Composition Methods for Link Discovery

Michael Hartung, Anika Groß, Erhard Rahm

Department of Computer Science, University of Leipzig
{hartung,gross,rahm}@informatik.uni-leipzig.de

Abstract: The Linked Open Data community publishes an increasing number of data sources on the so-called Data Web and interlinks them to support data integration applications. We investigate how the composition of existing links and mappings can help discovering new links and mappings between LOD sources. Often there will be many alternatives for composition so that the problem arises which paths can provide the best linking results with the least computation effort. We therefore investigate different methods to select and combine the most suitable mapping paths. We also propose an approach for selecting and composing individual links instead of entire mappings. We comparatively evaluate the methods on several real-world linking problems from the LOD cloud. The results show the high value of reusing and composing existing links as well as the high effectiveness of our methods.

1 Introduction

The Linked Open Data (LOD) community publishes an increasing number of data sources from different domains [BHBL09]. These sources are frequently linked with each other to support distributed queries and other forms of data integration. The support of open standards and uniform data and link representation in RDF simplifies the broad use of LOD sources in diverse applications. In addition to general data sources such as DBpedia [BLK$^+$09] there are hundreds of domain-specific sources. For instance, Bio2RDF [BNT$^+$08] provides many life science datasets and ontologies while GeoNames[1] and the New York Times[2] publish data about geographical entities.

There are already numerous RDF links between LOD sources available ($\approx$500 million in Sep. 2012[3]). Still, there is a strong need for increasing the number of links as most sources are linked to only one or a few other sources and new sources need to be linked. The size of the sources makes a manual link discovery infeasible, hence (semi-) automatic match algorithms are needed to determine so-called *mappings* (sets of links) between sources. Many approaches have thus been proposed to directly match the objects of different sources (see Related Work). We aim at complementing these approaches by reusing and composing existing links and mappings to indirectly create new links. Such an approach is especially promising for domains with many existing mappings, e.g., in the life sciences.

[1] GeoNames: http://www.geonames.org/
[2] New York Times - Linked Open Data: http://data.nytimes.com/
[3] http://www4.wiwiss.fu-berlin.de/lodcloud/state/

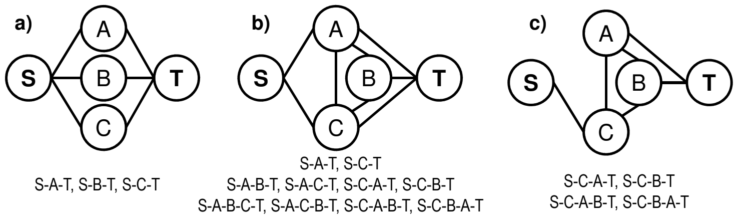


Figure 1: Example scenarios with alternative routes for mapping composition

We already investigated mapping composition for matching biomedical ontologies [GHKR11]. That work focused on scenarios as shown in Fig. 1a where we only compose two mappings (via one intermediate source) per path. By combining several such composed mappings via different intermediates we were able to achieve high quality results with little computation overhead. In [HGKR12] we also started to investigate methods to select the most promising routes for cases when we can compose across several intermediate sources. A main goal of the present paper is to investigate mapping composition for more general mapping topologies and for different domains. Furthermore, we study not only the composition and combination of entire mappings but also the composition of individual links.

As shown in Fig. 1 there are typically many alternative paths to create a mapping between two sources, S and T. For instance, in Fig. 1b the intermediates are connected with each other resulting in ten possible composition routes compared to only three in Fig. 1a (for the same sources). There can be also situations like in Fig. 1c where no route between S and T exists with only one intermediate. Thus, one must consider longer mapping chains consisting of >2 mappings. We therefore need an automatic and general approach to select the most suitable routes that likely result in the best composed mappings.

In this paper, we make the following contributions:

- We study the composition of mappings for link discovery in general, i.e., for arbitrary mapping topologies and paths of arbitrary length.
- We propose different methods to select and combine composed mappings along different paths. (Sec. 3) We further propose a link-based composition approach for selecting and composing individual links instead of entire mappings. (Sec. 4)
- We comparatively evaluate the methods for two domains, namely to interconnect anatomy ontologies and geographical data sources. The results show that we are able to select the most promising routes along sources and entities for efficient mapping composition resulting in high quality mappings. (Sec. 5)

In Sec. 2 we introduce our source and mapping model, discuss the concept of mapping composition and outline the problem that we investigate. We discuss related work in Sec. 6 and summarize in Sec. 7. The Appendix provides the pseudo-code for the algorithms proposed in the paper.

2 Preliminaries

We first describe our source and mapping model. We then discuss mapping composition for two and multiple mappings. Finally, we outline the problem that we address.

2.1 Data Sources, Links and Mappings

A linked data source DS consists of a set of entities. Each entity has an unique URI that is used to reference the object. For instance, the city Leipzig in DBpedia is unambiguously referenced by `http://dbpedia.org/resource/Leipzig`. Entities and their relationships are described by RDF triples of the form (*subject*, *predicate*, *object*) where the third component is either a literal or a reference to an entity of the same or a different source. For example we can use the following triple with a literal to specify the population of Leipzig: (`http://dbpedia.org/resource/Leipzig`, `populationTotal`, 528049). On the other hand, we use an object reference to specify that Leipzig is the largest city of Saxony: (`http://dbpedia.org/resource/Leipzig`, `largestCity_of`, `http://dbpedia.org/resource/Saxony`).

For linking different sources, we mainly use links of type `owl:sameAs` denoting that the linked objects are equal, i.e., represent the same real-world entity. For example, the triple (`http://dbpedia.org/resource/Leipzig`, `owl:sameAs`, `http://data.nytimes.com/N86446625683764674801`) specifies that Leipzig in DBpedia matches to an entity in the New York Times data source. Note that there can be other link types but in this work we will focus on determining sameAs-links since they make up the majority of links between different data sources in the LOD.

A *mapping* between two data sources S and T, $M_{S,T} = \{(o_1, o_2, sim)|o_1 \in S, o_2 \in T, sim \in [0,1]\}$, consists of a set of sameAs-links between these sources, e.g., as determined by some link discovery (match) method. Each link (correspondence) interconnects two related objects o_1 and o_2. Their relatedness is represented by a similarity value sim between 0 and 1 determined by the used match approach. The greater the sim value the more similar are the corresponding objects. We assume a similarity of 1 for manually curated links.

2.2 Mapping Composition

2.2.1 Binary Mapping Composition

In general mapping composition is applied to derive new mappings between two data sources by reusing already existing mappings. Thus, new mappings are generated indirectly via one or more intermediate sources instead of a direct match between the two input sources. The basic situation is the following. We have two data sources (S,T) and two mappings ($M_{S,IS}$,$M_{IS,T}$) w.r.t. an intermediate source IS. Using `domain` and `range` of the mappings we can find out which entities of S, T or IS are covered by the given mappings, e.g., the entities covered by $M_{IS,T}$ in T are in the range of the mapping: `range`$(M_{IS,T})$. Mapping composition is then applied in the following way. A `compose` operator takes as input two mappings (from S and T to IS) and produces new links between objects of S and T if links share the same object in IS:

$$\begin{gathered} M_{S,T} = \texttt{compose}(M_{S,IS}, M_{IS,T}) = M_{S,IS} \circ M_{IS,T} = \\ \{(o_1, o_2, aggSim(sim_1, sim_2)) | o_1 \in S, o_2 \in T, b \in IS : \\ \exists(o_1, b, sim_1) \in M_{S,IS} \land \exists(b, o_2, sim_2) \in M_{IS,T}\} \end{gathered}$$

The similarity values of input links are aggregated (aggSim) into new similarity values, e.g., by computing their maximum, average or by multiplication.

2.2.2 n-ary Mapping Composition

To define the composition of more than two mappings, we first introduce the notion of mapping paths. In particular, a mapping path $P = (M_{S_1,T_1}, M_{S_2,T_2}, \ldots, M_{S_n,T_n})$ of size n w.r.t. a given set of mappings $\mathcal{M}$ is an ordered chain of mappings with the following properties:

1. *Composability*: $\forall M_{S_i,T_i} \in P : M_{S_i,T_i} \in \mathcal{M} \land T_i = S_{i+1}$
2. *Start/End Sources*: the input sources S and T form the start and end of the path, i.e., $S = S_1$ and $T = T_n$
3. *Max. Occurrence*: A mapping $M_{S_i,T_i} \in \mathcal{M}$ occurs at most one time in a path P
4. *Acyclicity*: P has no circles, i.e., there is no sub path $(M_{S_j,T_j}, \ldots, M_{S_k,T_k})$ in P such that $S_j = T_k$

Property 1 ensures that we can traverse (compose) along the path, i.e., the range of a mapping must equal the domain of the succeeding mapping. Furthermore, we can only use mappings available in $\mathcal{M}$. Property 2 guarantees that the start (end) of the path are our sources to be matched, i.e., S or T, respectively. According to property 3 we only allow one occurrence of a mapping within a path. Finally, property 4 restricts the number of possible paths to those with no circles. Together with property 3 we thus exclude paths of infinite length as well as paths visiting intermediate sources multiple times.

To generate a mapping $M_{S,T}$ using a mapping path $P = (M_{S_1,T_1}, M_{S_2,T_2} \ldots, M_{S_n,T_n})$ with $S_1 = S$ and $T_n = T$ we can n-1 times apply the binary `compose` operator ($\circ$) in the following way:

$$M_{S,T} = \texttt{compose}(P) = (\ldots(M_{S_1,T_1} \circ M_{S_2,T_2}) \circ \ldots) \circ M_{S_n,T_n}$$

Starting with the first mapping M_{S_1,T_1} we compose succeeding mappings along the mapping path with the binary operator. The result of one binary compose step is used as input for the next step until we processed the last mapping M_{S_n,T_n} of the path.

2.3 Problem Statement

For two data sources S and T and a given set of mappings $\mathcal{M}$, the problem we investigate is to use composition-based methods to determine a new mapping $M_{S,T}$ consisting of links between entities of S and T. The mappings in $\mathcal{M}$ should contain at least one mapping path between S and T but otherwise there are no restrictions about the number of mappings or the degree of connectedness. The resulting mapping should be of good quality, i.e., all discovered links should be correct (precision) and the number of discovered links should be as high as possible (recall). A composition method should be efficient and scalable to large sources and a large number of mappings.

3 Mapping-based Composition

In the following we propose different methods based on mapping composition to solve the problem we address. We first present an *All* strategy that composes and combines all mapping paths for a given set of mappings $\mathcal{M}$. We then present *Selection* methods that select the most promising mapping paths by considering their effectiveness or complement.

To exemplarily show how the proposed methods and algorithms work, we will use the simple yet comprehensive running example shown in Fig. 2. The sources and mappings are shown on the left side, while a more detailed view on the entities and links is provided on the right side. For simplicity, we assume that all links have an unique similarity of 1.0.

3.1 All Strategy

The idea behind the *All Strategy* is to evaluate all possible mapping paths between the two input sources S and T. For this purpose, we first need to find all possible paths. We can then compose the mappings per path and combine the composed mappings. The first part is related to computing the transitive closure of $\mathcal{M}$. However, we are only interested in all S-T paths and do not consider paths between all available sources.

The determination of all mapping paths $\mathcal{MP}$ between two sources S and T for a given set of mappings $\mathcal{M}$ requires a traversal of mappings starting in S (see Algorithm 1 in the Appendix). We assume that we can traverse a mapping in both directions, e.g., in our example we can traverse from A to B as well as from B to A using $M_{A,B}$. In our running example of Fig. 2(left), we would first select $M_{S,A}$ and $M_{S,B}$ as possible starting paths. In the first round, we consider $(M_{S,A}, M_{A,T})$ as a final path. Furthermore, temporary paths $(M_{S,A}, M_{A,B})$, $(M_{S,B}, M_{B,A})$ as well as $(M_{S,B}, M_{B,C})$ are created. The second round would produce $(M_{S,B}, M_{B,A}, M_{A,T})$ and $(M_{S,B}, M_{B,C}, M_{C,T})$ as final paths, one temporary path namely $(M_{S,A}, M_{A,B}, M_{B,C})$ remains. In the last round, we can use $M_{C,T}$ to build $(M_{S,A}, M_{A,B}, M_{B,C}, M_{C,T})$. Thus, we find four mapping paths between S and T.

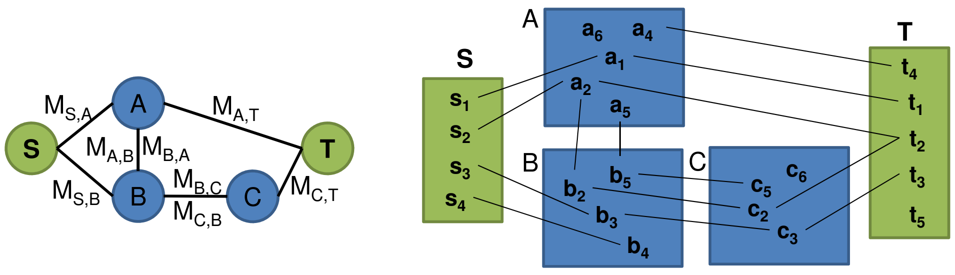


Figure 2: Composition scenario: sources and mappings (left), entity links (right)

Having found all possible paths between the input sources, we can now perform composition as described in Sec. 2.2.2. In particular, we generate $|\mathcal{MP}|$ composed (partial) mappings which we need to merge (unify) to create a final mapping between S and T (see Algorithm 2 in the Appendix). In this paper, we apply a union operator, i.e., the links from all partial mappings are unified. For our example, composing along $(M_{S,A}, M_{A,T})$ results in a mapping consisting of two links: (s_1,t_1) and (s_2,t_2). The mapping path along B and C produces one link: (s_3,t_3). No link is created when considering the $(M_{S,B}, M_{B,A}, M_{A,T})$ path. The longest path via A, B and C creates a link between s_2 and t_2: (s_2,t_2). We now merge all determined links to get the final mapping: $M_{S,T} = \{(s_1, t_1), (s_2, t_2), (s_3, t_3)\}$.

3.2 Selection Strategies

The introduced All Strategy evaluates all possible mapping paths. However, the individual mapping paths are often redundant by leading to the same links. The *Selection Strategy* tries to avoid the repeated calculation of the same links by selecting the most valuable mapping paths and only considers these paths for composition and combination. In the following we first introduce the notion of effectiveness for a mapping path. We will then use this measure as well as others to rate mapping paths w.r.t. their usefulness for composition.

The basic situation for composing two mappings via one intermediate is illustrated in Fig. 3 [HGKR12]. We observe that the compose can at best create new links between entities of S/T that are mapped to the intermediate source IS. The more entities are covered by a mapping to IS the more likely it is that they can be interlinked to entities in the other data source. Thus, intermediates where mappings only cover a small portion of S/T are less effective compared to those covering larger portions. Furthermore, there should be a high overlap of mapped objects in IS, i.e., many IS objects should be in both `range`($M_{S,IS}$) and `domain`($M_{IS,T}$). This is because new links can only be created if there are intermediate objects for the composition. By contrast, a small overlap will only result in a few correspondences, i.e., small and likely incomplete mappings.

Summarizing these observations, we can estimate the effectiveness of two mappings $M_{S,IS}$

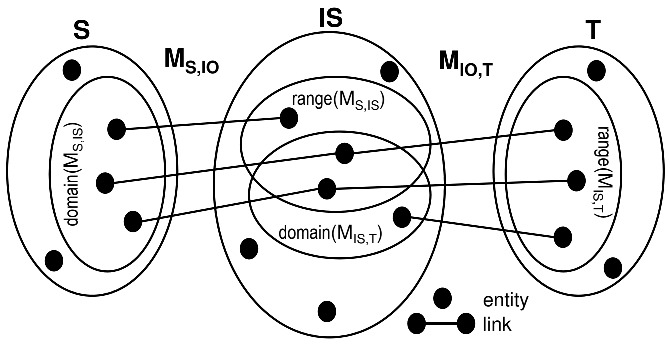


Figure 3: General situation for mapping composition with two mappings and one intermediate

/ $M_{IS,T}$ to be composed as follows:

$$eff(M_{S,IS}, M_{IS,T}) = \frac{2 \cdot |range(M_{S,IS}) \cap domain(M_{IS,T})|}{|S| + |T|}$$

The measure is mainly based on the size of overlapping objects in the intermediate, i.e., the larger the overlap the better the effectiveness. Second, we relate this overlap to the sizes of the sources to be matched S and T. Thus, only mappings with many links can produce a high overlap and a good coverage of objects in S and T. For instance, applying the measure to the scenario displayed in Fig. 3, we would get an effectiveness of $\frac{2 \cdot 2}{5+5}$=0.4.

We can generalize the effectiveness measure for mapping paths of arbitrary length. When performing composition along multiple mappings of a path, it is intuitive that the effectiveness of the path decreases with more mappings. Since each single compose step (see Sec. 2.2.2) has its own effectiveness, the overall effectiveness of a path $P = (M_{S_1,T_1}, \dots,$ $M_{S_n,T_n})$ can be estimated by multiplying the single effectiveness values for all mapping pairs along the path:

$$eff((M_{S_1,T_1}, \dots, M_{S_n,T_n})) = \prod_{i=1}^{n-1} eff(M_{S_i,T_i}, M_{S_{i+1},T_{i+1}})$$

Considering our running example from Fig. 2 we would derive the following effectiveness values for our paths. For $(M_{S,A}, M_{A,T})$ the effectiveness is $\frac{2 \cdot 2}{4+5} \approx 0.44$. The two paths of length three result in an effectiveness of $\frac{2 \cdot 0}{4+5} \cdot \frac{2 \cdot 1}{4+5} = 0$ for $(M_{S,B}, M_{B,A}, M_{A,T})$ and $\frac{2 \cdot 1}{4+4} \cdot \frac{2 \cdot 2}{4+5} \approx 0.11$ for $(M_{S,B}, M_{B,C}, M_{C,T})$, respectively. The longest path $(M_{S,A}, M_{A,B}, M_{B,C},$ $M_{C,T})$ has an effectiveness of $\frac{2 \cdot 1}{4+4} \cdot \frac{2 \cdot 2}{5+4} \cdot \frac{2 \cdot 2}{4+5} \approx 0.05$.

SelectByEffectiveness We can now use the effectiveness measure to select the most valuable (e.g., the best k) paths and compose and combine only these selected paths (*selectByEffectiveness*). For instance, in our example we could only use the two best paths ($(M_{S,A}, M_{A,T})$ and $(M_{S,B}, M_{B,C}, M_{C,T})$) for composition. This would lead to exactly the same mapping $M_{S,T}$ as performing the All Strategy described in Sec. 3.1.

SelectByComplement A second option for path selection is to consider complementing paths. The strategy would first select the most effective mapping path according to our effectiveness measure. After that, we iteratively select those paths with the largest complement compared to the already covered entities in S/T by the previous selected mapping paths. The intuition behind this procedure is to increase the number of covered entities in S/T in the mapping (and thus the recall). For instance, when linking two general data sources about geography, one might consider paths which include complementing knowledge about airports, countries, waters, cities etc.. For our running example and k=2 we would select $(M_{S,A}, M_{A,T})$ (most effective path) and the $(M_{S,B}, M_{B,C}, M_{C,T})$ path, since it offers the best complement (s_3 and s_4 in S, and t_3 in T).

The overall procedure of the *Selection Strategy* (see Algorithm 3 for details) first determines all possible mapping paths. Afterwards we apply the effectiveness measure on each of the possible paths to compute a ranked list of mapping paths. We then can select and compose the most promising (top k) paths either by their effectiveness or complement.

4 Link-based Composition

The introduced strategies so far composed and combined entire mappings. The *Link-based Strategy* aims at a more fine-grained approach by selecting and composing individual links to generate composed links between the two sources to interconnect. For this purpose, we model link discovery as a graph problem and reuse known graph algorithms such as Shortest Path to identify the most promising link paths for composition. In the following we first describe how we create the graph representation from the given sources and mappings. We then show how we select and compose the links to determine the mapping $M_{S,T}$.

We assume a directed, weighted graph $G = (V, E)$ consisting of vertexes V and edges E. Each directed edge $e = (v_1, v_2, weight) \in E$ interlinks two vertexes of V (from v_1 to v_2) including a similarity-based $weight$ which we will later use for path selection within the graph. The transformation from the given mappings in $\mathcal{M}$, the data sources S/T to be linked into such a graph $G = (V, E)$ can be described by some basic rules:

1. Each entity referenced by a link in a mapping of $\mathcal{M}$ becomes a vertex $v \in V$.
2. For each link (o_1, o_2, sim) in a mapping $M_{S',T'} \in \mathcal{M}$ we create edges as follows:
   (a) $if(S' = S)$: $(o_1, o_2, 1 - sim + \epsilon) \in E$
   (b) $if(T' = T)$: $(o_1, o_2, 1 - sim + \epsilon) \in E$
   (c) $otherwise$: $(o_1, o_2, 1 - sim + \epsilon) \in E$ and $(o_2, o_1, 1 - sim + \epsilon) \in E$
3. There exists an unambiguous target vertex $target \in V$.
4. All vertexes v representing entities of T are connected with $target$, i.e., we create edges $(v, target, \epsilon)$ for all $v \in T$.

The idea of the transformation is to model a routing problem, i.e., we like to find the shortest paths from each source vertex (representing an entity in S) to the unambiguous

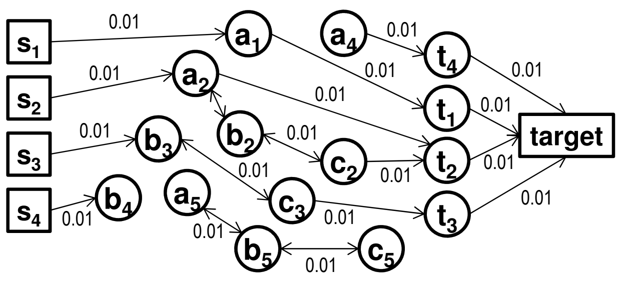


Figure 4: Resulting graph for running example displayed in Fig. 2(right)

$target$ vertex. We thus consider each entity as a vertex and transform links into directed, weighted edges. Vertexes representing source entities have no incoming edges, whereas vertexes of target entities have no outgoing edges (except the ones to the unambiguous $target$ vertex). Edges between entities of intermediate sources can be traversed in both directions (two edges for one link). The greater the similarity of a link, the smaller the weight of an edge, i.e., routing algorithms will likely traverse along edges with small weights. We consider basic costs ϵ for each edge to prefer short paths over longer ones. For our running example of Fig. 2(right) we would create the graph shown in Fig. 4 when using an ϵ of 0.01. We consider $s_1, \dots, s_4$ as starting vertexes with only outgoing edges. The unambiguous $target$ vertex is displayed on the right hand side. All other vertexes involved in at least one link are shown in circles. The vertexes $t_1, \dots, t_4$ of T have only outgoing edges to the unambiguous $target$ vertex. Links between entities of A, B, or C are binary, e.g., one can traverse from $b_2 \in B$ to $c_2 \in C$ and vice versa. Since we assume an unique similarity value of 1.0 for each edge, we have weights of 0.01 for each link, e.g., the link $(a_1, t_1, 1.0)$ is transformed into an edge $(a_1, t_1, 1.0 - 1.0 + 0.01) = (a_1, t_1, 0.01)$.

Using the generated graph we can now exploit the structure to find the most cost-effective routes between entities of S and T. In particular, we will make use of the Shortest-Path (Dijkstra) algorithm [Dij59] to solve the problem (see Algorithm 4 in the Appendix). We iterate over all entities of source S and try to find the shortest path to the $target$ vertex according to the given graph G. For paths found, we create a new link between the current source entity and the last entity before $target$ in the path belonging to T. The similarity is computed according to the formula described in Sec. 2.2.1. The newly created link is added to the mapping $M_{S,T}$. Considering the graph of our running example, we would detect the following paths and thus links between S and T. For entity s_1 there is only one route via a_1 and t_1 to reach $target$. Thus, we would create a link (s_1, t_1) for the mapping. For entity s_2 the shortest path is using the (s_2,a_2,t_2,$target$) route with costs of 0.03. The route via b_2 and c_2 has more costs (0.05) and is not considered. From the third entity s_3 one can traverse along b_3, c_3 and t_3 with minimum costs of 0.03 to the $target$ vertex. For s_4 no path to the $target$ exists (route stops in b_4). Hence, no link for s_4 can be created. In summary, we determine three links, namely (s_1, t_1), (s_2,t_2) and (s_3,t_3) for $M_{S,T}$.

The previously explained procedure returns only the shortest paths in one direction, namely from S to T. This could result in incomplete mappings, e.g., when one entity in a data source links to multiple entities in the opposite source. We therefore evaluate both directions from S to T, and from T to S to find all links between both sources. Traversing in the opposite direction (from T to S) is analogously implemented than the forward traversal already described. When constructing the graph we now insert an unambiguous $source$ vertex where all entities of S are connected with. Furthermore, vertexes representing entities of T have only outgoing edges and we search for the shortest paths from those vertexes to the unambiguous $source$ vertex. The overall procedure of link-based composition is shown in Algorithm 5 in the Appendix.

5 Evaluation

We evaluate our composition methods by analyzing four real-world link discovery problems from two domains. In particular, we produce mappings for the *Geography* instance matching tasks[4] and the *Anatomy*[5] match task of the Ontology Alignment Evaluation Initiative (OAEI). By doing so, we can evaluate the quality of our computed mappings w.r.t. the publicly available OAEI gold standard mappings using precision, recall and F-measure. We first introduce the experimental setup, the used data sources and mappings. We then compare the effectiveness and efficiency of our composition strategies and analyze the impact of the number k of selected mappings and the number of intermediate sources.

5.1 Setup and Overview

For *Geography*, we focus on interlinking NYTimes Data (NYT) with the three LOD sources DBpedia (DBp), FreeBase (FB) and GeoNames (GeoN), i.e., we compare NYT-DBp, NYT-FB and NYT-GeoN. In each case, two of the sources are not matched and can thus be used for composition. We further use mappings to three other intermediate sources from the LOD cloud, namely WorldFactBook (WFB), LinkedGeoData (LGeo) and YAGO. For *Anatomy*, we generate mappings between Adult Mouse Anatomy (MA) and the anatomy part of NCI Thesaurus (NCIT) by composing mappings to four further intermediate sources, namely RadLex, Foundational Model of Anatomy (FMA), Unified Medical Language System (UMLS) and Uberon.

While our composition methods should reuse existing high quality mappings, we did not have them for the considered scenarios. We thus precomputed approximate mappings between any two sources of a domain. These input mappings are generated by a standard metadata-based match technique using our prototype GOMMA [KGHR11]. We compute links between entities based on the similarity of their names and synonyms, i.e., we use links with similarity values between 0 and 1 in the evaluation. We include links of high TriGram similarity and select only the best correspondence(s) per entity. All experiments were performed on an Intel(R) Core (TM) i5-2500 CPU, 4x3.30GHz, 8GB memory ma-

[4]`http://www.instancematching.org/oaei/`
[5]`http://oaei.ontologymatching.org/2012/anatomy/`

(a)

Source Size	**MA**	**NCIT**	**UMLS**	**FMA**	**RadLex**	**Uberon**	
	2738	**3298**	**87913**	**81059**	**30773**	**4958**	
Mapping Size		*1270*	2975	1601	1082	2300	**MA**
			4214	2337	1347	1703	**NCIT**
				63051	17266	5497	**UMLS**
					21781	3504	**FMA**
						2374	**RadLex**
							UberOn

(b)

Source Size	**NYT**	**DBp**	**GeoN**	**FB**	**YAGO**	**LGeo**	**WFB**	
	1920	**1920**	**1780**	**1920**	**1086**	**436**	**254**	
Mapping Size		*1406*	*1781*	*1971*	1130	459	221	**NYT**
			1230	1997	1154	243	232	**DBp**
				1866	1088	472	234	**GeoN**
					1222	480	236	**FB**
						216	202	**YAGO**
							25	**LGeo**
								WFB

Figure 5: Source and mapping sizes for *Anatomy* (a) and *Geography* scenario (b).

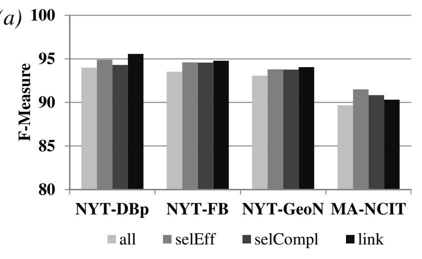
chine with 64-bit Windows 7 Professional OS and a 64-bit JVM.

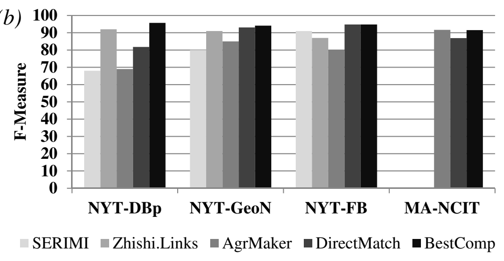
Fig. 5 gives an overview about the size of the used data sources and mappings between them. For *Anatomy* (Fig. 5a), there are two very large intermediate ontologies (UMLS, FMA) with more than 80,000 entities and a mapping between them with more than 63,000 links. Uberon is the smallest of the used intermediate sources. However, it provides links to 2,300 MA entities while the large FMA covers only ≈1,600 links to MA. UMLS provides most links to MA (≈3,000) and NCIT (≈4,200). Note, that we do not use the mapping between MA and NCIT (size printed in italic numbers) for composition.

The sources and mappings for the *Geography* domain are comparatively small (Fig. 5b). NYT, DBp and FB cover more than 1,900 geographical entities while LGeo and WFB comprise less then 500. While DBpedia and some of the other sources contain many more entities the goal of the OAEI Instance Matching task is to find links w.r.t. NYT in the geography area so that the sources were restricted to the relevant subsets. This also leads to small mapping sizes of <500 links between LGeo/WFB and the other sources. For each of the considered geographical data sources most links point to FB (Freebase). Again, we do not use the shown direct mappings for composition in case this is the mapping to be evaluated (italic numbers). For instance, when computing the NYT-DBp mapping we include direct mappings between all sources except the one between NYT and DBp.

5.2 Comparison of Composition Methods

We consider the All Strategy (*all*), the two selection strategies SelectByEffectiveness (*selEff*) and SelectByComplement (*selCompl*) as well as the Link-based Strategy (*link*) for evaluation. The results achieved for each method and match task are displayed in Fig. 6a. For each match task we used the maximum number of available mappings. This results in 325 possible paths for each of the three Geography tasks and 64 paths for Anatomy. For all tasks we are able to achieve F-measure values over 90%. However, there are some slight differences between the methods. The *all* strategy performs worst for all tasks, apparently because the large number of mapping paths lead to a relatively low precision (incorrect links). By contrast, *link* achieves the best quality in all geography tasks. The two selection methods and especially *selEff* also perform well. *selCompl* is slightly less

(a)


(b)


(c)

	NYT-DBp				NYT-FB				NYT-GeoN				MA-NCIT			
Strategy	all	selEff	selCompl	link	all	selEff	selCompl	link	all	selEff	selCompl	link	all	selEff	selCompl	link
Runtime in s	81	<1	<1	6	73	<1	<1	5	78	<1	<1	5	296	6	5	20

Figure 6: Comparison of composition strategies – *(a)* Composition results for all four tasks *(b)*, Comparison with direct match approaches, *(c)* Execution times (in s)

effective since it may select paths with a good complement but lower effectiveness when the complementing entities cannot be linked.

Regarding runtime efficiency the differences between the methods are even greater (see Fig. 6c). As expected, *all* requires the most time (with up to 5 minutes for Anatomy) since it composes all possible mapping paths. The selection methods are the fastest with $<$1s for each Geography task and about 5s for Anatomy. *link* requires some more time than the selection strategies due to the time needed for constructing the graphs and running the Shortest Path algorithm. In summary, the results show that using the selection or link strategy one can achieve high quality results with very short execution times.

We further compare the effectiveness of our composition approaches (BestComp) with those of the systems that participated in the OAEI 2011 campaign (SERMI [AHSdV11], Zhishi.Links [NRZW11], AgreementMaker [CSC$^+$11]) and with our own match strategy (DirectMatch) described in the setup. The results in Fig. 6b show that composition of existing mappings can improve the match quality compared to traditional match approaches. In particular, for all Geography tasks we achieve the best quality in terms of F-measure. Interestingly, the results of DirectMatch are topped by our composition methods which use mappings produced with DirectMatch. This shows that mapping composition can harvest additional knowledge in intermediate sources to discover more and better links. For Anatomy, AgreementMaker achieves the best quality (SERMI, Zhishi.Links did not participate in this track). They also exploit background knowledge from other anatomy ontologies and combine the results with those from a direct match of the sources.

5.3 Sensitivity Analysis

When applying the selection strategies, one needs to set the value of k to specify how many of the possibly numerous paths should be considered. The diagram presented in Fig. 7a

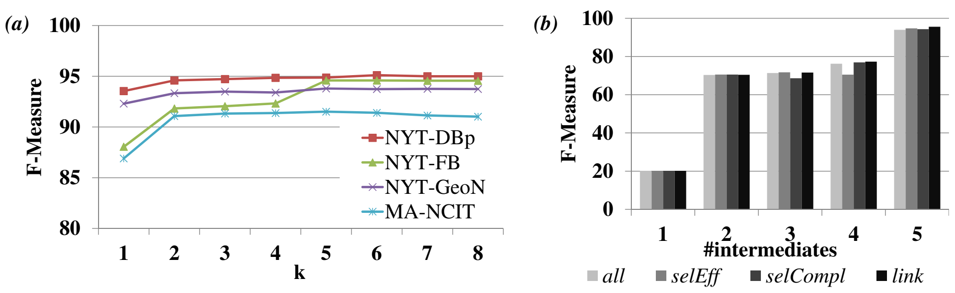


Figure 7: Sensitivity Analysis – *(a)* Influence of k for *selEff* strategy, *(b)* Results for increasing number of intermediates for NYT-DBp match task

shows how the number of selected paths for *selEff* influences the final match quality. We increased the number from 1 to 8 and noticed a similar behavior in all scenarios. A single mapping path generally leads to insufficient match quality but the combination of two or three paths achieves already high F-Measure values. The match quality can not be improved further for more than 6 mapping paths or may even decrease at some point (e.g., for Anatomy). These results show, that one can already achieve a good match quality when selecting only a few but effective paths.

In a further sensitivity experiment we test how the methods perform for a varying number of intermediates. In particular, we increase the number of possible intermediates (and thus mappings) and measured the quality. In each step we considered all available mappings among the used intermediates. The results for the NYT-DBp task are shown in Fig. 7b. We observe that an increasing number of intermediates leads to a better match quality, since more mapping paths can be exploited. When using one or two intermediates the methods do not differ due to the small number of possible paths, namely only 1 (4) paths for one (two) intermediates. *link* achieves the best quality for 5 intermediates with 325 paths. This shows that the link-based strategy is especially valuable for composition scenarios where a large number of possible paths need to be explored. When only a few paths exist, one can apply *all* or a selection strategy instead.

6 Related Work

Many approaches have already been published for link discovery and the related problems of entity resolution and ontology matching. General frameworks for link discovery include SILK [VBGK09], LIMES [NNA11] and Zhishi.links [NRZW11]. These approaches support different similarity measures to directly compute links between LOD data sources. Some of them incorporate methods to scale with large data sources, e.g., LIMES exploits the mathematical characteristics of metric spaces to speed up the match process, or SILK performs a blocking step to reduce the number of comparisons. Many more approaches have been proposed for entity resolution (see [KR10, EIV07] for surveys) as well as ontology matching (see [ES07, RB01] for surveys). Usually the approaches directly compare the input sources by employing different lexical or structural methods in workflows.

The principle of composition has mainly been studied for schemas [DL04, Rah11] and in model management [FKPT05, BM07]. Only a few approaches consider this technique for ontology matching or link discovery. For instance, [ZB05] utilizes the FMA ontology to derive mappings between MA and NCIT. Furthermore, the SAMBO system [LT06] or AgreementMaker [CSC$^+$11] utilize background knowledge like the UMLS or Uberon to find additional links in their match process. [TGO$^+$10] presents an empirical study of mapping composition with mappings from BioPortal. In own previous works, we investigated one-hop mapping composition for ontologies in the life sciences [GHKR11, HGKR12] and found out that the usage of multiple intermediates can help to increase the overall match quality.

In contrast to previous works, we study mapping composition for link discovery in general and differ in the following points. First, we match indirectly by reusing existing mappings and by applying composition along different mapping paths of different length. Second, the proposed methods can cope with various mapping composition scenarios, i.e., we can perform composition for a fully connected network of sources as well as for sparsely interconnected sources. Third, we evaluate the effectiveness and usefulness of paths to select and process only the most promising one for a fast and effective link discovery.

7 Summary and Future Work

We proposed general composition methods to solve the link discovery problem of the Data Web. The introduced mapping- and link-based methods can be applied in different link discovery scenarios with sparsely or heavily interconnected data sources. The evaluation on real-world link discovery problems showed that focusing on the most effective mapping paths / links is a good strategy to produce mappings of high quality in very short execution times. For scenarios with only few mapping paths one can apply a selection strategy or the all strategy to create new mappings. For more complex networks with a large number of possible paths the link-based strategy is most promising.

In future work we aim at investigating more complex kinds of mapping composition by also taking into account relationships within intermediate sources. We further plan to study other graph algorithms such as Ford & Fulkerson for selecting links for composition.

References

[AHSdV11] S. Araújo, J. Hidders, D. Schwabe, and A.P. de Vries. SERIMI - resource description similarity, RDF instance matching and interlinking. In *OM*, 2011.

[BHBL09] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, 5(3), 2009.

[BLK$^+$09] C. Bizer, J. Lehmann, G. Kobilarov, et al. DBpedia - A Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3), 2009.

[BM07] P.A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *Proc. of SIGMOD*, 2007.

[BNT$^+$08] F. Belleau, M.A. Nolin, N. Tourigny, et al. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5), 2008.

[CSC$^+$11] I.F. Cruz, C. Stroe, F. Caimi, et al. Using AgreementMaker to Align Ontologies for OAEI 2011. In *OM*, 2011.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 1959.

[DL04] E.C. Dragut and R. Lawrence. Composing Mappings Between Schemas Using a Reference Ontology. In *Proc. of CoopIS/DOA/ODBASE*, 2004.

[EIV07] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1), 2007.

[ES07] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag New York, 2007.

[FKPT05] R. Fagin, P.G. Kolaitis, L. Popa, and W.C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *Transactions on Database Systems*, 30(4), 2005.

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

[HGKR12] M. Hartung, A. Gross, T. Kirsten, and E. Rahm. Effective Mapping Composition for Biomedical Ontologies. In *Proc. of SIMI Workshop @ ESWC*, 2012.

[KGHR11] T. Kirsten, A. Gross, M. Hartung, and E. Rahm. GOMMA: A Component-based Infrastructure for managing and analyzing Life Science Ontologies and their Evolution. *Journal of Biomedical Semantics*, 2:6, 2011.

[KR10] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2), 2010.

[LT06] P. Lambrix and H. Tan. Sambo–A system for aligning and merging biomedical ontologies. *Web Semantics: Science, Services and Agents on the Web*, 4(3), 2006.

[NNA11] A.C. Ngonga Ngomo and S. Auer. LIMES: a time-efficient approach for large-scale link discovery on the web of data. In *Proc. Intl. Conf. on Artificial Intelligence*, 2011.

[NRZW11] X. Niu, S. Rong, Y. Zhang, and H. Wang. Zhishi.links results for OAEI 2011. In *OM*, 2011.

[Rah11] E. Rahm. Towards large-scale schema and ontology matching. *Schema matching and mapping*, 2011.

[RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4), 2001.

[TGO$^+$10] A. Tordai, A. Ghazvinian, J. Ossenbruggen, et al. Lost in translation? Empirical analysis of mapping compositions for large ontologies. In *OM*, 2010.

[VBGK09] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk–a link discovery framework for the web of data. In *Proc. of the 2nd Linked Data on the Web Workshop*, 2009.

[ZB05] S. Zhang and O. Bodenreider. Alignment of multiple ontologies of anatomy: Deriving indirect mappings from direct mappings to a reference. In *AMIA*, 2005.

A Algorithms

In the following, we show the pseudo-code of the algorithms used by the different strategies proposed in the paper.

Algorithm 1 (`findAllMappingPaths`) is used to determine all possible mapping paths between two sources S and T based on given mappings in a mapping set $\mathcal{M}$.

Algorithm 1: findAllMappingPaths

```
Input: source $S$, target $T$, set of all mappings $\mathcal{M}$
Output: all mapping paths $\mathcal{MP}$ between $S$ and $T$
$\mathcal{MP} \leftarrow \emptyset$;
$\mathcal{P} \leftarrow$ getAllMappingsWithDomain($\mathcal{M}, S$);
while $\mathcal{P} \neq \emptyset$ do
    $\mathcal{P}' \leftarrow \emptyset$;
    foreach $P \in \mathcal{P}$ do
        $lastDataSource \leftarrow$ getLastDataSource($P$);
        $\mathcal{CM} \leftarrow$ getAllMappingsWithDomain($\mathcal{M}, lastDataSource$);
        foreach $M_{S',T'} \in \mathcal{CM}$ do
            if $\neg$contains($P, T'$) then
                $P$.append($M_{S',T'}$);
                if $T = T'$ then
                    $\mathcal{MP} \leftarrow \mathcal{MP} \cup \{P\}$;
                else
                    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{P\}$;
    $\mathcal{P} \leftarrow \mathcal{P}'$
return $\mathcal{MP}$;
```

With the help of Algorithm 2 (`composeAndMergeMappingPaths`) we perform composition along the paths in $\mathcal{MP}$ to create the mapping $M_{S,T}$ between S and T.

Algorithm 2: composeAndMergeMappingPaths

```
Input: source $S$, target $T$, mapping paths $\mathcal{MP}$
Output: mapping $M_{S,T}$ between $S$ and $T$
$allMappings \leftarrow \emptyset$;
foreach $P \in \mathcal{MP}$ do
    $M_{S,Tmp} \leftarrow P$.getNextMapping();
    while $P$.hasNextMapping() do
        $M_{S',T'} \leftarrow P$.getNextMapping();
        $M_{S,Tmp} \leftarrow$ compose($M_{S,Tmp}, M_{S',T'}$);
    $allMappings \leftarrow allMappings \cup \{M_{S,Tmp}\}$;
$M_{S,T} \leftarrow$ union($allMappings$);
return $M_{S,T}$;
```

Algorithm 3 (composeSelectionStrategy) shows the overall procedure for selection-based mapping composition either by considering path effectiveness or complement.

Algorithm 3: composeSelectionStrategy

```
   Input: source $S$, target $T$, mappings $\mathcal{M}$
   Output: mapping $M_{S,T}$ between $S$ and $T$
1  $\mathcal{MP}_{all} \leftarrow$ findAllMappingPaths($S$,$T$,$\mathcal{M}$);
2  $\mathcal{MP}_{ranked} \leftarrow$ computeEffectiveness($\mathcal{MP}_{all}$);
3  $\mathcal{MP}_{topK} \leftarrow$ selectByEffectiveness($\mathcal{MP}_{ranked}$) or
   SelectByComplement($\mathcal{MP}_{ranked}$);
4  return composeAndMergeMappingPaths($S$,$T$,$\mathcal{MP}_{topK}$);
```

Using shortestPathCompose (Algorithm 4) we create a mapping $M_{S,T}$ by determining the shortest paths between entities of S and T in graph G.

Algorithm 4: shortestPathCompose

```
   Input: source $S$, target $T$, graph $G = (V, E)$
   Output: mapping $M_{S,T}$ between $S$ and $T$
1  $M_{S,T} \leftarrow \emptyset$;
2  foreach $s \in S$ do
3      $shortestPath \leftarrow$ getShortestPath($s, target, G$);
4      if $\neg shortestPath.$isEmpty() then
5          $link \leftarrow$ compose($shortestPath$);
6          $M_{S,T} \leftarrow M_{S,T} \cup \{link\}$;
7  return $M_{S,T}$;
```

Algorithm 5 (linkBasedCompose) shows the overall procedure for the link-based composition approach. In particular, we create a forward and a backward graph on which we perform the shortest path algorithm (shortestPathCompose). We finally unify the results to create the mapping $M_{S,T}$ between in the input sources.

Algorithm 5: linkBasedCompose

```
   Input: source $S$, target $T$, mappings $\mathcal{M}$
   Output: mapping $M_{S,T}$ between $S$ and $T$
1  $G_{forward} \leftarrow$ buildComposeGraph($S, T, \mathcal{M}$);
2  $M_{S,T} \leftarrow$ shortestPathCompose($S, T, G_{forward}$);
3  $G_{backward} \leftarrow$ buildComposeGraph($T, S, \mathcal{M}$);
4  $M_{T,S} \leftarrow$ shortestPathCompose($T, S, G_{backward}$);
5  $M_{S,T} \leftarrow M_{S,T} \cup$ inverse($M_{T,S}$);
6  return $M_{S,T}$;
```