

Making Holistic Schema Matching Robust: An Ensemble Approach*

Bin He, Kevin Chen-Chuan Chang
Computer Science Department
University of Illinois at Urbana-Champaign
binhe@uiuc.edu, kcchang@cs.uiuc.edu

ABSTRACT

The Web has been rapidly “deepened” by myriad searchable databases online, where data are hidden behind query interfaces. As an essential task toward integrating these massive “deep Web” sources, *large scale schema matching* (*i.e.*, discovering semantic correspondences of attributes across many query interfaces) has been actively studied recently. In particular, many works have emerged to address this problem by “holistically” matching many schemas at the same time and thus pursuing “mining” approaches in nature. However, while *holistic schema matching* has built its promise upon the large quantity of input schemas, it also suffers the robustness problem caused by noisy data quality. Such noises often inevitably arise in the automatic extraction of schema data, which is mandatory in large scale integration. For holistic matching to be viable, it is thus essential to make it robust against noisy schemas. To tackle this challenge, we propose a *data-ensemble* framework with sampling and voting techniques, which is inspired by *bagging predictors*. Specifically, our approach creates an ensemble of matchers, by randomizing input schema data into many independently downsampled *trials*, executing the same matcher on each trial and then aggregating their ranked results by taking majority voting. As a principled basis, we provide analytic justification of the effectiveness of this data-ensemble framework. Further, empirically, our experiments on real Web data show that the “ensemblization” indeed significantly boosts the matching accuracy under noisy schema input, and thus maintains the desired robustness of a holistic matcher.

Categories and Subject Descriptors

H.2.5 [Database Management]: Heterogeneous Databases; H.2.8 [Database Management]: Database Applications—*Data Mining*

General Terms

Algorithms, Experimentation, Performance

*This material is based upon work partially supported by NSF Grants IIS-0133199 and IIS-0313260. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.
Copyright 2005 ACM 1-59593-135-X/05/0008 ...\$5.00.

Keywords

data integration, deep Web, schema matching, ensemble, bagging predictors

1. INTRODUCTION

With the prevalence of online Web databases, large scale integration has become a pressing problem. In particular, we have witnessed the rapid growth of databases on the Web, or the so-called “deep Web.” A July 2000 survey [4] estimated that 96,000 “search cites” and 550 billion content pages in this deep Web. Our recent study [7] in April 2004 estimated 450,000 online databases. With the virtually unlimited amount of information sources, the deep Web is clearly an important frontier for data integration.

To enable the integration of the deep Web, it is critical to discover matchings among attributes across large scale sources. On the deep Web, numerous online databases provide dynamic *query*-based data access through their *query interfaces*, instead of static URL links. Each query interface accepts queries over its *query schemas* (*e.g.*, author, title, subject, ... for *amazon.com*). Schema matching (*i.e.*, discovering semantic correspondences of attributes) among many query interfaces is essential for mediating queries across deep Web sources.

In particular, with the proliferation of sources in various domains, we often face the challenges of integrating and thus matching “alternative” sources in the same domain (*e.g.*, Books, Airfares). For instance, we may build a comparison shopping service for books or airfares, *e.g.*, purchasing a book with lowest price among book sources or a flight ticket with the best trade-off between price and number of connections among airline sources. To enable such integration scenarios, we need to discover either simple 1:1 matchings, *e.g.*, subject = category in Books, or complex *m:n* matchings, *e.g.*, passengers = {adults, seniors, children, infants} in Airfares.

While schema matching has been a central issue in data integration [3, 18], the large scale sets new requirements on the matching task. Traditional schema matching works (*e.g.*, [17, 8, 16, 13, 15]) are developed for small scale and static integration scenarios, in which automatic matching technique is often an option to reduce human labor, as an aid to manually configured semantics. Schema matching under such scenarios is abstracted as finding pairwise attribute correspondences between two sources and thus cannot scale well. In contrast, in large scale data integration scenarios, the matching process needs to be as automatic as possible and scalable to large quantities of sources, as the large scale mandates.

The challenge of large scale lends itself to a novel opportunity for automatic large scale schema matching—Many recent works [9, 11,

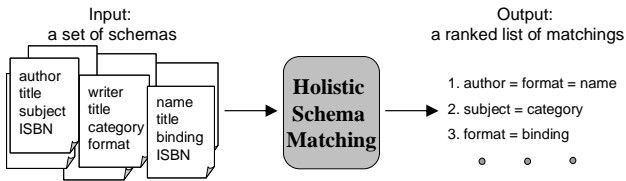


Figure 1: The holistic schema matching approach.

10, 19] have emerged to address schema matching “holistically” by matching many schemas at the same time and finding all matchings at once. Such *holistic schema matching* relies on large scale data quantity to discover semantic correspondences of attributes and thus essentially pursues a “data mining” approach in nature. In particular, statistical model discovery [9], clustering [11, 19] and correlation mining [10] approaches have been developed to “mine” matchings. Therefore, holistic schema matching can be abstracted as taking a set of schemas as input and outputting a ranked list of matchings, as Figure 1 shows.

However, while holistic schema matching leverages the opportunity of large data quantity, it also suffers the inherent problem of noisy data quality, which has not been extensively noticed and studied. Specifically, as the large scale mandates, to automate the matching process, the gathering of schemas (*i.e.*, extracting schemas from a set of query interfaces in HTML format) should also be automated— If the schemas are to be manually prepared, it would certainly defeat the purpose of automatic schema matching, especially for large scale integration. Although *automatic schema extraction* has been proposed recently [21, 12], as existing holistic schema matching works all adopt manually extracted schemas, the integration of these two “subsystems” remains unstudied. In particular, since errors are inevitable in automatic schema extraction, the input schemas of holistic schema matching are in fact noisy. As Section 2.1 will discuss, the noises made by schema extraction can compromise the matching performance to as much as 30%, rendering holistic matching almost unusable. Therefore, without essentially addressing the issue of data quality, holistic schema matching does not sustain itself as a viable technique.

We are thus facing a practical and challenging problem: *For holistic schema matching, how do we meet the opportunity of holistic quantity with the challenge of robust quality?* While large scale integration enables us to embrace the “blessing” of holistic quantity, — nothing comes for free— it also challenges us with the “curse” of non-robust quality, as data are inevitably noisy. Our goal is to maintain the robustness of a holistic matcher with the presence of noisy input schemas. In particular, as our design objective, we are searching for a solution that is: 1) *deployable*: We would like the solution to build upon an existing matching approach to maintain its robustness, instead of redesigning a new algorithm; 2) *general*: We would like the solution to make minimal assumption of specific matching approaches so that it can be widely applicable to different holistic matchers.

As a result, our solution builds on the critical insight that while large scale schema matching poses the robustness problem, the solution lies in the holistic nature itself. In particular, we observe that holistic schema matching only needs *sufficient* but not *all* schema data. As attribute information are repeatedly used across large scale deep Web sources, *e.g.*, with enough book sources presenting attributes *author*, *title*, ..., a subset of schemas may still contain sufficient information to represent the complete data set. Thus, we in fact need only sufficient correct schemas, instead of all of them, to execute a holistic matcher.

While this insight is promising, how to develop a principled

framework to realize it? The key challenges exist on both the input and output sides of the solution. On the input side, as we cannot generally differentiate noisy schemas from correct ones, it is infeasible to identify and remove noises. Thus, our approach has to essentially account for the presence of noisy data in the input. On the output side, our approach should give a predictable guarantee on the expected robustness, *e.g.*, how robust it can be when some characteristics of the schema data and the holistic matching algorithm are known.

To tackle these challenges, we develop a *data-ensemble* framework by exploiting sampling and voting techniques (Sections 3, 4). In particular, on the input side, we take an ensemble of multiple matchers, where each matcher is executed over an independent random sampling of the input schemas. On the output side, we take majority voting to aggregate the ranked results of all the matchers into a merged list of ranked matchings, which can alleviate the impact of noises and more accurately reflect the correct ranking of matchings. We build an analytic model to help us justify the effectiveness of this framework and predict its robustness.

We note that, our data-ensemble idea is inspired by *bagging predictors* [6] in machine learning— That is, we are essentially applying bagging techniques in a new scenario of mining matchings. Bagging predictors is a method for maintaining the robustness of “unstable” classification algorithms where small changes in the training set result in large changes in prediction. In particular, it creates multiple versions of a classifier, trains each classifier on a random redistribution of the training set and finally takes a plurality voting among all the classifiers to predict the class. Therefore, our data-ensemble approach has the same foundation as bagging predictors on exploiting majority voting to make an algorithm robust against outlier data in the input. In Section 3, we will further compare the difference of our framework with bagging predictors.

We evaluate the data-ensemble framework over our motivating integration scenario, *i.e.*, to make a holistic matcher robust against the noisy input from a schema extractor (Section 5). Our goals are twofold: (1) Verify our motivating observation that noises made by schema extraction can significantly affect the matching performance. (2) Validate the effectiveness of the data-ensemble framework over real Web data.

In our development, we also observe some open issues that warrant further research. Can we develop a systematic and principled method to determine the configuration of this data-ensemble framework? How to solve the uncertainty problem of matching result? What is the applicability of this framework? We discuss these open issues in Section 6.

In summary, the contributions of this paper are:

- As our **problem**, we identify noisy data quality as an inherent challenge for leveraging holistic quantity in large scale schema matching. Such a data quality problem is critical for sustaining holistic schema matching as a practical and viable technique.
- As our **solution**, we develop a data-ensemble framework with sampling and voting techniques, inspired by bagging predictors. We are essentially applying bagging techniques in a new scenario of mining semantic correspondences among attributes. Our experiments show the promise of this framework.

The rest of the paper is organized as follows: Section 2 reports issues we find in integrating schema extraction and holistic schema matching, and then motivates the data-ensemble framework. Section 3 models the data-ensemble framework and provides analytic justification of its effectiveness. Section 4 discusses technical details. Section 5 reports our experiments. Section 6 discusses several further opportunities and open issues and then concludes the paper.

2. MOTIVATION

As Section 1 discussed, toward building large scale schema integration systems, it is critical to integrate holistic schema matching with automatic schema extraction. This system integration inevitably raises a new challenge of noisy data quality, which has not been extensively investigated. In particular, what we have observed, when integrating schema matching with extraction, is the problem of *error cascade*— That is, the inevitable errors made by automatic schema extraction may cascade to holistic schema matching and thus significantly affect the matching performance. In this section, we first define, by way of brief summary, the two “subsystems” (*i.e.*, schema extraction and holistic schema matching) to be integrated, based on which we observe the error cascade phenomenon in putting them together (Section 2.1) and further motivate the insight of our solution (Section 2.2).

Schema Extraction [Subsystem SE]:

The subsystem *SE* extracts the schema information of a Web query interface in its HTML format. For instance, given the advanced book search of *amazon.com*, *SE* will extract its schemas as a set of query conditions, as shown in Figure 2. In particular, `[Author; {contain}; text]` means the value of attribute `author` can be filled with any text; `[Format; {=}; {hardcopy, paperback, ...}]` means the value of attribute `format` has to be selected from a give set of options.

Recent works [21, 12] developed automatic techniques for such schema extraction. Reference [21] introduces a parsing approach by hypothesizing the existence of hidden syntax, which connects attribute semantics of a query interface to its visual layout in the Web page. Reference [12] proposes a two-step extraction algorithm by first translating the HTML text into an internal interface expression (or IEXP) and then recognizing attribute semantics (*e.g.*, labeling of attribute name and grouping of elements) from the translated IEXP based on some rules.

Holistic Schema Matching [Subsystem SM]:

Given a set of schemas, the schema matching subsystem is to discover semantic correspondences (*i.e.*, matchings) among attributes. Some recent schema matching works specifically focus on discovering matchings among a set of query interfaces [9, 11, 10, 19]. Unlike traditional schema matching, which mostly targets at small scale integration by finding pairwise attribute correspondences between two schemas [17, 8, 16, 13, 15], these works match schemas in a “holistic” way by taking many schemas as input and outputting all the matchings among the input schemas. Since for any holistic matcher, regardless its matching techniques, each discovered matching is quantified with a “confidence” score, its output is thus a ranked list of scored matchings. Therefore, we abstract a holistic matcher as a module whose input is a set of schemas and output a ranked list of matchings, as Figure 1 shows.

Sharing the same abstraction of holistic schema matching, there are different realizations. In particular, the *MGS* approach [9] abstracts schema matching problem as hidden model discovery by hypothesizing the existence of a hidden schema model, which generates schemas with probabilistic behavior. The *DCM* approach [10] tackles the problem of finding complex matchings with a correlation mining approach, based on the observation that co-occurrence patterns across schemas often reveal complex semantic relationships. Reference [19] pursues a clustering-based matching approach by exploring the “bridging” effect among schemas. *WISE* [11] is a comprehensive query interface integrator, which combines multiple matching techniques such as clustering.

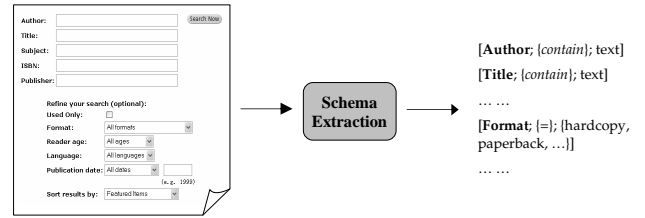


Figure 2: Subsystem *SE*: Schema extraction.

2.1 The Problem: Error Cascade

As Section 1 discussed, integrating *SE* and *SM* subsystems is a critical step toward building large scale schema integration systems. In our development, we choose the Schema Extractor we proposed in [21] and the *DCM* matcher we developed in [10] as our testbed subsystems to integrate. Our goal is to see whether *SE* can sustain the schema matching task *SM* with extraction errors. In particular, given Web pages containing query interfaces in the same domain as the input, we use the Schema Extractor to extract the schema of each interface and then execute the *DCM* matcher on all the extracted schemas to discover matchings.

While we must integrate the two complementary subsystems to build a complete system for automating holistic schema matching, can the “accuracy” of *SE* sustain the demand of data quality for *SM*? As [21] reported, when studied in isolation, *SE* delivers 85-90+% accuracy— thus it will make about 1-1.5 mistake for every 10 query conditions to extract. While seemingly satisfactory, putting in the context of the integrated system, is this accuracy good enough? As our experiment shows in Section 5, with noisy input, the accuracy of the holistic matcher may degrade up to 30%, comparing to the results reported in [10], which shows that errors indeed cascade along the execution of subsystems.

The performance degradation results mainly from the negative impact of the noisy input on the right ranking of matchings in the output of *SM*. When input schemas are noisy, the ranking of matchings is likely to be affected (*i.e.*, incorrect matchings maybe ranked higher than correct ones). Consequently, the ranking is less reliable for the “consumer” applications of *SM* to select correct matchings. For instance, an application-specific matching selection step is often introduced after *SM* to choose the most promising subset of matchings among all the discovered ones. Since such a selection naturally relies on the ranking of matchings, it is critical to make *SM* still output a good ranking with the presence of noises.

2.2 The Solution: Insight

While large scale data integration brings forward the inherent problem of noisy quality in schema extraction, the large scale also lends itself to an intriguing potential solution. An interesting question to ask is: *Do we need all input schemas in matching their attributes?* In principle, since pursuing a data mining approach, holistic schema matching exploits “statistics-based” evaluation (*e.g.*, clustering, correlation mining) in nature and thus needs only “sufficient observations.” As query interfaces tend to share attributes, *e.g.*, `author`, `title`, `subject`, `ISBN` are repeatedly used in many book sources, a subset of schemas may still contain sufficient information to “represent” the complete set of schemas. Thus, the holistic matcher in fact only needs sufficient correct schemas to execute, instead of all of them. This insight is promising, but it also brings a new challenge: As there is no way to differentiate noisy schemas with correct ones, how should we select the schemas to guarantee the robustness of our solution?

Tackling this challenge, we propose a *data-ensemble* framework, with sampling and voting techniques, to build upon and extend an existing holistic matcher, and meanwhile maintain its robustness. To begin with, we consider to execute the holistic matcher on a randomly sampled subset of input schemas. Such a *downsampling* has two attractive characteristics: First, when schemas are abundant, it is likely to contain sufficient correct schemas to be matched. Second, by sampling away some schemas, it is likely to contain fewer noises and thus has more chance to sustain the holistic matcher.

Further, while a single downsampling may (or may not) achieve good result, as a randomized scheme, the expected robustness can only be realized in “statistical” sense— Thus, we propose to take an ensemble of multiple matchers, where each matcher is executed over an independent downsampling of schemas. We expect the majority of these matchers have better results than directly running the matcher on all the schemas. Thus, by taking majority voting among these matchers, we can achieve a much better matching accuracy.

As Section 1 discussed, this data-ensemble idea essentially applies bagging techniques [6] in machine learning. However, our approach is different from bagging predictors in several aspects. First, *setting*: We apply the idea of the ensemble of randomized data for unsupervised learning (e.g., in our scenario, holistic schema matching with statistical analysis), instead of supervised learning, which bagging predictors is developed for. Second, *techniques*: Our concrete techniques are different from bagging predictors. In particular, in the sampling part, we take a downsampling other than random redistribution with replacement; in the voting part, we need to aggregate a set of ranked lists, which is more complicated than aggregate a set of labels in bagging predictors. Third, *analytic modeling*: We build an analytic modeling specific to our holistic schema matching scenario (Section 3), which enables us to validate the effectiveness of a particular configuration.

The following two sections will discuss in details about this data-ensemble framework. In particular, we first more formally model this framework and analyze its effectiveness (Section 3). Then we will present the technical details we developed for the data-ensemble framework (Section 4).

3. THE DATA-ENSEMBLE FRAMEWORK

In this section, we present our modeling of the data-ensemble framework (Section 3.1), based on which we can more formally analyze its effectiveness (Section 3.2).

3.1 Modeling

We develop a general modeling to formalize the data-ensemble framework motivated in Section 2.2. In particular, a schema extractor outputs a set of N schemas, denoted by $I = \{Q_1, Q_2, \dots, Q_N\}$. We denote the *SM* subsystem as an abstract module A . A holistic matcher A thus takes I as input and outputs a ranked list of matchings, denoted by R_I^A . Figure 3(a) illustrates the “base framework” of simply concatenating *SE* and *SM*, which suffers the error cascade problem and thus needs to be enhanced. Unlike the base framework in Figure 3(a), the data-ensemble framework views the matching module A as a black box *base algorithm* and extends it by exploiting sampling and voting techniques. Therefore, we need to first model the “behavior” of A and then the setting of the data-ensemble framework.

Overall, the essential goal of A is to generate matchings M_1, \dots, M_n in a correctly ranked order under the impact of imperfect data quality. In our modeling, we will focus on the impact of noises on a single matching M . As we will discuss later, our analysis should generally assume a representative “worst-case” matching,

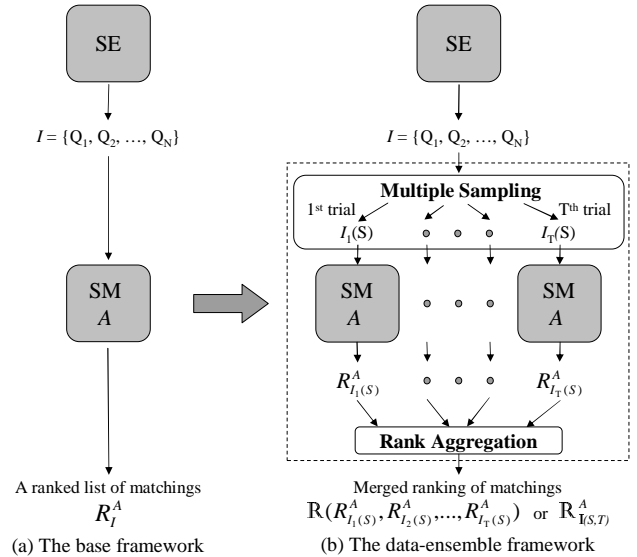


Figure 3: Integration of subsystems *SE* and *SM*.

based on which our analysis for a single matching can also cover all the matchings.

Specifically, given a set of N schemas I as input, assume there are W problematic schemas (i.e., noises) that affect the ranking of M . Suppose the holistic matcher A can correctly rank M if one trial draws no more than K noises ($K < W$)— i.e., in which case, M as a correct matching can actually be ranked higher.

Next, we need to model the data-ensemble framework, which consists of two steps: *multiple sampling* and *rank aggregation*, as Figure 3(b) shows. *First*, in the multiple sampling step, we conduct T downsamplings of the input schemas I , where each downsampling is a subset of independently sampled S schemas from I . We name such a downsampling as a *trial* and thus have T trials in total. We denote i th trial as $I_i(S)$ ($1 \leq i \leq T$). By executing the base algorithm A over each trial $I_i(S)$, we get a ranked list of matchings $R_{I_i(S)}^A$. *Second*, the rank aggregation step aggregates ranked matchings from all the trials, i.e., $R_{I_i(S)}^A$ ($1 \leq i \leq T$), into a merged list of ranked matchings, which we denote as $\mathbb{R}(R_{I_1(S)}^A, \dots, R_{I_T(S)}^A)$, or $\mathbb{R}_{I(S,T)}^A$ in short. We expect the aggregate ranking $\mathbb{R}_{I(S,T)}^A$ can alleviate the impact of noises and thus is better than R_I^A .

Since W is determined by “inherent” characteristics of input schemas I and K by the holistic matcher A , we name them as *base parameters*. Unlike W and K , the sampling size S and the number of trials T are “engineered” configurations of the data-ensemble framework and thus named as *framework parameters*.

3.2 Analysis

Our goal of analysis is to justify, given estimation of the base parameters, W and K , which characterize the data quality and the base algorithm, can certain configuration, in terms of S and T , of the data-ensemble framework achieve robustness?

In particular, given our modeling, we can derive the probability to correctly rank M in a single trial, which we name as *hit probability*, i.e., the chance of “hit” a correct ranking of M in a single trial (and as we will discuss later, we will do more trials to enhance the overall hit ratio). Given base parameters W and K of M , hit probability is a function of S (and not T as it is for a single trial) and thus denoted as $\alpha_M(S)$. To derive $\alpha_M(S)$, we first compute

the probability that there are exactly i noises in a single trial, denoted by $Pr(k = i|S)$, *i.e.*, with i noises out of W and $S - i$ correct ones out of $N - W$:

$$Pr(k = i|S) = \frac{\binom{W}{i} \binom{N-W}{S-i}}{\binom{N}{S}} \quad (1)$$

As our model assumes, M can be correctly ranked when there are no more than K noises. We thus have:

$$\alpha_M(S) = \sum_{i=0}^K Pr(k = i|S) \quad (2)$$

Next, we are interested in how many times, among T trials, can we observe M being ranked correctly? This problem can be transformed as the standard scenario of tossing an unfair coin in statistics: Given the probability of getting a “head” in each toss as $\alpha_M(S)$, with T tosses, how many times can we observe heads? With this equivalent view, we know that the number of trials in which M is correctly ranked (*i.e.*, the number of tosses to observe heads), denoted by O_M , is a random variable that has a binomial distribution [2] with the success probability in one trial as $\alpha_M(S)$. We use $Pr(O_M = t|S, T)$ to denote the probability that M is correctly ranked in exactly t trials. According to the binomial distribution, we have

$$Pr(O_M = t|S, T) = \frac{T!}{t!(T-t)!} \alpha_M(S)^t (1 - \alpha_M(S))^{T-t} \quad (3)$$

Since our goal is to take majority voting among all the trials (in rank aggregation), we need a sufficient number of trials to ensure that M is “very likely” to be correctly ranked in the relative majority of trials. As an analogy, consider the coin tossing: Even the probability to get a head in each toss is high, say 0.8, we may not always observe $0.8 \times T$ heads in T trials; the actual number of heads may even be a minority of trials— And our goal is to design a T such that “the number of heads” is very likely to be the majority. We thus need a sufficient number of trials to enable the majority voting. We name the probability that M can be correctly ranked in the majority of trials (*i.e.*, more than half of trials) as *voting confidence*. Voting confidence is a function of T (as just intuitively observed) and S (as it also depends on $\alpha_M(S)$ and thus S). We denote the voting confidence as $\beta_M(S, T)$. In particular, we have

$$\beta_M(S, T) = \sum_{t=\frac{T+1}{2}}^T Pr(O_M = t|S, T). \quad (4)$$

As a remark, in Equation 4, we constrain T as an odd number and thus $\frac{T+1}{2}$ is the minimum number of trials needed to be the majority¹.

Our modeling essentially captures the functional relationship of the sampling size S and the number of trials T to together achieve a desired voting confidence. The interpretation of Equation 4 is: Given S and T , we can use Equation 4 to evaluate how effective the framework is. In particular, we illustrate with Example 1 as a basis of understanding how the framework works.

¹When T is odd, the notion of majority is always well defined, as there are no ties (of equal halves).

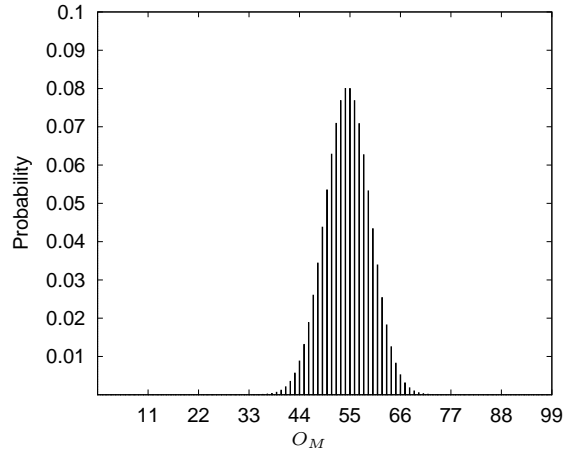


Figure 4: The binomial distribution of O_M , with $T = 99$ and $\alpha_M(S) = 0.55$.

Example 1: Assume there are 50 input schemas (*i.e.*, $N = 50$). As characteristics of the data quality and the base algorithm, suppose a matching M cannot be correctly ranked because of 6 noisy schemas (*i.e.*, $W = 6$); on the other hand, suppose M can be correctly ranked if there are no more than two noisy schemas (*i.e.*, $K = 2$). Also, as the configuration of the data-ensemble framework, suppose we want to sample 20 schemas in a single trial and conduct 99 trials (*i.e.*, $S = 20$ and $T = 99$).

According to Equation 1, in any single trial, we have 0.04 probability to get no noisy schema, 0.18 probability with one and 0.33 probability with two. Together, we have $0.04 + 0.18 + 0.33 = 0.55$ probability to correctly rank M in one trial (*i.e.*, $\alpha_M(S) = 0.55$).

Further, Figure 4 shows the binomial distribution of O_M . Going back to the coin tossing analogy, this figure essentially shows, if the probability to get a head in one toss is 0.55, after tossing 99 times, the probability of observing a certain number of heads. For instance, we have $Pr(O_M = 50|S, T) = 0.05$, which means the probability to observe 50 heads in 99 tosses is 0.05. According to Equation 4, we have 0.84 voting confidence to correctly rank M (or observe heads) in more than 49 trials (or tosses) (*i.e.*, $\beta_M(S, T) = 0.84$). Therefore, even $\alpha_M(S)$ is not very high, *e.g.*, 0.55 in this example, with sufficient number of trials, it is still very likely that M can be correctly ranked in the majority of trials. ■

Finally, while our analysis above focuses on a single matching, there are multiple matchings, M_1, M_2, \dots, M_n , to discover. We note that our analysis can generally assume a representative “worst-case” matching, based on which the analysis will also cover all the matchings. Specifically, the above modeling can be applied to any M_i with its corresponding W_i and K_i values. We then assume there is a “worst-case” matching M^* with base parameters W^* and K^* . We want to show that if we are likely to correctly rank M^* in the majority of trials under some setting, we are even more likely to correctly rank all the matchings M_1, M_2, \dots, M_n in the majority of trials with the same setting.

We show that the base parameters of the imaginary “worst-case” matching M^* can be set as $W^* = \max W_i$ and $K^* = \min K_i$, $1 \leq i \leq n$. Intuitively, the higher W is, the lower $\alpha_M(S)$ will be because we have more noises in the input schemas I ; on the other hand, the lower K is, the lower $\alpha_M(S)$ will be because the base algorithm A is less robust against noises. More formally, we can show that $\alpha_M(S)$ is monotonically decreasing with respect to W and monotonically increasing with respect to K . (The derivation is

straightforward and thus we do not provide a proof here.) Therefore, if we assume a matching M^* with base parameters W^* as the maximal value of W_i and K^* the minimal value of K_i , we have $\alpha_{M_i}(S) \geq \alpha_{M^*}(S)$ any matching M_i ($1 \leq i \leq n$).

Further, we can show that all the matchings also have higher voting confidence than M^* . Intuitively, if a matching M has higher hit probability, M should be more likely to be observed in the majority of trials, which means it also has a higher voting confidence. In particular, we can show that $\beta_M(S, T)$ is monotonically increasing with respect to $\alpha_M(S)$. (Similarly, the derivation is straightforward and thus we do not provide a proof here.) Therefore, since $\alpha_{M_i}(S) \geq \alpha_{M^*}(S)$ ($1 \leq i \leq n$), we have $\beta_{M_i}(S, T) \geq \beta_{M^*}(S, T)$ ($1 \leq i \leq n$). This inequality indicates that M^* is indeed the “worst-case” matching. Specifically, if we can find an appropriate setting of S and T to correctly rank M^* in the majority of trials with high confidence, we will have even more confidence to correctly rank all the matchings in the majority of trials with the same setting of S and T .

4. TECHNICAL DETAILS

Section 3 modeled and analyzed the data-ensemble framework in an abstract view; in this section, we discuss the technical details in our development. First, the multiple sampling step is straightforward and the only thing we need to consider is to determine the sampling size S and the number of trials T . As we will discuss in Section 4.1, in our current development, we empirically choose S and T values that can achieve the best performance. Second, we need to develop a rank aggregation strategy to aggregate the matching results from all the trials into a merged ranked list of matchings. We discuss this issue in Section 4.2.

4.1 Sampling and Trials: Configuration

The first phase of the data-ensemble framework is to choose appropriate sampling size and number of trials. On one hand, we want to reduce unnecessary downsampling. A very small S value may not be able to collect enough schemas to represent the complete input data and consequently degrade the accuracy of the base matching algorithm. However, a larger S may contain more noises and thus also affect the accuracy of the matching result. Therefore, it is important to choose an appropriate S value to achieve good matching performance. (Our experiment in Section 5 also reflects this observation.)

On the other hand, we want to reduce unnecessary trials. As Section 3.2 discussed, the more trials we have, the higher voting confidence will be. Considering the execution time of the data-ensemble framework, we do not want to be over-tried; therefore, within all the settings that can produce acceptable performance, we prefer the one with a smaller T .

In our development, we empirically determine the S and T values that achieve the best performance, as Section 5 will illustrate. We notice that systematically choosing the best (S, T) pair is a problem that deserves further investigation, since the best setting may be various in terms of different input data and base matching algorithms. In Section 6, we will discuss our future plan of developing a more principled approach to choosing S and T .

4.2 Voting: Rank Aggregation

The second phase of the data-ensemble framework is to aggregate rankings $R_{I_1(S)}^A, \dots, R_{I_T(S)}^A$ from the T trials into a merged list of ranked matchings $\mathbb{R}_{\mathbb{I}(S, T)}^A$. The main issue we are facing in this phase is thus to develop a rank aggregation strategy that can

make $\mathbb{R}_{\mathbb{I}(S, T)}^A$ reflect the rankings of matchings in the majority of $R_{I_1(S)}^A, \dots, R_{I_T(S)}^A$.

We notice that the rank aggregation in our situation is slightly different from the traditional rank aggregation problem. Traditional rank aggregation assumes all voters share the same set of candidates and only rank them in different orders. In contrast, in our scenario, no candidates are given before executing the base algorithm and each trial outputs its own matching result. Therefore, before aggregate rankings, we need to have a candidate selection step to select matching candidates.

Consequently, the rank aggregation phase consists of two sub-steps: 1) Candidate selection: To select candidates from each $R_{I_i(S)}^A$ to form a common pool of candidates \mathcal{C} . 2) Rank aggregation: To aggregate the T rankings $PR_{I_1(S)}^A, \dots, PR_{I_T(S)}^A$ into $\mathbb{R}_{\mathbb{I}(S, T)}^A$, where $PR_{I_i(S)}^A$ is the “projected” ranking of $R_{I_i(S)}^A$ on \mathcal{C} , as we will discuss below.

Candidate Selection

We select candidates based on the intuition that if a matching M is only discovered by a minority of trials, M is more likely to be a false matching. Therefore, we consider a matching as a candidate if it appears in the majority of T rankings, $R_{I_1(S)}^A, \dots, R_{I_T(S)}^A$. All the matchings whose numbers of occurrences are less than $\frac{T+1}{2}$ are thus pruned.

Let \mathcal{C} denote the union of all the candidates in each $R_{I_i(S)}^A$. After candidate selection, we will remove the non-candidate matchings from each $R_{I_i(S)}^A$ and meanwhile preserve the ordering of candidates; the corresponding new ranked list, which can be viewed as a “projection” of $R_{I_i(S)}^A$ on \mathcal{C} , contains only candidates and is denoted as $PR_{I_i(S)}^A$.

Example 2: Assume we execute the base algorithm A on three trials, *i.e.*, $T = 3$, and the outputs are thus three ranked lists $R_{I_1(S)}^A$, $R_{I_2(S)}^A$ and $R_{I_3(S)}^A$. Suppose $R_{I_1(S)}^A$ outputs ranking $M_1 > M_2 > M_3 > M_4$ in descending order, $R_{I_2(S)}^A$ outputs $M_2 > M_1 > M_3 > M_5$, and $R_{I_3(S)}^A$ outputs $M_3 > M_1 > M_2 > M_4$.

Since $\frac{T+1}{2} = 2$, any matching that occurs only once will be pruned. In particular, M_5 is pruned; other matchings, M_1, M_2, M_3 and M_4 , all at least occur twice and thus are selected as matching candidates. Therefore, we have $\mathcal{C} = \{M_1, M_2, M_3, M_4\}$.

The projected rankings are thus $PR_{I_1(S)}^A: M_1 > M_2 > M_3 > M_4$, $PR_{I_2(S)}^A: M_2 > M_1 > M_3$, and $PR_{I_3(S)}^A: M_3 > M_1 > M_2 > M_4$. In particular, M_5 does not appear in $PR_{I_2(S)}^A$ because it has been pruned. ■

Rank Aggregation

In rank aggregation, we need to construct an ordered list $\mathbb{R}_{\mathbb{I}(S, T)}^A$ for the candidates in \mathcal{C} , based on the individual ranks $PR_{I_1(S)}^A, \dots, PR_{I_T(S)}^A$. This problem is essentially a *rank aggregation* problem, which has been extensively studied as a particular *voting* system in social science [14, 5]. In the literature, many rank aggregation strategies have been proposed, such as Borda’s aggregation [5] and Kemeny optimal aggregation [14]. There does not exist an aggregation strategy that can beat other strategies in all aspects— Different strategies have different strength and weakness.

Before discussing concrete aggregation strategies, we first need to solve the partial list problem. Specifically, since the output of one trial may not contain all the candidates in \mathcal{C} , $PR_{I_i(S)}^A$ may be only a partially ordered list. To be able to apply the aggregation strategy (as we will discuss below), it is necessary to also assign ranks to the candidates not in the list. In our development, given a

trial with a partial list, we assign all the uncovered candidates with the same lowest rank. Therefore, in one trial, a covered candidate is always ranked higher than an uncovered one, and two uncovered candidates are equally ranked.

Although essentially any rank aggregation strategy can be applied in our scenario, in our development, we choose Borda’s aggregation [5]. A primary strength of Borda’s aggregation is that it is computationally very efficient: It can be implemented in linear time. Also, it satisfies the properties called anonymity, neutrality, and consistency in the social science community [20].

Specifically, in Borda’s aggregation, given a candidate M_j , let r_{ji} be number of matchings ranked lower than M_j in $PR_{I_i(S)}^A$, the *borda score* of M_j , denoted as $B(M_j)$, is defined as the sum of all r_{ji} , i.e., $B(M_j) = \sum_{k=1}^T r_{jk}$. The aggregation result $\mathbb{R}_{I(S,T)}^A$ is thus the descending ordering of all the candidates with respect to their borda scores.

Example 3: Continue on Example 2, after candidate selection, we first complete the partial lists. In particular, since $PR_{I_2(S)}^A$ only partially ranks the four candidates, we assign the lowest rank to the uncovered candidate M_4 , i.e., we rank M_4 as the 4th candidate in $PR_{I_2(S)}^A$.

Next, we compute the borda score for each candidate and then apply Borda’s aggregation. In particular, since M_1 is ranked higher than 3 candidates in $PR_{I_1(S)}^A$, 2 in $PR_{I_2(S)}^A$ and 2 in $PR_{I_3(S)}^A$, the borda score for M_1 is $3 + 2 + 2 = 7$. Similarly, the borda scores for M_2 to M_4 are 6, 5, 0 respectively. The final ranking $\mathbb{R}_{I(S,T)}^A$ is thus $M_1 > M_2 > M_3 > M_4$. ■

5. EXPERIMENTS

We evaluate the data-ensemble framework over our motivating integration scenario in Section 2.1, i.e., to make a holistic matcher robust against the noisy input schemas from a schema extractor. In particular, we implement the framework in Python 2.4 and test all the experiments on a Windows XP machine with Pentium M 1.6GHz CPU and 512M memory. We integrate concrete matching and extraction subsystems, and test both the base and data-ensemble frameworks over the integrated system with real query interfaces as input.

In particular, we test our approach over real query interfaces in two domains: Books and Airfares. First, we test the base framework by directly running a holistic matcher on the real data, which on one hand shows the error cascade problem, and on the other hand sets the baseline result we will compare to. Second, we run the data-ensemble framework over the real data and compare its accuracy with the baseline result. The result shows that our framework can significantly improve the accuracy of a holistic matcher. Third, to help us find the optimal configuration setting, we execute the data-ensemble framework under various parameter values. Section 5.1 discusses our experiment setup and Section 5.2 shows the experimental result.

5.1 Experiment Setup

As discussed in Section 2.1, we integrate the Schema Extractor developed in [21] and the DCM matcher in [10]. This integrated system naturally becomes our testbed of the data-ensemble framework over real data. In particular, the DCM matcher discovers not only simple 1:1 matchings but also complex $m:n$ matchings, e.g., `author = {last name, first name}`, with a correlation mining approach. As a mining approach in nature, DCM takes a set of schemas as input and outputs a list of discovered matchings ranked by a correlation measure, which fits our general modeling of a holistic matcher and is thus a qualified testbed.

To test the effectiveness of the data-ensemble framework over real data, we apply our approach to deep Web sources in two representative domains, Books and Airfares, in the TEL-8 dataset of the UIUC Web Integration Repository [1]. For each source, we use the Schema Extractor to automatically extract schemas from Web query interfaces. Then, for each domain, we use the DCM matcher to discover matchings among the extracted schemas.

To have a fair comparison of the matching results, we adopt the matching selection step and accuracy metrics developed in [10]. First, a *greedy* selection strategy is proposed in [10] to select a subset of most promising matchings among all the discovered ones. As the final matching accuracy is evaluated on selected matchings, to fairly compare the results of the data-ensemble framework and the base framework, we apply the same selection strategy to choose a subset of matchings from the aggregate ranking $\mathbb{R}_{I(S,T)}^A$.

Second, to evaluate the accuracy of selected matchings, we adopt the same metric, *target accuracy*, as DCM was validated. In particular, to compare selected matchings, denoted by \mathcal{M}_h , with correct matchings written by human experts, denoted by \mathcal{M}_c , reference [10] introduces a new term, *closenym*: Two attributes are *closenym* if they have one of the synonym, hyponym and hypernym relationships. Given an attribute A_j , its *closenym set* is the set of attributes that are *closenym*s of A_j with respect to a matching result \mathcal{M} , denoted as $Cls(A_j|\mathcal{M})$. The *target precision* and *target recall* of \mathcal{M}_h with respect to \mathcal{M}_c are defined as:

$$P(\mathcal{M}_h, \mathcal{M}_c) = \sum_{A_j} \frac{O_j}{\sum O_k} \frac{|Cls(A_j|\mathcal{M}_c) \cap Cls(A_j|\mathcal{M}_h)|}{|Cls(A_j|\mathcal{M}_h)|}$$

$$R(\mathcal{M}_h, \mathcal{M}_c) = \sum_{A_j} \frac{O_j}{\sum O_k} \frac{|Cls(A_j|\mathcal{M}_c) \cap Cls(A_j|\mathcal{M}_h)|}{|Cls(A_j|\mathcal{M}_c)|}$$

In the above definitions, the precision and recall of each individual attribute is weighted by $\frac{O_j}{\sum O_k}$, where O_j is the frequency of attribute A_j in the dataset (i.e., its number of occurrences in different schemas).

5.2 Experimental Result

The baseline result: The baseline result we will compare to is the result of executing the base framework (Figure 3(a)). The fourth and fifth columns in Figure 5 show the result, where the fourth column is the target precision and the fifth column the target recall. Comparing to the result reported in [10], where the input schemas are perfectly extracted, the accuracy of the base framework significantly degrades due to the error cascade problem. To make the comparison more illustrative, we list the corresponding accuracies of [10] in the second and third columns. We can see that the degradation of target accuracy is up to 30%. This accuracy degradation is mainly because the existence of noises affects the ranking of matchings and thus the result of matching selection.

The result of the data-ensemble framework: Next, we test the data-ensemble framework on the two domains with empirically obtained optimal parameter settings. In particular, for Books, we set the sampling size S as 20 and the number of trials T as 41; for Airfares, we set S as 16 and T as 41. As Section 6 will discuss, one of our future work is to find a systematic and principled way to determine S and T , probably building upon our analytic modeling in Section 3.

As the data-ensemble framework is essentially a data-randomized approach (with multiple random trials), it is “non-deterministic”—We thus measure the distribution of its performance. Specifically, we execute the framework 100 times on Books with the same setting $S = 20, T = 41$. Similarly, we execute it 100 times on Airfares

Domain	The base algorithm				The data-ensemble framework			
	perfect input		noisy input		average accuracy		best accuracy	
	precision	recall	precision	recall	precision	recall	precision	recall
Books	1.0	1.0	0.73	0.75	0.83	0.89	0.93	1.0
Airfares	1.0	1.0	0.67	0.68	0.79	0.79	1.0	0.88

Figure 5: The comparison of target accuracy on two domains.

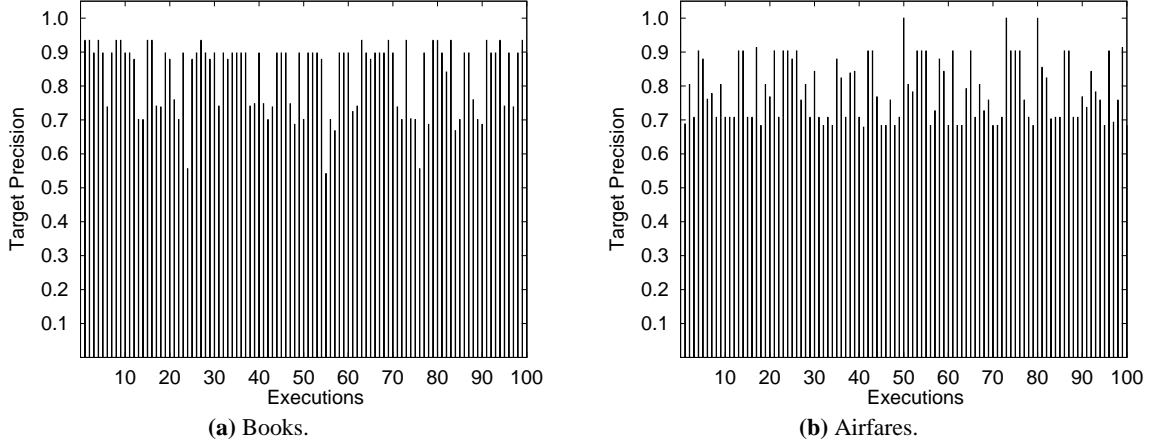


Figure 6: The target precision with 100 executions on two domains.

with the same setting $S = 16$, $T = 41$. To quantify the comparison with the baseline result, we measure two suites of target accuracies: the average target accuracy (*i.e.*, the average precision and recall of the 100 executions) and the best target accuracy (*i.e.*, the best precision and recall of the 100 executions).

The results of both average and best accuracies are listed in Figure 5 (columns 6-9). We can see that, comparing to the baseline result, both the precision and recall are improved. In particular, in most executions, the data-ensemble framework achieves better accuracy than the baseline result. For instance, Figure 6 shows the 100 target precisions of the 100 executions over Books and Airfares. We observe that, although accuracies may be various in different executions, most precisions in both Books and Airfares are better than their corresponding baseline precisions. Similar result can also be observed in Figure 7 for target recall. (Figure 7 looks more regular than Figure 6 because for recall, only the value on numerator is changing, while for precision, values on both numerator and denominator are changing.) Hence, this experiment indicates that the data-ensemble framework can indeed boost the matching accuracy under noisy schema input, and thus maintain the desired robustness of a holistic matcher.

The execution time of the data-ensemble framework is also acceptable. The 100 executions on Books take 118 seconds and on Airfares 109 seconds. Therefore, the average time for one execution is about only 1 second.

From Figure 6, we also observe an interesting phenomenon: It seems that there is an upper-bound of precision, which the data-ensemble framework cannot exceed. (The same phenomenon also exists in the recall part in Figure 7.) The existence of such an upper bound is because, in essence, there are two types of data quality problems, noises and missing data, and the data-ensemble framework can deal with noises, but not missing data.

First, noises are some observed data that ideally should not be observed, *i.e.*, they are *outliers*. Although noises may affect the accuracy of the base algorithm, they are minority in quantity. Down-

sampling is thus a good approach to filtering them out and consequently, the majority voting can be effective.

Second, *missing data* are some data that ideally should be observed, but in reality are not. For this missing data case, sampling and voting techniques will not help, since when the entire dataset has missing data, all the trials will also have missing data and their aggregate result thus cannot fix the problem.

Therefore, the accuracy affected by missing data cannot be fixed by the data-ensemble framework. The accuracy upper-bound of the data-ensemble framework is thus $1 - \phi$, where ϕ is the accuracy degradation caused by missing data.

The result under various configuration settings: The purpose of this set of experiments is to help us empirically find the optimal parameter setting of S and T .

First, we measure the accuracy of the data-ensemble framework with different sampling sizes on the two domains. In particular, we fix T at 41 and let S progressively increase from 10 to 55 with an increment size 5 (*i.e.*, 10, 15, 20, ..., 55) for Books and from 10 to 40 with an increment size 3 for Airfares. For each sampling size, we execute the data-ensemble framework 30 times and compute the average precision and recall. Figure 8 shows the experimental result.

From Figure 8, we can observe the same trend in both domains—That is, when sampling size increases, the target precision mostly keeps on decreasing, while the target recall goes up first and then goes down at some point. We give the explanation as below: A small sampling size may miss some attributes in downsampling and thus discover less matchings, which results in trivially high precision but low recall. With larger sampling size, we are able to cover more attributes and thus discover not only more correct matchings, but also a few false matchings. Consequently, the precision decreases and recall increases. When the sampling size is too large, a downsampling is likely to have many noises and thus the recall starts to decrease again.

The best sampling size we should take is thus some values in

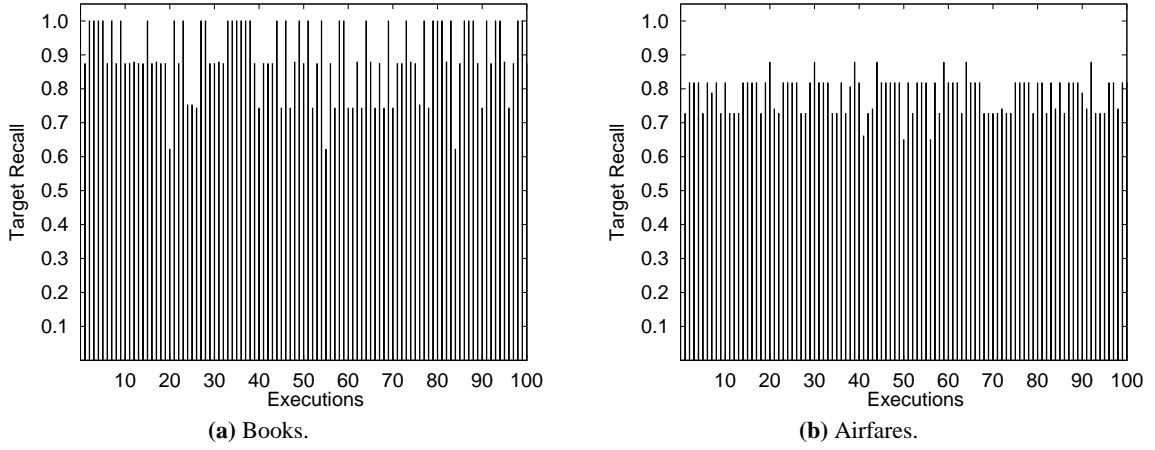


Figure 7: The target recall with 100 executions on two domains.

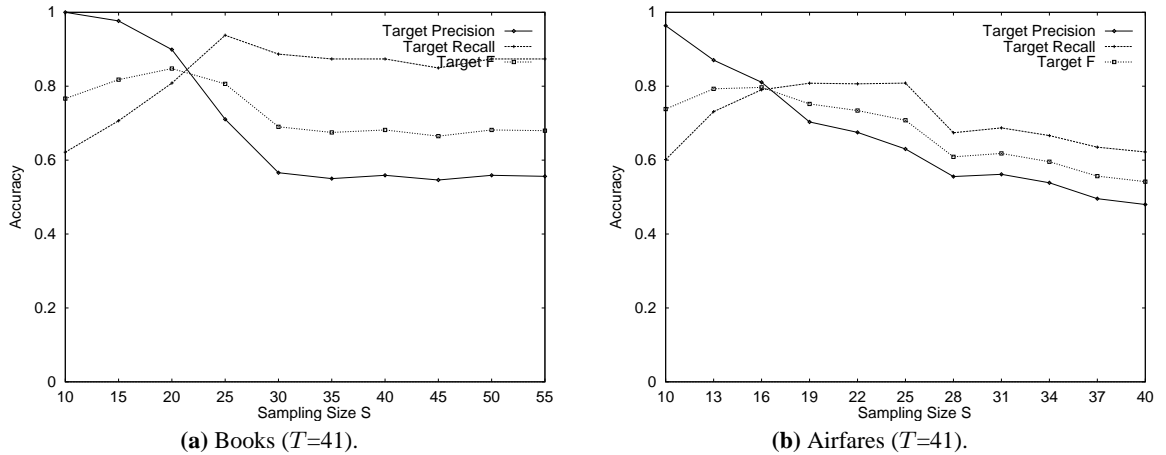


Figure 8: The target accuracy under various sampling sizes.

the middle. We choose the F -measure, which combines precision P and recall R as $F = \frac{2PR}{P+R}$, to measure the overall accuracy. From Figure 8(a) shows, we can see the best range of sampling size for Books, according to F -measure, is around 20. Similarly, from Figure 8(b), the best range of sampling size for Airfares is around 16. Therefore, we choose these two values as our optimal settings for S in Books and Airfares respectively.

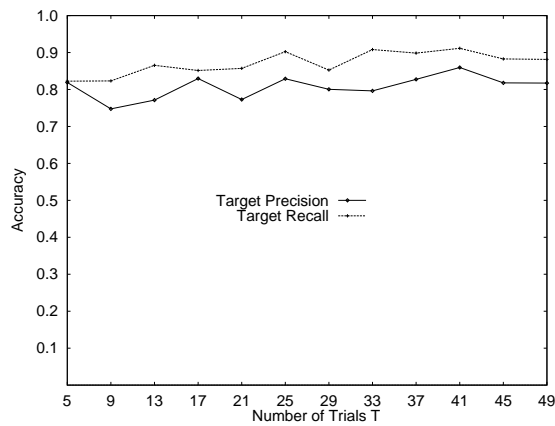
Second, we measure the accuracy of the data-ensemble framework with different numbers of trials on the two domains. In particular, we fix S at 20 for Books and 16 for Airfares. We change T from 5 to 49 with increment size 4 for both domains. For each T , we again execute the framework 30 times and compute the average precision and recall. Figure 9 shows the experimental result. From the result, we can see that, in both domains, both the precision and recall become more and more flat and stable when T increases. This result indicates that with as long as the T value is not very small, we can have roughly the same performance and thus the decision on T is not a critical factor. Specifically, according to Figure 9, setting T as 41 is good enough to obtain stable result in both domains.

6. CONCLUDING DISCUSSION

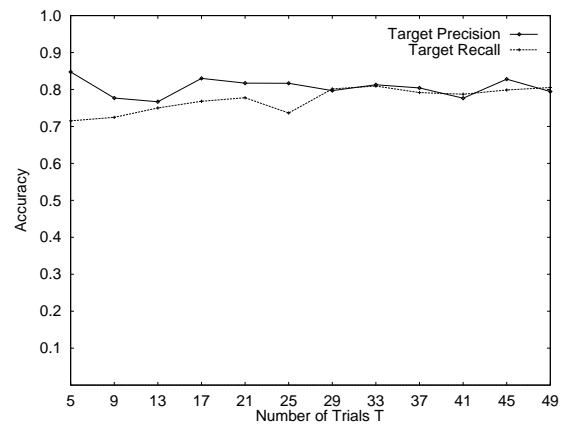
In our study for the data-ensemble framework, we also observed some open issues that warrant further research. First, in our cur-

rent development, we empirically set sampling size S and number of trial T ; thus, we naturally want to know whether it is possible to develop a principled method to automatically derive the appropriate S and T values. We notice that our analytic modeling in Section 3 can be a promising guidance for such a derivation. In particular, according to Equation 4, we can ask the question: *Given an objective voting confidence c , what are the appropriate values of S and T we should take to ensure $\beta_M(S, T) > c$?* The (S, T) pairs that satisfy the above requirements are our setting candidates. However, we need to solve two problems: 1) How to enumerate the space of all valid (S, T) pairs; 2) As the space often contains many (S, T) pairs, how can we evaluate whether a pair is a good setting or not. We plan to study this problem in our future work.

Second, since exploiting sampling techniques, the matching result of our framework may be various in each time; it is thus valuable to develop some strategy to address this uncertainty problem. In particular, we can again exploit the statistical voting strategy: Instead of running the framework only once, we execute the framework multiple times and choose the most frequently discovered matching result as the final output. On one hand, in a statistical sense, such a strategy should be able to deliver more stable matching result; on the other hand, since in most cases, the data-ensemble framework achieves better accuracy, such a strategy should also give a good matching quality, although may not be the best one.



(a) Books ($S=20$).



(b) Airfares ($S=16$).

Figure 9: The target accuracy under various number of trials.

Third, while this paper focuses on integrating holistic schema matching with schema extraction, to generalize, we believe the data-ensemble framework will be more widely applicable to the system integration of other large scale integration tasks. Our study is a first step toward understanding the system integration issue of building an integration system, which are often overlooked when we focus on well-abstracted and isolated tasks. Although this work addresses the noisy input problem in the context of integrating holistic schema matching with interface extraction, as our modeling and techniques are rather generic, we believe it will be more generally applicable beyond the holistic schema matching task. In our future work, we plan to apply this data-ensemble idea for other holistic matchers as well as other integration tasks.

In summary, this paper identifies robust quality as an inherent challenge for leveraging holistic quantity in large scale schema matching. Such a robustness issue inevitably arises in integrating holistic schema matching with automatic schema extraction. As the solution, we develop a data-ensemble framework with sampling and voting techniques, inspired by bagging predictors. We are essentially applying bagging techniques in a new scenario of mining semantic correspondences among attributes. Both the analytic justification and experimental result show the promise of our framework.

7. REFERENCES

- [1] The UIUC web integration repository. Computer Science Department, University of Illinois at Urbana-Champaign. <http://metaquerier.cs.uiuc.edu/repository>, 2003.
- [2] D. R. Anderson, D. J. Sweeney, and T. A. Williams. *Statistics for Business and Economics (Second Edition)*. West Pub. Co., 1984.
- [3] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
- [4] M. K. Bergman. The deep web: Surfacing hidden value. Technical report, BrightPlanet LLC, Dec. 2000.
- [5] J. C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.
- [6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [7] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.
- [8] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.
- [9] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *SIGMOD Conference*, 2003.
- [10] B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces: A correlation mining approach. In *SIGKDD Conference*, 2004.
- [11] H. He, W. Meng, C. Yu, and Z. Wu. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In *VLDB Conference*, 2003.
- [12] H. He, W. Meng, C. Yu, and Z. Wu. Automatic extraction of web search interfaces for interface schema integration. In *WWW Conference, poster paper*, 2004.
- [13] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD Conference*, 2003.
- [14] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88:571–591, 1959.
- [15] Y. Lee, A. Doan, R. Dhamankar, A. Halevy, and P. Domingos. imap: Discovering complex mappings between database schemas. In *SIGMOD Conference*, 2004.
- [16] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB Conference*, 2001.
- [17] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
- [18] L. Seligman, A. Rosenthal, P. Lehner, and A. Smith. Data integration: Where does the time go? *Bulletin of the Tech. Committee on Data Engr.*, 25(3), 2002.
- [19] W. Wu, C. T. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD Conference*, 2004.
- [20] H. P. Young. An axiomatization of borda's rule. *J. Economic Theory*, 9:43–52, 1974.
- [21] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *SIGMOD Conference*, 2004.