Task Composition in Crowdsourcing

Sihem Amer-Yahia, Eric Gaussier, Vincent Leroy, Julien Pilourdault, Ria Borromeo, Motomichi Toyama

Task Composition in Crowdsourcing

Sihem Amer-Yahia, Eric Gaussier, Vincent Leroy, Julien Pilourdault
Univ. Grenoble Alpes, CNRS, LIG
F-38000 Grenoble, France
firstname.lastname@imag.fr

Ria Mae Borromeo, Motomichi Toyama
Keio University
Yokohama, Japan
riamae@keio.jp, toyama@keio.jp

***Abstract*—Crowdsourcing has gained popularity in a variety of domains as an increasing number of jobs are "taskified" and completed independently by a set of workers. A central process in crowdsourcing is the mechanism through which workers find tasks. On popular platforms such as Amazon Mechanical Turk, tasks can be sorted by dimensions such as creation date or reward amount. Research efforts on task assignment have focused on adopting a requester-centric approach whereby tasks are proposed to workers in order to maximize overall task throughput, result quality and cost. In this paper, we advocate the need to *complement* that with *a worker-centric approach* to task assignment, and examine the problem of producing, for each worker, *a personalized summary of tasks* that preserves overall task throughput. We formalize task composition for workers as an optimization problem that finds a *representative* set of k *valid* and *relevant* Composite Tasks (CTs). Validity enforces that a composite task complies with the task arrival rate and satisfies the worker's expected wage. Relevance imposes that tasks match the worker's qualifications. We show empirically that workers' experience is greatly improved due to task homogeneity in each CT and to the adequation of CTs with workers' skills. As a result task throughput is improved.**

I. Introduction

Crowdsourcing has gained popularity in a variety of domains as an increasing number of jobs are "taskified" and completed independently by a set of workers. Tasks range from simple requests such as transcribing a store receipt, to ones requiring the participation of skilled workers, such as text translation. A central process in crowdsourcing is task assignment, the mechanism through which workers find tasks. In this paper, we advocate the need to complement existing requester-centric and platform-centric task assignment strategies, with *a worker-centric approach* to task assignment. We examine the problem of producing, for each worker, *a personalized summary of tasks* that preserves overall task throughput. To the best of our knowledge, this is the first work that addresses worker-centric task assignment and shows the impact of such an approach on the effectiveness of crowdsourcing.

Today, task assignment is achieved by letting workers self-appoint themselves via an interface showing a list of tasks. For example, on Amazon Mechanical Turk (AMT), workers can rank tasks by creation date, reward amount, etc. In practice, workers look for tasks they can easily complete, i.e., that match their skills, and that provide a desired compensation. Indeed, a thorough examination of crowdsourcing forums such as TurkerNation,[1] reveals that workers spend non-negligible amounts of time discussing how to best select tasks depending on one's goals, which requesters to ban, and which skills are required for the latest tasks on AMT. That calls for re-thinking task assignment and developing an approach to find the tasks that best fit workers' preferences. Recent research on task assignment is wither requester-centric or platform-centric, whereby tasks are proposed to workers in order to maximize overall task throughput, result quality, and cost [1], [2]. These approaches do not take into account workers' preferences. To combine the best of both worlds, we propose to help workers find relevant tasks and enforce the selection of tasks according to their arrival rate. Satisfying workers while keeping in mind the need to optimize task throughput is the challenge we are tackling in this work.

We conjecture that since not every worker is interested in every task, we need to provide *a personalized summary* of tasks to each worker. We propose to build for each worker, a set of Composite Tasks (CTs) where each CT is *valid*, i.e., contains tasks that reflect their arrival time and offering a total reward that satisfies the worker's desired compensation and is *representative* of the set of input tasks. Moreover, we conjecture that workers who come to a crowdsourcing platform with a target reward in mind, expect to see a set of tasks for which they are proficient. Hence, each CT must also be *relevant*, i.e., containing tasks that match the worker's skills.

The problem of building CTs is inspired from the problem of building composite items (CIs). CIs have been shown to be effective in solving complex information needs such as planning a city tour, selecting books for a reading club, and organizing a movie rating contest [3]–[12]. CTs are a form of personalized CIs due to the relevance component. The formulation in [6] has recently been shown to outperform previous work in building valid and representative CIs. We propose to extend it with relevance and express a maximization problem involving similarity between worker and task skills. We find that it is most natural to produce possibly overlapping CTs where each CT constitutes an option for the worker. Therefore, similarly to [6], we rely on fuzzy clustering to produce CTs. Each cluster of CTs is a CT and each task may belong to more than one cluster. Our algorithm differs from the one in [6] in that it solves a 3-way optimization problem where one part aims at identifying task representatives (i.e., cluster centroids obtained through fuzzy clustering), another ensures that the representatives chosen are close to valid CTs,

[1] *http://turkernation.com/*

and a last part ensures that the returned CTs are relevant to the worker.

Our user study involving tasks and workers from AMT, demonstrates the superiority of CTs over ranking tasks by creation date or by reward. We study the impact of the level of relevance for workers. We show empirically that improving workers' experience in finding relevant and useful tasks also improves task throughput.

In summary, this paper makes the following contributions:

- We introduce the problem of producing personalized summaries of tasks for a given worker. Our problem is based on building a set of k valid CTs that maximize task representativity and task relevance to a worker.
- We define validity that glues together tasks with different arrival rates into a single CT satisfying a worker's desired reward expressed using *a summarization vector* as follows:

  $$\langle 2(tdy), 2(tdy-2), 1(tdy-3), \$2\rangle$$

  This summarization vector specifies that each CT must contain 2 tasks from today, 2 from 2 days ago and 1 task from 3 days ago, and the total reward for those tasks must be at least \$2. Our clustering objective function captures representativity and relevance. Representativity aims at finding the k CTs that cover best the input set of tasks. Relevance is ensured by selecting tasks that match a worker's set of skills.
- We design a constraint-based fuzzy clustering algorithm that seamlessly integrates validity, representativity and relevance.
- We run experiments with real workers on AMT and explore the quality of CTs we produce. In particular, we show that our CTs have high quality (validity), provide a good coverage of input tasks (representativity), and are appreciated by workers (relevance). Our experiments validate the conjecture that workers who come to a crowdsourcing platform with a reward in mind, are looking for tasks they are proficient in. Indeed, since each CT contains similar tasks, workers are more likely to quickly complete them than to quickly complete a set of heterogeneous tasks. We also run performance experiments demonstrating that our algorithm scales linearly with different parameters.

This paper is organized as follows. Section II introduces our formalization and defines our task composition problem. Section III describes our algorithm. Experiments are provided in Section IV. Related work and conclusion are given in Sections V and VI respectively.

II. Model and Problem

We first define our model and then we formally introduce our problem statement.

A. Model

We are given a set of workers U and a set of micro-tasks $\mathcal{X} = \{x_1, \cdots x_n\}$ posted by a set of requesters $Q = \{q_1, \cdots q_m\}$.

Skills. $\mathcal{S} = \{s_1, \cdots, s_p\}$ is a set of p skills. A skill could represent a domain/topic of interest or expertise and refers to a required-by-task or acquired-by-worker skill. For example, a translation task requires language skills, a task that gathers school names in a city requires to be familiar with that city, and a task that verifies the presence of some items in an image does not require any particular skill.

Workers. Every worker $u \in \mathcal{U}$ is defined as:

$$u = \langle id_u, V_u, o\rangle$$

where id_u is a unique identifier, $V_u = \langle s_1^u, \cdots, s_p^u\rangle$ is a vector of skill weights for u and each s_j^u corresponds to u's proficiency in skill $s_j \in S$, and o_u is the opinion the worker u has regarding requesters and tasks.

Opinion. We adopt the definition in [13] where an opinion o of a worker u is a tuple containing entries of the form (E, F) where E is an entity and F is a vector of features. According to TurkerNation,[2] workers often discuss the requesters they do not want to work with and the skills they do not possess. Hence, we propose to model the opinion o of a worker u as follows: $o = (BannedRequesters, \langle q_1, \cdots\rangle), (UnSkilled, \langle s_1, \cdots\rangle))$.

Tasks. Every micro-task $x \in \mathcal{X}$ is defined as follows:

$$x = \langle id_x, V_x, d, q, r\rangle$$

where id_x is a unique identifier, V_x is defined in the same manner as V_u, and represents a vector of minimum skill weights required by task x, d is the timestamp representing the creation date of task x, $q \in Q$ is the requester who posted task x, and r is the reward of task x.

For example, on AMT, a task that identifies a relation between two named entities in a sentence pays \$0.15 and a receipt classification task pays \$0.02.

Definition 1 (Summarization Vector): Given a set $\mathcal{X}$ of micro-tasks, we define our summarization vector as follows:

$$\vec{sv} = \langle \#t_1, \cdots, \#t_l, R\rangle$$

where each $\#t_j, 1 \leq j \leq l$ specifies the number of tasks in a given time interval t_j and R is a total reward.

A summarization vector indicates which tasks should be made visible to a worker. It depends on the daily arrival rate of tasks and specifies a total reward R.

For example, the summarization vector

$$\vec{sv} = \langle 2(tdy), 1(tdy-1), 1(tdy-3), \$2\rangle$$

represents 2 tasks with creation date of today, 1 task with a creation date of yesterday, and 1 task from 3 days ago, with a minimum total reward of \$2.

The summarization vector is used to define valid Composite Tasks (CTs).

Definition 2 (Validity): Given a set of tasks $\mathcal{X}$ and a summarization vector $\vec{sv} = \langle \#t_1, \cdots, \#t_l, R\rangle$, a valid $CT = \{x_1, \cdots, x_g\,; x_i \in \mathcal{X},\ 1 \leq i \leq g\}$, is a set of tasks such that

[2] *http://turkernation.com/*

their arrival date fits the time intervals (i) and the total reward of tasks forming the CT is at least as high as R (ii):

$$\begin{cases} (i)\ \forall \#t_j \in \vec{sv},\ \sum_{i=1}^{g} \mathbb{1}(t_j, x_i.d) = \#t_j \\ (ii)\ \sum_{i=1}^{g} x_i.r \geq R \end{cases}$$

where $\mathbb{1}$ is an indicator function which is equal to 1 if the arrival date of the task $x_i.d$ belongs to the summarization vector time interval t_j and 0 otherwise. g is the number of tasks in the CT and $g \geq n$, where n is the number of timestamp values considered. We refer to the set of all valid CTs as $\mathcal{V}_{CT}$.

As an example, assume that we have the summarization vector $\vec{sv} = \langle 2(tdy), 1((tdy-3)\&(tdy-2)), \$2\rangle$ and three tasks x_1, x_2 and x_3 with arrival dates $x_1.d = 2015/06/09$, $x_2.d = 2015/06/05$, and $x_3.d = 2015/06/09$. If today's date is $tdy = 2015/06/05$ then, $\mathbb{1}(t_j, x_i.d)$ will assign 1 to tasks x_1 an x_3 and 0 to task x_2.

Skill Similarity. Skill similarity is measured either between tasks or between tasks and workers. We use an abstract function termed $sim()$ on on V_x and V_u, the vectors of skills describing tasks and workers, to represent either similarities. Several vector and set similarity functions could be used in practice. In our study, we propose to use Cosine Similarity that returns a value between 0 and 1.

B. Applicability of fuzzy clustering

Our goal is to produce a summary of tasks for each worker. A summary is formed by *valid* and *relevant* CTs, and is *representative* of the set of available tasks. The validity of a CT is expressed in terms of a summarization vector $\vec{sv} = \langle \#t_1, \ldots, \#t_n, R\rangle$ as presented in Definition 2. Its relevance is the adequation between the opinion of the worker and the tasks forming the CT, and is evaluated using $sim()$.

This objective bears a resemblance to the problem of generating representative composite items to summarize heterogeneous item collections [6]. KFC, the algorithm that solves that problem, relies on fuzzy clustering to position k centroids that "cover" the whole dataset. Composite items are then formed in the vicinity of these centroids, which ensures that they provide a good summary of the dataset. In the context of this work, we may want to see a given task in different CTs. Contrary to *hard* clustering, *fuzzy* clustering allows each data point to participate to each cluster [14]. Thus, KFC also allows this flexibility.

With KFC, the summaries are valid, representative, and cohesive (e.g., items forming a cluster are geographically close). Our work builds on that and adds the notion of relevance to a worker. Hence, instead of simply providing a summary of the dataset, our goal is to account for the profile of the user to personalize the results. We now state our problem more precisely.

C. Our Problem

Given a worker u, a set of n micro-tasks $\mathcal{X}$, and a summarization vector $\vec{sv}$, we define a set of k composite tasks $S_u = \{CT_1, CT_2, \cdots, CT_k\}$ where each $CT_i \subseteq \mathcal{X}$ is a valid composite task and S_u optimizes the following objective:

$$\begin{aligned} \underset{C,W}{\operatorname{argmax}}\ \alpha \sum_{j=1}^{k}\sum_{i=1}^{|\mathcal{X}|} w_{ij}^{m}\ \ sim(x_i, c_j) + \\ \sum_{j=1}^{k} \max_{CT_j \in \mathcal{V}_{CT}} \Big(\beta \sum_{x \in CT_j} sim(x, c_j) + \\ \gamma \sum_{x \in CT_j} sim(x, u) \Big) \\ s.t.\ \forall i \in [1, |\mathcal{X}|],\ \sum_{j=1}^{k} w_{ij} = 1 \end{aligned}$$

where C is a set of k centroids, W is a weight matrix of size $|\mathcal{X}| \times k$ which contains the w_{ij} weights indicating which task belongs to which cluster. α, β, γ are worker-dependent parameters controlling the weight of the optimization objectives, and $m \leq 1$ is the weighting exponent used in fuzzy clustering.

The first two components of the objective function are inherited from KFC [6] and capture representativity by choosing tasks x_i that are close to the centroid c_j of each of the k clusters. Those components serve to identify cluster centroids c_j that are representative of the complete dataset, while ensuring that the centroids obtained are close to some valid CT (CT_j). Maximizing the sum of the similarities of all tasks in a CT to its centroid additionally ensures the cohesion of the valid CT considered. It is the compromise between these two components that allows the identification of valid and representative CTs.

The last component, weighted with γ, captures relevance to the worker by comparing similarity between the skill vectors of a task x and the worker u, and provides personalization. Relevance is also enforced by taking into account the opinion of a worker u regarding requesters and skills as follows:

$$\forall CT_j \in \mathcal{V}_{CT} \text{ and } \forall x \in CT_j\text{:}$$

1) $x.q \notin u.o(BannedRequesters)$
2) $x.V_u \notin u.o(UnSkilled)$

We ensure this property by initially filtering $\mathcal{X}$ to remove any task that violates this definition. This allows the algorithm to focus on producing CTs that are representative of tasks that matter to this specific worker, rather than all tasks.

We will see in Section IV, how varying α, β, γ, affects the resulting CTs.

III. Algorithmic solution

Our approach to generating CTs builds on a fuzzy clustering algorithm similar to k-means. We first present the main loop of the algorithm, that iteratively updates centroid positions and CTs until convergence. Then we focus on the function that builds a CT around a centroid, and present our greedy approach to solving this problem.

A. Convergence loop

The problem of building composite tasks has similarities with building representative composite items. Thus, our algorithm for building CTs extends the KFC algorithm [6] and adopts the structure of fuzzy clustering, as described in Algorithm 1. Fuzzy clustering starts by initializing C, the set of k centroids to random positions (line 2). W is the weights' matrix representing the participation of each task from $\mathcal{X}$ to each cluster. Fuzzy clustering iteratively updates weights (line 6) and centroid positions (line 8) until convergence (line 9) [14]. Similarly to KFC, whenever centroids are updated, a CT is build around each centroid (line 7) in order to update S_u, the set of k composite tasks built for the worker u. To select the tasks added to each CT, we rely on a selection function f, which is detailed in Section III-B. Parameter α, β and γ are gradually adjusted to their target values over η cycles (lines 4 and 10). Hence, the algorithm starts by executing a standard fuzzy clustering ($\alpha = 1$, $\beta = \gamma = 0$), which ensures that centroids are well positioned, before gradually increasing the importance of cohesiveness and personalization in subsequent iterations (line 5).

The update rules for weights and centroids when using cosine similarity are given in [6] (Equations 6 and 7). While our problem definition differs from the one of KFC, the worker personalization term does not involve the position of the centroids, so the update rules are the same. However, the function that builds the CT corresponding to a centroid, represented in Algorithm 1 by f, needs to be adapted to account for the specificities of our problem. We describe that adaptation next.

Algorithm 1 Algorithm for building CTs

Input: set of tasks $\mathcal{X}$, summarization vector $\vec{sv}$, α, β, γ, number of iterations η, CT selection procedure f, worker u

Output: Set S_u of k CTs generated for worker u

1: $S_u \leftarrow \emptyset$; $\alpha' = 1$; $\beta' = 0$; $\gamma' = 0$
2: Initialize (e.g. through random assignment) C
3: **repeat**
4: $\quad \alpha' = \alpha' - \frac{1-\lambda}{\eta}$; $\beta' = \beta' + \frac{\beta}{\eta}$; $\gamma' = \gamma' + \frac{\gamma}{\eta}$
5: $\quad$ **repeat**
6: $\quad\quad$ Update cluster membership weights W
7: $\quad\quad$ Update S_u by generating CT for each centroid in C using f (Algorithm 2)
8: $\quad\quad$ Update centroid positions C
9: $\quad$ **until** convergence
10: **until** $\alpha' = \alpha \wedge \beta' = \beta \wedge \gamma' = \gamma$
11: $S_u \leftarrow f(C, \vec{sv}, u)$ (with the final f and C obtained)
12: **return** S_u

B. CT selection

We describe in detail the function f that builds the CT corresponding to a centroid position. Because the summarization vector $\vec{sv}$ considered here has two parts, related respectively to the daily rate and reward (see Definition 1), we rely on two scenarios associated to two different choices for f.

In the first scenario, we restrict ourselves to the daily rate constraint: $\vec{sv} = \langle \#t_1, \ldots, \#t_n \rangle$, without any minimum reward constraint. In that particular case, it is possible to efficiently compute, for any c_j, the CT CT_j that maximizes the objective function $\max_{CT_j \in \mathcal{V}_{CT}} \left(\beta \sum_{x \in CT_j} sim(x, c_j) + \gamma \sum_{x \in CT_j} sim(x, u) \right)$ through the following process:

1) Start with an empty CT: $CT_j \leftarrow \emptyset$
2) For each time interval $i = 1$ to l, add to CT_j the $\#t_i$ tasks of day type t_i that maximize $\beta\ sim(x, c_j) + \gamma\ sim(x, u)$
3) Return CT_j

The function f defined by the above algorithm has a complexity of $\mathcal{O}(|\mathcal{X}|g)$ in the worst case, where g is the cardinality of the CT. Indeed, without a reward constraint, maximizing the objective function is akin to a top-g query in which the score of each task is pre-computed. Given a centroid c_j, it returns the optimal CT according to the objective function.

In the second scenario, we consider the reward constraint in addition to the daily rate constraint, leading to the general summarization vector: $\vec{sv} = \langle \#t_1, \ldots, \#t_n, R \rangle$. In that case, one cannot directly use the above approach, as it does not always respect the minimum reward constraint. In fact, finding the optimal CT is NP-hard. The existence of a valid solution however can be easily computed in $\mathcal{O}(|\mathcal{X}|g)$, by selecting for each time interval the most rewarding task without considering the objective function. We developed an approximate algorithm, described in Algorithm 2, with a $\mathcal{O}(|\mathcal{X}|g^2)$ worst-case complexity for building the CT corresponding to a centroid. Our approach, described in Algorithm 2, always returns a valid CT if it exists. It may not lead to an optimal solution in the sense of our objective function; it will nevertheless yield a valid CT (close to the centroid considered and similar to the worker's skill vector).

C. Our algorithm

Our greedy solution is detailed in Algorithm 2. We assume there exists a solution and that $\mathcal{X}$ only contains valid candidates, i.e. candidates from a time interval with a non-zero cardinality constraint. The CT is iteratively built as the set of tasks CT, initially empty (line 1 in Algorithm 2). The main loop of lines 4–15 is executed as long the cardinality constraints are not met. Each of its executions greedily adds a task x to CT. The set X contains the candidate tasks, and the algorithm selects the task $x \in X$ that maximizes the

expression of line 6, i.e. the task that contributes the most to the objective function. Once CT contains the desired number of tasks for a given time interval, all candidates from the same interval t_x are eliminated (line 13), so CT cannot exceed the cardinality constraints. Combined with the termination condition of the loop (line 4), that ensures that CT does not violate the first validity property (Definition 2 (i)). The second validity property, i.e. the reward requirement, is enforced by the auxiliary function FILTER (lines 17–32). The goal of this function is to eliminate candidate tasks that, if added to CT, would make it impossible to reach the minimum reward due to the greedy approach. FILTER iterates over all time intervals in the loop of lines 18–30 in order to compute a minimum reward threshold m_i for the tasks of this interval. Given a task of interval t_i FILTER computes, in a second loop (lines 20–28), the highest reward that could be obtained by selecting the most rewarding tasks for all unallocated task slots in CT. This is done in line 26 by selecting X_j, the top-$\#t_j$ tasks of interval t_j in terms of reward. If FILTER is currently evaluating t_j ($i = j$), then only $\#t_j - 1$ other tasks can be selected. In line 29, all tasks of t_i that do not meet the minimum reward threshold m_i are eliminated from the set of candidates X. The algorithm uses the FILTER function to update X in line 5 before each addition to CT. This ensures that, despite the greedy approach, CTs generated by this algorithm always meet the reward constraint (Definition 2 (ii)), in addition to the cardinality constraints discussed previously.

Algorithm 2 Composite Task selection

Input: $\mathcal{X}$, centroid c, worker u, summarization vector $\vec{sv} = \langle \#t_1, \ldots, \#t_n, R \rangle$, β, γ
Output: Set CT, the CT matching centroid c

1: $CT \leftarrow \emptyset$
2: $totalReward \leftarrow 0$
3: $X \leftarrow \mathcal{X}$
4: **while** $\exists \#t_i \in \vec{sv}, \#t_i > 0$ **do**
5: $\quad X \leftarrow$ FILTER$(X, R - totalReward, \vec{sv})$
6: $\quad \underset{x \in X}{\text{argmax}}\ \beta\, sim(x, c) + \gamma\, sim(x, u)$
7: $\quad totalReward \leftarrow totalReward + x.r$
8: $\quad t_x \leftarrow t_i \in \vec{sv} | \mathbb{1}(t_i, x.d)$
9: $\quad CT \leftarrow CT \cup \{x\}$
10: $\quad X \leftarrow X \setminus \{x\}$
11: $\quad \#t_x \leftarrow \#t_x - 1$
12: $\quad$ **if** $\#t_x = 0$ **then**
13: $\quad\quad X \leftarrow X \setminus \{x_i \in X, \mathbb{1}(t_x, x_i.d)\}$
14: $\quad$ **end if**
15: **end while**
16: **return** CT

17: **procedure** FILTER$(X, minReward, \vec{sv})$
18: $\quad$ **for all** $\#t_i \in \vec{sv}$ **do**
19: $\quad\quad m_i \leftarrow minReward$
20: $\quad\quad$ **for all** $\#t_j \in \vec{sv}$ **do**
21: $\quad\quad\quad$ **if** $i = j$ **then**
22: $\quad\quad\quad\quad nb \leftarrow \#t_j - 1$
23: $\quad\quad\quad$ **else**
24: $\quad\quad\quad\quad nb \leftarrow \#t_j$
25: $\quad\quad\quad$ **end if**
26: $\quad\quad\quad \underset{X_j \subseteq X, |X_j| = nb, \forall x \in X_j, \mathbb{1}(t_j, x.d)}{\text{argmax}} \sum_{x \in X_j} x.r$
27: $\quad\quad\quad m_i \leftarrow m_i - \sum x \in X_j x.r$
28: $\quad\quad$ **end for**
29: $\quad\quad X \leftarrow X \setminus \{x_i \in X, \mathbb{1}(t_x, x_i.d), x.r < m_i\}$
30: $\quad$ **end for**
31: $\quad$ **return** X
32: **end procedure**

IV. Experiments

We evaluate the benefits of building Composite Tasks through a study on Amazon Mechanical Turk. We first describe in Section IV-A the options evaluated in this study, with two ranked-lists and two CTs configurations. Then, we present the setup of this experiment in Section IV-B. We evaluate the impact of using CTs on workers in Section IV-C by measuring their likelihood to select tasks presented to them. Finally, we evaluate the impact of CTs on the crowdsourcing platform (Section IV-D) in terms of task throughput and overall execution time.

Summary of results. Overall, we find that building CTs significantly improves the experience of workers, at is gives them direct access to a set of tasks that allows them to meet their reward objective. Personalizing the CTs with respect to the skills of the workers further improves the likelihood that they select one of the tasks returned. This contributes to reducing the time workers spend looking for relevant tasks and allows them to complete more tasks overall. We also find that the use of a summarization vector that reflects arrival rate improves task throughput and reduces starvation problems. We demonstrate that the time required to build CTs increases linearly with the size of the dataset, but that the benefits for workers significantly outweigh the extra computational cost.

A. Task display options evaluated

Table I summarizes the different task display options considered in our evaluation. In all cases, we eliminate tasks for which the workers do not have qualifications (*unSkilled*), and tasks from unwanted requesters (*BannedRequesters*). The default interface of AMT allows workers to browse tasks as a ranked list. Tasks can typically be sorted by creation date, to find the tasks most recently submitted, or by reward, to access the highest-paying tasks. We refer to those possibilities as CRL and RRL respectively. The ranked-list paradigm constitutes the baseline in our experiments, as it is representative of the system used in a standard crowdsourcing platform. This paper advocates for presenting tasks using CTs built specifically for workers. Tasks are no longer displayed as a flat list, but rather as groups of tasks meeting objectives defined in the summarization vector. These include both constraints on the creation dates of the tasks (platform objective), and on the total reward of tasks within a given CT (worker objective). Options SCT and PCT both build k composite tasks for

Task Display Options	Description
CRL	A list of tasks relevant to a worker and ranked by creation date
RRL	A list of tasks relevant to a worker and ranked by reward
SCT	A set of k CTs, non-personalized wrt. workers' skills
PCT	A set of k CTs personalized wrt. workers' skills

TABLE I
TASKS DISPLAY OPTIONS

workers using Algorithms 1 and 2. The difference between them lies in the choice of parameters α, β and γ used in our problem definition. For SCT, we assume that worker skill vectors are unavailable (e.g. a new worker joining the platform). Hence, CTs are valid, coherent and representative, but not personalized to match the skills of the worker receiving them ($\alpha = \frac{1}{2}, \beta = \frac{1}{2}, \gamma = 0$). In the case of PCT however, we rely on the worker's skill vector to personalize the CTs and make them more appealing ($\alpha = \frac{1}{3}, \beta = \frac{1}{3}, \gamma = \frac{1}{3}$). Throughout our experiments, we set the number of CTs k to 6, and the fuzziness parameter m to 0.8. This corresponds to moderate fuzziness and significant differences between CTs.

B. Setup

a) Tasks dataset: We gather a list of tasks available on AMT using a Web crawler. Every hour, we retrieve the 300 most recent tasks and add them to a database. In practice, this means that we obtain almost all tasks submitted during the crawling period. Over a period of 18 weeks, between July 24 and August 12 in 2015, we obtain a dataset of 25,644 hits from 11,563 distinct hit groups. This dataset is then used to generate different task displays for workers using the 4 different approaches described in Table I. For each task, we record its ID, creation date, title, description, keywords, requester name, reward, time allotted and qualifications. We use LDA [15] on the keywords of tasks in order to discover 15 different topics of tasks. We describe each topic as a bag of words, keeping the 5 most characterizing words. These topics are listed in Figure 1 (keywords selection part). We notice the presence in our datasets of tasks illustrating the typical variety of micro-tasks, with a prevalence of demographic surveys, image tagging tasks, and audio transcription for instance.

b) Worker recruitment: We submit on AMT a first task aimed at recruiting workers. For this user study, we recruited a total of 70 workers. Figure 1 presents a screenshot of this task as seen by the workers. The goal of this task is to build, for each worker participating to the user study, a profile as described in Section II. Each worker selects qualifications from a checkbox list containing 15 options, and indicates her desired reward, banned requesters and banned skills. Here are the 5 topics that are the most popular among workers recruited for this study:

- survey, demographics, psychology, research, study (95%)
- easy, picture, cooler, ocmp, image (70%)

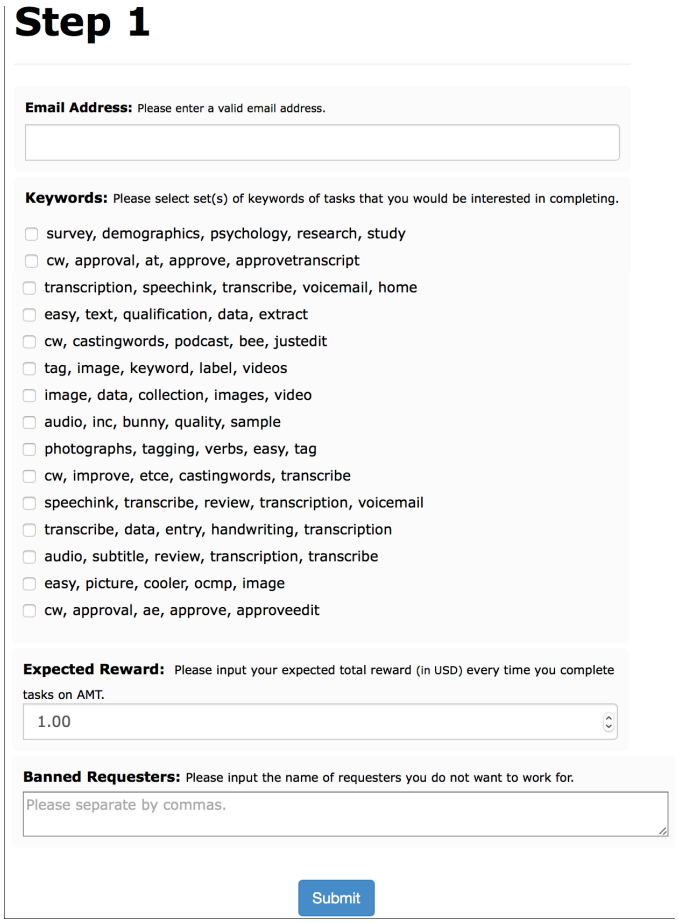

Fig. 1. Recruiting workers

- easy, text, qualification, data, extract (63%)
- tag, image, keyword, label, videos (61%)
- photographs, tagging, verbs, easy, tag (56%)

Thus, social studies are very popular, and so is annotating videos and images. Conversely, topics related to audio transcription had a low selection rate (15%). This is likely because such tasks require a quiet environment and dedicated equipment to listen to audio files. On average, each worker selected 6.2 topics, with a median at 5.

On average, workers indicated that they expected to earn \$1.23 each time they completed tasks on AMT. The standard deviation however is quite high, at \$1.24. 50% of the workers indicated a value of \$1, with a minimum of \$0.10 and a maximum of \$10. These very different values confirm that it is important to take each worker's personal objective into account when selecting sets of tasks to display.

After this recruitment phase, each worker is paid \$0.10 and is given a unique token in order to move on to the next phase of the study. The reward for this first task is low in order to encourage workers to perform the full study.

c) Task display options evaluation: For each worker, we build different task selections and displays according to the options described in Section IV-A. We first eliminate tasks the worker does not qualify for, and requesters the worker bans.

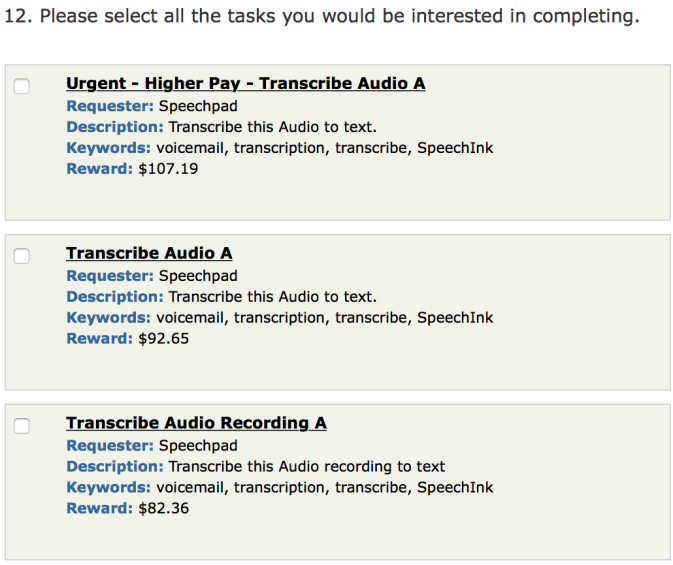

Fig. 2. Independent evaluation task

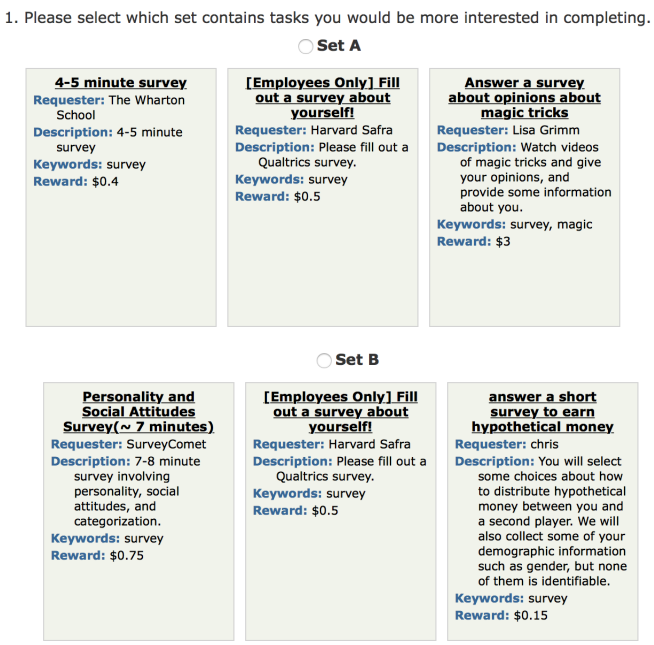

Fig. 3. Comparative evaluation task

Then, we rank tasks according to creation date and reward to produce CRL and RRL. In the case of CTs, we take into account the worker's target reward and skills are taken from the worker's profile gathered in the recruitment phase. The platform objective is the same for all workers, and is set to 2 tasks created in August 2015, and 6 tasks from September and November 2015. Hence, all CTs contain 8 tasks matching this interval distribution. Once these results are ready, we make them available on AMT through an evaluation task. The worker is identified by her unique token, and is presented the options generated from her profile.

We perform an independent evaluation of the task display options, in order to assess the relevance of the tasks they contain (Figure 2). We then compare pairs of display options to better understand the preferences of workers (Figure 3). We discuss the results of this user study in Section IV-C.

At the end of the evaluation, the worker is given a reward of $6.90, which makes the overall compensation for participating to the full study $7.

C. Worker impact

1) Independent evaluation: We first evaluate the 4 task display options independently. Workers are presented with tasks, either as lists (CRL and RRL), or as a set of $k = 6$ CTs (SCT and PCT) for each option. We ask them to select individual tasks that they would be interested in performing (Figure 2). The average and median task acceptance rates for each option are given in Table II.

CRL performs the worst, as listing tasks by creation date does not seem to bring any benefit to workers. They are willing to perform an average of 30% of tasks, but we note a very high variance. Indeed, some workers accept most tasks, and select many of them no matter the way they are displayed. However, selective workers, who constitute the majority of the workforce in this study, find few relevant tasks (median at 20%). Sorting tasks by reward RRL is an improvement, as it allows workers to reach their reward objective more easily by selecting tasks granting a high reward. With the SCT option, task acceptance rate is 38% on average, with low variance. Despite not being personalized, the CTs generated by SCT have multiple advantages. Each CT is homogeneous as it contains tasks that are similar to each other. This allows workers to quickly assess if they are interested in a whole set of tasks (i.e. the CT), rather than having to independently evaluate individual tasks. Workers also generally prefer performing several tasks of the same type in a row in order to be more efficient. Each CT is representative of a different types of tasks, which allows them to quickly get an overview the different type of tasks currently available on AMT. Workers are presented with sets of similar tasks that, taken simultaneously, let them achieve their reward objective. Hence, workers do not have to skim through long lists of tasks in order to find suitable ones. This finding validates our conjecture that workers who come to the platform with a reward in mind, do indeed look for similar tasks to be efficient. PCT outperforms SCT by 12% on average. This confirms that accounting for worker skills when building composite tasks further improves the relevance of the results presented to the workers, and thus their satisfaction.

2) Comparative evaluation: We then perform a comparative evaluation in which workers are shown simultaneously

Task Display Option	**Median**	**Average**
CRL	20%	30%
RRL	27%	35%
SCT	35%	38%
PCT	48%	50%

TABLE II
INDEPENDENT EVALUATION: ACCEPTANCE RATE OF TASKS PROPOSED

	CRL	RRL	SCT	PCT
CRL		45%	19%	21%
RRL	55%		14%	13%
SCT	81%	86%		44%
PCT	79%	87%	56%	

TABLE III
COMPARATIVE EVALUATION: PAIRWISE PREFERENCE OF TASK DISPLAY OPTIONS

two task display options, and are asked which one they prefer (Figure 3). In this context, a display option is either the top-8 results of a ranked list (RRL and CRL), or one of the 6 CTs chosen at random (PCT and SCT). Hence, each display option presents 8 tasks to the worker. The pairwise comparison results are given in Table III.

We first compare the two list-based task display options. We notice that workers favor RRL over CRL 55% of the time. This result is consistent with the independent evaluation, which shows that workers are more likely to perform tasks displayed by RRL, as their reward is higher. CT-based display options (SCT and PCT) always significantly outperform list-based options (RRL and CRL). The ratio goes from 79% for PCT against CRL to 87% for PCT against RRL. While the independent evaluation placed SCT relatively close to RRL, that was not the case in the comparative evaluation, as SCT is selected 81% of the time. This experiment shows a clear preference of workers for CT-based display options. Indeed, CTs allow workers to browse coherent sets of similar tasks, grouped together to allow workers to achieve their personal reward goal. This facilitates the tedious operation of browsing and selecting tasks to perform. This experiment also confirms that accounting for the worker's skills when generating CTs is preferable, as PCT is selected over SCT 56% of the time.

D. Platform impact

1) Task throughput: Presenting tasks as ranked list has the drawback of limiting the exposure of some of the tasks to workers. For instance, with CRL, a task can easily be hidden by a batch of more recent tasks submitted by a requester. This causes starvation problems, as these tasks are increasingly less likely to be selected by workers as they become older. Consequently, requesters tend to cancel those tasks and submit identical ones to obtain more recent creation timestamps and increase their visibility. We propose to solve this problem using CTs by letting the platform target specific types of tasks using the CT summarization vector (Section II). In the context of this experiment, the platforms ensures that each CT contains 2 tasks created in August 2015 and 6 tasks from September and November 2015. This solves the problem of starvation by promoting older tasks in the options presented to workers, and can be leveraged by the platform to adapt to varying tasks arrival rate.

We rely on the independent evaluation experiment presented in Section IV-C to measure the throughput of tasks depending on their creation time. Each CT contains 8 tasks that are ranked from the oldest to the most recent. We present in Figure 4 the likelihood of a task of a CT to be selected by a worker with respect to its creation time rank. We observe that this likelihood is quite stable for SCT, and completely even in the case of PCT. This demonstrates that the incorporation of creation time interval constraints in the creation of CTs does not negatively affect the results, as old tasks are as likely to be selected as recent ones. This also indicates that promoting older tasks can be a solution to increasing their visibility and helping reduce the variation in the latency of a task execution. Due to limited information from AMT, we were not able to evaluate extreme cases in which workers are purposely avoiding some very old tasks, because of their low reward for instance. We conjecture that, as long as the pool of tasks in each time interval remains sufficient, these results are valid. To deal with extreme cases, the most unattractive tasks could be discarded from the CT creation step when they have been presented many times but not selected by workers.

2) Execution time: Currently, crowdsourcing platforms display the same ranked list of tasks to all workers. Hence, very few resources are used to maintain this list of tasks. In this paper, we propose to personalize the tasks displayed for each worker. The first step (CRL and RRL) is to eliminate some tasks that do not match worker constraints in terms of skills and requesters. This filtering is relatively low cost, as it only requires reading the top of the global list of tasks

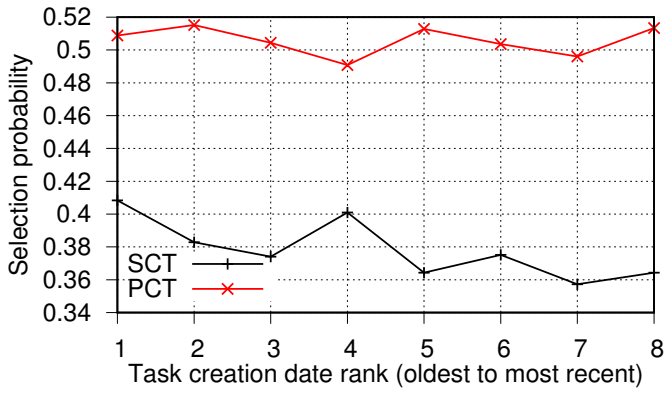

Fig. 4. Task throughput wrt. its creation time rank

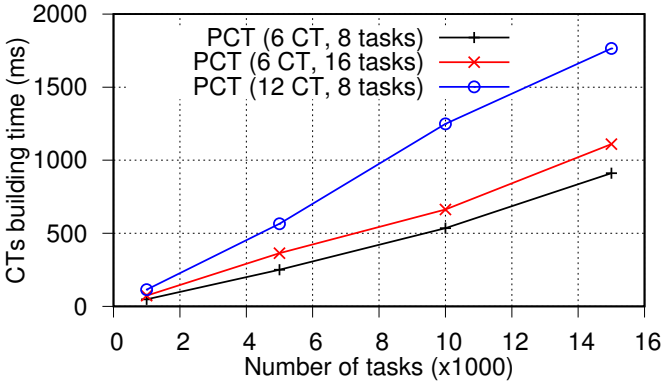

Fig. 5. CTs creation time

until the number of tasks matching the filtering criteria is sufficient. Maintaining CTs however is more costly. In this work, we present an algorithm derived from fuzzy clustering that builds CTs from scratch in a new dataset. In practice, the position of centroids can be used to speed up the execution when adding and removing a few tasks from the dataset. In this experiment, we show that even when building CTs from scratch for each user, the benefits of using CTs outweigh the extra computational cost for the crowdsourcing platform.

In Figure 5, we present the time required to build CTs for different configurations. The first configuration considers the case of 6 CTs of 8 tasks each built using PCT. Then, we double the number of tasks in each CT by altering the summarization vector to obtain 6 CTs of 16 tasks. Finally, we double the number of CTs to obtain 12 CTs of 8 tasks. This experiment is executed on an Intel i7-4870HQ CPU single threaded, and we vary the number of tasks selected initially in order to evaluate the scalability of the algorithm. We observe that the execution time increases linearly with the size of the dataset, which is consistent with the behavior of algorithms derived from k-means. We also observe a linear increase in the execution time with respect to the number of CTs. While the CT selection algorithm presented in Section III-C has a complexity that is quadratic in the number of tasks in a CT, the total execution time is dominated by the main loop of fuzzy clustering, so CT construction overall is not very sensitive to the number of tasks per CT. This execution time is negligible compared to the time it takes workers to perform tasks. This is particularly true as workers are more likely to select tasks contained in CTs. Hence, the platform CPU usage is acceptable.

V. Related Work

To the best of our knowledge, this work is the first to study personalized summaries of micro-tasks for workers. This work is related to building composite items, task decomposition and task assignment in crowdsourcing. We hence summarize the related work in all three areas.

a) Composite Items: Composite retrieval was studied with different semantics in recent work [3]–[9], [12]. Most existing algorithms rely on a two-stage process that decouples constraint satisfaction (e.g., a CI must contain one museum and 2 restaurants) from the optimization goal (e.g., each CI is a set of close landmarks in a city). In [6], it was shown that an integrated approach produces better representative CIs than a two-stage approach. We hence build on the approach presented in [6] and extend it to consider our goal, namely, building valid, representative and relevant CTs.

Personalized CIs were proposed in [16] to build customized city tours. Tailored itineraries are extracted from Flickr using an objective function that combines POIs popularity with the actual user preferences over POI categories. This approach is not directly applicable to ours since the personalization is merely a filtering of extracted trajectories while in our case, it is the task composition itself that is personalized (using the expected reward). That makes our problem computationally more challenging.

b) Task Decomposition: Crowdsourcing marketplaces such as AMT[3] are primarily designed for micro-tasks. Hence, complex tasks must be decomposed into easier sub-tasks [17]–[22]. For instance, in [19], each video annotation task is decomposed into a sequence of sub-tasks, based on a 3-stage workflow. To improve the quality of crowd work, the decomposition may also return collections of redundant sub-tasks [19]–[21], [23] and use voting to aggregate workers' responses. Previous work such as CrowdForge [24] and Turkomatic [25], proposed frameworks to design crowdsourcing workflows. *Task composition* is hence not the reverse operation of *task decomposition*. The micro-tasks that form a composite task remain small, independent and simple, thus preserving the benefits of task decomposition.

c) Task Assignment: While on AMT workers self-appoint themselves to tasks, some recent work studied the *task assignment problem* whereby tasks are assigned to workers so as to optimize some objective. Examples include the optimization of group design in collaborative crowdsourcing [26] or quality optimization in online settings [27]–[29]. Other studies assign several tasks to each worker [30]–[34]. For example, Roy et al. [33] assign to each worker a number of tasks in a range $[X_{min}, X_{max}]$ while maximizing the expected quality of crowd work. Zheng et al. [34] assign to each worker a task featuring k questions picked among n questions posted by a requester. Goel et al. [31] assign each worker a set of tasks in an online context where the objective is to maximize the utility of crowd work under budget constraints. None of the previous studies that assign tasks to a worker focus on defining valid, representative and relevant tasks. Only the optimization of some objective function measuring the quality of crowd work is considered.

VI. Conclusion

We presented and validated the first solution that combines requester and platform-centric crowdsourcing with worker-centric crowdsourcing. Indeed, to the best of our knowledge, existing work in task assignment is focused on optimizing requester goals such as budget, or platform goals such as task throughput. A thorough examination of TurkerNation[4] reveals that workers spend non-negligible amounts of time discussing how to best select tasks depending on one's goals, which requesters to ban, and which skills are required for the latest tasks on AMT. Therefore, we proposed to provide to workers homogeneous sets of tasks, termed Composite Tasks (CTs), that match their profiles and comply with their desired reward. Each CT also complies with task arrival rate as it contains a proportional number of tasks per time period.

We conducted a user study with AMT workers that compared four task display options: two ranked lists of worker-relevant tasks sorted by creation date or by reward, and two

[3]https://www.mturk.com/

[4]*http://turkernation.com/*

sets of CTs, one personalized using the worker's skills and one not. Our findings confirmed the superiority of personalized CTs in satisfying workers and improving task throughput.

We are currently exploring several avenues based on this work. The two short-term goals worth mentioning are: a classification of workers based on their reward and the time they spend completing tasks, and the use of that information to indicate whether CTs should be homogeneous or heterogeneous; the use of workers' feedback during task completion in order to refine their summarization vector. While existing work has focused on incentivizing workers for long-lasting tasks [35] [36] or entertaining workers during task completion [37], we would like to focus on learning workers' preferences as they evolve during task completion, and adapt each worker's summarization vector accordingly. This approach would account for the diversity of workers on a crowdsourcing platform and also for the evolution of their expectations during task completion.

ACKNOWLEDGMENT

This work was partially funded by ANR-13-CORD-0020.

REFERENCES

[1] 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, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, 2016, pp. 843–853.

[2] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Task assignment optimization in knowledge-intensive crowdsourcing," *VLDB J.*, vol. 24, no. 4, pp. 467–491, 2015.

[3] S. Amer-Yahia, F. Bonchi, C. Castillo, E. Feuerstein, I. Méndez-Díaz, and P. Zabala, "Composite retrieval of diverse and complementary bundles," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2662–2675, 2014.

[4] H. Bota, K. Zhou, J. M. Jose, and M. Lalmas, "Composite retrieval of heterogeneous web search," in *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, 2014, pp. 119–130.

[5] A. Brodsky, S. M. Henshaw, and J. Whittle, "Card: a decision-guidance framework and application for recommending composite alternatives," in *RecSys*, 2008, pp. 171–178.

[6] V. Leroy, S. Amer-Yahia, É. Gaussier, and S. H. Mirisaee, "Building representative composite items," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, 2015, pp. 1421–1430.

[7] A. Angel, S. Chaudhuri, G. Das, and N. Koudas, "Ranking objects based on relationships and fixed associations," in *EDBT*, 2009, pp. 910–921.

[8] S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu, "Constructing and exploring composite items," in *SIGMOD Conference*, 2010, pp. 843–854.

[9] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *HT*, 2010, pp. 35–44.

[10] A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd, "Time as essence for photo browsing through personal digital libraries," in *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2002, Portland, Oregon, USA, June 14-18, 2002, Proceedings*, 2002, pp. 326–335.

[11] A. Jaffe, M. Naaman, T. Tassa, and M. Davis, "Generating summaries and visualization for large collections of geo-referenced photographs," in *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2006, October 26-27, 2006, Santa Barbara, California, USA*, 2006, pp. 89–98.

[12] M. Xie, L. V. Lakshmanan, and P. T. Wood, "Breaking out of the box of recommendations: From items to packages," in *RecSys*, 2010.

[13] Q. Liu, Z. Gao, B. Liu, and Y. Zhang, "Automated rule selection for aspect extraction in opinion mining," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 1291–1297.

[14] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The Fuzzy c-Means Clustering Algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[16] I. R. Brilhante, J. A. F. de Macêdo, F. M. Nardini, R. Perego, and C. Renso, "Where shall we go today?: planning touristic tours with tripbuilder," in *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, 2013, pp. 757–762.

[17] V. Ambati, S. Vogel, and J. G. Carbonell, "Collaborative workflow for crowdsourcing translation," in *CSCW*, 2012, pp. 1191–1194.

[18] M. S. Bernstein, J. Teevan, S. T. Dumais, D. J. Liebling, and E. Horvitz, "Direct answers for search queries in the long tail," in *CHI*, 2012, pp. 237–246.

[19] J. Kim, P. T. Nguyen, S. A. Weir, P. J. Guo, R. C. Miller, and K. Z. Gajos, "Crowdsourcing step-by-step information extraction to enhance existing how-to videos," in *CHI*, 2014, pp. 4017–4026.

[20] W. S. Lasecki, C. D. Miller, A. Sadilek, A. Abumoussa, D. Borrello, R. S. Kushalnagar, and J. P. Bigham, "Real-time captioning by groups of non-experts," in *UIST*, 2012, pp. 23–34.

[21] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos, "Platemate: crowdsourcing nutritional analysis from food photographs," in *UIST*, 2011, pp. 1–12.

[22] K. Tuite, N. Snavely, D. Hsiao, N. Tabing, and Z. Popovic, "Photocity: training experts at large-scale image acquisition through a competitive game," in *CHI*, 2011, pp. 1383–1392.

[23] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD workshop on human computation*, 2010, pp. 64–67.

[24] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "Crowdforge: crowdsourcing complex work," in *UIST*, 2011, pp. 43–52.

[25] A. P. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *CSCW*, 2012, pp. 1003–1012.

[26] H. Rahman, S. B. Roy, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Task assignment optimization in collaborative crowdsourcing," in *ICDM*, 2015, pp. 949–954.

[27] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "icrowd: An adaptive crowdsourcing framework," in *SIGMOD*, 2015, pp. 1015–1030.

[28] C. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *AAAI*, 2012.

[29] D. R. Karger, S. Oh, and D. Shah, "Budget-optimal task allocation for reliable crowdsourcing systems," *CoRR*, vol. abs/1110.3564, 2011.

[30] S. Assadi, J. Hsu, and S. Jabbari, "Online assignment of heterogeneous tasks in crowdsourcing markets," in *HCOMP*, 2015, pp. 12–21.

[31] G. Goel, A. Nikzad, and A. Singla, "Mechanism design for crowdsourcing markets with heterogeneous tasks," in *HCOMP*, 2014.

[32] C. Ho, S. Jabbari, and J. W. Vaughan, "Adaptive task assignment for crowdsourced classification," in *ICML*, 2013, pp. 534–542.

[33] S. B. Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das, "Task assignment optimization in knowledge-intensive crowdsourcing," *VLDB J.*, vol. 24, no. 4, pp. 467–491, 2015.

[34] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "QASCA: A quality-aware task assignment system for crowdsourcing applications," in *SIGMOD*, 2015, pp. 1031–1046.

[35] D. Chandler and A. Kapelner, "Breaking monotony with meaning: Motivation in crowdsourcing markets," *CoRR*, vol. abs/1210.0962, 2012.

[36] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *Proceedings 11th ACM Conference on Electronic Commerce (EC-2010), Cambridge, Massachusetts, USA, June 7-11, 2010*, 2010, pp. 209–218.

[37] P. Dai, J. M. Rzeszotarski, P. Paritosh, and E. H. Chi, "And now for something completely different: Improving crowdsourcing workflows with micro-diversions," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW 2015, Vancouver, BC, Canada, March 14 - 18, 2015*, 2015, pp. 628–638.