Tracking Entities in the Dynamic World: A Fast Algorithm for Matching Temporal Records

Yueh-Hsuan Chiang [1], AnHai Doan [2], Jeffrey F. Naughton [3]
University of Wisconsin Madison
1210 W Dayton Street, Madison, WI 53706, USA
{[1]yhchiang, [2]anhai, [3]naughton}@cs.wisc.edu

ABSTRACT

Identifying records referring to the same real world entity over time enables longitudinal data analysis. However, difficulties arise from the dynamic nature of the world: the entities described by a temporal data set often evolve their states over time. While the state of the art approach to temporal entity matching achieves high accuracy, this approach is computationally expensive and cannot handle large data sets. In this paper, we present an approach that achieves equivalent matching accuracy but takes far less time. Our key insight is "static first, dynamic second." Our approach first runs an evidence-collection pass, grouping records without considering the possibility of entity evolution, as if the world were "static." Then, it merges clusters from the initial grouping by determining whether an entity might evolve from the state described in one cluster to the state described in another cluster. This intuitively reduces a difficult problem, record matching with evolution, to two simpler problems: record matching without evolution, then "evolution detection" among the resulting clusters. Experimental results on several temporal data sets show that our approach provides an order of magnitude improvement in run time over the state-of-the-art approach while producing equivalent matching accuracy.

1. INTRODUCTION

Record matching is a critical and well known problem (see [7, 9] for recent surveys). The vast majority of work in record matching so far has assumed that the records come with no temporal information. In practice, however, we often have temporal information in the form of time stamps associated with records. Examples include author records in DBLP [2], donor records in federal campaign finance data sets [3], and tweets [4]. For this sort of data, it is often desirable to construct an entire time line for each entity by grouping records according to their associated entities over time [14]. For example, grouping all mentions of a company in a Web achieve allows us to understand how the company and its products were doing over the years. Grouping all publication records according to their authors allows us to view the publication history of each author.

In many cases, the entities described by a temporal data set may change or evolve their attribute values over time. For example, people usually change their physical addresses or phone numbers over time. In other cases, people may change their names after getting married or divorced. Organizations may also change their titles or trademarks after restructuring in their identities. For this sort of data, it has been shown that approaches that use temporal information may do better than approaches that do not [12].

Table 1: A list of records from DBLP.

rid	eid	name	affiliation	co-authors	year
r1	e1	Wang Wei	Concordia Univ	Swamy M.N.S., Ahmad M.O., Wang Yuke	1999
r2	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O., Wang Y.	2000
r3	e2	Wang Wei	Fudan University		2001
r4	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O., Wang Y.	2002
r5	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O.	2002
r6	e2	Wang Wei	Fudan University	Q. Yuan, Y. Lou, H. Zhou, B. Shi	2002
r7	e1	Wang Wei	Univ of W Ontario	Swamy M.N.S., Ahmad M.O.	2003
r8	e3	Wang Wei	Zhejiang University	Wei G., Song G., Zheng Y., Luan C., Zhu C.	2004
r9	e1	Wang Wei	Univ of W Ontario	Swamy M.N.S., Ahmad M.O.	2004
r10	e1	Wang Wei	Purdue University	Swamy M.N.S., Ahmad M.O.	2004
r11	e3	Wang Wei	Zhejiang University	Zheng Y., Song G.	2005
r12	e3	Wang Wei	Zhejiang University	Wei G., Wang G., Zheng Y.	2005
r13	e1	Wang Wei	Purdue University	Liu S., Bowen F., King B.	2006
r14	e1	Wang Wei	Purdue University	Liu S., King B.	2007

Table 2: Processing Table 1 by a non-temporal clustering algorithm.

rid	eid	name	affiliation	co-authors	year
r1	e1	Wang Wei	Concordia Univ	Swamy M.N.S., Ahmad M.O., Wang Yuke	1999
r2	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O., Wang Y.	2000
r4	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O., Wang Y.	2002
r5	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O.	2002
r7	e1	Wang Wei	Univ of W Ontario	Swamy M.N.S., Ahmad M.O.	2003
r9	e1	Wang Wei	Univ of W Ontario	Swamy M.N.S., Ahmad M.O.	2004
r10	e1	Wang Wei	Purdue University	Swamy M.N.S., Ahmad M.O.	2004
r3	e2	Wang Wei	Fudan University		2001
r6	e2	Wang Wei	Fudan University	Q. Yuan, Y. Lou, H. Zhou, B. Shi	2002
r8	e3	Wang Wei	Zhejiang University	Wei G., Song G., Zheng Y., Luan C., Zhu C.	2004
r11	e3	Wang Wei	Zhejiang University	Zheng Y., Song G.	2005
r12	e3	Wang Wei	Zhejiang University	Wei G., Wang G., Zheng Y.	2005
r13	e1	Wang Wei	Purdue University	Liu S., Bowen F., King B.	2006
r14	e1	Wang Wei	Purdue University	Liu S., King B.	2007

EXAMPLE 1.1. *Consider a small matching task derived from the DBLP data set shown in Table 1. Here we would like to match records according to their authors. In reality, these records describe three real world authors — e_1, e_2, and e_3 — as indicated by the **eid** field of the records. e_1, e_2 and e_3 share the same value on the name attribute, and they evolve their values on other attributes. Clearly, if we ignore how entities might evolve over time, we are likely to separate records that belong to the same real world entity. For example, Table 2 shows a result obtained by a traditional clustering technique from [11] where records belonging to e_1 are separated into two groups.* □

The typical solution to temporal record matching consists of two major components. The first is a *temporal model* that captures how entities might evolve their attribute values, and that is built using a labeled data set. The second is a *temporal clustering algorithm*, which processes records in a time-dependent order and collects evidence of entity evolution. When determining whether two groups of records refer to the same real world entity, it utilizes both the usual similarity metrics and the temporal model with the collected evidence to make its decision.

The accuracy of a temporal model relies on not only the training data set but, more importantly, the quality of *evolution evidence* collected by the temporal clustering algorithm [12]. Evolution evidence, or evidence for short, is the intermediate clustering result that can be used to infer entity evolution. In the ideal case, each group of records in the intermediate clustering result should reflect the partial history of its associated entity over the time period described by the time stamp associated with the records. In Li's pioneering work [12], the adjusted binding algorithm, a specialized fuzzy clustering algorithm that iteratively accumulates and refines the collected evidence, outperforms existing non-temporal algorithms with respect to matching accuracy.

However, collecting high quality evidence does not come for free. The time complexity of adjusted binding is $O(N^3)$ in the worst case, because N input records are compared with N fuzzy clusters, and each fuzzy cluster contains up to N records. According to their experiments, the algorithm takes more than 10 minutes to process fewer than 2000 records [12]. At this rate, assuming the worst case complexity, it could take up to one week to process 20K records and years to process 200K records. Even if this worst case performance is unlikely, with linear scaling a 2M record data set would take about a week. Thus it is not practical for any but the smallest of data sets.

There are two ways to speed up the clustering process. The first is to use fewer stages of clustering. However, since evidence from later records may fix earlier mistakes [12], naive approaches that use fewer stages of clustering will hurt matching accuracy. The second is to consider fewer records or to use record signatures (that summarize groups of records) in similarity computations. However, since the entities evolve and change states over time, how to effectively construct a signature accurately that describe such an evolving entity is non-trivial.

EXAMPLE 1.2. *Consider again the example task shown in Table 1. This time we use a single-phase temporal clustering algorithm that processes records in increasing time. Then, after we group r_1 and r_2 into the same cluster, we might mistakenly group r_3 together with $\{r_1, r_2\}$ as we consider the possibility that their associated author may change his affiliation within two years. This mistake can be fixed by considering the later evidence described in r_4 and r_5, from which we will know that the associated author is less likely to change his affiliation twice within two years. However, fixing such mistakes typically requires multiple phases of clustering. Table 3 shows the example result obtained by early binding [12].* □

Table 3: Processing Table 1 by a single-phase temporal clustering algorithm.

rid	eid	name	affiliation	co-authors	year
r1	e1	Wang Wei	Concordia Univ	Swamy M.N.S., Ahmad M.O., Wang Yuke	1999
r2	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O., Wang Y.	2000
r3	e2	Wang Wei	Fudan University		2001
r4	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O., Wang Y.	2002
r5	e1	Wang Wei	Concordia University	Swamy M.N.S., Ahmad M.O.	2002
r6	e2	Wang Wei	Fudan University	Q. Yuan, Y. Lou, H. Zhou, B. Shi	2002
r7	e1	Wang Wei	Univ of W Ontario	Swamy M.N.S., Ahmad M.O.	2003
r9	e1	Wang Wei	Univ of W Ontario	Swamy M.N.S., Ahmad M.O.	2004
r10	e1	Wang Wei	Purdue University	Swamy M.N.S., Ahmad M.O.	2004
r13	e1	Wang Wei	Purdue University	Liu S., Bowen F., King B.	2006
r14	e1	Wang Wei	Purdue University	Liu S., King B.	2007
r8	e3	Wang Wei	Zhejiang University	Wei G., Song G., Zheng Y., Luan C., Zhu C.	2004
r11	e3	Wang Wei	Zhejiang University	Zheng Y., Song G.	2005
r12	e3	Wang Wei	Zhejiang University	Wei G., Wang G., Zheng Y.	2005

So a reasonable question is: how can we efficiently cluster temporal records without sacrificing matching accuracy? This paper tries to answer this question and focuses on efficient temporal clustering algorithms. We present SFDS, a two-phase clustering algorithm that achieves the same matching accuracy as the state-of-the-art temporal clustering algorithm but takes far less time. Our key insight is "static first, dynamic second (hence the name SFDS.)" That is, instead of facing the dynamic world directly, SFDS first assumes entities do not evolve, as if the world were "static", and focuses on collecting high quality evidence. In its first phase, it identifies the potential states of each entity by grouping records without considering the possibility of entity evolution. Then, in its second phase, SFDS merges clusters from the initial grouping by determining whether an entity might evolve from the states described in one cluster into the states described in a second cluster.

SFDS improves run time efficiency in two main ways. First, SFDS consists of only two stages of hard clustering: one for collecting evolution evidence and the other for resolving ambiguity caused by entity evolution. This is in contrast to the state-of-the-art approach that consists of multiple stages of fuzzy clustering. Second, SFDS uses signatures to speed up cluster-to-cluster similarity computations, while the state of the art approach considers the entire set of records in similarity computations. SFDS constructs a signature for each cluster formed in the first stage. Intuitively speaking, because this clustering stage does not consider the possibility of evolution, these first-phase clusters correspond to time intervals where entities did not change their states very much. Then, in the second phase, each cluster maintains a set of signatures. When two clusters are merged, the signature set of the resulting cluster is formed by taking the union of the signature sets of the two component clusters. Therefore, the hope is that such a signature set will be able to accurately describe entities that evolve and have multiple states over time.

Now we return to the question: how does SFDS efficiently match temporal records without sacrificing matching accuracy? First, since SFDS first runs an "evidence-collection" pass before handling entity evolution, all evolution inferences are based on evidence collected from all the records. This improves accuracy when determining whether to merge two groups of records. Second, SFDS divides a difficult problem (record matching with evolution) into two simpler problems (record matching without evolution, and "evolution detection" between the resulting clusters).

In the rest of this paper, we give the problem definition and necessary background in Section 2. Then we introduce SFDS in Section 3. Section 4 introduces AFDS, an optimization of SFDS that addresses a "corner case" not well handled by SFDS. Finally, in Section 5, we compare our approach with the state-of-the-art approach on various real world temporal record matching tasks.

2. DEFINITIONS AND BACKGROUND

This section gives the problem definition and briefly reviews the state-of-the-art technique for temporal record matching proposed in [12].

2.1 Problem Definition

Here we give the formal definition of temporal record matching:

DEFINITION 2.1. (TEMPORAL RECORD MATCHING) *Consider a domain $\mathcal{D}$ of entities (not known a priori) and a set* R *of records. Each record $r \in \mathrm{R}$ is of the form $\langle x_1, ..., x_n, t\rangle$, where t is the time stamp of the record r, and each x_i, $1 \leq i \leq n$, is the value of attribute A_i at time t for the associated entity in domain $\mathcal{D}$. The goal of* temporal record matching *is to find a clustering of the records in* R *such that: 1) records in the same cluster refer to the same entity in domain $\mathcal{D}$, and 2) records in different clusters refer to different entities in domain $\mathcal{D}$.* □

As the entities described by a temporal data set may change or evolve their attribute values over time, a temporal matching technique must deal with ambiguity caused by this evolution. One of such ambiguity is *within-entity temporal disagreement*, or *temporal disagreement* for short:

DEFINITION 2.2. (WITHIN-ENTITY TEMPORAL DISAGREEMENT). *Within-entity temporal disagreement arises when two records referring to the same entity appear to refer to two different entities because over time their associated entity evolves.* □

For example, a person may change his or her address and phone number, so two records referring to the same person at different times may disagree on those attributes.

A second kind of ambiguity is *between-entity temporal agreement*, or *temporal agreement* for short:

DEFINITION 2.3. (BETWEEN-ENTITY TEMPORAL AGREEMENT). *Between-entity temporal agreement arises when two records referring to two different entities appear to refer to the same entity because over time one of the entities evolved to have the same value in some attribute as that previously held by the other.* □

Returning to our example, it is possible that one person might get the phone number of a second person when the first person takes over the second person's office.

We say that a function is a *disagreement model* or an *evolution model*, denoted as f_d, if it computes the probability that temporal disagreement applies to a given pair of records or clusters. Likewise, we say a function is an *agreement model*, denoted as f_a, if it computes the probability that temporal agreement applies to a given pair of records or clusters.

Sometimes we say an entity evolves from one *state* to another. We are not proposing a formal definition of the "state of an entity." Rather, we use the term informally to refer to periods of time in which the values in the fields of records referring to the entity tend to be very similar. Intuitively, this corresponds to times in between changes in the "life" of an entity. For example, a "state" of a researcher could be the period of time during which she taught at a particular university. Note that for our algorithm to work we do not need an externally valid definition of state — indeed, for us, the "states of an entity" are defined empirically by the clusters that our algorithm finds during its first phase (described in Section 3.2). We tie this to the external notion of state only to provide an intuition for why there will likely be periods of time during which the records referring to an entity tend to be more similar.

2.2 Existing Temporal Matching Techniques

The typical solution to temporal record matching consists of two major components: a temporal model and a temporal clustering algorithm.

A temporal model usually consists of two components: one for handling temporal disagreement, and the other for handling temporal agreement. For example, in Li *et al.*'s pioneering work [12], they proposed the following two components:

DEFINITION 2.4. (DISAGREEMENT DECAY). *Let Δt be a time distance and $A \in \mathbf{A}$ be a single-valued attribute. Disagreement decay of A over time Δt is the probability that an entity changes its A-value within time Δt.* □

DEFINITION 2.5. (AGREEMENT DECAY). *Let Δt be a time distance and $A \in \mathbf{A}$ be an attribute. The agreement decay of A over time Δt is the probability that two different entities share the same A-value within time Δt.* □

As shown in [12] and illustrated in Example 1.2, simply using such a temporal model in the similarity computation step of existing record matching algorithms may not produce an acceptable matching result. In view of this, Li *et al.* proposed three temporal record matching algorithms that apply a temporal model in similarity computations and consider records in increasing temporal order. Specifically, *early binding* performs a single-phase hard-clustering that makes eager decisions and merges a record with an already created cluster if the similarity between the record and the cluster is sufficiently high; *late binding* performs instead a single-phase fuzzy-clustering that keeps all evidence and makes decisions at the end; and *adjusted binding* runs multiple phases of fuzzy clustering that in addition compares records with clusters that are created for records with later time stamps and iteratively refines the clustering result. Not surprisingly, adjusted binding outperforms all existing approaches in matching accuracy but is computationally expensive.

In contrast, as we will see in the next section, our proposed SFDS approach first assumes entities do not evolve and accumulates evidence by grouping records based on pure attribute value similarity without using a temporal model. Then, SFDS further merges possible clusters from the initial grouping by determining whether it is possible for one entity to evolve from what is described in one cluster to what is described in the second cluster.

In the rest of this paper, we will explain SFDS in more detail and explore its performance.

3. THE SFDS APPROACH

We now describe SFDS in detail. SFDS operates in two phases: *static* and *dynamic* (Figure 1). In the *static phase* (or *S-phase* for short), the goal is to collect evidence for later inferring entity evolution in the dynamic phase. This is done by forming an initial grouping such that records in the same group describe part of its entity's history where the entity does not evolve too much. SFDS achieves this goal by not using any temporal model in similarity computations. During the S-phase, each cluster maintains a temporal signature that consists of its earliest and latest records (i.e., the two with the earliest and the latest time stamps). As each cluster in the S-phase tends to describe portion of an entity's history where the entity has not evolved, such a temporal signature will likely capture the starting and ending points of that partial history of its entity. The output of the S-phase is an initial grouping of the input records and a set of associated temporal signatures.

In the *dynamic phase* (or *D-phase* for short), SFDS further merges possible resulting clusters from the output of the S-phase. It uses a temporal model to handle entity evolution. For each pair of resulting clusters, SFDS makes the merge decision by determining whether it is possible for an entity to evolve from the state described in one cluster to the state described in another cluster. In the D-phase, each cluster maintains a *signature set* that is built up from the temporal signatures created in the S-phase. When two clusters are merged, the signature set of the resulting cluster will be formed by taking the union of the signature sets of the two component clusters. By having multiple temporal signatures describing an entity,

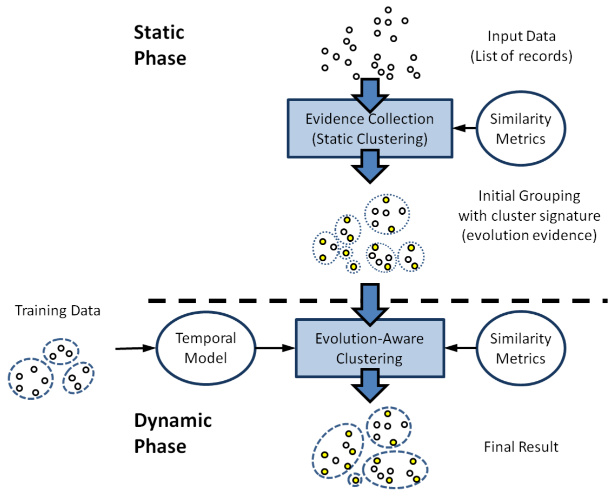
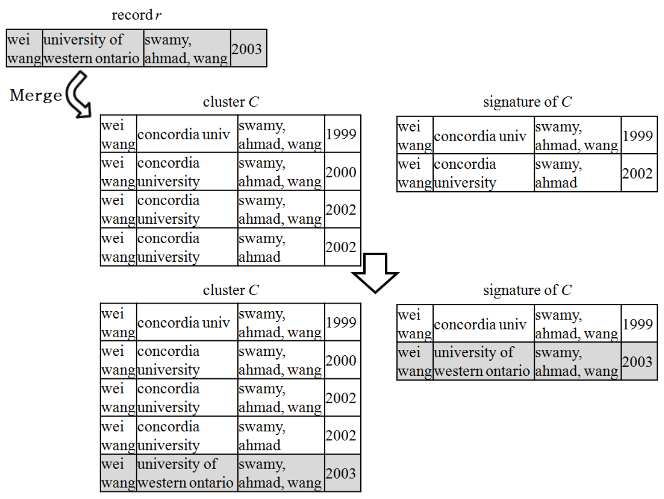

Static Phase
Input Data (List of records)
Evidence Collection (Static Clustering)
Similarity Metrics
Initial Grouping with cluster signature (evolution evidence)
Training Data
Temporal Model
Evolution-Aware Clustering
Similarity Metrics
Dynamic Phase
Final Result

Figure 1: The SFDS approach


record r
Merge
cluster C
signature of C
cluster C
signature of C

wei wang	university of western ontario	swamy, ahmad, wang	2003

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad, wang	2000
wei wang	concordia university	swamy, ahmad, wang	2002
wei wang	concordia university	swamy, ahmad	2002

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad	2002

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad, wang	2000
wei wang	concordia university	swamy, ahmad, wang	2002
wei wang	concordia university	swamy, ahmad	2002
wei wang	university of western ontario	swamy, ahmad, wang	2003

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	university of western ontario	swamy, ahmad, wang	2003

Figure 2: An example of signature update in the S-phase.

where each signature describes a partial history of the entity, SFDS is able to improve efficiency without sacrificing accuracy.

This section first reviews the process of learning temporal models. Then, it describes the two phases of SFDS. The next section (Section 4) describes AFDS — an optimized version of SFDS — which handles a special case not optimally handled by SFDS.

3.1 Preliminary: Learning Temporal Models

A temporal matching algorithm must use a temporal model to evaluate the probabilities that record attribute values might or might not evolve in specific ways. As the choice of temporal model is orthogonal to the clustering algorithm that employs the model, in this work we simply choose an existing temporal model, and we describe how it works in this section. Discovering better temporal models is an interesting problem outside the scope of this current paper.

We follow the approach in [12] to learn the temporal agreement and disagreement models, using the statistics of attribute life spans given a training data set. Consider an entity E and records $r_1, ..., r_n$ that refer to E and are arranged in increasing order of time stamp. We call a point in time τ a *change point* of an A-value in E if there exists some i, $1 < i \leq n$, such that $r_i.t = \tau$ and $r_{i-1}.A \neq r_i.A$. The intervals between consecutive change points are called *full life spans* of A-values. Before the first change point of E and after the last change point of E, we are unable to observe the exact life span of its A-value. We only know that E does not change its A-value within these intervals. These intervals are called *partial life spans*. From the training data set, we only learn the observed time interval for each partial life span indicating the lower-bound of a potential full life span.

Let $\bar{L}_f^A$ denote the bag of lengths of full life spans on A-value of a given data set and $\bar{L}_p^A$ denote the bag of lengths of observed partial life spans respectively. The temporal disagreement function f_d is formed by considering all the full life spans and the partial life spans with at least length Δt:

$$f_d(A, \Delta t) = \frac{|\{l \in \bar{L}_f^A | l \leq \Delta t\}|}{|\bar{L}_f^A| + |\{l \in \bar{L}_p^A | l \geq \Delta t\}|} \tag{1}$$

The temporal agreement function f_a is computed based on the bag of *span distance* $|\bar{L}^A|$. If two entities ever share the same value on attribute A, then their span distance on attribute A is defined as the time interval between the time periods when the two entities share the same value on attribute A. Otherwise, the span distance is defined as ∞.

$$f_a(A, \Delta t) = \frac{|\{l \in \bar{L}^A | l \leq \Delta t\}|}{|\bar{L}^A|} \tag{2}$$

The bags of full life spans, partial life spans, and span distances can be obtained by iterating through all the records in the given data set in increasing temporal order. Thus the temporal agreement and disagreement functions can be learned based on the above equations.

3.2 Static Phase: Evidence Collection

We now return to our proposed temporal clustering approach SFDS; specifically, the static phase. Given a set of records from the original input, the S-phase considers records in increasing temporal order based on their time stamps. For each record, the S-phase either merges the record with an already created cluster if it finds one that is sufficiently similar, or creates a new cluster otherwise. When determining whether two clusters of records should be merged, the S-phase computes the "static similarity" between the two clusters using any standard distance metric for record matching.

Specifically, consider record r and a set $\mathcal{C} = \{C_1, ..., C_n\}$ of clusters created earlier in the S-phase, where $n \in \mathbb{N}$ is the number of already created clusters. The S-phase proceeds in the following steps:

1. Compute the static similarity sim_s between r and each C_i, $1 \leq i \leq n$ without using any temporal model (subscript s stands for static).

2. Let $C_{max} \in \mathcal{C}$ be the cluster with the highest similarity to r. If $sim_s(r, C_{max}) < \theta_S$, where θ_S is a learned threshold indicating high similarity, then create a new cluster $C_{n+1} = \{\}$, append C_{n+1} to $\mathcal{C}$, and let $C_{max} = C_{n+1}$.

3. Append r to the end of the chosen cluster C_{max}. As the S-phase considers records in time order, simply appending r to the end of the cluster C_{max} will keep records in C_{max} sorted in increasing temporal order.

4. Update the temporal signature $s_{C_{max}}$ of cluster C_{max} for r. The temporal signature consists of the earliest and the latest records of its associated cluster, so this step can be done by simply replacing its latest record with r (Figure 2).

Note that since the S-phase processes records in increasing temporal order, the resulting set of clusters has the following properties: 1) records in each cluster are sorted by their time stamps in increasing order; 2) clusters in $\mathcal{C}$ are sorted by their starting time stamps (the earliest time stamp of its records) in increasing order.

We now describe the temporal signature and the similarity computation part in Step 1 and the threshold learning part in Step 2.

Temporal signature: Let $C = \{r_1, r_2, ...r_n\}$ be a cluster containing n records. The cluster signature s_C of C consists of the two member records with the earliest and the latest time stamps:

$$s_C = \{r_f, r_l\} \tag{3}$$

$$r_f = \arg\min_{r \in C} r.t \tag{4}$$

$$r_l = \arg\max_{r \in C} r.t \tag{5}$$

Since the S-phase considers records in increasing temporal order, the update of signature can be done by simply replacing r_l by the incoming record.

Similarity computation: the S-phase computes the "static similarity" between a record and a cluster without using a temporal model.

Specifically, consider a record r and an already created cluster C, the "static similarity" sim_s between r and C is computed as the average record similarities between r and the records in C's signature:

$$sim_s(r, C) = \frac{\sum_{r' \in s_C} sim_s(r, r')}{|s_C|} \tag{6}$$

where the static similarity $sim_s(r, r')$ is defined as the average of their attribute value similarities:

$$sim_s(r, r') = \frac{\sum_{A \in \mathbf{A}} sim_A(r.A, r'.A)}{|\mathbf{A}|} \tag{7}$$

where sim_A is the similarity metric for attribute A and $\mathbf{A}$ denotes the set of all attributes.

Threshold learning: We use the following heuristic with a customizable parameter min_R indicating a minimum recall to learn the threshold θ_S for the S-phase. If the minimum recall can be reached, then θ_S is set to the threshold that leads to the highest precision. Otherwise, it is set to the one that leads to the highest recall.

Specifically, given a set of training data with known ground truth and a specified minimum recall min_R, we first figure out whether there exists a threshold that can lead to a recall higher than the specified value min_R. If the answer is no, then θ_S is set to the threshold that yields the highest recall. Otherwise, θ_S is set to the threshold that yields the highest precision with its recall higher than min_R. Based on our experiments, a 0.2 to 0.3 minimum recall is suggested. Note that the final result quality does not heavily rely on the specified min_R. We report detailed results on how min_R might affect the final matching accuracy in Section 5.4.4.

PROPOSITION 3.1. *The S-phase runs in time $O(|R| \cdot |C|)$, where $|R|$ is the number of input records and $|C|$ is the number of clusters created in the S-phase.* □

For each input record, the S-phase compares it with every already-created cluster. Since the S-phase assumes no entity evolution and uses only the latest record of a cluster in record-to-cluster similarity computations, each record-to-cluster comparison runs in $O(1)$, and there are $O(|R| \cdot |C|)$ of such comparisons.

Algorithm 1 S-PHASE(R, θ_S)

Input:
R, a list of records sorted in increasing temporal order.
θ_S, a threshold to make merge decision.
Output:
$\mathcal{C}$, the clustering of records in R
$\mathbf{S}_{\mathcal{C}}$, the set of cluster signatures corresponding to $\mathcal{C}$, each cluster in $\mathcal{C}$ has a signature in $\mathbf{S}_{\mathcal{C}}$.

```
1:  $\mathcal{C} = \{\}$
2:  for all $r \in R$ do
3:      for all $C \in \mathcal{C}$ do
4:          compute $sim_s(r, C)$              // static similarity (Eq.12, 6)
5:      end for
6:      // if true, then merge to the best possible cluster
7:      if $\max_{C \in \mathcal{C}} sim_s(r, C) \geq \theta_S$ then
8:          $C_{max} = \arg\max_{C \in \mathcal{C}} sim_s(r, C)$
9:          $C_{max} \leftarrow C_{max} \cup r$                // append $r$ to $C_{max}$
10:         $r_l \leftarrow \arg\max_{r \in C_{max}} r.t$        // $O(1)$ since sorted by time
11:         $s_{C_{max}} \leftarrow s_{C_{max}} \setminus r_l$         // remove the last record
12:         $s_{C_{max}} \leftarrow s_{C_{max}} \cup r$        // append $r$ as the last record
13:     else
14:         let $C_{max} = \{r\}$                  // create a new cluster
15:         let $s_{C_{max}} = \{r\}$     // initialize the signature for $C_{max}$
16:         $\mathcal{C} \leftarrow \mathcal{C} \cup C_{max}$
17:         $\mathbf{S}_{\mathcal{C}} \leftarrow \mathbf{S}_{\mathcal{C}} \cup s_{C_{max}}$
18:     end if
19: end for
20: return $\mathcal{C}, \mathbf{S}_{\mathcal{C}}$
```

Note that in terms of time complexity, the S-phase never performs slower than an ordinary clustering approach, which typically compares all pairs of records and runs in $O(|R|^2)$. In the worst case where records never match with any already-created cluster, $|C|$ will degrade to $|R|$. In this situation, the run time performance of the S-phase will not take advantage of cluster signatures and degrades to $O(|R|^2)$.

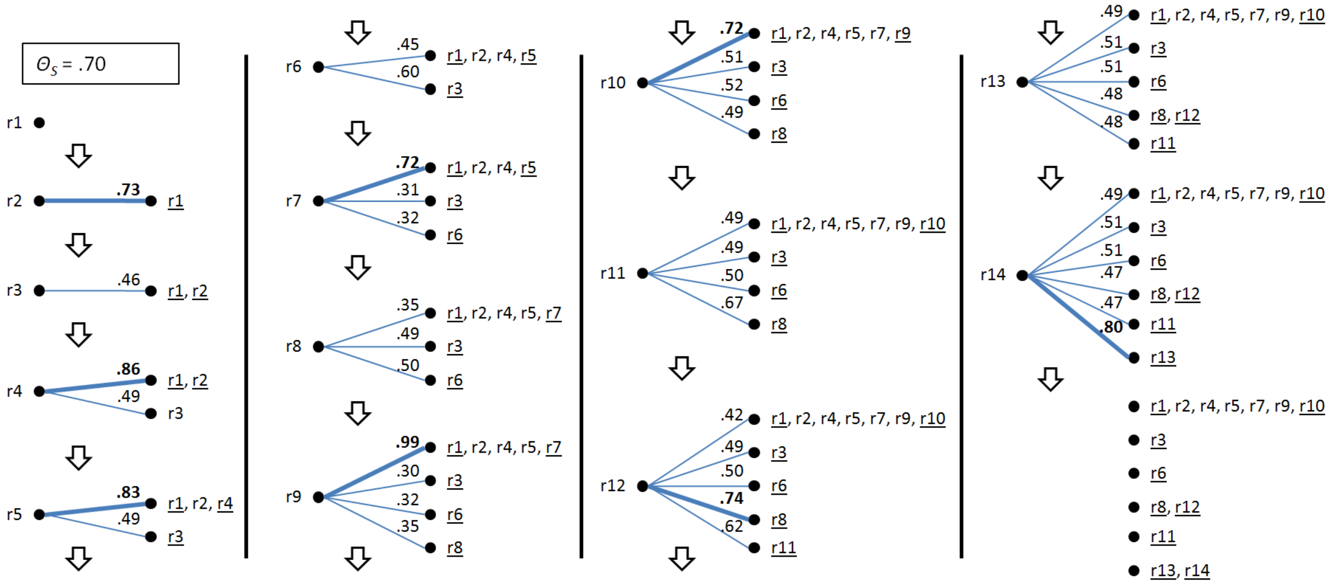
We give the algorithm of the S-phase in Algorithm 1.

EXAMPLE 3.1. *Figure 3 shows a step-by-step example of how the S-phase forms an initial grouping from the example task shown in Example 1.1. Records that are part of the signatures of their clusters are underscored. In the example, we can see that r_1, r_2, r_4, r_5, r_7, r_9 and r_{10}, which are associated with entity e_1, are merged together due to their high similarity on the affiliation and co-author attributes. However, r_{13} and r_{14}, which are also associated with entity e_1, are grouped into another cluster.* □

3.3 Dynamic Phase: Evolution Detection

The D-phase further merges some of the clusters generated by the S-phase. In the input of the D-phase, each cluster maintains exactly one signature, which describes one state of its associated entity. In addition, we may have multiple clusters associated with the same entity. Unlike the S-phase, the D-phase 1) takes a list of clusters as input instead of a list of records, 2) computes "dynamic similarity", which further uses temporal agreement and disagreement models, and 3) allows each cluster to have multiple temporal signatures.

Specifically, given a set $\mathcal{C} = \{C_1, ..., C_m\}$ of clusters obtained from the S-phase, where $m \in \mathbb{N}$ denotes the number of clusters, and clusters in $\mathcal{C}$ are sorted by their starting time stamps in increasing temporal order. Consider a cluster $C \in \mathcal{C}$ and a set



θ_S = .70
r1
r2 .73 r1
r3 .46 r1, r2
r4 .86 r1, r2 .49 r3
r5 .83 r1, r2, r4 .49 r3
r6 .45 r1, r2, r4, r5 .60 r3
r7 .72 r1, r2, r4, r5 .31 r3 .32 r6
r8 .35 r1, r2, r4, r5, r7 .49 r3 .50 r6
r9 .99 r1, r2, r4, r5, r7 .30 r3 .32 r6 .35 r8
r10 .72 r1, r2, r4, r5, r7, r9 .51 r3 .52 r6 .49 r8
r11 .49 r1, r2, r4, r5, r7, r9, r10 .49 r3 .50 r6 .67 r8
r12 .42 r1, r2, r4, r5, r7, r9, r10 .49 r3 .50 r6 .74 r8 .62 r11
r13 .49 r1, r2, r4, r5, r7, r9, r10 .51 r3 .51 r6 .48 r8, r12 .48 r11
r14 .49 r1, r2, r4, r5, r7, r9, r10 .51 r3 .51 r6 .47 r8, r12 .47 r11 .80 r13
r1, r2, r4, r5, r7, r9, r10
r3
r6
r8, r12
r11
r13, r14

Figure 3: A step-by-step example of applying the S-phase on the matching task shown in Table 1.

$\mathcal{D} = \{D_1, ..., D_n\}$ of clusters created earlier in the D-phase. The D-phase proceeds in the following steps:

1. Compute the dynamic similarity sim_d between cluster C and each already-created cluster $D_i, 1 \leq i \leq n$ using both temporal agreement and disagreement models.
2. Let $D_{max} \in \mathcal{D}$ be the cluster with the highest similarity to C. If their similarity $sim_d(C, D_{max})$ is less than the learned threshold θ_E, then create a new cluster $D_{n+1} = \{\}$, append D_{n+1} to the set $\mathcal{D}$ of the already created clusters in the D-phase, and let $D_{max} = D_{n+1}$.
3. Merge C with D_{max}. The merge process is done by simply appending all records in C to cluster D_{max} as their ordering will not be used in the rest of the clustering process.
4. Update the signature set $S_{D_{max}}$ of D_{max} by taking the union of the cluster signature sets of C and D_{max} (Figure 4).

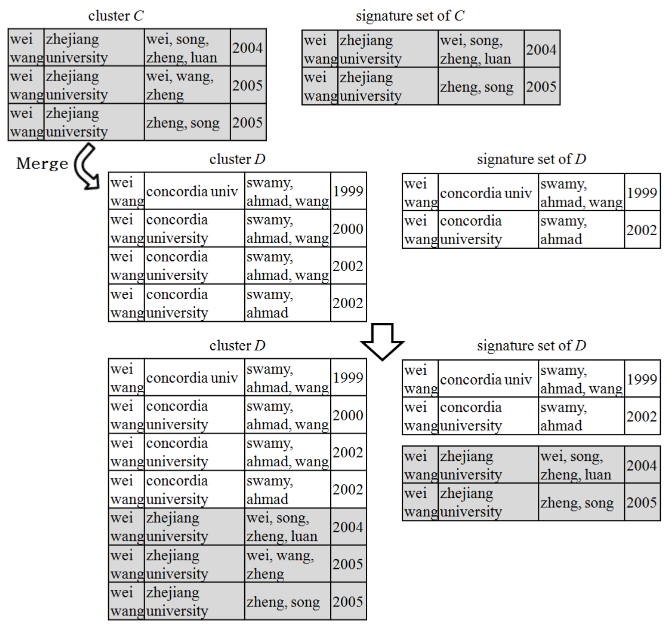
We now turn to the details of the signature update, similarity computation, and the threshold learning processes in the D-phase.
Signature update: In the D-phase, the signature set S_C of a cluster C consists of a set of temporal signatures created in the S-phase (Eq. 3) where each component signature describes one state of its referred entity:

$$S_C = \{s_1, s_2, ..., s_m\} \tag{8}$$

When merging two clusters C and D, the D-phase updates the signature set S of the resulting cluster by taking the union of the signature sets S_C and S_D from the two component clusters C and D:

$$S = S_C \cup S_D \tag{9}$$

Similarity computation: Given two clusters C and D and their signature sets S_C and S_D, the D-phase computes their "dynamic similarity" by taking the average of the filtered dynamic similarities of the record pairs chosen from S_C and S_D, where only those record pairs carrying evidence that supports at least one of the following statements are considered:



cluster *C*

wei wang	zhejiang university	wei, song, zheng, luan	2004
wei wang	zhejiang university	wei, wang, zheng	2005
wei wang	zhejiang university	zheng, song	2005

signature set of *C*

wei wang	zhejiang university	wei, song, zheng, luan	2004
wei wang	zhejiang university	zheng, song	2005

cluster *D*

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad, wang	2000
wei wang	concordia university	swamy, ahmad, wang	2002
wei wang	concordia university	swamy, ahmad	2002

signature set of *D*

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad	2002

cluster *D*

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad, wang	2000
wei wang	concordia university	swamy, ahmad, wang	2002
wei wang	concordia university	swamy, ahmad	2002
wei wang	zhejiang university	wei, song, zheng, luan	2004
wei wang	zhejiang university	wei, wang, zheng	2005
wei wang	zhejiang university	zheng, song	2005

signature set of *D*

wei wang	concordia univ	swamy, ahmad, wang	1999
wei wang	concordia university	swamy, ahmad	2002
wei wang	zhejiang university	wei, song, zheng, luan	2004
wei wang	zhejiang university	zheng, song	2005

Figure 4: An example of signature update in the D-phase.

- both clusters C and D describe at least one state of the same entity in common, or
- at least one evolution of the same entity is described in the union of the two clusters C and D but in neither of the two clusters alone.

Specifically, given records r_1 and r_2, their dynamic similarity is defined as the weighted average of their attribute value similarities where the weight is determined by the complement of the level of temporal agreement and disagreement estimated by the temporal model. The intuition is: the higher the level of temporal agreement or disagreement on one attribute is, the less we are able to trust the

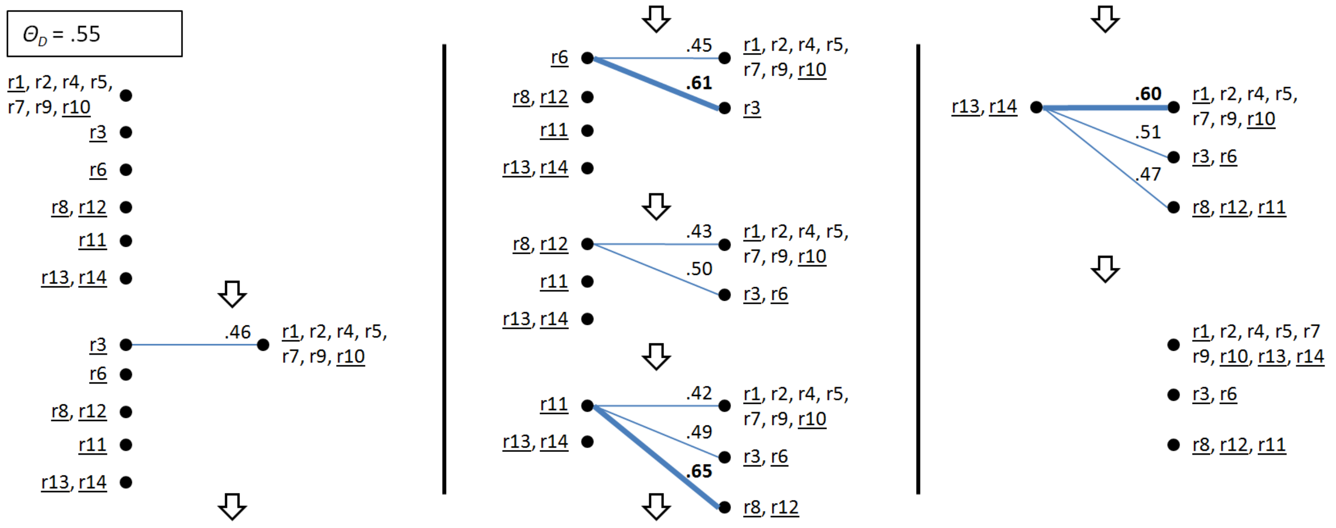


Figure 5: A step-by-step example of the D-phase, continued from Figure 3 and Example 3.1.

similarity on that attribute:

$$sim_d(r_1, r_2) = \frac{\sum_{A \in \mathbb{A}} w_d(A, sim_A(r_1.A, r_2.A), |r_1.t - r_2.t|) \cdot sim_A(r_1.A, r_2.A)}{\sum_{A \in \mathbb{A}} w_d(A, sim_A(r_1.A, r_2.A), |r_1.t - r_2.t|)} \tag{10}$$

$$w_d(A, s, \Delta t) = \begin{cases} 1 - f_a(A, \Delta t) & \text{when } s \geq \theta_A \\ 1 - f_d(A, \Delta t) & \text{otherwise} \end{cases} \tag{11}$$

where f_d and f_a are temporal disagreement and agreement functions respectively (Eq. 1 and Eq. 2). $A \in \mathbb{A}$ again denotes an attribute, sim_A is the similarity metric for attribute A, w_d is a weighting function to determine the importance of an attribute, and θ_A is a threshold for determining whether two records describe the same state on attribute A. θ_A can be learned from a labeled data set.

Finally, the D-phase computes the dynamic similarity sim_d between clusters C and D by taking the average of the filtered dynamic similarities of the record pairs chosen from their cluster signatures S_C and S_D that are above the learned threshold θ_D:

$$sim_d(C, D) = \frac{\sum_{\langle r_1, r_2 \rangle \in \mathbf{R}_{C,D,\theta_D}} sim_d(r_1, r_2)}{|\mathbf{R}_{C,D,\theta_D}|} \tag{12}$$

$$\mathbf{R}_{C,D,\theta_D} = \left\{ \langle r_1, r_2 \rangle \middle| \begin{array}{l} \exists s_i \in S_C \text{ s.t. } r_1 \in s_i \wedge \\ \exists s_j \in S_D \text{ s.t. } r_2 \in s_j \wedge \\ sim_d(r_1, r_2) \geq \theta_D \end{array} \right\} \tag{13}$$

where R_{C,D,θ_D} is a set of record pairs chosen from S_C S_D which dynamic similarity is higher than the learned threshold $theta_D$ indicating it supports at least one of the required conditions mentioned above.

Threshold learning: As the D-phase is the final step of SFDS, given a labeled data set, θ_E is set to the threshold that leads the highest F-1 score (defined in Section 5.1.4) in the labeled data set.

PROPOSITION 3.2. *The D-phase runs in time $O(|C|^2)$ where $|C|$ is the number of clusters created the S-phase.* □

Since the D-phase uses temporal signatures, comparing a input cluster with all the already created clusters is equivalent to comparing a input clusters with all the already created temporal signatures. In addition, the total number of signatures is equal to the total number of clusters created in the S-phase. As a result, each cluster-cluster similarity computation in the D-phase runs in $O(|C|)$, and there are $|C|$ of such computation. This gives time complexity $O(|C|^2)$.

PROPOSITION 3.3. *SFDS runs in time $O(|C| \cdot |R|)$, where $|C|$ is the number of clusters created in the S-phase and $|R|$ is the number of input records.*

SFDS performs $|C| \cdot |R| + |C|^2$ comparisons in average. In the worst case, it runs in time $O(|R|^2)$ and performs $2 \cdot |R|^2$ comparisons.

We give the algorithm for the D-phase in Algorithm 2.

EXAMPLE 3.2. *We continue with our running example described in Example 1.1, where we have generated the initial grouping in Example 3.1 shown in Figure 3. Figure 5 shows the step-by-step result of applying the D-phase to further merge clusters from the initial grouping. In the step-by-step result, we can see that the cluster $\{r_{13}, r_{14}\}$, which describes entity e_1, is now correctly merged into the cluster $\{r_1, r_2, ..., r_{10}\}$, which also describes e_1, with a dynamic similarity 0.60 higher than the learned threshold $\theta_D = 0.55$. Note that we are unable to correctly merge r_{13} in the S-phase even if we lower the threshold θ_S as its best match cluster is $\{r_3\}$, which describes entity e_2 instead. With the initial grouping created in the S-phase based on the states of the entities, we are able to reduce incorrect merges in the D-phase.* □

4. THE AFDS — AN OPTIMIZED VERSION OF SFDS

As we have noted, the first phase of SFDS ignores evolution and considers neither between-entity temporal agreement nor within-entity temporal disagreement. Ignoring within-entity temporal disagreement is in a sense "safe" because the errors it will make (splitting one entity into multiple clusters) have a chance of being rectified later (when the dynamic phase merges these clusters). However, ignoring between-entity temporal agreement is "dangerous" because this might include the records from two entities in a single cluster (because only between-entity temporal agreement provides the insight that similar attribute values in two records separated by time might not mean the records refer to the same entity). This is

Algorithm 2 E-PHASE($\mathcal{C}, \mathbf{S}_{\mathcal{C}}, \theta_E$)

```
Input:
$\mathcal{C}$, the clustering result of the S-phase.
$\mathbf{S}_{\mathcal{C}}$, the set of cluster signatures associated with $\mathcal{C}$.
$\theta_E$, a threshold to determine whether to merge two records.
Output:
$\mathcal{D}$, the final clustering result
 1: $\mathcal{D} = \{\}$
 2: for all $C \in \mathcal{C}$ do
 3:     $S_C = \{s_C\}$          // pack its signature into a signature set
 4:     for all $D \in \mathcal{D}$ do
 5:         compute $sim_d(C, D)$   // dynamic similarity (Eq.12, 10)
 6:     end for
 7:     if $\max_{D \in \mathcal{D}} sim_d(C, D) \geq \theta_E$ then
 8:         $D_{max} \leftarrow \arg\max_{D \in \mathcal{D}} sim_d(C, D)$
 9:         $D_{max} \leftarrow C \cup D_{max}$                // merge to the best
10:         $\mathbf{S}_{D_{max}} \leftarrow \mathbf{S}_C \cup \mathbf{S}_{D_{max}}$        // update signature
11:     else
12:         $\mathcal{D} \leftarrow \mathcal{D} \cup C$                  // keep the original cluster
13:     end if
14: end for
15: return $\mathcal{D}$
```

dangerous because at the end of the "S" phase, the clusters are replaced by signatures, so there is no chance of undoing such errors later.

Our experiments in Section 5.4.5 show that this is a small but real danger — typically it lowers F1 score by a few percentage points. Our "fix" is to add temporal agreement modeling to the clustering in the "S" phase. We call this optimization of the SDFS approach AFDS — agreement first, dynamic second.

In the rest of this section, we describe the only difference between AFDS and SFDS: the similarity computation in the first phase (the S-phase).

4.1 Similarity Computation

As the first phase of AFDS considers the possibility of between-entity temporal agreement, the "static similarity" between two records is defined as the weighted average of attribute value similarities, where the weight of each attribute is a function of temporal agreement.

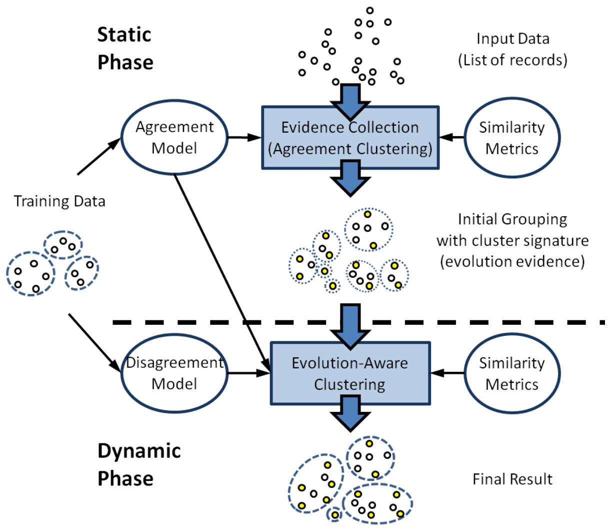
Consider records r_1 and r_2 for which we would like to compute static similarity. When r_1 and r_2 have a high similarity on attribute A, this indicates that the two records might either describe the same state of the same entity or describe similar states of different entities. In this case, the weight of their similarity on attribute A is determined by the complement of the level of between-entity temporal agreement estimated by a temporal agreement function f_a. Otherwise, a full weight will be assigned as it is less possible for r_1 and r_2 to describe the same state of the same entity given their low attribute value similarity on A.

To put it all together, AFDS uses the following equations to compute the static similarity between two temporal records:

$$sim_s(r_1, r_2) = \frac{\sum_{A \in \mathbb{A}} w_s(A, sim_A(r.A, r'.A), |r.t - r'.t|) \cdot sim_A(r.A, r'.A)}{\sum_{A \in \mathbb{A}} w_s(A, sim_A(r.A, r'.A), |r.t - r'.t|)} \tag{14}$$

$$w_s(A, s, \Delta t) = \begin{cases} 1 - f_a(A, \Delta t) & \text{when } s \geq \theta_A \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

where the subscripts s and a stand for static and temporal agreement respectively, $A \in \mathbb{A}$ is an attribute, sim_A is the similarity



Static Phase
Input Data (List of records)
Agreement Model
Evidence Collection (Agreement Clustering)
Similarity Metrics
Training Data
Initial Grouping with cluster signature (evolution evidence)
Disagreement Model
Evolution-Aware Clustering
Similarity Metrics
Dynamic Phase
Final Result

Figure 6: The AFDS approach — an optimized version of SFDS

metric for attribute A, w_s is the weighting function to determine the importance of an attribute, and θ_A is a threshold for determining whether it is possible for the two records to describe the same state on attribute A. θ_A can be learned from a labeled data set.

As for the static similarity between a record and a cluster, AFDS simply replaces the static record similarity $sim_s(r, r')$ in Equation 6 by Equation 14.

The reader may wonder at this point if considering disagreement modeling during this first phase might improve things still further. The answer is "no" — while it might improve the clustering produced during the first phase, as we will see in our experiments, it actually harms the final clustering after the second phase (merging clusters based on their signatures). Again, this is because if disagreement modeling makes a mistake that merges two entities into one cluster during the first phase, there is no opportunity to remedy this mistake in the second phase, since the clusters of records will have been replaced by signatures and cannot subsequently be "re-split." In addition, without having an evidence collection phase before resolving the temporal within-entity disagreement caused by entity evolution, we are more likely to make early mistakes that could be fixed by the evidence described in the "later" records, as illustrated earlier in Example 1.1.

PROPOSITION 4.1. *AFDS runs in time $O(|C| \cdot |R|)$, where $|C|$ is the number of clusters created in its first phase and $|R|$ is the number of input records.*

5. EXPERIMENTAL EVALUATION

This section describes experimental results on four temporal matching tasks from two real world data sets: a European patent data set [1] and three subsets of the DBLP data set [2]. Table 4 summarizes the matching tasks used in our experiment.

We first describe the experiment settings. In the main part of this section, we first present and discuss the run time efficiency of different approaches as the main focus of this work is on improving scalability. Then, we evaluate the matching accuracy of different approaches using the usual measures: precision, recall, and F-1 score.

5.1 Experiment Settings

This section gives the settings of our experiments.

Table 4: Temporal matching tasks

Name	# Records ($\|R\|$)	# training records	Attributes	Years	Note
DBLP-100K	103059	5945 (5%)	author's name, conference venue, paper title, co-authors	1957 - 2012	not well labeled; included only for scalability experiment.
DBLP-Ambi	2664	888 (33%)	author's name, affiliation (manually filled), paper title, co-authors	1987 - 2012	258 authors share only 21 names, most authors evolve on all other attributes.
DBLP-WW	738	246 (33%)	author's name, affiliation (manually filled), co-authors	1991 - 2010	all authors share the same name and mostly evolve on all other attributes, also used in [12]
Euro-Patent	1871	623 (33%)	author first name initial, author last name, affiliation	1978 - 2003	fewer evolving entities, also used in [12]

5.1.1 Settings for Scalability Experiments

To compare the scalability of different approaches, we consider the **D**BLP-100K matching task that contains around 100K records. This matching task is created by selecting multiple groups of records from the DBLP data set, where each group contains multiple authors sharing the same name. Although this is not a particularly temporally challenging matching task, for completeness we will also report the matching accuracy of different approaches. For this purpose, we compare the output of each approach with the silver standard provided by DBLP. However, in this test, our goal is not to test the quality of the result, rather it is to test the scalability of the approaches.

5.1.2 Settings for Matching Accuracy Experiments

To evaluate the matching accuracy of different approaches, we consider three subsets from two real world data sets: a benchmark of European patent data [1] and the DBLP data set [2]. Two of the tasks, Euro-Patent and DBLP-WW, are the tasks used in Li's work [12]. Table 4 summarizes the high level facts about the matching tasks, and we will detail necessary properties of each task when analyzing the matching accuracy of different approaches.

5.1.3 Implementation

We implemented the following clustering techniques:

- PARTITION: the partitioning algorithm from [11], a single pass, non-temporal clustering approach that clusters records based on pairwise record similarity while ensuring transitive closure.
- ADJUST: a specialized multi-stage fuzzy clustering approach proposed in [12], the current state-of-the-art for matching temporal records.
- AFDS: the optimized version of SFDS described in Section 4 based on Section 3.

For the temporal model used in ADJUST and AFDS, we use the agreement- and disagreement-decay models proposed in [12].

We implemented all the techniques as single-threaded Java programs, and we ran the experiments on a Linux machine with 2.40 GHz Intel CPU and 4GB of RAM.

Attribute similarity metrics: For matching accuracy experiments, we followed the settings used in [12] that 1) use editing distance metricfor single-value attributes, 2) apply Jaccard similarity metricfor multi-value attributes, and 3) put equal weight on all attributes.

For scalability tests, we reused the settings of similarity metrics for the matching accuracy experiments except the weight of each attribute was learned from a training data set based on the discriminability of each attribute. Here we omit the details of how we weight the importance of each attribute as, again, the main purpose of this experiment was scalability.

Models and thresholds learning: For scalability experiments, we used a 5% sample of the records in the DBLP-100K data set to learn the temporal models and the thresholds and then tested with the whole data set.

For the other matching tasks: Euro-Patent, DBLP-WW and DBLP-Ambi, we used a three-fold cross-validation, which divides the data set into three disjoint partitions of nearly equal size. We learned temporal models and thresholds from one partition at a time and used them to test with the remainder of the data set.

Parameter settings: We followed the parameter settings used in [12] for ADJUST, and we used a minimum recall $min_R = 0.2$ for the first phase of the proposed AFDS in our experiments.

5.1.4 Metrics

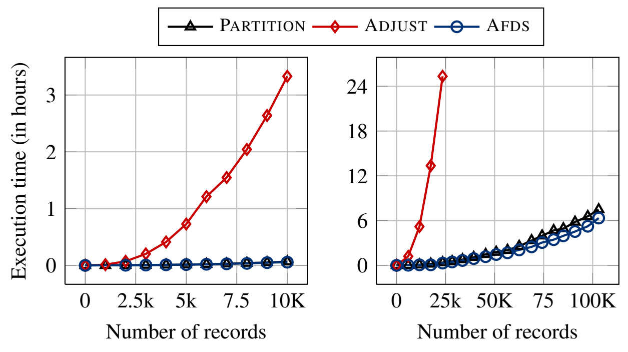
We compared pairwise matching decisions with the ground truth and measured the quality of the result by *precision* (P), *recall* (R), and *F-1 score* (F). We denote the set of false positive pairs by F_P, the set of false negative pairs by F_N, and the set of true positive pairs by T. Then, P $= \frac{|T|}{|T|+|F_P|}$, R $= \frac{|T|}{|T|+|F_N|}$, and F $= \frac{2\text{PR}}{\text{P+R}}$.

5.2 Scalability

First, we evaluate the scalability of the different techniques by running them with different fractions of the DBLP-100K data set. The experiment has two parts. In the first part, we ran all techniques with at most 10,000 records from the DBLP-100K data set and compare their run-time. The results in Figure 7a show that while both our approach AFDS and the traditional clustering technique PARTITION finish 10,000 records within 5 minutes, the state-of-the-art temporal clustering algorithm ADJUST takes 3 hours to finish the same amount of data. At the 10,000 records mark, our AFDS provides 60X improvement in run time over AFDS: AFDS finishes in 3.36 minutes while ADJUST finishes in 199.78 minutes, and the traditional approach PARTITION finishes in 3.99 minutes.

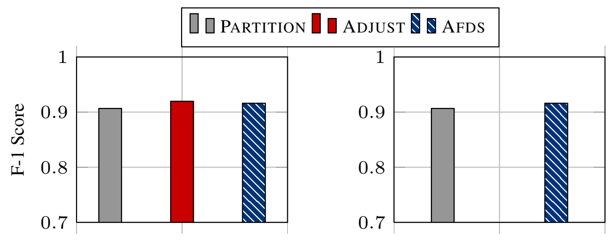
To further analyze the difference in scalability, we ran AFDS and PARTITION with the full DBLP-100K data set, and ran ADJUST with as many portion of the DBLP-100K data set as possible and stopped when it requires more than a day to perform the matching task. From the results shown in Figure 7b, we observe that ADJUST takes around a day to process 25k records, while AFDS and PARTITION is able to finish the full set of DBLP-100K matching task in 6.2 hours and 7.5 hours respectively.

The differences on time complexity between different approaches are also reflected in the scalability results. First, our AFDS runs slightly faster than the traditional approach PARTITION. This matches with the fact that AFDS runs in time $O(|R| \cdot |C|)$ while PARTITION runs in time $O(|R|^2)$, where $|R|$ is the number of input records and $|C|$ is the number of clusters created in the first phase of AFDS respectively. Second, based on the run-time curves of ADJUST, it performs faster than its worst case $O(|R|^3)$ but slower than its ideal case $O(|R|^2)$.



(a) on 10% DBLP-100K (b) on full DBLP-100K

Figure 7: Scalability test results on DBLP-100K matching task.



(a) on 10% DBLP-100K (b) on full DBLP-100K

Figure 8: Reference results on matching accuracy for the scalability on the DBLP-100K data set.

For completeness, we also provide the result qualities of all different approaches in Figure 8a and Figure 8b, although recall that this is not a particularly challenging case from a temporal perspective.

5.3 Results on Matching Accuracy

We now turn investigate whether or not AFDS sacrifices efficiency in its search for performance. To do so, we compared our AFDS approach with the existing approaches on three different temporal record matching tasks. The results are shown in Figures 9a, 9b, and 9c. Here we briefly discuss the results on different matching tasks.

DBLP-Ambi and DBLP-WW Matching Tasks: There are two important facts about this two matching tasks: 1) different entities are likely to have the same name; 2) most entities evolve their values on on all other attributes. As a result, the key to high matching accuracy is to resolve the ambiguity on name by handling evolution on all other attributes. The results in Figures 9a and 9b show that both AFDS and ADJUST improve up to 15% over the traditional non-temporal approach PARTITION, and our AFDS produces matching accuracy similar to ADJUST.

Euro-Patent Matching Task: Unlike the previous two matching tasks, in this data set there are relatively fewer entities that evolve over time. This has a two-fold effect. First, all approaches have less trouble separating records belonging to the same entity. Second, because fewer entities evolve over time, there are relatively fewer temporal clues for a temporal model to exploit. The results in Figure 9c reflect such an effect: all three approaches have acceptable matching accuracy while the two temporal clustering algorithms — AFDS and ADJUST — still improve matching accuracy by handling entity evolution.

Putting the results on scalability and matching accuracy together,

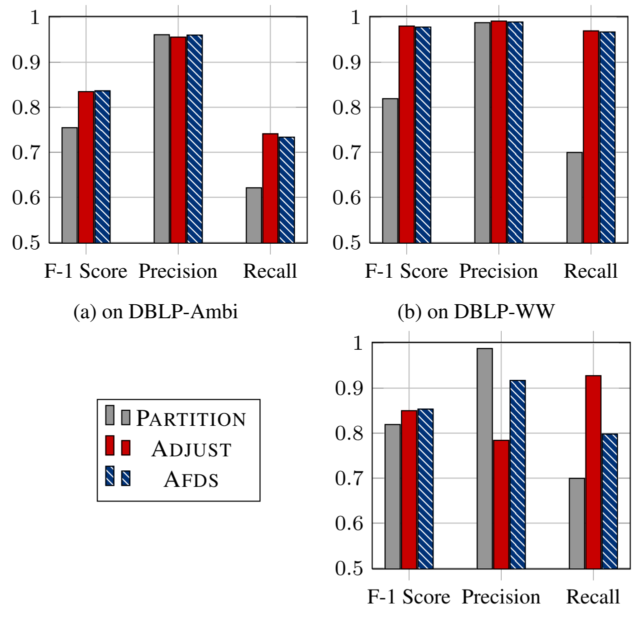


Figure 9: Result quality of different approaches on three different matching tasks.

our algorithm AFDS improves run time by an order of magnitude while providing equivalent matching accuracy.

5.4 Analysis and Discussion

Now we give a more detailed analysis of our proposed strategy.

5.4.1 Evaluation of "Static First, Dynamic Second"

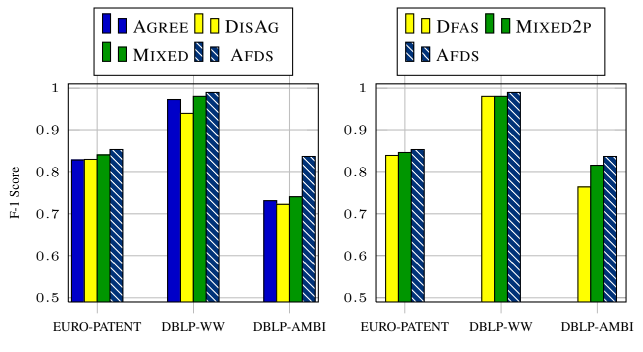
We verified the effectiveness of our "static first, dynamic second" strategy in two ways. First, we checked whether a single pass clustering algorithm with the use of temporal models could achieve equivalent or better accuracy over our proposed AFDS strategy. In this experiment, we considered using temporal models in a single stage hard clustering algorithm. The hard clustering algorithm we considered is identical to the S-phase described in Section 3.2 except the threshold is set to the value that leads to the highest F-1 score in the training data set:

1. AGREE: uses only temporal agreement model.
2. DISAG: only temporal disagreement model.
3. MIXED: uses full temporal model.

From the results shown in Figure 10a, we first observed that AFDS and MIXED improve result quality over AGREE and DISAG on all matching tasks. This finding indicates that when between-entity temporal agreement and within-entity temporal disagreement apply to part of the input data set, considering both types of ambiguity will improve result quality over handling only a single type of ambiguity. In addition, we also found that our proposed AFDS improves result quality over MIXED. This result shows by having an evidence collection phase before the handling of entity evolution, we can improve overall matching accuracy.

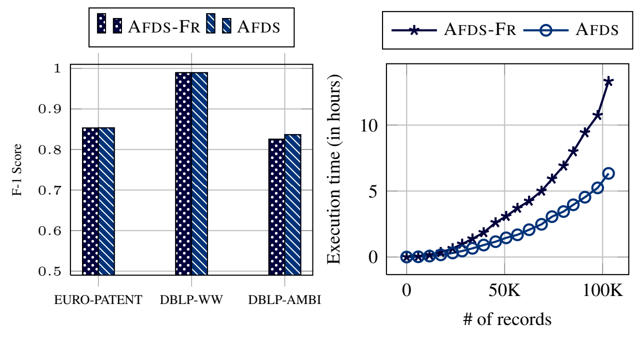
In the second experiment, we compared AFDS with the following two phase strategies, which handles temporal agreement and disagreement in different orders by running them with three different matching tasks:

1. DFAS: the inverse version of AFDS.



(a) with single-stage clusterings. (b) with two-staged clusterings.

Figure 10: Comparison between different uses of temporal models and AFDS.



(a) Matching accuracy (b) Scalability

Figure 11: Comparison between the use of signatures and the use of the full set of records.

2. MIXED2P: uses full temporal model in both phases.

Note that for all approaches, we use the same strategy to cluster records and construct cluster signatures as described in Section 3. The results in Figure 10b first indicate that AFDS yields better results than its complement — DFAS. This result indicates by considering temporal between-entity agreement, we can collect higher quality evidence than considering temporal within-entity disagreement. This result also suggests that resolving temporal within-entity disagreement requires more accurate evidence than resolving temporal between-entity agreement. Second, the result also shows that AFDS produces better matching quality than MIXED2P, which handles both types of ambiguity in both phases. This also supports our proposed "Static First, Dynamic Second" strategy that runs evidence collection phase by assuming no evolution before the handling of entity evolution instead of jumping into the dynamic world directly.

5.4.2 *Effectiveness of Temporal Signature*

We verified the effectiveness of our proposed temporal signature by comparing AFDS with its variation AFDS-FR that considers full sets of records without using signatures in similarity computations. We give the results on matching accuracy and scalability in Figure 11a and 11b. The results indicate that AFDS and AFDS-FR achieve equivalent matching accuracy, but AFDS improves 2X in run time efficiency over AFDS-FR.

5.4.3 *Results on Similarity Computation Strategies*

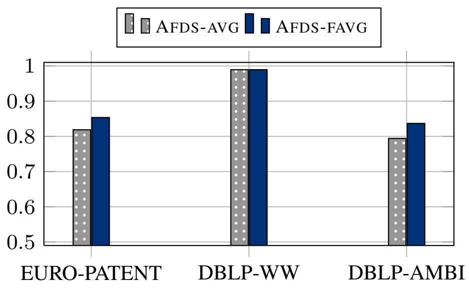


Figure 12: Result quality (F-1 score) comparison between different cluster similarity estimation strategies on different data sets.

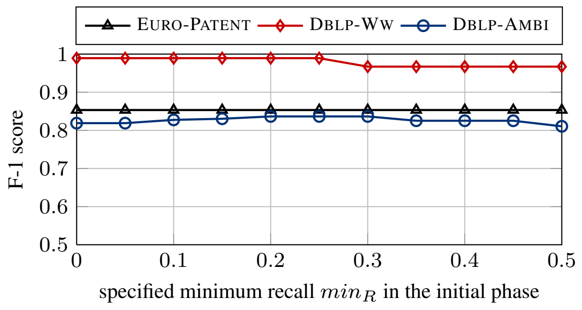


Figure 13: Results of varying minimum recall min_R in the initial phase of AFDS.

This experiment compares different strategies for computing temporal cluster similarity. In particular, we consider the following strategies:

1. AVG: directly averages record similarities.
2. FAVG: stands for filtered average, the proposed approach described in Equation 12.

Both strategies are paired with our AFDS approach and run on different matching tasks. The results shown in Figure 12 indicates that by using our proposed strategy — FAVG, which considers only those record pairs with high similarity that provide strong evolution evidence, we can make better inferences on entity evolution and improve matching accuracy.

5.4.4 *Results on Robustness*

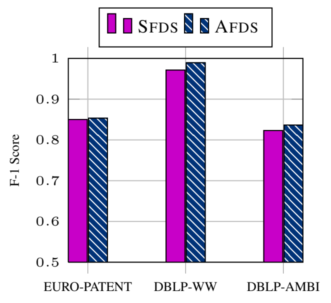
This experiment analyzes how the minimum recall parameter min_R (Section 3.2), used to learn the threshold θ_S in the first phase of our AFDS algorithm, affects the final matching accuracy. We give the results in Figure 13. The results indicate that (1) the final matching accuracy of AFDS does not heavily rely on the parameter min_R; (2) in our experiments, a minimum recall min_R between 0.2 to 0.3 produced the best result quality.

5.4.5 *Comparison between SFDS and AFDS*

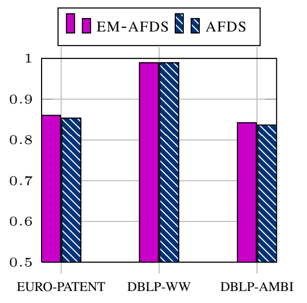
We compared the matching accuracy between SFDS and its optimized version (AFDS). The results show that we can improve up to 2% in matching accuracy by considering temporal agreement in the first phase (Figure 14a), and we observe no significant run-time difference between the two.

5.4.6 *Will EM-styled AFDS produce better accuracy?*

We developed an expectation-maximization version of our AFDS, where given a specific number I of iterations, we perform an EM-style S-phase in the first half of the iterations and an EM-style D-phase in the second half of the iterations. In our experiment, we



(a) SFDS vs. AFDS



(b) EM-Styled AFDS vs. AFDS

Figure 14: Comparison between different versions of AFDS.

set $I = 10$ for EM-AFDS. Surprisingly, we found that even with more stages of clustering, we did not produce noticeable improvement ($> 1\%$) on matching accuracy. Detailed results can be found in Figure 14b.

6. RELATED WORK

In [8, 10, 15], the authors proposed different record matching / de-duplication techniques, but their techniques all assume value difference are due to different representations of the same value and record values do not change over time. Shao *et al.* [13] combined time series similarity with density-based clustering, but the technique is for finding the optimal alignment of two time series, not for identifying possible evolution between time series. Yakout et al. proposed behavior based linkage [16]. While like our work they make cluster merging decisions by examining whether the resulting clusters form a reasonable entity history, our approach does not try to find periodical entity evolution patterns. Li *et al.* [12] proposed several clustering approaches for matching temporal records. Our approach differs from their approaches in the following ways. First, our approach resolves temporal agreement and disagreement in two distinct phases while their approaches mixes the two types of ambiguity in all phases. Second, the proposed clustering neither processes in EM-style nor in a fuzzy way. Finally, our approach allows one cluster to have multiple signatures while their approaches use either no signatures or a single signature per cluster. Both [5] and [6] used a temporal function to reduce the effect of older tuples on data analysis, but their definitions and the use of temporal functions are different from us. While [12] proposed temporal models, our focuses are not on temporal models but on clustering algorithm and signature construction instead.

7. CONCLUSION AND FUTURE WORK

We have proposed an efficient strategy to match temporal records with high accuracy. The key insight of our strategy is "static first, dynamic second," which allows simple, efficient clustering to do well. Our experimental results indicate that the proposed approach achieves the same level of result quality (within 1.0% in our experiments) as the state-of-the-art approach while providing an order of magnitude improvement (up to 60X at 10,000 records in our experiments) in run-time.

Substantial room for future work remains. One key issue is the blocking operator. Blocking is a common technique which speeds up clustering algorithms by partitioning the input data according to some key attributes. However, it is unclear how to apply blocking to a data set where the entities might change values on those key attributes over time. Here it would be interesting to see if our proposed "static first, dynamic second" strategy might work with existing blocking operators.

Other future work includes a more general model combining information from various dimensions such as spatial and temporal domains to enable deeper and wider data analysis.

8. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under grant no. IIS-1018792.

9. REFERENCES

[1] Academic patenting in Europe (APE-INV). `http://www.esf-ape-inv.eu/`.

[2] The DBLP computer science bibliography. `http://www.informatik.uni-trier.de/~ley/db/`.

[3] Fec-standardizer - an experiment to standardize individual donor names in campaign finance data. `https://github.com/cjdd3b/fec-standardizer`.

[4] Twitter - an online social networking and microblogging service. `https://twitter.com/`.

[5] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. *Journal of Algorithms*, 59(1):19–36, 2006.

[6] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu. Forward decay: A practical time decay model for streaming systems. In *IEEE 25th International Conference on Data Engineering (ICDE)*, pages 138–149. IEEE, 2009.

[7] A. Doan, A. Halevy, and Z. Ives. *Principles of Data Integration*, chapter Data Matching. Elsevier Science, 2012.

[8] P. Domingos. Multi-relational record linkage. In *Proc. of the KDD-2004 Workshop on Multi-Relational Data Mining*. KDD, 2004.

[9] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[10] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, pages 1183–1210, 1969.

[11] O. Hassanzadeh, F. Chiang, H. Lee, and R. Miller. Framework for evaluating clustering algorithms in duplicate detection. *Proceedings of the VLDB Endowment*, 2(1):1282–1293, 2009.

[12] P. Li, X. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *Proceedings of the VLDB Endowment*, 4(11):956–967, 2011.

[13] J. Shao, K. Hahn, Q. Yang, C. Bohm, A. Wohlschlager, N. Myers, and C. Plant. Combining time series similarity with density-based clustering to identify fiber bundles in the human brain. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 747–754. IEEE, 2010.

[14] G. Weikum, N. Ntarmos, M. Spaniol, P. Triantafillou, A. Benczúr, S. Kirkpatrick, P. Rigaux, and M. Williamson. Longitudinal analytics on web archive data: It's about time! In *Proceedings of the 5th biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, January*, pages 9–12, 2011.

[15] W. Winkler. Methods for record linkage and bayesian networks. Technical report, Statistical Research Division, US Census Bureau, Washington, DC, 2002.

[16] M. Yakout, A. Elmagarmid, H. Elmeleegy, M. Ouzzani, and A. Qi. Behavior based record linkage. *Proceedings of the VLDB Endowment*, 3(1-2):439–448, 2010.