Trust Evaluation of Data Provenance

Chenyun Dai, Dan Lin, Elisa Bertino, and Murat Kantarcioglu

Abstract With the increased need of data sharing among multiple organizations like government organizations, financial corporations, medical hospitals and academic institutions, it is critical to ensure data integrity so that effective decisions can be made based on these data. An important component of any solution for assessing data integrity is represented by techniques and tools to evaluate the trustworthiness of data provenance. However, few efforts have been devoted to investigate approaches for assessing how trusted the data are, based in turn on an assessment of the data sources and intermediaries. To bridge this gap, we propose a data provenance trust model. Our trust model takes into account various factors that may affect the trustworthiness and, based on these factors, assigns trust scores to both data and data providers. Such trust scores represent key information based on which data users may decide whether to use the data and for which purposes.

1 Introduction

Today the need of sharing data within and across the organizations is more critical than ever. The availability of comprehensive data makes it possible to extract more accurate and complete knowledge and thus supports more informed decision making. However reliance on data for decision making processes requires data to be of good quality and trusted. We refer to such requirements as *high-assurance data integrity*. Without high-assurance integrity, information extracted from available data cannot be trusted. While there has been some efforts to ensure confidentiality when

Chenyun Dai, Dan Lin, Elisa Bertino
Department of Computer Science, Purdue University
e-mail: {daic, lindan, bertino}@cs.purdue.edu
Murat Kantarcioglu
Department of Computer Science, The University of Texas at Dallas
e-mail: muratk@utdallas.edu

sharing data, the problem of high-assurance data integrity has not been widely investigated. Previous approaches have either addressed the problem of protection from data tampering, through the use of digital signature techniques, or the problem of semantic integrity, that is, making sure that the data is consistent with respect to some semantic assertions. However even though these techniques are important components of any comprehensive solution to high-assurance data integrity, they do not address the question on whether one can actually trust certain data. Those techniques, for example, do not protect against data deception, according to which a malicious party may provide on purpose some false data, or against the fact that a party is unable, for various reasons, to provide good data. Techniques, like those developed in the area of data quality (e.g. [1]), may help; however they often require the availability of good quality data sources against which one can compare the data at hand and correct them.

It is clear that in order to address the problem of high-assurance data integrity we need comprehensive solutions combining several different techniques. In particular, one important issue in determining data integrity is the trustworthiness of data provenance. For example, a malicious source provider may announce that a small company has successfully signed a big contract which is not true in reality. This information is then passed to a stock analysis agent, based on which the agent infers that the stock prize of that company will go up with high probability and send this information to end users. If the data users, based on this information, decide to acquire stocks of such company, they may end up with severe financial losses. In contrast, the data users will have a big chance to avoid such a loss if they know that the source provider is not very trustworthy. Though a lot of research has been carried out for data provenance, they mainly focus on the collection and semantic analysis of provenance information [2, 5, 4]. Little has been done with respect to the trustworthiness of data provenance.

To evaluate the trustworthiness of data provenance, we need to answer questions like "Where did the data come from? How trustworthy is the original data source? Who handled the data? Are the data managers trustworthy?" More specifically, for example, if data X is from source A, how do we determine the trustworthiness of source A. If X arrives at D via B and C, how to tell if X is accurate at D? Also if data X now from D and data Y coming from E are merged by intermediate agent F, then how do we determine the trustworthiness of the resulting data? To address these challenges, we propose a data provenance trust model which estimates the level of trustworthiness of both data and data providers by assigning trust scores to them. Based on the trust scores, users can make their more informed decisions whether or not to use the data.

To build such trust model, we take into account various aspects that may affect the trustworthiness of the data. In particular, these aspects are *data similarity*, *data conflict*, *path similarity* and *data deduction*. Similar data items are considered as supports to one another, while conflicting data items compromise trustworthiness of one another. Besides data similarity and data conflict, the way that the data was collected is also an important factor when determining the trustworthiness of the data. For example, if several independent sources provide the same data, such data is most likely to be true. Data deduction measures the effect of the process (e.g. data mining) on the data. Usually, the trustworthiness of the resulting data depends on the trustworthiness of input data and the on the parties that process the data.

We also observe that a data is likely to be true if it is provided by trustworthy data providers, and a data provider is trustworthy if most data it provides are true. Due to such inter-dependency between data and data providers, we develop an iterative procedure to compute the trust scores. To start the computation, each data provider is first assigned an initial trust score which can be obtained by querying available information about data providers. At each iteration, we compute the trustworthiness of the data based on the combined effects of the aforementioned four aspects, and recompute the trustworthiness of the data provider by using the trust scores of the data it provides. When a stable stage is reached, that is, when the changes of trust scores are negligible, the trust computation process stops.

In summary, this paper makes the following major contributions.

- We formulate the problem of evaluating data provenance in order to determine the trustworthiness of data and data providers.
- We propose a data provenance trust model which defines the trustworthiness of data and data providers. We models four key factors that influence the trust scores.
- We develop algorithms to compute trust scores and our experimental results demonstrate its efficiency.

The rest of the paper is organized as follows. Section 2 introduces some preliminary definitions. Section 3 presents the proposed data provenance trust model including the algorithms for trust score computation. Section 4 reports the experimental results. Section 5 reviews related work. Finally, Section 6 concludes the paper and outlines future research directions.

2 Preliminary Definitions

In this section, we first describe a scenario illustrating the problem of data provenance, and then introduce several definitions used in our trust model.

Data provenance includes information about the process thr-ough which data have been generated and the input and output data of these processes. In this paper, we consider a common scenario (see Figure 1) in which there are multiple parties characterized as data source providers, intermediate agents and data users. Each party is identified by a unique identifier. Data source providers could be sensor nodes or agents that continuously produce large volumes of *data items*. Those data items describe the properties of certain entities or events. Intermediate agents could simply pass the data items obtained from data source providers to data users, or make use of the data items to generate *knowledge items* consumed by data users or other intermediate agents. Data users are the final information consumers who expect to

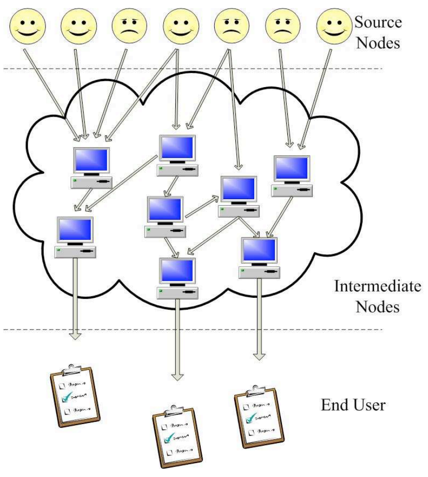


Fig. 1 Example scenario

receive trustworthy data. For representation simplicity, we will refer to a data item or a knowledge item as an item when the context is clear.

In this paper, we model an item (denoted as r) as a row in a relational table and each item has k attributes $A_1, ..., A_k$. For an item r, its value of attribute A_i $(1 \leq i \leq k)$ is denoted as $r(A_i)$. Table 1 gives an example of location reports that will be used throughout the paper. As shown in the table, there are five items, each of which has seven attributes ⟨RID, SSN, Name, Gender, Age, Location, Date⟩. *RID* is the identifier of each item.

RID	SSN	Name	Gender	Age	Location	Date
1	479065188	Chris	Male	45	West Lafayette	4pm 08/18/2007
2	47906518	Chris	Male	45	West Lafayette	4pm 08/18/2007
3	479065188	John	Male	45	Los Angels	7pm 08/18/2007
4	4790651887	Chris	Male	45	Chicago	4pm 08/18/2007
5	479065188	Chris	Male	45	Purdue University	4pm 08/18/2007

Table 1 Data Representation

Due to the possible presence of malicious source providers and inaccurate knowledge generated by intermediate agents, the information provided to the data users could be wrong or misleading. Therefore, it would be very helpful that each piece of information received by data users is rated by a trust score indicating the trustworthiness level of the information. By using the trust score, data users can determine whether they want to directly use the received information or need to further verify the information. Moreover, each data source provider (intermediate agent) is also assigned a trust score based on the amount of correct information it has provided. In the following, we present formal definitions of the level of trustworthiness for data items, knowledge items, data source providers and intermediate agents.

Definition 1. Trust of data items and knowledge items. The trustworthiness of a data items f (or a knowledge item k), denoted as $t(f)$ (or $t(k)$), is the probability of f (or k) being correct.

Definition 2. Trust of source providers and intermediate agents. The trustworthiness of a source provider s (or an intermediate agent a), denoted as $t(s)$ (or $t(a)$), is the expected trustworthiness of the data items provided by s (or a).

Different items about the same entity or event may be either supportive or conflicting. For example, one item in Table 1 claims that a person named "Chris" was in Lafayette at 4pm on 08/18/2007, while another item claims that he was in Chicago at that time. Obviously, at least one of the items is false. In another case, there is another item claiming that "Chris" was at Purdue University at 4pm on 08/18/2007, this item is more likely to report some true information provided that the first item is true. This is because Purdue University is located at West Lafayette, and the two items support each other. In order to represent such relationships, we introduce the notions of *data similarity* and *data conflict*. Specifically, data similarity models how the trust scores of similar items affect each other, while data conflict models how the trust scores of conflicting items affect each other.

In addition to data similarity and conflict, the source providers and routing paths also affect the trustworthiness of data items or knowledge items. Suppose each source provider (or intermediate agent) has associated with a probability quantifying the likelihood of reporting wrong information. The probability of multiple source providers (or intermediate agent) reporting the same wrong information is lower than that of a single source provider. Therefore, the less correlation among source providers and routing paths of the same information, the more can to trust this information. For example, the first two items in Table 1 reports the location information about the same person "Chris", both of which claimed that "Chris" was in West Lafayette at 4:00pm on 08/18/2007. If these two items came from different source providers and were routed to the data user through different paths, they can be considered as valuable supports to each other. If the two reports came from same source providers or shared very similar routing paths, the importance of considering these two reports as supportive of each other is reduced. Based on these observations, we introduce the notion of *path similarity*, which is the similarity between two *item generation paths* as defined below.

Definition 3. Item Generation Path. For an item r, let S be its source provider, and let $I_1, \ldots, I_m$ be m intermediate agents that processed r. The item generation path of r is a sequence of "$S \rightarrow I_1 \rightarrow \ldots \rightarrow I_m$".

In a network system, an item generation path (path for short) is corresponding to a sequence of IP addresses of source providers and intermediate agents. In information processing organizations, a path is corresponding to a sequence of department names. Consider the first item in Table 1. Suppose the item was generated by a source provider named "airport" and passed by intermediate agents

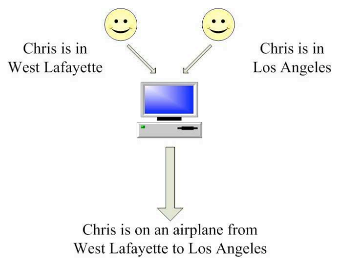

Chris is in West Lafayette
Chris is in Los Angeles
Chris is on an airplane from West Lafayette to Los Angeles

Fig. 2 An Example of Data Deduction

“$police - station1$” and “$policestation2$”, the path of this item is represented as “$airport \rightarrow policestation1 \rightarrow policestation2$”. In this paper, we assume that every source provider and intermediate agent has a unique identifier.

Items can be merged or processed by intermediate agents. As an example, consider the first and fourth items in Table 1. The first item states that “Chris” was in West Lafayette at 4:00pm 08/18/2007 and the fourth one states that “Chris” was in Los Angeles at 7:00pm 08/18/2007. Given these two inputs, an intermediate agent produces the knowledge (see Figure 2) that from 4:00pm to 7:00pm 08/18/2007 “Chris” was on an airplane from West Lafayette to Los Angeles. The trustworthiness of such generate knowledge items largely depends on the trustworthiness of input data and the agent. To model this scenario, we introduce another concept, namely *data deduction*.

3 A Trust Model for Data Provenance

In this section, we present our data provenance trust model, that we use for assigning trust scores to items (i.e., data items or knowledge items), source providers and intermediate agents. Trust scores range from 0 to 1; higher scores indicate higher trust levels.

The trust score of an item is computed by taking into account four factors: (i) data similarity; (ii) path similarity; (iii) data conflict; and (iv) data deduction. In what follows, we present the details on the evaluation of these factors.

3.1 Data Similarity

Data similarity refers to the likeness of different items. Similar items are considered as supportive to each other.

The challenge here is how to determine whether two items are similar. Consider the example in table 1. We can observe that the first two items are very similar since they both report the same locations of Chris at the same date. The only difference between these two items is a possible typo error in the person’s SSN. In contrast, the third item is different from the first two because the third one reports a totally different location. Based on these observations, we propose to employ a clustering algorithm to group items describing the same event. The purpose of the clustering is to eliminate minor errors like typos in the example and hence we adopt a strict threshold σ. After clustering, we obtain sets of items and each set represents a single event.

For each item r, the effect of data similarity on its trust score, denoted as $sim(r)$, is determined by the number of items in the same cluster and the size of the cluster. We introduce the formal definition of $sim(r)$ as follows.

$$sim(r) = e^{-\frac{\Phi_c}{N_c}} \tag{1}$$

In equation 1, N_C is the number of records in cluster C and ϕ_C is the diameter of cluster C. Here we define ϕ_C as the maximum distance between two records in the cluster. It is worth noting that items in the same cluster have same data similarity score.

We now proceed to elaborate the clustering procedure. The key of a clustering algorithm is the distance function that measures the dissimilarities among data items and the cost function which the clustering algorithm tries to minimize. The distance functions are usually determined by the type of data being clustered, while the cost function is defined by the specific objective of the clustering problem.

The data we consider in our problem contains different types of attributes. In this paper, we focus on three commonly used types of attributes, namely numerical, categorical and string. Note that it is very easy to extend our clustering method to all kinds of attributes.

- **Distance between two numerical values.**
  The numerical attribute values belong to integer, real, or date/time data types. The distance between two numerical values v_1 and v_2 is defined based on their difference as shown in equation 2, where $v_1, v_2 \in \mathtt{D}$. $|\mathtt{D}|$ is the domain size, which is measured by the difference between the maximum and minimum values in $\mathtt{D}$ defined by the system.

$$\delta_N(v_1, v_2) = |v_1 - v_2|/|\mathtt{D}| \tag{2}$$

- **Distance between two categorical values.**
  For the categorical values, we not only consider the exact match of two values, but also consider their semantics similarity. Let $\mathtt{D}$ be a categorical domain and $\mathtt{T}_{\mathtt{D}}$ be a taxonomy tree defined for $\mathtt{D}$. Then distance between two categorical values v_1 and v_2 ($v_1, v_2 \in \mathtt{D}$)[3] is defined as follows.

$$\delta_c(v_1, v_2) = H(R(v_1, v_2))/H(\mathtt{T}_{\mathtt{D}}) \tag{3}$$

  where $R(v_1, v_2)$ is the subtree rooted at the lowest common ancestor of v_1 and v_2, and $H(\mathtt{T}_{\mathtt{D}})$ represents the height of tree $\mathtt{T}_{\mathtt{D}}$.
- **Distance between two string values.**
  The distance between two string values is defined based on edit distance [6]. The edit distance between two strings of characters is the number of operations (i.e.,

change, *insert*, *delete*) required to transform one of them into the other. Given two string values v_1 and v_2, their distance is defined as follows.

$$\delta_S(v_1, v_2) = E(v_1, v_2)/|\mathtt{L}\ | \tag{4}$$

where $|v_1|$ and $|v_2|$ is the number of characters in v_1 and v_2 respectively, $\mathtt{L}$ is $\max\{|v_1|, |v_2|\}$, and $E(v_1, v_2)$ is the edit distance between v_1 and v_2.
Edit distance between two string values can be computed in $O(|v_1| \cdot |v_2|)$ time using a dynamic programming algorithm. If v_1 and v_2 have a 'similar' length, about 'n', this complexity is $O(n^2)$.

Combining the distance functions of numerical, categorical and string values, we are now able to define the *distance between two items*. Let $\mathtt{A}_{\mathtt{T}} = \{A_{N_1}, \ldots, A_{N_m}, A_{C_1}, \ldots, A_{C_n}, A_{S_1}, \ldots, A_{S_j}\}$ be the attributes of table T, where $A_{N_i}(i = 1, \ldots, m)$ is a numerical attribute, $A_{C_i}(i = 1, \ldots, n)$ is a categorical attribute and $A_{S_i}(i = 1, \ldots, j)$ is a string attribute. The distance between two items $r_1, r_2 \in T$ (denoted as $\triangle(r_1, r_2)$) is defined as follows.

$$\begin{aligned}\triangle(r_1, r_2) = &\sum_{i=1,\ldots,m} \delta_N(r_1[A_{N_i}], r_2[A_{N_i}]) + \sum_{i=1,\ldots,n} \delta_C(r_1[A_{C_i}], r_2[A_{C_i}]) \\ &+ \sum_{i=1,\ldots,j} \delta_S(r_1[A_{S_i}], r_2[A_{S_i}])\end{aligned} \tag{5}$$

Thus, the diameter of a cluster ϕ_C is computed as follows.

$$\phi_c = max_{r_1 \in C, r_2 \in C} \triangle(r_1, r_2) \tag{6}$$

Next, we present how to cluster items by using our defined distance functions. Our clustering algorithm cluster the items incrementally. First, we make the first item a cluster and the representative of that cluster. Second, for each unvisited item, we compare it with representatives of existed clusters; if we find a representative that the distance between the item and this representative is within the threshold σ and also the lest, we add this item to the cluster the representative belongs to; if it cannot find such a representative, we make this item a new cluster and the representative of this cluster.

This procedure continues until all the items have been visited. Finally, we obtain a set of clusters that the distance between the representative and the members of the cluster is within threshold σ.

Note that the value of σ is very small as the goal of the clustering process is to find most similar items. This makes it possible to randomly select an item as the representative of each cluster.

3.2 Path Similarity

As we already mentioned, we know that a larger number of similar data cannot guarantee a higher trust level of this set of data since path similarity affects the importance of supports obtained from similar data. In what follows, we show how to compute path similarity and how to integrate the impact of path similarity to the computation of the overall trust score.

Given two items r_1 and r_2, suppose their paths are P_1 and P_2 respectively. The path similarity between P_1 and P_2 is defined as follows.

$$pathsim(r_1, r_2) = \frac{max\{|P_1|, |P_2|\} - Idist}{max\{|P_1|, |P_2|\}} \tag{7}$$

where $max\{|P_1|, |P_2|\}$ is the maximum number of identifiers in the two sequences, and $Idist$ is the edit distance between two sequences. Note that unlike edit distance of two strings which is based on the difference of characters, $Idist$ is based on the difference of identifiers. An example of path similarity computation is given below.

Example 1. Suppose P_1 =“$purdueairport \rightarrow policestation1 \rightarrow policestation2$” and P_2=“$LosAngelsairport \rightarrow policestation1 \rightarrow policestation2$”. $max\{|P_1|, |P_2|\} = max\{3, 3\} = 3$. $Idist$ is 1, since only one difference exists between P_1 and P_2 which is the first identifier, the airport name. Finally, $pathsim$ is (3-1)/3=0.67.

Next, we modify equation (1) by taking into account the effect of path similarity. The new $sim(r)$ denoted as $sim^*(r)$ is defined as follows, where ω is the *path similarity factor* ranging from 0 to 1.

$$sim^*(r) = e^{-\frac{\Phi_C}{N_C \cdot \omega_C}} \tag{8}$$

In particular, ω_c is computed as follows. First, we randomly select an item from cluster C, denoted as r. We mark r and assign it a weighted value 1. Second, we randomly select an unmarked item from C, denoted as r'. We compute $pathsim$ between r' and all marked items, among which we keep the maximum $pathsim(r', r_i)$. Then we mark r' and assign it a weighted value $1 - pathsim(r', r_i)$. This procedure is continued until all items are marked. Finally, we take the average of the weighted values of all items in cluster C as ω_c.

3.3 Data Conflicts

Data conflict refers to inconsistent descriptions or information about the same entity or event. A simple example of a data conflict is that the same person appears at different locations during the same time period. It is obvious that data conflict has a negative impact on the trustworthiness of items, and hence in the following we discuss how to quantify its effect on the trust score computation.

There are various reasons for data conflicts, such as typos, false data items generated by malicious source providers, or misleading knowledge items generated by intermediate agents. Data conflict largely depends on the knowledge domain of the specific application. Therefore, our trust model allows users to define their own data conflict functions according to their application-dependent requirements.

To determine if two items conflict with each other, data users first need to define the exact meaning of conflict, which we call *prior knowledge*. Consider the example in Table 1 again. The attribute value of "SSN" in the first item is the same as that in the third item, but the attribute value of "Name" in the first item is different from that in the third one. This implies a data conflict, since we know that each single SSN should correspond to only one individual. We can further infer that there should be something wrong with either source providers (airports) or intermediate agents (police stations) whichever handled these two items. The prior knowledge we use here is that $if\ r_1(\text{“}SSN\text{”}) = r_2(\text{“}SSN\text{”})\ then\ r_1(\text{“}Name\text{”}) = r_2(\text{“}Name\text{”})$. If any two items cannot satisfy the condition stated by such prior knowledge, these two items are considered conflicting with each other.

To facilitate automatic conflict detection, we propose a simple language for data users to define prior knowledge. Prior knowledge with respect to a set of items contains a set of prior knowledge items which are defined as follows.

Definition 4. Prior Knowledge Item. Let r_1 be any item that has a set of attributes $\mathtt{A}_{r_1}$, and r_2 be any item that has a set of attributes $\mathtt{A}_{r_2}$. Let $\mathtt{A}_1, \mathtt{A}_2 \subseteq \mathtt{A}_{r_1} \cup \mathtt{A}_{r_2}$, and let $condition_1$ and $condition_2$ be Boolean expressions on $\mathtt{A}_1$ and $\mathtt{A}_2$ respectively. A *prior knowledge item* with respect to r_1 and r_2 has the form of: $condition_1 \Longrightarrow condition_2$. We say that r_1 and r_2 satisfy this prior knowledge item iff when $condition_1$ is evaluated true on the attribute values of r_1 and r_2, $condition_2$ are also evaluated true on the attribute values of r_1 and r_2.

Based on prior knowledge, we define the data conflict between two items.

Definition 5. Conflicting Items. Let r_1 and r_2 be two items, and PK_1, ..., PK_m be prior knowledge items. We say that r_1 conflicts with r_2 iff $\exists PK_i(1 \leq i \leq m)$, r_1 and r_2 cannot satisfy PK_i.

We proceed now to discuss how to automatically detect data conflicts according to the above definitions. Recall that items are clustered by using a strict threshold and hence items in the same cluster are very similar to one another. By leveraging this property, we do not need to compare all items to find out the conflicting pairs. Instead, we only check the representative of each cluster. Suppose that the representatives of two clusters C_1 and C_2 conflict with each other. The data conflict score of one cluster against another cluster is determined by the distance between two clusters and the number of items in the second cluster taking into account path similarity. In particular, we have the following equation of data conflict.

$$con_{c_1}(C_1, C_2) = e^{-\frac{1}{d(C_1,C_2)\cdot N_{c_2}\cdot \omega_{c_2}}} \tag{9}$$

where N_{c_2} is the number of items in cluster C_2, $d(C_1, C_2)$ is the distance between two clusters and ω_{c_2} is the path similarity factor.

3.4 Data Deduction

The trustworthiness of a knowledge item also depends on the trustworthiness of information used to generate it and the trustworthiness of parties that handle it. Generally speaking, if the source information or the responsible party is highly trusted, the resulting data will also be highly trusted. We define the function of data deduction as follows.

Definition 6. Data Deduction. Let a be an intermediate agent, and let k be a knowledge item generated by a based on items $r_1, ..., r_n$. The data deduction of k, represented as a function $Ded_k(t(a), t(r_1), ..., t(r_n))$, indicates the impact of the trustworthiness of r_1,..., r_n, and a on the trustworthiness of k.

We compute the effect of trustworthiness of a source provider or an agent on its resulting data by taking into account the actions it took on this data. Types of actions may differ in different applications. In this paper, we consider two typical actions, "PASS" and "INFER". "PASS" means merely passing data to another agent or data user, and "INFER" means that the agent produces a knowledge item based on the input information and possibly some local knowledge. Different actions may have different impact on the *trustworthiness* of the output information. For example, "PASS" operation simply passes the input items to successive parties. Since such operation does not change the content of the information, the trustworthiness of the output should be the same as that of the input if no error is introduced during the transmission. By contrast, "INFER" generates new knowledge based on the input information.

Given a set of items and an intermediate agent, we employ a weighted function to compute the trust score of the output information.

$$t(k) = \frac{w_i \cdot t(a) + \frac{\sum_{j=1}^{n} t(r_j)}{n}}{2} \tag{10}$$

Here, w_i is a parameter based on the operation the intermediate agent takes and its impact on the trustworthiness of knowledge k.

3.5 Computing Trust Scores

So far, we are clear how the four aspects influence the trustworthiness of items. In this subsection, we will present how to combine the effects of these four aspects to obtain an overall trust score for data items, knowledge items, source providers and intermediate agents.

The computation of the overall trust scores is an iterative procedure. Initially, we assign each source provider and intermediate agent an initial trust score by querying the information that the end users already knew. The initial *trustworthiness* of each data item and knowledge item is then set to the *trustworthiness* of its source providers and intermediate agent, denoted as $t(f)$ and $t(k)$ respectively. Then, we start the iteration. At each iteration, there are four main steps.

First, we update current trust score of each data item and knowledge item by using the $sim^*(r)$ function. For a data item f in cluster C with $sim^*(f)$, its updated trustworthiness is defined as:

$$t(f) = 1 - \prod_{r \in C}(1 - t(r) \cdot sim^*(f)) \tag{11}$$

The above equation can be rewritten as follows.

$$1 - t(f) = \prod_{r \in C}(1 - t(r) \cdot sim^*(f)) \tag{12}$$

After taking logarithm on both sides, we have

$$\ln(1 - t(f)) = \sum_{r \in C} \ln(1 - t(r) \times sim^*(f)) \tag{13}$$

Let $\tau(f) = -\ln(1 - t(f))$, and $\varphi(r) = -\ln(1 - t(r) \times sim^*(f)$. Equation 13 is revised as follows.

$$\tau(f) = \sum_{r \in C} \varphi(r) \tag{14}$$

Second, we integrate the effect of data conflict into the current trust score. For a f in cluster C_1, if C_1 conflicts with C_2, we update $\tau(f)$ to $\tau^*(f)$ as follows.

$$\tau^*(f) = \tau(f) + \tau(r_{c_2}) \times con_{c_1}(C_1, C_2) \tag{15}$$

where r_{c_2} is the representor of C_2 and $\tau(r_{c_2})$ is its trust score. Then, we revise $t(f)$ as follows.

$$t(f) = 1 - e^{-\tau^*(f)} \tag{16}$$

Similarly, we can compute $t(k)$ for each knowledge item.

Third, we consider the data deduction for the knowledge item and update $t(k)$.

Finally, we compute the trust scores for source providers S and intermediate agents I, denoted as $t(S)$ and $t(I)$ respectively. $t(S)$ is computed as the average trustworthiness of data items provided by S. $t(I))$ is computed as the average trustworthiness of data items and knowledge items provided by I.

$$t(S) = \frac{\sum_{f \in F(S)} t(f)}{|F(S)|} \tag{17}$$

$$t(I) = \frac{\sum_{f \in F(I)} t(f) + \sum_{k \in K(I)} t(k)}{|F(I)| + |K(I)|} \tag{18}$$

where $F(S)$ ($F(I)$) is the set of data items provided by S (I), and $K(I)$ is the set of knowledge items generated by I. The iteration stops when the changes on trust scores becomes negligible. An overview of the algorithm is shown in Figure 3.

4 Performance Study

Our experiments are conducted on a 2.6-GHz Pentium 4 machine with 1 Gbyte of main memory.

Due to the lack of appropriate real datasets, we generate synthetic datasets according to the scenario given in Table 1. We first generate a set of seeds which

Procedure Turst_Score_Computation

```
1. cluster data facts and knowledge items
2. for each cluster
3.    compute data similarity
4.    compute path similarity
5. compute data conflict
6. assign initial trust scores to all the source providers
   intermediate agents
7. repeat
8.    for each data fact and knowledge item
9.        compute its trust score
10.   for each knowledge item
11.       compute data deduction
12.       recompute trust score of the knowledge item
          by combining the effect of data deduction
13.   compute trust scores for all the source providers
      and intermediate agents
14. until the change of trust scores is ignorable
```

Fig. 3 Trust Score Computation

are tuples with seven attributes as shown in Table 1. Each seed represents a unique event. To generate items in the dataset, we randomly select a seed and slightly modify a randomly chosen attribute of this seed. In order to simulate data conflict, we use a prior knowledge item stating that items with same SSN and similar name must be in the same location at the same time. Then we modify the location information in the same set of seeds to generate conflicting items. The percentage of conflicting items is set to 10%. In this way, we ensure that a dataset contains both similar items and conflicting items.

To generate the item generation path, we constructs a network containing 100 source providers and 1000 intermediate agents. For each item, its generation path is then generated by randomly selecting a source provider and multiple agents. We have tested the cases when the average length of item generation path varies from 5 to 10. Table 2 offers an overview of the parameters used in the ensuing experiments, where values in bold are default values.

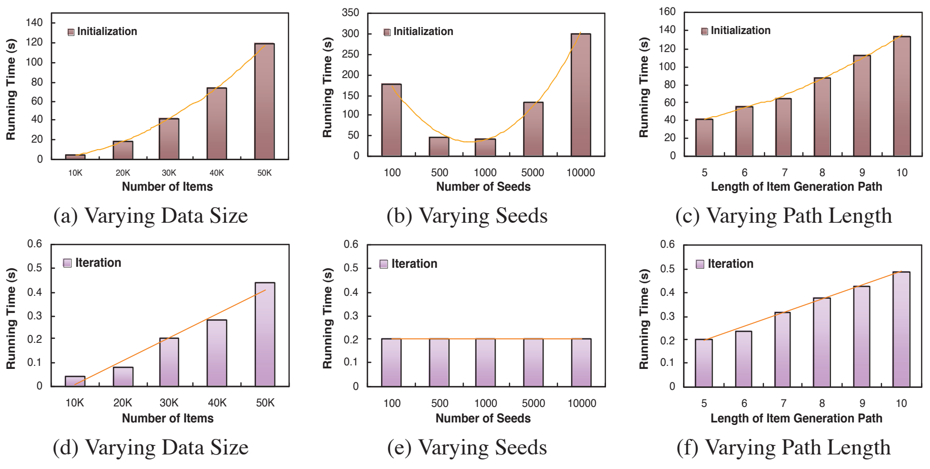
At current stage we mainly focus on evaluating efficiency of our approach, i.e., running time. We plan to carry out human studies in the future to evaluate the effectiveness of our approach.

4.1 Effect of Varying Data Sizes

We first study the efficiency of our approach when varying the size of datasets from 10K to 50K. The trust score computation consists of two main phases: initialization phase (steps 1-6 in the algorithm in Figure 3) and iteration phase (steps 7-14 in Figure 3). We plot running time of the two phases in Figure 4(a) and (d) respectively.

Parameter	**Setting**
Number of source providers	100
Number of intermediate agents	1000
Number of seeds	100, 500, **1000**, 5000, 10000
Percentage of conflicting items	10%
Average length of path	**5**, 6, 7, 8, 9, 10
Dataset Size	10k, 20k, **30k**, 40k, 50k

Table 2 Parameters and Their Settings



(a) Varying Data Size (b) Varying Seeds (c) Varying Path Length

(d) Varying Data Size (e) Varying Seeds (f) Varying Path Length

Fig. 4 Experimental Results

As shown in Figure 4(a), the running time of initialization phase increases quickly when the dataset size becomes large. This is because in the worst case the complexity of the clustering algorithm, the computation of data similarity, path similarity and data conflict are $O(n^2)$. Although the initialization phase looks costly, it takes less than two minutes for a datast of 50K items, which is still practical for off-line applications. Also, we need to note that this phase needs to be executed only once as long as the items remain the same. That is, the results obtained from this phase can be reused when trust scores of source providers or agents are changed. Further, an insertion or a deletion of an item only affects several clusters containing or close to this item, which means a large portion of previously computed results are still valid.

Compared to the initialization phase, the iteration phase is much faster (see Figure 4(d)). It needs less than one second to compute trust scores for 50K items. This is because the iteration phase simply computes score functions based on the results obtained from initialization phase and trust scores converge to stable values in a short time using our algorithm.

4.2 Effect of Varying the Number of Seeds

Next, we study the performance when varying the number of seeds to generate a dataset of 30K items. As shown in Figure 4(b), the running time of the initialization phase first decreases and then increases as the number of seeds increases. The reason of such behavior is the following. The number of seeds corresponds to the number of clusters. When there are few clusters, the computation of data conflict among clusters is very fast and most time is spent on computing data similarity and path similarity within the same cluster. In contrast, when the number of clusters is large, most time is spent on computing data conflict among clusters.

As for the iteration phase shown in Figure 4(e), the running time is not affected by the change of the number of seeds. This is because the performance of iteration phase is determined by the number of items and the network structure that affects data deduction computation. As long as these two factors remain unchanged, the performance is constant.

4.3 Effect of Varying the Length of Item Generation Path

In this experiment, we examine the effect of varying the average length of item generation path. As shown in Figure 4(c) and (f), the running time of both phases increases with the length of item generation path. This is because the longer the path is, the more time is needed to compare the similarity between two paths in initialization phase and the more time to compute the data deduction in the iteration phase.

5 Related Work

Whereas there has been a lot of work dealing with provenance [2, 4, 5], no work deals with evaluating trust of data sources and provenance paths. The most relevant work is by Yin et al. [7], which mainly deals conflicting information provided by different websites and aims at discovering *true* facts. Compared to our trust model, their approach has several drawbacks. First, they do not show how to compute implication between two items, whereas we introduce the notions of data similarity and data conflicts and propose an approach to quantify them. Second, their model contains a large number of of parameters which are hard to determine. Third, they assume that an identifier exists able to link different items corresponding to the same object. Without a correct identifier, their approach does not work. Such an assumption is strong and seems unrealistic in many real applications. In contrast, our approach can deal with more general situations.

6 Conclusions

In this paper we introduce and formulate the problem of determining the trustworthiness of data and data providers in the context of a data provenance framework. To address this problem, we propose a trust model which assigns trust scores to both data and data providers. We have considered four important factors that influence trustworthiness, and our trust model effectively combines these factors during the trust score computation. We have evaluated the efficiency of our approach and the experimental results indicate the feasibility of our approach. In the near future we plan to evaluate the effectiveness of our system. We will also investigate how to dynamically adjust our trust model when new information keeps streaming into the system.

References

1. C. Batini and M. Scannapieco. Data quality: Concepts, methodologies and techniques. *Springer*, 2006.
2. P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.
3. J.-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k -anonymization using clustering techniques. In *12th International Conference on Database Systems for Advanced Applications, (DASFAA'07)*, pages 188–200, 2007.
4. M. Greenwood, C. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau, and T. Oinn. Provenance of e-science experiments - experience from bioinformatics. In *UK OST e-Science second All Hands Meeting*, 2003.
5. D. P. Lanter. Design of a lineage-based meta-data base for gis. *Cartography and Geographic Information Systems*, 18:255–261, 1991.
6. R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.
7. X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (KDD'07)*, pages 1048–1052, 2007.