Semi-Supervised Truth Discovery

Xiaoxin Yin, Wenzhao Tan
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{xyin, wentan}@microsoft.com

ABSTRACT

Accessing online information from various data sources has become a necessary part of our everyday life. Unfortunately such information is not always trustworthy, as different sources are of very different qualities and often provide inaccurate and conflicting information. Existing approaches attack this problem using unsupervised learning methods, and try to infer the confidence of the data value and trustworthiness of each source from each other by assuming values provided by more sources are more accurate. However, because false values can be widespread through copying among different sources and out-of-date data often overwhelm up-to-date data, such bootstrapping methods are often ineffective.

In this paper we propose a semi-supervised approach that finds true values with the help of ground truth data. Such ground truth data, even in very small amount, can greatly help us identify trustworthy data sources. Unlike existing studies that only provide iterative algorithms, we derive the optimal solution to our problem and provide an iterative algorithm that converges to it. Experiments show our method achieves higher accuracy than existing approaches, and it can be applied on very huge data sets when implemented with MapReduce.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *search process.*

General Terms

Algorithms, Measurement, Experimentation.

Keywords

Truth discovery, Semi-supervised, Data quality.

1. INTRODUCTION

The web has become the major information source for most of us. However, is information on the web always trustworthy and accurate? It is not surprising that many people will say "no". According to a recent survey reported in [9], U.S. consumers have low trust in online information sources. Even the most trusted online source, company web sites, are only trusted by 22% of consumers. The inaccuracy of online information also causes problems for search engines that provide structured data as results. Figure 1 contains two examples of incorrect fact answers from Google as in August 2010. Figure 1 (a) shows the answer for the query "ps3 release date", which is obviously incorrect as PS3 has been on the market since 2006. The answer in Figure 1 (b) provides a number from various out-dated sources. The number from the recently updated official site (Figure 1 (c)) is actually much larger. Many good examples of erroneous information and their propagation on the web can be found in [8].

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

It is a very important task to distinguish between true and false information on the web. This task, which has been studied as the *truth discovery problem* by different researchers [6][10][16], is defined as follows. Given a set of data sources (e.g., web sites) and a set of facts each provided by one or more data sources, how do we predict the confidence of each fact (i.e., likelihood of being true) and the trustworthiness of each data source. In our usage the word "fact" is used to represent something claimed as true, whether it is right or wrong. In the three approaches described in [16], [6] and [10] the truth discovery problem is formulated as an unsupervised learning problem. It is assumed that a fact provided by more sources (especially more trustworthy and more independent sources) is more likely to be correct. They all use iterative approaches, which start by assigning the same trustworthiness to all data sources, and iterate by computing the confidence of each fact and propagating back to the data sources.

There are two major problems with the above approaches. First, each step of the iterative procedure is performed using simple weighted voting. Consequently the rich will get richer over iterations. However, voting by the majority is not very trustworthy. Information copying is extremely common on the web [5]. Erroneous information can often appear in many sites. The situation is even worse for facts changing with time, since out-of-date information often exists in more web sites than up-to-date information.

A good way to solve this problem is to introduce some level of supervision, so that the truth discovery procedure can be guided toward the right direction. It is usually easy to obtain a small set of highly confident facts, either by manual labeling or from a highly trusted source such as Wikipedia or government web sites. We can treat this set of facts as ground truth, and use them to infer the trustworthiness of data sources and confidence of facts.

The second problem with the existing approaches is that, although they all use iterative algorithms, they provide no guarantee

(a) Google's direct answer for "ps3 release date"

(b) Google's direct answer for "minot air force base population"

(c) Part of page www.minot.af.mil/library/factsheets/factsheet.asp?id=3787

Figure 1: Two incorrect fact answers from Google

of convergence, and no description of the objective function being optimized. In this paper we formulate the truth discovery problem as an optimization problem and prove that our iterative algorithm converges to the optimal solution.

In this paper we study the problem of *Semi-Supervised Truth Discovery*, which is a truth discovery problem where a small number of ground truth facts that are known. The goal of semi-supervised truth discovery is to assign a confidence score to each fact, so that true facts have higher scores than false facts. We call our approach *Semi-Supervised Truth Finder,* or *SSTF*, which considers truth discovery as a graph learning problem by treating each fact with a graph node and encoding the relationships between facts into graph edges.

There are three types of relationships between facts that help us infer the confidence scores of unlabeled facts from the labeled facts (i.e., ground truth facts). Some facts, such as "Google has 21000 employees" and "Google has 21500 employees", are mutual supportive. If one of them is correct, the other is likely to be correct as well. Thus they should have similar confidence scores. Some facts, such as "Microsoft was founded in New Mexico" and "Microsoft was founded in Washington", are mutual exclusive. If one of them has a positive score, the other should have a negative score. The third type of relationship is among facts from same data source. There should be some consistency among the scores of facts from the same data source. If we know a data source provides many true facts and few false facts, then it is trustworthy. Any other facts it provides are likely to be true as well. We represent all these relationships with edge weights in the graph, and convert the semi-supervised truth discovery problem into an optimization problem that aims to assign scores to graph nodes that are consistent with the relationships indicated by the graph edges. We first provide an analytical solution to this optimization problem, although it is very expensive to compute for large scale problems. Then we provide an iterative procedure that can be computed and prove it converges to the optimal solution.

Compared to existing approaches [6][10][16], we do not assign higher score to facts provided by more data sources, because it is very difficult to distinguish whether they are independently authored or copied from each other. The data sources play a different role in our approach: They help link different facts so that we can infer the confidence scores of facts from the ground truth.

First, we test our approach on five real-world data sets collected from the web, and find our approach achieves higher accuracy than existing approaches. Then we test our approach on a very large, diverse, and noisy data set containing attribute-values for all kinds of entities extracted from HTML tables of the whole web, with data from Wikipedia serving as ground truth. Our approach can distinguish true and false facts with high accuracy and is more accurate than previous approaches.

The remaining of this paper is organized as follows. Related work is discussed in Section 2. We define our problem in Section 3 and describe the analytical solution in Section 4. Section 5 presents the iterative solution. Experiments are presented in Section 6 and we conclude this study in Section 7.

2. RELATED WORK

There has been some research on the general problem of combining conflicting information, with a brief survey in [1]. Early work in this area is more focused on how to integrate conflicting answers from different sources, such as the approach in [15].

The truth discovery problem was first proposed in [16], which provides a probabilistic approach based on the assumption that different data sources are independent and thus false values appearing on different data sources should be different from each other. The same assumption is used in [10], which applies a different model for estimating the confidence of facts. The authors in [6] propose a method that considers the dependencies among data sources, although such dependencies need to be inferred from the confidence associated with each fact. Truth discovery on time-variant facts is studied in [7].

The approaches in [16] and [10] assign confidence scores to facts based on the principle that a fact provided by more (and more trustworthy) data sources is more likely to be correct. A data source providing mostly high-confidence facts is more trustworthy. This assumption holds when different data sources are independent. But this is generally not true as data copying is prevalent on the web [5].

The problem caused by data copying is alleviated in [6], which detects copying relationships during the iterative process of truth discovery. However, the copying relationships can only be detected by false facts, as two data sources sharing many true facts do not indicate copying behavior. Therefore, it still finds true and false facts according to the numbers of data sources providing each fact at the beginning. The existence of a false or out-of-date fact in many data sources will cause the fact to receive high confidence in the first iteration, which poisons the remaining iterations. Moreover, sometimes it is simply impossible to distinguish true facts from false ones in the data itself, especially when large amounts of out-of-date facts exist on more data sources than up-to-date facts.

In this paper we study the problem of truth discovery with semi-supervised graph learning, by using a small set of ground truth data to help distinguishing true facts from false ones as well as identifying trustworthy data sources. Semi-supervised graph learning has been studied by Zhu et al. [11][19][20] and Zhou et al. [18]. The main purpose of these approaches is to make predictions consistent with both labeled data and the graph structure. We adapt the approach in [19][20] to our problem and make it scale to very large data sets.

3. PROBLEM FORMULATION

In this paper we study semi-supervised truth discovery, which aims to distinguish true from false facts by utilizing a small set of ground truth facts. It is semi-supervised because a large amount of unlabeled facts also participate in the learning process.

The input to our problem is the same as traditional truth discovery, except that there is a subset of facts which are labeled as correct (i.e., ground truth facts). The goal is to assign a confidence score to each unlabeled fact, so that true facts have higher scores. In this paper we define a confidence score to be a real value between −1 and 1. A score close to 1 indicates we are very confident that a fact is true. A score close to −1 indicates the reverse. A score close to 0 indicates that we do not know if a fact is true or false. Each ground truth fact has a confidence score of 1.

Our approach is based on three basic principles. First, facts provided by the same data source should have similar confidence scores. This is also an important principle utilized in all existing approaches [6][10][16], which assign a trustworthiness score to each data source and estimate the confidence scores of facts using the trustworthiness of their data sources.

Second, similar (and therefore mutual supportive) facts should have similar confidence scores. For example, suppose one data source says the population of Seattle is 560,000 and another says

it is 561,000. If one of these two facts has a high confidence score, the other should have a high score as well.

Third, if two facts are conflicting, they cannot be both true. If one of them has a high positive confidence score, the other should have negative score. For example, if a ground truth fact says that Tom Hanks was born on 1956/07/09, while another fact says he was born on 1956/08/09, the second fact is likely to be wrong.

Here we provide a formal definition of the semi-supervised truth discovery problem. There are n facts $F = \{f_1, \dots, f_n\}$, each provided by one or more of the m data sources $D = \{d_1, \dots, d_m\}$. A subset of facts $F_l = \{f_1, \dots, f_l\}$ are ground truth and thus labeled as true, while the remaining facts $F_u = \{f_{l+1}, \dots, f_n\}$ are unlabeled. Each fact f is on a subject $s(f)$. For example, the fact "Tom Hanks was born on 1956/07/09" is about the subject "Tom Hanks' birth date". Two facts f_1 and f_2 on the same subject may be consistent or in conflict with each other. A function $\text{sim}(f_1, f_2)$ is provided to indicate the degree of consistency or conflict between them $(-1 \le \text{sim}(f_1, f_2) \le 1)$. $\text{sim}(f_1, f_2)$ will be used in our optimization to indicate how important it is to assign similar (or different) scores to f_1 and f_2. As mentioned in other works on truth discovery [6][16] and semi-supervised learning in graphs [20], the definition of $\text{sim}(f_1, f_2)$ is often domain-specific and usually needs to be provided by people with proper domain knowledge. The similarity function should be symmetric, i.e., $\text{sim}(f_1, f_2) = \text{sim}(f_2, f_1)$, and $\text{sim}(f, f) = 1$ for any fact f. Each data source can only provide one fact for each subject, although a fact can be a set-value, such as the authors of a book.

We model this problem as a graph optimization problem. The facts are modeled by a graph, with a node for each fact and an edge between each pair of related facts. The above three principles can be encoded into the graph using edge weights. w_{ij} is the weight of the edge between f_i and f_j, which indicates the relationship of their confidence scores. If f_i and f_j are provided by the same data source, then w_{ij} is set to a positive value α ($0<\alpha<1$) because if f_i has a high (or low) confidence score, f_j should probably have that as well. If f_i and f_j are on the same subject, then we set $w_{ij} = \text{sim}(f_i, f_j)$. Otherwise w_{ij} is set to zero.

In order to formulate the semi-supervised truth discovery as an optimization problem, we choose the loss function based on studies on semi-supervised graph learning [18][19][20], which have been widely used in many applications such as question answering [3] to image annotation [14].

Consider an assignment of confidence scores to facts $\boldsymbol{c} = (c_1, \dots, c_n)$, where $c_i \in [-1, 1]$ is the score of f_i. If $w_{ij} \ge 0$ for all i, j, the loss function in [20] is suitable:

$$E'(\boldsymbol{c}) = \frac{1}{2}\sum_{i,j} w_{ij} \left(c_i - c_j\right)^2. \quad (1)$$

By minimizing $E'(\boldsymbol{c})$ we minimize the weighted sum of differences between the confidence scores of related facts. Although this is an option, it does not consider conflicting relationships between facts, which causes much information to be lost. Furthermore, we can easily minimize $E'(\boldsymbol{c})$ by assigning the score of 1 to each fact.

The second option is to use the same loss function, but allow w_{ij} to be negative. If $w_{ij} < 0$ (i.e., facts f_i and f_j in conflict), then $E'(\boldsymbol{c})$ is minimized when c_i and c_j are different from each other. However, under this definition $E'(\boldsymbol{c})$ is not a convex function and may have many local minimums. Thus it is extremely difficult to optimize, especially for large-scale problems.

Finally we choose a loss function from [11], which is a variant of Equation (1) but handles both similarity and dissimilarity:

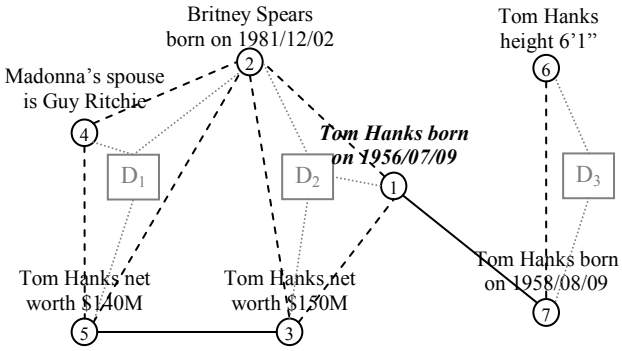


Figure 2: An example graph of facts

$$E(\boldsymbol{c}) = \frac{1}{2}\sum_{i,j} |w_{ij}| \left(c_i - s_{ij} c_j\right)^2, \quad (2)$$

where $s_{ij} = \begin{cases} 1, \text{ if } w_{ij} \ge 0 \\ -1, \text{ if } w_{ij} < 0 \end{cases}$.

In order to minimize $E(\boldsymbol{c})$, f_i and f_j should have similar confidence scores when $w_{ij} > 0$ (i.e., f_i and f_j are mutual supportive). When $w_{ij} < 0$ (i.e., f_i and f_j are mutual exclusive), f_i and f_j should have opposite scores or scores both close to zero. The scores of labeled facts are fixed at 1 and cannot be changed. By minimizing $E(\boldsymbol{c})$ we get an assignment of scores that are not long consistent with the relationships among facts, but also consistent with the scores given to the labeled facts.

An example graph of facts is shown in Figure 2. It contains seven facts $f_1, \dots, f_7$ provided by three data sources D_1, D_2, D_3. f_1 is a ground truth fact. Because f_7 is mutual exclusive from f_1, $\text{sim}(f_1, f_7)$ should be close to -1. Thus f_7 will have a low confidence score, which also leads to a low score for f_6. f_2 and f_3 have high scores because they are provided by the same data source as f_1. f_5 is consistent with f_3 and thus has high score as well. f_4 also has high score because of its connections to f_2 and f_5.

4. ANALYTICAL SOLUTION

Although Equation (2) is proposed in [11], the authors of [11] did not provide a solution to optimize it since the paper is focused on multi-class SVMs with dissimilarity. In this section we provide an analytical solution to minimize $E(\boldsymbol{c})$ and discuss when such a solution exists.

$E(\boldsymbol{c})$ is convex in $\boldsymbol{c}$ because each $\left(c_i - s_{ij} c_j\right)^2$ is convex. Therefore, to minimize $E(\boldsymbol{c})$ we only need to find $\boldsymbol{c}^*$ such that

$$\left.\frac{\partial E}{\partial \boldsymbol{c}}\right|_{\boldsymbol{c}=\boldsymbol{c}^*} = 0, \quad (3)$$

under the constraint that $c_1, \dots, c_l$ are fixed to their initial values. We split $\boldsymbol{c}$ into the labeled set $\boldsymbol{c}_l = (c_1, \dots, c_l)$ and the unlabeled set $\boldsymbol{c}_u = (c_{l+1}, \dots, c_n)$. With simple derivations, we can show that Equation (3) is equivalent to

$$\forall i \in \{l+1, \dots, n\}, \ \sum_j |w_{ij}| \cdot c_i - \sum_j w_{ij} c_j = 0. \quad (4)$$

We define the weight matrix $W = [w_{ij}]$, diagonal matrix D such that $D_{ii} = \sum_j |w_{ij}|$, and matrix $P = D^{-1}W$. We split the weight matrix W into four blocks as $W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}$, where W_{xy} is an $x \times y$ matrix. D and P are split similarly. Thus we can rewrite Equation (4) as

$$(D_{uu} - W_{uu})\boldsymbol{c}_u - W_{ul}\boldsymbol{c}_l = 0. \quad (5)$$

Therefore, we can solve

$$\boldsymbol{c}_u = (D_{uu} - W_{uu})^{-1} W_{ul} \boldsymbol{c}_l = (I - P_{uu})^{-1} P_{ul} \boldsymbol{c}_l \,, \quad (6)$$

if $(I - P_{uu})$ is invertible.

In [20] the authors provide the optimal solution to Equation (1), which shares some similarity with our solution, although it does not consider dissimilarity relationships among nodes. A major drawback of the approach in [20] is that it requires $w_{ij} > 0$ for all i, j, in order to guarantee that $(I - P_{uu})$ is invertible. This requirement is impractical for most real-world data sets. A data set with only a hundred thousand facts will have a matrix W with ten billion entries, which is too big to fit in memory. In this paper we work on a Web-scale data set with hundreds of millions of facts, and we have to find a scalable method that can handle sparse matrices and converge to the optimal solution.

We first analyze an example in which $(I - P_{uu})$ is not invertible. If for an unlabeled fact f_k ($k \in (l, n]$), $w_{ik} = w_{ki} = 0$ for any $i \neq k$, then the k^{th} row and k^{th} column of the matrix $(I - P_{uu})$ are 0, resulting in a non-invertible $(I - P_{uu})$. This is not surprising because f_k is not related to any labeled facts either directly or indirectly, and its confidence score will remain undefined. In such cases there is no unique solution that minimizes $E(\boldsymbol{c})$. Any confidence score of f_k yields the same $E(\boldsymbol{c})$, and therefore f_k may get an arbitrary confidence score.

We solve this problem by introducing a "neutral fact" to the set of labeled facts. It has a confidence score of 0 and is connected to every unlabeled fact. Suppose f_1 is the neutral fact and has score $c_1 = 0$. The weight of the edge between f_1 and an unlabeled fact f_i must be above zero, i.e., $w_{1i} = w_{i1} > 0$.

The neutral fact has two important roles. First, it guarantees the existence of a unique solution that minimizes $E(\boldsymbol{c})$, which is proved in Theorem 1 below. If an unlabeled fact is not connected to any labeled facts either directly or indirectly, it will have a confidence score of 0 since it is connected to the neutral fact. Second, the neural fact lowers the confidence scores of unlabeled facts that are only remotely connected to the labeled facts. This is desirable because there are noises in the connections among facts. Thus a long sequence of connections introduces more uncertainty, which should lower our confidence about a fact being true or not. This property will be studied in details in Section 5.3.

The weight on edges from/to the neutral fact can be defined in many ways. We discuss two simple definitions in this paper. The first definition is to use a constant weight:

$$w_{1i} = w_{i1} = \tau, i = l + 1, \dots, n, \quad (7)$$

where $\tau > 0$. The second definition is to assign a weight proportional to the total weight of edges from each node:

$$w_{1i} = w_{i1} = \mu \cdot \textstyle\sum_{j>1} |w_{ij}|, i = l + 1, \dots, n, \quad (8)$$

where μ is a small constant. The first definition is suitable for problems in which the distribution of edges is fairly uniform, i.e., the degrees of the nodes do not differ too much. The second definition is suitable for problems where different nodes have very different degrees, such as web-scale problems where some nodes have millions of edges while many others have only a few edges.

Theorem 1: There exists a unique solution to minimizing $E(\boldsymbol{c})$.

Proof: We first show that $(I - P_{uu})$ is positive-definite. Because $P = D^{-1}W$ and $D_{ii} = \sum_j |w_{ij}|$, we know $\sum_j |P_{ij}| = 1$ for $i = 1, \dots, n$. Because $w_{1i} > 0$, we know $P_{1i} > 0$ for $i = l + 1, \dots, n$. Since P_{uu} is a sub-matrix of P, we know that $\sum_j |[P_{uu}]_{ij}| < 1$ for $i = 1, \dots, n - l$. Let $M = I - P_{uu}$. $\forall x \in \mathbb{R}^{n-l}/\{\mathbf{0}\}$,

$$\begin{aligned} x^T M x &= x^T x - x^T P_{uu} x = \sum_i {x_i}^2 - \sum_{ij} [P_{uu}]_{ij} x_i x_j \\ &> \sum_{ij} |[P_{uu}]_{ij}| {x_i}^2 - \sum_{ij} [P_{uu}]_{ij} x_i x_j \\ &\geq \frac{1}{2} \sum_{ij} |[P_{uu}]_{ij}| \left({x_i}^2 - 2 x_i x_j + {x_j}^2\right) \geq 0 \end{aligned}$$

Therefore, $(I - P_{uu})$ is positive-definite and is thus invertible. As shown in Equation (5), $\boldsymbol{c}_u = (I - P_{uu})^{-1} P_{ul} \boldsymbol{c}_l$ is the unique solution to minimizing $E(\boldsymbol{c})$. ■

5. ITERATIVE COMPUTATION

Although Equation (6) provides an analytical solution to minimizing $E(\boldsymbol{c})$, it is very expensive or impractical to compute. In a real-world truth discovery problem, the number of facts is usually at least tens of thousands, and can even reach hundreds of millions in some problems. It is very expensive or impossible to compute the inverse of a matrix of such size. Sometimes it is even impossible to fully materialize the matrix W. In this section we will discuss how to use an iterative procedure to compute $\boldsymbol{c}_u$ efficiently.

5.1 Iterative Algorithm and Its Convergence

The goal of the iterative procedure is to compute $\boldsymbol{c}_u = (I - P_{uu})^{-1} P_{ul} \boldsymbol{c}_l$ without involving matrix inversion or other expensive operations. We use an iterative procedure similar to that in [19]. The confidence score vector $\boldsymbol{c}$ after t iterations is denoted by $\boldsymbol{c}^t$. We initialize the confidence scores by setting c_i to the labeled data for $i = 1, \dots, l$, and $c_i = 0$ for $i = l + 1, \dots, n$. In this way the initial confidence score vector is $\boldsymbol{c}^0 = (c_1, \dots, c_l, 0, \dots, 0)$. Then we repeat the following steps until $\boldsymbol{c}$ converges.

Step 1: $\boldsymbol{c}^t = P \boldsymbol{c}^{t-1}$

Step 2: Restore the confidence scores for the labeled facts, i.e., set ${c^t}_i = c_i$ for $i = 1, \dots, l$.

It can be shown that the above steps are equivalent to computing

$${\boldsymbol{c}_u}^t = P_{uu} {\boldsymbol{c}_u}^{t-1} + P_{ul} \boldsymbol{c}_l. \quad (9)$$

In order to prove this procedure converges, we need to first provide a bound to the sum of each column in P_{uu}, as shown in Lemma 1.

Lemma 1: $\exists \gamma < 1$, such that $\forall i = 1, \dots, u, \sum_j |[P_{uu}]_{ij}| \leq \gamma$.

Proof: Please recall that $P = D^{-1}W$, and thus

$$\textstyle\sum_j |[P_{uu}]_{ij}| = \frac{\sum_{j=l+1}^{n} |w_{ij}|}{\sum_{j=1}^{n} |w_{ij}|} \leq 1 - \frac{|w_{i1}|}{\sum_{j=1}^{n} |w_{ij}|},$$

where $w_{i1}$is the weight of the edge from f_i to the neutral fact f_1. According to our definitions in Equations (7) and (8), $w_{i1} = \tau$ or $w_{i1} = \mu \cdot \sum_{j>1} |w_{ij}|$. If $w_{i1} = \tau$, let $\omega_{\max} = \max_{1 \leq i \leq n} \left(\sum_{j=1}^{n} |w_{ij}|\right)$ and $\gamma = 1 - \frac{\tau}{\omega_{\max}}$. If $w_{i1} = \mu \cdot \sum_{j>1} |w_{ij}|$, then $1 - \frac{|w_{i1}|}{\sum_{j=1}^{n} |w_{ij}|} = \frac{1}{1+\mu}$, and we let $\gamma = \frac{1}{1+\mu}$. In both cases $\gamma < 1$ and $\sum_j |[P_{uu}]_{ij}| \leq \gamma$. ■

With Lemma 1 we can prove the convergence of our algorithm using the conclusions from [19], which we briefly describe here. It can be easily shown that

$$\lim_{t \to \infty} {\boldsymbol{c}_u}^t = \lim_{t \to \infty} P_{uu}^t {\boldsymbol{c}_u}^0 + \left[\textstyle\sum_{i=1}^{t} P_{uu}^{i-1}\right] P_{ul} \boldsymbol{c}_l. \quad (10)$$

We first study the sum of each column in matrix P_{uu}^t.

$$\textstyle\sum_j [P_{uu}^t]_{ij} = \sum_k [P_{uu}^{t-1}]_{ik} \sum_j [P_{uu}]_{kj} \leq \sum_k [P_{uu}^{t-1}]_{ik} \gamma \leq \gamma^t. \quad (11)$$

Because $\gamma < 1$, $\lim_{t\to\infty} P_{uu}^{t}\boldsymbol{c}_u{}^{0} = 0$, which means the initial point of $\boldsymbol{c}_u$ is inconsequential. It can be easily derived that $\boldsymbol{c}_u = (I - P_{uu})^{-1}P_{ul}\boldsymbol{c}_l$ is a fixed point for function $f(\boldsymbol{x}) = P_{uu}\boldsymbol{x} + P_{ul}\boldsymbol{c}_l$, which is our iterative procedure in Equation (8). It is the unique fixed point because the initial point of $\boldsymbol{c}_u$ is inconsequential. Thus it is the solution to the iterative algorithm.

5.2 Efficient Computation

The iterative procedure presented above converges to the optimal solution and avoids computing matrix inverse. However, in a real-world truth discovery problem there are often millions of facts (e.g., those provided by Wikipedia or IMDB), and thus there are often millions times millions edges in the graph, which makes it impossible to materialize and store the matrices W and P. In this subsection we describe a way to decompose these matrices so that computation can be done in an affordable way.

Let us go over the definition of a truth discovery problem to see how the computation can be simplified. There are n facts $F = \{f_1, \dots, f_n\}$ provided by m data sources $D = \{d_1, \dots, d_m\}$, and let $d(f)$ denote the set of data sources providing fact f. Each fact f is about a subject $s(f)$, and two facts f_1 and f_2 on the same subject may be consistent or in conflict with each other as indicated by $\mathrm{sim}(f_1, f_2)$. The graph of facts is usually built as follows:

1. *Facts on the same subject are connected to each other*: For any f_i and f_j that $s(f_i) = s(f_j)$, $w_{ij} = \mathrm{sim}(f_i, f_j)$.
2. *Facts from the same data source are connected to each other*: If a data source d_k provides both f_i and f_j, it will contribute a certain weight to the edge weight between f_i and f_j. Therefore, for any f_i and f_j that $d(f_i) \cap d(f_j) \neq \emptyset$, $w_{ij} = \alpha \cdot |d(f_i) \cap d(f_j)|$, where $\alpha \in (0,1)$.

Since in each iteration we need to compute

$$\boldsymbol{c}^t = P\boldsymbol{c}^{t-1} = D^{-1}W\boldsymbol{c}^{t-1}, \quad (12)$$

we will decompose both D and W for efficient computation.

As mentioned before, a data source cannot provide multiple facts on same subject, i.e., if $d(f_i) \cap d(f_j) \neq \emptyset$, then $s(f_i) \neq s(f_j)$. Thus matrix W can be decomposed into two sparse matrices without overlapping entries: $W = W_s + W_d$, where $[W_s]_{ij} = \mathrm{sim}(f_i, f_j)$ if $s(f_i) = s(f_j)$ and $[W_d]_{ij} = \alpha \cdot |d(f_i) \cap d(f_j)|$ if $d(f_i) \cap d(f_j) \neq \emptyset$. We also decompose D as $D = D_s + D_d$, where $[D_s]_{ii} = \sum_j |[W_s]_{ij}|$ and $[D_d]_{ii} = \sum_j |[W_d]_{ij}|$.

The number of non-zero entries in W_s is usually small because the number of unique values for each subject is usually small. Therefore, we can store W_s as a sparse matrix and compute D_s from it. In contrast, W_d may contain billions or trillions of non-zero entries because some data sources may provide millions of facts. Thus we have to further decompose W_d. Let V be a $n \times m$ matrix and $V_{ik} = \begin{cases} 1, & \text{if } d_k \in d(f_i); \\ 0, & \text{otherwise.} \end{cases}$ It can be shown that $|d(f_i) \cap d(f_j)| = \sum_{k=1}^{m} V_{ik}V_{jk}$, and thus $W_d = \alpha VV^{\mathrm{T}}$. Therefore,

$$W\boldsymbol{c}^{t-1} = W_s\boldsymbol{c}^{t-1} + \alpha VV^{\mathrm{T}}\boldsymbol{c}^{t-1}, \quad (13)$$

which can be easily computed because W_s is of manageable size, V is part of the input, and $VV^{\mathrm{T}}\boldsymbol{c}^{t-1}$ can be computed by two operations of multiplying a vector by a matrix.

The diagonal matrix D can also be computed efficiently. D_s can be computed from W_s, and D_d can be computed as:

$$[D_d]_{ii} = \alpha \sum_j \sum_{k=1}^{m} V_{ik}V_{jk} = \alpha \sum_{k=1}^{m} V_{ik}\left(\sum_j V_{jk}\right). \quad (14)$$

Let $|d_k|$ be the number of facts provided by d_k. Obviously $|d_k| = \sum_j V_{jk}$, and thus $[D_d]_{ii} = \alpha \sum_{k=1}^{m} V_{ik}|d_k|$. In this way D_s and D_d can be pre-computed, and we can easily compute $\boldsymbol{c}^t = D^{-1}W\boldsymbol{c}^{t-1}$. Since the only operation involved in each iteration is multiplying a vector by a sparse matrix, we easily implement this algorithm with MapReduce and run it in a distributed framework.

5.3 Complexity Analysis

Here we analyze the complexity of the algorithm presented in Section 5.2. Suppose there are n facts and m data sources. Let l be the total number of cases of a data source providing a fact, i.e., there are l non-zero entries in matrix V. Suppose for each fact f, on average there are q facts on the same subject as f.

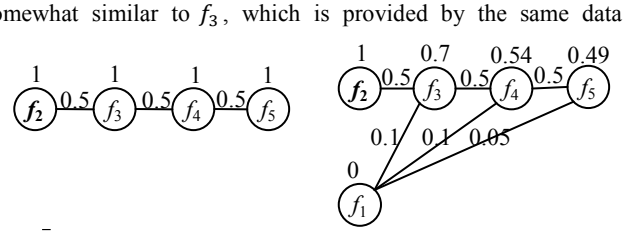
We discuss the complexity of computing Equations (13) and (14). W_s is a $n \times n$ matrix and there are $O(nq)$ non-zero entries in it. It takes $O(nq)$ time to compute W_s and $W_s\boldsymbol{c}^{t-1}$. It takes $O(l)$ time to compute $VV^{\mathrm{T}}\boldsymbol{c}^{t-1}$. Therefore, it takes $O(nq + l)$ time to compute $W\boldsymbol{c}^{t-1}$ in each iteration. We also need to compute matrix D which has two parts: D_s and D_d. D_s can be directly computed from W_s in $O(nq)$ time. To compute D_d as in Equation (14), we first compute $\sum_j V_{jk}$ for $k = 1, \dots, m$, and then iterate through the l non-zero entries in V to compute D_d, which takes $O(l)$ time in total. In summary, the time complexity of our algorithm is $O((nq + l)t)$ for t iterations.

5.4 Decay of Confidences in Propagation

It is mentioned before that the neutral fact is important to our algorithm as it guarantees the existence of a unique solution. In Section 5.1 we can see it is also important to the convergence of our iterative algorithm. In this subsection we further study how the neutral fact influences our algorithm, and show that it is equivalent to introducing a small decay to the confidence scores of facts in each iteration.

First let us compare two versions of algorithms, one without the neutral fact and one with it. We start from a simple example. Suppose there is one labeled fact f_2 with confidence score 1, and three unlabeled facts f_3, f_4, f_5. We created two graphs as shown in Figure 3: Graph $\bar{G}$ without the neutral fact and G with the neutral fact f_1. The weights of edges to and from the neutral fact is defined by Equation (8) with $\mu = 0.1$. The weights of edges and confidence scores are shown in the figure.

In order to minimize $E(\boldsymbol{c})$, in $\bar{G}$ the confidence scores of f_3, f_4, and f_5 should all be set to 1. In fact, in any graph where all labeled facts have confidence scores of 1 and there is no negative edge, any unlabeled fact connected to any labeled facts will have score of 1, no matter how far away it is from the labeled facts. Such assignment of scores is not reasonable because we have different confidences in the correctness of these facts. For example, f_5 may be provided by the same data source as f_4, which is somewhat similar to f_3, which is provided by the same data



(a) $\bar{G}$, without neutral fact (b) G, with neutral fact

Figure 3: Graphs without and with the neutral fact

source as f_2. Since we know f_2 is true, we are pretty confident that f_3 is true, somewhat confident for f_4, but not that confident with f_5, because each hop introduces uncertainty.

In order to model such uncertainty and the resulting decrease in confidence, we introduce the concept of *propagation decay*, and we will show such decay has exactly the same effect as adding the neutral fact. In the discussion below we will compare the computation in $\bar{G}$ and G, using $\bar{D}$, $\bar{S}$, $\bar{W}$, $\bar{P}$ and $\bar{\boldsymbol{c}}$ to represent the matrices and vectors in $\bar{G}$.

Let us consider the computation in $\bar{G}$ which has no neutral fact. As shown in Section 5.1, in each iteration we are propagating the confidence scores with equation $\bar{\boldsymbol{c}}^t = \bar{P}\bar{\boldsymbol{c}}^{t-1}$, i.e., propagating the confidence score from each node to its neighbors using the matrix $\bar{P}$. Now we introduce some decay in each iteration, defined as follows.

Definition 1. (Propagation Decay) In Step 1 of each iteration, when propagating confidence scores from a labeled fact f_i to an unlabeled fact f_j, we add the score $\rho\bar{P}_{ij}\bar{c}^{t-1}{}_i$ to $\bar{c}^t{}_j$, instead of $\bar{P}_{ij}\bar{c}^{t-1}{}_i$, where $\rho \in (0,1)$ is they decay factor. This can also be written as $\bar{\boldsymbol{c}}^t = \rho\bar{P}\bar{\boldsymbol{c}}^{t-1}$**.** ■

Here we will show that adding a propagation decay is equivalent to adding a neutral fact in the graph.

Theorem 2: Let $\bar{\boldsymbol{c}}_u{}^t$ be the confidence score vector of unlabeled facts in the graph $\bar{G}$ without a neutral fact after t iterations with propagation decay. Let $\boldsymbol{c}_u{}^t$ be the confidence score vector in graph G with a neutral fact but without propagation decay (weight of edges to/from the neutral fact is set as in Equation (8)). Then

$$\boldsymbol{c}_u{}^t = \rho\bar{\boldsymbol{c}}_u{}^t \text{ if } \rho = \frac{1}{(1+\mu)}. \tag{14}$$

Proof: We first look at the computation in G. In each iteration we compute $\boldsymbol{c}^t = P\boldsymbol{c}^{t-1}$, which can be rewritten as $\begin{bmatrix}\boldsymbol{c}_l{}^t \\ \boldsymbol{c}_u{}^t\end{bmatrix} = \begin{bmatrix}P_{ll} & P_{lu} \\ P_{ul} & P_{uu}\end{bmatrix}\begin{bmatrix}\boldsymbol{c}_l{}^{t-1} \\ \boldsymbol{c}_u{}^{t-1}\end{bmatrix}$. Because we restore $\boldsymbol{c}_l$ to its original value after each iteration, the computation in each iteration is actually $\boldsymbol{c}_u{}^t = P_{uu}\boldsymbol{c}_u{}^{t-1} + P_{ul}\boldsymbol{c}_l$. With induction we can easily prove that $\boldsymbol{c}_u{}^t = P_{uu}^t\boldsymbol{c}_u{}^0 + \left[\sum_{i=1}^t P_{uu}^{i-1}\right]P_{ul}\boldsymbol{c}_l$. Because we set $\boldsymbol{c}_u{}^0 = \mathbf{0}$, we have

$$\boldsymbol{c}_u{}^t = \left[\sum_{i=1}^t P_{uu}^{i-1}\right]P_{ul}\boldsymbol{c}_l. \tag{15}$$

Now we analyze how the neutral fact influences P_{uu} and P_{ul}. Remember that $P = D^{-1}W$. Since $D_{ii} = \sum_j |w_{ij}|$, $\bar{D}_{ii} = \sum_{j>1}|w_{ij}|$, and $w_{1i} = w_{i1} = \mu \cdot \sum_{j>1}|w_{ij}|$, we know that $D_{ii} = (1+\mu)\bar{D}_{ii}$ and thus $D_{uu} = (1+\mu)\bar{D}_{uu}$. From the definition of P_{ul} we know

$$P_{ul}\boldsymbol{c}_l = D_{uu}{}^{-1}W_{ul}\boldsymbol{c}_l$$

Because W_{ul} only differs with $\bar{W}_{ul}$ in the first column, and $\boldsymbol{c}_l = \bar{\boldsymbol{c}}_l$ and $\boldsymbol{c}_{l_1} = \bar{\boldsymbol{c}}_{l_1} = 0$**,** we have

$$W_{ul}\boldsymbol{c}_l = \bar{W}_{ul}\bar{\boldsymbol{c}}_l$$
$$\Rightarrow P_{ul}\boldsymbol{c}_l = \left((1+\mu)\bar{D}_{uu}\right)^{-1}\bar{W}_{ul}\bar{\boldsymbol{c}}_l = \frac{1}{(1+\mu)}\bar{P}_{ul}\bar{\boldsymbol{c}}_l$$

Because $W_{uu} = \bar{W}_{uu}$, we have $P_{uu} = \frac{1}{(1+\mu)}\bar{P}_{uu}$. Therefore,

$$\boldsymbol{c}_u{}^t = \frac{1}{(1+\mu)}\left[\sum_{i=1}^t \left(\frac{1}{(1+\mu)}\bar{P}_{uu}\right)^{i-1}\right]\bar{P}_{ul}\bar{\boldsymbol{c}}_l. \tag{16}$$

When iterating with propagation decay in $\bar{G}$, in each iteration we are computing $\bar{\boldsymbol{c}}_u{}^t = \rho\bar{P}_{uu}\bar{\boldsymbol{c}}_u{}^{t-1} + \bar{P}_{ul}\bar{\boldsymbol{c}}_l$. Similar to Equation (9) we can prove that $\bar{\boldsymbol{c}}_u{}^t = \left[\sum_{i=1}^t (\rho\bar{P}_{uu})^{i-1}\right]\bar{P}_{ul}\bar{\boldsymbol{c}}_l$, which is similar to Equation (16). If we let $\rho = \frac{1}{(1+\mu)}$, then $\boldsymbol{c}_u{}^t = \rho\bar{\boldsymbol{c}}_u{}^t$. ■

Theorem 2 shows that adding a neutral fact achieves the same effect as performing propagation decay in each iteration. Thus it is unnecessary to perform such decay when using the neutral fact. This is the second role of neutral fact (the first role is to guarantee the existence of a unique solution and convergence of the iterative algorithm), which is also very important to our approach.

6. EXPERIMENT RESULTS

We test our approach SSTF on six real-world data sets, including the data set containing book authors used in [16] and [6], four data sets from HTML tables on the web for entities of certain types, and a huge data set containing hundreds of millions of entity-attribute-value triples extracted from HTML tables all over the web. Small-scale experiments are performed on a PC with Intel Quad-Core 2.66GHz CPU, 32GB memory. Web-scale experiments are performed on a PC cluster based on Dryad [12] that supports MapReduce. We also test the scalability of our approach on synthetic data sets at different scales.

6.1 Book Authors Data Set

The first experiment is based on a real-world data set containing authors of computer science books, which is the only real-world data set used in [16] and [6]. This data set is extracted from AbeBooks.com. Each book is listed on a set of online bookstores, each providing the authors of the book. The goal is to find the correct list of authors for each book. There is a ground truth set containing the authors of 100 randomly selected books, created by manually looking at the images of the book covers. This data set contains 1263 books and 24364 listings from 877 bookstores.

Both [16] and [6] provide detailed experiment results on this data set, although using different evaluation criteria. We adopt the criteria of [6] so we can directly compare with its results. For each book, the author names are normalized into a list of names, where duplicate names are removed and middle names are ignored. The author names of a book are considered to be correct if and only if they exactly match with the ground truth after normalization. Cases such as additional, missing, mis-ordered and misspelled names are all considered incorrect.

We use the definition of similarity between two sets of authors from [16][1]. The parameters of SSTF are set as follows. The weight of an edge between two facts from same data source is set to $\alpha = 0.01$. The weight of an edge from the neutral fact to each fact is 1. After each iteration, we compute the relative change of the confidence score vector as $\|\boldsymbol{c}^t - \boldsymbol{c}^{t-1}\|/\|\boldsymbol{c}^{t-1}\|$**,** and the iterative procedure stops if the relative change is less than 0.01 after any iteration.

We first test how fast SSTF converges and how its accuracy changes over iterations. Figure 4 shows the accuracy of our approach SSTF (Semi-supervised Truth Finder), as we vary the amount of training data from 12.5% to 87.5% of the 100 labeled examples. *n*-fold validation is used in each experiment. For example, 8-fold validation is used when using 12.5% of labeled examples as training data, which means 12 training examples are used in four folds and 13 used in the other four folds. We can see the accuracy improves over iterations most of the time. The final

[1] The similarity definition from [16] is asymmetric and is in the range [0, 1]. To use it in our approach, we take the average of the similarities in both directions and scale it to [–1, 1].

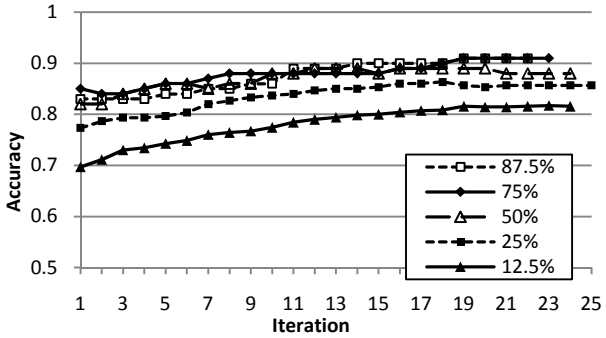


Figure 4: Accuracy of SSTF with different training set sizes

accuracy w.r.t. the training set size is shown in Table 1. SSTF achieves accuracy of 91% with only 75 training examples.

Table 1: Accuracy of SSTF w.r.t. Size of the Training Set

Avg. #Training examples	12.5	25	50	75	87.5
#Fold	8	4	2	4	8
Accuracy	.816	.857	.880	.910	.910

Figure 5 shows the relative change in the confidence score vector after each iteration. We can see that SSTF converges with a steady and reasonably fast pace. Figure 6 shows the sensitivity of SSTF to different parameter values, where we vary the weight of an edge to/from the neutral fact from 0.1 to 5 and α from 0.001 to 0.05. We can see that SSTF is not sensitive to changes in its parameters.

We compare our approach (with 75 training examples) to the following algorithms: (1) Voting, which considers the fact provided by most data sources as the true fact (a fact is randomly chosen in case of a tie); (2) TruthFinder as described in [16]; (3) Accu as described in [6]; (4) AccuWithSim as described in [6]; (5) 2-Estimates, which is described in [10] and has the highest accuracy among the methods in [10]. Because we use the same data set and evaluation criteria as in [6], we simply report their results of Accu and AccuWithSim. The other methods are implemented according to their papers. Table 2 shows the accuracy, number of iterations used, and total running time for each approach. (The running time of Accu and AccuWithSim are from [6], which may be using a less powerful computer, as the reported running time of TruthFinder in [6] is four times of that in our experiment.) Our approach, SSTF, achieves higher accuracy than existing approaches, especially when compared to TruthFinder, which does not detect data copying behaviors (and neither does SSTF). This experiment shows that SSTF can significantly improve accuracy with a small training set. It is also significantly

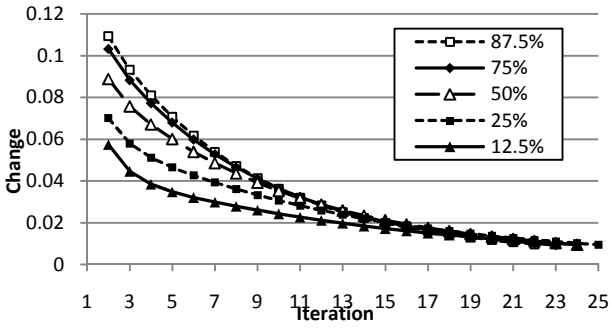


Figure 5: Changes after each iteration of SSTF

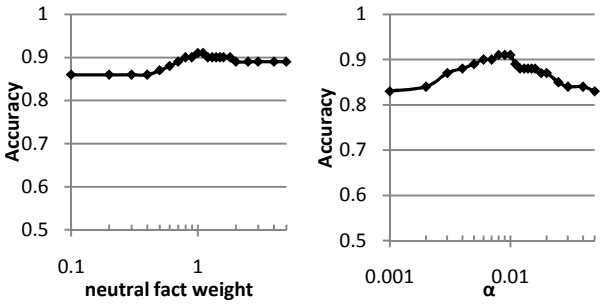


Figure 6: Parameter sensitivity of SSTF

faster than many existing approaches.

Table 2: Accuracy on the Book Authors data set

Approach	*Accuracy*	*#Iteration*	*Time (s)*
Voting	0.71	1	1.3
TruthFinder	0.83	5	2.9
Accu	0.87	22	185.8 ([6])
AccuWithSim	0.89	18	197.5 ([6])
2-Estimates	0.73	29	21.2
SSTF	0.91	23	7.9

6.2 Web Data Sets

6.2.1 Data Collection

It has been observed that HTML tables on the web provide a huge number of facts, though they contain much noise [4]. We extract facts from tables and use our approach to distinguish true facts from false ones. As in [4], we focus on attribute-value tables, each of which contains one column of attribute names and another column of values, with two examples shown in Figure 7. Such tables widely exist on the web and usually provide popular facts for each entity, making them the best subjects for truth discovery.

The following method is used to extract facts from HTML tables. We build a table classifier using the approach from [2], and train it with a manually labeled set of attribute-value tables. With this classifier, we extract 744M attribute-value tables from 20B web pages in Bing's index on 2010/06/22.

However, as these attribute-value tables are not associated with any entity, we use the following method to find the main entity that each web page talks about. The main entity of a web page can often be found by matching user queries leading to a click on this page with the page content. For example, a user may search for "Britney Spears music" and click on http://www.last.fm/music/Britney+Spears, whose title is "Britney Spears – Discover music, videos, concerts, & pictures at Last.fm". We find the longest common substring between the query and the title, which is "Britney Spears", the subject of this page. In addition, for each query and clicked web page, we try to match the query with the text in each <h1> element. If no match is found, it tries to match with each <h2> element, and so on, until it finds a match or has tried each element in the page. The matched part is considered as a candidate of the main entity of the page. For each candidate we build a wrapper based on HTML tag-paths [13]. For example, the wrapper for the above page from Last.fm is "<html><head><title>(*) – Discover music, videos, concerts, & pictures at Last.fm".

In order to select good wrappers and use them to extract entity names, we utilize the fact that many websites contain large num-

(a) A table in http://en.wikipedia.org/wiki/Tom_Hanks

(b) A table in http://www.celebrina.com/tom-hanks.html

Figure 7: Two examples of attribute-value tables

bers of web pages in the same format (e.g., all movie pages on IMDB). If we can build a wrapper for extracting the main entity from some pages, we can extract entities from other pages of same format. We use the approach in [17] to find sets of web pages in the same format. For each such set of pages, we choose the wrapper that extracts entities from the most pages that match the user queries. This wrapper will be used to extract the main entity from every page in that set.

We use all query-click logs from the U.S. market between 2008/08/01 and 2009/05/31, which contains each search query and all URLs clicked for it. Based on these queries and the 20B web pages in Bing's index on 2010/06/22, we extract the main entities from 93.3M pages with attribute-value tables, which have 164M tables in total. These entities are joined with the attribute-values extracted from the tables on these pages. 749M entity-attribute-value triples are created, which will be used in our experiment.

We extract the data from Wikipedia page titles and infobox tables (e.g., the table in Figure 7 (a)) and use them as ground truth data. We want to remove all web sites getting majority of their data from Wikipedia, in order to perform a fair comparison between our approach and unsupervised approaches. For any web site with at least half of its data identical to some Wikipedia data, we consider it as a "Wikipedia copier" and remove it from our data set. Although this may falsely remove some websites, we can be sure that the remaining web sites are getting the majority of their data from sources other than Wikipedia.

6.2.2 Domain-Specific Experiments

We first test our approach on four data sets in special domains and compare it with existing approaches. The four data sets are directors of American films, developers (i.e., studios) of video games, governors of U.S. states, and presidents of universities. We collect the four sets of entities from special Wikipedia categories, as shown in Table 3, where all entities with disambiguation pages are removed (except U.S. states). Then we collect the values on the specified attribute from the HTML table data set, and create four fact sets as described in Table 4. Please note entities without the specific attributes provided by any site are ignored.

SSTF uses the same parameters as in the book authors data set, and all other approaches are implemented according to their papers. The similarity function for directors of films is the same as that in the book authors data set. In the other three data sets the similarity function between two string values is defined as their edit-distance divided by the sum of their textual lengths (the similarity is scaled to [–1, 1] for SSTF). We ignore all ground truth facts that are not provided by any other web sites. 4-fold experiments are used for SSTF, with 75% of ground truth facts allocated

Table 3: Data sources for four classes of entities

Class of entity	*Num. Entity*	*Wikipedia categories*
American films	8772	*_american_films
video games	5110	*_video_games
U.S. states	50	states_of_the_united_states
universities	7191	universities_and_colleges_*

Table 4: Four data sets from HTML tables

Data set	*#entity*	*#provider*	*#facts*	*#distinct facts*	*#ground truth fact*
directors of films	4893	370	31154	7132	4105
developer of video games	3418	181	16961	8187	1911
governors of states	50	70	1195	312	49
presidents of universities	543	64	1890	756	254

as training data and 25% for testing in each fold. All other approaches use all ground truth facts for testing.

SSTF is compared to Voting, TruthFinder, AccuWithSim and 2-Estimates. Accuracy of an approach is defined as the percentage of entities for which the correct fact is selected. All algorithms converge on all data sets, except AccuWithSim which oscillates on three of the four data sets. When it oscillates among multiple states, we use the average accuracy of these states as its accuracy. Figure 8 shows the accuracies of each approach on these data sets. SSTF achieves accuracies between 78% to 96%, and it is significantly more accurate than the other approaches on all data sets except directors of films. On governors of U.S. states and presidents of universities it beats all other approaches by at least 10%. On directors of films the accuracies of different approaches are very close, with Vote and 2-Estimates being the most accurate and SSTF 0.4% behind. Figure 9 shows the running time of the five approaches on the four data sets.

6.2.3 Web-scale Truth Discovery

In this subsection we present an experiment that involves all 749M facts extracted from HTML tables as described in Section 6.2.1. A web site often provides facts about many attributes for many entities. If a web site provides correct values for an attribute on some entities (e.g., date of birth of some people), we can expect it to provide correct values on this attribute for other entities. Therefore, we treat each web site and each attribute as a data source. Similarly we also treat each web site and each entity as a data source. We do not consider two facts with different entities and attributes to be from the same data source, because a web site may have very different trustworthiness on different attributes or different entities.

There are 65.7M entities, 749M facts (591M distinct facts) from 33K websites. According to our definition above, there are 89M data sources, where 15.5M of them are website-attribute pairs and 73.5M are website-entity pairs. Some data sources provide very large numbers of facts (as many as 5.2M), while on average each data source only provides 8.42 facts. The numbers of edges from different facts vary greatly. Thus we define the weight of an edge to/from the neutral fact according to Equation (7): $w_{1i} = w_{i1} = \mu \cdot \sum_{j>1} |w_{ij}|$, where $\mu = 0.1$. The edge weight between two facts from the same data source is set to 0.5, since facts from each data source are highly homogenous: They are either about the same entity or the same attribute.

Again the facts from Wikipedia infobox tables are used as ground truth. There are three runs of our approach: One uses all

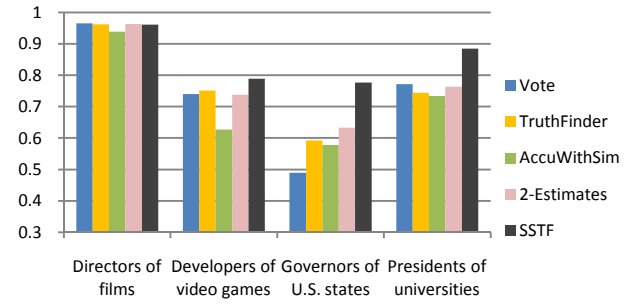


Figure 8: Accuracies of five approaches on four data sets

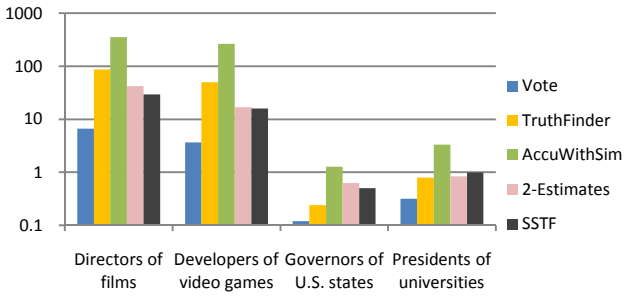


Figure 9: Running time (s) of five approaches on four data sets

ground truth facts for training (30.7M facts), one uses 10% of them, and the last uses 1% of them.

Instead of continuing using Wikipedia to measure the accuracy of our approach, we measure that based on whether such data is useful in answering queries containing an entity and an attribute, which better reflects the usefulness of such data for web search. For example, if the user searches for {Barack Obama date of birth} and our data set contains some values for the entity "Barack Obama" and attribute "date of birth", we will measure if the corresponding values are correct.

First we need to create a test set such that facts appearing more often in user queries have a larger chance to be selected. Our fact set is tail-heavy where most facts, such as price of a product or version of some software, are uninteresting. In order to avoid selecting many trivial facts for evaluation, we only consider facts that appear in the Bing web search queries between 2008/08/01 and 2009/05/31. A fact is consider to be in the query set if there is a query consisting of the entity name and attribute name. Each fact is weighted by the frequency of the corresponding query (i.e., number of times the query is submitted to Bing). We randomly select 1000 facts with the probability each fact is selected proportional to its weight. We exclude 20 facts that appear in the ground truth set. We also exclude 120 facts whose values are long textual contents such as the biography of people and symptoms of diseases, because it is hard to judge if such facts are correct or not.

We manually label each fact as true or false according to the semantic meaning of the corresponding query using the following method. Given a fact which is an entity-attribute-value triple, we need to first decide the identity of its entity, as the entity name for some facts can be very ambiguous (e.g., "system" as the entity). We consider the entity with the specified name in the first result page of Bing as the true entity. Let us consider the fact "Brazil: Time zone = UTC -2 to -5" (i.e., entity is "Brazil", attribute is "Time zone" and value is "UTC -2 to -5"). We consider the entity to be the country of Brazil which is the main entity in the first result page of Bing for query "Brazil time zone", and thus this fact is true. The fact "System: Security = Basic" is false because the first result page of the query "system security" is not about the entity "system". The second requirement for a fact to be true is that there exists an entity with the specified attribute and value, as some facts are simply wrong, such as "Sony: Camera = Canon 300D". Among the 860 labeled facts, 68 are true and 792 are false. Please note that most of the false facts are simply not facts, such as "baby: games = 32", "Comcast: email = enter missing info" and "myspace: comments = add". They are extracted because the HTML table classifier has limited accuracy.

We compare SSTF with TruthFinder [16]. We implement SSTF with MapReduce according to Section 5.2, and implement TruthFinder with MapReduce as well. The approach in [6] is not MapReduceable as it involves sorting. Therefore we do not include it in our evaluation.

Each approach assigns a confidence score to each fact. The accuracy is defined as the probability of a true fact having a higher score than a false fact. Let $l(f_i)$ and c_i be the label and confidence score of fact f_i, respectively. The accuracy of a confidence score assignment is defined as follows.

$$\text{Accuracy}(\boldsymbol{c}) = \frac{\sum_{l(f_i)=T,l(f_j)=F} I(c_i > c_j) + 0.5 \cdot \sum_{l(f_i)=T,l(f_j)=F} I(c_i = c_j)}{\left|\{(f_i, f_j) | l(f_i) = T, l(f_j) = F\}\right|}$$

In the rare case of $c_i = c_j$ we consider it to be half-correct.

Figure 10 compares the accuracy of TruthFinder and SSTF with varying amounts of training data. We can see SSTF achieves an accuracy of 83%, which is much higher than TruthFinder. The accuracy of SSTF is not significantly affected by training data size. It achieves an accuracy of 81% when only 1% of Wikipedia infobox facts are used as training data.

Figure 11 shows the relative changes in the confidence score

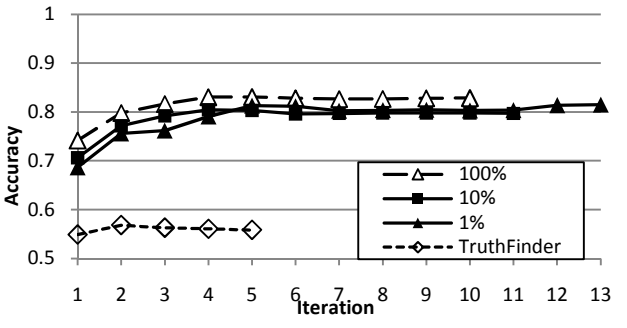


Figure 10: Accuracy of SSTF and TruthFinder on Attribute-value Table data

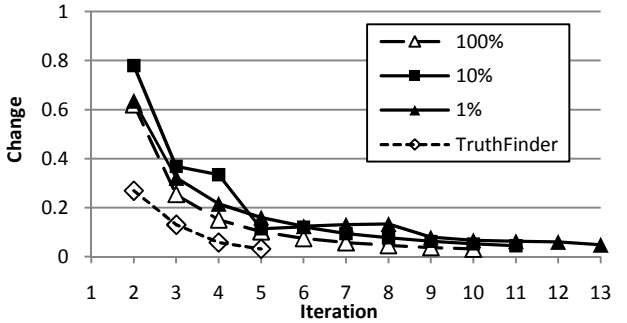


Figure 11: Changes after each iteration of SSTF with different amounts of training data and TruthFinder

vector after each iteration. Because of the high computation cost in processing hundreds of millions of facts, we stop iterating after the change falls below 0.05, and SSTF stops after 10 to 13 iterations. The precision/recall curve of SSTF and TruthFinder are shown in Figure 12. Considering only 7.9% of the facts are correct, SSTF achieves reasonably high accuracy, and is much more accurate than TruthFinder. Table 5 shows the running time, #CPUs used and bytes read/write of SSTF with all training data.

Table 5: Runtime of SSTF with MapReduce

	Time (min)	#CPU	#bytes read/write
Initialization	125	152	3.36T
Each iteration	56.3	184	4.80T

6.3 Scalability on Synthetic Data Sets

We test the scalability of our approach on synthetic data sets. Each data set contains 1000 websites that provide n facts in total on $n/5$ subjects. The trustworthiness of each website is uniformly sampled from [0, 1], with the true value of each subject uniformly sampled from [1000, 10000]. The probability that a fact is set to the true value is given by the trustworthiness of the website. If a fact is false, it is a random number that deviates at most by 50% from the true value. We perform a 5-fold experiment on each data set, using 80% of data for training and 20% for testing. The running time and memory usage of SSTF are shown in Figure 13. Running time grows by 150 times when number of facts grows by 100 times, while memory usage grows by 42 times. The accuracy is always above 95%.

7. CONCLUSIONS

As online sources often provide inaccurate and conflicting information, truth discovery has become an important research problem. Existing studies all employ unsupervised approaches, which are often ineffective as it is sometimes very difficult to distinguish between true and false facts using only the data itself. In this paper we propose a semi-supervised approach that finds true values with the help of a small amount of ground truth data. Unlike existing studies that only provide iterative algorithms, we derive the optimal solution and provide an efficient iterative algorithm that converges to it. Experiments show our method achieves higher accuracy than existing approaches and can be applied on very large data sets.

8. REFERENCES

[1] J. Bleiholder and F. Naumann. Conflict handling strategies in an integrated information system. *WWW'06*.

[2] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu and Y. Zhang. WebTables: Exploring the Power of Tables on the Web. *VLDB'08*.

[3] A. Celikyilmaz, M. Thint, Z. Huang. A graph-based semi-supervised learning for question answering. *IJCNLP'09*.

[4] E. Crestan and P. Pantel. Web-scale knowledge extraction from semi-structured tables. *WWW'10*.

[5] X. L. Dong, L. Berti-Equille, Y. Hu and D. Srivastava. Global detection of complex copying relationships between sources. In *VLDB'10*.

[6] X. L. Dong, L. Berti-Equille and D. Srivastava. Integrating conflicting data: The role of source dependence. *VLDB'09*.

[7] X. L. Dong, L. Berti-Equille and D. Srivastava. Truth discovery and copying detection in a dynamic world. *VLDB'09*.

[8] X. L. Dong. Presentation for [6]. http://www2.research.att.com/~lunadong/talks/depenDetection.pptx

[9] A. Enright. Consumers trust information found online less than offline messages. *Internet Retailer, Aug 25, 2010*.

[10] A. Galland, S. Abiteboul, A. Marian and P. Senellart. Corroborating information from disagreeing views. *WSDM'10*.

[11] A. B. Goldberg, X. Zhu and S. Wright. Dissimilarity in graph-based semi-supervised classification. *AISTATS'07*.

[12] M. Isard, M. Budiu, Y. Yu, A. Birrell and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *Operating Systems Review*, 41(3), 2007.

[13] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires and L. E. Moser. Extracting data records from the web using tag path clustering. *WWW'09*.

[14] J. Tang, H. Li, Q.-J. Qi and T.-S. Chua. Integrated graph-based semi-supervised multiple/single instance learning framework for image annotation. *ACM Multimedia'08*.

[15] M. Wu and A. Marian. Corroborating answers from multiple web sources. *WebDB'07*.

[16] X. Yin, J. Han and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *KDD'07*.

[17] X. Yin, W. Tan, X. Li and Y.-C. Tu. Automatic Extraction of Clickable Structured Web Contents for Name Entity Queries. *WWW'10*.

[18] D. Zhou, O. Bousquet, T.N. Lal, J. Weston and B. Schölkopf. Learning with local and global consistency. *NIPS'04*.

[19] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. CMU Technical Report CMU-CALD-02-107, 2002.

[20] X. Zhu, Z. Ghahramani and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. *ICML'03*.

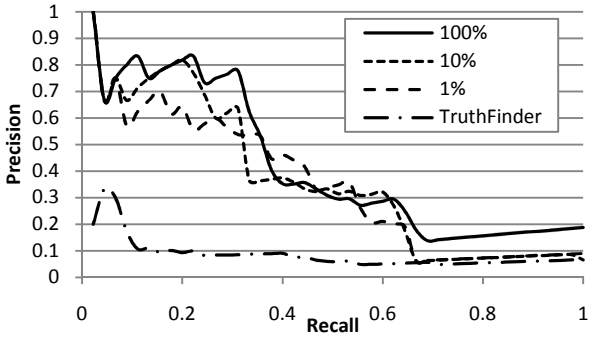


Figure 12: Precision and recall of SSTF and TruthFinder

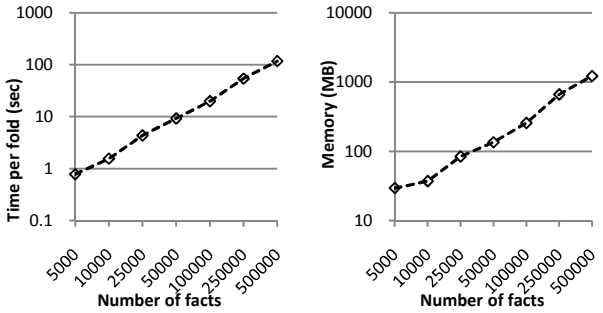


Figure 13: Runtime and accuracy of SSTF on synthetic data