Truth Discovery and Copying Detection in a Dynamic World

Xin Luna Dong
AT&T Labs–Research
lunadong@research.att.com

Laure Berti-Equille
Université de Rennes 1
berti@irisa.fr

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

ABSTRACT

Modern information management applications often require integrating data from a variety of data sources, some of which may copy or buy data from other sources. When these data sources model a dynamically changing world (*e.g.*, people's contact information changes over time, restaurants open and go out of business, etc.), sources often provide out-of-date data. Errors can also creep into data when sources are updated often. Given out-of-date and erroneous data provided by different, possibly dependent, sources, it is challenging for data integration systems to provide the true values. Straightforward ways to resolve such inconsistencies (*e.g.*, voting) may lead to noisy results, often with detrimental consequences.

In this paper, we study the problem of finding true values and determining the copying relationship between sources, when the update history of the sources is known. We model the quality of sources over time by their *coverage, exactness* and *freshness*. Based on these measures, we conduct a probabilistic analysis. First, we develop a Hidden Markov Model that decides whether a source is a copier of another source and identifies the specific moments at which it copies. Second, we develop a Bayesian model that aggregates information from the sources to decide the true value for a data item, and the evolution of the true values over time. Experimental results on both real-world and synthetic data show high accuracy and scalability of our techniques.

1. INTRODUCTION

Modern information management applications often require integrating data from a variety of data sources. Among these sources, some may cite others (often without proper attribution on the web), crawl or aggregate data from others (*e.g.*, Google), exchange data with or buy data from other sources [1]. Sources often provide out-of-date and erroneous data, and such data can be propagated by copiers. Resolving conflicts in data from different sources and determining the true values is critical for improving quality of integrated data. Recent work on this topic focuses on resolving conflicts from a snapshot of data [5, 13]. However, the real world is dynamically changing (*e.g.*, people's contact information changes over time, restaurants open and go out of business, etc.), and sources often frequently update their data to capture the changes. Such evolution presents new challenges to truth discovery.

VLDB '09, August 24-28, 2009, Lyon, France

Table 1: Sources in the motivating example. We also show the number of the out-of-business restaurants that Google Maps lists.

Source	Coverage	Exactness	Freshness	#Closed-rest
MenuPages	.66	.98	.86	29
TasteSpace	.44	.97	.3	106
NYMagazine	.43	.98	.54	59
NYTimes	.43	.98	.38	72
ActiveDiner	.41	.95	.86	70
TimeOut	.38	.99	.68	33
SavoryCities	.27	.99	.41	33
VillageVoice	.22	.94	.4	37
FoodBuzz	.18	.92	.3	59
NewYork	.13	.92	.45	28
OpenTable	.12	.92	.45	9
DiningGuide	.1	.9	.09	48
GoogleMaps	-	-	-	212

First, true values can evolve over time and in many applications we are interested in the whole history or a fragment of the history of true values for particular items (*e.g.*, a person's addresses in the past five years, the history of a customer's billing information, and the previous chairs of an organization). However, errors can creep into data and data can go out of date. It is challenging to determine which values were once true and in which periods they were true.

Second, sources are often of different quality and a natural thought is to take this into consideration when we decide true values. However, low-quality data can be caused by many reasons: some sources make a lot of errors in their provided data; some provide correct data but fail to update according to later changes; and some, though they update, do so slowly. All these reasons can lead to a low accuracy of a snapshot of data and we should treat them differently.

Third, a source may copy data from other sources and often copy erroneous and out-of-date data unknowingly. Straightforward ways to resolve conflicts (*e.g.*, voting) fall short in presence of copying. In addition, the copying relationship can evolve over time as well: a source can stop copying and become independent, can change sources from which it copies, and can copy at some times and provide data independently at other times. These can make copying detection extremely tricky.

EXAMPLE 1.1. *We collected data on Manhattan restaurants from 12 web sources (listed in Table 1) weekly from 1/2009 to 3/2009 and examined opening and closing of restaurants. There are 5149 restaurants mentioned by at least 2 data sources and among them we found that 248 went out of business recently.*

We decided life period of each of the 5149 restaurants from the data and the copying relationship between the sources (shown in Figure 1). Accordingly, we computed for each source its coverage (how many existing restaurants are provided and how many closed restaurants are removed), exactness (how many updates are

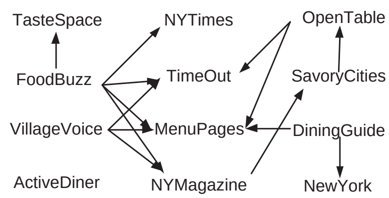

Figure 1: Copying relationship we discovered between sources in the motivating example. An arrow from one source to another indicates the former is a copier of the latter.

correct at the time of being made), freshness (how quickly sources capture changes and update), and the number of closed restaurants they still provide in their lists (shown in Table 1)[1]*. We observe that sources do provide stale data, their quality measures vary highly, and some do copy from others. In particular, we found that source* FoodBuzz*, which may be an aggregator, seems to have copied from several other sources, including some out-of-date listings, and accordingly has a lower exactness.* □

In this paper, we examine the update history of sources and study how to decide the evolving copying relationship between sources and the evolving true values. Our first contribution is to propose several quality measures of data sources, which play a key role in our probabilistic analysis. These measures include *coverage*–how many values in the history a source covers, *exactness*–how many updates conform to the reality, and *freshness*–how quickly a source captures a new value (Section 2). Note that these measures are orthogonal and all contribute to the accuracy of the latest version of the data, because a low accuracy of current data can be due to either a low exactness of provided data (erroneous data), or a low coverage or freshness for capturing recent transitions (outdated data).

Our second contribution is a set of Hidden Markov Models (HMM) that decide whether a source copies from another source and at which moment it copies (Section 3). Our models consider not only whether two sources share similar update history while one often updates later than the other, but also the coverage, freshness and exactness of the sources, to avoid identifying slow updaters as copiers. In addition, although the copying relationship between a pair of sources can evolve over time, frequent radical change is less likely; in other words, a copier is more likely to remain as a copier. Our HMM models capture this intuition and so are able to make more accurate decisions both on the copying relationship and on when the copying is conducted.

Third, we develop a Bayesian model to decide when the true value for a particular data item changes and what the new value is (Section 4). Our model considers both source quality and data copying, and so is less affected by possible wrong updates, stale data, and copied data. In addition, we consider different publish patterns, including instant publishing and delayed batch-mode publishing (Section 5).

We describe experimental results on both real-world data and synthetic data, showing that our models are accurate and scalable both in detection of the evolving copying relationship and discovery of transitions of true values (Section 6).

We note that although we propose our techniques in the framework of truth discovery, our techniques for detecting copying and evaluating source quality are of independent interest in a variety of data-integration tasks, including source recommendation, plagiarism detection, query optimization in an online query-answering system, and so on.

[1]We describe the data set, the measures, and our techniques in detail in the rest of this paper. As we show in Section 6, we have evidence to support some copyings we discovered.

2. OVERVIEW

This section formally defines the problem we solve in this paper and defines quality measures of data sources.

2.1 Problem definition

Let $\mathcal{O}$ be a set of objects. Each object $O \in \mathcal{O}$ is associated with a value at a particular time t and can be associated with different values at different times; if O does not exist at t, we consider it associated with a special value $\perp$. Formally, we define the *life span* of O as a sequence of *transitions* $(tr_1, v_1), \dots, (tr_l, v_l)$, where (1) l is the number of periods in O's life time; (2) the value of O changes to v_i at time $tr_i, i \in [1, l]$; (3) $v_1 \neq \perp$, and $v_i \neq v_{i+1}$ for each $i \in [1, l-1]$; and (4) $tr_1 < tr_2 < \dots < tr_l$. We denote by $\odot$ the beginning time we are interested in and $tr_1 = \odot$ if an object already exists at that time. In our paper we focus on atomic categorical values; we can treat set (or list) of atomic values as a whole and adapt techniques in [13] for value similarity.

Let $\mathcal{S}$ be a set of structured data sources. Each source $S \in \mathcal{S}$ can (but not necessarily) provide a value for an object at a particular time, and when the value of the object evolves, S can change the value accordingly. We observe data provided by the sources at different times; by comparing an observation with its previous observation, we can infer recent *updates*. Formally, we denote by $\mathbf{T} = \{t_0, \dots, t_n\}$ the set of observation points and by $\bar{U}(S, t_i), i \in [0, n]$, the updates we infer at time t_i; as a special case, $\bar{U}(S, t_0)$ contains values S provides at t_0[2]. Note that an update can happen at any time in $(t_{i-1}, t_i]$ and we may miss updates that are later overwritten. Our techniques can be adapted to the case where we know exact timestamps of each update.

We classify data sources into *independent* ones and *copiers*. An independent source updates according to its own observation of the real world. A copier can copy from one or more other sources. When a copier copies, it may copy only a subset of updates and may meanwhile observe the real world independently and conduct updates accordingly (validating or modifying a copied value is also considered as independent updating). A copier may not copy all the time: it can copy at some times and update independently at other times. On the other hand, a copier can stop copying from a particular source and vice versa. Note that the case of a source being independent and the case of a source being a copier but not copying at a particular moment are conceptually different, but not distinguishable from behavior of the source at that moment.

For now we consider *instant publishing*–publishing a value right after it is observed (from the real world or from another source); in other words, the published updates conform to the observation at the point of publishing (though the observation may be false and do not conform to the reality). We consider other publish patterns in Section 5.

EXAMPLE 2.1. *Consider the (synthetic) data sources in Table 2. They provide information on affiliations of five database researchers–* Stonebraker(S), Dewitt(D), Bernstein(B), Carey(C), Halevy(H)*, and we observe their data each year since 2000. Among the five sources, S_1 and S_2 are independent; S_3 was once a copier of S_2 and then changed to be a copier of S_1 since 2006 (despite difference of their latest data); S_4 is a copier of S_1; S_5 is a copier of S_3 but copies only periodically.* □

The goal of our research is to determine the evolving copying relationship between sources and the evolving true values of objects. Formally, we decide

[2]We assume a source starts providing data before t_0, but our techniques can be easily adapted for the opposite case.

Table 2: Researcher affiliation example. We emphasize the last update on each object by each source.

Life span	S_1	S_2	S_3	S_4	S_5
S ($\odot$, UCB) (02, MIT)	(03, **MIT**)	(00, **UCB**)	(01, UCB) (06, **MIT**)	(05, **MIT**)	(03, UCB) (05, **MS**)
D ($\odot$, Wisc) (08, MSR)	(00, Wisc) (09, **MSR**)	(00, UW) (01, Wisc) (08, **MSR**)	(01, UW) (02, **Wisc**)	(05, **Wisc**)	(03, UW) (05, $\perp$) (07, **Wisc**)
B ($\odot$, MSR)	(00, **MSR**)	(00, **MSR**)	(01, **MSR**)	(07, **MSR**)	(03, **MSR**)
C ($\odot$, Propel) (02, BEA) (08, UCI)	(04, BEA) (09, **UCI**)	(05, **IBM**)	(06, **BEA**)	(07, **BEA**)	(07, **BEA**)
H ($\odot$, UW) (05, Google)	(00, UW) (07, **Google**)	(00, Wisc) (02, UW) (05, **Google**)	(01, Wisc) (06, **UW**)	(05, **UW**)	(03, Wisc) (05, Google) (07, **UW**)

1. *copying:* for every $S_1, S_2 \in \mathcal{S}$ and $t \in \mathbf{T}$, the probability that S_1 is a copier of S_2 at t and if so, the probability of S_1 actively copying from S_2 at t;
2. *life span:* for every object $O \in \mathcal{O}$ and $t \in \mathbf{T}$, the true value (including $\perp$) of O at t.

In this paper we do not consider simultaneous cyclic copying, which happens rarely in practice.

2.2 Quality of data sources

Before we present our solutions to copying detection and life span discovery, we first introduce three quality measures of data sources, namely, *coverage*, *exactness*, and *freshness*, as a whole referred to as the *CEF-measure*, on which we rely heavily in our probabilistic analysis.

Consider a source $S \in \mathcal{S}$. Intuitively, its coverage measures the percentage of all transitions of different objects that it captures; its (in)exactness measures the percentage of all transitions it mis-captures (by providing a wrong value); its freshness measures how quickly it captures the transitions. Source S is high-coverage, exact and fresh, if it provides a new value for an object if and only if, and right after, the true value of the object evolves to that value. The definition of CEF-measure relies on two notions, *capture* and *mis-capture*, which we define next.

An update of S on a particular object O can be triggered either by a transition of O (to reflect the value change) or by a previous update of S (to fix a previous error). Thus, we consider all transition points of O and update points of S on O and sort them in ascending order. These points divide the whole observation period into a set of *slices*. The real value of O in each slice is the real value at the beginning of the slice. As an example, Figure 2 depicts the life span of *Dewitt*'s affiliation and updates by S_5 on it; we divide the whole observation period into 5 slices.

A slice is *capturable* if at its beginning, the value provided by S is different from the real value. A capturable slice is *captured* if it ends with an update of S to the real value. A slice is *mis-capturable* if S can update to a wrong value; when there are more than two values in the domain of O, each slice is mis-capturable. A mis-capturable slice is *mis-captured* if it ends with an update of S to a wrong value[3]. Thus, a slice that does not end with an update is neither captured nor mis-captured. In Figure 2, all 5 slices are mis-capturable, and L_1, L_2, L_3, and L_5 are capturable. Among them, L_1 and L_2 are mis-captured, and L_3 is captured.

[3]In the rest of the paper, when a transition and an update on the same object occur between two consecutive observations, in absence of knowledge of which happens earlier, we treat the update as correct once its value conforms to the value of the new transition or that of the previous transition.

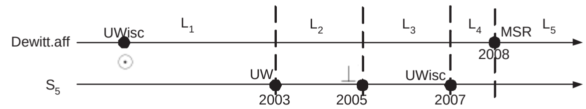

Figure 2: Computing CEF measure for S_5.

We denote by $cl(S,O), c(S,O), ml(S,O)$, and $m(S,O)$, respectively, the number of capturable, captured, mis-capturable, and mis-captured slices for S on O. We define the coverage of S, denoted by $C(S)$, as

$$C(S) = \frac{\sum_{O \in \mathcal{O}} c(S,O)}{\sum_{O \in \mathcal{O}} cl(S,O)}. \tag{1}$$

We define the exactness of S, denoted by $E(S)$, as

$$E(S) = 1 - \frac{\sum_{O \in \mathcal{O}} m(S,O)}{\sum_{O \in \mathcal{O}} ml(S,O)}. \tag{2}$$

We define freshness of S by a distribution function of length of the captured slices. If we denote by $c(S,\Delta), \Delta \geqslant 0$, the number of captured slices with length no larger than Δ, the freshness function of S, denoted by $F(S,\Delta)$, can be computed as follows (thus, $F(S,+\infty) = 1$).

$$F(S,\Delta) = \frac{c(S,\Delta)}{c(S)}. \tag{3}$$

Note that the three different measures are orthogonal and all contribute to the accuracy of data provided by the source at any moment: low exactness causes erroneous data whereas low coverage or freshness causes out-of-date data.

In practice, it is often easier to capture a long slice than a short one, and easier to make a mistake during a long slice than a short one. We can thus compute the weighted measure, where the weight of a capturable slice is proportional to its length and that of a mis-capturable slice is inversely proportional to its length. Our experimental results show higher accuracy with weighted CEF-measure.

3. DISCOVERING COPYING OF SOURCES

This section describes how we discover copying between data sources. As we need to reason about update pattern over time, a natural choice is to use a Hidden Markov Model (we compare with other options and validate its advantage in experiments (Figure 16)). We start from a review of the HMM model (Section 3.1), then describe a basic HMM model for copying discovery (Section 3.2), and next extend it for periodical copying (Section 3.3). This section assumes knowledge of the life span of each object and we present how we compute it in Section 4.

3.1 Review of HMM

A Hidden Markov Model (HMM) [11] contains a set of *hidden states* $\mathbf{H} = \{h_1, h_2, \ldots, h_N\}, N > 1$, and a set of observation symbols $\mathbf{O} = \{o_1, o_2, \ldots, o_M\}, M > 1$. At each time t, the model is in a particular hidden state $q_t \in \mathbf{H}$ and we observe a particular observation symbol $e_t \in \mathbf{O}$. An HMM $\lambda = (A, B, \pi)$ contains three components:

- the state transition probabilities, $A_{N,N} = \{a_{ij} | i, j \in [1, N]\}$, where $a_{ij} = P(q_{t+1} = h_j | q_t = h_i)$ is the probability that the model transits from state h_i to h_j;
- the observation probability distribution in each state, $B_{N,M} = \{b_{ij} | i \in [1, N], j \in [1, M]\}$, where $b_{ij} = P(e_t = o_j | q_t = h_i)$ is the probability of observing o_j in state h_i;
- the initial state distribution, $\pi_N = \{\pi_i | i \in [1, N]\}$, where $\pi_i = P(q_1 = h_i)$ is the initial probability of state S_i.

Given a sequence of observations, we can apply *Forward-backward* inference to decide the probability of each state at each time point. In addition, by applying *Baul-Welch* learning, we can decide the parameters in A, B and π from a set of observation sequences [11].

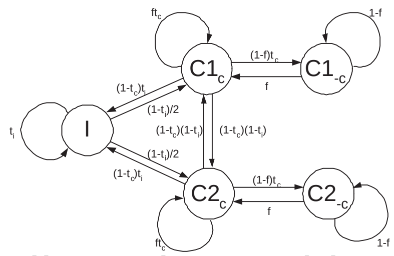

Figure 3: Hidden states and transitions in the basic HMM model.

3.2 The basic HMM

Let S_1 and S_2 be two data sources. We decide if one of them copies from the other. We apply an HMM, where the hidden states correspond to whether S_1 or S_2 is copying at a particular moment and the observations correspond to their updates. We next describe our model in detail.

3.2.1 Hidden states

We first decide the set of hidden states. As we assume acyclic copying, at each moment there can be at most one copier between S_1 and S_2. In case a particular source is a copier, it can copy or independently update at a particular observation point. Thus, there are five hidden states: $I, C1_c, C1_{\neg c}, C2_c$, and $C2_{\neg c}$. State I represents independence of S_1 and S_2. States $C1_c$ and $C1_{\neg c}$ represent that S_1 is a copier of S_2, while the former represents S_1 actively copying from S_2 and the latter represents S_1 not copying at that moment. Similarly, states $C2_c$ and $C2_{\neg c}$ represent S_2 copying or not copying while being a copier of S_1.

Note that $C1_{\neg c}$, $C2_{\neg c}$, and I are not distinguishable from the action of S_1 and S_2: in all cases S_1 and S_2 make independent updates. One may suggest we merge them into the same state; however, as we show later, the probability of transition from one of these states to state $C1_c$ (or $C2_c$) can be different: intuitively, S_1 is more likely to actively copy from S_2 (so in state $C1_c$) next when it is in state $C1_{\neg c}$ than in state I. To avoid ambiguity, we disallow transition between $C1_{\neg c}$ and any of $I, C2_c$, and $C2_{\neg c}$ (similar for $C2_{\neg c}$); thus, the period of S_1 being a copier starts from and ends after a real copying. Figure 3 shows the transition graph.

3.2.2 Initial and transition probabilities

We now consider how to assign initial and transition probabilities for the states. Note that many transitions include the same behavior, such as transformation between copiers and being independent; thus, instead of having different transition probabilities for the 15 possible transitions, we can compute them using only a few parameters, as we describe next.

Since the period of a source being a copier starts with a real copying, the initial state cannot be $C1_{\neg c}$ or $C2_{\neg c}$. Assume the a-priori probability of two sources being independent is α (parameters can be learned from real data by *Baul-Welth* learning). Then, we have initial probabilities as

$$P(I) = \alpha, \tag{4}$$

$$P(C1_c) = P(C2_c) = \frac{1-\alpha}{2}. \tag{5}$$

We next define three parameters that we use to compute probabilities of transitions between states.

- $f (0 < f \leqslant 1)$: the probability that a copier copies at a particular time point.
- $t_c (0 \leqslant t_c \leqslant 1)$: the probability that between a pair of sources, a copier remains as a copier of the other source. Intuitively, this is more likely to happen than transformation to independence, so typically $t_c > .5$.

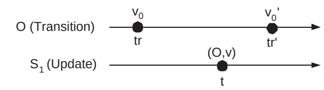

Figure 4: An update and its previous transition.

- $t_i (0 \leqslant t_i \leqslant 1)$: the probability that a pair of independent sources remain as independent. Typically, $t_i > .5$.

The probabilities of transitions are computed as follows. For convenience, we denote by $T_{h,h'}$ the transition from state h to h' and by $a_{h,h'}$ its probability.

- Transition $T_{I,I}$ happens when S_1 and S_2 remain as independent, so has probability t_i. Transitions T_{I,S_1} and T_{I,S_2} have the same probability, $\frac{1-t_i}{2}$.
- When S_1 is a copier of S_2, it transforms to be independent with probability $1 - t_c$. Once this happens, S_1 and S_2 become independent and remain so with probability t_i; otherwise, S_2 becomes a copier of S_1. Thus, $a_{C1_c,I} = (1-t_c) \cdot t_i$ and $a_{C1_c,C2_c} = (1-t_c)(1-t_i)$ (similar for $C2_c$).
- Once S_1 remains as a copier of S_2, it copies at a particular moment with probability f. At state $C1_c$, S_1 remains as a copier with probability t_c, so $a_{C1_c,C1_c} = f \cdot t_c$ and $a_{C1_c,C1_{\neg c}} = (1-f) \cdot t_c$. At state $C1_{\neg c}$, S_1 has to remain as a copier, so $a_{C1_{\neg c},C1_c} = f$ and $a_{C1_{\neg c},C1_{\neg c}} = 1 - f$ (similar for $C2_c$ and $C2_{\neg c}$).

3.2.3 Observation probability distribution

Now we consider the probability of S_1 and S_2 making particular sets of updates in a state. There are a huge number of possible updates for each source at each moment; enumerating them and assigning a probability for each is infeasible. Instead, we describe equations for computing the probability of a particular observation.

We focus our attention on three types of updates at each particular point: those made by S_1 and recently (we define "recently" shortly) by S_2, denoted by $\bar{U}_{S_1,S_2}$; those made by S_2 before, but not by S_1, denoted by $\bar{U}_{\neg S_1,S_2}$; and those made by S_1 but not recently by S_2, denoted by $\bar{U}_{S_1,\neg S_2}$. Typically, the more updates in $\bar{U}_{S_1,S_2}$, the more likely that S_1 is copying from S_2; the more updates in $\bar{U}_{\neg S_1,S_2}$ and $\bar{U}_{S_1,\neg S_2}$, the less likely that S_1 is copying. Note that we do not consider updates performed neither by S_1 and S_2, both because enumerating them is often infeasible, and because the set of updates that "should" be performed depends on previous updates and so varies for sources. We denote by Ω_1 the distribution of $\bar{U}_{S_1,S_2}, \bar{U}_{S_1,\neg S_2}, \bar{U}_{\neg S_1,S_2}$ at a particular moment and define Ω_2 for S_2 similarly (note that $\bar{U}_{S_1,S_2}$ and $\bar{U}_{S_2,S_1}$ can be different, similar for other pairs of sets). We summarize observations at each point using Ω_1 and Ω_2.

Intuitively, the fact that S_1 always follows updates of S_2 can ring an alarm in copying detection. However, this fact in itself does not necessarily imply S_1 being a copier, as S_1 might just be a slow updater (has low freshness). Source S_1 is more likely to be a copier of S_2 if in addition one of the following holds: (1) S_1 and S_2 have only low to medium coverage but their updates highly overlap in a close time frame; (2) S_1 and S_2 make a lot of common mistakes (*e.g.*, source S_2 and S_3 in Example 2.1); (3) the overlapped updates are performed by S_1 after the real values have already changed (*e.g.*, source S_3 and S_5's updates on *Halevy*'s affiliation since 2005 in Example 2.1). These three cases are more suspectable because they are low-probability events if S_1 and S_2 are independent. We next examine the probability of an update by a source conditioned on the source independently updating or copying.

We first consider the case where S_1 is independently updating, denoted by $S_1 \not\to S_2$, and compute the probability that S_1 makes an update U at time t. Assume U updates the value of O to v and the last transition on O by time t is (tr, v_0) (Figure 4). If $v = v_0$, the update is correct–S_1 does not make a mistake and captures the correct value within time $t - tr$, so the probability is

$$P(U, S_1, t|S_1 \not\to S_2, U \text{ true}) = E(S_1)C(S_1)F(S_1, t - tr). \quad (6)$$

If instead, $v \neq v_0$, S makes a mistake. Let m be the number of wrong values in the domain. Not assuming a-priori knowledge on which wrong values are more likely to be provided, we have

$$P(U, S_1, t|S_1 \not\to S_2, U \text{ false}) = \frac{1 - E(S)}{m}. \quad (7)$$

We denote the probability of S_1 performing U by $P(U)$. According to if U is correct or not, we apply Equation (6) or (7) to compute $P(U)$. Obviously,

$$P(U \in \bar{U}_{S_1,S_2} \cup \bar{U}_{S_1,\neg S_2}|S_1 \not\to S_2) = P(U); \quad (8)$$

$$P(U \in \bar{U}_{\neg S_1,S_2}|S_1 \not\to S_2) = 1 - P(U). \quad (9)$$

We next consider the case where S_1 is copying from S_2, denoted by $S_1 \to S_2$. Then, S_1 copies a subset of recent updates by S_2 and can also update independently. Let s be the *selectivity* of a copier (*i.e.*, probability of copying an update). If we denote by $P_c(U)$ the probability that a copier independently performs an update U, for S_2's recent updates, we have

$$P(U \in \bar{U}_{S_1,S_2}|S_1 \to S_2) = s + (1 - s)P_c(U); \quad (10)$$

$$P(U \in \bar{U}_{\neg S_1,S_2}|S_1 \to S_2) = (1 - s)(1 - P_c(U)). \quad (11)$$

For an update not performed by S_2 recently, we have

$$P(U \in \bar{U}_{S_1,\neg S_2}|S_1 \to S_2) = P_c(U). \quad (12)$$

We compute $P_c(U)$ in the same way as $P(U)$; however, we should use different CEF-measure for S_1 here: that of updates by S_1 but not previously by S_2. Computing such measure introduces a big overhead, as we need to compute for every pair of sources. We can approximate by assuming S_1 and S_2 have the same number of capturable and mis-capturable slices and so computing by

$$C(S_1|S_2) = C(S_1) - sC(S_2); \quad (13)$$

$$E(S_1|S_2) = E(S_1) + s(1 - E(S_2)); \quad (14)$$

$$F(S_1, \Delta|S_2) = F(S_1, \Delta). \quad (15)$$

Our experiments show that such approximation significantly reduces execution time without affecting the results much.

To make our computation tractable, we assume independence of updates by one source and can thus compute $P(\Omega_1|S_1 \to S_2)$ and $P(\Omega_1|S_1 \not\to S_2)$. Then, the probability of observations (Ω_1, Ω_2) for each state comes along naturally:

$$P(\Omega_1, \Omega_2|I) = P(\Omega_1, \Omega_2|C1_{\neg c}) = P(\Omega_1, \Omega_2|C2_{\neg c}) = P(\Omega_1|S_1 \not\to S_2) \cdot P(\Omega_2|S_2 \not\to S_1); \quad (16)$$

$$P(\Omega_1, \Omega_2|C1_c) = P(\Omega_1|S_1 \to S_2) \cdot P(\Omega_2|S_2 \not\to S_1); \quad (17)$$

$$P(\Omega_1, \Omega_2|C2_c) = P(\Omega_1|S_1 \not\to S_2) \cdot P(\Omega_2|S_2 \to S_1). \quad (18)$$

Note that since states $C1_{\neg c}, C2_{\neg c}$ and I are not distinguishable from the behavior of the data sources, the probabilities of Ω_1 and Ω_2 conditioned on them are the same.

We now present several features of our model that conform to the intuition presented early in our discussion.

THEOREM 3.1. *Let s be the selectivity of copying and m be the number of wrong values in the domain. The observation probability distribution has the following properties:*

1. *if $C(S_1) < s$, adding a correct update to $\bar{U}_{S_1,S_2}$ at time t increases probability of state $C1_c$ at t, and adding a correct update to $\bar{U}_{\neg S_1,S_2}$ decreases that probability;*
2. *if $m > \frac{1}{s}$, adding a wrong update to $\bar{U}_{S_1,S_2}$ at time t increases probability of state $C1_c$ at t, and adding a wrong update to $\bar{U}_{\neg S_1,S_2}$ decreases that probability;*
3. *if $E(S_1) > .5$, adding a correct update to $\bar{U}_{S_1,\neg S_2}$ at time t decreases probability of state $C1_c$ at t;*
4. *adding a wrong update to $\bar{U}_{S_1,\neg S_2}$ at time t decreases probability of state $C1_c$ at t.* □

PROOF. *Property 1:* If U is correct, since $C(S_1) < s$,

$$P(U) = C(S_1)E(S_1)F(S_1, \Delta) < s < s + (1 - s)P_c(U).$$

So $P(U \in \bar{U}_{S_1,S_2}|S_1 \not\to S_2) < P(U \in \bar{U}_{S_1,S_2}|S_1 \to S_2)$, and the probability of state $C1_c$ increases if we add U to $\bar{U}_{S_1,S_2}$.

Also, since $C(S_1) < s$,

$$C(S_1)E(S_1)F(S_1, \Delta) < s < s + (1 + s)P_c(U).$$

So

$$P(U \in \bar{U}_{\neg S_1,S_2}|S_1 \not\to S_2) = 1 - P(U) > (1 - s)(1 - P_c(U)) = P(U \in \bar{U}_{\neg S_1,S_2}|S_1 \to S_2).$$

Thus, the probability of state $C1_c$ decreases if we add U to $\bar{U}_{\neg S_1,S_2}$.

Property 2: If U is incorrect, since $m > \frac{1}{s}$, $\frac{1}{m} < s$. So

$$\frac{1 - E(S_1)}{m} < \frac{1}{m} < s < s + (1 - s)P_c(U).$$

So

$$P(U \in \bar{U}_{S_1,S_2}|S_1 \not\to S_2) < P(U \in \bar{U}_{S_1,S_2}|S_1 \to S_2),$$

and the probability of state $C1_c$ increases if we add U to $\bar{U}_{S_1,S_2}$.

According to the above equation, we also have

$$P(U \in \bar{U}_{\neg S_1,S_2}|S_1 \not\to S_2) = 1 - \frac{1 - E(S_1)}{m} > (1 - s)(1 - P_c(U)) = P(U \in \bar{U}_{\neg S_1,S_2}|S_1 \to S_2).$$

Thus, the probability of state $C1_c$ decreases if we add U to $\bar{U}_{\neg S_1,S_2}$.

Property 3: When U is correct, for the probability of state $C1_c$ to increase when we add U to $\bar{U}_{S_1,\neg S_2}$, we should have

$$C(S_1)E(S_1)F(S_1, \Delta) > C_c(S_1)E_c(S_1)F_c(S_1, \Delta) > (C(S_1) - sC(S_2))(E(S_1) + s(1 - E(S_2)))F(S_1, \Delta).$$

The above equation holds when $E(S_1) > 1 - E(S_1)$, and so when $E(S_1) > .5$.

Property 4: Because $E(S_1) \leqslant E_c(S_1)$, so

$$P(U \in \bar{U}_{S_1,\neg S_2}|S_1 \not\to S_2) = \frac{1 - E(S_1)}{m} \geqslant \frac{1 - E_c(S_1)}{m} = P(U \in \bar{U}_{S_1,\neg S_2}|S_1 \to S_2).$$

Thus, the probability of state $C1_c$ decreases if we add U to $\bar{U}_{S_1,\neg S_2}$. □

Table 3: Observation Ω for S_5 with respect to S_3 in Example 2.1. We skip the years when all sets are empty.

Year	U_{S_5,S_3}	$U_{\neg S_5,S_3}$	$U_{S_5,\neg S_3}$
2003	{(S, UCB), (B, MSR), (H,Wisc)}	$\emptyset$	{(D, UW)}
2005	$\emptyset$	$\emptyset$	{(S, MS), (D, $\perp$), (H, Google)}
2007	{(D, Wisc), (C, BEA), (H, UW)}	$\emptyset$	$\emptyset$
2009	$\emptyset$	{(S, MIT)}	$\emptyset$

Table 4: Probabilities of hidden states for S_5 vs. S_3.

State	03	04	05	06	07	08	09
Copy ($C1_c$)	1	.43	.2	.43	1	.39	.12
Idle ($C1_{\neg c}$)	0	.51	.89	.51	0	.35	.52

Recent updates: We next define what we mean by a *recent* update by a source. To avoid penalizing updates that are not copied immediately, we consider a *window* of size W. Assume S_1 makes $W+1$ consecutive copyings at time $t_{k_0}, \dots, t_{k_w}, 0 \leqslant k_0 < \dots < k_w \leqslant n$. The *recent* updates by S_2 include all of its updates at time $(t_{k_0}, t_{k_w}]$, not overwritten by S_2's later updates on the same objects, and not performed by S_1 yet. Here, we mark a time point as a possible "copying" point if updates by S_1 at that time overlap with recent updates by S_2.

EXAMPLE 3.2. *Consider sources S_3 and S_5 in the motivating example. Table 3 shows Ω for S_5 with respect to S_3. Year 2003 and 2007 are considered as copying points. As an example, S_5 has four updates in 2003: three overlap with S_3's recent updates and are in U_{S_5,S_3}, and one, (Dewitt, UW), does not overlap and is in $U_{S_5,\neg S_3}$ (S_3's update (Dewitt, UW) in 2001 is later overwritten in 2002).*

Table 5 shows the probability of states $C1_c$ and $C1_{\neg c}$ we infer for S_5 vs. S_3 (we set $t_c = t_i = .9, f = .5$). Thus, our HMM model is able to identify that S_5 is a copier of S_3, copying in the years of 2003 and 2007. □

3.2.4 Algorithm

We apply *Forward-Backward* inference to compute the probability of each state at each time point. The probability of S_1 being a copier at time t is the sum of the probability of state $C1_c$ and $C1_{\neg c}$ (similar for S_2).

We can apply *Baul-Welch* learning to learn the parameters α, s and the transition probabilities. Note that to keep the symmetry of the transition graph, we should infer t_i, t_c and t_f from the learned transition probabilities, and then use these parameters to re-compute the probabilities. As an example, $\frac{f}{1-f}$ can be computed by $\frac{a_{C1_c,C1_c}}{a_{C1_c,C1_{\neg c}}}, \frac{a_{C1_{\neg c},C1_c}}{a_{C1_{\neg c},C1_{\neg c}}}, \frac{a_{C2_c,C2_c}}{a_{C2_c,C2_{\neg c}}}, \frac{a_{C2_{\neg c},C2_c}}{a_{C2_{\neg c},C2_{\neg c}}}$, where the a's are learned transition probabilities.. We can then compute f and take the average. In particular, if we define $div(x) = \frac{x}{1+x}$, then

$$f = Avg(div(\frac{a_{C1_c,C1_c}}{a_{C1_c,C1_{\neg c}}}), div(\frac{a_{C1_{\neg c},C1_c}}{a_{C1_{\neg c},C1_{\neg c}}}), \quad (19)$$

$$div(\frac{a_{C2_c,C2_c}}{a_{C2_c,C2_{\neg c}}}), div(\frac{a_{C2_{\neg c},C2_c}}{a_{C2_{\neg c},C2_{\neg c}}})); \quad (20)$$

$$t_c = Avg(div(\frac{a_{C1_c,C1_c} + a_{C1_c,C1_{\neg c}}}{a_{C1_c,C2_c} + a_{C1_c,I}}), \quad (21)$$

$$div(\frac{a_{C2_c,C2_c} + a_{C2_c,C2_{\neg c}}}{a_{C2_c,C1_c} + a_{C2_c,I}}))); \quad (22)$$

$$t_i = Avg(a_{I,I}, div(\frac{a_{C1_c,I}}{a_{C1_c,C2_c}}), div(\frac{a_{C2_c,I}}{a_{C2_c,C1_c}})). \quad (23)$$

EXAMPLE 3.3. *Consider the motivating example. We assume knowledge of real life spans and set $t_c = t_i = .9, f = .5$. Table 5 shows the computed probabilities of certain hidden states. Our model detects transformation of copying between S_3 and S_1 and that between S_3 and S_2; in addition, our model discovers that S_5 is a copier of S_3, despite the independent updates in 2005.* □

Table 5: Probabilities of hidden states for Example 2.1.

Year	State	01	02	03	04	05	06	07	08	09
$S_3 \to S_1$	Copy	.15	.62	.35	.37	.4	1	.39	.34	0
	Idle	0	.02	.33	.37	.41	0	.35	.29	.52
$S_3 \to S_2$	Copy	1	.82	.37	.32	.28	0	.22	.2	0
	Idle	0	.1	.32	.26	.2	.37	.18	.15	.29
$S_5 \to S_3$	Copy	-	-	1	.43	.2	.43	1	.39	.12
	Idle	-	-	0	.51	.89	.51	0	.35	.52

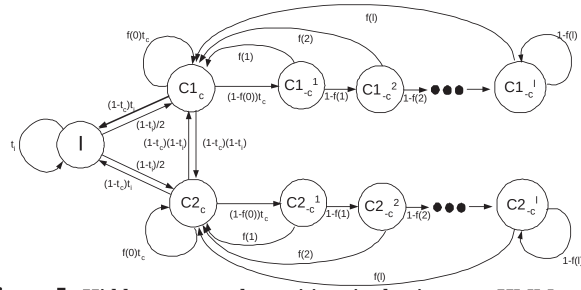

Figure 5: Hidden states and transitions in the timespan HMM model.

3.3 Considering time span

Once a source remains as a copier, it copies sooner or later. Typically, the longer it has not copied yet, the more likely that it copies next. It is also possible that a copier copies periodically: it makes independent updates for a period of time and then copies the recent updates by the original source.

To capture these intuitions, we need to reason about the time period that a copier has been independently updating; however, first-order Markov, where the probability of falling in a hidden state only relies on the state at the previous time, cannot capture this naturally. As we only care about the time period of state $C1_{\neg c}$ and $C2_{\neg c}$, we stick to first-order Markov, which is easy for learning and inference, and revise our HMM model by dividing state $C1_{\neg c}$ (similar for $C2_{\neg c}$) into a set of states $C1^1_{\neg c}, C1^2_{\neg c}, \dots, C1^q_{\neg c}$, where q is the number of observations within which a copier typically will conduct at least one copying (we discuss how to set q soon). Among $C1^1_{\neg c}, C1^2_{\neg c}, \dots, C1^q_{\neg c}$, $C1_c$ can transit only to state $C1^1_{\neg c}$; for each $i \in [1, q)$, $C1^i_{\neg c}$ can transit either to $C1^{i+1}_{\neg c}$ or to $C1_c$; finally, $C1^q_{\neg c}$ can transit to itself or to $C1_c$. Essentially, for a state $C1^i_{\neg c}$, i acts as a timer to count for how long S_1 has not copied yet. Note that this model is more meaningful when the lengths of time between consecutive observations are similar.

Figure 5 shows the revised HMM model, where $f(i), i \in [1, q]$, is the transition probability from $C1^i_{\neg c}$ to $C1_c$ and $f(0)$ is this probability when S_1 is in state $C1_c$ and remains as a copier (with probability t_c). There are various ways to define $f(i)$ and we here give a few examples.

- If we set them all as the same, the model is reduced to the basic HMM model (Figure 3).
- According to the intuition that the longer a copier has not copied, the more likely that it will copy next, we can use an attenuation function such as
$$f(i) = \frac{i+f}{i+1}, f \in (0, 1). \quad (24)$$
So $f(0) = f$ and $f(q) \to 1$. We can set q as the minimum i such that $f(i) > 1 - \theta$, where θ is a number close to 0.
- To model periodical copying where the copier copies once every k observations, we can set $q = k$, $f(i) = \theta$ for every $i \in [0, k)$ and $f(i) = 1 - \theta$ for $i = k$.

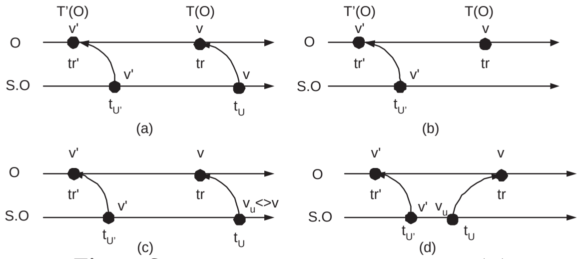

Figure 6: **Possibilities of the next update,** $U(S)$**.**

Finally, we can learn $f(i)$ by *Baul-Welch* learning. As different sources can have different copying patterns, this learning is *local* and is performed for *each* pair of sources, different from the *global* learning of other parameters.

4. DISCOVERING LIFE SPAN OF OBJECTS

We now present a Bayesian model that decides life span of an object. We start by considering a set of independent sources (Section 4.1), then extend our model by considering copiers (Section 4.2), and finally present the complete algorithm (Section 4.3).

4.1 Deciding life span

Consider an object $O \in \mathcal{O}$. To discover its life span, we need to decide both time and value of each transition. We proceed iteratively: we first decide the value of O at time t_0, then find the most likely time point and value for O's next transition, and repeat this process until we decide there is no more transition. Note that the transition points we decide have to be some observation points; in presence of precise time stamp of updates, we can extend our algorithm for more fine-grained results.

Deciding the initial value We denote by Ψ our observation of which value each source $S \in \mathcal{S}$ initially provides for O. Let $\mathcal{V}(O)$ be the domain of O. Then, our goal is to find $v \in \mathcal{V}(O)$ that maximizes $P(v|\Psi)$. According to the Bayes rule, we just need to find the v that maximizes $P(\Psi|v)$.

First, suppose $v \neq \perp$. Consider a source $S \in \mathcal{S}$. There are three cases for the initial value it provides for O:

- S provides the correct value, with probability $E(S)C(S)$ (we ignore freshness as the first observation contains updates accumulated over time);
- S does not provide a value for O, with probability $E(S)(1 - C(S))$;
- S provides a wrong value, with probability $\frac{1-E(S)}{m}$.

We denote by $\bar{S}(v)$ the set of sources providing v on O initially and by $\bar{S}(\emptyset)$ the set of sources not providing any value on O initially. Assuming independence of sources, we have

$$P(\Psi|v) = \Pi_{S \in \bar{S}(v)} E(S)C(S) \cdot \Pi_{S \in \bar{S}(\emptyset)} E(S)(1 - C(S)) \cdot \Pi_{S \in \mathcal{S} - \bar{S}(v) - \bar{S}(\emptyset)} \frac{1 - E(S)}{m}. \quad (25)$$

With similar analysis, when $v = \perp$, we have

$$P(\Psi|\perp) = \Pi_{S \in \bar{S}(\emptyset)} E(S) \cdot \Pi_{S \in \mathcal{S} - \bar{S}(\emptyset)} \frac{1 - E(S)}{m}. \quad (26)$$

We can thus decide the initial value of O accordingly.

Deciding the next transition Deciding the next transition is harder than deciding the initial value, as we need to consider an additional dimension–the time. Essentially, we solve the following problem. Given the last transition $T'(O) = (tr', v')$ we have discovered on O, decide the next transition $T(O) = (tr, v), v \in \mathcal{V}(O), v \neq v', tr \geqslant tr'$ ($T'(O)$ and $T(O)$ can happen within the same observation period). We start from a simple case, where for each source $S \in \mathcal{S}$, there is an update at time $t_{u'} \in [tr', tr)$ corresponding to $T'(O)$ (we describe other possibilities shortly). We denote S's next update by $U(S)$ and the observation of $U(S)$ for each $S \in \mathcal{S}$ by Φ. According to the Bayes rule, we need to find v and tr that maximizes $P(\Phi|T(O) = (tr, v))$.

Suppose $T(O) = (tr, v)$ exists. There are then three possibilities for $U(S)$.

Case I. S captures the transition by updating the value of O to v_c at time $[tr, t_n]$ (recall that t_n is the last observation point in $\mathbf{T}$) (Figure 6(a)). The probability of not making an error and capturing the transition is $E(S)C(S)$, and the probability of capturing it at a particular point is decided by the freshness function. Thus, (we assume $F(S, \Delta) = 0$ when $\Delta < 0$)

$$P\Big(U(S) = (t_u, v), t_u \in [tr, t_n] | T(O) = (tr, v)\Big) = E(S)C(S) \int_{t_{u-1}-tr}^{t_u-tr} F(S, t)dt. \quad (27)$$

Case II. S misses the transition by not updating at $[tr, t_n]$ (Figure 6(b)). The probability of not making an error but not capturing the transition either is then

$$P\left(\overline{U(S)} | T(O) = (tr, v)\right) = E(S)\left(1 - C(S)F(S, t_n - tr)\right). \quad (28)$$

Case III. S makes an error either by updating to a different value (Figure 6(c)). or by updating before tr (Figure 6(d)). The probability of making an error is $1 - E(S)$. The error can be made at any time between $(t_{u'}, t_n]$, so the probability of observing an error at each point t_u is $\frac{t_u - t_{u-1}}{t_n - t_{u'}}$. Among all values in $\mathcal{V}(O)$, the probability of providing a particular wrong value is approximately $\frac{1}{m}$. So we have

$$P\Big(U(S) = (t_u, v_u), t_u \in (t_{u'}, tr) \vee v_u \neq v | T(O) = (tr, v)\Big) = \frac{(1 - E(S))(t_u - t_{u-1})}{m(t_n - t_{u'})}. \quad (29)$$

As we assume independence of sources, we can compute $P(\Phi|T(O) = (tr, v))$ as

$$P\Big(\Phi | T(O) = (tr, v)\Big) = \Pi_{S \in \mathcal{S}} P(U(S) | T(O) = (tr, v)). \quad (30)$$

We apply similar analysis in case there does not exist any more transition after T', denoted by $\overline{T(O)}$. We have

$$P(\overline{U(S)} | \overline{T(O)}) = E(S); \quad (31)$$

$$P\Big(U(S) = (t_u, v_u), t_u > t_{u'} | \overline{T(O)}\Big) = \frac{(1 - E(S))(t_u - t_{u-1})}{m(t_n - t_{u'})} \quad (32)$$

We can thus choose the pair of tr and v with the maximum value of $P(\Phi|T(O) = (tr, v))$ as the next transition, or terminates when $P(\Phi|\overline{T(O)})$ has the maximum value.

This Bayesian model has the following properties, conforming to the intuition that the more sources update the value of O to v in a close time frame, the more likely that the transition involves value v and happens before their updates.

PROPOSITION 4.1. *Let S be a source and $T(O)$ be a transition. Among $F(S, 0)$ and $F(S, t_i) - F(S, t_{i-1}), i \in [1, n]$, let $F_{max}(S)$ be the maximum one and $F_{min}(S)$ be the minimum one. Consider Ψ_1, Ψ_2, Ψ_3, which differ only in that $U(S)$ conforms to $T(O)$, does not exist, or does not conform to $T(O)$, respectively. Then,*

- *If $C(S) > \frac{1}{F(S,0)+F_{max}(S)}$, $P(T(O)|\Psi_1) > P(T(O)|\Psi_2)$;*
- *If $E(S) > \frac{1}{1+mC(S)F_{min}(S)}$, $P(T(O)|\Psi_1) > P(T(O)|\Psi_3)$;*

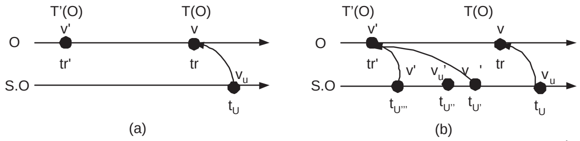

Figure 7: Possibilities of update(s) for the previous transition $T'(O)$.

- *If $E(S) > \frac{1}{1+m(1-C(S))}$, $P(T(O)|\Psi_2) > P(T(O)|\Psi_3)$.* □

PROOF. This can easily verified by comparing Equations (27-29). □

Finally, note that $U(S)$ is defined as the update after the update corresponding to $T'(O)$, denoted by $U'(S)$. Intuitively, $U'(S)$ should be an update that changes the value of O to v' in $[tr', tr)$. However, there can be three cases: (1) there is one and only one such update (Figure 6), so we consider it as $U'(S)$; (2) there is no such update (Figure 7(a)), so we consider $U'(S)$ not existing, $U(S)$ as the first update after tr', and $t_u = tr'$; (3) there are multiple such updates (Figure 7(b)). The third case is especially tricky because some of the updates may fix previous errors (so correspond to $T'(O)$) and some may respond to later transitions. We conduct a simple voting on the value of each moment since tr', and find the first point when the value of O changes from v' to another value. Accordingly, we consider updates to v' before this point as corresponding to $T'(O)$ and choose the last one as $U'(S)$.

4.2 Considering copiers

We next consider the case where some sources can copy from others. We first need to decide the probability that an update U is independent. At each observation point t, for each pair of sources S_1 and S_2, we can compute the probability that S_1 is actively copying from S_2, denoted by $P(S_1 \to S_2, t)$. We consider S_1 being a copier of S_2 at t if the probability of state I at t is less than .5 and the probability of S_1 being a copier ($P(C1_c) + P(C1_{\neg c})$) is larger than that for S_2. Given an update U at time t, we consider the probability of U being copied from S_2, denoted by $P(S_1 \xrightarrow{U} S_2)$, as $P(S_1 \to S_2, t)$ if (1) S_1 is a copier of S_2 at t, (2) S_2 provides the same value for O at t, and (3) S_2 makes that update no later than U, and as 0 otherwise. We assume copying between sources are independent; thus, the probability of S_1 independently making an update U is

$$P(U \text{ indep}) = \Pi_{S_2 \in \mathcal{S}, S_2 \neq S_1}(1 - P(S_1 \xrightarrow{U} S_2)). \quad (33)$$

When we compute $P(\Phi|T(O))$, we revise Equation (30):

$$P(\Phi|T(O)) = \Pi_{S \in \mathcal{S}} P(U(S)|T(O))^{P(U(S) \text{ indep})}. \quad (34)$$

Finally, we note that our copying discovery techniques cannot avoid a transitive inference of loop copying among more than two sources, and we can end up marking the same updates at the same time by sources in the loop all as copied (with a probability). However, such coincidence is typically rare and can be ignored.

4.3 Putting them all together

Finally, we consider how we decide life span of each object given updates by each source. In this process, we should take into consideration copying between sources and CEF-measure of sources. However, discovering source copying requires knowledge of life span of each object and CEF-measure of each source, and computing CEF-measure requires knowledge of life spans of objects.

Our algorithm thus proceeds in an iterative fashion. In each round, we first compute CEF-measure of each source (depending only on life spans), then compute probability of copying between sources, and finally (re)decide the life span of each object, until the discovered life spans do not change. Note that in the first round we do not know the life span yet; we initialize the CEF-measure of each source to the same default value except setting the coverage as the percentage of objects being covered by the source; and we assume each update has .5 probability to be true when applying the HMM model for copying detection.

```
LIFESPAN (S, O)
while life span changes && no oscillation of life span do
  for each S ∈ S do
    Compute CEF-measure of S; endfor
  HMMDEPEN(S, O); //Decide copying between sources
  for each update U of a source S ∈ S do
    Compute P(U indep); endfor
  for each O ∈ O do
    Decide life span of O; endfor
endwhile
```

Figure 8: Algorithm LIFESPAN: decide copying between data sources and life span of objects.

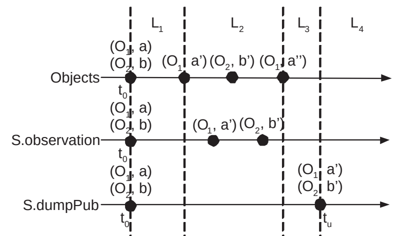

Figure 9: Different types of delayed publishing.

Figure 8 shows the complete algorithm. Convergence of the algorithm remains an open problem, but in our experiments we observe that when the total number of transitions is far more than the number of sources, the algorithm converges quickly. Time complexity of each round is shown as follows.

PROPOSITION 4.2. *Let m be the number of values in the domain of an object, and u be the total number of updates by sources in $\mathcal{S}$. We denote by $|\bar{X}|$ the size of set $\bar{X}$. Then, complexity of one round in Algorithm* LIFESPAN *is $O(|\mathcal{O}| \cdot |\mathcal{S}|^2 \cdot |\mathbf{T}| + m \cdot |\mathcal{O}| \cdot |\mathcal{S}| \cdot |\mathbf{T}|^2 + u \cdot |\mathcal{S}|)$.* □

PROOF. Computing CEF-measure of each source takes time $O(|\mathcal{O}| \cdot |\mathbf{T}|)$, and in total takes time $O(|\mathcal{O}| \cdot |\mathcal{S}| \cdot |\mathbf{T}|)$.

Deciding dependence for each pair of sources takes time $O(|\mathcal{O}| \cdot |\mathbf{T}|)$, and in total takes time $O(|\mathcal{O}| \cdot |\mathcal{S}|^2 \cdot |\mathbf{T}|)$.

Deciding independence probability for each update takes time $O(|\mathcal{S}|)$, and in total takes time $O(u \cdot |\mathcal{S}|)$.

Finally, deciding lifespan of each object takes time $O(m \cdot |\mathcal{S}| \cdot |\mathbf{T}|^2)$, and in total takes time $O(m \cdot |\mathcal{O}| \cdot |\mathcal{S}| \cdot |\mathbf{T}|^2)$.

We thus have the complexity bound. □

EXAMPLE 4.3. *Consider Example 2.1. Table 6 shows the life span discovered for* Halevy. *The algorithm converges in four rounds. The CEF of S_1 converges at a high coverage and exactness, whereas that of S_2 converges at a low coverage and exactness.* □

5. CONSIDERING DELAYED PUBLISHING

Previous sections assume instant publishing. In this section we consider *delayed publishing*, where a source can publish an update later than the change is observed (from the real world or from other sources). Delayed publishing can happen when a weekly newspaper publishes news collected through the whole week, when a web portal publishes data crawled in a period of time in a batch mode, when we observe only a subset of sources each time, and so on. We assume *dump publishing*, where a source publishes all data it has collected since the last publishing. This is common in practice and our techniques can be easily extended to the case where a source

Table 6: **Discovered life span for *Halevy* and computed CEF-measure for S_1 and S_2 in Example 2.1.**

Round	Life span for *Halevy*	S_1.C	S_1.E	S_1.F(0)	S_1.F(1)	S_2.C	S_2.E	S_2.F(0)	S_2.F(1)
0		.99	.95	.1	.2	.99	.95	.1	.2
1	(2000, Wisc), (2002, UW), (2003, Google)	.97	.94	.27	.4	.57	.83	.17	.3
2	(2000, UW), (2002, Google)	.92	.99	.27	.4	.64	.8	.18	.27
3	(2000, UW), (2005, Google)	.92	.99	.27	.4	.64	.8	.25	.42

publishes only a subset of collected data each time, but do so in an ordered manner.

In dump publishing, an update decision can be made at any point between two consecutive publishings; we revise our models accordingly to consider all possibilities.

CEF-measure For instant publishing, we decide if a slice is captured or mis-captured by comparing the update at the end of the slice (if any) with the real value of that slice. In case of delayed publishing, if a slice L ends with an update U, but the previous publishing happens before the beginning of L, then U may actually tend to capture the value of L's previous slices. As an example, in Figure 9, the update (O_1, a') at t_u is to capture the value of slice L_2, not L_3.

Formally, consider source S and object O. Let $L_1, \ldots, L_k$, $k \geqslant 1$, be a set of consecutive slices for S and O, such that L_k ends with an update U on O, and the previous publishing Pub happens in slice L_1. Let l_1 be the difference between the time of publishing Pub and the end of slice L_1, and let $l_i, i \in [2, k]$, be the length of L_i. Then, with probability $p_i = \frac{l_i}{\sum_{j=1}^{k} l_j}, i \in [1, k]$, U is collected (though not published) in slice L_i. Accordingly, L_i is captured by U with probability p_i if U provides the real value of L_i, and mis-captured with probability p_i otherwise. We then compute $c(S, O)$ and $m(S, O)$ by summing up the corresponding probabilities. Similarly, when we compute freshness, if a slice $L_i, i \in [1, k]$, is captured, we consider equal likelihood of the update being captured at any time in L_i.

Life span Deciding the next transition $T(O)$ relies on computing the conditional probability of $U(S) = (t_u, v_u)$. Assume S previously publishes at time $t_{pre}, pre \in [0, u)$. Then the update is published at t_u once the source decides to make the update at some time in $(t_{pre}, t_u]$, so we should take the sum of the probabilities:

$$P\Big(U(S,T) = (t_u, v_u)|T(O)\Big) = \sum_{i=pre+1}^{u} P\Big(U(S) = (t_i, v_u)|T(O)\Big). \tag{35}$$

Copying When deciding source copying, it is critical to compute the probability of a source independently making an update at a particular time, $P(U, S_1, t_u|S_1 \not\rightarrow S_2)$. In case of delayed publishing, an update decision actually can be made at any time after the previous publishing t_{pre}; the probability of the decision at a particular observation point is proportional to the length of that observation period. We thus take the weighted average of the probabilities (note the difference from computing $P\big(U(S) = (t_u, v_u)|T(O)\big)$):

$$P(U, S_1, t_u|S_1 \not\rightarrow S_2) = \sum_{i=pre+1}^{u} P(U, S_1, t_i|S_1 \not\rightarrow S_2) \cdot \frac{t_i - t_{i-1}}{t_u - t_{pre}}. \tag{36}$$

In addition, when we decide the independence probability of an update U by S_1, we consider that U is possibly copied from S_2 if the updated value is the same as a value once provided by S_2 since the last publishing of S_1.

6. EXPERIMENTAL RESULTS

This section presents experimental results on life-span discovery and copying detection. We first present results on a real-world data set (Sec. 6.2), showing that the problems we address in this paper are real issues in the world, and our methods can improve quality of the integrated data. Since in most cases we have no means to check the actual copying relationship and the precise life span of objects in the real world, we also experimented on synthetic data. We first present results on a data set that mimic complexity in the real world, examining contribution of different components to our algorithms (Sec. 6.3.2). We then consider a harder case, where we care about only existence of sources so variety of update traces by different sources significantly reduces. We examine performance and robustness of our models on life-span discovery (Sec. 6.3.3) and copying detection (Sec. 6.3.4).

6.1 Experiment setup

We consider a set of data sources and objects as described in Sec. 2 and refer to them as a *universe*. We refer to the special case where each object has only two possible values, existing and non-existing ($\perp$), as *binary universe*. Our goals are to decide life span of objects and copying between sources in a given universe.

For life-span discovery, our algorithm has three main components: *copy*–considering copying between sources, *CEF*–considering CEF-measure of sources, and *delay*–considering publish delay. We implemented several variants by combining different components.

- NAIVE: For each object, vote for its value at each observation point.
- SIMPLE: First apply NAIVE and then decide transition points iteratively: for each point t where the voted value changes to a new value v, find the earliest point since the last transition when a source provides v and does not update until point t.
- COPY: The same as SIMPLE except considering copying in voting (Eq. (33)).
- CEF: Consider CEF-measure (Eq. (25-32)).
- CEFDELAY: *CEF+delay* (Eq. (35)).
- COPYCEF: *copy+CEF*. (Algorithm LIFESPAN).
- COPYCEFDELAY: *copy+CEF+delay* (Eq. (35-36)).

For copying discovery, we compared static models (INIT, CURR, TRACE), various HMM models (HMM5, HMMN, HMM3), and HMM with consideration of publish delay:

- INIT/CURR: Consider only initial or latest values.
- TRACE: Compute Ω_1 and Ω_2 for each observation point and reason over the accumulated results (Eq. (16-18)).
- HMM3: An HMM model with three states $I, C1_c$, and $C2_c$, the same transition probabilities as Figure 3 except that $a_{I,I} = t_i - f, a_{I,C1_c} = a_{I,C2_c} = \frac{1-t_i+f}{2}, a_{C1_c,I} = a_{C2_c,I} = (1-t_c)t_i + t_c(1-f)$.
- HMM5: The basic HMM model with 5 states (Fig. 3).
- HMMN: The timespan HMM model (Fig. 5) with $f(i) = \frac{i+f}{i+1}, q = 5$.
- HMMDELAY: HMM5 with publish delay (Eq. (36)).

By default, we computed weighted CEF-measure and applied HMM5 in life-span discovery. Initially we set $\alpha = f = .5, t_i = t_c = 0.99, s = .8, m = 100$ when apply, but may apply *Baul-Welch* learning to learn the parameters in some experiments.

Table 7: Discovered ever-existing restaurants (#Rest) and closing restaurants in Manhattan.

Method	#Rest	Prec	Rec	F-msr	#Rnds	Time(s)
ALL	-	.54	1	.7	-	-
ALL2	-	.82	.31	.45	-	-
NAIVE	1100	.62	.92	.74	1	151
CEF	5056	.73	.85	.78	5	537
COPYCEF	5080	.76	.85	.8	5	1118
GOOGLE	-	.77	.15	.24	-	-

We measure lifespan-discovery results by *edit distance*, defined as the Levenshtein distance between decided life-span periods and real periods, where insertion or deletion of a period is penalized by the length of the period, and substitution of a period with the same value is penalized by the difference of the beginning points. Ideally, the edit distance should be 0. We describe how we measure copying-detection results in Sec. 6.3.4.

We implemented our models in Java and conducted experiments on a WindowsXP machine with AMD Athlon(tm) 64 2GHz CPU and 960MB memory.

6.2 Experiments on real-world data

We randomly selected 12 web sources (listed in Table 1 at the beginning of this paper; $F(0)$ is shown as freshness for each source) that provide information on restaurants in Manhattan. We crawled their data from 1/22/2009 to 3/12/2009, once every week, so 8 times in total. For each restaurant listing, we collected name, phone number, address, direction, neighborhood, and price range whenever possible. We identified restaurants by their names and considered only restaurants that are mentioned by at least two data sources. In total there are 5149 such restaurants; among them, 5113 appeared in our first crawling, and 5131 appeared in our last crawling. In these two months, we did not notice many changes on attribute values such as phone and address, thus focused on existence of restaurants (so a binary universe).

We considered two cases as deletion of a restaurant from a source: the source explicitly marks the restaurant as "(CLOSED)", or the source implicitly removes the restaurant from its list. We considered the set of restaurants that a source provided once but deleted later. There are 463 such restaurants. For each of them, we called its phone number to verify if it is still open and used it as the golden standard; we found that 248 of them are indeed closed[4]. We ran various algorithms to decide life span (existence periods) of each restaurants, and reported *precision*, *recall*, and *F-measure*, denoted respectively as $P, R, F1$, of our results. Formally, among the 463 restaurants, we define $\bar{G}$ as the set of restaurants that are *closed* in the golden standard and $\bar{R}$ as the set of restaurants that our algorithm decided as closed. Then, $P = \frac{|\bar{G}\cap\bar{R}|}{|\bar{R}|}, R = \frac{|\bar{G}\cap\bar{R}|}{|\bar{G}|}$, and $F1 = \frac{2PR}{P+R}$. In addition, we searched each of the restaurants on Google Maps and reported the three measures as well.

Table 7 shows results of various methods[5]. We observed as follows. (1) COPYCEF and CEF obtain high precision and recall. Between them, COPYCEF obtains higher precision and discovers more restaurants that have ever existed by considering copying. (2) NAIVE seems to have a high F-measure; however, as most restaurants are often mentioned by a few sources, it concludes that only 1100 out of 5149 restaurants have ever existed. (3) Considering all of the 463 restaurants as closed (referred to as ALL) has low precision (.54), while considering only restaurants that are removed by at least two sources as closed (referred to as ALL2) has low recall (.31). Finally, we observed that Google Maps lists many out-of-business restaurants, reflecting staleness of data on the web.

We observed that COPYCEF and CEF both converged at the 5th round and took 18.6 and 8.95 minutes respectively. Since life-span discovery is a one-time process, the execution time is acceptable.

Among the 66 pairs of sources, we detected copying relationship between 14 pairs (Figure 1). Among the sources, it is more likely that *FoodBuzz* and *VillageVoice* are copiers and *MenuPages* and *TimeOut* are being copied. Although we do not know the real copying relationship between sources, we have the following evidence to support some of our results. First, *FoodBuzz* and *VillageVoice* each formats addresses of different restaurants in very different ways, so may copy them from various sources; in addition, *FoodBuzz* inserted restaurants even after the restaurants were closed, so possibly copied the data. Second, *MenuPages* has been on the web for the longest time among the 12 sources and has the highest coverage (.66), so is possible to be copied by other sources.

[4]Some of the 248 restaurants were closed before 1/22/2009 and were already marked "(CLOSED)" by some sources on 1/22/2009.

[5]We skip results of COPYCEFDELAY as almost all sources update every week, and skip results of SIMPLE and COPY, which are similar to NAIVE.

Table 8: Source-generation parameters and their settings.

Parameter	p_t	p_f	f_0	f_u	ns_u	os_u	a_u
Default	.75	.99	.1	.5	.8	.2	.075
Range	0-1	.95-1	.1-1	.1-1	.1-1	.2	0-.15

6.3 Experiments on synthetic data

We next describe how we generated the synthetic data and reported experimental results.

6.3.1 Synthetic data

Objects: A universe contains 100 objects. In a multi-valued universe, the domain for each object contains 102 values (including $\perp$). We have 20 periodical observations at $t_0, \ldots, t_{19}$. A universe can be either *single-period* or *multi-period*: in the former an object exists at t_0 with probability $p_i = .5$ and at t_{19} with probability $p_e = .3$, but does not change value during existence; in the latter an object exists at t_0 with probability $p_i = .5$, transits at each observation point with probability $p_c = .1$, and once transits, disappears with probability $p_e = .1$ or changes to another random value otherwise. We note that results with more objects, more observations, or larger domains are similar.

Sources: According to different types of data sources, we classify a universe into three categories.

- *Independence universe* contains 10 independent sources.
- *Copier universe* contains 10 independent sources and 9 copiers, all copying from the same independent source.
- *Random universe* contains 10 independent sources and a number of copiers. Each copier copies from a randomly selected source, either independent or being a copier as well, but there is no loop copying.

For each independent source S and object O, at *each* observation point S updates the value of O to a random false value with probability p_f, and updates to the true value with probability $p_t \cdot f(\Delta)$, where Δ is the difference between the observation and the transition of that true value. We define $f(\Delta) = f_0 \cdot 2^{\Delta}$ when $0 \leqslant \Delta \leqslant -\log f_0$, and $f(\Delta) = 1$ when $\Delta > -\log f_0, 0 < f_0 \leqslant 1$. Note that although p_t, p_f and f are related to the CEF-measure, the definitions are not exactly the same and we chose to do so to test robustness of our model. Table 8 shows how we set the parameters. For the *random universe*, we randomize p_t, p_f, f_0 by Gaussian distribution with mean $.75, .95, .1$ respectively.

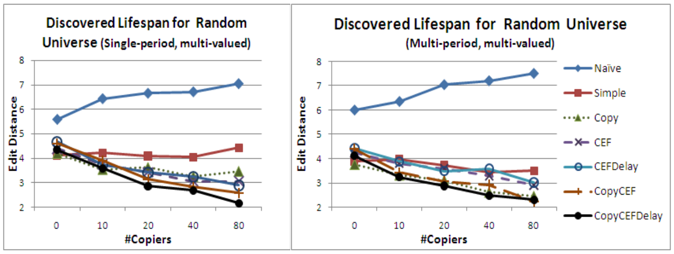

Figure 10: Life-span discovery for the random universe. COPYCEFDELAY **always obtains the best results.**

For each copier C and its original source S, at each observation point, C copies from S with probability f_u. For the *copier universe*, when C copies, with probability ns_u it copies a value provided by S since last copying and with probability os_u it copies a value provided by S earlier. For each object on which it does not copy, with probability a_u it independently provides the true value; Table 8 shows setting of these parameters. For the *random universe*, we randomize f_u, ns_u, a_u by Gaussian distribution with mean $.5, .8, .1$ respectively; when a copier does not copy on a particular object, it examines the object with probability a_u and provides a value according to p_t, p_f and f.

By default we consider instant publishing and no transformation between being a copier and being independent; though, we consider different settings in some experiments. In the *random universe*, half initial independent sources and half initial copiers can transform between being independent and being a copier, and the probability of transformation at each observation point is .1. Once a source transforms to a copier, it can chooses a source it has not copied from before. Among each kind of sources, half sources publish instantly and half can delay publishing.

For each setting, we ran the experiments 10 times and reported the average measure.

6.3.2 *Results for multi-valued universe*

Fig. 10 shows results of various methods for the *random universe*, which tries to mimic the complexity of the real world. We considered both single-period life span and multi-period life span and have several observations. (1) COPYCEFDELAY obtains the best results in most cases: when there are 80 copiers, it reduces the edit distance by 51% compared with SIMPLE and by 69% compared with NAIVE in the multi-period case. (2) COPYCEF obtains slightly worse results, but performs better than other methods. (3) CEF and COPY have similar results and improve over SIMPLE. Note that in case of multi-period life spans, the improvement by CEF is slight, as the length of each period tends to be short and the CEF-measure may not be computed accurately.

We next test robustness of our model on various settings of parameters for universe generation. We describe our results on the binary universe, which forms a harder case, but observed similar trend on multi-valued universe as well.

6.3.3 *Life-span discovery for binary universe*

Quality of sources: First, we compared various methods on the *copier universe* when we varied quality of the independent sources (Fig. 11[6]). We make the following observations. (1) CEF obtains better results than NAIVE and SIMPLE in all different settings. (2) COPY obtains similar, but sometimes even worse results than SIMPLE, showing that considering copying in itself is not enough in binary universe. Once we consider both copying and CEF-measure, the results are significantly improved: when $p_f = .01, f_0 = .1$, on

[6]We observed similar trends when p_f is higher, but higher edit distance for each method.

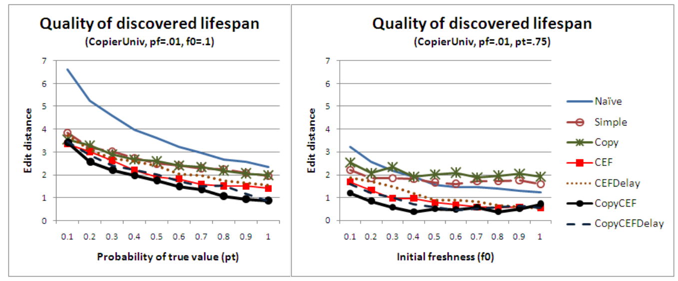

Figure 11: Life-span discovery for copier universe with instant publishing. COPYCEF **always obtains the best results while** COPYCEFDELAY **obtains slightly worse results.**

average COPYCEF reduces the edit distance by 27.8% over SIMPLE and by 18.3% over CEF. (3) CEFDELAY and COPYCEFDELAY obtain slightly worse results than CEF and COPYCEF respectively, showing that considering publish delay in case of instant publishing does lose information, but only slightly. (4) Typically, COPYCEF obtains better results with higher CEF-measure (lower p_f, higher p_t or f_0). The only exception is when the CEF-measures are all very high, so all sources are highly similar and COPYCEF can wrongly identify an independent source as a copier.

We observed similar trends on the *independence universe*, except that COPYCEF and CEF obtain similar results, showing no negative effect in considering copying when there is no copiers.

Quality of original source: We varied quality of the source that is copied while applied the default parameters for other independent sources. Fig. 12 shows difference of the edit distance when all independent sources have the same quality and that when the source being copied has different quality. We observe that the difference for COPYCEF is very low, and much lower than that of CEF; for example, when p_f increases from .01 to .05 and p_t decreases from .75 to .1, the edit distance of the results increases only by .28, while the difference for CEF is 1.36. Thus, COPYCEF is insensitive to quality of the sources that are copied.

Multi-period life spans: We constructed multi-period life spans in two ways: (1) randomly generate multiple periods observing $p_i = .5, p_c = .3$; (2) randomly choose the first existence point in $[0, l-1]$, and generate life span with periods of length l. Fig. 13 shows the results of COPY, CEF and COPYCEF. We observe that COPYCEF obtains the best results in most cases. For all models, the edit distance significantly decreases when the average length of the periods increases; the edit distance remains stable once the length reaches 10, as beyond this point the average length of existing and non-existing periods remains as 10. Thus, COPYCEF performs better when life-span periods are longer.

Publish delay: We constructed sources with delayed publishing in two ways: (1) each source publishes at the beginning and then publishes at a particular observation point with probability .5, (2) each source publishes at the beginning, randomly chooses the second publish point in $[1, d]$, and then publishes after every d observations. Fig. 14 shows the results. COPYCEFDELAY obtains the best results when there exists publish delay: edit distance of its results remains stable as d increases; on average it improves over COPYCEF by 26% in presence of delayed publishing; and in general, the higher the publish delay, the larger improvement. Thus, COPYCEFDELAY handles publish delay well.

6.3.4 *Copying detection for binary universe*

We next compare different models for copying detection.

Overtime copier We start with the case when a copier does not transform to be independent. We measure discovered copying by

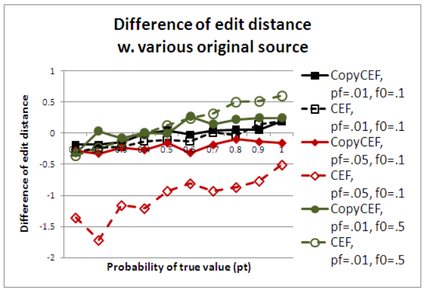

Figure 12: COPYCEF **is robust to quality of the source being copied.**

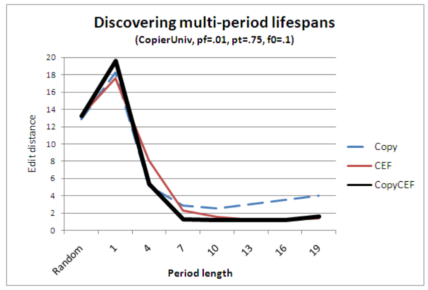

Figure 13: COPYCEF **obtains better results with longer-period life spans.**

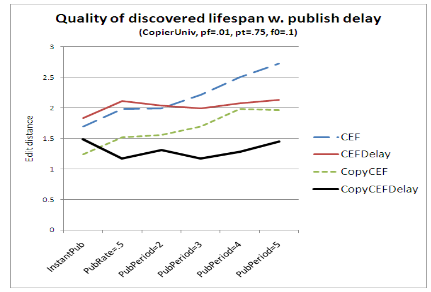

Figure 14: COPYCEFDELAY **is robust to publish delay.**

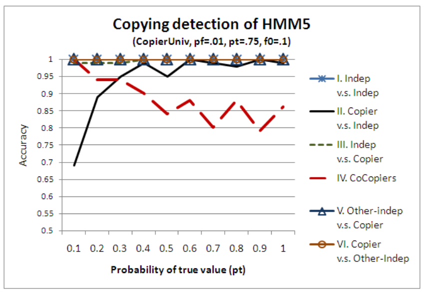

Figure 15: Copying detection of HMM5.

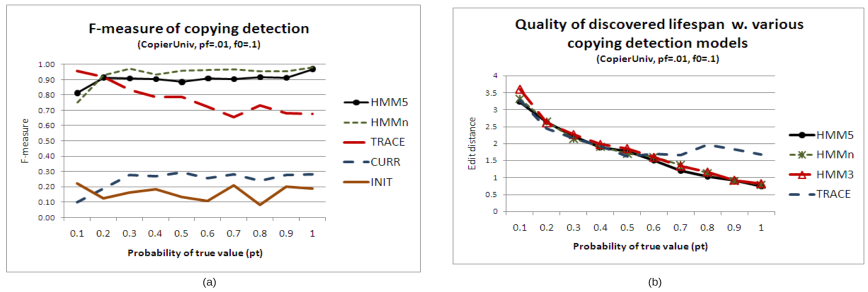

Figure 16: HMMN **obtains the highest F-measure among various copying detection models.**

precision, recall and F-measure. Let $\bar{G}$ be the set of ordered pairs (S_1, S_2) where S_2 is a copier of S_1, and $\bar{R}$ be such pairs returned by our model. We compute the three measures as we described earlier. Note that in the *copier universe*, $\bar{G}$ contains only 10 pairs but $\bar{R}$ can potentially be much larger. To balance $\bar{G}$ and $\bar{R}$, we divide the universe into sub-universes, each containing two copiers, their original source, and another independent source, and take the average over sub-universes.

We further examine accuracy of copying decision for the following categories: I. two independent sources, II. a copier vs. its original source, III. a source vs. its copier, IV. two co-copiers, V. an independent source vs. a copier of another source, and VI. a copier of a source vs. another independent source. Note that accuracy of Category II is the recall.

We start with analysis of HMM5. Fig. 15 shows its accuracy on various categories of source pairs. We observe that (1) HMM5 is good at identifying copiers: in Category II, when the coverage is above .2, we obtain an accuracy of above 95%; we can miss copiers when the coverage is low because the copiers actually conduct more independent updates than copied updates, so are hard to be distinguished from independent sources; (2) HMM5 achieves an accuracy of nearly 1 for identifying sources that are independent of each other (I, V, VI); (3) The only category for which we are not doing very well is between co-copiers (IV), as they share similar update patterns; however, the accuracy is still 88% on average.

We then compared various methods for copying detection: Fig. 16(a) shows their F-measure and Fig. 16(b) shows their effect on results of COPYCEF. We have several observations. First, considering only a snapshot of data obtains very low precision and recall (on average F1=.25 for CURR and F1=.16 for INIT). Second, TRACE obtains a low precision and significantly worsen results of COPYCEF, especially when f_t is high; this is because it accumulates a lot of overlapped updates over time and so is likely to conclude copying. Third, although we cannot compare HMM3 and HMM5 directly on F-measure of copying detection, we observe that the edit distance of discovered life span using HMM3 is 6.7% larger than HMM5, as HMM3 does not distinguish an independent source and an idle copier (not copying). Finally, HMMN obtains the highest F-measure in most cases by asserting that the longer a copier has not copied, the more likely it should copy next, though, it does not improve results of COPYCEF much.

Next, we compared HMM5 and HMMDELAY when there is publish delay (Fig. 17). HMMDELAY indeed improves over HMM5 by 8.5% on recall, 4.3% on precision, and 6.5% on F-measure on average; however, we did not observe obvious difference in life-span discovery.

Finally, we examined whether our model is robust with respect to different copy patterns by varying the selectivity of copying (s_u) from .1 to 1, and changing the copy rate to 1. We observe similar precision and show only recall. Fig. 18 shows that if we use the default selectivity (.8) in our HMM model, we can obtain a low recall when s_u is low (below .5). If we learn selectivity, the recall increases from .63 to .84 on average; if we learn copy rate in addition, the recall increases further to .94 and is high for almost all different values of s_u. The results show robustness of our HMM model to different initial settings of parameters.

Copying transformation We considered transformation and generated copiers that are initially copiers or independent sources, and then transform at a particular point. We compared the transformation points our model computes with the real ones; if the copier does not transform, we consider that the transformation point is 20 (the number of observations). Fig. 19 shows the average transformation points that HMM5 computes. On average the difference between real transformation points and those computed by HMM5 is small: 2 when the copiers are initially copying, and .44 when they are initially independent.

Summary Our experimental results on synthetic data have the following implications:

1. Considering CEF-measure and copying both contribute to improving quality of life-span discovery results.
2. Considering publish delay can significantly improve the results in presence of delay, and only slightly worsen the results in absence of delay.

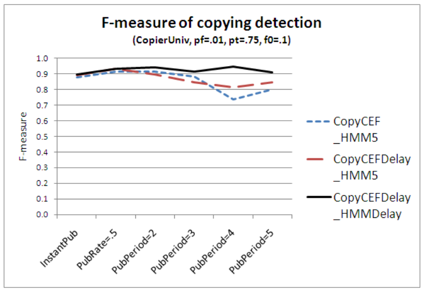

Figure 17: HMMDelay **is not sensitive to publish delay.**

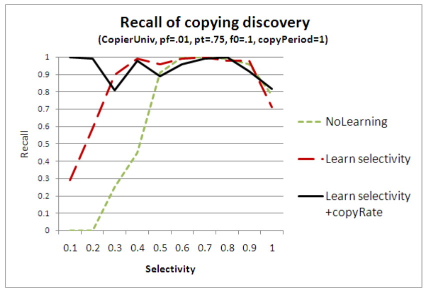

Figure 18: HMM5 **w. learning is robust to initial parameter settings.**

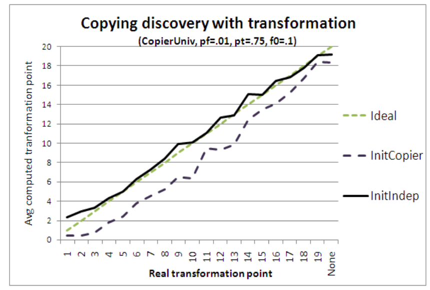

Figure 19: HMM5 **is good at detecting transformations of copiers.**

3. The basic HMM model is accurate in detecting copying; the lifespan HMM model improves the results only slightly, but with higher cost.
4. Our model is robust to different characteristics of data and initial HMM parameter settings.

7. RELATED WORK

There are three bodies of work related to our research: *truth discovery, copying detection* and *data freshness*. Recent work on truth discovery considers a snapshot of data. Bleiholder and Naumann [2] surveyed existing strategies for resolving inconsistency in structured or semi-structured data. Yin et al. [13] considers accuracy of sources in truth discovery. We consider discovering the whole life span of an object from history of source updates and we use more fine-grained source-quality measures: coverage, exactness, and freshness.

For copying detection, Berti-Equille et al. [1] recently sketched several high-level intuitions, but did not give concrete algorithms. Dong et al. [5] proposed detecting copying from a snapshot of data by examining overlapping errors between sources; such a model, however, can fall short in presence of large overlap of out-of-date data. We consider update history of sources in copying detection and decide in which period a source is a copier and at which particular moments it copies. We are not aware of any other work for copying detection on relational data. In addition, we distinguish our work from *data provenance* [3], which assumes knowledge of provenance and focuses on management of such information.

Finally, existing work on data freshness [12, 8, 4, 9, 7, 10] defines *freshness* as how stale the data in a materialized view are compared with the original sources, and emphasize update propagation. We have different focus and consider consistency of data with respect to evolution of real-world objects over time. We note that the notions of *completeness*, *consistency*, and *currency* in [7] are analog to our CEF-measure, but in different contexts.

8. CONCLUSIONS

This paper considers how we can explore update history of sources in improving quality of integrated data. We measure quality of source data by coverage, exactness, and freshness. Based on these measures, we developed an HMM model to decide whether a source is a copier of another source and at which moment it copies. Then, we developed a Bayesian model to decide life span of each object, taking into consideration CEF-measure of sources, copying between sources, and possible publish delay. Experimental results on real-world and synthetic data show high accuracy and efficiency of our models.

For future work, one direction is to apply our techniques in Web 2.0 applications to identify sources or users that are trustable. Another direction is to optimize query answering in a data integration system with knowledge of quality of sources and dependence between sources.

9. REFERENCES

[1] L. Berti-Equille, A. D. Sarma, X. Dong, A. Marian, and D. Srivastava. Sailing the information ocean with awareness of currents: Discovery and application of source dependence. In *CIDR*, 2009.

[2] J. Bleiholder and F. Naumann. Conflict handling strategies in an integrated information system. In *WWW*, 2006.

[3] P. Buneman and J. Cheney. Provenance in databases. In *Proc. of SIGMOD*, 2007.

[4] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *SIGMOD*, 2000.

[5] X. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. http://www.research.att.com/∼lunadong/publication/indep_techReport.pdf.

[6] X. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection from source update history. http://www.research.att.com/∼lunadong/publication/lifespan_techReport.pdf.

[7] H. Guo, P.-Å. Larson, and R. Ramakrishnan. Caching with 'good enough' currency, consistency, and completeness. In *VLDB*, 2005.

[8] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[9] A. Labrinidis and N. Roussopoulos. Exploring the tradeoff between performance and data freshness in database-driven web servers. *VLDB J.*, 13(3):240–255, 2004.

[10] C. Olston and J. Widom. Efficient monitoring and querying of distributed, dynamic data via approximate replication. *IEEE Data Eng. Bull.*, 28(1):11–18, 2005.

[11] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc of IEEE*, 77(2), 1989.

[12] D. Theodoratos and M. Bouzeghoub. Data currency quality satisfaction in the design of a data warehouse. *Int. J. Cooperative Inf. Syst.*, 10(3):299–326, 2001.

[13] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *Proc. of SIGKDD*, 2007.