In Vitro Study of Mapping Method Interactions in a Name Pattern Landscape

Ondřej Šváb, Vojtěch Svátek

University of Economics, Prague, Dep. Information and Knowledge Engineering,
Winston Churchill Sq. 4, 130 67 Praha 3, Prague, Czech Republic
svabo@vse.cz, svatek@vse.cz

Abstract. Ontology mapping tools typically employ combinations of methods, the mutual effects of which deserve study. We propose an approach to analysis of such combinations using synthetic ontologies. Initial experiments have been carried out for two string-based and one graph-based method. Most important target of the study was the impact of name patterns over taxonomy paths on the mapping results.

1 Introduction

Ideal taxonomy of ontology mapping methods clearly distinguishes different classes of methods such as string-based, graph-based or semantics-based, however most state-of-the-art systems nowadays exploit multiple approaches and combine them into (usually ad-hoc) workflows. Their results are then, e.g. within comparative tests such as OAEI[1], presented in their final form only; the contribution and mutual interaction of different constituent methods is not analysed.

In our current work we focused on the two classes of methods that are at the same time generally usable (unlike semantic methods, which depend on the presence of logical constraints in ontologies) and self-contained (unlike e.g. thesauri-based method, which depend on external thesauri): *string-based* and *graph-based*. In the first, proof-of-concept, series of experiments, we restricted our analysis to a single representative of graph-based methods plus two common string-based methods that can be used for its initialisation.

At the same time, we wanted to investigate the impact of *name patterns* that frequently follow the specialisation paths in ontologies[2]. The most typical patterns amount to the more specific (child) entity being right-, left- or internal extension of the more general (parent) entity. Clearly, the presence of such patterns could have influence on the effect of individual string matchers and, consequently, on the effect of the whole string-matching + graph-matching pipeline.

However, as we wanted to 'play' with different proportions and distributions of name patterns in the ontologies, as well as with different structural properties

[1] http://oaei.ontologymatching.org/

[2] We verified this assumption through informal analysis of about 50 ontologies 'randomly' picked from three popular ontology repositories.

of ontologies themselves, it was not convenient for us to stick to existing (real) ontologies. Instead, we made recourse to an 'in vitro' approach. We developed a *generator* that enabled us to automatically build 'ontologies' (syntactic OWL artefact) with customised structure and presence of name patterns as previously observed with real ontologies. Similarly to the 'calibration' collection of OAEI[3] we then derived the second model to be matched using pre-defined 'distortions'.

As the generated models lack any meaning and the 'distortions' are not necessarily identity-preserving, there cannot be any *reference mapping*. Our experimental results thus only consist in numerous observations about the strengths of mappings for different constellations of generated models and mapping methods, which we then attempt to interpret and generalise to some degree.

The structure of the paper is as follows. Section 2 very briefly characterises the mapping methods we took as testing material for our initial study. Section 3 describes the process of generating the synthetic 'ontologies' (as preliminary phase of our experiments). Section 4 is devoted to the actual mapping experiments and to case-by-case interpretation of their results. Section 5 contains some generalising discussion. Finally, section 6 reviews related work and section 7 outlines some future work.

2 Underlying Mapping Methods

The graph-based matching method, *similarity flooding* (SF), has been successfully tested on several mapping problems, especially those related to database schemata. The SF algorithm takes two models (database schemata, ontologies or the like) as graphs and transform them into a so-called *connectivity graph*, where each node (map pair) consists of two nodes from original models (one from each) that are connected with the same edges in their original models. Connectivity graph is then transformed into *propagation graph* where *propagation coefficients* are induced, expressing how well the similarity of a given map pair propagates to its neighbours and back [3].

As representatives of string-based methods we took two methods that merely differ in their granularity: the *Jaccard* method is token-based[4], while the *char-Jaccard* method is character-based (hence 'less semantic'). Both methods compute the similarity as the ratio of the intersection and union of the token/character sets of both entities.

[3] `http://oaei.ontologymatching.org/2007/benchmarks/`

[4] We also, originally, wanted to use the default (also token-based) string matcher contained in the SF package itself. It was quite interesting for our purposes, as it is (allegedly) based on common prefixes/suffixes of concept names—a feature relevant wrt. the name patterns we considered—but it produced apparently wrong results.

3 Generation of Synthetic Ontologies with Name Patterns

In order to swiftly obtain artifacts (in OWL language) syntactically similar to real ontologies, we developed a 'generator of ontologies'. The process of creating such an 'ontology' is guided by several *parameters*, such as the total number of classes, the length of the longest path, the maximum number of subclasses of one class etc. Parts of the process are 'random'[5], which concern the structure, labels (as 'reasonably long' sequences of random characters avoiding 'unpronounceable' sequences of more than two consonants), token delimiters, and, finally, name patterns. Currently we consider three patterns, all describing the relationship between the label of a parent and child in the taxonomy:

- (the most obvious) pattern 1 consists in left-hand side expansion of label, e.g. 'article' → 'scientificArticle'
- pattern 2 represents right-hand side expansion of label, e.g. 'IPR' → 'IPRTransfer' (example from the `ipronto.owl` from OntoSelect collection (`http://olp.dfki.de/ontoselect/`)),
- pattern 3 represents inward expansion of label, e.g. 'airMission' → 'airMovementMission' (example from the `ATO_Mission_Models.owl` ontology from DAML collection (`http://www.daml.org/ontologies/`).

In order to obtain pairs of ontologies for the mapping task, the generation process is followed with a *distortion* phase[6] where some classes are removed, some classes are added and/or there are changes in name patterns. In the section 4, there is list of all distortions that we tried.

Our generator enables us to prepare different ontologies with different features. For our current testing purposes we chose one synthetic ontology from a couple of diverse generated ontologies, see Figure 1. The chosen ontology has 24 classes, six of them being root classes. Only two root classes have children: `zutu` (for the sake of brevity we will shortcut it with Z) and `jonopizasfa` (J). J has two children and does not belong to any pattern. Z has three children and belongs to pattern 1. The occurence of this pattern among the children of Z is absolute (frequency is 100%), while in the case of its subclass `cexilipluZutu` the frequency of pattern 1 is 40%. J has subclass `dinicipi` (D) that has pattern 1 with 100% occurences in its children. In the experiments below, we took the classes J, Z and D as 'spy points' for observing the mapping results.

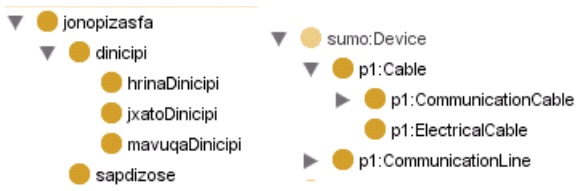
Although our ontology is synthetic, its structures are analogous to those of real ontologies. In Fig. 2, there is snippet of real ontology (`Communications.owl` from DAML collection) corresponding to J and in Fig. 3, there is snippet of the same real ontology corresponding to Z. Obviously, an ontology of this size could have been created purely by human effort, e.g. by manually tweaking

[5] I.e. governed by a sequence of pseudorandom numbers.

[6] In our first experiments, this phase was partly manual. Namely, for the current, rather small, testing model, it was much easier to do the small tweaks manually than to redesign the automated generation tool.

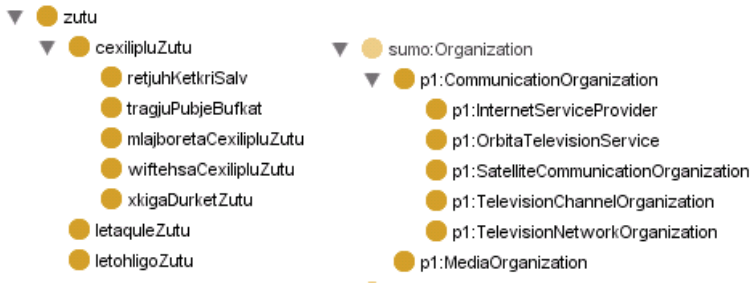
Fig. 1. Base synthetic ontology used in our experiments

real ontologies (which we actually did in prior research). We however anticipate further experiments where automatic generation would pay off, and also did not want to be distracted, during the result interpretation phase, by the human semantics of concrete labels.



(a) synthetic part (b) corresponding real part

Fig. 2. Part of real ontology corresponding to part of synthetic ontology



(a) synthetic part (b) corresponding real part

Fig. 3. Part of real ontology corresponding to part of synthetic ontology

4 Mapping Experiments

In our experiments we always tried to map the base synthetic ontology on some of its distortions. We observed the similarity values for three mappings: J with J (from the base ontology vs. from its distortion, in turn), Z with Z, and D with D. Their similarity values were computed with the similarity flooding algorithm; for initial mapping once the Jaccard method was used and once the charJaccard method.

We can roughly divide the distortions employed to the four groups below (particular distortions can belong to more than one group). They do not of course exhaustively represent all possibilities, nor are they very systematic; they rather reflect ad hoc 'polls' effected during a feedback-driven analysis process.

- (1) changes in structure, ie. adding/removing intermediate classes:
  - deletion of three root classes, except J and Z,
  - addition of some subclasses to root classes, except J and Z,
  - direct subclass of J was deleted,
  - direct subclass of J was deleted and all subclasses of D were also deleted,
  - deletion of direct subclasses of class D or Z,
  - all subclasses of D were relocated to root level,
- (2) changes in labels of classes:
  - two of three subclasses of D were relabelled in order to be different than original labels,
- (3) changes in delimiters in whole ontology (ceteris paribus); they are not considered below as they have no impact on similarity flooding,
- (4) changes in pattern, ie. altering the frequency of occurence of the pattern:
  - deletion of subclasses, that (does not) belong to pattern at class Z,
  - change pattern from pattern 1 to pattern 2,
  - relabeling all subclasses of Z so as to remove pattern.

Now we will present and discuss several 'interesting' situations (cases) with regard to the strengths of mappings for different constellations. Each case consists of textual description of distortions with regard to the original synthetic ontology (see Fig. 1), table with similarity values (charJaccard method was used for value in the first column, while Jaccard method was used for computation of value in the second column). All the cases are compared with the situation (initial case—Case #0) where the original synthetic ontology is mapped to itself: the absolute difference wrt. the initial case is always in parentheses. Some cases are additionally illustrated by figures. Finally, some tentative interpretations of observed results are formulated.

Case #1, see Fig. 4
Distortion: three root classes were deleted except J and Z.
Interpretation: Deletion of root classes leads to smaller connectivity graph, hence the effect of SF is smaller (higher similarity).

Case #2, see Fig. 5
Distortion: some classes were added to root classes, except to J and Z.
Interpretation: Addition of subclasses to root classes leads to larger connectivity graph, hence the effect of SF is greater (lower similarity).

Case #3, see Fig. 6 and 7 In this case we compare two situations - deletion of occurences of pattern or not.
Distortion: (a) three subclasses of `cexilipluZutu` were deleted, which had string Z in their labels (in Fig. 6 on the left-hand side); (b) two subclasses

mapping	charJaccard	Jaccard
J with J	0.28(+0.02)	0.44(+0.01)
Z with Z	0.52(+0.03)	0.82(+0.03)
D with D	0.49(+0.03)	0.77(+0.02)

(a) similarity table

(b) distortion

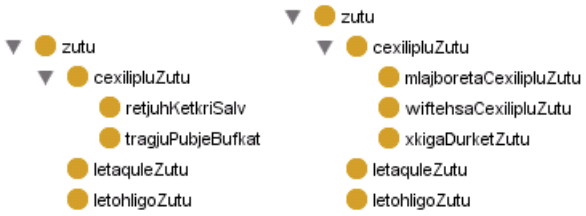
Fig. 4. Case #1

mapping	charJaccard	Jaccard
J with J	0.26(+0.00)	0.42(-0.01)
Z with Z	0.48(-0.01)	0.79(+0.00)
D with D	0.45(-0.01)	0.75(+0.00)

(a) similarity table

(b) distortion

Fig. 5. Case #2



(a) without Z (b) with Z

Fig. 6. Distortions for case #3

mapping	charJaccard	Jaccard
J with J	0.26(+0.00)	0.43(+0.00)
Z with Z	0.46(-0.03)	0.74(-0.05)
D with D	0.46(+0.00)	0.75(+0.00)

(a) similarity table without Z

mapping	charJaccard	Jaccard
J with J	0.26(+0.00)	0.43(+0.00)
Z with Z	0.47(-0.02)	0.77(-0.02)
D with D	0.46(+0.00)	0.75(+0.00)

(b) similarity table with Z

Fig. 7. Similarity tables for case #3

of `cexilipluZutu` were deleted, which had not string Z in their labels (in Fig. 6 on the right-hand side).
Interpretation: In both cases similarity values for mapping of classes Z decrease, but in the case on the right-hand side the value is higher. Higher values for the Z on the right-hand side situation could be strange because of greater connectivity graph. In this case (right-hand side) higher values can be explained with higher initial similarity values going from both string methods, because a name pattern is present.

Case #4, see Fig. 8
Distortion: Pattern 1 was changed to pattern 2 and one new subclass of D was added.
Interpretation: This change leads to greater connectivity graph, however higher similarity value is the effect of pattern at D.

Case #5, see Fig. 9
Distortion: All subclasses of D were relocated to root level.
Interpretation: It is similar example as in case #4 but now there is no pattern (by relocating classes pattern at D was aborted), thus greater connectivity graph leads to lower similarities at J and Z (similar to cases #1, and #2).

Case #6, see Fig. 10
Distortion: Z does not belong to pattern.

mapping	charJaccard	Jaccard
J with J	0.27(+0.01)	0.44(+0.01)
Z with Z	0.49(+0.00)	0.79(+0.00)
D with D	0.55(+0.09)	0.86(+0.11)

(a) similarity table

(b) distortion

Fig. 8. case #4

mapping	charJaccard	Jaccard
J with J	0.20(-0.06)	0.36(-0.07)
Z with Z	0.40(-0.09)	0.77(-0.02)
D with D	0.08(-0.38)	0.16(-0.59)

(a) similarity table

(b) distortion

Fig. 9. case #5

mapping	charJaccard	Jaccard
J with J	0.27(+0.01)	0.45(+0.02)
Z with Z	0.24(-0.25)	0.17(-0.62)
D with D	0.47(+0.01)	0.79(+0.04)

(a) similarity table

(b) distortion

Fig. 10. case #6

Interpretation: Lowering effect on Z because now there is no pattern. Higher effect (greater difference) in the case of Jaccard is expectable, because violation of prefixes and suffixes is more probable.

5 Discussion of Results

Our mapping experiments are in general consistent with the expectation that the SF algorithm computes lower similarity values for greater connectivity graphs. Greater connectivity graph means more candidate mapping pairs overall, and therefore there is provision for a high number of lower similarity values in neighbourhood of the map pair in question (eg. J with J), see cases #1, #2, and #5. In such a situation, the initially high similarity of the pair in question is largely redistributed to its neighbours in the connectivity graph.

Our experiments are also consistent with our hypothesis that the presence of name pattern leads to higher similarity values because of higher initial values going from both string based methods despite the size of connectivity graph (case #3, #4, and #6). This is produced by a relatively high degree of match among the neighbouring concepts pertaining to the same name pattern.

Cases #5, and #6 are also consistent with our hypothesis that the impact of name patterns on similarity values based on token-based string method Jaccard are higher than similarity values based on character-based string method charJaccard (differences between values in two column at appropriate tables). This naturally follows from the fact that patterns are represented as occurrences of same token within the labels of different neighbouring concepts.

Admittedly, the interpretations of current experiments suffer from our so far unsystematic, spontaneous approach to designing the experimental cases. However, we assume that the initial set of experiments will help creating such a systematic set of mapping experiments in the future.

6 Related Work

According to four (intertwined) main aspects of our work, we can distinguish among four areas of related research.

The effect of *combining string-based and graph-based mapping methods* was examined in several projects, e.g. [3, 1], however, essentially, in the sense of measuring the numerical improvement of quality (for a graph method applied on initial string-based mapping) using some kind of reference alignment. In contrast, we focus on detailed phenomena arising when applying the methods together, for concrete entities within their structural context, and without reference alignment. Additionally, in our own previous work we studied the interdependencies of several string-based methods alone, using probabilistic dependency models, namely Bayesian networks [4].

The analysis of *name patterns* along the ontology paths was carried out in numerous projects aiming at transformation of shallow ontologies such as thesauri or web directory headings into deeper ones (a recent representative is e.g. [2]). Our subject of study is however not transformation of the ontology but its mapping to another one; as this goal is less ambitious, our analysis is also much less thorough to date than the approaches mentioned.

The generation of *artificial ontologies* based on the features of real ones was envisaged by Tempich&Volz [6], for the purpose of creating benchmarks for semantic web reasoners. The difference of our generator lays in the focus on name patterns and local structural contexts rather than in large-scale statistical properties of ontologies. In addition, paper [6] does not contain any details about the actual generation process. As mentioned above, the distortion of original ontology in view of obtaining the two sides for matching experiments was carried out by J. Euzenat for the purpose of OAEI[7]. Our notion of distortion is at the same time finer grained and more restricted in its types. We also do not a priori assume that matching the original entity with its counterpart from the distorted model is 'correct' (even if its context and/or label are changed).

Finally, the problem of evaluating (or, rather, getting useful information from) ontology mapping results *without reference model* has also been investigated in our prior work [5]. We are not aware of any other major effort in this direction.

7 Conclusions and Future Work

We carried out a set of experiments with synthetic OWL artifacts ('ontologies' in syntactical sense) that should allow us get deeper insight into the behaviour of certain chosen mapping methods, applied in combination (in this case, sequence). An important feature we enforced to the synthetic models was the presence (to varying degrees) of name patterns that mimic those present in real ontologies. Rather than the concrete experimental results by themselves, the primary contribution of this initial study is meant to be the proof of concept of the whole

[7] `http://oaei.ontologymatching.org/2007/benchmarks/`

experimental setting, consisting in the 'ontology' generator (incl. the distortion component), interface to the scrutinised mapping methods, plus some (not yet too systematic) principles of human interpretation and generalisation of results.

Most imminent future work will consist in

- Doing a similar excercise for some other pairs of concepts in the structure (and possibly in differently generated 'ontologies'), including pairs that don't initially have the same label. Future experiments of this sort will be more systematic, possibly adhering to a clean formal model.
- Improvements of the generator, reflecting more sophisticated 'pattern' aspects (not confined to 'name' ones) of real OWL ontologies, and replacing the manual parts of the distortion phase with fully automatic ones (thus allowing for building large models if needed).
- Tests, possibly on the same data, of other string-based and graph-based methods. The latter could be e.g. probabilistic methods [1] or methods based on neighbour feature vector [7], provided we get access to their implementations.

In long term, we assume that the accumulated experience from testing current mapping approaches wrt. patterns in ontologies will help us design new mapping algorithms explicitly accounting for such patterns.

The research was partially supported by the IGA VSE grants no.12/06 "Integration of approaches to ontological engineering: design patterns, mapping and mining", no.20/07 "Combination and comparison of ontology mapping methods and systems", and by the Knowledge Web Network of Excellence (IST FP6-507482).

References

1. Doshi P., Thomas C.: Inexact Matching of Ontology Graphs Using Expectation-Maximization. In: Proc. AAAI 2006.
2. Hepp M., de Bruijn J.: GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies. In: Proc. ESWC 2007.
3. Melnik S., Garcia-Molina H., Rahm E.: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: Proc. 18th ICDE, 2002.
4. Šváb O., Svátek V.: Combining Ontology Mapping Methods Using Bayesian Networks. In: Proc. OM workshop at ISWC 2006.
5. Šváb O., Svátek V., Stuckenschmidt H.: A Study in Empirical and 'Casuistic' Analysis of Ontology Mapping Results. In: Proc. ESWC 2007.
6. Tempich C., Volz R.: Towards a benchmark for Semantic Web reasoners - an analysis of the DAML ontology library. In: EON Workshop at ISWC 2003.
7. Udrea O., Getoor L., Miller R. J.: Leveraging Data and Structure in Ontology Integration. In: Proceedings of ACM-SIGMOD 2007.