MaC: A Probabilistic Framework for Query Answering with Machine-Crowd Collaboration

Chen Jason Zhang, Lei Chen, Yongxin Tong
Hong Kong University of Science and Technology
Hong Kong, China
czhangad@cse.ust.hk, leichen@cse.ust.hk, yxtong@cse.ust.hk

ABSTRACT

The popularity of crowdsourcing has recently brought about brand-new opportunities for engaging human intelligence in the process of data analysis. Most existing works on crowdsourcing have developed sophisticated methods to utilize the crowd as a new kind of processor, a.k.a. Human Processor Units (HPU). In this paper, we propose a framework, called MaC, to combine the powers of both CPUs and HPUs. In order to build MaC, we need to tackle the following two challenges: (1) HIT Selection: Selecting the "right" HITs (Human Intelligent Tasks) can help reducing the uncertainty significantly and the results can converge quickly. Thus, we propose an entropy-based model to evaluate the informativeness of HITs. Furthermore, we find that selecting HITs has factorial complexity and the optimization function is non-linear, thus, we propose an efficient approximation algorithm with a bounded error. (2) Uncertainty Management: Crowdsourced answers can be inaccurate. To address this issue, we provide effective solutions in three common scenarios of crowdsourcing: (a) the answer and the confidence of each worker are available; (b) the confidence of each worker and the voting score for each HIT are available; (c) only the answer of each worker is available. To verify the effectiveness of the MaC framework, we built a hybrid Machine-Crowd system and tested it on three real-world applications - data fusion, information extraction and pattern recognition. The experimental results verified the effectiveness and the applicability of our framework.

1. INTRODUCTION

The recent success of crowdsourcing on various human-intrinsic tasks drives people to apply crowdsourcing to wider application areas. Existing works treat the crowd as a new kind of processors, a.k.a. Human Processing Units (HPU). As a consequence, a query executed on an HPU is called an HPU-based task. With respect to cost-efficiency, the optimization of an HPU-based task focuses on how to decompose the task into a small number of micro-tasks, namely *Human Intelligent Tasks (HITs)*, that are easy for crowdsourcing workers.

Most existing works treat the crowd as the sole information source for the queries. In other words, machines (i.e. CPUs) manage only the construction and publication of HITs, but do not informatively contribute to the answer. However, in many real-world applications, such as information extraction, data fusion and pattern recognition, the same queries handled by HPUs can also be answered by machine-alone systems, usually associated with learning-based techniques. This indicates that the requested information can at least be partially provided by machines. In general, the latency of an HPU is much higher than that of a CPU, and an HPU may lead to high monetary cost (e.g. Amazon Mechanical Turk, CrowdFlower). Thus, it is essential to combine the power of both methods to accomplish the tasks effectively. Motivated by this, in this paper, we propose a framework, namely MaC, to combine the power from both the crowd and the machines.

For each HIT, both machines and crowds can be seen as information sources. Whether using HPUs or CPUs, the major problem associated with the results is the data uncertainty. For results obtained from machines, the inherent uncertainty is due to the incapabilities of the current solutions in recognizing human-intuitive semantics via given data representation (e.g. images, nature language). For results obtained from the crowd, any uncertainty is due to sloppy workers, spammers, or the incapability of workers for domain-specific tasks. Since uncertainty is inherited from both the machine and the crowd, it is desirable to maintain the uncertainty in order to avoid information loss. Now we show an example of how to combine the machine and the crowd to improve the accuracy of the answer.

Running Example: suppose we have textual data '51A Hayward East New York', and the query is to extract information for three attributes - 'House no', 'Area' and 'City'. With a machine-only tool of information extraction, we have the results as shown in Table 1. Due to a lack of capability to fully understand human semantics, four possible results are generated , each of which is associated with a probability of being the correct answer. As a result, the best answer is o_1*, but with a confidence of only* 0.6*. On the other hand, assume we have a crowd* C *with accuracy* 0.75*, then if we use the crowd alone to answer this query, say with a HIT 'what are House no, Area and City of the given textual data', the crowdsourced answer would have a confidence of only* 0.75*.*

In the above example, we can observe that using the machine or crowd alone, would generate query answers with a confidence of 0.6 and 0.75, receptively. As pointed out in paper [11], even with simple tasks such as labelling, the quality of normal workers (not sloppy workers or spammers) is around 75%. For complex tasks such as information extraction, the accuracy of the crowd may be worse. In other words, 75% is already an optimistic estimation of the crowd accuracy for complex tasks.

Now we demonstrate how the collaboration between the machine and crowd is going to lead to a better result than just using either

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661880 .

Location address of T_1: 51A Hayward East New York

Xid	House no	Area	City	$Pr(t_i)$
o_1	51A	Hayward East	New York	0.6
o_2	51	Hayward	New York	0.3
o_3	51A	Hayward East	York	0.05
o_4	51	East	York	0.05

Table 1: running example - uncertain data generated by an information extraction tool

ID	HIT content
HIT1	Is the House no 51?
HIT2	Is the House no 51A?
HIT3	Is the Area Hayward?
HIT4	Is the Area Hayward East?
HIT5	Is the Area East?
HIT6	Is the City New York?
HIT7	Is the City York?

Table 2: candidate HITs

machine or crowd alone. For a given human-intrinsic query, we first apply machine-based systems to generate possible outcomes (answers) as well as probability distribution; then we selectively pose simple 'yes-no' questions to the crowd in order to reduce the answer uncertainty, hence improving the data quality. Continued on the running example, we ask the crowd a much easier HIT - "Is the House No 51A ?". Assuming the HIT is answered yes by crowd C with accuracy 0.75, then we have

$$Pr(o1|e : House\ No = 51A) = \frac{Pr(o1)Pr(e|o1)}{Pr(e)}$$
$$= \frac{Pr(o1)Pr(C\ is\ correct)}{Pr(o1)Pr(C\ is\ correct) + (1 - Pr(o1))Pr(C\ is\ incorrect)}$$
$$= \frac{0.75 * 0.6}{0.75 * 0.6 + 0.25 * 0.4} = 0.82 > 0.75 \quad (1)$$

From the above results, we can see that, the collaboration between the machine and the crowd generates a result with a confidence 0.82, which is higher than using machine or crowd alone. In general, if we can combine the results from machines and crowds wisely, the combined results could be better than the results from a single source. In our framework, we do not have explicit constraints for the uncertainties of the crowd, and how to manage the uncertainties are thoroughly studied in Section 4.

However, it is not trivial to propose a general framework for this collaboration, which requires the following challenges to be addressed:

(1) **HIT Selection:** Since information from humans (or the crowd) is very expensive, it is valuable to maximize the utility of the questions in order to improve the overall cost-efficiency. Therefore, we would like to investigate the following question: **which HITs should be published next based on current knowledge**. Clearly, it is non-trivial to design algorithmic approaches to select and construct such HITs.

(2) **Uncertainty Management:** The crowdsourced answers can be inaccurate. How should the accuracy of crowdsourced answers be estimated? Explicitly, for a given answer, we investigate how to compute the probability of its correctness.

In this paper, we proposed solutions to address the above two challenges and made the following contributions.

- In Section 2, we propose the novel framework, namely MaC, to help the machine collaborate with the crowd. In Section 3, we formally define the HIT Selection problem. Specifically, we propose an entropy-based model to evaluate the informativeness of HITs, and provide an efficient $(1+\epsilon)$ approximation algorithm. In Section 4, we consider the uncertainty of crowdsourced answers in three common scenarios. In each of them, we provide efficient and effective algorithms to estimate the posterior probability of HITs. Our solutions in particular assume that each worker from the crowd is an independent information source with a confidence. Thus, each answer from a worker of confidence p_i is a random variable following the Bernoulli distribution, with probability p_i to be correct.

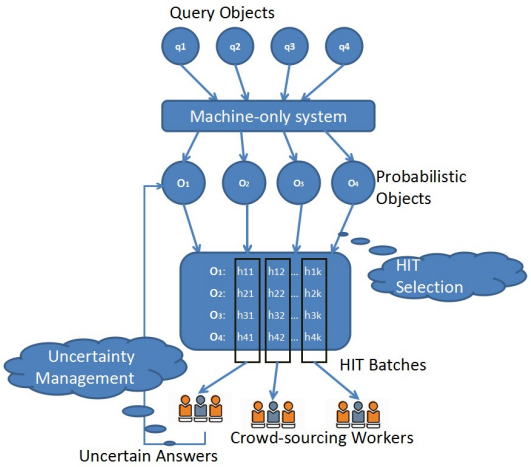

Figure 1: MaC Framework Overview

- In Section 5, we conduct extensive experiments with three real-world applications. The experimental results demonstrate significant improvements compared to machine-only approaches.

In addition, we discuss the related work in Section 6. The overall conclusion and possible future works are presented in Section 7.

2. MAC FRAMEWORK AND BASIC DEFINITIONS

In this section, we illustrate the MaC framework that collaborates machines with crowds, and provide formal definitions of some core concepts. The MaC framework is depicted in Figure 1. In general, MaC processes a set of analytic queries (query objects) with the following steps: (1) through some machine-only system, a set of possible outcomes and their distributions are provided for each query object, namely the "probabilistic object". (2) a number of HITs are constructed based on the machine-generated results (i.e. HIT selection problem). (3) HITs are batched as small groups and published to the crowd. (4) Based on crowdsourced answers (with uncertainty), the distributions are adjusted, and steps (2) to (4) are repeated if the budget is not exceeded. To be clear, we explain the MaC framework from top to bottom in the rest of this section.

2.1 Query Objects and Probabilistic Objects

As shown in Figure 1, the original input of the MaC framework is a set of *query objects*, which are human-intuitive queries to be processed on the machines or the crowd. Formally, we give the formal definition of query object, which captures a wide range of real-world tasks.

DEFINITION 2.1 (QUERY OBJECT). *A Query Object q is an analytical task with the following two characteristics:*

(1) The possible outcomes of q and its probability distribution can be provided by some machine-only systems.

(2) q is an easy and intuitive task for crowdsouring workers.

In the MaC framework, the query objects are first processed by a machine-only system. In the running example, the given image of

poker card is a typical query object, and the recognition system is corresponding to the machine-only system in the MaC framework.

For each query object, a set of possible outcomes are obtained. We call such a set of possible outcomes a *probabilistic object*. Since it is a fundamental concept for the later sections, we formally define this term as follows.

DEFINITION 2.2 (PROBABILISTIC OBJECT). *A Probabilistic Object O is a set of possible outcomes of a query object, described by a collection of attributes $O = \{A_1, ..., A_n\}$, with A_i taking value from a finite space $\Omega(A_i) = \{a_{i1}, .., a_{i|\Omega(A_i)|}\}$ with probabilities $\{Pr_i(a_{i1}), ..., Pr_i(a_{i|\Omega(A_i)|})\}$ respectively. For each attribute A_i, we have $\sum_{j=1}^{|\Omega(A_i)|} Pr_i(a_{ij}) = 1$.*

A probabilistic object depicts the partial information derived from computer-alone systems. In the running example, for the given textual data, we have probabilistic object containing three variables, i.e. House no (A_1) ,Area (A_2) and City (A_3). Their sample spaces are also well defined according to Table 1. For instance, $\Omega(A_1) = \{a_{11} = 51A, a_{12} = 51\}$ and $Pr(A_1 = a_{12}) = Pr_1(a_{12}) = 0.3 + 0.05 = 0.35$.

2.2 HIT Selection and Batching

After probabilistic objects are obtained, we aim to further reduce the data uncertainty via publishing HITs. Then the question is: *which kind of HITs should be asked?*

It is well-known that crowdsourcing works best when tasks can be broken down into very simple pieces. An entire machine-generated outcome may be too large a grain for a crowd - each crowd worker may have small quibbles with a given outcome, so that asking the crowd to directly support or reject each outcome may get mostly negative answers, with each worker declaring it less than perfect. In other words, determining the correctness of an entire machine-generated outcome is too difficult for crowdsourcing workers. On the other hand, asking open-ended questions is not recommended for a crowd, because it may be difficult to integrate multiple suggestions of heterogeneous semantics. As a result, we propose to have the question broken down into *per-attribute tasks*, which is formally defined in Definition 2.3. This kind of much simpler questions, in most applications, can be answered with a simple yes or no. As shown in Section 5, Yes/No questions are in fact useful in a wide range of applications.

DEFINITION 2.3 (HIT AND HIT SPACE). *For a given probabilistic object $O = \{A_1, ..., A_n\}$, a HIT is defined as an YES/NO question, in the form of "Does $A_i = a_{ij}$?". So, the HIT space of O includes totally $\sum_{i=1}^{n} |\Omega(A_i)|$ HITs. We denote HIT "Does $A_i = a_{ij}$?" by h_{ij}. Hence h_{ij} follows Bernoulli distribution with $Pr(h_{ij}) = Pr_i(a_{ij})$ to have "yes" as ground truth, and $1 - Pr(h_{ij})$ to have "no".*

In the running example, Table 1 lists all the HITs in the HIT space of the probabilistic object.

For each probabilistic object, an important optimization problem is to find the best k HITs. This is known as the "HIT selection problem", which will be thoroughly discussed in Section 3.

As a result of HIT selection, we have k HITs extracted from each probabilistic object. Clearly, the HITs from the same probabilistic object are generally correlated. In other words, one worker may provide correlated answers for HITS of the same probabilistic object. Since each HIT generates a unit of cost, asking correlated HITs to the same worker would be **cost-inefficient**.

In the running example, if the answer of HIT1 is answered ('yes' or 'no') by a worker, then we can infer that what he/she would answer for HIT2, since HIT1 and HIT2 are exclusive. So asking both HIT1 and HIT2 to the same worker does not make any difference than only asking one of them, but would lead to two units of cost.

The above intuition indicates that, the most economical way of task publishing is to ask each user a set of independent HITs. In MaC framework, we create and publish batches of HITs with the following definition.

DEFINITION 2.4 (HIT BATCH). *A HIT batch includes multiple HITs, each of which is generated from an distinct probabilistic object. So the HITs within the same batch are independent. In addition, when the batches are published, each worker is only allowed to answer HITs within one batch.*

2.3 Uncertainty Management and Utilization

After the crowdsourced answers are available, we need to manage their uncertainty, as discussed in Section 1. Since there are different paradigms of crowdsourcing, in Section 4, we develop different methods of uncertainty management based on the accessible information. For a HIT h_{ij}, the output of uncertainty management is the posterior probability after obtaining some answers Ans, i.e. $Pr(h_{ij}|Ans)$.

In the following, we apply the updated value $Pr(h_{ij}|Ans)$ to adjust the probability distribution of O. Recall that h_{ij} is the yes/no question "Does $A_i = a_{ij}$?", which means a_{ij} is the ground truth value of A_i with prior probability $Pr(h_{ij})$. Hence, for each possible outcome o of the probabilistic object O, with original probability $Pr(O = o)$, if $O = o$ entails $A_i = a_{ij}$, i.e. o is consistent with the answer, then we have $Pr(h_{ij}|O = o) = 1$. In the running example, given o_1 being the correct outcome, then the probability of "House no = 51A" becomes 1. As a result, we have

$$Pr(O = o|Ans) = Pr(O = o)Pr(h_{ij}|O = o)/Pr(h_{ij}|Ans)$$

$$= Pr(O = o)/Pr(h_{ij}|Ans)$$

On the other hand, if $O = o$ entails $A_i \neq a_{ij}$, we similarly have

$$Pr(O = o|Ans) = Pr(O = o)/\{1 - Pr(h_{ij}|Ans)\}$$

Please note that, the above computation is independent of the sequence of HITs. Therefore, the HIT answers of a probabilistic object would be collected from different batches, and utilized for probability adjustment, one after another.

As a result, the probability distributions are adjusted by the crowdsourced answers. If the budget is not exceeded, and the quality of the objects are unsatisfactory, the work-flow can be iteratively executed. Therefore, the computation of $Pr(h_{ij}|Ans)$ is conducted with at each iteration. Please note that, at each iteration, $Pr(h_{ij}|Ans)$ is computed based on all the answers received from previous iterations, rather than the current iteration only. This is because the essence of uncertainty management is statistical estimation, in which large sample size is preferred.

3. HIT SELECTION

We now formally define the first problem considered in this paper, namely the *Multiple HIT Selection Problem*. The input to our problem is a probabilistic object, and we need to model the importance of HITs and find the most important ones efficiently.

3.1 Problem Formulation

In this subsection, we give the definition of Multiple HIT Selection (MHS) problem in the context of probabilistic objects.

From the perceptive of uncertainty, we are trying to reduce the uncertainty of the given probabilistic object O, by requesting information from the crowd via HITs. Therefore, we model the importance of a HIT by its expected reduction of the uncertainty of O.

In order to construct this model, we first define the uncertainty of probabilistic object with Shannon Entropy.

DEFINITION 3.1 (OBJECT UNCERTAINTY). *Given a probabilistic object $O = \{A_1, ..., A_n\}$, let all the possible deterministic outcomes of O be $\Omega(O) = \{o_1, ..., o_{|\Omega(O)|}\}$, with probabilities $\{Pr(o_1), ..., Pr(o_{|\Omega(O)|})\}$ respectively. Then we measure the uncertainty of O with Shannon Entropy -*

$$H(O) = -\sum_{l=1}^{|\Omega(O)|} Pr(o_l) \log Pr(o_l)$$

The uncertainty of a probabilistic object is determined by the its marginal distribution. It reflects the information provided by the machine. In the running example, from the information extraction tool, the four possible labeling results (instances) are enumerated for the textual data (probabilistic object O), then we have the its uncertainty computed as: $H(O) = -(0.6 \log 0.6 + 0.3 \log 0.3 + 0.05 \log 0.05 + 0.05 \log 0.05) = 0.42$.

Since we aim to reduce the uncertainty of a given probabilistic object, we define the utility of a set of HITs based its expected uncertainty reduction.

DEFINITION 3.2 (UTILITY FUNCTION). *Given a probabilistic object O and a set of selected HITs $S_h = \{h_1, h_2, ..., h_{|S_h|}\}$, we define the utility of S_h by the expected uncertainty reduction of O, i.e.*

$$U(S_h) = H(O) - H(O|S_h)$$

$$= \sum_{s \in D_h} Pr(s)\{H(O) - H(O|S_h = s)\}$$

where D_h is all the possible outcomes of S_h. Formally, we have $D_h = \{s | s \in 2^{S_h}\ s.t.\ \forall h \in S_h,\ h\ has\ ground\ truth\ yes\ if\ and\ only\ if\ h \in s\}$.

Continued on the running example, we demonstrate the computation of utility of $S_h = \{HIT1, HIT4\}$ from Table 2. Based on the given data, we have $Pr(HIT1 = yes, HIT4 = yes) = 0, H(O|HIT1 = yes, HIT4 = yes) = 0; Pr(HIT1 = yes, HIT4 = no) = 0.35, H(O|HIT1 = yes, HIT4 = no) = 0.85; Pr(HIT1 = no, HIT4 = yes) = 0.65, H(O|HIT1 = no, HIT4 = yes) = 0.12;\ Pr(HIT1 = no, HIT4 = no) = 0, H(O|HIT1 = no, HIT4 = no) = 0$. Since $H(O) = 1.3$, we have the utility of S_h: $U(S_h) = 0 * (1.3 - 0) + 0.35 * (0.42 - 0.85) + 0.65 * (0.42 - 0.12) + 0 * (1.3 - 0) = 0.0445$.

Equipped with the above utility function, we are now ready to provide the problem statement of Multiple HIT Selection.

PROBLEM STATEMENT 1 (MULTIPLE HIT SELECTION). *For a probabilistic object O, there is a table $\Omega(O)$ containing its all possible deterministic outcomes, each of which is associated with probability to be true. Given budget of HITs k, we aim to find k HITs that maximize the utility function.*

3.2 Approximation Algorithm

3.2.1 Problem Reduction

One can see that the optimization problem is non-linear, due to the Shannon entropy. In addition, the optimization is essentially to find k HITs from the HIT space, so the searching space is of complexity $O(k!)$. However, we found that the utility of a set of HITs is mathematically equivalent to their joint entropy. As a result, the utility function is a sub-modular function of the set of HITs. We formally introduce this fact with the following theorems.

THEOREM 3.1 (EQUIVALENCY). *Given a probabilistic object O and a set of HITs $S = \{h_1, ...h_k\}$, the utility of S is equal to the joint entropy (i.e. the entropy of their joint distribution) of $h_1, ...h_k$, each of which is a discrete random variable following Bernoulli distribution. So we have*

$$U(S) = H(h_1, ..., h_k) \tag{2}$$

PROOF. *From Definition 3.2, first we have*

$$\begin{aligned} U(S_h) &= \sum_{s \in D_h} Pr(s)H(O) - \sum_{s \in D_h} Pr(s)H(O|S_h = s) \\ &= H(O) - \sum_{s \in D_h} Pr(s)H(O|S_h = s) \\ &= H(O) + \sum_{s \in D_h} Pr(s) \sum_{l=1}^{|\Omega(O)|} Pr(o_l|s) \log Pr(o_l|s) \\ &= H(O) + \sum_{s \in D_h} \sum_{l=1}^{|\Omega(O)|} Pr(s)Pr(o_l|s) \log Pr(o_l|s) \end{aligned} \tag{3}$$

By Bayes' theorem, $Pr(o_l|s) = Pr(o_l)Pr(s|o_l)/Pr(s)$, then

$$\begin{aligned} &U(S_h) = H(O) + \\ &\sum_{s \in D_h} \sum_{l=1}^{|\Omega(O)|} Pr(s) \frac{Pr(o_l)Pr(s|o_l)}{Pr(s)} \log \frac{Pr(o_l)Pr(s|o_l)}{Pr(s)} \\ &= H(O) + \sum_{s \in D_h} \sum_{l=1}^{|\Omega(O)|} Pr(o_l)Pr(s|o_l) \log \frac{Pr(o_l)Pr(s|o_l)}{Pr(s)} \\ &= H(O) + \\ &\sum_{s \in D_h} \sum_{l=1}^{|\Omega(O)|} Pr(o_l)Pr(s|o_l)\{\log(Pr(o_l)Pr(s|o_l)) - \log Pr(s)\} \\ &= H(O) + \sum_{l=1}^{|\Omega(O)|} \sum_{s \in D_h} Pr(o_l)Pr(s|o_l) \log (Pr(o_l)Pr(s|o_l)) \\ &\quad - \sum_{l=1}^{|\Omega(O)|} \sum_{s \in D_h} Pr(o_l)Pr(s|o_l) \log Pr(s) \end{aligned} \tag{4}$$

Note that $Pr(s|o_l)$ represents the probability of s being the correct answers of HITs, given o_l occurs, therefore, $Pr(s|o_l) = 1$ if s and o_l reflect the same facts about the HITs (denoted $o_l \Rightarrow s$), otherwise $Pr(s|o_l) = 0$ (denoted $o_l \Rightarrow s$). Hence, for each l, we have

$$\begin{aligned} U(S_h) &= H(O) + \sum_{l=1}^{|\Omega(O)|} \sum_{s \in D_h \wedge o_l \Rightarrow s} Pr(o_l) \log Pr(o_l) \\ &\quad - \sum_{l=1}^{|\Omega(O)|} \sum_{s \in D_h \wedge o_l \Rightarrow s} Pr(o_l) \log Pr(s) \\ &= H(O) + \sum_{l=1}^{|\Omega(O)|} Pr(o_l) \log Pr(o_l) - \sum_{s \in D_h} Pr(s) \log Pr(s) \\ &= H(O) - H(O) - \sum_{s \in D_h} Pr(s) \log Pr(s) \\ &= -\sum_{s \in D_h} Pr(s) \log Pr(s) = H(h_1, ..., h_k) \end{aligned} \tag{5}$$

□

COROLLARY 3.2 (SUB-MODULARITY). *[1] The utility function derived in Definition 3.2 is a monotonic sub-modular set function of HITs.*

As stated by the above theorems, selecting a k-element subset of HITs is a maximization problem of a sub-modular function.

In general, maximizing sub-modular functions is NP-hard. Concerning the computation of the value of information, [1] shows that, for a general reward function R_j (in our problem, $R_j = U(O)$), 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.

However, maximization of sub-modular functions can be approximated with a performance guarantee of $(1 - 1/e)$, by iteratively selecting the best one HIT, given the ones selected so far [8].

Formally, we have the optimization function at the k^{th} iteration:

$$\begin{aligned} x &:= \arg\max_x U(S_{k-1} \cup h_x) \\ &= \arg\max_x H(S_{k-1}, h_x) \end{aligned} \tag{6}$$

where S_{k-1} is the set of HITs selected from previous iterations. Now we consider S_{k-1} and h_x as two random variables, then by the definition of conditional entropy, we have

$$H(S_{k-1}, h_x) = H(S_{k-1}) + H(h_x|S_{k-1})$$

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

$$x := \arg\max_x H(h_x|S_{k-1}) \tag{7}$$

3.2.2 Selecting The First HIT

We first discuss the selection of the first HIT. This is particularly important, because one may be only interested in finding the best HIT (k=1) for each probabilistic object. In this case, the optimization is to find the HIT h_i that maximize $U(\{h_i\}) = H(h_i)$. Note

$$H(h_i) = Pr(h_i) \log Pr(h_i) + (1 - Pr(h_i)) \log(1 - Pr(h_i))$$

So by taking $\frac{\partial H(h_i)}{\partial Pr(h_i)} = 0$, we can conclude that, $U(\{h_i\})$ is a symmetric function of $Pr(c)$, with symmetry axis $Pr(c) = 0.5$. Besides, $U(\{h_i\})$ achieves maximum $Pr(c) = 0.5$, and is monotonic on $[0, 0.5]$ (increasing) and $[0.5, 1]$ (decreasing). Therefore, $U(\{h_i\})$ is maximized by taking the h_i with $Pr(h_i)$ being closest to 0.5.

In the running example, HIT1 (Pr(HIT1)=0.45) can be the first one to be selected, since no other possible HIT has probability closer to 0.5 than HIT1 does.

Please note that, setting $k = 1$ in MaC framework would lead to high utility-efficiency, but there would be only one HIT for each probabilistic object for the entire process. Hence, the overall time cost for spending the entire budget is significantly reduced.

3.2.3 Table-Partition Algorithm

For a probabilistic object O, we have a table containing n possible outcomes $\Omega(O) = \{o_1, ..., o_n\}$ with probabilities $\{Pr(o_1), ..., Pr(o_n)\}$, and $\sum_{i=1}^{n} Pr(o_i) = 1$. Each outcome is an entry of the table. For a given HIT h_0 asking "$Does\ A = a$", we denote $h_0 \in o_i$ if a is the value of A in o_i; $h_0 \notin o_i$ otherwise. So we have $Pr(h_0) = \sum_{h_0 \in o_i} Pr(o_i)$. In the running example, $Pr(HIT1) = Pr(o1) + Pr(o4) = 0.45$.

One can see that, a naive way of computing the conditional entropy at the each iteration is to compute all the joint probabilities, hence yields to $O(2^k)$ complexity. However, with creating a small $(O(n))$ in-memory index, we can achieve overall linear complexity $(O(kn \sum_{i=1}^{n} |\Omega(A_i)|))$ for finding k HITs, where n is the size of $\Omega(O)$.

We propose an novel algorithm named "Table Partition". The intuition of this algorithm is the fact that, for each HIT h_0, $\Omega(O)$ can be divided in to two parts, with $h_0 = yes$ and $h_0 = no$. Based on this intuition, the essential goal of selecting k HITs is to partition $\Omega(O)$ into at most 2^k parts, and the aggregated probability of each part (i.e. the sum of probabilities of entries within a part) is similar, in order to maximize the overall entropy. Now we illustrate the algorithm in detail as following steps.

Step 1: Find the HIT h_1 with $Pr(h_1)$ closest to 0.5 //Section 3.2.2

Step 2: Partition the table into two parts Ω_0 and Ω_1, where $\forall o \in \Omega_0, h_1 \in o$, and $\forall o \in \Omega_1, h_1 \notin o$.

Step 3: Update $Pr(o_i) = Pr(o_i)/Pr(h_1)$ for $o_i \in \Omega_0$ and $Pr(o_j) = Pr(o_j)/(1 - Pr(h_1))$ for $o_j \in \Omega_0$. Index $Pr(\Omega_0) = Pr(h_1)$ and $Pr(\Omega_1) = 1 - Pr(h_1)$.

Step 4: Find the HIT h_2 that maximize the conditional entropy:

$$\sum_{\Omega_l} Pr(\Omega_l) H_{\Omega_l}(h_2) \tag{8}$$

Step 5: Partition each part Ω_l further into two parts, Ω_{l0} and Ω_{l1}. Update $Pr(o_i) = Pr(o_i)/Pr(h_2)$ and $Pr(o_j) = Pr(o_j)/(1 - Pr(h_2))$ for Ω_{l0} and Ω_{l1}, respectively. Index $Pr(\Omega_{l0}) = Pr(\Omega_l)Pr(h_2)$ and $Pr(\Omega_{l1}) = Pr(\Omega_l)(1 - Pr(h_2))$

Step 6: repeat Step 4 and Step 5 until find k HITs

Step 7: compute the entropy of all the parts

Correctness and Complexity: The correctness of the algorithm is straightforward. After a HIT is selected, we update the probability of each outcome o_i to become the current probability conditioning on the selected HITs. Therefore, each part Ω_l corresponds to a point of the marginal distribution of the selected HITs.

At each iteration, for a given HIT, computing the entropy with in each Ω_l is of complexity $O(\Omega_l)$, hence the computation of Eq 8 is $O(\sum_l |\Omega_l|) = O(n)$. So the maximization of each iteration is of complexity $O(n \sum_{i=1}^{n} |\Omega(A_i)|)$. As a result, the overall complexity becomes $O(kn \sum_{i=1}^{n} |\Omega(A_i)|)$, which is essentially linear of the input.

4. UNCERTAINTY MANAGEMENT

In this section, we discuss the management of the uncertainty of HIT answers from crowdsourcing workers. As illustrated in Section 2, we consider a HIT batch as the basic unit of uncertainty management. In general, we have the following problem definition.

PROBLEM STATEMENT 2 (UNCERTAINTY MANAGEMENT). *A batch of HITs $\{h_1, ..., h_k\}$ is answered by a collection of workers. Each HIT h_i is at least answered by one worker, and each worker answers at least one HIT. Note that the HITs of a batch are independent, and we have a prior probability $Pr(h_i)$ of each HIT (i.e. $Pr(h_i)$ to yes, and $1 - Pr(h_i)$ to be no). We aim to estimate the posterior probability $Pr(h_i|Ans)$, where Ans denotes the crowdsourced answers obtained.*

Explicitly, we discuss the stated problem in three common scenarios of HIT publication: (1) the answer and confidence of each worker is available for the given batch of HITs; (2) the confidence of each worker is available, and we only have a voting score for each HIT (e.g. yes:no = 10:8); (3) the answer of each worker is available for the given set of HITs, but their confidences are unknown. Note that, we assume workers independently answer each HIT in all the above scenarios.

Remark: There are also other possible scenarios, in which the uncertainty can be trivially evaluated. For instance, when only a voting score $(x : y); x \geq y$ is available (no worker confidences), the uncertainty is simply $\frac{x}{x+y}$.

4.1 Confidence-Answer (CA) Scenario

In the rest of this paper, we naturally consider the confidence of a worker i as a random variable following a Bernoulli distribution with probability p_i to answer a HIT correctly. Before detailing any techniques, we first discuss the possible ways to obtain the confidences with individual workers. In general, a confidence can be estimated based on either frequency or inference. A) frequency: when the personal historical records (i.e. the HITs previously answered) are available, the confidence can be estimated by the percentage of correct ones. Typically, one may apply a set of HITs with ground truths (a.k.a. golden standard) for such estimation. B) inference: for a given HIT, the confidences can be inferred from the expertise of workers, according to the documents associated with each individual [9, 19]. Such inference is particularly appropriate for HITs with inexplicit answers, i.e. the ground truth is defined as "the experts' opinion" (e.g. schema matching, truth discovery).

The Confidence-Answer scenario is described as follows. For each HIT h_0 in the HIT batch, a set of answers $a_1, a_2, ..., a_m$ are collected from the crowd, answered by independent workers with confidences $p_1, p_2, ..., p_m$, respectively. We aim to re-estimate the probability of h_0 being yes, based on the given answers. Since there is no correlation between the HITs, we can reduce the problem of solely one HIT. Then the solution can be repetitively applied every HIT in the batch.

PROBLEM STATEMENT 3 (CA UNCERTAINTY MANAGEMENT). *Given a HIT h_0 with $Pr(h_0)$ to be yes; a set of answers of h_0 - $\{a_1, ..., a_m\}$ from independent workers of individual confidences $\{p_1, ..., p_m\}$. We need to compute the posterior probability of h_0,i.e.*

$$Pr(h_0|a_1, a_2, ..., a_m)$$

Computing the uncertainty is straightforward for CA scenario, since all the information of the process of crowdsourcing is obtained. From Bayes' theorem, we have

$$Pr(h_0|a_1, a_2, ..., a_m) = \frac{Pr(h_0)Pr(a_1, a_2, ..., a_m|h_0)}{Pr(a_1, a_2, ..., a_m)} \quad (9)$$

Due to the independence of workers, we have

$$\begin{aligned} Pr(a_1, a_2, ..., a_m|h_0) &= \prod_{a_i=yes} p_i \prod_{a_j=no} (1-p_j) \\ Pr(a_1, a_2, ..., a_m) &= Pr(h_0) \prod_{a_i=yes} p_i \prod_{a_j=no} (1-p_j) \\ &+ (1-Pr(h_0)) \prod_{a_i=yes} (1-p_i) \prod_{a_j=no} p_j \end{aligned} \quad (10)$$

As a result, the posterior probability can be efficiently computed with linear running time.

4.2 Majority Voting (MV) Scenario

Majority voting is one of the most popular paradigms of crowdsourcing, especially on on-line communities (e.g. social networks, forums). However, for some reasons (e.g. privacy), the voting is usually non-tangible, i.e. the individual results of workers is unavailable, and what we have is only the final voting score. In other words, for a yes-no HIT, the voting score T:(m-T) indicates that T out of m workers answered 'yes', others answered 'no'. Similar to the CA scenario, we can consider each HIT in the batch independently. Therefore, we define the problem as follows.

PROBLEM STATEMENT 4 (MV UNCERTAINTY MANAGEMENT). *We are given a HIT h_0 with $Pr(h_0)$ to be yes; a set m workers(voters) of individual confidences $\{p_1, ..., p_m\}$; and we know that T of them answered 'yes', while others answered 'no'. We need to compute the posterior probability of h_0,i.e.*

$$Pr(h_0|e_s : voting\ scores\ T : (m-T))$$

Comparing to the CA scenario, MV scenario preserves the privacy of individual option of workers. This is particularly advantageous when the HIT contains sensitive information.

In the follows, we introduce a Divide-and-Conquer algorithm for the stated problem. Similar to Eq 9, we want to compute

$$Pr(h_0|e_s) = \frac{Pr(h_0)Pr(e_s|h_0)}{Pr(e_s)} = \frac{Pr(h_0)Pr(e_s|h_0)}{Pr(e_s|h_0) + Pr(e_s|\neg h_0)} \quad (11)$$

where e_s is the event of voting scores being $T : (m-T)$. In fact, T can be considered as a discrete random variable following a Poisson Binomial distribution. Therefore, the probability mass function of T, given h_0 is

$$\begin{aligned} Pr(T=t|h_0) &= \sum_{A \in F_t} \prod_{i \in A} p_i \prod_{j \in A^c} (1-p_j) \\ Pr(T=t|\neg h_0) &= \\ &= Pr(T=(m-t)|\ h_0) = \sum_{A \in F_t} \prod_{i \in A} (1-p_i) \prod_{j \in A^c} p_j, \end{aligned} \quad (12)$$

where $F_t = \{A||A| = T; \forall i \in A, answer\ of\ worker\ p_i\ is\ consistent\ with\ h_0\}$. Eq 12 states the closed-form expression of the $Pr(T = t|h_0)$. However, the possible number of A is factorial of T, so traversing the searching space is infeasible in practice unless the number of workers is small.

Based on Formula 5, $Pr(T = t|h_0)$ aims to compute the probability when the $T = t$ under the condition of h_0 given. Because each p_i follows the Bernoulli distribution, T is the sum of p_i, each of which follows the Bernoulli distribution with different probability parameter. Thus, it is the core problem for obtaining the uncertainty to compute the probability mass function of T efficiently. Since the different value of T can be enumerated, the probability distribution of different values of T is represented as a vector. When the set of confidences is given, the probability distribution

Input: A set of answers with individual confidences P_m
Output: A vector of probability distribution of T, V_T
if $m > 1$ **then**
  Split m workers into two sub-groups $P_{\lceil \frac{m}{2} \rceil}$ and $P_{\lfloor \frac{m}{2} \rfloor}$;
  $V_{Upper} \leftarrow DC(P_{\lceil \frac{m}{2} \rceil})$;
  $V_{Lower} \leftarrow DC(P_{\lfloor \frac{m}{2} \rfloor})$;
  $V_T \leftarrow FFT(V_{Upper}, V_{Lower})$;
  return V_T;
end
else
  $V_T[0] \leftarrow 1 - p_1$;
  $V_T[1] \leftarrow p_1$;
  return V_T;
end

Algorithm 1: Divide-and-Conquer-based Algorithm (DC)

Input: A set of answers with individual confidences P_m;
  voting score t:(m-t)
Output: $Pr(h_0|e_0)$
$V_T \leftarrow DC(P_m)$//indexing;
$P_0 \leftarrow V_T[t]$ //Eq 12;
$P_1 \leftarrow V_T[m-t]$//Eq 12;
$Pr(h_0|e_0) \leftarrow \frac{Pr(h_0)P_0}{P_0 * P_1}$//Eq 11;

Algorithm 2: MV Uncertainty Management (MVUM)

of T can be computed by the following divide-and-conquer-based algorithm , as shown in Algorithm 1.

In Algorithm 1, the algorithm firstly divides the set of workers P_m into two groups of the same size as long as P_m includes more than one element. Then, the algorithm recursively solves the confidences of partitioned groups. In particular, we use the fast Fourier Transform (FFT) to speed up the recursive processing. Moreover, the recursive boundary is computed. Therefore, based on Algorithm 1, we obtain the probability distribution of $|C|$ and can compute Acc of the given set of workers P_m easily. Algorithm 2 shows the details of computing Acc.

According to Algorithm 1, the computational complexity is $O(m \log^2 m)$ where m is the size of the set of worker P_m, due to the Fast Fourier Transform acceleration. Based on the Algorithm 2, the total computational complexity is $O(m \log^2 m)$ for indexing the probability distribution plus $O(1)$ to answer a given query of voting scores. Therefore, when multiple HITs are answered by the same group of workers via majority voting, Algorithm 1 only needs to be executed once.

4.3 Answer-Only(AO) Scenario

In the Answer-Only(AO) scenario, we do not assume any prior information concerning the worker confidences. AO scenario is particularly appropriate when neither personal information nor historical records of the workers are available.

PROBLEM STATEMENT 5 (AO UNCERTAINTY MANAGEMENT). *A HIT batch $H = \{h_1, ...h_m\}$ are answered by a set of workers $U = \{u_1, ..., u_m\}$. Each HIT is answered by one or more workers, and each user answers at least one HIT. Given the set of answers $A = \{a_{ij} | a_{ij}$ is the answer for HIT h_i from worker $u_j\}$, our objective is to estimate $Pr(h_i|A)$ without knowing the individual confidences.*

Since the HITs are published in batch, we can estimate the confidences of workers if the h_i are known. On the other hand, we can also estimate $Pr(h_i|A)$ with the given confidences of workers. Based on this intuition, we adopt the Dawid-Skene's approach [3], which utilizes an EM algorithm for the problem presented above. The details of algorithm are illustrated in the appendix.

5. APPLICATIONS AND EVALUATIONS

By adopting MaC framework, we design hybrid machine-crowd systems for three real-world applications: data fusion, information extraction and pattern recognition. The experimental results verified the effectiveness of the proposed methods. We focus on evaluating two issues. First, we examine the effectiveness of MaC framework in reducing the **uncertainty** of probabilistic objects. Second, we verify the correctness of MaC framework, by evaluating the improvement of **overall accuracy**, i.e. the fraction of best outcomes that are the same as the ground truth. In particular, we compare hybrid machine-crowd systems with machine-only systems. The experimental results verified the effectiveness and applicability of our proposal. In this section, we present the details of our experimental studies on data fusion, information extraction and pattern recognition. The experimental study shows that MaC has wide applicability on various applications as well as different types of "crowds".

5.1 MaC-based Data Fusion System

Introduction to Data Fusion: Handling conflicting web data from a series of websites has been one of the most challenging

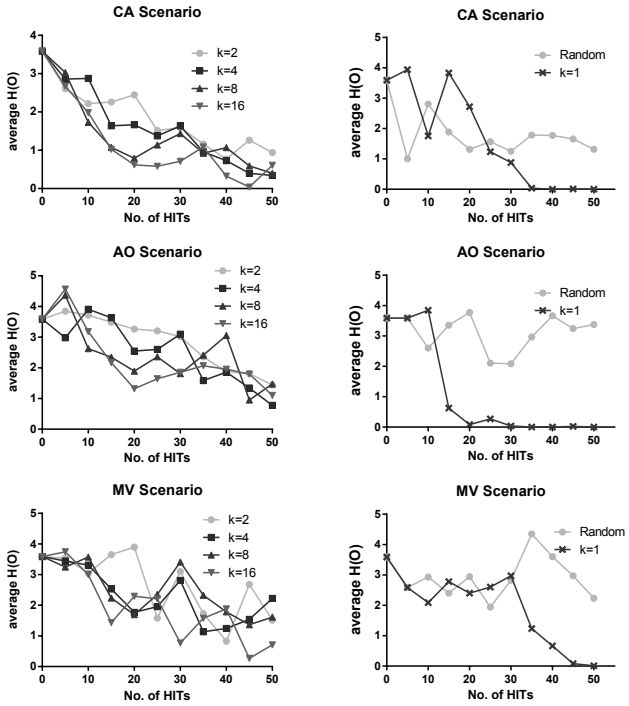

Figure 2: Uncertainty Reduction of Truth Discovery

issues in modern data management systems. In many real applications, the users are also willing to make a contribution to the application. Hence, such users can be considered a crowd, which brings the opportunity to resolve the conflicting information. In this experiment, we adopt MaC framework to build a hybrid machine-crowd fusion system.

Data sources: We identified a number of websites containing general information of computer-science conferences, including wikiCFP (wikicfp.com), confSearch (www.confsearch.org) and some personal web-pages manually maintained by computer scientists. We designed a crawler for each of the websites.

System overview: We implemented a system, namely ConfMGT, to integrate the information of computer-science conferences. This system is designed for researchers to conveniently manage the conferences with their cellphones. In general, a user can search conferences (s)he is interested in, and maintain personal categories of conferences. The whole system follows a classical Client-Server architecture. On the client side, a user may maintain a list of conferences, and synchronize the important dates with his or her calendar. On the server side, the crawlers constantly monitor the updates of the websites. Due to data entry errors or copying the same error web-pages [5], there is usually conflicting information for the same conference. Explicitly, we focused on the truth finding on the following three pieces of information: the conference starting time, place (city), and deadline of submission. We also manually extracted the ground truths from the official websites, which are used for evaluation.

Framework Adoption: According to MaC framework, we naturally consider each conference as a query object (Definition 2.1). We first adopt the fusion techniques proposed in [5], which considers the accuracy of the websites and the copying relationship among the websites in truth finding. These fusion techniques are corresponding to the machine-only system of MaC framework. As a result, we have several possible outcomes for each conference, which is a probabilistic object (Definition 2.2) with three attributes. An example is shown in Table 3.

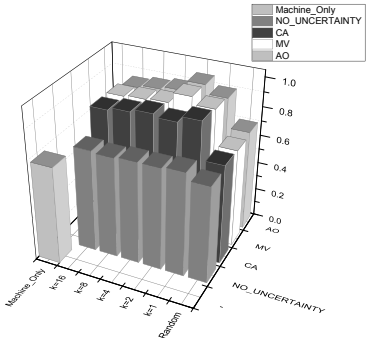

Figure 3: Accuracy Improvement of Data Fusion

Crowd: We leveraged the users of ConfMGT as the crowd to improve the data quality of our system. Users may answer simple YES-No HITs, which are generated by the method introduced in Section 3, and pushed to users. In detail, we have 70 conferences with conflicting information, and set $k = 2, 4, 8, 16$ for HIT selection of each conference. Note that these HITs are batched according to Definition 2.4.

id	Start Date	Deadline	Location	Prob
o1	22 Apr. 2013	Oct. 28, 2012	Wu Han,China	0.13
o2	25 April 2013	Oct. 28, 2012	Wu Han,China	0.02
o3	22 Apr. 2013	Nov. 2, 2012	Wu Han,China	0.43
o4	25 April 2013	Nov. 2, 2012	Shang Hai,China	0.42

Table 3: Probabilistic object in MaC-based Fusion System

Performance: For evaluation purpose, we manage the uncertainty of the crowd under the three scenarios introduced in Section 4. As shown in Figure 2, the average uncertainty of the conferences (with conflicting information) is significantly reduced as HITs are answered, comparing to selecting HITs randomly. On the other hand, the overall accuracy, i.e. the fraction of best outcomes that are the same as the ground truth, outperforms the machine-only system in all the three scenarios, as shown in Figure 3. It worth noticing that the change of uncertainty is not necessarily monotonic. The uncertainty may increase when some surprising answers are received. For instance, a worker with high confidence answered 'yes' for a HIT of prior probability $Pr(h) = 0.1$. Please note that, for each bar chart in Figure 3, the bar on the intersection of 'NO_UNCERTAINTY' and 'Random' refers to the scenario that all the HITs are randomly selected, and we consider the crowdsourced answer to be 100% accurate. When conflicting answers are received, the majority one is accepted.

An important phenomenon is that both the lowest average entropy and the highest accuracy are consistently found when $k = 1$. This is because we always choose the very best HIT at each iteration. When $k > 1$, not every HIT is at the top of the list when they are selected, so the uncertainty is reduced more rapidly with a smaller k.

5.2 MaC-based Information Extraction

Problem Introduction: Information Extraction (IE) is the problem of finding specific information from unstructured data, which is a critical issue for many applications. In general, it is still very difficult to tackle IE completely with an algorithmic approach. Actually, the uncertainty of information extraction is because the existing models (e.g. HMM or CRF) cannot fully capture the semantic of given data representation (e.g. human language). Therefore, we adopt our proposal to built a hybrid machine-crowd system of Information Extraction.

Datasets: In particular, we have a data set including 400 location addresses written in natural English (i.e. string segments). We have

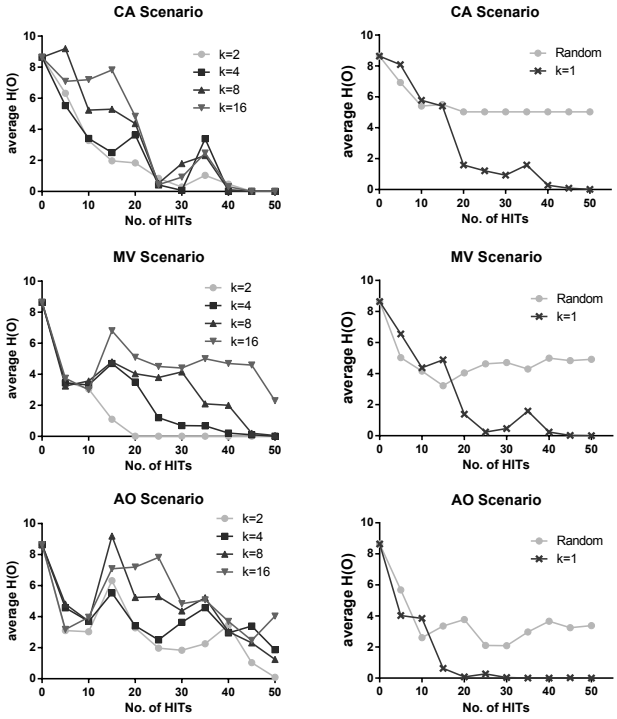

Figure 4: Uncertainty Reduction of Information Extraction

the following attributes of interests: house number, street name, and city name.

Framework Adoption: In information extraction problem, the input of a string segment is considered as a query object. The well-known learning-based tools (HMM or CRF) generate a set of ranked list of extracted data, each of which is associated with a probability. As a result, we consider the a learning-based IE tool as the machine-only system to obtain a set of possible extraction results, which can be seen as a probabilistic object defined in MaC framework. Naturally, each attribute of interests is an attribute of MaC framework, as illustrated in the running example of Section 1.

Crowd: In order to evaluate MaC framework with various crowdsourcing platforms, we adopt the Amazon Mechanical Turk (AMT), which is a widely used crowdsourcing marketplace, as the crowd. Empowered with the Amazon Mechanical Turk SDK for Java, we are able to interactively publish and manage the HITs. In our implementation, each HIT is priced US$0.05. Each HIT is associated with a batch id, and workers are informed that, they are only allowed to accept HITs with the same batch id. Otherwise, their answers would not be accepted (i.e. no payment). In addition, a qualification test is required for each worker. For CA and MV scenarios, the confidence is computed as the fraction of questions answered correctly in the qualification test.

Performance: We demonstrate the change of average uncertainty of the extraction results. Similar to Section 5, we use the crowdsourced answers under the three scenarios. As shown in Figure 4, the average entropy is reduced significantly. In addition, the accuracy is also improved in all the three scenarios, as shown in Figure 5. An important observation is that, in Figure 4, the curves with small k tend to have better performance. This is because that the smaller k leads to more knowledge for selecting each HIT. This phenomenon is intuitive, but not very obvious in the other two experiments. A possible reason could be that, each HIT is only answered by a small number of workers in AMT.

5.3 MaC-based Recognition System

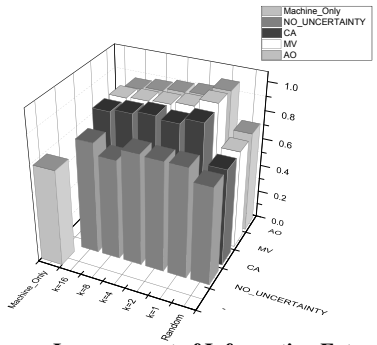

Figure 5: Accuracy Improvement of Information Extraction

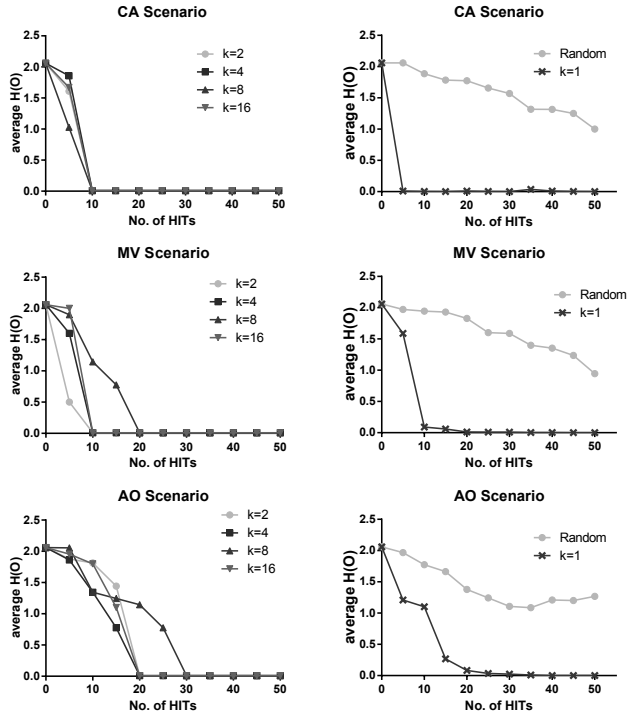

Figure 6: Uncertainty Reduction of Poker Card Recognition

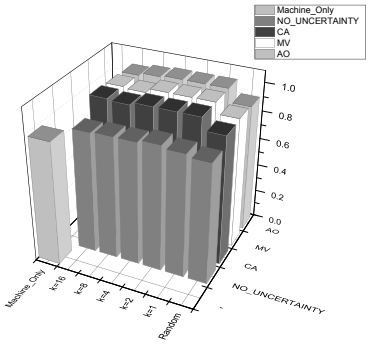

Figure 7: Accuracy Improvement of Recognition

Problem Introduction: In the field of pattern recognition, the study on recognition of poker cards is not only interesting but also practical and valuable, especially in gaming industry. When the environment is clear and stable, the accuracy of identifying ranks and suits of the poker images can be almost 100%. However, there are cases that are very difficult for a machine-only system in the real applications. For instance, the pokers may be overlapped; the light may change over time; and there may be different brands of pokers (i.e. different pattens). In this experiment, we adopt the MaC framework to build a MaC-based system for poker recognition.

Outcome	Suits	Ranks	Probability
o1	Hearts	A	Pr(o1)=0.4
o2	Diamonds	8	Pr(o2)=0.3
o3	Clubs	8	Pr(o3)=0.25
o4	Spades	A	Pr(o4)=0.05

Table 4: Probabilistic objects in MaC-based Recognition System

Framework Adoption: We consider each poker card as a query object, so the possible recognition results are probabilistic objects, as shown in Table 4. For experimental purpose, we develop a simple card recognition system as follows. We set up four cameras (10 million pixels). Each cameras is connected with a different classification program. Explicitly, we apply the following classical methods- Regularized discriminant analysis (RDA), Classification and Regression Trees (CART), Support Vector Machine (SVM), and Artificial Neural Networks (ANN). The cameras are set 1.5 meters above a Baccarat gambling table. For simplicity, we set that each image includes six poker cards.

With a testing set of 500 images, RDA,CART,SVM,ANN have the individual accuracies 0.62,0.63,0.85,0.88 respectively. Note that, the cards in the testing set are randomly placed (may have overlapping), and 40% 'salt and pepper' noise are added. As a result, a probabilistic object can be constructed when the classifiers have conflicting results. We select 50 probabilistic objects, with initial uncertainty (entropy) 2.06 on average.

Crowd Simulation and Performance: In this experiment, we conduct a simulation of the crowd's behavior according to three scenarios described in Section 4. Explicitly, we simulate 100 workers, with confidences following a uniform distribution on $[0.5, 1)$. Figure 6 illustrates the reduction of the average of entropies; and the accuracy is improved up to almost 100%, as shown in Figure 7. A valuable observation is that, the reduction is very rapid comparing to the other two experiments. This phenomenon indicates that, when confidences of workers follow Bernoulli distributions, the entropy would converge to zero reasonably fast. In addition, this experiment uses synthetic crowd responses, and the entropy converges to small values much faster than the previous two experiments. This is because there are other factors in real-world crowds that is not captured by our model.

6. RELATED WORK

In this section, we review the related work in two categories: crowdsourcing-based data management and active learning via crowds.

6.1 Crowdsourcing

The recent booming up of crowdsourcing brings us a new opportunity to engage human intelligence into the process of answering queries (see [4] as a survey). Crowdsourcing provides a new problem-solving paradigm [2, 10], which has been blended into several research communities. In particular, crowdsourcing-based data management techniques have attracted much attention in the database and data mining communities recently. In the practical viewpoint, [6] proposed and developed a query processing system using microtask-based crowdsourcing to answer queries. Moreover, in [14], a declarative query model is proposed to cooperate with standard relational database operators. In addition, in the viewpoint of theoretical study, many fundamental queries have been extensively studied, including filtering [13], max [7], etc. Besides, crowdsourcing-based solutions of many complex algorithms are also developed, such as categorization based on graph search [15], entity resolution [17], etc.

Although the previous studies have already proposed some fundamental data operations based on crowdsourcing, they only focus on solving individual queries or operations. In contrast, our work aims to propose a general framework for a series of crowdsourcing-based data processing problems.

6.2 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 [16]. In particular, [12, 18] proposed active learning methods specially designed for crowdsourced databases. Our work is essentially different from active learning in twofold: (1) the role of workers in active learning is to improve the learning algorithm (e.g. a classifier); in MaC framework, the involvement of workers is to answer queries. (2) The uncertainty of answers in active learning is usually assumed to be given before generating any questions; in MaC framework, the uncertainty of answers has to be estimated after the answers are received, since we cannot anticipate which workers would answer our questions.

7. CONCLUSION

In this paper, we proposed a novel framework, namely MaC, to form a collaboration with the power of machine and the wisdom of crowds. In particular, we tracked two challenging issues involved in the MaC framework - HIT selection and uncertainty management. For HIT selection, we provided an entropy-based model to measure the importance of HITs, and an efficient $(1+\epsilon)$ approximation algorithm is deigned for the selection. For uncertainty management, we discussed the modelling and computation of workers' uncertainty in three common scenarios. By adopting the MaC framework, we conducted extensive experiments on real-world applications. The experimental results demonstrate that the proposed framework is applicable to a wide range of applications.

8. ACKNOWLEDGMENT

This work is supported in part by the Hong Kong RGC Project N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200, National Natural Science Foundation of China (NSFC) Grant No. 61328202, Microsoft Research Asia Gift Grant and Google Faculty Award 2013.

9. REFERENCES

[1] C. G. Andreas Krause. A note on the budgeted maximization of submodular functions. Technical report, School of Computer Science, Carnegie Mellon University, March 2005.

[2] D. C. Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence February 2008 vol. 14 no. 1 75-90*, 2008.

[3] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, 28(1):20–28, 1979.

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

[5] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: The role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[6] A. Feng, M. J. Franklin, D. Kossmann, T. Kraska, S. Madden, S. Ramesh, A. Wang, and R. Xin. Crowddb: Query processing with the vldb crowd. *PVLDB*, 4(12):1387–1390, 2011.

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

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

[9] W. Li, C. Zhang, and S. Hu. G-finder: routing programming questions closer to the experts. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, OOPSLA '10, pages 62–73, New York, NY, USA, 2010. ACM.

[10] T. Malone, R. Laubacher, and C. Dellarocas. Harnessing crowds: Mapping the genome of collective intelligence. Research Paper No. 4732-09, MIT, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, USA, February 2009. Sloan Research Paper No. 4732-09.

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

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

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

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

[15] 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.

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

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

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

[19] Y. Zhou, G. Cong, B. Cui, C. S. Jensen, and J. Yao. Routing questions to the right users in online communities. In *ICDE*, pages 700–711, 2009.

APPENDIX

EM algorithm in Answer-Only(AO) Scenario:
We regard $T = \{p_1, p_2, ..., p_m\}$ as the missing data, which are the corresponding confidences of workers in U; and $\theta = \{Pr(h_1|A), ..., Pr(h_m|A)\}$ as the parameters to be estimated, while the set of answers A are fixed. Since EM algorithms are sensitive of initialization, we apply the prior probability $Pr(h_i)$ as the initialized value of $Pr(h_i|A)$ for the first iteration.

E-step: The E-step computes the expectation of the complete-data log-posterior w.r.t. the missing data T, given the current estimates of the parameters $\hat{\theta}$, the so-called Q-function is computed with the following definition:

$$\begin{aligned} &Q(\theta|\hat{\theta}) = E_T[\log Pr(W, T|\theta)] \\ &= \log\{ \prod_{a_{ij}=yes} (p_j Pr(h_i|A) + (1-p_j)(1-Pr(h_i|A)) \\ &\prod_{a_{ij}=no} (1-p_j)Pr(h_i|A) + p_j(1-Pr(h_i|A))\} \end{aligned} \tag{13}$$

where p_i is the estimation of personal confidence of user u_i at the current iteration. We denote the number of answers from u_i as $W(u_i)$, then we have

$$p_i = \frac{\sum_{a_{ij}=yes} Pr(h_i|A) + \sum_{a_{ij}=no}(1-Pr(h_i|A))}{W(u_i)} \tag{14}$$

By substituting Equation 14 into Equation 13, we have a Q-function that is differentiable for all $Pr(h_j|A)$.

M-step: The M-step then updates the parameter estimates for each h_j by maximizing the Q-function in Equation 13. By setting the partial derivative of the Q function w.r.t. h_j to zero, we get the update value of h_j by solving

$$\frac{\partial Q(\theta|\hat{\theta})}{\partial Pr(h_i|A)} = 0 \tag{15}$$