

Human computation for data, information, and knowledge management

Nguyen Thanh Tam

ABSTRACT

The paradigm of Human Computation has grown rapidly in recent years and has thus sparked great interest in both the industry and the research community. In this survey, we give an overview of the state-of-the-art of human computation in the context of data, information, and knowledge management (DIKM). On the one hand, we study how to use human intelligence to solve computation problems in DIKM applications such as data acquisition, data analysis, data curation, data storage, and data usage. On the other hand, we study how to help computational systems to solve human problems by understanding human needs. Towards this goal, we cover the kinds of human-computation systems; the various techniques for design and dissemination of tasks for human interaction; the methods employed for reconciling human inputs against given tasks and determining the quality of those inputs; and the various kinds of applications based on human computation.

1. INTRODUCTION TO HUMAN COMPUTATION

The paradigm of Human Computation has grown rapidly in recent years and has thus sparked great interest in both the industry and the research community. In this survey, we characterize human computation along two broad themes, combining human and computer abilities to address two generic tasks: 1) Using human intelligence to solve computational problems; and 2) Helping computer systems understand human needs so that they can assist them better.

Computers have long been used to solve a wide range of problems which require fast computation or need to be performed repeatedly. However, despite the advances in computational capacity, we still need human involvement in many computational problems that are beyond the scope of existing artificial intelligence and machine learning techniques. To motivate the need of human computation, a large body of work in the literature has delineated several reasons that could be summarized as follows:

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ACM X-XXXXX-XX-X/XX/XX.

First, a computer is simply a general-purpose device that is programmed to follow a set of pre-defined human instructions. Hence, as long as there exist certain classes of problems whose solutions cannot be written down programmatically, the need for human intelligence is unavoidable. This can be clearly seen, for example, in the domain of visual recognition. Although many automatic heuristics have been developed, there is no single general purpose solution. In such a scenario, employing human participants can provide better accuracy. A practical example tool has been presented in the literature [56], in which ‘Clickworkers’ (human volunteers) were shown images of the surface of Mars, and asked to mark the craters (something that could not be recognized correctly by computers). Some other domains which need human computation include audio recognition, understanding natural language, market analysis, etc.

Second, computers and their applications are designed to serve humanity. Since only we ourselves have correct understanding of what we really want, involving humans in computation is a natural way to ensure the quality and user satisfaction of applications. Examples can be found in the form of recommender systems such as IMDB [84], in which the movies need to be rated for the purposes of querying and recommending movies to users. Since movie preference varies from person to person, it is intuitively impossible to develop automatic rating algorithms that truly match human expectations. Instead, the system lets users provide ratings for movies and then aggregates all personal ratings to obtain a public rating for each movie, for the benefit of satisfying user intent.

Third, in the scientific and engineering domains there are many problems where the ground truth information is not easily available, and human answers can be used for scientific corroboration. For example, one can employ experts to validate the output of automatic tools [82]. Another example is collecting human inputs as training data for machine learning techniques [99] and for gathering pollution data [96].

Fourth, the rapid expansion of information and communication technologies has opened unprecedented opportunities for massive collaboration among people from across the world, something which was not possible earlier. Especially with the emergence of online and mobile platforms such as crowdsourcing [25], social network [8], and participatory sensing [16], anyone with access to the Internet can contribute to tackle problems which cannot be solved by solitary individuals. In crowdsourcing platforms such as Amazon Mechanical Turk, a worker can perform “human intelligence tasks” such as image labeling and text annota-

tion [64] among many others. In social networks, human activities and profiles can be used to discover social trends or to predict epidemics [40]. And in participatory sensing, humans can act as sensors to provide public information at large scales via personal smartphones [16].

All these systems are relatively new and have opened up exciting venues for the cooperation of ordinary human beings for carrying out computation tasks, and have also drawn spectacular attentions in the research community. Quinn et. al. [88] tries to position the human computation against other related topics such as crowdsourcing, collective intelligence, and social computing. Kittur et. al. [66] promotes the crowdsourcing as an alternative workplace for future generations and sketch out major challenges of implementing this promotion. Doan et. al. [25] classifies crowdsourcing systems in the world-wide-web and focuses on how these systems are correlated to each other. Law et. al. [69] covers the basic definitions and core research questions of different aspects in human computation but only provides baseline methods as references. However, to the best of knowledge, a systematic survey which covers the kinds of human-computation systems; the various techniques for design and dissemination of tasks for human consumption; the methods employed for aggregating human inputs against given tasks and determining the quality of those inputs; and the various kinds of applications based on human computation; has not been seriously undertaken.

We aim to fill this gap and in this paper, we give an overview of the state-of-the-art of human computation in the context of data, information, and knowledge management (DIKM). Specifically, we thematize our classification of human computation problems, systems, and techniques according to the function characteristics of the DIKM tasks that can be computed using human input. The remaining sections of this paper are organized as follows. Section 2 presents existing human computation systems. Section 3 reviews the well-known human computation problems and their models. Section 4 describes extensive methodologies to design, post, and control human computation tasks for the purposes of maximizing output quality under resource constraints (e.g. time, cost). Finally, Section 5 overviews applications that require human computation in the domain of data, information, and knowledge management.

2. KINDS OF HUMAN COMPUTATION SYSTEMS

Existing human computation systems can be categorized as follows.

2.1 Micro-work Systems

In micro-work systems, a crowd of users is employed to complete small tasks (ranging in time duration from a few seconds to a few minutes) for fair amounts of incentives. These systems provide an opportunity for time and money work that would otherwise be accomplished by a handful of hired experts, to be completed in a fraction of the time and money by a crowd of ordinary humans. For example, Amazon Mechanical Turk (AMT) is a well-known micro-work system with millions of active crowd workers and hundreds of thousands of human computation tasks. In Amazon Mechanical Turk (AMT), tasks range from labeling images with keywords to judging the relevance of search results and lin-

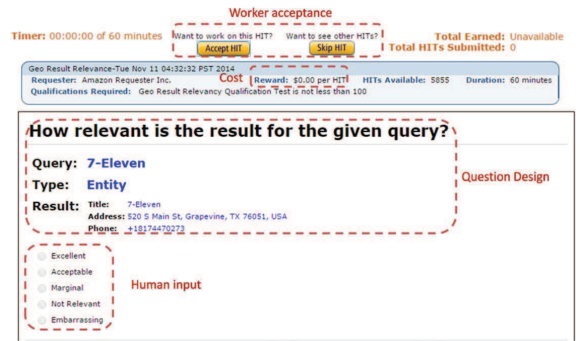


Figure 1: Micro-task on Amazon Mechanical Turk

guistic jobs (e.g. translate, proof-reading). Figure 1 illustrates a typical micro-task on AMT. Although workers from any country can work on tasks on AMT, only US and Indian workers can receive money directly in their bank accounts. Therefore, the majority of workers on AMT are US and Indian citizens. Moreover, the workers are young since more than 50% of AMT workers have age below 34 [92]. In addition, the workers keep getting younger as the average age decreases through time. The workers on AMT are highly educated since most of them have an undergraduate degree or higher [92]. Other well-known micro-work platforms include CrowdFlower, CloudCrowd, etc.

Implicit human computation also involves the completion of micro-tasks by crowds of human users. In implicit human computation users solve a problem as a side effect (passively) of something else they are doing. The ESP Game [109] provides an example of an implicit HC system that allows people to label images while enjoying themselves. In this game, the participant labels an input image by a keyword, which most properly describes the image, from a set of provided keywords. The ultimate goal is to obtain proper labels for each image. This effort is part of the larger goal of collecting proper labels for images on the Web, which would be an invaluable for information retrieval applications.

Another example is reCAPTCHA [110], which is a piggy-back HC system built on top of CAPTCHA (used by websites to prevent spam). By solving the CAPTCHA, users implicitly perform OCR tasks, such as digitizing books, newspapers and old time radio. In a reCAPTCHA task, a user is presented with two words. One word serves as a conventional CAPTCHA, while the other word cannot be recognized by automatic OCR techniques. If a user recognizes the recognized word, the answer to the unrecognized word is assumed to be correct, and is collected as training data for further OCR tools.

2.2 Social-based Systems

Social-based human computation systems encourage millions of people over the world to contribute to human computation problems via the Internet. With the growth of Web 2.0 technologies, there are many kinds of works that can be performed by people. The first type of social-based HC systems can be described as ‘Knowledge-Base’. Wikipedia is a well-known example of human computation knowledge-base, which has thousands of editors to continually edit articles and contribute knowledge for building the world’s most comprehensive free encyclopedia. The writing is distributed

and open in that essentially almost anyone who has access to the Internet can contribute.

Q&A sites that allow users to post questions, provides answer, and edit and organize the constructed information, also fall under the category of Social-based HC systems. A typical example is Yahoo! Answers, which is a general Q&A forum. The human computation in such systems involves answering the questions and the incentives include social benefits such as prestige, fun and social networking. The collected human answers provide a large body of human-knowledge data, that can be used for solving further AI problems. Other example Q&A sites are ask.fm, Quora, etc.

Some social-based HC systems take the form of competitions. In contrast to micro-tasks, HC competitions aim to solve complex problems and offer high prices. Instead of collaborating, human participants compete with each other to achieve higher ranks. The collected human solutions offer a great variety of ways to solve a particular computational problem, thus not only enhancing overall knowledge but also changing the way computers approach the problem. A well-known example is Topcoder.com, which offers various computer science problems, ranging from algorithms to software design and development. Another example is the Goldcorp Challenge ¹, which employs geological experts from all over the world to identify the locations of gold deposits.

‘Crowdfunding’ and ‘Skill Markets’ are yet other types of social-based HC systems. Crowdfunding is an HC-based strategy for funding one’s projects by asking a multitude of people to contribute small amounts of money instead of seeking huge contributions from a few big investors. The advantage of crowdfunding is that it is easy for people to invest a small amount of money. A multi-level payment mechanism is often applied, in which the more one contributes, the more rewards one gets such as e.g., souvenirs, progress updates, first copy of the product at discounted price, etc. In ‘Skill Markets’, people work as freelancers to complete jobs. Example marketplaces include Elance, ODesk, and Freelancer.com, where millions of freelancers do various kinds of work.

It is worth noting that social-network services are also considered as social-based human computation systems due to their long-existing nature of human computation. They enable the wisdom of the crowd efficiently by providing fundamental infrastructure to employ a mass amount of user in a short period of time. As such, social-network services are born a platform qualified not only for spreading human communications, but also for human computation tasks. For example in [17], the authors leverage a micro-blog service (i.e. Twitter) to collect answers for decision making questions by actively distributing the questions to workers via the “@” markup. However, social network services are only best-fit for knowledge collection. This is because it is often difficult to build a payment mechanism on top of social networks. As such, social networks cannot be used for micro-tasks that require monetary payment.

2.3 Pervasive Systems

Pervasive systems make use of activities that human beings perform in their daily lives to solve computational problems. With the rapid uptake of mobile technologies, we can access human computation power via smart phones to im-

¹<http://www.goldcorpchallenge.com/>

plement pervasive systems. The first type of such systems include community-based traffic navigation platforms (e.g. Waze), also called geosocial networks. In Waze [32], the human participants are drivers who report real-time traffic and road information such as accidents, traffic jams, speed-traps, and nearby police units. All this information is publicly shared among drivers for the purposes of routing and navigation. The Waze community has millions of active users mainly across Europe, Asia, and North America. Similarly, Google Maps and Google Earth also employ human-powered traffic information for various data visualization purposes.

Human-powered newspapers serve as another example of pervasive HC systems. Instead of using professional reporters, local people are employed to report rumours and stories in their communities. An example is Ushahidi [85], in which people in crisis situations (e.g., in disaster and conflict zones) submit their reports through the web and mobile phones. These reports are then aggregated and organized temporally and geospatially to give a general view of emerging situations. There are some distinct advantages of this approach. One is the relatively faster reporting of information as compared to traditional methods since professional journalists are limited in number. Also, in human powered newspapers, the information is untamed and covers different points of views of human participants, as opposed to the point of view of a single individual (the professional reporter).

Participatory sensing systems are also pervasive HC systems, in which people equipped with sensors, e.g., built in their smartphones, measure environmental conditions. An example is Common Sense [27], which is a human-power pollution monitoring application. Common Sense uses specialized handheld air quality sensing devices, which are deployed across a large number of human participants, to collectively measure the air quality of an area. Similarly, we can apply the same method to monitor other environmental conditions such as noise and water.

2.4 Management Systems

Management systems are general-purpose human computation systems that are designed to manage the entire human computation process, including designing and posting tasks, collecting and aggregating human inputs, and performing further analyses. The first example is CrowdDB [33], which aims to develop a declarative language to express the logic of the expected human computation task instead of describing it in natural language. CrowdDB employs human power via crowdsourcing to answer the uncertain queries that cannot be processed by automatic engines. For example, we can write the following query to retrieve all computer science departments across universities:

```
SELECT * FROM departments
WHERE name ~="CS";
```

However, different computer science departments in different universities could have different names or even be in different languages. As such, we need humans to perform matching between department entities. Based on the query, CrowdDB will automatically generate human computation tasks in HTML code and post them in Amazon Mechanical Turk. After collecting crowd answers, CrowdDB will

aggregate them to produce the final query result.

Another example is CrowdForge [65], which aims to manage human computation workflow, including decomposing large tasks into small ones, and assigning these tasks to human participants. Both dynamic and fixed workflows are supported to allow the parallelization of human computation tasks. More precisely, CrowdForge employs a MapReduce-like model to post micro-tasks into crowdsourcing platforms such as Amazon Mechanical Turk. A sample complex task studied is article writing [65], in which an article is partitioned into different Map tasks, such as collecting and writing facts about an entity. After all facts are collected, a Reduce task is performed by human users to combine the facts into one paragraph. For quality control purposes, CrowdForge uses majority voting to determine the best write-ups for the article.

As an industrial example, CrowdFlower [104] supports integrating human computation into business processes by offering three important features: workflows, taxonomy, and quality control. First, CrowdFlower help users define the workflows by pre-designing job templates for crowdsourcing tasks such as image categorization, text transcription, and sentiment analysis. On top of the user-defined workflows, the system will automatically route and process data between multiple CrowdFlower jobs and/or external services. Second, CrowdFlower help users manage a large number of jobs hierarchically by letting them define a taxonomy of tags and indexing the CrowdFlower jobs by these tags. Based on the tag index, users can search the jobs efficiently. Third, CrowdFlower allows to control the quality of workers by using test questions (whose answers are known before-hand) to discard the answers of workers who do not substantially pass the test questions. Moreover, the system also supports peer review and provides statistical reports on the outcome of worker answers, for the purposes of evaluating and profiling workers.

3. CLASSIFICATION OF HC PROBLEMS AND QUALITY CONTROL TECHNIQUES

In this section we will classify the general kinds of problems that humans can help solve in current human computation systems, and the techniques used to control the quality of human performance.

3.1 Discrete-function Problems

The techniques and models that fall in this category make use of human abilities for computation to solve classification problems. In general, the output of a classification problem consists of possible labels assigned for each object, given a set of existing objects. Document labeling, image tagging, relevance feedback, are all classification tasks that are typically completed online via crowdsourcing marketplaces. Using humans as labelers is particularly beneficial in various settings where ground truth exists but is unknown or the problem itself is subject to human judgements.

It is worth noting that many data integration and information extraction problems described in Section 5 can be mapped onto the classification problem. For example, the schema matching problem, where we employ humans to validate the correspondences generated by automatic matching tools [], can be formulated as that correspondence between scheme can be considered to be an object, and there are

two possible classification labels: YES (the correspondence is approved), and NO (the correspondence is disapproved). Similarly, other problems such as ‘entity resolution’ (users are asked to determine whether a given pair of records is duplicate) and ‘entity extraction’ (users are asked to determine whether a URI matches to an existing entity) can be modeled as discrete-function based problem.

Model. A large body of work has studied the problem of aggregating user answers, which can be formulated as follows: There are n objects $\{o_1, \dots, o_n\}$, where each object can be assigned by k users $\{w_1, \dots, w_k\}$ into one of m possible labels $L = \{l_1, l_2, \dots, l_m\}$. The aggregation techniques take as input the set of all user answers that is represented by an *answer matrix*:

$$M = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nk} \end{pmatrix} \quad (1)$$

where $a_{ij} \in L$ is the answer of user w_j for object o_i . The output of aggregation techniques is a set of aggregated values $\{\gamma_{o_1}, \gamma_{o_2}, \dots, \gamma_{o_n}\}$, where $\gamma_{o_i} \in L$ is the unique label assigned for object o_i . In order to compute aggregated values, we first derive the probability of possible aggregations $P(X_{o_i} = l_z)$, where X_{o_i} is a random variable of the aggregated value γ_{o_i} and its domain value is L . Each technique applies different models to estimate these probabilities. For simplicity sake, we denote γ_{o_i} and X_{o_i} as γ_i and X_i , respectively. After obtaining all probabilities, the aggregated value is computed by ²:

$$\gamma_i = \operatorname{argmax}_{l_z \in L} P(X_i = l_z) \quad (2)$$

Techniques. A rich body of research has proposed different techniques for answer aggregation. In what follows, we describe the most representative techniques for answer aggregation.

- *Majority Decision (MD)*: is a straightforward method that aggregates each object independently [68]. Given an object o_i , among k received answers for o_i , we count the number of answers for each possible label l_z . The probability $P(X_i = l_z)$ of a label l_z is the percentage of its count over k ; i.e. $P(X_i = l_z) = \frac{1}{k} \sum_{j=1}^k \mathbb{1}_{a_{ij}=l_z}$. However, MD does not take into account the fact that users might have different levels of expertise and it is especially problematic if most of them are malicious (e.g. spammers).
- *Filtering*: In general, filtering techniques try to detect low-quality users and minimize the ill-effects of their answers to the results. There are two well-known techniques in the literature, namely Honeypot and ELICE. In principle, Honeypot (HP) [71] operates as MD, except that untrustworthy users are filtered in a preprocessing step. In this step, HP merges a set of trapping questions Ω (whose true answer is already known) into original questions randomly. Users who fail to answer a specified number of trapping questions are neglected as spammers and removed. Then, the

²Note that $\sum_{l_z \in L} P(X_i = l_z) = 1$

probability of a possible label assigned for each object o_i is computed by MD among remaining users. However, this approach has some disadvantages: Ω is not always available or is often constructed subjectively; i.e. truthful users might be misidentified as spammers if trapping questions are too difficult.

Expert Label Injected Crowd Estimation (ELICE) [62] is an extension of HP. ELICE also uses trapping questions Ω , but to estimate the expertise level of each user, it calculates the ratio of her answers which are identical to the correct answers of Ω . Then, it estimates the difficulty level of each question by the expected number of users who correctly answer a specified number of the trapping questions. Finally, it computes the object probability $P(X_i = l_z)$ by *logistic regression* [45] that is widely applied in machine learning. In brief, ELICE considers not only the user expertise ($\alpha \in [-1, 1]$) but also the question difficulty ($\beta \in [0, 1]$). The benefit is that each answer is weighted by the user expertise and the question difficulty; and thus, the object probability $P(X_i = l_z)$ is well-adjusted. However, ELICE also has the same disadvantages regarding the trapping set Ω as previously described in the case of HP.

- *Maximum Likelihood*: This approach tries to transform the aggregation problem into a maximum likelihood formulation, in which human inputs are sample values and the aggregated results are parameters to be estimated. There are two well-known techniques in the literature, namely EM and GLAD. The Expectation Maximization (EM) technique [52] iteratively computes object probabilities in two steps: *expectation* (E) and *maximization* (M). In the (E) step, object probabilities are estimated by weighing the answers of workers according to the current estimates of their expertise. In the (M) step, the expertise of workers is re-estimated based on the current probability of each object. This iteration is repeated until all object probabilities are unchanged. Simply, EM is an iterative algorithm that aggregates many objects at the same time. Since it takes a lot of steps to reach convergence, the running time is a critical issue.

Generative model of Labels, Abilities, and Difficulties (GLAD) [117] is an extension of EM. This technique takes into account not only the worker expertise but also the question difficulty of each object. It tries to capture two special cases. The first case is when a question is answered by many workers, the natural assumption is that the workers with high expertise have a higher probability of answering correctly. Another case is when a worker answers many questions, the question with high difficulty has a lower probability of being answered correctly. In general, GLAD as well as EM-based approaches are sensitive to arbitrary initializations. Particularly, GLAD’s performance depends on the initial value of user expertise α and question difficulty β . In fact, there is no theoretical analysis for the performance guarantees and it is necessary to have a benchmark for evaluating different techniques in the same setting.

- *Learning*: This approach applies learning techniques to simultaneously estimate both the difficulty level of

human computation tasks and the expertise of users to compute the aggregated results. There are two techniques in the literature, namely SLME and ITER. In principle, Supervised Learning from Multiple Experts (SLME) [89] also operates as EM, but characterizes the user expertise by *sensitivity* and *specificity*—two well-known measures from statistics—instead of the confusion matrix. Sensitivity is the ratio of positive answers which are correctly assigned, while specificity is the ratio of negative answers which are correctly assigned. One disadvantage of SLME is that it is incompatible with multiple labels since the sensitivity and specificity are defined only for binary labeling (aggregated value $\gamma \in \{0, 1\}$). Iterative Learning (ITER) is an iterative technique based on standard belief propagation [59]. It also estimates the question difficulty and the user expertise, but is slightly different in details. While others treat the reliability of all answers of one user as a single value (i.e. user expertise), ITER computes the reliability of each answer separately. And the difficulty level of each question is also computed individually for each user. As a result, the expertise of each user is estimated as the sum of the reliability of her answers weighted by the difficulty of associated questions. One advantage of ITER is that it does not depend on the initialization of model parameters (answer reliability, question difficulty). Moreover, while other techniques often assume users must answer all questions, ITER can divide questions into different subsets and the outputs of these subsets are propagated in the end.

3.2 Continuous-function Problems

In this section, we study another class of problems, in which user inputs are real values, instead of discrete labels as in the classification problem. Several applications mentioned in Section 5 can be mapped to this problem formulation such as participatory sensing and measuring.

Model. A large body of work in this category has studied the rating aggregation problem, which is formulated as follows. There are n objects $\{o_1, \dots, o_n\}$, where each object can be rated by k users $\{u_1, \dots, u_k\}$ into a rating score r . The domain value of r could be real numbers, a predefined scale (e.g. from 1 to 10) [35], bounding boxes [116], etc. In other words, the problem input is the set of all user scores that is represented by a score matrix:

$$R = \begin{pmatrix} r_{11} & \dots & r_{1k} \\ \vdots & \ddots & \vdots \\ r_{n1} & \dots & r_{nk} \end{pmatrix} \quad (3)$$

where r_{ij} is the input score of user u_j for object o_i . The problem output is a set of aggregated scores $\{\gamma_{o_1}, \gamma_{o_2}, \dots, \gamma_{o_n}\}$, where γ_{o_i} is the final score assigned for object o_i .

Techniques. A rich body of research has proposed different techniques for rating aggregation. In what follows, we describe the most representative ones.

- *Average Approach*: The simplest approach to compute the aggregated scores is using the average: $\gamma_{o_i} = \frac{\sum_{j=1}^k r_{ij}}{k}$. This approach considers each object independently and does not take into account the differences (e.g. trustworthy, expertise) between users. Another disadvantage of this approach is the sensitivity to

outliers, e.g., some users could provide very high rating scores or very low rating scores. An improved version to avoid the outlier sensitivity is Beta Model [54], in which 5% upper scores and 5% lower scores of the score matrix will be excluded from the aggregation.

- *Trust-based Approach:* The average approach assumed that the user inputs are trustworthy and the provided values are equally reliable. However, in several contexts such as spam or malicious users, the aggregated scores could be wrong or misleading. To overcome this problem, the trust-based techniques associate each user u with a trust score $t(u)$, indicating the trustworthiness level of the user; i.e., the probability of u provides correct input. Moreover, each object o is also assigned a trust score indicating the trustworthiness of the object; i.e., the probability of the final score of o being correct. The idea is that a rating score is likely to be true if it is provided by trustworthy users, and a user is trustworthy if most rating scores he or she provides are trusted. Based on such inter-dependency between objects and users, an iterative procedure is often performed until convergence is reached, to compute the trust scores. To implement this procedure, a wide range of models have been proposed in the literature, such as factor-graph [23] and maximum likelihood estimation [107]. By using the trust scores, we can determine the final value of each item by using a weighted average computation of its values, where the weights are trustworthiness values of users.
- *Sampling-based Approach:* In practical settings, to obtain the true final rating, we need to employ a large number of users. Since humans make mistakes, their individual errors could be complemented with each other to produce the result very close to ground truth [100]. For example, a movie having an aggregated rating score of 8/10 over 100 users might be worse than another having an aggregated rating score of 7/10 over 10000 users. However, due to limited budget of time and cost, we can only obtain a small set of users.
To address this issue, the techniques under this category consider the obtained rating scores as a set of (random) sampling inputs over all possible users. As such, the aggregation of sampling scores is an estimation of the true rating scores for the objects. To measure the estimation error, a wide range of statistical and probabilistic models can be employed. For example, in [39], the authors simply compute the mean and standard variation of user scores, then perform ANOVA and t -test to compute the error rate. Another example is in [4], the authors compute the estimation error via the numerical integration of multivariate normal distributions, since the mean of sampling rating scores is considered as the sampling of a normal distribution (due to central limit theorem).
- *Knowledge-based Approach:* On top of the above techniques, one can develop further extensions by exploiting prior knowledge (e.g., user profiles, rating history, object features). For example, we can compute the trustworthiness of users from their rating history us-

ing the models in [114, 115]. Then, these apriori trustworthiness values can be incorporated into trust-based techniques. Another example is taking into account various data-level aspects (e.g., object similarity, object relationship), which can be computed from the pre-defined attributes of the objects [20]. Then, we could perform clustering to divide the objects into clusters, using similarity measures such as Pearson’s correlation coefficient and Jaccard metric [35]. The rating scores of the objects can be adjusted by clustering them together (e.g., the difference between rating scores in the same cluster cannot be relatively larger than the diameter of that cluster). Moreover, we can also leverage the collaborative filtering paradigm [95], whose idea is that a user is likely to provide close rating scores for similar objects and similar users are likely to provide close rating scores for the same object. The similarity between users and objects can be computed via user profiles and object features.

3.3 Partial-function Problems

In the two previous sections, we have studied the aggregation problems, whose output is to determine the final discrete/continuous values of given objects. In this section, we study another class of problems in which we try to compute the relationships between objects. In general, there are two typical problems studied in the literature as follows: (i) association rule aggregation – studies the appearance frequency between subsets, (ii) ranking aggregation – studies the ordering between objects.

It is noteworthy that several applications that we will discuss in Section 5 can be built on top of these models such as recommender systems, in which users provide ratings for artifacts (e.g. movies, books, musics) and their ratings are aggregated into a unified rating for the purposes of querying and recommending.

3.3.1 Association Rule Aggregation

Model. Let U be a set of human users, and let $I = \{i_1, i_2, \dots\}$ be a non-empty finite set of unique items. Let $0 \leq \theta_s, \theta_c \leq 1$ are predetermined threshold values for support and confidence. An input of user $u \in U$ is an association rule $r = A \rightarrow B$ with user support u_c and user confidence u_s , where $A, B \subseteq I$, $A \cap B = \emptyset$. Moreover, a rule is defined to be significant in U iff the average of support values and the average of confidence values given by all users in U are greater than θ_s and θ_c , respectively. However, in practice the number of relevant users and rules could be huge. Due to limited budget of time and cost, we can only ask a small sampling set of random users. As such, we need to estimate the significance of a rule given a small set of users.

Technique. In [4], the authors propose a sampled-based estimation to determine whether a rule is significant given a subset of sampling users. More precisely, for a particular rule $r = A \rightarrow B$, we have obtained so far the answers of n users: $(s_1, c_1), \dots, (s_n, c_n)$, where (s_i, c_i) is the pair of support and confidence values given by user u_i . There are two challenges. First, since we only have a sampling set of users, using the sample mean of support and confidence values to determine the rule significance might be misleading; i.e., $avg(s_i) \geq \theta_s$ and $avg(c_i) \geq \theta_c$ do not mean the rule r is truly significant. Second, the support and confidence values are dependent

since a user input is a support-confidence pair. In other words, we cannot consider the rule significance by treating the support and confidence thresholds individually. To overcome these challenges, the authors propose to estimate the bivariate probability distributions of the support and confidence of the rules. In other words, the bivariate probability p_r of rule r indicates likelihood that the true mean of support and confidence values of r greater than (θ_s, θ_c) . The idea is that if $p_r \geq 0.5$, r is significant; otherwise, r is not significant. The error rate of this conclusion is $1 - p_r$. Technically, they build a bivariate normal distribution between support and confidence values and compute the probability as the numerical integration of the distribution [36].

3.3.2 Ranking Aggregation

Model. Ranking (voting, rating) aggregation is the problem of combining different user ordering preferences on a limited set of items. Formally: Let $U = \{u_1, \dots, u_k\}$ be a set of human users, and let O be a non-empty finite set of unique items. An input of a user u_i is an ordering τ_i of a subset $S_i \subseteq O$; i.e., $\tau_i = [o_1 \geq o_2 \geq \dots \geq o_n]$, with each $o_j \in S$ and \geq is some ordering relation on S . τ_i is also called a rank list on O ; i.e. $\tau_i(o_j) = j$ is the rank of o_j w.r.t. τ_i . Let $|\tau_i|$ denote the number of elements in τ_i . τ_i might not be a full list; i.e. $|\tau_i| < |O|$. Denote $R = \{\tau_1, \dots, \tau_k\}$ is the set of all user rankings. The problem output is to determine an aggregated ranking τ such that τ is a full list over the union of elements of τ_1, \dots, τ_k ; i.e. $S = \cup_{\tau_i \in R} \cup_{o_j \in \tau_i} o_j$.

Techniques. The research efforts on solving the ranking aggregation problem can be categorized into the following methods.

- *Score based:* This approach aggregates the final ranking by computing the ranking scores for each item (higher the score, better the rank). The process consists of following steps. In the first step, for each item $o \in O$, we will compute the normalized ranking scores $w_{\tau_1}(o), \dots, w_{\tau_k}(o)$ over all user rankings $\tau_i \in R$. Several normalization computations [91] include score normalization, Z-score normalization, rank normalization, and Borda rank normalization. In the second step, we will compute the aggregated ranking score $s_\tau(o)$ of the item o by combining its normalized ranking scores; i.e. $s_\tau(o) = f(w_{\tau_1}(o), \dots, w_{\tau_k}(o))$. One simple way to implement the aggregation function $f(\cdot)$ is using the sum, min, or max [70]; e.g. $s_\tau(o) = \sum_{\tau_i \in R} w_{\tau_i}(o)$. A complex implementation is using a weighted version of the sum [70]: $s_\tau(o) = h_R(o) \sum_{\tau_i \in R} w_{\tau_i}(o)$, where $h_R(o)$ is the number of occurrences of o over all user rankings in R with the idea is that the items appear in more user rankings are likely to be more important. In the final step, the aggregated ordering can be simply obtained following the decreasing order of the aggregated ranking scores.
- *Distance based:* This approach computes the ranking aggregation by solving an optimization formulation, in which the objective function is the distance between user orderings. Formally, $D(\cdot)$ is the distance measurement between two or many orderings. The ranking aggregation problem then becomes finding an aggregated ranking τ such that the distance value $D(\tau, \tau_1, \dots, \tau_k)$

is minimal. A wide range of distance measures has been proposed [24], such as Spearman footrule distance (which uses the absolute difference between the ranks of an item according to the given rankings τ_i and τ_j) and Kendall distance (which uses the number of pair-wise adjacent transpositions needed to transform from ranking τ_i to another ranking τ_j).

In general, the optimization formulation of ranking aggregation is intractable (e.g., using the Kendall distance with $k = 4$ is NP-Hard [28]). Therefore, a wide range of important properties that an aggregation solution needs to satisfy have been studied in the literature, such as Condorcet property [121].

- *Probability based:* The methods in this category capture the item ranking via probabilistic interpretation. Technically, for two given items o_i and o_j , we will compute the probability of an item o_i having a greater order of an item o_j ; i.e. $Pr(o_i > o_j)$. Several probabilistic models to compute $Pr(o_i > o_j)$ have been proposed, such as Bradley-Terry model [13] and Thurstone model [101]. The Bradley-Terry model formulates a logistic function over the true ranking scores of o_i and o_j (i.e., $Pr(o_i > o_j) = \frac{e^{s(o_i)}}{e^{s(o_i)} + e^{s(o_j)}}$) and performs a log-likelihood maximization to compute all the pair-wise probability values and the true ranking scores simultaneously. The Thurstone model follows a similar process, in which the ranking score for each item has a Gaussian distribution.

With the similar idea of computing pair-wise ranking probabilities, one can use Markov chains [28] in which the states correspond to the items to be ranked and the transition probability from state i to state j is the probability of the item o_i has a higher order of the item o_j w.r.t. some user rankings $\tau_i \in R$. As such, computing the aggregated ordering is equivalent to determining the stationary probability distribution of the Markov chains, which can be done in polynomial time [28]. Probabilistic models in general and Markov chains in particular not only offer a parameterizable approach but also open ways to integrate different heuristics into the probability formulation (e.g., one could use other distributions rather than Gaussian distribution). As such, the ranking aggregation can be iteratively refined by these heuristics, producing a fine-grained aggregated ranking.

In sum, while the *score-based* focuses on computing a unified ranking score for the ranking, the *distance-based* method aims to minimize the differences between the final ranking and individual ones. Taking advantages of the two, the *probability-based* allows more fine-grained combination of ordering preferences.

3.4 Similarity based Problems

There are many applications whose core features rely on computing the similarity between two objects or the matching degree of an object against pre-defined conditions. An example application is detecting duplicate tuples for the purpose of data repair in databases, in which we need to define the duplication by proposing a similarity measurement between two tuples. Another example application is searching

data tuples against queries, in which we need to define the relevance of search results by capturing the similarity between a tuple and a query.

In general, the output of some similarity-based applications can be mapped onto discrete and continuous function problems. For example, the confirmation questions for validating whether or not two given objects are similar (e.g., scheme matching, image search) can be viewed as a classification problem with two labels – YES/NO. Another example is the rating questions for evaluating the similarity between objects (e.g. sorting, information integration), which can be viewed as a classification problem with multiple labels – less similar/similar/highly similar/etc.

However, some other applications have their own characteristics (e.g., clustering) that can be mapped onto the following formulation.

Model. Clustering/partition aggregation is the problem of finding a unified clustering, given a set of input clusterings from users. Denote $U = \{u_1, \dots, u_k\}$ as the set of users, $O = \{o_1, \dots, o_n\}$ as the set of objects where each object o_i can be represented/transformed into a vector of some multi-dimensional metric space. The set of user inputs is $IP = \{P_1, \dots, P_k\}$, where each P_i is the clustering provided by user u_i . A clustering/partition $P_i = \{C_1^i, C_2^i, \dots\}$ is a set of clusters, where each cluster $C_j^i \subseteq O$. We also denote \mathcal{P} as the set of all possible clusterings w.r.t. O . The problem output is to construct a consensus clustering $P^* \in \mathcal{P}$, which makes the best out of the input clusterings (e.g. in terms of the overlaps between the consensus clustering and individual clustering).

The clustering aggregation problem is often studied in metric spaces, in which the optimization formulation of the problem is:

$$P^* = \operatorname{argmax}_{P \in \mathcal{P}} V(P) = \operatorname{argmax}_{P \in \mathcal{P}} \sum_{i=1}^k S(P, P_i) \quad (4)$$

where $V(P)$ is called the objective value of P and $S(., .)$ is a similarity measure between clusterings. In other words, the consensus clustering is the clustering that maximizes its similarity with all input clusterings.

Techniques. Several similarity measures for clustering aggregation problem have been studied in the literature, such as Mirkin distance [44] (which was also proved to be NP-Complete), Jaccard coefficient [10], and normalized mutual information [98]. Based on these measures, research efforts on clustering aggregation can be categorized as follows.

- *Graph-based:* This kind of method transforms the clustering aggregation problem into a graph/hypergraph partitioning problem [98]. One formulation is constructing a fully connected graph, in which the nodes are the objects O and an edge between two objects is associated with a weight equal to the number of times they appear in the same cluster across the input clusterings IP . Then, the graph partitioning tool METIS [60] can be used to find the consensus clustering. Another formulation is constructing a hypergraph, in which the hypernodes are the objects O and a hyperedge represents a cluster C_j^i . Then, the hypergraph partitioning tool HMETIS [61] can be used, in which the minimal number of hyperedges is removed such that the remaining ones are not overlapping. Other

graph-based methods include Hyperbrid Bipartite Graph Formulation [31], Random Walker [1], meta-clustering [98], etc.

In general, the graph-based methods are popular since they are representative to understand. However, one of their weaknesses is that they are sensitive to the well-known graph partitioning algorithms, which could produce different outputs w.r.t. particular parameter sets.

- *Heuristics-based:* These methods employ out-of-the-box heuristics to solve the clustering aggregation problem. The general idea is that they often start from an initial clustering, then try to improve the clustering via different heuristics (local search, simulated annealing, genetic, etc.). The first heuristic is Best-of-k [41], which starts with an input clustering $P \in IP$ and then replaces with each of the remaining clusters if the objective value $V(P)$ is increased. The second heuristic is local search [106], in which we start with an initial clustering (could be the result of Best-of-k), then generate new clusterings by moving object from one cluster to another. The new clustering can be compared with the previous clustering via the objective value $V(P)$ or defining a changing cost. If there exists a move that generates better clusterings, the algorithm is continued; otherwise, the algorithm stops. The third heuristic is simulated annealing [41], which is used to improve the local search process by trying to avoid local optima. Several other heuristics have also proposed, such as pivot-based algorithm [2] and Balls algorithm [38].

In this category, we also have a wide range of genetic algorithms [73, 120]. In such algorithms, the initial population is the set of input clusterings and we generate new possible clusterings via crossover and mutation steps until the population with highest fitness value is achieved. These genetic algorithms differ in the generation mechanisms (cross, mutation) and the definitions of fitness value. In sum, the heuristic-based methods also endure the issue of being sensitive to parameters. Although theoretical approximation bounds are provided, there is no absolute winner for all datasets and settings.

- *Probability-based:* The methods in this category view the clustering as assigning labels to the objects. In other words, the objects with the same label belongs to the same cluster. Then, finding the consensus clustering is interpreted under probability terms as computing, for each object o_i , the probability that o_i is assigned to a label c_i . Formally, we would like to compute $p(o_i = c_i | \Theta)$, where Θ is the parameter set derived from the input clusterings IP . As such, the clustering aggregation becomes a maximum likelihood estimation problem, in which the label set $\{c_1, c_2, \dots\}$ and the parameter set Θ is estimated simultaneously to maximize their log-likelihood function [103]. In the literature, the maximal likelihood problem is often solved by the expectation-maximization algorithm [76], which repeats two steps (E-step and M-step) until convergence. The idea is that while the E-step computes the

expected values of the current parameters, the M-step maximizes the current likelihood value by replacing a new and better parameter set.

In particular, the probability-based methods can deal with partial input clusterings, where users can only perform clustering on a subset of objects [42]. This is because the original object set O is often large and the resource budget (time, cost) for employing users is limited. In this case, the other methods in this category might not be applicable since two partial input clusterings can have no common object.

- *Others:* There are also miscellaneous methods proposed for specific cases of the clustering aggregation problem. In [26], the authors designed locally adaptive clustering algorithms to work with numerical data. In [102], the authors define the similarity measure as a category utility function, which is based on information theory and works for categorical data. In [105], the authors studied a weighted version of the problem, by assigning each input clustering with a weight. This is practically useful since the clusterings of different users might not have the same quality. For example, the weights can be used to capture the user reliability or the confidence of users on the quality of their input clusterings. In [87], the authors studied fuzzy-version of clustering aggregation. Similar to probability-based methods, the idea of using fuzzy clustering is to soften the clusterings where the objects are associated with fuzzy numbers instead of being fixed into clusters.

4. TASK DESIGN AND DISSEMINATION

In human computation, human users participate in computation problems by providing inputs for human computation tasks. In this section, we study different aspects of human computation tasks, including task design, task workflows, and task posting. Task design studies the way to design human computation tasks that can positively affect human inputs (e.g., motivating human users to provide correct answers). Task workflow studies how to decompose a big problem into subtasks that can be distributed effectively to human participants. Task posting studies different strategies to elicit human inputs that can achieve high-quality output while saving the computational resources (e.g., time, cost)

4.1 Task Design

A task can be viewed as a piece of work where human users work to solve a certain computational problem. In general, a task can be modeled as a tuple $\langle I, c \rangle$ where I is the basic information and c is the reward to pay human participants. Basic information includes inputs, what is being computed, and possible outputs. Formally, $I = \langle O, Q, R, \rangle$ where O is a set of objects, Q is the question being asked about the object(s), and R is the domain of possible outputs.

4.1.1 Question

In the literature, there are two types of questions that are often considered [4]:

- *Closed questions:* In this kind of question, we concretely define the object O and the output domain R .

The answers are often designed in the form of multiple-choice lists to make sure that the user provides an answer $\in R$.

- *Open questions:* In this kind of question, users define the object themselves and provide the corresponding answer. Users may record their answers in natural language, which requires *Natural Language Processing* (NLP) tools for interpreting the answer.

While closed questions avoid extra cost for answer processing and are robust to invalid answers, open questions are useful for enriching the knowledge base by allowing users to extend the set of objects.

4.1.2 Incentives

Incentives refer to how human workers will be rewarded (or penalized) for the successful (or failed) execution of a computational task. For example, in Amazon Mechanical Turk, we often find many tasks with payment 0.1\$ per user. Human users are not a homogeneous group and can have wide-ranging levels of expertise, experiences, and motives. Users seek to understand which activities are rewarded and whether those rewards are worthy of their knowledge and effort. Thus, it is necessary to reward users properly, according to their level of effort and expertise. There are several ways to motivate users to participate and complete their tasks effectively, including financial incentives, non-financial incentives, and hybrid incentives. Financial incentives are the method of rewarding users via monetary payment [97] (e.g. by transferring directly to a bank account or using digital currency such as bitcoin). Non-financial incentives involve virtual rewards (e.g. points, badges), user satisfaction, social recognition and the feeling of contributing towards the greater good [90]. Hybrid incentives combine both financial rewards and non-financial rewards in a systematic way [97]. Many research studies have shown the effect of these incentive strategies on the performance of human work, especially crowdsourcing. Moreover, literature also suggests that designers should understand variations in human users behaviors (e.g. competence, enjoyment, and autonomy) [37]; therefore, many works tackle the challenge of clearly understanding the desired behavior of users and designing the rewards accordingly [6, 9].

4.1.3 Requirements for Effective Tasks

To achieve good results, tasks need to be properly designed so that the human participants are actually computing the correct results under the influence of appropriate incentives. Many practical design requirements/guidelines have been concluded in the literature, such as:

- (R1) Graphical interface: the task requires a graphical user interface with human-readable instructions. The layout and design of the interface can have a direct effect on the speed and accuracy with which people complete the tasks [122].
- (R2) Context information: it is a best practice to provide context information. More context information enables humans to better understand the task and spend their effort [67].
- (R3) Individually workable: the tasks need to be atomic and workable by humans. In the literature, large tasks are

often decomposed into smaller tasks for multiple participants. Thus, designers need to show the position of each task in the overall decomposition. By knowing the information as a whole, the human participants can adequately solve their tasks [67].

- (R4) Communication language: we need to visually separate task language from context language. Workers often have difficulty identifying the true purpose of the task in a complex question they were being asked to carry out. For example, the authors in [67] used background colors to emphasize and separate the primary task instructions from explanatory and context information.
- (R5) Task complexity: Higher task complexity drastically discourages malicious workers from attempting to cheat [29]. In other words, task must be designed so that giving spam or malicious answers must take approximately equal time to giving correct answers. As such, short completion duration may indicate a malicious worker.
- (R6) Variance of task: Greater variability and more context changes discourage malicious workers as the task appears less susceptible to automation or cheating, in other words less profitable [29].
- (R7) Self-reporting, verifiable questions, and task duration: Add self-report response from workers prevents cheating since an inconsistent self-report response could indicate a malicious worker. Short completion time may also indicate a malicious worker. Moreover, the tasks must have verifiable questions to test the worker reliability [64].

4.2 Task Workflows

A complex problem should be divided into a collection of tasks where one or many workers can participate. These tasks might depend on each other, motivating the need for decomposing tasks and assembling them into a workflow. Figure 2 presents a general workflow that integrates various research challenges. One typical research challenge is how to effectively design the workflow to avoid overhead costs and overlapping works between human participants.

- *Fixed Workflow*: the tasks are performed in a predefined order. Moreover, the task assignment (how tasks are distributed to users) are also computed beforehand. A well-known strategy is *Find-Fix-Verify* which is effective in countering low-quality workers. The Find-Fix-Verify strategy splits a complex problem into sequential tasks and review phases to utilize user voting and mutual agreement to obtain reliable results [12]. More precisely, in the Find phase, a set of users will identify which needs to be done. Then in the Fix phase, a new group of users is asked to solve the identified issues. Last, in the Verify phase, other independently group of workers performs verification on the submitted work. Another strategy is *Solve-Decompose-Combine* [67], which allows the task owners to monitor the workflow and intervene if necessary. The Solve step asks a user to solve a task. The Decompose step asks another user to decompose a task into various sub-tasks. The Combine step asks another user to merge the outputs from subtasks. Particularly, some Verify

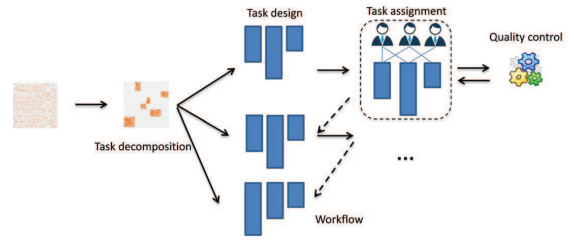


Figure 2: Generic crowdsourcing workflow

controls are included to improve the result quality at each step.

- *Dynamic Workflow*: The sequence of task execution and which tasks are assigned to which users are not decided in advance. Two strategies are often studied: semi-automatic and automatic. In the semi-automatic strategy, tasks are designed with the help of workers. For example in [122], a traveling plan is designed using the workers on Amazon Mechanical Turk. The requesters define several constraints for their plan. First, workers will pose ideas that are essentially tasks for other workers to complete if they believe the idea is reasonable. The system also help by identifying violations: if the traveling plan does not satisfy a constraint, a new task will be asked to resolve this violation. In the automatic strategy, tasks are designed automatically based on the user answers received thus far. For example in [21], the choice of the next question to be asked and the number of workers needed, is reached automatically based on decision theory.

4.3 Task Posting

The goal of task posting strategies is to determine and (dynamically) vary the assignment of tasks to human participants, so that the computational resources (e.g., total monetary cost expended and the total time used for completing the tasks) are minimized. There are two types of posting strategies in the literature: *cost-driven posting strategies* and *time-driven posting strategies*.

Cost-driven posting strategies. The goal of these strategies is to minimize the expected total expenditure that must be paid to achieve a target overall accuracy. This cost minimization problem can be also stated in another formulation: given a limited budget of user inputs, we aim to maximize the overall accuracy of the output results. To solve one or the other problem formulation, research studies have proposed the following representative strategies:

- *Graph-based*: A possible approach for cost-driven task posting is designing an offline scheme that computes all the task assignments before any participant arrives. In [58], the authors implement task allocation via a bipartite graph with one type of nodes corresponding to each of the tasks and another set of nodes corresponding to each of the workers. An edge (i, j) indicates that task i is performed by j , weighted by the corresponding cost. With such a graph, they propose a probabilistic model to compute the probability of output error. On

top of the probabilistic model, they use the maximal likelihood estimation to find the best task assignment given a predefined error threshold.

- *Filtering*: The number of tasks to be given to a user can be reduced by using feature-based filters that specify some features of the objects must be true regardless of user input. For example in image data [74], two profile images should not be matched unless they have the same gender, hair color, and skin color. This predicate allow us to filter all pairs of images that cannot be matched, thus avoiding redundant input from user.
- *Decision theory*: The monetary cost for human participation can be reduced by asking the questions incrementally. In many cases, we cannot ask users to answer all necessary questions due to limited budget. Moreover, the answers to different questions might lead to different output quality. As such, the decision on which questions are to be asked is important, for making the most out of user input. In [53], the authors quantify the potential benefit of a question, whose object is a candidate match, by the changes this candidate match makes to the utility of the dataspace. Another implementation is [82], in which the authors measure the potential benefit of a question, whose object is a correspondence, by the reduction this correspondence makes to the uncertainty of a matching network. Given the received answers thus far, these systems always continue to ask users the most beneficial question.
- *Dependency*: In general, the tasks might depend on each other; i.e., the answer of a task affects the answer of another task. We can leverage the task dependency to reduce the cost by deriving the answers of some tasks without asking users. In [112], the authors apply transitive relations, using which some pairs' labels can be deduced without asking humans to label them, thus reducing the number of crowdsourced pairs. For example, if o_1 and o_2 are matching, and o_2 and o_3 are matching, we do not need to crowdsource the label for (o_1, o_3) since they can be deduced as matching based on transitive relations. In [49], the authors harness network-level integrity constraints to reduce the error rate of aggregating human inputs. The idea is that the correctness of one human input can be used to justify the others via constraint evidence. In the end, the number of human inputs is reduced given a predefined error threshold.
- *Active learning*: Having a similar idea to the Dependency method, this method tries to predict the answers of tasks without user input. In [118], the authors aim to build a classification model for automatically repairing imprecise data. Since the data are huge, we can only obtain a small set of user inputs, which then are used for training the classification model. However, different training inputs might lead to different model accuracy. The goal then becomes how to obtain the best inputs from users. To achieve this goal, the authors design an active learning process, in which the tasks with highest learning benefit are chosen to be

given to users. The learning benefit of a task is measured by the accuracy of possible classification models that could be learned by the user inputs received so far.

Time-driven posting strategies. The goal of these strategies is to minimize the expected total time to complete all assigned tasks, given that the monetary budget for tasks is fixed upfront. Another formulation of the time-driven goal is, given a quality threshold for resulting outputs, we aim to minimize the completion time of the tasks necessary to achieve the quality threshold. A wide range of strategies have been proposed in the literature, including the following most representative ones:

- *Grouping*: The interdependence between tasks can be used to reduce the time for completing them. For example, the work in [7] designs a task allocator that tries to allocate many similar tasks to the same user. Putting multiple related questions in the same task allows human users to complete them faster since these related questions provide “context” for each other (i.e. the information in one question can be used to answer another question).
- *Game theoretical approach*: In practice, human users might be lazy to complete the tasks. In [30], the authors propose a game-theoretical model for pricing the tasks so that the users compete with each other to finish the tasks as fast as possible. The idea is that, for those who finish first, they will get more money and move to the other tasks earlier (with the potential to continue earning much more money).
- *Parallel strategy*: Since human workers have latencies, deadline becomes a critical problem in applications that require fast computation such as searching. To improve the completion time, we can post the tasks in parallel, in which multiple tasks are answered and/or each task are performed by multiple workers simultaneously. In [119], the authors propose a delay prediction model to determine the overall completion time and the overall accuracy of user inputs. Since parallel posting is potential wasteful (the cost for posted tasks is multiply), this prediction model helps to decide the minimal level of parallel (how many questions/workers should be performed/employed at the same time) given an accuracy acceptance threshold. It is worth noting that the parallel strategy might be not applicable for the cases where the output of a task is the input of another task (as in workflows).
- *Humans as real-time processing units*: Some researches make use of human computation in real-time applications, in which each human is considered as a human processing unit (HPU) that has latency as the time to complete a job [22]. This setting draws inspiration from traditional performance optimization on computers with CPUs. In general, there are two types of latency that need to be optimized: (i) *waiting time* – the duration from when a job is published in a human computation market to when it is accepted by a human worker, and (ii) *completion time* – the time for

a human worker to complete a job. The latter can be controlled by monetary reward and difficulty level of tasks as shown in survival analysis models [30, 113]. Whereas, optimizing the former is often more difficult due to the stochastic behavior of human workers [11] (i.e. a human worker might choose not to accept a task for an unknown reason). To tackle this issue, Bernstein et. al. [11] proposes a retainer model, whose idea is to employ a set of prepaid workers, so that they can wait online and perform a task immediately when it is published. They also consider rewarding additionally incentives for workers who accept the tasks more quickly. However, a major limitation of the retainer model is that the quality might be even reduced since the workers could be careless or stressed to handle the tasks simultaneously (a prepaid task might come in the middle of work of another normal task).

4.4 Working Examples

We now present working examples of task-related techniques and methodologies that are applied in existing human computation frameworks, in practical settings.

Human annotation for videos. As a first example, the authors of [63] designed a system that enables crowd workers to extract step-by-step structure from an existing video, in which each step is a meaningful segment that provides textual and visual annotations of the video content. A fixed human computation workflow is used, consisting of three stages: Find, Verify, and Expand. In the Find stage, the human computation task is to mark possible steps for a video segment. A worker input is a collection of timestamps and textual annotations for each step. At the end of the Find step, an aggregation operation is performed: timestamps from different workers are combined via a clustering algorithm (e.g. DBSCAN) to produce the steps (final timestamps are the mean timestamps of the produced clusters). In the Verify stage, the human computation task is to vote on the best annotations for each step, which are output from the Find step. The authors used majority voting to make the final decision. In case of tie-breaks, the longest description is picked. In the Expand stage, the human computation task is to select a thumbnail that best summarizes each step, in which a multiple-choice question with static thumbnails of the video is presented to the workers. To aggregate the worker choices of thumbnails, the authors use majority voting.

Human categorization for text data. As a another example, the authors of [5] designed a framework that employs crowd workers to extract categories from text data, for the purpose of text clustering. The studied dataset is on the contribution types of top Wikipedia contributors. A fixed human computation workflow is used with two steps: Re-Representation and Iterative Clustering. In the first step, the human computation task is to provide a label of contribution type for an individual contributor or a group of contributors. The worker inputs are fed to the next step, in which an iterative clustering process is performed for clustering similar contributors. In the second step, the human computation task is to group similar contributors. This is similar to the clustering aggregation problem yet the difference is that the workers subsequently refine the clustering results of each other without using any automatic clustering

technique.

Human-power search engine for querying images. In [119], the authors designed a mobile application that allows for searching images via an query image, in which human workers are employed to validate whether a candidate image is similar to the query image or not. The human computation task consists of an image pair between the query image and a candidate image retrieved by automatic tools (Figure 3). Two possible answers are given: YES – the two images are similar, and NO – other. No further context information is provided since images are quite illustrative data for humans. After all worker inputs are collected, they are aggregated by majority rule to determine the best image as the retrieved result (Figure 5).

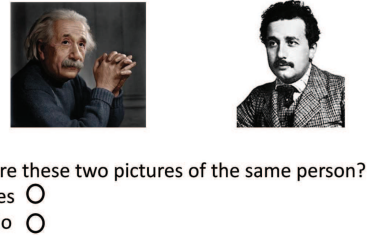


Figure 3: Image Search Task

Human validation for automatic matching results [51]. In [51], the authors designed a system that allows the crowd workers to validate the results of automatic schema matching tools. The human computation task is to validate the correctness of an attribute correspondence generated by automatic tools (Figure 4). Different types of context information are provided based on the relationships of the given correspondences with the others in a large network of schemas. Two possible answers are allowed: YES – the correspondence is approved, and NO – otherwise. For each correspondence, the worker inputs are aggregated via the EM-algorithm (see Section 3.1) to compute the final validation for the correspondence (along with an error rate value). A dynamic workflow is used for the crowdsourcing process, in which worker inputs are elicited until the error rate of aggregated results is less than a pre-defined threshold. More precisely, for each correspondence, if the worker inputs agree on its approval or disapproval, the system stops asking more questions to the crowd. On top of this workflow, a cost-driven task posting strategy is used, in which the number of worker inputs is minimized given an error threshold of aggregated results. To this end, the authors leverage the relationships between correspondences to justify their validation against each other, thus requiring less worker inputs to achieve the same error rate.

5. APPLICATIONS

In this section, we present well-known applications that make use of human computation, spanning various domains: (i) Data acquisition – measure real-world physical conditions and collect the measured data, (ii) Data analysis – inspect data characteristics to discover useful information and patterns, (iii) Data curation – maintain and extract important information for reuse and preservation, (iv) Data storage – store data in readable formats efficiently, and (v) Data usage

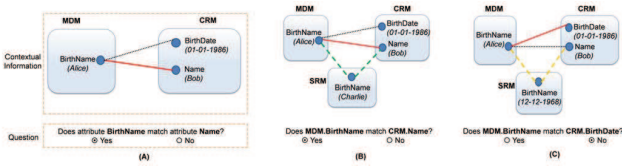


Figure 4: Schema Matching Task with 3 different contextual information: (A) All alternative targets, (B) Transitive closure, (C) Transitive violation

– support exploring and visualizing data for decision-support purposes.

5.1 Data Acquisition

Human-assisted Machine Learning. In many machine learning applications (e.g., classification, visual recognition, and object detection), training data is needed to parameterize the automatic models. However, in traditional systems, training data is often limited (e.g., only a single expert is hired) and out of date (e.g., old data is used several times for modern techniques). To address this problem, a large body of works [14, 99, 108] employs human computation in the form of crowdsourcing to iteratively improve the training data. All of these works are feasible due to the mass availability of thousands of crowd workers and their cheap hiring cost in online platforms (e.g., as stated earlier, Amazon Mechanical Turk, CrowdFlower).

In general, these works design an active learning process, which is executed as follows. Firstly, automatic results are produced based on existing training data. Then, crowd workers are employed to validate these results. Combining worker inputs, the system generates new training data. On top of the new training data, another iteration is performed using the same automatic techniques. In brief, this active learning process entails a human computation loop, in which human users iteratively contribute to improve the output quality of automatic tools.

Participatory Sensing and Measuring. In participatory sensing, people are equipped with built-in sensors in their smartphones to measure real world physical conditions. There are various applications in this category such as environmental monitoring and tracking daily activities.

Traditional environmental monitoring systems rely on aggregate statistics by fixed sensors to measure and report environmental pollution over an area (e.g., community, city, state). These systems have several limitations. For example, since the cost of sensor deployment is high, the sensors might not cover all the desired regions. Moreover, due to various factors including low battery or damaged components, the deployed sensors might report imprecise measurement. To overcome these limitations, participatory sensing systems (e.g. Common Sense [27]) use specialized handheld air quality sensing devices, which are distributed to a large number of human participants, for collectively measuring various air quality indices such as carbon emissions, noise levels, and water conditions.

Similar methods are demonstrated in PEIR – another participatory sensing system [78], which uses mobile phones to infer daily activities of human participants. For example, the combination of personalized accelerometer data and a

sequence of locations from the GPS can recognize the transportation mode of a user, such as biking, driving, or taking public transports (bus, metro).

5.2 Data Analysis

Entity Extraction. Entity linking is the problem of matching URIs to entities in the Web context. In [23], the authors design the human computation tasks to be published on a crowdsourcing platform using three elements: i) the name of the textual entity, ii) contextual information generated by the original HTML document from which the entity was extracted, and iii) the current top- k matches for the entity provided by the automatic tools. They experimented with various types of contextual information such as the text corresponding to the extracted entity and the text snippets around each occurrence of the entity in the HTML page. The question of a matching task is asking crowdsourcing workers to select one or many among the candidate URIs for the given entity. The answers given by workers can be aggregated into final matching results or used as training data for machine-learning algorithms.

Linked Data. Automatic relationship extraction in the literature has limitations in terms of accuracy while relationship extraction using experts incurs a very high cost. To circumvent this issue, the work in [18] shows how to create a taxonomy using crowd workers. The human workers are presented with three crowdsourcing tasks: generate categories from items, select the best category for an item from a list of categories and categorize the item into appropriate categories. The categories and the items are then collected to generate a full taxonomy using an automatic structure inference. Another approach in relationship extraction is provided in [57] where the workers are asked to specify the relationships between simple tags. These basic relationships are then aggregated incrementally as new tag relations arrive to generate a complete structure.

Sentiment Analysis. While sentiment analysis is a very challenging task for computers, it is a trivial task for humans as it involves human natural language processing. The feasibility of using human for sentiment analysis is shown in [72] where the authors proposed a method to analyze sentiments in Twitter tweets regarding a keyword. After gathering enough tweets related to the keywords, a crowdsourcing task is created for each tweet according to a pre-defined template. Crowd workers are recruited to provide their feelings about the tweets such as Good, Satisfied, Not Satisfied. The answers are then aggregated based on a probabilistic model to get an estimated correct answer.

5.3 Data Curation

Credibility and Trust. Information credibility has been studied in the literature to evaluate the trustworthiness of a data source or an artifact, indicating whether the information is trustable or not. With the growth of the web, information credibility is applied to assess the credibility of websites. However, assessing the credibility of information on the web is still a challenging issue. As a publicly available platform in which anyone can share anything, the web is inherently uncertain, in which information published cannot be easily verified for validity, legitimacy and trustworthiness. Moreover, as the web contents are shared by humans,

automatic techniques might not be able to truly assess the credibility of web content.

Overcoming this issue, a large body of work employs human computation to evaluate the credibility of websites. In general, these works collect users’ feedback by allowing them to provide ratings on a web page. Possible rating scores can be binary (e.g., Positive/Negative [39]) or multiple (Trustfulness/Unbiased/Security/Page design [46]). The ratings are then combined to produce the credibility score of the web page (e.g., by computing the means and standard variances of user rating scores [39]).

Annotation. Annotation is the process of attaching meta-data (e.g., comments, tags, markups) to different types of data such as images, text, media, etc. Since data has different characteristics and formats, automatic annotation tools might not be able to produce meaningful annotations that satisfy user needs. Moreover, humans often understand the content of data more easily than computers (e.g., watching videos). As such, many research works employ human computation for the purposes of data annotation. As an example, the authors of [63] designed a system that enables human participants to annotate step-by-step structure for an existing video, in which each step is a meaningful segment with textual and visual annotations of the video content. Human annotations are then combined by majority voting to decide the best annotations for the videos.

Entity Resolution. Entity Resolution (also known as entity reconciliation, duplicate detection, record linkage) is the task of finding different records that refer to the same entity in database systems. In [111], the authors generated two types of human computation tasks for entity resolution. The first type of task is asking the humans to verify, for each pair of records, whether they refer to the same entity or not. The possible outputs of pair-based tasks, for example, are “These items are similar” or “These items are different”. The second type of task is asking the humans to find all duplicate records among a small group of individual records. User interface of group-based tasks often contains a list of records, in which to indicate duplicate records, users assign them to the same label from a drop-down list at the front of each record. In both types of tasks, the context information includes a brief description of the task and more detailed instructions are displayed below. In general, the first type of tasks takes more monetary cost to employ users since all pair of records need to be verified, whereas the second type of task increases the difficulty level of the question since users have to consider multiple records at the same time.

5.4 Data Storage

Schema Matching. Schema matching is the process of identifying attribute correspondences between database schemas. Since automatic matching tools rely on heuristics, their results are inherently uncertain. As such, a post-matching human effort is needed to validate the matching results. There are various works that employ human computation for schema matching. Typical examples are [51, 75], in which the system employs human users via crowdsourcing platforms (e.g., Amazon Mechanical Turk) to validate the correctness of generated attribute correspondences. Crowdsourcing fits in this application since validating one correspondence is a small task that requires very little cost and

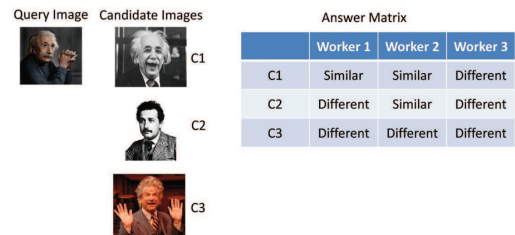
effort. Figure 4 depicts a typical crowdsourcing task, in which a worker is asked to validate an attribute correspondence in schema matching. Different types of context information are provided such as top-*k* candidate matches and constraint-related evidence [51].

Ontology Alignment. Ontology alignment is the process of determining correspondences between concepts. A set of correspondences is also called an alignment. Similar to schema matching, human computation is also employed to validate the correctness of automatically generated correspondences, which are inherently uncertain. A well-known work is [93], in which the authors employ crowd workers via crowdsourcing platforms (e.g. CrowdFlower) to work on a set of candidate mappings between two ontologies for improving their accuracy. Two types of human computation tasks are proposed: validation tasks and identification tasks. A validation task presents workers with two concepts and the relationship that connects them and ask the workers whether or not they agree with the relationship. An identification task (or creation task) asks workers to identify/create the relationship between two given concepts. Further context information is also provided such as the data instance of the two concepts.

Sorting/Ordering/Joining. In practice, there are always sorting tasks which cannot be computed automatically such as “order these variants of a sentence by quality”, or “order the images of animals by adult size”. The reason is that the order of the sorting elements is subjective to human evaluation. In [74], the authors studied this problem by designing human computation tasks for sorting such datasets. The human participants are asked to compare pairs of items against each other or assign a rating to each item. Based on human inputs, the authors that an actual algorithm that sort items using pairwise comparisons or their ratings can be obtained.

5.5 Data Usage

Query Answering. Automatic image search techniques in the literature have limitations in terms of accuracy. Going beyond this issue, the work in [119] employs humans due to their natural ability for comparing images. In that, a human computation task is validating the automatic search results. More precisely, for a query image, their system generates a set of candidate search results. Each pair of <query image, candidate image> is then validated by crowd workers. Possible outputs are YES (if the worker thinks two images are similar) and NO (if the worker thinks two images are not similar). Figure 5 shows the user interface where the image search engine returns three candidate images, and each candidate image is validated by independent workers.



	Answer Matrix		
	Worker 1	Worker 2	Worker 3
C1	Similar	Similar	Different
C2	Different	Similar	Different
C3	Different	Different	Different

Figure 5: Image Search Aggregation

User Relevance Feedback. Relevance evaluation is an important requirement in information retrieval systems. For example, it is a necessity when developing a new search engine or comparing different search engines with each other. The challenge is that the search results of automatic techniques are inherently uncertain, since there does not exist an automatic algorithm to perfectly evaluate the relevance of search results to user (otherwise that algorithm itself entails a perfect search method for the given retrieval tasks). To overcome this challenge in the context of document retrieval, early research in relevance evaluation employs a small set of examiners (e.g. editors, volunteers, students) to determine the relevance of every document in a corpus to a set of test queries, thus creating a test collection. Newly developed retrieval techniques are then evaluated by comparing their top- k documents with those in the test collection. Using this human computation method, various test collections have been created and refined annually such as TREC [19], NTCIR [55], and CLEF [15].

However, using test collections has two major limitations. First, it is assumed that the test collections should cover important characteristics of the document corpus. This might not hold when the corpus is at large scale and changes frequently. Second, the domains of retrieval tasks are limited to only those studied by the test collections. For a new search domain, it takes a lot of time and cost to employ the examiners for creating the test collections. To tackle this issue, Alonso et. al. [3] employs another scale of human computation by using crowdsourcing services to obtain user relevance feedback as a result of micro-tasks. Due to the mass number and high availability of crowdsourcing workers, the search results are quickly evaluated to measure the effectiveness of a given retrieval technique regardless of the corpus and its domain.

Recommender Systems. As aforementioned, implicit human computation systems generally exploit the user traces in an underlying platform to solve computational problems such as spelling correction and optical character recognition. This human computation method is further employed to help the computational systems serve users better as in collaborative filtering and adaptive web-sites. On the one hand, collaborative filtering applications [94] exploit historical user purchases to recommend products that truly match user expectations. More precisely, existing collaborative filtering techniques leverage existing users' ratings based on similarity between users and/or products in order to select those that have highest ratings for recommendation. The core idea is that products recommended by a particular user will be preferred by those other users who are similar to her. On the other hand, there is a large body of work on developing adaptive web sites [86] by exploiting the click logs of users to improve the presentation of a Web page. For example, based on the sequence of clicked URLs, the web designer can design a better navigation between the URLs [77].

Visualization. Human computation is also applied to improve the visualization of data in terms of graphical perception. Since there is no universal definition of visualization effectiveness (e.g. "beauty" is a perceptual concept that can vary from culture to culture, from person to person, and even throughout history), a straightforward approach is to employ a sample set of users to evaluate a given visualization design. To make the sample evaluation significant with

low cost, Heer et. al. [43] leverage crowdsourcing services (e.g. AMT) to perform the evaluation. Their experimental findings show that using crowdsourced results not only provide the design guidelines similar to those of the experts in the field but also gain new understandings in designing visual elements.

6. CONCLUSION

We have discussed the state-of-the-art of human computation in the context of data, information, and knowledge management (DIKM). Our purpose in developing this classification of human computation systems, models, and applications is to understand the entire life-cycle of human computation systems in an ordered way [34, 83].

While the focus of our classification lies on the function characteristics of DIKM problems, we offer a practical guideline of techniques and tools to deal with different aspects of human computation such as modeling of problems, reconciliation of human input, and task design and dissemination. Specifically, we discuss the applicability and limitations of such techniques to help both researchers and developers. While researchers can use our survey to position their research directions and identify unsolved problems, developers can leverage the surveyed techniques for their potential applications [47–50, 79–81].

References

- [1] Daniel Duarte Abdala, Pakaket Wattuya, and Xiaoyi Jiang. "Ensemble clustering via random walker consensus strategy". In: *ICPR*. 2010, pp. 1433–1436.
- [2] Nir Ailon, Moses Charikar, and Alantha Newman. "Aggregating inconsistent information: ranking and clustering". In: *JACM* (2008), p. 23.
- [3] Omar Alonso, Daniel E. Rose, and Benjamin Stewart. "Crowdsourcing for Relevance Evaluation". In: *SIGIR Forum* (2008), pp. 9–15.
- [4] Yael Amsterdamer et al. "Crowd Mining". In: *SIGMOD*. 2013, pp. 241–252.
- [5] Paul André, Aniket Kittur, and Steven P Dow. "Crowd synthesis: Extracting categories and clusters from complex data". In: *CSCW*. 2014, pp. 989–998.
- [6] Judd Antin and Aaron Shaw. "Social desirability bias and self-reports of motivation: a study of amazon mechanical turk in the US and India". In: *CHI*. 2012, pp. 2925–2934.
- [7] Amos Azaria, Yonatan Aumann, and Sarit Kraus. "Automated agents for reward determination for human work in crowdsourcing applications". In: *AA-MAS*. 2013, pp. 1–22.
- [8] Eugene Barsky and Michelle Purdon. "Introducing Web 2.0: social networking and social bookmarking for health librarians". In: *JCHLA* (2006), pp. 65–67.
- [9] Benjamin B Bederson and Alexander J Quinn. "Web workers unite! addressing challenges of online laborers". In: *CHI*. 2011, pp. 97–106.
- [10] Asa Ben-Hur, Andre Elisseeff, and Isabelle Guyon. "A stability based method for discovering structure in clustered data". In: *PSB*. 2001, pp. 6–17.

- [11] Michael S. Bernstein et al. “Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces”. In: *UIST*. 2011, pp. 33–42.
- [12] Michael S Bernstein et al. “Soylent: a word processor with a crowd inside”. In: *UIST*. 2010, pp. 313–322.
- [13] Ralph Allan Bradley and Milton E Terry. “Rank analysis of incomplete block designs: I. The method of paired comparisons”. In: *Biometrika* (1952), pp. 324–345.
- [14] Steve Branson et al. “Visual recognition with humans in the loop”. In: *ECCV*. 2010, pp. 438–451.
- [15] Martin Braschler. “CLEF 2001 – Overview of results”. In: *Evaluation of Cross-Language Information Retrieval Systems*. 2002, pp. 9–26.
- [16] Jeffrey A Burke et al. “Participatory sensing”. In: *Center for Embedded Network Sensing* (2006).
- [17] Caleb Chen Cao et al. “Whom to Ask?: Jury Selection for Decision Making Tasks on Micro-blog Services”. In: *VLDB*. 2012, pp. 1495–1506.
- [18] Lydia B Chilton et al. “Cascade: Crowdsourcing taxonomy creation”. In: *CHI*. 2013, pp. 1999–2008.
- [19] Charles L Clarke, Nick Craswell, and Ian Soboroff. *Overview of the trec 2009 web track*. Tech. rep. DTIC Document, 2009.
- [20] Chenyun Dai et al. “An approach to evaluate data trustworthiness based on data provenance”. In: *SDM*. 2008, pp. 82–98.
- [21] Peng Dai, Daniel Sabey Weld, et al. “Decision-theoretic control of crowd-sourced workflows”. In: *AAAI*. 2010.
- [22] James Davis et al. “The hpu”. In: *CVPRW*. 2010, pp. 9–16.
- [23] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. “ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking”. In: *WWW*. 2012, pp. 469–478.
- [24] Persi Diaconis. “Group representations in probability and statistics”. In: *Lecture Notes-Monograph Series* (1988), pp. i–192.
- [25] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. “Crowdsourcing systems on the world-wide web”. In: *CACM* (2011), pp. 86–96.
- [26] Carlotta Domeniconi et al. “Locally adaptive metrics for clustering high dimensional data”. In: *DMKD* (2007), pp. 63–97.
- [27] Prabal Dutta et al. “Common sense: participatory urban sensing using a network of handheld air quality monitors”. In: *SensSys*. 2009, pp. 349–350.
- [28] Cynthia Dwork et al. “Rank aggregation methods for the web”. In: *WWW*. 2001, pp. 613–622.
- [29] Carsten Eickhoff and Arjen P. de Vries. “How Crowd-sourcable is your Task?” In: *CSDM*. 2011, pp. 11–14.
- [30] Siamak Faradani, Björn Hartmann, and Panagiotis G Ipeirotis. “What’s the Right Price? Pricing Tasks for Finishing on Time.” In: *Human Computation* (2011).
- [31] Xiaoli Zhang Fern and Carla E Brodley. “Solving cluster ensemble problems by bipartite graph partitioning”. In: *ICML*. 2004, p. 36.
- [32] Michael Fire et al. “Data mining opportunities in geosocial networks for improving road safety”. In: *IEEEI*. 2012, pp. 1–4.
- [33] Michael J. Franklin et al. “CrowdDB: Answering Queries with Crowdsourcing”. In: *SIGMOD*. 2011, pp. 61–72.
- [34] Avigdor Gal et al. “Completeness and ambiguity of schema cover”. In: *CoopIS*. 2013, pp. 241–258.
- [35] Florent Garcin et al. “Rating aggregation in collaborative filtering systems”. In: *RecSys*. 2009, pp. 349–352.
- [36] Alan Genz. “Numerical computation of rectangular bivariate and trivariate normal and t probabilities”. In: *Statistics and Computing* (2004), pp. 251–260.
- [37] Elizabeth M. Gerber and Julie Hui. “Crowdfunding: Motivations and Deterrents for Participation”. In: *TOCHI* (2013), 34:1–34:32.
- [38] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. “Clustering aggregation”. In: *TKDD* (2007), p. 4.
- [39] Katherine Del Giudice. “Crowdsourcing credibility: The impact of audience feedback on Web page credibility”. In: *ASIST* (2010), pp. 1–9.
- [40] Peter A Gloor et al. “Web science 2.0: Identifying trends through semantic social network analysis”. In: *CSE*. 2009, pp. 215–222.
- [41] Andrey Goder and Vladimir Filkov. “Consensus Clustering Algorithms: Comparison and Refinement.” In: *ALENEX*. 2008, pp. 109–117.
- [42] Ryan G Gomes et al. “Crowdclustering”. In: *NIPS*. 2011, pp. 558–566.
- [43] Jeffrey Heer and Michael Bostock. “Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design”. In: *CHI*. 2010, pp. 203–212.
- [44] Kenneth Higbee. “Mathematical Classification and Clustering”. In: *Technometrics* (1998), pp. 80–80.
- [45] David W. Hosmer and Stanley Lemeshow. *Applied logistic regression*. Wiley-Interscience Publication, 2000.
- [46] Zhicong Huang, Alexandra Olteanu, and Karl Aberer. “CredibleWeb: a platform for web credibility evaluation”. In: *CHI*. 2013, pp. 1887–1892.
- [47] Nguyen Quoc Viet Hung et al. “ERICA: Expert guidance in validating crowd answers”. In: *SIGIR*. 2015, pp. 1037–1038.
- [48] Nguyen Quoc Viet Hung et al. “Minimizing efforts in validating crowd answers”. In: *SIGMOD*. 2015, pp. 999–1014.
- [49] Nguyen Quoc Viet Hung et al. “On leveraging crowdsourcing techniques for schema matching networks”. In: *DASFAA*. 2013, pp. 139–154.
- [50] Nguyen Quoc Viet Hung et al. “SMART: A tool for analyzing and reconciling schema matching networks”. In: *ICDE*. 2015, pp. 1488–1491.
- [51] NguyenQuocViet Hung et al. “On Leveraging Crowdsourcing Techniques for Schema Matching Networks”. In: *DASFAA*. 2013, pp. 139–154.

- [52] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. “Quality management on Amazon Mechanical Turk”. In: *HCOMP*. 2010, pp. 64–67.
- [53] Shawn R Jeffery, Michael J Franklin, and Alon Y Halevy. “Pay-as-you-go user feedback for dataspace systems”. In: *SIGMOD*. 2008, pp. 847–860.
- [54] Audun Jsang and Roslan Ismail. “The beta reputation system”. In: *BECC*. 2002, pp. 41–55.
- [55] Noriko Kando et al. “Overview of IR tasks at the first NTCIR workshop”. In: *NTCIR*. 1999, pp. 11–44.
- [56] Bob Kanefsky, Nadine G Barlow, and Virginia C Gulick. “Can distributed volunteers accomplish massive data analysis tasks”. In: *Lunar and Planetary Science* (2001).
- [57] Dimitris Karampinas and Peter Triantafillou. “Crowdsourcing taxonomies”. In: *ESWC*. 2012, pp. 545–559.
- [58] David R Karger, Sewoong Oh, and Devavrat Shah. “Budget-optimal task allocation for reliable crowdsourcing systems”. In: *Operations Research* (2014), pp. 1–24.
- [59] DR Karger, S Oh, and D Shah. “Iterative learning for reliable crowdsourcing systems”. In: *NIPS*. 2011, pp. 1953–1961.
- [60] George Karypis and Vipin Kumar. “A fast and high quality multilevel scheme for partitioning irregular graphs”. In: *SIAM Journal on scientific Computing* (1998), pp. 359–392.
- [61] George Karypis et al. “Multilevel hypergraph partitioning: applications in VLSI domain”. In: *VLSI* (1999), pp. 69–79.
- [62] FK Khattak and A Salleb-Aouissi. “Quality Control of Crowd Labeling through Expert Evaluation”. In: *NIPS-CSS*. 2011.
- [63] Juho Kim et al. “Crowdsourcing Step-by-step Information Extraction to Enhance Existing How-to Videos”. In: *CHI*. 2014, pp. 4017–4026.
- [64] Aniket Kittur, Ed H. Chi, and Bongwon Suh. “Crowdsourcing user studies with Mechanical Turk”. In: *CHI*. 2008, pp. 453–456.
- [65] Aniket Kittur et al. “Crowdforge: Crowdsourcing complex work”. In: *UIST*. 2011, pp. 43–52.
- [66] Aniket Kittur et al. “The Future of Crowd Work”. In: *CSCW*. 2013, pp. 1301–1318.
- [67] Anand Kulkarni, Matthew Can, and Björn Hartmann. “Collaboratively Crowdsourcing Workflows with Turkomatic”. In: *CSCW*. 2012, pp. 1003–1012.
- [68] LI Kuncheva, CJ Whitaker, and CA Shipp. “Limits on the majority vote accuracy in classifier fusion”. In: *Pattern Anal. Appl.* (2003), pp. 22–31.
- [69] Edith Law and Luis von Ahn. “Human computation”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2011), pp. 1–121.
- [70] Joon Ho Lee. “Analyses of multiple evidence combination”. In: *SIGIR*. 1997, pp. 267–276.
- [71] Kyumin Lee, James Caverlee, and Steve Webb. “The social honeypot project: protecting online communities from spammers”. In: *WWW*. 2010, pp. 1139–1140.
- [72] Xuan Liu et al. “Cdas: a crowdsourcing data analytics system”. In: *VLDB*. 2012, pp. 1040–1051.
- [73] Huilan Luo, Furong Jing, and Xiaobing Xie. “Combining multiple clusterings using information theory based genetic algorithm”. In: *CIS*. 2006, pp. 84–89.
- [74] Adam Marcus et al. “Human-powered sorts and joins”. In: *VLDB*. 2011, pp. 13–24.
- [75] R. McCann, Warren Shen, and AnHai Doan. “Matching Schemas in Online Communities: A Web 2.0 Approach”. In: *ICDE*. 2008, pp. 110–119.
- [76] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.
- [77] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. “Creating adaptive web sites through usage-based clustering of URLs”. In: *KDEX*. 1999, pp. 19–25.
- [78] Min Mun et al. “PEIR, the personal environmental impact report, as a platform for participatory sensing systems research”. In: *MobiSys*. 2009, pp. 55–68.
- [79] Hung Quoc Viet Nguyen et al. “Minimizing human effort in reconciling match networks”. In: *ER*. 2013, pp. 212–226.
- [80] Quoc Viet Hung Nguyen et al. “An evaluation of aggregation techniques in crowdsourcing”. In: *WISE*. 2013, pp. 1–15.
- [81] Quoc Viet Hung Nguyen et al. “BATC: a benchmark for aggregation techniques in crowdsourcing”. In: *SIGIR*. 2013, pp. 1079–1080.
- [82] Quoc Viet Hung Nguyen et al. “Pay-as-you-go reconciliation in schema matching networks”. In: *ICDE*. 2014, pp. 220–231.
- [83] Thanh Tam Nguyen et al. “Result selection and summarization for Web Table search”. In: *ICDE*. 2015, pp. 231–242.
- [84] Andrei Oghina et al. “Predicting imdb movie ratings using social media”. In: *ECIR*. 2012, pp. 503–507.
- [85] Ory Okolloh. “Ushahidi, or ‘testimony’: Web 2.0 tools for crowdsourcing crisis information”. In: *Participatory learning and action* (2009), pp. 65–70.
- [86] Mike Perkowitz and Oren Etzioni. “Adaptive web sites”. In: *CACM* (2000), pp. 152–158.
- [87] Kunal Punera and Joydeep Ghosh. “Consensus-based ensembles of soft clusterings”. In: *Applied Artificial Intelligence* (2008), pp. 780–810.
- [88] Alexander J. Quinn and Benjamin B. Bederson. “Human Computation: A Survey and Taxonomy of a Growing Field”. In: *CHI*. 2011, pp. 1403–1412.
- [89] Vikas C. Raykar et al. “Supervised Learning from Multiple Experts: Whom to Trust when Everyone Lies a Bit”. In: *ICML*. 2009, pp. 889–896.
- [90] Steven Reiss. “Multifaceted nature of intrinsic motivation: The theory of 16 basic desires.” In: *Review of General Psychology* (2004), p. 179.
- [91] M Elena Renda and Umberto Straccia. “Web metasearch: rank vs. score based rank aggregation methods”. In: *SAC*. 2003, pp. 841–846.

- [92] J. Ross et al. “Who are the crowdworkers?: shifting demographics in mechanical turk”. In: *CHI*. 2010, pp. 2863–2872.
- [93] Cristina Sarasua, Elena Simperl, and Natalya F Noy. “Crowdmap: Crowdsourcing ontology alignment with microtasks”. In: *ISWC*. 2012, pp. 525–541.
- [94] Badrul Sarwar et al. “Item-based collaborative filtering recommendation algorithms”. In: *WWW*. 2001, pp. 285–295.
- [95] Mudhakar Srivatsa, Li Xiong, and Ling Liu. “Trust-Guard: countering vulnerabilities in reputation management for decentralized overlay networks”. In: *WWW*. 2005, pp. 422–431.
- [96] Matthias Stevens and Ellie DâĂŽHondt. “Crowdsourcing of pollution data using smartphones”. In: *Workshop on Ubiquitous Crowdsourcing*. 2010.
- [97] Osamuyimen Stewart, Juan M Huerta, and Melissa Sader. “Designing crowdsourcing community for the enterprise”. In: *KDD*. 2009, pp. 50–53.
- [98] Alexander Strehl and Joydeep Ghosh. “Cluster ensembles—a knowledge reuse framework for combining partitionings”. In: *AAAI*. 2002, pp. 93–99.
- [99] Chong Sun et al. “Chimera: Large-Scale Classification using Machine Learning, Rules, and Crowdsourcing”. In: *VLDB*. 2014.
- [100] James Surowiecki. “The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business”. In: *Economies, Societies and Nations* (2004).
- [101] Leon L Thurstone. “The method of paired comparisons for social values.” In: *JAPSYCH* (1927), p. 384.
- [102] Alexander Topchy, Anil K Jain, and William Punch. “Clustering ensembles: Models of consensus and weak partitions”. In: *TPAMI* (2005), pp. 1866–1881.
- [103] Alexander P Topchy, Anil K Jain, and William F Punch. “A Mixture Model for Clustering Ensembles.” In: *SDM*. 2004, pp. 379–390.
- [104] <http://www.crowdfunder.com/>. In: *ONLINE* (2014).
- [105] Sandro Vega-Pons, Jyrko Correa-Morris, and José Ruiz-Shulcloper. “Weighted partition consensus via kernels”. In: *Pattern Recognition* (2010), pp. 2712–2724.
- [106] Sandro Vega-Pons and José Ruiz-Shulcloper. “A survey of clustering ensemble algorithms”. In: *IJPRAI* (2011), pp. 337–372.
- [107] Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. “Crowdsourcing Spatial Phenomena Using Trust-Based Heteroskedastic Gaussian Processes”. In: *HCOMP*. 2013.
- [108] Sudheendra Vijayanarasimhan and Kristen Grauman. “Large-scale live active learning: Training object detectors with crawled data and crowds”. In: *CVPR* (2014), pp. 97–114.
- [109] Luis Von Ahn and Laura Dabbish. “Labeling images with a computer game”. In: *CHI*. 2004, pp. 319–326.
- [110] Luis Von Ahn et al. “recaptcha: Human-based character recognition via web security measures”. In: *Science* (2008), pp. 1465–1468.
- [111] Jiannan Wang et al. “CrowdER: Crowdsourcing Entity Resolution”. In: *VLDB*. 2012, pp. 1483–1494.
- [112] Jiannan Wang et al. “Leveraging Transitive Relations for Crowdsourced Joins”. In: *SIGMOD*. 2013, pp. 229–240.
- [113] Jing Wang, Siamak Faridani, and P Ipeirotis. “Estimating the completion time of crowdsourced tasks using survival analysis models”. In: *CSDM*. 2011.
- [114] Yao Wang and Julita Vassileva. “Bayesian network-based trust model”. In: *WI*. 2003, pp. 372–378.
- [115] Yao Wang and Julita Vassileva. “Trust and reputation model in peer-to-peer networks”. In: *P2P*. 2003, pp. 150–157.
- [116] Peter Welinder and Pietro Perona. “Online crowdsourcing: rating annotators and obtaining cost-effective labels”. In: *CVPRW*. 2010, pp. 25–32.
- [117] Jacob Whitehill et al. “Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise”. In: *NIPS*. 2009, pp. 2035–2043.
- [118] Mohamed Yakout et al. “Guided data repair”. In: *VLDB*. 2011, pp. 279–289.
- [119] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. “CrowdSearch: Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones”. In: *MobiSys*. 2010, pp. 77–90.
- [120] Hye-Sung Yoon et al. “Heterogeneous clustering ensemble method for combining different cluster results”. In: *BioDM*. 2006, pp. 82–92.
- [121] H Peyton Young and Arthur Levenglick. “A consistent extension of Condorcet’s election principle”. In: *SIAM Journal on Applied Mathematics* (1978), pp. 285–300.
- [122] Haoqi Zhang et al. “Human Computation Tasks with Global Constraints”. In: *CHI*. 2012.