Chapter 2
Crowdsourcing Task Marketplaces

Abstract In this chapter, we discuss detailed statistics of the popular Amazon Mechanical Turk (AMT) marketplace to provide insights in task properties and requester behavior. We present a model to automatically infer requester communities based on task keywords. Hierarchical clustering is used to identify relations between keywords associated with tasks. We present novel techniques to rank communities and requesters by using a graph-based algorithm. Furthermore, we introduce models and methods for the discovery of relevant crowdsourcing brokers who are able to act as intermediaries between requesters and platforms such as AMT.

Keywords Crowdsourcing · Mechanical turk · Hierarchical clustering · Community detection · Community ranking · Broker discovery

2.1 Introduction

In this chapter we define the notion of *communities* in the context of crowdsourcing. Communities are not predefined but emerge bottom-up based on posted tasks. Here we use keyword information applied to tasks to identify communities and community members (i.e., requesters). Hence, communities are mainly driven by *requesters*. For example, the keywords 'classification' and 'article' identify a community who makes tasks regarding the categorization of articles available. Managing the community standing of requesters in an automated manner helps to identify those requesters who contribute to a valuable marketplace.

In this chapter, we present the following key contributions:

- *Basic AMT Marketplace Statistics*. We thoroughly examine an AMT dataset and study properties regarding task distribution, rewarding, requester behavior and task keyword usage. The analysis of basic features and statistics provides the basis for the discovery of communities and the requester ranking model.

D. Schall, *Service-Oriented Crowdsourcing*, SpringerBriefs in Computer Science,
DOI: 10.1007/978-1-4614-5956-9_2, © The Author(s) 2012

- *Keyword Clustering Approach*. Hierarchical clustering is used to identify relations between keywords associated with tasks, and finally requester communities demanding for workers in particular expertise areas. This is an important step toward a community ranking model. To our best knowledge, there is no existing work that shows how to automatically discover communities in task-based crowdsourcing marketplaces.
- *Community Ranking Model*. We propose link analysis techniques derived from popular Web mining algorithms to rank requesters and communities. This model helps to rate requesters with respect to their task involvement on AMT.
- *Broker Discovery Model*. We present a novel model for the discovery and ranking of *crowdsourcing brokers*. Brokers act as intermediaries between requesters and platform providers. The duty of brokers is to provide a specialized interface towards crowdsourcing platforms by the provisioning of additional services such as quality assurance or validation of task results.
- *Evaluation of the Community Ranking Model and Broker Discovery Approach*. Our evaluation and discussions are based on the properties of a real crowdsourcing marketplace.

This chapter is organized as follows. Section 2.2 outlines important related work. In Sect. 2.3 we highlight the basic properties of the AMT marketplace, including interactions and the system context model. This is the basis for Sect. 2.4, where we discuss a hierarchical clustering approach in order to group keywords and subsequently associate tasks. Using that model, we introduce a task requester and community ranking model. In Sect. 2.5 we present the broker discovery and ranking model. Section 2.6 details our experiments that are based on real data obtained from the AMT platform. Section 2.7 concludes the chapter.

2.2 Background

The notion of **crowdsourcing** was coined by Howe [27, 28] and is defined as *'the act of taking a job traditionally performed by a designated agent and outsourcing it to an undefined, generally large group of people in the form of an open call'*. The crowdsourcing paradigm [14, 43] has recently gained increased attention from both academia and industry, and is even considered for application in large-scale enterprises.

Crowdsourcing offers a attractive way to solve resource intensive tasks that cannot be processed by software [49]; typically all kinds of tasks dealing with matching, ranking, or aggregating data based on fuzzy criteria. Some concrete examples include relevance evaluation [1], evaluation of visual designs and their perception by large user groups [24], and ranking of search results [8]. Numerous further approaches deal with the seamless integration of crowds into business processes and information system architectures: CrowdDB [20] uses human input via crowdsourcing to process queries that neither database systems nor search engines can adequately answer.

Others study algorithms which incorporate human computation as function calls [35]. One of the largest and most popular crowdsourcing platforms is AMT. Besides tagging images and evaluating or rating objects, creating speech and language data with AMT [7] and the transcription of spoken language [36] are in the focus of the large application area of language processing and language studies [38]. Recently various platforms have been established that interface with and harness AMT, in order to provide more customized services, such as SmartSheet [56] and CrowdFlower [13].

Tagging is used to improve the navigation in folksonomies [21] and has been widely studied [22]. Applying tags to objects helps users to discover and distinguish relevant resources. For instance, users manually annotate their photos on Flickr [18] using tags, which describe the contents of the photo or provide additional contextual and semantical information. This feature is also utilized on the AMT platform, where tasks are described by tags, informing potential workers about the nature of a task and basic required skills. In contrast to predefined categories, tags allow people to navigate in large information spaces, unencumbered by a fixed navigational scheme or conceptual hierarchy. Previous works [54] investigated concepts to assist users in the tagging phase. Tags can also assist in creating relationship between semantic similarity of user profile entries and the social network topology [3].

Several approaches have been introduced, dealing with the construction of **hierarchical structures** of tags [15, 26, 55], generating user profiles based on collaborative tagging [37, 53], and collaborative filtering in general [25]. Our work aims at a similar goal by clustering tags and recommending categories of keywords to requesters and workers looking for interesting tasks. Our approach uses various methods and techniques from the information retrieval domain, including term-frequency metrics [46], measuring similarities [51], and hierarchical clustering [44].

With regards to **community and role detection**, community detection techniques can be used to identify trends in online social networks [9]. A context-sensitive approach to community detection is proposed in [5] whereas [45] proposes random walks to reveal community structure. Actors in large scale online communities typically occupy different roles within the social network [17]. The authors in [16] present methods for classification of different social network actors. Certain actors may act as moderators to separate high and low quality content in online conversations [34]. We specifically focus on the notion of *community brokers* who have the ability to assemble a crowd according to the offered knowledge [58]. Brokers in a sociological context may bridge segregated collaborative networks [52]. These community brokers could be ranked according to their betweenness centrality in social networks (see [33] for identifying high betweenness centrality nodes). The idea of structural holes, as introduced by Burt [6], is that gaps arise in online social networks between two individuals with complementary resources or information. When the two are connected through a third individual (e.g., the broker) the gap is filled, thereby creating important advantages for the broker. Competitive advantage is a matter of access to structural holes in relation to market transactions [6].

We position our work in the context of crowdsourcing with the focus on *requester communities*. Some works [29, 31] already studied the most important aspects of the

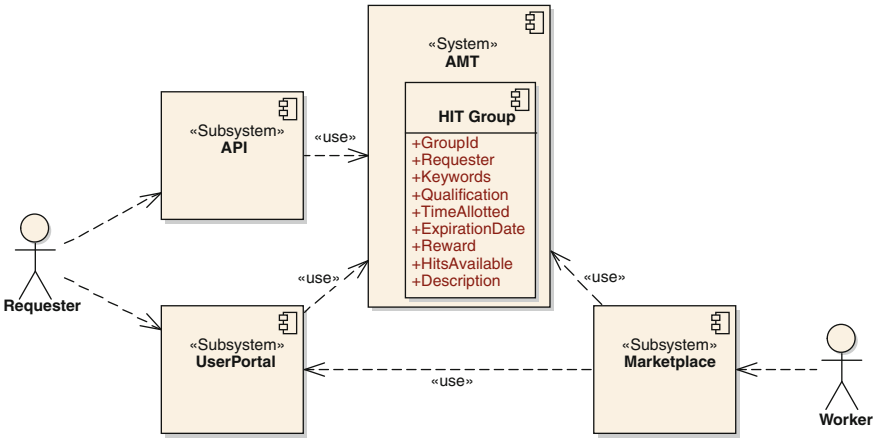

Fig. 2.1 Crowdsourcing system context model

AMT user community and describe analysis results of their structure and properties. While these works provide a basis for our experiments, we go important steps further:

1. We introduce *crowdsourcing communities* that are identified bottom-up through the analysis of the hierarchical keyword structure.
2. We develop a sophisticated link-based ranking approach to rank communities and requesters within the AMT community.
3. Here we propose the discovery of community brokers by adapting popular link mining techniques. The novelty of our approach is that the crowdsourcing broker discovery is based on query sensitive personalization techniques.

In the next section we introduce a basic task-based crowdsourcing model and discuss the statistics of the popular AMT crowdsourcing marketplace.

2.3 Basic Model and Statistics

2.3.1 System Context Overview

In this section we detail the basic system elements and user interactions. Figure 2.1 shows the high-level model and a set of generic building blocks. We illustrate the system context model by using the AMT platform and its HIT data model as an example of a task-based crowdsourcing marketplace.

- At the core, the **AMT** middleware offers the task management with a definition of the basic model for a **HIT Group**. The *GroupId* is a unique identifier of a HIT group. A HIT group encapsulates a number of HIT instances (*HitsAvailable*).

Workers can claim HIT instances within a group. The *Requester* identifier associates a task requester with a HIT group. Each HIT has a set of *Keywords* that will play a central role in subsequent discussions. The requester can define *Qualification* requirements such as geographical location. HITs are given a duration (specified as *TimeAllotted*) and an *ExpirationDate*. Workers receive a monetary *Reward* after successfully finishing a HIT instance. The *Description* attribute provides additional textual information.

- **Requesters** post tasks to the platform by using either the **User Portal** or a Web services based **API**. APIs help to automate the creation and monitoring of HITs. In addition, 3rd party crowdsourcing platform providers have the ability to build their own platforms on top of the AMT middleware.
- **Workers** are able to claim HIT instances from the **Marketplace** if they qualify for a given task. Additional constrains can be given by the requester, such as required skills or desired quality. Quality management is typically provided by 3rd party crowdsourcing platform providers (e.g., CrowdFlower) and not by the AMT system itself.

2.3.2 Marketplace Task Statistics

The techniques presented in this work are generally applicable to task-based crowdsourcing environments. To illustrate the application and rational behind our community discovery and ranking model, we will discuss the key features and statistics of real world crowdsourcing systems such as the AMT marketplace. We collected a HIT-dataset by periodically crawling AMT's Web site between February and August 2011 (in total seven months). The dataset contains 101027 HITs (5372355 HIT instances) and 5584 distinct requesters that were active during the time frame by making new HITs available.

Figure 2.2 shows the basic task statistics from the obtained dataset. In Fig. 2.2a we show the number of tasks and the number of requesters in a scatter plot with logarithmic scale on both axis (in short, log-log scale). The basic task-requester distribution follows the law that only few requesters post many tasks (the top-requester SpeechInk [57] posts 32175 HITs) while a large portion of requesters only post few tasks. A number of 2393 requesters (i.e. 43 %) only posts one task.

Next in Fig. 2.2b we show the number of tasks that require qualification versus tasks that do not require any particular qualification. The x-axis shows the number of task instances available within a HIT group and the y-axis depicts the number of tasks grouped by the amount of task instances available. Generally, more tasks require some sort of qualification like based on location ('Location is not in India') or based on qualification ('Image Transcription Description Qualification is greater than 88'). Thus, from the requester point of view, there is already some pre-selection of workers. However, AMT offers limited support to actually filter and rank tasks and requesters.

Figure 2.2c shows the time allotted to tasks (in minutes). The largest segment of tasks is concentrated around 60–100 min. This means that most tasks are relatively

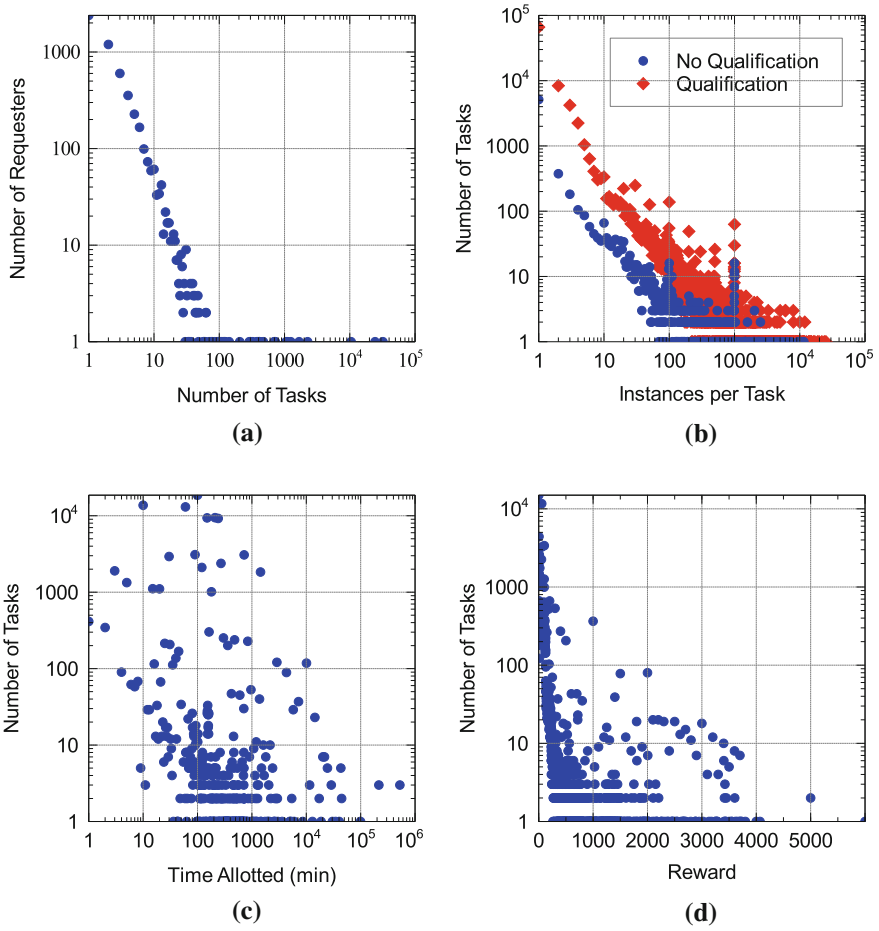

Fig. 2.2 Task statistics. **a** Number of tasks. **b** Qualification. **c** Time alloted. **d** Task reward

simple such as searching for email addresses or tagging of images. Finally, Fig. 2.2d shows the reward in cents (US currency) given for processing tasks (the x-axis is shown on a linear scale). The maximum reward given for a task is 60$. However, we find that most tasks have relatively little reward (26604 tasks have less than 55 cents reward).

Our community discovery and ranking approach uses task-keyword information as input for clustering of communities. Figure 2.3 shows the most important keyword statistics (all log-log scale). Figure 2.3a shows the number of keywords versus the number of requesters. The x-axis is based on the total number of keywords used by requesters. The distribution has its maximum (y-axis) at 4 keywords amounting for 758 requesters. Next, Fig. 2.3b depicts the average number of HIT keywords in relation to the number of requesters. By inspecting the maximum value (y-axis),

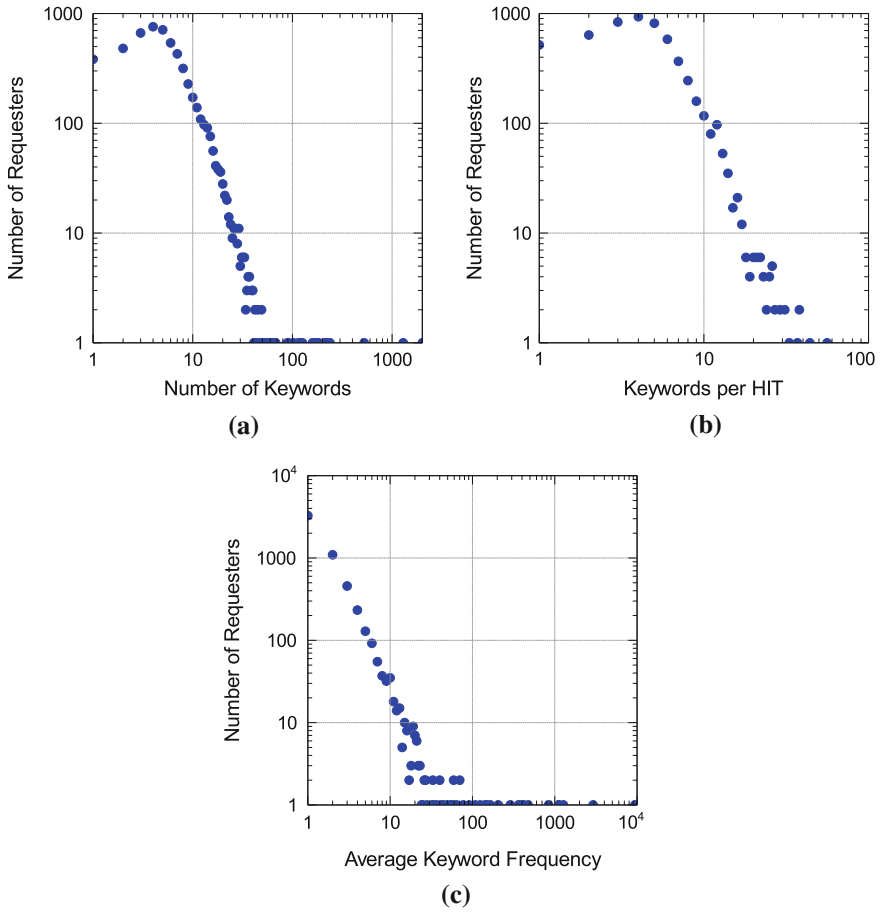

Fig. 2.3 Keyword statistics. **a** Keywords. **b** Keywords HIT. **c** Frequency

we observe that 935 requesters apply on average 4 keywords per HIT. The last keyword related statistic is shown in Fig. 2.3c depicting how often a particular keyword is used. By looking at the raw (unfiltered) keyword set, keywords with the highest frequency include company names of 3rd party platform providers such as SpeechInk, CastingWords or CrowdFlower.

However, a prerequisite for creating meaningful (hierarchical) clusters is a set of keywords that is not distorted by such keyword relations. Thus, we performed some filtering and cleaning of all stopwords ('and', 'on', 'is' and so forth) and also company names. Amongst those top-ranked keywords are 'survey', 'data', or 'collection'. Ranked by total reward, the keywords 'data' (79492$), 'transcribe' (55013$), 'search' (54744$), 'transcription' (47268$), 'collection' (43156$), and 'voicemail' (42580$) would be among the top ones.

2.4 Clustering and Community Detection

2.4.1 Clustering Approach

Current crowdsourcing platforms such as AMT offer limited search and navigation support for both requesters and workers. Workers are usually presented a long list of tasks to potentially work on and need to navigate from page-to-page to discover new and interesting tasks. Some workers may prefer to work on tasks regarding a specific topic of interest. Therefore, workers should be able to discover communities that post tasks regarding the desired topic. Requesters pick keywords they would like to apply to tasks freely and independently without following a particular convention or taxonomy. The positive aspect of a bottom-up approach (i.e., freely choosing keywords) is a domain vocabulary that may actually change based on the keywords chosen by the requesters. On the downside, problems include spelling mistakes, ambiguity, or synonyms because a large amount of different keywords may be used to describe the same type of task.

We propose *hierarchical clustering* to structure the flat set of task-based keywords into a hierarchy. The general idea is to first calculate the co-occurrence frequency of each keyword (how many times a particular keyword is used in combination with another keyword) and second group pairs of keywords into clusters based on a distance metric. Each HIT keyword starts in its own cluster. Subsequently pairs of clusters are merged by moving up the hierarchy. In other words, the correlation between keywords increases by moving from the top (root) to the bottom (leaves).

We have tested different distance metrics and configurations of the clustering algorithm. Based on our experiments, the following configuration yielded the best results (i.e., hierarchical structure). *Pairwise average-link clustering* merges in each iteration the pair of clusters with the highest cohesion. We used the *city-block distance*, alternatively known as the Manhattan distance, to measure the cohesiveness between pairs of clusters. In the conducted experiments, the input for the clustering algorithm was a set of about 300 keywords that have already been filtered as described in the previous section. Furthermore, we only used those keywords that had a co-occurrence frequency of at least 10 with some other keyword (minimum threshold). In total, the algorithm generates 328 clusters.

The next step in our approach is to create communities using the layout of the keyword-based hierarchy. This is shown in Algorithm 1. It is important to note that in Line 7 of the algorithm the keywords of all child-clusters are retrieved as well. To calculate the overlap in Line 9, the set intersection between KW_{HIT} and $KW_{Cluster}$ is divided by the set size $|KW_{Cluster}|$. Note that by associating collections of HITs to clusters (Line 16) we extend the notion of clusters to *communities* (i.e., an extended structure of a cluster with associated tasks and requesters). As a next step, we calculate basic statistics of the resulting community structure.

First, we show how many tasks requesters have in each cluster (Fig. 2.4). Since many requesters post only one task, also the count of requesters that have only one task in a cluster is high (757 requesters have one task in a cluster). In the middle

Algorithm 1 Creating communities using keyword hierarchy.

```
1: input: Set of Tasks, Keyword Hierarchy
2: for each HIT in Set of Tasks do
3:   for each Cluster in Hierarchy do
4:     // Keywords of HIT
5:     KW_HIT ← GetKeywords(HIT)
6:     // Keywords of Cluster and Children
7:     KW_Cluster ← GetKeywords(Cluster)
8:     // Save Overlap
9:     Overlap(HIT, Cluster) ← |KW_HIT ∩ KW_Cluster| / |KW_Cluster|
10:  end for
11:  // Sort Clusters by Overlap
12:  SortedList ← GetSortedClusterList(Overlap, HIT)
13:  // Pick highest ranked Cluster
14:  Cluster ← PickFirst(SortedList)
15:  // Add HIT to Collection associated with Cluster
16:  CollectionAdd(Cluster, HIT)
17: end for
```

segment, 744 requesters have 2 to 10 tasks in clusters. The high score is one requester with 2058 in one cluster.

A natural question when performing clustering is the quality of the resulting hierarchy. As mentioned before, we have evaluated the resulting clusters by looking at the groupings of keywords, which were consistent. Another possible metric for measuring the quality is the distance of similar tasks. Recall, each HIT is associated with a cluster (Algorithm 1, Line 16). Also, the hierarchy of clusters can be represented as a directed graph $G(V, E)$ where vertices V represent clusters and edges E the set of links between clusters (e.g., the root node points to its children and so forth). To understand whether the hierarchy represents a good structure, we sampled 1000 pairs of tasks randomly from the entire set of tasks and calculated the keyword-based similarity between the pair. Next, we calculated the Dijkstra shortest path distance between the pair of tasks using $G(V, E)$. The results are depicted by Fig. 2.5.

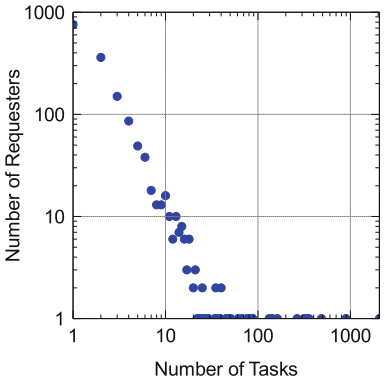

Fig. 2.4 Number of tasks in community clusters

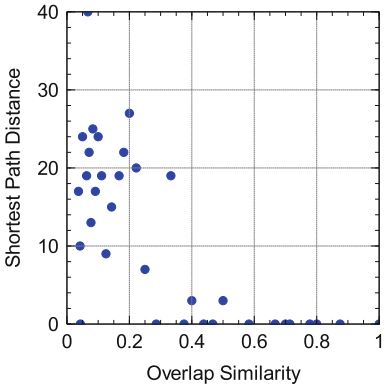

Fig. 2.5 Overlap versus distance in community structure

One can see that the hierarchy of clusters represents a good mapping between overlap similarity and shortest path distance: similar tasks have a low distance and tasks with little overlap similarity are very distant from each other in the graph G. Another positive effect of the proposed clustering and community discovery method (task association to clusters) is that spam or unclassified tasks can be identified.

2.4.2 Community-Based Ranking Model

In the previous section we investigated clustering methods to build communities using the keyword-based hierarchy. Here we attempt to answer the following question: which are the most important communities and who are the most recommendable requesters? The applications of the presented ranking approach are, for example, recommending relevant communities to both workers and requesters (e.g., to find interesting tasks) and also rating tasks of requesters with a high community standing. First, we need to detail the meaning of 'relevant communities' and 'community standing' of requesters. A relevant community is identified based on the authority of the requesters that post tasks to it. The community standing of requesters (i.e., authority) is established upon the relevancy of the communities the requester posts tasks to. Mathematically, the idea of this ranking model can be formalized using the notion of *hubs and authorities* as introduced in [32].

Formally, this recursive definition is written as

$$\mathscr{H}(c) = \sum_{(c,r)\in E_{CR}} w(r \to c)\mathscr{A}(r) \tag{2.1}$$

$$\mathscr{A}(r) = \sum_{(c,r)\in E_{CR}} w(r \to c)\mathscr{H}(c) \tag{2.2}$$

with $\mathscr{H}(c)$ being the hub score of the community $c \in V_C$ in the set of communities V_C, $\mathscr{A}(r)$ the authority score of the requester $r \in V_R$ in the set of requesters V_R, $(c, r) \in E_{CR}$ an edge in the community-requester bipartite graph $G_{CR}(V_C, V_R, E_{CR})$, and $w(r \rightarrow c)$ a weighting function based on the number of tasks posted by r in a community c. Notice, by posting tasks to a community c, an edge is established between the community c and the requester r.

2.5 Crowdsourcing Broker Discovery

There are multiple companies that provide marketplaces where users can post tasks that are processed by workers. Among the previously discussed AMT, there are also platform providers such as oDesk [39] and Samasource [47]. In contrast other companies, such as CrowdFlower [13] and ClickWorker [12] act as intermediaries allowing large businesses and corporations to not have to worry about framing and posting tasks to crowdsourcing marketplaces [41]. We call such intermediaries *brokers*. Brokers post tasks on behalf of other crowdsourcing requesters. Typically, such brokers offer additional services on top of platforms like AMT including quality control (e.g., see CrowdFlower [13]) or the management of Service-Level-Agreements (SLAs) [42].

In this work we provide a model for the discovery and ranking of brokers based on requester profile information. A requester's profile is created based on the task posting behavior and associated keywords. The requester profile contains a set of keywords and their frequency. The profile is defined as follows:

$$P_u = \left\{ \left\langle k_1, f_{k_1} \right\rangle, \left\langle k_2, f_{k_2} \right\rangle, \ldots, \left\langle k_n, f_{k_n} \right\rangle \right\} \tag{2.3}$$

where $\left\langle k_n, f_{k_n} \right\rangle$ denotes the tuple of keyword k_n and its frequency f_{k_n}. Next we propose the creation of a directed *profile graph* $G_{PG}(V_C, E_{PG})$ that is created using the following algorithm:

Algorithm 2 Creating profile graph G_{PG} using requester profile information.

```
1: input: Set V_R of Requesters
2: for each Requester u ∈ V_R do
3:   for each Requester v ∈ V_R do
4:     if u ≠ v then
5:       // Calculate match between u and v
6:       pm ← match(u, v)
7:       if (pm > ξ) or (ξ = 1 and pm = ξ) then
8:         // Add profile relation to G_PG
9:         GraphAddRelation(u, v)
10:      end if
11:    end if
12:  end for
13: end for
```

The idea of our approach (as highlighted in Algorithm 2) is to establish a directed edge $(u, v) \in E_{PG}$ from u to v if there is high match (i.e., profile similarity) between u and v from u's point of view. The parameter ξ can be used to adjust the similarity threshold that the profile match pm must exceed to connect u and v through a profile relation edge.

$$match(u, v) = 1 - \sum_{k \in KW_u} w_k Max((f_k(u) - f_k(v))/f_k(u), 0) \tag{2.4}$$

The calculation of the degree of match (see 2.4) is not symmetric. This is done because of the following reason. The requester v's profile P_v might exactly match u's profile P_u with $P_u \subseteq P_v$ ($match(u, v)$) but this may not be true when matching u's profile from v's point of view ($match(v, u)$). Suppose we calculate $match(u, v)$, a match of 1 means that v perfectly matches u's profile. Thus, it can be assumed that v has posted some tasks that are very similar to those posted by u. Therefore, it can be said that there is a high degree of interest similarity between u and v and v could potentially act as a broker for u. Again, keyword information associated with the requesters' HITs is used to create profiles through mining. Also keyword frequency is taken into account when calculating profile matches. A detailed description on the symbols depicted in (2.4) can be found in Table 2.1.

As mentioned before, a broker could submit the task on behalf of another requester and monitor the task's progress or could even segment the task into subtasks and submit the subtasks to one or more crowdsourcing platforms. At this stage, we focus on the discovery and ranking techniques of brokers without discussing the actual broker-requester interaction model. For instance, the management of SLAs in crowdsourcing environments has been addressed in our previous work [42] and is not the focus of this research.

Here we focus on the discovery and ranking of relevant brokers. Compared to the previously defined community requester graph G_{CR}, the profile graph G_{PG} consists only of a single type of nodes V_R. In this case, the requester importance is not influenced by the relevance of communities but rather by the degree of connectivity within the graph G_{PG}. A well-known and popular model to measure importance in directed networks is PageRank [40]. A advantage over the hubs and authority method [32] is that the PageRank model corresponds to a random walk on the graph.

Table 2.1 Description of profile matching calculation

Symbol	Description
$match(u, v)$	The matching of profiles between u and v. A value between [0, 1]
KW_u	The set of keywords used by u
f_k	The frequency of a keyword k. The frequency $f_k(u)$ is counted based on how many times the keyword k has been applied to tasks posted by u
w_k	The weight of a specific keyword k. The weight is calculated as $\frac{f_k}{\sum_{k_i} f_{k_i}}$

The PageRank $pr(u)$ of a node u is defined as follows:

$$pr(u) = \frac{\alpha}{|V_R|} + (1-\alpha) \sum_{(v,u)\in E_{PG}} w(v,u)\, pr(v) \tag{2.5}$$

At a given node u, with probability α the random walk continues by following the neighbors $(v, u) \in E_{PG}$ connected to u and with probability $(1 - \alpha)$ the walk is restarted at a random node. The probability of 'teleporting' to any node in the graph is given as by the uniform distribution $\frac{1}{|V_R|}$. The weight of the edge (v, u) is given as $w(v, u)$. The default value for the transition probability between v and u is given as $\frac{1}{\texttt{outdegree}(v)}$. The function `outdegree` returns the count of the edges originating from v. The model can also be personalized by assigning non-uniform 'teleportation' vectors, which is shown in the following:

$$ppr(u;Q) = \alpha p(u;Q) + (1-\alpha) \sum_{(v,u)\in E_{PG}} \frac{ppr(v)}{\texttt{outdegree}(v)} \tag{2.6}$$

The personalized PageRank $ppr(u; Q)$ is parameterized by the keyword based query Q. Instead of assigning uniform teleportation probabilities to each node (i.e., $\frac{1}{|V_R|}$), we assign preferences to nodes that are stored in $p(u; Q)$. This approach is similar to the topic-sensitive PageRank proposed by [23] (see also [10, 19, 30, 50]). Whereas in PageRank the importance of a node is implicitly computed relative to all nodes in the graph now importance is computed relative to the nodes specified in the personalization vector. The query Q is defined as a simple set of keywords $Q = \{k_1, k_2, \ldots, k_n\}$ that are selected to depict a particular topic(s) of interest. Algorithm 3 shows how to compute the values within the personalization vector $p(u; Q)$.

2.6 Experiments

The discussions on our evaluation and results in separated into two sections: first we discuss experiments of our community-based ranking model followed by discussions of the crowdsourcing broker ranking approach.

2.6.1 Community Discovery and Ranking

Here we discuss ranking results obtained by calculating $\mathscr{H}$ and $\mathscr{A}$ scores using the community-requester graph G_{CR}. Communities are visualized as triangular shapes (in blue color) and requesters are visualized as circles (in red color). The size of each shape is proportional to the $\mathscr{H}$ and $\mathscr{A}$ scores respectively. The line width of an edge is based on the weight $w(r \rightarrow c)$.

Algorithm 3 Assigning personalization vector $p(u;Q)$ based on query Q.

input: Set V_R of Requesters and query Q
// Variable $totalSum$ is used for normalization
$totalSum \leftarrow 0$
for each Requester $u \in V_R$ **do**
  $currentSum \leftarrow 0$
  for each Keyword $k^Q \in Q$ **do**
    for each Keyword $k \in KW_u$ **do**
      // If k^Q and k matches
      if $Equals(k^Q, k)$ and $f_k(u) > 0$ **then**
        // Add frequency $f_k(u)$ to $currentSum$
        $currentSum \leftarrow currenSum + f_k(u)$
      end if
    end for
  end for
  PersonalizationVectorAdd(u, $currentSum$)
  $totalSum \leftarrow totalSum + currentSum$
end for
// Normalize the values in $p(u;Q)$
for each Requester $u \in V_R$ **do**
  $weight \leftarrow$ PersonalizationVectorGet(u)
  PersonalizationVectorAdd(u, $weight/currentSum$)
end for

First we look at the top-10 requesters and communities they are associated with. The top-10 requester graph is depicted by Fig. 2.6. Table 2.2 shows the number of clusters the requester is associated with (3rd column) and the number of tasks (4th column). In addition, the table depicts in the last column the rounded values of the $\mathscr{A}$ scores to show how the scores among top-ranked requesters are distributed.

The requester Smartsheet.com Clients clearly outranks other requesters. It has also posted a large number of tasks to relevant communities. Between the 5th to the 10th ranked requesters one can see less significant differences in the ranking scores because the number of tasks and clusters are also not significantly different. The number one ranked community in AMT using our community discovery and ranking approach is the community dealing with 'data' and 'collection'. Each requester in the top-10 list is associated with a number of communities and all requesters are also connected with the top community.

Next, we filter the graph and show the top-10 communities and the associated requesters in Fig. 2.7 (best viewed online). Descriptions on the top-ranked communities are given in Table 2.3.

The top-ranked community deals with 'data' and 'collection' and can be easily located in the graph by looking at the triangular node which has a dense neighborhood of requesters (top left in Fig. 2.7). Table 2.3 shows in addition to the cluster-based keywords (last column) the total number of tasks found in the community (3rd column), the number of top-10 ranked requesters connected to a given community (4th column), and the number of top-10 ranked requester tasks (also in 4th column in parentheses). One can observe that top-ranked communities have also top-ranked

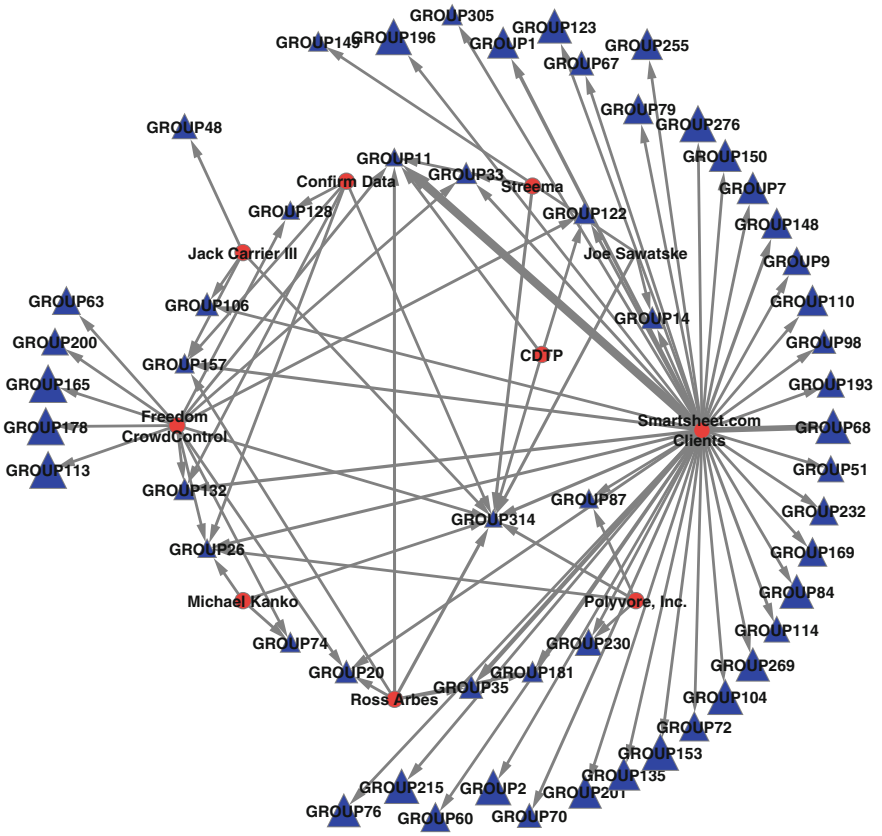

Fig. 2.6 Top-10 requester graph

Table 2.2 Description of top-10 requester graph

Rank	Requester name	Number of clusters	Number of tasks	$\mathscr{A}$ (rounded)
1	Smartsheet.com Clients	46	877	0.002706
2	Freedom CrowdControl	14	51	0.002387
3	Ross Arbes	6	56	0.002269
4	Confirm Data	5	16	0.002249
5	Polyvore, Inc.	4	20	0.002206
6	Streema	3	22	0.002203
7	CDTP	3	13	0.002197
8	Jack Carrier III	4	12	0.002194
9	Michael Kanko	3	14	0.002192
10	Joe Sawatske	3	19	0.002183

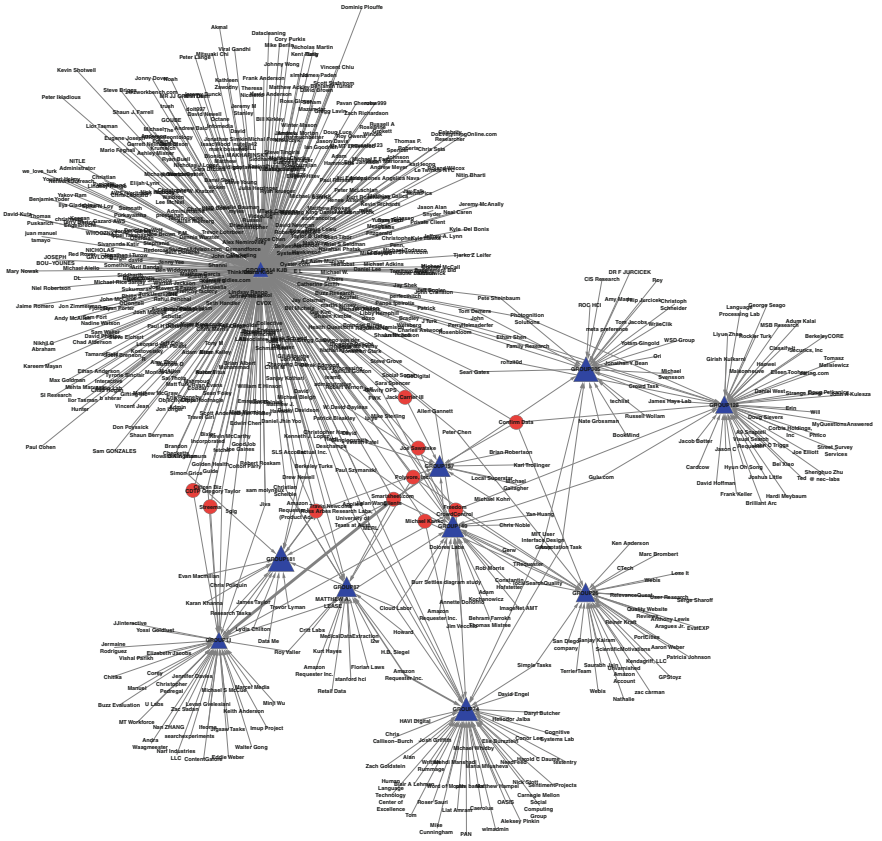

Fig. 2.7 Top-10 community graph

requesters associated with them, which is the desired behavior of our ranking model. Also, since the algorithm takes weighted edges into account, the number of tasks that are actually posted in a community play a key-role.

To conclude our discussions, top-ranked requesters are identified based on their active contribution (posting tasks) to top-ranked communities.

2.6.2 Recommendation of Crowdsourcing Brokers

In this section, we discuss the performed experiments to discovery crowdsourcing brokers. First, we take the entire set V_R and establish the keyword-based profiles of each requester. In the next step, we apply the Algorithm 2 to construct G_{PG} using the matching function as defined in (2.4). To find a suitable threshold ξ, we generated a number of graphs with varying thresholds $0.0 \leq \xi \leq 1.0$ and measured the indegree

Table 2.3 Description of top-10 community graph

Rank	Community Id	Number of tasks	Number top-requesters	Keywords
1	GROUP314	1222	10 (109)	Data, collection
2	GROUP11	801	5 (352)	Actors, website
3	GROUP128	136	2 (4)	Photo, article, social
4	GROUP26	146	5 (20)	TV, web
5	GROUP87	139	2 (4)	Review, shopping
6	GROUP157	34	3 (18)	Marketing, phone
7	GROUP149	53	1 (1)	Moteur, question
8	GROUP74	250	2 (10)	Assistance, text
9	GROUP305	92	1 (4)	Fast, quick
10	GROUP181	28	2 (9)	Interesting, business

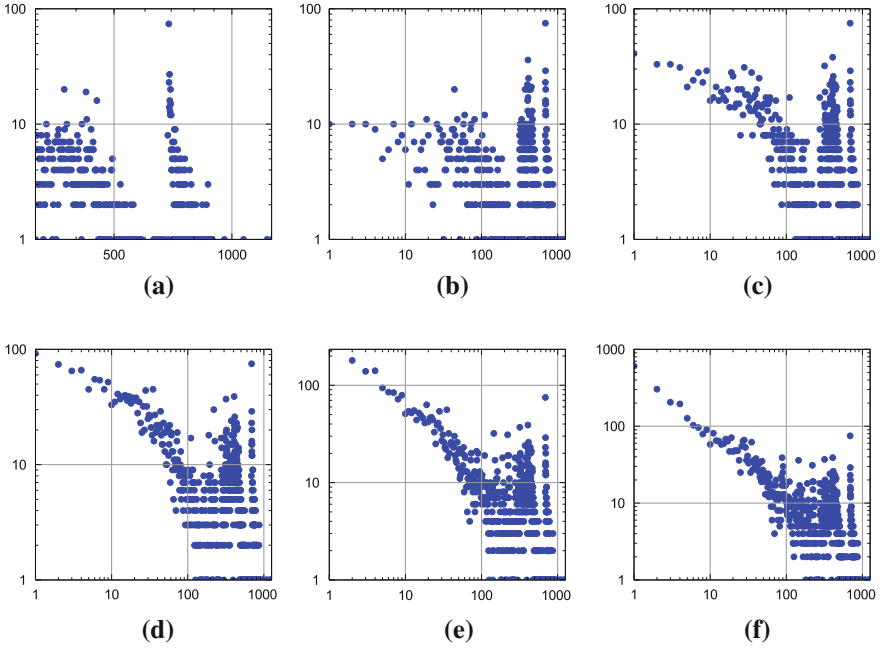

Fig. 2.8 Degree distributions under different connectivity thresholds. **a** $\xi = 0.0$. **b** $\xi = 0.1$. **c** $\xi = 0.2$. **d** $\xi = 0.3$. **e** $\xi = 0.4$. **f** $\xi = 0.5$

of each node in G_{PG}. The following series of scatter plots and shows the indegree distributions for the interval $0.0 \leq \xi \leq 0.5$ (Fig. 2.8) and $0.6 \leq \xi \leq 1.0$ (Fig. 2.9) respectively. On the horizontal axis, the indegree is shown and on the vertical axis the number of nodes. Recall that the set V_R holds 5584 distinct requesters.

The plots in Figs. 2.8a and 2.9e can be regarded as boundaries. However, notice that the profile match *pm* must be greater than ξ (see Algorithm 2). Otherwise each

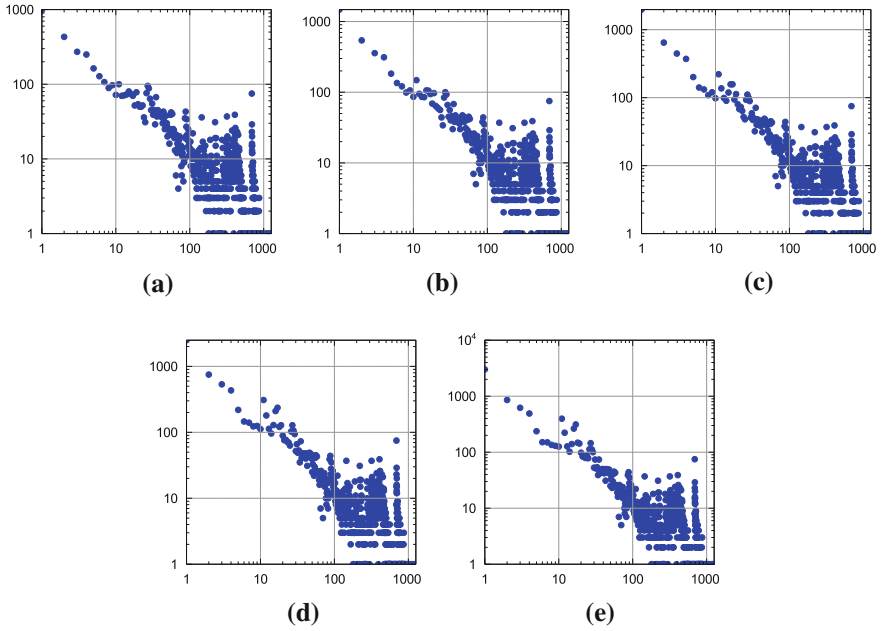

Fig. 2.9 Degree distributions under different connectivity thresholds. **a** $\xi = 0.6$. **b** $\xi = 0.7$. **c** $\xi = 0.8$. **d** $\xi = 0.9$. **e** $\xi = 1.0$

requester would be connected to all other requesters yielding an indegree of 5583 for all requesters. In the case of $pm > 0.0$ (Fig. 2.8a), the indegree is almost evenly distributed with an average degree of 500.

Next we increased ξ and observe that the indegree yields a shape similar to those indegree distributions found in naturally emerging graphs [2]. The majority of nodes has a low indegree whereas a lower number of nodes has a high indegree. This scaling law is also clearly visible when setting a threshold of $\xi > 0.4$. This behavior fits well our proposal where a few requesters would qualify for being brokers that transmit tasks on behalf of others (i.e., the majority of nodes) to crowdsourcing platforms such as AMT or oDesk. Within the interval $\xi = [0.5, 1.0]$ the degree distributions exhibit a similar shape.

In subsequent discussions, we chose a threshold of $\xi = 0.5$ since higher thresholds would not drastically change the shape of the distribution.

The next step in the broker ranking approach is to take the graph G_{PG} and calculate ppr scores using (2.6). To illustrate the approach, we set the query keywords as $Q = \{$'korrigieren', 'deutsch'$\}$ to find and rank requesters which would be suitable brokers for tasks related to correcting German related documents, articles, etc. The personalization vector $p(u; Q)$ was calculated using Algorithm 3. Furthermore, (2.6) was parameterized with $\alpha = 0.15$. The top-10 results are visualized by Fig. 2.10. The node size is based on the position (1–10) in the ranking results where the number 1

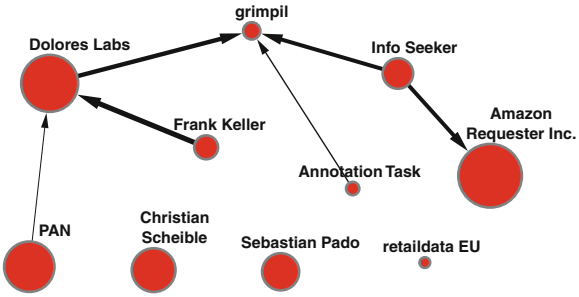

Fig. 2.10 Top-10 ranked requesters in G_{PG}

ranked node has the largest size and the number 10 ranked node the smallest size. The edge width is based on the matching score calculated by using (2.4). Only edges among the top-10 ranked requesters are shown. Information regarding the top-10 ranked requesters is further detailed in Table 2.4.

We made the following important observation when performing broker discovery in G_{PG}. The greatest benefit of applying a network-centric approach to ranking requesters is the discovery of related requesters that may not actually match any of the keywords provided by Q. Suppose the following case where $\alpha = 1$ yielding

$$ppr(u;Q) = p(u;Q), \quad \text{with} \quad \alpha = 1. \tag{2.7}$$

Requesters are assigned a value of 0 if none of their profile keywords match Q and otherwise the weight based on keyword frequency as described in Algorithm 3. In other words, requesters are ranked based on simple keyword-based matching and frequency-based weighting. The results are quite similar by having the top-7 requesters (see Table 2.4) ranked in the same order:

Table 2.4 Description of top-10 ranked requesters in G_{PG}

Rank	Requester name	Indegree	Number of tasks (total)	*ppr* (rounded)
1	Amazon Requester Inc.	25	1021	0.139038
2	Dolores Labs	933	2063	0.006523
3	PAN	12	11	0.002715
4	Christian Scheible	27	7	0.002036
5	Sebastian Pado	1	2	0.001357
6	Info Seeker	4	2	0.001357
7	Frank Keller	16	3	0.000679
8	Grimpil	60	3	0.000577
9	Annotation Task	227	18	0.000245
10	Retaildata EU	43	21	0.000002

Table 2.5 Top-ranked requesters using (2.7)

Rank	Requester name
1	Amazon Requester Inc.
2	Dolores Labs
3	PAN
4	Christian Scheible
5	Sebastian Pado
6	Info Seeker
7	Frank Keller

Notice, however, only 7 out of 5583 requesters exactly match the query $Q = \{\text{'korrigieren', 'deutsch'}\}$. Thus, all other nodes will receive a ranking score of 0. By applying our proposed approach using (2.6) we have the following important benefits:

- Since importance of requesters is computed relative to the nodes specified in the personalization vector, *all* nodes (requesters) receive a ranking score.
- Requesters that do not match the query but are connected in G_{PG} with other high ranked requesters will be able to improve their position in the ranking results.

Amazon Requester Inc. is the highest ranked requester in either case, $\alpha = 0.15$ and $\alpha = 1$, with regards to the keywords specified in Q. Therefore, this requester would be the most recommendable broker. However, by using $\alpha = 0.15$ other requesters such as grimpil (see requester with rank 8 in Table 2.4) who has strong inbound links from Dolores Labs and Info Seeker are also discovered in the top-10 list. This requester, for example, would have not been discovered otherwise.

Overall, our approach provides an important tool for the discovery of brokers by establishing a profile-relationship graph G_{PG} and by ranking requesters based on their actual match (i.e., $p(u;Q)$) and based on their degree of connectivity (the second part of (2.6)—that is $\sum_{(v,u)\in E_{PG}} \frac{ppr(v)}{\texttt{outdegree}(v)}$).

Indeed, both components cannot be computed independently by simply summing them up. Instead, the *pr* and *ppr* scores must by computed in an iterative process that needs to converge towards a fixed value (see [23, 40]). Finally, our approach lets other requesters desiring to utilize brokers for crowdsourcing their tasks to discovery the best matching brokers and also pathways revealed by G_{PG} to matching brokers (e.g., a suitable path to Amazon Requester Inc. can be established via Info Seeker).

As a final remark on the experiments, time complexity of the presented clustering and ranking algorithms becomes an issue as the size of the number of keywords and the number of requesters increases. As mentioned in Sect. 2.4, at this point we have considered a subset of about 300 keywords that have already been filtered. Furthermore, we only used those keywords that had a co-occurrence frequency of at least 10 with some other keyword (minimum threshold). Thus, time complexity has not been an important issues in our currently conducted experiments but deserves attention in our future experiments with larger crowdsourcing datasets.

2.7 Conclusion and Future Work

Crowdsourcing is a new model of outsourcing tasks to Internet-based platforms. Models for community detection and broker discovery have not been provided by existing research. In this work we introduced a novel community discovery and ranking approach for task-based crowdsourcing markets. We analyzed the basic marketplace statistics of AMT and derived a model for clustering tasks and requesters. The presented approach and algorithm deliver very good results and will help to greatly improve the way requesters and workers discover new tasks or topics of interest.

We have motivated and introduced a broker discovery and ranking model that lets other requesters discovery intermediaries who can crowdsource tasks on their behalf. The motivation for this new broker based model can be manifold. As an example, brokers allow large businesses and corporations to crowdsource tasks without having to worry about framing and posting tasks to crowdsourcing marketplaces

In future work we will compare the presented hierarchical clustering approach with other techniques such as Latent Dirichlet Allocation (LDA) [4]. In addition, we will evaluate the quality of the keyword-based cluster as well as the community rankings through crowdsourcing techniques (cf. also [11]). With regards to community evolution, we will analyze the dynamics of communities (birth, expansion, contraction, and death) by looking at the task posting behavior of requesters. This will help to make predictions about the needed number of workers with a particular set of skills. The crowdsourcing platform may manage resource demands by creating training tasks to prevent shortcomings in the availability of workers that satisfy task requirements. Some of the related issues have been tackled in our previous work [48] but an integration with the present work is needed.

Furthermore, based on our broker discovery approach, we will look at different negotiation and service level agreement setup strategies. The personalization vector could be computed based on further parameters such as costs, the requesters availability and reliability constraints. Finally, standardization issues of interfaces towards crowdsourcing platforms in general as well as interfaces for brokers will be part of future research.

References

1. Alonso, O., Rose, D.E., Stewart, B.: Crowdsourcing for relevance evaluation. SIGIR Forum **42**(2), 9–15 (2008)
2. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509 (1999)
3. Bhattacharyya, P., Garg, A., Wu, S.: Analysis of user keyword similarity in online social networks. Soc. Netw. Anal. Min. **1**, 143–158 (2011). doi:10.1007/s13278-010-0006-4
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
5. Branting, L.: Context-sensitive detection of local community structure. Soc. Netw. Anal. Min. **1**, 1–11 (2012). doi:10.1007/s13278-011-0035-7

6. Burt, R.S.: Structural Holes: The Social Structure of Competition. Harvard University Press, Cambridge (1992)
7. Callison-Burch, C., Dredze M.: Creating speech and language data with amazon's mechanical turk. In: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, CSLDAMT '10. Association for Computational Linguistics, pp. 1–12. Stroudsburg, PA, USA (2010)
8. Carvalho, V.R., Lease, M., Yilmaz, E.: Crowdsourcing for search evaluation. SIGIR Forum **44**(2), 17–22 (2011)
9. Cazabet, R., Takeda, H., Hamasaki, M., Amblard, F.: Using dynamic community detection to identify trends in user-generated content. Soc. Netw. Anal. Min. 1–11 (2012). doi:10.1007/s13278-012-0074-8
10. Chakrabarti, S.: Dynamic personalized pagerank in entity-relation graphs. In: Proceedings of the 16th International Conference on World Wide Web, WWW '07, pp. 571–580. ACM, New York (2007)
11. Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., Blei, D.: Reading tea leaves: How humans interpret topic models. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems 22, pp. 288–296. MIT press, Cambridge (2009)
12. ClickWorker: http://www.clickworker.com/ (2012). Accessed 20 Aug
13. CrowdFlower: http://crowdflower.com/ (2012). Accessed 20 Aug
14. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. Commun. ACM **54**(4), 86–96 (2011)
15. Eda, T., Yoshikawa, M., Yamamuro, M.: Locally expandable allocation of folksonomy tags in a directed acyclic graph. In: Proceedings of the 9th International Conference on Web Information Systems Engineering, WISE '08, pp. 151–162. Springer, Berlin, Heidelberg (2008)
16. Fazeen, M., Dantu, R., Guturu, P.: Identification of leaders, lurkers, associates and spammers in a social network: context-dependent and context-independent approaches. Soc. Netw. Anal. Min. **1**, 241–254 (2011). doi:10.1007/s13278-011-0017-9
17. Fisher, D., Smith, M., Welser, H.T.: You are who you talk to: detecting roles in usenet newsgroups. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS '06, Vol. 3, p. 59.2, IEEE Computer Society, Washington, DC, USA (2006)
18. Flickr: http://www.flickr.com/ (2012). Accessed 20 Aug
19. Fogaras, D., Rácz, B., Csalogány, K., Sarlós, T.: Towards scaling fully personalized pagerank: algorithms, lower bounds, and experiments. Internet Math. **2**(3), 333–358 (2005)
20. Franklin, M.J., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: Crowddb: answering queries with crowdsourcing. In: Proceedings of the 2011 International Conference on Management of Data, SIGMOD '11, pp. 61–72. ACM, New York (2011)
21. Gemmell, J., Shepitsen, A., Mobasher, B., Burke, R.: Personalizing navigation in folksonomies using hierarchical tag clustering. In: Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery, DaWaK '08, pp. 196–205. Springer, Berlin, Heidelberg (2008)
22. Golder, S., Huberman, B.A.: Usage patterns of collaborative tagging systems. J. Inform. Sci. **32**(2), 198–208 (2006)
23. Haveliwala, T.H.: Topic-sensitive pagerank. In: Proceedings of the 11th International Conference on World Wide Web, WWW '02, pp. 517–526. ACM, New York (2002)
24. Heer, J., Bostock, M.: Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In: Proceedings of the 28th International Conference on Human factors in Computing Systems, CHI '10, pp. 203–212. ACM, New York (2010)
25. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)
26. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical report, Computer Science Department, Standford University, April (2006)
27. Howe, J.: The rise of crowdsourcing. Wired **14**(14), 1–5 (2006)

28. Howe, J.: Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business. Crown Business, New York (2008)
29. Ipeirotis, P.G.: Analyzing the amazon mechanical turk marketplace. XRDS **17**, 16–21 (2010)
30. Jeh, G., Widom, J.: Scaling personalized web search. In: Proceedings of the 12th International Conference on World Wide Web, WWW '03, pp. 271–279. ACM, New York (2003)
31. Kittur, A., Chi, E.H., Suh, B.: Crowdsourcing user studies with mechanical turk. In: Proceedings of the 26th Annual SIGCHI Conference on Human factors in Computing Systems, CHI '08, pp. 453–456. ACM, New York (2008)
32. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
33. Kourtellis, N., Alahakoon, T., Simha, R., Iamnitchi, A., Tripathi, R.: Identifying high betweenness centrality nodes in large social networks. Soc. Netw. Anal. Min. 1–16 (2012). doi:10.1007/s13278-012-0076-6
34. Lampe, C., Resnick, P.: Slash(dot) and burn: distributed moderation in a large online conversation space. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, pp. 543–550. ACM, New York (2004)
35. Little, G., Chilton, L.B., Goldman, M., Miller, R.C.: Turkit: human computation algorithms on mechanical turk. In: Proceedings of the 23rd Annual ACM symposium on User Interface Software and Technology, UIST '10, pp. 57–66. ACM, New York (2010)
36. Marge, M., Banerjee, S., Rudnicky, A.I.: Using the amazon mechanical turk for transcription of spoken language. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 5270–5273, 2010
37. Michlmayr, E., Cayzer, S.: Learning user profiles from tagging data and leveraging them for personal(ized) information access. In: Tagging and Metadata for Social Information Organization, Workshop, WWW07, 2007
38. Munro, R., Bethard, S., Kuperman, V., Lai, V.T., Melnick, R., Potts,C., Schnoebelen, T., Tily, H.: Crowdsourcing and language studies: the new generation of linguistic data. In: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, CSLDAMT '10, pp. 122–130. Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
39. oDesk: http://www.odesk.com/ (2012). Accessed 20 Aug
40. Page, L., Brin, S., Motwani, R., Winograd, T.: Bringing order to the web. The pagerank citation ranking (1999)
41. Parameswaran, A., Park, H., Garcia-Molina, H., Polyzotis, N., Widom, J.: Deco: declarative crowdsourcing. Technical report, Stanford University (2011)
42. Psaier, H., Skopik, F., Schall, D., Dustdar, S.: Resource and agreement management in dynamic crowdcomputing environments. In: EDOC, pp. 193–202. IEEE Computer Society, Washington, DC (2011)
43. Quinn, A.J., Bederson, B.B.: Human computation: a survey and taxonomy of a growing field. In: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, CHI '11, pp. 1403–1412. ACM, New York (2011)
44. Romesburg, C.: Cluster Analysis for Researchers. Krieger, Florida (2004)
45. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. PNAS **105**, 1118 (2008)
46. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)
47. Samasource: http://samasource.org/ (2012). Accessed 20 Aug
48. Satzger, B., Psaier, H., Schall, D., Dustdar, S.: Stimulating skill evolution in market-based crowdsourcing. In: BPM, pp. 66–82. Springer, Berlin (2011)
49. Schall, D.: A human-centric runtime framework for mixed service-oriented systems. Distributed and Parallel Databases, **29**, 333–360 (2011). doi:10.1007/s10619-011-7081-z(Springer, Berlin)
50. Schall, D.: Expertise ranking using activity and contextual link measures. Data Knowl. Eng. **71**(1), 92–113 (2012). doi:10.1016/j.datak.2011.08.001

51. Schall, D., Skopik, F.: An analysis of the structure and dynamics of large-scale q/a communities. In: Eder, J., Bieliková, M., Tjoa, A.M. (eds.) ADBIS, Lecture Notes in Computer Science, vol. 6909, pp. 285–301. Springer, Berlin (2011)
52. Schall, D., Skopik, F., Psaier, H., Dustdar, S.: Bridging socially-enhanced virtual communities. In: Chu, W.C., Wong, W.E., Palakal, M.J., Hung, C.-C. (eds.) SAC, pp. 792–799. ACM, New York (2011)
53. Shepitsen, A., Gemmell, J., Mobasher, B., Burke, R.: Personalized recommendation in social tagging systems using hierarchical clustering. In: Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys'08, pp. 259–266. ACM, New York (2008)
54. Sigurbjörnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: Proceedings of the 17th International Conference on World Wide Web, WWW'08, pp. 327–336. ACM, New York (2008)
55. Skopik, F., Schall, D., Dustdar, S.: Start trusting strangers? bootstrapping and prediction of trust. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE, Lecture Notes in Computer Science, vol. 5802, pp. 275–289. Springer, Berlin (2009)
56. SmartSheet. http://www.smartsheet.com/ (2012). Accessed 20 Aug
57. SpeechInk. http://www.speechink.com/ (2012). Accessed 20 Aug
58. Vukovic, M.: Crowdsourcing for enterprises. In: Proceedings of the 2009 Congress on Services-I, SERVICES '09, pp. 686–692. IEEE Computer Society, Washington, DC (2009)

http://www.springer.com/978-1-4614-5955-2

Service-Oriented Crowdsourcing
Architecture, Protocols and Algorithms
Schall, D.
2012, XI, 94 p. 30 illus., Softcover
ISBN: 978-1-4614-5955-2