Validation of Network Reconciliation

Ya Le
Stanford University
Dept. of Statistics
yle@stanford.edu

Eric Bax
Yahoo Labs
baxhome@yahoo.com

Nicola Barbieri
Yahoo Labs
barbieri@yahoo-inc.com

David Garcia Soriano
Yahoo Labs
davidgs@yahoo-inc.com

Jitesh Mehta
Yahoo Inc.
jiteshm@yahoo-inc.com

James Li
Yahoo Labs
jamesyili@yahoo-inc.com

Abstract

Network reconciliation is the problem of identifying nodes in separate networks that represent the same entity, for example matching nodes across social networks that correspond to the same user. We introduce a technique to compute probably approximately correct (PAC) bounds on precision and recall for network reconciliation algorithms. The bounds require some verified matches, but those matches may be used to develop the algorithms. The bounds do not require knowledge of the network generation process, and they can supply confidence levels for individual matches.

1 Introduction

Network reconciliation is the problem of matching nodes from different networks correspond to the same entity. Successful network reconciliation combines information from multiple networks to produce richer data about the entities represented in the networks and the connections among them. For example, it is possible to get a more accurate understanding of a person and their relationships if both their social and business contacts can be identified by matching their profiles across the social networks Facebook and LinkedIn. Similarly, it is possible to better understand the mechanisms and interactions of cell biology if we can learn which substances play similar roles in different types of cells. Network reconciliation algorithms match nodes based on connectivity, node data, or both.

Matching based on connections is similar to the classical graph isomorphism problem, in which the goal is to find, if it exists, a one-to-one and onto mapping between the node sets of two graphs such that each pair of nodes in one graph share an edge if and only if their map images in the other graph share an edge [12]. Network reconciliation requires a less strict definition of matching than graph isomorphism, because, for example, for most people, their set of social contacts is not the same as their set of business contacts. So, instead of strict isomorphism, the goal in this example is to use the fact that for some people there is some overlap between their social and business contacts to match each node representing a person in a social network to any node representing the same person in a business network. So the goal is not to identify a strict isomorphism, but rather an accurate mapping, in the sense that it matches nodes that correspond to the same person with a high rate of success (high recall) and has a low rate of matching nodes that do not correspond to the same person (high precision).

Matching nodes based on node data is similar to solving a weighted bipartite matching problem, where each pair of elements with one from each of two sets has a score, and the goal is to identify a set of pairs that maximizes the sum of scores and has each element in at most one pair. In network reconciliation, the two sets are the sets of nodes from two networks, and the score for a pair of nodes may be based on similarities between their node data. (In practice, the node data may be extended

to include some data about the node's neighbors as well. Node data alone may be insufficient for effective network reconciliation because it may be incomplete, hidden from data gathering, or not specific enough: there are over 35,000 people named John Smith on LinkedIn.) Algorithms for weighted bipartite matching have time complexity greater than O(V^2), where V is the number of elements [9]. In big data settings, this can be prohibitive, so the problem must be relaxed, for example by allowing a node in one network to match multiple nodes in another, or heuristics must be applied to compute a nearly optimal solution.

Some network reconciliation algorithms are iterative. They establish some initial "seed matches" based on node data or by matching the highest-degree nodes in the different networks to each other [13]. Then they use shared connections to establish more matches, based on the idea that if more of the connections to a node in one network match connections to a node in another network, then the nodes are more likely to match. With each iteration, more matches may be established, offering opportunities for further matches. This iterative matching process is called percolation. Refer to [21] for some theory on how many seeds are needed for percolation to succeed. For more network reconciliation algorithms, refer to [19, 1, 22, 18, 8].

Our matching problem setting is similar to the transductive setting for classification [20], where there is a set of training examples with known inputs and class labels and a set of working examples with known inputs and unknown class labels, and the goal is to use the available training and working data to develop a classifier that classifies the working examples with a low error rate. For results on validation of network classifiers in transductive settings, refer to [15, 4]. For network reconciliation, we assume that we know network data, consisting of some node data and the links, for both networks involved in the matching, and our goal is to use network data to match nodes as accurately as possible between the networks.

This paper introduces a technique to compute probably approximately correct (PAC) bounds on the precision and recall of network reconciliation algorithms applied to big data. The technique does not require knowledge of how the networks were generated or rely on any specific model of network generation. The technique requires some verified matches. However, it produces valid bounds even if the verified matches are used to develop the matching algorithm. So we can use all available data for matching and also produce bounds on precision and recall.

This paper is organized as follows. Section 2 reviews bounds for expectations of random variables based on samples drawn without replacement, which we apply in later sections. Section 3 presents techniques to validate precision and recall for network reconciliation algorithms that produce matches for all nodes before validation is performed. We refer to these algorithms as *batch algorithms*. Section 4 presents techniques to validate network reconciliation algorithms that identify candidate matches for nodes on demand and for which it may be prohibitively expensive to compute candidate matches for all nodes in order to perform validation. We call these *query algorithms*. Section 5 shows how to extend the results to provide confidence levels for matches from different algorithms or from algorithms that provide match scores, to compute bounds on matching error rate rather then precision and recall, and to obtain a sample for validation as a subsample of a combined training and validation set. Section 6 concludes with some directions for future research.

2 Bounds for Sampling Without Replacement

The validation methods in later sections use averages over samples to estimate averages over larger sets, for example using a subset of matches to estimate the fraction of pairs of nodes with one from each network that are matches. Since the networks are finite, these are estimates based on random sampling without replacement to estimate means over finite sets of random variables. In machine learning, we often apply bounds for sampling with replacement to estimate means over infinite sets [10, 6, 2, 17]. There are similar bounds for our case; we review them in this section.

Let $\mathcal{X}$ be a size-n finite population and $\mathcal{S}$ be a size-s random sample drawn uniformly at random without replacement from $\mathcal{X}$. Let f denote a real-valued function defined on $\mathcal{X}$, i.e., $f : \mathcal{X} \to \mathbb{R}$. Assume that f is bounded: there exist a and b such that $a \leq f(x) \leq b, \forall x \in \mathcal{X}$.

The goal is to estimate the population mean:

$$\mu = \mathbb{E}_{X \sim \mathcal{U}(\mathcal{X})} \{f(X)\} = \frac{1}{n} \sum_{x \in \mathcal{X}} f(x) \tag{1}$$

and to compute PAC bounds for it, using the sample $\mathcal{S}$. (We use $\mathcal{U}()$ to denote a uniform distribution.) The sample mean:

$$\hat{\mu} = \mathbb{E}_{X \sim \mathcal{U}(\mathcal{S})} \{f(X)\} = \frac{1}{s} \sum_{x \in \mathcal{S}} f(x) \tag{2}$$

is an unbiased estimator for μ. Let $p^+(\mathcal{X}, \mathcal{S}, f, a, b, \delta)$ and $p^-(\mathcal{X}, \mathcal{S}, f, a, b, \delta)$ denote a PAC upper bound and a PAC lower bound on μ, with bound failure probability at most δ:

$$Pr\left\{\mu > p^+(\mathcal{X}, \mathcal{S}, f, a, b, \delta)\right\} \leq \delta, \tag{3}$$

$$Pr\left\{\mu < p^-(\mathcal{X}, \mathcal{S}, f, a, b, \delta)\right\} \leq \delta. \tag{4}$$

We will outline some methods to compute these bounds.

2.1 Hoeffding's inequality

Hoeffding [10] and Chvatal [7] show that the well-known Hoeffding inequality for sampling with replacement also applies to sampling without replacement:

$$p^+(\mathcal{X}, \mathcal{S}, f, a, b, \delta) = \hat{\mu} + (b-a)\sqrt{\frac{\ln 1/\delta}{2s}}, \tag{5}$$

$$p^-(\mathcal{X}, \mathcal{S}, f, a, b, \delta) = \hat{\mu} - (b-a)\sqrt{\frac{\ln 1/\delta}{2s}}. \tag{6}$$

Hoeffding and Chvatal also offer tighter bounds if the variance of f is known to be small. (The bounds above are based on a worst-case assumption about the variance of f.)

2.2 Empirical Bernstein-Serfling inequality

Audibert [2] developed and Maurer and Pontil [17] improved empirical Bernstein bounds (named for the Bernstein bound [5].) Empirical Bernstein bounds use the variance of f over the sample to allow the use of tighter versions of traditional bounds if the standard deviation of f is small compared to its range. This is common in validation of error rates, since error rates tend to be small in the cases of interest. Bardenet and Maillard [3] have developed similar bounds, called empirical Bernstein-Serfling bounds, for sampling without replacement:

$$p^+(\mathcal{X}, \mathcal{S}, f, a, b, \delta) = \hat{\mu} + \hat{\sigma}_s\sqrt{\frac{2\rho_s \log(5/\delta)}{s}} + \frac{\kappa(b-a)\log(5/\delta)}{s} \tag{7}$$

$$p^-(\mathcal{X}, \mathcal{S}, f, a, b, \delta) = \hat{\mu} - \hat{\sigma}_s\sqrt{\frac{2\rho_s \log(5/\delta)}{s}} - \frac{\kappa(b-a)\log(5/\delta)}{s}, \tag{8}$$

where

$$\rho_s = \begin{cases} 1 - (s-1)/n & \text{if } s \leq n/2; \\ (1 - s/n)(1 + 1/n) & \text{if } s > n/2, \end{cases} \tag{9}$$

$\hat{\sigma}_s^2 = \sum_{i,j=1}^{s}(x_i - x_j)^2/(2s^2)$ and $\kappa = \frac{7}{3} + \frac{3}{\sqrt{2}}$.

2.3 Direct computation

If f can only take values 0 or 1, then we can use direct computation to produce tight bounds. For sampling with replacement, we can use binomial tail inversion [11, 14]. For sampling without replacement, we can use hypergeometric tail inversion [7], as follows.

If f can only take value 0 or 1, then $s\hat{\mu} = \sum_{i=1}^{s} f(x_i)$ follows a hypergeometric distribution. Let $m = n\mu = \sum_{x \in \mathcal{X}} f(x)$. Then

$$Pr\left\{s\hat{\mu} = k\right\} = \binom{m}{k}\binom{n-m}{s-k}\Big/\binom{n}{s}. \tag{10}$$

The hypergeometric tail distribution is

$$H^{+}(m,n,s,k) = \sum_{j=k}^{s} Pr\left\{s\hat{\mu} = j\right\} = \sum_{j=k}^{s} \binom{m}{j}\binom{n-m}{s-j}\Big/\binom{n}{s}, \tag{11}$$

$$H^{-}(m,n,s,k) = \sum_{j=0}^{k} P\left\{s\hat{\mu} = j\right\} = \sum_{j=0}^{k} \binom{m}{j}\binom{n-m}{s-j}\Big/\binom{n}{s}. \tag{12}$$

So we have PAC bounds:

$$p^{+}(\mathcal{X},\mathcal{S},f,a,b,\delta) = \max\left\{\hat{p} : H^{-}(m,n,s,s\hat{p}) \geq \delta\right\} \tag{13}$$

$$p^{-}(\mathcal{X},\mathcal{S},f,a,b,\delta) = \min\left\{\hat{p} : H^{+}(m,n,s,s\hat{p}) \geq \delta\right\}, \tag{14}$$

which are tight bounds for μ, except for any rounding or approximation errors from the computation. (Use either Loader's method [16] or Stirling's approximation to compute H^{-} and H^{+}.)

3 Validation for Batch Algorithms

The next two sections present methods that use the single-variable bounds from the previous section to compute PAC bounds on precision and recall for network reconciliation algorithms. First, some notation. Let $\mathcal{X}$ and $\mathcal{Y}$ be the sets of nodes for two networks. Refer to x-y pairs with $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ that correspond to the same entity as *actual matches*. Refer to x-y pairs that an algorithm identifies as matches as *identified matches*.

In this paper, we distinguish between two types of algorithms for network reconciliation: batch algorithms and query algorithms. Batch algorithms compute all identified matches prior to validation. Query algorithms do not. Instead, query algorithms compute identified matches for $x \in \mathcal{X}$ upon request, and it may be infeasible for them to compute identified matches for all nodes in $\mathcal{X}$. Validation methods for batch algorithms can rely on knowledge of all identified matches; query algorithms cannot. This section covers validation of batch algorithms; the next section covers query algorithms.

We assume that we have a sample of nodes or node pairs for which the actual matches are known, called *validation data*. Initially, we derive bounds for matching algorithms developed independently of the validation data, which we call *holdout algorithms*. Then we extend the bounds to cover *complete algorithms*, which can be developed using all available data, including the validation data. To get from validation of a holdout algorithm to validation of a complete algorithm, we combine a holdout algorithm bound with a bound on the rate of disagreement between the holdout and complete algorithms.

3.1 Holdout Batch Algorithms

In this subsection, we derive bounds on precision and recall for holdout batch algorithms. The validation data is a sample of actual matches and the actual matches for a sample from $\mathcal{X}$. Let $\mathcal{S}_{\mathcal{M}}$ be a sample of verified matches, selected uniformly at random without replacement from $\mathcal{M}$. Let $\mathcal{S}_{\mathcal{X}}$ be a sample drawn uniformly at random without replacement from $\mathcal{X}$, and for which we know the actual matches. Let $\mathcal{M}$ be the set of actual matches and $\hat{\mathcal{M}}_H$ be the set of x-y pairs the holdout batch algorithm identifies as matches.

The precision and recall of the holdout classifier, which we denote as P_H and R_H respectively, are defined as follows:

$$P_H = \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}|}{|\hat{\mathcal{M}}_H|} = Pr\left\{(X,Y) \in \mathcal{M} | (X,Y) \in \hat{\mathcal{M}}_H\right\}, \tag{15}$$

$$R_H = \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}|}{|\mathcal{M}|} = Pr\left\{(X,Y) \in \hat{\mathcal{M}}_H | (X,Y) \in \mathcal{M}\right\}, \tag{16}$$

where (X, Y) is a random pair following a uniform distribution on $\mathcal{X} \times \mathcal{Y}$. We will start with recall, then use Bayes' rule to compute a bound on precision using a bound on recall.

Lemma 1 (Holdout Batch Recall)**.** *For any bound failure probability $\delta_r > 0$, with probability at least $1 - \delta_r$:*

$$R_H \geq p^-(\mathcal{M}, \mathcal{S}_\mathcal{M}, \mathbf{1}_{\hat{\mathcal{M}}_H}, 0, 1, \delta_r), \tag{17}$$

Proof.

$$R_H = \mathbb{E}_{(X,Y)\sim\mathcal{U}(\mathcal{M})}\left\{\mathbf{1}_{\hat{\mathcal{M}}_H}(X, Y)\right\}. \tag{18}$$

So

$$\mathbb{E}_{(X,Y)\sim\mathcal{U}(\mathcal{S}_\mathcal{M})}\left\{\mathbf{1}_{\hat{\mathcal{M}}_H}(X, Y)\right\} \tag{19}$$

is an unbiased estimate of R_H, based on samples drawn uniformly at random without replacement. □

Theorem 1 (Holdout Batch Precision)**.** *For any bound failure probabilities δ_r and δ_p, with probability at least $1 - \delta_r - \delta_p$:*

$$P_H \geq \frac{|\mathcal{X}|}{|\hat{\mathcal{M}}|} p^-(\mathcal{M}, \mathcal{S}_\mathcal{M}, \mathbf{1}_{\hat{\mathcal{M}}_H}, 0, 1, \delta_r) p^-(\mathcal{X}, \mathcal{S}_\mathcal{X}, m, 0, k_y, \delta_p), \tag{20}$$

where:

- *$m(x)$ is the number of matches in $\mathcal{M}$ that include x.*
- *k_y is an upper bound on the number of $y \in \mathcal{Y}$ that match any one $x \in \mathcal{X}$.*

Proof. Let (X, Y) be an uniform random pair in $\mathcal{X} \times \mathcal{Y}$. By Bayes' rule:

$$\begin{aligned} P_H &= Pr\left\{(X, Y) \in \mathcal{M} | (X, Y) \in \hat{\mathcal{M}}_H\right\} && (21) \\ &= \frac{Pr\left\{(X, Y) \in \hat{\mathcal{M}}_H | (X, Y) \in \mathcal{M}\right\} Pr\left\{(X, Y) \in \mathcal{M}\right\}}{Pr\left\{(X, Y) \in \hat{\mathcal{M}}_H\right\}}. && (22) \end{aligned}$$

We can use Lemma 1 to bound $R_H = Pr\left\{(X, Y) \in \hat{\mathcal{M}}_H | (X, Y) \in \mathcal{M}\right\}$. For $Pr\left\{(X, Y) \in \mathcal{M}\right\}$, note that

$$\mathbb{E}_{X\sim\mathcal{U}(\mathcal{X})}\left\{m(X)\right\}|\mathcal{X}| = |\mathcal{M}|, \tag{23}$$

so

$$Pr\left\{(X, Y) \in \mathcal{M}\right\} = \frac{|\mathcal{M}|}{|\mathcal{X}||\mathcal{Y}|} = \frac{\mathbb{E}_{X\sim\mathcal{U}(\mathcal{X})}\left\{m(X)\right\}}{|\mathcal{Y}|}. \tag{24}$$

Since $\mathbb{E}_{X\sim\mathcal{U}(\mathcal{S}_\mathcal{X})}\left\{m(X)\right\}$ is an unbiased estimate of $\mathbb{E}_{X\sim\mathcal{U}(\mathcal{X})}\left\{m(X)\right\}$, based on samples drawn uniformly at random without replacement, for $\delta_p > 0$, with probability at least $1 - \delta_p$,

$$Pr\left\{(X, Y) \in \mathcal{M}\right\} \geq \frac{p^-(\mathcal{X}, \mathcal{S}_\mathcal{X}, m, 0, k_y, \delta_p)}{|\mathcal{Y}|}. \tag{25}$$

For $Pr\left\{(X, Y) \in \hat{\mathcal{M}}_H\right\}$, we have $\hat{\mathcal{M}}_H$ since we are using a batch algorithm, so we can compute exactly:

$$Pr\left\{(X, Y) \in \hat{\mathcal{M}}_H\right\} = \frac{|\hat{\mathcal{M}}_H|}{|\mathcal{X}||\mathcal{Y}|}. \tag{26}$$

Substitute Lemma 1, the RHS of (25), and the RHS of (26) into Bayes' rule in (22) to complete the proof. □

3.2 Batch Algorithms That Use All Available Data

Let $\hat{\mathcal{M}}$ be the set of identified matches returned by the complete classfier trained on $\mathcal{S}_{\mathcal{M}}$, $\mathcal{S}_{\mathcal{X}}$ and the training samples used for the holdout classifier. Let P and R denote the precision and recall of the complete classifier. Since $\hat{\mathcal{M}}$ depends on the holdout samples $\mathcal{S}_{\mathcal{M}}$ and $\mathcal{S}_{\mathcal{X}}$, we cannot directly apply the above bounds. Instead, we use the holdout samples to bound precision and recall of the holdout classifier, and then we bound the difference in precision and recall between the holdout classifier and the complete classifier.

Theorem 2 (Complete Batch Recall)**.** *For any bound failure probabilities δ_1 and δ_2, with probability at least $1 - \delta_1 - \delta_2$:*

$$R \geq p^{-}\left(\mathcal{M}, \mathcal{S}_{\mathcal{M}}, \mathbf{1}_{\hat{\mathcal{M}}_H}, 0, 1, \delta_1\right) - \frac{|\hat{\mathcal{M}}_H \backslash \hat{\mathcal{M}}|}{|\mathcal{X}|\, p^{-}\left(\mathcal{X}, \mathcal{S}_{\mathcal{X}}, m, 0, k_y, \delta_2\right)}, \tag{27}$$

where $m(x)$ and k_y are defined as in Theorem 1.

Proof. By the definition of recall,

$$\begin{aligned} R &= \frac{|\hat{\mathcal{M}} \cap \mathcal{M}|}{|\mathcal{M}|} && (28)\\ &\geq \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}| - |(\hat{\mathcal{M}}_H \backslash \hat{\mathcal{M}}) \cap \mathcal{M}|}{|\mathcal{M}|} && (29)\\ &\geq \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}|}{|\mathcal{M}|} - \frac{|\hat{\mathcal{M}}_H \backslash \hat{\mathcal{M}}|}{|\mathcal{M}|}. && (30) \end{aligned}$$

Inequality (29) holds because an actual match identified by the holdout classifier must either be identified by the complete classifier as well or result in a disagreement between these two classifiers.

According to Lemma 1, with probability at least $1 - \delta_1$,

$$R_H = \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}|}{|\mathcal{M}|} \geq p^{-}\left(\mathcal{M}, \mathcal{S}_{\mathcal{M}}, \mathbf{1}_{\hat{\mathcal{M}}_H}, 0, 1, \delta_1\right). \tag{31}$$

Note that $\mathbb{E}_{X \sim \mathcal{U}(\mathcal{X})}\{m(X)\}\,|\mathcal{X}| = |\mathcal{M}|$ and $\mathcal{S}_{\mathcal{X}}$ is a sample drawn uniformly at random without replacement from $\mathcal{X}$. Therefore, for $\delta_2 > 0$, with probability at least $1 - \delta_2$,

$$|\mathcal{M}| \geq |\mathcal{X}|\, p^{-}\left(\mathcal{X}, \mathcal{S}_{\mathcal{X}}, m, 0, k_y, \delta_2\right). \tag{32}$$

We get the desired inequality by substituting (31) and (32) into (30) and using the union bound. □

Theorem 3 (Complete Batch Precision)**.** *For any bound failure probabilities δ_1 and δ_2, with probability at least $1 - \delta_1 - \delta_2$:*

$$\begin{aligned} P \geq\ & \frac{|\mathcal{X}|}{|\hat{\mathcal{M}}|} p^{-}\left(\mathcal{M}, \mathcal{S}_{\mathcal{M}}, \mathbf{1}_{\hat{\mathcal{M}}_H}, 0, 1, \delta_1\right) p^{-}\left(\mathcal{X}, \mathcal{S}_{\mathcal{X}}, m, 0, k_y, \delta_2\right)\\ & - \frac{|\hat{\mathcal{M}}_H \backslash \hat{\mathcal{M}}|}{|\hat{\mathcal{M}}|}, \end{aligned} \tag{33}$$

where m and k_y are as defined in Theorem 1.

Proof. By the definition of precision,

$$\begin{aligned} P &= \frac{|\hat{\mathcal{M}} \cap \mathcal{M}|}{|\hat{\mathcal{M}}|} && (34)\\ &\geq \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}| - |(\hat{\mathcal{M}}_H \backslash \hat{\mathcal{M}}) \cap \mathcal{M}|}{|\hat{\mathcal{M}}|} && (35)\\ &\geq \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}|}{|\hat{\mathcal{M}}_H|} \cdot \frac{|\hat{\mathcal{M}}_H|}{|\hat{\mathcal{M}}|} - \frac{|\hat{\mathcal{M}}_H \backslash \hat{\mathcal{M}}|}{|\hat{\mathcal{M}}|}. && (36) \end{aligned}$$

According to Theorem 1, with probability at least $1 - \delta_1 - \delta_2$,

$$P_H \tag{37}$$

$$= \frac{|\hat{\mathcal{M}}_H \cap \mathcal{M}|}{|\hat{\mathcal{M}}_H|} \tag{38}$$

$$\geq \frac{|\mathcal{X}|}{|\hat{\mathcal{M}}_H|} p^- \left(\mathcal{M}, \mathcal{S}_\mathcal{M}, \mathbf{1}_{\hat{\mathcal{M}}_H}, 0, 1, \delta_1\right) p^- \left(\mathcal{X}, \mathcal{S}_\mathcal{X}, m, 0, k_y, \delta_2\right). \tag{39}$$

Substituting (39) into (36) completes the proof. □

4 Validation for Query Algorithms

In this section we present PAC bounds on precision and recall for query algorithms, which compute identified matches for any $x \in \mathcal{X}$ upon request, but for which it may be infeasible to compute identified matches for all $x \in \mathcal{X}$ prior to validation. As validation data, the bounds use a sample from $\mathcal{X}$ for which all actual matches are known and identified matches are computed. As for validation of batch algorithms, we begin with bounds for holdout algorithms. Then we derive bounds for complete algorithms by combining bounds on holdout algorithms with bounds on rates of disagreement between holdout and complete algorithms.

4.1 Holdout Query Algorithm

For $x \in \mathcal{X}$, let $\hat{\mathcal{M}}_H(x)$ be the set of x-y pairs identified as matches by the holdout classifier that include x. Let $\mathcal{M}(x)$ be the set of actual matches that include x. Let (X, Y) be a pair selected uniformly at random from $\mathcal{X} \times \mathcal{Y}$.

Define single-node precision as

$$p_H(x) = \frac{\hat{\mathcal{M}}_H(x) \cap \mathcal{M}(x)}{\hat{\mathcal{M}}_H(x)} = Pr\left\{(X, Y) \in \mathcal{M}(x) | (X, Y) \in \hat{\mathcal{M}}_H(x)\right\}. \tag{40}$$

If the algorithm identifies at most one match for each x and there is at most one actual match, then $p_H(x)$ is one if an identified match is an actual match and zero otherwise. In general, $p_H(x)$ is the fraction of identified matches for x that are actual matches. If the algorithm identifies no matches for x, then $p_H(x)$ is undefined. So let

$$\hat{\mathcal{X}}_H = \left\{x \in \mathcal{X} | \hat{\mathcal{M}}_H(x) \neq \emptyset\right\}, \tag{41}$$

and define query precision as

$$P_H = \mathbb{E}_{X \sim \mathcal{U}(\hat{\mathcal{X}}_H)} \left\{p_H(X)\right\}. \tag{42}$$

Similarly, define single-node recall as

$$r_H(x) = \frac{\hat{\mathcal{M}}_H(x) \cap \mathcal{M}(x)}{\mathcal{M}(x)} = Pr\left\{(X, Y) \in \hat{\mathcal{M}}_H(x) | (X, Y) \in \mathcal{M}(x)\right\}. \tag{43}$$

If there are no actual matches for x, then $r_H(x)$ is undefined. So let

$$\mathcal{X}' = \left\{x \in \mathcal{X} | \mathcal{M}(x) \neq \emptyset\right\}, \tag{44}$$

and define query recall to be

$$R_H = \mathbb{E}_{X \sim \mathcal{U}(\mathcal{X}')} \left\{r_H(X)\right\}. \tag{45}$$

Let $\mathcal{S}_\mathcal{X}$ be a sample drawn from $\mathcal{X}$, uniformly at random without replacement. Suppose we know all actual matches for each $x \in \mathcal{S}_\mathcal{X}$ and we compute their identified matches. Then we can compute $p_H(x)$ for $x \in \mathcal{S}_\mathcal{X} \cap \hat{\mathcal{X}}_H$ and $r_H(x)$ for $s \in \mathcal{S}_\mathcal{X} \cap \mathcal{X}'$. We can use empirical means over $\mathcal{S}_\mathcal{X} \cap \hat{\mathcal{X}}$ and $\mathcal{S}_\mathcal{X} \cap \mathcal{X}'$ to estimate query precision and recall, respectively.

Theorem 4 (Holdout Query Precision and Recall)**.** *For any bound failure probability $\delta > 0$, with probability at least $1 - \delta$:*

$$P_H \geq p^-(\hat{\mathcal{X}}_H, \mathcal{S}_\mathcal{X} \cap \hat{\mathcal{X}}_H, p_H, 0, 1, \delta). \tag{46}$$

Also, with probability $1 - \delta$:

$$R_H \geq p^-(\mathcal{X}', \mathcal{S}_\mathcal{X} \cap \mathcal{X}', r_H, 0, 1, \delta). \tag{47}$$

Proof. Since the elements of $\mathcal{S}_\mathcal{X}$ are selected uniformly at random without replacement from $\mathcal{X}$, $\mathcal{S}_\mathcal{X} \cap \hat{\mathcal{X}}_H$ has the same distribution as selecting a size-$|\mathcal{S}_\mathcal{X} \cap \hat{\mathcal{X}}_H|$ sample from $\hat{\mathcal{X}}_H$ uniformly at random without replacement. Similarly, $\mathcal{S}_\mathcal{X} \cap \mathcal{X}'$ has the same distribution as selecting a size-$|\mathcal{S}_\mathcal{X} \cap \mathcal{X}'|$ sample from $\mathcal{X}'$ uniformly at random without replacement. □

Theorem 4 is valid if we continue to draw samples from $\mathcal{S}_\mathcal{X}$ until $\mathcal{S}_\mathcal{X} \cap \hat{\mathcal{X}}_H$ or $\mathcal{S}_\mathcal{X} \cap \mathcal{X}'$ reaches a desired sample size. If there is a method to sample directly from $\hat{\mathcal{X}}_H$ or $\mathcal{X}'$, then those samples also produce valid bounds.

4.2 Query Algorithms That Use All Available Data

Let $\hat{\mathcal{M}}$ be the set of identified matches returned by the complete classfier, which can be developed using all available data, including the validation data. Let $\mathcal{S}'_\mathcal{X}$ be a sample independent of $\mathcal{S}_\mathcal{X}$ and drawn uniformly at random without replacement from $\mathcal{X}$. Assume that $\hat{\mathcal{M}}$ and $\hat{\mathcal{M}}_H$ are unknown, but $\hat{\mathcal{M}}(x)$ and $\hat{\mathcal{M}}_H(x)$ are computed for $x \in \mathcal{S}_\mathcal{X}$ and $x \in \mathcal{S}'_\mathcal{X}$. Define $p(x)$, $\hat{\mathcal{X}}$, P, $r(x)$, and R as complete versions of the holdout-oriented definitions in Subsection 4.1:

$$p(x) = \frac{\hat{\mathcal{M}}(x) \cap \mathcal{M}(x)}{\hat{\mathcal{M}}(x)}, \tag{48}$$

$$\hat{\mathcal{X}} = \left\{ x \in \mathcal{X} | \hat{\mathcal{M}}(x) \neq \emptyset \right\}, \tag{49}$$

$$P = \mathbb{E}_{X \sim \mathcal{U}(\hat{\mathcal{X}})} \left\{ p(X) \right\}, \tag{50}$$

$$r(x) = \frac{\hat{\mathcal{M}}(x) \cap \mathcal{M}(x)}{\mathcal{M}(x)}, \tag{51}$$

$$R = \mathbb{E}_{X \sim \mathcal{U}(\mathcal{X}')} \left\{ r(X) \right\}. \tag{52}$$

There are two remarks that we need to address before presenting the results:

1. An extra sample $\mathcal{S}'_\mathcal{X}$ is used to validate rates of disagreement between the holdout classifier and complete classifiers. For this sample we do not require knowledge of actual matches. We only need to know the identified matches returned by the holdout classifier and the complete classifier. Therefore, collecting data for $\mathcal{S}'_\mathcal{X}$ can be cheaper than collecting validation data for $\mathcal{S}_\mathcal{X}$. If so, then we can make the size of sample $\mathcal{S}'_\mathcal{X}$ very large to produce a tight bound on the rate of disagreement between the holdout and complete classifiers.
2. For simplicity of notation, we use same bound failure probability δ for each term in our results. But, in practice, one can use different bound failure probabilities for different terms.

Theorem 5 (Complete Query Recall)**.** *For any bound failure probability $\delta > 0$, with probability at least $1 - 3\delta$,*

$$R \geq p^- \left(\mathcal{X}', \mathcal{S}_\mathcal{X} \cap \mathcal{X}', r_H, 0, 1, \delta \right) - \frac{p^+ \left(\mathcal{X}, \mathcal{S}'_\mathcal{X}, d_r, 0, 1, \delta \right)}{p^- \left(\mathcal{X}, \mathcal{S}_\mathcal{X}, \mathbf{1}_{\mathcal{X}'}, 0, 1, \delta \right)}, \tag{53}$$

where $d_r(x) = \mathbf{1}_{\{\hat{\mathcal{M}}_H(x) \setminus \hat{\mathcal{M}}(x) \neq \emptyset\}}$.

Proof. Since $\mathcal{S}_\mathcal{X}$ is a sample selected uniformly at random without replacement from $\mathcal{X}$, by Theorem 4, the recall of the holdout classifier satisfies

$$Pr \left\{ R_H \geq p^- \left(\mathcal{X}', \mathcal{S}_\mathcal{X} \cap \mathcal{X}', r_H, 0, 1, \delta \right) \right\} \geq 1 - \delta. \tag{54}$$

The difference between the recalls of the complete classifier and the holdout classifier is

$$
\begin{aligned}
& R - R_H && (55)\\
= \; & \frac{1}{|\mathcal{X}'|} \sum_{x \in \mathcal{X}'} \frac{|\hat{\mathcal{M}}(x) \cap \mathcal{M}(x)| - |\hat{\mathcal{M}}_H(x) \cap \mathcal{M}(x)|}{|\mathcal{M}(x)|} && (56)\\
\geq \; & -\frac{1}{|\mathcal{X}'|} \sum_{x \in \mathcal{X}'} \frac{|(\hat{\mathcal{M}}_H(x) \backslash \hat{\mathcal{M}}(x)) \cap \mathcal{M}(x)|}{|\mathcal{M}(x)|} && (57)\\
\geq \; & -\frac{1}{|\mathcal{X}'|} \sum_{x \in \mathcal{X}'} \mathbf{1}_{\{\hat{\mathcal{M}}_H(x) \backslash \hat{\mathcal{M}}(x) \neq \emptyset\}} && (58)\\
\geq \; & -\frac{|\mathcal{X}|}{|\mathcal{X}'|} \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} d_r(x). && (59)
\end{aligned}
$$

Since $\mathcal{S}'_{\mathcal{X}}$ is independent of $\mathcal{S}_{\mathcal{X}}$, $\mathcal{S}'_{\mathcal{X}}$ is still a sample drawn uniformly at random without replacement from $\mathcal{X}$ conditional on $\mathcal{S}_{\mathcal{X}}$. Note that $\hat{\mathcal{M}}(x)$ is a fixed function conditional on $\mathcal{S}_{\mathcal{X}}$ since we assume that the training samples of the holdout classifier are fixed, therefore we can use $\mathcal{S}'_{\mathcal{X}}$ to validate $\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} d_r(x)$ conditional on $\mathcal{S}_{\mathcal{X}}$:

$$
Pr\left\{ \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} d_r(x) \leq p^+ \left(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, d_r, 0, 1, \delta\right) | \mathcal{S}_{\mathcal{X}} \right\} > 1 - \delta. \tag{60}
$$

We derive the unconditional bound by integrating out $\mathcal{S}_{\mathcal{X}}$:

$$
Pr\left\{ \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} d_r(x) \leq p^+ \left(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, d_r, 0, 1, \delta\right) \right\} > 1 - \delta. \tag{61}
$$

Since $\frac{|\mathcal{X}'|}{|\mathcal{X}|} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \mathbf{1}_{\mathcal{X}'}(x)$, by section 2

$$
Pr\left\{ \frac{|\mathcal{X}'|}{|\mathcal{X}|} \geq p^- \left(\mathcal{X}, \mathcal{S}_{\mathcal{X}}, \mathbf{1}_{\mathcal{X}'}, 0, 1, \delta\right) \right\} > 1 - \delta. \tag{62}
$$

Combining (54), (61) and (62) by the union bound we get the result stated in the theorem. □

Theorem 6 (Complete Query Precision)**.** *For any bound failure probability $\delta > 0$, with probability at least $1 - 4\delta$,*

$$
\begin{aligned}
P \; \geq \; & \left[p^-(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, \mathbf{1}_{\hat{\mathcal{X}}_H}, 0, 1, \delta) p^-(\hat{\mathcal{X}}_H, \mathcal{S}_{\mathcal{X}} \cap \hat{\mathcal{X}}_H, p_H, 0, 1, \delta) \right. \\
& \left. - p^+ \left(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, d_p, -1, 2, \delta\right) \right] / p^+(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, \mathbf{1}_{\hat{\mathcal{X}}}, 0, 1, \delta), \qquad (63)
\end{aligned}
$$

where

$$
d_p(x) = \mathbf{1}_{\hat{\mathcal{X}} \cap \hat{\mathcal{X}}_H}(x) \mathbf{1}_{\{\mathcal{M}(x) \neq \mathcal{M}_H(x)\}} \left(1 + \frac{|\hat{\mathcal{M}}_H(x) \backslash \hat{\mathcal{M}}(x)|}{|\hat{\mathcal{M}}(x)|} \right) + \mathbf{1}_{\hat{\mathcal{X}}_H \backslash \hat{\mathcal{X}}}(x). \tag{64}
$$

Proof. Note that

$$
\begin{aligned}
P \; = \; & \frac{1}{|\hat{\mathcal{X}}|} \sum_{x \in \hat{\mathcal{X}}} \frac{|\hat{\mathcal{M}}(x) \cap \mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|} && (65)\\
= \; & \frac{1}{|\hat{\mathcal{X}}|} \left(\sum_{x \in \hat{\mathcal{X}}} \frac{|\hat{\mathcal{M}}(x) \cap \mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|} - \sum_{x \in \hat{\mathcal{X}}_H} \frac{|\hat{\mathcal{M}}_H(x) \cap \mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|} \right) \\
& + \frac{|\hat{\mathcal{X}}_H|}{|\hat{\mathcal{X}}|} \frac{1}{|\hat{\mathcal{X}}_H|} \sum_{x \in \hat{\mathcal{X}}_H} \frac{|\hat{\mathcal{M}}_H(x) \cap \mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|}. && (66)
\end{aligned}
$$

The first term can be decomposed into three parts

$$\sum_{x\in\hat{\mathcal{X}}}\frac{|\hat{\mathcal{M}}(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|}-\sum_{x\in\hat{\mathcal{X}}_H}\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|} \tag{67}$$

$$= \sum_{x\in\hat{\mathcal{X}}\cap\hat{\mathcal{X}}_H}\left(\frac{|\hat{\mathcal{M}}(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|}-\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|}\right) + \sum_{x\in\hat{\mathcal{X}}\setminus\hat{\mathcal{X}}_H}\frac{|\hat{\mathcal{M}}(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|}-\sum_{x\in\hat{\mathcal{X}}_H\setminus\hat{\mathcal{X}}}\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|} \tag{68}$$

$$= \mathrm{I}+\mathrm{II}-\mathrm{III}. \tag{69}$$

Since $|\hat{\mathcal{X}}\setminus\hat{\mathcal{X}}_H|$ and $|\hat{\mathcal{X}}_H\setminus\hat{\mathcal{X}}|$ are usually small, we can simply use 0 and $-|\hat{\mathcal{X}}_H\setminus\hat{\mathcal{X}}|$ as the lower bound for II and III. For I, we have

$$\mathrm{I} \geq \sum_{\substack{x\in\hat{\mathcal{X}}\cap\hat{\mathcal{X}}_H\\ \hat{\mathcal{M}}(x)\neq\hat{\mathcal{M}}_H(x)}}\left(\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|}-\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|} -\frac{|(\hat{\mathcal{M}}_H(x)\setminus\hat{\mathcal{M}}(x))\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}(x)|}\right) \tag{70}$$

$$\geq \sum_{\substack{x\in\hat{\mathcal{X}}\cap\hat{\mathcal{X}}_H\\ \hat{\mathcal{M}}(x)\neq\hat{\mathcal{M}}_H(x)}}\left(-1-\frac{|\hat{\mathcal{M}}_H(x)\setminus\hat{\mathcal{M}}(x)|}{|\hat{\mathcal{M}}(x)|}\right). \tag{71}$$

Therefore,

$$P$$

$$\geq \frac{1}{|\hat{\mathcal{X}}|}\left(-\sum_{\substack{x\in\hat{\mathcal{X}}\cap\hat{\mathcal{X}}_H\\ \hat{\mathcal{M}}(x)\neq\hat{\mathcal{M}}_H(x)}}\left(1+\frac{|\hat{\mathcal{M}}_H(x)\setminus\hat{\mathcal{M}}(x)|}{|\hat{\mathcal{M}}(x)|}\right)-|\hat{\mathcal{X}}_H\setminus\hat{\mathcal{X}}|\right) +\frac{|\hat{\mathcal{X}}_H|}{|\hat{\mathcal{X}}|}\frac{1}{|\hat{\mathcal{X}}_H|}\sum_{x\in\hat{\mathcal{X}}_H}\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|} \tag{72}$$

$$= \frac{|\mathcal{X}|}{|\hat{\mathcal{X}}|}\Bigg\{-\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\left(\mathbf{1}_{\hat{\mathcal{X}}\cap\hat{\mathcal{X}}_H}(x)\mathbf{1}_{\{\hat{\mathcal{M}}(x)\neq\hat{\mathcal{M}}_H(x)\}}\left(1+\frac{|\hat{\mathcal{M}}_H(x)\setminus\hat{\mathcal{M}}(x)|}{|\hat{\mathcal{M}}(x)|}\right) +\mathbf{1}_{\hat{\mathcal{X}}_H\setminus\hat{\mathcal{X}}}(x)\right)+\frac{|\hat{\mathcal{X}}_H|}{|\mathcal{X}|}\frac{1}{|\hat{\mathcal{X}}_H|}\sum_{x\in\hat{\mathcal{X}}_H}\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|}\Bigg\} \tag{73}$$

Note that

$$P_H=\frac{1}{|\hat{\mathcal{X}}_H|}\sum_{x\in\hat{\mathcal{X}}_H}\frac{|\hat{\mathcal{M}}_H(x)\cap\mathcal{M}(x)|}{|\hat{\mathcal{M}}_H(x)|}. \tag{74}$$

By Theorem 4, we have a lower bound for the precision of the holdout classifier with probability at least 1-δ,

$$P_H\geq p^-\left(\hat{\mathcal{X}}_H,\mathcal{S}_{\mathcal{X}}\cap\hat{\mathcal{X}}_H,p_H,0,1,\delta\right). \tag{75}$$

Since $\mathcal{S}'_{\mathcal{X}}$ is independent of $\mathcal{S}_{\mathcal{X}}$, $\mathcal{S}'_{\mathcal{X}}$ is still a sample selected uniformly at random without replacement from $\mathcal{X}$ conditional on $\mathcal{S}_{\mathcal{X}}$. Note that $\hat{\mathcal{M}}(x)$ is a fixed function conditional on $\mathcal{S}_{\mathcal{X}}$, therefore we can use $\mathcal{S}'_{\mathcal{X}}$ to validate $\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}d_p(x)$ conditional on $\mathcal{S}_{\mathcal{X}}$, where $-1\leq d_p(x)\leq 2$:

$$Pr\left\{\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}d_p(x)\leq p^+\left(\mathcal{X},\mathcal{S}'_{\mathcal{X}},d_p,-1,2,\delta\right)|\mathcal{S}_{\mathcal{X}}\right\}>1-\delta. \tag{76}$$

We derive the unconditional bound by integrating out $\mathcal{S}_{\mathcal{X}}$:

$$Pr\left\{\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} d_p(x) \leq p^+\left(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, d_p, -1, 2, \delta\right)\right\} > 1-\delta. \tag{77}$$

Since $\frac{|\hat{\mathcal{X}}|}{|\mathcal{X}|} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \mathbf{1}_{\hat{\mathcal{X}}}(x)$ and $\frac{|\hat{\mathcal{X}}_H|}{|\mathcal{X}|} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \mathbf{1}_{\hat{\mathcal{X}}_H}(x)$, by section 2 we have PAC bounds

$$Pr\left\{\frac{|\hat{\mathcal{X}}|}{|\mathcal{X}|} \leq p^+\left(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, \mathbf{1}_{\hat{\mathcal{X}}}, 0, 1, \delta\right)\right\} \quad > \quad 1-\delta, \tag{78}$$

$$Pr\left\{\frac{|\hat{\mathcal{X}}_H|}{|\mathcal{X}|} \geq p^-\left(\mathcal{X}, \mathcal{S}'_{\mathcal{X}}, \mathbf{1}_{\hat{\mathcal{X}}_H}, 0, 1, \delta\right)\right\} \quad > \quad 1-\delta. \tag{79}$$

Combining (75), (77), (78) and (79) by the union bound we get the result stated in the theorem. □

5 Extensions

This section outlines a few ways to extend the results in this paper. One extension provides different confidence levels for matches identified by different algorithms or identified by a single algorithm that provides match scores as well as matches. Other extensions bound matching error rate, instead of precision or recall, and validate matching between data fields, instead of nodes in different networks. We also explain how to partition a sample of nodes with verified actual matches or a sample of actual matches into a training and validation set in such a way that the validation methods developed in this paper apply.

5.1 Confidence Levels for Different Types of Matches

The methods described in previous sections validate a single matching algorithm. In some applications, multiple network reconciliation algorithms are applied, producing multiple sets of identified matches. Similarly, some network reconciliation algorithms return a score for each match. The score is designed to indicate a level of similarity between the nodes in a matched pair. In this case, the range of scores may be partitioned, and scores in different parts of the range can be placed in different sets of identified matches.

Use union bounds to produce simultaneous PAC bounds for precision or recall over the different sets of identified matches. For example, if there are two sets of identified matches, and we validate the recall of one to be at least 80% and the other to be at least 90%, each with bound failure probability at most 2.5%, then those recall rates both hold with probability at least 100% - $2(2.5\%)$ = 95%. Simultaneous bounds support different levels of confidence for different sets of identified matches, which may affect how we choose to handle the matches in applications. For example, in an application to merge contact information based on matching user profiles of contacts across social networks, if we have only 80% precision for a set of identified matches, then we may choose to ask the user to verify matches from that set. If we have 99% precision from a different set, we may choose to merge those matches automatically, without asking the user for confirmation.

5.2 Validation of Error Rates

The validation methods for precision and recall can be easily extended to other measures of accuracy for network reconciliation methods. For example, for query algorithms, we can define single-node error for node $x \in \mathcal{X}$ as

$$w(x) = I(\mathcal{M}(x) \neq \hat{\mathcal{M}}(x)), \tag{80}$$

meaning that the algorithm has an error on a node if it either fails to identify all matches for the node or identifies any false matches for it. The error rate for the algorithm can be defined as the average of single-node errors over all nodes in X. As for precision and recall rates, we can validate the error rate by using a sample of nodes with verified actual matches to bound the error rate for a holdout algorithm and using a sample of nodes to validate the rate of disagreement between the holdout algorithm and the algorithm developed on all available data.

5.3 Matching Data Fields

In some applications, we are given a set of data fields – for example names, phone numbers, email addresses, and locations – and some information about links between them – for example that they are in the same entry of a contact list or that they occur together in a document. The goal is to match pairs of data fields that refer to the same entity or to aggregate data fields by entity. We can extend our validation methods to algorithms for field matching or aggregation, as follows.

An application that connects an email address to a phone number in case someone wants to reply to an email by phone is matching pairs of fields. For matching pairs of fields, we can set $\mathcal{X} = \mathcal{Y}$, so that we are matching nodes (fields, in our case) from a set to other nodes in the same set. Do not include pairs consisting of the same node twice in the sets of actual matches $\mathcal{M}$ or identified matches $\hat{\mathcal{M}}$, since we want to validate the matching of different fields to each other.

An application that extracts fields from free text and structured data then aggregates those fields for each of a set of people or organizations performs aggregation. For these matching algorithms, we can use $\mathcal{X}$ as the set of entities and $\mathcal{Y}$ as the set of fields. In general, each $x \in \mathcal{X}$ will match to multiple $y \in \mathcal{Y}$. In many applications, it is also possible for multiple $x \in \mathcal{X}$ to match a single $y \in \mathcal{Y}$, for example, multiple people may share a single home phone number and address.

5.4 Sampling Procedure for Algorithms That Use All Available Data

Our validation methods use samples $\mathcal{S}_{\mathcal{X}}$ and $\mathcal{S}_{\mathcal{M}}$ drawn uniformly at random without replacement from $\mathcal{X}$ and $\mathcal{M}$, respectively. In some applications, we have such samples, but we wish to use subsamples for validation so that we can use some of the samples to develop the holdout algorithm. If we simply partition a sample into a validation set to be withheld and a training set to develop the holdout classifier, then the holdout classifier is not developed independently of the selection of the validation subsample, as required for our bounds to be valid, because the training subsample specifically excludes the validation subsample. Instead, we must allow a chance for some intersection between subsamples so that they are distributed as if the training and validation samples were drawn separately, and each draw was uniformly at random without replacement from $\mathcal{X}$ or $\mathcal{M}$.

Consider the case of sampling from $\mathcal{X}$; the case of $\mathcal{M}$ is similar. Let $\mathcal{L} \subseteq \mathcal{X}$ be a sample of $\mathcal{X}$ drawn uniformly at random without replacement for which we have verified the actual matches. Select a validation subsample size s and a training subsample size t such that $s + t = |\mathcal{L}|$. Generate training subsample $\mathcal{T}$ and validation subsample $\mathcal{S}$ as follows:

1. Select t samples uniformly at random without replacement from $\mathcal{L}$. These samples are $\mathcal{T}$.
2. Generate intersection size i from a hypergeometric distribution with population size $|\mathcal{X}|$, number of success states in the population t and number of draws s. (So $Pr\{i\} = \frac{\binom{t}{i}\binom{|\mathcal{X}|-t}{s-i}}{\binom{|\mathcal{X}|}{s}}$.)
3. Select i samples uniformly at random without replacement from $\mathcal{T}$.
4. Select $s - i$ samples uniformly at random without replacement from $\mathcal{L} \backslash \mathcal{T}$.
5. The validation subsample $\mathcal{S}$ consists of the samples from Steps 3 and 4.

Let $\mathcal{T}'$ and $\mathcal{S}'$ denote two independent samples of size t and s respectively, drawn uniformly at random without replacement from the population $\mathcal{X}$. Then the subsamples $\mathcal{T}$ and $\mathcal{S}$ generated by the above procedure have the same distribution as $\mathcal{T}'$ and $\mathcal{S}'$. This is because the size of the intersection between $\mathcal{T}$ and $\mathcal{S}$ has the same distribution as that between $\mathcal{T}'$ and $\mathcal{S}'$, and the samples drawn in Steps 1, 3 and 4 have the same distribution as those drawn uniformly at random without replacement from the population. See the Appendix for a rigorous proof.

6 Discussion

This paper introduces methods to validate precision and recall rates for network reconciliation algorithms. The methods apply regardless of the process that generated the networks' link structure and node data. In fact, the methods apply to matching algorithms in general, not just those that

match nodes between networks. The methods include validation of matching algorithms that use all available data, so data used for validation may be used for matching as well. We also extended the methods to provide confidence bounds for matches by different algorithms or matches based on different levels of evidence.

One direction for future research is to use the validation methods developed in this paper to guide network reconciliation processes. For example, some iterative matching algorithms begin with a seed set of matches and use matches from previous iterations to infer new matches in successive iterations [1, 13, 18, 21]. In each iteration, validated precision and recall rates may be useful to determine which pairs to adopt as new matches and how much to rely on them to infer future future matches. Pedarsani et al. [18] use Bayesian confidence measures in this way, under the assumption that the features used for matching have independent distributions. The validation methods developed in this paper could be used in a similar way when the feature distributions are correlated or unknown.

Another direction for future research is to perform validation in the absence of verified matches. Perhaps we could use the identified matches that are supported by the most evidence in place of verified match data. We would need to adjust the bounds to accommodate some errors in the validation data. A similar process might be used to bootstrap validation: start with a seed set of verified matches, use them to validate a set of matches identified by one algorithm, then use the verified and identified matches to validate the set of matches identified by another algorithm.

Finally, it would be interesting to develop validation methods for network reconciliation in non-transductive settings, where the goal is to assess precision and recall for nodes to be added to the network in the future, whose data and links are not yet known. For these settings, we may need to assume that we know how the network will grow, or at least have a probabilistic model for network growth. Alternatively, we may be able to develop validation methods based on the assumption that soon-to-be-added nodes will be generated by the same distribution as recently-added nodes, even if we do not know the distribution, as in work on classifier validation using cohorts [4].

Appendix

Proof of Subsampling Procedure from Section 5.4

Proof. Let $|\mathcal{X}| = n$, $|\mathcal{T}| = t$ and $|\mathcal{S}| = s$. Note that $|\mathcal{L}| = t + s$. It suffices to show that

$$Pr\{\mathcal{T}\} = \frac{1}{\binom{n}{t}}, \tag{81}$$

$$Pr\{\mathcal{S}|\mathcal{T}\} = \frac{1}{\binom{n}{s}}. \tag{82}$$

Since $\mathcal{L}$ is selected uniformly at random without replacement from $\mathcal{X}$, we have

$$Pr\{\mathcal{T}\} = \sum_{\mathcal{L}} Pr\{\mathcal{L}\} Pr\{\mathcal{T}|\mathcal{L}\} \tag{83}$$

$$= \sum_{\mathcal{L}\supseteq\mathcal{T}} \frac{1}{\binom{n}{t+s}\binom{t+s}{t}} \tag{84}$$

$$= \frac{\binom{n-t}{s}}{\binom{n}{t+s}\binom{t+s}{t}} \tag{85}$$

$$= \frac{1}{\binom{n}{t}}. \tag{86}$$

This shows that $\mathcal{T}$ has the same distribution as a size-t sample selected uniformly at random without replacement from $\mathcal{X}$.

The distribution of $\mathcal{S}$ conditional on $\mathcal{T}$ satisfies

$$Pr\{\mathcal{S}|\mathcal{T}\} = \sum_{\mathcal{L}} Pr\{\mathcal{L}|\mathcal{T}\}\, Pr\{\mathcal{S}|\mathcal{T},\mathcal{L}\} \tag{87}$$

$$= \sum_{\mathcal{L}\supseteq\mathcal{T}\cup\mathcal{S}} Pr\{\mathcal{L}|\mathcal{T}\}\, Pr\{i = |\mathcal{S}\cap\mathcal{T}|\}\, Pr\{\mathcal{S}|N,\mathcal{T},\mathcal{L}\} \tag{88}$$

$$= \frac{\binom{n-t-s+|\mathcal{S}\cap\mathcal{T}|}{|\mathcal{S}\cap\mathcal{T}|}\binom{t}{|\mathcal{S}\cap\mathcal{T}|}\binom{n-t}{s-|\mathcal{S}\cap\mathcal{T}|}}{\binom{n-t}{s}\binom{n}{s}\binom{t}{|\mathcal{S}\cap\mathcal{T}|}\binom{s}{s-|\mathcal{S}\cap\mathcal{T}|}} \tag{89}$$

$$= \frac{1}{\binom{n}{s}}. \tag{90}$$

Thus $\mathcal{S}$ is independent of $\mathcal{T}$ and has the same distribution as a size-s sample selected uniformly at random without replacement from $\mathcal{X}$. □

References

[1] A. N. andP. Calad and B. Martins. Resolving user identities over social networks through supervised learning and rich similarity features. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 728–729, New York, NY, USA, 2012. ACM.

[2] J.-Y. Audibert. *PAC-Bayesian Statistical Learning Theory*. PhD thesis, Laboratoire de Probabilities et Modeles Aleatoires, Universites Paris 6 and Paris 7, 2004.

[3] R. Bardenet and O.-A. Maillard. Concentration inequalities for sampling without replacement. 2013.

[4] E. Bax, J. Li, A. Sonmez, and Z. Cataltepe. Validating collective classification using cohorts. *NIPS Workshop: Frontiers of Network Analysis: Methods, Models, and Applications*, 2013.

[5] S. N. Bernstein. On certain modifications of Chebyshev's inequality. *Doklady Akademii Nauk SSSR*, 17(6):275–277, 1937.

[6] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities – A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[7] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285–287, 1979.

[8] P. Foggia, G. Percannella, and M. Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(01):1450001, 2014.

[9] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[10] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[11] P. G. Hoel. *Introduction to Mathematical Statistics*. Wiley, 1954.

[12] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, 1993.

[13] N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. *Proceedings of the 40th International Conference on Very large Databases (VLDB)*, pages 377–388, 2014.

[14] J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

[15] J. Li, A. Sonmez, Z. Cataltepe, and E. Bax. Validation of network classifiers. *Structural, Syntactic, and Statistical Pattern Recognition Lecture Notes in Computer Science*, 7626:448–457, 2012.

[16] C. Loader. Fast and accurate computation of binomial probabilities, 2000.

[17] A. Maurer and M. Pontil. Empirical bernstein bounds and sample-variance penalization. In *COLT*, 2009.

[18] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser. A bayesian method for matching two similar graphs without seeds. In *Fifty-first Annual Allerton Conference*, pages 1598–1607, 2013.

[19] Y. Tian and J. M. Patel. Tale: A tool for approximate large graph matching. In *IEEE 24th International Conference on Data Engineering, 2008. ICDE 2008*, pages 963–972, 2008.

[20] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[21] L. Yartseva and M. Grossglauser. On the performance of percolation graph matching. *Conference on Online Social Networks (COSN)*, pages 119–129, 2013.

[22] M. Zaslavskiy, F. Bach, and J.-P. Vert. Path following algorithm for the graph matching problem. *CoRR*, abs/0801.3654, 2008.