A Versatile Record Linkage Method by Term Matching Model Using CRF

Quang Minh Vu, Atsuhiro Takasu, and Jun Adachi

National Insitute of Informatics, Tokyo 101-8430, Japan
{vuminh,takasu,adachi}@nii.ac.jp

Abstract. We solve the problem of record linkage between databases where record fields are mixed and permuted in different ways. The solution method uses a conditional random fields model to find matching terms in record pairs and uses matching terms in the duplicate detection process. Although records with permuted fields may have partly reordered terms, our method can still utilize local orders of terms for finding matching terms. We carried out experiments on several well-known data sets in record linkage research, and our method showed its advantages on most of the data sets. We also did experiments on a synthetic data set, in which records combined fields in random order, and verified that it could handle even this data set.

1 Introduction

Information on the web is growing at an explosive rate [10], and information about an object may appear in several places. To retrieve such information effectively, we need to merge all the spread out information on the same object. Some of the previous studies have dealt with collecting information about companies, people, etc. [16,1,3]. In this study, we tried to collect information about journal papers from different data resources. An application is extraction of information about papers from publication lists posted on the web and matching them with records in research paper databases. Since databases and publication lists are different resources, the field orders and field permutations might be different; this makes linkage a more challenging task. We devised a versatile method for linking records that can work well with field-reordered records.

Most of the previous studies on record linkage targeted databases with records that have similar field orders [4,13,8]. For example, some focused on field segmentation records, where two sets of fields in two databases are the same. The methods that were developed in these studies work well with long string combinations from the same set of fields and when the fields are combined in the same order. Hereafter, we call such field segmentation records and field combination string records *segmentation records* and *string records*, for short. For these kinds of records, the previous studies built matching models based on the string edit distance to find common information between two records and to find different information that is inserted/deleted to/from one record. Although methods based on the string edit distance are effective for records with the same field order, they are of limited benefit when the records are from different databases and have different field orders. For example, in these two records: *"Four Seasons Grill Room, 854 Seventh Ave., New York City"* and *"854 Seventh Ave., New York City, Four Seasons Grill Room"*, the common text "Four Seasons Grill Room" appear at the head of one record but at the tail of the other record, and the string edit distance method will regard this text as being deleted from both records. This fault may degrade the duplicate detection performance.

There is another record linkage method that builds bags of words for records and measures the weights of common terms [4]. This method can find record pairs that have different field orders, but it neglects the term orders in a string. Therefore, it is difficult to detect overlapping phrases, and this drawback may degrade the duplicate detection performance.

To detect duplications in records that have different field orders, we tried to find matching information at the term level and combine the matching information for the duplicate detection process. Our approach uses a labeling method to find matching terms. Terms in one record are used as labels for matching terms in another record. We solve this labeling problem by using a conditional random fields (CRF)[11,15] model. CRF is usually used to solve the sequence labeling problem. For example, we applied CRF to bibliographic information extraction from title pages of academic articles where term sequences appearing in title pages are labeled with bibliographic components [14]. Unlike standard CRF applications including our previous study [14], here we use CRF to align terms in a pair of bibliographic records. Our way of CRF application is similar to the study [13], where CRF was used to measure the costs of a learnable string edit distance. However, their model suffers when the term orders of the compared strings are different. Our CRF model is analogous to a CRF model which aligns terms in machine translation [5]. Our model detects reordered terms in different records, while their model [5] detects terms having the same meaning in two languages.

The rest of this paper is organized as follows. Section 2 summarizes the related studies on record linkage. Section 3 presents our term matching model that is based on the CRF model and our duplicate detection classifier that is based on the support vector machine (SVM) method [7]. Section 4 shows experimental results on several well-known data sets and compares them with the results of previous research. Section 5 discusses the advantages of our model and compares it with other CRF models in other applications so that the reader can better understand the characteristics of our approach. Finally, Section 6 concludes this research.

2 Related Work

In [9,4,13], the string edit distance method is used to detect duplicate parts and different parts of two records. The authors carefully considered edit operations, so that they could be used to recognize important text changes that help to differentiate two distinct records. To recognize important text changes, they used an SVM model in [4] and a CRF model in [13]. However, these approaches

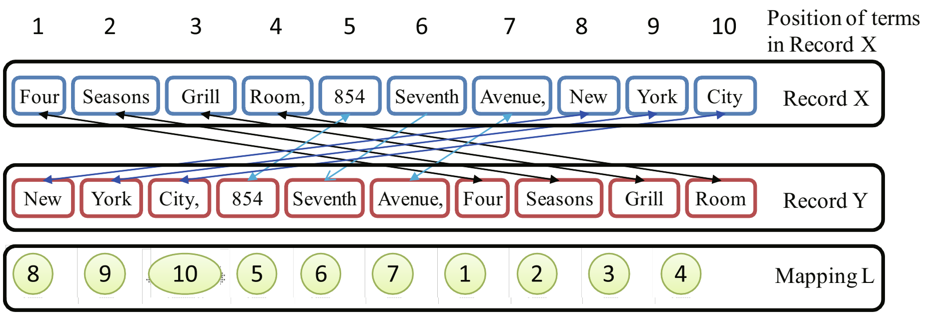

1 2 3 4 5 6 7 8 9 10
Position of terms in Record X
Four Seasons Grill Room, 854 Seventh Avenue, New York City
Record X
New York City, 854 Seventh Avenue, Four Seasons Grill Room
Record Y
8 9 10 5 6 7 1 2 3 4
Mapping L

Fig. 1. Example of term matching using the proposed model

have trouble handling records with different field orders. If the fields of the two records are not aligned or their field combinations are in different orders, there are no reasonable edit operations to transform one record into another record.

In [4], the authors also used bags of words of records to find duplications. They improved the term frequency-inverse document frequency (tf-idf) weighting scheme [2,12] in the traditional vector space model by using a model to learn important words that are strongly related to a specific database. This method can be applied to records that have different data field orders. However, since the bag-of-words approach ignores term orders in a string, it can not recognize terms' consecutiveness, and therefore, it can not utilize matching phrases in records.

3 Term Matching CRF Model

3.1 Application of CRF to Term Matching Problem

Let us define the notations used in this paper. For a sequence $\boldsymbol{X}$, $|\boldsymbol{X}|$ denote the length of the sequence. For a set S and integer l, S^l denotes a set of sequences of elements of S whose length is l. We detect identical record pairs by performing the following two steps:

1. *term mapping*: map terms in a pair of records using CRF, and
2. *resolution*: decide the identity of the pair based on the term mapping using SVMs.

Let us consider the pair of records in Section 2. Figure 1 shows a term mapping between the pair of records $\boldsymbol{X}$ and $\boldsymbol{Y}$. Note that the mapping does not preserve the order of terms.

A record is denoted by a sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$, where each element $\boldsymbol{x}_i$ represents a term in the record. For a segmentation record, we encode the field type and field ID together with the spelling of the term in order to utilize them as features in CRF. Therefore, each term $\boldsymbol{x}_i$ contains three pieces of information: $\boldsymbol{x}_i = (x_i^{spell}, x_i^{field}, x_i^{id})$, where x_i^{spell} is the term spelling, x_i^{field} type of field where $\boldsymbol{x}_i$ exists, and x_i^{id} the field ID. For example, the field type of the first to third terms of the record $\boldsymbol{X}$ in Fig. 1 is restaurant name, whereas the field type of the 8th to the 10th terms is city name. Records often contain multiple values for a field such as the authors of the article. The field ID is used to discriminate fields of the same type. For example, regarding a term $\boldsymbol{x}_i$ for the first author and a term $\boldsymbol{x}_j$ for the second author, their field types are the same, but their field IDs are different: i.e., $\boldsymbol{x}_i^{field} = \boldsymbol{x}_j^{field}$ and $\boldsymbol{x}_i^{id} \neq \boldsymbol{x}_j^{id}$. Since a string record does not have any field information, we introduce an imaginary field type *string* and assign it to all terms, i.e., $\hat{\boldsymbol{x}}_i = (x_i^{spell}, \text{string}, 1)$ for any ith term in a string record, where the field ID of all terms is 1.

We denote a mapping between terms in record $\boldsymbol{X}$ and $\boldsymbol{Y}$ as a list $\boldsymbol{m} \equiv (m_1, m_2, \cdots, m_{|\boldsymbol{Y}|})$ of positions of terms in $\boldsymbol{X}$ where m_i denotes the position of the term in $\boldsymbol{X}$ that is mapped to the ith term in $\boldsymbol{Y}$. For example, the mapping in Fig. 1 is represented with $(8, 9, 10, 5, 6, 7, 1, 2, 3, 4)$ by which, for instance, the first term "New" in record $\boldsymbol{Y}$ is mapped to the 8th term "New" in record $\boldsymbol{X}$.

To handle the term mapping by using a linear chain CRF model, for each pair of records $(\boldsymbol{X}, \boldsymbol{Y})$, we use the terms in one record as labels and assign one of them to each term in the other record. Formally, we initially set a null position 0 that means that there is no corresponding term in $\boldsymbol{X}$. For a pair of records $(\boldsymbol{X}, \boldsymbol{Y})$, we use $L_{\boldsymbol{X}} \equiv \{0, 1, 2, \cdots, |\boldsymbol{X}|\}$ as the set of labels for terms in $\boldsymbol{Y}$. The term mapping between a record $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as a label sequence $\boldsymbol{m} \in L_{\boldsymbol{X}}^{|\boldsymbol{Y}|}$.

As in the linear chain CRF model, we assume that the mapping of term $\boldsymbol{y}_i$ is determined by the mapping of $\boldsymbol{y}_{i-1}$ and $\boldsymbol{y}_i$ itself. Then, the probability that $\boldsymbol{m} = (m_1, m_2, \cdots, m_{|\boldsymbol{Y}|}) \in L_{\boldsymbol{X}}^{|\boldsymbol{Y}|}$ is a mapping between a record $\boldsymbol{X}$ and $\boldsymbol{Y}$ is given by

$$p(\boldsymbol{m} \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) \propto \exp\left(\sum_{i=1}^{|\boldsymbol{Y}|} \sum_k \lambda_k f_k(\boldsymbol{x}_{m_i}, \boldsymbol{y}_i) + \sum_{i=2}^{|\boldsymbol{Y}|} \sum_h \mu_h g_h(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i) \right) \tag{1}$$

where $f_k(\boldsymbol{x}_{m_i}, \boldsymbol{y}_i)$ and $g_h(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i)$ are feature functions discussed in the next section; λ_k and μ_h are parameters of the proposed term matching CRF model; and $\boldsymbol{\theta}$ denotes the parameters $\{\lambda_k\}_k \cup \{\mu_h\}_h$. The optimal term mapping for a record $\boldsymbol{X}$ and $\boldsymbol{Y}$ is obtained by solving the optimization problem

$$\underset{\boldsymbol{m} \in L_{\boldsymbol{X}}^{|\boldsymbol{Y}|}}{\operatorname{argmax}} \; p(\boldsymbol{m} \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) \; . \tag{2}$$

3.2 Parameter Learning Algorithm

Given a set of training data $\{(\boldsymbol{X}_1, \boldsymbol{Y}_1, \boldsymbol{m}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2, \boldsymbol{m}_2), \cdots, (\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{m}_n)\}$, we have to find an optimal set of parameters $\boldsymbol{\theta} = \{\lambda_k\}_k \cup \{\mu_l\}_l$ that best models the learning data. The object function is the summary of likelihoods of all training data, as follows.

$$\varPhi(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(\boldsymbol{m}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i) \tag{3}$$

CRF uses regularization to avoid overfitting: it gives a penalty to weight vectors whose norm is too large. The penalty used is based on the Euclidean norm of $\boldsymbol{\theta}$ and on a regularization parameter $1/2\sigma^2$ that determines the strength of the penalty. Therefore, the object function becomes.

$$\varPhi(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(\boldsymbol{m}_i|\boldsymbol{Y}_i, \boldsymbol{X}_i) - \sum_{k} \frac{\lambda_k^2}{2\sigma^2} - \sum_{l} \frac{\mu_l^2}{2\sigma^2} \tag{4}$$

To maximize $\varPhi(\boldsymbol{\theta})$, CRF uses the LBFGS algorithm[6] to update parameters $\boldsymbol{\theta}$ iteratively so that $\varPhi(\boldsymbol{\theta})$ approaches the global maximum point. At each iteration step, the LBFGS algorithm uses partial differential coefficients $\frac{\partial \varPhi}{\partial \lambda_k}, \frac{\partial \varPhi}{\partial \mu_l}$ to update parameters. See [15] for details about calculating these coefficients.

3.3 Label Assigning Algorithm

To find term associations between two records, we have to maximize $p(\boldsymbol{m} \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta})$ which is equivalent to the following optimization problem.

$$\begin{aligned} \boldsymbol{m} &= \underset{\boldsymbol{m}}{\operatorname{argmax}} \left(\sum_{i=1}^{|\boldsymbol{Y}|} \sum_{k} \lambda_k f_k(\boldsymbol{x}_{m_i}, \boldsymbol{y}_i) + \sum_{i=2}^{|\boldsymbol{Y}|} \sum_{h} \mu_h g_h(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i) \right) \\ &= \underset{\boldsymbol{m}}{\operatorname{argmax}} \left(\sum_{i=1}^{|\boldsymbol{Y}|} \sum_{k} \phi(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i) \right) \end{aligned} \tag{5}$$

where $\phi(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i) = \sum_k \lambda_k f_k(\boldsymbol{x}_{m_i}, \boldsymbol{y}_i) + \mu_h g_h(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i)$ Let $\psi(l, \boldsymbol{m}_l)$ be $\sum_{i=1}^{l} \phi(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i)$, where $\boldsymbol{m}_l = (m_1, m_2, \cdots, m_l)$. Then, we have $\psi(l+1, \boldsymbol{m}) = \psi(l, \boldsymbol{m}) + \phi(\boldsymbol{x}_{m_l}, \boldsymbol{x}_{m_{l+1}}, \boldsymbol{y}_l, \boldsymbol{y}_{l+1})$. We also have

$$\max_{\boldsymbol{m}_{l+1}} \psi(l+1, \boldsymbol{m}_l, m_{l+1}) = \max_{m_{l+1}} \left(\max_{\boldsymbol{m}_{l-1}} \psi(l, \boldsymbol{m}_{l-1}, m_l) + \phi(\boldsymbol{x}_{m_l}, \boldsymbol{x}_{m_{l+1}}, \boldsymbol{y}_l, \boldsymbol{y}_{l+1}) \right) \tag{6}$$

Using Eq. (6), we can solve $\max_{\boldsymbol{m}_{l+1}} \psi(l+1, \boldsymbol{m}_l, m_{l+1})$ consecutively in a dynamic programming manner, which starts at $\max_{\boldsymbol{m}_0} \psi(0, \cdot, m_0) = 0$, where m_0 is a dummy mapping. When we finish at $l = |\boldsymbol{Y}|$, we get the optimal solution of Eq. (5).

3.4 Feature Functions for Term Matching Model

We use a linear chain graph for our term matching model and build feature functions for nodes and edges as follows.

Node Feature Functions. A node feature function $f_k(\boldsymbol{x}, \boldsymbol{y})$ is defined for each field type t and measures the similarity of terms $\boldsymbol{x}$ and $\boldsymbol{y}$. It is defined as

$$f_k(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \sigma(x^{spell}, y^{spell}) & \text{if } x^{field} = y^{field} = t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\sigma(x^{spell}, y^{spell})$ is one of the following string similarities:

– *Full matching function*: $\sigma(x^{spell}, y^{spell}) = 1$ if $x^{spell} = y^{spell}$; otherwise, it is 0.
– *Abbreviation matching function*: $\sigma(x^{spell}, y^{spell}) = 1$ if x^{spell} is the abbreviated form of y^{spell} or y^{spell} is the abbreviated form of x^{spell}. Otherwise it is 0.
– *Close string matching function*: $\sigma(x^{spell}, y^{spell})$ is the edit distance between x^{spell} and y^{spell}.
– *Close number matching function*: $\sigma(x^{spell}, y^{spell}) = 1$ if both x^{spell} and y^{spell} are numeric and $|x^{spell} - y^{spell}|$ is less than a threshold. This function is used to measure the similarity of numeric fields, such as year published in bibliographic records.

Since the string record does not have any field information, we introduce a wild card *General* of the field type that matches any field x^{field} and y^{field} in eq. (7);, i.e., a node feature function for the type *General* returns $\sigma(x^{spell}, y^{spell})$ independent of the field types x^{field} and y^{field}.

Edge Feature Functions. An edge feature function $g_h(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i)$ is defined for each field type *type* and a mapping $\boldsymbol{m}$. It measures the similarity of two consecutive terms. It is defined as

$$g_h(\boldsymbol{x}_{m_{i-1}}, \boldsymbol{x}_{m_i}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i) = \begin{cases} \sigma(x^{spell}_{m_{i-1}}, y^{spell}_{i-1}) \cdot \sigma(x^{spell}_{m_i}, y^{spell}_{i}) & \text{if } x^{field}_{m_{i-1}} = x^{field}_{m_i} = y^{field}_{i-1} = y^{field}_{i} = type, \\ & \quad x^{id}_{m_{i-1}} = x^{id}_{m_i}, y^{id}_{i-1} = y^{id}_{i} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $\sigma(x^{spell}, y^{spell})$ is same as the node feature function.

3.5 Feature Vectors for Resolution by SVM

For resolution by SVM, we have to build feature vectors to represent records' similarity and use these feature vectors to separate duplicate pairs from non-duplicates. We create two feature values as follows.

Feature Values Derived from CRF Feature Functions. For each feature function in the CRF model, we summarize its values across all nodes and normalize it by the record length. Let $\boldsymbol{m}^* = (m^*_1, m^*_2, \cdots, m^*_{|\boldsymbol{Y}|})$ denote the optimal mapping obtained by solving the problem (2). For records $\boldsymbol{X}$ and $\boldsymbol{Y}$, the feature $\hat{f}_k$ derived from the node feature function $f_k(\cdot,\cdot)$ is

$$\hat{f}_k = \frac{1}{|\mathbf{Y}|} \sum_{i=1}^{|\boldsymbol{Y}|} f_k(\boldsymbol{x}_{m_i^*}, \boldsymbol{y}_i) \ . \tag{9}$$

Similarly, for a record $\boldsymbol{X}$ and $\boldsymbol{Y}$, the feature $\hat{g}_h$ derived from the edge feature function $g_h(\cdot,\cdot,\cdot,\cdot)$ is

$$\hat{g}_h = \frac{1}{|\mathbf{Y}|} \sum_{i=1}^{|\boldsymbol{Y}|-1} g_h(\boldsymbol{x}_{m_{i-1}^*}, \boldsymbol{x}_{m_i^*}, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i) \ . \tag{10}$$

Heuristic Feature Values. In addition to feature values described in the previous subsection, we also create the following heuristic features from matching terms that are useful for duplicate detection.

1. Number of terms to be deleted/inserted: Since the CRF model only calculates features from matching terms, we created this feature to take into account different terms for the duplicate detection process.
2. Number of consecutive terms to be deleted/inserted: this feature can put more penalty points on deleted/inserted phrases.
3. Position of terms to be deleted/inserted: For journal citation records, information such as author name often appears at the beginning of strings, whereas venue information often appears at the end. We use these features to differentiate term importance on the basis of position.

4 Experiments

4.1 Experimental Methods

Data sets. We carried out experiments on three well-known data sets that have been used in previous studies. The first data set contains records on restaurant information. We use four fields in this data set: name, address, city and cuisine. The second and the third data sets are the *Cora* and *Citeseer* data sets, and

Table 1. Number of records and duplications in the data sets

Data set	Number of records	Duplications
Restaurant	864	112 duplicate pairs
Cora	1295	122 unique papers
Citeseer Reasoning	514	196 unique papers
Citeseer Face	349	242 unique papers
Citeseer Reinforcement	406	148 unique papers
Citeseer Constraint	295	199 unique papers

they contain citations of papers. In the *Cora* dataset, citations are segmented into fields, and we used five fields in our experiments: author, title, venue, year, and page number. As for the *Citeseer* data set, we used the same subset as in [13] that consists of papers about four topics: *Reasoning*, *Face*, *Reinforcement*, and *Constraint*. Citations in this data set are long string records whose fields are not segmented. Details regarding the number of records and the number of duplications are shown in Table 1.

Table 2. Feature functions for segmentation records of citations in CRF term matching model

Feature type	Field type	Label difference	Matching type 1	Matching type 2
Edge	Author	1	Full	Full
Edge	Author	1	Abbreviation string	Full
Edge	Author	1	Full	Abbreviation string
Edge	Author	1	Abbreviation string	Abbreviation string
Edge	Author	-1	Full	Full
Edge	Author	-1	Abbreviation string	Full
Edge	Author	-1	Full	Abbreviation string
Edge	Author	-1	Abbreviation string	Abbreviation string
Node	Author		Full	
Edge	Title	1	Full	Full
Edge	Title	1	Full	Abbreviation string
Edge	Title	1	Abbreviation string	Full
Edge	Title	1	Full	Close string
Edge	Title	1	Close string	Full
Node	Title		Full	
Edge	Venue	1	Full	Full
Edge	Venue	1	Abbreviation string	Full
Edge	Venue	1	Full	Abbreviation string
Node	Venue		Full	
Node	Page		Full	
Node	Page		Close number	
Node	Year		Full	
Node	Year		Close number	

Feature Functions in the CRF Term Matching Model. In our CRF term matching model, feature functions are used to find the best way to match terms between record pairs. Tables 2 and 3 list examples of feature functions used for citation segmentation records and restaurant string records, respectively. In these tables, the "label difference" column means the difference between the positions of two labels. The "matching type 1" and "matching type 2" columns mean the string matching functions used to match node terms and label terms at the current position and at the previous position, respectively. The notation "Full", "Abbreviation string", "Close string", and "Close number" mean two terms match exactly, one term is abbreviation of another, two terms have small string edit distance, and two term numbers have a small difference in value.

Table 3. Feature functions for string records of restaurant information in CRF term matching model

Feature type	Field type	Label difference	Matching type 1	Matching type 2
Edge	General	1	Full	Full
Edge	General	1	Abbreviation string	Full
Edge	General	1	Full	Abbreviation string
Edge	General	1	Close string	Full
Edge	General	1	Full	Close string
Node	General		Full	
Node	General		Abbreviation string	

Parameter Learning for the CRF Term Matching Model. Our CRF term matching model requires the parameters of the feature functions to be tuned. To find an optimal set of parameters, we prepared a set of duplicate records and annotated matching terms in record pairs. These duplicate records are from a neutral resource other than the restaurant data set, the Citeseer data set, and the Cora data set. We then ran the traditional CRF parameter learning algorithm to find the optimal parameters.

SVM Classifier Learning. We used an SVM classifier to decide the identity of each pair of records. In this experiment, we used the SVM*light* tool.[1] We created training and test data for the SVMs as follows. First, we group records into clusters of duplicates. When making pairs of records even from these clusters, the data is imbalanced; i.e., it contains only a few positive pairs and many negative pairs. Therefore, we do sampling to prepare training data. The sampling method affects the observed. Hence, we used the following sampling methods that are similar to those of the previous studies [4,13].

1. **Selection of negative pairs from the top**
   In [4], the authors roughly grouped records into overlapping clusters and selected all pairs of records in the same cluster for their experiments. This way of sampling resulted in most of the positive pairs and negative pairs with high similarity being selected. Our first sampling method was similar to this method. For positive pairs, we selected all positive pairs in the data set. For negative pairs, we first measured their similarities using the tf-idf vector space model and ranked pairs by their similarities. We then selected negative pairs from the top so as to get k times more negative pairs than positive pairs. We call this sampling method *similarity sampling.*
2. **Selection of negative pairs by random sampling**
   In [13], the authors selected all positive pairs in the data set. Next, they removed negative pairs which were too different and sampled the rest of the negative pairs randomly. They selected ten times more negative pairs than positive pairs. Our second method of pair selection is similar to this one. For positive pairs, we select all positive pairs in the data set. For negative

[1] http://svmlight.joachims.org

pairs, we first measure the similarities of the record pairs by using the tf-idf vector space model. Then, we choose negative pairs with the top similarities to get $2k$ times as many negative pairs as positive pairs. From these negative pairs, we randomly sampled pairs to get half of them. In the end, the number of remaining negative pairs was k times larger than the number of positive pairs. We call this sampling method *random sampling.*

These two ways of sampling create two data sets that have different characteristics. We set $k = 10$ in the experiments with the restaurant data set and the four subsets in the *Citeseer* data set. Regarding the *Cora* data set, the duplicate clusters are large, so the number of positive pairs is also large. Therefore, we set $k = 3$ in the experiments with the *Cora* data set.

Evaluation Metrics. As in the previous studies, we sorted the record pairs by referring to the scores obtained by the SVM classifier and calculated the precision, recall, and f-measure values from the sorted results. We recorded the maximum f-measure value in each test. We used a 50/50 training/test split of data and repeated the random split process 10 times and did cross validations. Then, we took the average of the maximum f-measure values across all tests.

4.2 Experiments on Traditional Data Sets

We carried out experiments on the restaurant data set, *Cora* data set, and *Citeseer* data set. We carried out three experiments on the restaurant data set, using name only, address only, and four fields of name, address, city, and cuisine. We carried out one experiment on the *Cora* data set by using five fields of author, title, venue, year, and page and four experiments on four subsets of the *Citeseer* data: *Reasoning*, *Face*, *Reinforcement*, and *Constraint.*

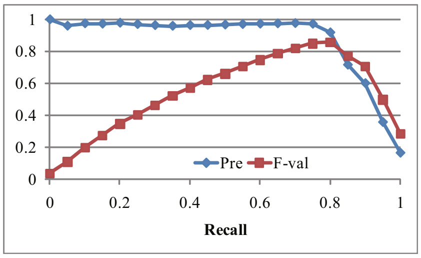
Comparison with Bilenko's Method. We generated the training data and test set by using similarity sampling. Since it is similar to the selection method in [4], it allows us to compare Bilenko's approach directly. The results are shown in Table 4, where Bilenko's results are copied from [4]. As can be seen in Table 4, our approach outperforms Bilenko's approach on six of the eight sets.

Fig. 2 shows the precisions and f-measures for each recall. Graphs (a), (b) and (c) respectively show the performances for the restaurant data set when using the field name only, the field address only, and four fields. Graph (d) shows the performance for Cora data set. As shown in this graph, the proposed method keeps high precision until high recall.

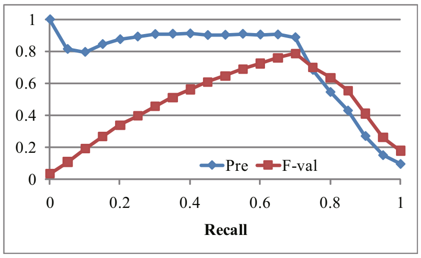
Comparison with McCallum's Method. In this experiment, we generated training and test data by random sampling. Since it is similar to the selection method in [13], and it allows us to compare our method with McCallum's approach directly. The results are shown in Table 5, where McCallum's results are copied from [13]. As can be seen, our approach outperforms McCallum's approach on all six sets.

Table 4. Comparison with Bilenko's approach

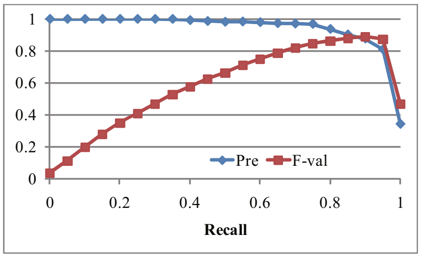
Data set	*Restaurant*			*Cora*	*Citeseer*			
Fields / topic	Name	Address	Name, address, city, cuisine	Author, title, venue, page, year	*Reason-ing*	*Face*	*Reinforce-ment*	*Constr-aint*
Bilenko's approach	43.3%	71.2%	**92.2%**	86.7%	93.8%	**96.6%**	90.7%	94.1%
Our approach	**86.2%**	**74.7%**	90.16%	**87.4%**	**95.6%**	94.4%	**94.9%**	**96.9%**



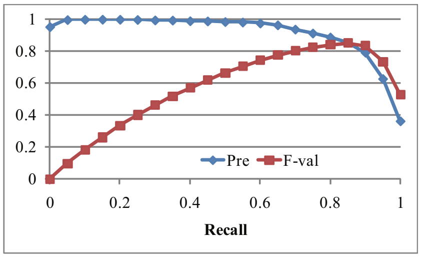
(a) *Restaurant* name



(b) *Restaurant* address



(c) *Restaurant* data set using four fields



(d) *Cora* data set

Fig. 2. Relationship between recall and precision

4.3 Experiments on Synthetic Data Sets

We carried out two experiments on synthetic data sets whose records had their fields permuted. Records in the *Cora* data set are segmented into fields. We combined fields in random order to create record pairs with different field orders. The two experiments are as follows. In the first experiment, we combined fields in only one record to create a pair between one segmentation record and one string record. In the second experiment, we combined fields in both records to create a pair of string records. These two combinations created record pairs with permuted orders, and they have not been used in previous research. The results are listed in Table 6. As can be seen, the first experiment produced results that are equivalent to those for the records with same field orders. This outcome can be explained by arguing that the term matching results are the equivalent to those in the previous experiment and information about field types can be

Table 5. Comparison with McCallum's approach

Data set	*Restaurant*		*Citeseer*			
Field / topic	Name	Address	*Reasoning*	*Face*	*Reinforcement*	*Constraint*
McCallum's approach	44.8%	78.3%	96.4%	91.8%	91.7%	97.6%
Our approach	**88.4%**	**79.6%**	**96.5%**	**95.4%**	**96.6%**	**97.8%**

Table 6. Experimental results on the synthetic data set created from the *Cora* data set

Experiment method	Performance
Pairs of one field record and one string record	**87.5%**
Pairs of two string records	83.9%

exploited from the field segmentation records in this experiment. In the second experiment, the performance slightly deteriorates, but the result is reasonable. In this experiment, the information on field types was removed from both records in each pair, and this was the main cause of degradation. Fig. 3 is an example alignment output of a pair of re-ordered records. As can be seen, our approach can detect consecutive matched terms and calculate a good mapping result.

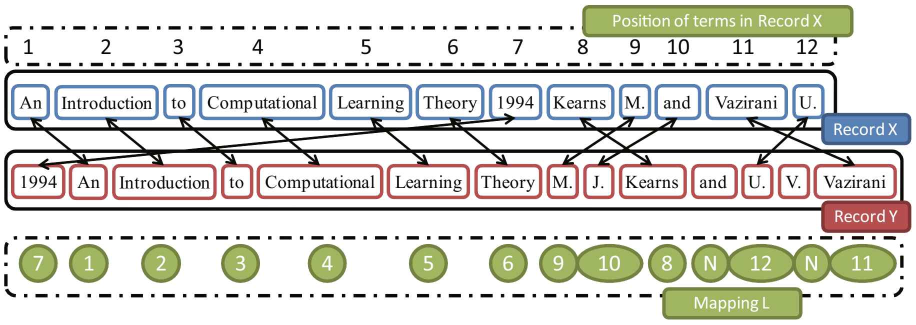

Position of terms in Record X
1 2 3 4 5 6 7 8 9 10 11 12
An Introduction to Computational Learning Theory 1994 Kearns M. and Vazirani U.
Record X
1994 An Introduction to Computational Learning Theory M. J. Kearns and U. V. Vazirani
Record Y
7 1 2 3 4 5 6 9 10 8 N 12 N 11
Mapping L

Fig. 3. An alignment result on a pair of string records

5 Discussion

Duplicate detection of records that have different field orders is more difficult than duplicate detection of records that have the same field order because words are partly reordered in records. To detect duplicate records effectively, the linkage method must be robust to this reordering. Furthermore, although record fields are in different orders, the terms in the same field still keep their order across duplicate records. Therefore, an effective linkage method should be able to recognize this local order for the matching process. Neither the string edit distance approach, nor the bag-of-words approach satisfies both requirements. The string edit distance can recognize local orders but it is weak when faced with field reordering. On the other hand, the bag of words approach is robust to field reordering but it is weak in regard to recognition of terms' local orders. Our approach, on the other hand, satisfies both requirements. It encodes each term by one node in a chain graph, and the algorithm to label pairs of matching terms is robust to field reordering. Our approach can also capture the local order of terms, since it creates feature functions on edges between two consecutive nodes and outputs matching weights for consecutive matching terms. The improvements on most data sets, in particular, on the restaurant name data set and the citation data sets, confirm the advantages of our approach. The results on synthetic records whose fields were randomly ordered are the same as on records that have the same field order. This fact shows that although the fields are randomly reordered, our approach can still utilize terms' local orders to detect consecutive matching terms between duplicate records. This is the main advantage of our approach in comparison with the previous ones. Compared with previous approaches, our method is little more expensive: There is a small cost to prepare term alignments in the training phase. The training dataset is independent from test datasets, so the trained model can be used with different test datasets. In our experiment, we used a training set of 58 aligned pairs. We updated the parameters 1000 times by using the LBFGS algorithm, and it took 32 minutes on a four 3.2Ghz CPU, 8GB memory machine.

In [4,13], the authors consider the details of string edit operations by determining which letters or terms are deleted/inserted. Our approach, on the other hand, only considers the weights of matching terms. However, it can be extended to consider deleted/inserted terms in detail. For example, from the output of the CRF term matching model, we can create features on deleted/inserted terms and use an SVM model to differentiate the importance of deleted/inserted terms.

The proposed term matching model allows many-to-one mappings between a record $\boldsymbol{X}$ and $\boldsymbol{Y}$, as shown in the definition of the mapping. That is, different terms in the record $\boldsymbol{Y}$ may be mapped to the same term in the record $\boldsymbol{X}$. This feature of the term matching model is not preferable. However, each term in the record $\boldsymbol{Y}$ tends to mapped to different term in the record $\boldsymbol{X}$ because of the edge feature functions.

6 Conclusions

We proposed a new method for solving the problem of record linkage between different databases. Our method can find duplicate records from databases that have permuted field orders. We built a term matching model based on the CRF method to find matching terms in records. After extracting the matching terms, we built matching vectors for the record pairs and used an SVM classifier to separate duplicate pairs from non-duplicate pairs.

We experimented on traditional data sets, and our approach showed improvements in comparison with the previous approaches that used either the string edit distance method or the vector space model method. Our method has the good point of the string edit distance method as well as the good points of the vector space model method. That is, it can utilize term orders inside the same field, and it can cope well with field reordering among databases. We also created a synthetic data set by reordering record fields in a random manner, and the results on this synthetic data set were equivalent to those for well-aligned record fields.

References

1. Asano, Y., Nishizeki, T., Toyoda, M., Kitsuregawa, M.: Mining communities on the web using a max-flow and a site-oriented framework. IEICE - Trans. Inf. Syst. E89-D(10), 2606–2615 (2006)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley Longman Publishing (1999)
3. Bhattacharya, I., Getoor, L.: A latent dirichlet model for unsupervised entity resolution. In: SDM (2006)
4. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD 2003: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 39–48. ACM Press, New York (2003)
5. Blunsom, P., Cohn, T.: Discriminative word alignment with conditional random fields. In: ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, Morristown, NJ, USA, pp. 65–72. Association for Computational Linguistics (2006)
6. Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-newton matrices and their use in limited memory methods. Math. Program. 63(2), 129–156 (1994)
7. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
8. Hernandez, M.A., Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery 2(1), 9–37 (1998)
9. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. Journal of the American Statistical Association 84(406), 414–420 (1989)
10. Kitsuregawa, M.: 'Socio Sense' and 'Cyber Infrastructure' for information explosion era': Projects in japan. In: DASFAA, pp. 1–2 (2007)
11. Lafferty, J., Mccallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conf. on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco (2001)
12. Manning, C.D., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (2003)
13. Mccallum, A., Bellare, K., Pereira, F.: A conditional random field for discriminatively-trained finite-state string edit distance. In: Conference on Uncertainty in AI, UAI (2005)
14. Ohta, M., Yakushi, T., Takasu, A.: Bibliographic element extraction from scanned documents using conditional random fields. In: Proc. 3rd International Conf. on Digital Information Management, pp. 99–104 (2008)
15. Sutton, C., Mccallum, A.: An introduction to conditional random fields for relational learning. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
16. Vu, Q.M., Takasu, A., Adachi, J.: Improving the performance of personal name disambiguation using web directories. Inf. Process. Manage. 44(4), 1546–1561 (2008)