Graph Mining Meets Crowdsourcing: Extracting Experts for Answer Aggregation

Yasushi Kawase[1,3] , **Yuko Kuroki**[2,3] , **Atsushi Miyauchi**[3]

[1]Tokyo Institute of Technology
[2]The University of Tokyo
[3]RIKEN AIP

kawase.y.ab@m.titech.ac.jp, ykuroki@ms.k.u-tokyo.ac.jp, atsushi.miyauchi.hv@riken.jp

Abstract

Aggregating responses from crowd workers is a fundamental task in the process of crowdsourcing. In cases where a few experts are overwhelmed by a large number of non-experts, most answer aggregation algorithms such as the majority voting fail to identify the correct answers. Therefore, it is crucial to extract reliable experts from the crowd workers. In this study, we introduce the notion of *expert core*, which is a set of workers that is very unlikely to contain a non-expert. We design a graph-mining-based efficient algorithm that exactly computes the expert core. To answer the aggregation task, we propose two types of algorithms. The first one incorporates the expert core into existing answer aggregation algorithms such as the majority voting, whereas the second one utilizes information provided by the expert core extraction algorithm pertaining to the reliability of workers. We then give a theoretical justification for the first type of algorithm. Computational experiments using synthetic and real-world datasets demonstrate that our proposed answer aggregation algorithms outperform state-of-the-art algorithms.

1 Introduction

Crowdsourcing, which has become popular in recent years, is a process that requires completion of specific tasks by crowd workers. In crowdsourced single-answer multiple-choice questions, workers are asked to select one answer out of multiple candidates for each given question. Such a scheme is used, for instance, in annotating named entities for microblogs [Finin *et al.*, 2010], sentiment classification on political blogs [Hsueh *et al.*, 2009], and image tagging [Lin *et al.*, 2015].

For the purpose of quality control, crowdsourcing systems usually assign multiple workers to the same questions and then aggregate their answers using some rule or algorithm. A critical fact here is that the workers often have different levels of expertise, skills, and motivation; some workers may guess the correct answers with their rich knowledge, while others may answer almost randomly to get a reward without any effort. Let us call the workers of the former and latter type *experts* and *non-experts*, respectively.

	$\mathcal{W}\backslash\mathcal{Q}$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
	correct answer	D	C	D	E	B	C	A	E
Experts	w_1	D	C	D	E	B	C	A	E
	w_2	D	C	B	E	B	C	A	C
	w_3	D	C	D	D	B	C	A	E
Non-experts	w_4	C	B	A	A	E	C	D	B
	w_5	A	B	E	A	B	E	E	C
	w_6	C	A	B	E	B	B	A	C

Table 1: A small example of crowdsourced single-answer multiple-choice questions.

When only a few experts are overwhelmed by a large number of non-experts, the most intuitive answer aggregation rule, *majority voting*, often fails to acquire the correct answers. Consider a small example in Table 1. There are six workers $w_1, \ldots, w_6$ assigned to eight questions $q_1, \ldots, q_8$. For each question, the candidate answers are A, B, C, D, and E. Among these workers, w_1, w_2, and w_3 are experts who almost always give the correct answers, while the others, w_4, w_5, and w_6, are non-experts who give random answers. Let us apply the majority voting to their answers. The majority answer for q_1 is D, which is the correct answer. However, the majority answer for q_8 is C, which is not the correct answer. In addition, we need tie-breaking for q_3 because B and D get the same number of votes. It is very likely that as the fraction of non-experts increases, the quality of the majority voting answers deteriorates.

Various answer aggregation algorithms exist in addition to the majority voting (see Related Work). However, most of these algorithms implicitly strengthen the majority answers and therefore fail to provide the true answers when the majority answers are incorrect. To overcome this issue, Li et al. [2017] recently proposed a sophisticated answer aggregation algorithm for such a hard situation. More specifically, they introduced the notion of *hyper questions*, each of which is a set of single questions. Their algorithm applies the majority voting (or other existing answer aggregation algorithms) for the hyper questions and then decodes the results to votes on individual questions. Finally, it applies the majority voting again to obtain the final answers. The results of their experiments demonstrate that their algorithm outperforms existing algorithms.

1.1 Our Contribution

In this study, we further investigate the above-mentioned hard situation. Our contribution can be summarized as follows:

1. We introduce a graph-mining-based efficient algorithm that accurately extracts the set of experts;
2. We propose two types of answer aggregation algorithms based on the above experts extraction algorithm;
3. We provide a theoretical justification of our proposed algorithms;
4. We conduct thorough computational experiments using synthetic and real-world datasets to evaluate the performance of our proposed algorithms.

First result. To design a powerful answer aggregation algorithm for the above hard situation, it is crucial to extract reliable experts from the crowd workers. In the example above, if we recognize that w_1, w_2, and w_3 are experts, we can obtain the correct answers for all questions by simply applying the majority voting to the answers of the experts. The fundamental observation we use is as follows: as the experts almost always give the correct answers, each pair of experts frequently gives the same answer to a question. Let us now consider constructing an edge-weighted complete undirected graph in which each vertex corresponds to a worker and each edge weight represents the agreement rate of the answers of two workers. From the above observation, it is very likely that there is a dense component consisting of the experts, which we call the *expert core*. Note that the formal definition of the expert core will be given in Section 2. Figure 1 depicts an edge-weighted graph constructed from the example in Table 1. As can be seen, experts w_1, w_2, and w_3 form a dense component. Although, in this example, we simply set the edge weight to the number of same answers of two workers, we will use more suitable values in our algorithm. To extract the expert core, we use a well-known dense subgraph extraction algorithm called the *peeling algorithm* [Asahiro *et al.*, 2000], which removes the most unreliable worker one by one.

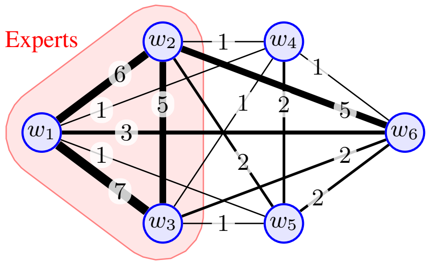

Figure 1: An edge-weighted graph constructed from the example in Table 1.

Second result. Based on the expert core extraction algorithm, we propose two types of answer aggregation algorithms. The first one incorporates the expert core into existing answer aggregation algorithms such as the majority voting. Indeed, once we extract the expert core, we can apply existing algorithms to the answers of the workers in the expert core. The second one utilizes the information pertaining to the reliability of workers provided by the expert core extraction algorithm, which is quite effective when the task is very hard.

Third result. We first demonstrate that the expert core is very unlikely to contain a non-expert if the number of questions is sufficiently large (Theorem 2). We then prove that the majority voting is asymptotically correct if there are only experts (Theorem 3), but not reliable if there are a large number of non-experts (Theorem 4). Theorems 2 and 3 provide a theoretical justification for the first type of our algorithm. In fact, combining these two theorems, we see that if the number of questions and the number of workers in the expert core are sufficiently large, our proposed algorithm (i.e., the majority voting among the workers in the expert core) gives the correct answer with high probability. On the other hand, Theorem 4 provides the limitation of the majority voting, which implies that it is quite important to exclude non-experts when we use the majority voting.

Fourth result. We conduct thorough computational experiments using synthetic and real-world datasets to evaluate our proposed algorithms. To simulate the hard situation in this study, we use six datasets recently collected by Li et al. [2017] as real-world datasets, all of which have difficult heterogeneous-answer questions that require specialized knowledge. We demonstrate that the expert core counterparts of existing answer aggregation algorithms perform much better than their original versions. Furthermore, we show that our novel algorithm based on the information of the reliability of workers outperforms the other algorithms particularly when the task is quite hard.

1.2 Related Work

To date, a large body of work has been devoted to developing algorithms that estimate the quality of workers [Dawid and Skene, 1979; Whitehill *et al.*, 2009; Welinder *et al.*, 2010; Raykar *et al.*, 2010; Karger *et al.*, 2011; Wauthier and Jordan, 2011; Bachrach *et al.*, 2012; Demartini *et al.*, 2012; Kim and Ghahramani, 2012; Liu *et al.*, 2012a; Liu *et al.*, 2012b; Zhou *et al.*, 2012; Aydin *et al.*, 2014; Li *et al.*, 2014b; Venanzi *et al.*, 2014; Ma *et al.*, 2015; Li *et al.*, 2017; Zheng *et al.*, 2017; Li *et al.*, 2018]. In most existing work, the quality of workers and correct answers are estimated by an iterative approach comprising the following two steps: (i) infer the correct answers based on the quality of workers estimated and (ii) estimate the quality of workers based on the correct answers inferred. For example, Whitehill et al. [2009] modeled each worker's ability and each task's difficulty, and then designed a probabilistic approach, which they called *GLAD* (Generative model of Labels, Abilities, and Difficulties).

Finding a dense component in a graph is a fundamental and well-studied task in graph mining. A typical application of dense subgraph extraction is to identify components that have some special role or possess important functions in the underlying system represented by a graph.

Examples include communities or spam link farms extraction in Web graphs [Dourisboure *et al.*, 2007; Gibson *et al.*, 2005], identification of molecular complexes in protein–protein interaction graphs [Bader and Hogue, 2003], and expert team formation in collaboration graphs [Tsourakakis *et al.*, 2013], The peeling algorithm [Asahiro *et al.*, 2000] is known to be effective in various optimization problems for dense subgraph extraction (e.g., the densest subgraph problem and its variations [Andersen and Chellapilla, 2009; Charikar, 2000; Kawase and Miyauchi, 2018; Miyauchi and Kakimura, 2018; Khuller and Saha, 2009] as well as other related problems [Tsourakakis *et al.*, 2013; Miyauchi and Kawase, 2015]).

2 Model

An instance of our problem is a tuple $(\mathcal{W}, \mathcal{Q}, \mathcal{C}, \mathcal{L})$, where each component is defined as follows: There is a finite set of workers $\mathcal{W} = \{w_1, \dots, w_n\}$ and a finite set of questions $\mathcal{Q} = \{q_1, \dots, q_m\}$. Each question q has a set of candidate answers $\mathcal{C}_q = \{c_1^q, \dots, c_s^q\}$ $(s \geq 2)$. Suppose that, for each question q, worker w answers $l_{wq} \in \mathcal{C}_q$ and let $\mathcal{L} = (l_{wq})_{w \in \mathcal{W},\, q \in \mathcal{Q}}$. Our task is to estimate the unknown correct answers to the questions.

Suppose that, among the workers $\mathcal{W} = \{w_1, \dots, w_n\}$, there are n_{ex} experts $\mathcal{E}$ $(\subseteq \mathcal{W})$ who give the correct answer with probability p_{ex} $(> 1/s)$, and the other n_{non} $(= n - n_{\mathrm{ex}})$ workers are non-experts who give an answer independently and uniformly at random. If an expert makes a mistake, she selects a wrong answer independently and uniformly at random. Thus, for a question $q \in \mathcal{Q}$ with a correct answer $a_q \in \mathcal{C}_q$ and an answer $c \in \mathcal{C}_q$, it holds that

$$\Pr[l_{wq} = c] = \begin{cases} p_{\mathrm{ex}} & (c = a_q,\ w \in \mathcal{E}), \\ \frac{1-p_{\mathrm{ex}}}{s-1} & (c \neq a_q,\ w \in \mathcal{E}), \\ 1/s & (w \in \mathcal{W} \setminus \mathcal{E}). \end{cases}$$

It should be noted that, by showing the candidate answers in random order for each worker, we can handle some biases (e.g., some non-expert workers always choose the first candidate of answers) using this model.

Let us consider the probability that a pair of workers $u, v \in \mathcal{W}$ $(u \neq v)$ gives the same answer for a question $q \in \mathcal{Q}$. If at least one of u and v is a non-expert, then we have $\Pr[l_{uq} = l_{vq}] = 1/s$. On the other hand, if both workers are experts, then $\Pr[l_{uq} = l_{vq}] = \frac{(p_{\mathrm{ex}} - 1/s)^2}{1 - 1/s} + \frac{1}{s}$, which is strictly larger than $1/s$.

For each pair of workers $u, v \in \mathcal{W}$, let $\tau(u, v)$ be the number of questions such that u and v give the same answer, that is, $\tau(u, v) = |\{q \in \mathcal{Q} : l_{uq} = l_{vq}\}|$. Here, if $\Pr[l_{uq} = l_{vq}] = p$, then u and v give the same answers for at least $\tau(u, v)$ questions with probability $\sum_{i=\tau(u,v)}^{m} \binom{m}{i} p^i (1-p)^{m-i}$.

For a given $p \in (0, 1)$, a subset of workers $W \subseteq \mathcal{W}$ is called a *θ-expert set* with respect to p if

$$\prod_{v \in W \setminus \{u\}} \sum_{i=\tau(u,v)}^{m} \binom{m}{i} p^i (1-p)^{m-i} \leq \theta$$

holds for all $u \in W$. Intuitively, a θ-expert set with small θ is a set of workers that is very unlikely to contain a non-expert. Let $\theta(W)$ be the minimum threshold such that W is a θ-expert set, that is,

$$\theta(W) = \max_{u \in W} \prod_{v \in W \setminus \{u\}} \sum_{i=\tau(u,v)}^{m} \binom{m}{i} p^i (1-p)^{m-i}.$$

Then, we employ $W \subseteq \mathcal{W}$ that minimizes $\theta(W)$ as the estimated set of experts. We refer to such a set as the *expert core* (with respect to p). As will be shown in Theorem 2, the expert core is very unlikely to contain a non-expert if the number of questions is sufficiently large.

Algorithm 1: Peeling algorithm

```
Input: Edge-weighted graph $G = (V, E, \omega)$
Output: $S \subseteq V$ ($k$-core with maximum $k$)
1 $S_{|V|} \leftarrow V$;
2 for $i \leftarrow |V|, \dots, 2$ do
3     Find $v_i \in \mathrm{argmin}_{v \in S_i} d_{S_i}(v)$ and $S_{i-1} \leftarrow S_i \setminus \{v_i\}$;
4 return $S_i \in \{S_1, \dots, S_{|V|}\}$ that maximizes $d_{S_i}(v_i)$;
```

3 Algorithms

In this section, we design an algorithm to compute the expert core, and then propose two types of answer aggregation algorithms. Our expert core extraction algorithm first constructs an edge-weighted complete undirected graph that represents the similarity of the workers in terms of their answers. Then, it extracts a dense component in the graph using the peeling algorithm [Asahiro *et al.*, 2000].

3.1 Peeling Algorithm

We first revisit the peeling algorithm. The algorithm iteratively removes a vertex with the minimum weighted degree in the current graph until we are left with only one vertex. Let $G = (V, E, \omega)$ be an edge-weighted graph. For $S \subseteq V$ and $v \in S$, let $d_S(v)$ denote the weighted degree of v in the induced subgraph $G[S]$, that is, $d_S(v) = \sum_{e=\{u,v\} \in E:\, u \in S} \omega(e)$. Then, the procedure can be summarized in Algorithm 1. Note that the algorithm here returns $S_i \in \{S_1, \dots, S_{|V|}\}$ that maximizes $d_{S_i}(v_i)$, although there are other variants depending on the problem at hand [Charikar, 2000; Kawase and Miyauchi, 2018; Miyauchi and Kawase, 2015; Tsourakakis *et al.*, 2013]. Algorithm 1 can be implemented to run in $O(|E| + |V| \log |V|)$ time.

Algorithm 1 indeed produces the *k-core decomposition* of graphs. For $G = (V, E, \omega)$ and a positive real k, a subset $S \subseteq V$ is called a *k-core* if S is a maximal subset in which every vertex $v \in S$ has a weighted degree of at least k in the induced subgraph $G[S]$. Note that a k-core is unique for a fixed k. The k-core decomposition reveals the hierarchical structure of k-cores in a graph and is particularly focused on finding the k-core with maximum k. Algorithm 1 is suitable for this scenario; in fact, it is evident that the algorithm returns the k-core with maximum k.

3.2 Expert Core Extraction Algorithm

Here, we present an algorithm to compute the expert core. In particular, we explain the construction of an edge-weighted graph, which represents the similarity of the workers in terms of their answers.

In our algorithm, we set p to the average agreement probability, that is,

$$p = \frac{1}{m}\sum_{q\in\mathcal{Q}}\left|\left\{\{u,v\}\in\binom{\mathcal{W}}{2} : l_{uq}=l_{vq}\right\}\right|/\binom{n}{2},$$

where $\binom{\mathcal{W}}{2} = \left\{\{u,v\} : u,v\in\mathcal{W},\ u\neq v\right\}$, and extract the expert core with respect to this p via the peeling algorithm. We construct a complete graph $(\mathcal{W}, \binom{\mathcal{W}}{2})$ with weight $\gamma(u,v) = -\log\sum_{i=\tau(u,v)}^{m}\binom{m}{i}p^i(1-p)^{m-i}$, where recall that $\tau(u,v) = |\{q\in\mathcal{Q} : l_{uq}=l_{vq}\}|$. Then, we compute the k-core with maximum k for the edge-weighted graph $(\mathcal{W}, \binom{\mathcal{W}}{2}, \gamma)$ using Algorithm 1. As a result, we can obtain a set of workers $W\subseteq\mathcal{W}$ such that the following value is maximized:

$$\begin{aligned}&\min_{u\in W}\sum_{v\in W\setminus\{u\}}\gamma(u,v)\\ &= \min_{u\in W}\left(-\log\prod_{v\in W\setminus\{u\}}\sum_{i=\tau(u,v)}^{m}\binom{m}{i}p^i(1-p)^{m-i}\right)\\ &= -\log\theta(W).\end{aligned}$$

As $-\log x$ is monotone decreasing, the obtained set minimizes $\theta(W)$ and hence is the expert core.

Note that we can construct the edge-weighted graph $(\mathcal{W}, \binom{\mathcal{W}}{2}, \gamma)$ in $O(n^2m)$ time and Algorithm 1 for the graph runs in $O(\binom{n}{2} + n\log n) = O(n^2)$ time. Thus, we have the following theorem.

Theorem 1. *The expert core can be found in $O(n^2m)$ time.*

3.3 Answer Aggregation Algorithms

Once we extract the expert core, we can apply existing answer aggregation algorithms (e.g., the majority voting) to the answers of the workers in the expert core, which implies the expert core counterparts of the existing algorithms. In addition, we propose a novel answer aggregation algorithm. The peeling algorithm produces the ordering that represents the reliability of the workers. With this ordering, we can obtain a majority-voting-based algorithm as follows. At the beginning, we obtain the ordering of workers using the peeling algorithm for $(\mathcal{W}, \binom{\mathcal{W}}{2}, \gamma)$. Then we pick the most reliable pair of workers, that is, the pair left until the second last round of the peeling algorithm. For each question, if the two workers select the same answer, then we employ this answer. If the two workers select different answers, then we add the next most reliable worker one by one according to the ordering until two of them give the same answer. Our algorithm, which we call Top-2, is summarized in Algorithm 2. Note that the algorithm runs in $O(n^2m)$ time, which is the same as that of the expert core extraction algorithm.

Algorithm 2: Top-2

```
  Input: Worker set $\mathcal{W}$; Question set $\mathcal{Q}$; Answer set $\mathcal{L}$
  Output: Estimated true answers $(z_q)_{q\in\mathcal{Q}}$
1 Calculate the average agreement probability $p$;
2 Construct the edge-weighted graph $(\mathcal{W}, \binom{\mathcal{W}}{2}, \gamma)$;
3 Let $S_1, \dots, S_n$ be the sets computed in Algorithm 1;
4 for $q \in \mathcal{Q}$ do
5     for $i \leftarrow 2, \dots, n$ do
6         if $\exists c^* \in \mathcal{C}_q$ s.t. $|\{w \in W_i : l_{wq} = c^*\}| = 2$
            then $z_q \leftarrow c^*$ and break ;
7         else if $i = n$ then $z_q \leftarrow l_{wq}$, where $\{w\} = S_1$ ;
8 return $(z_q)_{q\in\mathcal{Q}}$;
```

4 Theoretical Results

In this section, we provide a theoretical justification for the first type of our proposed algorithm. Owing to space limitations, we omit the proofs, all of which can be found in the full version [Kawase *et al.*, 2019]. Suppose that each worker gives answers according to the model described in Section 2.

The following theorem states that the expert core does not contain any non-expert with high probability if the number of questions is sufficiently large.

Theorem 2. *Let $W^*\subseteq\mathcal{W}$ be the expert core. If $n_{\rm ex}\geq 2$ and $m\geq\frac{2n^4\log\frac{n^2}{\epsilon}}{(p_{\rm ex}-1/s)^4}$ for $\epsilon>0$, then we have $\Pr[W^*\subseteq\mathcal{E}]\geq 1-\epsilon$.*

The next theorem states that the majority voting gives the correct answer with high probability if the number of workers is sufficiently large and all of them are experts. Let $\mathtt{MV}_q$ be the output of the majority voting for question $q\in\mathcal{Q}$.

Theorem 3. *If $n = n_{\rm ex}\geq\frac{2\log\frac{2s}{\epsilon}}{(p_{\rm ex}-1/s)^2}$ for $\epsilon>0$, then we have $\Pr[\mathtt{MV}_q = a_q]\geq 1-\epsilon$ ($\forall q\in\mathcal{Q}$).*

Finally, the following theorem states that, the majority voting does not provide the correct answer with high probability if the number of non-experts is much larger than the number of experts.

Theorem 4. *If $n_{\rm non}\geq\frac{n_{\rm ex}^2}{2\pi\epsilon^2}$ for $\epsilon>0$, then we have $\Pr[\mathtt{MV}_q = a_q]\leq\frac{1}{2}+\epsilon$ ($\forall q\in\mathcal{Q}$).*

It should be noted that when the number of candidate answers is 2, even the random choice gives the correct answer with probability $1/2$. Thus, the above theorem indicates that the majority voting is no better than the random choice if the number of non-experts is much larger than the number of experts.

5 Experiments

In this section, we report the results of computational experiments. The objective of our experiments is to examine the performance of our proposed algorithms from various aspects using both synthetic and real-world datasets. The main component of our experiments comprises comparing our proposed answer aggregation algorithms with the following three existing algorithms: MV (the majority voting), GLAD [Whitehill *et al.*, 2009], and Hyper-MV [Li *et al.*,

Dataset	m	n	s	BEST
Chinese	24	50	5	0.79
English	30	63	5	0.70
IT	25	36	4	0.84
Medicine	36	45	4	0.92
Pokémon	20	55	6	1.00
Science	20	111	5	0.85

Table 2: Real datasets used in our experiments. The last column gives the best accuracy rate (i.e., the fraction of the number of correct answers) among the workers.

2017]. As our proposed algorithms, we employ the expert core counterparts of the above three algorithms, which we denote by Ex-MV, Ex-GLAD, and Ex-Hyper-MV, respectively, in addition to Top-2.

5.1 Datasets

As for synthetic datasets, we generate a variety of instances using the model described in Section 2. Recall the following five parameters: number of workers $n = |\mathcal{W}|$, number of questions $m = |\mathcal{Q}|$, number of candidate answers (for each question) s, number of ground-truth experts n_{ex}, and the probability p_{ex} that an expert gives the correct answer. Throughout the experiments, we set $s = 5$.

Table 2 summarizes six datasets that we use as real-world datasets. They were recently collected by Li et al. [2017] using Lancers, a commercial crowdsourcing platform in Japan. Consistent with the hard situation being addressed here, these datasets have difficult heterogeneous-answer questions that require specialized knowledge. In fact, as shown later, the majority voting performs poorly on these datasets. Note that the classical datasets used in other previous work (e.g., *Bluebird* and *Dog* in image tagging [Welinder *et al.*, 2010; Zhou *et al.*, 2012], *Web* in Web search relevance judging [Zhou *et al.*, 2012], *Price* in product price estimation [Liu *et al.*, 2013], and *RTE* and *Temp* in textual entailment recognition [Snow *et al.*, 2008]) usually have relatively easy homogeneous-answer questions, which are not within the scope of this study. During the collection of the above six datasets, all workers were asked to answer all questions. This may not be the usual assumption in crowdsourcing but is effective in the current hard situation. Indeed, if we can identify reliable experts by asking all workers to answer a small number of questions at an early stage, then it is possible to ask only the identified experts to answer to the remaining questions, which may reduce the overall cost. This scenario has been studied in previous work [Li *et al.*, 2017; Li *et al.*, 2014a].

5.2 Experts Extraction

To evaluate the performance of our expert core extraction algorithm in terms of extracting reliable experts, we conducted simulations using synthetic datasets. We set the number of workers as n to 20 and the number of experts as n_{ex} to 4. In these settings, we generated two types of instances to simulate different situations. In the first type, to investigate the effect of the number of questions, we vary the number of questions m from 5 to 50 (in increments of 5) under a fixed value of $p_{\text{ex}} = 0.8$. In the second type, to investigate the effect of the expertise of workers, we vary the probability p_{ex} from 0.5 to 1.0 (in increments of 0.05) under a fixed value of $m = 25$.

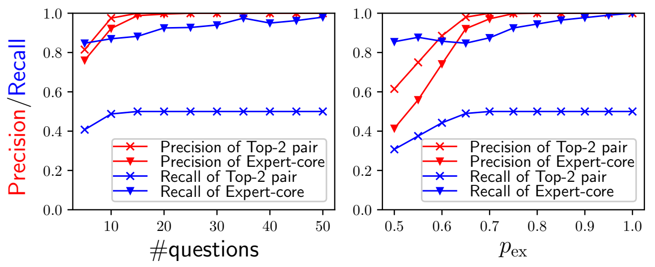

Figure 2: Results of experts extraction for synthetic datasets, where $n = 20$ and $n_{\text{ex}} = 4$.

As the sets of workers extracted by our peeling algorithm, we employ the Top-2 pair (i.e., a pair of workers left until the second last round of the peeling algorithm) and the expert core. To evaluate the quality of the set of workers at hand, we adopt the following two measures: the *precision* (number of ground-truth experts in the set divided by the size of the set) and the *recall* (number of ground-truth experts in the set divided by the number of all ground-truth experts).

The results are shown in Figure 2. Each point corresponds to an average over 100 data realizations. As can be seen, the precision of the expert core becomes better as the number of questions m or the probability p_{ex} increases. Moreover, the expert core is robust in terms of the recall; in fact, even when p_{ex} is small (i.e., $p_{\text{ex}} \leq 0.7$), the expert core achieves an average recall of 0.8. The Top-2 pair achieves better precision in all parameter settings.

5.3 Ordering Defined by Peeling

The objective here is to demonstrate that the ordering of workers defined by the peeling algorithm for $(\mathcal{W}, \binom{\mathcal{W}}{2}, \gamma)$ reflects the accuracy rate of the workers. To this end, we perform the peeling algorithm for the real-world datasets. The results are shown in Figure 3. In the subfigures, the horizontal axis represents the ordering of workers defined by the peeling algorithm; specifically, the peeling algorithm has removed the workers from right to left. The vertical axis represents the accuracy rate of the workers. The bars corresponding to the workers contained in the expert core are colored black, while those corresponding to the other workers are colored gray.

As can be seen, workers with smaller order tend to have higher accuracy rates, In addition, the expert core contains almost all workers with significantly high accuracy rates. which demonstrates the reliability of our peeling algorithm. In particular, for *Pokémon* and *Science*, the ordering defined by the peeling algorithm correctly reproduces the ordering of the top five workers in terms of accuracy rates.

5.4 Comparison with Existing Algorithms

Using both synthetic and real-world datasets, we compare the performance of our proposed answer aggregation algorithms, that is, Top-2, Ex-MV, Ex-GLAD, and Ex-Hyper-MV, with existing algorithms, that is, MV, GLAD, and Hyper-MV. We

$(p_{\mathrm{ex}}, n_{\mathrm{ex}})$	Top-2	Ex-MV	Ex-GLAD	Ex-Hyper-MV	MV	GLAD	Hyper-MV
$(0.7, 2)$	**0.519** ±0.218	0.285 ±0.079	0.277 ±0.098	0.276 ±0.079	0.273 ±0.058	0.264 ±0.092	0.272 ±0.062
$(0.7, 4)$	**0.892** ±0.047	0.388 ±0.120	0.563 ±0.212	0.513 ±0.120	0.348 ±0.070	0.549 ±0.212	0.514 ±0.131
$(0.7, 6)$	**0.911** ±0.040	0.624 ±0.247	0.886 ±0.073	0.782 ±0.247	0.426 ±0.066	0.865 ±0.083	0.755 ±0.106
$(0.8, 2)$	**0.750** ±0.097	0.320 ±0.110	0.300 ±0.131	0.405 ±0.110	0.284 ±0.069	0.303 ±0.139	0.380 ±0.126
$(0.8, 4)$	**0.956** ±0.026	0.783 ±0.259	0.896 ±0.138	0.866 ±0.259	0.373 ±0.069	0.814 ±0.160	0.813 ±0.095
$(0.8, 6)$	0.958 ±0.028	**0.984** ±0.018	0.982 ±0.020	0.969 ±0.018	0.484 ±0.065	0.950 ±0.031	0.962 ±0.030
$(0.9, 2)$	**0.879** ±0.048	0.356 ±0.172	0.384 ±0.214	0.702 ±0.172	0.290 ±0.067	0.353 ±0.180	0.704 ±0.146
$(0.9, 4)$	**0.991** ±0.014	0.989 ±0.015	0.991 ±0.014	0.988 ±0.015	0.416 ±0.063	0.941 ±0.091	0.982 ±0.021
$(0.9, 6)$	0.989 ±0.014	**0.998** ±0.005	0.998 ±0.006	0.996 ±0.005	0.550 ±0.073	0.984 ±0.016	0.997 ±0.007

Table 3: Results of answer aggregation algorithms for synthetic datasets, where $n = 100$ and $m = 50$. Each cell gives the average and standard deviation of the accuracy rate over 100 data realizations. For each setting, the best average accuracy rate is written in bold.

Dataset	Top-2	Ex-MV	Ex-GLAD	Ex-Hyper-MV	MV	GLAD	Hyper-MV
Chinese	**0.750**	0.667	0.667	0.700 (±0.026)	0.625	0.542	0.696 (±0.042)
English	**0.733**	0.400	0.533	0.490 (±0.049)	0.467	0.567	0.542 (±0.060)
IT	0.800	0.760	0.800	0.802 (±0.008)	0.760	0.720	**0.828** (±0.018)
Medicine	0.944	**0.972**	**0.972**	0.951 (±0.022)	0.667	0.694	0.848 (±0.019)
Pokémon	**1.000**	**1.000**	**1.000**	**1.000** (±0.000)	0.650	0.850	**1.000** (±0.000)
Science	**0.900**	0.650	0.650	0.603 (±0.025)	0.550	0.550	0.606 (±0.018)

Table 4: Results of answer aggregation algorithms for real-world datasets. Each cell gives the accuracy rate. The average and standard deviation over 100 runs are listed for the hyper-question-based algorithms. For each dataset, the best accuracy rate is written in bold.

first explain the details of GLAD and Hyper-MV. GLAD is a method that takes into account not only the worker expertise, denoted by α, but also the difficulty of each question, denoted by β, We set $\alpha \sim \mathcal{N}(1, 1)$ and $\beta \sim \mathcal{N}(1, 1)$ as in Li et al. [2017]. It is known that GLAD runs in $O(nmsT)$ time, where T is the number of iterations to converge. Hyper-MV is a method that applies the majority voting to the hyper questions rather than the original individual questions. Because the number of possible hyper questions may be too large, Li et al. [2017] suggested to apply the following random sampling procedure for r times: (i) shuffle the order of all single questions uniformly at random to generate a permutation and (ii) from this permutation, pick every k single questions from the beginning of the queue to generate hyper questions as long as they can be picked. Then, the overall time complexity of Hyper-MV is given by $O(rmn)$. We performed the sampling procedure with $k = 5$ for $r = 100$ times, as suggested by Li et al. [2017].

Table 3 shows the results for synthetic datasets. We list the results for nine settings in which the probability p_{ex} varies in $\{0.7, 0.8, 0.9\}$ and n_{ex} varies in $\{2, 4, 6\}$. We set the number of workers n to 100 and the number of questions m to 50. As can be seen, each expert core counterpart achieves better performance than the original algorithm. In particular, Ex-MV significantly improves the performance of MV. Top-2 outperforms the other algorithms particularly when the problem is quite difficult, although Ex-MV performs better when the problem is relatively easy.

Table 4 summarizes the results for the six real-world datasets. As can be seen, for all datasets except *IT*, our proposed algorithms achieve the best performance. In fact, for almost all datasets, the performance of the existing algorithms is improved by using the expert core. Among our proposed algorithms, Top-2 provides the best performance; in particular, for *English* and *Science*, the accuracy rate of Top-2 is even higher than those of the other algorithms. It should be noted that, for *English*, *Medicine*, and *Science*, the accuracy rate of Top-2 is strictly higher than the best accuracy rate among workers (presented in Table 2), which emphasizes the power of answer aggregation in crowdsourcing. According to the trend observed in synthetic datasets, it is very likely that the high performance of Top-2 stems from the high fraction of non-experts in real-world datasets.

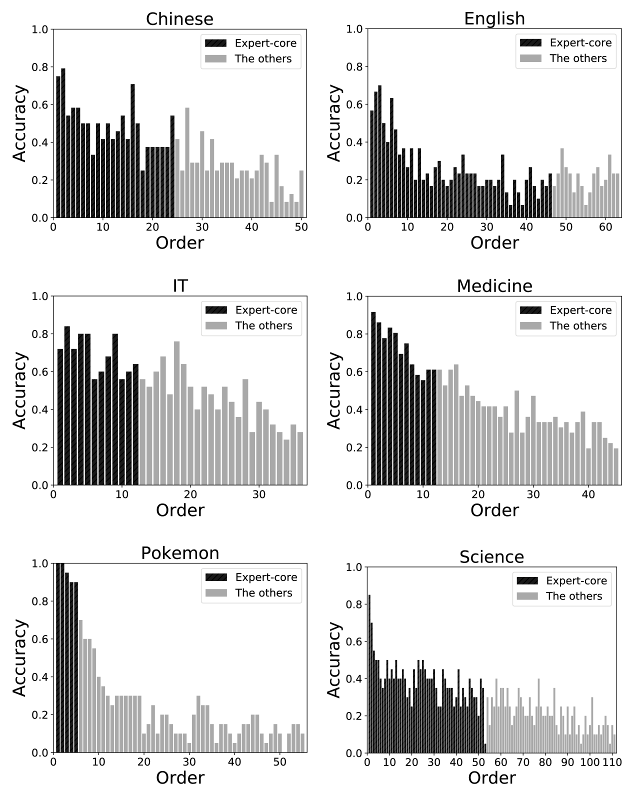

Figure 3: Ordering defined by our peeling algorithm for real-world datasets.

6 Conclusion

In this study, we addressed the answer aggregation task in crowdsourcing. Specifically, we focused on a hard situation wherein a few experts are overwhelmed by a large number of non-experts. To design powerful answer aggregation algorithms for such situations, we introduced the notion of *expert core*, which is a set of workers that is very unlikely to contain a non-expert. We then designed a graph-mining-based efficient algorithm that exactly computes the expert core. Based on the expert core extraction algorithm, we proposed two types of answer aggregation algorithms. The first one incorporates the expert core into existing answer aggregation algorithms. The second one utilizes the information provided by the expert core extraction algorithm pertaining to the reliability of workers. In particular, we provided a theoretical justification for the first type of algorithm: if the number of questions and the number of workers in the expert-core are sufficiently large, our proposed algorithm gives the correct answer with high probability. Computational experiments using synthetic and real-world datasets demonstrated that our proposed answer aggregation algorithms outperform state-of-the-art algorithms.

There are several directions for future research. Our model assumes that all experts give the correct answer with the same probability and all non-experts give an answer independently and uniformly at random. However, in reality, experts themselves may have different levels of expertise and non-experts may not be completely random. Although we have already confirmed that our proposed algorithms work well on real-world datasets, it is interesting to extend our model to such a more general scenario. Another direction is to generalize our model in a higher-level perspective. This study has focused on crowdsourced *closed-ended* questions, where workers can select an answer from candidates. On the other hand, there are often cases where we wish to handle crowdsourced *open-ended* questions, where workers have to answer without any candidates. We believe that our proposed algorithms may be applicable to this more general scenario by introducing a measure of similarity of answers (and thus similarity of workers).

Acknowledgments

This work was partially supported by JST ACT-I Grant Number JPMJPR17U79 and JSPS KAKENHI Grant Numbers 16K16005, 17H07357, 18J23034, and 19K20218.

References

[Andersen and Chellapilla, 2009] Reid Andersen and Kumar Chellapilla. Finding dense subgraphs with size bounds. In *WAW*, pages 25–37, 2009.

[Asahiro *et al.*, 2000] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. *J. Algorithms*, 34(2):203–221, 2000.

[Aydin *et al.*, 2014] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

[Bachrach *et al.*, 2012] Yoram Bachrach, Tom Minka, John Guiver, and Thore Graepel. How to grade a test without knowing the answers: A Bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *ICML*, pages 819–826, 2012.

[Bader and Hogue, 2003] Gary D. Bader and Christopher W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinform.*, 4(1):1–27, 2003.

[Charikar, 2000] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 84–95, 2000.

[Dawid and Skene, 1979] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Appl. Stat.*, pages 20–28, 1979.

[Demartini *et al.*, 2012] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.

[Dourisboure *et al.*, 2007] Yon Dourisboure, Filippo Geraci, and Marco Pellegrini. Extraction and classification of dense communities in the web. In *WWW*, pages 461–470, 2007.

[Finin *et al.*, 2010] Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, and Justin Martineau. Annotating named entities in Twitter data with crowdsourcing. In *NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88, 2010.

[Gibson *et al.*, 2005] David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB*, pages 721–732, 2005.

[Hsueh *et al.*, 2009] Pei-Yun Hsueh, Prem Melville, and Vikas Sindhwani. Data quality from crowdsourcing: A study of annotation selection criteria. In *NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 27–35, 2009.

[Karger *et al.*, 2011] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.

[Kawase and Miyauchi, 2018] Yasushi Kawase and Atsushi Miyauchi. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12):3461–3480, 2018.

[Kawase *et al.*, 2019] Yasushi Kawase, Yuko Kuroki, and Atsushi Miyauchi. Graph mining meets crowdsourcing: Extracting experts for answer aggregation. *arXiv:1905.08088*, 2019.

[Khuller and Saha, 2009] Samir Khuller and Barna Saha. On finding dense subgraphs. In *ICALP*, pages 597–608, 2009.

[Kim and Ghahramani, 2012] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In *AISTATS*, pages 619–627, 2012.

[Li *et al.*, 2014a] Hongwei Li, Bo Zhao, and Ariel Fuxman. The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing. In *WWW*, pages 165–176, 2014.

[Li *et al.*, 2014b] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. A confidence-aware approach for truth discovery on long-tail data. In *VLDB*, pages 425–436, 2014.

[Li *et al.*, 2017] Jiyi Li, Yukino Baba, and Hisashi Kashima. Hyper questions: Unsupervised targeting of a few experts in crowdsourcing. In *CIKM*, pages 1069–1078, 2017.

[Li *et al.*, 2018] Jiyi Li, Yukino Baba, and Hisashi Kashima. Incorporating worker similarity for label aggregation in crowdsourcing. In *ICANN*, pages 596–606, 2018.

[Lin *et al.*, 2015] Yi-ling Lin, Christoph Trattner, Peter Brusilovsky, and Daqing He. The impact of image descriptions on user tagging behavior: A study of the nature and functionality of crowdsourced tags. *J. Assoc. Inform. Sci. Tech.*, 66(9):1785–1798, 2015.

[Liu *et al.*, 2012a] Qiang Liu, Jian Peng, and Alexander T. Ihler. Variational inference for crowdsourcing. In *NIPS*, pages 692–700, 2012.

[Liu *et al.*, 2012b] Xuan Liu, Meiyu Lu, Beng Chin Ooi, Yanyan Shen, Sai Wu, and Meihui Zhang. CDAS: A crowdsourcing data analytics system. In *VLDB*, pages 1040–1051, 2012.

[Liu *et al.*, 2013] Qiang Liu, Alexander T. Ihler, and Mark Steyvers. Scoring workers in crowdsourcing: How many control questions are enough? In *NIPS*, pages 1914–1922, 2013.

[Ma *et al.*, 2015] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *KDD*, pages 745–754, 2015.

[Miyauchi and Kakimura, 2018] Atsushi Miyauchi and Naonori Kakimura. Finding a dense subgraph with sparse cut. In *CIKM*, pages 547–556, 2018.

[Miyauchi and Kawase, 2015] Atsushi Miyauchi and Yasushi Kawase. What is a network community? A novel quality function and detection algorithms. In *CIKM*, pages 1471–1480, 2015.

[Raykar *et al.*, 2010] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

[Snow *et al.*, 2008] Rion Snow, Brendan O'Conno, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263, 2008.

[Tsourakakis *et al.*, 2013] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *KDD*, pages 104–112, 2013.

[Venanzi *et al.*, 2014] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based Bayesian aggregation models for crowdsourcing. In *WWW*, pages 155–164, 2014.

[Wauthier and Jordan, 2011] Fabian L. Wauthier and Michael I. Jordan. Bayesian bias mitigation for crowdsourcing. In *NIPS*, pages 1800–1808, 2011.

[Welinder *et al.*, 2010] Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432, 2010.

[Whitehill *et al.*, 2009] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, and Jacob Bergsma. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043, 2009.

[Zheng *et al.*, 2017] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? In *VLDB*, pages 541–552, 2017.

[Zhou *et al.*, 2012] Dengyong Zhou, John C. Platt, Sumit Basu, and Yi Mao. Learning from the wisdom of crowds by minimax entropy. In *NIPS*, pages 2195–2203, 2012.