CrowdGP: A Gaussian process model for inferring relevance from crowd annotations

Li, D.; Ren, Z.; Kanoulas, E.

DOI
10.1145/3442381.3450047

Publication date
2021

Document Version
Final published version

Published in
The Web Conference 2021

License
CC BY

Link to publication

Citation for published version (APA):
Li, D., Ren, Z., & Kanoulas, E. (2021). CrowdGP: A Gaussian process model for inferring relevance from crowd annotations. In *The Web Conference 2021: proceedings of the World Wide Web Conference WWW 2021 : April 19-23, 2021, Ljubljana, Slovenia* (pp. 1821-1832). Association for Computing Machinery. https://doi.org/10.1145/3442381.3450047

CrowdGP: a Gaussian Process Model for Inferring Relevance from Crowd Annotations

Dan Li
University of Amsterdam
Amsterdam, The Netherlands
d.li@uva.nl

Zhaochun Ren*
Shandong University
Qingdao, China
zhaochun.ren@sdu.edu.cn

Evangelos Kanoulas
University of Amsterdam
Amsterdam, The Netherlands
e.kanoulas@uva.nl

ABSTRACT

Test collection has been a crucial factor for developing information retrieval systems. Constructing a test collection requires annotators to assess the relevance of massive query-document pairs. Relevance annotations acquired through crowdsourcing platforms alleviate the enormous cost of this process but they are often noisy. Existing models to denoise crowd annotations mostly assume that annotations are generated independently, based on which a probabilistic graphical model is designed to model the annotation generation process. However, tasks are often correlated with each other in reality. It is an understudied problem whether and how task correlation helps in denoising crowd annotations.

In this paper, we relax the independence assumption to model task correlation in terms of relevance. We propose a new crowd annotation generation model named CrowdGP, where true relevance labels, annotator competence, annotator's bias towards relevancy, task difficulty, and task's bias towards relevancy are modelled through a Gaussian process and multiple Gaussian variables respectively. The CrowdGP model shows better performance in terms of interring true relevance labels compared with state-of-the-art baselines on two crowdsourcing datasets on relevance. The experiments also demonstrate its effectiveness in terms of selecting new tasks for future crowd annotation, which is a new functionality of CrowdGP. Ablation studies indicate that the effectiveness is attributed to the modelling of task correlation based on the auxiliary information of tasks and the prior relevance information of documents to queries.

CCS CONCEPTS

• **Information systems** → **Relevance assessment**; • **Computing methodologies** → **Learning in probabilistic graphical models**.

KEYWORDS

Crowdsourcing, relevance label, Gaussian process, Bayesian model

ACM Reference Format:
Dan Li, Zhaochun Ren, Evangelos Kanoulas. 2021. CrowdGP: a Gaussian Process Model for Inferring Relevance from Crowd Annotations. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450047

1 INTRODUCTION

Test collections have significantly benefited the development of information retrieval (IR) systems [42]. They are used both as training data to develop new retrieval models and as test data to evaluate model performance. Building such test collections requires assessing the relevance of massive documents to a set of queries. Traditionally the assessment of relevance is performed by trained professionals in a controlled lab environment [42]. As the need for creating new test collections to support the development of algorithms increases, so does the number of documents that need to be labeled. This makes the construction of test collections expensive and time-consuming. Crowdsourcing has been widely adopted as a cost-effective solution by the IR community [1, 12, 15–17]. Typically, the *task* of obtaining the relevance label of a document to a query is assigned to crowd *annotators* (or workers) in the form of human intelligence tasks.

Despite its cost-effectiveness, crowdsourcing introduces a major challenge – improving the quality of the crowd labels through careful label aggregation approaches [19]. majority voting (MV) has been the most prominent aggregation method, with early experiments showing that labels derived by a majority vote of multiple untrained crowd annotators can reach a quality comparable to that of a trained NIST assessor [2]. More recent work has shown that the quality of crowd annotations is affected by a number of factors such as the *difficulty of tasks* and the *competence of annotators* [13, 17], which are not considered by majority voting. Mainstream approaches assume different annotation generation processes and use different probabilistic graphical models (PGMs) to model the two factors. The hypothesis is that if a task is inherently easy, the labels from different crowd annotators will be consistent; otherwise, the labels will differ. Similarly, if an annotator is familiar with the topic of the search query and is well motivated, he or she is more likely to give correct answers. Note that they also assume that the *latent true labels* of tasks are independent and the annotations are generated independently by different annotators as well.

However, the independence assumption existing in most PGMs is over simplifying the actual crowd annotation and hence they do not show consistently better performance than majority voting on diverse benchmarks [22]. In fact, the label of a task may be roughly inferred from the labels of those similar to it. It has not been well investigated on how to model this property to improve label quality from noisy crowd annotations. In this paper, we extend the existing work by two means: (1) relaxing the independence assumption and instead modelling task correlation in terms of relevance, and (2)

*Zhaochun Ren is the corresponding author.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450047

assuming a different annotation generation process to model the latent true labels using a Gaussian process (GP), as well as annotator competence and annotator's bias towards relevancy, task difficulty and task's bias towards relevancy through multiple Gaussian variables. The model is named CrowdGP as it is a GP-based model to denoise crowd annotations.

In particular, we make the hypothesis that *a document is correlated with other documents close to it in some document feature space in terms of their relevance labels to an information need.* This hypothesis has been examined extensively in the IR community, e.g., in the *cluster hypothesis* [28]. The consideration of using a GP prior to model task correlation is that it allows us to integrate auxiliary information of tasks (such as query-document text and document ranks produced by different system runs) and prior knowledge of relevance (such as pretrained ranking models); moreover, the model learned on crowd annotations can be further used to predict labels for new tasks that have no crowd annotations, which is not supported in most existing PGM approaches. Second, we assume that *both the noise from tasks and the noise from annotators affect the observed crowd labels.* This is evidenced in work of Maddalena et al. [27] who have empirically shown that topic difficulty, annotator competence and relevance label (relevant or nonrelevant) all affect the quality of crowd annotations. We use multiple Gaussian variables to model task noise and annotator noise, respectively. The advantage is that we can use the variance parameter of a Gaussian distribution to model how the labels differ, and use the mean parameter to model any bias towards or against relevancy, both per task and per annotator. Besides, such choice makes it easy to calculate the likelihood function in Equation (15). Finally, the new annotation generation process is assumed as follows: *an observed (discrete) crowd label is generated from a Bernoulli distribution, of which the positive probability is determined by three variables: the latent GP, the task noise and the annotator noise.* The model is optimized using a variational expectation maximization (VEM) algorithm [33].

The main contributions in this paper are the following:

- We propose a new probabilistic graphical model CrowdGP, which captures latent true labels, annotator competence, annotator's bias towards relevancy, task difficulty, and task's bias towards relevancy for observed tasks. The model is able to infer true labels from crowd annotations and predict labels for new tasks that have no crowd annotations.
- We propose to use a VEM algorithm to effectively and efficiently learn model parameters. We empirically demonstrated its effectiveness compared against stochastic gradient descent (SGD).
- We empirically demonstrated the effectiveness of CrowdGP in terms of inferring latent true labels and selecting new tasks for crowd annotation. We provide detailed analysis of the effect of its components including task feature and mean function on performance, as well as the learning behaviour of CrowdGP under different hyperparameters.

2 RELATED WORK

2.1 Crowdsourcing in Information Retrieval

Relevance assessment is critical to generate high quality IR evaluation collections. The wide application of crowdsourcing relevance assessment introduces a new challenge of query control. The text retrieval conference (TREC) crowdsourcing tracks from 2011 to 2013 [39–41] investigated the use of crowdsourcing techniques to evaluate retrieval systems. Alonso and Mizzaro [1] have shown that crowdsourcing labels and the expert labels produced by TREC assessors correlate well in the IR evaluation measures.

The quality of crowd annotations are significantly affected by human factors. For example, Kazai et al. [17] showed that an annotator's motivation, interest and familiarity with the task, perceived task difficulty, and satisfaction with the offered payment all influence the quality of the crowd annotations; Han et al. [13] concluded that task instruction, task subjectivity, task type, and the monetary reward of tasks all influence the quality of crowd annotations. Various quality control methods to ensure crowd annotation quality during the running of crowdsourcing tasks have been investigated. For example, limiting the time available for relevance labelling is useful to construct a high-quality test collection [26]; asking for annotator's rationale such as a justification more than just a relevance label improves the quality of crowd annotations [30]. These studies lay the foundation of the hypotheses in Section 4.2.

2.2 Probabilistic Models for Aggregating Crowd Annotations

The goal of label aggregation is to infer the true label of each task given redundant and maybe inconsistent crowd annotations. The mainstream solution is to design a PGM to model the annotation generation process. Usually, the latent true labels of tasks are independent from each other; and given the latent true label of a task, each observed label is assumed to be generated independently from other observed labels. Let y_i^j denote the annotator j's label to task i, and y_i denote the latent true label for task i. Let Y denote the annotations for all the tasks and the corresponding annotators, $\boldsymbol{y}$ denote the latent true labels for all tasks. The joint distribution of Y and $\boldsymbol{y}$ is defined as $p(Y, \boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_i p(y_i \mid \boldsymbol{\theta}) \prod_j p(y_i^j \mid y_i, \boldsymbol{\theta})$ for most PGMs, but the major difference is the way they model $p(y_i \mid \boldsymbol{\theta})$ and $p(y_i^j \mid y_i, \boldsymbol{\theta})$. The likelihood of the observed annotations is $p(Y \mid \boldsymbol{\theta}) = \int_{\boldsymbol{y}} p(Y, \boldsymbol{y} \mid \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{y}$. The parameter $\boldsymbol{\theta}$ models factors like the difficulty of each task and the competence of each annotator etc. $\boldsymbol{\theta}$ is learned by maximizing the likelihood.

Detailed review of most existing work can be found in [21, 22, 46]. We briefly explain several representative models related to the work in this paper. Dawid and Skene (DS) [8] models $p(y_i^j \mid y_i)$ by a parameterizing a confusion matrix $v_{jkl} = p(y_i^j = l \mid y_i = k)$ for each annotator j, which can be understood as an annotator competence matrix, and models $p(y_i)$ by a categorical distribution $\tau_k = p(y_i = k)$; then the expectation maximization (EM) algorithm is employed to optimize model parameters v_{jkl} and τ_k. learning from crowd (LFC) [36] extends DS by adding a Dirichlet prior for $\boldsymbol{v_{jk}}$ and $\boldsymbol{\tau}$; again, the EM algorithm is employed to optimize model parameters. independent Bayesian classifier combination (iBCC) [23] is a Bayesian version of LFC; the Gibbs sampling method is used for parameter learning. generative model of labels, abilities, and difficulties (GLAD) [43] models $p(y_i^j \mid y_i)$ by a logistic function $\frac{1}{1+e^{-\alpha_i \beta_j}}$ where α_i is the difficulty of task i and β_j is the competence of annotator j, and models $p(y_i)$ by the same categorical distribution $\tau_k = p(y_i = k)$; model parameters are largely compressed into $M+N$ instead of $M \times K \times K$ as in DS; the parameters are inferred with the EM algorithm. multi-annotator competence estimation (MACE) [14] models $p(y_i^j = l \mid y_i = k)$ using a confusion matrix similarly with DS, where $v_{jkl} = (1-\theta_j)\epsilon_{jl}$ if $k \neq l$ and $v_{jkl} = \theta_j + (1-\theta_j)\epsilon_{jl}$ if $k = l$; again, this confusion matrix reduces the number of parameters; the EM algorithm is used for parameter learning. Although these approaches model the annotation generation process differently, it is difficult to determine which performs the best in practice [22]. Our model extends the independence assumption of these models and model the correlation of tasks in terms of relevance; further, it allows us to incorporate auxiliary information of tasks and prior knowledge on relevance.

There is also work jointly tackling label aggregation problems with other tasks such as downstream classification task and evaluation of retrieval systems. For example, Zhan et al. [45] proposed a joint classification and aggregation framework where the classification module provides feedbacks and boost the aggregation module. Ferrante et al. [10] use crowd relevance annotations as sources of uncertainty and design IR evaluation metrics based on the crowd annotations.

2.3 Gaussian Process for Aggregating Crowd Annotations

Several GP-based models have been proposed for label aggregation task [11, 31, 37, 38]. Groot et al. [11] aggregated crowd annotations of real-value type by averaging multiple crowd labels to one single label and apply a vanilla GP regression model. Rodrigues et al. [37] proposed a GP-based model to account for multiple annotators with different levels of expertise. Ruiz et al. [38] proposed a model of binary label aggregation by adding a GP prior on top of the confusion matrix in the DS model. A novel variational inference algorithm is proposed for the model. Further, the model is extended in order to deal with large-scale datasets, e.g., with approximately 1 million tasks, in the work of [31]. Our work is different from these models in the sense that it assumes a different annotation generation process, and accordingly, a different inference method is used to learn model parameters.

3 PRELIMINARIES: GAUSSIAN PROCESS CLASSIFICATION

The Gaussian process classification model is introduced to provide necessary knowledge to understand our model. Given a set of training samples $\boldsymbol{C} \triangleq \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \triangleq (\boldsymbol{X}, \boldsymbol{y})$, where N is the number of samples in $\boldsymbol{C}$, $\boldsymbol{x}_i$ is the input point, y_i is the corresponding label. The goal is to predicts label for a new point $\boldsymbol{x}_*$.

A GP is a stochastic process with the important characteristics that any finite number of random variables follow a joint Gaussian distribution [3]. A GP classification model assumes the observed data are generated through the following process: a GP first maps the input point $\boldsymbol{x} \in \mathbb{R}^D$ to a latent variable $f \in \mathbb{R}$, then a link function maps f to a real value $y \in [0,1]$. The link function can be a *logistic* function or a *probit* function. We use a probit function $\Phi(f) \triangleq \int_{-\infty}^{f} \mathcal{N}(z \mid 0,1)\, \mathrm{d}z$ in the work of this paper. In the binary classification setting, we denote the positive class as 1 and negative class as 0. Hence, according to the property of $\Phi(f)$, the positive class probability is $p(y=1 \mid \boldsymbol{x}) = \Phi(f)$ and the negative class probability is $p(y=0 \mid \boldsymbol{x}) = \Phi(-f)$.

Formally, the data generation process is rephrased as follows. First, the latent variable $\boldsymbol{f}$ follows a GP, denoted by:

$$\boldsymbol{f} \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\right) . \tag{1}$$

Second, each observed variable y_i follows a Bernoulli distribution conditioned on its corresponding latent variable f_i, denoted by:

$$y_i \mid f_i \sim Bernoulli\left(\Phi(f_i)\right) . \tag{2}$$

Now let us calculate the predictive probability of a new point $\boldsymbol{x}_*$ being classified as positive $p(y_* = 1 \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$. We need to first compute the posterior distribution of the latent variable f_* when the training data $\boldsymbol{C}$ being observed using

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f})\, p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})\, \mathrm{d}\boldsymbol{f} , \tag{3}$$

and then compute the positive class probability using

$$\begin{aligned} p(y_* = 1 \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) &= \Phi\left(\int f_* p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})\, \mathrm{d}f_*\right) \qquad (4) \\ &= \Phi(\mathbb{E}[f_*]) . \end{aligned}$$

Equation (4) is easy to calculate if knowing $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$. Next, we discuss how to calculate $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f})$ and $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$ in Equation (3). Given the training inputs and the new input point $[\boldsymbol{X}, \boldsymbol{x}_*]$, the corresponding latent variable $[\boldsymbol{f}, f_*]$ follows a multivariate Gaussian distribution

$$\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \Bigg| \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{x}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \mu_* \end{bmatrix}, \begin{bmatrix} K & K_* \\ K_*^T & K_{**} \end{bmatrix}\right) \tag{5}$$

where

$$\boldsymbol{\mu} \triangleq m(\boldsymbol{X}), \mu_* \triangleq m(x_*)$$

$$K \triangleq \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

$$K_*^T \triangleq \left[k(\boldsymbol{x}_1, \boldsymbol{x}_*), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x}_*)\right], K_{**} \triangleq k(\boldsymbol{x}_*, \boldsymbol{x}_*)$$

$\boldsymbol{\mu}$ is the mean vector of X, μ_* is the mean value at x_*, K is the covariance matrix of X, K_*^T is the covariance vector between X and $\boldsymbol{x}_*$, and finally K_{**} is the covariance between $\boldsymbol{x}_*$ and $\boldsymbol{x}_*$.

Based on the conditional Gaussian distribution rule [35], the distribution of f_* conditioned on $\boldsymbol{f}$ is

$$f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{\mu}_{f_*}, \boldsymbol{\sigma}_{f_*}\right) , \tag{6}$$

where $\boldsymbol{\mu}_{f_*} = m(\boldsymbol{x}_*) + K_* K^{-1}(\boldsymbol{f} - m(\boldsymbol{X}))$, $\boldsymbol{\sigma}_{f_*} = K_{**} - K_*^T K^{-1} K_*$. Now let us approximate $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$, the joint posterior distribution of the latent variables. By using Bayes rule, $p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y})$ becomes

$$p(\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{f} \mid X)\, p(\boldsymbol{y} \mid \boldsymbol{f})}{p(\boldsymbol{y} \mid X)} . \tag{7}$$

The prior $p(\boldsymbol{f} \mid \boldsymbol{X})$ is presented in Equation (5), and the likelihood $p(\boldsymbol{y} \mid \boldsymbol{f})$ is

$$p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_{i=1}^{N} p(y_i \mid f_i) = \prod_{i=1}^{N} \Phi\left((-1)^{(1-y_i)} f_i\right) . \tag{8}$$

However, the computation of the evidence $p(\boldsymbol{y} \mid X)$ is not trivial, because it is not a Gaussian distribution due to the multiplication of the Bernoulli likelihood with the GP prior. The solution can be either analytic approximations of integrals (e.g. expectation propagation (EP) and Laplace approximation) or numerical approximation methods (e.g. Monte Carlo sampling), where a Gaussian distribution is used to approximate the posterior distribution $p(\boldsymbol{f} \mid X, \boldsymbol{y})$ and the evidence $p(\boldsymbol{y} \mid X)$ [35].

In general, the mean function $m(\cdot)$ and the covariance function $k(\cdot, \cdot)$ of the GP classification model both contains parameters to be learned. To learn these parameters, one needs to maximize the likelihood of the observed data, which is $\mathrm{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{y} \mid X, \boldsymbol{\theta})$; or adopt a Bayesian view by considering both the likelihood and the prior of parameters, which turns to be $\mathrm{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{y} \mid X, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$ Note that here we rewrite $p(\boldsymbol{y} \mid X)$ as $p(\boldsymbol{y} \mid X, \boldsymbol{\theta})$ when we explicitly take $\boldsymbol{\theta}$ into consideration. The prior item $p\ (\boldsymbol{\theta})$ can avoid model overfitting and the likelihood $p\ (\boldsymbol{y} \mid X, \boldsymbol{\theta})$ "incorporates a trade-off between model fit and model complexity" [35].

4 GAUSSIAN PROCESS CLASSIFICATION ON CROWD ANNOTATIONS

4.1 Problem Formulation

Different from the vanilla GP classification model where in the observed date each example has one single label, in the crowd annotation case, each task has labels annotated by multiple annotators. Assume that there are N unique tasks and M unique annotators in the crowd annotation dataset. Denote the annotations for the i-th task by $\boldsymbol{C}_i \triangleq \{(\boldsymbol{x}_i, y_i^1), (\boldsymbol{x}_i, y_i^2), ..., (\boldsymbol{x}_i, y_i^{M_i})\}$, and denote the complete annotations by $\boldsymbol{C} \triangleq \{\boldsymbol{C}_1, \boldsymbol{C}_2, ..., \boldsymbol{C}_N\}$. Our goal is to infer the true label y_i for each task $\boldsymbol{x}_i$. Moreover, we also want to select new tasks for future crowd annotation.

4.2 The Model

In this section we propose three hypotheses for crowd annotation generation process based on which we propose the GP classification model for crowd annotations named CrowdGP.

Hypothesis 1 (Correlation between tasks). *A task is correlated with other tasks close to it in some space in terms of their labels.*

In the domain of IR this hypothesis is reflected as the widely accepted *cluster hypothesis* which states that closely associated documents tend to be relevant to the same information need [28]. We use a GP to capture the correlation between tasks, presented formally as:

$$\boldsymbol{f} \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\right), \tag{9}$$

where $\boldsymbol{f} \triangleq [f_1, f_2, ..., f_N]$ are continuous values and can be converted to discrete labels through a link function like probit, the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ captures the correlation across tasks, and the mean function $m(\boldsymbol{x})$ captures prior knowledge on labels.

Hypothesis 2 (Noise from tasks and annotators). *Both the noise from tasks and the noise from annotators affect the observed crowd labels.*

Intuitively if a task is easy, different crowd annotators tend to reach a consensus which leads to the same crowd labels; otherwise, the crowd labels will be very different. To this end, we use a Gaussian variable ϵ_i to model the noise of each task T_i:

$$\epsilon_i \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right), \tag{10}$$

where μ_i models the inherent bias of the task towards relevance and σ_i^2 models the difficulty level of the task.

Similarly, the competence of an annotator determines the quality of his or her annotations. Besides, an annotator has his or her own criterion or bias of relevance assessment. Similarly to ϵ_i, we use another Gaussian variable ϵ^j:

$$\epsilon^j \sim \mathcal{N}\left(\mu_j, \sigma_j^2\right), \tag{11}$$

where μ_j models the bias of the annotator towards relevance, and σ_j^2 models his or her competence.

Hypothesis 3 (Annotation generation process). *An observed crowd label (discrete) is generated from a Bernoulli distribution, of which the positive probability is determined by three variables: the latent GP, the task noise and the annotator noise.*

Based on the three hypotheses, we assume the observed crowd annotations are generated through the following process. Each task T_i corresponds to a latent variable $f_i \in \mathbb{R}$, and the latent variables for all the tasks conform to a GP. When a task T_i is distributed to an annotator A_j, a Gaussian noise ϵ_i is added to f_i to generate a_i^j, and a Gaussian noises ϵ^j is added to a_i^j to generate b_i^j. We assume that ϵ_i is independent from f_i and any other noise $\epsilon_{l(l \neq i)}$; and ϵ^j is independent from any f_i, any ϵ_i, and any other $\epsilon^{k(k \neq j)}$. A probit function maps b_i^j to a value in interval $[0, 1]$ which represents the probability of y_i^j being relevant, denoted by $\Phi(b_i^j)$. A crowd label y_i^j is generated from the Bernoulli distribution B$ernoulli\,(\Phi(b_i^j))$. The process is illustrated in Figure 1.

4.3 Label Inference for Crowd Tasks

Given a new point or an existing point $\boldsymbol{x}_*$, our goal is to predict the latent true label y_*. Similar to the vanilla GP classification model (Equation (4)), we calculate the positive class probability:

$$p(y_* = 1 \mid \boldsymbol{x}_*, X, Y) = \Phi\left(\int f_* p(f_* \mid \boldsymbol{x}_*, X, Y)\, \mathrm{d}f_*\right) \tag{12}$$
$$= \Phi(\mathbb{E}[f_*]) .$$

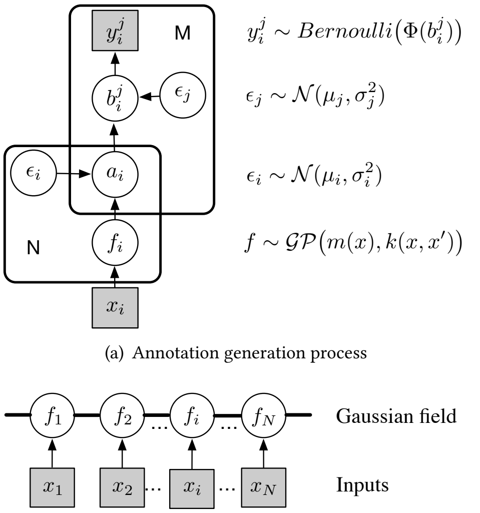
The integral item $p(f_* \mid \boldsymbol{x}_*, X, Y)$ is further rewritten as:

$$p(f_* \mid \boldsymbol{x}_*, X, Y) = \int p(f_* \mid \boldsymbol{x}_*, X, \boldsymbol{f})\, p(\boldsymbol{f} \mid X, Y)\, \mathrm{d}\boldsymbol{f} . \tag{13}$$

The integral item $p(\boldsymbol{f} \mid X, Y)$ can be further rewritten using the Bayes' rule as:

$$p(\boldsymbol{f} \mid X, Y) = \frac{p(\boldsymbol{f} \mid X)\, p(Y \mid \boldsymbol{f})}{\int p(\boldsymbol{f} \mid X)\, p(Y \mid \boldsymbol{f})\, \mathrm{d}\boldsymbol{f}} = \frac{p(\boldsymbol{f} \mid X)\, p(Y \mid \boldsymbol{f})}{p(Y \mid X)} . \tag{14}$$

The prior $p(\boldsymbol{f} \mid X)$ is the GP prior in Equation (5). The likelihood of the observed crowd annotations – $p(Y \mid \boldsymbol{f})$ – consists of multiple


y_i^j
M
b_i^j
ϵ_j
ϵ_i
a_i
N
f_i
x_i
$y_i^j \sim Bernoulli(\Phi(b_i^j))$
$\epsilon_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$
$\epsilon_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$
$f \sim \mathcal{GP}(m(x), k(x, x'))$
f_1
f_2
f_i
f_N
Gaussian field
x_1
x_2
x_i
x_N
Inputs

(a) Annotation generation process

(b) GP prior

Figure 1: Graphical model for the CrowdGP model. Squares represent observed variables and circles represent latent variables. Figure 1(b) illustrates a set of fully connected nodes following a GP. Figure 1(a) illustrates the annotation generation process.

Bernoulli likelihoods. Based on the independence assumption in the annotation generation process, $p(Y \mid \boldsymbol{f})$ can be written as:

$$\begin{aligned} p(Y \mid \boldsymbol{f}) &= \prod_{i=1}^{N}\prod_{j=1}^{M}\int \mathcal{N}\left(b_i^j \mid f_i + \mu_i + \mu_j, \sigma_i^2 + \sigma_j^2\right) \\ &\quad \Phi\left((-1)^{(1-y_i^j)} b_i^j\right) \mathrm{d}b_i^j \,. \end{aligned} \tag{15}$$

We give the detailed derivation as below, which is one of the contribution of this work.

$$p(Y \mid \boldsymbol{f}) = \prod_{i=1}^{N}\prod_{i=1}^{M} p\left(y_i^j \mid f_i\right) \tag{16a}$$

$$p\left(y_i^j \mid f_i\right) = \iint p\left(y_i^j \mid a_i^j, b_i^j, f_i\right) p\left(a_i^j, b_i^j | f_i\right) \mathrm{d}a_i^j \, \mathrm{d}b_i^j \tag{16b}$$

$$= \iint p\left(a_i^j \mid f_i\right) p\left(b_i^j \mid a_i^j\right) p\left(y_i^j \mid b_i^j\right) \mathrm{d}a_i^j \, \mathrm{d}b_i^j \tag{16c}$$

$$= \iint \mathcal{N}\left(a_i^j \mid f_i + \mu_i, \sigma_i^2\right) \mathcal{N}\left(b_i^j \mid a_i^j + \mu_j, \sigma_j^2\right) \Phi\left((-1)^{1-y_i^j} b_i^j\right) \mathrm{d}a_i^j \, \mathrm{d}b_i^j \tag{16d}$$

$$= \int \mathcal{N}\left(b_i^j \mid f_i + \mu_i + \mu_j, \sigma_i^2 + \sigma_j^2\right) \Phi\left((-1)^{(1-y_i^j)} b_i^j\right) \mathrm{d}b_i^j \tag{16e}$$

Equation (16b) applies the total probability rule; Equation (16c) holds because a_i^j is only dependent on f_i, b_i^j is only dependent on a_i^j, and y_i^j is only dependent on b_i^j; Equation (16d) holds because the sum of a constant f_i and a Gaussian variable ϵ_i is a Gaussian variable, similarly the sum of a constant a_i^j and a Gaussian variable ϵ_j is a Gaussian variable; Equation (16e) integrates a_i^j out by applying the Gaussian marginal and conditional rule [32, page 93].

Now we continue the discussion of Equation (14). Note that $p(\boldsymbol{f} \mid \boldsymbol{X}, Y)$ is not a Gaussian distribution due to the multiplication of the Bernoulli likelihood with the GP prior and thus the computation of the integral is not trivial. The major idea is to use a multivariate Gaussian distribution $q(\boldsymbol{f})$ to approximate $p(\boldsymbol{f} \mid \boldsymbol{X}, Y)$. The problem is solved together with the optimization of model parameters (see Section 4.5).

4.4 Selection of New Tasks

Except for inferring latent true labels for tasks that have crowd annotations, we are also interested in selecting new tasks for future annotation. Finding all relevant document for relevance assessment is one of the goals of building a high-quality test collection [20]. Note that the CrowdGP model outputs a predictive Gaussian distribution given a new point which is shown in Equation (6), making it possible to apply existing acquisition functions for new task search.

An *acquisition function* is a scoring function in a search space, which finds an optimal trade-off between exploration (the predicted variance is high) and exploitation (the predicted mean is high). In this work, we use expectation improvement (EI) [9] as the acquisition function, defined as:

$$\alpha(\boldsymbol{x}) \triangleq \mathbb{E}(max(f(\boldsymbol{x}), \beta)) \tag{17}$$

$\boldsymbol{x}$ is a new point, $f(\boldsymbol{x})$ is a random variable conforming to the predictive Gaussian distribution. $\alpha(\cdot)$ is the acquisition function. A negative β means exploration and a positive β means exploitation. For example, if β is negative, given two points that have the same predicted mean value but different predicted variance values, EI will prioritize the point with big variance.

4.5 Model Optimization

The proposed model contains parameters from the mean function and the covariance function which are common for all GP models, as well as parameters $\{(\mu_i, \sigma_i^2) \mid i = 1, \cdots, N\}$ from tasks and $\{(\mu_j, \sigma_j^2) \mid j = 1, \cdots, M\}$ from annotators. We denote all the parameters by $\boldsymbol{\theta}$. Our goal is to optimize the model with regard to $\boldsymbol{\theta}$. Similar to the vanilla GP classification model, we adopt a Bayesian view and maximize the log likelihood of the observed data plus the log of the parameter prior: $\log p(Y, \boldsymbol{\theta} \mid \boldsymbol{X}) = \log p(Y \mid \boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. We formally write the optimization problem as:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \{\log p(Y \mid \boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\} \,. \tag{18}$$

As the first part $\log p(Y \mid \boldsymbol{X}, \boldsymbol{\theta}) = \int p(\boldsymbol{f} \mid \boldsymbol{X}) p(Y \mid \boldsymbol{f}) \mathrm{d}\boldsymbol{f}$ is intractable, which is explained in the discussion of Equation (14), we instead maximize its evidence lower bound (ELBO), which is tractable. The derivation of ELBO is as follows:

$$
\begin{aligned}
\log p(Y \mid X, \boldsymbol{\theta}) &\triangleq \log p(Y) = \int q(f) \log p(Y)\, \mathrm{d}f \\
&= \int q(f) \log \frac{p(Y, f)}{q(f)}\, \mathrm{d}f + \left(-\int q(f) \log \frac{p(f \mid Y)}{q(f)}\, \mathrm{d}f \right) \\
&= \int q(f) \log \frac{p(Y, f)}{q(f)}\, \mathrm{d}f + KL(q(f) \,||\, p(f \mid Y)) \\
&\geqslant \int q(f) \log \frac{p(Y, f)}{q(f)}\, \mathrm{d}f \\
&= \mathbb{E}_{q(f)} [\log p(Y \mid f)] - KL[q(f) \,||\, p(f)] \\
&\triangleq ELBO\,, \qquad (19)
\end{aligned}
$$

where $q(f) \triangleq q(f \mid \boldsymbol{\psi})$ is the parameterized variational functional to be learned, which is assumed a multivariate Gaussian distribution approximating $p(f \mid Y)$; $p(f) \triangleq p(f \mid \boldsymbol{\theta})$ is the prior distribution in Equation (5) and $p(Y \mid f) \triangleq p(Y \mid f, \boldsymbol{\theta})$ is the likelihood of the observed data in Equation (15), both parameterized by $\boldsymbol{\theta}$. Therefore we also denote $ELBO \triangleq ELBO(\boldsymbol{\psi}, \boldsymbol{\theta})$.

Finally, we adopt the VEM algorithm [33] to maximize the objective function, which is reduced to $ELBO(\boldsymbol{\psi}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$. Both the E and M steps maximize the same function, the difference is that the E step maximizes it with respect to the parameters of $q(f)$ while the M step maximizes it with respect to the model parameters $\boldsymbol{\theta}$. The optimization method is summarized in Algorithm 1.

Algorithm 1: Optimization of the CrowdGP model.

```
Input: A set of task features and crowd annotations $(X, Y)$.
Output: $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$.
Initialize $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$;
while not converge do
    E step Fix $\boldsymbol{\theta}^{old}$ and maximize
      $ELBO(\boldsymbol{\psi}, \boldsymbol{\theta}^{old}) + \log p(\boldsymbol{\theta}^{old})$ with respect to $\boldsymbol{\psi}$ to get
      new $\boldsymbol{\psi}^{new}$;
    M step Fix $\boldsymbol{\psi}^{new}$ and maximize
      $ELBO(\boldsymbol{\psi}^{new}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ to get
      $\boldsymbol{\theta}^{new}$;
    Check for convergence;
end
```

4.6 Task Representation

Different from most label aggregation models, where tasks are represented as a set of indicators of $\{1, 2, \ldots, N\}$, the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ in CrowdGP requires the tasks to be represented as a set of vectors of $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$. The vectors can incorporate any auxiliary information. In this work, we use lexical features, semantic features and ranking features as they are easy to acquire for most and have been demonstrated effective in IR related tasks.

Lexical features We consider several lexical features which have been shown effective in learning to rank algorithms [24]: (1) a term frequency score measuring the frequency of a query in a document (denoted by $\sum_{t \in q \cap d} TF(t, d)$), (2) a inverse document frequency score measuring how much information a query provides ($\sum_{t \in q} IDF(t)$), (3) a TF-IDF score measuring both (denoted by $\sum_{t \in q \cap d} TF(t, d)\, IDF(t)$), (4) a cosine value between TF-IDF vectors of a query and a document, measuring their lexical similarity, (5) a BM25 score between a query and a document, measuring their lexical similarity ($\sum_{t \in q} IDF(t)\, \frac{TF(t,d)\,(k_1+1)}{TF(t,d)+k_1\,(1-b+b\, \frac{|d|}{avgdl})}$), and (6) a probability of observing a query given a document, which is based on language modelling method and measures lexical their similarity [44] (denoted by $\sum_{t \in q \cap d} \log \frac{p(t|d)}{\alpha_d\, p(t|C)} + |q| \log \alpha_d + \sum_t \log p(t \mid C)$). In the aforementioned formulas, q is the query, d is the document, $| \cdot |$ is the number of terms, $avgdl$ is the average document length in a document collection, C is the document collection, k_1, b, and α_d are hyperparameters with default values of 1.2, 0.75, and 0.5.

Semantic features Semantic features are important supplementary of lexical features to capture task correlation. To acquire relevance-guided representation of query-document text, we pre-train a text representation model (*BERT-FirstP*) following [7] because it uses the same ClueWeb09 collection with our crowdsourcing dataset and it is a relevance classification task. We input `query [SEP] document` and output the vector of the `[CLS]` token as the semantic features.

Ranking features Except for lexical and semantic features, it is easy to acquire document rank information from multiple retrieval systems for a relevance crowdsourcing task. Combining multiple ranked lists produced by various retrieval systems (known as *meta search*) does help relevance classification [4]. We use the available ranked lists that covers the queries and documents of our crowdsourcing datasets, 35 in total, produced by the participating teams in the TREC 2009 million query track.

4.7 Mean and Covariance Function

The mean function of CrowdGP gives a set of latent function values without seeing any crowdsourcing data. We can use any pretrained relevance classification model as the mean function, in order to incorporate prior relevance knowledge. For idea validation, we simply pretrain a logistic regression model using the ground truth relevance labels in the corresponding training set of the crowdsourcing dataset, and use the predicted logits as the mean function value.

The covariance function of CrowdGP measures the correlation between two tasks. We employ the linear covariance $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \boldsymbol{x} \cdot \boldsymbol{x}'$ for semantic features and the RBF covariance $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{l^2})$ for lexical and ranking features. The length scale parameter l indicates with what scale the changes of input will not cause "large" change in output; the signal variance σ indicates the amplitude of the latent function.

4.8 Implementation

We employ GPflow [29] to implement our model. The model contains a number of parameters including the length scale l and variance σ of the covariance function, the mean and variance of the Gaussian noise for the annotators $\{(\mu_j, \sigma_j^2) \mid j = 1, \cdots, M\}$, the mean and variance of the Gaussian noise for the tasks $\{(\mu_i, \sigma_i^2) \mid i = 1, \cdots, N\}$. Their initial values are set to be $l = 1$, $\sigma = 1$, $\mu_i = \mu_j = 0$, and $\sigma_i = \sigma_j = 1$ respectively. We set $\mathcal{N}(0, 1)$ as the prior for all the mean parameters, and *Gamma* $(1, 1)$ as the prior for the length scale parameter and all the variance parameters. We use Adam [18] as the gradient descent optimizer in both the E step and the M step in Algorithm 1. Training step is set to 5000 to make sure CrowdGP converges. The code is publicly available.[1]

5 EXPERIMENTS

5.1 Research Questions

In the remainder of the work we aim to answer the following research questions:

RQ1 How does the model perform in terms of inferring latent true labels from crowd annotations compared with baselines?

RQ2 How does the model perform in terms of selecting new tasks for future crowd annotation?

RQ3 How do the auxiliary information of tasks (via task feature) and prior relevance knowledge (via mean function) affect model performance?

RQ4 How do different optimization methods and different initialization of model hyperparameters affect the learning curve?

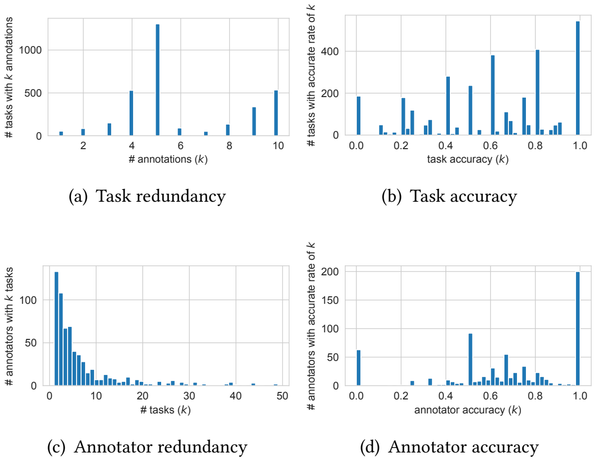
5.2 Experimental Setup

5.2.1 *Dataset.*

We evaluate CrowGP on two crowdsourcing datasets: the crowdsourcing dataset of the TREC 2010 relevance feedback track [5] (CS2010[2]), and the crowdsourcing development dataset of TREC 2011 crowdsourcing track aggregation task (CS2011[3]). Each example is a tuple of query ID, document ID, annotator ID, ground truth label, and crowd label. The queries are from the TREC 2009 million query track [6] and the documents are from the ClueWeb09 collection. We also use the corresponding document ranks produced from the participation retrieval systems in the TREC 2009 million query track. The original CS2010 dataset contains 100 queries and 19,902 documents, and in total 20,232 unique query-document pairs and 96,883 relevance annotations given by 766 annotators. The ground truth labels were given by NIST experts in previous TREC tracks. We remove examples that are marked as invalid due to reasons such as broken links, or that have no corresponding ground truth label, or that have no text or ranks available for the document. Consequently, there remains 3,275 unique query-document pairs and 18,479 relevance annotations with ground truth labels. As our model is designed for binary labels, we turn the original ternary scale into a binary scale by mapping highly relevant or relevant labels to relevant labels. The original CS2011 dataset contains 25 queries and 3,557 documents, and in total 3,568 unique query-document pairs and 10,752 binary relevance annotations given by 181 annotators. The ground truth labels are also from NIST experts. Similarly, invalid annotations are removed, resulting in 711 unique query-document pairs and 2,181 relevance annotations with ground truth labels. The statistics of the two datasets after preprocessing are shown in Table 1.

Table 1: Statistics of two crowdsourcing datasets.

Data set	CS2010	CS2011
# topic	100	25
# task	3,275	711
# rel	1,775	589
# nonrel	1,500	122
# annotator	722	181
# annotation	18,479	2,181



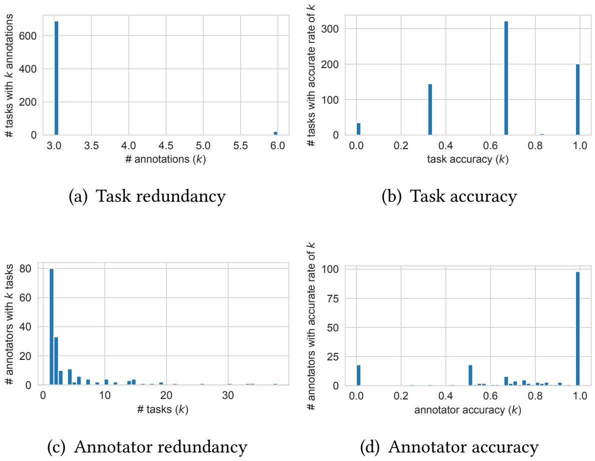
(a) Task redundancy

(b) Task accuracy

(c) Annotator redundancy

(d) Annotator accuracy

Figure 2: Annotation distribution of CS2010.



(a) Task redundancy

(b) Task accuracy

(c) Annotator redundancy

(d) Annotator accuracy

Figure 3: Annotation distribution of CS2011.

To understand the annotation distribution and quality of the two datasets, we are interested in: (1) how many redundant crowd annotations are collected for each task (*task redundancy*), (2) how accurate the crowd annotations are for each task (*task accuracy*), (3) how many annotations each annotator gives (*annotator redundancy*), and (4) how accurate each annotator is (*annotator accuracy*). Formally, we define the *task redundancy* of task i as the number of

[1] https://github.com/dli1/magp
[2] https://www.ischool.utexas.edu/~ml/data/trec-rf10-crowd.tgz
[3] https://sites.google.com/site/treccrowd/2011

its crowd annotations, denoted by M_i; following [21], we define the *task accuracy* of task i as the accurate rate of its crowd annotations, denoted by $\frac{\sum_{j=1}^{M_i} \mathbb{1}(y_i^j = y_i^*)}{M_i}$; we define the *annotator redundancy* of annotator j as the number of crowd annotations he or she gives, denoted by N_j; following [21], we define the *annotator accuracy* of annotator j as the accurate rate of his or her crowd annotations, denoted by $\frac{\sum_{i=1}^{N_j} \mathbb{1}(y_i^j = y_i^*)}{N_j}$. We plot the histograms in Figure 2 and 3. Overall, the quality of the crowd annotations of CS2011 is better than CS2010.

5.2.2 Baselines. In order to evaluate its capability of inferring true labels given crowd annotations, we compare CrowdGP with MV and four popular PGMs which models annotators and tasks through different annotation generation assumptions, including DS [8], LFC [36], MACE [14], and GLAD [43]. Furthermore, as our CrowdGP model allows using existing training crowd annotations (when pretraining the mean function) and auxiliary task information (task features), for fair comparison, we propose an intuitive approach – a Classifier – where the auxiliary information of tasks and crowd annotations are used in a supervised way to infer true labels of tasks given their crowd annotations. Finally, to understand to role the GP prior and the annotation generation process in CrowdGP, we propose two CrowdGP variants: Gaussian process with majority-voted label (MVGP) removes the annotation generation process component, and likelihood (LK) removes the GP prior component.

MV MV is a straightforward method to aggregate crowdsourcing annotations. The label with the highest number of votes among the annotators is considered as the true label. Annotators are treated with equal weights, which is not the truth in reality.

DS [8] DS is a widely used PGM for label aggregation. It models the true label of tasks with a multinomial distribution of K classes. It models the competence of each annotator with a $K \times K$ confusion matrix, where K is the number of class. In total, there are $M \times K \times K$ parameters for annotators and K parameters for tasks. It is an un-supervised method and only the crowd annotations are needed to infer the true labels. In the experiments we use the implementation of Zheng et al. [46].

LFC [36] LFC extends DS by adding a Dirichlet prior over its parameters. In the experiments we use the implementation of Zheng et al. [46].

MACE [14] MACE reduces the parameterized confusion matrix of DS into a parameterized vector. As a compensation, it introduces a new variable that indicates whether an annotator j is spamming on task i. The number annotator parameters is $M \times K + M$. In the experiments we use the implementation of Hovy et al. [14].

GLAD [43] GLAD models task difficulty and annotator competence with scale-value parameters. Annotator parameters are largely compressed into $M + N$ (M parameters for annotators and N parameters for tasks). In the experiments we use the implementation of Zheng et al. [46].

Classifier As a set of crowd annotations with ground truth labels are available, it is also possible to model the task of inferring true labels given crowd annotations as a supervised classification task. An intuitive way is to use crowd annotations for task features and train a simple classifier such as logistic regression model. The features of the Classifier consist of two parts: the features which are the same with that used in CrowdGP (lexical/semantic/ranking), and the features constructed using crowd annotations. We construct the second type of features in the following way: assume that for each query-document pair a set of crowd annotations are available, we use the mean, standard variation, median, maximum, and minimum values of the crowd annotations as the features. For example, assume a query-document is associated with 5 crowd annotations 1,1,1,0,0, the second type of features will be [0.6, 0.5, 1, 1, 0]. The goal of this baseline is to explore another way of utilizing existing crowd annotations. CrowdGP utilizes crowd annotations by calculating the likelihood of a probabilistic generative model, while the Classifier baseline utilizes crowd annotations as its features. We implement this baseline ourselves.

MVGP MVGP is a simplified version of CrowdGP. It is the same with CrowdGP except that the multiple crowd annotations are replaced with the single label aggregated by MV. It can be viewed as a vanilla GP classification model. The goals is to examine how important is the role the crowd annotations are playing in CrowdGP. We implement this baseline ourselves.

LK LK is also a simplified version of CrowdGP. It removes the GP prior in CrowdGP and only keeps the annotation generation process part (see Figure 1(a)). The goals is to examine how important is the role the GP prior is playing in CrowdGP. We implement this baseline ourselves.

5.3 Label Inference for Crowd Tasks (RQ1)

In this experiment we study whether CrowdGP is able to correctly infer the latent true labels of tasks, and whether it performs better than the baseline models. We run the experiment on the both the CS2010 and CS2011 datasets and report the *accuracy* and the *F1* score. Note that Classifier needs extra data containing crowdsourcing relevance annotations, and CrowdGP allows a pretrained mean function which needs to be trained on extra data containing ground truth relevance labels, therefore the corresponding training sets are used. The rest of the models are unsupervised and thus only the test set is used. For the CrowdGP model, we use all the three task features and the pretained mean function.

We conducted 5-folds cross validation on both the two datasets. Table 2 shows the mean values of accuracy and F1 over the test sets of the 5-folds cross validation for all the models. First, MV is a strong baseline. The three PGMs (DS, LFC, and MACE) perform better than MV on CS2010 and worse than MV on CS2011, while GLAD performs worse than MV on CS2010 and better than MV on CS2011. The five models only take crowd annotations as input instead of any auxiliary information of tasks. MV simply treats each worker equally while the four PGMs model tasks and annotators in different ways to improve the quality of the inferred labels, however, the performance of the four PGM is not consistently better than MV. Similar finding can be found in [22, page 22], where MV and PGMs such as DS and GLAD are compared on 9 benchmark datasets, and

MV is found a strong baseline compared against the rest PGMs. This indicates that we may need to resolve to auxiliary information of tasks or annotators, or other prior knowledge of the labels to improve label inference performance.

Second, CrowdGP performs consistently better than MV. It also performs the best among all the baselines. Furthermore, paired t-test indicates that MVGP and CrowdGP performs significantly better than MV, which is marked bold in Table 2. The effectiveness is mainly attributed to the GP prior instead of the annotation generation process, which can be seen from the fact that MVGP performs much better than MV, while LK performs not consistently better than MV. For MVGP even though multiple crowd annotations are reduced to one annotation by MV the task auxiliary information is still sufficient to learn task correlation for better label inference. The merit of CrowdGP is that it allows to integrate any auxiliary information of tasks through the covariance function of the GP prior, on top of which a PGM that fits the label distribution of crowdsourcing data is stacked. We compare the CrowdGP with the Classifier. The Classifier is another intuitive way of use task auxiliary information and crowdsourcing data, both of which are treated equally as features. We find the CrowdGP performs better than the Classifier. Moreover, it also points out a direction for future work: adding a GP prior to existing PGMs like DS in order to adapt to different crowdsourcing tasks.

To sum up, the CrowdGP model shows consistent improvement compared with strong baseline MV, and also performs the best among all the baselines. The effectiveness is mainly attribute to the GP prior.

Table 2: Inferring true labels for crowd tasks.

	CS2010		CS2011	
	Acc	F1	Acc	F1
MV	0.63	0.72	0.74	0.84
DS [8]	0.67	0.70	0.65	0.77
LFC [36]	0.67	0.71	0.67	0.79
MACE [14]	0.69	0.73	0.67	0.78
GLAD [43]	0.57	0.71	0.79	0.88
Classifier	0.64	0.67	0.69	0.78
LK	0.67	0.67	0.64	0.75
MVGP	**0.64**	**0.73**	**0.78**	**0.86**
CrowdGP (ours)	**0.71**	**0.78**	**0.89**	**0.94**

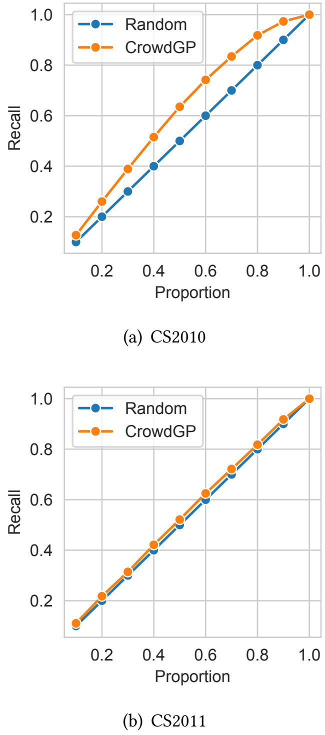
5.4 New Tasks Selection (RQ2)

In this experiment we explore another important functionality of CrowdGP. Given that CrowdGP is applied (trained) on a dataset consisting of crowdsourcing annotations, is it able to effectively select new tasks (i.e. query-document pairs) that are potentially positive and therefore relevant for future annotation? The task is interesting as finding all relevant document is one of the goals of building a high-quality test collection [20].

The two datasets of CS2010 and CS2011 are used. We first train the CrowdGP model on the corresponding training set, and then we rank tasks in the test set according to their EI scores (Equation (17)) and select a top proportion (10% – 100%). The selected tasks are assumed to acquire crowd annotation in practice. In our experiment, we treat the selected tasks as positive and report the *recall* scores at different selection proportions.

Figure 4 shows the recall-cost curve of CrowdGP and random sampling. In general, CrowdGP is more effective than random sampling in recalling tasks that have relevant labels. The effectiveness is less obvious on CS2011 than on CS2010, due to the fact that the proportion of relevant tasks on CS2011 is as high as 87% and thus the recall increases almost linearly with selection proportion.

To sum up, except for inferring the latent true labels from crowd annotation, the CrowdGP model can also effectively select potentially relevant tasks for crowd annotation. This functionality of CrowdGP helps to reduce crowdsourcing cost.



(a) CS2010

(b) CS2011

Figure 4: Performance of new task selection for CrowdGP and random sampling.

5.5 The Effect of Task Feature and Mean Function on Label Inference (RQ3)

In Section 5.3 we demonstrated that it is the GP prior of the CrowdGP model that contributes the most of the effectiveness. In this experiment we study the effect of the task features and the mean function on label inference performance, as they are key to determine what knowledge to incorporate in the GP prior.

We run different combination of three types of task features: lexical features, semantic features and ranking features introduced in Section 4.6. The other settings are same as the default setting

Table 3: The effect of task feature on label inference.

	CS2010		CS2011	
Feature	Acc	F1	Acc	F1
Lexical	0.63	0.77	0.83	0.91
Ranking	0.65	0.77	0.84	0.91
Semantic	0.70	0.78	0.88	0.93
Lexical+Ranking	0.68	0.78	0.83	0.90
Ranking+Semantic	0.69	0.76	0.89	0.94
Lexical+Semantic	0.71	0.79	0.88	0.93
Lexical+Ranking+Semantic	0.71	0.78	0.89	0.94

Table 4: The effect of mean function on label inference.

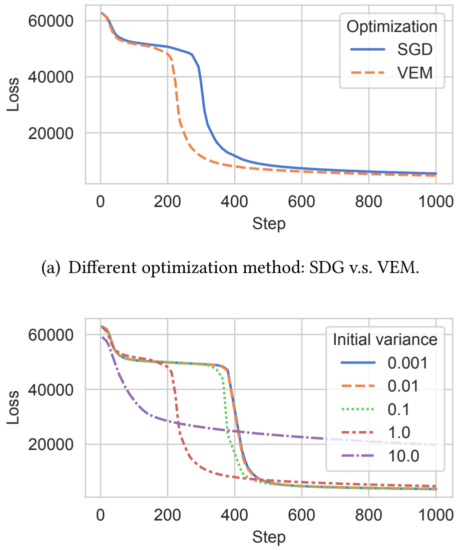
	CS2010		CS2011	
Mean function	Acc	F1	Acc	F1
Zero	0.64	0.75	0.84	0.91
Linear	0.69	0.77	0.89	0.94
Pretrain	0.71	0.78	0.89	0.94

of CrowdGP (we use the pretrained mean function). We run the experiment in a 5-folds cross validation setting on both CS2010 and CS2011 and report the *accuracy* and the *F1* score on the test sets.

Table 3 shows the performance of six different task feature combinations. It can be observed that on both CS2010 and CS2011 the semantic features perform better than the ranking features, and the ranking features performs better than the lexical features. The combination of the lexical and the semantic performs the best on CS2010 and the combination of ranking and semantic performs the best on CS2011. It indicates that the CrowdGP model is sensitive to task features and therefore careful designing of task features is necessary in application.

Similarly we examine three different mean functions: a zero function, a linear function, and a pretrained ranking function introduced in Section 4.7. The other settings are same as the default setting of CrowdGP (we use the all the lexical, the ranking and the semantic as the task features). We run the experiment in a 5-folds cross validation setting on both CS2010 and CS2011 and report the averaged *accuracy* score and *F1* score on the test sets. Table 4 shows the performance of three different mean functions. It can be observed that the linear mean function performs than the zero, and the pretrain performs better than the linear, which is expected because the pretrain introduced prior knowledge of relevance, the linear learn this knowledge of relevance from the given crowdsourcing annotations, and the zero does not learn any prior knowledge of relevance at all.

To sum up, both the task features and the mean function affect the CrowdGP model, and the task features has stronger effect than the mean function. The designing of task features and mean function can be adapted accordingly in specific crowdsourcing applications.



(a) Different optimization method: SDG v.s. VEM.

(b) Different hyperparameter initialization: $\sigma^2 \in \{0.001, 0.01, 0.1, 1, 10\}$.

Figure 5: Loss curves of different model hyperparameters.

5.6 Configuration Choices of CrowdGP (RQ4)

In this section we illustrate the learning behaviour of CrowdGP under different hyperparameters. The experiment is run on the test set of CS2010. Figure 5(a) shows the loss curve of two optimization method including SDG and VEM. The actual training step is 5000, for better visualisation we only present 1000 steps. It is observed that both optimization methods lead to convergence, and VEM converges faster and achieves lower loss value. Figure 5(b) shows the loss curve of 5 different initial values of the variances of the Gaussian variables for tasks and annotators. We only study the initialization of the variances as it is the key to determines task difficulty or annotator competence, we set the initial value of the means to be 0 as there is no prior knowledge on bias of task or annotator. Smaller σ^2 leads to slower convergence but lower loss value and bigger σ^2 leads to faster convergence but larger loss value.

To sum up, different model hyperparameters have very limited effect on model performance, but they do affect the convergence speed.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we study the problem of relevance inference from noisy crowd annotations. We propose a new PGM name CrowdGP. It assumes a new annotation generation process and models the true relevance labels of tasks, the difficulty and bias of tasks, and the competence and bias of annotators by using a Gaussian process and multiple Gaussian variables.

We evaluate CrowdGP on two datasets, the crowdsourcing dataset of the TREC 2010 relevance feedback track, and the crowdsourcing development dataset of the TREC 2011 crowdsourcing track. The CrowdGP model performs consistently better than majority voting in terms of interring true relevance labels for tasks that have crowd annotations, while several state-of-the-art baselines including DS, LFC, GLAD and MACE perform comparably with majority voting. The CrowdGP model is also effective in terms of selecting new tasks for crowdsourcing labelling. It is a new functionality which is not supported in DS, LFC, GLAD and MACE etc. Moreover, ablation studies demonstrate that the effectiveness is attributed to the modelling of task correlation based on the auxiliary information of tasks and the prior relevance information of documents to queries.

One of the future work can be the Bayesian modelling of existing PGMs. For example, a GP prior can be integrated with DS in order to adapt to different crowdsourcing tasks. Another improvement may be achieved with a better task selection approach. In the current work, new tasks are selected for crowdsourcing labelling by a cutting off an EI score list at the predefined threshold. It is interesting to study whether iterative search approaches like Bayesian optimization can select new tasks more effectively, similar to the work on multi-armed bandits by Losada et al. [25], Rahman et al. [34] and whether it leads to biased or unbiased annotations [20].

ACKNOWLEDGMENTS

This research was supported by the NWO Innovational Research Incentives Scheme Vidi (016.Vidi.189.039), the NWO Smart Culture - Big Data / Digital Humanities (314-99-301), the H2020-EU.3.4. - SOCIETAL CHALLENGES - Smart, Green And Integrated Transport (814961), the National Key R&D Program of China with grant No. 2020YFB1406704, the Natural Science Foundation of China (61972234, 61902219), and the China Scholarship Council.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Omar Alonso and Stefano Mizzaro. 2012. Using crowdsourcing for TREC relevance assessment. *Information processing & management* 48, 6 (2012), 1053–1066.

[2] Omar Alonso, Daniel E Rose, and Benjamin Stewart. 2008. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, Vol. 42. ACM New York, NY, USA, 9–15.

[3] Mauricio Álvarez, Lorenzo Rosasco, Neil Lawrence, et al. 2012. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning* 4, 3 (2012), 195–266.

[4] Javed A Aslam and Mark Montague. 2001. Models for metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* 276–284.

[5] Chris Buckley, Matthew Lease, and Mark D. Smucker. 2010. Overview of the TREC 2010 Relevance Feedback Track (Notebook). In *The Nineteenth Text Retrieval Conference (TREC) Notebook.*

[6] Ben Carterette, Virgiliu Pavlu, Hui Fang, and Evangelos Kanoulas. 2009. Million Query Track 2009 Overview.. In *TREC.*

[7] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 985–988.

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

[9] Laurence Charles Ward Dixon and Giorgio Philip Szegö. 1978. *Towards global optimisation 2.* North-Holland Amsterdam.

[10] Marco Ferrante, Nicola Ferro, and Maria Maistro. 2017. AWARE: Exploiting Evaluation Measures to Combine Multiple Assessors. *ACM Transactions on Information Systems (TOIS)* 36, 2 (2017), 20.

[11] Perry Groot, Adriana Birlutiu, and Tom Heskes. 2011. Learning from multiple annotators with Gaussian processes. In *International Conference on Artificial Neural Networks.* Springer, 159–164.

[12] Lei Han, Eddy Maddalena, Alessandro Checco, Cristina Sarasua, Ujwal Gadiraju, Kevin Roitero, and Gianluca Demartini. 2020. Crowd worker strategies in relevance judgment tasks. In *Proceedings of the 13th International Conference on Web Search and Data Mining.* 241–249.

[13] Lei Han, Kevin Roitero, Ujwal Gadiraju, Cristina Sarasua, Alessandro Checco, Eddy Maddalena, and Gianluca Demartini. 2019. All those wasted hours: On task abandonment in crowdsourcing. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.* 321–329.

[14] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 1120–1130.

[15] Oana Inel, Giannis Haralabopoulos, Dan Li, Christophe Van Gysel, Zoltán Szlávik, Elena Simperl, Evangelos Kanoulas, and Lora Aroyo. 2018. Studying topical relevance with evidence-based crowdsourcing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* 1253–1262.

[16] Ayush Jain, Akash Das Sarma, Aditya Parameswaran, and Jennifer Widom. 2017. Understanding workers, developing effective tasks, and enhancing marketplace dynamics: a study of a large crowdsourcing marketplace. *Proceedings of the VLDB Endowment* 10, 7 (2017), 829–840.

[17] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. 2013. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval* 16, 2 (2013), 138–178.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Matthew Lease. 2011. On Quality Control and Machine Learning in Crowdsourcing. In *Proceedings of the 11th AAAI Conference on Human Computation (AAAIWS'11-11).* AAAI Press, 97–102. http://dl.acm.org/citation.cfm?id=2908698.2908717

[20] Dan Li and Evangelos Kanoulas. 2020. When to Stop Reviewing in Technology-Assisted Reviews: Sampling from an Adaptive Distribution to Estimate Residual Relevant Documents. *ACM Trans. Inf. Syst.* 38, 4, Article 41 (Sept. 2020), 36 pages. https://doi.org/10.1145/3411755

[21] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. 2016. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2296–2319.

[22] Yuan Li. 2019. *Probabilistic models for aggregating crowdsourced annotations.* Ph.D. Dissertation.

[23] Yuan Li, Benjamin Rubinstein, and Trevor Cohn. 2019. Exploiting Worker Correlation for Label Aggregation in Crowdsourcing. In *International Conference on Machine Learning.* 3886–3895.

[24] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. 2007. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, Vol. 310. ACM Amsterdam, The Netherlands.

[25] David E Losada, Javier Parapar, and Álvaro Barreiro. 2016. Feeling lucky? Multi-armed bandits for ordering judgements in pooling-based evaluation. In *proceedings of the 31st annual ACM symposium on applied computing.* 1027–1034.

[26] Eddy Maddalena, Marco Basaldella, Dario De Nart, Dante Degl'Innocenti, Stefano Mizzaro, and Gianluca Demartini. 2016. Crowdsourcing relevance assessments: The unexpected benefits of limiting the time to judge. In *Fourth AAAI conference on human computation and crowdsourcing.*

[27] Eddy Maddalena, Kevin Roitero, Gianluca Demartini, and Stefano Mizzaro. 2017. Considering assessor agreement in IR evaluation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval.* 75–82.

[28] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval.* Cambridge university press.

[29] De G Matthews, G Alexander, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. 2017. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research* 18, 1 (2017), 1299–1304.

[30] Tyler McDonnell, Matthew Lease, Mucahid Kutlu, and Tamer Elsayed. 2016. Why is that relevant? Collecting annotator rationales for relevance judgments. In *Fourth AAAI Conference on Human Computation and Crowdsourcing.*

[31] Pablo Morales-Álvarez, Pablo Ruiz, Raúl Santos-Rodríguez, Rafael Molina, and Aggelos K Katsaggelos. 2019. Scalable and efficient learning from crowds with Gaussian processes. *Information Fusion* 52 (2019), 110–127.

[32] Nasser M Nasrabadi. 2007. Pattern recognition and machine learning. *Journal of electronic imaging* 16, 4 (2007), 049901.

[33] Radford M Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models.* Springer, 355–368.

[34] Md Mustafizur Rahman, Mucahid Kutlu, and Matthew Lease. 2019. Constructing Test Collections using Multi-armed Bandits and Active Learning. In *The World Wide Web Conference.* 3158–3164.

[35] Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning.* Springer, 63–71.

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

[37] Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2014. Gaussian process classification and active learning with multiple annotators. In *International Conference on Machine Learning*. 433–441.

[38] Pablo Ruiz, Pablo Morales-Álvarez, Rafael Molina, and Aggelos K Katsaggelos. 2019. Learning from crowds with variational Gaussian processes. *Pattern Recognition* 88 (2019), 298–311.

[39] Mark D. Smucker, Gabriella Kazai, and Matthew Lease. 2012. Overview of the TREC 2012 Crowdsourcing Track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012 (NIST Special Publication, Vol. 500-298)*, Ellen M. Voorhees and Lori P. Buckland (Eds.). National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec21/papers/CROWD12.overview.pdf

[40] Mark D. Smucker, Gabriella Kazai, and Matthew Lease. 2012. Overview of the TREC 2012 Crowdsourcing Track. In *Proceedings of The Twenty-First Text REtrieval Conference, TREC 2012, Gaithersburg, Maryland, USA, November 6-9, 2012 (NIST Special Publication, Vol. 500-298)*, Ellen M. Voorhees and Lori P. Buckland (Eds.). National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec21/papers/CROWD12.overview.pdf

[41] Mark D. Smucker, Gabriella Kazai, and Matthew Lease. 2013. Overview of the TREC 2013 Crowdsourcing Track. In *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013 (NIST Special Publication, Vol. 500-302)*, Ellen M. Voorhees (Ed.). National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec22/papers/CROWD.OVERVIEW.pdf

[42] Ellen M Voorhees, Donna K Harman, et al. 2005. *TREC: Experiment and evaluation in information retrieval*. Vol. 63. MIT press Cambridge.

[43] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*. 2035–2043.

[44] Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, Vol. 51. ACM, 268–276.

[45] Xueying Zhan, Yaowei Wang, Yanghui Rao, and Qing Li. 2019. Learning from Multi-annotator Data: A Noise-aware Classification Framework. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 26.

[46] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment* 10, 5 (2017), 541–552.