SERIMI: Class-based Disambiguation for Effective Instance Matching over Heterogeneous Web Data

Samur Araujo
Delft University of Technology
Delft, the Netherlands
s.f.cardosodearaujo@tudelft.nl

Duc Thanh Tran
Karlsruher Institute of Technology
Germany
ducthanh.tran@kit.edu

Arjen P. de Vries
Delft University of Technology
Delft, the Netherlands
a.p.devries@itudelft.nl

Jan Hidders
Delft University of Technology
Delft, the Netherlands
a.j.h.hidders@tudelft.nl

Daniel Schwabe
Informatics Department
PUC-Rio
Rio de Janeiro, Brazil
dschwabe@inf.puc-rio.br

ABSTRACT

Instance matching has been studied with focus on the single-domain setting, while less attention is given to the heterogeneous environment of the Web, where data comes from different domains and are associated with different schemas. For this heterogeneous setting, we propose an unsupervised schema-agnostic approach that focuses on the refinement (*disambiguation*) of candidate instances (resulting from blocking). Given instances of a source dataset that belong to a class, it computes candidates in the target datasets and refines them such that the *remaining matches correspond to the source instances at the class level*. However, no schema knowledge and explicit correspondences between classes in the source and target datasets are required for this. Rather, the disambiguation is performed based on an *instance-based representation of classes* computed online. We evaluated our work using experiments on large-scale real-world datasets provided by a benchmark. The proposed solution outperformed two alternative approaches for instance matching in 70% of the cases, and in those cases we improved average F-measure by 10%.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems; H.2.5 [**Database Management**]: Heterogeneous Databases

Keywords

data integration, instance matching, linked data

1. INTRODUCTION

Instance matching [3] (also known under other names such as entity resolution or record linkage) refers to the problem of determining whether two descriptions are about the same real-world entity. Traditionally, research in this context was focused on the *single-domain setting*, where data come from the same or similar datasets. Basically, given the descriptions of entities available as records in databases, RDF descriptions on the Web, etc., the instance matching task breaks down to the core problems of (1) finding a suitable *representation* (i.e., selecting attributes), (2) using this for *matching*, and (3) finally *selecting* the most similar ones (according to a threshold). There are *data blocking* techniques that based on simple representations of entities, can quickly identify candidate records [7, 4]. Then, for more sophisticated and *effective matching*, there are different types of similarity measures [3, 6], and different techniques for *learning* the right combination of attributes, similarity measures and threshold to be used for computing and selecting the resulting matches [20].

While these single-domain solutions have shown high quality results in enterprise data integration scenarios, their applicability to the large-scale heterogeneous Web setting is less clear. Assumptions implicitly embodied in these solutions no longer apply. Firstly, in the larger scale Web setting that involves multiple domains, it is more expensive to obtain the necessary amount of *training data*. More importantly, instances are assumed to have similar representations (i.e. schemas) so that a subset of their common attributes can be selected for matching. This *similar representation* assumption however, holds only for instances that are from the same dataset – or similar ones with largely overlapping schemas that have been aligned upfront – but it does not apply to instance data on the Web that come from heterogeneous datasets. The following example illustrates the challenges in this setting of heterogeneous Web data integration.

Example. There are two descriptions of the `anemia` disease that were extracted from two different datasets (Diseasome and DBpedia). While the description from Diseasome describes genetic aspects (Fig. 1, line 1), the one from DBpedia captures general aspects (Fig. 1, line 7) of `anemia`. The only token they have in common is "Anemia", while their schemas do not overlap at all. Using existing blocking techniques [15] that compare instances simply by tokens, these instances can be identified to be candidate matches. However, this token

 Fifteenth International Workshop on the Web and Databases (WebDB 2012), May 20, 2012 - Scottsdale, AZ, USA.

```
1  diseases:85
2      diseasome:associatedGene gene:ABCB7, gene:ALAS2 ;
3      diseasome:possibleDrug drug:3628, drug:1349 ;
4      diseasome:degree "3" ;
5      diseasome:name "Anemia".
6
7  dbpedia:Anemia
8     prop:deathCause dbpedia:Simon_Monjack ;
9     a ont:Disease ;
10    rdfs:label "Anemia"@en ;
11
12 dbpedia:Anemia_%28fern%29
13     a dbpedia-owl:Eukaryote, dbpedia-owl:Fern, dbpedia-owl:Plant;
14     dbpedia-owl:order   dbpedia:Schizaeales ;
15     dbpprop:name    "Anemia"@en .
```

Figure 1: Examples for "Anemia" in N3 notation (prefixes are used for brevity).

match is not enough to guarantee these instances refer to the same disease. In particular, blocking may yield other candidates, such as `anemia` as a plant, as shown in Fig. 1, line 12. Applying more sophisticated techniques to refine these candidates is not directly possible in this setting because there are no common attributes that can be used, e.g. attribute-specific learning and tuning of similarity measures and thresholds [3] do not apply here.

We noted that the specific problem of instance matching in the Web setting with possibly non-aligned and non-overlapping schemas is largely unsolved. To the best of our knowledge, only schema-agnostic blocking techniques are applicable here, e.g. the one recently proposed for the same setting of heterogeneous Web data integration, which simply extracts all tokens from entity descriptions and used them to compute candidate matches [15]. Complementary to this line of work, we propose class-based disambiguation that helps to refine these candidates.

Contributions. This paper introduces SERIMI, an approach that focuses on the effective matching of candidate instances resulting from blocking. It specifically addresses the mentioned challenges. It is completely unsupervised and thus does not require training data. More importantly, it supports the matching of instances that are from different domains and schemas. The technical contribution behind this work is the *class-based disambiguation of instances*. Given instances of a particular class of interest (e.g. a RDF class or a database relation) in the source dataset, SERIMI quickly finds candidate matches in the target dataset, computes the class in the target dataset that corresponds to the class in the source, and finally, uses it to filter out candidates that do not belong to the class of interest. Because the target class is represented based on instances in the target dataset and is computed on-the-fly, SERIMI neither relies on knowledge about the schema nor explicit correspondences between classes (i.e. does not require schemas to be pre-aligned). We performed experiments on large-scale Web datasets, and compared SERIMI with two recently proposed solutions, RiMOM [10] and ObjectCoref [8]. SERIMI outperformed these alternative approaches in 70% of the cases, and in those cases it improved average F-measure by 10%.

Outline. This paper is organized as follows: After this introduction, we discuss related work in Section 2. In Section 3, we discuss the problem of disambiguation in instance matching. In Section 4, we elaborate on our class-based disambiguation method. Section 5 presents the experimental results, and Section 6 concludes this paper.

2. RELATED WORK

Instances are similar, thus, are considered candidate matches if their *features* are similar [5]. Features used are derived from flat attributes, structure information of instances (e.g. relations between RDF resources) [13, 16] or semantic information. While we focus on the use of flat attribute values in the experiment, SERIMI is also applicable to other features.

Instance matching using flat features typically relies on string comparison using different *similarity metrics*. Although there are many metrics, there is no single one that applies in all cases [2]. Learning the right metrics for the given features, and combining different metrics [1] are the best strategies. Which metrics to be used is also not the focus here, where we simply employ a string-based metric for the experiment.

Orthogonal to features and metrics, different *matching techniques* have been proposed to address both the efficiency and effectiveness of instance matching. *Data blocking techniques* [7] aims to make it more efficient by reducing the number of unnecessary comparisons between records. Based on a feature that is distinctive (also called Blocking Key Value, BKV), instances are partitioned into blocks such that potentially similar instances (i.e. candidate results to be further refined) are placed in the same block [7, 12]. Recently, an unsupervised blocking technique has been explicitly proposed for the heterogeneous Web setting, where the BKV is simply the set of all tokens that can be extracted from the instance data [15]. Silks [9] is another solution for this setting, which however, requires a manual identification of the BKV.

There are two major kinds of approaches that target the effectiveness of matching. Usually, they are employed after blocking for the disambiguation of candidate matches. There are *learning-based approaches* that can be further distinguished in terms of training data and degree of supervision, respectively (i.e. supervised, semi-supervised, unsupervised [19, 17, 14]). ObjectCoref is a supervised approach that self-learns the discriminativeness of RDF properties. Then, matches are computed based on comparing values of a few discriminative properties. RIMON is an unsupervised approach that firstly applies blocking to produce a set of candidate resources and then, uses a document-based similarity metric (cosine similarity) for disambiguating candidate resources. *Collective matching* represents the other kind of approach [16]. It exploits the intuition that two instances are similar if their neighbours are similar. Similarity flooding [13] is a generic graph-matching algorithm that implements this intuition.

3. PROBLEM DEFINITION

We target the setting of heterogeneous Web data where only little or no overlap between the schemas of the source and target instances exist.

3.1 Preliminary

Web data, including relational data, XML and RDF (Resource Description Framework), can be conceived as graphs. RDF data consist of triples, which collectively form a graph.

Closely resembling this RDF data model we conceive Web data as graphs:

Definition 1. Data model - Data on the Web are modeled as a set of graphs $\mathbb{G}$, where every graph G $\in \mathbb{G}$ is a set of triples, each of the form (s, p, o) where s $\in$U (called subject), p $\in$ U (predicate) and o $\in$ U $\cup$ L (object). Here, U denotes the set of Uniform Resource Identifiers (URIs) and L the set of literals, which together, form the vocabulary V = U $\cup$ L.

With respect to this model, instances are resources that appear at the subject position of triples, and the attribute-value pairs of an instance s correspond to the predicate-object pairs that appear in triples where s is the subject (attribute and predicate are used interchangeably in the following). The representation of an (set of) instance is defined as follows:

Definition 2. Instance Representation -The *instance representation* $IR : \mathbb{G} \times 2^U \rightarrow \mathbb{G}$ is a function, which given a graph G and a set of instances W, yields a set of triples in which $s \in W$ appears as the subject, i.e. $IR(G, W) = \{(s, p, o)|(s, p, o) \in G, s \in W\}$.

Notice that a representation of a single instance s is given by $IR(G, \{s\})$. For simplicity, we use in this work the outgoing edges (s, p, o) of a resource s to form its representation $IR(G, \{s\})$. Based on this representation, instance matching can be posed as a *direct matching problem* as follows:

Definition 3. Instance Matches - Given two instances s and t, from G_s and G_t respectively, and a similarity relation over their representations, denoted as $\sim_I$, they match if IR (G_s, {s}) $\sim_I$ IR(G_t, {t}).

3.2 Solution Overview

Our solution goes beyond this direct matching (i.e. matching an instance directly against one other) to perform an additional step of class-based disambiguation. It is based on the observation that for a collection of semantically related source instances, target matches are also semantically related among themselves. In particular, we consider the case where resources are semantically related in the sense that together, they form a *class*. Notice, that, in this paper, a class means a set of instances that share some similar attributes. Then, the intuition behind our approach is as follows: Given source instances that belong to a particular class C, and a set of candidate matches, we consider candidates as correct when (1) they belong to the same class. (2) Further, because candidates should match source instances, this class must be similar to C. In RDF, class information may be explicitly given in the form of (s, rdf:type, o) triples, or directly derived from the data by grouping together instances that share some attributes [11].

We cast the instance-matching problem posed above as follows: Given a set of instances S in a graph G_s that belong to a class C (e.g. Diseases), and for each instance $s \in S$, let the set of instances $T \in \mathbb{T}$ in a target graph G_t be candidate matches. Only some of these candidates are correct matches, and based on our observation, we infer that correct matches must belong to a class similar to C. The problem then breaks down to (1) finding a class representation in G_t that correspond to C, and (2) refining $\mathbb{T}$ by filtering out the instances $t \in T \in \mathbb{T}$, which do not belong to this class.

We leverage existing work on blocking to deal with the first sub-problem: For each instance $s \in S$, we obtain the candidate sets $T \in \mathbb{T}$ that we call *pseudo-homonym sets*:

Definition 4. Candidate Matches / Pseudo-homonym Set - Given the source dataset G_s, the target dataset G_t and a similarity relation over the vocabulary V, denoted as $\sim_V$, a *pseudo-homonym set* of an instance s is $PH(s) = \{s'|(s, p, o) \in G_s, (s', p', o') \in G_t, o \sim_V o'\}$.

Note that $PH(s)$ are in fact instance matches obtained via direct matching as discussed before. We explicitly introduce this notion to make clear that $PH(s)$ are only candidate matches obtained via a simple vocabulary-based matching function $\sim_V$. Further, because these matches are obtained for instances $s \in S$ of the same class C, we use the union set $PH(S) = \bigcup_{s \in S} PH(s)$ as an instance-based representation of the class in G_t that corresponds to C. That is, we assume that if there exists such a class in G_t, it is captured by instances in $PH(S)$. Then, we filter out candidates that do not belong to this "class" $PH(S)$.

Definition 5. Refined Matches - Given the instances S in G_s that belong to the class C, and $PH(S)$ the candidate matches for S and the class in G_t corresponding to C, respectively, then *refined matches* are $PH'(S) = \{t|t \in PH(S), t \sim_S PH(S)\}$.

The similarity relation $\sim_S$ used here is set-based, and is for determining whether a candidate s belong to the class of interest. This work is about implementing $\sim_S$ and performing the disambiguation based on it.

4. CLASS-BASED DISAMBIGUATION FOR INSTANCE MATCHING

In this section, we present the whole process of instance matching performed by SERIMI, and then, focus on the class-based disambiguation step.

4.1 The Instance Matching Process

The overall matching process is depicted in Fig.2. Starting from a set of instances S of the class C in the source dataset, we firstly perform candidate selection to obtain the pseudo-homonym set $PH(s)$ in the target dataset for each $s \in S$. Existing blocking techniques [7, 12, 15] can be used for this. We adopt an entropy-based approach to find blocking keys and then use key values (i.e. tokens extracted from an attribute such as name, title etc.) to determine all candidates in the target dataset that match source instances. Table 1 shows example results obtained for the `Diseasome` diseases 85, 379 and 502 using `Diseasome` as source and `DBpedia` as the target dataset. Then, more effective matching is achieved through the second step of disambiguation, where the candidates in $PH(s)$ are refined.

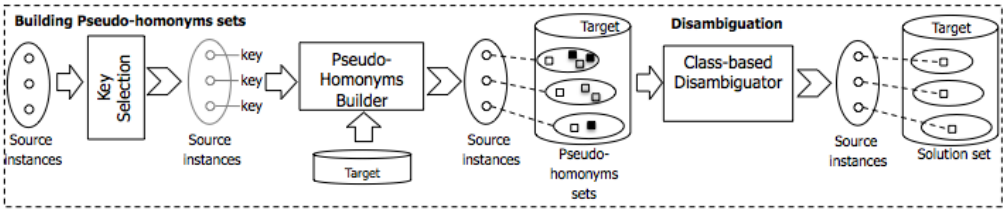

Figure 2: The process of finding and disambiguating the pseudo-homonym sets given a set of source instances.

Table 1: Pseudo-homonym sets for Anemia, Erythrocytosis and Hemophilia.

diseasome:85 Token: Anemia	diseasome:379 Token: Erythrocytosis	diseasome:502 Token: Hemophilia
db:Anemia	db:Erythrocytosis	db:Hemophilia
db:Anemia_fern	db:Familial_erythroc.	db:Hemophilia_A
db:Aplastic_anemia		db:Porphyric_Hemo.

4.2 Class-based Disambiguation

Similarity Measure. We firstly decompose the instance representation into different parts, then we elaborate on the similarity used for $\sim_S$, i.e. the measure used to disambiguate instances based on the class of interest.

Definition 6. Features- Given a graph G and a set of instances X in G, we employ the following sets of features:

- $P(X) = \{p|(s,p,o) \in IR(G,X) \wedge s \in X\}$,
- $D(X) = \{o|(s,p,o) \in IR(G,X) \wedge s \in X \wedge o \in L\}$,
- $O(X) = \{o|(s,p,o) \in IR(G,X) \wedge s \in X \wedge o \in U\}$,
- $T(X) = \{(p,o)|(s,p,o) \in IR(G,X) \wedge s \in X\}$.

Intuitively, $P(X)$ is the set of predicates that appear in the representation of X, $D(X)$ the set of literals, $O(X)$ the set of URIs, and $T(X)$ is the set of predicate-object pairs. For implementing $\sim_S$, we need to capture the similarity between sets of instances A and B, which is realized by the function we call RDS as follows:

$$RDS(A,B) = SetSim(P(A),P(B)) + SetSim(D(A),D(B)) + SetSim(O(A),O(B)) + SetSim(T(A),T(B)) \quad (1)$$

We want $SetSim$ to reflect the intuition that two sets A and B that have n features in common should be more similar than two sets with $n-1$ features in common, no matter the number of features both sets may have. We define $SetSim$[1] as follows:

$$SetSim(A,B) = |A \cap B| - \left(\frac{|A-B| + |B-A|}{2|A \cup B|}\right) \quad (2)$$

Class-based Disambiguation. In the disambiguation process, given a set of pseudo-homonyms sets $PH(S)$, we reduce our problem to the one of finding instances t from each pseudo-homonym set, i.e. $t \in PH(s) \in PH(S)$, which is more similar to all the other sets of pseudo-homonyms sets $PH(S)^- = PH(S) \setminus PH(s)$. The RDS function is used to compute this similarity, i.e. $RDS(\{t\}, PH(s')), PH(s') \in PH(S)^-$. This process is depicted in Fig. 3a for the instance h_{11}, where it is compared to the pseudo-homonyms sets $H2$ and $H3$. After the similarity is computed for all instances t in this fashion, the instance in each pseudo-homonyms set with highest score is selected (Fig. 3b). Instead of using the top-1 in the Fig. 3b, SERIMI may return top-k matches.

[1] In our experiments, this $SetSim$ measure beats the common Jaccard and Dice index by a small but consistent margin.

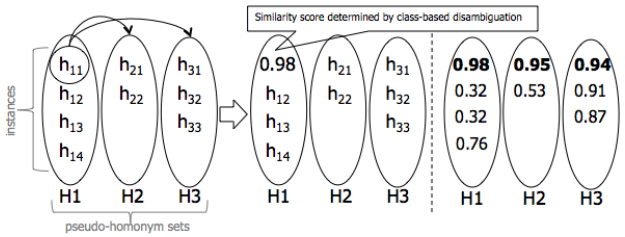

Figure 3: A) Computing the similarity score for h_{11}. B) The similarity score for all resources.

The comparisons between t and the other pseudo-homonyms sets $PH(S)^-$ is captured by Equation 3 where the individual score $RDS(\{t\}, PH(s'))$ is weighted by the cardinality of $PH(s)$, such that a $PH(s)$ with high cardinality has a smaller impact on the final aggregated measure. We do this to capture the intuition that small sets containing only a few representative instances are better representation of the class of interest.

$$URDS(t, PH(S)^-) = \sum_{PH(s') \in PH(S)^-} \frac{RDS(\{t\}, PH(s'))}{|PH(s')|} \quad (3)$$

We normalize the results of Equation 3 by the maximum score among all instances as

$$CRDS(t, PH(s), PH(S)^-) = \frac{URDS(t, PH(S)^-)}{MaxScore(PH(s), PH(S)^-)} \quad (4)$$

where

$$MaxScore(PH(s), PH(S)^-) = MAX\{URDS(t', PH(S)^-) | t' \in PH(s) \in PH(S)\} \quad (5)$$

This yields a score in the range $[0,1]$. Using this function, an instance t is considered as a solution if $CRDS(t, PH(s), PH(S)^-)$ is higher than a defined threshold δ or its rank is within the top-k. We found that a value for δ that performs well is the maximum of the means and medians of the scores obtained for all instances in $PH(S)$, which we will refer to as δ_m. In the experiment, we tested using the different settings $\delta = \delta_m$, $\delta = 1.0$, $\delta = 0.95$, $\delta = 0.9$ and $\delta = 0.85$. Also, we evaluated different top-k settings, where only the top-1, top-2, top-5 and top-10 matches were selected.

4.3 Optimization

Increasing Efficiency. When the source dataset is large, the number of pseudo-homonyms sets to consider increases, affecting the computation time of CRDS. Therefore, given S, we execute the process described so far sequentially over chunks of instances in S of size μ, where $\mu \geq 2$. Thus, we execute the CRDS function $|S|/\mu$ times. We tested the set of sizes {2, 5, 10, 20, 50, 100}. Although total time to process n instances was smaller for small μ, the variation is not significant; also, we found that the precision of matches is not affected by this parameter.

Reinforcing Evidences. Another advantage of using chunks instead of the entire set S is that at every iteration (after processing each chunk), we can select the instance with the highest score and add it as a singleton set (a set with one element) to the set $PH(S)$ of pseudo-homonym sets to be used in subsequent iterations. This extra singleton set acts as additional evidence for the class of interest.

5. EXPERIMENTAL EVALUATION

In this section, we describe our evaluation that is based on the instance-matching track of the Ontology Alignment Evaluation Initiative (OAEI). This track focuses on evaluating the effectiveness of instance-matching approaches over Web data, which is exactly the goal of the evaluation here. SERIMI was the second best system in OAEI 2011. In this paper, we do not include these 2011 results but focus on the ones we obtained for the datasets used in 2010. This is due to space limitation, and also because these 2010 datasets are more diverse in terms of heterogeneity (differences in classes of entities), which provide richer insights to the advantages and limitations of the class-based disambiguation proposed.

5.1 Experiment Setting

Collections. We used the life science (LS) collection (which includes DBPedia, Sider, Drugbank, LinkedCT,Dailymed TCM, and Diseasome) and the Person-Restaurant (PR) collection provided by this benchmark.

Evaluation metrics and alternative approaches. We used precision, recall and F1 to measure the effectiveness of the proposed approach. We considered as true positives the provided reference mapping (the ground truth). False positives are the mappings found by SERIMI that do not exist in the ground truth. For comparison, we used the results of the related approaches RiMOM and ObjectCoref as reported for OAEI 2010.

5.2 Experiment Results

Fig. 4 and Fig. 5 show SERIMI's performance as we changed δ and k. We observed that the standard deviation of precision and recall is close to zero in the cases where the pseudo-homonym sets are small. The parameters δ and k had no effect in these cases because the same (number of) instances were selected (e.g. only one) as results. Otherwise, performances varied because differences in δ (and also differences in k) leaded to a different selection of instances. The use of the δ_m, an automatically computed threshold as discussed in Section 4.3, performed relatively well on average. Therefore, for all other experiments in this paper, we used $\delta = \delta_m$.

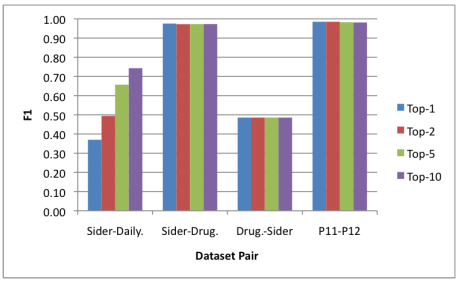

Figure 4: Top-k F-measure

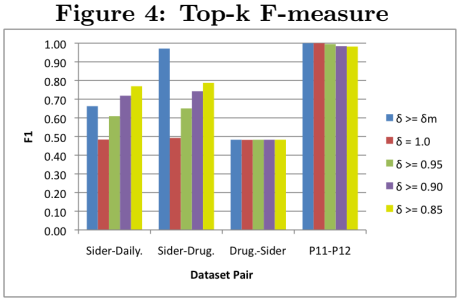

Figure 5: $\delta_{threshold}$ F-measure

As we can see in Table 4, SERIMI outperformed the alternative approaches in 70% of the cases, and in those cases it substantially improved F1 by 10% on average. Sider-Diseasome and Sider-Drugbank were problematic cases for the alternative approaches, where SERIMI achieved a gain of 42% and 47% in F1, respectively. It seems that SERIMI was more successful in refining candidates. For example, there were many candidate instances labeled "Magnesium" in DBPedia (e.g., "Isotopes_of_magnesium", "Magnesium", "Category:Magnesium", "Book:Magnesium"). SERIMI used information within the other pseudo-homonyms sets to resolve this ambiguity. Because there were much more instances of the type drugs in the other pseudo-homonyms sets (e.g., `Morphine`, `Diazepam`, `Diclofenac`) than instances of the type `book` and `category`, SERIMI was able to select the correct instance `Magnesium` that belongs to the class `drugs`.

The poor performance in the Person21-Person22 pair is due to the nature of the data. We noticed that the CRDS function did not perform well here because the instances in these datasets are exactly of the same class (i.e. `person`). SERIMI is designed for the heterogeneous case where instances to be matched belong to multiple domains. Matching instances in this single domain (same class) setting is indeed "problematic" because the idea behind SERIMI is to use class information for disambiguation. Because all candidates belong to the same class, there was not enough information for SERIMI to distinguish and disambiguate instances. Clearly, there exists a wide range of techniques for instance matching in this single domain setting and they should be used instead of SERIMI. We consider our method as a complementary approach that specifically targets the heterogeneous setting.

As a final observation, we noticed that SERIMI had a better performance that the alternative systems, specially in the hard cases(e.g. Dailymed-DBPedia, Dailymed-Linkedct, Dailymed-TCM and Drugbank-Sider).

Table 2: The precision and recall for all dataset pairs. ObjectCoref's results are not available for all pairs.

Dataset Pair / Approaches	Sider DBPedia			Sider Dailymed			Sider Diseasome			Sider DrugBank			Sider TCM		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SERIMI	0.50	0.62	0.55	0.78	0.58	0.66	0.92	0.83	0.87	0.97	0.97	0.97	0.97	0.98	0.97
RiMOM	0.71	0.48	0.57	0.57	0.71	0.62	0.32	0.84	0.45	0.96	0.34	0.50	0.78	0.81	0.79
ObjectCoref	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Dataset Pair / Approaches	Dailymed DBpedia			Dailymed LinkedCT			Dailymed TCM			Dailymed Sider			Drugbank Sider		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SERIMI	0.61	0.33	0.43	0.23	0.05	0.08	0.23	0.91	0.37	0.54	0.87	0.67	0.33	0.92	0.48
RiMOM	0.25	0.29	0.26	0.07	0.24	0.11	0.16	0.54	0.23	0.57	0.71	0.62	-	-	-
ObjectCoref	-	-	-	-	-	-	-	-	-	0.55	0.99	0.70	0.30	0.99	0.46

Dataset Pair / Approaches /	Diseasome Sider			Person11 Person12			Person21 Person22			Restaurant1 Restaurant2					
	P	R	F1	P	R	F1	P	R	F1	P	R	F1			
SERIMI	0.83	0.90	0.87	1.00	1.00	1.00	0.56	0.39	0.46	0.77	0.77	0.77			
RiMOM	-	-	-	1.00	1.00	1.00	0.95	0.99	0.97	0.86	0.77	0.81			
ObjectCoref	0.84	0.67	0.74	1.00	0.99	0.99	1.00	0.90	0.95	0.99	0.80	0.88			

In conclusion, we observed a considerable improvement of accuracy when we applied the proposed class-based disambiguation to results obtained from a simple blocking procedure. Our approach is specially recommended for cases where there is little overlap between the schemas of the instances being matched. SERIMI is available at GitHub [2].

6. CONCLUSIONS

We investigated the instance matching problem in the large and heterogeneous environment of the Web. We proposed SERIMI as a completely unsupervised schema-agnostic approach that focuses on the effective matching of candidate instances (resulting from blocking). Our approach was able to refine the ambiguous matches provided by existing blocking techniques. We outperformed two alternative approaches approaches in 70% of the cases, and in those cases we improved F1 by 10% on average. Our approach is especially recommended in situations where there are few overlaps between the source and target schemas (i.e. where traditional single domain approaches are not applicable).

7. REFERENCES

[1] K. Branting. A comparative evaluation of name-matching algorithms. In *ICAIL*, pages 224–232, 2003.

[2] W. Cohen, P. Ravikumar, and S. Fienberg. *A comparison of string metrics for matching names and records*. Aug. 2003.

[3] C. F. Dorneles, R. Gonçalves, and R. dos Santos Mello. Approximate data instance matching: a survey. *Knowl. Inf. Syst.*, 27(1):1–21, 2011.

[4] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[5] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):pp. 1183–1210, 1969.

[6] M. Hadjieleftheriou and D. Srivastava. Approximate string processing. *Foundations and Trends in Databases*, 2(4):267–402, 2011.

[7] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. pages 127–138, 1995.

[2] https://github.com/samuraraujo/SERIMI-RDF-Interlinking

[8] W. Hu, Y. Qu, and X. Sun. Bootstrapping object coreferencing on the semantic web. *J. Comput. Sci. Technol.*, 26(4):663–675, 2011.

[9] R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *WebDB*, 2011.

[10] J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Trans. Knowl. Data Eng.*, 21(8):1218–1232, 2009.

[11] Y. Ma and T. Tran. Unsupervised learning of blocking keys for web data integration. Technical report, AIFB, Karlsruhe Institute of Technology, 2011.

[12] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, pages 169–178, 2000.

[13] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

[14] X. Niu, X. Sun, H. Wang, S. Rong, G. Qi, and Y. Yu. Zhishi.me: weaving chinese linking open data. In *Proceedings of the 10th international conference on The semantic web - Volume Part II*, ISWC'11, pages 205–220, Berlin, Heidelberg, 2011. Springer-Verlag.

[15] G. Papadakis and W. Nejdl. Efficient entity resolution methods for heterogeneous information spaces. In *ICDE Workshops*, pages 304–307, 2011.

[16] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. In *J. Data Semantics IV*, pages 146–171. 2005.

[17] D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *International Semantic Web Conference (1)*, pages 649–664, 2011.

[18] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[19] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *International Semantic Web Conference*, pages 650–665, 2009.

[20] C. Xiao, W. Wang, X. Lin, and H. Shang. Top-k set similarity joins. In *ICDE*, pages 916–927, 2009.