Reducing Uncertainty of Schema Matching via Crowdsourcing with Accuracy Rates

Chen Jason Zhang, Lei Chen, *Member, IEEE,* H. V. Jagadish, *Member, IEEE,* Mengchen Zhang, and Yongxin Tong, *Member, IEEE,*

Abstract—Schema matching is a central challenge for data integration systems. Inspired by the popularity and the success of crowdsourcing platforms, we explore the use of crowdsourcing to reduce the uncertainty of schema matching. Since crowdsourcing platforms are most effective for simple questions, we assume that each *Correspondence Correctness Question* (CCQ) asks the crowd to decide whether a given correspondence should exist in the correct matching. Furthermore, members of a crowd may sometimes return incorrect answers with different probabilities. Accuracy rates of individual crowd workers are probabilities of returning correct answers which can be attributes of CCQs as well as evaluations of individual workers. We prove that uncertainty reduction equals to entropy of answers minus entropy of crowds and show how to obtain lower and upper bounds for it. We propose frameworks and efficient algorithms to dynamically manage the CCQs to maximize the uncertainty reduction within a limited budget of questions. We develop two novel approaches, namely "Single CCQ" and "Multiple CCQ", which *adaptively* select, publish and manage questions. We verify the value of our solutions with simulation and real implementation.

Index Terms—crowdsourcing, uncertainty reduction, schema matching

✦

1 INTRODUCTION

1.1 Background and Motivation

Schema matching refers to finding correspondences between elements of two given schemata, which is a critical issue for many database applications such as data integration, data warehousing, and electronic commerce [35]. Figure 1 illustrates a running example of the schema matching problem: given two relational schemata A and B describing faculty information, we aim to determine the correspondences (indicated by dotted lines), which identify attributes representing the same concepts in the two. There has been significant work in developing automated algorithms for schema matching (please refer to [35] [40] [2] [1] for comprehensive surveys). Most approaches use linguistic, structural and instance-based information. In general, it is still very difficult to tackle schema matching completely with an algorithmic approach: some ambiguity remains. This ambiguity is unlikely to be removed because it is believed that typically "the syntactic representation of schemata and data do not completely convey the semantics of different databases" [27].

Given this inherent ambiguity, many schema matching tools will produce not just one matching, but rather a whole set of

- *Chen Jason Zhang is with School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan, Shandong, China and Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Kowloon, Hong Kong, SAR China E-mail: czhangad@cse.ust.hk*
- *Lei Chen and Mengchen Zhang are with Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Kowloon, Hong Kong, SAR China E-mail: leichen@cse.ust.hk, mzhangag@connect.ust.hk*
- *H. V. Jagadish is with Department of Electrical Engineering and Computer Science, University of Michigan, USA E-mail: jag@eecs.umich.edu*
- *Yongxin Tong is with State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing, China. E-mail: yxtong@buaa.edu.cn*

Fig. 1. Example of Schema Matching Problem

TABLE 1
Uncertain Schema Matching

Possible Matchings	probability
m_1={ <(Professor)Name,[first name, last name] >, <Position, Position>, <Gender,Sex>, <(Department) Name, Department>}	.45
m_2={ <(Professor)Name,[first name, last name] >, <Gender, Sex>, <(Department) Name, Department>}	.3
m_3={ <(Department)Name, first name>, <Position, Position> <Gender,Sex >}	.25

Correspondence	probability
c_1=<(Professor)Name,[first name, last name] >	.75
c_2=<Position, Position>	.7
c_3=<Gender,Sex >	1
c_4=<(Department) Name, Department>	.75
c_5=<(Department)Name,first name>	.25

possible matchings. In fact, there is even a stream of work dealing with models of possible matchings, beginning with [8]. The matching tool can produce a result similar to the upper part of Table 1, with one matching per row, associated with a probability that it is the correct matching.

Given a set of possible matchings, one can create an integrated database that has uncertain data, and work with this using any of several systems that support *probabilistic query processing* over uncertain data, such as [16] [6]. However, preserving the uncertainty complicates query processing and increases storage cost. So we would prefer to make choices earlier, if possible, and eliminate (or reduce) the uncertainty to be propagated. It has been suggested [33] that **human insights are extremely conducive for reducing the uncertainty of schema matching**, so the correct matching can be manually chosen by the user from among the possible matchings offered by the system. In a traditional back-end

database environment, where the human 'user' is a DBA, setting up a new integrated database, such a system can work well.

However, in today's world, with end-users performing increasingly sophisticated data accesses, we have to support users who are interested, say, in combining data from two different web sources, and hence require an 'ad hoc' schema matching. Such users may not be experts, and will typically have little knowledge of either source schema. They may not even know what a schema is. They are also likely to have little patience with a system that asks them to make difficult choices, rather than just giving them the desired answer. In other words, users may not themselves be a suitable source of human insight to resolve uncertainty in schema matching.

Fortunately, we have crowdsourcing technology as a promising option today. Many recent works, such as [17], [18], [9] and [29], have suggested leveraging the crowd to improve schema matching. Platforms such as Amazon Mechanical Turk provide convenient access to crowds. The data concerning an explicit problem can be queried by publishing questions, named Human Intelligent Tasks (a.k.a HITs). The work-flow of publishing HITs can be automated with available APIs (e.g. REST APIs) [10]. To the extent that our end-user is not an expert, the opinion of a crowd of other non-experts is likely to be better than that of our end-user.

1.2 Problem Formulation and Contributions

It is well-known that crowdsourcing works best when tasks can be broken down into very simple pieces. An entire schema matching task may be too large a grain for a crowd – each individual may have small quibbles with a proposed matching, so that a simple binary question on the correctness of matchings may get mostly negative answers, with each user declaring it less than perfect. On the other hand, asking open-ended questions is not recommended for a crowd, because it may be difficult to pull together a schema matching from multiple suggestions. We address this challenge by posing to the crowd questions regarding individual correspondences for pairs of attributes, one from each schema being matched. This much simpler question, in most circumstances, can be answered with a simple yes or no. Of course, this requires that we build the machinery to translate between individual attribute correspondences and possible matchings. Fortunately, this has been done before, in [8], and is quite simple: since schema match options are all mutually exclusive, we can determine the probability of each correspondence by simply adding up the probabilities of matchings in which the correspondence holds.

Our problem then is to choose wisely the correspondences to ask the crowd to obtain the highest certainty of correct schema matching at the lowest cost. For schema matching certainty, we choose entropy as our measure – we are building our system on top of a basic schema-matching tool, which can estimate probabilities for schema matches it produces. When the tool obtains a good match, it can associate a high probability. When there is ambiguity or confusion, this translates into multiple lower probability matches, with associated uncertainty and hence higher entropy.

Our first algorithm, called Single CCQ (CCQ is short for Correspondence Correctness Question), determines the single most valuable correspondence query to ask the crowd, given a set of possible schema matchings and associated correspondences, all with probabilities.

Intuitively, one may try a simple greedy approach, choosing the query that reduces entropy the most. However, there are three issues to consider. First, the correspondences are not all independent, since they are related through candidate matchings. So it is not obvious that a greedy solution is optimal. Second, even finding the query that decreases entropy the most can be computationally expensive. Third, we cannot assume that every person in the crowd answers every question correctly – we have to allow for wrong answers too. We address all three challenges below.

Usually, we are willing to ask the crowd about more than one correspondence, even if not all of them. We could simply run Single CCQ multiple times, each time greedily resolving uncertainty in the most valuable correspondence. However, we can do better. For this purpose, we develop Multiple CCQ, an extension of Single CCQ, that maintains k most useful questions to ask the crowd, and dynamically updates questions according to newly received answers.

In a previous conference paper [44], we addressed this problem assuming crowds to be always correct. In this paper we consider more realistic situations: (1) Each CCQ has a probability to be answered correctly depending on the hardness of CCQ; (2) Each crowd worker has a probability to answer a CCQ correctly, which shows the trustworthiness of the worker. Therefore [44] can be viewed as a special case of our paper (probabilities all equal to 1). Combining above two situations together, we could compute the probabilities of CCQs to be answered correctly, and publish k CCQs chosen by our model to crowds with accuracy rates.

To summarize, we have made following contributions,

1. In Section 3.1 and Section 4.1, we propose an entropy-based model to formulate the uncertainty reduction caused by a single CCQ and multiple CCQs, respectively.

2. For the Single CCQ approach, we propose an explicit framework to choose a CCQ, and derive an efficient algorithm in Section 3. We introduce an index structure and pruning technique for efficiently finding the Single CCQ.

3. In Section 3.3 and 4.3, we prove for both Single CCQ approach and Multiple CCQ approach that uncertainty reduction equals to entropy of answer minus entropy of crowds. In Section 3.3 we give the property of uncertainty reduction for Single CCQ approach. In Section 4.4 we obtain optimal upper and lower bounds for Multiple CCQ approach.

4. For the Multiple CCQ approach, we prove its NP-hardness in Section 4.5, and propose an efficient $(1 + \epsilon)$ approximation algorithm, with effective pruning techniques in Section 4.6.

5. Section 5 reports and discusses the experimental study on both simulation and real implementation. We review and compare our solutions with related work in Section 6. In Section 7, we conclude the paper and discuss future work.

2 PROBLEM STATEMENT

In this section, we give definitions related to the problem that we are working on in this paper.

Definition 1 (Correspondence)**.** *Let S and T be two given schemata. A correspondence c is a pair (A_s, A_t), where A_s and A_t are two subsets of attributes from S and T respectively.*

Remark: Here we consider correspondences between subsets of S and T, which means c could be not only 1:1 matching, but also n:m matching. For example in Table 1, c_1 is a 2:1 matching.

Definition 2 (Possible Matching)**.** *Let S and T be two given schemata. Possible matching $m_i = \{c_1, c_2, \ldots, c_{|m_i|}\}$ is a set of correspondences between S and T which satisfies that no attribute participate in more than one correspondence.*

Remark: For example, in Table 1, m_1, m_2 and m_3 are three possible matchings. Note that not every set of correspondences is a possible matching. In practice, possible matchings are generated by schema matching tools with probabilities to be correct.

Definition 3 (Result Set). *For two given schemata S and T, let the result set R be the set of possible matchings generated by some semi-automatic tool of schema matching, together with a probability assignment function $\mathbb{P} : R \rightarrow [0, 1]$. Each matching $m_i \in R$ has the probability $\mathbb{P}(m_i)$ to be correct, and we have $\sum_{m_i \in R} \mathbb{P}(m_i) = 1$*

Remark: In the example of Table 1, the set $\{m_1, m_2, m_3\}$ is result set. In practice, schema matching tools may use some threshold to eliminate possible matchings with very low probability, and return only a few higher probability candidates. If such thresholding is performed, we ignore the low probability matchings that are already pruned, and set their probability to zero. We also mention that [29] discussed another way to establish the probability of each matching.

Definition 4 (Correspondence Set). *Let R be the result set for two given schemata S and T, the correspondence set C is the set of all correspondences contained by possible matchings in R, i.e. $C = \bigcup_{m_i \in R} m_i$*

Remark: Note that a correspondence can appear in more than one possible matching, so for any correspondence $c \in C$, let $\mathbb{P}(c)$ be the probability of c being in the correct matching, then

$$\mathbb{P}(c) = \sum_{\substack{m_i \in R \\ c \in m_i}} \mathbb{P}(m_i) \qquad (1)$$

As a simple extension, for a set of correspondences $U \subseteq C$, let $\mathbb{P}(U)$ be the probability that all correspondences of U are in the correct matching, then

$$\mathbb{P}(U) = \sum_{\substack{m_i \in R \\ U \subseteq m_i}} \mathbb{P}(m_i) \qquad (2)$$

For example in Table 1, $\{c_1, c_2, c_3, c_4, c_5\}$ is correspondence set. Since c_1 is in m_1 and m_2, $\mathbb{P}(c_1) = 0.75$.

Definition 5 (Uncertainty of Schema Matching). *For two given schemata S and T, given result set R and probability assignment function $\mathbb{P}()$, we measure the uncertainty of R with Shannon entropy $H(R) = -\sum_{m_i \in R} \mathbb{P}(m_i) \log \mathbb{P}(m_i)$*

Remark: Shannon entropy has been widely used in information theory and many other fields since 1948 [39] to measure the uncertainty, disorder or unpredictability of a system. Another way to measure uncertainty is to use variance or covariance matrix. For some special cases like Bernoulli distribution, Shannon entropy has maximal value or minimal value when its variance is maximal or minimal. A major reason for utilizing Shannon entropy as is its non-parametric nature. The probability distribution of possible matchings is very dynamic, depending not only on the given schemata, but also on the schema matching tools. Entropy does not require any assumptions about the distribution of variables. Besides, entropy permits non-linear models, which is important for categorical variables [21], such as possible matchings.

Definition 6 (Crowd's Accuracy). *Given a crowd worker W, the crowd's accuracy (or accuracy for short), denoted by $P_W \in [0.5, 1]$, is the probability that W correctly answers each HIT.*

Remark: While some papers assume that crowdsourced answers are 100% accurate, we adopt a more general error model, which requires only that the answer returned by each crowd worker is always correct with a probability no lower than $1/2$. This is a classical crowdsourcing model widely used by a stream

TABLE 2
Meanings of Symbols Used

Notation	Description
c_i, C, Q_{c_i}	correspondence, correspondence set, CCQ w.r.t c_i
$m_i, \|m_i\|$	a possible matching, number of elements in m_i
$\mathbb{P}(c_i)$	probability of c_i being in the correct matching
$\mathbb{P}(m_i)$	probability that m_i is the correct matching
A or A_{c_i}	the answer or the answer for correspondence c_i
R, W, P_W	result set, a crowd worker, crowd's accuracy
$H(R), H(W), H(A)$	entropy of result set, crowd, answer
ΔH_{Q_c}	uncertainty reduction by publishing Q_c
D_A	the domain of answers of k CCQs
$H(D_A)$	joint entropy of k answers
D_U	the domain of k correspondences

of works [5], [15], [23], [30]. Crowd workers may have different accuracies for different domains. The accuracy for a domain can be easily estimated with a set of sample HITs in which ground truth is known. Before we ask a CCQ, we could assume that this CCQ will be answered by a crowd with accuracy P_W. Since we do not know who will answer this CCQ, P_W is likely to represent the hardness of CCQ as an attribute of the correspondence.

Definition 7 (Entropy of Crowd). *Given a crowd worker W and its accuracy P_W , the entropy of W is defined by*

$$H(W) = -P_W \log P_W - (1 - P_W) \log (1 - P_W) \qquad (3)$$

Given the crowd's accuracy, $H(W)$ is a positive constant measuring the randomness of the crowd's behaviour.

Definition 8 (Correspondence Correctness Question). *A Correspondence Correctness Question (CCQ) asks whether a correspondence is correct. The CCQ w.r.t a correspondence c is denoted as Q_c, where $c \in C$.*

Remark: A running example where we calculate entropy and conditional probability after we have an answer of CCQ is given in Section 3.3.1.

Definition 9 (Entropy of Answer). *Given result set R, probability assignment function $\mathbb{P}()$, and crowd's accuracy $P_W \in [0.5, 1]$, the entropy of answer A corresponding to question Q_c is defined by*

$$H(A) = -\mathbb{P}(A = Y) \log \mathbb{P}(A = Y) - \mathbb{P}(A = N) \log \mathbb{P}(A = N) \quad (4)$$

where

$$\begin{aligned} \mathbb{P}(A = Y) &= \mathbb{P}(c)P_W + (1 - \mathbb{P}(c))(1 - P_W) \\ \mathbb{P}(A = N) &= (1 - \mathbb{P}(c)) P_W + \mathbb{P}(c)(1 - P_W) \end{aligned} \qquad (5)$$

Definition 10 (Problem Statement). *On two given schemata S and T, let the result set R and probability assignment function $\mathbb{P}$ be generated by some schema matching tools. Each CCQ is assumed to be answered independently. Let B be the budget of the number of CCQs to be asked to the crowd. Our goal is to maximize the reduction of $H(R)$ without exceeding the budget.*

3 SINGLE CCQ APPROACH

In this section, we study how to choose a single CCQ well. To be able to do this, we first address the formalization of uncertainty reduction. Then we develop the *Single CCQ Approach*, a framework to address the uncertainty reduction problem using a sequence of Single CCQ. Compared with [44], we give a new proof for uncertainty reduction under the condition of accuracy probability $P_W \in [0.5, 1]$. We prove the equivalent form of uncertainty reduction and its property. Finally, we propose efficient algorithms to implement the computations in this approach.

3.1 Formulation of Uncertainty Reduction

In order to design an effective strategy for manipulating CCQs, it is essential to define a measurement to estimate the importance of CCQs before they are answered. Since the final objective is to reduce uncertainty, we use uncertainty reduction caused by individual CCQs as the measurement. In the following, we provide

the formulation of the uncertainty reduction in the context of the Single CCQ Approach.

Let Q_c be a CCQ w.r.t an arbitrary correspondence c. We assume crowdsourcing workers provide answers independently with accuracy $P_W \in [0.5, 1]$. Since Q_c is a Yes/No question, we consider the answer A as a random variable following a Bernoulli distribution. Firstly, we have $D_A = \{Y, N\}$ and $\mathcal{P} = \{\mathbb{P}(A = Y), \mathbb{P}(A = N)\}$, where $\mathbb{P}(A = Y)$ and $\mathbb{P}(A = N)$ can be computed by Eq 5. We write $\mathbb{P}(Y)$ and $\mathbb{P}(N)$ for short. For two discrete random variables X and Y with p.m.f. function p, the conditional entropy is defined by

$$H(Y|X) = \sum_{x \in \mathcal{X}} p_X(x) H(Y|X = x)$$
$$= - \sum_{x \in \mathcal{X}} p_X(x) \sum_{y \in \mathcal{Y}} p_Y(y|X = x) \log p_Y(y|X = x)$$

Let ΔH_{Q_c} be the uncertainty reduction caused by Q_c, we have
$$\triangle H_{Q_c} = H(R) - H(R|A) \tag{6}$$
where
$$-H(R|A) = \mathbb{P}(Y) \sum_{m_i \in R} [\mathbb{P}(m_i|Y) \log \mathbb{P}(m_i|Y)]$$
$$+ \mathbb{P}(N) \sum_{m_i \in R} [\mathbb{P}(m_i|N) \log \mathbb{P}(m_i|N)]$$
$$\mathbb{P}(m_i|Y) = \frac{\mathbb{P}(m_i)\mathbb{P}(Y|m_i)}{P_W \mathbb{P}(c) + (1 - P_W)(1 - \mathbb{P}(c))}$$
$$\mathbb{P}(m_i|N) = \frac{\mathbb{P}(m_i)\mathbb{P}(N|m_i)}{P_W(1 - \mathbb{P}(c)) + (1 - P_W)\mathbb{P}(c)} \tag{7}$$

The uncertainty reduction w.r.t a given Q_c can be computed by Eq 6 provided that we know the values for parameters: $\mathbb{P}(c)$, $\mathbb{P}(m_i)$, $\mathbb{P}(Y|m_i)$ and $\mathbb{P}(N|m_i)$. $\mathbb{P}(c)$ can be computed by Eq 1. $\mathbb{P}(Y|m_i)$ and $\mathbb{P}(N|m_i)$ depend on if c is a correspondence included in m_i.

$$\mathbb{P}(Y|m_i) = \begin{cases} P_W & c \in m_i \\ 1 - P_W & c \notin m_i \end{cases};$$
$$\mathbb{P}(N|m_i) = \begin{cases} 1 - P_W & c \in m_i \\ P_W & c \notin m_i \end{cases} \tag{8}$$

Remark: The harmlessness of random answer If a worker W randomly answers a CCQ Q_c, i.e. $P_W = 0.5$ and $\mathbb{P}(Y) = 0.5$, it does not affect the uncertainty of schema matching. In other word, by Eq 7, we have $\mathbb{P}(m_i|Y) = \mathbb{P}(m_i)$.

Eq 7 is applied recursively as multiple answers are received, to take all of them into account. If multiple answers all agree, each iteration will make the truth of c more certain, whereas disagreeing answers will pull the probability closer to the middle. In other words, disagreements between workers are gracefully handled. It is easy to perform the algebraic manipulations to show that, for any two answers A_1 and A_2, we have $\mathbb{P}(m_i|A_1, A_2) = \mathbb{P}(m_i|A_2, A_1)$. This equation indicates that the result of adjustment is **independent** of the sequence of the answers. In other words, when we have a deterministic set of questions (CCQs), it does not matter in what sequence the answers are used for adjustment. In contrast, what matters is to determine the set of CCQs to be asked, which is the core challenge addressed in this paper.

3.2 Framework of Single CCQ

Having developed a technique to find the best Single CCQ, we can place this at the heart of an approach to solve the schema matching problem, as shown in Framework 1. The idea is to greedily select the single CCQ in each iteration that will result in the greatest reduction of uncertainty. We publish this CCQ; when it is answered and returned with accuracy rate (line 4), we adjust $\mathbb{P}(m_i)$ by $\mathbb{P}(m_i|A)$ (line 5), and then generate a new CCQ (line 7&8).

Framework 1 Single CCQ
1: $CONS \leftarrow 1$ // consumption of the budget
2: Find and publish Q_{c_i} that maximize ΔH_{Q_c} //
3: **while** there exists a CCQ in the crowd, we constantly monitor the CCQ **do**
4:    **for** answer A_i of Q_{c_i}, accuracy rate P_{W_i} **do**
5:       $\forall m_i \in R$ Adjust the $\mathbb{P}(m_i)$ by $\mathbb{P}(m_i|A)$ //
6:       **if** $CONS < B$ **then**
7:          Finding Q_{c_j} maximizing ΔH_{Q_c} //
8:          publish Q_{c_j},
9:          $CONS = CONS + 1$
10:       **else**
11:          terminate (no more budget)
12:       **end if**
13:    **end for**
14: **end while**

In the framework of Single CCQ, one can see that an important task is to find the CCQ with the highest uncertainty reduction as soon as the probability distribution of R is adjusted (line 7). We can formally pose this as a query as follows, and focus on efficiently processing such a query in the rest of this section.

Definition 11 (Single CCQ Selection (SCCQS)). *Given result set R, probability assignment function $\mathbb{P}()$, crowd's accuracy $P_W \in [0.5, 1]$, the Single CCQ Selection Query retrieves a CCQ maximizing the uncertainty reduction ΔH_{Q_c} in Eq 6.*

3.3 Query Processing of SCCQS

Based on the formulation in Section 3.1, we are able to compute the uncertainty reduction of each CCQ. So a naive approach of selection is to traverse all the CCQs. Such traversal results in an algorithm with time complexity $\mathcal{O}(|R|^2|C|)$, i.e. the square of the number of possible matchings multiplied by the number of correspondences. This can be a very large number for complex schema.

In this subsection, we first provide a lossless simplification, by proving the uncertainty reduction is mathematically equivalent to the entropy of the answer of a CCQ minus the entropy of the crowd. Then, in order to further improve the efficiency, we propose an index structure based on binary coding, together with a pruning technique.

3.3.1 Simplification of Single CCQ Selection

When we need to determine a strategy of selecting CCQs, a very intuitive idea is to prioritize the ones that we are more uncertain. In case of Single CCQ, this idea suggests that we select the CCQ with probability closest to 0.5. This idea is trivially correct when all the correspondences are independent. However, with the model of possible matchings, there are correlations among the correspondences. Then, a non-trivial question is: should we still pick the CCQ with probability closest to 0.5 with the presence of correlation?

Interestingly, we discover that the answer is positive. By Theorem 3.1, we prove that the uncertainty reduction ΔH_{Q_c} of a correspondence c is equivalent to the entropy of the answer $H(A)$ minus the entropy of the crowd $H(W)$. In other words, for a fixed P_W, ΔH_{Q_c} is only determined by $\mathbb{P}(c)$. As a result, searching for the CCQ that maximize ΔH_{Q_c} has the complexity decreased to $\mathcal{O}(|R||C|)$, by computing $\mathbb{P}(c)$ for each $c \in C$. In addition, Theorem 3.2 states that we only need to find the correspondence that has probability closest to 0.5, based on the fact that ΔH_{Q_c} is a symmetric function of $\mathbb{P}(c)$, with symmetry axis $\mathbb{P}(c) = 0.5$ and achieves maximum when $\mathbb{P}(c) = 0.5$.

Theorem 3.1. *Given result set R, probability assignment function $\mathbb{P}()$, crowd's accuracy $P_W \in [0.5, 1]$, for correspondence $c \in C$, we have*
$$\Delta H_{Q_c} = H(A) - H(W)$$

where we recall $\triangle H_{Q_c}$ in Eq 6 and $H(A)$, $H(W)$ are defined in Eq 4, Eq 3.

Proof. Using Eq 7 into Eq 6, we have

$$\triangle H_{Q_c}$$
$$= H(R) + \sum_{m_i \in R} [\mathbb{P}(m_i)\mathbb{P}(Y|m_i)\log \mathbb{P}(m_i)]$$
$$+ \sum_{m_i \in R} [\mathbb{P}(m_i)\mathbb{P}(Y|m_i)\log \mathbb{P}(Y|m_i) - \mathbb{P}(m_i)\mathbb{P}(Y|m_i)\log \mathbb{P}(Y)]$$
$$+ \sum_{m_i \in R} [\mathbb{P}(m_i)\mathbb{P}(N|m_i)\log \mathbb{P}(m_i) + \mathbb{P}(m_i)\mathbb{P}(N|m_i)\log \mathbb{P}(N|m_i)]$$
$$- \sum_{m_i \in R} [\mathbb{P}(m_i)\mathbb{P}(N|m_i)\log \mathbb{P}(N)]$$
$$:= H(R) + J_1 + J_2 + J_3 + J_4 + J_5 + J_6$$

By Eq 8, we obtain that

$$J_1 + J_4 = \sum_{m_i \in R} \mathbb{P}(m_i)\log \mathbb{P}(m_i) = -H(R)$$

and

$$J_2 + J_5$$
$$= \sum_{\substack{m_i \in R \\ c \in m_i}} \mathbb{P}(m_i) P_W \log P_W + \sum_{\substack{m_i \in R \\ c \notin m_i}} \mathbb{P}(m_i)(1 - P_W)\log(1 - P_W)$$
$$+ \sum_{\substack{m_i \in R \\ c \notin m_i}} \mathbb{P}(m_i) P_W \log P_W + \sum_{\substack{m_i \in R \\ c \in m_i}} \mathbb{P}(m_i)(1 - P_W)\log(1 - P_W)$$
$$= -H(W) \tag{9}$$

Recall Eq 1. It follows that

$$J_3 + J_6$$
$$= - \sum_{\substack{m_i \in R \\ c \in m_i}} \mathbb{P}(m_i) P_W \log \mathbb{P}(Y) - \sum_{\substack{m_i \in R \\ c \notin m_i}} \mathbb{P}(m_i)(1 - P_W)\log \mathbb{P}(Y)$$
$$- \sum_{\substack{m_i \in R \\ c \notin m_i}} \mathbb{P}(m_i) P_W \log \mathbb{P}(N) - \sum_{\substack{m_i \in R \\ c \in m_i}} \mathbb{P}(m_i)(1 - P_W)\log \mathbb{P}(N)$$
$$= - [\mathbb{P}(c) P_W + (1 - \mathbb{P}(c))(1 - P_W)]\log \mathbb{P}(Y)$$
$$- [(1 - \mathbb{P}(c)) P_W + \mathbb{P}(c)(1 - P_W)]\log \mathbb{P}(N)$$
$$= H(A)$$

This completes the proof. □

Theorem 3.2. *The uncertainty reduction of single CQQ is always non-negative for $P_W \in [0.5, 1]$. For any two correspondence $c, c' \in C$, if $|0.5 - \mathbb{P}(c)| \leq |0.5 - \mathbb{P}(c')|$ then $\Delta H_{Q_c} \geq \Delta H_{Q_c'} \geq 0$. In addition, $\Delta H_{Q_c} = 1 - P_W$ if $\mathbb{P}(c) = 0.5$*

Proof. By Theorem 3.1, ΔH_{Q_c} is a function of $\mathbb{P}(Y)$ and P_W. $\mathbb{P}(Y)$ is a function of $\mathbb{P}(c)$ and P_W. We first consider

$$\frac{\partial \triangle H_{Q_c}}{\partial \mathbb{P}(Y)} = -\log \frac{\mathbb{P}(Y)}{1 - \mathbb{P}(Y)}$$

We could obtain that

$$-\log \frac{\mathbb{P}(Y)}{1 - \mathbb{P}(Y)} = 0 \Leftrightarrow \mathbb{P}(Y) = 0.5$$

It is easy to check that ΔH_{Q_c} is a symmetric function of $\mathbb{P}(Y)$, with symmetry axis $\mathbb{P}(Y) = 0.5$. Besides, the function achieves maximum $\Delta H_{Q_c} = 1 - H(W)$ when $\mathbb{P}(A = Y) = 0.5$, and is monotonic on $[0, 0.5]$ (increasing) and $[0.5, 1]$ (decreasing). We also know that $\mathbb{P}(Y) = \mathbb{P}(c) P_W + (1 - \mathbb{P}(c))(1 - P_W)$. So $\mathbb{P}(Y)$ is increasing w.r.t. $\mathbb{P}(c)$ and achieves the value 0.5 when $\mathbb{P}(c) = 0.5$ or $P_W = 0.5$. Thus ΔH_{Q_c} achieves maximum $1 - H(W)$ when $\mathbb{P}(c) = 0.5$. Secondly we consider

$$\frac{\partial \triangle H_{Q_c}}{\partial P_W} = (2\mathbb{P}(c) - 1)\log \frac{\mathbb{P}(c) - (2\mathbb{P}(c) - 1) P_W}{(2\mathbb{P}(c) - 1) P_W - \mathbb{P}(c) + 1} + \log \frac{P_W}{1 - P_W}$$

Since ΔH_{Q_c} is a symmetric function of $\mathbb{P}(A = Y)$, with symmetry axis $\mathbb{P}(A = Y) = 0.5$ and $\mathbb{P}(A = Y)$ is increasing w.r.t. $\mathbb{P}(c)$, we choose $\mathbb{P}(c) = 0$ and $\mathbb{P}(c) = 1$ in order to obtain minimum of ΔH_{Q_c}. When $\mathbb{P}(c) = 0$ or $\mathbb{P}(c) = 1$, we have

$\frac{\partial \triangle H_{Q_c}}{\partial P_W} = 0$ and $\Delta H_{Q_c} = 0$. Thus we prove that ΔH_{Q_c} is non-negative. □

Running Example (Selecting First Two CCQs): Now we illustrate the process of selecting the first two CCQs in Framework 1 with the example of Table 1. In line 2, the first correspondence to be asked is c_2, since its probability is closest to 0.5 among $\{c_1, c_2, c_3, c_4, c_5\}$. Explicitly, $\Delta H_{Q_{c_2}} = -0.7 * log(0.7) - 0.3 * log(0.3) = 0.88$. Suppose an answer "$a = yes$" is received from a crowd worker, whose personal error rate is $1 - P_W = 0.2$ (line 4). Then we conduct the adjustment according to Eq 7, and have $\mathbb{P}(m_1|a) = 0.58$, $\mathbb{P}(m_2|a) = 0.10$ and $\mathbb{P}(m_3|a) = 0.32$. This adjustment is referring to the first-time execution of line 5. Then, in line 7, the next CCQ is to be selected. Note that, since the probabilities of possible matchings are adjusted, probabilities of correspondences should be recomputed by Eq 1: $\mathbb{P}(c_1) = 0.68$, $\mathbb{P}(c_2) = 0.9$, $\mathbb{P}(c_3) = 1$, $\mathbb{P}(c_4) = 0.68$, $\mathbb{P}(c_5) = 0.32$. Therefore, in line 7, we select the CCQ based on the updated probabilities of correspondences, i.e. c_1 would be selected. (There is a tie among c_1, c_4 and c_5, and we break the tie sequentially.)

3.3.2 Binary Coding and Pruning Techniques

One can see that a basic computation of our algorithm is to check whether a given correspondence c is in a given possible matching m_i. Since the correspondences included in each possible matching do not change with the value of overall uncertainty, we propose to index R with a binary matrix M_R, where element $e_{ij} = 1(0)$ representing $c_j \in m_i (c_j \notin m_i)$. Equipped with this index, we apply a pruning technique derived from Theorem 3.2.

Now we illustrate the procedure of generating the correspondence with probability closest to 0.5. For each c_j, we traverse m_i and accumulate $\mathbb{P}(m_i)$ if $e_{ij} = 1$. Let $c_{best\_so\_far}$ be the best correspondence so far, with probability $\mathbb{P}(c_{best\_so\_far})$. Then, let c_j be the current correspondence, and P_{acc} be its accumulated probability after reading some $\mathbb{P}(m_i)$, then c_j can be safely pruned if we have $P_{acc} - 0.5 \geq |\mathbb{P}(c_{best\_so\_far}) - 0.5|$.

4 MULTIPLE CCQ APPROACH

A drawback of single CCQ is that only one correspondence is resolved at a time. Each resolution, even if quick, requires human time scales, and comes with some overhead to publish the corresponding HIT and tear it down. Gaining confidence in a single schema matching may require addressing many CCQs. The time required to do this in sequence may be prohibitive.

An alternative we consider in this section is to issue multiple (k) CCQs simultaneously. Different workers can then pick up these tasks and solve them in parallel, cutting down wall-clock time. However, we pay for this by having some questions answered that are not at the top of the list – we are issuing k good questions rather than only the very best one.

Note that there are three possible states for a published CCQ: (1) **waiting** - no one has accepted the question yet; (2) **accepted** - someone in the crowd has accepted the question and is working on it; (3) **answered** - the answer of the CCQ is available. What's more important, one can withdraw published CCQs that are still at state waiting (e.g. forceExpireHIT in Mechanical Turk APIs) [10]. In other words, publishing a CCQ does not necessarily consume the budget. It is possible that a CCQ is published, and then withdrawn before anyone in the crowd answers it. In such case, the budget is not consumed. Because of the dependence between correspondences, we can withdraw or replace some of the published CCQs that are at "waiting" state. Equipped with this

power, we propose the Multiple CCQ approach to dynamically keep k best CCQs published at all times.

In the rest of this section, we provide the formulation and framework of Multiple CCQs, by extending our results of Single CCQ. Compared with our conference paper [44], we give new proofs for uncertainty reduction under more general condition that crowd workers have accuracy probabilities $P_{W_i} \in [0.5, 1]$. These probabilities can show hardness of CQQs or how professional workers are. Accuracy probabilities are assumed before we ask CCQs and are returned with answers after we publish CCQs. They can be totally different for different correspondences and different crowd workers. We prove **the uncertainty reduction equals to joint entropy of answers minus sum of entropies of crowds**. We also show upper and lower bounds for uncertainty reduction. Results in [44] can be viewed as a special case when workers are always correct. Finally, we prove the NP-hardness of the multiple CCQs selection problem, and propose an efficient approximation algorithm with bounded error.

4.1 Formulating Uncertainty Reduction of Multiple CCQ Approach

For a set of CCQs of size k - $S_Q = \{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}$, A_{c_1}, A_{c_2}, ..., A_{c_k} denote answers of k CCQs given by k workers W_1, W_2, ..., W_k with accuracy $P_{W_1}, P_{W_2}, ..., P_{W_k}$. We want to derive the uncertainty reduction caused by the aggregation of the answers of these k CCQs. Let D_A and $\mathcal{P}_A$ be the domain and probability distribution of answers respectively. Each element of D_A is a possible set of answers for k CCQs (a sequence of Y and N) with a corresponding probability in $\mathcal{P}_A$. Then first we have

$$D_A = \left\{ a_i \,\middle|\, a_i = \left\{ A_{c_1}^{(i)}, A_{c_2}^{(i)}, ..., A_{c_k}^{(i)} \right\}, \; A_{c_j}^{(i)} = Y \text{ or } N \right\}$$
$$\mathcal{P}_A = \{\mathbb{P}(a_1), \mathbb{P}(a_2), ..., \mathbb{P}(a_{2^k})\}$$

where $|D_A| = 2^k$. As we know, each correspondence c_t has a probability to show its ground truth, i.e. with $\mathbb{P}(c_t)$ to be true before crowds answer CCQs. We view $U = \{c_t; t = 1, ..., k\}$ as a set of random variables which follow Bernoulli distribution and take value True/False. Note that they are not independent and their joint p.m.f. can be calculated by Eq 2. Let D_U and $\mathcal{P}_U$ be the domain and probability distribution of $\{c_t; t = 1, ..., k\}$ respectively. Each element of D_U is a sequence of T and F with a corresponding probability in $\mathcal{P}_U$. Thus we have

$$D_U = \left\{ u_i \,\middle|\, u_i = \left\{ c_1^{(i)}, c_2^{(i)}, ..., c_k^{(i)} \right\}, \; c_t^{(i)} = T \text{ or } F \right\} \quad (10)$$
$$\mathcal{P}_U = (\mathbb{P}(u_1), \mathbb{P}(u_2), ..., \mathbb{P}(u_{2^k}))$$

where $|D_U| = 2^k$. By Eq 2, we have

$$\mathbb{P}(u_i) = \sum_{\substack{m_j \in R \\ t=1,...,k \\ \forall c_t^{(i)}=T, \, c_t \in m_j \\ \forall c_t^{(i)}=F, \, c_t \notin m_j}} \mathbb{P}(m_j) \quad (11)$$

Remark: Complexity We remark that in computation of all $\mathbb{P}(u_i)$, $i = 1, ..., 2^k$, each $\mathbb{P}(m_j)$, $j = 1, ..., |R|$ will be used once and only once. Therefore, the number of elements with positive probability in D_U is less than or equal to $|R|$ and time complexity of computing all $\mathbb{P}(u_i)$ is bounded by $\mathcal{O}(k|R|)$.

Similar to Eq 6, we are able to compute the uncertainty reduction caused by the S_Q, denoted by ΔH_{S_Q}. We have

$$\Delta H_{S_Q} = H(R) - H(R|A_{c_1}, ..., A_{c_k})) \quad (12)$$
$$= H(R) + \sum_{a_i \in D_A} \mathbb{P}(a_i) \sum_{m_j \in R} [\mathbb{P}(m_j|a_i) \log \mathbb{P}(m_j|a_i)]$$
$$= H(R) + \sum_{\substack{m_j \in R \\ a_i \in D_A}} \mathbb{P}(m_j)\mathbb{P}(a_i|m_j) \log \frac{\mathbb{P}(m_j)\mathbb{P}(a_i|m_j)}{\mathbb{P}(a_i)}$$

Computation of $\mathbb{P}(a_i)$: For one CQQ c_i, A_{c_i} is the answer given by a worker with accuracy P_{W_i}. When A_{c_i} is Yes, c_i may be True and worker is correct, or c_i is False and worker is incorrect. It is easy to see that

$$\mathbb{P}(A_{c_i} = Y) = \mathbb{P}(c_i)P_{W_i} + (1 - \mathbb{P}(c_i))(1 - P_{W_i})$$
$$\mathbb{P}(A_{c_i} = N) = (1 - \mathbb{P}(c_i))P_{W_i} + \mathbb{P}(c_i)(1 - P_{W_i})$$

For k CQQs, the answers in a_i are denoted by $A_{c_1}^{(i)}, A_{c_2}^{(i)}, ..., A_{c_k}^{(i)}$. Similarly, $\mathbb{P}(a_i)$ can be computed by Eq 11.

$$\mathbb{P}(a_i) = \sum_{j=1}^{2^k} \mathbb{P}(u_j) q_{ij} \quad (13)$$

where

$$q_{ij} = \prod_{\substack{t=1,...,k \\ c_t^{(j)}=T \\ A_{c_t}^{(i)}=Y}} P_{W_t} \prod_{\substack{t=1,...,k \\ c_t^{(j)}=T \\ A_{c_t}^{(i)}=N}} (1 - P_{W_t}) \prod_{\substack{t=1,...,k \\ c_t^{(j)}=F \\ A_{c_t}^{(i)}=Y}} (1 - P_{W_t}) \prod_{\substack{t=1,...,k \\ c_t^{(j)}=F \\ A_{c_t}^{(i)}=N}} P_{W_t}$$

Computation of $\mathbb{P}(a_i|m_j)$: Similar to Single CCQ, Eq 8, $\mathbb{P}(a_i|m_j)$ depends on whether correspondences are in the possible matching m_i. In definition 10 we assume that each CCQ is answered independently. Therefore, given that m_j is the correct matching, we know the correct answers for k CCQs and answers $A_{c_1}^{(i)}, A_{c_2}^{(i)}, ..., A_{c_k}^{(i)}$ are k independent Bernoulli random variables. It follows that

$$\mathbb{P}(a_i|m_j) =$$
$$\prod_{\substack{t=1,...,k \\ c_t \in m_j \\ A_{c_t}^{(i)}=Y}} P_{W_t} \prod_{\substack{t=1,...,k \\ c_t \in m_j \\ A_{c_t}^{(i)}=N}} (1 - P_{W_t}) \prod_{\substack{t=1,...,k \\ c_t \notin m_j \\ A_{c_t}^{(i)}=Y}} (1 - P_{W_t}) \prod_{\substack{t=1,...,k \\ c_t \notin m_j \\ A_{c_t}^{(i)}=N}} P_{W_t}$$

Running Example: In the example of Table 1, we assume two CCQs c_1 and c_2 are answered by two workers with $P_{W_1} = 0.8$ and $P_{W_2} = 0.6$. Domains of correspondences and answers are

$$D_U = \{(T,T), (T,F), (F,T), (F,F)\}$$
$$D_A = \{(Y,Y), (Y,N), (N,Y), (N,N)\}$$

Probability distribution for D_U is given by Eq 11:

$$\mathbb{P}(u_i = (T,T)) = \mathbb{P}(m_1) = 0.45$$
$$\mathbb{P}(u_i = (T,F)) = \mathbb{P}(m_2) = 0.3$$
$$\mathbb{P}(u_i = (F,T)) = \mathbb{P}(m_3) = 0.25$$
$$\mathbb{P}(u_i = (F,F)) = 0$$

Probability distribution for D_A is given by Eq 13: $\mathbb{P}(a_i = (Y,Y)) = 0.45 P_{W_1} P_{W_2} + 0.3 P_{W_1}(1 - P_{W_2}) + 0.25(1 - P_{W_1})P_{W_2} = 0.342$. Similarly, $\mathbb{P}(a_i = (Y,N)) = 0.308$, $\mathbb{P}(a_i = (N,Y)) = 0.198$ and $\mathbb{P}(a_i = (N,N)) = 0.152$. Given m_1 is the correct matching, we know that c_1 and c_2 are T. Thus

$$\mathbb{P}(a_i = (Y,Y)|m_1) = P_{W_1} P_{W_2} = 0.48$$
$$\mathbb{P}(a_i = (Y,N)|m_1) = P_{W_1}(1 - P_{W_2}) = 0.32$$
$$\mathbb{P}(a_i = (N,Y)|m_1) = (1 - P_{W_1})P_{W_2} = 0.12$$
$$\mathbb{P}(a_i = (N,N)|m_1) = (1 - P_{W_1})(1 - P_{W_2}) = 0.08$$

4.2 Framework of Multiple CCQ

As shown in Framework 2, the best size-k set of CCQs are initially selected and published with accuracy rates to show their hardness, and then we constantly monitor their states. Whenever one or more answers are available, three operations are conducted. First, all CCQs at state "waiting" are withdrawn. Second, the probability distribution is adjusted with the new answers (line 8&9). Last, we regenerate and publish a set of CCQs that are currently most contributive (lines 12&15). In general, we keep the best k CCQs in the crowd, by interactively changing CCQs based on newly received answers. Note that the number of CCQs may be less than k when the budget is insufficient (line 14-16). The whole procedure terminates when the budget runs out and all the CCQs are answered (line 3).

In contrast with Single CCQ, the essential query of Multiple CCQ is to find a group of k CCQs, which maximize the uncertainty reduction. Formally, we have following definition:

Framework 2 Multiple CCQ

1: $CONS \leftarrow k$ // consumption of the budget
2: given initial accuracy rates for all correspondences, find and publish a set of CCQs - $S_Q = \{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}$ that maximize ΔH_{S_Q} //(See 4.1)
3: **while** there exists CCQs in the crowd, we constantly monitor the CCQs **do**
4:    **if** receive the one or more answers $A_1, A_2, ..$ with accuracy $P_{W_1}, P_{W_2}, ...$ **then**
5:      withdraw all the CCQs at waiting state
6:      $k' \leftarrow$ *the number of CCQs withdrawn*
7:      $k'' \leftarrow$ *the number of answers received*
8:      **for** each A_i, P_{W_i} **do** //Adjustment
9:        $\forall m_i \in R$ Adjust the $\mathbb{P}(m_i)$ by $\mathbb{P}(m_i | A_1, A_2, ..)$ //
10:      **end for**
11:      **if** $CONS + k'' <= B$ **then**
12:        find a set of CCQs - S'_Q of size $(k' + k'')$ that currently maximize $\Delta H_{S'_Q}$ //(See 4.1)
13:        $CONS = CONS + k''$
14:      **else** // no sufficient budget for maintaining k CCQs
15:        find a set of CCQs - S'_Q of size $(B - CONS)$ that currently maximize $\Delta H_{S'_Q}$ //(See 4.1)
16:        $CONS = B - k'$
17:      **end if**
18:      $\forall Q'_{c_i} \in S'_Q$ publish Q'_{c_i}
19:    **end if**
20: **end while**

Definition 12 (Multiple CCQ Selection (MCCQS)).
Given result set R, probability assignment function $\mathbb{P}()$, and an integer k, the multiple CCQ selection problem is to retrieve a set of k CCQs, denoted by S_Q, such that the uncertainty reduction, ΔH_{S_Q}, is maximized.

One can see that, if we set $k = B$ (recall B is the budget of CCQs), the problem of MCCQS selects the optimal set of correspondences at which to ask CCQs in order to maximize the uncertainty reduction. Similar to [32] and [43], MCCQS itself is an interesting and valuable optimization problem to investigate.

4.3 Simplification of Multiple CCQ Selection

In case of Single CCQ, considering each CCQ as a random variable, we proved that the uncertainty reduction of a CCQ is equivalent to entropy of answer minus entropy of crowd. In Multiple CCQ, analogously, we are interested to find a relation between uncertainty reduction and entropy for a size-k set of CCQs. This is complex since the correspondences are correlated.

As shown in Theorem 4.1, under the assumption that crowds give correct answers with accuracy probability, we prove that the uncertainty reduction by a set of CCQs is equivalent to their ***joint entropy*** (denoted by $H(D_A)$) minus sum of entropies of crowds, while in previous conference paper [44], the result can be viewed as a special case of this result when crowds' accuracies equal to 1. Facilitated with this theorem, we could reduce MCCQS to a special case of joint entropy maximization problem. Similarly with definition 9, the joint entropy $H(D_A)$ of answers $A_{c_1}, A_{c_2}, ..., A_{c_k}$ w.r.t. CCQs $Q_{c_1}, Q_{c_2}, ..., Q_{c_k}$ are defined by

$$H(D_A) = - \sum_{a_i \in D_A} \mathbb{P}(a_i) \log \mathbb{P}(a_i) \tag{14}$$

where $\mathbb{P}(a_i)$ can be computed by Eq 13.

Theorem 4.1. *Given result set R, probability assignment function $\mathbb{P}()$, a set of CCQs $S_Q = \{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}$, answers $A_{c_1}, A_{c_2}, ..., A_{c_k}$, accuracies of crowd workers $P_{W_1}, P_{W_2}, ..., P_{W_k}$ in $[0.5, 1]$, we have*

$$\Delta H_{S_Q} = H(D_A) - \sum_{t=1}^{k} H(W_t)$$

Proof. By Eq 12, we have
ΔH_{S_Q}

$$= H(R) + \sum_{\substack{m_j \in R \\ a_i \in D_A}} \mathbb{P}(m_j) \mathbb{P}(a_i | m_j) \log \mathbb{P}(m_j)$$

$$+ \sum_{\substack{m_j \in R \\ a_i \in D_A}} [\mathbb{P}(m_j) \mathbb{P}(a_i | m_j) \log \mathbb{P}(a_i | m_j) - \mathbb{P}(m_j) \mathbb{P}(a_i | m_j) \log \mathbb{P}(a_i)]$$

$$:= H(R) + J_1 + J_2 + J_3$$

By definition 5, we have

$$J_1 = \sum_{m_j \in R} \left[\sum_{a_i \in D_A} \mathbb{P}(a_i | m_j) \right] \mathbb{P}(m_j) \log \mathbb{P}(m_j)$$

$$= \sum_{m_j \in R} \mathbb{P}(m_j) \log \mathbb{P}(m_j) = -H(R)$$

By Eq 14, we have

$$J_3 = - \sum_{a_i \in D_A} \left[\sum_{m_j \in R} \mathbb{P}(m_j | a_i) \right] \mathbb{P}(a_i) \log \mathbb{P}(a_i)$$

$$= - \sum_{a_i \in D_A} \mathbb{P}(a_i) \log \mathbb{P}(a_i) = H(D_A)$$

Given $m_i, A_{c_1}, ..., A_{c_k}$ are independent. For J_2, by the property of joint entropy of independent random variables, we have

$$\sum_{a_i \in D_A} \mathbb{P}(a_i | m_j) \log \mathbb{P}(a_i | m_j) = -H(A_{c_1}, ..., A_{c_k} | m_j)$$

$$= - \sum_{t=1}^{k} H(A_{c_t} | m_j)$$

Therefore, similarly with Eq 9,

$$J_2 = - \sum_{m_j \in R} \mathbb{P}(m_j) \sum_{t=1}^{k} H(A_{c_t} | m_j)$$

$$= \sum_{t=1}^{k} \left[\sum_{m_j \in R} \mathbb{P}(m_j) \mathbb{P}(A_{c_t} = Y | m_j) \log \mathbb{P}(A_{c_t} = Y | m_j) \right.$$

$$\left. + \sum_{m_j \in R} \mathbb{P}(m_j) \mathbb{P}(A_{c_t} = N | m_j) \log \mathbb{P}(A_{c_t} = N | m_j) \right]$$

$$= \sum_{t=1}^{k} [P_{W_t} \log P_{W_t} + (1 - P_{W_t}) \log (1 - P_{W_t})] = - \sum_{t=1}^{k} H(W_t)$$

This completes the proof. $\qquad\square$

4.4 Upper bound and lower bound of Uncertainty Reduction

In this subsection, we show the upper and lower bounds for $H(D_A)$, which can be applied to improve approximate algorithm. We recall $U = \{c_t; t = 1, ..., k\}$, D_U, $\mathcal{P}_U$ Eq 10 and $\mathbb{P}(u_i)$ Eq 11. Now we define joint entropy $H(D_U)$ by

$$H(D_U) = - \sum_{u_i \in D_U} \mathbb{P}(u_i) \log \mathbb{P}(u_i) \tag{15}$$

We remark that $H(D_U)$ measures the uncertainty of k correspondences, while $H(D_A)$ measures the uncertainty of answers for k correspondences. Intuitively, this difference is caused by the fact that crowds make mistakes. If $P_{W_i} = 1$ for all $i = 1, ..., k$, $H(D_U) = H(D_A)$.

As mentioned in subsection 4.1, the number of elements with positive probability in D_U is at most $|R|$. Time complexity of computing all $\mathbb{P}(u_i)$ is bounded by $\mathcal{O}(k|R|)$. However $|D_A| = 2^k$, by Eq 13, time complexity of computing all $\mathbb{P}(a_i)$ will be $\mathcal{O}(2^k)$. Thus we hope to bound $H(D_A)$ by $H(D_U)$.

Theorem 4.2. *Under the assumption of Theorem 4.1, let*

$$h^{(u)}(D_A) = \min \left\{ H(D_U) + \sum_{t=1}^{k} H(W_t), - \sum_{t=1}^{k} \log (1 - P_{W_t}) \right\}$$

and

$$h^{(l)}(D_A)$$

$$= \max \left\{ -\sum_{t=1}^{k} \log P_{W_t}, \quad H(D_U) + \sum_{t=1}^{k} H(W_t) \right.$$

$$+ \prod_{t=1}^{k} P_{W_t} \log \prod_{t=1}^{k} P_{W_t} + \left(1 - \prod_{t=1}^{k} P_{W_t}\right) \log \left(1 - \prod_{t=1}^{k} P_{W_t}\right)$$

$$\left. - \left(1 - \prod_{t=1}^{k} P_{W_t}\right) \min\left\{\log(2^k - 1), H(D_U)\right\} \right\}$$

We have

$$h^{(l)}(D_A) \leq H(D_A) \leq h^{(u)}(D_A) \tag{16}$$

Proof. **Upper bound**: By the chain rule of conditional entropy, we have

$$H(D_A) = H(D_A, D_U) - H(D_U|D_A) \tag{17}$$
$$= H(D_A|D_U) + H(D_U) - H(D_U|D_A)$$

where

$$H(D_A|D_U) = -\sum_{u_j \in D_U} \mathbb{P}(u_j) \left[\sum_{a_i \in D_A} \mathbb{P}(a_i|u_j) \log \mathbb{P}(a_i|u_j) \right]$$

Given u_j, we know the true correspondences and false ones in U, thus $A_{c_t}, t = 1, ..., k$ are independent. We obtain that

$$H(D_A|D_U)$$

$$= \sum_{u_j \in D_U} \mathbb{P}(u_j) \sum_{t=1}^{k} H(A_{c_t}|u_j)$$

$$= -\sum_{t=1}^{k} [\sum_{u_j \in D_U} \mathbb{P}(u_j)\mathbb{P}(A_{c_t} = Y|u_j) \log \mathbb{P}(A_{c_t} = Y|u_j)$$

$$+ \sum_{u_j \in D_U} \mathbb{P}(u_j)\mathbb{P}(A_{c_t} = N|u_j) \log \mathbb{P}(A_{c_t} = N|u_j)]$$

$$= -\sum_{t=1}^{k} [\sum_{\substack{u_j \in D_U \\ c_t^{(j)}=F}} \mathbb{P}(u_i)(1 - P_{W_t}) \log(1 - P_{W_t})$$

$$+ \sum_{\substack{u_j \in D_U \\ c_t^{(j)}=T}} \mathbb{P}(u_i)P_{W_t} \log P_{W_t} + \sum_{\substack{u_j \in D_U \\ c_t^{(j)}=F}} \mathbb{P}(u_i)P_{W_t} \log P_{W_t}$$

$$+ \sum_{\substack{u_j \in D_U \\ c_t^{(j)}=T}} \mathbb{P}(u_i)(1 - P_{W_t}) \log(1 - P_{W_t})]$$

$$= \sum_{t=1}^{k} H(W_t)$$

Thus we get

$$H(D_A) \leq H(D_U) + \sum_{t=1}^{k} H(W_k) \tag{18}$$

On the other hand, by definition of $H(D_A)$ Eq 14 and $\mathbb{P}(a_i)$ Eq 13, we have

$$H(D_A) = -\sum_{i=1}^{2^k} \sum_{j=1}^{2^k} \mathbb{P}(u_j)q_{ij} \log \mathbb{P}(u_j)q_{ij} \tag{19}$$

Note that $\sum_{j=1}^{2^k} \mathbb{P}(u_j) = 1$, which means $\sum_{j=1}^{2^k} \mathbb{P}(u_j)q_{ij}$ is a linear combination of $q_{ij}, j = 1, 2, ..., 2^k$. It is easy to see that

$$\sum_{j=1}^{2^k} \mathbb{P}(u_j)q_{ij} \geq \min_j q_{ij} = \prod_{t=1}^{k}(1 - P_{W_t})$$

where last equation holds because each $P_{W_t} \in [0.5, 1]$. Then we have

$$H(D_A) \leq -\sum_{i=1}^{2^k} \sum_{j=1}^{2^k} \mathbb{P}(u_j)q_{ij} \log \prod_{t=1}^{k}(1 - P_{W_t}) = -\sum_{t=1}^{k} \log(1 - P_{W_t})$$

Together with Eq 18, we achieve the upper bound.

Lower bound: The difference between $H(D_A)$ and $H(D_U)$ is that crowds have probability to make mistakes. Inspired by this,

we consider the indicator function that crowds make at least one mistake, i.e.

$$Y = \begin{cases} 0 & \text{Answers are all correct} \\ 1 & \text{Crowds make mistake} \end{cases} \tag{20}$$

Obviously, we have $\mathbb{P}(Y = 0) = \prod_{t=1}^{k} P_{W_t}$. In order to obtain lower bound, it is sufficient to bound the term $H(D_U|D_A)$ in Eq 17. Thus we rewrite

$$H(D_U|D_A)$$

$$= H(D_U|D_A) - H(D_U|D_A, Y) + H(D_U|D_A, Y)$$

$$= H(Y|D_A) - H(Y|D_A, D_U) + H(D_U|D_A, Y) \tag{21}$$

$$= H(Y|D_A) + H(D_U|D_A, Y)$$

$$\leq H(Y) + H(D_U|D_A, Y)$$

$$= H(Y) + \sum_{a_i \in D_A} [\mathbb{P}(a_i, Y = 0)H(D_U|a_i, Y = 0) \tag{22}$$

$$+ \mathbb{P}(a_i, Y = 1)H(D_U|a_i, Y = 1)] \tag{23}$$

where the second equation Eq 21 is obtained by chain rule of entropy. Please note that when $Y = 0$, $A_{c_t}^{(i)} = Y$ if $c_t^{(i)} = T$ and $A_{c_t}^{(i)} = N$ if $c_t^{(i)} = F$. Thus in Eq 22, we have

$$H(D_U|a_i, Y = 0) = 0$$

The entropy is maximized when each possible outcome has the same probability. Since $|D_U| = 2^k$ and when $Y = 1$, we know that the number of possible outcome is $2^k - 1$. Therefore in Eq 23, we have

$$H(D_U|a_i, Y = 1) \leq \min\left\{H(D_U), \log\left(2^k - 1\right)\right\}$$

Now we write

$$H(D_U|D_A)$$

$$\leq H(Y) + \min\left\{H(D_U), \log\left(2^k - 1\right)\right\} \sum_{a_i \in D_A} \mathbb{P}(a_i, Y = 1)$$

$$= H(Y) + \min\left\{H(D_U), \log\left(2^k - 1\right)\right\} \mathbb{P}(Y = 1)$$

$$= -\prod_{t=1}^{k} P_{W_t} \log \prod_{t=1}^{k} P_{W_t} - \left(1 - \prod_{t=1}^{k} P_{W_t}\right) \log\left(1 - \prod_{t=1}^{k} P_{W_t}\right)$$

$$+ \left(1 - \prod_{t=1}^{k} P_{W_t}\right) \min\left\{H(D_U), \log\left(2^k - 1\right)\right\}$$

Substitute this bound into Eq 17, we achieve that

$$H(D_A) \geq H(D_U) + \sum_{t=1}^{k} H(W_t) + \prod_{t=1}^{k} P_{W_t} \log \prod_{t=1}^{k} P_{W_t}$$

$$+ \left(1 - \prod_{t=1}^{k} P_{W_t}\right) \log\left(1 - \prod_{t=1}^{k} P_{W_t}\right)$$

$$- \left(1 - \prod_{t=1}^{k} P_{W_t}\right) \min\left\{H(D_U), \log\left(2^k - 1\right)\right\}$$

On the other hand by Eq 19, we have

$$H(D_A) \geq -\sum_{i=1}^{2^k} \sum_{j=1}^{2^k} \mathbb{P}(u_j)q_{ij} \log \prod_{t=1}^{k} P_{W_t} = -\sum_{t=1}^{k} \log P_{W_t}$$

This completes the proof. □

Remark: When $P_{W_t} = 1$ for all $t = 1, ..., k$, we can check that

$$h^{(l)}(D_A) = h^{(u)}(D_A) = H(D_U) = H(D_A)$$

When $P_{W_t} = 0.5$ for all $t = 1, ..., k$, we can check that

$$h^{(l)}(D_A) = h^{(u)}(D_A) = k = H(D_A)$$

Our result is optimal in the sense that lower bound equals to upper bound in two extreme cases: When crowds always give correct answers ($P_{W_t} = 1$) and when crowds always give random answers without any consideration ($P_{W_t} = 0.5$).

4.5 NP-hardness of Multiple CCQ Selection

By Theorem 4.1, searching a group of k CCQs with maximal uncertainty reduction is equivalent to *finding k CCQs with maximal joint entropy*. It is known the joint entropy of a set of random variables is a *monotone sub-modular function*. In general, maximizing sub-modular functions is NP-hard. Concerning the computation of the value of information, [20] shows that, for a general reward function R_j (in our problem, $R_j = \Delta H_{S_Q}$), it is $NP^{PP} - hard$ to select the optimal subset of variables even for discrete distributions that can be represented by polytree graphical models. $NP^{PP} - hard$ problems are believed to be much harder than NPC or $\#PC$ problems. In the problem of multiple CCQ selection, every variable is binary and their marginal distribution is represented by a binary matrix. As a result, a naive traversal would lead to an algorithm of $\mathcal{O}(|R||C|^k)$ complexity, since the searching space (i.e. the number of subsets to select) is always of size $C_{|C|}^k$.

With the Theorem 4.3, we prove that Multiple CCQ Selection is NP-hard. Encountering this NP-hardness, we propose a efficient approximation algorithm based on the sub-modularity of joint entropy.

Theorem 4.3. *The Multiple CCQ Selection is NP-hard.*

Proof. To reach the proof of Theorem 4.3, it is sufficient to prove the NP-completeness of its decision version, Decision MCCQS (DMCCQS), i.e. given result set *R*, probability assignment function $\mathbb{P}()$, an integer k, and a value ΔH, decide whether one can find a set S_Q of k CCQs such that $\Delta H_{S_Q} >= \Delta H$.

To reach the NP-completeness of DMCCQS, it is sufficient to prove a special case of DMCCQS is NPC. First we let accuracy rates equals to 1. Moreover we state the special case of DMCCQS by adding the following constraint on R: for each way of partitioning R into two subsets S_1 and S_2, there exists a correspondence c such that $(\forall m_i \in S_1, c \in m_i) \land (\forall m_j \in S_2, c \notin m_i)$. Equipped with this constraint, we this reduce special case of DMCCQS to the *set partition problem*.

The partition problem is the task of deciding whether a given multiset of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2.

Transformation: Given a set partition problem with input multiset S, let $Sum = \sum_{x \in S} x$. We create a possible matching m_i for each positive integer $x_i \in S$, and assign its possibility $\mathbb{P}(m_i) = x_i / Sum$. Let the correspondences satisfy the constraint, and we set $k = 1, \Delta H = -\log(0.5) = 1$ for DMCCQS.

($\Longrightarrow$) If there is a yes-certificate for the set partition problem, then the R can be partitioned into two subsets, each with aggregate probability 0.5. According to the constraint, the exists a correspondence c with $\mathbb{P}(c) = 0.5$. Then, selecting $S_Q = \{Q_c\}$ would achieve uncertainty reduction $H_{S_Q} = -0.5 \log 0.5 - 0.5 \log 0.5 = 1$. Therefore, $\{Q_c\}$ serves as yes-certificate for the special case of DMCCQS.

($\Longleftarrow$) Assume there is yes-certificate for the special case of DMCCQS when $k = 1, \Delta H = -\log(0.5) = 1$. Since $k = 1$, H_{S_Q} is actually equivalent to H_c. Then by Theorem 3.2, there exists a correspondence c such that $\mathbb{P}(c) = 0.5$. Therefore, by the constraint, there is a way to partition R into two subsets, each with aggregate probability 0.5. Since the mapping from the positive integers to the possible matchings is one-to-one, we obtain an yes-certificate for the special case of DMCCQS. $\square$

4.6 Approximation Algorithm

It is known that the joint entropy of a set of random variables is a *monotone sub-modular function* [20]. And the problem of

Fig. 2. Illustration of R Partitioning

selecting a k-element subset maximizing a monotone sub-modular function can be approximated with a performance guarantee of $(1-1/e)$, by iteratively selecting the most uncertain variable given the ones selected so far [19]. Formally, we have the optimization function at the kth iteration:

$$X := \arg \max_{Q_{c_k}} \Delta H_{S_Q^{k-1} \cup \{Q_{c_k}\}} \quad (24)$$

Let $A^{(k-1)}$ denote answers for S_Q^{k-1}. By the chain rule of conditional entropy, we have

$$H\left(D_{A^{(k-1)}}, A_{c_k}\right) = H\left(D_{A^{(k-1)}}\right) + H\left(A_{c_k} | D_{A^{(k-1)}}\right)$$

Thus we only need to maximize the conditional entropy at each iteration, i.e.

$$X := \arg \max_{A_{c_k}} H\left(A_{c_k} | D_{A^{(k-1)}}\right)$$

and

$$\begin{aligned} &H\left(A_{c_k} | D_{A^{(k-1)}}\right) \\ &= - \sum_{a_i \in D_{A^{(k-1)}}} \mathbb{P}(a_i) \left[\mathbb{P}\left(A_{c_k} = Y | a_i\right) \log \mathbb{P}\left(A_{c_k} = Y | a_i\right) \right. \\ &\quad \left. + \mathbb{P}\left(A_{c_k} = N | a_i\right) \log \mathbb{P}\left(A_{c_k} = N | a_i\right)\right] \end{aligned} \quad (25)$$

Eq 25 indicates that, at each iteration, we are searching the most uncertain correspondence, given the correspondences selected in previous iterations. In particular, after the $(k-1)^{th}$ iteration, the possible matchings are at most split into 2^{k-1} partitions, each of which corresponds to an element $a_i \in D_{A^{(k-1)}}$. We aim to find the kth correspondence, in order to further split them to at most 2^k partitions, such that then entropy of resulting partitions is maximized. Figure 2 illustrates a partitioning of the first two iterations. Motivated with this interpretation, we propose to apply an in-memory index to maintain the list of partitions for each iteration. One can see that each partition corresponding to a_i is essentially a set of possible matchings. In addition, also index $\mathbb{P}(a_i)$ associated with each partition.

As a result, the computation of $H(A_{c_k}|D_{A^{(k-1)}})$ for each candidate correspondence is simply traversing the list of partitions. Note the number of partitions is at most $|R|$ (i.e. each partition has only one possible matching), so the overall complexity is upper bounded by $\mathcal{O}(k|R||C|)$. However, there is still room for the further pruning of the search space. In the follows, we derive four pruning techniques to avoid traversing all the partitions. Each pruning indicates a condition that guarantees certain partitions are unnecessary to be considered, hence speed up the overall computation. For simplicity, we just use the notation a_i to represent the partition corresponding to a_i. Then, for the iteration, we have partitions $a_1, a_2, ..., a_n$ with probabilities $\mathbb{P}(a_1), \mathbb{P}(a_2), ..., \mathbb{P}(a_n)$ respectively. As follows, we present four pruning rules.

Pruning Rule 4.4. *If a partition a_i has only one matching, a_i can be safely pruned, i.e. we can remove a_i from the list of partitions.*

Pruning rule 4.4 utilizes the intuition that the correctness of a possible matching m can be fully determined by the selected correspondences, when m is the only one in its partition. In other words, the remaining correspondences of m would not contribute any more information, hence should not be selected.

Pruning Rule 4.5. *Let c be a candidate correspondence, then c can be safely pruned (for the rest of the iterations), if all a_i, one of the following conditions are met for :(1) $\forall m_i \in a_i, c \in m_i$, (2)$\forall m_i \in a_i, c \notin m_i$*

Similar to Pruning rule 4.4, Pruning rule 4.5 indicates the condition that the correctness of c can be determined by selected correspondences.

Next, we introduce Pruning Rule 4.6 and 4.7, which derives two non-trivial upper bounds, which enable effective pruning.

Pruning Rule 4.6. *Let $H_{best\_so\_far}$ be the best value of Eq 25 so far for the current iteration, then for the correspondence c, let $a_1, a_2, ..., a_m$ be the partitions c already traversed Let*

$$H_0 = -\sum_{i=1}^{m} \mathbb{P}(a_i) \left[\mathbb{P}(Y|a_i) \log \mathbb{P}(Y|a_i) + \mathbb{P}(N|a_i) \log \mathbb{P}(N|a_i) \right]$$

Then c can be pruned for the current iteration, if we have

$$H_{best\_so\_far} - H_0 \leq \sum_{j=m+1}^{n} \mathbb{P}(a_j) \log \frac{\mathbb{P}(a_j)}{2}$$

Proof. For the rest of partitions $a_{m+1}, a_{m+2}, ..., a_n$, the optimal situation is they are all perfectly bisected, that is $\forall i \in [m+1, n]$, $\mathbb{P}(Y|a_i) = \mathbb{P}(N|a_i) = 0.5$. Therefore, their contribution to the optimization function has a upper bound

$$\sum_{j=m+1}^{n} \mathbb{P}(a_j) \log \frac{\mathbb{P}(a_j)}{2}$$

$\square$

Pruning Rule 4.7. *Let $H_{best\_so\_far}$ be the best value of Eq 25 so far for the current iteration. For a correspondence c, let $H(A_c|D_{A^{(k-2)}})$ be the conditional entropy computed from a previous iteration. Then, c can be pruned for the current iteration if*

$$H\left(A_c|D_{A^{(k-2)}}\right) \leq H_{best\_so\_far}$$

Proof. This pruning rule reflects the sub-modularity of the joint entropy. S_Q^{k-2} is the set of CCQs selected in the previous iteration, so $S_Q^{k-2} \subset S_Q^{k-1}$, where S_Q^{k-1} is the CCQs selected for the current iteration. Then by sub-modularity, we have

$$H\left(D_{A^{(k-2)}}, A_c\right) - H\left(D_{A^{(k-2)}}\right) \geq H\left(D_{A^{(k-1)}}, A_c\right) - H\left(D_{A^{(k-1)}}\right)$$

and equivalently, $H\left(A_{c_k}|D_{A^{(k-2)}}\right) \geq H\left(A_{c_k}|D_{A^{(k-1)}}\right)$, which completes the proof. $\square$

At last we use Theorem 4.2 to show a pruning rule.

Pruning Rule 4.8. *Given the selected correspondences S_Q^{k-1} in previous (k-1)th iterations, two current potential selected correspondences c_1 and c_2, correspondence c_2 could be safely filtered if these two correspondences satisfy*

$$h^{(l)}(D_{A^{(k-1)}} \cup A_{c_1}) \geqslant h^{(u)}(D_{A^{(k-1)}} \cup A_{c_2})$$

5 EXPERIMENTAL RESULTS

We conducted extensive experiments to evaluate our approaches, based on both simulation and real implementation. We focus on evaluating two issues. First, we examine the effectiveness of our two frameworks in reducing the uncertainty for possible matchings. Second, we verify the correctness of our approaches, by evaluating the precision and recall of the best matchings.

5.1 Experimental Setup

We adopt the schema matching tool OntoBuilder [11], [13], which is one of the leading tools for schema matching. In particular, we conduct our experiments on five datasets, each of which includes five schemata. The schemata are extracted from web forms from different domains. We describe the characteristics of each dataset in Table 3. By OntoBuilder, schemata are parsed into xml schemata, and attributes refer to nodes with semantic information. We conduct pairwise schema matching within each domain, so there are totally 40 pairs of schemata (10 for each domain). In OntoBuilder, four schema matching algorithms are implemented,

Fig. 3. Single CCQ v.s. Random - Simulation

namely *Term*, *Value*, *Composition* and *Precedence*. For each pair of schemata, we generate 400 unique possible matchings (100 for each algorithm). In addition, each possible matching is associated with a global score, which indicates the goodness of the matching. We obtain the probabilities of matchings by normalizing the global scores. The details of these algorithms can be found in [11].

5.2 Simulation

To evaluate the effectiveness of our two approaches, we first conduct a simulation of the crowd's behaviour, based on our formulation in Section 3.1. First, we manually select the best matching from the 400 possible matchings, and treat the selected matching as the correct matching (i.e. ground truth). So for any correspondence, its correctness depends on whether it is in the selected matching. Second, for each published CCQ, we randomly generate an accuracy rate $P_W \in [0.5, 1]$ following an uniform distribution. Third, given a CCQ, we generate the correct yes-no answer with probability P_W (i.e. generate the wrong answer with probability $1 - P_W$, and then return the answer and P_W as the inputs for adjustment (Section 3.1).

First, we present the effectiveness of Single CCQ approach (Framework 1), by comparing its performance with randomly selecting CCQs. We set the budget $B = 50$, and each CCQ is generated after receiving the answer of the previous one. Figure 3 illustrates the average change of uncertainty (vertical axis) with the number of answers of CCQs received (horizontal axis). With the increase of number of CCQs, the uncertainty converges to zero rapidly. From the experimental results, our proposed Single CCQ approach (SCCQ) outperforms the random approach (Random) significantly. Please note that all the results plotted in Section 5.2 and 5.3 are averages over 10 runs. The distribution is quite dense within each domain, but diverse for different domains.

Next, we examine the performance of Multiple CCQ (Framework 2). Recall that we need to constantly monitor the CCQs, and update the CCQs whenever new answers are received. In the simulation of conference paper [44], we check the states of published CCQs every time unit. Each published CCQ is initially at state "waiting". For each time unit, each CCQ in state "waiting" may change to "accepted" with probability P_0 (remain unchanged with probability $1 - P_0$), where P_0 is a random number generated from $(0, 0.5)$; and each CCQ at state "accepted" may change to "answered" with probability P_1 (remain unchanged with probability $1 - P_1$), where P_1 follows a Poisson distribution. Figure 4 illustrates the performance of Multiple CCQ by varying k, where we set the budget $B = 50$. Recall that k, a parameter

TABLE 3
DATASETS

Notation	Source	No.of attributes
Hotel	hotel searching websites	14-20
Aviation	homepages of airline companies	12-18
BookStore	the webpages of advanced search in online book stores	13-21
ComplaintForm	the complaint forms of government websites	27-34
News	news websites	43-60

Fig. 4. Multiple CCQ with different k - Simulation

of Framework 2, represents the number of CCQ in the crowd. Whenever a CCQ is answered, we dynamically updated the k CCQs, to make sure the k CCQs are the best according to the all received answers. In particular, when k=1, Framework 2 becomes the Single CCQ approach. One can observe that the curves with smaller k tend to have better performance in terms of reducing uncertainty. In fact, the larger k is, the less advantage MCCQ has comparing to a random selection. Recall each time we select k out of $|C|$ correspondences, and when $k = |C|$, MCCQ is the same as random selection, i.e. select all of the correspondences we have.

As discussed in Section 4, the increase of k leads to less uncertainty reduction (which is consistent with the result in Figure 4), but improves the overall time efficiency. Since there are multiple uncontrollable factors affecting the completion time of workers, the time cost of the proposed approaches are hard to be simulated. Nevertheless, we analyse the relation between k and the time cost in the real-world implementation in Section 5.3.

5.3 Testing on Amazon Mechanical Turk

We implement our two approaches on Amazon Mechanical Turk (AMT), which is a widely used crowdsourcing marketplace. Empowered with the Amazon Mechanical Turk SDK, we are able to interactively publish and manage the CCQs. Each HIT of AMT includes all the attributes of two schemata, one CCQ, and the URLs of the source web-pages. Each HIT is priced US$0.05. One can see that each HIT is essentially a CCQ. For the rest of this section, the terms "HIT" and "CCQ" are exchangeable.

In analogy to the simulation, Figure 5 and Figure 6 illustrate the performances of Single CCQ and Multiple CCQ respectively, where we set the budget $B = 50$. In terms of uncertainty reduction, one can see that the performance is basically consistent with the simulation. A very important finding is that, in contrast with the simulation, the uncertainty is likely to increase when the first several CCQs are answered. The increase can happen when a surprising answer is obtained, i.e. a yes answer is returned for low-probability correspondence, or vice versa. This phenomenon indicates that, the budget should be large enough to achieve satisfactory reduction of uncertainty.

Fig. 5. Single CCQ v.s. Random - on Amazon Mechanical Turk

Fig. 6. Multiple CCQ with different k on Amazon Mechanical Turk

Another important finding is that, the uncertainty convergence to zero in real implementation is much slower than that in the simulation. A possible reason is that, we use a Bernoulli distribution to model the error rate of workers. But in reality, the error rate follows a much complex distribution, which may be related to the dataset.

Lastly, we present the overall time cost of Single CCQ and Multiple CCQ approaches in the real implementations, where totally 50 CCQs are published and answered. As shown in Figure 7, the curves with larger k tend to have less time cost. Please note that, the case of Single CCQ is indicated with $k = 1$. When k is increased, we get faster initial reduction on uncertainty, but the overall reduction tend to be limited. Actually, there are many uncontrollable factors would affect the completion time, such as the difficulty of the CCQs, the time of publication etc.

5.4 Data Quality

In this subsection, we verify the correctness of our approaches, by evaluating the precision and recall of the best matching, i.e. the possible matching with the highest possibility after the uncertainty reduction. Precision is computed as the ratio of correct correspondences out of the total number of correspondences in the correct matching (ground truth). Recall is computed as the ratio of correct correspondences out of the total number of correspondences in the correct matching. Since the performances are very similar on different datasets, we merge the four datasets into one, and present the precision and recall averaged from 40 runs.

Figure 8 illustrates the quality of the best matching after uncertainty reduction with budget $B = 50$. The suffixes "_S" and

Fig. 7. Time Cost with different k on Amazon Mechanical Turk

Fig. 8. Data Quality with Budget Constraint- Precision & Recall

"_R" represent the data obtained from the simulation and the real-world implementation on AMT, respectively. B mainly depend on how much money the HIT requester will pay for the task. In the simulation, the precision and recall are almost 100%. In the real-world implementation, 50 questions by SCCQ make precision and recall over 90%, which are significantly better than that of the "machine-only" methods when k is small. However, in the real implementation, we find that when k is increased, the precision and recall tend to be decreased dramatically. In particular, for cases $k = 8$ and $k = 16$, the MCCQ is only slightly better than the *Composition*. The reason is twofold: first, comparing to SCCQ, there is averagely less information for selecting CCQs in MCCQ; second, due to the NP-hardness, we are only able to select CCQs that are near-optimal.

Recall that the motivation of MCCQ is to improve the time efficiency. Therefore, we conducted another set of experiments where time is the constraint, in order to investigate the relation between k and data quality. Explicitly, we preform SCCQ and MCCQ for 50 minutes, without any limit on the budget. The precision and recall are demonstrated in Fig 9. From the experimental results, we conclude that **the MCCQ with large k has outstanding performance for time-constrained situations**. Therefore, we conclude that k **should be set to a small value when the budget is the main constraint; whereas a large value is suggested for k if time-efficiency is the primary constraint**.

5.5 New Experiments

With a more realistic model in this paper, we conduct experiments of MCCQ again. In simulation, firstly we randomly generate accuracy rates following uniform distribution on $[0.5, 1]$ for all correspondences as their hardness attribute. We publish k initial CCQs with state "waiting". We still check the states of published CCQs every time unit. For each time unit, each CCQ in state "waiting" may change to "accepted" with probability P_0 and each CCQ at state "accepted" may change to "answered" with probability P_1. Each answer is returned with an accuracy rate P_{W_i} as the trustworthiness of the crowd. Accuracy rates P_{W_i} also follows uniform distribution on $[0.5, 1]$. We still set budget $B = 50$ and Figure 10 shows the performance of Multiple

Fig. 9. Data Quality with Time Constraint - Precision & Recall

Fig. 10. Multiple CCQ with different k - Simulation(New)

CCQ by varying k. Then in Figure 11 we apply our MCCQ approach on Amazon Mechanical Turk. The difference between new experiments and the old ones in [44] is that we consider initial accuracy rates and different accuracy rates in each step. In [44], assumption of theoretical results is that accuracy rates equal to 1, while in experiments we chose accuracy rates less than 1. Moreover, in this paper we obtain optimal upper bound and lower bound for entropy reduction, so that pruning rules are more efficient. These are major reasons that our new choices for CCQs are comparatively better in terms of entropy reduction with less fluctuation.

At last we consider a new dataset with more attributes and we set $B = 70$, $k = 8$. Let X be beta distribution $Beta(2, 2)$. In Figure 12, we try different distributions for P_{W_i}. Line 1 shows P_{W_i} follows uniform distribution on $[0.5, 1]$ with mean 0.75 and variance $1/48$. In Line 2, $P_{W_i} = 0.5X + 0.5$, thus $P_{W_i} \in [0.5, 1]$ with mean 0.75 and variance $1/80$. In Line 3, $P_{W_i} = 0.4X + 0.6$, thus $P_{W_i} \in [0.6, 1]$ with mean 0.8 and variance $1/125$. In line 4, $P_{W_i} = 0.6X + 0.4$, thus $P_{W_i} \in [0.4, 1]$ with mean 0.7 and variance $9/500$. Line 5 shows the result in AMT. Comparing first four lines, we can see Line 3 perform best as P_{W_i} has biggest mean and smallest variance. Line 4 perform worst since in practice we do not choose a crowd worse than 0.5.

6 RELATED WORK

6.1 Uncertainty in Schema Matching

The model of possible matching, namely "probabilistic schema mappings", was first introduced in [8]. In their work, algorithmic approaches generate a set of matchings between two schemata, with a probability attached to each matching. After the collection of possible matchings is determined, the probability of each correspondence can be computed by summing up the probabilities of possible matchings in which the correspondence is included. Later, Sarma et al. [37] used well-known schema matching tools (COMA, AMC, CLIO, Rondo, etc.) to generate a set of correspondences associated with confidence values between two schemata. Then, the possible matchings are constructed from these correspondences and data instances. A more intuitive method of constructing possible matchings is proposed in [12]. In detail, [12] generates top-k schema matchings by combining the matching

Fig. 11. Multiple CCQ with different k on Amazon Mechanical Turk(New)

Fig. 12. MCCQ with different P_{W_i}

results generated by various matchers, and each of the k matchings is associated with a global score. Then possible matchings are constructed by normalizing the global scores. Additionally, the model of possible matchings has been adopted in [14] as a core foundation for answering queries in a data integration system with uncertainty. Gal [11] used the top-K schema mappings from a semi-automatic matchers to improve the quality of the top mapping. [8] [14] and [34] were devoted to the parallel use of uncertain schema matchings, and proposed new semantics of queries.

The uncertainty in schema matching has been intensively studied, primarily focusing on the query processing in the presence of uncertainty. X.Dong et al. [8] concentrated on the semantics and properties of probabilistic schema mappings. We assume that a set of probabilistic schema matchings is provided by an existing algorithm, such as one of those mentioned above. How to efficiently process uncertain data is an orthogonal issue, which has been well addressed, such as [16], [41], [42].

A probabilistic matching network model was established in [29] to reduce uncertainty of schema matching. Authors developed pay-as-you-go reconciliation approach. Probabilities of correspondences are defined in their model independently of schema matching tools. [36] discussed schema matching prediction which is an assessment mechanism to support schema matchers in the absence of an exact match.

6.2 Crowdsourcing and Data Integration
Such as schema matching, some queries cannot be answered by machines only. The recent booming up of crowdsourcing brings us a new opportunity to engage human intelligence into the process of answering such queries (see [7] [22] [3] as survey for crowdsourcing). In general, [10] proposed a query processing system using microtask-based crowdsourcing to answer queries. Many classical queries are studied in the context of crowdsourced database, including max [15], filtering [30], sorting [24] etc. In [31], a declarative query model is proposed to cooperate with standard relational database operators. In [4], crowdsourcing is used for top-K query processing over uncertain data. As a typical application related to data integration, [43] utilized a hybrid human-machine approach on the problem of entity resolution. [26] studied knowledge base semantic integration using crowdsourcing.

[25] engages crowdsourcing into schema matching. In particular, [25] proposed to enlist the multitude of users in the community to help match the schemata in a Web 2.0 fashion. The difference between our work and [25] is threefold: (1) From the conceptual level, "crowd" in [25] refers to an on-line community (e.g. a social network group); while we explicitly consider the crowd as crowdsourcing platforms (e.g. Mechanical Turk). (2) The essential output of [25] is determined by the "system builders", which means the end users still have to get involved in the process of schema matching. (3) We focus on the optimization between the cost (the number of CCQs) and performance (uncertainty reduction).

6.3 Active Learning
Active learning is a form of supervised machine learning, in which a learning algorithm is able to interact with the workers (or some other information source) to obtain the desired outputs at new data points. A widely used technical report is [38]. In particular, [28], [45] proposed active learning methods specially designed for crowd-sourced databases. Our work is essentially different from active learning in two perspectives: (1) the role of workers in active learning is to improve the learning algorithm (e.g. a classifier); in this paper, the involvement of workers is to reduce the uncertainty of given matchings. (2) The uncertainty of answers are usually assumed to be given before generating any questions; in this paper, the uncertainty of answers has to be considered after the answers are received, since we cannot anticipate which workers would answer our questions. To our best knowledge, there is no algorithm in the field of active learning can be trivially applied to our problem.

7 CONCLUSION AND FUTURE WORK
In this paper, we propose two novel approaches, namely Single CCQ and Multiple CCQ, to apply crowdsourcing to reduce the uncertainty of schema matching generated by semi-automatic schema matching tools. These two approaches adaptively select and publish the optimal set of questions based on new received answers. Technically, we significantly reduce the complexity of CCQ selection by proving that the expectation of uncertainty reduction caused by a set of CCQs are mathematically equivalent to the join entropy of answers minus entropy of crowds. In addition, we obtain optimal bounds for uncertainty reduction, prove NP-hardness of MCCQ Selection, and design an $(1 + \epsilon)$ approximation algorithm, based on its sub-modular nature. One challenge we overcome is to investigate difficulties of CCQs and trustworthiness of crowd-sourced answers by accuracy rates of crowds.

Uncertainty is inherited in many components in modern data integration systems, such as entity resolution, schema matching, truth discovery, name disambiguation etc. We believe that embracing crowdsourcing as a component of a data integration system would be extremely conductive for the reduction of uncertainty, hence effectively improve the overall performance. Our work represents an initial solution towards automating uncertainty reduction of schema matching with crowdsourcing.

A future work regarding to MCCQ is that: in Theorem 4.1, we distribute k CCQs to crowds each time. We obtain a formula of uncertain reduction under the assumption that we take back k answers. In reality, we do not know how many CCQs can be answered. We may withdraw or replace some CCQs after a waiting time. The choice of next k CCQs is best only when all k CCQs are answered. Therefore investigating a more realistic and complete

model with answer rates(a difficult CCQ may has a probability that no one accept it) may further help reducing the matching uncertainty.

REFERENCES

[1] Z. Bellahsene, A. Bonifati, and E. Rahm. *Schema Matching and Mapping*. Springer, 2011.

[2] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic schema matching, ten years later. *PVLDB*, 4(11):695–701, 2011.

[3] A. I. Chittilappilly, L. Chen, and S. Amer-Yahia. A survey of general-purpose crowdsourcing techniques. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2246–2266, 2016.

[4] E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Crowd-sourcing for top-k query processing over uncertain data. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):41–53, 2016.

[5] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *Joint 2013 EDBT/ICDT Conferences*, pages 225–236, 2013.

[6] L. Detwiler, W. Gatterbauer, B. Louie, D. Suciu, and P. Tarczy-Hornoch. Integrating and ranking uncertain scientific data. In *ICDE*, pages 1235–1238, 2009.

[7] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, 2011.

[8] X. L. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. *VLDB J.*, 18(2):469–500, 2009.

[9] J. Fan, M. Lu, B. C. Ooi, W.-C. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *ICDE*, pages 976–987, 2014.

[10] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.

[11] A. Gal. Managing uncertainty in schema matching with top-k schema mappings. *J. Data Semantics VI*, 4090:90–114, 2006.

[12] A. Gal. *Uncertain Schema Matching*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[13] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB J.*, 14(1):50–67, 2005.

[14] A. Gal, M. V. Martinez, G. I. Simari, and V. S. Subrahmanian. Aggregate query answering under uncertain schema mappings. In *ICDE*, pages 940–951, 2009.

[15] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396, 2012.

[16] J. Huang, L. Antova, C. Koch, and D. Olteanu. Maybms: a probabilistic database management system. In *SIGMOD*, pages 1071–1074, 2009.

[17] N. Q. V. Hung, N. T. Tam, Z. Miklos, and K. Aberer. On leveraging crowdsourcing techniques for schema matching networks. In *International Conference on Database Systems for Advanced Applications*, pages 139–154, 2013.

[18] N. Q. V. Hung, N. T. Tam, Z. Miklós, and K. Aberer. Reconciling schema matching networks through crowdsourcing. *EAI Endorsed Trans. Collaborative Computing*, 1(2):e2, 2014.

[19] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[20] A. Krause and C. Guestrin. A note on the budgeted maximization on submodular functions. (CMU-CALD-05-103), 2005.

[21] P. Lemay. *The Statistical Analysis of Dynamics and Complexity in Psychology: A Configural Approach*. Université de Lausanne, Faculté des sciences sociales et politiques, 1999.

[22] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 28(9):2296–2319, 2016.

[23] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.

[24] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[25] R. McCann, W. Shen, and A. Doan. Matching schemas in online communities: A web 2.0 approach. In *ICDE*, pages 110–119, 2008.

[26] R. Meng, L. Chen, Y. Tong, and C. Zhang. Knowledge base semantic integration using crowdsourcing. *IEEE transactions on knowledge and data engineering*, 29(5):1087–1100, 2017.

[27] R. J. Miller, L. M. Haas, and M. A. Hernández. Schema mapping as query discovery. In *VLDB*, pages 77–88, 2000.

[28] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden. Active learning for crowd-sourced databases. *CoRR*, abs/1209.3686, 2012.

[29] Q. V. H. Nguyen, T. T. Nguyen, Z. Miklos, K. Aberer, A. Gal, and M. Weidlich. Pay-as-you-go reconciliation in schema matching networks. In *ICDE*, pages 220–231. IEEE, 2014.

[30] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, pages 361–372, 2012.

[31] A. G. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In *CIDR*, pages 160–166, 2011.

[32] A. G. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *PVLDB*, 4(5):267–278, 2011.

[33] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.

[34] Y. Qi, K. S. Candan, and M. L. Sapino. Ficsr: *f*eedback-based *inc*onsistency *r*esolution and query processing on misaligned data sources. In *SIGMOD Conference*, pages 151–162, 2007.

[35] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[36] T. Sagi and A. Gal. Schema matching prediction with applications to data source discovery and dynamic ensembling. *The VLDB Journal*, 22(5):689–710, 2013.

[37] A. D. Sarma, X. Dong, and A. Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD Conference*, pages 861–874, 2008.

[38] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[39] C. E. Shannon and W. Weaver. A mathematical theory of communication. *Bell Syst. Tech. J.*, 1948.

[40] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. In *Journal on data semantics IV*, pages 146–171. Springer, 2005.

[41] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu. Mining frequent itemsets over uncertain databases. *PVLDB*, 5(11):1650–1661, 2012.

[42] Y. Tong, L. Chen, and B. Ding. Discovering threshold-based frequent closed itemsets over probabilistic data. In *ICDE*, pages 270–281, 2012.

[43] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[44] C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 6(9):757–768, 2013.

[45] L. Zhao, G. Sukthankar, and R. Sukthankar. Robust active learning using crowdsourced annotations for activity recognition. In *Human Computation*, 2011.

Chen Jason Zhang received the PhD degree from the Department of Computer Science and Engineering(CSE) at the Hong Kong University of Science and Technology(HKUST) in 2015. He is currently associate professor in Shandong University of Finance and Economics. His research interests include crowdsourcing and data integration.

Lei Chen received the PhD degree in Computer Science from the University of Waterloo, Canada, in 2005. He is currently a Professor in department of CSE, HKUST. His research interests include crowdsourcing, uncertain databases and data integration.

H. V. Jagadish is currently the Bernard A Galler Collegiate Professor of Electrical Engineering and Computer Science at the University of Michigan. He received his Ph.D. from Stanford University in 1985. His research interests include databases and Big Data.

Mengchen Zhang received his Ph.D. degree from department of Mathematics at HKUST in 2017. He is currently a research assistant in department of CSE, HKUST. His research interests include Stein's method in probability , crowdsourcing and data integration.

Yongxin Tong received the Ph.D. degree in department of CSE, HKUST in 2014. He is currently an associate professor in the School of Computer Science and Engineering, Beihang University. His research interests include crowdsourcing, uncertain data mining and social network analysis.

Determining Correspondences (Batch *No. 2*)

Given Schema A and Schema B, describing two web pages of two different online book stores; attribute1 and attribute2 (colored in red) are two attributes from Schema A and B, respectively. Please detemine whether the attribute1 and attribute2 are corresponding to each other.

The two attributes are corresponding to each other if they reflect the same concept. Please answer yes if you think there is no other attribute in Schema B better suited for attribute1, and no other attribute in Schema A better suited for attribute2. If you think the two attributes are irrelevent, please answer no. Each attribute has two parts, separated by ":" . The term on the left of ":" is the data type of the attribute, the term on the right of ":" is name of the attribute.

A given attribute to be matched may have data type "group", which means you should consider the attributes with the same name as a whole component.

attribute1: submit: Go()

- Schema A -
- 1 select: All (url)
- 2 text: (field-keywords)
- 3 submit: Go ()
- 4 text: Keywords (field-keywords)
- 5 text: Author (field-author)
- 6 text: Title (field-title)
- 7 text: ISBN(s) (field-isbn)
- 8 text: Publisher (field-publisher)
- 9 text: Pub. Date Month Year (field-dateyear)
- 10 select: Pub. Date Month Year (field-dateop)
- 11 select: Pub. Date Month Year (field-datemod)
- 12 image: Go (Adv-Srch-Books-Submit)
- 13 select: Subject (node)
- 14 select: Condition (field-p_n_condition-type)
- 15 select: Format (field-feature_browse-bin)

  Show Source Link

attribute2: submit: search(submit)

- Schema B -
- 1 text: (SearchString)
- 2 image: search (imageField)
- 3 text: title (title)
- 4 radio: title (titlePos)
- 5 text: author (author)
- 6 radio: partial exact beginning (authPos)
- 7 text: isbn (ISBN)
- 8 text: keyword (keyword)
- 9 submit: search (submit)
- 10 select: format (mediaFormat)
- 11 select: Imprint (imprint)
- 12 select: Series (series)

  Show Source Link

○ Yes

○ No

Any comments are welcome, we appreciate your participation!

Fig. 13. screen shot of CCQ on AMT