Integrating Conflicting Data: The Role of Source Dependence

Xin Luna Dong
AT&T Labs–Research
180 Park Ave.
Florham Park, NJ, USA
lunadong@research.att.com

Laure Berti-Equille*
Université de Rennes 1
Campus de Beaulieu
Rennes cedex, France
berti@irisa.fr

Divesh Srivastava
AT&T Labs–Research
180 Park Ave.
Florham Park, NJ, USA
divesh@research.att.com

ABSTRACT

Many data management applications, such as setting up Web portals, managing enterprise data, managing community data, and sharing scientific data, require integrating data from multiple sources. Each of these sources provides a set of values and different sources can often provide conflicting values. To present quality data to users, it is critical that data integration systems can resolve conflicts and discover true values. Typically, we expect a true value to be provided by more sources than a particular false one, so we can take the value provided by the majority of the sources as the truth. Unfortunately, a false value can be spread through copying and that makes truth discovery extremely tricky. In this paper, we consider when there are a large number of sources among which some may copy from others, how to find true values from conflicting information.

We present a novel approach that considers *dependence* between data sources in truth discovery. Intuitively, if two data sources provide a large number of common values and many of these values are rarely provided by other sources (*e.g.*, particular false values), it is very likely that one copies from the other. We apply Bayesian analysis to decide dependence between sources and design an algorithm that iteratively detects dependence and discovers truth from conflicting information. We also extend our model by considering *accuracy* of data sources and *similarity* between values. Our experiments on synthetic data as well as real-world data show that our algorithm can significantly improve accuracy of truth discovery and is scalable when there are a large number of data sources.

1. INTRODUCTION

Many data management applications require integrating data from multiple sources, each of which provides a set of values as "facts". However, "facts and truth really don't have much to do with each other" (*by William Faulkner*). Different sources can often provide conflicting values, some being true while some being false. To provide quality data to users, it is critical that data integration systems can resolve conflicts and discover true values. Typically, we expect a true value to be provided by more sources than a particular false one, so we can apply voting and take the value provided by the majority of the sources as the truth. Unfortunately, a value provided by one data source, no matter true or false, can be copied by many other sources. "A lie told often enough becomes the truth" (*by Vladimir Lenin*); telling the truth from conflicting information becomes extremely tricky in such a situation. In this paper, we consider the following problem: from the conflicting values provided by a large number of sources among which some may copy from others, how can one decide which is the true value?

We are mainly motivated by integrating data from the Web. As an example, consider *AbeBooks.com*, which integrates information on books from different online bookstores. A search on "Digital Visual Fortran Programmer's Guide" (ISBN: 1555582184) returned 12 bookstores that sell the book[1]. Among them 7 claimed Michael Etzel is the author, 4 claimed Michael Etzel and Karen Dickinson together are the authors, and 1 claimed Micheal Etzel and Karen Dickinson are the authors. The website needs to decide who on earth authored the book so that it can correctly answer queries such as asking for books authored by Karen Dickinson. Although there are more bookstores claiming that Michael Etzel by himself is the author, some of them are likely to derive data from others and indeed, both Michael Etzel and Karen Dickinson are the authors. Similar scenarios can arise in integrating enterprise information, managing community data, sharing scientific data, and so on.

Ideally, when applying voting we would like to ignore information that is copied; however, this raises at least three challenges. First, in many applications we do not know how each source obtains its data, nor do we have an update log of the sources, so we have to discover copiers from a snapshot of data. The discovery is non-trivial as sharing common data does not in itself imply copying; for example, two sources can both know the true values and provide them independently. Second, even when we decide that two sources are dependent, with only a snapshot of data it is not obvious which of them is a copier. Third, a copier can also provide some data by itself or verify some of the copied data, so it

*Work conducted when visiting AT&T Labs–Research, supported by the Marie Curie International Fellowship within the 6th European Community Framework Programme (FP6-MOIF-CT-2006-041000).

VLDB '08, August 24-30, 2008, Auckland, New Zealand

[1]This result is according to a data set collected by [13] in 2007.

Table 1: The motivating example: five data sources provide information on the affiliations of five researchers. Only S_1 provides all true values.

	S_1	S_2	S_3	S_4	S_5
Stonebraker	MIT	Berkeley	MIT	MIT	MS
Dewitt	MSR	MSR	UWisc	UWisc	UWisc
Bernstein	MSR	MSR	MSR	MSR	MSR
Carey	UCI	Oracle	BEA	BEA	BEA
Halevy	Google	Google	UW	UW	UW

is inappropriate to ignore all data it provides.

In this paper, we present a novel approach that considers *dependence* between data sources in truth discovery. Our technique considers not only whether two sources share the same values, but also whether the shared values are true or false. Intuitively, for a particular object, there are often multiple distinct false values but usually only one true value. Sharing the same true value does not necessarily imply sources being dependent; however, sharing the same false value is typically a rare event when the sources are fully independent. Thus, if two data sources share a lot of false values, they are more likely to be dependent. We develop Bayes models that compute the probability of two data sources being dependent and take the result into consideration in truth discovery. Note that detection of dependence between data sources is based on recognition of true values, whereas correctly deciding true values requires knowledge of source dependence. There is an inter-dependence between them and we solve the problem by iteratively deciding dependence between sources and discovering truth from conflicting information. To the best of our knowledge, source-dependence analysis has not been investigated for the purpose of truth discovery.

We also consider *accuracy* in voting: we trust the values provided by an accurate data source more and give them a higher weight. This method requires identifying not only if a pair of sources are dependent, but also which source is the copier. Indeed, accuracy in itself is a clue in deciding the direction of dependence: given two data sources, if the accuracy of their common data is highly different from that of one of the sources, that source is more likely to be a copier. Note that considering accuracy of sources in truth discovery has been explored in [13]. Whereas we share the basic idea, we present a different model for computing source accuracy and extend it to incorporate the notion of source dependence. We present more detailed comparison in Section 5.4.

We now illustrate the main techniques of the paper with an example.

EXAMPLE 1.1. *Consider the five data sources in Table 1. They provide information on affiliations of five researchers and only S_1 provides the correct affiliations for all researchers. However, since the affiliations provided by S_3 are copied by S_4 and S_5 (with certain errors during copying), a naive voting would consider them as the majority and so make wrong decisions for three out of five researchers.*

We solve the problem by considering dependence between data sources. If we knew which values are true and which are false, we would suspect that S_3, S_4 and S_5 are dependent, because they provide the same false values. On the other hand, we would suspect the dependence between S_1 and S_2 much less, as they share only true values. Based on this analysis, we would ignore the values provided by S_4 and S_5 and then be able to decide the correct affiliation for four out of five researchers (except Carey).

Further, we consider accuracy of data sources. Based on the voting results we have obtained with consideration of source dependence, S_1 is more accurate than S_2 and S_3. Thus, we would trust S_1 more and this leads to the correct decision on Carey's affiliation. Note that if we do not consider dependence between S_3, S_4 and S_5, we would consider S_3 and S_4 as the most accurate and that further strengthens the wrong information they provide. □

In summary, our paper makes the following three contributions:

1. First, we formalize the notion of source dependence and presents a classification of copiers. We develop Bayes models that are able to discover copiers of various kinds.
2. Second, we incorporate the notion of accuracy of sources in the analysis of source dependence, and design an algorithm that considers both dependence between sources and accuracy of sources in truth discovery. We further extend this algorithm by considering similarity between values and distribution of false values.
3. Finally, we tested our algorithms on synthetic data and a real-world data set. The experimental results show that our algorithm can significantly improve accuracy of truth discovery, is robust in presence of particular cases of falsification by malicious sources, and is scalable when there are a large number of data sources.

We envision our work as a first step towards integrating data from multiple sources where some may copy from others. We expect our techniques to have broad impact on various aspects of data sharing, including resolving conflicts from multiple sources, removing duplications of various forms, answering queries efficiently from multiple sources with awareness of copiers, and recommending trustworthy data sources.

This paper is structured as follows. Section 2 formally defines the problem we solve and the notion of dependence between data sources. Section 3 and 4 describe the core models that detect copiers and discover truths accordingly. Section 5 describes our algorithm that considers both dependence and accuracy in truth discovery. Section 6 presents several extensions and Section 7 describes experimental results. Finally, Section 8 discusses related work and Section 9 concludes.

2. OVERVIEW

This section formally describes the problem we solve, defines dependence between data sources, and overviews models we present in this paper.

2.1 Problem statement

We consider a set of *data sources* $\mathcal{S}$ and a set of *objects* $\mathcal{O}$. An object represents a particular aspect of a real-world entity, such as the affiliation of a person; in a relational database, an object corresponds to a cell in a table. For each object $O \in \mathcal{O}$, a source $S \in \mathcal{S}$ can (but not necessarily) provide a *value*. Among different values provided for an object, one correctly describes the real world and is *true*, and the rest are *false*. Table 2 lists the variables and parameters

Table 2: Variables and parameters in this paper.

Name	Description
$\mathcal{O}$	Objects
O	An object
$\mathcal{V}(O)$	Domain of O
$n(O)$	Number of incorrect values of O
$\varepsilon(O)$	Probability of making an error on an independently provided value for O
$\Psi(O)$	Observation of which source votes for which value on O
v	A value
$\bar{S}_o(v)$	The set of sources that provide value v on object O
$P(v)$	The probability that v is a correct value of O
$C(v)$	The confidence of v is a correct value of O
$V(v)$	The total vote count of v as a value of O
$\mathcal{S}$	Data sources
S	A data source
$S(O)$	The value provided by data source S on object O
$A(S)$	The accuracy of S
$A'(S)$	The accuracy score of S. $A'(S) = -\ln(\frac{1}{A(S)} - 1) + \ln n$
$\varepsilon(S)$	Error rate of S. $\varepsilon(S) = 1 - A(S)$
$I(S)$	The vote count of S on a particular value
$S_1 \perp S_2$	S_1 and S_2 are independent of each other
$S_1 \sim S_2$	S_1 and S_2 are dependent
$S_1 \rightarrow S_2$	S_1 copies from S_2
O_t	Objects where two data sources provide the same true value
k_t	Size of O_t
O_f	Objects where two data sources provide the same False value
k_f	Size of O_f
O_{d1}	Objects where S_1 provides the true value while S_2 provides a false one
k_{d1}	Size of O_{d1}
O_{d2}	Objects where S_2 provides the true value while S_1 provides a false one
k_{d2}	Size of O_{d2}
O_{d0}	Objects where two data sources provide different false values
k_{d0}	Size of O_{d0}
Φ	Observation of distribution of O_t, O_f, O_{d1}, O_{d2} and O_{d0}
α	A-priori probability of dependence between two data sources
c	Percentage of copied values over all values provided by a copier
ε	Probability of making an error on an independently provided value
n	Number of incorrect values for each object in the underlying domain

we use in this paper and we shall explain each of them at the time of use.

In this paper we solve the following problem: given a snapshot of data sources in $\mathcal{S}$, decide the true value for each object $O \in \mathcal{O}$.

We note that in real-world applications, the value provided by a data source can either be atomic or a set or list of atomic values (*e.g.*, author list). In the latter case, we consider the set or list as a whole. This problem setting already fits many real-world applications and the solution is non-trivial. There exist applications where the true value is a set of atomic values while each data source typically provides one or a few of the values. We defer handling this situation to future work.

2.2 Dependence between sources

We say there exists a *dependence* from data source S to T if S copies directly from T. We allow that S copies only a subset of T's values, revises some of the copied values, and adds additional values; though, in the latter two cases, the revised and added values are considered as independent contributions by S. Note that this definition considers only *direct* copying; in other words, the fact that source A copies from B and B copies from C does not imply dependence from A to C, as A may not copy directly from C. We allow a copier to copy from multiple sources (by union, intersection, etc.). As we consider only a snapshot of data, cyclic copying on a particular object is impossible.

Based on this definition, there are two types of data sources: *independent sources* and *copiers*. An *independent source* provides all values independently. We further distinguish *good* independent sources from *bad* ones: a data source is considered to be good if for each object it is more likely to provide the true value than a *particular* false value; otherwise, it is considered to be bad.

A *copier* copies some values from other sources (independent sources or copiers) and provides the rest of the values independently (by examining and revising copied values or adding new values). We categorize copiers into *benevolent* and *malicious* ones. A copier is *benevolent* if each value it provides independently is more likely to be a true value than a particular false one and *malicious* otherwise. For exam-

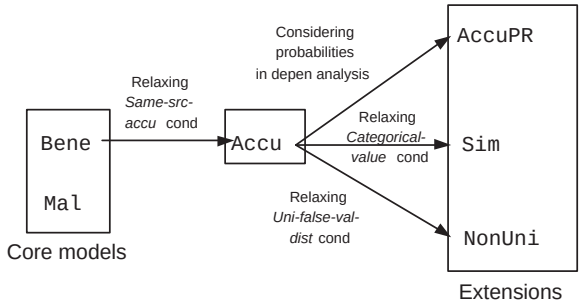

Figure 1: Models for truth discovery.

ple, a copier that tries to fix some false values it copies is benevolent, whereas a copier that randomly changes some copied values to avoid being detected as a copier, or makes random mistakes in copying, is malicious. As a special case, a copier that does not provide any value independently is considered as benevolent.

2.3 Models in this paper

We start our discussion from a core case that satisfies the following three conditions:

- **Same source accuracy:** For each object, all independent data sources have the same probability of providing a true value.
- **Uniform false-value distribution:** For each object, there are multiple false values in the underlying domain and an independent source has the same probability of providing each of them.
- **Categorical value:** For each object, values that do not match exactly are considered as completely different.

In this core case, independent sources are *good* under the following condition. For each $O \in \mathcal{O}$, let $\varepsilon(O)$ be the probability that a source provides a false value (*i.e.*, error rate) on O and $n(O)$ be the number of false values on O in the underlying domain. Then, if $1-\varepsilon(O) > \frac{\varepsilon(O)}{n(O)}$ (*i.e.*, $\varepsilon(O) < \frac{n(O)}{n(O)+1}$), independent sources in $\mathcal{S}$ are good. Intuitively, given such a set of independent data sources, we can discover true values by voting. The following proposition, which we prove and generalize in Section 5, formalizes this intuition.

PROPOSITION 2.1 (VOTING). *Let O be an object and $\bar{S}_O$ be a set of independent sources voting for O. In the core case, if $\varepsilon(O) < \frac{n(O)}{n(O)+1}$, among the different values on O provided by $\bar{S}_O$, the one provided by the maximum number of sources has a higher probability to be true.* □

Even for this core case, discovering dependence between data sources and deciding true values are non-trivial; we solve the problem by two models BENE (Section 3) and MAL (Section 4). Then, we relax the *Same-source-accuracy* condition and present the ACCU model (Section 5). As extensions (Section 6), we present the ACCUPR model considering probabilities of a value being true in dependence discovery, the SIM model relaxing the *Categorical-value* condition, and the NONUNI model relaxing the *Uniform-false-value-distribution* condition. Figure 1 depicts the relationships between these models.

Throughout the paper, all of our models make the following assumptions.

- **Assumption 1 (Independent values).** The values independently provided by a data source on different objects are independent of each other.
- **Assumption 2 (Independent copying).** The dependence between a pair of data sources is independent of the dependence between any other pair of data sources.
- **Assumption 3 (Random copying).** A copier randomly copies a fraction (or all) of the values provided by the original source.

Our experimental results on real-world data show that even when these assumptions might be violated, our techniques still work well. Note that under Assumption 3, loop copying between data sources is impossible; otherwise, a data source can end up copying values provided by itself.

3. MODEL FOR BENEVOLENT COPIERS

This section describes how we detect benevolent copiers and discover truth from conflicting information accordingly. We consider malicious copiers in the next section.

3.1 Dependence of data sources

Assume $\mathcal{S}$ consists of two types of data sources: good independent sources and benevolent copiers. Let n ($n > 1$) be the number of false values in the underlying domain for each object. Let c ($0 < c \leq 1$) be the probability that a value provided by a copier is copied. Let ε ($0 \leq \varepsilon < \frac{n}{n+1}$) be the *error rate*; that is, the probability that an independently provided value is false.

Given two data sources S_1 and S_2, we want to compute the probability that they are dependent. Let $\bar{O}_t$ be the set of objects on which they provide the same true value, $\bar{O}_f$ be the set of objects on which they provide the same false value, and $\bar{O}_d$ be the set of objects on which they provide different values. These three sets are disjoint and $\bar{O}_t \cup \bar{O}_f \cup \bar{O}_d \subseteq \mathcal{O}$. We denote by Φ the observation of $\bar{O}_t, \bar{O}_f$ and $\bar{O}_d$ and denote by k_t, k_f and k_d their sizes respectively. Intuitively, it is more likely for a good independent source to provide the true value than a particular false value on an object; thus, if we fix $\bar{O}_t \cup \bar{O}_f$ and $\bar{O}_d$, the more common false values that S_1 and S_2 provide, the more likely that they are dependent. On the other hand, if we fix $\bar{O}_t$ and $\bar{O}_f$, the fewer objects on which S_1 and S_2 provide different values, the more likely that they are dependent. We conduct a probability analysis based on these intuitions.

We first consider what is the probability of observing Φ if S_1 and S_2 are independent, denoted by $S_1 \perp S_2$. Since there is a single true value, the probability that S_1 and S_2 provide the same true value for object O is

$$Pr(O \in \bar{O}_t | S_1 \perp S_2) = (1-\varepsilon)^2. \quad (1)$$

Under the *Uniform-false-value-distribution* condition, the probability that a data source provides a particular false value for object O is $\frac{\varepsilon}{n}$. Thus, the probability that S_1 and S_2 provide the same false value for O is

$$Pr(O \in \bar{O}_f | S_1 \perp S_2) = n \cdot (\frac{\varepsilon}{n})^2 = \frac{\varepsilon^2}{n}. \quad (2)$$

Then, the probability that S_1 and S_2 provide different values on an object O, denoted by P_d for convenience, is

$$Pr(O \in \bar{O}_d | S_1 \perp S_2) = 1 - (1-\varepsilon)^2 - \frac{\varepsilon^2}{n} = P_d. \quad (3)$$

Following the *Independent-values* assumption, the conditional probability of observing Φ is

$$Pr(\Phi|S_1 \perp S_2) = \frac{(1-\varepsilon)^{2k_t}\varepsilon^{2k_f}P_d^{k_d}}{n^{k_f}}. \quad (4)$$

We next consider what is the probability of observing Φ if S_1 and S_2 are dependent, denoted by $S_1 \sim S_2$. There are two cases where S_1 and S_2 provide the same value v. First, with probability c, one copies v from the other and so v is true with probability $1-\varepsilon$ and false with probability ε. Second, with probability $1-c$, the two sources provide v independently and so its probability of being true or false is the same as in the case where S_1 and S_2 are independent. Thus, we have

$$Pr(O \in \bar{O}_t|S_1 \sim S_2) = (1-\varepsilon)\cdot c + (1-\varepsilon)^2 \cdot (1-c), \quad (5)$$

$$Pr(O \in \bar{O}_f|S_1 \sim S_2) = \varepsilon \cdot c + \frac{\varepsilon^2}{n} \cdot (1-c). \quad (6)$$

Finally, the probability that S_1 and S_2 provide different values on an object is that of S_1 providing a value independently and the value differs from that provided by S_2:

$$Pr(O \in \bar{O}_d|S_1 \sim S_2) = P_d \cdot (1-c). \quad (7)$$

Thus, the conditional probability of observing Φ is

$$Pr(\Phi|S_1 \sim S_2) = \frac{(1-\varepsilon)^{k_t}(1-\varepsilon+c\varepsilon)^{k_t}\varepsilon^{k_f}(cn+\varepsilon-c\varepsilon)^{k_f}(1-c)^{k_d}P_d^{k_d}}{n^{k_f}} \quad (8)$$

By applying the Bayes Rule, we obtain the probability of $S_1 \sim S_2$ conditioned on the observation of Φ as follows.

$$\begin{aligned} &Pr(S_1 \sim S_2|\Phi) \\ &= \frac{Pr(\Phi|S_1 \sim S_2)Pr(S_1 \sim S_2)}{Pr(\Phi|S_1 \sim S_2)Pr(S_1 \sim S_2) + Pr(\Phi|S_1 \perp S_2)Pr(S_1 \perp S_2)} \\ &= \left(1 + (\frac{1-\alpha}{\alpha})(\frac{1-\varepsilon}{1-\varepsilon+c\varepsilon})^{k_t}(\frac{\varepsilon}{cn+\varepsilon-c\varepsilon})^{k_f}(\frac{1}{1-c})^{k_d}\right)^{-1} \end{aligned} \quad (9)$$

Here $\alpha = Pr(S_1 \sim S_2)(0 < \alpha < 1)$ is the a-priori probability that two data sources are dependent. Note that we should avoid setting α to 0 or 1, because otherwise $Pr(S_1 \sim S_2|\Phi) = \alpha$ and no Φ can change our a-priori belief of the likelihood of dependence. We discuss how we set the parameters at the end of this section.

Equation (9) has several nice properties that conform to the intuitions we discussed early in this section, formalized as follows.

THEOREM 3.1. *Let $\mathcal{S}$ be a set of good independent sources and benevolent copiers. Equation (9) has the following three properties on $\mathcal{S}$.*

1. *Fixing $k_t + k_f + k_d$, when $k_t + k_f$ increases and none of k_t and k_f decreases, the probability of dependence increases;*
2. *Fixing $k_t + k_f$ and k_d, when k_f increases, the probability of dependence increases;*
3. *Fixing k_t and k_f, when k_d decreases, the probability of dependence increases.* □

PROOF. We prove the three properties as follows.

1. Let $k_0 = k_t + k_f + k_d$. Then, $k_d = k_0 - k_t - k_f$. We have

$$\begin{aligned} &Pr(S_1 \sim S_2|\Phi) \\ &= \left(1 + (\frac{1-\alpha}{\alpha})(\frac{1-\varepsilon-c+c\varepsilon}{1-\varepsilon+c\varepsilon})^{k_t}(\frac{\varepsilon-c\varepsilon}{cn+\varepsilon-c\varepsilon})^{k_f}(\frac{1}{1-c})^{k_0}\right)^{-1}. \end{aligned}$$

   As $0 < c < 1$, we have $0 < \frac{1-\varepsilon-c+c\varepsilon}{1-c\varepsilon} < 1$ and $0 < \frac{\varepsilon-c\varepsilon}{cn+\varepsilon-c\varepsilon} < 1$. When k_t or k_f increases, $(\frac{1-\varepsilon-c+c\varepsilon}{1-c\varepsilon})^{k_t}$ or $(\frac{\varepsilon-c\varepsilon}{cn+\varepsilon-c\varepsilon})^{k_f}$ decreases. Thus, $Pr(S_1 \sim S_2|\Phi)$ increases.
2. Let $k_c = k_t + k_f$. Then, $k_t = k_c - k_f$. We have

$$\begin{aligned} &Pr(S_1 \sim S_2|\Phi) \\ &= \left(1 + (\frac{1-\alpha}{\alpha})(\frac{1-\varepsilon}{1-\varepsilon+c\varepsilon})^{k_c}(\frac{\varepsilon(1-\varepsilon+c\varepsilon)}{(1-\varepsilon)(cn+\varepsilon-c\varepsilon)})^{k_f}(\frac{1}{1-c})^{k}\right)^{-1}. \end{aligned}$$

   Because $\varepsilon < \frac{n}{n+1}$, $\varepsilon(1-\varepsilon+c\varepsilon) < (1-\varepsilon)(cn+\varepsilon-c\varepsilon)$. Thus, when k_f increases, $(\frac{\varepsilon(1-c\varepsilon)}{(1-\varepsilon)(n-cn+c\varepsilon)})^{k_f}$ decreases and so $Pr(S_1 \sim S_2|\Phi)$ increases.
3. Because k_d increases, $(\frac{1}{1-c})^{k_d}$ increases, and so $Pr(S_1 \sim S_2|\Phi)$ decreases.

□

Note that Equation (9) does not indicate direction of dependence. Indeed, without considering accuracy of individual sources, we do not have evidence for deciding direction in presence of benevolent copiers. Section 4 and 5 describe two models that detect directions of dependencies for malicious copiers and sources of different accuracy. Also note that Equation (9) computes probability of dependence based on shared values between sources. In case of transitive copying or co-copying, it can compute (incorrectly) a high probability of dependence (*i.e.*, direct copying) between the transitive copier and the original source, or between copiers who copy from the same source. However, as we discuss shortly and as our experiments show, this type of mis-classification does not affect vote count much and so seldom change truth discovery results.

3.2 Vote count of a value

We have described how we decide if a pair of data sources are dependent. However, even if a data source copies from another, it is possible that it provides some of the values independently and it would be inappropriate to ignore these values. We next describe how to count the vote for a particular value once we know the probability of dependence. We first analyze what the ideal vote count should be and then describe an algorithm that estimates the ideal count.

3.2.1 Ideal vote count

We start from the case where we know deterministically the dependence relationship between sources and discuss probabilistic dependence subsequently. Consider a specific value v for a particular object O and let $\bar{S}_o(v)$ be the set of data sources that provide v on O. Suppose for each pair of sources in $\bar{S}_o(v)$, we know if one copies from the other and which is the copier. Accordingly, we can draw a *dependence graph* G, where for each $S \in \bar{S}_o(v)$, there is a node, and for each $S_1, S_2 \in \bar{S}_o(v)$ where S_1 copies from S_2, there is an edge from S_1 to S_2.

For each $S \in \bar{S}_o(v)$, we denote by $d(S, G)$ the out-degree of S in G, corresponding to the number of data sources from

which S copies. If $d(S,G) = 0$, S is independent and its vote count for v is 1. Otherwise, according to the *random-copying* assumption, for each source S' that S copies from, S provides a value independently of S' with probability $1-c$. According to the *Independent-copying* assumption, the probability that S provides v independently of any other source is $(1-c)^{d(S,G)}$ and the total vote count of v with respect to G is

$$V(v,G) = \sum_{S \in \bar{S}_o(v)} (1-c)^{d(S,G)}. \quad (10)$$

However, recall that Equation (9) computes only a probability of dependence between a pair of sources and does not indicate the direction of the dependence. Thus, we have to enumerate all possible dependence graphs and take the sum of the vote count with respect to each of them, weighted by the probability of the graph. Let $\bar{D}_o$ be the set of possible dependencies between sources in $\bar{S}_o(v)$ and we denote the probability of $D \in \bar{D}_o$ by $p(D)$. Consider a subset $\bar{D} \subseteq \bar{D}_o$ of m dependencies. According to the *Independent-copying* assumption, the probability that all and only dependencies in $\bar{D}$ hold is

$$Pr(\bar{D}) = \Pi_{D \in \bar{D}} p(D) \Pi_{D \in \bar{D}_o - \bar{D}} (1 - p(D)). \quad (11)$$

There are up to 2^m acyclic dependence graphs with this set of dependencies and we denote them by $\bar{G}(\bar{D})$. Our a-priori belief would be that all dependence graphs in $\bar{G}(\bar{D})$ have the same probability; however, by analyzing the probability of multiple sources independently providing the same value, we can obtain more accurate probability of each dependence graph.

Consider a dependence graph $G \in \bar{G}(\bar{D})$. Let k be the number of independent data sources in G. We first compute given G, what is the probability that all data sources in $\bar{S}_o(v)$ provide the same value v for a particular object, denoted by $Pr(v|G)$. This probability is the probability that all independent sources in G provide value v. By applying the same analysis as in the last section, we obtain that the probability is $(1-\varepsilon)^k$ if v is true and $\frac{\varepsilon^k}{n^{k-1}}$ otherwise. Thus, the conditional probability of v being provided by all sources in $\bar{S}_o(v)$ is

$$Pr(v|G) = (1-\varepsilon)^{k+1} + \frac{\varepsilon^{k+1}}{n^{k-1}}. \quad (12)$$

As we believe a-priori that all dependence graphs in $\bar{G}(\bar{D})$ have the same probability, by applying the Bayes Rule, we compute the probability of G as

$$Pr(G|v,\bar{D}) = \frac{P(v|G,\bar{D})}{\sum_{G_0 \in \bar{G}(\bar{D})} P(v|G_0,\bar{D})} = \frac{P(v|G)}{\sum_{G_0 \in \bar{G}(\bar{D})} P(v|G_0)}. \quad (13)$$

According to Equations (12) and (13), G has the highest probability when it has the minimum number of independent sources; that is, when each connected subgraph in G contains a single independent source. When the number of independent sources is increased, the probability of G is reduced dramatically.

Based on Equations (11) and (13) we can compute the probability of G as

$$Pr(G|v) = Pr(G|v,\bar{D}) Pr(\bar{D}). \quad (14)$$

Finally, the vote count of v should be the sum of the vote count with respect to each dependence graph weighted by the probability of that dependence graph:

$$V(v) = \sum_G V(v,G) Pr(G|v). \quad (15)$$

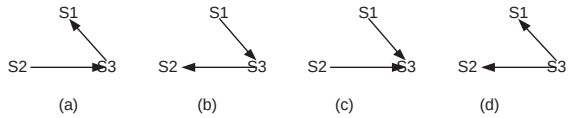

Figure 2: Dependence graphs with a dependence between S_1 and S_3 and one between S_2 and S_3, where S_1, S_2, and S_3 provide the same value on an object.

EXAMPLE 3.2. *Consider three data sources S_1, S_2 and S_3 that provide the same value v on an object. Assume $c = .8$ and between each pair of sources the probability of dependence is .4. We can compute v's vote count by enumerating all possible dependence graphs.*

- *There is 1 graph with no dependence. All sources are independent so the vote count is $1+1+1 = 3$. The probability of this graph is $(1-.4)^3 = .216$.*
- *There are 6 graphs with only one dependence. The total probability of graphs that contain a particular dependence is $(1-.4)^2 * .4 = .144$. Each dependence has two directions, so the probability of each such graph is $.144/2 = .072$. No matter which direction the dependence is in, the vote count is $1+1+.2 = 2.2$.*
- *There are 12 graphs with two dependencies. Figure 2 shows the four that contain a dependence between S_1 and S_3, and a dependence between S_2 and S_3. The sum of their probabilities is $(1-.4) * .4^2 = .096$. For each of the first three graphs (Figure 2(a)-(c), each with a single independent source), the vote count is $1+.2+.2 = 1.4$ and by applying the Bayes Rule we compute its probability as $.32 * .096 = .03$. For the last one (Figure 2(d), with two independent sources), the vote count is $1+1+.2^2 = 2.04$ and its probability is $.04 * .096 = .004$.*
- *Finally, there are 6 acyclic graphs with three dependencies (we ignore the details), where each has vote count $1+.2+.2^2 = 1.24$ and probability $.4^3/6 = .011$.*

By taking the weighted sum, we compute the total vote count of v as 2.08. □

We note that if we have a-priori knowledge of correlations between copyings, such as one copier copying from a single source, we can extend our model by applying such knowledge to prune some dependence graphs, normalizing the probabilities of the rest of the graphs, and then taking the weighted sum as the vote count.

3.2.2 Estimating vote count

As there are an exponential number of dependence graphs, computing the vote count by enumerating all of them can be quite expensive. Let d be the number of possible dependencies. For each dependence, we need to consider three possibilities: the dependence does not hold, it holds in one direction, and it holds in the other direction. Thus, there are 3^d dependence graphs and so computing the vote count takes exponential time. To make the analysis of source dependence scalable, we need to find a way to estimate the vote count in polynomial time.

We estimate a vote count by considering the data sources one by one. For each source S, we denote by $\overline{Pre}(S)$ the set of sources that have already been considered and by $\overline{Post}(S)$ the set of sources that have not been considered yet. We compute the probability that the value provided by S is independent of any source in $\overline{Pre}(S)$ and take it as the vote count of S. The vote count computed in this way is not precise because if S depends only on sources in $\overline{Post}(S)$ but some of those sources depend on sources in $\overline{Pre}(S)$, our estimation still (incorrectly) counts S's vote. To minimize such error, we wish that the probability that S depends on a source $S' \in \overline{Post}(S)$ and S' depends on a source $S'' \in \overline{Pre}(S)$ be the lowest. Thus, we take a greedy algorithm and consider data sources in such an order: in the first round, we select a data source that is associated with a dependence of the highest probability; in later rounds, each time we select a data source that has the maximal dependence on one of the previously selected sources.

We now consider how to compute the vote count of v once we have decided an order of the data sources. Let S be a data source that votes for v and we denote by $P(S \sim S_0)$ the probability of dependence between sources S and S_0. The probability that S provides v independently of any data source in $\overline{Pre}(S)$ is

$$V(S, \overline{Pre}(S)) = \Pi_{S_0 \in \overline{Pre}(S)}(1 - cP(S \sim S_0)). \quad (16)$$

Let $I(S) = V(S, \overline{Pre}(S))$. The total vote count of v is $\sum_{S \in \bar{S}_o(v)} I(S)$.

EXAMPLE 3.3. *Continue with Example 3.2. As all dependencies have the same probability, we can consider the data sources in any order. We choose the order of S_1, S_2, S_3. The vote count of S_1 is 1, that of S_2 is $1 - .4 * .8 = .68$, and that of S_3 is $.68^2 = .46$. So the estimated vote count is $1 + .68 + .46 = 2.14$, very close to the real vote count, 2.08.* □

Finally, as a further optimization, we would like to consider a dependence only if it has the potential to change vote counts substantially. If the probability of a dependence is lower than a threshold η, we ignore the dependence and view the two data sources as independent. Our experimental results show that setting η to a low number, such as 0.1, typically does not change the voting results, but can improve the efficiency by ???.

Algorithm BENEVOTECOUNT describes how we count votes for each value provided on a particular object O. In the algorithm we denote by $S(O)$ the value S provides on object O. We formalize the properties of BENEVOTECOUNT as follows, showing scalability of our estimation algorithm.

THEOREM 3.4. BENEVOTECOUNT *has the following two properties.*

1. *Let t_0 be the ideal vote count of a value and t be the vote count computed by* BENEVOTECOUNT. *Then, $t_0 \leq t \leq 1.5t_0$.*
2. *Let s be the number of sources that provide information on an object. We can estimate the vote count of all values of this object in time $O(s^2 \log s)$.* □

PROOF. 1. Consider m data sources that vote for a value and assume BENEVOTECOUNT ranks them as $S_1, \ldots, S_m$. Let $\bar{D}$ be a subset of dependencies. Let G be a dependence graph with dependencies in $\bar{D}$. If G contains nodes $S_i, S_j, S_k, 1 \leq i < j < k \leq m$, where S_j depends on S_k and S_k depends on S_i, our estimation will (incorrectly) count the vote by S_j; otherwise, our estimation computes the correct vote count for G. For any three nodes in G, the probability that the previously described case happens is at most $\frac{1}{3}$ (by Bayes analysis). So the probability that BENEVOTECOUNT estimates more vote count for G than the ideal vote count is at most $\frac{1}{3}$. Thus, the total vote count estimated by BENEVOTECOUNT is at most $\frac{1/3}{1-1/3} = .5$ more than the ideal vote count.

2. Let d be the number of dependencies between the sources; $d \leq \frac{s(s-1)}{2}$. Step 1 scans all dependencies in $\bar{D}_o$ and takes time $O(d)$. Step 2 at most adds all dependencies in $\bar{D}_o$ into the heap. Inserting a dependence into the heap and finding the dependence with the maximal probability in the heap both take time $O(\log d)$, so Step 2 takes time $O(d \log d)$. Step 3 computes vote count for each source and takes time $O(s^2)$. In total, BENEVOTECOUNT takes time $O(d \log d + s^2) = O(s^2 \log s)$.

□

0: **Input**: $O, \bar{S}_o, \bar{D}_o$. *//O is an object, $\bar{S}_o$ is the set of sources providing values on O, and $\bar{D}_o$ is the set of dependencies between sources in $\bar{S}_o$.*
Output: The vote count of each value on O.
1: *//Find dependence between sources that vote the same value*
for each (dependence $d = (S_1, S_2) \in \bar{D}_o$)
if ($Pr(d) > \eta$ && $S_1(O) = S_2(O)$)
Add d to $\bar{D}$; *// η is the threshold for a valid dependence*
2: *//Sort sources in $\bar{S}_o$ to an ordered list $\bar{S}$:*
$Heap = \emptyset$; //A binary heap for sorting
while ($\bar{D} \neq \emptyset$)
Let $d = (S_1, S_2)$ be the depen in $\bar{D}$ w. max pr;
Remove d from $\bar{D}$; Add S_1, S_2 to $\bar{S}$;
Move all depens in $\bar{D}$ related to S_1 or S_2 to $Heap$;
while ($Heap \neq \emptyset$)
Let $d_0 = (S_1, S_2)$ be the depen in $Heap$ w. max pr;
if ($S_1 \in \bar{S}$ && $S_2 \in \bar{S}$)
Remove d from $Heap$;
else if (S_1 (or S_2) $\in \bar{S}$)
Add S_2 (or S_1) to $\bar{S}$;
Remove d from $Heap$;
Move all depens in $\bar{D}$ w. S_2 (or S_1) to $Heap$;
if ($|\bar{S}| < |\bar{S}_o|$) Add $\bar{S}_o - \bar{S}$ to the end of $\bar{S}$;
3: *//Compute the vote count:*
for each ($S \in \bar{S}$)
Compute S's vote count according to Equation (16);
Add S's vote count to the vote count of $S(O)$;

Algorithm 1: BENEVOTECOUNT: Compute vote count for values on a particular object using the benevolent model.

3.3 Finding the true values

Once we know the vote count of each value, we can decide the true values by voting. However, computing vote counts requires knowing probabilities of dependencies between data sources, whereas computing the probabilities of dependencies requires knowing the true values. There is a recursive dependence between them and we solve the problem by computing them iteratively.

Algorithm VOTE describes how to discover true values from conflicting information provided by multiple data sources.

0: **Input**: $\mathcal{S}, \mathcal{O}$.
Output: The true value for each object in $\mathcal{O}$.
1: $\mathcal{D} = \emptyset$; *//dependencies between sources*
$\bar{V} = \emptyset$; *//decided true values*
$\bar{V}_0 = null$; *//true values decided in the last round*
2: **while** ($\bar{V} \neq \bar{V}_0$)
3: $\bar{V}_0 = \bar{V}$; $\bar{V} = \emptyset$;
4: **for each** ($S_1, S_2 \in \mathcal{S}, S_1 \neq S_2$)
Add depen between S_1 and S_2 to $\mathcal{D}$ (if not yet);
Compute $Pr(S_1 \sim S_2)$ according to (19) in 1st round and to (9) later;
5: **for each** ($O \in \mathcal{O}$)
Find the set $\bar{S}_o \subseteq \mathcal{S}$ of sources that vote for O and the set $\bar{D}_o \subseteq \mathcal{D}$ of dependencies between sources in $\bar{S}_o$;
BeneVoteCount($O, \bar{S}_o, \bar{D}_o$);
Select the value with the maximal vote count and add to $\bar{V}$ (if there are several values winning the same number of votes, choose the previously selected one if possible and randomly choose one otherwise);

Algorithm 2: Vote: Discover true values.

Vote iteratively computes the probability of dependence between each pair of data sources and the vote count of each value, and then for each object takes the value with the maximal vote count as the true value. This process repeats until the voting results converge.

Note that it is critical to consider the dependence between sources from the beginning; otherwise, a data source that has been duplicated many times can dominate the vote results in the first round and make it hard to detect the dependence between it and its copiers (as they share only "true" values). However, in the first round we do not know which values are true and which are false; in other words, we cannot distinguish $\bar{O}_t$ and $\bar{O}_f$. We instead consider $\bar{O}_c = \bar{O}_t \cup \bar{O}_f$ and denote by k_c the size of $\bar{O}_c$. By applying similar analysis, we obtain the probability that two independent sources provide the same value as

$$Pr(o \in \bar{O}_c | S_1 \perp S_2) = (1-\varepsilon)^3 + \frac{\varepsilon^3}{n}. \tag{17}$$

Let $P_c = (1-\varepsilon)^3 + \frac{\varepsilon^3}{n}$. The probability of two dependent sources providing the same value is

$$Pr(o \in \bar{O}_c | S_1 \sim S_2) = c + P_c \cdot (1-c). \tag{18}$$

By applying the Bayes Rule, we derive from Equations (3)(7)(17)(18) the probability of two data sources being dependent conditioned on the observation of Φ as follows:

$$Pr(S_1 \sim S_2 | \Phi) = \left(1 + (\frac{1-\alpha}{\alpha})(\frac{P_c}{c + P_c - cP_c})^{k_c}(\frac{1}{1-c})^{k_d}\right)^{-1}. \tag{19}$$

We can prove that when there are a finite number of objects in $\mathcal{O}$, Algorithm Vote cannot change the decision for an object O back and forth between two different values forever; thus, the algorithm converges. In practice, our experiments show that the algorithm typically converges in only a few rounds.

Theorem 3.5 (Convergence of Vote). *Let $\mathcal{S}$ be a set of good independent sources and benevolent copiers that provide information on objects in $\mathcal{O}$. Let l be the number of objects in $\mathcal{O}$ and n_0 be the maximum number of values provided for an object by $\mathcal{S}$. The* Vote *algorithm converges in at most $2ln_0$ rounds on $\mathcal{S}$ and $\mathcal{O}$.* □

Proof. Assume in contrast, there exists $\mathcal{S}$ and $\mathcal{O}$ on which Vote does not converge. Then there must be at least one object $O \in \mathcal{O}$ on which the true value decided by Vote is changed back and forth between v and v'. Let $\bar{S}$ be the set of sources that provide v and $\bar{S}'$ be the set of sources that provide v'.

Suppose at the k_1-th round, v is decided to be the true value on O and at the k_2-th ($k_2 > k_1$) round, v' is decided to be the true value on O. Because of this decision, the probability of dependence between sources in $\bar{S}$ is increased and the vote count is decreased. Let Δ_1 ($\Delta_1 < 0$) be the change of the total vote count of $\bar{S}$. On the other hand, the probability of dependence between sources in $\bar{S}'$ is decreased and the vote count is increased. Let Δ_1' ($\Delta_1' > 0$) be the change of the total vote count of $\bar{S}'$.

Suppose at the k_3-th round ($k_3 > k_2$) the true value on O is changed back to v. Then, between the k_2-th round and the (k_3-1)-th round, there must exist a set $\bar{O}_1$ of objects on which the decision of the true values was changed. Let Δ_2 be the total change of vote count of sources in $\bar{S}$ resulted from $\bar{O}_1$ and Δ_2' be this change for sources in $\bar{S}'$. If $\bar{O}_1$ contains a single value, $\Delta_2' - \Delta_2$ is maximum when all sources in S_1 provide the newly selected value and all sources in S_2 provide the originally selected value. Thus, we must have $\Delta_2 \leq -\Delta_1$ and $-\Delta_2' \leq \Delta_2'$. Hence, $\Delta_1 + \Delta_2 \leq 0 \leq \Delta_1' + \Delta_2'$ and $\bar{S}$ still cannot win the vote back on O. So there must be at least two values in $\bar{O}_1$ to make $\Delta_1 + \Delta_2 > \Delta_1' + \Delta_2'$. After this round, because the decision on the true value of O changes, the vote count of S_1 and S_2 are further changed. Let $\Delta_3 > 0$ be the change for $\bar{S}$ and $\Delta_3' < 0$ be the change for $\bar{S}'$ resulted from O.

Now consider the k_4-th round ($k_4 > k_3$) in which the true value on O is changed again to v'. Between the k_3-th round and the $(k_4) - 1$ round, there must exist a set $\bar{O}_2$ of objects on which the decision of the true values was changed. Let Δ_4 be the total change of vote count of sources in $\bar{S}$ and Δ_4' be this change for sources in $\bar{S}'$ for this time. If $\bar{O}_2$ contains two values, then following the same analysis, we must have $\Delta_4 \geq -\Delta_2 - \Delta_3$ and $-\Delta_4' \geq \Delta_2' + \Delta_3'$. Hence, $\Delta_2 + \Delta_3 + \Delta_4 \geq 0 \geq \Delta_2' + \Delta_3' + \Delta_4'$ and $\bar{S}'$ still cannot win the vote on O. So there must be at least three values in $\bar{O}_2$.

For the same reason, next time when the decision on the true value of O is changed to v, there must exist at least 4 other objects on which the decided true values are changed; and then the next time 5 objects are required to have decision change for the decision on the true value of O to be changed to v', and so on. As there are only a finite number of objects in $\mathcal{O}$, this process cannot continue forever, contradicting our assumption that Vote changes back and forth between v and v'.

We now consider the maximum number of rounds required for convergence. Let O be the object with the maximum number of values being provided by the sources. The decision of its true values should converge in at most $(l-1)(n_0-1)+1$ rounds. This is because for each two values, there are at most $l-1$ rounds that the decided true value for O switch between the two values; after the $l-1$ rounds, one of them is eliminated and can never be decided to be the true value again. To eliminate $n_0 - 1$ values, we thus need at most $(l-1)(n_0-1)$ rounds.

During these rounds, the total number of changes of de-

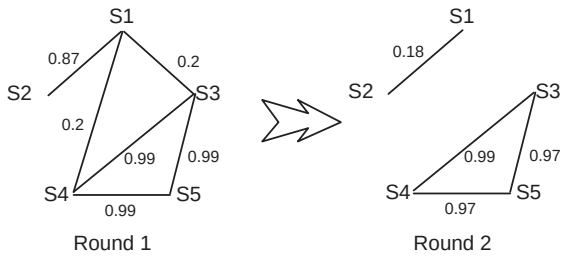

Figure 3: Probabilities of dependencies computed by BENE on the motivating example. We only show dependencies with a probability over .1.

Table 3: Vote counts of affiliations for *Carey* and *Halevy* in the motivating example.

	Carey			*Halevy*	
	UCI	Oracle	BEA	Google	UW
Round 1	1	1	1.24	1.3	1.24
Round 2	1	1	1.25	1.85	1.25

cided true values for other objects are at least

$$(n_0 - 1)\sum_{i=1}^{l-1} i = \frac{l(l-1)(n_0-1)}{2}.$$

The average number of changes for each object is thus $\frac{l(n_0-1)}{2}$. Thus, there must be an object, O', for which at least $\frac{l(n_0-1)}{2l} = \frac{n_0-1}{2}$ values are already eliminated. The object O' has at most n_0 values and so we need to eliminate another $\frac{n_0-1}{2}$ values. As the value of O cannot change any more, eliminating one value of O' takes at most $l-2$ rounds, and so in total $\frac{(l-2)(n_0-1)}{2}$ rounds. We can compute the number of remaining rounds for convergence of the true fact for the rest of the objects in a similar way. So the total rounds is

$$1 + (l-1)(n_0-1) + (l-2)\frac{n_0-1}{2} + (l-3)\frac{n_0-1}{4} + \ldots$$
$$< 1 + 2(l-1)(n_0-1) < 2ln_0$$

□

We next illustrate the algorithm on the motivating example.

EXAMPLE 3.6. *We run Algorithm* VOTE *on data sources in Example 1.1. Figure 3 shows the probabilities of dependencies computed in each round and Table 3 shows the vote count of affiliations for* Carey *and* Halevy.

Initially, we do not know which values are true and which are false and so apply Equation (19). We compute the probability of dependence between S_1 and S_2 (sharing three values) as .87 and those between S_3, S_4, S_5 (sharing four or five values) as .99. Accordingly we decide that the affiliations for the five researchers are MIT, MSR, MSR, BEA, Google respectively.

In the second round, we refine the dependence probabilities according to the selected true values. The probability between S_1 and S_2 (sharing only true values) is reduced to .18 and those between S_3, S_4, S_5 (sharing two or three false values) remain high; thus, the refined probabilities more closely reflect the reality. Though the new probabililities change the vote counts, they do not further change our voting results. The voting converges in this round and finds correct affiliations for four out of five researchers.

As we show later, if we consider accuracy of data sources, the new vote counts shall increase our belief of S_1's accuracy and can eventually lead to the correct decision for Carey's affiliation. □

Setting parameters: Finally, we discuss how we set parameters in Equations (9) and (16) and refine them during voting. Initially, we set them according to our a-priori knowledge of the probabilities of dependence, errors, etc. During the voting process, in each iteration we can refine α, ε and c based on the computed dependence probabilities and the decided true values, and use the new parameters in the next iteration. Our experimental results show that different initial parameter settings lead to similar voting results (Section 7), providing evidence of robustness.

We note that the probabilities that a data source makes mistakes on different objects and the probabilities that a value provided by different copiers is copied can be different. Equation (9) uses the average probability. We can also extend it to use different probabilities. We next show the extension for different error rates as an example.

Let $\varepsilon(O)$ be the probability of providing a false value on object o. Then, we can refine Equation (9) as follows:

$$\begin{aligned} & Pr(S_1 \sim S_2|\Phi) \\ = \; & (1 + (\frac{1-\alpha}{\alpha})(\frac{1}{1-c})^{k_d}\Pi_{O\in\bar{O}_t}\frac{1-\varepsilon(O)}{1-(1-c)\varepsilon(O)} \\ & \Pi_{O\in\bar{O}_f}\frac{\varepsilon(O)}{\varepsilon(O)+cn-c\varepsilon(O)})^{-1} \end{aligned} \quad (20)$$

In Section 5 we consider the case where the error rate is different for different sources.

4. MODEL FOR MALICIOUS COPIERS

This section describes how we discover true values in presence of malicious copiers.

4.1 Dependence of data sources

We now consider a set $\mathcal{S}$ of good independent sources and malicious copiers. Assume when a malicious copier provides a value independently, among the n values that are not provided by the original source on the same object, it chooses a particular one with probability $\frac{1}{n}$.

Consider two data sources S_1 and S_2. Intuitively, it is more likely that a malicious copier changes a true value into some false one than the opposite. Thus, if the true values provided by S_2 is a subset of those provided by S_1, it is likely that S_2 copies from S_1; on the other hand, if the true values provided by S_1 and those provided by S_2 merely overlap, it is less likely that S_1 or S_2 is a malicious copier.

To incorporate this intuition, we further divide $\bar{O}_d$, the set of objects on which S_1 and S_2 provide different values, into three sets: $\bar{O}_{d1}$ contains objects on which S_1 provides true values but S_2 provides false values, $\bar{O}_{d2}$ contains objects on which S_2 provides true values but S_1 provides false values, and $\bar{O}_{d0}$ contains objects on which S_1 and S_2 provide different false values. As there is a single true value, it is not possible for S_1 and S_2 to provide different true values. In addition, we consider different directions of dependence,

denoting S_1 depending on S_2 by $S_1 \rightarrow S_2$ and S_2 depending on S_1 by $S_2 \rightarrow S_1$.

We now compute the probability of Φ given a particular dependence relationship between S_1 and S_2. If S_1 and S_2 are independent, with the same analysis as in Section 3, we compute the probability of a particular category as follows:

$$Pr(O \in \bar{O}_t | S_1 \perp S_2) = (1-\varepsilon)^2 \quad (21)$$

$$Pr(O \in \bar{O}_f | S_1 \perp S_2) = \frac{\varepsilon^2}{n} \quad (22)$$

$$Pr(O \in \bar{O}_{d1} | S_1 \perp S_2) = \varepsilon(1-\varepsilon) \quad (23)$$

$$Pr(O \in \bar{O}_{d2} | S_1 \perp S_2) = \varepsilon(1-\varepsilon) \quad (24)$$

$$Pr(O \in \bar{O}_{d0} | S_1 \perp S_2) = \varepsilon^2 \cdot \frac{n-1}{n} \quad (25)$$

In the last equation, ε^2 is the probability that both S_1 and S_2 provide false values and $\frac{n-1}{n}$ is the probability that S_2 provides a different false value from S_1.

If S_2 copies from S_1, the probability of a particular category can be computed as follows.

$$Pr(O \in \bar{O}_t | S_2 \rightarrow S_1) = (1-\varepsilon) \cdot c \quad (26)$$

$$Pr(O \in \bar{O}_f | S_2 \rightarrow S_1) = \varepsilon \cdot c \quad (27)$$

$$Pr(O \in \bar{O}_{d1} | S_2 \rightarrow S_1) = (1-\varepsilon)(1-c) \quad (28)$$

$$Pr(O \in \bar{O}_{d2} | S_2 \rightarrow S_1) = \varepsilon \cdot (1-c) \cdot \frac{1}{n} \quad (29)$$

$$Pr(O \in \bar{O}_{d0} | S_2 \rightarrow S_1) = \varepsilon \cdot (1-c) \cdot \frac{n-1}{n} \quad (30)$$

Equations (26)(27) consider the case that S_2 copies the value from S_1 (with probability c), where the value is true with probability $1-\varepsilon$ and false with probability ε. Equations (28)(29)(30) consider the case that S_2 provides the value independently (with probability $1-c$). If S_1 provides a true value, then S_2 must provide a false one. If S_1 provides a false value, then S_2 provides a true value with probability $\frac{1}{n}$ and provides a different false value with probability $\frac{n-1}{n}$.

When S_1 copies from S_2, we have the same equations except for $O \in \bar{O}_{d1}$ and $O \in \bar{O}_{d2}$:

$$Pr(O \in \bar{O}_t | S_2 \rightarrow S_1) = (1-\varepsilon) \cdot c \quad (31)$$

$$Pr(O \in \bar{O}_f | S_2 \rightarrow S_1) = \varepsilon \cdot c \quad (32)$$

$$Pr(O \in \bar{O}_{d1} | S_1 \rightarrow S_2) = \varepsilon \cdot (1-c) \cdot \frac{1}{n} \quad (33)$$

$$Pr(O \in \bar{O}_{d2} | S_1 \rightarrow S_2) = (1-\varepsilon) \cdot (1-c) \quad (34)$$

$$Pr(O \in \bar{O}_{d0} | S_2 \rightarrow S_1) = \varepsilon \cdot (1-c) \cdot \frac{n-1}{n} \quad (35)$$

Thus, we compute different probabilities for different directions of dependence.

Assume a-priori we believe the probability of dependence of each direction is α, we can apply the Bayes Rule to compute the probabilities of $S_1 \perp S_2$, $S_1 \rightarrow S_2$ and $S_2 \rightarrow S_1$, conditioned on the observation of Φ. Note that the distribution of $\bar{O}_{d1}, \bar{O}_{d2}$ and $\bar{O}_{d0}$ can dramatically change the probability of dependence, so the properties in Theorem 3.1 do not hold for the malicious model.

4.2 Vote count of a value

Ideally, the vote count of a value is the sum of the vote counts with respect to each dependence graph weighted by the probability of that graph. Since we now know the probability of each direction of dependence, we can compute the probability of a dependence graph by taking the product of the probabilities of the corresponding dependence relationship between each pair of sources. As a further step, for all dependence graphs that contain dependence between the same pairs of sources, we take the computed probabilities as the a-priori probabilities and apply Bayes analysis to adjust the probability of each graph as we describe in Section 3.2.1.

We next consider what is the vote count of each graph. Recall that in the benevolent model, even if a data source is a copier, with probability $1-c$ it provides a value independently, so we need to count its vote to a certain fraction. When a malicious copier changes a value, it only randomly chooses another value so its vote count should be ignored. Thus, the vote count with respect to a graph G is the number of independent sources in G.

Again, computing the vote count by enumerating all possible dependence graphs takes exponential time and we need a quick estimation. The new challenge we face is to consider different directions of dependencies. Our algorithm sorts the sources as follows. If between a pair of sources S_1 and S_2, the probability of $S_1 \rightarrow S_2$ is much higher than that of $S_2 \rightarrow S_1$, we consider S_1 as a copier and order S_2 before S_1; otherwise, we consider both directions as equally possible and sort the sources following the same rule as for the benevolent model. The details is described in Algorithm MalVoteCount.

Finally, we can still use Algorithm Vote to find true values except that we apply the malicious model. Note that under the malicious model Vote may not converge. When we select different values as the true values, the direction of the dependence between two sources can change and accordingly we may consider different data sources as copiers and neglect their votes; thus, the vote counts we compute may change dramatically and suggest different true values. We stop after a certain number of rounds and use the results of that round as the final results. We observe in our experiments (see Section 7) that even when Vote does not converge, the selected true values vary for only a few objects, so in which round we stop does not change the number of correctly decided true values much.

4.3 Comparison of the two models

Both the benevolent model and the malicious model have their own advantages and disadvantages. As both benevolent copiers and malicious copiers tend to share many false values with the original source, the benevolent model can detect both all of them; however, it computes a lower dependence probability for malicious copiers than the malicious model and cannot decide the direction of dependence. Even when it discovers a malicious copier, it assumes the copier is benevolent and considers it in vote counting.

On the other hand, the malicious model is good at detecting malicious sources, but can miss some of the benevolent copiers if they also change false values to true ones. Even if it discovers benevolent copiers, it neglects all of its provided values, though some of them are indeed independent. In addition, it can ring false alarm when there are highly accurate sources, in which case it considers less accurate sources as copiers and neglects their votes.

As our experiments show (see Section 7), the benevolent model and the malicious model are suitable for different types of data sources. If we know beforehand the types of sources, we can choose the more appropriate model. We can also combine these two models by assuming there are

0: **Input**: $o, \bar{S}_0, \bar{D}_0$. *//o is an object, $\bar{S}_0$ is the set of sources providing values on o, and $\bar{D}_0$ is the set of dependencies between sources in $\bar{S}_0$.*
Output: The vote count of each value on o.
1: *//Generate the set $\bar{D}$ of relevant dependencies and record dependence direction:*
for each $(S \in \bar{S}_0)$
$Dep(S) = \emptyset$; //sources that S depends on
for each (dependence $d = (S_1, S_2) \in \bar{D}_0$)
if $(Pr(S_1 \to S_2) + Pr(S_2 \to S_1) > \eta$ && $S_1(O) = S_2(O))$
Add d to $\bar{D}$;
if $Pr(S_1 \to S_2) - Pr(S_2 \to S_1) > \theta)$ (resp. opposite)
Insert S_2 (resp. S_1) to $Dep(S_1)$;
Sort dependencies in $\bar{D}$ in descendant order of probabilities;
2: *//Sort sources in $\bar{S}_0$ to an ordered list $\bar{S}$:*
$\bar{S} = \emptyset$; $\bar{V} = \emptyset$; *//set of values we have handled.*
while $(|\bar{S}| \neq |\bar{S}_0|)$
$\bar{R} = \emptyset$; *//set of sources with zero original sources*
for each $(S \in (\bar{S}_0 - \bar{S}))$
if $(Dep(S) == \emptyset)$
Add S to $\bar{R}$;
for each $(S_0 \in (\bar{S}_0 - \bar{S})$ && $S_0 \neq S)$
Remove S from $Dep(S_0)$;
if $(\bar{R} == \emptyset)$
Choose S whose $Dep(S)$ has minimal size, add S to $\bar{R}$ and remove it from other sources' original source set
$\bar{D}' = \emptyset$;
while $(\bar{R} \neq \emptyset$ && $\bar{D} \neq \bar{D}')$
for each $(d = (S_1, S_2) \in \bar{D})$
if $(S_1(O) \notin \bar{V}$ && $S_1 \in \bar{R}$ && $S_2 \in \bar{R})$
Add S_1 and S_2 to $\bar{S}$; Add $S_1(O)$ to $\bar{V}$;
Remove d from $\bar{D}$; Remove S_1 and S_2 from $\bar{R}$;
break;
else if $(S_1 \in \bar{S}$ && $S_2 \in \bar{S})$
Remove d from $\bar{D}$; **break**;
else if (S_1 (or S_2) $\in \bar{S}$)
Add S_2 (or S_1) to $\bar{S}$;
Remove S_2 (or S_1) from $\bar{R}$; Remove d from $\bar{D}$;
break;
if $(|\bar{S}| < |\bar{S}_0|)$
Add $\bar{S}_0 - \bar{S}$ to the end of $\bar{S}$;
3: *//Compute the vote count:*
for each $(S \in \bar{S})$
$Pr(S) = 1$; //probability of S being independent
for each (S_0 before S in $\bar{S}$)
$Pr(S)$*=$(1 - Pr(S_0 \to S) - Pr(S \to S_0))$;
Add $Pr(S)$ to the vote count of $S(O)$.

Algorithm 3: MALVOTECOUNT: Compute vote count for values on a particular object using the malicious model.

both malicious copiers and benevolent copiers, and for each pair of data sources S_1 and S_2, we compute the probabilities of S_1 and S_2 being independent, one of S_1 and S_2 being a benevolent copier, S_1 being a malicious copier, and S_2 being a malicious copier. Another option is to assume a copier can change some of the copied values in a benevolent manner and some of the copied values in a malicious manner (or make mistakes in copying), and assign different change rates to different change manners. In the next section we describe another model, which takes into consideration the accuracy of sources and in general performs better than both the benevolent model and the malicious model.

5. CONSIDERING ACCURACY OF SOURCES

Data sources are often of different accuracy: some data sources make few mistakes whereas others make a lot. When we discover true values, we are more willing to trust values provided by the former sources. This section describes the ACCU model, which considers *accuracy* of data sources. We first discuss how the accuracy of sources can affect our belief of dependence between sources, and then describe how we compute accuracy and take it into consideration when we count votes.

Our ACCU model indeed computes a probabilistic distribution of various values in the underlying domain for a particular object. We can either choose the value with the highest probability as the true value, or store all possible values with their probabilities using a probabilitic database.

5.1 Dependence w.r.t. accuracy of sources

Intuitively, if between two data sources S_1 and S_2, the accuracy of the common values is closer to the overall accuracy of S_1, then it is more likely that S_2 copies from S_1. We incorporate this intuition by considering accuracy of sources when we compute the probability of dependencies. Our model can discover not only benevolent copiers, but also malicious ones, as malicious copiers can have very different accuracy on values they copy and values they provide independently.

Let S be a data source. We denote by $A(S)$ the *accuracy* of S and by $\varepsilon(S)$ the *error rate* of S; $\varepsilon(S) = 1 - A(S)$. We describe how to compute $A(S)$ shortly. Note that accuracy of a source is orthogonal to whether a source is good or bad, as the former considers the probability of a source providing true values, while the latter compares the probability of providing a true value and a particular false value for an object.

Consider two data sources S_1 and S_2. We conduct the same analysis on the probability of S_1 and S_2 providing the same or different values as in Section 3, except that everywhere we used to consider the error rate of a value, we now consider the error rates of the sources. A similar analysis as in Section 3 leads to the following sets of equations. When S_1 and S_2 are independent, we have

$$Pr(O \in \bar{O}_t | S_1 \perp S_2) = (1 - \varepsilon(S_1))(1 - \varepsilon(S_2)) = P_t, \quad (36)$$

$$Pr(O \in \bar{O}_f | S_1 \perp S_2) = \frac{\varepsilon(S_1)\varepsilon(S_2)}{n} = P_f, \quad (37)$$

$$Pr(O \in \bar{O}_d | S_1 \perp S_2) = 1 - P_t - P_f. \quad (38)$$

When S_2 copies from S_1, we have

$$Pr(O \in \bar{O}_t | S_2 \to S_1) = (1 - \varepsilon(S_1)) \cdot c + P_t \cdot (1 - c) \quad (39)$$

$$Pr(O \in \bar{O}_f | S_2 \to S_1) = \varepsilon(S_1) \cdot c + P_f \cdot (1 - c), \quad (40)$$

$$Pr(O \in \bar{O}_d | S_2 \to S_1) = (1 - P_t - P_f) \cdot (1 - c). \quad (41)$$

When S_1 copies from S_2, we obtain the same set of equations as in case of S_2 copying from S_1 except that $\varepsilon(S_1)$ is replaced with $\varepsilon(S_2)$ everywhere.

$$Pr(O \in \bar{O}_t | S_1 \to S_2) = (1 - \varepsilon(S_2)) \cdot c + P_t \cdot (1 - c) \quad (42)$$

$$Pr(O \in \bar{O}_f | S_1 \to S_2) = \varepsilon(S_2) \cdot c + P_f \cdot (1 - c), \quad (43)$$

$$Pr(O \in \bar{O}_d | S_1 \to S_2) = (1 - P_t - P_f) \cdot (1 - c). \quad (44)$$

Here, the probability of S_1 and S_2 providing the same true or false value is different when we have different directions of dependence.

By applying the Bayes Rule, we can compute the probabilities of $S_1 \perp S_2$, $S_1 \to S_2$ and $S_2 \to S_1$. If we consider $Pr(S_1 \to S_2) + Pr(S_2 \to S_1)$ as the probability of dependence between S_1 and S_2, the three properties in Theorem 3.1 still hold.

5.2 Accuracy of a data source

We next consider how one can compute the accuracy of a data source. A naive way is to compute the fraction of true values provided by the source. However, we do not know for sure which are the true values, especially among values that are voted by similar number of sources, we can make wrong decisions. We instead compute the accuracy of a source as the average probability of the values provided by it being true.

Formally, let $\bar{V}(S)$ be the values provided by S and let m be the size of $\bar{V}(S)$. For each $v \in \bar{V}(S)$, we denote by $P(v)$ the probability that v is true. We compute $A(S)$ as follows.

$$A(S) = \frac{\Sigma_{v \in \bar{V}(S)} P(v)}{m}. \tag{45}$$

Now we need a way to compute the probability that a value is true. We base our computation both on how many sources provide this value and on the accuracy of those sources.

We start with the case where all data sources are independent. Consider an object $O \in \mathcal{O}$. Let $\mathcal{V}(O)$ be the domain of O; that is, the set of all possible values on O. The size of $\mathcal{V}(O)$ is $n+1$: n false values and one true value. Let $\bar{S}_o$ be the sources that provide information on O. For each $v \in \mathcal{V}(O)$, we denote by $\bar{S}_o(v) \subseteq \bar{S}_o$ the set of sources that vote for v (note that $\bar{S}_o(v)$ can be empty). We denote by $\Psi(O)$ the observation of which value each $S \in \bar{S}_o$ votes for.

We first consider how to compute the probability that we observe a particular voting $\Psi(O)$. Consider a value $v \in \mathcal{V}(O)$. If v is true, the conditional probability of $\Psi(O)$ is the probability that sources in $\bar{S}_o(v)$ each provides the true value and other sources each provides a particular false value:

$$\begin{aligned} Pr(\Psi(O)|v \text{ true}) &= \Pi_{S \in \bar{S}_o(v)} A(S) \cdot \Pi_{S \in \bar{S}_0 - \bar{S}_o(v)} \frac{1-A(S)}{n} \\ &= \Pi_{S \in \bar{S}_o(v)} \frac{nA(S)}{1-A(S)} \cdot \Pi_{S \in \bar{S}_o} \frac{1-A(S)}{n}. \end{aligned} \tag{46}$$

Among the values in $\mathcal{V}(O)$, there is one and only one true value. Assume our a-priori belief of each value being true is the same, denoted by β. We then have

$$\begin{aligned} Pr(\Psi(O)) &= \sum_{v \in \mathcal{V}(O)} (Pr(\Psi(O)|v \text{ true}) \cdot Pr(v \text{ true})) \\ &= \sum_{v \in \mathcal{V}(O)} \left(\beta \cdot \Pi_{S \in \bar{S}_o(v)} \frac{nA(S)}{1-A(S)} \cdot \Pi_{S \in \bar{S}_o} \frac{1-A(S)}{n} \right). \end{aligned} \tag{47}$$

We now apply the Bayes Rule to compute the probability that a particular value $v \in \mathcal{V}(O)$ is true given our observation of $\Psi(O)$.

$$\begin{aligned} &P(v) = Pr(v \text{ true}|\Psi(O)) \\ &= \frac{Pr(\Psi(O)|v \text{ true}) \cdot Pr(v \text{ true})}{Pr(\Psi(O))} = \frac{\Pi_{S \in \bar{S}_o(v)} \frac{nA(S)}{1-A(S)}}{\sum_{v_0 \in \mathcal{V}(O)} \Pi_{S \in \bar{S}_o(v_0)} \frac{nA(S)}{1-A(S)}}. \end{aligned} \tag{48}$$

To simplify the notation, we define the *normalization factor* ω as

$$\omega = \sum_{v_0 \in \mathcal{V}(O)} \Pi_{S \in \bar{S}_o(v_0)} \frac{nA(S)}{1-A(S)}. \tag{49}$$

So ω is independent of $\bar{S}_o(v)$. We define the *confidence* of v, denoted by $C(v)$, as

$$C(v) = \ln P(v) + \ln \omega = \sum_{S \in \bar{S}_o(v)} \ln \frac{nA(S)}{1-A(S)}. \tag{50}$$

If we define the *accuracy score* of a data source S as

$$A'(S) = \ln \frac{nA(S)}{1-A(S)} = -\ln(\frac{1}{A(S)} - 1) + \ln n, \tag{51}$$

we have

$$C(v) = \sum_{S \in \bar{S}_o(v)} A'(S). \tag{52}$$

So we can compute the confidence of a value by summing up the accuracy scores of its providers. Finally, $P(v) = \frac{e^{C(v)}}{\omega}$ and $\omega = \sum_{v_0 \in \mathcal{D}(O)} e^{C(v_0)}$.

A value with a higher confidence has a higher probability to be true; thus, rather than comparing vote counts, we compare confidence of values. The following theorem shows three properties of Equation (52).

THEOREM 5.1. *Equation (52) has three properties:*

1. *If all data sources have the same accuracy, when the size of $\bar{S}_o(v)$ increases, $C(v)$ increases;*
2. *Fixing all sources in $\bar{S}_o(v)$ except S, when $A(S)$ increases for S, $C(v)$ increases.*
3. *If there exists $S \in \bar{S}_o(v)$ such that $A(S) = 1$ and no $S' \in \bar{S}_o(v)$ such that $A(S') = 0$, $C(v) = +\infty$; if there exists $S \in \bar{S}_o(v)$ such that $A(S) = 0$ and no $S' \in \bar{S}_o(v)$ such that $A(S') = 1$, $C(v) = -\infty$.* □

PROOF. We prove the three properties as follows.

1. When all data sources have the same accuracy, they have the same accuracy score. Let A' be the accuracy score and s be the size of $\bar{S}_o(v)$. Then $C(v) = s \cdot A'$, so $C(v)$ increases with s.
2. When $A(S)$ increases for a source S, $A'(S)$ increases as well and so $C(v)$ increases.
3. When $A(S) = 1$ for a source S, $A'(S) = \infty$ and $C(v) = \infty$. When $A(S) = 0$ for a source S, $A'(S) = -\infty$ and $C(v) = -\infty$.

□

Note that the first property is actually a justification of the voting strategy (Proposition 2.1). The third property shows that we should be careful not to assign very high or very low accuracy to a data source. We avoid this problem by defining the accuracy of a source as the average probability of its provided values.

Finally, if a data source S copies a value v from other sources, we should ignore S when computing the confidence of v. Following the same analysis, we compute the confidence of a value v as follows ($I(S)$ is the same as in Equation (16)).

$$C(v) = \sum_{S \in \bar{S}_o(v)} A'(S) I(S). \tag{53}$$

0: **Input**: $\mathcal{S}, \mathcal{O}$.
Output: The true value for each object in $\mathcal{O}$.
1: Set the accuracy of each source as $1 - overall\_error\_rate$;
2: **while** (accuracy of sources do not change much && no oscillation of decided true values)
3: Compute probability of dependence between each pair of sources;
4: Sort sources according to the dependencies;
5: Compute confidence of each value for each object;
6: Compute accuracy of each source;
7: **for each** ($O \in \mathcal{O}$)
Among all values on O, select the one with the highest confidence as the true value;

Algorithm 4: AccuVote: Discover true values by considering accuracy and dependence of data sources.

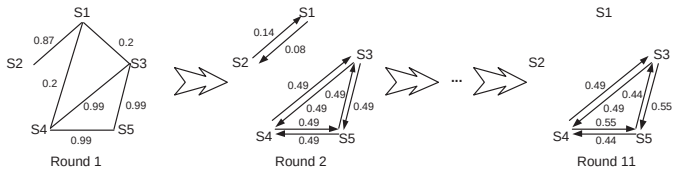

Figure 4: Probabilities of dependencies computed by Accu on the motivating example. We only show dependencies where the sum of the probabilities on both directions is over .1.

5.3 Combining accuracy and dependence

We now extend the Vote algorithm to incorporate analysis of accuracy. We need to compute three measures: accuracy of sources, dependence between sources, and confidence of values. Accuracy of a source depends on confidence of values; dependence between sources depends on accuracy of sources and the true values selected according to the confidence of values, and confidence of values depends on both accuracy of and dependence between data sources.

One possible solution is to interleave voting by considering dependence and voting by considering accuracy till the two types of voting agree on the voting results. This method fails because there are cases where the two types of voting never agree with each other. For example, in the motivating example, voting by considering dependence always converges with results (MIT, MSR, MSR, BEA, Google), while voting by considering accuracy always converges with results (MIT, UWisc, MSR, BEA, UW).

Another option is to conduct analysis of both accuracy and dependence in each round. Specifically, Algorithm AccuVote starts by setting the probability of each value as one minus the overall error rate, iteratively (1) computes accuracy and dependence based on the confidence of values computed in the previous round, and (2) updates confidence of values accordingly, and stops when the accuracy of the sources becomes stable. Note that as we consider direction of dependence, AccuVote may not converge. We stop the process after we detect oscillation of decided true values. Again, our experiments show that the results generated by different rounds during oscillation have similar overall quality (see Section 7).

Example 5.2. *Continue with the motivating example. Figure 4 shows the probability of dependence* Accu *computes, Table 4 shows the accuracy of each data source* Accu *computes, and Table 5 shows the confidence of affiliations computed for* Carey *and* Halevy. *Starting from the second round, S_1 is considered more accurate and the values it provides are given higher weights. In later rounds,* Accu *gradually increases the accuracy of S_1 and decreases that of S_3, S_4 and S_5. At the fourth round,* Accu *decides that* UCI *is the correct affiliation for* Carey *and finds the right affiliations for all researchers. Finally,* Accu *terminates at the eleventh round and the source accuracy and dependence probability it computes converge to close to the real ones.* □

Table 4: Accuracy of data sources computed by Accu on the motivating example.

	S_1	S_2	S_3	S_4	S_5
Round 1	.52	.42	.53	.53	.53
Round 2	.63	.46	.55	.55	.41
Round 3	.71	.52	.53	.53	.37
Round 4	.79	.57	.48	.48	.31
...	...	...	...	...	...
Round 11	.97	.61	.40	.40	.21

Table 5: Confidence of affiliations computed for *Carey* and *Halevy* in the motivating example.

	Carey			*Halevy*	
	UCI	Oracle	BEA	Google	UW
Round 1	1.61	1.61	2.0	2.1	2.0
Round 2	1.68	1.3	2.12	2.74	2.12
Round 3	2.12	1.47	2.24	3.59	2.24
Round 4	2.51	1.68	2.14	4.01	2.14
...	...	...	...	...	...
Round 11	4.73	2.08	1.47	6.67	1.47

5.4 Comparison with TruthFinder

Yin et al. [13] proposed TruthFinder, which considers accuracy of sources in truth discovery. Whereas we both consider accuracy of sources, our model differs from theirs in two aspects.

The most important difference is that we consider the dependence between sources. TruthFinder uses a *dampening factor* to address the possible dependence between sources; however, this approach is not necessarily effective and for our motivating example, TruthFinder incorrectly decides that all values provided by S_3 are true. Our model considers dependence in a principled fashion. By examining the probability that a pair of data sources are dependent and its effect on voting, we are able to apply dampening only when appropriate and apply different "dampening factors" for different data sources.

Another major difference is that we compute the probability of a value being true in a different way. TruthFinder computes it as the probability that at least one of its providers provides the true value and ignores sources that vote for other values. As they pointed out, they have the problem of "overly high confidence" if they do not apply the dampening factor. Our computation (Equation (48)) considers all data sources and considers both the possibility that the value is true and the possibility that the value is false.

Section 7 presents an experimental comparison between the two approaches.

6. EXTENSIONS

This section describes several extensions of the Accu model

by considering probabilities of a value being true in dependence discovery and by relaxing the *Categorical-value* condition and the *Uniform-false-value-distribution* condition. The extensions we present are complementary to each other and can be easily combined for a full model.

AccuPR: Our dependence discovery is based on the observation that two independent data sources are more likely to provide the same true value than the same false value. However, in practice we often cannot decide for sure if a value is true or false; instead, the ACCU model computes for each value a probability that it is true and we can use it in our dependence analysis. In particular, we denote by $Pr(S, v)$ the probability that source S provides value v; then,

$$Pr(S, v) = P(v) \cdot A(S) + (1 - P(v)) \cdot \frac{1 - A(S)}{n}. \quad (54)$$

Accordingly, we compute probability of two sources providing a particular pair of values in dependence analysis. We denote by $Pr(v_1, v_2|C)$ the conditional probability of S_1 providing v_1 and S_2 providing v_2. Then,

$$Pr(v, v|S_1 \perp S_2) = P(S_1, v) \cdot P(S_2, v) = P_c, \quad (55)$$
$$Pr(v_1, v_2|S_1 \perp S_2) = P(S_1, v_1) \cdot P(S_2, v_2) = P_d, \quad (56)$$
$$Pr(v, v|S_2 \to S_1) = cP(S_1, v) + (1 - c)P_c, \quad (57)$$
$$Pr(v_1, v_2|S_2 \to S_1) = (1 - c)P_d. \quad (58)$$

Here, we show the probabilities conditioned on $S_1 \perp S_2$ and $S_2 \to S_1$; those conditioned on $S_1 \to S_2$ can be computed in a similar way. We can apply the Bayes Rule to compute the probability of dependence accordingly.

SIM: We consider similarity between values. Let v and v' be two values that are similar. Intuitively, the sources that vote for v' also implicitly vote for v and should be considered when counting votes for v. For example, a source that claims *UW* as the affiliation may actually mean *UWisc* and should be considered as an implicit voter of *UWisc*.

We extend ACCU by incorporating the similarity model in [13]. Formally, we denote by $sim(v, v') \in [0, 1]$ the similarity between v and v', which can be computed by edit distance of strings, difference between numerical values, etc. After computing the confidence of each value of object O, we adjust them according to the similarities between them as follows:

$$C^*(v) = C(v) + \rho \cdot \sum_{v' \neq v} C(v') \cdot sim(v, v'), \quad (59)$$

where $\rho \in [0, 1]$ is a parameter controlling the influence of similar values. We then use the adjusted confidence in computation in later rounds.

We can go further by considering also values that are *transitively* similar; that is, if both $sim(v_1, v_2)$ and $sim(v_2, v_3)$ are high, we consider v_3 when computing the confidence of v_1. Let $\bar{S}(v)$ be the set of values that are similar to v. Our revision, adopting the underlying idea of PageRank, is as follows.

$$C^*(v) = (1 - \rho)C(v) + \rho \cdot \sum_{v' \neq v} C(v') \cdot \frac{sim(v, v')}{\sum_{v_0 \in \bar{S}(v')} sim(v_0, v')}. \quad (60)$$

Note that this revision is not needed when the transitivity of similarity holds; that is, high $sim(v_1, v_2)$ and $sim(v_2, v_3)$ imply a high $sim(v_1, v_3)$.

NonUni: In reality, false values of an object may not be uniformly distributed; for example, an out-of-date value or a value similar to the true value can occur more often than others. We extend ACCU for this situation as follows.

Assume we know the distribution of false values described by $f(d), d \in [0, 1]$, the percentage of false values whose distribution probability is d; thus, $\int_0^1 f(d) = 1$. Then, the probability that two false-value providers provide the same value is $\int_0^1 d^2 f(d)$ instead of $(\frac{1}{n})^2 \cdot n = \frac{1}{n}$. Accordingly, we revise Equation (37) as

$$Pr(O \in \bar{O}_f|S_1 \perp S_2) = \varepsilon(S_1)\varepsilon(S_2) \int_0^1 d^2 f(d) = P_f. \quad (61)$$

Similarly, we need to revise Equation (46) as follows. Let $E = e^{\int_0^1 \ln df(d)(|\bar{S}_o| - |\bar{S}_o(v)|)}$.

$$Pr(\Psi(O)|v \text{ true}) = \Pi_{S \in \bar{S}_o(v)} A(S) \cdot \Pi_{S \in \bar{S}_o - \bar{S}_o(v)} (1 - A(S)) \cdot E. \quad (62)$$

Finally, we note that a false value v that occurs frequently is not a strong indicator of dependence. This is reflected in the ACCUPR model, which in such case will compute a high probability of v being true and accordingly lower the probability of dependence between v's providers.

7. EXPERIMENTAL RESULTS

To understand how well our algorithm performs on data sources with different characteristics, we experimented on both synthetic and real-world data. Our experimental results show that when there are only independent sources, our algorithms obtain results that are the same as or better than that obtained by naive voting, and when there are copiers, our algorithms are able to detect them and significantly improve the results. In addition, we examined if our algorithm can prevent falsification of true values.

7.1 Experimental settings

We first describe the synthetic data sets we generated and leave the description of the real-world data set to Section 7.5. We consider five types of *universes* of data sources, where each source provides information for all objects.

1. *Indep-source universe* contains 10 independent sources;
2. *Bene-copier universe* contains 10 independent sources and 9 benevolent copiers that copy from the same independent source and provide 20% of the values independently;
3. *Mal-copier universe* contains 10 independent sources and 9 malicious copiers that copy from the same independent source and provide 20% of the values randomly;
4. *Naive-copier universe* contains 10 independent sources and 9 copiers that copy all data from the same independent source.
5. *Pareto universe* contains 25 to 100 data sources, of which 20% are independent and 80% are copiers. Among the independent sources, 20% have an error rate of .2 and 80% have an error rate of .5. Among the copiers, 80% copy from one of the more accurate independent sources and 20% copy from one of the less accurate independent sources. Also, among the copiers, 50% are benevolent and provide 20% of the values independently with an error rate of .1, 25% are malicious and

provide 20% of the values randomly, and 25% copy all values[2].

For the first four types of universes, we consider three cases: *no-authority, copy-from-non-authority* and *copy-from-authority*. In the *no-authority* case, every independent source has the same error rate. In the *copy-from-non-authority* case, there is an *authority* source that provides the true value for each object, but the copiers copy from a non-authority source. In the *copy-from-authority* case, there is an authority source and the copiers copy from it. Note that in the *Indep-source universe*, the latter two cases are the same since we have no copier.

For each type of universe and each case, we randomly generated the set of data sources according to ε_u, the error rate, n_u, the number of incorrect values, and o_u, the number of objects. The values range from 0 to n_u, where we consider 0 as the true value and the others as false. We varied ε_u from .1 to .9, n_u from 5 to 100, and o_u from 5 to 100. For *Pareto universe*, we in addition randomly decided from which source a copier copies according to the distribution. For each set of parameters, we randomly generated the data set 100 times, applied our algorithms to decide the true values, and reported the average precision of the results. We define *precision* of the results as the fraction of objects on which we select the true values (as the number of true values we return and the real number of true values are both the same as the number of objects, the *recall* of the results is the same as the precision). Note that this definition is different from that of accuracy of sources.

We implemented models BENE, MAL, ACCU, ACCUPR and SIM as described in this paper. We also implemented the following methods for comparison:

- NAIVE conducts naive voting;
- NAIVESIM conducts naive voting but considers similarity between values;
- ACCUNODEP considers accuracy of sources as we described in Section 5, but assumes all sources are independent;
- TF applies the model presented in [13].
- TFNOSIM is the same as TF except that it does not consider similarity between values.
- TFNODAM is the same as TFNOSIM except that it does not apply the dampening factor (0.3).

For all methods, when applicable we (1) set $\alpha = .2$ and $c = .8$, (2) set ε and n to the value used in generating the data sources, (3) set $\varepsilon = .25$ for the *Pareto universe*, and (4) set $\rho = 1$ for SIM. We implemented the algorithms in Java and conducted our experiments on a WindowsXP machine with AMD Athlon(tm) 64 2GHz CPU and 960MB memory.

7.2 Comparing dependence-detection models

We first compare our dependence-detection methods, namely, BENE, MAL and ACCU, on the first four types of universes. We use NAIVE as a baseline. We report our results for $n_u = 100$ and $o_u = 100$ and briefly discuss the results for other parameter settings at the end of this section. Figure 5 plots the precision on the *Indep-source universe*. Figure 6 plots the precision on the *Bene-copier*, *Mal-copier*

[2]We call it *Pareto universe* as it observes the *Pareto Rule (80/20 Rule)* in many aspects.

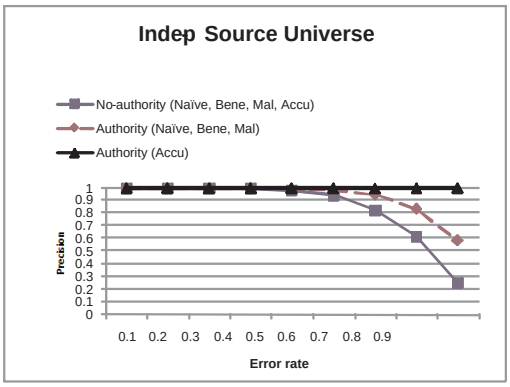

Figure 5: Precision in the *Indep-source universe*.

and *Naive-copier universes*. Table 6 summarizes the performance of different models on different types of universe.

In the *Indep-source universe*, each method obtains higher precision when there is an authority source; also, the higher the error rate, the lower the precision. BENE and MAL obtain the same precision as NAIVE, and ACCU obtains a precision of 1.0 when an authority source exists. Being able to obtain the same results as simple voting in absence of copiers is an important feature of our models, showing that in dependence discovery we do not generate false positives that can change the voting results.

In the *Bene-copier universe*, since the number of copiers is close to the number of independent sources, if we apply naive voting, the source being copied can almost dominate the results and the precision of the results is similar to the accuracy of that source (not exactly the same because the benevolent copiers also independently provide some information). However, our algorithms are able to detect the dependence and so obtain much higher precision. Specifically, we make the following observations.

- BENE successfully detects the dependence. As it considers the independent contribution made by copiers, it obtains a higher precision than in the case where there are no copiers.
- MAL is also able to detect the dependence. In general it obtains similar precision to the no-copier case as it ignores the independent contribution from the copiers. However, MAL has a low precision in the *copy-from-non-authority* case when the error rate is low. In this case, the non-authority independent sources share many true values with the authority, and the rest of their values are all false; thus, MAL suspects them to be copiers even more than the real copiers. As a result, the real copiers have more effect on the voting results and the false values they copy have higher chance to win.
- ACCU obtains a precision of 1.0 when there exists an authority source and the same precision as BENE otherwise. The only exception is when $\varepsilon = .9$ in the *no-authority* case. With this high error rate and low number of independent sources, the independent sources often do not agree with each other on the true value of an object. Then, the values that are copied 9 times (even though ACCU detects the copying) have a slightly higher confidence in the first round. Thus, the copiers

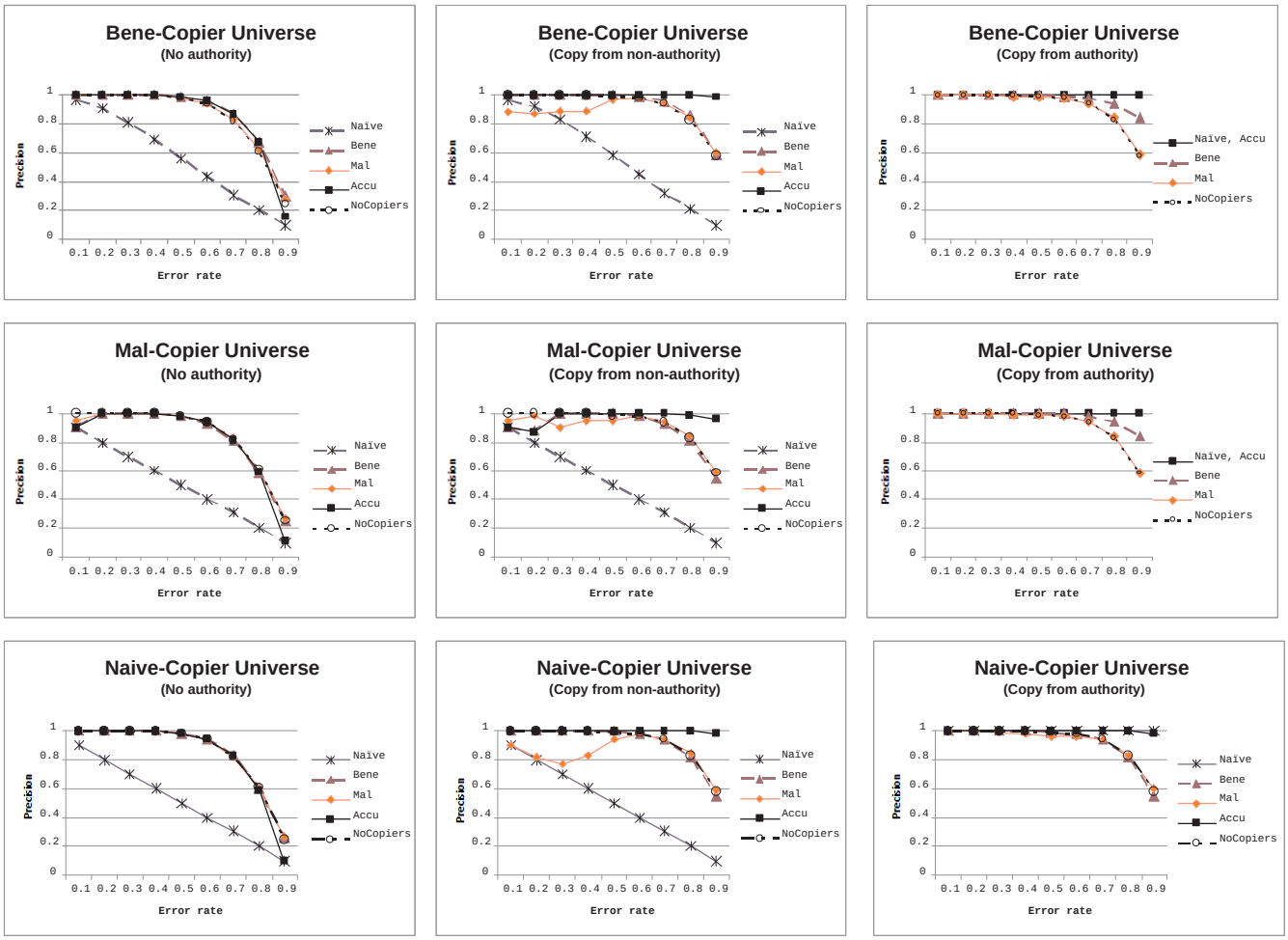

Figure 6: Precision in the *Bene-copier universe* and *Mal-copier universe*. For comparison, we also plotted the precision of NAIVE when there is no copier.

are considered to be more accurate and gradually dominate the results. This problem disappears when more independent sources are present.

In the *malicious-copier universe*, we observe that the naive model obtains almost the same precision as the precision of the source being copied, as the malicious copiers do not contribute trustable information. The major difference from the *benevolent-copier universe* is that when $\varepsilon_u \leq .2$, there are more common values between independent sources than between a source and its copier. Thus, our dependence-detection techniques ring false alarms and obtain a slightly lower precision. In addition, because BENE and ACCU consider the vote of copiers with a certain weight, they give the copiers the chance to strengthen the results they copy, so MAL in general has slightly higher precision than BENE.

Finally, in the *Naive-copier universe*, we observe very similar pattern to the *benevolent-copier universe*, showing that even though $c = .8$ does not conform to the real data, our algorithms still perform well. The precision of BENE and ACCU is slightly lower, as now the copiers do not contribute independent information. The precision of MAL is lower in the *copy-from-non-authority* case when the error rate is low, as naive copiers are actually harder to be detected than malicious copiers using the malicious model.

Note that we observed that MAL and ACCU do not converge on some data sets and the probability that MAL does not converge is much higher. In case of inconvergence, the selected true values are changed back and forth on a few objects, but the overall precision remains similar.

Effects of parameters: We also conducted experiments with different parameter settings and have the following observations.

First, we varied parameters in the random generation of data sources. We observed that when there are fewer objects or fewer false values in the domain, there is less statistical evidence for dependence detection and so the precision of the results can be lower and less stable. On the other hand, when there are more data sources, true values are provided by more sources and the precision can be higher.

Second, we varied voting parameters, including α, c, ε and n. We observed that our models are robust with respect to parameters that represent a-priori probabilities; in particular, ranging α and ε from .1 to .9 does not change precision of the results. This observation is common in Bayes analysis. We also observed that ranging n from 10 to 100 does not change the precision and setting it to 1000 or 10000 even

Table 6: Summary of performance of different dependence-detection models on different types of universe. We abbreviate *no-authority* as NA, *copy-from-non-authority* as CNA, and *copy-from-authority* as CA.

Model	Indep Srcs	Benevolent Copiers	Malicious Copiers	Naive Copiers	Pareto
BENE	Good	Good	Fair	Good	Excellent
MAL	Good	Fair	Good except low ε in the CNA case	Good except low ε in the CNA case	Good
ACCU	Excellent	Excellent except when high ε & low #sources in the NA case	Excellent except when high ε & low #sources in the NA case or low ε in the NA and CNA cases	Excellent except when high ε & low #sources in the NA case	Excellent

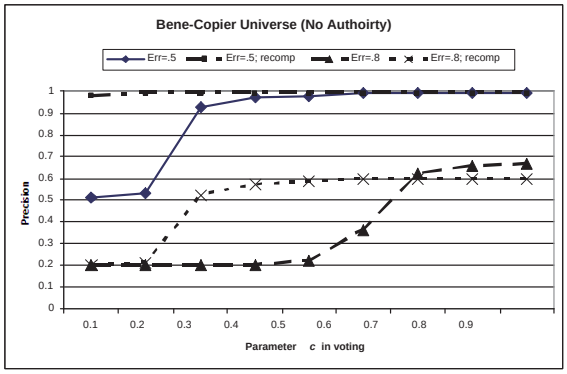

Figure 7: Precision of results when parameter c varies. Recomputation of c can significantly improve the precision.

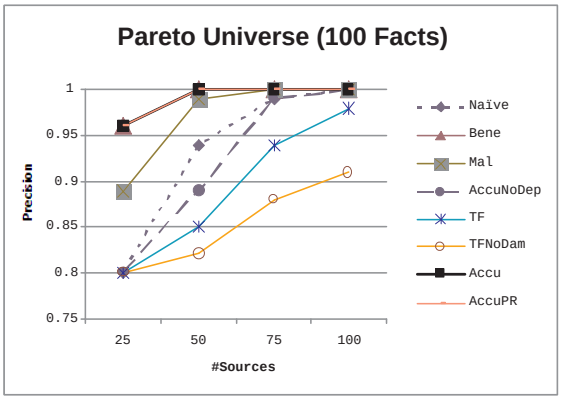

Figure 8: Precision in the *Pareto universe*.

increases the precision. However, we observed that ranging c from 0 to 1 can significantly change the precision. As an example, Figure 7 shows the precision of the results in the *benevolent-copier universe* ($c_u = .8$) and *no-authority* case. When we set c to the same as c_u, we obtain a precision of .99 when $\varepsilon_u = .5$ and .66 when $\varepsilon_u = .8$. However, when we range c from .1 to .9, in case of $\varepsilon_u = .5$, the precision of the results drops significantly when c is set to .2; in case of $\varepsilon_u = .8$, the precision drops significantly when c is set to .6. Recomputing c in each iteration in voting can effectively solve the problem: the precision remains stable when $\varepsilon_u = .5$ and drops at $c = .2$ when $\varepsilon_u = .8$.

7.3 Comparing truth-discovery algorithms

To examine the effect of each algorithm in a more complex universe, we experimented on the Pareto universe. Figure 8 shows the precision for a universe with 100 values. We observe that ACCU and BENE obtain the highest precision, showing that considering dependence between sources significantly improve results of truth discovery, and when more accurate sources are copied more often, considering accuracy of sources does not necessarily help. MAL also performs reasonably well. ACCUNODEP, TF and TFNODAM obtain even lower precision than NAIVE, showing that considering accuracy of sources while being unaware of dependence can become more vulnerable in presence of duplications. ACCUNODEP and TFNODAM both extend NAIVE with only analysis of source accuracy but do so in different ways; between them ACCUNODEP obtains better results. Finally, we observe that ACCUPR does not obtain higher precision than ACCU, showing that considering the probabilities of values being true in dependence analysis does not necessarily improve the results.

7.4 Preventing falsification

We next studied whether our algorithms can prevent falsification. We consider a Pareto universe with 25 (normal) data sources (so 5 independent sources) and a set of falsifiers who intend to falsify the true values on a set of f objects. Among the falsifiers, one is a bad independent source and the others are malicious copiers. For each object that the falsifiers intend to falsify, all falsifiers provide value -1, which is not provided by any other source in the universe. For the rest of the objects, the independent falsifier provides values observing a certain error rate, and the copiers copy from the independent falsifier in a malicious manner.

We classify falsifiers into the following four categories (the error rate is that on the objects not to be falsified).

- *Innocent:* The independent falsifier has an error rate $\varepsilon_f = .5$ and the copiers copy all values;
- *Knowledgeable:* The independent falsifier has an error rate of 0 and the copiers copy all values;
- *Smart:* The independent falsifier has an error rate $\varepsilon_f = .5$ and the copiers provide a certain percentage of the values randomly ($c_f < 1$);
- *Sophisticated:* The independent falsifier has very low error rate and the copiers provide a certain percentage of the values randomly;

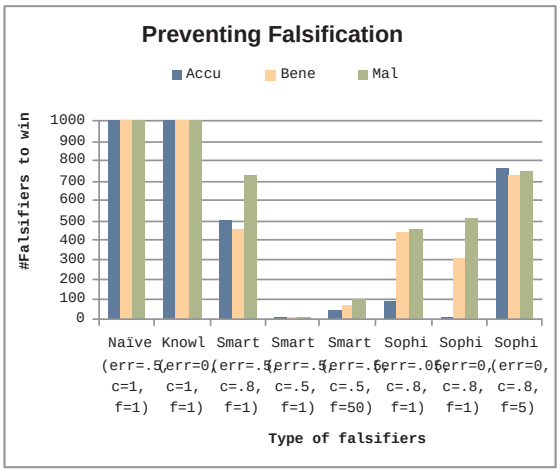

Figure 9: Number of falsifiers required to falsify a set of true values. Our algorithms can effectively prevent innocent and knowledgeable falsifiers from falsifying one true value and prevent smart and sophisticated falsifiers from falsifying multiple true values.

We want to find out how many falsifiers are required to falsify the true values for f objects. For each category of falsifiers, we started from one randomly generated independent falsifier and gradually added copier falsifiers. If -1 is selected as the true value for each object of falsification three times consecutively, we stopped and reported the number of falsifiers. If after reaching 1000 falsifiers the falsifiers still cannot succeed, we stopped and reported 1000. Figure 9 shows the results for different types of falsifiers. We have several observations.

First, it is very hard for innocent and knowledgeable falsifiers to falsify the true values: even 1000 falsifiers cannot falsify one true value against 25 normal sources.

Second, when $c_f = .8$, it requires around 500 smart falsifiers to falsify one true value; but when $c_f = .5$, the falsifiers look like independent sources and only 6 falsifiers are required under the ACCU algorithm. From another perspective, this result indicates that even if a wrong value is provided by a set of more accurate sources, a number of independent normal sources *do* have the chance to fix it.

Third, under the ACCU model it is easy for sophisticated falsifiers with $\varepsilon_f = 0$ and $c_f = .8$ to win: only 4 falsifiers are required to falsify one value. From another perspective, this indicates that a value provided by an authority data source is more likely to be considered as true, even if a different value is provided by several less accurate sources. BENE and MAL do not consider accuracy of sources so can still effectively prevent falsification: more than 300 falsifiers are required to falsify one value. If the independent falsifier has an error rate of .05, then even ACCU can detect them and nearly 100 falsifiers are required to win.

Fourth, it is hard even for sophisticated falsifiers to falsify multiple true values. Falsifying 5 true values requires more than 700 sophisticated falsifiers with $\varepsilon_f = 0$ and $c_f = .8$. However, it is still easy for smart falsifiers to falsify a set of true values: no more than 100 sources are required to falsify 50 true values, reflecting one direction for improvement: intuitively, if a number of sources provide the same values on a number of objects and the values are different from those provided by other sources, while provide very different values on other objects, we should suspect the dependence between them.

Finally, though ACCU seems to be less effective than MAL in preventing falsification, for each case that it cannot prevent falsification, there is another scenario with equivalent sources where the decision of ACCU is more appropriate.

Table 7: Results on the book data set. For each method, we report the precision of the results, the runtime, and the number of rounds for convergence.

Model	Precision	Rounds	Time(s)
NAIVE	.71	1	.2
NAIVESIM	.74	1	.2
ACCUNODEP	.79	23	1.1
BENE	.83	3	28.3
ACCU	.87	22	185.8
SIM	.89	18	197.5
TFNOSIM	.71	10	.5
TF[5]	.83	8	11.6

7.5 Experiments on real-world data

We experimented on a real-world data set also used in [13][3] (we removed duplicates). The data set was extracted by searching computer-science books on *AbeBooks.com*. For each book, *AbeBooks.com* returns information provided by a set of online bookstores. Our goal is to find the list of authors for each book. In the data set there are 877 bookstores, 1263 books, and 24364 listings (each listing contains a list of authors on a book provided by a bookstore).

We did a pre-cleaning of authors' names and generated a normalized form that preserves the order of the authors and the first name and last name (ignoring the middle name) of each author. On average, each book has 19 listings; the number of different author lists after cleaning varies from 1 to 23 and is 4 on average.

We applied various models on this data set (we did not apply MAL as we believe all sources are benevolent) and set $\alpha = .2, c = .8, \varepsilon = .2$ and $n = 100$ when applicable. We compared similarity of two author lists using 2-gram Jaccard distance.

We used a golden standard that contains 100 randomly selected books and the list of authors found on the cover of each book (the same golden standard as used in [13]). In the golden standard we represent the author list in the same normalized format. We compared the results of each method with the golden standard and reported the precision. We consider missing or additional authors, misordering, misspelling, and missing first name or last name as errors; though, we do not report missing or misspelled middle names[4].

Precision and Efficiency Table 7 lists the precision of

[3]We thank authors of [13] for providing us the data set.

[4]Note that the precision we reported is not comparable with that reported in [13], as their partially correct results are each given a partial score between 0 and 1, mis-ordering of authors is not penalized, but incorrect or missing middle name is penalized.

[5][13] reports that correct authors were provided for 85 books; however, they did not count mis-ordering of authors as incorrect.

Table 8: Bookstores that are likely to be copied by more than 10 other bookstores. For each bookstore we show the number of books it lists and its accuracy computed by SIM.

Bookstore	#Copiers	#Books	Accu
Caiman	17.5	1024	.55
MildredsBooks	14.5	123	.88
COBU GmbH & Co. KG	13.5	131	.91
THESAINTBOOKSTORE	13.5	321	.84
Limelight Bookshop	12	921	.54
Revaluation Books	12	1091	.76
Players Quest	11.5	212	.82
AshleyJohnson	11.5	77	.79
Powell's Books	11	547	.55
AlphaCraze.com	10.5	157	.85
Avg	12.8	460	.75

each algorithm. SIM obtained the best results and improved over NAIVE by (.89-.71)/.71=25.4%. NAIVESIM, ACCUNODEP and BENE each extends NAIVE on one aspect; while all of them increased the precision, BENE increased the most. We also observed that considering similarity between author lists increased the precision of ACCU only slightly (by 2.3%), but increased the precision of TFNOSIM significantly (by 16.9%); indeed, TFNOSIM obtained the same precision as NAIVE.

To further understand how considering dependence and precision of sources can affect our results, we looked at the books on which ACCU and NAIVE generated different results and manually found the correct authors. There are 143 such books, among which ACCU and NAIVE gave correct authors for 119 and 15 books respectively, and both gave incorrect authors for 9 books.

Finally, BENE was quite efficient and finished in 28.3 seconds. It took ACCU and SIM longer time to converge (3.1 minutes, 3.3 minutes respectively); though, truth discovery is often a one-time process and so taking a few minutes is still reasonable.

Dependence and source accuracy: Out of the 385,000 pairs of bookstores, 2916 pairs provide information on at least the same 10 books and among them SIM found 508 pairs that are likely to be dependent. Among each such pair S_1 and S_2, if the probability of S_1 depending on S_2 is over 2/3 of the probability of S_1 and S_2 being dependent, we consider S_1 as a *copier* of S_2; otherwise, we consider S_1 and S_2 each has .5 probability to be a *copier*. Table 8 shows the bookstores whose information is likely to be copied by more than 10 bookstores. On average each of them provides information on 460 books and has accuracy .75. Note that among all bookstores, on average each provides information on 28 books. This conforms to the intuition that small bookstores are more likely to copy data from large ones. Interestingly, when we applied NAIVE on only the information provided by bookstores in Table 8, we obtained a precision of only .58, showing that bookstores that are large and referenced often actually can make a lot of mistakes.

Finally, we compare the source accuracy computed by our algorithms with that sampled on the 100 books in the golden standard. Specifically, there were 46 bookstores that provide information on more than 10 books in the golden standard. For each of them we computed the *sampled accuracy* as the fraction of the books on which the bookstore provides the same author list as the golden standard. Then, for each bookstore we computed the difference between its accuracy computed by one of our algorithms and the sampled accuracy. Table 9 shows the comparison. The source accuracy computed by SIM is the closest to the sampled accuracy, indicating the effectiveness of our model on computing source accuracy and showing that considering dependence between sources helps obtain more precise source accuracy. The source accuracy computed by TFNOSIM is high, consistent with the observation of *overly high confidence* made in [13].

Table 9: Difference between accuracy of sources computed by our algorithms and the sampled accuracy on the golden standard.

	Sampled	SIM	ACCU	ACCU NODEP	TF NOSIM
Avg src accu	.542	.607	.614	.623	.908
Avg diff	-	.082	.087	.096	.366

8. RELATED WORK

We are not aware of any existing work on detecting dependence between data sources. *Data provenance* [4] is a hot research topic but it focuses on managing provenance information already provided by users or applications. *Opinion pooling*, which combines probability distribution from multiple experts and arrives at a single probability distribution to represent the consensus behavior, has been studied in the context of dependent experts in [5, 6, 9]; however, these works did not study how to discover such dependence. Moss [10] detects plagiarism of programs by comparing *fingerprints* (k-grams) of the programs; our method is different in that we consider values provided for different objects in databases.

There has been many works studying how to assess trustworthiness of data sources. Among them, PageRank [3], Authority-hub analysis [8], etc., decide authority based on link analysis [2]. EigenTrust [7] and TrustMe [11] assign a global trust rating to each data source based on its behavior in a P2P network. The strategy that is closest to ours is TruthFinder [13], with which we have compared in detail in Section 5.4 and in experiments.

Finally, a lot of research has been done on combining conflicting data from multiple sources. Bleiholder and Naumann [1] surveyed existing strategies for resolving inconsistency in structured or semi-structured data and showed how to implement them within an information integration system. Wu and Marian [12] proposed aggregating query results from different web sources by considering importance and similarity of the sources. Our algorithm differs from theirs in that we developed formal models to discover dependence between data sources and accuracy of sources, based on which we decide truth from conflicting information.

9. CONCLUSIONS

In this paper we studied how to improve truth discovery by detecting dependence between sources and analyzing accuracy of sources. We considered a snapshot of data and developed Bayesian models that discover copiers by analyzing values shared between sources. The results of our models can be considered as a probabilistic database, where each object is associated with a probability distribution of various values in the underlying domain. Experimental results

show that our algorithms can significantly improve accuracy of truth discovery and are scalable when there are a large number of data sources.

Our work is a first step towards integrating data among sources where some can copy from others. There are many future topics under this umbrella. First, we plan to extend our current models by considering evolution of data. Second, we plan to combine techniques of record linkage and truth discovery to enhance both of them. Third, we plan to leverage knowledge of dependence between sources to answer queries more efficiently in a data integration system.

10. REFERENCES

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

[2] A. Borodin, G. Roberts, J. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM TOIT*, 5:231–297, 2005.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] P. Buneman, J. Cheney, and S. V. Wang-Chiew Tan. Curated databases. In *Proc. of PODS*, 2008.

[5] K. Chang. *Combination of opinions: the expert problem and the group consensus problem*. PhD thesis, University of California, Berkeley, 1985.

[6] S. French. Updating of belief in the light of someone else's opinion. *Jour. of Roy. Statist. Soc. Ser. A*, 143:43–48, 1980.

[7] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proc. of WWW*, 2003.

[8] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *SODA*, 1998.

[9] D. Lindley. Reconciliation of probability distributions. *Oper. Res.*, 31:866–880, 1983.

[10] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: Local algorithms for document fingerprinting. In *Proc. of SIGMOD*, 2003.

[11] A. Singh and L. Liu. TrustMe: anonymous management of trust relationshiops in decentralized P2P systems. In *IEEE Intl. Conf. on Peer-to-Peer Computing*, 2003.

[12] M. Wu and A. Marian. Corroborating answers from multiple web sources. In *Proc. of WebDB*, 2007.

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