Reducing Uncertainty of Schema Matching via Crowdsourcing

Chen Jason Zhang[§] Lei Chen[§] H. V. Jagadish [†] Chen Caleb Cao[§]
[§]Hong Kong University of Science and Technology, Hong Kong, China
[†]University of Michigan, Ann Arbor, MI, USA
czhangad@cse.ust.hk, leichen@cse.ust.hk, jag@umich.edu, caochen@cse.ust.hk

ABSTRACT

Schema matching is a central challenge for data integration systems. Automated tools are often uncertain about schema matchings they suggest, and this uncertainty is inherent since it arises from the inability of the schema to fully capture the semantics of the represented data. Human common sense can often help. Inspired by the popularity and the success of easily accessible crowdsourcing platforms, we explore the use of crowdsourcing to reduce the uncertainty of schema matching.

Since it is typical to ask simple questions on crowdsourcing platforms, we assume that each question, namely *Correspondence Correctness Question* (CCQ), is to ask the crowd to decide whether a given correspondence should exist in the correct matching. We propose frameworks and efficient algorithms to dynamically manage the CCQs, in order 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 the questions. We verified the value of our solutions with simulation and real implementation.

1. INTRODUCTION

Schema matching refers to finding correspondences between elements of the two given schemata, which is a critical issue for many database applications such as data integration, data warehousing, and electronic commerce [20]. 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 [20] for a comprehensive survey). The majority combines 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" [14].

Proceedings of the VLDB Endowment, Vol. 6, No. 9

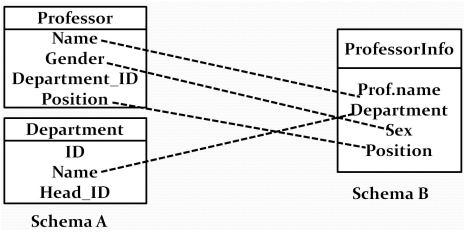

Figure 1: Example of Schema Matching Problem

Table 1: UNCERTAIN SCHEMA MATCHING

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

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

Given this inherent ambiguity, many schema matching tools will produce not just one matching, but rather a whole set of possible matchings. In fact, there is even a stream of work dealing with models of possible matchings, beginning with [3]. 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 [9][1]. However, preserving the uncertainty complicates the 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 [18] 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 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. There exist platforms, such as Amazon Mechanical Turk, which can serve as sources of human perceptions. The data concerning an explicit problem can be queried via publishing questions, named Human Intelligent Tasks (a.k.a HITs). The workflow of publishing HITs can be automated with available APIs (e.g. REST APIs) [4]. 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.

It is well-known that crowdsourcing works best when tasks can be broken down into very simple pieces. An entire schema matching 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 [3], 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. For example, in Table 1, the correspondence c_1 holds in m_1 and m_2, but not m_3. Therefore, its probability is obtained as $0.45 + 0.3 = 0.75$.

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 is estimating 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, 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 contributive questions in the crowd, and dynamically updates questions according to newly received answers.

To summarize, we have made the 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. In Section 4.3, we prove that the uncertainty reduction is equivalent to the joint entropy of CCQs, by considering each CCQ as a binary variable. We further show that greedy is indeed optimal.

3. For the Single CCQ approach, we propose an explicit framework, and derive an efficient algorithm for the "Single CCQ" in Section 3. We introduce an index structure and pruning technique for efficiently finding the Single CCQ. We also develop the mathematical machinery to deal with possibly erroneous answers from the crowd.

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

In addition, 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 several future works.

2. PROBLEM STATEMENT

In this section, we give the 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 sets of attributes from S and T respectively.*

DEFINITION 2 (POSSIBLE MATCHING). *Let S and T be two given schemata. Every possible matching m_i is a set of correspondences between S and T, i.e. $m_i = \{c_1, c_2, \ldots, c_{|m_i|}\}$*

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 particular, we require that no attribute participate in more than one correspondence in a possible matching.

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

In the example of Table 1, the set $RS = \{m_1, m_2, m_3\}$ is the result set. We have $Pr(m_1) + Pr(m_2) + Pr(m_3) = 1$. In practice, schema matching tools may use some thresholding 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.

DEFINITION 4 (CORRESPONDENCE SET). *Let RS be the result set for two given schemata S and T, the correspondence set CS*

is the set of all correspondences contained by possible matchings in RS, i.e.

$$CS = \bigcup_{m_i \in RS} m_i \quad (1)$$

Note that a correspondence can appear in more than one possible matching, so for any correspondence $c \in CS$, let $Pr(c)$ be the probability of c being in the correct matching, then

$$Pr(c) = \sum_{m_i \in RS \wedge c \in m_i} Pr(m_i) \quad (2)$$

As a simple extension, for a set of correspondences $S \subseteq CS$, let $Pr(S)$ be the probability that all correspondences of S are in the correct matching, then

$$Pr(S) = \sum_{m_i \in RS \wedge S \subseteq m_i} Pr(m_i) \quad (3)$$

For example, in Table 1, we have Correspondence Set $CS = \{c_1, c_2, c_3, c_4, c_5\}$. Since c_1 is in m_1 and m_2, we have $Pr(c_1) = Pr(m_1) + Pr(m_2) = 0.75$. In addition, let $S = \{c_1, c_2\}$, then $S \subseteq m_1, S \nsubseteq m_2, S \nsubseteq m_3$. Therefore we have $Pr(S) = 0.45$.

DEFINITION 5 (UNCERTAINTY OF SCHEMA MATCHING). *For two given schemata S and T, given result set RS and probability assignment function $Pr()$, we measure the uncertainty of RS with Shannon entropy -*

$$H(RS) = -\sum_{m_i \in RS} Pr(m_i) \log Pr(m_i)$$

In the example of Table 1, we have the uncertainty of Result Set $H(RS) = -0.45 \log 0.45 - 0.3 \log 0.3 - 0.25 \log 0.25 = 1.54$ (base=2).

Remark: A major reason for utilizing Shannon entropy as the measurement of uncertainty 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 [12], such as the possible matchings.

If we know a particular correspondence to be true (or false), this will automatically disqualify any matching in which this correspondence is absent (respectively, present). The probability for any such matching is then set to zero. Since the probabilities of all possible matchings must sum to one, the probabilities of the remaining (not invalidated) matchings are increased proportionately. Once matching probabilities are updated, correspondence probabilities have to be recomputed as well, to reflect the new matching probabilities.

For example, given Table 1, we issue an HIT (sending c_2) to the crowd and get the confirmation that c_2 is correct, that is $Pr(c_2) = 1$. Then we have $Pr(m_1|c_2) = Pr(m_1)Pr(c_2|m_1)/Pr(c_2) = 0.45/0.7 = 0.643$. Similarly, $Pr(m_2|c_2) = 0$ and $Pr(m_3|c_2) = 0.357$. Once the confidences of possible matchings RS are updated, the entropy, $H(RS)$, now becomes 0.94, smaller than 1.54, which reflects the reduction of the uncertainty.

After the probabilities of the possible matchings are adjusted, the probabilities of correspondences are updated, to continue to reflect the sums of probabilities of the appropriate matchings, as follows: $Pr(c_1) = 0.643$, $Pr(c_2) = 1$, $Pr(c_3) = 1$, $Pr(c_4) = 0.643$, $Pr(c_5) = 0.357$.

DEFINITION 6 (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 $c \in CS$.*

Table 2: MEANINGS OF SYMBOLS USED

Notation	Description
c or c_i	a correspondence
$Pr(c_i)$	the probability of c_i being in the correct matching
Q_{c_i}	the Correspondence Correctness Question (CCQ) w.r.t c_i
m_i	a possible matching
$Pr(m_i)$	the probability that m_i is the correct matching
$Pr(m_i\|A)$	the probability that m_i is the correct matching given answer A from the crowd
$E(\Delta H_{c_i})$	expected uncertainty reduction by publishing Q_{c_i}
S_Q	a set of CCQs
D_A	the domain of the answers of some CCQs
p_A	the probability for each element of D_A
$E(\Delta H_{S_Q})$	expected uncertainty reduction by all the CCQs in S_Q

Again, in the example of Table 1, the CCQ w.r.t c_1, denoted as Q_{c_1}, is a question "Do you think the correspondence (Name, Prof.Name) is correct?"

DEFINITION 7 (PROBLEM STATEMENT). *For two given schemata S and T, given result set RS and probability assignment function Pr generated by some schema matching tools. Let B be the budget of the number of CCQs to be asked to the crowd. Our goal is to maximize the reduction of $H(RS)$ 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 two core functions: (1) Expected Uncertainty Reduction - the expectation of reduction contributed by a single CCQ; (2) Distribution Adjustment - modifying the probability distribution of RS after receiving an answer. We then put these together to develop the *Single CCQ Approach*, a framework to address the uncertainty reduction problem using a sequence of Single CCQ. Finally, we propose efficient algorithms to implement the computations in this approach.

3.1 Formulation

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 the expectation of uncertainty reduction caused by individual CCQs as the measurement. In the following, we provide the formulation of the expected uncertainty reduction in the context of the Single CCQ Approach.

3.1.1 Expected Uncertainty Reduction of Single CCQ

Let Q_c be a CCQ w.r.t an arbitrary correspondence c. Since Q_c is a Yes/No question, we consider Q_c as a random variable following a Bernoulli distribution. Then $Pr(c)$ indicates the probability that Q_c is correct, and $1 - Pr(c)$ that it is incorrect. Equivalently, we have

$$D_A = \{yes, no\}; p_A = (Pr(c), 1 - Pr(c)) \quad (4)$$

Let ΔH_c be the amount of uncertainty reduced in result set RS by Q_c, we have

$$\Delta H_c = H(RS) - H(RS|Q_c)$$
$$= H(RS) - \left(- \sum_{m_i \in RS} Pr(m_i|Q_c) \log Pr(m_i|Q_c) \right)$$

Then, we can compute the expected uncertainty reduction caused by Q_c, denoted by $E(\Delta H_c)$, as

$$E(\Delta H_c) = Pr(c)\{H(RS) + \sum_{m_i \in RS} Pr(m_i|yes)\log Pr(m_i|yes)\}$$
$$+ (1 - Pr(c))\{H(RS) + \sum_{m_i \in RS} Pr(m_i|no)\log Pr(m_i|no)\}$$
$$= Pr(c)\{H(RS) + \sum_{m_i \in RS} (\frac{Pr(m_i)Pr(yes|m_i)}{Pr(c)}\log\frac{Pr(m_i)}{Pr(c)})\}$$
$$+ (1 - Pr(c))$$
$$\{H(RS) + \sum_{m_i \in RS} (\frac{Pr(m_i)Pr(no|m_i)}{1 - Pr(c)}\log\frac{Pr(m_i)}{1 - Pr(c)})\} \quad (5)$$

The uncertainty reduction w.r.t a given Q_c can be computed by Eq 5 provided that we know the values for three parameters: $Pr(c)$, $Pr(yes|m_i)$ and $Pr(no|m_i)$. We see next how to obtain these values.

Computation of $Pr(c)$:By Eq 2, we can compute $Pr(c)$ by summing up the probabilities of possible matchings in which c is included, i.e.

$$Pr(c) = \sum_{m_i \in RS \wedge c \in m_i} Pr(m_i) \quad (6)$$

Computation of $Pr(yes|m_i)$ **and** $Pr(no|m_i)$: Please note that $Pr(yes|m_i)$ ($Pr(no|m_i)$) represents the probability that c is correct (incorrect) given that m_i is a correct matching. As a result, $Pr(yes|m_i)$ and $Pr(no|m_i)$ are either 0 or 1, depending on if c is a correspondence included in m_i.

$$Pr(yes|m_i) = \begin{cases} 1 & c \in m_i \\ 0 & c \notin m_i \end{cases}; Pr(no|m_i) = \begin{cases} 0 & c \in m_i \\ 1 & c \notin m_i \end{cases} \quad (7)$$

Finally, equipped with Eq 6 and Eq 7, the uncertainty reduction w.r.t a given Q_c can be computed by Eq 5.

3.1.2 Adjustment with Crowdsourced Answers

In a crowdsourcing environment, workers may make mistakes. To handle this issue, we must allow for the possibility that any crowdsourced answer could be wrong. The probability of this happening can be estimated by the error rate of the worker.

Running Example: Continuing with the example of Table 1, suppose c_2 is identified as correct by a worker W with confidence 0.8, then we have

$$Pr(m_1|e : c_2 \, is \, answered \, from \, W) = \frac{Pr(m_1)Pr(e|m_1)}{Pr(e)}$$
$$= \frac{Pr(m_1)Pr(W \, is \, correct)}{Pr(c_2)Pr(W \, is \, correct) + (1 - Pr(c_2))Pr(W \, is \, incorrect)}$$
$$= \frac{.45 * .8}{.7 * .8 + .3 * .2} = .58$$

Similarly, we have $Pr(m_2|e) = 0.10$ and $Pr(m_3|e) = 0.32$. The uncertainty of the adjusted possible matchings, the confidence difference between m_1 and m_2 (or m_3), is reduced, and the confidence of selecting $m_1 = 0.58 > 0.5$ is still improved. The entropy is also decreased, to 1.31. This decrease is somewhat less than if the answers were obtained with no error. In short, even imperfect answers can be useful for reducing the uncertainty.

For a given Q_c, answer ans (yes or no), with error rate $P_e \in (0, 1)$, we need to compute $Pr(m_i|ans)$, and replace $Pr(m_i)$ with

Framework 1 Single CCQ

```
1: CONS ← 1 // consumption of the budget
2: Find and publish Q_{c_i} that maximize E(ΔH_c) //(See 3.1.1)
3: while there exists a CCQ in the crowd, we constantly monitor the CCQ
   do
4:     for answer a_i of Q_{c_i}, error rate P_{e_i} do
5:         ∀m_i ∈ RS Adjust the Pr(m_i) //(See 3.1.2)
6:         if CONS < B then
7:             Finding Q_{c_j} maximizing E(ΔH_c) //(See 3.1.1)
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
```

$Pr(m_i|ans)$ for all possible matchings. We assume crowdsourcing workers provide answers independently, so P_e and $Pr(c)$ are independent. Then, we have the functions:

$$Pr(m_i|ans = yes) = Pr(m_i)Pr(ans = yes|m_i)/Pr(ans = yes)$$
$$= \frac{Pr(m_i)Pr(ans = yes|m_i)}{(1 - P_e)Pr(c) + P_e(1 - Pr(c))}$$
$$Pr(m_i|ans = no) = Pr(m_i)Pr(ans = no|m_i)/Pr(ans = no)$$
$$= \frac{Pr(m_i)Pr(ans = no|m_i)}{(1 - P_e)(1 - Pr(c)) + P_e Pr(c)} \quad (8)$$

where $Pr(c)$ is given in Eq 6, $Pr(m_i)$ is defined in Def 3, and $Pr(ans = yes|m_i)$ and $Pr(ans = no|m_i)$ are computed as follows:

$$Pr(ans = yes|m_i) = \begin{cases} 1 - P_e & c \in m_i \\ P_e & c \notin m_i \end{cases};$$
$$Pr(ans = no|m_i) = \begin{cases} P_e & c \in m_i \\ 1 - P_e & c \notin m_i \end{cases} \quad (9)$$

Eq 8 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 ans_1 and ans_2, we have

$$Pr(m_i|ans_1, ans_2) = Pr(m_i|ans_2, ans_1) \quad (10)$$

Eq 10 indicates that the final 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 (line 4), we adjust $Pr(m_i)$ based on the answer and corresponding error rate (line 5), and then generate a new CCQ (line 7&8).

In the framework of Single CCQ, one can see that an important task is to find the CCQ with the highest expectation of uncertainty

reduction as soon as the probability distribution of RS 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 8 (SINGLE CCQ SELECTION (SCCQS)). *Given result set RS, probability assignment function $Pr()$, the Single CCQ Selection Query retrieves a CCQ maximizing the expected uncertainty reduction, ΔH_c.*

3.3 Query Processing of SCCQS

Based on the formulation in Section 3.1, we are able to compute the expected 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 $\mathcal{O}(|RS|^2|CS|)$, 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 expectation of uncertainty reduction is mathematically equivalent to the entropy of the answer of a CCQ. 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 indicates to 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 postive. By Theorem 3.1, we prove that the expected uncertainty reduction $E(\Delta H_c)$ of a correspondence c is equivalent to the entropy of the answer of Q_c. In other words, $E(\Delta H_c)$ is only determined by $Pr(c)$. As a result, searching for the CCQ that maximize $E(\Delta H_c)$ has the complexity decreased to $\mathcal{O}(|RS|.|CS|)$, by computing $Pr(c)$ for each $c \in CS$. In addition, Corollary 3.2 states that we only need to find the correspondence that has probability closest to 0.5, based on the fact that $E(\Delta H_c)$ is a symmetric function of $Pr(c)$, with symmetry axis $Pr(c) = 0.5$ and achieves maximum when $Pr(c) = 0.5$.

THEOREM 3.1. *For correspondence $c \in CS$, c has probability $Pr(c)$ to be correct. Then we have*

$$E(\Delta H_c) = -Pr(c)\log Pr(c) - (1 - Pr(c))\log(1 - Pr(c))$$

PROOF. Please see appendix. □

COROLLARY 3.2. *For any two correspondence $c, c' \in CS$, if $|0.5 - Pr(c)| \geq |0.5 - Pr(c')|$ then $E(\Delta H_c) \geq E(\Delta H_{c'}) \geq 0$. In addition, $E(\Delta H_c) = 1$ if and only if $Pr(c) = 0.5$*

PROOF. Please see appendix. □

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, $E(\Delta H_{c_2}) = -0.7 * log(0.7) - 0.3 * log(0.3) = 0.88$. Now we assume an answer "$a = yes$" is received from a crowdsourcing worker, whose personal error rate is $P_e = 0.2$ (line 4). Then we conduct the adjustment according to Eq 8, and have $Pr(m_1|a) = 0.58$, $Pr(m_2|a) = 0.10$ and $Pr(m_3|a) = 0.32$. 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 6: $Pr(c_1) = 0.68$,$Pr(c_2) = 0.9$,$Pr(c_3) = 1$,$Pr(c_4) = 0.68$,$Pr(c_5) = 0.32$. 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 RS with a binary matrix M_{RS}, where element $e_{ij} = 1(0)$ representing $c_j \in m_i (c_j \notin m_i)$, as shown in follows.

$$M_{RS} = \begin{array}{c} \\ m_1 \\ \vdots \\ m_i \\ \vdots \\ m_{|RS|} \end{array} \begin{array}{c} \begin{array}{cccc} c_1 & c_2 & \dots & c_{|CS|} \end{array} \\ \begin{pmatrix} 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ e_{i1} & e_{i2} & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \end{array} \tag{11}$$

Equipped with this index, we apply a pruning technique derived from Corollary 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 $Pr(m_i)$ if $e_{ij} = 1$. Let $c_{best\_so\_far}$ be the best correspondence so far, with probability $Pr(c_{best\_so\_far})$. Then, let c_j be the current correspondence, and P_{acc} be its accumulated probability after reading some $P(m_i)$, then c_j can be safely pruned if we have

$$P_{acc} - 0.5 \geq |Pr(c_{best\_so\_far}) - 0.5|$$

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)[4]. 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 of 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 first provide the formulation and framework of Multiple CCQs, by extending our results of Single CCQ. Then, we present a complete proof that the **expected uncertainty reduction of a set of CCQs is equivalent to the joint entropy of the CCQs**. Finally, we prove the NP-hardness of the multiple CCQs selection problem, and propose an efficient approximation algorithm with a bounded error.

4.1 Formulating Expected Uncertainty Reduction of Multiple CCQ Approach

For a given set of CCQs of size k - $S_Q = \{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}$, we want to derive the expected uncertainty reduction caused by the aggregation of the answers of these k CCQs. Let D_A and p_A being the domain and probability distribution of the answer respectively. Each possible answer, i.e. an element of D_A, can be seen as an event that some of the k CCQs are correct, while others are incorrect. Then first we have

$$\begin{aligned} &D_A = \{a_i | a_i \subseteq 2^{\{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}} \; and \\ &\forall Q_{c_j} \in a_i \, , c_j \; is \; correct; \; and \; \forall Q_{c_j} \notin a_i \, , c_j \; is \; incorrect\} \\ &p_A = (Pr(a_1), Pr(a_2), ..., Pr(a_{2^k})) \end{aligned} \quad (12)$$

Similar to Eq 5, we are able to to compute the expected uncertainty reduction caused by the S_Q, denoted by $E(\Delta H_{S_Q})$, then

$$\begin{aligned} E(\Delta H_{S_Q}) = \sum_{a_i \in D_A} Pr(a_i) \sum_{m_i \in RS} \{-Pr(m_i) \log Pr(m_i) + \\ \frac{Pr(m_i) Pr(a_i|m_i)}{Pr(a_i)} \log \frac{Pr(m_i) Pr(a_i|m_i)}{Pr(a_i)}\} \end{aligned} \quad (13)$$

Computation of $Pr(a_i)$: $Pr(a_i)$ is equivalent to the possibility that all the correspondences queried by CCQs of a_i are in the correct matching. Then by Eq 3, we have

$$Pr(a_i) = \sum_{m_i \in RS \wedge (\forall c_j \in m_i, Q_{c_j} \in a_i)} Pr(m_i) \quad (14)$$

Computation of $Pr(a_i|m_i)$: Similar to Single CCQ, $Pr(a_i|m_i)$ is either 1 or 0 depending on whether the correspondences queried by a_i (i.e. $Q_{c_j} \in a_i$ in Eq 12) are all in the possible matching m_i. Formally,

$$Pr(a_i|m_i) = \begin{cases} 1 & \forall Q_c \in a_i, c \in m_i \\ 0 & \exists Q_c \in a_i, c \notin m_i \end{cases} \quad (15)$$

4.2 Framework of Multiple CCQ

As shown in Framework 2, the best size-k set of CCQs are initially selected and published, 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 expected uncertainty reduction. Formally, we have following definition:

Framework 2 Multiple CCQ

1: $CONS \leftarrow k$ // consumption of the budget
2: find and publish a set of CCQs - $S_Q = \{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}$ that maximize $E(\Delta H_{S_Q})$//(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 error rate $P_{e_1}, P_{e_2}, ...$ **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_{e_i} **do** //Adjustment
9: $\forall m_i \in RS$ Adjust the $Pr(m_i)$ //(See 3.1.2)
10: **end for**
11: **if** $CONS + k'' <= B$ **then**
12: find a set of CCQs - S'_Q of size $(k' + k'')$ that currently maximize $E(\Delta H_{S'_Q})$ //(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 $E(\Delta H_{S'_Q})$ //(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 9 (MULTIPLE CCQ SELECTION (MCCQS)). *Given result set RS, probability assignment function Pr, and an integer k, the multiple CCQ selection problem is to retrieve a set of k CCQs, denoted by S_Q, such that the expected 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 expected uncertainty reduction. Similar to [17] and [25], 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 expected uncertainty reduction of a CCQ is equivalent to its entropy. 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, we prove that the expectation of uncertainty reduction by a set of CCQs is equivalent to their ***joint entropy*** (denoted by $H(D_A)$), i.e.

$$E(\Delta H_{S_Q}) = H(D_A) = -\sum_{a_i \in D_A} Pr(a_i) \log Pr(a_i)$$

Facilitated with this theorem, we reduce MCCQS to a special case of joint entropy maximization problem.

THEOREM 4.1. *Given a set of CCQs $S_Q = \{Q_{c_1}, Q_{c_2}, ..., Q_{c_k}\}$, the answer has domain*

$$\begin{aligned} D_A = \{a_i | a_i \subseteq 2^{S_Q} \; and \; \forall Q_{c_j} \in a_i \, , c_j \; is \; correct; \\ and \; \forall Q_{c_j} a_i \, , c_j \; is \; incorrect\} \end{aligned}$$

and distribution $p_A = (Pr(a_1), Pr(a_2), ..., Pr(a_{2^k}))$, then we have $E(\Delta H_{S_Q}) = -\sum_{a_i \in D_A} Pr(a_i) \log Pr(a_i)$

PROOF. Please see appendix. □

4.4 NP-hardness of Multiple CCQ Selection

By Theorem 4.1, searching a group of k CCQs with maximal expected 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, [11] 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 $O(|RS||CS|^k)$ complexity, since the searching space (i.e. the number of subsets to select) is always of size $C^k_{|CS|}$.

With the Theorem 4.2, 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 4.2. *The Multiple CCQ Selection is NP-hard.*

PROOF. To reach the proof of Theorem 4.2, it is sufficient to prove the NP-completeness of its decision version, Decision MC-CQS (DMCCQS), i.e. given result set *RS*, probability assignment function Pr, 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. Now we state the special case of DMCCQS by adding the following constraint on RS: for each way of partitioning RS into two subsets S_1 and S_2, there exists a correspondence c such that

$$(\forall m_i \in S_1, c \in m_i) \wedge (\forall m_j \in S_2, c \notin m_i)$$

Equipped with this constraint, we this reduce special case of DM-CCQS 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 $Pr(m_i) = x_i/Sum$. 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 RS can be partitioned into two subsets, each with aggregate probability 0.5. According to the constraint, the exists a correspondence c with $Pr(c) = 0.5$. 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 DM-CCQS when $k = 1, \Delta H = -\log(0.5) = 1$. Since $k = 1$, H_{S_Q} is actually equivalent to H_c (see Eq 5). Then by Corollary 3.2, there exists a correspondence c such that $Pr(c) = 0.5$. Therefore, by the constraint, there is a way to partition RS 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. □

4.5 Approximation Algorithm

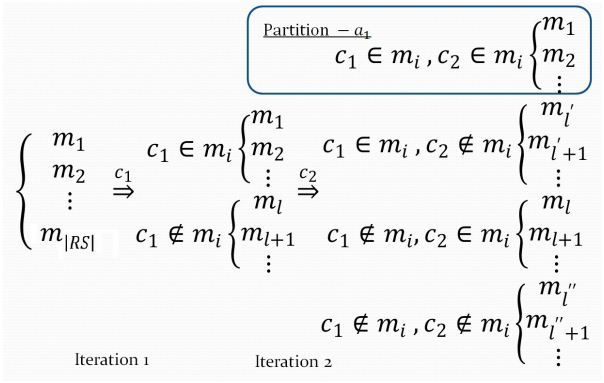

Figure 2: Illustration of RS Partitioning

It is known that the joint entropy of a set of random variables is a *monotone sub-modular function* [11]. And the problem of 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 [10]. Formally, we have the optimization function at the k^{th} iteration:

$$X := \arg\max_{Q_{c_X}} E(\Delta H_{S_Q^{k-1} \cup \{Q_{c_X}\}}) \tag{16}$$

Considering S_Q^{k-1} and Q_{c_X} as two variables, their joint entropy $H(S_Q^{k-1}, Q_{c_X}) = H(S_Q^{k-1}) + H(Q_{c_X}|S_Q^{k-1})$, so we only need to maximize the conditional entropy at each iteration, i.e. $X := \arg\max_{Q_{c_X}} H(Q_{c_X}|S_Q^{k-1})$ and

$$H(Q_{c_X}|S_Q^{k-1}) = -\sum_{a_i \in D_{S_Q^{k-1}}} Pr(a_i)\{Pr(yes|a_i)\log Pr(yes|a_i) + Pr(no|a_i)\log Pr(no|a_i)\} \tag{17}$$

Eq 17 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 $a_i \in D_{S_Q^{k-1}}$. We aim to find the k^{th} 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 $P(a_i)$ associated with each partition.

As a result, the computation of $H(Q_{c_X}|S_Q^{k-1})$ for each candidate correspondence is simply traversing the list of partitions. Note the number of partitions is at most $|RS|$ (i.e. each partition has only one possible matching), so the overall complexity is upper bounded by $O(k|RS||CS|)$. 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 $Pr(a_1), Pr(a_2), ..., Pr(a_n)$ respectively. As follows, we present four pruning rules.

PRUNING RULE 4.3. *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 4.3 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 4.4. *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 4.3, Pruning rule 4.4 indicates the condition that the correctness of c can be determined by selected correspondences.

Next, we introduce Pruning Rule 4.5 and 4.6, which derives two non-trivial upper bounds, which enable effective pruning.

PRUNING RULE 4.5. *Let $best\_so\_far$ be the best value of Eq 17 so far for the current iteration, then for the correspondence c, let $a_1, a_2, ..., a_m$ be the partitions c already traversed, then let*

$$H_0 = -\sum_{i=1}^{m} Pr(a_i)\{Pr(yes|a_i)\log Pr(yes|a_i) + Pr(no|a_i)\log Pr(no|a_i)\}$$

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

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

PROOF. For the rest of partitions $a_{m+1}, a_{m+2}, ..., a_n$, the optimal situation is they are all perfectly bisected, that it $\forall i \in [m+1, n], Pr(yes|a_i) = Pr(no|a_i) = 0.5$.

Therefore, their contribution to the optimization function has a upper bound $\sum_{j=m+1}^{n} Pr(a_j)\log\frac{Pr(a_j)}{2}$ □

PRUNING RULE 4.6. *Let $best\_so\_far$ be the best value of Eq 17 so far for the current iteration. For a correspondence c, let $H(Q_c|S_Q^{k-2})$ be the conditional entropy computed from a previous iteration. Then, c can be pruned for the current iteration if*

$$H(Q_c|S_Q^{k-2}) \leq best\_so\_far$$

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

$$H(S_Q^{k-2}, Q_c) - H(S_Q^{k-2}) \geq H(S_Q^{k-1}, Q_c) - H(S_Q^{k-1})$$

and equivalently, $H(Q_{c_X}|S_Q^{k-2}) \geq H(Q_{c_X}|S_Q^{k-1})$, which completes the proof. □

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.

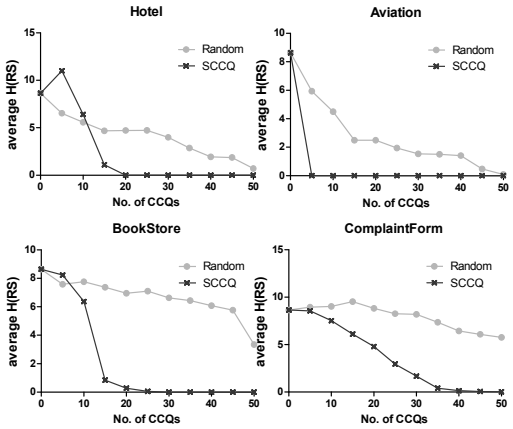

Figure 3: Single CCQ v.s. Random - Simulation

5.1 Experimental Setup

We adopt the schema matching tool OntoBuilder [7, 5], which is one of the leading tools for schema matching. In particular, we conduct our experiments on four 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, the schemata are parsed into xml schemata, and the 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, 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 [5].

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 error rate $P_e \in (0, 0.5)$ following an uniform distribution. Third, given a CCQ, we generate the correct yes-no answer with probability $(1 - P_e)$ (i.e. generate the wrong answer with probability P_e), and then return the answer and P_e as the inputs for adjustment (Section 3.1.2).

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.

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

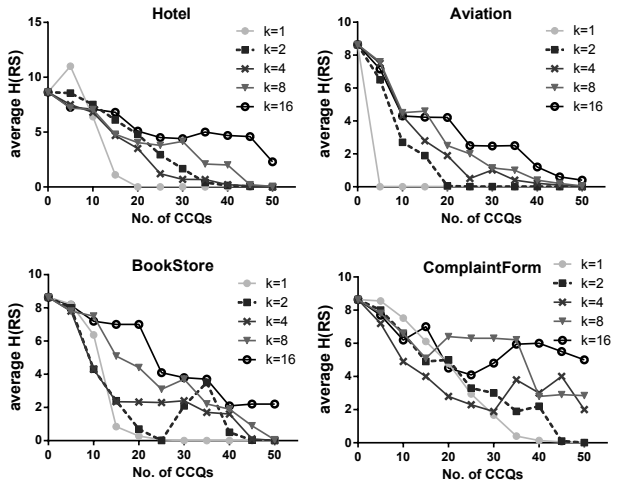

Figure 4: Multiple CCQ with different k - Simulation

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, 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 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 $|CS|$ correspondences, and when $k = |CS|$, 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

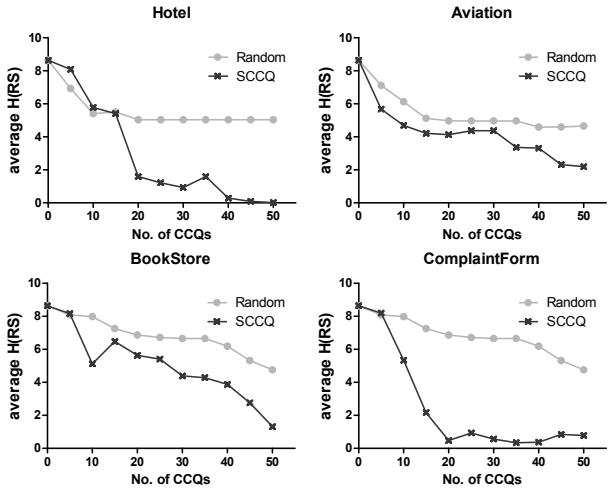

Figure 5: Single CCQ v.s. Random - on Amazon Mechanical Turk

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.

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 "_R" represent the data obtained from the simulation and the real-world implementation on AMT, respectively. In the simulation, the preci-

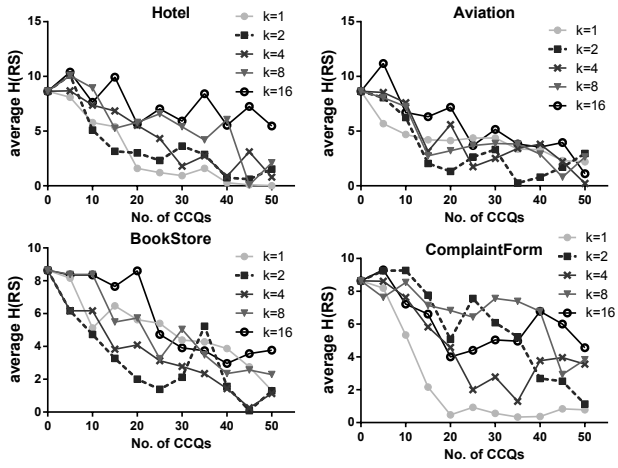

Figure 6: Multiple CCQ with different k on Amazon Mechanical Turk

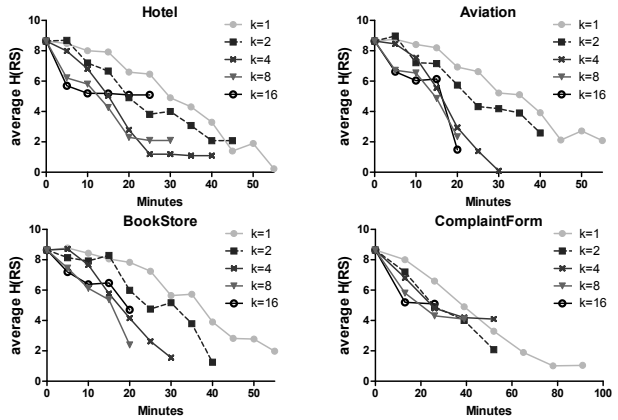

Figure 7: Time Cost with different k on Amazon Mechanical Turk

sion and recall are almost 100%. In the real-world implementation, the performance is less perfect, but still 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.

6. RELATED WORK

6.1 Uncertainty in Schema Matching

The model of possible matching, namely "probabilistic schema mappings", was first introduced in [3]. In their work, algorithmic approaches generate a set of matchings between two schemata,

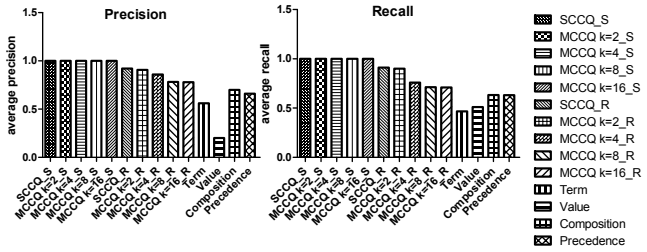

Figure 8: Data Quality with Budget Constraint- Precision & Recall

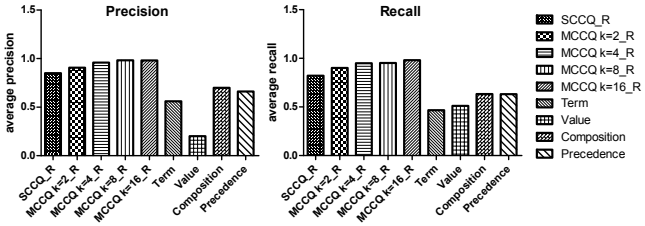

Figure 9: Data Quality with Time Constraint - Precision & Recall

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. [21] 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 [6]. In detail, [6] generates top-k schema matchings by combining the matching 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 [8] as a core foundation for answering queries in a data integration system with uncertainty. Gal [5] used the top-K schema mappings from a semi-automatic matchers to improve the quality of the top mapping. [3] [8] and [19] 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. [3] 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 [23, 24, 9].

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 [2] as a survey). In general, [4] proposed a query processing system using microtask-based crowdsourcing to answer queries. In [16], a declarative query model is proposed to cooperate with standard relational database operators. As a typical application related to data integration, [25] utilized a hybrid human-machine approach on the problem of entity resolution.

To our best knowledge, [13] is only previous work engaging crowdsourcing into schema matching. In particular, [13] 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 [13] is threefold: (1) From the conceptual level, "crowd" in [13] 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 [13] 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 [22]. In particular, [15, 26] 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. DISCUSSION 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 the CCQs. In addition, we prove the NP-hardness of the problem of Multiple CCQ Selection, and design an $(1 + \epsilon)$ approximation algorithm, based on its sub-modular nature.

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.

One challenge with schema matching is that while many parts of a schema matching are obvious to any human, often the meaning of some portions of the data is not clear to a non-domain expert. Therefore, investigating the difficulties of CCQs and the trustworthiness of crowd-sourced answers may further help reducing the matching uncertainty. In future work, we will try to fold this into a more realistic and more complete model of worker error rates. We have also ignored more complex worker incentives, including boredom and thoughtless answer selection, as being standard across crowdsourcing platforms. While this orthogonality assumption is a good approximation for now, in future work we will incorporate it into the main algorithm.

8. ACKNOWLEDGMENTS

This work is supported in part by the Hong Kong RGC Project M-HKUST602/12, National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200, Microsoft Research Asia Grant, NSF grant IIS-1250880, and Huawei Noahs ark lab project HWLB06-15C03212/13PN.

9. REFERENCES

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

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

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

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

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

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

[7] 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.

[8] 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.

[9] J. Huang, L. Antova, C. Koch, and D. Olteanu. Maybms: a probabilistic database management system. In *SIGMOD Conference*, 2009.

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

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

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

[13] R. McCann, W. Shen, and A. Doan. Matching schemas in online communities: A web 2.0 approach. In *ICDE* [13], pages 110–119.

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

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

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

[17] 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.

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

[19] Y. Qi, K. S. Candan, and M. L. Sapino. Ficsr: *f*eedback-based *i*nconsistency *r*esolution and query processing on misaligned data sources. In *SIGMOD Conference*, pages 151–162, 2007.
[20] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
[21] A. D. Sarma, X. Dong, and A. Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD Conference*, pages 861–874, 2008.
[22] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.
[23] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu. Mining frequent itemsets over uncertain databases. *PVLDB*, 5(11):1650–1661, 2012.
[24] Y. Tong, L. Chen, and B. Ding. Discovering threshold-based frequent closed itemsets over probabilistic data. In *ICDE*, pages 270–281, 2012.
[25] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.
[26] L. Zhao, G. Sukthankar, and R. Sukthankar. Robust active learning using crowdsourced annotations for activity recognition. In *Human Computation*, 2011.

APPENDIX

Proof of Theorem 3.1

PROOF. First, we simplify Eq 5

$$\begin{aligned}
&E(\Delta H_c) = H(RS)+\\
&Pr(c)\sum_{m_i\in RS}(\frac{Pr(m_i)Pr(yes|m_i)}{Pr(c)}\log\frac{Pr(m_i)}{Pr(c)})+\\
&(1-Pr(c))\sum_{m_i\in RS}(\frac{Pr(m_i)Pr(no|m_i)}{1-Pr(c)}\log\frac{Pr(m_i)}{1-Pr(c)})\\
&= H(RS)+\sum_{m_i\in RS}(Pr(m_i)Pr(yes|m_i)\log\frac{Pr(m_i)}{Pr(c)})\\
&+\sum_{m_i\in RS}(Pr(m_i)Pr(no|m_i)\log\frac{Pr(m_i)}{1-Pr(c)})
\end{aligned}\tag{18}$$

Now we substitute Eq 7 into Eq 18, then

$$\begin{aligned}
&E(\Delta H_c) = H(RS)+\sum_{m_i\in RS\wedge c\in m_i}\left(Pr(m_i)\log\frac{Pr(m_i)}{Pr(c)}\right)\\
&+\sum_{m_j\in RS\wedge c\notin m_j}\left(Pr(m_j)\log\frac{Pr(m_j)}{1-Pr(c)}\right)\\
&= H(RS)+\\
&\sum_{m_i\in RS\wedge c\in m_i}(Pr(m_i)\log Pr(m_i)-Pr(m_i)\log Pr(c))+\\
&\sum_{m_j\in RS\wedge c\notin m_j}(Pr(m_j)\log Pr(m_j)-Pr(m_j)\log(1-Pr(c)))\\
&= H(RS)+\sum_{m_i\in RS\wedge c\in m_i}Pr(m_i)\log Pr(m_i)\\
&-\sum_{\substack{m_i\in RS\\ \wedge c\in m_i}}Pr(m_i)\log Pr(c)+\sum_{\substack{m_j\in RS\\ \wedge c\notin m_j}}Pr(m_j)\log Pr(m_j)\\
&-\sum_{m_j\in RS\wedge c\notin m_j}Pr(m_j)\log(1-Pr(c))
\end{aligned}$$

Note that, $H(RS)$ is the uncertainty of entire result set, that is $H(RS) = -\sum_{m_i\in RS\wedge\ c\in m_i}Pr(m_i)-\sum_{m_j\in RS\wedge\ c\notin m_j}Pr(m_j)$ so, we have

$$\begin{aligned}
&-E(\Delta H_c)\\
&=\sum_{\substack{m_i\in RS\\ \wedge c\in m_i}}(Pr(m_i)\log Pr(c))+\sum_{\substack{m_i\in RS\\ \wedge c\notin m_i}}(Pr(m_j)\log(1-Pr(c))\\
&=\log Pr(c)\sum_{\substack{m_i\in RS\\ \wedge c\in m_i}}Pr(m_i)+\log(1-Pr(c))\sum_{\substack{m_i\in RS\\ \wedge c\notin m_i}}Pr(m_j)
\end{aligned}$$

Finally, by Eq 2, we have $E(\Delta H_c) = -Pr(c)\log Pr(c)-(1-Pr(c))\log(1-Pr(c))$ □

Proof of Corollary 3.2

PROOF. let $\frac{dE(\Delta H_c)}{dPr(c)} = 0$, then we have

$$-\log Pr(c)/(1-Pr(c)) = 0 \Leftrightarrow Pr(c) = 0.5$$

$E(\Delta H_c)$ is a symmetric function of $Pr(c)$, with symmetry axis $Pr(c) = 0.5$. Besides, the function achieves maximum $E(\Delta H_c) = 1$ when $Pr(c) = 0.5$, and is monotonic on $[0, 0.5]$ (increasing) and $[0.5, 1]$ (decreasing). □

Proof of Theorem 4.1

PROOF. First,we rewrite Eq 13, and $H(RS) = -\sum_{m_i\in RS}Pr(m_i)\log Pr(m_i)$

$$\begin{aligned}
&E(\Delta H_{S_Q}) = \sum_{a_i\in D_A}\{-\sum_{m_i\in RS}Pr(a_i)Pr(m_i)\log Pr(m_i)+\\
&\sum_{m_i\in RS}Pr(m_i)Pr(a_i|m_i)\log\frac{Pr(m_i)Pr(a_i|m_i)}{Pr(a_i)}\}\\
&=\sum_{a_i\in D_A}\{Pr(a_i)\Delta H(RS)+\\
&\sum_{m_i\in RS}Pr(m_i)Pr(a_i|m_i)\log[Pr(m_i)Pr(a_i|m_i)]-\\
&\sum_{m_i\in RS}Pr(m_i)Pr(a_i|m_i)\log Pr(a_i)\}
\end{aligned}$$

Then, we substitute Eq 15,

$$\begin{aligned}
&E(\Delta H_{S_Q}) = \sum_{a_i\in D_A}\{Pr(a_i)\Delta H(RS)+\\
&\sum_{m_i\in RS\wedge(\forall Q_c\in a_i, c\in m_i)}Pr(m_i)\log Pr(m_i)-\\
&\sum_{m_i\in RS\wedge(\forall Q_c\in a_i, c\in m_i)}Pr(m_i)\log Pr(a_i)\}\\
&=\left(\sum_{a_i\in D_A}\Pr(a_i)\right)\Delta H(RS)+\\
&\sum_{a_i\in D_A}\sum_{m_i\in RS\wedge(\forall Q_c\in a_i, c\in m_i)}Pr(m_i)\log Pr(m_i)-\\
&\sum_{a_i\in D_A}\left(\sum_{m_i\in RS\wedge(\forall Q_c\in a_i, c\in m_i)}Pr(m_i)\right)\log Pr(a_i)\\
&=\Delta H(RS)+\sum_{m_i\in RS}Pr(m_i)\log Pr(m_i)-\\
&\sum_{a_i\in D_A}\left(\sum_{m_i\in RS\wedge(\forall Q_c\in a_i, c\in m_i)}Pr(m_i)\right)\log Pr(a_i)\\
&=-\sum_{a_i\in D_A}\left(\sum_{m_i\in RS\wedge(\forall Q_c\in a_i, c\in m_i)}Pr(m_i)\right)\log Pr(a_i)
\end{aligned}$$

Finally by Eq 14, $E(\Delta H_{S_Q}) = -\sum_{a_i\in D_A}Pr(a_i)\log Pr(a_i)$, which completes the proof. □