Corroborating Facts from Affirmative Statements

Minji Wu
Department of Computer Science
Rutgers University
minji-wu@cs.rutgers.edu

Amélie Marian
Department of Computer Science
Rutgers University
amelie@cs.rutgers.edu

ABSTRACT

Web sources often provide different and even conflicting information. Simple voting-based strategies have already shown limitations at identifying the correct answer to a user query with the presence of unreliable sources. In order to identify the correct answer, corroboration techniques have been proposed and proved to be effective for such tasks. In this paper, we investigate the corroboration problem in which most or all facts have only affirmative statements from sources. A fact is either true or false, and an affirmative statement from a source indicates its support for a fact being true. Unfortunately, state-of-the-art corroboration techniques rely on conflicting information to differentiate the trustworthiness of the sources and we demonstrate their limitations in our scenario. Different from existing techniques that consider a single trust score for each source, we propose a novel algorithm that utilizes a multi-value trust score toward different subsets of facts. By considering the information entropy of the unknown facts, our algorithm incrementally evaluates facts and updates the estimates on the trust scores for the sources. We conduct experiments using both synthetic and real-world datasets and demonstrate that our algorithm significantly outperforms existing approaches in precision and accuracy.

1. INTRODUCTION

Web sources often provide different and even conflicting information. Users trying to find answers to their information needs are faced with the task of identifying the correct answer among the available data. A simple method for identifying the correct information is to take the majority vote as the correct answer. Unfortunately, with the pervasive presence of low quality sources, the correct answer is often out-voted by incorrect ones. As an example, consider a query that asks for 'the total government revenue of Japan in 2011'. Several sources (including CIA Factbook and quandl.com) reported \$1.8 trillion. The correct answer \$1.1 trillion[1], which was only found in Wikipedia, is outvoted by incorrect ones. In fact, Wikipedia itself provides 2 conflicting numbers (\$1.1 and \$1.97 trillion) in separate pages which further complicates the task of identifying the correct answer. As such, in the presence of multiple conflicting answers, a strategy that takes into account the quality of sources needs to be developed to identify the correct one. However, in a scenario in which multiple sources agree on the same answer, it is still not certain that the answer is correct, as illustrated in the following example.

EXAMPLE 1: *Consider we want to identify a list of restaurants that are up and running in a certain region. There exist several web sources that provide valuable information for this task. For instance, local search engines such as* `Yellowpages` *and* `Citysearch` *provide business listings including restaurants. Social web sites such as* `Yelp` *and* `Foursquare` *allow users to check-in at dining venues. Most of the restaurant listings on these web sources are hints that the restaurants exist (except for those listed as 'CLOSED'). However, the fact that a restaurant is listed at one or several of these web sources is not definite evidence that it is still open. As an example, consider a restaurant named 'Danny's Grand Sea Palace' located at '346 West 46th St, New York', which is backed by both* `Yellowpages` *and* `Citysearch`*. A follow-up check*[2] *revealed that the restaurant is no longer in business and that the listings were inaccurate.*

In this paper, we investigate the problem of identifying the veracity of facts in the presence of mostly affirmative statements. A fact is either true or false, and an affirmative statement indicates support from a source that the fact is true. Intuitively, for a fact with only affirmative statements, there should be no ambiguity that it is true, since there is no suggestion from any source that the fact may be false. However, as we see in Example 1, this is not necessarily the case. Although there are two affirmative statements for the fact *'Danny's Grand Sea Palace is open'*, it is still factually false. In addition to the above example, we also observe similar cases in other domains. For instance, technology blogs usually provide claims regarding major product releases, each of which could be viewed as facts with only supportive statements.

In such a problem, the main difficulty is to correctly identify false facts, since facts with only affirmative statements appear to be true. In principle, a false fact could be revealed

[1]We obtained this number from the web page of Japan's finance ministry.
[2]The restaurant is one of a set of restaurants that we checked in person.

10.5441/002/edbt.2014.15

if its affirmative statements are from sources with low trustworthiness. Unfortunately, in a scenario where most facts have affirmative statements only, it is hard to compute the correct trustworthiness of the sources. We listed below the challenges of the problem we focus in this paper.

- **Quality of sources** Information on the Internet is fast changing and goes out-dated fast. For a certain task, there might not exist a source that is fresh and yet with good coverage. A serious website such as `Yelp` which allow users to post authentic reviews contains erroneous restaurant listings. This is different from applications for which existing corroboration techniques have been successful. For instance [21], `imdb` is a near-perfect source for the *movie* dataset. Assessing the quality of the source is critical to derive the correctness of the facts it reports.

- **Apparent consensus** The principle of corroboration is to differentiate the sources' quality, hence treating the information from each source differently. Existing corroboration techniques work well in tasks with conflicting statements because they can dampen the trust scores for the sources that have incorrect statements. Unfortunately, this is not the case in our scenario since sources provide mostly same statements (*i.e.*, affirmative statements). As a result, it is extremely difficult to identify any errors from the sources.

To tackle these challenges, we propose a novel corroboration algorithm that uses a multi-value trust score for the sources. Each fact is evaluated using one of the trust values from each source. Unlike with a single trust score for the sources where all facts would have the same corroboration result in a scenario where most or all facts have only affirmative statements, we can correctly identify false facts by considering a lower trust score for the sources reporting them. Intuitively, a source may have different trustworthiness on different sets of facts, and our algorithm leverages such observation to improve corroboration quality. We summarize our contributions as follows.

- We investigate the problem of corroborating facts in the presence of mostly affirmative statements and demonstrate the limitations of state-of-the-art methods.

- We propose a novel corroboration method that adopts a multi-value trust score for each source; each fact is evaluated using one set of source trust values.

- Our corroboration algorithm incrementally selects facts by considering the information entropy in the unprocessed facts and updates the trust scores for the sources.

- We conduct experiments over synthetic and real world datasets and show that our algorithm significantly outperforms existing approach on precision and accuracy.

To the best of our knowledge, our corroboration algorithm is the first to consider different trust scores from the same source for different sets of facts. In the following discussion, we show that a multi-value trust score is not only effective, but necessary in the corroboration problem considered in this paper. The rest of the paper is organized as follows. A detailed motivating example is shown in Section 2. We formally define the corroboration problem in Section 3 and introduce the multi-value trust score strategy in Section 4. We present our incremental algorithm in Section 5. Experiment results are shown in Section 6. We discuss related work in Section 7. Finally, we conclude the paper in Section 8.

2. A MOTIVATING EXAMPLE

We use an instance of Example 1 as the motivating example to illustrate the limitation of existing methods. Consider a scenario with 5 sources $\{s_1, s_2, s_3, s_4, s_5\}$ and 12 restaurant listings $\{r_1, ..., r_{12}\}$. For the ease of discussion, we use `T` and `F` votes to refer to affirmative and disagreeing statements. The votes from the sources are shown in Table 1, where in the last column we list if each restaurant is actually open (ground truth). A source can vote either *for* (`T`) (*e.g.*, by listing the restaurant) or *against* (`F`) a restaurant (*e.g.*, by listing the restaurant as `CLOSED`). A '-' indicates that a source does not list the restaurant. As shown, all the sources cast votes only for a subset of restaurants. In addition, most restaurants (except for r_6 and r_{12}) receive `T` votes only. If we know the correct result for each restaurant (as shown in the last column) a priori, it could be computed that the global trust scores for all the sources are {1, 0.8, 1, 0.5, 0.625}, respectively. In the following, we examine the performance of 2 state-of-the-art corroboration techniques.

	s_1	s_2	s_3	s_4	s_5	*correct value*
r_1	-	T	-	T	-	true
r_2	T	T	-	T	T	true
r_3	T	-	T	-	T	true
r_4	-	-	-	T	T	false
r_5	T	-	-	T	-	false
r_6	-	-	F	T	-	false
r_7	-	T	-	T	T	true
r_8	-	T	-	T	T	true
r_9	-	-	T	-	T	true
r_{10}	-	-	-	T	T	false
r_{11}	-	-	T	T	T	true
r_{12}	-	F	F	T	-	false

Table 1: A scenario with 5 sources and 12 restaurants

2.1 The TwoEstimate Algorithm

Galland et al. [8] introduced a set of iterative algorithms that are very related to our corroboration task. Among those, the `TwoEstimate` algorithm is directly applicable to our scenario[3]. The `TwoEstimate` works by iteratively estimating the probability that each restaurant is open and the trustworthiness of the sources until convergence is reached. A direct application of the `TwoEstimate` algorithm on the motivating example yields a result of *true* for all the restaurants except for r_{12}, and a trust score of {1, 1, 0.8, 0.9, 1} for the 5 sources, respectively.

Although the `TwoEstimate` algorithm has a recall of 1, the precision and accuracy are only 0.64 and 0.67 respectively.

[3]Note that although the `ThreeEstimate` algorithm has shown better performance, it calculates a measure using the number of `T` and `F` votes for each fact. Since for most restaurants there are `T` votes only, the `ThreeEstimate` algorithm essentially simplifies to the `TwoEstimate` algorithm in this scenario.

The reason for the result can be explained as follows. First, since the majority of the restaurants only have `T` votes, the only possible corroboration outcome for restaurants other than r_6 and r_{12} is *true*. In addition, consider r_6 with a `T` vote from s_4 and an `F` vote from s_3. Although there is one `F` vote, the `T` vote is from s_4 which has more correct votes for other restaurants than s_3. In a sense, the `F` vote is 'out-voted' by the `T` vote since s_4 has a higher trust score than s_3. Second, in order to guarantee convergence, the `TwoEstimate` normalizes the probability of a restaurant or the trustworthiness of a source to 1 if it is greater than or equal to 0.5. This normalization process essentially translates a restaurant with uncertainty into an absolute `T` or `F` and then uses it as feedback for the calculation of its sources. The effect of this *reinforcement* mechanism is greatly amplified in our scenario since, there is little conflict for the vast majority of the restaurants and consequently, sources receive a near-perfect trust score.

2.2 The BayesEstimate Algorithm

Zhao et al. [21] proposed a Bayesian probabilistic graphical model (termed as `BayesEstimate`) that infer true facts and source quality. Instead of using a single value for the trustworthiness, the `BayesEstimate` algorithm leverages two-sided errors (number of false positive and false negative) of each source. In essence, the `BayesEstimate` algorithm is tailored for real world corroboration tasks in which the algorithm has some prior knowledge about the source quality (*e.g.*, high precision but low recall). In our scenario, although we do have some sources that match such profile, we have other sources with relatively poor precision (*e.g.*, s_4). In addition, the `BayesEstimate` algorithm also suffers from the fact that there is little conflict for most of the restaurants, and hence has similar corroboration result as the `TwoEstimate` algorithm. Using the `BayesEstimate` algorithm we obtain a result of *true* for all restaurants, which translates to a precision of 0.58 and recall of 1. The reason that `BayesEstimate` did not identify r_{12} as false is because it considers a high-precision-low-recall prior, and therefore giving `F` vote less weight.

2.3 Our strategy

Consider a simplified version of our strategy, which does not apply to all the restaurants at once. Instead, it divides the corroboration task into 3 sub-tasks that are carried out in 3 rounds, as shown in Figure 1. We start our algorithm with a default trust value (*e.g.*, 0.9) for each source and pick restaurants r_9 and r_{12} to process. By using the default trust scores, our method computes a corroborated result of `true` and `false` for the restaurants, respectively. In addition, the trust scores for the sources are then computed as {-, 1, 1, 0, 1}. During the second round, we choose $\{r_5, r_6\}$ (the shaded objects in Figure 1 refer to the restaurants which have been evaluated), which results in `false` for the two restaurants. Note that although we have `T` votes from s_4 for both restaurants, since it has a trust score of 0 from the first round, the corroboration assigns a low score for both restaurants. The trust scores for the sources are updated to {0, 1, 1, 0, 1}. During the last round, the corroboration is applied to the rest of the restaurants and results in *true* for all the remaining restaurants due to the fact that each restaurant is backed by at least one of the "*good*" sources (s_2, s_3, s_5). Overall, the sources have a trust score of {0.67, 1, 1, 0.7, 1} respectively and the corroboration results in a precision of 0.78 and a recall of 1.

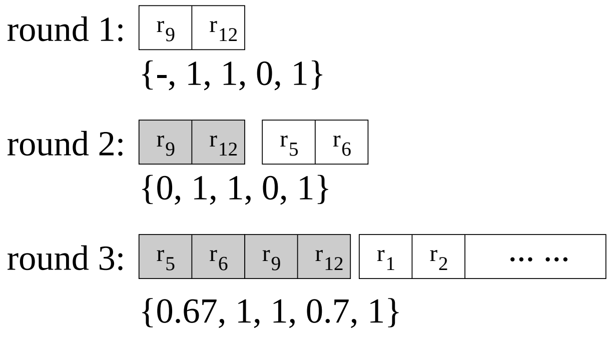


Figure 1: Illustration of our strategy

The rationale behind considering a multi-round corroboration strategy can be explained as follows. In order to identify as many corrupt listings (restaurants that are no longer open or are not at the address) as possible, we need to have sources with low trust scores. However, in a scenario where all sources are generally good (*i.e.*, with a trust score above 0.5), it is impossible to find bad listings. By applying corroboration in a step-by-step fashion, we are able to obtain a low trust score for some sources over a subset of restaurants. For instance, the above strategy calculates a trust score of 0 for s_4 over r_{12}. During the second round, it aggressively selects all listings that are projected to be corrupt based on the current trust scores of the sources ($\{r_5, r_6\}$). In the third round, since all remaining restaurants are projected to be valid, it processes all of them and finish corroboration. Note that this strategy is a simplified version of our algorithm (Section 5), which adopts an entropy-driven method in selecting facts at each round.

	Precision	Recall	Accuracy
`TwoEstimate`	0.64	1	0.67
`BayesEstimate`	0.58	1	0.58
`Our strategy`	0.78	1	0.83

Table 2: Results of the strategies

Table 2 lists the results of the three methods described above on Example 1, where it shows that our strategy is advantageous compared with two state-of-the-art algorithms. The key difference between our strategy and existing methods is that existing approaches use the same trust measure (derived at the final iteration) to determine the correct value for all the restaurants, while our strategy groups restaurants that share similar features (source votes) and incrementally evaluates each restaurant group, one at a time, using the knowledge available *at the current iteration* to make correctness decisions.

3. PROBLEM FORMULATION

We consider a problem that consists a set of sources $\mathcal{S} = \{s_1, s_2, \cdots\}$ and a set of facts $\mathcal{F} = \{f_1, f_2, \cdots\}$. A fact could be either *true* (meaning it is correct) or *false* (erroneous). For instance, a fact could be "*A restaurant called 'M Bar' located at 12 W 44th St is a legitimate restaurant*".

3.1 Sources

We consider a source $s \in \mathcal{S}$ as a real-world object that expresses opinions about facts. A source may agree or disagree with a fact in the form of casting a true (`T`) vote or false (`F`) vote. For instance, a source may disagree with the legitimacy of a restaurant by listing it as `CLOSED`. We use $s(f)$ to denote the vote of source s over a fact f, illustrated below.

$$s(f) = \begin{cases} T, & \text{if } s \text{ agrees with } f \\ F, & \text{if } s \text{ disagrees with } f \\ -, & \text{if } s \text{ has no knowledge about } f \end{cases} \tag{1}$$

Note that in certain scenarios, an unknown vote (*i.e.*, the '-' vote) from a source may slightly indicate that it disagrees with the fact (*e.g.*, a source may delete a restaurant listing after it went out of business), we cannot differentiate such cases from cases where the source has no knowledge about the object.

We associate with each source s a trustworthiness score $\sigma(s)$ which represents its precision. The trust score is a real number between 0 and 1, with 1 indicating a perfect source and 0 indicating a completely wrong source. We define sources with a trust score $\sigma(s)$ between 0.5 and 1 as *positive* sources. In principle, positive sources are the sources that have more correct votes than incorrect ones. Similarly, a *negative* source is defined as a source with a trust score between 0 and 0.5.

3.2 Facts

A fact $f \in \mathcal{F}$ is an expression over a real-world object that is of interest to the users. A fact is either true or false. In order to estimate the correct value (*i.e.*, true or false) of a fact f, we propose techniques to compute a probability $\sigma(f)$ which represents the likelihood that f is true. A fact with a probability of 1 (or 0) is a true (or false) fact. A corroboration algorithm determines the value of a fact f if $\sigma(f)$ is greater than a certain threshold. In this paper, we use 0.5 as the threshold value, shown below.

$$f = \begin{cases} true, & \text{if } \sigma(f) \geq 0.5 \\ false, & \text{if } \sigma(f) < 0.5 \end{cases} \tag{2}$$

Entropy of unknown facts: In information theory [3], the entropy is a measure of uncertainty of a random variable. Since we consider $\sigma(f)$ the probability of a fact f being true, we can calculate the entropy $H(f)$ of the unknown fact f as follows.

$$H(f) = -\sigma(f) \cdot \log \sigma(f) - (1 - \sigma(f)) \cdot \log(1 - \sigma(f) \tag{3}$$

It is easy to see that a fact f has an entropy of 0 if its probability is 1 or 0 (*i.e.*, no uncertainty) and has the highest entropy of 1 if its probability of 0.5. Intuitively, we expect a fact to have an entropy between 0 and 1 given the votes from the sources. We discuss how to iteratively select facts by leveraging the fact entropy in the following section.

3.3 The corroboration problem

Given a set of sources $\mathcal{S}$ and a set of facts $\mathcal{F}$, the corroboration problem is to identify the correct value of each fact and estimate the trustworthiness of each source. In this paper, we focus on a corroboration problem in a specific scenario in which most facts in $\mathcal{F}$ only receive affirmative statements. More formally, let $\mathcal{F}^*$ be a subset of $\mathcal{F}$ such that for each fact $f \in \mathcal{F}^*$ there are only `T` votes only. We focus on a corroboration problem in which most facts in $\mathcal{F}$ are in $\mathcal{F}^*$ (*i.e.*, $|\mathcal{F}^*| \gg |\mathcal{F} - \mathcal{F}^*|$).

4. TRUST SCORES OF SOURCES

Our corroboration algorithm is built upon the concept of a multi-value trust score for each source. In the following discussions, we first formally define the single-value trust score and multi-value trust score (Section 4.1). We then demonstrate the limitation of an algorithm using a single-value trust score (Section 4.2). We finally present our method of implementing multi-value trust scores (Section 4.3).

4.1 Definition

Traditional corroboration techniques [18, 15, 8, 10, 21, 19] usually calculate a trust score for each source that is used to evaluate facts. In such a setting, the same trust score of each source is used to evaluate *every* fact. There are exceptions in which a technique may consider more than one trust score for each source. For instance, the `BayesEstimate` method considers a two-sided errors for each source which capture both the false positive and false negative rates. However, the same measures for a source are used to evaluate *every* fact for which the source casts a vote.

Formally, we define a *single-value trust score* that is used in existing techniques as *one* measure $\sigma(s)$ that is computed for each source s. The measure $\sigma(s)$ is used to evaluate each fact $\{f | s(f) \in \{T, F\}\}$ that s casts vote. Note that such measure $\sigma(s)$ could contain more than one value (*e.g.*, `BayesEstimate`). In this following, we focus our discussion on the case where $\sigma(s)$ is a single value.

In contrast, we propose to use a *multi-value trust score* in our corroboration algorithm. A *multi-value trust score* is defined as a group of values assigned to each source, as shown below.

$$\boldsymbol{\sigma}(s) =< \sigma_1(s), \sigma_2(s), \cdots, > \tag{4}$$

where we call $\sigma_i(s)$ one of the trust values of source s. In such a setting, each fact f is evaluated using one of the trust values of $\boldsymbol{\sigma}(s)$ of sources that have voted for f. Consider 2 facts f_1, f_2 for which a source s has a `T` vote. A single-value based algorithm evaluates both facts use the same measure $\sigma(s)$, while a multi-value based algorithm may use different trust values of $\boldsymbol{\sigma}(s)$.

As an example, assume we adopt the scoring used in the `TwoEstimate` [8] to compute the probability for facts. Formally, let $f_i \in \mathcal{F}^*$ be a fact with only `T` votes and $\mathcal{S}_i^+$ be the set of sources that have a `T` vote for f_i. We use $p(\boldsymbol{\sigma}(s))$ to denote the function that picks a trust value from $\boldsymbol{\sigma}(s)$. A multi-value trust score based algorithm computes the probability of f as follows.

$$\sigma(f_i) = \frac{\sum_{s \in \mathcal{S}_i^+} p(\boldsymbol{\sigma}(s))}{|\mathcal{S}_i^+|} \tag{5}$$

4.2 Single-value trust score

A single-value trust score based algorithm works by iteratively estimate the probability of facts and the trust score for the sources. As shown in Equation 6 and 7, the probability of facts is calculated using the trust score for the sources from the previous iteration (the `Corrob` method). In return, the trust score for the sources is updated using the probability of the facts (the Update method). The algorithm terminates once convergence is reached.

$$\sigma^{(k)}(f_i) = Corrob(\sigma(\mathcal{S}_i^{(k-1)})) \tag{6}$$

$$\sigma^{(k)}(s_i) = Update(\sigma^{(k)}(\mathcal{F}_i)) \tag{7}$$

In a regular corroboration task in which there exist conflicting votes (*i.e.*, both Ts and Fs), a single-value trust score often works because incorrect votes can be identified based on the corroborated result of each fact. For instance, consider a fact f with a T vote from s_1, s_2 and an F vote from s_3. Assume that the right result (*true*) for f has been derived by the corroboration algorithm. Since s_3 has the incorrect vote, its trust score is discounted and is reflected in the corroboration of other facts. However, in a scenario where most facts have T votes only, it is difficult to identify any incorrect votes.

Let A be an iterative corroboration algorithm using a single-value trust score. Let us simulate the procedure of A and explain that after applying A, all the facts $f \in \mathcal{F}^*$ have the same corroboration result and all the sources have near perfect (or completely wrong) trust scores. Suppose the algorithm starts with an initial trust score λ (*e.g.*, 0.9) for each source. A computes the probability for each fact using the $Corrob()$ operation. Since for each $f \in \mathcal{F}^*$ there are T votes only and sources have a high initial trust score, facts in $\mathcal{F}^*$ receive a high probability. This is based on the assumption that if a fact is vouched by a number of accurate sources, it is likely to be true. In return, A then updates the trust score for each source based on the calculated probabilities of the facts. Recall that most of the facts are in $\mathcal{F}^*$, *i.e.*, $|\mathcal{F}^*| \gg |\mathcal{F} - \mathcal{F}^*|$. Therefore the computation of $\sigma(s)$ is dominated by the probabilities of facts in $\mathcal{F}^*$. Since A considers that each source has the correct vote for each fact $f \in \mathcal{F}^*$, it assigns a high trust score to each source. This is based on the assumption that the more correct votes a source has, the more trustworthy it is. In addition, in order to avoid converging to a local optima (*i.e.*, all sources have a trust score of 0.5), a common fix is to use a normalization process that converts the probability to 1 (or 0) if it is above or equal to (or less than) 0.5. Once A converges, it results in *true* for each fact $f \in \mathcal{F}^*$ and a trust score close to 1 for each source.

From the information entropy perspective, such result indicates that the entropy of all unknown facts is 0 since for each fact f the probability is 1, and therefore we have $H(f)$ = 0. In other words, a single-value based method dismisses the uncertainty of facts and considers them true with a probability of 1. This result is counter-intuitive as we expect that in real life scenarios, each source has a trust score between 0 and 1, and each fact has a level of uncertainty quantified by its entropy $H(f)$.

4.3 Multi-value trust score

Since a method that uses a single measure to evaluate all facts does not work well for our scenario, we now resort to a strategy that processes unknown facts separately. However, we have to address two fundamental challenges: 1) how do we calculate the trust values for each source; and 2) for each fact, how do we select the trust values from each source (*i.e.*, the $p(\boldsymbol{\sigma}(s))$ function) to compute its correct value.

Both challenges above can be addressed by incrementally evaluating facts and updating the trust score for the sources. We repeatedly select a subset of the facts to process and update the trust value that represents the trust score over the facts that have been evaluated. During each round, we use the latest trust value for the sources and leverage heuristics in selecting unevaluated facts. We formally define the incrementally calculated trust score as follows.

Definition 1 (Incrementally calculated trust score). *Consider $\{t_0, t_1, \cdots, t_m\}$ be a set of finite time points. We define an incrementally calculated trust score for source s as $\boldsymbol{\sigma}(s) = \{\sigma_0(s), \sigma_1(s), \cdots, \}$ where $\sigma_i(s)$ is the trust score of s at time t_i. At each time point t_i, a subset of facts F_i is selected for evaluation. The facts in F_i are evaluated using the trust scores $\sigma_i(\mathcal{S})$=$\{\sigma_i(s_1), \sigma_i(s_2), \ldots, \}$. In return, we update the trust score of the sources to $\sigma_{i+1}(\mathcal{S})$ by incorporating the corroboration result of the facts in F_i. Let $t(f)$ denote the time point at which f is selected. In essence, the trust score $\sigma_i(\mathcal{S})$ at time t_i represents the trustworthiness of the sources over the facts $\{f|t(f) < t_i\}$ that have been evaluated up to t_i. When the algorithm terminates at t_m, the probability $\sigma(f)$ is used to determine the corroborated result of each fact.*

The advantage of using an incrementally calculated trust score is two-fold. First, it enables us to incrementally calculate the trust values for each source. Second, the function to choose a trust value from the sources for facts $p(\boldsymbol{\sigma}(s))$ can be set to $\sigma_i(\mathcal{S})$ at time t_i, By incrementally calculating the trust score for each source, the challenge is now how to select facts at each time point so that the correct result could be computed for as many facts as possible. We detail this process in the following section.

5. CORROBORATION

In this section, we investigate strategies of selecting facts at each time point and present our corroboration algorithm (denoted as IncEstimate). In the following discussion, we assume the scoring of the TwoEstimate algorithm (Equation 5) is used. We first introduce our fact selecting strategy (namely IncEstHeu) in Section 5.1. We then present the IncEstimate algorithm int Section 5.2. We analyze the complexity of IncEstHeu in Section 5.3.

5.1 Selecting facts

Recall that in Section 4 we showed the main limitation of existing algorithms is that they use a universal trust score for each source and incorrectly dismisses the uncertainty of facts. In order to uncover the correct value of unknown facts, the key challenge is to evaluate each fact f at a point t_i such that $\sigma(\mathcal{S})$ is a more accurate measure for the facts.

Let $\bar{\mathcal{F}}_i \subseteq \mathcal{F}$ be the set of facts that have not been evaluated at t_i and $\sigma_i(\mathcal{S})$ be the trust values for the sources. It is yet difficult to decide a set of facts $F_i \in \bar{\mathcal{F}}_i$ such that $\sigma_i(\mathcal{S})$ is accurate for F_i. Now let us again look at the problem from the information entropy perspective. Since we know that the entropy for an unknown fact is 0 if its probability is 1 (or 0), we model the fact selection problem as a problem to maximize the collective entropy $H(\bar{\mathcal{F}}_i)$ of unknown facts (the sum of entropy of each fact in $\bar{\mathcal{F}}$).

One possible greedy strategy is to select facts with the highest entropy at each t_i. However, such a strategy does not necessarily maximize $H(\bar{\mathcal{F}}_i)$ since the selected F_i would impact the trust values $\sigma(\mathcal{S})$ for the sources. In turn, the updated trust values would affect the entropy of the remaining facts $\bar{\mathcal{F}}_i - F_i$. For instance, suppose we select r_1 (which has the highest entropy of 1) at round 2 in the motivating example. Such a selection results in $\sigma(\mathcal{S}) = \{-, 1, 1, 0.5, -\}$ which would decrease the entropy of remaining facts. As a consequence, we would be unable to identify the false facts r_4 and r_{10}.

Given a set of unknown facts $\bar{\mathcal{F}}_i$, it is easy to see that there exist $2^{|\bar{\mathcal{F}}_i|}$ ways of selecting facts at t_i. However, it is computationally expensive to explore all possibility so as to maximize $H(\bar{\mathcal{F}}_i)$. We approach the problem by selecting a set of facts such that the updated trust values $\sigma_{i+1}(\mathcal{S})$ is unlikely to decrease the entropy of the remaining facts. Consider below 2 cases of trust values $\sigma_{i+1}(\mathcal{S})$ after evaluating F_i.

- $\sigma_{i+1}(s_j) > \sigma_i(s_j)$ for each j or
- $\sigma_{i+1}(s_j) < \sigma_i(s_j)$ for each j

For simplicity, let us assume that all the remaining facts $\bar{\mathcal{F}}_i$ have `T` votes only ($\bar{\mathcal{F}}_i \subseteq \mathcal{F}^*$). Consider the case in which by selecting F_i the trust value increases for each source (*i.e.*, case 1). Since the probability of facts is calculated as the average trust scores of its sources (Equation 5) and a higher trust value for the sources would increase the probability of the facts, the entropy decreases for facts with a probability above 0.5 (recall a fact has the highest entropy if it has a probability of 0.5). On the other hand, the updated trust value increases the entropy for the facts with a probability smaller than 0.5 and $\sigma_{i+1}(\mathcal{S})$ brings its probability closer to 0.5 than $\sigma_i(\mathcal{S})$. Similarly, a smaller trust value for the sources would bring down the entropy for facts with a probability smaller than 0.5 and could raise the entropy for facts with a probability greater than 0.5.

Now let us examine the relationship between facts and the trust value changes. We observe that if the evaluation results of F_i are true (or false), the trust value for the sources increase (or decrease). This is based on the intuition that the more correct votes a source has, the more trustworthy it is. Let $\sigma_{i+1}(s)$ be the trust value for source s after evaluating F_i and for each $f \in F_i$ the corroboration result is true, $\sigma_{i+1}(s)$ can be calculated as follows.

$$\begin{aligned}\sigma_{i+1}(s) &= \frac{\sum_{f_j \in \hat{\mathcal{F}}_i} \sigma(f_j)}{|\hat{\mathcal{F}}_i|} \\ &= \frac{\sum_{f_j \in \hat{\mathcal{F}}_{i-1}} \sigma(f_j) + \sum_{f_j \in F_i} \sigma(f_j)}{|\hat{\mathcal{F}}_{i-1}| + |F_i|} \\ &> \frac{\sum_{f_j \in \hat{\mathcal{F}}_{i-1}} \sigma(f_j)}{|\hat{\mathcal{F}}_{i-1}|} = \sigma_i(s) \qquad (8)\end{aligned}$$

where $\hat{\mathcal{F}}_{i+1}$ refers to the facts that have been evaluated up to t_i. Note that the above calculations consider the probability to be 1 for true facts.

Based on the discussions above, we now have a viable strategy. We first group unevaluated facts based on the sources of the votes. Facts in the same group receive votes from the same set of sources. The intuition behind is that facts with the same votes should have the same corroboration result. As an instance in the motivating example, r_5 and r_8 are grouped together since they have the same votes. We then calculate a score $\Delta H(\bar{\mathcal{F}})$ for each fact group FG that represents the entropy change for the remaining facts if FG is selected. We rank fact groups in decreasing order of their $\Delta H(\bar{\mathcal{F}})$ scores and pick the one with the highest score.

$$\Delta H(\bar{\mathcal{F}})_{FG} = \sum_{FG' \in \bar{\mathcal{F}} - FG} (H_{i+1}(\bar{\mathcal{F}})_{FG'} - H_i(\bar{\mathcal{F}})_{FG'}) \qquad (9)$$

There is one special case in which given $\sigma(\mathcal{S})$, all remaining facts have a probability above (or below) 0.5. In this case, the facts would be evaluated to be true (or false) which is equivalently as having a entropy of 0 (recall true facts have a normalized probability of 1). Such a scenario could be caused if `IncEstHeu` repeatedly selects facts that evaluated to be true facts. To avoid this effect, we slightly modify our strategy as follows. During each time point t_i, we divide fact groups into positive part (fact groups with probability above 0.5) and negative part (fact group with probability below 0.5). We then pick one fact group from each part with the highest $\Delta H(\bar{\mathcal{F}})$ score. In addition, we require that the same number of facts are selected from each group. Let FG_i^+ and FG_i^- denote the positive and negative fact group, and we use $size(FG)$ to denote the number of facts of group FG. `IncEstHeu` selects n facts from each group where $n = min\{size(FG_i^+), size(FG_i^-)\}$. The rationale behind is that as FG_i^+ and FG_i^- become extremely disparate in sizes, the updated trust scores are dominated by the larger fact group.

5.2 The Algorithm

Algorithm 1 Incremental Estimate (`IncEstimate`)

Input: $\mathcal{F}$: a collection of facts; $\mathcal{S}$: a set of sources
Output: $\sigma(\mathcal{S})$: estimations for the sources, $\sigma(\mathcal{F})$: estimations for the facts

```
1: Initialize $\sigma_0(\mathcal{S})$, $\sigma(\mathcal{F})$
2: while $|\bar{\mathcal{F}}| > 0$ do
3:     $F_i \leftarrow Select\_Facts(\bar{\mathcal{F}}, \sigma_i(\mathcal{S}))$
4:     $\bar{\mathcal{F}} \leftarrow \bar{\mathcal{F}} - F_i$
5:     for all $f \in F_i$ do
6:         $\sigma(f) \leftarrow Corrob(f, \sigma_i(\mathcal{S}))$
7:         $\mathcal{W} \leftarrow \mathcal{W} \cup f$
8:     end for
9:     $\sigma_{i+1}(\mathcal{S}) \leftarrow Update\_Trust(\mathcal{W})$
10: end while
11: return $\sigma(\mathcal{S}), \sigma(\mathcal{F})$
```

Algorithm 1 demonstrate the overall flow of our `IncEstimate` algorithm. Our `IncEstimate` takes a set of facts and a set of sources as input, and output the estimations of the trust scores of the sources as well as the probabilities of the facts. At first, we initialize $\sigma_0(\mathcal{S})$ and $\sigma(\mathcal{F})$ with a default value (*e.g.*, 0.9). Our algorithm then repeatedly selects a new subset of facts in each round. During each round (line 2-10), the new set of facts are selected using the $Select\_Facts(\bar{\mathcal{F}}, \sigma_i(\mathcal{S}))$ function (line 3, defined in Algorithm 2). The corroboration calculates the probability for each selected fact (line 6) and inserts it into a set $\mathcal{W}$ that contains facts which have been evaluated (line 7). The trust scores of the sources are then updated incorporating the results of facts that have been selected. Our algorithm terminates when all facts have been evaluated and returns $\sigma(\mathcal{S})$ and $\sigma(\mathcal{F})$.

We briefly discuss the `IncEstHeu` strategy defined in Algorithm 2. At first, the `IncEstHeu` strategy initializes a set $\mathcal{W}$ which is the set of facts that are to be selected, and $\mathcal{P}$

Algorithm 2 *Select_Facts*$(\bar{\mathcal{F}}, \sigma(\mathcal{S}))$

```
1: IncEstHeu:
2:     $\mathcal{W} \leftarrow \emptyset, \mathcal{N} \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset$
3:     $\mathcal{P} \leftarrow$ all fact groups $FG$ in $\bar{\mathcal{F}}$ s.t. $\sigma(FG) > 0.5$
4:     $\mathcal{N} \leftarrow \bar{\mathcal{F}} - \mathcal{P}$
5:     sort $\mathcal{P}, \mathcal{N}$ in decreasing order of $\Delta H(\bar{\mathcal{F}})$
6:     $FG^{+} \leftarrow peek(\mathcal{P})$  $FG^{-} \leftarrow peek(\mathcal{N})$
7:     $n \leftarrow min\{size(FG^{+}), size(FG^{-})\}$
8:     for $i = 1 \rightarrow n$ do
9:         $\mathcal{W} \leftarrow \mathcal{W} \cup peek(FG^{+}) \cup peek(FG^{-})$
10:    end for
11:    return $\mathcal{W}$
```

and $\mathcal{N}$ which represents the set of positive and negative fact groups respectively (line 5). The set $\mathcal{P}$ and $\mathcal{N}$ are then filled with positive and negative facts that have not been evaluated (line 6 and 7). Note that the *peek*$(\mathcal{P})$ function pops the first elements from $\mathcal{P}$. We then sort the set $\mathcal{P}$ and $\mathcal{N}$ based on their $\Delta H(\bar{\mathcal{F}})$ in decreasing order (line 8). From $\mathcal{P}$ and $\mathcal{N}$, we pick the fact group that has the highest $\Delta H(\bar{\mathcal{F}})$ score, denote as FG^{+} and FG^{-} respectively (line 9). We select n facts from each group where n is the number of facts of the smaller group between FG^{+} and FG^{-} (line 10-13).

5.3 Complexity analysis

In this section, we analyze the complexity of our IncEstimate algorithm. As a comparison, a voting based method simply counts the number of votes for each fact and therefore incurs a cost of $\Theta(|\mathcal{F}|)$. Methods that iteratively computes the scores for the facts and sources have a cost of $\Theta(m(|\mathcal{F}|+|\mathcal{S}|))$, where m is the number of iterations needed for convergence.

Our IncEstimate algorithm incurs additional cost when calculating projected scores for unevaluated facts and updating trust scores for the sources at each time point. Let t_m be the number of time points IncEstimate needs to evaluate all facts. As we see before, IncEstHeu evaluates at least one fact group at each time point, therefore we can bound t_m by the number of fact groups in $\mathcal{F}$. Recall that a vote takes a value from $\{T, F, -\}$, therefore the maximum number of fact groups is $3^{|\mathcal{S}|} - 2|\mathcal{S}| - 1$ (we excluded fact groups with only one vote or no vote). Further, since we focus on scenarios where most facts receive T votes only, the bound for the number of fact groups can be reduced to $O(2^{|\mathcal{S}-\mathcal{S}^*|} \cdot 3^{|\mathcal{S}^*|})$ where $\mathcal{S}^*$ is the set of sources that cast F votes ($\mathcal{S}^* < \mathcal{S}$).

At each time point, IncEstHeu calculates projected scores for unevaluated facts and update trust scores for the sources. In the worse case scenario, each of the fact groups evaluated at t_1 through t_{m-1} contains one fact, and the fact groups evaluated at t_m contains $|\mathcal{F}| - 2(t_m - 1)$ facts. The total cost on calculating projected score is therefore $t_m \cdot (t_m - 1) + (|\mathcal{F}| - 2t_m + 2) \cdot t_m$. In the best case scenario, the majority facts $|\mathcal{F}| - 2(t_m - 1)$ are evaluated at t_1 and 2 facts are evaluated from t_2 through t_m, which brings the cost to $(|\mathcal{F}| - 2t_m + 2) + t_m \cdot (t_m + 1)$. On the average case where the number of facts in fact groups is uniformly distributed, the cost is $\frac{|\mathcal{F}|(t_m+1)}{2}$. By adding the cost for calculating trust scores for the sources, the total cost for IncEstimate can be bounded by $O(|\mathcal{F}| \cdot 2^{\mathcal{S}-\mathcal{S}^*} \cdot 3^{\mathcal{S}^*})$.

While the cost of IncEstimate is exponential in the number of sources, in typical cases this number is small. In addition, whether adding more sources results in better corroboration quality is still an open question [14]. Moreover, t_m is also bounded by the number of facts $|\mathcal{F}|$ since at least one fact is evaluated at each time point, and therefore the cost for IncEstimate can be bounded by a polynomial term $O(|\mathcal{F}|^2)$. Our experimental results in Section 6.2.5 show the overhead of a more sophisticated algorithm is acceptable in exchange for better corroboration results.

6. EXPERIMENTS

In this section, we present our experimental results. Section 6.1 describes the experiments setup. We first present our results on the real-world dataset, the restaurant application mentioned before in Section 6.2. We then show the experimental results on synthetic datasets in Section 6.3.

6.1 Setup

In this section, we present our experiment setup and evaluation metrics, as well as the algorithms we implemented for comparison.

6.1.1 *Algorithms*

[**Baseline methods**]: We provided two baseline approaches named Counting and Voting. The Counting method assigns a *true* result to each fact if more than half the sources report it true. In contrast, the Voting method considers a fact as *true* if there exist more sources reporting it true than false.

[**Corroboration methods**]: We implemented the strategy IncEstHeu of our incremental algorithm introduced in Section 5. We used a default trust score $\sigma(\mathcal{S})$ of 0.9 for each source to start our algorithm. We tested other default values and we observed all default value above 0.5 generate the same corroboration result. This is because despite different $\sigma_0(\mathcal{S})$ used, the same facts are selected at t_0, and therefore they result in the same trust value at t_1. We are also interested in how a different fact selection strategy would impact the performance of IncEstimate. To that end, we implemented IncEstPS, a simple strategy that selects the fact group with the highest probability at each time point. The rationale behind is that facts with higher probability are more likely to receive correct corroboration results. Compared with a balanced strategy IncEstHeu that considers both positive and negative facts, we want to see how a naive greedy strategy performs in the competition. We also implemented the TwoEstimate algorithm [8] and the BayesEstimate algorithm [21] for comparison. For the BayesEstimate algorithm, we used the same assumption as in [21] that sources have low false positive rate but high false negative rate. In particular, we set α_0 = (100, 10000), α_1 = (50, 50), and β = (10, 10).

[**ML-based methods**]: Since our problem can be naturally seen as a classification problem, we also tested machine learning based algorithms using the votes as features. In particular, we tested 2 classifiers using Weka: a SVM classifier (using SMO implementation) and a logistic classifier with default parameter. We report the results using 10-fold cross validation.

6.1.2 *Environment and Metrics*

We implemented all the algorithms using Java SDK 6. All the experiments were conducted on a Mac OS 10.8.2 with a quad-core CPU of 3.3 GHz and 8GB Ram. We use the following metrics to evaluate the results of various algorithms.

Precision, Recall, Accuracy: We first report standard information retrieval metrics to evaluate the results of all algorithms.
Mean square error of trust score (MSE): We use $t(s_i)$ to denote the trustworthiness of source s_i over a sampled golden set, and let $\sigma(s_i)$ denote the computed trust score of s_i by a corroboration algorithm. The mean square error of trust score is computed as follows.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(t(s_i) - \sigma(s_i))^2 \qquad (10)$$

6.2 Real-World Dataset

We report our experiment results over real world datasets in this section.

6.2.1 Dataset

We used the restaurant example discussed in Section 2 in our experimental evaluation. We used the same example in [15] and reported results of existing techniques on a small sample of restaurant listings. In this study, we expanded our investigation and conducted experiments in a much larger scale. We crawled data from six major sources for restaurant listings[4], namely Yellowpages, Foursquare, Menupages, Opentable, Citysearch, and Yelp. Some of the web sources allow accesses to the list of restaurant listings at a given a location (*e.g.*, Manhattan), while for others we have to do random accesses to probe as many listings as possible. In this particular experiment, we consider restaurant listings in the greater New York City area. Our initial crawling yielded 42969 restaurant listings but contains numerous duplicates due to various presentation of the same listing. In order to clean the data and remove duplicates, we employ the following strategy. We first wrote a rule-based script to normalize the addresses of all listings. Listings that share the same address are then grouped together. We calculate a similarity score between each pair of listings within a group and we consider two listings are the same entities if their similarity score is above a certain threshold. For the purpose of this project, we adopted the cosine similarity score at the term level as well as 3-gram level and used a threshold of 0.8. After the deduplication process, we recorded 36916 restaurant listings from these 6 sources[5]. Among those, only 654 listings (<2%) have F votes from sources. More specifically, F votes come from 3 sources, Foursquare (10), Menupages (256), and Yelp (425).

Table 3 reports source coverage (the fraction of the total restaurant listings contained in each source), as well as the source overlap (a measure of how much two sources have in common). As shown, all sources contain only a fraction of the entire listings. Among all sources, 2 sources (*i.e.*, Yellowpages, Citysearch) have significantly more coverage (>50%) compared with others.
Golden set: In order to evaluate the performance of various algorithms, we must create a golden set of listings for which we know for certain their correct value (true or false). Unfortunately, there does not exist an authoritative source that can provide such information. To that end, we selected restaurant listings from 3 zip codes and investigated their legitimacy in person[6]. Overall, our golden set contains 601 listings, out of which 340 are true and 261 are false. We also calculated the accuracies of all sources in the golden set, listed in Table 3. Unsurprisingly, sources that have direct connection with restaurants (*e.g.*, OpenTable and Menupages) have high accuracies (>0.9). Interestingly, the two sources with significantly higher coverage (Yellowpages, Citysearch) are also the sources with low accuracy (∼0.6).

Identifying legitimate restaurants is not a trivial task. Before we embarked on designing a corroboration algorithm for this task, we tried to leverage the reviews information from some of the sources to predict whether a restaurant listing is legitimate. In particular, we used a variety of meta data (number of reviews, average interval of review time stamp, length since last review, etc) as well as the review content as features and tested using a SVM classifier. However, the classifier resulted in a less-than-satisfactory accuracy (< 0.7).

6.2.2 Corroboration quality

	Precision	Recall	Accuracy	F-1
`Voting`	0.65	1.00	0.66	0.79
`Counting`	0.94	0.65	0.76	0.77
`BayesEstimate`	0.63	1.00	0.67	0.77
`TwoEstimate`	0.65	1.00	0.66	0.79
`ML-SVM (SMO)`	0.98	0.74	0.77	0.84
`ML-Logistic`	0.86	0.85	0.82	0.82
`IncEstPS`	0.66	1.00	0.68	0.79
`IncEstHeu`	0.86	0.86	**0.83**	**0.86**

Table 4: Result of real-world dataset

Table 4 lists the performance of various algorithms as well as the `Counting` and `Voting` methods. Since for most of the listings there exist only `T` votes, the `Voting` method assigns them a *true* result, thus results in a perfect recall value (1.0) but a low precision (0.65). In contrast, the `Counting` method uses a high threshold to filter out listings with insufficient `T` votes, hence has a high precision (0.94). However, the threshold is high enough to reject a lot of legitimate listings, therefore resulting in a low recall value (0.65). The two non-corroboration based approaches have an accuracy of 0.66 and 0.76, respectively.

Existing corroboration-based approaches do not perform much better than the two baseline approaches. As they cannot leverage conflicting information from the data, the `TwoEstimate` algorithm has almost the same result as the `Voting` method by assigning a *true* result to every listing except for a small set for which there are more `F` votes than `T` votes. The `BayesEstimate` algorithm also has very similar results as `TwoEstimate`. In addition to suffering from the lack of conflicting votes among facts, the `BayesEstimate` algorithm relies heavily on the prior knowledge regarding the sources. The *high-precision low-recall* prior that is used by `BayesEstimate` is clearly different from the actual trustworthiness of the sources, as we see in Table 3 that both Yellowpages and Citysearch have poor precisions.

The two machine learning based algorithms perform noticeably better than both baseline and existing corroboration methods. In particular, the support vector classifier

[4]We comducted the crawling in Feb 2012.
[5]http://www.cs.rutgers.edu/∼amelie/RelaxedQ/datasets/Restaurants/
[6]The in-person check-up was conducted during Apr 2012.

Source coverage	YellowPages	Foursquare	MenuPages	OpenTable	CitySearch	Yelp
	0.59	0.24	0.20	0.07	0.50	0.35
Source overlap	**YellowPages**	**Foursquare**	**MenuPages**	**OpenTable**	**CitySearch**	**Yelp**
YellowPages	1	0.22	0.18	0.04	0.43	0.26
Foursquare	0.22	1	0.30	0.08	0.22	0.29
MenuPages	0.18	0.30	1	0.09	0.17	0.29
OpenTable	0.04	0.08	0.09	1	0.05	0.07
CitySearch	0.43	0.22	0.17	0.05	1	0.27
Yelp	0.26	0.29	0.29	0.07	0.27	1
Source accuracy	**YellowPages**	**Foursquare**	**MenuPages**	**OpenTable**	**CitySearch**	**Yelp**
	0.59	0.78	0.93	0.96	0.62	0.84

Table 3: Source coverage and overlap

improves on both accuracy (0.77) and recall (0.84). In comparison, the logistic classifier proves to be a better model in this case, with a precision of 0.86 and an accuracy of 0.82. Unsurprisingly, the most discriminating features are the F votes from the 3 sources. The performance gain of machine learning algorithms is largely due to the consideration of missing votes among sources. As we discussed earlier, a missing vote could be seen as either a F vote or that a source has no knowledge about the fact. By taking advantage of the missing votes, machine learning based algorithms can leverage extra knowledge and therefore improve accuracy of predictions.

Our IncEstHeu strategy significantly outperforms the baseline and existing corroboration methods, and slightly improve on accuracy and recall compared with machine learning based algorithms. The improvement is statistical significant for both baseline and existing corroboration techniques (with p-value < 0.001). The IncEstPS strategy has a similar result as existing approaches and improves on accuracy only marginally. This is due to the way IncEstPS selects facts at each time point. We observe IncEstPS repeatedly selects facts with high probability which are evaluated to be true. As a consequence, the trust values remain high and most of the facts are evaluated to be true except for a few with more F votes than T votes. The IncEstHeu has a good balance between precision and recall, results in the best value for accuracy and F-1. In particular, IncEstHeu results in more true negatives (141 in the golden set). The best machine learning method (ML-Logistic) has 137 true negatives.

Although the improvement of our IncEstHeu over the machine learning based approaches is not statistically significant, we argue that our method is advantageous in such task. For one thing, our approach does not require any training data, which could be difficult to obtain in certain applications. In addition, the machine learning methods are trained using a small dataset and it is unclear whether it would scale to a larger unseen dataset due to the fact the model could be overfitting over the small golden set.

6.2.3 *Mean square error*

Table 5 lists the corroborated trust scores of the sources of various algorithms as well as their MSEs. For IncEstHeu we report the trust scores for the sources at the end of last time point, which reflects their trustworthiness over the entire dataset. Compared with the actual source accuracy over the golden set, the IncEstHeu is clearly the best performer (almost identical trust score for Menupages, Opentable), thus results in the smallest MSE (0.005) among all corroboration techniques. This is due to the fact that it adapts its trust value for each fact group. The TwoEstimate algorithm, which is unable to identify most illegitimate listings, concludes all the sources as perfect or near perfect sources. Similar as precision and recall, the BayesEsimate algorithm assign a trust score to each source similar to TwoEstimate. The machine learning method ML-Logistic has the best MSE value overall, slightly outperform our best strategy. This is because the machine learning methods are specifically trained using the golden set. In addition, our method reports the trust score at the end of the last round, which represents the trust score over the entire dataset, and therefore it is not surprising that it deviates from the trustworthiness of the sources on the golden set.

6.2.4 *Multi-value trust score*

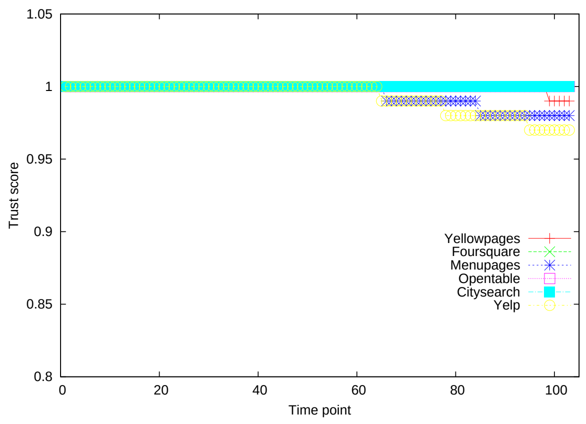
In this section, we illustrate how the trust score for each source changes when using different strategies of the IncEstimate algorithm. Figure 2 plots, for each strategy, different trust scores that are used for corroboration at each time point.

Since we use an initial trust score $\sigma(\mathcal{S})_0$ for each source, all sources share the same trust score at t_0. As the IncEstimate algorithm continues, each strategy develops different trust score trajectory for the sources. In particular, the IncEstPS strategy (Figure 2(a)) chooses the set of facts with the highest probability which are evaluated to be true. In return, the true facts boost the trust score for the sources that cast votes. Since IncEstPS favors facts with high probability, the trust scores for the sources remain at 1 until all facts with only T votes have been evaluated. It is not surprising that from then on, trust values for sources with F votes start to decrease since facts with F votes are evaluated to be true. Eventually, IncEstPS is only able to identify 2 true negatives, which is similar as existing corroboration techniques.

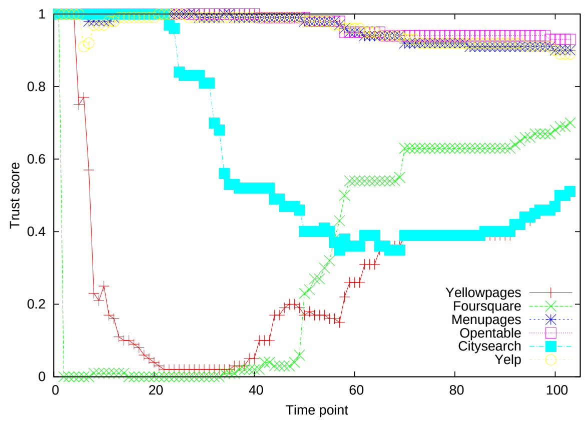
In contrast, the IncEstHeu strategy overcomes the limitation of IncEstPS by selecting both positive and negative listings during each round. This results in significantly different trust score change from the IncEstPS strategy (Figure 2(b)) While after evaluating F_0 all sources are positive sources, the trust scores for both Citysearch and Yellowpages begin to dip as more false facts with F votes are identified, which effectively makes them negative sources (after t_{12}). With the presence of negative sources, IncEstHeu is able to

	YellowPages	Foursquare	MenuPages	OpenTable	CitySearch	Yelp	MSE
Source accuracy	0.59	0.78	0.93	0.96	0.62	0.84	-
`TwoEstimate`	1.00	1.00	0.98	1.00	1.00	0.98	0.063
`BayesEsitmate`	1.00	1.00	1.00	1.00	1.00	1.00	0.066
`ML-Logistic`	0.62	0.85	0.98	0.92	0.65	0.95	**0.004**
`IncEstHeu`	0.51	0.70	0.90	0.93	0.51	0.89	0.005

Table 5: The mean square error of trust score



(a) `IncEstPS`



(b) `IncEstHeu`

Figure 2: Multi-value trust score at each time point

	Time cost (secs)
`Voting`	0.60
`Counting`	0.61
`BayesEstimate`	7.38
`TwoEstimate`	0.69
`ML-SMO`	0.99
`ML-Logistic`	0.91
`IncEstPS`	1.13
`IncEstHeu`	1.15

Table 6: Time cost of various algorithms

	Number of errors
`Voting`	292
`Counting`	327
`TwoEstimate`	269
`ThreeEstimate`	270
`IncEstHeu`	262

Table 7: Results of various algorithms over the Hubdub dataset

uncover false facts from $\mathcal{F}^*$ which explains a significantly higher number of true negatives. `IncEstHeu` then continues to evaluate facts and the trust scores eventually converges to the actual accuracy for the sources.

6.2.5 *Time cost*

Inevitably, a more sophisticated corroboration algorithm incurs additional time cost in computation. Our `IncEstimate` algorithm suffers from the overhead of the multiple round corroboration. Table 6 lists the time cost of various algorithms over the real world dataset. We used the ‘time’ command to test the time cost of each algorithm and report the *real* part. The two baseline approaches, `Voting` and `Counting`, which only considers the number of `T` and `F` votes, are the fastest ones, with a time cost of 0.6 and 0.61 seconds, respectively. The `TwoEstimate` algorithm, which applies corroboration on all the listings at once, is also fairly efficient, with a time cost of 0.69 seconds. The `BayesEstimate` algorithm requires a burning period before stabilizing and results in the longest time (7.38 secs). The two machine learning based approaches take less than 1 second largely due to the fact that they only run over the golden set. The best strategy of our `IncEstimate` algorithm results in a little more than 1 second, with the best performing strategy having an acceptable time cost of 1.15 secs.

6.2.6 *The Hubdub Dataset*

Our paper focuses on the scenario where most or all facts have only `T` votes. Nevertheless, we do not believe that our incremental algorithm is limited to such cases. To demonstrate the effectiveness of `IncEstimate` in dataset with ample conflicting votes, we use the Hubdub dataset from [8]. The Hubdub dataset was constructed using a snapshot of settled questions from hubdub.com, which contains 830 facts from 471 users on 357 questions.

Table 7 report the results of various algorithms on the Hubdub dataset. We did not include the machine learning based methods since this task involves more than two candidate answers. For comparison, we report the same metric used in [8], the number of errors (the sum of false positive and false negative). The best performance in [8] was from `TwoEstimate`, which recorded 269 errors. Our `IncEstHeu` outperforms all existing methods by reducing it to 262 errors. This proves that `IncEstHeu` is not only suitable for the corroboration problem discussed in this paper, but also effective in scenarios with conflicting statements.

6.3 Synthetic Dataset

We present our experiment results on synthetic datasets in this section. We first provide details on how we generate synthetic datasets (Section 6.3.1) and then present the corroborated results (Section 6.3.2).

6.3.1 Dataset

We use the following model to generate synthetic datasets. We consider all the sources are positive sources, *i.e.*, with a trust score of greater than 0.5. For each source s, let $\sigma(s)$ and $c(s)$ denote its trust score and coverage. For each fact, we randomly assign a correct value of either true or false. We also consider a factor η that determines the percentage of facts that have `F` votes. The parameters $\sigma(s)$ and $c(s)$ controls whether and how a source s casts votes on facts. Motivated by the observation in the real world dataset, we divide the sources into accurate sources (*e.g.*, Menupages) and inaccurate sources (*e.g.*, Yellowpages). In particular, we create sources as follows.

- **Accurate sources** are created with a trust score uniformly distributed in [0.7, 1.0]. In addition, each accurate source s is associated with a probability $m(s)$ that it casts a `F` vote for a false fact. We set $m(s)$ to be uniformly distributed in [0, 0.5].

- **Inaccurate sources** are created with a trust score uniformly distributed in [0.5, 0.7]. Inaccurate sources do not cast `F` votes for any fact.

We generate coverage for each source by following the intuition that inaccurate sources have a higher coverage compared with accurate sources. In particular, the coverage is calculated using Equation 11,

$$c(s) = 1 - \sigma(s) + random() * 0.2 \tag{11}$$

where $random()$ is a function that generates a random real number in [0, 1]. For each synthetic dataset we generate 20000 facts which are randomly assigned a true or false value.

6.3.2 Results

Figure 3 plots the performance comparison of various algorithms in the synthetic datasets. In particular, Figure 3(a) illustrates the accuracy of algorithms with a varying total number of sources. In this experiment, we fix the number of inaccurate sources at 2. As shown, our `IncEstHeu` algorithm consistently outperforms all other methods by a large margin. As the number of accurate sources increases, the accuracy of the `IncEstHeu` improves. In contrast, except for the `Counting` method, all methods remain almost flat as the number of sources change. The performance of existing algorithms is not unexpected. Although the majority sources are accurate, their low coverage, coupled with the fact there exist very few conflict votes, renders the state-of-the-art methods incapable of identifying false facts.

Figure 3(b) demonstrates the results under a varying number inaccurate sources, with the total number of sources fixed at 10. We are seeing similar results as shown in Figure 3(a). Unsurprisingly, as the number of inaccurate sources increases, the accuracy of the `IncEstHeu` decreases and eventually drops to the same level when 9 out 10 sources are inaccurate. Our `IncEstHeu` outperforms all other methods by as much as 37%.

Figure 3(c) shows the results with a different percentage η of facts that have `F` votes (from 0.01 to 0.05). We fix the number of total and inaccurate sources at 10 and 2 respectively. Again, the `IncEstHeu` algorithm generates significantly more accurate corroboration results than any other methods.

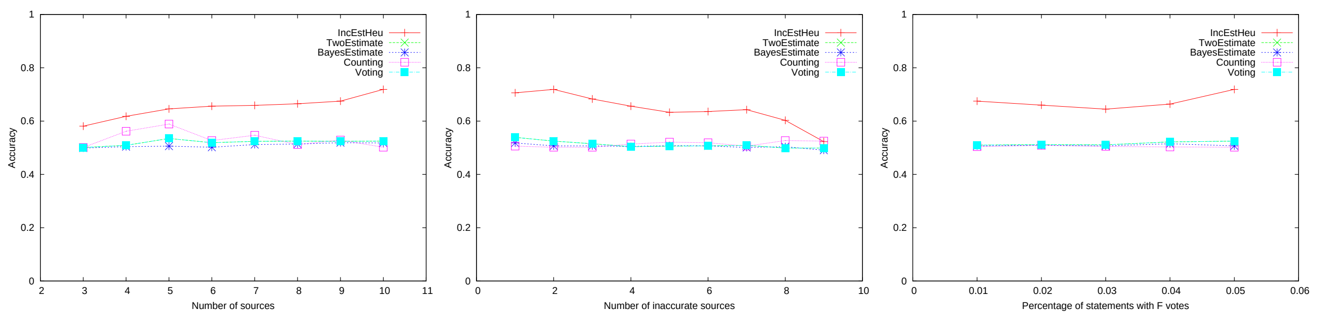
7. RELATED WORK

Corroboration was first used in the task of question answering in its early stage [1, 7, 18]. Different from traditional methods which only consider the frequency of extracted answers [6, 13], techniques that leverage corroboration consider the quality of the sources from which answers are extracted. For example, based on the link structure of the World Wide Web, one may use the PageRank score [2] or the authoritative score [11] as a measure for the source. However, such measures may not be available if only a small set of sources are accessible. The AskMSR system [1, 7] ranks answers based on the number of extraction rules or high quality query rewrites. In [18], Wu et al. presented a comprehensive corroboration approach that considers not only the originality of the sources but also the prominence of answers within each source.

More recently, corroboration techniques [20, 5, 8, 15, 16, 14] have also been used in finding the correct value among a set of conflicting values for an object (termed as the truth finding problem). Yin et al. [20] proposed a novel algorithm called `TruthFinder` that uses Bayesian analysis and finds true facts among conflicting information. Dong et al. [5] investigate dependence among sources and assign a higher weight to independent sources. Galland et al. [8] proposed the (`ThreeEstimate`) algorithm that considers not only the trustworthiness of the sources, but also how difficult each statement is in terms of the level of disagreement. Pasternack et al. [16] approach the truth finding problem by incorporating prior knowledge and propose a set of algorithms (`AvgLog, Invest and PooledInvest`). However, all the techniques target corroboration tasks with explicit uncertainty and therefore is ineffective in our problems with implicit uncertainty.

Corroboration has also been applied to applications in database area [12, 17]. In [12], the authors proposed a framework called Multiple Join Path for obtaining high quality information by linking fields across multiple databases with uncertainty. In [17], the authors extended the problem by considering a join graph over databases with low quality data and proposed a novel algorithm to efficiently compute the top-k answers.

Dong et al. [14] investigated a set of state-of-the-art corroboration techniques and concluded that there are possible improvements to information corroboration. In particular, the authors observed that fractions of data from the same source can have different quality and suggested that differentiating source quality for different categories of data could improve corroboration quality. To the best of our knowledge, we are the first to use a multi-value trust score for sources in a corroboration technique. By considering multiple trust scores of each source, we are able to uncover the



(a) Varying number of sources (b) Varying number of inaccurate sources (c) Varying percentage of F votes

Figure 3: Corroboration results of synthetic datasets

errors of sources in a scenario in which there exists little conflicting votes among sources. The `BayesEstimate` [21] algorithm considers a two-value trust score (*i.e.*, sensitivity and specificity). However, it still applies the same score of each source to all the statements which makes it ineffective as we showed in Section 6.

Another related area is fact checking that has been studied in the Semantic Web domain. Fact checking is the process of identifying the provenance of Web data represented using Resource Description Framework (RDF). Dividino et al. in [4] developed a generic framework that aims to retrieve multiple dimensions of meta knowledge (*e.g.*, source, time of validity, certainty) with respect to RDFs. In [9], Hartig and Zhao presented a framework that tracks the provenance for Web data. In comparison, data corroboration usually assumes the knowledge of provenance for the data and focuses on the computation of the trustworthiness for the sources and the probability for the data.

8. CONCLUSION

We studied the corroboration problem in a scenario in which there exists little conflicting information. We tackle the problem by proposing an algorithm based on a multi-value trust score for each source. For each source, we use a different trust score when evaluating different sets of statements. We leverage the entropy of unknown facts and derive strategies of choosing facts during each round in our algorithm. We conduct experiments on both synthetic and real-world datasets and demonstrate that our algorithm significantly outperforms state-of-the-art corroboration methods and improves accuracy over machine learning based approaches.

9. REFERENCES

[1] D. Azari, E. Horvitz, S. Dumais, and E. Brill. Web-based question answering: A decision-making perspective. In *Proc. of the 19th Conference on Uncertainty in Artificial Intelligence (UAI'03)*, 2003.

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

[3] T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley-Interscience, New York, NY, USA, 1991.

[4] R. Q. Dividino, S. Sizov, S. Staab, and B. Schueler. Querying for provenance, trust, uncertainty and other meta knowledge in rdf. *J. Web Sem.*, 7(3):204–219, 2009.

[5] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: The role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[6] D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, 2005.

[7] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *SIGIR*, 2002.

[8] A. Galland, S. Abiteboul, A. Marian, and P. Senellart. Corroborating information from disagreeing views. In *WSDM*, pages 131–140, 2010.

[9] O. Hartig and J. Zhao. Publishing and consuming provenance metadata on the web of linked data. In *IPAW*, pages 78–90, 2010.

[10] G. Kasneci, J. V. Gael, D. H. Stern, and T. Graepel. Cobayes: bayesian knowledge corroboration with assessors of unknown areas of expertise. In *WSDM*, pages 465–474, 2011.

[11] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[12] Y. Kotidis, A. Marian, and D. Srivastava. Circumventing data quality problems using multiple join paths. In *CleanDB*, 2006.

[13] C. C. T. Kwok, O. Etzioni, and D. S. Weld. Scaling question answering to the web. In *WWW*, 2001.

[14] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 6(1):550–561, 2013.

[15] A. Marian and M. Wu. Corroborating information from web sources. *IEEE Data Eng. Bull.*, 34(3):11–17, 2011.

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

[17] M. Wu, L. Berti-Equille, A. Marian, C. M. Procopiuc, and D. Srivastava. Processing top-k join queries. *PVLDB*, 3(1):860–870, 2010.

[18] M. Wu and A. Marian. A framework for corroborating answers from multiple web sources. *Inf. Syst.*, 36(2):431–449, 2011.

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

[20] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Trans. Knowl. Data Eng.*, 20(6):796–808, 2008.

[21] 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.