TDG4Crowd:Test Data Generation for Evaluation of Aggregation Algorithms in Crowdsourcing

Yili Fang , **Chaojie Shen** , **Huamao Gu** , **Tao Han** and **Xinyi Ding**[*]
School of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou 310018, China
{fangyl,ghmsjq,hantao,xding}@zjgsu.edu.cn, qacket@126.com

Abstract

In crowdsourcing, existing efforts mainly use real datasets collected from crowdsourcing as test datasets to evaluate the effectiveness of aggregation algorithms. However, these work ignore the fact that the datasets obtained by crowdsourcing are usually sparse and imbalanced due to limited budget. As a result, applying the same aggregation algorithm on different datasets often show contradicting conclusions. For example, on the *RTE* dataset, *Dawid and Skene* model performs significantly better than *Majority Voting*, while on the *LableMe* dataset, the experiments give the opposite conclusion. It is challenging to obtain comprehensive and balanced datasets at a low cost. To our best knowledge, little effort have been made to the fair evaluation of aggregation algorithms. To fill in this gap, we propose a novel method named *TDG4Crowd* that can automatically generate comprehensive and balanced datasets. Using Kullback Leibler divergence and Kolmogorov–Smirnov test, the experiment results show the superior of our method compared with others. Aggregation algorithms also perform more consistently on the synthetic datasets generated using our method.

1 Introduction

Crowdsourcing has been successfully applied in various fields where tasks could not be automatically dealt with by computers, e.g. named image annotation [Russell *et al.*, 2008] and entity recognition [Wang *et al.*, 2013]. Some known popular crowdsourcing platforms include Amazon Mechanical Turk (AMT), Appen (used name: Crowdflower), etc. Requesters use these platforms to publish tasks and workers from Internet get paid by completing them based on platform's reward policy. Due to the diverse background of workers, for tasks like image annotation, in order to get more accurate labeling results, one task is usually sent to different workers and an aggregation algorithm will be applied to infer the ground truth. Researchers have proposed many aggregation algorithms, such as *Majority Voting(MV)* [Jing *et al.*, 2018], *Dawid and Skene (DS)* [Dawid and Skene, 1979], *GLAD* [Whitehill *et al.*, 2009], etc. They all have achieved empirical success in practice.

Existing work mainly evaluate different aggregation algorithms on small scale datasets collected from crowdsourcing. However, the performance of these aggregation algorithms highly depend on the crowdsourced datasets, which makes it difficult to fairly compare these algorithms. For example, in [Imamura *et al.*, 2018], *MV* and *DS* are used on datasets *RTE*, *BIRD*, and *DOG*. The results show that the accuracy of *DS* is significantly higher than that of *MV*. However, in [Zhang *et al.*, 2019], on datasets *Leaves* [Zhang *et al.*, 2016a], *LableMe*, and *Music Genre* [Long and Hua, 2016], the experiments give the opposite conclusion that *MV* performs better than *DS*. This is caused by the problems the crowdsourced datasets have, which could manifest in two aspects. On one hand, this could due to the imbalanced distribution of data [Snow *et al.*, 2008], including the feature distribution of annotators, the feature distribution of instances, and the distribution of corresponding annotations. On the other hand, research shows that aggregation accuracy could change with task redundancy [Galal and El-Sharkawi, 2019]. We argue such comparisons of aggregation algorithms may be invalid due to the issues crowdsourced datasets have.

Due to limited budget, requesters can either choose to publish fewer tasks with high redundancy [Li and Yu, 2014; Li *et al.*, 2013], or more tasks with low redundancy [Wauthier and Jordan, 2011; Liu *et al.*, 2012]. Requesters with fewer tasks and low redundancy sometimes get little or no data at all [Zhang *et al.*, 2016b]. In general, the resulting datasets are not able to fully cover various situations, leading to unstable, or even contradictory aggregation results [Imamura *et al.*, 2018; Zhang *et al.*, 2019; Sinha *et al.*, 2018]. We argue the importance of comprehensive and balanced datasets for the fair evaluation of aggregation algorithms. Collecting such datasets through crowdsourcing platforms is difficult due to the tedious process of collection and high cost. To our best knowledge, in the context of crowdsourcing, currently there is little research on generating data in an automated way. In this paper[1], we try to solve this problem by training a generative

[*]Corresponding author

[1]The apendices and souce code are available at https://github.com/Qacket/TDG4Crowd

model that can automatically generate instances, annotators, and corresponding annotations. Our main goal is to obtain a dataset whose distribution follows closely to that of the real one.

We propose a novel automatic test data generation method for the evaluation of aggregation algorithms in crowdsourcing named *TDG4Crowd*. Unlike many other work that only use one latent variable to represent one worker's ability or the difficult level of one task, we learn the latent vector distribution of instances and annotators based on *variational autoencoder (VAE)* [Mescheder *et al.*, 2017] and train the annotation process in a supervised manner. Using our method could provide comprehensive and balanced synthetic datasets for the fair evaluation of aggregation algorithms. The contributions of this paper are summarized as follows:

- We propose a novel method *TDG4Crowd* that can automatically generates synthetic datasets based on a small number of real seed data points.
- We discuss real world scenarios that may result in imbalanced datasets and design different strategies of generating synthetic data.
- We conduct comprehensive experiments and the results show that the synthetic data generated using our method follows more closely to the real distribution compared with other baseline models. Besides, existing aggregation algorithms also perform more consistently on our synthetic data.

2 Related Work

In crowdsourcing, requesters publish tasks redundantly on platforms like AMT and workers from Internet get paid by completing these tasks based on the platform's reward policy. An aggregation algorithm will then be applied on these noisy labels to infer the ground truth. Thus, the design of aggregation algorithms becomes a core problem in crowdsourcing research. Many aggregation algorithms have been proposed in recent years [Ho *et al.*, 2013; Roy *et al.*, 2015; Yu *et al.*, 2017]. Majority Voting [Sheng, 2011; Sheng *et al.*, 2008] is a simple yet effective method in which one instance will be assigned the label that has the most votes. Dawid Skene *et al.* [Dawid and Skene, 1979] proposed a method to infer ground truth based on the EM algorithm, in which a confusion matrix is used to describe the behavior of annotators. GLAD [Whitehill *et al.*, 2009] is a probablistic model that considers the ability of workers and the difficult level of tasks and could be used to infer the true label in image annotation. Based on DS, Li *et al.* proposed the Homogenous Dawid-Skene model (HDS) [Li and Yu, 2014] . Albarqouni *et al.* extended on the work of DS, replacing the logistic classifier with a deep neural network [Albarqouni *et al.*, 2016]. Rodrigues *et al.* also use neural network to estimate one annotator's ability, training the model in an end-to-end manner [Rodrigues and Pereira, 2018]. A multiple noisy label distribution propagation method is proposed to exploit the intercorrelation among labels in [Zhang *et al.*, 2019]. All these work focus on how to better modeling tasks and worker ability to improve the aggregation results and they have all shown empirical success on different datasets.

To fairly evaluate the effectiveness of different aggregation algorithms, current research mainly use test datasets or theoretical analysis. For the first approach, real crowdsourced datasets are mainly used. However, due to limited budget, the crowdsourced datasets are often sparse and imbalanced, resulting in conflicting observations. For example, in work [Sinha *et al.*, 2018], when the SentimentPolarity (SP) dataset is used to evaluate aggregation methods DS and FDS, DS performs better than FDS. But for the AffectAnnotation dataset, the experiments give the opposite results. In [Whitehill *et al.*, 2009], the accuracy of GLAD is higher than that of MV, while in [Kawase *et al.*, 2019], MV performs better than GLAD. This indicates the results are unreliable when evaluating aggregation algorithms on sparse or imbalanced datasets. When it comes to theoretical analysis, current research work mainly focus on the upper bound of error rate and the cost complexity. For the upper bound of error rate, Wang *et al.* study the annotation quality of MV theoretically and the results show the error rate declines exponentially with the increasing of the redundancy level of tasks [Wang and Zhou, 2015]. [Gao *et al.*, 2016] proposed the optimal error rate for aggregating labels provided by a set of non-expert workers. [Heinecke and Reyzin, 2019] utilize the PAC theory to establish the relationship between error rate and worker ability. Considering machine learning based on crowdsourcing, [Pan *et al.*, 2023] analyze the upper bound of minimally sufficient number of crowd labels required for learning a probably approximately correct (PAC) classification model in crowdsourcing learning. For the cost complexity, [Fang *et al.*, 2018] provide a general theoretical method to model the trade-off between costs and quality, which can be used to theoretically analyze crowdsourcing algorithms based on statistical learning. These theoretical results, although could provide reliability guarantee in some degree, they often fail in real environment. A real environment usually contains a lot different factors, among which the ability of workers plays a significant role. Overall, existing work using real test datasets or theoretical analysis can not fairly evaluate different aggregation algorithms.

It is usually very expensive to obtain comprehensive and balanced real datasets and the theoretical analysis of different algorithms often fail in real environment. In this study, we focus on automatic test data generation for the evaluation of aggregation algorithms. We want our generated data follows closely to the distribution of the real one. There exists a few work that also investigate using synthetic data. In [Whitehill *et al.*, 2009], authors use normal distribution to model workers' ability, as well as task difficult level. [Li and Yu, 2014] use beta distribution and one probabilistic model to fit worker responses. Zhang *et al.* build a confusion matrix for worker ability to evaluate algorithms like D&S, etc. [Zhang *et al.*, 2016b]. All these work use prior knowledge when modeling worker ability, which will usually fail in real environment where these prior assumptions are not hold. In this paper, different with existing work, we build models to learn worker and task related parameters directly from a few real data points, and our models allows large scale generation of comprehensive and balanced datasets for the fair evaluation of aggregation algorithms.

3 Problem Description

In this section, we describe our problem formulation of automatic data generation. Our main goal is to generate data that follow closely to the distribution of real one. The advantages of our generated datasets could be evaluated in two perspectives. For common metrics like KL divergence and Kolmogorov–Smirnov test that are often used to measure the distance of two distributions, we hope we will have small KL or KS values when applying to our synthetic datasets compared with the real ones. Besides, since our main motivation is conflicting results of different aggregation algorithms on datasets, we hope these aggregation algorithms will have more consistent performance on our synthetic datasets.

When considering the similarities between our generated datasets and real ones. The smaller the KL or KS values we have the better. Let's define the automatic generation method as function g, g: $\mathbf{Q} \rightarrow \mathbf{Q}'$, where $\mathbf{Q} = \{x_i | i = 1, ..., n\}$ is the real dataset, new data $\mathbf{Q}' = \{x_{i'} | i' = 1, ..., m\}$ is automatically generated from $\mathbf{Q}$, we define the Distribution-oriented Data Generation problem as follows:

Definition 1 (Distribution-oriented Data Generation problem–DDG). *Let's use Q to denote the real dataset, g as a generative function. σ: $\mathbf{R^n} \times \mathbf{R^m} \rightarrow \mathbf{R}$ is a similarity function that measures the distance of two distributions. The problem of DDG is to find a function g^* satisfying:*

$$g^* = \arg\max_g \sigma(Q, g(Q)), \tag{1}$$

when $\mathbf{Q}'$ and $\mathbf{Q}$ are from the same distribution, we get the maximum value for σ, and in this case, we find the true underlying generative model that generates $\mathbf{Q}$. However, this is usually impossible in practice. Thus, we aim to find a good enough generative model in the candidate model set. For example, if we adopt a neural network as our generative model, we use gradient descent to find some local optimum on the training set.

KS and KL could be used to measure the distance of two distributions, but they can not tell us the consistency of the performance of aggregation algorithms on different datasets. To be more specific, we try to answer the question: Is the ranking of different aggregation algorithms regard to accuracy on one dataset the same (or similar) to that on a different dataset? Currently, there are no such metrics we can directly use. In this study, we use subjective judgment and will go deep into the consistency analysis in the experiment section.

4 The TDG4Crowd Method

In this section, we first present our *TDG4Crowd* method, which can separately learn the feature distributions of instances and annotators, as well as the distribution of corresponding annotations. We then discuss the consideration of our design and training strategies for our proposed method.

Our proposed method *TDG4Crowd* is depicted in Figure 1 and it consists of three main modules: 1) *Annotator model* to learn the features of annotators. 2) *Instance model* to learn the features of instances. 3) *Inferring component* to infer the corresponding annotations. Taking a generative approach, we want to learn rich latent representations for both annotators and instances. After obtaining these two latent representations, we could learn the corresponding annotations in a supervised way.

4.1 Learn the Features of Annotators

For a given crowdsourced dataset, we could learn the latent variable z_a of one annotator using the following equation.

$$p(z_a|e_r) = \frac{p(e_r|z_a)p(z_a)}{\int p(e_r|z_a)dz_a}. \tag{2}$$

Here e_r refers to the feature vector of one annotator. Since the integral part of evidence cannot be computed directly, we estimate $p(z_a|e_r)$ by variational inference. We want to find a distribution $q(z_a|e_r)$ that could best approximates $p(z_a|e_r)$. To measure the difference or the distance between two distributions, we can use the Kullback-Leibler divergence.

$$D_{KL}\left[q(z_a|e_r) \| p(z_a|e_r)\right] = E_q[\log \frac{q(z_a|e_r)}{p(z_a|e_r)}]. \tag{3}$$

Use a few rearrangement, we could have the following equation:

$$\begin{aligned} \log p(e_r) =& D_{KL}[q(z_a|e_r) \| p(z_a|e_r)] \\ &+ E_q[\log p(e_r, z_a) - \log q(z_a|e_r)], \end{aligned} \tag{4}$$

since KL divergence is always non-negative, we have

$$\log p(e_r) \geq E_q[\log p(e_r, z_a) - \log q(z_a|e_r)]. \tag{5}$$

The term on the right of the inequality is known as the Evidence Lower Bound, or ELBO for short. We can find a distribution q that minimizes KL divergence by maximizing ELBO. We can rewrite ELBO as follows to make it more clear:

$$ELBO = E_q[\log p(e_r|z_a)] - D_{KL}[q(z_a|e_r) \| p(z_a)]. \tag{6}$$

By maximizing ELBO, the first term on the right side of Equation 6 makes the data generated from z_a as close to the real distribution as possible. Unlike many other models that usually use one latent variable to represent the ability of a worker, we learn a random latent vector z_a based on VAE. The input is a feature vector of one worker e_r. In the simplest case, if there are no other features available, we could use one-hot encoding. We assume the worker latent variable follows a multivariable normal distribution, that is $q(z_a|e_r) \sim N(\mu, \sigma^2)$, thus the output of the encoder are the mean and variance (in implementation, usually use the log of variance for numerical stability reasons) of $q(z_a|e_r)$. We use a standard Gaussian prior acting as a regularizer. $p(z_a) \sim N(0, 1)$.

$$D_{KL}[q(z_a|e_r) \| p(z_a)] = D_{KL}[N(\mu_a, \sigma_a^2) \| N(0, 1)]. \tag{7}$$

For the backpropagation algorithm to work properly, the sampling process uses the reparametrization trick, that is $z_a = \mu_a + \sigma_a \cdot \epsilon$, $\epsilon \sim N(0, 1)$. Thus the actual sampling is from the standard normal distribution. The decoder's output is $\hat{e_r}$ and the reconstruction loss is.

$$\mathcal{L}_{recons} = \|e_r - \hat{e_r}\|^2. \tag{8}$$

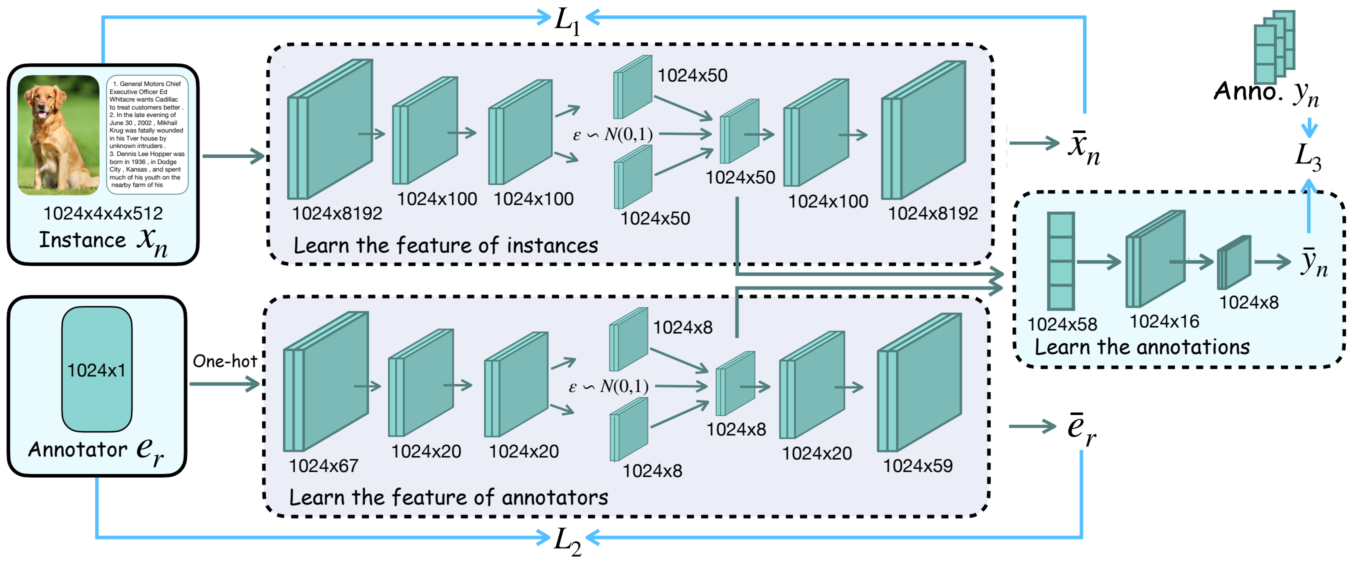
1. General Motors Chief Executive Officer Ed Whitacre wants Cadillac to treat customers better . 2. In the late evening of June 30 , 2002 , Mikhail Krug was fatally wounded in his Tver house by unknown intruders . 3. Dennis Lee Hopper was born in 1936 , in Dodge City , Kansas , and spent much of his youth on the nearby farm of his

Figure 1: The *TDG4Crowd* framework

Therefore, the loss function for annotators can be expressed as:

$$\mathcal{L}_{anno} = \mathcal{L}_{recons} + D_{KL}[N(\mu_a, \sigma_a^2) \| N(0,1)]. \quad (9)$$

The $\mathcal{L}_{recons}$ loss makes the model learn a lower dimensional representation of annotators, while the KL divergence loss makes the representation follow the normal distribution as close as possible. Please note here we are not directly optimizing the KL divergence between the real dataset and the synthetic one.

4.2 Learn the Features of Instances

The process of learning the latent distribution of instances is similar to that of annotators, that is we also learn latent representation for instances based on VAE. But for images, we first use a pre-trained light weight VGG-16 model to extract the features for each instance. Thus, the input for the instance model is the pre-processed feature vector, not raw images. We use x_n to represent the feature vector for instance n. We assume the latent variable for each instance follows a multivariable normal distribution, thus the output of the encoder are the mean and variance of $q(z_i|x_n)$. We use a standard Gaussian prior acting as a regularizer.

$$D_{KL}[q(z_i|x_n) \| p(z_i)] = D_{KL}[N(\mu_i, \sigma_i^2) \| N(0,1), \quad (10)$$

the reconstruction loss is the cross entropy between $\hat{x_n}$ and x_n.

$$\mathcal{L}_{recons} = \|x_n - \hat{x_n}\|^2. \quad (11)$$

The loss function for instance model is similar to that of the annotator model.

$$\mathcal{L}_{inst} = \mathcal{L}_{recons} + D_{KL}[N(\mu_i, \sigma_i^2) \| N(0,1)]. \quad (12)$$

The $\mathcal{L}_{recons}$ loss makes the model learn a lower dimensional representation of instances, while the KL divergence loss makes the representation follow the normal distribution as close as possible.

4.3 Inferring

One of our goals is to learn a model that could generate new annotations. The supervised learning component concatenates the sampling result of z_a and z_i and send it to a fully connected network. The output of this fully connected network is the predicted annotation $\hat{y_n}$ for instance x_n. We train this component in a supervised manner and construct the loss function as follows:

$$\mathcal{L}_{cross} = \|y_n - \hat{y_n}\|^2. \quad (13)$$

The method of *TDG4Crowd* includes *Annotator model*, *Instance model* and *Inferring component*. We first train *Annotator model* and *Instance model* separately until the loss of these two models start to stabilize, then we add in the inferring component and train them as a whole.

5 Experiments

To evaluate our proposed method, we conduct comprehensive experiments on both real and synthetic datasets. In this section, we will first describe the experiment setup. We use annotations collected through the Appen and AMT crowdsourcing platforms. Next, we describe the basesline models used for comparison. Finally, we compare our proposed method with other models for generating new data points. The strategy we use is we regenerate the annotations using the annotators and instances we already have, instead of sampling completely new annotators or instances from the standard normal distribution. That is to say, we use the specific mean and variance for corresponding worker or task for sampling. We use half dataset for training and half for testing and all these regeneration experiments are conducted on the test dataset.

5.1 Setup

We use two real datasets *Labelme* [2] and *Relation* [3] for evaluation. They suffer serious sparsity issues and include instances with rich features that could help infer worker ability. The details of these two datasets can be found in appendix. Similar preprocessing was performed for these two datasets. We use half of them for training our models and half for testing. All the results in the following sections are from this test set. To make sure both the training set and test set have all annotators. We first index the whole dataset using annotators, then we randomly select half instances for each annotator.

We consider three real world scenarios in which why crowdsourced dataset might be imbalanced due to limited budget. 1) One requester may choose to have high redundancy level and publish fewer instances. In this case, we want to generate new instances and corresponding annotations. 2) One requester may choose to have low redundancy but release more instances. Such that we want to generate more annotators and their annotations on these existing instances. 3) The worst case, one requester only have a few instances with low redundancy. We need to generate both new instances and annotators, as well as their annotations. Besides the above three experiments, we also conduct experiments to see how existing aggregation algorithms perform on synthetic datasets. Basically, we try to answer the question: Do these aggregation algorithms act consistently on the synthetic datasets derived from real ones?

5.2 Baseline Models

To evaluate the effectiveness of *TDG4Crowd*, we compare the following methods for data generation with the real dataset.

- ***G_DS***: Derived from the *DS* model [Dawid and Skene, 1979]. When using *DS* for aggregation, the ability of each annotator is estimated according to the EM algorithm. A new annotation is obtained by combining the annotator's ability with ground truth.
- ***G_HDS***: Derived from the *HDS* model [Li and Yu, 2014]. It's a variant of DS. Each annotator is assumed to have the same accuracy on each class of instance, and have the same error probability as well.
- ***G_MV***: This method is based on *MV* model [Sheng, 2011]. In the *MV* model, we regenerate a new annotation based on the probability of voting and the ground truth.
- ***G_GLAD***: Derived from the *GLAD* model [Whitehill *et al.*, 2009]. The annotator ability and the instance difficulty will be obtained using the *GLAD* algorithm for aggregation. We regenerate a new annotation based on these two parameters and the ground truth.
- ***G_IRT***: This approach is based on the *IRT* [Baker *et al.*, 2017]. We use the joint maximum likelihood function to estimate the parameters of the *IRT* model, that is, the ability of the annotator and the difficulty of the instance. Once we have these two parameters, we can regenerate the labels.

[2]https://fprodrigues.com/deep_LabelMe.tar.gz
[3]https://doi.org/10.5281/zenodo.1472330

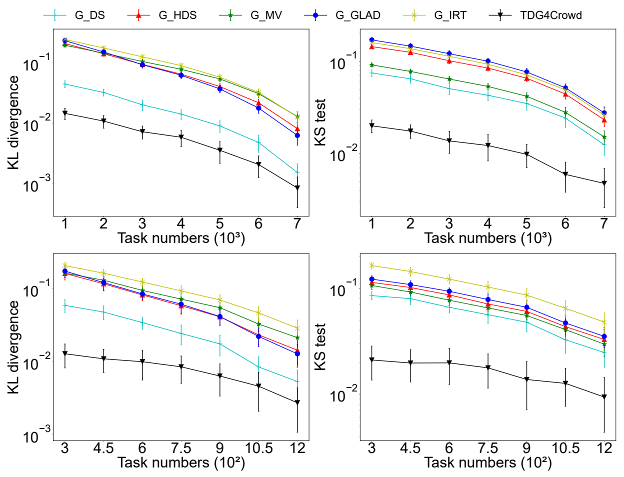

Figure 2: The KL and KS values between the synthetic datasets and real datasets, with the increasing of the percent of real instance numbers. Top: *LabelMe*, Bottom: *Relation*.

- ***TDG4Crowd***: This is the method we propose in this paper. The feature distributions of annotators and instances are respectively learned through the *Annotator model* and *Instance model*, and then the *Inferring component* is used to learn the annotation distribution.

5.3 Results

Generate instances. In this scenario, the requester sets high redundancy level but releases fewer instances. Thus, we want to generate more instances and their annotations. We keep the number of annotators fixed (59 in our case) and generate new annotations on new instances. We want to see if our new generated annotations follow the distribution of the real dataset. We set the size of our synthetic dataset the same as the real dataset. For example, if we have 8000 instances in the real dataset, we first regenerate 7000 instances and their annotations, together with other 1000 real instances and annotations, we get one synthetic dataset and compare it with the real dataset. We then gradually increase the number of real instances (increase by 1000 each time). To evaluate the similarity between the synthetic dataset and the real one, we use KL divergence and Kolmogorov-Smirnov test (KS) as similarity function σ. Both KL and KS can be used to measure the similarity between two distributions. The smaller the value, the more similar these two distributions are. We calculate the KL and KS between the synthetic dataset generated by the above six methods and the real dataset. Since *Labelme* is an eight-category dataset and *Relation* is a thirteen-category dataset, the distribution of each category might be different. Thus, we calculate the KL and KS for each category according to the empirical probability distribution, then weight average all categories. We use the total number of instances in that category as weights in this study and the final results are shown in Figure 2. As we can see from Figure 2, as the percent of real annotations increase in the synthetic dataset, both the KS and KL values decrease. This is true for all data generating methods. More importantly, for

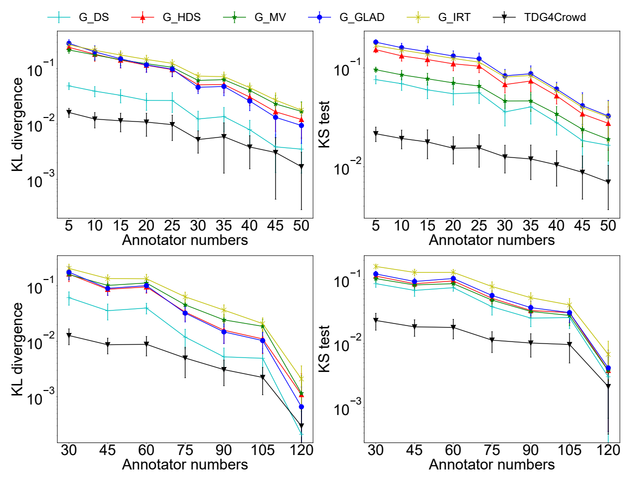

Figure 3: The KL and KS values between the synthetic datasets and real datasets, with the increasing of the percent of real worker numbers. Top: *LabelMe*, Bottom: *Relation*.

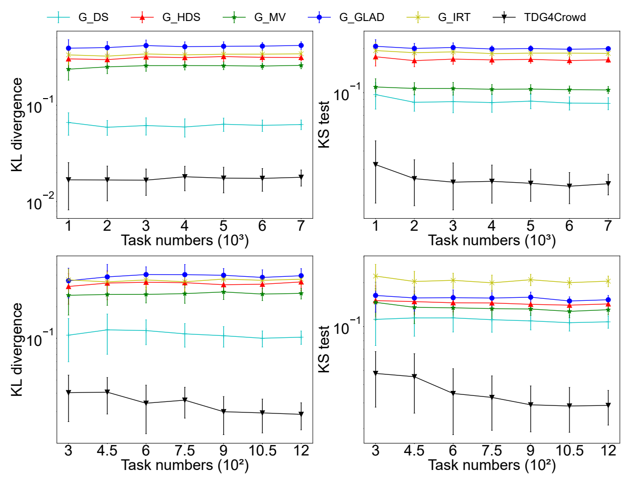

Figure 4: The KL and KS values between the synthetic datasets and real datasets, with the increasing of the number of instances, as well as the number of workers. Top: *LabelMe*, Bottom: *Relation*.

all levels, our proposed method *TDG4Crowd* has the lowest KL and KS values, indicating that the generated annotations using our method follows more closely to the distribution of the real dataset. Among other methods, the performance of *G_DS* comes next. *G_MV* is among the worst performance, this could be explained by the fact that *G_MV* does not have parameters to model the ability of annotators or the difficulty level of instances.

Generate annotators. In this scenario, the requester publishes more instances, but with low redundancy. In this regard, our goal is to generate new annotations for new annotators. Thus, we keep the number of instances fixed, but gradually increase the number of real annotators (increase by 5 each time). Similarly, we use the encoders of VAE modules to get the mean and variance for each annotator and instances. The fully connected network is used to get the annotation predictions. For easy comparison, we set the size of the synthetic dataset and real dataset the same. The results are shown in Figure 3. As we can see the overall trend is similar to the one in Figure 2. With the increase of the percent of real annotators, the values of KL and KS start to decrease, which is not very surprising. Among all these methods, our proposed method *TDG4Crowd* has the best performance, *G_DS* comes next. *G_IRT* and *G_MV* are among the worst performance models.

Generate both instances and annotators. Due to limited budget, one requester may publish a small number of instances with low redundancy level. In such cases, the requester might get a small scale imbalanced dataset. Our goal is then to generate new instances and annotators, as well as the corresponding annotations. In this experiment, we sample annotators and instances through the specific mean and variance to regenerate annotations. We first pick 1000 instances with real annotations, then compare it with the regenerated annotations (from the same instances and corresponding annotators). We repeat the experiment several times, each time we increase the size of instances. Again, we use KL and KS to measure the similarities between the datasets and the results are shown in Figure 4. As we can see from this figure, the size of the instances does not have too much impact on the KS and KL values. In other words, all these methods have steady performance and the performance will not decrease in the scenario of large dataset generation. Among all these methods, our proposed method *TDG4Crowd* has the best performance.

Generate unseen annotations. For all above mentioned experiments, what we did is we regenerate annotations for those that we already have. The reason for doing so is for easy comparison with real dataset. In other words, for a confusion matrix, which is usually very sparse, we did not fill in the holes. In this experiment, we want to see how our model will perform fixing the sparsity issue, which is quite common for crowdsourced datasets. In this setup, we first pick 1000 instances and their corresponding annotators. We use our model to generate new annotations that did not exist before. Then we compare it with the real dataset by calculating the KL divergence and KS values. We repeat the experiment several times, each time we increase the number of instances by 1000 and the results are shown in Figure 5. We can basically get the same conclusion as from Figure 4. The overall trend is quite stable, which means the data size does not have too much impact on the performance of data generation models. Among all these methods, our proposed method *TDG4Crowd* has the best performance.

Aggregation consistency. The main motivation of our work is that aggregation algorithms like *DS*, *HDS*, *MV*, *GLAD* perform inconsistently on different datasets. In Figure 6, we show the rankings of different aggregation algorithms on real datasets with different scales. We randomly sample different number of instances and we can see, for the *Labelme* dataset, when the number of instances is 1000, *HDS* performs the best, then comes *GLAD* and *MV*, *DS* performs the worst. When we gradually increase the number of instances, we can see how the rankings change. When

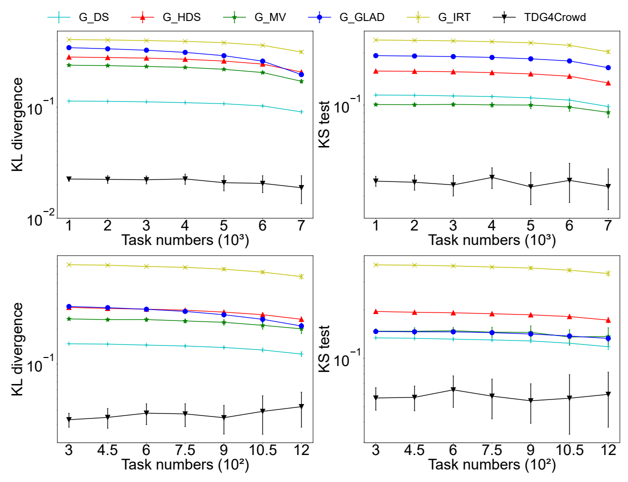

Figure 5: The KL and KS values between the synthetic datasets and real datasets. The synthetic datasets include those annotations that do not exist. Top: *LabelMe*, Bottom: *Relation*.

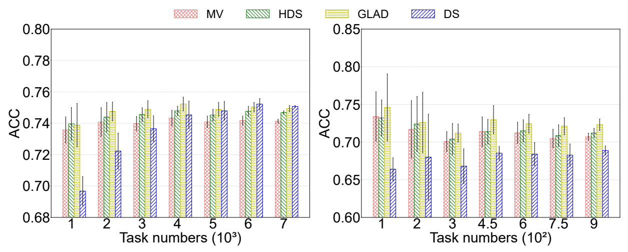

Figure 6: Accuracy of different aggregation algorithms on different size of real datasets. Left: *LabelMe*, Right: *Relation*.

we increase the number of instances to around 6000, the rankings start to stabilize. At this time, *DS* performs the best, then comes the *GLAD* and *HDS*, *MV* performs the worst. We have similar observations for the *Relation* dataset. We argue a comprehensive and balanced dataset is crucial for the assessment of these aggregation algorithms. Figure 7 shows the performance of different aggregation algorithms on the synthetic datasets (derived from the *LableMe* dataset) with different proportion of real data. In this synthetic dataset, we generate unseen annotations for existing annotators and instances, and we gradually increase the number of real instances in this synthetic dataset. We can see when there are only 1000 real instances, the performance of these aggregation algorithms on different synthetic dataset differ dramatically and the rankings of these aggregation algorithms are also different from the real dataset. This is especially ture for the datasets generated using *G_DS*, *G_HDS*, *G_GLAD* and *G_IRT*. Algorithms on these datasets have much higher accuracy than those on the real dataset. With the increase of the percent of real data on these synthetic datasets, the performance of aggregation algorithms start to get close to that on the real dataset, and the rankings also start to converge to the ranking on the real dataset. We have similar observations for other experiments like only generating

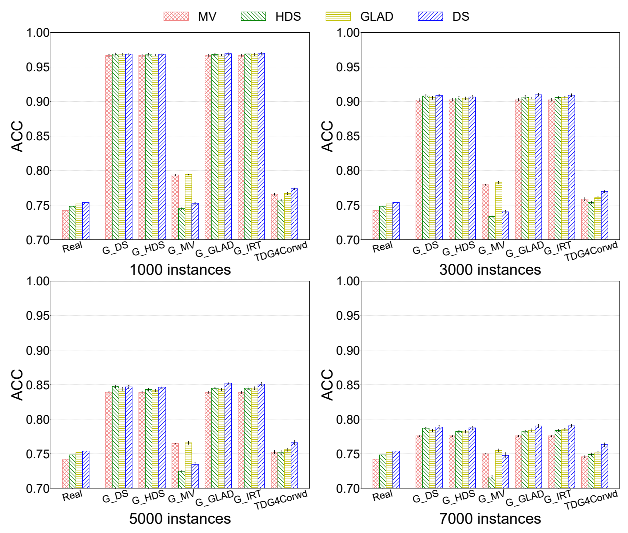

Figure 7: The performance of different aggregation algorithms on synthetic datasets for generating unseen annotations (*LabelMe*)

instances, only generating annotators or generating instances and annotators. These experiments results could be found in the appendix.

6 Conclusions

In this paper, we first discuss the problem crowdsourced datasets may have and how they can fail the evaluation of different aggregation algorithms. We then propose the *TDG4Crowd* method for the automatic generation of comprehensive, balanced data and demonstrate how *TDG4Crowd* could be used to solve the data issues caused due to budget constraints under different real world scenarios. We conduct extensive experiments on image and sentence datasets to show the effectiveness of our proposed method. By calculating the KL divergence and KS stats between the synthetic dataset and the real one, we show that our method is able to generate dataset that follows more closely to the distribution of real ones, compared with other baseline methods. The accuracy rankings of existing aggregation algorithms are calculated on real datasets and synthetic ones. The experimental results also show that the synthetic data generated by *TDG4Crowd* is more consistent with the real data from the ranking perspective. In this study, we use simple annotation tasks. For future work, we will investigate more challenging tasks that involve context information.

Acknowledgments

This research has been supported by the National Nature Foundation of China under grant 61976187, the Natural Science Foundation of Zhejiang Province under grant (LZ22F020008, LQ22F020002, LY20F030002) and Zhejiang Gongshang University ”Digital+” Disciplinary Construction Management Project (No SZJ2022A001).

References

[Albarqouni *et al.*, 2016] Shadi Albarqouni, Christoph Baur, Felix Achilles, Vasileios Belagiannis, Stefanie Demirci, and Nassir Navab. Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE transactions on medical imaging*, pages 1313–1321, 2016.

[Baker *et al.*, 2017] Frank B Baker, Seock-Ho Kim, et al. *The basics of item response theory using R*. Springer, 2017.

[Dawid and Skene, 1979] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, pages 20–28, 1979.

[Fang *et al.*, 2018] Yili Fang, Hailong Sun, Pengpeng Chen, and Jinpeng Huai. On the cost complexity of crowdsourcing. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1531–1537, 2018.

[Galal and El-Sharkawi, 2019] Sh Galal and Mohamed E El-Sharkawi. Trr: Reducing crowdsourcing task redundancy. In *Database and Expert Systems Applications - 30th International Conference (DEXA)*, pages 102–117, 2019.

[Gao *et al.*, 2016] Chao Gao, Yu Lu, and Dengyong Zhou. Exact exponent in optimal rates for crowdsourcing. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, pages 603–611, 2016.

[Heinecke and Reyzin, 2019] Shelby Heinecke and Lev Reyzin. Crowdsourced pac learning under classification noise. In *Proceedings of the 7th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, pages 41–49, 2019.

[Ho *et al.*, 2013] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 534–542, 2013.

[Imamura *et al.*, 2018] Hideaki Imamura, Issei Sato, and Masashi Sugiyama. Analysis of minimax error rate for crowdsourcing and its application to worker clustering model. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 2152–2161, 2018.

[Jing *et al.*, 2018] Jing, Zhang, Victor, S, Sheng, Tao, Li, Xindong, and Wu. Improving crowdsourced label quality using noise correction. *IEEE Transactions on Neural Networks Learning Systems*, pages 1675–1688, 2018.

[Kawase *et al.*, 2019] Yasushi Kawase, Yuko Kuroki, and Atsushi Miyauchi. Graph mining meets crowdsourcing: Extracting experts for answer aggregation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1272–1279, 2019.

[Li and Yu, 2014] Hongwei Li and Bin Yu. Error rate bounds and iterative weighted majority voting for crowdsourcing. *CoRR*, abs/1411.4086, 2014.

[Li *et al.*, 2013] Hongwei Li, Bin Yu, and Dengyong Zhou. Error rate bounds in crowdsourcing models. *CoRR*, abs/1307.2674, 2013.

[Liu *et al.*, 2012] Qiang Liu, Jian Peng, and Alexander Ihler. Variational inference for crowdsourcing. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012 (NIPS)*, pages 701–709, 2012.

[Long and Hua, 2016] Chengjiang Long and Gang Hua. Multi-class multi-annotator active learning with robust gaussian process for visual recognition. *International Conference on Computer Vision (ICCV)*, 2016.

[Mescheder *et al.*, 2017] Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2391–2400, 2017.

[Pan *et al.*, 2023] Yigong Pan, Ke Tang, and Guangzhong Sun. Theoretical guarantee for crowdsourcing learning with unsure option. *Pattern Recognition*, page 109316, 2023.

[Rodrigues and Pereira, 2018] Filipe Rodrigues and Francisco C. Pereira. Deep learning from crowds. In *AAAI conference on artificial intelligence*, pages 1611–1618, 2018.

[Roy *et al.*, 2015] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. *The VLDB Journal*, pages 467–491, 2015.

[Russell *et al.*, 2008] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. *International journal of computer vision*, pages 157–173, 2008.

[Sheng *et al.*, 2008] Victor S. Sheng, Foster J. Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 614–622, 2008.

[Sheng, 2011] Victor S Sheng. Simple multiple noisy label utilization strategies. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM)*, pages 635–644, 2011.

[Sinha *et al.*, 2018] Vaibhav B Sinha, Sukrut Rao, and Vineeth N Balasubramanian. Fast dawid-skene: A fast vote aggregation scheme for sentiment classification. *arXiv preprint arXiv:1803.02781*, 2018.

[Snow *et al.*, 2008] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 conference on empirical methods in natural language processing (EMNLP)*, pages 254–263, 2008.

[Wang and Zhou, 2015] Wei Wang and Zhi-Hua Zhou. Crowdsourcing label quality: a theoretical analysis. *Science China Information Sciences*, pages 1–12, 2015.

[Wang *et al.*, 2013] Jiannan Wang, Guoliang Li, Tim Kraska, Michael J. Franklin, and Jianhua Feng. Leveraging transitive relations for crowdsourced joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 229–240, 2013.

[Wauthier and Jordan, 2011] Fabian L. Wauthier and Michael I. Jordan. Bayesian bias mitigation for crowdsourcing. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems 2011 (NIPS)*, pages 1800–1808, 2011.

[Whitehill *et al.*, 2009] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems 2009 (NIPS)*, pages 2035–2043, 2009.

[Yu *et al.*, 2017] Han Yu, Chunyan Miao, Yiqiang Chen, Simon Fauvel, Xiaoming Li, and Victor R Lesser. Algorithmic management for improving collective productivity in crowdsourcing. *Scientific reports*, page 12541, 2017.

[Zhang *et al.*, 2016a] Jing Zhang, Xindong Wu, and Victor S. Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, pages 543–576, 2016.

[Zhang *et al.*, 2016b] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *Journal of Machine Learning Research*, pages 1–44, 2016.

[Zhang *et al.*, 2019] Hao Zhang, Liangxiao Jiang, and Wenqiang Xu. Multiple noisy label distribution propagation for crowdsourcing. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1473–1479, 2019.