MAKING TASK RECOMMENDATIONS IN CROWDSOURCING CONTESTS

Jiahui Mo, Nanyang Business School, Nanyang Technological University, Singapore, Singapore, jhmo@ntu.edu.sg

Sumit Sarkar, Jindal School of Management, The University of Texas at Dallas, Richardson, TX, USA, sumit@utdallas.edu

Syam Menon, Jindal School of Management, The University of Texas at Dallas, Richardson, TX, USA, syam@utdallas.edu

Abstract

Crowdsourcing contests have emerged as an innovative way for firms to solve business problems by acquiring ideas from participants external to the firm. To facilitate such contests a number of crowdsourcing platforms have emerged in recent years. A crowdsourcing platform provides a two-sided marketplace with one set of members (*seekers*) posting tasks, and another set of members (*solvers*) working on these tasks and submitting solutions. As crowdsourcing platforms attract more seekers and solvers, the number of tasks that are open at any time can become quite large. Consequently, solvers search only a limited number of tasks before deciding which one(s) to participate in, often examining only those tasks that appear on the first couple of pages of the task listings. This kind of search behavior has potentially detrimental implications for all parties involved: (i) solvers typically end up participating in tasks they are less likely to win relative some other tasks, (ii) seekers receive solutions of poorer quality compared to a situation where solvers are able to find tasks that they are more likely to win, and (iii) when seekers are not satisfied with the outcome, they may decide to leave the platform; therefore, the platform could lose revenues in the short term and market share in the long term. To counteract these concerns, platforms can provide recommendations to solvers in order to reduce their search costs for identifying the most preferable tasks. This research proposes a methodology to develop a system that can recommend tasks to solvers who wish to participate in crowdsourcing contests. A unique aspect of this environment is that it involves competition among solvers. The proposed approach explicitly models the competition that a solver would face in each open task. The approach makes recommendations based on the probability of the solver winning an open task. A multinomial logit model has been developed to estimate these winning probabilities. We have validated our approach using data from a real crowdsourcing platform.

Keywords: Crowdsourcing, Competition, Recommendation Systems.

1 INTRODUCTION

Crowdsourcing contests have emerged as an innovative way for firms to acquire ideas to solve business problems from participants who are usually external to the firm. In a typical contest, a firm posts its requirements about a task on a website. The firm also specifies the time frame for the contest and the prize money that the winner of the task would receive. The contest is usually open to everyone who wishes to participate. After the deadline is reached, the firm evaluates the solutions and picks a winner. Typically, there is a single winner who receives the entire prize money; the other participants do not get any monetary reward for their efforts.

A number of crowdsourcing platforms have emerged in recent years to facilitate such contests. A crowdsourcing contest platform is an intermediary who provides a two-sided marketplace with one set of members (usually firms) posting tasks, and another set of members working on these tasks by submitting solutions. The members of the first set are the *seekers*, while the members of the other are the *solvers*. The emergence of these intermediaries makes it possible to connect the demands of a variety of seekers with a large community of potential solvers. Crowdsourcing platforms provide a good venue for seekers to access a wider array of solvers than those in one internal organization, and to undertake problems for which internal resources may be scarce. At the same time, by providing access to many tasks in a centralized location, these platforms make it easier for solvers to generate income by using their expertise and skills (Lacy 2011). Platforms often support tasks across a variety of categories such as developing software and web sites, and designing logos, business cards, jewelry, t-shirts, etc.

Such contests have resulted in many successful outcomes for both parties, and as a result crowdsourcing platforms have becomes quite popular. The number of tasks in crowdsourcing platforms has also increased rapidly. For example, a popular crowdsourcing platform *99designs*, which was founded in 2008, handled about 100,000 contests by 2012 (Lacy 2012), and has almost tripled that number as of April 2014 (99designs.com). Gartner Inc. (2013) has predicted that by 2017, more than half of all consumer goods manufacturers will receive 75% of their consumer innovation and R&D capabilities from crowdsourcing solutions.

As crowdsourcing platforms attract more seekers and solvers, the number of tasks that are open at any time can become quite large (a task is open for solvers to participate in from the time it is posted by a seeker until the seeker-specified deadline after which new solvers are not able to submit solutions). For instance, on *99designs*, there are around a thousand open tasks in some categories at any point in time, and a typical task stays open for four to seven days. Therefore, when a solver visits the platform with the intention of participating in a task, she would have to evaluate a very large number of task postings to identify suitable tasks to participate in. Because a solver has to complete a task in a reasonably short amount of time (from a few hours to a few days at most, given the duration a typical task is open), it is usually not worthwhile for the solver to spend a substantial amount of time and effort in trying to identify tasks to participate in. Platforms list open tasks in order of some pre-determined dimension, along with some alternative orderings that a solver can deploy. A common dimension is the posting time for a task, with the most recently posted task appearing on top. Other typical dimensions include the amount of prize money and the time left before a task will close. While such options provide a solver some flexibility in examining the descriptions of open tasks, it is nevertheless very difficult, if not impossible, for a solver to identify and examine in detail all the tasks she could be potentially interested in. Consequently, solvers search only a limited number of tasks before deciding which one(s) to participate in, often examining only those tasks that appear on the first couple of pages of the task listings (Chilton et al. 2010).

This type of search behavior has potentially serious implications for all parties involved in crowdsourcing contests. First, solvers typically end up participating in tasks for which they are not as well suited as other harder to find tasks (Ambati et al. 2011). This reduces the chance of the solver winning such a contest. Further, it can also lead to seekers receiving solutions of poorer quality

compared to a situation where solvers are able to find the tasks that they are more likely to win. A seeker may not even receive the minimum number of submissions that makes the exercise worthwhile – platforms usually have to refund the prize money to a seeker in such circumstances (e.g., *Zhubajie* and *DesignCrowd*). If seekers are not satisfied with the outcome, they may leave the platform; as a result, the platform could lose revenues not only for current tasks but could also lose future business opportunities.

Recognizing these concerns, some platforms have started providing recommendations to solvers in order to reduce their search costs. The recommendations are usually provided whenever a solver visits the site. These approaches (e.g., in *Zhubajie* and *Freelancer*) are typically based on recommendation systems deployed for e-commerce sites. In some cases, the tasks recommended to a solver are those that have requirements that best match the skills of the solver (which are usually explicitly provided by the solvers when they first register at the site). In other cases, tasks recommended to a solver are those that are most similar (based on some platform determined metric) to closed tasks that the solver had participated in. Emergent research on recommendation systems for crowdsourcing environments address micro-task platforms (and not contest platforms). They also propose content-based or collaborative approaches. For example, Ambati et al. (2011) develop a system based on task descriptions and solvers' skill sets. Yuen et al. (2012) use a Probabilistic Matrix Factorization (PMF) approach to develop a preference-based task recommendation framework.

However, an important and interesting distinction between crowdsourcing micro-task platforms and crowdsourcing contest platforms is that the former are non-competitive environments while the latter involve competitions among many solvers and there is only one winner in each contest. In crowdsourcing contests, solvers are usually most interested in participating in tasks that maximize their chances of winning (Lakhani et al. 2007, Brabham 2010). A solver's chances of winning are greatly influenced by the other solvers who are participating in the same task. Traditional recommendation systems (whether for e-commerce or micro-task platforms) do not capture the competitive environments of crowdsourcing contests.

In this research, we develop a system to recommend tasks that a solver is most likely to win given the current set of participants in the various open tasks in a crowdsourcing platform. The central component of such a recommender system is a methodology to determine the probability of a focal solver winning a task if she participates in it. This probability is determined by considering the solver's characteristics known to the platform (self-reported skill sets, task history, past performance, etc.) as well as the known characteristics of the competitors participating in the task. A multinomial logit model is developed to determine the winning probabilities for each solver participating in a task. For a focal solver, once the winning probabilities are determined for each task, a desired number of tasks (e.g., top-*k* tasks) can be recommended by comparing these probabilities. We validate the probability model using data from a real crowdsourcing contest platform *99designs*.

When recommending open tasks to a solver, a platform will not know the full competition when the tasks close; instead the platform will have to make the recommendations based on whatever is the competition for each open task at the time the recommendation has to be made. An interesting question is whether the existing competitions for different open tasks that are at the same stage in their overall timelines are representative of the eventual competition that the target solver will face at the time the task closes. We conduct experiments to examine this issue by ranking pairs of such tasks for different target solvers. Our experiments show that the rankings are remarkably consistent over time. Thus, even though the probability that the target solver will win a task changes over time as more participants join in, the likelihood of winning one task relative to another one is not very different when the task closes.

We describe next the problem in detail, and develop a model to predict the probability that a solver would win an open task with a given set of competitors. We then describe how we validate our

approach using data from the *99designs*. We conclude by discussing the potential contributions of our work.[1]

2 RECOMMENDATION SYSTEMS FOR A CROWDSOURCING CONTEST PLATFORM

2.1 Problem Description

The context we model here is one where a platform wishes to provide recommendations to a solver when the solver visits the platform's website (referred to as the *site*). We assume that the solver has already registered on the site and provided information as required by the platform. This information typically includes, along with personal identifying information, the skill sets of the solver. For example, a solver interested in designing logos may list formal training in graphic design (additional details of such information for a specific platform are described later). We refer to such a solver as the target solver. When the target solver logs on to the site, the platform could provide, along with the default ordering of open tasks, the tasks recommended for the target solver based on the predicted probability of her winning those tasks. In order to estimate the target solver's probability of winning a given open task, the platform needs to consider not only the target solver's skills and past participation history, but also the skill sets and participation histories of other competitors who are participating in the contest. The current participants define the *competition structure* for the task. We note that for ease of exposition we discuss how to order recommendations based on the target solver's *probability* of winning a task. Our model naturally and easily extends to where tasks are recommended based on their expected payoffs (product of prize money and winning probability) to the solver.

2.2 Determining a Target Solver's Probability of Winning an Open Task

The main component of the proposed recommendation system is a module to estimate the target solver's probability of winning an open task. As discussed earlier, we expect this probability to depend primarily on the characteristics of the target solver and of the other solvers who comprise the competition structure. Therefore, determining the winning probability for the target solver can be viewed as a prediction problem, given the characteristics (attribute values) of all the participants.

Several models have been proposed in the literature to predict probabilities of outcomes based on attributes relevant to the problem domains of interest (Boulier & Stekler 2003, Clarke & Dyte 2000). For example, for binary classification types of task, some widely used models include, among others, logit, probit, decision trees, naïve Bayes, and support vector machines. In our problem, there are many solvers for a given task, and only one person is eventually chosen as a winner. To predict the probability of an event happening within multiple classes (or solvers, in our context), we also consider a multinomial logit model. This model has been widely used in multi-class prediction problems, such as cancer prediction (Zhou et al. 2006) and winner prediction in horse racing competitions (Bolton and Chapman 1986).

We determine the probability of the target solver winning an open task i as follows. We denote the number of solvers (including the target solver) in the task by M_i, and index the solvers using j (j = 1,…, M_i). X_{ij} denotes the attribute values of solver j participating in task i. We denote the *performance* of solver j in task i by V_{ij}. We then model the performance of solver j in task i to be a function of the attribute values X_{ij} as follows (Terwiesch and Xu 2008, Yang et al. 2011):

[1] While there exists a growing body of work on crowdsourcing contests, we did not include a complete literature review section for space considerations. Extant literature deals with issues such as mechanism design for tournaments and contests (Dahan and Mendelson 2001, Moldovanu and Sela 2001, Terwiesch and Xu 2008), motivations and incentives for solvers' participation and effort levels (Brabham 2010, Boudreau et al. 2011), and solvers task selection patterns (Yang et al. 2008). The findings of these researches are tangential to our work.

$$V_{ij} = \alpha X_{ij} + \varepsilon_{ij} . \qquad (1)$$

Here, X_{ij} is a vector of weights that correspond to the importance of each attribute in a solver's performance, and ε_{ij} corresponds to an error term which captures the impact of unobserved factors (e.g., the solver's effort level, the solver's ability to anticipate the seeker's taste, etc.) that impact the performance of a solver. The intuition behind this model is analogous to that of choice models, with the eventual performance depending on a collection of deterministic and random components.

In order for solver *j* to win task *i*, her performance should be the best among all the participants. Therefore, the winning probability for solver *j* in task *i*, P_{ij}, is given by

$$P_{ij} = P(V_{ij} > V_{ik}, k \neq j, k = 1, \dots, M_i) . \qquad (2)$$

To obtain the probability P_{ij}, we need the distribution function for the error term in equation 1. We assume that the error term ε_{ij} follows a Type 1 extreme value distribution (Greene 2011) for tractability. This results in the following closed-form expression for P_{ij}:

$$P_{ij} = \frac{exp(\alpha X_{ij})}{\sum_{k=1}^{M_i} exp(\alpha X_{ik})} . \qquad (3)$$

The above model can be operationalized as follows. First, we need to identify the set of attributes that are observable by the platform and that can be expected to impact a solver's performance for a given task. Then, we need to estimate the values of the parameters α that best reflect the impact the attributes have on solver performances. Analogous to traditional multinomial models, maximum likelihood techniques can be used to estimate these parameters given historical data on task participants and task outcomes. Once these parameters are obtained, the target solver's probability of winning an open task can be determined using Equation 3. When an open task finishes, it will be included in the historical data set, and the parameter α will be reestimated. Therefore, all the solvers' last time's submissioins will be considered for subsequent task performences.

The proposed approach is computationally quite efficient. The probabilities can be easily computed if the parameters have been obtained beforehand (which would normally be the case). If the maximum number of open tasks at any point in time is *n*, and the maximum number of solvers for an open task is *M*, the worst case computational complexity of the recommendation system will be *O*(*nM*). Given the values likely to be observed for *n* (in the thousands) and *M* (in the hundreds), a platform can determine in real-time the tasks to recommend to the target solver when she logs on to the site, while accounting for all the competitors who are participating in each of the open tasks.

3 MODEL VALIDATION

3.1 Data Collection and Description

We have validated the model using data collected from *99designs*, which claims to be the largest online marketplace for graphic design in the world. Given the large number of open tasks, solvers will find it difficult even to browse through all of them, let alone evaluate each task effectively for participation. *99designs* has eight categories of tasks – (i) logo & identity, (ii) website & app, (iii) business & advertising, (iv) clothing & merchandise, (v) art & illustration, (vi) packing & label, (vii) book & magazine, and (viii) other. We have chosen to work with the logo & identity category, as it is the most popular category in the platform, with approximately 60% of the open tasks coming from this category (99designs.com). Most tasks in this category are posted by firms looking to design logos for official use – on business cards, business stationary, or webpages. The logo & identity category is further divided into five subcategories: logo design, logo & business card, business card, stationery and business identity pack.

We have collected data using a crawler written in Perl. The data collection started from the task listing page, which provides information on all the open tasks and some of the recently closed

(completed) tasks. The crawler collected data on all the completed tasks that were listed on the site on a specific day. It then went to the task landing page for each task in the listing page and collected all the information available there. This included, for each task, information such as task title, task number, and task prize amount. This also included task descriptions, firm (seeker) industry, logo style, color requirements, as well as target customers of the firm. The crawler then visited the corresponding solution page to collect information on all the solutions. This included the list of solvers who have submitted entries, the sequence in which entries were submitted, and the winner as determined by the seeker. Finally, the crawler collected all the available information for each participating solver for the selected tasks. This included both solver reported information as well as platform generated information on the solver.

Our final data set consists of 1917 tasks and about 13,000 solvers. On average, there are 37 solvers for each task, while the average prize money is about $367. We have divided the dataset into two equal parts for analysis, with the first half of the tasks being used for training and the second half reserved for testing.

3.2 Attributes Influencing Winning Probability

In order to build the model, we need to identify factors which can potentially influence a solver's performance and her winning probability. Performance has been investigated in many research areas, such as economics, organizational behavior, and psychology. First, past performance has been observed to be a good predictor of future performance (McGonigle and Curnow 2006). Further, we expect that a solver with expertise specific to the domain of interest would have a better chance of winning a task than otherwise (Ericsson 2006). Prior research also suggests that an individual's performance is limited by the extent of her expertise (Terwiesch and Xu 2008). We identify as many variables along these dimensions as we can in the context of crowdsourcing platforms based on what we observed from the *99designs* website.

The platform has developed two capability measures (based on a solver's capability demonstrated in prior tasks) that it lists on the solver landing pages – *points* and *capability level*. The crawler has collected this data, and they are included as potentially important attributes for inclusion in our model. In addition, three other solver-specific attributes are also provided by the site – the number of times the solver has won across all categories, the number of times the solver was shortlisted across all categories, and the number of categories in which the solver considers herself an expert (the site provides the number of sub-categories, from which we identify the number of categories). These solver-specific variables are shown in Table 1.

X1 : points	X4 : # of times shortlisted (across all categories)
X2 : capability level	X5: # of categories with reported expertise
X3 : # of wins (across all categories)	

Table 1. Site Provided Solver Specific Attributes

Because the platform does not disclose how these measures are determined, we have decided to create a set of additional measures (attributes) based on the task participation data that we have collected for each solver. These attributes complement those already provided by the site. The attributes are specific to a solver at the time a specific task is available for participation. For example, a solver who has participated in tasks T25, T200 and T603 in our dataset while winning task T200 will have the values of number of prior wins (variable X7) set to 0, 0, and 1, respectively for each of these tasks. This is because the solver has not won any task when she decides to participate in tasks T25 and

T200, while she has won one task when she opts to participate in task T603.[2] Table 2 lists all the additional attributes we derived from our training data. Note that the values of all these attributes are calculated as of the time the solver participates in the current task (within the training data). [3]

X6: success rate	X16: average number of solutions per participated task
X7: number of solver's prior wins in the industry of the current task	X17: fraction of solver's prior participations in the subcategory of the current task
X8: fraction of solver's prior wins in the industry of the current task	X18: number of solver's prior participations in the industry of the current task
X9: number of solver's prior wins in the subcategory of the current task	X19: fraction of solver's prior participations in the industry of the current task
X10: number of distinct industries with wins	X20: number of distinct industries participated in
X11: fraction of solver's prior wins in the subcategory of the current task	X21: number of solver's prior participations in the subcategory of the current task
X12: number of distinct subcategories with wins	X22: average listed prize per participated task
X13: number of tasks already participated in	X23: number of distinct subcategories participated in
X14: aggregate number of solutions submitted by solver	X24: 1 if solver reports expertise in current task category; 0 if not
X15: average number of competing solvers per participated task	X25: 1 if solver reports expertise in current task subcategory; 0 if not

Table 2. Additional Attributes Considered at the Solver-Task Levels

3.3 Winning Prediction Evaluation

As discussed in Section 2.2, many data mining models can be used to predict the winning probability. We first adopt Naïve Bayes and Bayesian Network to make classifications, because these two methods can provide probability estimates when making classifications. We also apply a Multinomial Logit Model (MNL) which can capture competition explicitly. The models are trained on the training data set, and the associated parameters estimated. We use these models to predict the winning probability for each solver in each test task. The solvers in each test task are then ranked according to the predicted winning probability, and the solver with the highest predicted winning probability is projected as the winner of the task. A prediction is considered correct if the projected winner of a task is the true winner, and the proportion of the number of correct predictions among all the tasks in the test set is the accuracy of the model involved.

So far, we have focused on predicting a single winner for each task. Predicting the specific winner correctly with well over thirty solvers on average is difficult in general. For instance, there may be multiple strong solvers who are competing in the same task. For our model to make a correct prediction for such a task, it has to precisely predict the performance of one of these strong solvers to be superior to all the others. From this point of view, measuring accuracy based on predicting only one

[2] We note that while task specific attributes such as prize amount are common across all participants and will not impact solvers' winning probabilities given the competition structure, other attributes such as task industries and subcategories will lead to differential attribute-values across the participants and can influence solvers' winning probabilities differently (e.g., X7).

[3] We note here that even for new solvers, some variables, such as X5, X24, and X25, are still applicable, and these variables have values. When variables are not applicable to new solvers, we fill them with 0.

winner may be a very stringent requirement. Therefore, we consider another way to measure the qualities of the models. This approach is more nuanced than measuring accuracy based on the prediction of a single winner. For example, a model is considered wrong any time the predicted winner is not the actual winner. However, if the model predicts the actual winner to have the second highest winning probability for a task, while for another task, it predicts the actual winner to have the lowest winning probability, the prediction quality is quite different for these two tasks. This aspect of performance is not captured if we consider accuracy based on predicting only a single winner.

Therefore, in addition to predicting a single winner, we rank solvers based on winning probability and predict the top *n* solvers. When one of the top *n* predicted solvers is the actual winner, we consider the prediction to be a success. We have considered in our experiments values of two to five for *n*. The results for the three models with various values for *n* are presented in Table 1.

	n=1	*n*=2	*n*=3	*n*=4	*n*=5
Naive Bayes	90	160	231	283	351
	9.39%	16.70%	24.11%	29.54%	36.64%
Bayesian Network	106	176	256	313	360
	11.06%	18.37%	26.72%	32.67%	37.58%
MNL	123***,*	214***,***	292***,***	372***,***	427***,***
	12.84%	22.34%	30.48%	38.83%	44.57%

*, **, ***refer to improvements at the 0.1, 0.05, and the 0.01 significance levels, respectively.
The first significance level is for the performance of MNL relative to Naive Bayes, while the second is relative to Bayesian Network.

Table 3. Accuracies of the three models

When predicting only one winner, the accuracy of MNL is significantly better than that of both Naive Bayes (at a 0.01 significance level) and Bayesian Network (at a 0.1 significance level). When we relax *n* from one to three, the accuracy of MNL increases from 12.84% to 30.48%, while it increases further to 44.57% when *n* is five. That is, the winner will be predicted to be within top 5 almost half the time. While all these models can be used to predict the winning probability, MNL predicts winners more accurately as it considers competition structure – something the other two models do not do. These experiments suggest that MNL should be preferred to predict a solver's winning probability, at least in our context.

We also compared the prediction accuarcy of MNL with two other methods. The first approach randomly selects a solver as the winner, while the second predicts the solver with the highest capability points as the winner. The first provides a measure of how useful a good recommender system can be, while the second uses the variable the platform considers to be a good predictor of solver capability. Results are presented in Table 4.

	n=1	*n*=2	*n*=3	*n*=4	*n*=5
Random Selection	35	83	118	161	196
	3.65%	8.66%	12.32%	16.81%	20.46%
Points	68***	144***	216***	292***	342***
	7.10%	15.03%	22.55%	30.48%	35.70%
MNL	123***,***	214***,***	292***,***	372***,***	427***,***
	12.84%	22.34%	30.48%	38.83%	44.57%

*, **, ***refer to improvements at the 0.1, 0.05, and the 0.01 significance levels, respectively.

The first significance level is for the performance of MNL relative to Random Selection, while the second is relative to the model that uses Points.

Table 4. Accuracy Comparisons of MNL with Random Selection and Point-based Prediction

In the random selection method, a randomly chosen solver is projected as the winner. In the model based on Points, the solvers in a task are ranked by points, and the solver with the highest points projected as the winner. The platform has developed the Points measure to capture solvers' capabilities. Therefore, this should predict the winner effectively if the platform has constructed this measure well. Table 4 shows that Points measure does improve on random selection significantly for all five values of *n*. However, MNL improves on the Points based model significantly as well, with the improvements in the accuracy of predictions for all the five values of *n* being significant at the 0.01 level. When *n*=1, the accuracy is increased by 5.74%. When *n* is relaxed to 4 or 5, the accuracy increases by over 8%.

In sum, we find that MNL performs sgnificantly better as it captures competition structures explicitly. We should point out that we have collected data that capture roughly two weeks of activity on the site, and using just this limited amount of data, we have been able to demonstrate the value of using additional attributes for prediction. A crowdsourcing platform would naturally be able to do a better job of fine-tuning the predictive model given the data available to it.

3.4 Ranking Tasks with Incomplete Information

Our previous analysis demonstrates that a crowdsourcing site can make reasonable predictions of the probability of a solver winning a contest based on the competition structure. In order to make recommendations to a target solver, the platform would need to estimate her winning probability for all the open tasks, and then recommend a pre-determined number of tasks based on these probabilities (e.g., rank the tasks based either on expected winning probability or expected payoffs). Naturally, tasks that have been posted recently have fewer participants than tasks that are about to close, and therefore at a given point in time the winning probabilities for newly posted tasks would usually be higher than that for the other tasks. To make the comparisons more meaningful, our system could provide recommendations to a solver segregated by different timelines; for instance, the system can recommend tasks for each closing date separately, with the recommendations for each closing date sorted by the predicted winning probabilities.

When recommending open tasks to a solver, the platform will not know the final competition structures of the tasks; instead the platform will have to make the recommendations based on whatever is the competition structure for each open task at the time the recommendation has to be made. An important consideration then is the reliability of the task rankings for a solver when tasks have received only a fraction of the eventual set of solutions. An assumption one could make is that the existing competition structures for different open tasks that are at the same stage in their overall timelines are representative of the eventual competition that the target solver will face at the time the task is closed. In that case, even though the probability that the target solver will win a task changes over time as more participants join in, the likelihood of winning one task relative to another one may not be very different when the task closes, i.e., the rank order of the winning probabilities for different open tasks at a similar stage of completion may not change very much over time.

To examine if this assumption is reasonable, we conduct the following experiment. We consider tasks which are at the same level of completion. For example, we consider the competition structures for tasks when they have received only the first 25% of all their eventual solutions. We then compare how two tasks would be ranked for a target solver if only their current sets of competitors were considered, relative to the ranking that would be obtained if their final competition structures were known. If the ranks are consistent across the two approaches, then it would indicate that it is reasonable to make recommendations using tasks with partial (incomplete) competition structures. We repeat this for all feasible task pairs for each solver for three partial competition structures: specifically, when the first 25%, 50%, and 75% of the whole competition structures are known. We

find that based on the first 25% of the competition, 89.32% of the task pairs are ranked consistently. The consistencies in rankings increase to 93.56% and 96.47% when 50% and 75% of the competition are known, respectively. This suggests that the proposed approach is able to identify tasks to recommend to a solver reasonably well even with incomplete information regarding the final competition structure.

4 CONTRIBUTIONS OF OUR WORK

This research contributes to two domains of considerable interest to information systems researchers currently, crowdsourcing contests and recommendation systems. The research proposes a methodology to design a system that can recommend tasks to solvers who wish to participate in crowdsourcing contests. A unique aspect of this environment is that it involves competition among solvers. The proposed approach explicitly models the competition that solvers would face in each open task. The approach makes recommendations based on the probability of the solver winning an open task. A multinomial logit model has been developed to estimate these winning probabilities. We have validated our approach using data from a real crowdsourcing platform. We first identified a potential set of attributes that could be used to predict a solver's probability of winning a task. We then used part of the contests for which data were collected to train three alternative models. The trained models were tested on the remaining tasks and found to perform quite well (especially given the difficulty of picking a winner among many participants). The relative ranking of tasks for a target solver was shown to be quite reliable even when only 25% of the eventual solvers had submitted their solutions to open tasks.

Our proposed approach has important implications for crowdsourcing contest platforms. First, by helping solvers find winnable tasks, it would enable them to devote more effort on solving the tasks. It will also help seekers by recognizing if a posted task has not received many solutions (which would make it more winnable than other tasks), recommending that task to other solvers. This will reduce the chance that a task does not receive the minimum acceptable number of solutions. This will directly benefit the platform as well in terms of potential revenues from such tasks – for instance, we have observed in *99designs* that seekers have not picked a winner in more than 10% of the posted tasks. In addition, the benefits to the solvers and the seekers would have positive long-term benefits to the health of the platform.

The proposed approach has been developed to make recommendations whenever an existing solver visits the platform (i.e., in a push mode). However, the probability model we have developed is very flexible and can also be used in a pull mode by a solver – for instance, a solver can shortlist a set of tasks based on her own private preferences, and then ask the platform to rank these tasks based on her winning probability. Such a feature can be very valuable to a solver; even if a solver were able to identify tasks that match her skill set, she would still find it very difficult to evaluate by herself her chances of winning each of those tasks given the potentially large number of different solvers participating in each of those tasks. The proposed system can be used in conjunction with traditional recommendation approaches (e.g., skill-based) as well; the different approaches can be viewed as complementary to each other. Furthermore, the proposed system can also easily incorporate many other factors when making recommendations. We can take time pressure to finish the task into considerationg. For example, for tasks to be finished in one day, the system displays tasks by winnable probability.

A potential area of future research is to examine how often such recommendations are adopted by solvers, and the changed competitive structures that ensue. It would be interesting to see if the changed competition structures affect the underlying models – although our methodology will still apply, a platform may need to revise the model by re-training it with data collected from the new competition environment.

References

Ambati, V., Vogel, S. and Carbonell, J.(2011). Towards task recommendation in micro-task markets. In Proceedings of the 25th AAAI Workshop in Human Computation, San Francisco, CA.

Boulier, B.L. and Stekler, H.O. (2003). Predicting the outcomes of national football league games. International Journal of Forecasting, 19(2), 257–270.

Bolton, R.N. and Chapman, R.G. (1986). Searching for positive returns at the track: A multinomial logit model for handicapping horse races. Management Science, 32(8), 1040-1060.

Boudreau, K. J., Lacetera, N. and Lakhani, K. R. (2011). Incentives and problem uncertainty in innovation contests: An empirical analysis. Management Science, 57(5), 843-863.

Brabham, D.C. (2010). Moving the crowd at threadless: Motivation for participation in a crowdsourcing application. Information, Communication & Society, 13(8), 1122-1145.

Chilton, L. B., Horton, J. J., Miller, R. C. and Azenkot, S. (2010). Task search in a human computation market. In Proceedings of the ACM SIGKDD Workshop on Human Computation, p.1-9, New York, NY,

Clarke, S.R. and Dyte, D. (2000). Using official ratings to simulate major tennis tournaments. International Transactions in Operational Research, 7(6), 585-594.

Dahan, E. and Mendelson, H. (2001). An extreme-value model of concept testing. Management Science, 47(1), 252-276.

Ericsson, K.A. (2006). The influence of experience and deliberate practice on the development of superior expert performance. The Cambridge Handbook of Expertise and Expert Performance. Cambridge University Press, Cambridge, United Kingdom.

Gartner. (2013). Gartner reveals top predictions for it organizations and users for 2014 and beyond. http://www.gartner.com/newsroom/id/2603215

Greene, W.N. (2011). Econometric Analysis (Seventh Edition), Prentice Hall.

Lacy, S. (2011). Accel invests $35m. in 99designs…after years of trying. Techcrunch. http://techcrunch.com/2011/04/28/accel-invests-35m-in-99designs-after-years-of-trying/

Lacy, S. (2012). http://pando.com/2012/01/24/get-over-it-haters-99designs-has-tipped/

Lakhani, K.R., Jeppesen, L.B., Lohse, P.A. and Panetta, J. A. (2007). The value of openness in scientific problem solving. Harvard Business School Working Paper.

Moldovanu, B. and Sela, A. (2001). The optimal allocation of prizes in contests. American Economic Review,91 (3), 542-558.

McGonigle, T.P. and Curnow, C.K. (2006). Measures of training and experience. Applied Measurement Methods: Industrial Psychology in Human Resources Management. 1st Edition. Psychology Press.

Terwiesch, C. and Xu, Y. (2008). Innovation contests, open innovation, and multiagent problem solving. Management Science, 54(9),1529-1543.

Yang, J., Adamic, L.A. and Ackerman, M.S. (2008). Crowdsourcing and knowledge sharing: strategic user behavior on taskcn. In Proceedings of the 9th ACM Conference on Electronic Commerce, p. 246-255, New York, NY, July.

Yang, Y., Chen, P. and Banker, R. (2011). Winner determination of open innovation contests in online markets. In Proceedings of the International Conference on Information Systems, p.16, Shanghai, China.

Yuen, M., King, I. and Leung, K. (2011). Task matching in crowdsourcing. IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing, p. 409-412, Dalian, China.

Zhou, X.,Wang X. and Dougherty, E.R. (2006). Multi-class cancer classification using multinomial probit regression with bayesian gene selection. Systems Biology, 153(2), 70-80.