A Two-step Blocking Scheme Learner for Scalable Link Discovery

Mayank Kejriwal and Daniel P. Miranker

University of Texas at Austin
{kejriwal,miranker}@cs.utexas.edu

Abstract. A two-step procedure for learning a link-discovery blocking scheme is presented. Link discovery is the problem of linking entities between two or more datasets. Identifying *owl:sameAs* links is an important, special case. A blocking scheme is a one-to-many mapping from entities to blocks. Blocking methods avoid $O(n^2)$ comparisons by clustering entities into blocks, and limiting the evaluation of link specifications to entity pairs within blocks. Current link-discovery blocking methods use blocking schemes tailored for *owl:sameAs* links or that rely on assumptions about the underlying link specifications. The presented framework learns blocking schemes for arbitrary link specifications. The first step of the algorithm is unsupervised and performs *dataset mapping* between a pair of dataset collections. The second supervised step *learns* blocking schemes on structurally heterogeneous dataset pairs. Application to RDF is accomplished by representing the RDF dataset in *property table* form. The method is empirically evaluated on four real-world test collections ranging over various domains and tasks.

Keywords: Heterogeneous Blocking, Instance Matching, Link Discovery

1 Introduction

With the advent of Linked Data, discovering links between entities has emerged as an active area of research [7]. Given a link specification, a naive approach would discover links by conducting $O(n^2)$ comparisons on the set of n entities. In the *Entity Resolution* (ER) community, a preprocessing technique called *blocking* mitigates full pairwise comparisons by clustering entities into blocks. Only entities within blocks are paired and compared [3]. Blocking is critical in data integration systems [5],[3].

Blocking methods require a *blocking scheme* to cluster entities. Advanced methods have been proposed to use a given blocking scheme effectively; relatively fewer works address the learning of blocking schemes. Even within ER, blocking scheme learners (BSLs) have met practical success only recently [2],[14],[10]. In the Semantic Web, the problem has received attention as scalably discovering *owl:sameAs* links [19]. ER is an important, but special, case of the *link discovery* problem, where the underlying link specification can be arbitrary. Such specifications can be learned, but brute-force applications would still be $O(n^2)$. Current

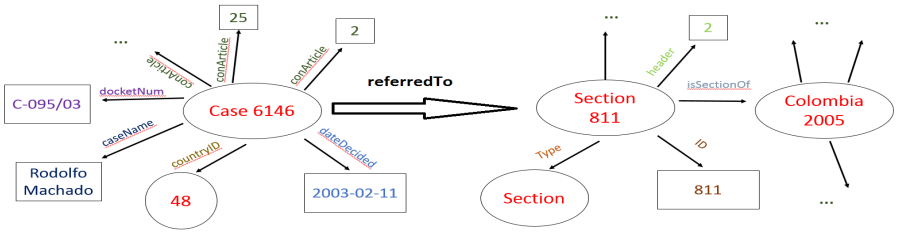


Fig. 1. Cases decided in Colombia must be *linked* to relevant sections of the constitution used in deciding that case. Only the single number *2* is relevant here for linking.

link discovery systems aim to be efficient by using token-based pre-clustering or metric space assumptions [7],[15].

Learning link-discovery blocking schemes for arbitrary underlying links remains unaddressed. Because the link can be arbitrary, a training corpus is required. Consider the example in Figure 1. Given a small number of such examples, the proposed BSL *adaptively* learns a scheme that covers true positives while reducing full quadratic cost, without relying on the formal link specification itself. Note that the learned blocking scheme is different from a learned link specification. In this paper, we exclusively address blocking.

In the Big Data era, *scalability*, *automation* and *heterogeneity* are essential components of systems and hence, practical requirements for real-world link discovery. Scalability is addressed by blocking, but current work assumes that the *dataset* pairs between which entities are to be linked are provided. In other words, datasets A and B are[1] input to the pipeline, and entities in A need to be linked to entities in B. Investigations in some important real-world domains show that pairs of dataset *collections* also need to undergo linking. Each collection is a *set* of datasets. An example is government data. Recent government efforts have led to release of public data as batches of files, as one of our real-world test sets demonstrates. Thus, *two* scalability issues are identified: at the collection level, and at the dataset level. That is, datasets in one collection first need to be mapped to datasets in the second collection, after which a blocking scheme is learned (and later, applied) on each mapped dataset pair.

Automation implies that human intervention needs to be kept to a minimum. In practice, this means that methods need to rely on fewer training examples. Finally, a heterogeneity issue arises if datasets in the collections are in two different data models, such as RDF and tabular.

The proposed BSL addresses these challenges in a combined setting. It takes as input two collections of datasets, with each collection an arbitrary mix of RDF and tabular datasets. The first step of the BSL performs unsupervised *dataset mapping* by relying on document similarity and efficient solutions to the Hungarian algorithm [9]. Each chosen dataset pair is input to the second step, which learns a link-discovery blocking scheme given a *constant* size training cor-

[1] If A and B are the same dataset, the problem is commonly denoted *deduplication*.

pus that does not require growing with the dataset. RDF datasets are reconciled with tabular datasets by representing them as *property tables* [22]. The problem thus reduces to learning schemes on tabular datasets with different schemas. Elmagarmid et al. refer to this as the *structural heterogeneity* problem [5]. To robustly deal with small, constant training sets, the BSL uses *bagging* [20]. To the best of our knowledge, this is the first paper that uses bagging, dataset mapping and property tables to address automation, scalability and heterogeneity respectively in the link-discovery blocking context.

The rest of this paper is structured as follows. Section 2 describes the related work in this area. Section 3 describes the property table representation and the BSL in detail. Section 4 describe the experimental results, and the paper concludes in Section 5.

2 Related Work

Link Discovery has been researched actively since the original Silk paper [21], which currently uses genetic programming to learn links [7]. Since we discuss learning of *blocking schemes*, most link specification learners are compatible, not competitive, with the proposed system. A full pipeline that can perform data fusion is RDF-AI [18]. We note that *active learning* techniques have been proposed to address the automation issue [16]; in this paper, we show a complementary solution using *bagging*.

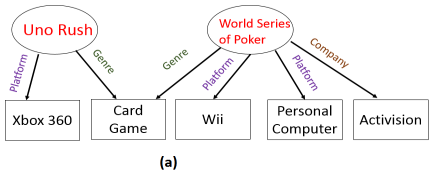
Blocking has been extensively studied in the record linkage community [5], with a comprehensive survey by Christen [3]. Initial BSLs were supervised [2],[14]. Recently, an unsupervised feature selection based BSL was proposed by us, but assumed only *owl:sameAs* links and did not use bagging [10]. Token-based clustering has also been applied to the problem [13], together with Locality Sensitive Hashing (LSH) techniques [11]. However, LSH is usually applicable only to select distance measures like Jaccard or cosine. As such, it is more popularly applied to ontology matching [4],[6]. Other Semantic Web efforts for *owl:sameAs* include the approach by Song and Heflin [19]. Ma et al. proposed a system based on type semantics exclusively for ER on two individual datasets [12].

Multiple techniques for *using* blocking schemes have been investigated in the Semantic Web community [17]. An example (used in the Silk framework) is MultiBlock [8]. Another effort that assumes *metric* spaces is LIMES [15]. Finally, an advanced survey of bagging can be found in the work by Verikas et al. [20].

3 Algorithm

In this section, the two steps of the overall BSL in Figure 3 are described. Note that *dataset mapping* is an *optional*[2] step, but the second step (the core learner) is essential. As a preliminary, the property table representation is also summarized.

[2] Albeit empirically advantageous, when applied, as Section 4 will show.



Subject	Genre	Platform	Company
World Series of Poker	Card game	Wii; Personal Computer	Activision
Uno Rush	Card game	Xbox 360	*null*

(b)

Fig. 2. The property table representation. For subjects 'missing' a property, the reserved keyword *null* is entered. *;* is a reserved delimiter allowing fields to be *sets*.

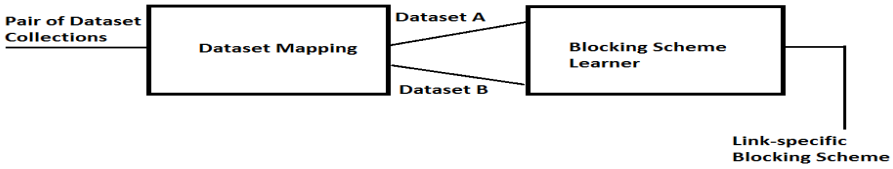


Fig. 3. The overall framework proposed in the paper.

3.1 Property Table Representation

Property tables were first proposed as *physical* data structures to efficiently implement triple stores [22]. This is the first application using them for link-discovery blocking. Figure 2 shows an example. Property tables reduce the problem of linking RDF and tabular datasets (and also RDF-RDF linkage) to tabular structural heterogeneity, that is, tables with different schemas. The rest of the paper assumes property table representation of RDF.

3.2 Dataset Mapping

The pseudocode for dataset mapping is given in Algorithm 1. The inputs to the algorithm are the dataset collections $\mathcal{R}$ and $\mathcal{S}$ and a boolean *confidence* function that is subsequently described. We define a dataset collection as a *set* of *independently released* datasets. Without loss of generality, assume that $|\mathcal{R}| \leq |\mathcal{S}|$. The output desired is a *confident* mapping $\mathcal{M} \subseteq \mathcal{R} \times \mathcal{S}$.

Algorithm 1 represents each dataset in each collection as a *term frequency* (TF) vector. The TF vector for each dataset is constructed by assigning a unique position to each distinct token and recording the count of that token in the dataset. Each TF vector is normalized by dividing each element by the total count of the respective token in the corresponding collection. A normalized TF vector is different from a TFIDF[3] vector.

[3] The TFIDF vector is constructed by dividing each element of a TF vector by the *number of datasets* in the corresponding collection in which the token occurred at least once (in lines 6 and 7) rather than the *total count of the token* in that collection.

A matrix $\boldsymbol{Q}$ is initialized with $\boldsymbol{Q}[i][j]$ containing the dot product of normalized TF vectors R_i and S_j. Once the matrix is constructed, the *max Hungarian* algorithm is invoked on the matrix, which has at least as many columns as rows, by the assumption above [9]. The algorithm must *assign* each row to some column, such that the sums of corresponding matrix entries are maximized. The problem is also equivalent to maximum weighted bipartite graph matching.

The confidence of each mapping is evaluated by $\mathcal{C}$. As an example of a confidence function, suppose the function returns *True* for a returned mapping (i, j) iff $\boldsymbol{Q}[i][j]$ is the *dominating* score, that is, greater than every score in its constituent row and column. Intuitively, this means that the mapping is not only the best possible, but also non-conflicting. In some sense, this assumes an aggressive strategy against false positives. Other[4] strategies can be formulated for other requirements; we leave these for future work.

Assuming the dominating strategy *a priori*, the Hungarian algorithm can be modified to terminate in linear time (in $\mathcal{R}$ and $\mathcal{S}$), otherwise it is cubic in the collection size [9]. Empirically, a reasonable confidence strategy would lead to savings if the total number of records is far greater than the number of datasets. For large collections, preferrable strategies should have theoretical guarantees, like the dominating strategy. We observed dataset mapping to achieve near-instantaneous runtime, even with a standard Hungarian implementation.

Intuitively, dataset mapping is expected to be a well-performing heuristic because constituent datasets are *independently* released. Algorithms like LIMES, Canopy Clustering and unsupervised methods benefit because they can cluster entities in isolated dataset pairs, rather than all the entities in the collection. With correct mapping, both quality and scalability are expected to improve. Experimentally, the gains are demonstrated in Section 4.

3.3 Heterogeneous Blocking Scheme Learner

Given two structurally heterogeneous tabular sources, the goal of the second step is supervised learning of a *heterogeneous* blocking scheme. In earlier works, *DNF blocking schemes* were found to be fairly representative and learned using set-covering algorithms [2],[14]. In recent work, we showed that a *feature selection* technique outperformed state-of-the-art DNF BSLs [10]. To summarize the work briefly, given training sets D and N containing duplicate and non-duplicate tuple pairs respectively, each pair is first converted into a vector with $O(m_1 m_2)$ *binary* features, where m_1, m_2 is the number of attributes in datasets R_1,R_2 respectively. Thus, two sets F_D and F_N containing labeled feature vectors are obtained. A set of features must now be chosen such that a minimum fraction ϵ of positives are covered, and with no individual feature covering more than a fraction η of negatives. For further details on these parameters and the feature conversion and selection process, we refer the reader to the original work [10].

Note that the originally proposed algorithm had no concept of bagging and assumed the training set was representative enough. Algorithm 2 shows how bag-

[4] For example, using a threshold to map multiple datasets to each other.

Algorithm 1 Perform dataset mapping.

Input: Dataset Collections $\mathcal{R}$ and $\mathcal{S}$, boolean confidence function $\mathcal{C}$
Output: A mapping $\mathcal{M}$ between $\mathcal{R}$ and $\mathcal{S}$

1. **for all** datasets $R_i \in \mathcal{R}$ **do**
   $\overrightarrow{T_i^{\mathcal{R}}}$:= Term Frequency vector of terms in R_i
2. **end for**
3. **for all** datasets $S_i \in \mathcal{S}$ **do**
   $\overrightarrow{T_i^{\mathcal{S}}}$:= Term Frequency vector of terms in S_i
4. **end for**
5. Construct vectors $\overrightarrow{T^{\mathcal{R},\mathcal{S}}}$, with j^{th} element $\overrightarrow{T^{\mathcal{R},\mathcal{S}}}[j] := \Sigma_i \overrightarrow{T_i^{\mathcal{R},\mathcal{S}}}[j]$
6. Normalize each $\overrightarrow{T_i^{\mathcal{R}}}$ by applying once, $\forall j, \overrightarrow{T_i^{\mathcal{R}}}[j] := \overrightarrow{T_i^{\mathcal{R}}}[j] / \overrightarrow{T^{\mathcal{R}}}[j]$
7. Normalize each $\overrightarrow{T_i^{\mathcal{S}}}$ by applying once, $\forall j, \overrightarrow{T_i^{\mathcal{S}}}[j] := \overrightarrow{T_i^{\mathcal{S}}}[j] / \overrightarrow{T^{\mathcal{S}}}[j]$
8. Initialize empty matrix $\boldsymbol{Q}$ with $|\mathcal{R}|$ rows and $|\mathcal{S}|$ columns
9. **for all** $i \in 1 \ldots |\mathcal{R}|$ **do**
   for all $j \in 1 \ldots |\mathcal{S}|$ **do**
   $\boldsymbol{Q}[i][j] := \overrightarrow{T_i^{\mathcal{R}}}.\overrightarrow{T_j^{\mathcal{S}}}$
   end for
10. **end for**
11. Let $\mathcal{M}$ be the results of running *max* Hungarian algorithm on $\boldsymbol{Q}$
12. **for all** $(i, j) \in \mathcal{M}$ **do**
   if applying $\mathcal{C}$ on $score(R_i, S_j)$ yields *False* **then**
   Remove (R_i, S_j) from $\mathcal{M}$
   end if
13. **end for**
14. **return** $\mathcal{M}$

ging can be incorporated, to work with small[5] training sets. Bagging parameters τ, β are now also input. β specifies the number of bagging iterations and τ is the sampling rate for bagging. In each bagging iteration, a fraction τ of the overall training sample is chosen to undergo feature selection by calling *FisherDisjunctive* (Algorithm 3 in [10]). A feature is technically a *specific blocking predicate* (SBP) e.g. *CommonToken(Last Name, Name)*. Intuitively, the SBP implies that two entities (from different datasets) with a common token in their respective *Last Name* and *Name* field values *share* a block. Features (or SBPs) chosen in each bagging iteration are added to $\mathcal{B}'$. The final DNF blocking scheme $\mathcal{B}$ is *heterogeneous* precisely because it accommodates two *different* schemas, as the SBP example above shows.

In some cases, training examples might not be available at all. If the task is learning *owl:sameAs* links, the automatic training set generator in our original work can be used to generate noisy training samples [10]. The rest of the procedure remains the same. We evaluate this scenario in one of our test suites.

[5] Small training sets affects learning algorithm quality, but compensate well for $O(m_1 m_2)$ feature-vector dimensionality. Bagging allows controlled compromise between scalability and quality.

Algorithm 2 Learn Link-Specific Blocking Scheme

Input : Positive feature-vectors set F_D, negative feature-vectors set F_N, coverage parameter ϵ, pruning parameter η, bagging iterations β, sampling size τ
Output : Blocking scheme $\mathcal{B}$

1. Initialize $\mathcal{B}' := \phi$
2. **for all** $iter = 1 \ldots \beta$ **do**
   Randomly sample (with replacement) $\tau|F_D|$ and $\tau|F_N|$ vectors from F_D and F_N, insert into new sets F'_N and F'_D respectively
   $\mathcal{B}' := \mathcal{B}' \cup FisherDisjunctive(F'_D, F'_N, \epsilon, \eta)$
3. **end for**
4. Output disjunction of elements in $\mathcal{B}'$ as $\mathcal{B}$

Table 1. Test dataset details. The notation, where applicable, is (first collection)/(second collection) or (first collection)×(second collection)

Collection	**Number of datasets**	**Task**	**Total entity pairs**	**True positive pairs**	**Data model**
Case Law/Constitute (Colombia)	1/2	non-ER	1204 × 2220 ≈ 2.67 million	5577	RDF/RDF
Case Law/Constitute (Venezuela)	1/2	non-ER	1503 × 1601 ≈ 2.4 million	555	RDF/RDF
JCT/Treasury	5/5	non-ER	1135 × 845 ≈ 1 million	24,227	Tab./Tab.
Dbpedia/vgchartz	1/1	ER	16740 × 20000 = 334.8 million	10,000	RDF/Tab.

4 Experiments

In this section, the algorithm is experimentally evaluated. Datasets, metrics and baseline are first described, followed by a set of results and a discussion.

4.1 Datasets

The algorithm is evaluated on four real-world dataset collections over three different domains, described in Table 1. In the first test set, the first collection consists of a single RDF dataset describing court cases decided in Colombia, along with various properties of those cases. The second collection has two RDF datasets, only one of which is relevant for linkage. This dataset describes article numbers (as literal object values) in the Colombia constitution[6]. The task is to predict links between the cases in the first collection and the articles in the second collection used to decide the case. An example was shown in Figure 1. The second test set is similar, but for Venezuela. The third test set consists of ten US

[6] `constituteproject.org`

government estimated budget datasets from 2009 to 2013, released separately by the Treasury department and the Joint Committee on Taxation. All datasets in this collection were published in tabular form, but the two collections are structurally heterogeneous. The goal is to link entities (describing a particular budget allocation) that share the same budget function (such as *health*) in the *same* year. These three test cases are proper dataset collections, given at least one collection contains more than one dataset. Other such collections can also be observed on the respective website[7].

The fourth test set contains collections derived from the *video games* domain and differs from the other three test sets in three important respects. First, the dataset mapping step is not applicable, since each collection only contains one dataset. Note that the datasets in this test set are large compared with the other test sets. Second, the first dataset is RDF and was queried from DBpedia[8] while the second dataset is tabular and from a popular charting website[9]. Finally, the noisy training set generator can be used, given the link is *owl:sameAs*. We have collected all publicly available test cases on a single portal[10], along with other implementation details such as the *features* (or SBPs) used in the experiments.

4.2 Metrics

We adopted two metrics, Pairs Completeness (PC) and Reduction Ratio (RR), from the blocking literature [3]. PC measures *recall* or *effectiveness* in the blocking setting; specifically, the ratio of true positives that have fallen within the block and the total number of true positives. RR is the percentage of comparisons that have been avoided compared to full quadratic cost and represents *efficiency*. For example, an RR of 99 percent means that the blocking scheme has reduced the complexity of the full pairwise task by that amount. Note that the *optimal* RR for 100 percent PC for the datasets in Table 1 can be calculated by using the formula $1 - C_5/C_4$ where C_p stands for Column p. Following previous research, PC-RR graphs are used to quantify the effectiveness-efficiency tradeoff [2].

4.3 Baseline

As earlier stated, many popular link discovery systems like Silk [7] use token-based pre-matching to reduce complexity. *Canopy Clustering*, originally proposed by McCallum et al. [13], represents such methods since it is token-based, makes few assumptions and has been shown to be experimentally robust [1]. It was also used as the baseline in another competitive system [2]; hence, we use it as our baseline. In the best performing implementation[11], the algorithm

[7] e.g. `http://www.pewstates.org/research/reports/` for the third test case.
[8] `dbpedia.org`
[9] `vgchartz.com`
[10] `https://sites.google.com/a/utexas.edu/mayank-kejriwal/datasets`
[11] Documented by Baxter et al. [1].

randomly chooses a seed entity from one dataset to represent a cluster, and all entities in the second dataset with TFIDF scores above a threshold are placed in that cluster. Clusters may overlap.

4.4 Methodology

The dataset mapping step was applied on the first three test sets in Table 1. It is not applicable to the fourth test set. The dominating strategy introduced earlier was used as the confidence function in Algorithm 1. For the second step, note that the baseline chooses seeds randomly. We compensated by conducting ten trials for each run of the algorithm, and averaging PC and RR. The threshold was tuned and set to a low value of 0.0005 to maximize baseline recall.

The parameters in Algorithm 2 were set to values that were found after some initial tuning (on a subset of the first test collection). The size of initial training set was set to 300 each for both duplicates ($|F_D|$) and non-duplicates ($|F_N|$). β was set to 10, τ to 30, ϵ to 0.8 and η to 0.2. In additional experiments, we varied each of these parameters by 50%. There was no significant difference from the results we subsequently show with these parameter settings. Future work will investigate automatic parameter tuning. In our earlier work, the feature selection was also found robust to varying parameters [10]. Note that the training set is kept constant, regardless of dataset growth.

To evaluate the learned blocking scheme in a practical blocking method, one additional parameter, *maxBucketPairs* is used to enable a technique called *block purging* [17]. The technique discards blocks that have more than *maxBucketPairs* candidate pairs, since the *cost* of processing these blocks is greater than *expected gain*. Block purging was also used in the baseline, for consistency. Varying *maxBucketPairs* from values typically ranging from 1000 to 100,000 effectively varies RR. Data points showing PC at different values of RR are obtained and plotted. All experiments were run on an Intel Core 2 Duo machine with 3 GB of memory and 2.4 GHz clock speed. All code was implemented in Java.

4.5 Results and Discussion

With the dominating strategy, dataset mapping yielded perfect mappings for all three test sets. We ran some additional experiments, including using more government test data (from years 2003-2013 instead of 2009-2013) and Constitute and Case Law data from other countries, and the mappings were still perfect. It would seem, therefore, that the normalized TF measures and the dominating strategy are suited to the problem, at least on the tested domains.

Figure 4 shows the PC-RR tradeoff results of the learned blocking scheme for Canopy Clustering and the proposed method both with and without dataset mapping[12], on the Colombia and Venezuela collections. The gains of dataset

[12] Recall that dataset mapping was designed to be compatible with other algorithms as well, including Canopy Clustering.

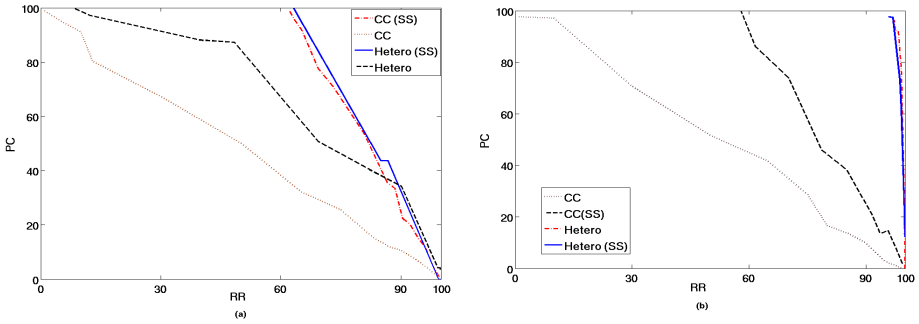


Fig. 4. Results of the proposed method (*Hetero*) against baseline (*CC*) on the (a) Colombia and (b) Venezuela datasets, with *SS* indicating that dataset mapping (or *Source Selection* as it is denoted in the codebase) was utilized. *PC* is Pairs Completeness and *RR* is Reduction Ratio.

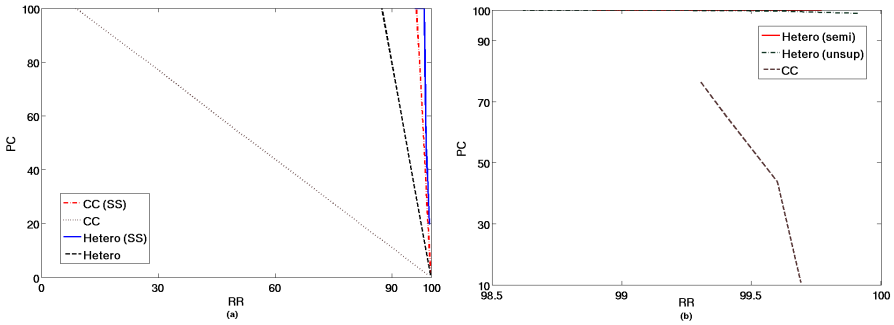


Fig. 5. *Hetero* vs. *CC* on the (a) government 2009-13 budget and (b) video game datasets. In (b), the underlying link was *owl:sameAs*. The noisy training set generator was used for the *unsup* version of *Hetero*, while perfectly labeled examples were provided for *semi*. Note the changed scale (esp. X-axis) in (b).

mapping are readily apparent for *CC* in both cases. The proposed method, *Hetero*, outperforms the non dataset-mapping version of *CC* but the gap narrows considerably when dataset mapping is employed. This shows that, in cases where training sets are not readily available, an off-the-shelf dataset mapping algorithm can boost performance. The dataset mapping gains for *Hetero* aren't significant on Venezuela, mainly because the algorithm performs well on this dataset even without mapping. On *CC*, however, the gains are again apparent. Note that Venezuela (*(b)* in Figure 4) represents some of the challenges of doing link discovery versus just ER. The proposed BSL was able to overcome these challenges by employing bagging and feature selection.

Figure 5a shows the results on the five-pair government budget data. For this collection, the gains of dataset mapping are amplified. This is because there are more datasets in the collection, so the dataset mappings are particularly useful.

We ran further experiments on the full government data (from years 2003-2013) and confirmed this. This time, *Hetero* also shows noticeable gains, with the curve shifting to the right when dataset mapping is employed. Figure 5b shows the results for learning blocking schemes for ER. The BSL is able to significantly outperform *CC*, regardless of whether it is completely unsupervised or with a provided perfectly labeled training set.

Finally, we repeated the experiments above but *without* bagging. Highest *f-scores*[13] of PC and RR declined on all cases by at least 5%, with 95% statistical significance using Student's distribution. Otherwise, the graphical trends were similar. We do not repeat the figures here.

5 Future Work and Conclusion

In this paper, a link-discovery blocking scheme learner was proposed. The first step of the method operates in an unsupervised fashion and performs *dataset mapping* by employing document-level similarity measures. It is compatible with existing clustering and blocking algorithms, experimental savings demonstrated on two such methods. The second step is a heterogeneous BSL that uses techniques like bagging to achieve robust performance, even as the training sets remain constant and the datasets grow in size.

Future work will evaluate the dataset mapping step and the accompanying confidence strategies more extensively, and develop parameter tuning techniques for the learner itself. Another important aspect is investigating scalability of the learner; in particular, we are developing techniques for 'pruning' property tables so that the learner can efficiently scale by learning schemes in a reduced feature space. We believe that this provides an excellent opportunity for cross-fertilizing ongoing scalability efforts in the ontology matching community [6].

Acknowledgments. The authors would like to thank Juan Sequeda for providing the Constitute and Case Law datasets.

References

1. R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *ACM SIGKDD*, volume 3, pages 25–27. Citeseer, 2003.
2. M. Bilenko, B. Kamath, and R. J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 87–96. IEEE, 2006.
3. P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *Knowledge and Data Engineering, IEEE Transactions on*, 24(9):1537–1555, 2012.
4. S. Duan, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Instance-based matching of large ontologies using locality-sensitive hashing. In *The Semantic Web–ISWC 2012*, pages 49–64. Springer, 2012.

[13] Given by 2.RR.PC/(PC+RR)

5. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.
6. J. Euzenat, P. Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.
7. R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649, 2012.
8. R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *WebDB*, 2011.
9. R. Jonker and T. Volgenant. Improving the hungarian assignment algorithm. *Operations Research Letters*, 5(4):171–175, 1986.
10. M. Kejriwal and D. P. Miranker. An unsupervised algorithm for learning blocking schemes. In *Data Mining, 2013. ICDM'13. Thirteenth International Conference on*. IEEE, 2013.
11. H.-s. Kim and D. Lee. Harra: fast iterative hashed record linkage for large-scale data collections. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 525–536. ACM, 2010.
12. Y. Ma, T. Tran, and V. Bicer. Typifier: Inferring the type semantics of structured data. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 206–217. IEEE, 2013.
13. A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM, 2000.
14. M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 440. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
15. A.-C. N. Ngomo. A time-efficient hybrid approach to link discovery. *Ontology Matching*, page 1, 2011.
16. A.-C. N. Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *The Semantic Web: Research and Applications*, pages 149–163. Springer, 2012.
17. G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 535–544. ACM, 2011.
18. F. Scharffe, Y. Liu, and C. Zhou. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*, 2009.
19. D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *The Semantic Web–ISWC 2011*, pages 649–664. Springer, 2011.
20. A. Verikas, A. Gelzinis, and M. Bacauskiene. Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 44(2):330–349, 2011.
21. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *The Semantic Web-ISWC 2009*, pages 650–665. Springer, 2009.
22. K. Wilkinson, C. Sayers, H. A. Kuno, D. Reynolds, et al. Efficient rdf storage and retrieval in jena2. In *SWDB*, volume 3, pages 131–150, 2003.