Leveraging Unlabeled Data to Scale Blocking for Record Linkage

Yunbo Cao[†‡], **Zhiyuan Chen**[§], **Jiamin Zhu**[†], **Pei Yue**[*], **Chin-Yew Lin**[‡], **Yong Yu**[†]
[†]Department of Computer Science, Shanghai Jiao Tong University, China
[‡]Microsoft Research Asia, China
[§]School of Software, Dalian University of Technology, China
[*]Microsoft Corporation, USA
{yunbo.cao, peiyue, cyl}@microsoft.com; {zy.chen, jmzhu}@live.com; yyu@apex.sjtu.edu.cn

Abstract

Record linkage is the process of matching records between two (or multiple) data sets that represent the same real-world entity. An exhaustive record linkage process involves computing the similarities between all pairs of records, which can be very expensive for large data sets. Blocking techniques alleviate this problem by dividing the records into blocks and only comparing records within the same block. To be adaptive from domain to domain, one category of blocking technique formalizes 'construction of blocking scheme' as a machine learning problem. In the process of learning the best blocking scheme, previous learning-based techniques utilize only a set of labeled data. However, since the set of labeled data is usually not large enough to well characterize the unseen (unlabeled) data, the resultant blocking scheme may poorly perform on the unseen data by generating too many candidate matches. To address that, in this paper, we propose to utilize unlabeled data (in addition to labeled data) for learning blocking schemes. Our experimental results show that using unlabeled data in learning can remarkably reduce the number of candidate matches while keeping the same level of coverage for true matches.

1 Introduction

Record linkage is the process of matching records between two (or multiple) data sets that represent the same real-world entity. Record linkage plays a central role in many applications. For example, a price comparison system collects offers from different online shopping sites (e.g., Amazon[1], eBay[2]) that may refer to the same product. As another example, to provide an integrated search experience for Facebook[3] and LinkedIn[4], one web search engine may want to merge the records from the two sites: for a given person that has accounts at both sites, it creates a composite record that combines the information from both sites.

In principle, record linkage needs to compare every pair of records from different data sets, which makes it problematic to scale the process to large data sets. For example, if two databases, $\mathbf{A}$ and $\mathbf{B}$, are to be linked, the total number of potential record pair comparisons thus equals $|\mathbf{A}| \times |\mathbf{B}|$ ($|\cdot|$ denotes the number of records in a database). The number can be extremely large in real applications (e.g., $|\mathbf{A}| = |\mathbf{B}| = 10^7$). To reduce the large amount of potential record pair comparisons, typical record linkage approaches employ a technique called *blocking*: a single record attribute, or a combination of attributes, is used to partition (or group) records into blocks such that records having the same value in the blocking attribute are grouped into one block, and then only records within the same block are compared by assuming that records in different blocks are unlikely to match.

Various blocking methods have been proposed for record linkage. One category of method [Newcombe and Kennedy, 1962; McCallum *et al.*, 2000; Baxter and Christen, 2003; Gu and Baxter, 2004] is based on manual selection of blocking attributes and manual parameter tuning. However, an appropriate blocking scheme can be highly domain-dependent, which may fail the blocking methods due to the ad-hoc construction and manual tuning. To address that, another category of method [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006; Evangelista *et al.*, 2010] based on machine learning (thus also considered *adaptive*) is introduced. Given a set of labeled data (matching or non-matching record pairs), learning techniques are employed to produce *blocking attributes* and *comparison methods* in the attributes by using the *objective*: maximize the number of matching record pairs found and minimize the number of non-matching record pairs in the same block. In this paper, we are interested in *extending the machine learning based methods by incorporating unlabeled data into the learning process.*

When dealing with large-scale (real world) data collections (e.g., the price comparison scenario), the machine-learning-based blocking methods tend to generate either a large number of blocks or a large number of candidate matches in each block (or both). Thus, the number of matches to be compared is still intractable although it has been reduced much by *blocking* (e.g., in our evaluation, one baseline method can generate 4×10^{10} matches). The reasons are as follows.

[1]http://www.amazon.com
[2]http://www.ebay.com
[3]http://wwww.facebook.com
[4]http://www.linkedin.com

First, as labeled data is usually expensive to obtain in terms of either human effort or time, the size of labeled data cannot be large enough to well characterize unseen data within concerned domains. Second, blocking schemes learned from labeled data try to accommodate the labeled data as much as possible. Therefore, one blocking scheme working best for labeled data may not work well for unlabeled data in the same domain. For example, *blocking product offers by brands* may work well for a labeled data in which each brand only has one product (and multiple offers for the product), but cannot work well for a large amount of unseen data as one brand can provide hundreds (or even thousands) of products.

To address the above issue, we propose to incorporate unlabeled data into the learning process as well by introducing a new *objective* function. Specifically, the new objective differs from the one used in previous work [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006; Evangelista *et al.*, 2010] in that it considers an additional factor of minimizing the number of candidate matches (of records) in unlabeled data. Due to its large scale, it is impractical (or even infeasible) to take into account all unlabeled data in the learning process. To address that, we then propose an ensemble method which is on the basis of a combinative use of a series of subsets of unlabeled data. Each subset is randomly sampled from the unlabeled data such that its size is much smaller than that of the original set. With the ensemble method, we are then able to make use of unlabeled data efficiently and effectively.

The contributions of this paper can be summarized as follows. (a) We propose to *use unlabeled data to help the learning about blocking schemes*. Although the concept of using unlabeled data has been explored much in the field of machine learning, none of previous work studies its use for blocking. (b) We design an algorithm for effectively and efficiently utilizing unlabeled data by *sampling the data*. (c) We empirically verify the correctness of the proposal in a large scale within a price comparison scenario.

2 Our Proposal

2.1 Blocking Scheme

Our learning-based method for blocking is about learning one best *blocking scheme* from both labeled data and unlabeled data. In the following, we introduce our definition for *blocking scheme*.

Following [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006], we base our definition for *blocking scheme* on a set of predicates, referred to as *blocking predicates*.

Definition A *blocking predicate* is a pair of <*blocking attribute*, *comparison method*>. Thus, if we have t blocking attributes and d comparison methods, we will have $t \times d$ possible blocking predicates.

Using product offers as an example, an attribute can be '*name*', '*part number*', '*category*', etc. And a comparison method can be '*exact match*', '*same 1st three tokens*', etc. For example, one blocking predicate, <*part number*, *exact match*>, means that two records are grouped into the same block if they share *exactly same part number*.

Definition A *blocking scheme* is a disjunction of conjunctions of blocking predicates.

An example *blocking scheme* can be like (<*name, same 1st three tokens*> $\wedge$ <*brand, exact match*>)$\vee$ <*part number, exact match*>, which consists of two conjunctions.

In the following subsection, we introduce our formalization for the problem of learning a blocking scheme on the basis of both labeled data and unlabeled data.

2.2 Problem Formalization

Following the standard machine learning setup, we denote the input and output spaces by $\mathcal{X}$ and $\mathcal{Y}$, then formulate our task as learning a hypothesis function $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict a $\mathbf{y}$ when given $\mathbf{x}$. In this setup, $\mathbf{x}$ represents a record consisting of m attributes. $\mathbf{y}$ represents the true object (or entity) identifier for $\mathbf{x}$. h is uniquely determined by a *blocking scheme* P (a disjunction of conjunctions), and thus is also denoted as h_P. Given two records $\mathbf{x}_1$ and $\mathbf{x}_2$, $h_P(\mathbf{x}_1) = h_P(\mathbf{x}_2)$ if and only if $\mathbf{x}_1$ and $\mathbf{x}_2$ are grouped into the same block (namely, they are linked to the same true object).

In our learning scenario, we assume that the training set consists of two subsets: $\mathbf{D} = \mathbf{D}_L \cup \mathbf{D}_U$, where $\mathbf{D}_L = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l}$ and $\mathbf{D}_U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$. In real applications, usually $u \gg l$. We also denote $\{\mathbf{x}_i\}_{i=1}^{l}$ by $\mathbf{D}_L^x$.

The goal of the learning problem is to find the best hypothesis function h_P such that

G1) minimize the number of the candidate matches in $\mathbf{D}_L$,

G2) minimize the number of the candidate matches in $\mathbf{D}_U$,

G3) maximize the number of the true matches in the candidate set generated from $\mathbf{D}_L$.

Formally, the goal can be expressed as the following objective function:

$$\arg\min_{h_P} \quad \text{cost}(\mathbf{D}_L^x, P) + \alpha \cdot \text{cost}(\mathbf{D}_U, P) \tag{1a}$$

$$\text{subject to} \quad \text{cov}(\mathbf{D}_L, P) > 1 - \epsilon \tag{1b}$$

where cost$(*, *)$ and cov$(*, *)$ are two functions defined as follows:

$$\text{cost}(\mathbf{A}, p) = \sum\nolimits_{\mathbf{x}\in\mathbf{A},\mathbf{x}'\in\mathbf{A},\mathbf{x}\neq\mathbf{x}'} \frac{I[h_p(\mathbf{x})=h_p(\mathbf{x}')]}{|\mathbf{A}|(|\mathbf{A}|-1)} \tag{2}$$

$$\text{cov}(\mathbf{Z}, p) = \sum_{\substack{(\mathbf{x},\mathbf{y})\in\mathbf{Z},(\mathbf{x}',\mathbf{y}')\in\mathbf{Z} \\ \mathbf{x}\neq\mathbf{x}'}} \frac{I[h_p(\mathbf{x})=h_p(\mathbf{x}'),\mathbf{y}=\mathbf{y}']}{2M(\mathbf{Z})} \tag{3}$$

where $\mathbf{A}$ is a set of records without labels, $\mathbf{Z}$ is a set of records with labels, and p is a blocking scheme. $M(\mathbf{Z})$ is the number of true matches in $\mathbf{Z}$, and $I[.]$ is an indicator function that equals to one if the condition holds and zero otherwise. The first term and the second term in equation (1a) correspond to G1 and G2, and the constraint (1b) corresponds to G3. ϵ is a small value indicating that up to ϵ true matches may remain uncovered, thus accommodating noise and particularly difficult true matches. And the parameter α is used to control the effect of the unlabeled data $\mathbf{D}_U$. If α is set to 0.0, the objective function is reduced to the objective used in [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006] where only labeled data is used in the process of learning *blocking scheme* P. And $\alpha = 1.0$ means that we treat labeled data and unlabeled data equally.

Linking to the metrics *reduction ratio* and *pairs completeness* [Elfeky *et al.*, 2002], minimizing the two terms in equation (1a) is equivalent to maximizing the *reduction ratios* calculated with $\mathbf{D}_L$ and $\mathbf{D}_u$ respectively. And the constraint assures that the *pairs completeness* is above a threshold $(1-\epsilon)$.

The size of $\mathbf{D}_U$ can be rather large in real applications. For example, our evaluation set regarding a price comparison scenario includes about 29 million product offers. On one hand, the large number demonstrates the necessity of leveraging unlabeled data in learning since a small amount of labeled data cannot characterize the data well. On the other hard, it presents a challenge for fully utilizing it. As will be elaborated later, the process of learning the best blocking scheme with the above objective function involves multiple passes of the data. This is also true for the algorithms introduced in [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006]. Thus, it cannot be practical to include all unlabeled data into the learning process. Instead, we propose to make use of sampling techniques to deal with the dilemma. Specifically, we first construct s subsets $\{\mathbf{D}_U^i\}_{i=1}^s$ by randomly sampling records from $\mathbf{D}_U$. By controlling the sampling rate, we can assure that $|\mathbf{D}_U^i| \ll |\mathbf{D}_U|$ $(1 \le i \le s)$. Then, we approximate the second term of equation (1a) with the following value:

$$\alpha \cdot f(\{\text{cost}_i\}_{i=1}^s) \tag{4}$$

where $f(\cdot)$ is an aggregation function that returns a single value from a collection of input values. And cost_i is calculated as,

$$\text{cost}_i = \text{cost}(\mathbf{D}_U^i, P) \tag{5}$$

In our context, the aggregation function can be $average$ and max. Specifically,

$$f_{avg}(\{\text{cost}_i\}_{i=1}^s) = \frac{1}{s} \cdot \sum_{i=1}^{s} \text{cost}_i \tag{6}$$

$$f_{max}(\{\text{cost}_i\}_{i=1}^s) = \max_{i=1}^{s} \text{cost}_i \tag{7}$$

By $average$, we examine every sample subset and treat them equally. In contrast, by max, we care about only the worse case, the subset generating most candidate matches.

2.3 Algorithm

The objective defined in Section 2.2 can be mapped to the classical *set cover problem* [Karp, 1972]. Specifically, applying each conjunction of *blocking predicates* to the data $\mathbf{D}_L^x$ can form a subset which includes all the matches in the block satisfying the conjunction. Then the constraint (1b) in the objective function can be translated as: Given multiple subsets of $\mathbf{D}_L^x$ (with each being determined by one conjunction in the blocking scheme) as inputs, select a number of these subsets so that the selected sets contain almost all the elements of a set formed by all the matches in $\mathbf{D}_L$. The process of selecting subsets needs to assure that equation (1a) is minimized.

As the set cover problem is NP-hard, we have to use greedy algorithms to find an approximate solution. Particulary, we utilize a modified version of sequential covering algorithm (MSCA, algorithm 1) [Michelson and Knoblock, 2006; Mitchell, 1997].

Algorithm 1 Modified Sequential Covering Algorithm

```
1:  Input: Training set $\mathbf{D} = \mathbf{D}_L \cup \mathbf{D}_U$,
           Set of blocking predicates $\{p_i\}$
           Number of sample subsets $s$
2:  $P^* \leftarrow \emptyset$;
3:  $\mathbf{D}' = \mathbf{D}_L$;
4:  for $i = 1$ to $s$ do
5:      $\mathbf{D}_U^i$ = RANDOM-SAMPLING($\mathbf{D}_U$);
6:      $\mathbf{D}' = \mathbf{D}' \cup \mathbf{D}_U^i$;
7:  end for
8:  repeat
9:      $conj$ = LEARN-ONE-CONJUNCTION($\mathbf{D}'$,$\{p_i\}$);
10:     If $conj$ covers any conjunction in $P^*$, remove these
        covered conjunctions from $P^*$;
11:     $P^* = P^* \cup conj$;
12:     $\mathbf{D}' \leftarrow \mathbf{D}' - \{$examples covered by $conj\}$;
13: until the constraint (1b) is satisfied
14: return the set of conjunctions $P^*$
```

Within the main loop (line 8-13), MSCA forms a disjunction by learning a series of conjunctions of blocking predicates with the function LEARN-ONE-CONJUNCTION (detailed in algorithm 2). At each iteration, after a conjunction is learned, the examples (or records) covered by the conjunction are removed from the training data $\mathbf{D}'$. The loop stops with the constraint (1b) satisfied to accommodate noise and particularly difficult true matches. MCSA learns conjunctions independently of each other since LEARN-ONE-CONJUNCTION does not take into consideration any previously learned conjunctions. Therefore, there is the possibility that the set of records covered by one rule is the subset of the records covered by the other rule. For that case, MCSA removes the first rule (line 10).

Algorithm 1 differs from [Michelson and Knoblock, 2006] in that the unlabeled data $\mathbf{D}_U$ is incorporated into the learning process as well. The incorporation is enabled by approximating $\mathbf{D}_U$ with its randomly sampled subsets (line 3-7).

Algorithm 2 presents the implementation of LEARN-ONE-CONJUNCTION. LEARN-ONE-CONJUNCTION uses a greedy and general-to-specific beam search. General-to-specific beam search makes each conjunction as restrictive as possible because at each iteration it adds only one new *blocking predicate* p to the best conjunction c^*. Although any individual conjunction learned by general-to-specific beam search might only have a minimum coverage σ (line 7), the final disjunction P^* will combine all the conjunctions to increase the coverage. Thus, the goal of each LEARN-ONE-CONJUNCTION is to learn a conjunction that minimizes the objective (1a) as much as it can, so when the conjunction is disjoined with others, it contributes as few false-positive candidate matches to the final candidate set as possible. In the objective (1a), we take into account the effect of the conjunction on the unlabeled data $\mathbf{D}_U'$ as well (line 13). That is, both the number of candidate matches for $\mathbf{D}_L'$ and

Algorithm 2 LEARN-ONE-CONJUNCTION

```
1:  Input: Training set $\mathbf{D}'$,
           Set of blocking predicates $\{p_i\}$
           A coverage threshold parameter $\sigma$
           A precision threshold parameter $\tau$
           A parameter for beam search $k$
2:  $c^* \leftarrow$ null; $C \leftarrow \{p_i\}$;
3:  repeat
4:    $C' = \emptyset$;
5:    for all $c \in C$ do
6:      for all $p \in \{p_i\}$ do
7:        if cov$(\mathbf{D}', c \wedge p) < \sigma$ then
8:          continue;
9:        end if
10:       $c' \leftarrow c \wedge p$;
11:       $C' = C' \cup \{c'\}$;
12:       Remove any $c'$ that are duplicates from $C'$;
13:       if cost$(\mathbf{D}'_L, c') + \alpha \cdot$ cost$(\mathbf{D}'_U, c') <$ cost$(\mathbf{D}'_L, c^*) +$
          $\alpha \cdot$ cost$(\mathbf{D}'_U, c^*)$ $precision(c') > \tau$ then
14:         $c^* \leftarrow c'$;
15:       end if
16:     end for
17:   end for
18:   $C \leftarrow$ best $k$ members of $C'$;
19: until $C$ is empty
20: return $c^*$
```

that for $\mathbf{D}'_U$ are minimized. In contrast, previous work [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006; Evangelista *et al.*, 2010] only considers the first term.

The coverage threshold, σ, is used to control how aggressive the algorithm is to claim uncovered labeled data. Michelson and Knoblock [2006] set it to 0.5, which means that any learned conjunction has to cover at least 50% of uncovered labeled data. However, our experience shows that, for price comparison application, none of single conjunctions can cover more than 30% labeled data. Thus, we set σ to 0.2, instead. When σ is smaller, minimizing the objective (1a) may have a higher probability of returning a highly inaccurate blocking predicate (not addressed in [Michelson and Knoblock, 2006]). To deal with that, we thus introduce another condition (the second one in line 13) where $precision$ is calculated on the basis of a development set. By varying the threshold τ, we can also manage to balance between precision and recall of the resultant blocking scheme. Larger τ enforces every conjunction to be highly accurate and thus gives a high precision of the final blocking scheme, but it lowers recall as it restricts the space of candidate conjunctions; vise versa. For the formal definitions on precision and recall, please refer to Section 3.1.

3 Empirical Evaluation

3.1 Experiment Setup

Dataset. We made use of a data collection from a commercial search engine (SE) which provides a function for price comparison in addition to its basic functions for web pages/images/videos search. To help users compare product offers (the things including price information) from different sites, the SE has to link them to right products. We compared our proposal with previous work within this scenario. The data collection (a subset of the data in the SE) includes about 29 million product offers, which covers 14 categories such as 'cameras & optics', 'movies', 'beauty & fragrance', 'toys', and etc. We denote it by **RAW**.

From **RAW**, 30,073 offers were randomly selected and then linked to their corresponding products by humans. In our experiments, we randomly separated the 30,073 offers into four folds. And then, we used two folds as training data (**TR**) for predicate scheme learning with given parameters (e.g, s, σ, and α), one fold as development data (**DEV**) for tuning the parameter τ evaluating the precision of each individual conjunction (line 13 of algorithm 2), and one fold as test data (**TST**). Thus, compared to the entire data set **RAW**, the volume of the labeled data used for learning, **TR** and **DEV**, is rather small. As we will demonstrate later, only using **TR** and **DEV** is not enough. For algorithm 1, we sampled the s unlabeled data set from **RAW**.

Evaluation Metrics. We utilized both **TST** and **RAW** to to evaluate the performance of blocking methods. On **TST**, we adopted precision and recall, the metrics extensively used in record linkage and information retrieval. As shown in the following, they can be linked to the functions cov$(*, *)$ and cost$(*.*)$.

$$\begin{aligned} Precision(P) &= \frac{2 \cdot M(\mathbf{D}_L) \cdot \text{cov}(\mathbf{D}_L, P)}{|\mathbf{D}^x_L| \cdot (|\mathbf{D}^x_L| - 1) \cdot \text{cost}(\mathbf{D}^x_L, P)} \\ Recall(P) &= \text{cov}(\mathbf{D}_L, P) \end{aligned}$$

On **RAW**, we made use of the metric Reduction Ratio ($RR = 1 -$ cost$(\mathbf{RAW}, P)$) to evaluate how effective our proposal is in the sense of reducing the number of candidate matches. As be shown later, both the baseline method and our proposal can achieve a higher RR. However, we argue that, for real applications, even a small difference (99.8% vs. 99.9%) in RR is of great values. To illustrate the significance of the difference, we also reported the number of candidate matches (denoted by #Cands).

Other Configurations. The set of attributes for offers or products varies from category to category. In our experiments, we just picked up a subset of them which can be applied to more than one categories, and then used them as *blocking attributes*. The *blocking attributes* include 'name', 'brand', 'category', 'part number', 'GTIN', 'color', and 'gender'. As for 'comparison methods', we used 'exact match', 'case-insensitive match', 'same 1st two tokens', 'same 1st three tokens', 'same last token', 'same last two tokens', and 'same last three tokens'.

For the data sampling, to demonstrate the necessity of using multiple sample subsets, s was set to 1 or 10. And the sample size is 20,000, which is comparable to the size of the labeled data **TR** plus **DEV**. For the beam search, in order to consider as many good conjunctions as possible, k was set to 10. For the precision threshold τ, we tried the values 0.0, 0.1, $\cdots$, 1.0.

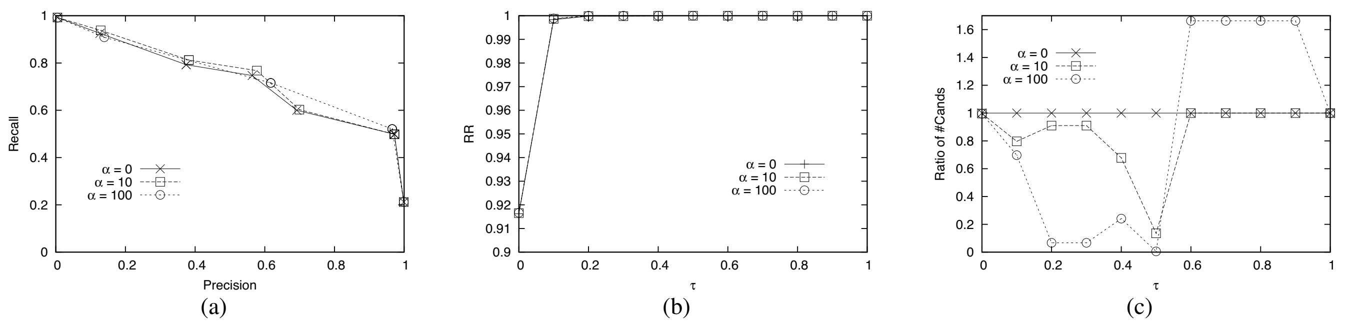

Figure 1: The blocking results when $s = 10$.

3.2 Results

Figure 1 presents the experimental results when $s = 10$. Specifically, Figure 1(a) show the precision-recall curves obtained with different values of α. Each precision-recall curve is generated by varying the precision threshold τ, where the blocking scheme with a bigger τ has a higher precision and a lower recall (and vise versa). Thus, this result shows that *we can manage to control precision and recall by setting different values to* τ. We also see that our proposal of using unlabeled data ($\alpha > 0$) in learning performs comparably with (even better than) the blocking method not using unlabeled data ($\alpha = 0$). This result confirms that our proposal can make use of labeled data well while taking into consideration of unlabeled data. At the same time, *our proposal can remarkably reduce the number of candidate matches on the unseen data* RAW. If we check this only through Figure 1(b) (in terms of RR), the reductions are not so obvious. However, if we examine how many candidate matches each blocking scheme generated directly (Figure 1(c)), the reductions are significant. The y axis in Figure 1(c) is the ratio of #Cands of one blocking scheme to #Cands of the baseline method (not using unlabeled data). As seen in the figure, for all the values of τ, the ratio for our proposal with $\alpha = 10$ is always below 1.0, which indicates that our proposal generates less candidate matches than the baseline method. For example, when $\tau = 0.5$, we even reduced more than 86.5% ($= 1 - 13.5\%$) of candidate matches. At the same point, according to Figure 1(a), we can also achieve a good balance between precision and recall. In a summary, *our proposal is able to save much (downstream) running time for record pair comparison by reducing the potential number of matches in the data to be linked while well balancing precision and recall.*

In all the above experiments, only *max* aggregation was employed. To see the difference between the aggregation methods (equations (6) and (7)), we present a comparison on them in Figure 2 where $s = 10$. From the figure, we observe that the two types of aggregation perform exactly same when $\alpha = 10$. However, when $\alpha = 100$, they perform quite differently from each other. In addition to the observations, we can also have the suggestions as follows: (a) We should employ max as aggregation method when $\tau \leq 0.5$, and employ $average$, otherwise; and (b) to simplify that, we can always use $average$ due to its robustness relative to max. If we seek for a higher recall (the main purpose of blocking) and at the same time want to avoid comparing/verifying too many candidate matches, we should set τ to a value smaller than 0.5, emphasize the contribution of unlabeled data ($\alpha = 100$), and use max aggregation to reduce the number of candidate matches in unlabeled data.

We also conducted the experiments to verify the necessity of employing aggregation methods. Within the experiments, we carried on three trials by setting $s = 1$ and $\alpha = 100$. The individual trials were fed with different sample subsets. Figure 3 provides the results in terms of #Cands. From the figure, we can see that the individual blocking schemes obtained with the different unlabeled data (subsets A, B, and C) can perform quite differently from each other. That is also to say, the behavior of the blocking scheme learned by one single unlabeled subset cannot be predicted. Thus, it is not safe to use unlabeled data without aggregation in the process of learning blocking schemes.

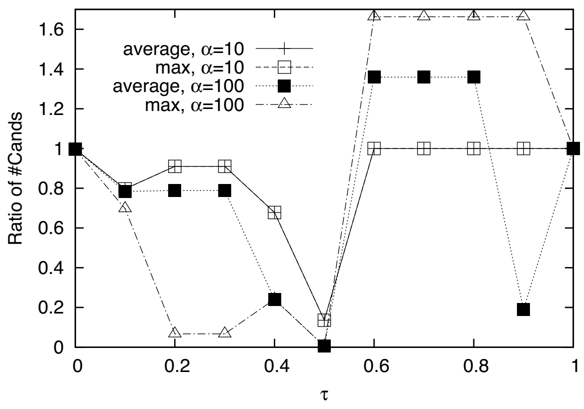

Figure 2: Comparison on different types of aggregation.

4 Related Work

Various blocking methods have been proposed for record linkage. Among them, learning-based methods are most related to our research. Given a set of labeled data (matching or non-matching record pairs), learning-based blocking methods [Bilenko *et al.*, 2006; Michelson and Knoblock, 2006; Evangelista *et al.*, 2010] are to produce *blocking attributes*

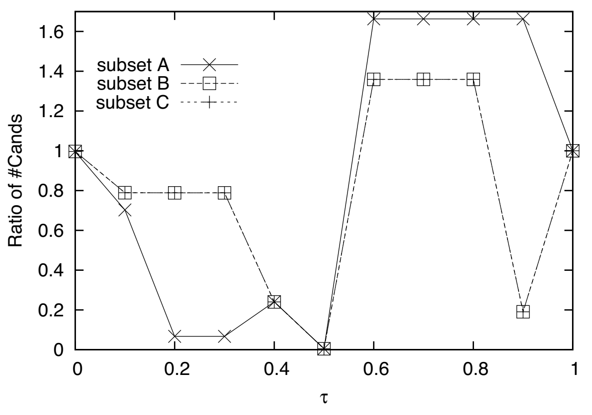

Figure 3: The ratio of #Cands when $s = 1$.

and *comparison methods* in the attributes by using the *objective*: maximize the number of matching record pairs found and minimize the number of non-matching record pairs in the same block. All the methods base their blocking schemes on disjunctive normal form (DNF) of blocking predicates. They differ from each other mostly in the way that the blocking predicates are combined in the process of searching for the best blocking scheme. Specifically, Bilenko et al. [2006] utilized a modified version of Peleg's greedy algorithm [Peleg, 2000], Michelson and Knoblock [2006] used a modified sequential covering algorithm [Mitchell, 1997], and Evangelista et al. [2010] used genetic programming. All the methods make use of only a set of labeled data to learn the blocking scheme. In contrast, in this paper, we propose to take into consideration unlabeled data into the learning as well for the aim of minimizing the number of candidate matches on the data to be integrated (or linked).

A number of methods on the basis of fast nearest-neighbor searching aims at adaptively solving the performance issue on the large scale of data, too. For example, Andoni and Indykthe [2006] proposed to use the Locality-Sensitive Hashing (LSH) scheme as an indexing method in approximate nearest neighbor search problem. Yan et al., [2007] proposed to achieve the adaptivity by dynamically adjusting the sizes of sliding windows for the well-known algorithm, the sorted neighborhood method. Later, Athitsos et al. [2008] proposed a distance-based hashing (DBH) to address the issue of finding a family of hash functions in LSH. Recently, Kim and Lee [2010] proposed an iterative LSH for a set of blocking methods called iterative blocking [Whang *et al.*, 2009]. However, the methods typically rely on strong metric assumption in the data space, while learning-based blocking methods (including ours) work with arbitrary blocking predicates.

There exists also a large amount of work adopting ad-hoc construction and manual tuning [Newcombe and Kennedy, 1962; McCallum *et al.*, 2000; Baxter and Christen, 2003; Gu and Baxter, 2004]. These methods focus on improving efficiency under the assumption that an accurate blocking function is known.

5 Conclusions

In this paper, we have proposed a novel algorithm for the problem of *blocking for record linkage*. The algorithm learns (not manually constructs) a good blocking scheme on the basis of both labeled data and unlabeled data. In contrast, previous learning-based methods for blocking can only utilize labeled data in learning. Our experimental results showed that the use of unlabeled data in learning can remarkably reduce the number of candidate matches while keeping the same level of coverage for true matches.

There exist two interesting directions for further extending our work: First, in the current implementation, to incorporate the large scale of unlabeled data in learning, we made use of only a simple sampling method, random sampling. It would be interesting to see how other sampling methods (e.g., guide the sampling by the distribution of the blocking attributes) can help. Second, in our experiments, we only studied the use of our proposal within a price comparison scenario. We would like to extend the study to other scenarios as well (e.g., linking Facebook records to LinkedIn records).

References

[Andoni and Indyk, 2006] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, 2006.

[Athitsos *et al.*, 2008] Vassilis Athitsos, Michalis Potamias, Panagiotis Papapetrou, and George Kollios. Nearest neighbor retrieval using distance-based hashing. In *ICDE*, 2008.

[Baxter and Christen, 2003] Rohan Baxter and Peter Christen. A comparison of fast blocking methods for record linkage. In *SIGKDD*, pages 25–27, 2003.

[Bilenko *et al.*, 2006] Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. Adaptive blocking: Learning to scale up record linkage. In *ICDM*, pages 87–96, 2006.

[Elfeky *et al.*, 2002] Mohamed G. Elfeky, Ahmed K. Elmagarmid, and Vassilios S. Verykios. Tailor: A record linkage tool box. In *ICDE*, pages 17–28, 2002.

[Evangelista *et al.*, 2010] Luiz Osvaldo Evangelista, Eli Cortez, Altigran Soares da Silva, and Wagner Meira Jr. Adaptive and flexible blocking for record linkage tasks. *JIDM*, 1(2):167–182, 2010.

[Gu and Baxter, 2004] Lifang Gu and Rohan A. Baxter. Adaptive filtering for efficient record linkage. In *ICDM*, 2004.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Kim and Lee, 2010] Hung-Sik Kim and Dongwon Lee. Harra: Fast iterative hashed record linkage for large-scale data collections. In *EDBT*, pages 525–536, 2010.

[McCallum *et al.*, 2000] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *SIGKDD*, pages 169–178, 2000.

[Michelson and Knoblock, 2006] Matthew Michelson and Craig A. Knoblock. Learning blocking schemes for record linkage. In *AAAI*, volume 1, pages 440–445, 2006.

[Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997.

[Newcombe and Kennedy, 1962] Howard B. Newcombe and James M. Kennedy. Record linkage: making maximum use of the discriminating power of identifying information. *Commun. ACM*, 5(11):563–566, 1962.

[Peleg, 2000] David Peleg. Approximation algorithms for the label-cover$_{\max}$ and red-blue set cover problems. In *SWAT*, pages 220–230, 2000.

[Whang *et al.*, 2009] Steven Euijong Whang, David Menestrina, Georgia Koutrika, Martin Theobald, and Hector Garcia-Molina. Entity resolution with iterative blocking. In *SIGMOD*, pages 219–232, 2009.

[Yan *et al.*, 2007] Su Yan, Dongwon Lee, Min-Yen Kan, and C. Lee Giles. Adaptive sorted neighborhood methods for efficient record linkage. In *JCDL*, pages 185–194, 2007.