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, 14] (see [18] for survey) and the second, as the *object mapping* problem [1, 3–5, 8, 11, 12, 17, 19, 20].

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, Jaccard). However, this assumption is often challenged in practice where sources use various different representations for describing their data. This problem poses a substantial challenge to the existing object mapping techniques. We name this problem as the *value mapping* problem.

In order to address the value mapping problem, in our previous work [13], we introduced an iterative and interactive framework where the system incrementally builds the mapping through iteration while incorporating the user feedback. In this work we propose an unsupervised technique that can automatically produce a highly accurate subset of mappings which can serve as the seed mappings for the subsequent iterative mapping process. The proposed technique can work as a stand-alone process as well as a preprocessor generating seed mappings in our iterative framework. It works in two steps as follows. 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 such that the distance between the two models is minimized. The alignment with the minimum distance is returned as the mapping. In what follows, we formally define the problem and give the 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 [18]; 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.

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.

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.

	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.

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 (i.e., 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 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()**.

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: $C_i = T_i \times T_i^T$.

This 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 table X. The co-occurrence matrix, C_2, for table Y can be constructed similarly. 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 (e.g., 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.

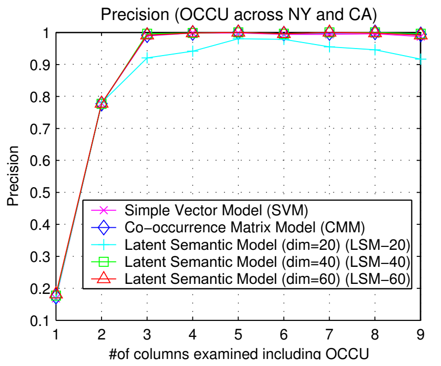
LSI uses *Singular Value Decomposition* (SVD) for its co-occurrence analysis. SVD decomposes a value-row matrix T into the product of three distinct matrices: $T = U \times S \times V^T$, where S is a diagonal matrix that contains singular values in a decreasing order, and U and V are orthogonal matrices 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 [10] for more details).

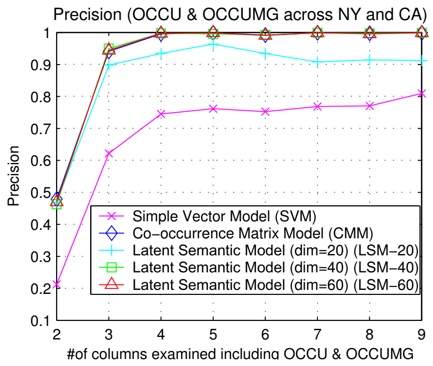
3 Step 2: Matching Models

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 [15] to solve the linear assignment problem.

For both the co-occurrence matrix model and the latent semantic model, we use the Hill-climbing algorithm to align the models being matched. It optimizes on the Euclidean distance between the models to find the matches. 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.



(a) Matching across two OCCU columns



(b) Matching OCCU & OCCUMG.

Fig. 1. Precision graphs.

4 Validating the Framework

Set-up: We implemented our two-step matching algorithm using Matlab 6.5. Experiments were performed using a machine running Windows XP Professional with 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. 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. 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. 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. For comparison, we implemented five variations: Simple Vector Model (SVM), Co-occurrence Matrix Model (CMM), and three Latent Semantic Models with ranks 20, 40 and 60 (LSM-20, LSM-40, LSM-60, respectively).

Experiments: 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

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

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. Figure 1(b) shows a similar result from the experiment where the mappings of the two columns (OCCU and OCCUMG) are unknown. In this test, the performance of SVM deteriorated significantly while the other matrix models sustained.

In addition to the two graphs reported in this paper, we performed additional tests with up to seven unknown columns. Throughout the test, LSM-60 was the best performer while CMM was a close second. There was no severe performance degradation among the top performing algorithms. LSM-60 achieved slightly over 70% accuracy in the seven column unknown test. We also compared the computational complexity of the algorithms. CMM was the slowest among all, taking more than 400 seconds to finish the seven column mapping test (with all extra columns used). On the other hand, LSMs showed a stable performance over the course of tests, taking only 60 seconds to finish the same test. This result is quite striking because both LSM and CMM run the same Hill-climbing algorithm to align the models. Moreover, LSM has an additional cost of doing Singular Value Decomposition (SVD) while CMM does not. We speculated that the noise canceling helped speed up the Hill-climbing process by smoothing the gradient search space of the Hill-climbing algorithm.

5 Related Work and Conclusion

Our “value mapping” problem is closely related to the object mapping problem, which is also known as various names in diverse contexts: e.g., record linkage [9,20], citation matching [16,17], identity uncertainty [17], merge-purge [12], duplicate detection [1, 19], and approximate string join [5, 11]. 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. We believe our algorithms can complement many existing object mapping algorithms as it exploits information that is not used by the existing algorithms.

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. 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.
9. I. P. Fellegi and A. B. Sunter. "A Theory for Record Linkage". *J. of the American Statistical Society*, 64:1183–1210, 1969.
10. G. H. Golub and C. F. van Loan. *"Matrix computations"*. The Johns Hopkins University Press, 1999.
11. L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. "Text Joins for Data Cleansing and Integration in an RDBMS". In *IEEE ICDE*, 2003.
12. M. A. Hernandez and S. J. Stolfo. "The Merge/Purge Problem for Large Databases". In *ACM SIGMOD*, 1995.
13. J. Kang, T. S. Han, D. Lee, and P. Mitra. "Establishing Value Mappings using Statistical Models and User Feedback". In *ACM CIKM*, 2005.
14. J. Kang and J. F. Naughton. "On Schema Matching with Opaque Column Names and Data Values". In *ACM SIGMOD*, San Diego, CA, Jun. 2003.
15. H. W. Kuhn. "The Hungarian Method for the Assignment Problem". *Naval Research Logistics Quarterly*, 2:83–97, 1955.
16. 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.
17. 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.
18. E. Rahm and P. A. Bernstein. "A survey of approaches to automatic schema matching". *VLDB J.*, 10(4), Dec. 2001.
19. S. Sarawagi and A. Bhamidipaty. "Interactive Deduplication using Active Learning". In *ACM SIGMOD*, 2002.
20. W. E. Winkler. "The State of Record Linkage and Current Research Problems". Technical report, US Bureau of the Census, Apr. 1999.