WorkerRank: Using Employer Implicit Judgements To Infer Worker Reputation

Maria Daltayanni
UC Santa Cruz
mariadal@cs.ucsc.edu

Luca de Alfaro
UC Santa Cruz
luca@cs.ucsc.edu

Panagiotis Papadimitriou
Elance-oDesk
papadimitriou@elance-odesk.com

ABSTRACT

In online labor marketplaces two parties are involved; employers and workers. An employer posts a job in the marketplace to receive applications from interested workers. After evaluating the match to the job, the employer hires one (or more workers) to accomplish the job via an online contract. At the end of the contract, the employer can provide his worker with some rating that becomes visible in the worker online profile. This form of explicit feedback guides future hiring decisions, since it is indicative of worker true ability. In this paper, first we discuss some of the shortcomings of the existing reputation systems that are based on the end-of-contract ratings. Then we propose a new reputation mechanism that uses Bayesian updates to combine employer implicit feedback signals in a link-analysis approach. The new system addresses the shortcomings of existing approaches, while yielding better signal for the worker quality towards hiring decision.

Categories and Subject Descriptors

H.4.4.3.4 [**Information Systems**]: World Wide Web—*Web applications, Crowdsourcing, Reputation Systems*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Elo Ratings, Reputation, Link Analysis, Crowdsourcing

1. INTRODUCTION

In online labor marketplaces, such as oDesk [1], Elance [2] and Freelancer [3], two parties are involved; employers and workers. Employers post job openings and candidate workers apply to them, based on their qualifications, skills and interests. The employers review the applicants' online resumes, and interview few applicants to decide hiring. The worker reputation, i.e., the ratings that the worker has received in his past jobs in the platform, is one of the most important considerations for the employer hiring decision, since it reveals how other employers evaluate the worker true ability in real job scenarios. Although the reputation information is a useful signal, reputation scores are usually *skewed* towards high ratings [19], because employers care about the impact of their feedbacks on the workers' future opportunities for jobs in the marketplace. The skewed distribution of ratings makes them less helpful in identifying very competent workers.

The reputation signal is also very *sparse*, since a worker needs to apply, get hired and complete few jobs to obtain a representative reputation score. Usually, an unknown rating implies that we have no explicit information about the employer's preference for the worker. In that case we need to build a model to predict the unknown information, or, alternatively make inferences from the employer's behavior[4].

To address the limitations of the existing reputation systems in labor marketplaces, we present *WorkerRank*, a new reputation system that leverages employers' implicit judgements at the application evaluation moment, rather than the employer's explicit feedback at the job completion moment. Although the implicit judgements are more noisy than the explicit ones, they are more broadly available, since the number of applications is usually one to two orders of magnitudes higher than the number of hires. Moreover, the implicit actions of the employers are not revealed and, consequently, the employers do not bias their judgements towards high ratings (as happens when they aim to avoid the negative impact on the workers). As a result, the obtained ratings are not skewed.

We consider an employer decision to hire worker A, thus ranking A above some other candidate B, as an input that "A won over B" in a match. The employer decisions can thus be interpreted as a set of match outcomes. There are many algorithms ([16], [17], [18], [31]) that can be used to aggregate match outcomes. Our reputation system builds upon the Elo ratings system[16] that is widely used to evaluate chess players. In particular, we assign each worker an initial rating and we treat the applicants to a job opening as the participants in a chess tournament. Applicants that get hired get their scores increased and those who are rejected get their scores decreased. The extent of the increase or the decrease depends upon the ratings of the other applicants, i.e., the better the rejected applicants are, the more the rating of the hired worker increases. Similarly, the worse the hired applicants are, the more the ratings of the rejected applicants decrease.

To deal with the noise of implicit judgements, we assign each employer a score that quantifies the agreement of his decisions with the observed quality of the workers. We then use the obtained

[1]http://www.odesk.com
[2]http://www.elance.com
[3]http://www.freelancer.com

scores to weigh the employer judgements. For example, if an employer tends to take decisions that are very different from the rest of the employers, his score will be low and his hiring decisions will have a small impact on the worker ratings. The rest of the paper is organized as follows. In Section 2 we present some notation and in Section 3 we introduce WorkerRank, the new proposed reputation system. We evaluate the new reputation approach on a real-world dataset from oDesk in Section 4. Our results show that the new reputation system not only provides information for far more workers in the marketplace, but it also serves as a better discriminatory signal for hiring decisions. In Section 5 we discuss some related work and we conclude in Section 6.

2. NOTATION

We represent the labor marketplace data with a directed bipartite graph $G = (U, V, A)$ (Figure 1); U is the set of jobs posted by employers within a specific time period; V is the set of workers who applied to the posted jobs (see Figure 1). Edge $(v, u) \in A$ represents the application of worker $v \in V$ to job $u \in U$. Edge $(u, v) \in A$ represents the employer action on the the worker's application. We consider the following six employer actions:

- *hire*, the employer hires the worker;
- *interview*, the employer contacts the worker to obtain a better understanding of his skills, but the worker is not eventually hired;
- *shortlist*, the employer shortlists the worker for future consideration, but the worker is not invited for interview;
- *ignore*, the employer reviews the worker online resume, but he takes no action on it;
- *hide*, the employer reviews the worker resume and he "hides" the applicant without notifying him; and
- *reject*, the employer reviews the worker resume and notifies him that he will not be considered for the job.

Among the six actions, we consider the first three as positive indications of the worker ability to accomplish the posted job, while the last three are rather negative. We also assume that the employer actions imply some ranking on the applicant perceived ability to accomplish the job in the following decreasing order: hire > interview > shortlist > ignore > hide > reject. For example, a worker that is selected to be interviewed is considered to be a better fit for the job than a worker who is ignored.

The goal of this paper is to compute a score $r(v)$ for each worker v that is informative of the worker ability to accomplish the jobs that he applies to. A score $r(v)$ is considered informative if the relative difference between scores $r(v)$ and $r(v')$ for workers v and v' is predictive for the relative ranking of v and v' in the future jobs that they apply.

3. PROPOSED REPUTATION SYSTEM

In this section we describe a reputation system that builds upon the employer decisions on the worker applications. In Section 3.1 we provide our generic approach and in Section 3.2 we show how we can improve our scores by leveraging job specific information. Finally, in Section 3.3 we discuss how we can combine our reputation system scores with the end-of-contract ratings to obtain a hybrid reputation system.

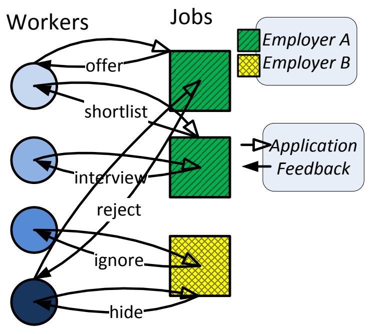

Figure 1: Bipartite graph between workers and jobs posted by employers

3.1 WorkerRank

The WorkerRank reputation system assigns *reputation* scores $r(v)$ to each worker $v \in V$. Along with reputation scores, WorkerRank also computes an *importance* score $b(u)$ for each opening $u \in U$ that reflects how a job is important in terms of how objective its employer is when judging candidates. The scores are computed via a reputation calculation process on the application graph G, using the Elo constants for t_{elo}, K, as shown in Algorithm 1.

Comparing Performances: The intuition of simulating jobs by tournaments, and looking at the worker performances at each job in pairwise manner, offers a dynamic way for comparing performances in several cases; first, higher label workers clearly win lower label opponents (for example, hire wins over interview, shortlist wins over hide). Also, draws in positive label workers provide useful information about how their qualities compare. While a draw between two hired workers is an instance of equality between their qualities, the same does not necessarily hold, though, in the case of draws between negative label workers (such as two rejected candidates). That is because in certain cases, rejecting or hiding a candidate may reflect the fact that the employer had offered only a limited number of positions hence he had to reject some good quality candidates. Other similar exceptions may apply too. Hence in our algorithm we exclude draws among negative label workers.

Algorithm: In step 1 we initialize reputation $r(v)$ of each worker v to 1.0 and importance $b(u)$ of each opening to 1.0. Then, in steps 3 - 13 we consider each job as a tournament and in steps 5 - 13 we update the worker scores by considering every pair (v, v') of worker applications at a job u to be a game in the job tournament with possible outcome of matches:

$$t(v, v', u) = \begin{cases} 0, & \text{if } v \text{ lost against } v' \text{ at job } u \\ 0.5, & \text{if } v \text{ came to draw with } v' \text{ at job } u \\ 1, & \text{if } v \text{ won against } v' \text{ at job } u \end{cases} \quad (1)$$

At step 3 we initialize the outcome variables $T_{v,u}, X_{v,u}$ to 0 for each candidate v who applied to job u. At step 5, we compute $T_{v,u}$ as the sum of the actual points that v scored in job u against the other opponent candidates. At step 6, we compute $X^i_{v,u}$ as the sum of expected points that v would earn at time i against each opponent candidate $v' \neq v$ at job u, according to Elo's formula[16]:

$$t^i_{elo}(v, v', u) = \frac{1}{1 + 10^{(r^i(v') - r^i(v))/400}}, \forall v' : (v', u) \in A \quad (2)$$

For example, consider workers v_1, v_2, v_3 and v_4 who applied to a job; v_1 gets an offer, v_2 is interviewed but never hired and v_3 and

Algorithm 1 WorkerRank: Compute Workers Reputation Scores and Jobs Importance Scores

```
Input: Graph $G = (U, V, A)$
Output: Reputation scores $r(v)$ for workers $v \in V$, importance
    scores $b(u)$ for jobs $u \in U$
 1: Initialize $i = 0$; $r^0(v) = 1, \forall v \in V$; $b^0(u) = 1, \forall u \in U$
 2: for $u \in U$ do                                  ▷ for each job tournament
 3:     $T_{v,u} = 0, X^i_{v,u} = 0, \forall v : \exists(v, u) \in A$
 4:     for $v, v' : (v, u) \in A, (v', u) \in A, v \neq v'$ do
 5:         $T_{v,u}$ += $t(v, v', u)$, and $T_{v',u}$ += $t(v', v, u)$
 6:         $X^i_{v,u}$ += $t^i_{elo}(v, v', u)$, and $X^i_{v',u}$ += $t^i_{elo}(v', v, u)$
 7:     end for                                   ▷ for each worker pair game
 8:     $i \leftarrow i + 1$
 9:     $\delta^i(v, u) \leftarrow b^{i-1}(u) \cdot (T_{v,u} - X^{i-1}_{v,u})$
10:     $\delta^i(v', u) \leftarrow b^{i-1}(u) \cdot (T_{v',u} - X^{i-1}_{v',u})$
11:     $r^i(v) \leftarrow r^{i-1}(v) + \frac{K}{\binom{n}{2}} \cdot \delta^i(v, u)$        ▷ Update reputation
12:     $r^i(v') \leftarrow r^{i-1}(v') + \frac{K}{\binom{n}{2}} \cdot \delta^i(v', u)$
13:     $b^i(u) \leftarrow \frac{n^i_c(e) - n^i_w(e)}{n^i_c(e) + n^i_w(e)}$        ▷ Update job importance
14: end for
```

v_4 are rejected without interview. Each applicant participates in 3 games versus the other applicants. Worker v_1 wins all three games versus v_2, v_3 and v_4, since he received an offer which is the most positive employer judgement. Worker v_2 loses against v_1 but wins over v_3 and v_4. Finally, each of the workers v_3 and v_4 loses in the games against v_1 and v_2 but they draw when they face each other. The worker points in this job are 3 for v_1, 2 for v_2 and 0.5 for either of v_3 and v_4.

At step 9, the rating update $\delta^i(v, u)$ of worker v due to his application to job u at the i-th time step is calculated as follows:

$$\delta^i(v, u) = b^{i-1}(u)(T_{v,u} - X^{i-1}_{v,u}) \tag{3}$$

where $b^{i-1}(u)$ is the importance score of job u from the $i-1$-th step, $T_{v,u}$ is the sum of the actual points that v scored in job u and $X^{i-1}_{v,u}$ is the sum of points he was expected to score based on his rating and the ratings of the other applicants from the previous step. Time steps advance at each new job occurrence. Note that the score update is multiplied by the importance of the job, $b(u)$, so that employer bias is taken into account, as shown in Equation 5. Also note that the more the applicants in an opening, the more the expected points, since the update includes a summation term for each applicant. What is more, the higher the difference between the rating of worker v and the ratings of the other applicants of job u, the more the points that v is expected to score. This is particularly useful since application success of an applicant is not independent from the application success of the remaining candidates at a particular job.

Finally, to obtain the rating of worker v at the i-th time step, at step 11, we add the average of his partial rating updates (Equation 3) to his rating from the previous time step, $r^{(i-1)}(v)$:

$$r^i(v) = r^{i-1}(v) + \frac{K}{\binom{n}{2}} \cdot \delta^i(v, u) \tag{4}$$

The K-factor which represents the maximum possible adjustment per game is set to 32, and is normalized by dividing over $\binom{n}{2}$, where n is the number of the job applicants and 2 is the number of players in the pair-wise worker comparison. Normalization is completed by the summation over all pairs of candidates. Normalization is important, since without it, worker scores would be increased or decreased by the square of the number of applicants, which would lead to dramatic inflation/deflation of scores at very popular jobs (that is, jobs with high application rate). Very popular jobs usually require generic skills, hence inflated scores would not necessarily imply that the quality of their hired workers is higher.

After calculating the worker ratings, at step 13 we compute the importance scores of jobs $b(u)$. The intuition in Formula 5 is to give more credence to rating updates (Equation 4) that come from jobs posted by unbiased employers. The importance score is high for employers who make decisions that respect the worker ratings and it is low for employers who do not. Notice that after the occurrence of a small amount of job tournaments, the worker ratings is an outcome taken out of the aggregation of all employers judgements (step 11). At the same time, Elo scoring is based on a self-correcting rating system[1]. Hence the following formula reflects importance of a job as a measure of judgement deviation between the job's respective employer and the rest employers:

$$b^i(u) = \frac{n^i_c(e, u) - n^i_w(e, u)}{n^i_c(e, u) + n^i_w(e, u)} \tag{5}$$

$e \in E$ denotes employer (in the set of employers E) who posted job $u \in U$, $n_c(e, u)$ denotes the number of applicant pairs that were "correctly" ranked by employer e in job u and $n_w(e, u)$ denotes the number of applicant pairs that were "wrongly" ranked. We regard a pair of applicants as correctly ranked if the employer prioritizes the applicant with the highest reputation score. For example, if applicants v and v' have ratings $r(v) = 1$ and $r(v') = 2$ and employer e hires v' and rejects v, then the pair is considered to be correctly ranked.

3.2 Skill WorkerRank

The approach described in Section 3.1 predicts a reputation score for each worker based on application data. That score is computed in a global scope over all jobs where workers have applied. Although global scores provide a signal for the quality of workers, they may not be as powerful to discriminate among similar score workers and guide hiring with accuracy. For example, consider candidates v_1, v_2 in the example shown in Figure 2. The skills listed under each job reflect the required skill-set the employer has set in order for the candidates to get hired. Also assume that v_2 is better in java than v_1, however v_1 is better overall than v_2, which can be caused from the fact that v_1 has applied to more jobs where he has been successful (received offers). At this point our reputation system would prioritize v_1 in the candidates list, missing the fact that v_2 is better in java.

Our goal is to achieve higher accuracy at predicting worker quality and prioritize candidates appropriate for the particular job. As mentioned above, besides the information regarding which worker applied to which job, there is further information regarding skills; what skills each job requires and what skills each worker claims in their profile description. Given this information, we use WorkerRank to derive scores for candidates in a skill-wise fashion, such that eventually we learn how good each worker is at each particular skill. Then, if a job requires a skill, we may rank candidates according to their reputation scores at that particular skill and suggest the top ranked ones for getting hired. In the used example, workers v_1, v_2 who both claim to be experts in java (and they apply to job u_2 which requires java) will obtain a java score that will show their quality in that skill. The expectation is that ranking v_2 on top will lead to successful hiring decision.

Skill-wise reputation algorithm: We consider set of skills S, where $S_u \subseteq S$ denotes the skills required for job $u \in U$ and $S_v \subseteq S$ denotes the skills claimed by worker $v \in V$. Also, we

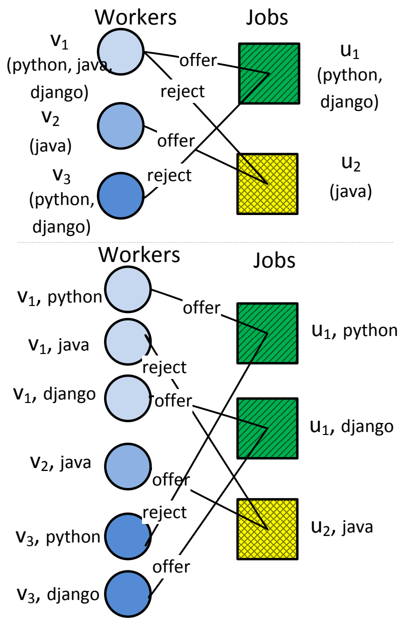

Figure 2: Skill-wise bipartite graph

consider bipartite graph $G_S = (U \times S, V \times S, A_S)$ similar to the definition in Section 3.1, where:

- each worker node v is replaced by set of pair {worker, skill} nodes, $\{v, s\}$, one node for each skill claimed by the worker
- each job node u is replaced by set of pair {job, skill} nodes, $\{u, s\}$, one node for each skill required for the job
- each (worker, job) edge (v, u) is replaced by set of pair ({worker, skill}, {job, skill}) edges, $(\{v, s\}, \{u, s\})$, one edge for each skill that the worker claims and the job requires.

Then we run Algorithm 1 on G_S. The reputation and importance scores are derived skill-wise, such that we obtain a set of reputation scores $r(v, s)$, where$(v, s) \in V \times S$ and a set of importance scores $b(u, s)$, where$(u, s) \in U \times S$. Obtaining reputation scores for each {worker, skill} pair provides information about the performance of the worker at the particular skill.

Figure 2 describes an example for graphs G and G_S, with candidates v_1, v_2, v_3 applying to jobs u_1, u_2. Workers v_1, v_3 apply to job u_1 (v_1 receives an offer) and v_1, v_2 apply to job u_2 (v_2 receives an offer). The skills required for job u_1 are {python, django} and the skills required for job u_2 are {java}. Worker v_1 claims to have skills {python, java django}, v_2 claims {java}, and worker v_3 claims {python, django}.

Correlation between skills and hires: Looking at the application data (enriched with skills information) we observe that in most cases jobs require more than one skills. In that case, we need to decide a ranking for candidates based on the intersection of their scores on a set of different skills. That ranking will reflect their suitability for the multi-skill requiring job. In our example where job u_1 requires python and django, we need to rank candidates according to their quality in python and their quality in django. However, the python-score may be more informative about hiring than the django-score. For example, it may more beneficial to hire a candidate with a high python-score than hire one with a high django-score. Hence it is important to measure how each skill contributes towards hiring, before we rank candidates according to skills.

In order to combine a set of scores for each worker across the set of skills required for a job, we allow for a weighted average over the worker's skill-wise scores. We use logistic regression to compute coefficients for skills as features, where we use the binary outcome of the application (hire/no-hire) as the response variable. Coefficients for skill scores will eventually show how informative each skill is about the quality of the worker, measured by the worker's potential of getting hired.

In Algorithm 2 we aggregate skill-wise scores into a single reputation/importance score for each worker/job respectively. The input of the new algorithm is the set of scores derived by Algorithm 1 for the sets of {worker, skill}, {job, skill} pairs. The output of Algorithm 2 is the final reputation score for each worker and importance score for each job, after examining workers' quality across their skill-set and tuning it according to the significance of each skill.

Algorithm 2 Combine Skill-wise Scores into Reputation

```
Input: Set of skill-wise scores $r(v, s)$, $b(u, s)$, where $(v, s) \in V \times S$, $(u, s) \in U \times S$
Output: Reputation scores $r(v)$ for workers $v \in V$, Importance scores $b(u)$ for jobs $u \in U$
1: Consider feature variables $f(s) \leftarrow r(\cdot, s)$, $\forall s \in S$, and set of feature variables $F \leftarrow \cup_{s \in S} f(s)$
2: Consider response variable $y \leftarrow$ hiring outcome, where $y \in \{hire, no\text{-}hire\}$
3: Learn coefficients $w(f) \leftarrow LR(F, R), \forall f \in F$
4: for $v \in V$ and $u \in U$ do
5:     $r(v) \leftarrow \frac{\sum_{s \in S} r(v, s) \cdot w(s)}{\sum_{s \in S} w(s)}$
6:     $b(u) \leftarrow \frac{\sum_{s \in S} b(u, s) \cdot w(s)}{\sum_{s \in S} w(s)}$
7: end for
```

In step 1 we consider set of features F, where one's reputation score $r(\cdot, s)$ at a particular skill $s \in S$ is regarded as a feature. We use $f(s) = r(\cdot, s)$ to denote feature regarding reputation score at skill s. For example, if $s =$ python, then any (denoted by '$\cdot$') worker's score in python, $r(\cdot, \text{python})$, is a feature. Recall that the coefficients pertain to how each skill score of a worker contributes towards his getting hired. In step 2 we consider response variable y to be the binary hiring outcome $y \in \{hire, no\text{-}hire\}$. Then in step 3 we run logistic regression (LR) on the set of features F with response variable y and we obtain coefficients $w(f)$ for each skill-score variable $f \in F$. Finally, in steps 5 - 6 we aggregate the input skill-wise scores using weights across skills learned at step 3. The output of the algorithm is a single reputation score for each worker (step 5) and a single importance score for each job of the application data (step 6).

3.3 Hybrid Model

While it is interesting to compare implicit judgements against explicit judgements in order to infer a quality measurement for workers, we expect that a hybrid model which combines both, shall yield better results. In this section we use rank aggregation to combine WorkerRank ranking with feedback ranking into an optimal listing of workers such that we predict true ranking (as specified by employer judgements) with higher accuracy.

In particular, we use the weighted rank aggregation method described in [29]. In this approach the function performs rank aggregation via a Cross-Entropy Monte Carlo algorithm. The algorithm searches for a desired list which is as close to the provided ordered lists as possible. In our implementation we use the Spearman distance to measure the correlation of the ordered lists of implicit and explicit reputation rankings. The convergence criterion used is the repetition of the same minimum value of the objective function in a number of consecutive iterations.

4. EXPERIMENTAL RESULTS

To evaluate the proposed reputation system, we compare WorkerRank with baseline schemes in terms of a) the sparsity of the signal in the marketplace, b) the time needed to obtain a signal for new workers, and c) discriminatory power for hiring decisions. During the evaluation, WorkerRank is compared against a baseline collaborative filtering approach which uses data of *implicit* reputation to rank workers (as a reminder, WorkerRank also runs on implicit reputation data). In addition, we compare WorkerRank against the current ranking approach that is based on *explicit* reputation data.

4.1 Setting

Dataset: We use a sample of real-world application data, along with explicit reputation scores provided by oDesk[27]. The oDesk dataset spans the time period of 53 weeks between January 2013 through January 2014 and it contains approximately 10M applications submitted by 0.5M workers to 1.1M job openings posted by 0.2M employers. Note that we do not account for unseen applications (case where employer has taken no action). In table 1 we provide some statistics regarding the dataset. In the experiments we use a sample of the applications submitted during the testing period. Tables 2 and 3 show a real job posting example. This example includes information about the job title, category, description, required skills, candidate applicants along with their declared skills, hire decision, reputation scores learned via WorkerRank and via skill-based WorkerRank. We observe overlapping skills among the job required skills and the skill-sets declared by the candidates. In Figure 4 we show the distribution of the ratings data. As studied in [19], we encounter skewness towards the high rating values.

Baseline - Explicit Data: A baseline approach currently used in the marketplace is to represent the quality of workers based on *explicit* data; in particular, it collects the explicit star ratings assigned to each worker by the employers according to their performance on accomplished jobs, and aggregates them in an average ratings score. Then the workers quality is estimated according to the average employers judgements. The average employers rating $q(e, v)$ is used to rank candidate workers $v \in V$ who apply at a new job posting of employer $e \in E$, as follows:

$$q(e,v) = \frac{1}{N} \sum_{e':(v,u')\in A} \sum_{u'\in U_{e'}} q(e',v) \qquad (6)$$

where $(v, u') \in A$ denotes application of worker v to job u' posted by employer e', $U_{e'}$ is the set of jobs posted by e', $q(e', v)$ is the explicit ratings score in $[1, 5]$ assigned to v by e', and $N = \sum_{e'} |U_{e'}|$ is the number of accomplished jobs, posted by employers e'. Note that employers provide explicit reviews only to workers who have accepted their offer and upon completion of the job.

Baseline - Implicit Data: In certain cases, reputation systems are used to provide rating scores in recommendation problems. Since WorkerRank is applied on implicit reputation data represented on a hiring actions bipartite graph, we use a collaborative filtering score scheme as one of the baseline approaches, viewing WorkerRank as a collaborative filtering approach based on implicit data. The baseline for representing quality of workers based on *implicit* reputation data, is to compute their hire rate in the training set neighborhoods. We collect the sets of jobs at which workers received an offer within the study training period. Then for each employer we count the total number of offers assigned to each worker by the neighbor employers in the training set and we estimate worker quality score according to their hire rate. For employer e who posts a new job u, we compute recommended score for candidates v as the hire rate in jobs posted by employers e', neighbors of e, weighed by their similarity, $sim(e, e')$. Denoting employer implicit actions (offer, interview, reject, and more) by $q_{im}(e', v)$, we consider the following baseline implicit reputation score:

$$q_{im}(e,v) = \frac{1}{\sum_{e'} |sim(e,e')|} \sum_{e':(v,u)\in A} \sum_{u\in U_{e'}} sim(e,e') q_{im}(e',v) \qquad (7)$$

where $q_{im}(e', v)$ takes values in $\{0$ if v was not hired, 1 else$\}$. *Similarity* between employer e and neighbor e', $sim(e, e')$, is defined by their cosine similarity based on their ratings on workers that e and e' have co-rated. Then e receives recommendations from the neighbor e' judgements [4].

In the testing phase we rank candidates by collaborative implicit hire rates and by explicit ratings reputation to recommend the top ranked workers for the new job openings. Then we compare the two baseline performances with WorkerRank performance.

Table 1: Dataset Statistics

	Training	Testing
Application dates	$03/01/2013$ – $03/01/2014$	$03/02/2014$ – $18/02/2014$
#Jobs	$1,151,859$	$1,446$
#Workers	$477,464$	$21,642$
#Employers	$232,014$	$1,405$
#Applications	$9,214,557$	$34,054$
Avg # cands / job	24.1	22.5
Min # cands / job	11	11
Max # cands / job	50	50
Median # cands / job	23	20

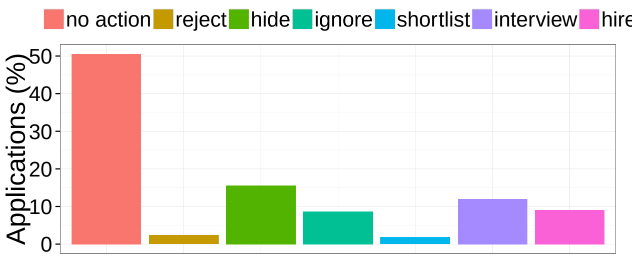

Figure 3: Histogram of application success label rates

Table 2: Job Posting Example (Applicants shown in table 3)

Title	*Wordpress Developer*
Category	*Web Development*
Required Skills	*wordpress, css, php*
Description	*1.Need Wordpress theme developed. Slider, logo, left sidebar menu, copyright.* *2.Must be experienced Wordpress developer.* *3.Must know CSS, php, Wordpress, html.* *4.Will provide visual design guide.*

Table 3: Skill-based reputation versus global reputation scores

Candidate	App.Success	Skill 1	Skill 2	Skill 3	Skill 4	Skill5	reputation	skill-reputation
v_1	offer	css3	php	wordpress	html	css	2.54	2.28
v_2	**offer**	css3	php	wordpress	html5	css	1.42	**4.67**
v_3	offer	web dev	sofw dev	–	–	–	1.39	2.05
v_4	**reject**	css3	php	wordpress	html5	ajax	**4.82**	3.40
v_5	reject	gr.design	visual-c++	wordpress	web design	illustration	1.38	1.06
v_6	reject	css3	php	javascript	html	jquery	1.05	2.45

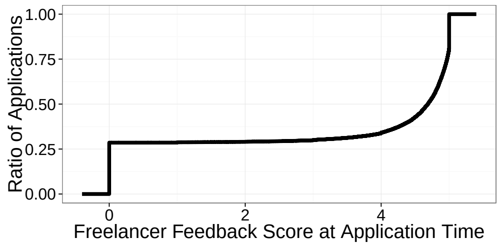

Figure 4: Data Skewness

4.2 Coverage

First, we show that since WorkerRank's results become available at the time of application, the coverage of workers for whom we obtain reputation signal is higher compared to the coverage obtained from explicit ratings. In particular, we run WorkerRank over the applications of the first 52 weeks of the dataset. During this time period we also keep track of the feedback ratings that the workers receive after the job is completed. Then we report the number of applications of the 53-rd week for which there is a WorkerRank score versus the applications for which there is an employer feedback score. Our results show that out of 88, 294 applications in the 53-rd week, we have WorkerRank scores for 79, 083 (89.6%), while we have feedback scores only for 52, 471 (59.4%). The increase in the marketplace application coverage is +50.8%. We present these results in table 5. Note that the above measurements account for both active and inactive applications.

4.3 Cold Start

Second, we show that WorkerRank is faster in acquiring signal for new workers joining the system, compared to the explicit ratings approach. Since the online marketplaces grow fast, the identification of new competent workers is very significant for their healthy development. For all workers who joined the platform during the last 3 months of our study period, we calculate the percentage of workers for whom we obtain reputation signals within X days. X is varying from 1 to 120 days. As presented in Figure 5, the WorkerRank scores are available for more than 60% of the new workers within 3 months of their joining the platform and the percentage ratio grows to 75% after another 4 months. On the contrary, there are less than 10% of new workers who received feedback at the end of their first 3 months in the platform and this percentage does not exceed 18% at the end of the additional 4-month period.

4.4 Ranking Precision

Third, we show that WorkerRank outperforms baseline approaches accuracy in ranking workers by quality, thus yielding a more reliable system for ranking candidates in new job openings.

WorkerRank vs Implicit vs Explicit Baselines:

MAP: We use Mean Average Precision (MAP) to evaluate rankings produced by the baseline approaches and the WorkerRank scores. The truth rankings each method is compared against, are specified by employer true scores that they implicitly assigned to candidates through their hiring actions in past jobs. As shown in Figure 6, by using WorkerRank, employers encounter approximately 1 good worker for every 3 workers in the ranked list. That performance is compared against 1 good worker for every 5 (or, 4 respectively) workers that the employer encounters by using the baseline collaborative filtering ranking (or, explicit ratings, respectively). Overall, the new algorithm improves the chances of identifying good workers in the top results by 39.1% (33.3%, respectively).

Lift: To evaluate the quality of WorkerRank scores, we compare

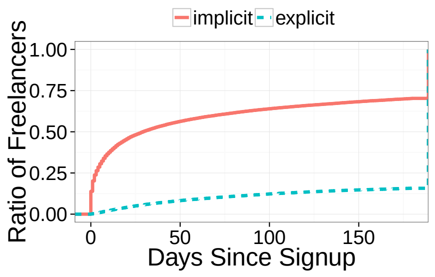

Figure 5: Cold Start: WorkerRank vs Explicit Feedback

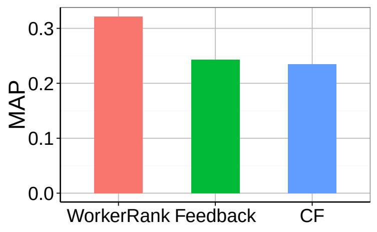

Figure 6: MAP in predicting the hiring outcome

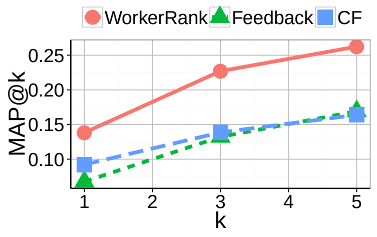

Figure 7: MAP@k in predicting the hiring outcome

Table 4: Performance Improvement: WorkerRank vs Baselines

	Explicit Feedback	Implicit Reputation		% Improvement	
	Ratings Based (RB)	Collab Filtering (CF)	WorkerRank (WR)	WR vs CF	WR vs RB
MAP	0.24	0.23	0.32	**+39.1%**	**+33.3%**
AUC	0.59	0.52	0.65	**+25.0%**	**+10.1%**
% Covered applications	59.4%	–	80.8%	–	**+50.8%**

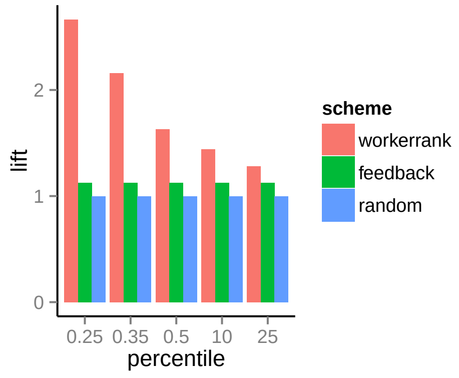

Figure 8: Lift in predicting the hiring outcome

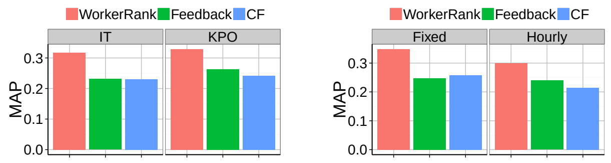

Figure 9: MAP across different job segments and types

them against the explicit reputation scores as signals for taking hiring decisions. We use the data of the first 52 weeks of our dataset to calculate the WorkerRank scores, and we then use these scores as predictors for the hiring outcomes of the applications submitted during the 53-rd week. In particular, we rank all of the applications by the WorkerRank scores of the applicants. Then we calculate the hiring *lift* in the top x percent of the applications as follows:

$$lift(x) = \frac{\text{hiring probability in the top-}x\text{\% applicants}}{\text{hiring probability across all applicants}} \quad (8)$$

Lift shows the performance of WorkerRank versus the performance of a random scoring of applicants. Similarly, we calculate the lift for the explicit ratings scores and we present the results in the barplot of Figure 8. The plot has five triplets of bars and each triplet looks at a different percentage value of the top ranked workers, $x \in \{0.25, 0.35, 0.5, 10, 25\}$. Left bars look at the Elo ratings obtained by WorkerRank, center bars look at the explicit feedback ratings, while right bars reflect random worker ranking. The height of each bar shows the lift value for the corresponding scheme and for the corresponding x value. For example, the first left bar from the left shows that the top-0.25% of applicants as ranked by the Elo ratings are 2.66 times more likely to be hired than a random applicant. We observe that the lift of the explicit feedback scores is flat at 1.1 for all x values, since the top 25% of the applications correspond to workers with perfect 5-score rating. As a result, the existing reputation system does not provide a sufficient signal to discriminate high quality workers. On the other hand, WorkerRank Elo ratings yield an increasing lift as we limit the percentage of the top-$x\%$ applications that we consider. The Elo lift is already higher than the feedback lift for $x = 25\%$ and it exceeds 2.5 as we limit x to the top 0.25% of the applications. Note that for an average of 36 applications per job, that percentage implies that the top 9 candidates are 2.66 times more likely to be hired than any random candidate, as opposed to the almost equal likelihood that explicit feedback yields (1.1 times more likely than random) .

Performance across Job Types: We also illustrate how WorkerRank outperforms the baseline approaches when studied in category subsets of the datasets. Figure 9(a) shows how the algorithms perform when applied on the different segments of jobs, Knowledge Processing Outsourcing (KPO) and Information Technology (IT). Figure 9(b) shows how the algorithms perform when applied on different type jobs (fixed-price (fp) versus hourly-rate (hr)). An interesting illustration is that of Figure 4.4, which shows how the different approaches behave on different job categories. For example, employers in software engineering jobs appear to provide more precise feedback about the quality of workers they have collaborated with, whereas in sales and marketing feedback ratings do not reflect the overall quality of workers as accurately.

Hybrid: WorkerRank and Explicit Reputation: In Figures 11(a) and 11(b) we show how the hybrid approach performs at MAP and MAP@k: for $k \in [1, 5]$, compared to the two approaches it combines; WorkerRank and explicit reputation. The hybrid model appears to slightly improve WorkerRank, the best of two approaches, although the improvement is not as high as it was in the comparison between implicit versus explicit reputation in Figures 6 and 7. It is interesting to mention that in all cases the hybrid model improves WorkerRank and explicit feedback, except for the hourly rate jobs and the Web Development category, where WorkerRank marginally outperforms the hybrid model and significantly outperforms the explicit reputation model.

Finally, during rank aggregation which derives the hybrid model, we tested a few weighting combinations to prioritize the influence of one of the two rankings (WorkerRank or explicit). Equal weights on the two rankings appears to be the best combination that makes the hybrid model behave optimally. The Figures shown for the hybrid model performance assume equal weight on the two lists.

Skill-wise WorkerRank vs WorkerRank: In Figure 12 we show how skill-wise WorkerRank performs at MAP compared to the plain WorkerRank approach. Since skills are available for approximately 65% of the jobs, we use a subset of the dataset in this experiment. As mentioned earlier, skill-wise WorkerRank produces specialized scores for workers, pertaining specifically to the skills that potential jobs require. The results show that ranking workers by skill-wise scores is more accurate than ranking them based on the WorkerRank scores. In particular, the improvement shown in the MAP Figure provides a better system to rank workers. However in certain cases skill-wise scores do not appear experimentally to differ from WorkerRank scores. That is because for several jobs a sin-

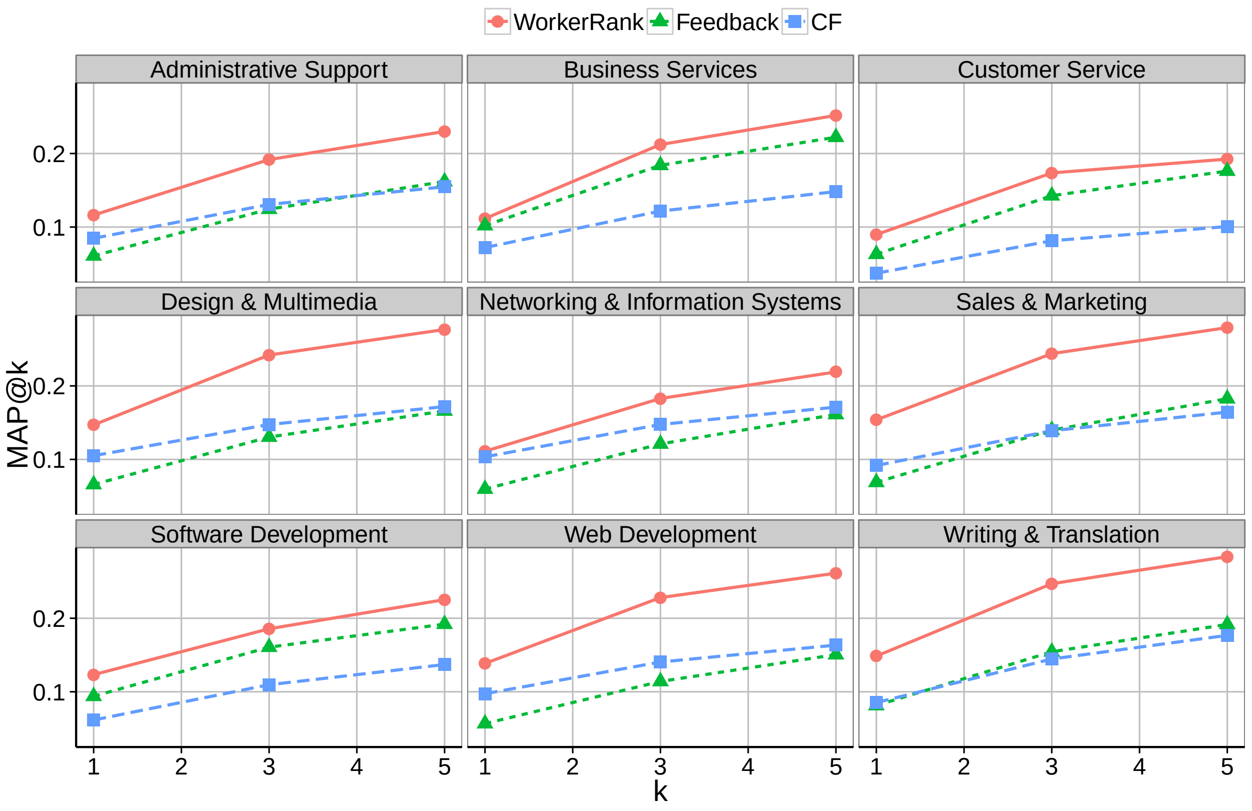

Figure 10: MAP across different job categories

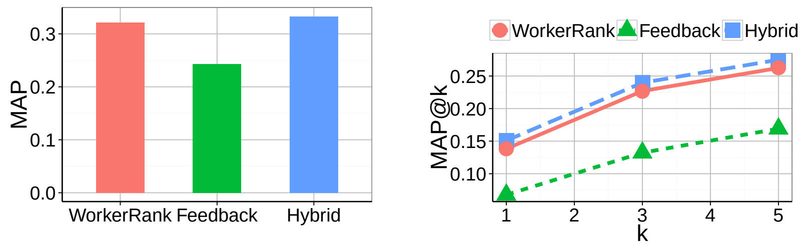

Figure 11: (a) MAP@Inf, (b) MAP@k for hybrid model in predicting the hiring outcome

Table 5: Performance Improvement: Hybrid vs WorkerRank

	WorkerRank	Hybrid	% Improvement
	WR	HB	HB vs WR
MAP@1	0.14	0.15	**+9.1%**
MAP@3	0.23	0.24	**+5.6%**
MAP@5	0.26	0.27	**+4.8%**
MAP@Inf	0.32	0.35	**+3.5%**

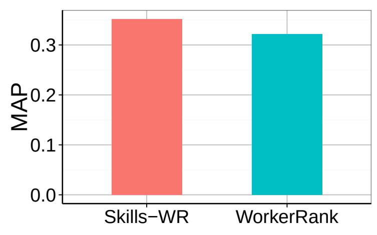

Figure 12: MAP in skill-wise WorkerRank

gle skill is specified instead of a set of skills. This is one of the reasons why skill scores do not add significant knowledge to our estimations about workers quality. We apply logistic regression to combine skill-wise scores in a weighted fashion such that the intersection of our knowledge about the ability of the workers on multiple scores is incorporated.

5. RELATED WORK

The problem tackled in this paper overlaps with four research fields; graph link analysis, building online reputation systems, game competition match analysis and predicting high-quality response in query-answering.

Graph link analysis research is related to our work, since we represent our data using a bipartite graph and we perform link analysis to examine the job applications of workers along with the respective employer feedback (edge weight). Several approaches have been proposed about ranking graph nodes in a network, such as PageRank [28], [10] and HITS [21], while Donato et. al. extend the study of HITS in [9] and Zhang et. al. in [36] study how PageRank and HITS perform when applied on the Java forum domain. Finally, Mishra et.al. [26], and Lescovec et. al. in [24] and [23] , present their node scoring methods with the presence of both positive and negative edge weights. Note that in our approach we also implicitly make use of negative information about applications, such as the "ignore", "hire" and "reject" feedback responses by the employer. However we only account for the relativity among different feedback labels, that is, who won over whom, hence we do not face restrictions of edge positivity.

Research on online reputation systems is directly related to our work, as we build a reputation system for workers in the labor marketplace, and we derive additional heuristic de-bias scores for employers. In [3] and [2] Adler et.al. tackle the problem of measuring the quality of contributions in Wikipedia, while Tan et.al. expand on this problem in [33]. Kokkodis et.al. in [22] discuss how to address data sparseness in building labor marketplace reputation systems. In [15] , Dellarocas summarizes online reputation mechanisms and challenges they face in terms of usage and evaluation. What is more, Archack in [7] discusses how reputation challenges strategic behavior of contestants in the TopCoder marketplace, while Chen in [12] describes their de-biasing mechanism for building a reputation system in a comments rating environment. TwitterRank [35] is another reputation system which aims to build reputation scores such that they incorporate a measure of influence for the Twitter users. Finally, in our past approaches in [14] and [13], we discuss the usage of link analysis using weighting schemes in order to build reputation systems.

It is interesting to reference a few game competition works such as the Elo method [16] that we are using in our current approach in order to predict the expected hire probability of each worker given our prior knowledge about their opponent's performance and their own performance. Elo is using a Bayesian update scheme to score chess game players based on past matches activity and update their scores by their expected performance in future tournaments. Glickman in [17] presents an improved approach, which keeps updating the mean and variance of the player scores such that confidence information is also carried along with the player's quality estimation. Methods tackling further improvement of match updates are proposed, such as TrueSkill [18] which tackles multi-player and multi-team challenges, while Nikolenko et.al. further improve TrueSkill's challenges of multiway ties and variable team size. Finally, Sismanis in [31] proposes a re-visit on the Elo method which incorporates tournament recency and other parameters in the tournament analysis in order to avoid over-fitting of the player ratings.

Moreover, query answering methods are referenced since they tackle the challenge of predicting quality of a response content such as question answers and social media content; that challenge is similar to our work's goal of predicting worker quality scores. Several approaches have been proposed aiming to identify quality in social media content such as Agichtein et. al. [5] and Bian et. al. [8]. Shah et.al. [30], Suryanto et.al. [32], Jurczyk et. al. [20] and Anderson et. al. [6] study quality of answers in question answering, while Tsaparas et.al. [34], study quality of online review systems, such as in Yelp or Epinions. Finally, Chen et. al. [11] study de-biasing approaches to set votes more informative in question-answering systems towards higher quality in answer and expert ranking.

Finally, Kokkodis et al. [25] formulate the problem of hiring in work marketplaces as a binary classification problem, where the target variable is the hiring decision. An informative reputation mechanism like the one we present in this paper can be a very predictive signal in such classifiers.

6. CONCLUSIONS

The results of our experiments show that WorkerRank improves ranking of candidates compared to baseline approaches, since its reputation scores reflect worker quality more accurately. What is more, WorkerRank solves the basic problems encountered in explicit reputation systems (unreliable employer ratings, limited coverage of worker scores, cold start problem for new workers with no history information). Our future work includes research on weighting schemes as discussed in[14] and modeling implicit actions on the marketplace website.

7. REFERENCES

[1] Elo rating system - Wikipedia, the free encyclopedia.

[2] B. T. Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 261–270, New York, NY, USA, 2007. ACM.

[3] B. T. Adler, L. de Alfaro, I. Pye, and V. Raman. Measuring author contributions to the wikipedia. In *Proceedings of the 4th International Symposium on Wikis*, WikiSym '08, pages 15:1–15:10, New York, NY, USA, 2008. ACM.

[4] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[5] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 183–194, New York, NY, USA, 2008. ACM.

[6] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites: A case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 850–858, New York, NY, USA, 2012. ACM.

[7] N. Archak. Money, glory and cheap talk: Analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on topcoder.com. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 21–30, New York, NY, USA, 2010. ACM.

[8] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 51–60, New York, NY, USA, 2009. ACM.

[9] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Transactions on Internet Technology (TOIT)*, 5(1):231–297, 2005.

[10] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[11] B.-C. Chen, A. Dasgupta, X. Wang, and J. Yang. Vote calibration in community question-answering systems. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 781–790, New York, NY, USA, 2012. ACM.

[12] B.-C. Chen, J. Guo, B. Tseng, and J. Yang. User reputation in a comment rating environment. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 159–167, New York, NY, USA, 2011. ACM.

[13] M. Daltayanni and L. de Alfaro. Recommending workers in the labor marketplace. *In Workshop: Data Design for Personalization: Current Challenges and Emerging Opportunities. In conjunction with WSDM 2014*.

[14] M. Daltayanni, L. de Alfaro, P. Papadimitriou, and P. Tsaparas. On assigning implicit reputation scores in an online labor marketplace. In *EDBT*, pages 724–725, 2014.

[15] C. Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Manage. Sci.*, 49(10):1407–1424, Oct. 2003.

[16] A. E. Elo. *The rating of chessplayers, past and present*. Arco Pub., New York, 1978.

[17] M. E. Glickman. The glicko system. 1995.

[18] R. Herbrich, T. Minka, and T. Graepel. Trueskilltm: A bayesian skill rating system. In B. Schoelkopf, J. Platt, and T. Hoffman, editors, *NIPS*, pages 569–576. MIT Press, 2006.

[19] N. Hu, J. Zhang, and P. A. Pavlou. Overcoming the j-shaped distribution of product reviews. *Commun. ACM*, 52(10):144–147, Oct. 2009.

[20] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 919–922, New York, NY, USA, 2007. ACM.

[21] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, Sept. 1999.

[22] M. Kokkodis and P. G. Ipeirotis. Have you done anything like that?: predicting performance using inter-category reputation. In S. Leonardi, A. Panconesi, P. Ferragina, and A. Gionis, editors, *WSDM*, pages 435–444. ACM, 2013.

[23] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 641–650, New York, NY, USA, 2010. ACM.

[24] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1361–1370, New York, NY, USA, 2010. ACM.

[25] P. I. D. Marios Kokkodis, Panagiotis Papadimitriou. On hiring decisions in online labor markets.

[26] A. Mishra and A. Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 567–576, New York, NY, USA, 2011. ACM.

[27] oDesk. https://www.odesk.com.

[28] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[29] V. Pihur, S. Datta, and S. Datta. Weighted rank aggregation of cluster validation measures. *Bioinformatics*, 23(13):1607–1615, July 2007.

[30] C. Shah and J. Pomerantz. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 411–418, New York, NY, USA, 2010. ACM.

[31] Y. Sismanis. How i won the "chess ratings - elo vs the rest of the world" competition. *CoRR*, abs/1012.4571, 2010.

[32] M. A. Suryanto, E. P. Lim, A. Sun, and R. H. L. Chiang. Quality-aware collaborative question answering: Methods and evaluation. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 142–151, New York, NY, USA, 2009. ACM.

[33] C. H. Tan, E. Agichtein, P. Ipeirotis, and E. Gabrilovich. Trust, but verify: Predicting contribution quality for knowledge base construction and curation. In *WSDM*, 2014.

[34] P. Tsaparas, A. Ntoulas, and E. Terzi. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 168–176, New York, NY, USA, 2011. ACM.

[35] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.

[36] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: Structure and algorithms. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 221–230, New York, NY, USA, 2007. ACM.