Human-in-the-loop in Machine Learning

Chengliang Chai, Guoliang Li
Tsighua University, Tsighua University
{chaicl15@mails.tsinghua.edu.cn, liguoliang@tsinghua.edu.cn}

Abstract

Humans are playing more and more significant roles in the machine learning pipeline, which consists of data preprocessing, data labeling, model training and inference. Humans not only can provide training data for machine learning applications, but also directly accomplish some tasks that are hard for the computer in the pipeline, with the help of machine-based approaches. In this survey, we summarize overall techniques in human-in-the-loop machine learning, including: (1) Quality Improvement: Humans may return noisy results, so effective techniques need to be applied to achieve high quality; (2) Cost Reduction: Since the humans are not free, we should reduce the monetary cost; (3) Latency Reduction: Compared with the computers, humans can be slow, so latency reduction techniques are required. (4) Active Learning: In most cases, budget for labeling the training data is limited, so active learning selects the most interesting examples to label iteratively. (5)Weak Supervision; Enough perfect labels are expensive to acquire, so weak supervision is proposed to generate high quality results from weak labels. Then we survey how to apply the above techniques to different modules in the machine learning pipeline, introduce some future works and finally conclude.

1 Introduction

Machine learning(ML) is having a profound impact on a wide variety of applications, such as image and speech recognition, natural language processing and heath care. It has made breakthroughs due to large-scale data and sophisticated algorithms, but the power of humans can not be neglected. More specifically, large-scale training data needs humans to create and some ML algorithms require humans to improve the performance iteratively. For example, ImageNet [16] is a representative benchmark that promotes the development of computer vision area. It is constructed through crowdsourcing, which is an effective way to address a wide variety of tasks by utilizing hundreds of thousands of ordinary workers (i.e., humans).

Humans play important roles in the entire ML pipeline from data preparation to result inference, as shown in Figure 1. First, data scientists spend more than 60% of their time in preprocessing the data [1] before building a ML model. In most cases, the original data we can utilize to build a model may be structured or semi-structured, which needs to be extracted as structured data to construct features. Data extraction is to use rules(functions) or machine learning techniques [40, 49, 19] to extract data from non-structured data, where humans can provide rules or training data. Then given the structured data from multiple sources, we always have to integrate them [13, 67, 65] to enrich the records and features. Data integration is used to identify duplicated records or

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

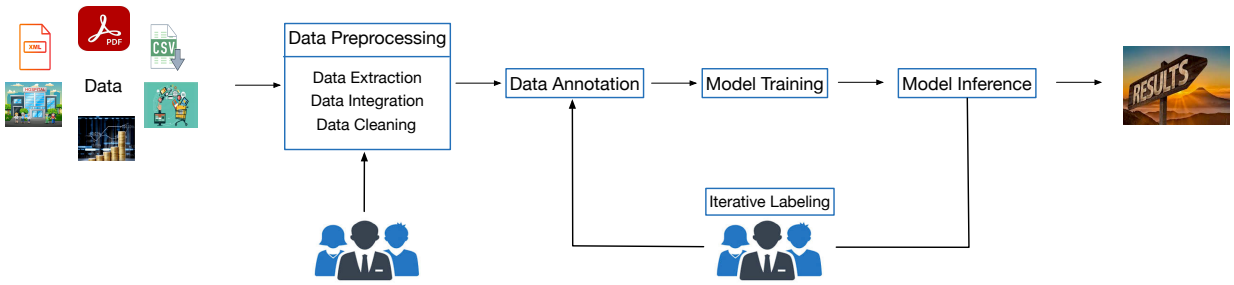

Figure 1: A Human-in-the-loop Machine Learning Pipeline

cells in different columns that refer to the same entity, such as "Apple iphone 8" and "Iphone 8th", and then integrate them. Humans can improve the performance of data integration by providing answers of entity pairs that are hard for the computer. Also, ML techniques can also be used to address the problem, where humans can provide training data. In the real world, data is always dirty because of some missing values, outliers and records that violate integrity constraints, so data cleaning can detect and repair these data, which may improve the ML performance. Second, each record has to be labeled to construct the training sets. Humans can provide labels directly or write rules for generating labels. Then the model is trained and tested on the labeled data. If the performance cannot satisfy the user's requirement, humans will be asked to label another batch of records iteratively until a good performance is achieved.

Although humans contribute to different modules in the ML pipeline, there are several common important problems in human-in-the-loop machine learning.

(1) Quality improvement. Humans are likely to make mistakes no matter what kinds of tasks they do because they may have different levels of expertise, and an untrained human is not qualified to accomplish certain tasks. To achieve high quality labels, we need to tolerate human errors and infer high quality results from noisy answers. Therefore, we should model the humans' quality and tasks' difficulty, assign tasks to appropriate humans and infer final results. We discuss quality improvement methods in Section 2.1.

(2) Cost reduction. Since humans are not free, if there are large quantities of tasks, it is expensive to leverage humans to address all of them. For example, in entity resolution, if there are 10, 000 records, there will be about 50 million pairs. Even if the price per pair is 1 cent, it still takes much money. There are several effective cost reduction techniques. The first is pruning, which utilizes machine-based algorithms to remove some unnecessary(easy) tasks. The second is task selection, which prioritizes which tasks will be assigned to humans. The third is the answer deduction, which utilizes current answers to deduce the results of other tasks. The fourth is sampling, which samples a subset of tasks to ask humans and then propagates the answers to the entire dataset. We review cost reduction methods in Section 2.2.

(3) Latency reduction. Generally speaking, humans are much slower than the computer. To accomplish the tasks efficiently, we should reduce the latency, which is the time from the user submits the first task to the final answer is returned. There are mainly two strategies for latency control. The first is the round-based model, which leverages the idea that tasks can be published in multiple rounds. If there are enough humans to answer them, the latency of answering tasks in each round can be regarded as constant time. Thus the overall latency is modeled as the number of rounds. The second one is the statistical model, which uses the collected statistics from previous tasks to build statistical models that can estimate the humans latency for different tasks. We review latency control methods in Section 2.3.

(4) Active learning. Given a task, a user does not always have enough budget to label a large number of training data. Therefore, active learning is proposed to involve humans to label the most interesting examples iteratively so that the examples each iteration impact the model as much as possible. There are several methods to describe how interesting each example is, including uncertainty, expected model change, expected error

reduction, etc. We review latency control methods in Section 2.4.

(5) Weak supervision. In active learning, we always assume the labels provided by humans are perfect, but it does not hold in reality. Therefore, weak supervision is proposed to obtain a relative high quality result through weak labels, which are provided by humans with different qualities or functions(rules). Weak supervision can be categorized into two classes. The first one is data programming that generates a large number of weak labels using multiple labeling functions, which are written bu humans. The second one is fact extraction, which generates weak labels using existing sources like knowledge bases. We review weak supervision in Section 2.5.

Given the above general techniques about human-based machine learning, we introduce how to apply them in different modules in ML pipeline in section 3, including data extraction, data integration, data cleaning and iterate labeling. Then we discuss some future directions in section 4 and conclude in section 5.

2 Overall Techniques of Human-in-the-loop Machine Learning

As shown in section 1, humans are involved frequently and necessarily in a machine learning pipeline. They can not only contribute to the data preprocessing steps, but also provide a large amount of labeled training data to build a well-performed machine learning model, especially for the deep learning [16]. No matter what roles humans play in a ML pipeline, there exist some common sophisticated techniques to apply. In this section, we summarize some significant techniques in human involved machine learning. First, when humans are asked to conduct data annotation or data preprocessing, they are always required to provide high quality results, so we should study how to improve the quality of human answers in section 2.1. Second, since humans are not free, we study how to save monetary cost while not sacrificing much quality in section 2.2. Third, since humans cannot perform as quickly as machines, latency should be reduced to accelerate the entire ML process(section 2.3). Besides, active learning(section 2.4) focuses on selecting the most interesting examples to human for labeling to improve the model iteratively, which is an advanced technique in the field of machine learning. Lastly, in section 2.5, we discuss the situation where the user cannot derive a number of high quality labels, so she has to use weak supervision techniques to build a model based on weak labels, still with satisfying performance.

2.1 Quality Improvement

Human answers may not be reliable because (1) there exist malicious humans that randomly return answers, especially in the crowdsourcing scenario and (2) some tasks are difficult for humans to answer. Therefore, it is significant to discover different characteristics of humans and tasks, which can be leveraged to improve the quality. There are two commonly-used techniques for quality improvement, i.e., truth inference and task assignment, which will be introduced as follows.

Truth Inference. To control the quality, an intuitive idea is to assign each task to multiple humans, aggregate the answers and infer the truth. Note that humans may provide low quality or even malicious answers because they may also have different levels of expertise, and an untrained human may be incapable of answering certain tasks. Therefore, to achieve high quality, we need to tolerate human errors and infer high-quality results from noisy answers.

A unified quality control framework consists of the following three steps. First, we initialize each human's quality. Second, we infer the truth based on the collected answers and current quality. Third, we estimate the quality according to the inferred truth. Then we iterate the second and third steps until converge. Based on the unified framework, existing works [45, 70] can be categorized based on the following three factors: task modeling, human modeling and applied techniques(how to use task and human modeling to infer the truth).

(1) Task Modeling. This describes how existing solutions model a task, mainly including the difficulty of a task and the latent topics in a task [45, 70]. First, some recent works model the difficulty levels of a task instead of assuming that a human has the same quality for answering all the tasks. The more difficult a task, the harder a

human can provide a perfect answer for it. For example, in [70], $Pr(v_i^w = v_i^*|d_i, q^w) = 1/(1 + e^{d_i q^w})$ denotes the probability that human w correctly answers task t_i, where $d_i \in (0, +\infty)$ represents the difficulty of task t_i. The higher d_i, the easier task the t_i . Intuitively, for a fixed human quality $q^w > 0$, an easier task (high value of d_i) leads to a higher probability that the human correctly answers the task. Second, some recent works model the difficulty as a vector with K values instead of a single value. The basic idea is to exploit diverse topics of a task, where K is the pre-defined number of topics. For example, existing works [20, 45] apply topic model techniques on text description of each task to derive the topic vector. Besides, entity linking techniques are utilized to infer the topic vector for each task [81].

(2) Human Modeling. This describes how existing works model a human's quality, which is always denoted as a single real number $q^w \in [0, 1]$, representing the probability that human w answers a task correctly. This straightforward model has been widely adopted by existing works [41, 15, 4]. More specifically, for single-choice tasks, existing works [63, 31, 55] extend the above model to the confusion matrix to model the human quality in a more fine-grained way. Suppose each task in has l fixed choices, then the confusion matrix q^w is an ll matrix, where the j-th $(1jl)$ row, i.e., $q_j^w = [q_{j,1}^w, q_{j,2}^w, ..., q_{j,l}^w]$, represents the probability distribution of human ws possible answers for a task if the truth of the task is the j-th choice. Each element $q_{j,k}^w$ denotes that given the truth of a task is the j-th choice, the probability that human w selects the k-th choice. For numeric tasks, human bias and variance are proposed to model the human quality [55, 69]. Bias measures the effect that a human may underestimate (or overestimate) the truth of a task and variance measures the variation of errors around the bias. What's more, existing works [28, 38] introduce confidence in quality control, i.e., if a human answers many tasks, then the estimated quality for her is confident; otherwise the estimated quality is not confident. Inspired by this , [38] assigns higher qualities to the humans who answer plenty of tasks.

(3) Applied Techniques. In this part, we discuss how existing works leverage task models and human models to solve the truth inference problem. In general, existing works adopt the aforementioned unified framework, which can be categorized as the following three classes: straightforward computation [23, 50], optimization methods [4, 38, 39, 83] and probabilistic graphical model methods [30, 41]. First, the straightforward computation are some baseline models that estimate the truth without modeling the human or tasks. For single-label tasks, they always use the majority voting to address. For numerical tasks, mean and median are two baseline methods that regard the mean and median of humans answers as the truth. Second, optimization methods focus on designing optimization functions that capture the relations between humans qualities and tasks truth, and then provide an iterative method to compute these two sets of parameters. The differences among existing works [4, 38, 39, 83] are that they model humans qualities based on the above human modeling part differently. Third, probabilistic graph models a humans quality as a node and utilize graphical model inference to iteratively derive humans models [30, 41], where a graphical model is a graph, containing nodes and edges between pairs of nodes. Each node represents a random variable, which can be unknown parameters or observed data, and each edge represents the possible relationship (e.g., conditional dependency) between the linked pair of nodes.

Task Assignment. Since humans diverse backgrounds and qualities on tasks, a sophisticated task assignment algorithm will judiciously select tasks to right humans. Existing works mainly focus on two scenarios: (1) human-based, i.e., given a task, which subset of humans should be selected to answer the task; (2) task-based, i.e., when a human comes, which subset of tasks should be assigned to the human.

(1) Human-based. In this scenario, given a task and a set of candidate humans, the focus is on studying which subset of humans should be selected to answer the task in order to maximize the tasks quality without exceeding the overall budget. The problem is often called the Jury Selection Problem [6, 80]. Intuitively, humans with high quality should be selected. To this end, Cao et al. [9] provide a framework that first studies how to compute the quality of a given subset of humans before they give answers, called Jury Quality (JQ). Since the answers are unknown in advance, all possible cases of humans answers should be considered to compute the quality. To address this, Cao et al. [6] propose a Majority Voting strategy to compute the JQ. Zheng et al. [80] prove that Bayesian Voting is the optimal strategy under the definition of JQ. That is, given any fixed subset of humans S, the JQ of S w.r.t. the Bayesian Voting strategy is not lower than the JQ of S w.r.t. any other strategy. Therefore,

given a set of humans, its JQ w.r.t. Bayesian Voting strategy is the highest among all voting strategies.

(2) Task-based. In this scenario, when a human comes, the focus is on studying which subset of tasks should be assigned to the coming human. This problem is often called the Online Task Assignment Problem. When a human comes, [42, 5] compute an uncertainty score for each task based on collected answers, select the k most uncertain tasks, and assign them to the human. There are multiple methods to define the uncertainty. Liu et al. [42] use a quality-sensitive answering model to define each tasks uncertainty, and Boim et al. [5] leverage an entropy-like method to compute the uncertainty of each task. Besides, some other works [78, 79] model humans to have diverse skills among different domains, and choose the tasks from the domains that a coming human is good at to assign. What's more, many machine learning techniques [27, 82, 48] aim to assign a set of tasks to workers that are most beneficial to their trained models.

2.2 Cost Reduction

Humans are not free. Even if we turn to some cheap resources, like crowdsourcing for help to address the work, it can be still very expensive when there are a large number of tasks. Therefore, how to reduce the cost without sacrificing the quality is a big challenge. In this part, we introduce four kinds of techniques to reduce the human costs.

Pruning. Given a large number of tasks, pruning means that the user can conduct some preprocessing operations on then, so that some tasks are not necessary to be checked by humans. The basic idea is that some easy tasks can be addressed by the computer while the hard ones are left to humans. Pruning has been widely adopted in the area of human-powered join [15, 65, 67, 13, 12] an selection [74]. For example, the crowdsourcing join asks the human to identify records that refer to the same entity in the real world. To this end, the machine can compute a string similarity score for each pair of entities. Intuitively, those entities with very low(high) score are likely to be non-matching(matching) pairs, which can be easily solved purely by machine. For the rest hard ones, we can turn to the human for help. The advantage of this technique is that it is very straightforward, easy to implement and effective in many scenarios. However, the risk is that those pruned tasks cannot be checked by human, which may incur noise. Also, the threshold of deciding which part to prune is difficult to set.

Task Selection. Task selection has been introduced in section 2.1 for quality improvement. From another point of view, task selection can be seen as minimizing the human cost with a quality constrain. Different applications need different task selection strategies, such as join [15, 65, 67, 13, 12], top-k/sort [24, 37] categorize [51], etc. The basic idea is that given a task, a task selection strategy is first used to judiciously select a set of most beneficial tasks. Then after these tasks are sent to a platform with humans, a task assignment strategy is then used to collect high-quality answers from them. In a word, the task selection can achieve a good trade-off between cost and quality, especially the cost saving under a quality requirement. However, the downside is that it will incur much latency because the tasks are sent out iteratively.

Answer Deduction. Answer deduction can be adopted when the given tasks have some inherent relationships, which can be utilized to reduce the cost. Specifically, given a set of tasks, after deriving some results from humans, we can use this information to deduce some other tasks results, saving the cost of asking the crowd to do these tasks. Many operators have such property, e.g., join [67, 13, 12], planning [29, 77], mining [3]. For example, suppose a join operator generates three tasks: (A, B), (B, C), and (A, C). If we have already known that A is equal to B, and B is equal to C, then we can deduce that A is equal to C based on transitivity, thereby avoiding the crowd cost for checking (A, C).

Sampling. A sampling-based technique only utilizes the humans to process a sample of data and then leverage their answers on the sample to deduce the result on the entire data. This technique has been shown to be very effective in human-powered aggregation [46], and data cleaning [119]. For example, Wang et al. [66] propose a sample-and-clean framework that allows the human to only clean a small sample of data and uses the cleaned sample to obtain high-quality results from the entire data.

2.3 Latency Reduction

Given all tasks submitted by a user, latency denotes the time until all tasks have been accomplished. Since humans need time to think and answer, they will be much slower than the machine, so it is necessary to reduce the latency. Existing approaches can be categorized into the round-based model and statistical model.

Round-based Model. In some cases, tasks are answered in multiple rounds. In each round, we can utilize task selection techniques to select a bunch of tasks. For these tasks, we can leverage multiple humans to answer them so that the latency can be reduced. Concretely, suppose there are enough humans, some existing works [58, 64] simplify the definition of latency by assuming that each round spends 1 unit time, and then the latency is modeled as the number of rounds. They use the round model to do latency control. To this end, answer deduction is applied to reduce the number of tasks. More specifically, tasks that do not have relationships will be asked in parallel in a single round, so that some answers of other tasks can be deduced without any more costs. Therefore, since the total number of tasks can be reduced, the latency will be reduced.

Statistical Model. Some existing works [74, 22] utilize statistics information from real crowdsourcing platforms to model workers behaviors. Yan et al. [74] build statistical models to predict the time of answering a task, which considers (1) delay for the arrival of the first response; (2) the inter-arrival times between two responses. Faradani et al. [22] leverage statistical models to predict workers arrival rate in a crowdsourcing platform and characterize how workers select tasks from the platform.

2.4 Active Learning

Active learning is a commonly used technique in machine learning, which involves humans to label the most interesting examples iteratively. It always assumes that humans can provide accurate answers. The key challenge is that given a limited budget, how to select the most appropriate examples in each iteration. Active learning has been covered extensively in surveys [59, 56], so we only cover the most prominent techniques in this part. Next, we will introduce several strategies of selecting items to be labeled in each iteration.

Uncertainty sampling. Uncertainty sampling [34] is one of the simplest and commonly used methods in active learning, which selects the next unlabeled example which the current model regards as the most uncertain one. For example, when using a probabilistic model for binary classification, uncertainty sampling chooses the example whose probability is the nearest to 0.5. If there are more than three labels, a more general uncertainty sampling variant should be query the example whose prediction is the least confident. However, this approach throws away the information of other possible labels. Therefore, some researchers propose the marginal sampling, which chooses the example whose probability difference between the most and second likely labels is the smallest. This method can be further generalized by introducing the entropy for measuring the uncertainty.

Query-by-committee (QBC). The QBC [62] approach extends uncertainty sampling by maintaining a committee of models which are trained on the same labeled data. Each committee member can vote when testing each example, and the most informative example is considered to be the one where most models disagree with each other. The fundamental idea is to minimize the version space, which is the space of all possible classifiers that give the same classification results as the labeled data.

Expected model change. Another general active learning framework utilizes the decision-theoretic approach, choosing the example that would introduce the greatest change to the current model with the assumption that the label is known. A strategy of this framework is the expected gradient length (EGL) approach [61] for a discriminative probabilistic model, which can be applied to any learning problem where gradient-based training is used. In EGL, the change to the model can be measured as the length of training gradient. In other words, we should select the example that will lead to the largest gradient if it is labeled. However, since the true label is not known, we should compute the length of training gradient as an expectation over possible labels.

Expected error reduction. Another decision-theoretic method [57] aims to measure how much its generalization error is likely to be reduced rather than how much the model is likely to change. Given an example, the

basic idea is to first estimate the expected future error of the model trained using the example together with current labeled data on the remaining unlabeled examples. Then the example induced the smallest error is selected. Similar to the EGL method, since we do not know the true label of each unlabeled example, the expectation of future error over all possible labels should be computed.

Density-weighted methods. The mentioned frameworks above are likely to choose the outlier examples, which might be uncertain and disagreeing but not representative. However, most time the outliers contribute less than the representative examples which follow the similar distribution of the entire dataset. Therefore, existing works [60, 71] focus on choosing examples not only uncertain or disagreeing, but also representative of the example distribution.

2.5 Weak Supervision

In the above section, active learning approaches always involve experts without generating noise into the machine learning iterations. However, some real applications always need a large number of training labels and asking experts to do so heavy work is expensive. Therefore, existing works [53, 19, 47] have focused on the weak supervision, which generates large amount of labels semi-automatically. These labels are not perfect but good enough to result in a reasonably-high accuracy. Next we summarize two techniques with respect to the weak supervision.

Data programming. Data programming [54] has been proposed to generate a large number of weak labels using multiple labeling functions rather than labeling for each example. Each function can be written by the human and the Snorkel system [53] provides a friendly interface to support it. Obviously, a single function is not effective enough to derive a well-performed model, so multiple functions should be combined to generate labels. The most straightforward method of combination is majority voting, but it does not consider the correlations and qualities of different functions. To address this, Snorkel [53] proposed a probabilistic graphical model to generate the weak labels which is followed by a discriminative model trained on the weak labels.

Fact extraction. Fact extraction is another way to generate weak labels using knowledge base, which contains facts extracted from different sources including the Web. A fact usually describes entities with attributes and relations, such as <China, capital, Beijing>, which indicates the capital of China is Beijing. The facts can be regarded as labeled examples, which can be used as seed labels for distant supervision [47]. Besides fact extraction can also be considered as extracting facts from multiple resources to construct a knowledge base. The Never-Ending Language Learner (NELL) system [19] continuously extracts structured information from the unstructured Web and constructs a knowledge base. Initially, NELL starts with seeds that consist of an ontology of entities and relationships among them. Then NELL explores large quantities of Web pages and identifies new entities pairs, which has the same relationships with seeds based on the matching patterns. The resulting entity pairs can then be used as the new training data for constructing even more patterns. The extraction techniques can be regarded as distant supervision generating weak labels.

3 Human-in-the-loop Machine Learning Pipeline

As shown in Figure 1, humans play many significant roles in machine learning pipeline. First, given some unstructured data, we have to transform it structured data, in order to construct features for ML. Then for structured data from multiple sources, we should integrate them for enriching data and features to achieve well-performed ML model. What's more, data is always dirty in the real world. To further improve the performance, we should clean the data, such as repairing records that violate integrity constraint and removing outliers and duplicates. Finally, we should annotate the data for building the model. For all above steps in the pipeline, humans can contribute their intelligence to provide high quality training data and improve the ML model. Next, we will introduce what humans can do in these steps, using the techniques proposed in section 3.

3.1 Data Extraction

Extracting structured data from unstructured data is an important problem both in industry and research, which has been studied broadly from rule-based [40] systems to ML-based approaches [49, 19]. However, these methods either need domain experts to design rules or humans to provide large quantities of labels. Recently, DeepDive [75] is a representative system in this area, which provides declarative language for non-expert users to extract data. The execution of DeepDive can be divided into three parts: candidate generation, supervision, statistical inference and learning. Humans mainly contribute in the first part, i.e., candidate generation. In this part, humans write some extraction rules described by declarative languages to retrieve data with attributes or relations, such as entity B is the wife of A if there exists mention "and his wife" between A and B in a corpus. The goal of this part is to generate candidates with high recall, low precision. Secondly, we come to the supervision part, which applies distant supervision rules from knowledge bases or incomplete databases to provide labels for some of the candidates. The rules do not need to label all candidates from the first part, which are intended to be a low recall, high precision. For the last part, they construct a graphical model that represents all of the labeled candidate extractions, train the model, and then infer a correct probability for each candidate. At the end of this stage, they apply a threshold to each inferred probability and then derive the extractions to the output database. In conclusion, Deepdive leverages humans to provide extraction candidates with high recall, use weak supervision(distance supervision) to label them and finally train a statistical ML model to fine-tune the labels.

3.2 Data Integration

Given relational tables from multiple sources, in many cases we want to integrate them for extending existing datasets, including features and records. To this end, schema matching [76, 21] and join (entity resolution [65, 67]) have to be done, where the first part is going to align the columns and the second will match records from different tables. Recently, many existing works have been focused on leveraging human intelligence to do that.

For schema matching, existing works [76] utilize human-machine hybrid approaches to improve the performance. They utilize machine-based schema matching tools to generate a set of possible matchings, each of which has a probability to be matched. They define a correspondence correctness question(CCQ) for humans to answer, which denotes a pair of attributes from two columns, so each matching consists of a set of correspondences. Then the problem is to wisely choose the correspondences to ask the human to obtain the highest certainty of correct schema matching at the lowest cost. The uncertainty is measured by entropy on top of the probabilities that the tools generate. In the correspondence selection, they consider the column correlations, selection efficiency and human quality to match schemes effectively and efficiently. Fan et.al. [21] introduce knowledge base together with humans to do schema matching, First, they propose a concept-based approach that maps each column of a table to the best concept in knowledge bases. This approach overcomes the problem that sometimes values of two columns may be disjoint, even though the columns are related, due to incompleteness in the column values. Second, they develop a hybrid machine-crowdsourcing framework that leverages human intelligence to discern the concepts for difficult columns. The overall framework assigns the most beneficial column-to-concept matching tasks to the human under a given budget and utilizes the answers to infer the best matching.

After the schemes are aligned, we can integrate different relational tables by the join operation. Traditionally, join is always executed by exact matching between values of attributes from two tables. However, in the real world, data is always dirty. For example, "Apple iphone 8" and "iphone 8th" refer to the same entities and should be joined, which cannot be done by a traditional database. Therefore, the human-based join is proposed to address this problem. Wang et.al. [65] propose crowd-based join framework, which generates many candidate pairs, uses similarity based pruning techniques to eliminate dissimilar pairs and ask the crowd to answer the rest pairs. To further reduce the cost, Wang et.al. [67] leverage the transitivity technique to deduce unknown answers

based on current answers from humans. Chai et.al. [13, 12] build a partial-order graph based on value similarities of different attributes and utilize the graph to prune pairs that are not necessary to ask. To improve the quality, Wang et.al. [68] first cluster the entities to be joined and then leverage humans to refine the clusters. Yalavarthi et.al. [73] select questions judiciously considering the crowd errors.

3.3 Data Cleaning

Data is always dirty in the real world, which is likely to hurt the ML performance. For example, some values may be out of range (e.g., a age is beyond 120 or below 0) or utilize wrong units (e.g., some distance are in meters while other are in kilometers); Some records refer to the same entity; Integrity constraints (e.d. functional dependencies) are violated among records. Recently, many researchers have focused on leveraging human to clean the data. For instance, crowd-based entity resolution [65, 13, 12] is always applied to remove duplicates. Chai et.al. [7] use human experts to identify outliers among the data. Specifically, they first utilize machine-based outlier detection algorithms to detect some outlier candidates as well as inlier candidates, and then human is asked to verify these candidates by comparing outlier candidates with inliers. Chu et.al. [14] clean the data that violates integrity constraints with the help of knowledge base and humans. They first identify the relationships between columns using knowledge base and then use humans to verify them. Then the discovered relationships can be utilized to detect errors among data, and then these error can be repaired by the knowledge base and humans.

Recently, a line of interesting data cleaning works focus on cleaning with the explicit goal of improving the ML results. Wang et al. [32] propose a cleaning framework ActiveClean for machine learning tasks. Given a dataset and machine learning model with a convex loss, it selects records that can most improve the performance of the model to clean iteratively. ActiveClean consists of 4 modules, sampler, cleaner, updater and estimator. Sampler is used to select a batch of records to be cleaned. The selection criterion is measured by how much improvement can be made after cleaning a record, i.e., the variation of the gradient, which is estimated by the Estimator. Then the selected records will be checked and repaired by the Cleaner, which can be humans. Next, the Updater updates the gradient based on these verified dirty data. The above four steps are repeated until the budget is used up. BoostClean [33] cleans the data where an attribute value is out of range. It takes as input a dataset and a set of functions for detecting errors and repair functions. These functions can be provided by humans. Each pair of detection and repair functions can produce a new model. BoostClean uses statistical boosting to find the best ensemble of pairs that maximize the final performance. Recently, TARS [17] was proposed to clean human labels using oracles, which provides two pieces of advice. First, given test data with noisy labels, TARS estimates the performance of the model on true labels, which is shown to be unbiased and confidence intervals are computed to bound the error. Second, given training data with noisy labels, TARS determines which examples to be sent to an oracle so as to maximize the expected model improvement of cleaning each noisy label.

3.4 Iterative Labeling

After the above steps of data preprocessing, we can label the data in relational tables for ML tasks. The most straightforward method is to directly leverage humans to annotate a bunch of data for training. Thus we can adopt the cost control and quality control approaches proposed in section 2 to derive high quality labels with low cost (see [36] for a survey). However, in many cases, a user does not have enough budget to obtain so many annotations. Therefore, many researchers have focused on how to label data iteratively and make the model performance better and better using techniques like active learning or weak supervision.

Mozafari et.al. [48] use active learning to scale up the human labeling, which can be utilized in two scenarios, the upfront and iterative scenario. In the upfront scenario, the user cares more about the latency than the cost. Therefore, given a budget and an initial model, the algorithm uses a ranker to rank and select some of the most

informative examples to label while the rest are predicted by the model. In the iterative scenario, since the user cares more about the cost, the ranker selects a batch of examples to label, retrains the model and selects again until the budget is used up. There are two strategies(Uncertainty and MinExpError) that the user can choose for ranking. Leveraging the traditional active learning technique, Uncertainty selects examples that the current model is the most uncertain about. MinExpError uses a more sophisticated algorithm that considers both the uncertainty and expected model change. Besides, the work also utilizes the bootstrap theory, which makes the algorithms available to any classifier and also enables parallel processing. Also, active learning techniques in section 2.4 can also be integrated in the framework.

DDLite [18] leverage human to conduct data programming rather than hand-labeling data, in order to generate large quantities of labels. Given a set of input documents, DDLite aims to produce a set of extracted entities or relation mentions, which consists of four steps. First, given input documents, preprocessing like domain-specific tokenizers or parsers of the raw text has to be performed. Second, DDLite provides a library of general candidate extraction operators, which can be designed by humans. Third, humans develop a set of labeling functions through iterating between labeling some small subsets and analyzing the performance of labeling functions. Lastly, features are automatically generated for the candidates, and then the model is trained using the labeling functions. The humans then analyze the performance on a test set.

4 Future Work

Data discovery for ML. Suppose an AI developer aims to build a ML model. Given a dataset corpus, the user requires to find relevant datasets to build the model. Data discovery aims to automatically find relevant datasets from data warehouse considering the applications and user needs. Many companies propose data discovery systems, like Infogather [72] in Microsoft and Goods [25] in Google. However, most such systems focus on keyword-based dataset search or just linking datasets. Therefore, it may be worth studying to discover datasets that directly can maximize the performance of the downstream ML model. The key challenges lie in how to find valuable features and data among the corpus.

Crowdsourcing. Recently, famous crowdsourcing platforms like AMT [2] provide hundreds of thousands of humans who can process big data tasks. Even though many works [35, 10, 44, 11, 8, 9, 52, 26, 43] have been proposed to study how to improve the humans' quality and save the cost, there still exist some challenges to be solved. For example, existing crowdsourcing works mainly focus on micro-tasks like classification because the tasks are easy to decompose and the quality is easy to control. However, macro-tasks, like designing a data repairing rule, are also important in ML task. Therefore, how to design macro tasks and derive high quality results are challenging problems.

Benchmark A large variety of TPC benchmarks (e.g., TPC-H for analytic workloads, TPC-DI for data integration) standardize performance comparisons for database systems and promote the development of the database community. Even though there are some open datasets crowdsourcing or machine learning tasks, there is still lack of standardized benchmarks that covered the entire human involved machine learning pipeline. To better explore the research topic, it is significant to study how to develop evaluation methodologies and benchmarks for the human-in-the-loop machine learning system.

Modules selection in ML pipeline. Figure 1 shows the standard ML pipeline from data preparation to the model training and testing which consists of several modules like schema matching, data cleaning and integration, etc, and data cleaning can also be extended to many scenarios, like missing values, outliers and so on. Given a ML task, asking the humans to process all modules is expensive, but may not be necessary. Thus, we can study which modules are significant to the ML model and drop the other ones. For example, given a classification task, some data cleaning tasks like removing duplicates are not necessary. Therefore, how to select modules to optimize the ML pipeline is worth to study.

Trade-off between human quality and model performance. Some existing works [53, 19, 47] focus on

acquire weak labels to derive a model with good performance. One idea is to study the trade-off between human quality and model performance. That is, given a performance requirement, such as 80% F-measure, we can decide how to select humans to label the training data with the goal of optimizing the cost. Also, given human qualities, we can study how to produce results with the highest quality.

5 Conclusion

In this paper, we review extensive studies in human-in-the-loop machine learning. We first introduce five commonly used techniques in this field, including quality improvement, cost reduction, latency reduction, active learning and weak supervision. For quality, we discuss the truth inference and task assignment techniques. For cost, we review the pruning, task selection, answer deduction, and sampling, techniques. For latency, we survey the round-based and statistical model. Also we study many active learning strategies to select interesting examples and weak supervision(including data programming and fact extraction). Then we review how to apply these techniques to the ML pipeline including data extraction, data integration, data cleaning and labeling.

References

[1] https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/.

[2] Amazon mechanical turk. https://www.mturk.com/.

[3] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. OASSIS: query driven crowd mining. In *SIGMOD 2014*, pages 589–600.

[4] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In C. E. Brodley and P. Stone, editors, *AAAI, 2014*, pages 2946–2953.

[5] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W. C. Tan. Asking the right questions in crowd data sourcing. In *(ICDE 2012*, pages 1261–1264.

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

[7] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden. Human-in-the-loop outlier detection. In *SIGMOD 2020*, pages 19–33.

[8] C. Chai, J. Fan, and G. Li. Incentive-based entity collection using crowdsourcing. In *ICDE 2018*.

[9] C. Chai, J. Fan, G. Li, J. Wang, and Y. Zheng. Crowdsourcing database systems: Overview and challenges. In *ICDE 2019*, pages 2052–2055.

[10] C. Chai, G. Li, J. Fan, and Y. Luo. Crowdsourcing-based data extraction from visualization charts. In *ICDE, 2020*, pages 1814–1817.

[11] C. Chai, G. Li, J. Fan, and Y. Luo. Crowdchart: Crowdsourced data extraction from visualization charts. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[12] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *SIGMOD 2016*, pages 969–984.

[13] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. A partial-order-based framework for cost-effective crowdsourced entity resolution. *VLDB J.*, 27(6):745–770, 2018.

[14] X. Chu, M. Ouzzani, J. Morcos, I. F. Ilyas, P. Papotti, N. Tang, and Y. Ye. KATARA: reliable data cleaning with knowledge bases and crowdsourcing. *Proc. VLDB Endow.*, 8(12):1952–1955, 2015.

[15] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *WWW 2012*, pages 469–478.

[16] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR 2009*, pages 248–255.

[17] M. Dolatshah, M. Teoh, J. Wang, and J. Pei. Cleaning crowdsourced labels using oracles for statistical classification. *Proc. VLDB Endow.*, 12(4):376–389, 2018.

[18] H. R. Ehrenberg, J. Shin, A. J. Ratner, J. A. Fries, and C. Ré. Data programming with ddlite: putting humans in a different part of the loop. In *HILDA@SIGMOD 2016*, page 13.

[19] T. M. M. et.al. Never-ending learning. In *AAAI 2015*, pages 2302–2310.

[20] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD, 2015*, pages 1015–1030.

[21] J. Fan, M. Lu, B. C. Ooi, W. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *ICDE 2014*, pages 976–987.

[22] S. Faradani, B. Hartmann, and P. G. Ipeirotis. What's the right price? pricing tasks for finishing on time. In *AAAI Workshop 2011*.

[23] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 61–72. ACM, 2011.

[24] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD 2012*, pages 385–396.

[25] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods: Organizing google's datasets. In *SIGMOD 2016*, pages 795–806.

[26] S. Hao, C. Chai, G. Li, N. Tang, N. Wang, and X. Yu. Outdated fact detection in knowledge bases. In *ICDE 2020*, pages 1890–1893.

[27] C. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 534–542.

[28] M. Joglekar, H. Garcia-Molina, and A. G. Parameswaran. Evaluating the crowd with confidence. In *SIGKDD 2013*, pages 686–694.

[29] H. Kaplan, I. Lotosh, T. Milo, and S. Novgorodov. Answering planning queries with the crowd. *Proc. VLDB Endow.*, 6(9):697–708, 2013.

[30] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *NIPS 2011*, pages 1953–1961.

[31] H. Kim and Z. Ghahramani. Bayesian classifier combination. In N. D. Lawrence and M. A. Girolami, editors, *AISTATS 2012*, volume 22 of *JMLR Proceedings*, pages 619–627.

[32] S. Krishnan, M. J. Franklin, K. Goldberg, J. Wang, and E. Wu. Activeclean: An interactive data cleaning framework for modern machine learning. In *SIGMOD 2016*, pages 2117–2120.

[33] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu. Boostclean: Automated error detection and repair for machine learning. *CoRR*, abs/1711.01299, 2017.

[34] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR 1994*, pages 3–12.

[35] G. Li and C. C. et.al. CDB: optimizing queries with crowd-based selections and joins. In *SIGMOD, 2017*.

[36] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 39–40. IEEE Computer Society, 2017.

[37] K. Li, X. Zhang, and G. Li. A rating-ranking method for crowdsourced top-k computation. In *SIGMOD 2018*, pages 975–990.

[38] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han. A confidence-aware approach for truth discovery on long-tail data. *Proc. VLDB Endow.*, 8(4):425–436, 2014.

[39] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD 2014*, pages 1187–1198.

[40] Y. Li, F. Reiss, and L. Chiticariu. Systemt: A declarative information extraction system. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - System Demonstrations*, pages 109–114. The Association for Computer Linguistics, 2011.

[41] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. In *NIPS 2012*, pages 701–709.

[42] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *Proc. VLDB Endow.*, 5(10):1040–1051, 2012.

[43] Y. Luo, C. Chai, X. Qin, N. Tang, and G. Li. Interactive cleaning for progressive visualization through composite questions. In *ICDE, 2020*, pages 733–744.

[44] Y. Luo, X. Qin, C. Chai, N. Tang, G. Li, and W. Li. Steerable self-driving data visualization. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[45] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *SIGKDD 2015*, pages 745–754.

[46] A. Marcus, D. R. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. *Proc. VLDB Endow.*, 6(2):109–120, 2012.

[47] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL 2009*, pages 1003–1011.

[48] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *Proc. VLDB Endow.*, 8(2):125–136, 2014.

[49] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM 2011*, pages 227–236.

[50] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM 2012*, pages 1203–1212.

[51] A. G. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *Proc. VLDB Endow.*, 4(5):267–278, 2011.

[52] X. Qin, C. Chai, Y. Luo, N. Tang, and G. Li. Interactively discovering and ranking desired tuples without writing SQL queries. In *SIGMOD 2020*, pages 2745–2748.

[53] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: rapid training data creation with weak supervision. *VLDB J.*, 29(2-3):709–730, 2020.

[54] A. J. Ratner, C. D. Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *NIPS 2016*, pages 3567–3575.

[55] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(4), 2010.

[56] F. Ricci, L. Rokach, and B. Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.

[57] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *(ICML 2001*, pages 441–448.

[58] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and A. Y. Halevy. Crowd-powered find algorithms. In *ICDE 2014*, pages 964–975.

[59] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[60] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP 2008*, pages 1070–1079.

[61] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *NIPS 2007*, pages 1289–1296.

[62] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT 1992*, pages 287–294.

[63] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *WWW 2014*, pages 155–164.

[64] V. Verroios, P. Lofgren, and H. Garcia-Molina. tdp: An optimal-latency budget allocation strategy for crowdsourced MAXIMUM operations. In *SIGMOD 2015*, pages 1047–1062.

[65] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *Proc. VLDB Endow.*, 5(11):1483–1494, 2012.

[66] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD 2014*, pages 469–480.

[67] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD 2013*, pages 229–240.

[68] S. Wang, X. Xiao, and C. Lee. Crowd-based deduplication: An adaptive approach. In *SIGMOD 2015*, pages 1263–1277.

[69] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[70] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. R. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *NIPS 2009*, pages 2035–2043.

[71] Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *ECIR 2007*, volume 4425 of *Lecture Notes in Computer Science*, pages 246–257.

[72] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD 2012*, pages 97–108.

[73] V. K. Yalavarthi, X. Ke, and A. Khan. Select your questions wisely: For entity resolution with crowd errors. In *CIKM 2017*, pages 317–326.

[74] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys 2010*, pages 77–90.

[75] C. Zhang, J. Shin, C. Ré, M. J. Cafarella, and F. Niu. Extracting databases from dark data with deepdive. In *SIGMOD 2016*, pages 847–859.

[76] C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *Proc. VLDB Endow.*, 6(9):757–768, 2013.

[77] C. J. Zhang, Y. Tong, and L. Chen. Where to: Crowd-aided path selection. *Proc. VLDB Endow.*, 7(14):2005–2016, 2014.

[78] Z. Zhao, F. Wei, M. Zhou, W. Chen, and W. Ng. Crowd-selection query processing in crowdsourcing databases: A task-driven approach. In *EDBT 2015*, pages 397–408.

[79] Z. Zhao, D. Yan, W. Ng, and S. Gao. A transfer learning based framework of crowd-selection on twitter. In *KDD 2013*, pages 1514–1517.

[80] Y. Zheng, R. Cheng, S. Maniu, and L. Mo. On optimality of jury selection in crowdsourcing. In *EDBT 2015*, pages 193–204. OpenProceedings.org.

[81] Y. Zheng, G. Li, and R. Cheng. DOCS: domain-aware crowdsourcing system. *Proc. VLDB Endow.*, 10(4):361–372, 2016.

[82] J. Zhong, K. Tang, and Z. Zhou. Active learning from crowds with unsure option. In Q. Yang and M. J. Wooldridge, editors, *IJCAI 2015*, pages 1061–1068.

[83] D. Zhou, J. C. Platt, S. Basu, and Y. Mao. Learning from the wisdom of crowds by minimax entropy. In *NIPS 2012*, pages 2204–2212.