sigma_i(a, b)$ similarity value computed by the matching technique i
 - w_i weight or importance of the matching technique i , where $\sum_{i=1}^n w_i = 1$
 - **Selection** selects alignments whose similarity values satisfy a condition, e.g., thresholding.
 - **User-defined** represents matching techniques capturing functionality beyond our classification.
3. **Model flow block** specifies how several matching techniques interact to each other to deliver alignments.

Listing. 2 shows the excerpt of an AML algorithm named *LevenshteinBothMaxSim*. For the sake of brevity we present the model flow block and omit matching rule declarations⁴. *LevenshteinBothMaxSim* involves the following number of matching rules: 3 creation, 1 label-based similarity, 1 structure-based similarity, 1 aggregation, 2 selection, and 1 user-defined. They have been inspired by [3][4][16][9]. We have chosen such techniques because they report good results. Below we describe their functionality:

Creation *TypeClass*, *TypeReference*, and *TypeAttribute* create alignments between two metamodel elements conforming to the same metamodel type (class, reference or attribute).

⁴The reader interested in that may check [10]

Similarity

- *Levenshtein* returns a value depending on the edition operations that should be applied to a label to obtain the other one. *Levenshtein* reuses the SimMetrics API [1].
- *Propagation*, *SF*, and *Normalization* instrument the algorithm Similarity Flooding described in [17]. This algorithm strengthens previously computed similarities based on the metamodel structure. *Propagation* associates two alignments (*I1* and *I2*) if there is a relationship between the linked elements, for example, *I1* links classes, and *I2* links attributes, classes contain attributes. *SF* propagates the similarity value from *I1* to *I2* because of their relationship, e.g., containment. *Normalization* divides all the similarity values by the maximal similarity value.

Aggregation The *Merge* matching technique puts together the alignments returned by other techniques. We have implemented it by means of an AML aggregation construct.

Selection *BothMaxSim* selects a given alignment if its similarity value is the highest among the values of alignments of two sets (e.g., *leftSet* and *rightSet*). The *leftSet* contains all the alignments linking a left concept, and the *rightSet* the alignments linking a right concept. *ThresholdMaxDelta* moreover selects an alignment when its similarity satisfies the range of tolerance [*Threshold*–*Delta*, *Threshold*]. While *BothMaxSim* provides 1:1 alignments, *ThresholdMaxDelta* provides n:m. These matching techniques (along with the thresholds) have been borrowed from [3].

User-Defined Whereas *Propagation* levers the execution of *SF* (a similarity matching technique), *Propagation* is user-defined.

Listing 2. LevenshteinBothMaxSim model flow

```

1 modelsFlow{
2   tp = TypeClass[map]
3   typeRef = TypeReference[map]
4   typeAtt = TypeAttribute[map]
5
6   merged = Merge[1.0:tp, 1.0:typeRef, 1.0:typeAtt]
7
8   nam = Levenshtein[merged]
9
10  filtered = ThresholdMaxDelta[nam]
11  prop = Propagation[filtered]
12  sf = SF[filtered](prop)
13  norm = Normalization[sf]
14
15  tmpresult = WeightedAverage[0.5 : norm, 0.5:nam]
16
17  result = BothMaxSim[tmpresult]
18 }
```

Besides the techniques mentioned above, we have implemented other sophisticated heuristics, for example, *MSR* (Measures of Semantic Relatedness), which extracts semantic similarity between two labels based on a large corpora, e.g., Google or Wikipedia. Besides creation techniques, our library has 10 techniques⁵.

AML is build on top of ATL, one of the most popular transformation language. AML inherits the ATL declarative nature which allows to straightforwardly implement matching rules. In addition, the model flow block enables to easily assembly matching techniques and tune algorithms. With respect to testing features, users can verify the correctness of alignments by displaying them in the AMW graphical interface [2]. AML moreover provides a functionality that compares computed alignments to gold standards and gives the results in HTML format.

⁵See <http://wiki.eclipse.org/AML>

Note that AML allows to develop user-defined matching techniques in ATL (to go beyond pre-established DSL semantics) and even invoke Java code. Thus, it is possible to integrate the vast set of matching algorithms developed in the past.

5 Evaluation: Deploying and Assessing AML Algorithms

Based on a megamodel, our mechanism automatically deploys algorithms and computes metrics. The megamodel indicates the algorithms to be executed, the required inputs, and the reference alignments. Our mechanism computes four matching metrics [9]: $Precision(R, A') = \frac{R \cap A'}{|A'|}$, $Recall(R, A') = \frac{R \cap A'}{|R|}$, $Fscore(R, A') = \frac{2 * Recall(A', R) * Precision(A', R)}{Recall(A', R) + Precision(A', R)}$, and $Overall(A', R) = Recall(A', R) * (2 - \frac{1}{Precision(A', R)})$. This moreover updates the megamodel with records associating A' , R , and $M(A', R)$.

The use of DSLs and megamodels facilitates the evaluation phase. Since there is an unique programming interface, launching the algorithms become easier. By using the megamodel, our mechanism can execute algorithms over a large set of test cases and to obtain results by itself. This promotes a more confident evaluation of performance and accuracy.

The ontology community has contributed several tools to compute sophisticated matching metrics (e.g., *PrecEvaluator* [19]). We can take advantage of these tools by translating our alignments to their formats.

6 Experimentation: Results and Discussion

The approach has been tested over the “ATL Transformation Zoo” [5]. This repository has so far 103 ATL transformation projects contributed by the m2m Eclipse community [7]. Our mechanism automatically populated the megamodel with 1064 entities, 185 of them are metamodels and 156 are transformations. It took around 20 minutes to build the megamodel.



Figure 4. Automatic modeling artifact discovery results

After filtering the megamodel, for obtaining only the data matter of our interest, we obtained 30 test cases. Fig. 5 shows the results of applying the *LevenshteinBothMaxSim* algorithm to the test cases. The algorithm spent 40 minutes matching the pairs of metamodels. Fig. 5 reports low fscores (lower than 0.5). There are two reasons behind these results:

1. **The suitability of AML algorithms.** The space of matching algorithm is quite large, and sometimes it is difficult to find an algorithm that accurately matches a certain pair of metamodels. Thus, it is quite natural that *LevenshteinBothMaxSim* provides good results for few pairs of metamodels. We can see better the accuracy of an algorithm as we compare its results to other algorithms’.
2. **The quality of reference alignments.** When transformations are complex or incomplete, our mechanism may not extract enough reference alignments.

- **Complex** means that a given transformation implements imperative patterns (rules with only `outPattern`). In contrast, our mechanism relies on declarative patterns (as described in Section 3.2, rules with `inPattern` and `outPattern`). One example is the transformation `Measure2Table` which contains 6 rules, all of them imperative. The complexity of this transformation explains why our mechanism does not obtain any reference alignment, and why the `fscore` is zero.
- **Incomplete** means that a transformation lacks items. The justification is that developers write transformations in terms of the data instances that they expect to have in the source and target models. If the data instances are not relevant, developers do not written rules associating certain metamodel concepts. Thus, even if an AML algorithm gives good results, it is possible to obtain low `fscores` when reference alignments are poor.

Reference Alignment	Pres	Rec	Overall	Fscore
XML2Ant	0,3	0,08	-0,11	0,13
KM32CONFATL	0,03	0,22	-6,89	0,05
OWL2XML	0,07	0,06	-0,76	0,06
A2B	0,5	0,5	0,0	0,5
Ant2XML	0,22	0,14	-0,37	0,17
Make2Ant	0,04	0,13	-2,69	0,06
XML2Make	0,13	0,07	-0,43	0,09
XHTML2XML	0,04	0,02	-0,48	0,03
Table2TabularHTML	0,11	0,25	-1,75	0,15
Measure2Table	0,0	0,0	0,0	0,0
XHTML2XML	0,04	0,02	-0,48	0,03
TypeA2TypeB_v4	0,8	0,67	0,5	0,73
ATOM2RSS	0,06	0,19	-2,71	0,09
ATOM2XML	0,05	0,1	-1,9	0,06
RSS2ATOM	0,06	0,22	-3,17	0,1
RSS2XML	0,04	0,17	-3,67	0,07
XML2ATOM	0,05	0,02	-0,46	0,03
XML2RSS	0,17	0,09	-0,36	0,12
TypeA2TypeB_v1	0,25	0,33	-0,67	0,29
TypeA2TypeB_v2	0,75	0,6	0,4	0,67
SimpleSBVR2SimpleUML	0,1	0,2	-1,6	0,13
Syntax2SimpleSBVR	0,46	0,61	-0,1	0,53
TT2BDD	0,4	0,43	-0,21	0,41
SVG2XML	0,08	0,08	-0,77	0,08
HTML2XML	0,1	0,1	-0,76	0,1
Table2TabularHTML	0,11	0,17	-1,17	0,13
Tree2List	0,33	0,33	-0,33	0,33
Tree2List_usingATLResolveAlgorithm	0,33	0,25	-0,25	0,29
OWL2XML	0,07	0,06	-0,76	0,06
XML2XSLT	0,67	0,29	0,14	0,4

Figure 5. Results

In the future, we wish to apply a larger set of AML algorithms to the test cases extracted from the ATL transformation Zoo. With respect to the quality of reference alignments, we think that it is acceptable: only one transformation out of thirty has produced an empty reference alignment, i.e., *Measure2Table*, the one containing imperative rules. In general, ATL transformations contain declarative patterns which allows to extract reference alignments much more easily. We believe that transformations are a good source of reference alignments. So that, we can get an intuition about matching algorithm accuracy as we use them.

7 Acknowledgments

The authors would like to thank the ANR project FLFS for the provided funding, as well as Jean Bézin and Pierre Cointe for discussions that have contributed to the formation of the views reflected in this paper.

8 Conclusion

We presented a mechanism to automatize the evaluation of model matching systems. Our early results are motivating. Firstly, the mechanism extracts complete test cases from open-source repositories. The number of candidate test cases raises every year. Secondly, the use of a DSL makes it possible to reuse matching techniques. Thus, developers can improve their productivity. Finally, by using a megamodel, our mechanism extensively evaluates matching algorithms over extracted test cases. We outlined the tools needed to adapt our mechanism to the ontology context. They are mostly based on model transformations. In the near future, we wish to join an evaluation initiative, like the OAEI. The idea is to test our mechanism, notably, the development of further matching algorithms.

REFERENCES

- [1] Sam Chapman, *SimMetrics*, <http://sourceforge.net/projects/simmetrics/>, 2009.
- [2] Marcos Didonet Del Fabro, Jean Bézin, Frédéric Jouault, Erwan Breton, and Guillaume Gueltas, ‘AMW: a generic model weaver’, in *Proceedings of the 1re Journée sur l’Ingénierie Dirigée par les Modèles*, (2005).
- [3] Hong Hai Do, *Schema Matching and Mapping-based Data Integration*, Ph.D. dissertation, University of Leipzig, 2005.
- [4] Anhai Doan, Pedro Domingos, and Alon Halevy, ‘Reconciling schemas of disparate data sources: A machine-learning approach’, in *In SIGMOD Conference*, pp. 509–520, (2001).
- [5] Eclipse/M2M. Atl transformations zoo. <http://www.eclipse.org/m2m/at1/at1Transformations>, 2009.
- [6] Eclipse.org, *TCS project*, <http://www.eclipse.org/gmt/tcs/>, 2008.
- [7] Eclipse.org, *Model to Model (M2M)*, 2009.
- [8] Jerome Euzenat, *An alignment API*, <http://alignapi.gforge.inria.fr/>.
- [9] Jerome Euzenat and Pavel Shvaiko, *Ontology Matching*, Springer, Heidelberg (DE), 2007.
- [10] Kelly Garcés, Frédéric Jouault, Pierre Cointe, and Jean Bézin, ‘Adaptation of models to evolving metamodels’, Technical report, INRIA, (2008).
- [11] Kelly Garcés, Frédéric Jouault, Pierre Cointe, and Jean Bézin, ‘A Domain Specific Language for Expressing Model Matching’, in *Proceedings of the 5ère Journée sur l’Ingénierie Dirigée par les Modèles (IDM09)*, Nancy, France, (2009).
- [12] Dragan Gašević, Dragan Djurić, and Vladan Devedzic, *Model driven architecture and ontology development*, Springer Verlag, pub-SV:adr, 2006.
- [13] Guillaume Hillairet, *EMFTriple*, <http://code.google.com/p/emftriple/>, 2009.
- [14] Frédéric Jouault and Ivan Kurtev, ‘Transforming models with ATL’, in *Proceedings of the Model Transformations in Practice Workshop, MoDELS 2005*, Montego Bay, Jamaica, (2005).
- [15] Yoonkyong Lee, Mayssam Sayyadian, Anhai Doan, and Arnon S. Rosenthal, ‘etuner: tuning schema matching software using synthetic scenarios’, *The VLDB Journal*, **16**(1), 97–122, (2007).
- [16] S. Melnik, H. Garcia-Molina, and E. Rahm, ‘Similarity flooding: A versatile graph matching algorithm and its application to schema matching’, in *Proc. 18th ICDE*, San Jose, CA, (2002).
- [17] Sergey Melnik, *Generic Model Management: Concepts and Algorithms*, Ph.D. dissertation, University of Leipzig, 2004.
- [18] Marjan Mernik, Jan Heering, and Anthony Sloane, ‘When and how to develop domain-specific languages’, *CSURV: Computing Surveys*, **37**, (2005).
- [19] OAEI, *Towards a methodology for evaluating alignment and matching algorithms*, <http://oaei.ontologymatching.org/doc/oaei-methods.1.pdf>.
- [20] OMG, *OCML 2.0 Specification, OMG Document formal/2006-05-01*, <http://www.omg.org/docs/ptc/05-06-06.pdf>, 2006.
- [21] Fernando Silva Parreiras, Steffen Staab, and Andreas Winter, ‘On marrying ontological and metamodeling technical spaces’, in *ESEC/SIG*

SOFT FSE (Companion), eds., Ivica Crnkovic and Antonia Bertolino,
pp. 439–448. ACM, (2007).

- [22] Erhard Rahm and P Bernstein, 'A survey of approaches to automatic schema matching', *The VLDB Journal*, **10**(4), 334–350, (2001).
- [23] Eric Vpa, Jean Bzivin, Hugo Brunelire, and Frdric Jouault, 'Measuring model repositories', in *Proceedings of the 1st Workshop on Model Size Metrics (MSM'06) co-located with MoDELS'2006*, (2006).

A Framework for Mapping Refinement Specification

F. Hamdi¹, C. Reynaud¹, B. Safar¹

Abstract. Ontology alignment is an important task for information integration. There are many ongoing efforts to develop matching systems but these systems are not efficient for all application domains or ontologies. Very often the quality of the results can be improved by considering the specificities of the domain ontologies. In this paper, we propose an approach implemented in TaxoMap Framework, based on the alignment tool TaxoMap, which helps an expert to specify treatments exploiting mappings. The aim is to refine an alignment or to merge, restructure or enrich ontologies. We apply our approach to mapping refinement in the topographic field within the ANR (The French National Research Agency) project, GéOnto.

1 INTRODUCTION

The explosion of the number of data sources available in the web increases the need for techniques which allow the integration of these sources. The ontologies are essential elements in integration systems. However, they have to be aligned to achieve interoperability. There are many ongoing efforts to develop matching systems [4] but they are not efficient in all domains neither in all ontologies. They are very good in some cases, worse in others. The quality of their results is not always guaranteed and could often be improved if the alignment process took more into account the specificities of the aligned ontologies.

Considering these specific aspects can be done in different ways: (1) during the alignment process itself or (2) by refining the results generated by the alignment. In the first case, the adaptation of the handled ontologies is possible by the modification of the alignment process parameters or by the definition of a particular combination of alignment systems. In the second case, the refinement of mappings (the alignment results) extends and completes the alignment process, applied in the same way to all ontologies. This solution allows a finer adaptation of the alignment to the specificities of the handled ontologies. It allows also performing differentiated refinements according to the generated results. We retain this solution and extend it to consider, not only the improvement of the quality of an alignment, but also other tasks such as merging, restructuring or enriching ontologies. All these tasks are based on mappings and on the characteristics of the concerned ontologies. They must be made in interaction with the expert. Currently, there is no tool which allows to specify easily particular treatments to be applied to an alignment. The environment, called TaxoMap Framework, based on the alignment tool TaxoMap [14] [7], allows such specifications.

Our contributions, in this paper, focus on the conception of this environment, on the definition of a first set of primitives to

support the specification of treatments exploiting mappings, and on the presentation of a use of the environment for mapping refinement in the topographic field.

The paper is organized as follows. In the next section, we present the context of this work, in particular the ontology alignment tool TaxoMap and the objectives aimed by the conception of TaxoMap Framework. In Section 3 we present the approach adopted in our framework and in Section 4 we describe the use of this approach for mapping refinement applied in the topographic field within the ANR project GéOnto [5]. In Section 5 we present some related works. Finally we conclude and give some perspectives in Section 6.

2 CONTEXT

TaxoMap Framework is based on the alignment tool TaxoMap [14] [7]. We describe the tool in Section 2.1 and the objectives of the approach in Section 2.2.

2.1 TAXOMAP

TaxoMap has been designed to align *owl* ontologies $O = (C, H)$ (C is a set of concepts and H is a subsumption hierarchy). The alignment process is an oriented process which tries to connect the concepts of the source ontology O_s to the concepts of the target ontology O_t . The correspondences found are equivalence relations (*isEq*), subsumption relations (*isA*) or proximity relations (*isClose*).

To identify these correspondences, TaxoMap implements techniques which exploit the labels of the concepts and the subsumption links that connect the concepts in the hierarchy [6]. The morpho-syntactic analysis tool, *TreeTagger* [16], is used to classify the words of the labels of the concepts and to divide them into two classes, *full words* and *complementary words*, according to their category and their position in the labels. This repartition between *full* and *complementary words* is then used first, by a similarity measure that compares the tri-grams of the labels of the concepts [12] and gives more weight to the common *full words*, and second, by the alignment techniques. For example, one technique named “*isAStrictInclusion*” generates an “*isA*” mapping between X and Y if (1) the concept Y is the concept of O_t having the highest similarity value with the concept X of O_s , (2) one of the labels of Y is included in one of the labels of X , (3) all the words of the included label of Y are classified as *full words* by *TreeTagger*.

Mappings identified by TaxoMap are generated in the Alignment format [3] used as a standard in the OAEI campaign (Ontology Alignment Evaluation Initiative) [9]. We added to this standard the information about the names of the techniques

¹ LRI – University of Paris-Sud 11, CNRS & INRIA Saclay Île-de-France
Parc Orsay Université – 4 rue Jacques Monod – 91893 Orsay (France)
{Paycal.Hamdi, Chantal.Reynaud, Brigitte.Safar}@lri.fr

used in the alignment process. The aim is to facilitate the specification of treatments exploiting mappings which have been generated by these techniques.

2.2 TAXOMAP FRAMEWORK OBJECTIVES

Many ontology alignment tools have been developed in these last years but as shown in the results of the OAEI campaigns [1], no tool reaches 100% of precision and recall, even if the results obtained by some of them are very good. We observed TaxoMap through its results in this competition in the two last years [7][6] and also through our participation in the ANR project G  nto [5]. The aim of this project is the construction of a topographic ontology and its enrichment with elements coming from other geographic ontologies using alignment techniques. In this setting, tests performed on taxonomies provided by the COGIT-IGN (project partner) have shown that TaxoMap gives very good results (precision 92.3%) but these results could still be improved.

A more closely study showed that the improvements desired by the experts are often specific to the aligned ontologies. Our aim was not that TaxoMap becomes a tool dedicated to the alignment of such topographical taxonomies (and thus the quality of results would not be guaranteed when TaxoMap would be used to align other ontologies coming from other domains). Therefore, we proposed to the experts of the project an environment allowing them to specify and perform different treatments. This environment will be used to improve the quality of an alignment provided by TaxoMap, but also for any other treatment based on the results of an alignment between ontologies, such as merging, restructuring or enriching ontologies.

The expert will be able to select all primitives through an appropriate GUI. Note that the approach is based on the use of TaxoMap as alignment tool, but it could be based on another tool if the primitives associated with the tool have been defined.

3 TAXOMAP FRAMEWORK APPROACH

The approach TaxoMap Framework has been designed to meet the objectives described in section 2.2. We describe the approach and a diagram representing the architecture of this environment respectively in Section 3.1 and 3.2. This environment allows the specification of treatments from predefined primitives. These are presented in Section 3.3.

3.1 PRESENTATION OF THE APPROACH

An important feature of the approach is to allow a declarative specification of treatments based on particular alignment results and concerning particular ontologies, using a set of generic and predefined primitives.

Treatments which can be specified depend on the characteristics of the concerned ontologies and the aimed task (mapping refinement, ontology merging, restructuring, enriching). These treatments are thus associated to independent specifications modules, one for each task, having each their own set of primitives of specification. The approach is extensible and a priori applicable to any treatment based on the alignment results.

This approach should help to refine the alignment results produced by TaxoMap. It must be possible, for example, to specify that the subsumption mapping “*isA*” generated between “Way and coastal path” and “Path”, as shown in Figure 1 must be replaced by a mapping of the same type but between “Way and coastal path” and “Way”. Indeed, “Path” is defined in O_T as a kind of “Way” and the term “Way” itself is used in the label “Way and coastal path”. The expert would thus prefer to establish a mapping directly between “Way and coastal path” and “Way”.

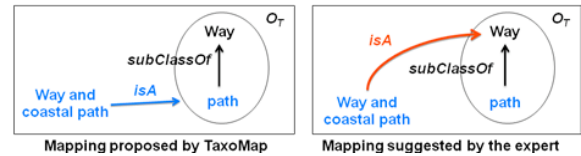


Figure1. Example of treatment to specify

The specification of treatments should be as generic as possible. Thus, the treatment shown in Figure1 should not refer directly to the concepts denoted by “Way”, “Path” and “Way and coastal path”. To help the expert to clarify the conditions for applying the treatment he wishes to implement, we propose a set of predefined generic primitives. These primitive allow a representation of various conditions which can be tested on the mapping concepts identified by TaxoMap. By analyzing alignment results and by leaning on proposed primitives, the expert will be able to identify a “group” of mappings requiring the same refinement and to specify the appropriate treatment to apply to each identified group. The specification will be so declared in a generic way then instantiated on the alignment results and the concerned ontologies to perform the corresponding treatments.

The approach should also allow other treatments such as the restructuring of an ontology O' built from O_S and O_T , and the alignments generated by TaxoMap between these two ontologies. Thus, it should help to choose, for example, what mappings “*isA*” must be transformed into “*subClassOf*” relations and what dependant concepts have to be imported in O' .

3.2 ARCHITECTURE OF TAXOMAP FRAMEWORK

Figure 2 presents the specification environment implemented in TaxoMap Framework. This environment has three modules: a “controller”, a “knowledge” and a “treatment” module.

The “knowledge” module includes all pieces of knowledge that may be used in the specification process. It includes the ontologies aligned by TaxoMap, O_S and O_T , and the generated alignment (stored in a mappings database). According to the treatment performed, we may also have other ontologies corresponding to the result of a merging or a restructuring process.

The “treatment” module includes the alignment tool TaxoMap and all the modules associated to the different tasks to perform. TaxoMap executes sequentially 9 techniques, which may or may not be chosen during a particular session. In addition, the execution order of these techniques is customizable. The modules associated to the tasks allow to specify particular treatments that an expert wishes to implement

on particular ontologies or alignments, and also to execute these treatments. Additional modules can easily be added if they combine the appropriate primitives (which may be taken from the primitives proposed in the other modules). The “controller” module manages all the treatments that can be performed using this environment, i.e. the specification of treatments and their execution, the access to relevant data and the storage of the obtained results.

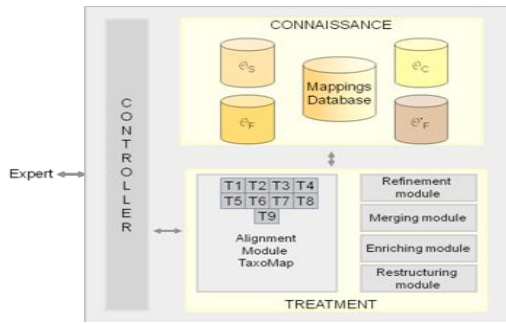


Figure 2. Architecture of TaxoMap Framework

3.3 PRIMITIVES OF TAXOMAP FRAMEWORK

The vocabulary used in the mapping refinement specification language contains:

- a set of predicate constants: $\{isEquivalence, isAStrictInclusion, isCloseStrictInclusion, \dots\}$
- a set of individual constants: $\{a, b, c, \dots\}$
- a set of variables: $\{x, y, z, \dots\}$
- a set of procedures: $\{Add_Mapping, Delete_Mapping, \dots\}$
- a set of logical symbols: $\{\exists, \wedge, \neg\}$

This vocabulary corresponds to the primitives that can be selected by the expert to specify treatments. The sets of primitive differ according to the aimed task (mapping refinement, ontology merging, restructuring, enriching).

The specification of a treatment has two parts: a “condition” part which must be satisfied to make the execution of the treatment possible, and an “action” part which expresses the process to achieve when the “condition” is satisfied.

The condition part is expressed through a set of primitives which corresponds mainly to the names of predicates. These predicates are designed to test (1) the technique used to identify the considered mapping, (2) the structural constraints on mapped elements, for example, the fact that they are related by a subsumption relation to concepts verifying or not some properties, or (3) the terminological constraints, for example, the fact that the labels of a concept are included in the labels of other concepts. These conditions are represented using three kinds of predicates:

The predicates relating to the type of techniques applied in the identification of a mapping by TaxoMap. By testing the existence in the mappings database of a particular relation generated by a given technique, these predicates test implicitly the conditions for the application of this technique. The expert neither needs to know precisely the techniques used nor to re-specify them. For example the primitive “ $isAStrictInclusion(X, Y)$ ” tests the existence of a mapping “ isA ” generated between

two concepts X and Y using the technique t_2 . It validates implicitly at the same time the conditions for the application of t_2 , i.e. (1) one of the labels of Y is included in one of the labels of X , (2) all the words of the labels of Y are classified as *full words* by *TreeTagger*, and (3) the concept Y is the concept of O_T having the highest similarity value with the concept X .

TaxoMap including several alignment techniques and thus, several predicates will be defined. More formally, let:

$R_M = \{isEq, isA, isClose\}$, the set of correspondence relations used by TaxoMap,

$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$, the set of techniques.

T_M , the table storing generated mappings in the form of 4-tuple (x, y, r, t) where $x \in C_S, y \in C_T, r \in R_M, t \in T$.

The pairs of variables (X, Y) which can instantiate these primitives will take their values in the set $\{(x, y) \mid (x, y, r, t) \in T_M\}$.

The primitives corresponding to predicate symbols necessary to illustrate the task of refinement in this paper are *isEquivalence*, *isAStrictInclusion* and *isCloseStrictInclusion*. They are defined as follows:

- *isEquivalence* (X, Y) is true iff $\exists (X, Y, isEq, t_1) \in T_M$
- *isAStrictInclusion* (X, Y) is true iff $\exists (X, Y, isA, t_2) \in T_M$
- *isCloseStrictInclusion* (X, Y) is true iff $\exists (X, Y, isClose, t_3) \in T_M$

These primitive will be presented to the expert via an interface with comments clarifying their conditions of validity as well as examples and counter-examples of use.

The predicates expressing structural relations between concepts X and Y of the same ontology $O = (C, H)$. Note that the instances of variables in these predicates will be constrained, either directly because they instantiate the previous predicates, i.e. concerning the type of the applied techniques, or indirectly by having to be in relation with other instances.

- *isSubClassOf* (X, Y, O) is true $\Leftrightarrow subClassOf(X, Y) \in H$
- *isSuperClassOf* (X, Y, O) is true $\Leftrightarrow subClassOf(Y, X) \in H$

The predicates expressing terminological relations between the labels of the concepts:

- *strictInclusionLabel* (X, Y) is defined as follows:

For each label L_1 of X

For each label L_2 of Y

If $L_1 \subseteq FullWords(L_2, L_1)$ then return true

End

where X and $Y \in C_S \cup C_C$ and $FullWords(L_2, L_1)$ is a function, which calculates all the terms of L_2 considered as full words in its comparison with L_1 .

- *conceptsDifferent* (X, Y) is true $\Leftrightarrow ID(X) \neq ID(Y)$ with $ID(X)$ is the identifier of the concept X .
- *inclusionInLabel* (X, Y) is true iff $\exists$ a label L_1 of Y / $X \subset L_1$, where $X \in \{\text{“and”}, \text{“or”}\}$ and $Y \in C_S \cup C_T$.

The action part describes the procedures to be performed. We identified an initial set of actions. They are represented using the following three procedures:

- *Add_Mapping* (X, Y, R) has the effect of adding a tuple to the table T_M which becomes $T_M \cup \{(X, Y, R, t)\}$ where R and t are fixed in the treatment condition, by instantiating the predicate identifying the technical question.

- *Delete_Mapping* $(X, Y, _)$ has the effect of removing a tuple from the table T_M which becomes $T_M - \{(X, Y, _, t)\}$

- *Add_Relation*(X, Y, R) corresponds to the addition of a relation R between X and Y with $R \in R_M \cup \{\text{subClassOf}\}$.

4 APPLICATION TO THE MAPPING REFINEMENT

The mapping refinement module is the first module of TaxoMap Framework, realized within the ANR project, GéOnto [5]. We describe the application setting and then we present the specifications of the treatment of mapping refinement required by the experts of the COGIT-IGN (project partner).

4.1 APPLICATION DOMAIN

One of the goals of the GéOnto project is to build an ontology of topographic concepts, as complete as possible, by enriching an initial taxonomy of terms. Topo-Cogit is an ontology already achieved by the COGIT. The enrichment is carried out by the alignment of this ontology with other ones of the same domain. Thus, within the project, other partners are developing an ontology based on the topographic specifications of the IGN databases and on travel books using NLP techniques [8, 10]. The enrichment process must be automated and reusable in the future on other domain ontologies. As this process is based on alignment results, these results must be as accurate as possible to minimize the contributions of the experts.

The first tests have been performed using a second ontology, Carto-Cogit, built manually from the specification of the IGN database. We aligned the 495 concepts of the source ontology Carto-Cogit with the 600 concepts of the target ontology Topo-Cogit in order to enrich Topo-Cogit. 326 mappings have been identified by TaxoMap and presented to the experts. 25 mappings (precision 92.33%) have been deemed as invalid. For other mappings, the expert proposed alternative mappings. The treatments of mapping refinement proposed below intended to obtain these alternative mappings.

4.2 THE MAPPING REFINEMENT SPECIFICATION

We present in this section two expected changes and in each change the specification of treatments such as they can be expressed in TaxoMap Framework.

- **Case 1:** The first improvement is presented in Section 3.1 (see Figure 1). Generally, it concerns mappings connecting by a subsumption relation “*isA*” a concept c_s of the source ontology O_s to a concept c_{TMax} of the target ontology O_T , such as one of the labels of c_{TMax} is included in the labels of c_s . If one of labels of the concept a that subsumes c_{TMax} in O_T is also included in the label c_s , (see Figure 3), the expert prefers to attach c_s with a , the most general concept of O_T .

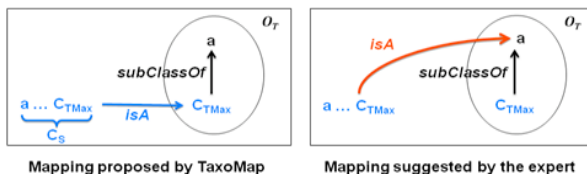


Figure 3. Illustration of the treatment1

The specification of the correspondent treatment is:

Conditions of application:

$\exists X \exists Y \text{ isAStrictInclusion}(X, Y)$
 $\wedge \exists Z \text{ isSubClassOf}(Y, Z, O_T)$
 $\wedge \text{strictInclusionLabel}(Z, X)$

Actions: *Delete_Mapping*(X, Y, _)
 $\wedge \text{Add_Mapping}(X, Z, \text{isA})$

The application of this treatment on the example presented in Figure 1 allows first to select from the mappings database the mapping (*ID*(“Way and coastal path”), *ID*(“Path”), *isA*, t_2) satisfying *isAStrictInclusion*(*ID*(“Way and coastal path”), *ID*(“Path”). The variables X and Y are instantiated respectively by X/*ID*(“Way and coastal path”) and Y/*ID*(“Path”). The use of the structural predicate *isSubClassOf*(*ID*(“Path”), Z, O_T) allows the instantiation of the variable Z, Z/*ID*(“Way”) and the verification of the terminological predicate *strictInclusionLabel*(*ID*(“Way”), *ID*(“Way and coastal path”). The mapping (*ID*(“Way and coastal path”), *ID*(“Path”), *isA*, t_2) is removed from the database and replaced by the mapping (*ID*(“Way and coastal path”), *ID*(“Way”), *isA*, t_2).

- **Case 2:** This case concerns the mapping connecting by a relation of proximity “*isClose*” a concepts c_s of the source ontology O_s to a concept c_{TMax} of the target ontology O_T , such as one of the labels of c_s is included in c_{TMax} labels. If another label of a concept in O_T contains also c_s labels and if this concept has the same father p in O_T that c_{TMax} , the expert prefers to connect to c_s to p . An illustration is given in Figure 4.

The specification of the treatment associated to the case 2 is the following:

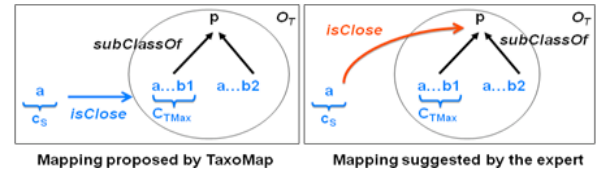


Figure 4. Illustration of the treatment2

Conditions of application:

$\exists X \exists Y (\text{isCloseStrictInclusion}(X, Y))$
 $\wedge \exists Z \exists P (\text{isSuperClassOf}(P, Y, O_T))$
 $\wedge \text{isSuperClassOf}(P, Z, O_T)$
 $\wedge \text{conceptsDifferent}(Y, P)$
 $\wedge \text{strictInclusionLabel}(X, Z)$

Actions: *Delete_Mapping*(X, Y, _)
 $\wedge \text{Add_Mapping}(X, P, \text{isClose})$

5 RELATED WORKS

The closest work we know is the COMA++ system [2]. It aims to build powerful alignment tools by the combination of existing matchers then to refine the obtained alignment results considered as preliminary. The refinement process is here totally automatic. The COMA++ alignment process is re-applied on groups of elements whose proximity has been established by a first treatment applied to ontologies. Other works deal with alignment refinement or transformation which are close

activities but not similar. In [15], correspondence patterns are used to assist the design of precise and complex ontology alignments when parts of both ontologies represent the same conceptualizations but modeled in two different ways. This approach can be seen as a way to refine one to one correspondences which can then be used to transform an ontology into another. Other works propose services to transform alignments. The Alignment API [3] generates transformations which are implementations for rendering the alignments, but the alignments are not modified.

Regarding our environment, another related work is PROMPT-Suite integrating the ontology merging tool IPROMPT [13], the alignment tool Anchor-PROMPT, versioning, comparison, translation functionalities. All these tools are interactive and semi-automatic. For example, in the fusion process the system makes suggestions. The expert can hold one of them or specify an operation to perform. The system executes then the operation, calculates the resulting changes, makes other suggestions and detects any inconsistencies.

All systems combining several alignment systems are very modular. The possibility of defining the strategy of combination makes them adaptable to a new field of application. This modularity and adaptability are strong points which also characterize our approach. However, TaxoMap Framework differs from existing tools, such as COMA++ or PROMPT-Suite, by considering that performance of an alignment tool implementing general alignment algorithms is necessarily limited. Some improvements can be obtained only after taking into account the particularities of the aligned ontologies, which involves improvements depending on the ontologies. The definition of such improvements needs to be familiar with the aligned ontologies. This process cannot thus be automatic. Only an expert of the domain is able to make it. As in PROMPT-Suite, we offer an interactive environment to help the expert to carry out this task, but we do it differently. We allow him to define particular generic treatments. In PROMPT-Suite, this is not possible. The treatments are all pre-defined.

6 CONCLUSION AND FUTURE WORK

TaxoMap Framework is an environment for the specification of treatments based on the alignment results generated by TaxoMap. We presented the implemented approach in this system, its architecture, then a first set of pre-defined primitives allowing an expert of the domain to specify easily the treatments to apply to an alignment. We presented the module for assistance to the mapping refinement that we conceived basing on the results of experiments realized within the ANR project, GéOnto.

The conception of TaxoMap Framework is adapted to the specification of other treatments such as merging, restructuring and enriching of ontologies based on alignment results. These modules are being implemented as well as the graphical interface that will allow to experts to easily select the appropriate primitives.

ACKNOWLEDGMENTS

This research is financed by The French National Research Agency through the GEONTO project (ANR-07-MDCO-005, <http://geonto.lri.fr/>).

REFERENCES

- [1] C. Caraciolo, J. Euzenat, L. Hollink, R. Ichise, A. Isaac, V. Malaisé, C. Meilike, J. Pane, P. Shvaiko, H. Stuckenschmidt, O. Svab, V. Svatek, *Results of the ontology alignment initiative 2008*, in P. Shvaiko, J. Euzenat, F. Giunchiglia, H. Stuckenschmidt (Eds), *Proceedings 3rd ISWC workshop on Ontology Matching*, Karlsruhe (DE), pp 73-119, 2008.
- [2] H.-H. Do, E. Rahm, *Matching large schemas: Approaches and Evaluation*, Information Systems 32 (2007), 857-885.
- [3] J. Euzenat, An API for ontology alignment. *An API for ontology alignment*, ISWC, Hiroshima (JP), LNCS 3298:698-712, 2004.
- [4] J. Euzenat, P. Shvaiko, *Ontology Matching*, Springer-Verlag, Heidelberg (DE), 2007.
- [5] GéOnto <http://geonto.lri.fr/>
- [6] F. Hamdi, B. Safar, N. Niraula, C. Reynaud, *TaxoMap in the OAEI 2009 alignment contest*. – *Ontology Matching*, 2009.
- [7] F. Hamdi, H. Zargayouna, B. Safar, C. Reynaud, *TaxoMap in the OAEI 2008 alignment contest*. – *Ontology Matching* 2008.
- [8] M. Kamel, N. Aussenac-Gilles, *Ontology Learning by Analysing XML Document Structure and Content*, KEOD, Madère, Portugal, 2009.
- [9] <http://www.ontologymatching.org/>
- [10] E. Kergosien, M. Kamel, C. Sallaberry, M.-N. Bessagnet, N. Aussenac, M. Gaio, *Construction automatique d'ontologie et enrichissement à partir de ressources externes*, JFO, Poitiers, 2009.
- [12] D. Lin, *An Information-Theoretic definition of Similarity*, ICML, Madison, pp. 296-304, 1998.
- [13] N. F. Noy, M.A. Musen, *The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping*, IJHCS, 59(6), pp. 983-1024, 2003.
- [14] C. Reynaud, B. Safar, *Techniques structurales d'alignement pour portails Web*, Revue RNTI W-3, Fouille du Web, Cépaduès, 2007.
- [15] F. Scharffe, J. Euzenat, D. Fensel, *Towards Design Patterns for Ontology Alignment*, SAC, pp. 2321-2325, 2008.
- [16] H. Schmid, *Probabilistic Part-of-Speech Tagging Using Decision Trees*, Int. Conf. on New Methods in Language Processing, 1994.

Adaptive user interface assistance in smart environments

Maximilian Kern and Frank Trollmann and Marco Blumendorf and Sahin Albayrak¹

Abstract. More than half of the products, which appear to be malfunctioning are often brought back to stores by the customers although they are in full working order. The reason for this is either the customers' lack of understanding of how to use such products or the unmanageable complexity of the products. This fact claims for intelligent user interfaces which adapt to the user at runtime. Such systems also require new kinds of help systems, which also need to be adaptive. In this paper, the work in progress of a user interface assistance approach in smart environments is presented. This approach aims at the utilization of model-based development in order to build self-explanatory systems which are able to guide and help the user dynamically. We take benefit of the self-describing nature of models and generate help for navigation within user interfaces. In the future, we will extend this approach in order to use the same semantics of the model which we consider for help generation in order to offer the user an alternative goal-driven interaction pattern.

1 INTRODUCTION

Software assistants provide information on the usage of a software system to the user. Their purpose is manifold. Software assistants support the user to orientate herself within a user interface and provide her explanations regarding available options, reasons for disabled options and instructions for applying certain actions. The increasing complexity of today's devices requires intelligent user interfaces which adapt to the user at runtime. Such systems require new techniques for integrating help systems. Hence, the help systems also need to be adaptive. Adaptation can happen either on context information through changes made by the user or within dynamic environments where ad-hoc ensembles are formed. A premise for help systems to adapt is to have a description, more specifically, a model of the software system which should be guided and a description of the system state at runtime.

We developed a user interface assistance approach, the MASP Guide, which was integrated into the MASP (Multi Access Service Platform), a model-based framework for user interface generation. As we focussed on model definitions for UI generation, we now aim at the utilization of the same models for explaining the user how the system works. The approach is capable of generating explanations for the usage of the system, that are based on various models with additional annotations. Following features are currently covered:

- explain the user which tasks are available at one moment.
- tell the user how to fulfil an appropriate task.
- describe which interaction devices are available and how to apply them.

¹ Technische Universität Berlin, Germany, email: firstname.lastname@dai-labor.de

First an overview of the state of the art in assistant systems is given in the next section. After this, in section 3, we define the problem which our approach addresses. In section 4, the requirements for solving the problem are declared. Then, previous work is presented in section 5, on which our approach is based. Subsequently, the current state of our approach is described in section 6. Finally we conclude our approach and discuss future work.

2 STATE OF THE ART

The earliest approaches for software assistants emerged around 1966 with the HELP system developed under the Genie project [10][11]. The HELP system provides answers to questions about commands and entities available on a UNIX based terminal window. While such approaches were restricted to low computing performance at this time, the ongoing technological improvement enables recent assistants being capable of understanding, interpreting and speaking human language, capturing and considering context information of the user and learn from the user by observing her interaction.

Rich et al. [9] [8] introduced the DiamondHelp system which establishes a collaborative dialogue between the user and a system based on the system task model. The screen is split horizontally, where in the upper half the dialogue and in the lower area the user interface of the system is displayed. Thus, the user remains able to manipulate the underlying user interface directly. The user can choose between a 'guided' interaction in form of a chat with the system or 'unguided' interaction by interacting with classical user interface elements such as buttons, labels, etc..

[4] introduced a hybrid approach, 'The Companions', which is able to incorporate knowledge retrieved from a social network or news site into a local rdf-based knowledge base. In order to give the user the impression of talking with a human, the face of the avatar is displayed. The system was designed to enrich photo albums with semantic information about recognized people and places such as their relations or detailed information. In [12] an information-seeking chat bot is presented. This chat bot supports a tourist resided in Potsdam to find sight-seeing places and gain background information related to those places such as architects, historical persons, entrance fees and public transports. It integrates an ontology with topic maps applied as the discourse of the dialogue with the user. Furthermore, this approach utilizes templates for generating natural utterances which wrap the requested information.

In [1] task models are used to connect sequences of observed user interactions to abstract tasks. Based on this information, possible interactions of users are predicted and according user interface elements are highlighted in order to provide user interface guidance.

By comparing these approaches to our approach introduced in this paper, many similarities can be recognized. One aspect, which is connecting these approaches among each other, is the challenge of utiliz-

ing knowledge contained in databases, ontologies or the Internet respectively by offering alternative ways of interaction and assistance, which pay more attention to the human nature. These approaches go beyond the WIMP-based² user interfaces, which use classical control elements as a enforcedly replacement for more intuitive user-centric interaction paradigms.

3 THE PROBLEM

In conventional user interface systems, each system state is well defined and can be already foreseen at design-time. Hence it is trivial to integrate static help for each system state. Intelligent user interfaces nowadays are highly adaptive and try to adjust themselves to the user upon context information which is gathered from sensors observing the environment, from extracted behaviour patterns and available interaction devices. The system needs to be able to offer help within any state that can be reached, although the concrete states are not predictable at design-time. Because of the resulting volatile dynamic of intelligent user interfaces, help functions and their integration within such environments became an increasingly challenging issue. The system needs to be capable of generating help which is at least as dynamic as its surrounding runtime environment. The requirements necessary to cope with this problem are discussed in the following section.

4 REQUIREMENTS

An essential premise for adaptive help is a comprehensive description of the system not only at design-time, but also at runtime. An application needs to be able to introspectively access and understand arbitrary system states at runtime, e.g. to explain required or optional tasks at the current user interface mask or explain the interaction with the displayed control elements.

Also, the adaption itself needs to be explained to the user. By introspection of the application's logic, reasons for inactivated control elements can be explained to the user and which prerequisites for their activation can be given. For instance, a search button will be activated by entering a key word or the user needs to change her location close to the device in order to activate the shown control elements for specific security-related devices such as a mixer.

Adhoc ensembles such as discovering new devices, which are available at the current location of the user, need to provide information about how to use them. As new devices are integrated into the user interface, this information needs to be dynamically weaved into the help which is available at the current step of the adapting user interface. For multimodal systems, available interaction devices should also be considered, e.g. if a microphone is available, the generated help should explain alternative interaction patterns such as using speech in order to control the user interface.

Conventional help documentations include clippings of the user interface to provide the user a link to the real user interface off the help documentation. For intelligent user interfaces it can be difficult for the user to identify those clippings after they are adapted, because their appearance is unpredictable. In order to provide meaningful help within adaptive systems, it is important to include help within the user interface itself at the current runtime state and offering direct manipulation of the user interface. Thus, by highlighting according areas of the user interface the user is able to apply given hints directly.

² Windows, Icons, Mouse, Pointer

By linking artefacts of the system to concepts of ontologies, multiple relations, which are not taken into account by the application designer, can be found and concluded to higher-level information. By applying ontology matching algorithms [5], the content presented to the user can be augmented by retrieving additional information from the internet or in case of a recipe search application, ingredients not available currently can be substituted by other appropriate ingredients. Furthermore, word analysis can be incorporated into adaptive help. Synonyms of the initiated vocabulary can be used in order to adjust the wording of the help description to the user's age or preference automatically. As the degree of adaptation is inexhaustible, the possibilities for help generation of such systems is following.

5 PRIMARILY WORK

Within the SerCHO (Service Centric HOme) project, the MASP was developed. The MASP is a model-based approach to generate user interfaces for several platforms. It incorporates well-known model based approaches such as CTT-task notation [7] and different abstraction layers such as task level, abstract user interface (AUI), concrete user interface (CUI) and final user interface (FUI) [3][6]. Each of the used models describes different aspects of the user interface, e.g. the context, the interface elements (with different abstractions), services and mappings between the models to mention only some of them. Furthermore, each model conforms to a meta-meta-model in order to separate model artefacts describing design-time- and run time-aspects and thus, allowing the models to execute themselves [2] and to adapt to any situation. Since those models describe themselves in a comprehensive way, we are convinced that they are a well-suited basis to generate meaningful support to the user. In this regard, the models can also be comprehended as an ontology which describes the system.

We evaluated the MASP by generating a virtual cook application, which offers functionalities for finding recipes with specified criteria, specifying the number of persons, generating a shopping list and prepare the chosen recipe. For each functionality a separate user interface mask is presented to the user, where she can specify desired options and run appropriate tasks. In order to improve usability and offer a more effective, intuitive and flexible interaction pattern, we extended the MASP by a user support function, which is providing help for navigation to the user. In the following section we describe this approach.

6 THE APPROACH

For the first version of our approach for generating help and guiding the user, we initially defined the requirements to be fulfilled. The MASP Guide should be capable of responding to help requests from the user in naturally spoken language. The user should be informed about available tasks at the current user interface mask. Furthermore, the user should receive detailed instructions on how to fulfil the available tasks, i.e. which information is needed, where she needs to specify them and through which modality she can enter these information. In order to attract the attention of the user to the area of the user interface which the system is currently mentioning, the appropriate area should be highlighted.

As a next step, we identified artefacts of the models, which are useful for fulfilling our requirements and information that needs to be defined additionally. In order to highlight elements as mentioned earlier, we gain layout information such as position and dimension of interface elements from the layout model. For identifying the used

interface elements (e.g. checkbox, radio button, button, etc.), the user interface model is considered. By checking the type of the available interaction task and its content, keywords which can be spoken to select a certain option are retrieved. Thus, we are able to determine which modality is available. The content (domain elements) of the current user interface is derived from the domain model. In order to

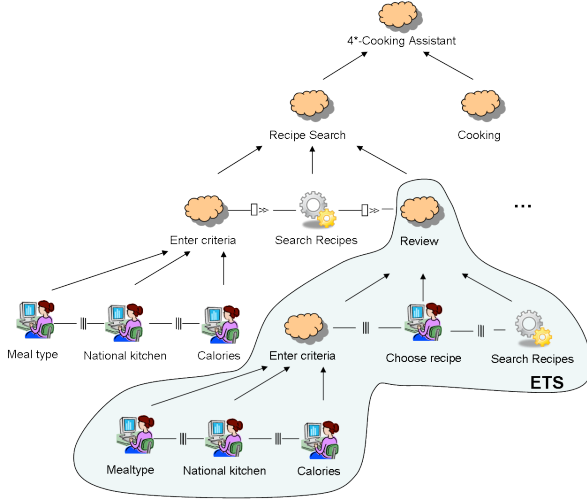


Figure 1. Excerpt of the task model of the virtual cook application.

explain a task, its task model is examined. An example for the task model is given in figure 1. Unfortunately, the information included within the task model is not sufficient for generating naturally spoken text. Therefore, we annotated descriptions to each interaction task and abstract task of the task model as depicted in table 1. Application

Task	Annotated description
Recipe Search	search for recipes
Enter criteria	define criteria for the recipe search
Meal type	specify the desired meal type
National kitchen	specify the desired national kitchen
Calories	define parameters for health awareness
Choose recipe	choose a recipe from the list of found recipes

Table 1. Annotated task descriptions.

tasks are executed by the system. At the current development phase, they are not yet considered. Since the task explanations are generated dynamically according to the enabled task set (ETS), the slices of the task description may theoretically be assembled in an arbitrary order and thus have to follow a distinct way of formulation. Additionally, the descriptions are used to identify the information requested by the user. We annotated global words for concatenating the slices, words for the beginning of the explanations, verbs for expressing the modality (e.g. speak and checking) as well as starting words for questions. Some of these definitions can be seen in table 2. Using these definitions and the task descriptions, as given in table 1, the virtual cooking assistant could formulate the sentence "You can specify the desired meal or specify the desired national kitchen or define parameters for health awareness or choose a recipe from the list of found recipes.", describing the users' current options in the situation as depicted in figure 1.

The MASP guide application itself is a standalone application within the MASP with privileges for accessing any model artefact

Description	Words
Global question	What can I do here?
Detail question	How can I ...?
Explain global start	Here you can ...
Detailed explanation start	You can ...
Concatenation word	or

Table 2. Global word definitions.

of applications running within the same MASP instance. As a consequence, the MASP Guide has its own task model, which is presented in figure 2. Initially the MASP guide is starting with the application task 'Collect description' in order to collect the information of current active applications and their enabled tasks. After collecting the information mentioned above, an interaction task 'Request help', which is waiting for an input from the user, is started. As soon as suited input is received, the assembling application task is started, which is building up the explanation by collecting the first occurring annotated task description from the top task of the ETS. For instance, as for the review task the task description below the abstract task 'Enter criteria' is not yet considered. They are considered as soon as the user is requesting details for the task 'Enter criteria'.

The abstract task 'Assemble & Explain' is iterative in order to man-

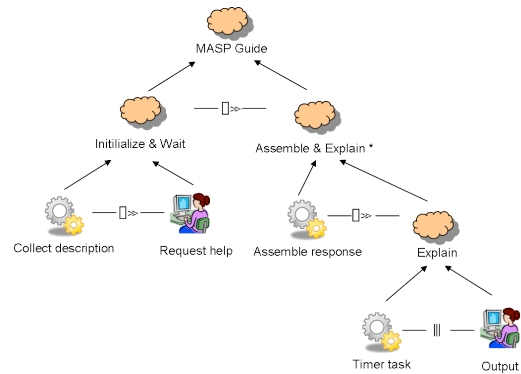


Figure 2. Task model of the MASP Guide

age the highlighting of the described area of the user interface. After building up the first/next portion of the response text, it is delegated to the explain task. The explain task is an abstract task with two child tasks, a timer task and an output task. The output task is an interaction task, which is responsible for the voice output of the previously assembled explanation and for highlighting the appropriate area of the user interface. Since there is no specification about how long an interaction task lasts, the timer task is waiting for the time span of the voice output. In the following, the pseudo code of the task 'Assemble & Explain' for the ETS highlighted in figure 1 is provided, assuming, the user asks "What can I do here?":

1. Task 'Assemble & Explain' and its subtask 'Assemble response' are activated.
2. Retrieve ETS from active application. Collect next unvisited task, if there is no unvisited task, clear state and stop the task 'Assemble & Explain'.
3. Retrieve box within the user interface, e.g. which contains control elements for entering criteria (linked to abstract task 'Enter criteria', see also left box within figure 3).
4. If first time executed, append word 'Explain global start' to the response, if not append the 'Concatenation word' to the response.

5. Retrieve description of the current task, e.g. 'Enter criteria', and append it to the response. Mark task as visited.
6. Delegate box and response to abstract task 'Explain'.
7. Abstract task 'Explain' and its subtasks 'Timer task' and 'Output' are activated.
8. 'Timer task' is executed for estimated time of voice output of the response. Abstract task 'Explain' ends after time lapses.
9. In parallel, the task 'Output' is emitting the response via voice output and pulsates the corresponding box.
10. Start over from 1.

With all mentioned annotations and model artefacts, the MASP Guide is able to explain any MASP application. Within an arbitrary state of the virtual cook application, the user can ask "What can I do here?". For instance, in the review task (see figure 3) the system replies "Here you can define the criteria for the recipe search or choose a recipe from the list of found recipes.". For retrieving detailed information about an interaction task, the user can ask e.g. "How can I define criteria for the recipe search?" and the system accordingly responds "You can specify criteria by choosing the meal type, the national kitchen or the desired calories.". To get information about how to specify e.g. criteria for meal type the user can ask "How can I choose the meal type?", the system responds "You can choose a meal type by saying or checking main menu, dessert, appetizer or pastries."

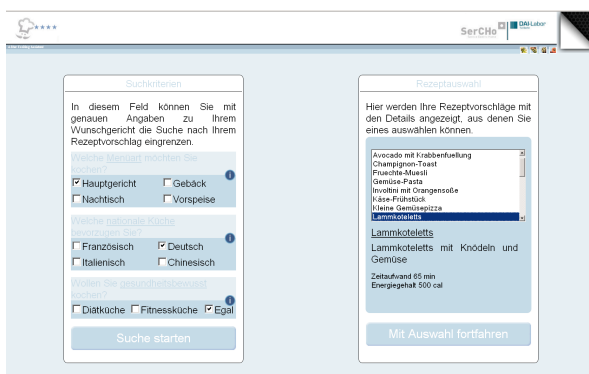


Figure 3. Recipe result mask

7 CONCLUSION AND OUTLOOK

Our approach is based on the models of the virtual cook application augmented with additional information. The MASP Guide is able to generate appropriate answers to a small set of questions. The approach is supposed to be extended by incorporating further models in future. Advanced assistants provide knowledge or call system functionalities, which are requested by the user through a dialog with the system. Assistants are a promising approach for providing the user a goal-oriented way of interaction. The user is enabled to focus on his goal rather than on single system functions. While humans are thinking on a conceptual level when interacting with a system, e.g. 'Printing a document', computers offer this abstract task as a sequence of basic operations (*open a document* → *choose menu 'file'* → *choose 'print...'* → *set up parameters* → *click button 'print'*). Assistants work as a mediator between human and systems in order to find a link between the goal, which the user has in mind and system functionalities available within a system.

In the research area of model-based software engineering, task models help a software designer to focus on a conceptual level rather than on underlying low-level functionalities. Thus, task models can be adapted in order not only to generate help, but also understand semantics of interactions on a more abstract level. Due to the nature of the CTT notation with tasks and their temporal relations, an essential ascertainment is the chance to predict any possible interaction path the user can reach within the application as described in [1]. Furthermore, it can be deduced which domain objects, more specifically which user input, is necessary for satisfying a user interaction task and run the related application task.

Based on this gnosis, we intend to provide an auto-complete function as a new interaction pattern for the MASP. With this auto-complete function the user will be able to execute multiple tasks by specifying one single sentence. For instance, the user says "I want to cook a German meal for 2 persons!" and the MASP is running the according application tasks with specified parameters, leading the user to the recipe list of German meals and skip the user interface mask, where the amount is calculated upon the number of persons.

Finally, we want to analyze wordings (find synonyms, word relations) in order to give the user more freedom in her expression. This function may become a great support to the user and may also be seen as a first step towards a better understanding of what the user wants to accomplish.

REFERENCES

- [1] Matthias Bezold, 'Describing user interactions in adaptive interactive systems', in *UMAP*, pp. 150–161, (2009).
- [2] Marco Blumendorf, Grzegorz Lehmann, Sebastian Feuerstack, and Sahin Albayrak, 'Executable models for human-computer interaction', in *Interactive Systems. Design, Specification, and Verification: 15th International Workshop, DSV-IS 2008 Kingston, Canada, July 16-18, 2008 Revised Papers*, eds., T. C. Nicholas Graham and Philippe Palanque, pp. 238–251, Berlin, Heidelberg, (2008). Springer-Verlag.
- [3] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt, 'A unifying reference framework for multi-target user interfaces', *Interacting with Computers*, **15**(3), 289–308, (2003).
- [4] A. Dingli, Y. Wilks, R. Catizone, and W. Cheng, 'The companions: Hybrid-world approach', in *International Joint Conference on Artificial Intelligence (IJCAI)*, (2009).
- [5] Jérôme Euzenat and Pavel Shvaiko, *Ontology Matching*, Springer-Verlag, Berlin-Heidelberg, 2007.
- [6] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon and Murielle Florins, and Daniela Trevisan, 'Usixml: A user interface description language for context-sensitive user interfaces', in *ACM AVI 2004 Workshop: UIXML*, pp. 55–62, (2004).
- [7] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi, 'Concurtask-trees: A diagrammatic notation for specifying task models', in *Proceedings of Interact'97*, ed., Gitte Lindgaard Steve Howard, Judy Hammond, Part of the INTERACT: IFIP International Conference On Human-Computer Interaction conference series. Chapman and Hall, (1997).
- [8] Charles Rich and Candace Sidner, 'Diamondhelp: A generic collaborative task guidance system', *AI Magazine*, **28**(2), (2007).
- [9] Charles Rich, Candace Sidner, Neal Lesh, Andrew Garland, Shane Booth, and Markus Chimani, 'Diamondhelp: a new interaction design for networked home appliances', *Personal Ubiquitous Comput.*, **10**(2-3), 187–190, (2006).
- [10] Roger Roberts, 'C s carr help—an on line computer information system', Technical Report P-4, Project Genie Document, (January 1966).
- [11] Roger Roberts, 'Help: a question answering system', in *AFIPS '70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference*, pp. 547–554, New York, NY, USA, (1970). ACM.
- [12] Manfred Stede and David Schlangen, 'Information-seeking chat: Dialogue management by topic structure', in *Proceedings of the 8th Workshop on Semantics and Pragmatics of Dialogue, CATALOG 04, Barcelona, 2004*, (2004).

The Role of Taxonomy Properties in Information Content Metrics

Raúl Ernesto Menéndez-Mora¹² and Ryutaro Ichise¹

Abstract. Assessing the semantic similarity between words is a generic problem in many research fields such as artificial intelligence, biomedicine, linguistics, cognitive science and psychology. The difficulty of this task lies in how to find an effective way to simulate the process of human judgment of word similarity by combining and processing a number of information sources. This paper presents an ongoing research where some properties of the WordNet taxonomy are used in the semantic similarity computation process. A new metric for calculating the Information Content (IC) of word senses is proposed as an alternative to the corpus based model. In an experimental evaluation the performance of the new metric is tested through their use in some IC WordNet based semantic similarity measures on a standard 28 word pair dataset.

1 Introduction

Essential for human categorization and reasoning, semantic similarity of words has become a topic of many research fields such as artificial intelligence, biomedicine, linguistics, cognitive science, and psychology. In general, semantic similarity is extensively used in a variety of applications, like words sense disambiguation [17], detection and correction of malapropisms [1], information retrieval [5], automatic hypertext linking [3] and natural language processing. Several applications to the field of artificial intelligence are discussed in [19]. However, despite numerous practical applications today, its theoretical foundations lie elsewhere, in cognitive science and psychology where it was the subject of many investigations and theories.

Semantic similarity measures have been recently applied and developed in biomedical ontologies, e.g. the Gene Ontology. They are mainly used to compare genes and proteins based on the similarity of their functions rather than on their sequence similarity [14]. Let take another example of peer-to-peer networks [4] into which semantic similarity also has found its way. Assuming a shared taxonomy among the peers to which they can annotate their content, similarities among peers can be inferred by computing similarities among their representative concepts in the shared taxonomy. This way, the more two peers are similar, the more efficient it is to route messages toward them. Numerous similar applications is the reason for the increasing interest in this subject, whose ultimate goal is to mimic human judgment with respect to the similarity of word pairs.

Semantic similarity of words is often represented by the similarity between the concepts associated with the words. Several methods have been developed to compute word similarity. Mostly operating

on the taxonomic dictionary WordNet and exploiting its hierarchical structure. Nodes based methods which generally use as its information source the Information Content (IC) of the concept, have a weakness. The conventional way of measuring the IC of word senses is to combine knowledge of their hierarchical structure from an ontology with statistics on their actual usage in text as derived from a large corpus, doing IC based methods corpus dependent.

The increasing need for better measures has led us to this study in the hope of upgrading existing semantic similarities measures. In particular, we exploits some foundations of the Intrinsic IC metric [20] approach for developing a novel metric for computing the IC. The metric, completely derived from WordNet without the need for external resources from which statistical data is gathered, is applied to recently developed semantic similarity measures [11].

The paper is structured as follows. The next section reviews some background knowledge and related work. Section 3 presents our approach as well as the novel IC metric. Section 4 discusses the results of the experiment, and section 5 summarizes our work, draws some conclusions, and outlines future tasks.

2 WordNet Similarity Measures

2.1 WordNet

A number of semantic similarity computation methods operate on the taxonomic dictionary WordNet exploiting its hierarchical structure. WordNet [2] is a machine-readable lexical database that is organized by meanings, and it was developed at Princeton University. Synonymy¹, hyponymy², meronymy³ and other arbitrarily typed semantic relationships between concepts are represented in this lexical network of English words.

WordNet, as an ontology, is intended to model the human lexicon, and psycholinguistic findings were taken into account during its design. It is classified as a light-weight ontology, because it is heavily grounded on its taxonomic structure that employs the IS-A inheritance relation; and as a lexical ontology, because it contains both linguistic and ontological information [10]. Figure 1 shows a fragment of WordNet's structure where solid lines represent is-a links and dashed lines indicate that some intervening nodes were omitted to save space. Nouns, verbs, adjectives, and adverbs are each organized into networks of synonym sets (*synsets*) that each represent one underlying lexical concept and are interlinked with a variety of relations. A polysemous⁴ word will appear in one synset for each of its

¹ National Institute of Informatics 2-1-2 Hitotsubashi Chiyoda-ku, Tokyo, 101-8430 Japan. {menendez, ichise}@nii.ac.jp

² Facultad de Informática y Matemática, Universidad de Holguín Ave. XX Aniversario, Piedra Blanca, Holguín, 80100 Cuba. raul@facinf.uho.edu.cu

¹ The semantic relation that holds between two words that can (in a given context) express the same meaning.

² The semantic relation of being subordinate or belonging to a lower rank or class.

³ The semantic relation that holds between a part and the whole.

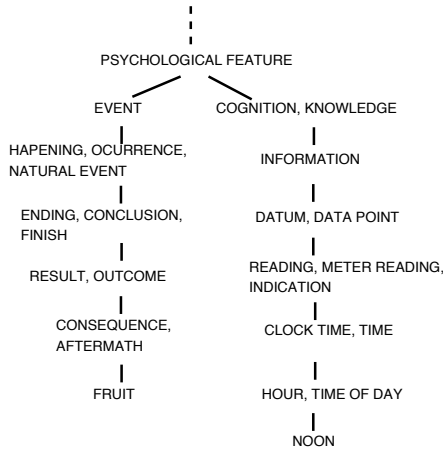


Figure 1. Fragment of WordNet taxonomy.

senses. The backbone of the noun network is the subsumption hierarchy (*hyponymy/hypernymy*), which accounts for close to 80% of the relations in WordNet.

2.2 WordNet based Semantic Similarity Measures

Based on WordNet and depending on the features taken into consideration, semantic similarity measures can be classified into two different types: *edge-based* and *node-based semantic similarity measures*.

Edge based semantic similarity measures tend to focus on structural semantic information like the distance between nodes, i.e., the number of edges separating two nodes. So, distance of the shortest path between the concepts' nodes, depth of the concepts' nodes as well as the depth of their *lowest common superset*⁵, or *subsumer (lcs)* [21] are the main features considered by this type of metrics. Some measures which classify under this category are: Wu & Palmer [21], Leacock & Chodorow [7] and Li [8] semantic similarity measure.

On the other hand, node based semantic similarity measures compute the similarity between two nodes by associating a weight to each of them. From the perspective of information theory this weight stand for the information content (IC) of the concept represented by the node. Some of the semantic similarity measures which classify under this category are: Resnik [16], Jiang & Conrath [6], Lin [9] and Pirró & Seco [15] semantic similarity measure. In the next subsection we will deeper in how this value is computed.

2.3 Information Content

Previous information theoretic approaches for computing semantic similarity [16, 6, 9] obtain the needed IC values by statistically analyzing a corpus. They associate probabilities to each concept in the taxonomy based on word occurrences in a given corpus. The IC value is then obtained by considering the negative log likelihood:

$$IC(c) = -\log p(c) \quad (1)$$

⁴ Polysemy: The ambiguity of an individual word or phrase that can be used (in different contexts) to express two or more different meanings.

⁵ For example, in Figure 1, the *lcs* between the concepts *fruit* and *noon* is the concept *psychological feature*.

where c is some concept in WordNet and $p(c)$ is the probability of encountering c in a given corpus. Resnik [16] was the first to consider the use of this formula for the purpose of semantic similarity judgments.

Seco et al. [20] proposed a different approach where they decided to use WordNet as a statistical resource with no need for external ones. Their method of obtaining IC values rests on the assumption that concepts with many hyponyms convey less information than concepts that are leaves. So, the more hyponyms a concept has the less information it expresses, otherwise there would be no need to further differentiate it:

$$IC(c) = 1 - \frac{\log(hypo(c) + 1)}{\log(\max_{wn})} \quad (2)$$

where the function *hypo* returns the number of hyponyms of a given concept and $\max_{wn}$ is a constant that is set to the maximum number of concepts that exist in the taxonomy.

2.4 Commonalities and Differences in Similarity Measures

Let introduce now the measures we will used during the experiment. Relying on the fact that most of the WordNet based semantic similarity measures just take into consideration semantic commonalities among concepts for computing their values, Menéndez and Ichise [11] proposed a featured based model for computing the semantic similarity between words where the strength of semantic differences were also exploited:

$$Sim(c_1, c_2) = \alpha * Comm(c_1, c_2) - \beta * Diff(c_1, c_2) \quad (3)$$

where $Comm(c_1, c_2)$ stands for *semantic commonalities*, $Diff(c_1, c_2)$ the *semantic differences*, and α and β tuning factors ($0 \leq \alpha, \beta \leq 1$) that represent the importance of the commonalities and differences in the model. By applying their model they proposed five semantic similarity measures as modifications of traditional WordNet based metrics from which we will be using two.

The modified Resnik measure considers the *semantic commonalities* to be the information content of the *lcs* and the *semantic differences* to be the information content encompassed by concepts, minus the one already considered in the *lcs*.

$$Sim'_{res}(c_1, c_2) = \frac{\alpha * IC(lcs(c_1, c_2)) - \beta * (IC(c_1) + IC(c_2) - 2 * IC(lcs(c_1, c_2)))}{2 * IC(lcs(c_1, c_2))} \quad (4)$$

Menéndez and Ichise observed the modified Jiang & Conrath similarity expression $Sim'_{j\&c}(c_1, c_2)$ they obtained was identical to the modified Resnik expression (Equation 4), and it was a generalization of the Pirró & Seco similarity measure [15].

$$Sim_{P\&S} \subset Sim'_{res}(c_1, c_2) = Sim'_{j\&c}(c_1, c_2) \quad (5)$$

$$Sim_{P\&S}(c_1, c_2) = \begin{cases} 3IC(lcs(c_1, c_2)) - IC(c_1) - IC(c_2) & \text{if } c_1 \neq c_2 \\ 1 & \text{if } c_1 = c_2 \end{cases} \quad (6)$$

Equation 7 represents the modified Lin measure:

$$Sim'_{lin}(c_1, c_2) = \frac{\alpha * \frac{2 * IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)} - \beta * \frac{IC(c_1) + IC(c_2) - 2 * IC(lcs(c_1, c_2))}{IC(c_1) + IC(c_2)}}{2} \quad (7)$$

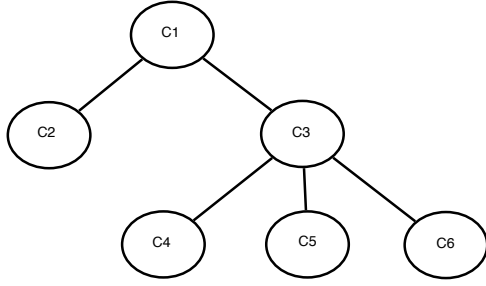


Figure 2. Abstract taxonomy.

Despite proposing a general model, Menendez and Ichise just experimented with the semantic differences [11]. In the following sections we will propose a novel metric for computing the IC which we will use in Pirr6 & Seco measure as well as in the modified Resnik and Lin measures. We will also study the performance of these semantic similarity measures with the new IC metric.

3 Method

3.1 Approach

As pointed out before, similarity metrics grounded on IC obtain IC’s values for concepts by statistically analyzing large corpora and associating a probability to each concept in the taxonomy based on its occurrences within the considered corpora. From a practical point of view, this approach has two main drawbacks: (i) it is time consuming and (ii) it heavily depends on the type of corpora considered. Toward mitigating these drawbacks Seco et al. [20] proposed the Intrinsic Information Content (Equation 2). The IC values of concepts rest on the assumption that the taxonomic structure of WordNet is organized in a meaningful and structured way, where concepts with many hyponyms convey less information than concepts that are leaves, that is, the more hyponyms a concept has the less information it expresses.

From Seco et al. perspective, concepts that are leaf nodes are the most specific in the taxonomy so the information they express is maximal. This means it does not matter how deep is the leaf in the taxonomy. However, our approach, although founded in the same ideas, is different. We stand the depth in the taxonomy it is also an important factor to consider. *The deeper a concept is found in a taxonomy means the amount of previous knowledge is larger and it should bear a higher value of information content.*

For example, in Figure 2 concepts C2 and C4 should not have the same IC value, but since they both have the same number of hyponyms under Seco et al. approach they have equal values of IC. We have to clarify that, as well as generally assumed in WordNet, we are considering each of the edges hold the same amount of importance, which indeed it is not true. In the next subsection we introduce the mathematical foundation behind our idea.

3.2 A Novel Information Content Metric

As we made clear in the previous section, our corpora independent IC metric will be a function of the *number of hyponyms* ($hypo(c)$) and the *concept’s depth* ($depth(c)$). For example in Figure 2 we say $Sim(c_2, c_1) > Sim(c_4, c_1)$, but if we use Seco et al. intrinsic IC’s

approach [20] this is not true:

$$\begin{aligned}
 hypo(c_2) &= hypo(c_4) \\
 IC(c_2) &= IC(c_4) \\
 3IC(c_1) - IC(c_2) - IC(c_1) &= 3IC(c_1) - IC(c_4) - IC(c_1) \\
 Sim_{P\&S}(c_2, c_1) &= Sim_{P\&S}(c_4, c_1)
 \end{aligned}$$

To avoid the previous situation we define some properties which should be fulfilled by our IC metric:

$$IC(c) \propto depth(c)$$

$$IC(c) \propto \frac{1}{hypo(c)}$$

$$IC(c_1) = IC(c_2) \Leftrightarrow \begin{aligned} &hypo(c_1) = hypo(c_2) \\ &\wedge depth(c_1) = depth(c_2) \end{aligned}$$

$$IC(c_1) < IC(c_2) \Leftrightarrow \begin{aligned} &hypo(c_1) > hypo(c_2) \\ &\wedge depth(c_1) = depth(c_2) \end{aligned}$$

$$IC(c_1) < IC(c_2) \Leftrightarrow \begin{aligned} &hypo(c_1) = hypo(c_2) \\ &\wedge depth(c_1) < depth(c_2) \end{aligned}$$

The Information Content metric would be:

$$IC_{hd}(c) = 1 - \frac{\log((hypo(c) + 1) * (max_{depth} - depth(c) + 1))}{\log(max_{wn} * max_{depth})} \quad (8)$$

where function $hypo$ returns the number of hyponyms of a given concept, max_{wn} is a constant that is set to the maximum number of concepts that exist in the taxonomy, function $depth$ returns the depth of a given concept and max_{depth} represents the maximum depth of the corresponding taxonomy. The denominator, which is equivalent to the value of the most informative concept, serves as a normalizing factor in that it assures that IC values are in the range $[0, 1]$. The above formulation guarantees that the information content decreases monotonically. Moreover, the information content of the imaginary top node of WordNet would yield an information content value of 0.

4 Experiments and results

4.1 Experimental Settings

The purpose of the experiment was to evaluate the new information content metric. We use it into some information content based similarity measures (Equations 4, 6 and 7) and compare the results through the performance of those similarity measures with and without our new information content metric. As a baseline for subsequent comparisons we started with the results presented in [11]. We used the human judgments of Pirr6 and Seco experiment [15] (P&S in the following) for the word pairs on the Miller and Charles dataset (M&C in the following).

Unfortunately, there is a distinct lack of standards for evaluating semantic similarities, which means that the accuracy of a computational method for evaluating word similarity can only be established by comparing its results against human common sense. That is, a method that comes close to matching human judgments can be deemed accurate. Moreover, some datasets for this evaluation are commonly used. In particular, the Rubenstein and Goodenough dataset (R&G in the following) and Miller and Charles dataset (M&C) are standards dataset for evaluating semantic similarities.

In 1965, Rubenstein and Goodenough [18] obtained “synonymy judgments” of word pairs by hiring 51 subjects to evaluate 65 pairs of nouns. The subjects were asked to assign a similarity from 0 to 4, from “semantically unrelated” to “highly synonymous”. Miller and

Charles [12], 25 years later, extracted 30 pairs of nouns from the R&G dataset and repeated their experiment with 38 subjects. The M&S experiment achieved a correlation of 0.97 with the original experiment of R&G. Resnik [16], in 1995, replicated the M&C experiment with 10 computer science students, obtaining a correlation of 0.96. Pirró and Seco [15] (P&S) in 2008 also recreated the R&G experiment this time with 101 subjects, and arrived at a correlation coefficient of 0.972 for the full dataset ($P\&S_{full}$).

As mentioned above, we used the judgments of the P&S experiment for the word pairs of the M&S dataset. However, we considered only 28 word pairs of the 30 used in the M&S experiment: a word missing in WordNet made it impossible to compute ratings for the other two word pairs. All the evaluations were performed using WordNet 3.0 [2] and the Brown Corpus was used for the calculation of the corpora dependent's information content metric. The computation used Pedersen's WordNet::Similarity Perl module [13] as the core. Table 1 shows the parameters values we used for the computation during the experiment.

Table 1. Values of the parameters for the corpus independent information content computation.

Parameter	Taxonomy	Value
$\max_{wn}$	Noun Verb	82115 25047
$\max_{depth}$	Noun Verb	20 14

For the modified measures, we varied the importance of the difference's factor β in the range of [0,1] and calculated human judgments of the P&S experiment as done in [11]. Nevertheless, we also varied the importance of the semantic commonalities factor α in the range of [0,1] and that's why our results for the modified Resnik measure are better than the one presented in [11].

4.2 Results and Discussion

Table 2 is a summary of the evaluations in the current state of the research. It compiles the correlation values of three similarity measures (Sim'_{in} , Sim'_{res} and $Sim_{P\&S}$) for the traditional corpus based information content, the intrinsic IC and our proposed metric (IC , IIC and IC_{hd} respectively).

Table 2. Maximum values of correlation obtained using different IC metrics

	IC	IIC	IC_{hd}
Sim'_{in}	0.858	0.879	0.878
Sim'_{res}	0.831	0.897	0.895
$Sim_{P\&S}$	0.831	0.891	0.892

Analyzing the obtained data we realize of the competitiveness of our new IC metric compared to the intrinsic information content. Furthermore it does outperform the traditional corpora based IC metric for all the similarity measures tested. At this point of the research we still have to find out a better function for describing the relationship between the depth, the number of hyponyms and the IC value of a concept. Due to the "odd" behavior of the function $hypo$ in the domain of English words with a wide range of possible values ($hypo(c) \in [0, 82115]$) it is difficult to properly model the relation between IC , $hypo$ and $depth$. So we have to model how does behaves

an information content metric given two concepts (c_1 and c_2) under the following conditions:

$$\begin{aligned} hypo(c_1) &> hypo(c_2) \\ depth(c_1) &< depth(c_2) \end{aligned}$$

We try to statistically approximate the metric expression through a linear regression but as we expect the relation between $hypo$ and $depth$ functions with IC it is not linear.

5 Concluding Remarks and Future Work

In the present paper we show a new approach for computing IC values in a corpora independent way by using taxonomy properties. We proposed a new IC metric which achieves better results than the traditional corpus based IC metric. Our approach solves the weakness of the intrinsic IC metric while keeping the competitiveness of the metric. We plan to apply some machine learning or evolutive computing methods find out and model different relations between the taxonomy properties an the IC metrics, so we can achieve a bigger improvement of the semantic similarity measures.

REFERENCES

- [1] Alexander Budanitsky and Graeme Hirst, 'Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures', in *Workshop on WordNet and Other Lexical Resources, 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, (2001).
- [2] *Wordnet: An Electronic Lexical Database*, ed., Christiane Fellbaum, Bradford Books, 1st edn., 1998.
- [3] Stephen J. Green, 'Building hypertext links by computing semantic similarity', *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, **11**(5), 713–730, (1999).
- [4] Jin Hai and Chena Hanhua, 'Semrex: Efficient search in a semantic overlay for literature retrieval', *Future Generation Computer Systems*, **11**(6), 475–488, (2008).
- [5] Angelos Hliaoutakis, Giannis Varelas, Epimenidis Voutsakis, Euripides G. M. Petrakis, and Evangelos E. Milios, 'Information retrieval by semantic similarity', *Int. Journal on Semantic Web and Information Systems (IJSWIS)*, **2**(3), 55–73, (2006).
- [6] Jay J. Jiang and David W. Conrath, 'Semantic similarity based on corpus statistics and lexical taxonomy', in *Int. Conf. on Research in Computational Linguistics*, pp. 19–33, (1997).
- [7] Claudia Leacock and Martin Chodorow, 'Combining local context and wordnet similarity for word sense identification', in *WordNet: A Lexical Reference System and its Application*, pp. 265–283, (1998).
- [8] Yuhua Li, Zuhair A. Bandar, and David McLean, 'An approach for measuring semantic similarity between words using multiple information sources', *IEEE Transactions on Knowledge and Data Engineering*, **15**(4), 871–882, (2003).
- [9] D. Lin, 'An information-theoretic definition of similarity', in *15th Int. Conf. on Machine Learning*, pp. 296–304, (1998).
- [10] Laurent Mazuel and Nicolas Sabouret, 'Semantic relatedness measure using object properties in an ontology', in *The Semantic Web - ISWC 2008*, volume 5318/2008 of *LNCIS*, pp. 681–694, (2008).
- [11] Raúl Ernesto Menéndez-Mora and Ryutaro Ichise, 'Effect of semantic differences in wordnet-based similarity measures', in *23rd Int. Conf. on Industrial, Engineering & Other Applications of Applied Intelligent Systems: IEA-AIE 2010*, (2010). Accepted.
- [12] G.A. Miller and W.G. Charles, 'Contextual correlates of semantic synonymy', *Languages and Cognitive Processes*, **6**(1), 1–28, (1991).
- [13] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi, 'Wordnet::similarity - measuring the relatedness of concepts', in *19th National Conf. on Artificial Intelligence (AAAI-04)*, pp. 1024–1025, (2004).
- [14] Catia Pesquita, Daniel Faria1, Andre O. Falcao, Phillip Lord, and Francisco M. Couto, 'Semantic similarity in biomedical ontologies', *PLoS Computational Biology*, **5**(7), (2009).

- [15] Giuseppe Pirró and Nuno Seco, 'Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content', in *On the Move to Meaningful Internet Systems: OTM 2008*, pp. 1271–1288, (2008).
- [16] Philip Resnik, 'Using information content to evaluate semantic similarity in a taxonomy', *Int. Joint Conf. on Artificial Intelligence*, **14**(1), 448–453, (1995).
- [17] Philip Resnik, 'Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language', *Artificial Intelligence Research*, **11**, 95–130, (1999).
- [18] Herbert Rubenstein and John B. Goodenough, 'Contextual correlates of synonymy', *Communications of ACM*, **8**(10), 627–633, (1965).
- [19] Nuno Seco, *Computational Models of Similarity in Lexical Ontologies*, Master's thesis, University College Dublin, 2005.
- [20] Nuno Seco, Tony Veale, and Jer Hayes, 'An intrinsic information content metric for semantic similarity in wordnet', in *European Conference on Artificial Intelligence*, pp. 1089–1090, (2004).
- [21] Zhibiao Wu and Martha Palmer, 'Verb semantics and lexical selection', in *32nd Annual Meeting of the Association for Computational Linguistics*, pp. 133–138, (1994).

Evaluation of Error Taxonomy for OWL2

Naseer Ahmed Sajid¹, Dr. Muhammad Abdul Qadar²

Abstract. Ontology Evaluation is one of the most critical phases in ontology engineering. Applications depending upon ontology can have serious and appalling problems if ontology itself is infected with errors. Therefore, ontology should be consistent, complete and errors free. An error taxonomy using OWL constructs and axioms was developed to overcome the possibility of occurring errors in ontology. Recently, some new features have been proposed in OWL2 specification to overcome the limitation of expressive power of OWL. In this paper we evaluate the extended error taxonomy of OWL with the new features of OWL2. We also evaluate that while using OWL2 features, occurrence of possibility of new types of errors get reduced or not. On the basis of different examples for each error, we conclude that the error taxonomy is also valid for OWL2. This is because of the fact that every OWL ontology is a valid OWL2 ontology and every OWL2 ontology (excluding new features) is also a valid OWL ontology. The zero level circulatory errors can also be removed if we use the reflexive property and self restriction (local reflexive) property of OWL2. Otherwise, we have to generate a warning omission for zero level circulatory error.

1 Introduction

An explicit or formal specification of the domain is called ontology. It is a declarative representation which includes the terminology (or vocabulary) for referring to the terms in that domain and the logical statements that illustrate what the terms are, how they are related to each other, and how they can or cannot be related to each other. Ontology should represent formal specification about the domain concepts and relationships among them [5]. Ontologies therefore provide a vocabulary for representing and communicating knowledge about some domain and a set of relationships that hold among the terms in that vocabulary.

Like other component of a system, ontology has to go through a repetitive process of refinement and evaluation during its development lifecycle. Ontology content evaluation is one of the critical phases of Ontology Engineering because if ontology itself is infected with errors then the applications dependent on it, may have to face some critical and appalling problems and ontology may not serve its purposes. Various types of research efforts have used to evaluate ontologies but besides the evaluation of ontologies, there is a need to evaluate the error taxonomy while development an ontologies. If ontology has some errors then there

is possibility of errors in capturing and representing of different aspects of real world. So the evaluation of ontology ensures the quality of content and methodology.

Ontology evaluation includes ontology verification, validation and assessment. Ontology verification insures that its definition implement correctly, all the competency questions and requirements or functions correctly in the real world. Ontology validation insures the ontology definitions really model the real world for which the ontology was created. Ontology assessment is done by different types of users and application, judging the content of ontology from the user's point of view.

OWL is not enough because some ontologies use limited expressive power. It is not expressive enough as there is no user define data types [2, 3], no meta-modeling support [2, 3] and also limited property support [4].

OWL2 extends ontology web language by adding some new features and some extension in OWL. Users request these new small but useful features while they had problems in developing ontology. New features include extra syntactic sugar, additional property and qualified cardinality constructors, extended data type support, simple meta-modeling and extended annotations [1].

The remainder of this paper is structured as follows: the very next section is about OWL2 and its constructs and axioms. Error taxonomy is described in section 3. In section 4, we evaluate error taxonomy for OWL2 with examples. Paper is concluded in section 5.

2 OWL2

OWL2 is an ontology web language. It is an extension and revision of OWL developed by the web ontology working group of W3C. OWL2 is fully backwards compatible with OWL i.e. every OWL ontology is a valid OWL2 ontology and every OWL2 ontology (not using new features) is also a valid OWL ontology. OWL and OWL2 are intended to facilitate the sharing and development of ontology through web, with the purpose to easily access the contents of web to machines [6]. OWL2's ontologies provide classes, instances, properties and data values and stored in a semantic web documents. These ontologies can be written in RDF documents.

OWL2 provides some new features i.e. syntactic sugar, additional properties, database style keys, qualified cardinality constructors, and extended data types, simple meta-modeling and extended annotations. These new features increase the expressive power in OWL2. OWL2's profiles may meet better performance requirements and may be easier to implement. Main goals of OWL2 are to define profiles for OWL that are easier to implement and deploy as said above. It may cover all the important application areas and are easily understandable to non-expert users.

Faculty of Computer Science and Engineering, Muhammad Ali Jinnah University, Islamabad, Pakistan. ¹ nasajid2005@gmail.com, ² aqadir@jinnah.edu.pk

3 Extended Error Taxonomy

Errors taxonomy was classified by Gomez et al in [8], she formed error taxonomy for the assessment of ontologist. Ontology engineers used errors taxonomy for the development of errors free ontology. Keeping in mind of this errors taxonomy, developers forms a well define classification of concepts for sound semantic web vision. Errors taxonomy has basically three types of errors which are usually encountered by the ontologists. These are inconsistencies, incompleteness and redundancy [10]. There was a contribution in error taxonomy by many other researchers. Fahad et al [9], extend the error taxonomy for the evaluation of ontology. Errors Taxonomy with extended errors is shown in Figure 1.

4 Evaluation

In this section, we evaluate different types of errors which can be occurred while developing an ontology.

4.1. Inconsistency Errors

While reasoning from ontology, due to inconsistency and contradiction, three types of errors can be occurred. These errors are partition errors, circulatory errors and semantic inconsistency errors.

4.1.1. Partition Errors

Partition errors occur when we decompose a concept into various sub-concepts. Partition error occurs when a developer creates a common class which belongs to different disjoint classes or properties [10]. Concept classification can be defined in a disjoint/exhaustive decomposition.

Common classes in disjoint decompositions and partitions

There is no any axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. But other way round, there is an *DisjointUnion* axiom in OWL2 in which we can define a class and union of its disjoint classes. It is a modeling error and can be occur due to an un-experience developer. It can only be overcome by an experience person who is going to develop ontology.

Example:

If we define a MadCow class as a subclass of both Herbivorous and Carnivorous. Also if we define a MS Leading to Ph.D partition as a subclass of both Graduate and Post-Graduate partition.

Common instances in disjoint decompositions and partitions

These errors occur when a common instance belong to more than one class of a disjoint decomposition and partition.

Example:

If we define an instance ABC of both Herbivorous and Carnivorous classes. Also if we define a Naseer as an instance of both Graduate and Post-Graduate partition.

There is no any axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. It is a

modeling error and can be occur due to an un-experience developer. It can only be reduce by an experience person who is going to develop ontology.

#	Type	Errors
1	Inconsistency	Common classes in disjoint decomposition and partitions of classes
2		Common instances in disjoint decomposition and partitions
3		External instances in exhaustive decomposition and partitions
4		Common property in disjoint decomposition and partitions of properties
5		Circulatory errors in class or in property hierarchy (Zero Level)
6		Circulatory errors in class or in property hierarchy (Non-Zero Level)
7	Semantic	More generalized concept as subclass
8		Domain violation by subclass
9		Disjoint Domain by subclass
10	Incompleteness	Sufficient knowledge omission
11		Disjoint knowledge omission among properties
12		Disjoint knowledge omission among classes
13		Exhaustive knowledge omission
14		Functional property omission
15		Inverse-Functional property omission
16	Redundancy	Incomplete concept classification
17		Identical formal definition of some classes
18		Identical formal definition of some properties
19		Identical formal definition of some instances
20		Redundancies of subclass of relations
21		Redundancies of sub-property of relations
22		Redundancies of instances of relations
23		Redundancies of disjoint of relations

Figure 1. Errors Taxonomy

External instances in exhaustive decompositions and partitions.

External instances in exhaustive decomposition and partition errors occur when there is at least one instance which belongs to the base concept but it does not belong to the sub-concepts.

Example:

If we can define a MadCow class as a subclass of Living animals but we have both Herbivorous and Carnivorous as exhaustive decomposition of Living Animals. Also if we can define a Ph.D Student class as subclass of Student but we only partitions of Graduate and UnderGraduate.

If a developer overlook some concept classification to disjoint with others then this may be occur. It is a modeling error and can be occur due to an un-experience developer and insufficient knowledge about that domain. There is no any axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. But other way round, there is an *DisjointUnion* axiom in OWL2 in which we can define a class and union of its disjoint classes. It can only be reduce by an experience person who is going to develop ontology.

Common property in disjoint decompositions and partitions in properties.

OWL allows to specify disjoint axioms between properties as well. Like common class, this error occurs when a common property is created in a disjoint decomposition of properties.

Example:

If we define *hasAccess* property as subProperty of both disjoint *hasManagerPrivileges* and *hasClientPrivileges* property.

In OWL2, there is no such axiom or feature which can reduce the possibility of occurrence of this type of error. It is a modeling error and can be occur due to an un-experience developer and insufficient knowledge about that domain. It can only be reduce by an experience person who is going to develop ontology.

4.1.2. Circulatory Errors

Circulatory errors occur when a developer or ontologist defines a class or property which is sub-class (or sub-property) or super-class (super-property) of itself at any level of hierarchy in the ontology [10].

Issues

There are some issues while applying the OWL2 new features. As in the literature it is already defined that there exist a circulatory zero level error e.g. *Traveler* is subclass-of itself, is a circulatory zero level error. OWL2 provides a reflexive property [7], which relates everything to itself. For examples, everybody has himself as a relative or every class is its own subclass or every person is a person. So reflexive property avoids these types of circular zero level error. But here it is again, if a developer or ontologist overlook this OWL2 new feature i.e. reflexive property and define such types of classes then circulatory zero level error may occur.

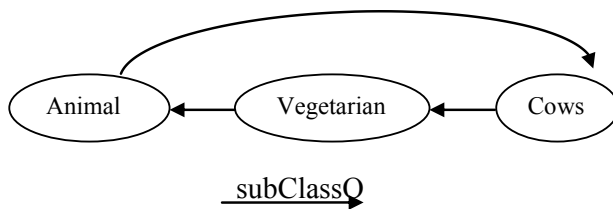


Figure 2a. Circulatory error in class hierarchy

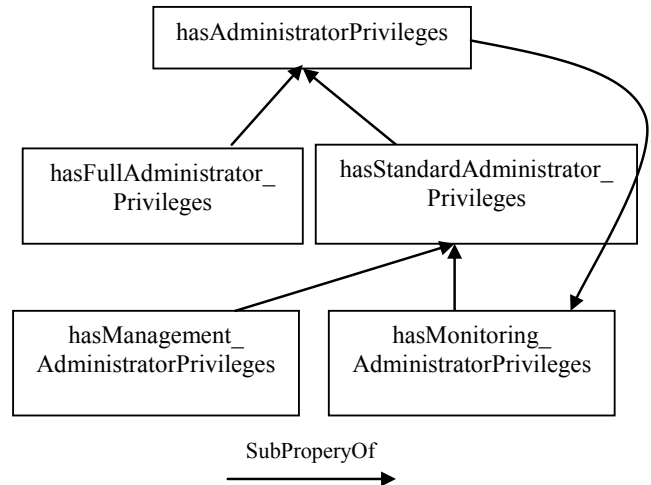


Figure 2b. Circulatory error in property hierarchy

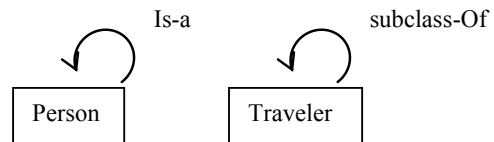


Figure 3. Reflexive

Similarly, OWL2 provides a self restriction or local reflexive property [7], which interlink instances or an individual might be link to itself. For example, when we say that all *NarcissisticPersons* love themselves or some persons killed themselves.

EquivalentClasses(:NarcissisticPerson ObjectHasSelf(:loves))

Self-restriction (local reflexivity): $\exists$ killed. Self

Or $Kills(X,X) \rightarrow Suicide(X)$

$\exists Kills. Self \sqsubseteq Suicide$

and

$Suicide(X) \rightarrow Kills(X,X)$

$Suicide \sqsubseteq \exists Kills. Self$

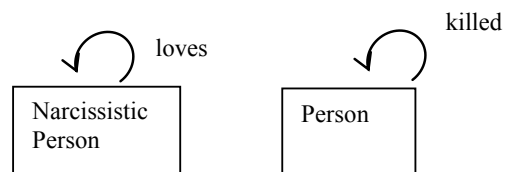


Figure 4. Self Restriction

Suicide is basically same as kills.

So self restriction or local reflexive property avoids these types of circular zero level error. But if a developer or ontologist overlook this OWL2 new feature i.e. self restriction or local reflexive property and define such types of classes or properties then circulatory zero level error may occur.

Hence to overcome above types of issues we have to generate warning for the ontologist or developer while developing ontology.

4.1.3. Semantic Errors

Semantic inconsistency errors occur when developer or ontologist create a class or subclass which violates the class hierarchy because that class does not actually belong to it [10].

Weaker Domain Specified by SubClass

These types of errors occur when ontologists specify subclass that possesses more general (weaker) domain as compared to the superclass domain [14], also when ontologists forget to specialize the concept going down in the hierarchy of concepts.

Example:

Person $\sqsubseteq$ 5.hasChild.male
Person $\sqsubseteq$ 2.hasChild.Person

If a developer is an un-experience person and has insufficient knowledge about that domain for which he/she is going to develop ontology, then this type of error occur. As there is also no such axiom or feature in OWL2 which can reduce the possibility of occurrence of this type of error.

Domain Breach Specified by SubClass

These types of errors occur when ontologists specify subclass that overlaps other classes.

Example:

Person $\sqsubseteq$ hasSalary $\geq$ 20,000
Person $\sqsubseteq$ (hasSalary $\geq$ 10,000 $\sqsubseteq$ 50,000)
Person $\sqsubseteq$ hasSalary $\geq$ 40,000

Overlapping of subclasses may be specify by a developer if he/she is an un-experience and has insufficient knowledge about that domain for which he/she is going to develop ontology, then this type of error occur. As there is also no such axiom or feature in OWL2 which can reduce the possibility of occurrence of this type of error.

Disjoint Domain Specified by SubClass

This error occur when ontologist specify a subclass that is disjoint with its domain of superclass.

Example:

Person $\sqsubseteq$ hasSalary $\geq$ 20,000
Person $\sqsubseteq$ hasSalary $\leq$ 15,000
Person $\sqsubseteq$ hasSalary $\geq$ 40,000

OWL2 also has axiom of **DisjointWith**, it depends on the ontologist or developer to explicitly mention the disjointness among the classes and subclasses to avoid this type of errors. It can be also be occur due to a developer who is an un-experience and has insufficient knowledge about that domain for which he/she is going to develop ontology.

4.2. Incompleteness Errors

Sometime ontologist or developer while developing ontology, some information about the classification is overlooked by ontologist or developer. If some information is imprecisely model or possible information is missing about the decomposition, then some ambiguity and lack of reasoning arise due to this

incompleteness. It is also possible when someone omitted concepts during the definition of the taxonomy and also if someone overlook some important information while classification of concepts.

4.2.1. Partition Errors

These types of errors can be occurred when some partitions of concepts is omitted. Some of these errors are as follow:

Disjoint Knowledge Omission among Classes

When a concept is classified into many subclasses and partition and knowledge about the disjointness of partition is omitted then this type of incompleteness error can be occur.

Example:

Qadir et al. [15] presented a case study of Access_Policy ontology in which disjoint knowledge omission between classes' administrator and user produces catastrophic results.

Example:

If we donot define AccessPolicy that userAccess and AdministratorAccess are disjoint subclasses of AccessPolicy.

OWL2 also has axiom of **DisjointWith**, it depends on the ontologist or developer to explicitly mention the disjointness among the classes and subclasses to avoid this type of errors. It can be also be occur due to a developer who is an un-experience and has insufficient knowledge about that domain for which he/she is going to develop ontology.

Disjoint Knowledge Omission among Properties

Like disjoint knowledge omission among classes, it is also significance importance of disjoint axioms between properties as well. So ontologist should check and specify knowledge between properties and avoid creating common instance among them.

Example:

Similarly, we have to mention **DisjointWith** among the properties. OWL2 also provides a facility to explicitly mention the **DisjointWith** among the properties.

Sufficient Knowledge Omission

Every concept should have necessary and sufficient knowledge so that application can reason about them. Necessary information provide by subclass of relation and sufficient information by other axioms of DL. Sufficient knowledge is not provided in the description of concepts, thus application can not differentiate between them. If there is only necessary description available and sufficient description of concepts is not available, due to these two reason sufficient knowledge Omission Error [12] can be occur.

Example:

Previlages

Administrator_Full_Previlages
Administrator_Standard_Previlages

There is no axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. It can reduce only by an experience person who has a sufficient knowledge about that domain for which he/she is going to model an ontology.

Exhaustive Knowledge Omission

This error occurs when ontologists do not follow the completeness constraint while decomposition of concept into subclasses and partitions [8].

Example:

In *Vehicle* ontology, *Bus* and *Motor-Cycle* classes are disjoint subclasses of *Vehicle* concept, but do not specified that whether or not this classification forms an exhaustive decomposition.

There is no axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. Ontologist or developer must have a good knowledge about the domain.

4.2.2. Functional and Inverse Functional Property Omission

By default owl takes multiple values for a given property. If a property takes just at most one value then it should be declared as functional property. e.g. age, height, User ID and Inverse Functional property is that for which two different objects cannot have the same value. e.g., NIC of Person, User ID of user.

```
< owl:DataProperty rdf:ID = "UserID" >
  < rdfs:domain rdf:resource = #User/>
  < owl:range rdf:resource = #int/>
</owl:DataProperty >
Needs restriction to take one and unique value
```

OWL2 provides both types of properties i.e. functional and inverse functional properties, but ontologists or developers have to mention it explicitly to avoid these types of errors.

4.2.3. Incomplete Concept Classification

This error occurs when ontologists overlook some of the concepts present in the domain while classification of particular concept [8].

Example:

If we want to make a *Living Animals* ontology and we ignore the *Omnivorous Animals*, then it is incomplete classification of concept *Living Animals*.

There is no axiom or feature available which can reduce the possibility of occurrence of this type of incompleteness errors. Experience ontologist or developer can reduce this error of incomplete concept classification.

4.3. Redundancy Errors

These types of errors occur when a particular information is derived from more than one classes, instances and their relationships. Detection of these types of redundancies is harder.

4.3.1. Identical Formal Definition of Classes/Instances

When Ontologist provides same formal definition of some classes but defines the different names for that classes [13].

Example:

```
User ⊆ hasPrivilege AdministratorAccessPolicy
User' ⊆ hasPrivilege AdministratorAccessPolicy
```

Both classes have same formal definition but with different names. Similarly, identical formal definition of instances cab also be occurred.

In OWL2, we can give same name to different types of entities with certain restrictions. E.g. The name used for a class or a datatype can also be used for an individual, object property, data property or annotation property. The name used for an individual can also be used for a class, datatype, object property, data property or annotation property

4.3.2. Identical Formal Definition of Properties

This type of error occur when ontologists specify multiple names which are semantically same for properties having same domain and range.

Example:

```
hasAccess(User, AccessPolicy)
hasPrevilage(User, AccessPolicy)
```

Here domain and range is same for given properties but both properties semantically are same.

Similarly, in OWL2, we can give same name to different types of entities with certain restrictions. E.g. The name used for an object or data or annotation property can also be used for an individual, class or datatype.

4.3.3. Grammatical Errors

There are some grammatical errors due to which redundancy may happen when developing taxonomies.

Redundancies in Subclass-of Relation

Redundancy of subclass-of relation errors take place when developer or ontologist make classes which have directly or indirectly subclass-of relations more than once [8].

Example:

If we want to make an *Vehicle* ontology and we define *Bus* as a subclass of *PassengerVehicle* and *MotorVehicle* creates redundancy as *PassengerVehicle* is already a subclass of *MotorVehicle*. Here indirect subclass of relation exists between *Bus* and *MotorVehicle* creating redundancy.

In OWL2, there is no such axiom or feature which can reduce the possibility of occurrence of this type of error. As it is a modeling error and can only reduce by an experience ontologist or developer who is going to develop an ontology for that particular domain. He/she should have as much knowledge as to require for a that particular domain.

Redundancies in Subproperty-of Relation

Redundancy of subproperty-of relation errors take place when developer or ontologist make data properties or object properties which have directly or indirectly subproperty relations more than once [8].

Example:

A datatype property *TelenorNo* is specified as subproperty of *MobileNo* and *ContactNo* as well, then redundancies of subproperty of relation exist.

Similarly, in this case, there is no axiom or feature in OWL2 which can reduce the possibility of occurrence of this type of error.

Redundancies in Instance-of Relation

When ontologists specify an instance x to a concept (C) and again specify x as instance of its (C) ancestor concepts [8], then this

type of error occurs. When a concept is specified as instance of a particular concept (C) then it is implicitly known as an instance of its parent concepts and does not need explicit description of instantiation to its super concepts.

Example:

If we specify instance *+923465100010* as an instance of *TelenorNo* and *ContactNo* classes, then it creates redundancy of instance of relation, as it is already defined that *TelenorNo* is a subclass of *ContactNo*. This explicit instance of relation between *923465100010* and *ContactNo* creates redundancy as *923465100010* is indirect instance of *ContactNo* as it is a superclass of *TelenorNo* class.

There is no restriction in OWL2 which can reduce this type of error. But ontologist or developer can do this, due to which this type of error will occur.

Redundancies in Disjoint-of Relation

If a concept is specified as disjoint with other concepts more than once [12], then this type of error is occurred. According to description logic rules [16], if a concept is disjoint with any concept then it is also disjoint with its sub-concepts. Only possibility of this error is that the concept is specified as disjoint with a concept but it was already specified that their parents were disjoint. Likewise redundancy of disjoint relation among property hierarchies exists in datatype property or object property hierarchies.

In OWL2, there are also tags for the disjoint of classes and properties. If we wrongly use these tags and mention the disjointness among classes and properties then this type of error will occur. Because we cannot specify the disjoint concept more than once.

5 Conclusion

We have discussed different types of errors and we have come to know that there are no such constructs or axioms or features in OWL2 specification which can remove the possibilities of occurring errors which is shown in Table 1. Hence all the errors are also valid in OWL2 except circulator zero level error. Although expressive power is increased by using OWL2 features. OWL2 also provide the punning (meta-modeling) concept which is used to give same name to different types of entities, with certain restrictions. Some issues are also occurred while using the OWL2 features i.e. due to omission of specifying reflexive property and self restriction or local reflexive property, there is a possibility of circulatory zero level errors. To overcome these issues, we have to generate warning for the developer or ontologist who is going to develop or model an ontology for that particular domain. For this purpose ontologist or developer should have sufficient knowledge to that particular domain for which he/she is going to develop an ontology.

We are also working to extend the error taxonomy by using the OWL2 constructs and axioms and will also generate warning for the omission of reflexive and self restriction property.

#	Errors	OWL	OWL2
1	Common classes in disjoint decomposition and partitions of classes	Valid	Valid
2	Common instances in disjoint decomposition and partitions	Valid	Valid
3	External instances in exhaustive decomposition and partitions	Valid	Valid
4	Common property in disjoint decomposition and partitions of properties	Valid	Valid
5	Circulatory errors in class or in property hierarchy (Zero Level)	Valid	Invalid
6	Circulatory errors in class or in property hierarchy (Non-Zero Level)	Valid	Valid
7	More generalized concept as subclass	Valid	Valid
8	Domain violation by subclass	Valid	Valid
9	Disjoint Domain by subclass	Valid	Valid
10	Sufficient knowledge omission	Valid	Valid
11	Disjoint knowledge omission among properties	Valid	Valid
12	Disjoint knowledge omission among classes	Valid	Valid
13	Exhaustive knowledge omission	Valid	Valid
14	Functional property omission	Valid	Valid
15	Inverse-Functional property omission	Valid	Valid
16	Incomplete concept classification	Valid	Valid
17	Identical formal definition of some classes	Valid	Valid
18	Identical formal definition of some properties	Valid	Valid
19	Identical formal definition of some instances	Valid	Valid
20	Redundancies of subclass of relations	Valid	Valid
21	Redundancies of sub-property of relations	Valid	Valid
22	Redundancies of instances of relations	Valid	Valid
23	Redundancies of disjoint of relations	Valid	Valid

Table 1. Comparison of OWL and OWL2 Error Taxonomy

References

- [1]. "OWL2 Web Ontology Language: New Features and Rationale", W3C working group, <http://www.w3.org/TR/2008/WD-owl2-new-features-20081202/>
- [2]. Jeff Z. Pan. Description Logics: Reasoning Support for the Semantic Web. Ph.D. Thesis, School of Computer Science, University of Manchester, 2004.
- [3]. Jeff Z. Pan, Ian Horrocks and Guus Schreiber. OWL FA: A Metamodeling Extension of OWL DL. In Proc. of the International workshop on OWL: Experience and Directions (OWL-ED2005).
- [4]. Jeff Z. Pan and Ian Horrocks. OWL-Eu: Adding Customised Datatypes into OWL (Extended version). In Journal of Web Semantics, 4(1): 29-39. 2006.
- [5]. Antoniou, G., and Harmelen, F.V.,. *A Semantic Web Primer*. MIT Press Cambridge, ISBN 0-262-01210-3, 2004.
- [6]. "OWL2 Web Ontology Language Document Overview", W3C working group, <http://www.w3.org/TR/2009/WD-owl2-overview-20090327/>
- [7]. "OWL 2 Primer", W3C proposed recommendation 22 September, 2009 <http://www.w3.org/TR/2009/PR-owl2-primer-20090922/>
- [8]. Gomez-Perez, A., Fernández-López, M., Corcho, O., *Ontological engineering: With examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Springer, London, 2004.
- [9]. Fahad, M., and Qadir, M.A. *A Framework for Ontology Evaluation*. Proceeding of 16th International Conference on Conceptual Structures (ICCS'08), Toulouse, France. Vol-354, pages 149-158, July 2008.
- [10]. A. Gomez-Perez, M.F. Lopez, and O.C. Garcia, *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Springer ISBN:1-85253-55j-3, 2001.

- [11]. M.A. Qadir, M. Fahad, S.A.H. Shah, Incompleteness Errors in Ontologies. Proc. of Intl GRC 07, USA. IEEE Computer Society. pp 279-282, 2007b.
- [12]. W. Noshairwan, M.A. Qadir, M.A., M. Fahad, Sufficient Knowledge Omission error and Redundant Disjoint Relation in Ontology. InProc. 5th Atlantic Web Intelligence Conference June 25-27, France, 2007a.
- [13]. Fahad. M., Qadir.M.A. and Noshairwan. W., “Ontological Errors-Inconsistency, Incompleteness, Redundancy”, In International Conference on Enterprise Information Systems (ICEIS) 2008, Barcelona, Spain.
- [14]. Fahad, M., Qadir, M.A., and Noshairwan, W., *Semantic Inconsistency Errors in Ontologies*. Proceeding of International Conference on Granular Computing, Silicon Valley, USA, IEEE Computer Society, (2007b).
- [15]. Qadir, M.A., Noshairwan, W., (2007c). *Warnings for Disjoint Knowledge Omission in Ontologies*. In Proc. of 2nd ICIW'07, Morne, Mauritius (2007), IEEE C.S, pp. 45
- [16]. Nardi, D., et al. *The Description Logic Handbook: Theory, Implementation, and Applications*. ISBN: 9780521781763, 2002.
- [17]. Grau. B., Horrocks. I., Motik. B., Parsia. B., Patel. P., Sattler. U., “OWL2: The Next Step for OWL”, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 6, No. 4, pp 309-322, 2008.

Integrating Technical Terms into Thesaurus using Extracted Related Words

Yoshimi Suzuki and Fumiyo Fukumoto¹

Abstract. Thesaurus expansion is important for most of NLP applications. Because word senses are useful information for them and new words which are not listed in a dictionary appear in documents: web pages, patent documents, and so on. This paper describes a method to integrate new technical terms which appear in patent documents into a thesaurus. To integrate new words into a thesaurus which has hierarchical semantic features, we used some words which are listed in the thesaurus and are extracted as the similar words of the new words. For validity test of the method, we performed some experiments using words which are listed in the thesaurus as test data. And also we performed some experiments using new words. The results suggest that the proposed method is effective for thesaurus expansion.

1 Introduction

There are a large collection of patent documents, and many people with different interests and with different specialized fields. When a person who is not the expert in the area retrieves patents, it is very difficult for him to find the relevant patent documents. Because he may not know the technical terms which are related to the keywords in the area. Also, researchers in same area may not remember keywords in a moment. For example, researchers of a company use 'stall' and researchers of another company use 'hang up'. Therefore, it is important to construct thesaurus of technical terms for the person who is not the expert in the area. For example, using thesaurus of the technical terms, user can use the words which he did not know but are listed in the thesaurus as synonyms of the keyword. Also using the thesaurus, the system can retrieve patent documents using query term expansion.

Currently there are a lot of electronic thesauri, e.g. Roget's Thesaurus, WordNet[5], BunruiGoiHyo [7] and EDR concept dictionary. However, they are thesauri of common words and they have few technical terms. For example, in EDR concept dictionary which has 410 thousands words and has about 8 levels of hierarchy, the hypernym of "collector (one of electrodes of transistor)" is "machine component", and "base" and "emitter" which are also electrodes of transistor are not listed in EDR concept dictionary. EDR concept dictionary which lists 410 thousands words can not be expanded by integrating technical terms, because its classification for technical terms is very coarse. Therefore, it is necessary to construct thesaurus of technical terms.

There are a lot of studies for thesaurus construction and thesaurus expansion. Tokunaga, et al. [11] and Uramoto [8] proposed methods for extending an existing thesaurus by classifying new words in terms of that thesaurus. However, their studies are for commonly

used words and not for technical terms. Also there are many studies of ontology mapping, ontology alignment and ontology matching (e.g., [10], [2], [9]). These studies are able to use structure information of both ontology. However, for integrating unknown words into a thesaurus, we can not use semantic structure of unknown words.

For thesaurus construction or thesaurus expansion, we have to extract similar word pairs. In order to extract similar word pairs, some methods ([3], [4] and [6]) based on dependency relationships are proposed. However, their methods are for commonly-used words and they did not mention whether their methods were effective for technical terms.

For extracting similar words of technical terms, we have to deal with the following difficulties.

- Some technical terms appear less frequency.
- Some technical terms are used in same context. Therefore it is difficult to obtain various dependency relationships.

In this paper, for deal with these difficulties, we present a method for detecting word senses of unknown technical terms using extracted coordinate terms of the technical terms and hierarchical semantic features of a concept dictionary. Also we propose a method to integrate new technical terms into a core thesaurus. We perform some experiments in order to confirm effectiveness of the proposed method.

2 System Overview

Figure 1 shows the overview of our system. Our system consists of 4 procedures.

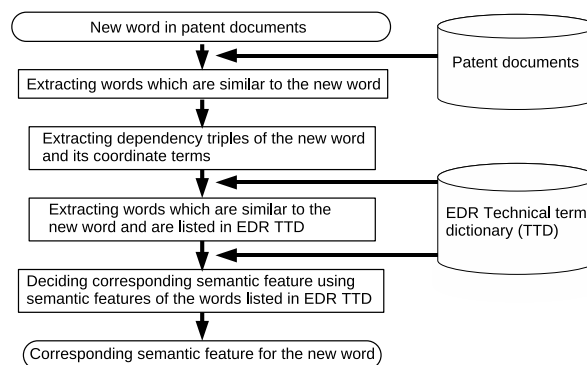


Figure 1. System Overview

We explain 4 procedures of our system using examples of the new technical terms: 'JBIG', 'corpus' and 'packet'.

¹ University of Yamanashi, Japan, email: ysuzuki@yamanashi.ac.jp

3 Extracting words Which are Similar to the new word

Some technical terms do not frequently appear in even if large corpora. Table 1 shows frequencies of 5 technical terms in the patent documents. Although the technical terms in Table 1 are technical terms about data compression, 'JPEG' appears frequently, however 'JBIG' appears few times. Therefore, in order to deal with the technical terms like a 'JBIG', we have to extract technical terms which are the coordinate terms of the technical terms.

Table 1. Frequencies of 5 technical terms in the patent documents

Technical terms	code	frequency
JBIG	none	200
JPEG	none	1446
GIF	none	2
image compression	3cd3e6	3643
data compression	3c8276	7783

3.1 Dependency Relationship

Dependency information is used for extracting semantic similar pairs. For example, Lin proposed "dependency triple" [4]. A dependency triple consists of two words: w, w' and the grammatical relationship between them: r in the input sentence. $||w, r, w'||$ denotes the frequency count of the dependency triple (w, r, w') . $||w, r, *||$ denotes the total occurrences of $w - r$ relationships in the corpus, where "*" indicates wild card.

We used 10 kinds of case particles as r . Table 2 illustrates the case particles which we used as r .

Table 2. 10 case particles

case particle	description
<i>ga</i>	nominative case
<i>no</i>	genitive case
<i>wo</i>	accusative case
<i>ni</i>	dative case
<i>he</i>	goal
<i>to</i>	comitative case
<i>kara</i>	relative case
<i>yor</i>	from, at
<i>de</i>	inessive case
<i>ya</i>	coordination

For example, a dependency triple (apply, object, JBIG) is obtained for the unknown word "JBIG". However, some technical terms do not appear frequently, then we have to use hierarchical semantic features for smoothing technique.

4 Extracting Dependency Triples of the new word and Its Coordinate Terms

In order to extract the corresponding semantic feature of the new word, we extract dependency triples of the new word and the extracted words. Table 3 illustrates examples of dependency triples of the extracted words. Using some extracted words, many types of dependency triples are extracted.

Table 3. Examples of dependency triples of the extracted words

w	r	w'
message	object (<i>wo</i>)	save
message	object (<i>wo</i>)	send
frame	object (<i>wo</i>)	discard
information	subject (<i>ga</i>)	change
signal	subject (<i>ga</i>)	contain

5 Extracting Words Which are Similar to the new word and are Listed in EDR Technical term dictionary

We extract words which are similar to the new word and are listed in the core thesaurus. For extracting the similar words from the core thesaurus, first, $I(w, r, w')$ is calculated using Formula (1).

$$\begin{aligned}
 I(w, r, w') &= -\log(P_{MLE}(B)P_{MLE}(A|B)P_{MLE}(C|B)) \\
 &\quad -(-\log P_{MLE}(A, B, C)) \\
 &= \log \frac{||w, r, w'|| \times ||*, r, *||}{||w, r, *|| \times ||*, r, w'||} \quad (1)
 \end{aligned}$$

where P_{MLE} is the maximum likelihood estimation of a probability distribution.

Let $T(w)$ be the set of pairs (r, w') such that $\log \frac{||w, r, w'|| \times ||*, r, *||}{||w, r, *|| \times ||*, r, w'||}$ is positive. The similarity $Sim(w_1, w_2)$ between two words: w_1 and w_2 is defined by Formula (2).

$$\begin{aligned}
 Sim(w_1, w_2) &= \frac{\sum_{(r, w) \in T(w_1) \cap T(w_2)} (I(w_1, r, w) + I(w_2, r, w))}{\sum_{(r, w) \in T(w_1)} I(w_1, r, w) + \sum_{(r, w) \in T(w_2)} I(w_2, r, w)} \quad (2)
 \end{aligned}$$

Table 4 shows examples of extracted similar words of "JBIG".

Table 4. Extracted similar words of "JBIG" (top 5)

rank	words in the core thesaurus	similarity
1	image compression	0.331
2	data compression	0.313
3	algorithm	0.182
4	program	0.155
5	command	0.153

6 Detecting Corresponding Semantic Features

Finally, candidates of corresponding semantic features of the new word are detected using the hierarchical semantic features of the core thesaurus. Table 5 illustrates the extracted technical terms which are similar to "packet" and are listed in the core thesaurus. In Table 5, most of top 10 words are technical terms about transmitting data.

Figure 2 illustrates part of hierarchical semantic features associated with "packet". For estimation, we use node distance between each node of the extracted word and node of the new word (packet). We defined node distance was number of arcs from the new word to the node which was the ancestor of the new word and the extracted word. For example, in Figure 2 node distance between "packet" and

Table 5. Extracted proper nouns for “3c8e70 packet” (top 10)

rank	noun	code	similarity
1	message	3c9112	0.542
2	frame	2e6a69	0.535
3	information	2dd8aa	0.532
4	signal	3c8966	0.523
5	bit	2dee6f	0.442
6	transmit data	2fb213	0.415
7	content	2dec4b	0.410
8	image data	3c74fa	0.404
9	received data	2fe07a	0.398
10	block	2def72,3c8561	0.396

“bit” is 1. In Figure 2, average of node distance of top 10 words and “packet” is 2.36 ($= \frac{1+1+2+2+2+2+2+3+3+4+4}{11}$).

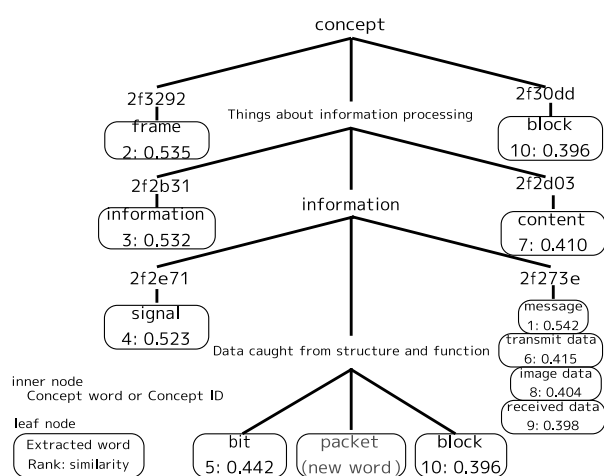


Figure 2. Hierarchical semantic features of “packet” and its similar nouns (top 10)

7 Experiments

7.1 Experimental setup

We used patent publication bulletins written in Japanese (1993) which were provided by patent retrieval task of NTCIR Workshop 4 [1]. The patent publication bulletins consist of about 210 thousands patent documents. The new unknown technical terms are selected from the patent documents.

We used technical term dictionary (domain: information processing) by EDR as a core dictionary. It has about 90 thousands concept classification records. Figure 3 illustrates hierarchic structure of semantic feature of “corpus” in the dictionary..

The depth of the hierarchical semantic features is 6. ‘corpus’ is one of child nodes of ‘document’. The lower semantic features of ‘document’ are ‘corpus’, ‘archive’ and so on.

Figure 4 shows three types of related words and dictionary structure of the core thesaurus. “processing products” is the hypernym of ‘documents’. ‘corpus’, ‘archive’ and ‘thesaurus’ are hyponyms of ‘documents’. And also these words are coordinate terms of each other.

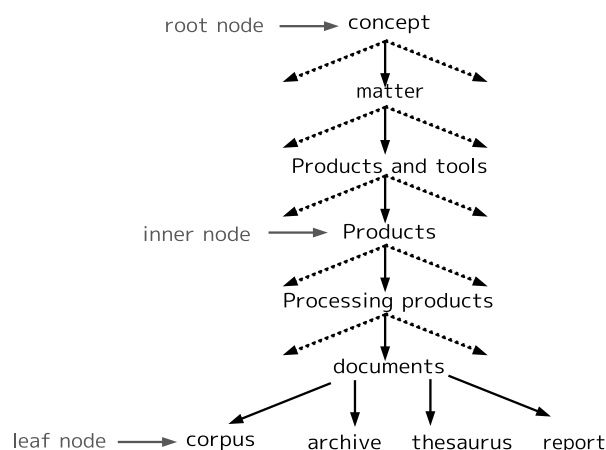


Figure 3. Hierarchical semantic feature of “corpus” in EDR Concept Classification Dictionary of Technical Terms

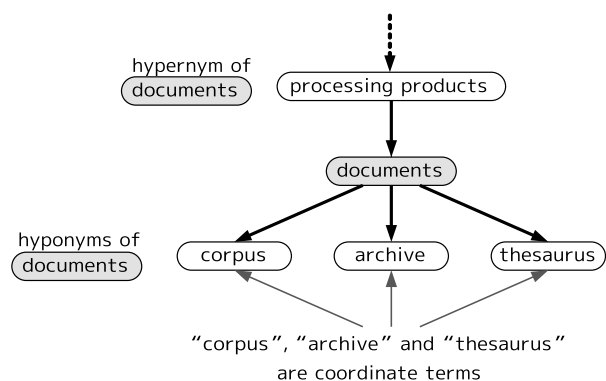


Figure 4. Dictionary structure and related words

7.2 Experimental results and Discussion

We performed experiments of integrating technical terms into a thesaurus using 100 test words. We confirmed that 100 technical terms are classified into the corresponding semantic features by using our method. The technical terms were selected from top 400 words which are frequently searched in “Dictionary of IT technical term ‘e-Words’”².

Table 6 shows the results.

Table 6. 100 technical terms and their average of node distance

Technical terms	Average of node distance
CCD	4.12
LAN	3.56
LED	3.20
algorithm	3.43
bit	3.30
buffer	3.71
computer	3.22
counter	3.85
database	2.79
data bus	3.33
device	2.61
driver	3.21
flag	1.89
flip-flop	2.66
format	4.05
header	3.52
interface	2.23
low pass filter	4.00
microprocessor	3.87
memory	2.41
modem	1.00
module	3.91
network	3.89
node	2.97
offset	2.92
packet	2.36
parameter	3.11
password	4.83
pixel	3.33
processor	3.85
protocol	4.22
polling	3.54
port	2.34
register	4.75
scanning	4.00
segment	3.88
socket	2.26
software	3.71
subroutine	3.74
throughput	2.62
...	...
average of 100 words	3.54

In Table 6, “average of node distance” indicates average of node distance between the new word and its similar words (top 10 words and their similarities are higher than 0.5) which are listed in the core thesaurus. As the results, we obtained average of node distance was 3.54 using the hierarchical semantic features whose depth is 8.

The worst case was ‘password’, whose average of node distance was 4.83. Table 7 shows the extracted similar words of ‘password’. Although the average of node distance of ‘password’ is not so good and node distance between ‘passcode’ and ‘password’ is 6, ‘passcode’ and ‘password’ are similar to each other. From these results,

it appears that the core thesaurus is too small and some polysemous words are unconsidered.

Table 7. Similar words of ‘password’

rank	similar word	similarity	node distance
1	passcode	0.548	6
2	identification number	0.538	6
3	number	0.534	5
4	ID	0.532	5
5	ID number	0.521	2
6	identification code	0.509	2

Table 8 shows the results of the words which are not listed in the core thesaurus. As the results a node is selected as the hypernym the words.

Table 8. Unknown words and the selected node as the hypernym of the words

num	word	extracted hypernym
1	filament	parts of hardware
2	solar battery	parts of hardware

Figure 5 shows how to integrate new word: ‘filament’ into core thesaurus. For deciding the hypernym of ‘filament’, we trace links from root node to a leaf node. If a node has more than one child node, we select a child node whose subtree has extracted words more than other subtrees. If the extracted words in the subtree are less than 2, the nodes are decided as the hypernym of the new words.

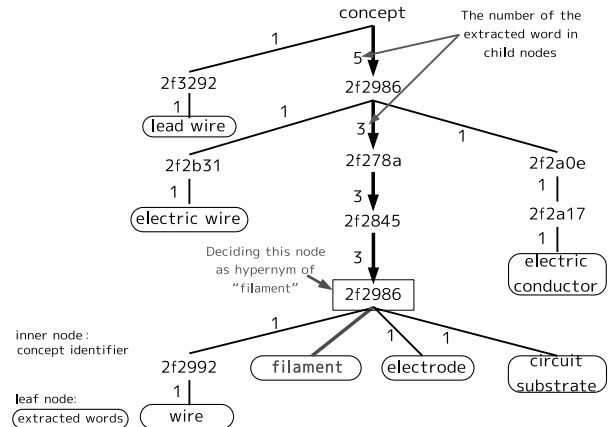


Figure 5. Experimental Results: “filament”

Although we use very few test data and use a core thesaurus which has coarse classification, we can integrate new technical terms into a thesaurus.

8 Conclusion

This paper proposed a method to integrate new technical terms into a thesaurus using extracted similar words of the technical terms. Using the extracted words from patent documents and hierarchical structure of a technical term dictionary, we showed the possibility of thesaurus expansion.

² <http://e-words.jp/>

9 Acknowledgments

This work was supported by the Grant-in-aid for the Japan Society for the Promotion of Science (JSPS).

REFERENCES

- [1] A.Fujii, M.Iwayama, and N.Kando, ‘Test collection for patent-to-patent retrieval and patent map generation in ntcir-4 workshop’, in *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pp. 1643–1646, (2004).
- [2] Domenico Beneventano and Daniele Montanari, ‘Ontological mappings of product catalogues’, in *In Proceedings of The Third International Workshop on Ontology Matching*, (2008).
- [3] D.Hindle, ‘Noun classification from predicate-argument structures’, in *Proceedings of 28th Annual Meeting of the Association for Computational Linguistics*, pp. 268–275, (1990).
- [4] D.Lin, ‘Automatic retrieval and clustering of similar words’, in *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics Proceedings of the Conference*, pp. 768–774, (1998).
- [5] Christiane Fellbaum, *WordNet: An Electronic Lexical Database*, Bradford Books, 1998.
- [6] M. Hagiwara, Y. Ogawa, and K. Toyama, ‘Selection of effective contextual information for automatic synonym acquisition’, in *In Proc. of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp. 353–360, (2006).
- [7] National Language Research Institute, *Bunruigoihyo*, Shuei publisher (In Japanese), 1964.
- [8] N.Uramoto, ‘Positioning unknown words in a thesaurus by using information extracted from a corpus’, in *In proceedings of COLING’96*, pp. 956–961, (1996).
- [9] Dominique Ritze, Christian Meilicke, Ondrej Svab-Zamazal, and Heiner Stuckenschmidt, ‘A pattern-based ontology matching approach for detecting complex correspondences’, in *In Proceedings of The 4th International Workshop on Ontology Matching*, (2008).
- [10] Konstantin Todorov and Peter Geibel, ‘Ontology mapping via structural and instance-based similarity measures’, in *In Proceedings of The Third International Workshop on Ontology Matching*, (2008).
- [11] T.Tokunaga, ‘Extending a thesaurus by classifying words’, in *In Proceedings of the ACL-EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources*, pp. 16–21, (1997).

An evidence-based approach to collaborative ontology development

Emma Tonkin¹, Heather D. Pfeiffer² and Andrew Hewson¹

Abstract. The development of ontologies for various purposes is now a relatively commonplace process. A number of different approaches towards this aim are evident; empirical methodologies, giving rise to data-driven procedures; self-reflective (innate) methodologies, resulting in artifacts that are based on intellectual understanding; collaborative approaches, which result in the development of an artifact representing a consensus viewpoint.

We compare and contrast these approaches through two parallel use cases, in work that is currently ongoing. The first explores a case study in creation of a knowledge base from raw, semi-structured information available on the Web. This makes use of text and data mining approaches from various sources of information, including semi-formally structured metadata, interpreted using methods drawn from statistical analysis, and data drawn from crowd-sourced resources such as Wikipedia.

The second explores ontology development in the area of physical computing, specifically, context-awareness in ubiquitous computing, and focuses on exploring the significant impact of an evidence-led approach. Both examples are chosen from domains in which automated extraction of information is a significant use case for the resulting ontology. In the first case, automated extraction takes the form of indexing for search and browse of the archived data. In the second, the predominant use cases relate to context-awareness.

Via these examples, we identify a core set of design principles for software platforms that bring together evidence from each of these processes, exploring participatory development of ontologies intended for use in domains in which empirical evidence and user judgment are allied.

1 INTRODUCTION

Ontologies, defined in the computer science area as “agreement about a shared, formal, explicit and partial account of a conceptualisation” (Spyns, 2002) are increasingly visible in various disciplines, particularly in the area of knowledge management, where the encoding of static domain knowledge is a key process (Aldea et al, 2003). Ontologies are generally applied for a number of purposes, including the following (Noy and McGuinness, 2001): to share common understanding of the structure of information among people or software agents; to enable reuse of

domain knowledge; to make domain assumptions explicit; to separate domain knowledge from the operational knowledge; and to analyze domain knowledge.

Various methods for ontology generation have been identified: these could be described in general as introspection, or self-reflection (externalisation of expert knowledge), collaborative development through introspection and discussion (joint use of expert knowledge; see for example Della Valle et al, 2008), and data-driven or corpus-driven means (ie. unsupervised methods of ontology generation). Linking two or more of these methods together is also possible; for example, Carvalheira and Gomi (2007) describe a method that makes use of automated ontology generation to fuel a semi-automatic, or ‘hybrid’, overall process. Ontologies are also becoming so large that it is hard for a single group to effectively develop and build a single ontology (T. Tudorache, et. Al, 2008); therefore, some collaborations have gone to using systems such as Protégé to keep track of the development process.

The debate surrounding the process of ontology building contains parallels with other forms of knowledge organisation and historical discussions on the topic of eliciting information about knowledge, language and the structures that underlie our everyday activities. In certain domains, the choice between introspective and data-driven approaches is one that defines the shape of the discipline. One example is the study of grammar in human languages; another is the perceived gulf between structured taxonomy/vocabulary for classification and the use of unstructured or very loosely-structured approaches, such as social tagging.

Here, we discuss an approach that links together a data-first approach with a collaborative discovery. Reports synthesised from expert knowledge have the advantage of very closely approaching the individual's own viewpoint; if then bolstered by discussion with others, the result may approach a consensus viewpoint. Such an approach does not take into account the visibility or availability of these features within the data that is available. Under some circumstances, this characteristic is not a defect for an ontology; however, if it is to be used for a data-driven or highly data-dependent application, such as for example a system that classifies documents within an ontology, or a context-aware wearable device using of a set of sensor signals to characterise and perhaps identify the current context in which the user stands, it is advantageous for the ontology to approach the dataset (to the extent that concepts and datasets may be expected to map).

¹ UKOLN, University of Bath. Email: {e.tonkin, a.hewson}@ukoln.ac.uk.

² Klipsch School of Electrical and Computer Engineering, NMSU. Email: hdp@cs.nmsu.edu.

2 A SOFTWARE-ENGINEERING APPROACH TO ONTOLOGY DEVELOPMENT

The practical importance in a collaborative context of an agile, open process was repeatedly emphasised during each case study, particularly in more interpretive developments. Agile development is supported by a number of new techniques, practices and tools have been developed; it tends to favour working solutions over future capabilities and encourages near-continuous engagement with users, non-specialist participants taking part in the development process, responding to changes in functional requirements as both the developer and the user increase their understanding of the problem space.

This approach emphasises test driven development (TDD), frequent testing of software during the development approach, rather than employing testing at the end of the main period of development - a late stage in the development process. Clear, measurable functional requirements are of importance, although the tests chosen may be revised flexibly at any point in the process. Outputs are tested frequently, during very short development iterations, which reduces the risk of 'drifting' away from core functional requirements.

Our work explores application of these principles to the development of structures designed for knowledge management (KM). These are generally aimed at a clearly defined problem-space. Functional requirements are derived from close association with users, domain information and evidence, in an ongoing iterative development process of agile development and review. We look towards a process of agile, test-driven ontology development.

3 IDENTIFYING SOFTWARE PLATFORMS FOR ONTOLOGY DEVELOPMENT

Various software have been evaluated, including CharGer (DeLugach, 2001), Protégé and custom software developed for non-expert use. A number of informal procedures for collaborative ontology development were sketched out, with particular focus on actively working with existing patterns of collaboration within the domains under investigation. The use of a variety of tools implying very different levels of detail and technical accuracy or direct applicability implies a greater load on those responsible for completion of this progress.

However, certain tools offered greater accessibility for the non-expert. The corresponding benefit of this process was that, given an accessible, simplified surrogate, participants were able to quickly reach a level at which they were comfortable to discuss and contribute actively on issues such as the perceived quality, relevance or completeness of an ontology. The strawman surrogates under discussion, and the results of the discussion, are usually ambiguous, incomplete or contain invalid statements or assertions, so the raw results are indicative rather than directly applicable.

4 TESTS, METRICS AND EVALUATION

Various methods were identified and used for evaluation of ontologies. These can be compared to the general typology offered by Brank et al (2005):

- Comparison to a gold standard: an advantage of simplified representation is participants' ability to compare candidate structures to models originating within other knowledge management/ information retrieval domains.
- The use of the ontology within an application: test-driven approaches here were taken from the tradition of paper prototyping, including modifications of the card sorting methodology and use of a 'storyboarding' approach to facilitate discussion of the developed knowledge structure within a practical context of use of relevance to the participant.
- Comparison with a source of data: in each use case, data and text mining offered the likelihood of retrieving useful information. In the former case (textual resources), various automated methods were applied in order to retrieve information of a mostly statistical nature; this, it was felt, is primarily useful for refinements such as ranking options or identifying areas that are likely to contain weaknesses. In the ubiquitous computing domain, however, much the data available is more closely tied to a relatively static set of physical characteristics, notably geographic/positional information, and hence may be applied much more directly in the ontology development process.
- Evaluation by humans: how well an ontology meets pre-defined criteria. The definition of those criteria is a complex process in itself – indeed, the development process mentioned in this extended abstract is as concerned with exploring potential criteria for success as it is with development of structures designed to correspond with that vision. Brank et al. note that this process is particularly suitable to structural and syntactic levels, which corresponds well to our observations.

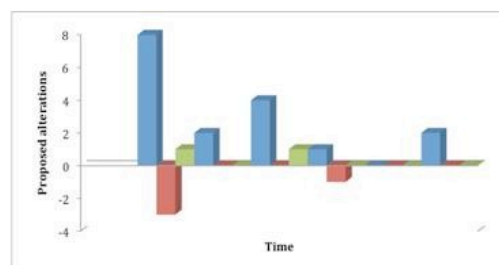


Figure 1. Data suggesting a trend towards stability with additions, pruning and structural changes. (Case Study 1)

During the case studies, a trend towards increasing structural stability has additionally been observed, particularly in the latter case. That is, the quantity, severity and nature of requested changes has been observed in general to diminish over time, suggesting that, as Viegas et al (2004) observe in their exploration of the stability of collaboratively-created user content, the first users to work on a given structure generally sets the tone of the result and, therefore, their work usually has the highest survival rate (see Figure 1).

5 CONCLUSION

The work described here essentially attempts to explore how a specialised and complex process may be rendered more accessible to a wider audience of potential users and contributors. Exploring the boundaries between ontology development and related areas of research may, as suggested by Brank et al., permit participants in a development process may integrate insights from related domains. We found that prototyping approaches using simple surrogates are useful in encouraging discussion and input, and that the relevance of data sources is dependent on the area, aspirations and contexts of use. The usability engineer is seldom considered a key participant in ontology development, but in cases in which ontology development becomes a collaborative (usually computer-supported) process, the participants become a key element in the success or failure of that development process.

REFERENCES

- [1] Aldea, A., Bañares-alcántara, R., Bocio, J., Gramajo, J. and Isern, D. (2003) An Ontology-Based Knowledge Management Platform. In: Proceedings of the Workshop on Information Integration on the Web (IIWeb-03) at the 18 th International Joint Conference on Artificial Intelligence. Retrieved Jan 20th, 2009 from <http://www.isi.edu/infoagents/workshops/ijcai03/papers/DIsern-article-ijcai.pdf>
- [2] Brank, J., Grobelnik, M., Mladenić, D. (2005). A Survey of Ontology Evaluation Techniques. Conference on Data Mining and Data Warehouses (SiKDD 2005), Ljubljana, Slovenia, 2005.
- [3] Carvalheira, Luiz C. C. and Gomi, Edson Satoshi (2007) A Method for Semi-automatic Creation of Ontologies Based on Texts. Advances in Conceptual Modeling – Foundations and Applications. Springer.
- [4] Della Valle, Emanuele, Celino, Irene and Cerizza, Dario (2008). Agreeing While Disagreeing, a Best Practice for Business Ontology Development, In: Business Information Systems, Lecture Notes in Business Information Processing, Volume 7. ISSN 1865-1348 Springer Berlin Heidelberg.
- [5] Delugach, Harry (2001). CharGer: A graphical Conceptual Graph editor. In *CGTools Workshop Proceedings in connection with ICCS 2001*, Stanford, CA, 2001. URL:<http://www.cs.nmsu.edu/hdp/CGTOOLS/proceedings/index.html>, Online Access: July 2001.
- [6] Noy, Natalya F. and McGuinness, Deborah L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
- [7] Spyns, Peter, Meersman, Robert and Jarrar, Mustafa (2002). Data modelling versus Ontology engineering Retrieved Jan 20th, 2009 from <http://lsdis.cs.uga.edu/SemNSF/SIGMOD-Record-Dec02/Meersman.pdf>.
- [8] Tudorache, Tania, Noy, Natalya F., Tu, Samson, and Musen, Mark A. (2008) Supporting Collaborative Ontology Development in Protege. In *Proceedings of ISWC*.
- [9] Viégas, Fernanda B., Wattenberg, M. and Dave, Kushal (2004). Studying cooperation and conflict between authors with history flow visualizations, Proceedings of the SIGCHI conference on Human factors in computing systems, p.575-582, April 24-29, 2004, Vienna, Austria.

Studying Concept Shift in Political Ontologies

Shenghui Wang and Stefan Schlobach and Janet Takens and Wouter van Atteveldt¹

Abstract.

For some years now ontologies have been used in Social Science, e.g., in the annotation of newspaper articles for disambiguating concepts within Media Analysis. The static view of those ontologies becomes problematic when they are associated with instances over distant years. These ontologies and annotations have now become objects of study in their own right, as they implicitly represent the shift of meaning of political concepts over time. In this paper, we address the problem of concept shifting from different aspects: extensional correlations between concepts from the same and different time periods, as well as intensional links, provided by hierarchical information and manually built links between concepts. We carry out an empirical study of concept shift in a case-study from Communication Science on a corpus with ontologies describing the Dutch election campaigns since 1994. Different types of concept shift are identified and the preliminary results lead to more open questions which needs to be investigated further.

1 Introduction

Large datasets of coded textual content are often required to answer research questions in Communication Science. Traditionally, textual documents are annotated with controlled vocabularies (of increasing expressiveness). Gradually, communication scientists adopt the Semantic Web techniques in their coding routine [9]. The idea is to code the meaning of sentences and articles in a formalised graph representation (called NET) similar to RDF triples. In these triples actors and issues are taken from an ontology, and the predicate usually represents opinions and moods. Take as an example a sentence in a newspaper article during the Dutch election period in 2006.

Het Openbaar Ministerie (OM) wil de komende vier jaar mensen- handel uitroeien. (*The Justice Department (OM) wants to eliminate human trafficking within the next four years.*)

The sentence is coded as `<om, -1, human trafficking>`, where `om` and `human trafficking` are two concepts in the ontology, while `-1` indicates the Justice Department is negative about human trafficking. After the whole corpus is triplified, communication scientists then do the analysis over such RDF graphs.

However, these datasets are often comprised of textual content from different time periods or genres. Similar to the dataset created from newspapers during five recent Dutch election campaigns from 1994 to 2006, the Manifesto Research Group is creating a dataset of 2162 topic-coded party manifestos from 25 countries during the period from 1945 to 2006 [1, 4]. Another example is the Comparative Policy Agendas project, which is combining coded media content with manifestos, legislative texts, and government press statements

of the last decades [3]. Since large manual coding projects require a lot of time, energy and money, it is often only possible to create such data sets by combining the efforts of coders from different time periods, research groups or even countries.

In order to reach interoperability between the datasets coded with different ontologies, or during different periods of time or even from different countries, there are two issues which need to be addresses: *ontology merging* and *concept shift*:

Ontology merging. If different datasets are combined to create a larger dataset, the ontologies or coding schemas used in both datasets need to be the same. If they are different, they need to be merged by identifying which differently named fields correspond to each other, or, more likely, creating more general concepts which subsume concepts from both ontologies. For example, if the ontology of one dataset contains a concept `illegal immigration`, and the other contains `labour migrants`, we can merge the ontologies by creating a more general concept `immigration` that contains both concepts. However, if good mappings do exist, it can be difficult and time-consuming to create these mappings for large ontologies.

Concept shift. Even if data from different datasets is coded using the same ontology, or if a good mapping has been found between the different ontologies, there is the danger that the concepts have a different semantics in different datasets. Interpretation of these concepts can vary depending on the time period, the location or the cultural background of the coders, which leads to the problems of reliability [5, 6]. In this paper, we do not consider this reliability issue and assume the interpretation of concepts are more or less stable within a certain period of time. We focus on whether the semantics of concepts shift over time. For example, we study whether the concept `feminism` shift in its meaning from centuries ago to today. We will investigate methods which can detect and represent such shift.

2 Related work

The problem addressed here is related to a broader problem of *ontology change*, which refers to the “problem of deciding the modifications to perform an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements” [2]. Flouris *et al.* classified the problem of ontology change into 11 subfields, including *ontology evolution* and *ontology versioning*. The existing research on ontology evolution and versioning mainly addresses this problem at the macro ontological level, that is, the effect of certain change operations over the ontology elements, including concepts, relations and

¹ Vrije Universiteit Amsterdam, email: {swang, schlobac}@few.vu.nl, {JH.Taken, WH.van.Atteveldt}@fsw.vu.nl

instances, as well as the interoperability issue between different variants (versions) over time.

In this paper, we look at the change problem at a micro level, that is, the underlying semantics of concepts (which can also be extended to other elements of ontologies, such as relations). Under the same labels, the semantics of concepts may change over time. We expect this to be more likely the case for the extensional semantics, reflected by the contextual information associated with the concepts, which are sensitive to the time periods, locations, and even the cultural background of human coders. We propose to measure the changes of concept semantics to provide users a finer-grained access to the background knowledge represented in the form of a formal ontology.

3 What is concept shift?

In philosophy of language, the concepts *extension* and *intension* signify two different ways of defining the semantics of a concept. *Intensional* semantics define a concept by listing the defining properties of a concept (*i.e.* its intension), such as stating that a bird is an egg-laying vertebrate that can fly. For each object it can be determined whether it matches this definition by checking whether it has these properties. *Extensional* semantics, on the other hand, define a concept by listing the objects that match its definition (*i.e.* its extension) [7]. For example, the concept Dutch cities can be defined by listing Amsterdam, Rotterdam, The Hague, *etc.*

In the communication science study, the intension of a concept in the ontology stands for the core part of the concept that the communication scientists want to measure, while the extension is formed by all the instance contextual information assigned to the concept by the human coders. We state that concept shift occurs when *the extensional and intensional semantics of a concept differ, and the extensional semantics changes over time.*

4 How can we detect concept shift?

If the intensional semantics of a concept is relatively stable, then we can measure the extensional semantics using its extensions, that is, the instances associated with it. We propose to study concept shift by using extensional semantics to derive relations between concepts from the same years and different versions of the concepts used in different years. Combining with intensional information of the concepts, such as hierarchical information, we observe the subtle changes in meaning of the related concepts over time.

Extension mappings between different years Following our previous work [8] we use information retrieval techniques to calculate document similarity between annotated articles, which we use subsequently to identify similarity between concepts. As reported in [10], chains between the most similar of these concepts thus produce graph-like structures (lines, trees, or DAGs), which “tell their own” story of Dutch politics over the past 15 years.

Co-occurrence links within same years Since the extensional semantics of one concept are reflected by the instances which are associated with it, we can study the co-occurrence of this concept and other concepts. By comparing the overlap between instances, we can measure the strength of the association between concepts within the same year. In this way, if one concept is correlated with different concepts in different years, it is likely the extensional semantics of this concept may change between these years.

Hierarchical links within same years The intensional semantics of a concept can be defined in the ontology by its property and its structural association within the ontology. Unfortunately, the ontologies at hand have no intensional properties defining the concepts involved. We can only rely on little hierarchical information to obtain some indication of the intensional semantics of the concepts involved.

Once these three kinds of links are put together, we can generate a whole picture of the change of concept semantics over time.

5 Experiments

For our experiments we used 5 different ontologies from the Dutch election campaigns in 1994, 1998, 2002, 2003 and 2006. Our experiment were conducted by mapping each of those ontologies with each other. Each ontology was used manually to annotate (around 5000) newspaper articles of the respective campaign. There exists a manually created (intensional) alignment between these 5 ontologies, which gives us an opportunity to compare the difference between extensional and intensional semantics.

As reported in [10], the consistency between manual (intensional) and extensional mappings is not big, as the precision and recall in terms of the manual mappings are not satisfactory. However, it is not the goal to reproduce the manual mappings by using the extensional method, instead, we would like to study the differences between extensional and intensional mappings, which will say something about the concept shift in general.

Figure 1 gives two examples of such chains of concept shift. The three types of links introduced in the previous section together with manual mappings provided by the domain experts are combined and visualised in terms of different colours:

- blue links: extensional mappings between different years
- red links: manual (intensional) mappings between different years
- black links: co-occurrence links within the same years
- green links: hierarchical links within the same years, with the starting concept being the child concept.

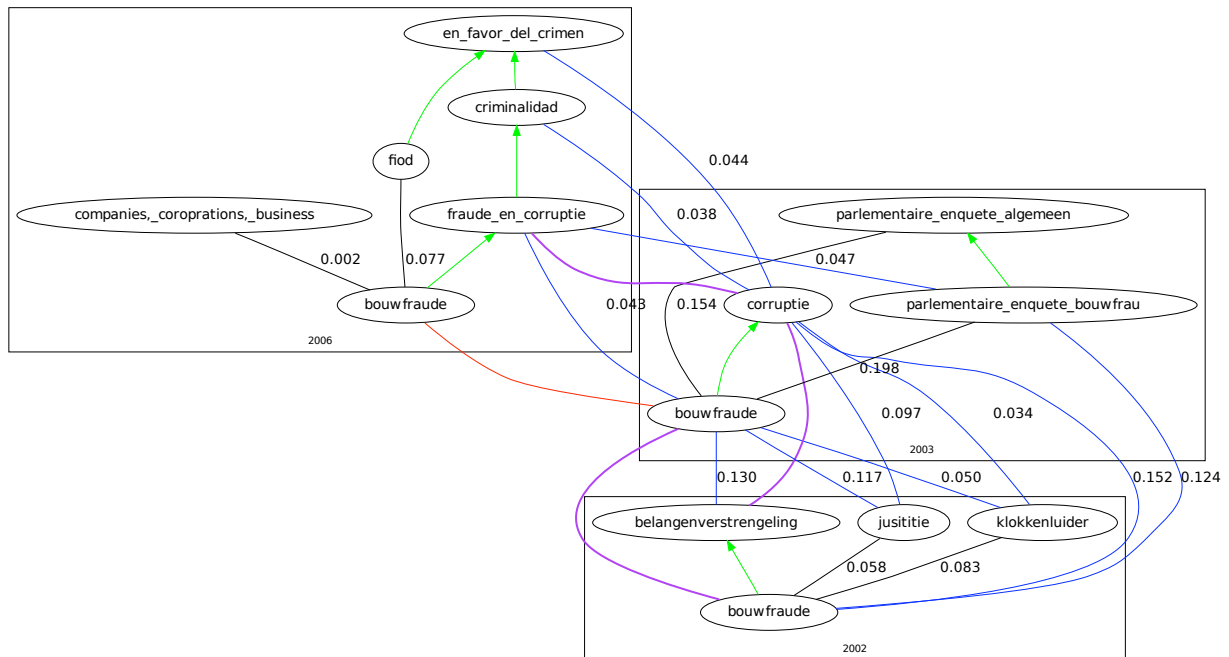
plus

- purple links: when extensional and intensional mappings agree with each other

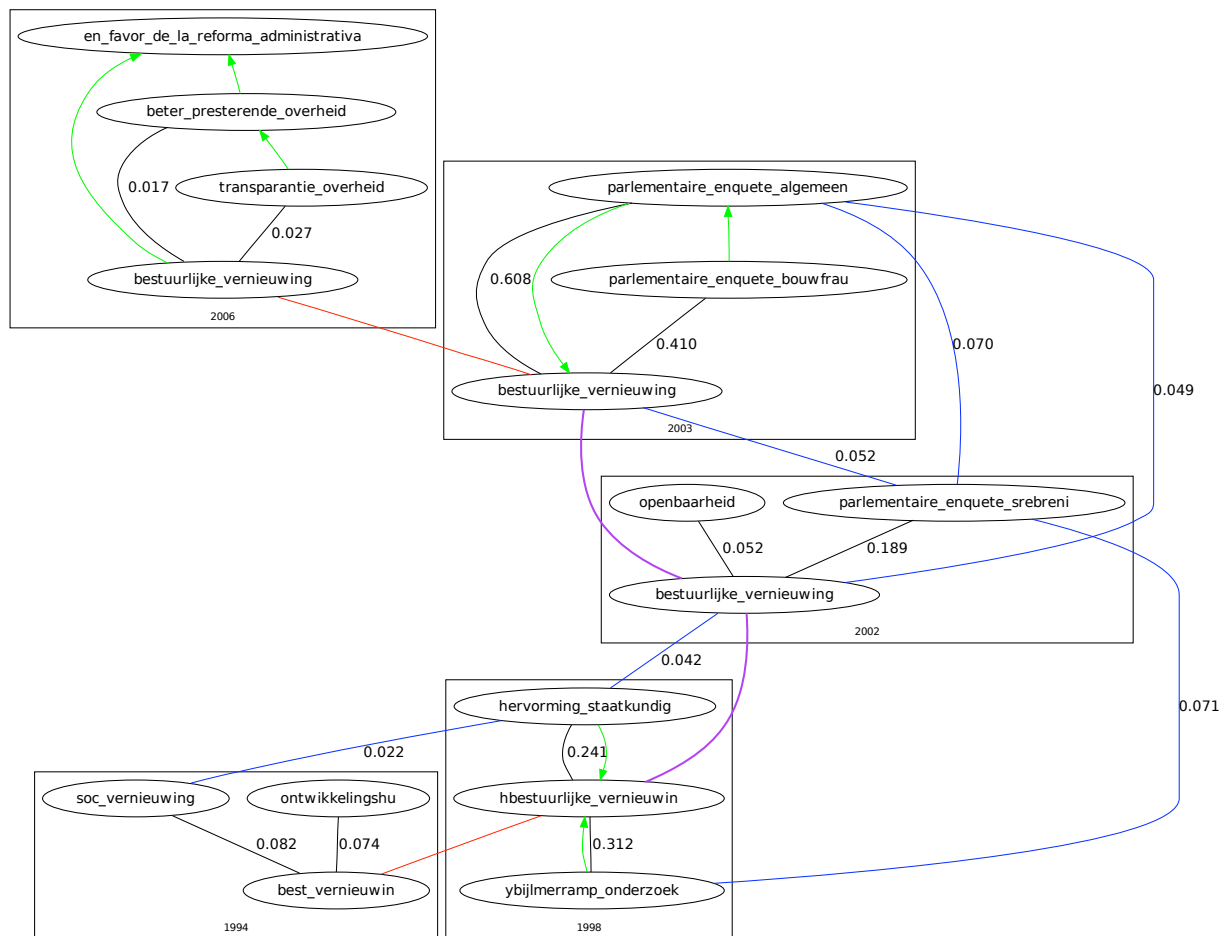
The number next to the links are the confidence value of such links. Manual mappings are considered to have a confidence value of 1, which is omitted in the figures.

What kinds of concept shift can be detected?

- association shifting
 - In 1994, *best vernieuwin* (administrative reforms) is strongly related to *ontwikkelingshu* (development aid) and *soc vernieuwing* (social reforms); while in 1998, it is related to *hervorming staatkundig* (state reforms) and *ybijmerramp onderzoek* (Inquiry in Bijlmer disaster). It suggests the concept “reforms” was about social reforms and development in 1994, but in 1998, when political parties mentioned “reforms”, they particular meant the reforms due to the disaster that happened in 1998. While again, in 2002, it is related to the right of expressing opinions about reforms (*inspraak*).



(a) Bouwfraude (builder's fraud)



(b) Bestuurlijke Vernieuwing (administrative reforms)

Figure 1. The change of concept semantics over time

- generalisation/specialisation of concepts
 - `bouwfraude` (builder's fraud) in 2003 is linked to `fraude en corruptie` (fraud and corruption) in 2006, which is the parent concept of `bouwfraude` — **this concept shifts to have a more general semantics**
 - The two equivalent concepts in 2002, `bestuurslijke vernieuwing` and `bestuurlijke vernieuwing` (administrative reforms), are both mapped to `parlementaire enquête algemeen` (parliamentary inquiry; general) in 2003, which is a child concept of `bestuurlijke vernieuwing` — **this concept shifts to have a more specific semantics**
- overlapping concepts
 - `vno.ncw` in 1994 is mapped to `rwerkgevers` in 1998, which co-occurs extensively with `rvno`, the intensional mapping of `vno.ncw` of 1994. — **this concept shifts to have a different but overlapping semantics**

[10] Shenghui Wang, Stefan Schlobach, Janet Takensl, and Wouter van Atteveldt, 'Mapping-chains for studying concept shift in political ontologies', in *Proceedings of the Ontology Matching workshop (OM 2009)*, Washington, USA, (October 2009).

6 Open questions

We have shown in [10] that studying a representative sample of mapping chains through our domain expert indicates clearly that mapping chains can be interesting given the reasonable quality of the extensional mappings our method can produce.

However, we have a few open questions which need to be investigated further:

- What types of concept shift can we identify and automatically detect?
- What is an appropriate (formal) representation for the detected shift?
- How can we evaluate the detected concept shift, both qualitatively and quantitatively?

REFERENCES

- [1] Ian Budge, Hans-Dieter Klingemann, Andrea Volkens, Judith Bara, Eric Tanenbaum with Richard C. Fording, Derek J. Hearl, Hee Min Kim, Michael McDonald, and Silvia Mendez, *Mapping Policy Preferences. Estimates for Parties, Electors, and Governments 1945-1998*, Oxford University Press, Oxford, 2001.
- [2] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, and Grigoris Antoniou, 'Ontology change: classification and survey', *Knowledge Eng. Review*, **23**(2), 117–152, (2008).
- [3] B. D. Jones and F. R. Baumgartner, *The politics of attention: How government prioritizes problems*, University of Chicago Press, Chicago, 2005.
- [4] Hans-Dieter Klingemann, Andrea Volkens, Judith Bara, Ian Budge, and Michael Macdonald, *Mapping Policy Preference II: Estimates for Parties, Electors and Governments in Eastern Europe, the European Union and the OECD, 1990-2003*, Oxford University Press, Oxford, 2006.
- [5] K Krippendorff, 'Reliability in content analysis', *Human Communication Research*, **30**, 411–433, (2004).
- [6] Matthew Lombard, Jennifer Snyder-Duch, and Cheryl Campanella Bracken, 'A call for standardization in content analysis reliability', *Human Communication Research*, **30**(3), 434–437, (2004).
- [7] A. S. Maida and S. C. Shapiro, 'Intensional concepts in propositional semantic networks', *Cognitive Science*, **6**(4), 291–330, (1982).
- [8] Balhasar Schopman, Shenghui Wang, and Stefan Schlobach, 'Deriving concept mappings through instance mappings', in *Proceedings of the 3rd Asian Semantic Web Conference*, Bangkok, Thailand, (2008).
- [9] Wouter Van Atteveldt, *Semantic Network Analysis: Techniques for Extracting, Representing, and Querying Media Content (dissertation)*, BookSurge, Charleston, SC, 2008.