Aligning Ontologies of Geospatial Linked Data

Rahul Parundekar, Craig A. Knoblock and José Luis Ambite

Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{*parundek,knoblock,ambite*}*@isi.edu*

Abstract. The Web of Linked Data is characterized by linking structured data from different sources using equivalence statements, such as *owl:sameAs*, as well as other types of linked properties. However, the ontologies behind these sources are not linked. The work described in this paper aims at providing alignments between these ontologies by using an extensional approach. We look at geospatial data sources on the Web of Linked Data and provide equivalence and subsumption relationships between the classes from these ontologies after building hypotheses that are supported by *owl:sameAs* links. We are able to model one geospatial source in terms of the other by aligning the two ontologies and thus understand the semantic relationships between the two sources.

1 Introduction

The last few years have witnessed the beginnings of a paradigm shift from publishing isolated data from various organizations and companies to publishing data that is *linked* to related data from other sources, using the structured model of the Semantic Web. As the data being published on the Web of Linked Data starts to grow, the availability of related data can be used to supplement one's own knowledge base. This provides a huge potential in various domains, like the geospatial domain, where it is used in the integration of data from different sources. Apart from generating structured data, a necessary step to publish data in the Web of Linked Data is to provide links from the generated instances to other data 'out there,' based on background knowledge (e.g. linking DBpedia to Wikipedia), common identifiers (e.g. ISBN numbers), or pattern matching (e.g. names, latitude, longitude and other information used to link Geonames to DBpedia). These links are often expressed by using *owl:sameAs*. A common occurrence when such links between instances are asserted is the missing link between their corresponding concepts. Such a link would ideally be able to help a consumer of the information (agent/human) to model data from the other source in terms of their own knowledge. This is widely known as ontology alignment, a special form of schema alignment. Schema alignment has been extensively studied in domains like schema integration, data warehouses, E-commerce, semantic query processing, etc. [1]. There is also a considerable amount of work in ontology matching which is summarized in Euzenat et al. [2]. The advent of the Web of Linked Data warrants a renewed inspection of these methods. Our approach provides alignments between classes from two ontologies in the Web of Linked Data by looking at the instances which are linked as the same. We

believe that providing an alignment between the ontologies of the two sources on the Web of Linked Data provides valuable knowledge in understanding and modeling the semantic relationships among sources.

The paper is organized as follows. We first provide background on geospatial Linked Data and describe the sources that are the subject of this paper. We then describe our approach to ontology alignment, where alignments are conjunctions of restriction classes. We follow with an empirical evaluation on three large geospatial Linked Data sources, namely GEONAMES, DBPEDIA, and LINKEDGEODATA. Finally, we briefly describe related and future work and discuss the contributions of this paper.

2 Background on Linked Geospatial Data

In this section, we provide a brief introduction to the popular use of Linked Data and the data sources from the geospatial domain that we consider.

2.1 Nature of Linked Data

The Linked Data movement, as proposed by Berners-Lee [3], aims to provide machine-readable connections between data in the Web. Bizer et al. [4] describes several approaches to publishing such Linked Data. Most of the Linked Data is generated automatically by converting existing structured data sources (typically relational databases) into RDF, using an ontology that closely matches the original data source. For example, GEONAMES gathers its data from around 40 different sources and it primarily exposes its data in the form of a flat-file structure,[1] which is described with a simple ontology [5]. Such an ontology might have been different if designed at the same time as the collection of the actual data. For example, all instances of GEONAMES have the same *rdf:type* of *geonames:Feature*, however they could have been more effectively typed based on the *featureClass* & *featureCode* properties.

The links in the Linked Data on the Web make the Semantic Web browsable and, moreover, increases the amount of knowledge by complementing data already presented by a resource. A popular way of linking data on the Web is the use of *owl:sameAs* links to represent *identity links* [6]. Instead of reusing existing URIs, new URIs are automatically generated while publishing linked data and an *owl:sameAs* link is used to say that two URI references refer to the same thing. Both Ding et al. [7] and Halpin et al. [6] discuss the popular usage of the *owl:sameAs* property. Halpin et al. [6] summarizes its usages as belonging to one of the four types *i)* same thing as but different context *ii)* same thing as but referentially opaque *iii)* represents and *iv)* very similar to. We however refrain ourselves from going into the specifics and use the term as asserting *same thing*. For example, GEONAMES uses the *owl:sameAs* link to represent an alias or a co-reference.

2.2 Linked geospatial Data Sources

We consider three linked data sources GEONAMES[2], DBPEDIA[3] and LINKEDGEODATA[4].

[1] http://download.geonames.org/export/dump/readme.txt
[2] http://www.geonames.org/ontology/
[3] http://wiki.dbpedia.org/About
[4] http://linkedgeodata.org/About

GEONAMES is a geographical database that contains over 8 million geographical names consisting of 7 million unique features including 2.6 million populated places and 2.8 million alternate names. This database is downloadable free of charge and accessible in various forms like a flat table, web services, etc. We use the Linked Data representation of the database available as an RDF dump containing 6,520,110 features and 93,896,732 triples. The structure behind the data is the Geonames ontology [5], which closely resembles the flat-file structure. A typical individual in the database is an instance of type *Feature* and has a *Feature Class* associated with it. These *Feature Class*es can be administrative divisions, populated places, structures, mountains, water bodies, etc. Though the *Feature Class* is subcategorized into 645 different *Feature Code*s, the *Feature Code* is associated with a *Feature* instance and not as a specialization of the property *featureClass* (this is probably due to automatically exporting of existing relational data into RDF rather than building data conforming to an ontology). A *Feature* also has several other properties, such latitude, longitude, and an *owl:sameAs* property linking it to an instance from DBPEDIA.

DBPEDIA is a source of structured information extracted from WIKIPEDIA containing about 1.5 million objects that are classified with a consistent ontology. Because of the vastness and diversity of the data in DBPEDIA, it presents itself as a hub for links in the Web of Linked Data from other sources [8]. The dataset includes individuals not limited to geospatial data such as 312,000 persons, 94,000 music albums, etc. We are interested in a subset of this data which GEONAMES links itself with (e.g. places, mountains, etc.). As DBPEDIA contains a large variety of data (e.g. abstracts, links to other articles, images, etc.) we limit our approach to RDF containing the *rdf:type* assertion and info boxes, which provide factual information.

LINKEDGEODATA is a geospatial source with its data imported from the Open Street Map (OSM) [9] project containing about 2 billion triples comprising approximately 350 million *nodes* and 30 million *ways*. In order to generate the RDF, the data extracted from the OSM project was linked to DBPEDIA instances using the user created links on OSM to WIKIPEDIA articles. These links were then used as the training set on which machine learning algorithms were applied with a heuristic on a combination of the LINKEDGEODATA type information, spatial distance, and name similarity [10]. As a result the set of *owl:sameAs* links was then expanded to include more instance matchings based on the model learnt. This data is provided by LINKEDGEODATA as an RDF dump.

We loaded these three RDF datasets into a local database. In this paper we only consider instances from DBPEDIA and GEONAMES that are linked using the *owl:sameAs* property. Similarly, we focus on instances from LINKEDGEODATA linked by *owl:sameAs* to DBPEDIA instances. In order to reduce the number of self joins required on a source table in the database, the values related to each individual were converted into a single row identified with its URI. This forms a table where the columns represent all of the properties (predicates) that occur in the triples of that source. For example, the table for GEONAMES subset contains 11 columns not including the identifier. The values for the properties (object part of a triple) go into the value of the column for the row identified by URI of the individual (subject part of a triple). We call this a *vector*. In cases of multivalued properties, the row is replicated in such a way that each cell contains a single value but the number of rows equals the number of multiple values. Each new

row however, is still identified with the same URI, thus retaining the number of distinct individuals. In general, the total number of rows for each individual is the product of cardinalities of the value sets for each of its properties. Throughout this paper we describe our methodology using the GEONAMES-DBPEDIA dataset, but we include the findings of the LINKEDGEODATA-DBPEDIA source in the results section.

3 Ontology Alignment Using Linked Data

3.1 Ontology Alignment

An *Ontology Alignment* is "a set of correspondences between two or more ontologies" where a *correspondence* is "the relation holding, or supposed to hold according to a particular matching algorithm or individual, between entities of different ontologies."[2] *Entities* here, can be classes, individuals, properties or formulas. Our algorithm relies on data analysis and statistical techniques for matching the two ontologies, which Euzenat et al. [2] classify as a *common extension comparison* approach for aligning the structure. This approach considers two sets *A* and *B* of instances belonging to the two ontologies out of which some instances are common. Set containment relationships between the set of instances *A* and *B* suggest the alignment between the two as follows: *A* equals *B* ($A \cap B = A = B$), *A* contains *B* ($A \cap B = B$), *A* is contained in *B* ($A \cap B = A$), *A* is disjoint from *B* ($A \cap B = \emptyset$) and *A* overlaps with *B* ($A \cap B \neq \emptyset$). Euzenat et al. [2] also argues that using a Hamming distance is more robust than using simple set containment as it tolerates misclassified individuals and still produces acceptable results. We identify common instances between the two ontologies required for this technique using the *owl:sameAs* links, where the instance identifier in each ontology gets replaced with a combination of the URIs from both ontologies. Instead of limiting ourselves to existing classes, we extend the scope of our alignments by using *restriction classes* as explained in the following section. In its stricter sense, an alignment between classes would also consider structural conformity like cardinality constraints on properties, class hierarchies, domain and ranges of properties, etc. We however do not consider this in order to provide simpler alignments as most of these ontologies are rudimentary and based on pre-existing relational structures.

3.2 Restriction Classes

In the alignment process, instead of focusing only on classes defined by *rdf:type*, we also consider classes defined by conjunctions of property value restrictions (i.e, *hasValue* constraints in the Web Ontology Language), which we will call *restriction class*es in the rest of the paper. For example, each instance from GEONAMES has its *rdf:type* as *Feature*. Traditional alignments would then only be between the class *Feature* from GEONAMES and another class from DBPEDIA, for example *Place*. As mentioned before, instances in GEONAMES are also categorized by the *featureCode* and *featureClass* properties. A restriction on values of such properties gives us classes that we can effectively align with classes from DBPEDIA. For example, the *restriction class* formed

from the concept *Feature* with its value of *featureCode* property constrained to the instance 'geonames:A.PCLI' (Independent political entity) aligns with the *Country* concept from DBPEDIA. Our algorithm defines restriction classes from the source ontologies and generates alignments between such restrictions clases using subset or equivalence relationships.

Defining the Space of Restriction Classes for a Vector The space of *restriction classes* to which an instace belongs is simply the powerset of the property-value pairs of the vector representing the instance. For example assume that the GEONAMES source has only three properties: *rdf:type*, *featureCode* and *featureClass*; and that the vector for the instance *Saudi Arabia* is (geonames:Feature, geonames:A.PCLI, geonames:A). Then this instance belongs to the *restriction class* defined by (*rdf:type*=geonames:Feature & *featureClass*=geonames:A). The other sets from the powerset also form such *restriction classes* as shown in Figure 1.

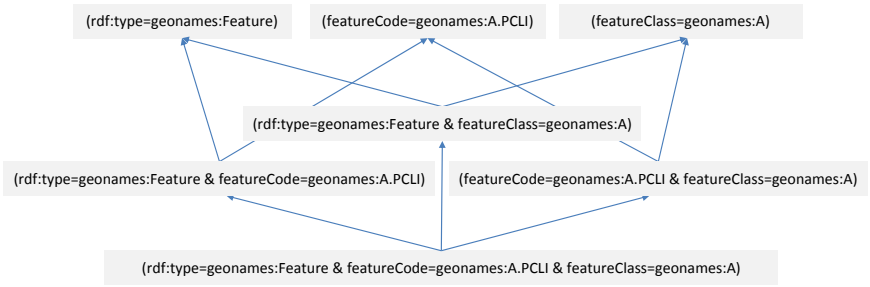

Fig. 1. Hierarchy showing how restriction classes are built

We employ a bottom-up approach to determining the *restriction classes* to which an instance belongs. For example, the instance *vector* for *Saudi Arabia* directly supports the *restriction class* (*rdf:type*=geonames:Feature & *featureCode*=geonames:A.PCLI & *featureClass*=geonames:A). This *restriction* in turn supports the three *restrictions* shown in Figure 1. These subsequently support *restrictions* formed by eliminating a constraint on one of the properties in it. In general, if for the *n* number of properties for the *vector* at the current level we select m properties to constrain our *restrictions*, then that level would contain $\binom{n}{m}$ *restrictions* that represent elements from the combinatorial set *C (n,m)* from the set *S* (see Section 3.2). These $\binom{n}{m}$ *restrictions* would in turn support *restriction sets* formed by choosing *(m-1)* properties from *S* (i.e. *C (n,m-1)*). For a *restriction* at level *l* we call all the *restrictions* at level *(l-1)* that it supports as its *parent restrictions*. We can see that, if a *restriction* is supported by some individuals, then those individuals also support any of its *parent restrictions*. Using this hierarchical approach for any *vector*, we can build a set of *restriction classes* that it supports (denoted by R), in a bottum-up fashion, starting with a *restriction class* corresponding to the *vector*. It is evident that in order to consider all *restriction classes*, the algorithm would be exponential. We thus need some preprocessing that eliminates those columns that are not useful.

3.3 Preprocessing of the data

In order to reduce the search space of alignment hypotheses, we identify properties that do not contribute to the alignment. Inverse functional properties resemble secondary keys in databases and identify an instance uniquely. Thus, if a *restriction class* is constrained on the value of an inverse functional property, it would only have a single element in it. As an example, consider the *wikipediaArticle* property in GEONAMES. An instance in GEONAMES has links to multiple pages in *Wikipedia*, each in a different language. For example, the instance for the country Saudi Arabia[5] has links to 237 different articles. Each of these in turn, however, could be used to identify only *Saudi Arabia*. On the other hand, each of the seven million unique features is also classified into nine different *Feature Class*es. Thus, *featureCode* and *featureClass* properties are not inverse functional properties and instances can be grouped by *restrictions* with constraints on these properties. We remove a column from the table of *vectors* if its corresponding property is inverse functional. In some of the cases (like the *alternateName* and *wikipediaArticle*) the multivalued properties get eliminated and thus the number of rows in the flat table gets reduced.

From these individual vectors, we then perform a join on the *owl:sameAs* property from GEONAMES such that we get a combination of properties from both ontologies. We call this concatenation of two *vectors* an *instance pair*. In case of multiple vectors for the same URI (as a result of multivalued properties), we get *instance pair*s whose count is equal to the product of the number of vectors from each of the ontologies.

3.4 Forming Alignment Hypotheses

We build our *alignment hypothesis* in a bottom-up fashion by examining each *instance pair* and the *restriction class*es that its corresponding *vector* supports. Each *alignment hypothesis* is a two part conjunction of a *restriction class* from the first ontology (GEONAMES) and a *restriction class* from the second one (DBPEDIA). In order to come up with these hypotheses we look at *vectors* that compose each of the *instance pairs*. If we build a set of *restrictions* R_1 from the first vector and R_2 from the second vector, then the set of hypotheses that this *instance pair* supports is the Cartesian product of R_1 and R_2. We then would aggregate such hypotheses over all *instance pairs* and eliminate the hypotheses that contain only a singleton set of *instance pairs* supporting it. As a result we would be left with alignments supported by multiple instances between the two sources. In practice however this brute force method is very inefficient in space and time considerations. Hence we use an algorithm that capitalizes on the set containment property of the *restrictions* and thus finds alignments inspired from the bottum-up method described in Section 3.2.

For building hypotheses of alignments, we begin with each *instance pair*. We assert a seed hypothesis such that if $vector_1$ is the first part (from GEONAMES) of this *pair* and $vector_2$ is the second part (from DBPEDIA), then the *restriction class* corresponding to $vector_1$ (built of constraints on each of its property-value pairs) has an alignment with the *restriction class* corresponding to $vector_2$. Let r_1 & r_2 be the *restriction classes* constituting this hypothesis. We compute P_1 & P_2 as the set of *parent restrictions* of

[5] http://sws.geonames.org/102358/about.rdf

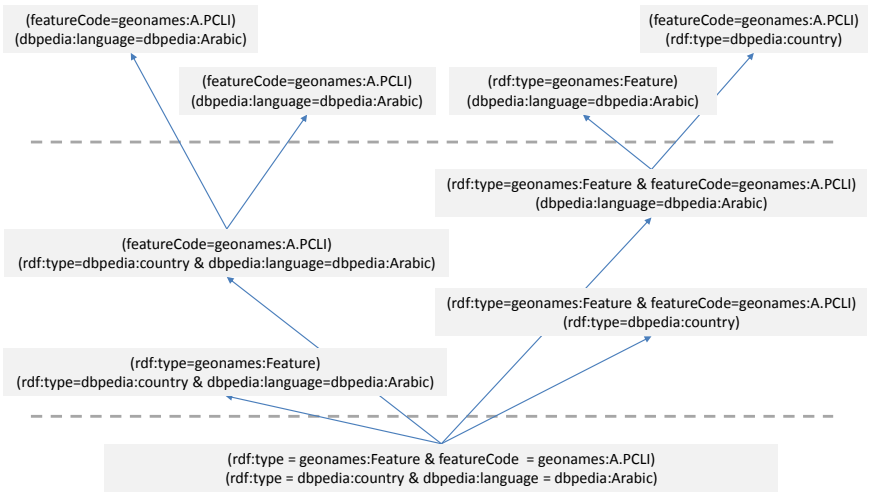

Fig. 2. Hierarchy showing how hypotheses are built

r_1 & r_2. We now build a set of *parent hypotheses* from the seed source that contains an alignment from each *restriction* p_1 from P_1 to r_2 and vice versa. We keep a track of all our hypotheses and all the *instances pairs* that support them. If an *instance pair* supports one hypothesis, then it also supports all of its *parent hypotheses*. We can say this, because the *restrictions* constituting our hypothesis support their own *parent restrictions*, which in turn constitute our *parent hypotheses*. This process is illustrated in Figure 2.

3.5 Evaluating Alignment Hypotheses

After building the hypotheses, we score each hypothesis to ascertain the degree of confidence with which we hold this alignment to be true. For each hypothesis, we refer to the *restriction classes* constituting it. We then find all instances belonging to the *restriction class* r_1 from the first source and r_2 from the second source. Figure 3 shows the sets of such instances. We then compute the *image* of r_1 (denoted by $I(r_1)$), which is the set of instances from the second source that form *instance pairs* with instances in r_1, by following the *owl:sameAs* links. The dashed lines in the figure represent these *instance pairs*. All the pairs that match both restrictions r_1 and r_2 also support our hypothesis and thus is equivalent to the *instance pairs* corresponding to instances belonging to the intersection of the sets r_2 and $I(r_1)$. This set of *instance pairs* that support our hypothesis is depicted as the shaded region. We can now capture subset and equivalence relations between the *restriction classes* by set-containment relations from the figure. For example, if the set of *instance pairs* identified by r_2 are a subset of $I(r_1)$, then the set r_2 and the shaded region would be entirely contained in the $I(r_1)$. We use two metrics *P* and *R* to quantify these set-containment relations. Figure 4 summarizes these metrics and also the different cases of intersection. In order to allow a certain margin

of error induced by the dataset, we are lenient on the constraints and use the relaxed versions *P'* and *R'* as part of our scoring mechanism.

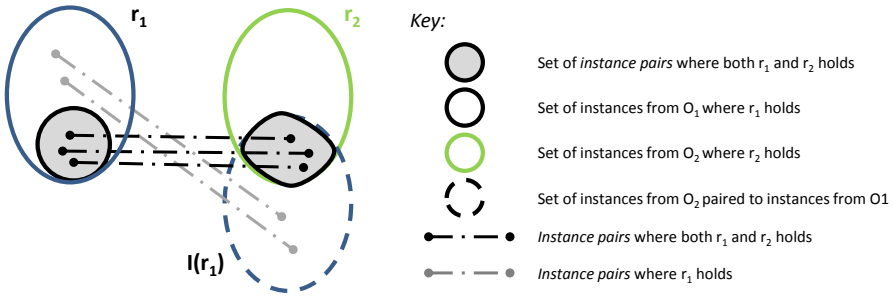

Fig. 3. Scoring of a hypothesis

Set Representation	Relation	P = $\frac{\lvert I(r_1) \cap r_2 \rvert}{\lvert r_2 \rvert}$	R = $\frac{\lvert I(r_1) \cap r_2 \rvert}{\lvert r_1 \rvert}$	P'	R'
	Disjoint	= 0	= 0	≤ 0.01	≤ 0.01
	$r_1 \subset r_2$	< 1	= 1	> 0.01	≥ 0.90
	$r_2 \subset r_1$	= 1	< 1	≥ 0.90	> 0.01
	$r_1 = r_2$	= 1	= 1	≥ 0.90	≥ 0.90
	Not enough support	0 < P < 1	0 < R < 1	0.01 < P' < 0.90	0.01 < R' < 0.90

Fig. 4. Metrics

While calculating results, we find equivalences and subsets as follows. Consider a hypothesis from the *restriction class* 'featureClass=geonames:H' from GEONAMES to *restriction class* 'rdf:type=dbpedia:BodyOfWater' from DBPEDIA. Based on the support for the hypothesis we have $|I(r_1)|$ = 2132, $|r_2|$ = 1989 and $|I(r_1) \cap r_2|$ = 1959. Thus, *P'* = 0.98 & *R'* = 0.92 and we say that it is an equivalence alignment as both *P'* and *R'* are greater than a threshold of 0.9 as explained in Figure 4. Similarly, we also identify alignments like the *restriction class* 'featureCode=geonames:P.PPL' $\subset$ *restriction class* 'rdf:type=geonames:Place' and others as shown in Table 1.

GEONAMES *restriction*	DBPEDIA *restriction*	*P'*	*R'*	$\|I(r_1) \cap r_2\|$	Relation
geonames:featureClass=H	rdf:type=BodyOfWater	0.92	0.98	1959	$r_1 = r_2$
geonames:featureCode=S.AIRP	rdf:type=Airport	0.99	0.99	2042	$r_1 = r_2$
geonames:featureCode=S.BDG	rdf:type=Bridge	0.92	0.95	144	$r_1 = r_2$
geonames:featureCode=S.SCH	rdf:type=EducationalInstitution	0.95	0.92	380	$r_1 = r_2$
geonames:featureCode=S.SCH & geonames:inCountry=US	rdf:type=EducationalInstitution	0.95	0.92	377	$r_1 = r_2$
rdf:type=Feature	rdf:type=Thing	0.99	0.99	71003	$r_1 = r_2$
rdf:type=Feature	rdf:type=Place	0.98	0.99	69947	$r_1 = r_2$
geonames:featureClass=P	rdf:type=PopulatedPlace	0.97	0.91	54927	$r_1 = r_2$
geonames:featureCode=P.PPL	rdf:type=Place	0.81	0.99	56751	$r_1 \subset r_2$
geonames:featureCode=H.LK	rdf:type=Lake	0.97	0.76	1452	$r_1 \subset r_2$
geonames:featureCode=T.MT	rdf:type=Mountain	0.97	0.78	1728	$r_1 \subset r_2$
geonames:featureCode=A.PCLI	rdf:type=Country	0.98	0.68	184	$r_1 \subset r_2$
geonames:featureCode=P.PPL	rdf:type=PopulatedPlace	0.96	0.87	53469	$r_1 \subset r_2$
rdf:type=Feature	rdf:type=PopulatedPlace	0.99	0.84	60102	$r_2 \subset r_1$
geonames:featureCode=S.HSP	rdf:type=Hospital	0.82	1.00	23	$r_2 \subset r_1$
LINKEDGEODATA *restriction*	DBPEDIA *restriction*	*P'*	*R'*	$\|I(r_1) \cap r_2\|$	Relation
rdf:type=lgd:lighthouse	rdf:type=Lighthouse	100	100	7	$r_1 = r_2$
rdf:type=lgd:station	rdf:type=Station	100	100	42	$r_1 = r_2$
rdf:type=lgd:country	rdf:type=Country	100	98.58	139	$r_1 = r_2$
rdf:type=lgd:node	rdf:type=Thing	99.99	97.27	22987	$r_1 = r_2$
rdf:type=lgd:node	rdf:type=Place	99.5	97.38	22875	$r_1 = r_2$
rdf:type=lgd:node	rdf:type=PopulatedPlace	92.77	99.75	21328	$r_1 = r_2$
rdf:type=lgd:university	rdf:type=University	97.14	94.44	34	$r_1 = r_2$
rdf:type=lgd:aerodrome	rdf:type=Airport	100	90.94	251	$r_1 = r_2$
rdf:type=lgd:island	rdf:type=Island	99.44	90.81	178	$r_1 = r_2$
rdf:type=lgd:stadium	rdf:type=Stadium	100	81.81	54	$r_1 \subset r_2$
rdf:type=lgd:River	rdf:type=River	94.73	78.26	18	$r_1 \subset r_2$
rdf:type=lgd:way	rdf:type=BodyOfWater	74.53	94.3	480	$r_2 \subset r_1$
rdf:type=lgd:way	rdf:type=Lake	71.11	94.23	458	$r_2 \subset r_1$

Table 1. Example alignments from the GEONAMES-DBPEDIA and LINKEDGEODATA-DBPEDIA datasets

4 Results

We generated alignments using the above described method on a subset of GEONAMES and DBPEDIA that contains only instances that are linked together. After having generated alignments on a subset of the data, we found that most of the properties left after preprocessing and removing the inverse-functional properties in DBPEDIA did not have enough support in the database. Moreover, as the classes in the DBPEDIA ontology are well-defined and highly specialized, we focused only on the *rdf:type* property on the DBPEDIA side for the results. Our approach generates 20116 hypotheses that cannot be specialized further and have a support of more than one instance pair. From this we get 8 equivalences, 2937 $r_1 \subset r_2$ relations and 60 $r_2 \subset r_1$ relations. Table 1 shows some examples of the equivalence classes that our algorithm finds. Note that as we consider only the subset of GEONAMES and DBPEDIA consisting of instances linked

via *owl:sameAs* links, our algorithm classifies as equivalence alignments like the one between 'rdf:type=geonames:Feature' and 'rdf:type=dbpedia:Place', which with more linked data would probably not hold.

We also ran our algorithm on the LINKEDGEODATA-DBPEDIA subset. As these ontologies contain classes that are highly specialized, we chose to restrict the alignments only between the *rdf:type*s of the sources. We generate 176 hypotheses that cannot be specialized further and have a support of more than one instance pair. From this we get 9 equivalences, 66 $r_1 \subset r_2$ relations and 14 $r_2 \subset r_1$ relations. These results are summarized in Table 1.

It should be noted that these alignments closely follow the semantics behind the sources. For example, looking at the results we would assume that the alignment of 'geonames:featureCode=T.MT' (Mountain) with 'rdf:type=dbpedia:Mountain' would be equivalent. Closer inspection of the GEONAMES dataset shows, however, that there are some places with *Feature Codes* like 'T.PK' (Peak), 'T.HLL' (Hill), etc. from GEONAMES whose corresponding instances in DBPEDIA are all typed 'dbpedia:Mountain'. This implies that the interpretation of the concept 'Mountain' is different in both the sources and only a subsumption relation holds.

5 Related Work

There is large previous literature on ontology matching [2]. Ontology matching has been based on *terminological* (e.g. linguistic and information retrieval techniques [11]), *structural* (e.g. graph matching[12]) and *semantic* (e.g. model based) approaches or their combination. The FCA-merge algorithm [13] uses extension techniques over common instances between two ontologies to generate a concept lattice in order to merge them, and thus align them indirectly. This algorithm, however, relies on a domain expert (users) to generate the merged ontology and is based on a single corpus of documents instead of two different sources, unlike our approach. Perhaps a strong parallel to our work is found in Duckham et al. [14] which also uses an extensional approach for fusion and alignment of ontologies in geospatial domain. The difference in our approach in comparison to their work (apart from the fact that it predates Linked Data) is that while their method fuses ontologies and aligns only existing classes, our approach is able to generate alignments between classes that are derived from the existing ontology by imposing restrictions on values of any or all of the properties not limited to the class *type*. Most of the work in information integration within the Web of Linked Data is in instance matching as explained in Bizer et al.[4]. Raimond et al. [15] use string and graph matching techniques to interlink artists, records, and tracks in two online music datasets (Jamendo and MusicBrainz) and also between personal music collections and the MusicBrainz dataset. Our approach solves a complimentary piece of the information integration problem on the Web of Linked Data by aligning ontologies of linked data sources.

6 Conclusion

Linked Data on the Web contains linked instances from multiple sources without the ontologies of the sources being themselves linked. However, it is useful to the consumers

of the data to define the alignments between such ontologies. Our algorithm generates alignments, consisting of conjunctions of *restriction classes*, that define subsumption and equivalence relations between the ontologies. This paper focused on automatically finding alignments between the ontologies of geospatial data sources. However, the technique is general and can be applied to other Web of Linked Data data sources.

In our future work, we plan to improve the scalability of our approach, specifically, improve the performance of the algorithm that generates alignment hypotheses by using a more heuristic exploration of the space of alignments. We also plan to apply our alignment techniques to other sources in the Web of Linked Data beyond the geospatial domain, such as the richly interlinked genetic data published by Bio2RDF [16].

References

1. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal **10**(4) (2001) 334–350
2. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag (2007)
3. Berners-Lee, T.: Design Issues: Linked Data. http://www.w3.org/DesignIssues/LinkedData.html (2009)
4. Bizer, C., Cyganiak, R., Heath, T.: How to publish linked data on the web. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/ (2007)
5. Vatant, B., Wick, M.: Geonames ontology. http://www.geonames.org/ontology/
6. Halpin, H., Hayes, P.J.: When owl:sameAs isn't the same: An analysis of identity links on the semantic web. In: International Workshop on Linked Data on the Web, Raleigh, North Carolina (2010)
7. Ding, L., Shinavier, J., Finin, T., McGuinness, D.L.: owl:sameAs and Linked Data: An Empirical Study. In: Second Web Science Conference, Raleigh, North Carolina (2010)
8. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: Sixth International Semantic Web Conference, Busan, Korea (2007) 11–15
9. Haklay, M.M., Weber, P.: OpenStreetMap: user-generated street maps
10. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: Adding a Spatial Dimension to the Web of Data. In: Eight International Semantic Web Conference, Washington, DC (2009) 731–746
11. Euzenat, J.: An API for Ontology Alignment. In: Third International Semantic Web Conference, Hiroshima, Japan (2004) 698–712
12. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: International Conference on Data Engineering, San Jose, California (2002) 117–128
13. Stumme, G., Maedche, A.: FCA-Merge: Bottom-up merging of ontologies. In: International Joint Conference on Artificial Intelligence, Seattle, Washington (2001) 225–234
14. Duckham, M., Worboys, M.: An algebraic approach to automated geospatial information fusion. International Journal of Geographical Information Science **19**(5) (2005) 537–557
15. Raimond, Y., Sutton, C., Sandler, M.: Automatic interlinking of music datasets on the semantic web. In: First Workshop on Linked Data on the Web, Beijing, China (2008)
16. Belleau, F., Tourigny, N., Good, B., Morissette, J. In: Bio2RDF: A Semantic Web atlas of post genomic knowledge about human and mouse. Springer (2008) 153–160