# Adaptive Temporal Entity Resolution on Dynamic Databases*

Peter Christen[1] and Ross W. Gayler[2]

[1] Research School of Computer Science, The Australian National University, Canberra ACT 0200, Australia
peter.christen@anu.edu.au
[2] Veda, Melbourne VIC 3000, Australia
ross.gayler@veda.com.au

**Abstract.** Entity resolution is the process of matching records that refer to the same entities from one or several databases in situations where the records to be matched do not include unique entity identifiers. Matching therefore has to rely upon partially identifying information, such as names and addresses. Traditionally, entity resolution has been applied in batch-mode and on static databases. However, increasingly organisations are challenged by the task of having a stream of query records that need to be matched to a database of known entities. As these query records are matched, they are inserted into the database as either representing a new entity, or as the latest embodiment of an existing entity. We investigate how temporal and dynamic aspects, such as time differences between query and database records and changes in database content, affect matching quality. We propose an approach that adaptively adjusts similarities between records depending upon the values of the records' attributes and the time differences between records. We evaluate our approach on synthetic data and a large real US voter database, with results showing that our approach can outperform static matching approaches.

**Keywords:** Data matching, record linkage, dynamic data, real-time matching.

## 1 Introduction

Entity resolution is the process of identifying, matching, and merging records that correspond to the same entities from several databases [1]. The entities under consideration commonly refer to people, such as patients or customers. The databases to be matched often do not include unique entity identifiers. Therefore, the matching has to be based on the available attributes, such as names, addresses, and dates of birth. Entity resolution is employed in many domains, the most prominent being health, census statistics, national security, digital libraries, and deduplication of mailing lists [2–4].

* This research was funded by the Australian Research Council (ARC), Veda Advantage, and Funnelback Pty. Ltd., under Linkage Project LP100200079.

J. Pei et al. (Eds.): PAKDD 2013, Part II, LNAI 7819, pp. 558–569, 2013.

| RecID | EntID | Given name | Surname | Street address | City | Postcode | Time-stamp |
|---|---|---|---|---|---|---|---|
| $r1$ | $e1$ | Gale | Miller | 13 Main Road | Sydney | 2000 | 2006-01-21 |
| $r2$ | $e2$ | Peter | O'Brian | 43/1 Miller Street | Sydney | 2010 | 2006-02-21 |
| $r3$ | $e1$ | Gail | Miller | 11 Town Place | Melbourne | 3003 | 2007-01-28 |
| $r4$ | $e1$ | Gail | Smith | 42 Ocean Drive | Perth | 6010 | 2007-07-12 |
| $r5$ | $e2$ | Pete | O'Brien | 43 Miller Street | Sydney | 2610 | 2008-01-11 |
| $r6$ | $e1$ | Abigail | Smith | 42 Ocean Drive | Perth | 6010 | 2008-06-30 |
| $r7$ | $e2$ | Peter | OBrian | 12 Nice Terrace | Brisbane | 7011 | 2009-01-01 |
| $r8$ | $e1$ | Gayle | Smith | 11a Town Place | Sydney | 2022 | 2009-04-29 |

**Fig. 1.** Example data set of eight records representing two entities

While entity resolution has traditionally been applied in batch mode and on static databases, an increasingly common scenario in many application domains is the model of data streams [5], where query records (potentially from several sources) need to be matched with (and potentially inserted into) a database that contains records of known entities.

An example application of entity resolution in such a dynamic environment is the verification of identifying information provided by customers at the time of a credit card or loan application. In many countries, the financial institutions where customers apply will send the customers' identifying information to a central credit bureau. The responsibility of this bureau is to match the query to a pre-existing credit file to retrieve that customer's credit history, and to determine that the customer is the person they claim to be [6]. The credit bureau receives a stream of query records that contain identifying information about people, and the bureau's task is to match these records (in real-time) to a large database of known validated entities. In this application, accurate entity resolution is crucial to avoid inappropriate lending and prevent credit application fraud [6].

Temporal and dynamic aspects in the context of entity resolution have only been investigated in the past three years [7–11]. Most similar to our work is the technique proposed by Li et al. [8, 9] on linking temporal records. Their approach assumes that all records in a database have a time-stamp attached, as illustrated in Fig. 1. These time-stamps are used to calculate *decay* probabilities for combinations of individual attributes and time differences. Based on these probabilities the similarities between records (calculated using approximate string comparison functions on individual attributes [2]) are adjusted. The *agreement decay* was defined by Li et al. [8] as the probability that two different entities, represented by two records with a time difference of $\Delta t$, have the same value in an attribute. The *disagreement decay* was defined as the probability that the value in an attribute for a given entity changes over time $\Delta t$. Such a change occurs if, for example, a person moves to a new address.

In Fig. 1, for the 'City' attribute and $\Delta t \in [0, 1)$ year, there are three pairs of records by the same entity where the attribute value has changed: *(r3,r4)* and *(r6,r8)* for *e1*, and *(r5,r7)* for *e2*, and one pair where the value did not change: *(r4,r6)*. The disagreement probability for this attribute and $\Delta t \in [0, 1)$ year is therefore 75%. As both entities live in 'Sydney' in 2006, we can calculate an agreement probability for this value. However, due to the sparseness of this data set, we cannot calculate agreement probabilities for other 'City' values.

Li et al. [8] calculated both agreement and disagreement probabilities such that they increase monotonically as $\Delta t$ gets larger. However, as Fig. 2 shows, this is not necessarily the case. Their approach also assumes that the databases from which records are matched are static; that the decay probabilities are learned once in an off-line process using a supervised learning technique (learning disagreement probabilities is of linear complexity in the number of entities, while learning agreement probabilities is of quadratic complexity); and that these decay probabilities are independent of the frequencies of individual attribute values. The experiments conducted on a bibliographic data set showed that taking such temporal information into account can improve matching quality [8].

In contrast, our approach is aimed at facilitating an efficient adaptive calculation of weights that are used to adjust similarities between records. While our approach to calculate disagreement probabilities is similar to Li et al. [8], we calculate agreement probabilities that incorporate the frequency distributions of the attribute values. This is similar to frequency-based weight adjustments as applied in traditional probabilistic record linkage [4]. As an example, if two records have an agreeing surname value 'Smith', which is common in many English speaking countries, then it is more likely that they correspond to two different entities compared to two records that have the uncommon surname value 'Dijkstra'. As our experimental study on a large real-world database shows, taking these frequencies into account can lead to improved matching accuracy.

Our contributions are (1) an adaptive matching approach for dynamic databases that contain temporal information; (2) an efficient temporal adjustment method that takes the frequencies of attribute values into account; and (3) an evaluation of our approach on both synthetic data (with well controlled characteristics) and a large real voter database from North Carolina in the USA.

## 2 Related Work

Most research in entity resolution over the past decade has focused on improving quality and scalability when matching databases. Several recent surveys provide overviews of the research field [2–4]. Besides the work by Li et al. [8, 9] on linking temporal records (described above), several other recent approaches have investigated temporal or dynamic aspects in entity resolution.

Whang et al. [10] developed an approach to matching databases where matching rules can evolve over time, and where a complete re-run of an entity resolution process is not required when new data become available. The assumption is that a user provides an initial set of matching rules and over time refines these rules. While the rules in this approach are evolving, no temporal information in the records that are matched is taken into account in the matching process.

Ioannou et al. [7] proposed an entity query system based on probabilistic entity databases, where records and their attributes are assigned probabilities of uncertainty that are incorporated into the matching process. These probabilities correspond to the confidence one has that an attribute value has been recorded correctly for an entity. During the matching process a dynamic index data structure is maintained which contains sub-sets of entities that are connected through

common attribute values. Related to this work, Christen et al. [12, 13] investigated techniques to facilitate the real-time matching of query records to a large static database by incorporating the similarities calculated between attribute values into a novel index data structure. Both of these approaches however do not consider temporal aspects of the databases that are matched.

Yakout et al. [11] developed a matching approach for transactional records that correspond to the behaviour of entities (such as shopping baskets of individuals) over time, rather than the actual entity records. Their approach converts the transactions of an entity into a behaviour matrix. Entities are then matched by calculating similarities between condensed representations of their behaviour matrices. While temporal information is used to generate sequences of transactions for an entity, this information is not used in the matching process.

Pal et al. [14] recently presented an approach to integrate Web data from different sources where temporal information is unreliable and unpredictable (such as updates of changes are missing or incomplete). The temporal aspects of updates of entities are modelled with a hidden semi-Markov process. Results on a diverse set of data, from Twitter to climate data, showed encouraging results. The focus of this work is however to compute the correct value of an entity's attributes, not to identify and match records that refer to the same entity.

A large body of work has been conducted in the areas of stream data mining [5], where temporal decays are commonly used to give higher weights to more recent data, and temporal data mining [15], where the aim is to discover patterns over time in temporal data. To summarise, while either temporal information or dynamic data have been considered in recent approaches to entity resolution, our approach is a first to consider both, with the aim to achieve an entity resolution approach that adapts itself to changing data characteristics.

## 3 Modelling Temporal Changes of Entities

We assume a stream of query records $q$, which are to be matched (as detailed in Sect. 4) to a database $\mathbf{R}$ that contains records about known entities. We denote records in $\mathbf{R}$ with $r_i$, $1 \leq i \leq N$, with $N$ being the number of records in $\mathbf{R}$. At any point in time $N$ is fixed, but over time new records (such as matched query records) are added to $\mathbf{R}$ while the existing records are left unchanged. The records in $\mathbf{R}$ consist of attributes, $r_i.a_j$, $1 \leq j \leq M$, with $M$ the number of attributes. Examples of such attributes include name and address details, phone numbers, or dates of birth. The records also contain a time-stamp $r_i.t$, and an entity identifier, $r_i.e$. All records that correspond to the same entity are assumed to have the same unique value in $r_i.e$. In a supervised setting, the $r_i.e$ are the true known entity identifiers, while in an unsupervised setting the values of $r_i.e$ are based on the match classification, as will be described further in Sect. 4.

Assume we want to calculate the similarity between query record $q$ and record $r_i$ in $\mathbf{R}$. These two records have a difference in their time-stamps of $\Delta t = |q.t - r_i.t|$. For a single attribute $a_j$, we classify the attribute to be *agreeing* if both records have the same attribute value ($q.a_j = r_i.a_j$), or if the two attribute

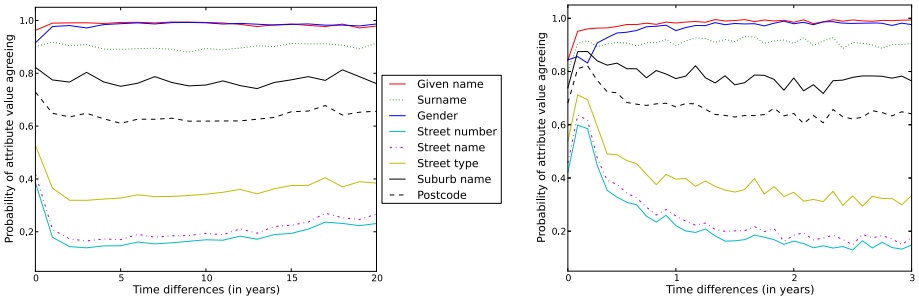


**Fig. 2.** The probabilities of an entity not changing its attribute values over time (i.e. the attribute values are agreeing), $P(A_j, \Delta t|S)$, as calculated from over 2.4 million entities in a North Carolina (NC) voter database. The x-axes show the time-differences $\Delta t$ between two records that refer to the same entity for up-to 20 years, with the right-hand side plot zooming into the time differences up-to three years. As can be seen, address values are more likely to change over time than name values.

values are highly similar, i.e. $sim_j(q.a_j, r_i.a_j) \geq s_{same}$, with $sim_j(\cdot,\cdot)$ being the similarity function used to compare values for attribute $a_j$, and $s_{same}$ a minimum similarity threshold. If, on the other hand, $sim_j(q.a_j, r_i.a_j) < s_{same}$, then we classify the attribute to be *disagreeing*. Similarity values are assumed to be normalised, with $sim_j(\cdot,\cdot) = 0$ for two attribute values that are completely different, and $sim_j(\cdot,\cdot) = 1$ for two values that are the same. Similarity functions vary by attribute and may be domain specific. For string attributes, such as names, approximate string similarity functions are commonly used [2].

To take temporal aspects into account, we need to consider the following events. We denote the event that $q$ and $r_i$ actually refer to the same entity with $S$, and the event that they actually refer to two different entities with $\neg S$. Furthermore, we denote the event that $q$ and $r_i$ have an agreeing value in attribute $a_j$ with $A_j$, and the event that they have a disagreeing value in attribute $a_j$ with $\neg A_j$.

We now consider the following two probabilities. As discussed in Sect. 3.1 below, they are used to adjust the similarities $sim_j(\cdot,\cdot)$ according to the time difference $\Delta t$ between records $q$ and $r_i$.

$$P(A_j, \Delta t|S) = P(sim_j(q.a_j, r_i.a_j) \geq s_{same} \wedge |q.t - r_i.t| = \Delta t \mid q.e = r_i.e) \quad (1)$$

is the probability that a query and a database record that actually refer to the same entity have an agreeing value in attribute $a_j$ over $\Delta t$ (i.e. the value does not change). It holds that $P(A_j, \Delta t|S) = 1 - P(\neg A_j, \Delta t|S)$.

$$P(\neg A_j, \Delta t|\neg S) = P(sim_j(q.a_j, r_i.a_j) < s_{same} \wedge |q.t - r_i.t| = \Delta t \mid q.e \neq r_i.e) \quad (2)$$

is the probability that a query and a database record that actually refer to two different entities have disagreeing (i.e. different) values in attribute $a_j$ over $\Delta t$. Clearly, $P(\neg A_j, \Delta t|\neg S) = 1 - P(A_j, \Delta t|\neg S)$.

The probability $P(A_j, \Delta t|S)$ can be learned for individual attributes and different $\Delta t$ by counting the number of agreeing and disagreeing attribute values for pairs of records of the same entity that have a time difference of $\Delta t$.

Calculating the probability $P(\neg A_j, \Delta t|\neg S)$ is more difficult because it requires the comparison of attribute values across records that have a time difference of $\Delta t$ and where the records refer to different entities. Such an approach, which is of quadratic complexity in the number of entities, was employed by Li et at. [8, 9]. Our approach, described in Sect. 3.2, is aimed at efficiently calculating the two probabilities (1) and (2) in an adaptive fashion. First we present how these probabilities are used to adjust the similarities between records.

### 3.1 Adjusting Similarities between Records

Assume the attributes of a query record $q$ and a database record $r_i$ have been compared using a set of similarity functions $s_j = sim_j(q.a_j, r_i.a_j)$, $1 \leq j \leq M$, such as approximate string comparison functions [2], with $s_j$ the similarity value calculated for attribute $a_j$, and $M$ the number of compared attributes.

Without taking temporal effects into account, we use $s_j$ as an estimate of the probability that two attribute values are the same, and $(1-s_j)$ as the probability they are different [8] (remember we assume $0 \leq s_j \leq 1$). In a temporal setting, we aim to assign weights to individual attributes that indicate their importance according to the likelihood of a change of value in this attribute, as discussed above. Following Li et al. [8], we use (3) to adjust and normalise similarities.

$$sim(q, r_i) = \frac{\sum_j^M w_j(s_j, \Delta t) \cdot s_j}{\sum_j^M w_j(s_j, \Delta t)}, \tag{3}$$

where $s_j = sim_j(q.a_j, r_i.a_j)$ and $\Delta t = |q.t - r_i.t|$. This equation is a heuristic that attempts to adjust the similarity in a qualitatively reasonable manner. We calculate the weights $w_j(s_j, \Delta t)$ using the minimum similarity threshold $s_{same}$:

- If $s_j \geq s_{same}$ we set $w_j(s_j, \Delta t) = s_j \cdot (1 - P(A_j, \Delta t|\neg S)) = s_j \cdot P(\neg A_j, \Delta t|\neg S)$. This means that the more likely it is that two entities have the same value in attribute $a_j$ over time difference $\Delta t$, the less weight should be given to the similarity value $s_j$ of this agreement.
- If $s_j < s_{same}$ we set $w_j(s_j, \Delta t) = s_j \cdot (1 - P(\neg A_j, \Delta t|S)) = s_j \cdot P(A_j, \Delta t|S)$. This means that the more likely it is that a value in attribute $a_j$ changes over time difference $\Delta t$, the less weight should be given to this disagreement.

For example, as can be seen from Fig. 2, almost nobody in the NC voter database has changed their given name even over long periods of time, compared to changes in address attributes (such as street, suburb and postcode). Therefore, agreeing given name values are a strong indicator in this database that two records refer to the same entity. On the other hand, a low similarity in an address attribute is a weak indicator that two records refer to different entities.

During the matching process a query record is compared with one or more database records, and the resulting adjusted similarities $sim(q, r_i)$ as calculated with (3) are ranked. Details of this matching process are presented in Sect. 4.

### 3.2 Learning Agreement and Disagreement Probabilities

In a dynamic setting, the aim is to efficiently learn the probabilities given in (1) and (2) in an adaptive way. The agreement probability $P(A_j, \Delta t|S)$ can be learned from data if it is known which records correspond to the same entity. To facilitate an efficient calculation of this probability, we discretise the time differences $\Delta t$ into equal sized intervals $\Delta t_k$, of durations such as weeks, months, or years (depending on the overall expected time span in an application). We keep two arrays for each attribute $a_j$, $\mathbf{A}_j[\Delta t_k]$ and $\mathbf{D}_j[\Delta t_k]$. The first array, $\mathbf{A}_j[\Delta t_k]$, keeps track of the number of times a value in attribute $a_j$ is agreeing for two records of the same entity that have a discretised time difference of $\Delta t_k$, while $\mathbf{D}_j[\Delta t_k]$ keeps track of the number of times the values in $a_j$ are disagreeing for two records of the same entity. Using these two counters, we calculate:

$$P(A_j, \Delta t_k|S) = \frac{\mathbf{A}_j[\Delta t_k]}{(\mathbf{A}_j[\Delta t_k] + \mathbf{D}_j[\Delta t_k])} \tag{4}$$

for $\Delta t_1, \ldots, \Delta t_{max}$, with $\Delta t_{max}$ the maximum discretised time difference encountered between two records of the same entity in $\mathbf{R}$. The update of the counters $\mathbf{A}_j[\Delta t_k]$ and $\mathbf{D}_j[\Delta t_k]$, and calculating (4) for a certain discretised $\Delta t_k$, requires a single increase of a counter and one division for each query record that is matched to a database record, making this a very efficient approach.

It is possible that no pair of records of the same entity with a certain discretised time difference of $\Delta t_k$ does occur in a data set, and therefore (4) cannot be calculated for this $\Delta t_k$. To overcome this data sparseness issue, we also calculate the average value for $P(A_j, \Delta t|S)$ over all $\Delta t_k$ for each attribute $a_j$, and use this average value in case $P(A_j, \Delta t_k|S)$ is not available for a certain $\Delta t_k$.

To calculate $P(\neg A_j, \Delta t|\neg S)$, we use the probability that two records that refer to two different entities have the same value $v$ in attribute $a_j$, which equals to $P(A_j, \Delta t|\neg S) = 1 - P(\neg A_j, \Delta t|\neg S)$. This probability depends upon how frequently a certain value $v$ occurs in an attribute. The probability $P_j(v)$ that an entity has the value $v$ in attribute $a_j$ can be estimated from the count of how many records in $\mathbf{R}$ have this value: $P_j(v) = \frac{|r_i \in \mathbf{R}, r_i.a_j = v|}{|\mathbf{R}|}$. For example, only a few records in $\mathbf{R}$ might have the rare surname 'Dijkstra', and so the probability that two records from different entities both have this value is very low.

We keep an array $\mathbf{V}_j$ for each attribute $a_j$ with a counter for each distinct value $v$ of how many records in $\mathbf{R}$ have this value in $a_j$. We then calculate $P(\neg A_j, \Delta t|\neg S) = 1 - P_j(v)$ for all attributes values $v$. If a value in a query record $q$ in attribute $q.a_j$ has not previously occurred in $\mathbf{R}$ (and thus $P_j(q.a_j) = 0$), we set $P(\neg A_j, \Delta t|\neg S) = 1.0$ for this value. Updating the counters $\mathbf{V}_j$ and probabilities $P_j(v)$ requires one integer increment and one division per attribute. For a single query record, the calculations of both agreement and disagreement probabilities are thus of complexity $O(1)$, making this approach very efficient.

The database record to which a query record is matched determines the calculation of $P(A_j, \Delta t_k|S)$. In a supervised setting, the true match status of (a subset of) record pairs is known, allowing the calculation of this probability based on these training pairs. For a query record $q$, if there is a previous record $r_i$ of the

**Algorithm 1.** ***Adaptive Temporal Matching Process***

*Input:*
- Initial database with known entity records: $\mathbf{R} = \mathbf{R}_{init}$
- Arrays with counters for agreements, disagreements, and values: $\mathbf{A}_j$, $\mathbf{D}_j$, and $\mathbf{V}_j$
- Attribute similarity functions: $sim_j(\cdot,\cdot)$; and thresholds: $s_{same}$ and $s_{match}$
- Stream of query records: $q$

*Output:*
- Identifiers of matched entities: $q.e$

1: Set $count_{ent}$ to number of entities in $\mathbf{R}$
2: **while** $q$ **do**:
3: Get candidate records $\mathbf{C}$ from $\mathbf{R}$: $\mathbf{C} = get\_cand\_records(q, \mathbf{R})$
4: **for** $c \in \mathbf{C}$ **do:**
5: $\Delta t = |q.t - c.t|$; $s_j = sim_j(q.a_j, c.a_j), 1 \leq j \leq M$
6: Calculate $sim(q, c)$ using Equation (3)
7: $c_{best} = max(c_i : sim(q, c_i) > sim(q, c_k), c_i \in \mathbf{C}, c_k \in \mathbf{C}, c_i \neq c_k)$
8: **if** $sim(q, c_{best}) \geq s_{match}$ **then** $q.e = c_{best}.e$
9: **else** $q.e = count_{ent}$; $count_{ent} = count_{ent} + 1$
10: Insert $q$ into $\mathbf{R}$: $insert(q, \mathbf{R})$
11: Update $\mathbf{A}_j$, $\mathbf{D}_j$, and $\mathbf{V}_j$, and $P(A_j, \Delta t_k|S)$ and $P_j(v)$, $1 \leq j \leq M$.
12: Pass $q.e$ to application

same entity in $\mathbf{R}$, we calculate $\Delta t = |q.t - r_i.t|$ and which attributes agree or disagree for these two records using $sim_j(q.a_j, r_i.a_j)$ and $s_{same}$, and we update the appropriate counters in $\mathbf{A}_j$ and $\mathbf{D}_j$ and the corresponding probabilities.

If no training data are available, then for a query record the best matching database record, i.e. the record with the highest adjusted similarity according to (3), is assumed to be the true match. We can then calculate the time differences and update the relevant counters and probabilities in the same way as in a supervised setting. Due to limited space, we only consider the supervised case.

## 4 Adaptive Temporal Matching

Algorithm 1 provides an overview of the main steps of our matching process. The required input is a database $\mathbf{R}_{init}$ of known entities. In a supervised setting, where (some of) the true entities are known, the entity identifiers $r_i.e$ in $\mathbf{R}_{init}$ allow us to generate for an entity a chain of records sorted according to the timestamps $r_i.t$. Using these entity chains, the counters $\mathbf{A}_j$ and $\mathbf{D}_j$ are populated, and the initial values for the $P(A_j, \Delta t_k|S)$, $1 \leq j \leq M$ are calculated. In an unsupervised setting, no such initial database is available, and thus $\mathbf{R} = \emptyset$.

The other input parameters required are a set of similarity functions $sim_j(\cdot,\cdot)$, one per attribute $a_j$ used, and the two thresholds $s_{same}$ and $s_{match}$. The former is used to decide if two attribute values are agreeing or not (as described in Sect. 3), while the latter is use to decide if a query record is classified as a match with a database record or not (lines 8 and 9 in Algo. 1).

The algorithm loops as long as query records $q$ are to be matched. We assume the case where query records are inserted into the database once they are matched (line 10). Removing this line will mean that $\mathbf{R}$ and the counters in $\mathbf{A}_j$, $\mathbf{D}_j$ and $\mathbf{V}_j$ are not updated, and therefore our approach becomes non-adaptive (while still taking temporal aspects into account).

In line 3, a set of candidate records $\mathbf{C}$ is retrieved from $\mathbf{R}$ using an index technique for entity resolution. For example, $\mathbf{C}$ might be all records from $\mathbf{R}$ that have

the same postcode value as $q$. In our implementation, we employ a similarity-aware index technique that has shown to be efficient for real-time matching [13]. For each candidate record $c \in \mathbf{C}$, its time difference and similarities with $q$ are calculated in line 5 and adjusted in line 6, as described in Sect. 3.1.

The candidate record with the highest adjusted similarity, $c_{best}$, is identified in line 7 by finding the candidate record that has the maximum similarity with the query record $q$. If this similarity is above the minimum match similarity $s_{match}$, then $q$ is assigned the entity identifier of this best match (line 8). Otherwise $q$ is classified as a new entity and given a new unique entity identifier number in line 9. In line 10, the query record $q$ is inserted into the database $\mathbf{R}$, and in line 11 the counters in $\mathbf{A}_j[\Delta t]$, $\mathbf{D}_j[\Delta t]$, and $\mathbf{V}_j$, as well as the relevant probabilities, are updated for all attributes $a_j$. Finally, the entity identifier of $q$ is passed on to the application that requires this information, allowing the application to extract all records from $\mathbf{R}$ that refer to this entity.

## 5 Experiments and Discussion

We evaluate our adaptive temporal entity resolution approach on both synthetic and real data. Synthetic data allows us to control the characteristics of the data, such as the number of records per entity, and the number of modifications introduced [16]. We generated six data sets, each containing 100,000 records, with an average of 4, 8, or 16 records per entity, and either having a single modification per record (named 'Clean' data) or eight modifications per record (named 'Dirty' data). Modifications introduced were both completely changed attribute values, as well as single character edits (like inserts, deletes and substitutions).

As real data we used the North Carolina (NC) voter registration database [17], which we downloaded every two months since October 2011 to build a compound temporal data set. This data set contains the names, addresses, and ages of more than 2.4 million voters, as well as their unique voter registration numbers. Each record has a time-stamp attached which corresponds to the date a voter originally registered, or when any of their details were changed. This data set therefore contains realistic temporal information about a large number of people. There are 111,354 individuals with two records, 2408 with three, and 39 with four records in this data set. An exploration of this data set has shown that many of the changes in the given name attribute are corrections of nicknames and small typographical mistakes, while changes in surnames and address attributes are mostly genuine changes that occur when people get married or move address.

We sorted all data sets according to their time-stamps, and split each into a training and test set such that around 90% of the second and following records of an entity (of those that had more than one record) were included in the training set. We compare our proposed adaptive temporal matching technique with a static matching approach that does not take temporal information into account, and with a simple temporal matching approach where an additional temporal similarity value is calculated for a record pair as $sim_t(q.t, r_i.t) = \frac{|q.t - r_i.t|}{\Delta T_{max}}$, with $\Delta T_{max}$ being the difference between the time stamps of the earliest and latest

**Table 1.** Matching results for synthetic data sets shown as percentage of true matches correctly identified. The parameter pair given refers to $s_{same}$ / $s_{match}$.

| Parameters | None | | | Temp Attr | | | Adapt | | | Non Adapt | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Avrg rec per ent | 4 | 8 | 16 | 4 | 8 | 16 | 4 | 8 | 16 | 4 | 8 | 16 |
| Clean, 0.8 / 0.7 | 92.3 | 90.5 | 87.2 | 91.9 | 90.6 | 87.4 | 92.3 | 90.8 | 87.3 | 92.3 | 91.1 | 87.3 |
| Clean, 0.9 / 0.8 | 63.8 | 57.9 | 50.7 | 63.2 | 57.9 | 50.7 | 64.2 | 59.1 | 50.9 | 63.9 | 59.6 | 50.9 |
| Dirty, 0.8 / 0.7 | 47.1 | 35.1 | 25.5 | 46.5 | 38.6 | 29.9 | 53.5 | 40.5 | 29.2 | 53.6 | 40.5 | 29.2 |
| Dirty, 0.9 / 0.8 | 16.7 | 10.3 | 5.9 | 14.7 | 10.5 | 6.6 | 21.9 | 13.5 | 7.4 | 21.9 | 13.5 | 7.4 |

record in **R**. Note that we cannot compare our approach to the temporal linkage method proposed by Li et al. [8, 9], because their method is only applicable on static databases, as was described in Sect. 1. We label the non-temporal matching approach with 'None'; the above described temporal attribute approach with 'Temp Attr'; our proposed adaptive approach described in Sect. 4 as 'Adapt'; and with 'Non Adapt' a variation of this adaptive approach that calculates the probabilities $P(A_j, \Delta t_k|S)$ and $P_j(v)$ only once at the end of the training phase, but that does not adjust these probabilities for each of the following query records (i.e. line 11 in Algo. 1 is not executed for query records in the test set).

In Table 1 and Fig. 3 we report matching accuracy as the percentage of true matches correctly identified, i.e. if a candidate record $c_{best}$ in line 7 in Algo. 1 was a record of the same entity as the entity of query record $q$, $c_{best}.e = q.e$. For the NC voter data set we additionally report the percentage of true matches that occurred in the ten top-ranked candidate records. In a real-time query environment, returning the ten top-ranked results is a realistic assumption.

We implemented our approach using Python 2.7.3, and ran experiments on a server with 128 GBytes of memory and two 6-core CPUs running at 2.4 GHz. We used the Jaro-Winkler string comparison function [2] as $sim_j(\cdot, \cdot)$ to compare name and address attribute values. We ran four sets of experiments with parameter settings: $s_{same} = \{0.8, 0.9\}$ and $s_{match} = \{0.7, 0.8\}$. To facilitate repeatability, the programs and synthetic data sets are available from the authors.

The results shown in Table 1 for the synthetic data sets indicate that all four matching approaches are sensitive to the choice of parameter settings, and if the data are clean or dirty. With data that contain a larger number of variations, identifying true matches becomes much more difficult, as would be expected. Matching also becomes harder with more records per entity, because more attributes will have their values changed over time. The proposed approach to adjust the similarities between records is able to improve matching quality most when data are dirty. Such data are likely to occur in dynamic entity resolution applications, where it is not feasible to apply expensive data cleaning operations on query records prior to matching them to a database of entity records.

The experiments conducted on the NC voter database show quite different results (Fig. 3), with the proposed similarity adjustment approach outperforming the two baseline approaches. Taking top ten matches into account, our approach, which adjusts the similarities between records, is able to identify nearly twice as many true matches compared to the two baseline approaches. However, the

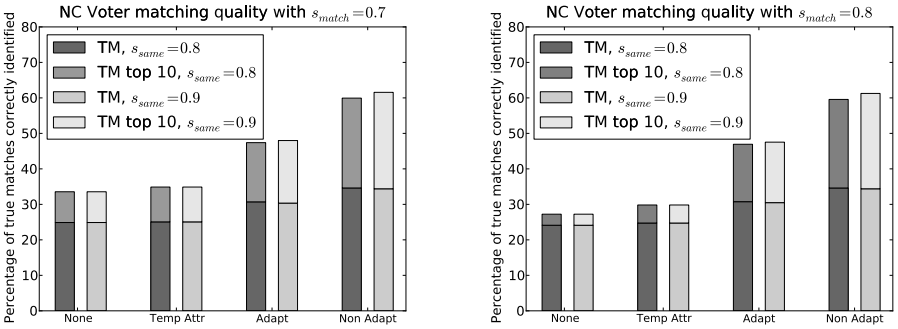


**Fig. 3.** Matching results for the NC voter database shown as percentage of true matches (TM) correctly identified. Different from the results in Table 1, these results are robust with regard to parameter settings, and they are also significantly better for the proposed adaptive approach compared to the baseline approaches 'None' and 'Temp Attr'.

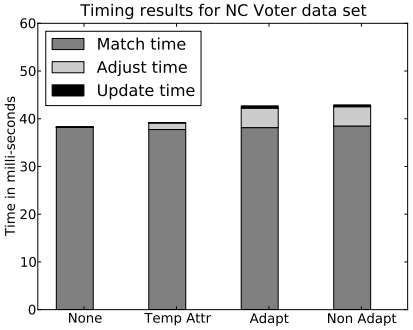


**Fig. 4.** Average time for matching a single query record to the North Carolina voter database. The adjustment of similarities using (3) adds only around 10% extra time.

adaptive approach does not perform as well as the non-adaptive approach. This is likely because the number of true matches compared to all query records is very small, which distorts the calculation of the probabilities in (4).

As Fig. 4 illustrates, our approach is very efficient, even on the large real database with over 2.4 million records. The time needed to adjust the similarity values between records, as done in (3), only adds around 10% to the total time required to match a query record, while the time needed to update the counters $\mathbf{A}_j$, $\mathbf{D}_j$, and $\mathbf{V}_j$, and $P(A_j, \Delta t_k|S)$, is negligible. This makes our approach applicable to adaptive entity resolution on dynamic databases.

## 6 Conclusions and Future Work

We have presented an approach to facilitate efficient adaptive entity resolution on dynamic databases by adaptively calculating temporal agreement and

disagreement probabilities that are used to adjust the similarities between records. As shown through experiments on both synthetic and real data, our temporal matching approach can lead to significantly improved matching quality.

Our plans for future work include to conduct experiments for unsupervised settings as discussed in Sect. 3.2 and run experiments on other data sets, to conduct an in-depth analysis of our proposed algorithm, and to extend our approach to account for attribute and value dependencies. For example, when a person moves, most of their address related attribute values change, and the probability that a person moves also depends upon their age (young people are known to change their address more often than older people). We also plan to integrate our approach with traditional probabilistic record linkage [4], and we will investigate how to incorporate constraints, such as only a few people can live at a certain address at any one time.

## References


1. Winkler, W.E.: Methods for evaluating and creating data quality. Elsevier Information Systems 29(7), 531–550 (2004)
2. Christen, P.: Data Matching. In: Data-Centric Systems and Appl., Springer (2012)
3. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)
4. Herzog, T., Scheuren, F., Winkler, W.: Data quality and record linkage techniques. Springer (2007)
5. Aggarwal, C.: Data Streams: Models and Algorithms. Database Management and Information Retrieval, vol. 31. Springer (2007)
6. Anderson, K., Durbin, E., Salinger, M.: Identity theft. Journal of Economic Perspectives 22(2), 171–192 (2008)
7. Ioannou, E., Nejdl, W., Niederée, C., Velegrakis, Y.: On-the-fly entity-aware query processing in the presence of linkage. VLDB Endowment 3(1) (2010)
8. Li, P., Dong, X., Maurino, A., Srivastava, D.: Linking temporal records. Proceedings of the VLDB Endowment 4(11) (2011)
9. Li, P., Tziviskou, C., Wang, H., Dong, X., Liu, X., Maurino, A., Srivastava, D.: Chronos: Facilitating history discovery by linking temporal records. VLDB Endowment 5(12) (2012)
10. Whang, S., Garcia-Molina, H.: Entity resolution with evolving rules. VLDB Endowment 3(1-2), 1326–1337 (2010)
11. Yakout, M., Elmagarmid, A., Elmeleegy, H., Ouzzani, M., Qi, A.: Behavior based record linkage. VLDB Endowment 3(1-2), 439–448 (2010)
12. Christen, P., Gayler, R.: Towards scalable real-time entity resolution using a similarity-aware inverted index approach. In: AusDM 2008, Glenelg, Australia (2008)
13. Christen, P., Gayler, R., Hawking, D.: Similarity-aware indexing for real-time entity resolution. In: ACM CIKM 2009, Hong Kong, pp. 1565–1568 (2009)
14. Pal, A., Rastogi, V., Machanavajjhala, A., Bohannon, P.: Information integration over time in unreliable and uncertain environments. In: WWW, Lyon (2012)
15. Laxman, S., Sastry, P.: A survey of temporal data mining. Sadhana 31(2) (2006)
16. Christen, P., Pudjijono, A.: Accurate synthetic generation of realistic personal information. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS (LNAI), vol. 5476, pp. 507–514. Springer, Heidelberg (2009)
17. North Carolina State Board of Elections: NC voter registration database, `ftp://www.app.sboe.state.nc.us/` (last accessed September 11, 2012)