Information Integration Over Time in Unreliable and Uncertain Environments

Aditya Pal
Dept. of Computer Science
University of Minnesota
Minneapolis, MN, USA
apal@cs.umn.edu

Vibhor Rastogi
Yahoo! Research
Santa Clara, CA, USA
rvibhor@yahoo-inc.com

Ashwin Machanavajjhala
Yahoo! Research
Santa Clara, CA, USA
mvnak@yahoo-inc.com

Philip Bohannon
Yahoo! Research
Santa Clara, CA, USA
plb@yahoo-inc.com

ABSTRACT

Often an interesting true value such as a stock price, sports score, or current temperature is only available via the observations of noisy and potentially conflicting sources. Several techniques have been proposed to reconcile these conflicts by computing a weighted consensus based on source reliabilities, but these techniques focus on *static* values. When the real-world entity evolves over time, the noisy sources can delay, or even miss, reporting some of the real-world updates. This temporal aspect introduces two key challenges for consensus-based approaches: (i) due to delays, the mapping between a source's noisy observation and the real-world update it observes is unknown, and (ii) missed updates may translate to missing values for the consensus problem, even if the mapping is known.

To overcome these challenges, we propose a formal approach that models the history of updates of the real-world entity as a hidden semi-Markovian process (HSMM). The noisy sources are modeled as observations of the hidden state, but the mapping between a hidden state (i.e. real-world update) and the observation (i.e. source value) is unknown. We propose algorithms based on Gibbs Sampling and EM to jointly infer both the history of real-world updates as well as the unknown mapping between them and the source values. We demonstrate using experiments on real-world datasets how our history-based techniques improve upon history-agnostic consensus-based approaches.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases; H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms, Experimentation

Keywords

Information Integration, Semi-Markov, Probabilistic Model

1. INTRODUCTION

While integrating "opinions" from multiple sources, a system is often required to resolve *conflicts*. This arises in a variety of settings, but one common setting is that of an information integration system, in which multiple sources provide information about the same real-world entity. If different, incompatible values are provided for the same attribute, the sources are said to conflict, and the process of resolving the conflict to assign a value is referred to as "corroboration" or "truth finding". This task is frequently challenging, as illustrated by the following example. Consider a restaurant aggregator that seeks to compile a database of restaurants from three feeds, s_1, s_2, and s_3. Let e be *phone* attribute of one business, "Truya Sushi". For example, both sources s_1 and s_2 might report that the current value of e is "555-1234", while source s_3 reports the current value as "555-4444". However, suppose a single value must be chosen to show first for "Truya Sushi".

There exist two broad classes of opinion aggregation techniques – (a) meta-learning approaches [2, 14], which would employ classification/regression with the values provided by each source as covariates, and (b) graph propagation approaches [6, 15], which propagate object properties to source properties and vice-versa via incident edges. Most graph propagation approaches assume source independence, but recently, the source-independence assumption was relaxed by [1] which considered *copying* between sources, and by [3] which considered *internal dependence* between data items. However, all work of which we are aware of has ignored an exceptionally important factor in estimating the value of an unknown variable from conflicting sources, the *history of updates*, as we now illustrate.

Temporal Information Example: Consider two *histories* for e shown in Table 1 and 2. Each scenario shows the values given for e by $s_1 - s_3$ at two times in the past, t_1 and t_2. Sources need not report values at the same time, this is for simplicity of presentation. We now turn to the additional inferences that can be made. The main idea is to change from modeling a single opinion to try to ascertain the *latest true update to e.* To illustrate this idea, consider the first scenario, in which source s_3 has steadily asserted the "555-4444" number, while source s_1 started with "555-4444" and

Source	t_1	t_2	$t_3 = now$
s_1	555-4444	555-8566	555-1234
s_2	555-8566	555-1234	555-1234
s_3	555-4444	555-4444	555-4444

Table 1: Scenario 1

Source	t_1	t_2	$t_3 = now$
s_1	555-8566	555-1233	555-1234
s_2	555-8566	555-1234	555-1234
s_3	555-8566	555-1234	555-4444

Table 2: Scenario 2

s_2 started with "555-8566", but then both s_1 and s_2 evolved to "555-1234". In this scenario, given the greater accuracy scores for sources s_1 and s_2, it is reasonable to output "555-1234" as the value of e. In addition, there is some chance that "555-4444" is an old value and s_3 has never updated. However, consider scenario two. In this scenario, at previous time slices, s_3 *agreed* with s_1 and s_2. In this case, we argue that much more weight should be given to s_3's opinion, since it has either received a new update not yet seen by s_1 and s_2 *or* has accepted an update from a wildly inaccurate source. This illustrates the need to consider what is the *last true update*, rather than the majority alone.

Other uses of history integration include tracking mobile units [17], determining the rate of change of a variable or attribute, estimating the typical lag introduced by each source s, etc. In this paper, however, we focus on the estimation of Z, and leave these applications to future work.

Challenges: The history aggregation problem is hard due to three main challenges:

1. **Missing updates**: Streams can miss updates about the entities. E.g. a restaurant changes its phone number, but a stream retained its old phone number.
2. **Independent error**: Streams can send noisy data. E.g. a restaurant changes its street name to *Picasso street*, but a stream's update read *Pic. st.*.
3. **Arbitrary lag**: Streams can send updates after an arbitrary time. E.g. one stream sent the update within minutes of actual change, while other stream sent an update after a week.

The above three challenges occur in practice due to the fact that the information streams are manually updated and may be reformatted.

Contributions: To our knowledge, this paper is the first to consider the implications of a value update history for truth corroboration.

1. Our first contribution is a formal generative model of corroboration as determining the value of an unknown history of real-world updates Z, based on the observation streams S_k for several sources $k = 1, \ldots, n$. In our model, we assume that updates in a stream S_k are noisy versions of an update in the real world from the past, and this is modeled using a latent mapping vector O_k. Furthermore, we model the fact that streams can miss or repeat the same real world update by imposing a Markovian structure on the mapping vectors.
2. We give algorithms to jointly infer the hidden variables: (i) mappings O_k corresponding to stream S_k, and (ii) true stream Z. We show that when Z is known, then mappings can be optimally estimated using dynamic programming approach. We also show that when O_k is known then Z can be estimated using particle filtering method (and exact inference is possible under some natural functional assumptions). Based on this we give two inference algorithms, (i) EMA that alternates inference of the two hidden variables by fixing one and optimizing the other, converging to a local optima, and (ii) *Gibbs* that samples a set of possible mapping vectors, and infers the best Z for the sampled set.
3. Finally, we demonstrate, using three real world tasks (including one of estimating NFL scores via Twitter), the performance of our joint inference algorithms, and show that our model outperforms existing consensus based approaches that are either history-agnostic, or use history in a naive way.

Outline: Rest of the paper is organized as follows. We describe the history integration problem in Sec. 2. We present our model in Sec. 3 and the inference algorithms in Sec. 4 and 5. Experimental evaluation is discussed in Sec. 6 and 7. Finally we conclude by discussing related work in Sec. 8.

2. HISTORY INTEGRATION PROBLEM

The state changes that occur in any real-world entity can be succinctly described through a temporal sequence. We define a "temporal sequence" as follows.

Definition 1 (Temporal Sequence). *A temporal sequence Φ is a sequence of pairs $[(v_1, t_1), \ldots, (v_m, t_m)]$, such that the following constraints hold:*

$$\forall i \in [1, m), v_i \neq v_{i+1} \tag{1}$$

$$\forall i \in [1, m), t_i < t_{i+1} \tag{2}$$

We use the following notations: (i) $\Phi^V(i) = v_i$ is the i^{th} value in the sequence, (ii) $\Phi^T(i) = t_i$ is the time corresponding to the i^{th} value, (iii) $|\Phi|$ $(= m)$ is the number of entries in Φ, and (iv) $\Phi(1 : i)$ is all the entries of Φ starting from the first pair onwards to the i^{th} pair.

Let $Z = [(v_1, t_1), \ldots, (v_m, t_m)]$ be the temporal sequence that represents the state changes of an entity. Note that since Z captures the changes in the entity value, consecutive values $Z^V(i)$ and $Z^V(i+1)$ are different. We call Z as an entity sequence. Let n streams make independent observations of Z and publish their views in the form of temporal sequences $S_1, \ldots, S_n$. We call them as stream sequences. The stream sequence S can have errors w.r.t Z obtained by applying one or more of the following four operations on Z.

OP$_1$ *Missing updates*: Stream S may not report some of the updates in Z.

OP$_2$ *Noisy/Erroneous observations*: A stream S may report a value $S(j)$ that is different from the observed value $Z(i)$.

OP$_3$ *Duplicate observations*: Stream S may report different noisy observations of the same value $Z(i)$ multiple times (as say, $S(j)$ and $S(j+1)$).

OP$_4$ *Lagged observations*: Stream S may report an observation $Z(i)$ that occurred in the past (i.e., $Z^T(i) \leq S^T(j)$). Nevertheless, we assume that updates in Z are reported in the same order by S.

Definition 2 (Feasible Stream). *If a stream sequence S can be obtained from another sequence Z by the (possibly repeated) applications of one or more of the operations $\{OP_1, \dots, OP_4\}$, then we call the pair (Z, S) as feasible.*

Example: For instance, suppose entity sequence Z is [(Picasso Street, 5/11/11), (Seurat Ave., 7/7/11), (Rembrandt Lane, 10/21/11)]. A stream S could have updates: [(Pic. St., 5/31/11), (Picasso Street, 7/5/11), (Rembrant Road, 11/21/11)]. Notice that, $Z(1)$ has been reported twice, both times with some lag (20 days and 2 months respectively). $S(1)$ and $S(3)$ have mistakes in the reported value. And $Z(2)$ is completely missed by S. Suppose the streams only modify values by making spelling mistakes (small edit distance to $Z(i)$) and abbreviations. Then a stream S' having updates [(Pic. St., 5/31/11), (Van Gogh Straat, 7/5/11), (Rembrant Road, 9/21/11)] is non-feasible because Van Gogh Straat never appears in Z and the last update was reported by S before it was observed in the real world (in Z).

Next we define the history integration problem formally as below.

Problem 1 (History Integration Problem). *Given a set of stream sequences $\mathbf{S} = \{S_1, \dots, S_n\}$, construct an entity sequence $Z^\star$, s.t., $\forall k, (Z^\star, S_k)$ is feasible, and the following holds:*

$$Z^\star = \arg\max_Z \{P(\mathbf{S}, Z)\} \tag{3}$$

where P is a joint probability distribution of the real observed updates in Z and reported values in the streams $\mathbf{S}$. In the next section, we explicitly model P by assuming a generative model for the streams.

3. OUR MODEL

We assume a first-order semi-Markovian prior over the true updates Z, where $P(Z(i+1)|Z(i))$ represents the transition probability for the value and the time of the $(i+1)^{th}$ update given the value and the time of the i^{th} update, independent of the first $i-1$ updates. Note that this is semi-Markovian since the probability for the duration $Z^T(i+1) - Z^T(i)$ of update i can be a general distribution, rather than a geometric distribution.

We would like to note that our prior can capture *domain knowledge* about the true updates. For instance, one can encode the fact that the score of an NFL game typically does not decrease, and usually changes in increments of 3 and 7 (and less frequently by 6 and 8), using a transition model that assigns very small probability to a score transition of $Z^V(i+1) - Z^V(i) = \delta$, when δ is not 0, 3, 6, 7, or 8. We use such a prior in our experiments. Similarly one can encode the domain knowledge that the temperature at a place seldom drops by 20 degrees in a minute. As a final example, businesses are either open or closed, and typically, an open business may close but the reverse does not happen. This can be encoded using a two state Markov model, where the close to open transition probability is very small.

In order to model feasible streams that may miss/repeat true updates, and report updates with lag, we propose the concept of a *mapping vector*.

Definition 3 (Mapping Vector). *For a given stream S, the mapping vector O is a vector having entry $O[j] = i$, iff the update $S(j)$ observes $Z(i)$. More formally, we say that $S^V(j)$ is conditionally independent of all other variables given $Z^V(O[j])$, and write*

$$P(S^V|Z, O) = \prod_{j=1}^{|S|} P(S^V(j)|Z^V(O[j])) \tag{4}$$

From its definition, O has to obey the following constraints:

1. *$|O| = |S|$; i.e., the length of the mapping vector is same as the length of the stream.*
2. *$\forall j, 1 \le O[j] \le |Z|$; i.e., mapping vector maps updates in S to updates in Z.*
3. *$\forall j, \forall k$, s.t., $j \le k \implies O[j] \le O[k]$, i.e., streams report updates in the same order as observed in Z.*
4. *$\forall j, S^T(j) \ge Z^T(O[j])$; i.e., a stream can not publish an update that has not yet occurred in Z.*

Note that the above mapping vector constraints ensures the feasibility condition (see Defn. 2) for the streams. We consider a set of mapping vectors, $\mathbf{O} = \{O_1, \dots, O_n\}$, one for each stream.

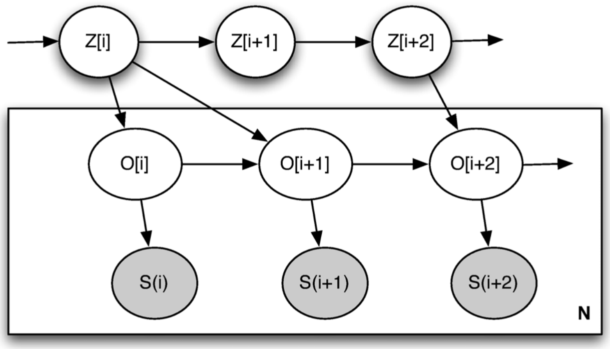


Figure 1: Probabilistic Model for the History Integration Problem. The prior on Z, the true updates, is semi-Markovian. The streams are generated by first creating a mapping vector O, and then sampling noisy values according to $Z(O(j))$.

Generative model for S. Next we describe our model of how the streams $\mathbf{S}$ are generated for a entity sequence Z. First, we assume that the streams are conditionally independent given Z, so we can describe the generative model for $S_k \in \mathbf{S}$ independent of the other streams for the given Z. Second we assume that S_k^V and S_k^T are independent of each other given Z and O_k (i.e. random noise and lags are independent of each other). Finally, as explained in Eq. 4, we assume that $S_k^V(j)$ is conditionally independent of all other variables given $Z^V(O_k[j])$. Under these assumptions, to generate S_k from Z, first the mapping vector O_k is generated, and then the j^{th} update in S_k is generated using the $O_k[j]^{th}$ update in Z, as explained in the following steps.

1. **Modeling misses/repeats:** The j^{th} entry of O_k is generated using the $(j-1)^{th}$ entry and Z. Thus probability distribution over $O_k[j]$ depends only on $O_k[j-1]$ and Z, and is denoted as $P(O_k[j]|O_k[j-1], Z)$.

2. **Modeling lags:** Time of j^{th} update in S_k is generated using the $(j-1)^{th}$ update and the time of $O[j]^{th}$ update of Z. Thus probability distribution over $S_k^T(j)$ depends only on $S_k^T(j-1)$ and $Z^T(O_k[j])$, and is denoted as $P(S_k^T(j)|S_k^T(j-1), Z^T(O_k[j]))$.
3. **Modeling noise:** Value of j^{th} update in S_k is generated using the value of $O[j]^{th}$ update of Z. Thus probability distribution over $S_k^V(j)$ depends only on $Z^V(O_k[j])$, and is denoted as $P(S_k^V(j)|Z^V(O_k[j]))$.

Figure 1 illustrates this generative probabilistic model pictorially. Given this generative process for $\mathbf{S}$, we can write the joint probability distribution of $P(\mathbf{S}, Z)$ as follows.

$$\begin{aligned} P(\mathbf{S}, Z) &= \sum_{\mathbf{O}} P(Z)P(O_k|Z)P(S_k^T|O_k, Z)P(S_k^V|O_k, Z) \quad (5) \\ &= \sum_{\mathbf{O}}[P(Z) \qquad \textit{(domain knowledge)} \\ &\times \prod_{k=1}^{n}\prod_{j=1}^{|S_k|} P(O_k[j]|O_k[j-1], Z) \quad \textit{(miss/repeat)} \\ &\times \prod_{k=1}^{n}\prod_{j=1}^{|S_k|} P(S_k^T(j)|S_k^T(j-1), Z^T(O_k[j])) \quad \textit{(lag)} \\ &\times \prod_{k=1}^{n}\prod_{j=1}^{|S_k|} P(S_k^V(j)|Z^V(O_k[j]))] \qquad \textit{(noise)} \end{aligned}$$

The key assumptions made in our generative model and equation 5 is that given the complete history of the entity (Z) and the mapping of observed variables to Z (essentially $\mathbf{O}$), the stream update values ($\mathbf{S}^V$) are independent of each other. This is true in most, but not all, practical scenarios. For eg., some sources might be copying each other (see [3]), and hence be correlated, but we assume, for the model's simplicity, that copying sources have been detected, and removed from the analysis.

4. GENERAL INFERENCE ALGORITHM

Recall that the history integration problem is to compute $Z^\star = \arg\max_Z\{P(\mathbf{S}, Z)\}$, where $P(\mathbf{S}, Z)$ is given by Equation 5. Here we describe our inference techniques to compute $Z^\star$. This involves finding the set of updates Z^V as well as the times when these updates occurred Z^T. We start this section by describing a special case, where the mapping vector $\mathbf{O}$ is known. Then using the solution for the special case, we develop two approximate inference algorithms for the entire problem – Gibbs, an approach based on Gibbs Sampling [4], and EMA, an approach based on Expectation-Maximization. In all approaches, the overall idea is to iteratively find a mapping vector O (that represents missing/repeat updates) based on the current estimates of Z, and then conditioned on the current O optimize for Z^V and Z^T (accounting for lags and noise).

4.1 Mapping Vector is Known

Assume that we know the set of mapping vectors $\mathbf{O}$ then the problem is to maximize $P(Z|\mathbf{S}, \mathbf{O})$, where $Z(i)$ forms the hidden state. Furthermore, since each O_k maps the updates in a stream S_k to the updates in Z, the hidden states in Z form a HSMM with transitions dictated by the prior on Z and the emission from $Z(i)$ given by the corresponding stream reports $S_k(j)$, where $O_k[j] = i$. Note that though this special case of problem with fixed $\mathbf{O}$ is similar to the one of HSMM inference, it has many distinguishing characteristics. First, not every hidden state $Z(i)$ emits an observation; we have to deal with missing values. Moreover, some hidden states emit multiple observations (when multiple stream updates are mapped to the same $Z(i)$). Second, $Z(i)$ values are not discrete – in many of our experiments the Z^V values are modeled using Gaussian distributions, and in general the transition and emission probabilities can be any distributions from the exponential family.

We address missing updates by allowing hidden states to emit a special *null* observation that has equal probability of being emitted by each of the hidden states. To address multiple emissions, we think of the multiple observations as a single observation emitted with probability equal to the product of the emission probability of each observation (as they are conditionally independent given the hidden state).

Once missing/multiple updates are addressed, in the discrete case, Z can be inferred exactly using standard Viterbi algorithm for HMM(HSMM) inference. In the continuous case, one can use the general technique of *particle filtering*, also known as sequential monte carlo, for inferring the hidden values of Z. However, it is not possible to perform exact inference for general prior distributions of Z, and one can use *sequential importance sampling* [5] to perform an approximate inference of Z.

4.2 Gibbs Inference Algorithm

Here we consider the problem when the mapping vectors $\mathbf{O}$ are unknown. In the Gibbs algorithm, we begin by sampling a random $\mathbf{O}$ from the prior $P(\mathbf{O}|Z)$; this prior depends on the length of Z alone. Since we do not know $|Z|$, we generate sample $\mathbf{O}$'s for various values of $|Z|$ and we pick the one with the maximum likelihood. Starting with this initial $\mathbf{O}$, we iteratively find the Z, with the maximum $P(Z|\mathbf{S}, \mathbf{O})$ using techniques from Sec. 4.1 for a fixed $\mathbf{O}$, and then find a new $\mathbf{O}$ by sampling from the posterior distribution of $P(\mathbf{O}|\mathbf{S}, Z)$ as follows. For all streams k:

$$P(O_k|\mathbf{O}_{-k}, \mathbf{S}, Z) \propto P(O_k|Z) \cdot P(S_k|Z, O_k) \qquad (6)$$

The above sampling distribution can be used to sample O_k for all streams k. Z can be recomputed from the sampled $\mathbf{O}$ and the process can be repeated either until convergence or until a maximum number of iterations are performed.

4.3 EMA Inference Algorithm

The Gibbs algorithm can take many steps to converge. An alternative is to use the Expectation Maximization technique. When Z is known, we can estimate an optimal $\mathbf{O}$, such that $P(\mathbf{O}|\mathbf{S}, Z)$ is maximized. This can be done independently for all streams, as $P(\mathbf{O}|\mathbf{S}, Z)$ is

$$\begin{aligned} \propto \prod_{k=1}^{n}\prod_{j=1}^{|S_k|} & P(S_k^V(j)|Z^V(O_k[j])) \cdot P(S_k^T(j)|Z^T(O_k[j])) \\ & \cdot P(O_k[j]|O_k[j-1], Z) \qquad (7) \end{aligned}$$

where we used equation 5 to get above factorization. We can use Viterbi algorithm [12] to get the best mapping (see algorithm 1). EstimateMapping algorithm exploits the optimal sub-structure property and runs in $O(m \cdot \ell^2)$ time, where ℓ is the length of Z and m denotes the number of updates in a stream. $O_k = \text{EstimateMapping}(Z, S_k)$ gives mapping for

Algorithm 1 EstimateMapping(Z, S)

```
{$\pi, \sigma$ model missing update and $\psi$ models noise and lag}
Let $\pi(i) = \log P(O[1] = i|Z)$
Let $\sigma(j, i, k) = \log P(O[j] = i|O[j-1] = k, Z)$
Let $\psi(j, i) = \log P(S(j)|Z(i))$

$l = |Z|, m = |S|$
for $i = 1$ to $l$ do
  $c[1, i] = \pi(i) + \psi(1, i)$
end for
for $j = 2$ to $m$ do
  for $i = 1$ to $l$ do
    $r = \arg\max_r\{c[j-1, r] + \sigma(j, i, r) : \forall r \in [1, i]\}$
    $c[j, i] = c[j-1, r] + \sigma(j, i, r) + \psi(j, i)$
    $d[j, i] = r$
  end for
end for
$O[m] = \arg\max_i\{c[m, i] : 1 \leq i \leq l\}$
for $j = m-1$ to $1$ do
  $O[j] = d[j+1, O[j+1]]$
end for
Return $O$
```

stream k. The EMA algorithm takes an initial Z and then alternates between finding $\mathbf{O}$ and Z iteratively. These iterations are repeated until a fixed point is reached. The EMA algorithm works with the intuition that in any iteration, the perturbations in $\mathbf{O}$ would improve the likelihood of the subsequent Z. Note that EMA algorithm aims at finding local maxima Z, $\mathbf{O}$ for the probability distributions $P(Z, \mathbf{S}, \mathbf{O})$. The key problem with the EMA algorithm is that it can get stuck in a local optima, and its performance is critically dependent on the quality of initial Z. One way to alleviate this is to choose a good initial Z_{init} using Gibbs and then run the EMA algorithm starting from Z_{init}; we denote this hybrid algorithm as $Gibbs +$ EMA.

5. INFERENCE FOR A NATURAL INSTANTIATION

In this section, we describe an instantiation of our general model, where we assume natural functional forms for the various probabilities. Our choices allow us to tractable solve the history integration problem. We show in our experiments that this specific model works well on a number of datasets. We first describe the distributions used to model domain knowledge, missing/repeat updates, lags and noise (from Equation 5). We then present tractable algorithms for estimating the most likely $Z^\star$ given $\mathbf{O}$ as well as for sampling from the posterior distribution of $P(\mathbf{O}|\mathbf{S}, Z)$. Finally we conclude this section with a brief note on parameter learning for our natural instantiation.

5.1 Instantiation

Domain Knowledge $[P(Z)]$**:** We assume that the prior on Z factorizes such that $Z^V(i)$ only depends on $Z^V(i-1)$, the previous different value in Z, and $Z^T(i)$ only depends on the time of the previous update $Z^T(i-1)$. We use the Exponential distribution to model the intervals between consecutive updates: $P(Z^T(i) - Z^T(i-1) = \tau) \propto exp(-\gamma_Z \cdot \tau)$. When Z^V values are continuous, we use a Gaussian prior $Z^V(i) \sim N(Z^V(i-1), \sigma_Z)$. The first element, $Z(1)$, of Z is considered to have a uniform prior.

Missing/Repeated Updates $[P(O_k[j]|O_k[j-1], Z)]$**:** The

Algorithm 2 Estimating Z^T

```
$Z^T(|Z|) = min\{S_k^T(j) : \forall k, \forall j, O_k[j] = |Z|\}$
{If no mapping for last element of $|Z|$, then set $|Z| = |Z| - 1$
and re-run algorithm}
for $i = |Z| - 1$ to $1$ do
  if $\forall k, \forall j, O_k[j] \neq i$ then
    {No mapping vector, cant estimate time of update}
    $Z^T(i) = Z^T(i+1) - \epsilon$
  else
    $Z^T(i) = min\{S_k^T(j) : \forall k, \forall j, O_k[j] = i\}$
    if $Z^T(i) \geq Z^T(i+1)$ then
      $Z^T(i) = Z^T(i+1) - \epsilon$
    end if
  end if
end for
Return $Z^T$
```

difference between consecutive indices of a mapping vector $O_k[j]$ and $O_k[j+1]$ determine whether an update in Z is repeated/missed. We model the distance between $O_k[j]$ and $O_k[j+1]$, independent of Z, using a Poisson distribution: $P(O_k[j+1] - O_k[j] = x) \propto e^{-\lambda}\lambda^x/x!$. This is to model the fact that a stream typically gets one out of every λ update.

Lags $[P(S_k^T(j)|S_k^T(j-1), Z^T(O_k[j]))]$**:** We assume that $S_k^T(j)$ is independent of $S_k^T(j-1)$ and model the lag, i.e. $|S_k^T(j) - Z^T(O_k[j])|$, using an Exponential distribution that penalizes streams with large reporting delays: $P(S_k^T(j) - Z^T(O_k[j]) = \tau) \propto exp(-\gamma_k \cdot \tau)$.

Noise $[P(S_k^V(j)|Z^V(O_k[j]))]$**:** There is no one model for the way values in the stream are mis-reported by streams. this is very application specific and depends on the kind of data being reported. We present the following two example instantiations. When values being reported are continuous (like in the weather readings or trading volume), one can use Guassian noise: $S_k^V(j) \sim N(Z^V(O_k[j]), \sigma_k)$. When the reported values are discrete (like phone numbers, or football scores), one may use following simple noise model: $S^V(j)$ is the same as $Z^V(O[j])$ with probability p_k, and different with probability $1 - p_k$.

5.2 Estimating Z given $\mathbf{O}$

In this section, we provide efficient algorithms to extend Sec. 4.1 for finding Z given O on the specific distributions instantiated in the previous section. Since the prior on Z factorizes into independent terms containing only Z^T and Z^V, we can optimize for the times and values independently.

Finding Z^T**:** When the lag follows an Exponential distribution, i.e. $P(S_k^T(j) - Z^T(O_k[j]) = \tau) \propto exp(-\gamma_k \cdot \tau)$, we can exactly compute the time of the i^{th} update in Z, by solving the following minimization

$$\min\{\gamma_z[Z^T(|Z|) - Z^T(1)] + \sum_{\substack{\forall i, \forall k, \forall j \\ O_k[j]=i}} \gamma_k[S_k^T(j) - Z^T(i)]\} \quad (8)$$

Subject to following constraints

$$Z^T(i-1) < Z^T(i) < Z^T(i+1)$$

$$Z^T(i) \leq min\{S_k^T(j) : \forall k, \forall j, O_k[j] = i\}$$

In general, Algorithm EstimatingZ^T provides a valid solution to Z^T that satisfies the required constraints. Under certain conditions it is provably optimal, i.e. minimizes Eq. 8 as shown in the lemma below.

LEMMA 5.1. *Algorithm 2 always returns a feasible solution to Equation 8, and the solution is guaranteed to be optimal when $\epsilon \to 0$ and $\gamma_z \leq \gamma_k$, $\forall k$.*

If $\lambda_z > \lambda_k$ then we can use Z^T as initial solution and sample $Z_1^T(i)$ uniformly in the range $[Z_1^T(i-1), Z^T(i)]$ and pick the solution that minimizes Equation 8.

Finding Z^V for continuous values: When Z^V follows an Gaussian distribution, i.e. $P(S_k^V(j) - Z^V(O_k[j]) = \tau) \propto exp(-\beta_k \cdot \tau^2)$, we can exactly compute the values of the i^{th} update in Z, by solving the following equation.

$$Z^V(i) = \frac{\beta_z[Z^V(i{+}1) + Z^V(i{-}1)] + \sum_{\substack{\forall k,\forall j\\ O_k[j]=i}} \beta_k \cdot S_k^V(j)}{2\beta_z + \sum_{\substack{\forall k,\forall j\\ O_k[j]=i}} \beta_k} \quad (9)$$

where $\beta = 1/\sigma^2$ is the precision of the Gaussian distribution. Above equation is obtained by taking the Equation 7 on log scale and taking its derivative w.r.t $Z^V(i)$ and equating it to 0. This is similar to Kalman Filter, and we can estimate the values in Z using an iterative algorithm.

Finding Z^V for discrete values: For discrete values, in the absence of a prior on Z, estimating $Z^V(i)$ corresponds to finding v such that the probability $\prod_{O_k[j]=i} P(S_k^V(j) = v)$ is maximized. We can show that v is just the majority update amongst all the stream updates that map to i

Parameter learning. Given Z and stream observation S_k, we can compute the error incurred by the stream. The standard deviation of the error is our new estimate of σ_k. Similarly the lag for all the observations can be used to compute most likely γ_k and the number of missing updates can be used to compute most likely λ_k. This falls naturally from our underlying assumption that stream's noise, lag and missing characteristics are independent.

6. DATASET DESCRIPTION

We experimentally evaluate the performance of our model on four datasets. These datasets from diverse domains exhibit varied stream characteristics and thus validate the wide applicability of our model.

Twitter Dataset. We used Twitter api to extract all the tweets posted on Twitter regarding NFL[1] games, such as, *Jets vs Charger*, *Raven vs Jaguar*, *Packers vs Vikings*. These games were played in October, 2011. From the collected tweets, we removed all the tweets that did not mention the game score in the format `\d+ - \d+`. We also discarded tweets containing words such as "predict" as these tweets were found to be speculative and were not a representative of the actual game score. The resulting dataset consists of tweets from 20 users for *Jets vs Chargers* game, 37 users for *Raven vs Jaguar* game, 23 users for *Packers vs Vikings* game. The extracted tweets were of the order of number of users, as users rarely posted score more than once. Our goal in the experiment is to construct the game score timeline

[1]http://www.nfl.com/

# updates per game	7.3
Num streams (users) per game	26.67
# tweets per update per game	2.9
Relevant tweets per game	20
Non-relevant tweets per game	8
Average stream lag (sec)	1388.4
# Missing update per stream	6.67 / 7.3

Table 3: Twitter dataset characteristics aggregated for all games.

with this dataset. Even thought the dataset looks small, it is extremely challenging due to fact that the underlying streams (users in this case) are missing a lot of updates, they have a large lag and many irrelevant tweets (see Table 3).[2]

Climate Dataset. We consider the temperature data provided by Earth System Research Laboratory, USA. The data, which is available for public download[3], consists of monthly temperature means for all the locations on Earth from 2001 to 2010. The dataset consists of 10512 gridded locations (with a grid resolution of 2.5° longitude x 2.5° latitude) and a temperature series of length 120 (one entry per month).

Our goal here is to estimate the series of mean temperatures for the 120 months between 2001 and 2010 of a given location based on its neighboring locations. This is a reasonable goal due to high spatial auto-correlation present in the Earth science data. The neighboring locations can be viewed as noisy streams with no lag and missing update. We simiulate missing updates by deleting temperature reading from all stream at random points in time. Random lags is simulated by adding $\lambda_k * r$ to the time of the update, i.e. $S_k^T(i) = max\{S_k^T(i-1), 20 \cdot i\} + \lambda_k * r$, where r is a random number between $[0, 1]$ and λ_k is set randomly to an integer in the range $[20, 100]$. Note that this dataset is challenging as the stream's noise (unknown to our model) varies due to several environmental factors such as topology, height, latitude, longitude of the selected locations.

NASDAQ Dataset. Our third dataset is NASDAQ's volume trading data.[4] The dataset contains 5 minute updates of the trading volume that occurred from Jan 2008 to Dec 2010. We generate synthetic stream observations for this dataset, using Guassian noise, Exponential lag and Poisson missing/repeated updates as described in Section 5. Stream generation parameters are not known to our algorithms.

This dataset is interesting for two reasons: (1) It is very hard to come up with a prior for the volume trading data. As a result the output of the model is completely dependent on the stream emissions. (2) Trading data has several changes with small amplitude. Baseline models can be tempted to combine several different updates together due to the close proximity of values. Our model does not suffer from this as stream observations can have random lags and they are penalized if two seemingly different updates (in terms of time) are being grouped together.

[2]For instance, we saw tweets corresponding to NHL scores involving a team which have the same name as a NFL team (Winnipeg Jets and New York Jets).

[3]http://www.esrl.noaa.gov/psd/data/

[4]http://www.eoddata.com/products/historicaldata.aspx

Synthetic Dataset. We also evaluate our model on a synthetically generated data. We consider $Z^V = [1, 2, \ldots, m]$, where m is varied from 10 to 100 and consider 5-15 streams. Unlike NASDAQ and Climate data, the time of updates (Z^T) is randomly initialized to non-regular spaced intervals (e.g. $Z^T = [1, 3, 10, 38, 39, \ldots]$). The stream generation process is as follows: A stream would pick elements of the Z vector sequentially and could perform the following three operations: a) *Simulate missing update:* Ignore the picked element and move to the next element with Bernouilli probability $= pmiss_k$, b) *Simulate independent error:* Add Gaussian noise with precision $\beta_k > 1$, c) *Simulate Lag:* Publish the noisy update after lag governed by Uniform distribution in the range $[1 - 10]$. Note that streams for synthetic data differs from NASDAQ data in terms of the lag and the missing update distributions.

Dataset Summary. The above four datasets present different challenges to the model. Twitter dataset presents an interesting aspect of how the model performs for sparse datasets. On the other hand, the climate dataset presents challenges since the underlying noise is difficult to model[5]. NASDAQ dataset presents challenges due to the small variability between adjacent updates. Finally, with the synthetic dataset we present more extensive analysis of our model under several different conditions.

7. RESULTS AND EXPERIMENTS

In this section, we present our results over the four datasets. The main objective of our experiments is to show that our model can be adapted to run quite nicely for several different domains and under different conditions. The second objective is to show that in comparison to an intuitive baseline model it performs better as it models all the parameters that a stream can exhibit while reporting the updates.

7.1 Algorithms

We compare our algorithm against three baseline approaches.
Our: We use the hybrid algorithm *Gibbs* + EMA that is described in Section 4 which runs a few iterations of Gibbs to find a Z and then runs EMA.
B1 (PickLastUpdate): This baseline picks the last update across all streams as the true state of entity at that time. This algorithm performs a merge sort of updates based on ascending time order.
B2 (PickMajorityUpdate): This algorithm picks the majority of the stream updates at any given time. In order to compute the merge stream it uses the following procedure: Consider that a stream published an update at time t. We collect the last update for all streams that was published at time t or before it. The majority value amongst these updates is set in the merge sequence at time t. This is done for all t.
B3 (OptimalAlignment): This algorithm is based on the intuition presented in [17], where streams can have fixed lags. The algorithm considers the optimal sequence alignment strategy to evaluate all possible alignments and then fixes $\mathbf{O}$ to compute Z that maximizes the joint-likelihood.

[5]Climate models that are used to extrapolate climate data over missing points on Earth such as over Oceans, etc are extremly sophisticated and take into account tens of variables.

	Precision	Recall	F-measure	# Last correct
Our	**0.93**	0.68	**0.79**	3
B3	0.77	0.71	0.74	1
B2	0.49	**1**	0.66	0

Table 4: Precision recall of our model vs the best baseline model averaged over three games.

Baseline 3 performs better than the other two baselines for the selected datasets and hence for the sake of clarity we only present results with this baseline.

7.2 Results on Twitter Dataset

We treat each user as an independent stream providing updates about the game score. For two teams playing a game, users can post score of team1 first (7-0) or team2 first (0-7), hence while computing Z, we consider both permutations of the scores and pick the configuration with least noise. We used $Poisson(\lambda = 5)$ to model missing updates and $Exponential(\gamma = 10)$ to model stream lag. The merging strategy is to consider the majority element amongst the stream updates that map to same index in Z (since scores can be thought of as discrete values). Once the best Z is obtained, model parameters are recomputed and streams with relatively small likelihood $P(S|Z, O)$ are discarded and Z is recomputed. This is repeated for 2-3 rounds.

Figure 2 and 3 shows updates computed by our algorithm along with the corresponding mapping tweets. Even though dataset was very sparse, with many streams having one or two updates, it was able to retain the key updates that constitute the real timeline of the game along with an accurate mapping of the tweets. We also discovered that the model is able to correctly discard the set of tweets that were not related to the game score.

Comparison with Baseline models. Next we compare the timeline predicted by our model with baseline models B2 and B3 (using the majority merging strategy). We define *precision* as the fraction of updates output by the algorithm that are correct. Similarly, we define *recall* as the fraction of actual updates of true timeline that are output by the algorithm. Additionally, we also check whether the last update output by each algorithm is correct.

Table 4 shows the performance of our model in comparison with the best baseline model aggregated for all the selected game datasets. Overall we see that our model has higher precision and F1 over the best baseline model. We also observe that the last update of Z presented by our model is actually correct whereas this is not the case for the baseline. This happens because after the end of the game, some tweets mention the number of wins and losses for a team, and the baseline model would consider that as an update. On the contrary, our model correctly discards such updates from Z.

7.3 Results on Climate Dataset

We selected several locations randomly for different latitudes and longitudes and considered their temperature series. For each selected location, we considered 8 adjacent neighbors to it as 8 streams. Typically locations are roughly 250 kms apart despite being neighbors due to 2.5° resolution. The mean of absolute temperature difference between 8 neighbors to a location is 1.8 (on average) with a mean standard deviation of 0.93. Our model is run with

	(13:09) UGH terrible strip, 7-0 hole. Need to score now. #jets	(14:10): Mark Sanchez to Plaxico Burress for a 3yd score to bring the #Jets closer to the #Chargers...14-10	(16:05): *WHAT THE HELL #Chargers!!! You're up 21-10 and now you're losing 24-21 and the #Jets are ready to score again?? SMH.*		(16:45): 27-21 JetsRT "@bonniblucakes: What was the score #jets ???"
	(13:06): Lmaoooooo dammmmmn Keller coughed up da ball #Chargers score thanks 2 Butler 7-0 #chargers	(14:09): Sanchez to Burress. #Jets score TD but trail #Chargers 14-10.	(14:26): #Chargers score 21-10	(15:47): Sanchez to Burress again, 3-yard score, #Jets in front 24-21, 8:41 left	(16:17): #Jets blank #Chargers in second half, score 17 unanswered to win, 27-21. What a game. #NYJ
Score	7-0	14-10	21-10	21-14	21-27

Figure 2: *Jets vs Chargers* game score output by our algorithm along with the corresponding tweets.

	(21:16): How the **** is the score still 0-0 ? C'Mon #ravens	(21:23): #Jaguars score the first points of the game with the field goal, up 3-0 over #Ravens with 1:42 left in the first quarter	(23:39): **** score is 9-0? What is this baseball? Need more offense #ravens and #jaquars. This is the #NFL -_-	(23:47): Baltimore #Ravens finally score a TD. #Jaguars lead cut to 9-7 with 2:02 left.	(01:58): Just checked the #Ravens score en route to work... 12-7 to JACKSONVILLE?! Are you kidding me?! Woeful. Utter disgrace.
	(21:18): 0-0. Ravens hvn't showed up though. Defense got a fumble tho.	(21:23): #Jaguars score first 3-0 over #Ravens.	(22:51): #Jaguars up 9-0. In this game, that's a three-score lead.	(23:52): 7-9	(00:59): #Jaguars defeat the #Baltimore #Ravens by a score of 12-7 on #MNF! ...
Score	0-0	0-3	0-9	7-9	7-12

Figure 3: *Raven vs Jaguars* game score output by our algorithm along with the corresponding tweets.

Gaussian($\sigma = 1$) noise, Exponential($\gamma_k = 10$) lag, Poisson ($\lambda_k = 0.1$) missing update. We also consider a prior on length of Z, as we aim to get a update vector of same length as true Z. Once model computes Z, stream parameters are re-estimated and few more such rounds of algorithm is run.

We first compare the quality of a Z output by *Gibbs* + EMA and baseline B3 with respect to the true temperature series $Z^\star$ as the number of missing updates per stream increases. Quality is computed as the sum of $(Z^V(m) - Z^{\star V}(m))^2$ over all times (months) m for which Z outputs an update. Figure 4(a) shows *Gibbs*+EMA outperforms the baseline B3.

7.4 Results on NASDAQ Dataset

Figures 4(b) shows the performance of our model in comparison with the best baseline B3 over the NASDAQ. We plot the log of negative log-likelihood due to scale of the values, and so lower value implies that model has higher likelihood. The likelihood of the true Z that generated the data is also plotted for absolute comparisons. We see that *Gibbs*+EMA performs statistically significantly better than the best baseline (paired ttest, $p < 0.01$, 99% CI). Moreover, the average absolute error for last element of Z (i.e. current state of the entity) was at least 10 times lower for *Gibbs* + EMA compared to the best baseline B3.

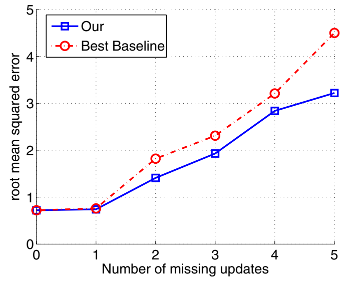
7.5 Results on Synthetic Dataset

Figure 4(c) shows the performance of *Gibbs* + EMA in comparison to B3 over the synthetic dataset. Like in the NASDAQ data, we observe that our model performs much better than B3. We perform further controlled experiments on the synthetic data to better evaluate our algorithm.

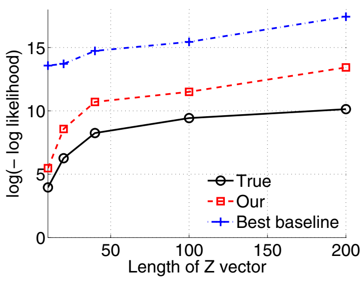
Comparison between Gibbs vs Gibbs+EMA: Figure 5 shows the model performance of the Gibbs and *Gibbs*+EMA algorithms for various values of $|Z^\star|$. We see that *Gibbs* + EMA slightly improves the performance of the model in comparison to stand alone Gibbs.

Effect of variation in length of Z and the number of streams: Figure 6(a) plots the quality of the Z output by *Gibbs* + EMA as the number of streams increases. Each line corresponds to a true $Z^\star$ of different lengths. Quality is measured over the values as $\sum_i \left\{(Z^V(i) - Z^{\star V}(i))^2\right\}^{\frac{1}{2}}$. We can clearly see that the absolute error in the prediction increases as the length of $Z^\star$ increases. This intuitively makes sense as we expect the error to increase for higher length Z (as more iterations might be required for mixing of O and inference). We also see that as more streams are added, the error sharply goes down. This also makes sense as adding more streams decreases the probability of missing updates. Additionally, we observe that even though the underlying streams imperfect their aggregation is quite robust to noise, lags and missing updates.

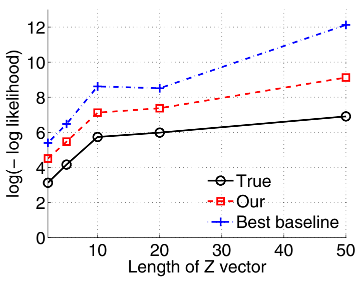
Effect of One Good Stream: In several practical scenarios, there is one (or a few) good stream(s); i.e. streams with small lag, low error and small miss probability. We simulate such a scenario by using a goodness criteria (g), such that, a few good streams miss updates with probability e^{-g}, have lags with parameter e^{-g}, and noise with standard deviation e^{-g}. The rest of the streams are bad – miss updates with probability picked uniformly between 0 and 1, have lags with parameter 10 and noise with standard deviation 1. We call stream goodness to be e^{-g} and as g is increased, streams



(a) Variance of our model's Z w.r.t. true Z over the climate dataset.



(b) Model comparison over NASDAQ dataset.



(c) Model comparison over Synthetic dataset.

Figure 4: Performance on Climate, NASDAQ and Synthetic data

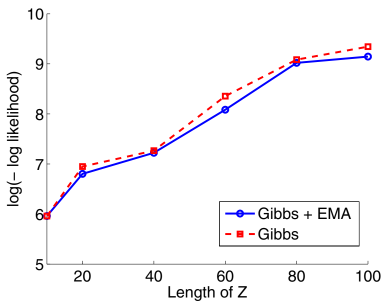


Figure 5: Performance of Gibbs sampling algorithm vs Gibbs+EMA.

are even better. In this scenario, we run our algorithm and compute the mean squared error of the output with the true Z. Figure 6(b) shows the mean squared error on log scale. For the sake of clarity, we do not plot the performance of the baseline models as they either perform equivalent or worse as compared to our model. We observe here that as stream goodness is increases the error decreases. The result also indicates that as more streams turns good then the error further decreases (log-linear).

Effect of One Good Parameter: Another special case is when one stream is good on only one of the parameters (noise, lags or missing/repeated updates). To simulate this, we consider the stream goodness parameter (as discussed in previous result) and make one stream miss update with a very small probability, other with a small lag and a third with small noise. Figure 6(c) shows the performance of the model in comparison where one stream has all the goodness parameters. We see that when goodness is low then the errors are relatively close. But when goodness increases then the gap between errors increase. This happens because, we observe that if a stream which is less noisy but misses a lot of updates, then algorithm relies more on the updates presented by other stream, whereas a stream which is good in all the criteria practically dictates the inference of Z.

8. RELATED WORK

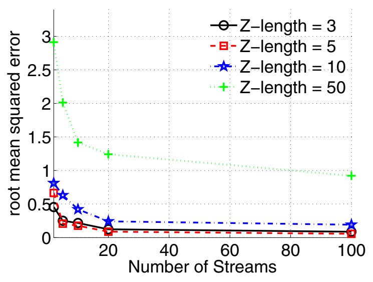
The history integration problem is most related to the following three fields – temporal data integration, multiple sequence alignment and reconciliation of values in static data. We review each of these fields.

Temporal Data Within temporal data integration, perhaps the work most closely related to ours is [17] that studies inference for the purpose of mobile location tracking. They also model the hidden variable (user's actual location), and have multiple observations streams, but with fixed (yet unknown) lags. That model makes sense for the mobile location setting, since the lag of sensor's reading to the real world would be fixed, but might not be correct for scenarios discussed in this paper, when a single source may have varying lags for different observations . Due to the assumption of fixed lags, their optimization problem is technically simpler, and can be solved by trying out all possible variations of the fixed lag (i.e. the O_k mapping). This strategy would be too inefficient for our problem as the number of possible O_k mapping vectors is exponential in $|S_k|$, the number of observations in the source stream. Apart from mobile domain, HSMM based models have been used in several major applications such as speech recognition, handwriting recognition, network traffic modeling, and functional MRI brain mapping. Zheng et al. [16] presents in detail how inferencing is done in HSMM (also its variants) and presents a summary of its applications in several domains.

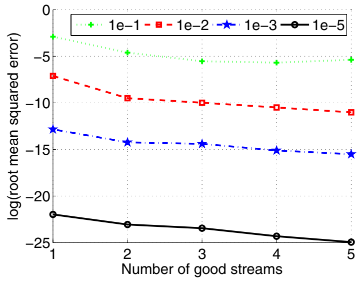
Temporal data is often aggregated in sensor networks with two critical differences: (i) Lags for sensor reading are assumed to be known (O_k is known), and (ii) data is often aggregated (rather than integrated) as the domain for the hidden variable (e.g. temperature) is often continuous, e.g. [8].

Finally, a recent paper [7] considers the problem of deduplicating entities from a temporal stream of updates. While their techniques model the evolution of an entity's attribute value, their focus is to cluster the temporal updates corresponding to the same entity, and not compute the correct value of the entity's attribute.

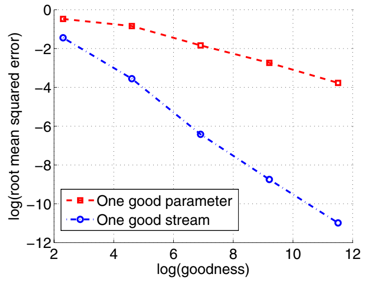
Multiple Sequence Alignment The goal of multiple sequence alignment [9, 10, 11, 13] is to construct an alignment of symbols in n sequences by adding gaps between consecutive characters in the sequence. There is a penalty for adding gaps, and a penalty when two sequences have differing characters at some position; an optimal alignment minimized the total pairwise penalty across all sequences. While the history integration problem seems very similar to the alignment problem (since we are trying to align stream updates to real world updates using a mapping vector), there are key differences between the two. For instance, we explicitly model time in our problem. One could think of modeling time im-



(a) Effect of varying length of Z and number of streams on absolute error.



(b) Effect of good stream on the squared error, where stream goodness varies from $1e-1$ to $1e-5$.



(c) One good stream vs One good parameter per stream

Figure 6: Experiments on Synthetic data

plicitly in the alignment problem by discretizing times and replaying the same value $Z^V(i)$ for all times between $Z^T(i)$ and $Z^T(i+1)$, but this no longer is flexible enough to model all kinds of missing updates and lags.

Reconciliation of Static Data We also mention here some of the static techniques that do not look at historical updates for integration. Thus they cannot directly be applied for history integration, but we mention them for the sake of completeness. In his seminal paper, Kleinberg introduced the hubs and authorities framework (HITS), where each node is associated with an authority and a hub score [6]. Each directed edge is deemed as an endorsement of authority in one direction and of connectivity ("hubness") in the opposite direction. The graph structure is then used to propagate these scores till an equilibrium is attained.

Recently, Yin et al. [15], proposed the *TruthFinder* algorithm specifically focused on opinion aggregation for binary opinions following an approach similar to HITS. However, unlike HITS, the predicate truth scores are computed by probabilistic aggregation of agent opinions assuming independence as in [2]. This paper also proposes simple heuristics for handling dependencies between predicates and was shown to be more effective than a simple voting strategy.

9. CONCLUSION

In this paper, we studied the problem of merging historical information from multiple sources. Unlike prior work, which assumes explicit mapping between source values and the real values they observe, we model the mapping as a hidden unknown variable. We then perform inference to compute an estimate of the history of true updates, together with their mapping to the source values. We presented two approximation algorithms for this inference task, and evaluate their performance against several baseline methods that either ignore history, or use it in a naive way. These experiments show that our techniques are able to approximate both the unknown history and the final value significantly more accurately than baseline techniques.

10. REFERENCES

[1] X. L. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. *Proc. VLDB Endow.*, 2:562–573, August 2009.

[2] D. Freitag. Multistrategy learning for information extraction. In *ICML*, pages 161–169, 1998.

[3] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *WSDM*, pages 131–140. ACM, 2010.

[4] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, Nov. 1984.

[5] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, Apr. 1993.

[6] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[7] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *VLDB 2011*.

[8] D. Liu, P. Ning, A. Liu, C. Wang, and W. Du. Attack-resistant location estimation in wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.

[9] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

[10] D. Sankoff. Matching sequences under deletion/insertion constraints. *PNAS*, 69(1):4–6, 1972.

[11] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):pp. 787–793, 1974.

[12] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Info. Theory*, 13(2):260–269, Apr. 1967.

[13] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *JACM*, 21:168–173, 1974.

[14] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[15] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *KDD*, pages 1048–1052, 2007.

[16] S. Z. Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010.

[17] S.-Z. Yu and H. Kobayashi. A hidden semi-markov model with missing data and multiple observation sequences for mobility tracking. *Signal Processing*, 83(2):235–250, 2003.