Linguistic Analysis for Complex Ontology Matching

Dominique Ritze[1], Johanna Völker[1], Christian Meilicke[1], and Ondřej Šváb-Zamazal[2]

[1]University of Mannheim, [2]University of Economics, Prague
dritze@mail.uni-mannheim.de,
{johanna,christian}@informatik.uni-mannheim.de,
ondrej.zamazal@vse.cz

Abstract. Current ontology matching techniques focus on detecting correspondences between atomic concepts and properties. Nevertheless, it is necessary and possible to detect correspondences between complex concept or property descriptions. In this paper, we demonstrate how complex matching can benefit from natural language processing techniques, and propose an enriched set of correspondence patterns leveraging linguistic matching conditions. After elaborating on the integration of methods for the linguistic analysis of textual labels with an existing framework for detecting complex correspondences, we present the results of an experimental evaluation on an OAEI dataset. The results of our experiments indicate a large increase of precision as compared to the original approach, which was based on similarity measures and thresholds.

1 Introduction

Ontology matching can be considered one of the key technologies for efficient knowledge exchange and the successful realization of the Semantic Web. Bridging the gap between different terminological representations is an indispensable requirement for a large variety of tools and technologies including, for example, distributed reasoning, instance migration, and query rewriting in distributed environments.

In the past, ontology matching was commonly considered the task of detecting similar or equivalent concepts and properties in two ontologies. However, this view on ontology matching seems too narrow for many application scenarios, and new requirements have motivated several extensions to the original task definition. Among the challenges of the last Ontology Alignment Evaluation Initiative (OAEI) [4], for example, we find the task of instance matching as well as a track that aims at the generation of correspondences expressing subsumption (instead of equivalence) between concepts.

In our work, we suggest to extend the classical way of ontology matching in a different direction – the generation of correspondences between complex concept and property descriptions. We refer to these correspondences as *complex correspondences* and call the process of generating them as *complex ontology matching*. Our work is motivated by the insight that equivalence or even subsumption correspondences between atomic entities are often not applicable or not expressive enough to capture important dependencies between ontological entities.

This paper is based on our previous approach [11] which we found to have several disadvantages, including problems related to the precision of the patterns as well as the prerequisite of a reference alignment as additional input. In order to address these issues, we modified the original approach in the following way:

- A partial reference alignment is no longer required as input. We generate the alignment in a preprocessing step and use it as an anchor alignment later on.
- The complete logic required to express the conditions for generating correspondences is now described declaratively by means of XML. The XML-based specification of the matching conditions is interpreted and executed by our tool, while the concrete implementation remains transparent to the user.
- We changed the output format of our system so that it adheres to the *Expressive Declarative Ontology Alignment Language* (EDOAL) for complex correspondences, that is supported by the alignment API [3].
- We extended our approach by various methods for the linguistic analysis of concept or property labels. According to our experiments, the appropriate use of these methods results in a significantly increased precision.

The last point refers to the most essential contribution of this paper. In our previous approach, many matching conditions included similarity thresholds. Thanks to linguistic methods, we can now avoid the need for finding appropriate thresholds. In the remainder of this paper, we show that the use of these methods significantly improves the quality of complex matching. In particular, we find that the linguistic analysis enables us to achieve a significantly higher precision with respect to the previously detected [11] pattern instantiations. Moreover, we present a new correspondence pattern leveraging linguistic matching conditions and illustrate the advantages of the new approach by means of concrete examples.

Our paper is structured as follows. In Section 2 we describe our approach to complex matching. We introduce the terminology we adopt, and sketch the core elements of our algorithm. Section 3 discusses the related work, whereas in Section 4, we describe the linguistic methods that we apply to detect non-trivial semantic relations between concepts and properties. The pattern-specific matching conditions, which constitute the heart of our approach, are presented in Section 5. In Section 6, we report on our evaluation experiments, before concluding in Section 7.

2 Approach

Now we introduce the basics of our approach and explain the terminology we use in the subsequent sections. First of all, we adopt and slightly simplify the generic terminology defined in [6]. Thus, we understand an *alignment* between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ as a set of correspondences. A *correspondence* is a triple $\langle X, Y, Z\rangle$ where X is an entity from $\mathcal{O}_1$, Y is an entity from $\mathcal{O}_2$ and R is a relation such as equivalence or subsumption. Whenever it is required to refer to the origin of a specific concept, we write $C_{\#i}$ to indicate that C belongs to $\mathcal{O}_i$.

State-of-the-art ontology matching techniques are bound to detect correspondences as $\langle \mathit{Paper}, \mathit{Article}, \equiv\rangle$ or $\langle \mathit{writes}, \mathit{writesPaper}, \subseteq\rangle$. In the following we describe an ap-

proach that allows to detect correspondences where X and Y are complex concept or property descriptions.

Remember that the power of description logic originates from the ability to build complex concept and property descriptions from atomic ones, i.e. from concept and property names. The following listing shows some of the different ways to construct complex descriptions in description logics (here at the example of $\mathcal{SHOIN}$).

$\neg C$	*(atomic negation)*	$\forall P.C$	*(value restriction)*
$B \sqcap C$	*(conjunction)*	$\exists_{\leq n} P$	*(at least restriction)*
$B \sqcup C$	*(disjunction)*	$\exists_{\geq n} P$	*(at most restriction)*
$\{o_1, \ldots, o_n\}$	*(one of)*	P^{-1}	*(inverse property)*
$\exists P.C$	*(exists restriction)*		

In this listing, C refers to an arbitrary concept description and P refers to a property name. We define a correspondence $\langle X, Y, Z \rangle$, where X or Y is built according to one of the rules, as *complex correspondence*. An alignment that contains a *complex correspondence* as defined as a *complex alignment*. Note also that several of these rules can be applied sequentially according to their use in standard description logics.

In the following we will talk about *matching conditions* and *correspondence patterns*. A correspondence pattern describes a special type of complex correspondence. Suppose, for example, that our algorithms detects a correspondence $\langle \exists \mathit{earlyRegistered}.\{\mathit{true}\}, \mathit{EarlyRegisteredParticipant}, \equiv \rangle$. This correspondence is a concrete instantiation of the general correspondence pattern $\langle \exists P.\{\mathit{true}\}, C, \equiv \rangle$, where P and C denote variables. A correspondence pattern can coincide with one of the construction rules listed above, but can also be compounded of several rules and might contain constants, as shown in the example.

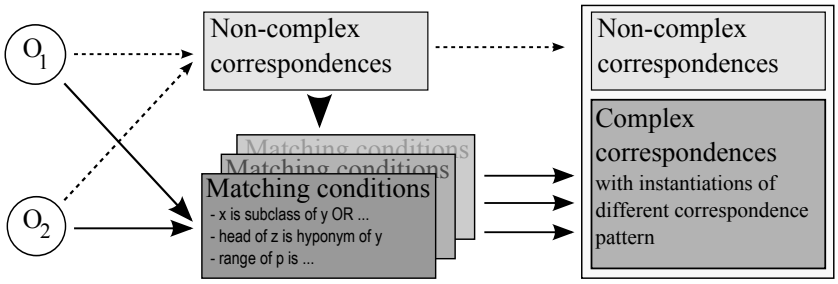

Fig. 1. Applying matching conditions to detect instances of correspondence patterns in order to generate a complex alignment.

In our approach we define for each correspondence pattern a set of matching conditions. If these conditions are fulfilled, we generate a concrete instantiation of the pattern and output a complex correspondence. The corresponding approach is depicted in Figure 1. First of all we generate by state of the art methods a non-complex alignment. This alignment is used as a kind of anchor that allows to check if certain relations hold

between entities of $\mathcal{O}_1$ and $\mathcal{O}_2$ such as "is concept C in $\mathcal{O}_1$ a subconcept of concept D in $\mathcal{O}_2$". The matching conditions used to detect a certain pattern comprise structural conditions as well as the linguistic conditions are presented in Section 4.

The following section reviews related work in the field of complex ontology matching as well as recent approaches to leveraging linguistic tools and resources in ontology matching.

3 Related Work

Considering the state-of-the-art in complex ontology matching, we find recent approaches to be distinguished by three key dimensions: the design, the representation and discovery of complex correspondences. In [12], complex correspondences are mainly considered in terms of design and representational aspects. The author proposes *alignment patterns*[1] as a solution for recurring mismatches raised during the alignment of two ontologies.[2] According to Scharffe [12], complex matching is a task that has to be performed by a human user (e.g., a domain expert), who can be supported by templates for capturing complex correspondences. However, similar patterns can also be exploited by automated matching approaches, as demonstrated in this paper. The alignment patterns from [12] are expressed in terms of EDOAL[3] [5], an extension of the alignment format proposed by [3]. It covers concept and property descriptions, concept restrictions, property value transformations, comparators for restriction over entities, and variables for representing ontology entities in patterns. In this paper, we adhere to this expressive language for capturing our correspondence patterns.

Šváb-Zamazal et al. [14] consider complex ontology matching as a use case for ontology transformation. An ontology is transformed in such a way that it can be more easily matched with other ontologies. Each transformation is performed by means of a transformation pattern containing several source and target ontology patterns as well as an appropriate pattern transformation, which captures the relationships between them. Each source ontology pattern specifies detection conditions such as structural and naming conditions. The authors argue that successful non-complex matching applied to the transformed ontology can be used for finding complex correspondences by tracking changes back to the original ontology. This approach, however, lacks experimentation.

Further work related to the discovery of complex correspondences relies on machine learning techniques such as Inductive Logic Programming, for example [9]. This type of approach takes correspondences with more than two atomic terms into account, but requires the ontologies to include matchable instances – a prerequisite that is not fulfilled in many application scenarios. The approach proposed in this paper does not require the existence of instances. Moreover, we can find related work in the field of database schema matching. In [2] the authors describe complex matching as the task of

[1] In the ODP taxonomy of patterns the notion of an *alignment pattern* is used instead of the notion of a *correspondence pattern*. In particular, correspondence patterns are considered as more general, having alignment patterns and reengineering patterns as subcategories. However, we stick to the terminology introduced in [6] where an alignment is a set of correspondences.

[2] These patterns are now being included within *OntologyDesignPatterns.org* (ODP).

[3] http://alignapi.gforge.inria.fr/edoal.html

finding corresponding composite attributes (e.g., a name is equivalent with concatenation of a first-name and a last-name). There are several systems dealing with this kind of database schema matching (e.g., [1]).

According to Euzenat and Shvaiko [6] *linguistic* approaches to ontology matching can be distinguished into language-based methods and methods which are based on linguistic resources, whereas the more general class of *terminological* approaches also includes string-based methods. The latter type of approach, i.e., similarity measures on the lexical layer of ontologies, is part of almost every state-of-the-art matcher. There is also a large body of work acknowledging the benefits of linguistic resources such as WordNet when it comes to detecting lexical-semantic relations between concept or property labels (see [7] for an overview). In addition, the low coverage of WordNet in certain application domains has motivated the development of methods which leverage more implicit evidence for those relationships [15], and of methods based on distributional similarities which can be computed from textual information associated with ontology entities [10, 8]. Only very few matchers, however, make use of natural language processing techniques that go beyond tokenization and lemmatization [17].This paper highlights the potential that lies within linguistic and in particular language-based methods for ontology alignment.

4 Linguistic Analysis

In order to facilitate the integration of state-of-the-art tools and resources for natural language processing into the matcher, we developed LiLA[4] (*Linguistic Label Analysis*) – a Java-based framework for the linguistic analysis of class and property labels which provides a single uniform interface to the following open-source tools:

JWNL (version 1.4.1)[5] is a programing interface for accessing the WordNet dictionary (version 3.0)[6] which contains information about more than 200,000 English words and their lexical semantic relationships.

OpenNLP (version 1.3.0)[7] is a framework for linguistic analysis including, for instance, components for determining the lexical categories of words (e.g., *adjective*).

MorphAdorner (version 1.0)[8] is a text processing framework which amongst other components provides means for morphological analysis and generation, i.e., inflection of words.

LexParser (version 1.6.3)[9] also known as the *Stanford Parser* is a syntactic parser which can be used to determine the grammatical structure of phrases (e.g., noun phrases such as "accepted paper") or sentences.

[4] http://code.google.com/p/lila-project/
[5] http://sourceforge.net/projects/jwordnet/
[6] http://wordnet.princeton.edu
[7] http://opennlp.sourceforge.net
[8] http://morphadorner.northwestern.edu
[9] http://nlp.stanford.edu/software/lex-parser.shtml

In addition, LiLA features a simple word sense disambiguation component and a spell checker. The remainder of this section illustrates the core functionalities of LiLA by virtue of a noun phrase, which serves as a running example.

paper written by clever students

Part-of-Speech Tagging. Each word in natural language belongs to a syntactic category (or *part-of-speech*), which defines its basic syntactic behavior. Accordingly, a part-of-speech tagger is a linguistic processing component for assigning appropriate syntactic categories to a given set of words. While in principle, each POS tagger may use its own set of category labels (*tags*), tag sets such as the *Penn Treebank Tag Set*[10] for English are widely used, and certain naming conventions have emerged as quasi-standards. Here, `NN` and `NNS` denote common nouns (singular or plural, respectively), `IN` stands for a preposition, `JJ` indicates an adjective and `VBN` is the tag for a past participle verb.

paper [`NN`] *written* [`VBN`] *by* [`IN`] *clever* [`JJ`] *students* [`NNS`]

Morphological Analysis. The field of morphology is concerned with the internal structure of words, more precisely the morphological rules for inflection and word-formation that enable humans to build a rich vocabulary from a basic inventory of *morphemes* – the smallest units in natural language that carry meaning. Each word consists of one or more morphemes. Words that are built from more than one morpheme can be split into a *stem* and an *affix*, i.e., a morph attached to a stem like "student"+"s", for example. In this case, the plural "s" is an inflectional morpheme, which alters the base form (also called *lexeme*) without changing its syntactic category. A component which reduces each word to its *lemma* (i.e., the canonical form of a lexeme which is typically included in the lexicon of a language) is called a *lemmatizer*.

LiLA relies upon the MorphAdorner framework for performing both lemmatization and morphological synthesis, i.e., the generation of specific word forms (e.g., "students") from lexemes (e.g., "student"). This also works well for irregular verbs such as "write" for which we are able to generate, for instance, the past participle ("written") by means of *conjugation*. This way LiLA can convert between singular and plural of the same noun (*declination*), as well as between active and passive *voice* or different tenses of a given verb. The functionality to obtain derivations such as *nominalizations* of verbs (e.g., "accept" and "acceptance"), for example, is provided by JWPL.

Lexical Semantic Analysis. *Lexical semantics* is the branch of linguistics that studies the meaning of words and their relationships. The popular lexical database of WordNet, which covers a wide range of such lexical semantic relations, is queried by LiLA through the API of JWPL. Thus, given a word such as "clever" or "student", LiLA can get access to detailed information about the possible meanings of the word (see *homonymy*), as well as about its synonyms, hyponyms, antonyms and otherwise related senses (e.g., meronyms).

Synonymy, at least *true* synonymy, is rarely found in natural language. However, there are many so-called *near-synonyms* (or *plesionyms*), i.e., words that share a common

[10] http://www.cis.upenn.edu/ treebank/

meaning in a given context. Hence, two words are considered synonyms (or near-synonyms) if they can be exchanged for one another in a sentence without altering its truth conditions (e.g., "student" and "scholar").

Homonymy and polysemy are types of semantic ambiguity. Two words are considered *homonymous* if they are spelled (*homograph*) and pronounced (*homophone*) in the same way, while having distinct meanings (or *senses*). Homonyms with related meanings, are called (regular) *polysemes* (e.g., "paper" as a substance or a writing sheet made thereof). In case a query posed to the WordNet API returns multiple senses for a given concept or property label, LiLA's **word sense disambiguation** component selects the most likely sense based on a vector-based representation of the current lexical context.[11]

Hyponymy is a kind of subordination relating one lexical unit to another one with a more general sense. The former is then called a *hyponym*, whereas the latter, i.e., the superordinate, represents the *hypernym*. Like meronymy, hyponymy is only transitive within one and the same category (e.g., functional). A verb which is more specific than another verb is sometimes called *troponym* (e.g., "write" and "create").

Antonymy is a kind of oppositeness that mostly holds between adjectives, but also some verbs and even nouns can be considered antonyms if they exhibit opposite semantic traits. One can distinguish between different types of antonyms such as *gradable* ("early" and "late"), *complementary* ("acceptable" and "unacceptable") and *relational* ("student" and "professor") antonyms.

Syntactic Parsing in computational linguistics typically refers to the analysis of syntactic structures. Each parsing algorithm relies upon a certain grammar, that is a formalism developed to describe the syntactically well-formed structures in a given language. Essentially, two types of grammars – dependency grammars and phrase structure grammars – have emerged as the most wide-spread means to analyze and generate syntactically well-formed utterances. The phrase structure depicted further below (left column) has been generated by the Stanford Parser.[12] `NP` and `VP` are phrasal categories denoting a *noun phrase* or *verb phrase*, respectively.

```
(S                                                  nsubj(written-2, paper-1)
 (NP (NN paper))                                    pobj(by-3, students-5)
 (VP (VBN written)                                  amod(students-5, clever-4)
   (PP (IN by)                                      prep(written-2, by-3)
     (NP (JJ clever) (NNS students)))))
```

Given such a syntactic analysis and an appropriate set of rules for the English language, we can determine (e.g., by means of OpenNLP or the Stanford Parser) the *head* of a phrase.[13] In this case the head, i.e., the word which determines the category of a phrase and carries the most essential semantic information, is the noun "paper". Note that the widely used heuristic of considering the right-most word as the head a phrase

[11] For the experiments reported in this paper, we initialized this vector by adding all of the entity labels found in the respective ontology.

[12] The phrasal category of the top-most node in the syntax tree should be `NP` (*noun phrase*) rather than `S`, which denotes a sentence.

[13] We omit the distinction between syntactic and semantic heads.

(*righthand head rule*) only works well for morphological units such as compounds (e.g., "student paper").

In addition, the Stanford Parser provides us with a list of syntactic *dependencies* between the individual words of the phrase (right column). For example, it identifies "paper" as the *subject* of "written" and "clever" as an *adjective modifier* of "students".

In the following, we will explain how a linguistic analysis along the dimensions outlined in this section can improve the results of a complex matching approach.

5 Matching Conditions

In the following we show how to use the linguistic analysis combined with a set of simple structural techniques to detect complex correspondences. In particular, we specify four correspondence patterns and define for each of them a set of matching conditions. If each of these conditions is fulfilled we generate a correspondence as instance of this pattern.

Class by Attribute Type (CAT) A correspondence $A_{\#1} \equiv \exists R_{\#2}.B_{\#2}$ of the *CAT* pattern is generated by our algorithm, if the following conditions hold.

1. The label of $B_{\#2}$ is the nominalization of the modifier of the label of $A_{\#1}$.
2. The class $B_{\#2}$ is subclass of the range of $R_{\#2}$.
3. One of the following two conditions holds:
   (a) The class $A_{\#1}$ is a subclass of the domain of $R_{\#2}$ due to the anchor alignment.
   (b) The label of $A_{\#1}$ is a hyponym of the label of the domain of $R_{\#2}$.

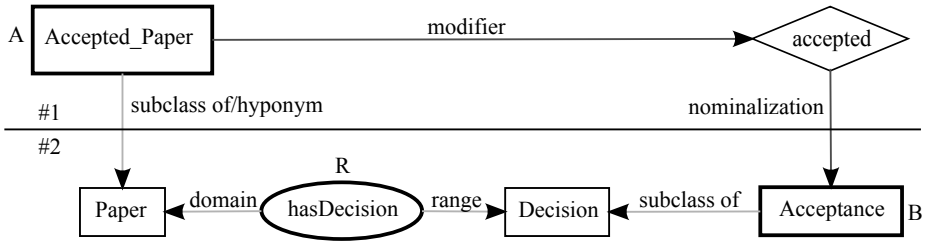

Fig. 2. Conditions relevant for detecting *CAT* correspondence $Accepted\_Paper_{\#1} \equiv \exists hasDecision_{\#2}.Acceptance_{\#2}$.

A typical example is $Accepted\_Paper_{\#1} \equiv \exists hasDecision_{\#2}.Acceptance_{\#2}$. In Figure 2 we depict the three matching conditions relevant for this example: 1) The linguistic analysis reveals that "Acceptance" is the nominalization of the active form of "Accepted", which is in turn the modifier of "AcceptedPaper". A morphological analysis indicates that the first condition is fulfilled. 2) We use a reasoner to check whether $Acceptance_{\#2}$ is a subclass of the range of $hasDecision_{\#2}$, and find that the second condition is fulfilled, too. 3) The third condition is a disjunction. In this concrete case the anchor alignment contains correspondence $Paper_{\#1} \equiv Paper_{\#2}$, which allows us to conclude that the third condition is fulfilled. The third condition is a disjunction,

because on the one hand it might happen that the anchor alignment does not contain the required correspondence, but the linguistic analysis detects the lexical-semantic relation of hyponomy. On the other hand the linguistic analysis might fail, but the information encoded in the anchor alignment might be sufficient. We defined similar disjunctions for some of the other patterns.

In our previous approach we computed e.g., the edit-distance between "Acceptance" and "Accepted" to detect a relation between $AcceptedPaper_{\#1}$ and $Acceptance_{\#2}$. In case it exceeded a certain threshold the counterpart of the first condition was fulfilled. Similarity-based conditions are now replaced by conditions based on linguistic analysis.

Class by Inverse Attribute Type (CAT^{-1}) A correspondence $A_{\#1} \equiv \exists R^{-1}_{\#2}.\top$ of the *CAT*$^{-1}$ type is generated if the following conditions hold.

1. The label of $A_{\#1}$ is the nominalization of the active form of the label of $R_{\#2}$.
2. There exists a class $B_{\#2}$ which is a proper subclass of the range of $R_{\#2}$.
3. One of the following two conditions holds:
   (a) $A_{\#1}$ is, due to the anchor alignment, a subclass of $B_{\#2}$.
   (b) The label of $A_{\#1}$ is a hyponym of the label of $B_{\#2}$.

This pattern and the conditions to detect it are similar to the *CAT* pattern and its conditions. Due to the lack of space we omit a detailed description.

Class by Attribute Value (CAV) Here, we restrain ourselves to detect the boolean variant of the general *CAV* pattern whereby the the attribute values are *true* and *false*. Let in the following $\texttt{adjm}(X)$ refer to the adjective modifier of the phrase X, let $\texttt{advm}(X)$ be the adverbial modifier in X and let $\texttt{vp}(X)$ refer to a verb phrase contained in X. A correspondence $A_{\#1} \equiv \exists R_{\#2}.\{\mathit{false}\}$ is generated by our algorithm, if the following conditions hold.

1. The range of the datatype property $R_{\#2}$ is `Boolean`.
2. One of the following two conditions holds:
   (a) The class $A_{\#1}$ is a subclass of the domain of $R_{\#2}$ due to the anchor alignment.
   (b) The label of $A_{\#1}$ is a hyponym of the label of the domain $R_{\#2}$.
3. $\texttt{advm}(\texttt{label}(R_{\#2}))$ is the antonym of $\texttt{advm}(\texttt{adjm}(\texttt{label}(A_{\#1})))$.
4. The head of $\texttt{label}(R_{\#2})$ is the nominalization of $\texttt{vp}(\texttt{adjm}(\texttt{label}(A_{\#1})))$.

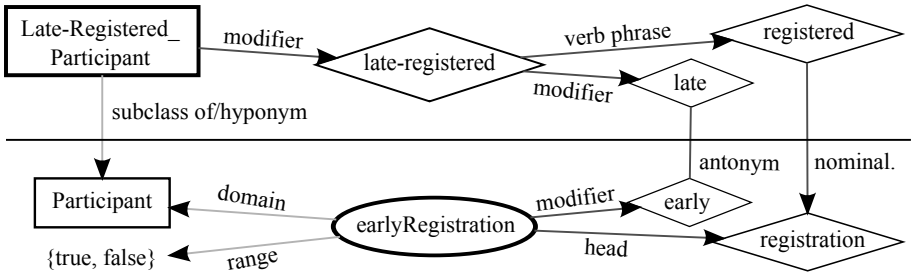

Fig. 3. Conditions relevant for detecting *CAV* correspondence $Late-Registered\_Participant_{\#1} \equiv \exists earlyRegistration_{\#2}.\{\mathit{false}\}$.

Regarding this pattern we use the linguistic analysis to detect antonyms. We expect that modifiers, which are antonyms, will be used to describe a pair of disjoint classes. Complex expressions as $\exists R_{\#2}.\{\mathit{true}\}$ and $\exists R_{\#2}.\{\mathit{false}\}$, given that $R_{\#2}$ is a functional property, refer also – for logical reasons – to a pair of disjoint classes.

Inverse Property (IP) A correspondence $R_{\#1}^{-1} \subseteq P_{\#2}$ of type *IP* is generated, if all following conditions hold.

1. The verb phrase of the label of $R_{\#1}$ is the active voice of the verb phrase of the label of $P_{\#2}$.
2. One of the following two conditions holds:
   (a) The domain of $R_{\#1}$ is a subclass of the range of $P_{\#2}$.
   (b) The label of the domain of $R_{\#1}$ is a hyponym of the label of the range of $P_{\#2}$.
3. One of the following two conditions holds:
   (a) The range of $R_{\#1}$ is a subclass of the domain of $P_{\#2}$.
   (b) The label of the range of $R_{\#1}$ is a hyponym of the label of the domain of $P_{\#2}$.

The IP pattern is the simplest pattern regarding its set of conditions. The first condition is based on the fact that two properties are inverse properties with higher probability, if both contain the same verb phrase in a different voice (active or passive voice). The two other structural conditions ensure that there is a subsumption (possibly equivalence) relation between domain and range of both properties.

It is surprising that it is sometimes harder to detect a simple property equivalence or subsumption than an instance of the *IP* pattern. An example found in our experiments is the following one. In one ontology we have the property $writtenBy_{\#1}$ and its inverse $authorOf_{\#1}$, while in the other ontology we have a property $write\_paper_{\#2}$. Regarding these properties there are two correct correspondences, namely $authorOf_{\#1} \subseteq write\_paper_{\#2}$ and $writtenBy_{\#1}^{-1} \subseteq write\_paper_{\#2}$. While the first one is hard to detect, the second one fulfills all of the conditions listed above and is thus detected by our approach. It is now possible to derive the first correspondence from the second one.

6 Experiments

For our experiments we used the dataset of the OAEI conference track [16]. This dataset consists of several, relatively expressive ontologies that describe the domain of organizing conferences from different perspectives. Regarding this dataset, we made experiments on the full set of ontologies.[14] However, since we want to compare our approach to its predecessor, we also present results restricted to a subset of 9 ontologies that we have used in [11]. To decide whether the structural conditions hold, we used the Pellet reasoner [13]. For checking the linguistic conditions we rely on LiLA. Structural conditions that compare entities from different ontologies require an anchor alignment of non-complex correspondences. We generated this alignment by computing a similarity measure based on the Levensthein distance thresholding the results at a high value.

[14] Since we could not process the ontologies LINKLINGS, COCUS, CONFIOUS with the Pellet reasoner we have excluded them from our experiments.

Dataset	Approach	True Positives					False Positives					Precision
		CAT	CAT^{-1}	*CAV*	*IP*	$\sum$	*CAT*	CAT^{-1}	*CAV*	*IP*	$\sum$	
subset	similarity	4	3	2	-	9	4	7	0	-	11	0.450
subset	linguistic	4	2	2	8	16	1	0	0	0	1	0.941
full set	linguistic	9	4	2	17	32	5	1	0	1	7	0.821

Table 1. Experimental results in terms of true and false positives.

In Table 1, the approach described in this paper is referred to as *linguistic* approach, its predecessor is called *similarity* approach. For each of the four patterns we show the number of true and false positives, as well as the precision of the approach. We can see that the use of linguistic methods helped us to increase precision of the overall approach by a large degree from 45% to 94%, while the number of correctly detected correspondences stays nearly stable. Notice that based on the similarity approach it was not possible to define matching conditions for the *IP* pattern. If we include the IP pattern in our analysis, we come up with the conclusion that we signficantly increased recall.

With the similarity approach we could additionally define conditions for a pattern called *property chain* (not depicted here). We omitted this pattern here, as we thought that the rationales underlying this pattern are hard to justify and that the chosen conditions were partially overfitting to certain aspects of the dataset. Note that the values given for the similarity approach are based on the optimal threshold: raising or lowering the threshold results in a clear loss of recall or precision, while the additional gain is rather limited as shown in [11]. The linguistic methods are different in that there is no threshold whose value would be crucial for the overall performance of the approach.

Our previous approach has been criticized for the requirement of a correct and complete input alignment. However, the new results indicate that such an input alignment is not necessary. It is sufficient to generate an incomplete and partially incorrect alignment in a prior step and to use it as an anchor in the subsequent matching process. Thus, the approach is robust against the noise introduced by such an imperfect anchor alignment.

7 Conclusion

In this paper we have described how to integrate linguistic techniques into a pattern-based approach for detecting complex correspondences. In particular, we have presented correspondence patterns and defined for each of them a set of matching conditions. While in a previous approach [11] many of these conditions were based on computing a simple string-based similarity value, we argued now that it is more appropriate to substitute these conditions by a set of conditions that make use of a linguistic analysis. In our experiments we showed that the new approach yields a significantly higher precision. The tool used to conduct the experiments is open source and available online.[15] Due to its modular structure, matching conditions for new correspondence pattern can easily be specified in a generic XML syntax.

Acknowledgments Johanna Völker is financed by a Margarete-von-Wrangell scholarship of the European Social Fund (ESF) and the Ministry of Science, Research and the Arts Baden-Württemberg. Ondřej Šváb-Zamazal is partly supported by grant no. P202/10/1825 of the Grant Agency of the Czech Republic.

[15] http://code.google.com/p/generatingcomplexalignments/

References

1. Robin Dhamankar, Yoonkyong Lee, Anhai Doan, Alon Halevy, and Pedro Domingos. iMAP: discovering complex semantic matches between database schemas. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.
2. AnHai Doan and Alon Y. Halevy. Semantic-integration research in the database community. *AI Magazine*, 26:83–94, 2005.
3. Jérôme Euzenat. An API for ontology alignment. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, pages 698–712, 2004.
4. Jérôme Euzenat, Alfio Ferrara, Laura Hollink, Antoine Isaac, Cliff Joslyn, Véronique Malaisé, Christian Meilicke, Andriy Nikolov, Juan Pane, Marta Sabou, François Scharffe, Pavel Shvaiko, Vassilis Spiliopoulos, Heiner Stuckenschmidt, Ondřej Šváb-Zamazal, Vojtěch Svátek, Cássia Trojahn dos Santos, George A. Vouros, and Shenghui Wang. Results of the ontology alignment evaluation initiative 2009. In *Proceedings of the 4th International Workshop on Ontology Matching (OM-2009)*, 2009.
5. Jérôme Euzenat, François Scharffe, and Antoine Zimmermann. Expressive alignment language and implementation. Deliverable 2.2.10, Knowledge web, 2007.
6. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2007.
7. Feiyu Lin and Kurt Sandkuhl. A survey of exploiting WordNet in ontology matching. In Max Bramer, editor, *IFIP AI*, volume 276 of *IFIP*, pages 341–350. Springer, September 2008.
8. Giuseppe Pirrò and Domenico Talia. LOM: a linguistic ontology matcher based on information retrieval. *Journal on Information Science*, 34(6):845–860, 2008.
9. Han Qin, Dejing Dou, and Paea LePendu. Discovering Executable Semantic Mappings Between Ontologies. *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 832–849, 2007.
10. Yuzhong Qu, Wei Hu, and Gong Cheng. Constructing virtual documents for ontology matching. In *Proceedings of the 15th International Conference on World Wide Web (WWW)*, pages 23–31, New York, NY, USA, 2006. ACM.
11. Dominique Ritze, Christian Meilicke, Ondřej Šváb-Zamazal, and Heiner Stuckenschmidt. A pattern-based ontology matching approach for detecting complex correspondences. In *Proceedings of the ISWC workshop on ontology matching*, Washington DC, USA, 2009.
12. François Scharffe. *Correspondence Patterns Representation*. PhD thesis, University of Innsbruck, 2009.
13. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: a practical OWL-DL reasoner,. *Journal of Web Semantics*, 5:51–53, 2007.
14. Ondřej Šváb-Zamazal, Vojtěch Svátek, and Luigi Iannone. Pattern-based ontology transformation service exploiting OPPL and OWL-API. In *Knowledge Engineering and Knowledge Management by the Masses. EKAW-2010.*, 2010. Accepted.
15. Willem Robert van Hage, Sophia Katrenko, and Guus Schreiber. A method to combine linguistic ontology-mapping techniques. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference (ISWC)*, volume 3729 of *LNCS*, pages 732–744. Springer, November 2005.
16. Ondřej Šváb, Vojtěch Svátek, Petr Berka, Dušan Rak, and Petr Tomášek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Track of the International Semantic Web Conference (ISWC)*, Galway, Ireland, 2005.
17. Pinar Wennerberg, Manuel Möller, and Sonja Zillner. A linguistic approach to aligning representations of human anatomy and radiology. In *Proceedings of the International Conference on Biomedical Ontologies (ICBO)*, 7 2009.