An Evaluation of Aggregation Techniques in Crowdsourcing

Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer

École Polytechnique Fédérale de Lausanne
{quocviethung.nguyen,tam.nguyenthanh,ngoc.lam,karl.aberer}@epfl.ch

Abstract. As the volumes of AI problems involving human knowledge are likely to soar, crowdsourcing has become essential in a wide range of world-wide-web applications. One of the biggest challenges of crowdsourcing is aggregating the answers collected from the crowd since the workers might have wide-ranging levels of expertise. In order to tackle this challenge, many aggregation techniques have been proposed. These techniques, however, have never been compared and analyzed under the same setting, rendering a 'right' choice for a particular application very difficult. Addressing this problem, this paper presents a benchmark that offers a comprehensive empirical study on the performance comparison of the aggregation techniques. Specifically, we integrated several state-of-the-art methods in a comparable manner, and measured various performance metrics with our benchmark, including *computation time, accuracy, robustness to spammers,* and *adaptivity to multi-labeling*. We then provide in-depth analysis of benchmarking results, obtained by simulating the crowdsourcing process with different types of workers. We believe that the findings from the benchmark will be able to serve as a practical guideline for crowdsourcing applications.

1 Introduction

In recent years, crowdsourcing becomes a promising methodology to overcome problems that require human knowledge such as image labeling, text annotation, and product recommendation [14]. Leveraging this methodology, a wide range of applications [5] (e.g. ESP game [1], reCaptcha [2], and ZenCrowd [3]) have been developed on top of more than 70 platforms [1] such as Amazon Mechanical Turk and CloudCrowd. The rapid growth of such applications opens up a variety of technical challenges [16,9,8].

One of the most important technical challenges of crowdsourcing is answer aggregation [17], which aggregates a set of human answers into a single value. In our setting, we consider a broad class of problems in which there is an objective ground truth external to human judgment; i.e. each question has an exact answer but no one knows what it is. The goal of answer aggregation is to find this hidden ground truth from a set of answers given by the crowd workers. This goal is, however, difficult to achieve for two main reasons. First, the crowd workers have wide-ranging levels of expertise [20] , leading to high contradiction and uncertainty in the answer set. Second, the questions vary in different degrees of difficulty, resulting in an incorrect assessment of the true expertise between truthful workers and malicious workers. To fully overcome this challenge, a rich body of research has proposed different techniques for the answer aggregation.

[1] `http://www.crowdsourcing.org/`

In general, the aggregation techniques are broadly classified into two categories according to their computing model:

- **Non-iterative:** uses heuristics to compute a single aggregated value of each question separately. One simple approach is Majority Decision (MD) [13], in which the answer with highest votes is selected as the final aggregated value. Other techniques are Honeypot (HP) [15] and ELICE [12].
- **Iterative:** performs a series of iterations, each consisting of two updating steps: (i) updates the aggregated value of each question based on the expertise of workers who answer that question, and (ii) adjusts the expertise of each worker based on the answers given by him. This incremental mechanism serves as the basis in EM [7], GLAD [22], SLME [18], and ITER [10].

While many aggregation techniques have been developed over the last decades, there has been no work on the evaluation of their performance altogether. The main reason is the lack of a common setting (i.e. no common dataset and no common metrics of success). As a result, understanding the performance implications of these techniques is challenging, since each of them has distinct characteristics. One, for example, may achieve very high accuracy over certain types of workers, while another is sensitive to spammers. Moreover, aggregation techniques have never been compared systematically, and each work often reported its superior performance generally using a limited variety of datasets or evaluation methodologies. Therefore, there is a need of common settings to test, research, and assess the advantage and disadvantage of these techniques.

The primary goal of this paper is to evaluate aggregation techniques within a common framework. To this end, we present a benchmark that offers an overview of comprehensive performance comparison among the aggregation techniques, describes in-depth analysis on the performance behavior of each method, and provides guidance on the selection of appropriate aggregation schemes. Moreover, potential users (e.g. researchers and developers) can utilize our benchmarking framework to assess their own techniques as well as reuse its components to reduce the development complexity. Specifically, the salient features of the benchmark are highlighted as follows:

- We developed or integrated, in a fair manner for comparisons, the most representative state-of-the-art techniques in each category of answer aggregation approaches, including [2] MD, HP, ELICE, EM, GLAD, SLME, and ITER.
- We designed a generic, extensible benchmarking framework to assist in the evaluation of different aggregation techniques, so that subsequent studies are able to easily compare their proposals with the state-of-the-art techniques.
- We simulated different types of crowd workers and questions. In addition, our benchmark allows users to customize the distribution of these workers. By this way, the users can predict the accuracy of worker answers and save their money before really posting the questions to the crowd.
- We offer extensive as well as intensive performance analyses. We believe that the analyses can serve as a practical guideline for how to select a well-suited aggregation technique on particular application scenarios.

[2] Full names of all abbreviations are given in section 2.

The remainder of the paper is organized as follows. Section 2 reviews state-of-the-art aggregation techniques. We then describe the methodology used in the benchmark in Section 3. Section 4 offers in-depth discussions on the benchmark results. Section 5 finally summarizes and concludes this study, where we provide important suggestions for the applications that consider employing an aggregation technique.

2 Answer Aggregation Techniques

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

$$M = \begin{pmatrix} a_{11} & \ldots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nk} \end{pmatrix} \tag{1}$$

where $a_{ij} \in L$ is the answer of worker w_j for object o_i. The output of aggregation techniques is a set of aggregated values $\{\gamma_{o_1}, \gamma_{o_2}, \ldots \gamma_{o_n}\}$, 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 [3]:

$$\gamma_i = \arg\max_{l_z \in L} P(X_i = l_z) \tag{2}$$

In the following, we offer the details of aggregation techniques commonly used in the literature. We organize them into two categories: (i) *non-iterative aggregation*, including MD, HP, and ELICE; and (ii) *iterative aggregation*, including EM, GLAD, SLME, and ITER. Table 1 summarizes the important notations used in this paper.

2.1 Non-Iterative Aggregation

The literature suggests various non-iterative techniques, including Majority Decision (MD)[13], Honeypot (HP)[15], and ELICE[12]. They differ in the preprocessing step as well as the probability computation. In particular, MD does not require preprocessing. HP filters the answers of spammers in advance, whereas ELICE considers both worker expertise and question difficulty. This section presents the details for these three techniques, which cover the characteristics of other non-iterative methods.

Majority Decision Majority Decision (MD) is a straightforward method that aggregates each object independently. 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 workers might have different levels of expertise and it is especially problematic if most of them are spammers.

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

Table 1: Summary of important notations

Symbol	Description
$M_{n \times k}$	answer matrix of n objects and k workers
o_i, w_j, l_z	an object, a worker, a possible label
a_{ij}	answer of worker w_j for object o_i
γ_{o_i} or γ_i	aggregated value of object o_i
$P(X_i = l_z)$	the probability of object o_i that its aggregated value γ_i is l_z
Ω	a set of trapping questions used to test worker expertise

Table 2: Characteristics of aggregation techniques

algo	trapping set	aggregation model	worker expertise	question difficulty	computing model
MD	no	non-iterative	no	no	online
HP	yes	non-iterative	no	no	online
ELICE	yes	non-iterative	yes	yes	offline
EM	no	iterative	yes	no	offline
SLME	no	iterative	yes	no	offline
GLAD	no	iterative	yes	yes	offline
ITER	no	iterative	yes	yes	offline

Honeypot In principle, Honeypot (HP) operates as MD, except that untrustworthy workers 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. Workers who fail to answer a specified number of trapping questions are neglected as spammers and removed. Then, the probability of a possible label assigned for each object o_i is computed by MD among remaining workers. However, this approach has some disadvantages: Ω is not always available or is often constructed subjectively; i.e truthful workers might be misidentified as spammers if trapping questions are too difficult.

Expert Label Injected Crowd Estimation Expert Label Injected Crowd Estimation (ELICE) is an extension of HP. Similarly, ELICE also uses trapping questions Ω, but to estimate the expertise level of each worker by measuring the ratio of his answers which are identical to true answers of Ω. Then, it estimates the difficulty level of each question by the expected number of workers who correctly answer a specified number of the trapping questions. Finally, it computes the object probability $P(X_i = l_z)$ by *logistic regression* [6] that is widely applied in machine learning. In brief, ELICE considers not only the worker expertise ($\alpha \in [-1, 1]$) but also the question difficulty ($\beta \in [0, 1]$). The benefit is that each answer is weighted by the worker 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 about the trapping set Ω like HP as previously described.

2.2 Iterative Aggregation

Iterative aggregation is the approach that consists of a sequence of computational rounds. In each round, object probabilities–probability about possible labels of each object–are updated incrementally and this computation is repeated until convergence. This approach also differs from non-iterative one in the fact that the trapping set Ω is not required. The widely used techniques in this category includes EM, SLME, GLAD, and ITER. Each of them has different ways to initialize and update object probabilities. While EM and SLME only concern about worker expertise, GLAD and ITER consider both worker expertise and question difficulty. The details are explained as follows.

Expectation Maximization The Expectation Maximization (EM) technique [7] iteratively computes object probabilities in two steps: *expectation* (E) and *maximization* (M). In the (E) step, object probabilities are estimated by weighting the answers of workers according to the current estimates of their expertise. In the (M) step, EM re-estimates the expertise of workers based on the current probability of each object. This iteration is repeated until all object probabilities are unchanged. Briefly, EM is an iterative algorithm that aggregates many objects at the same time. Since it takes a lot of steps to reach convergence, running time is a critical issue.

Supervised Learning from Multiple Experts In principle, Supervised Learning from Multiple Experts (SLME) [18] also operates as EM, but characterizes the worker 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 $\gamma \in \{0, 1\}$).

Generative model of Labels, Abilities, and Difficulties Generative model of Labels, Abilities, and Difficulties (GLAD) [22] 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 worker 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 worker 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.

Iterative Learning Iterative Learning (ITER) is an iterative technique based on standard belief propagation [10]. It also estimates the question difficulty and the worker expertise, but slightly different in details. While others treat the reliability of all answers of one worker as a single value (i.e. worker expertise), ITER computes the reliability of each answer separately. And the difficulty level of each question is also computed individually for each worker. As a result, the expertise of each worker is estimated as the sum of the reliability of his 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 workers must answer all questions, ITER can divide questions into different subsets and the outputs of these subsets are propagated in the end.

2.3 Summary

To sum up, we already implemented seven aggregation techniques—MD, HP, ELICE, EM, SLME, GLAD, ITER—which aggregate worker answers by computing the probability of possible labels. Each technique exhibits various aggregation characteristics. In fact, often these characteristics are not exclusive; a technique might have multiple ones. Table 2 features each implemented technique with following key characteristics.

- **Trapping set:** the set of trapping questions, whose answers are known before-hand. It is mainly used to filter spammers and initialize the expertise of other workers.
- **Aggregation model:** computation model of answer aggregation. It provides the basic categorization of aggregation techniques and the indication of their complexity.

– **Worker expertise:** the ability to capture the behavior of a worker; i.e. the accuracy and reliability of his answers. This ability is important since human workers often have wide-ranging levels of knowledge.
– **Question difficulty:** the ability to measure the difficulty degree of questions. This ability is a supplement of worker expertise: answering an easy question incorrectly is worse than answering a difficult question incorrectly.
– **Computing model:** the ability to perform (online or offline) in response to the new arrival of worker answers. An online technique can process answer-by-answer in a serial fashion, whereas offline ones have to re-compute the whole aggregation.

One interesting point to note is that all of the above techniques support aggregation on questions with binary choices (i.e. yes/no questions). For the questions with multiple choices, only three algorithms—MD, HP, and EM—are applicable. Another worth-noting point is that estimating worker expertise can serve as a quality indicator in practical scenarios such as payment mechanism and worker profiling.

3 Benchmark Setup

This section describes the setup used in our benchmark. We first present the details for our benchmarking framework as well as the simulation of crowdsourcing process. We then offer an insight of the implementation of aggregation techniques followed by descriptions of the measures used to assess their performance.

3.1 Framework

A primary goal of this study is to provide a flexible and powerful tool to support the comparison and facilitate the benchmarking analysis of aggregation techniques. To this end, we have developed a framework that employs original performance studies of each technique. Figure 1 illustrates the simplified architecture of the framework. It is built upon a component-based architecture having three layers. (1) The data access layer abstracts the underlying data objects, and loads the data to the upper layer. (2) The application layer interacts with users to receive configurable parameters and visualize outputs from the computing layer. (3) The computing layer consists of two modules: (i) *aggregation module* and (ii) *simulation module*. On one hand, the aggregation module is responsible for invoking plugged algorithms (algorithm component) upon inputs from data access layer and delivering summarized information (evaluation component) to the application layer. On the other hand, the simulation module simulates the crowdsourcing process in which the workers (worker simulator) label a set of objects by answering various questions (answer simulator). This simulation will be described in Section 3.2.

We believe that subsequent studies are able to easily compare their algorithms with the state-of-the-art techniques by using our framework. It is flexible and extensible, since a new technique as well as a new measurement can be easily plugged in. Moreover, users are also supported to use their crowd simulators or real datasets. The framework is available for download from our website[4].

[4] `https://code.google.com/p/benchmarkcrowd/`

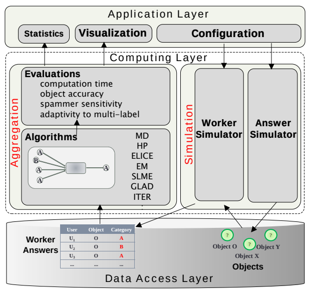

Fig. 1: Benchmarking framework

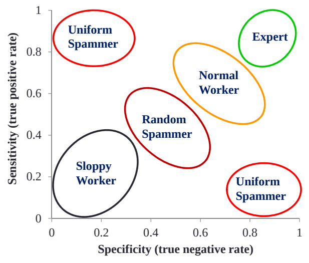

Fig. 2: Characterization of worker types

3.2 Crowd Simulation

The simulation module helps benchmark users simulate the crowdsourcing process in the literature. It is implemented with two components: (i) worker simulator—simulates different types of workers—and (ii) answer simulator—generates numbers of objects (questions) and their true labels (answers). Both of them demonstrate an online process where each worker is assigned to answer a set of questions. Details are provided below.

Worker simulator Many previous studies [11,21] characterized different types of crowd workers to reflect their expertise. Based on the classification in [21], we simulate 5 worker types as depicted in Figure 2. (1) *Experts:* who have deep knowledge about specific domains and answer questions with very high reliability. (2) *Normal workers:* who have general knowledge to give correct answers, but with few occasional mistakes. (3) *Sloppy workers:* who have very little knowledge and thus often give wrong answers, but unintentionally. (4) *Uniform spammers:* who intentionally give the same answer for all their own questions. (5) *Random spammers:* who carelessly give the random answer for any question. To model these types of workers, we use two parameters: *sensitivity*—the proportion of actual positives that are correctly identified—and *specificity*—the proportion of negatives that are correctly identified. Following the statistical result in [11], we set randomly the sensitivity and specificity of each type of workers as follows. For experts, the range is $[0.9, 1]$. For normal workers, it falls into $[0.6, 0.9]$. For sloppy workers, the range $[0.1, 0.4]$ is selected. For random spammers, it varies from 0.4 to 0.6. Especially for uniform spammers, there are two regions: (i) $sensitivity \in [0.8, 1]$, $specificity \in [0, 0.2]$ and (ii) $sensitivity \in [0, 0.2]$, $specificity \in [0.8, 1]$.

Answer simulator This component generates worker answers for two types of questions. (1) *Binary-choice (yes/no):* in the literature, the *two-coin* model [19] is used to generate worker answers for each object. Each worker is associated with *sensitivity* and *specificity*, as described above. If the true label is *yes*, the worker answers *yes* with the probability *sensitivity*. If the true label is *no*, the worker answers *no* with the probability $specificity$. (2) *Multiple-choice:* since the two-coin model is only compatible with binary-choice questions, we adapt to multi-choice questions by using a reliability degree $r \in [0, 1]$ for each worker. Given a question with k choices, the probability of the worker answer being the same as and being different from the true label is r and $(1 - r)/k$, respectively. Note that the reliability degree is a special case of sensitivity and specificity; i.e. if $sensitivity = specificity$ then $sensitivity = specificity = r$. It is important to note that real objects can also be used instead of simulated ones. Users can plug in their own datasets under different formats. For benchmarking purposes, we also provide well-known datasets of the data integration domain in our website [4].

3.3 Evaluation Measures

We characterize the aggregation methods compared in the benchmark using four measures: *computation time, accuracy, robustness to spammers*, and *adaptivity to multi-labeling*. We describe the details for each of the measures in the sequel.

Computation time A simple metric for evaluating aggregation techniques is computation time. Various applications (e.g. CrowdSearch [23]) often have constraints on computing speed, or limitations in using server resources. As a result, the computation time becomes an important aspect, when we characterize an aggregation method. In our benchmark, all techniques are evaluated on the same standard. Specifically, we randomly generate the answer matrix M $(n \times k)$, while varying its size: $n = 10, 50, 100$ and $k = 10, 50, 100$. For each setting, we measure the average computation time—from when M is processed until aggregated values are computed—over 100 runs.

Accuracy Obviously, the most important aspect of an aggregation technique is its accuracy. It is straightforward how to measure that—accuracy is defined as the percentage of input objects that are correctly labeled:

$$accuracy = \frac{\#correctly\ labeled\ objects}{\#total\ objects} \tag{3}$$

The higher accuracy, the higher power of aggregation method. In experiments, we measure accuracy of each method while varying the number of answers per question and the number of questions per worker. In that, we find which algorithm requires least answers and which algorithm requires least workers to achieve the accuracy requirements.

Robustness to spammers In reality, spammers always exist in online community, especially crowdsourcing. Many experiments [21,4] in the literature showed that the proportion of spammers could be up to 40%. As a result, it is important for crowdsourcing applications to know how each aggregation technique performs when the worker answers are not trustworthy. In the benchmark, we studied the robustness to spammers by recording the accuracy, while varying the ratio of spammers. To this end, we artificially included spammers to the worker population, while applying different appearance ratio of spammers $p_{spam} = 5\%, 10\%, \ldots, 40\%$.

Adaptivity to multi-labeling In the literature, many applications are designed for multiple-choice questions. Therefore, it is important to know the adaptivity of aggregation techniques to this setting; i.e. which one is compatible and which one is not. Moreover, we would like to examine if there are significantly differences of their performance characteristics between the binary and the multiple setting. In the benchmark, we study the adaptivity to multi-labeling in terms of three aspects—computation time, accuracy, and robustness to spammers—while varying the number of possible labels equals to 2 or 4. Studying more than 4 labels is out of interest since these kinds of questions might be overwhelming to human workers.

4 Experimental Evaluation

We proceed to report results of applying the benchmark to the seven aggregation techniques presented in Section 2. The main goal of the experiments is not only to compare the aggregation performances, but also to analyze the effects of worker characteristics on the performance behavior. In order to compare them in a fair manner, we provide the key insights under a wide range of settings to verify their performance. All the experiments ran on an Intel Core i7 processor 2.8 GHz system with 4 GB of main memory.

4.1 Computation time

This experiment helps to choose the right techniques for a particular input size under time constraints. It takes server resources to process worker answers. In some real-time applications like CrowdSearch [23], final aggregations need to be returned within minutes or even seconds. As a result, quickly aggregating the worker answers is a key factor. Table 3 shows the computation time of each technique, averaged over 100 runs, when varying the input size from 10×10 to 100×100 (#questions $\times$ #workers).

Table 3: Average computation time (s) over 100 runs (the lower, the better)

Size of M [*]	**MD**	**HP**	**ELICE**	**EM**	**SLME**	**GLAD**	**ITER**
10×10	1	1	1	11	1	12	1
10×50	1	1	2	51	3	59	15
10×100	1	1	2	153	5	108	45
50×10	1	1	2	33	3	45	19
50×50	1	2	3	234	12	141	102
50×100	1	2	6	928	27	238	355
100×10	1	1	3	52	7	91	53
100×50	1	2	9	529	24	272	336
100×100	1	2	15	1591	46	473	915

[*] $n \times k$: n questions and k workers

MD, HP, and ELICE are clear winners on this concern. They by far outperform the others (their computation time is less than one minute with the size 100×100 of M). This result is straightforward to understand—these techniques are one-time computation and do not execute any expensive routines. In contrast, EM and ITER exhibits high computing time (with 100×100 input size, more than 15 min). While, SLME and GLAD exhibit satisfactory performance. In fact, we had expected slower performance from SLME and GLAD before having the results, since they take relatively sophisticated computation to update the worker expertise and the question difficulty in their iterations. However, the updating formulas in the EM steps of SLME and GLAD are less complex than EM and ITER. Briefly, this experiment suggests that MD, HP, and ELICE are fast enough for applications that prefer low response time, while the others should not probably be used for large inputs. Moreover, recall that iterative techniques (EM, SLME, GLAD, ITER) must re-compute the whole input when a new answer is received. We recommend using them for off-line analyses, when the answer set is fixed.

4.2 Accuracy

In order to reflect the accuracy of aggregation, in which the intuition behind this metric was explained in Section 3.3, our benchmark studies two dimensions of interest: *number*

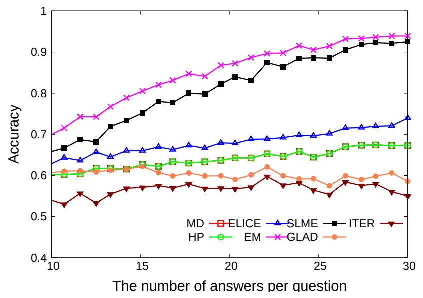

Fig. 3: Accuracy: effects of #apq (the higher, the better)

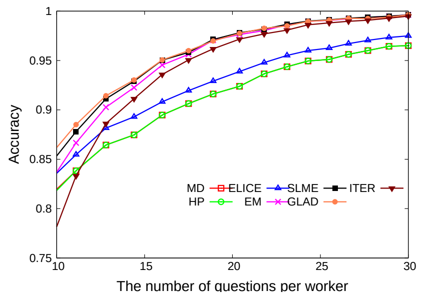

Fig. 4: Accuracy: effects of #qpw (the higher, the better)

of answers per question (#apq) and *number of questions per worker (#qpw)*. On one hand, #*apq* is the number of answers received for each question (i.e. #columns of input matrix *M*). When we have more answers from workers, the accuracy of aggregation increases since these answers will justify each other. This factor is important to study the trade-off between the cost (of paying workers) and the accuracy (of aggregated values). On the other hand, #*qpw* is the number of questions assigned for each worker (i.e. #rows of *M*). It should not be too large to ask a human worker or too small to assess his expertise. Some aggregation techniques (e.g. EM) consider the quality of answers of each worker to justify aggregated values. This factor is crucial for this purpose. Our benchmark will help potential users opt for appropriate values of these two factors.

The number of answers per question (#apq) The experiment was conducted with #*apw* varying from 10 to 30. The worker types follow the distribution as previously described in Section 3.2. Figure 3 illustrates the results obtained by computing the average over 100 runs. In general, the accuracy of all techniques increases with the increase of #*apw*. However, each algorithm behaves with the changes of #*apw* very differently.

Overall, the iterative techniques perform significantly better when the #*apw* is higher. This is because the same questions are answered by multiple workers (overlapping between workers). As a result, the answers of each worker can be justified by the answers of others through iterations. Among iterative techniques, EM is the best performer in this experiment. This is because EM captures the worker expertise by a confusion matrix, whereas the other iterative algorithms use a single parameter α. Subsequently, the characteristics of workers are more specific. Moreover, we can see that EM's accuracy is at least 25% higher than others in the end. In brief, we suggest using EM for high-accurate results, in case the computation time is not concerned.

The number of questions per worker (#qpw) In this experiment, we vary the number of questions per worker—hereby denoted as #*qpw*—from 10 to 30. The same worker population is used. In general, all techniques achieve higher accuracy when the #*qpw* increases. But there is no significant difference between them. When the #*qpw* > 20, the accuracy of all techniques is more than 90%. Figure 4 depicts the result.

At starting points (#*qpw* 10), ITER, HP, and MD are the worst techniques. For MD, this is because the majority effect: not enough trustworthy answers to dominate untrustworthy ones. For HP, this effect is more severe since truthful workers have too few correct answers to pass trapping questions. For ITER, it is due to the lacks of initial

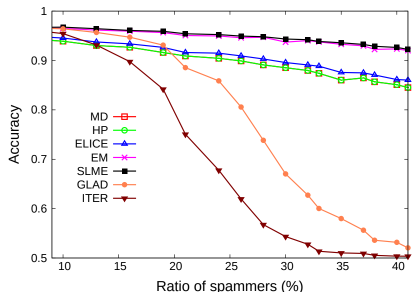

Fig. 5: Effects of uniform spammers (the higher, the better)

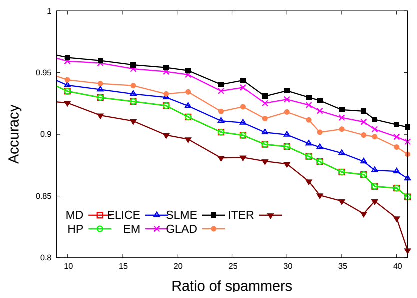

Fig. 6: Effects of random spammers (the higher, the better)

information. However, as the #*qpw* increases, the difference among all techniques is reduced (less than 0.05 with 30 #*qpw*). In addition, each of them has a ”convergent point”: continue increasing #*qpw* above this point will not improve the accuracy significantly (e.g. EM achieves 95% accuracy at #*qpw* ≈ 18, doubling #*qpw* only increases the accuracy up to 5% more). Another interesting observation is that iterative techniques are slightly better non-iterative ones: the difference is only 5% when the #*qpw* reaches to 30. This can be explained by the fact that in iterative techniques, worker answers are more refined by multiple of computational rounds.

4.3 Robustness to Spammers

In this experiment, we will increase the ratio of spammers to study its effects on accuracy. First, we remove sloppy workers from the crowd due to their lacks of knowledge, which generates many wrong answers in the input. By this way, we can see a clear effect of spammers. The spammer ratio is varied from 5% to 40%. Based on previous results, we fix the number of answers per object to 20 because this gives a high starting point of accuracy. Figure 5 and 6 illustrate the effects of uniform spammers and random spammers on accuracy, respectively. In general, the accuracy of all techniques decreases when the spammer ratio is higher. But their behaviors are significantly different.

Uniform spammers The effects of uniform spammers are presented in Figure 5. An interesting observation is that ITER and GLAD are the worst in this setting. At the starting point (5% spammers), their accuracies are already lower than the others'—about 0.6 and 0.75 respectively. After the spammer ratio rises to 25%, ITER and GLAD drops rapidly to nearly 0.5. By the time more than 25% spammers, their behaviors are like random (accuracy converges to 0.5). This observation could be explained by their underlying models. First, ITER's algorithm depends on the entropy of a worker's answers. Since uniform spammers always give identical answers, the uncertainty of the input is high and it ends up with a poor accuracy. Second, GLAD is not able to identify and prioritize truthful workers (i.e. all workers are initially weighted as equal), resulting in a random accuracy in the end. In brief, ITER and GLAD are very sensitive to the spammers. Another key finding is that among the five remaining techniques, we can observe two distinct groups. The first group consists of EM and SLME, which are the better than the second group including MP, HP, and ELICE. However, the difference between them is not significant (less than 0.1 with 40% uniform spammers).

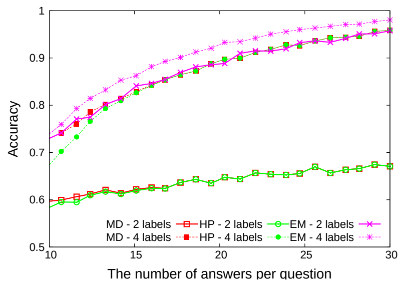

Fig. 7: Multi-label effect on accuracy

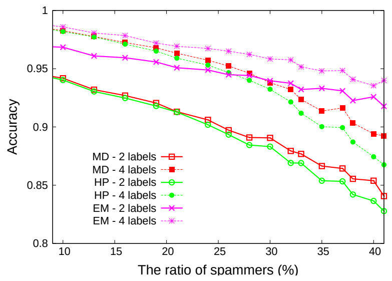

Fig. 8: Multi-label effect on spammer robustness

Random spammers The effects of random spammers are depicted in Figure 6. Similar to random spammers case, most of the techniques are robust to random spammers—their accuracy decreases up to 10% at the end. However, their accuracies are better in this case. This is reasonable because the answers of spammers are different from each other, which cannot dominate the answers of other workers. Another noticeable observation is that ITER and GLAD perform better than before. For ITER, although starting with high accuracy, its accuracy reduces more than 15% at the end. This could be explained by the same way. Since the answers are random but different, the entropy value is lower, resulting in a higher accuracy. For GLAD, it uses the answers of other workers to justify the spammers', rendering the similar performance like EM and SLME.

4.4 Adaptivity to Multi-labeling

In this experiment, we study the adaptivity to multi-labeling of aggregation techniques. Only three techniques—MD, HP, and EM—are retained while the others fail to adapt this setting. SLME models worker expertise by *sensitivity* and *specificity*, which are applicable for binary question only. Regarding ITER and ELICE, their original papers indicate that they are only applied for binary questions. Besides, they use the sign (positive or negative) of aggregated value to classify object. Regarding GLAD, we checked the source code and confirmed that it was implemented for only binary-choice questions. Similar to previous experiments, we proceed to report the performance characteristics of applicable techniques (MP, HP, and EM) in three aspects below.

Computation time In general, MD and HP are not affected by #labels—their computation time keeps unchanged when #labels increases. This is because the complexity of majority rule only depends on the number of answers per object (the label with highest number of answers wins). For EM, its completion time increases a little bit since it uses a confusion matrix to capture worker expertise. The size of this matrix is $n \times n$, where n is the number of labels. However, as n only up to 4, this shows no significant difference. Therefore, the results of computation time are omitted due to page limits.

Accuracy Using the same worker distribution of Section 4.2, we measure the accuracy against different numbers of answers per question. The result for different numbers of questions per worker is omitted due to similar findings. Figure 7 illustrates the result. In general, with more labels, the accuracy is better. EM is still the winner in both cases—2 labels and 4 labels. However, the superiority of EM in comparison with MD and HP is reduced when the number of labels increase. For example, the difference is between 0.12 and 0.28 with 2 labels, whereas this difference is less than 0.02 with 4 labels. This is, in fact, reasonable. The incorrect answers are now distributed among several choices, which is unlikely to dominate the majority of the true answers.

Sensitive to spammers Like previous experiments in Section 4.3, we increase the spammer ratio to study the accuracy reduction. Results are presented in Figure 8. In both 2-label and 4-label settings, EM is more robust to spammers than MD and HP. Surprisingly, MD and HP become better with 4 labels. This can be explained by the majority property: answers given by spammers no longer dominate those of other workers. Since there are more choices, it is unlikely that spammers give the same answer together.

5 Summary and Conclusions

This paper presented a thorough evaluation and comparison of answer aggregation techniques widely used in crowdsourcing. We offered an overview of two major classes (non-iterative and iterative) of aggregation techniques, while discussing about the characteristics of their underlying probabilistic models. We then introduced the component-based benchmarking framework, in which a new aggregation technique as well as a new measurement can be easily plugged. During the framework development, we made the best effort to re-implement and integrate the most representative aggregation techniques, and evaluated them in a fair manner. We also analyzed various performance factors for each technique, including *computation time, accuracy, robustness to spammers,* and *adaptivity to multi-labeling*. The crowdsourcing process is simulated by letting five different types of workers answer binary or multiple-choice questions.

We here summarize our principal findings as a set of recommendations for how to select a well-suited aggregation technique on particular application scenarios:

- Overall, EM and SLME achieve highest accuracy and work robustly against spammers. In particular, they outperform the others when #answers per question is high. Regarding #questions per worker, there are two runner-ups (GLAD and ITER).
- If the crowd contains many spammers ($\geq$ 30%), we suggest using SLME or EM. Interestingly, the performance of non-iterative techniques (MD, HP, ELICE) is not significantly lower than SLME and EM. If accuracy is not highly required, they are best-suited for applications that require fast computation. In contrast, we strongly suggest not using GLAD and ITER since they are most sensitive to spammers.
- Only MD, HP, and EM can adapt to multi-labeling. For binary labeling, EM is the winner. In case of 4 labels, MD and HP are also appropriate choices since the difference between them and EM is not distinguishable.
- For applications that require fast computation, MD and HP are the winners. Oppositely, we strongly suggest not using iterative techniques. Not only is their computation time much higher than the non-iterative techniques, but also they require to re-compute the whole answer set upon the new arrival of worker answers.

category	winner	2nd best	worst
computation time	**MD**	HP	EM
accuracy	**EM**	SLME	HP
robustness to spammers	**SLME**	EM	ITER
adaptivity to multi-labeling *	**EM**	MD	HP

* other techniques (ELICE, SLME, GLAD, ITER) only work with binary labeling due to their implementation limitation

As a concluding remark, we recommend potential applications to use our benchmarking framework as a tool to find out the best-suited aggregation technique accordingly, since there is no absolute winner that outperforms the others in every case. As the source codes as well as datasets used in the benchmark are publicly available, we expect that the experimental results presented in this paper will be refined and improved by the research community, in particular when more data become available, more experiments are performed, and more techniques are integrated into the framework in the future.

References

1. von Ahn, L. et al.: Labeling images with a computer game. In: CHI (2004)
2. von Ahn, L. et al.: recaptcha: Human-based character recognition via web security measures. Science (2008)
3. Demartini, G. et al.: Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: WWW (2012)
4. Difallah, D.E. et al.: Mechanical cheat: Spamming schemes and adversarial techniques on crowdsourcing platforms. In: CrowdSearch (2012)
5. Doan, A. et al.: Crowdsourcing systems on the world-wide web. CACM (2011)
6. Hosmer, D.W. et al.: Applied logistic regression. Wiley-Interscience Publication (2000)
7. Ipeirotis, P.G. et al.: Quality management on amazon mechanical turk. In: HCOMP (2010)
8. Kamar, E. et al.: Combining human and machine intelligence in large-scale crowdsourcing. In: AAMAS (2012)
9. Kamar, E. et al.: Incentives for truthful reporting in crowdsourcing. In: AAMAS (2012)
10. Karger, D. et al.: Iterative learning for reliable crowdsourcing systems. In: NIPS (2011)
11. Kazai, G. et al.: Worker types and personality traits in crowdsourcing relevance labels. In: CIKM (2011)
12. Khattak, F. et al.: Quality Control of Crowd Labeling through Expert Evaluation. In: NIPS (2011)
13. Kuncheva, L. et al.: Limits on the majority vote accuracy in classifier fusion. Pattern Anal. Appl. (2003)
14. Law, E. et al.: Human Computation. Morgan & Claypool Publishers (2011)
15. Lee, K. et al.: The social honeypot project: protecting online communities from spammers. In: WWW (2010)
16. Mason, W. et al.: Conducting behavioral research on amazon mechanical turk. BRM (2012)
17. Quinn, A.J. et al.: Human computation: a survey and taxonomy of a growing field. In: CHI (2011)
18. Raykar, V. et al.: Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In: ICML (2009)
19. Raykar, V.C. et al.: Learning from crowds. Mach. Learn. Res. (2010)
20. Ross, J. et al.: Who are the crowdworkers?: shifting demographics in mechanical turk. In: CHI (2010)
21. Vuurens, J. et al.: How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In: CIR (2011)
22. Whitehill, J. et al.: Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: NIPS (2009)
23. Yan, T. et al.: CrowdSearch: exploiting crowds for accurate real-time image search on mobile phones. In: MobiSys (2010)