Using Hierarchical Skills for Optimized Task Assignment in Knowledge-Intensive Crowdsourcing

Panagiotis Mavridis
Université de Rennes 1
IRISA
panagiotis.mavridis@irisa.fr

David Gross-Amblard
Université de Rennes 1
IRISA
david.gross-amblard@irisa.fr

Zoltán Miklós
Université de Rennes 1
IRISA
zoltan.miklos@irisa.fr

ABSTRACT

Besides the simple human intelligence tasks such as image labeling, crowdsourcing platforms propose more and more tasks that require very specific skills, especially in participative science projects. In this context, there is a need to reason about the required skills for a task and the set of available skills in the crowd, in order to increase the resulting quality. Most of the existing solutions rely on unstructured tags to model skills (vector of skills). In this paper we propose to finely model tasks and participants using a skill tree, that is a taxonomy of skills equipped with a similarity distance within skills. This model of skills enables to map participants to tasks in a way that exploits the natural hierarchy among the skills. We illustrate the effectiveness of our model and algorithms through extensive experimentation with synthetic and real data sets.

Keywords

skill modeling, crowdsourcing, task mapping

1. INTRODUCTION

Crowdsourcing platforms such as Crowdflower[1], Amazon Mechanical Turk[2] or FouleFactory[3] engage more than 500k participants [20] who perform simple microtasks on a daily basis. More knowledge-intensive tasks can be found on specialized platforms, such as Zooniverse[4], BumbleBeeWatch[5] or SPIPOLL[6]. In this later, benevolent participants upload and annotate insect images according to a precise taxonomy of species.

While one can obtain useful results through these platforms for a number of tasks (that would be otherwise difficult for computers), controlling the quality of the results is a challenging issue, due to the unreliability, volatility or lack of skills of participants. In particular, in knowledge-intensive crowdsourcing [19], where a specific expertise is needed to complete a task, the platform should ideally assign or propose the task to a participant who has this specific skill, or at least some experience in the domain. Generic crowdsourcing platforms already provide basic skill labeling (such as qualifications in Amazon Mechanical Turk[7]: these are short descriptions of qualifications for certain skills a participant might have or the requester might require). Similarly, academic research [4, 19, 24] is also considering skill models to improve result quality. These existing approaches rely on flat, unstructured skill models such as tags or keywords.

However applications often require at least some basic forms of reasoning about the skills (such as, for example, knowing that the skill *English writing* is "more specific" than *English reading*, in the sense that anyone who can write English can also read). Even such simple reasoning operations are not easy to realize with the above mentioned flat skill models. Many platforms could benefit from such a structured skill approach. On the one hand, it would allow a precise and better targeting of tasks. On the other hand, skill reasoning capacities, especially skill substitutions, would enable the participation of the full available workforce of the platform, even if skills do not correspond exactly to requirements. It is noteworthy that rich skill taxonomies are available and used in other contexts, such as ESCO[8], which is used to help European citizens in their job search and represents 5,000 skills in a structured way. Skill taxonomies are also recommended for companies using collaborative solutions, so that "employees have a common language to self-profile themselves" (the Wandinc company sells such taxonomies with more than 1,400 personal or business skills[9]).

In this paper we propose to finely model tasks and participants using a skill taxonomy. Our contributions are the following:

- we propose an effective taxonomy-based skill model for crowdsourcing, allowing to reason about skill substitutions;
- we define a distance measure between the skills of a participant and the skill required by a task that reflects how well they correspond to each other;

[1] http://www.crowdflower.com
[2] https://www.mturk.com
[3] http://www.foulefactory.com
[4] www.zooniverse.org
[5] http://www.bumblebeewatch.org
[6] Photographic monitoring of pollinator insects, http://www.spipoll.org/

[7] http://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/Concepts_QualificationsArticle.html
[8] ESCO: European Skills, Competences Qualifications and Occupations https://ec.europa.eu/esco/home.
[9] http://www.wandinc.com/wand-skills-taxonomy.aspx

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
DOI: http://dx.doi.org/10.1145/2872427.2883070.

- we formalize the optimization problem of task assignment to most suitable participants in participatory crowdsourcing;
- we propose several task assignment heuristics that perform better than methods based on unstructured skills;
- we demonstrate the effectiveness of our heuristics on synthetic and real data sets.

The rest of the paper is organized as follows. In Section 2 we give details about our skill model and formulate the task assignment problem in our terminology. In Section 3 we present our algorithms to address the task assignment problem. We report the results of our extensive experimental evaluation in Section 4. In Section 5 we present the related work and we conclude the paper in Section 6.

2. MODELING SKILLS FOR KNOWLEDGE-INTENSIVE CROWDSOURCING

2.1 Assumptions, skills, tasks and participants

Before formalizing our problem, we summarize the assumptions we made. We suppose that a skill taxonomy is available at the crowdsourcing platform that we can use to model required skills as well as participant expertise (Figure 1). We assume that the concepts present in this taxonomy are used (by the task requester) to annotate the tasks which are listed at the platform, with the required skills. We restrict our attention to tasks that only require a single skill. We believe that this assumption is realistic, as such tasks are more adapted for micro-task crowdsourcing. It is also reasonable to assume that in the launch of a crowdsourcing application there will be no knowledge of the participant profiles. This is known as the cold-start problem, it is inherent to social network applications such as crowdsourcing platforms and it is out of the scope of this study as there is related work that studies it (as mentioned in [19] and as studied in [9, 21, 25, 26]). However in this current paper we suppose that the given skills are correct and assessed. Several methods exist for this purpose, such as qualification tests in Amazon Mechanical Turk or ground truth questions like the ones presented in [10] and [19]. Another interesting approach is the endorsement of skills as the one proposed in LinkedIn[10]. Eventually the profiles of the participants can be corrected and updated by their participation on the platform and requester feedback (as in [10]). In Section 4.4 we use a simple method based on the above to obtain safer participant profiles using ground truth questions.

More formally, we model a skill taxonomy as a tree $S = (\{s_1, s_2, ...\}, \text{is-a}, \leq)$ whose nodes are elementary skills $\{s_1, s_2, ...\}$ and *is-a* is the parent-child relationship (subclass relationship). Relation $\leq$ denotes the partial order within skills. Taking as example the skill taxonomy of Figure 1, for $s = $ *core Java* and $s' = $ *Java 1.8 thread*, then $s \leq s'$. Informally, this partial order means that any participant with skill s' can perform a task requiring skill $s \leq s'$.

Let $depth(s) \in \mathbb{N}$ be the depth of skill s in the taxonomy S. We consider only taxonomies with at least two skills (hence $depth(S) > 0$). We use $T = \{t_1, t_2, ...\}$ and $P = \{p_1, p_2, ...\}$ to denote a set of tasks and participants, respectively. For a given task t, we denote the required skill specified by the task requester by $skill(t) \in S$. A skill profile of a participant is the set of skills she possesses. We denote the skill profile of a participant p by $skill(p) \subseteq S$. We insist that $skill(t)$ for a task t refers to a single skill, while a skill profile $skill(p)$ for a participant p might contain several skills.

[10] `http://www.linkedin.com`

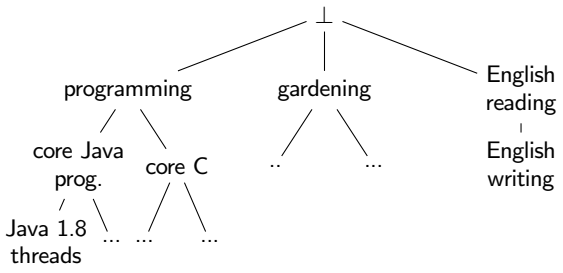

Figure 1: a skill taxonomy

In this work we suppose that the announced skills are correct (see Section 5 for existing skill estimation techniques). However we ensure the safety of participant profiles using these simple evaluation methods and calculate their profile. We do not study separately spammers for the sake of simplicity, as they can be ruled out by well-known crowd management techniques, such as majority voting or participant response-quality estimation based on a test set [22].

2.2 Task assignment

Given a set of tasks and participants, a task assignment $\mathcal{A}$ is a mapping from T to P that maps a task $t \in T$ to $\mathcal{A}(t) = p \in P$. A task assignment is partial (a task may not be assigned) and injective (a participant can only perform one task during this assignment). As a participant can only participate in one task at a time, the maximum number of tasks that can be assigned is $\min(|T|, |P|)$. Indeed, if there are less tasks than participants, some participants may not be assigned. We focus here on *covering task assignments*, where the available workforce is maximally assigned: the number of assigned tasks is $\min(|T|, |P|)$. More subtle models with partial assignments of tasks could be envisioned, where we would prefer not to invite some participants with very low expertise, but we leave these considerations for future work.

If the platform assigns a participant to a task, we assume that she accepts the assignment and completes the task to the best of her knowledge. Remark that, to obtain a more realistic model, we could add a probability of acceptance of a task, and a reliability of a participant for a given skill, but this does not lighten the skill mapping problem on its own. For the same reason we do not map the same task to several participants. However we can assume as in [1] that there can be several assignments rounds to simulate this feature of multiple assignment. This is more realistic and also used as a classic method for auction-based transactions in economics.

We next model the quality of an assignment. The best situation is to map a task with required skill s to a participant with this exact skill. Note also that a participant with a more specialized skill $s' \geq s$ can perform the task. If such skills are not available in the crowd, more generic participants can be used, but at the expense of a lower quality. In order to capture these situations, we consider a *skill distance* between the required skill and the available ones.

2.3 Assignment quality

Our definition is inspired by the classical Resnik similarity [18] that is generally used to measure the similarity between words

and concepts, while taking into account their relative frequencies. Our definition uses a simplified Resnik similarity for words with uniform distribution.

Let d_{max} be the maximum depth of the taxonomy S. Let $lca(s,s') \in S$ be the lower common ancestor of s and s' in the taxonomy. Then the (normalized) *skill distance* is given by

$$d(s,s') = \frac{d_{max} - depth(lca(s,s'))}{d_{max}}.$$

EXAMPLE 2.1. *According to Figure 1, consider a participant who is knowledgeable in* Java 1.8 threads *and* English reading. *The maximum depth d_{max} of our taxonomy S is 3. Hence*

$$d(\text{Java 1.8 threads,core Java prog.}) = \frac{3-2}{3} = 1/3,$$
$$d(\text{core Java prog.,programming}) = \frac{3-1}{3} = 2/3,$$
$$d(\text{Java 1.8 threads,English reading}) = \frac{3-0}{3} = 1.$$

It is noteworthy that this distance favors close skills that are deeper in the taxonomy. As an example, although Java 1.8 threads *and* programming *are both separated from* core Java *by one edge,* Java 1.8 threads *is considered closer. Moreover, unrelated skills such as* Java 1.8 threads *and* English reading *have only the root of the taxonomy as a common ancestor, and the distance is then 1 (the maximum).*

By extension, the distance $D(t,p)$ between a task t and a participant p is given by

$$D(t,p) = \begin{cases} 0 \text{ if } \exists s \in skill(p) \text{ s.t. } s \geq skill(t), \\ min_{s \in skill(p)} d(skill(t),s) \text{ otherwise.} \end{cases}$$

With these definitions, the distance is 0 if the participant has the required skill or she is more specialized. Otherwise, it depends on the distance between the task skill and the best available participant skill. Note however that d and D are not metric distances (no symmetry, no triangular inequality).

Finally, the quality of an assignment $\mathcal{A}$ is measured by the *cumulative distance* $\mathcal{D}(\mathcal{A})$, which is the sum of distances between each pair of assigned tasks and participants, i.e.:

$$\mathcal{D}(\mathcal{A}) = \sum_{(t,p) \text{ s.t. } \mathcal{A}(t)=p} D(t,p).$$

The *normalized cumulative distance* is $\mathcal{D}(\mathcal{A})$ divided by the total number of assigned participants. With this definition, the closer the participants are to the required skill of their task, the smaller is the distance and the better is the assignment.

We can now define the task assignment problem:

DEFINITION 2.2. (OPTIMAL COVERING TASK ASSIGNMENT PROBLEM)
`INPUT` *: a taxonomy S, a set of tasks T and participants P, skill functions.*
`OUTPUT`*: a covering task assignment $\mathcal{A}$ such that $\mathcal{D}(\mathcal{A})$ is minimized.*

In our model we want to assign a maximum number of available tasks. If there are more tasks than participants, hence only $|P|$ tasks can be performed during the assignment round. On the contrary, if there are less tasks than participants, some participants will not have any task to do on the assignment round. However based on [1] we can assume that there can be several rounds and waiting periods between them which is more realistic for real crowdsourcing and can fill in the gap of more participants or tasks at a given round.

In the following section (Section 3) we consider several heuristic algorithms for the task assignment problem.

3. TASK-ASSIGNMENT ALGORITHMS

The complexity of the task assignment problem we study is in P, as it can be reduced to a Minimum Weight Perfect Bipartite Graph Matching problem. There are several variants of the problem (e.g. where participants may take several tasks, or situation where one would like to simultaneously optimize the costs and the quality of the assignment [19]) which are NP-complete.

As a baseline, we use the Hungarian method [12], the combinatorial optimization algorithm often used to find perfect matchings, to obtain assignments with minimal normalized cumulative distance. We observed that for our specific problem, performance enhancements can be achieved. We considered two different heuristics: MATCHPARTICIPANTFIRST and PROFILEHASH.

Before we go on with the description of the above heuristics, please note that in our implementation, as a natural encoding we represent skills of tasks and participants as a set of words, such that each word denotes a path in the taxonomy S. For example, according to Figure 1, *core Java programming* is encoded by 00 and *English writing* by 20 (one digit per level).

In MATCHPARTICIPANTFIRST (Algorithm 1), we try to assign the most specialized tasks first to the participants with the lowest number of skills (hence saving most diverse participants for other tasks). More precisely, we reverse-sort the task skills alphabetically, hence the most specific skills of each branch of the taxonomy appear first. For instance the skill 01243 will appear before 0124. We also sort participants according to their number of skills, so that the least diverse participants appear first. Then, for each distance, starting from 0 to d_{max}, and for each sorted task skill, we scan the list of sorted participants and assign the task to the first available participant at this distance. We go on with increasing distances until there is no task or participant left.

The next heuristic, PROFILEHASH (Algorithm 2), uses indexes (hashmap) to organize participants' skills. It implements the following heuristic:

- Try to assign the most specialized tasks first (those that are more difficult to assign).
- For each task, search first for participants with the exact required skill (hence the quality is perfect without wasting more specialized participants).
- If no such participant is available, search for participants with more specialized skills, starting with the least specialized (again, quality will be perfect, and we attempt to save the even more specialized participants).
- If no such participant is available, we progressively relax the skill required for the task (the quality will decrease, but we try to minimize this loss by using the most specialized participants first).

The search order of this heuristic is depicted in Figure 2. In order to avoid a systematic traversal of the taxonomy tree, we speed-up the skill search by indexing each participant skills. More precisely, we build hashmaps that associate a skill to a list of participants with this skill. We consider a hashmap for each different skill depth. Also, in order to ease the search of prefixes

Algorithm	Time $O(.)$	Space $O(.)$
HUNGARIAN METHOD	n^3	n^2
MATCHPARTICIPANTFIRST	$d_{max}.m_s.n^2$	$m_s.n$
PROFILEHASH	$d_{max}.m_s.n$	$d_{max}.m_s.n$

Table 1: Complexity assuming that $|P| = |T| = n$, m_s is the maximum number of skills of a participant and d_{max} the maximum depth of the skill taxonomy.

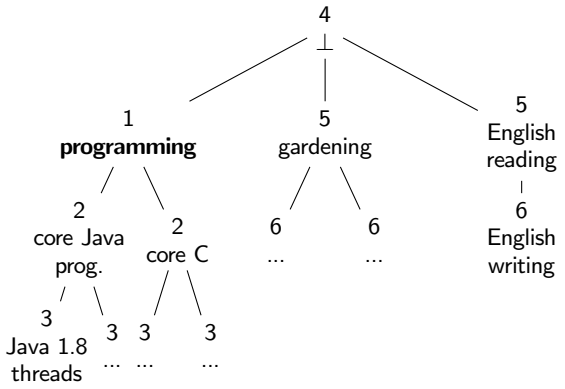

Figure 2: Search order, trying to assign a *programming* task (no specified order within skills with the same number on this picture).

or extensions of a given target skill, and trading memory for speed, we index also all *prefixes* of a participant skill. We insert and consume the prefix of each skill in a $FIFO$ order, so that we favor the more specialized skills first.

The time complexity of MATCHPARTICIPANTFIRST is composed by the sorting of T and P ($O(|T|\ln|T|+|P|\ln|P|)$), and the scan of participants and distances for each task ($O(d_{max}.m_s.|P|.|T|)$), where d_{max} is the maximum distance (depth) of the skill taxonomy and m_s is the maximum number of skills a participant has. Note also that our algorithms require distance computations, that impose lower common ancestor (lca) computations in the taxonomy. Constant time algorithms exists [3] (we rely on a simpler, linear time implementation in $O(d_{max})$ in our experiments). Space complexity is limited to input data storage ($O(|T|+m_s|P|)$).

For PROFILEHASH, beside task sorting (time $O(|T|\ln|T|)$), we have to pay for the indexing of all participant skills plus their prefixes. This latter costs a constant time for a hashmap. We also have to include the search for each task skill and for each of its prefix if necessary. This yields a $O(|T|\ln|T|+d_{max}(|P|m_s)+|T|m_s|P|))$ time complexity and $d_{max}m_s|P|$ space complexity for hashmap. We summarize the worst-case complexity of our algorithms in Table 1, assuming that $|T|$ and $|P|$ are of the same order of magnitude n for readability.

One can observe that the data required to handle our problem is likely to fit in main memory. Indeed, let us consider the realistic Amazon Mechanical Turk setting with 500,000 participants and 250,000 tasks. If we suppose one skill per task and a maximum of ten skills per participants and we assume that a skill can be encoded as a ten bytes word (a path in a $d_{max}=10$ taxonomy), the amount of memory to handle this information is 50 MB for participants and 2.5 MB for tasks, for a total of 53 MB. According to Table 1, the maximal

Data: Participants P, tasks T, *skill*() functions
Result: Assignment $\mathcal{A}$
reverse sort T according to task skills, alphabetically ;
sort P (w.r.t. $|skill(p)|$,$p \in P$) ;
foreach *distance* $i=0$ *to* d_{max} **do**
  foreach *task* $t \in T$ **do**
    foreach *participant* $p \in P$, *hence starting with the one with less skills* **do**
      if $d(skill(p),skill(t)) \leq i$ **then**
        $\mathcal{A}(p) \leftarrow t$;
        remove t from T;
        remove p from P;
      end
    end
  end
end

Algorithm 1: MATCHPARTICIPANTFIRST

Data: Participants P, tasks T, *skill*() functions
Result: Assignment $\mathcal{A}$
Initialize $d_{max}+1$ hashmaps, $M[0],...,M[d_{max}]$;
reverse sort T by skill depth ;
foreach *distance* $i=0$ *to* d_{max} **do**
  foreach *participant* p **do**
    foreach *skill* s *of* p **do**
      /* take s except the last i levels */
      $s' \leftarrow s[0...length(s)-i]$;
      /* append at
         the end of the correct skill list */
      $M[|length(s')][s'].append(p)$;
    end
  end
end
foreach *distance* $i=0$ *to* d_{max} **do**
  foreach *task* $t \in T$ **do**
    /* take $skill(t)$ except the last i levels */
    $s \leftarrow skill(t)[0...length(skill(t))-i]$;
    $C \leftarrow M[length(s)][s]$;
    /* take the first available
       participant in C, while respecting
       the skill specialization order */
    $p \leftarrow C.first()$;
    $\mathcal{A}(t) \leftarrow p$;
    remove t from T;
    remove p from P;
  end
end

Algorithm 2: PROFILEHASH algorithm

space required for running our algorithms for $d_{max} = 10$, $n=500{,}000$, $m_s=10$ and 10 bytes to store a skill is then around ($10.d_{max}.m_s.n=$)500 MB. A typical computer as used in our experiments can handle around thirty two times this amount.

Note also that the transactional aspect of the problem is negligible: participant and task skills can be stored securely in a database and updated, while the task assignment is performed. The discrepancy between the stored version and the in-memory version will not harm the process, as participant/task skill updates and taxonomy changes are not so frequent.

4. EXPERIMENTAL EVALUATION

4.1 Overall experimental setting

The evaluation was performed on an Apple Macbook Pro featuring an Intel i7 quad core CPU running at 2.8 GHz, 16GB of RAM (running on 1600MHz DDR3), 1TB SSD disk and the Mac OS X Yosemite operating system. The code was written in Java and compiled with the latest Java 8 Oracle's compiler. In order to assess our model, we used both a synthetic and a real data set of participants and tasks that we will explain in separate sections.

4.2 Synthetic data setting

Our first assessment of the model was carried out with the generation of a synthetic data set that consists of a taxonomy, a set of participants and a set of tasks. The taxonomy S that we used on most cases, unless stated differently, was a taxonomy tree of depth ten with ten children per node. Domain specific taxonomies with $d_{max} = 5$ or more, like the one in SPIPOLL, can be incorporated to more generic taxonomies to create a deeper and greater taxonomy of $d_{max} = 10$. Also, for simplicity the taxonomy we used for the Synthetic Data was a balanced taxonomy. However this is not a limitation because our distance definition (see Section 2) will favor more specialized skills in case they exist. Please note that in the real setting (see Section 4.4) the taxonomy was not balanced.

For the synthetic experiment, we created task skills and participant skills with respect to our taxonomy. Each task is associated with only one skill and is generated as a random path on the taxonomy tree.

Unlike a task, a participant might have multiple skills. To create the participant skills we propose a *budget method*. This method assumes that a participant can have several skills (nodes) on the taxonomy and distributes them randomly, but according to a budget. This is reasonable if we assume that to become an expert in a skill a participant should pass some time on the given domain, and that this occupation makes the participant less available to learn another skill in another domain. This method can eventually create an expert profile or a general knowledge profile. A combination of both in a given domain is also possible.

More precisely we carried out the experiments as follows. First, we generated a number of tasks with the above-mentioned method. Then we generated an equal number of participants, using the budget method. Afterwards, we used the generated tasks and participants as input to both our baseline algorithms and our heuristic propositions. For our experiments, we repeated this procedure ten times so that each point in each figure represents the aggregated result of ten repetitions of random data with the same characteristics (taxonomy, participants, tasks). The variance is also calculated and shown in our figures.

As a baseline we compare with the following algorithms:

- RANDOM and
- EXACTTHENRANDOM.

The RANDOM algorithm assigns randomly tasks to participants. More precisely it shuffles the tasks' ids and the participants' ids and then makes a complete one to one assignment between them. The EXACTTHENRANDOM algorithm matches first each task skill with a participant that has the *exact same skill*, and assigns the remaining tasks and participants randomly. This algorithm recognizes skills, but does not take advantage of the taxonomy structure. Instead, it can be interpreted as a keyword-based (vector of skill) matching of the participant's skills with the tasks.

The figures that follow show how the results we obtained with our different algorithms are affected by the different configurations of the taxonomy, the participants and the tasks. They also demonstrate how in terms of quality our algorithms outperform both the RANDOM and EXACTTHENRANDOM algorithms.

4.3 Synthetic data results and discussion

To assess our approach we propose *six* different figures, supporting its scalability and improved quality, time cost and effectiveness. We simulate with one curve on each figure the result of each of the five considered algorithms (two baselines, two heuristics and the optimal HUNGARIANMATCH algorithm[11]) that were used:

- RANDOM,
- EXACTTHENRANDOM,
- MATCHPARTICIPANTFIRST,
- PROFILEHASH,
- HUNGARIANMATCH.

On each of the following figures we assume that the number of participants and tasks are equal which does not harm the generality of our results. For qualitative measurements we chose a relatively small amount of participants and tasks (1,000 to 3,000). However, in order to show the scalability in terms of participants and tasks for the time needed to perform a complete assignment, we chose higher values of participants and tasks (10,000 to 500,000).

Figure 3 shows the normalized cumulative distance related to the assignment with respect to the number of participants. Each participant is created with twenty nodes of budget. We can easily distinguish how RANDOM is outperformed by all the algorithms. This can be interpreted as a motivation for the need of fine skill modeling. On the other hand the simple exact match assignment of EXACTTHENRANDOM is also outperformed by all our algorithms which strengthens the choice of an inference model for skills. Our PROFILEHASH algorithm perform as good as the exhaustive MATCHPARTICIPANTFIRST algorithm even though with minor differences. It goes without saying that the optimal HUNGARIANMATCH algorithm gives the lower bound of the cumulative distance that we can achieve. It is also noticeable that our heuristic algorithms perform very close to the optimal solution.

Figure 4 presents the time (in ms) needed for each algorithm to make a complete assignment of all the participants to all the tasks. We use the same skill budget for every simulated participant as before and the same taxonomy characteristics ($d_{max} = 10$, ten children per node). The x-axis is the number of participants (and tasks) that we keep the same as before. For qualitative reasons, we present the RANDOM, EXACTTHENRANDOM, MATCHPARTICIPANTFIRST and HUNGARIANMATCH algorithms along with the theoretical Hungarian match time for comparison of the time needed for a complete task to participant assignment. Not suprisingly the EXACTTHENRANDOM is the fastest and the MATCHPARTICIPANTFIRST is considerably faster than the HUNGARIANMATCH. We omit the presentation of the PROFILEHASH algorithm here because of its extreme performance which is the outcome of its efficiently indexed data structures.

In order to show how our algorithms can scale for more participants and tasks in terms of speed, keeping the same taxonomy

[11]Adapted from K. Stern's implementation, `https://github.com/KevinStern`

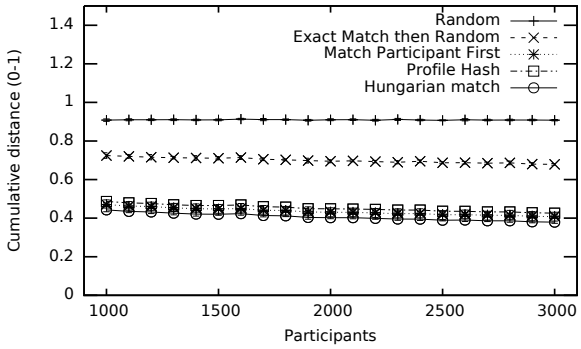

Figure 3: Normalized cumulative distance of assignment with respect to the number of participants

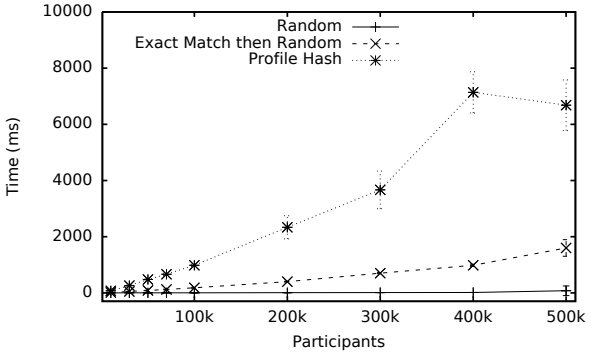

Figure 5: Assignment time for larger number of participants

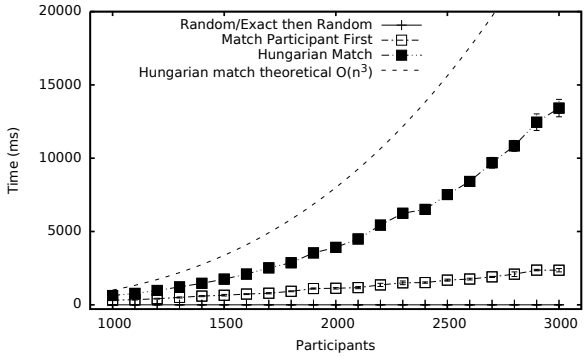

Figure 4: Assignment time with respect to the number of participants

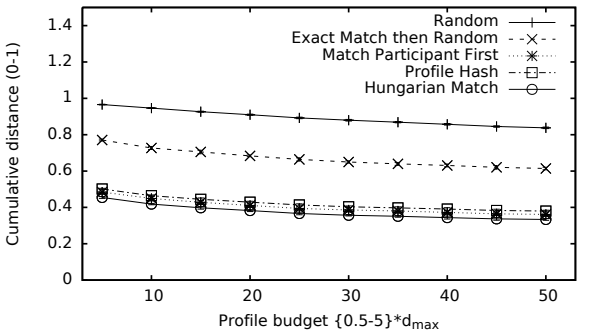

Figure 6: Normalized cumulative distance with respect to participant profile budget

characteristics and participant budget parameters, we simulate the time needed for a complete mapping of 10,000 to 500,000 participants and tasks shown in Figure 5. We compare in this figure only the ProfileHash with the baseline Random and ExactThenRandom algorithms. MatchParticipantFirst and the HungarianMatch are very slow and their computational complexities and time values show that they are not practical for such high participant and task numbers. We chose these values because currently Amazon Mechanical Turk, the oldest and most well known generic crowdsourcing platform, occupies about 500k participants and hosts about 274k tasks. In our setting we show that we can accommodate this size with even more tasks (500k instead of 274k). This indicates that both of our algorithms can perform (almost) real-time task mapping in such platforms. Of course Random or ExactThenRandom algorithms are faster but they lose a lot of potentially good participants (see Figures 3, 6, 7, 8). This would be an extreme waste of workforce that should not be missed.

Figure 6 shows how the budget on the skills of the participants affects the assignment in finding better mappings. We kept a fixed number of participants (3,000) the same taxonomy properties as before and we experimented with different participant profile budget. The x-axis is the budget for participant skill creation that varies from $0.5*depth$ to $5*depth$ (5 to 50 for our taxonomy) and the y-axis is the normalized cumulative distance (0-1). We can observe that the quality increased with the availability of more specialized skills. It is equally noticeable that our algorithms outperform the other algorithms in terms of quality at all budget values simulated. Moreover the faster ProfileHash perform slightly worse than MatchParticipantFirst. Again we can see that we are not very far from the optimal solution that the HungarianMatch provides.

In Figure 7 we show how the depth of the taxonomy affects the quality of the assignment. The x-axis represents the maximum depth of the taxonomy (d_{max}) from $d_{max}=2$ to 20 ($d_{max}=1$ is not as useful because it would assume falling to the vector-like model). The y-axis is the normalized cumulative distance. Because of the small number of participants and tasks and the great depth of the taxonomy we see that participant skills are quite sparse among the taxonomy and that makes it almost impossible to obtain good matches. Also, it is noteworthy that at about $d_{max}=8$ we have the maximum gap between ExactThenRandom and ProfileHash algorithms for these number of tasks, participants and current taxonomy setting (10 children per node). While d_{max} is below 4 we can see that both ExactThenRandom and our algorithms perform very well, practically performing all the possibly good assignments (cumulative distance 0). Again in the general case and for further d_{max} values our algorithms outperform both baseline algorithms. We can again notice that the faster ProfileHash perform again slightly worse than MatchParticipantFirst in this setting. Finally, we can also make another two observations based on this figure. On the one hand, MatchParticipantFirst, ProfileHash and HungarianMatch converge at a great depth which shows the importance of a diversity of skills in a given crowd. An increased budget-to-depth ratio which means having more skillful

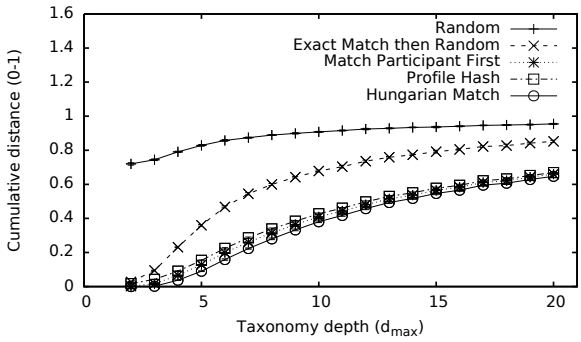

Figure 7: Normalized cumulative distance with respect to the d_{max} of the taxonomy

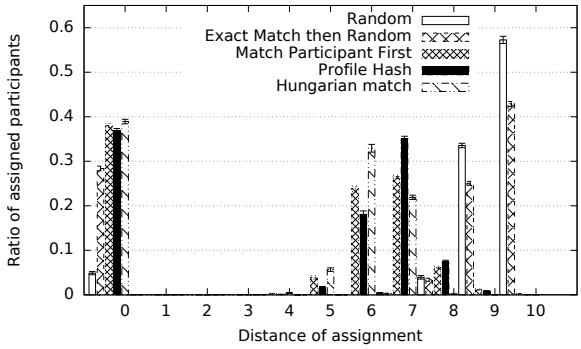

Figure 8: Ratio of assigned participants per distance

participants can improve the expected quality and thus decrease the cumulative distance measured. On the other hand, a more numerous crowd assigned to a greater number of tasks can also increase the probability of better task-to-participant matchings which will also lead to a decreased cumulative distance.

Figure 8 shows the distribution of assignment distances between algorithms. For this experiment, we keep a fixed number of participants and tasks to 3,000. The x-axis is the distance of quality with respect to the taxonomy while the y-axis is the ratio of tasks assigned per distance. We see that ProfileHash algorithm outperforms the other two random based algorithms because they make more assignments to lower distances. The fact that there is an interchange between MatchParticipantFirst and ProfileHash has to do with the fact that they are heuristics (do not provide the optimal solution) and that the former is an exhaustive algorithm. It is also noteworthy how the HungarianMatch would assign to the closest distances in order to give the optimal cumulative distance. We can also notice that due to the Resnik similarity (which is far from a shortest path distance) and the span of task and participant skills, our task and participant generation does not provide a lot of assignments of short distances (distances 1, 2, 3 and 4 for instance in Figure 8). To a greater extent, consider as an example a $d_{max}=10$ taxonomy, where a depth 4 ($length=5$) task skill will be matched with a depth 3 ($length=4$) participant skill (thus having a depth(lca(s,s')=3). Essentially, due to our distance definition it will obtain a $d(t,p)=10-3=7$ distance and not a $d'(t,p)=4-3=1$ that would be the value for the intuitive shortest path distance between performed task and participant skills.

As mentioned, we performed a series of extensive random generated data experiments to show the significance of our model and assignment algorithms. To the best of our knowledge we are the first to mention and implement a concrete model that uses fine skills based on a taxonomy. Anticipating the results of our experiments we can observe that in all cases the heuristics we provided give significantly better results than the Random or ExactThenRandom algorithms. We believe that the results speak loud themselves concerning the need of a finer model of skills for both tasks and participants in knowledge-intensive crowdsourcing. They also show that ProfileHash is an excellent candidate for a scalable, high quality mapping algorithm. In section 4.4 we will see how the above are supported with a real experimentation based on a real dataset and different crowds.

4.4 Real data setting

In order to obtain a more realistic data set and test our algorithms we followed a quiz procedure and recruited participants from different sources (our University laboratory and CrowdFlower). As a topic we chose computer science, a topic where we could elaborate a skill taxonomy, asked participants for their knowledge profile beforehand, assessed their profiles and then gave them the quiz to answer.

More precisely, we chose fifty eight multiple choice questions on computer science that we could easily elaborate a taxonomy of skills S. The taxonomy had a $d_{max}=4$ and a different number of children per node depending on the node and category which in our case was at least two. In this case the taxonomy was not symmetrical. Then we asked participants with different backgrounds and mostly computer programming to choose four of their preferred programming languages from a list. This first step provided us with participants with a maximum of four skills each (respecting the fixed profile budget used in Section 4.3).

The procedure continued with the real quiz that consisted of fifty eight multiple choice computer science questions mainly on programming languages, generic computer science, databases and computer architecture questions. All the questions were preannotated with respect to the taxonomy.

In order to carry out the questionnaire we recruited two different groups of participants. The first group consisted of 31 participants from the University of Rennes 1 that were either students on computer science or computer engineers or computer scientists. All these participants had a proven experience in computer science. The second group was a group of 176 participants that were recruited from CrowdFlower after six qualification questions (gold mining questions in CrowdFlower). In addition we also considered a third group made of the combination of these two groups (207 participants) which provides with a more diverse case of crowd.

Finally, before applying the algorithms we assessed the profiles given from the participants using test questions on the skills they submitted. Then we applied the two baseline algorithms (Random and ExactThenRandom) and our ProfileHash algorithm in order to obtain the results. Having the ground truth for all the questions and the answers from all the participants we could easily assess the quality of the task to participant assignment. After one hundred repetitions of the assignment algorithms we obtained results supporting our choice of model and methods.

4.5 Real data results and discussion

In order to show the feasibility of our approach we run two baseline mapping algorithms and one of our algorithms (Pro-

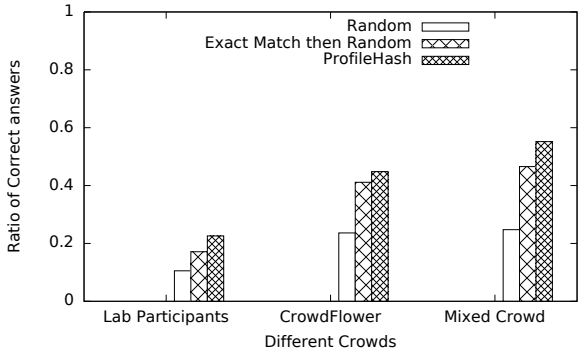

Figure 9: Ratio of correct answers with respect to different crowds and algorithm comparisons. Questions are annotated lower in the taxonomy.

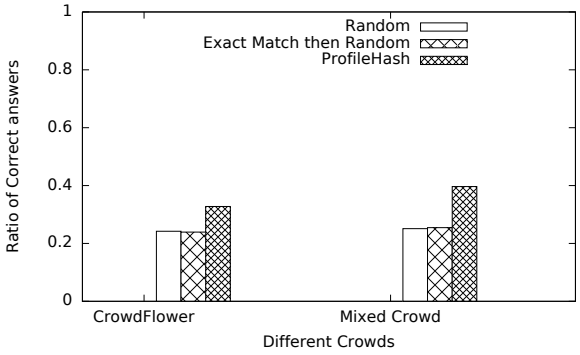

Figure 10: Ratio of correct answers with respect to different crowds and algorithm comparisons. Questions are annotated higher in the taxonomy.

FILEHASH) on the questions and the participants. Unless stated differently we also simulate the three different crowds mentioned below:

- Laboratory participants,
- Crowdflower participants,
- Mixed Crowd.

Every bar on the figures presents the average results of one hundred repetitions for the mapping algorithms applied and mentioned below:

- RANDOM,
- EXACTTHENRANDOM,
- PROFILEHASH.

The first crowd (Laboratory participants) consists of the thirty one participants familiar with Computer science. These participants including graduate students, researchers and engineers were recruited from our laboratory and agreed to participate voluntarily by replying to the survey we made in Google forms for them. The same survey was also given to the Crowdflower participants. The profiles of the participants were verified with test questions. One test question for each skill of the participant. Then the different algorithms were applied to perform the question-to-participant mappings (one to one). Finally with the help of the ground truth questions we could calculate the efficiency of the assignments. We discuss the results of these simulations below with the help of two figures.

Figure 9 shows the ratio of correct answered questions with respect to the three different crowds. We can observe how our method outperfoms the baselines in all the different cases and how the different crowds affect the percentage of correct answers. It also supports the intuition that with crowdsourcing the good use of a greater number of participants (from the laboratory participants towards the mix crowd) improves the result. In addition it shows that even a simple model of a vector of skills, such as the one simulated by EXACTTHENRANDOM can improve significantly the expected results compared to the RANDOM method.

Figure 10 shows again the ratio of correct answered questions with respect to the different crowds but this time the questions annotated two levels higher in the taxonomy. We performed this test to simulate the realistic case where a question could be annotated at a higher level. For instance imagine a Java question on sockets. This is clearly a networking task in Java and should be annotated lower in the taxonomy. It should be neither annotated as core Java nor as Object Oriented Programming. However such mistakes or lack of knowledge for a given task could occur in crowdsourcing platforms. With this figure we show that when the questions are poorly annotated we can obtain a greater result than that of the baselines because our model supports search for more specialized participants. This is very important especially for real life crowdsourcing where automatic keyword similarity methods annotating the questions, such as the ones described in [10], could fail capturing the most specialized and important keywords that characterize the question and thus gives more robustness to our model.

To sum up, in this experimental section we saw how the different crowds and task annotations worked in favor of our model and provided with very interesting results. According to Section 4.3 and more precisely figure 7 we were expecting to have an observable improvement with the use of taxonomies that have d_{max} >5. However we could show that even less deep taxonomies equipped with our model and mapping algorithms could give a real life improvement. Even when there was lack of information on the tasks our algorithm performed a better mapping that provided better results. All the above showed the robustness of our model despite the potential existence of spammers (people lying about their skills, or giving bad answers). To the best of our knowledge we are the first to use such an hierarchy of skills for reasoning on the substitution of participants on crowdsourcing applications and equip it with similarity metrics for better task to participant assignments.

5. RELATED WORK

5.1 Taxonomies of skills vs. Keywords

Skill management is one of the key issues for crowdsourcing platforms. Being able to gather a huge number of diverse and complementary skills constitutes one of the strengths of these platforms, beside their ability to deliver tasks on a regular basis.

It is noteworthy that most of the existing platforms and most of the related work in the literature rely on skills described as keywords from a free vocabulary, proposed by task requesters or by participants (see for example [10, 17, 19]). To reason about skills, keyword similarity metrics can be used, such as in [10]. First, this approach has the major advantage of being extremely

flexible: if a new skill appears (say, an expertise on a new electronic device), the skill name can be added by any participant and then become available for the overall community. Second, these systems can be seen as self-adaptive and self-tuning, as inappropriate or deprecated keywords can be ruled out by the natural effect of collaborative filtering. But flexibility can also be seen as a drawback for this approach:

- keyword dispersion: participants may use very different keywords for the same skill, leading to a blurred description of available competences;
- low precision: for a task requester looking for a precise skill s, a keyword-based system will return all skills s' equal to s or similar to s up to a given threshold. Hence the task requester is not guaranteed to obtain only the desidered skills. As a naïve example, looking for "Java" experts (computer language) could return "Java" experts (the real Island). Using vocabularies and entity resolution to solve this problem means indeed that a kind of taxonomy as the one we propose is required.
- low recall: due to the openness of the vocabulary, a task requester looking for a skill s is never sure to have exhausted all the possible keywords that correspond to her goal.

In this paper we advocate the use of structured vocabularies for skill management. A structured vocabulary can be a tree (a taxonomy of skills) or a more general ontology of skills (that we do not consider in the present work). We would like to illustrate the various benefits we envision:

- Complementarity: it should be observed that our proposal is complementary to keyword-based systems, and that existing methods for skill estimation can be enriched with ours (given nevertheless technical adaptation). Our work proposes to structure the similarity between skills, so it could extend the comparison methods used in the related work.
- Reasoning on skills: being structured as a tree, skills can be substituted by more specific or more generic skills with a clear semantics.
- Availability: participatory platforms in science, especially in biology and ecology, are already using taxonomies to structure image labeling and hence participant abilities (see for example the SPIPOLL platform or BumbleBee-Watch). In these domains, taxonomies are already available to apply our technique. The situation is less advanced for generic applications, but as mentioned in the introduction, several generic skill taxonomies already exist (e.g. ESCO).
- Support for crowd assessment: finally, using taxonomies to structure the crowd allows for an efficient design of targeted qualification tests or quizzes. For example, participants can start by answering generic questions (top of the taxonomy) and go on with more and more specialized questions (bottom of the taxonomy).

Table 2 sums up this comparison.

Criteria	keyword-based	taxonomy based
flexibility	+	-
dynamicity	+	-
precision	-	+
recall	-	+
reasoning	-	+
stuctured assessment	-	+

Table 2: Taxonomy- vs. keyword-based skill management

5.2 Skill models and crowdsourcing platforms

Several papers propose the use of skill models to improve the task assignment techniques in crowdsourcing platforms.

Bozzon et al. [4] present a way to extract experts in different domains from social networks data. The method they propose consists of a meta model of evaluating resources shared in a social network and social relations between people participating in these networks. The resources in this model are assessed with a direct and an indirect way. The direct method considers the direct profile info of the participants, while the indirect method relates the resources with a depth distance of two from each corresponding user. They also comprised the bi-directionality of friendship to asses the quality of the relation between people. A user's expertise is represented as a label and a normalized value from the interval zero to one. They do not rely on a taxonomy of skills.

Maary et al. [14] in their vision paper present a potential model of an ontology-based skill-modeling methodology. The goal of their approach is to assure the quality of answers based on the level of available skills in the crowd. As their work is a vision paper, they did not provide an implementation and evaluation of the platform. The authors argue that uncertainty is inevitable when human workers are involved in the computation and that is why quality assurance is needed to eliminate the unqualified workers who do not pass the quality threshold.

Instead of error rate matching techniques they claim that finding the potential match is more efficient so they propose a skill ontology and competencies model that given an annotated task the corresponding to this domain of expertise participants will be questioned. The quality of the taxonomy can be judged in 4 different levels. The result's quality, the platform's quality, the task's quality and the worker's quality. Respecting this taxonomy they propose a skill ontology based model consisting of the skill ontology, the ontology merger, the skill library of assessments, the skill aligner, the reputation system and the task assigner.

In their article [19] Roy et al. formalize the problem of task assignment given a certain pool of workers and propose different algorithms to solve it. Similarly to our work, they address problems in the context of knowledge-intensive crowdsourcing. They define knowledge-intensive crowdsourcing as the process of collaborative creation of knowledge content. In their paper, they assume -as we also do- that workers are never interrupted: they complete their task once they accept doing it. Their other assumption is that minimum task per user is one and maximum task per user is two. They provide three different algorithms to find an optimal task assignment: the C-DEX optimal, the C-DEX approximative (deterministic and non-deterministic) and the C-DEX+. The first uses the reduction form the multiple knapsack problem and thus is using the dual-primal problem properties from integer linear programming. The second is a greedy approximation based on the knapsack problem with either deterministic or stochastic criteria. The third of their algorithms reduces the variables of the problem by grouping the

participants that have same or equivalent profiles to one virtual worker group represented by the highest amount of money among them and the minimum knowledge among them. The algorithm then solves the reduced variable problem with the same integer linear programming method using the same tasks as in the C-DEX optimal but also the virtual workers instead of all the workers. All algorithms have an offline and an online phase. In the offline phase the indexes are created for all participants no matter their connectivity status. On the online status only the online participants are taken into account in the reduced problem. The main difference to our work is the model for representing worker skills. While Roy et al. [19] rely on a simple keyword-based skill modeling (associated with a value from [0,1]), we exploit the presence of a skill taxonomy. This allows us to take into account the whole taskforce of the platform by reasoning about skills w.r.t. to the skill taxonomy.

Amsterdamer et al. [2] also rely on an ontology to model a crowd, but they take a different perspective. Their goal is to extend a knowledge base, by focusing questions to relevant crowd members ("mining the crowd"). This approach is distinct from task assignment, where every task has to be assigned to a participant, even if her skills are not perfectly relevant.

Karger et al. [11] study the task allocation problem, similar to ours. However, they make completely different assumptions in their work, in particular, they do not use at all skill profiles in their algorithms.

In [17], the authors study the problem of team building in collaborative settings. They consider the optimization problem of building the best team of participants to reach a given task quality. The related skill model is also vector-based, but enriched with a probabilistic distribution.

5.3 Skill modeling

Modeling the skills of a group of people is a challenge in various contexts. Various models have been developed and are in use. A review of learner skill assessment techniques can be found in [8]. The goal of the authors is to provide models that could be useful in the context of eLearning platforms. Recommender systems have also been extended with skill profiles [15]. Campion et al. [6] survey the recent competence modeling techniques, for generic applications.

Bradley at al. [5] have developed a case-based system for providing adaptive multimedia content. In particular, in their work they analyze a Web search engine for jobs. They use similarity metrics to relate the different cases in their case-based system with respect to the user queries. Similarly to our approach their system can make simple reasoning about related (or subsumed) skills when it relates queries to the cases. The similarity function (between a query and the stored cases) is a simple weighted sum, while we use a more sophisticated distance function, inspired by the Resnic distance.

5.4 Task assignment for crowdsourcing platforms

Several recent work address the question of assigning tasks to participants in crowdsourcing platforms. Despite the similarity of the overall goal, the settings and assumptions of these works are rather different. For example, [23] studies the problem of task assignment in a streaming setting. On the other hand, crowd data analytic platforms of Fan et al. [10] and Liu et al. [13], assume a bounded budget for the assignments, that we do not have. Mo et al. [16] also assume a limited budget, and propose methods to optimize the overall quality of (aggregated) responses by varying the number of participants who perform the task. Cao et al. [7] and Zheng et al. [27] assign each task to multiple participants (in a jury form) and they analyze the trade-off between budget and the expected (aggregated) result that one can obtain this way. In our work the quality of an assignment is characterized by the overall compatibility of the skills of the participants and the tasks, which is more realistic in platforms such as AMT and Crowdflower, while other work try to find assignments where the tasks are expected to be completed timely, based on the participant's history records. In the version of the assignment problem studied by Acemoglu et al. [1], the participant skills are described by a hierarchical structure, however (unlike in our setting) the difficulties of the tasks are not known in advance. They also provide a pricing mechanism that results to an optimized result quality after multiple matching rounds. However, their work is motivated and studied within the domain of *Finance* and not *Computer Science*.

6. CONCLUSION AND FUTURE WORK

In this paper we have demonstrated the use of taxonomy-based skill modeling for crowdsourcing. Our techniques allow a simple form of reasoning about skills and participant substitution that is particularly useful for optimizing task assignment quality. We proposed several heuristics for task assignment to participants, and evaluated their respective performances in terms of quality and scalability through extensive experimentation.

In our future work, we will consider the relaxation of this model to incorporate participants with uncertain skills. We plan to investigate several further questions, with the help of our proposed model: 1) how to construct skill profiles (from their answer traces for instance), 2) how to identify and recruit experts in order to maximize the expected resulting quality, 3) how to optimize the task assignments in the presence of personal preferences, 4) how to include a cost model for task-cost estimation and 5) how to model complex tasks requiring more than one skills in order to be performed.

Acknowledgements

There are several people that contributed to the completion of this work. We would like to thank each of them separately. First, we would like to thank all the people that participated in the real experimentation filling in the Google forms from University of Rennes 1, IRISA and INRIA. Then, we thank the members of our research team DRUID, for the review of the several drafts of the paper before submission and the reviewers of WWW2016 for their helpful comments. Also, we would like to thank Ivaylo Petrov for his useful debugging advice. In addition, we would like to thank Romain Julliard and the SPIPOLL group from the Muséum national d'histoire naturelle (MNHM), for pointing us their challenges in participative science. Finally, we would like to thank the CNRS Mastodons initiative on big data, and especially the ARESOS project.

7. REFERENCES

[1] Acemoglu, D., Mostagir, M., and Ozdaglar, A. Managing innovation in a crowd. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation* (New York, NY, USA, 2015), EC '15, ACM, pp. 283–283.

[2] Amsterdamer, Y., Davidson, S. B., Milo, T., Novgorodov, S., and Somech, A. OASSIS: query driven crowd mining. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014* (2014), pp. 589–600.

[3] Bender, M. A., Farach-Colton, M., Pemmasani, G., Skiena, S., and Sumazin, P. Lowest common ancestors in trees and directed acyclic graphs. *J. Algorithms 57*, 2 (2005), 75–94.

[4] Bozzon, A., Brambilla, M., Ceri, S., Silvestri, M., and Vesci, G. Choosing the right crowd: Expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology* (2013), EDBT '13, pp. 637–648.

[5] Bradley, K., Rafter, R., and Smyth, B. Case-based user profiling for content personalization. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems* (2000), Springer-Verlag, pp. 62–72.

[6] Campion, M. A., Fink, A. A., Ruggeberg, B. J., Carr, L., Phillips, G. M., and Odman, R. B. Doing competencies well: Best practices in competency modeling. *Personnel Psychology 64*, 1 (2011), 225–262.

[7] Cao, C. C., She, J., Tong, Y., and Chen, L. Whom to ask?: Jury selection for decision making tasks on micro-blog services. *Proc. VLDB Endow. 5*, 11 (July 2012), 1495–1506.

[8] Desmarais, M. C., and d Baker, R. S. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction 22*, 1-2 (2012), 9–38.

[9] Enrich, M., Braunhofer, M., and Ricci, F. Cold-start management with cross-domain collaborative filtering and tags. In *E-Commerce and Web Technologies* (2013), C. Huemer and P. Lops, Eds., vol. 152 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 101–112.

[10] Fan, J., Li, G., Ooi, B. C., Tan, K.-l., and Feng, J. icrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (2015), SIGMOD '15, pp. 1015–1030.

[11] Karger, D. R., Oh, S., and Shah, D. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research 62*, 1 (February 2014), 1–24.

[12] Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2* (1955), 83–97.

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

[14] Maarry, K., Balke, W.-T., Cho, H., Hwang, S.-w., and Baba, Y. Skill ontology-based model for quality assurance in crowdsourcing. In *Database Systems for Advanced Applications*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 376–387.

[15] Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 54–88.

[16] Mo, L., Cheng, R., Kao, B., Yang, X. S., Ren, C., Lei, S., Cheung, D. W., and Lo, E. Optimizing plurality for human intelligence tasks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013* (2013), pp. 1929–1938.

[17] Rahman, H., Thirumuruganathan, S., Roy, S. B., Amer-Yahia, S., and Das, G. Worker skill estimation in team-based tasks. *PVLDB 8*, 11 (2015), 1142–1153.

[18] Resnik, P. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res. (JAIR) 11* (1999), 95–130.

[19] Roy, S. B., Lykourentzou, I., Thirumuruganathan, S., Amer-Yahia, S., and Das, G. Task assignment optimization in knowledge-intensive crowdsourcing. *VLDB J. 24*, 4 (2015), 467–491.

[20] Tamir, D. 500000 worldwide mechanical turk workers. In *Techlist* (Retrieved September 17, 2014).

[21] Victor, P., Cornelis, C., Teredesai, A. M., and De Cock, M. Whom should i trust?: The impact of key figures on cold start recommendations. In *Proceedings of the 2008 ACM Symposium on Applied Computing* (New York, NY, USA, 2008), SAC '08, ACM, pp. 2014–2018.

[22] Vuurens, J., and de Vries, A. Obtaining High-Quality Relevance Judgments Using Crowdsourcing. *IEEE Internet Computing 16*, 5 (2012), 20–27.

[23] Wang, D., Abdelzaher, T., Kaplan, L., and Aggarwal, C. C. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on* (2013), IEEE, pp. 530–539.

[24] Zhang, J., Tang, J., and Li, J.-Z. Expert finding in a social network. In *DASFAA* (2007), K. Ramamohanarao, P. R. Krishna, M. K. Mohania, and E. Nantajeewarawat, Eds., vol. 4443 of *Lecture Notes in Computer Science*, Springer, pp. 1066–1069.

[25] Zhang, W., and Wang, J. A collective bayesian poisson factorization model for cold-start local event recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2015), KDD '15, ACM, pp. 1455–1464.

[26] Zhao, Z., Cheng, J., Wei, F., Zhou, M., Ng, W., and Wu, Y. Socialtransfer: Transferring social knowledge for cold-start cowdsourcing. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (New York, NY, USA, 2014), CIKM '14, ACM, pp. 779–788.

[27] Zheng, Y., Cheng, R., Maniu, S., and Mo, L. On optimality of jury selection in crowdsourcing. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.* (2015), pp. 193–204.