Semantic Enrichment of Ontology Mappings: Detecting Relation Types and Complex Correspondences

Patrick Arnold[*]
Universität Leipzig
arnold@informatik.uni-leipzig.de

ABSTRACT

While there are numerous tools for ontology matching, most approaches provide only little information about the true nature of the correspondences they discover, restricting themselves on the mere links between matching concepts. However, many disciplines such as ontology merging, ontology evolution or data transformation, require more-detailed information, such as the concrete relation type of matches or information about the cardinality of a correspondence (one-to-one or one-to-many). In this study we present a new approach where we denote additional semantic information to an initial ontology mapping carried out by a state-of-the-art matching tool. The enriched mapping contains the relation type (like equal, is-a, part-of) of the correspondences as well as complex correspondences. We present different linguistic, structural and background knowledge strategies that allow semi-automatic mapping enrichment, and according to our first internal tests we are already able to add valuable semantic information to an existing ontology mapping.

Keywords

ontology matching, relation type detection, complex correspondences, semantic enrichment

1. INTRODUCTION

Ontology matching plays a key role in data integration and ontology management. With the ontologies getting increasingly larger and more complex, as in the medical or biological domain, efficient matching tools are an important prerequisite for ontology matching, merging and evolution. There are already various approaches and tools for ontology matching, which exploit most different techniques like lexicographic, linguistic or structural methods in order to identify the corresponding concepts between two ontologies [16], [2]. The determined correspondences build a so-called alignment or ontology mapping, with each correspondence being a tripe (s, t, c), where s is a concept in the source ontology, t a concept in the target ontology and c the confidence (similarity).

These tools are able to highly reduce the effort of manual ontology mapping, but most approaches solely focus on detecting the matching pairs between two ontologies, without giving any specific information about the true nature of these matches. Thus, a correspondence is commonly regarded an equivalence relation, which is correct for a correspondence like (zip code, postal code), but incorrect for correspondences like (car, vehicle) or (tree trunk, tree), where is-a resp. part-of would be the correct relation type. This restriction is an obvious shortcoming, because in many cases a mapping should also include further kinds of correspondences, such as is-a, part-of or related. Adding these information to a mapping is generally beneficial and has been shown to considerably improve ontology merging [13]. It provides more precise mappings and is also a crucial aspect in related areas, such as data transformation, entity resolution and linked data.

An example is given in Fig. 1, which depicts the basic idea of our approach. While we get a simple alignment as input, with the mere links between concepts (above picture), we return an enriched alignment with the relation type annotated to each correspondence (lower picture). As we will point out in the course of this study, we use different linguistic methods and background knowledge in order to find the relevant relation type. Besides this, we have to distinguish between simple concepts (as "Office Software") and complex concepts, which contain itemizations like "Monitors and Displays", and which need a special treatment for relation type detection.

Another issue of present ontology matchers is their restriction to (1:1)-correspondences, where exactly one source concept matches exactly one target concept. However, this can occasionally lead to inaccurate mappings, because there may occur complex correspondences where more than one source element corresponds to a target element or vice versa, as the two concepts *first name* and *last name* correspond to a concept *name*, leading to a (2:1)-correspondence. We will show in section 5 that distinguishing between one-to-one and one-to-many correspondences plays an important role in data transformation, and that we can exploit the results from the relation type detection to discover such complex matches in a set of (1:1)-matches to add further knowledge to a mapping.

In this study we present different strategies to assign the relation types to an existing mapping and demonstrate how

[*]

25th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 28.05.2013 - 31.05.2013, Ilmenau, Germany.

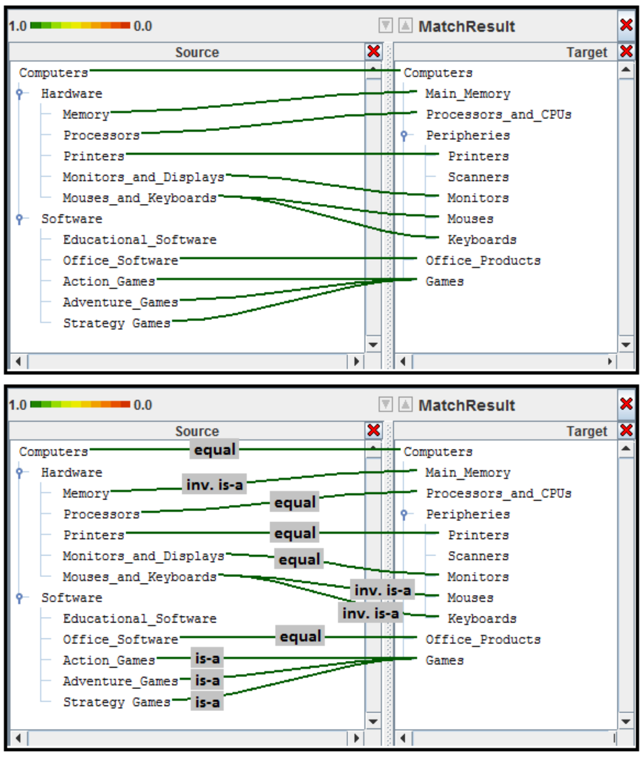


Figure 1: Input (above) and output (below) of the Enrichment Engine

complex correspondences can be discovered. Our approach, which we refer to as Enrichment Engine, takes an ontology mapping generated by a state-of-the-art matching tool as input and returns a more-expressive mapping with the relation type added to each correspondence and complex correspondences revealed. According to our first internal tests, we recognized that even simple strategies already add valuable information to an initial mapping and may be a notable gain for current ontology matching tools.

Our paper is structured as follows: We discuss related work in section 2 and present the architecture and basic procedure of our approach in section 3. In section 4 we present different strategies to determine the relation types in a mapping, while we discuss the problem of complex correspondence detection in section 5. We finally conclude in section 6.

2. RELATED WORK

Only a few tools and studies regard different kinds of correspondences or relationships for ontology matching. S-Match [6][7] is one of the first such tools for "semantic ontology matching". They distinguish between equivalence, subset (is-a), overlap and mismatch correspondences and try to provide a relationship for any pair of concepts of two ontologies by utilizing standard match techniques and background knowledge from WordNet. Unfortunately, the result mappings tend to become very voluminous with many correspondences per concept, while users are normally interested only in the most relevant ones.

Taxomap [11] is an alignment tool developed for the geographic domain. It regards the correspondence types equivalence, less/more-general (is-a / inverse is-a) and is-close ("related") and exploits linguistic techniques and background sources such as WordNet. The linguistic strategies seem rather simple; if a term appears as a part in another term, a more-general relation is assumed which is not always the case. For example, in Figure 1 the mentioned rule holds for the correspondence between *Games* and *Action_Games*, but not between $Monitors$ and $Monitors\_and\_Displays$. In [14], the authors evaluated Taxomap for a mapping scenario with 162 correspondences and achieved a recall of 23 % and a precision of 89 %.

The LogMap tool [9] distinguishes between equivalence and so-called weak (subsumption / is-a) correspondences. It is based on Horn Logic, where first lexicographic and structural knowledge from the ontologies is accumulated to build an initial mapping and subsequently an iterative process is carried out to first enhance the mapping and then to verify the enhancement. This tool is the least precise one with regard to relation type detection, and in evaluations the relation types were not further regarded.

Several further studies deal with the identification of semantic correspondence types without providing a complete tool or framework. An approach utilizing current search engines is introduced in [10]. For two concepts A, B they generate different search queries like "A, such as B" or "A, which is a B" and submit them to a search engine (e.g., Google). They then analyze the snippets of the search engine results, if any, to verify or reject the tested relationship. The approach in [15] uses the Swoogle search engine to detect correspondences and relationship types between concepts of many crawled ontologies. The approach supports equal, subset or mismatch relationships. [17] exploits reasoning and machine learning to determine the relation type of a correspondence, where several structural patterns between ontologies are used as training data.

Unlike relation type determination, the complex correspondence detection problem has hardly been discussed so far. It was once addressed in [5], coming to the conclusion that there is hardly any approach for complex correspondence detection because of the vast amount of required comparisons in contrast to (1:1)-matching, as well as the many possible operators needed for the mapping function. One key observation for efficient complex correspondence detection has been the need of large amounts of domain knowledge, but until today there is no available tool being able to semi-automatically detect complex matches.

One remarkable approach is iMAP [4], where complex matches between two schemas could be discovered and even several transformation functions calculated, as $RoomPrice = RoomPrice*(1+TaxPrice)$. For this, iMAP first calculates (1:1)-matches and then runs an iterative process to gradually combine them to more-complex correspondences. To justify complex correspondences, instance data is analyzed and several heuristics are used. In [8] complex correspondences were also regarded for matching web query interfaces, mainly exploiting co-occurrences. However, in order to derive common co-occurrences, the approach requires a large amount of schemas as input, and thus does not appear appropriate for matching two or few schemas.

While the approaches presented in this section try to achieve both matching and semantic annotation in one step, thus often tending to neglect the latter part, we will demonstrate a two-step architecture in which we first perform a schema mapping and then concentrate straight on the enrichment of the mapping (semantic part). Additionally, we want to analyze several linguistic features to provide more qualitative mappings than obtained by the existing tools, and finally develop an independent system that is not restricted to schema and ontology matching, but will be differently exploitable in the wide field of date integration and data analysis.

3. ARCHITECTURE

As illustrated in Fig. 2 our approach uses a 2-step architecture in which we first calculate an ontology mapping (match result) using our state-of-the-art matching tool COMA 3.0 (step 1) [12] and then perform an enrichment on this mapping (step 2).

Our 2-step approach for semantic ontology matching offers different advantages. First of all, we reduce complexity compared to 1-step approaches that try to directly determine the correspondence type when comparing concepts in O_1 with concepts in O_2. For large ontologies, such a direct matching is already time-consuming and error-prone for standard matching. The proposed approaches for semantic matching are even more complex and could not yet demonstrate their general effectiveness.

Secondly, our approach is generic as it can be used for different domains and in combination with different matching tools for the first step. We can even re-use the tool in different fields, such as entity resolution or text mining. On the other hand, this can also be a disadvantage, since the enrichment step depends on the completeness and quality of the initially determined match result. Therefore, it is important to use powerful tools for the initial matching and possibly to fine-tune their configuration.

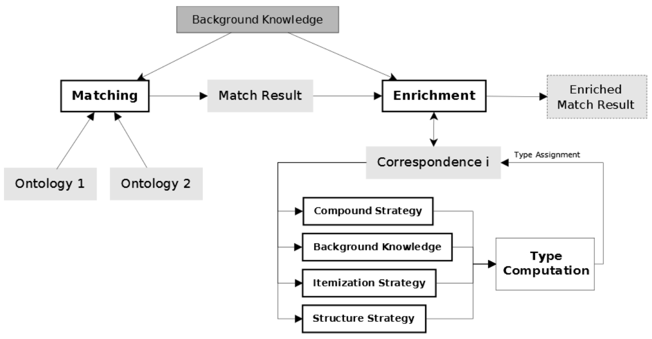


Figure 2: Basic Workflow for Mapping Enrichment

The basics of the relation type detection, on which we focus in this study, can be seen in the right part of Fig. 2. We provide 4 strategies so far (Compound, Background Knowledge, Itemization, Structure), where each strategy returns the relation type of a given correspondence, or "undecided" in case no specific type can be determined. In the Enrichment step we thus iterate through each correspondence in the mapping and pass it to each strategy. We eventually annotate the type that was most frequently returned by the strategies (type computation). In this study, we regard 4 distinct relation types: equal, is-a and inv. is-a (composition), part-of and has-a (aggregation), as well as related.

There are two problems we may encounter when computing the correspondence type. First, all strategies may return "undecided". In this case we assign the relation type "equal", because it is the default type in the initial match result and possibly the most likely one to hold. Secondly, there might be different outcomes from the strategies, e.g., one returns is-a, one equal and the others undecided. There are different ways to solve this problem, e.g., by prioritizing strategies or relation types. However, we hardly discovered such cases so far, so we currently return "undecided" and request the user to manually specify the correct type.

At the present, our approach is already able to fully assign relation types to an input mapping using the 4 strategies, which we will describe in detail in the next section. We have not implemented strategies to create complex matches from the match result, but will address a couple of conceivable techniques in section 5.

Strategy	equal	is-a	part-of	related
Compounding		X		
Background K.	X	X	X	X
Itemization	X	X		
Structure		X	X	

Table 1: Supported correspondence types by the strategies

4. IMPLEMENTED STRATEGIES

We have implemented 4 strategies to determine the type of a given correspondence. Table 1 gives an overview of the strategies and the relation types they are able to detect. It can be seen that the Background Knowledge approach is especially valuable, as it can help to detect all relationship types. Besides, all strategies are able to identify is-a correspondences.

In the following let O_1, O_2 be two ontologies with c_1, c_2 being two concepts from O_1 resp. O_2. Further, let $C = (c_1, c_2)$ be a correspondence between two concepts (we do not regard the confidence value in this study).

4.1 Compound Strategy

In linguistics, a compound is a special word W that consists of a head W_H carrying the basic meaning of W, and a modifier W_M that specifies W_H [3]. In many cases, a compound thus expresses something more specific than its head, and is therefore a perfect candidate to discover an is-a relationship. For instance, a blackboard is a board or an apple tree is a tree. Such compounds are called *endocentric compounds*, while *exocentric compounds* are not related with their head, such as buttercup, which is not a cup, or saw tooth, which is not a tooth. These compounds are of literal meaning (metaphors) or changed their spelling as the language evolved, and thus do not hold the is-a relation, or only to a very limited extent (like airport, which is a port only in a broad sense). There is a third form of compounds, called *appositional* or *copulative* compounds, where the two words are at the same level, and the relation is rather more-general (inverse is-a) than more-specific, as in Bosnia-Herzegowina, which means both Bosnia and Herzegowina, or bitter-sweet, which means both bitter and sweet (not necessarily a "specific bitter" or a "specific sweet"). However, this type is quite rare.

In the following, let A, B be the literals of two concepts of a correspondence. The Compound Strategy analyzes whether B ends with A. If so, it seems likely that B

is a compound with head A, so that the relationship B is-a A (or A inv. is-a B) is likely to hold. The Compound approach allows us to identify the three is-a correspondences shown in Figure 1 (below).

We added an additional rule to this simple approach: B is only considered a compound to A if $length(B) - length(A) \geq 3$, where $length(X)$ is the length of a string X. Thus, we expect the supposed compound to be at least 3 characters longer than the head it matches. This way, we are able to eliminate obviously wrong compound conclusions, like *stable* is a *table*, which we call *pseudo compounds*. The value of 3 is motivated by the observation that typical nouns or adjectives consist of at least 3 letters.

4.2 Background Knowledge

Background knowledge is commonly of great help in ontology matching to detect more difficult correspondences, especially in special domains. In our approach, we intend to use it for relation type detection. So far, we use WordNet 3.0 to determine the relation that holds between two words (resp. two concepts). WordNet is a powerful dictionary and thesaurus that contains synonym relations (equivalence), hypernym relations (is-a) and holonym relations (part-of) between words [22]. Using the Java API for WordNet Search (JAWS), we built an interface that allows to answer questions like "Is X a synonym to Y?", or "Is X a direct hypernym of Y?". The interface is also able to detect cohyponyms, which are two words X, Y that have a common direct hypernym Z. We call a correspondence between two cohyponyms X and Y related, because both concepts are connected to the same father element. For example, the relation between *apple tree* and *pear tree* is related, because of the common father concept *tree*.

Although WordNet has a limited vocabulary, especially with regard to specific domains, it is a valuable source to detect the relation type that holds between concepts. It allows an excellent precision, because the links in WordNet are manually defined, and contains all relation types we intend to detect, which the other strategies are not able to achieve.

4.3 Itemization

In several taxonomies we recognized that itemizations appear very often, and which cannot be processed with the previously presented strategies. Consider the correspondence ("books and newspapers", "newspapers"). The compound strategy would be mislead and consider the source concept a compound, resulting in the type "is-a", although the opposite is the case (inv. is-a). WordNet would not know the word "books and newspapers" and return "undecided".

Itemizations thus deserve special treatment. We first split each itemization in its atomic items, where we define an item as a string that does not contain commas, slashes or the words "and" and "or".

We now show how our approach determines the correspondence types between two concepts C_1, C_2 where at least one of the two concepts is an itemization with more than one item. Let I_1 be the item set of C_1 and I_2 the item set of C_2. Let w_1, w_2 be two words, with $w_1 \neq w_2$. Our approach works as follows:

1. In each set I remove each $w_1 \in I$ which is a hyponym of $w_2 \in I$.
2. In each set I, replace a synonym pair $(w_1 \in I, w_2 \in I)$ by w_1.
3. Remove each $w_1 \in I_1$, $w_2 \in I_2$ if there is a synonym pair (w_1, w_2).
4. Remove each $w_2 \in I_2$ which is a hyponym of $w_1 \in I_1$.
5. Determine the relation type:
   (a) If $I_1 = \emptyset, I_2 = \emptyset$: equal
   (b) If $I_1 = \emptyset, |I_2| \geq 1$: is-a
   If $I_2 = \emptyset, |I_1| \geq 1$: inverse is-a
   (c) If $|I_1| \geq 1, I_2 \geq 1$: undecided

The rationale behind this algorithm is that we remove items from the item sets as long as no information gets lost. Then we compare what is left in the two sets and come to the conclusions presented in step 5.

Let us consider the concept pair C_1 = "books, ebooks, movies, films, cds" and C_2 ="novels, cds". Our item sets are $I_1 = \{books, ebooks, movies, films, cds\}$, $I_2 = \{novels, cds\}$. First, we remove synonyms and hyponyms within each set, because this would cause no loss of information (steps 1+2). We remove $films$ in I_1 (because of the synonym *movies*) and *ebooks* in I_1, because it is a hyponym of *books*. We have $I_1 = \{books, movies, cds\}$, $I_2 = \{novels, cds\}$. Now we remove synonym pairs between the two item sets, so we remove *cds* in either set (step 3). Lastly, we remove a hyponym in I_1 if there is a hypernym in I_2 (step 4). We remove *novel* in I_2, because it is a *book*. We have $I_1 = \{books, movies\}$, $I_2 = \emptyset$. Since I_1 still contains items, while I_2 is empty, we conclude that I_1 specifies something more general, i.e., it holds C_1 inverse is-a C_2.

If neither item set is empty, we return "undecided" because we cannot derive an equal or is-a relationship in this case.

4.4 Structure Strategy

The structure strategy takes the structure of the ontologies into account. For a correspondence between concepts Y and Z we check whether we can derive a semantic relationship between a father concept X of Y and Z (or vice versa). For an is-a relationship between Y and X we draw the following conclusions:

- X equiv $Z \rightarrow Y$ is-a Z
- X is-a $Z \rightarrow Y$ is-a Z

For a part-of relationship between Y and X we can analogously derive:

- X equiv $Z \rightarrow Y$ part-of Z
- X part-of $Z \rightarrow Y$ part-of Z

The approach obviously utilizes the semantics of the intra-ontology relationships to determine the correspondence types for pairs of concepts for which the semantic relationship cannot directly be determined.

4.5 Comparison

We tested our strategies and overall system on 3 user-generated mappings in which each correspondence was tagged with its supposed type. After running the scenarios, we checked how many of the non-trivial relations were detected by the program. The 3 scenario consisted of about 350

.. 750 correspondences. We had a German-language scenario (product catalogs from online shops), a health scenario (diseases) and a text annotation catalog scenario (everyday speech).

Compounding and Background Knowledge are two independent strategies that separately try to determine the relation type of a correspondence. In our tests we saw that Compounding offers a good precision (72 .. 97 %), even without the many exocentric and pseudo-compounds that exist. By contrast, we recognized only moderate recall, ranging from 12 to 43 %. Compounding is only able to determine is-a relations, however, it is the only strategy that invariably works.

Background Knowledge has a low or moderate recall (10 .. 50 %), depending on the scenario at hand. However, it offers an excellent precision being very close to 100 % and is the only strategy that is able to determine all relation types we regard. As matter of fact, it did not work on our German-language example and only poorly in our health scenario.

Structure and Itemization strategy depend much on the given schemas and are thus very specific strategies to handle individual cases. They exploit the Compound and Background Knowledge Strategy and are thus not independent. Still, they were able to boost the recall to some degree.

We realized that the best result is gained by exploiting all strategies. Currently, we do not weight the strategies, however, we may do so in order to optimize our system. We finally achieved an overall recall between 46 and 65 % and precision between 69 and 97 %.

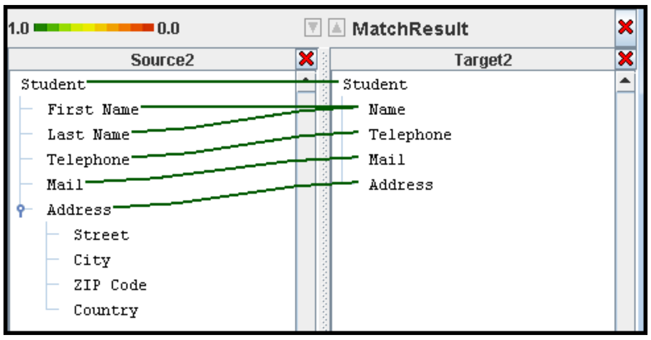
5. COMPLEX CORRESPONDENCES

Schema and ontology matching tools generally calculate (1:1)-correspondences, where exactly one source element matches exactly one target element. Naturally, either element may take part in different correspondences, as in (name, first name) and (name, last name), however, having these two separate correspondences is very imprecise and the correct mapping would rather be the single correspondence ((first name, last name), (name)). These kind of matches are called complex correspondences or one-to-many correspondences.

The disambiguation between a complex correspondence or 2 (or more) one-to-one correspondences is an inevitable premise for data transformation where data from a source database is to be transformed into a target database, which we could show in [1]. Moreover, we could prove that each complex correspondence needs a transformation function in order to correctly map data. If elements are of the type string, the transformation function is normally concatenation in (n:1)-matches and split in (1:n)-matches. If the elements are of a numerical type, as in the correspondence ((costs), ((operational costs), (material costs), (personnel costs))), a set of numerical operations is normally required.

There are proprietary solutions that allow to manually create transformation mappings including complex correspondences, such as Microsoft Biztalk Server [19], Altova MapForce [18] or Stylus Studio [20], however, to the best of our knowledge there is no matching tool that is able to detect complex correspondences automatically. Next to relation type detection, we therefore intend to discover complex correspondences in the initial mapping, which is a second important step of mapping enrichment.

We already developed simple methods that exploit the structure of the schemas to transform several (1:1)-correspondences into a complex correspondence, although these approaches will fail in more intricate scenarios. We used the structure of the schemas and the already existing (1:1)-matches to derive complex correspondences. Fig. 3 demonstrates this approach. There are two complex correspondences in the mapping, ((First Name, Last Name), (Name)) and ((Street, City, Zip Code, Country), Address), represented by simple (1:1)-correspondences. Our approach was able to detect both complex correspondences. The first one (name) was detected, because first name and last name cannot be mapped to one element at the same time, since the name element can only store either of the two values. The second example (address) is detected since schema data is located in the leaf nodes, not in inner nodes. In database schemas we always expect data to reside in the leaf nodes, so that the match (Address, Address) is considered unreasonable.



```
Student                Student
  First Name             Name
  Last Name              Telephone
  Telephone              Mail
  Mail                   Address
  Address
    Street
    City
    ZIP Code
    Country
```

Figure 3: Match result containing two complex correspondences (name and address)

In the first case, our approach would apply the concatenation function, because two values have to be concatenated to match the target value, and in the second case the split function would be applied, because the Address values have to be split into the address components (street, city, zip code, country). The user needs to adjust these functions, e.g., in order to tell the program where in the address string the split operations have to be performed.

This approach was mostly based on heuristics and would only work in simple cases. Now that we are able to determine the relation types of (1:1)-matches, we can enhance this original approach. If a node takes part in more than one composition relation (part-of / has-a), we can conclude that it is a complex correspondence and can derive it from the (1:1)-correspondences. For instance, if we have the 3 correspondences (day part-of date), (month part-of date), (year part-of date) we could create the complex correspondence ((day, month, year), date).

We have not implemented this approach so far, and we assume that detecting complex correspondences and the correct transformation function will still remain a very challenging issue, so that we intend to investigate additional methods like using instance data to allow more effectiveness. However, adding these techniques to our existing Enrichment Engine, we are able to present a first solution that semi-automatically determines complex correspondences, which is another step towards more precise ontology matching, and an important condition for data transformation.

6. OUTLOOK AND CONCLUSION

We presented a new approach to semantically enrich ontology mappings by determining the concrete relation type of a correspondence and detecting complex correspondences. For this, we developed a 2-step architecture in which the actual ontology matching and the semantic enrichment are strictly separated. This makes the Enrichment Engine highly generic so that it is not designed for any specific ontology matching tool, and moreover, can be used independently in various fields different from ontology matching, such as data transformation, entity resolution and text mining.

In our approach we developed new linguistic strategies to determine the relation type, and with regard to our first internal tests even the rather simple strategies already added much useful information to the input mapping. We also discovered that some strategies (Compounding, and to a less degree Itemization and Structure) are rather independent from the language of the ontologies, so that our approach provided remarkable results both in German and English-language ontologies.

One important obstacle is the strong dependency to the initial mapping. We recognized that matching tools tend to discover equivalence relations, so that different non-equivalence correspondences are not contained by the initial mapping, and can thus not be detected. It is future work to adjust our tool COMA 3.0 to provide a more convenient input, e.g., by using relaxed configurations. A particular issue we are going to investigate is the use of instance data connected with the concepts to derive the correct relation type if the other strategies (which operate on the meta level) fail. This will also result in a time-complexity problem, which we will have to consider in our ongoing research.

Our approach is still in a rather early state, and there is still much space for improvement, since the implemented strategies have different restrictions so far. For this reason, we will extend and fine-tune our tool in order to increase effectiveness and precision. Among other aspects, we intend to improve the structure strategy by considering the entire concept path rather than the mere father concept, to add further background knowledge to the system, especially in specific domains, and to investigate further linguistic strategies, for instance, in which way compounds also indicate the part-of relation. Next to relation type detection, we will also concentrate on complex correspondence detection in data transformation to provide further semantic information to ontology mappings.

7. ACKNOWLEDGMENT

This study was partly funded by the European Commission through Project "LinkedDesign" (No. 284613 FoF-ICT-2011.7.4).

8. REFERENCES

[1] Arnold P.: The Basics of Complex Correspondences and Functions and their Implementation and Semi-automatic Detection in COMA++ (Master's thesis), University of Leipzig, 2011.

[2] Bellahsene., Z., Bonifati, A., Rahm, E. (eds.): Schema Matching and Mapping, Springer (2011)

[3] Bisetto, A., Scalise, S.: Classification of Compounds. University of Bologna, 2009. In: The Oxford Handbook of Compounding, Oxford University Press, pp. 49-82.

[4] Dhamankar, R., Yoonkyong, L., Doan, A., Halevy, A., Domingos, P.: iMAP: Discovering Complex Semantic Matches between Database Schemas. In: SIGMOD '04, pp. 383–394

[5] Doan, A., Halevy, A. Y.: Semantic Integration Research in the Database Community: A Brief Survey. In AI Mag. (2005), pp. 83–94

[6] Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: An Algorithm and an Implementation of Semantic Matching. Proceedings of the European Semantic Web Symposium (2004), LNCS 3053, pp. 61–75

[7] Giunchiglia, F., Autayeu, A., Pane, J.: S-Match: an open source framework for matching lightweight ontologies. In: Semantic Web, vol. 3-3 (2012), pp. 307-317

[8] He, B., Chen-Chuan Chang, H., Han, J.: Discovering complex matchings across web query interfaces: A correlation mining approach. In: KDD '04, pp. 148–157

[9] Jiménez-Ruiz, E., Grau, B. C.: LogMap: Logic-Based and Scalable Ontology Matching. In: International Semantic Web Conference (2011), LNCS 7031, pp. 273–288

[10] van Hage, W. R., Katrenko, S., Schreiber, G. A Method to Combine Linguistic Ontology-Mapping Techniques. In: International Semantic Web Conference (2005), LNCS 3729, pp. 732–744

[11] Hamdi, F., Safar, B., Niraula, N. B., Reynaud, C.: TaxoMap alignment and refinement modules: Results for OAEI 2010. Proceedings of the ISWC Workshop (2010), pp. 212–219

[12] Massmann, S., Raunich, S., Aumueller, D., Arnold, P., Rahm, E. Evolution of the COMA Match System. Proc. Sixth Intern. Workshop on Ontology Matching (2011)

[13] Raunich, S.,Rahm, E.: ATOM: Automatic Target-driven Ontology Merging. Proc. Int. Conf. on Data Engineering (2011)

[14] Reynaud, C., Safar, B.: Exploiting WordNet as Background Knowledge. Proc. Intern. ISWCŠ07 Ontology Matching (OM-07) Workshop

[15] Sabou, M., d'Aquin, M., Motta, E.: Using the semantic web as background knowledge for ontology mapping. Proc. 1st Intern. Workshop on on Ontology Matching (2006).

[16] Shvaiko, P., Euzenat, J.: A Survey of Schema-based Matching Approaches. J. Data Semantics IV (2005), pp. 146–171

[17] Spiliopoulos, V., Vouros, G., Karkaletsis, V: On the discovery of subsumption relations for the alignment of ontologies. Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010), pp. 69-88

[18] Altova MapForce - Graphical Data Mapping, Conversion, and Integration Tool. `http://www.altova.com/mapforce.html`

[19] Microsoft BizTalk Server. `http://www.microsoft.com/biztalk`

[20] XML Editor, XML Tools, and XQuery - Stylus Studio. `http://www.stylusstudio.com/`

[21] Java API for WordNet Searching (JAWS), `http://lyle.smu.edu/~tspell/jaws/index.html`

[22] WordNet - A lexical database for English, `http://wordnet.princeton.edu/wordnet/`