Computing Crowd Consensus with Partial Agreement

Nguyen Quoc Viet Hung [*], Huynh Huu Viet [*], Nguyen Thanh Tam [#], Matthias Weidlich [†], Hongzhi Yin [*], Xiaofang Zhou [*]

[#] *École Polytechnique Fédérale de Lausanne* [*] *The University of Queensland* [†] *Humboldt-Universität zu Berlin*

Abstract—Crowdsourcing has been widely established as a means to enable human computation at large-scale, in particular for tasks that require manual labelling of large sets of data items. Answers obtained from heterogeneous crowd workers are aggregated to obtain a robust result. However, existing methods for answer aggregation are designed for *discrete* tasks, where answers are given as a single label per item. In this paper, we consider *partial-agreement* tasks that are common in many applications such as image tagging and document annotation, where items are assigned sets of labels. Common approaches for the aggregation of partial-agreement answers either (i) reduce the problem to several instances of an aggregation problem for discrete tasks or (ii) consider each label independently. Going beyond the state-of-the-art, we propose a novel Bayesian nonparametric model to aggregate the partial-agreement answers in a generic way. This model enables us to compute the consensus of partially-sound and partially-complete worker answers, while taking into account mutual relationships in labels and different answer sets. We also show how this model is instantiated for incremental learning, incorporating new answers from crowd workers as they arrive. An evaluation of our method using real-world datasets reveals that it consistently outperforms the state-of-the-art in terms of precision, recall, and robustness against faulty workers and data sparsity.

Index Terms—Crowdsourcing, Nonparametric Models, Bayesian Models, Answer Aggregation.

1 Introduction

Fuelled by the massive availability of Internet users, crowdsourcing has been widely established as a means for human computation at large-scale [1]. Tasks that are rather trivial for humans, but computationally expensive or even unsolvable for machines can be efficiently addressed by crowdsourcing. Specifically, crowdsourcing has been applied for such diverse applications as data acquisition, data integration, data mining, information extraction, and information retrieval [2], [3], [4], [5], [6]. Today, a large number of platforms, such as Amazon Mechanical Turk and CrowdFlower, facilitate the development of crowdsourcing applications.

Aggregation of Crowd Answers. Most crowdsourcing setups are based on questions (aka tasks) that, once posted to a crowdsourcing platform, are answered by users (aka crowd workers) for financial rewards. Yet, each task is answered by multiple workers to accommodate for their different levels of expertise and motivation. Aggregation of answers shall complement individual errors, thereby exploiting the 'wisdom of the crowd'.

Answer aggregation is challenging for several reasons. The worker population may contain faulty workers (e.g., spammers) that give random answers, but are hard to identify before-hand in the absence of detailed worker information. Furthermore, workers may be unintentionally biased by personal interest or systematic misunderstanding of the tasks [7]. Aggregation of answers is also complicated by limited mutual information between workers and tasks, e.g., some workers are assigned with too few tasks and vice-versa [7]. To overcome these challenges, various methods for automatic answer aggregation have been proposed in the literature (see [8], [9] for a survey), including (i) non-iterative techniques which compute the aggregated answer as a linear combination of votes, and (ii) iterative techniques which leverage mutual reinforcing relations between workers and answers.

Aggregation with Partial Agreement. Depending on the type of questions, different types of crowdsourcing tasks are distinguished: In *continuous* tasks, objects are assigned real values (e.g., scores) [10]. In *partial-agreement* tasks, which define objects as rules [2] or evaluate matches/orderings between entities [11], [12].

In this paper, we focus on a special type of *partial-agreement* tasks, where workers shall provide a set of labels per item. Such tasks, also known as *multi-label* tasks, received much attention recently in many crowdsourcing applications, such as text categorization, image classification, and medical diagnosis [13], [14]. We use the term *partial-agreement* to distinguish the respective tasks from *multi-class, single-label* tasks [15], [16], in which tasks offer multiple choices/labels, but workers only give a single answer/label. Rather, the partial-agreement tasks considered in this work are a generalization of multi-label tasks [17], [18], [19], which usually decompose a multi-label problem into several instances of a single-label problem, having each worker giving a Boolean answer for each possible label.

The aforementioned aggregation methods have been developed for single-label tasks only [8], [9] and are often extended to solve the single-label instances of multi-label tasks. Only a few methods truly target the multi-label setting, typically based on some form of majority voting [17], [18] or by reusing worker information across several instances of an aggregation problem [19]. Yet, these methods still consider each label separately.

Due to the freedom to choose multiple labels per item, partial-agreement tasks allow a more fine-grained level of aggregation. Worker answers are now partially-sound (some given labels are incorrect) and partially-complete (some correct labels are missing). This is in contrast to the single-label setting, where answers are either correct or incorrect. It also goes beyond the predominant approach to trace back answer aggregation for multi-label tasks to several instances of single-label aggregation, since, in these cases, not providing a label is implicitly taken as a negative answer. By exploiting the partial agreement of answers, however, correctness and completeness of answer aggregation can be improved.

Beside the general challenges of answer aggregation, computing crowd consensus with partial-agreement is inherently complex. The labels obtained as part of different answers are often correlated. For example, in movie classification, movies about a *superhero* are often associated with the genre *action* [20]. Identifying the correct set of labels needs to deal with the exponential growth of combinations of labels and dependencies between them. Also,

workers no longer either agree or disagree on an answer to a question, but consensus becomes partial. Hence, it becomes difficult to assess the reliability of workers since they may provide supposedly correct and incorrect labels at the same time.

Approach. In this paper, we propose a Bayesian nonparametric model in order to capture the distinct properties of partial-agreement answer aggregation. That is, co-occurrence dependencies between labels are represented by the notion of latent label clusters. This notion is motivated by the observation that items can often be grouped together, if they share similar labels. Furthermore, partial consensus between workers is modelled by grouping together workers with similar answers. This enables us to construct an aggregated answer based on the consensus of groups of workers instead of consensus among individuals. The resulting model generalises the multi-label setting of answer aggregation and enables incremental learning using the principles of stochastic variational inference.

Contribution and Structure. Our contributions along with the structure of the paper can be summarized as follows:

Problem Setting (§2) We motivate the need for partial-agreement answer aggregation and elaborate on types of crowd workers. We further formalize the problem setting, and outline requirements for partial-agreement answer aggregation.

Novel Model for Partial-Agreement Answer Aggregation (§3) We present a generative model for partial-agreement answer aggregation, referred to as *Generic Crowdsourcing Consensus with Partial Agreement (CPA)*. Specifically, we show how to perform model inference (finding the probability distribution of model parameters given information on worker answers or true labels) and model instantiation (estimating item labels based on the given information and the inferred parameter distributions).

Scalable Model Inference and Prediction (§4) Even a linear model inference algorithm becomes intractable for very large datasets. Therefore, we propose (i) to exploit incremental computation, so that model parameters are updated based on new data instead of inferring a model from scratch; (ii) to leverage parallelisation to scale-out and scale-up model inference and prediction.

Evaluation (§5) Experiments with real-world datasets highlight the effectiveness of the proposed CPA model. It outperforms baseline methods by up to 134% in precision and recall. Also, when using our methods for scalable model inference, we observe speed-ups of up to 32× in runtime.

Finally, §6 reviews related work, before §7 concludes the paper.

2 Problem Setting

We first introduce a motivating example for partial-agreement answer aggregation and elaborate on challenges induced by different types of crowd workers. Then, we present a problem statement and discuss requirements for potential solutions.

2.1 Motivating Example

We consider an image tagging task, in which workers assign one or more labels to a picture. For simplicity, these labels are encoded by numbers from 1 to 5. Table 1 illustrates an exemplary crowdsourcing result, in which five workers (u_1 - u_5) provided their answers to four pictures (i_1 - i_4). The correct, yet generally unknown, label assignment is shown in a separate column.

TABLE 1: Answers provided by five workers for four pictures.

	u_1	u_2	u_3	u_4	u_5	Correct	Majority [17], [18]
i_1	{4,5}	{4,5}	{4}	{1}	{5}	{5}	{4,5}
i_2	{2,3}	{1,4}	{4}	{2}	{3,4}	{3,4}	{4}
i_3	{1,2}	{4}	{4}	{3}	{4,5}	{4,5}	{4}
i_4	{1,2}	{2,3}	{4}	{4}	{1,2,3}	{1,2,3}	{2}

1: sky, 2: plane, 3: sun, 4: water, 5: tree

Answer aggregation calculates a joint answer for each item based on the input provided by workers. A common method to derive an aggregated answer is majority voting [17], [18], which considers all labels separately. If the ratio of 'votes' from workers for a label is larger than 0.5, the respective label is included in the aggregation result. Compared to the actually correct assignment, the result obtained in this case has two issues, though: (i) it is partially incorrect (label 4 is not correct for i_1), and (ii) partially incomplete (labels 1 and 3 shall also be assigned to i_4).

These issues have two main causes. First, workers are considered equally, whereas it is known that there are different types of crowd workers [21], [8]: (1) Reliable workers have deep knowledge about specific domains and provide correct answers; (2) Normal workers tend to give correct answers, but make mistakes occasionally; (3) Sloppy workers have little knowledge and often give wrong answers unintentionally; (4) Uniform spammers intentionally answer all question the same; (5) Random spammers give random answers. In our example, u_3 is a uniform spammer, assigning one label to all pictures. Yet, these answers are reflected in aggregated result, while removing u_3 yields the correct result for picture i_1. Worker u_4 is a random spammer, whereas the remaining workers can be classified as reliable (u_5) or normal (u_1 and u_3).

However, existing worker models [21], [22], [23], [24] cannot be applied directly in partial-agreement answer aggregation, since worker answers may be partially overlapping. Moreover, interpreting a missing label as a negative answer is not always correct and thus shall be cross-checked by answers from other workers.

A second cause for the issues observed in the example is the neglect of dependencies between labels. For instance, label 2 often co-occurs with labels 3 and 1. Such correlation can be useful in the aggregation. If we also include label 1 and label 3 whenever label 2 has been assigned, for instance, the obtained result would be correct for picture i_4 when using majority voting.

2.2 Problem Statement

We capture the setting of partial-agreement answer aggregation by a set of workers $\mathcal{U}$, identified by their indices, $\mathcal{U} \triangleq \{1,\dots,U\}$ that provide answers for a set of items $\mathcal{N}$, also identified by their indices, $\mathcal{N} \triangleq \{1,\dots,I\}$. $\mathcal{Z} \triangleq \{1,\dots,C\}$ is the set of all possible labels for these items. Each answer by a crowd worker is a subset of $\mathcal{Z}$. Formally, answers are modelled as an $I \times U$ *answer matrix*:

$$\mathcal{M} \triangleq \begin{pmatrix} x_{11} & \dots & x_{1U} \\ \dots & \dots & \dots \\ x_{I1} & \dots & x_{IU} \end{pmatrix}$$

where $x_{iu} \subseteq \mathcal{Z}$ is the set of labels assigned to item i by worker u, or $x_{iu} = \emptyset$ if worker u has not provided an answer for item i.

Problem 1 (Partial-Agreement Answer Aggregation). Given a set of items $\mathcal{N}$, a set of workers $\mathcal{U}$, a set of labels $\mathcal{Z}$, and an answer matrix $\mathcal{M}$, the problem of partial-agreement answer aggregation is the construction of a *deterministic assignment* $d : I \rightarrow 2^{\mathcal{Z}}$ assigning a set of labels to each item.

A baseline solution to the above problem is to construct the assignment d by majority voting, as illustrated above. Yet, observing the issues that stem from the application of majority voting, we derive requirements to be met by answer aggregation in order to be useful for answers of partial-agreement tasks.

(R1) Consideration of worker communities. In practice, there is little control over the selection of crowd workers. Answer aggregation, thus, shall capture worker characteristics, to assess the likelihood of them providing correct answers and to justify their effects in the aggregated result. The existence of different worker types, as illustrated above, has been verified in various studies [25], [26] as well as in our experimental evaluation.

(R2) Support for partial answer validity. Against the background of diverse worker types and their distribution in practice, see [27], the correctness of answers shall be assessed in a fine-granular manner, i.e., at the level of individual labels. This is a prerequisite to make efficient use of normal and sloppy workers in particular. In the above example (Table 1), u_2 may be an expert for label 4 (two out of three assignments are correct), but potentially lacks knowledge related to label 1 (one incorrect and one missing assignment). The existence of worker communities for different labels can indeed be observed in practice (see §5.5).

(R3) Exploitation of label dependencies. In many multi-label settings, similar items are assigned overlapping sets of labels. Such dependencies between labels, e.g., their co-occurrence in the answers provided by crowd workers, shall be exploited to improve the soundness and completeness of answer aggregation. As an example, Fig. 1 illustrates five representative labels of a multi-label dataset (discussed in detail in our experimental evaluation) along with their co-occurrence dependencies. We observe two clusters of labels, {*sky, birds, cloud*} and {*flower, road*}. Such relations between labels represent domain knowledge that can be explored to characterize workers on a fine-granular level. Moreover, for automatic classification it is well-known that the neglect of label dependencies—treating a multi-label problem as several instances of a single-label problem—yields weak results [28], [29]. Moreover, asking workers to confirm each label separately would incur higher time and cost. While there are other types of label dependencies, such as hierarchies [18], those require domain-specific knowledge to be configured appropriately. Clusters can be considered as the most generic form of label dependencies, since workers may still use the raw labels, rather than abstract labels that represent a group of raw labels [30], [13].

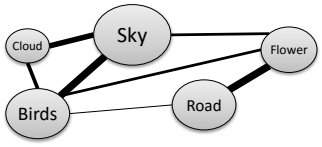


Fig. 1: Co-occurrence of labels in the NUS-WIDE dataset [13]; vertex sizes represent the occurrence cardinality and edge weights represent the strength of the co-occurrence dependency.

(R4) Adaptivity of aggregation model. The characteristics of a crowdsourcing application (e.g., the number of worker communities) may vary across different domains or over time, upon the arrival of new data. This requires dynamic adaptation of the model to reflect the evolving relations between the obtained answers. Again, the practical relevance of this requirement is illustrated by our experiments in §5.5, which illustrate the diversity of worker communities across real-world datasets. Dynamic adaptation of the aggregation model is needed to cope with such diversity.

3 Partial-agreement Answer Aggregation in Crowdsourcing

This section introduces our novel model for partial-agreement answer aggregation, referred to as *Generic Crowdsourcing Consensus with Partial Agreement* (CPA). We first give an intuitive overview of the model, before we turn to its formalisation. Then, we outline the application of CPA for answer aggregation: we derive a scalable inference method with variational Bayes and show how to predict the labels of non-grounded items.

3.1 Overview of the Approach

We approach answer aggregation by considering each element of the given answer matrix as an observed random variable. The true labels of items are also modelled as a random variable. While a few of them may be observed (e.g., due to test questions [31]), the vast majority of these variables are unobserved. To predict the value of unobserved variables, i.e., to estimate the labels for an item for which the true labels are not available, we rely on a generative process based on a Bayesian network. All random variables are generated from parametrised probability distributions and the respective parameters are inferred from the observed variables. Here, worker communities are represented by a clustering of workers, modelled nonparametrically by a Chinese Restaurant Process. Then, values of the unobserved variables are predicted.

We address the outlined requirements regarding the consideration of worker communities (R1) by a notion of worker clusters that group together workers based on their trustworthiness and domain knowledge. In contrast to methods that evaluate individual workers, e.g., by means of confusion matrices [32], models that rely on clusters of workers are less prone to errors when data is sparse. This makes them particularly suited for crowdsourcing where each item is processed only by a fraction of the worker population due to budget constraints.

Since our approach is grounded in a nonparametric model, the number of parameters is adjusted to the data, thereby enabling adaptive aggregation (R4). The Bayesian property of the model helps to reduce over-fitting by inferring probability distribution over random variables instead of singleton values. In addition, Bayesian models are well suited to cope with online settings—new information can be encoded into posterior distributions used in the inference and prediction process.

The specific challenges of answer aggregation for partial-agreement tasks are addressed as follows. Dependencies between labels (R3) are incorporated by clustering items in the answer aggregation process. Items in a cluster are assumed to be similar and, thus, be assigned the same set of labels. The latter implicitly encodes dependencies between labels in terms of co-occurrence.

To support partial answer validity (R2), we follow the intuition that obtaining a label for an item can be seen as randomly selecting labels of the respective item cluster, given a worker community. Hence, we model the labels as being generated from a multinomial distribution over the item clusters and worker communities. Since this is a random process, workers in a community may still provide different labels for items of the same cluster.

3.2 The Model of CPA

The input of partial-agreement answer aggregation (Problem 1) is a set of items $\mathcal{N} \triangleq \{1,\dots,I\}$, a set of workers $\mathcal{U} \triangleq \{1,\dots,U\}$, a set of labels $\mathcal{Z} \triangleq \{1,\dots,C\}$, all identified by the indices of their elements, and an answer matrix $\mathcal{M}$. We define the model of Generic Crowdsourcing Consensus with Partial Agreement (CPA) as follows (notations are summarised in Table 2). All non-empty answers in $\mathcal{M}$ are modelled as observed variables $\boldsymbol{x} \in (2^{\mathcal{Z}})^{I\times U}$, where x_{iu} denotes the set of labels assigned to item i by worker u. Further, $y \in (2^{\mathcal{Z}})^{I}$ are random variables modelling the true labels of all items. True labels may be known for some items, which is captured by a set of observed variables $\boldsymbol{y} \subseteq y$ ($\boldsymbol{y}$ may be empty). The values of variables $\boldsymbol{x}$ and $\boldsymbol{y}$ can be represented as a C-dimensional vector, such that each of its components is set to one, if the respective label is present. Thus, observed values of $\boldsymbol{x}$ and $\boldsymbol{y}$ can be seen as samples from a multinomial distribution.

Worker communities, item clusters, and label selection are modelled as follows (Fig. 2 shows a graphical representation):

Worker Communities. There is a finite set of worker communities π, identified by indices, $\pi \triangleq \{1,\dots,M\}$, that partition the set of workers and are not known in advance. The (unknown) assignment of workers to communities is captured by a set of random variables $z \in \pi^{U}$, such that z_u denotes the community of worker u. We generate π nonparametrically using a Chinese Restaurant Process (CRP), which can be interpreted as the induced distribution over the partition space by a Dirichlet Process [33], [24]. Technically, if π follows a CRP distribution, $\pi \sim \mathrm{CRP}(\alpha)$, the samples from this prior follow the following distributions:

$$p(z_u = m \mid z_{-u}, \pi, \alpha) \propto \begin{cases} n_m^{-u} & \text{if } \exists\, z_{u'} \in z_{-u} : z_{u'} = m \\ \alpha & \text{otherwise} \end{cases}$$

where $z_{-u} = z \setminus \{z_u\}$ and n_m^{-u} is the number of elements in z_{-u} with community m. Generatively, π can be constructed using a stick-breaking process as follows. Let π'_m, $m = 1,2,3,\dots$ be sampled from a Beta distribution $\mathrm{Beta}(1,\alpha)$. Then, the community proportion π_m is calculated using the above sticks π'_m, such that

$$\pi_1 = \pi'_1,\ \dots\ , \pi_m = \pi'_m \prod_{j=1}^{m-1} \left(1-\pi'_j\right), \cdots \tag{1}$$

When conducting inference, we will only estimate the stick distribution π', since the original distribution π is calculated directly from π' as above. Note that the nonparametric approach generalises the extreme cases. If M tends to infinity, each worker is a single community (e.g., no two workers provide similar answers). If M tends to zero, all workers form a single community (e.g., only expert workers) and the result is similar to majority voting.

TABLE 2: Overview of notations.

$\mathcal{N}$	Set of I items, identified by indices, $\mathcal{N} \triangleq \{1,\dots,I\}$
$\mathcal{U}$	Set of U workers, identified by indices, $\mathcal{U} \triangleq \{1,\dots,U\}$
$\mathcal{Z}$	Set of C possible labels, identified by indices, $\mathcal{Z} \triangleq \{1,\dots,C\}$
$\mathcal{M}$	$I \times U$ Answer matrix
π	Set of M worker communities, identified by indices, $\pi \triangleq \{1,\dots,M\}$
τ	Set of T item clusters, identified by indices, $\tau \triangleq \{1,\dots,T\}$
z_u	Community of worker u
x_{iu}	Labels assigned to item i by worker u
ψ^t_m	Label assignment probabilities of workers in community m for items in cluster t
l_i	Cluster of item i
y_i	True labels assigned to item i
ϕ_t	Label assignment probabilities for items in cluster t

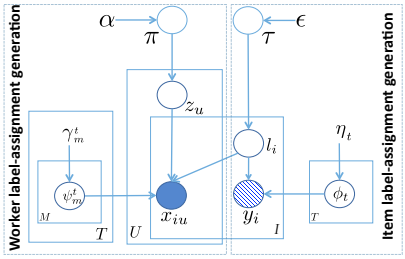


Fig. 2: Graphical representation of the CPA model.

Item Clusters. To model clusters of similar items, which tend to get assigned the same sets of labels, we follow the approach introduced for worker communities. There is a finite set τ of clusters, identified by indices, $\tau \triangleq \{1,\dots,T\}$, that partition the set of items and are not known in advance. The (unknown) assignment of items to these clustered is captured by random variables $l \in \tau^{I}$, such that l_i denotes the cluster of item i. Again, τ is generated nonparametrically by a CRP, i.e., $\tau \sim \mathrm{CRP}(\varepsilon)$. This avoids additional efforts to ask for expert knowledge on the label information [34], [35]. Although it is beyond the scope of this paper, our model can be extended by expressing the prior knowledge as conditional probabilities and integrating them into the generative label selection of our model.

The assignment of sets of labels to item clusters is modelled as a generation from a multinomial distribution. For cluster t, this distribution is parameterised by $\phi_t \triangleq \{\phi_{t,1},\dots,\phi_{t,C}\}$, where $\phi_{t,c}$ is the probability that a given item in cluster t will have the label c. Items in a cluster may have different true labels as a result of the generating random process—yet, being in the same cluster, they are similar and thus share the labelling probabilities.

Label Selection. We model the labels obtained from workers as being generated from a multinomial distribution over the labels of an item cluster, given a worker community. Each worker is characterised by a $C \times T$ confusion matrix ψ_m, where m is the community that the worker belongs to. We denote by ψ^t_m a column vector of C-dimensions, which contains the probabilities that a worker in community m assigns the respective labels given an item of cluster t. This model has the advantage that, instead of considering exponentially many subsets of labels, it is based on the number of all possible item clusters, which is tractable in practice.

Consider a setting with four labels {1: girl, 2: boy, 3: dog, 4: cat}, two item clusters {1: people, 2: animal}, and two worker communities {1: trustworthy, 2: problematic}. Then, a confusion matrix column vector $\psi^1_1 = [0.5, 0.5, 0, 0]$ means that workers of community 1 (trustworthy) assign an item of cluster 1 (people) the labels 1 (girl), 2 (boy), 3 (dog), or 4 (cat) with probabilities 0.5, 0.5, 0, or 0, respectively.

Generative Process. Let Cat and Multi be categorical and multinomial distributions, respectively. Then, the generative process for the CPA model is defined as follows:

(1) For each item in $\mathcal{N}$ (right-hand side of Fig. 2):
   a) Generate the cluster for each item:
      $l_i \mid \tau \sim \mathrm{Cat}(\tau)$
   b) Generate the labels for each item from the cluster:
      $y_i \mid l_i, \phi \sim \mathrm{Mult}(\phi_{l_i})$

(2) For each worker in $\mathcal{U}$ (left-hand side of Fig. 2):
   a) Generate the community for each worker:
      $z_u \mid \pi \sim \mathrm{Cat}(\pi)$
   b) Generate the set of assigned labels for each worker and

item from the labels of the item cluster and the confusion matrix of the worker's community:
$x_{iu} \mid z_u, l_i, \psi \sim \text{Mult}\left(\psi_{z_u}^{l_i}\right)$

Model Parameters. The CPA model is nonparametric since the number of worker communities in π and the number of item clusters in τ are not known in advance—they change with more observations ($\boldsymbol{x}$ and $\boldsymbol{y}$) becoming available. In particular, they grow when more and more different workers and items are processed, avoiding any bias potentially introduced by fixing these parameters before-hand.

In a Bayesian setting, we use the following priors for the parameters related to the worker communities and item clustering (Dir being a Dirichlet distribution):

$$\pi \sim \text{CRP}(\alpha) \qquad \psi_m^t \sim \text{Dir}(\gamma_m^t)$$
$$\tau \sim \text{CRP}(\varepsilon) \qquad \phi_t \sim \text{Dir}(\eta_t)$$

with $1 \leq t \leq T$ and $1 \leq m \leq M$. Here, T and M are the maximal number of worker communities and item clusters for numerical purposes in later model inference, which can safely be set to large values, e.g., 1000. Both τ and π are unknown.

3.3 Inference

Inferring the parameters of the CPA model is, in fact, the estimation of values of the above priors $(\alpha, \varepsilon, \gamma, \eta)$. This is equivalent to inferring the posterior distribution of the unobserved variables $(\pi, \tau, z, l, \psi, \phi)$ under the observed variables $(\boldsymbol{x}, \boldsymbol{y})$, which is $p(\pi, \tau, z, l, \psi, \phi \mid \boldsymbol{x}, \boldsymbol{y})$, or $p(\pi', \tau', z, l, \psi, \phi \mid \boldsymbol{x}, \boldsymbol{y})$ using the stick-breaking representation for π and τ (see Eq. 1).

Approaches for (approximate) inference for statistical models can be classified into simulation methods and deterministic variational methods. The use of simulation such as Markov Chain Monte Carlo algorithms (such as Gibbs sampling and random walk) is problematic when applied to large-scale data sets since convergence is often slow and unpredictable [36], [37]. Thus, we resort to variational inference and propose a novel scalable method that follows the principles of variational Bayesian inference.

In variational inference, instead of computing the posterior distribution directly, we infer an approximation $q(\pi', \tau', z, l, \psi, \phi)$, referred to as variational distributions:

$$q(\pi', \tau', z, l, \psi, \phi) = q(\pi' \mid \rho)\, q(\tau' \mid \upsilon) \prod_{u=1}^{U} q(z_u \mid \kappa_u) \prod_{i=1}^{I} q(l_i \mid \varphi_i) \prod_{m=1}^{M} \prod_{t=1}^{T} q(\psi_m^t \mid \lambda_m^t) \prod_{t=1}^{T} q(\phi_t \mid \zeta_t)$$

where $q(z_u \mid \kappa_u)$ and $q(l_i \mid \varphi_u)$ are M-dimensional and T-dimensional Multinomial distributions; and $q(\psi_m^t \mid \lambda_m^t)$ and $q(\phi_t \mid \zeta_t)$ are C-dimensional Dirichlet distributions.

For the variational distributions $q(\pi' \mid \rho)$ and $q(\tau' \mid \upsilon)$, we rely on a stick-breaking truncation representation for a Chinese Restaurant Process similar to those in [37], which are truncated to M and T, respectively. The variational distributions are:

$$q(\pi' \mid \rho) = \prod_{m=1}^{M-1} \text{Beta}(\pi'_m \mid \rho_{m1}, \rho_{m2})$$
$$q(\tau' \mid \upsilon) = \prod_{t=1}^{T-1} \text{Beta}(\tau'_t \mid \upsilon_{t1}, \upsilon_{t2})$$

To approximate the posterior distributions p by variational distributions q, we use the *KL*-divergence between them, $KL(q \mid p)$. With $\boldsymbol{\Theta} \triangleq \{\pi', \tau', z, l, \psi, \phi\}$, it is defined as:

$$\begin{aligned} KL(q \mid p) &\triangleq -\int q(\boldsymbol{\Theta}) \ln \frac{p(\boldsymbol{\Theta} \mid \boldsymbol{x}, \boldsymbol{y})}{q(\boldsymbol{\Theta})} d\boldsymbol{\Theta} \\ &= -\int q(\boldsymbol{\Theta}) \ln \frac{p(\boldsymbol{\Theta}, \boldsymbol{x}, \boldsymbol{y})}{q(\boldsymbol{\Theta})} d\boldsymbol{\Theta} + \ln p(\boldsymbol{x}, \boldsymbol{y}) \\ &\geq -\int q(\boldsymbol{\Theta}) \ln \frac{p(\boldsymbol{\Theta}, \boldsymbol{x}, \boldsymbol{y})}{q(\boldsymbol{\Theta})} d\boldsymbol{\Theta} \triangleq -\mathcal{L}(\boldsymbol{\Theta}) \end{aligned}$$

$\mathcal{L}(\boldsymbol{\Theta})$ is called *evidence lower bound* (ELBO) and denotes the variational objective function. Using variational theory [36], taking derivatives of this lower bound with respect to each variational parameter, we derive the following coordinate ascent updates [37].

Local Updates. We first update local variables (connected to a single data point), i.e., z and l in our model. We update the respective distributions $q(z_u \mid \kappa_u)$ and $q(l_i \mid \varphi_i)$ as follows (details on the computation of these equations are given in the appendix):

$$\kappa_{um} \propto \exp\left(\sum_{i=1}^{I} \sum_{t=1}^{T} \varphi_{it} \mathbb{E}\left[\ln p(x_{iu} \mid \psi_m^t)\right] + \mathbb{E}\left[\ln \pi_m\right]\right) \tag{2}$$

$$\varphi_{it} \propto \exp\left(\mathbb{E}\left[\ln p(y_i \mid \phi_t)\right] + \mathbb{E}\left[\ln \tau_t\right]\right) \tag{3}$$

Global Updates. Next, we consider the updates for global (or outer) variables (connected to multiple data points), i.e., π, τ, ψ, and ϕ in our model. We update $q(\pi' \mid \rho)$ and $q(\tau' \mid \upsilon)$ as follows:

$$\rho_{m1} = 1 + \sum_{u=1}^{U} \kappa_{um} \qquad \rho_{m2} = \alpha + \sum_{u=1}^{U} \sum_{l=m+1}^{M} \kappa_{ul} \tag{4}$$

$$\upsilon_{t1} = 1 + \sum_{i=1}^{I} \varphi_{it} \qquad \upsilon_{t2} = \varepsilon + \sum_{i=1}^{I} \sum_{l=t+1}^{T} \varphi_{il} \tag{5}$$

Here, $\alpha, \varepsilon > 0$ are 'prior beliefs' on the maximum number of worker communities and item clusters. Yet, their effects are marginal, as the updates are dominated by the observed information (κ and φ), so that α, ε can safely be set to large values. Distributions $q(\psi_m^t \mid \lambda_m^t)$ and $q(\phi_t \mid \zeta_t)$ are update by means of:

$$\lambda_{mc}^t = \lambda_{m0}^t + \sum_{i=1}^{I} \varphi_{it} \sum_{u=1}^{U} \kappa_{um} x_{iu} \tag{6}$$

$$\zeta_{tc} = \zeta_{t0} + \sum_{i=1}^{I} \varphi_{it} y_i \tag{7}$$

We summarise our inference algorithm to learn the model parameters in Algorithm 1. It iteratively updates local parameters (κ, φ) and global parameters $(\rho, \upsilon, \lambda, \zeta)$. The observed data $(\boldsymbol{x}, \boldsymbol{y})$ is used in the updates of these parameters whenever their associated variables are connected to the observed variables. Note that many updates of variables are independent, which can be exploited to scale up the performance. For instance, the individual updates of κ parameters and φ parameters can be parallelised. The convergence of variational inference is proved in [38].

Time Complexity. Our inference algorithm scales linearly in terms of the size of data, i.e., the number of answers (worker-item pairs that are answered). In each iteration, the membership of each worker in worker communities (Eq. 2) and the membership of each item in clusters (Eq. 3) are updated, which is linear in the number of worker communities M and the number of item clusters T, respectively. Next, we update the worker distribution and confusion matrix of each worker community. Again, given

Algorithm 1 Variational Inference for the CPA Model

Input : Worker answers $\boldsymbol{x}$ and known true labels $\boldsymbol{y}$
Output: Estimated model parameters $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$

Random initialisation of $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$
while *not converged* **do**
  // Update the local variables
  for $u \leftarrow 1,\ldots,U$ *and* $m \leftarrow 1,\ldots,M$ **do** Update κ_{um} using Eq. 2 ;
  for $i \leftarrow 1,\ldots,I$ *and* $t \leftarrow 1,\ldots,T$ **do** Update φ_{it} using Eq. 3 ;
  // Update the global variables
  for $m \leftarrow 1,\ldots,M$ **do**
    Update ρ_{m1} and ρ_{m2} using Eq. 4
    for $t \leftarrow 1,\ldots,T$ *and* $c \leftarrow 1,\ldots,C$ **do** Update λ^t_{mc} using Eq. 6 ;
  for $t \leftarrow 1,\ldots,T$ **do**
    Update υ_{t1} and υ_{t2} using Eq. 5
    for $c \leftarrow 1,\ldots,C$ **do** Update ζ_{tc} using Eq. 7 ;
return $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$

the bounded numbers of worker communities M, item clusters T, and possible labels C, Eq. 4 and Eq. 6 are updated in linear time w.r.t. the number of workers and the number of answers, respectively. Finally, the item distribution and label assignment of each item cluster is updated. Since the number of item clusters T and possible labels C is bounded, updating Eq. 5 and Eq. 7 takes linear time w.r.t. the number of items.

From the above, it follows that the overall complexity is linear w.r.t the number of answers scaled by the number of iterations. Since variational methods require a small number of iterations (≤ 10 for 98% accuracy) [36], [37], our inference algorithm scales with the number of actual data points, which significantly saves computation cost due to the sparseness of crowdsourcing.

3.4 Prediction

To solve the partial-agreement answer aggregation problem, we construct a deterministic assignment $d : I \rightarrow 2^C$ using the maximum likelihood principle (MAP) [39]. After approximating the values of model parameters $\mathcal{P} \triangleq \{\alpha, \varepsilon, \gamma, \eta\}$, we predict the labels of non-grounded items. Technically, given an item i, we denote by $\boldsymbol{x}_{U_i} \triangleq \{x_{vi} \mid v \in U_i\}$ the labels assigned by the workers U_i who provided answers for this item. Further, $\mathcal{D} \triangleq \{\boldsymbol{x}, \boldsymbol{y}\}$ denotes the assigned labels as well as known labels as used in the inference. We now compute y_i using MAP estimation of the probability $p(y_i \mid \boldsymbol{x}_{U_i}, \mathcal{D}, \mathcal{P})$:

$$y_i^* = \underset{y_i}{\operatorname{argmax}}\ p(y_i \mid \boldsymbol{x}_{U_i}, \mathcal{D}, \mathcal{P}) = \underset{y_i}{\operatorname{argmax}}\ p(y_i, \boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P}) \quad (8)$$

since $p(y_i \mid \boldsymbol{x}_{U_i}, \mathcal{D}, \mathcal{P}) = p(y_i, \boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P}) / p(\boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P})$, there is no direct dependency between y_i and $\boldsymbol{x}_{U_i}$ in the graphical representation, and the divisor does not depend on y_i. The above formulation of the conditional probability of y_i, i.e. $p(y_i \mid \boldsymbol{x}_{U_i}, \mathcal{D}, \mathcal{P})$, has the advantage that it covers diverse crowdsourcing settings. For instance, the absence of known true labels ($\boldsymbol{y} = \emptyset$ in $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}$) or a separation of training data and testing data ($\boldsymbol{x}_{U_i} \nsubseteq \boldsymbol{x}$) can be directly encoded in this formulation.

To compute $p(y_i, \boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P})$, we factorise over all probabilistic dependencies in the graphical model representation. Using the derivation outlined in the appendix, we arrive at:

$$p(y_i, \boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P}) = \sum_{t=1}^{T} \varphi_{it} \prod_{u \in U_i} \left(\sum_{m=1}^{M} \kappa_{um} p\left(x_{ui} \mid \psi_m^{(t)MAP}\right) \right) p\left(y_i \mid \phi_t^{MAP}\right)$$

where $\psi_m^{(t)MAP}$ and ϕ_t^{MAP} are maximum a posteriori (MAP) estimates (aka modes) of the inferred distributions of ψ_m^t and ϕ_t.

However, the maximization problem in Eq. 8 is a zero-one integer problem, which is known to be NP-hard—the exhaustive search needs to explore $2^C - 1$ combinations of labels. Against this background, we may use a greedy search algorithm to approximate the mode y_i^* of the above distribution. Initially, all elements y_{ic}^* of the vector y_i^* are set as zeros. Then, we proceed iteratively and, in each iteration, set to one the element y_{ic^*} that leads to the largest increase of $p(y_i^*, \boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P})$. This procedure terminates once $p(y_i^*, \boldsymbol{x}_{U_i} \mid \mathcal{D}, \mathcal{P})$ can not be further increased. The final configuration of y_i^* will be the instantiation value for the deterministic assignment. Note that this instantiation can be done independently for all items, so that this step can be parallelised.

4 Scalable Model Inference and Prediction

Today's crowdsourcing datasets are very large [26], [40], so that answer aggregation becomes impractical even with a linear inference algorithm. Interactive crowdsourcing applications [5] might require very fast response time. However, simple divide-and-conquer methods such as matrix partitioning [41] are not applicable, since a split-up of the answer matrix causes information loss on worker communities and item clusters.

This section proposes two levels of scaling model inference and prediction as introduced in §3.3. First, we exploit incremental computation in the model inference to achieve online learning, i.e., we avoid recomputation of the whole model when new data arrives. Second, we leverage parallelisation to scale-out and scale-up inference and prediction. Finally, we discuss to which extent these techniques reduce the runtime of answer aggregation.

4.1 Online Learning

The inference and prediction methods introduced above for the CPA model solve the partial-agreement answer aggregation problem in a static setting. However, in many cases, tasks are not answered immediately when posted on a crowdsourcing platform. Rather, the set of worker answers is gradually building up over time and intermediate aggregation results are valuable from an application point of view [5]. For instance, intermediate results may indicate that a task is too difficult for workers, so that it shall be re-designed. Also, if intermediate results are of high quality, the crowdsourcing process can be terminated early to save cost.

We cater for such an online setting by means of incremental computation for the CPA model. We present an inference algorithm that incrementally updates the model parameters based on new data, which are then used for predicting the true labels of all items. In each learning iteration, we maintain only the most recent parameter values, thereby avoiding the cost of repeatedly building the model from the complete set of answers. While this approach comes with a modest reduction in aggregation quality (explored in our experiments), it greatly improves aggregation efficiency.

Stochastic Variational Inference (SVI). The deterministic variational inference (VI) presented in the previous section for the static setting maximises the EBLO function $\mathcal{L}(\boldsymbol{\Theta})$ using coordinate-ascent for each of the parameters of variational distributions. To realise incremental learning, we rely on stochastic variational inference [42] and apply stochastic optimization to the EBLO function based on newly received data. By relying on stochastic gradient descent (SGD) in Algorithm 1, we do not need all available data, but only a small subset of it, to update the parameters in each iteration. More precisely, while VI needs many epochs

to converge (each epoch corresponds to an iteration scanning the entire datasets), SVI is designed to converge in one epoch, which is why it is favoured in an online setting where each answer is used once. The convergence of stochastic variational inference is proved in [42].

Technically, data is received as a series of batches $b = 1, 2, \ldots$. Each batch b contains the answers of a fixed number of workers $\mathcal{U}_b$ (with U_b being the cardinality of $\mathcal{U}_b$) for a set of items $\mathcal{N}_b$. We consider new answers as a subsample and, based thereon, derive a stochastic gradient. Specifically, we compute the difference ∇ between old and new values of each parameter. Following [42], [43], we classify variational distributions as being *global* or *local*. In our setting, $q(l_i \mid \varphi_i)$, $q(\pi' \mid \rho)$, $q(\tau' \mid \upsilon)$, $q(\psi_m^t \mid \lambda_m^t)$, and $q(\phi_t \mid \zeta_t)$ are global, whereas $q(z_u \mid \kappa_u)$ is local (φ now becomes global as we consider multiple items in one update).

Natural Gradients. For the local distribution, we reuse the update formulation given for VB inference (i.e., Eq. 2). The respective distribution is connected to a single data point, which can be computed directly from the new data. In contrast, for the global distributions, natural gradients are obtained for each variable over all $u \in \mathcal{U}_b$ as follows (see the appendix for a detailed derivation):

$$\nabla_{\lambda_m^t} \mathcal{L}_u = \frac{-\lambda_m^t + \gamma_m^t + U \sum_{i \in \mathcal{N}_b} \varphi_{it} \kappa_{um} x_{iu}}{U} \tag{9}$$

$$\nabla_{\zeta_t} \mathcal{L}_u = \frac{-\zeta_t + \eta + \sum_{i \in \mathcal{N}_b} \varphi_{it} y_i}{U} \tag{10}$$

$$\nabla_{\rho_{m1}} \mathcal{L}_u = \frac{-\rho_{m1} + 1 + U \kappa_{um}}{U} \tag{11}$$

$$\nabla_{\rho_{m2}} \mathcal{L}_u = \frac{-\rho_{m2} + \alpha + U \sum_{l=m+1}^{M} \kappa_{um}}{U} \tag{12}$$

$$\nabla_{\upsilon_{t1}} \mathcal{L}_u = \frac{-\upsilon_{t1} + 1 + \sum_{i \in \mathcal{N}_b} \varphi_{it}}{U} \tag{13}$$

$$\nabla_{\upsilon_{t2}} \mathcal{L}_u = \frac{-\upsilon_{t2} + \varepsilon + \sum_{l=t+1}^{T} \sum_{i \in \mathcal{N}_b} \varphi_{il}}{U} \tag{14}$$

The natural gradient for $q(l_i \mid \varphi_i)$ is difficult to compute since the mean-parameterisation requires the constraints $\sum_{t=1}^{T} \varphi_{it} = 1$ and $0 \leq \varphi_{it} \leq 1$ to be satisfied. Hence, we prefer to work with a minimal canonical parameterisation in exponential family form, parametrising the distribution by μ instead of φ:

$$q(l_i \mid \mu_i) = \exp\left(\langle \mu_i, S(l_i) \rangle - B(\mu_i)\right).$$

In the above distribution, $\mu_i = \left[\mu_{i1}, \ldots \mu_{i(T-1)}\right]^\mathsf{T}$ is a $T-1$-dimensional vector parameter, $B(\mu_i) = 1 + \sum_{t=1}^{T-1} \exp(\mu_{it})$ is a normalisation function, and $S(l) = \left[\mathbf{I}(l-1), \ldots, \mathbf{I}(l-T+1)\right]^\mathsf{T}$ is a sufficient statistic function. The idea of sufficient statistics is to only maintain the minimal/sufficient information instead of all data points to compute the probability distribution. In our case, the sufficient function is defined as a $T-1$-dimensional binary vector, containing a value of one only at position l. That is, $\mathbf{I}(x)$ is an indicator function, $\mathbf{I}(x) = 1$ if $x = 0$; and $\mathbf{I}(x) = 0$, otherwise. The natural gradient for parameter μ is:

$$\nabla_{\mu_{it}} \mathcal{L}_u = \frac{-\mu_{it} + \mathbb{E}[\varepsilon_t] - \mathbb{E}[\varepsilon_T] + U(a_{it} - a_{tT})}{U} \tag{15}$$

where $a_{it} = \sum_{m=1}^{M} \kappa_{um} \mathbb{E}[\ln p(x_{iu} \mid \psi_m^t)]$ for $t = 1, \ldots, T$. To derive φ from μ, we use the following transformation:

$$\varphi_{it} = \frac{\exp(\mu_{it})}{1 + \sum_{t=1}^{T-1} \exp(\mu_{it})} \quad \text{for } t = 1, \ldots T-1 \tag{16}$$

$$\varphi_{iT} = \frac{1}{1 + \sum_{t=1}^{T-1} \exp(\mu_{it})} \tag{17}$$

Learning Rate. In incremental learning, a learning rate ω_b needs to be specified as a function of the batch index b. To ensure the convergence of the gradients, ω_b shall satisfy two conditions:

$$\sum_{b=1}^{\infty} \omega_b = \infty \quad \text{and} \quad \sum_{b=1}^{\infty} \omega_b^2 < \infty.$$

The learning rate depends on r, aka the *forgetting rate*. If r is large, ω_b becomes small, and the old parameter values are only slightly changed. Finding a good value for r is specific to a dataset. Yet, any value of $r \in (0.5, 1]$ leads to convergence [42]. Larger values of r often lead to higher learning quality and faster convergence (but not monotonically). We varied r in our experiments and found out that fixing r to $[0.85, 0.9]$ achieves good results.

Online Updates. Using the above gradients, the updates of all necessary parameters in the online setting become:

$$\lambda \leftarrow \lambda + \omega_b \frac{U}{U_b} \sum_{u \in \mathcal{U}_b} \nabla_\lambda \mathcal{L}_u \qquad \zeta \leftarrow \zeta + \omega_b \frac{U}{U_b} \sum_{u \in \mathcal{U}_b} \nabla_\zeta \mathcal{L}_u \tag{18}$$

$$\rho \leftarrow \rho + \omega_b \frac{U}{U_b} \sum_{u \in \mathcal{U}_b} \nabla_\rho \mathcal{L}_u \qquad \upsilon \leftarrow \upsilon + \omega_b \frac{U}{U_b} \sum_{u \in \mathcal{U}_b} \nabla_\upsilon \mathcal{L}_u \tag{19}$$

$$\mu \leftarrow \mu + \omega_b \frac{U}{U_b} \sum_{u \in \mathcal{U}_b} \nabla_\mu \mathcal{L}_u \tag{20}$$

The algorithm for incremental learning for the CPA model is illustrated in Algorithm 2. In each iteration, a pre-defined number of answers are collected from the crowd. Based on the new data, we compute the natural gradients and update the model parameters.

Algorithm 2 Stochastic Variational Inference for the CPA model

Input : Continuously updated worker answers $\boldsymbol{x}$ and known true labels $\boldsymbol{y}$
Output: Estimated model parameters $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$

Random initialisation of $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$
$b \leftarrow 1$ `// The batch index`
while *more answers are available* **do**
  Fetch the b-th batch of answers of users $\mathcal{U}_b$ for items $\mathcal{N}_b$ and set $b \leftarrow b+1$
  `// Update the local variables`
  for $u \in \mathcal{U}_b$ *and* $m \leftarrow 1, \ldots, M$ **do** Update κ_{um} using Eq. 2. ;
  `// Update the global variables`
  Compute the natural gradients using Eq. 9 to 15.
  Set learning rate $\omega_b = (1+b)^{-r}$
  Update $\lambda, \zeta, \rho, \upsilon, \mu$ using Eq. 18 to 20.
  Compute φ using Eq. 16 to 17.
return $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$

Online Prediction. Online prediction enables us to perform the instantiation of labels incrementally, upon the arrival of new answers. Different from the inference procedure for incremental learning, online prediction does not compute the difference between the old and new labels assignments. The reason is that the most recent parameter values constitute the probability distributions of all data obtained so far. Each time new answers are obtained, the parameter values are updated and their values can be used to generate the corresponding approximated posterior distributions of model variables required for instantiation, i.e., $q^{(b)}(l_i \mid \varphi_i)$, $q^{(b)}(z_u \mid \kappa_u)$, $q^{(b)}(\psi_m^t \mid \lambda_m^t)$, $q^{(b)}(\phi_t \mid \zeta_t)$, $q^{(b)}(\pi' \mid \rho)$ and $q^{(b)}(\tau' \mid \upsilon)$, where $b = 1, 2, \ldots$ is the batch index. These posteriors are approximations of their offline counterparts and, thus, are used as input of the instantiation procedure in §3.4.

4.2 Parallelisation

As a second angle for achieving scalability of model inference, we incorporate parallelisation of stochastic inference (Algorithm 2)

following the MapReduce principle [44]. The general idea can be summarised as follows: When the global variables are given, the updates to local variables become independent and can thus be computed concurrently.

Algorithm 3 outlines how to parallelise model inference using MapReduce primitives. It achieves scale-out by distributing the calculation of local variables κ_{um} and a_{it} for pairs of workers and items (MAP phase). Subsequently, the global variables $\lambda, \zeta, \rho, \vartheta, \mu$ and φ are calculated in a centralised manner (REDUCE phase). In each iteration, the observed data is partitioned for parallel processing. In our case, each user u represents a suitable key for such partitioning.

Algorithm 3 MapReduce for Stochastic Variational Inference

Input : Continuously updated worker answers $\boldsymbol{x}$ and known true labels $\boldsymbol{y}$
Output: Estimated model parameters $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$

Random initialisation of $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$
$b \leftarrow 1$ // The batch index
while *more answers are available* **do**
  Fetch the b-th batch of answers of users $\mathcal{U}_b$ for items $\mathcal{N}_b$ and set $b \leftarrow b+1$
  begin MAP $(\{x_{ui}\}, \{\lambda, \zeta, \rho, \upsilon, \varphi\})$:
    for $m \leftarrow 1, \ldots, M$ **do** Update κ_{um} using Eq. 2;
    for $\leftarrow 1, \ldots, I$ *and* $t \leftarrow 1, \ldots, T$ **do**
      Compute a_{it} in Eq. 15;
      emit $\{\kappa_{um}, a_{it}\}$
  Set learning rate $\omega_b = (1+b)^{-r}$
  begin REDUCE $(\omega_b, \{\kappa_{um}, a_{it}\})$:
    Accumulate ψ_i for all $u \in \mathcal{U}_b$
    Compute the natural gradients using Eq. 9 to 15.
    Update $\lambda, \zeta, \rho, \upsilon, \mu$ using Eq. 18 to 20.
    Compute φ using Eq. 16 to 17.
    broadcast globals $\{\lambda, \zeta, \rho, \upsilon, \varphi\}$
return $\lambda, \zeta, \rho, \upsilon, \kappa, \varphi$

Further, the prediction procedure is also parallelised in a trivial manner: the instantiation of labels is independent for all items and therefore can be done in parallel.

4.3 Scalability Analysis

We now assess the influence of online learning and parallelisation on the time complexity of model inference. Denote by T_1 and T_2 respectively the runtime for local updates and global updates in each iteration of Algorithm 1. Then, the total runtime is $(T_1 + T_2) \cdot C_1$ where C_1 is the number of iterations to reach the convergence. For the algorithm that exploits incremental computation to support online learning (Algorithm 2), the total runtime is $(\frac{T_1}{B} + T_2) \cdot C_2$ where B is the number of batches and C_2 is the number of iterations to reach the convergence. In large-scale datasets, we commonly have $C_2 \ll C_1$ and, in practice, this difference is about two orders of magnitude [43]. This is also reflected in our experiments as the accumulated runtime of the incremental approach is significantly faster than the non-incremental procedure. For the algorithm that also parallelises the calculation in each iteration, the total runtime would ideally be $(\frac{T_1}{B \cdot P} + T_2) \cdot C_2$ where P is the number of parallel processors (i.e., cores of a single machine or machines in a cluster). In practice, parallelisation still induces a certain overhead due to the synchronisation in the REDUCE phase, see also Amdahl's law [44].Accumulating the runtime for the complete data, we arrive at a total runtime of $(\frac{T_1}{B \cdot P} + T_2) \cdot C_2 \cdot B$.

Online learning and parallelisation also speed-up the prediction, with the speed-up ratio being $\frac{1}{B \cdot P}$ since prediction is scaled by batches (online learning) and items (parallelisation). The accumulated total runtime for prediction is $\frac{1}{B \cdot P} \cdot B$.

5 Evaluation

We evaluated our approach to partial-agreement answer aggregation along several dimensions. We first elaborate on our experimental setup (§5.1), before evaluating the following aspects:

- The effectiveness of our CPA model for answer aggregation in crowdsourcing (§5.2).
- The effectiveness and efficiency of CPA when using online learning and parallelisation (§5.3).
- The importance of representing worker communities and item clusters in the CPA model (§5.4).

As mentioned in §2.2, we further verify the existence of worker communities in real-world datasets. However, this is not the focus of our work, so that these results can be found in §5.5.

5.1 Experimental Setup

Task Design. Aiming at a realistic evaluation setup, we follow best practices on task design for crowdsourcing:

Batch processing: Each task consists of multiple items that are to be labelled by a single user. To mediate the trade-off between the overhead of switching tasks and the cognitive load of a single task, we follow recent studies on crowdsourcing effectiveness [45], suggesting a task size of 10 items.

Pricing: We vary the price for a task over the datasets based on the difficulty of the respective tasks. Considering that a simple task would take five minutes to complete and that the average wage of workers is around 2.00$/h [46], we set the task price to 0.1$, 0.2$, and 0.3$ for simple, medium, and difficult tasks, respectively.

Datasets. Our experiments have been conducted using five real-world datasets, spanning diverse application scenarios:

(1) Image annotation: From the NUS-WIDE set of tagged web images [13], we randomly selected 2000 images, such that tags are uniformly distributed. Each image has up to 10 tags, which serve as ground truth in our experiments. Workers were asked to assign a subset of around 30 possible tags to each image (the total number of labels in the taxonomy is 81).

(2) Topic annotation: We relied on a random sample of 2000 Twitter messages from the collection that was used in the TREC 2011 Microblog track [14]. This dataset assigns up to five topics (from a set of 49 topics) to each tweet and, again, we ensured a uniform distribution of topic labels in our sample.

(3) Aspect extraction: This dataset is about assigning evaluation aspects (e.g., *price* or *menu*) to restaurant reviews [47]. The ground truth, provided by [47], assigns up to five aspects (out of 262) to 3710 reviews. We designed crowdsourcing tasks that asked workers to assign a subset of 20 possible labels to each review. The set of possible labels contains the true labels and is filled up with the labels that have the highest co-occurrences with the true labels.

(4) Entity extraction: The T-NER dataset [48] contains 2400 tweets, to which entities (of ten categories such as products or facilities) shall be assigned. It also includes the ground truth for all tweets. We asked workers to tag each word of a tweet as being an entity or non-entity, so that any tweet is assigned a set of entities.

(5) Movie tagging: This dataset has been created by crawling the IMDB website, randomly selecting 500 movies from a total of 22 genres. Thus, the ground truth stems from the IMDB website. Workers have been asked to assign genres to the movies.

We employed workers to perform item labelling using the CrowdFlower platform [1]. In total, we spent a budget of 8772 tasks

[1] http://www.crowdflower.com/

TABLE 3: Statistics for the real-world datasets

Quantity	Dataset				
	(1) image	(2) topic	(3) aspect	(4) entity	(5) movie
# Items	269,648	16M	3710	2400	500
# Labels	81	49	262	1450	22
# Questions	2000	2000	3710	2400	500
# Workers	416	313	482	517	936
# Answers	22920	15080	19780	15510	14430
Unit Price ($)	0.01	0.02	0.03	0.02	0.01

for all datasets and ended up having a repository of 87720 label annotations for 10610 items from 2664 users, see Table 3.

The resulting datasets cover diverse crowdsourcing scenarios: the distribution of worker answers is skewed in datasets (1) and (5), whereas it is normal in (3); tasks in datasets (2), (3), and (4) require understanding of unstructured text, which is more difficult than the tasks in (1) and (5); labels in (1), (2), and (4) are strongly correlated, whereas there is little correlation between labels in (5).

Large-Scale Simulation. To evaluate the scalability of our approach in the context of very large crowdsourcing datasets as described in [26], [40], [49], we rely on simulation. To this end, we adapt existing tools [8] for the multi-label setting. When simulating large crowds, we vary several parameters in the data generation to obtain datasets that exhibit similar characteristics as real-world data: (i) the number of objects, (ii) the number of workers, and (iii) the number of labels. Moreover, we follow the guidelines described in [8] to simulate worker characteristics. Specially, we distribute the worker population into α% reliable workers, β% sloppy workers and γ% spammers ($\gamma/2$% random spammers and $\gamma/2$% uniform spammers). For example, random spammers and uniform spammers are simulated by setting their confusion matrix column vector to the same value for all labels or to 1 for one random label, respectively. Based on insights in worker populations of real-world crowdsourcing services [21], we select the following default parameters: $\alpha = 43$, $\beta = 32$ and $\gamma = 25$. In the simulation experiments, the ground truth is generated based on a multinomial distribution.

Metrics. In partial-agreement answer aggregation, results can be partially correct. We therefore rely on the set-based definition of precision and recall to evaluate the individual correctness of each item. Per item i, *individual precision* P_i is the ratio of correctly predicted labels and the total number of predicted labels, whereas *individual recall* R_i is the ratio of correctly predicted labels and the total number of true labels. For a complete dataset, *precision* P and *recall* R are the respective averages over all items. With $Y_i \triangleq \bigcup_{j \in Z}\{j \mid y_{ij} = 1\}$ and $Y_i^* \triangleq \bigcup_{j \in Z}\{j \mid y_{ij}^* = 1\}$, the measures are defined as:

$$P_i \triangleq \frac{|Y_i \cap Y_i^*|}{|Y_i^*|} \quad R_i \triangleq \frac{|Y_i \cap Y_i^*|}{|Y_i|} \quad P \triangleq \sum_{i=1}^{I} \frac{P_i}{I} \quad R \triangleq \sum_{i=1}^{I} = \frac{R_i}{I}$$

Baselines. We compare CPA against several state-of-the-art answer aggregation methods. Unlike our approach, most existing methods target single-label tasks. For a comparative analysis, therefore, we regard the multi-label problem as several instances of a single-label problem (each worker giving a Boolean answer for a given label) when applying the respective methods. In the end, each item is assigned with a probability of accepting or rejecting a given label. If this probability is larger than 0.5, the respective label is included in the aggregation result. All experiments work without knowledge on true labels ($\boldsymbol{y} = \emptyset$).

- *Majority voting (MV)* is the most applicable aggregation method for multi-label tasks [17], [18], even though it still considers all labels separately. The probability to accept a label for an item is computed as the ratio of 'votes' from workers who provided an answer for an item.
- *Expectation Maximization (EM)* is an answer aggregation model that implicitly captures worker communities in the joint estimation of the items' true labels and the workers' reliability [39]. This model is further refined by penalising each worker with an extra mislabelling cost [15].
- *Community-based Bayesian Classifier Combination (cBCC)* is a recently proposed extension of Bayesian Classifier Combination (BCC), the latter being the nonparametric version of the EM model [50]. cBCC extends BCC by explicitly modelling worker communities [23], [24]. Recent studies illustrate that cBCC outperforms BCC in general, providing promising results in particular for sparse data [23], [24].

The above baselines enable us to perform a generic comparison. While there are further aggregation methods for single-label tasks, they are either subeffective or require domain-specific knowledge [8], [9]. Other methods for multi-label tasks, e.g., [19], [51], [52], are not directly comparable, since they either consider labels independently (similar to *MV*), utilize content-based features of items, consider only 2-subsets of labels (which is biased), or consider all subsets of possible labels (which is intractable).

Experimental Environment. Most experimental results have been obtained on an Intel Core i7 system (3.4GHz, 12GB RAM). To mitigate the effect of randomness on model inference, we take the average result of 10 runs, in which the dataset is shuffled randomly. For the scalability experiments, we relied on an implementation of our approach in Apache Spark running on Intel Xeon 2.6GHz system (16 cores, 16GB RAM).

5.2 Effectiveness of *CPA*

Accuracy. We first evaluate the accuracy of our approach based on the CPA model against the baseline methods in a static setting. That is, we measure precision and recall of the aforementioned five datasets. Table 4 shows precision and recall obtained by our CPA with the three baseline methods (MV, EM, cBCC).

TABLE 4: Overall accuracy

Dataset	Precision				Recall			
	MV	EM	cBCC	CPA	MV	EM	cBCC	CPA
image	0.65	0.66	0.7	**0.81**	0.57	0.62	0.63	**0.74**
topic	0.57	0.60	0.62	**0.79**	0.54	0.54	0.55	**0.70**
aspect	0.52	0.61	0.65	**0.74**	0.53	0.56	0.6	**0.64**
entity	0.63	0.57	0.60	**0.79**	0.55	0.50	0.53	**0.70**
movie	0.61	0.74	0.78	**0.80**	0.56	0.68	0.7	**0.73**

An interesting finding is that for the datasets with strong label correlations (*image, topic, entity*), the accuracy of complex methods (EM, cBCC) is similar to majority voting (MV). Specifically for *entity* dataset, the EM and cBCC methods are even worse than MV (0.57 and 0.60 compared to 0.63 at precision, 0.50 and 0.53 compared to 0.55 at recall). For these datasets, our CPA model, significantly outperforms all baseline methods. This is because these methods neglect the dependencies between labels. For example, the correctness probability of two correlated labels might be larger than the individual probability of each label when considered separately; thus the existing models might predict only one of the two labels or none as the correct answer.

For datasets with little label correlations (*aspect, movie*), our CPA model still consistently outperforms the baseline methods. Taking the *movie* dataset as an example, cBCC achieves precision and recall values of 0.78 and 0.68, respectively, whereas CPA yields 0.80 precision and 0.73 recall. This difference can be attributed to the fact that an instance of a multi-label problem is not equivalent to the union of several instances of a single-label problem. A single worker in the multi-label setting will be considered as different entities in single-label settings, which could lead to misclassification of worker types across different labels even if the labels are not correlated.

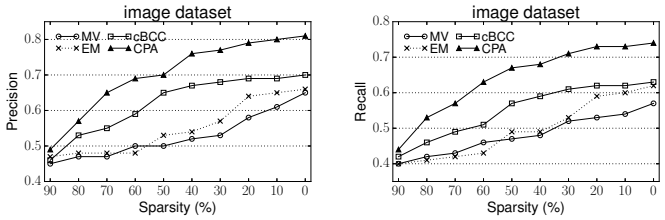


Fig. 3: Effects of sparsity

Robustness against Sparsity. In crowdsourcing scenarios, the answer matrix is typically sparse: most workers process only a few of the items of a particular application. We investigate the effect of sparsity on aggregation accuracy by randomly removing a certain share of the answers, in the step of leaving 10% of the data per dataset (i.e. the sparsity level increases from 0% to 100%). We then measure precision and recall, averaged over 100 runs.

As illustrated in Fig. 3, precision and recall decrease if answers are removed (sparsity is visualised in decreasing order). However, answer aggregation based on our CPA model is affected less by data sparsity compared to the baseline methods. For instance, for *image* tasks, when removing half of the input data (sparsity level 50%), the precision of our method is already 86% of the precision obtained using all answers. The baselines achieve at most 78% of the precision obtained using all answers in this case. This effect is due to the notion of worker communities in the CPA model that helps to identify consistent answers for an item even if it was processed only by a few workers.

Robustness to Spammers. As discussed in §2, crowdsourcing applications suffer from faulty workers, such as random and uniform spammers, which can account for up to 40% of the worker population [22]. Even though we may be able to detect different types of workers (based on their characteristics), the predicted labels may be incorrect, since faulty answers can be dominating. We investigate this aspect by adding answers of spammers to the datasets, such that they account for 20% or 40% of the data. The cBCC method turns out to yield the best results of all baselines, so that report it as the *baseline* for comparison.

As expected, the results in Fig. 4 show that precision and recall decrease when spammers are included. However, our approach is less affected by spammers as is the *baseline* method, in particular for large amounts of spammers (40%). For example, for the *aspect* dataset, the precision ratio of the baseline method decreases from 0.65 to 0.51, whereas it stays nearly constant with our approach, achieving 0.81 and 0.80, respectively. This highlights that our approach can not only detect communities of spammers, but also limits their influence on the aggregation result. Although cBCC has been designed to capture worker communities [23], [24], it does not perform well in a multi-label setting, which needs a more fine-grained level of assessing worker types. There may be cases in which the baseline wrongly classifies workers. For example, if a spammer gives only a single label as an answer, they may be considered reliability if the label appears in the correct answer.

Effects of Label Dependencies. Next, we study the effects of label dependencies in partial-agreement answer aggregation. Since baseline methods solve the multi-label problem as several instances of a single-label problem, they often treat missing correct labels in worker answers incorrectly. We study whether adding these missing correct labels into worker answers will improve their performance. If so, this is equivalent to the information loss when considering each label separately. For this experiment, we use the *entity* dataset as it shows the strongest correlations between labels. We simulate the effects of label dependencies in worker answers by randomly adding missing labels from the ground truth to worker answers that contain at least one correct label, varying from the amount of total missing labels between 10% and 30%. Again, we consider report solely on cBCC as the *baseline* since it yielded the best results over all baseline methods.

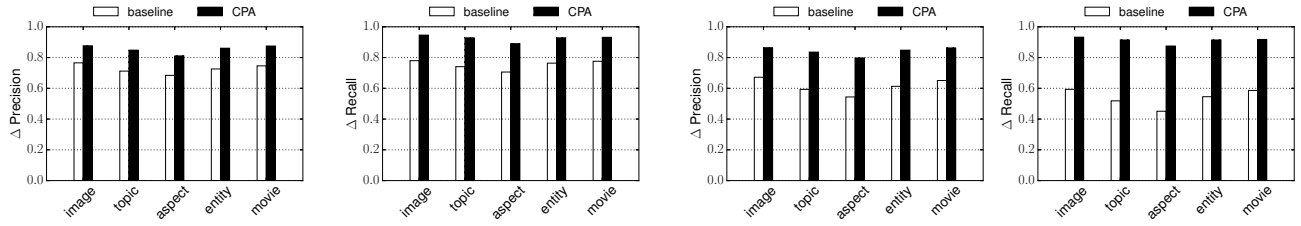
The result is depicted in Fig. 5, where precision and recall are reported in reverse ratios compared to the original performance of each method. Here, the *baseline* incurs a lot of information loss when ignoring the label dependencies, whereas our CPA model preserves such dependencies. For example, at dependency level 30%, the *baseline* loses nearly a half of precision and more than a half of recall. This result verifies that existing methods for the single-label problem cannot directly be used for the multi-label problem, as the information about label dependencies will be lost. As a result, these methods commonly do not predict the full set of labels for each item.

5.3 Scalable Model Inference and Prediction

Accuracy. Incremental computation for the CPA model as introduced in §4.1 aims at increasing the efficiency of computation. Yet, it may come at the expense of decreased effectiveness, i.e., lower accuracy in terms of precision and recall. We therefore compare the accuracy of the CPA model, once with the inference mechanisms for a static setting (*offline*) and once with the approach with incremental learning (*online*). Since both algorithms are approximation methods that might converge to local optima, we follow an empirical approach [42] to evaluate the performance boundaries between them. We simulate an online setting by randomly selecting new worker answers to represent newly arriving data, in steps of 10% of the dataset size. The forgetting rate of incremental learning r is varied in $(0.5, 1]$, with the best values observed for r falling into $[0.85, 0.9]$. For the non-incremental setting, this setup corresponds to a step-wise increase of the sparsity level from 90% to 0% and the prediction is always based on the complete set of answers received so far.

The result for the *image* dataset is shown in Fig. 6. We notice that indeed, precision and recall are worse when using incremental computation for the CPA model. Yet, even with incremental computation, the results are significantly better than those of the baselines (see previous experiments). This underlines that the summarised information about item clusters and worker communities maintained by our incremental inference method still enables competitively accurate aggregation.

The result for the *image* dataset in Fig. 6 is representative for all datasets. Table 5 shows precision and recall obtained after all answers have been processed, including the deviation when



(a) Spammer ratio = 20% **(b)** Spammer ratio = 40%

Fig. 4: Effects of spammers (compared to performance at spammer ratio = 0% as ratio)

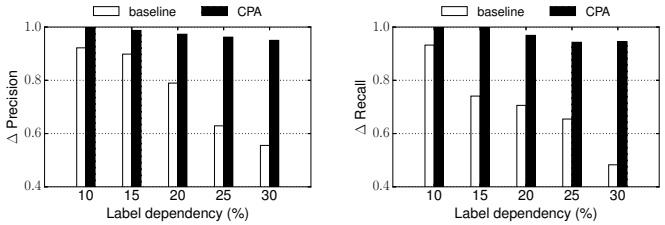


Fig. 5: Effects of label dependency

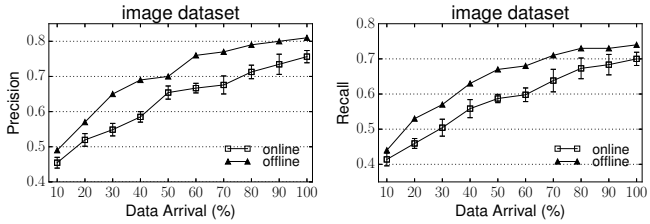


Fig. 6: Effects of data arrival

TABLE 5: Effects of data arrival (at 100%)

Dataset	Precision		Recall	
	online	offline	online	offline
image	0.76 ±0.02	0.81	0.70 ±0.02	0.74
topic	0.71 ±0.03	0.79	0.65 ±0.01	0.70
aspect	0.67 ±0.01	0.74	0.59 ±0.03	0.64
entity	0.70 ±0.02	0.79	0.64 ±0.01	0.70
movie	0.74 ±0.03	0.80	0.68 ±0.02	0.73

shuffling data and varying the forgetting rate. The incremental computation based on the CPA model incurs a competitive accuracy compared to the non-incremental approach.

Efficiency. Turning to the efficiency of our methods for scalable model inference and prediction, we measure the runtime for the static setting (*offline*) and of the incremental inference, without parallelization (*online*) or with parallelization using 4 or 16 cores (*online-4* and *online-16*), respectively, in relation to the size of the input data. We also include the runtime of baseline methods for comparison purposes. Since they run on multiple instances of single-label problem, we normalize all runtime measurements by the number of labels.

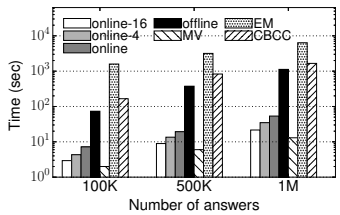


Fig. 7: Runtime of CPA inference and prediction mechanisms

Using the setup for large-scale simulation described in §5.1, we generate a synthetic dataset, comprising 10^4 items and 10^4 workers and 10 labels per item. Taking this worker pool, we vary the number of workers per item from 10 to 100 to randomly generate the answer matrix as input. Non-incremental inference is said to converge, if all model parameter differences in two consecutive inference iterations are below 10^{-3}. For incremental inference, we set the batch size to 100 answers and the forget rate to 0.875. We average results of 100 experiment runs.

As shown in Fig. 7, the scalable model inference and prediction is indeed much more efficient than the offline version, up to 32× faster. This speed-up is achieved mainly by scalable inference, which is performed on a fixed number of newly received answers as well as parallelisation over the number of answers. Moreover, our methods outperform other baselines except *MV*, which is also efficient since it maintains only the number of positive and negative answers to decide the majority for each label.

5.4 Importance of Model Requirements

Finally, we assess the importance of explicitly capturing worker communities (R1) and item clusters (R3) by comparing the accuracy of our CPA model with two simplified versions: *No_Z* removes the community structure (variable z) from the model, i.e., each worker is a singleton community; *No_L* removes the item cluster structure (variable l), i.e., each item represents a singleton cluster. Note that *No_L* solves the same problem as CPA, yet needs to compute the probability for all 2^C possible subsets of labels to then return the most probable subset. However, the *No_L* model turned out to be intractable for all except the *movie* dataset, which has a total of 22 possible labels.

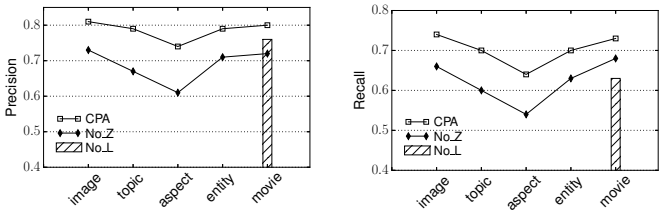


Fig. 8: Effects of model aspects

Fig. 8 shows that the CPA model consistently achieves the highest precision and recall. Improvements over the *No_Z* model are particularly large for the more difficult datasets (*topic* and *aspect*), since differentiation of workers is effective in these cases.

We further note that the *No_L* model achieves higher precision, but lower recall than the *No_Z* model. This highlights that worker communities help to improve correctness by identifying faulty workers, whereas item clusters improve completeness by exploiting label co-occurrence dependencies.

5.5 Experiments on Community Detection

Using the setup outlined in §5.1, this section reports on experiments to verify the existence of worker communities in real-world datasets, especially in the setting of multi-label answer aggregation. We will show that a worker may belong to different communities for different labels they provide. To this end, we compare the worker answers against the ground truth and visualise the differences between workers at the label level.

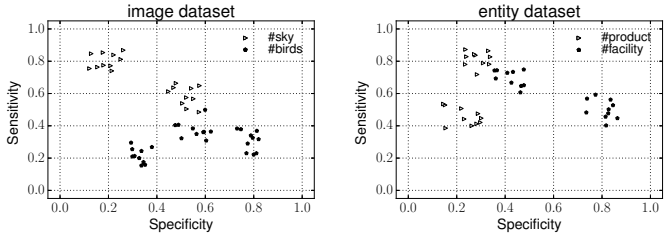


Fig. 9: Worker communities in real datasets

Fig. 9 shows the worker communities that our method infers for the *image* dataset and the *entity* dataset, for different degrees of task difficulty and label correlation. Each data point is a worker with their sensitivity (true positives) and specificity (true negatives) for each label. For example, if a worker provides the label sky in their answer and this label is contained in the ground truth, their answer is counted as one true positive for the label.

An important finding is that different communities build up for various labels. For example, there are 2 communities for label *sky* whereas there are 3 communities for label *birds*. Moreover, in the specificity range $[0,4,0.6]$ and the sensitivity range $[0.4,0.6]$, there is one *sky* community close to a *birds* community, hinting at a correlation between these two labels. From that it can be concluded that answer aggregation shall be based on worker communities that are characterised not only by different labels, but also the correlation between them. Another interesting observation is that the above characteristics is different for the *entity* dataset in terms of the sensitivity and specificity ranges as well as the number of communities. This calls for a nonparametric approach, which enables adaptation to particular domains and datasets.

6 Related Work

Having discussed the context of answer aggregation in crowdsourcing in §1 and §2, below, we now review the state-of-the art techniques and discuss further related areas.

State-of-the-Art in Answer Aggregation. Taking the requirements outlined for partial-agreement answer aggregation outlined in §2.2 as a starting point, it turns out that most existing algorithms for aggregating crowd answers are inapplicable, see [8], [7] for a comprehensive evaluation. The vast majority of aggregation methods do not collectively incorporate diverse characteristics of workers and their implications for answer correctness, and thus fail to address requirement (R1). For example, the two-coin model [53] captures the sensitivity (true positive rate) and specificity (true negative rate) of the worker, which is applicable for binary answers only. EM-based models [15], [39], [54] associate each worker with a confusion matrix; however, they are error prone to user-chosen initialization and data sparsity. One of the latest optimization-based methods is the one presented by Das Sarma et al. [55], which achieves a global optimum rather than a local optimum by reducing the item-label configuration space. Yet, the method assumes uniform quality of workers (which is problematic especially when the number of workers per item is high). More severely, the method clusters items with the same answers, which makes the method inapplicable for the multi-label problem where different items can have overlapping but different answers.

A notable extension of the EM model has been proposed by Kim et al. [50]. Based thereon, the models in [23], [24] incorporate worker communities. As our CPA model, these approaches are based on a Bayesian nonparametric generative model [33]. Yet, they have been developed for a single-label problem and thus neither support partial answer validity (R2) nor exploit label dependencies (R3). As shown in our experimental evaluation that considered this line of work as one of the baselines, addressing these requirements is crucial to obtain accurate results. We further note that our problem setting is different from the one of top-k algorithms for crowdsourcing tasks [12], as each item may have a different number of labels.

Scalable Crowdsourcing. Most answer aggregation algorithms operate in batch mode; hence, the aggregated answers would be recomputed from scratch every time a new worker answer arrives. There are only a few approaches on incremental answer aggregation, such as online EM [56]–which targets incremental updates when a new answer arrives and *i*-EM [41]–which targets incremental updates whenever the ground truth is extended. However, tailoring such incremental methods for our setting is non-trivial due to the dependency between labels and the community modelling of workers and items; e.g. a single new answer can be propagated to change the whole model, resulting in inefficient execution. Matrix partitioning [41] may also be used to reduce the computational effort, yet this incurs the loss of information about the dependencies between workers, which is essential for modelling the worker community. CPA provides a first method for answer aggregation for multi-label tasks, which scales not only in the number of possible labels, but also in the number of possible answers (subsets of possible labels).

Worker Modeling. The existence of workers with different characteristics calls for models that capture these characteristics in crowdsourcing. Traditional techniques model worker types implicitly as a part of answer aggregation, via prior-knowledge, via known difficulty of questions, or textual analysis of crowd questions [57], [58], [8], [31]. However, these approaches are domain-specific, sensitive to data and do not provide a meaningful description of the worker population. Recent work tackles this issue by explicitly introducing the concept of worker community [21], [22]. However, defining a fixed number of worker communities is error-prone since different domains exhibit different numbers and distributions of worker communities. Our work follows a nonparametric approach that allows the formation of worker communities to be adaptive to the considered data. Dynamically modelling of worker communities was also considered in [23], [24]. However, these approaches are tailored to a single-label problem, where each worker belongs to a single community. Our CPA model supports workers being in different communities per label and also enables propagation of community information across different labels via the dependencies between labels.

Multi-label Problems. Multi-label problems have been solved in related research fields, such as multi-label classification [59], ordinal classification [60], and data streams classification [61]. Multi-label classification aims at learning classifiers to associate each item with a set of labels. Yet, different from the crowdsourc-

ing setting, it is based on the features of the data itself, such as image pixels or textual indicators [59]. Ordinal classification studies a similar setting, but assumes a natural ordering among labels. It can be traced back to multi-label classification through membership functions [60]. Our work considers a more generic relation between labels in terms of co-occurrence dependencies. Data streams classification aims at multi-label classification in an online setting, processing data in real-time [61]. Despite the differences in the underlying classification problem (answer aggregation is not based on features of the items to be labelled), this setting is similar to the online setting in crowdsourcing.

Multi-label problems have been studied in the context of crowdsourcing before, yet the focus has been primarily on minimizing cost when posting tasks, see [62]. Another example is work on optimising the cost of hiring workers when generating training data for classifiers [51]. However, these approaches assume labels to be independent and consider all workers equally—adopting some form of majority voting to aggregate answers or unifying all single-label aggregation results [17], [18], [51]. Other techniques utilize the content-based features of each item [19], which is not generally available; consider only 2-subsets of labels [51], which is biased; or consider all subsets of possible labels, which is intractable in practice [52]. Answer aggregation for multi-label crowdsourcing that takes into account the worker communities, partial answer validity, label dependencies, and adaptivity of the aggregation model, in turn, has not been addressed before.

Bayesian Models. Graphical probabilistic models have been successfully applied in various domains, such as image processing, video encoding, and machine learning [63]. Their main benefit is the ability to explicitly capture dependencies between random variables, e.g., in terms of factor graphs. While graphical models have been applied in crowdsourcing with similar tools (e.g. probabilistic distributions, generative functions), their customizations to specific requirements and applications often lead to significant difference [50], [23], [24]. Most attempts to use graphical models for multi-label answer aggregation, e.g., [52], [19], [51], [64], show three major limitations: (1) the models are parametric or use external knowledge, which enforces assumptions on the true distribution of crowdsourcing data and is domain-dependent, even though ground truth is commonly not available; (2) they ignore worker communities, even though spammers may have a huge impact on the aggregation result, see [27]; (3) they neglect or are limited to simple (e.g. pair-wise) or intractable (all possible subsets) dependencies between labels. Our CPA model overcomes these limitations by capturing worker communities and item clusters explicitly in a Bayesian nonparametric model.

Relations between labels may stem from external expert knowledge [34], [35], instead being learned from the data itself, as done with our approach in a domain-independent manner. However, such expert knowledge is orthogonal, meaning that it could be incorporated in our approach. Prior knowledge could be expressed as conditional probabilities, which are then integrated in the label selection, i.e., step 2b of the generative process of the CPA model.

7 Conclusion

In this paper, we presented a novel Bayesian nonparametric approach to aggregate partial-agreement crowdsourcing answers. The key features of the proposed CPA model are its ability to capture worker characteristics (by worker communities) and dependencies between the labels assigned to items (by item clusters). The former improves precision by separating answers of faulty workers from those of reliable workers; the latter improves recall by exploiting co-occurrence dependencies to complete results. We further presented inference and prediction mechanisms for the CPA model. In particular, aiming at answer aggregation for very large datasets, we proposed scalable model inference and prediction based on incremental computation and parallelisation. Our experimental results showed that answer aggregation based on the CPA model outperforms state-of-the-art methods for answer aggregation by up to 134% in precision and recall, while being robust against spammers and answer sparsity. In future work, we intend to lift our model to other types of crowdsourcing tasks (e.g., assignment of continuous labels), and incorporate domain-specific information, such as question difficulty and label hierarchies.

References

[1] A. J. Quinn and B. B. Bederson, "Human computation: a survey and taxonomy of a growing field," in *CHI*, 2011, pp. 1403–1412.

[2] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart, "Crowd mining," in *SIGMOD*, 2013, pp. 241–252.

[3] C. Sun, N. Rampalli, F. Yang, and A. Doan, "Chimera: Large-scale classification using machine learning, rules, and crowdsourcing," in *VLDB*, 2014, pp. 1529–1540.

[4] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, "Crowdsourced poi labelling: Location-aware result inference and task assignment," in *ICDE*, 2016, pp. 61–72.

[5] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "icrowd: An adaptive crowdsourcing framework," in *SIGMOD*, 2015, pp. 1015–1030.

[6] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "Qasca: A quality-aware task assignment system for crowdsourcing applications," in *SIGMOD*, 2015, pp. 1031–1046.

[7] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: is the problem solved?" in *VLDB*, 2017, pp. 541–552.

[8] N. Q. V. Hung, N. T. Tam, L. N. Tran, and K. Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *WISE*, 2013, pp. 1–15.

[9] A. Sheshadri and M. Lease, "Square: A benchmark for research on computing crowd consensus," in *HCOMP*, 2013, pp. 156–164.

[10] P. Welinder and P. Perona, "Online crowdsourcing: rating annotators and obtaining cost-effective labels," in *CVPR*, 2010.

[11] R. G. Gomes, P. Welinder, A. Krause, and P. Perona, "Crowdclustering," in *NIPS*, 2011, pp. 558–566.

[12] X. Zhang, G. Li, and J. Feng, "Crowdsourced top-k algorithms: An experimental evaluation," in *VLDB*, 2016, pp. 612–623.

[13] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *CIVR*, 2009, pp. 48:1–48:9.

[14] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff, "Overview of the trec-2011 microblog track," in *TREC*, 2011.

[15] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *KDD*, 2010, pp. 64–67.

[16] D. R. Karger, S. Oh, and D. Shah, "Efficient crowdsourcing for multi-class labeling," in *SIGMETRICS*, 2013, pp. 81–92.

[17] S. Nowak and S. Rüger, "How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation," in *MIR*, 2010, pp. 557–566.

[18] J. Deng, O. Russakovsky, J. Krause, M. S. Bernstein, A. Berg, and L. Fei-Fei, "Scalable multi-label annotation," in *CHI*, 2014, pp. 3099–3102.

[19] S.-Y. Li, Y. Jiang, and Z.-H. Zhou, "Multi-label active learning from crowds," *arXiv preprint arXiv:1508.00722*, 2015.

[20] S. Sen, S. K. Lam, A. M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F. M. Harper, and J. Riedl, "Tagging, communities, vocabulary, evolution," in *CSCW*, 2006, pp. 181–190.

[21] G. Kazai, J. Kamps, and N. Milic-Frayling, "Worker types and personality traits in crowdsourcing relevance labels," in *CIKM*, 2011, pp. 1941–1944.

[22] J. Vuurens, A. de Vries, and C. Eickhoff, "How much spam can you take? an analysis of crowdsourcing results to increase accuracy," in *CIR*, 2011, pp. 48–55.

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

[24] P. G. Moreno, A. Artes-Rodriguez, Y. W. Teh, and F. Perez-Cruz, "Bayesian nonparametric crowdsourcing," *JMLR*, pp. 1607–1627, 2015.

[25] G. Li, J. Wang, Y. Zheng, and M. Franklin, "Crowdsourced data management: A survey," *TKDE*, 2016.

[26] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios, "Challenges in data crowdsourcing," *TKDE*, 2016.

[27] B. Zhao, B. I. Rubinstein, J. Gemmell, and J. Han, "A bayesian approach to discovering truth from conflicting sources for data integration," in *VLDB*, 2012, pp. 550–561.

[28] W. Zhang, Y. Lu, X. Xue, and J. Fan, "Automatic image annotation with weakly labeled dataset," in *MM*, 2011, pp. 1185–1188.

[29] X. Xue, W. Zhang, J. Zhang, B. Wu, J. Fan, and Y. Lu, "Correlative multi-label multi-instance image annotation," in *ICCV*, 2011, pp. 651–658.

[30] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *TKDE*, pp. 1819–1837, 2014.

[31] K. Lee, J. Caverlee, and S. Webb, "The social honeypot project: protecting online communities from spammers," in *WWW*, 2010, pp. 1139–1140.

[32] V. C. Raykar and S. Yu, "Ranking annotators for crowdsourced labeling tasks," in *NIPS*, 2011, pp. 1809–1817.

[33] T. Ferguson, "A bayesian analysis of some nonparametric problems," *AOS*, pp. 209–230, 1973.

[34] T. Han, H. Sun, Y. Song, Y. Fang, and X. Liu, "Incorporating external knowledge into crowd intelligence for more specific knowledge acquisition," in *IJCAI 2016*, 2016, pp. 1541–1547.

[35] Y. Fang, H. Sun, G. Li, R. Zhang, and J. Huai, "Effective result inference for context-sensitive tasks in crowdsourcing," in *DASFAA*, 2016, pp. 33–48.

[36] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul, "An introduction to variational methods for graphical models," *ML*, pp. 183–233, 1999.

[37] D. Blei and M. Jordan, "Variational inference for dirichlet process mixtures," *Bayesian Anal.*, pp. 121–143, 2006.

[38] M. J. Wainwright, M. I. Jordan *et al.*, "Graphical models, exponential families, and variational inference," *FTML*, pp. 1–305, 2008.

[39] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm," *Applied statistics*, pp. 20–28, 1979.

[40] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, and S. Madden, "Scaling up crowd-sourcing to very large datasets: a case for active learning," in *VLDB*, 2014, pp. 125–136.

[41] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer, "Minimizing efforts in validating crowd answers," in *SIGMOD*, 2015, pp. 999–1014.

[42] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *JMLR*, pp. 1303–1347, 2013.

[43] C. Wang, J. Paisley, and D. Blei, "Online variational inference for the hierarchical dirichlet process," in *AISTATS*, 2011, pp. 752–760.

[44] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *CACM*, pp. 107–113, 2008.

[45] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *EC*, 2010, pp. 209–218.

[46] J. Ross, L. Irani, M. Silberman, A. Zaldivar, and B. Tomlinson, "Who are the crowdworkers?: shifting demographics in mechanical turk," in *CHI*, 2010, pp. 2863–2872.

[47] J. Pavlopoulos and I. Androutsopoulos, "Aspect term extraction for sentiment analysis: New datasets, new evaluation measures and an improved unsupervised method," in *ACL*, 2014, pp. 44–52.

[48] A. Ritter, S. Clark, O. Etzioni *et al.*, "Named entity recognition in tweets: an experimental study," in *ACL*, 2011, pp. 1524–1534.

[49] P. G. Ipeirotis and E. Gabrilovich, "Quizz: targeted crowdsourcing with a billion (potential) users," in *WWW*, 2014, pp. 143–154.

[50] H.-C. Kim and Z. Ghahramani, "Bayesian classifier combination," in *AISTATS*, 2012, pp. 619–627.

[51] J. Bragg, D. S. Weld *et al.*, "Crowdsourcing multi-label classification for taxonomy creation," in *HCOMP*, 2013, pp. 25–33.

[52] L. Duan, S. Oyama, H. Sato, and M. Kurihara, "Separate or joint? estimation of multiple labels from crowdsourced annotations," *ESWA*, pp. 5723–5732, 2014.

[53] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Mach. Learn. Res.*, pp. 1297–1322, 2010.

[54] Y. Zheng, R. Cheng, S. Maniu, and L. Mo, "On optimality of jury selection in crowdsourcing," in *EDBT*, 2015.

[55] A. Das Sarma, A. Parameswaran, and J. Widom, "Towards globally optimal crowdsourcing quality management: The uniform worker setting," in *SIGMOD*, 2016, pp. 47–62.

[56] A. Artikis, M. Weidlich, F. Schnitzler, I. Boutsis, T. Liebig, N. Piatkowski, C. Bockermann, K. Morik, V. Kalogeraki, J. Marecek *et al.*, "Heterogeneous stream processing and crowdsourcing for urban traffic management." in *EDBT*, 2014, pp. 712–723.

[57] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver, "How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing," in *ICML*, 2012.

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

[59] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *KDD*, 2010, pp. 999–1008.

[60] W. Kotlowski and R. Slowinski, "On nonparametric ordinal classification with monotonicity constraints," *TKDE*, pp. 2576–2589, 2013.

[61] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "A survey of classification methods in data streams," in *Data Streams*, 2007, pp. 39–59.

[62] C. C. Cao, J. She, Y. Tong, and L. Chen, "Whom to ask?: jury selection for decision making tasks on micro-blog services," in *VLDB*, 2012, pp. 1495–1506.

[63] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[64] Y. Zheng, G. Li, and R. Cheng, "Docs: a domain-aware crowdsourcing system using knowledge bases," in *VLDB*, 2016, pp. 361–372.

Nguyen Quoc Viet Hung is a full-time Postdoctoral Research Fellow in DKE Group at University of Queensland, working with Prof. Xiaofang Zhou and Prof Yufei Tao since October 2015. Before joining UQ, he earned his Master and PhD degrees from EPFL (Switzerland), under the supervision of Prof. Karl Aberer. He also spent 2 years as a postdoc at EPFL. His research focuses on Data Integration, Data Exploration, Data Quality, Uncertainty Management and Crowdsourcing, with special emphasis on web data and sensor data. He published several papers in top-tier conferences and journals such as SIGMOD, ICDE, SIGIR, TKDE.

Huynh Huu Viet is a PhD student at Deakin University. His research interests mainly focus on Bayesian nonparametric for large-scale datasets. The aim of his research is to construct novel Bayesian nonparametric models for multiple data sources which are highly correlated and to develop new methods for fast inference in Bayesian nonparametric models.

Nguyen Thanh Tam is a PhD student at EPFL-LSIR under supervision of Prof. Karl Aberer. He has received his Master degree at Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. He has a BSc degree in Computer Science from Ho Chi Minh City University of Technology (HCMUT). His research interests include database technology, information theory, and machine learning.

Matthias Weidlich is a junior professor at the Department of Computer Science at Humboldt-Universität zu Berlin. He leads the Process-Driven Architectures group, which is funded by the German Research Foundation (DFG) through the Emmy-Noether Programme. Before joining HU in April 2015, he hold positions at the Department of Computing at Imperial College London and at the Technion - Israel Institute of Technology. He holds a PhD from the Hasso Plattner Institute (HPI), University of Potsdam. His research focuses on process-oriented and event-driven systems and his results appear regularly in the premier conferences (VLDB, SIGMOD, ICDE) and journals (TSE, TKDE) in the field.

Hongzhi Yin received the PhD degree in computer science from Peking University in 2014. His research interests include recommender system, user profiling, topic models, deep learning, social media mining, and location-based services. He has published over 30 papers in the most prestigious journals and conferences such as SIGMOD, KDD, VLDB, ICDE, the IEEE Transactions on Knowledge and Data Engineering, the ACM Transactions on Information Systems the ACM Transactions on Intelligent Systems and Technology, and the ACM Transactions on Knowledge Discovery from Data. Besides, he has one monograph published by Springer. He is an ARC DECRA Fellow with the University of Queensland.

Xiaofang Zhou received the BSc and MSc degrees in computer science from Nanjing University, China, in 1984 and 1987, respectively, and the PhD degree in computer science from the University of Queensland, Australia, in 1994. He is a professor of computer science at the University of Queensland and adjunct professor in the School of Computer Science and Technology, Soochow University, China. His research interests include spatial and multimedia databases, high performance query processing, web information systems, data mining, bioinformatics, and e-research.