An Online Cost Sensitive Decision-Making Method in Crowdsourcing Systems

Jinyang Gao†, Xuan Liu†, Beng Chin Ooi†, Haixun Wang§, Gang Chen‡
†School of Computing, National University of Singapore, Singapore
§Microsoft Research Asia, Beijing, P.R. China
‡College of Computer Science, Zhejiang University, Hangzhou, P.R. China
†{jinyang.gao, liuxuan, ooibc}@comp.nus.edu.sg, §haixunw@microsoft.com, ‡cg@zju.edu.cn

ABSTRACT

Crowdsourcing has created a variety of opportunities for many challenging problems by leveraging human intelligence. For example, applications such as image tagging, natural language processing, and semantic-based information retrieval can exploit crowd-based human computation to supplement existing computational algorithms. Naturally, human workers in crowdsourcing solve problems based on their knowledge, experience, and perception. It is therefore not clear which problems can be better solved by crowdsourcing than solving solely using traditional machine-based methods. Therefore, a cost sensitive quantitative analysis method is needed.

In this paper, we design and implement a cost sensitive method for crowdsourcing. We online estimate the profit of the crowdsourcing job so that those questions with no future profit from crowdsourcing can be terminated. Two models are proposed to estimate the profit of crowdsourcing job, namely the linear value model and the generalized non-linear model. Using these models, the expected profit of obtaining new answers for a specific question is computed based on the answers already received. A question is terminated in real time if the marginal expected profit of obtaining more answers is not positive. We extends the method to publish a batch of questions in a HIT. We evaluate the effectiveness of our proposed method using two real world jobs on AMT. The experimental results show that our proposed method outperforms all the state-of-art methods.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous

Keywords

Crowdsourcing, Decision-making

1. INTRODUCTION

Crowdsourcing has attracted a great deal of interest as a platform for leveraging crowd-based human computation and intelligence. It is to some extent inspired by the vast amount of collaboration in Web 2.0 communities, where users share not only information, but also their knowledge and intelligence. Platforms such as, for example, the Amazon Mechanical Turk (AMT) [1], facilitate the refinement of data and reduction of noise by passing complex jobs to human workers. These complex jobs include image tagging, semantic-based information retrieval, and natural language processing, which are hard for computers, but relatively easy for human workers. Instead of designing sophisticated algorithms or spending a lot of money to consult experts, many of these jobs can be solved by human workers on a crowdsourcing platform at a much lower cost. Some successful crowdsourcing applications that appear recently include CrowdDB [4], CrowdSearch [19], and HumanGS [14].

Despite the success of crowdsourcing systems, employing crowdsourcing effectively remains challenging for three reasons. First, for most crowdsourcing jobs, we need to obtain multiple answers to guarantee their quality. Thus, we have to decide when to stop obtaining new results provided by human workers. Most existing work uses the accuracy or the cost as the optimization objective. These turn out to be too rigid in practice. However, the tradeoff is not trivial. Typically crowdsourcing jobs may have different level of difficulty (e.g. homework of kids vs. research problems), risks (e.g. Flickr image tagging vs. cancer diagnosing), and profits (e.g. survey for personal interest vs. design for investment model). Therefore, it is important to have an online economic model that considers all these factors of crowdsourcing jobs.

Second, none of the current research focuses on whether a problem is suitable for crowdsourcing or not. Intuitively, crowdsourcing based techniques are more fitting for problems that require semantic processing. For example, sentiment analysis, image tagging, and information retrieval are good problems for crowdsourcing while large scale numerical analytics are better handled by machines.

Finally, the estimation of crowdsourcing quality is still primitive. Low quality answers may sharply reduce the quality of crowdsourcing, and introduce noises. To resolve the quality issue, several methods have been proposed, such as Crowdscreen [12] and CDAS [9]. Both proposals only consider the accuracy of the answers provided by the workers, without taking into account the difficulty of the tasks. However, it is very challenging to predict the difficulty of the tasks, and a robust algorithm that can handle problems of varied levels of difficulty and answers of varied levels of quality is needed.

In this paper, we propose a novel online cost sensitive decision-making model to address the above three challenges. Our contributions include:

SIGMOD'13, June 22–27, 2013, New York, New York, USA.

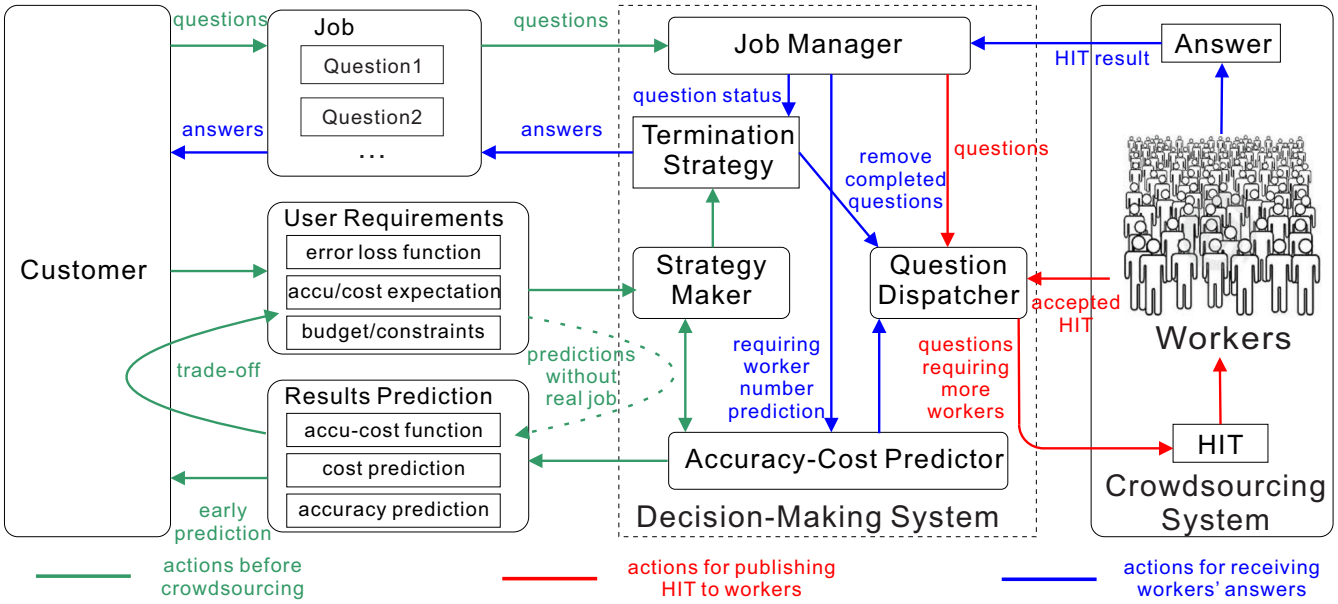

Figure 1: The Architecture of Cost Sensitive Decision Making System

- We propose an online, cost sensitive decision-making model to analyze and decide whether to stop the question in its current status, given the value of the question, the risk of getting incorrect answers, and the cost of workers in the crowdsourcing system. To the best of our knowledge, our model is the first that provides an online quantitative profit analysis of crowdsourcing jobs. We further extend our algorithm to support online cost sensitive decision-making with constraints, such as limited budgets etc.
- An application or task may contain questions of diverse difficulty levels. We propose a model for measuring the difficulty of a question and We design and implement a robust algorithm that handles such questions.
- We propose a novel algorithm called Accuracy-Cost to perform the marginal analysis of the accuracy and the cost in crowdsourcing. The algorithm calculates the incremental profit when the number of workers is increased.
- We conduct extensive experimental studies on two real datasets obtained from the answers of workers in AMT to evaluate the effectiveness of our proposed method. The results show that our method obtains precise results while keeping the cost low. We also develop an automatic question dispatching method that assigns multiple questions in a HIT while each question can be terminated at any time.

The rest of this paper is organized as follows. Section 2 gives an overview of our method. Section 3 explains the preliminaries. Section 4 presents our proposed linear model to get real time decisions and analyse the profit. Section 5 extends our linear model to the non-linear model, which demonstrates the relationship between cost and accuracy and gives our model widespread applicability. Section 6 discusses the experimental studies. Section 7 reviews the related works. Finally, we conclude in Section 8.

2. OVERVIEW

In this section, we give an overview of our cost sensitive decision-making method for crowdsourcing.

Figure 1 shows the architecture of our proposed method. The core part is the decision-making system (the dashed box), which consists of five components, namely the job manager, the question dispatcher, the accuracy-cost predictor, the strategy maker, and the termination strategy manager. The job manager passes the questions from the crowdsourcing customer to the question dispatcher. It keeps collecting the answers of the workers and reporting the updated question status to the termination strategy manager. After receiving the questions, the question dispatcher allocates these questions to several workers using a question dispatching algorithm (will be discussed in Subsection 5.2). The termination strategy manager determines whether we should stop getting more answers for a question and return the results to the customer based on the status of the question. These termination strategies are generated by the strategy maker according to user's requirements, namely the error loss function, the accuracy/cost expectation, the budget, or other constraints. The strategy maker employs a linear model we propose in Section 4. Note that the generated strategies are stored in the termination strategy manger so that our decision-making system is able to decide if we should terminate a question in real-time. The accuracy-cost predictor provides two major functionalities. First, it predicts the results as well as the accuracy/cost ratio. Second, it generates the termination strategies using a generalized non-linear model together with the strategy maker. The details of the non-linear model is discussed in Section 5.

In summary, our proposed decision-making system automatically dispatches questions to the workers. It terminates the questions and returns the answers in real-time according to the generated termination strategies. This decision-making system also provides an early prediction of the results before running a real crowdsourcing job in order to help the customer set proper user requirements. In the following sections, we present the idea and the method of implementing this decision making system.

3. PRELIMINARIES

Before describing our method in Section 4 and 5, we discuss the required prior knowledge to understand the problem.

For the sake of brevity, in our model we assume the questions are two-choice problems. We can think of the two choices as binary values 0 and 1. However, we can easily extend our method to problems that have more than two choices.

Question status: We model the status of a question by a pair (m, l) that represents the numbers of the two different answers received from the workers. Since the values 0 and 1 are just two symbols to represent both choices, the 0s and 1s in the answers can be exchanged. As a result, the following two cases: (1) m 0s and l 1s; (2) m 1s and l 0s can be viewed as identical to represent the agreement of the workers on the two choices. Without loss of generality, in the remainder of this paper, we use (m, l) $(m \geqslant l)$ to represent the above two cases.

Intuitively, the question status indicates the difficulty of the question. When m is far larger than l, most of the workers agree on one choice. It could be that this question is easy. On the other hand, when m is close to l, it is likely that the question is too difficult so that the workers are just making guesses.

Question run: A question run r is a sequence of question statuses as we get answers from workers for this question, i.e.,

$$r = \{(m_0, l_0), (m_1, l_1), \cdots, (m_n, l_n)\}$$

where $(m_0, l_0) = (0, 0)$ is the initial status. Every non-initial status (m_i, l_i) has exactly one more answer than its previous status (m_{i-1}, l_{i-1}), i.e., $(m_i, l_i) = (m_{i-1}, l_{i-1}+1)$ or $(m_{i-1}+1, l_{i-1})$. In the above question run r, the question is terminated at question status (m_n, l_n), after which it will not accept more answers. For example, $\{(0, 0), (1, 0), (2, 0), (2, 1)\}$ is a valid question run that stops after getting three answers but $\{(0, 0), (1, 0), (2, 1), (2, 0)\}$ is not a valid question run.

Accuracy of question answer: We use A_Q to denote the accuracy of an answer to question Q, i.e. the probability that a worker provides the correct answer. Most of existing works considering the accuracy of answers assume that the accuracy is a fixed value that can be computed from sampled answers, e.g. [9][12]. The major drawback of these models is that they do not take the difficulty of questions into consideration. Using these models, for a single worker, his answers to different questions would have the same quality. However, this observation contradicts the intuition that the answers of a hard question might be poor.

In our paper, instead of modelling the accuracy as a single fixed value and employing sampling based methods to estimate this value, we represent the accuracy as a probability distribution. The probability distribution provides the ability of modelling answer quality based on the observed question status. We model the accuracy A_Q as a random variable and we estimate the value of A_Q by the observation of the question status (m, l). Specifically, we assume that A_Q obeys the Beta distribution:

ASSUMPTION 1. *Given a question Q, the probability density function of A_Q is:*

$$f(A_Q = \mu) = \mu^{a-1}(1-\mu)^{b-1}/Beta(a, b)$$

We thus have $E[A_Q] = a/(a+b)$. a and b are parameters of the Beta distribution, representing the prior prediction of the random variable. The Beta distribution is actually a two-dimensional Dirichlet distribution [1]. We can replace the Beta distribution by the more general Dirichlet distribution for multiple choice questions.

If the prior distribution is $\mathrm{B}(\mu|1, 1)$, i.e., the uniform distribution, then the estimation of A_Q is only determined by the observation of the question status. In other words, our method is a generalized form of both the traditional fixed accuracy model and the uniform distribution. It takes both the prior knowledge and the question status observation into consideration to adapt the estimated distribution of the difficulty of problems. Ideally the estimated distribution performs best when it is the same as the empirical distribution of problem difficulty. However, the empirical distribution is always difficult to obtain. Moreover, it is not feasible to compute Bayesian inference based on the empirical distribution. The Beta distribution is used to simplify the computation, as it is the conjugate prior distribution of Binomial and Bernoulli distribution. While on the other hand, Beta distribution with proper parameters fits the accuracy distribution well.

We have observed that our model is robust since using different prior prediction parameters a, b almost does not change the results. This is because our model is based on a probability distribution and it can be adaptively adjusted to be applied on questions with diversified difficulties. Moreover, we observed that the empirical distribution of A_Q in several real question sets, including the tweet sentiment analysis questions and common sense questions used in our experiments, is similar to the $\mathrm{B}(\mu|6, 2)$ distribution. On the other hand, our experiments also show that when distribution changes, this estimation still works well. In Section 4, we explain the reason that the probability distribution based accuracy model outperforms the fixed value based accuracy models. In the experiment section, we use empirical data to support the above two observations.

Accuracy of question result: We use A_R to represent the accuracy of the result based on a majority voting on the current question status (m, l). We always choose the answer represented by the majority m. As a result, A_R is the probability that the majorities m choose the correct answer. We define $\langle x, y\rangle$ as the status having x correct answers and y incorrect answers. As the status (m, l) represents either m correct answers or m incorrect answers, $(m, l) = \langle m, l\rangle \cup \langle l, m\rangle$. By Bayesian analysis, the conditional probability, A_R given A_Q and observation (m, l) is:

$$A_R = \frac{Pr(\langle m, l\rangle)}{Pr(\langle m, l\rangle) + Pr(\langle l, m\rangle)} = \frac{{A_Q}^{m-l}}{{A_Q}^{m-l} + (1-A_Q)^{m-l}}$$

Note that A_R is also a random variable as A_Q is a random variable.

The notations used in the following sections of this paper are listed in Table 1.

4. LINEAR DECISION-MAKING

In this section, we introduce a linear model for online decision making. We first estimate the accuracy of the answers according to the question status and prior distribution. Based on the estimation of the answer accuracy, we obtain the marginal income and the profit of each status. This enables us to make decisions at each status according to the economic profit. In this paper, we employ a dynamic programming algorithm to calculate the profit and generate the strategy for each question status. We prove that the time complexity of our strategy generating algorithm is $O(n^2)$ (n is the maximum possible number of answers for a single question, i.e. the search space), which is a significant improvement compared with existing methods such as CrowdScreen's $O(n^4)$ linear programming. In this section, we also discuss the techniques of extending our method to solve a more complex problem, namely linear model with constraints of accuracy and cost.

4.1 Linear model

For a question Q, the linear model has three variables: the question value V_Q, the error loss L_Q, and the question cost per worker

[1] http://en.wikipedia.org/wiki/Dirichlet_distribution

Table 1: Notations

Q	a question in a crowdsourcing job
V_Q	the value of a crowdsourcing question Q
L_Q	the loss of getting a wrong answer for Q
C_Q	the cost of assigning a question to a worker
(m, l)	the status of the crowdsourcing question Q
A_R	the probability that the voting result of Q is correct
A_Q	the probability that a worker provides the correct answer to the question Q
$MI(m, l)$	the marginal income of the question accuracy A_R in status (m, l)
$P(m, l)$	the expected economic profit of the question in status (m, l)
$P_S(m, l)$	the expected economic profit of stopping the question in status (m, l)
$P_C(m, l)$	the expected economic profit of continuing the question in status (m, l)
$\mathrm{B}(\mu\|a, b)$	PDF of Beta distribution
$\Gamma(n)$	Gamma function
$Beta(a, b)$	Beta function

C_Q. As discussed in Section 2, these values are preset by the crowdsourcing customer. V_Q is the value of this question given an answer (not necessarily a correct one). The error loss L_Q is the penalty of obtaining a wrong answer for Q. C_Q represents the cost of hiring a worker for Q. According to the definition of V_Q, L_Q and C_Q, we propose the following linear model:

DEFINITION 4.1 (PROFIT OF A QUESTION). *Suppose a question Q ends after it receives k answers, and the result is ans. The profit P of Q is:*

(1) ans is correct: $P = V_Q - kC_Q$.

(2) ans is incorrect: $P = V_Q - L_Q - kC_Q$.

However, in practice we do not know whether the answer ans is correct or not. Thus, we compute the expected value $E[P]$ of the profit to estimate the average profit of question Q. The expected profit is computed by:

$$\begin{aligned} E[P] &= (V_Q - kC_Q)E[A_R] + (V_Q - L_Q - kC_Q)(1 - E[A_R]) \\ &= L_Q E[A_R] + V_Q - L_Q - kC_Q \end{aligned}$$

where A_R is the probability that ans is correct. Note that A_R is a random variable, we thus use the expected $E[P]$ with respect to all possible worlds of A_R.

We use two functions, namely the value function $f^V(E[A_R])$ and the cost function $f^C(E[A_R])$, to describe the expected profit function $E[P](E[A_R])$. $f^V(E[A_R])$ is the expected gain from the job, while $f^C(E[A_R])$ is the cost of crowdsourcing. In this model, obviously $f^V(E[A_R])$ (i.e. $L_Q E[A_R] + V_Q - L_Q$) is a linear function of accuracy $E[A_R]$. It is remarkable that the expected profit function is not a linear function for $E[A_R]$, since larger $E[A_R]$ requires more answers (larger number of k). We will discuss the non-linear $f^V(E[A_R])$ in next section for some special needs in multi-question situation.

In Subsection 4.2, we describe the method to estimate $E[A_R]$ based on the prior distribution $\mathrm{B}(\mu|a, b)$ of the difficulty A_Q and the observations on the final status (m, l) of Q.

We formally define the decision-making problem based on the linear model.

PROBLEM DEFINITION 1. *Given V_Q, L_Q and C_Q of a question Q, find the decision-making algorithm to maximize $E[P]$ among all possible runs of Q on any answer sequence provided by the workers.*

We next show the method of deriving $E[A_Q]$, $E[A_R]$ etc. in Subsection 4.2 and we discuss the algorithm of finding the question run that maximizes $E[P]$ in Subsection 4.3.

4.2 Accuracy Estimation

Suppose the observation of the final status is (m, l), we can derive the $E[A_Q]$ based on the prior distribution $\mathrm{B}(\mu|a, b)$ using Bayesian Analysis. The posterior result is our estimation of $E[A_Q]$.

THEOREM 4.1.

$$E[A_Q] = \frac{\Gamma(a+m+1)\Gamma(b+l) + \Gamma(a+l+1)\Gamma(b+m)}{(a+b+m+l)(\Gamma(a+m)\Gamma(b+l) + \Gamma(a+l)\Gamma(b+m))}$$

where $\Gamma(n)$ is the Gamma function such that $\Gamma(n) = (n-1)!$ for any positive integer n.

PROOF. We prove the theorem using Bayesian Theorem. The conditional probability density function of the observation $O = (m, l)$ given the condition $A_Q = x$ ($x \sim \mathrm{B}(\mu|a, b)$) is

$$f_O(m, l|A_Q = x) = \binom{m+l}{m}(x^m(1-x)^l + x^l(1-x)^m)$$

Note that the probability density function contains two cases: (1) m correct answers, l incorrect answers and (2) m incorrect answers, l correct answers. Using Bayesian Theorem, we have:

$$\begin{aligned} f_{A_Q}(A_Q = x|m, l) &= \frac{f_O(m, l|A_Q = x) f_{A_Q}(A_Q = x)}{f_O(m, l)} \\ &= \frac{(x^m(1-x)^l + x^l(1-x)^m)x^{a-1}(1-x)^{b-1}}{\int_0^1 (x^m(1-x)^l + x^l(1-x)^m)x^{a-1}(1-x)^{b-1}dx} \\ &= \frac{x^{a+m-1}(1-x)^{b+l-1} + x^{a+l-1}(1-x)^{b+m-1}}{\int_0^1 x^{a+m-1}(1-x)^{b+l-1} + x^{a+l-1}(1-x)^{b+m-1}dx} \end{aligned}$$

Therefore, the expected value of A_Q is:

$$\begin{aligned} E[A_Q] &= \int_0^1 x f_{A_Q}(A_Q = x|m, l)dx \\ &= \int_0^1 \frac{x^{a+m}(1-x)^{b+l-1} + x^{a+l}(1-x)^{b+m-1}}{\int_0^1 (x^{a+m-1}(1-x)^{b+l-1} + x^{a+l-1}(1-x)^{b+m-1})dx}dx \\ &= \frac{\int_0^1 x^{a+m}(1-x)^{b+l-1}dx + \int_0^1 x^{a+l}(1-x)^{b+m-1}dx}{\int_0^1 (x^{a+m-1}(1-x)^{b+l-1}dx + \int_0^1 x^{a+l-1}(1-x)^{b+m-1})dx} \\ &= \frac{Beta(a+m+1, b+l) + Beta(a+l+1, b+m)}{Beta(a+m, b+l) + Beta(a+l, b+m)} \\ &= \frac{\Gamma(a+m+1)\Gamma(b+l) + \Gamma(a+l+1)\Gamma(b+m)}{(a+b+m+l)(\Gamma(a+m)\Gamma(b+l) + \Gamma(a+l)\Gamma(b+m))} \end{aligned}$$

□

Theorem 4.1 gives the average accuracy of the answers. Based on this theorem, we can further derive the expectation of the accuracy of the results A_R using Bayesian Analysis. We have the following theorem:

THEOREM 4.2. *Given observations on the final status (m, l) of a question, the expected accuracy of the results $E[A_R]$ is:*

$$E[A_R] = \frac{\Gamma(a+m)\Gamma(b+l)}{\Gamma(a+m)\Gamma(b+l) + \Gamma(a+l)\Gamma(b+m)}$$

Table 2: Trends of A_Q and A_R

	Question Status / Prior Distribution	(1,0)	(3,3)	(4,0)	(8,2)	(100,100)	(101,100)	(110, 100)
A_Q	Fixed accuracy 0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
	B(μ|6, 2)	0.75	0.643	0.821	0.762	0.51	0.51	0.513
	Fixed accuracy 0.80	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	B(μ|8, 2)	0.8	0.687	0.853	0.79	0.514	0.514	0.52
A_R	Fixed accuracy 0.75	0.75	0.5	0.988	0.999	0.5	0.75	1
	B(μ|6, 2)	0.75	0.5	0.962	0.953	0.5	0.51	0.591
	Fixed accuracy 0.80	0.8	0.5	0.996	1	0.5	0.8	1
	B(μ|8, 2)	0.8	0.5	0.985	0.983	0.5	0.514	0.634

PROOF. Note that A_R is a function of A_Q. Thus, the expected value of A_R is:

$$\begin{aligned} & E[A_R] = \int_0^1 A_R f_{A_Q}(A_Q = x|m,l)dx \\ = & \int_0^1 \frac{x^m(1-x)^l}{x^m(1-x)^l + x^l(1-x)^m} \\ & \times \frac{x^{a+m-1}(1-x)^{b+l-1} + x^{a+l-1}(1-x)^{b+m-1}}{\int_0^1 x^{a+m-1}(1-x)^{b+l-1} + x^{a+l-1}(1-x)^{b+m-1}dx}dx \\ = & \frac{\int_0^1 x^{a+m-1}(1-x)^{b+l-1}dx}{\int_0^1 x^{a+m-1}(1-x)^{b+l-1} + x^{a+l-1}(1-x)^{b+m-1}dx} \\ = & \frac{Beta(a+m, b+l)}{Beta(a+m, b+l) + Beta(a+l, b+m)} \\ = & \frac{\Gamma(a+m)\Gamma(b+l)}{\Gamma(a+m)\Gamma(b+l) + \Gamma(a+l)\Gamma(b+m)} \end{aligned}$$

□

Theorem 4.1 and 4.2 show that the expected values of A_Q and A_R only depend on the status (m, l) and the prior parameters a, b.

To illustrate Theorem 4.1 and 4.2, we show several examples of trends of A_Q and A_R on different question status given specified prior distribution of difficulty of questions in Table 2. We compare the two Beta distributions B(μ|6, 2) and B(μ|8, 2) with two fixed value based models with accuracy 0.75 and 0.8, which are the expected values of the two Beta distributions respectively. We obtain the following facts from Table 2: in status $(3, 3)$, the predicted value of A_Q of the B(μ|8, 2) distribution model (0.687) is smaller than that of fixed value model (0.75); in status $(4, 0)$, the predicted value of A_Q of the B(μ|6, 2) distribution model (0.821) is larger than that of fixed value model (0.8). These two facts show that the Beta distribution model provides a better prediction than the fixed value based models, since the predicted values of A_Q according to the fixed value based models are always the same no matter what the current question status is.

Moreover, the predicted values of different distribution models converge while the number of answers goes up. Therefore, the choice of prior parameters in the distribution model is not important, because the distribution model can adaptively adjust the predicted values according to the observation of question status Consider the status $(8, 2)$, the two fixed value models (0.75 and 0.8) provide very aggressive predictions on A_R, i.e. 0.999 and 1 respectively. However, in practice, 8 of 10 workers agree may not always guarantee the correctness of the results.

When the status is (100, 100), we intuitively believe A_Q is likely to be close to 0.5 rather than 0.75 or 0.8. Based on the status $(100, 100)$, we consider the case of accepting one more answer, i.e. question status $(101, 100)$. Obviously A_R should still be close to 0.5 as the votes of both choices are very close. However, the two fixed value based models provide the prediction of A_R as 0.75 and 0.8 respectively, which are the same as the predicted values in question status $(1, 0)$. This example shows that the Beta distribution model outperforms the fixed value based models by considering the question status.

According to Theorem 4.1 and 4.2, we can predict the possible future answers based on $E[A_Q]$, $E[A_R]$ and the current status (m, l). We define two transitive probabilities:

(1) $Pr(+1|m, l)$: probability from status (m, l) to $(m + 1, l)$.

(2) $Pr(-1|m, l)$: probability from status (m, l) to $(m, l + 1)$.

Intuitively $Pr(+1|m, l)$ and $Pr(-1|m, l)$ represent the probability that the next answer is the same value as the value voted by m and l answers respectively. Obviously $Pr(+1|m, l) + Pr(-1|m, l) = 1$. We have the following theorem:

THEOREM 4.3.

$$\begin{aligned} & E[Pr(+1|m,l)] \\ = & \frac{\Gamma(a+m+1)\Gamma(b+l) + \Gamma(b+m+1)\Gamma(a+l)}{(a+b+m+l)(\Gamma(a+m)\Gamma(b+l) + \Gamma(a+l)\Gamma(b+m))} \end{aligned}$$

The proof is analogous to Theorem 4.1. Based on these above theorems, we can quantitatively derive the decision-making strategies.

4.3 Decision Making

We consider the marginal income of accuracy at each status (m, l). Then we derive the profit according to the marginal income of accuracy. We define the marginal income of accuracy at status (m, l) as the expected increase of the $E[A_R]$ after obtaining a new answer. We denote the marginal income of accuracy at status (m, l) as $MI(m, l)$.

DEFINITION 4.2.

$$MI(m,l) = E'[A_R|m,l] - E[A_R|m,l]$$

where $E'[A_R|m, l]$ is the expected accuracy of the results after getting one more answer, i.e. $E'[A_R|m, l] =$

if m>l: $E[E[A_R|m+1, l]Pr(+1|m, l)+E[A_R|m, l+1]Pr(-1|m, l)]$, i.e. $E[A_R|m+1, l]E[Pr(+1|m, l)]+E[A_R|m, l+1]E[Pr(-1|m, l)]$

if m=l: $E[E[A_R|m+1, l]Pr(+1|m, l)+E[A_R|l+1, m]Pr(-1|m, l)]$, i.e. $E[A_R|m+1, l]E[Pr(+1|m, l)]+E[A_R|l+1, m]E[Pr(-1|m, l)]$

By simplifying the equations in Definition 4.2, we have the following theorem:

THEOREM 4.4. *The marginal income of accuracy $MI(m, l)$ satisfies:*

(1) $MI(m, l) = 0$ when $m > l$.

(2) $MI(m, m) = \frac{a-b}{2(a+b+2m)}$.

Algorithm 1: Generate Linear Strategy Algorithm

Input: Parameter a, b of prior distribution, Error Loss L_Q, question cost C_Q
Output: Strategy $\mathcal{S}$ for each status (m, l)

$M \leftarrow \lceil \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b)) \rceil$;
$\mathcal{S}.M \leftarrow M$;
for *i from M to 0* **do**
  $\mathcal{S}.(M,i) \leftarrow stopping$;
  $\mathcal{S}.P_S(M,i) \leftarrow V_Q - (1 - E[A_R|M,i])L_Q - (M+i)C_Q$;
  $\mathcal{S}.P(M,i) \leftarrow \mathcal{S}.P_S(M,i)$;
for *i from M − 1 to 0* **do**
  for *j from i to 0* **do**
    $\mathcal{S}.P_S(i,j) \leftarrow V_Q - (1 - E[A_R|i,j])L_Q - (i+j)C_Q$;
    $\mathcal{S}.P_C(i,j) \leftarrow E[Pr(+1|i,j)]\mathcal{S}.P(i+1,j) + E[Pr(-1|i,j)]\mathcal{S}.P(i,j+1) - C_Q$;
    $\mathcal{S}.P(i,j) \leftarrow \max\{\mathcal{S}.P_S(i,j), \mathcal{S}.P_C(i,j)\}$;
    if $\mathcal{S}.P_S(i,j) >= \mathcal{S}.P_C(i,j)$ **then**
      $\mathcal{S}.(i,j) \leftarrow stopping$;
    else
      $\mathcal{S}.(i,j) \leftarrow continuing$;
return $\mathcal{S}$

PROOF. We prove the theorem by calculating $MI(m,l)$. When $m > l$, getting an extra answer cannot change the result of the majority voting. Therefore, the marginal income of accuracy is 0. As a result, $MI(m,l) = 0$ when $m > l$.

When $m = l$,

$$\begin{aligned} & MI(m,m) = E[A_R|(m+1,m)] - E[A_R|(m,m)] \\ = & \frac{\Gamma(a+m+1)\Gamma(b+m)}{\Gamma(a+m+1)\Gamma(b+m) + \Gamma(a+m)\Gamma(b+m+1)} \\ & - \frac{\Gamma(a+m)\Gamma(b+m)}{\Gamma(a+m)\Gamma(b+m) + \Gamma(a+m)\Gamma(b+m)} \\ = & \frac{(a+m)\Gamma(a+m)\Gamma(b+m)}{((a+m)+(b+m))\Gamma(a+m)\Gamma(b+m)} - \frac{1}{2} \\ = & \frac{a+m}{a+b+2m} - \frac{1}{2} \\ = & \frac{a-b}{2(a+b+2m)} \end{aligned}$$

□

Note that $a > b$ is guaranteed by the prior distribution. Therefore, $MI(m,m)$ decreases while m increases.

Based on the marginal income of the accuracy, we can further derive the profit $P(m,l)$ of a question in status (m,l). The profit is defined as the maximum economic profit at status (m,l) given the linear model in Subsection 4.1. We introduce two more profit functions before formally defining $P(m,l)$, namely the profit $P_S(m,l)$ of stopping the question at status (m,l) and the profit $P_C(m,l)$ of continuing the question at status (m,l). Since a question has only two choices (stopping and continuing) at any status, we therefore formally define the profit at status (m,l) as:

DEFINITION 4.3.

$$P(m,l) = \max\{P_S(m,l), P_C(m,l)\}$$

Intuitively, we should stop the question when $P_S(m,l) >= P_C(m,l)$ because we cannot benefit from this question any more by waiting for more answers.

Now we consider the two profit functions $P_S(m,l)$ and $P_C(m,l)$. Given the accuracy $E[A_R]$, we know that the result is incorrect with probability $1 - E[A_R]$. Therefore, according to the linear model, the profit $P_S(m,l)$ satisfies:

$$P_S(m,l) = V_Q - (1 - E[A_R])L_Q - (m+l)C_Q$$

When we continue a question, we will pay one more C_Q and get a new answer from a new worker. Meanwhile, the status is also changed to a new status. It is easy to find that the next status is $(m+1,l)$ with probability $Pr(+1|m,l)$ and $(m,l+1)$ with probability $Pr(-1|m,l)$. Thus, the profit $P_C(m,l)$ is recursively defined as following:

$$\begin{aligned} P_C(m,l) &= E[Pr(+1|m,l)]P(m+1,l) \\ &\quad + E[Pr(-1|m,l)]P(m,l+1) - C_Q \end{aligned}$$

Given the recursive definition, it is difficult to calculate them directly. Now we discuss the condition to guarantee the strategy of one status (m,l) stopping, i.e.

$$P_S(m,l) \geqslant P_C(m,l)$$

m that satisfies $MI(m,m)L_Q < C_Q$ is obvious a lower bound, as we will not get a positive profit in each future step. Moreover, we have found a looser lower bound of m. We have the following theorem:

THEOREM 4.5 (TERMINATION THEOREM). *A sufficient condition of $P_S(m,l) \geqslant P_C(m,l)$ is*

$$m \geqslant \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b))$$

and $l < m$.

PROOF. We define $P'(m,l) = P(m,l) - P_S(m,l)$. Based on this definition, we can rewrite the Definition 4.3 by

$$\begin{aligned} P'(m,l) &= \max\{0, MI(m,l)L_Q + E[Pr(+1|m,l)]P'(m+1,l) \\ &+ E[Pr(-1|m,l)]P'(m,l+1) - C_Q\} \end{aligned}$$

We have the observations $P'(m,l) \geqslant P'(m,l-1)$ and $P'(m,l) \geqslant P'(m+1,l)$ for all $m > l$. For a large enough number m, we have $MI(m,m) < C_Q/L_Q$ since $\lim_{m\to+\infty} MI(m,m) = 0$. Based on $MI(m,m)L_Q < C_Q$, there exist some m such that $P'(m+1,m) = 0$. We can prove the following Lemma:

LEMMA 4.1. *$P'(m,m-1) = 0$ if $P'(m+1,m) = 0$ and $m \geqslant \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b))$*

Based on Lemma 4.1, we therefore state that $P'(m,m) = 0$ when $m \geqslant \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b))$. □

PROOF OF LEMMA 4.1. *We prove the lemma by contradiction. If $P'(m,m-1) > 0$, we have the following inequality: $MI(m,m-1)L_Q + E[P(+1|m,m-1)]P'(m+1,m-1) + E[P(-1|m,m-1)]P'(m,m) - C_Q > 0$ where $MI(m,m-1) = 0$. Meanwhile, we have*

$$P'(m+1,m-1) <= P'(m+1,m)$$

As a result, $P'(m+1,m-1) = 0$. Obviously, $E[P(-1|m,m-1)] < 1/2$. Therefore, we have $P'(m,m) > 2C_Q$.

Note that

$$P'(m,m) = MI(m,m)L_Q + P'(m+1,m) - C_Q$$

According to the fact that $P'(m+1,m) = 0$, we have

$$P'(m,m) = MI(m,m)L_Q - C_Q$$

Therefore,

$$MI(m,m)L_Q - C_Q > 2C_Q$$

Algorithm 2: Generate Linear Strategy Algorithm with Constraints

Input: Parameter a, b of prior distribution, Error Loss L_Q, question cost C_Q, Budget constraint θ
Output: Strategy $\mathcal{S}$ for each status (m, l)

1 $M \leftarrow \lceil \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b)) \rceil$;
2 $\mathcal{S}.M \leftarrow M$;
3 **for** *i from M to 0* **do**
4 $\mathcal{S}.(M, i) \leftarrow stopping$;
5 $\mathcal{S}.P_S(M, i) \leftarrow V_Q - (1 - E[A_R|M, i])L_Q - (M+i)C_Q$;
6 $\mathcal{S}.P(M, i) \leftarrow \mathcal{S}.P_S(M, i)$;
7 **for** *i from* $M-1$ *to 0* **do**
8 **for** *j from i to 0* **do**
9 **if** $(i+j+1)C_Q > \theta$ **then**
10 $\mathcal{S}.P_S(i, j) \leftarrow V_Q - (1 - E[A_R|i, j])L_Q - (i+j)C_Q$;
11 $\mathcal{S}.P(i, j) \leftarrow \mathcal{S}.P_S(i, j)$;
12 $\mathcal{S}.(i, j) \leftarrow stopping$;
13 **else**
14 $\mathcal{S}.P_S(i, j) \leftarrow V_Q - (1 - E[A_R|i, j])L_Q - (i+j)C_Q$;
15 $\mathcal{S}.P_C(i, j) \leftarrow E[Pr(+1|i, j)]\mathcal{S}.P(i+1, j) + E[Pr(-1|i, j)]\mathcal{S}.P(i, j+1) - C_Q$;
16 $\mathcal{S}.P(i, j) \leftarrow \max\{\mathcal{S}.P_S(i, j), \mathcal{S}.P_C(i, j)\}$;
17 **if** $\mathcal{S}.P_S(i, j) >= \mathcal{S}.P_C(i, j)$ **then**
18 $\mathcal{S}.(i, j) \leftarrow stopping$;
19 **else**
20 $\mathcal{S}.(i, j) \leftarrow continuing$;
21 **return** $\mathcal{S}$

Moreover, $MI(m, m) = (a-b)/(2(a+b+2m))$. *As a result, we have* $m < \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b))$, *which contradicts with* $m \geqslant \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b))$. □

We have designed the linear model based algorithm (Algorithm 1) to generate the strategy deciding whether the question should stop at each status (m, l). The search space bound M is the upper-bound of m in all possible *continuing* status (m, l) in Theorem 4.5:

$$M = \lceil \frac{1}{2}(\frac{L_Q(a-b)}{6C_Q} - (a+b)) \rceil \quad (1)$$

We apply dynamic programming to iteratively compute the decision of the generated strategy $\mathcal{S}$ from the upper-bound back to 0. For each status (m, l), we compute $P_S(m, l)$ and $P_C(m, l)$ based on $P(m+1, l)$ and $P(m, l+1)$. In strategy $\mathcal{S}$, the decision is made for each status (m, l) by comparing $P_S(m, l)$ and $P_C(m, l)$. As a result, the question run that maximizes $P(m, l)$ is found by stopping the question at the first status (m^*, l^*) such that $P_S(m^*, l^*) \geqslant P_C(m^*, l^*)$. Algorithm 1 can be applied to find the question run that maximize $P(m, l)$ for all input 0-1 sequences. Obviously, the time complexity of Algorithm 1 is $O(M^2)$ where M is computed in Equation 1. Note that Algorithm pre-computes all possible decisions offline and it only takes $O(M)$ time to make decisions online by querying $S.(m, l)$ in $O(1)$ time for each status (m, l).

4.4 Model with Constraints

In this subsection, we discuss the decision-making problem with constraints on the accuracy and cost of the result of the question. Suppose the constraints are represented by $cost \leqslant Budget$. We solve this problem by simply adapting Algorithm 1. We mark the status (m, l) as stopping when $(m+l+1)C_Q > Budget$. This adapted algorithm is outlined in Algorithm 2. We can extend algorithm 2 to support other constraints such as accuracy etc.

Algorithm 3: Accuracy-Cost Algorithm

Input: Parameter a, b of prior distribution, question cost C_Q
Output: List of tuples of accuracy, cost and error loss $\{\langle accu, cost, L_Q \rangle\}$

1 result $\leftarrow \emptyset$;
2 $L_Q \leftarrow 1000C_Q$;
3 **while** $L_Q > 0$ **do**
4 $\mathcal{S} \leftarrow$ GenerateLinearStrategy(a, b, L_Q, C_Q);
5 accu $\leftarrow$ ComputeAccuracy($\mathcal{S}$);
6 cost $\leftarrow$ ComputeCost($\mathcal{S}$);
7 result $\leftarrow$ result $\cup \langle$accu[0][0], cost[0][0], $L_Q \rangle$;
8 $max \leftarrow 0$;
9 **foreach** *Non-stop status* (m, l) *in* $\mathcal{S}.status$ **do**
10 $c \leftarrow$ cost$[m][l]$;
11 $l_Q \leftarrow cL_Q/(P_C(m, l) - P_S(m, l) + c)$;
12 **if** $l_Q > max$ **then** $max \leftarrow l_Q$;
13 $L_Q \leftarrow max$;
14 **return** result;

In Algorithm 2, the upper bound of search space M is computed in line 1. Line 3-6 pre-computes the strategy for status (M, i). All statuses (M, i) are set to be *stopping*. Line 7-20 the strategy of each status (i, j) is computed. When status (i, j) does not satisfy the constraints, it is set to be *stopping* in line 10-12. Otherwise, $P_S(i, j)$ and $P_C(i, j)$ are computed respectively in line 14-15. The decision of status (i, j) is decided by comparing $P_S(i, j)$ with $P_C(i, j)$ in line 17-20. The computed strategy $\mathcal{S}$ is returned as the result of Algorithm 2 (line 21).

5. NON-LINEAR DECISION-MAKING

In this section, we discuss a non-linear model based approach to predict the relationship between the accuracy and the cost of the question without prior knowing L_Q. We have presented the decision making algorithm in Section 4 for a single question, where the job value is a linear function of the accuracy. However, considering the case of making decisions on a batch of questions, the value function might be more complicated than a linear function. For example, we list three non-linear functions in Figure 2, representing the cases that we have constraints on the quality of the data and we use the information entropy to measure the informativeness of the data. Therefore, the value functions are not linear. Moreover, the non-linear model based method is also driven by situations where the customers are not able to estimate the error loss function L_Q of some problems. The method we have discussed in Section 4 cannot be simply applied to make decisions for the question in the above situations. Since the quality of data can be well estimated by the accuracy when the number of questions is large enough, we propose Algorithm 3 to find the relationship between accuracy and cost (shown in the bottom of Figure 2) in Subsection 5.1. Algorithm 3 calculates the difference between the value function and the cost function as the profit function (as illustrated on the right hand side of Figure 2). Thus, the maximum point of the profit function (star point in Figure 2) is the trade-off point to maximize the profit. We extend our method to solve the problem of decision making for multiple questions in Subsection 5.2.

5.1 Accuracy-Cost Relationship

To obtain the cost function of accuracy, we propose the non-linear model based algorithm (Algorithm 3) by iteratively applying Algorithm 1. According to the Algorithm 1, we know that given fixed $a, b, L_Q/C_Q$, we have a determined strategy to find the question run that maximizes the profit.

The basic idea of this non-linear model based algorithm is to:

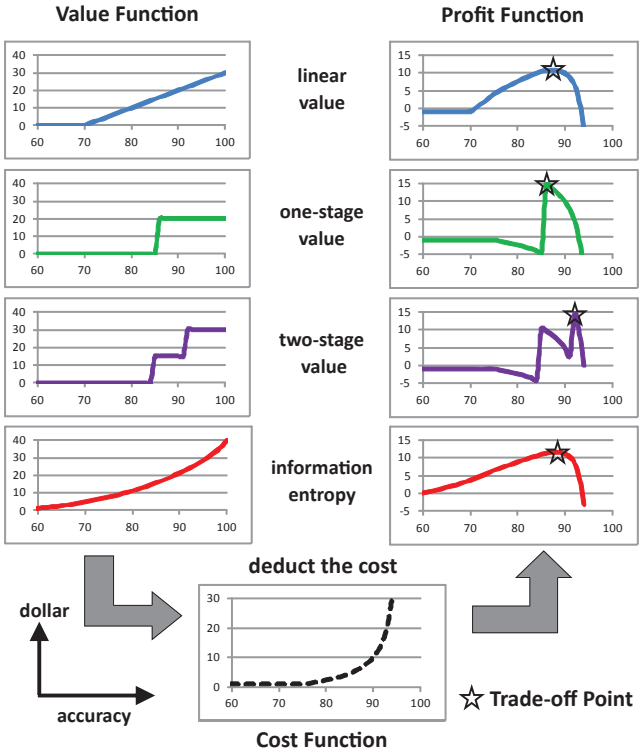

Figure 2: Examples of non-linear decision making

Step 1. initialize L_Q to be a sufficiently large value (i.e. $1000C_Q$) and compute the accuracy and cost based on L_Q.

Step 2. iteratively adjust the value of L_Q by reducing L_Q such that the decision of only one status (m, l) changes from *continuing* to *stopping*.

Step 3. compute accuracy and cost and go back to Step 2.

This method is to enumerate the accuracy and cost pairs by gradually reducing the value of L_Q. We show the detailed method in Algorithm 3. In Algorithm 3, we record the expected accuracy and cost for each status (m, l) using the strategy generated from L_Q. Algorithm 3 iteratively reduces L_Q by selecting the maximum l_Q that can change exactly one *continuing* status to *stopping*. l_Q is computed in line 11. The correctness is guaranteed by the following lemma:

LEMMA 5.1. *Given a continuing status (m, l), we reduce L_Q to be $cost[m][l]L_Q/(P_C(m, l) - P_S(m, l) + cost[m][l])$. If the stopping/continuing status of all status (m', l') other than (m, l) are not changing, we have $P_C(m, l) = P_S(m, l)$.*

PROOF OF LEMMA 5.1. *By the definition of the two profits, i.e. $P_S(m, l)$ and $P_C(m, l)$:*

$$\begin{aligned} P_S(m,l) &= V_Q - L_Q + E[A_R|m,l]L_Q - (m+l)C_Q \\ P_C(m,l) &= V_Q - L_Q + accu[m][l]L_Q - (m+l)C_Q \\ &\quad -cost[m][l] \end{aligned}$$

We therefore have

$$P_S(m,l) - P_C(m,l) = L_Q \Delta accu - cost[m][l]$$

where $\Delta accu = accu[m][l] - E[A_R|m, l]$. We assume that $l_Q \Delta accu - cost[m][l] = 0$. Thus,

$$l_Q = \frac{cost[m][l]}{\Delta accu} = \frac{cost[m][l]L_Q}{P_C(m,l) - P_S(m,l) + cost[m][l]} \quad \square$$

Algorithm 4: Compute Accuracy

Input: Strategy $\mathcal{S}$
Output: Expected accuracy accu of the problem in every status (m, l) using Strategy $\mathcal{S}$

```
1  M ← S.M;
2  for i from M to 0 do
3      for j from i to 0 do
4          if (i, j) is stopped then
5              accu[i][j] ← E[A_R|i, j];
6          else
7              accu[i][j] ←
               Pr(+1|i, j)accu[i+1][j]+Pr(−1|i, j)accu[i][j−1];
8  return accu
```

Algorithm 5: Compute Cost

Input: Strategy $\mathcal{S}$
Output: Expected accuracy cost of the problem in every status (m, l) using Strategy $\mathcal{S}$

```
1  M ← S.M;
2  for i from M to 0 do
3      for j from i to 0 do
4          if (i, j) is stopped then
5              cost[i][j] ← (i + j)C_Q;
6          else
7              cost[i][j] ← Pr(+1|i, j)cost[i + 1][j] +
               Pr(−1|i, j)cost[i][j − 1] + C_Q;
8  return cost
```

The strategy is generated according to the newly updated L_Q. We employ dynamic programming method to compute the accuracy and cost in Algorithm 4 and 5. These two algorithms compute the accuracy and cost respectively for each status (m, l) from (M, M) (*stopping*) back to $(0, 0)$.

We formally define the non-linear value function problem as:

PROBLEM DEFINITION 2. *Given the functions of the value of question quality $f^V(accu)$ and cost $f^C(accu)$ with respect to the accuracy of the result, find the strategy to maximize $f^P(accu) = f^V(accu) - f^C(accu)$ on each possible question run.*

Algorithm 3 provides the accuracy-cost relationship $f^C(accu)$, which makes it possible to find a trade-off point when user requirements $f^V(accu)$ is clear. Moreover, this can be used to give customer the accuracy and cost predictions early and help the user to choose suitable requirements like $f^V(accu)$ or special point of $\langle accu, cost, L_Q \rangle$.

The solution of those non-linear $f^V(accu)$ requirements can be done as follows:

1. Compute $result = \{\langle accu, cost, L_Q \rangle\}$ by calling Algorithm 3.
2. Find $\max_{accu \in result}\{f^V(accu) - f^C(accu)\}$ by computing the difference between the non-linear value and cost for each $accu$ in the $result$. Generate the strategy using the corresponding L_Q of $accu$ as parameter of Algorithm 2.

Notice that in general case the $f^P(accu)$ is not convex (e.g. the profit function for stage value). As a result, we have to calculate every point for computing $f^P(accu)$.

This method can also be applied to solve the problem with constraints on the expectation of accuracy or cost. We only need to modify the non-linear functions $f^V(accu)$ to present the constraints (e.g.stage value function gives a strict constraint of accuracy).

Algorithm 6: Question Dispatching Algorithm

Dispatcher:
Initialize min-heap $heap$ to be empty;
foreach *non-stop question* Q_i **do**
  $C_i \leftarrow cost[0][0]$;
  $P_i \leftarrow 0$;
  $E_i \leftarrow C_i - P_i$;
  if $heap.size()$<k **then**
    $heap.push(Q_i)$;
while *not all questions are stopped and a new worker comes* **do**
  foreach *question* Q_i *in* $heap$ **do**
    Assign Q_i to the HIT; $P_i \leftarrow P_i + 1$;
    $E_i \leftarrow C_i - P_i$;
  foreach *non-stop question* Q_i *in* $heap$ **do**
    Maintain $heap$ using E_i;

Updater:
while *any question* Q_i *gets into a stopping status* $(Q_i.m, Q_i.l)$ **do**
  Mark Q_i as a stopped question.
while *any question* Q_i *gets a new answer to the status* $(Q_i.m, Q_i.l)$ **do**
  $P_i \leftarrow P_i - 1$;
  $C_i \leftarrow cost[Q_i.m][Q_i.l]$;
  $E_i \leftarrow C_i - P_i$;
  Maintain $heap$ using E_i;

5.2 Decision-Making for Multiple Questions

We discuss the question dispatching algorithm (Algorithm 6) in this subsection. This algorithm aims to build a question dispatcher that assigns HITs containing a batch of questions to the workers. Batching questions in a HIT is an effective approach to reduce the average cost of each question. However, our proposed algorithm only generates strategies for a single question. Questions may be included multiple times in different HITs and their decisions are made in real-time. The numbers of required workers are different among the questions. Therefore, we need to design an algorithm to dynamicly batch non-stopped questions in a HIT. The question dispatching algorithm is designed in order to reduce the rate of already stopped questions in the HITs.

The key idea of our question dispatching algorithm is to manage all the questions to be finished at almost the same time. We store all the unfinished questions in a question pool and maintain the number of questions included in the HITs. Attributed to the fact that the randomness of receiving order of the answers, it is difficult to design a deterministic method to find the best assignment of questions. Instead, we maintain the expected number of asking questions from current status such that these numbers are synchronously decreased. As a result, we put questions with the largest expected number of asking questions into a HIT.

We outline the function of dispatcher and updater in Algorithm 6. For the question, the number of expected asking questions C_i at status $(Q_i.m, Q_i.l)$ is retrieved from $cost[Q_i.m][Q_i.l]$ computed by Algorithm 5. Meanwhile, there are some HITs on the worker's hand. We record the number of HITs that are posted but have not yet received answers of each question Q_i as P_i. We use $E_i = C_i - P_i$ as the estimation of expected asking questions. When a HIT is posted or its answer is received, the expectations of effected questions are updated and the questions with the largest expected numbers are maintained with a min heap.

6. EXPERIMENTAL STUDIES

This section will discuss the experiment results of our methods. To evaluate the performance of our proposed methods, we have conducted extensive experiments on two real-world datasets on the AMT. Besides, various robustness tests and theoretical results are studied using synthetic datasets. We show that our proposed method: (1) works better than any other existing method in terms of both the accuracy and the cost; (2) is robust to handle questions with diversified difficulty distribution and unexpected data quality; (3) is scalable when the maximum number of workers is increased; (4) works well on various kinds of crowdsourcing questions.

6.1 Experiment Setup

We use the following two datasets for our performance studies, namely tweet sentiment analysis (*TSA*) dataset and common sense question (*CSQ*) dataset. Humans are good at comprehension and perform well on problems requiring background knowledge. These two datasets focus on the two main advantages of crowdsourcing respectively.

TSA dataset: A real-world tweets dataset containing 400 comments of 20 movies is crawled from Twitter. We generate a sentiment analysis question (positive, negative) for each of the comments as a candidate crowdsourcing task. We assign each of the 400 questions to up to 50 workers using the question dispatching algorithm (Algorithm 6). We repeat the question dispatching algorithm 10 times to get 10 different question runs for each question. In total, we get 200,000 answers from the workers as the *TSA* dataset.

CSQ dataset: We crawl 400 common sense problems from the Internet as the candidate crowdsourcing tasks. We also assign each of the 400 questions to up to 50 workers by repeatedly using Algorithm 6 10 times. In total, we also get 200,000 answers from the workers as the *CSQ* dataset.

Figure 3 compares the distribution of the difficulty of the questions with $\mathrm{B}(\mu|6,2)$ distribution. The average accuracy of results A_R is tested on more than 200 answers. This figure indicates that $\mathrm{B}(\mu|6,2)$ distribution can be well used to model the distribution of the difficulty of the Tweet sentiment analysis questions. Moreover, the results also show that about 6.8% of questions have an average A_R smaller than 0.5, which fits the prediction well. The predicted value of the proportion is 6.25% based on $\mathrm{B}(\mu|6,2)$ distribution. The accuracy of these questions become even worse when more workers answer them. As a result, the overall accuracy of results on all questions can only achieve 93%-94% rather than very close to 100%. We also observe that the questions in both datasets are a bit more difficult than the expectation based on the prior $\mathrm{B}(\mu|6,2)$ distribution.

To compare the performance of proposed algorithm with other existing algorithms, we implement four algorithms, i.e. Accuracy-Cost (Algorithm 3), Crowdscreen [12], Majority Voting and Naive Majority Voting Algorithm. In the majority voting algorithm, when the number of providers of a value is more than a half of the maximum number of workers, the online majority voting algorithm outputs this value and stops, whereas in the offline naive majority voting algorithm, all the values from all workers are collected and output the majority results.

6.2 Problem-Crowdsourcing Fitness

In this subsection we give the analysis on whether a job is appropriate for crowdsourcing or not. We use C_Q as the unit. The results in Figure 4 show that the lower bound of V_Q to get benefit from crowdsourcing job in various L_Q. The loss and cost $V_Q - P(0,0)$ means the total cost of solving a question by crowdsourcing (notice that $P(0,0)$ contains V_Q in it, this measure only contains the error loss expectation and questions cost and is irrelevant to V_Q).

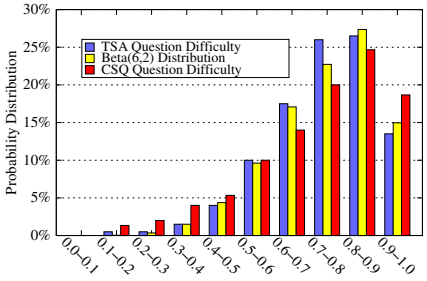

Figure 3: **Distribution of the difficulty of problems vs. B($\mu|6,2$) distribution**

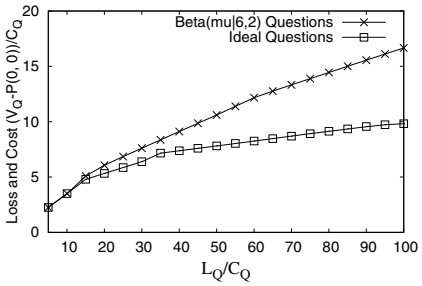

Figure 4: **Lower bound of V_Q**

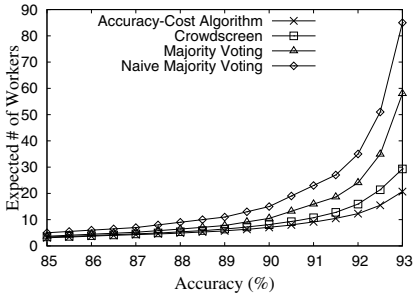

Figure 5: **Expected number of workers given required accuracy**

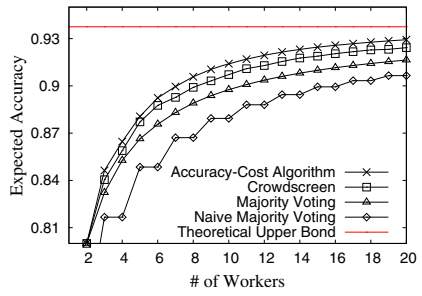

Figure 6: **Expected accuracy given the number of workers**

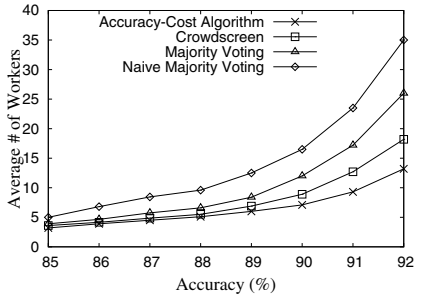

Figure 7: **Empirical number of workers given required accuracy on TSA dataset**

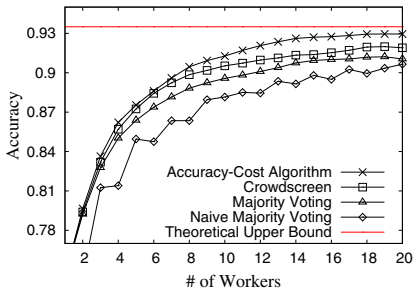

Figure 8: **Empirical accuracy given the number of workers on TSA dataset**

For those jobs where L_Q is too high or V_Q is not enough, crowdsourcing is not a cost-effective way. We compare the loss and cost on two set of questions, namely the questions obeying B($\mu|6,2$) distribution and the questions having the ideal same $A_Q = 0.75$.

6.3 Performance of Accuracy-Cost Algorithm

We test the performance of our Accuracy-Cost Algorithm by investigating the accuracy-cost relationship on both datasets *TSA* and *CSQ*. The results of the number of workers given required accuracy and the accuracy given the number of workers are showed here. We compare the four algorithms, namely Accuracy-Cost, CrowdScreen, Majority Voting and Naive Majority Voting. All four algorithms make the trade-off according to their strategy.

Figure 5 and Figure 6 present the theoretical predictions of the number of workers given a required accuracy and the accuracy given the number of workers respectively based on the B($\mu|6,2$) distribution model.

Experimental results on the *TSA* dataset: Figure 7 shows the empirical number of workers hired in the crowdsourcing system. In Figure 7, the results show that our Accuracy-Cost Algorithm needs the smallest number of workers and Naive Majority Voting needs the largest number. Meanwhile, comparing Figure 7 with Figure 5, the empirical results validate the theoretical predictions. Figure 8 shows the empirical accuracy of the crowdsourcing tasks on the *TSA* dataset. The results in Figure 8 show that our Accuracy-Cost Algorithm has the highest accuracy while Naive Majority Voting has the lowest. The empirical results also fit the theoretical predictions well by comparing Figure 8 with Figure 6. Our Accuracy-Cost Algorithm outperforms other algorithms because our Accuracy-Cost Algorithm computes the maximum accuracy for each possible cost and the smallest cost for each possible accuracy (in Algorithm 3).

Experimental results on the *CSQ* dataset: Figure 9 shows the empirical number of workers hired in the crowdsourcing system. We observe similar trends in Figure 9, i.e. the number of workers of Accuracy-Cost Algorithm is the smallest. The empirical results in Figure 9 also fit the theoretical prediction of the results in Figure 5 well. Figure 10 shows the empirical accuracy of the crowdsourcing tasks on the *CSQ* dataset. The results in Figure 10 are similar to the results in Figure 6.

The experimental results on these two datasets show that our method can be applied on various crowdsourcing questions and yield good accuracy while requiring the least number of workers.

6.4 Robustness

We study the robustness of our algorithm by varying the distribution of question difficulty. We vary variance and expectation of the difficulty distribution of questions such that the distribution is different from our prior B($\mu|6,2$) distribution. The robustness of obtaining satisfied results on unexpected hard questions (or low quality users) is another key requirement, since low data quality without expectation is usually unacceptable.

Figure 11 demonstrates the empirical accuracy of our Accuracy-Cost Algorithm, CrowdScreen and Majority Voting working on unexpected low quality answers with an average 65% accuracy. Note that these four algorithms expect the difficulty of questions to be 75% based on the prior B($\mu|6,2$) distribution. Figure 11 shows that our algorithm still produces results with accuracy very close to the accuracy required by the customer while the other two algorithms fail to obtain results with high accuracy. This phenomenon is due to the fact that our algorithm models the probability of a worker providing an answer as a random variable. This property provides the ability to detect the decrease of accuracy and guarantee high quality results by asking more questions automatically. We can see that

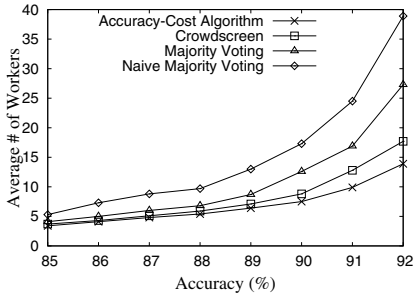

Figure 9: **Empirical number of workers given required accuracy on CSQ dataset**

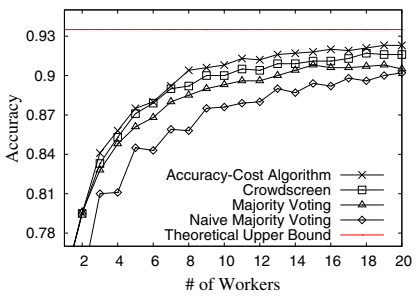

Figure 10: **Empirical accuracy given the number of workers on CSQ dataset**

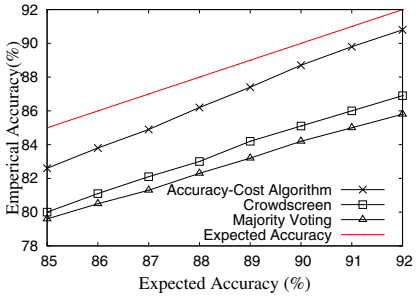

Figure 11: **Accuracy of the algorithms on hard questions**

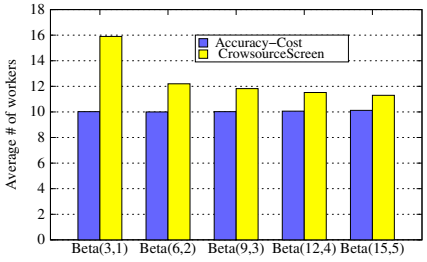

Figure 12: **Vary the variance of difficulty of questions**

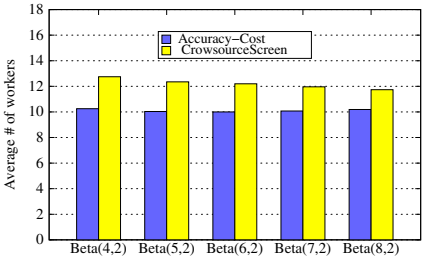

Figure 13: **Vary the expectation of difficulty of questions**

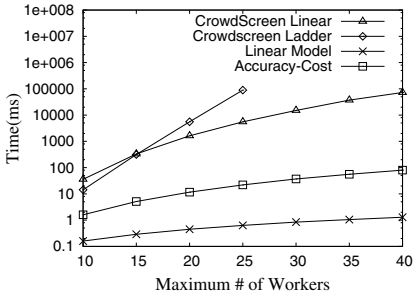

Figure 14: **Strategy Generating Time**

with the growth of the number of workers(i.e. the increase of accuracy requirement), our method obtains more answers and produces more precise estimation, which results in stronger resistance.

Figure 12 and Figure 13 show the performance when we vary the difficulty distribution of questions. In both experiments algorithm with $\mathrm{B}(\mu|6,2)$ distribution and CrowdScreen algorithm are compared to the algorithm using the exact prior distribution (our algorithm outputs the best strategy when data exactly match the prior distribution). In our experiments, we use the accuracy that the exact prior distribution algorithm using 10 workers can achieve as the accuracy requirements. We report the number of needed workers to achieve the accuracy requirements for both algorithms. Figure 12 reports the results of varying the variance of the distribution and Figure 13 reports the results of varying the expectation of the distribution. The results in these two figures indicate that our Accuracy-Cost algorithm needs almost the same number (less than 2% increment) of workers as the exact prior distribution algorithm when we vary the difficulty distribution of the questions, while Crowdscreen algorithm needs more workers when the variance of the distribution is increased.

Instead of using a fixed value, this random variable based probability model is more robust.This implies that our algorithm can by applied to solve unexpected hard questions or working on data sources with unpredictable quality in real crowdsourcing applications.

6.5 Scalability and Question Dispatching

In this subsection, we discuss the scalability of our algorithm when the number of workers and the number of questions increases.

We first report the results on the running time of our algorithms with respect to the number of workers in Figure 14. We compare both our linear and non-linear algorithms with the CrowdScreen's linear and ladder algorithms. Note that the time reported is the offline strategy generating time. The results show that our algorithms are scalable when the number of workers is increased. The linear strategy generating algorithm takes 1.29 milliseconds and nonlinear algorithm takes 80.06 milliseconds when there are 40 workers. The results also show that the two CrowdScreen algorithms need to take a lot more time to generate the strategy.

We report the performance of the question dispatching algorithm in Table 3. In this experiment, the maximum number of questions in a HIT is varied from 400 to 10. The measurements contain the average percentage of valid questions in each HIT, average finishing time, average cost per question and average cost per effective question. The valid question refers to the not yet stopped questions in the HIT. We compare the average percentage of valid questions of our question dispatching algorithm to that of randomly assigning questions. The results show that our question dispatching algorithm assigns up to 10.9% and on average 4.53% more valid questions in each HIT than randomly assigning algorithm.

7. RELATED WORK

Crowdsourcing has been widely used to solve challenging problems by human intelligence in comprehensive areas. In crowdsourcing systems, complex and difficult problems are partitioned to simple tasks. These tasks are assigned to several workers. The crowdsourcing system collects and integrates the answers from the workers as the results of the crowdsourcing jobs. Kitter et al. [8] studied the user behaviour in micro-task markets to show that user performs different behaviours.

Recently, crowdsourcing has been adopted and applied in several research areas such as database researches, machine learning and information retrieval. CrowdDB [3, 4], Qurk [10, 11] and TurkDB [13] designed three databases that are incorporated with crowdsourcing systems. These three databases allow queries to be partially answered through the AMT system. Selke et al. [16] expanded database schemas with additional attributes through querying the crowdsourcing systems. CrowdER [17] applied crowd-

Table 3: Performance of Question dispatching algorithm (2 USD per hour per worker on average)

# of questions in a HIT	400	200	100	50	20	10
Effective questions rate using question dispatcher	48.20%	88.40%	94.60%	98.40%	99.50%	99.80%
Effective questions rate by randomly assigning questions	48.20%	77.50%	87.30%	93.60%	96.70%	98.40%
Average HIT finish time (s)	2850	1472	765	393	189	113
Average cost per question (0.01 USD)	0.396	0.409	0.425	0.437	0.526	0.627
Average cost per effective question (0.01 USD)	0.822	0.463	**0.449**	**0.444**	0.529	0.628

sourcing to find the matching entities. In [9], we proposed a quality sensitive answering model for our system CDAS to manage the crowdsourcing tasks. In [15], Raykar et al. discussed the method of applying crowdsourcing in supervised learning without absolute golden standard. In [5], Guo et al. proposed a method to find the maximum element in a crowdsourcing database. Alonso et al. [2] developed a crowdsourcing based relevance evaluation method for information retrieval while Kazai et al. [7] proposed a crowdsourcing based book search evaluation method. Ipeirotis et al. [6] designed an approach to rank the workers by quality. Welinder et al. [18] proposed a crowdsourcing based online algorithm to find the ground truth. Crowdsourcing techniques has also been applied on other database based applications, such as graph search [14].

CrowdScreen [12] was designed to improve the accuracy and reduce the cost of binary choice problems in crowdsourcing systems by using a probabilistic method. Our work is different in that (1) we consider the three factors that affect the profit of the crowdsourcing job, namely the value of the questions, the risk of obtaining an incorrect answer and the cost of assigning questions to workers; we consider both linear and non-linear model of the accuracy-cost relationship; (2) we model the accuracy of an answer to a question as a random variable instead of a fixed value; (3) we have optimized the algorithms to build more scalable and robust algorithms.

8. CONCLUSION

Crowdsourcing has attracted a great deal of interest in solving challenging problems by integrating human intelligence with algorithms. However, the system cannot be applied with unreliable data quality. Moreover, it is even harder to decide whether a problem is suitable to be solved by crowdsourcing. In this paper, we propose an online cost sensitive decision making method with novel data quality estimation. To show the effectiveness and efficiency of our method, we conduct extensive experiments over two real datasets on the Amazon Mechanical Turk. The experimental results show that our proposed method achieves a better accuracy-cost performance than all the existing methods. Moreover, our method is both scalable and robust such that it outputs reliable answers with diversified crowdsourcing data quality.

9. ACKNOWLEDGEMENT

The work of this paper was in part supported by a Faculty Award from Microsoft Research Asia and Singapore NRF grant R-252-000-511-281.

10. REFERENCES

[1] `http://www.mturk.com`.

[2] O. Alonso, D. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. In *SIGIR Forum*, volume 42, pages 9–15. ACM, 2008.

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

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

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

[6] P. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. *SIGKDD workshop*, pages 64–67. ACM, 2010.

[7] G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. *SIGIR*, 2011.

[8] A. Kittur, E. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. *SIGCHI*, pages 453–456. ACM, 2008.

[9] X. Liu, M. Lu, B. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: a crowdsourcing data analytics system. *VLDB*, 5(10):1040–1051, 2012.

[10] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Crowdsourced databases: Query processing with people. CIDR, 2011.

[11] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller Demonstration of Qurk: a query processor for humanoperators. *SIGMOD*, pages 1315–1318. ACM, 2011.

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

[13] A. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. CIDR, 2011.

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

[15] V. Raykar, S. Yu, L. Zhao, G. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *JMLR*, 11:1297–1322, 2010.

[16] J. Selke, C. Lofi, and W. Balke. Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. *VLDB*, 5(6):538–549, 2012.

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

[18] P. Welinder and P. Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. *CVPR workshop*, pages 25–32. IEEE, 2010.

[19] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. *MobiSys*, pages 77–90, 2010.