A Constrained Clustering Approach to Duplicate Detection among Relational Data

Chao Wang, Jie Lu and Guangquan Zhang

Faculty of Information Technology, University of Technology, Sydney
PO Box 123, Broadway, NSW 2007, Australia
{cwang, jielu, zhangg}@it.uts.edu.au

Abstract. This paper proposes an approach to detect duplicates among relational data. Traditional methods for record linkage or duplicate detection work on a set of records which have no explicit relations with each other. These records can be formatted into a single database table for processing. However, there are situations that records from different sources can not be flattened into one table and records within one source have certain (semantic) relations between them. The duplicate detection issue of these relational data records/instances can be dealt with by formatting them into several tables and applying traditional methods to each table. However, as the relations among the original data records are ignored, this approach generates poor or inconsistent results. This paper analyzes the characteristics of relational data and proposes a particular clustering approach to perform duplicate detection. This approach incorporates constraint rules derived from the characteristics of relational data and therefore yields better and more consistent results, which are revealed by our experiments.

1 Introduction

Data mining tasks usually work on large data warehouses where data often comes from multiple sources. The quality of mining results largely depends on the quality of data. One problem that degrades the data quality is the duplicated data records among the sources. Duplicate detection/elimination then is an essential preprocessing for data mining tasks and different methods have been proposed to deal with this problem [1,2,3,4]. The main idea of these methods is to use certain metrics to determine if certain pairs of data records are similar enough to be duplicates. In these methods, each data record is mostly of one same type and exists as an independent instance during the duplicate detecting process.

On the other hand, relational data is common in reality. Databases in complicated applications often have multiple tables to store multi-type records with relations. Semi-structured data over the Web also has the relational characteristic in terms of the referencing via hyperlinks. The requirement of duplicate detection among relational data is then obvious. Traditional methods can still work but without acknowledging the characteristics of relational data, they tend to produce inadequate and even inconsistent results. Recently, several models [5]

[6] have been proposed to address this issue. These models are built on probability theories. They capture the relational features between records to collectively de-duplicate them with more accuracy. To make the models work, labeled samples should be supplied for estimating model parameters and this training process often takes a considerable amount of time due to the complexity of the model.

This paper then proposes an efficient approach to detect duplicates among relational data. The characteristics of relational data are analyzed from the perspective of duplicate detection. We define constraint rules that capture these characteristics. Our approach then incorporates these constraint rules into a typical canopy clustering process for duplicate detection. Experiments show that our approach performs well with improved accuracy. Furthermore, as our approach is based on clustering, no labeled samples are essentially required and no extra training process is involved, which sometimes is good for large and raw data sets.

The rest of the paper is organized as follows. Section 2 discusses related work. Situation of relational data and its characteristics are discussed in Section 3. Section 4 defines constraint rules for duplicate detection in relational data. Section 5 presents the constrained clustering approach. Experiments and evaluation results are shown in Section 6. Section 7 concludes the paper and discusses the future works.

2 Related Work

Duplicate detection of data was initially studied in database community as "record linkage" [7]. The problem was formalized with a model in [1] and was further extended in [2]. This model computes over features between pairs of records and generates similarity scores for them. Those pairs with scores above a given threshold are treated as duplicates and transitive closure is performed over them to yield the final result. In [8], clustering-based methods are proposed to identify duplicates in publication references. Their approach performs quick canopy clustering with two thresholds at the first stage and perform more expensive clustering methods within each canopy cluster at the second stage for refined results. The records for de-duplication in these methods are not relational, which means each record is a separate instance with no explicit relation with another.

Supervised learning methods have also been employed to make duplicate detection more adaptive with given data. Cohen et al [3] propose an adaptive clustering method and introduces the notion of pairing function that can be learned to check duplicates. Tejada et al [9] use a mapping-rule learner consisting of a committee of decision tree classifiers and a transformation weight learner to help create mapping between records from different data sources. Bilenko and Mooney [4] use a stochastic model and Support Vector Machine (SVM) [10] to learn string similarity measures from samples so that accuracy can be improved for the given situation. These methods are more adaptive and accurate because of their various learning processes which require an adequate amount of labeled

data. Again, all of these methods work on traditional data records with no relational features.

Recently, the relations between data records have been noticed in duplicate detection research community. Singla and Domingos [5] build a collective model that relies on Conditional Random Fields (CRFs) [11] to capture the relation of records for de-duplication. The relationship is indicated just by common field values of data records. The model proposed by Culotta and McCallum [6], which is based on CRFs as well, deals with multi-type data records with relations other than common field values. These methods improve the accuracy of de-duplication by capturing relational features in their models. They also belong to the learning paradigm, which requires labeled samples for training the model parameters. Due to the complexity of the model, the training and inferencing require considerable time, which poses scalability problem.

3 Situations and Characteristics

3.1 Situations

One situation of duplicated relational data can be found in [6], which gives an example of duplicated records of papers and their venues. In this example, the details of papers (author, title) form a database table and the details of venues (conference/journal name) form another table. Obviously, each paper links to a certain venue, forming a relation between the two records. This example can be further extended so that authors may form a separate table containing details of authors (e.g., name, email address) and papers link to certain records in the author table. This kind of normalization is common in designing databases. But it is not the favorite situation for traditional duplicate detection.

Data on the emerging Semantic Web [12] also has this relational feature. Ontologies are introduced to align data on the Semantic Web. A data record (or instance) then has several property values according to the underlying ontology. Particularly, it may have certain property values that refer to other records. Examples are like that an author record has a "publish" property with values pointing to several publication records. More over, unlike the strict database schema, ontology allows data records to be described in a very flexible way with different angles. For example, a publication record can use a reverse property of "publish", say "writtenBy", to refer to the author records. This flexibility, together with the characteristics of decentralization on the Semantic Web, poses challenges to record deduplication.

3.2 Characteristic of relational data

The main characteristic of relational data is certainly the relational feature, i.e., the links between different data records. This often implies that data records may have different types, like the discussed situation where author records link to publication records. Then, multi-type is another characteristic.

In the discussed Semantic Web situation, data instances are not formatted as well as in databases. They are often presented in XML format or described by certain languages (for example, OWL [13]). Therefore, such data instances are semi-structured. In addition, as users can choose different ways to express, the resulting data instances then have different perspectives, not as unified as those in databases.

4 Duplication in relational data

Duplication in relational data can happen on every type of related data records. However, due to the characteristics of relational data, there are some certain patterns among them, which allow us to define constraints. We first introduce some basic notations and then define constraint rules.

4.1 Notations

First, for a particular domain of interest, we can obtain a set of types T, and a set of properties P. There are two types of properties in P: data type properties that allow instances to be described with numbers and/or string values; and object properties that link instances to other instances with particular meanings (following the notions in OWL [13]). An instance then can be described with a type and a subset of properties and their corresponding values (numbers, strings, or other instances).

We identify two classes of instances. If an instance d_i has certain object property values that let it link to a set D_i of other instances, then d_i is identified as "*primary instance*". For any instance d_j ($d_j \in D_i$), d_j is identified as "*derived instance*". The two classes are not exclusive. That is, an instance can be both "primary" and "derived" as long as it points to other instances and has other instance pointing to itself. Given an object property link between two instances (denoted by $d_i \rightarrow d_j$), it is easy to determine the classes of the instances.

If two instances d_i and d_j actually refer to one same real world entity, then the two instances are regarded as duplicates (denoted as $d_i = d_j$). Duplicated instances may not be same in terms of their types, property values as they often come from different sources with different qualities and perspectives. But usually they have similar values. Traditional methods thus use certain similarity measures to compute degrees of similarity of two instances. Given a similarity function f, a clustering process can be conducted to group instances with high similarity degrees into same clusters. For an instance d_i grouped into cluster c_k, we denote as $d_i \in c_k$ or simply $c_k(d_i)$.

4.2 Constraint rules

We define five constraint rules for duplicate detection using clustering approaches. Please note although we call all of them constraints, some actually act more like general rules with little constraint features.

Derived distinction. Given an instance d_p and $D_p = \{d_r | d_p \rightarrow d_r\}$, if $\forall d_i, d_j \in D_p, i \neq j$, then $d_i \neq d_j$.

This rule indicates that all the derived instances from *one* same primary instance should not be duplicates of each other. The reason is quite obvious. Firstly, the application of relating one instance to two or more same other instances is very rare. A paper is always written by different authors if it has more than one author. A conference, in principle, never allows two same papers to be accepted and published. Secondly, the relation between one instance and other several instances often occurs within one data source. Therefore, it is quite easy to maintain so that the derived instances from one same instance are not duplicates. Consider that a person manages his publications to ensure no duplicates occur on his/her web pages. As a result, the instance of this person links to different instances of publications.

Primary similarity. Given two *primary* instances d_a, d_b and one of the resulting clusters c, if $c(d_a, d_b)$, then d_a and d_b have high confidence to be duplicates. We denote $d_a \approx d_b$.

This rule prefers similar primary instances. This rule is based on the observation of the characteristic that primary instances are often described with more detailed and accurate information while derived instances are usually given less attention and hence have less and vaguer details. Therefore, similarity between primary instances are more reliable for duplicate detection.

Derived similarity. Given two *primary* instances d_a, d_b and $d_a \approx d_b$, if we have instances d_x, d_y and cluster c such that $d_a \rightarrow d_x$, $d_b \rightarrow d_y$, $c(d_x, d_y)$, then $d_x \approx d_y$.

This rule treats derived instances that fall in same cluster as duplicates if their corresponding primary instances are treated as duplicates. Strictly speaking, if two primary instances are duplicates, all of their corresponding derived instances should be duplicates as well. However, as noise often exists, it can not be guaranteed that the seeming primary instance duplicates are actual duplicates. To ensure high precision and to prevent false duplicate spreading, we only identify those derived instances that fall in same clusters to be duplicates.

Reinforced similarity. Given instances d_i, d_j, d_m, d_n and clusters c_k, c_l, if we have $d_i \rightarrow d_m$, $d_n \rightarrow d_j$, $c_k(d_i, d_j)$ and $c_l(d_m, d_n)$, then $d_i \approx d_j$ and $d_m \approx d_n$.

This rule addresses the issue of data expressed with different perspectives. Different sources have their own views and describe data from different angles. An entity may be described as a detailed primary instance in one source; But in another source, it could be a simple derived instance. while we may not be confident in the similarity between a primary instance and a derived instance that fall in one same cluster c_k, this similarity will be reinforced if their derived/primary instances also fall into one same cluster c_l. As a result, we treat both pairs as duplicates.

Boosted similarity. Given *derived* instances d_i, d_j, d_m, d_n and clusters c_k, c_l such that $c_k(d_i, d_j)$ and $c_l(d_m, d_n)$, if there exist instances d_x, d_y such that $d_x \not\approx d_y$, $d_x \rightarrow [d_i, d_m]$ and $d_y \rightarrow [d_j, d_n]$, then $d_i \approx d_j$ and $d_m \approx d_n$.

This rule reflects the notion of co-referencing. It is possible that two different instances mention two seemingly same instances that turn out to be different. But the possibility would be much less if more than one (unique) instances mention two sets of seemingly same but different instances. For example, two different papers may have one author's name in common which actually refers to two different persons; But it rarely happens that two papers have two authors' names in common which refers to four different persons. Ideally, if more frequent primary instances are found pointing to more sets of similar derived instances (which may be implemented by frequent item set mining [14]), the confidence of the results would be much higher.

Fig. 1 serves to illustrate the application patterns of different constraint rules we've defined.

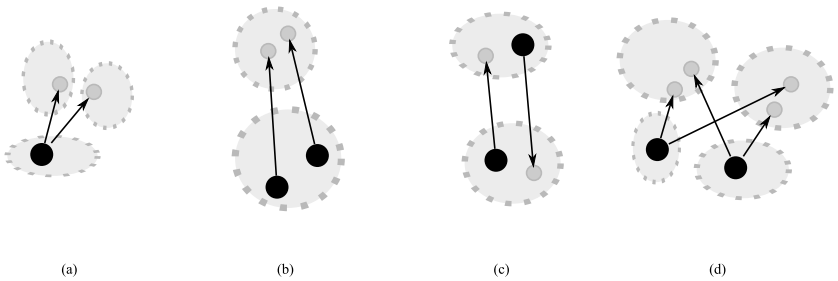

Fig. 1. Illustration of applications of different constraint rules in corresponding situations. (a) derived distinction; (b) primary similarity and derived similarity; (c) reinforced similarity; (d) boosted similarity.

5 Constrained Clustering

This section discusses how the above rules are incorporated in the clustering process. First we present the commonly used canopy clustering method in duplicate detection. Then we focus on our approach.

5.1 Canopy clustering

Canopy clustering [8] is commonly used in duplicate detection [3,4,5]. It uses two similarity thresholds (T_{tight}, T_{loose}) to judge if an instance is closely/loosely similar to a randomly selected instance that acts as canopy center. All loosely similar instances will fall into this canopy cluster. But those closely similar instances will be removed from the list and never compared to another canopy center. Canopy clustering is very effective in duplicate detection as most instances are clearly non-duplicates and thus fall in different canopies. It is also very efficient since it often uses quick similarity measures such as TFIDF [15] computed using inverted index techniques.

Since the resulting canopies may be still large and overlap with each other, a second stage process such as Greedy Agglomerative Clustering (GAC) or

Expectation-Maximization (EM) cluster are usually conducted within each canopy to yield refined results [8].

When canopy clustering is applied to duplicate detection in relational data directly, the performance may not be as good as it is used in normal data. This is because it ignores particular characteristics of relational data. For example, for two derived instances which may represent two different papers of one person, they can be so similar that canopy clustering (even with GAC or EM) treats them as duplicates.

5.2 Canopy clustering with constraints

To improve the performance of duplicate detection in relational data, we modified canopy clustering by incorporating the constraints we've defined. The resulting approach can be divided into four steps.

Input: Set of instances $D = \{d_1, d_2, \cdots, d_N\}$;
Similarity threshold T_{tight}, T_{loose}.
Output: Set of canopy clusters $C_1 = \{c_1, c_2, \cdots, c_K\}$.
Begin
$C_1 = \emptyset$; $D_{tmp} = D$;
while $D_{tmp} \neq \emptyset$ **do**
Create a new canopy cluster $c_{canopy} = \emptyset$;
Pick a random d_r in D_{tmp};
Let $c_{canopy} = \{d_i | d_i \in D_{tmp} \wedge sim(d_i, d_r) > T_{loose}\}$
subject to condition:
$\forall d_x, d_y \in c_{canopy}\ (x \neq y) \Rightarrow \{d_z | d_z \rightarrow d_x \wedge d_z \rightarrow d_y\} = \emptyset$;
Let $c_{core} = \{d_i | d_i \in c_{canopy} \wedge sim(d_i, d_r) > T_{tight}\}$;
$D_{tmp} = D_{tmp} - c_{core}$; $C_1 = C_1 + c_{canopy}$;
End while
Output C_1;
End

Fig. 2. Algorithm of step 1

The first step (step 1) is much like the first stage of canopy clustering except that it subjects to the constraint that no any two derived instances from one same instance fall into one same canopy. Fig. 2 shows the algorithm of this step. In the algorithm, function $sim(d_i, d_r)$ computes the degree of similarity between the instance d_i and d_r.

Although each resulting cluster is constrained to contain no two derived instances of one same instance, it still can not guarantee **derived distinction** due to the existence of overlapping canopies. If two clusters, each of which contains a derived instance of one same instance, both have an instance $d_{overlap}$, this instance then actually bridges the two different derived instances when we take a transitive closure. As a result, it violates **derived distinction**.

Step 2 then is designed to ensure **derived distinction** thoroughly. it is done by checking the overlapping instances and only allowing them to be with the most similar derived instance. Fig. 3 shows the algorithm of step 2.

```
Input: Set of instances D = {d_1, d_2, ..., d_N};
       Set of clusters C_1 generated from step 1.
Output: Set of clusters C_2.
Begin
  for each d_i ∈ D do
    D_derived = {d_j | d_i → d_j};
    for any d_x, d_y ∈ D_derived (x ≠ y) do
      if ∃d_z ∈ D, c_m, c_n ∈ C_1 such that d_z, d_x ∈ c_m and d_z, d_y ∈ c_n
        let δ = sim(d_z, d_x) − sim(d_z, d_y);
        if δ > 0 then remove d_z from c_n else remove d_z from c_m;
      end if
    end for
  end for
  Output C_1 as C_2;
End
```

Fig. 3. Algorithm of step 2

The purpose of step 3 is to extract high confident duplicate pairs within each cluster in C_2 by following the definition of **primary similarity**, **derived similarity**, and **reinforced similarity**. In step 4, **boosted similarity** is implemented to extract frequent co-referenced instance pairs as potential duplicates from the clusters. The algorithms of step 3 and 4 are illustrated in Fig. 4 and Fig. 5 respectively. After all the potential duplicate pairs are extracted, a transitive closure is performed to generate the final results.

```
Input:
  Set of instances D;
  Set of clusters C_2 from step 2.
Output:
  Set of duplicate pairs P_1.
Begin
  P_1 = ∅;
  for each c_i ∈ C_2 do
    for any d_x, d_y ∈ c_i (x ≠ y) do
      //primary similarity
      //and derived similarity
      if d_x → d_m and d_y → d_n and
∃c_j ∈ C_2, c_j(d_m, d_n)
        P_1 = P_1 + (d_x, d_y) + (d_m, d_n);
      end if
      //reinforced similarity
      if d_x → d_m and d_n → d_y and
∃c_j ∈ C_2, c_j(d_m, d_n)
        P_1 = P_1 + (d_x, d_y) + (d_m, d_n);
      end if
    end for
  end for
  Output P_1;
End
```

Fig. 4. Algorithm of step 3

```
Input:
  Set of instances D;
  Set of clusters C_2 from step 2.
  Set of Pairs P_1 from step 3.
Output:
  Set of duplicate pairs P_2.
Begin
  P_2 = ∅;
  for any d_x, d_y ∈ D such that
  x ≠ y, d_x ≁ d_y do
    P_tmp = ∅;
    while ∃d_m, d_n, c such that
    d_x → d_m, d_y → d_n, c ∈ C_2, c(d_m, d_n)
do
      P_tmp = P_tmp + (d_m, d_n);
    end while
    if |P_tmp| > 1 then P_2 = P_2 + P_tmp;
  end for
  P_2 = P_2 + P_1;
  Output P_2;
End
```

Fig. 5. Algorithm of step 4

Please note the constraint rules reflected in these steps are not incompatible with other refinement processes such as GAC. They can be added in the pro-

cedure to work together with the constraint rules. For example, GAC can be added after step 2 to further refine clusters.

5.3 Computational complexity

We informally address the complexity of our approach. The algorithm in step 1 performs a constraint check that normal canopy clustering doesn't have. This extra check does about $O(km^2)$ judgements where k is the number of clusters and m is the average size of each cluster. In the setting of duplicate detection, the size of each cluster usually is not very big ($k \gg m$). The complexity of cluster adjustments in step 2 depends on the number of primary instances (p) and the average size of derived instances a primary instance has (q), which is about $O(pq^2)$. Normally, $n > p \gg q$ where n is the number of all the instances. In step 3, the extraction of potential duplicate pairs out of each cluster performs at the complexity level of $O(km^2 + km^2q^2)$ if we include the checking for the derived instances. The complexity in step 4 depends on the implementation. Our simple implementation operates at $O(p^2q^2)$. After all, it should be noted that all the above operations (checking, adjusting, extracting) don't involve very expensive computations. In fact, our experiments reveal that a lot of time is spent in computing the similarity between instances.

6 Experiments

There exist some commonly used data sets for duplicate detection experiments, but data instances in them don't have many types and in-between relations. And mostly they are presented from one unified perspective. This doesn't represent well the real world situations of relational data. Therefore, we collected data from different sources to build the data set for our experiments. The data set is mainly about papers, authors, conferences/jounals, publishers and their relations. Such data is collected from DBLP web site (`http://dblp.uni-trier.de`) and authors' home pages. These data instances are converted into a working format but types, relations and original content values are preserved. Manual labeling work is done to identify the true duplicates among the data for the purpose of evaluation of approaches in the experiments. Totally, there are 278 data instances in the data set referring to 164 unique entities. The size may not be so big, but duplicate detection in it may not be easy since there are a certain amount of different instances with very high similarity, for example, different papers within same research fields and different authors with same/similar names. The distribution of duplicates is not uniform. About two-third of instances have one or two references to their corresponding entities. The most duplicated entity has 13 occurrences.

Same as [8], we use standard metrics in information retrieval to evaluate the performance of clustering approaches for duplicate detection. They are precision, recall and F measure. Precision is defined as the fraction of correct duplicate predictions among all pairs of instances that fall in the same resulting cluster.

Recall is defined as the fraction of correct duplicate predictions among all pairs of instances that fall in the same original real duplicate cluster. F measure is the harmonic average of precision and recall.

We evaluate our approach in comparison with the canopy-based greedy agglomerative clustering approach (CB+GAC) [8]. CB+GAC also performs canopy clustering first but with no constraints. It then refine each canopy cluster using GAC: initialize each instance in the canopy to be a cluster, compute the similarity between all pairs of these clusters, sort the similarity score from highest to lowest, and repeatedly merge the two most similar clusters until clusters reach to a certain number. Table 1 shows the evaluation results of different approaches. The two threshold parameters for canopy clustering in this evaluation are set as $T_{tight} = 0.5$ and $T_{loose} = 0.35$, which are obtained through a tuning on a sampled data set. The number of clusters is then automatically determined by the two parameters. In the table, "CB+GAC" is the general clustering approach we have just discussed. "Step 12" is the approach that only performs step 1 and step 2 (refer to Section 5.2) and then returns the resulting clusters. "Step 12+GAC" is the approach that performs GAC after step 1 and step 2. "Step 1234" obviously is the approach that performs all the steps to impose all the constraints we've defined on the clusters. From the table, we can see that by incorporating constraint rules, the overall F measure improves along with the precision. In particular, when all the constraints are applied, the precision increases up to 20%, which indicates that our approach can predict duplicate with very high accuracy.

Table 1. Performance of different approaches.

Approach	Precision	Recall	F score
CB+GAC	0.717	0.806	0.759
Step 12	0.728	0.877	0.796
Step 12+GAC	0.784	0.817	0.800
Step 1234	0.921	0.721	0.809

Fig. 6 shows the sensitiveness of precision of different approaches to the loose similarity threshold (T_{loose}) in the canopy clustering. Since in our approach some constraint rules are used to extract duplicate pairs out of working clusters, the quality of the initial canopy clustering may affect the performance. That is, when T_{loose} becomes more loose, each canopy cluster may have more false duplicates, which might affect the performance of those constraint rules used for duplicate extraction. The trend of dropping precision while T_{loose} decreases is well revealed in approach "Step 12". However, the dropping trend of approach "1234" is slightly better than that of "Step 12", which means that constraint rules used in step 3 and 4 can tolerate noisy canopy clusters to certain degrees.

Table 2 shows the precision of detecting duplicated pairs in different steps in our approach. This can be used to roughly estimate contributions of different constraint rules as they are implemented in different steps. The evaluation on

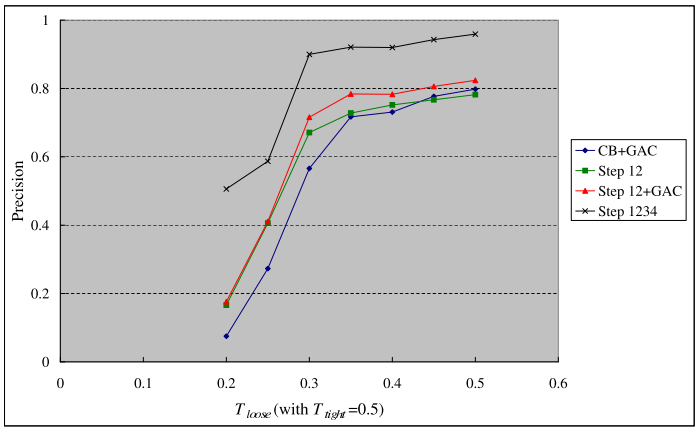

Fig. 6. Sensitiveness of precision to T_{loose} for different approaches

our data set shows that the main contribution to the improved precision is made in step 3, where constraint rules of "primary similarity", "derived similariy" and "reinforced similarity" are imposed.

Table 2. Precision of detection of duplicated pairs in different steps

	Step 1	Step 2	Step 3	Step 4
Precision	0.650	0.682	0.881	0.888

7 Conclusions and Future Works

This paper discusses the characteristics of relational data from the perspective of duplicate detection. Based on these characteristics, we have defined constraint rules, which are implemented and incorporated in our cluster-based approach. Experiments show that our approach performs well with improved accuracy in term of precision and recall. Experimental evaluations also reveal that the use of constraint rules increases the precision of duplicate detection for relational data with multiple perspectives.

One of the further studies is to conduct further experiments with larger data sets. Currently, we are keeping collecting data from different sources and converting and labeling them to build larger data sets. Besides the evaluation of accuracy on the large data sets, the efficiency of the approach will be formally evaluated.

Another further study is to design quantitative metrics to reflect characteristics of duplicated relational data. The ideal metrics will act as *soft* constraint rules. Thus, they are expected to be more adaptive to different duplicate problems.

Acknowledgements

The authors sincerely thank the anonymous reviewers for their valuable comments. The work presented in this paper was partially supported by Australian Research Council (ARC) under discovery grant DP0559213.

References

1. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association **64** (1969) 1183–1210
2. Winkler, W.E.: Methods for record linkage and bayesian networks. Technical report, U.S. Census Bureau, Statistical Research Division (2002)
3. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2002) 475–480
4. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2003) 39–48
5. Singla, P., Domingos, P.: Collective object identification. In Kaelbling, L.P., Saffiotti, A., eds.: IJCAI, Professional Book Center (2005) 1636–1637
6. Culotta, A., McCallum, A.: Joint deduplication of multiple record types in relational data. In: Fourteenth Conference on Information and Knowledge Management (CIKM). (2005)
7. Newcombe, H., Kennedy, J.M., Axford, S.J., James, A.P.: Automatic linkage of vital records. Science **130** (1959) 954–959
8. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2000) 169–178
9. Tejada, S., Knoblock, C.A., Minton, S.: Learning domain-independent string transformation weights for high accuracy object identification. In: KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2002) 350–359
10. Vapnik, V.N.: The nature of statistical learning theory. 2nd edn. Statistics for engineering and information science. Springer, New York (1999)
11. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Brodley, C.E., Danyluk, A.P., eds.: ICML, Morgan Kaufmann (2001) 282–289
12. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
13. McGuinness, D.L., Harmelen, F.v.: Owl web ontology language overview. w3c recommendation. http://www.w3.org/tr/2004/rec-owl-features-20040210 (2004)
14. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. **22**(2) (1993) 207–216
15. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5) (1988) 513–523