A Statistical and Schema Independent Approach to Identify Equivalent Properties on Linked Data

Kalpa Gunaratna
Kno.e.sis Center
Wright State University
Dayton OH, USA
kalpa@knoesis.org

Krishnaprasad Thirunarayan
Kno.e.sis Center
Wright State University
Dayton OH, USA
tkprasad@knoesis.org

Prateek Jain
IBM T J Watson Research Center
Yorktown Heights NY, USA
jainpr@us.ibm.com

Amit Sheth
Kno.e.sis Center
Wright State University
Dayton OH, USA
amit@knoesis.org

Sanjaya Wijeratne
Kno.e.sis Center
Wright State University
Dayton OH, USA
sanjaya@knoesis.org

ABSTRACT

Linked Open Data (LOD) cloud has gained significant attention in the Semantic Web community recently. Currently it consists of approximately 295 interlinked datasets with over 50 billion triples including 500 million links, and continues to expand in size. This vast source of structured information has the potential to have a significant impact on knowledge-based applications. However, a key impediment to the use of LOD cloud is limited support for data integration tasks over concepts, instances, and properties. Efforts to address this limitation over properties have focused on matching data-type properties across datasets; however, matching of object-type properties has not received similar attention. We present an approach that can automatically match object-type properties across linked datasets, primarily exploiting and bootstrapping from entity co-reference links such as *owl:sameAs*. Our evaluation, using sample instance sets taken from Freebase, DBpedia, LinkedMDB, and DBLP datasets covering multiple domains shows that our approach matches properties with high precision and recall (on average, F measure gain of 57% - 78%).

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Linked Open Data, Property Alignment, Relationship Identification, Statistical Equivalence

I-SEMANTICS 2013, 9th Int. Conf. on Semantic Systems, Sept. 4-6, 2013, Graz, Austria

1. INTRODUCTION

About 6 years ago, Sir Tim Berners-Lee introduced the idea of openly publishing RDF based datasets. This idea was based on four simple rules as described in [2]. The ultimate objective was to promote interlinking between datasets and lay the foundation for "Web of Data". He advocated the use of unique URIs to identify and distinguish things on the Web. Use of different namespaces and URIs frees the dataset publishers from worrying about name conflicts with existing resources of the same kind on the LOD. This step was significant from the point of view of encouraging RDF publication as, in the span of 6 years, we have approximately 295 datasets interlinked with each other on the LOD cloud[1]. These datasets cover numerous domains such as entertainment, life sciences, and governmental legislations. Furthermore, this interlinked collection of diverse and structured datasets has the potential to be used and exploited for numerous tasks such as enhanced browsing and search. Search using LOD datasets can be very significant in practice as it can provide structured representation of entities and important information associated with them returned as results. Thus, a search for term "Tim Berners-Lee" will not only return Web pages of him, but also aggregate information from his FOAF and DBpedia profiles.

Some of the existing LOD search engines such as Sig.ma [15] provide these capabilities by integrating data from different data sources. These search engines utilize the *owl:sameAs* relationships created by tools like SILK [16] and LIMES [8]. Such search engines are an important step towards presenting and using information on LOD.

However, there are certain areas where these systems can be improved. For example, a quick glance at the search results on Sig.ma points to redundant information about entities. For "Tim Berners-Lee", the two properties named *birth place* and *born in* list similar values[2]. Here, the *birth place* property comes from DBpedia and *born in* comes from Yago. These two property names can be aligned to present an unified view under one property name or used to improve coverage and organization. Therefore, property align-

[1] http://linkeddata.org
[2] As of 23-03-2013

ment is imperative for data integration and consuming tasks over LOD. Furthermore, properties capture the meaning of RDF triples and understanding the interconnection between them is considered to be important in the Semantic Web context[12].

In this work, we present an approach to align properties between two different linked datasets. Our approach relies on utilizing the entity co-reference relationships (*ECR*) such as those formalized using *owl:sameAs* and *skos:exactMatch*. Our approach analyzes occurrences of equivalent subject and object values across datasets to align properties. E.g., given two matching subject and object pairs that are connected by *ECR* links, we check whether the associated property names have the potential to be equivalent. This is done in a robust manner by analyzing the aggregated statistical results related to matching subject-object pairs for a given pair of properties. Using these results, we show how existing entity co-reference links between resources in LOD cloud can be used to align properties. The main contributions of the paper are as follows:

- It introduces an efficient approach that utilizes property extensions and resource inter-links for property alignment.
- It uses the notion of *Statistical Equivalence* to approximate *owl:equivalentProperty*.

The rest of the paper is organized as follows. In Section 2, we discuss the state of the art techniques used in various property alignment tasks. Section 3 elaborates on the problem and the proposed algorithm. Section 4 presents the results obtained and shows that *Statistically Equivalent* matches (defined later) outnumber coincidental matches as the algorithm is run on a large number of instances. Section 5 discusses interesting matching patterns and facts relevant to the matching process, while Section 6 concludes with our findings and future work.

2. BACKGROUND

Property alignment, which is crucial for data integration tasks, has not been addressed adequately compared to concept and instance alignment. In contrast with instance and concept alignment, properties exhibit complex structure and meaning. Current techniques used for property alignment (including object-type) fall into three categories: (i) Syntactic/ dictionary-based, where predicate similarity is garnered via string matching or using WordNet, (ii) Schema dependent, and (iii) Schema independent, where instance level information is utilized. Some use a mix of these techniques.

There are a few efforts for matching data-type properties in ontologies [10][14]. Nunes et al. [10] utilized mutual information present at the instance level using genetic algorithms. They discuss matching of one to many complex relationships in different ontologies, but are limited to data-type properties. [14] discusses a cluster based similarity aggregation methodology for ontology matching where they focus on four different similarity measures to align properties. They calculate string, WordNet, profile, and instance similarities based on the domains and ranges of the properties. Even though [14] tries to match object-type properties, they mention that the results are not strong enough to distinguish matching and non-matching property names. [13] incorporates a density estimation approach using Kernel Density Estimation (KDE) to map opaque properties (properties conveying the same meaning irrespective of their names) in ontologies. However, they transform values into numeric form to be compatible with KDE. This transformation is not easy in the LOD context where each instance contains many triples.

Zhao et. al [17] presented a graph based ontology analysis complementing their previous work related to building a mid-level ontology utilizing WordNet and string based similarity measures to group properties. The approach is not suitable for identifying equivalent properties since it groups any two related properties (using object value similarity) and does not take into account the effect of coincidental matches in initial grouping. SLINT [9] is an instance matching system that uses an IR based heuristic to calculate object values overlap. Both these approaches are coarse grained and hence not suitable for identifying equivalent properties as they aggregate properties on the overlap (not on the individual subject pairs). E.g., they can confuse conceptually different predicates "placeOfBirth" and "placeOfDeath". These approaches are also different from ours in the sense that they use only object values and their overlap whereas we strictly try to match the property extensions minimizing false positives. TripleRank's effort [4] in faceted browsing computes latent predicate similarity (i.e., similar properties within a dataset) indirectly as a byproduct of SVD and it is hard to verbalize the results in terms of extensions. Furthermore, it does not provide an evaluation on intra/inter dataset property alignment in terms of precision and recall. The analysis of *owl:sameAs* networks and their implications for detecting schema-level inconsistencies and ontology alignment are discussed in [3]. Some of the alignment techniques and applications have used these networks showing their effectiveness in practice [11][17]. We also use a similar link traversal network model in our approach to match property extensions.

3. APPROACH

Property alignment is a non-trivial research problem in the Semantic Web domain whose accomplishment can lead to significant advancements in data integration tasks. The objective of property alignment is to identify equivalent or sub-property relationships between a property pair P_1 and P_2, which may be in the same or different datasets. Since property names in different datasets have independent origin and relationships capture complex meaning in a triple [12], calculating string similarity or synonym based measurements on property names alone does not suffice. To solve this problem, our "extensional" approach determines related properties (P) by finding similar triple patterns across datasets by matching subject (S) and object (O) values in triples of the form (SP_1O) and (SP_2O).

3.1 Property alignment between datasets

OWL [1] defines the concept of equivalent property (*owl: equivalentProperty*) as two properties having the same extension. For example, if property P is defined by triples $\{\ a\ P\ b,\ c\ P\ d,\ e\ P\ f\ \}$ and property Q is defined by triples $\{\ a\ Q\ b,\ c\ Q\ d,\ e\ Q\ f\ \}$, then they are equivalent properties because they have the same extension $\{\{a, b\}, \{c, d\}, \{e, f\}\}$. Since it is hard to expect exactly the same property extensions in real datasets, we approximate it by a signifi-

cant overlap in matching subject-object pairs. For this purpose, we define *statistical equivalence* of properties on linked datasets. For example, if property P is defined by triples $\{ a\ P\ b,\ c\ P\ d,\ e\ P\ f \}$ and property Q is defined by triples $\{ a\ Q\ b,\ c\ Q\ d,\ g\ Q\ h \}$, then property extensions are not the same, but P and Q have matching subject and object values two times out of three providing statistical evidence in support of equivalence. When we utilize evidence for extension matching, we need to overcome the potential problem of incorrect matches in complex data representation contexts.

We first determine the 'relatedness' between a property pair to decide a match, which also reduces the search space. Note that while SKOS[7] is a formal specification of concept relatedness in ontologies, there is no such specification for properties. We now present some notions to help represent property alignment on linked data. We first define the notion of *candidate match* between two properties.

The following statement is true for all the definitions in the paper. Let $S_1P_1O_1$ and $S_2P_2O_2$ be two triples in two different datasets D_1 and D_2 respectively representing relations $P_1(S_1, O_1)$ and $P_2(S_2, O_2)$. *ECR* are the entity co-reference links described in Section 1.

Definition 1: Candidate Match
The two properties P_1 and P_2 are a *candidate match* iff $S_1 \xleftrightarrow{\text{ECR}^*} S_2$ and $O_1 \xleftrightarrow{\text{ECR}^*} O_2$. We say two instances are connected by an ECR^* link if there is a link path between the instances using ECR links (where, * is the *Kleene star* notation).

Candidate matches can provide supportive evidence for property alignment. But there can be coincidental (spurious) matching of properties. Consider the following two triples in the datasets DBpedia(d) and Freebase(f):

d:Arthur Purdy Stout *d:place of birth* *d:New York City*
f:Arthur Purdy Stout *f:place of death* *f:New York City*

Arthur Purdy Stout is a person (in fact, a surgeon and pathologist in real life) who lived in New York City. Given that *d:Arthur Purdy Stout* is the same as *f:Arthur Purdy Stout* and *d:New York City* is the same as *f:New York City*, *d:place of birth* and *f:place of death* properties are a candidate match according to the definition. But clearly these two properties should not be treated as equivalent because they have different intentional semantics. Therefore, this coincidental match is not an equivalent match.

To minimize mis-identification of coincidental matches as equivalent (ideally eliminating them), our approach aggregates additional evidence in support of a statistical match, to approximate equivalent match (defined formally using extensions). Therefore, we keep track of some statistical measures along with the candidate matches to compute statistical equivalence. For a candidate matching property pair (P_1,P_2), *Match Count* $\mu(P_1,P_2)$ and *Co-appearance Count* $\lambda(P_1,P_2)$ can be defined as follows.

Match Count $\mu(P_1,P_2)$ is the number of triple pairs for P_1 and P_2 that participate in candidate matches. That is,

$$\mu(P_1, P_2) = |\{S_1P_1O_1 \in D_1 \mid \exists\, S_2P_2O_2 \in D_2 \wedge S_1 \xleftrightarrow{\text{ECR}^*} S_2 \wedge O_1 \xleftrightarrow{\text{ECR}^*} O_2\}| \quad (1)$$

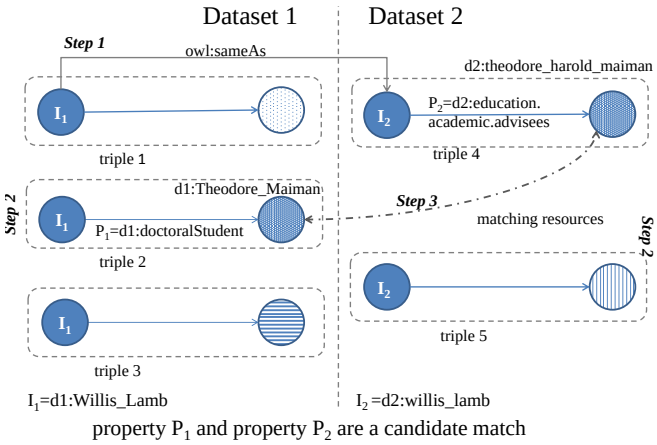

Figure 1: Process of Candidate Matching. Matching resources are in the same color/pattern.

Co-appearance Count $\lambda(P_1,P_2)$ is the number of triple pairs for P_1 and P_2 that have matching subjects. That is,

$$\lambda(P_1, P_2) = |\{S_1P_1O_1 \in D_1 \mid \exists\, S_2P_2O_2 \in D_2 \wedge S_1 \xleftrightarrow{\text{ECR}^*} S_2\}| \quad (2)$$

Statistical equivalence in this work is measured by analyzing candidate matches over co-appearances of a property pair, which provides statistical evidence, i.e., it will have many matching subject-object pairs over common subjects. Therefore, the number of matching subject-object pairs in the property extensions and co-appearances of a property pair directly influence the decision function F (defined below) for selection (captured using a confidence threshold α). Also, a property pair must co-appear enough times (supporting evidence) to be picked as a match, to overcome coincidental matches, by achieving a sufficient match count μ. Therefore, this minimum number of match count μ should be greater than a constant k. This constant k filters out many incorrect random candidate matches. Now, statistically equivalent property pairs can be defined as follows.

Definition 2: Statistically Equivalent Properties
The pair of properties P_1 and P_2 are *statistically equivalent* to *degree* (α, k) iff

$$F = \mu(P_1, P_2)/\lambda(P_1, P_2) \geq \alpha \quad where,\ \mu(P_1, P_2) \geq k,\ and\quad 0 < \alpha \leq 1,\ k > 1 \quad (3)$$

Note that, a list of statistically equivalent properties can, strictly speaking, consist of equivalent properties, sub-properties and incorrectly mapped coincidental matches.

3.2 Resource matching and generating property alignments

Our algorithm is based on exploiting entity co-reference links that exist between instances in linked data for candidate matching of property pairs. This process is further illustrated in Figure 1 by matching an instance I_1 (Willis Lamb) with I_2 using *owl:sameAs* as the co-reference (ECR) link. This is achieved in three steps. In step 1, the corresponding instance for I_1 is identified as I_2 by following

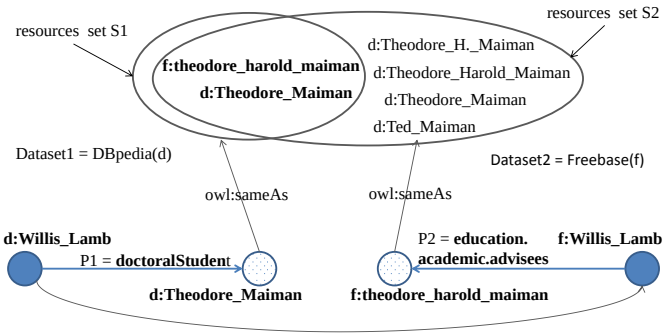

Figure 2: Property matching with overlapping sets of resources

an *owl:sameAs* link from I_1 to I_2. In step 2, the subject instances are expanded using triples they consist of. I.e., dataset 1 has three triples for I_1 as the subject and dataset 2 has two triples for I_2 as the subject. In step 3, since all subject values are matching for the triples of I_1 and I_2, finding two matching object values leads to a *candidate match*, i.e., in triples 2 and 4, object values can be matched by following an *owl:sameAs* link between them. Therefore both subject and object values are matching for triples 2 and 4 (also an extension match) leading to a *candidate match* for *doctoralStudent* and *education.academic.advisees* properties. This process is further outlined in Algorithm 1.

Algorithm 1 GenerateCandidateMatches(X, *namespace*)

Input: X, *namespace*
Output: λ, μ for each property pair

```
1:  for i = 1 → Size(X) do
2:      subject S1 ← X[i]
3:      subject S2 ← GetCorsEntity(subjectS1, namespace)
4:      map l1 ← ExtractPO(S1)
5:      map l2 ← ExtractPO(S2)
6:      for each property p ∈ l1 do
7:          object o1 ← l1(p)
8:          Set o1_ecr ← GetECRLinks(o1)
9:          for each property q ∈ l2 do
10:             if p exactmatch q then
11:                 p matches q
12:             else
13:                 update λ(p, q)
14:                 object o2 ← l2(q)
15:                 Set o2_ecr ← GetECRLinks(o2)
16:                 isMatch ← Match(o1_ecr, o2_ecr)
17:                 if isMatch then
18:                     update μ(p, q)
19:                 end if
20:             end if
21:         end for
22:     end for
23: end for
```

Comparison of the object values of triples is computed by checking for an overlap of *ECR* links, which includes checking for ECR^*. This is further illustrated in Figure 2 with only direct linking between them (one level deep path) for clarity. Further, this overlap based computation allows us to have link chains (ECR^*) to arrive (converge) at a decision. The process of candidate matching is presented in Algorithm 1 where the input to the procedure is a set X of subject instances of the first dataset and *namespace* identifier of the second dataset. The *GetCorsEntity* procedure returns the relevant corresponding entity in the second dataset for a given subject instance in the first dataset (by following an ECR link and *namespace* identifier). The *ExtractPO* procedure is for extracting all the property-object pairs for a given subject instance and returns a *map* data structure. The *map* data structure has object values stored for each property. The *GetECRLinks* procedure returns all the available corresponding entities without considering any namespaces. If two property names can be matched exactly (except namespaces), then they are considered as matched, as outlined in line 11 of the algorithm. As the process outlined in Algorithm 1 is performed on a sample instance set taken from dataset 1, candidate matches are found with μ and λ counts. Then applying function F with α and k in equation 3 for aggregated μ and λ for each property pair in the whole instance set will yield *statistically equivalent* property pairs for the two datasets. Note that *GetCorsEntity* and *GetECRLinks* procedures can be configured to use a specific ECR link or multiple types of ECR links (like *owl:sameAs, skos:exactMatch,* etc).

D1:Dataset 1	D2:Dataset 2	Matching Subject Pairs	Matching Object Pairs
S_1 doctoralStudent O_1	S_1 academic.advisees O_1	(S_1 , S_1)	(O_1 , O_1)
S_1 birth_place O_2	S_1 place_of_birth O_3	(S_1 , S_1)	---
S_2 doctoralStudent O_4	S_2 influenced O_4	(S_2 , S_2)	(O_4 , O_4)
S_2 birth_place O_5	S_2 place_of_birth O_5	(S_2 , S_2)	(O_5 , O_5)
	S_2 place_of_death O_5	(S_2 , S_2)	(O_5 , O_5)
S_3 doctoralStudent O_6	S_3 academic.advisees O_6	(S_3 , S_3)	(O_6 , O_6)
	S_3 influenced O_7	(S_3 , S_3)	---
S_3 birth_place O_8	S_3 place_of_birth O_8	(S_3 , S_3)	(O_8 , O_8)

Generated Candidate Matching Property Lists – Matches selected are in Boldface

[D1:doctoralStudent] ⟶ [**[D2:academic.advisees, 2:2]** , [D2:influenced, 1:2]] *list1*

[D1:birth_place] ⟶ [**[D2:place_of_birth, 2:3]** , [D2:place_of_death, 1:1]] *list2*

Property Pair	MatchCount	Co-appearanceCount
[D1:doctoralStudent, D2:academic.advisees]	2	2
[D1:birth_place, D2:place_of_birth]	2	3

Figure 3: Calculating *MatchCount* and *Co-appearanceCount* values

Figure 3 shows an example for calculating μ and λ. In this example, the algorithm used α=0.5 and k=2 to decide on matches from each list of candidate matching pairs and eliminate coincidental matches. Recall that α is the minimum fraction of matching triples over co-appearances and k is the minimum number of matching triples required to be considered for equivalence. The need for k together with α can be further explained using *list1* & *list2* in Figure 3, where "doctoralStudent" matched to "influenced" and "birth_place" matched to "place_of_death" with satisfying α. Thus use of k avoids such coincidental (incorrect) matches. It also illustrates the nature of our bootstrapping algorithm that decides on a matching property pair after analyzing the evidence from all matching subject-object pairs of the property extensions.

3.2.1 Complexity Analysis

Let the average number of properties for an entity be x and average number of objects for a property be j. Then, Algorithm 1 has to compare $j * j * x * x$ object value pairs for an entity pair. For n number of subjects, it would be $n * j * j * x * x$. To extract property-object pairs, it requires $2 * n$ operations[3]. x and j are independent of n as number of properties per instance and number of object values per property do not change on average for larger or smaller n. Since $n > j$ and $n > x$, $O(n * j^2 * x^2 + 2n)$ is n. The inherent parallel nature of Algorithm 1 allows us to adapt it

[3]If comparisons are not restricted this way to each subject, a naive algorithm would need $n^2 * j^2 * x^2 + 2n$ comparisons.

to Map-Reduce paradigm to improve efficiency.

3.3 Implementation details

The algorithm is implemented using Java, Apache Hadoop and Jena framework for processing RDF triples. Datasets are replicated locally using Virtuoso triple store except for Freebase dataset for which we used their public APIs.

In this experiment, we mainly used *owl:sameAs* and *skos: exactMatch* links as *ECR* (*owl:sameAs* was the dominant link of the two for the selected datasets). When searching for *ECR* links between entities, we investigated one level deep paths in this experiment (see Figure 2) since it was sufficient for the experimental datasets. But this can be extended further by examining more than one level deep link paths (ECR^*) for enhanced coverage. We also used exact matching of *rdfs:labels* of the two objects when comparing *ECR* links for object equivalence. This is used as an approximation to *ECR* links for the comparison of object values (line 16 of Algorithm 1) to improve coverage in absence of *ECR* links. This is because, we are looking for exact matching of labels for objects that belong to the same entity in two datasets. It is a good approximation as the subject (entity) is guaranteed to be the same, since only *ECR* links are used for subject matching. This approximation also tries to compensate for sparseness of *ECR* links.

The process presented in Algorithm 1 requires a lot of comparisons and could grow for larger instance sets. But the algorithm was easily adapted to Map-Reduce framework by distributing subject instances among mappers. The specific implementation is explained below:

- **Map phase**
  - Let the number of subject instances in dataset 1 (D1) be X and namespace of dataset 2 (D2) be ns. For each subject $i \in X$, start a mapper and call GenerateCandidateMatches(i, ns) outlined in Algorithm 1.
  - Each mapper outputs (key,value) pairs as (p:q, μ(p,q):λ(p,q)) to reducer, where property p$\in$D1 and property q$\in$D2.
- **Reducer phase**
  - Collects output from all mappers and aggregates μ(p,q) and λ(p,q) values for each key p:q.

The process can be parallelized since computation of one subject instance is independent of the others. We implemented the algorithm in Hadoop Map-Reduce 1.0.3 framework and achieved significant improvements in running time on a 14 node cluster (speedup of 833% compared to the desktop version on average). Moreover, we may achieve faster times when more resources are available to paralleize. Further detailed discussion on this is out of scope of this paper.

When the algorithm is run for a sample set of instances starting from dataset $D1$, we can generate candidate matches of property pairs. One property may be matched to many other properties because of similar extensions, as explained in Section 3.1. After recording candidate matches, each property in $D1$ has been mapped to a list of properties that are in dataset $D2$ with match counts (see Figure 3). The most probable matching properties will have higher match counts in the list and also have higher values for function F. We sort each list in descending order based on match counts and then test the first property pair from each sorted list to see whether it satisfies threshold α and k for a match[4]. Applying these thresholds and sorting the lists remove many of the coincidental matches and leave probable matches. The algorithm outputs these as *statistically equivalent*. In this experiment, we were able to remove 76%, 87%, 67%, 83%, and 25% coincidental matches respectively from Experiments 1 to 5 [5]. These statistics also reveal that simple object value overlap or grouping mechanisms [9][17] are not adequate for finding equivalent properties in the LOD context.

4. RESULTS AND EVALUATION

The objective of this evaluation is to show that *statistical equivalence* of properties can successfully approximate *owl:equivalentProperty* in the LOD context and the developed algorithm performs considerably better than the existing techniques used for property alignment, which includes string similarity and external hierarchies (i.e., WordNet) based approaches. To prove our claim, we chose DBpedia[6], Freebase[7], LinkedMDB[8], DBLP L3S[9], and DBLP RKB Explorer[10] datasets and extracted sample instance sets for the following reasons. (1) They have more or less complete information for instances. (2) They are well interlinked. (3) They have complex/opaque[11] properties. (4) They cover several dimensions of our evaluation as multi-domain to multi-domain, specific-domain to multi-domain and specific-domain to specific-domain dataset property alignment.

Our property alignment results are presented in Table 1. We randomly selected 5000 subject instances for each experiment, numbered 1 to 5 in Table 1. The experiments cover object-type properties in person, film, and software domains between DBpedia and Freebase, films between DBpedia and LinkedMDB, and articles between DBLP RKB explorer and DBLP L3S datasets. DBLP RKB explorer and DBLP L3S are two separate datasets for DBLP that use two different ontologies. Furthermore, the DBLP RKB explorer project had different requirements, where it needed to achieve high precision and recall on search services compared to DBLP L3S dataset. DBpedia and Freebase are both multi-domain huge data hubs in LOD with overlapping information but have non-trivial differences in their schema. Freebase uses more blank nodes, increasing its complexity compared to DBpedia. LinkedMDB is a specialized dataset for movies and has its own schema which is completely different from DBpedia. Therefore, the datasets selected for the experiments from LOD are different from each other in both complexity and variety of data representation.

[4]Assuming both datasets do not have duplicate properties. In our experiments, DBpedia had duplicate properties.
[5]From 732, 355, 221, 255 and 4 generated candidate property pairs respectively.
[6]http://dbpedia.org/
[7]http://freebase.com/
[8]http://linkedmdb.org/
[9]http://dblp.l3s.de/d2r/
[10]http://dblp.rkbexplorer.com
[11]Complex or opaque properties are semantically the same but have different word selections in describing the relationship.

	Measure Type	DBpedia-Freebase (Person)[1]	DBpedia-Freebase (Film)[2]	DBpedia-Freebase (Software)[3]	DBpedia-LinkedMDB (Film)[4]	DBLP_RKB-DBLP_L3S (Article)[5]	Average
Our Algorithm	Precision	**0.8758**	**0.9737**	0.6478	0.7560	**1.0000**	**0.8427**
	Recall	**0.8089***	**0.5138**	**0.4339**	**0.8157**	**1.0000**	**0.7145**
	F measure	**0.8410***	**0.6727**	**0.5197**	**0.7848**	**1.0000**	**0.7656**
Dice Similarity	Precision	0.8064	0.9666	0.7659	**1.0000**	0.0000	0.7078
	Recall	0.4777*	0.4027	0.3396	0.3421	0.0000	0.3124
	F measure	0.6000*	0.5686	0.4705	0.5098	0.0000	0.4298
Jaro Similarity	Precision	0.6774	0.8809	**0.7755**	0.9411	0.0000	0.6550
	Recall	0.5350*	**0.5138**	0.3584	0.4210	0.0000	0.3656
	F measure	0.5978*	0.6491	0.4903	0.5818	0.0000	0.4638
WordNet Similarity	Precision	0.5200	0.8620	0.7619	0.8823	**1.0000**	0.8052
	Recall	0.4140*	0.3472	0.3018	0.3947	0.3333	0.3582
	F measure	0.4609*	0.4950	0.4324	0.5454	0.5000	0.4867

Table 1: Alignment results of object-type properties. Experiments are numbered 1 to 5.

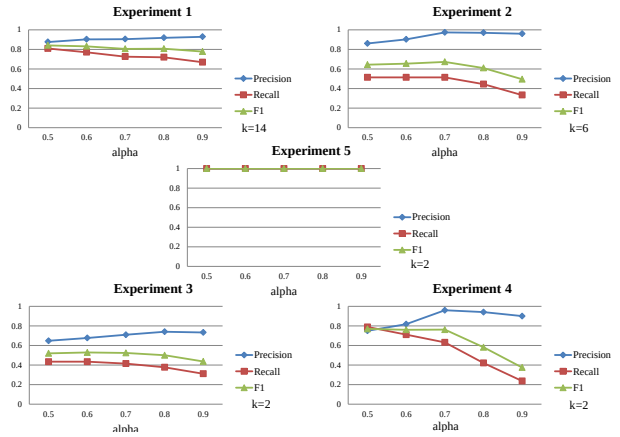

Figure 4: Precision, Recall and F measures for varying α values

4.1 Deciding α and k values

Estimating the values for α and k was done based on the following facts. (1) Data in different datasets on LOD are not complete and contain similar but not identical information. (2) Representation of these data is not uniform due to multiple authors and naming preferences. (3) A resource is not guaranteed to be linked to all matching resources. Because of these reasons, the same property in two different datasets cannot be expected to have close values for μ and λ for a higher F value closer to 1. Therefore, based on the above observations and our empirical evaluation, a threshold value closer to 0.5 for α seemed to be appropriate considering F measure. Figure 4 further clarifies this claim in our empirical evaluation showing precision, recall, and F measure for varying α with the chosen k values. Results presented in Table 1 are using 0.5 for α except for experiment 2 where it is 0.7 for optimal F measure results.

The constant k is also affected by the above reasons and we followed a data driven approach to approximate a suitable value for k as follows. First the algorithm output is filtered using α=0.5 and k=2 which means, the lowest possible matching with a positive confidence level[12]. Then we get the μ values for property pairs matched using exact string matching, which are not identified by the algorithm and get the average of the counts as constant k. Our analysis over k suggests that most of the time, optimal performance for the algorithm is achieved by a value closer to this approximation of k (default set to 2). Following this approach, we used 14, 6, 2, 2, and 2 for k in the experiments numbered as 1, 2, 3, 4, and 5 respectively in Table 1. Increasing both these thresholds α and k yields high confidence matches as they demand a higher number of extension matching.

4.2 Experiment setting

We compared our bootstrapping based algorithm with two string matching algorithms and WordNet[13] similarity suggested by [17]. We calculated WordNet similarity[14] by searching for at least one matching object value [17] and then applying WordNet similarity on pre-processed terms of the properties. We tokenized and removed stop words from properties to calculate WordNet similarity and added Porter's stemming algorithm for string similarity. The results shown for these similarities are the optimal ones considering F measure metric. The threshold values used for string similarity and WordNet similarity algorithms were 0.92 and 0.80 respectively. For experiment 5, Dice and Jaro similarity didn't show any matching properties even for a threshold value of 0.5.

We used three independent reviewers to evaluate the matching of the properties and we chose the majority vote to determine the correct matches. The evaluators were given some sample matches (2-3), and if the meaning of the properties were hard to understand by their name (which is the case for most properties), they were provided with queries to execute and explore details about them, like instances they connect to and their domain and range. They were not asked specifically to distinguish between an exact match and a sub-property match in the alignment. For the experiment numbered 1 in Table 1, we could not find all possible correct matches because the total number of combinations is large for a manual evaluation($\sim$ 39k pairs). Therefore, we gave evaluators all the mappings found by the algorithm without any threshold applied (extracted from property lists as shown in Figure 3). Hence, recall and F measure are just approximations and they are marked with an * in Table 1 for experiment 1. Among the above mentioned comparable techniques, in every case, our bootstrapping based approach showed higher recall and F measures.

4.3 Types of properties identified

Our algorithm identified different kinds of matching property pairs. It can potentially subsume approaches that use techniques such as string similarity and synonym checking as shown in Table 2 using extensional matching. The sample pairs listed in Table 2 are all correctly identified by our algorithm. To explain example matching pairs in Table 2,

[12] α<0.5 is a negative confidence level since less than half of subject-object pairs got matched from common subjects.

[13] http://wordnet.princeton.edu/

[14] http://www.sussex.ac.uk/Users/drh21/

Matching Category	Dataset 1	Dataset 2
String Similarity	db:nationality db:religion	fb:nationality fb:religion
Synonymous	db:occupation db:battles	fb:profession fb:participated_in_conflicts
Complex Matches	db:screenplay db:doctoralStudents	fb:written_by fb:advisees

Table 2: Sample of matching properties under different categories. namespaces: db for DBpedia and fb for Freebase.

consider the following observations.

"String Similarity Matches" can be identified using string similarity measures such as Dice's Coefficient or simple character comparison on property names. *"Synonymous Matches"* are mappings that can be identified by analyzing synonyms of the property names. For example, occupation and profession can be mapped using a dictionary, but interestingly the WordNet approach failed to match this pair for the provided threshold (showed very low similarity value of 0.5). In fact, both pairs were missed by the WordNet approach. *"Complex Matches"* are the last category shown in Table 2 and are harder to identify. All of the approaches outlined in the evaluation missed this mapping except our algorithm. This is because our approach exploits property extensions in aligning properties. There are also false positives in our result set. One such property pair is *http://dbpedia.org/property/issue* and *http://rdf.freebase.com/ns/people.person.children*, which happens to have similar extensions but has different intentions.

5. DISCUSSION

The results show that our approach can effectively identify equivalent object properties by utilizing different statistical parameters presented above. It performs well even when complex properties exist, like in experiments 1 and 5 in terms of F measure metric. Our algorithm also performs well in terms of precision when datasets contain more literally similar property names as in experiments 2, 3, and 4, and discovers more interesting matches in every case, showing higher recall, e.g., experiment 5 is about aligning properties in the same domain (publication) with high overlapping information, but represents data using two completely different ontologies. These two ontologies do not have similar word selection or synonyms in their schema exemplifying complex data representation typically found on LOD. Because of this, string similarity based approaches do not perform well and synonym based approaches also show poor coverage over terms. Therefore, novel techniques such as ours for discovering equivalent properties on Linked Data (i.e., Complex Matches in Table 2) are indispensable[15].

We selected DBpedia, Freebase, LinkedMDB, and DBLP for our evaluation because of the existence of many entity co-reference links between their entities, specifically, DBpedia and Freebase provide diverse and complex property sets. Alignment between DBpedia and Freebase evaluates multi-domain property alignment whereas DBpedia and LinkedMDB covers multi-domain and specific-domain dataset alignment. The DBLP alignment evaluates our algorithm between specific-domain datasets. Therefore, our evaluation covers property alignment between different types of datasets that can arise in the LOD domain. When analyzing results of matching frequency for property pairs, it was observed that some matching properties do not appear frequently enough. Consequently, the algorithm's confidence in picking them as a match is low. This is mainly due to the nature of properties, as discussed in Section 3. When a random sample set is selected, it contains instances belonging to various types, e.g., a person type can have instances belonging to athlete, artist, etc., which have rare properties. We can run the algorithm iteratively for more subject instances that have these less frequent property pairs to improve precision. Our algorithm assumes that there are no duplicate properties in both datasets being aligned. If both datasets have duplicate properties, it requires additional inferencing mechanism to identify missing pairs, since only one matching pair from each candidate list is selected.

We observed that certain properties that are mapped as equivalent properties are actually sub-properties. This happens mainly because we do not distinguish between equivalent properties and sub-properties. For an example, *http://dbpedia.org/property/mother* and *http://rdf.freebase.com/ns/people.person.parents* are two such properties matched as equivalent, while, in fact, the first is a sub-property of the second. We would like to distinguish sub-properties in our future experiments for more fine grained matches. Also, we believe ECR^* links are useful in general, where many datasets are linked to central hubs like DBpedia and the algorithm will be able to discover more connections between resources by finding link paths via these hubs.

We believe the uniqueness of our approach has resulted in high quality results compared to existing approaches because it, (1) does extension matching using entity co-reference links, (2) bootstraps from candidate matches and aggregates the results, and (3) makes final alignments using statistical measures analyzing aggregated confidence of the matching pairs. While, string similarity and synonymous matching have been shown to be effective in the past (primarily for literals), they do not have sufficient coverage for resources as shown in our evaluation. We use *owl:sameAs* as an entity co-reference link but its use to link two semantically equivalent instances seems to be controversial in the LOD community [5][6]. In spite of well-known shortcomings, (like equating "London" to "Greater London"), our algorithm is expected to be sufficiently robust for property alignment, because it is based on aggregating information from a large number of entity equivalence assertions. In other words, we believe that the effect of a few misused links will not affect the final result much, because our algorithm does not decide on a property match by analyzing a single matching triple. Even though the interlinking (entity co-reference) relationships are small compared to the size of similar instances between datasets today, we expect that these links will become prevalent as the datasets evolve and are maintained as part of the "linked data life cycle" (with projects such as http://latc-project.eu/ and http://stack.lod2.eu/). Moreover, we can see other recent successful efforts in using *owl:sameAs* networks in the LOD context[11][17] in spite of these known issues.

We also identified some property pairs which are matched but have incorrect meaning in the property name. For example, *http://dbpedia.org/property/issue* and *http://rdf.freebase*

[15]More details can be found at `http://wiki.knoesis.org/index.php/Property_Alignment`

.com/ns/people.person.children property pair between DBpedia and Freebase. This happened because of mixed values present in the DBpedia property *http://dbpedia.org/property /issue*, that has both integer values and names of children as object values. In this case, we can regard this kind of a property as ambiguous and having noise, which has a negative effect on the matching process. Another instance where the algorithm can go wrong is when it encounters special cases where the two datasets have enough facts for the process to identify two properties as a match, but actually they are not. For an example, for film domain, it maps *http://dbpedia.org/ontology/distributor* to *http://rdf.freebase .com/ns/film.film.production_companies*. This mainly happens because a number of production companies who produce also distribute films. Thus, they have multiple roles and the extensional equality may choose to match one of the many roles (extensions match but different intentions).

6. CONCLUSION AND FUTURE WORK

We have developed and evaluated an extension-based approach to match object-type properties on linked datasets. We have defined a computable concept of *statistical equivalence* of properties to approximate *owl:equivalentProperty* by leveraging entity co-reference links. Our algorithm is unique and novel in how it computes extensional equivalence iteratively by building candidate matches in parallel. This approach ultimately determines statistically equivalent property pairs. The algorithm is easily parallelizable as evidenced by our straight forward Map-Reduce implementation. The empirical evaluation shows that our approach is superior to the state of the art using F measure metric (on average, F measure gain is in the range 57% - 78%). This suggests that it is possible to align object-type properties on LOD datasets effectively. Our approach subsumes conclusions arrived at by string-based and WordNet-based similarity metrics and discovers more hidden and interesting matches well-suited for the LOD cloud. However, we can use string-based or WordNet-based approaches to further improve confidence in our results to minimize coincidental matches. In fact, our data driven approach can also be adapted to align data-type properties by using similarity metrics for object values in triples. Another beneficial side effect of object-type property alignment is that it may be used to generate or discover potential co-reference links between instances. Furthermore, we expect to test the algorithm on different levels (strength) of entity co-reference links. In future, we will extend the coverage to more types of data and discover domains in which properties exist for better alignment. The alignment problem can be further refined to determine sub-properties in the future that will help with fine grained data integration tasks.

Acknowledgements

This work was supported by the National Science Foundation under award 1143717 “III: EAGER Expressive Scalable Querying over Linked Open Data”.

7. REFERENCES

[1] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, et al. Owl web ontology language reference. *W3C recommendation*, 10:2006–01, 2004.

[2] T. Berners-Lee. Linked data-design issues. *W3C*, 2009(09/20), 2006.

[3] L. Ding, J. Shinavier, Z. Shangguan, and D. McGuinness. Sameas networks and beyond: analyzing deployment status and implications of owl: sameas in linked data. *The Semantic Web–ISWC 2010*, pages 145–160, 2010.

[4] T. Franz, A. Schultz, S. Sizov, and S. Staab. Triplerank: Ranking semantic web data by tensor decomposition. *The Semantic Web-ISWC 2009*, pages 213–228, 2009.

[5] H. Halpin, P. J. Hayes, J. P. McCusker, D. L. McGuinness, and H. S. Thompson. When owl: sameas isnâĂŹt the same: An analysis of identity in linked data. In *The Semantic Web–ISWC 2010*, pages 305–320. Springer, 2010.

[6] P. Jain, P. Hitzler, P. Yeh, K. Verma, and A. Sheth. Linked data is merely more data. *Linked Data Meets Artificial Intelligence*, pages 82–86, 2010.

[7] A. Miles and S. Bechhofer. Skos simple knowledge organization system reference. *W3C Recommendation*, 2008.

[8] A. Ngomo and S. Auer. Limes-a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, 2011.

[9] K. Nguyen, R. Ichise, and B. Le. Slint: A schema-independent linked data interlinking system. *Ontology Matching*, page 1, 2012.

[10] B. Nunes, A. Mera, M. Casanova, K. Breitman, and L. Leme. Complex matching of rdf datatype properties. *MCC12/11, Dept Informatics, PUC-Rio (September 2011)*.

[11] R. Parundekar, C. A. Knoblock, and J. L. Ambite. Discovering concept coverings in ontologies of linked data sources. In *The Semantic Web–ISWC 2012*, pages 427–443. Springer, 2012.

[12] A. Sheth, I. Arpinar, and V. Kashyap. Relationships at the heart of semantic web: Modeling, discovering, and exploiting complex semantic relationships. *Studies in Fuzziness and Soft Computing*, 139:63–94, 2004.

[13] J. Sleeman, R. Alonso, H. Li, A. Pope, and A. Badia. Opaque attribute alignment. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 17–22. IEEE, 2012.

[14] Q. Tran, R. Ichise, and B. Ho. Cluster-based similarity aggregation for ontology matching. *Ontology Matching*, page 142, 2011.

[15] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig. ma: live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):355–364, 2010.

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

[17] L. Zhao and R. Ichise. Graph-based ontology analysis in the linked open data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 56–63. ACM, 2012.