Linking Temporal Records for Profiling Entities

Furong Li[†], Mong Li Lee[†], Wynne Hsu[†] and Wang-Chiew Tan[‡]
[†]National University of Singapore [‡]UC Santa Cruz
{furongli,leeml,whsu}@comp.nus.edu.sg tan@cs.ucsc.edu

ABSTRACT

To harness the rich amount of information available on the Web today, many organizations start to aggregate public (and private) data to derive new knowledge bases. A fundamental challenge in constructing an accurate integrated knowledge repository from different data sources is to understand how facts across different sources are related to one another over time. This challenge, referred to as the *temporal record linkage* problem, goes far beyond the traditional record linkage problem as it requires a fine-grained analysis of how two facts are temporally related if they both refer to the same entity.

In this paper, we present a new solution for understanding how two facts may be temporally related and exploit the knowledge to profile how entities evolve over time. Our solution makes use of a novel *transition model* which captures sophisticated patterns of value transitions. Specifically, our transition model captures the probability that an entity may change to a particular attribute value after some time period. This transition model can be considered jointly with various source quality metrics to fine-tune how records should be temporally linked to entities. In particular, we showcase how the freshness of data sources can be built into a *source-aware temporal matching* algorithm that jointly considers the value transitions and the freshness of data sources to link temporal records to entities in the right time period. In this way, an increasingly complete and up-to-date entity profile can be derived as more and more temporal records are aggregated from different sources. Our suite of experimental results on real world datasets demonstrate that our proposed method is able to outperform the state-of-the-art techniques and build more complete profiles for entities by identifying their true matching temporal records at the right time period.

1. INTRODUCTION

To harness the rich amount of information available on the Web today, many organizations aggregate public (and private) data to derive new knowledge bases. Examples include popular public knowledge bases such as the YAGO [22] and DBPedia [2], which provide a central knowledge repository for entities based on information that are extracted and aggregated from various sources on the Web, and private knowledge bases built by companies such as Instant Checkmate [14], which aggregate police records, background reports, and even online profiles to build a complete understanding of an individual.

Table 1: Employment history of a job seeker

Name	Organization	Title	Start	End
David Brown	S3	Engineer	2000	2001
	XJek	Engineer	2000	2002
	Aelita	Manager	2003	2005
	Quest Software	Manager	2006	2009

A fundamental challenge in constructing an accurate integrated knowledge base from different data sources is to understand how facts across different sources are related to one another *over time*. This goes beyond the traditional *record linkage problem* [1], which refers to the task of identifying facts that correspond to the same entity. In this case, the goal is to understand whether two facts refer to the same entity at possibly different times, which is a more "fine-grained" analysis of the relationship between two facts.

To determine whether or not two records (or facts) refer to the same entity, existing record linkage algorithms [10, 12] usually compute the degree of similarity between the two records. The degree by which the two records *match* will determine whether or not they refer to the same entity. While many of these techniques have been applied to data in several different domains with good results [15], they are typically inadequate for identifying whether or not two records, especially if they are dissimilar, actually refer to the same entity at different times. This is because an entity may change several of its attribute values over time (e.g., age, location, and job), and records about the same entity at different times may be deemed dissimilar according to the similarity metric. In short, traditional record linkage algorithms are typically agnostic to the temporal dimension and may thus fail to identify entity changes over time. For this reason, a number of temporal record linkage models have been recently proposed to address this issue (see Section 2 for further details).

In this paper, we present a new solution for understanding how facts may be temporally related and exploit the knowledge to profile how entities evolve over time. Our framework, called MAROON, makes use of a novel *time-aware* transition model and a *source-aware* matching algorithm to effectively construct entity profiles over time. We illustrate the problem and the key ideas behind MAROON with examples next.

Example 1. Consider an online recruitment system, e.g., Monster.com, where organizations advertise positions available for job

[1]Also known as *entity resolution* or *reference reconciliation*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGMOD'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.
Copyright 2015 ACM 978-1-4503-2758-9/15/05 ...$15.00.
http://dx.doi.org/10.1145/2723372.2737789.

Table 2: Records obtained from various sources ($\surd$ for match and $\times$ for unmatch)

	Name	Organization	Title	Location	Interests	Time	Source
$\surd$ r_1	David Brown	S3, XJek	Engineer			2001	Google+
$\surd$ r_2	David Brown		Engineer			2002	Google+
$\surd$ r_3	David Brown	S3, XJek	*Engineer*			2004	Facebook
$\surd$ r_4	David Brown		Manager	Chicago		2004	Twitter
$\surd$ r_5	David Brown	Quest Software	Director		Technology	2011	Google+
$\times$ r_6	David Brown	Quest Software	IT Contractor			2011	Google+
$\surd$ r_7	David Brown		*Engineer*	Chicago	Sports, Politics	2012	Facebook
$\surd$ r_8	David Brown		President	Chicago		2013	Twitter
$\surd$ r_9	David Brown	WSO2	President		Technology	2013	Google+

seekers. This system also value-adds by analyzing the information of its users to recommend relevant jobs to a job seeker, or suitable candidates to an employer. Table 1 shows the employment history submitted by a job seeker called David Brown.

In order to provide good recommendations, the system may want more complete profiles of its users, such as demographics, skill sets and interests. To achieve this, the online recruitment system may mine further information about "David Brown" from the Web. Table 2 shows some additional information about "David Brown" that is obtained from the Web after the search results have been transformed into structured records. Each row in the table depicts some aspects of David Brown obtained from a data source and the "Time" attribute indicates when this record was first published by the source. An attribute value that is empty denotes that no information about that attribute was derived from that data source.

The next step is to determine which records in Table 2 actually refer to the entity David Brown who has submitted his employment history as shown in Table 1. If we use traditional record linkage techniques to match records based on the similarity of attribute values, we are able to match records r_1-r_4 to the target entity David Brown because they share the same organization and job titles. However, for records r_5 and r_6, where the job titles are different from any of the job titles in Table 1, these techniques will conclude that r_5 and r_6 do not refer to this David Brown.

A closer observation will reveal that the timestamps of these two records (r_5 and r_6) fall outside the employment history in Table 1. These records could very well refer to the same David Brown but at a later time (i.e., 2011) and that they describe how his job titles evolved in 2011. Hence, with traditional record linkage techniques, the system may miss the opportunity to augment the profiles of its users with more up-to-date information. □

To understand the evolution of entities, the work of [18] first proposed a time decay temporal model that captures the likelihood that an attribute value of an entity will change within a time period. This model was recently extended by [5], where a mutation model was used to learn the probability that an attribute value will reappear over time.

Although the existing temporal models capture the likelihood that an entity will change its attribute values, they do not account for the sophisticated patterns of value transitions that may exist. For example, consider again the records r_5 and r_6 in Table 2 which have the same organization but different titles. Suppose the temporal model in [5] gives a low recurrence probability on the attribute Title, i.e., the job titles in Table 1 are not going to recur in year 2011. Then it will lower the weight of attribute Title on the overall similarity score, and as a result, both r_5 and r_6 are equally likely to refer to the target David Brown. However, r_6 may not be a true match as it is highly unlikely for David, who was a Manager in 2010, to become an IT Contractor in 2011. On the other hand, the record r_5 is possibly a match as it is more likely that David will be promoted to Director, after having been a Manager for 8 years.

Table 3: Updated profile of David Brown

Organization	Title	Location	Interests	Start	End
S3	Engineer			2000	2001
XJek	Engineer			2000	2002
Aelita	Manager	Chicago		2003	2005
Quest Software	Manager	Chicago		2006	2009
Quest Software	Director	Chicago	Technology Sports, Politics	2011	-

Our key observation is that when the attribute values of an entity change, they do not change arbitrarily. Rather, the new value that an attribute evolves to is typically dependent on its previous value and also the duration by which the previous value holds. As our previous example illustrates, it is more likely for David's job title to evolve from "Manager" to "Director", than to "IT Contractor". Further, the promotion to the position of a "Director" is more likely to happen after 8 years than just after 1 year.

Based on the above observations, we design a novel *transition model* for MAROON that learns the probabilities of the transitions between different values over time. This transition model enables us to answer the following question: if an attribute of an entity currently has a value v, what is the probability that this value will change to a value v' after Δt time? For example, given that David Brown's current title is "Manager", our transition model will compute the probabilities that his job title will change to "Director" and, respectively, "IT Contractor" after n years, for some positive number n. This allows us to discriminate correctly between records such as r_5 and r_6. By determining that r_5 (and not r_6) is a likely match, we can then augment David's employment history with the fact he was a Director in Quest Software from 2011 onwards.

Another major challenge with linking temporal records from multiple sources is to understand how the quality of the sources (e.g., whether the information published by a source is reliable and up-to-date) can be effectively used to tilt the decision on whether or not to match a record to an entity. In this paper we focus leveraging the freshness of the sources. Other qualities such as source reliability (i.e., the likelihood that a source publishes erroneous values) has been studied in [17] and can be incorporated in the future.

Example 2. Going back to our example in Table 2, suppose the transition model gives a low probability that the job title will change from "Director" to "Engineer" in the next year. Then, r_7 will not be linked to David. However, if we account for the fact that the source of r_7 is Facebook, which provides reliable and up-to-date contact information and interests, even though the job titles may not be as up-to-date. Then one should still consider r_7 as a match for David Brown and augment his profile with interests "Sports, Politics" (see Table 3). □

As we shall describe, our framework MAROON combines the transition model, which can capture sophisticated transition patterns between different values over time, with a source freshness model to arrive at an effective matching algorithm that is both *time-aware* and *source-aware* when matching records. Intuitively, the matching algorithm first generates a set of clusters from the input records, where the placement of a record depends on the *update delays* (i.e., freshness) of its source. Subsequently, the algorithm iteratively identifies a subset of clusters that match with the target entity, and updates the entity profile based on these matches. The process by which a cluster is matched with the entity takes into account both the probability of value transitions and the support of the data sources. To the best of our knowledge, this is the first solution for temporal record linkage that jointly models the temporal nature of entities and the freshness of data sources to determine if two records with (conflicting) information can be linked. The transition model is a novel contribution on its own and we show how it is instrumental for obtaining a finer-grained understanding of how entities evolve in our experiments.

Contributions. In summary, this paper makes the following contributions to the problem of building entity profiles via linking temporal records:

1. We develop a novel transition model which captures the probability that an entity will change to a particular attribute value at a later time. This transition model enables a fine-grained understanding of how entities evolve over time.
2. We design a source-aware temporal matching algorithm that jointly considers the probabilities of value transitions and the update delays of data sources to identify future states for a target entity.
3. We evaluate the effectiveness of MAROON with an extensive set of experiments on real world datasets. Our experimental results demonstrate that the proposed framework outperforms the state-of-the-art techniques and it is able to build more complete profiles for entities through effective matching of temporal records to the correct entities at the proper time periods.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 provides the preliminaries. Section 4 presents our proposed framework MAROON, and Section 5 contains experimental results. We conclude in Section 6.

2. RELATED WORK

The problem of identifying information that corresponds to the same real world entity has been extensively studied in the literature [10, 12, 15, 16]. Techniques to match records can be roughly categorized into two categories: learning-based vs. non-learning algorithms. Learning-based algorithms (e.g., [1, 3]) train a classifier to label each pair of records as match or unmatch, while the non-learning methods may derive a set of rules or constraints to link records (e.g., [11, 23]). However, as mentioned in the Introduction, these approaches are usually based on similarity calculation that is agnostic to the temporal dimension.

Temporal Record Linkage Models. Two temporal models [18, 5] have been proposed to link records over time. The *time decay model* [18] captures entity evolution by modeling the probability that an entity changes its attribute value within a time interval (i.e., disagreement decay) and the probability that two different entities share the same attribute value over time (i.e., agreement decay). The *mutation model* [5] learns the probability that an attribute value reappears over time. The model constructs a recurrence function which tells how likely a value on an attribute will reappear after Δt time. The mutation model provides better insight than the time decay model for attributes such as the co-authors of a researcher whose values may change back and forth. Furthermore, it makes matching decisions based on the entire history of an entity, and not on a single time point as was done in [18].

Both the time decay and mutation models focus on the changing behavior over time, and do not consider the complex value transition patterns that may exist in the entities over time. The mutation model does not capture the fact that even for the same attribute, different values may have different recurrence probabilities. Also, both models cannot discriminate the various values that an entity may change to. In contrast, our proposed transition model overcomes the above limitations and can thus offer more fine-grained analysis of the values on an attribute.

In [4], declarative rules were used to link records based on temporal information. Our approach can complement [4] by learning how records are temporally related for cases where no obvious declarative rules can be specified.

Temporal Clustering. Several algorithms have been proposed in the past for clustering records with timestamps into clusters so that each cluster represents the history of an entity. In [18], three such algorithms were proposed. The first is an early binding algorithm that merges a record to an existing cluster if the record is sufficiently similar to the records in the cluster. Otherwise, a new cluster with that record is created. In contrast, the late binding algorithm first generates a soft clustering of records, where a record can belong to more than one cluster, before a decision is made on which cluster each record should belong to. The last algorithm is the adjusted binding algorithm, which starts with the soft clustering obtained by the late binding algorithm, and iteratively refines the clusters. Unlike the previous two methods, the adjusted binding algorithm allows a record to be compared with clusters that are created later. The method in [6] clusters records in two phases. The first phase assumes that records are static and groups them based on attribute value similarities. The second phase 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.

The above clustering algorithms do not take into account the characteristics of the data sources. In contrast, our proposed algorithm considers whether the temporal records published by the data sources are up-to-date. This enables us to obtain more accurate intervals for the clusters. The subsequent matching step iteratively links a cluster to the target entity profile based on both the transition probability and the support of the data sources. In addition, we adopt a more fine-grained cluster signature to describe the states for each attribute, where the signature of each attribute is a sequence of values across some time interval.

Use of Data Sources. The knowledge about data sources has been shown to be helpful for record linkage. The work of [21] estimates the semantic ambiguity of each source, and applies either a relaxed or a conservative matching criteria on the source based on how ambiguous the source is. In [17], the authors capture the source reliabilities for different attributes with a reliability matrix, which is then used to lower the impact of erroneous values on the matching decisions. Here, our matching algorithm considers the freshness of the published records.

The work in [20] examines the problem of source selection for data integration where the source content may change over time. Their focus is on estimating the quality of a source at a future time according to the changes of the real world and the effectiveness of the data sources on capturing these changes. They do not consider

how the obtained source qualities can help in the temporal linkage of records.

3. PRELIMINARIES

We first provide a definition of the terms and notations used in this paper.

Entity, Attribute, Temporal Sequence. An *entity* refers to a real world object. We consider a set $\mathcal{N}$ of entities where each entity $n \in \mathcal{N}$ is associated with a set $\mathcal{A}$ of *attributes* with corresponding *attribute values*. Table 1 shows an example of an entity David Brown with attributes Name, Organization and Title. The attribute values of an entity may evolve over *time*, and an attribute of an entity can take multiple values at the same time. For example, David Brown is at both organizations S3 and XJek in 2000.

We view time as a linear structure $(\mathbf{T}, \prec)$, where $\mathbf{T}$ is a discrete set of time instants and $\prec$ is a precedence relation on $\mathbf{T}$. The granularity of $\mathbf{T}$ depends on the application. Then a *triple* $\langle b, e, V\rangle$ denotes that the set V is known to be valid for time interval $[b, e]$ where $b \in \mathbf{T}$, $e \in \mathbf{T}$, $b \leq e$, and V is a finite set of values over the domain of some attribute. We model the set of values that is associated with an attribute of an entity over time as a *temporal sequence* which we define next.

Definition 1. A *temporal sequence* Seq of an attribute is a finite list of triples where for every consecutive pair $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ of triples in the sequence, it must be that $e < b'$ and $V \neq V'$.

Value, Interval, Lifespan. We use $\text{Values}(Seq, t)$ to denote the set of values that occur at some time point t in a temporal sequence Seq, that is, $\text{Values}(Seq, t) = V$ where $t \in [b, e]$ and $\langle b, e, V\rangle \in Seq$. Similarly, we write $\text{Intervals}(Seq, v)$ to denote the set of intervals during which the value v occurs, that is, $\text{Intervals}(Seq, v) = \{[b, e] \mid \langle b, e, V\rangle \in Seq, v \in V\}$.

Let $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ be the first and the last triples in Seq respectively. We define $\text{Lifespan}(Seq) = e' - b + 1$.

Entity Profile. For an entity n, we use the notation Φ_n to denote the *entity profile* of n. Intuitively, the entity profile of n describes how the attribute values of n change over time. The evolution of attribute A of n is captured by $\Phi_n[A]$, which is a temporal sequence such that $\langle b, e, V\rangle \in \Phi_n[A]$ if and only if the entity n has value V on attribute A in time interval $[b, e]$. We write Φ to denote a set of entity profiles, and $\Phi[A]$ for the set of temporal sequences on attribute A in Φ. We illustrate these concepts with Example 3.

Example 3. From Table 1, we have two profile sequences:

$\Phi_{David}[\text{Organization}] = [\ \langle 2000, 2001, \{\text{S3}, \text{XJek}\}\rangle,$
$\langle 2002, 2002, \{\text{XJek}\}\rangle, \langle 2003, 2005, \{\text{Aelita}\}\rangle,$
$\langle 2006, 2009, \{\text{Quest Software}\}\rangle\ ]$

$\Phi_{David}[\text{Title}] = [\ \langle 2000, 2002, \{\text{Engineer}\}\rangle,$
$\langle 2003, 2009, \{\text{Manager}\}\rangle\ ]$

Then for the attribute Title, we have:

$\text{Values}(\Phi_{David}[\text{Title}], 2002) = \{\text{Engineer}\},$
$\text{Intervals}(\Phi_{David}[\text{Title}], \text{Engineer}) = \{[2000, 2002]\},$
$\text{Lifespan}(\Phi_{David}[\text{Title}]) = 10.$ □

Completeness. Let $\text{Intervals}(Seq)$ be the list of intervals in all the triples of a temporal sequence Seq. We say a profile sequence $\Phi_n[A]$ is *complete* w.r.t. the interval $[b, e]$ if $\bigcup \text{Intervals}(\Phi_n[A]) = [b, e]$. In other words, the intervals found in $\text{Intervals}(\Phi_n[A])$ constitute the set of all time instants in $[b, e]$. Then an entity profile Φ_n is *complete* w.r.t. $[b, e]$ if for every attribute $A \in \mathcal{A}$, the temporal sequence $\Phi_n[A]$ is complete w.r.t. $[b, e]$. This means that for every attribute A, the values of A captured by $\Phi_n[A]$ cover every time instant in the interval $[b, e]$. To exemplify, $\Phi_{David}[\text{Organization}]$ shown in the Example 3 is not complete w.r.t. $[2000, 2013]$ as it does not contain any value for the time period $[2012, 2013]$.

Data Source, Temporal Record. We assume there is a set $\mathcal{S}$ of independent *data sources* that publish their observations of real world entities as *temporal records*, and these records have been mapped to a uniform schema through schema matching techniques. A *temporal record* r is a record $\langle A_1\colon v_1, ..., A_N\colon v_N, t, s\rangle$ of attribute-value pairs, together with the timestamp t and the data source s to indicate the time at which the record is published by the source. We use $r.A$ to denote the value on attribute A of record r. Table 2 shows a set of temporal records with attributes Name, Organization, Title, Location, Interests, Time, and Source. Each temporal record states some properties about an entity that is valid for a particular time instant. A record may contain missing values for some of the attributes (in this case $r.A$ is empty).

4. THE MAROON FRAMEWORK

We are now ready to provide a statement of the problem to enhance an entity profile over time under our MAROON framework.

Problem Definition. Given a target entity n and a set $\mathcal{R}$ of temporal records, the goal is to identify the records in $\mathcal{R}$ that refer to n, and augment its entity profile Φ_n with accurate information.

In other words, given an entity profile Φ_n and a set $\mathcal{R}$ of temporal records from multiple sources, we want to determine all the records in $\mathcal{R}$ that refer to the entity represented by Φ_n. For example, if $\mathcal{R}$ is the set of temporal records in Table 2, and Φ_n refers to the entity David Brown in Table 1, then the goal is to determine which records in Table 2 refer to this David Brown. The MAROON framework provides a new solution to this problem that overcomes two immediate challenges.

First, the attribute values of a record $r \in \mathcal{R}$ may be different from those in Φ_n because r is a different entity from n, or r is the same entity as n but at a different time. Thus one needs to understand that if r and n both refer to the same entity, how likely is it that r represents an evolution of n. We propose a transition model that captures the probability of value transitions over time as a solution. More precisely, the transition model answers the following question: if the value of attribute A of an entity is v, what is the probability for this value to be v' after Δt time? This transition model is learnt from a set of entity profiles and we present the details in Section 4.1.

Second, not all the values in a record are accurate as they may be obsolete. The transition model can be considered jointly with various source quality metrics to fine-tune how records should be temporally linked to entities. In this paper, we characterize the freshness of data sources to obtain a distribution of the *update delays* (i.e., freshness) for each attribute of each data source. Section 4.2 describes how we model the source freshness.

Given the value transition model and the source freshness model, we show how freshness can be built into a source-aware temporal matching algorithm that jointly considers the evolution of entities and the freshness of data sources to link temporal records to entities in the right time period. As we shall describe in Section 4.3, our matching algorithm first leverages the source freshness to generate a set of clusters from the input records. Subsequently, it iteratively identifies clusters that best match the entity, and updates its profile.

4.1 A Transition Model for Values

We first introduce our transition model for values, which is learnt from a set of clean and complete entity profiles. This transition

model is based on the key observation that an entity is unlikely to change its attribute values arbitrarily. Instead, the new value that an attribute takes is usually *dependent* on some of the previous values in its history. For instance, a person who used to work as an engineer is more likely to evolve to a project manager than, say, a real estate agent. In addition, the probability of observing a specific attribute value is also dependent on the *time interval* from the occurrence of the previous value. For example, it is more likely that a person's job title changes from "Engineer" to "Manager" *after five years* than just after one year.

4.1.1 Construct Transition Tables

Based on the above observations, we design a *transition model* to capture the value transition behavior for each attribute over time. More precisely, given that the value on attribute A is v, the model determines the probability that it will change to v' after Δt time.

We introduce the notion Δt*-transition* to indicate that a value v' occurs Δt time after an occurrence of v on some attribute A in an entity profile.

Definition 2. Given a set Φ of entity profiles and an attribute A, a pair (v, v') of values forms a Δt*-transition* if there exists a time instant t and a profile sequence $Seq \in \Phi[A]$ such that $v \in \text{Values}(Seq, t)$ and $v' \in \text{Values}(Seq, t + \Delta t)$.

Example 4. Consider the temporal sequence $\Phi_{David}[\text{Title}]$ in Example 3. If $\Delta t = 3$, then we have two Δt-transitions: (Engineer, Manager) and (Manager, Manager). □

To obtain the probabilities of the Δt-transitions for an attribute A, we first construct a transition table $T^A_{\Delta t}$ for each A and for each Δt, where Δt is a discrete value in the range $[1, L]$ and L is the maximum lifespan of the profiles in $\Phi[A]$. To simplify discussion, we assume the granularity of Δt to be year. In practice it can be set according to the frequency of entity evolutions. A transition table $T^A_{\Delta t}$ is derived from a set $\Phi[A]$ of temporal sequences. It is essentially a key-value store where each transition (v, v') serves as a key, while the value is the number of occurrences of (v, v') observed in $\Phi[A]$.

Conceptually, we can construct $T^A_{\Delta t}$ by sliding a window of size Δt to traverse each temporal sequence in $\Phi[A]$. The pair of values at the start point and, respectively, the end point of the window forms a Δt-transition. The occurrence of each Δt-transition is aggregated to the corresponding entries in $T^A_{\Delta t}$.

Example 5. Figure 1 shows two temporal sequences that depict the job titles of two entities "David" and, respectively, "Tom" over a lifespan of 10 years. Suppose Δt=3, one can compute T^{Title}_3 by sliding a window of size 3 across each temporal sequence.

We start from David's profile, the first window w_1 covers the time interval $[2000, 2003]$, where David is an "Engineer" at the beginning of the interval and he is a "Manager" at the end of the interval. This corresponds to a transition (Engineer, Manager). We create this transition entry in T^{Title}_3 and initialize its count to 1. Similarly, the next window w_2 covers the interval $[2001, 2004]$. Again, David is an "Engineer" at the beginning of the interval and he is a "Manager" at the end of the interval. This constitutes another occurrence of the transition (Engineer, Manager), and the corresponding entry in T^{Title}_3 increases to 2. The above process terminates when the end point of the sliding window reaches the year 2010. At this point, there are only two entries in the table: the count for transition (Engineer, Manager) is 3 and the count for transition (Manager, Manager) is 4.

Continuing with Tom's profile, the leftmost window w_3 identifies a transition from "Engineer" to "Analyst". Hence, the count

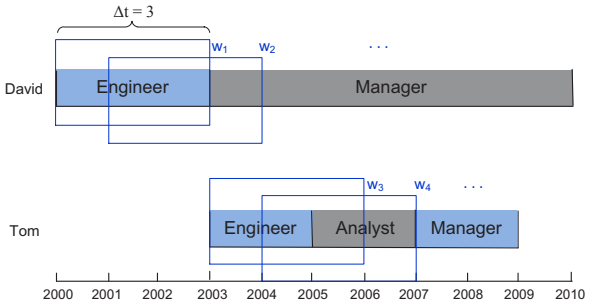


Figure 1: Count Δt-transitions with a sliding window (Δt=3)

Table 4: Transition table T^{Title}_3 obtained from Figure 1

(v, v')	count
(Engineer, Manager)	4
(Manager, Manager)	4
(Engineer, Analyst)	1
(Analyst, Manager)	1

for the entry (Engineer, Analyst) in T^{Title}_3 is initialized to 1. When we slide the window to w_4, we find an occurrence of the transition (Engineer, Manager) and therefore, increase the value of this transition by 1. Table 4 shows the resulting transition table that is obtained after going through these two profile sequences. □

We show next that $T^A_{\Delta t}$ can in fact be constructed by directly reasoning about the intervals associated with the triples in each temporal sequence. We first observe that the valid Δt for a pair of triples of the form $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ where $b \le b'$ can be computed directly, as shown by the following lemma.

LEMMA 1. *Let $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ be two triples from a temporal sequence where $b \le b'$. Then Δt ranges from $\max\{1, b' - e\}$ to $(e' - b)$.*

PROOF. As shown in Figure 2, the gap between the intervals $[b, e]$ and $[b', e']$ is $\max\{0, b' - e - 1\}$. Thus, the smallest window that can overlap with both intervals is $\Delta t_{min} = \max\{1, b' - e\}$, which gives the minimum value for Δt. Similarly, the largest value for Δt is given by $\Delta t_{max} = e' - b$. □

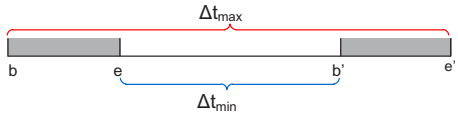


Figure 2: The valid range of Δt for a pair of triples

Next, we show how the number of occurrences of a Δt-transition (v, v') in a pair of triples can be computed directly.

PROPOSITION 1. *Given two triples $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ from a temporal sequence where $b \le b'$, the number of occurrences of a Δt-transition $(v, v') \in \mathcal{V} \times \mathcal{V}'$ is given by*

$$\min\{e, e' - \Delta t\} - \max\{b, b' - \Delta t\} + 1$$

where $\max\{1, b' - e\} \le \Delta t \le (e' - b)$.

PROOF. Suppose we have an interval $[x, x + \Delta t]$ that overlaps with both $[b, e]$ and $[b', e']$. Then we have

$$x \in [b, e]$$
$$x + \Delta t \in [b', e']$$

Simplifying, we have $x \in [\max\{b, b'-\Delta t\}, \min\{e, e'-\Delta t\}]$.

Thus, the number of occurrences of a Δt-transition is given by $(\min\{e, e'-\Delta t\} - \max\{b, b'-\Delta t\} + 1)$. $\square$

Proposition 1 enables us to directly process each pair $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ of triples, where $b \leq b'$, to determine the number of occurrences of all valid Δt-transitions from these triple pairs. More specifically, to compute the transition table for an attribute A of an entity n, one simply needs to consider all pairs $\langle b, e, V\rangle$ and $\langle b', e', V'\rangle$ of triples, where $b \leq b'$, and then count the number of transitions of each pair of values in $V \times V'$ according to Proposition 1. Algorithm 1 describes how we obtain a set $\mathcal{T}_A$ of transition tables for an attribute A from a set $\Phi[A]$ of profile sequences.

Algorithm 1: Construct Transition Tables

input : Set Φ of entity profiles and attribute A
output: Set $\mathcal{T}_A$ of transition tables for A

```
1  T_A ← ∅;
2  foreach Φ_n[A] ∈ Φ[A] do
3      foreach pair ⟨b, e, V⟩ and ⟨b', e', V'⟩, b ≤ b' do
4          for Δt ← max{1, b' − e} to (e' − b) do
5              foreach (v, v') ∈ V × V' do
6                  if transition (v, v') does not exist in T_Δt then
7                      Create entry (v, v') and initialize its value to 1;
8                  else
9                      T_Δt[(v, v')] ← T_Δt[(v, v')] +
                       (min{e, e'−Δt} − max{b, b'−Δt}) + 1
10     T_A ← T_A ∪ {T_Δt} ;
```

4.1.2 Derive Transition Probabilities

We show next how we derive a transition probability function $\Pr(v, v', \Delta t, A)$ based on the transition tables for an attribute A that we have constructed in the previous section. The function $\Pr(v, v', \Delta t, A)$ is the probability that the attribute A of an entity is v' given that the value at Δt time before was v.

Given a transition table $T_{\Delta t} \in \mathcal{T}_A$, let $\mathcal{V}$ and, respectively, $\mathcal{V}'$ be the set of all values that occur in the first component and, respectively, second component of the pairs of values that occur in $T_{\Delta t}$. More precisely, $\mathcal{V} = \{v \mid \exists v' \; \exists c \; \langle (v, v'), c\rangle \in T_{\Delta t}\}$ and $\mathcal{V}' = \{v' \mid \exists v \; \exists c \; \langle (v, v'), c\rangle \in T_{\Delta t}\}$. Then for each transition (v, v') in each transition table $T_{\Delta t} \in \mathcal{T}_A$, we define $\Pr(v, v', \Delta t, A)$ as the conditional probability that the value v' appears Δt time after v appears:

$$\Pr(v, v', \Delta t, A) = \frac{T_{\Delta t}[(v, v')]}{\sum\limits_{x \in \mathcal{V}'} T_{\Delta t}[(v, x)]} \tag{1}$$

In the equation above, $T_{\Delta t}[(v, v')]$ denotes the count that is associated with the transition (v, v') in the transition table $T_{\Delta t}$.

In our computation, the extreme cases are defined as follows. If Δt = 0, we set the transition probability to 1.0. If $\Delta t \geq L$, the transition probability is taken to be the probability at $(L-1)$. In other words,

$$\Pr(v, v', \Delta t, A) = \begin{cases} 1.0 & \Delta t = 0 \\ \Pr(v, v', \Delta t, A) & \Delta t \in (0, L) \\ \Pr(v, v', L-1, A) & \Delta t \geq L \end{cases} \tag{2}$$

However, in practice, the set of entity profiles (or training data) may not contain all the possible transition pairs (v, v') for all Δt. That is, transitions that are not known a priori will have to be accounted for. Suppose (u, u') is an unseen Δt-transition. We have 4 possible scenarios.

- Case 1. $u \in \mathcal{V}$ and $u' \in \mathcal{V}'$.
  This corresponds to the case that there are Δt-transitions that originate from u, but not to the value u', or transitions that end with u' but do not originate from u. In this case, we will consider the transition (u, u') to be rare and define its probability as the minimum transition probability w.r.t. the value u:
  $$\Pr(u, u', \Delta t, A) = \min_{v' \in \mathcal{V}'} \{\Pr(u, v', \Delta t, A)\} \tag{3}$$
- Case 2. $u \in \mathcal{V}$ and $u' \notin \mathcal{V}'$.
  This case is similar to Case 1. We have seen u but not u' and this makes (u, u') an unseen Δt-transition. Once again, we will consider the transition (u, u') to be rare and define its probability as the minimum transition probability w.r.t. the value u:
  $$\Pr(u, u', \Delta t, A) = \min_{v' \in \mathcal{V}'} \{\Pr(u, v', \Delta t, A)\} \tag{4}$$
- Case 3. $u \notin \mathcal{V}$ and $u' \in \mathcal{V}'$.
  This is the opposite of Case 2. In this case, we have only seen the value u' but not u. We estimate the probability of seeing a transition to u' as the prior probability of u' within Δt window:
  $$\Pr(u, u', \Delta t, A) = \frac{\sum\limits_{v \in \mathcal{V}} T_{\Delta t}[(v, u')]}{\sum\limits_{v' \in \mathcal{V}'} \sum\limits_{v \in \mathcal{V}} T_{\Delta t}[(v, v')]} \tag{5}$$
  The numerator gives the number of occurrences of u' within the Δt window while the denominator is the sum of all the counts in the transition table $T_{\Delta t}$.
- Case 4. $u \notin \mathcal{V}$ and $u' \notin \mathcal{V}'$.
  We have two possibilities under this case. If $u = u'$, this case reduces to the recurrence probability [5] of seeing the same value after Δt. Hence, we have:
  $$\Pr(u, u, \Delta t, A) = \frac{\sum\limits_{v \in \mathcal{V}} T_{\Delta t}[(v, v)]}{\sum\limits_{v' \in \mathcal{V}'} \sum\limits_{v \in \mathcal{V}} T_{\Delta t}[(v, v')]} \tag{6}$$
  If $u \neq u'$, we first obtain the expected number of occurrences where the attribute values are different. Let X be a random variable denoting the number of occurrences of each attribute value. We have
  $$\mathbb{E}(X) = \sum_{v \neq v'} Pr(v, v', \Delta t) \cdot T_{\Delta t}[(v, v')] \tag{7}$$
  Then the probability of seeing the transition (u, u') is the ratio of $\mathbb{E}(X)$ to the total number of such occurrences:
  $$\Pr(u, u', \Delta t, A) = \frac{\mathbb{E}(X)}{\sum\limits_{v \neq v'} T_{\Delta t}[(v, v')]} \tag{8}$$

Note that when u'=u, our transition model falls back to the recurrence probability [5]. However, there is a critical difference between our model and that of [5]. The model of [5] computes a "global" recurrence probability for an attribute A. In contrast, our model captures probabilities for different values of A. The ability to derive individual transition probabilities for different values is particularly important to properly model many real-world scenarios. For instance, the probability that the job title remains as "Full Professor" after 10 years is different from the probability that the

title remains as "Assistant Professor" after 10 years. Another example is that the probability that the immigration status remains as "citizen" after 10 years is different from the probability that the immigration status remains as "foreign worker" after 10 years.

Discussion. The proposed model learns the value transition probabilities based on the statistics of their occurrences in the entity profiles. Hence, before generating the transition tables, we first obtain the statistics on the frequency of the attribute values. For the attribute values that have low frequencies, we will utilize the general recurrence probabilities instead of calculating specific transition probabilities. On the other hand, if the percentage of distinct values for some attributes exceeds a certain threshold, that is, these attribute values are too specific and fine-grained, we will map them to values in a more general category (e.g., higher level in some taxonomic hierarchy) to avoid overfitting of the transition model. For example, instead of using the actual name of an organization, we can map the organization name to the name of the industry that this organization belongs to. Similarly, we can use the city of an address instead of the exact address. For numerical values, we can define buckets in ranges that are meaningful, such as the ranges of ages and salaries. The transition tables can then be computed as before but with an extra step to lookup the mapping table before we count the Δt-transitions.

4.2 A Freshness Model for Data Sources

We have presented a model that describes how an entity may change its attribute values over time in the previous section. In this section, we focus on evaluating whether a data source publishes fresh data, i.e., up-to-date information. If a source frequently publishes outdated attribute values, then we need to take this into consideration when matching its published records to an entity.

Before we discuss how one can measure the freshness of a data source on an attribute, we first introduce the notion of *delay* to reflect the currency of a particular attribute value in a temporal record.

Definition 3. Let v be a value of attribute A in a temporal record r with time attribute t, and let n be the entity that r refers to. If v is in the profile of the entity n, that is, $\text{Intervals}(\Phi_n[A], v) \neq \emptyset$, and t_{max} is the maximum time point in $\text{Intervals}(\Phi_n[A], v)$ that is less than $r.t$, then the *delay* η of this value v is given by:

$$\eta = \begin{cases} 0 & \text{if } r.t \text{ is among intervals of } \text{Intervals}(\Phi_n[A], v) \\ r.t - t_{max} & \text{otherwise} \end{cases} \tag{9}$$

Intuitively, if the value v of a record r occurs in the entity profile, and the timestamp $r.t$ falls in the intervals of v, then this value correctly reflects the status of the entity n at time $r.t$, and we set the delay to 0. Otherwise, since the existing entity profile is assumed to be correct, the value v in r is thus considered out-of-date. We then calculate the delay to be the difference between t_{max} and $r.t$.

Example 6. Consider record r_3 in Table 2, where the value of the attribute Title is "Engineer". We see from Table 1 that the intervals of this attribute for the entity David Brown is:

$\text{Intervals}(\Phi_{David}[\text{Title}], \text{Engineer}) = \{[2000, 2002]\}$.

Since the maximum time instant in this interval is 2002 and the time stamp of r_3 is 2004, the delay η of "Engineer" is thus 2. □

We obtain a distribution of delays for each source s on each attribute A as follows. Let $\mathcal{R}_s$ denote the set of records published by the data source s. For each attribute A of each record $r \in \mathcal{R}_s$, we compute the delay η of the value v in $r.A$ using Equation 9. In this way, we obtain a list of delay values for each attribute A. From this list, we compute the count of each delay value, and normalize the counts to between 0 and 1. Then the probability that a source s will publish a value for attribute A with a delay equals to η, denoted as $\text{Delay}(\eta, s, A)$, is given by the normalized count of η.

Note that in real world, the freshness of the data sources could be more complex: the update delays may vary from entity to entity; the freshness of a particular source can change over time. Although the proposed freshness model simplifies these cases, it shows promise in matching potential stale records in our experiments.

4.3 Matching Algorithm

We are now ready to describe how we utilize the value transition model and the source freshness model to link records to entity profiles and augment the corresponding entity profiles. We present a *source-aware temporal matching* algorithm that first finds potential states from the input records. This step takes into account the freshness of the data sources. Afterwards, the algorithm proceeds to identify the matching states for the target entity based on both the transition probabilities and the support of the sources, and finally updates the profile of the entity.

More concretely, given an (incomplete) entity profile Φ_n and a set $\mathcal{R}$ of temporal records from a set $\mathcal{S}$ of data sources, our matching algorithm outputs a set $\mathcal{R}' \subseteq \mathcal{R}$ of records where each record $r \in \mathcal{R}'$ matches the entity n. In addition, the profile Φ_n is also updated with records in $\mathcal{R}'$, where the updated entity profile contains a more complete and up-to-date history of the entity.

The matching algorithm consists two phases. The first phase takes the input records and reorganizes them into a set of clusters. In this phase, a stale record may be placed into multiple clusters depending on its probability of delay. The second phase iteratively matches the clusters to the entity profile and augments the profile with the best matched cluster. We describe the two phases next.

4.3.1 Phase I: Generate Clusters

Given a set $\mathcal{R}$ of temporal records, this phase outputs a set $\mathcal{C}$ of *clusters* where each cluster represents the state of some entity over some time period. Each cluster $c \in \mathcal{C}$ has an associated cluster signature Θ_c that describes the attribute values and the time interval of the cluster.

Definition 4. The *cluster signature* Θ_c of a cluster c is defined as follows. For each attribute $A \in \mathcal{A}$, the signature Θ_c contains a triplet $\langle A, V, \beta \rangle$, where β is the confidence that the cluster c has the set V of values on attribute A in some state. In addition, Θ_c also contains two time stamps t_{min} and t_{max}, where $[t_{min}, t_{max}]$ represents the interval of this cluster. We use V_c^A to denote the set V of values in triplet $\langle A, V, \beta \rangle \in \Theta_c$, and $\text{conf}(c, A)$ to denote β.

The goal of this phase is to identify different states of a target entity n that may be described by the set $\mathcal{R}$ of records. A prior solution to this problem was described in [6], where the algorithm clusters records with similar attribute values together. In each cluster, the earliest and the latest timestamps of the records in the cluster serve as the interval of the cluster. However, since a published record may be describing the state of an entity at a time point that is different from the timestamp of the record, and there may be different update delays for different attributes of a record, this method may obtain incorrect time intervals for the clusters, which would degrade the quality of the matching results. For example, recall record r_7 in Table 2. This record may reveal the latest location of David Brown, however, the job title is outdated since 10 years ago, as his clean profile in Table 1 shows that he last held the Engineer title in 2002. Thus the method in [6] may compute incorrect time intervals for the clusters.

To overcome the problem above, we first obtain an initial set of clusters based on the records that are published from sources with small delays. After this, records from sources that have larger delays are matched against these clusters. The matching process will take into account the distribution of update delays obtained in Section 4.2.

We say a source s is fresh if $\text{Delay}(0, s, A) > \mu$ for every attribute $A \in \mathcal{A}$, where μ is a predefined threshold. In other words, for every attribute in the record, the probability that s publishes up-to-date values (i.e., with 0 delay) is greater than μ. Let $\mathcal{R}_f \subseteq \mathcal{R}$ be the set of records from fresh sources. After we have determined the set $\mathcal{R}_f$, we apply a traditional record linkage technique (e.g., single pass clustering [13]) to place the records in $\mathcal{R}_f$ into a set $\mathcal{C}$ of clusters, where records in the same cluster refer to the same state of an entity.

Next, we form the signature of each cluster $c \in \mathcal{C}$. The timestamps t_{min} and t_{max} for c are set to the minimum and maximum timestamps of the records in c. Then for each attribute $A \in \mathcal{A}$, we obtain a triplet $\langle A, V, \beta \rangle$. To determine the set V of values for each attribute A, several data fusion techniques [8, 9, 19] have been proposed to integrate and resolve conflicting values. Here, we adopt a simple fusion method by taking the majority vote. Hence, V is the set of values that have the highest number of occurrences in c for attribute A. The confidence score β is initialized to 0.

After this, we process the records that originate from sources with larger delays (i.e., records in $\mathcal{R} \setminus \mathcal{R}_f$). As these records may contain obsolete attribute values, we check the clusters in $\mathcal{C}$ to see whether a record r may be describing the state represented by c due to its update delay. Formally, for each record $r \in \mathcal{R} \setminus \mathcal{R}_f$ and a cluster $c \in \mathcal{C}$ where $r.t \geq c.t_{min}$, we check if the following condition holds for an attribute $A \in \mathcal{A}$:

$$\text{Delay}(\max\{r.t - c.t_{max}, 0\}, r.s, A) > \mu' \tag{10}$$

where μ' is a predefined constant.

Recall that $\text{Delay}(\eta, r.s, A)$ gives the probability that the value on attribute A of r has a delay of η. Hence, if Equation 10 is true, then $r.A$ is highly likely to correspond to the time period of c. In this case, we will put r in c if the values $r.A$ and V_c^A are similar. After we have checked all the clusters in $\mathcal{C}$ for each attribute A, if there are some attributes or attribute values in r that are not captured by any cluster in $\mathcal{C}$, we create a new cluster for r and form its signature using these attribute values. Note that a record r may be placed in multiple clusters since different attributes of r could be describing values that are valid at different times due to different update delays.

Finally, we update the confidence score by the delay probabilities as follows:

$$\text{conf}(c, A) = \sum_{s \in \mathcal{S}_c} \sum_{r \in c \cap \mathcal{R}_s} \frac{\text{Delay}(\max\{r.t - c.t_{max}, 0\}, s, A)}{|c \cap \mathcal{R}_s|} \tag{11}$$

where $\mathcal{S}_c$ is the set of sources that have published records in c. Intuitively, $\text{conf}(c, A)$ depicts the number of sources that support the set of values in c for attribute A, weighted by the delay probabilities of its published records.

This cluster generation phase is described in Algorithm 2. We first obtain a set $\mathcal{C}$ of clusters from records published by fresh sources, and form the cluster signature for each cluster $c \in \mathcal{C}$ (lines 2-7). Then for each record r from the less fresher sources, we will attempt to match r to the existing clusters, or create a new cluster (lines 8-19). Finally, we set the confidence score for each attribute in each cluster.

Algorithm 2: Generate Clusters

input : Set $\mathcal{R}$ of temporal records
output: Set $\mathcal{C}$ of clusters

```
1  Let R_f ⊆ R be the set of records from fresh sources;
2  Generate a set C of clusters from R_f;
3  foreach c ∈ C do
4      c.t_min ← min_{r∈c}{r.t};
5      c.t_max ← max_{r∈c}{r.t};
6      foreach A ∈ A do
7          Let V is the set of values with the highest frequency;
           Create triplet ⟨A, V, 0⟩;
8  foreach r ∈ R \ R_f do
9      A' ← ∅;
10     foreach c ∈ C do
11         if r.t ≥ c.t_min then
12             foreach A ∈ A do
13                 if condition in Equation 10 is true then
14                     if c.A ≈ r.A then
15                         c ← c ∪ {r};
16                         A' ← A' ∪ {A};
17     if A \ A' ≠ ∅ then
18         Create a cluster c with the values on attributes A \ A';
19         C ← C ∪ {c};
20 foreach c ∈ C do
21     foreach A ∈ A do
22         Compute conf(c, A) using Equation 11;
```

Example 7. Let us consider the records in Table 2. Suppose the sources "Google+" and "Twitter" are fresh with small delays while "Facebook" is likely to publish outdated values on the attribute Title. We first cluster records from the fresh sources by directly comparing all their attribute values. We obtain 5 clusters in this way: c_1={r_1, r_2}, c_2={r_4}, c_3={r_5}, c_4={r_6}, c_5={r_8, r_9}.

Next, we process the remaining records r_3 and r_7. Observe that $c_1.t_{min} < r_3.t$ and suppose $\text{Delay}(2, \text{Facebook}, \text{Organization}) > \mu'$. Then we will place r_3 in c_1 since the Organization value in r_3 is the same as that in c_1's signature. The same condition holds for attribute Title. Similarly, we would place r_7 in c_1. However, we need to create a new cluster c_6 as the source "Facebook" is up-to-date for attribute Location and none of the existing clusters has the same Interests with r_7. Table 5 shows the clusters generated after this phase. □

4.3.2 *Phase II: Match and Augment Profile*

This phase takes as input an entity profile Φ_n and the set $\mathcal{C}$ of clusters generated by Algorithm 2 as described in the previous section. A cluster $c \in \mathcal{C}$ is linked to Φ_n to obtain a more complete profile if we are confident that c describes a future state of the entity n. This requires us to derive a match score between a cluster c and an entity profile Φ_n based on the transition model we developed in Section 4.1.

Recall the transition probability between a pair (v, v') of values defined in Equation 1. Based on this, we obtain the probability that a set V' of values for attribute A will occur Δt time after the set V of values has been seen. For each value $v' \in V'$, we take the maximum probability that a value $v \in V$ will change to v', and

Table 5: Clusters generated for Table 2

Cluster	Records	Signature for attributes	t_{min}	t_{max}
c_1	r_1, r_2 r_3, r_7	⟨Organization,{S3, XJek},1.6⟩ ⟨Title,{Engineer{,1.5⟩ ⟨Location,{Chicago},2.0⟩	2001	2002
c_2	r_4	⟨Title,{Manager},0.8⟩ ⟨Location,{Chicago},1.0⟩	2004	2004
c_3	r_5	⟨Organization,{Quest Software},1.0⟩ ⟨Title,{Director},1.0⟩ ⟨Interests,{Technology},1.0⟩	2011	2011
c_4	r_6	⟨Organization,{Quest Software},1.0⟩ ⟨Title,{IT Contractor},1.0⟩	2011	2011
c_5	r_8, r_9	⟨Organization,{WSO2},1.8⟩ ⟨Title,{President},1.8⟩ ⟨Location,{Chicago},2.0⟩ ⟨Interests,{Technology},2.0⟩	2013	2013
c_6	r_7	⟨Interests,{Sports,Politics},1.0⟩ ⟨Location,{Chicago},1.0⟩	2012	2012

return the average over all the values in V'. In other words,

$$\Pr(V, V', \Delta t, A) = \frac{1}{|V'|} \sum_{v' \in V'} \max_{v \in V} \{\Pr(v, v', \Delta t, A)\} \quad (12)$$

Note that the values in V and V' are valid within some time intervals. Suppose the time interval for V is I and that of V' is I'. We obtain the transition probability of V to V' by averaging $\Pr(V, V', \Delta t, A)$ over all possible Δt given the intervals I and I' as follows:

$$\overline{\Pr}(\mathrm{V}, \mathrm{V}', \mathrm{I}, \mathrm{I}', \mathrm{A}) = \frac{1}{|I||I'|} \sum_{t \in I} \left(\sum_{\substack{t' \in I' \\ t' \geqslant t}} \Pr(V, V', t' - t, A) + \sum_{\substack{t' \in I' \\ t' < t}} \Pr(V', V, t - t', A) \right) \quad (13)$$

With this, we can compute a score between an entity profile Φ_n and a cluster signature Θ_c on an attribute A based on the transition probability between the set V of values of the triples in $\Phi_n[A]$ and the values V_c^A in cluster c. Formally we have:

$$\mathrm{transitPr}(\Phi_n[A], c, A) = \frac{\sum_{<b,e,V> \in \Phi_n[A]} \overline{\Pr}(V, V_c^A, I, I', A)}{|\Phi_n[A]|} \quad (14)$$

where $I = [b, e]$ and $I' = [c.t_{min}, c.t_{max}]$.

Finally, we aggregate the transition probabilities for all the attributes in $\mathcal{A}$ and obtain the overall match score between an entity profile Φ_n and a cluster c as follows:

$$\mathrm{match}(\Phi_n, c) = \frac{1}{|\mathcal{A}|} \sum_{A \in \mathcal{A}} \mathrm{conf}(c, A) \times \mathrm{transitPr}(\Phi_n, c, A) \quad (15)$$

The first part $\mathrm{conf}(c, A)$ of the formula depicts the number of sources that support a state, while the second part of the formula captures how likely that an entity with history $\Phi_n[A]$ will have a state described by the cluster c. We will award clusters with high transition probabilities or are supported by more sources, and consider both as evidence of a true match.

We are now ready to present how this profile match and augmentation phase works. Given a target entity profile Φ_n, and the set $\mathcal{C}$ of clusters obtained in the first phase, we iteratively identify the cluster that best describes the entity n at different time periods and update the profile Φ_n.

For each cluster $c \in \mathcal{C}$, we calculate its match score with the entity profile Φ_n based on Equation 15. The cluster c' with the highest match score will be considered as a true match if the match score exceeds a certain threshold λ, and we update the profile Φ_n by inserting a triple $\langle c'.t_{min}, c'.t_{max}, V_c^A \rangle$ into the temporal sequence $\Phi_n[A]$.

Note that the values in a cluster c may conflict with the entity profile. This occurs for single-valued attributes. For example, a person cannot be at multiple locations at the same time point. In this case, we remove c from $\mathcal{C}$. That is to say, if a cluster $c \in \mathcal{C}$ overlaps with an entity profile at a time point but they contain different values for some single-valued attribute, thus violating some constraint that is known to hold, we remove c from $\mathcal{C}$.

We repeat the above steps to find the next cluster with the highest match score, update profile, and prune the conflicting clusters in $\mathcal{C}$. Note that the match score can be calculated incrementally based on the updated profile Φ_n since Equation 14 is the average over all triples. This process terminates when the set $\mathcal{C}$ is empty or none of the clusters has a match score higher than λ.

Algorithm 3 shows the details. We first initialize the set $\mathcal{R}'$ of matched records as empty. Lines 3-5 compute the match score for each cluster in $\mathcal{C}$ with the target profile Φ_n, and find the cluster c' with the highest match score. If the match score for c' exceeds some threshold λ, we add the records in c' into $\mathcal{R}'$, and remove c' from $\mathcal{C}$ (Lines 6-8). Lines 9-13 update the entity profile with c' and prune conflicting clusters. This process is repeated until $\mathcal{C} = \emptyset$ or the best match score is less than λ.

Note that the augmented entity profile may contain triples with overlapping intervals. In this case, we will initiate a post-processing step to sort the triples in profile and resolve overlapping intervals.

Algorithm 3: Match and Augment Profile

input : Entity profile Φ_n, set $\mathcal{C}$ of clusters, threshold λ
output: Updated entity profile Φ_n and the set $\mathcal{R}'$ of linked temporal records

```
1  R' ← ∅;
2  repeat
3      foreach c ∈ C do
4          Compute match(Φ_n, c) using Equation 15;
5      c' ← arg max_{c∈C} match(Φ_n, c);
6      if match(Φ_n, c') > λ then
7          Add records in c' into R';
8          C ← C \ {c'};
9          foreach A ∈ A do
10             Insert the triple ⟨c'.t_min, c'.t_max, V_{c'}^A⟩ into Φ_n[A];
11         foreach c ∈ C do
12             if c conflicts with c' then
13                 C ← C \ {c};
14     else
15         break;
16 until C = ∅;
```

Example 8. Back to our running example. Given the entity profile in Example 3 and the set of clusters depicted in Table 5, and suppose we have already matched two clusters c_1 and c_2 with

the profile. Hence we have $\mathcal{R}' = \{r_1, r_2, r_3, r_4, r_7\}$ and $\mathcal{C} = \{c_3, c_4, c_5, c_6\}$.

We use Equation 15 to calculate the match score between each cluster in $\mathcal{C}$ and the target profile. Suppose c_3 has the highest score and it exceeds the threshold λ. We will add r_5 to $\mathcal{R}'$ and update the profile with triples $\langle\{\text{QuestSoftware}\}, 2011, 2011\rangle$, $\langle\{\text{Director}\}, 2011, 2011\rangle$ and $\langle\{\text{Technology}\}, 2011, 2011\rangle$ for attributes Organization, Tile, and Interests respectively. After this, we remove c_3 from $\mathcal{C}$. In addition, we remove c_4 from $\mathcal{C}$ because c_4 conflicts with c_3. That is, David Brown cannot be both an IT Contractor and Director in 2011.

We repeat the above process to find the next cluster that has the highest match score with the updated profile, until no match can be found. Table 3 shows the updated profile obtained from this matching algorithm. □

5. PERFORMANCE STUDY

We now present the results of our experimental study to evaluate the performance of the MAROON framework.

5.1 Experimental Setting

Datasets. We use two real world datasets for the experiments.

- *DBLP data*. This is the DBLP-Ambi dataset used in [5]. It contains 2,664 records on 239 authors that share 21 unique names. After removing entities that have only 1 record, we obtain 216 entities with 2,641 records. Each record corresponds to a paper published at some time instant and has attributes Affiliation, Co-authors and Paper title. All these records are assumed to be clean and up-to-date. For each entity, we use the information provided by the first 5 records as the available clean entity profile. For entities that have less than 5 records, we use the first record as their clean profile. The records that have the same name with the entity are considered as input temporal records.
- *Recruitment data*. In our attempt to understand how MAROON will perform on larger datasets, we developed our own real world Recruitment dataset which has more than 40 times the number of entities compared to the DBLP-Ambi dataset. This dataset is obtained by crawling the public profiles of users from 3 popular social networks, i.e., LinkedIn[2], Google+[3], and Twitter[4]. We start with the top 200 Google+ users whose profile contain the keyword "Computer Science". Then we expand the user base by carrying out a breadth first search to crawl the friends of these 200 users. For each of these Google+ user, we find his/her accounts that may also exist in the other two websites using the "Links" information in the Google+ profile. Finally, we obtain 10,193 users with publicly available profiles on LinkedIn and at least one of the other two websites, and we call these users the *target entities*.

  We assume that the LinkedIn profiles of the target entities are the ground truth, that is, these are the complete and clean entity profiles. Given the lifespan of each target entity profile, we use the first 30% as the clean profile, and attempt to augment it with temporal records which are generated as follows.

  We use the name of each target entity to query all the 3 websites. For each result returned from Google+ or LinkedIn that contains the history of a user, we scan the history and construct a record whenever there is change in the value of some attribute. For the results returned from Twitter, we generate a record with timestamp equals to the latest active time of the account. The attributes we consider in our experiments are Organization, Title, and Location. Table 6 shows the statistics of the Recruitment dataset. For example, there are a total of 24,416 Google+ records that match the 10,193 target entities, and 86% of them provide up-to-date information.

[2] http://www.linkedin.com
[3] http://plus.google.com
[4] http://twitter.com

Table 6: Recruitment Dataset. Column 2 shows the number of temporal records while Column 3 shows the number of records that matches the profiles of target entities.

#Target entities = 10,193, Avg. lifespan=15 years

Source	#Records	#Matched	Period	Freshness
LinkedIn	196,075	49,789	1980-2014	1.00
Google+	80,694	24,416	1980-2014	0.86
Twitter	50,617	6,671	2006-2014	0.90
Total	327,386	80,876	1980-2014	-

Implementations. We implemented all the algorithms of MAROON in Python, and the experiments were conducted in a Windows 7 machine with 3.40 GHz Intel CPU and 8 GB of RAM.

In our implementation, we utilize the PARTITION method [13] to obtain the initial clusters of records. This is a traditional single-pass clustering algorithm for record linkage. It merges records based on attribute value similarities, and does not consider entity evolution or the freshness of the data sources.

We randomly select 50% of the clean entity profiles to learn the transition model. The standard TF-IDF metric is used to measure the similarity of set-valued attributes, while the similarity of a pair of values is given by Jaro-Winkler distance [7]. We set μ=0.9 and μ'=0.2 for MAROON.

Each experiment is repeated 3 times and we report the average performance.

5.2 Findings from Our Transition Model

We first report some interesting transition patterns learnt by our transition model.

DBLP data. Figure 3 shows the transition probabilities on Affiliation that are learnt from the DBLP dataset. Affiliations are categorized into either university or industry. For example, the probability that a person remains at the same university (red triangle line) has a large variance and has a decreasing trend over time. For a short time period (e.g., less than 5 years), people at the university (either graduate students or professors) are unlikely to change their affiliations. However, in the long run, the probability for people to remain in the same university is low. This is not surprising as few people work in the university where they graduated from and occasionally, faculty do leave a university for the industry.

Figure 3 reveals interesting trends for people in the industry. The probability that people from the industry will remain in the same company (black square line) decreases between years 1 through 7, and increases after 7 years. In contrast, the probability for a person to change company rises initially and decreases after 7 years (cyan diamond line), indicating that people may have become more stable in the later stage of their careers.

We also observe that if a person is currently affiliated with a university, then the probability that she moves to another university tends to increase over time (blue circle line). This probability is much higher than the probability that a person will move from a university to an industry (green star line). Furthermore, after about 10 years, the longer a person is at a university, the less likely the person is to move to an industry. This is consistent with our under-

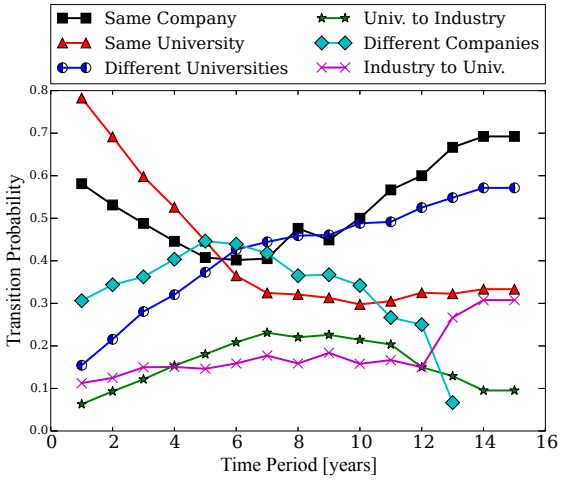


Figure 3: Transition probability for Affiliation

standing that it is rare for a senior faculty in a university to switch to the industry after 10 years.

Finally, we see that the probability a person will move from an industry to a university is generally low in the first 12 years and increases sharply after that (magenta cross line). While it is difficult to explain precisely why there is an increase after 12 years, this phenomena may reflect the desire for some people to have a more stable career which a university can provide.

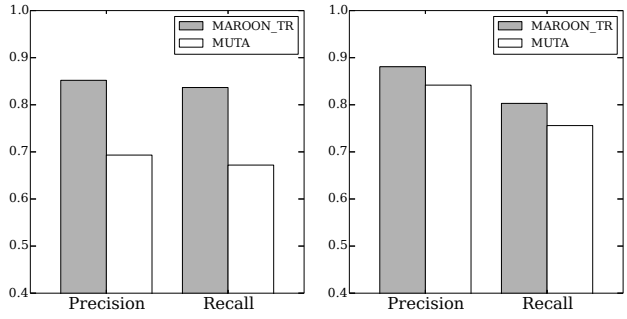
Recruitment data. Table 7 shows the transition probabilities for job title learnt from the Recruitment dataset. We observe that different job titles have different probabilities, and job titles that have more "seniority" tend to have a higher probability of remaining the same. For example, the probability that a person who is currently a Director and remains as a Director after 5 years is as twice as that of an Engineer. This table demonstrates the importance of learning the transition probabilities of different values. It also provides us with a more fine-grained understanding of the entity evolution behavior.

Table 7: Transition probability for Job Title

v	v'	probability at Δt			
		3	5	8	10
Engineer	Engineer	0.332	**0.201**	0.101	0.087
Engineer	Sr. Engr.	0.141	0.238	0.163	0.094
Engineer	Manager	0.034	0. 082	0.086	0.112
Manager	Manager	0.418	0.304	0.202	0.161
Manager	Director	0.049	0.061	0.093	0.134
Manager	Consultant	0.019	0.023	0.030	0.041
Director	Director	0.576	**0.410**	0.334	0.249
Director	CEO	0.059	0.090	0.106	0.125
Director	President	0.040	0.051	0.070	0.077

5.3 Performance of Temporal Model

We compare the proposed transition model MAROON_TR (described in Section 4.1) with MUTA [5]. This is the state-of-the-art temporal model that gives the probability that a value recurs. To ensure a fair comparison, we apply AFDS [6], the state-of-the-art temporal clustering algorithm, to match records to the profiles of the target entities. This clustering algorithm links clusters with the profile based on weighted attribute similarities, where the weight is



(a) Recruitment data (b) DBLP data

Figure 4: Comparison of temporal models

the probability provided by the temporal models. Note that AFDS does not consider source freshness when generating clusters.

To measure how well the temporal records are matched to target entities, we use the evaluation metrics *Precision* and *Recall*. Let $Match$ denote the set of records that match with a target entity, and let $Result$ denote the set of records returned by our algorithm. Then we have

$$Precision = \frac{|Match \cap Result|}{|Result|}$$

$$Recall = \frac{|Match \cap Result|}{|Match|}$$

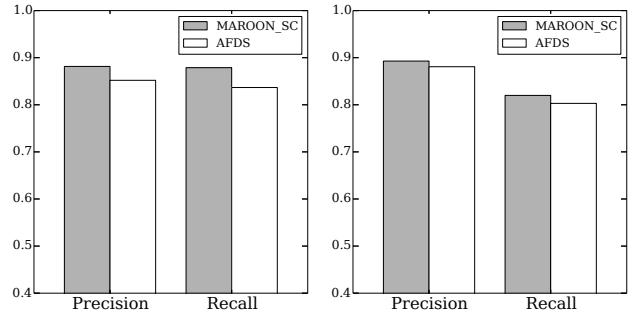
Figure 4(a) shows the performance of the two models on the Recruitment data. We observe that MAROON_TR is significantly better than MUTA on both *Precision* and *Recall*, demonstrating that considering only the recurrence probability over time is inadequate in general. As MAROON_TR models the transition probability between specific values, it is thus able to discriminate the various potential states described by the clusters. Hence, our MAROON_TR achieves significantly better precision and recall, which are at least 50% higher than MUTA.

Figure 4(b) shows the results for the DBLP data. Again, MAROON_TR outperforms MUTA. This is due to the fine-grained entity evolution behaviors captured by our transition model, as we have already illustrated in Figure 3. However, the difference narrows on this dataset as 50% of the entities never change affiliations. In contrast, in the Recruitment dataset, all the entities have at least one transition on Organization or Title and 80% of the entities change their organization and job title simultaneously.

5.4 Performance of Clustering Algorithm

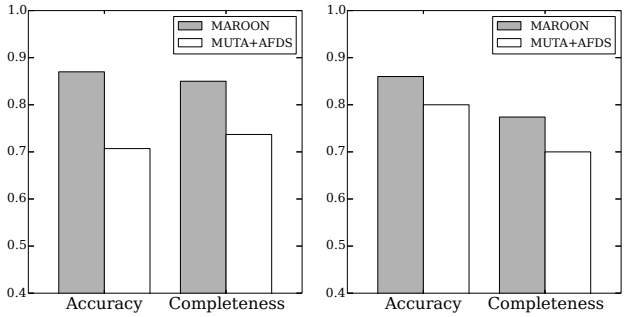
Next, we compare the performance of the proposed source-aware temporal clustering algorithm MAROON_SC (see Section 4.3) with AFDS [6]. Note that MAROON_SC considers source freshness when matching records. For a fair comparison, both methods use the proposed transition model to capture the entity evolutions.

Figure 5 shows the results of two clustering algorithms. We observe that MAROON_SC improves the precision and recall of AFDS for the Recruitment dataset as it incorporates the source delay when matching possibly stale records to the clusters that have been initialized by fresher records. Therefore, MAROON avoids generating incorrect intervals for the clusters. Further, it rewards the clusters that are supported by more sources in the match score calculation, thereby increasing the chance of linking the right records to the entity profiles.



(a) Recruitment data (b) DBLP data

Figure 5: Comparison of clustering methods



(a) Recruitment data (b) DBLP data

Figure 6: Results on profile augmentation

For the DBLP dataset, although the records all come from a single source, MAROON_SC is still able to improve the performance of AFDS. Furthermore, we shall show in Section 5.6 that MAROON_SC takes less time overall.

5.5 Effectiveness of MAROON

In this set of experiments, we study the effectiveness of the proposed MAROON framework to augment entity profiles in terms of two quality metrics: *Accuracy* and *Completeness*.

Recall that we have already obtained a clean and complete profile for each target entity. Let GT_n be the complete profile (ground truth) for target entity n, and $Result_n$ be the augmented profile constructed by the algorithms. Then we have

$$Accuracy = \frac{|GT_n \cap Result_n|}{|Result_n|}$$

$$Completeness = \frac{|GT_n \cap Result_n|}{|GT_n|}$$

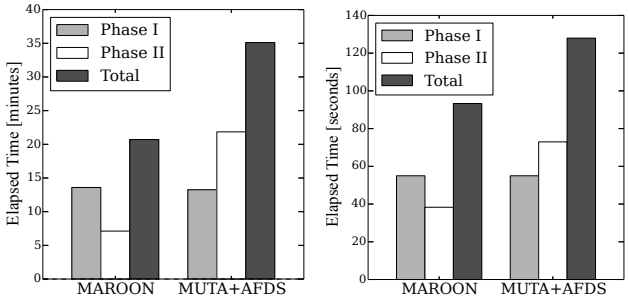
We compare MAROON with MUTA+AFDS. For AFDS, we construct the corresponding entity profile based on the set of records in the cluster signature of entity n. More precisely, we sort the records in increasing time order, and for any consecutive pair (r_1, r_2) of records, we insert a triple $\langle r_1.A, r_1.t, r_2.t - 1\rangle$ into the profile sequence $\Phi_n[A]$.

Figure 6(a) shows the results for the Recruitment dataset. We observe that MAROON performs much better than MUTA+AFDS, by 45% on accuracy and 36% on completeness. This is because our transition model is able to link the right clusters to the target entity. As the algorithm iteratively updates the entity profile with the next best-matched cluster, the profile becomes more complete, which in turn, leads to more accurate linkage decisions. At the same time, the source-aware clustering algorithm reduces the incorrect intervals generated from the obsolete values which further improves the accuracy of the profiles obtained.

The results for the DBLP dataset are shown in Figure 6(b). We see that MAROON performs better than MUTA+AFDS. Again, the margin by which MAROON outperforms MUTA+AFDS is less on the DBLP dataset because the Recruitment dataset is more "diverse" than the DBLP dataset, where a large proportion of the entities do not change affiliations.

5.6 Efficiency of MAROON

Finally, we compare the runtime of MAROON and MUTA+AFDS. Figure 7 shows the results. MAROON and MUTA+AFDS take similar amount of time in the first phase since they both group records into clusters by comparing the attribute values. However, MAROON



(a) Recruitment data (b) DBLP data

Figure 7: Comparison of running time

is more efficient on the second phase. This is because MAROON makes matching decisions based on the transition probability, while MUTA+AFDS calculates a weighted attribute similarity, which is computationally more expensive. Although MAROON has an iterative process in the second phase, the incremental calculation of Equation 14 and the pruning of conflicting clusters are effective in limiting the overhead incurred. As a result, the total runtime of MAROON is still lower than that of MUTA+AFDS.

6. CONCLUSION AND FUTURE WORK

In this paper, we have addressed the problem of linking temporal records from different sources in order to build a complete and up-to-date history for a real world entity. We have proposed a novel transition model that captures the probability that an entity may change to a specific attribute value after some time period. This model provides fine-grained understanding of how two records with distinct values may be temporally related. We have also designed a matching algorithm that jointly considers the value transition probability and the freshness of the data sources to link temporal records to the entities in the right time period. Extensive experiment results show that the proposed approach significantly outperforms the state-of-the-art techniques.

For future work, we plan to enhance MAROON to work with data sources that could have more complex characteristics, such as varying update delays across entities within the same source, varying quality of a data source over time [20], and possible erroneous values contained in the records. In addition, the correlation of attributes can also be exploited to develop more sophisticated temporal models.

Acknowledgements. Tan is partially supported by NSF grant IIS-1450560.

7. REFERENCES

[1] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD*, 2010.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data the semantic web. *The Semantic Web*, pages 722–735, 2007.

[3] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, 2003.

[4] D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. Stanoi, S. Vaithyanathan, and S. R. Das. Extracting, linking and integrating data from public sources: A financial case study. *IEEE Data Engineering Bulletin*, pages 60–67, 2011.

[5] Y.-H. Chiang, A. Doan, and J. F. Naughton. Modeling entity evolution for temporal record matching. In *SIGMOD*, 2014.

[6] Y.-H. Chiang, A. Doan, and J. F. Naughton. Tracking entities in the dynamic world: A fast algorithm for matching temporal records. *PVLDB*, pages 469–480, 2014.

[7] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, 2003.

[8] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, pages 550–561, 2009.

[9] X. L. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. *PVLDB*, pages 562–573, 2009.

[10] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, pages 1–16, 2007.

[11] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma. Dynamic constraints for record matching. *The VLDB Journal*, pages 495–520, 2011.

[12] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice and open challenges. *PVLDB*, pages 2018–2019, 2012.

[13] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB*, pages 1282–1293, 2009.

[14] Instant checkmate. `http://www.instantcheckmate.com`.

[15] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, pages 484–493, 2010.

[16] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, 2006.

[17] F. Li, M. L. Lee, and W. Hsu. Entity profiling with varying source reliabilities. In *KDD*, 2014.

[18] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *PVLDB*, pages 956–967, 2011.

[19] A. Pal, V. Rastogi, A. Machanavajjhala, and P. Bohannon. Information integration over time in unreliable and uncertain environments. In *WWW*, 2012.

[20] T. Rekatsinas, X. L. Dong, and D. Srivastava. Characterizing and selecting fresh data sources. In *SIGMOD*, 2014.

[21] W. Shen, P. DeRose, L. Vu, A. Doan, and R. Ramakrishnan. Source-aware entity matching: A compositional approach. In *ICDE*, 2007.

[22] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

[23] S. E. Whang, O. Benjelloun, and H. Garcia-Molina. Generic entity resolution with negative rules. *The VLDB Journal*, pages 1261–1277, 2009.