2008

Record Linkage Based on Entities' Behavior

Mohamed Yakout

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Hazen Elmeleegy

Mourad Ouzzani

Report Number:
08-026

Record Linkage Based on Entities' Behavior

Mohamed Yakout
Ahmed Elmagarmid
Hazem Elmeleegy
Mourad Ouzzani

CSD TR #08-026
October 2008

Record Linkage Based on Entities' Behavior

Mohamed Yakout, Ahmed Elmagarmid, Hazem Elmeleegy, Mourad Ouzzani

Department of Computer Sciences, Purdue University
West Lafayette, IN 47907, USA
{myakout,ake,hazem,mourad}@cs.purdue.edu

Abstract— **Record linkage is the problem of identifying similar records across different data sources. Traditional record linkage techniques focus on using simple database attributes in a textual similarity comparison to decide on matched and non-matched records. Recently, record linkage techniques have considered useful extracted knowledge and domain information to help enhancing the matching accuracy. In this paper, we present a new technique for record linkage that is based on entity's behavior, which can be extracted from a transaction log. In the matching process, we measure the improvement of identifying a behavior when comparing two entities by merging their transaction log. To do so, we use two matching phases; first, a candidate generation phase, which is fast and provide almost no false negatives, while producing low precision. Second, an accurate matching phase, which enhances the precision of the matching at high run time cost. In the candidates phase generation, behavior is represented by points in the complex plan, where we perform approximate evaluations. In the accurate matching phase, we use a heuristic called *compressibility*, where identified behaviors are more compressible. Our experiments show that the proposed technique can be used to enhance the record linkage quality while being practical for large logs. We also perform extensive sensitivity analysis for the technique's accuracy and performance.**

I. INTRODUCTION

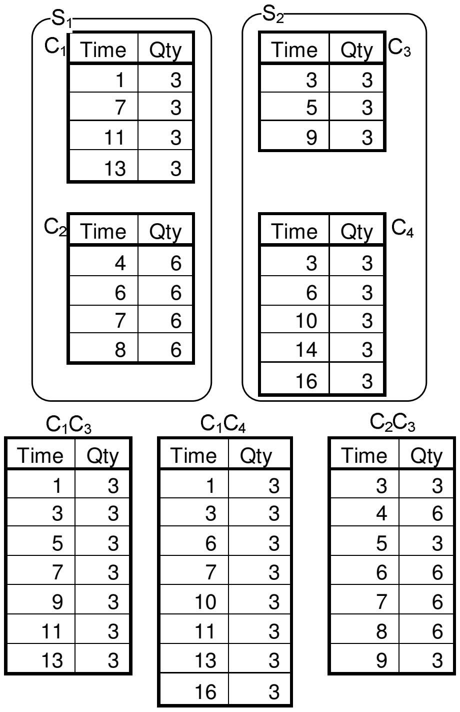
Record linkage is the process of identifying similar records that represent the same real world entity. Linking records across different sources has many applications like improving the quality of the data by comparing to a more accurate source or for data analysis and mining. Record linkage is important when integrating two data sources into one. For example if two companies are merging, it is important for the merge to succeed that shared customers are discovered. Record linkage is also referred to duplicate detection when identifying similar records is performed within the same source.

Prior work in record linkage focused on simple attributes similarities (refer to [10] for a recent complete survey). In the linking process, more than one technique is employed to enhance the matching accuracy. The techniques are also domain specific and depend on the availability of some data features. Recently, record linkage techniques have evolved to consider more information extracted from the existing raw data for enhancing the process of matching.

In this paper, we observe that in some applications the entities to be linked have a behavior that is recorded in some transaction log. Such behavior can be then used to determine whether two entities are in fact the same. For example, in a super market, customers buying transactions are stored attached to some customer id. Most of the time, stores does not store customer's personal information while depending on their credit cards to identify them in their databases. If two stores are considering merging, they can not use the credit card information for linking the records due to privacy. Therefore, the customer's buying behavior which is stored in the transaction log is the only information that can help in identifying common customers. In addition, in some surveillance systems, it may be possible to register entities' actions in the premises being monitored. Linking the records across such systems may help in crime investigations. In both situations the entities' behavior can play an important role in the linkage process since persons tend to follow similar or correlated behavior in different places.

The behavior of an entity is usually represented in a transaction log as a set of actions performed at a given time with specific features, e.g. in the supermarket scenario, the actions are the items and the features are the quantities that the customer bought from each item.

In this paper, we present a new technique for record linkage based on the entities' behavior, which is stored in a transaction log. The transaction log registers each action performed and eventually the action's feature describing how the action was performed. We should note however that when comparing two entities from two sources or within the same source for duplicate detection, we are not usually expecting to have exactly the same behavior or transactions in the two sources for the same entity. Instead, our objective is to analyze the "merged behavior" and determine how likely the merged behavior corresponds to the same entity. Our approach in comparing behaviors can be better described using the example in Fig. 1. We assume that there are two stores S_1 and S_2, where S_1 has customers C_1 and C_2, and store S_2 has customers C_3 and C_4. Beside each customer, the transactions of buying milk are shown. When linking the customers from S_1 to the ones in S_2, the similarity between the transactions cannot be a correct measure of similarity; it is not expected for the customer (entity) to buy the same items (perform the same actions) in two different stores (in two different systems) at the same time or with the same pattern. Let us now instead look at the merged transactions from the two stores. The merged transaction log of C_1 and C_3, appears under C_1C_3. We note that a pattern or behavior in buying the milk can be recognized; C_1C_3 is a customer buying 3 gallons of milk every two days. Therefore, most probably C_1 and C_3 represent the same customer. Note also that each of C_1 and C_3 alone does not demonstrate a recognized behavior in buying milk. When merging C_1 and C_4, we cannot identify a behavior and consequently, C_1 and C_4 cannot be the same. In the case of C_2C_3, although there is a recognized pattern, where the customer buys milk every day, the numbers of gallon are different and deviate; thus C_2 and C_3 are not the same customer as well.



S1

C1

Time	Qty
1	3
7	3
11	3
13	3

C2

Time	Qty
4	6
6	6
7	6
8	6

S2

C3

Time	Qty
3	3
5	3
9	3

C4

Time	Qty
3	3
6	3
10	3
14	3
16	3

C1C3

Time	Qty
1	3
3	3
5	3
7	3
9	3
11	3
13	3

C1C4

Time	Qty
1	3
3	3
6	3
7	3
10	3
11	3
13	3
16	3

C2C3

Time	Qty
3	3
4	6
5	3
6	6
7	6
8	6
9	3

Fig. 1 illustrating example

Based on the described example, the process of comparing two entities is performed by merging their transactions with the goal of obtaining a more *identifiable behavior* if these two entities represent in fact the same entity. Therefore, our approach focuses on measuring the improvement in identifying behavior after merging entities' transactions. Since the transaction log is expected to be large, we avoid visiting it several times and instead proceed through a two-phased approach for matching the entities.

In the first phase, we use a relaxed matching approach to quickly produce candidate pairs for matching. The matching in the first phase is meant to be fast and guarantee high recall (i.e. almost no false negatives) with minimum false negatives. The improvement in identifying the behavior is evaluated by representing every action as point in the complex plan. The representation is based on the time at which the action was performed. It can be explained as if the total period covering the transactions is distributed over a circle centered at the origin and every time the action is performed, the complex number point is pulled in the corresponding direction. Patterns with stable features (less deviated) become identifiable as the magnitude of the complex number is close to the origin.

In the second phase, a more accurate but expensive matching function is used to improve the precision of the initial matching results. The underlying idea is that when performing actions, repeated patterns and stable features will be more "*compressible*" if a behavior can be well recognized. The compressibility heuristic stems from the idea of representing behaviors as images and compressing the images. Therefore, when merging two entities' transactions, we propose to measure a compressibility gain for each of the entities to help each entity select its best match using a stable marriage technique.

To the best of our knowledge this is the first work that considers the use of entities' behavior for the record linkage purpose.

The rest of the paper is organized as follows; we begin by discussing the behavior characteristics and formulate the problem studied in section II. Section III presents the candidates matching phase and section IV describes the accurate matching phase. In section V, we discuss the filtration of the matching results. The conducted experiments are discussed in VI. Section VII contains the related work and finally we conclude the paper in section VIII.

II. Preliminaries

In this section, we first introduce a characterization of the behavior along two dimensions. We then outline the record linkage process through composite matchers and formulate the problem in the context of behavior identification improvement.

A. Behavior Dimensions

Informally, the behavior of a given entity, e.g., a person, a gene, or a particle, can be characterized through the actions this entity performs (using an action log for example) along two main dimensions: action repetition patterns and action features.

Action repetition patterns can be recognized when an entity repeats specific actions on a regular basis following a pattern or trend. Such patterns could differ from an entity to another or could be similar. For example, a customer (entity) who buys cat food (action) every week (pattern). These patterns can also be fixed, increasing, decreasing, oscillating or seasonal.

Action features are some attributes that are attached to every type of actions and describe how the action was performed. Entities could vary in their performance of the same action in terms of these features. Sometimes, there exists a preference to perform an action according to specific feature values. The features preference can be recognized in behaviors when stabilization is followed. For example, when a customer buys milk (action), he or she buys 3 gallons (feature of the action of buying milk) while other customers prefer buying 1 gallon.

Other behavior characterization can be also considered like action relationships, which can take different forms including such as association and implication. However, in this paper, we focus on the action repetition patterns and features.

From the above discussion, we propose a definition for a conceptual representation of behavior called *Behavior Matrix*.

Definition: Behavior Matrix

Given a finite set of n actions performed over a discrete finite period of time of length m by an entity E, the *Behavior Matrix (BM)* of E is an $n \times m$ matrix, such that :

$$BM_{i,j} = \begin{cases} C_{ij} & \text{if action } j \text{ is performed} \\ 0 & \text{otherwise} \end{cases}$$

Where, C_{ij} represents the feature value when performing action j at time i, $i = 0, 1, 2, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$,

The Behavior Matrix is an interesting conceptual approach for representing entity's behavior. Using this matrix, we can visualize the behavior as an image where the actions' features are considered as colors. This would allow to visually look at the two behavior's dimensions. The action's pattern and features, interpreted as repeated blocks in the images, led us to the *compressibility* heuristic approach described in section IV.

B. Problem Definition

Record linkage is a process of identifying pairs of records across two or more databases that correspond to the same real world entity. The behavior is considered as a complex attribute of an entity and can be used in improving the results of the record linkage problem. Basically, the process is composed of building *Matching Functions* that take as input a set of thresholds and a pair of records to classify them as *match* or *mismatch* according to a predefined decision rule.

Definition: Matching Function

Given two relations with the same attributes R_A $(a_1, a_2 \ldots, a_k)$ and R_B $(a_1, a_2 \ldots, a_k)$. A matching function MF takes as input triple $(r_A, r_B, \{\theta_1, \ldots \theta_k\})$ and produces a Boolean output $\{True, False\}$ corresponding to $\{match, mismatch\}$, where:

- $r_A \in R_A$ is a record with attribute values $(r_A(a_1), \ldots r_A(a_k))$ and $r_A(a_1) \in Dom(R_A.a_1) \ldots r_A(a_k) \in Dom(R_A.a_k)$.
- $r_B \in R_B$ is a record with attribute values $(r_B(a_1), \ldots r_B(a_k))$ and $r_B(a_1) \in Dom(R_B.a_1) \ldots r_B(a_k) \in Dom(R_B.a_k)$.
- $\{\theta_1, \ldots \theta_k\}$ are predefined similarity thresholds for the corresponding attributes $a_1, \ldots a_k$ in both the relations R_A and R_B.

The output of the matching function MF is decided based on

$$MF(r_A, r_B, \{\theta_1, \ldots \theta_k\}) = \begin{cases} True & iff \quad \bigcap_{i=1}^{k} f_i(r_A(a_i), r_B(a_i)) \le \theta_i \\ False & \quad otherwise \end{cases}$$

Where $f_i : Dom(R_A.a_i) \times Dom(R_B.a_i) \to \Re^+$, $i = 1, \ldots k$, are predefined similarity measures or distance functions defined over the domains of corresponding attribute a_i for the relations R_A and R_B.

The decision rule used in the above definition to identify a pair as matches or not is considered strict. Fellegi and Sunter [23] presented a more flexible formulation for the rules that depends on the output of the similarity functions f_i

Traditionally, the input to the matching function was simple database attributes and the function would compute the scores based on hamming or edits distances. Recently, many approaches targeted the use of other extracted information and sometimes adopt iterative approaches over the data aiming at returning more accurate similarity scores. For example, the use of the relationship with other referenced entities in a database [4] and same entity-to-entity relationship [3]. Practically, many matching functions are used with different types of input in a record linkage process, depending on the nature of the dataset.

Sometimes, the matching function performs a *relaxed matching* [12], where it is expected to obtain accurate decision about mismatches, while decision about matches may not be accurate. Relaxed matching is used when the accurate matching function is computationally expensive and therefore, a relaxed cheap matching function is employed first to produce an input to a more expensive but accurate matching function. This usually leads to a two-phased approach. The first phase uses fast relaxed matching functions with the goal of eliminating the number of false negatives to maintain high recall. The second phase is more expensive and takes care of improving the precision. The blocking technique is an example of a two-level matching process that has been used in the record linkage problem [15].

We consider an entity's behavior as a complex attribute with two components representing the two behavior's dimensions described above (action repetition pattern and features). The process of determining similarity between two entities based on their behavior is composed of two steps: first, merge their transactions, representing the actions they perform and then determine to what extent the resulting merged behavior becomes *identifiable* compared to the original behaviors. To measure the identification of behavior, we need to measure the *support improvement* for the two behavior dimensions. Since, attacking this problem through accurate measurements is not practical; we introduce a combination of simple heuristic techniques instead. The proposed heuristic approach resulted in an acceptable accuracy as explained in section VI.

We propose a technique that heuristically measures the enhancement in the two behavior's dimensions to help better identifying the overall entity's behavior. The technique is composed of two phases; candidate matches generation and accurate matching. In the candidates generation phase, the behavior is represented by points on the complex plan, where we apply distance measurements, while in the second accurate phase, we use a concept of compressibility to identify homogeneous behaviors.

III. Candidate Matches Generation

In the candidate matches generation phase, we use the dimensions of the behavior mentioned in section II and generate candidates pairs of entities for matching. We represent the actions log in a compact way to allow for fast computations in generating candidate matches. However, this quick computation comes at the cost of poor precision while eliminating false negatives.

The main goal of this phase is to minimize the number of candidate matches, while eliminating the false negatives. This phase should satisfy two important conditions; (1) Only one transaction log scan, and (2) use a small number of simple computations.

The matching function represented in this section classifies the records as *mismatch* and *likely-match*. The accurate matching operation is left for the Phase 2 matching.

Let E_x and E_y be two entities to be compared according to their behavior represented in their transaction log. In the following, we explain the use of the action pattern improvement and feature stabilization when merging two entities' transactions to determine an eventual similarity between them.

To compare two entities, each row in their Behavior Matrix, which corresponds to an action, is first converted to a complex number. Then, by merging the two entities' transactions, we expect the resulting magnitude of the complex number to become smaller or close to the original entities' magnitude if they are similar. This is because of filling gaps in the sequence and supporting the pattern. This observation will be illustrated shortly.

Each row in the Behavior Matrix can be converted to a complex number as follows: Suppose that a row vector i in the Behavior Matrix of E contains the sequence $x_{0,i}$, $x_{1,i}$, …, $x_{m-1,i}$ for action a_i, the complex number representation of this row can be computed as follow:

$$c(E.a_i) = \sum_{k=0}^{m-1} x_{k,i} e^{2k\pi\sqrt{-1}/m}$$

Consequently, the entity's behavior is represented by a vector of length n that contains complex numbers, where n is the number of actions performed by entity E. When comparing two entities' behaviors, the merged transaction's representation is obtained by adding two complex numbers without *re-visiting* the transaction log again. For example, suppose we are comparing the behavior of entities E_x and E_y when performing action a_i. The complex number representation for E_x when performing a_i is $c(E_x.a_i) = X_x + iY_x$ and for E_y when performing a_i is $c(E_y.a_i) = X_y + iY_y$. Consequently, the merged transactions representation when performing a_i is $c(E_{xy}.a_i) = (X_x + X_y) + i(Y_x + Y_y)$. The complex numbers are represented by either two coordinates on the real and imaginary axis or using a magnitude and an angle. The magnitude and angle representation is more interesting in our case to compute matching scores.

To see why the magnitude-angle representation can better help in detecting the existence of a similarity between two entities, consider the example described in Fig. 2. For clarity, we assume that there is only one action in the system and we need, by applying transactions merge, to know to what extent the behavior identification has been improved to suggest a potential similarity between the two entities. At the left of Fig. 2, entity E_1 complex number results in $mag_1 = 4.45$, and for E_2, it is $mag_2 = 4.62$. When merging E_1 and E_2's transactions, a pattern can be recognized and surprisingly the resulting $mag_{12} = 0.35$, is a smaller magnitude. The smaller magnitude length resulted because E_1 and E_2 transactions together formulate a smooth pattern in performing the action by filling gaps in the action sequence. Moreover, the feature values are close to each other. In the resulted vector E_{12}, the action is performed every 2 or 3 point of time and this produced a balanced vector, which is recognized as a pattern. Also, the feature values are close to each other, it is either 3 or 4 (i.e. it is stabilizing around these values). When converting a balanced vector to a complex number as described, the magnitude becomes small, because every entry in the vector pulls the resultant magnitude to a direction along a circle centered in the origin. In this case, we say the action pattern was enhanced and the features stabilization is supported. At the right of Fig 2, E_2 was merged with E_3. $mag_3 = 3.1$ and $mag_2 = 4.62$ and after merging the transactions, the resulted $mag_{23} = 7.18$, which is bigger. We should note that the resulting sequence from the merge does not have a recognized pattern; moreover, the feature values deviated further away (between 3 and 5). We are not interested in understanding the change in the angles and including it in the computations. Our aim is to come up with simple fast technique to produce candidate matches and minimize the cost spent in this operation.

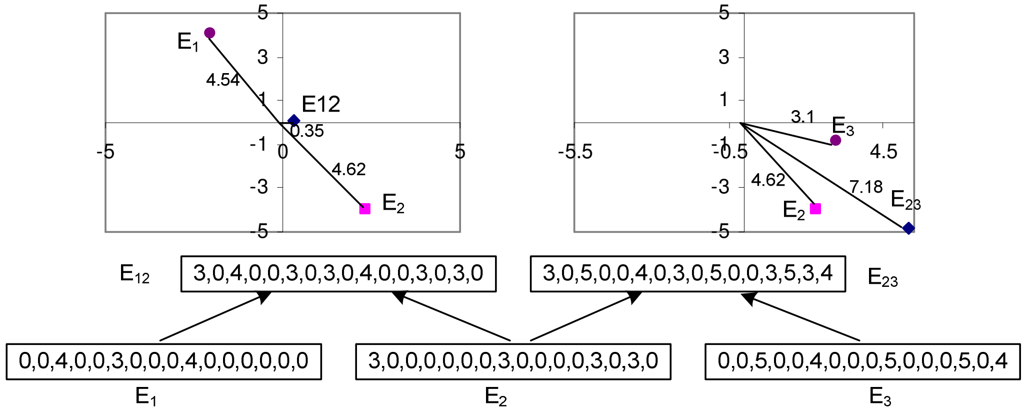


Fig. 2 Actions patterns in the complex and the effect on the magnitude

To develop a scoring formula based on the above observation, we consider merging the transactions of E_x and E_y with the assumption that there is only one action in the system. We also consider mag_x and mag_y as the magnitudes of the complex number representation of E_x and E_y respectively. The resulting magnitude of the merge, mag_{xy}, can take values between 0 and (mag_x+mag_y). The closer mag_{xy} is to 0, the more likely this supports the existence of pattern with similar features. The closer mag_{xy} is to (mag_x+mag_y) the less likely to have a pattern enhancement and consequently, the less likely for E_x and E_y to be similar. Accordingly, we propose a matching score formula as follow:

$$sim(E_x, E_y) = 1 - \frac{mag_{xy}}{mag_x + mag_y}$$

When we have *n common* actions between E_x and E_y, then the final formula will be:

$$sim(E_x, E_y) = \sum_{i=0}^{n-1} \left(1 - \frac{mag_{xy}(a_i)}{mag_x(a_i) + mag_y(a_i)} \right) \frac{S(E_{xy}.a_i)}{S(E_{xy})}$$

Where $S(E_{xy}.a_i)$ is the number of occurrences of action a_i in the merged transactions and $S(E_{xy})$ is the total number of merged transactions. Note that this is applied to only common actions between E_x and E_y, since uncommon actions will not be affected by merging the transactions.

The intuition behind including the percentage of the number of transactions of action a_i within the total number of transactions is to give a weighted effect for the actions. In another word, the higher the relative number of transactions for a given action, the more effect is expected in computing the score.

The proposed formula guarantees a score $sim \in \Re^+$ between 0 and 1. However, very low values are expected since a score equal to 1 is reached when all the actions performed by E_x are the same as E_y and the resulting transactions' merge magnitude equal zero in all the actions. This situation can hardly happen in real world situations. Instead, we normalize the resulted scores according to the maximum reached score (i.e. if sim_{max} is the maximum score reached, and then all scores are divided by sim_{max}). In the experiment, we show how to select a threshold, t_c, maintain no false negatives while achieving high reduction in the number of candidates generated.

The advantage of the described technique is that, by performing a one scan on the entire transaction log, we can compute the complex numbers for each action per a given entity. When computing the matching scores to produce candidates, simple complex numbers operations are employed. In our implementation, the sine and cosine values are pre-calculated to compute the complex numbers and all operations have been reduced to simple additions and multiplication in addition to small number of square roots to get the magnitudes.

Efficiency: Suppose that there are two sources P and Q with p and q entities respectively and each source has a log of size T_p and T_q respectively. The number of actions that can be performed and registered is n. First, a scan to both the transaction logs is performed to represent each action per entity as a complex number. This takes time of $O(T_p+T_q)$ and space of $O(np)$ and $O(nq)$ for sources P and Q respectively. Therefore, the total space requires is $O(n(p+q))$. Afterward, comparing every possible pq pairs of entities requires $O(n_c pq)$ since during the comparison all common actions n_c between every two entities are used. According to a matching threshold, only candidates for Phase 2 are stored in $O(C)$ space, where C is the number of resulted candidates and it is bounded by pq. Hence, the total time is $O(T_p+T_q+ n_c pq)$ and total space is $O(n(p+q)+C)$.

IV. Accurate Matching Phase

We now present the more accurate matching approach to identify similar entities that will have as input the candidate matches computed in the first phase. In this section, we propose the *Compressibility* approach that uses both actions repeating patterns and actions features stabilization to detect potential enhancements in identifying behavior that is more likely to represent one single entity.

Identifying behavior through repeating patterns and stabilized features can be heuristically achieved by compressing all this information and comparing the compression ratio with the original data size. We conjecture that significant higher compression ratio implies better identification of behavior. We thus introduce *compressibility* as a measure of confidence to identify behaviors. High compressibility means improved resolution for behavior.

Real world transaction logs usually build sparse Behavior Matrices with a lot of zeros. These zeros result in a miss-leading compression ratio without signification information about the behavior itself. Therefore, we use the vector-pair as a more practical representation for the behavior aiming at getting more meaningful information from the compressibility process.

In the Behavior Matrix, each action is represented by a row. In each row, at time point *i*, the cell contains either zero, if the action was not performed, or contains a feature value to represent how the action was performed. In the vector-pair representation, one vector represents the time at which the action was performed and the other stores the corresponding feature value. The time vector contains the inter-arrival time between every two consecutive occurrences of the action.

Example:

Suppose that an action has the following row in the Behavior Matrix

{3,0,4,0,0,3,0,3,0,4,0,0,3,0,3,0}

The vector-pair will be:

{{1,2,3,2,2,3,2}{3,4,3,3,4,3,3}}

If an entity performs an action regularly following a pattern, the time vector will contain inter-arrival time values that follow a certain level of correlation showing the action rate.

Moreover, the features vector will contain similar values to represent how the action was performed. Consequently, with this representation, we get rid of the zeros and at the same time the compressibility technique becomes more appealing to produce more significant information about the behavior.

Most of the existing compression techniques use data repetition and encodes it in a more compact representation. There are two types of compression techniques; lossless and lossy. Lossless techniques are used when every single bit in the compressed original data is important and should be exactly reconstructed upon decompression. Conversely, lossy techniques allow reconstructing data that is close enough to the original while achieving better compression ratio. In our case, the lossy compression is more attractive; we are not compressing the data for the sake of decompression, but rather we are trying to get a sense of how compressible the data is.

The Discrete Cosine Transformation (DCT) is widely used in the signal and image processing, especially for lossy compression techniques. It has the property of strong "energy compaction" [2], that is if the original data (signal) exhibit a correlation then most of the signal information tends to be concentrated in a few low-frequency components of the DCT. Therefore, by storing low-frequency coefficient, we can reconstruct data that is close enough to the original.

The most common DCT definition of a 1-D sequence $x_0, x_1, \ldots, x_{N-1}$ of length N is

$$T(u) = \alpha(u)\sum_{k=0}^{N-1} x_k \cos\left(\frac{\pi(2k+1)u}{2N}\right), \text{ for } u = 0, 1, 2, \ldots, N-1.$$

$$\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{N}} & for \quad u = 0 \\ \sqrt{\dfrac{2}{N}} & for \quad u \neq 0 \end{cases}$$

If $u = 0$, $T(u=0) = \sqrt{\frac{1}{N}}\sum_{k=0}^{N-1} x_k$ Thus, the first transformation coefficient is the average value of the sequence. Usually this value is referred to as the *DC coefficient*. All other transformation coefficients are called the *AC coefficient*.

The cosine basis functions which are produced from $\cos\left(\frac{\pi(2k+1)u}{2N}\right)$, $u = 1,2,\ldots N-1$ and $k = 1,2,\ldots.N-1$, are independent from the sequence $x_0, x_1, \ldots, x_{N-1}$. Therefore, these functions can be pre-computed and hence improving the performance.

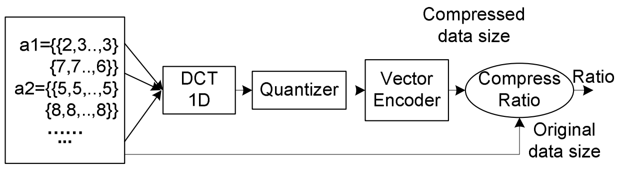


Fig. 3 Compression process

To perform a compression for the behavior, we follow the same approach used in JPEG [1]. However, instead of applying the procedure in two dimensions, we apply it in one dimension for the vector-pairs. The compression operation is illustrated in Fig. 3. First, we compute the 1-D DCT for the vector. Then, we divide the resulted DCT vector by a quantizer vector then round each value to the nearest integer. The quantization values are computed based on the sequence average and the position in the vector to reduce the amount of information in the high frequency components. The lost data should help distinguish minor details in the behavior representation which are not important and most of the time is noise. Finally, a straightforward vector encoding technique is used to compress the transformed vector.

Based on the compressibility approach, we can identify eventually similar behavior; if the merged transactions of the two entities E_x and E_y exhibit more compressibility, then they are more likely to match.

To compute the matching score of two entities E_x and E_y, we define the directed compressibility gain $g(E_x,E_y)$ and $g(E_y,E_x)$. $g(E_x,E_y)$ is the gain score for E_x when merged with E_y and $g(E_y, E_x)$ goes in the other direction. To compute each of these gains, we suppose there are n actions. Assume also that when compressing the vector-pairs, we obtain a compression ratio $cr(E_x.a_i)$ for each action a_i in E_x's transactions and for E_y $cr(E_y.a_i)$. After merging their transactions and compressing the resulting vector-pairs, we obtain a compression ratio $cr(E_{xy}.a_i)$ for each a_i. The overall compression ratio of E_x and E_y:

$$cr(E_x) = \frac{\sum_{i=0}^{n-1}\left[S(E_x.a_i)cr(E_x.a_i)\right]}{S(E_x)} \quad \text{and} \quad cr(E_y) = \frac{\sum_{i=0}^{n-1}\left[S(E_y.a_i)cr(E_y.a_i)\right]}{S(E_y)}$$

Similarly, the compression ratio of E_{xy},

$$cr(E_{xy}) = \frac{\sum_{i=0}^{n-1}\left[S(E_{xy}.a_i)cr(E_{xy}a_i)\right]}{S(E_{xy})}$$

The support values are included to provide a weighted effect for the actions on the overall compression ratio for the entity's behavior.

The gain in the compressibility for E_x from the merge with E_y is:

$$g(E_x, E_y) = cr(E_{xy}) - cr(E_x)$$

and

$$g(E_y, E_x) = cr(E_{xy}) - cr(E_y)$$

In the following section, we see how to use these scores to produce the final matching results.

Efficiency: Suppose that the number of candidate pairs is C and there are n actions in the systems that can be registered. Also, suppose the average vector-pair length is $t`$ (note that the $t`$ is bounded by the t, the total time period length). In our implementation, the transactions are stored in a database to facilitate the retrieval. For each candidate entity pairs, a query is submitted to get their transactions (i.e. $O(C)$ queries). Afterward, for each action (i.e. $O(n)$) a vector-pair is formulated to be compressed. The compression took $O(t`^2)$. Hence, the total time is $O(Cnt`^2)$. The space requirement is constant $O(1)$.

V. Filtering Matches

We discuss now how to use the computed scores from the compressibility phase to produce the final matches. The set of matched records are called a mapping, where every record is mapped to its matched one. The problem of finding the best mapping is closely related to well-known matching problems in bipartite graphs (see e.g. [24, 25]). A bipartite graph is one whose nodes form two disjoint parts such that no edge connects any two nodes in the same part. Thus, a mapping can be viewed as an undirected weighted bipartite graph.

We use the intuition of the *stable marriage* [24] problem to help us finding the best mapping. In an instance of the stable marriage problem, each of *n* women and *n* men lists the members of the opposite sex in order of preference. The goal is to find the best match between men and women. A stable marriage is defined as a complete matching of men and women with the property that there are no two couples (x, y) and (x', y') such that x prefers y' to y and y' prefers x to x'. Such situation would be regarded as unstable.

In our case, we do not have equal number of men and women (entities) on both sides and there are men and women (entities) that should not be mapped, but they will have a matching score anyway even if it is very low. We overcome this by small modification to (1) allow for non equal numbers of men and women to be mapped, and (2) use a threshold to filter entities that should not be mapped. This approach is similar to the *SelectTheshold* technique described in [26], with the difference that the authors used relative scoring in an undirected graph and we are using global scores in a directed graph.

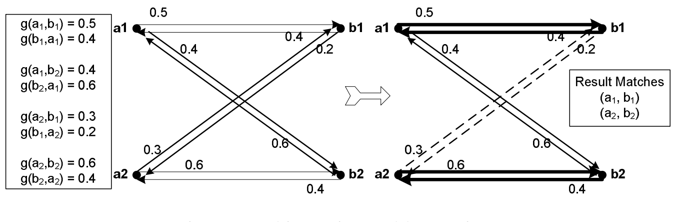


Fig. 4 Matching using stable marriage

To demonstrate how the filtering works, consider Fig. 4, which shows 4 entities with the compressibility gain scores represented by the weights on the directed edges. First, we remove the directed edges with gain scores less than a threshold t_m=0.3. The discarded scores are shown on the right of Fig. 4 with doted edges. The rest of the scores are then used to order the preference of each node to apply a stable marriage algorithm and finally get the mapping. In this case, b_1 will reject a_2 as a match and the final mapping will be (a_1, b_1), (a_2,b_2).

VI. Experiments

In this section, we report the results of our experimental study. The goals of the study are as follows:

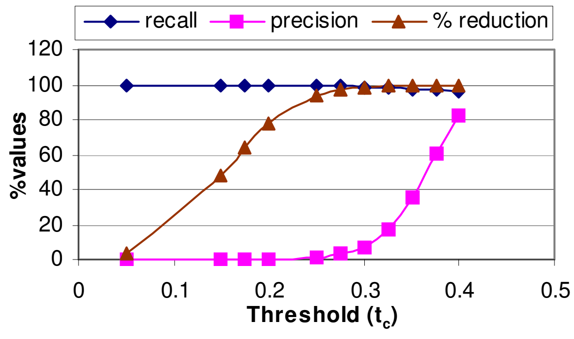
- Evaluating the matching quality of the proposed technique and demonstrating the effectiveness of the two matching phases. Also, the data characteristics effect is considered in the study.
- Studying the performance of the approach and the effectiveness of the candidates generation phase on the overall performance. Also the dataset characteristics are considered in the evaluation.
- Demonstrating the scalability of the technique along three parameters; log size, number of entities and number of actions in the systems.

In the experiments, we used a real world transactions log, representing transactions of a Walmart store customers. The transactions we have cover the period from July 31, 1999 to November 2, 2000 and contain more than 5 million customers, who can buy from 432,223 items. The total number of transactions is over 800 million. To simulate the existence of two data sources whose customers (entities) need to be linked, we divided the Walmart data into two. We randomly divide the transactions of some customers, selected randomly, between the two stores. We also control the expected overlapping between the customers in the two virtual stores. This large dataset helped us to create different datasets with different characteristics. This way, we can study our technique sensitivity with respect to different data properties.

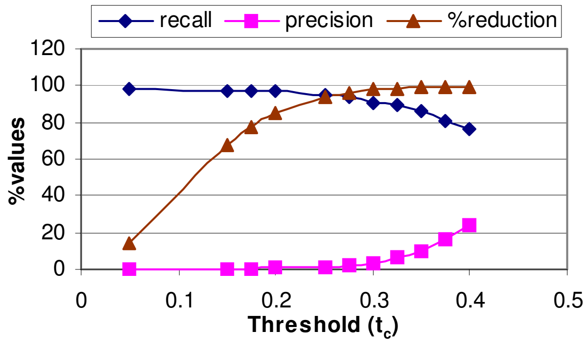
All the experiments were conducted on a PC with a 3 GHz Pentium 4 processor and 1 GB RAM running Windows XP. We used Java to implement the proposed technique and we used MySql DBMS to store and query the transactions for processing and to store intermediate results.

A. Quality

The matching quality of the proposed technique is studied by reporting precision and recall of the resulting mapping. The recall measures the percentage of correctly matched pairs over all pairs of records that refer to the same entity, and the precision measures the percentage of correctly matched pairs over all true matches. Since we are controlling the number of overlapping entities in each of the datasets, we can identify the already matched entities to get the precision and recall. In this experiment, we used two subsets from the divided Walmart datasets; one with about 1200 average number of transaction per customer. This is considered a dense dataset and referred as Dataset 1. The other has about 700 transactions per customer, which is less dense dataset, and referred as Dataset 2. We used different density of transaction for the purpose of studying how this data characteristic will affect the matching accuracy.



(a) Dataset 1



(b) Dataset 2

Fig. 5 Candidate phase effectiveness

In Fig. 5.a and 5.b, we illustrate the effectiveness of the candidates generation phase for each of the two datasets. We report the recall, precision and the percentage of reduction on the number of candidates against the matching score threshold t_c. The candidates' reduction is computed as follows: suppose that the two data sources contain p and q records and that the number of generated candidates is C pairs. The reduction percentage $r = 100(pq - C)/pq$. For Dataset 1, it is noted that most of the time the recall is significantly high up to more than 90%. On the other hand, the precision takes low values and improves from 10% to about 80% with the increase of t_c between 0.3 and 0.4. For high values of t_c, the recall decreases and the precision increases. This is expected since the matching decision becomes stricter while using inaccurate matching in this phase and consequently; this leads to have more false negatives. The percentage of reduction in generated candidates started with low values and quickly increases to more than 90% with the increase in t_c especially after 0.3. This is also because the matching becomes stricter. Minimizing the number of candidates results in less effort in the compressibility phase which is expensive.

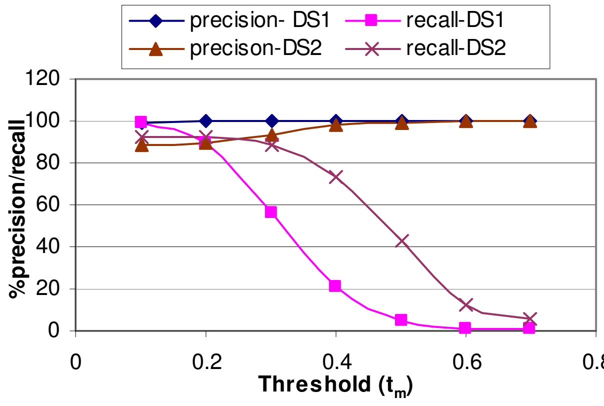
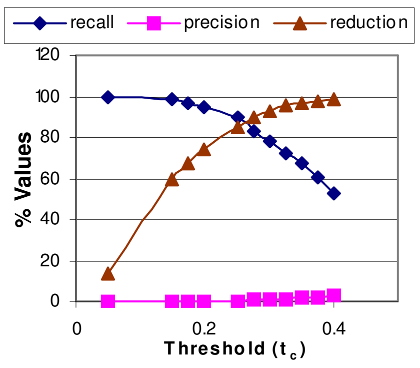


Fig. 6 Overall accuracy: Dataset 1 & 2

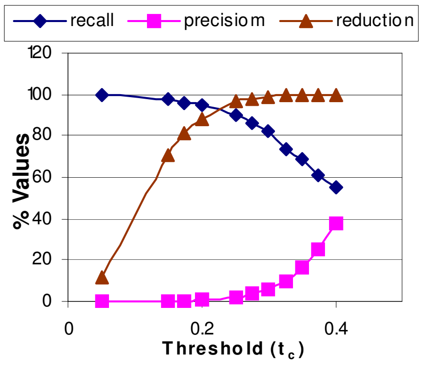
For Dataset 2 in Fig. 5.b, the precision increases slowly, while the recall drops much faster than for Dataset 1. This is because Dataset 2 is less dense and contains less information. The reduction in the candidate matches is almost the same as Dataset 1. For both Fig 5.a and 5.b, it is noted that the candidates generated for the compressibility phase is significantly reduced to more than 90% especially for t_c values more than 0.25, while maintaining high recall. The precision is not expected to be high during this phase, however. Dataset 1 showed a noticeable high precision for high t_c values. This is because Dataset 1 is dense; the more transactions, the more information are available for better matching decision even in the first phase.

The overall matching accuracy of the process after applying the compressibility phase and the mapping filter is illustrated for each of the two datasets in Fig 6. We report for each of the datasets the precision and recall. To get this results we used $t_c = 0.25$ as similarity threshold value in the candidates generation phase. This value showed for both datasets more than 95% recall and more than 90% reduction in candidates' number. In both datasets, high recall and precision values have been achieved for low mapping threshold values t_m especially between 0.1 and 0.2. As t_m decreases, the recall slightly deceases while the precision significantly increases. Dataset 1 showed higher recall and precision than Dataset 2 for t_m between 0.1 and 0.2. This is because Dataset 1 is denser and contains more information for matching. Generally, it is noted how significantly the precision and overall matching accuracy are improved by the compressibility phase.

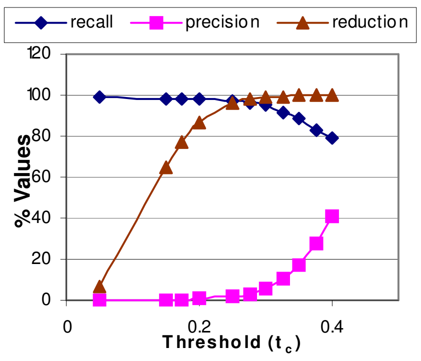
In our next experiment, we study the effect of distributing an entity's transactions between the data sources. In our two stores example, a customer may use one of the stores more than the other, or he may equally use them. Therefore, we decide to study the effect of the percentage of distributing an entity's transaction among the data sources. To do this, we used Walmart dataset to produce 3 pairs of datasets each representing different two stores. We managed in each of the dataset pairs to divide randomly some the customers' transactions to reach the percentage of division 40%, 25%, and 10% respectively.



(a) Transactions divided by 10%



(b) Transactions divided by 25%



(c) Transactions divided by 40%

Fig. 7 Studying the transactions division on the candidates matching performance.

Fig. 7 shows the results of the first matching phase for each of the dataset pairs. We observe that as division percentage increases, the best achieved values for both the recall and candidates reduction increase. This can be noted in the three figures at thresholds between 0.2 and 0.3. In Fig 7.c where the percentage is 40%, we are able to reach 95% recall while achieving also more than 95% reduction in the candidates. The precision is very low as expected; however, it gets worse as the division percentage is reduced to 10% in Fig 7.a, where the precision can hardly reach 5%.

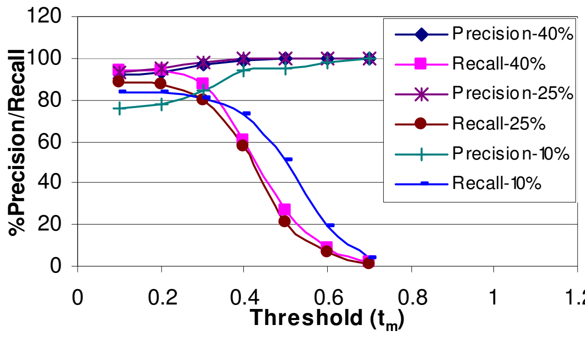


Fig. 8 Studying transactions division: Overall accuracy

The overall accuracy when linking the three datasets pairs is illustrated in Fig. 8. We report in this figure the achieved precision and recall after applying the accurate matching phase. To get these results, we used in the first phase threshold value $t_c = 0.25$, which demonstrated high recall and at the same time high reduction in the number of candidates. It is noted for all the datasets that the precision and recall behaves similar to the results achieved in Fig. 6 in the previous experiment. Both the precision and recall take reasonably high values between 77% and 95% for low t_m values. As t_m increases the recall values gets improved while the precision dramatically decreases. The effect of transactions percentage distribution is noticed such that pair datasets with smaller percentage values (i.e. customers tend o use one of the store most of the time) show worst precision and recall in its best cases at t_m between 0.1 and 0.2. In the figure, pair dataset with division 10%, achieved around 80% recall and precision at the same time for t_m between 0.1 and 0.2. On the other hand, dataset pair with division 40% achieved more then 90% recall and precision for the same t_m values. To conclude, when entities' transactions are divided almost equally between two data sources, this helps in achieving high matching accuracy. Despite this fact, our technique matching quality for low transactions division was acceptable.

B. Performance

Our next set of experiments study the execution time of the matching process. We mainly focus on analyzing the time spent in each of the two phases of the proposed technique. Also, we illustrate the effectiveness of the threshold t_c used in the candidates generation phase on the overall linking time.

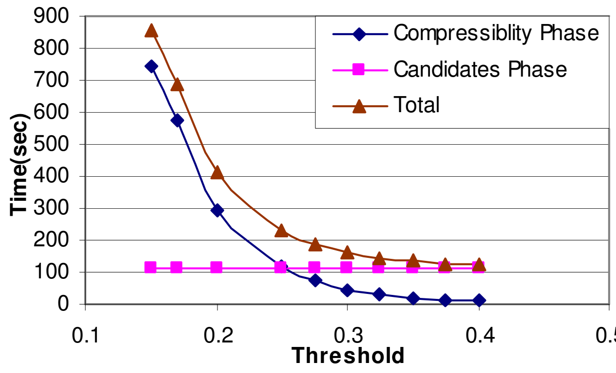
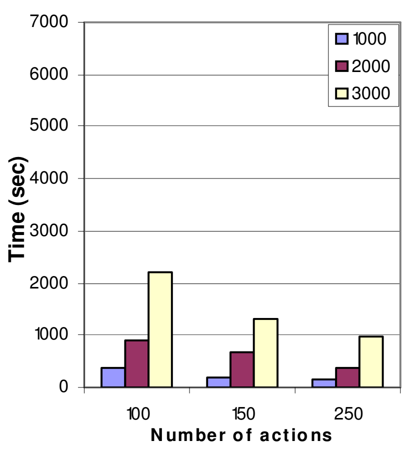
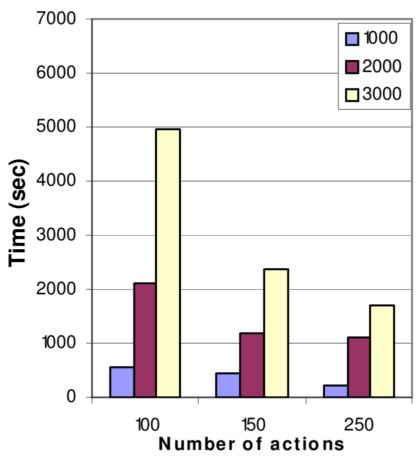


Fig. 9 Studying execution time.

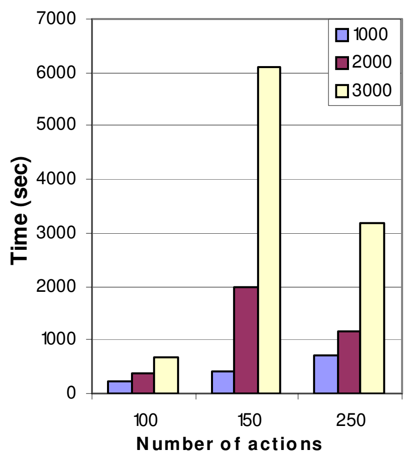
To study the execution time of our technique, we linked two datasets each with about 1000 customers with average number of transactions for each customer is 1000. In Fig. 9 we report the total execution time in addition to the time spent in each phase against different values of t_c threshold, which is used in the candidates phase. The candidates phase took 115 sec; the candidate phase execution time is not affected by the t_c, because all the pairs of records should be compared anyway and then filtered based on t_c selected value. For each value of the threshold, the candidates are passed to the compressibility phase to produce the final mapping.



(a) Avg. transactions/entity = 400



(b) Avg. transactions/entity = 550



(c) Avg. transactions/entity = 700

Fig. 10 Studying scalability sensitivity.

For high t_c values, the compressibility phase execution time is very low and hence the overall time is low. The reason is that the number of produced candidates for high t_c is small. As illustrated in the previous experiment of Fig. 6, this comes at the cost of accuracy and results in a lot of false negatives. As t_c decreases the compressibility time dramatically increases and consequently the overall time increases. For very low values of t_c, the candidate phase produces almost all possible pairs to be match in the compressibility phase. We also note that the compressibility phase is very expensive if it is used alone without the candidates phase.

A. *Scalability*

In the following experiment, we study the scalability of our technique along three important data characteristics; the number of transaction, the number of entities and the number of actions that can be registered. We used Walmart dataset and constructed different pairs of datasets to be linked. In each dataset pair, we controlled the number of entities, the average number of transactions per entity and the number of actions. Note that the numbers of entities along with the average number of transactions per entity control the total number of transactions.

The result of the experiment is depicted in Fig. 10 along three graphs. From left to right we use 400, 550 and 700 as average number of transactions per entity. Within each graph, we report the execution time when the number of entities takes the values 1000, 2000 and 3000 against changing the number of actions in the system among 100, 150 and 250.

It is noted in each of Fig. 10.a and 10.b that with the increase in the number of actions, the execution time decreases; however, this property does not hold in Fig. 10.c where the average number of transactions per entity increased to 700. It is worthy noting that with the increase in the number of actions and fixing the average transactions per entity, the entity's behavior will contain high number of actions that are rarely performed. In our implementation, we neglect such actions and so this minimizes the execution time, while maintaining more accurate results. In Fig. 10.c, increasing the average transactions allows for having more effective actions in the computations and consequently, increases the execution time.

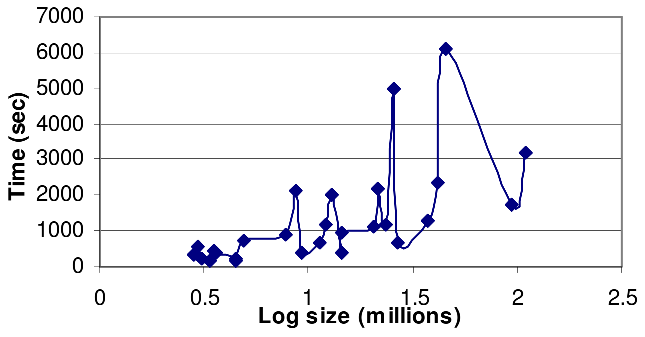


Fig. 11 Scalability with log size

Generally from the three graphs in Fig. 10, increasing the number of entities along with increasing their average transactions increases the total execution time because this results in larger log to be processed. However in Fig. 11 when reporting (for the same datasets used in Fig. 10) the execution time verses the size of the log, the performance vary because the log size is not the most effective parameter for our technique however; the number of entities and actions could be more effective and this supports the practical sense of the technique.

VII. Related Work

Record linkage has received significant attention in the literature and it has many variations like de-duplication [8], hardening soft databases [6], reference matching [7], object identification [5], identity uncertainty [9], entity resolution [3], mention matching [12] and reference reconciliation [4].

Most of the existing techniques for record linkage depend on textual based attributes and use several approaches for string approximate matching (refer to [10] and [11] for recent surveys). Recently, more involved techniques presented to make use of extracted information from the data to improve the linkage accuracy. We view our contribution as complementary to these techniques.

The idea of extracting information and knowledge to capture similarities between entities has recently been explored in the data mining and machine learning community. In [16], a complex generative model is proposed that captures dependencies between various classes and attributes and also possible errors during entities matching. In [17], a dependency model is proposed that propagates similarity decisions through shared attribute values. Both the above approaches entail learning a global detailed probabilistic model from training data, and having the entire matching process guided by that probabilistic model. In [18] and [19], associations are used to compute similarities and relate matching decisions. [20] proposed an approach in which entities are matched by a sequence of comparison and matching steps with different similarity measures being used in different steps. Merging between steps was used to increase information about individual references. The use of negative information was proposed in [21] to validate individual resolution decisions. Also, an interesting approach for making use of aggregate constraints in a relational database to improve records matching was introduced in [22].

Performing the records matching level-wise or on a compositional manner was introduced in [14], [15] and [12]. The work in [14] and [15] focuses on improving the run-time efficiency of the matching process. While the work in [12] introduces a more general, compositional, multi-component approach for records matching.

VIII. Conclusions

In this paper, we presented a new technique that uses a given entity behavior, which can be recognized in an entity's transaction log; to help in improving the record linkage accuracy. We characterized the behavior in two main dimensions; actions repetition pattern and actions features. When comparing two entities for matching; first, their transactions are merged and then, we measure the improvement in identifying the behavior.

Since the transaction log is expected to be long, we proposed a two-phase matching process; in the first phase, candidate pairs are quickly generated for matching, while having negligible false negatives and many false positives. The second matching phase improves the matching precision by eliminating the false positives. The second phase is based on a heuristic called compressibility. It is based on the fact that repeated patterns and stable features result in a Behavior Matrix that is more compressible. Our experiments prove that the technique can effectively improve the matching quality, while being practical to handle large logs. Also, the run time performance is dramatically affected by the number of actions and entities in the system, while it is slightly affected by the total number of transactions.

References

[1] G. K. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Transaction on Consumer Electronics*, Vol. 38, No 1 , Feb 1992.

[2] K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, Applications", *Academic Press Professional*, Boston, 1990.

[3] I. Bhattacharya and L. Getoor, "Collective Entity Resolution in Relational Data", *ACM Transactions on Knowledge Discovery from Data*. Vol. 1, Article 5, March 2007.

[4] X. Dong, A. Halevy and J. Madhavan, "Reference Reconciliation in Complex Information Spaces", *Proceedings of the 2005 ACM SIGMOD*, June 2005.

[5] S. Tejada, C. Knoblock, and S. Minton. "Learning domain-independent string transformation weights for high accuracy object identification". In *SIGKDD*, 2002.

[6] W. W. Cohen, H. Kautz, and D. McAllester. "Hardening soft information sources". In *SIGKDD*, 2000.

[7] A. K. McCallum, K. Nigam, and L. H. Ungar. "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching". In *SIGKDD*, 2000.

[8] S. Sarawagi and A. Bhamidipaty. "Interactive deduplication using active learning". In *SIGKDD*, 2002.

[9] A. McCallum and B. Wellner. "Toward conditional models of identity uncertainty with application to proper noun coreference". In *IIWEB*, 2003.

[10] A.K. Elmagarmid, G.I. Panagiotis, S.V. Verykios, "Duplicate Record Detection: A survey", IEEE TKDE 19 (2007), no. 1

[11] N. Koudas, S. Sarawagi and D. Srivastava, "Record Linkage: Similarity measures and algorithms." In *Proc of ACM SIGMOD,* 2006.

[12] W. Shen, P. DeRose, L. Vu, A. Doan and R. Ramakrishnan, "Source-aware Entity Matching: A Compositional Approach", In the *Proc of ICDE*, 2007.

[13] S. Chaudhuri, A. D. Sarma, V. Ganti and R. Kaushik, "Leveraging aggregate constraints for deduplication", In *the Proc. of ACM SIGMOD*, 2007.

[14] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, J. Widom, "Swoosh: A Generic Approach to Entity Resolution", The VLDB Journal, 2008.

[15] M. G. Elfeky,A. K. Elmagarmid and V. S. Verykios,"TAILOR: A Record Linkage Tool Box", In *the Proc. of ICDE*, 2002.

[16] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. "Identity Uncertainty and Citation Matching". In *NIPS*, 2002.

[17] Parag and P. Domingos. "Multi-relational Record Linkage". In *MRDM*, 2004.

[18] I. Bhattacharya and L. Getoor. "Iterative Record Linkage for Cleaning and Integration". In *DMKD*, 2004.

[19] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. "Exploiting Relationships for Domain-independent Data Cleaning". In *SIAM Data Mining (SDM)*, 2005.

[20] X. Dong, A. Halevy, E. Nemes, S. Sigurdsson, and P. Domingos. "Semex: Toward on-the-fly Personal Information Integration". In *IIWeb*, 2004.

[21] A. Doan, Y. Lu, Y. Lee, and J. Han. "Object Matching for Information Integration: A Profiler-based Approach". In *IIWeb*, 2003.

[22] X. Dong and A. Halevy. "A Platform for Personal Information Management and Integration". In *Proc. of CIDR*, 2005.

[23] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," Journal of the American Statistical Association, vol. 64, no. 328, pp. 1183-1210, 1969.

[24] D. Gusfield and R. Irving. "The Stable Marriage Problem: Structure and Algorithms". *MIT Press*, Cambridge, MA, 1989.

[25] L. Lovasz and M. Plummer. "Matching Theory". North- Holland, Amsterdam, 1986.

[26] S. Melnik, H. Garcia-Molina and E. Rahm, "Similarity flooding: a versatile graph matching algorithm and its application to schema matching", In the *Proc. of ICDE*, 2002.