Improving Multiclass Classification in Crowdsourcing by Using Hierarchical Schemes

Xiaoni Duan
Kyoto University
Kyoto, Japan
duan@dl.soc.i.kyoto-u.ac.jp

Keishi Tajima
Kyoto University
Kyoto, Japan
tajima@i.kyoto-u.ac.jp

ABSTRACT

In this paper, we propose a method of improving accuracy of multiclass classification tasks in crowdsourcing. In crowdsourcing, it is important to assign appropriate workers to appropriate tasks. In multiclass classification, different workers are good at different subcategories. In our method, we reorganize a given flat classification task into a hierarchical classification task consisting of several subtasks, and assign each worker to an appropriate subtask. In this approach, it is important to choose a good hierarchy. In our method, we first post a flat classification task with a part of data and collect statistics on each worker's ability. Based on the obtained statistics, we simulate all candidate hierarchical schemes, estimate their expected accuracy, choose the best scheme, and post it with the rest of data. In our method, it is also important to allocate workers to appropriate subtasks. We designed several greedy worker allocation algorithms. The results of our experiments show that our method improves the accuracy of multiclass classification tasks.

KEYWORDS

task design; worker assignment; task assignment; annotation

ACM Reference Format:
Xiaoni Duan and Keishi Tajima. 2019. Improving Multiclass Classification in Crowdsourcing by Using Hierarchical Schemes. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313749

1 INTRODUCTION

Classification task, such as image categorization, is one of the most common types of tasks in crowdsourcing. Classification into multiple categories are called multiclass classification. For multiclass classification, flat classification schemes, which show all choices of categories to workers at once, is commonly used. Many crowdsourcing platforms, such as Amazon Mechanical Turk (http://www.mturk.com), provide templates of flat classification scheme.

In crowdsourcing, it is important to assign workers to tasks that they are good at in order to improve the quality of the task outputs. In multiclass classification, some workers are good at some subcategories while other workers are good at other subcategories. In the ordinary flat classification scheme, however, all the workers are assigned to the same task including all subcategories.

We propose a new approach to multiclass classification tasks. Given a flat multiclass classification task, we reorganize it into a hierarchical classification task consisting of several subtasks. We can then assign each worker to a subtask that she is good at.

For example, a flat classification task of classifying images of Canis animals into 7 categories (Siberian Husky, Alaskan Malamute, Samoyed, German Shepherd, wolf, coyote and dhole) can be reorganized into a two-level hierarchical classification task shown in Figure 1. This task consists of the following three subtasks: (1) a root task of classifying images into two top-level subcategories: (Husky, Malamute, Samoyed, Shepherd) and (wolf, coyote and dhole), which intuitively correspond to subcategories "dogs" and "wild species", respectively, (2) a task of classifying images that have been classified into the dog category further into the four breeds, and (3) a task of classifying images that have been classified into the wild species category further into the three species.

In this hierarchical task, we can assign different workers to different subtasks. If we have workers who are good at distinguishing these four dog breeds but not familiar with the wild species, we can assign them to the subtask of classifying dogs. Similarly, if we have workers who are familiar with wild animals but not familiar with dog breeds, we can assign them to the subtask of classifying wild species. Some workers are assigned to the root task of distinguishing dogs and wild species, which is relatively easy. We expect this assignment would improve the overall classification accuracy.

Another advantage of hierarchical schemes is simplicity of tasks. We expect that each subtask with a fewer choices requires less cognitive load and less execution time than the original flat task.

On the other hand, a disadvantage of hierarchical schemes is its cost. In a two-level hierarchical classification scheme, each data goes through two subtasks, thus the total number of the required task instances are double of that of the flat scheme. However, when we have enough budget and enough time, a method that can improve the accuracy of the output by fully utilizing these resources is useful.

Another standard method of improving the quality of crowdsourcing tasks by using more resources is the majority voting: we assign multiple workers to one task, and take majority vote. Improvement by the use of votes by two workers can be bigger than the improvement by the use of two-level hierarchical scheme. However, the majority voting and hierarchical reorganization are not exclusive with each other. For example, we can assign five workers to each subtask in a two-level hierarchical scheme, and take their majority vote. It requires ten workers for each data, so we could instead assign ten workers in a flat classification scheme. We should compare the accuracy of these two schemes. In other words, we

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313749

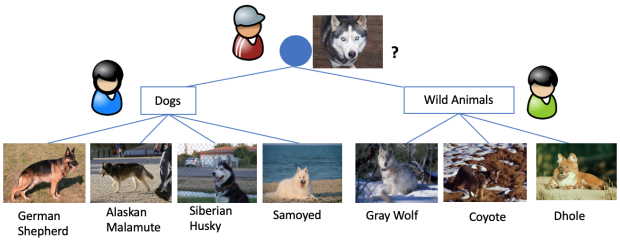

Figure 1: Example of a hierarchical classification scheme

should compare the benefit of two-level schemes with the benefit of doubling the number of workers in the majority voting scheme.

In the majority voting, we can assign as many workers to one task as we want, but the improvement is likely to stop after we reach some number of workers. For example, increasing the number of workers from five to ten is not likely to improve the accuracy significantly. We therefore expect that the benefit of the two-level schemes is bigger than the benefit of doubling the number of workers when we have already assigned enough number of workers.

In our hierarchical reorganization approach, there are many ways to reorganize a given flat classification task into a hierarchical one, and we need to choose an appropriate hierarchy. In this paper, we propose a method of selecting an appropriate hierarchy. We first post a flat classification task with a part of data as a worker qualification task, and collect statistics on each worker's ability. Based on the obtained statistics, we simulate all candidate hierarchical schemes and estimate their expected accuracy. We then choose the best scheme, and post it with the rest of data.

In our method, it is also important to allocate workers to appropriate subtasks in the simulation step, and when posting a hierarchical task. We designed several greedy worker allocation algorithms.

In order to evaluate our method, we ran several classification tasks on Amazon Mechanical Turk. The result shows that hierarchical schemes can improve the accuracy of multiclass classification if we choose appropriate hierarchies.

The contribution of this paper is summarized as follows:

- we propose a hierarchical reorganization approach for improving the accuracy of multiclass classification tasks,
- we propose a method of selecting appropriate hierarchy and a method of assigning workers to appropriate subtasks, and
- we experimentally confirmed that our method can improve the accuracy of multiclass classification tasks.

2 RELATED WORK

Hierarchical classification has been studied much in machine learning [4, 10, 14], but only in a few studies in crowdsourcing.

Vempaty et al. [15] proposed a method of transforming a flat multiclass classification (e.g., into multiple dog breeds) into a combination of many binary classifications (e.g., large/small dogs, and long/short tail dogs). They represent the results of the binary classifications of a data by a sequence of 0/1 and decode it into a class (e.g., a breed) based on coding theories. Their method uses a combination of binary classifications, not hierarchical one. In addition, appropriate binary classifications need to be specified manually.

Weng et al. [16] proposed a method of reorganizing a task of selecting data satisfying multiple conditions. Instead of asking all conditions at once, they reorganize a task into a hierarchical one where questions are asked one by one. In this scheme, they can reduce the number of tasks by narrowing down the candidates in earlier steps. Their goal is to reduce the workload in selection tasks while our goal is to improve the accuracy in classification tasks.

Otani et al. [12] proposed a method of aggregating multiple answers in flat multiclass classification tasks where classes have hierarchical relationships. For example, suppose two workers classify a given book into Society.Sociology category, two workers classify it into Science.Medical, and one worker classify it into Science.Others. In this case, Society.Sociology and Science.Medical have two votes, but they choose the latter because three of the five workers voted to Science category. The task shown to workers in their method is a flat multiclass classification task, and discovery of hierarchical relationship is not discussed.

Chilton et al. [2] and Bragg et al. [1] proposed a workflow for creating hierarchical taxonomy by crowdsourcing. In their methods, each worker classifies a small fraction of the data, and their methods create a globally consistent taxonomy based on the answers. Their purpose is to create a semantically meaningful hierarchy while our purpose is to select a hierarchy that best improve the accuracy of a classification task. As shown later in our experiment, the best hierarchy is not necessarily a semantically meaningful hierarchy.

There have also been many proposals of methods of assigning workers to appropriate tasks. The Hungarian algorithm [9] is a classical algorithm for the optimal task-worker assignment. However, we cannot simply use the Hungarian algorithm for assigning workers to hierarchical subtasks because the accuracy of subtasks in hierarchical classification tasks have inter-dependency: if the results of the upper subtasks are wrong, it cannot be corrected in the lower subtasks. In addition, the Hungarian algorithm is too slow when we have many workers. We develop new greedy algorithms for assigning workers to hierarchical classification subtasks.

Mavridis et al. [11] proposed a worker allocation method based on the hierarchical relationship between required skills. Their skill hierarchy, however, does not correspond to classification hierarchy, and we cannot use it for worker allocation in our method.

Our hierarchical schemes require more workers, and such resources can instead be used for repeated labeling as explained before. Sheng et al. [13] showed that repeated labeling does not always improve the output quality. For deciding whether we should assign more workers or not, Gao et al. [5] designed two models to estimate the profit of additional tasks. Kamar and Horvitz [7] also proposed a method of deciding whether to hire more workers or not. Karger et al. [8] has shown a theoretical result on the trade-off between the quality and the number of workers for each task in multiclass classification tasks. We can use these methods to determine whether we should introduce hierarchical schemes or should increase the number of workers for one data instead.

3 OUR METHOD

In this section, we explain the details of the four steps of our method: (1) posting qualification tasks, (2) generating candidate hierarchical schemes, (3) estimating ability of workers at subtasks in the

candidate schemes based on the statistics obtained in the qualification tasks, and (4) allocating workers to subtasks in the candidate schemes for estimating the expected accuracy of the schemes.

3.1 Qualification Task

As explained in Section 1, we first post an ordinary flat classification task with a part of data as a qualification task. Qualification tasks are popularly used in crowdsourcing for eliminating spammers. We use it also for calculating the accuracy of each worker for each category, and creating a confusion matrix of each worker.

For these purposes, we need the ground truth for the data used in the qualification task. In our experiment, we used data with the ground truth. When it is not available, we can estimate it by majority voting or more recent alternatives [3, 6, 17].

In our experiment, 20 to 30 items from each category was enough to obtain stable statistics. We use the same data set for all users.

3.2 Hierarchical Scheme Generation

We then generate candidate hierarchical schemes, and run simulations of them. There are, however, too many possible hierarchies to run simulation of. In this paper, we only consider two-level schemes because the number of levels corresponds to the number of subtasks each data goes through, and the cost of the whole classification task is proportional to the number of levels. The scheme in Figure 1 is an example of a two-level scheme with one root task, two top-level categories, two leaf tasks for them, and seven final leaf classes.

Because the number of two-level schemes is still large, we only consider schemes with at most $\lfloor n/2 \rfloor$ top-level categories when we have n leaf classes. Schemes with more top-level categories have at least one top-level category with only one leaf class, and are probably inefficient. With this restriction, the number of candidate schemes for n final classes is $\sum_{k=2}^{\lfloor n/2 \rfloor} S(n,k)$, where $S(n,k)$ is the Stirling numbers of the second kind.

3.3 Estimation of Worker Ability

Given a candidate scheme, we estimate two types of accuracy of each worker based on the results of the qualification task: (1) accuracy in the root task, and (2) accuracy in each leaf task.

When we estimate the root task accuracy of a worker, if the worker has classified an item of one class into another class within the same top-level category in the qualification task, it is regarded as a correct answer. For example, suppose we estimate the accuracy of a worker at the root subtask in the scheme in Figure 1. If she has classified a Husky image into the Samoyed class in the qualification task, it is regarded as correct. We calculate the ratio of the correct answers to the all answers in the qualification task.

On the other hand, when we calculate the accuracy of a worker at a leaf task corresponding to a top-level category A, we only consider items in the qualification task whose correct classes belong to A. In the qualification task, the worker sometimes classifies an item in A into some wrong leaf class c which does not belong to A. When the worker is really assigned to the leaf task for A, she cannot classify the item to c because c is not in the choices. In such cases, we determine which class she would choose based on the probability distribution of her answer in the qualification task.

For example, suppose we estimate the accuracy of a worker at the leaf task for the top-level category "dog" in the scheme in Figure 1. In the qualification task, the worker classified 6 images of Husky: 3 into the Husky class, 2 into Malamute, and 1 into wolf. When the worker is assigned to the leaf task for the dog category, she cannot classify an image into the wolf class, which is not in the choices. When calculating her accuracy in this subtask, we assume that she would classify the image she misclassified into wolf, either into Husky or Malamute in the probability 3/5 and 2/5, respectively.

We also collect statistics on task speed of workers. In order to minimize the response time of a task, the number of tasks assigned to workers should be proportional to their task speed.

3.4 Worker Allocation Algorithms

We have calculated the accuracy of workers at subtasks in the candidate scheme. In order to estimate the expected overall accuracy of the scheme, we also need to decide worker allocation to subtasks.

Notice that we cannot simply allocate each worker to a subtask for which she has her best accuracy. If we did it, we would have no worker for difficult subtasks. We need to assign appropriate number of workers to each subtask for minimizing response time.

We can assign a worker to more than one subtasks, but even if we do it, the worker would only work on a subtask with the best reward/time ratio until all the data at the subtask is processed. In this paper, we assume that each subtask has far more data than one worker can process. We, therefore, only consider simple worker allocation where each worker is allocated only to one subtask.

There are too many ways to allocate workers to subtasks, and we cannot calculate expected accuracy for all of them. Even with only three subtasks and 20 workers, we have $3^{20} > 3$ billion ways to allocate workers. To decide worker allocation without an exhaustive search, we designed four greedy worker allocation algorithms.

Algorithm 1 gives priority to workers whose accuracy largely changes depending on tasks. It is important to assign such workers to appropriate tasks. On the contrary, we give the least priority to workers who have the same accuracy for all tasks.

After calculating accuracy of workers for subtasks in the given scheme, we create a worker list l_r, which is sorted by the accuracy of workers at the root subtask in descending order. We also create worker lists $l_1, \ldots, l_k$, each of which is sorted by the accuracy of workers at the corresponding leaf subtask. Let $r_r^u, r_1^u, \ldots, r_k^u$ be u's ranks in $l_r, l_1, \ldots, l_k$. We calculate the variance of $r_r^u, r_1^u, \ldots, r_k^u$ for each worker. Here we measure the variance of u's ability by using her ranks instead of the accuracy values because the accuracy values for different subtasks are incomparable. We then produce a list of workers, W, sorted by the variance in descending order.

We also maintain variables $j_r, j_1, \ldots, j_k$ storing how many jobs are remaining at the root task and each leaf task. Suppose we have m data items, n leaf classes, and a hierarchical scheme with k top-level categories, which includes $n_1, \ldots, n_k$ leaf classes, respectively. We may not know the ratio of data in each leaf class, so we assume that each leaf class has m/n items. We initialize j_r to m and initialize each j_i $(i = 1, \ldots, k)$ to either $m * n_i/n$ if $n_i > 1$, or 0 if $n_i = 1$. We are also given w_u for each u, the relative task speed of u.

We scan W starting from its top, and for each u in W, we find the largest value among $r_r^u, r_1^u, \ldots, r_k^u$, i.e., the best subtask for u.

If it has remaining jobs, we assign u to that task and subtract the expected number of jobs the worker u will do, i.e., $2m * w_u/\Sigma_j w_j$, from the number of its remaining jobs. If it has no remaining job, we assign u to the next best task for u.

Algorithm 2 is same as Algorithm 1 except that we use the accuracy values instead of ranks. Because the accuracy values in different tasks are incomparable, we normalize them to the standard scores. Let a_i^u be the accuracy of worker u for the task i. For each task i ($i = r, 1, \ldots, k$), we calculate the mean of accuracy $\bar{a}_i = \sum_u a_i^u/|W|$ and the standard deviation $\sigma_i = \sqrt{\sum_u (a_i^u - \bar{a}_i)^2/|W|}$, where $|W|$ is the number of workers. We then calculate s_i^u, the standard score of the worker ability, by $s_i^u = (a_i^u - \bar{a}_i)/\sigma_i$. We use the variance of s_i^u instead of the variance of r_i^u when we sort W and when we select the best subtask for u.

Algorithm 3 is the opposite of Algorithm 1 and 2. It gives priority to tasks whose accuracy largely changes depending on the allocated workers. In this case, we do not need to use rankings or standard scores. For each task i, we calculate the average accuracy $\bar{a}_i = \sum_u a_i^u/|W|$ and the variance $\sigma_i^2 = \sum_u (a_i^u - \bar{a}_i)^2/|W|$. We then give priority to a task i with the largest σ_i^2, and assign it a remaining worker with the highest accuracy for i.

Algorithm 4 gives priority to a task with the largest ratio of remaining jobs. It repeatedly selects a task corresponding to the largest value among $j_r/j_r^0, j_1/j_1^0, \ldots, j_k/j_k^0$, where $j_r^0, j_1^0, \ldots, j_k^0$ are the initial values of $j_r, j_1, \ldots, j_k$ explained before, and allocates a remaining worker with the highest accuracy for that task. We repeat it until all the workers are assigned.

For all combinations of candidate schemes and our four allocation algorithms, we compute worker allocation, and compute the expected overall accuracy of the task under that worker allocation. We then choose the combination that achieved the highest accuracy, post the classification task using that hierarchical scheme, and assign workers to subtasks by using that worker allocation.

4 EXPERIMENTS

We ran experiments on two data sets on Amazon Mechanical Turk (AMT). The results are explained in this section.

4.1 Experiment of classifying Canis Animals

In the first experiment, we compare the accuracy of flat and hierarchical classification tasks classifying images of 7 kinds of Canis animals explained in Section 1. In this experiment, we only consider hierarchical schemes with two top-level categories, which we call A and B. We have $2^7 - 1 = 63$ such schemes (excluding one scheme that is equivalent to the flat classification).

We collected over 100 images of each of these seven kinds of Canis animals, and obtained 800 images in total. We posted a flat classification task with randomly chosen 200 images on AMT as a qualification task. The remaining 600 images are used in the main task later. One task instance consists of 20 images. We obtained information on 153 workers, and removed 11 workers as spammers.

We then estimated overall accuracy of 63 hierarchical schemes by allocating workers with Algorithm 1 and 4. For all 63 schemes, both Algorithm 1 and 4 achieved better expected accuracy than the accuracy observed in the flat classification task posted as the qualification task, which was 0.746. In addition, for all 63 schemes, Algorithm 1 achieved better accuracy than Algorithm 4. Table 1 lists schemes that achieved the three best and the three worst accuracies with Algorithm 1. The columns A and B show the leaf classes allocated to the two top-level categories.

Table 1: Best/Worst Hierarchical Schemes with Algorithm 1

rank	accuracy	A	B
1	0.866	(Samoyed, wolf, coyote)	others
2	0.862	(Malamute, Husky, dhole)	others
3	0.862	(Shepherd, Malamute, Husky)	others
⋮	⋮	⋮	⋮
61	0.807	(Shepherd)	others
62	0.800	(Malamute, wolf, dhole)	others
63	0.776	(Malamute, coyote)	others
flat	0.746		

Table 2: Confusion Matrix for All Users (bold fonts: > 0.1)

	Malamute	coyote	dhole	wolf	Shepherd	Samoyed	Husky
Malamute	0.611	0.002	0.009	0.016	0.024	0.038	**0.300**
coyote	0.018	0.618	**0.134**	**0.182**	0.016	0.018	0.015
dhole	0.010	**0.209**	0.698	0.030	0.012	0.018	0.015
wolf	0.053	**0.118**	0.026	0.674	0.022	0.029	0.077
Shepherd	0.009	0.009	0.022	0.008	0.917	0.018	0.017
Samoyed	**0.104**	0.006	0.045	0.020	0.015	0.755	0.055
Husky	**0.218**	0.010	0.008	0.040	0.028	0.023	0.674

Table 3: Accuracy of Real Tasks on AMT

Scheme	Total	Root	Sub-A	Sub-B
best scheme in Table 1 (3 to 4)	0.875	0.970	0.853	0.850
A: (wolf) B: others (1 to 6)	0.765	0.900	0.880	0.749
worst scheme in Table 1 (2 to 5)	0.590	0.610	0.444	0.644
flat (majority voting)	0.833	-	-	-
flat (weighted majority voting)	0.767	-	-	-

Table 2 shows the confusion matrix calculated from the answers by the workers in the qualification task. For example, it shows that 0.2% of Malamute images were classified as coyote. Misclassification with a probability higher than 0.1 is shown in bold fonts. This table shows that Malamute and Husky are most difficult to distinguish, coyote and dhole are the next, and coyote and wolf follow.

In all the top three schemes shown in Table 1, the most difficult pair, Malamute and Husky, are in the same top-level category, while they are in different top-level categories in the two worst schemes. It suggests that we can improve overall accuracy by allocating these two kinds to the same top-level category, and assigning workers who are good at distinguishing them to the leaf task corresponding to that top-level category. Another confusing pair, wolf and coyote, are also in the same top-level category in all the top three schemes.

We then posted three hierarchical classification tasks on AMT to know their real accuracy (i.e., not the estimation by our simulation). We chose the best scheme and the worst scheme in Table 1, and also the best 1-to-6 scheme (i.e., one class in A and 6 classes in B) as a medium-level scheme. Because the best scheme is 3-to-4 scheme and the worst scheme is 2-to-5 scheme, these three schemes include all n-to-m cases. Table 3 lists the three schemes and their results.

We have 600 images excluding 200 images used in the qualification task. We randomly divide them into three sets of 200 images and allocate them to the three hierarchical tasks. All 142 workers are invited to the three tasks, which were posted at the same time. We also posted a flat classification task with the same 600 images.

In all the tasks, five workers are assigned to each image. We used two methods to aggregate the five answers: majority voting and the standard EM-based weighted voting method [3]. When we have a tie in majority voting (e.g., 2 votes to wolf, 2 to coyote, and 1 to dhole), we randomly choose one of the classes involved in the tie.

We show the result in Table 3. Because the accuracy of the flat classification with the weighted voting is lower than that with the simple majority voting, we only used the simple majority voting in the hierarchical tasks. The hierarchical scheme which was best in the simulation achieved higher accuracy than the flat scheme. It confirms that we can improve the accuracy of a multiclass classification task by reorganizing it into a hierarchical task.

However, the accuracy of the other two hierarchical schemes are worse than that of the flat scheme although their expected accuracy was higher than it. This result suggests that the choice of hierarchical schemes significantly influence workers' performance, and if you choose an inappropriate scheme, it can result in the accuracy that is worse than the expected accuracy estimated from the workers' performance in the flat classification task.

Also note that none of the best three schemes in Table 1 is the semantically meaningful one shown in Figure 1. It shows that it is not easy to manually determine the good hierarchy, and we should compute it based on the ability of currently available workers.

To confirm that good hierarchies depend on the workers available at the time, we run the experiment of classifying Canis animals twice. In the previous run, the most confusing pair, Malamute and Husky, are allocated to one top-level category, and another confusing pair, wolf and coyote, are allocated to the other top-level category, in the three best schemes. In the second run, these two pairs are assigned to the same top-level category in the two best schemes. This happened because the set of workers who are good at distinguishing Malamute and Husky, and the set of workers who are good at distinguishing wolf and coyote, were relatively disjoint in the previous run, but have more overlaps in the second run. As a result, allocating both pairs to the same top-level category becomes a better strategy. As shown in this result, the best hierarchical schemes depends on the set of workers currently available.

4.2 Experiment with Reptiles and Amphibians

Next, we compare flat and hierarchical tasks classifying images of the following 10 kinds of reptiles and amphibians: Komodo dragon, frilled agama, chameleon, gecko, iguana, skink, giant salamander, tuatara, and basilisk. In the previous experiment, we only generated hierarchical schemes with two top-level categories, but in this experiment, we use 2 to 5 top-level categories.

We collected over 100 images for each class and obtained over 1000 images in total. We then published a qualification task on AMT with 200 images, collected answers from 307 workers, and we eliminated 44 workers as spammers, which result in 263 workers.

We computed the confusion matrix from their answers. We omit the details, but the most frequent misclassification was 21.7% of tuatara images classified into the iguana class, which look similar and better known. Compared with Canis animals, the distinction between confusing pairs and the others are not clear, and the confusion matrix is less symmetric. This is because the 10 classes in this task include both well-known classes and less-known classes.

We then generated all the hierarchical schemes with 2 to 5 top-level categories, and ran Algorithm 1 to 4. We obtain the best accuracy with Algorithm 2, which uses the standard score of the accuracy of workers. Table 4 shows some results obtained with Algorithm 2. It lists the best scheme, which has four top-level categories, the best one among those with five top-level categories (202nd), the best one among those with three top-level categories (6422nd), and the worst scheme. The worst scheme has five top-level categories, four of which include only one leaf class. The best to the 201th schemes all have four top-level categories.

In the best scheme shown in Table 4, basilisk is allocated to the same top-level category as frilled agama and chameleon, to which basilisk images are often misclassified. In this scheme, newt and giant salamander, which are often misclassified to each other, are also allocated to the same top-level category. Newt and giant salamander are allocated to the same top-level categories in most schemes with high accuracy (although it is not shown in Table 4). This is similar to what happened in the previous experiment.

We posted hierarchical classification tasks on AMT with the four schemes shown in Table 4, and assigned 263 qualified workers to the subtasks by Algorithm 2. We divide 800 images that were not used in the qualification task to four tasks so that each of them has 200 images. In each subtask, each image is classified by five workers, and we take the majority votes.

In this experiment, we calculate the accuracy of the flat scheme by using the answers in the qualification task. In the hierarchical tasks, we take majority votes by 5 workers, but in the flat task, we take majority votes by 10 workers because our two-level schemes require twice as many workers as the flat scheme. We randomly choose 10 users who participated in the hierarchical classification tasks, take the majority votes of their answers in the flat qualification task, and calculate the overall accuracy. We repeat the random selection of 10 users five times and took the average of the accuracy. We only choose users who also participated in the hierarchical tasks in order to be fair. If we include workers who did not participate in the hierarchical tasks, the accuracy of the flat scheme was lower.

Table 5 shows the results. It also shows the result we would have if we take majority votes by 3 workers for hierarchical schemes and 6 workers for the flat scheme. When we calculate the accuracy in these cases, we randomly choose 3 (or 6) workers from 5 (or 10) workers and take their majority votes. We repeat it five times and calculate the average of the accuracy.

When we take majority votes by 5 and 10 workers, our top and the 202th schemes achieved higher accuracy than the flat scheme. The 202th scheme achieved better accuracy than the top scheme. However, the 6422th scheme shows lower accuracy than the flat scheme, so it is again important to choose a good hierarchy.

On the other hand, when we take majority votes by 3 and 6 workers, the flat scheme achieved higher accuracy. The results in these two settings shows that the improvement we obtain by increasing the number of workers from 6 to 10 in the flat scheme is not as big as the improvement we obtain by increasing it from 3 to

Table 4: Part of Hierarchical Schemes and their Expected Accuracy Obtained with Algorithm 2

rank	accuracy	*A*	*B*	*C*	*D*	*E*
1	0.918	skink, tuatara	Komodo dragon, gecko	agama, chameleon, basilisk	others	-
202	0.903	basilisk	Komodo dragon, gecko	skink, agama, iguana	newt	others
6422	0.878	iguana, gecko, tuatara	skink, Komodo dragon, newt, basilisk	agama, chameleon, giant salamander	-	-
last	0.626	Komodo dragon	chameleon	skink	iguana	others

Table 5: Result with Majority Votes by 5 Workers (10 for Flat Scheme) and 3 Workers (6 for Flat Scheme)

Scheme	by 5 (10)	by 3 (6)
1st	0.885	0.855
202th	**0.890**	0.845
6422th	0.830	0.820
last	0.860	0.755
flat (majority voting)	0.863	**0.860**
flat (weighted majority voting)	0.730	-

Table 6: Average Task Execution Time (in second) for Reptile and Amphibian Images

	flat task			hierarchical task	
	all u	u at root	u at leaf	root	leaf
best	916.798	2372.66	578.958	2311.78	278.880

5 in the hierarchical schemes, and as a result, the accuracy of some hierarchical schemes become higher than that of the flat scheme when we have 5 (or 10) workers. This result suggests that our hierarchical scheme is useful when we have a budget to hire more workers, but we have already allocated enough number of workers to each task, and the benefit of doubling the number of workers is likely to be smaller than the benefit of hierarchical schemes.

4.3 Temporal Factors

When we take majority voting by the half number of workers in a hierarchical task, the number of required task instances is the same as in the flat task. We should also compare the workload of one task instance. Subtasks in hierarchical tasks have a fewer choices, so it may require lighter cognitive load than the tasks in the flat scheme.

To confirm it, we measured the average task execution time for a task instance including 20 reptile and amphibian images in the following cases: the average over flat tasks by all workers, the average over flat tasks done by workers who are allocated to the root task when they worked in the best hierarchical scheme, the average over flat tasks done by workers who are allocated to the leaf tasks when they worked in the best hierarchical scheme, the average over the root task in the best scheme, and the average over the leaf tasks in the best scheme. Table 6 shows the results.

The average execution time of flat tasks is 916.798, but the average over flat tasks done by workers who were allocated to the root task in the best hierarchical scheme is 2372.66. It means that our algorithm tend to allocate slow workers to the root task in this scheme. The average execution time of the root task in the best hierarchical scheme is almost the same. This is reasonable because both of them show workers all the leaf classes as the choices. The average execution time of the leaf tasks in the best hierarchical scheme is shorter than that of the flat task done by the same workers. This is also reasonable because the leaf tasks of the hierarchical scheme shows fewer choices to workers. The sum of the average execution time of the root tasks and the leaf tasks is bigger than the double of that of the flat task. Therefore, when we have a time constraint, hierarchical schemes have a disadvantage.

We also measured the time required by the worker allocation algorithms. In the experiment with images of Canis animals, it processed 63 schemes in a few minutes. In the experiment with images of reptiles and amphibians, we have a huge number of schemes because we generate schemes with 2 to 5 top-level categories, but our algorithms processed all of them within a day. This cost is not impractical when we classify a large number of data in crowdsourcing because such a task usually takes far longer. If we have more classes, and want to generate schemes including a larger number of top-categories, the time required in this step can be prohibitive. In that case, we should limit the number of top-categories. On the other hand, the time complexity of the Hungarian algorithm is in $O(n^3)$, and it is also infeasible when n is large.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a new approach to multiclass classification tasks in crowdsourcing. By reorganizing a task into a hierarchical classification task consisting of multiple subtasks, we can allocate workers with different expertise more appropriately to subtasks.

In our method, we generate all reasonable schemes, and estimate their expected accuracy by virtually assigning workers to its subtasks. For this step and also for the real execution of hierarchical tasks, we developed four greedy worker allocation algorithms. They give priority either to workers whose ability largely changes depending on tasks, to tasks whose accuracy largely changes depending on workers, or to tasks with higher remaining job ratio.

We posted hierarchical tasks on AMT with two data sets. We confirmed that we can achieve higher accuracy than the flat tasks if we choose appropriate hierarchical schemes, and also confirmed that our method can choose appropriate schemes. Because the best algorithm among our four algorithms depends on the data set, the best strategy is to use all of them to run simulation for all candidate schemes, and choose the one that achieved the best accuracy.

Our worker allocation algorithms have large room for improvement. We also need a method for efficiently enumerating promising candidate schemes because we cannot enumerate all schemes when we have many classes. They are remaining issues for future work.

6 ACKNOWLEDGMENTS

This work was supported by JST CREST Grant Number JPMJCR16E3, Japan.

REFERENCES

[1] Jonathan Bragg, Mausam, and Daniel S. Weld. 2013. Crowdsourcing Multi-Label Classification for Taxonomy Creation. In *Proc. of HCOMP*. 25–33.

[2] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing Taxonomy Creation. In *Proc. of SIGCHI*. 1999–2008.

[3] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* 28, 1 (1979), 20–28.

[4] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. 2013. Learning hierarchical features for scene labeling. *IEEE TPAMI* 35, 8 (2013), 1915–1929.

[5] Jinyang Gao, Xuan Liu, Beng Chin Ooi, Haixun Wang, and Gang Chen. 2013. An online cost sensitive decision-making method in crowdsourcing systems. In *Proc. ACM SIGMOD*. 217–228.

[6] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proc. of ACM SIGKDD workshop on human computation*. 64–67.

[7] Ece Kamar and Eric Horvitz. 2015. Planning for Crowdsourcing Hierarchical Tasks. In *Proc. of AAMAS*. 1191–1199.

[8] David R Karger, Sewoong Oh, and Devavrat Shah. 2013. Efficient crowdsourcing for multi-class labeling. *ACM SIGMETRICS Performance Evaluation Review* 41, 1 (2013), 81–92.

[9] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)* 2, 1-2 (1955), 83–97.

[10] Shailesh Kumar, Joydeep Ghosh, and Melba M Crawford. 2002. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis & Applications* 5, 2 (2002), 210–220.

[11] Panagiotis Mavridis, David Gross-Amblard, and Zoltán Miklós. 2016. Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing. In *Proc. of WWW Conf.* 843–853.

[12] Naoki Otani, Yukino Baba, and Hisashi Kashima. 2015. Quality Control for Crowdsourced Hierarchical Classification. In *Proc. of IEEE ICDM*. 937–942.

[13] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proc. ACM SIGKDD*. 614–622.

[14] Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22, 1-2 (2011), 31–72.

[15] Aditya Vempaty, Lav R Varshney, and Pramod K Varshney. 2014. Reliable crowdsourcing for multi-class labeling using coding theory. *IEEE Journal of Selected Topics in Signal Processing* 8, 4 (2014), 667–679.

[16] Xueping Weng, Guoliang Li, Huiqi Hu, and Jianhua Feng. 2017. Crowdsourced Selection on Multi-Attribute Data. In *Proc. ACM CIKM*. 307–316.

[17] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R. Movellan. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Proc. of NIPS*. 2035–2043.