Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang
Wilko Horn, Camillo Lugaresi, Shaohua Sun, Wei Zhang
Google Inc.
{lunadong|gabr|kpmurphy|vandang|wilko|camillol|sunsh|weizh}@google.com

ABSTRACT

The quality of web sources has been traditionally evaluated using *exogenous* signals such as the hyperlink structure of the graph. We propose a new approach that relies on *endogenous* signals, namely, the correctness of factual information provided by the source. A source that has few false facts is considered to be trustworthy.

The facts are automatically extracted from each source by information extraction methods commonly used to construct knowledge bases. We propose a way to distinguish errors made in the extraction process from factual errors in the web source per se, by using joint inference in a novel multi-layer probabilistic model.

We call the trustworthiness score we computed *Knowledge-Based Trust (KBT)*. On synthetic data, we show that our method can reliably compute the true trustworthiness levels of the sources. We then apply it to a database of 2.8B facts extracted from the web, and thereby estimate the trustworthiness of 119M webpages. Manual evaluation of a subset of the results confirms the effectiveness of the method.

1. INTRODUCTION

> *"Learning to trust is one of life's most difficult tasks."*
> – Isaac Watts.

Quality assessment for web sources[1] is of tremendous importance in web search. It has been traditionally evaluated using exogenous signals such as hyperlinks and browsing history. However, such signals mostly capture how popular a webpage is. For example, the gossip websites listed in [16] mostly have high PageRank scores [4], but would not generally be considered reliable. Conversely, some less popular websites nevertheless have very accurate information.

In this paper, we address the fundamental question of estimating how trustworthy a given web source is. Informally, we define the trustworthiness or *accuracy* of a web source as the probability that it contains the correct value for a fact (such as Barack Obama's nationality), assuming that it mentions any value for that fact. (Thus we do not penalize sources that have few facts, so long as they are correct.)

We propose using *Knowledge-Based Trust (KBT)* to estimate source trustworthiness as follows. We extract a plurality of facts from many pages using information extraction techniques. We then jointly estimate the correctness of these facts and the accuracy of the sources using inference in a probabilistic model. Inference is an iterative process, since we believe a source is accurate if its facts are correct, and we believe the facts are correct if they are extracted from an accurate source. We leverage the redundancy of information on the web to break the symmetry. Furthermore, we show how to initialize our estimate of the accuracy of sources based on authoritative information, in order to ensure that this iterative process converges to a good solution.

The fact extraction process we use is based on the *Knowledge Vault* (KV) project [10]. KV uses 16 different information extraction systems to extract (subject, predicate, object) *knowledge triples* from webpages. An example of such a triple is *(Barack Obama, nationality, USA)*. A subject represents a real-world entity, identified by an ID such as *mid*s in *Freebase* [2]; a predicate is predefined in *Freebase*, describing a particular attribute of an entity; an object can be an entity, a string, a numerical value, or a date.

The facts extracted by automatic methods such as KV may be wrong. One method for estimating if they are correct or not was described in [11]. However, this earlier work did not distinguish between factual errors on the page and errors made by the extraction system. As shown in [11], extraction errors are far more prevalent than source errors. Ignoring this distinction can cause us to incorrectly distrust a website.

Another problem with the approach used in [11] is that it estimates the reliability of each webpage independently. This can cause problems when data are sparse. For example, for more than one billion webpages, KV is only able to extract a single triple (other extraction systems have similar limitations). This makes it difficult to reliably estimate the trustworthiness of such sources. On the other hand, for some pages KV extracts tens of thousands of triples, which can create computational bottlenecks.

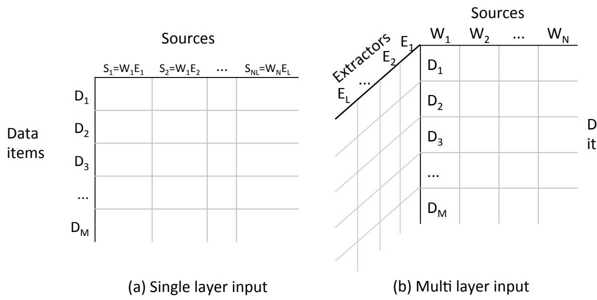
The KBT method introduced in this paper overcomes some of these previous weaknesses. In particular, our contributions are three-fold. Our main contribution is a more sophisticated probabilistic model, which can distinguish between two main sources of errors: incorrect facts on a page, and incorrect extractions made by an extraction system. This provides a much more accurate estimate of the source reliability. We propose an efficient, scalable algorithm for performing inference and parameter estimation in the proposed probabilistic model (Section 3).

[1] We use the term "web source" to denote a specific webpage, such as `wiki.com/page1`, or a whole website, such as `wiki.com`. We discuss this distinction in more detail in Section 4.

Articles from this volume were invited to present their results at the 41st International Conference on Very Large Data Bases, August 31st - September 4th 2015, Kohala Coast, Hawaii.
Proceedings of the VLDB Endowment, Vol. 8, No. 9
Copyright 2015 VLDB Endowment 2150-8097/15/05.

Table 1: Summary of major notations used in the paper.

Notation	Description
$w \in \mathcal{W}$	Web source
$e \in \mathcal{E}$	Extractor
d	Data item
v	Value
X_{ewdv}	Binary indication of whether e extracts (d, v) from w
X_{wdv}	All extractions from w about (d, v)
X_d	All data about data item d
X	All input data
C_{wdv}	Binary indication of whether w provides (d, v)
T_{dv}	Binary indication of whether v is a correct value for d
V_d	True value for data item d under single-truth assumption
A_w	Accuracy of web source w
P_e, R_e	Precision and recall of extractor e



(a) Single layer input

(b) Multi layer input

Figure 1: Form of the input data for (a) the single-layer model and (b) the multi-layer model.

Our second contribution is a new method to adaptively decide the granularity of sources to work with: if a specific webpage yields too few triples, we may aggregate it with other webpages from the same website. Conversely, if a website has too many triples, we may split it into smaller ones, to avoid computational bottlenecks (Section 4).

The third contribution of this paper is a detailed, large-scale evaluation of the performance of our model. In particular, we applied it to 2.8 billion triples extracted from the web, and were thus able to reliably predict the trustworthiness of 119 million webpages and 5.6 million websites (Section 5).

We note that source trustworthiness provides an additional signal for evaluating the quality of a website. We discuss new research opportunities for improving it and using it in conjunction with existing signals such as PageRank (Section 5.4.2). Also, we note that although we present our methods in the context of knowledge extraction, the general approach we propose can be applied to many other tasks that involve data integration and data cleaning.

2. PROBLEM DEFINITION AND OVERVIEW

In this section, we start with a formal definition of *Knowledge-based trust* (KBT). We then briefly review our prior work that solves a closely related problem, *knowledge fusion* [11]. Finally, we give an overview of our approach, and summarize the difference from our prior work.

2.1 Problem definition

Input: We are given a set of web sources $\mathcal{W}$ and a set of extractors $\mathcal{E}$. An extractor is a method for extracting `(subject, predicate, object)` triples from a webpage. For example, one extractor may look for the *pattern "\$A, the president of \$B, ..."*, from which it can extract the triple *(A, nationality, B)*. Certainly, this is not always correct (*e.g.*, if A is the president of a company, not a country). In addition, an extractor reconciles the string representations of entities into entity identifiers such as Freebase mids, and sometimes this fails too. It is the presence of these common extractor errors, which are separate from source errors (*i.e.*, incorrect claims on a webpage), that motivates our work.

Table 2: Obama's nationality extracted by 5 extractors from 8 webpages. Column 2 (Value) shows the nationality truly provided by each source; Columns 3-7 show the nationality extracted by each extractor. Wrong extractions are shown in italics.

	Value	E_1	E_2	E_3	E_4	E_5
W_1	USA	USA	USA	USA	USA	*Kenya*
W_2	USA	USA	USA	USA	*N.Amer.*	
W_3	USA	USA		USA	*N. Amer.*	
W_4	USA	USA		USA	*Kenya*	
W_5	Kenya	Kenya	Kenya	Kenya	Kenya	Kenya
W_6	Kenya	Kenya		Kenya	*USA*	
W_7	-			*Kenya*		*Kenya*
W_8	-					*Kenya*

In the rest of the paper, we represent such triples as (data item, value) pairs, where the data item is in the form of `(subject, predicate)`, describing a particular aspect of an entity, and the `object` serves as a value for the data item. We summarize the notation used in this paper in Table 1.

We define an observation variable X_{ewdv}. We set $X_{ewdv} = 1$ if extractor e extracted value v for data item d on web source w; if it did not extract such a value, we set $X_{ewdv} = 0$. An extractor might also return confidence values indicating how confident it is in the correctness of the extraction; we consider these extensions in Section 3.5. We use matrix $X = \{X_{ewdv}\}$ to denote all the data.

We can represent X as a (sparse) "data cube", as shown in Figure 1(b). Table 2 shows an example of a single horizontal "slice" of this cube for the case where the data item is $d^* =$ *(Barack Obama, nationality).* We discuss this example in more detail next.

EXAMPLE 2.1. *Suppose we have 8 webpages, $W_1 - W_8$, and suppose we are interested in the data item* (Obama, nationality). *The value stated for this data item by each of the webpages is shown in the left hand column of Table 2. We see that $W_1 - W_4$ provide* USA *as the nationality of Obama, whereas $W_5 - W_6$ provide* Kenya *(a false value). Pages $W_7 - W_8$ do not provide any information regarding Obama's nationality.*

Now suppose we have 5 different extractors of varying reliability. The values they extract for this data item from each of the 8 webpages are shown in the table. Extractor E_1 extracts all the provided triples correctly. Extractor E_2 misses some of the provided triples (false negatives), but all of its extractions are correct. Extractor E_3 extracts all the provided triples, but also wrongly extracts the value Kenya *from W_7, even though W_7 does not provide this value (a false positive). Extractor E_4 and E_5 both have poor quality, missing a lot of provided triples and making numerous mistakes.* □

Knowledge-based trust (KBT): For each web source $w \in \mathcal{W}$, we define its *accuracy*, denoted by A_w, as the probability that a value it provides for a fact is correct (*i.e.*, consistent with the real world). We use $A = \{A_w\}$ for the set of all accuracy parameters. We now formally define the problem of KBT estimation.

DEFINITION 2.2 (KBT ESTIMATION). *The* Knowledge-Based Trust (KBT) estimation task *is to estimate the web source accuracies $A = \{A_w\}$ given the observation matrix $X = \{X_{ewdv}\}$ of extracted triples.* □

2.2 Estimating the truth using a single-layer model

KBT estimation is closely related to the *knowledge fusion* problem we studied in our previous work [11], where we evaluate the true (but latent) values for each of the data items, given the noisy observations. We introduce the binary latent variables T_{dv}, which represent whether v is a correct value for data item d. Let $T = \{T_{dv}\}$. *Given the observation matrix* $X = \{X_{ewdv}\}$*, the knowledge fusion problem computes the posterior over the latent variables,* $p(T|X)$.

One way to solve this problem is to "reshape" the cube into a two-dimensional matrix, as shown in Figure 1(a), by treating every combination of web page and extractor as a distinct data source. Now the data are in a form that standard *data fusion* techniques (surveyed in [22]) expect. We call this a *single-layer model*, since it only has one layer of latent variables (representing the unknown values for the data items). We now review this model in detail, and we compare it with our work shortly.

In our previous work [11], we applied the probabilistic model described in [8]. We assume that each data item can only have a single true value. This assumption holds for functional predicates, such as *nationality* or *date-of-birth*, but is not technically valid for set-valued predicates, such as *child*. Nevertheless, [11] showed empirically that this "single truth" assumption works well in practice even for non-functional predicates, so we shall adopt it in this work for simplicity. (See [27, 33] for approaches to deal with multi-valued attributes.)

Based on the single-truth assumption, we define a latent variable $V_d \in \text{dom}(d)$ for each data item to present the true value for d, where $\text{dom}(d)$ is the domain (set of possible values) for data item d. Let $V = \{V_d\}$ and note that we can derive $T = \{T_{dv}\}$ from V under the single-truth assumption. We then define the following observation model:

$$p(X_{sdv} = 1|V_d = v^*, A_s) = \begin{cases} A_s & \text{if } v = v^* \\ \frac{1-A_s}{n} & \text{if } v \neq v^* \end{cases} \quad (1)$$

where v^* is the true value, $s = (w, e)$ is the source, $A_s \in [0, 1]$ is the *accuracy* of this data source, and n is the number of false values for this domain (*i.e.*, we assume $|\text{dom}(d)| = n + 1$). The model says that the probability for s to provide a true value v^* for d is its accuracy, whereas the probability for it to provide one of the n false values is $1 - A_s$ divided by n.

Given this model, it is simple to apply Bayes rule to compute $p(V_d|X_d, A)$, where $X_d = \{X_{sdv}\}$ is all the data pertaining to data item d (*i.e.*, the d'th row of the data matrix), and $A = \{A_s\}$ is the set of all accuracy parameters. Assuming a uniform prior for $p(V_d)$, this can be done as follows:

$$p(V_d = v|X_d, A) = \frac{p(X_d|V_d = v, A)}{\sum_{v' \in \text{dom}(d)} p(X_d|V_d = v', A)} \quad (2)$$

where the likelihood function can be derived from Equation (1), assuming independence of the data sources:[2]

$$p(X_d|V_d = v^*, A) = \prod_{s,v:X_{sdv}=1} p(X_{sdv} = 1|V_d = v^*, A_s) \quad (3)$$

This model is called the ACCU model [8]. A slightly more advanced model, known as POPACCU, removes the assumption that the wrong values are uniformly distributed. Instead, it uses the empirical distribution of values in the observed data. It was proved that the POPACCU model is monotonic; that is, adding more sources would not reduce the quality of results [13].

In both ACCU and POPACCU, it is necessary to jointly estimate the hidden values $V = \{V_d\}$ and the accuracy parameters $A = \{A_s\}$. An iterative EM-like algorithm was proposed for performing this as follows ([8]).

- Set the iteration counter $t = 0$.
- Initialize the parameters A_s^t to some value (*e.g.*, 0.8).
- Estimate $p(V_d|X_d, A^t)$ in parallel for all d using Equation (2) (this is like the E step). From this we can compute the most probable value, $\hat{V}_d = \text{argmax}\, p(V_d|X_d, A^t)$.
- Estimate $\hat{A}_s^{(t+1)}$ as follows:

$$\hat{A}_s^{t+1} = \frac{\sum_d \sum_v \mathbb{I}(X_{sdv} = 1) p(V_d = v|X_d, A^t)}{\sum_d \sum_v \mathbb{I}(X_{sdv} = 1)} \quad (4)$$

  where $\mathbb{I}(a = b)$ is 1 if $a = b$ and is 0 otherwise. Intuitively this equation says that we estimate the accuracy of a source by the average probability of the facts it extracts. This equation is like the M step in EM.
- We now return to the E step, and iterate until convergence.

Theoretical properties of this algorithm are discussed in [8].

2.3 Estimating KBT using a multi-layer model

Although estimating KBT is closely related to knowledge fusion, the single-layer model falls short in two aspects to solve the new problem. The first issue is its inability to assess trustworthiness of web sources independently of extractors; in other words, A_s is the accuracy of a (w, e) pair, rather than the accuracy of a web source itself. Simply assuming all extracted values are actually provided by the source obviously would not work. In our example, we may wrongly infer that W_1 is a bad source because of the extracted *Kenya* value, although this is an extraction error.

The second issue is the inability to properly assess truthfulness of triples. In our example, there are 12 sources (*i.e.*, extractor-webpage pairs) for *USA* and 12 sources for *Kenya*; this seems to suggest that *USA* and *Kenya* are equally likely to be true. However, intuitively this seems unreasonable: extractors $E_1 - E_3$ all tend to agree with each other, and so seem to be reliable; we can therefore "explain away" some of the *Kenya* values extracted by $E_4 - E_5$ as being more likely to be extraction errors.

Solving these two problems requires us to distinguish extraction errors from source errors. In our example, we wish to distinguish correctly extracted true triples (*e.g.*, *USA* from $W_1 - W_4$), correctly extracted false triples (*e.g.*, *Kenya* from $W_5 - W_6$), wrongly extracted true triples (*e.g.*, *USA* from W_6), and wrongly extracted false triples (*e.g.*, *Kenya* from $W_1, W_4, W_7 - W_8$).

In this paper, we present a new probabilistic model that can estimate the accuracy of each web source, factoring out the noise introduced by the extractors. It differs from the single-layer model in two ways. First, in addition to the latent variables to represent the true value of each data item (V_d), the new model introduces a set of latent variables to represent whether each extraction was correct or not; this allows us to distinguish extraction errors and source data errors. Second, instead of using A to represent the accuracy of (e, w) pairs, the new model defines a set of parameters for the accuracy of the web sources, and for the quality of the extractors; this allows us to separate the quality of the sources from that of the extractors. We call the new model the *multi-layer model*, because it contains two layers of latent variables and parameters (Section 3).

The fundamental differences between the multi-layer model and the single-layer model allow for reliable KBT estimation. In Section 4, we also show how to dynamically select the granularity of a source and an extractor. Finally, in Section 5, we show empirically how both components play an important role in improving the performance over the single-layer model.

[2] Previous works [8, 27] discussed how to detect copying and correlations between sources in data fusion; however, scaling them up to billions of web sources remains an open problem.

3. MULTI-LAYER MODEL

In this section, we describe in detail how we compute $A = \{A_w\}$ from our observation matrix $X = \{X_{ewdv}\}$ using a multi-layer model.

3.1 The multi-layer model

We extend the previous single-layer model in two ways. First, we introduce the binary latent variables C_{wdv}, which represent whether web source w actually provides triple (d, v) or not. Similar to Equation (1), these variables depend on the true values V_d and the accuracies of each of the web sources A_w as follows:

$$p(C_{wdv} = 1 | V_d = v^*, A_w) = \begin{cases} A_w & \text{if } v = v^* \\ \frac{1-A_w}{n} & \text{if } v \neq v^* \end{cases} \quad (5)$$

Second, following [27, 33], we use a two-parameter noise model for the observed data, as follows:

$$p(X_{ewdv} = 1 | C_{wdv} = c, Q_e, R_e) = \begin{cases} R_e & \text{if } c = 1 \\ Q_e & \text{if } c = 0 \end{cases} \quad (6)$$

Here R_e is the *recall* of the extractor; that is, the probability of extracting a truly provided triple. And Q_e is 1 minus the *specificity*; that is, the probability of extracting an unprovided triple. Parameter Q_e is related to the recall (R_e) and precision (P_e) as follows:

$$Q_e = \frac{\gamma}{1-\gamma} \cdot \frac{1-P_e}{P_e} \cdot R_e \quad (7)$$

where $\gamma = p(C_{wdv} = 1)$ for any $v \in \mathrm{dom}(d)$, as explained in [27]. (Table 3 gives a numerical example of computing Q_e from P_e and R_e.)

To complete the specification of the model, we must specify the prior probability of the various model parameters:

$$\theta_1 = \{A_w\}_{w=1}^{W}, \theta_2 = (\{P_e\}_{e=1}^{E}, \{R_e\}_{e=1}^{E}), \theta = (\theta_1, \theta_2) \quad (8)$$

For simplicity, we use uniform priors on the parameters. By default, we set $A_w = 0.8$, $R_e = 0.8$, and $Q_e = 0.2$. In Section 5, we discuss an alternative way to estimate the initial value of A_w, based on the fraction of correct triples that have been extracted from this source, using an external estimate of correctness (based on *Freebase* [2]).

Let $V = \{V_d\}$, $C = \{C_{wdv}\}$, and $Z = (V, C)$ be all the latent variables. Our model defines the following joint distribution:

$$p(X, Z, \theta) = p(\theta)p(V)p(C|V, \theta_1)p(X|C, \theta_2) \quad (9)$$

We can represent the conditional independence assumptions we are making using a graphical model, as shown in Figure 2. The shaded node is an observed variable, representing the data; the unshaded nodes are hidden variables or parameters. The arrows indicate the dependence between the variables and parameters. The boxes are known as "plates" and represent repetition of the enclosed variables; for example, the box of e repeats for every extractor $e \in \mathcal{E}$.

3.2 Inference

Recall that estimating KBT essentially requires us to compute the posterior over the parameters of interest, $p(A|X)$. Doing this exactly is computationally intractable, because of the presence of the latent variables Z. One approach is to use a Monte Carlo approximation, such as Gibbs sampling, as in [32]. However, this can be slow and is hard to implement in a Map-Reduce framework, which is required for the scale of data we use in this paper.

A faster alternative is to use EM, which will return a point estimate of all the parameters, $\hat{\theta} = \mathrm{argmax}\, p(\theta|X)$. Since we are using a uniform prior, this is equivalent to the maximum likelihood estimate $\hat{\theta} = \mathrm{argmax}\, p(X|\theta)$. From this, we can derive $\hat{A}$.

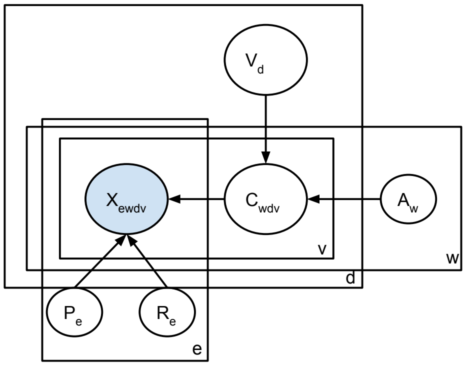


Figure 2: A representation of the multi-layer model using graphical model plate notation.

```
Algorithm 1: MultiLayer($X, t_{max}$)
Input   : $X$: all extracted data;
          $t_{max}$: max number of iterations.
Output: Estimates of $Z$ and $\theta$.
Initialize $\theta$ to default values;
for $t \in [1, t_{max}]$ do
    Estimate $C$ by Eqs.(15, 26, 31);
    Estimate $V$ by Eqs.(23-25);
    Estimate $\theta_1$ by Eq.(28);
    Estimate $\theta_2$ by Eqs.(32-33);
    if $Z, \theta$ converge then
        break;
return $Z, \theta$;
```

As pointed out in [26], an exact EM algorithm has a quadratic complexity even for a single-layer model, so is unaffordable for data of web scale. Instead, we use an iterative "EM like" estimation procedure, where we initialize the parameters as described previously, and then alternate between estimating Z and then estimating θ, until we converge.

We first give an overview of this EM-like algorithm, and then go into details in the following sections.

In our case, Z consists of two "layers" of variables. We update them sequentially, as follows. First, let $X_{wdv} = \{X_{ewdv}\}$ denote all extractions from web source w about a particular triple $t = (d, v)$. We compute the extraction correctness $p(C_{wdv}|X_{wdv}, \theta_2^t)$, as explained in Section 3.3.1, and then we compute $\hat{C}_{wdv} = \mathrm{argmax}\, p(C_{wdv}|X_{wdv}, \theta_2^t)$, which is our best guess about the "true contents" of each web source. This can be done in parallel over d, w, v.

Let $\hat{C}_d = \hat{C}_{wdv}$ denote all the estimated values for d across the different websites. We then compute $p(V_d|\hat{C}_d, \theta_1^t)$, as explained in Section 3.3.2, and then we compute $\hat{V}_d = \mathrm{argmax}\, p(V_d|\hat{C}_d, \theta_1^t)$, which is our best guess about the "true value" of each data item. This can be done in parallel over d.

Having estimated the latent variables, we then estimate θ^{t+1}. This parameter update also consists of two steps (but can be done in parallel): estimating the source accuracies $\{A_w\}$ and the extractor reliabilities $\{P_e, R_e\}$, as explained in Section 3.4.

Algorithm 1 gives a summary of the pseudo code; we give the details next.

3.3 Estimating the latent variables

We now give the details of how we estimate the latent variables Z. For notational brevity, we drop the conditioning on θ^t, except

Table 3: Quality and vote counts of extractors in the motivating example. We assume $\gamma = .25$ when we derive Q_e from P_e and R_e.

	E_1	E_2	E_3	E_4	E_5
$Q(E_i)$	.01	.01	.06	.22	.17
$R(E_i)$	.99	.5	.99	.33	.17
$P(E_i)$	.99	.99	.85	.33	.25
$Pre(E_i)$	4.6	3.9	2.8	.4	0
$Abs(E_i)$	-4.6	-.7	-4.5	-.15	0

where needed.

3.3.1 Estimating extraction correctness

We first describe how to compute $p(C_{wdv} = 1|X_{wdv})$, following the "multi-truth" model of [27]. We will denote the prior probability $p(C_{wdv} = 1)$ by α. In initial iterations, we initialize this to $\alpha = 0.5$. Note that by using a fixed prior, we break the connection between C_{wdv} and V_d in the graphical model, as shown in Figure 2. Thus, in subsequent iterations, we re-estimate $p(C_{wdv} = 1)$ using the results of V_d obtained from the previous iteration, as explained in Section 3.3.4.

We use Bayes rule as follows:

$$\begin{aligned} & p(C_{wdv} = 1|X_{wdv}) \\ = \ & \frac{\alpha p(X_{wdv}|C_{wdv} = 1)}{\alpha p(X_{wdv}|C_{wdv} = 1) + (1-\alpha) p(X_{wdv}|C_{wdv} = 0)} \\ = \ & \frac{1}{1 + \frac{1}{\frac{p(X_{wdv}|C_{wdv}=1)}{p(X_{wdv}|C_{wdv}=0)} \frac{\alpha}{1-\alpha}}} \\ = \ & \sigma\left(\log \frac{p(X_{wdv}|C_{wdv} = 1)}{p(X_{wdv}|C_{wdv} = 0)} + \log \frac{\alpha}{1-\alpha}\right) \qquad (10) \end{aligned}$$

where $\sigma(x) \triangleq \frac{1}{1+e^{-x}}$ is the *sigmoid* function.

Assuming independence of the extractors, and using Equation (6), we can compute the likelihood ratio as follows:

$$\frac{p(X_{wdv}|C_{wdv} = 1)}{p(X_{wdv}|C_{wdv} = 0)} = \prod_{e:X_{ewdv}=1} \frac{R_e}{Q_e} \prod_{e:X_{ewdv}=0} \frac{1-R_e}{1-Q_e} \qquad (11)$$

In other words, for each extractor we can compute a *presence vote* Pre_e for a triple that it extracts, and an *absence vote* of Abs_e for a triple that it does not extract:

$$\begin{aligned} \text{Pre}_e & \triangleq \log R_e - \log Q_e & (12) \\ \text{Abs}_e & \triangleq \log(1 - R_e) - \log(1 - Q_e). & (13) \end{aligned}$$

For each triple (w, d, v) we can compute its *vote count* as the sum of the presence votes and the absence votes:

$$VCC(w, d, v) \triangleq \sum_{e:X_{ewdv}=1} \text{Pre}_e + \sum_{e:X_{ewdv}=0} \text{Abs}_e \qquad (14)$$

Accordingly, we can rewrite Equation (10) as follows.

$$p(C_{wdv} = 1|X_{wdv}) = \sigma\left(VCC(w, d, v) + \log \frac{\alpha}{1-\alpha}\right). \qquad (15)$$

EXAMPLE 3.1. *Consider the extractors in the motivating example (Table 2). Suppose we know Q_e and R_e for each extractor e as shown in Table 3. We can then compute Pre_e and Abs_e as shown in the same table. We observe that in general, an extractor with low Q_e (unlikely to extract an unprovided triple;* e.g., *E_1, E_2) often has a high presence vote; an extractor with high R_e (likely to extract a provided triple;* e.g., *E_1, E_3) often has a low (negative) absence vote; and a low-quality extractor (*e.g., *E_5) often has a low presence vote and a high absence vote.*

Table 4: Extraction correctness and value truthfulness for the data in Table 2, using the extraction parameters in Table 3. Columns 2-4 show $p(C_{wdv} = 1|X_{wdv})$, as explained in Example 3.1. The last row shows $p(V_d|\hat{C}_d)$, as explained in Example 3.2; note that this distribution does not sum to 1.0, since not all of the values are shown in the table.

	USA	Kenya	N.Amer.
W_1	1	0	-
W_2	1	-	0
W_3	1	-	0
W_4	1	0	-
W_5	-	1	-
W_6	0	1	-
W_7	-	.07	-
W_8	-	0	-
$p(V_d\|\hat{C}_d)$	.995	.004	0

Now consider applying Equation (15) to compute the likelihood that a particular source provides the triple t^* =(Obama, nationality, USA)*, assuming $\alpha = 0.5$. For source W_1, we see that extractors $E_1 - E_4$ extract t^*, so the vote count is* $(4.6 + 3.9 + 2.8 + 0.4) + (0) = 11.7$ *and hence $p(C_{1,t^*} = 1|X_{w,t^*}) = \sigma(11.7) = 1$. For source W_6, we see that only E_4 extracts t^*, so the vote count is* $(0.4) + (-4.6 - 0.7 - 4.5 - 0) = -9.4$, *and hence $p(C_{6,t^*} = 1|X_{6,t^*})) = \sigma(-9.4) = 0$. Some other values for $P(C_{wt} = 1|X_{wt})$ are shown in Table 4.* □

Having computed $p(C_{wdv} = 1|X_{wdv})$, we can compute $\hat{C}_{wdv} = \text{argmax}\, p(C_{wdv}|X_{wdv})$. This serves as the input to the next step of inference.

3.3.2 Estimating true value of the data item

In this step, we compute $p(V_d = v|\hat{C}_d)$, following the "single truth" model of [8]. By Bayes rule we have

$$p(V_d = v|\hat{C}_d) = \frac{p(\hat{C}_d|V_d = v)p(V_d = v)}{\sum_{v' \in \text{dom}(d)} p(\hat{C}_d|V_d = v')p(V_d = v')} \qquad (16)$$

Since we do not assume any prior knowledge of the correct values, we use a uniform prior $p(V_d = v)$, so we just need to focus on the likelihood. Using Equation (5), we have

$$\begin{aligned} & p(\hat{C}_d|V_d = v) \\ = \ & \prod_{w:\hat{C}_{wdv}=1} A_w \prod_{w:\hat{C}_{wdv}=0} \frac{1-A_w}{n} & (17) \\ = \ & \prod_{w:\hat{C}_{wdv}=1} \frac{nA_w}{1-A_w} \prod_{w:\hat{C}_{wdv}\in\{0,1\}} \frac{1-A_w}{n} & (18) \end{aligned}$$

Since the latter term $\prod_{w:\hat{C}_{wdv}\in\{0,1\}} \frac{1-A_w}{n}$ is constant with respect to v, we can drop it.

Now let us define the vote count as follows:

$$VCV(w) \triangleq \log \frac{nA_w}{1-A_w} \qquad (19)$$

Aggregating over web sources that provide this triple, we define

$$VCV(d, v) \triangleq \sum_w \mathbb{I}(\hat{C}_{wdv} = 1)VCV(w) \qquad (20)$$

With this notation, we can rewrite Equation (16) as

$$p(V_d = v|\hat{C}_d) = \frac{\exp(VCV(d, v))}{\sum_{v' \in \text{dom}(d)} \exp(VCV(d, v'))} \qquad (21)$$

EXAMPLE 3.2. *Assume we have correctly decided the triple provided by each web source, as in the "Value" column of Table 2. Assume each source has the same accuracy* $A_w = 0.6$ *and* $n = 10$, *so the vote count is* $\ln(\frac{10*0.6}{1-0.6}) = 2.7$. *Then* USA *has vote count* $2.7 * 4 = 10.8$, Kenya *has vote count* $2.7 * 2 = 5.4$, *and an unprovided value, such as* N.Amer, *has vote count* 0. *Since there are 10 false values in the domain, so there are 9 unprovided values. Hence we have* $p(V_d = USA|\hat{C}_d) = \frac{\exp(10.8)}{Z} = 0.995$, *where* $Z = \exp(10.8) + \exp(5.4) + \exp(0) * 9$. *Similarly,* $p(V_d = Kenya|\hat{C}_d) = \frac{exp(5.4)}{Z} = 0.004$. *This is shown in the last row of Table 4. The missing mass of* $1 - (0.995 + 0.004)$ *is assigned (uniformly) to the other 9 values that were not observed (but in the domain).* □

3.3.3 An improved estimation procedure

So far, we have assumed that we first compute a MAP estimate $\hat{C}_{wdv}$, which we then use as evidence for estimating V_d. However, this ignores the uncertainty in $\hat{C}$. The correct thing to do is to compute $p(V_d|X_d)$ marginalizing out over C_{wdv}.

$$\begin{aligned} p(V_d|X_d) &\propto P(V_d)P(X_d|V_d) \\ &= p(V_d)\sum_{\vec{c}} p(C_d = \vec{c}|V_d)p(X_d|C_d) \end{aligned} \tag{22}$$

Here we can consider each $\vec{c}$ as a *possible world*, where each element c_{wdv} indicates whether a source w provides a triple (d, v) (value 1) or not (value 0).

As a simple heuristic approximation to this approach, we replace the previous vote counting with a weighted version, as follows:

$$VCV'(w, d, v) \triangleq p(C_{wdv} = 1|X_d) \log \frac{nA_w}{1 - A_w} \tag{23}$$

$$VCV'(d, v) \triangleq \sum_w VCV'(d, w, v) \tag{24}$$

We then compute

$$p(V_d = v|X_d) \approx \frac{\exp(VCV'(d, v))}{\sum_{v' \in \text{dom}(d)} \exp(VCV'(d, v'))} \tag{25}$$

We will show that such improved estimation procedure improves upon ignoring the uncertainty in $\hat{C}_d$ in experiments (Section 5.3.3).

3.3.4 Re-estimating the prior of correctness

In Section 3.3.1, we assumed that $p(C_{wdv} = 1) = \alpha$ was known, which breaks the connection between V_d and C_{wdv}. Thus, we update this prior after each iteration according to the correctness of the value and the accuracy of the source:

$$\hat{\alpha}^{t+1} = p(V_d = v|X)A_w + (1 - p(V_d = v|X))(1 - A_w) \tag{26}$$

We can then use this refined estimate in the following iteration. We give an example of this process.

EXAMPLE 3.3. *Consider the probability that* W_7 *provides* $t' =$ *(Obama, nationality, Kenya). Two extractors extract* t' *from* W_7 *and the vote count is -2.65, so the initial estimate is* $p(C_{wdv} = 1|X) = \sigma(-2.65) = 0.06$. *However, after the previous iteration has finished, we know that* $p(V_d = Kenya|X) = 0.004$. *This gives us a modified prior probability as follows:* $p'(C_{wt} = 1) = 0.004 * 0.6 + (1 - 0.004) * (1 - 0.6) = 0.4$, *assuming* $A_w = 0.6$. *Hence the updated posterior probability is given by* $p'(C_{wt} = 1|X) = \sigma(-2.65 + \log\frac{1-0.4}{0.4}) = 0.04$, *which is lower than before.* □

3.4 Estimating the quality parameters

Having estimated the latent variables, we now estimate the parameters of the model.

3.4.1 Source quality

Following [8], we estimate the accuracy of a source by computing the average probability of its provided values being true:

$$\hat{A}_w^{t+1} = \frac{\sum_{dv:\hat{C}_{wdv}=1} p(V_d = v|X)}{\sum_{dv:\hat{C}_{wdv}=1} 1} \tag{27}$$

We can take uncertainty of $\hat{C}$ into account as follows:

$$\hat{A}_w^{t+1} = \frac{\sum_{dv:\hat{C}_{wdv}>0} p(C_{wdv} = 1|X)p(V_d = v|X)}{\sum_{dv:\hat{C}_{wdv}>0} p(C_{wdv} = 1|X)} \tag{28}$$

This is the key equation behind Knowledge-based Trust estimation: it estimates the accuracy of a web source as the weighted average of the probability of the facts that it contains (provides), where the weights are the probability that these facts are indeed contained in that source.

3.4.2 Extractor quality

According to the definition of precision and recall, we can estimate them as follows:

$$\hat{P}_e^{t+1} = \frac{\sum_{wdv:X_{ewdv}=1} p(C_{wdv} = 1|X)}{\sum_{wdv:X_{ewdv}=1} 1} \tag{29}$$

$$\hat{R}_e^{t+1} = \frac{\sum_{wdv:X_{ewdv}=1} p(C_{wdv} = 1|X)}{\sum_{wdv} p(C_{wdv} = 1|X)} \tag{30}$$

Note that for reasons explained in [27], it is much more reliable to estimate P_e and R_e from data, and then compute Q_e using Equation (7), rather than trying to estimate Q_e directly.

3.5 Handling confidence-weighted extractions

So far, we have assumed that each extractor returns a binary decision about whether it extracts a triple or not, $X_{ewdv} \in \{0, 1\}$. However, in real life, extractors return confidence scores, which we can interpret as the probability that the triple is present on the page according to that extractor. Let us denote this "soft evidence" by $p(X_{ewdv} = 1) = \overline{X}_{ewdv} \in [0, 1]$. A simple way to handle such data is to binarize it, by thresholding. However, this loses information, as shown in the following example.

EXAMPLE 3.4. *Consider the case that* E_1 *and* E_3 *are not fully confident with their extractions from* W_3 *and* W_4. *In particular,* E_1 *gives each extraction a probability (*i.e.*, confidence) .85, and* E_3 *gives probability .5. Although no extractor has full confidence for the extraction, after observing their extractions collectively, we would be fairly confident that* W_3 *and* W_4 *indeed provide triple* $T =$(Obama, nationality, USA).

However, if we simply apply a threshold of .7, we would ignore the extractions from W_3 *and* W_4 *by* E_3. *Because of lack of extraction, we would conclude that neither* W_3 *nor* W_4 *provides* T. *Then, since* USA *is provided by* W_1 *and* W_2, *whereas* Kenya *is provided by* W_5 *and* W_6, *and the sources all have the same accuracy, we would compute an equal probability for* USA *and for* Kenya. □

Following the same approach as in Equation (23), we propose to modify Equation (14) as follows:

$$VCC'(w, d, v) \triangleq \sum_e [p(X_{ewdv} = 1)\text{Pre}_e + p(X_{ewdv} = 0)\text{Abs}_e] \tag{31}$$

Similarly, we modify the precision and recall estimates:

$$\hat{P}_e = \frac{\sum_{wdv:\overline{X}_{ewdv}>0} p(X_{ewdv}=1)p(C_{wdv}=1|X)}{\sum_{wdv:\overline{X}_{ewdv}>0} p(X_{ewdv}=1)} \tag{32}$$

$$\hat{R}_e = \frac{\sum_{wdv:\overline{X}_{ewdv}>0} p(X_{ewdv}=1)p(C_{wdv}=1|X)}{\sum_{wdv} p(C_{wdv}=1|X)} \tag{33}$$

4. DYNAMICALLY SELECTING GRANULARITY

This section describes the choice of the granularity for web sources; at the end of this section we discuss how to apply it to extractors. This step is conducted before applying the multi-layer model.

Ideally, we wish to use the finest granularity. For example, it is natural to treat each webpage as a separate source, as it may have a different accuracy from other webpages. We may even define a source as a specific predicate on a specific webpage; this allows us to estimate how trustworthy a page is about a specific kind of predicate. However, when we define sources too finely, we may have too little data to reliably estimate their accuracies; conversely, there may exist sources that have too much data even at their finest granularity, which can cause computational bottlenecks.

To handle this, we wish to dynamically choose the granularity of the sources. For too small sources, we can "back off" to a coarser level of the hierarchy; this allows us to "borrow statistical strength" between related pages. For too large sources, we may choose to split it into multiple sources and estimate their accuracies independently. When we do merging, our goal is to improve the statistical quality of our estimates without sacrificing efficiency. When we do splitting, our goal is to significantly improve efficiency in presence of data skew, without changing our estimates dramatically.

To be more precise, we can define a source at multiple levels of resolution by specifying the following values of a feature vector: $\langle$`website, predicate, webpage`$\rangle$, ordered from most general to most specific. We can then arrange these sources in a hierarchy. For example, $\langle$*wiki.com*$\rangle$ is a parent of $\langle$*wiki.com*, date_of_birth$\rangle$, which in turn is a parent of $\langle$*wiki.com*, date_of_birth, *wiki.com/page1.html*$\rangle$. We define the following two operators.

- **Split:** When we split a large source, we wish to split it randomly into sub-sources of similar sizes. Specifically, let W be a source with size $|W|$, and M be the maximum size we desire; we uniformly distribute the triples from W into $\lceil \frac{|W|}{M} \rceil$ buckets, each representing a sub-source. We set M to a large number that does not require splitting sources unnecessarily and meanwhile would not cause computational bottleneck according to the system performance.
- **Merge:** When we merge small sources, we wish to merge only sources that share some common features, such as sharing the same predicate, or coming from the same website. Hence, we only merge children with the same parent in the hierarchy when their size is below a pre-defined minimum size m. We set m to a small number that does not require merging sources unnecessarily while maintaining enough statistical strength.

EXAMPLE 4.1. *Consider three sources:* $\langle$website1.com, date_of_birth$\rangle$, $\langle$website1.com, place_of_birth$\rangle$, $\langle$website1.com, gender$\rangle$*, each with two triples, arguably not enough for quality evaluation. We can merge them into their parent source by removing the second feature. We then obtain a source* $\langle$website1.com$\rangle$ *with size* $2*3=6$*, which gives more data for quality evaluation.* □

```
Algorithm 2: SplitAndMerge($\mathbf{W}, m, M$)
Input   : $\mathbf{W}$: sources with finest granularity;
          $m/M$: min/max source size in desire.
Output  : $\mathbf{W}'$: a new set of sources with desired size.
1  $\mathbf{W}' \leftarrow \emptyset$;
2  for $W \in \mathbf{W}$ do
3      $\mathbf{W} \leftarrow \mathbf{W} \setminus \{W\}$;
4      if $|W| > M$ then
5          $\mathbf{W}' \leftarrow \mathbf{W}' \cup$ SPLIT($W$);
6      else if $|W| < m$ then
7          $W_{par} \leftarrow$ GETPARENT ($W$);
8          if $W_{par} = \perp$ then
9              // Already reach the top of the hierarchy
               $\mathbf{W}' \leftarrow \mathbf{W}' \cup \{W\}$;
10         else
11             $\mathbf{W} \leftarrow \mathbf{W} \cup \{W_{par}\}$;
12     else
13         $\mathbf{W}' \leftarrow \mathbf{W}' \cup \{W\}$;
14 return $\mathbf{W}'$;
```

Note that when we merge small sources, the result parent source may not be of desired size: it may still be too small, or it may be too large after we merge a huge number of small sources. As a result, we might need to iteratively merge the resulting sources to their parents, or splitting an oversized resulting source, as we describe in the full algorithm.

Algorithm 2 gives the SPLITANDMERGE algorithm. We use $\mathbf{W}$ for sources for examination and $\mathbf{W}'$ for final results; at the beginning $\mathbf{W}$ contains all sources of the finest granularity and $\mathbf{W}' = \emptyset$ (Ln 1). We consider each $W \in \mathbf{W}$ (Ln 2). If W is too large, we apply SPLIT to split it into a set of sub-sources; SPLIT guarantees that each sub-source would be of desired size, so we add the sub-sources to $\mathbf{W}'$ (Ln 5). If W is too small, we obtain its parent source (Ln 7). In case W is already at the top of the source hierarchy so it has no parent, we add it to $\mathbf{W}'$ (Ln 8); otherwise, we add W_{par} back to $\mathbf{W}$ (Ln 11). Finally, for sources already in desired size, we move them directly to $\mathbf{W}'$ (Ln 13).

EXAMPLE 4.2. *Consider a set of 1000 sources* $\langle W, P_i, URL_i \rangle$, $i \in [1, 1000]$*; in other words, they belong to the same website, each has a different predicate and a different URL. Assuming we wish to have sources with size in* $[5, 500]$, MULTILAYERSM *proceeds in three stages.*

In the first stage, each source is deemed too small and is replaced with its parent source $\langle W, P_i \rangle$*. In the second stage, each new source is still deemed too small and is replaced with its parent source* $\langle W \rangle$*. In the third stage, the single remaining source is deemed too large and is split uniformly into two sub-sources. The algorithm terminates with 2 sources, each of size 500.* □

Finally, we point out that the same techniques apply to extractors as well. We define an extractor using the following feature vector, again ordered from most general to most specific: $\langle$`extractor, pattern, predicate, website`$\rangle$. The finest granularity represents the quality of a particular extractor pattern (different patterns may have different quality), on extractions for a particular predicate (in some cases when a pattern can extract triples of different predicates, it may have different quality), from a particular website (a pattern may have different quality on different websites).

5. EXPERIMENTAL RESULTS

This section describes our experimental results on a synthetic data set (where we know the ground truth), and on large-scale real-world data. We show that (1) our algorithm can effectively estimate the correctness of extractions, the truthfulness of triples, and the accuracy of sources; (2) our model significantly improves over the state-of-the-art methods for knowledge fusion; and (3) KBT provides a valuable additional signal for web source quality.

5.1 Experiment Setup

5.1.1 Metrics

We measure how well we predict extraction correctness, triple probability, and source accuracy. For synthetic data, we have the benefit of ground truth, so we can exactly measure all three aspects. We quantify this in terms of *square loss*; the lower the square loss, the better. Specifically, SqV measures the average square loss between $p(V_d = v|X)$ and the true value of $\mathbb{I}(V_d^* = v)$; SqC measures the average square loss between $p(C_{wdv} = 1|X)$ and the true value of $\mathbb{I}(C_{wdv}^* = 1)$; and SqA measures the average square loss between $\hat{A}_w$ and the true value of A_w^*.

For real data, however, as we show soon, we do not have a gold standard for source trustworthiness, and we have only a partial gold standard for triple correctness and extraction correctness. Hence for real data, we just focus on measuring how well we predict triple truthfulness. In addition to SqV, we also used the following three metrics for this purpose, which were also used in [11].

- *Weighted deviation (WDev)*: WDev measures whether the predicted probabilities are *calibrated*. We divide our triples according to the predicted probabilities into buckets $[0, 0.01)$, $\dots, [0.04, 0.05), [0.05, 0.1), \dots, [0.9, 0.95), [0.95, 0.96), \dots,$ $[0.99, 1), [1, 1]$ (most triples fall in $[0, 0.05)$ and $[0.95, 1]$, so we used a finer granularity there). For each bucket we compute the accuracy of the triples according to the gold standard, which can be considered as the real probability of the triples. WDev computes the average square loss between the predicted probabilities and the real probabilities, weighted by the number of triples in each bucket; the lower the better.
- *Area under precision recall curve (AUC-PR)*: AUC-PR measures whether the predicted probabilities are *monotonic*. We order triples according to the computed probabilities and plot PR-curves, where the X-axis represents the recall and the Y-axis represents the precision. AUC-PR computes the area-under-the-curve; the higher the better.
- *Coverage (Cov)*: Cov computes for what percentage of the triples we compute a probability (as we show soon, we may ignore data from a source whose quality remains at the default value over all the iterations).

Note that on the synthetic data Cov is 1 for all methods, and the comparison of different methods regarding AUC-PR and WDev is very similar to that regarding SqV, so we skip the plots.

5.1.2 Methods being compared

We compared three main methods. The first, which we call SINGLELAYER, implements the state-of-the-art methods for knowledge fusion [11] (overviewed in Section **??**). In particular, each source or "provenance" is a 4-tuple `(extractor, website, predicate, pattern)`. We consider a provenance in fusion only if its accuracy does not remain default over iterations because of low coverage. We set $n = 100$ and iterate 5 times. These settings have been shown in [11] to perform best.

The second, which we call MULTILAYER, implements the multi-layer model described in Section 3. To have reasonable execution time, we used the finest granularity specified in Section 4 for extractors and sources: each extractor is an `(extractor, pattern, predicate, website)` vector, and each source is a `(website, predicate, webpage)` vector. When we decide extraction correctness, we consider the confidence provided by extractors, normalized to $[0, 1]$, as in Section 3.5. If an extractor does not provide confidence, we assume the confidence is 1. When we decide triple truthfulness, by default we use the improved estimate $p(C_{wdv} = 1|X)$ described in Section 3.3.3, instead of simply using $\hat{C}_{wdv}$. We start updating the prior probabilities $p(C_{wdv} = 1)$, as described in Section 3.3.4, starting from the third iteration, since the probabilities we compute get stable after the second iteration. For the noise models, we set $n = 10$ and $\gamma = 0.25$, but we found other settings lead to quite similar results. We vary the settings and show the effect in Section 5.3.3.

The third method, which we call MULTILAYERSM, implements the SPLITANDMERGE algorithm in addition to the multi-layer model, as described in Section 4. We set the min and max sizes to $m = 5$ and $M = 10K$ by default, and varied them in Section 5.3.4.

For each method, there are two variants. The first variant determines which version of the $p(X_{ewdv}|C_{wdv})$ model we use. We tried both ACCU and POPACCU. We found that the performance of the two variants on the single-layer model was very similar, while POPACCU is slightly better. However, rather surprisingly, we found that the POPACCU version of the multi-layer model was worse than the ACCU version. This is because we have not yet found a good way to combine the POPACCU model with the improved estimation procedure described in Section 3.3.3. Consequently, we only report results for the ACCU version in what follows.

The second variant is how we initialize source quality. We either assign a default quality ($A_w = 0.8, R_e = 0.8, Q_e = 0.2$) or initialize the quality according to a gold standard, as explained in Section 5.3. In this latter case, we append $+$ to the method name to distinguish it from the default initialization (*e.g.*, SINGLELAYER+).

5.2 Experiments on synthetic data

5.2.1 Data set

We randomly generated data sets containing 10 sources and 5 extractors. Each source provides 100 triples with an accuracy of $A = 0.7$. Each extractor extracts triples from a source with probability $\delta = 0.5$; for each source, it extracts a provided triple with probability $R = 0.5$; accuracy among extracted subjects (same for predicates, objects) is $P = 0.8$ (in other words, the precision of the extractor is $P_e = P^3$). In each experiment we varied one parameter from 0.1 to 0.9 and fixed the others; for each experiment we repeated 10 times and reported the average. Note that our default setting represents a challenging case, where the sources and extractors are of relatively low quality.

5.2.2 Results

Figure 3 plots SqV, SqC, and SqA as we increase the number of extractors. We assume SINGLELAYER considers all extracted triples when computing source accuracy. We observe that the multi-layer model always performs better than the single-layer model. As the number of extractors increases, SqV goes down quickly for the multi-layer model, and SqC also decreases, albeit more slowly. Although the extra extractors can introduce much more noise extractions, SqA stays stable for MULTILAYER, whereas it increases quite a lot for SINGLELAYER.

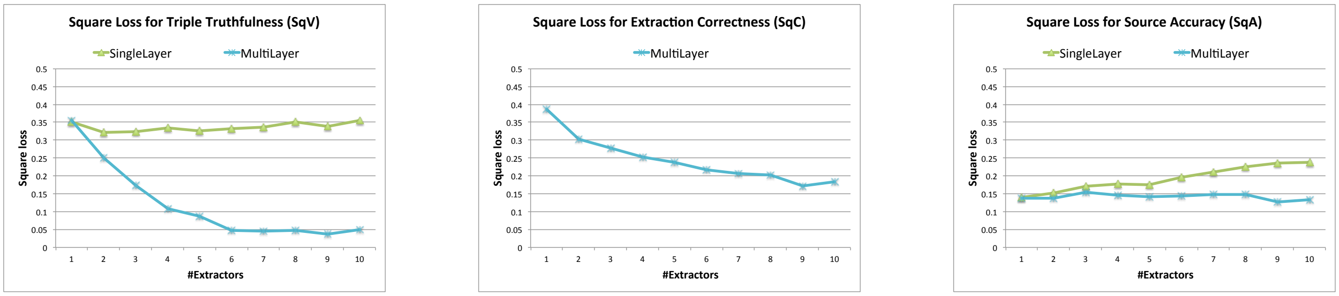


Figure 3: Error in estimating V_d, C_{wdv} and A_w as we vary the number of extractors in the synthetic data. The multi-layer model has significantly lower square loss than the single-layer model. The single-layer model cannot estimate C_{wdv}, resulting with one line for SqC.

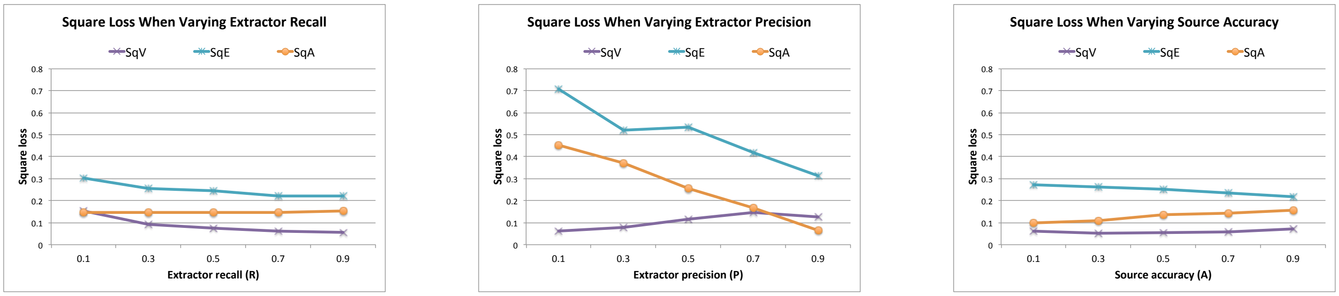


Figure 4: Error in estimating V_d, C_{wdv} and A_w as we vary extractor quality (P and R) and source quality (A) in the synthetic data.

Next we vary source and extractor quality. MultiLayer continues to perform better than SingleLayer everywhere and Figure 4 plots only for MultiLayer as we vary R, P and A (the plot for varying δ is similar to that for varying R). In general the higher quality, the lower the loss. There are a few small deviations from this trend. When the extractor recall (R) increases, SqA does not decrease, as the extractors also introduce more noise. When the extractor precision (P) increases, we give them higher trust, resulting in a slightly higher (but still low) probability for false triples; since there are many more false triples than true ones, SqV slightly increases. Similarly, when A increases, there is a very slight increase in SqA, because we trust the false triples a bit more. However, overall, we believe the experiments on the synthetic data demonstrate that our algorithm is working as expected, and can successfully approximate the true parameter values in these controlled settings.

5.3 Experiments on KV data

5.3.1 Data set

We experimented with knowledge triples collected by Knowledge Vault [10] on 7/24/2014; for simplicity we call this data set *KV*. There are 2.8B triples extracted from 2B+ webpages by 16 extractors, involving 40M extraction patterns. Comparing with an old version of the data collected on 10/2/2013 [11], the current collection is 75% larger, involves 25% more extractors, 8% more extraction patterns, and twice as many webpages.

Figure 5 shows the distribution of the number of distinct extracted triples per URL and per extraction pattern. On the one hand, we observe some huge sources and extractors: 26 URLs each contributes over 50K triples (a lot due to extraction mistakes), 15 websites each contributes over 100M triples, and 43 extraction patterns each extracts over 1M triples. On the other hand, we observe long tails: 74% URLs each contributes fewer than 5 triples, and 48% extraction patterns each extracts fewer than 5 triples. Our SplitAndMerge strategy is exactly motivated by such observations.

Table 5: Comparison of various methods on *KV*; best performance in each group is in bold. For SqV and WDev, lower is better; for AUC-PR and Cov, higher is better.

SqV	SqV	WDev	AUC-PR	Cov
SingleLayer	0.131	0.061	**0.454**	**0.952**
MultiLayer	0.105	0.042	0.439	0.849
MultiLayerSM	**0.090**	**0.021**	0.449	0.939
SingleLayer+	0.063	0.0043	0.630	0.953
MultiLayer+	**0.054**	0.0040	**0.693**	0.864
MultiLayerSM+	0.059	**0.0039**	0.631	**0.955**

To determine whether these triples are true or not (gold standard labels), we use two methods. The first method is called the *Local-Closed World Assumption (LCWA)* [10, 11, 15] and works as follows. A triple (s, p, o) is considered as `true` if it appears in the Freebase KB. If the triple is missing from the KB but (s, p) appears for any other value o', we assume the KB is locally complete (for (s, p)), and we label the (s, p, o) triple as `false`. We label the rest of the triples (where (s, p) is missing) as `unknown` and remove them from the evaluation set. In this way we can decide truthfulness of 0.74B triples (26% in *KV*), of which 20% are true (in Freebase).

Second, we apply type checking to find incorrect extractions. In particular, we consider a triple (s, p, o) as `false` if 1) $s = o$; 2) the type of s or o is incompatible with what is required by the predicate; or 3) o is outside the expected range (*e.g.*, the weight of an athlete is over 1000 pounds). We discovered 0.56B triples (20% in KV) that violate such rules and consider them both as `false` triples and as extraction mistakes.

Our gold standard include triples from both labeling methods. It contains in total 1.3B triples, among which 11.5% are true.

5.3.2 Single-layer vs multi-layer

Table 5 compares the performance of the three methods. Figure 8 plots the calibration curve and Figure 9 plots the PR-curve. We see that all methods are fairly well calibrated, but the multi-layer model has a better PR curve. In particular, SingleLayer often predicts a low probability for true triples and hence has a lot of false negatives.

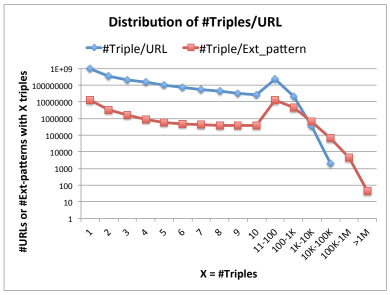


Figure 5: Distribution of #Triples per URL or extraction pattern motivates SPLITANDMERGE.

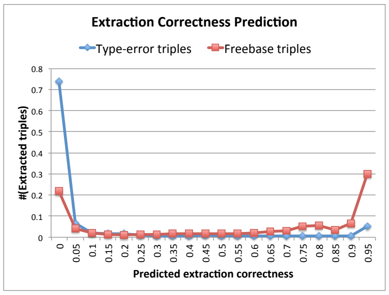


Figure 6: Distribution of predicted extraction correctness shows effectiveness of MULTILAYER+.

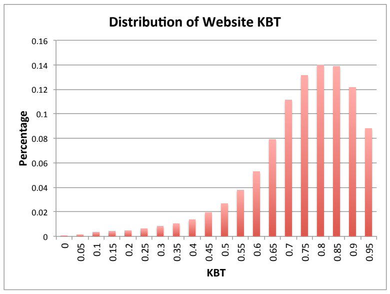


Figure 7: Distribution on KBT for websites with at least 5 extracted triples.

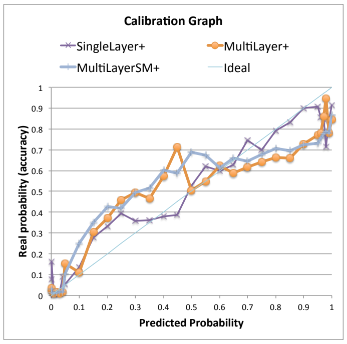


Figure 8: Calibration curves for various methods on KV data.

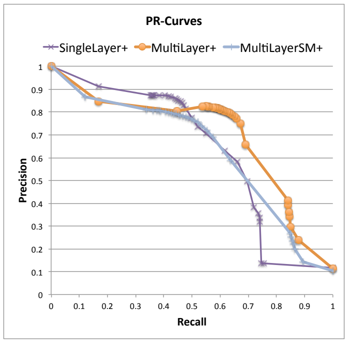


Figure 9: PR-curves for various methods on KV data. MULTILAYER+ has the best curve.

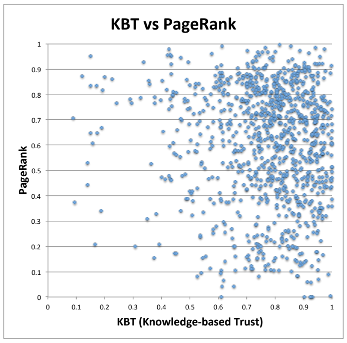


Figure 10: KBT and PageRank are orthogonal signals.

We see that MULTILAYERSM has better results than MULTILAYER, but surprisingly, MULTILAYERSM+ has lower performance than MULTILAYER+. That is, there is an interaction between the granularity of the sources and the way we initialize their accuracy.

The reason for this is as follows. When we initialize source and extractor quality using default values, we are using unsupervised learning (no labeled data). In this regime, MULTILAYERSM merges small sources so it can better predict their quality, which is why it is better than standard MULTILAYER. Now consider when we initialize source and extractor quality using the gold standard; in this case, we are essentially using semi-supervised learning. Smart initialization helps the most when we use a fine granularity for sources and extractors, since in such cases we often have much fewer data for a source or an extractor.

Finally, to examine the quality of our prediction on extraction correctness (recall that we lack a full gold standard), we plotted the distribution of the predictions on triples with type errors (ideally we wish to predict a probability of 0 for them) and on correct triples (presumably a lot of them, though not all, would be correctly extracted and we shall predict a high probability). Figure 6 shows the results by MULTILAYER+. We observe that for the triples with type errors, MULTILAYER+ predicts a probability below 0.1 for 80% of them and a probability above 0.7 for only 8%; in contrast, for the correct triples in Freebase, MULTILAYER+ predicts a probability below 0.1 for 26% of them and a probability above 0.7 for 54%, showing effectiveness of our model.

5.3.3 *Effects of varying the inference algorithm*

Table 6 shows the effect of changing different pieces of the multi-layer inference algorithm, as follows.

Row $p(V_d|\hat{C}_d)$ shows the change we incur by treating C_d as observed data when inferring V_d (as described in Section 3.3.2), as opposed to using the confidence-weighted version in Section 3.3.3. We see a significant drop in the AUC-PR metric and an increase in SqV by ignoring uncertainty in C_d; indeed, we predict a probability below 0.05 for the truthfulness of 93% triples.

Table 6: Contribution of different components, where significantly worse values (compared to MULTILAYER+) are shown in italics.

SqV	SqV	WDev	AUC-PR	Cov
MULTILAYER+	0.054	0.0040	0.693	0.864
$p(V_d\|\hat{C}_d)$	*0.061*	0.0038	*0.570*	0.880
Not updating α	0.055	*0.0057*	0.699	0.864
$p(C_{dwv}\|\mathbb{I}(\overline{X}_{ewdv} > \phi))$	0.053	0.0040	0.696	0.864

Row “Not updating α” shows the change we incur if we keep $p(C_{wdv} = 1)$ fixed at α, as opposed to using the updating scheme described in Section 3.3.4. We see that most metrics are the same, but WDev is significantly worse, showing that the probabilities are less well calibrated. It turns out that not updating the prior often results in over-confidence when computing $p(V_d|X)$, as shown in Example 3.3.

Row $p(C_{dwv}|\mathbb{I}(\overline{X}_{ewdv} > \phi))$ shows the change we incur by thresholding the confidence-weighted extractions at a threshold of $\phi = 0$, as opposed to using the confidence-weighted extension in Section 3.5. Rather surprisingly, we see that thresholding seems to work slightly better; however, this is consistent with previous observations that some extractors can be bad at predicting confidence [11].

5.3.4 *Computational efficiency*

All the algorithms were implemented in FlumeJava [6], which is based on Map-Reduce. Absolute running times can vary dramatically depending on how many machines we use. Therefore, Table 7 shows only the relative efficiency of the algorithms. We reported the time for preparation, including applying splitting and merging on web sources and on extractors; and the time for iteration, including computing extraction correctness, computing triple truthfulness, computing source accuracy, and computing extractor quality.

Table 7: Relative running time, where we consider one iteration of MULTILAYER as taking 1 unit of time. We see that using split and split-merge is, on average, 3 times faster per iteration.

Task		Normal	Split	Split&Merge
Prep.	Source	0	0.28	0.5
	Extractor	0	0.50	0.46
	Total	*0*	*0.779*	*1.034*
Iter.	I. ExtCorr	0.097	0.098	0.094
	II. TriplePr	0.098	0.079	0.087
	III. SrcAccu	0.105	0.080	0.074
	IV. ExtQuality	0.700	0.082	0.074
	Total	*1*	*0.337*	*0.329*
Total		5	2.466	2.679

For each component in the iterations, we report the average execution time among the five iterations. By default $m = 5, M = 10K$.

First, we observe that splitting large sources and extractors can significantly reduce execution time. In our data set some extractors extract a huge number of triples from some websites. Splitting such extractors has a speedup of 8.8 for extractor-quality computation. In addition, we observe that splitting large sources also reduces execution time by 20% for source-accuracy computation. On average each iteration has a speed up of 3. Although there is some overhead for splitting, the overall execution time dropped by half.

Second, we observe that applying merging in addition does not add much overhead. Although it increases preparation by 33%, it drops the execution time in each iteration slightly (by 2.4%) because there are fewer sources and extractors. The overall execution time increases over splitting by only 8.6%. Instead, a baseline strategy that starts with the coarsest granularity and then splits big sources and extractors slows down preparation by 3.8 times.

Finally, we examined the effect of the m and M parameters. We observe that varying M from 1K to 50K affects prediction quality very little; however, setting $M = 1K$ (more splitting) slows down preparation by 19% and setting $M = 50K$ (less splitting) slows down the inference by 21%, so both have longer execution time. On the other hand, increasing m to be above 5 does not change the performance much, while setting $m = 2$ (less merging) increases WDev by 29% and slows down inference by 14%.

5.4 Experiments related to KBT

We now evaluate how well we estimate the trustworthiness of webpages. Our data set contains 2B+ webpages from 26M websites. Among them, our multi-layer model believes that we have correctly extracted at least 5 triples from about 119M webpages and 5.6M websites. Figure 7 shows the distribution of KBT scores: we observed that the peak is at 0.8 and 52% of the websites have a KBT over 0.8.

5.4.1 KBT vs PageRank

Since we do not have ground truth on webpage quality, we compare our method to PageRank. We compute PageRank for all webpages on the web, and normalize the scores to $[0, 1]$. Figure 10 plots KBT and PageRank for 2000 randomly selected websites. As expected, the two signals are almost orthogonal. We next investigate the two cases where KBT differs significantly from PageRank.

Low PageRank but high KBT (bottom-right corner): To understand which sources may obtain high KBT, we randomly sampled 100 websites whose KBT is above 0.9. The number of extracted triples from each website varies from hundreds to millions. For each website we considered the top 3 predicates and randomly selected from these predicates 10 triples where the probability of the extraction being correct is above 0.8. We manually evaluated each website according to the following 4 criteria.

- *Triple correctness*: whether at least 9 triples are correct.
- *Extraction correctness*: whether at least 9 triples are correctly extracted (and hence we can evaluate the website according to what it really states).
- *Topic relevance*: we decide the major topics for the website according to the website name and the introduction in the "About us" page; we then decide whether at least 9 triples are relevant to these topics (*e.g.*, if the website is about business directories in South America but the extractions are about cities and countries in SA, we consider them as not topic relevant).
- *Non-trivialness*: we decide whether the sampled triples state non-trivial facts (*e.g.*, if most sampled triples from a Hindi movie website state that the language of the movie is Hindi, we consider it as trivial).

We consider a website as truly trustworthy if it satisfies all of the four criteria. Among the 100 websites, 85 are considered trustworthy; 2 are not topic relevant, 12 do not have enough non-trivial triples, and 2 have more than 1 extraction errors (one website has two issues). However, only 20 out of the 85 trustworthy sites have a PageRank over 0.5. This shows that KBT can identify sources with trustworthy data, even though they are tail sources with low PageRanks.

High PageRank but low KBT (top-left corner): We consider the 15 gossip websites listed in [16]. Among them, 14 have a PageRank among top 15% of the websites, since such websites are often popular. However, for all of them the KBT are in the bottom 50%; in other words, they are considered less trustworthy than half of the websites. Another kind of websites that often get low KBT are forum websites. For instance, we discovered that *answers.yahoo.com* says that *"Catherine Zeta-Jones is from New Zealand"* [3], although she was born in Wales according to *Wikipedia*[4].

5.4.2 Discussions

Although we have seen that KBT seems to provide a useful signal about trustworthiness, which is orthogonal to more traditional signals such as PageRank, our experiments also show places for further improvement as future work.

1. To avoid evaluating KBT on topic irrelevant triples, we need to identify the main topics of a website, and filter triples whose entity or predicate is not relevant to these topics.
2. To avoid evaluating KBT on trivial extracted triples, we need to decide whether the information in a triple is trivial. One possibility is to consider a predicate with a very low variety of objects as less informative. Another possibility is to associate triples with an IDF (inverse document frequency), such that low-IDF triples get lower weight in KBT computation.
3. Our extractors (and most state-of-the-art extractors) still have limited extraction capabilities and this limits our ability to estimate KBT for all websites. We wish to increase our KBT coverage by extending our method to handle open-IE style information extraction techniques, which do not conform to a schema [14]. However, although these methods can extract more triples, they may introduce more noise.
4. Some websites scrape data from other websites. Identifying such websites requires techniques such as copy detection. Scaling up copy detection techniques, such as [7, 8],

[3] https://answers.yahoo.com/question/index?qid=20070206090808AAC54nH.

[4] http://en.wikipedia.org/wiki/Catherine_Zeta-Jones.

has been attempted in [23], but more work is required before these methods can be applied to analyzing extracted data from billions of web sources.

5. Finally, there have been many other signals such as PageRank, visit history, spaminess for evaluating web-source quality. Combining KBT with those signals would be important future work.

6. RELATED WORK

There has been a lot of work studying how to assess quality of web sources. PageRank [4] and Authority-hub analysis [19] consider signals from link analysis (surveyed in [3]). EigenTrust [18] and TrustMe [28] consider signals from source behavior in a P2P network. Web topology [5], TrustRank [17], and AntiTrust [20] detect web spams. The knowledge-based trustworthiness we propose in this paper is different from all of them in that it considers an important *endogenous* signal–the correctness of the factual information provided by a web source.

Our work is relevant to the body of work in *Data fusion* (surveyed in [1, 12, 23]), where the goal is to resolve conflicts from data provided by multiple sources and find the truths that are consistent with the real world. Most of the recent work in this area considers trustworthiness of sources, measured by link-based measures [24, 25], IR-based measures [29], accuracy-based measures [8, 9, 13, 21, 27, 30], and graphical-model analysis [26, 31, 33, 32]. However, these papers do not model the concept of an extractor, and hence they cannot distinguish an unreliable source from an unreliable extractor.

Graphical models have been proposed to solve the data fusion problem [26, 31, 32, 33]. These models are more or less similar to our single-layer model in Section 2.2; in particular, [26] considers single truth, [32] considers numerical values, [33] allows multiple truths, and [31] considers correlations between the sources. These prior works do not model the concept of an extractor, and hence they cannot capture the fact that sources and extractors introduce qualitatively different kinds of noise. In addition, the data sets used in their experiments are typically 5-6 orders of magnitude smaller in scale than ours, and their inference algorithms are inherently slower than our algorithm.

Finally, the most relevant work is our previous work on knowledge fusion [11]. We have given detailed comparison in Section 2.3, as well as empirical comparison in Section 5, showing that MultiLayer improves over SingleLayer for knowledge fusion and gives the opportunity of evaluating KBT for web-source quality.

7. CONCLUSIONS

This paper proposes a new metric for evaluating web-source quality–knowledge-based trust. We proposed a sophisticated probabilistic model that jointly estimates the correctness of extractions and source data, and the trustworthiness of sources. In addition, we presented an algorithm that dynamically decides the level of granularity for each source. Experimental results have shown both promise in evaluating web-source quality and improvement over existing techniques for knowledge fusion.

8. REFERENCES

[1] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 41(1):1–41, 2008.

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

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

[4] 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.

[5] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. In *SIGIR*, 2007.

[6] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. R. Henry, R. Bradshaw, and N. Weizenbaum. Flumejava: Easy, efficient data-parallel pipelines. In *PLDI*, pages 363–375, 2010.

[7] X. L. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava. Global detection of complex copying relationships between sources. *PVLDB*, 2010.

[8] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1), 2009.

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

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

[11] X. L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From data fusion to knowledge fusion. *PVLDB*, 2014.

[12] X. L. Dong and F. Naumann. Data fusion–resolving data conflicts for integration. *PVLDB*, 2009.

[13] X. L. Dong, B. Saha, and D. Srivastava. Less is more: Selecting sources wisely for integration. *PVLDB*, 6, 2013.

[14] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: the second generation. In *IJCAI*, 2011.

[15] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422, 2013.

[16] Top 15 most popular celebrity gossip websites. http://www.ebizmba.com/articles/gossip-websites, 2014.

[17] Z. Gyngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *VLDB*, pages 576–587, 2014.

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

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

[20] V. Krishnan and R. Raj. Web spam detection with anti-trust rank. In *AIRWeb*, 2006.

[21] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198, 2014.

[22] X. Li, X. L. Dong, K. B. Lyons, W. Meng, and D. Srivastava. Truth finding on the Deep Web: Is the problem solved? *PVLDB*, 6(2), 2013.

[23] X. Li, X. L. Dong, K. B. Lyons, W. Meng, and D. Srivastava. Scaling up copy detection. In *ICDE*, 2015.

[24] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). In *COLING*, pages 877–885, 2010.

[25] J. Pasternack and D. Roth. Making better informed trust decisions with generalized fact-finding. In *IJCAI*, pages 2324–2329, 2011.

[26] J. Pasternack and D. Roth. Latent credibility analysis. In *WWW*, 2013.

[27] R. Pochampally, A. D. Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *Sigmod*, 2014.

[28] 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.

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

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

[31] X. Yin and W. Tan. Semi-supervised truth discovery. In *WWW*, pages 217–226, 2011.

[32] B. Zhao and J. Han. A probabilistic model for estimating real-valued truth from conflicting sources. In *QDB*, 2012.

[33] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han. A Bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB*, 5(6):550–561, 2012.