

# Scalable Entity Matching Computation with Materialization

Sanghoon Lee, Jongwuk Lee, Seung-won Hwang  
Department of Computer Science and Engineering  
Pohang University of Science and Technology (POSTECH)  
Pohang 790-784, Republic of Korea  
{sanghoon, julee, swhwang}@postech.edu

## ABSTRACT

Entity matching (EM) is the task of identifying records that refer to the same real-world entity from different data sources. While EM is widely used in data integration and data cleaning applications, the naive method for EM incurs quadratic cost with respect to the size of the datasets. To address this problem, this paper proposes a scalable EM algorithm that employs a pre-materialized structure. Specifically, once the structure is built, our proposed algorithm can identify the EM results with sub-linear cost. In addition, as the rules evolve, our algorithm can efficiently adapt to new rules by selectively accessing records using the materialized structure. Our evaluation results show that our proposed EM algorithm is significantly faster than the state-of-the-art method for extensive real-life datasets.

## Categories and Subject Descriptors

H.2.m [Database Management]: Miscellaneous.data cleaning; H.2.8 [Database Management]: Database Applications.data mining

## General Terms

Algorithms, experimentation, performance

## Keywords

Entity matching, resolution, materialization, rule evolution

## 1. INTRODUCTION

Entity matching (EM) is the process of identifying records that come from different data sources but represent the same real-world entity. EM is also known as entity resolution [1], reference reconciliation [3], record linkage [5], or object identification [10]. EM is widely used in various real-world applications such as data integration and data cleaning: (1) When entities are stored in a distributed manner across heterogeneous databases, we need to maintain those databases for data integration. (2) When many records in a single database may indicate the same entity, we need to eliminate redundant records for data consistency.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.  
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...\$10.00.

## 1.1 Basic Model of EM

We first introduce basic notations to define the EM problem. We assume two finite sets  $\mathcal{P}$  and  $\mathcal{Q}$  of records from different sources, *i.e.*,  $\mathcal{P} = \{p_1, \dots, p_m\}$  and  $\mathcal{Q} = \{q_1, \dots, q_n\}$ , as input. Each record has a finite set of fields (or attributes) with various data types such as numbers and strings. For simplicity, we assume that  $\mathcal{P}$  and  $\mathcal{Q}$  have compatible schemas: the records from  $\mathcal{P}$  and  $\mathcal{Q}$  have corresponding values for some fields.

We then formalize a *match function* (or a *matcher*) to compare records in a pair-wise manner. Specifically, we follow a *join model* over  $\mathcal{P} \times \mathcal{Q}$ . A join operation takes two sets  $\mathcal{P}$  and  $\mathcal{Q}$  as input, and returns a set of record pairs  $(p, q) \in \mathcal{P} \times \mathcal{Q}$ , referring to the same real-world entity, along with a match function  $\mathcal{M}$ . The match function can thus be represented by a Boolean condition for the same fields of  $\mathcal{P}$  and  $\mathcal{Q}$  if  $\mathcal{M}(p, q)$  is true,  $p$  and  $q$  indicate the same entity; otherwise, they are different. Without loss of generality, we define the match function  $\mathcal{M}$  as:

$$\mathcal{M}(p, q) \leftarrow \begin{cases} 1, & \text{if } \mathcal{F}(p, q) \leq \delta, \\ 0, & \text{otherwise,} \end{cases}$$

where  $\mathcal{F}(p, q)$  is a user-specific function calculating the similarity between records and  $\delta$  is a threshold parameter.

Considering the match function  $\mathcal{M}$  as a *user-specific rule*, we can define the *rule-based EM problem* [1, 7, 9, 11].

**DEFINITION 1 (ENTITY MATCHING)** Given two sets  $\mathcal{P}$  and  $\mathcal{Q}$  of records and a match function  $\mathcal{M}$ , *entity matching* is to identify a set  $\mathcal{S}$  of record pairs  $(p, q) \in \mathcal{P} \times \mathcal{Q}$  such that  $\mathcal{S} = \{(p, q) | p \in \mathcal{P}, q \in \mathcal{Q}, \mathcal{M}(p, q) = 1\}$ .

To support a sophisticated matching rule  $\mathcal{M}$  we exploit  $\mathcal{F}$  by using an *aggregate function* combining multiple fields, as commonly adopted in the EM literature [7]. For simplicity, we assume that  $\mathcal{F}$  is *monotone* [4].

$$\mathcal{F}(p, q) = \sum_{f \in \Omega} w_f \times f(p, q),$$

where  $\Omega$  is a space of all possible similarity functions, and  $f(p, q)$  is the similarity function between  $p$  and  $q$ . Also,  $w_f$  is a weight vector reflecting the importance of function  $f(p, q)$ .

For example, we describe the toy datasets related to restaurants (Table 1). For the first three fields *name*, *address* and *type*, we can exploit well-known string similarity functions such as *Levenshtein* distance and *Jaro-Winkler* distance. For the fourth field, *coordinate*, we can make use of numeric similarity functions such as *Manhattan* distance and *Euclidean* distance.

Table 1: Toy datasets from Fodor’s and Zagat’s restaurant resources

<i>id</i>	<i>name</i>	<i>address</i>	<i>type</i>	<i>coordinate</i>
$p_1$	Fringale	570 4th St. San Francisco	French	[37.778483, -122.396783]
$p_2$	Philippe’s The Original	1001 N. Alameda St. Los Angeles	American	[34.059619, -118.237073]
$p_3$	Ocean Avenue	1401 Ocean Ave. Santa Monica	American	[34.014045, -118.497473]

(a) Dataset  $\mathcal{P}$  from Fodor’s restaurant resource

<i>id</i>	<i>name</i>	<i>address</i>	<i>type</i>	<i>coordinate</i>
$q_1$	Ocean Park Cafe	3117 Ocean Park Blvd. Santa Monica	American	[34.021079, -118.452519]
$q_2$	Fringale	570 Fourth St. San Francisco	French Bistro	[37.778559, -122.39718]
$q_3$	Philippe The Original	1001 N. Alameda St. Chinatown	Cafeteria	[34.059673, -118.23702]

(b) Dataset  $\mathcal{Q}$  from Zagat’s restaurant dataset resource

## 1.2 Challenges of EM

Once the match function  $\mathcal{M}$  is determined, we identify a pair of records that refer to the same entity over  $\mathcal{P} \times \mathcal{Q}$ . The brute-force approach for EM first enumerates all possible record pairs  $(p, q) \in \mathcal{P} \times \mathcal{Q}$ , and then checks whether  $p$  and  $q$  match: if  $\mathcal{M}(p, q)$  is true,  $(p, q)$  is inserted into an EM result set; otherwise,  $(p, q)$  is skipped.

We describe how this naive method works using toy datasets. We generate nine possible record pairs over  $\mathcal{P} \times \mathcal{Q}$  (Table 2). Suppose that similarity functions  $f_{name}$  and  $f_{type}$  are defined as *Levenshtein* distance,  $f_{addr}$  is defined as *Jaro-Winkler* distance, and  $f_{coord}$  is defined as *Manhattan* distance. On the basis of some background knowledge, we set the weight vector  $w$  used in  $\mathcal{F}$  as  $\langle 0.1, 5, 0.02, 0.1 \rangle$  and the threshold  $\delta$  as 1. The aggregate function values for each pair are given in the last column in Table 2. Finally, we can identify the EM result as  $\{(p_1, q_2), (p_2, q_3)\}$ , *i.e.*, “Fringale” and “Philippe The Original” and are correctly resolved.

Table 2: All possible record pairs and their aggregated scores with weight vector  $\langle 0.1, 5, 0.02, 0.1 \rangle$

record pair	distance score				aggregated score
	$f_{name}$	$f_{addr}$	$f_{type}$	$f_{coord}$	
$(p_1, q_1)$	12	0.431	6	7.702	4.244
$(p_1, q_2)$	0	0.091	7	0.000	0.594
$(p_1, q_3)$	18	0.414	8	7.879	4.819
$(p_2, q_1)$	20	0.449	0	0.254	4.272
$(p_2, q_2)$	20	0.462	11	7.879	5.317
$(p_2, q_3)$	2	0.115	7	0.306	0.948
$(p_3, q_1)$	8	0.250	0	0.052	2.054
$(p_3, q_2)$	10	0.421	11	7.664	4.090
$(p_3, q_3)$	18	0.295	7	0.306	3.444

To provide exact EM results, it is critical to determine meaningful similarity functions  $f \in \Omega$  and tune the user-specific parameters  $w$  and  $\delta$ . To address these needs, rules can continuously evolve either by adding/removing functions from  $\Omega$  or adjusting  $w$  and  $\delta$ . Efficient support for evolving rules has also been emphasized by Whang and Garcia-Molina [11]. However, their proposed solution makes restrictive assumptions, *e.g.*, rule monotonicity, and suffers in performance when rules evolve significantly. Even when their assumptions hold (as we empirically study later), our proposed algorithm significantly outperforms the other solution, which still incurs a prohibitive cost for intermediate EM results.

## 1.3 Our Contributions

We propose a scalable EM algorithm using a materialization structure. Specifically, we observe that the process of finding EM results satisfying a threshold condition is similar to that of *top-k queries* [2, 4, 6], which identify the best  $k$  records with the highest scores. We thus adopt the *materialized lists* used for top- $k$  processing without accessing all

records. The materialized lists for EM store the similarity values of all record pairs for single-field functions as a basic access unit for EM. By accessing records in materialization lists at a time, we can identify a final EM result with sub-linear cost. Furthermore, we can interactively evolve various rules and efficiently obtain EM results with high accuracy.

## 2. ALGORITHM

This section proposes a scalable EM algorithm that identifies EM results using a pre-materialized structure. Towards this goal, we first introduce a *threshold algorithm* (TA) [4] that identifies the top- $k$  results by performing a fuzzy join with sub-linear cost. With similar intuition, we then adopt the materialization used in TA-family algorithms, and develop an EM algorithm that efficiently evolves for new rules.

### 2.1 Threshold Algorithm (TA)

We first provide the overview of top- $k$  query computation. A top- $k$  query aims to identify the best  $k$  records with respect to a user-specific *scoring function*  $\mathcal{F}$ . Suppose that the function  $\mathcal{F}$  is represented by a linear combination of  $d$  attributes, *i.e.*,  $\mathcal{F}(f_1(A_1), \dots, f_d(A_d))$ , where  $f_i(A_i)$  is the scoring function of attribute  $A_i$ . The naive approach to finding the top- $k$  results requires to access all records in the worst case.

To address this problem, TA exploits a pre-computed data structure. Specifically, they first maintain  $d$  sorted lists in increasing order of  $f_i(A_i)$ , where smaller scores are better. On these sorted lists, they then access records by performing *sorted access* or *random access* until the top- $k$  records with the highest scores are identified.

TA keeps the current top- $k$  records found thus far while accessing the sorted lists. When the aggregated score of the  $k^{th}$  ranked record is larger than the aggregated score of the last seen attribute values in the sorted lists, TA stops attempting to access further records and returns the current top- $k$  records as an answer. Because this early-termination condition avoids searching the whole dataset, it is enough to consider few records distributed at the heads of sorted lists.

This key idea of TA can be applied to the EM problem in the sense that both problems only deal with some of the records satisfying a threshold condition. Therefore, we can get the EM results quickly by probing a small number of record pairs.

### 2.2 Threshold-Based EM Algorithm

We first devise a materialization structure inspired by TA. Before starting an EM algorithm, we construct lists for each similarity function  $f_i(p, q)$ . These lists are sorted in increasing order of function values, *i.e.*, the most similar record pairs are accessed first. We call them *materialized lists*, denoted as  $M_i$  for  $f_i(p, q)$ .

---

**Algorithm 1**  $TEM(\mathcal{M}, w, \delta, \mathcal{H})$ 

---

**Input:** set of materialized lists  $\mathcal{M} = \langle M_1, \dots, M_d \rangle$ , weight vector  $w = \langle w_1, \dots, w_d \rangle$ , threshold  $\delta$  and EM history  $\mathcal{H}$   
**Output:** set of record pairs representing the same entity,  $\mathcal{S}$

```
1:  $\langle t_1, \dots, t_d \rangle \leftarrow 0$  // a threshold vector for seen records
2:  $\mathcal{S} \leftarrow \{\}$ 
3: if  $\mathcal{H}.w \prec w$  then
4:    $\mathcal{S} \leftarrow \mathcal{H}.\mathcal{S}$  // start from the result of history
5:   for  $\forall M_i \in \mathcal{M}$  do
6:      $M_i.\text{access-point} \leftarrow \mathcal{H}.M_i.\text{access-point}$ 
7:   end for
8: end if
9: while  $\sum_{i=1}^d w_i \cdot t_i \leq \delta$  do
10:  for  $\forall M_i \in \mathcal{M}$  do
11:     $\langle (p, q), r_i \rangle \leftarrow \text{sorted-access}(M_i)$  // get record pair with
    the next largest similarity for  $f_i$ , and its score
12:     $t_i \leftarrow r_i$ 
13:    for  $\forall M_j \in \mathcal{M}, j \neq i$  do
14:       $r_j \leftarrow \text{random-access}(M_j, (p, q))$  // get similarity
      score of  $(p, q)$  for  $f_j$ 
15:    end for
16:    if  $\sum_{i=1}^d w_i \cdot r_i \leq \delta$  then
17:       $\mathcal{S} \leftarrow \mathcal{S} \cup (p, q)$  //  $(p, q)$  is a matched record pair
18:    end if
19:  end for
20: end while
21: return  $\mathcal{S}$ 
```

---

For example, we describe the materialized lists using the toy datasets in Table 2. For ease of representation, suppose that three similarity functions on the fields of *name*, *address*, and *coordinate* are used for EM. We calculate all possible pair-wise similarities and sort them (Table 3).

**Table 3: Materialized lists built from 3 similarity function values of pair-wise records in Table 2**

$M_{name}$		$M_{addr}$		$M_{coord}$	
pair	$f_{name}$	pair	$f_{addr}$	pair	$f_{coord}$
$(p_1, q_2)$	0	$(p_1, q_2)$	0.091	$(p_1, q_2)$	0.000
$(p_2, q_3)$	2	$(p_2, q_3)$	0.115	$(p_3, q_1)$	0.052
$(p_3, q_1)$	8	$(p_3, q_1)$	0.250	$(p_2, q_1)$	0.254
$(p_3, q_2)$	10	$(p_3, q_3)$	0.295	$(p_2, q_3)$	0.306
$(p_1, q_1)$	12	$(p_1, q_3)$	0.414	$(p_3, q_3)$	0.306
$(p_1, q_3)$	18	$(p_3, q_2)$	0.421	$(p_3, q_2)$	7.664
$(p_3, q_3)$	18	$(p_1, q_1)$	0.431	$(p_1, q_1)$	7.702
$(p_2, q_1)$	20	$(p_2, q_1)$	0.449	$(p_1, q_3)$	7.879
$(p_2, q_2)$	20	$(p_2, q_2)$	0.462	$(p_2, q_2)$	7.879

We then develop a *threshold-based EM algorithm (TEM)* using the materialized lists. In a method similar to TA, we use two access modes: sorted access and random access. Algorithm 1 describes the pseudo code of *TEM*. Let  $d$  denote the number of fields used in EM. Specifically, the overall procedure of *TEM* is as follows:

1. We do sorted access on each materialized list. If a new record pair  $(p, q)$  appears in  $M_i$ , we do random access on the other materialized lists to identify unseen similarity scores. Then, we compute the overall similarity function score  $\mathcal{F}(p, q)$  and check whether  $\mathcal{F}(p, q)$  is less than or equal to  $\delta$ . If  $p$  and  $q$  are matched, we can insert  $(p, q)$  into an EM result set.
2. For each list  $M_i$ , let  $t_i$  be the last similarity score seen by sorted access. We compute a threshold  $\sum_{i=1}^d w_i \cdot t_i$  such that  $t = \langle t_1, \dots, t_d \rangle$ . If  $\sum_{i=1}^d w_i \cdot t_i$  is smaller than  $\delta$ , return to step 1.
3. Return all matched record pairs as EM results.

In addition to the above basic procedure, we can also exploit the EM history  $\mathcal{H}$  and evolve from an old rule. If a weight vector of  $\mathcal{H}$  is dominated by  $w$  (line 3), the result in the history will also be a part of the result for  $w$ . Therefore, we reuse results for the old rule (line 4) and save access costs by restarting from the last access points of  $\mathcal{H}$  (lines 5-7).

We prove the correctness of *TEM* as follows.

**THEOREM 1 (CORRECTNESS OF TEM)** Given a match function  $\mathcal{M}$ , *TEM* correctly returns all record pairs such that  $\mathcal{M}(p, q) = 1$ .

**PROOF.** Suppose there is a matched record pair  $(p', q')$ , but *TEM* does not return it as an answer. That is,  $(p', q')$  is not applied to a match function, since it is not visible by sorted access on the materialized lists. This means that any similarity function value of  $(p', q')$  is not smaller than the corresponding value of the last seen similarity score  $t_i$ . By the definitions of the aggregate and match functions,  $\mathcal{F}(p', q') \geq \sum_{i=1}^d w_i \cdot t_i > \delta$ . This contradicts the supposition that  $(p', q')$  is a matched record pair. Therefore, there does not exist a false negative such as  $(p', q')$ , and *TEM* correctly returns all matched record pairs.  $\square$

We explain *TEM* using materialized lists (Table 3). Suppose an aggregation function  $\mathcal{F}$  consists of  $w = \langle 0.1, 5, 0.1 \rangle$ , and  $\delta$  is 1. The threshold vector  $t$  is initially a zero vector. Then, we access the materialized lists from  $M_{name}$  to  $M_{coord}$ . The first sorted access steps for the three materialized lists retrieve  $(p_1, q_2)$ , whose aggregate score  $\mathcal{F}$  is smaller than the threshold  $\delta = 1$ , and we insert  $(p_1, q_2)$  into an EM result set. Then  $t$  is updated to  $\langle 0, 0.091, 0 \rangle$ . The second sorted access and random access steps calculate  $\mathcal{F}(p_2, q_3)$  and  $\mathcal{F}(p_3, q_1)$ . The  $(p_2, q_3)$  pair is matched and inserted into the result set. Now,  $t$  is updated to  $\langle 2, 0.11, 0.052 \rangle$  and  $\mathcal{F}(t) < \delta$ , so we continue. The third access steps get  $(p_3, q_1)$  and  $(p_2, q_1)$ . The pair  $(p_3, q_1)$  is was already addressed at the second step, and  $\mathcal{F}(p_2, q_1) > \delta$ ; thus, we dismiss them. At this moment,  $t$  is  $\langle 8, 0.250, 0.254 \rangle$  and  $\mathcal{F}(t) = 2.075 > \delta$ . Finally, we stop accessing the materialized lists and return the result set  $\{(p_1, q_2), (p_2, q_3)\}$ .

### 3. EVALUATION

This section shows experimental results for our proposed EM algorithm using extensive real-life datasets. Specifically, we exploited four datasets used in existing EM experimental study [8] (Table 4). Two dataset pairs, Abt-Buy and Amazon-GoogleProducts are used in e-commerce applications, where they consist of four fields, *product name*, *description*, *manufacturer*, and *price*. The other pairs DBLP-ACM and DBLP-Scholar are used in the bibliographic domain, where they consist of four fields, *title*, *authors*, *venue*, and *year*. For simplicity, we used *Levenshtein* distance as the similarity function for each field. (By carefully designing similarity functions, we can further improve the accuracy of EM results.)

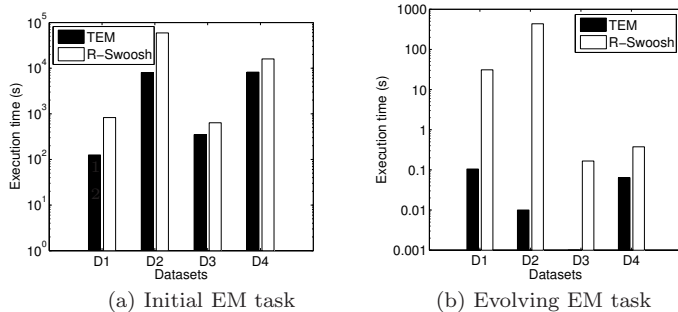
Our experiments were conducted using a 64-bit Linux server with an Intel Core i7 Processor (2.8 GHz) and 24 GB of RAM. All the algorithms were implemented in Java.

Figure 1 depicts the execution time for two EM tasks. Note that these figures have a log-scaled  $y$ -axis. When constructing the initial EM task (Figure 1a), the *TEM* execution time is the sum of those for both building materialization lists and computing EM results. Nevertheless, our

**Table 4: Four real-life datasets used in evaluation of TEM**

real-life dataset pairs	dataset size		search space	perfect mapping size
	$ \mathcal{P} $	$ \mathcal{Q} $	$ \mathcal{P} \times \mathcal{Q} $	
$\mathcal{D}_1$ (Abt-Buy)	1,081	1,092	1,180,452	1,097
$\mathcal{D}_2$ (Amazon-GoogleProducts)	1,363	3,226	4,397,038	1,300
$\mathcal{D}_3$ (DBLP-ACM)	2,616	2,294	6,001,104	2,224
$\mathcal{D}_4$ (DBLP-Scholar)	2,616	64,263	168,112,008	5,347

algorithm is two to seven times faster than the state-of-the-art EM algorithm *R-Swoosh* [1]. Although *R-Swoosh* focuses on reducing the search space for all possible pairs without additional indices, as a hierarchical tree structure it inherently requires a large number of comparisons to obtain intermediate EM results. These comparisons can deteriorate the overall performance. In contrast, after building sorted lists, our algorithm selectively accesses records to minimize unnecessary computation.


**Figure 1: Execution times to perform an EM task from scratch<sup>1</sup> and to evolve for a new rule<sup>2</sup>**

We also report on the execution time for evolving rules after the materialized lists are built up. We compared our algorithm with the rule evolution algorithm *HCDS* [11] based on *R-Swoosh*. We generated evolving rules satisfying some constraints, *i.e.*, *rule monotonicity* and *rule-context free*, mentioned the previous study [11]. Note that these rule generation constraints are unfavorable for our work, which does not have these restrictions. In addition, the *TEM* used in this experiment does not exploit the history. However, for all datasets, our algorithm is up to two orders of magnitude faster than the existing algorithm. This implies that our algorithm selectively accesses small portions of the materialization lists over an extensive set of rules.

**Table 5: Ratio of sorted accesses in materialized lists**

datasets	#record pairs (a)	#access pairs (b)	access ratio (b/a)
$\mathcal{D}_1$	1,180,452	1,995	0.0017
$\mathcal{D}_2$	4,397,038	22,079	0.0050
$\mathcal{D}_3$	6,001,104	3,540	0.0006
$\mathcal{D}_4$	168,112,008	11,655	0.0001

Specifically, Table 5 reports on the ratio of accessed tuples in the materialized lists. This ratio determines the execution time to resolve entities for our proposed method. Empirically, sorted access of less than 0.5% of the materialized lists is sufficient to obtain EM results.

<sup>1</sup> Weight vectors for  $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ , and  $\mathcal{D}_4$  are  $\langle 1.87, 0.06, 0.04 \rangle$ ,  $\langle 1.10, 0.32, 0.17, 0.18 \rangle$ ,  $\langle 1.34, 0.61, 0.32, 2.06 \rangle$ , and  $\langle 1.55, 0.94, 0.14, 0.08 \rangle$ , respectively.

<sup>2</sup> Weight vectors for  $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ , and  $\mathcal{D}_4$  are  $\langle 1.98, 0.07, 0.04 \rangle$ ,  $\langle 1.16, 0.34, 0.18, 0.19 \rangle$ ,  $\langle 1.41, 0.64, 0.34, 2.17 \rangle$ , and  $\langle 1.63, 0.99, 0.15, 0.08 \rangle$ , respectively. This is the best rule among rules with  $<5\%$  weight difference. Although our work is not constrained by this condition, we add it to accommodate *R-Swoosh* building on this constraint.

Finally, we report on the accuracy of our algorithm using  $F_1$  score (Table 6). We evolved rules by changing a weight vector  $w$  in a simple hill-climbing manner, where the functions and threshold  $\delta$  are fixed. In clear contrast to previous methods, our algorithm achieved comparable accuracy from  $\mathcal{D}_3$  and  $\mathcal{D}_4$  in the benchmark set in addition to adapting to rule changes efficiently. These algorithms require complete re-training to try out new matching techniques, which can also be repeated until convergence.

**Table 6: Accuracy comparisons ( $F_1$  score) for ours and ML-based EM algorithms**

datasets	TEM	PPJoin+	FEbRL	MARLIN
$\mathcal{D}_3$	0.965	0.919	0.976	0.974
$\mathcal{D}_4$	0.854	0.778	0.876	0.894

## 4. CONCLUSION

This paper has addressed the scalability of EM. Towards this goal, we exploited a materialization structure inspired by top- $k$  query processing. Using the materialization, we developed a scalable EM algorithm for evolving rules. Our evaluation demonstrated that our proposed algorithm is two to seven times faster than existing ER algorithms. In addition, when rules are evolved, our algorithm is two orders of magnitude faster than existing algorithms.

## Acknowledgement

This work was supported by Microsoft Research Asia and the Ministry of Knowledge Economy (MKE), Korea under Information Technology Research Center (ITRC) support program supervised by the National IT Industry Promotion Agency (NIPA-2011-C1090-1131-0009).

## 5. REFERENCES

- [1] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB Journal*, 18:255–276, 2009.
- [2] K. C. Chang and S. Hwang. Minimal probing: supporting expensive predicates for top-k queries. In *SIGMOD*, pages 346–357, 2002.
- [3] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96, 2005.
- [4] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, pages 102–113, 2001.
- [5] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
- [6] S. Hwang and K. C. Chang. Optimizing top-k queries for middleware access: A unified cost-based approach. *ACM Trans. Database Syst.*, 32(1), 2007.
- [7] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data Knowl. Eng.*, 69(2):197–210, 2010.
- [8] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.
- [9] D. Menestrina, S. E. Whang, and H. Garcia-Molina. Evaluating entity resolution results. *PVLDB*, 3:208–219, 2010.
- [10] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Inform. Syst.*, 26:607–633, 2001.
- [11] S. E. Whang and H. Garcia-Molina. Entity resolution with evolving rules. *PVLDB*, 3:1326–1337, 2010.