On Leveraging Crowdsourcing Techniques for Schema Matching Networks

Nguyen Quoc Viet Hung[1], Nguyen Thanh Tam[1], Zoltán Miklós[2], and Karl Aberer[1]

[1] École Polytechnique Fédérale de Lausanne
{quocviethung.nguyen,tam.nguyenthanh,karl.aberer}@epfl.ch
[2] Université de Rennes 1
zoltan.miklos@univ-rennes1.fr

Abstract. As the number of publicly-available datasets are likely to grow, the demand of establishing the links between these datasets is also getting higher and higher. For creating such links we need to match their schemas. Moreover, for using these datasets in meaningful ways, one often needs to match not only two, but several schemas. This matching process establishes a (potentially large) set of attribute correspondences between multiple schemas that constitute a *schema matching network*. Various commercial and academic schema matching tools have been developed to support this task. However, as the matching is inherently uncertain, the heuristic techniques adopted by these tools give rise to results that are not completely correct. Thus, in practice, a post-matching human expert effort is needed to obtain a correct set of attribute correspondences.

Addressing this problem, our paper demonstrates how to leverage crowdsourcing techniques to validate the generated correspondences. We design validation questions with contextual information that can effectively guide the crowd workers. We analyze how to reduce overall human effort needed for this validation task. Through theoretical and empirical results, we show that by harnessing natural constraints defined on top of the schema matching network, one can significantly reduce the necessary human work.

1 Introduction

There are more and more services on the internet that enable users to upload and share structured data, including Google Fusion Tables [13], Tableausoftware[1], Factual[2]. These services primarily offer easy visualization of the uploaded data as well as tools to embed the visualisation to blogs or Web pages. As the number of publicly available datasets grows rapidly and they are often fragmented into different sources, it is essential to create the interlinks between these datasets [7]. For example, in Google Fusion Tables, the coffee consumption data are distributed among different tables in that each table represents for a specific region [13]. In order to extract generic information for all regions, we need to aggregate and mine across multiple tables. This raises the challenges for interconnecting table schemas to achieve an integrated view of data.

[1] http://www.tableausoftware.com/public
[2] http://www.factual.com/

W. Meng et al. (Eds.): DASFAA 2013, Part II, LNCS 7826, pp. 139–154, 2013.

One of the major challenges in interconnecting the datasets is to establish the connections between attributes of individual schemas that describe the datasets. The process of establishing correspondences between the attributes of two database schemas has been extensively researched, and there is a large body of work on heuristic matching techniques[4, 22]. Beside the research literature, numerous commercial and academic tools, called schema matchers, have been developed. Even though these matchers achieve impressive performance on some datasets, they cannot be expected to yield a completely correct result since they rely on heuristic techniques. In practice, data integration tasks often include a post-matching phase, in which correspondences are reviewed and validated by human experts.

Given our application context, the large number of schemas and (possible) connections between them, the validation task would require an extreme effort. In this paper we demonstrate the use of crowdsourcing techniques for schema matching validation. Specifically, we study a setting in which the two schemas to be matched do not exist in isolation but participate in a larger matching network and connect to several other schemas at the same time. Beside interconnecting structured data on the Internet, there are a number of application scenarios in which such model can be applied, for example schema matching in large enterprises [18, 24] or service mashups [9].

Crowdsourcing techniques have been successfully applied for several data management problems, for example in CrowdSearch [26] or CrowdScreen [19]. McCann et al. [17], have already applied crowdsourcing methods for schema matching. In their work, they focused on matching a pair of schemas, but their methods are not directly applicable for the matching network that is our main interest. Leveraging network information, we define natural constraints that not only effectively guide the crowd workers but also significantly reduce the necessary human efforts.

Our contributions can be summarized as follows.

- We analyze the schema matching problem in networks whose schemas are matched against each other. On top of such networks, we exploit the relations between correspondences to define the matching network constraints.
- We design questions presented to the crowd workers in a systematic way. In our design, we focus on providing contextual information for the questions, especially the transitivity relations between correspondences. The aim of this contextual information is to reduce question ambiguity such that workers can answer more rapidly and accurately.
- We design an aggregate mechanism to combine the answers from multiple crowd workers. In particular, we study how to aggregate answers in the presence of matching network constraints. Our theoretical and empirical results show that by harnessing the network constraints, the worker effort can be lowered considerably.

The rest of the paper is structured as follows. The next section gives an overview of our framework. In Section 3, we describe how to design the questions that should be presented to crowd workers. In Section 4, we formulate the problem of aggregating the answers obtained from multiple workers. Section 5 clarifies our aggregate methods that exploit the presence of matching network constraints. Section 6 presents experimental results. Section 7 summarizes related work, while Section 8 concludes the paper.

2 Overview

Schema matching network is a network of schemas, together with the pairwise attribute correspondences between the attributes of the corresponding schemas. In our setting we suggest that these schema matching networks shall be constructed in the following two-step incremental process: (1) generate pairwise schema matchings using existing tools such as COMA [10] and AMC [20], (2) validate the generated matching candidates by crowd workers (i.e. decide whether the generated correspondence is valid or not). After the first step, the schema matching network is constructed and defined as a tuple $(\mathcal{S}, C)$, where $\mathcal{S}$ is a set of schemas and C is a set of correspondences generated by matching tools.

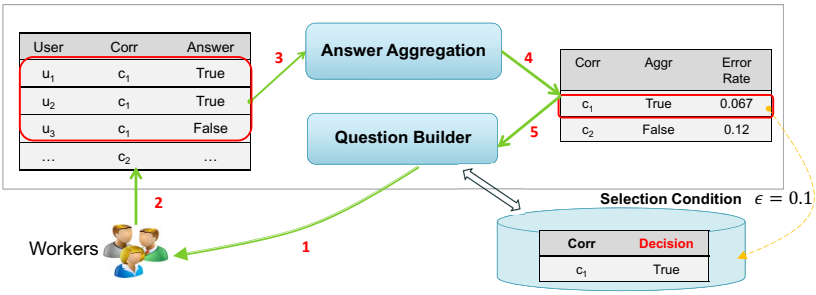

Fig. 1. Architecture of the crowdsourcing framework

For realizing the second second step of validating the correspondences, we propose the framework depicted in Figure 1. The input to our framework is a set of correspondences C. These correspondences are fetched to *Question Builder* component to generate questions presented to crowd workers. A worker's answer is the validation of worker u_i on a particular correspondence $c_j \in C$, denoted as a tuple $\langle u_i, c_j, a \rangle$, where a is the answer of worker u_i on correspondence c_j. Domain values of a are $\{true, false\}$, where $true/false$ indicates c_j is approved/disapproved.

In general, the answers from crowd workers might be incorrect. There are several reasons for this, such as the workers might misunderstand their tasks, they may accidentally make errors, or they simply do not know the answers. To cope with the problem of possibly incorrect answers, we need aggegation mechanismes, realized in the *Answer Aggregation* component. We adopt probabilistic aggregation techniques. We estimate the quality of the aggregated value by comparing the answers from different workers. The aggregated result of a correspondence is a tuple $\langle a^*, e \rangle$, where a^* is the aggregated value, e is the error rate of aggregation. If the error rate e is greater than a pre-defined threshold ϵ, we continue to fetch c into *Question Builder* to ask workers for more answers. Otherwise, we make the decision a^* for the given correspondence. This process is repeated until the halting condition is satisfied. In our framework, the halting condition is that all correspondences are decided.

In our setting, it is reasonable to assume that there is an objective ground truth, i.e., there exists a single definitive matching result that is external to human judgment. However, this truth is hidden and no worker knows it completely. Therefore, we leverage the wisdom of the crowd in order to approximate the hidden ground truth (with the help of our aggregation techniques). However, approximating the ground truth with limited budget raises several challenges: (1) *How to design the questions for effective answers?* (2) *How to make aggregation decision based on the answers from workers?* (3) *How to reduce the number of questions with a given quality requirement?* In the following sections, we will address these challenges.

3 Question Design

In this section, we demonstrate how to design questions using the set of candidate correspondences. Generally, a question is generated with 3 elements: (1) *Object*, (2) *Possible answers* and (3) *Contextual information*. In our system, the object of a question is an attribute correspondence. The possible answers which a worker can provide are either *true* (approve) or *false* (disapprove). The last element is contextual information, which plays a very important role in helping workers answer the question more easily. It provides a meaningful context to make the question more understandable. In our work, we h have used three kinds of contextual information:

- **All alternative targets:** We show a full list of candidate targets generated by matching tools. By examining all possible targets together, workers have can better judge whether the given correspondence is correct or not as opposed to evaluating a single value correspondence. Figure 2(A) gives an example of this design.
- **Transitive closure:** We do not only display all alternatives, but also the transitive closure of correspondences. The goal of displaying the transitive closure is to provide a context that shall help workers to resolve the ambiguity, when otherwise these alternatives are hard to distinguish. For example, in Figure 2(B), workers might not be able to decide which one of two attributes CRM.BirthDate and CRM.Name corresponds to the attribute MDM.BirthName. Thanks to the transitive closure MDM.BirthName → CRM.Name → SRM.BirthName, workers can confidently confirm the correctness of the match between CRM.Name and MDM.BirthName.
- **Transitive violation:** In contrast to transitive closure, this design supports a worker to identify incorrect correspondences. Besides all alternatives, the contextual information contains a circle of correspondences that connects two different attributes of the same schema. For instance, in Figure 2(C), workers might find it difficult to choose the right target among CRM.BirthDate, CRM.Name for MDM.BirthName. The transitive violation CRM.Name → SRM.BirthName → MDM.BirthName → CRM.BirthDate is the evidence that helps worker to reject the match between MDM.BirthName and CRM.BirthDate.

Comparing to the question generating and posting strategy presented in [17], our question design is more general. In our approach, both the pairwise information (i.e., data

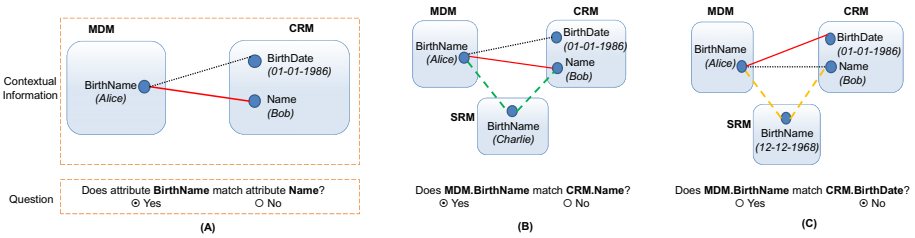

Fig. 2. Question designs with 3 different contextual information: (A) All alternative targets, (B) Transitive closure, (C) Transitive violation

value and all alternatives) and the network-level contextual information (i.e., transitive closure and transitive violation) are displayed to help the workers to answer the question more effectively. To evaluate the effectiveness of the question design, we conducted some experiments in section 6. It turned out that the contextual information proposed as above is critical. Having the contextual information at hand, the workers were able to answer the questions faster and more accurately. Subsequently, the total cost could be substantially reduced since the payment for each task can be decreased [2].

4 Aggregating User Input

In this section we explain our aggregation techniques. After posting questions to crowd workers (as explained in Section 3), for each correspondence $c \in C$, we collect a set of answers π_c (from different workers) in which each element could be $true$(approve) or $false$(disapprove). The goal of aggregation is to obtain the aggregated value a_c as well as estimate the probability that a_c is incorrect. This probability is also called the error rate of the aggregation e_c.

In order to compute the aggregated value a_c and error rate e_c, we first derive the probability of possible aggregations $Pr(X_c)$. In that, X_c is a random variable of aggregated values of c and domain values of X_c is $\{true, false\}$. This value refers to the ground truth, however that is hidden from us, thus we try to estimate this probability with the help of aggregation methods. There are several techniques proposed in the literature to compute this probability such as majority voting [2] and expectation maximization (EM) [8]. While majority voting aggregates each correspondence independently, the EM method aggregates all correspondences simultaneously. More precisely, the input of majority voting is the worker answers π_c for a particular correspondence c, whereas the input of EM is the worker answers $\pi = \bigcup_{c \in C} \pi_c$ for all correspondences.

In this paper, we use EM as the main aggregation method to compute the probability $Pr(X_c)$. The EM method differs from majority voting in considering the quality of workers, which is estimated by comparing the answers of each worker against other workers answers. More precisely, the EM method uses maximum likelihood estimation to infer the aggregated value of each correspondence and measure the quality of that value. The reason behind this choice is that the EM model is quite effective for labeling tasks and robust to noisy workers [23].

After deriving the probability $Pr(X_c)$ for each correspondence $c \in C$, we will compute the aggregation decision $\langle a_c, e_c \rangle = g_\pi(c)$, where a_c is the aggregated value and e_c is the error rate. The aggregation of this decision is formulated as follows:

$$g_\pi(c) = \begin{cases} \langle true, 1 - Pr(X_c = true) \rangle & \text{If } Pr(X_c = true) \geq 0.5 \\ \langle false, 1 - Pr(X_c = false) \rangle & \text{Otherwise} \end{cases} \quad (1)$$

In equation 1, the error rate is the probability of making wrong decision. In order to reduce error rate, we need to reduce the uncertainty of X_c (i.e., entropy value $H(X_c)$). If the entropy $H(X_c)$ is closed to 0, the error rate is closed to 0. For the experiments described in section 6, in order to achieve lower error rate, we need to ask more questions. However, with given requirements of low error rate, the monetary cost is limited and needs to be reduced. In next section, we will leverage the constraints to solve this problem.

5 Leveraging Constraints to Reduce User Efforts

For experiments described in section 6, we found that to achieve lower error rate, more answers are needed. This is, in fact, the trade-off between the cost and the accuracy[26]. The higher curve of Figure 3 depicts empirically a general case of this trade-off.

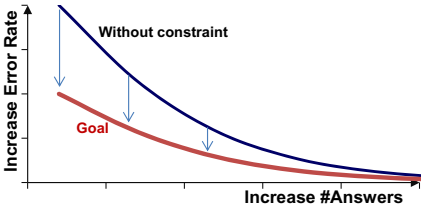

Fig. 3. Optimization goal

We want to go beyond this trade-off by lowering this curve as much as possible. When the curve is lower, with the same error rate, the number of answers is smaller. In other words, with the same number of answers, the error rate is smaller. To achieve this goal, we leverage the network consistency constraints to adjust the error rate with the same number of answers. In this section, we will show how to exploit these constraints.

5.1 Aggregating with Constraints

In section 4, we already formulate the answer aggregation. Now we leverage constraints to adjust the error rate of the aggregation decision. More precisely, we show that by using constraints, it requires fewer answers to obtain aggregated result with the same error rate. In other words, given the same answer set on a certain correspondence, the error rate of aggregation with constraint is lower than the one without constraint. We consider very natural constraints that we assume to hold; in other words we assume that these are hard constraints.

Given the aggregation $g_\pi(c)$ of a correspondence c, we compute the justified aggregation $g_\pi^\gamma(c)$ when taking into account the constraint γ. The aggregation $g_\pi^\gamma(c)$ is obtained similarly to equation 1, except that the probability $Pr(X_c)$ is replaced by the conditional probability $Pr(X_c|\gamma)$ when the constraint γ holds. Formally,

$$g_\pi^\gamma(c) = \begin{cases} \langle true, 1 - Pr(X_c = true|\gamma)\rangle & \text{If } Pr(X_c = true|\gamma) \geq 0.5 \\ \langle false, 1 - Pr(X_c = false|\gamma)\rangle & \text{Otherwise} \end{cases} \quad (2)$$

In the following, we describe how to compute $Pr(X_c|\gamma)$ with 1-1 constraint and circle constraint. Then, we show why the affect of constraints can reduce error rate. We leave the investigation of other types of constraints as an interesting future work.

5.2 Aggregating with 1-1 Constraint

Our approach underlies the intuition illustrated in Figure 4(A), depicting two correspondences c_1 and c_2 with the same source attribute. After receiving the answer set from workers and applying probabilistic model (section 4), we obtained the probability $Pr(X_{c_1} = true) = 0.8$ and $Pr(X_{c_2} = false) = 0.5$. When considering c_2 independently, it is hard to conclude c_2 being approved or disapproved. However, when taking into account c_1 and 1-1 constraint, c_2 tends to be disapproved since c_1 and c_2 cannot be approved simultaneously. Indeed, following probability theory, the conditional probability $Pr(X_{c_2} = false|\gamma_{1-1}) \approx 0.83 > Pr(X_{c_2} = false)$.

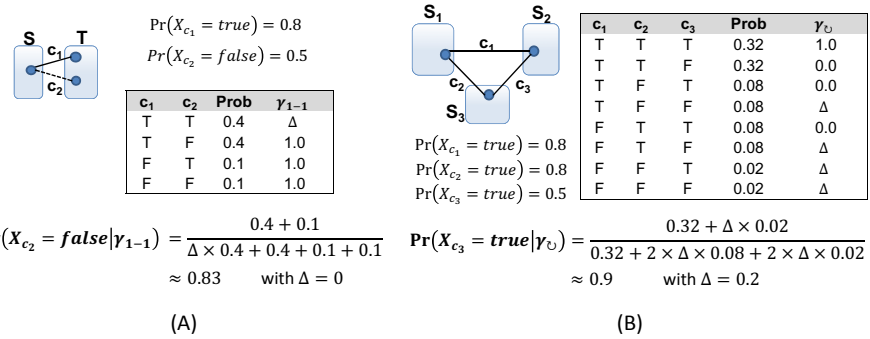

Fig. 4. Compute conditional probability with (A) 1-1 constraint and (B) circle constraint

In what follows, we will formulate 1-1 constraint in terms of probability and then show how to compute the conditional probability $Pr(X_c|\gamma_{1-1})$.

Formulating 1-1 Constraint. Given a matching between two schemas, let us have a set of correspondences $\{c_0, c_1, \ldots, c_k\}$ that share a common source attribute. With respect to 1-1 constraint definition, there is at most only one c_i is approved (i.e., $X_{c_i} = true$). However there are some exceptions where this constraint does not hold. For instance, the attribute *name* might be matched with $firstname$ and $lastname$. But these cases only happen with low probability. In order to capture this observation, we formulate 1-1 constraint as follows:

$$Pr(\gamma_{1-1}|X_{c_0}, X_{c_1}, \ldots, X_{c_k}) = \begin{cases} 1 & \text{If } m \leq 1 \\ \Delta \in [0,1] & \text{If } m > 1 \end{cases} \quad (3)$$

where m is the number of X_{c_i} assigned as *true*. When $\Delta = 0$, there is no constraint exception. In general, Δ is close to 0. The approximated value of Δ can be obtained through statistical model [6].

Computing Conditional Probability. Given the same set of correspondence $\{c_0, c_1, \ldots, c_k\}$ above, let denote p_i as $Pr(X_{c_i} = true)$ for short. Without loss of generality, we consider c_0 be the favourite correspondence whose probability p_0 is obtained from the worker answers. Using the Bayesian theorem and equation 3, the conditional probability of correspondence c_0 with 1-1 constraint γ_{1-1} is computed as:

$$Pr(X_{c_0} = true|\gamma_{1-1}) = \frac{Pr(\gamma_{1-1}|X_{c_0} = true) \times Pr(X_{c_0} = true)}{Pr(\gamma_{1-1})} = \frac{(x + \Delta(1-x)) \times p_0}{y + \Delta(1-y)} \quad (4)$$

$$\text{where} \quad x = \prod_{i=1}^{k}(1-p_i)$$
$$y = \prod_{i=0}^{k}(1-p_i) + \sum_{i=0}^{k}\left[p_i \prod_{j=0, j\neq i}^{k}(1-p_j)\right]$$

x can be interpreted as the probability of the case where all other correspondences except c being disapproved. y can be interpreted as the probability of the case where all correspondences being disapproved or only one of them being disaproved. The precise derivation of equation 4 is put in the Appendix.

5.3 Aggregating with Circle Constraint

Figure 4(B) depicts an example of circle constraint for three correspondences c_1, c_2, c_3. After receiving the answer set from workers and applying probabilistic model (section 4), we obtained the probability $Pr(X_{c_1} = true) = Pr(X_{c_2} = true) = 0.8$ and $Pr(X_{c_3} = true) = 0.5$. When considering c_3 independently, it is hard to conclude c_3 being *true* or $false$. However, when taking into account c_1, c_2 under the 1-1 constraint, c_3 tends to be *true* since the circle created by c_1, c_2, c_3 shows an interoperability. Therefore, following probability theory, the conditional probability $Pr(X_{c_3} = true|\gamma_{1-1}) \approx 0.9 > Pr(X_{c_3} = true)$.

In the following we will formulate circle constraint in terms of probability and then show how to compute the conditional probability $Pr(X_c|\gamma_{\cup})$.

Formulating Circle Constraint. Following the notion of cyclic mappings in [6], we formulate the conditional probability of a circle as follows:

$$Pr(\gamma_{\cup}|X_{c_0}, X_{c_1}, \ldots, X_{c_k}) = \begin{cases} 1 & \text{If } m = k+1 \\ 0 & \text{If } m = k \\ \Delta & \text{If } m < k \end{cases} \quad (5)$$

Where m is the number of X_{c_i} assigned as *true* and Δ is the probability of compensating errors along the circle (i.e., two or more incorrect assignment resulting in a correct reformation).

Computing Conditional Probability. Given a closed circle along $c_0, c_1, \ldots, c_k$, let denote the constraint on this circle as $\gamma_\cup$ and p_i as $Pr(X_{c_i} = true)$ for short. Without loss of generality, we consider c_0 to be the favorite correspondence whose probability p_0 is obtained by the answers of workers in the crowdsourcing process. Following the Bayesian theorem and equation 5, the conditional probability of correspondence c_0 with circle constraint is computed as:

$$Pr(X_{c_0} = true|\gamma_\cup) = \frac{Pr(\gamma_\cup|X_{c_0} = true) \times Pr(X_{c_0} = true)}{Pr(\gamma_\cup)} = \frac{(\prod_{i=1}^{k}(p_i) + \Delta(1-x)) \times p_o}{\prod_{i=0}^{k}(p_i) + \Delta(1-y)} \quad (6)$$

$$\textit{where} \qquad x = \prod_{i=1}^{k}(p_i) + \sum_{i=1}^{k}[(1-p_i)\prod_{j=1,j\neq i}^{k} p_j]$$
$$y = \prod_{i=0}^{k}(p_i) + \sum_{i=0}^{k}[(1-p_i)\prod_{j=0,j\neq i}^{k} p_j]$$

x can be interpreted as the probability of the case where only one correspondence among $c_1, \ldots, c_k$ except c_0 is disapproved. y can be interpreted as the probability of the case where only one correspondence among $c_0, c_1, \ldots, c_k$ is disapproved. The detail derivation of equation 6 is put in the Appendix.

5.4 Aggregating with Multiple Constraints

In general settings, we could have a finite set of constraints $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$. Let denote the aggregation with a constraint $\gamma_i \in \Gamma$ is $g_\pi^{\gamma_i}(c) = \langle a_c^i, e_c^i \rangle$, whereas the aggregation without any constraint is simply written as $g_\pi(c) = \langle a_c, e_c \rangle$. Since the constraints are different, not only could the aggregated value a_c^i be different ($a_c^i \neq a_c^j$) but also the error rate e_c^i could be different ($e_c^i \neq e_c^j$). In order to reach a single decision, the challenge then becomes how to define the multiple-constraint aggregation $g_\pi^\Gamma(c)$ as a combination of single-constraint aggregations $g_\pi^{\gamma_i}(c)$.

Since the role of constraints is to support reducing the error rate and the aggregation $g_\pi(c)$ is the base decision, we compute the multiple-constraint aggregation $g_\pi^\Gamma(c) = \langle a_c, e_c^\Gamma \rangle$, where $e^\Gamma = min(\{e_c^i | a_c^i = a_c\} \cup e_c)$. Therefore, the error rate of final aggregated value is reduced by harnessing constraints. For the experiments in real datasets described in the next section, we will show that this aggregation reduces a half of worker efforts while preserving the quality of aggregated results.

6 Experiments

The main goal of the following evaluation is to analyze the use of crowdsourcing techniques for schema matching network. To verify the effectiveness of our approach, three experiments are performed: (i) effects of contextual information on reducing question ambiguity, (ii) relationship between the error rate and the matching accuracy, and (iii) effects of the constraints on worker effort. We proceed to report the results on the real datasets using both real workers and simulated workers.

6.1 Experimental Settings

Datasets. We have used 3 real-world datasets: Google Fusion Tables, UniversityApp-Form, and WebForm. They are publicly available on our website [3]. In the experiments, the topology of schema matching network is a complete graph (i.e. all graph nodes are interconnected with all other nodes). In that, the candidate correspondences are generated by COMA [10] matcher.

Worker Simulation. In our simulation, we assume that the ground truth is known in advance (i.e. the ground truth is known for the experimenter, but not for the (simulated) crowd worker). Each simulated worker is associated with a pre-defined reliability r that is the probability of his answer being correct against the ground truth.

6.2 Effects of Contextual Information

In this experiment, we select 25 correct correspondences (i.e., exist in ground truth) and 25 incorrect correspondences (i.e., not exists in ground truth). For each correspondence, we ask 30 workers (Bachelor students) with three different contextual information: (a) all alternatives, (b) transitive closure, (c) transitive violation. Then, we collect the worker answers for each correspondence.

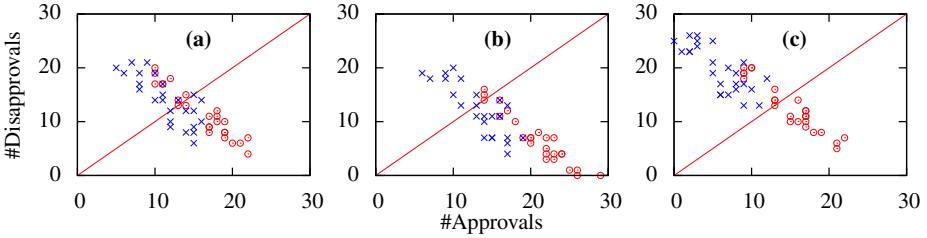

Fig. 5. Effects of contextual information. (a) all alternatives, (b) transitive closure, (c) transitive violation

Figure 5 presents the result of this experiment. The worker answers of each case are presented by a collection of 'x' and 'o' points in the plots. In that, 'o' points indicate correspondences that exist in ground truth, whereas 'x' points indicate correspondences that do not exist in ground truth. For a specific point, X-value and Y-value are the number of workers approving and disapproving the associated correspondences, respectively. Therefore, we expect that the 'o' points are placed at the right-bottom of the coordinate plane, while the 'x' points stay at the left-top of the coordinate plane.

Comparing Figure 5(b) with Figure 5(a) , the 'o' points tend to move down to the bottom-right of the baseline (# 'approve' answers increases and # 'disapprove' answers decreases). Whereas, the movement of the 'x' points is not intensive. This can be interpreted that presenting the transitive closure context help workers to give feedback more exactly but also make them misjudge the incorrect correspondences.

[3] `http://lsirwww.epfl.ch/schema_matching`

In order to study the effects of transitive violation, we compare Figure 5(c) with Figure 5(a). Intuitively, the 'x' points move distinctly toward the top-left of the baseline, while the position of 'o' points keeps stable. This observation shows that transitive violations help workers identify the incorrect correspondences, in contrast to the effect of transitive satisfactions mentioned above.

Since in real settings the ground truth is not known before-hand, we cannot choose appropriate design type for each question. Following the principle of maximum entropy, in order not to favour any of the design types, we design each question in type (b) and (c) with probability of 0.5. In case the given correspondence does not involve in any transitive satisfaction and violation, we design its question in type (a).

6.3 Relationship between Error Rate and Matching Accuracy

In order to assess the matching accuracy, we borrow the *precision* metric from information retrieval, which is the ratio of correspondences existing in ground truth among all correspondences whose aggregated value is *true*. However, the ground truth is not known in general. Therefore, we use an indirect metric—error rate—to estimate the matching quality. We expect that the lower error rate, the higher quality of matching results.

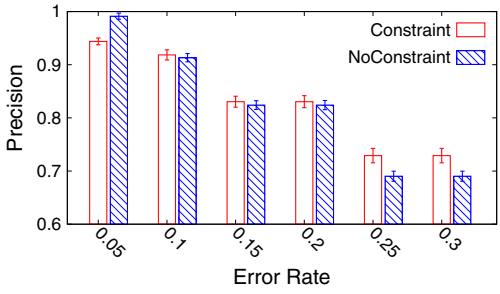

Fig. 6. Relationship between error rate and precision

The following empirical results aim to validate this hypothesis. We conduct the experiment with a population of 100 simulated workers and their reliability scores are generated according to normal distribution $\mathcal{N}(0.7, 0.04)$. Figure 6 depicts the relationship of the error rate and precision. In that, we vary error threshold ϵ from 0.05 to 0.3, meaning that the questions are posted to workers until the error rate of aggregated value is less than the given threshold ϵ. The precision is plotted as a function of ϵ. We aggregate the worker answers by two strategies: without constraint and with constraint. Here we consider both 1-1 constraint and circle constraint as hard constraints, thus $\Delta = 0$.

The key observation is that when the error rate is decreased, the precision approaches to 1. Reversely, when the error rate is increased, the precision is reduced but greater than $1 - \epsilon$. Another interesting finding is that when the error rate is decreased, the value distribution of precision in case of with and without constraint is identical. This indicates our method of updating the error rate is relevant.

In summary, the error rate is a good indicator of the quality of aggregated results. In terms of precision, the quality value is always around $1 - \epsilon$. In other words, the error threshold ϵ can be used to control the real matching quality.

6.4 Effects of the Constraints

In this experiment set, we will study the effects of constraints on the expected cost in real datasets. In Section 5, we already seen the benefit of using constraints in reducing error rate. Therefore, with given requirement of low error, the constraints help to reduce the number of questions (i.e., the expected cost) that need to ask workers. More precisely, given an error threshold ($\epsilon = 0.15, 0.1, 0.05$), we iteratively post questions to workers and aggregate the worker answers until the error rate is less than ϵ. We use simulated workers with reliability r varying from 0.6 to 0.8. Similar to the above experiment, we set $\Delta = 0$. The results are presented in Figure 7.

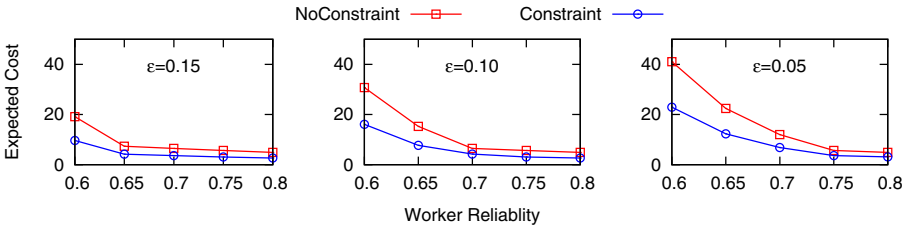

Fig. 7. User Efforts: effects of constraints

A significant observation in the results is that for all values of error threshold and worker reliability, the expected cost of the aggregation with constraints is definitely smaller (approximately a half) than the case without constraints. For example, with worker reliability is $r = 0.6$ and error threshold $\epsilon = 0.1$, the expected number of questions is reduced from 31 (without constraints) to 16 (with constraints). This concludes the fact that the constraints help to reduce the error rate, and subsequently reduce the expected cost.

Another key finding in Figure 7 is that, for both cases (using vs. not using constraints in the aggregation), the expected cost increases significantly as the value for error threshold ϵ decreases. For example, it requires about 20 questions (without constraints) or 10 questions (with constraints) to satisfy error threshold $\epsilon = 0.15$. Whereas, it takes about 40 questions (without constraints) or 20 questions (with constraints) to satisfy error threshold $\epsilon = 0.05$. This result supports the fact that to reduce error rate, we need to ask more questions.

7 Related Work

We now review salient work in schema matching and crowdsourcing areas that are related to our research.

Schema Matching. Database schema matching is an active research field. The developments of this area have been summarized in two surveys [4, 22]. Existing works on schema matching focused mainly on improving quality parameters of matchers, such as precision or recall of the generated matchings. Recently, however, ones started to realize that the extent to what precision and recall can be improved may be limited for general-purpose matching algorithms. Instead of designing new algorithms, there has been a shift towards matching combination and tuning methods. These works include YAM [11], systematic matching ensemble selection [12] or automatic tuning of the matcher parameters [15].

While there is a large body of works on schema matching, the post-matching reconciliation process (that is central to our work) has received little attention in the literature. Recently, there are some works [14, 17, 21] using pay-as-you-go integration method that establishes the initial matching and then incrementally improves matching quality. While the systems in [14, 21] rely on one user only, the framework in [17] relies on multiple users.

Schema Matching Network. The idea of exploiting the presence of a large set of schemas to improve the matchings has been studied before. Holistic matching [25] attempted to exploit statistical co-occurrences of attributes in different schemas and use them to derive complex correspondence. Whereas, corpus-based matching [16] attempted to use a 'corpus' of schemas to augment the evidences that improve exist matchings and exploit constraints between attributes by applying statistical techniques. Network level constraints, in particular the circle constraints, were originally considered in [1, 6] in which they study the establishment of semantic interoperability in a large-scale P2P network. In this paper, we study contextual information and integrity constraints (e.g., 1-1 and circle constraints) on top of the schema matching network.

Crowdsourcing. In recent years, crowdsourcing has become a promising methodology to overcome human-intensive computational tasks. Its benefits vary from unlimited labour resources of user community to cost-effective business models. The book [2] summarized problems and challenges in crowdsourcing as well as promising research directions for the future. A wide range of crowdsourcing platforms, which allows users to work together in a large-scale online community, have been developed such as Amazon Mechanical Turk and CloudCrowd.

On top of these platforms, there are also many crowdsourcing applications that have been built for specific domains. For example, in [26], the crowdsourcing is employed to validate the search results of automated image search on mobile devices. In [3], the authors leveraged the user CAPTCHAs inputs in web forms to recognize difficult words that cannot solved precisely by optical character recognition (OCR) programs.

Regarding the utilization of constraints, there are some previous works such as [5, 27]. In [27], the constraints were used to define the tasks for collaborative planning systems whereas in [5], the constraints were used to check worker quality by quantifying the consistency of worker answers. In our work, the constraints are used to adjust the error rate for reducing worker efforts.

8 Conclusions and Future Work

Using shared datasets in meaningful ways frequently requires interconnecting several sources, i.e., one needs to construct the attribute correspondences between the concerned schemas. The schema matching problem has, in this setting, a completely new aspect: there are more than two schemas to be matched and the schemas participate in a larger *schema matching network*. This network can provide contextual information to the particular matching tasks.

We have presented a crowdsourcing platform that is able to support schema matching tasks. The platform takes the candidate correspondences that are generated by pairwise schema matching and generates questions for crowd workers. The structure of the matching network can be exploited in many ways. First, as this is a contextual information about the particular matching problem, it can be used to generate questions that guide the crowd workers and help them to answer the questions more accurately. Second, natural constraints about the attribute correspondences at the level of the network enable to reduce the necessary efforts, as we demonstrated this through our experiments.

Our work opens up several future research directions. First, one can extend our notion of schema matching network and consider representing more general integrity constraints (e.g., functional dependencies or domain-specific constraints). Second, one can devise more applications which could be transformed into the schema matching network. While our work focuses on schema matching, our techniques, especially the constraint-based aggregation method, can be applied to other tasks such as entity resolution, business process matching, or Web service discovery.

Acknowledgment. This research has received funding from the NisB project - European Union's Seventh Framework Programme (grant agreement number 256955) and the PlanetData project - Network of Excellence (grant agreement number 257641).

References

[1] Aberer, K., Cudré-Mauroux, P., Hauswirth, M.: Start making sense: The Chatty Web approach for global semantic agreements. JWS, 89–114 (2003)
[2] von Ahn, L.: Human computation. In: DAC, pp. 418–419 (2009)
[3] von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: Recaptcha: Human-based character recognition via web security measures. Science, 1465–1468 (2008)
[4] Bernstein, P.A., Madhavan, J., Rahm, E.: Generic Schema Matching, Ten Years Later. PVLDB, 695–701 (2011)
[5] Chen, K.T., Wu, C.C., Chang, Y.C., Lei, C.L.: A crowdsourceable qoe evaluation framework for multimedia content. In: MM, pp. 491–500 (2009)
[6] Cudré-Mauroux, P., Aberer, K., Feher, A.: Probabilistic message passing in peer data management systems. In: ICDE, p. 41 (2006)
[7] Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R., Yu, C.: Finding related tables. In: SIGMOD, pp. 817–828 (2012)
[8] Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the EM algorithm. J. R. Stat. Soc., 20–28 (1979)
[9] Di Lorenzo, G., Hacid, H., Paik, H.: y., Benatallah, B.: Data integration in mashups. In: SIGMOD, pp. 59–66 (2009)

[10] Do, H., Rahm, E.: COMA: a system for flexible combination of schema matching approaches. In: PVLDB, pp. 610–621 (2002)
[11] Duchateau, F., Coletta, R., Bellahsene, Z., Miller, R.J.: (Not) yet another matcher. In: CIKM. pp. 1537–1540 (2009)
[12] Gal, A., Sagi, T.: Tuning the ensemble selection process of schema matchers. JIS, 845–859 (2010)
[13] Gonzalez, H., Halevy, A.Y., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W., Goldberg-Kidon, J.: Google fusion tables: web-centered data management and collaboration. In: SIGMOD, pp. 1061–1066 (2010)
[14] Jeffery, S.R., Franklin, M.J., Halevy, A.Y.: Pay-as-you-go user feedback for dataspace systems. In: SIGMOD, pp. 847–860 (2008)
[15] Lee, Y., Sayyadian, M., Doan, A., Rosenthal, A.S.: eTuner: tuning schema matching software using synthetic scenarios. JVLDB 16, 97–122 (2007)
[16] Madhavan, J., Bernstein, P.A., Doan, A., Halevy, A.: Corpus-based schema matching. In: ICDE, pp. 57–68 (2005)
[17] McCann, R., Shen, W.: Matching schemas in online communities: A web 2.0 approach. In: ICDE, pp. 110–119 (2008)
[18] Nguyen, H., Fuxman, A., Paparizos, S., Freire, J., Agrawal, R.: Synthesizing products for online catalogs. PVLDB, 409–418 (2011)
[19] Parameswaran, A.G., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., Widom, J.: Crowdscreen: algorithms for filtering data with humans. In: SIGMOD, pp. 361–372 (2012)
[20] Peukert, E., Eberius, J., Rahm, E.: AMC - A framework for modelling and comparing matching systems as matching processes. In: ICDE, pp. 1304–1307 (2011)
[21] Qi, Y., Candan, K.S., Sapino, M.L.: Ficsr: feedback-based inconsistency resolution and query processing on misaligned data sources. In: SIGMOD, pp. 151–162 (2007)
[22] Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. JVLDB, 334–350 (2001)
[23] Sheng, V.S., Provost, F.: Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. In: SIGKDD, pp. 614–622 (2008)
[24] Smith, K.P., Morse, M., Mork, P., Li, M., Rosenthal, A., Allen, D., Seligman, L., Wolf, C.: The role of schema matching in large enterprises. In: CIDR (2009)
[25] Su, W., Wang, J., Lochovsky, F.: Holistic schema matching for web query interfaces. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 77–94. Springer, Heidelberg (2006)
[26] Yan, T., Kumar, V.: CrowdSearch: exploiting crowds for accurate real-time image search on mobile phones. In: MobiSys, pp. 77–90 (2010)
[27] Zhang, H., Law, E., Miller, R., Gajos, K., Parkes, D., Horvitz, E.: Human computation tasks with global constraints. In: CHI, pp. 217–226 (2012)

Appendix

Compute Conditional Probability $Pr(X_{c_0}|\gamma_{1-1})$**:** According to Bayes theorem, $Pr(X_{c_0}|\gamma_{1-1}) = \frac{Pr(\gamma_{1-1}|X_{c_0}) \times Pr(X_{c_0})}{Pr(\gamma_{1-1})}$. Now we need to compute $Pr(\gamma_{1-1})$ and $Pr(\gamma_{1-1}|X_{c_0})$. Let denote $p_i = Pr(X_{c_i} = true)$, for short. In order to compute $Pr(\gamma_{1-1})$, we do following steps: (1) express $Pr(\gamma_{1-1})$ as the sum from the full joint of $\gamma_{1-1}, c_0, c_1, \ldots, c_k$, (2) express the joint as a product of conditionals. Formally, we have:

$$
\begin{aligned}
Pr(\gamma_{1-1}) &= \textstyle\sum_{c_0,c_1,\ldots,c_k} Pr(\gamma_{1-1}, X_{c_0}, X_{c_1}, \ldots, X_{c_k}) \\
&= \textstyle\sum Pr(\gamma_{1-1}|X_{c_0}, X_{c_1}, \ldots, X_{c_k}) \times Pr(X_{c_0}, X_{c_1}, \ldots, Xc_k) \\
&= 1 \times Pr(X_{c_0}, X_{c_1}, \ldots, X_{c_k}|m(X_{c_0}, X_{c_1}, \ldots, X_{c_k}) \leq 1) \\
&+ \Delta \times Pr(X_{c_0}, X_{c_1}, \ldots, X_{c_k}|m(X_{c_0}, X_{c_1}, \ldots, X_{c_k}) > 1) \\
&= y + \Delta \times (1 - y)
\end{aligned}
$$

where m is function counting the number of X_{c_i} assigned as *true*
$y = \prod_{i=0}^{n} (1 - p_i) + \sum_{i=0}^{n} [p_i \prod_{j=0, j\neq i}^{n} (1 - p_j)]$

Similar to computing $Pr(\gamma_{1-1})$, we also express $Pr(\gamma_{1-1}|X_{c_0})$ as the sum from the full joint of γ_{1-1}, $c_1, \ldots, c_k$ and then express the joint as a product of conditionals. After these steps, we have $Pr(\gamma_{1-1}|X_{c_0} = true) = x + \Delta \times (1 - x)$, where $x = \prod_{i=1}^{k} (1 - p_i)$. After having $Pr(\gamma_{1-1})$ and $Pr(\gamma_{1-1}|X_{c_0})$, we can compute $Pr(X_{c_0}|\gamma_{1-1})$ as in equation 4.

Compute Conditional Probability $Pr(X_{c_0}|\gamma_{\cup})$**:** According to Bayes theorem, $Pr(X_{c_0}|\gamma_{\cup}) = \frac{Pr(\gamma_{\cup}|X_{c_0}) \times Pr(X_{c_0})}{Pr(\gamma_{\cup})}$. In order to compute $Pr(\gamma_{\cup}|X_{c_0})$ and $Pr(\gamma_{\cup})$, we also express $Pr(\gamma_{\cup}|X_{c_0})$ as the sum from the full joint of γ_{1-1}, $c_0, c_1, \ldots, c_k$ and then express the joint as a product of conditionals. After some transformations, we can obtain equation 6.