SemRep: A Repository for Semantic Mapping

Patrick Arnold, Erhard Rahm

University of Leipzig
arnold@informatik.uni-leipzig.de
rahm@informatik.uni-leipzig.de

Abstract: In schema and ontology matching, background knowledge such as dictionaries and thesauri can considerably improve the mapping quality. Such knowledge resources are especially valuable to determine the semantic relation type (e.g., equal, is-a or part-of) that holds between related concepts. Previous match tools mostly use WordNet as their primary resource for background knowledge, although WordNet provides only a limited coverage and currentness. We present the design and use of a new comprehensive repository called *SemRep* that combines concepts and semantic relationships from different resources. It integrates both manually developed resources (including WordNet) and semi-automatically extracted relations from Wikipedia. To determine the semantic relationship between two concepts of interest, SemRep also considers indirect relationships of possibly different types. An initial evaluation shows the general effectiveness and efficiency of using SemRep for ontology matching.

1 Introduction

Background knowledge plays an important role for semantic data interoperability, in particular to automatically determine schema and ontology mappings. Many generic match strategies have been developed in the past to determine related concepts between two schemas resp. ontologies, most of which utilize the lexical similarity of element names, the structural similarity of elements or the similarity of associated instance data [BBR11]. However, such strategies often fail in real-world scenarios, e.g., for synonymous concepts with no or low lexicographic similarity (e.g. *car/automobile*) or in the presence of homonyms like *mouse (computer) / mouse (animal)*. A main solution to this issue is the use of background knowledge such as thesauri, making it easy to look-up and use synonymous concept names. Thesauri can also help to determine more complex relationships between concepts of different ontologies, such as is-a and part-of relations.

Unfortunately, there is only a small number of background knowledge resources suitable for matching. There are some elaborated background knowledge ontologies for specific domains, like UMLS for the medical domain, but for general matching scenarios only few resources exist so far. WordNet is among the most popular ones, but is still rather limited in size and currentness. For example, we discovered that WordNet does not contain specific concepts we wanted to match, e.g., furniture concepts such as *teapoy* (a little table with three legs), *cassone* (a historical chest) or *basinette* (synonym for cradle). While these concepts are represented in Wikipedia, they are also missing in Wikipedia-based knowledge resources such as DBpedia or Yago which focus on entities (persons, geographic places, movies etc.), rather than concepts (see discussion in Related Work). As a result we are missing a comprehensive knowledge resource about concepts and their relations, e.g., to help determine mappings between schemas and ontologies.

For this reason, we are building up a comprehensive semantic repository called *SemRep* combining background knowledge from diverse sources. The focus is on collecting concepts and their semantic relations (equal, is-a, part-of and their inverses) in order to support automatic schema and ontology matching, albeit further use cases such as term disambiguation or text mining may also be served.

Table 1 provides details about the four resources that are currently integrated in SemRep, including the number of concepts and relations we gain from those resources. We combine knowledge from three curated resources including WordNet. By far most concepts and relations are derived from an automatic extraction from Wikipedia, based on our methods described in [AR14b]. SemRep is thus many times more comprehensive than WordNet so that it promises a much better coverage for matching tasks. This is also because SemRep can derive the semantic relation type for indirectly related concepts by evaluating many possible relation paths between concepts, e.g., *automobile equal car is-a vehicle*. Evaluating such paths poses several challenges for efficiency and determining the correct relation type, especially for paths with different relation types.

Resource	**Lang.**	**Creation**	**#Concepts**	**#Relations**	**File size**
WordNet	English	Manually	116,326	1,694,505	45 MB
Wikipedia	English	Automatically	1,051,170	2,843,428	149 MB
UMLS	English	Manually	109,599	281,972	19 MB
OpenThesaurus	German	Manually	58,473	914,864	25 MB

Table 1: Resources for the imported relations used in the repository.

We make the following contributions:

- We present the design and implementation of SemRep, an extensible repository combining mapping-related background knowledge from multiple resources of different domains.
- We discuss how relations between two concepts can be directly and indirectly resolved using the repository, and what obstacles may arise.
- We evaluate the usefulness of SemRep for different ontology matching tasks. The evaluation is performed with our semantic ontology matching tool STROMA that determines correspondences with their semantic relation type (equal, is-a, inverse is-a, has-a, part-of or related) [AR14a]. We will show in our evaluation how the use of SemRep can improve the mapping quality compared to the sole use of WordNet.

Our study is organized as follows: Section 2 discusses related work. Section 3 gives an overview on our repository architecture while section 4 discusses the execution of queries and the determination of relation types. We will evaluate our approach in section 5 and conclude in section 6.

2 Related Work

Background knowledge is widely used in schema and ontology matching and helps considerably to improve the quality of the resulting mappings [AKtKvH06]. Background knowledge includes dictionaries, thesauri, and domain-specific taxonomies, e.g., UMLS for the medical domain, as well as existing schemas and mappings (corpus-based matching) [MBDH05, GHKR11]. These resources are especially valuable for determining semantic mappings consisting of correspondences of different relation types (is-a, part-of, etc.). Some tools like COMA and GOMMA also exploit previously generated match results as background knowledge to derive new correspondences, which is referred to as mapping reuse [DR02, GHKR11] For example, the composition of a given mapping between schemas S_1, S_2 with an existing mapping between schemas S_2, S_3 leads to a new mapping between S_1 and S_3 if one assumes transitivity of correspondences.

Background knowledge resources are developed either manually or automatically. In the first category, resources are created and curated either by individual experts or collaboratively by a community of volunteers (e.g., OpenThesaurus). In the linguistic domain, WordNet is certainly the most popular resource for the English language [GM98]. It was manually developed by linguistic experts and thus has a high quality. However, WordNet is relatively limited w.r.t. size and currentness. The latest available version is from 2006 and misses many modern terms like *netbook* or *smart phone*.

Resources with automatically generated background knowledge promise a much better coverage and currentness, but typically face significantly more quality issues than manually controlled resources. We devised an automatic approach to extract concepts and semantic relations between them from Wikipedia [AR14b]. This is achieved by analyzing the first (definitional) sentence of each Wikipedia article and looking for specific semantic patterns like *A is a specific form of B*. The resulting relations have been included in the SemRep repository (Table 1). A similar approach was used in [WLWZ12] to extract is-a relations from more than 1.7 billion websites resulting in a large taxonomy of 2.7 million concepts. So far, such automatically generated background knowledge is not yet exploited for schema and ontology matching. Existing tools mostly use WordNet as their primary resource, in particular for semantic matching (e.g. in S-Match [GAP12] or TaxoMap [RS07]).

Several recent approaches focus on the integration of thesauri from different languages. For example, EuroWordNet combines WordNet with thesauri from eight European languages. UBY-LMF is a relatively new framework to represent linguistic relations and dictionary data for many languages, which may differ greatly w.r.t. flexion, grammatical cases, morphology etc. [EKGH$^+$12]. BabelNet also combines linguistic relations for many languages by aligning Wikipedia pages to WordNet concepts [NP10]. In contrast to these approaches, we do not focus on integrating knowledge for different languages but rather on comprehensively integrating semantic relations between concepts from several sources.

SemRep also differs from knowledge bases providing information about entities (like persons, countries, movies, books, music albums, buildings etc.) rather than concepts. Entities are commonly less related by semantic relations, but primarily by specific relations like *wasBornIn*, *livesIn*, *wasFoundedBy*, *isDirectorOf* etc. Popular resources of this kind include DBpedia [ABK$^+$07], Freebase [BEP$^+$08] and the more domain-specific Geonames. Yago is a special resource, as it combines DBpedia with WordNet, so a knowledge base with a linguistic resource [SKW07]. Still, it provides little advantage over WordNet for supporting schema and ontology matching.

Sabou et al. describe a different way of using background knowledge. Instead of exploiting local resources, they resolve correspondences dynamically by using the ontology search engine Swoogle in order to find relevant background knowledge ontologies for a mapping scenario at hand [SdM06]. They are facing similar problems as we do, like how to determine the relation type in paths larger than 1, or how to deal with contradicting results. Such web-based approaches are generally slower than approaches using local resources, however, the authors did not evaluate the performance or execution times of their approach.

3 Repository Overview

Our goal is to provide with SemRep a comprehensive repository of semantic concept relations in order to support ontology and schema matching. We focus on the general relations listed in Table 2. To achieve a broad coverage we aim at an extensible design making it easy to include the knowledge from several resources. As already mentioned, we have so far covered knowledge from the four resources listed in Table 1 so that SemRep provides semantic relations for concept names from two languages (English and German).

Relation type	**Example**	**Linguistic relation**
equal	river, stream	Synonyms
is-a	car, vehicle	Hyponym – Hypernym
inverse is-a	computer, laptop	Hypernym – Hyponym
has-a	body, leg	Holonym – Meronym
part-of	roof, building	Meronym – Holonym

Table 2: Supported relation types by SemRep.

SemRep must be able to quickly determine for two concepts a and b the semantic relation between these concepts if they are recorded in the repository. This is a simple lookup query if a and b are interrelated by a single relation that was provided by one of the input resources. However, the real usefulness of SemRep comes from the possibility to combine several relations, possibly from different input resources, to indirectly derive semantic relations. Due to the different kinds of relations this is a non-trivial task and we describe

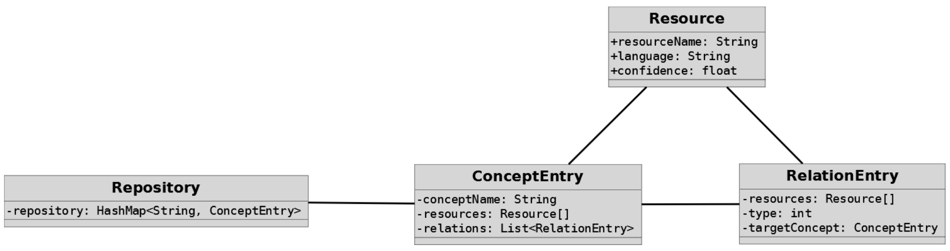


Figure 1: Repository infrastructure in UML notation.

in the next section how we address it. There are also high efficiency requirements, since all queries should be answered very quickly to be usable for matching large ontologies with possibly ten thousands of concepts.

In the rest of this section, we briefly discuss the current implementation approach and how we import semantic relations into the repository.

3.1 Implementation aspects

SemRep represents concepts and their semantic relations in a large directed graph structure where nodes refer to concept names and edges to semantic relations. In an initial implementation we tried to use a database system to maintain this graph, especially a relational DBMS (mySQL) as well as a graph DBMS (Neo4j). Unfortunately, we could not achieve acceptable query execution times in both cases. For example, it took up to 30 seconds with the graph DBMS to determine an indirect correspondence over three steps.

We therefore decided to develop a tailored repository implementation utilizing a Java-based hash map data structure to store the concept nodes. The implementation adopts the simple UML model shown in Fig. 1. Each concept entry has a name, a list of resources where it appeared and a list of relation entries. A relation entry has a specific type, a list of resources where it appeared, as well as a target concept referring to an existing concept in the hash map. As an example, we may have the concept car with two relations to $automobile$ and $vehicle$. Let us assume that the first relation was provided by WordNet, the second by Wikipedia. The concept is thus represented as (car, <WordNet, Wikipedia>, $< r_1, r_2 >$) with the two relations being r_1 = (equal, <WordNet>, automobile) and r_2 = (is-a, <Wikipedia>, vehicle).

Since we can keep the hash map in main memory, this simple approach achieves very fast query execution times despite the large number of concepts and relations.

3.2 Data Import

SemRep makes it easy to include semantic relations from about any source by importing the relations in a simple triple format: (source concept, target concept, relation type). Listing 1 shows a sample extract from an import file; relation types are encoded by a single digit to facilitate the parsing (0 = equal, 1 = is-a, 4 = part-of). To include a new resource to the repository, its semantic relations have to be written into a correctly formatted text file. The file importer can then import these relations and add them to the repository. For is-a and part-of relations, we also add the inverse relations. The simple file format and repository structure make it very fast to launch and populate SemRep. Loading the four resources into the repository takes only about 75 seconds, i.e., we import about 180,000 relations per second. 9 GB of RAM are needed for the graph structure.

Listing 1: Excerpt from an import file

```
car :: vehicle :: 1
car :: automobile :: 0
mountain bike :: bike :: 1
bike ring :: handlebars :: 4
```

There are several options to configure the data import for improved flexibility and extensibility. First it can be specified what should be imported, e.g., only English resources or only WordNet. For each resource, the language and a confidence value has to be specified, e.g., to apply higher confidence values for manually curated resources like WordNet than for automatically generated input such as our Wikipedia relations (see next section). There is also an optional import filter to block relations which may cause problems for query execution. We can therefore filter relations that are involved with general terms such as *object*, *unit*, *entity*, *thing* or *abstraction*. Relations like (furniture is-a object) are meaningless, and since everything may be an object, such relations could lead to false conclusions when querying a semantic relation. We therefore use a blacklist of concepts which we dot not wish to have in our repository and block respective relations. In the future, we plan additional tests to improve data quality, e.g., to check whether imported relations cause inconsistencies. To avoid an increased launch time, we can eliminate such problematic relations from the input file to avoid their import in future launches of the repository.

The number of English-language concepts in the repository after filtering is 1,219,233 and the number of relations is 4,553,688, which means an average number of 3.7 relations per concept. 29 % of all relations are of type equal, 42 % of type is-a or inverse is-a, and 28 % of type part-of or has-a.

4 Querying SemRep

Executing queries to determine the relation type holding between two terms a, b consists of the four steps shown in Fig. 2. The first and last step, indicated by dashed lines, is optional while the second and third step is always carried out. We start with some preprocessing on the terms in case that at least one of the two terms is not contained in the repository. Next, we retrieve a set of paths from a to b within a specified maximal path length. We then determine for each result path a path type and a score (step 3). If no valid path is found, we apply some post-processing, trying to still find some valid path. Finally, the path type of the highest-scored path is returned.

In the following, we describe the introduced steps.

4.1 Preprocessing

Apparently, a relation between concepts a, b cannot be found if the repository is not containing one of the terms as it was not provided by any input resource. This is often the case for very specific concepts which are most frequently compound words that are too obvious to be listed in any resource. For instance, the two words *kitchen* and *chair* can be expected to appear in any English dictionary or thesaurus. The rather simple compound *kitchen chair*, which is obviously a specific chair usually found in the kitchen, is not contained by most dictionaries (including WordNet, Wikipedia, Wiktionary, the Oxford Dictionary and the Free Dictionary), because of its simplicity and because of the huge amount of such compound words that could be created.

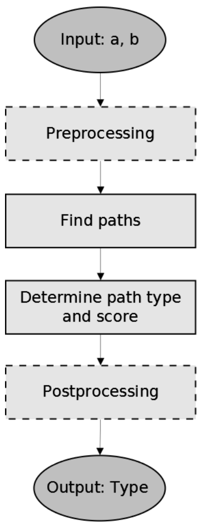


Figure 2: Querying Workflow.

In matching, however, such compound words are of critical importance, and treating compounds correctly can help to overcome the issue that terms are not contained by the repository. We have already shown in [AR13] that a compound consists of a *head* C_H, which is the right-hand word of the compound and specifies its basic meaning, as well as at least one so-called *modifier* C_M appearing left of the head. The compound is normally a specification of the compound head, thus a *kitchen chair* is a specific form of a *chair*. We proposed a technique called *gradual modifier removal* (GMR) to gradually remove the modifiers of the compound until one finally finds the word in the dictionary. If we want to determine the relation type between *kitchen chair* and *seat*, we would remove the modifier *kitchen* from *kitchen chair* and see that *chair* is contained by our repository. The repository suggests the relation *chair is-a seat*, and since a kitchen chair is a specific chair, we can conclude the relation *kitchen chair is-a seat*. With this, we have determined a correct semantic relation type, even though the term *kitchen chair* is not contained in our repository.

Apart from that, we implemented two additional techniques for special compound correspondences. We noticed two important cases:

1. Two compounds can have the same modifier, e.g., *US vice president* and *US politician*.
2. Two compounds can have the same head, e.g., *apple cake* and *fruit cake*.

The first case is relatively easy. If the repository does not contain both input words, we remove the two identical modifiers of both compounds and end up with comparing *vice president* and *politician*, which are both in the repository. The second case is a little more complicated. We can remove the compound head of both concepts and focus our comparison on the two remaining modifiers if, and only if, they are synonyms or in a hypernym relation. For instance, an *apple* is a specific *fruit* and therefore an *apple cake* is a specific *fruit cake*. If the modifiers are unrelated (or in a part-of relation), we cannot come to any conclusion, though. For instance, there is no obvious relation between a *kitchen chair* and a *database chair*, as the two words *kitchen* and *database* are unrelated.

Before we start our query, we thus check if both words appear in the repository. In case that they do not appear, we use the presented techniques of compound preprocessing. With this, we can handle many more terms than the ones that appear in our repository.

4.2 Finding Paths

The memory-based implementation of SemRep allows for a very fast determination of paths between concepts. We use the following procedure to determine all directed paths between two concepts a, b for up to length 4:

1. We first iterate through all direct relations of a. If we find b as one target concept, we found a path of length 1.
2. For paths of length 2, we determine for both a and b all outgoing paths of length 1 and the target concepts, denoted as $T(a), T(b)$. Then we determine all concepts c for which holds $c \in T(a), c \in T(b)$ to return a path of length 2 between a and b.
3. For length 3, we determine for a the target concepts of all outgoing paths of length 2, denoted as $T'(a)$. Similarly as in the step before, we now look for all concepts c for which holds $c \in T'(a), c \in T(b)$. This way, we obtain paths of length 3 between a and b.
4. Finally, to determine paths of length 4 we calculate $T'(b)$ and determine all concepts c for which holds $c \in T'(b), c \in T(a) \cup T'(a)$.

Thus, in the worst case we calculate all paths of length 2 for both concepts which is much faster than calculating all paths of length 4 as a default breadth-first algorithm would do. Each path that was found is checked on its validity, i.e., whether we can determine a reasonable semantic relation type according to the approach discussed in the next subsection.

We support different configurations regarding the path lengths to evaluate. The algorithm can stop as soon as at least one path is found, which we call *First Path*, or it can run all steps in order to determine the most reliable path (we call this *All Paths*). Both configurations can retrieve multiple paths for a single query, but differ in quality and processing time. According to our experiences, *All Paths* allows rarely better results than *First Path*, because longer paths are usually less reliable than shorter ones but cause much longer execution times. We will attend to this issue in more detail in the evaluation section.

4.3 Determining the Path Type

For paths of length 2 or more it is necessary to determine what semantic relation holds between concepts a and b and how reliable this path is. The goal is to return the relation type of the path with the highest confidence, though we could offer several candidates for manual selection by a user.

Let P be a path of length = 2, consisting of three concepts c_1, c_2, c_3 with two relations r_1, r_2 in between. The path type of the combined path is called r. To determine r from r_1 and r_2 we have to consider multiple cases and Fig. 3 shows which resulting path type we choose for SemRep. As one can see and we discuss below, in some cases we derive a new, weaker relation type $related$ and in some cases we cannot derive a semantic relation type. Relation type $related$ is not stored in SemRep but can only be derived as the result of query processing.

c_2-c_3 / c_1-c_2	**equal**	**is-a**	**inv. is-a**	**part-of**	**has-a**
equal	*equal*	*is-a*	*inv. is-a*	*part-of*	*has-a*
is-a	*is-a*	*is-a*	*related*	*part-of*	*has-a*
inv. is-a	*inv. is-a*	–	*inv. is-a*	*part-of*	*has-a*
part-of	*part-of*	*part-of*	*part-of*	*part-of*	*related*
has-a	*has-a*	*has-a*	*has-a*	–	*has-a*

Figure 3: Relation type resulting from complex paths between concepts c_1, c_2, c_3.

In the following discussion we distinguish three main cases for deriving the combined relation type for paths of length 2 where $f^{-1}(r)$ denotes the inverse of a relation type r:

1. $r_1 = r_2$ (homogeneous path)
2. $r_1 \neq r_2, r_1 \neq f^{-1}(r_2)$ (heterogenous, non-inverse path)
3. $r_1 \neq r_2, r_1 = f^{-1}(r_2)$ (heterogenous, inverse path)

Homogeneous paths

Handling homogeneous paths is easy. Since all relation types are assumed to be transitive, it simply holds $r = r_1 = r_2$. We may, however, encounter the problem that c_2 is a homonym which can lead to a wrong relation. For instance, the homonym *table* could lead to the path *desk is-a table is-a data structure*, which would lead to the false relation *desk is-a data structure*. SemRep has not yet a solution to deal with such homonym problems since we only capture concept names. Dealing with homonyms is a general problem in linguistic resources and matching that needs more attention. One possible approach for SemRep could be to maintain several concepts for homonyms and differentiate them by a domain specification or other context information.

Heterogeneous, non-inverse paths

We made a significant observations for heterogeneous paths: The relation types have different binding strengths. Equality has the least binding strength, followed by is-a and followed by part-of, which has the highest binding strength. Thus, the relation type of the highest binding strength in the path determines the overall path type and Fig. 3 shows all possible combinations.

For example, in *engine part-of car is-a motor vehicle* the relation *engine part-of motor vehicle* holds, because part-of dominates against is-a. Quite similarly, in *Computer has-a RAM is-a memory* the relation *Computer has-a memory* holds.

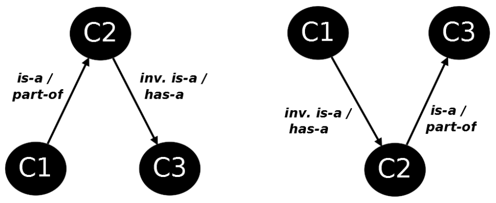


Figure 4: The two different kinds of inverse paths: Paths where a semantic relation can be derived (left) and where it cannot be derived (right).

Heterogeneous, inverse paths

There are 4 inverse combinations, is-a + inv. is-a, part-of + has-a and the opposite cases. These cases need special treatment.

1. $\langle c_1$ is-a c_2 inv. is-a $c_3\rangle$: In this case the relation type *related* is derived. Both c_1 and c_3 have a common father concept c_2, i.e., they share some attributes, but also differ in some attributes. A typical example is *apple – fruit – banana* (see also Fig. 4, left).

2. $\langle c_1$ inv. is-a c_2 is-a $c_3\rangle$ In this case, we have a completely different situation, though it is just the inverse path of case 1 (see also Fig. 4, right). Now, c_2 is a subconcept of c_1 but then the path leads to a different concept c_3. The concept c_2 shares attributes

with both c_1 and c_3, but it is unclear whether c_1 and c_3 are related or share any attributes. Thus, in this case it is impossible to derive a specific relation type. Examples can be *fruit – apple – plant tissue* (is-a), *fruit – apple – tree fruits* (inv. is-a) and *fruit – apple – fructus* (equal), showing that many relation types can actually be possible.

3. $\langle c_1$ part-of c_2 has-a $c_3\rangle$ This case is similar to case 1, but (c_1, c_2) and (c_2, c_3) do not share any attributes here. The only way c_1 and c_3 are related is their co-occurrence in c_2. We can also derive the type *related* here, though it may not always be sensible. Some examples are *CPU – Computer – Memory* (so memory and CPU are related) and *roof – house – cellar* (roof and cellar are related).
4. $\langle c_1$ has-a c_2 part-of $c_3\rangle$ This last case is similar to case 2 and no relation can be unequivocally derived. Examples include *Laptop – CPU – Notebook* (equal) and *Laptop – CPU – Computer* (is-a) and *Laptop – CPU – Netbook* (inv. is-a).

Therefore, we will not return any relation type in cases 2 and 4 and instead return *unrelated*.

Paths of length 3 or more

Paths of length 3 or more can be dealt with by combining the first two relation types according to Fig. 3 and combining the resulting relation type step by step with further relation types. Consider the example *combustion engine is-a engine part-of car inv. is-a convertible*. The first two relations (is-a + part-of) lead to *combustion engine part-of car*. To complete the path, we have to resolve *combustion engine part-of car inv. is-a convertible*. According to Fig. 3 it holds *combustion engine part-of convertible*.

For the result type $related$ it is not meaningful to combine it with further relations; hence we have not considered it as a possible input type in Fig. 3.

4.4 Confidence Calculation

As we may have different paths suggesting different relation types for a specific query, we need a measure to score paths and thus to be able to find the most likely type to hold.

In our scoring function, we consider three parameters:

1. The resource of each relation within the path
2. The relation type of each relation within the path
3. The path length

The rationale behind these parameters is that relations from manually curated resources (like WordNet) are considered as more reliable than automatically generated ones and that

Parameter	**Value**
c(r) = WordNet	0.96
c(r) = Wikipedia	0.90
c(r) = UMLS	0.98
c(r) = OpenThesaurus	0.95
c(t) = equal	0.98
c(t) = is-a / inv. is-a	0.96
c(t) = has-a / part-of	0.8
INV	0.15

Table 3: Values for the types and resource confidences used in formula (1).

some relation types are generally more reliable (is-a) than others (part-of). Finally, the longer a path becomes, the more it is prone to errors.

Let r be a relation within the path, $c(r)$ be the confidence of the resource where this relation comes from and $c(t)$ the confidence of the specific relation type of r ($c(r), c(t) \in [0, 1]$). Our scoring function is:

$$s = c(r_1) * c(t_1) \ * \ c(r_2) * c(t_2) \ * \ ... \ * \ c(r_n) * c(t_n) \ - \ INV \qquad (1)$$

In formula (1), INV is a parameter that is used if we have a inverse path ("related" type).

Currently, we set the configuration shown in Table 3, which proved to lead to the best results in our experiments. The values also reflect the rationale outlined above by giving Wikipedia a slightly lower confidence than the manually curated resources WordNet or UMLS.

4.5 Handling Empty Results

In some cases, the repository cannot find any path from source to target concept. We discovered this in the example *(wine region, location)*, which is actually a simple is-a correspondence. If the concept *wine region* would not be contained by the repository, the preprocessor would immediately reduce it to the word *region* (GMR) and the repository would compare *region* with *location*. There is indeed an is-a relation between those concepts expressed in the repository. However, the preproecessor is only ran if at least one of the two input concepts is not contained by the dictionary, so GMR is not applied in this case as it does not seem necessary.

To further improve the recall, we perform a so-called Post GMR. If no relation is found and one of the two concepts is a compound word, just as in the above example, we apply GMR on this compound word and run the query executor again. With this, we can correctly resolve correspondences like *(wine region, location)*, even though there is no path between the two.

Benchmark	Src. Nodes	Trg. Nodes	# Corr.	equal	is-a	part-of	rel.
Furniture	25	174	136	15	111	10	0
Groceries	58	333	169	32	127	2	8
Clothing	31	153	144	5	130	8	0

Table 4: Overview of the evaluation scenarios and benchmark mappings.

5 Evaluation

We evaluate the use of the introduced SemRep repository for three semantic ontology matching tasks from different domains. The matching tasks are addressed by the tool STROMA that so far used only manually curated knowledge resources such as WordNet. After describing the data sets and experimental setup we present the obtained F-measure results using SemRep, as well as the query execution times. We also provide a comparative evaluation of using STROMA without and with SemRep.

5.1 Experimental Setup

We evaluate the effectiveness of using the SemRep repository for three semantic ontology match tasks called Furniture (F), Groceries (G) and Clothing (C). Benchmarks F and G match parts of the category tree of Amazon and Ebay while benchmark C matches between Amazon and Zalando. We manually generated a perfect mapping for each benchmark together with the semantic relation type per correspondence. Table 4 gives an overview of the three benchmarks, together with the number of correspondences and specific correspondence types they contain.

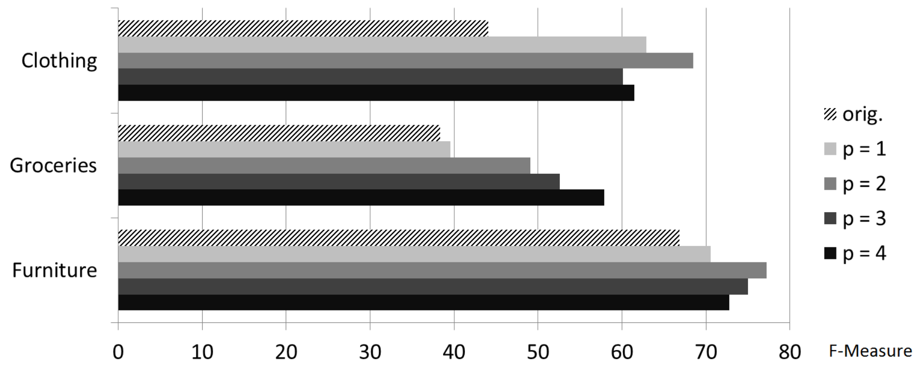
For ontology matching we use our tool STROMA [AR14a] which applies a so-called enrichment strategy. That means it enriches the mapping result obtained by a standard ontology match tool that only determines untyped correspondences in a post-processing step to determine the semantic relation type per correspondence. For this purpose it applies several linguistic and structural strategies and utilizes background knowledge resources such as WordNet or UMLS (for the benchmarks only WordNet is relevant). It also applies a so-called Multiple Linkage strategy, which automatically assigns the is-a relation type to correspondences that refer to a node where many correspondences point to ([AR14a]). We decided to deactivate this function in our evaluation since it infers with the use of the background knowledge resources, especially since the ontologies in the benchmarks differ quite substantially in size.

SemRep is executed on a Windows Server 2008 with 72 GB RAM and an Intel XEON 2.53 GHz processor (30 GB were reserved for the system). We use the configuration *First Path*.

5.2 F-Measure and Execution Times

We first evaluate the F-Measure results for semantic matching with STROMA using SemRep as a background knowledge resource to look up the semantic relation type between pairs of concepts occurring in the input ontologies. Fig. 5 shows the obtained F-measure results for the three match tasks without using SemRep ("'orig."') and using SemRep for different maximal path lengths p (1-4) to identify a semantic relation type between two concepts of interest. While the results are somewhat heterogeneous, we observe that the use of SemRep improves match quality in all cases. For two of the three benchmarks the best F-Measure is achieved for p=2; for the Clothing match task F-Measure improves from 43% to 68%, which is an excellent outcome. Considering path lengths larger than 2 can help to improve recall, but at the expense of higher execution times (see below). This effort payed off only for the test case G (Groceries) which had the poorest F-Measure without using SemRep.

Figure 5: F-Measure for the 3 scenarios for the different path lengths 1 to 4. The column *orig.* shows the original STROMA results without using the repository.



Next we evaluate the use of SemRep w.r.t. to the minimum acceptance threshold the confidence of a determined semantic relation type must meet to be returned. Table 5 shows the results for acceptance thresholds between 0.4 and 0.8 and for maximal path lengths of 2 and 3. It can be seen that for p=3 the results decrease for thresholds higher than 0.6 and that no better results can be obtained than for p=2 (except for the groceries benchmark). Since the minimal score of a path of length 2 is $0.8 * 0.8 = 0.64$, the results remain the same for thresholds below 0.6. The experiments suggest that a threshold of 0.6 is a good default setting as well as a maximal path length of 2.

Table 6 shows the execution times for the different benchmark scenarios. Table 6a shows the execution times for each benchmark in seconds, while Table 6b shows the average execution time for one correspondence. If paths until a maximum length of 2 are used, the execution time is very fast (2.2 to 4.9 ms per query). However, as soon as a maximum path length of 3 is allowed, complex paths have to be calculated and the execution time

T	**p = 2**			**p = 3**		
	F	G	C	F	G	C
0.8	67.6	43.7	66.4	72.1	47.9	58.0
0.7	74.3	47.3	67.1	74.3	48.5	58.7
0.6	77.2	49.1	68.5	75.7	50.9	58.7
0.5	77.2	49.1	68.5	75.0	53.2	59.4
0.4	77.2	49.1	68.5	74.2	53.2	60.1

Table 5: F-measure for path lengths of 2 and 3 and different acceptance thresholds T.

	p=1	**p=2**	**p=3**	**p=4**
F	0.33	0.37	26	39
G	0.37	0.44	19	22
C	0.65	0.7	130	136

(a) Total execution times for each benchmark.

	p=1	**p=2**	**p=3**	**p=4**
F	2.4	2.7	191	287
G	2.2	2.6	112	130
C	4.5	4.9	902	944

(b) Average execution times per correspondence.

Table 6: Execution times for the three benchmarks using different maximal path lengths (1 to 4).

increases considerably. Maximum path lengths of 4 increase execution times further, but only to a little degree. We assume this is because most paths retrieved by the repository are of length 3 or less.

Initially we also experimented with a standard breadth-first search. However it took several times longer that the optimized approach, e.g. already 8 seconds for evaluating paths of maximal length 3.

5.3 Relative Quality Impact of Using SemRep

In the last experiment, we compare the effectiveness of different STROMA configurations without and with using SemRep to determine the impact of the new repository on the match quality. We use two different configurations of STROMA: Configuration 1 (C1) is the default configuration where all implemented strategies are enabled, i.e., both the background knowledge strategy and generic strategies like linguistic or structural techniques. In configuration 2 (C2), all generic strategies are disabled so that the match quality is solely determined by the used repository for background knowledge (WordNet or SemRep). We assume that this configuration can better illustrate the relative usefulness of the repositories, while the first configuration provides their overall impact on mapping quality. For SemRep we apply the default strategy with p=2.

Table 7 shows the F-Measure results and total runtimes for the three benchmarks. Columns C_1 for both the original STROMA with WordNet and the new version with SemRep show that SemRep allows for a substantial overall mapping improvement between 10.7 and 24.4

	Original results			**New results**		
	C_1	C_2	time	C_1	C_2	time
F	66.9	23.5	4.33 s	82.3	44.9	0.37 s
G	38.4	27.2	4.73 s	48.5	39.1	0.44 s
C	44.1	11.8	2.41 s	72.0	44.8	0.7 s

Table 7: F-measure and execution times for the three mappings using STROMA, for the original tool without the repository (left) and for the new tool using the repository (right).

%. The relative quality of the repository becomes even clearer with the results in columns C_2. While in the original STROMA using only WordNet, only about 12 to 27 % F-measure could be reached, the use of SemRep alone allows for F-measure values between about 43 and 53 %. The difference is especially visible for benchmark G, where SemRep alone (42.6 %) achieves better results than the original STROMA running all strategies (38.4 %). The boost in F-measure is mostly caused by the additional knowledge from Wikipedia, which provides relations to many more (and more specific) relations, and the combination of Wikipedia and WordNet relations.

In addition to the quality improvements, execution times were considerably reduced, now being constantly below one second. This improvement is caused by keeping the data in main memory, which was not the case in the previous implementation where we used an API to access WordNet data locally. As a result, using SemRep provides significant improvements both in quality and performance.

6 Outlook and Future Work

We presented SemRep, a new repository infrastructure to integrate different dictionaries and thesauri to support schema and ontology mapping tasks. We successfully imported more than a million concepts and around 4.5 million relations. SemRep allows a fast lookup of semantic concept relations and the combination of several relations, possibly of different types, to improve the coverage and to interconnect knowledge from different sources.

There are still several issues and opportunities for improving SemRep and its applications. First, we could invest more to check and improve the quality of automatically extracted semantic relations. Second, we need to better deal with homonyms in order to limit the derivation of wrong semantic relations. We may also support more user interaction to check the validity of derived semantic relations and propagate back corrections into the repository.

7 Acknowledgments

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

References

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC/ASWC, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[AKtKvH06] Z. Aleksovski, M. Klein, W. ten Kate, and F. van Harmelen. Matching Unstructured Vocabularies using a Background Ontology. In S. Staab and V. Svatek, editors, *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management*, number 4248 in Lecture Notes in Artificial Intelligence, pages 182–197. Springer, 2006.

[AR13] Patrick Arnold and Erhard Rahm. Semantic Enrichment of Ontology Mappings: A Linguistic-Based Approach. In *Advances in Databases and Information Systems*, volume 8133, pages 42–55. Springer, 2013.

[AR14a] Patrick Arnold and Erhard Rahm. Enriching Ontology Mappings with Semantic Relations. *Data and Knowledge Engineering*, 2014.

[AR14b] Patrick Arnold and Erhard Rahm. Extracting Semantic Concept Relations from Wikipedia. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics*, pages 26:1–26:11, New York, NY, USA, 2014. ACM.

[BBR11] Z. Bellahsene., A. Bonifati, and E. Rahm. *Schema Matching and Mapping*. Springer, 2011.

[BEP+08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, New York, NY, USA, 2008. ACM.

[DR02] Hong Hai Do and Erhard Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proc. 28th Intl. Conference on Very Large Databases (VLDB), Hongkong*, 2002.

[EKGH+12] Judith Eckle-Kohler, Iryna Gurevych, Silvana Hartmann, Michael Matuschek, and Christian M. Meyer. UBY-LMF - A Uniform Model for Standardizing Heterogeneous Lexical-Semantic Resources in ISO-LMF. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 275–282, 5 2012.

[GAP12] Fausto Giunchiglia, Aliaksandr Autayeu, and Juan Pane. S-match: An Open Source Framework for Matching Lightweight Ontologies. *Semantic Web*, 3(3):307–317, August 2012.

[GHKR11] A. Gross, M. Hartung, T. Kirsten, and E. Rahm. Mapping Composition for Matching Large Life Science Ontologies. In *2nd Intl. Conference on Biomedical Ontology (ICBO)*, 2011.

[GM98] Christian Fellbaum George Miller. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[MBDH05] Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. Corpus-Based Schema Matching. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 57–68, Washington, DC, USA, 2005. IEEE Computer Society.

[NP10] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: Building a Very Large Multilingual Semantic Network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 216–225, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[RS07] Chantal Reynaud and Brigitte Safar. Exploiting WordNet as Background Knowledge. In *International ISWC Ontology Matching Workshop*, 2007.

[SdM06] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Using the Semantic Web as Background Knowledge for Ontology Mapping. In *Ontology Matching*, 2006.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.

[WLWZ12] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. Probase: A Probabilistic Taxonomy for Text Understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 481–492, New York, NY, USA, 2012. ACM.