Overcoming Schema Heterogeneity between Linked Semantic Repositories to Improve Coreference Resolution

Andriy Nikolov, Victoria Uren, Enrico Motta, and Anne de Roeck

Knowledge Media Institute, The Open University, Milton Keynes, UK
{a.nikolov,v.s.uren,e.motta,a.deroeck}@open.ac.uk

Abstract. Schema heterogeneity issues often represent an obstacle for discovering coreference links between individuals in semantic data repositories. In this paper we present an approach, which performs ontology schema matching in order to improve instance coreference resolution performance. A novel feature of the approach is its use of existing instance-level coreference links defined in third-party repositories as background knowledge for schema matching techniques. In our tests of this approach we obtained encouraging results, in particular, a substantial increase in recall in comparison with existing sets of coreference links.

1 Introduction

With the emergence of the Linking Open Data initiative[1] the amount of semantic data available on the Web is constantly growing. New datasets are being published and connected to existing ones according to the Linked Data principles. Coreference links between entities (RDF individuals) in different datasets constitute a major added value: these links allow combining data about the same individuals stored in different locations. Due to large volumes of data, which have to be processed, these links cannot be produced manually, so automatic techniques are usually employed. However, discovering these links and querying data distributed over different repositories is often problematic due to the semantic heterogeneity problem: datasets often use different ontological schemas to describe data from the same domain.

In particular, ontological heterogeneity represents an obstacle for automatic coreference resolution tools, which discover links between individuals: it is not clear which classes represent overlapping sets of individuals that should be compared and which of their properties are relevant for similarity computation. For example, if a newly published dataset contains information about computer scientists, the tool needs to know which other datasets potentially contain coreferring individuals and to which classes they belong. Although the use of common schema ontologies such as FOAF, SKOS or Dublin Core is encouraged [1], existing datasets often employ their own schemas or, in other cases, terms of common ontologies do not provide full information about the data they describe

[1] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

A. Gómez-Pérez, Y. Yu, and Y. Ding (Eds.): ASWC 2009, LNCS 5926, pp. 332–346, 2009.

and many important usage patterns remain implicit: e.g., in DBLP[2] a generic *foaf:Person* class in fact refers only to people related to computer science. Because of these issues, coreference resolution algorithms must be specially tuned for each pair of datasets to be linked. As a result, when a new repository is integrated into the the Linked Data cloud it is often hard for the publisher to connect it to all relevant datasets, which contain co-referring individuals, and many coreference links can remain undiscovered.

Thus, schema-level matching and instance-level coreference resolution constitute two stages of the process needed to discover coreference links between individuals stored in different repositories. Our data fusion tool KnoFuss [2] was originally developed to perform the second stage of the task. In this paper we extend its workflow and focus on resolving the first part of the problem (schema-level) in order to improve the performance at the second stage (instance-level).

Our approach focuses on inferring schema-level mappings between ontologies employed in Linked Data repositories. It implements the following novel features, which we consider our contribution:

- Use of instance-level coreference links to and from individuals defined in third-party repositories as background knowledge for schema-level ontology matching.
- Generating schema-level mappings suited for the needs of the instance coreference resolution process. In particular, our algorithm produces fuzzy mappings representing degree of overlap between classes of different ontologies rather than strict equivalence or subsumption relations.

The primary effect of the approach for the instance-level coreference resolution stage is the increase in recall because of newly discovered coreference links, which were initially missed because relevant subsets of individuals in two datasets were not directly compared. We obtained promising results in our test scenarios: coreference resolution recall increased (by 15% in the worst case and by 75% in the best case) in comparison with existing publicly available sets of links without substantial loss of precision.

The rest of the paper is organized as follows: in the section 2 we briefly discuss the most relevant existing approaches. Section 3 gives a general idea of our approach, provides illustrative example scenarios and outlines the overall workflow of the system. Sections 4 and 5 focus on the schema-level and data-level stages of the approach respectively. In the section 6 we present the results of our experiments performed with test datasets. Finally, section 7 summarizes our contribution and outlines directions for future work.

2 Related Work

This paper is related to both schema-level and data-level aspects of data integration. Both these aspects were originally studied in the database research community where many of the solutions, which were later extended and adapted in the Semantic Web domain, were originally proposed.

[2] http://www4.wiwiss.fu-berlin.de/dblp/

Considering the schema matching problem, two classes of techniques emerged: schema-level and instance-level approaches [3], which respectively rely on evidence defined in database schemas and in the data itself. With the emergence of ontological languages for the Semantic Web, which had different expressive capabilities from relational database schemas, specific solutions for ontology matching were developed [4]. The features of the Linked Data environment are the presence of large volumes of data and the availability of many interconnected information sources, which can be used as background knowledge. Therefore, two types of approaches are particularly relevant for this environment.

First, instance-level techniques for schema matching can be utilized. The advantage of instance-level methods is their ability to provide valuable insights into the contents and meaning of schema entities from the way they are used. This makes them suitable for the Linked Data environment for two reasons: (i) the need to capture the actual use pattern of an ontological term rather than how it was intended to be used by an ontology designer and (ii) the availability of large volumes of evidence data. In several approaches instances of classes and relations are considered as features when computing similarity between schema-level entities: e.g., CIDER [5] and RiMOM [6]. One particularly interesting approach, which uses schema alignment and coreference resolution in combination, was introduced in the ILIADS system [7]. ILIADS focuses on the traditional ontology matching scenario, where two schemas have to be integrated, and performs schema-level and instance-level matching in a loop, where newly obtained instance-level mappings are used to improve schema-level alignment and vice versa. This is similar to our approach where schema-level matching is performed to enhance the instance coreferencing process. However, unlike ILIADS, our approach was primarily motivated by the instance-level integration scenario and exploits information from many sources rather than only two.

The second relevant category of schema-matching techniques are those which utilize external sources as background knowledge. An approach proposed in [8] performs matching of two ontologies by linking them to an external third one and then using semantic relations defined in the external ontology to infer mappings between entities of two original ontologies. The SCARLET tool [9] employs a set of external ontologies, which it searches and selects using the Watson ontology search server[3]. These approaches, however, only consider schema-level evidence from third-party sources, while our approach relies on instance-level information.

Existing ontology matching tools usually produce as their output strict mappings such as equivalence and subsumption. In the Linked Data environment such mappings in many cases are impossible to derive: sometimes even strong semantic similarity between concepts does not imply strict equivalence. For instance, the concept *dbpedia:Actor* denotes professional actors (both cinema and stage), while the concept *movie:actor* in LinkedMDB refers to any person who played a role in a movie, including participants in documentaries, but excluding stage actors. Since our goal is to discover pairs of classes, which are likely to contain equivalent individuals, in our approach we produce mappings of a more

[3] http://watson.kmi.open.ac.uk/WatsonWUI/

generic type, which would reflect the fact that there is a significant degree of overlap between classes.

The problem of coreference resolution between instances (also called record linkage [10]) also has been studied for a long time in the database domain. In the classical model proposed in [10] by Fellegi and Sunter the decision regarding whether two individuals (records) describe the same entity was taken based on calculating an aggregated similarity between their field values. This model served as the base for the majority of algorithms developed in the domain (see [11] for a survey). Other proposed approaches exploit additional kinds of available information. For example, the algorithms described in [12] and [13] analyse links between instances and consider graphs instead of atomic data individuals, in [14] ontological schema restrictions are used as evidence and in [15] provenance information about data sources is exploited.

While originally in the Semantic Web domain the research primarily focused on the schema-level ontology matching, with the emergence of the Linked Data the problem of instance-level integration also gained importance. Popular existing tools were created and applied to discovery of the *owl:sameAs* coreference links for Linked Data. These follow the classical Fellegi-Sunter model: e.g., SILK [16] and ODDLinker (the latter used to create links from the LinkedMDB dataset [17]). However, given the decentralized character of data and the need to deal with large numbers of connections between entities in different repositories, the problem of managing and maintaining these connections received special attention in the Semantic Web community along with the problem of creating them (e.g., OKKAM [18] and CRS [19]). A particularly interesting approach is idMesh [20], where maintenance of links constitutes a complementary stage of the link discovery process: the system combines coreference links into graphs and considers their impact on each other to reason about their correctness and reliability of their sources. While these solutions focused on instance-level links, the UMBEL ontology[4] was developed to connect schemas used by Linked Data repositories. It provides an important "common denominator" for different ontologies, but, in our view, it does not always capture the actual usage patterns of ontological terms, but provides instead its own view on data, which often differs from other ontologies (e.g., the class *umbel:Dragon* contains such instances as *dbpedia:Azure_Dragon* describing the creature from the Chinese mythology and *dbpedia:Bernard_Guasch* referring to a CEO of the French rugby team "Catalans Dragons"). The VoiD ontology[5] provides meta-level descriptions of repositories including their main topic. However, these descriptions do not go into detail about the sets of individuals, which a dataset contains.

These approaches, however, focus either on the schema or the data level and do not consider them in combination, in particular, how mappings between entities at both levels influence each other. Our approach can be seen as complementary to these methods as it takes into account which schema-level mappings are needed

[4] http://www.umbel.org/
[5] http://rdfs.org/ns/void/html

to discover coreference links between instances and which schema-level relations are implied by available instance-level correspondences.

3 Overview: Example Scenarios and the Overall Workflow

As was said before, the focus of our approach was to produce schema-level mappings to assist instance-level coreference resolution. These mappings are primarily needed to identify, which subsets of two data repositories are likely to contain co-referring individuals, so that such subsets can be processed by a link discovery algorithm afterwards. Let us consider an example scenario, where deriving such mappings is problematic and hampers coreference resolution (Fig. 1). Both DBPedia and DBLP datasets contain individuals representing computer scientists. In many cases the same person is described in both repositories, but under different URIs. However, only a small proportion of possible coreference links between them is available[6]. More links can be discovered by performing automatic coreference resolution, but this task is complicated by two issues:

- Datasets do not contain overlapping properties for their individuals apart from personal names.
- Individuals which belong to overlapping subsets are not distinguished from others: in DBLP all paper authors belong to the *foaf:Person* class, while in DBPedia the majority of computer scientists is assigned to a generic class *dbpedia:Person* and not distinguished from other people. As a result, it becomes complicated to extract the subset of computer scientists from DBPedia which can be also represented in DBLP.

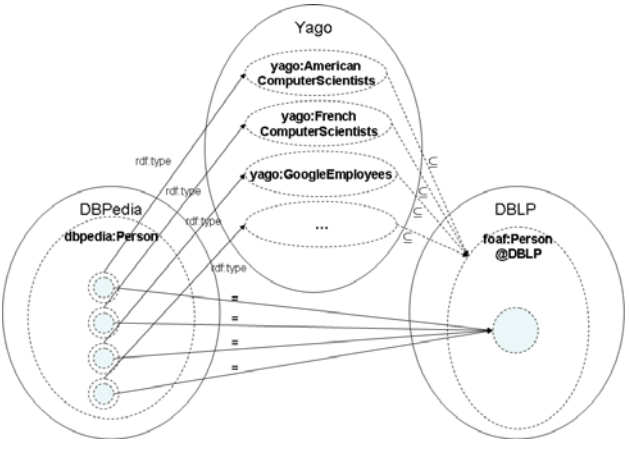

Fig. 1. DBPedia and DBLP: exploiting schema-level links with third-party datasets. Solid arrows show existing *owl:sameAs* (=) and *rdf:type* links. Dashed arrows represent discovered schema relations. The system identifies the subset of *dbpedia:Person* instances, which overlaps with DBLP *foaf:Person* instances, as a union of classes defined in YAGO.

[6] 196 links in total in DBPedia 3.2 on 13/06/2009

Applying name comparison for all *foaf:Person* and *dbpedia:Person* individuals is likely to produce many false positive results because of ambiguity of personal names. Before performing instance matching we need to narrow the context and exclude from comparison individuals which are unlikely to appear in both datasets. Since the actual schema ontologies used by repositories, which have to be connected, are not sufficiently detailed, then evidence data defined in other data sources should be utilized.

For the reasons outlined in section 2, instance-based ontology matching techniques are particularly suitable to infer schema-level mappings in the Linked Data environment. These techniques operate on ontologies, which share the same sets of individuals. Sometimes this scenario is present in Linked Data repositories directly: for instance, in the example shown in Fig. 1, DBPedia individuals are structured by the DBPedia own ontology, but also have *rdf:type* links to the classes defined in the YAGO[7] and Umbel ontologies. However, more often such sets can be constructed by clustering together individuals connected via existing *owl:sameAs* coreference links. Such sets are likely to be incomplete because intermediate datasets may not contain all individuals represented in their neighbour repositories or because some links on the path are not discovered, but they can still be used to derive relations between classes.

A crucial difference between the Linked Data environment and the traditional ontology matching scenario, which focuses on matching two ontologies, is the possibility of using individuals and concepts defined in other repositories and links between them as background knowledge. In our approach we exploit two types of background knowledge:

- Schema-level evidence from third-party repositories.
- Data-level evidence from third-party repositories.

In the following subsections 3.1 and 3.2 we will describe these types of evidence and briefly outline how they are used to produce schema-level mappings. Then, in the subsection 3.3 we will briefly describe the overall workflow of our KnoFuss system, which employs these mappings to discover new coreference links between individuals.

3.1 Schema-Level Evidence

In the example shown in Fig. 1 the problem is caused by insufficiently detailed classification of individuals provided by the repositories' ontologies. In this situation additional schema-level information has to be introduced from external sources.

Individuals in DBPedia are connected by *rdf:type* links to classes defined in the YAGO repository. The YAGO ontology is based on Wikipedia categories and provides a more detailed hierarchy of classes than the DBPedia ontology. Our algorithm uses this external ontology to identify the subset of DBPedia which overlaps with the DBLP repository. The procedure involves the following steps:

[7] http://www.mpi-inf.mpg.de/yago-naga/yago/

1. Construct clusters of identical individuals from DBPedia and DBLP using existing *owl:sameAs* mappings. In this scenario each cluster corresponds to one *owl:sameAs* link and contains two individuals: one from DBLP and one from DBPedia.
2. Connect these clusters to classes in the YAGO and DBLP ontologies respectively. In the later case only the class *foaf:Person* is involved. For example, the cluster containing the individual *dbpedia:Andrew_Herbert* is connected to several YAGO classes (e.g., *yago:MicrosoftEmployees*, *yago: BritishComputerScientists* and *yago:LivingPeople*) and to *foaf:Person*.
3. Infer mappings between YAGO classes and the *foaf:Person* class used in DBLP using instance-based matching (see section 4). A set of overlapping YAGO classes is produced as a result: e.g., mappings between *foaf:Person* and *yago:MicrosoftEmployees* and between *foaf:Person* and *yago: BritishComputerScientists*.
4. Run instance-level coreference resolution for individuals belonging to mapped classes to discover more coreference resolution links. For example, at this stage we discover the link between the individual *dbpedia:Charles_P._Thacker* belonging to the class *yago:MicrosoftEmployees* and its DBLP counterpart, which did not exist in the original link set.

3.2 Data-Level Evidence

Data-level evidence includes individuals defined in third-party repositories and coreference links to and from them. The scenario shown in Fig. 2 illustrates the use of this type of evidence. The LinkedMDB repository[8] contains data about movies structured using a special Movie ontology. Many of its individuals are also mentioned in DBPedia under different URIs. Some of these coreferent individuals, in particular, those belonging to classes *movie:film* and *movie:actor*, are explicitly linked to their counterparts in DBPedia by automatically produced *owl:sameAs* relations. However, for individuals of some classes, direct links are not available. For instance, there are no direct links between individuals of the class *movie:music_contributor* representing composers, whose music was used in movies, and corresponding DBPedia resources. Then, there are relations of the type *movie:relatedBook* from movies to related books in RDF Book Mashup but not to books mentioned in DBPedia. Partially, such mappings can be obtained by computing a transitive closure for individuals connected by coreference links. However, many links are missed in this way because of the omission of an intermediate link in a chain (e.g., 32% of *movie:music_contributor* instances were not connected to corresponding DBPedia instances). Again, such links can be discovered by comparing corresponding subsets of LinkedMDB and DBPedia directly. To discover these subsets our approach computes a transitive closure over existing mappings and combines co-referring individuals into clusters. These clusters are used as evidence for the schema matching procedure to derive schema-level mappings: in our example, we derive the correspondence between

[8] http://data.linkedmdb.org/

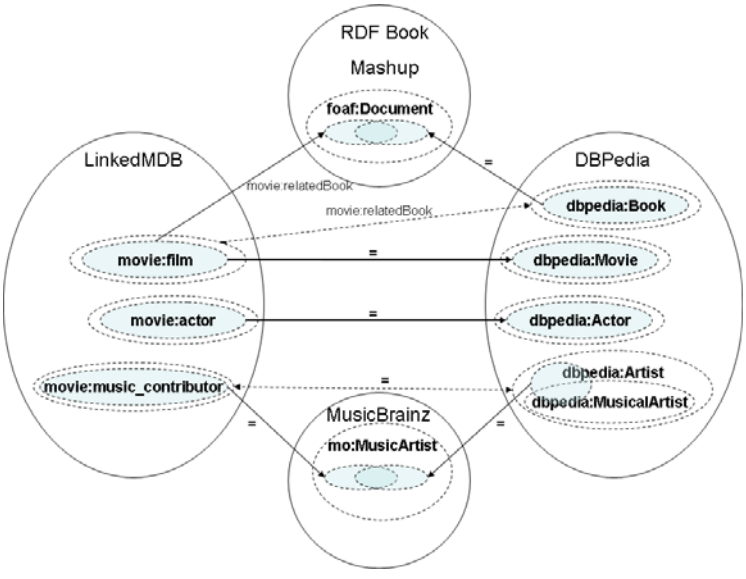

Fig. 2. LinkedMDB and DBPedia: exploiting instance-level coreference links with third-party datasets. Solid arrows show existing *owl:sameAs* (=) and *movie:relatedBook* links. Dashed arrows connect sets containing potentially omitted links.

movie:music_contributor and *dbpedia:Artist* and the *rdfs:range* relation between the property *movie:relatedBook* and the class *dbpedia:Book*. These mappings are used afterwards to perform coreference resolution over related subsets.

3.3 KnoFuss Architecture and the Fusion Workflow

Performing integration at the data level constitutes the main focus of our data fusion tool called KnoFuss [2], originally implemented to merge datasets (knowledge bases) structured according to the same ontology. The KnoFuss architecture implements a modular framework for semantic data fusion. The fusion process is divided into subtasks as shown in the Fig. 3 and the original architecture focuses on its second stage: *knowledge base integration*. The first subtask is *coreference resolution*: finding potentially coreferent instances based on their attributes. The next stage, *knowledge base updating*, refines coreferencing results taking into account ontological constraints, data conflicts and links between individuals (algorithms of this stage were not employed in the tests described in this paper). The tool uses SPARQL queries both to select subsets of data for processing and to select the most appropriate processing techniques depending on the type of data. In order to make the tool applicable to datasets, which use different ontologies, it was extended with the *ontology integration* stage. This stage consists of two subtasks: *ontology matching*, which produces mappings between schema-level concepts, and *instance transformation*, which uses these mappings to translate

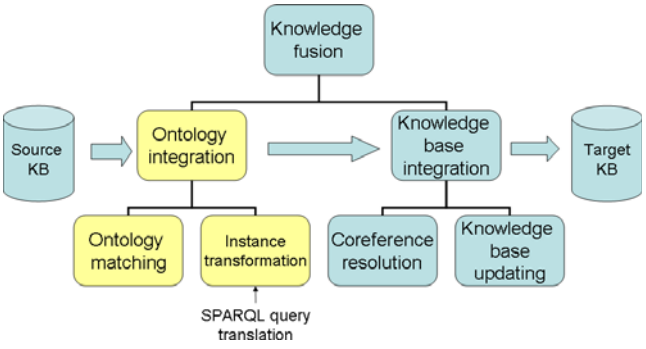

Fig. 3. Fusion task decomposition incorporating schema matching

SPARQL queries into different ontologies, so that the following stages can operate in the same way as in the single-ontology scenario. In the workflow described in this paper the *ontology integration* stage is performed in a "bottom-up" way exploiting links defined at the data level while the *knowledge base integration* stage uses them in a "top-down" way. In sections 4 and 5 we will describe these two main stages of the workflow in more detail.

4 Inferring Schema Mappings: The "Bottom-Up" Stage

The process of inferring schema mappings starts by composing clusters of individuals from different repositories. At this stage pairs of connected individuals belonging to different datasets are retrieved. Then the system forms clusters of coreferent individuals by computing transitive closures over available links.

These clusters represent the basic evidence, which we use to infer schema-level mappings. For each individual in a cluster we extract its class assertions. We consider that a cluster belongs to a certain class if at least one individual from this cluster belongs to a class. At this stage classes which are used in different datasets are always treated as different classes, even if they have the same URI. For instance, in our example, the Movie ontology used in LinkedMDB and the Music ontology used in Musicbrainz both extend the standard FOAF ontology. But we treat the class foaf:Person in both these ontologies as two distinct classes: *foaf:Person@Movie* and *foaf:Person@Music*. This is done in order to discover the actual usage pattern for each class, which may implicitly extend its ontological definition, as was pointed out before.

At the next step we construct mappings between classes. As we said before, instead of equivalence and subsumption the algorithm produces a special type of relation, which we called *#overlapsWith*. Formally this relation is similar to the *umbel:isAligned* property[9] and states that two classes share a subset of

[9] http://www.umbel.org/technical_documentation.html

their individuals. However, in our case, a quantitative assessment of the relation is necessary to distinguish between strongly correlated classes (like *dbpedia:Actor* and *movie:actor*) and merely non-disjoint ones (like *movie:actor* and *dbpedia:FootballPlayer*, which share several instances such as "Vinnie Jones"). This relation has a quantitative measure varying between 0 (meaning the same as *owl:disjointWith*) and 1 (meaning that there is a *rdfs:subClassOf* relation in one direction or both). We calculate similarities between classes based on the sets of clusters assigned to them. Two criteria are used to produce the output set of *#overlapsWith* relations between classes:

1. The value of the overlap coefficient compared to a threshold.

$$sim(A,B) = overlap(c(A), c(B)) = \frac{|c(A) \cap c(B)|}{min(c(A), c(B))} \geq t_{overlap},$$

   where c(A) and c(B) are sets of instance clusters assigned to classes A and B respectively. The overlap coefficient was chosen as a similarity metric to reduce the impact of dataset population sizes. If the first dataset is populated to a lesser degree than the second one, then for most classes $|c(A)| << |c(B)|$. In this case relatively small changes in $|c(B)|$ would have a big impact on such distance metrics as Jaccard score or Dice coefficient, while different values of $|c(A)|$ would not change the value significantly.
2. Choosing the "best match" mapping among several options. It is possible that for the same class, A, several relations are produced, which connect it to classes at different levels of the hierarchy. For instance, we can have both *overlapsWith(A, B)* and *overlapsWith(A, C)*, where $B \sqsubseteq C$. In our scenario the relation with a more generic class will always override the more specific one: it will mean that individuals of A will have to be compared to individuals of C. Only one such relation should be chosen, and the original overlap coefficient value cannot be used as a criterion: if $|c(A)| \leq |c(B)|$, then the relation $sim(A,B) \leq sim(A,C)$ always holds. Selecting the relation with the more generic class will mean that possibly more coreference resolution links will be discovered between individuals of A and $C \setminus B$. On the other hand, if the overlap between A and $C \setminus B$ is small and $|C \setminus B|$ is big, then more erroneous mappings can be produced and the damage to results quality due to the loss of precision will be higher than a possible gain from recall increase. To make this decision we use the following criterion:

$$\frac{(|A \cap C| - |A \cap B|)/|A \cap C|}{(|C| - |B|)/|C|} \geq \lambda,$$

   where λ reflects both the expected ratio of errors for the instance coreference resolution algorithm and relative importance of precision comparing to recall. If the inequality holds, then *overlapsWith(A, C)* is chosen, otherwise *overlapsWith(A, B)* is preferred.

In our tests we used an additional restriction: pairs of classes (A, B) where either $|A| = 1$, $|B| = 1$ or $|A \cap B| = 1$ were ignored. This was done to filter

out weak overlap mappings such as the one between *foaf:Person@DBLP* and *yago:PeopleFromRuralAlberta*, which led to noise at the instance-level matching stage.

In general, schema-level mappings obtained by the system in this way can be saved and used on its own in any scenario where individuals stored in several datasets have to be queried by class. As was said before, in our approach we focus on one specific use case scenario where these mappings are reused at the "top-down" stage of the KnoFuss workflow to produce coreference resolution links between individuals and improve the recall in comparison with existing relations.

5 Exploiting Inferred Schema Mappings for Coreference Resolution: The "Top-Down" Stage

The schema-level mappings obtained at the previous stage are used to identify sets of individuals in different repositories, which are likely to contain equivalent individuals not discovered before. These sets of relevant schema-level mappings are provided as input to the *instance transformation* stage of the KnoFuss tool (Fig. 3). It uses schema mappings to translate SPARQL queries, which select sets of individuals to be compared, from the vocabulary of one ontology into the terms of another one. It is possible that a class in one ontology is found to be connected to several classes in another ontology (not related via a *rdfs:subClassOf* relation). Such mappings are aggregated into a single *ClassUnion* mapping. For instance, in our DBLP example to select individuals from the DBLP dataset we use the following query:

SELECT *?uri* **WHERE**
{ *?uri rdf:type foaf:Person*}

To select potentially comparable individuals from the DBPedia repository this query is translated into:

SELECT *?uri* **WHERE**
{ {*?uri rdf:type yago:AmericanComputerScientists*}
UNION { *?uri rdf:type yago:GermanComputerScientists*}
UNION { *?uri rdf:type yago:GoogleEmployees*}
UNION... }

Using these translated queries individuals from both repositories are processed in the same way as if they shared the same schema. The system can employ several basic matching techniques, which can be selected and configured depending on the type of data as described in [2].

To avoid redundancy and potential errors, individuals, which were already connected either directly or indirectly via a third-party dataset, are excluded from analysis. The final set of instance-level mappings produced by the tool then can be added to existing ones.

A decision, which has to be taken at this stage, concerns the choice, whether two subsets of individuals should be compared at all: it is possible that all relevant

links between individuals were already discovered and the algorithm can only produce errors. A naive way to decide it could be to use an upper threshold on the overlap degree between classes assuming that many existing links between individuals can mean that all possible ones were already discovered. However, this misses important information regarding how existing mappings were produced. Using a coreference resolution system in combination with a link maintenance and trust computation system such as idMesh [20] can be interesting.

6 Evaluation

For our initial experiments we used three scenarios mentioned before:

1. Finding equivalence links between individuals representing people in DBPedia and DBLP (auxiliary dataset: YAGO, gold standard size 1229).
2. Finding equivalence links between *movie:music_contributor* individuals in LinkedMDB and corresponding individuals in DBPedia (auxiliary dataset: Musicbrainz, gold standard size 942).
3. Finding *movie:relatedBook* links between *movie:film* individuals in LinkedMDB and books mentioned in DBPedia (auxiliary dataset: RDF Book Mashup, gold standard size 419).

Our goal was to check the applicability of our approach in general and, in particular, the possibility to improve coreference resolution recall in comparison with already existing links. Thus, all three scenarios were of relatively small scale so that both precision and recall could be checked manually and the actual coreference resolution was performed using simple label-based similarity (Jaro metric[10]). Test results (precision, recall and F1 measure) are given in the Table 1. For each scenario three sets of results are provided:

- Baseline, which involves computing the transitive closure of already existing links.
- Results obtained by the algorithm when applied to all comparable individuals.
- Combined set of existing results and new results obtained by the algorithm.

As was expected, in all cases applying instance-level coreference resolution using automatically produced class-level mappings led to improvement in recall due to the discovery of previously missed mappings, and to a better overall performance, as measured by the F1-measure. In all cases, the best performance and F1-measure was achieved by combining newly produced mappings with existing ones. It means that the algorithms, which produced these sets of links, could generate complementary results and no set of links was redundant.

Obviously, the precision of the combined set of links was lower than the precision of the best algorithm in all three tests. In our tests this decrease was relatively small and was compensated by the increase in recall. However, in cases where the same data were already processed by algorithms of higher quality, the

[10] In the first two scenarios the metric was adapted for personal names, e.g., to match complete name with initials.

Table 1. Test results

Dataset	Test	Precision	Recall	F1
DBPedia vs DBLP	Baseline	0.90	0.14	0.25
	All individuals	0.95	0.88	0.91
	Combined set	0.93	0.89	0.91
LinkedMDB vs DBPedia (music contributors)	Baseline	0.99	0.68	0.81
	All individuals	0.98	0.91	0.94
	Combined set	0.98	0.97	0.98
LinkedMDB vs DBPedia (books)	Baseline	0.97	0.82	0.89
	All individuals	0.98	0.90	0.93
	Combined set	0.96	0.97	0.96

situation can be different. It makes the issue of tracing provenance of existing links important, as mentioned in section 5.

Considering the schema matching stage we found two factors which were potential causes of errors. The first factor was insufficient evidence. When only a small number of existing coreference links are available as evidence, distinguishing between "weakly overlapped" and "strongly overlapped" classes is problematic. For example, in the DBPedia-DBLP scenario the class *yago: FellowsOfWolfsonCollege,Cambridge* received a higher overlap score with *foaf: Person@DBLP* than the class *yago:IsraeliComputerScientists*, which in fact was strongly overlapped. This happened because for both of these classes there were only 2 evidence links available and the class *yago: FellowsOfWolfsonCollege,Cambridge* contained fewer instances. At the coreference resolution stage instances of such weakly overlapped classes caused the majority of false positive mappings because of name ambiguity.

The second factor concerned the quality of the ontologies themselves and of the class assertion statements. For instance, in the DBPedia dataset many musicians were not assigned to an appropriate class *dbpedia:MusicalArtist* but instead were assigned to more general classes *dbpedia:Artist* or even *dbpedia:Person*. As a result the "best fit" mappings produced by the algorithm did not correspond to the originally intended meaning of classes, because this originally intended meaning was not followed in the dataset (e.g., based on instance data the class *movie:music_contributor* was mapped to the class *dbpedia:Artist* instead of *dbpedia:MusicalArtist*). More serious issues involved instances being assigned to classes, which were actually disjoint (e.g., the individual *dbpedia:Jesse_Ventura* was classified as both a *dbpedia:Person* and a *dbpedia:TelevisionShow*). While in our scenarios spurious schema mappings caused by these errors were filtered out by the threshold, in other cases their impact can be significant. Explicit specification of ontological constraints can help to deal with such situations.

7 Conclusion and Future Work

In this paper we presented an approach which uses existing links between individuals stored in different repositories as evidence to generate schema-level

mappings between classes. A distinctive feature of our approach is the use of information defined in third-party datasets as background knowledge to enhance instance-based ontology matching techniques. These mappings are then used to discover new coreference links between individuals, which were missed before. Our initial experiments have shown an improvement in resulting coreference resolution performance in comparison with existing sets of links. However, there are still issues, which have to be resolved in the future work.

First, we plan to continue our experiments with instance-based schema alignment algorithms over different public datasets in order to evaluate their capabilities when applied to large networks of connected datasets and determine factors and conditions in real-world datasets, which influence their performance, in order to improve the reusability of the approach. In particular, one such factor is the presence of subsets of individuals, for which there are few or no connections available to infer any schema-level patterns.

In the test scenarios described in this paper there was no need for matching properties in addition to classes: only data described by standard attributes such as *rdfs:label*, *foaf:name* and *dc:title* was available. In the future inferring schema mappings between properties and reuse of axioms should be elaborated.

Considering the data-level integration stage, as mentioned in the section 5, there is an issue of automatic assessment of datasets and distinguishing the cases when new coreference links can be discovered with a sufficient precision.

Finally, there are infrastructural issues, which have to be taken into account to make the approach reusable. In particular, this concerns storing, publishing and maintaining both schema-level and instance-level links. There are several interesting directions to follow, such as applying the coreference bundles approach [19] instead of maintaining sets of pairwise *owl:sameAs* links and integrating with the *idMesh* approach [20], which reasons about sets of coreference links and their reliability.

Acknowledgements

This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) (to appear)
2. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: Integration of semantically annotated data by the KnoFuss architecture. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 265–274. Springer, Heidelberg (2008)
3. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4), 334–350 (2001)
4. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)

5. Gracia, J., Mena, E.: Matching with CIDER: Evaluation report for the OAEI 2008. In: 3rd Ontology Matching Workshop (OM 2008), Karlsruhe, Germany (2008)
6. Zhang, X., Zhong, Q., Li, J., Tang, J., Xie, G., Li, H.: RiMOM results for OAEI 2008. In: 3rd Ontology Matching Workshop (OM 2008), Karlsruhe, Germany (2008)
7. Udrea, O., Getoor, L., Miller, R.J.: Leveraging data and structure in ontology integration. In: SIGMOD 2007, Beijing, China, pp. 449–460 (2007)
8. Aleksovski, Z., Klein, M.C.A., ten Kate, W., van Harmelen, F.: Matching unstructured vocabularies using a background ontology. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 182–197. Springer, Heidelberg (2006)
9. Sabou, M., d'Aquin, M., Motta, E.: Exploring the Semantic Web as background knowledge for ontology matching. In: Spaccapietra, S., Pan, J.Z., Thiran, P., Halpin, T., Staab, S., Svatek, V., Shvaiko, P., Roddick, J. (eds.) Journal on Data Semantics XI. LNCS, vol. 5383, pp. 156–190. Springer, Heidelberg (2008)
10. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of American Statistical Association 64(328), 1183–1210 (1969)
11. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
12. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD 2005, pp. 85–96. ACM, New York (2005)
13. Kalashnikov, D.V., Mehrotra, S.: Domain-independent data cleaning via analysis of entity-relationship graph. ACM Transactions on Database Systems 31(2), 716–767 (2006)
14. Saïs, F., Pernelle, N., Rousset, M.C.: L2R: a logical method for reference reconciliation. In: AAAI 2007, Vancouver, BC, Canada, pp. 329–334 (2007)
15. Shen, W., DeRose, P., Vu, L., Doan, A., Ramakrishnan, R.: Source-aware entity matching: A compositional approach. In: ICDE 2007, Istanbul, Turkey (2007)
16. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk - a link discovery framework for the web of data. In: Workshop on Linked Data on the Web (LDOW 2009), Madrid, Spain (2009)
17. Hassanzadeh, O., Consens, M.: Linked movie data base. In: Workshop on Linked Data on the Web (LDOW 2009), Madrid, Spain (2009)
18. Bouquet, P., Stoermer, H., Bazzanella, B.: An Entity Name System (ENS) for the Semantic Web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 258–272. Springer, Heidelberg (2008)
19. Glaser, H., Jaffri, A., Millard, I.: Managing co-reference on the semantic web. In: Workshop on Linked Data on the Web (LDOW 2009), Madrid, Spain (2009)
20. Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K., de Meer, H.: idMesh: Graph-based disambiguation of linked data. In: WWW 2009, Madrid, Spain, pp. 591–600. ACM, New York (2009)