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

Samur Araujo[1], Duc Thanh Tran[2], Arjen P. de Vries[1], Jan Hidders[1], and Daniel Schwabe[3]

[1] Delft University of Technology, PO Box 5031, 2600 GA Delft, the Netherlands
`S.F.CardosodeAraujo, A.P.deVries, A.J.H.Hidders@tudelft.nl`
[2] Karlsruher Institute of Technology, Germany
`ducthanh.tran@kit.edu`
[3] Informatics Department, PUC-Rio Rua Marques de Sao Vicente, 225, Rio de Janeiro, Brazil
`dschwabe@inf.puc-rio.br`

Abstract. The instance matching problem has been vastly studied with focus on the single-domain setting, where data come from the same or similar datasets, while less attention is given to the large and heterogeneous environment of the Web, where data come from different domains and are associated with different schemas. The applicability of single-domain solutions to this setting is less clear. In this paper, we propose a completely unsupervised schema-agnostic approach that focuses on the refinement (*disambiguation*) of candidate instances (resulting from blocking, a preprocesing step). It starts with instances of a source dataset that belong to a class. Then, candidates in the target datasets are disambiguated and refined such that *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 relevant approaches for instance matching in 70% of the cases, and in those cases we improved average F1, between precision and recall, by 10%.

Keywords: data integration, RDF interlinking, instance matching, linked data, entity recognition, entity search, collective inference, disambiguation.

1 Introduction

Instance matching [7] refers to the problem of determining whether two descriptions are about the same real-world entity. Also known as object consolidation, duplicate detection, record linkage, entity resolution or co-reference reconciliation, it represents a crucial task in data integration, entailing non-trivial problems that are actively studied in many research communities. Traditionally, research in this context was focused on the *single-domain setting*, where data come from the same or similar datasets. Recently, active research towards Web-scale integration can also be observed - largely due to the large increase in availability and importance of Web data. For instance, this problem is actively studied by Google researchers [22] as well as the large community of Semantic Web researchers due to the rapidly growing Linked Data Web (LDW)[4]. To date, the LDW contains hundreds of publicly available datasets, capturing billions of resource descriptions in RDF (Resource Description Framework). Creating links through matching RDF resources across datasets is the topmost goal actively pursued by projects such as Linking Open Data [2]. As opposed to the single-domain setting, the *Web setting* involves heterogeneous data associated with *varying schemas* that are created to capture different domains.

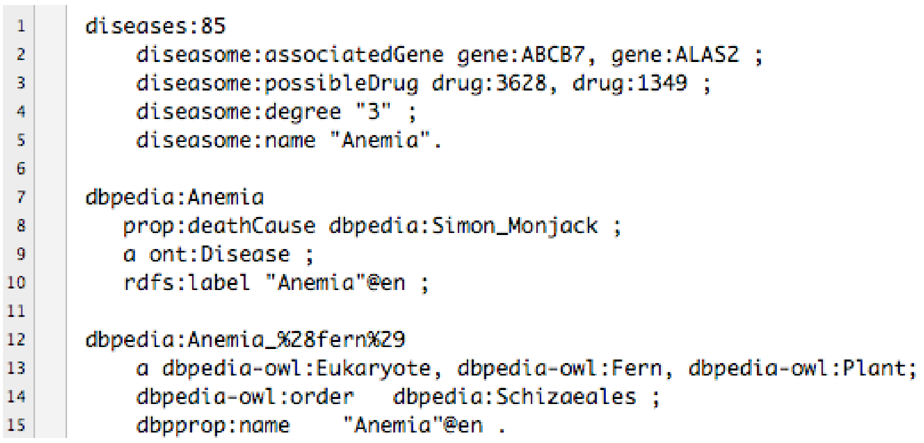
Basically, given descriptions of entities available as records in databases, RDF descriptions on the Web, etc., the instance matching task breaks down to the core problems of (1) finding a suitable representation (i.e., selecting attributes), (2) using this representation to match a source record against candidate records in the same or other target datasets, and (3) finally selecting the most similar ones (according to a threshold). For the single-domain setting, different solutions have been proposed to solve these individual sub-problems. There are *data blocking* techniques that based on simple representations of entities, can quickly identify candidate records [8]. Then, for more sophisticated and *effective matching*, there are different types of similarity measures [7, 3, 10], and different techniques for *learning* the right combination of attributes, the similarity measures as well as the similarity threshold to be used for computing and selecting the resulting matches [26].

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* for learning the right combination of attributes, similarity measures and thresholds. More importantly, instances to be matched 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 Web data integration.

Example. There are two descriptions of the `anemia` disease that were extracted from two different datasets. The first one is the Diseasome dataset, which specifically represents data from the Life Science domain. The second one

[4] http://linkeddata.org/

is DBPedia, a cross-domain encyclopedic type of dataset, which contains data extracted from Wikipedia. While the description from the Diseasome dataset 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 an existing blocking technique that compares instances simply by tokens, these instances can be identified to be candidate matches. However, this overlap is not enough to guarantee these instances refer to the same disease, because this blocking may yield other candidates, such as the third description of `anemia` in Fig. 1, line 12, which actually refers to a plant. Applying more sophisticated techniques to disambiguate these candidates is not possible in this setting because there are no common attributes that can be selected. Hence, attribute-specific learning and tuning of similarity measures and thresholds [7] do not apply.



```
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 .
```

Fig. 1. Examples of resources labeled "Anemia" in notation N3 (the prefixes are omited for matter of clarity)

Recently, a few proposals for data integration in the Web setting have been made. For instance, Google researchers actively pursue the concepts of *pay-as-you-go* [13] and *search-driven Web-scale integration* [22]. Initial work towards *schema-level integration* in the LDW setting has been reported [17]. However, we noted that the specific problem of instance matching in the Web setting with multiple domains and schemas is largely unsolved. To the best of our knowledge, the only work in this direction is a schema-agnostic blocking technique that simply extracts all tokens from entity descriptions to compute candidate matches [19]. However, as discussed in the previous example, blocking techniques like this one are meant for selecting candidates, while further processing is needed to refine them.

Contributions. This paper introduces SERIMI, an approach that focuses on the effective matching of candidate instances resulting from blocking [12]. It specifically addresses the mentioned challenges in the Web setting: It is completely unsupervised so that no training data are needed. 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 in the source dataset, SERIMI quickly finds candidate matches in the target dataset, computes the class in the target dataset which 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 rely on knowledge about the schema nor explicit correspondences between classes (thus, is *schema-agnostic*). We implemented our approach and performed experiments on large-scale real-world datasets extracted from the LDW. We show that through the disambiguation performed by SERIMI, results from token-based blocking can be substantially improved. We compare SERIMI with two preliminary works recently published, RiMOM [14] and ObjectCoref [13], and show that SERIMI outperformed these baselines in 70% of the cases, and in those cases we improved average F1, between precision and recall, by 16%.

Outline. This paper is organized as follows. After this introduction, we discuss related work in Section 2. In Section 3, we introduce the problem of disambiguation in instance matching. In Section 4, we elaborate on our class-based disambiguation method. Section 5 and 6 presents the experimental evaluation and results, and Section 7 concludes this paper.

2 Related Work

The problem of Web-scale instance matching has only been studied recently, while most of the work so far focused on the single domain context. We will now briefly discuss existing work along the main dimensions of attributes (or *features* in general), *similarity measures* and *matching techniques*. Also, we will present the main directions towards *Web-scale integration* and position our contributions along this line.

2.1 Matching Features

Instances are similar and thus, are considered candidate matches if their features are similar [9]. A great variety of features have been employed in order to solve this instance matching problem, while instances in different settings, may be represented as different types of database records or RDF resources. Traditionally, features are derived from flat attributes or structure information of instances. Structure in this sense, includes schema information, constraints and complex relationships (e.g. foreign key relationships between records in the database or relations between RDF resources in the RDF graph) [16, 20]. In the LDW context, where some resources are described through expressive formal ontologies, semantic information has also been employed. For instance, ObjectCoref [13] considers two resources as matches if their *discriminative attributes* are similar. The discriminativeness is inferred from attribute characteristics captured by the underlying ontologies (expressed using the Web Ontology Language, OWL), including `owl:InverseFunctionalProperty`, `owl:FunctionalProperty` and `owl:cardinality`.

2.2 Similarity Measures

Instance matching using flat features typically relies on string comparison techniques using different similarity metrics. Character-based metrics (e.g. Jaro, Q-grams) work well for detecting typographical errors. Token-based metrics (e.g., SoftTF-IDF, Jaccard) work well when features have many words and the arrangement of words cannot be captured by character-based metrics (e.g., "Michael Jackson" vs. "Jackson, Michael"). Document-based similarity metrics (e.g., cosine similarity) are employed when the number of tokens to be compared is large. Phonetic metrics (e.g., Soundex) can deal with features that are phonetically similar. Numeric metrics were designed to capture the nuances of numeric features. String-based metrics are not able to detect similarity between synonyms, nicknames, abbreviations and acronyms (e.g, "Big Apple" and "New York"). To address this, lexical semantic relatedness metrics [11, 4] were proposed. Semantic information used for that are from general knowledge sources (e.g., Wikipedia) or lexical sources (e.g., WordNet). Although there are many similarity metrics, there is no single one that applies in all cases [5]. Different features have different characteristics that demand different metrics. Some authors propose that the best way to approach this problem is by learning the right metrics for the given features, and by combining different metrics [3].

2.3 Matching Techniques

Orthogonal to features and metrics, different matching techniques have been proposed to address both the efficiency and effectiveness of instance matching. Data blocking techniques [12] aims to make it more efficient by reducing the number of unnecessary comparisons between records. Based on a feature that is distinctive and can be processed efficiently (also called Blocking Key Value, BKV), instances are partitioned into blocks such that potentially similar instances are placed in the same block. Each block can be considered as a set of candidate matches, which subsequently are disambiguated through a more sophisticated and effective method. Examples of blocking techniques include the sorted neighborhood approach [12] and canopy clustering [15]. So far, only one blocking technique has been explicitly designed to work in the heterogeneous Web setting, where the BKV is simply the set of all tokens that can be extracted from the instance data [19]. This approach differs from the single-domain techniques in that instead of using specific attributes, it simply uses tokens from all attributes. Thus, it can deal with multiple domains and schemas because instances to be compared do not have to share common attributes (i.e. their schemas are not assumed to be same or pre-aligned).

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 [24, 21, 18]). 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. As features, it models an instance as a vector of terms that are extracted from the structure formed by the instance and its neighbors. *Collective matching* represents the other kind of approaches [20]. It exploits the intuition that two instances are similar if their neighbors are similar. Similarity flooding [16] is a generic graph-matching algorithm that implements this intuition. This algorithm was adopted to the LDW setting to deal with ontology matching [25].

2.4 Instance matching on the Web

Recently, different directions towards Web-scale matching have been pursued. One prominent concept is pay-as-you-go data integration [6], which recognizes that in large-scale scenarios, it is not affordable to perform integration completely upfront but rather, it shall be considered as a continuous process that involves users, where integration results are incrementally obtained and refined as the system evolves. In particular, Google researchers have studied keyword search-based data integration, where matching tasks are carried out during user search activities [22]. Besides these general concepts, a few technical solutions have also been proposed to deal with schema and ontology matching [4] in the LDW context. However, besides the blocking technique [19] and the two prelimanriy works, ObjectCoref and RIMON, mentioned above, there exists no solution for the instance-matching problem, which specifically deal with multiple domains and schemas.

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. This may be due to the lack of alignment between schemas that can be computed upfront, or because correspondences between classes simply cannot be established. Without schema overlap, existing approaches [14, 13] that perform *direct matching* between the source and target instances based on some common attributes are not applicable. As opposed to direct matching, we propose a disambiguation technique, which involves only instances of the same target dataset. In this section, we introduce the data model and then, present the problem and the main ideas behind our solution.

3.1 Preliminary

Web data, including relational data, XML and RDF, can be conceived as graphs. For instance, tuples in the relational database setting correspond to graph nodes and foreign key relationships between them represent edges. RDF data consist of triples, which collectively form a graph. Closely resembling this RDF data model (omitting special RDF semantics and constructs such as blank nodes), 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, i.e. $V{=}U \cup L$.*

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 $IR : G \times 2^U \rightarrow G$ *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 $IR(G, \{s\})$. We will use the terms instance and instance representation interchangeably from now on. For simplicity, we only use the outgoing edges (s, p, o) of a resource s to form its representation $IR(G, \{s\})$. However, other types of representations that may include incoming edges, as well as more complex structures in the data (e.g. paths instead of edges), are applicable. 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 between instances to perform an additional step of class-based disambiguation. It is based on the observation that target matches for a collection of semantically related source instances are also semantically related among themselves, i.e. target matches are related when source instances are related. In particular, we consider the case where resources are semantically related in the sense that together, they form a class. Then, the intuition behind our approach is as follows: Given source instances 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 ***cite our TR*[dtr]**.

We note this class-based disambiguation resembles similarity flooding [16] in the sense that a class can be considered as a neighbor, and we gain more confidence that two instances match, when their class neighbors also match. Thus, our approach can be seen as a particular implementation of this framework. However, whereas previous works in this direction operate on one single graph, and employ neighbors that are explicitly given as nodes, our work involves multiples graphs, and computes the class representation of target instances on the fly.

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 $PH(S) = \{PH(s) | s \in S\}$ 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 be 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 used for determining whether a candidate s belong to the class of interest. Mainly, this work is about implementing this similarity relation 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 as discussed 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. Then, more effective matching is achieved through the second step of disambiguation, where the candidates in $PH(s)$ are refined.

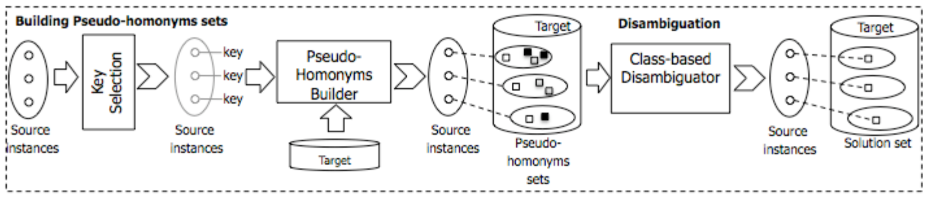


Fig. 2. The process of finding and disambiguating the pseudo-homonym sets given a set of source instances.

4.2 Building Pseudo-Homonym Sets

Here we will briefly describe the approach used in SERIMI to build the pseudo-homonyms sets. Our approach is similar to work on learning blocking keys [19]. Given the class of source instances (e.g. `Diseases`), we select its most discriminative attribute (e.g. label) as key, and use key values to find matches on the target dataset that make up the pseudo-homonyms set. The selection of attribute is based on entropy: Higher entropy attributes are more discriminative than attributes with lower entropy. We select the attributes with entropy $\omega \geq \omega_{threshold}$, where $\omega_{threshold}$ is simply the average entropy of all attributes in the source dataset. We consider attributes that have literal values with less than 200 characters. Given the attribute a and its literal values $O(a) = \{o|(s, a, o) \in G\}$, let p be the probability mass function of a, then the entropy $H(a)$ is as follows:

$$H(a) = - \sum_{o \in O(a)} p(o) log_2 p(o) \quad (1)$$

We use the selected attribute(s) to implement the instance representation of instances (both in source and target datasets). Target instances that match source instances based on these attributes are selected to compose the pseudo-homonym sets. In this work, we use Jaro-Winkler as similarity measure and the threshold is set to 0.7, a value commonly used in literature. Clearly, other measures and settings may be applied – the blocking and the learning of attributes and threshold in particular, is not focus on this work. We use blocking simply to obtain pseudo-homonym sets. Table 1 shows example results obtained for the `Diseasome` diseases 85, 379 and 502 using Diseasome as source and DBpedia as the target dataset.

Table 1. Pseudo-homonym sets for the source instances Anemia, Erythrocytosis and Hemophilia.

diseasome:85 Token: Anemia	diseasome:379 Token: Erythrocytosis	diseasome:502 Token: Hemophilia
dbpedia:Anemia	dbpedia:Erythrocytosis	dbpedia:Hemophilia
dbpedia:Anemia_fern	dbpedia:Familial_erythrocytosis	dbpedia:Hemophilia_A
dbpedia:Aplastic_anemia	1	dbpedia:Hemophilia_B
		dbpedia:Hemophilia_C
		dbpedia:Porphyric_Hemophilia

4.3 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:

$$RDS(A,B) = SetSim(P(A),P(B)) + SetSim(D(A),D(B)) + SetSim(O(A),O(B)) + SetSim(T(A),T(B)) \quad (2)$$

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. Based on Tversky's contrast model [23], we define $SetSim$[5] as follows:

$$SetSim(A,B) = |A \cap B| - \left(\frac{|A-B| + |B-A|}{2|A \cup B|}\right) \quad (3)$$

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 only the top-1, SERIMI may yield several matches.

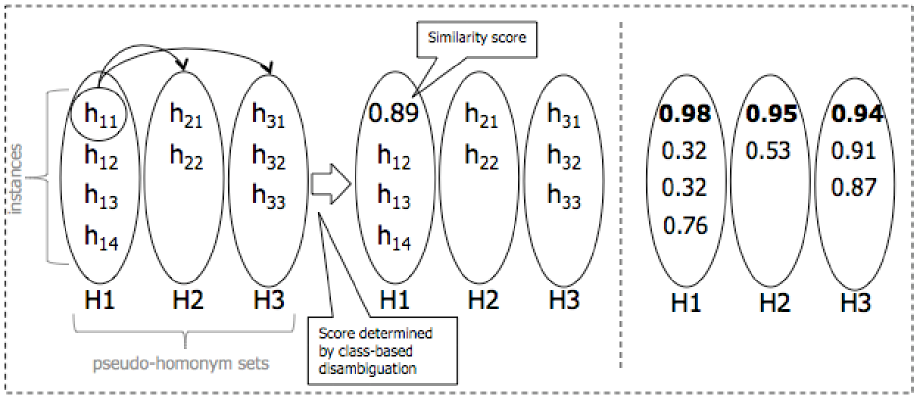


Fig. 3. A) Computing the similarity score for h_{11}. B) The similarity score for all resources.

The comparisons between t and the other pseudo-homonyms sets $PH(S)^-$ is captured by Equation 4 where we the individual score $RDS(\{r\}, PH(s))$ is

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

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 larger pseudo-homonyms sets contain more noise and smaller sets containing only a few representative instances are better representation of the class of interest.

$$URDS(t, PH(S)^-) = \sum_{PH(s') \in PH(S)^-} \frac{RDS(\{t\}, PH(s'))}{|PH(s')|} \tag{4}$$

We normalize the results of Equation 4 by the maximum score among all instances as

$$CRDS(t, PH(s), PH(S)^-) = \frac{URDS(t, PH(S)^-)}{MaxScore(PH(s), PH(S)^-)} \tag{5}$$

where $MaxScore(PH(s), PH(S)^-) = MAX\{URDS(t', PH(S)^-)|t' \in PH(s) \in PH(S)\}$. 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. It has been shown that learning this threshold value automatically is possible by applying a score distribution function [1]. In this work, we simply consider this as a parameter. 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.

4.4 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 the precision of matches is not affect 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 pseudo-homonym set (a set with one single 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. In the experiment, we add a (the size of the chunk) singleton sets to $PH(S)$ in every iteration, where we added, in total, a maximum of μ pseudo-homonym sets.

5 Experimental Evaluation

In this section we describe the employed collections, evaluation metrics and baselines applied in our evaluations. Our evaluation is based on the instance-matching track of the Ontology Alignment Evaluation Initiative (OAEI 2010). This track focuses on evaluating the effectiveness of instance-matching approaches over Web Data, which is exactly the goal of the evaluation here.

5.1 Experiment Setting

Collections. We used the life science (LS) collection (which includes DBPedia,[6] Sider,[7] Drugbank,[8] LinkedCT,[9] Dailymed[10] TCM,[11] and Diseasome[12]) and the Person-Restaurant (PR) collection provided by this benchmark.

Table 2 shows the main statistics about these datasets (the number of triples, and the number of reference mappings that make up the ground truth). These datasets capture several cases of ambiguity, which make instance matching hard. For example, two distinct people called John White with different birth date. We evaluated pairs of datasets that were also used by the other systems that participated in this benchmark. The Person-Restaurant collection contains 3 pairs of datasets. Two of these pairs describe people and the other pair describes restaurants. The other datasets make up 11 pairs.

We have loaded the datasets into an open-source instance of Virtuoso Universal server[13] installed on a local workstation, where around 2GB of data were loaded. An exception was the DBpedia dataset, which we accessed online via its SPARQL endpoint.

Table 2. Number of instances per dataset.

Dataset	Instances	Dataset	Instances	Dataset	Instances	Dataset	Instances
Diseasome	8150	Drugbank	19692	Dailymed	10004	Sider	2673
Tcm	24579	Person21	2400	Person22	800	Person11	1000
Person12	2000	Restaurant1	339	Restaurant2	2256	DBPedia	> 1000000

Evaluation metrics and baseline. In order to evaluate the effectiveness of the proposed instance matching approach, we used precision and recall (and F1 also called F-measure, which considers both). We considered as true positives the

[6] http://dbpedia.org/About
[7] http://www4.wiwiss.fu-berlin.de/sider/
[8] http://www4.wiwiss.fu-berlin.de/drugbank/
[9] http://data.linkedct.org/
[10] http://www4.wiwiss.fu-berlin.de/dailymed/
[11] http://code.google.com/p/junsbriefcase/wiki/RDFTCMData
[12] http://www4.wiwiss.fu-berlin.de/diseasome/
[13] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/

Table 3. Number of reference mapping for pair of datasets evaluated.

Dataset Pair	Mapping	Dataset Pair	Mapping
Sider→ DBPedia	1509	Dailymed→DBpedia	2549
Sider→Dailymed	1634	Dailymed→LinkedCT	27729
Sider→Diseasome	173	Dailymed→TCM	33
Sider→DrugBank	1140	Dailymed→Sider	1592
Sider→TCM	171	Drugbank→Sider	284
Diseasome→Sider	238	Person11→Person12	500
Restaurant1→Restaurant2	112	Person21→Person22	400

reference mapping in the ground truth. False positives are the mappings found by SERIMI that do not exist in the ground truth.

We used two baselines in our experiments: RiMOM and ObjectCoref. As discussed, these two preliminary work are the two main solutions for effective instance matching in this heterogeneous setting. Using the same datasets and reference mappings, we aim to provide a fair and direct comparison. In addition, we investigated how SERIMI performs using different settings for the parameters δ and k.

5.2 Experiment Results

Fig. 4 and Fig. 5 shows SERIMI's performance when we vary δ. We observed that the standard deviation of precision and recall is close to zero in the cases where the pseudo-homonym sets are small, e.g. have cardinality equal to 1 (Drugbank-Sider). The parameters δ and k have no effect in these cases because the same (number of) instances are selected (e.g. only one) as results. Otherwise, performances vary because differences in δ (and also differences in k) lead to a different selection of instances. This seems to have a clear impact on precision and recall. The use of the δ_m, an automatically computed threshold as discussed in Section 4.3, performs relatively well on average. Therefore, for all other experiments shown in this paper, we used $\delta = \delta_m$.

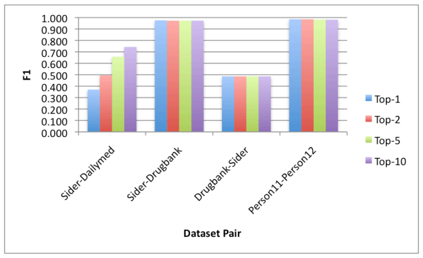


Fig. 4. Top-k F-measure

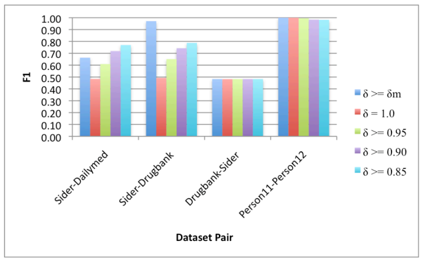


Fig. 5. $\delta_{threshold}$ F-measure

Table 4. It shows SERIMI, RiMOM and ObjectCoref the precision and recall for all dataset pairs. ObjectCoref's results are not available for all pairs of datasets.

Dataset Pair / Approaches	Sider DBPedia		Sider Dailymed		Sider Diseasome		Sider DrugBank		Sider TCM	
	P	R	P	R	P	R	P	R	P	R
SERIMI	0.479	**0.617**	**0.780**	0.567	**0.922**	0.831	**0.973**	**0.969**	**0.970**	**0.976**
RiMOM	**0.717**	0.482	0.567	**0.706**	0.315	**0.837**	0.961	0.342	0.778	0.812
ObjectCoref	-	-	-	-	-	-	-	-	-	-

Dataset Pair / Approaches	Dailymed DBpedia		Dailymed LinkedCT		Dailymed TCM		Dailymed Sider		Drugbank Sider	
	P	R	P	R	P	R	P	R	P	R
SERIMI	**0.611**	**0.330**	**0.234**	0.051	**0.233**	**0.911**	0.541	0.868	**0.328**	0.915
RiMOM	0.246	0.293	0.070	0.235	0.159	0.535	**0.567**	0.706	-	-
ObjectCoref	-	-	-	-	-	-	0.548	**0.999**	0.302	**0.996**

Dataset Pair / Approaches /	Diseasome Sider		Person11 Person12		Person21 Person22		Restaurant1 Restaurant2	
	P	R	P	R	P	R	P	R
SERIMI	0.833	**0.901**	**1.000**	**1.000**	0.557	0.385	0.768	0.768
RiMOM	-	-	1.000	1.000	0.952	**0.990**	0.860	0.768
ObjectCoref	0.837	0.668	1.000	0.998	**0.989**	0.887	**0.989**	**0.804**

As we can see in Table 4, SERIMI outperforms the baselines in 70% of the cases, and in those cases we substantially improved F1 by 10% on average. In the cases of Sider-Diseasome and Sider-Drugbank, SERIMI achieved a gain of 42% and 46% in F-measure, respectively. In both cases, SERIMI clearly was able to disambiguate more matches than the baselines. In the case of DailyMed-DBpedia, where the pseudo-homonym sets had in average 27 ambiguous instances, SERIMI added a gain of 17% in precision compared to the baselines. In this case, considering that the blocking strategy was the main component responsible for the increase in the recall, it shows a significant improvement in the precision caused by the class-based disambiguation. For example, there are many 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 the ambiguity, and 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 in the ambiguous pseudo-homonym set the correct instance, labeled *Magnesium*, that refer to the class drugs. An interesting conclusion about this case is that, in general, SERIMI is very effective to disambiguate pseudo-homonyms sets that contains instances that belong to multiple different classes, as in the previous example.

The poor performance in the Person21-Person22 pair is due to the nature of the data. These datasets where built by adding spelling mistakes to the properties and literals values of their original datasets. The blocking strategy used by SERIMI is not specifically tuned to deal with this problem of misspellings, and thus, was not able to build the pseudo-homonyms sets properly, resulting in low recall. In contrary, ObjectCoref and RiMOM use more specific functions for dealing with the similarity of misspelled data. In order to further improve the performance of SERIMI, more advanced similarity matching functions may be considered to yield better result candidates, which can then be refined by SERIMI's disambiguation.

We further noticed that the CRDS function does not work properly for this pair Person21-Person22 because the instances in these datasets are exactly of the same class, e.g. two instances that have the label `John White`. We note that SERIMI is rather designed for the heterogeneous case where instances to be matched belong to multiple domain. Matching instances in this single domain (same class) setting is indeed "problematic" because the idea behind SERIMI is to use class information of disambiguation, i.e. to filter out candidates that do not belong to the class of interest. However, because all candidates belong to the same class, there is not enough information for SERIMI to distinguish and disambiguate instances. Clearly, there exist a wide range of techniques for instance matching in this single domain setting and they shall be used instead of SERIMI. We consider SERIMI is a complementary tool that specifically targets the heterogeneous setting. However, we observe that extending SERIMI to capture not only the instances, but also their neighbors as features, can provide additional class information (i.e. the classes of the neighbors) that helps the disambiguation even in this single domain setting.

As a final observation, we observed that poor performances (e.g. for some cases in Dailymed-DBPedia, Dailymed-Linkedct, Dailymed-TCM, Drugbank-Sider and Sider-Dailymed) are due to errors in the reference mapping. Below we list some examples:

Sider-Dailymed:
Source resource:
http://www4.wiwiss.fu-berlin.de/sider/resource/drugs/151165
Current reference alignment:
http://www4.wiwiss.fu-berlin.de/dailymed/resource/drugs/1050
Missing alignment:
http://www4.wiwiss.fu-berlin.de/dailymed/resource/drugs/1618
Dailymed-DBPedia:
Source resource:
http://www4.wiwiss.fu-berlin.de/dailymed/resource/organization/Allergan,_Inc.
http://www4.wiwiss.fu-berlin.de/dailymed/resource/organization/AstraZeneca
http://www4.wiwiss.fu-berlin.de/dailymed/resource/organization/B._Braun
Missing alignment:
http://dbpedia.org/resource/Allergan
http://dbpedia.org/resource/AstraZeneca
http://dbpedia.org/resource/B._Braun_Melsungen

Although this reference mapping is thus not ideal, it is the best known ground truth provided by the community. This quality problem affects all systems equally. Hence, the comparisons provided here still yields fair conclusions.

Yet, this observation made here suggests that for a better understanding of instance matching solutions, some more work is needed in the future for establishing a higher quality benchmark.

In conclusion, we observed a considerable improvement of accuracy when we apply the proposed class-based disambiguation to results obtained from a simple blocking procedure. We expect further improvement when more sophisticated techniques for blocking and computing results are employed. Our approach is specially recommended for cases where there are few overlaps between the schemas of the instances being matched. For single domain matching tasks (i.e. perfect overlap), we recommend existing works, and will integrated them into our matching procedure as future work. SERIMI's implementation is open source and available for download at GitHub [14].

6 Conclusions

We investigate the instance matching problem in the large and heterogeneous environment of the Web, where data come from different domains and are associated with different schemas. We proposed SERIMI as a completely unsupervised schema-agnostic approach that focuses on the effective matching of candidate instances (resulting from blocking). Our approach is able to refine the ambiguous matches provided by existing blocking techniques. We evaluated the accuracy of SERIMI based on experiments on large-scale real-world datasets. We outperformed two baseline 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). As future work, we consider the use of more advanced blocking strategy to further improve the accuracy of candidates that are refined by SERIMI. Also, we will integrate the ideas of single-domain matching into the overall solution so that SERIMI can perform well both in the traditional single-domain and the heterogenous multiple-domain setting.

References

1. Arampatzis, A., Robertson, S.: Modeling score distributions in information retrieval. Information Retrieval 14, 26–46 (Aug 2010), http://www.springerlink.com/index/10.1007/s10791-010-9145-5
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 5, 1–22 (2009), http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jswis.2009081901
3. Branting, L.K.: A comparative evaluation of name-matching algorithms. p. 224. ACM Press (2003), http://portal.acm.org/citation.cfm?doid=1047788.1047837

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

4. Budanitsky, A., Hirst, G.: Evaluating WordNet-based measures of lexical semantic relatedness. Computational Linguistics 32, 13–47 (Mar 2006), http://www.mitpressjournals.org/doi/abs/10.1162/coli.2006.32.1.13
5. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string metrics for matching names and records (Aug 2003), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.9007
6. Das Sarma, A., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. p. 861. ACM Press (2008), http://portal.acm.org/citation.cfm?doid=1376616.1376702
7. Dorneles, C.F., Gonalves, R., Santos Mello, R.: Approximate data instance matching: a survey. Knowledge and Information Systems 27, 1–21 (Apr 2010), http://www.springerlink.com/index/10.1007/s10115-010-0285-0
8. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. on Knowl. and Data Eng. 19(1), 116 (Jan 2007), http://dx.doi.org/10.1109/TKDE.2007.9
9. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association 64(328), pp. 1183–1210 (1969), http://www.jstor.org/stable/2286061
10. Hadjieleftheriou, M.: Approximate string processing. Foundations and Trends in Databases 2, 267–402 (2009), http://www.nowpublishers.com/product.aspx?product=DBS&doi=1900000010
11. Han, X., Zhao, J.: Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. p. 5059. ACL '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010), http://dl.acm.org/citation.cfm?id=1858681.1858687
12. Hernndez, M.A., Stolfo, S.J.: The merge/purge problem for large databases. ACM SIGMOD Record 24, 127–138 (May 1995), http://portal.acm.org/citation.cfm?doid=568271.223807
13. Hu, W., Qu, Y., Sun, X.: Bootstrapping object coreferencing on the semantic web. Journal of Computer Science and Technology 26, 663–675 (Jul 2011), http://www.springerlink.com/index/10.1007/s11390-011-1166-z
14. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: a dynamic multistrategy ontology alignment framework. IEEE Transactions on Knowledge and Data Engineering 21, 1218–1232 (Aug 2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4633358
15. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. pp. 169–178. ACM Press (2000), http://portal.acm.org/citation.cfm?doid=347090.347123
16. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering. p. 117. ICDE '02, IEEE Computer Society, Washington, DC, USA (2002), http://dl.acm.org/citation.cfm?id=876875.879024
17. Nikolov, A., Uren, V., Motta, E., Roeck, A.: Overcoming schema heterogeneity between linked semantic repositories to improve coreference resolution. In: Gmez-Prez, A., Yu, Y., Ding, Y. (eds.) The Semantic Web, vol. 5926, pp. 332–346. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), http://www.springerlink.com/index/10.1007/978-3-642-10871-6_23

18. Niu, X., Sun, X., Wang, H., Rong, S., Qi, G., Yu, Y.: Zhishi.me: weaving chinese linking open data. In: Proceedings of the 10th international conference on The semantic web - Volume Part II. pp. 205–220. ISWC'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2063076.2063091
19. Papadakis, G., Nejdl, W.: Efficient entity resolution methods for heterogeneous information spaces. pp. 304–307. IEEE (Apr 2011), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5767671
20. Shvaiko, P., Euzenat, J.: A survey of Schema-Based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV, vol. 3730, pp. 146–171. Springer Berlin Heidelberg, Berlin, Heidelberg (2005), http://www.springerlink.com/index/10.1007/11603412_5
21. Song, D., Heflin, J.: Automatically generating data linkages using a domain-independent candidate selection approach. In: Proceedings of the 10th international conference on The semantic web - Volume Part I. pp. 649–664. ISWC'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2063016.2063058
22. Talukdar, P.P., Ives, Z.G., Pereira, F.: Automatically incorporating new sources in keyword search-based data integration. In: SIGMOD Conference. pp. 387–398 (2010)
23. Tversky, A.: Features of similarity. Psychological Review 84, 327–352 (1977), http://content.apa.org/journals/rev/84/4/327
24. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009, vol. 5823, pp. 650–665. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), http://www.springerlink.com/index/10.1007/978-3-642-04930-9_41
25. Wang, Y., Liu, W., Bell, D.A.: A structure-based similarity spreading approach for ontology matching. In: Proceedings of the 4th international conference on Scalable uncertainty management. p. 361374. SUM'10, Springer-Verlag, Berlin, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1926791.1926827
26. Xiao, C., Wang, W., Lin, X., Shang, H.: Top-k set similarity joins. pp. 916–927. IEEE (Mar 2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4812465