Data Linking: Capturing and Utilising Implicit Schema-level Relations

Andriy Nikolov
a.nikolov@open.ac.uk
Knowledge Media Institute
Open University
Milton Keynes, UK

Victoria Uren
v.uren@dcs.shef.ac.uk
Department of Computer Science
University of Sheffield
Sheffield, UK

Enrico Motta
e.motta@open.ac.uk
Knowledge Media Institute
Open University
Milton Keynes, UK

ABSTRACT

Schema-level heterogeneity represents an obstacle for automated discovery of coreference resolution links between individuals. Although there is a multitude of existing schema matching solutions, the Linked Data environment differs from the standard scenario assumed by these tools. In particular, large volumes of data are available, and repositories are connected into a graph by instance-level mappings. In this paper we describe how these features can be utilised to produce schema-level mappings which facilitate the instance coreference resolution process. Initial experiments applying this approach to public datasets have produced encouraging results.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous;
D.2 [**Software**]: Software Engineering

Keywords

Data fusion, coreference resolution, linked data

1. INTRODUCTION

The Web of Data is constantly growing [1], and the coreference links between data instances stored in different repositories represent a major added value of the Linked Data approach. These links connect individuals which refer to the same real-world entities using different URIs. Based on these links, it is possible to combine bits of information about the same real-world entity which were originally stored in several physical locations. Because of the large amount of data, it is not possible to generate these links manually, and automatic coreference resolution tools are used. However, the usage of these tools is complicated by semantic heterogeneity between repositories: although reusing common terminologies (e.g., FOAF[1] or Dublin Core[2]) is encouraged [1], existing repositories often use their own schemas. If repositories use different ontological schemas, it is not clear which sets of individuals should be compared by the coreference resolution tool, and which properties can be used to measure similarity between individuals. Thus, as a preprocessing step for generating coreference links between individuals, it is desirable to align schema terms in an automated way as well.

Although the schema matching task (discovering mappings betweem classes and properties) is an established research topic both in the database and the Semantic Web communities [4], the Linked Data environment has its specific features which are not utilised by existing methods and can be exploited to support the schema matching process. In particular:

- It is possible to consider several interlinked datasets in combination instead of comparing each pair in isolation and to involve information contained in third-party datasets as background knowledge to support matching.
- Large volumes of instance data are available, which makes it possible to learn and exploit data patterns not represented explicitly in the ontologies.
- Actual relations between concepts and properties are fuzzy and cannot be adequately captured using description logic terms: i.e., we are dealing with relations like “class overlap” or “relation overlap” rather than strict equivalence and subsumption.

In this paper we describe how these features can be utilised to perform schema-level matching between Linked Data repositories and, in turn, to facilitate instance coreference resolution. We have implemented this approach and obtained encouraging results in the test experiments.

2. RELATED WORK

The problem of instance-level coreference resolution is well-recognised in the Linked Data community [1]. Although in some key property values (inverse functional properties) can be compared [6], this is not sufficient in general case. In order to deal with it, the methods developed in the database community are commonly adopted, in particular, determining equivalence based on aggregated attribute-based similarity [5] and the use of string similarity to compare property values [3]. For example, these principles are implemented in SILK [10]. However, applying such a tool to a new pair of datasets requires significant user effort: the user has to specify which sets of individuals from two datasets are potentially overlapping, which attributes should be compared,

[1] http://xmlns.com/foaf/0.1/
[2] http://www.purl.org/dc/

LDOW2010, April 27, 2010, Raleigh, USA.
.

and which similarity metrics should be used for comparison. If afterwards one of the datasets has to be connected to another repository which uses a different schema, the user has to redefine these settings.

To minimise this user effort, it is currently a common practice that a newly published repository is only linked to one or a few "hub" repositories. DBPedia is the most popular generic "hub" repository, while there are also several domain-specific ones (e.g., Geonames for geographical data and Musicbrainz for music-related information). Then, in order to obtain complete information about a certain entity we need to compute a transitive closure of coreference links and gather all URIs used to represent this entity in different datasets. These transitive closures can be maintained in a centralised way [7][3] and the mutual impact of atomic mappings can be analysed [2]. However, this approach often leads to the loss of information. For example, it can happen that several datasets connected to the same "hub" repository mention the same entity under different URIs. If the "hub" repository itself does not mention this entity, then the coreference links between these URIs cannot be established. It is also possible that one of the intermediate coreference links is omitted due to an error of the coreference resolution tool.

In order to discover such missing links, the coreference resolution procedure has to be directly applied to the corresponding subsets of datasets which are linked via one or several intermediate repositories. To identify such corresponding subsets and comparable properties, the above-listed features of the Linked Data environment can be exploited. Because large volumes of data and partial sets of equivalence links are available, it is possible to apply the instance-based ontology matching techniques [4]. This is implemented in our approach.

3. USING BACKGROUND DATA FOR ONTOLOGY MATCHING

Our approach is a further extension of the work presented in [9]. Its main idea is to use a pre-existing set of instance-level links for two purposes:

- Infer schema-level relations between concepts and properties of two different repositories. For example, by analysing the LinkedMDB repository[4], which describes movies from the IMDB database, and DBPedia[5], which describes Wikipedia entries, together with their incoming and outgoing instance-level links, we can establish relations between their classes *movie:music_contributor* and *dbpedia:Artist*, and between properties *movie:actor* and *dbpedia:starring*. These schema-level mappings can afterwards be utilised by an instance-level coreference resolution tool.
- Infer data patterns which hold for instances of these concepts and properties. For example, it is possible to infer that identical movies usually have the same release year and overlapping sets of actors. Later, these patterns can be used to highlight the existing identity links which violate these patterns and are likely to be spurious.

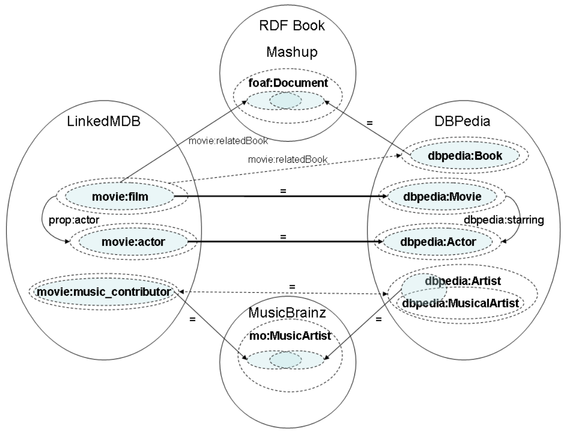

Figure 1: LinkedMDB and DBPedia: exploiting instance-level coreference links with third-party datasets. Solid arrows show existing *owl:sameAs* (=) and *movie:relatedBook* links. Dashed arrows connect sets containing potentially omitted links.

In the following subsections we will describe these two parts of our approach in more detail using illustrative examples from actual Linked Data repositories.

3.1 Inferring schema-level mappings

In order to produce schema-level mappings between two data repositories based on existing instance-level links, the Linked Data environment allows two types of background knowledge to be utilised:

- *Data-level evidence.* This includes instance coreference links between the two repositories being analysed and third-party repositories. These links can be aggregated to indicate potentially overlapping sets of individuals in two original datasets.

- *Schema-level evidence.* This includes ontological schemas used in third-party repositories. Schema-level evidence can be utilised when (a) one dataset uses two different vocabularies which model the domain with different levels of detail or (b) the same schema is reused by several repositories. This schema-level evidence can provide additional insights into the meaning of concepts and properties based on their usage.

3.1.1 Data-level evidence

Let us consider an example shown in Fig. 1. The LinkedMDB repository 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

[3] http://www.sameas.org, http://www.rkbexplorer.com
[4] http://data.linkedmdb.org/
[5] http://dbpedia.org

to related books in RDF Book Mashup[6], but not to books mentioned in DBPedia. Partially, such mappings can be obtained by computing a transitive closure for individuals connected by coreference links via intermediate repositories (MusicBrainz[7] for composers and Book Mashup for books). In this way, many links are not discovered because of omissions of an intermediate link in a chain (e.g., 32% of *movie: music_contributor* instances were not connected to corresponding DBPedia instances). Such links can be discovered by applying an instance coreference resolution tool (like SILK [10] or KnoFuss [8]) directly to corresponding subsets of LinkedMDB and DBPedia. However, in order to apply them, it is necessary to separate these corresponding subsets from irrelevant data, in other words, to specify mappings between classes which are likely to contain identical individuals.

In this situation, we can use our schema matching approach which includes the following steps:

1. Combining identical individuals into clusters. At this stage all identical individuals from a set of datasets are combined into clusters based on the transitive closure of existing *owl:sameAs* relations.

2. Establishing relations between clusters and schema terms. For example, if one individual in the cluster belongs to the class *dbpedia:Artist*, then we say that the whole cluster belongs to this class. The same applies for properties of each individual in the cluster.

3. Inferring mappings between schema terms using instance set similarity. Instead of strict *owl:equivalentClass* or *owl:subClassOf* relations we produce fuzzy relations *#overlapsWith*. Formally this relation is similar to the *umbel: isAligned* property defined in the Umbel vocabulary[8] and states that two classes share a subset of their individuals. This relation has a quantitative measure (a number between 0 and 1) which is used 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 measure is computed as the value of the overlap coefficient:

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

where c(A) and c(B) - sets of instance clusters assigned to classes A and B respectively. The strength of a relation between properties is computed as

$$sim(r_1, r_2) = \frac{|c(X)|}{|c(Y)|},$$

where c(X) - a set of clusters which have equivalent values for properties r_1 and r_2 and c(Y) - a set of all clusters which have values for both properties r_1 and r_2.

Resulting mappings are filtered by comparing the strength of each relation with a pre-defined threshold, and weak mappings are removed from the resulting set. The resulting set of mappings is passed to the coreference resolution tool (in our case, KnoFuss) which compares instances belonging to mapped classes and generates instance coreference mappings.

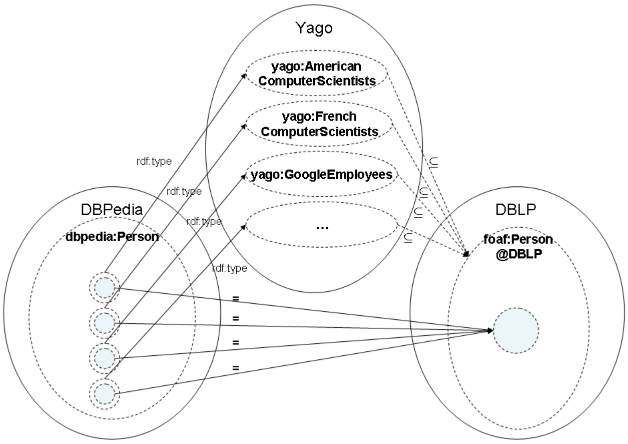

Figure 2: DBPedia and DBLP: exploiting schema-level links with third-party datasets. Solid arrows show existing *owl:sameAs* (=) and *rdf:type* links. Dashed arrows represent discovered schema relations. The system identifies the subset of *dbpedia:Person* instances, which overlaps with DBLP *foaf:Person* instances, as a union of classes defined in Yago.

3.1.2 Schema-level evidence

In the example shown in Fig. 1 the main source of background knowledge are existing instance-level coreference links with third-party repositories (MusicBrainz and Book Mashup). One case when schema-level evidence can be utilised is when instances in a dataset are linked to a schema used by a third-party repository. For example, in Fig. 2 both DBPedia and DBLP contain individuals representing the same computer scientists. However, only a small proportion of these individuals is explicitly linked by *owl:sameAs* mappings (196 links). Applying automatic coreference resolution, which could derive more mappings, 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 are assigned to a generic class *dbpedia:Person* and not distinguished from other people.

Using name similarity to produce mappings between instances is likely to produce many false positive links due to ambiguity of personal names. The source of the schema-level evidence which can resolve this issue is the Yago repository[9]. The Yago ontology is based on Wikipedia categories and provides a more detailed hierarchy of classes than the DBPedia ontology. Using the procedure described in section 3.1.1, we can approximate the boundaries of the DBPedia subset which overlaps with DBLP. The algorithm returns a set of mappings between the Yago classes and the *foaf:Person* class in DBLP, e.g., between *foaf:Person* and *yago:MicrosoftEmployees* and between *foaf:Person* and *yago:BritishComputerScientists*. Having these mappings, the

[6] http://www4.wiwiss.fu-berlin.de/bizer/bookmashup/

[7] http://dbtune.org/musicbrainz/

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

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

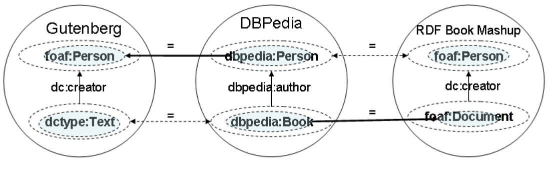

Figure 3: Gutenberg/DBPedia/Book Mashup: Aligning relations *dc:creator* and *dbpedia:author* which have strongly overlapping domains (Book Mashup/DBPedia) and ranges (Gutenberg/DBPedia)

instance-level coreference resolution can be applied only to instances of mapped classes and produce results with higher accuracy.

Another scenario where schema-level evidence can be utilised is the case when one ontology is reused in several repositories. Then data from all these repositories can be used to reason about the usage patterns of the terms of this ontology. For example, in Fig. 3 three datasets (Gutenberg project[10], RDF Book Mashup, and DBPedia) describe books and their authors, and two of them (Book Mashup and Gutenberg) use the Dublin Core vocabulary. There exist a set of *owl:sameAs* links between books in RDF Book Mashup and DBPedia, and a set of links between authors in DBPedia and Gutenberg project. However, there are no links between Book Mashup and DBPedia authors or between Gutenberg and DBPedia books. Again, direct classes *foaf:Person* and *dbpedia:Person* are too generic to provide useful input for the coreference resolution stage: comparing all Person individuals in DBPedia and RDF Book Mashup is likely to produce many spurious mappings between people having the same names. But, using evidence from all three repositories, it is possible to establish a relation between the properties *dc:creator* and *dbpedia:author*. Based on the set of co-referent books from DBPedia and Book Mashup we can infer that these properties have the same domain, and based on the mappings between authors we can infer that they have the same range. Given that no property in DBPedia is a stronger candidate for the matching relation, we can produce the schema-level mapping between properties *dc:creator* and *dbpedia:author*. After that we can establish missing relations from Gutenberg to DBPedia (books) and from Book Mashup to DBPedia (authors) by comparing individuals which are connected via these properties to already mapped ones.

When processing schema-level evidence, it is important to bear in mind that the same ontological terms can be used in different repositories with different interpretation: e.g., in our example, in DBLP a generic *foaf:Person* class in fact refers only to people related to computer science.

3.2 Inferring data patterns and refining the set of existing mappings

It is sometimes the case that the existing set of *owl:sameAs* mappings contains spurious mappings connecting distinct individuals: it is hard to avoid errors when an automatic coreference resolution tool applies some fuzzy similarity metrics to process large amounts of data. If the resulting set of mappings is large, it is not feasible to check their correctness manually. However, by analyzing the data patterns it is possible to select subsets of mappings which are more likely to contain spurious mappings and highlight them. For instance, in the example shown in Fig. 1, we established the relations between pairs of properties $\{$*movie:actor*; *dbpedia:starring*$\}$ ($sim = 0.98$) and $\{$*movie:initial_release_date*; *dbpedia:releasedDate*$\}$ ($sim = 0.96$). In other words, most equivalent movies have the same release date and are related to overlapping sets of actors. Thus, we can hypothesise that the mappings between individuals representing movies where these patterns do not hold are more likely to be spurious.

Currently, our algorithm can infer two kinds of patterns corresponding to the functionality and inverse functionality property restrictions:

- *Property value equivalence.* This states that for a pair of aligned properties $\{r_1,\ r_2\}$, equivalent individuals $I_1 \equiv I_2$ should have equivalent values: $r_1(I_1, x)$, $r_2(I_2, y)$, $x \equiv y$
- *Property subject equivalence.* This states that for a pair of aligned properties $\{r_1,\ r_2\}$, if the objects of these properties are equivalent individuals $I_1 \equiv I_2$, the subjects should be equivalent as well: $r_1(I_3, I_1)$, $r_2(I_4, I_2)$, $I_3 \equiv I_4$.

It is important to note that these data patterns can be useful for refinement of existing mapping sets only if they were not taken into account by the original instance coreference resolution algorithm. Otherwise, they become tautological: e.g., by analysing a set of mappings produced by computing label similarity we can infer that equivalent instances usually have similar labels. Therefore, the refinement procedure can be used in two cases: (i) where the provenance of original mappings is available and the algorithm which produced them is known, or (ii) where a significant body of new evidence is discovered, e.g., a new set of instance mappings as a result of the process described in section 3.1.

4. EXPERIMENTS

In order to test our approach, we experimented with existing Linked Data repositories mentioned in our examples:

1. Finding equivalence links between individuals representing people in DBPedia and DBLP[11] (auxiliary dataset: Yago, gold standard size 1229).
2. Finding equivalence links between *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).

[10] http://www4.wiwiss.fu-berlin.de/gutendata/

[11] DBLP contains a substantial proportion of internal coreference errors: e.g., several authors having the same URI or the same person having several URIs. In our tests we did not consider these issues: e.g., a mapping between instances from DBLP and DBPedia was considered correct if the DBLP instance was linked to at least one publication which was written by the person represented by the DBPedia instance.

Table 1: Test results

N	Dataset	Test	Precision	Recall	F1
1	DBPedia/DBLP	Baseline	0.90	0.14	0.25
		Aligned	0.93	0.89	0.91
2	LinkedMDB/DBPedia (composers)	Baseline	0.99	0.68	0.81
		Aligned	0.98	0.97	0.98
3	LinkedMDB/DBPedia (books)	Baseline	0.97	0.82	0.89
		Aligned	0.96	0.97	0.96
4	LinkedMDB/DBPedia (films)	Baseline	0.993	1.0	0.996
		Aligned	1.0	0.999	1.0
5	Gutenberg/ DBPedia (books)	Baseline	N/A	N/A	N/A
		Aligned	1.0	1.0	1.0
6	Book Mashup/ DBPedia (authors)	Baseline	N/A	N/A	N/A
		Aligned	1.0	1.0	1.0

4. Refining existing equivalence links between *movie:film* individuals in LinkedMDB and corresponding individuals mentioned in DBPedia by analysing related actors and release dates (total link set size 18512).

5. Finding equivalence links between books mentioned in Gutenberg project and DBPedia (auxiliary dataset: RDF Book Mashup, gold standard size 1201).

6. Finding equivalence links between book authors mentioned in DBPedia and RDF Book Mashup, (auxiliary dataset: Gutenberg project, gold standard size 1235).

These tests were relatively small scale due to the need to construct gold standard mappings manually. In the tests we initially applied our instance-based schema-matching algorithm to the datasets to obtain schema-level relations. Then, these relations were passed as input to our data-level coreference resolution tool KnoFuss, which processed the datasets to discover *owl:sameAs* links between instances. As a similarity measure, we used Jaro string similarity applied to the label. Test results (precision, recall and F1 measure) are given in the Table 1. Two sets of results are provided: (i) baseline, which involves computing transitive closure of already existing links[12], and (ii) combined set of existing results and new results obtained by the algorithm after the schema alignment. As was expected, the usage of automatically produced schema alignments led to an improvement in recall (rows 1, 2, 3, 5, 6) because initially missed links were discovered. In case 4 precision was affected because the mappings which did not conform to the data pattern were removed. The change in precision was small due to the large size of the dataset (140 mappings were removed from the set of 18512), however, the precision of the refinement procedure was high: out of 140 mappings identified as potentially incorrect, 132 were actually incorrect.

When analysing the results of the experiments, we looked into the limiting factors which caused errors. The most important factor involved the quality of the datasets themselves, in particular, improper use of schema entities and incorrect data statements. For example, in the tests where inferred data patterns were applied to the filter out incorrectly linked *movie:film* entities (row 4), we found that the equivalence of release dates cannot be used as a restriction on its own: in about 50% of cases the mapping was correct, while the release date was not provided correctly in one of the datasets. Incomplete information could also lead to problems: for example, 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 mapping was established between classes *movie: music_contributor* and *dbpedia:Artist* instead of *dbpedia: MusicalArtist*. As a result, KnoFuss had to be applied to a larger set containing many irrelevant individuals and produced several erroneous coreference links between movie composers and non-musical artists. Given that occurrences of incorrect data are inevitable in the Linked Data environment, these issues have to be taken into account when designing matching algorithms.

5. FUTURE WORK

In this paper, we described an approach which captured schema-level relations between linked data repositories based on available instance data and reused these relations to facilitate generation of new coreference links. In our experiments, we applied this approach to small subgraphs of the Linked Data cloud. In future, we plan to analyse the "schema cloud" consisting of schema vocabularies used by Linked Data repositories in combination with the "data cloud" including the datasets connected by instance-level links. In particular, it is interesting to investigate how the usage patterns of the same vocabularies differ between repositories, to which extent it is possible to capture relations between their terms, and under which conditions these relations can be utilised to support the coreference resolution process.

6. REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *IJSWIS*, 5(3):1–22.

[2] P. Cudré-Mauroux, P. Haghani, M. Jost, K. Aberer, and H. de Meer. idMesh: Graph-based disambiguation of linked data. In *WWW 2009*, 2009.

[3] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE KDE Transactions*, 19(1):1–16, 2007.

[4] J. Euzenat and P. Shvaiko. *Ontology matching.* Springer-Verlag, Heidelberg, 2007.

[5] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *JASA*, 64(328):1183–1210, 1969.

[6] A. Hogan, A. Harth, and S. Decker. Performing object consolidation on the Semantic Web data graph. In *I3 Workshop, WWW2007*, Banff, Canada, 2007.

[7] A. Jaffri, H. Glaser, and I. Millard. Managing URI synonymity to enable consistent reference on the Semantic Web. In *IRSW2008 Workshop*, Tenerife, Spain, 2008.

[8] A. Nikolov, V. Uren, E. Motta, and A. de Roeck. Integration of semantically annotated data by the KnoFuss architecture. In *EKAW 2008*, Acitrezza, Italy, 2008.

[9] A. Nikolov, V. Uren, E. Motta, and A. de Roeck. Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In *ASWC 2009*, 2009.

[10] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *ISWC 2009*, 2009.

[12] As was said in section 3.1.2, for Gutenberg and Book Mashup we did not have any baseline links available.