Pick-A-Crowd: Tell Me What You Like, and I'll Tell You What to Do

A Crowdsourcing Platform for Personalized Human Intelligence Task Assignment Based on Social Networks

Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux
eXascale Infolab
U. of Fribourg—Switzerland
{firstname.lastname}@unifr.ch

ABSTRACT

Crowdsourcing allows to build hybrid online platforms that combine scalable information systems with the power of human intelligence to complete tasks that are difficult to tackle for current algorithms. Examples include hybrid database systems that use the crowd to fill missing values or to sort items according to subjective dimensions such as picture attractiveness. Current approaches to Crowdsourcing adopt a pull methodology where tasks are published on specialized Web platforms where workers can pick their preferred tasks on a first-come-first-served basis. While this approach has many advantages, such as simplicity and short completion times, it does not guarantee that the task is performed by the most suitable worker. In this paper, we propose and extensively evaluate a different Crowdsourcing approach based on a push methodology. Our proposed system carefully selects which workers should perform a given task based on worker profiles extracted from social networks. Workers and tasks are automatically matched using an underlying categorization structure that exploits entities extracted from the task descriptions on one hand, and categories liked by the user on social platforms on the other hand. We experimentally evaluate our approach on tasks of varying complexity and show that our push methodology consistently yield better results than usual pull strategies.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Experimentation, Human Factors.

Keywords

Crowdsourcing, Social Network, Expert Profiling

1. INTRODUCTION

Crowdsourcing techniques have recently gained in popularity as they allow to build hybrid human-machine information systems that combine *scalable* algorithms with the power of *human intelligence* to complete tasks that are difficult to tackle for current machines. Examples of such systems include hybrid database systems [12] that use the crowd to answer SQL queries with subjective ranking criteria such as picture attractiveness, hybrid linking approaches to relate unstructured text to structured entities [10], or data integration systems [21]. Crowdsourcing approaches must provide incentive schemes to motivate the crowd to perform a given Human Intelligence Task (HIT). While crowdsourcing games (also known as games with a purpose) exploit the fun incentive (see, for example, the ESP Game [25]), the most widely adopted incentive is financial, that is, to grant a small economic reward to each human worker that completes a task on the crowdsourcing platform.

Paid crowdsourcing is commonly run on top of platforms such as Amazon Mechanical Turk[1] (AMT), which provides programmatic APIs as well as a Web interface for requesters to design and deploy online tasks. Such platforms adopt a *pull* methodology where tasks published by requesters are available on specialized Web sites where workers can pick their preferred tasks on a first-come-first-served basis. This approach has several advantages including simplicity and minimization of task completion times, since any available worker from the crowd can pick and perform any HIT. However, this mechanism does not guarantee that the worker who performs the task is the best fit: More knowledgeable workers may be available in the crowd but they might be unable to pick the HIT if they were not quick enough.

In this paper, we propose and extensively evaluate *Pick-A-Crowd*, a software architecture to crowdsource micro-tasks based on *pushing* tasks to specific workers. Our system constructs user models for each worker in the crowd in order to assign HITs to the most suitable available worker. We build such worker profiles based on information available on social networks using, for instance, information about the worker personal interests. The underlying assumption is that if a potential worker is interested in several specific categories (e.g., movies), he/she will be more competent at tackling HITs related to that category (e.g., movie genre classification). In our system, workers and HITs are matched based on an underlying taxonomy that is defined on categories extracted both from the tasks at hand and from the work-

Copyright is held by the International World Wide Web Conference Committee (IW3C2). IW3C2 reserves the right to provide a hyperlink to the author's site if the Material is used in electronic media.
WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

[1] `http://www.mturk.com`

ers' interests. Entities appearing in the users' social profiles are linked to the Linked Open Data (LOD) cloud[2], where they are then matched to related tasks that are available on the crowdsourcing platform. We experimentally evaluate our pull methodology and compare it against traditional crowdsourcing approaches using tasks of varying types and complexity. Results show that the quality of the answers is significantly higher when using a *push* methodology.

In summary, the contributions of this paper are:

- a novel Crowdsourcing framework that focuses on *pushing* HITs to the crowd;
- a software architecture that implements the newly proposed *push* crowdsourcing methodology;
- category-based, text-based, and graph-based approaches to assign HITs to workers based on links in the LOD cloud;
- an empirical evaluation of our method in a real deployment over different crowds showing that our Pick-A-Crowd system is on average 29% more effective than traditional pull crowdsourcing platforms over a variety of HITs.

The rest of this paper is structured as follows: We review the state of the art in Crowdsourcing, Recommender Systems, and Expert Finding in Section 2. Section 3 gives an overview of the architecture of our system, including its HIT publishing interface, its crowd profiling engine, and its HIT assignment and reward estimation components. We introduce our formal model to match human workers to HITs using category-based, text-based, and graph-based approaches in Section 4. We describe our evaluation methodology and discuss results we obtained from a real deployment of our system in Section 5, before concluding in Section 6.

2. RELATED WORK

Crowdsourcing.

Crowdsourcing has recently gained much attention from different research communities. In the database community, hybrid human-machine database systems have been proposed [12, 23]. In those systems, crowdsourcing is for example used to explain null values in returned results, or to define subjective 'ORDER BY' operators that allow to express queries such as 'I want pictures for motivational slides'. In the Information Retrieval community, crowdsourcing has been mainly used for effectiveness evaluation purposes. Relevance judgements used to evaluate the results of a search engine can now be created by asking the crowd about the relevance of a document or, more generally, of a retrieved resource. Crowdsourcing can be used to produce relevance judgements for documents [2], books [16, 17], or entities [5]. In the Semantic Web community, crowdsourcing has also been recently considered, for instance to link [10] or map [21] entities. In both cases, the use of crowdsourcing can significantly improve the quality of generated links or mappings as compared to purely automatic approaches. In the context of Natural Language Processing, games to crowdsource the Word Sense Disambiguation task[22] have recently been proposed.

[2]`http://linkeddata.org/`

An incentive that if often leveraged to get input from the crowd is fun. Games with a purpose have studied how to design entertaining applications that can generate useful data to be processed by further algorithms. An example of a successful game that at the same time generates meaningful data is the ESP game [25] where two human players have to agree on the words to use to tag a picture. An extension of this game is Peekaboom: a game that asks the player to detect and annotate specific objects within an image [26]. A successful crowdsourcing application is Recaptcha [27], which generates captcha codes that human users have to type on their machines and which contain scanned words (from books) that would be otherwise complex to identify by means of automated OCR approaches. Thus, by entering valid captcha codes, human users help to digitalize a large amount of textual content only available on paper.

Designing such games makes high-quality data available at no cost. Another incentive that is often simpler to put in place is the financial incentive: A small monetary reward is granted to workers who perform a simple task online. While in this way it gets easier to recruit a larger crowd, this raises questions about the quality of the results. Indeed, quality control is a common issue of paid crowdsourcing platforms, given the presence of malicious workers whose intent is to game the system to obtain the monetary reward without properly completing the task. Approaches to control quality by identifying low-quality workers and, thus, stopping them from performing further work have already been proposed [16, 10]. Such approaches use either majority agreement or a set of known answers to check for errors and to identify workers who make many mistakes. However, such approaches require workers to complete several tasks in order to be evaluated. Instead, our system is able to model workers *before* they complete any HIT on the platform and thus assign tasks exclusively to those who are known to show interest in the task.

A first attempt to crowdsource micro-tasks on top of social networks has been proposed by [11], where authors describe a framework to post questions as tweets that users can solve by tweeting back an answer. As compared to this early approach, we propose a more controlled environment where workers are known and profiled in order to push tasks to selected social network users.

Crowdsourcing over social networks is also used by CrowdSearcher [6, 7, 8], which improves automatic search systems by means of asking questions to personal contacts. The crowdsourcing architecture proposed in [9] considers the problem of assigning tasks to selected workers. However, authors do not evaluate automatic assignment approaches but only let the requesters manually select individual workers, which they want to push the task to. In this paper instead, we assess the feasibility and effectiveness of automatically mapping HITs to workers based on their social network profiles.

Also related to our system is the study of trust in social networks. Golbeck [13], for instance, proposes different models to rank social network users based on trust and applies them to recommender systems as well as other end-user applications.

Recommender Systems.

Assigning HITs to workers is similar to the task performed by recommender systems (e.g., recommending movies to buy

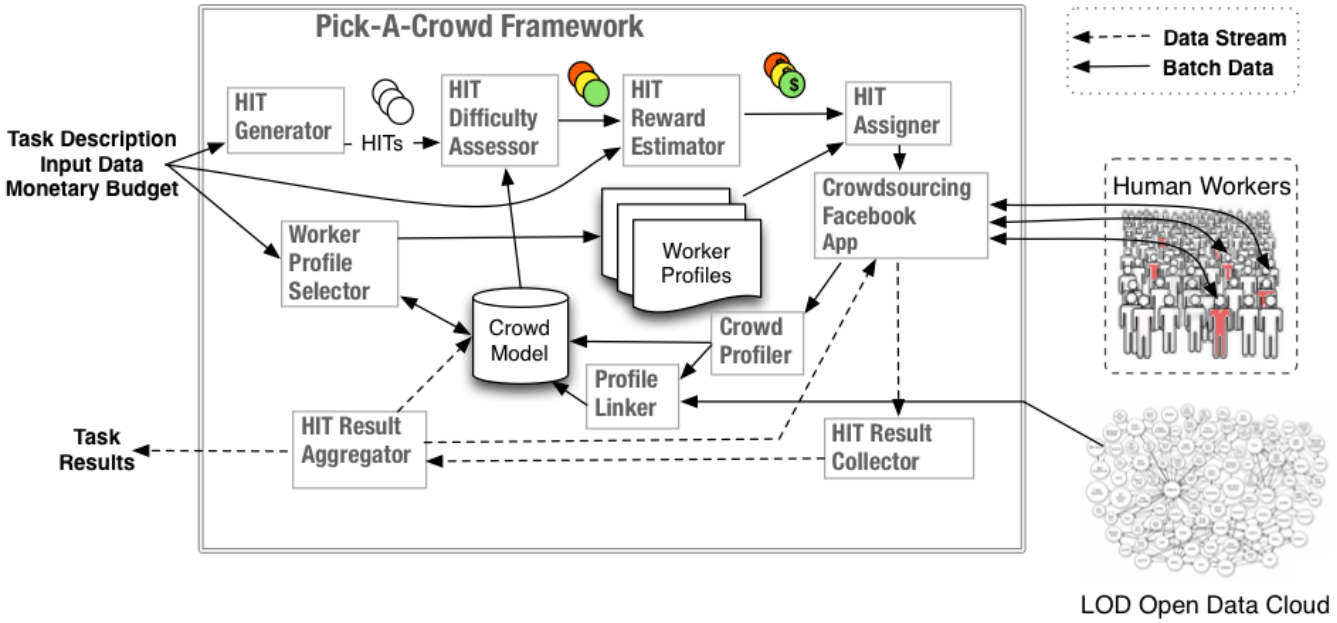

Figure 1: Pick-A-Crowd Component Architecture. Task descriptions, Input Data, and a Monetary Budget are taken as input by the system, which creates HITs, estimates their difficulty and suggests a fair reward based on the skills of the crowd. HITs are then pushed to selected workers and results get collected, aggregated, and finally returned back to the requester.

to potential customers). We can categorize recommender systems into content-based and collaborative filtering approaches. The former approaches exploit the resources contents and match them to user interests. The latter ones only use similarity between user profiles constructed out of their interests (see [19] for a survey). Recommended resources are those already consumed by similar users. Our systems adopts techniques from the field of recommender systems as it aims at matching HITs (i.e., tasks) to human workers (i.e., users) by constructing profiles that describe worker interests and skills. Such profiles are then matched to HIT descriptions that are either provided by the task requester or by analyzing the questions and potential answers included in the task itself (see Section 4). Recommender systems built on top of social networks already exits. For example, in [1], authors propose a news recommendation system for social network groups based on community descriptions.

Expert Finding.

In order to push tasks to the right worker in the crowd, our system aims at identifying the most suitable person for a given task. To do so, our Worker Profile Selector component generates a ranking of candidate workers who can be contacted for the HIT. This is highly related to the task of Expert Finding studied in Information Retrieval. The Enterprise track at the TREC evaluation initiative[3] has constructed evaluation collections for the task of expert finding within an organizational setting [4]. The studied task is that of ranking candidate experts (i.e., employees of a company) given a keyword query describing the required expertise. Many approaches have been proposed for such tasks (see [3] for a comprehensive survey). We can classify most of them as either document-based, when document ranking is performed before identifying the experts, or as candidate-based, when expert profiles are first constructed before being ranked given a query. Our system follows the former approach by ranking online social network pages and using them to assign work to the best matching person.

[3]http://trec.nist.gov

3. SYSTEM ARCHITECTURE

In this section, we describe the Pick-A-Crowd framework and provide details on each of its components.

3.1 System Overview

Figure 1 gives a simplified overview of our system. Pick-a-Crowd receives as input tasks that need to be completed by the crowd. The tasks are composed of a textual description, which can be used to automatically select the right crowd for the task, actual data on which to run the task (e.g., a Web form and set of images with candidate labels), as well as a monetary budget to be spent to get the task completed. The system then creates the HITs, and predicts the difficulty of each micro-task based on the crowd profiles and on the task description. The monetary budget is split among the generated micro-tasks according to their expected difficulty (i.e., a more difficult task will be given a higher reward). The HITs are then assigned to selected workers from the crowd and published on the social network application. Finally, answers are processed as a stream from the crowd, aggregated and sent back to the requester.

We detail the functionalities provided by each component of the system in the following.

3.2 HIT Generation, Difficulty Assessment, and Reward Estimation

The first pipeline in the system is responsible for generating the HITs given some input data provided by the requester. HITs can for instance be generated from i) a Web template to classify images in pre-defined categories, together with ii) a set of images and iii) a list of pre-defined categories. The HIT Generator component dynamically creates as many tasks as required (e.g., one task per image to categorize) by combining those three pieces of information.

Next, the HIT Difficulty Assessor takes each HIT and determines a complexity score for it. This score is computed based on both the specific HIT (i.e., description, keywords, candidate answers, etc.) and on the worker profiles (see Section 4 for more detail on how such profiles are constructed). Different algorithms can be implemented to assess the difficulty of the tasks in our framework. For example, a text-based approach can compare the textual description of the task with the skill description of each worker and compute a score based on how many workers in the crowd could perform well on such HITs.

An alternative a more advanced prediction method can exploit entities involved in the task and known by the crowd. Entities are extracted from the textual descriptions of the tasks and disambiguated to LOD entities. The same can be performed on the worker profiles: each Facebook page that is liked by the workers can be linked to its respective LOD entities. Then the set of entities representing the HITs and the set of entities representing the interests of the crowd can be directly compared. The task is classified as difficult when the entities involved in the task heavily differ from the entities liked by the crowd.

A third example of task difficulty prediction method is based on Machine Learning. A classifier assessing the task difficulty is trained by means of previously completed tasks, their description and their result accuracy. Then, the description of a new task is given as a test vector to the classifier, which returns the predicted difficulty for the new task.

Finally, the Reward Estimation component takes as input a monetary budget B and the results of the HIT assessment to determine a reward value for each HIT h_i.

A simple way to redistribute the available monetary budget is simply by rewarding the same amount of money for each task of the same type. An second example of reward estimation function is:

$$reward(h_i) = \frac{B \cdot d(h_i)}{\sum_j d(h_j)} \quad (1)$$

which takes into account the difficulty $d()$ of the HIT h_i as compared to the others and assigns a higher reward to more difficult tasks.

A third approach computes a reward based on both the specific HIT as well as the worker who performs it. In order to do this, we can exploit the HIT assignment models adopted by our system. These models generate a ranking of workers by means of computing a function $match(w_j, h_i)$ for each worker w_j and HIT h_i (see Section 4). Given such a function, we can assign a higher reward to better suited workers by

$$reward(h_i, w_j) = \frac{B \cdot match(w_j, h_i)}{\sum_{k,l} match(w_k, h_l)} \quad (2)$$

More advanced reward schemes can be applied as well. For example, in [15], authors propose game theoretic based approaches to compute the optimal reward for paid crowdsourcing incentives in the presence of workers who collude in order to game the system.

Exploring and evaluating different difficulty prediction and reward estimation approaches is not the focus of this paper and is left as future work.

3.3 Crowd Profiler

The task of the Crowd Profiler component is to collect information about each available worker in the crowd. Pick-A-Crowd uses contents available on the social network platform as well as previously completed HITs to construct the workers' profiles. Those profiles contain information about the skills and interests of the workers and are used to match HITs with available workers in the crowd.

In detail, this module generates a set of worker profiles $C = \{w_1, .., w_n\}$ where $w_i = \{P, T\}$, P is the set of worker interests (e.g., when applied on top of the Facebook platform $p_i \in P$ are the Facebook pages the worker likes) and $T_i = \{t_1..t_n\}$ is the set of tasks previously completed by w_i. Each Facebook page p_i belongs to a category in the Facebook Open Graph[4].

3.4 Worker Profile Linker

This component is responsible for linking each Facebook page liked by some worker to the corresponding entity in the LOD cloud. Given the page name and, possibly, a textual description of the page, the task is defined as identifying the correct URI among all the ones present in the LOD graph using, for example, a similarity measure based on adjacent nodes in the graph. This is a well studied problem where both automatic [14] or crowdsourcing-based techniques [10] can be used.

3.5 Worker Profile Selector

HITs and workers are matched based on the profiles described above. Intuitively, a worker who only likes many music bands will not be assigned a task that asks him/her to identify who is the movie actor depicted in the displayed picture. The similarity measure used for matching workers to tasks takes into account the entities included in the workers' profiles but is also based on the Facebook categories their liked pages belong to. For example, it is possible to use the corresponding DBPedia entities and their YAGO type. The YAGO knowledge-base provides a fine-grained high-accuracy entity type categorization which has been constructed by combining Wikipedia category assignments with WordNet synset information. The YAGO type hierarchy can help the system better understand which type of entity correlates with the skills required to effectively complete a HIT (see also Section 4 for a formal definition of such methods). For instance, our graph-based approach concludes that for our music related task, the top Facebook pages that indicate expertise on the topic are 'MTV' and 'Music & top artists'.

A generic similarity measure to match workers and task

[4] `https://developers.facebook.com/docs/opengraph/`

is in equation 3

$$sim(w_j = \{P, T\}, h_i = \{t, d, A, Cat\}) = \frac{\sum_{k,l} sim(p_k, a_l)}{|P| \cdot |A|} \quad \forall p_k \in P, a_l \in A \tag{3}$$

where A is the set of candidate answers for task h_i and $sim()$ measures the similarity between the worker profile and the task description.

3.6 HIT Assigner and Facebook App

The HIT Assigner component takes as input the final HITs with the defined reward and publishes them onto the Facebook App. We developed a dedicated, native Facebook App called OpenTurk[5] to implement this final component of the Pick-A-Crowd platform. Figure 2 shows a few screenshots of OpenTurk. As any other application on the Facebook platform, it has access to several pieces of information about the users that accept to use it. We follow a non-intrusive approach; In our case, the liked pages for each user are stored in an external database that is used to create a worker profile containing his/her interests. The application we developed also adopts crowdsourcing incentive schemes different than the pure financial one. For example, we use the *fan* incentive where a competition involving several workers competing on trivia questions on their favorite topic can be organized. The app also allows to directly challenge other social network contacts by sharing the task, which is also helpful to enlarge the application user base. While from the worker point of view this represents a friendly challenge, from a platform point of view this means that the HIT will be pushed to another expert worker, following the assumption that a worker would challenge someone who is also knowledgeable about the topic addressed by the task.

3.7 HIT Result Collector and Aggregator

The final pipeline is composed of stream processing modules, where the Facebook App answers are being streamed from the crowd to the answer creation pipeline. The first component collects the answers from the crowd and is responsible for a first quality check based on potentially available gold answers for a small set of training questions. Then, answers that are considered to be valid (based on available ground-truth data) are forwarded to the HIT Result Aggregator component, which collects and aggregates them in the final answer for the HIT. When a given number of answers has been collected (e.g., five answers), then the component outputs the partial aggregated answer (e.g., based on majority vote) back to the requester. As more answers reach the aggregation component, the aggregated answer presented to the requester gets updated. Additionally, as answers are collected, the workers' profiles get updated and the reward gets granted to the workers who performed the task through the Facebook App.

4. HIT ASSIGNMENT MODELS

In this section, we define the HIT assignment tasks and describe several approaches for assigning workers to such tasks. We focus on HIT assignment rather than on other system components as the ability to assign tasks automatically is the most original feature of our system as compared to other crowdsourcing platforms.

[5]http://apps.facebook.com/openturk/

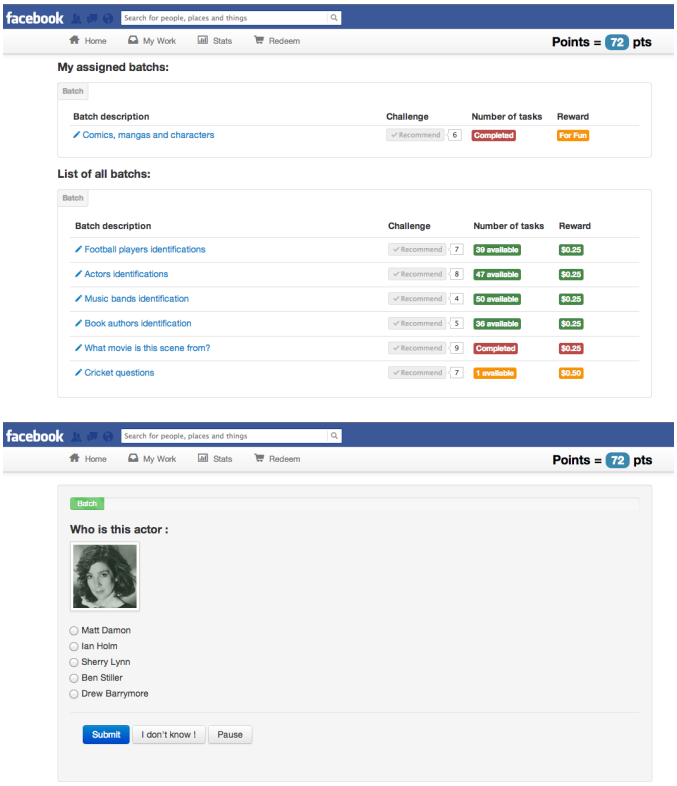

Figure 2: Screenshots of the OpenTurk Facebook App. Above, the dashboard displaying HITs assigned to a specific worker. Below, a HIT about actor identification assigned to a worker who likes several actors.

Given a HIT $h_i = \{t_i, d_i, A_i, Cat_i\}$ from the requester, the task of assigning it to some workers is defined as ranking all available workers $C = \{w_1, .., w_n\}$ on the platform and selecting the top-n ranked workers. A HIT consists of a textual description t_i (e.g., the task instruction which is being provided to the workers)[6], a data field d_i that is used to provide the context for the task to the worker (e.g., the container for an image to be labelled), and, optionally, the set of candidate answers $A_i = \{a_1, .., a_n\}$ for the multiple-choices tasks (e.g, a list of music genres used to categorize a singer) and a list of target Facebook categories $Cat_i = \{c_1, ..c_n\}$. A worker profile $w_j = \{P, T\}$ is assigned a score based on which it is ranked for the task h_i. This score is determined based on the likelihood of matching w_j to h_i. Thus, the goal is to define a scoring function $match(w_j, h_i)$ based on the worker profile, the task description and, possibly, external resources such as the LOD datasets or a taxonomy.

4.1 Category-based Assignment Model

The first approach we define to assign HITs to workers is based on the same idea that Facebook uses to target adver-

[6]When applied to hybrid human-machine systems t_i can be defined as the *data context* of the HIT. For example, in crowdsourced databases t_i can be the name of the column, table, etc. the HIT is about.

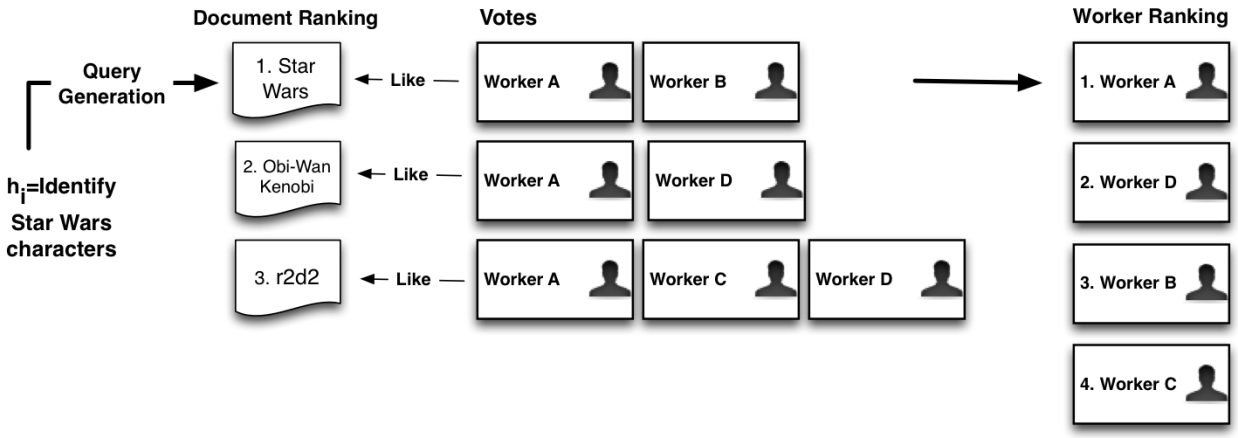

Figure 3: An example of the Expert Finding Voting Model. The final ranking identifies worker A as the top worker as he likes the most pages related to the query.

tisements to its users. A requester has to select the target community of users who should perform the task by means of selecting one or more Facebook pages or page categories (in the same way as someone who wants to place an ad). Such categories are defined in a 2 levels structure with 6 top levels (e.g., "Entertainment", "Company"), each of them having several sub-categories (e.g., "Movie", "Book", "Song", etc. are sub-categories of "Entertainment").

Once some second-level categories are selected by the requester, the platform can generate a ranking of users based on the pages they like. More formally, given a set of target categories $Cat = \{c_1, ..c_n\}$ from the requester, we define $P(c_i) = \{p_1, .., p_n\}$ as the set of pages belonging to category c_i. Then, for each worker $w_j \in C$ we take the set of pages he/she likes $P(w_j)$ and measure its intersection with the pages belonging to any category selected by the requester $RelP = \cup_i P(c_i)$. Thus, we can assign a score to the worker based on the overlap between the likes and the target category $|P(w_j) \cap RelP|$ and rank all $w_j \in C$ based on such scores.

4.2 Expert Profiling Assignment Model

A second approach we propose to rank workers given a HIT h_i is to follow an expert finding approach. Specifically, we define a scoring function based on the Voting Model for expert finding [18]. For the HIT we want to assign, we take the set of its candidate answers A_i, when available. Then, we define a disjunctive keyword query based on all the terms composing the answers $q = \wedge_i a_i$. In case A_i is not available, for example because the task is asking an open-ended question, then q can be extracted out of t_i by mining entities mentioned in its content. The query q is then used to rank Facebook pages using an inverted index built over the collection of documents $\cup_i P_i \; \forall w_j \in C$. We consider each ranked page as a vote for the workers who like them on Facebook and rank workers accordingly. That is, if $RetrP$ is the set of pages retrieved with q, we can define a worker ranking function as $|P(w_j) \cap RetrP|$. More interestingly, we can take into account the ranking generated by q and give a higher score to workers liking pages that were ranked higher. An example of how to rank workers following the voting model is depicted in Figure 3.

4.3 Semantic-Based Assignment Model

The third approach we propose is based on third-party information. Specifically, we first link candidate answers and pages to an external knowledge base (e.g., DBPedia) and exploit its structure to better assign HITs to workers. For a given HIT h_i, the first step is to identify the entity corresponding to each $a_j \in A_i$ (if A_i is not available, entities in t_i can be used instead). This task is related to entity linking [10] and ad-hoc object retrieval [20, 24] where the goal is to find the correct URI for a description of the entity using keywords. In this paper, we take advantage of state-of-the-art techniques for this task but do not focus on improving over such techniques. Then, we identify the entity that represents each page liked by the crowd whenever it exists in the knowledge base. Once both answers and pages are linked to their corresponding entity in the knowledge base, we exploit the underlying graph structure to determine the extent to which entities that describe the HIT and entities that describe the interests of the worker are similar. Specifically, we define two scoring methods based on the graph.

The first scoring method takes into account the vicinity of the entities in the entity graph. We measure how many worker entities are directly connected to HIT entities using SPARQL queries over the knowledge base as follows:

```
SELECT ?x
WHERE { <uri(a_i)> ?x <uri(p_i)> }.
```

This follows the assumption that a worker who likes a page is able to answer questions about related entities. For example, if a worker likes the page 'FC Barcelona', then he/she might be a good candidate worker to answer a question about 'Lionel Messi' who is a player of the soccer team liked by the worker.

Our second scoring function is based on the type of entities. We measure how many worker entities have the same type as the HIT entity using SPARQL queries over the knowledge base as follows:

```
SELECT ?x
WHERE { <uri(a_i)> <rdf:type> ?x .
        <uri(p_i)> <rdf:type> ?x
      }.
```

The underlying assumption in that case is that a worker who likes a page is able to answer questions about entities of the same type. For example, if a worker likes the pages 'Tom Hanks' and 'Julia Roberts', then he/she might be a good candidate worker to answer a question about 'Meg Ryan' as it is another entity of the same type (i.e., actor).

5. EXPERIMENTAL EVALUATION

Given that the main innovation of Pick-A-Crowd as compared to classic crowdsourcing platforms such as AMT is the ability to push HITs to workers instead of letting the workers select the HITs they wish to work on, we focus in the following on the evaluation and comparison of different HIT assignment techniques and compare them in terms of work quality against a classic crowdsourcing platform.

5.1 Experimental Setting

The Facebook app OpenTurk we have implemented within the Pick-A-Crowd framework currently counts more than 170 workers who perform HITs requiring to label images containing popular or less popular entities and to answer open-ended questions. Overall, more than 12K distinct Facebook pages liked by the workers have been crawled over the Facebook Open Graph. OpenTurk is implemented using cloud-based storage and processing back-end to ensure scalability with an increasing number of workers and requesters. OpenTurk workers have been recruited via AMT, thus making a direct experimental comparison to standard AMT techniques more meaningful.

The type of task categories we evaluate our approaches on are: actors, soccer players, anime characters, movie actors, movie scenes, music bands, and questions related to cricket. Our experiments cover both multiple answer questions as well as open-ended questions: Each task category includes 50 images for which the worker either has to select the right answer among 5 candidate answers or to answer 20 open-ended questions related to the topic. Each type of question can be skipped by the worker in case he/she has no idea about that particular topic.

In order to analyze the performance of workers in the crowd, we measure Precision, Recall (as the worker is allowed to skip questions when he/she does not know the answer), and Accuracy of their answers for each HIT obtained via majority vote over 3 and 5 workers[7].

5.2 Motivation Examples

As we can see from Figure 4, the HITs that asks questions about cricket clearly show how workers can perform differently in terms of accuracy. There are 13 workers out of 35 who were not able to provide any correct answer while the others spread over the Precision/Recall spectrum, with the best worker performing at 0.9 Precision and 0.9 Recall. This example motivates the need to selectively assign the HIT to the most appropriate worker and not following a first-come-first-served approach as proposed, for example, by AMT.

[7]The set of HITs and correct answers we used in our experiments are available for comparative studies online at: `http://exascale.info/PickACrowd`

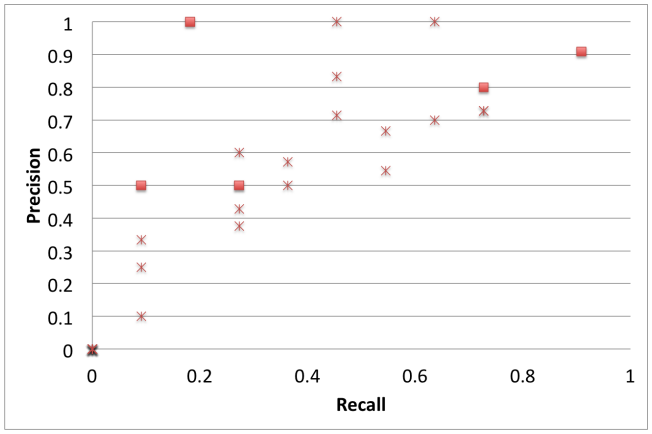

Figure 4: Crowd performance on the cricket task. Square points indicate the 5 workers selected by our graph-based model that exploits entity type information.

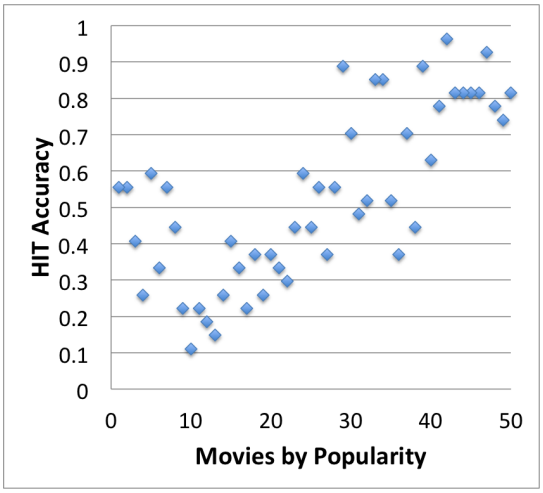

Figure 5: Crowd performance on the movie scene recognition task as compared to movie popularity.

Thus, the goal of Pick-A-Crowd is to adopt HIT assignment models that are able to identify the workers in the top-right area of Figure 4, based solely on their social network profile. As an anecdotal observation, a worker from AMT provided as feedback to the cricket task in the available comment field the following comment "I had no idea what to answer to most questions..." which clearly demonstrates that for the tasks requiring background knowledge, not all workers are a good fit.

An interesting observation is the impact of the popularity of a question. Figure 5 shows the correlation between task accuracy on the movie scene recognition task and the popularity of the movie based on the overall number of Facebook likes on the IMDB movie page. We can observe that when a movie is popular, then workers easily recognize it. On the other hand, when a movie is not so popular it becomes more difficult to find knowledgeable workers for the task.

5.3 OpenTurk Crowd Analysis

Figure 6 shows some statistics about the user base of OpenTurk. The majority of workers are in the age interval

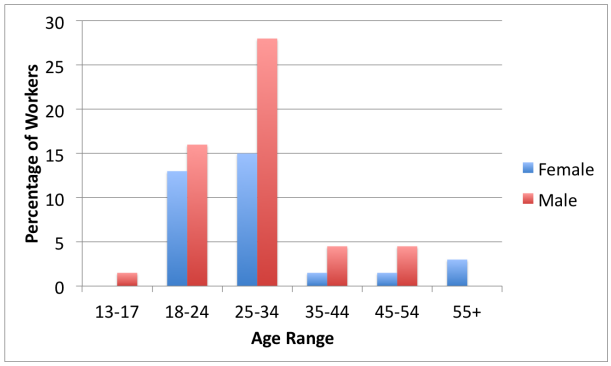

Figure 6: OpenTurk Crowd age distribution.

25-34 and are from the United States. Another interesting observation can be made about the Facebook Notification click rate. Once the Pick-A-Crowd system selects a worker for a HIT, the Facebook app OpenTurk sends a notification to the worker with information about the newly available task and its reward. Figure 7 shows a snapshot of the notifications clicked by workers as compared to the notification sent by OpenTurk over a few days. We observe an average rate of 57% clicks per notification sent.

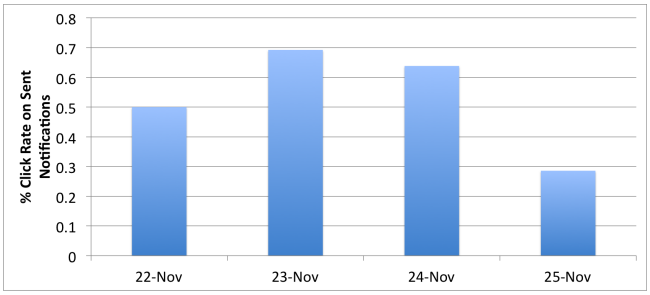

Figure 7: OpenTurk Notification click rate.

A third analysis looks on how the relevant likes of a worker correlates with his/her accuracy for the task. Figure 8 shows a distribution of worker accuracy over the relevant pages liked using the category-based HIT assignment model to define the relevance of pages. In a first look, we do not see a perfect correlation between the number of likes and the worker accuracy for any task. On the other hand, we observe that when many relevant pages are in the worker profile (e.g., >30), then accuracy tends to be high (i.e., the bottom-right part of the plot is empty). However, when only a few relevant pages belong to the worker profile, then it becomes difficult to predict his/her accuracy. Note that not-liking relevant pages is not an indication of being unsuitable for a task: Having an incomplete profile just does not allow to model the worker and to assign him/her the right tasks (i.e., the top-left part of the plot contains high-accuracy workers with incomplete profiles). Having worker profiles containing several relevant pages is not problematic when the crowd is large enough (as it is on Facebook).

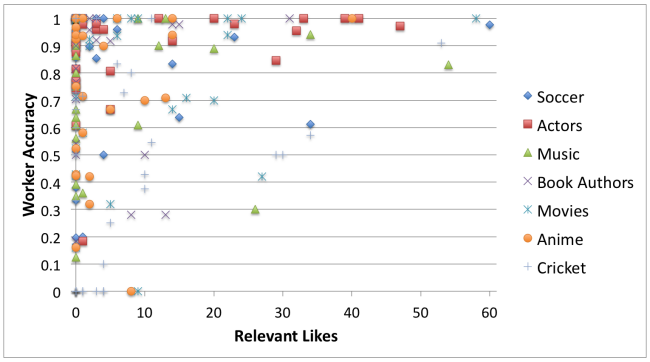

Figure 8: OpenTurk Crowd Accuracy as compared to the number of relevant Pages a worker likes.

Table 1: A comparison of the task accuracy for the AMT HIT assignment model assigning each HIT to the first 3 and 5 workers and to Amazon MTurk Masters.

Task	AMT 3	AMT 5	AMT Masters 3
Soccer	0.8	0.8	0.1
Actors	0.82	0.82	0.9
Music	0.76	0.7	0.7
Book Authors	0.7	0.5	0.58
Movies	0.6	0.64	0.66
Anime	0.94	0.86	0.1
Cricket	0.004	0	0.72

5.4 Evaluation of HIT Assignment Models

In the literature, common crowdsourcing tasks usually adopt 3 or 5 assignments of the same HIT in order to aggregate the answers from the crowd, for example by majority vote. In the following, we compare different assignment models evaluating both the cases where 3 and 5 assignments are considered for a given HIT. As a baseline, we compare against the AMT model that assigns the HIT to the first n workers performing the task. We also compare against AMT Masters who are workers being awarded a special status by Amazon based on their past performances[8]. Our proposed models first rank workers in the crowd based on their estimated accuracy and then assign the task to the top-3 or top-5 workers.

Table 1 presents an overview of the performances of the assignment model used by AMT. We observe that while on average there is not a significant difference between using 3 or 5 workers, Masters perform better than the rest of the AMT crowd on some tasks but do not outperform the crowd on average (0.54 versus 0.66 Accuracy). A per-task analysis shows that some tasks are easier than others: While tasks about identifying pictures of popular actors obtain high accuracy for all three experiments, topic-specific tasks such as cricket questions may lead to a very low accuracy.

Table 2 gives the results we obtained by assigning tasks based on the Facebook Open Graph categories manually selected by the requester. We observe that the Soccer and

[8]Note that to be able to recruit enough Masters for our tasks we had to reward $1.00 per task as compared to $0.25 granted to standard workers.

Table 2: A comparison of the effectiveness for the category-based HIT assignment models assigning each HIT to 3 and 5 workers with manually selected categories.

Task	Requester Selected Categories	Category-based 3	Category-based 5
Soccer	Sport,Athlete,Public figure	0.94	0.98
Actors	Tv show, Comedian, Movie, Artist, Actor/director	0.94	0.96
Music	Musician/band,Music	0.96	0.96
Book Authors	Author,Writer,Book	0.98	0.94
Movies	Movie,Movie general,Movies/music	0.44	0.74
Anime	Games/toys,Entertainment	0.62	0.7
Cricket	Sport,Athlete,Public figure	0.63	0.54

Table 3: Effectiveness for different HIT assignments based on the Voting Model assigning each HIT to 3 and 5 workers and querying the Facebook Page index with the task description $q = t_i$ and with candidate answers $q = A_i$ respectively.

Task	VotingModel $q = t_i$ 3	VotingModel $q = t_i$ 5		VotingModel $q = A_i$ 3	VotingModel $q = A_i$ 5
Soccer	**0.92**	**0.92**		0.86	0.86
Actors	0.92	**0.94**		0.92	0.88
Music	**0.96**	**0.96**		0.76	0.78
Book Authors	0.94	**0.96**		0.3	0.84
Movies	**0.70**	0.60		**0.70**	0.42
Anime	0.54	**0.84**		0.56	0.54
Cricket	0.63	**0.72**		**0.72**	**0.72**

Table 4: Effectiveness for different HIT assignments based on the entity graph in the DBPedia knowledge base assigning each HIT to 3 and 5 workers.

Task	En. type 3	En. type 5	1-step 3	1-step 5
Soccer	**0.98**	0.92	0.86	0.86
Actors	**0.92**	**0.92**	**0.92**	0.90
Music	0.62	**0.68**	0.64	0.54
Book Authors	0.28	0.50	0.50	**0.82**
Movies	0.70	**0.78**	0.46	0.62
Anime	0.46	**0.90**	0.62	0.62
Cricket	0.63	**0.82**	0.63	0.63

Cricket tasks have been assigned to the same Facebook category which does not distinguish between different types of sports. Anyhow, we can see that for the cricket task the category-based method does not perform well, as the pages contained into the categories cover many different sports and, according to our crowd at least, soccer-related tasks are simpler than cricket-related tasks.

Table 3 presents the results when assigning HITs following the Voting Model for expert finding.

We observe that in the majority of cases, assigning each task to 5 different workers selected using the Facebook Page indexing the task description as query leads to the best results.

Table 4 shows the results of our graph-based approaches. We observe that in the majority of these cases, the graph-based approach that follows the entity type ("En. type") edges and selects workers who like Pages of the same type as the entities involved in the HIT outperforms the approach that considers the directly-related entities within one step in the graph ("1-step").

Table 5: Average Accuracy for different HIT assignment models assigning each HIT to 3 and 5 workers.

Assignment Method	Average Accuracy
AMT 3	0.66
AMT 5	0.62
AMT Masters 3	0.54
Category-based 3	0.79
Category-based 5	0.83
Voting Model t_i 3	0.80
Voting Model t_i 5	**0.85**
Voting Model A_i 3	0.69
Voting Model A_i 5	0.72
En. type 3	0.66
En. type 5	0.79
1-step 3	0.66
1-step 5	0.71

5.5 Comparison of HIT Assignment Models

Table 5 presents the average Accuracy obtained over all the HITs in our experiments (which makes a total of 320 questions) by each HIT assignment model. As we can see, our proposed HIT assignment models outperform the standard first-come-first-served model adopted by classic crowdsourcing platforms such as Amazon MTurk. On average over the evaluated tasks, the best performing model is the one based on the Voting Model defined for the expert finding problem where pages relevant to the task are seen as votes for the expertise of the workers. Such an approach ob-

tains on average a 29% relative improvement over the best accuracy obtained by the AMT model.

6. CONCLUSIONS

Crowdsourcing is a popular new means of performing large-scale, high-quality Human Intelligence tasks. The long-term vision of current crowdsourcing research is to create hybrid human-machine systems capable of chimeric functionalities, which were not conceivable just few years ago. One of the key impediments towards that vision is to obtain high-quality answers from the crowd. This is currently an open-issue, given the way crowdsourcing tasks are commonly advertised and run today: Tasks are posted on simple platforms where the first worker who is available will perform the task. This simplistic task allocation procedure represents a clear threat towards more qualitative results, especially when we consider paid crowdsourcing tasks where the monetary reward granted to workers who complete tasks attracts malicious people willing to earn money without spending too much time and efforts on the actual task.

For all these reasons, we proposed in this paper Pick-A-Crowd, a system exploiting a novel crowdsourcing scheme focusing on pushing tasks to the right worker rather than letting the workers pull the tasks they wished to work on. We proposed a novel crowdsourcing architecture that builds worker profiles based on their online social network activities and tries to understand the skills and interests of each worker. Thanks to such profiles, Pick-A-Crowd is able to assign each task to the right worker dynamically.

To demonstrate and evaluate our proposed architecture, we have developed an deployed OpenTurk, a native Facebook application that pushes crowdsourced tasks to selected workers and collects the resulting answers. We additionally proposed and extensively evaluated HIT assignment models based on 1) Facebook categories manually selected by the task requester, 2) methods adapted from an expert finding scenario in an enterprise setting, and 3) methods based on graph structures borrowed from external knowledge bases. Experimental results over the OpenTurk user-base show that all of the proposed models outperform the classic first-come-first-served approach used by standard crowdsourcing platforms such as Amazon Mechanical Turk. Our best approach provides on average 29% better results than the Amazon MTurk model.

A potential limitation of our approach is that it may lead to longer task completion times: While on pull crowdsourcing platforms the tasks gets completed quickly (since any available worker can perform the task), following a push methodology may lead to delays in the completion of the tasks. We anyway believe that this would be an acceptable tradeoff, as crowdsourcing focuses on obtaining high-quality answers rather than real-time data from the crowd.

As future steps, we would like to test various HIT assignment models based on Machine Learning techniques where, given a few tasks with known answers used as training data, specific worker features (e.g., education level) could be learnt and leveraged to match tasks to workers with a high accuracy.

Acknowledgments.

We thank the anonymous reviewers for their helpful comments. This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459.

7. REFERENCES

[1] M. Agrawal, M. Karimzadehgan, and C. Zhai. An online news recommender system for social networks. In *Proceedings of ACM SIGIR workshop on Search in Social Media*, 2009.

[2] O. Alonso and R. A. Baeza-Yates. Design and Implementation of Relevance Assessments Using Crowdsourcing. In *ECIR*, pages 153–164, 2011.

[3] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, and L. Si. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(2-3):127–256, 2012.

[4] K. Balog, P. Thomas, N. Craswell, I. Soboroff, P. Bailey, and A. De Vries. Overview of the trec 2008 enterprise track. Technical report, DTIC Document, 2008.

[5] R. Blanco, H. Halpin, D. Herzig, P. Mika, J. Pound, H. S. Thompson, and D. T. Tran. Repeatable and reliable search system evaluation using crowdsourcing. In *SIGIR*, pages 923–932, 2011.

[6] A. Bozzon, M. Brambilla, and S. Ceri. Answering search queries with CrowdSearcher. In *WWW*, pages 1009–1018, New York, NY, USA, 2012. ACM.

[7] A. Bozzon, M. Brambilla, S. Ceri, and A. Mauri. Extending search to crowds: A model-driven approach. In *SeCO Book*, pages 207–222. 2012.

[8] A. Bozzon, M. Brambilla, and A. Mauri. A model-driven approach for crowdsourcing search. In *CrowdSearch*, pages 31–35, 2012.

[9] A. Bozzon, I. Catallo, E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. A framework for crowdsourced multimedia processing and querying. In *CrowdSearch*, pages 42–47, 2012.

[10] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, New York, NY, USA, 2012.

[11] E. Diaz-Aviles and R. Kawase. Exploiting twitter as a social channel for human computation. In *CrowdSearch*, pages 15–19, 2012.

[12] A. Feng, M. J. Franklin, D. Kossmann, T. Kraska, S. Madden, S. Ramesh, A. Wang, and R. Xin. CrowdDB: Query Processing with the VLDB Crowd. *PVLDB*, 4(11):1387–1390, 2011.

[13] J. A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, College Park, MD, USA, 2005. AAI3178583.

[14] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *SIGIR*, pages 765–774, New York, NY, USA, 2011. ACM.

[15] R. Jurca and B. Faltings. Mechanisms for making crowds truthful. *J. Artif. Intell. Res. (JAIR)*, 34:209–253, 2009.

[16] G. Kazai. In Search of Quality in Crowdsourcing for Search Engine Evaluation. In *ECIR*, pages 165–176, 2011.

[17] G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. In *SIGIR*, pages 205–214, 2011.

[18] C. Macdonald and I. Ounis. Voting techniques for expert search. *Knowl. Inf. Syst.*, 16(3):259–280, 2008.

[19] S. Perugini, M. A. Gonçalves, and E. A. Fox. Recommender systems research: A connection-centric survey. *J. Intell. Inf. Syst.*, 23(2):107–143, Sept. 2004.

[20] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. In *WWW*, pages 771–780, 2010.

[21] C. Sarasua, E. Simperl, and N. F. Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *ISWC*, pages 525–541, 2012.

[22] N. Seemakurty, J. Chu, L. von Ahn, and A. Tomasic. Word sense disambiguation via human computation. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 60–63, New York, NY, USA, 2010. ACM.

[23] J. Selke, C. Lofi, and W.-T. Balke. Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. *Proc. VLDB Endow.*, 5(6):538–549, Feb. 2012.

[24] A. Tonon, G. Demartini, and P. Cudre-Mauroux. Combining inverted indices and structured search for ad-hoc object retrieval. In *SIGIR*, pages 125–134, 2012.

[25] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, Aug. 2008.

[26] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 55–64, New York, NY, USA, 2006. ACM.

[27] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.