Complex Schema Mapping and Linking Data: Beyond Binary Predicates

Jacobo Rouces
Aalborg University, Denmark
jrg@es.aau.dk

Gerard de Melo
Tsinghua University, China
gdm@demelo.org

Katja Hose
Aalborg University, Denmark
khose@cs.aau.dk

ABSTRACT

Currently, datasets in the Linked Open Data (LOD) cloud are mostly connected by properties such as `owl:sameAs`, `rdfs:subClassOf`, or `owl:equivalentProperty`. These properties either link pairs of entities that are equivalent or express some other binary relationship such as subsumption. In many cases, however, this is not sufficient to link all types of equivalent knowledge. Often, a relationship exists between an entity in one dataset and what is represented by a complex pattern in another, or between two complex patterns. In this paper, we present a method for linking datasets that is expressive enough to support these cases. It consists of integration rules between arbitrary datasets and a mediated schema. We also present and evaluate a method to create these integration rules automatically.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.1.0 [**Information Systems**]: Models and Principles—*General*

Keywords

Linked Open Data, Schema matching, Heterogeneous knowledge, Linguistic frames, Data integration

1. Introduction

The most common way in which datasets in the Linked Open Data (LOD) cloud are currently connected to each other is by means of triples expressing simple one-to-one relationships. The most well-established property is `owl:sameAs`, which indicates equivalence between entities. Others, such as `owl:equivalentClass` and `owl:equivalentProperty` describe equivalences for classes and properties. Previous work has exposed widespread cases of misuse of the `owl:sameAs` property and proposed alternatives expressing different forms of near-identity [4, 8]. Additionally, properties such as `rdfs:subClassOf` and `rdfs:subPropertyOf` denote subsumption between classes and properties. Still, all of these properties have in common that they are binary predicates. Thus, they always link two individual items.

Knowledge in different datasets can, however, be related in other ways than via a direct one-to-one relationship between a pair of entities. Often, a relation may exist between an entity in one dataset and what is captured using a complex pattern in another, or between two complex patterns. For example, when using binary predicates, an example class such as `BirthEvent` in one dataset cannot simply be linked to a property such as `bornOnDate` or `bornInPlace` from another schema or dataset. Yet, these two sources clearly capture the same sort of knowledge, especially if the class `BirthEvent` is the domain of properties such as `personBorn`, `date`, and `place`. The first-order logic expression in Figure 1 formalizes one of these more complex relations. Even more complex patterns are possible, as illustrated in Figure 2. In this example, the complex pattern covers more information than captured by the simpler pattern using the property `stopConstructionOf`.

In contrast to binary relationships, related work has only paid minimal attention to complex patterns. The EDOAL format [3] and an ontology of correspondence patterns [14] have been created as a way to express and categorize complex correspondences between ontologies. These methods, however, do not address the actual integration, i.e., the method to establish these relationships. The iMAP system [5] explores a space of possible complex relationships between the values of entries in two relational databases, for instance `address = concat(city,state)`. Ritze et al. [10] use a rule-based approach for detecting specific kinds of complex alignment patterns between entities in small ontologies.

In this paper, we generalize to a more general method for linking datasets through a mediated schema, that can support cases such as the above-mentioned examples. In addition, we propose an automatic approach to create some complex integration rules, which can be combined with existing 1-to-1 links. We use a mediated schema as a hub because the resulting star topology reduces the complexity of the overall linking from quadratic to linear with respect to the number of datasets. More specifically, we use FrameBase [11–13], a rich schema based on linguistic frames, as the mediated schema, because it is highly expressive and possesses the metadata and structures that enable automatic creation of mappings. Additionally, it has a strong connection to natural language.

As SPARQL has become a common standard with the necessary expressiveness to support logical rules, we implement schema mappings and integration rules as SPARQL construct queries (Figures 1 and 2). However, the system can easily be adapted to other formalisms and implementations.

WWW2016 Workshop: Linked Data on the Web (LDOW2016).

$$
\begin{aligned}
&\forall v_1 v_2(\\
&\quad \exists e_1(\\
&\qquad t_f(e_1, a, BirthEvent) \wedge \\
&\qquad t_f(e_1, subject, v_1) \wedge \\
&\qquad t_f(e_1, date, v_2) \wedge \\
&\quad) \\
&\quad \leftrightarrow \\
&\quad t_s(v_1, bornOnDate, v_2) \\
&)
\end{aligned}
$$

Figure 1: Complex relation between two schemas, expressed in first-order logic

$$
\begin{aligned}
&\forall v_1 v_2 \ (\\
&\quad \exists e_1 \ (\\
&\qquad t_f(e_1, a, Construction) \wedge \\
&\qquad t_f(e_1, createdEntity, v_1) \wedge \\
&\qquad t_f(e_2, a, StopProcess) \wedge \\
&\qquad t_f(e_2, cause, v_2) \wedge \\
&\quad) \\
&\quad \leftrightarrow \\
&\quad t_s(v_2, stopsConstructionOf, v_1) \\
&)
\end{aligned}
$$

Figure 2: Very complex relation between two schemas, expressed in first-order logic.

2. Complex Integration Rules

In order to connect complex patterns across different data sources, we develop an automatic method to produce integration rules that convert information from arbitrary sources to information expressed using the FrameBase schema. This method consists of three operations.

1. **Creating Candidate Properties in FrameBase:** We first identify complex patterns within FrameBase that might match properties from other sources. For each of these complex patterns, we define a new candidate property as a shorthand form, with a concise human-readable text label. All of this is done automatically by exploiting the linguistic annotations available in FrameBase. The result is a large set of matching candidates in FrameBase.
2. **Processing Candidate Properties in the Source Datasets:** We canonicalize properties in the source datasets by extending their names.
3. **Matching:** We match the (refined) names of the properties from the source dataset with the names of the binary candidate properties prepared for FrameBase. When a sufficiently high match is encountered, we produce an integration rule that connects the source property with the complex triple pattern.

2.1 Creating Candidate Properties in FrameBase

The first step is to identify complex patterns in FrameBase to which other data sources could potentially be matched. For each of these complex patterns, we define a simple new binary candidate properties that serves as a shorthand form between two variables present in the complex pattern. In FrameBase terminology, these new properties are called Direct Binary Predicates (DBPs) and their relationship to the original complex patterns in FrameBase is expressed via Reification–Dereification (ReDer) rules [11]. As a result, we obtain a large candidate set of binary predicates to which properties from other datasets can be matched.

In order to detect relevant complex patterns in FrameBase, we exploit its ties to natural language.

Binary Predicates Based on Verbs and Nouns

For relationships typically expressed using verbs, we have already equipped FrameBase with a set of direct binary predicates and corresponding reification-dereification rules [11]. Figure 3 illustrates the structure of such a ReDer rule, while Figure 4 provides an example of a DBP with the verb "destroyed" as the syntactic head.

```
?s  <DIRECT_BINARY_PREDICATE>  ?o
↕
<FRAME_INSTANCE>  a  <FRAME_CLASS>
<FRAME_INSTANCE>  <FRAME_ELEMENT-S>  ?s
<FRAME_INSTANCE>  <FRAME_ELEMENT-O>  ?o
```

Figure 3: The general pattern of a simple dereification rule

```
?s dbp-Destroying-destroyedByMeans ?o
↕
f type frame-Destroying-destroy.v
f fe-Destroying-Undergoer ?s
f fe-Destroying-Means ?o
```

Figure 4: Example of a simple dereification rule

Our previous work has already produced some DBPs based on verbs [11] (such as in the example in Figure 4), and nouns [13], such as `isCauseOfEffect`.

However, in all of these cases, the reified side of the Reification–Dereification rules (the one at the bottom in the examples) was restricted to a specific pattern using three lines, such as the ones in Figures 3 and 4.

Binary Predicates Based on Adjectives

To these noun and verb-based DBPs, we now add new binary predicates based on adjectives, using a generalization to more complex patterns such as the one illustrated in Figure 2. One can view these as *very complex patterns*, as they are more involved than the ones considered previously in Figure 3.

FrameNet [1], a database of frames used to annotate the semantics of natural language, forms the backbone of frames in FrameBase. In FrameNet, different frames represent different kinds of events or situations with participants, called Frame Elements (FEs). Frames also have Lexical Units (LUs), which are words and terms that are associated to that frame, and may evoke that frame when appear in a text. Example texts are *semantically parsed* by annotating places where a LU is evoking a frame, and neighboring words or phrases are the values of some of the FEs belonging to that

Creation Rule: Copula+Adjective
Create DBP with name "is LU PREP FE-O" *if*
ISADJECTIVE(LU) AND phrase-type-o==PP[PREP] AND grammatical-function-s==Ext AND grammatical-function-o==Dep

Figure 5: Method for creating new adjective-based DBPs

```
?s dbp-Sound_level-isLoudToDegree ?o
↕
f type frame-Sound_level-loud.a
f fe-Sound_level-Entity ?s
f fe-Sound_level-Degree ?o
```

Figure 6: Example of an adjective-based dereification rule using the copula "to be"

frame. FrameBase represents frames and LUs as classes that can be instantiated and FEs as properties whose frame is their domain.

Figure 5 summarizes how to create adjective-based DBPs using FrameNet's example annotations of English sentences. We define two target types of FEs: FE-s, the FE that should connect to the subject of the triple whose property is the DBP; and FE-o, the FE that should connect to the object. Figure 3 shows how these two FEs are used in a ReDer rule. For an adjective in a sentence that is annotated with a frame, we need to check whether the text annotation also contains two FEs that fulfill the following conditions. First, the phrase type of FE-o needs to be a prepositional phrase (PP) with preposition PREP. Second, the grammatical function of FE-s needs to be that of a subject (Ext). And third, the grammatical function of FE-o needs to be that of a dependent (Dep). Figure 6 presents an example of a DBP created with this method.

Although most occurrences of adjectives in the FrameNet annotations involve the verb "to be", pseudo-copulas "to seem" and "to become" can also be combined with any adjective. Therefore, we generate all possible DBPs with these three copulas for all adjectives. For "to be" there are no additional semantics (Figure 6). The pseudo-copulas, however, carry additional semantics, which are expressed in a more complex pattern with an additional frame instance (Figures 7 and 8). Figure 7 presents an example using the `Becoming` frame and the FE `fe-Becoming-Final_state` instead of `fe-Becoming-Final_category` (in FrameNet the former is used with adjectives and adjective phrases, while the latter is used with nouns and noun phrases). Figure 8 shows an example for the `Appearance` frame.

2.2 Processing Candidate Properties in the Source Datasets

Having prepared a set of candidate predicates, each standing for a complex pattern in FrameBase, we now turn to the source datasets that we wish to connect. For a given source dataset, we process all its properties. Property names are often composed of a single word that is highly polysemous. This is particularly true when the verbs "to be" or "to have" are omitted, which unfortunately is very often the case. For example, many datasets use property names such as `address` instead of `hasAddress`, or `father` instead of `isFatherOf`.

Our approach consists of the following six steps.

```
?s dbp-Sound_level-becomesLoudToDegree ?o
↕
f type frame-Sound_level-loud.a
f fe-Sound_level-Entity ?s
f fe-Sound_level-Degree ?o
f' type frame-Becoming-become.v
f' fe-Becoming-Entity ?s
f' fe-Becoming-Final_state f
```

Figure 7: Example of an adjective-based dereification rule using the pseudo-copula "to become"

```
?s dbp-Sound_level-seemsLoudToDegree ?o
↕
f type frame-Sound_level-loud.a
f fe-Sound_level-Entity ?s
f fe-Sound_level-Degree ?o
f' type frame-Appearance-seem.v
f' fe-Appearance-Phenomenon ?s
f' fe-Appearance-Inference f
```

Figure 8: Example of an adjective-based dereification rule using the pseudo-copula "to seem"

1 If the name p of a property is a past participle, it can be extended with the prefix "is" (without postfix "of").

2 If the name p of a property is a noun or a noun phrase, and a range is declared for the property, let X be a set containing p's name and the hypernyms of all its word senses (obtained from WordNet [6]). If for any element x in X, p is a substring of x or x is a substring of p, then p can be extended with the prefix "has".

3 The same rule as above, but using the domain instead of the range, which allows p to be extended with the prefix "is" and postfix "of".

4 If the property is symmetric, we can introduce extensions both with "has" and with "is"+ ... +"of".

5 For every property p corresponding to the pattern "is X of", an inverse property can be created of the form "has X".

6 For every property p corresponding to the pattern "has X", an inverse property can be created of the form "is X of".

This process resembles a sort of canonicalization of entity names [7], but in our case for properties. Note that steps 4–5 can also be carried out on the DBPs with identical results.

This canonicalization has independent value beyond its use for matching as in this paper; especially when the canonicalization, as in our case, does not merely make the names conform to a given pattern but also less ambiguous as well as easier to understand by humans.

2.3 Matching

The final step is to match properties across datasets. We focus on creating matches between direct binary predicates in FrameBase and the refined property names of other sources.

In order to find matches, we use bag-of-words cosine similarity measures that are optimized for the task at hand. We tokenize the names, but do not use stemming, since we want to increase specificity. We also do not perform stopword removal, because, unlike in the typical use case of matching large documents, common words such as prepositions can be relevant in this context (consider "run for" versus "run against").

Each source dataset property is compared to each DBP using a weighted combination of measures.

$$w_1 \cos(v_1^{\mathsf{SDP}}, v_1^{\mathsf{DBP}}) + w_2 \cos(v_2^{\mathsf{SDP}}, v_2^{\mathsf{DBP}}) + w_3 c_1 + w_4 c_2$$

- $\cos(v_1^{\mathsf{SDP}}, v_1^{\mathsf{DBP}})$ is the cosine between the vector for the name of the source dataset property v_1^{SDP} and the vector for the DBP's name v_1^{DBP}. For DBPs, we remove the "frame-element-object (FE-o)" name [11] because these do not occur very frequently. For instance, "original" is ommitted for "is copy of original".
- $\cos(v_2^{\mathsf{SDP}}, v_2^{\mathsf{DBP}})$ is the cosine between vectors with additional terms describing the properties' semantics. v_2^{SDP} includes terms from the name of the property, plus from the domain and the range if available. v_2^{DBP} includes the terms from the DBP's name, plus the FE-o, the FE-s, and the name and description of the associated frame as well as all its superframes.
- c_1 has value 1 if the frame element FE-o is classified as *Core FE* if FrameNet, which means that it instantiates a conceptually necessary component of a frame. These kinds of frames are more likely to appear. The value is also 1 if the FE is about Time or Place, because this information is also frequent in datasets.
- c_2 is the same for FE-s.

The DBP with the highest score is chosen, if this is higher than a threshold T. The vector of weights w is set to $w = (0.7, 0.1, 0.1, 0.1)$ so that the three last elements favor the closest match whenever there is a tie for $\cos(v_1^{\mathsf{SDP}}, v_1^{\mathsf{DBP}})$, which can happen between two DBPs that only differ by the FE-o name. $\cos(v_2^{\mathsf{SDP}}, v_2^{\mathsf{DBP}})$ is computationally more heavy and, for reasons of efficiency, it is only evaluated when $\cos(v_1^{\mathsf{SDP}}, v_1^{\mathsf{DBP}})$ is higher than Tw_1. The value of the global threshold is set at $T = w_1$ so $\cos(v_1^{\mathsf{SDP}}, v_1^{\mathsf{DBP}}) = 1$ is enough to fire a rule.

3. Results

We test our method on DBpedia [2]. We canonicalized 1,608 DBpedia properties and evaluated a random sample of 40, out of which 32 turned out to be correct. Of the 8 that were incorrect, 2 were also incorrect in their original DBpedia form, resulting in a true precision of 85%. Some examples are presented in Table 1.

Table 1: Example canonicalized properties.

source property IRI	
source property name	**canonicalization**
`http://dbpedia.org/property/currentlyRunBy`	
currently run by	is currently run by
`http://dbpedia.org/ontology/goldenRaspberryAward`	
golden raspberry award	has golden raspberry award
`http://dbpedia.org/ontology/statistic`	
statistic	is statistic of
`http://dbpedia.org/ontology/linkTitle`	
link title	has link title
`http://dbpedia.org/ontology/firstLeader`	
first leader	has first leader

We obtained a total of 315 integration rules (some examples below). We evaluated a random sample of 40, of which 29 were valid and symmetric, 1 was valid but mapped to a generalization of the meaning, 8 were wrong originating from a correct (yet sometimes with incomplete name) property, and 2 were wrong but also incorrect in DBpedia. The resulting precision for valid rules was 79%. Below we reproduce some obtained integration rules.

```
CONSTRUCT {
 _:r a :frame-Appearance-smell.v .
 _:r :fe-Appearance-Phenomenon ?S .
 _:r :fe-Appearance-Characterization ?O .
} WHERE {
 ?S <http://dbpedia.org/property/smellsLike> ?O .
}

CONSTRUCT {
 _:r a :frame-Residence-reside.v .
 _:r :fe-Residence-Resident ?S .
 _:r :fe-Residence-Location ?O .
} WHERE {
 ?S <http://dbpedia.org/property/residesIn> ?O .
}

CONSTRUCT {
 _:r a :frame-Experiencer_focus-dislike.v .
 _:r :fe-Experiencer_focus-Experiencer ?S .
 _:r :fe-Experiencer_focus-Content ?O .
} WHERE {
 ?S <http://dbpedia.org/property/dislikes> ?O .
}

CONSTRUCT {
 _:r a :frame-Possession-own.v .
 _:r :fe-Possession-Owner ?S .
 _:r :fe-Possession-Possession ?O .
} WHERE {
 ?S <http://dbpedia.org/ontology/owns> ?O .
}
```

This is an example of a wrong rule.

```
CONSTRUCT {
 _:r a :frame-Education_teaching-school.v .
 _:r :fe-Education_teaching-Student ?S .
 _:r :fe-Education_teaching-Skill ?O .
} WHERE {
 ?S <http://dbpedia.org/property/schooledAt> ?O .
}
```

4. Future Work

We are currently working on creating very complex patterns for certain DBPs whose syntactic head is a noun for which the governing verb (the verb whose object is the noun) adds semantics in a similar way as the pseudo-copulas in Section 2.1. Figure 9 shows an example of a very complex noun-based ReDer rule. In this case, it is not possible to work with all possible combinations of governing verbs as we did with the copulas, because many verbs will not make sense (compare "develop understanding" with "run understanding"). Therefore, we must use the governing verb from FrameNet's example sentences. Because many verbs can be

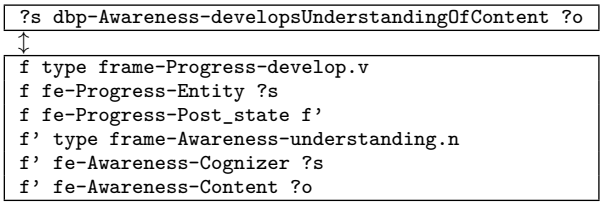

Figure 9: Example of a very complex noun-based ReDer rule

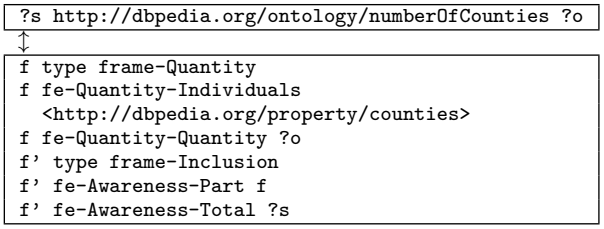

Figure 10: Example of a very complex integration rule to express amounts

associated with different frames, the right frame must be chosen on the reified side of the ReDer rule. Due to the high number of possible verbs that could be governing nouns in the example sentences, an automatic disambiguation method is necessary. Likewise, an automatic selection of the FE connecting the frames for the noun and the governing verb is necessary, e.g., `Post_state` in Figure 9.

We are also working on creating integration rules to express very complex reification patterns for certain linguistic patterns in the property name. For instance, Figure 10 shows an example expressing amounts for property names satisfying the regular expression `(has )?number of (.*)`, which is relatively common among LOD datasets mined from tables with statistics. The recall of this method can be increased if the canonicalization is also extended to complete these patterns in case parts of them are omitted. For instance, for the example about amounts given above, the prefix "has number of" could be added to those properties whose name is a countable noun or a noun phrase, and whose range is a positive integer (in LOD datasets typically implemented as literals with datatypes `xsd:nonNegativeInteger`).

Finally, we are also working on combining all these rules with other types of rules that map entities of the same type (classes with classes, properties with properties), and can be built re-using existing `owl:sameAs` ontology alignment systems. This combination will allow arbitrarily complex mappings, not only between the external datasets and FrameBase, but transitively between the external datasets.

5. Conclusion

In this paper, we have shown the importance of establishing complex mappings between linked open datasets, transcending the space of binary relationships that can be captured using simple links of type `owl:sameAs`, `rdfs:subClassOf`, or `rdfs:subPropertyOf`. We have shown schema-level methods to create these complex mappings, using a star-based topology with a wide schema as a central hub, and exploiting its connections to computational linguistics. As part of this process, we have also provided heuristics to extend, disambiguate, and canonicalize the names of properties in the source datasets. We have evaluated our approach on DBpedia, finding that it yields encouraging results across different domains. Finally, we have outlined future work to create even more integration rules involving complex patterns.

Acknowledgments

The research leading to these results has received funding from National Basic Research Program of China Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 61550110504, as well as by the Danish Council for Independent Research (DFF) under grant agreement No. DFF-4093-00301.

6. References

[1] Collin F Baker, Charles J Fillmore, and John B Lowe. The Berkeley FrameNet Project. In *ICCL '98*, pages 86–90, 1998.

[2] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia-A crystallization point for the Web of Data. *J. Web Sem.*, 7(3):154–165, 2009.

[3] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The Alignment API 4.0. *Semantic Web*, 2(1):3–10, 2011.

[4] Gerard de Melo. Not Quite the Same: Identity Constraints for the Web of Linked Data. In *AAAI'13*, 2013.

[5] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos. iMAP: Discovering Complex Semantic Matches Between Database Schemas. In *SIGMOD'04*, pages 383–394, 2004.

[6] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[7] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M. Suchanek. Canonicalizing Open Knowledge Bases. In *CIKM'14*, pages 1679–1688, 2014.

[8] Harry Halpin and Patrick J. Hayes. When owl: sameAs isn't the Same: An Analysis of Identity Links on the Semantic Web. In *LDOW'10*, 2010.

[9] Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *ACL'03*, pages 423–430, 2003.

[10] Dominique Ritze, Christian Meilicke, Ondrej Sváb-Zamazal, and Heiner Stuckenschmidt. A Pattern-based Ontology Matching Approach for Detecting Complex Correspondences. In *OM'09*, 2009.

[11] Jacobo Rouces, Gerard de Melo, and Katja Hose. FrameBase: Representing N-ary Relations Using Semantic Frames. In *ESWC'15*, 2015.

[12] Jacobo Rouces, Gerard de Melo, and Katja Hose. Representing Specialized Events with FrameBase. DeRiVE '15, 2015.

[13] Jacobo Rouces, Gerard de Melo, and Katja Hose. A Frame-Based Approach for Connecting Heterogeneous Knowledge. 2016. Submitted.

[14] François Scharffe and Dieter Fensel. Correspondence patterns for ontology alignment. In *EKAW'08*, pages 83–92, 2008.