Task Personalization for Inexpertise Workers in Incentive Based Crowdsourcing Platforms

Ayswarya R Kurup *, G P Sajeev†
Dept of Computer Science and Engineering
Amrita School of Engineering
Amrita Vishwa Vidyapeetham, Amritapuri
India
Email: *ayswaryarkurup16@gmail.com, †sajeevgp@am.amrita.edu

***Abstract*—Crowdsourcing is an emerging technology which enables human workers to perform the task that cannot be done using automated tools. The crucial constituent of crowdsourcing platform is human workers. Since crowdsourcing platforms are overcrowded, workers find difficulty in selecting a suitable task for them. Employing task recommendation systems could improve this situation. However, task recommendation for new and inexpert workers is not explored well. We address this problem by designing a task recommendation model using skill taxonomy and participation probability of existing expert workers. The proposed model is validated through experimentation with both real and synthetic dataset.**

***Index Terms*—crowdsourcing; task recommendation; skill heirarchy; task personalization; inexpertise worker.**

I. Intorduction

Advances in Web 2.0 technologies has promoted the arise and use of collective intelligence. Crowdsourcing is such one technology which utilizes the collective intelligence effectively to turn social media in to social productivity [1]. Crowdsourcing is a distributed problem solving model which engages a group of online people for obtaining various services through an open call [2]. Now a days, it is widely used by organizations and individuals for obtaining input from human on problems for which automated computation is not possible or difficult and prohibitively costly [3].

A conceptual illustration of crowdsourcing workflow is illustrated in Fig. 1. The requester publicizes the tasks which to be solved in the crowdsourcing platforms. The workers choose tasks from the task pool according to their own choices and submits the completed task to the platform. The requester verifies the submissions and decides whether or not to reward workers based on the quality of the task [4]. The crowdsourcing platform is responsible for defining the rules governing the process of Human Intelligent Tasks (HIT) submission, task allocation, quality, and incentives. Furthermore, the platform defines the protocols for communication among the workers and in between the requester and worker, since certain platforms such as Innocentive [5], Topcoder [6] and open innovation [7] are encouraging the collaboration among workers.

It is reported that the number of tasks posted in crowdsourcing platforms has increased in the recent past [8], [9]. Hence, the workers should spend a significant amount of time for choosing the tasks that matches to them. This increases the pickup time and execution time of the tasks which affect the throughput of these platforms. Moreover, a wrong selection of tasks adversely affect the winning chance of the workers. Since crowdsourcing platforms are working in a competitive mode, success rate of workers has a significant role. Hence, improving the quality of work and output is challenging. Task recommendation helps in solving this problem by suggesting suitable tasks to the workers. Several task recommendation systems are proposed as [10], [11], [12], [13]. A task recommendation system based on workers success rate produce more precise recommendations than skill based approaches. We have developed a task recommendation model [4] in an earlier work which recommend tasks to workers based on their winning chances. Tasks are recommended by computing the participation and winning probability using the information such as win and participation history. However, it does not consider the case of new and inexpertise workers joining the platform.

New and inexpert workers represent a major chunk of the worker pool. In a recent analysis [8], regarding the participation of workers in tasks, it is observed that the participation rate and activeness of new workers are higher than that of expert workers. Hence it is desirable, to attract new workers to the platform, with a support system, such as task recommendation. Recommendation systems based on skills alone cannot ensure the participation of workers. Rather, we need to consider the worker's interest in participating on bids for recommending tasks and ensuring the quality of the system. Hence, the system for recommending tasks to inexpert workers consist of two subsystems. The first is based on a skill taxonomy and partcipation probabilty of expert workers. The latter is based on skill modelling of tasks and workers.

The rest of this paper is structured as follows: Section II describes our task personalization models in detail. In section III we compare the proposed model with some baseline methods using simulations. Some recent works in task recommendation are detailed in section IV. We conclude in Section V.

II. Task recommendation models

Recommending relevant tasks to new and inexpertise worker is a difficult task due to the unavailability of previous his-

978-1-5386-5314-2/18/$31.00 ©2018 IEEE

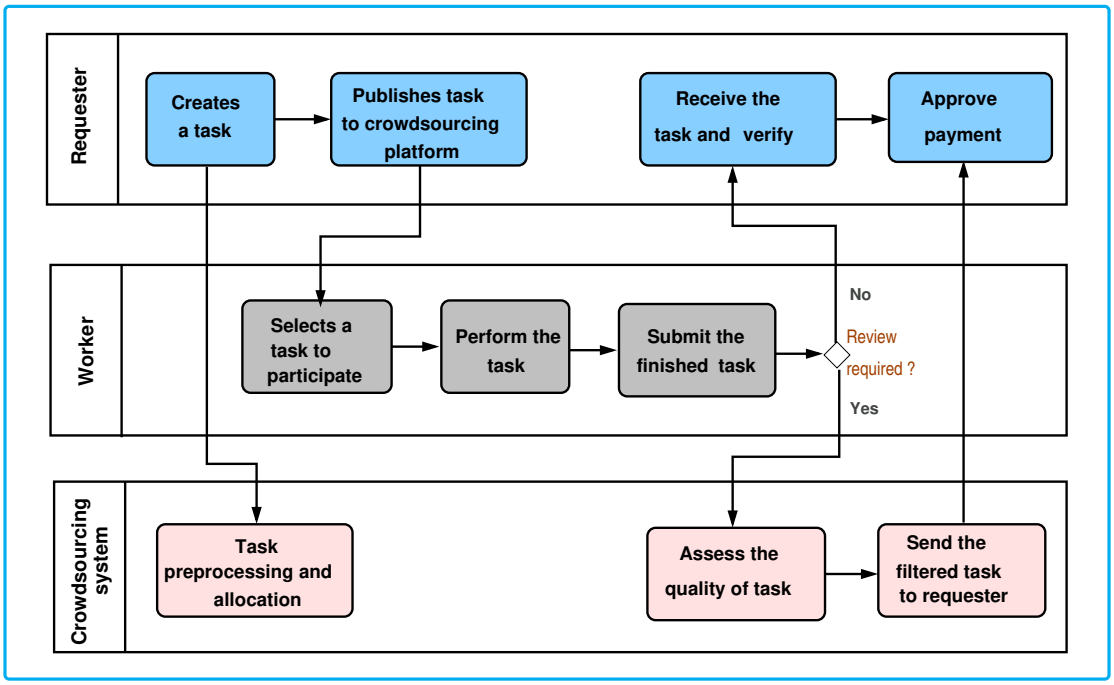


Fig. 1: Conceptual illustration of work flow in a crowdsourcing platform.

tory. Though, for inexpert workers, the participation history is available, there is no relevance in recommending tasks in which they have recently participated since chances of success are less. The objective of our model is to improve the participation of workers and their success rate, and thus reducing the pickup time of tasks. As noted above, since only skill profile is available for new and inexpert workers, we propose two models of task recommendation, first is an *Expert matching* model and the latter is a *Task matching* model.

A. Constructing a skill heirarchy

For finding the similarity between the skills of workers and tasks, we map the skills using skill taxonomy. Skill taxonomy is represented in the form of a graph, in which each node represents a skill and its children map the specialized skill related to its parent. In our model we assume that the skill taxonomy is available in the crowdsourcing platform. An instance of the skill taxonomy we used in our experiments is given in Fig. 2. We use the taxonomy of software skills for our simulations.

Let W be the set of available workers and T be the set of available tasks. Let $S = (s_1, s_2, ...s_n, is-a, \leq)$ represents the tree of skill taxonomy, where each node s_i represents the elementary skills for a worker W where $i \in W$ and s_j represent skills of task T such that $j \in T$. For example in the given skill taxonomy of Fig. 2, let $s_1 = Java$ and $s_2 = JavaSE8$, then $s_1 \leq s_2$. That is $\leq$ relation implies that any worker with s_2 skill is able to perform a task which requires skill s_1. Also any worker with s_2 skill is able to perform a task that a worker with skill s_1 can do.

Here we present two algorithms for task recommendation. In *Expert matching* model, we match the skills of inexpert workers and expert workers by mapping to the skill taxonomy. Then using the participation and winning probability of the expert workers, we recommend tasks to the inexpert workers. The second algorithm is *Task matching model* in which we compute the similarity between the set of skills of workers and tasks using heirarchical approaches.

B. Expert matching model

Let W_{ne} be the set of inexpert workers and W_e be the set of expert workers. A new worker W_{ne} should map to another participant W_e who has exact or very similar skills. At first, we compute the distance between the skill set of an inexpert worker and the available expert workers. To find the similarity between the inexpert worker and the available participants *Resinik* similarity method is used [14]. *Resinik* similarity is an information theoretic approach, used for computing the similarity between concepts or words by considering their relative frequencies. Let Dep_{max} represents the maximum depth of the skill taxonomy S and $lca(s_1, s_2) \in S$ is the lowest common ancestor of s_1 and s_2. Therefore the distance between the skills is computed using the following equation.

$$dist(s_1, s_2) = \frac{Dep_{max} - (Dep(lca(s_1, s_2))}{Dep_{max}}. \quad (1)$$

For e.g., consider a worker W_1 who knows $JavaSE8$ and another worker W_2 who knows $Java$ in our skill taxonomy as given in Fig. 2. The maximum depth Dep_{max} of the given taxonomy is 3. Therefore, the $dist(W_1, W_2)$ is the distance

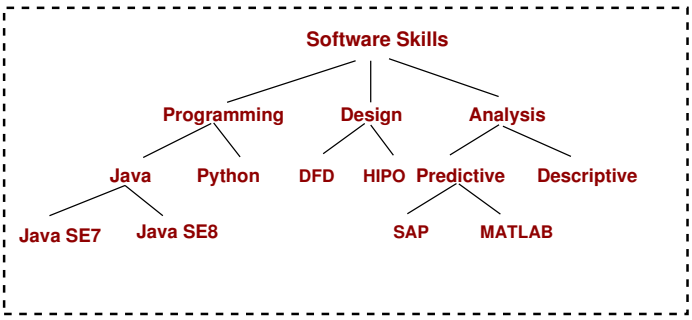


Fig. 2: An instance of taxonomy of software skills.

between their skills.
$dist(JavaSE8, Java) = \frac{3-2}{3} = \frac{1}{3}$.

Similarly, consider another worker W_3 whose skill is in $Analysis$, then distance between the workers W_1 and W_3, $dist(W_1, W_2)$, is computed as
$dist(JavaSE8, Analysis) = \frac{3-0}{3} = 1$.

The distance is inversely proportional to the similarity in skill level of a new worker to that of an existing worker, in the heirarchy. Here $JavaSE8, Java$ are closer in the taxonomy since the distance between them is only one edge. However, $JavaSE8$ and $Analysis$ are entirely different skill sets and hence shows a maximum distance of 1. The distance, $Dist(W_{ne}, W_e)$, between two workers W_{ne} and W_e is formulated as

$$= \begin{cases} 0, if\ \exists\ s \in s(W_e) \mid\ \ s \geq s(W_{ne}), \\ min\_dist(s(W_{ne}), s),\ \ otherwise. \end{cases} \quad (2)$$

Note that the minimum distance is computed for $s \in s(W_e)$. To find the similarity, we compute the sum of the distance between each pair of skills of the workers and that cumulative dsitance δ is stated as

$$\delta(Al) = \sum_{i \in W} dist(W_{ne}, W_e). \quad (3)$$

where Al denotes the alikeness of the workers.

From the above definitions, we infer that when the participants are closer to each other, the distance is smaller and gives more similarity. Therefore the value of $\delta(Al)$ is minimum for workers with similar skill sets. We compute the probabilty for those workers with mininum $\delta(Al)$ for recommending the new workers. The skills are represented as words in the skill taxonomy for encoding and each word represents a path in the taxonomy. If the W_{ne} has specialized skills he could get good quality recommendations. The process of matching a new worker to an expert worker is given in Algorithm.1. At first, the new worker list is reverse ordered, for appearing the most specific skills of each branch first. In the next step, we sort the list of expert workers according to their skills. Then, we match the new workers to expert workers with lowest number of skills, so that the recommended tasks is more relevant. For that the distance between the skills of expert and inexpert workers is computed in next line. Next, we match the inexpert worker with expert worker of minimum distance. Then, we compute the participation probability of the expert worker in each tasks, and those tasks with higher probability are recommended to the inexpert worker.

Algorithm 1: Algorithm for matching inexpert workers to expert workers and recommending tasks.

input : Set of expert workers W_e, Set of inexpert workers W_{ne}, skill(), set of tasks T
output: Match$(W_e) \leftarrow W_{ne}$, a list of tasks L

reverse sort W_{ne} according to skills alphabetically;
sort W_e according to $(skill(i) | i \in W_e)$;

/* Matching with the mininum distance */

for *each distance* $k = 0$ *to* d_{max} **do**
  for *each* $i \in W_e$ **do**
    /*do from lesser number of skills*/
    if $d(skill(i), skill(j)) \leq k$ **then**
      $Match(i) \leftarrow j$;
      Compute the participation probability of W_j;
      /* compute L based on the participation probability of W_j */
      Recommend L;
      delete j from W_{ne};
    end
  end
end

For computing the winning and participating probability of expert workers, we use the methods explained in our previous work [4].

$$W\left[Z_{ij} = 1\right] = \frac{1}{1 + exp(-B_{ij})} \quad (4)$$

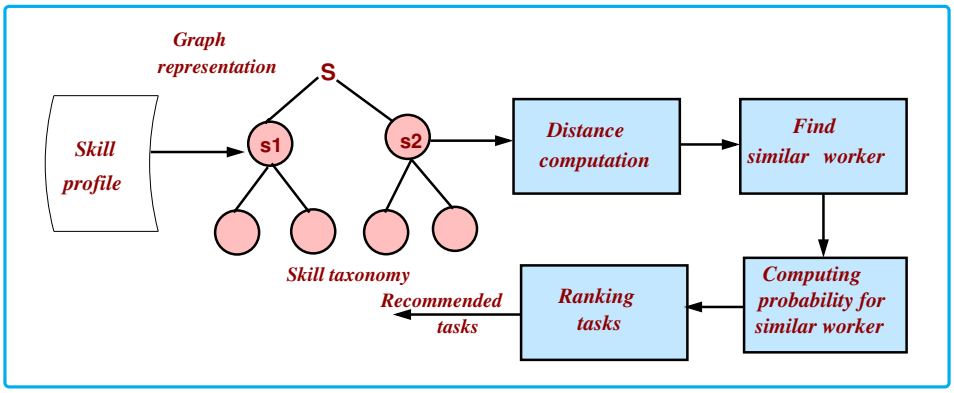


Fig. 3: Process of recommending tasks to a new worker

where B_{ij} is the willingness of a worker i to choose a task j.

C. Task matching model

In this model we compute the matching between nonexpert workers and tasks using heirarchical mapping. Here, the skills of nonexpert workers and tasks are mapped to the taxonomy. Then we compute the similarity between the workers and tasks using the similarity measures. The distance, $Dist(W_{ne}, T_j)$, between two workers W_{ne} and T_j is computed as

$$= \begin{cases} 0, \ if \ \exists s \in s(T_j) \mid \ s \geq s(W_{ne}). \\ min\_dist(s(W_{ne}), s), \ otherwise. \end{cases} \tag{5}$$

The minimum distance, min_dist, is computed for $s \in s(T_j)$ and the cumulative dsitance δ is stated as

$$\delta(Al) = \sum_{i \in W_{ne}} dist(W_{ne}, T_j). \tag{6}$$

Algorithm 2 describes the task matching process. Tasks with higher similarity in skills are recommended to the worker. The task skills are reverse sorted in alphabetical order, thus the specific skills in each branch of the skill taxonomy appear first. The nonexpert workers is also reverse sorted according to the skills, so that the workers with specific skills appears first in the taxonomy. Subsequently, for each distance, 0 to $dist_{max}$, and for each skill we find the list of sorted tasks and recommend to the participant at this distance. The process is continued by increasing the distance until all partcipants are recommeneded with a list of tasks. The advantages of such a taxonomy mapping is that it reduces the complexity in computing the similarity, compared to matrix approaches. Also this helps in improving the quality of recommendations.

Algorithm 2: Algorithm for skill matching of tasks and workers.

input : W_{ne}, set of tasks T, skill()
output: Match$(T_j) \leftarrow W_{ne}$, L

reverse sort W_{ne} in alphabetical order of skills;
sort T according to $(skill(j)|j \in T)$;

for *each distance* $k = 0$ *to* d_{max} **do**
  for *each* $j \in T$ **do**
    /*do from lesser number of skills*/
    if $d(skill(i), skill(j)) \leq k$ **then**
      /* for minimum distance */
      $Match(i) \leftarrow j$;
      Recommend T_j to W_{ne}
      Update L
      delete i from W_{ne};
    end
  end
end

III. Experimentation

To assess the effectiveness of the proposed task recommendation method we conduct simulations using both synthetic and real world dataset. The simulation framework is developed in Python environment. For creating the samples, we use Map Reduce library and random library. The matrices are generated using Scipy library [15], [16] and the mathematical expressions are optimized with the help of Theano library [17].

A. Dataset

We have collected dataset from CrowdFlower [18], a well known crowdsourcing platform. CrowdFlower dataset is publicly available from which the observations from 2014 onwards are collected and a set of 423 workers is randomly choosen for the analysis along with 5613 jobs. From this, we summarized our required data that corresponds to the features of the matrices used. The details of the dataset are depicted in Table I.

Furthermore, to assess our recommendation method a synthetic dataset is used. The synthetic dataset contains data of 350 workers along with 4000 tasks. For each worker and

TABLE I: Preliminary Statistics of dataset.

Parameters	Dataset summary
Number of Workers(I)	423
Number of Tasks(J)	5613
Worker feature's dimension(N)	2560
Task feature's dimension(M)	6321
Number of elements in Worker-Reward matrix	5613
Number of elements in Worker-Task matrix	17, 109
Average of the number of tasks each worker participated	23.85
Average number of workers in each task	2.90

task, the profile and features are created. A skill taxonomy is generated for wokers and tasks. The generated taxonomy is of maximum depth 5 and each node contains 5 children. For qualitative analysis we choose a small amount of workers and tasks. However, in order to measure the time needed to perform the recommendation, we consider higher number of workers and tasks.

For demonstrating the performance of the recommendation model, each data set is divided into two parts, 20% of all data are selected for testing, and the remaining as training set. From this details we extract the binary values corresponding to feature vectors of both tasks and workers. To balance the values, we use *one-of-N* coding [19].

B. Benchmarks

To assess our recommendation algorithms we used two baselines, namely *Skill-matching* and *Random-matching* [20]. In the first benchmark we consider recommending tasks by finding similarity between the skills of tasks and worker. In this case we do not consider the heirarchical structure. In latter we randomly select a list of tasks for recommending to nonexpert workers.

C. Evaluation metric

Mean Percentile Ranking (MPR) is used as the first evaluation metric, since we have to take in account competition among workers, participation and repeat feedback. MPR is a recall-based evaluation metric which evaluates a worker's preferences over a list of recommended tasks [21]. MPR is defined as

$$\frac{\sum_{ij} C_{ij} P_{ij}}{\sum_{ij} C_{ij}} \tag{7}$$

where p_{ij} is the percentile ranking of the task j for worker i. In our model recommendation is based on worker-reward probability matrix which is estimated from worker-participation probability matrix, $P = RZ$, where R is the worker-reward matrix and Z is the worker-task matrix. As a second evaluation method PR curve (Precision Recall curve) is used. This method is used for evaluating the number of tasks in the test set, for which the probability can be correctly anticipated by taking top n percentage from the recommended tasks. Value of n is varied accordingly to obtain the PR curve.

D. Results

We estimate the performance of our models in comparison with the baseline methods. The experiments are conducted by varying number of workers for measuring the recommending time, estimating the accuracy and precision. The results are presented here, by taking average of four simulation runs. We observe that our proposed models outperforms baseline models, in terms of MPR and PR.

Fig. 4a shows the evaluation results of all the algorithms using PR mathod. The algorithms are evaluated for varying number of participants. *Expert matching* and *Task matching* gives better results than other baseline approaches.

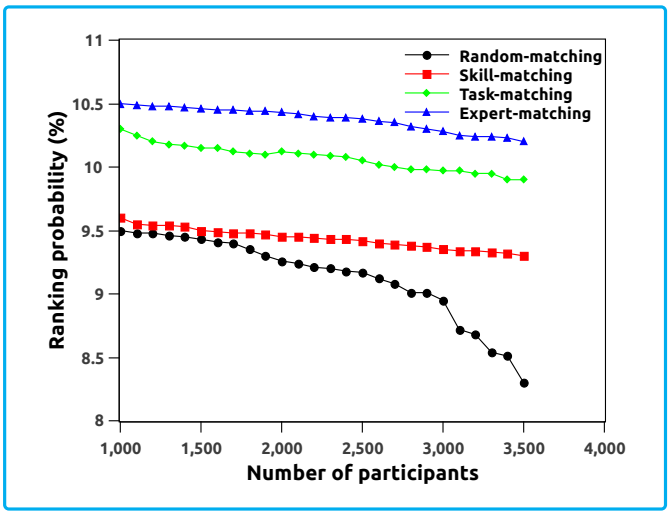
Fig. 4b measures the time (in ms) required for each algorithm for recommending tasks for all the participants. We use the same skill taxonomy characteristics with $depth_{max}$ = 5 and also we keep the same number of workers as before. The *Expert matching* is the fastest and then *Task matching* also gives considerably good results than of *Random-matching*.

Table II shows the MPR results obtained for all the four algorithms. The lower values of MPR indicate higher performance of the model. Note that *Expert-matching* gives the better results. *Task matching* is also gives comparatively better results than that of the baselines. The smaller number of workers, tasks and the extensive depth of the skill taxonomy, cause sparsity in the worker skills and that makes it difficult to obtain good matches. We notice that having more skillful workers improve the quality of the recommendation and decrease the cumulative distance.

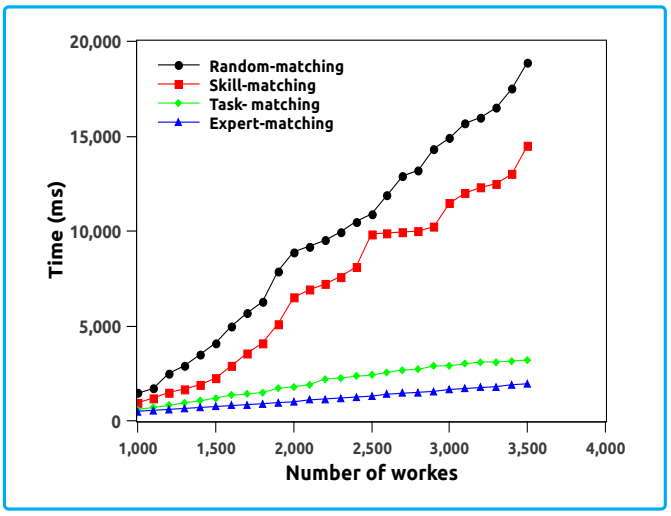
IV. Related work

Kathrin Borchert et.al [22] studied the influences of recommender systems on the success rate of crowdsourcing platforms and earnings of the workers. They proposed a model for evaluating the effect of recommender systems for task recommendation in various crowdsourcing platforms and observed that even a simple recommendation systems leads to improvement in crowdsourcing platforms.

Estimating the chances of a worker getting rewarded is tedious because of the two significant features of crowd sourcing platforms. Only one or few workers are rewarded for a particular job, and for the tasks of current interest no reward or winner is decided. This is similar to the cold-start problem in recommender systems where information is very sparse to calculate affinity between workers or tasks, which degradates the performance prediction [23]. To address this data sparsity problem in task recommendation the reserved information including features of workers and tasks, participation histories can be deployed [19]. They come up with a good recommendation model, based on the participation information. But the proposed method is restricted to contest



(a) Using PR curve for recommending the tasks to a new worker



(b) Recommendation time

Fig. 4: Comparison of proposed methods with baseline algorithms

TABLE II: MPR results of the proposed system

Algorithm	MPR
Random-matching	12.406
Expert-matching	**7.98**
Skill-matching	9.01
Task-matching	11.03

style tasks. Further, it is suitable for recommending tasks only to the expert workers.

In incentive based crowdsourcing systems, the main motivation of the workers are incentives and performance. Hence a good recommendation system should recommend tasks in such a way that their success rate and participation rate increases. In our previous work [4], we study the task recommendation in crowdsourcing systems and a method is implemented based on workers partcipation history and win history. It recommend tasks based on the participation probability and winning probability of workers, hence improve the chances of success. But such a system should be able to recommend tasks to expert workers or those has a previous experience. Hence we improve our model by addressing the problem of recommending tasks to inexpertise or new workers.

V. Conclusion

In this paper, we have proposed a task recommendation model for crowdsourcing systems which recommend tasks for inexpert and new workers. Two algorithms are devised for task recommendation. The similarity between the inexpert and expert workers, are computed using heirarchical skills and Resnik similarity measures. We observed that the proposed models are efficient in terms of computing time for recommendation lists and task pickup time. The simulation results confirmed that this computation yields a list for recommending tasks to new and inexpert workers. We have experimented these algorithms using only single skill of workers and tasks. Extending for multiple skills could be a possible future work.

References

[1] A. Ghezzi, D. Gabelloni, A. Martini, and A. Natalicchio, "Crowdsourcing: a review and suggestions for future research," *International Journal of Management Reviews*, vol. 20, no. 2, pp. 343–363, 2018.

[2] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information science*, vol. 38, no. 2, pp. 189–200, 2012.

[3] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh, "Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, p. 7, 2018.

[4] A. R. Kurup and G. P. Sajeev, "Task recommendation in Reward-Based crowdsourcing systems," in *Fifth International Symposium on Women in Computing and Informatics (WCI-2017)*, Manipal, Mangalore, India, Sep. 2017, pp. 1511–1518.

[5] K. R. Lakhani and E. Lonstein, "Innocentive.com (c)," 2011/8/17 2011. [Online]. Available: https://hbr.org/product/innocentive-com-c/612027.

[6] K. Li, J. Xiao, Y. Wang, and Q. Wang, "Analysis of the key factors for software quality in crowdsourcing development: An empirical study on topcoder. com," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*. IEEE, 2013, pp. 812–817.

[7] K. L. Guth and D. C. Brabham, "Finding the diamond in the rough: Exploring communication and platform in crowdsourcing performance," *Communication Monographs*, vol. 84, no. 4, pp. 510–533, 2017.

[8] A. Jain, A. D. Sarma, A. Parameswaran, and J. Widom, "Understanding workers, developing effective tasks, and enhancing marketplace dynamics: A study of a large crowdsourcing marketplace," *Proc. VLDB Endow.*, vol. 10, no. 7, pp. 829–840, Mar. 2017. [Online]. Available: https://doi.org/10.14778/3067421.3067431

[9] V. Chandrasekaran, S. V. Rajan, R. K. Vasani, A. Menon, P. B. Sivakumar, and C. S. Velayutham, "A crowdsourcing-based platform for better governance," in *Proceedings of the International Conference on Soft Computing Systems*. Springer, 2016, pp. 519–527.

[10] M.-C. Yuen, I. King, and K.-S. Leung, "Taskrec: A task recommendation framework in crowdsourcing systems," *Neural Processing Letters*, vol. 41, no. 2, pp. 223–238, 2015.

[11] V. Ambati, S. Vogel, and J. G. Carbonell, "Towards task recommendation in micro-task markets." *Human computation*, vol. 11, p. 11, 2011.
[12] C. H. Lin, E. Kamar, and E. Horvitz, "Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing." in *AAAI*, 2014, pp. 908–915.
[13] G. Sajeev and P. Ramya, "Effective web personalization system based on time and semantic relatedness," in *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*. IEEE, 2016, pp. 1390–1396.
[14] P. Resnik *et al.*, "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language," *J. Artif. Intell. Res.(JAIR)*, vol. 11, pp. 95–130, 1999.
[15] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam [online]. Available: http://www.python.org/doc/current/ref/ ref.html., 1995.
[16] E. Jones, T. Oliphant, and P. Peterson, "others. scipy: Open source scientific tools for python. 2001," *[Online]. Available: http://www. scipy. org*, 2016.
[17] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, pp. 55–79, 2016.
[18] "Crowdflower, https://crowdflower.com/," Accessed: 2017-07-05.
[19] Y. Baba, K. Kinoshita, and H. Kashima, "Participation recommendation system for crowdsourcing contests," *Expert Systems with Applications*, vol. 58, pp. 174–183, 2016.
[20] P. Mavridis, D. Gross-Amblard, and Z. Miklós, "Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 843–853.
[21] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 263–272.
[22] K. Borchert, M. Hirth, S. Schnitzer, and C. Rensing, "Impact of task recommendation systems in crowdsourcing platforms," in *FATREC Workshop on Responsible Recommendation Proceedings*, 2017, p. 10.
[23] V. Kavinkumar, R. R. Reddy, R. Balasubramanian, M. Sridhar, K. Sridharan, and D. Venkataraman, "A hybrid approach for recommendation system with added feedback component," in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*. IEEE, 2015, pp. 745–752.