Automatic Selection of Background Knowledge for Ontology Matching

Christoph Quix, Pratanu Roy, David Kensche
Informatik 5 (Information Systems), RWTH Aachen University, Germany
lastname@dbis.rwth-aachen.de

ABSTRACT

Background knowledge in form of ontologies is an important source of information for many tasks in the semantic web, e.g., ontology matching, ontology construction and editing, natural language processing. In particular, ontology matching and integration can benefit from background ontologies as semantic relationships may be discovered which cannot be identified otherwise. In existing approaches, the background ontology has to be provided often by the user. Therefore, we present an approach that uses background knowledge for matching; but in contrast to other approaches, our approach is able to identify appropriate background ontologies automatically. We implemented this approach in our matching framework *GeRoMeSuite* and tested it with several data sets from the Ontology Alignment Evaluation Initiative (OAEI) campaign. The evaluation shows that the use of background knowledge improves the result in most cases and that our ontology discovery process is able to find appropriate background knowledge to bridge the gap between the two input ontologies.

Categories and Subject Descriptors

H.2.1 [**Database Management**]: Logical Design—*Data models*; H.2.1 [**Database Management**]: Heterogeneous Databases

General Terms

Algorithms

Keywords

Ontology Matching, Background Knowledge, Schema Matching

1. INTRODUCTION

Ontology matching is the process of aligning two ontologies by detecting semantic relationships between them. Various methods and systems [21] to automatically discover these relationships have been proposed. Most of these methods concentrate on features encoded in the ontologies (e.g., lexical and structural methods). While these methods deliver good results for some cases, they are limited to the information contained in the input ontologies. Therefore, background knowledge in form of an additional ontology may be useful to detect semantic relationships [7].

It has been shown in previous work that using an ontology as background knowledge can improve the match result [4], but the selection of the background ontology is obviously an important step. While earlier work either relied on the user to provide such an ontology [4], or used very general upper ontologies (e.g., SUMO, [14]), our approach is able to select the background ontology automatically. The ontology can be selected from a repository or it can be retrieved from the web. A similar approach to ours for selecting ontologies as background knowledge has been proposed in [19], but it has two main drawbacks: (i) background ontologies are selected for each pair of concepts to be matched, which requires significant additional effort if the ontologies to be matched are large; (ii) the ontology selection relies on the ranking of external search engines which is based on popularity rather than on quality of the ontologies. In contrast, in our approach we select background ontologies for the pair of ontologies to be matched, and not for each pair of concepts; thus, the expensive steps of finding ontologies and computing anchor matches have to be done only once. In addition, our method uses an adaptable measure for the similarity between the input ontologies and the background ontologies. The ranking of external search engines is only used for an initial preselection phase, the final selection uses our own measures.

The contribution of this paper is a method for finding background knowledge in the form of ontologies. The method constructs a local repository of ontologies which can be used as background knowledge in ontology matching or other ontology engineering tasks. If the repository does not yet contain suitable ontologies for a given task, the method will search the web for new ontologies, thereby extending the knowledge of the repository. Our method applies an efficient and scalable solution based on information retrieval techniques. Ontologies are translated into text documents by extracting concept names, comments and labels, but also structural features of ontologies are taken into account.

The approach is validated by a systematic evaluation of our system. As the evaluation gives some interesting results about the features of our approach as well as about the applicability of background ontologies in matching, the evaluation results are useful in general for further research in this direction.

The structure of the paper is as follows. The next section presents our approach for the automatic selection of the background ontology. In section 3, we will discuss the integration of the approach into our matching framework *GeRoMeSuite* [12]. Section 4 presents the evaluation results. Related work is discussed in sec-

SWIM 2011, June 12, 2011, Athens, Greece.

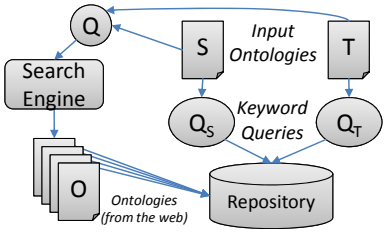

Figure 1: Selection of the background ontology

tion 5, before we conclude our paper and point out future work.

2.SELECTING BACKGROUND KNOWLEDGE

A background ontology should reduce the semantic gap between the source and target ontology. In some cases, the user knows which ontology is suitable for a matching task. However, in general, it is unlikely for a user to know which ontology to use; or, the matching is performed in a fully automatic scenario. Therefore, the matching system must be able to select an ontology to be used as background knowledge automatically.

An overview of our approach is given in Fig. 1. We select ontologies as background knowledge from a local repository using two queries Q_S and Q_T representing the input ontologies S and T. For both queries, we will get a ranked list of ontologies with a similarity score. We select those ontologies as background ontologies which have the highest combined similarity to both input ontologies. The details on the construction of the queries Q_S and Q_T and the similarity measure are given in section 2.1 and 2.2 below.

In case of no appropriate ontology is found in the local repository, we generate a single query to query a search engine to retrieve ontologies from the web. Only a single query is used because the external search engines just return a ranked list of results, without similarity scores. Thus, a combination of similarity scores as in the case for the local repository queries is not possible. The strategies for generating the web query will be discussed in section 2.3.

2.1 Querying the Local Repository

The selection approach should be automatic and efficient. It is not possible to analyze all the stored ontologies in detail. To perform this task, we apply *information retrieval* (IR) techniques. Basically, we ask the question which ontology looks similar to the given ontology. This is done by looking for an ontology in the repository that has a very high keyword similarity both with the source and target ontology. With the *vector space* (VS) information retrieval model [20], it is possible to check whether a document looks similar to another document in an efficient and scalable way. We apply the VS retrieval model for the search in the local ontology repository by using Apache Lucene.

The ontologies to be added to the repository are translated to a corresponding *background document* (*BGdoc*). A *BGdoc* is created by extracting information such as concept names, comments and labels from the ontology and applying some text processing, like stemming and tokenization. While selecting an ontology from the repository as background knowledge, we would like to choose an ontology that is richly structured, as more relationships can be inferred from such an ontology. Hence, we collect several structural features and calculate a boosting factor for each of the ontologies. This value is also stored in the corresponding *BGdoc*. When the document is indexed by Lucene, the lexical portion is used as the content of the Lucene Document object and the calculated boosting factor is used as the Lucene $DocBoost$ value. The current implementation of boosting prioritizes an ontology that has a higher class hierarchy. We also tried other functions to compute the boosting value, but the selected function performed best.

For the given source and target ontology separate *BGdocs* are created. The lexical portion of the *BGdocs* are used to construct two separate queries Q_S and Q_T, which are executed using Lucene to get the similarity assessments wrt. all the stored documents.

2.2 Similarity Measure

For the source and target ontologies, we compute a ranking along with the similarity measures of the ontologies that are stored in the index. For a background ontology O, the similarity measures with the source and target ontologies are $sim(O,S)$ and $sim(O,T)$, respectively. We want to select an ontology that (i) is most similar to S and T (i.e., maximize $sim(O,S) + sim(O,T)$), (ii) is similar to both ontologies, and not only to one (i.e., minimize $|sim(O,S) - sim(O,T)|$), and (iii) there should be a minimal similarity of the ontology to both S and T (i.e., thresholds for $sim(O,S)$ and $sim(O,T)$). Thus, for the set of candidate background ontologies B, we want to find $O \in B$ which maximizes $(\alpha(sim(O,S) + sim(O,T)) - \beta|sim(O,S) - sim(O,T)|)$.

The objective of this equation is to select an ontology that is similar to both the source and target ontologies and in case there is a tie, higher emphasis is given on the individual similarities (i.e., $\alpha > \beta$). Using a very high value for α does not perform well, as this may prefer ontologies that are very similar to one of the input ontologies and highly dissimilar to the other. Hence, we set α to a value only slightly higher than β. Furthermore, the threshold for the required minimal similarity has to be defined. Higher thresholds will only select ontologies that have very high keyword similarity with both the source and target models; but, such an ontology may not exist. On the other hand, with a lower threshold, it is possible to select an ontology that overlaps only in certain parts with the source and target ontologies. As a result, we initially set this threshold to a higher value. This value gets readjusted by a minimization factor to a lower value in subsequent attempts. Between two attempts to select an ontology from the repository, the web is searched for ontologies and new ontologies are added to the repository as explained below. Lowering the threshold will have no impact on a good background ontology that is added to the repository between two attempts as it should have a similarity higher than the threshold. The initial value for the threshold and a lower bound are based on our evaluations and set to 0.4 and 0.1, respectively. During the evaluation, we will show that even if we use an irrelevant ontology, it does not deteriorate the result by much. On the other hand, any background ontology may allow us to find correspondences that cannot be found by directly matching the source and the target.

2.3 Web Query

If no ontology in the repository satisfies the selection criteria, we need to develop a way of adding more ontologies to the repository. We use several search engines to find new ontologies in the web. On the one hand, we use standard web search engines and query specifically for OWL and RDF files (only Google supports searches for these file types). On the other hand, we also use ontology search engines such as Swoogle [8] and Watson [6]. Like [19], we made the experience that the top results returned by these systems are often very general ontologies (e.g., FOAF), which are not helpful as background knowledge in ontology matching.

Both types of search engines need a set of keywords as input. However, the APIs of the systems have a limited query length and increasing the number of keywords in the query reduces the number of results. Thus, we need to select keywords from the source and target ontologies that represent them best. The selection of keywords is based on the *term frequency* (tf) and the *inverse document frequency* (idf) measures from information retrieval [20].

$$(tf \cdot idf)_{w,D} = \left(\frac{tf_{w,D}}{tf_{w,D} + k \cdot |D|}\right) \cdot \left(log \frac{|C|}{df_w}\right)$$

$tf_{w,D}$ is the number of occurences of a term w in a document D, $|D|$ is the size of the document, and k is a squashing parameter. $|C|$ is the size of the document collection and df_w is the number of documents in which the term w occurs. Note that in the left part of the $tf \cdot idf$ formula, we use a variant that squashes repeated occurrences of a term w in the document D, i.e., the first occurrence has a higher weight than the following occurrences of the term [13]. We use 2 as value for the squashing parameter k in our implementation, which favors smaller ontologies.

If the input ontologies contain terms that are not yet in the local repository (e.g., ontologies from a new domain have to be matched) the idf part of the formula above cannot be used to find good keywords and we can rely only on the term frequency in the input ontologies. In that case, we might have to query the search engines several times with different keywords, and might have to download more ontologies than we need for matching. However, this enriches at the same time our local repository and as a consequence, the selection process of keywords in subsequent matching tasks is improved. Please note that the final selection of ontologies will be made by the query to the local repository, as described above.

Our current implementation tries the following four strategies to formulate a query. All of these approaches select keywords from the *BGdoc* representation of the corresponding ontology.

Top-k strategy based on $(tf \cdot idf)$: We select a certain number of keywords which have the highest values for $(tf \cdot idf)_{w,D}$.

High-*idf* strategy: A keyword, which is specific to a document and rare in the document collection, will have a high *idf* value.

Concept name based strategy: Sometimes selecting keywords from the bag of words is not very useful, as these terms are tokenized and stemmed. Querying with only the original terms produces in some cases better results. Hence, we select keywords based on $(tf \cdot idf)$ only if a keyword exists as an original term (concept name or label) in the input ontologies.

Random strategy: This is an optimistic way of generating web queries using random keywords from the input ontologies. Although this strategy might not find relevant ontologies, the repository should be rich with many ontologies and the search for new ontologies should not fail.

We try each of the strategies with different numbers of keywords from the input ontologies. As an example, we start by applying the top-k strategy with $k = 5$, i.e., we choose five words from both source and target ontologies. Moreover, we use a set of filters on the selected keywords, e.g., we do not allow a word and its stemmed root to co-exist in the web query. Finally, we apply the query to search engines and download the found ontologies and add them to our local repository. If we cannot retrieve an appropriate background ontology with the top-5 strategy we apply the top-k strategy with $k = 3$ and $k = 2$. We apply a similar approach for the other strategies as well.

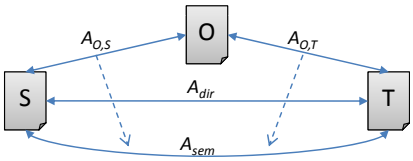

Figure 2: Matching using a background ontology

3. SEMANTIC MATCHING

The presented matching approach has been implemented in our matching framework of *GeRoMeSuite* [12]. *GeRoMeSuite* supports the development of model management operators and is neither limited to ontologies nor to schema matching. The generic approach of *GeRoMeSuite* favors heterogeneous matching tasks (e.g., matching XML Schema and OWL ontologies) [18], but also 'pure' ontology matching is possible [17]. The tool provides several well known matching strategies for lexical and structural matching. Complex matching strategies using several individual matchers can be easily composed in a graphical user interface.

The presented approach using background knowledge is implemented as a new matcher component and can be easily integrated into new or existing match configurations. The matcher can also make use of existing matching strategies already defined in *GeRoMeSuite*, which is the case for anchor matching with the background ontology and for direct matching. The overall matching strategy using background knowledge has five steps (cf. Fig. 2):

1. Compute the direct alignment A_{dir} between S and T: we use a match configuration DM (Direct Match) which is based on the configuration used in OAEI 2010 [17].

2. Use the match configuration DM to compute the *anchor matches*: for each background ontology O, one alignment $A_{O,S}$ between O and S and another alignment $A_{O,T}$ between O and T is computed.

3. Derive semantic matches: for each pair of correspondences from $A_{O,S}$ and $A_{O,T}$, determine whether there exists a relationship (i.e., equivalence, subsumption) between the model elements from O. We use standard reasoning over subsumption and equivalence relationships in $A_{O,S}$, $A_{O,T}$ and O. All the correspondences found in this way are called *semantic matches* and are stored in A_{sem}.

4. Aggregate the results of A_{dir} and A_{sem} for all selected background ontologies by creating the union of the alignments to get the final result set.

5. Consistency check: finally, we apply logical consistency checks defined by validation rules as in ASMOV [11] on the aggregated result to remove inconsistencies.

With the help of automatic selection and semantic matching, the approach returns a richer set of correspondences between source and target ontologies. As a side of effect of the approach, we can provide semantic relationships in our match result, i.e., subsumption relationships as well as equivalence relationships, although the anchor matches and direct match contain only equivalence relationships. Of course, also subsumption relationships could be present in $A_{O,S}$, $A_{O,T}$, and $A_{S,T}$, but it is not required for deriving subsumptions in the final result. In case automatic selection fails to find an ontology, the semantic matching returns the direct match A_{dir} as a result. Also, in case the automatic selection finds more than one ontology as background knowledge, they are used for semantic matching independently and a unified result set is produced.

4. EVALUATION RESULTS

4.1 Metrics and Data Set

Precision and recall are the common measures for the quality of an alignment, but they are not suitable for semantic alignments, which include subsumption relationships [9]. Two alignments may share the same precision and recall value, although one might be very close to the expected result and the other quite distant from it. To overcome this weakness, *semantic precision* and *semantic recall* are proposed [9]. Let A be the computed alignment and R the given reference alignment. The basic idea of the proposed semantic measures is to take into account the relationships which can be derived from the computed and the reference alignments. The α-*consequence* of an alignment A (same for R) contains all the correspondences that can be inferred from the correspondences in A, and is written as $C_n(A)$ (or $C_n(R)$). *Semantic precision* ($P_{sem}(A,R)$) and *semantic recall* ($R_{sem}(A,R)$) are defined as

$$P_{sem}(A,R) = \frac{|A \cap C_n(R)|}{|A|} \qquad R_{sem}(A,R) = \frac{|C_n(A) \cap R|}{|R|}$$

The *semantic f-measure* ($F_{sem}(A,R)$) is defined as the harmonic mean of $P_{sem}(A,R)$ and $R_{sem}(A,R)$.

We used data sets from the benchmark and conference tracks of OAEI. We merged reference alignments from OAEI 2010 (which contain only equivalences), from the oriented track in OAEI 2009 (which contain only subsumptions), and the semantic reference alignments provided on the CSR web site[1] for the conference track. We divided the benchmark data sets into nine groups: the easy tasks 1xx, seven groups in the 2xx series grouped according to their difficulty, and the real world ontologies in 3xx.

4.2 Results

The evaluation addresses the following issues:
1. Number of background ontologies: We analyzed the impact of the quality of the background ontology on the match results. We used relevant and irrelevant ontologies as background knowledge and checked how the match results change if one or several background ontologies are used.
2. Performance of automatic selection: We evaluated how various parameters control the quality of automatic selection and how automatic selection controls the quality of the match result.
3. Overall performance: The overall performance of the matcher is compared to the direct match result (A_{dir}).
4. Runtime performance: Finally, we take a look at the runtime required by the matcher.

As matcher for computing the direct match A_{dir} and the anchor matches $A_{O,S}$ and $A_{O,T}$ we used our matching tool *GeRoMeSuite* [12, 18] with the configuration used for OAEI 2010 [17], which produced a result in the top 5 for the benchmark track. The configuration favors precision over recall, i.e., the result should not contain too many false positives. This is beneficial for a matching approach using background knowledge as we will explain below.

1. How many ontologies should be used as background knowledge? We first compared the results of the semantic matcher using only one or several background ontologies. In the case of the conference track of OAEI, we used 11 ontologies as background knowledge which were not used as source or target ontology of the present matching task (cf. fig. 3). *Aggregated Recall* is the recall which can be achieved if we take the union of the alignments

[1] http://www.icsd.aegean.gr/ai-lab1/csr/

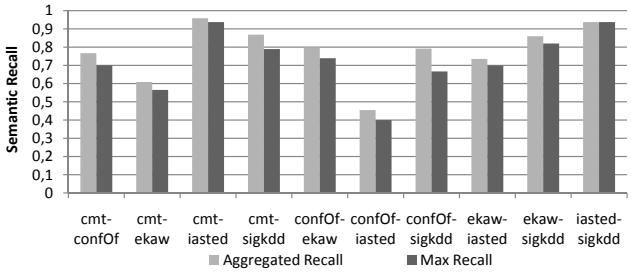

Figure 3: Aggregated vs. Max Recall in Conference Track

across all background ontologies. *Max Recall* is the maximum recall value that can be achieved when using only one ontology as background knowledge. It is easy to see, that the aggregated result is in most cases significantly better than the result from a single ontology. Thus, in general no single ontology can achieve the same result as the aggregated result of several ontologies as background knowledge. These results are in line with the results of [3].

Using additional background ontologies obviously increases recall, but false positives might be added to the match result and harm precision. The question is whether the f-measure for the match result decreases if we use many ontologies (including also irrelevant ontologies). We evaluated this by using ontologies as background knowledge which have *not* been chosen by our automatic selection approach. As we use a 'pessimistic' matcher which aims at a high precision rather than a high recall, the f-measure is not significantly decreased. Only few matches with irrelevant background ontologies will be introduced by the anchor matches; therefore, the overall result has still a high f-measure which is comparable with the f-measure of the direct match result.

To summarize, we have shown that one should use more than one ontology as background knowledge. As a compromise between result quality and runtime, we use atmost three ontologies.

2. What is the performance of automatic selection? The automatic selection procedure should select ontologies, which produce the highest f-measure for the given matching task compared to other ontologies available as background knowledge. This is the upper bound for the automatic selection procedure. The lower bound is the comparison of the match result using the selected background ontology with a randomly chosen background ontology. To simulate a random choice, we computed the average f-measure over all background ontologies stored in the repository.

Fig. 4 shows the semantic f-measure of the match results for the benchmark and conference tracks of OAEI. For the benchmark track, we computed the semantic f-measure using all other benchmark ontologies as background ontology separately and took the average and maximal semantic f-measure. Then, we compared these two values with the computed match result using the automatic selection procedure. As we select at most three ontologies, we aggregate the results for the selected background ontologies (take the union of the alignments). For group 4, 7, 8, 9, the result is significantly higher than the average result and very close to the maximum attainable semantic f-measure, as a 'bridging' ontology is chosen as background ontology, i.e., an ontology that can build a semantic bridge between source and target ontologies. For all other groups, the direct match result is already very good (f-measure $> 90\%$), so the scope of improvement is not too high. For the benchmark track, we can conclude that automatic selection performs very well, but we must admit that the artificial structure of the matching tasks is advantageous for our approach (only in this artificial setting, there are ontologies, which are *exact* semantic

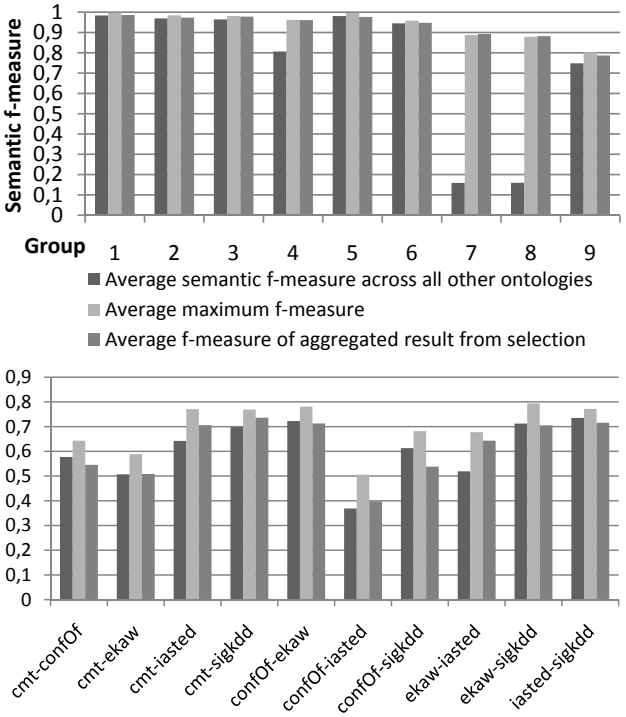

Figure 4: Automatic Selection for Benchmark and Conference

bridges between the source and target ontology).

The results for the conference track (lower part of fig. 4) with real world ontologies underlines the advantages of our approach. For each of the tasks, we computed the semantic f-measure using all other conference ontologies as background and computed the average and maximum of the individual results. In 6 cases, we had a higher result than the average result and in 4 cases, lower values. The problem with each of those 4 cases is that there was no obviously good background ontology for the matching task. As a result, our approach selected multiple ontologies. Among the selected ontologies, one ontology performed bad and incorporated several false positives, which descreased precision. Although the recall increased, the drop of precision resulted in the aggregated f-measure to be lower than the average result.

During the evaluation, we made two observations:

A. Automatic selection does not prefer large ontologies: One of the concerns of this approach is that it might prefer large ontologies because they will share many keywords with the input ontologies. In those cases, it might be possible that a large ontology will be preferred over more relevant small ontologies. Our method automatically retrieved some large ontologies from the web (like SUMO). However, the automatic selection method selected almost always (except for one case) an ontology from the benchmark or conference track as background ontology. This shows that the generated *BGdocs* and the similarity function implemented in Lucene to compare *BGdocs* work very well. Thus, automatic selection strictly tries to select ontologies that are more relevant, and the concern of selecting larger ontologies is not apparent from the test cases.

B. The richness of the repository impacts the selection procedure: It is obvious that with more ontologies in the repository, we have a higher chance to choose a good background ontology. But also the similarity values and thereby the ranking of existing ontologies changes if more ontologies are added to the repository: the value of *idf* changes with adding new ontologies and automatic selection performs more precisely.

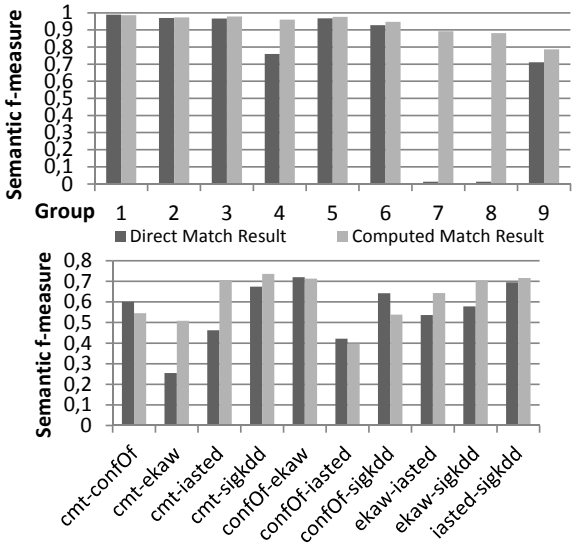

Figure 5: F-Measure in Benchmark and Conference Track

3. What is the overall performance? We compare the overall performance of our approach with a direct matcher that does not use background ontologies (the configuration of *GeRoMeSuite* for OAEI 2010). Fig. 5 shows the results of the direct matcher and the presented approach. As the direct matcher computes only equivalence relationships, we added a simple reasoning procedure over the computed equivalence relationships and the axioms (subsumption and equivalence) in the input ontologies. As it can be seen in the upper part of fig. 5, the direct matcher computes already very good results for the five easy groups in the benchmark track. As there is not much room for improvement, the semantic matcher performs only slightly better in these cases. In the more difficult tasks in groups 4, 7, 8, and 9, the benefit of background ontologies is more visible as the f-measure can be increased significantly.

The lower diagram in fig. 5 shows the results for the conference track. The result is improved in 6 cases; in 4 cases the f-measure is increased by more than 0.1. On the other hand, in 4 cases the f-measure is slightly decreased. As discussed before, this is caused by one background ontology in the aggregated result introducing many false positives. For future work, we will consider these problems in more detail; first ideas to avoid too many false positives include a new aggregation method (e.g., majority vote instead of taking the union) or a stricter method for alignment validation.

Compared to the results of the oriented track in OAEI 2009, we achieve a similar or higher performance than the best matching systems in this track. However, we have to note that the measures used are different: the oriented track in OAEI 2009 just evaluated subsumption relationships, no equivalence relationships, and thus, did not use the *semantic* precision and recall. Therefore, we do not show here diagrams comparing the measures. For the benchmark track, we also achieve on average an f-measure of slightly more than 0.9 (similar to ASMOV [11], the best system in this category). For the conference track, we achieve in many tasks a better performance than the machine-learning based approach of CSR [22].

In general, the approach works well if the ontologies to be matched are semantically different such that a background ontology can be used to detect new relationships. If the source and target ontology are more similar to each other than to any other background ontology, matching using background ontologies should not be applied.

4. What is the runtime performance of the approach? As we

align three pairs of ontologies, we need at least the triple runtime of a direct match approach. The overhead for constructing the background documents, querying Lucene, etc. is not very high (about 2 seconds for tasks in the benchmark track, compared to about 10 seconds for matching one pair of ontologies directly, using a standard PC with 2GHz CPU and 2 GB RAM). The approach scales also to larger ontologies with several thousands elements as the information retrieval techniques, which are at the core of our background knowledge selection approach, are made for large documents and large document collections.

If we have to query the web, the runtime of the system is not anymore under our control as it depends on many factors, e.g., response time of the search engines, download time for the ontologies, etc.

5. RELATED WORK

The closest work of this paper is done in [19] which uses multiple, automatically selected ontologies as this increases the coverage of the background knowledge. To detect a relationship between a pair concepts, they create a query for Swoogle [8] to retrieve multiple background ontologies. The main problem of this approach is that background ontologies are searched for every pair of concepts, which is very inefficient if thousands of pairs have to be matched.

Background knowledge in the form of a domain ontology is used in [4]. The approach is similar to our semantic matching method introduced in section 3 as it also matches the input ontologies with the background ontology and then composes the computed anchor matches to derive new relationships. However, the background ontology is provided by the user. In a following work [3], the benefit of using multiple ontologies as background knowledge is studied; the results are similar to ours.

Upper ontologies (e.g., WordNet, SUMO-OWL) are also applied as background knowledge [14], but there is no significant improvement as domain specific relationships cannot be detected. S-Match is a semantic matcher that uses WordNet senses as a source of background knowledge [10]. In this approach, the concept names are translated to logical formulas between their constituents, which are mapped to the corresponding WordNet senses. As for matching with upper ontologies, the ontologies may contain domain specific knowledge, which are not covered in WordNet.

Information retrieval techniques have been also applied in ontology matching [5, 16]. The basic idea is to convert each concept into a virtual document with a number of fields that encode the structural, lexical and semantic information associated with that concept. Each virtual document is represented as a term vector; and a match between two models is computed using cosine similarity between two term vectors.

Semantic search engines such as Swoogle [8] or Watson [6] support keyword queries to find ontologies, as we use them also in our approach. Searching for similar ontologies of a given ontology is not directly supported. Other semantic search engines (e.g., Sindice [15]) are focused on finding individual objects or triples in linked data rather than complete ontologies. An approach based on query expansion of given keywords to find ontologies is presented in [2], but the system does not include a ranking mechanism. An approach for ranking of ontologies based on structural features is presented in [1], but the ranking is done only wrt. to keywords given by a user and is not based on the similarity to other ontologies.

6. CONCLUSION

Previous work on ontology alignment has introduced ideas to harness background knowledge given in the form of domain knowledge encoded in an ontology. However, the problem of selecting an appropriate ontology has been so far ignored or upper ontologies were used that often are not appropriate for matching detailed domain ontologies. In this work, we have presented a novel approach for automatically selecting background knowledge from a local repository. Our approach does neither favor general upper ontologies nor too specific ontologies that only match one of the two input ontologies. Moreover, the repository of background knowledge ontologies is successively built and expanded while more ontology alignment tasks are handled by the system.

The evaluation has shown that our approach improves the direct match result in many cases significantly. We have shown that using more background ontologies improves the results, and that even selecting a bad ontology as background knowledge usually does not harm the results. For future work, we want to combine this approach for semantic matching with other approaches semantic matching, e.g., based on machine learning.

Acknowledgements: This work is supported by the DFG Research Cluster on Ultra High-Speed Mobile Information and Communication (UMIC, `http://www.umic.rwth-aachen.de`).

7. REFERENCES

[1] H. Alani, C. Brewster, N. Shadbolt. Ranking Ontologies with AKTiveRank. *Proc. ISWC*, pp. 1–15. 2006.

[2] H. Alani, N. F. Noy, N. Shah, N. Shadbolt, M. A. Musen. Searching ontologies based on content: experiments in the biomedical domain. *Proc. K-CAP*, pp. 55–62. 2007.

[3] Z. Aleksovski, W. ten Kate, , F. van Harmelen. Using multiple ontologies as background knowledge in ontology matching. *Proc. CISWeb Workshop*, pp. 35–49. 2008.

[4] Z. Aleksovski, M. Klein, W. ten Kate, F. van Harmelen. Matching Unstructured Vocabularies Using a Background Ontology. *Proc. EKAW*. 2006.

[5] B. Byrne, A. Fokoue, A. Kalyanpur, K. Srinivas, M. Wang. Scalable Matching of Industry Models - a Case Study. *Proc. Workshop on Ontology Matching*. 2009.

[6] M. d'Aquin, E. Motta. Watson, more than a Semantic Web search engine. *Semantic Web Journal*, 2011.

[7] M. d'Aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, D. Guidi. Toward a New Generation of Semantic Web Applications. *IEEE Intelligent Sys.*, **23**(3):20–28, 2008.

[8] L. Ding, R. Pan, T. W. Finin, A. Joshi, Y. Peng, P. Kolari. Finding and Ranking Knowledge on the Semantic Web. *Proc. ISWC*, pp. 156–170. 2005.

[9] J. Euzenat. Semantic precision and recall for ontology alignment evaluation. *Proc. IJCAI*, pp. 348–353. 2007.

[10] F. Giunchiglia, P. Shvaiko, M. Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. *Proc. ESWS*, *LNCS*, vol. 3053, pp. 61–75. Springer, 2004.

[11] Y. R. Jean-Mary, E. P. Shironoshita, M. R. Kabuka. Ontology matching with semantic verification. *Journal of Web Semantics*, **7**(3):235–251, 2009.

[12] D. Kensche, C. Quix, X. Li, Y. Li. *GeRoMeSuite*: A System for Holistic Generic Model Management. *Proc. VLDB*, pp. 1322–1325. 2007.

[13] V. Lavrenko. Lecture Notes of Text Technologies (Boolean and Vector Space Model). University of Edinburgh, 2008.

[14] V. Mascardi, A. Locoro, P. Rosso. Automatic Ontology Matching via Upper Ontologies: A Systematic Evaluation.

IEEE Trans. on Knowl. & Data Eng., **22**:609–623, 2010.

[15] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *Intl. Journal of Metadata, Semantics and Ontologies*, **3**(1):37–52, 2008.

[16] Y. Qu, W. Hu, G. Cheng. Constructing virtual documents for ontology matching. *Proc. WWW Conf.*, pp. 23–31. 2006.

[17] C. Quix, A. Gal, T. Sagi, D. Kensche. An integrated matching system: GeRoMeSuite and SMB - results for OAEI 2010. *Proc. 5th Intl. Workshop on Ontology Matching*. 2010.

[18] C. Quix, D. Kensche, X. Li. Matching of Ontologies with XML Schemas using a Generic Metamodel. *Proc. ODBASE*, pp. 1081–1098. 2007.

[19] M. Sabou, M. d'Aquin, E. Motta. Exploring the Semantic Web as Background Knowledge for Ontology Matching. *Journal on Data Semantics*, **XI**:156–190, 2008.

[20] G. Salton, A. Wong, C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, **18**(11):613–620, 1975.

[21] P. Shvaiko, J. Euzenat. A Survey of Schema-Based Matching Approaches. *Journal on Data Semantics*, **IV**:146–171, 2005.

[22] V. Spiliopoulos, G. A. Vouros, V. Karkaletsis. On the discovery of subsumption relations for the alignment of ontologies. *Web Semantics*, **8**(1):69–88, 2010.