Rating Worker Skills and Task Strains in Collaborative Crowd Computing: A Competitive Perspective

George Trimponias*
Huawei Noah's Ark Lab
Hong Kong

Xiaojuan Ma*
Hong Kong University of Science and Technology
Hong Kong

Qiang Yang
Hong Kong University of Science and Technology
Hong Kong

ABSTRACT

Collaborative crowd computing, e.g., human computation and crowdsourcing, involves a team of workers jointly solving tasks of varying difficulties. In such settings, the ability to manage the workflow based on workers' skills and task strains can improve output quality. However, many practical systems employ a simple additive scoring scheme to measure worker performance, and do not consider the task difficulty or worker interaction. Some prior works have looked at ways of measuring worker performance or task difficulty in collaborative settings, but usually assume sophisticated models. In our work, we address this question by taking a competitive perspective and leveraging the vast prior work on competitive games. We adapt TrueSkill's standard competitive model by treating the task as a fictitious worker that the team of humans jointly plays against. We explore two fast online approaches to estimate the worker and task ratings: (1) an ELO rating system, and (2) approximate inference with the Expectation Propagation algorithm. To assess the strengths and weaknesses of the various rating methods, we conduct a human study on Amazon's Mechanical Turk with a simulated ESP game. Our experimental design has the novel element of pairing a carefully designed bot with human workers; these encounters can be used, in turn, to generate a larger set of simulated encounters, yielding more data. Our analysis confirms that our ranking scheme performs consistently and robustly, and outperforms the traditional additive scheme in terms of predicted accuracy.

CCS CONCEPTS

• **Human-centered computing → HCI design and evaluation methods**; **Collaborative interaction**; **HCI theory, concepts and models**; *Collaborative and social computing*.

KEYWORDS

Human computation; crowdsourcing; ranking, rating; collaborative.

ACM Reference Format:
George Trimponias, Xiaojuan Ma, and Qiang Yang. 2019. Rating Worker Skills and Task Strains in Collaborative Crowd Computing: A Competitive Perspective. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313569

*Equal contributions.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313569

1 INTRODUCTION

Human computation [38] and crowdsourcing [14] leverage the human processing power of online crowds to solve tasks that are beyond the capabilities of computer algorithms. Even though most of these tasks involve independent human workers, a significant portion of them requires the collaborative effort of two or more workers. One representative example is multi-player human computation games [40], originally proposed by Luis von Ahn as a money-free alternative to incentivize humans to perform such diverse tasks as image annotation [39], music tagging [27], collecting common sense facts [41], or locating objects in images [42]. These games usually require the agreement of the players[1] for data verification. Collaboration thus serves as a quality control mechanism. For example, in the ESP game [39], two randomly paired-up players simultaneously provide labels for a given image without communicating one with another. Collaboration also occurs in highly complex crowdsourcing tasks such as multilingual translation, to exploit complementary knowledge from different workers to generate high quality results [23, 32].

The performance of a joint task mainly depends on three factors. The first is task difficulty, which intuitively measures how hard it is for workers to generate outcomes intended by requesters. For example, in the ESP Game, a picture of a lion would be easy to name, since the majority of players are able to give specific labels, e.g., "lion". In contrast, an image of a tapir is more challenging, given that most people can only identify it as an "animal". The second factor is the skill of individual workers, which can be affected by their education, personal experience, etc. Normally, skilled workers are able to generate outputs of a higher quality compared to the low-skilled ones. The third, also the most distinctive factor of collaborative crowd computing, concerns the inter-worker interaction, i.e., the mechanism to determine the collective output given individual contributions. For instance, several human computation games rely on the agreement between two or more players as a verification mechanism. As a result, low-skilled players may have a greater impact on the game outcome. In the tapir image example, a knowledgeable player may key in the exact name of the tropical mammal, which, however, will not be accepted by the game if the partner fails to recognize it. Such crowd computing tasks can be best characterized as weak-link games [13]. The dual type, strong-link interactions in which more skilled workers may have a greater impact on the group outcome, also occurs in collaborative crowd computing. Crowd translation for instance only takes the

[1] Whenever there is no ambiguity, we use the terms worker and human interchangeably. In game contexts, we additionally use all three terms worker, player, and human interchangeably.

best translation derived from various answers generated by a team of workers [23].

To achieve the desired output quality and maintain worker engagement in collaborative crowd computing, one approach is to rate human participants based on their skill and tasks based on their difficulty. First, a rating system gives workers the incentive to devote more efforts to the tasks for potentially higher monetary rewards. This can, in turn, lead to better outcomes. Second, it allows for balanced collaborations, teaming up workers of similar skill levels. Third, it enables more proper task assignments, i.e., workers receive tasks that match their skills. Existing crowd computing projects often use an additive scoring scheme to assess worker performance [27, 39, 41]. Upon the completion of a collaborative task, each member of the group receives a score according to the joint performance. A project can rank all its workers based on their total scores summed over all the tasks they have performed, possibly in different groups. This scoring scheme is not an accurate indicator of personal skills, as a low-skilled worker that finishes significantly more tasks than a more skillful counterpart may probably end up with a higher score. Taking the average score can avoid this issue, but it still suffers from two significant shortcomings as a performance measure. First, it does not take into account the task difficulty, only the outcome. Intuitively, a high-quality output of a challenging task should be valued more than that of an easy one. Second, it ignores worker interaction, which can be a critical factor. Motivated by its pivotal role and the aforementioned shortcomings, some prior work has looked at ways of jointly rating workers and tasks in collaborative crowd computing [4, 7, 11, 20, 21, 33, 34, 36, 43–45]. These works typically assume fairly sophisticated models; the ratings of workers and/or tasks are subsequently inferred by maximizing a proper log-likelihood function.

In this work, we adopt an alternative perspective: instead of coming up with new schemes, we seek to leverage the state-of-the-art work on competitive games. Our model is the standard Bayesian rating scheme of TrueSkill for competitive games [12], which additionally treats the task as a fictitious worker that the team of humans plays against. We further discuss two online methods to estimate the ratings under this model: (1) an ELO rating scheme [9] that iteratively updates the ratings based on the expected and the actual outcome; and (2) Expectation Propagation (EP) [30, 31], a message-passing algorithm that performs approximate Bayesian inference. We evaluate the efficacy of the proposed rating framework using data from a simulated ESP Game with 100 participants conducted on Amazon Mechanical Turk (MTurk). We empirically show that our framework is more robust than the average scoring scheme, with the ELO scheme having the most consistent performance. We further assess the strengths and weaknesses of all methods. Our contributions can be summarized as follows:

- We show how to leverage TrueSkill's standard rating model in competitive games for the purpose of rating participants and tasks in collaborative projects. In this direction, we derive, for the first time, the update equations for the online schemes of ELO and EP in the collaborative scenario.
- We perform a human study on MTurk by properly adjusting the ESP game. Our study allows us to empirically assess the strengths and weaknesses of our proposed methods to the average scoring scheme, commonly employed in several human computation projects.
- We propose a novel experimental design, where we pair a human player with a carefully designed bot instead of a human player. This allows us, in turn, to generate a large set of encounters from a smaller dataset at a low cost.

The rest of the paper is structured as follows. We first survey related work. We then introduce our Bayesian TrueSkill-based rating model, and explore worker and task rating estimation via (1) a simple ELO rating scheme, and (2) an Expectation Propagation message passing algorithm. Following that, we present our human study on MTurk and the experimental results. Finally, we conclude the paper with discussions and future work.

2 RELATED WORK

We review existing scoring schemes in collaborative crowd computing, as well as rating and ranking in competitive games.

2.1 Scoring in Collaborative Crowd Computing

Additive scoring is the most common way to assess worker performance in existing crowd computing projects. In the ESP game for instance, a match results in a score varying from 50 to 150, with more specific labels rewarding a higher score. Similarly, in TagATune for tagging music clips [27] and Verbosity for collecting commonsense facts [41], both players earn 1 point if they reach an agreement or 0 points otherwise. These games order players according to their total scores across all the tasks they have performed, and thus the number of jobs completed may outweigh individual skill in the ranking. Computing the average score per task can avoid this issue, but still fails to consider the influence of task difficulty and intra-group dynamics. In contrast, although under a different setting, the more complicated schemes for ranking or rating players proposed for conventional competitive games may provide useful insights for collaborative crowd computing.

2.2 Ranking and Rating in Competitive Games

Assume a game with a finite set of players. A ranking is defined as a total preorder $\preceq$ on the set of players. For two players i, j, the relationship $i \preceq j$ denotes that player i ranks at most as high as player j. A rating, on the other hand, is a quantitative assessment index that denotes the skill level of a player. Higher-skilled players are associated with higher ratings in contrast to those with lower skills. Compared to rating, ranking merely shows that one player is better than another, but how much better remains unknown.

For ranking, there are three main approaches [3]. (1) The model-based approach determines the player ratings and subsequently calculates the induced ranking by sorting the computed ratings. (2) The score-based approach measures player performance with a scoring scheme (e.g., total or average score), and ranks players based on their score. (3) The axiomatic approach is inspired by axiomatic economic theory [35]. The idea is to determine a ranking scheme based on a set of axioms that must be satisfied.

Model-based rating methods usually view skill as a latent variable that characterizes the players. They assume that a fixed probability distribution (model) governs the outcome of an encounter between two players of given skills [3]. In particular, $F(r_i, r_j)$ denotes the

probability that player i with skill r_i beats player j with skill r_j; note that $F(r_i, r_j) = 1 - F(r_j, r_i)$. The probability distribution is computed a priori based on historical data. It primarily follows two models: the Thurstone Case V model [37] assumes the Gaussian distribution $F(r_i, r_j) = \Phi(r_i - r_j)$, where $\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$, whereas the Bradley-Terry model [2] assumes the logistic variant $F(r_i, r_j) = \frac{1}{1+e^{-(r_i - r_j)}}$. Subsequently, one can use a maximum likelihood approach to estimate the players' skills based on paired-comparison data, and rank them accordingly. In practice, the skill parameters are called ratings, and the function F is the rating function. Under the linear paired comparison model [6], we assume that the result of an encounter between any two players depends only on their rating difference.

Given that maximum likelihood estimation is complex and requires the complete paired-comparison data, an alternative is the ELO rating system [9]. Under this online scheme, every player is associated with an ELO rating. Whenever an encounter between two players occurs, the scheme computes the probability that each of the two players wins. It subsequently rewards or punishes players, depending on whether they performed better or worse than expected. The ELO rating system has been widely deployed for player ranking in real-world competitive games such as chess, association football, American football, basketball, Major League Baseball, Scrabble, and snooker.

The above rating methods do not directly handle the uncertainties over the ratings. Hence, prior research has proposed Bayesian rating schemes to tackle this [10]. The idea is to model the belief about a player's skill as a Gaussian belief distribution characterized by a mean μ and a standard deviation σ. Recently, TrueSkill has been introduced as a more general online Bayesian rating system which can further handle (1) teams of players rather than just individual players, and (2) settings where more than two teams compete simultaneously [12].

Of course, the topic of ranking and rating from pairwise comparisons with items of various difficulties has attracted considerable attention in the machine learning community, see e.g., [4, 7, 11, 20, 21, 33, 34, 36, 43–45]. In particular, several sophisticated studies within the machine learning community investigate the problem from a theoretical perspective. In this work, we take a different approach by looking into fast online schemes inspired by competitive games that are relatively simple to implement and deploy. Our interest in TrueSkill is due to its enormous success in both identifying and tracking the skills of gamers in Xbox Live by Microsoft, and its ability to properly match gamers into competitive matches. Our goal is thus to examine how the most popular and successful rating schemes in competitive games can be adapted to ameliorate ranking in collaborative crowd computing in a fast online manner and empirically assess them, rather than investigate new theory and propose novel sophisticated ranking schemes.

Finally, note that there has been extensive research within the crowdsourcing community aimed at improving worker performance, including different filtering mechanisms, the use of golden standards, explicitly verifiable questions as well as gamification with scoring mechanisms aimed at improving data quality [1, 8, 15, 24, 25, 29]. Such work is complementary to ours, since our sole focus in on schemes for jointly rating the workers' skills and task strains.

3 BAYESIAN RATING MODEL

In this section, we introduce a TrueSkill-based Bayesian rating model and two online rating estimation methods for assessing works and tasks in collaborative crowd computing. Our model takes into account both the worker interaction and the task difficulty.

3.1 High-level Overview

Before examining our model in detail, we first provide a high-level overview. Since both the worker skills and the task strains are unknown, we represent them as random variables, which are characterized by two important parameters, the mean and the variance. Higher values for the former are associated with a belief for a higher worker skill (or task strain) compared to lower values. On the other hand, the variance describes the degree of certainty in our belief, with lower values corresponding to higher certainty.

Given the random variables, a model describes how the various variables are related to each other. In collaborative tasks, the model has in general two parts: (1) the first part describes how the performance of the team of workers depends on their individual skills; (2) the second part describes when the team of workers solves a task of a given strain successfully or not.

Since the random variables are unknown, our goal is to then infer their values given the model from a set of encounters, where teams of workers try to solve a task. There are two general families of methods. Online methods update the random variables after each encounter, whereas offline methods do the update based on the entire dataset. Offline methods can in principle lead to optimal estimates, but online methods have the advantage of being faster and simpler. Practical rating schemes for competitive games have traditionally relied on online methods (see section 2.2), and we take a similar approach in our work.

3.2 Model

Our scheme is inspired by TrueSkill [12] and by prior work on question difficulty estimation in community question answering services [28].

Let i and j be the two workers[2] with ratings r_i and r_j, respectively, that collaboratively try to solve human computation task t. We denote the encounter between i and j when they perform task t as $< i, j, t >$. TrueSkill assumes that the belief about the rating r_i of player i follows a normal distribution $\mathcal{N}(r_i; \mu_i, \sigma_i^2)$ with mean μ_i and variance σ_i^2. When i performs a task t, its performance p_i will follow a normal distribution $\mathcal{N}(p_i; r_i, \delta^2)$. The variance δ^2 is due to the fact that the performance of a player may vary due to assorted factors that we cannot directly model. When players i and j collaboratively perform task t, their joint performance p will depend on the type of interaction between them. As we argued in the Introduction, in some human computation games, the inter-group interaction is asymmetric and takes the form of a weak-link game, whereas certain crowdsourcing tasks such as collaborative design can be modeled as the dual strong-link game. In the former

[2]For teams consisting of just two workers, we use the terms team and pair interchangeably.

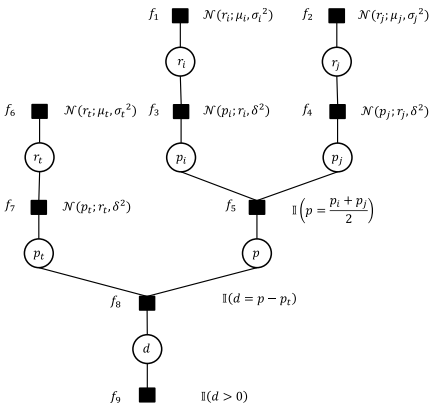

Figure 1: Factor graph for an encounter $< i, j, t >$ where the pair of workers (i, j) correctly solves task t.

(latter) case, workers of higher (lower) skill have a higher impact on the task output. In our work, we adopt a simpler symmetric approach where the interaction dynamics is simply modeled as the average of the player performances. This is similar to the model that TrueSkill considers to measure the performance of teams consisting of several players. Despite being less accurate, this model has the advantage of being linear (as opposed to the non-linear minimum or maximum operators), and allows for tractable inference.

In order to take into account the second feature of collaborative human computation, i.e., the task difficulty, we properly adapt the approach proposed in [28]. The idea is to model each task as an additional player, whose rating and performance also follow normal distributions. After encounter $< i, j, t >$ has taken place, we review the task output that the players jointly produced. If the pair of workers produced high quality output (e.g., informative or specific labels in the ESP game), we assume that the pair beat the fictitious worker that represents task t. In the opposite case, the pair loses to the fictitious worker. We do not consider the case of a draw. This approach actually turns the collaborative task into a competitive game. Note that we use the same variance δ^2 for the performance of the human players and the task for simplicity, but we could in principle also adopt different variances depending on the collaborative project at hand.

Figure 1 schematically depicts an example factor graph in the case where the pair of workers answers the question correctly, or "beats" the question. It has the form of a factor graph [26], which is basically a bipartite graph that consists of variables (circles) and nine factor nodes $f_1, \ldots, f_9$ (squares). The structure of the factor graph gives information about the dependencies of the factors.

Note that we can extend the model of Figure 1 in three ways. First, we can model the fact that ratings vary over time by assuming a Gaussian drift of skill between time steps given by $Pr(r_i^t | r_i^{t-1}) = \mathcal{N}(r_i^t; r_i^{t-1}, \tau^2)$ [5, 12]. Second, we can handle the case of more players by just adding one rating node r_k and one performance node p_k for each additional player k, and then defining the joint performance to be equal to the average of the individual player performances, i.e., $p = \frac{p_i+p_j+p_k+\cdots}{n}$, where n the number of players. Third, we can handle situations where the task output is not binary (correct versus incorrect or specific versus generic), but may take on one of $L > 2$ possible values (e.g., low, medium, or high specificity). To accommodate this, we can extend Figure 1 by considering $L-1$ tasks rather than just one, where each task also consists of a rating node r_t^l and one performance node p_t^l, where $2 \leq l \leq L$. Furthermore, we add $L-1$ difference nodes d^l that correspond to the difference between the team performance p and task performance p_t^l, where $2 \leq l \leq L$. Whenever an encounter occurs, we determine the level l^o of the outcome that the team of workers produced. Subsequently, we assume that the team beats all the tasks that correspond to level equal to or less than l^o, and loses to all task nodes that correspond to levels greater than l^o, i.e., $d_l > 0$ if and only if $l \leq l^o$. The natural interpretation for this is that if a team of workers generates an outcome of level l^o, where $1 \leq l^o \leq L$, this is equivalent to assuming that the team has (1) beaten all task levels lower than or equal to level l^o, and (2) has lost to all task levels greater than l^o. Note also that we do not need a node for the lowest level, since the team of workers is always guaranteed to beat that level; for instance, in a correct versus incorrect scenario, the team of workers can do no worse than giving an incorrect answer.

As a final remark, our model assumes that each worker is characterized by a single rating $\mathcal{N}(r_i; \mu_i, \sigma_i^2)$. In reality, a worker may have different skills for different tasks; for instance, they may be very knowledgeable about sports but know little about books. It is possible to consider multiple ratings per worker, one per task. In the remainder of the paper, as a proof of concept, we use a single rating per worker, which can be viewed as an indicator of the worker's average ability to provide high quality output on any task.

3.3 ELO Rating Scheme

Let $< i, j, t >$ be the encounter between i and j when they jointly perform task t. The task serves as a fictitious participant that plays against the human pair as shown in the graph model of Figure 1. Before the encounter takes place, assume the ratings of the workers are r_i and r_j, and the rating of the task node is r_t. The combined rating for the pair of workers will be according to our model $r = \frac{r_i+r_j}{2}$. The task node exhibits a performance p_t that is normally distributed around r_t with a fixed variance δ^2, while the performance p of the human pair node is normally distributed around r with a fixed variance $\frac{\delta^2+\delta^2}{4} = \frac{\delta^2}{2}$. But then the random variable $p - p_t$ will be normally distributed around $r - r_t$ with a variance $\delta^2 + \frac{\delta^2}{2} = \frac{3\delta^2}{2}$. The probability that the pair of workers beats the task will then be equal to $Pr(p > p_t | r, r_t) = \Phi(\frac{\sqrt{2}(r-r_t)}{\sqrt{3}\delta})$.

Our online ELO rating scheme will update the pair rating and the task rating after the encounter to reflect the game outcome, based on the premise that an entity should be "punished" if it did worse than expected, or boosted if it exceeded expectations. The expected outcome (in terms of the pair of workers beating the task node) will be $expected^{<i,j,t>} = \Phi(\frac{\sqrt{2}(r-r_t)}{\sqrt{3}\delta})$. The actual outcome $actual^{<i,j,t>}$ will be 1 if the pair of humans beats the task, and 0 otherwise. The resulting ELO update is:

$$r' = r + K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$
$$r'_t = r_t - K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$

Parameter $K > 0$ is a positive real number called the K-factor fixed in advance that is related to the maximum adjustment from a single encounter. Note that we increase the rating of an entity if it does better than expected, or we decrease it in the opposite case; the adjustment is proportional to (1) the K-factor, and (2) the difference between the actual and the expected outcome.

The update rule for the human pair can be rewritten as:

$$\frac{r_i' + r_j'}{2} = \frac{r_i + r_j}{2} + K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>}) \Leftrightarrow$$
$$r_i' + r_j' = r_i + r_j + 2 \cdot K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$

But then we can split the $2 \cdot K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$ update amount equally to the two workers, so that we have:

$$r_i' = r_i + K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$
$$r_j' = r_j + K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$

Regarding convergence, in the presence of balanced matches, the ELO ratings converge exponentially fast to the actual player ratings [16]. If, on the other hand, the schedule is unbalanced, convergence can be very slow and a plethora of equilibria can emerge, depending on the exact initial distribution of initial ratings. Note that despite these theoretical concerns, ELO is a very robust scheme and widely used in several popular competitive games (see section 2.2). Algorithm 1 describes the update rules for ELO rating.

ALGORITHM 1: ELO Rating for Bayesian Factor Graph of Figure 1.

Input: An encounter $< i, j, t >$, prior ratings r_i, r_j, r_t, performance variance δ^2, and K-factor K.

Output: Updated ratings r_i', r_j', r_t' after encounter $< i, j, t >$.

Compute:

$$expected^{<i,j,t>} = \Phi(\frac{\sqrt{2}(r - r_t)}{\sqrt{3}\delta})$$

$$actual^{<i,j,t>} = \begin{cases} 1, & \text{if pair } (i, j) \text{ beats task } t \\ 0, & \text{otherwise} \end{cases}$$

Update:

$$r_i' = r_i + K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$
$$r_j' = r_j + K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$
$$r_t' = r_t - K \cdot (actual^{<i,j,t>} - expected^{<i,j,t>})$$

3.4 Bayesian Rating Scheme

We can estimate the unknown ratings with Expectation Propagation (EP), a deterministic approximate inference technique [30, 31]. Our goal is to determine the posterior distributions over the variables r_i and r_j, after encounter $< i, j, t >$ has occurred. EP is essentially an online message-passing algorithm for the factor graph of Figure 1.

Before going into the details, we briefly examine inference algorithms on factor graphs, i.e., graphs where the global function can be factorized into a product of local functions (factors) that depend on the node variables. The main algorithm for this class of graphs is the sum-product algorithm [26], which operates on the factor graph and attempts to compute various marginal functions. The basic idea is that if the graph is acyclic (as in our case), then it is possible to execute a message-passing algorithm where messages are passed from factor nodes to variable nodes, and vice versa. The algorithm terminates once two messages have passed over every edge, one in each direction. The message passing procedure for continuous variables is described by the following equations:

$$\textit{marginal distribution: } p(v) = \prod_{f \in n(v)} m_{f \to v}(v)$$

$$\textit{variable to factor: } m_{v \to f}(v) = \prod_{h \in n(v) - f} m_{h \to v}(v)$$

$$\textit{factor to variable: } m_{f \to v}(v) = \int \cdots \int f(V) \prod_{u \in n(f) - v} m_{u \to f}(u) dV_{-v}$$

where $m_{v \to f}(v)$ the message sent from variable v to factor f, $m_{f \to v}(v)$ the message sent from factor f to variable v, $n(x)$ is the set of neighbors of node x, V the set of variables connected to factor f, and V_{-v} the components of vector V except for variable v.

Of particular importance in our framework are the sum and difference of normal distributions. Assume two normal distributions $\mathcal{N}(x_1; \mu_1, \sigma_1^2)$ and $\mathcal{N}(x_2; \mu_2, \sigma_2^2)$. Their sum will be distributed according to $\mathcal{N}(x_1 + x_2; \mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$, and their difference according to $\mathcal{N}(x_1 - x_2; \mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2)$. Moreover, in the message-passing algorithm, we often need to compute the product of two edge messages, or the ratio of a node message to an edge message. Since the messages in our framework are normal densities, given two normal densities $m_1 = (\mu_1, \sigma_1^2)$ and $m_2 = (\mu_2, \sigma_2^2)$, their product will be $m_1 \cdot m_2 = (\frac{\sigma_2^2 \cdot \mu_1 + \sigma_1^2 \cdot \mu_2}{\sigma_1^2 + \sigma_2^2}, \frac{\sigma_1 \cdot \sigma_2}{\sqrt{\sigma_1^2 + \sigma_2^2}})$ and their ratio $m_1 \div m_2 = (\frac{\sigma_2^2 \cdot \mu_1 - \sigma_1^2 \cdot \mu_2}{\sigma_2^2 - \sigma_1^2}, \frac{\sigma_1 \cdot \sigma_2}{\sqrt{|\sigma_2^2 - \sigma_1^2|}})$.

The only complication is created by factor f_9, which is non-Gaussian. To tackle this, we approximate the non-Gaussian posterior marginal $p(d)$ of node d with a normal distribution, using the technique of moment matching [31]. This technique consists in computing the mean and variance of the non-Gaussian message, and subsequently approximating it with a Gaussian distribution with the same first two moments. Moment matching provably minimizes the Kullback-Leibler divergence for Gaussian distributions. Assuming a message $m_{f_8 \to d} = (\mu_{f_8 \to d}, \sigma_{f_8 \to d}^2)$, the approximate posterior marginal $\hat{p}(d)$ will have mean and variance [19]:

$$\hat{\mu}_d = \mu_{f_8 \to d} + \sigma_{f_8 \to d} \cdot u\Big(-\frac{\mu_{f_8 \to d}}{\sigma_{f_8 \to d}}\Big) \quad (1)$$

$$\hat{\sigma}_d^2 = \sigma_{f_8 \to d}^2 \cdot \Big(1 - w\Big(-\frac{\mu_{f_8 \to d}}{\sigma_{f_8 \to d}}\Big)\Big) \quad (2)$$

where

$$u(x) = \begin{cases} \frac{\phi(x)}{1 - \Phi(x)}, & \text{if } x < 0 \\ -\frac{\phi(x)}{\Phi(x)}, & \text{if } x > 0 \end{cases}, w(x) = u(x) \cdot (u(x) - x).$$

Note that ϕ and Φ denote the probability density function and the cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$, respectively. Algorithm 2 presents the message-passing inference process.

Finally, note that even though Figure 1 and Algorithm 2 assume a single encounter, it is possible to assume multiple encounters, as long as the underlying factor graph does not have cycles [30, 31].

ALGORITHM 2: Approximate Inference for Bayesian Factor Graph of Figure 1.

Input: An encounter $< i, j, t >$, the prior ratings (μ_i, σ_i^2), (μ_j, σ_j^2), (μ_t, σ_t^2), and the performance variance δ^2.
Output: The posteriors $p(r_i \mid < i, j, t >)$, $p(r_j \mid < i, j, t >)$, and $p(r_t \mid < i, j, t >)$.

//Forward messages
1. $m_{f_1 \to r_i} = (\mu_i, \sigma_i^2)$, $m_{f_2 \to r_j} = (\mu_j, \sigma_j^2)$, $m_{f_6 \to r_t} = (\mu_t, \sigma_t^2)$
2. $m_{r_i \to f_3} = m_{f_1 \to r_i}$, $m_{r_j \to f_4} = m_{f_2 \to r_j}$, $m_{r_t \to f_7} = m_{f_6 \to r_t}$
3. $m_{f_3 \to p_i} = (\mu_{r_i \to f_3}, \sigma^2_{r_i \to f_3} + \delta^2)$, $m_{f_4 \to p_j} = (\mu_{r_j \to f_4}, \sigma^2_{r_j \to f_4} + \delta^2)$, $m_{f_7 \to p_t} = (\mu_{r_t \to f_7}, \sigma^2_{r_t \to f_7} + \delta^2)$
4. $m_{p_i \to f_5} = m_{f_3 \to p_i}$, $m_{p_j \to f_5} = m_{f_4 \to p_j}$, $m_{p_t \to f_8} = m_{f_7 \to p_t}$
5. $m_{f_5 \to p} = m_{p_i \to f_5} + m_{p_j \to f_5}$, $m_{p \to f_8} = m_{f_5 \to p}$
6. $m_{f_8 \to d} = m_{p \to f_8} - m_{p_t \to f_8}$
7. $\hat{p}(d) = (\hat{\mu}_d, \hat{\sigma}_d^2)$ (Equations 1 and 2)

//Backward messages
8. $m_{d \to f_8} = \frac{\hat{p}(d)}{m_{f_8 \to d}}$
9. $m_{f_8 \to p} = m_{p_t \to f_8} + m_{d \to f_8}$, $m_{f_8 \to p_t} = m_{p \to f_8} - m_{d \to f_8}$
10. $m_{p \to f_5} = m_{f_8 \to p}$, $m_{p_t \to f_7} = m_{f_8 \to p_t}$
11. $m_{f_5 \to p_i} = 2 \cdot m_{p \to f_5} - m_{p_j \to f_5}$, $m_{f_5 \to p_j} = 2 \cdot m_{p \to f_5} - m_{p_i \to f_5}$
12. $m_{p_i \to f_3} = m_{f_5 \to p_i}$, $m_{p_j \to f_4} = m_{f_5 \to p_j}$
13. $m_{f_3 \to r_i} = (\mu_{p_i \to f_3}, \sigma^2_{p_i \to f_3} + \delta^2)$, $m_{f_4 \to r_j} = (\mu_{p_j \to f_4}, \sigma^2_{p_j \to f_4} + \delta^2)$, $m_{f_7 \to r_t} = (\mu_{p_t \to f_7}, \sigma^2_{p_t \to f_7} + \delta^2)$

//Posterior rating marginals
14. $p(r_i \mid < i, j, t >) = m_{f_3 \to r_i} \cdot m_{f_1 \to r_i}$, $p(r_j \mid < i, j, t >) = m_{f_4 \to r_j} \cdot m_{f_2 \to r_j}$, $p(r_t \mid < i, j, t >) = m_{f_7 \to r_t} \cdot m_{f_6 \to r_t}$

However, in practice the player encounters can form cycles, for instance when every player is paired with every other player. For this reason, we favor our simpler model in Figure 1, which considers a single encounter and avoids cycles in the underlying factor graph.

4 EXPERIMENT

We test our framework using the ESP game as a case study, as it is typically viewed as the most influential human computation game.

4.1 System and Task Design

Figure 2 depicts the graphical user interface of the game used in our experiment. It closely resembles the original Google Image Labeler, the commercially licensed version of the ESP game [17]. Our implementation is web-based and written in JavaScript and PHP, and also uses an Apache HTTP web server.

The rules in our implementation have both similarities and differences, compared to the original version. First, players have 40 seconds to match on a label for any image, while in the original game players were given 120 seconds to try to match as many pictures as possible. This is related to the fact that we do not include a passing mechanism, as we discuss later. Second, if a pair of players match on a label, they both earn a score ranging from 10 to 100 depending on the label specificity, which they can immediately see upon agreement. Third, if time is out, players win 0 points each (no agreement reached), and they move on to the next image. Fourth, for every picture we additionally include a short list of 0 to 2 taboo (off-limits) labels that players are not allowed to provide. As opposed to the original game, our version does not allow players to skip an image and move on to the next one, because we need to know their performances in all tasks to be able to rate them better.

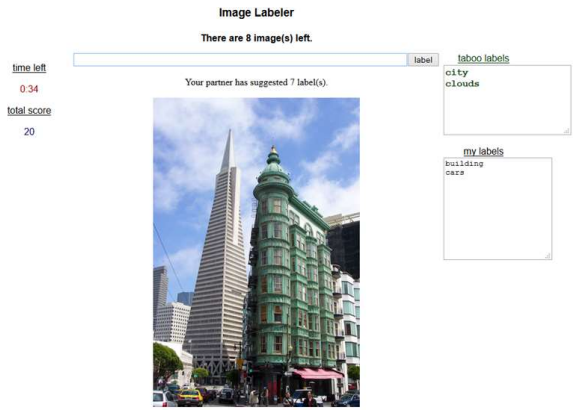

Figure 2: The interface of our simulated ESP game.

Regarding the actual tasks, we select 10 pictures from the Internet (Figures 3a-3j). Our tasks satisfy two properties: (1) they cover various common themes in the original ESP game such as animals, food, Hollywood, architecture, and science, and (2) they have both a generic and a specific component. For instance, Tapir accepts generic labels such as "animal" or "large" as well as the specific animal name. Transamerica can be labeled with the generic label "skyscraper" or the specific label "San Francisco".

Figure 3: Pictures from our MTurk experimental evaluation.

4.2 Experiment Design

An interesting design of our experiment is that we do not directly ask two humans to play with each other. Instead, we develop a bot that simulates a high-skilled player; a human worker is then paired with our bot. The motivation behind this decision is threefold. First,

a bot allows us to control the label scores. It is practically impossible to know in advance all possible labels that two human players may agree on and assign scores to them. A bot with a predefined set of labels with associated scores can tackle this issue. Of course, this requires that the label set is large enough and carefully selected; we discuss this point in more detail later. Second, it is easier to infer human players' true skill as the baseline, since the bot does not act as the weak link in the collaboration. Third, as we discuss below it is possible to simulate a very large number of human encounters based on the bot responses, which would be very difficult and costly to do directly. Interestingly, our design novelty is appropriate for any collaborative task where the goal is to generate many encounters at a lower cost. This method can be applied to real collaborative crowd computing tasks in cold-start situations to generate an initial assessment of task difficulty and worker skill.

We did not mention to the players they are playing against a bot; instead, we instructed them that they would be paired with another human player. In addition, we generate the time period between two consecutive labels randomly from 1 to 5 seconds to give the human workers the impression that they are playing with a real human instead of a bot, especially given that they only play together for 10 images. The bot provides 12-14 representative labels per image that we carefully select so that: (1) they range from very specific to very generic in decreasing order to prioritize high-quality labels, and (2) they cover the main objects or semantic aspects of the image. This ensures that both high-skilled and low-skilled players can reach an agreement with the bot. In addition, we created variations of these representative labels, such as singular and plural form, capitalized first letter for names etc. For each image, the bot generates the labels in decreasing order of specificity, because in game-theoretical terms it is a dominant strategy for a rational player to output the labels in that order [18]; otherwise, the pair of players might first match on a label with lower specificity (and thus lower score). The label selection was based on a preliminary pilot study on MTurk, where we asked 20 workers to provide both generic and specific labels for each picture. Following that, we asked two researchers to rank the labels of each image from the most generic to the most specific independently. The Spearman's coefficient of 0.82 (p-value < 0.05) suggests strong inter-rater reliability. We invited a third researcher into the group discussion to resolve disagreement and come up with the final assessment. The top six or seven labels from the generic and specific categories then form the label set for each image. We subsequently assigned scores using even intervals between consecutive terms, so that the most generic picture gets a score of 10 and the most specific a score of 100, and we rounded the remaining scores to the nearest multiple of 10.

Finally, we upload our game as a human intelligence task (HIT) on Amazon Mechanical Turk (MTurk). Upon successful completion of the HIT, i.e., reaching an agreement in at least eight out of the ten images presented in a randomized order, workers earn $0.25, for an average hourly rate per worker of about $3.0. To motivate workers to put more effort into the game, we include a bonus of $5 to the 5 workers with the highest total scores. For quality control purposes, we only allowed MTurk workers with acceptance rates for submitted tasks of at least 80%, and with at least 10 submitted tasks in their history. Note that a worker may be truthful but still have low skill, the two notions are not contradictory. Given that our study involves specific labels, we restrict our tasks to MTurk workers in the US.

4.3 Data Collection

In total, we accepted data from 100 workers. For each worker, we recorded the set of labels provided for each image until either a match or a timeout occurred, as well as the corresponding timestamp. From these results, we were able to generate two datasets. (1) The first dataset, which we call BOT-PARTNER, contains the 10×100=1,000 encounters between the population of the 100 players and the bot. (2) The second dataset, which we name HUMAN-PARTNER, contains the (hypothetical) encounters between any two players from the 100 workers. It is indeed possible to simulate these encounters, based on the (label, timestamp) recorded information. One problem is that not all encounters are valid. For instance, assume that one player reaches an agreement with the bot in the first 5 seconds, whereas it takes 35 seconds for another player to reach an agreement for the same image. In the simulated encounter where these two players are paired together, there may be no match because the first player only played for a limited time. To account for this, we ignore (invalid) encounters that result in no match, when at least one of the two players played the corresponding image for less than 40 seconds (this happens when the player did not manage to match on any label). By construction, HUMAN-PARTNER is not an unbiased dataset: encounters involving players who match with the bot in BOT-PARTNER may be ignored; on the other hand, encounters where both players did not match with the bot can never be ignored (they either match on a new label or they do not match yielding a score of 0). Given this bias, we expect any given method will not produce identical rankings on the two datasets.

In particular, HUMAN-PARTNER contains 17,357 (valid) encounters in total (instead of $10 \times \binom{100}{2} = 49,500$); furthermore, by construction it has a bias toward low-skilled workers, as it takes these workers longer to come to a match with the bot and as a result their encounters are more likely to be kept. This is an inherent limitation of this design scheme that we also discuss in Section 5.4. Note that in BOT-PARTNER a player can only generate a match on the predetermined 10-12 labels that the bot is programmed to provide; in contrast, in HUMAN-PARTNER matches occur on labels that both human workers provided, which may be different from the predetermined labels of the bot. The total number of matched labels over all 17,357 encounters that were different from the bot's labels were 129: 22 for "Pumpernickel", 10 for "Tapir", 13 for "Hunger Games", 7 for "Captcha", 31 for "Declaration", 10 for "Soba Noodles", 14 for "Transamerica", 8 for "Albert Einstein", 1 for "Cheetah", and 13 for "Phi-beta-sigma". Note that intuitively easier tasks such as "Cheetah" or "Albert Einstein" resulted in fewer new matches than harder tasks such as "Pumpernickel", or tasks that are associated with a plethora of objects and/or concepts such as "Declaration". In order to assign a score to matched labels not in the predetermined set, we asked the previous three raters that we used to select labels for the bot to score these new labels from 10 to 100 in multiples of 10 according to how specific they were. The value of 60 was used as a cutoff: labels with specificity 10-50 were classified as generic, whereas labels with specificity 60-100 were classified as specific. The average of the three values was then selected as the label's

score. For labels where the three values differed significantly, we invited a fourth rater to confirm and finalize the score.

Table 1 provides descriptive statistics about the matched labels (specific, generic, or no match) in BOT-PARTNER for each of the ten tasks. Note that the first three columns sum up to 100, i.e. the number of workers since in BOT-PARTNER each worker is involved in one encounter per task. Images where fewer players managed to match on a specific label are usually associated with a higher average matching time (the Pearson's correlation coefficient is -70.65%). These images naturally correspond to harder tasks; for instance, "Transamerica" or "Pumpernickel" are harder than "Albert Einstein" or "Cheetah". Alternatively, the Specific:Generic ratio can serve as an indicator of the image's difficulty. Easier tasks tend to have high values for that ratio; for instance, for "Cheetah" this ratio stands at 24. Difficult tasks, on the other hand, are associated with a lower ratio; for example, for "Transamerica" it stands at 0.59.

Table 1: Statistics for the ten tasks in BOT-PARTNER

	Specific	Generic	No match	Matching time (sec)
Pumpernickel	45	40	15	12.71 ± 7.34
Tapir	76	22	2	11.44 ± 6.95
Hunger Games	90	3	7	8.93 ± 7.68
Captcha	61	36	3	9.69 ± 8.06
Declaration	69	20	11	14.82 ± 8.58
Soba Noodles	71	23	6	9.15 ± 6.00
Transamerica	35	59	6	13.33 ± 12.08
Albert Einstein	93	6	1	8.07 ± 7.93
Cheetah	96	4	0	7.15 ± 8.37
Phi-beta-sigma	76	20	4	11.78 ± 8.79

Table 2 summarizes the descriptive statistics of matched labels for HUMAN-PARTNER. We have added a column "Invalid" that contains the number of invalid encounters. Note that there are totally $\binom{100}{2} = 4,950$ encounters for every image; hence, the first four columns sum up to 4,950. It is not surprising that most encounters are invalid, given that the majority of encounters in BOT-PARTNER result in a match and thus last less than 40 seconds. A higher Specific:Generic ratio is again associated with easier tasks, which are in turn associated with lower matching times. However, we notice in Tables 1 and 2 that for any image the ratios between the specific and generic matches are not necessarily the same in BOT-PARTNER and HUMAN-PARTNER. The percentage of generic matches over all 10 tasks is 24.5% in the former but 28.23% in the latter dataset. The reason for this discrepancy is that in the first dataset the bot simulates a highly skilled player, so encounters are more likely to occur on specific labels. Thus, even though the two datasets are based on the same 10 tasks and the same set of 100 workers, they may result in different workers' skills and task strains.

Note that the above tables do not contain information about how individual workers did across the 10 tasks. Intuitively, we expect that high-skilled workers tend to produce more specific tasks. To test for this, we considered the ELO rating (see Section 5) for the 100 workers based on BOT-PARTNER. We then isolate the top-25 workers as well as the bottom-25 ones, and subsequently simulate encounters among the top 25 workers only; or, among the top 25 and the bottom 25 workers; or, among the bottom 25 workers only.

Table 2: Statistics for the ten tasks in HUMAN-PARTNER

	Specific	Generic	No match	Invalid	Matching time
Pumpernickel	413	451	41	4045	14.65±8.16
Tapir	819	978	6	3147	11.66±7.63
Hunger Games	1405	33	20	3492	10.02±5.28
Captcha	1043	400	20	3487	11.84±7.25
Declaration	806	199	83	3862	18.19±9.30
Soba Noodles	1163	922	5	2860	11.45±7.37
Transamerica	437	1497	51	2965	15.12±9.30
Albert Einstein	2580	61	4	2305	8.73±6.56
Cheetah	1771	66	0	3113	8.42±5.68
Phi-beta-sigma	1816	213	54	2867	13.08±9.13

We only keep the valid encounters. The percentage of encounters resulting in specific matches stands at 88.31%, 61.20%, and 31.44%, respectively, which is in full accordance with our intuition. We also observe that the percentage of valid encounters that result in no match is 0.57%, 3.44%, and 9.44%, respectively. This suggests that encounters involving low-skilled workers are significantly more likely to result in a generic match or no match at all.

5 RATING RESULTS AND ANALYSIS

For each encounter in either dataset, we asked three raters from our research group who were blind to the conditions to vote whether the quality of the matched label (if a match occurred) is high (e.g., informative and specific) or low; we combine the answers with majority voting. We then rate the workers with the following three methods: (1) AVERAGE score, where a specific match yields 1 point and a generic match or a no match yields 0 points, (2) ELO rating of Algorithm 1 with $K = 24.0$, and (3) EP of Algorithm 2 with $\mu_0 = 25, \sigma_0 = \frac{25}{3}, \delta = \frac{\sigma_0}{2}$ (based on [12]). In BOT-PARTNER, we consider that each team consists of a single human player, whereas in HUMAN-PARTNER each team consists of a pair of workers. We compare our proposed rating framework with the baseline average scoring scheme in terms of consistency, accuracy, and robustness.

5.1 Output Similarity across Rating Schemes

First, we investigate how similar the three methods perform on each dataset. To compare the methods, we consider the rankings that the worker ratings and task strains induce, and subsequently calculate the similarities between them using Spearman's rank correlation coefficient[3] [22]. The correlation metric in this context makes sense as a way of capturing the degree to which the different methods agree on the generated rankings; intuitively, they should be positively correlated since they rank based on the same data.

Table 3 contains our results. In general, HUMAN-PARTNER is more challenging for AVERAGE, because (1) not all encounters are valid, and (2) it has a bias toward low-skilled workers. As a result, the task schedules are unbalanced, and the number and type of encounters for each worker varies. In comparison, rankings produced by the three methods have stronger correlations in BOT-PARTNER, where the schedule is balanced, i.e., each player plays with the bot in all 10 tasks for a total number of 10 encounters as

[3]The p-values of all reported correlation coefficients are very low due to the high sample size and we do not report them here.

Table 3: Spearman's correlation coefficient between worker rankings and task rankings from different methods

	BOT-PARTNER		HUMAN-PARTNER	
	WORKER	TASK	WORKER	TASK
AVERAGE-ELO	95.66%	89.09%	73.89%	67.27%
AVERAGE-EP	75.90%	53.94%	50.85%	30.91%
ELO-EP	85.94%	76.97%	85.63%	74.55%

opposed to the less regular second dataset. ELO and EP are generally strongly correlated, since they both rely on the same Bayesian factor graph of Figure 1. One possible reason for the weaker correlation between AVERAGE and EP is that we perform EP after each individual encounter instead of simultaneously optimizing over multiple encounters. This may make EP less accurate, as it needs both the mean and variance. In contrast, the simpler update rules of ELO using only the mean can produce results closer to those of AVERAGE. We also note that the rankings that different methods produce are less correlated for the images than for the workers, possibly due to the very low number of tasks (10) compared to the higher number of workers (100). This is more pronounced for EP.

Next, we investigate whether the same method yields consistent rankings when applied to BOT-PARTNER and HUMAN-PARTNER based on Spearman's correlation coefficient. Using the rank correlation coefficient to compare rankings in different datasets is tricky, because a higher correlation coefficient does not necessarily imply a better ranking. For example, using a random but fixed ranking across different datasets yields a correlation coefficient of 1, even though the underlying random ranking may be totally unrelated to the ground truth. Despite this, Spearman's correlation coefficient can serve as a necessary condition for consistency: given the two datasets refer to the same set of workers and tasks, we intuitively expect similar rankings in the two datasets.

Table 4 shows the correlation coefficients for worker and task rankings for all methods. We observe that worker rankings are positively correlated at around 65%-70%, while task rankings are very strongly correlated at above 90%, with AVERAGE being the least consistent. Task rankings are probably more consistent because a task participates in many more encounters than a worker, so its strain can be estimated more accurately. The positive correlations suggest that dataset HUMAN-PARTNER that we generated by emulation is consistent with the original dataset BOT-PARTNER and can be used in conjunction with it.

Table 4: Spearman's correlation coefficient between worker rankings and task rankings in two datasets

	WORKER RANKINGS	TASK RANKINGS
AVERAGE	65.61%	90.30%
ELO	67.60%	95.15%
EP	71.65%	92.73%

To gain insight into the different rankings, we consider the image "Phi-beta-sigma" in HUMAN-PARTNER. AVERAGE ranks it as the 5th easiest, whereas ELO ranks it as the 3rd easiest. The reason for this discrepancy is that AVERAGE only considers the specificity of the outputs, whereas ELO additionally considers information about the workers involved. We manually verified that there were five players who did not perform well in other tasks, yet managed to score high in "Phi-beta-sigma". This is because these workers provided specific labels such as "fraternity", "frat", or "sorority". Furthermore, three players that did well in other tasks did not perform well in that image, possibly because they were unaware of both the exact Greek characters or their connection to fraternities and sororities. Given that low-rating workers provided specific answers, ELO and EP adjust the image rating to be lower (easier).

The above example additionally demonstrates that the various methods can coexist and play a complementary role. When they are in agreement, this suggests very small uncertainty over the ranking. Disagreement, on the other hand, may be indicative of some interesting fact about the task or the workers.

5.2 Prediction Accuracy

Ratings can be used for balanced task assignment. To confirm this, we investigate whether the worker and task ratings of the three methods can accurately predict output quality. In particular, we predict that a worker (pair of workers) in BOT-PARTNER (HUMAN-PARTNER) will match on an informative label, if its rating (their average rating) is greater than the task rating. Otherwise, we predict that the worker (pair of workers) will agree on a generic label. Table 5 contains the accuracy for each method on each dataset.

Table 5: Prediction accuracy of task output

	BOT-PARTNER	HUMAN-PARTNER
AVERAGE	60.83%	64.56%
ELO	75.63%	70.08%
EP	62.29%	58.13%

Overall, ELO performs significantly better than EP in both datasets. This may be because (1) the EP update occurs after each individual encounter instead of optimizing over all encounters simultaneously, and (2) EP considers both the mean and the variance while ELO only needs the mean. Furthermore, we note that ELO achieves consistently higher accuracy than AVERAGE. It is telling that even in HUMAN-PARTNER with the 17,357 encounters, AVERAGE does not manage to outperform ELO. This demonstrates our original claim that even in large datasets AVERAGE does not capture the complex dynamics of the worker interaction and the task strain, which can hurt performance.

5.3 Effect of Worker Interaction

We next investigate the effect of worker interaction. For this purpose, we consider a modified Bayesian rating scheme, where a player's performance is only determined by itself and not the joint effort (this only affects ELO and EP). In other words, even though the team of players jointly produces the output, we assume in this experiment that each player's performance in an encounter depends only on their individual skill. In BOT-PARTNER, this was already the case since we considered teams consisting of a single worker. We now assume this to also be the case in HUMAN-PARTNER, and we investigate how the correlation coefficients for worker and task rankings between AVERAGE, ELO, and EP change. We show our

results in Table 6. We make two observations. First, the correlation of both ELO and EP compared to AVERAGE decreases substantially. This is because we assume a less accurate model that ignores worker interaction. Second, ELO and EP are still highly correlated, since they both estimate the ratings for the same (wrong) model.

Table 6: Correlation coefficient between worker rankings and task strains in HUMAN-PARTNER without interaction

	WORKER RANKINGS	TASK STRAINS
AVERAGE-ELO	44.44%	58.79%
AVERAGE-EP	30.27%	39.39%
ELO-EP	88.46%	92.73%

Finally, we study the robustness of the various schemes by looking at the effect of unbalanced task schedules. For this purpose, we consider the dataset BOT-PARTNER, but this time we assume that each encounter occurs with a fixed probability p. Trivially, the case of p=1 corresponds to the full dataset where all encounters are considered. In Figure 4, we calculate Spearman's coefficient between the ranking that each method produces when p=1 and the ranking it generates as p decreases from 0.1 to 0.9. For each value of p, we randomly sample 20 subsets of encounters, and report the average. We observe that AVERAGE has consistently the worst performance, in the sense that the ranking it produces changes more significantly as p decreases. ELO is the most robust, followed by EP. This confirms our claim that the average score is very good as long as the task schedules are balanced, whereas ELO and EP are more robust and superior for unbalanced schedules.

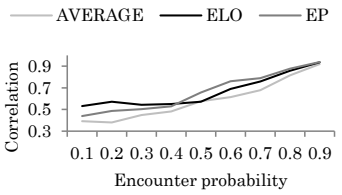

Figure 4: Spearman's correlation coefficient for worker rankings in terms of encounter probability in BOT-PARTNER.

5.4 Discussion

Our proposed experimental design based on simulated encounters has the potential to generate very large datasets from a small initial dataset that acts as a seed in cold-start situations. It is thus a cost-efficient solution, tailored for costly human studies involving large numbers of participants. Furthermore, it has the advantage that we can control how large the generated dataset will be by adjusting the encounter probability p. In principle, we can even generate a larger number of smaller datasets by doing many simulations with small values of p. A downside is that the generated dataset can contain a large number of invalid encounters. Furthermore, our scheme is not necessarily unbiased, since it is more likely to keep encounters with low-skilled workers as it takes these workers longer to match with the bot. Note that this is not a general property of our proposed scheme but related to the specifics of the ESP game where players immediately move on to the next task as soon as a match is reached.

All three methods are online and thus fast to implement in any collaborative crowd computing scenario. Relaxing the original model without considering the worker interaction leads to very inconsistent rankings and discrepancy between ELO and AVERAGE. This suggests that worker interaction is indeed an important ingredient of our model and leads to better rankings. ELO has the highest prediction accuracy of over 70% out of the three methods, followed by AVERAGE. Furthermore, ELO leads to the most consistent worker ranking even as the encounter probability decreases, followed by EP. This suggests that ELO is a very robust algorithm that generally outperforms AVERAGE. Its strengths lie in the facts that (1) it takes into account the worker dynamics and the task strains, and (2) it has a simple update rule that only relies on the mean. Compared to ELO, EP converges much faster [12]. It also leads to consistent rankings, and in some cases beats AVERAGE. EP tries to optimize over both the first two moments, i.e., the mean and the variance information. ELO, on the other hand, is based on the same model but uses a simpler update rule with only the mean, making it generally more robust when we do not simultaneously optimize over multiple encounters. The fact that our Bayesian graph model is approximate and does not capture the true dynamics may also contribute to the robustness of the simpler ELO scheme.

In general, all three methods are valuable and can convey important information about the worker ratings and task strains. Of particular interest are cases of disagreement, where the models produce very different ratings. This may indicate that low-skilled workers have produced high-quality output for a task that workers of higher rating have failed, or that a task has been surprisingly hard for high-skilled workers. The service requester can utilize this information to generate better assignments and improve the output quality. Even though AVERAGE is generally a resilient method, especially as the number of encounters grows, our empirical evaluation demonstrates that a collaborative ELO rating system can outperform it in terms of consistency, accuracy, and robustness. This implies that ELO could serve as the primary rating system.

6 CONCLUSION

Collaborative human computation and crowdsourcing are characterized by tasks of varying difficulties and involve interactions among human workers. In this work, we show how to leverage the standard rating schemes in competitive games for rating participants and tasks in collaborative crowd computing. Experimental results demonstrate that collaborative ranking using our TrueSkill-inspired rating framework can lead to better results than the traditional additive scoring scheme. In the future, we would like to explore how the specific type of interaction (e.g., weak-link, strong-link, linear) affects the rating process in our model. We will also consider the uncertainty over the output that the workers jointly produce. This is especially true for human computation games, where the game service provider relies on the player agreement to accept the output, but has no other way to know for sure whether it is correct or specific, etc. It would thus be valuable to extend our frameworks so that they can handle cases where there is additional uncertainty over the worker output.

REFERENCES

[1] Lora Aroyo and Chris Welty. 2014. The Three Sides of CrowdTruth. *Journal of Human Computation* 1 (2014), 31–34. Issue 1.

[2] Ralph Allan Bradley and Milton E. Terry. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

[3] Miguel Brozos-VÃązquez, Marco Antonio Campo-Cabana, JosÃĺ Carlos DÃŋaz-Ramos, and Julio GonzÃąlez-DÃŋaz. 2008. Ranking participants in tournaments by means of rating functions. *Journal of Mathematical Economics* 44, 11 (2008), 1246 – 1256.

[4] Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise Ranking Aggregation in a Crowdsourced Setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM '13)*. 193–202.

[5] Pierre Dangauthier, Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill Through Time: Revisiting the History of Chess. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS'07)*. 337–344.

[6] Herbert A. David. 1988. *The method of paired comparisons* (2nd. ed.). Chapman and Hall, London.

[7] A. P. Dawid and A. M. Skene. 1979. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 20–28.

[8] Julie S. Downs, Mandy B. Holbrook, Steve Sheng, and Lorrie Faith Cranor. 2010. Are Your Participants Gaming the System?: Screening Mechanical Turk Workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. 2399–2402.

[9] Arpad E. Elo. 1978. *The rating of chess players, Past and Present.* Arco Publishing, New York.

[10] Mark E. Glickman. 2001. Dynamic paired comparison models with stochastic variances. *Journal of Applied Statistics* 28, 6 (2001), 673–689.

[11] Bruce Hajek, Sewoong Oh, and Jiaming Xu. 2014. Minimax-optimal Inference from Partial Rankings. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS'14)*. 1475–1483.

[12] Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill™: A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems 19*. 569–576.

[13] Jack Hirshleifer. 1983. From Weakest-Link to Best-Shot: The Voluntary Provision of Public Goods. *Public Choice* 41, 3 (1983), 371–386.

[14] Jeff Howe. 2006. The rise of crowdsourcing. *Wired magazine* 14, 6 (2006), 1–4.

[15] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality Management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*. 64–67.

[16] Pierre-Emmanuel Jabin and StÃĺphane Junca. 2015. A Continuous Model For Ratings. *SIAM J. Appl. Math.* 75, 2 (2015), 420–442.

[17] Nassim Jafarinaimi. 2012. Exploring the Character of Participation in Social Media: The Case of Google Image Labeler. In *Proceedings of the 2012 iConference (iConference '12)*. 72–79.

[18] Shaili Jain and David C. Parkes. 2008. A Game-Theoretic Analysis of Games with a Purpose. In *Proceedings of the 4th International Workshop on Internet and Network Economics (WINE '08)*. 342–350.

[19] N. L. Johnson, S. Kotz, and N. Balakrishnan. 1995. *Continuous Univariate Distributions, Vol. 1-2* (2nd. ed.). John Wiley and Sons, New York.

[20] David R. Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative Learning for Reliable Crowdsourcing Systems. In *Advances in Neural Information Processing Systems 24*. 1953–1961.

[21] David R. Karger, Sewoong Oh, and Devavrat Shah. 2014. Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. *Oper. Res.* 62, 1 (2014), 1–24.

[22] Maurice J. Kendall. 1970. *Rank correlation methods* (4th. ed.). Hafner Press, New York.

[23] Aniket Kittur. 2010. Crowdsourcing, Collaboration and Creativity. *XRDS* 17, 2 (2010), 22–26.

[24] Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing User Studies with Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. 453–456.

[25] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: Crowdsourcing Complex Work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. 43–52.

[26] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. 2006. Factor Graphs and the Sum-product Algorithm. *IEEE Trans. Inf. Theor.* 47, 2 (2006), 498–519.

[27] Edith Law and Luis von Ahn. 2009. Input-agreement: A New Mechanism for Collecting Data Using Human Computation Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. 1197–1206.

[28] Jing Liu, Quan Wang, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. Question Difficulty Estimation in Community Question Answering Services. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 85–90.

[29] Panagiotis Mavridis, David Gross-Amblard, and Zoltán Miklós. 2016. Using Hierarchical Skills for Optimized Task Assignment in Knowledge-Intensive Crowdsourcing. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. 843–853.

[30] Thomas P. Minka. 2001. Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI '01)*. 362–369.

[31] Thomas P. Minka. 2001. *A Family of Algorithms for Approximate Bayesian Inference.* Ph.D. Dissertation.

[32] Robert Munro. 2010. Crowdsourced translation for emergency response in Haiti: the global collaboration of local knowledge.

[33] Sahand Negahban, Sewoong Oh, and Devavrat Shah. 2012. Iterative ranking from pair-wise comparisons. In *Advances in Neural Information Processing Systems 25*. 2474–2482.

[34] Habibur Rahman, Saravanan Thirumuruganathan, Senjuti Basu Roy, Sihem Amer-Yahia, and Gautam Das. 2015. Worker Skill Estimation in Team-based Tasks. *Proc. VLDB Endow.* 8, 11 (2015), 1142–1153.

[35] Ariel Rubinstein. 1980. Ranking the Participants in a Tournament. *SIAM J. Appl. Math.* 38, 1 (1980), 108–111.

[36] Nihar B. Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin J. Wainwright. 2016. Estimation from Pairwise Comparisons: Sharp Minimax Bounds with Topology Dependence. *J. Mach. Learn. Res.* 17, 1 (2016), 2049–2095.

[37] L. L. Thurstone. 1927. A law of comparative judgment. *Psychological Review* 34, 4 (1927), 273–286.

[38] Luis Von Ahn. 2005. *Human Computation.* Ph.D. Dissertation.

[39] Luis von Ahn and Laura Dabbish. 2004. Labeling Images with a Computer Game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. 319–326.

[40] Luis von Ahn and Laura Dabbish. 2008. Designing Games with a Purpose. *Commun. ACM* 51, 8 (2008), 58–67.

[41] Luis von Ahn, Mihir Kedia, and Manuel Blum. 2006. Verbosity: A Game for Collecting Common-sense Facts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. 75–78.

[42] Luis von Ahn, Ruoran Liu, and Manuel Blum. 2006. Peekaboom: A Game for Locating Objects in Images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. 55–64.

[43] Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Advances in Neural Information Processing Systems 22*. 2035–2043.

[44] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. 2016. Spectral Methods Meet EM: A Provably Optimal Algorithm for Crowdsourcing. *J. Mach. Learn. Res.* 17, 1 (2016), 3537–3580.

[45] Dengyong Zhou, John C. Platt, Sumit Basu, and Yi Mao. 2012. Learning from the Wisdom of Crowds by Minimax Entropy. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*. 2195–2203.