Identifying Value Mappings for Data Integration: An Unsupervised Approach

Jaewoo Kang[1], Dongwon Lee[2], and Prasenjit Mitra[2]

[1] NC State University, Raleigh NC 27695, USA
[2] Penn State University, University Park PA 16802, USA

Abstract. The Web is a distributed network of information sources where the individual sources are autonomously created and maintained. Consequently, syntactic and semantic heterogeneity of data among sources abound. Most of the current data cleaning solutions assume that the data values referencing the same object bear some textual similarity. However, this assumption is often violated in practice. "Two-door front wheel drive" can be represented as "2DR-FWD" or "R2FD", or even as "CAR TYPE 3" in different data sources. To address this problem, we propose a novel two-step automated technique that exploits statistical dependency structures among objects which is invariant to the tokens representing the objects. The algorithm achieved a high accuracy in our empirical study, suggesting that it can be a useful addition to the existing information integration techniques.

1 Introduction

As the Web has become the primary vehicle of information dissemination and exchange, we witness increasing numbers of databases published on the Web. No individual website, however, can satisfy the information needs of all applications. Useful information is often scattered over multiple sites. Thus, information integration across diverse sources is essential. Integrating such heterogeneous Web sources often involves two related subtasks: 1) reconciling structural heterogeneity of data by mapping schema elements across the data sources and 2) resolving semantic heterogeneity of data by mapping data instances across the tables. The first task is commonly referred to as the *schema matching* problem [2, 7, 9, 15, 17, 19, 20, 22, 23] and the second, as the *object mapping* problem [1, 3–5, 10, 12, 14, 16, 21, 28].

The object mapping problem discussed in previous literature typically refers to the problem of finding duplicate tuples within or across the tables to be integrated. Virtually all previously proposed work assume the data values in each corresponding columns are drawn from the same domain or at least they bear some textual similarity that can be measured using a string distance algorithm (e.g., edit distance, TF/IDF etc.). However, this assumption is often challenged in practice where sources use various different representations for describing their data. For example, "two-door front wheel drive" can be represented as "2DR-FWD" or "R2FD", or even as "CAR TYPE 3" in different data sources. Some

Name	Gender	Title	Degree	Status
J. Smith	M	Professor	Ph.D.	Married
R. Smith	F	T.A.	B.S.	Single
B. Jones	F	T.A.	M.S.	Married
T. Hanks	M	Professor	Ph.D.	Married

(a) Table X

Name	Gender	Title	Degree	Status
S. Smith	F	Emp10	D7	SGL
T. Davis	M	Emp3	D3	SGL
R. King	M	Emp10	D7	MRD
A. Jobs	F	Emp3	D2	MRD

(a) Table Y

Table 1. University Employee Tables.

smart string distance algorithms may be able to suggest correspondences among the first three representations, but they will fail to establish any mapping for "CAR TYPE 3" as it bears no syntactic or semantic clue except it is about car type. This problem poses a substantial challenge to the existing object mapping techniques. We name this problem as the *value mapping* problem. This paper concerns addressing this problem.

To gain insight into our approach, consider the employee tables in Table 1. Values from the *Status* column, for instance, can be easily matched using a conventional matching technique (e.g., edit distance) because the values in $Y.Status$ are abbreviations of ones in $X.Status$: ($Married \rightarrow MRD$, $Single \rightarrow SGL$). However, matching values in $Title$ and *Degree* columns is not so straightforward. Most traditional techniques relying on textual similarity will fail to identify the mapping.

To make progress in such a difficult situation, our technique exploits the co-occurrence relation between the values in each table. For example, suppose we are trying to find the value in Table 1(b) that maps to "Professor" in Table 1(a). To make the exposition simpler, let us assume that we know the correspondences between the values in the *Degree* columns (*e.g.*, $Ph.D. \rightarrow D7$, $M.S. \rightarrow D3$, $B.S. \rightarrow D2$). Intuitively, we will see a higher correlation between "Ph.D." and "Professor" than between "Ph.D." and "T.A.", as is the case in Table X. If we can measure the correlation between "D7" (which we know is "Ph.D.") and the two values, "EMP10" and "EMP3", in Table Y, and can compare the measurements across the tables, we may be able to find further correspondences. Now, let us assume that the mappings between the two *Degree* columns were not known *a priori* (i.e., the mappings for both $Title$ and *Degree* are unknown.). How should we proceed?

We propose a two-step automated technique that addresses this problem. In the first step, it constructs a statistical model that captures the correlations between all pairs of values in each table to be matched. In the second step, the constructed models are aligned in the second step such that the distance between the two models is minimized. The alignment with the minimum distance is returned as the mapping. Our algorithm is invariant to the actual tokens used to represent the objects. Moreover, the proposed technique is not dependent on any domain-specific knowledge, and thus is applicable to many different domains without training or configuration. In this paper we make the following contributions:

1. We present the value mapping problem as an important problem in information integration that is a real and hindering problem in practice.
2. We propose a matching framework that does not rely on the syntactic similarity of values and thus applicable to many different domains, even including domains to which the system has not previously been exposed.
3. The proposed algorithm can complement many existing algorithms as it utilizes different types of information that is not commonly used in the existing algorithms. The prediction result (mapping) produced by our algorithm can be combined with the results produced by other existing methods in order to improve the accuracy of the mapping.

2 Problem & Solution Overview

Problem Definition. We have two tables: the source table, S, with columns, s_1, ..., s_n, and its corresponding target table, T, with columns, t_1, ..., t_n. We focus on the value mapping problem in this paper and assume schema matching is done beforehand using existing solutions [2,7,9,15,17,19,20,22,23]; that is, the column mapping $f : s_i \rightarrow t_j$ is given. Also, the domain and range of a column c_i is denoted as $D(c_i)$ and $R(c_i)$, respectively. Then, formally, we consider the following as the **Value Mapping Problem**:

> For a pair of corresponding columns, s_i and t_j, where $f : s_i \rightarrow t_j$, find a bijective value mapping function $g : D(s_i) \rightarrow R(t_j)$ that maps two values representing the same real-world object.

Solution Overview. Our value mapping algorithm finds mappings using co-occurrence information gathered from tables without interpreting individual values. Two values are said to *co-occur* if they occur in the same row. A *co-occurrence model* captures the co-occurrence of the values in a table. An example of a simple co-occurrence model is a co-occurrence matrix. The rows and columns of it represent the set of unique values in the table, and the entries represent the co-occurrence counts of the corresponding value pairs.

Our algorithm consists of two-steps: (1) the first step, **Table2CoocurrenceModel()**, takes two table instances as input and produces corresponding co-occurrence models, and (2) the second step, **ModelMatch()**, using the co-occurrence models, produces the value mapping between the two models (i.e., a set of matching value pairs). Typically, the models are matched based on a distance metric. The distance metric captures how similar a value from one table is to another value from the other table. Optimizing on the distance metric over all pairs of matched values gives us the bijective mapping from values in one table to those in another.

3 The Matching Algorithms

3.1 Step 1: Modeling Co-occurrence Relation

We introduce the following co-occurrence models that can be used in the first step of our algorithm, **Table2CooccurrenceModel()**.

	row_1	row_2	row_3	row_4
v1:M	1	0	0	1
v2:F	0	1	1	0
v3:Professor	1	0	0	1
v4:TA	0	1	1	0
v5:Ph.D	1	0	0	1
v6:M.S	0	0	1	0
v7:B.S	0	1	0	0
v8:Married	1	0	1	1
v9:Single	0	1	0	0

(a) Value-Row Matrix T_1

	v1	v2	v3	v4	v5	v6	v7	v8	v9
v1	2	0	2	0	2	0	0	2	0
v2	0	2	0	2	0	1	1	1	1
v3	2	0	2	0	2	0	0	2	0
v4	0	2	0	2	0	1	1	1	1
v5	2	0	2	0	2	0	0	2	0
v6	0	1	0	1	0	1	0	1	0
v7	0	1	0	1	0	0	1	0	1
v8	2	1	2	1	2	1	0	3	0
v9	0	1	0	1	0	0	1	0	1

(b) Co-occurrence Matrix C_1

Table 2. Value-row matrix and Co-occurrence matrix of Table X.

Simple Vector Model. Consider tables in Table 1. Suppose we want to find the value mapping between terms used in three columns, *Gender*, *Title*, and *Degree* across the two tables. One simple solution is to use frequencies of values such that values appearing more often correspond to each other while values appearing less often correspond to each other. This approach, however, may not work when values are evenly distributed (e.g., *Gender*) or when the frequencies of several values in a column are very similar. To remedy this problem, *Simple Vector Model* considers pair-wise term frequencies.

We first generate a "value-row" matrix for each table instance as shown in Table 2. Table 2(a) shows a value-row matrix T_1 generated from tables X. A column vector of T_1 corresponds to a matching row in table X, and a row vector of T_1 encodes occurrences of the corresponding value in each row of table X. For example, T_1("M", row_1) is 1 because "M" occurs in row_1 of table X. The value-row matrix T_2 of Table Y can be constructed similarly. With the value-row matrix, one can easily calculate pair-wise co-occurrences by taking product of a matrix T_i and its transpose T_i^T: $C_i = T_i \times T_i^T$.

The co-occurrence matrix C_i captures pair-wise value co-occurrences between all pairs of values in the corresponding table. Table 2(b) shows the co-occurrence matrix C_1 for tables X. Now let us assume that value mappings of three columns are known as follows: $Gender(M \rightarrow M, F \rightarrow F)$, $Status(Married \rightarrow MRD$, $Single \rightarrow SGL)$, and $Degree(Ph.D \rightarrow D7$, $M.S \rightarrow D3$, $B.S \rightarrow D2)$. Given such mappings and the co-occurrence matrices, how can one find the remaining value mappings – the ones between the values in the *Title* columns?

We could, perhaps, first align the rows and columns of C_1 and C_2 according to the known mappings, and then, for each term vector (row) in C_1, try to find the closest term vector from C_2 by comparing only the parts that we know are correctly aligned. For example, suppose we are trying to find a value in table X that corresponds to "Emp10" in table Y. First we take the term vector of "Emp10" from C_2 (not shown): C_2("Emp10") = [1 1 2 0 2 0 0 1 1]. The third and fourth entries are for "Emp10" and "Emp7" for which we do not know the correct mapping across the tables; so we ignore them and keep only entries for known values as follows: C_{2a}("Emp10") = [1 1 2 0 0 1 1]. Then, if we measure the

typical Euclidean distances between C_{2a}("Emp10") and the two term vectors corresponding to the terms "Professor" and "TA" from C_1, we get 2 for the "Emp10-Professor" pair and 2.83 for the "Emp10-TA" pair, suggesting that the "Emp10-Professor" pair is a better match.

One way to improve the performance of this scheme is to weight terms according to their *information content*. That is, rare terms carry more weights when they co-occur than terms that occur frequently (e.g., "male" or "female"). In this work, we use a standard inverse document frequency weighting [6]. The weighted value-row matrix is defined as follows: $T(i,j) = 1 - \frac{k}{N}$, if term i occurs in row j, or 0 otherwise, where k is the number of rows where term i occurs and N is the total number of rows in the table. Note that when term i occurs in all rows, $T(i,j)$ is 0 for all j. We incorporated the weighting scheme in all the models for our experimentation, but have not weighted the examples in this paper to retain their simplicity.

Co-occurrence Matrix Model. One straightforward extension of the simple vector model is to compare two co-occurrence matrices as a whole rather than limiting the focus to only vector-wise comparisons. This approach works as follows. We compute the distances between the two co-occurrence matrices while we permute the second matrix, and find the permutation that minimizes the distance between the two matrices. The algorithm then returns the permutation as the proposed mapping.

Latent Semantic Model. *Latent Semantic Indexing* (LSI) [6] is an information retrieval technique introduced to address the inherent difficulty of handling semantic heterogeneity in traditional inverted-index based text indexing schemes. For example, if a user asks for documents about "human computer interface," then traditional inverted-index based techniques with term overlap measures will fail to return documents that using "HCI" instead of its full wording because the query terms are not appearing in these documents. LSI tries to overcome this shortcoming by exploiting the co-occurrence information. For example, if "HCI" frequently co-occurs with "user" and "interaction" and the "user" and "interaction" co-occur frequently with "human computer interface", it may implies that "HCI" and "human computer interface" are semantically relevant.

LSI uses *Singular Value Decomposition* (SVD) for its co-occurrence analysis. SVD decomposes a value-row matrix T into the product of three distinct matrices: $T = U \times S \times V^T$, where S is a diagonal matrix that contains singular values in a decreasing order, and U and V are orthogonal matrices that contain corresponding left and right singular vectors (principal components), respectively.

An interesting property of SVD is that we can use it to reduce the noise in the model. Data from databases often contain errors due to various reasons. These errors result in very small singular values occurring in the diagonal of matrix S. We can eliminate the effect of these errors by throwing away the small singular values, thereby reducing the dimensionality of U and V. The side effect of this dimensionality reduction is that the resulting model captures indirect multi-level co-occurrence patterns which are not apparent in the original co-

occurrence matrix. The *Latent Semantic Model* is built upon this result. Using SVD, co-occurrence matrix C_1 can be calculated as follows: $C_1 = T_1^k \times (T_1^k)^T$ where T_1^k is a best rank-k approximation of T_1 obtained by $U \times S \times V^T$ using only top k principal components and singular values (see [13] for more details).

3.2 Step 2: Matching Models

Matching process for the Simple Vector Model is relatively easy as the matching problem naturally reduces to a linear assignment problem (e.g., bipartite-graph matching problem). We use the Hungarian method [18] to solve the linear assignment problem. For the two matrix models, we first define a distance metric to measure the closeness of the models, and discuss how to match the values in the two tables using this distance metric. In our implementation, we used the Euclidean distance metric as a distance measure between the two models as follows:

Definition 1 (Euclidean Distance Metric) *Let A and B be two co-occurrence matrices with the same dimensions, and a_{ij} and b_{ij} be the co-occurrences between the i^{th} and j^{th} values of A and B, respectively. Let m be a mapping function that maps an index of a value in A to the index of a matching value in B. Then, the Euclidean Distance Metric for matrices A and B is: $D(A, B) = \sqrt{\Sigma_{i,j}(a_{ij} - b_{m(i)m(j)})^2}$* □

To find the correspondences between the values in the two tables, we can simply find the mapping function m that minimizes the Euclidean distance between the two co-occurrence matrices of corresponding tables. Given the distance metric, our problem is nicely re-cast into a form of the graph matching problem. A large volume of literature has devoted to finding efficient and accurate graph matching techniques – exact or approximate.

For both the co-occurrence matrix model and the latent semantic model, we use the Hill-climbing algorithm. The Hill-climbing algorithm is a greedy approach such that it moves, in each state transition, to a state where the most improvement can be achieved. A state represents a permutation that corresponds to a mapping between the two graphs. We limit the set of all states reachable from one state in a state transition, to a set of all permutations obtained by one swapping of any two nodes in the current state. The algorithm stops when there is no next state available better than the current state. As we can see, the Hill-climbing algorithm is non-deterministic; depending on where it starts, even for the same problem, the final states may differ. In our experiments, we ran the Simple Vector Model first, and then ran the Hill-climbing algorithm using the result of the Simple Vector Model as its starting point.

4 Validating the Framework

4.1 Set-up

We implemented our two-step matching algorithm using Matlab 6.5. Experiments were performed using a machine running Windows XP Professional with

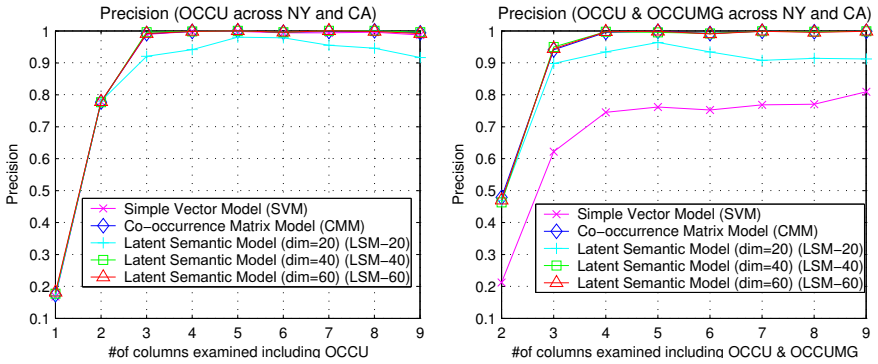
Col. No.	Col. name	# of unique values	Description
1	OCCU	45	occupation code
2	OCCUMG	14	occupation - major group
3	INDU	48	industry code
4	INDUMG	23	industry - major group
5	COWORK	8	class of worker (fed., priv., etc.)
6	EMPSTA	8	employment status
7	FLTPRT	4	full time, part time, etc.
8	AGE	91	age
9	EDUCA	17	highest degree earned
10	MARITL	7	marital status
11	AFEVER	5	army veteran
12	FAMINC	16	total family income
13	LABUNI	3	union member? (y/n/null)
14	SEX	2	sex
15	NUMHOU	12	# of household members
16	RACE	4	race
17	TENURE	3	own / rent / null
18	HOUSTY	5	household type
Total # of unique values for each table is 315			

Table 3. Census table summary: NY and CA table characteristics

2.4Ghz Pentium 4 and 1 GB of memory. We ran our tests using census data tables obtained from the website U.S. Census Bureau[3]. We used two state-census-data files, "CA" and "NY", in our experiments. The CA table contains approximately 10,000 tuples and NY table contains about 7,500 tuples. Each table has 18 columns; there exists a one-to-one correspondence between the columns across the tables. Table 3 shows the number of common unique values in each column of the two tables. For example, the OCCU column has 45 unique values that are common to both tables. There are small numbers of low frequency values (11, out of total of 641 unique values from both the tables) that appear only in one side of the tables, either in NY or CA; we ignore them because in this work, we limit our focus to the problems where one-to-one correspondences exist between the values to be mapped.

In experiments, we ran tests for both non-weighted and weighted cases (discussed in Section 3). However, since weighted models with the standard inverse document frequency weighting outperformed non-weighted models on average by 15%, we present the results of only weighted models in the following. We use the precision to measure the effectiveness of various algorithms as follows: $Precision = \frac{\#\ of\ correct\ mappings\ by\ an\ algorithm}{\#\ of\ true\ mappings}$. The number of true mappings is the total number of mappings we know exist from our encoding process. We then divide the number of correct mappings produced by an algorithm in testing by this number (#of true mappings) to calculate the precision.

[3] http://dataferrett.census.gov/TheDataWeb/index.html



(a) Matching across two OCCU columns (b) Matching OCCU & OCCUMG.

Fig. 1. Precision graphs.

4.2 Experiments

We iterated the tests 20 times at each data point with random samples of 6,500 tuples chosen from each table, and averaged the results. Since to the best of our knowledge, there is no other known un-interpreted value mapping algorithm, we compared five algorithms of ours against each other: Simple Vector Model (SVM), Co-occurrence Matrix Model (CMM), and three Latent Semantic Models with ranks 20, 40 and 60 (LSM-20, -40, -60, respectively).

Figure 1(a) presents the results of mapping between the two OCCU (occupation code) columns across the tables. We ran the experiment while incrementally adding more columns (whose mappings between the values are known) to the tables in each step. The left-most data point (x=1) in Figure 1(a) represents the precision of the matching where only the two OCCU columns were given to the algorithm. All five algorithms that we compared yielded almost identical results on the first data point because no co-occurrence information is available to exploit.

The second data point (x=2) shows the results of the test where we compared the same OCCU pairs, but with extra information of mappings between OCCUMG pairs, i.e., the correct mappings of OCCUMGs are given to the algorithms. The precisions of the algorithms improved to around 78% from 18%. Note that the OCCUMG column contains the codes of major occupation groups while OCCU contains detailed occupation codes. It is very likely that significant amount of information about the contents of the OCCU column is captured in the contents of the OCCUMG column.

As shown in Figure 1(a), the match precision reached almost 100% at the third data point (x=3) for all but Latent Semantic Model with dimension 20 (LSM-20), after taking the third column (INDU) into account. It appeared that the added information given from the two extra columns was just enough for finding the correct mapping for all 45 unique values between the two OCCU columns. Note that the total number of possible mappings in this test was 45!.

Considering the gigantic search space, the result was quite encouraging. For each test in the remaining data points from 4 to 9, similarly we added extra columns by one at a time in each step in the order of columns in Table 3.

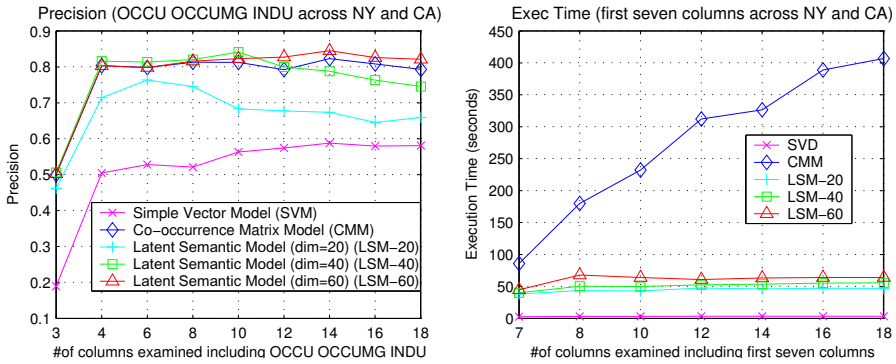
Now, assume that mappings for both OCCU and OCCUMG are unknown. How would the performance of each algorithm change? Figure 1(b) shows the result. There are two notable differences in this result from the previous result. First, the performance of SVM deteriorated significantly; it only achieved up to 80% accuracy while the other previous top performers still yielded close to 100% accuracy as before. Second, the performance of the three top performers at the first data point (x=2) improved significantly from approximately 18% accuracy (x=1 in Figure 1(a)) to 48% accuracy, where only the two unknown columns, OCCU and OCCUMG, were given. In contrast, the total number of unknown values increased to 59 (sum of the numbers of unique values in OCCU and OCCUMG) from 45 (unique values in OCCU), and the size of the search space increased to $45! \times 14!$ from 45!. What accounted for this performance improvement?

One explanation to the question can be found from the models themselves. Recall that both CMM and LSM consider the co-occurrences between the pairs of unknown values as well as those between the pairs of the unknown and known values, while SVM only considers the co-occurrences of the unknown and known value pairs. Although the mappings of the two columns (OCCU and OCCUMG) were not given *a priori*, their strong correlations made it easier to find the correct mappings. The same explanation applies to the first difference where SVM deteriorated significantly while the others did not. SVM was penalized by not considering the value co-occurrences of the two unknown columns whose values are highly correlated.

The three top performers, CMM, LSM-40, and LSM-60, yielded almost identical performance results. They achieved 100% accuracy at the third data point (x=4) where we had two extra columns, INDU and INDUMG, in to consideration. Let us now consider the case where we have three columns unknown: OCCU, OCCUMG and INDU. The results of the tests are shown in Figure 2(a). The results shown on the first data point (x=3) are from running the algorithms with just the three unknown columns; the subsequent data points show the results with increasing numbers of extra columns, as usual.

In addition to the three graphs reported in this paper, we performed additional tests with up to seven unknown columns. The performance order of the algorithms was same to the three column unknown test shown in Figure 2 and the best performer was LSM-60. There was no severe performance degradation among the top performing algorithms. LSM-60 still achieved slightly over 70% accuracy in the seven column unknown test.

Another important criterion in choosing right algorithms could be the computational complexity. Figure 2(b) shows the five algorithms' execution times measured in the test where we mapped the first seven columns across the tables.Unlike CMM, LSM showed a stable performance over the full range of tests. This result is quite striking because both LSM and CMM run the same Hill-



(a) Matching OCCU, OCCUMG & INDU (b) Execution time

Fig. 2. Result of matching three unknown columns: OCCU, OCCUMG & INDU and execution time.

climbing algorithm in the course of matching the two models of the corresponding tables. Moreover, LSM has an additional cost of doing Singular Value Decomposition (SVD) while CMM does not. Apparently in our test, the noise canceling helped to speed up the Hill-climbing process by smoothing the gradient search space of the Hill-climbing algorithm.

5 Related Work

There are vast amount of related work to ours – notably (1) *schema matching* (e.g., [2, 7, 9, 15, 17, 19, 20, 22, 23]) and (2) *object mapping* (e.g., [1, 3–5, 10, 12, 14, 16, 21, 28]) problems.

(1) In this work, we assumed that we knew the correspondences between the columns across the tables. However, in general settings, such correspondences are not known but have to be found first. To generate such correspondences, various schema matching techniques have been proposed. Some employs Machine Learning (e.g., LSD [9]), rules (e.g., TranScm [23]), Neural Network (e.g., SemInt [19]), structural similarity (e.g. Cupid [20]), or interactive user feedback (e.g., Clio [15]). Recent development (e.g., iMAP [7]) even enables to find not only 1-1, but also more complex *n-m* schema matches (e.g., `name` = concat(`first`, `last`) or `euro` = 1.32 × `dollar`). For a good survey and comparison, see [8, 26]. In particular, our proposals in this paper is an extension of authors' previous attempt [17] to the object mapping problem – in [17], the schema matching problem in the presence of opaque column names and data values are addressed.

(2) Our "value mapping" problem is more closely related to the object mapping problem (i.e., value is the object to map), which is also known as various names in diverse contexts: e.g., record linkage [11, 28], citation matching [21, 25], identity uncertainty [25], merge-purge [16], duplicate detection [1, 24, 27], and

approximate string join [5,14]. Common to all these is the problem to find similar objects (e.g., values, records, tuples, citations). Although different proposals have adopted different approaches to solve the problem in different domains, by and large, they focus on syntactic similarities of objects under comparison. On the other hand, our value mapping solutions can identify mappings where two objects have little syntactic similarity. To cope with such difficulties, we proposed to explore statistical characteristics of objects such as co-occurrence frequency or entropy.

6 Conclusion

In this paper, we investigated the value mapping problem to locate matching pairs of values from two information sources. We proposed a two-step matching algorithm to discover the mapping. Since our algorithm does not depend on the syntactic distance among values, it works well for opaque or domain-specific values encountered while integrating heterogeneous sources. Our algorithm uses value co-occurrences as the basis for establishing value mappings. Specifically, we proposed three co-occurrence models – Simple Vector, Co-occurrence Matrix, and Latent Semantic Models – and conducted an extensive experimentations to find their effectiveness. The Latent Semantic Model performed slightly better than the Co-occurrence Matrix Model and both outperformed the Simple Vector Model. The two algorithms identified unknown value mappings with 70-100% accuracy. We believe our algorithms form an important contribution towards enabling automated integration of information from diverse websites.

References

1. R. Ananthakrishna, S. Chaudhuri, and V. Ganti. "Eliminating Fuzzy Duplicates in Data Warehouses". In *VLDB*, 2002.
2. P. Andritsos, R. J. Miller, and P. Tsaparas. Information-theoretic tools for mining database structure from large data sets. In *ACM SIGMOD*, 2004.
3. M. Bilenko and R. J. Mooney. "Adaptive Duplicate Detection Using Learnable String Similarity Measures". In *ACM KDD*, Washington, DC, 2003.
4. S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. "Robust and Efficient Fuzzy Match for Online Data Cleaning". In *ACM SIGMOD*, 2003.
5. W. W. Cohen. "Integration of Heterogeneous Databases Without Common Domains using Queries based on Textual Similarity". In *ACM SIGMOD*, 1998.
6. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. "Indexing by Latent Semantic Analysis". *J. of the American Society of Information Science*, 41(6):391–407, 1990.
7. R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos. "iMAP: Discovering Complex Mappings between Database Schemas". In *ACM SIGMOD*, 2004.
8. H. Do, S. Melnik, and E. Rahm. "Comparison of Schema Matching Evaluations". In *GI-Workshop "Web and Databases"*, Oct. 2002.
9. A. Doan, P. Domingos, and A. Y. Halevy. "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach". In *ACM SIGMOD*, 2001.

10. A. Doan, Y. Lu, Y. Lee, and J. Han. "Object Matching for Data Integration: A Profile-Based Approach". In *Workshop on Info. Integration on the Web*, 2003.
11. I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 64:1183–1210, 1969.
12. H. Galhardas, D. Florescu, D. Shasha, and E. Simon. "An Extensible Framework for Data Cleaning". In *IEEE ICDE*, 2000.
13. G. H. Golub and C. F. van Loan. *"Matrix computations"*. The Johns Hopkins University Press, 1999.
14. L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins for Data Cleansing and Integration in an RDBMS". In *IEEE ICDE*, 2003.
15. M. A. Hernandez, R. J. Miller, and L. M. Haas. "Clio: A Semi-Automatic Tool for Schema Mappiong". In *ACM SIGMOD*, 2001.
16. M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.
17. J. Kang and J. F. Naughton. "On Schema Matching with Opaque Column Names and Data Values". In *ACM SIGMOD*, San Diego, CA, Jun. 2003.
18. H. W. Kuhn. "The Hungarian Method for the Assignment Problem". *Naval Research Logistics Quarterly*, 2:83–97, 1955.
19. W.-S. Li and C. Clifton. "SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases using Neural Networks". *VLDB J.*, 10(4), Dec. 2001.
20. J. Madhavan, P. A. Bernstein, and E. Rahm. "Generic Schema Matching with Cupid". In *VLDB*, 2001.
21. A. McCallum, K. Nigam, and L. H. Ungar. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *ACM KDD*, Boston, MA, 2000.
22. S. Melnik, H. Garcia-Molina, and E. Rahm. "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching". In *IEEE ICDE*, 2002.
23. T. Milo and S. Zohar. "Using Schema Matching to Simplify Heterogeneous Data Translation". In *VLDB*, 1998.
24. A. E. Monge. *"Adaptive Detection of Approximately Duplicate Database Records and the Database Integration Approach to Information Discovery"*. PhD thesis, University of California, San Diego, 1997.
25. H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. "Identity Uncertainty and Citation Matching". In *Advances in Neural Information Processing Systems*. MIT Press, 2003.
26. E. Rahm and P. A. Bernstein. "On Matching Schemas Automatically". *VLDB J.*, 10(4), Dec. 2001.
27. S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM SIGMOD*, 2002.
28. W. E. Winkler. "The State of Record Linkage and Current Research Problems". Technical report, US Bureau of the Census, Apr. 1999.