Discovering Matching Dependencies*

Shaoxu Song
Hong Kong University of Science and Technology, Hong Kong
sshaoxu@cs.ust.hk

Lei Chen
Hong Kong University of Science and Technology, Hong Kong
leichen@cs.ust.hk

ABSTRACT

Matching dependencies (MDs) are recently proposed for various data quality applications such as detecting the violation of integrity constraints and duplicate object identification. In this paper, we study the problem of discovering matching dependencies for a given database instance. First, we formally define the measures, support and confidence, for evaluating the utility of MDs in the given database instance. Then, we study the discovery of MDs with certain utility requirements of support and confidence. Exact algorithms are developed, together with pruning strategies to improve the time performance. Finally, our experimental evaluation demonstrates the efficiency of the proposed methods.

Categories and Subject Descriptors: H.2.0 [Database Management]: General

General Terms: Algorithms

1. INTRODUCTION

To make dependencies adapt to this real-world scenario, i.e., to be tolerant of various representation formats, Fan [4] proposed a new concept of data dependencies, called *matching dependencies* (MDs). Informally, a matching dependency targets on the fuzzy values like text attributes and defines the dependency between two set of attributes according to their matching quality measured by some matching operators (see [1] for a survey), such as *Euclidean distance* and *cosine similarity*. For example, considering the Contacts relation in Table 1, we may have a MD as

$$\mathsf{md}_1 \quad : \quad ([\mathsf{Street}] \rightarrow [\mathsf{City}], < 0.8, 0.7 >)$$

which states that for any two tuples from Contacts, if they agree on attribute Street (the matching similarity, e.g. *cosine similarity*, on the attribute Street is greater than a threshold 0.8), then the corresponding City attribute should match as well (i.e. similarity on City is greater than the corresponding threshold 0.7).

MDs can be applied in many tasks [4]. For example, in data cleaning, we can use MDs to detect the inconsistent data, that is, data that do not follow the constraint (rule) specified by MDs. For example, according to the above md example, for any two tuples t_i and t_j having similarity greater than 0.8 on Street, they should be matched on City as well (similarity ≥ 0.7). If their City similarity is less than 0.7, then there must be something wrong in t_i and t_j, i.e., inconsistency. Such inconsistency on text attributes cannot be detected by using FDs and extensions based on exact matching. In addition to locating the inconsistent data, object identification, another important work for data cleaning, can also employ MDs as matching rules [5]. For instance, according to

$$\mathsf{md}_2 : ([\mathsf{Name}, \mathsf{Street}] \rightarrow [\mathsf{SIN}], < 0.9, 0.9, 1.0 >)$$

if two tuples have high similarities on Name and Street (both similarities are greater than 0.9), then these two tuples probably denote the same person in the real world, i.e., having the same SIN.

Table 1: Example of Contacts relation $\mathcal{R}$

SIN	Name	CC	ZIP	City	Street	
584	Claire Green	44	606	Chicago	No.2, Central Rd.	t_1
584	Claire Greem	44	606	Chicago	No.2, Central Rd.	t_2
584	Claire Gree_	44	606	Chicago	#2, Central Rd.	t_3
265	Jason Smith	01	021	Boston	No.3, Central Rd.	t_4
265	J. Smith	01	021	Boston	#3, Central Rd.	t_5
939	W. J. Smith	01	021	Chicago	#3, Central Rd.	t_6

Though the concept of matching dependencies is given in [4], the authors did not discuss how to discover useful MDs. In fact, given a database instance, there are enormous MDs that can be discovered if we set different similarity thresholds on attributes. Note that if all thresholds are set to 1.0, MDs have the same semantics as traditional FDs, in other words, traditional FDs are special cases of MDs. For instance, the above fd can be represented by a MD $([\mathsf{ZIP}] \rightarrow [\mathsf{City}], < 1.0, 1.0 >)$. Clearly, not all the settings of thresholds for MDs are useful.

The utility of MDs in the above applications is often evaluated by *confidence* and *support*. Specifically, we consider a MD of a relation $\mathcal{R}$, denoted by $\varphi(X \rightarrow Y, \lambda)$, where X and Y are the attribute sets of $\mathcal{R}$, λ is a pattern specifying different similarity thresholds on each attribute in X and Y. Let λ_X and λ_Y be the projections of thresholds in pattern λ on the attributes X and Y respectively. The *support* of φ is the proportion of tuple pairs whose matching similarities are higher than the thresholds in φ on both attributes of X and Y. The *confidence* is the ratio of tuple pairs whose matching similarities satisfy λ_X also satisfying λ_Y. In real applications like inconsistency detection, in order to achieve high detection accuracy, we would like to use MDs with high confidence. On the other hand, if users need high recall of detection, then MDs with high support are preferred. Intuitively, we would like to discover those MDs with high support and high confidence. Therefore, in this work, we would like to discover proper settings of matching

*Funding for this work was provided by the Hong Kong RGC grant No. 611608

CIKM'09, November 2–6, 2009, Hong Kong, China.

similarity thresholds for MDs, which can satisfy users' utility requirements of support and confidence.

Contributions. In this paper, given $X \rightarrow Y$ and a relation instance, we study the issues of discovering matching dependencies. Our main contributions are summarized as follows:

First, we propose the utility evaluation of *matching dependencies*. Specifically, the confidence and support evaluations of MDs are formally defined. To the best of our knowledge, this is the first paper to study the utility evaluation and discovery of MDs.

Second, we study the algorithms for discovering MDs. The MDs discovery problem is to find settings of matching similarity thresholds on attributes X and Y for MDs that can satisfy the required confidence and support. We first present an exact solution and then study pruning strategies by the minimum requirements of support.

Third, we report an extensive experimental evaluation. Proposed algorithms on discovering MDs are studied. Our pruning strategies can significantly improve the efficiency in discovering MDs.

Related Work. The concept of matching dependencies (MDs) is first proposed in [4] for specifying matching rules for the object identification (see [3] for a survey). The MDs can be regarded as a generalization of FDs, which are based on identical values having matching similarity equal to 1.0 exactly. Thus, FDs can be represented by the syntax of MDs as well. For any two tuples, if their X values are identical (with similarity threshold 1.0), then a FD ($X \rightarrow Y$) requires that their Y values are identical too, i.e., a MD ($X \rightarrow Y, < 1.0, 1.0 >$). Koudas et al. [7] also study the dependencies with matching similarities on attributes Y when given the *exactly* matched values on X, which can be treated as a special case of MDs. The reasoning mechanism for deducing MDs from a set of given MDs is studied in [5]. The MDs and their reasoning techniques can improve both the quality and efficiency of various record matching methods.

2. UTILITY MEASURES

In this section, we formally introduce the definitions of MDs. Then, we develop utility measures for evaluating MDs over a given database instance.

Traditional functional dependencies FDs and their extensions rely on the exact matching operator $=$ to identify dependency relationships. However, in the real world application, it is not possible to use exact matching operator $=$ to identify matching over fuzzy data values such as text values. For instance, Jason Smith and J.Smith of attribute Name may refer to the same real world entity. Therefore, instead of FDs on identical values, the *matching dependencies* MDs [4] are proposed based on the matching quality.

Consider a relation $\mathcal{R}(A_1, \ldots, A_M)$ with M attributes. Following similar syntax of FDs, we define MDs as following: [1]

DEFINITION 1. *A* matching dependency (MD) *φ is a pair $(X \rightarrow Y, \lambda)$, where $X \subseteq \mathcal{R}, Y \subseteq \mathcal{R}$ are two sets of attributes, and λ is a* threshold pattern *of matching similarity thresholds on attributes in $X \cup Y$, e.g., $\lambda[A]$ denotes the matching similarity threshold on attribute A.*

A MD φ specifies a constraint on the set of attributes X to Y. Specifically, the constraint states that, for any two tuples t_1 and t_2 in a relation instance r of $\mathcal{R}$, if $\bigwedge_{A_i \in X} t_1[A_i] \approx_{\lambda[A_i]} t_2[A_i]$, then $\bigwedge_{A_j \in Y} t_1[A_j] \approx_{\lambda[A_j]} t_2[A_j]$, where $\lambda[A_i]$ and $\lambda[A_j]$ are the *matching similarity thresholds* on the attributes of A_i and A_j respectively. In the above constraint, for each attribute $A_i \in X \cup Y$, the similarity matching operator $\approx$ returns true, if the similarity between $t_1[A_i]$ and $t_2[A_i]$ satisfies the corresponding threshold $\lambda[A_i]$. For example, a MD $\varphi([\mathsf{Street}] \rightarrow [\mathsf{City}], < 0.8, 0.7 >)$ in the Contacts relation denotes that if two tuples have similar Street (with matching similarity greater than 0.8) then their City values are probably similar as well (with similarity at least 0.7).

Like FDs and CFDs [6, 2], we adopt *support* and *confidence* measures to evaluate the matching dependencies. According to the above constraint of MDs, we need to consider the matching quality (e.g., cosine similarity or edit distance) of any pair of tuples t_1 and t_2 for $\mathcal{R}$. Therefore, we compute a statistical distribution (denoted by $\mathcal{D}$) of the matching quality of pair-wised tuple matching for $\mathcal{R}$. The statistical distribution has a schema $\mathcal{D}(A_1, \ldots, A_M, P)$, where each attribute A_i in $\mathcal{D}$ corresponds to the matching quality values on the attribute A_i of $\mathcal{R}$, and P is the statistical value. Let s be a statistical tuple in $\mathcal{D}$. The statistic $s[P]$ denotes the probability that any two tuples t_1 and t_2 of $\mathcal{R}$ have the matching quality values $s[A_i], \forall A_i \in \mathcal{R}$. With a pair-wised evaluation of matching quality of all the N tuples for $\mathcal{R}$, we can easily compute P by $\frac{count(s)}{N*(N-1)/2}$, where $count(s)$ records the pairs of tuples having matching quality s. Different matching operators have various spaces of matching values, such as cosine similarity in $[0.0, 1.0]$ while edit distance having edit operations $1, 2, \ldots$. In order to evaluate in a consistent environment, we map these matching quality values $s[A]$ to a unified space, say $[0, d-1]$, which is represented by $\mathsf{dom}(A)$ with d elements. Table 2 shows an example of the statistical distribution $\mathcal{D}$ computed from Contacts in Table 1 by mapping the cosine similarities in $[0.0, 1.0]$ to elements in $[0, d-1]$ of $\mathsf{dom}(A)$ with $d = 10$. According to $\mathsf{dom}(A)$ in our example, the first tuple $(1, 0, 3, \ldots, 0.065)$ denotes that there are about 6.5% matching pairs in all pair-wised tuple matching, whose similarities are $1, 0, 3, \ldots$ on the attribute $A_1, A_2, A_3, \ldots$ respectively.

Table 2: Example of statistical distribution $\mathcal{D}$

A_1	A_2	A_3	A_4	A_5	A_6	P	
1	0	3	5	8	4	0.065	s_1
7	4	0	0	4	1	0.043	s_2
0	4	8	1	6	2	0.124	s_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Then, we can measure the support and confidence of MDs, with various attributes X and Y, based on the statistical distribution $\mathcal{D}$. Let λ_X and λ_Y be the projections of matching similarity threshold pattern λ on the attributes of X and Y respectively in a MD φ, which are also specified in terms of elements in $\mathsf{dom}(A)$ of each $A \in X \cup Y$. Let Z be the set of attributes not specified by φ, i.e., $\mathcal{R} \setminus (X \cup Y)$. The definitions of support and confidence for the MD $\varphi(X \rightarrow Y, \lambda)$ are presented as follows:

$$\begin{aligned} \mathsf{support}(\varphi) &= P(X \vDash \lambda_X, Y \vDash \lambda_Y) \\ &= \sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z) \\ \mathsf{confidence}(\varphi) &= P(Y \vDash \lambda_Y \mid X \vDash \lambda_X) \\ &= \frac{\sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z)}{\sum_{Y,Z} P(X \vDash \lambda_X, Y, Z)} \end{aligned}$$

where $\vDash$ denotes the *satisfiability* relationship, i.e., $X \vDash \lambda_X$ denotes that the similarity values on all attributes in X satisfy the corresponding thresholds listed in λ_X. For example, we say that a statistical tuple s in $\mathcal{D}$ satisfies λ_X, i.e., $s[X] \vDash \lambda_X$, if s has similarity values higher than the corresponding minimum threshold, i.e., $s[A] \geq \lambda[A]$, for each attribute A in X.

[1] The MDs syntax is described with two relation schema R_1, R_2 for object identification in [4], which can also be represented in a single relation schema R as the FDs.

3. ALGORITHM

We now study the determination of matching similarity threshold pattern for MDs based on the statistical distribution, which is a new problem different from FDs. In fact, once the $X \rightarrow Y$ is given for a FD, it already implies the similarity threshold to be 1.0, that is, $(X \rightarrow Y, < 1.0, 1.0 >)$ if it is represented by the MD syntax. Unlike FDs, we have various settings of matching similarity thresholds for MDs. Therefore, in this section, we discuss how to find the right similarity thresholds in order to discover the MDs satisfying the required support and confidence.

Problem Statement. In order to discover a MD φ with the minimum requirements of support η_s and confidence η_c, the following preliminary should be given first: **(I)** what is Y? and **(II)** what is matching quality requirement λ_Y. These two preliminary questions are usually addressed by specific applications. For example, if we would like to use discovered MDs to guide objet identification in the Contacts table, then $Y = \mathsf{SIN}$. The λ_Y is often set to high similarity thresholds by applications to ensure high matching quality on Y attributes. For example, λ_Y is set to 1.0 for $Y = \mathsf{SIN}$ in the object identification application. Note that without the preliminary λ_Y, the discovered MDs will be meaningless. For example, a MD with $\lambda_Y = 0$ can always satisfy any requirement of η_c, η_s. Since all the statistical tuples can satisfy the thresholds $\lambda_Y = 0$, the corresponding support and confidence will always be 1.0.

DEFINITION 2. *The threshold determination problem of* MDs *is: given the embedded attributes X and Y, the minimum requirements of support and confidence η_s, η_c, and the matching similarity threshold pattern λ_Y, to find all the* MDs $\varphi(X \rightarrow Y, \lambda)$ *with threshold pattern λ_X on attributes X having* $\mathsf{confidence}(\varphi) \geq \eta_c$ *and* $\mathsf{support}(\varphi) \geq \eta_s$*, if exist; otherwise return* infeasible.

The attributes X can be initially assigned to $\mathcal{R} \setminus Y$ if no suggestion is provided by specific applications, since our discovery process can automatically remove those attributes that are not required in X for a MD φ. Specifically, when a possible discovered threshold $\lambda[A]$ on attribute A is $0 \in \mathsf{dom}(A)$, it means that any matching similarity value of the attribute $A \in X$ can satisfy the threshold 0 and will not affect the MD φ at all. In other words, the attribute A can be removed from X of the MD φ.

Exact Algorithm. Now, we present an algorithm to compute the similarity thresholds on attributes X for MDs having support and confidence greater than η_s and η_c, respectively. Let $A_1, \ldots, A_{m_X}$ be the m_X attributes in X. For simplicity, we use λ to denote the threshold pattern projection λ_X with $\lambda[A_1], \ldots, \lambda[A_{m_X}]$ on all the m_X attributes of X. Since, each threshold $\lambda[A_i]$ on attribute A_i is a value from $\mathsf{dom}(A_i)$, i.e., $\lambda[A_i] \in \mathsf{dom}(A_i)$, we can investigate all the possible candidates of threshold pattern λ. Let $\mathcal{C}_t$ be the set of all the possible threshold pattern candidates, having

$$\mathcal{C}_t = \mathsf{dom}(A_1) \times \cdots \times \mathsf{dom}(A_{m_X}) = \mathsf{dom}(X).$$

The total number of candidates is $c = |\mathcal{C}_t| = |\mathsf{dom}(X)| = d^m$, where d is the size of $\mathsf{dom}(A_i)$.

Let n be the number of statistical tuples in the input statistical distribution $\mathcal{D}$. We consider two statistical values $P_i^j(X, Y)$ and $P_i^j(X)$, which record $P(X \vDash \lambda_X, Y \vDash \lambda_Y)$ and $P(X \vDash \lambda_X)$ respectively for the candidate $\lambda_j \in \mathcal{C}_t$ based on the information of the first i tuples in $\mathcal{D}$, initially having $P_0^j(X, Y) = P_0^j(X) = 0$. The recursion is defined as follows, with i increasing from 1 to n and j increasing from 1 to c.

$$P_i^j(X,Y) = \begin{cases} P_{i-1}^j(X,Y) + s_i[P], & \text{if } s_i[X] \vDash \lambda_j, s_i[Y] \vDash \lambda_Y \\ P_{i-1}^j(X,Y), & \text{otherwise} \end{cases}$$

$$P_i^j(X) = \begin{cases} P_{i-1}^j(X) + s_i[P], & \text{if } s_i[X] \vDash \lambda_j \\ P_{i-1}^j(X), & \text{otherwise} \end{cases}$$

Finally, those λ_j can be returned if $\mathsf{support} = P_n^j \geq \eta_s$ and $\mathsf{confidence} = \frac{P_n^j(X,Y)}{P_n^j(X)} \geq \eta_c$.

Algorithm 1 Exact algorithm $\mathbf{EA}(\mathcal{D}, \mathcal{C}_t)$

```
for each candidate λ_j ∈ C_t, j : 1 → c do
    P_0^j(X, Y) = P_0^j(X) = 0
    for each statistical tuples s_i ∈ D, i : 1 → n do
        compute P_i^j(X, Y), P_i^j(X)
return λ_j with confidence and support satisfying η_c, η_s
```

We can implement the exact algorithm (namely EA) by considering all the statistical tuples s_i in $\mathcal{D}$ with i from 1 to n, whose time complexity is $\mathcal{O}(nc)$.

Pruning Strategies. Since the original exact algorithm needs to traverse all the n statistical tuples in $\mathcal{D}$ and c candidate threshold patterns in $\mathcal{C}_t$, which is very costly. In fact, with the given η_s and η_c, we can investigate the relationship between similarity thresholds and avoid checking all candidate threshold patterns in $\mathcal{C}_t$. Therefore, in the following, we present pruning techniques based on the given support requirement η_s.

We first study the relationships among different threshold patterns, based on which we then propose rules to filter out candidates that have supports lower than η_s.

DEFINITION 3. *Given two similarity threshold patterns λ_1 and λ_2, if $\lambda_1[A] \leq \lambda_2[A]$ holds for all the attributes, $\forall A \in X$, then λ_1* dominates *λ_2, denoted as $\lambda_1 \lessdot \lambda_2$.*

Based on the *dominate* definition, the following Lemma describes the relationships of supports between similarity threshold patterns.

LEMMA 1. *Given two* MDs*, $\varphi_1 = (X \rightarrow Y, \lambda_1)$ and $\varphi_2 = (X \rightarrow Y, \lambda_2)$ over the same relation instance of $\mathcal{R}$, if λ_1 dominates λ_2, $\lambda_1 \lessdot \lambda_2$, then we have* $\mathsf{support}(\varphi_1) \geq \mathsf{support}(\varphi_2)$.

PROOF. Let $\mathsf{cover}(\lambda_1)$ and $\mathsf{cover}(\lambda_2)$ denote the set of statistical tuples that satisfy the threshold λ_1 and λ_2 respectively, e.g., $\mathsf{cover}(\lambda_2) = \{s \mid s[X] \vDash \lambda_2, s \in \mathcal{D}\}$. According to the minimum similarity thresholds, for each attribute A, we have $\lambda_2[A] \leq s[A]$. In addition, since $\lambda_1 \lessdot \lambda_2$, for any tuple $s \in \mathsf{cover}(\lambda_2)$, we also have $\lambda_1[A] \leq \lambda_2[A] \leq s[A]$ on all the attributes A. In other words, the set of statistical tuples covered by λ_2 also satisfy the threshold of λ_1, i.e., $\mathsf{cover}(\lambda_2) \subseteq \mathsf{cover}(\lambda_1)$. Referring to the definition of support, we have $\mathsf{support}(\varphi_1) \geq \mathsf{support}(\varphi_2)$. □

According to Lemma 1, given a candidate similarity threshold pattern λ_j having support lower than the user specified requirement η_s, i.e., $P_n^j(X, Y) < \eta_s$, all the candidates that are dominated by λ_j should have support lower than η_s and can be safely pruned without computing their associated support and confidence.

We present the implementation of pruning by support (namely EPS) in Algorithm 2.

In order to maximize the pruning, we can heuristically select an ordering of candidates in $\mathcal{C}_t$ that for any $j_1 < j_2$ having $\lambda_{j_1} \lessdot \lambda_{j_2}$. That is, we always first process the candidates that dominate others.

Algorithm 2 Pruning by support **EPS**$(\mathcal{D}, \mathcal{C}_t)$

1: **for** each candidate $\lambda_j \in \mathcal{C}_t, j : 1 \rightarrow c$ **do**
2: $P_0^a j = P_0^b j = 0$
3: **for** each tuple $s_i \in \mathcal{D}, i : 1 \rightarrow n$ **do**
4: compute $P_i^a j, P_i^j(X)$
5: **if** $P_n^a j < \eta_s$ **then**
6: remove all the remaining candidates λ' dominated by λ_j from $\mathcal{C}_t$ {Pruning by support, $\lambda' \gg \lambda_j$}
7: **return** λ_j with confidence and support satisfying η_c, η_s

In fact, we can use a DAG (directed acyclic graph), $\mathcal{G}$, to represent candidate similarity patterns as vertices and dominant relationships among the similarity patterns as edges. Thus, the dominant order of candidate patterns can be obtained by a BFS traversal upon $\mathcal{G}$.

4. EXPERIMENTAL EVALUATION

Experiment Setting. In the experimental evaluation, we use two real data sets. The *Cora*[2] data set, prepared by McCallum et al. [8], consists of 12 attributes including author, volume, title, institution, venue, etc. The *CiteSeer*[3] data set is selected with attributes including title, author, address, affiliation, subject, description, etc. We use the *cosine* similarity to evaluate the matching quality of the tuples in the original data. By applying the $\mathsf{dom}(A)$ mapping in Section 2, we can obtain statistical distributions with at most $186,031$ statistical tuples in *Cora*, and $314,382$ statistical tuples in *CiteSeer*. Our experimental evaluation is then conducted in several off-line pre-processed statistical distributions with various data sizes, i.e., statistical tuples n from $10,000$ to $150,000$ respectively.

We mainly observes the efficiency of proposed algorithms. Since our main task is to discover MDs under the required η_s and η_c, we study the runtime performance in various distributions with different η_s and η_c settings. The discovery algorithms determine the matching similarity settings of attributes for MDs. Suppose that users want to discover MDs on the following $X \rightarrow Y$ of two data sets respectively: **i)** the dependencies on

$$Cora : \mathsf{author}, \mathsf{volume}, \mathsf{title} \rightarrow \mathsf{venue}$$

with the preliminary requirement of minimum similarity 0.6 on venue; **ii)** the dependencies on

$$CiteSeer : \mathsf{address}, \mathsf{affiliation}, \mathsf{description} \rightarrow \mathsf{subject}$$

with preliminary 0.1 on subject, respectively.

A returned result is either infeasible, or a MD with threshold pattern on the given $X \rightarrow Y$, for example, one of the result returned by real experiment on *Cora* is:

$$\varphi(\mathsf{author}, \mathsf{volume}, \mathsf{title} \rightarrow \mathsf{venue}, < 0.6, 0.0, 0.8, 0.6 >)$$

with $\mathsf{support}(\varphi) = 0.020$ and $\mathsf{confidence}(\varphi) = 0.562$ both greater than the specified requirements of η_s and η_c respectively.

Approach Evaluation. We evaluate the performance of pruning by support (EPS) compared with the original exact algorithm (EA). As shown in (a) and (b) in Figure 1 and 2, the EA, which verifies all the possible candidates, should have the same cost no matter how η_s and η_c set. Therefore, the time cost of EA in (a) is exactly the same as that in (b) in all two data sets.

Moreover, the EPS achieves significantly lower time cost in all the statistical distributions, which is only about $1/10$ of that of the EA. These results demonstrate that our EPS approach can prune most of candidates without costly computation. Note that the time costs of approaches increase linearly with data sizes, which shows the scalability of discovering MDs on large data.

[2]http://www.cs.umass.edu/~mccallum/code-data.html
[3]http://citeseer.ist.psu.edu/

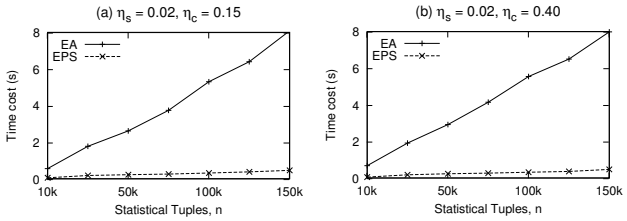

Figure 1: Pruning on *Cora*

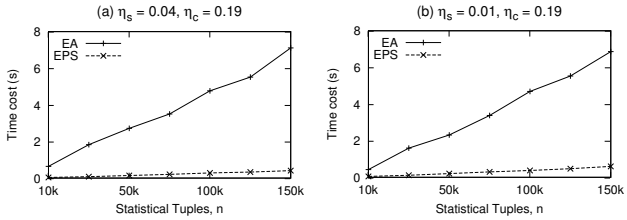

Figure 2: Pruning on *CiteSeer*

According to the pruning strategy, the EPS performance is only affected by support requirement η_s. In other words, different η_c settings take no effect on EPS. Thus, EPS has similar time costs in Figure 1 (a) and (b) with the same η_s but different η_c.

Recall that the EPS approach conducts the pruning based on the given requirement of support η_s. It is natural that a higher η_s turns to the better pruning performance. Therefore, EPS with $\eta_s = 0.04$ in Figure 2 (a) shows lower time cost, e.g., about 0.4s for 150k, than that of $\eta_s = 0.01$ in (b), e.g., 0.6s for the same 150k. Similar results with different η_s are also observed on *Cora*, which are not presented due to the limit of space.

5. CONCLUSIONS

In this paper, we study the discovery of matching dependencies. First, we formally define the utility evaluation of matching dependencies by using support and confidence. Then, we introduce the problem of discovering the MDs with minimum confidence and support requirements. Pruning strategies of the exact algorithm are studied. The pruning by support can filter out the candidate patterns with low supports. The experimental evaluation demonstrates the performance of proposed methods.

6. REFERENCES

[1] M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.
[2] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.
[3] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
[4] W. Fan. Dependencies revisited for improving data quality. In *PODS*, pages 159–170, 2008.
[5] W. Fan, J. Li, X. Jia, and S. Ma. Reasoning about record matching rules. *PVLDB*, 2009.
[6] L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *PVLDB*, 1(1):376–390, 2008.
[7] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric functional dependencies. In *ICDE*, pages 1275–1278, 2009.
[8] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, pages 169–178, 2000.