A Time Machine for Information: Looking Back to Look Forward

Xin Luna Dong
Google Inc.
lunadong@google.com

Anastasios Kementsietsidis
Google Inc.
akement@google.com

Wang-Chiew Tan*
UC Santa Cruz
tan@cs.ucsc.edu

ABSTRACT

Historical data (also called *long data*) holds the key to understanding when facts are true. It is through long data that one can understand the trends that have developed in the past, form the audit trails needed for justification, and make predictions about the future. For searching, there is also increasing interest to develop search capabilities over long data.

In this article, we first motivate the need to develop a *time machine* for information that will help people "look back" so as to "look forward". We will overview key ideas on three components (extraction, linking, and cleaning) that we believe are central to the development of any time machine for information. Finally, we conclude with our thoughts on what we believe are some interesting open research problems. This article is based on the material presented in a tutorial at VLDB 2015.

1. INTRODUCTION

> *"The longer you can look back, the farther you can look forward."*
> — Winston Churchill

There is general consensus that while *big data* are important, *long data* (i.e., historical data or temporal data) are even more important [29, 48], as it provides the potential to understand when facts are true and forms the basis for audit trails needed by organizations and individuals. It is for this reason that companies use temporal databases to provide support for rollbacks and auditing. For Web pages, information from the Wayback machine of the Internet Archive [30], which periodically keeps a snapshot of most webpages on the Web, has been used as legal evidence.

With the abundant availability of information one can mine from the Web today, there is increasing interest to develop capabilities to search long data on the Web, to develop a complete understanding of the history of an entity (i.e., a person, a company, a music genre, a country, etc.), and to depict trends over time, based on Web data. An example is the search for answers to the query "*Google's CEO before Sundar Pichai*". Current search results on Google returns articles about Sundar Pichai and it is only after one reads through some articles about the history of Google that one can infer that the CEO of Google before Sundar Pichai is Larry Page.

Another compelling example to motivate the need to understand *when* a fact is true comes from reports that are filed with the U.S. Securities and Exchange Commission (SEC) [20] at different times. Companies are required, by federal regulations, to file reports periodically to SEC to disclose the stock holdings of its executives. There are now millions of electronic filings in EDGAR and the number of such filings is increasing over time. Given the millions of SEC reports, how can one find out the stock holdings of an executive during a certain period of time? Were *Ann* and *Bob* affiliated with the same *company X* during a certain time period? Or perhaps more interestingly, did *Ann* purchase a significant number of shares of *Company Y* before it was announced that *Company X* bought *Company Y*? Techniques for integrating and aggregating data over time have also been explored in a number of projects [2, 5, 25, 33, 38, 40, 41, 45].

Finally, historical data provides an understanding of what is important or trending over time. For example, the understanding that most people believed earth was flat versus spherical and the trending topics of discussions allow tremendous opportunities for further knowledge [19, 46, 47].

The task of aggregating long data into a meaningful whole remains a largely difficult and manual task despite more than a couple of decades of research in the areas of temporal databases and data integration. The difficulty to create a comprehensive understanding of entities over time largely stems from the lack of (explicit) temporal data, and tools for interpreting such data even if they were available. Ideally, we would like to develop a time machine for information, where one can

*Tan is partially supported by NSF Grant IIS-1450560 and IIS 1524382.

easily and incrementally ingest temporal data to develop an ever increasing understanding of entities over time, search and query facts for a particular time period, understand trending patterns over time, and perform analytics that would allow one to, for example, understand the prevalent "knowledge" in the previous decade. In this article, we describe the techniques critical in building such a *time machine for information*, and discuss how far (or close) we are in achieving this goal.

The development of such a time machine would necessarily involve many of the challenges that occur in data integration [13, 17] and knowledge curation (see, for example, [4, 15, 27]), which are notoriously difficult tasks. The data integration process often includes the fundamental steps of extracting information about different types of (heterogeneous) entities, transforming and cleaning the information, and curating facts regarding different aspects or properties of those entities consistently together. An additional challenge today is to perform these tasks at scale; that is, we need to collect information from a large number of data sources, where each source may contain lots of data, and the schemas of the sources may be diverse in their structure and quality or may even be unavailable. The ability to inter-operate amongst heterogeneous data sources with varying quality is thus a key ingredient to the successful development of this time machine.

Another key ingredient to the successful development of such time machine is to make every step of the data integration process *time-aware*. In other words, we need the capability to understand the valid time period for each fact. To achieve this goal, one would inevitably require text extraction rules or techniques to extract structured temporal data from unstructured and semi-structured data sources. Furthermore, techniques need to be developed to map and transform temporal data into a desired format before temporal entity resolution can be applied. And finally, information about the extracted entities is temporally integrated and conflicting information is resolved to arrive at an integrated archive. This process may repeat as new datasets are discovered or when new versions of the same datasets are available to further enrich the information time machine.

Outline In this article, we first look back at past work related to (bi-)temporal databases. We discuss why new techniques beyond (bi-)temporal databases need to be developed to integrate and manage long data in general (Section 2). We will then present some existing work on three components (extraction, linking, and cleaning) that we believe are central to the development of any time machine for information (Sections 3-5) before we conclude with our thoughts by looking forward on some of the interesting open research problems (Section 6).

While one goal of this article is to disseminate the above described material, a parallel goal is to motivate the reader to pursue research in the direction of managing and integrating temporal data. Ultimately, we hope that there will be more research along these directions to bring us closer to realizing the goal of building a time machine for information that will record and preserve history accurately, and to help people "look back" and, so as to, "look forward".

We note that this article is based on material presented in a tutorial at VLDB 2015 [18].

2. TEMPORAL DATABASES

The need for enterprises to track and query long data, roll back to previous states of the database to provide audit trials has led to the provision of temporal data management features in relational databases. Research in temporal databases has a long history (see, for example, [9, 10, 42, 43]) but the ability to create and manage temporal tables only found its way to the SQL standard in 2011, as part of the SQL:2011 standard. Since then, major database vendors such as IBM DB2 10 [12], Oracle's Total Recall, and Teradata [1] have also begun to support the temporal features.

There are two primary notions of time in temporal databases, namely, *valid time* (the time period during which a tuple is true) and *transaction time* (where the beginning of this time period represents when the tuple and its valid time period was first recorded in the database, and the duration of the transaction represents the time period during which the recorded tuple and its valid time was true). These are also known as *application time* and, respectively, *system time* in SQL:2011. Naturally, transaction time (or system time) can only increase since the next tuple that is recorded can only occur at a later time than the previously recorded tuple. This is in contrast with valid time, which may refer to the past or future regardless of when it was entered into the database.

For example, Anna was at restaurant Fleur De Lys at 12pm on March 28. Suppose this information was extracted and entered into the database on 10am March 29, the transaction time begins on 10am March 29, even though its valid time begins earlier on 12pm March 28. Later, we may also discover that Anna was at San Francisco's Museum of Modern Art (MOMA) on 11am on March 28. Suppose this information was only determined and entered into the database on 12pm March 30, the valid time begins on March 28, 11am, while the transaction time begins on March 30, 12noon. Each subsequent fact that is entered into the database has a later transaction time than the previous fact even though the valid time may occur at different (out-of-order) times. The first two records in Figure 1 illustrate the example we just discussed. The third record shows that Anna

Name	Location	When	Known since
Anna	Fleur De Lys	Mar 28, 12pm	Mar 29, 10am
Anna	SF MOMA	Mar 28, 11am	Mar 30, 12pm
Anna	SF MOMA	Mar 28, 1:15pm	Mar 30, 2pm

Figure 1: Records of locations of Anna.

was at MOMA on 1:15pm March 28 and this fact was entered into the database on 2pm March 30.

SQL statements can be issued to record the above events as-is with the corresponding application and system times into a bitemporal database. However, without additional application logic, these records are insufficient for the purposes of consistently integrating this information to arrive at the understanding that Anna was at SF MOMA from 11am to 12noon, at Fleur De Lys from 12noon to 1:15pm, and then back at SF MOMA from 1:15pm onwards. Even worse, in general, the time at which the records are entered into the database may not even correspond to the time when the information was known, since different pieces of information may be derived from different data sources at different times. The above inference of where and when is Anna at a location is done based on the preference that information with a later known time is preferred to information from an earlier known time, and that Anna can only be at one location at any point in time. With other types of preferences (e.g., information from one source is preferred over the other), other conclusions may be derived.

The above discussions point out the need for techniques to consistently integrate long data that goes beyond what bi-temporal databases and existing data integration support. In particular, support for user-defined preferences to decide how conflicting long information should be resolved is needed. In fact, techniques such as [2, 5] have taken a step in this direction to consistently aggregate possibly conflicting long information from different sources. In the next three sections, we will present some existing work on three major components (extraction, linking, and fusion) that are central to the development of any end-to-end data integration and management system for long data.

3. EXTRACTION

3.1 Techniques for conventional data

To build any type of time machine, or big data repository, we require data. While large repositories of proprietary data are often hidden behind corporate firewalls, or are available with restrictive licensing, the web is a rich source of information and is publicly available for use by anyone. In addition, web data provides a wide coverage on almost any topic of interest, and is updated constantly. The challenge is thus to *extract* useful data from this fairly unstructured world. In what follows, we discuss some of the challenges in extracting data directly from texts, short texts, and fairly structured web tables.

Information extraction, which refers to the task of understanding text and extracting structured data from texts, has been an active area of research for quite some time [31]. Its objective is to take a sentence such as *"Anna was at restaurant Fleur De Lys on March 28, 12 noon"* and extract from it structured entries that might look like a set of triples of the form

(Anna, location, Fleur De Lys),
(Anna, on, March 28).

Notice that not all information might be succesfully extracted in the process of extraction. For example, we might not extract the fact that Anna was at the restaurant at 12 noon, or that Fleur De Lys is actually a restaurant (and not a city). So, extractions are often incomplete by nature. The extraction task is typically further exacerbated by the need to consider (a) the richness of the language and that the same fact can be expressed in a number of different ways; and (b) processing cannot be limited to a single sentence at a time since understanding the semantics of a sentence often requires understanding its context (e.g., other sentences in the same paragraph). As an example, assume that the sentence above is followed by the statement *"She ordered pasta"*. To extract the simple fact that *(Anna, ate, pasta)* we need to dereference *"She"* and understand that it refers to Anna. Successful approaches in information extraction often rely on machine learning [49] or on a combination of machine learning and natural language understanding techniques [21].

More recently, the popularity of services such as Twitter gave momentum to a new class of short text sources that is rich in information. A distinguishing characteristic of such short text sources is that although there is a huge volume of data, the information content is very small in each short text (140 characters in the case of Twitter). On the surface, the analysis of a 140-character tweet seems like a much easier task when compared to the analysis of a document with many sentences. However, the simplicity of short texts is deceiving and in fact, the lack of context often makes the analysis of short texts significantly harder.

Consider for example two simple short texts *"I like pink songs"* and *"I like pink shoes"*. Even though a human might immediately understand that the former short text is likely to refer to the songs of the artist Pink and the latter short text refers to the fact that the author of the text likes shoes that are pink in color, the lack of context makes the automatic analysis of these short texts very challenging. In addition to the lack of context, part of the challenge also has to do with the fact that short texts often lack any syntax (due to the limitations in length). As a result, traditional natural lan-

guage based techniques often fail to properly extract information, and new techniques that are customized for the analysis of short text are necessary [28].

In a more ideal extraction scenario, one may be able to find the data of interest in a fairly structured setting. For example, the data of interest may exist as a web table [6, 36]. Web tables are typically small relational tables crawled from HTML tables on the Web. Consider Anna from our example in the previous section. If Anna is a celebrity whose sightings are tracked at some fan web page, one can conceivably find a table like the one shown in Figure 1 that offers a row for each fan encounter with Anna. While a table such as this one is a great starting point for data extraction, there are still significant challenges to be addressed. For one thing, it is non-trivial to identify the data types of each column since such tables often come either without or with ambiguous metadata (e.g., column headings). Sometimes, such tables do not even come with column headings. More importantly, while structured data are designed with explicit connections between the different parts of the schema (e.g., foreign keys), figuring out how different web tables relate to each other often requires techniques that go beyond the extraction of column types and understanding the text surrounding the table.

3.2 Techniques for temporal data

There is a large body of work on *temporal information extraction* [37] that is pertaining to the construction of a time-machine, since beyond extracting facts about the world we would also like to know the time that these facts were valid. Conceptually, there are three main challenges in temporal information extraction, namely (a) the identification of temporal references in the input, (b) the mapping of a temporal reference to a time point (or interval), and (c) the assignment of time point (or interval) to a fact. In what follows, we provide a high-level review of the first two time-related challenges.

Part of the challenge in identifying temporal references is that time is expressed in a variety of ways in text. So, while one can find a clear temporal reference in the sentence *"Anna was at restaurant Fleur De Lys on March 28, 12 noon"*, the following two sentences include temporal references that are less obvious: *"Last week, Anna was at restaurant Fleur De Lys."*, and *"Anna was at restaurant Fleur De Lys."*. Notice that both the first and second sentences have an *explicit* time reference (in the form of "March 28, 12 noon", and "Last week"), while the last sentence has an implicit time reference since the past tense of the verb implies that the event happened some time in the past (see [24] of a more complete list of explicit and implicit examples).

To be useful in practice, an identified time reference must be assigned to a time point (or interval). In the examples above, it seems fairly straighforward to assign the "March 28, 12 noon" to a point in time, assuming we know the year that this date is referring to. Similarly, for the assignment of the reference "Last week" we need some indication as to what is the week the text (paragraph) was written, or in the worst case ground the time reference to a time point based on a signal like the document creation time (which resembles the transaction time in temporal databases). Similarly, for the last sentence we need to identify the time the text was written (or is referring to) so that we assign a time interval to the event that is upper-bounded by the identified time. Not surprisingly, mapping time references to a time dimension is probably the most challenging part of temporal information extraction. Techniques in this space might rely on rules [44], or a combination of machine-learning and syntactic analysis [37].

4. RECORD LINKAGE

After structured data is obtained, another major step is to identify records that refer to the same real-world entities. The *record linkage* problem refers to the following problem: *given a set of records, each describing an entity by its attribute values, partition the records so that each partition contains records that refer to a distinct real-world entity.*

4.1 Techniques for conventional data

Solutions to the record linkage problem typically include the following three steps [17].

1. *Blocking* refers to the process where blocking functions are applied to partition input records into multiple smaller blocks such that records in different blocks are very unlikely to refer to the same real-world entity. Blocking helps reduce the number of pairwise comparison that would otherwise be needed in subsequent steps.
2. *Pairwise matching* refers to the process that compares every pair of records in a block to decide if they indeed refer to the same entity.
3. *Clustering* refers to the process that examines local pairwise matching decisions to arrive at a globally consistent decision of partitioning the records from the same block such that each partition refers to a distinct entity.

Among the three steps, the blocking step is performed to achieve *scalability* (i.e., to handle massive data), while the pairwise matching and clustering steps are used to ensure the *semantics* of record linkage. In other words, by blocking, one avoids unnecessary comparisons between two records in the pairwise matching and clustering steps, which determine whether or not two records refer to the same real-world entity.

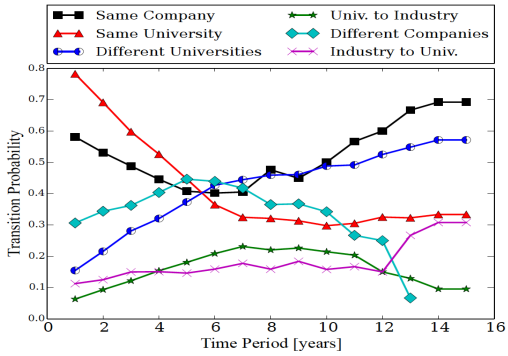

Figure 2: Transition rate on affiliation learned from DBLP [32].

4.2 Techniques for temporal data

Traditional linkage techniques typically reward high similarity between attribute values, and, likewise, penalize low similarity between attribute values. Such measures are not always appropriate for temporal record linkage. This is because, as time elapses, attribute values of an entity tend to evolve; for example, a person may change her name after marriage, change affiliations as a result of changing jobs, and may also change her phone number, address, and so on. Hence, blindly penalizing low value similarity can lead to a lot of false negatives in general; that is, records that refer to the same entity at different times are not matched because of value evolution. At the same time, as time elapses, different entities are increasingly likely to share the same attribute values, such as the shared name *Adam Smith* between a British philosopher in the 18th century and a current politician in the US. Hence, blindly rewarding high value similarity can lead to a lot of false positives in general; that is, records that refer to different entities at different times are wrongly matched together because they happen to share the same attribute values.

The discussions above point out that temporal records present new challenges to the *temporal record linkage problem* and new techniques are needed to perform temporal record linkage.

In the next two sections, we describe two key solutions for the temporal record linkage problem; one exploits pairwise matching techniques, and the other exploits clustering techniques. These techniques make crucial use of the following observations about temporal records in their solutions.

- First, entities typically evolve smoothly, and typically only a few attribute values of an entity change at any given time.
- Second, the values of an entity do not change erratically and, in particularly, they rarely change back and forth.
- Third, if the data set is fairly complete, records that refer to the same real-world entity typically (though not necessarily) observe continuity, or similarity in time gaps between adjacent records.

4.2.1 Time decay

A key insight by Li *et al.* is the notion of *time decay* [34] Time decay is often used in data analytics to reduce the impact of older records on the analysis and can be used effectively to capture the impact of time elapse on attribute-value evolution. Two types of decay, *disagreement decay* and *agreement decay*, were proposed in [34].

- Consider an attribute A and a time gap ΔT. The *disagreement decay* of A over ΔT is the probability that an entity changes its A-value within time ΔT. The dsiagreement decay is denoted by $d^{\neq}(A, \Delta T)$.
- Consider an attribute A and a time gap ΔT. The *agreement decay* of A over ΔT is the probability that the A-value is the same for two distinct entities within time ΔT. The notation we use for agreement decay is $d^{=}(A, \Delta T)$.

It is easy to see that both disagreement and agreement decays are values in $[0, 1]$, and typically monotonically non-decreasing as a function of their second argument ΔT. Intuitively, the disagreement decay is used to reduce the penalty for value disagreement, while the agreement decay is used to reduce the reward for value agreement, over a long time period. More formally, this is done by defining the pairwise similarity between two records R_1 and R_2 as

$$Sim(R_1, R_2) = \frac{\Sigma_{A \in \bar{A}} dw_A(s(R_1.A, R_2.A), \Delta T) * s(R_1.A, R_2.A)}{\Sigma_{A \in \bar{A}} dw_A(s(R_1.A, R_2.A), \Delta T)}$$

where $dw_A(s(), \Delta T)$ denotes the decayed weight of attribute A with value similarity $s()$ and time gap $\Delta T = |R_1.T - R_2.T|$. When the value similarity $s()$ is low, $dw_A(s(), \Delta T)$ is set to $w_A * (1 - d^{\neq}(A, \Delta T))$; when the value similarity $s()$ is high, $dw_A(s(), \Delta T)$ is set to $w_A * (1 - d^{=}(A, \Delta T))$, where w_A is the non-decayed weight of attribute A.

Li *et al.* [34] also describe ways to learn the disagreement and agreement decays empirically from a labeled data set. As an example, Figure 3 shows disagreement decay and agreement decay on attribute address learned from a data set that contains 1871 European Patent from 359 inventors in years of 1978-2003. We observe that while inventors over different periods of time seldom shared the same address over time, many of them have changed their address over time.

There are two extensions for time decays. First, Chiang *et al.* [7] define *recurrence rate*, which considers the

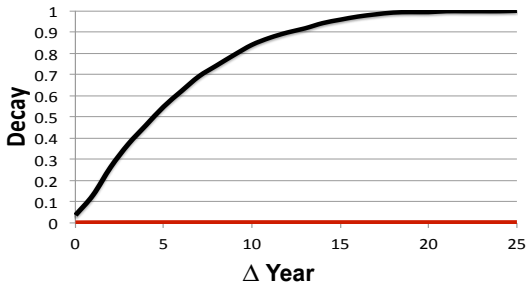

Figure 3: Agreement (flat line) and disagreement decay (curved line) for inventor address learned from a European patent data set [34].

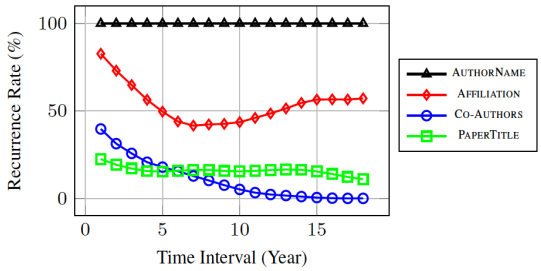

Figure 4: Recurrence rate on four attributes learned from DBLP [7].

probability of the same value re-occurring for an entity attribute. The intuition is that after a value changes, it may change back at a future time point; for example, a person may return to the same affiliation, may keep co-authoring with the same authors after a period of time, and so on. Figure 4 shows the recurrence rate on four attributes learnt from DBLP.

Second, Li *et al.* [32] consider a finer-grained disagreement decay, called the *transition probability*. Intuitively, the transition probability captures the likelihood of a particular (type of) value transitioning to different other (types of) values. These probabilities vary widely across different pairs of values. For example, it is far more likely for someone with the title of an *"Associate Professor"* to transit to the value *"Full Professor"* than *"Accountant"*. Figure 2 shows the transition rate between different types of values for affiliation learned from DBLP.

4.2.2 *Two-stage temporal clustering*

In addition to time decay, which is used in pairwise comparison, Li *et al.* [34] also propose a two-stage temporal clustering strategy, where the first stage requires high value consistency and obtains high-precision results, and then the second stage adjusts results from the first stage to improve recall. The key intuition is that, unlike traditional clustering techniques that are time-agnostic, considering the time order of records can often provide important clues for correct record linkage. The two stages proceed as follows.

1. The first stage considers the records in temporal order. It compares a record with each previously created cluster and computes a probability of merging. After processing all the records, it makes the clustering decision to maximize the overall probability of record-cluster matching.

2. The second stage augments the clustering results by also comparing a record with clusters that are created later. The comparisons take into account record continuity and clusters are adjusted based on these comparisons.

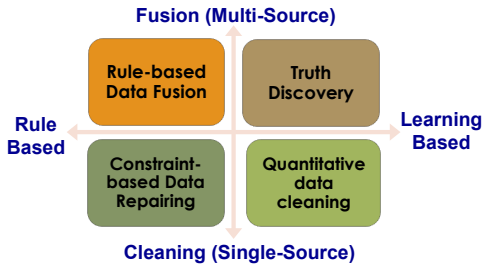

Figure 5: Data cleaning techniques classified along two dimensions.

There are also two extensions for the two-stage clustering method. The solution proposed by Chiang *et al.* [8] makes a clean cut on the two stages: the first stage, called *static stage*, requires high consistency on attribute values and does not consider time decay; the second stage, called *dynamic stage*, considers time decay and further merges clusters generated in the static stage.

In the solution proposed by Li *et al.* [32], the first stage may place a record into multiple different clusters, if it believes that the provider of the record is likely to provide stale data. The second stage then merges clusters from the first stage once it decides that the probability of transition is high, and ignores the clusters consisting of only stale records.

5. CLEANING AND FUSION

5.1 Techniques for conventional data

Data cleaning refers to the following problem: *given a snapshot of data, decide which piece of data is incorrect w.r.t. the ground truth and find the fix.*

As shown in Figure 5, data cleaning techniques can be classified along two dimensions. The first dimension concerns whether the data cleaning process considers only a single source or data over multiple sources. The term *data cleaning* generally refers to the task of

cleaning a single data source whereas the term *data fusion* is generally used to refer to the task of merging and cleaning data from multiple sources [16]. The second dimension concerns whether the underlying technology used for data cleaning is *rule-based* or *learning-based*. The combinations along these two dimensions give rise to four categories of methods.

- **Rule-based data cleaning–*Constraint-based data repairing*:** This type of data cleaning relies on the use of constraints, such as functional dependencies and, more recently, CFDs (Conditional Functional Dependencies), to specify the relationship between data values [22]. When such relationships are violated, the data are typically cleaned (or repaired) by finding the smallest set of changes to the data values such that the constraints are no longer violated.
- **Learning-based data cleaning–*Quantitative data cleaning*:** Such methods (e.g., [11, 26]) apply statistical techniques to detect outliers of the data. It suggests cleaning strategies such that the cleaned data have close distribution to an ideal data set; however, a quantitative repair is not guaranteed to be logically consistent.
- **Rule-based data fusion–*Declarative data fusion*:** Such methods (e.g., [3]) specify rules such as computing the average value from a list of values, finding the most popular value or the latest update to resolve conflicting values from multiple sources.
- **Learning-based data fusion–*Truth discovery*:** Such methods (e.g., [35]) find the truths that are consistent with the real world by applying machine learning models that consider trustworthiness of sources and copying relationships between the sources.

5.2 Techniques for temporal data

Temporal data add a new dimension, the time dimension, to the problem. The problem thus becomes the following: *given a set of temporal data, decide which piece of data is incorrect for the claimed time point or period of time and find the fix.*

Temporal data raise two new challenges for data cleaning and fusion. First, as mentioned before, the true values can evolve over time; so it is critical to distinguish between false data and out-of-date data, and to decide the time period that a value is true. Second, there are many causes for low-quality data: in addition to providing false and imprecise data, a previously good data source may stop updating (a part of) data or updating data infrequently and thus provide stale data.

To overcome these challenges, observe that (1) the update history from data sources can give hints for changes of the real world; for example, a source may remove information about a restaurant because the restaurant is closed; and (2) certain attribute values observe partial order; for example, for marry_status, value *single* should occur before *married* in a timeline, which in turn should occur before *divorced* in time.

We next describe how each category of data-cleaning techniques are extended for temporal data. We omit the category of learning-based data cleaning, as we are not aware of any quantitative cleaning strategies for temporal data.

5.2.1 Rule-based data cleaning: data currency

Fan *et al.* studied *data currency* [23] for cleaning temporal data. It considers data *without* timestamps as input, and aims at finding the *up-to-date* values.

The key idea of the solution is to extend conventional database constraints with *currency constraints*, which express currency relationships from the semantics of the data. An example currency constraint is stated below:

$$\forall s, t\text{: } s[\mathsf{status}] = \text{“married”} \wedge t[\mathsf{status}] = \text{“single”} \rightarrow t \prec_{status} s$$

This constraint states that given two tuples s and t about the same entity (*i.e.*, having the same EID), if s provides value *married* for marriage status while t provides value *single*, then s must have a more recent value than t on status.

In addition to currency constraints, their work also considers a simple class of CFDs, called *constant CFDs*. A constant CFD is a constraint that asserts an equality to a constant value. A *specification* is defined as a triple $S_e = (I_t, \Sigma, \Gamma)$, where I_t specifies existing knowledge on partial order of tuples regarding their timestamps, Σ denotes currency constraints, and Γ denotes constant CFDs.

Fan *et al.* [23] investigated four problems around data currency.

- *Satisfiability:* Decide if a specification is *valid*; in other words, whether the partial orders, the currency constraints, and the constant CFDs have conflicts. It is shown that this problem is NP-complete in general.
- *Implication:* Decide if any other currency orders can be implied by a given specification. It is shown that the problem is coNP-complete in general.
- *True value deduction:* Decide whether true values of an entity can be derived from a specification. Like implication, this problem is coNP-complete in general.
- *Coverage analysis:* Decide the minimum coverage of a given specification (*i.e.*, additional partial orders or currency constraints to be provided) to make true

values derivable. This problem is Σ_2^p-complete in general.

It is also proved that the above complexity results remain even if only currency constraints or only constant CFDs are present.

Although all these problems are intractable in general, Fan *et al.* [23] developed practical algorithms that integrate inferences of data consistency and currency in a single framework, deriving true values and finding a minimum set of attributes that require users' input to find their true values.

5.2.2 Rule-based data fusion: preference-aware union

Alexe *et al.* [2] propose a *preference-aware union* operator for fusing temporal data from multiple data sources. The intended use of the preference-aware union operator is typically at the final step of an entity integration workflow, where the outcome of this step is the set of integrated entity profiles derived by aggregating information from multiple sources.

Intuitively, the operator takes as input temporal data (from multiple sources) and resolves temporal conflicts that may occur in the data according to a given set of constraints and user-specified *preference rules*. For example, a constraint may assert that a person can have at most one affiliation at any time point, and a conflict arises if the input data corroborates there is more than one affiliation at some time point. A preference rule may state that one should prefer the values from a later timestamp or it may specify that one should prefer values from one source over another. With the given constraints and preference rules, the operator will then resolve a conflict, to the extent possible, to conclude what should be the affiliation at the time points when conflicts occur according to the specified preferences.

In the event that not all conflicts can be resolved through preferences, one can then enumerate each possible consistent interpretation of the result returned by the operator at a given time point through a polynomial-delay algorithm. A key property of the solution is that the operator produces the same integrated outcome, modulo representation of time, regardless of the order in which data sources are integrated.

5.2.3 Learning-based data fusion: temporal data fusion

Temporal data fusion takes data *with* timestamps as input, and tries to decide not only the currently correct values, but also *the correct values in the history and their valid time period*.

Specifically, consider a set $\mathcal{D}$ of *data items*, each describing the attribute of an entity, such as affiliation of a person. A data item is associated with a value at each particular time t and can be associated with different values at different times. The *life span* of a data item D is defined as a sequence of *transitions*, each associated with a value change regarding D at a particular time point. On the other hand, consider a set $\mathcal{S}$ of data sources, each providing values for data items in $\mathcal{D}$ and can change the data over time. Data provided by the sources are observed at different times; by comparing an observation with its previous observation, a series of *updates* can be inferred. Given $\mathcal{D}$ and $\mathcal{S}$, the *temporal data fusion* problem computes the life span of each data item in $\mathcal{D}$.

The solutions to this problem typically contains two major parts. The first component computes quality metrics of the sources for temporal data. The second component conducts inferences to decide the lifespan of each data item.

Source quality: For conventional static data, source quality is typically measured by its accuracy [35]. The metrics are far more complex for the dynamic case [14]. Ideally, a high-quality source should provide a new value for a data item *if and only if*, and *right after*, the value becomes true. These three conditions are captured by three measures: (a) the *coverage* of a source measures the percentage of all transitions of different data items that it captures (by updating to the correct value); (b) the *exactness* is the complement of the percentage of transitions that the source mis-captures (by providing a wrong value); and (c) the *freshness* is a function of time ΔT, measuring among the captured transitions, the percentage that are captured within time ΔT. These three measures are orthogonal and collectively referred to as the *CEF-measure*.

Inference: There are two inference approaches to decide the lifespan of a data item. The first inference approach uses Bayesian analysis [14]. Consider a data item $D \in \mathcal{D}$. To discover its life span, both the time and the value of each transition need to be decided. The Bayesian analysis is based on the CEF-measures of D's providers.

1. First, decide the value of D at a beginning time point.
2. Then, find for D's next transition the most likely time point and the most likely value, and repeat this process until it is decided that there is no more transition.

The second approach applies the Markov model [39]. As in the aforementioned Bayesian analysis, the Markov model considers the delay between real-world changes and source observation, and the delay between source observation and source update. However, there are two differences. First, the Markov model additionally encodes domain knowledge such as the partial order between values for a particular attribute. Second, it as-

sumes that the data contains only minor errors such as mis-spellings, and ignores possible erroneous data.

Finally, note that there can be copying relationships between the sources and the copying relationships may even change over time. Dong *et al.* [14] describe how to apply a Hidden Markov Model model to find such evolving relationships.

6. LOOKING FORWARD

There are many problems related to the extraction and integration of temporal data that are yet to be solved.

First, the Web is a rich source of information. It is possible to extract information from the Web, connect the dots, and recover the history of an entity. For example, by extracting information about the talks, tweets, or even resumes and home pages, one can build a profile of the affiliation (and even location) of a person over time. How can one automatically find the relevant web pages and tweets? How can one automatically ingest these sources of information and combine the temporal information for different attributes?

Second, as it is already challenging to build history for a single entity, it is even more challenging to build history for collective entities, such as the history of Rome, the history of World War II, and the history of rock music. The key here is to identify relevant single entities such as people and events, rank them according to their importance regarding the collective entity, and build the history taking into account the time dimension.

Third, as the facts for an entity are evolving over time, people's perspective on the entity can also evolve. For example, people used to believe that the earth is flat, and it is only in the recent two centuries that people started to believe that human evolved from apes. Distinguishing the fact changes and the perspective changes proposes new challenges in managing temporal data.

Finally, while there is already substantial body of work on temporal text extraction and temporal entity resolution, earlier foundational work on mapping and data exchange, conflict resolution, and query answering in inconsistent and incomplete databases have been largely time-agnostic. It will be desirable to develop a principled framework for managing inconsistent temporal data and managing incompleteness (that may change with time) in temporal data.

7. REFERENCES

[1] M. Al-Kateb, A. Ghazal, A. Crolotte, R. Bhashyam, J. Chimanchode, and S. P. Pakala. Temporal query processing in teradata. In *EDBT*, pages 573–578, 2013.

[2] B. Alexe, M. Roth, and W. Tan. Preference-aware integration of temporal data. *PVLDB*, 8(4):365–376, 2014.

[3] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2009.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[5] D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. Stanoi, S. Vaithyanathan, and S. R. Das. Extracting, linking and integrating data from public sources: A financial case study. *IEEE Data Eng. Bulletin.*, 34(3):60–67, 2011.

[6] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *PVLDB*, 1(1):538–549, Aug. 2008.

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

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

[9] J. Chomicki. Temporal query languages: A survey. In *ICTL*, pages 506–534, 1994.

[10] J. Chomicki and D. Toman. Temporal databases. In *Foundations of Artificial Intelligence*, pages 429–467. Elsevier, 2005.

[11] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, Inc., New York, 2003.

[12] A matter of time: Temporal data management in db2 10, 2012. `http://www.ibm.com/developerworks/data/library/techarticle/dm-1204db2temporaldata/`.

[13] A. Doan, A. Halevy, and Z. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[14] X. L. Dong, L. Berti-Equille, and D. Srivastava. Truth discovery and copying detection in a dynamic world. *PVLDB*, 2(1):562–573, 2009.

[15] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 2014.

[16] X. L. Dong and F. Naumann. Data fusion - resolving data conflicts for integration. *PVLDB*, 2(2):1654–1655, 2009.

[17] X. L. Dong and D. Srivastava. *Big Data Integration*. Morgan & Claypool, 2015.

[18] X. L. Dong and W. Tan. A time machine for information: Looking back to look forward. *PVLDB*, 8(12):2044–2055, 2015.

[19] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. *ACM Transactions on the Web (TWEB)*, 1(2):7, 2007.
[20] The EDGAR Public Dissemination Service. `http://www.sec.gov/edgar.shtml`.
[21] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction: The second generation. In *IJCAI*, pages 3–10, 2011.
[22] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.
[23] W. Fan, F. Geerts, N. Tang, and W. Yu. Conflict resolution with data currency and consistency. *ACM Journal of Data and Information Quality*, 5, 2014.
[24] E. Filatova and E. Hovy. Assigning time-stamps to event-clauses. In *Workshop on Temporal and spatial inf. proc. -Volume 13*, page 13, 2001.
[25] D. Graus, M.-H. Peetz, D. Odijk, O. de Rooij, and M. de Rijke. yourhistory–semantic linking for a personalized timeline of historic events. *Workshop: LinkedUp Challenge at Open Knowledge Conference (OKCon) 2013*, 2013.
[26] J. Hellerstein. Quantitative data cleaning for large databases. Technical report, UC Berkeley, 2008.
[27] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In *WWW*, pages 229–232, 2011.
[28] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou. Short text understanding through lexical-semantic analysis. In *ICDE*, pages 495–506, 2015.
[29] Big Data, Meet Long Data. `http://www.informationweek.com/big-data/big-data-analytics/big-data-meet-long-data/d/d-id/1109325?`, Apr 1, 2013.
[30] Internet Archive Wayback Machine. http://waybackmachine.org, Aug. 26, 2011.
[31] J.-T. Kim and D. I. Moldovan. Acquisition of semantic patterns for information extraction from corpora. In *Artificial Intelligence for Appl.*, pages 171–176, 1993.
[32] F. Li, M. L. Lee, W. Hsu, and W.-C. Tan. Linking temporal records for profiling entities. In *SIGMOD*, 2015.
[33] J. Li and C. Cardie. Timeline generation: tracking individuals on twitter. In *WWW*, pages 643–652, 2014.
[34] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *PVLDB*, 4(11):956–967, 2011.
[35] X. Li, X. L. Dong, K. Lyons, W. Meng, , and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 2013.
[36] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1-2):1338–1347, 2010.
[37] X. Ling and D. S. Weld. Temporal information extraction. In *AAAI*, volume 10, pages 1385–1390, 2010.
[38] A. Mazeika, T. Tylenda, and G. Weikum. Entity timelines: Visual analytics and named entity evolution. In *CIKM*, pages 2585–2588, 2011.
[39] A. Pal, V. Rastogi, A. Machanavajjhala, and P. Bohannon. Information integration over time in unreliable and uncertain environment. In *WWW*, pages 789–798, 2012.
[40] R. Qian. Timeline: Understanding Important Events in Peoples Lives. http://blogs.bing.com/search/2014/02/21/timeline-understanding-important-events-in-peoples-lives/, February 2014. Last retrieved on Oct 27, 2014.
[41] M. Roth and W.-C. Tan. Data integration and data exchange: It's really about time. In *CIDR*, 2013.
[42] R. T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer, 1995.
[43] R. T. Snodgrass and I. Ahn. Temporal databases. *IEEE Computer*, 19(9):35–42, 1986.
[44] J. Strötgen and M. Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Intl. Workshop on Semantic Evaluation*, pages 321–324, 2010.
[45] T. A. Tuan, S. Elbassuoni, N. Preda, and G. Weikum. Cate: context-aware timeline for entity illustration. In *WWW*, pages 269–272, 2011.
[46] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In *EDBT*, pages 697–700, 2010.
[47] G. Weikum, N. Ntarmos, M. Spaniol, P. Triantafillou, A. A. Benczúr, S. Kirkpatrick, P. Rigaux, and M. Williamson. Longitudinal analytics on web archive data: It's about time! In *CIDR*, pages 199–202, 2011.
[48] Stop hyping big data and start paying attention to 'long data'. `http://www.wired.com/2013/01/forget-big-data-think-long-data/`, Jan 29, 2013.
[49] F. Wu, R. Hoffmann, and D. S. Weld. Information extraction from wikipedia: Moving down the long tail. In *SIGKDD*, pages 731–739, 2008.