ISSN 0361-7688, Programming and Computer Software, 2018, Vol. 44, No. 6, pp. 476–491. © Pleiades Publishing, Ltd., 2018.
Original Russian Text © R.A. Gilyazev, D.Yu. Turdakov, 2018, published in Trudy Instituta Sistemnogo Programmirovaniya, 2018, Vol. 30, No. 2.

Active Learning and Crowdsourcing: A Survey of Optimization Methods for Data Labeling

R. A. Gilyazev[a,b,*] and D. Yu. Turdakov[a,c,d,]**

[a] *Ivannikov Institute for System Programming, Russian Academy of Sciences, ul. Solzhenitsyna 25, Moscow, 109004 Russia*

[b] *Moscow Institute of Physics and Technology, Institutskii per. 9, Dolgoprudnyi, Moscow oblast, 141701 Russia*

[c] *Moscow State University, Moscow, 119991 Russia*

[d] *National Research University Higher School of Economics, ul. Myasnitskaya 20, Moscow, 101000 Russia*

**e-mail: gilyazev@ispras.ru*

***e-mail: turdakov@ispras.ru*

Received October 20, 2018

Abstract—High-quality annotated collections are a key element in constructing systems that use machine learning. In most cases, these collections are created through manual labeling, which is expensive and tedious for annotators. To optimize data labeling, a number of methods using active learning and crowdsourcing were proposed. This paper provides a survey of currently available approaches, discusses their combined use, and describes existing software systems designed to facilitate the data labeling process.

DOI: 10.1134/S0361768818060142

1. INTRODUCTION

Increasing the size of training samples makes it possible to improve accuracy and completeness for the majority of applied solutions based on supervised machine learning. However, to obtain training samples, in many cases, manual data labeling has to be used, which increases the cost of these solutions, especially if it is required to involve domain experts. To optimize data labeling, a number of approaches were proposed.

Active learning is one of the ways to facilitate the data labeling process. Active learning methods are aimed at finding the most informative samples for the classifier. On each iteration, the algorithm selects one sample from the unlabeled set; then, it is handed over to the oracle (expert) for labeling, and the classifier is re-trained on the updated set of training samples. A detailed survey of the corresponding methods was presented in [37]. Active learning is an important technique that provides a significant reduction in the amount of labeled data, which was confirmed both theoretically and experimentally.

Active learning assumes that, at any instant of time, labeling is carried out only by one expert (oracle). However, it is often required to parallelize the load, thus allowing the expert to label more samples. In this case, crowdsourcing platforms, e.g., Amazon Mechanical Turk (MTurk),[1] CrowdFlower,[2] and Yandex.Toloka,[3] are employed. Their purpose is to connect employers and employees. Any user of the platform can publish a task, e.g., a set of several samples to be labeled. Another user (annotator), having found an interesting task, performs it for a certain fee, which is much lower than the cost of involving an expert.

The main disadvantage of this approach is the low quality of the tasks performed. In [18], Kilgarriff singled out three main causes of low-quality labeling: data ambiguity, poor guidelines for annotators, and lack of motivation or knowledge.

Nevertheless, various investigations show that crowdsourcing can be useful, and the goal is usually achieved by aggregating the answers given by several annotators on the same sample. For instance, Nowak et al. [31] considered the problem of multi-label image annotation by experts and ordinary annotators from the MTurk crowdsourcing platform. Having measured several agreement statistics, the authors concluded that, with high-quality guidelines for annotators, several labels for an object are redundant if annotators are experts. In the opposite case, despite the good accuracy of annotators (kappa statistic was higher than

[1] https://www.mturk.com/
[2] https://www.crowdflower.com/
[3] https://toloka.yandex.ru/

Table 1. Classification of label inference algorithms

Annotator modeling	Level of competence	[4, 44]
	Error matrix	[3, 19, 34, 53]
	Parameters of the classifier	[14] [12]
	Dependence structure	[19] [42]
	Vector of competence (depending on the topic of a task)	[5] [51]
	Implicit modeling	[9] [15]
Sample modeling	Feature vector	[14] [34] [42] [45]
	Complexity	[17] [44]
	Distribution of topics	[5] [51]
Label inference	Majority voting	[4] [11] [16] [40]
	Likelihood maximization using the EM algorithm	[3] [34] [43] [44] [53]
	Iterative process	[4] [15] [51]
	Inference in the graphical model	[19] [25] [42]
	Solving the optimization problem	[14] [41]
Parameter initialization	Based on the results of labeling test samples	[4] [11] [16] [17] [40] [51]
	Majority voting	[3] [47] [34]
	Random initialization	[15] [19] [34] [42]
Task allocation	Random	[3] [14] [34] [44] [47]
	Iterative	[7] [11] [38] [43] [45] [54]
	Online	[5] [12] [20] [51] [53]
Quality testing	Quality of label inference	[3] [8] [9] [25] [51]
	Quality of the classifier trained on crowdsourced data	[12] [14] [34] [42]

appropriate threshold), several annotators performing one task yield a dataset of much higher quality.

Snow et al. [40] considered a number of text classification tasks. The data were labeled by experts and annotators from MTurk. Having measured the Pearson correlation between them, the authors concluded that non-expert labels are significantly inferior to expert ones. However, taking them into account makes it possible to improve the final result, e.g., in some problems, a simple averaging of non-expert labels proved to be competitive with the result of one expert already for four annotators.

Similar results were obtained by other researchers [2, 20] who answered the following question: what is more important when generating a training set for the classifier, coverage or high-quality labeling? In other words, there are two options: spend all resources to label as much data as possible one time or label a small number of samples, each with several annotators. The best strategy is to prefer high-quality labeling with weak annotators and coverage with strong annotators.

Crowdsourcing is widely used to solve problems that are not amenable to automatic calculations and require human effort. There are three factors that stand in the way of getting the maximum benefit from crowdsourcing platforms. The first one is quality, i.e., we need algorithms that most accurately distinguish between true labels and the other ones. It is also necessary to take into account the cost of labeling: it is not always reasonable to solve a problem by increasing the number of annotators per sample; this is the second factor. The third factor is that, sometimes, the speed of labeling is of primary importance; in this case, it is required to minimize the latency of task execution.

Most works are aimed at investigating the first two factors, and currently available solutions often take them both into account. There can be different classifications of related works. A possible classification is presented in Table 1. In crowdsourcing, label inference algorithms are based on the annotator model. Most often, it involves one or more parameters that characterize the reliability or competence of the annotator on each class. In addition, the relationship among annotators is sometimes modeled, e.g., by a Markov network. Less often, an individual classifier is trained for each annotator. Some algorithms do not have an explicit annotator model, but the difference between annotators is considered important. In other cases, it is assumed that annotators are equivalent.

Related works can be grouped based on the method of sample description. Many inference algorithms are not directly related to machine learning and are meant only for high-quality labeling, so the feature-based description of objects is rarely used. However, the characteristics of the samples should be taken into account. That is why the complexity of samples (the

higher it is, the less correct answers should be expected) and the distribution of topics (a vector that indicates to which predefined topics a task belongs) are often modeled.

Works can also be grouped based on the label inference algorithm employed. The simplest one is majority voting and its modifications. Other algorithms take into account the annotator model and use more complex inference procedures.

An important step is the initialization of model parameters, e.g., competences of annotators. This significantly affects the accuracy of the solution. To maximize the impact of resources and active learning, optimal task allocation is also important.

Finally, there are two ways to check the quality of algorithms: estimate the quality of label inference and estimate the quality of the classifier trained on labeled data.

The labeling speed problem is not very popular among researchers. In [10], three types of time delays were singled out. The first one is associated with the time interval between the publication of a task and the beginning of its execution. It includes the time to find an annotator, as well as the time it takes for the annotator to study the corresponding guideline and, possibly, perform some training tasks. The second type is associated with the execution of the task. The third type (which is discussed below in connection with online task allocation) is represented by the delays associated with the operation of the task generation algorithm (e.g., duration of an active learning iteration).

Active learning and crowdsourcing (here, by crowdsourcing we mean any parallelization of the labeling process) seem to be the only methods to optimize data labeling. Thus, a question naturally arises: how to combine them in order to achieve better results? In particular, how to improve task allocation among annotators?

In this work, we overview some data labeling methods that use crowdsourcing, as well as analyze the ways of applying active learning in the case of multiple annotators working in parallel. In addition, we investigate the frameworks that implement these methods.

The paper is organized as follows. The next section discusses some related surveys. Section 3 addresses the problem of quality assurance in crowdsourcing. Section 4 describes task allocation methods and online interaction between the users and the system. Section 5 is devoted to currently available crowd computing frameworks.

2. RELATED SURVEYS

As noted by A.V. Ponomarev in [55], the problem of discovering true labels arose relatively recently, simultaneously with the first crowdsourcing platforms. In that paper, some methods for quality assurance in crowd computing were discussed (there, crowd computing was interpreted more broadly than just data labeling for classification problems, in fact, it meant any information processing). The author conducted an analytical investigation of the problem and singled out a large number of methods for its solution: consensus methods, workflow design methods, centralized task allocation methods, game-theoretic methods, methods that take into account properties of tasks, and methods that analyze user actions, as well as influences the user is exposed to.

A detailed description of crowdsourcing methods was presented in [48]. Globally, the authors distinguished between works of two types: those that do not use features of objects and those in which inference is associated, in one way or another, with the machine learning model (including active learning). However, the authors did not consider the problem of parallel labeling. In addition, these methods were classified according to basic premises used in inference.

In [23], various models used in crowdsourcing were classified. The following problems were addressed: task modeling, quality control, cost control, and latency control. Particular attention was paid to crowdsourced operators that cover almost all types of crowd computing. These are selection operators, sort operators, aggregation operators, matching operators, etc. For each operator, quality control techniques were described. In addition, an overview of currently available crowdsourcing platforms and frameworks that facilitate operation with them by using relational databases was presented.

A related problem (which can be considered even more general than discovering the truth in crowdsourced data) is extracting truth from multiple sources. These sources can be news media, websites, or other sources of information. The formal statement of the problem is as follows. Suppose that we have a set of sources S, a set of objects O, and statements v_o^s; thus, $o \in O$ is an object to which a statement refers and $s \in S$ is a source. For each object o, there is the truth v_o^*; each source is characterized by its reliability w_s. Thus, given the set of objects O and statements about them, it is required to discover truths for the objects while inferencing w_s.

In [24], methods for solving this problem were divided into three following classes.

• Iterative methods: since the processes of inferencing true labels and reliabilities are related, labels are often estimated before the weights of sources are (these steps are repeated until convergence).

• Optimization-based methods that generally solve the following problem:

$$\underset{v_o^*, w_s}{\operatorname{argmin}} \sum_{o \in O} \sum_{s \in S} w_s d(v_o^s, v_o^*),$$

where d is a certain distance function.

• Methods based on probabilistic graphical models.

All these methods are similar to methods for discovering true labels (some of them coincide); however, in these methods, special attention is paid to interdependence among sources. For instance, one source often cites another; false and true statements can be duplicated.

Many researchers describe the inference algorithm as an iterative process [23, 24]. First, model parameters are initialized; then, two steps—inference of labels and re-evaluation of parameters (see Algorithm 1)—are repeated until convergence.

There are several open implementations and comparisons of the corresponding methods. The systems BATC (2013) [13] and SQUARE (2013) [39] were among the first. The authors of the survey [48] implemented some newer methods in the CEKA system (2015). Among all related works, we can single out the work [52] that compared 17 algorithms while presenting a detailed description of the test data. It was found that the algorithms based on EM and its modifications generally yield good results.

Unfortunately, in all these surveys, methods that take into account features of samples (related to machine learning) were not compared due to the high cost of the corresponding real-world experiment and a small number of suitable datasets.

3. MAIN LABEL INFERENCE METHODS IN CROWDSOURCING

In this section, we consider label aggregation methods while assuming that the system does not affect the tasks allocation process and analysis is always (except in specific cases) carried out upon collecting all answers. Let us begin with the formal statement of the problem. Here, we follow the classical statement that occurs in many papers [48, 55].

Suppose that we have N objects $x_1,\dots,x_N$, each belonging to one of J the classes $\{1,\dots,J\}$. Suppose also that we have K annotators, each of which has classified some objects, i.e., we have a matrix of labels $y_i^j \in \{0,\dots,J\}$, where $i \in \{1,\dots,N\}$, $j \in \{1,\dots,K\}$, and class 0 means that the corresponding annotator has not provided any labels for the sample i. The problem is formulated as follows: given a set of labels $\{y_i^j\}_{j=1}^K$, predict a correct label y_i for each sample i (here, we assume that this label exists and is unique). In other words, it is required to minimize the empirical risk

$$R = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(y_i = \hat{y}_i),$$

where $\hat{y}_i$ is the prediction of the label and $\mathbb{1}(x)$ is the indicator function that is 1 for a true argument x and is 0 otherwise.

The obvious solution is to use majority voting: for each sample, a label is selected that occurs most frequently for this sample. Suppose that $2k+1$ annotators have labeled a certain sample i with each annotator providing a correct label with the same accuracy p. Then, the probability to obtain a correct answer with majority voting is

$$\sum_{j=1}^{k}\binom{2k+1}{j} p^{2k+1-i}(1-p)^i.$$

This formula is often used to estimate the number of labels required to achieve the desired accuracy [26].

In this connection, it is interesting to recall Condorcet's jury theorem, which states that, if the number of jurors (annotators) tends to infinity, then, for $p > 0.5$, the probability to choose a correct answer tends to 1, while for $p < 0.5$, it tends to 0.

As mentioned above, noise is inherent in crowdsourcing, and labels are provided by people with different levels of competence and experience. That is why this approach is not always good because each annotator equally contributes to the final result. It seems reasonable to modify this method by assigning a weight w_j to each annotator (as a degree of his or her reliability), which reflects the probability for the annotator to tag an arbitrary sample with a correct label, and then conduct weighted voting:

$$y_i = \operatorname*{argmax}_{c\in[1,\dots,J]}\left(\sum_{j=1}^{K} w_j \mathbb{1}(y_i^j = c)\right).$$

In [50], the authors analyzed a number of voting algorithms with known probabilities q_j for each annotator to give a correct answer and proved that Bayesian voting, where the probability of a class is proportional to its likelihood, is optimal:

$$\Pr(y_i = c) \propto \prod_{j=1}^{K} q_j^{\mathbb{1}(y_{ij}=c)}\left(\frac{1-q_j}{J-1}\right)^{\mathbb{1}(y_i^j\neq c)}.$$

Normalization also allows one to estimate probabilities of classes.

Thus, the basic problem is evaluating the characteristics of the annotator, e.g., q_j. For this purpose, test sets of samples with a priori known answers are often used [11, 16, 40]. In some cases, unreliable annotators are excluded from the system [21]. In the ZenCrowds system [4], annotators were offered to solve the entity linking problem. The annotator j was described by a single parameter q_j (the proportion of correct answers). This parameter was initially derived from the results of labeling a test set; if the test set was not available, then it was assumed that $q_j = 0.5$. Next, two-step iterations were carried out until convergence. For each sample, a label was selected by weighted voting among reliable annotators; for this purpose, a reliability threshold was

set. Then, the weights q_j were re-estimated while assuming that the previously selected labels were true ones. In the general form, this procedure is described by Algorithm 1.

Algorithm 1: Iterative inference

input: L is the matrix of answers and L_{ij} is the answer of the annotator j on the sample i

output: labels $\hat{\mathbf{y}} = [\hat{y}_1, ..., \hat{y}_N]$ and qualities of annotators $\mathbf{q} = [\hat{q}_1, ..., \hat{q}_K]$

1: Initialize $\mathbf{q}$
2: Until convergence
3: for $i := 1$ to N do:
4: Estimate $\hat{y}_i$ based on $\mathbf{q}$ and L
5: for $j := 1$ to K do:
6: Estimate q_j based on $\hat{\mathbf{y}}$ and L

However, this approach does not take into account that reliability can vary depending on data or true labels; moreover, involving experts to label test sets results in additional spending. That is why reliability weights are used only in complex models. Nevertheless, this is a key idea in inferencing true labels.

In [3], Dawid and Skene proposed a broader interpretation of reliability. For each annotator, they computed an error matrix, where $\pi_{qj}^{(k)}$ is the probability for the annotator k to provide the label j if the true label is q. In that case, the annotator was characterized only by its error matrix. Suppose that p_{ij} is the prior distribution of classes for the sample i and $n_{il}^{(k)}$ is the number of times that the annotator provides the label l for the sample i (it is assumed that the annotator can label the same sample several times). Then, the likelihood of the probabilistic model is written as

$$\prod_{i=1}^{N}\left(\sum_{j=1}^{J} p_{ij}\prod_{k=1}^{K}\prod_{l=1}^{J}(\pi_{jl}^{(k)})^{n_{il}^{(k)}}\right).$$

In this case, the following important assumptions are made:

1. labeling does not depend on the sample, but depends only on the true label;

2. annotators provide their labels independently of each other.

This problem can be solved using the EM algorithm to find the parameters that maximize the likelihood. Iterations are carried out as follows: for fixed $\pi_{qj}^{(k)}$, the probabilities of classes are estimated; then, for fixed p_{ij}, the error matrices that maximize the likelihood are found. This model was called the DS model (anagram of the first letters of the authors' names). The model has obvious disadvantages: the EM algorithm does not guarantee convergence to the optimal solution, the initial parameters $\pi_{qj}^{(k)}$ need to be accurately selected, and the sample x_i itself is not used anywhere in the model (it does not take features into account). And, of course, assumptions 1 and 2 are not always justified.

This idea was further developed by many authors and gave rise to a class of methods that take into account (in one way or another) the annotator model and often employ the EM algorithm.

In [49], an original approach based on the spectral method was used to initialize the error matrix. Annotators were divided into three groups and, for each group, their averaged answers were computed. Then, the method of moments was employed to estimate error matrices for each group as if there were only three annotators. Based on these estimates, the initial approximations for all annotators were found.

The algorithm proposed by Raykar et al. [34] was used to solve the problem with two classes. In that case, the annotator model had two parameters:

- sensitivity $\alpha^j = \Pr[y^j = 1 | y = 1]$, which is a probability that the annotator j correctly determines the positive class;

- specificity $\beta^j = \Pr[y^j = 0 | y = 0]$, which is a probability to correctly determine the negative class.

The original DS model does not use the sample x_i itself. In [34], this disadvantage was eliminated by introducing the logistic regression model

$$\Pr[y = 1 | \mathbf{x}, \mathbf{w}] = \sigma(\mathbf{w}^T\mathbf{x}),$$

where $\sigma(z) = \dfrac{1}{1+e^{-z}}$. In that case, the label depended not only on the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ but also on $\mathbf{x}$ and weight vector $\mathbf{w}$; thus, the likelihood was written as

$$\Pr[D|\theta] = \prod_{i=1}^{N} Pr[y_i^1, ..., y_i^K | \mathbf{x}_i, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{w}].$$

Then, this expression was transformed taking into account assumptions 1 and 2, as well as its logarithm:

$$\ln(\Pr[D|\theta]) = \sum_{i=1}^{n} y_i \ln(p_i)a_i + (1 - y_i)\ln(1 - p_i)b_i,$$

where

$$p_i = \sigma(\mathbf{w}^T\mathbf{x}_i),$$

$$a_i = \prod_{j=1}^{K}[\alpha_j]^{y_i^j}[1 - \alpha_j]^{1-y_i^j},$$

$$b_i = \prod_{j=1}^{K}[\beta_j]^{1-y_i^j}[1 - \beta_j]^{y_i^j}.$$

The maximum was found by the EM algorithm: at the E step, y_i were estimated; at the M step, α_i and β_i

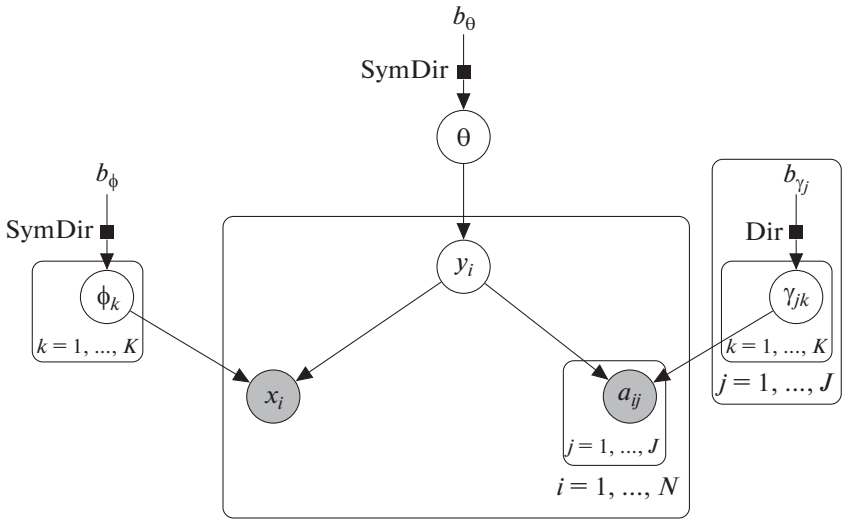

Fig. 1. Graphical model of label inference (shaded nodes correspond to observable variables). $\forall i : y_i | \boldsymbol{\theta} \sim Categorical(\boldsymbol{\theta})$, $\forall i : x_i | y_i, \boldsymbol{\psi} \sim Multinomial(|x_i|_1, \boldsymbol{\psi}_{\mathbf{y_i}})$, $\forall i, j : a_{ij} | y_i, \boldsymbol{\gamma}_{\mathbf{j}} \sim Multinomial(|a_{ij}|_1, \boldsymbol{\gamma}_{\mathbf{jy_i}})$.

were computed explicitly, while $\mathbf{w}$ was found by gradient descent. To initialize y_i, majority voting was used. In addition, it was noted that the algorithm works even if not all annotators have provided their labels for each sample and the method can be generalized to any probabilistic classifier. Experiments on several real-world problems showed that the method outperforms majority voting. However, comparison with other methods was not carried out. In [35], the method and its extensions, including application to the regression problem, were described in more detail.

A Bayesian extension of the method is of interest. The authors assumed that, if there are any prior preferences for annotators, then the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ belong to the Beta distribution. Then, instead of ML, the MAP estimate was used. Similar approaches occur in many papers. In the general case, probabilistic assumptions are made for all variables, and the system is described by a more complex graphical model inference in which can no longer use the EM algorithm. In this case, Markov chain Monte Carlo (MCMC) methods can be employed.

For instance, in [19], some additional parameters were introduced assuming that the error matrix and distribution of classes belong to the Dirichlet distribution.

$$\pi_j^{(k)} \sim \mathrm{Dir}(\alpha_{j,1}^{(k)}, ..., \alpha_{j,J}^{(k)}),$$

$$\mathbf{p} \sim \mathrm{Dir}(\nu_1, ..., \nu_J).$$

The parameters $\alpha_{j,l}^{(k)}$ were generated by the exponential distribution $\mathrm{Exp}(\lambda_{j,l}^{(k)})$. Thus, assuming the independence of annotators, the posterior distribution was written as

$$P(p, \pi, y, \alpha | c) = \prod_{i=1}^{N} \left(p_{y_i} \prod_{j=1}^{M} \pi_{y_i, c_i^{(k)}}^{(k)} \right) \times p(p|\nu) p(\pi|\alpha) p(\alpha|\lambda).$$

The inference was made iteratively based on the Gibbs scheme. In addition, a case of independent annotators modeled by a Markov network was considered. In turn, the graphical model described in [42] used features of objects. The authors considered a finite set of factors, linear weights that corresponded to a feature vector. The annotator was characterized by a binary sum of these factors, while the probability for the annotator to provide a positive class was described by probit regression. In [8], some parameters were introduced to model the distribution of samples for the text classification problem. This model is schematically shown in Fig. 1.

An interesting approach was proposed by Karger et al. [15]. The problem with two classes (1 and -1) was considered based on assumptions 1 and 2. To describe the distribution of tasks among annotators, a random (l, r) regular bipartite graph $G(\{t_i\}_{i=1}^m \cup \{w_j\}_{j=1}^n, E)$ was constructed, where l is the number of questions for each sample and r is the number of samples labeled by each annotator; the number of annotators n was found from $lm = rn$.

The inference was made iteratively. Messages of two types—$x_{i \to j}$ and $y_{j \to i}$—were used, where $(i, j) \in E$; $y_{j \to i}^{(0)}$ were initialized randomly from the normal distribution $\mathcal{N}(1,1)$. Then, for a specified $k_{\max}$ and labels $\{L_{ij}\}_{(i,j) \in E}$, the following $k_{\max}$ steps were performed:

– for all $(i, j) \in E$,

$$x^k_{i\to j} \leftarrow \sum_{j'\in\delta(i)\setminus j} L_{ij'} y^{(k-1)}_{j'\to i};$$

– for all $(i, j) \in E$,

$$y^k_{j\to i} \leftarrow \sum_{i'\in\delta(j)\setminus i} L_{i'j} x^{(k)}_{i'\to j},$$

where $\delta(u)$ denotes neighbors of the vertex u.

The resulting labels were found as

$$x_i = \sum_{j\in\delta(i)} L_{ij} y^{(k_{\max}-1)}_{j\to i}.$$

The parameters $x_{i\to j}$ and $y_{j\to i}$ can be easily interpreted as follows: $x_{i\to j}$ is a label for the sample i that is selected based on the votes of all annotators, except the jth one, while $y_{j\to i}$ is the reliability of the annotator j that is determined without taking into account his or her prediction for the ith sample. The authors provided theoretical justification for the correctness of the method and its asymptotic convergence. Comparisons with majority voting and DS model showed the effectiveness of the method on synthetic data. However, in [25], it was noted that the method is difficult to generalize to different annotator models and its applicability to real data is unclear. In turn, a more general approach based on a trust propagation algorithm was proposed.

Karger stated that his method is similar to the algorithm for finding the eigenvector of the matrix LL^T, with the only difference that one of the terms was omitted.

Ghosh et al. [9] proposed a method based entirely on the computation of the eigenvector for LL^T. The motivation was as follows. Suppose that we have two classes 1 and –1, the annotator i provides correct labels with the probability q_j, A is a matrix of labels, and y is a column of true labels. Then, it is easy to verify that $E(L) = \mathbf{y}(2\mathbf{q} - 1)^T$ and $E(LL^T) = \kappa\mathbf{y}\mathbf{y}^T + (n - \kappa)I$, where $\kappa = \sum_j (2q_j - 1)$ and I is the identity matrix. The maximum eigenvalue of the matrix $E(LL^T)$ is $\kappa\|\mathbf{y}\|^2 + (n - \kappa)$, while $\mathbf{y}$ is the eigenvector. Thus, as a vector of predictions, the algorithm returns the eigenvector of LL^T that corresponds to its maximum eigenvalue.

It should be noted that not all approaches take into account the feature vector of a sample: it is difficult to combine a label inference model and a machine learning algorithm. However, there are other ways to distinguish objects. For instance, the accuracy of label aggregation can be improved by making assumptions about the complexity of samples. In [44], a problem of image classification into two groups was solved. The parameter $1/\beta_i \in [0, +\infty)$, where $1/\beta_i = +\infty$, indicated that the image is so complex that even the expert labels it correctly with the probability 1/2, while the parameter $1/\beta_i = 0$ indicated that any annotator can correctly classify the image. The parameter $\alpha_j \in (-\infty, +\infty)$ characterized the reliability of an annotator, where $\alpha = +\infty$ corresponded to the ideal annotator, $\alpha = -\infty$ corresponded to the annotator that always provides the wrong class, and 0 corresponded to random choice. The probability that the label l_{ij} provided by the annotator j for the sample i is true was defined as a sigmoid $\sigma(\alpha_j\beta_i)$:

$$\Pr[l_{ij} = y_i \,|\, \alpha_j, \beta_i] = \frac{1}{1 + e^{-\alpha_j\beta_i}}.$$

Unknown parameters were found by the EM algorithm. The authors called their system GLAD.

In a similar way, the sample complexity parameter was used in the ELICE system [17].

For initial parameter estimation, reliable labels for n objects of the corpus were used. Based on these labels, the authors estimated the reliability of annotators as a difference between the numbers of correctly and incorrectly labeled samples

$$\alpha_j = \frac{1}{n}\sum_{i=1}^{n} [\mathbb{1}(y_i = y_i^j) - \mathbb{1}(y_i \neq y_i^j)]$$

and the complexity of samples

$$\beta_i = \frac{1}{M}\sum_{j=1}^{M} [\mathbb{1}(y_i = y_i^j)].$$

For unlabeled objects, labels were initially estimated by majority voting with weights α_j from the first step. Then, the resulting labels were used to estimate the parameter β_i; the final label was derived as follows:

$$F_i = \operatorname{sign}\left[\frac{1}{M}\sum_{j=1}^{M} \sigma(\alpha_j\beta_i) y_i^j\right].$$

Experiments on simulating annotators of different types showed that the algorithm is more robust to a large number of noisy labels and is effective when only 20% of the annotators provide high-quality labels. In that case, experts needed to label only 20 objects. The approach is applicable only to a two-class problem.

In [16], a modification of the method (ELICE-2) that exploits malicious annotators who deliberately provide incorrect labels was proposed. The algorithm was similar to the previous version, with the only difference that the parameters α_j and β_i were multiplied by $(1 - E(p))$, where $E(p) = -p\log(p) - (1 - p)\log(1 - p)$ is the entropy of the proportion of true labels for the annotator or sample, respectively. Obviously, random annotators have high entropy, while good and malicious annotators have low entropy; as in ELICE, the reliabilities of the latter had the opposite signs. The

final formula took into account the wrong labels provided by malicious annotators:

$$F_i = \operatorname{sign}\left[\frac{1}{M}\sum_{j=1}^{M}\sigma(|c\alpha_j\beta_i|) * l_{ij} * \operatorname{sign}(\alpha_j\beta_i)\right].$$

In turn, the DOCS [51] and CDAS [26] systems implemented the idea that the competence of a annotator depends on the topic of a task. In DOCS, for each task (sample), a vector of domains is introduced, which represents a certain distribution over selected topics, and each annotator is described by a set of numbers that characterize his or her competence in each domain. The inference is made iteratively, taking these domains into account. In CDAS, a task similarity graph is constructed: if the annotator copes well with a certain task, then he or she will probably cope with a similar task. Below, the ideas implemented in DOCS and CDAS are described in more detail.

There are several approaches that synthesize classifiers explicitly, without label inference. Sheng et al. [38] proposed to take into account all collected tags. If there are L_i labels for the sample i, then we can derive L_i training samples from it, each with its own label. Then, the sample is assigned a weight $1/L_i$ that is processed by the classifier.

Kajino [14] proposed an algorithm that generalizes logistic regression to the case of multiple annotators. For simplicity, two classes were considered. The general model was described as $\sigma(\mathbf{w}_0^T\mathbf{x})$. For each annotator j, its own model $\sigma(\mathbf{w}_j^T\mathbf{x})$ was constructed, where $\mathbf{w}_j$ characterized its deviation from the general model; this was expressed in terms of the following prior distribution:

$$\Pr[\mathbf{w}_0|\eta] \sim \mathcal{N}(0, \eta^{-1}I),$$

$$\Pr[\mathbf{w}_j|\mathbf{w}_0, \lambda] \sim \mathcal{N}(\mathbf{w}_0, \lambda^{-1}I).$$

Then, the logarithm of the posterior distribution was maximized, which is equivalent to the error function

$$-\sum_{j=1}^{J}\sum_{i\in I_j} l(y_i^j, \sigma(\mathbf{w}_j^T\mathbf{x}_i)) + \lambda/2\sum_{j=1}^{J}\|\mathbf{w}_j - \mathbf{w}_0\|^2 + \eta/2\|\mathbf{w}_0\|^2,$$

where $l(y, p) = -y\log(p) - (1-y)\log(1-p)$ is cross entropy.

Optimization was carried out as follows. Given $\mathbf{w}_j$, $\mathbf{w}_0$ was found analytically. Optimizations with respect to $\mathbf{w}_j$ were independent and were conducted individually, i.e., for each $\mathbf{w}_j$, an optimization step was made with subsequent computation of $\mathbf{w}_0$.

Now, let us consider the practical side of the problem. The comparisons [13, 48, 52] of the most popular methods show the absence of a clear leader: DS [3] and its modifications (RY [34]) yield good results for two classes, while complex models with a large number of parameters are not always applicable. Sometimes, very simple methods prove to be effective. For example, the clustering-based method [47] outperformed other approaches in multi-class problems [48]. In that case, each sample was described by a vector of the length $|J|$ (the number of classes). Each vector component was represented by the number of labels of the corresponding class. The solution was found by clustering these vectors into $|J|$ classes with the use of the k-means algorithm; the centers of clusters initialized the samples with the maximum number of votes for a given class.

4. TASK ALLOCATION

As mentioned above, almost all methods described in the previous section assume the simplest tasks allocation scenario. A set of objects to be labeled is selected in advance. Then, tasks with several samples in each are sent (each several times) to the crowdsourcing platform. Once all tasks are completed, the results are processed by a certain algorithm, i.e., labeling is carried out in offline mode.

In this section, we consider different scenarios of task allocation in which labeling is carried out iteratively, i.e., the analysis is conducted after each labeling iteration, rather than upon annotating all objects. This approach allows one to effectively use available resources, in particular, to implement the active selection of samples and annotators.

Traditional active learning assumes that samples are offered for labeling one by one. In this case, effectiveness is supported by theoretical justifications. In practice, however, this takes a long time: the classifier needs to be retrained after each iteration, and it is unclear how to process labels coming from the crowdsourcing platform (sending tasks with one sample is not cost effective). That is why a batch of several tasks, rather than one sample, is often generated. The effectiveness of this approach was confirmed experimentally. For instance, in [17], batches of 10–40 tasks were considered acceptable.

It was also confirmed by our experiments on several text classification problems. To simulate active learning, instead of annotators, we used datasets with a priori known true labels. As an active learning algorithm, we employed one of the simplest and most popular methods: the training set was supplemented with the samples on which the probabilistic classifier was least confident, namely, the samples with the minimum probability difference between two most popular predicted classes. As the classifier, we used logistic regression, while the features were represented by a

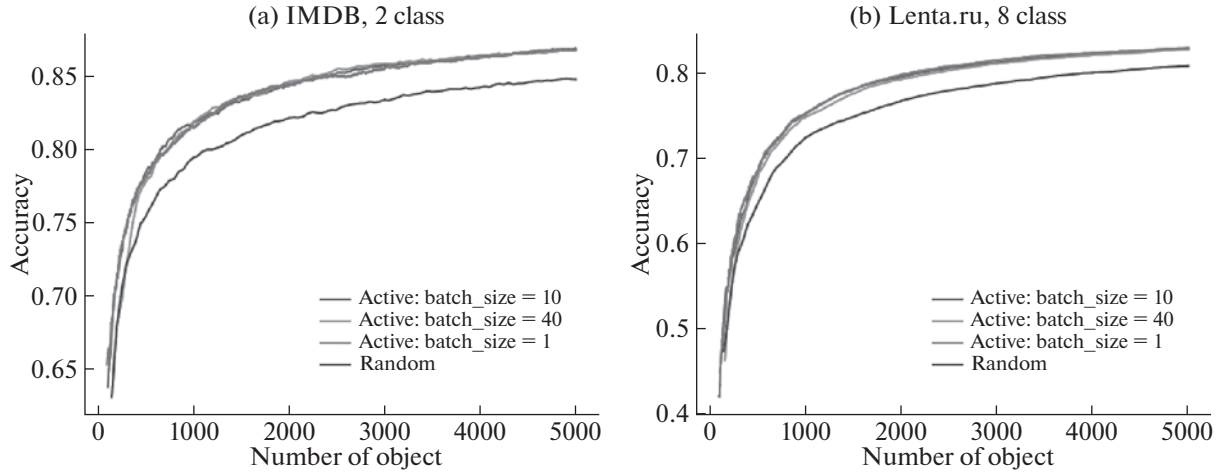

Fig. 2. Batch size versus accuracy: (a) sentiment analysis of IMDB comments and (b) news classification into eight classes.

bag of words. For experiments, we took the following datasets:

• IMDB dataset of 25000 samples (sentiment analysis of comments with two classes);

• set of 500000 samples (classification of news from lenta.ru[4] into eight classes).

In both cases, the classes were balanced. Figure 2 shows the experimental results. More specifically, it demonstrates the change in accuracy depending on the number of samples in the training set for different modes of operation: active learning with different batch sizes and random selection. The results were averaged over several (three) runs. The graphs show that, as compared to random labeling, active learning improves the accuracy; for the batch size of 10, the result is comparable to the classical approach, where the batch size is 1.

Existing approaches to batched task allocation can be divided into two groups. The first group uses iterative scheduling: having processed yet another batch of samples, the scheduler selects suitable objects or an annotator and sends the task to the crowdsourcing platform. In the second group, the annotator informs that he or she is ready to perform a task, and the system sends him or her samples online. In the case of a single annotator, both these approaches correspond to traditional active learning. However, of interest is parallel work of multiple annotators.

4.1. Iterative Labeling

The simplest approach is to vary the number of labels for each sample depending on its complexity. In [20], the number of annotators per sample was determined dynamically: once a sample was tagged several times, agreement among annotators on this sample was computed, and labeling was carried out until consensus was reached or the permissible number of iterations was exceeded. In [29], this number for each sample was computed in advance: all objects were clustered, cluster exemplars were labeled in test mode, and the resulting number was determined for each cluster based on the consensus on the samples selected.

Use of active learning, in one way or another, seems to be more promising. In one of the first related works [38], samples to be labeled were selected actively, and all collected labels were processed by the classifier. For an iterative selection of samples, several heuristics were proposed. Some of them computed uncertainty based on the labels available for the sample (in a simple case, it can be entropy of labels). Other heuristics employed certain active learning algorithms. These heuristics were combined by finding the geometric mean of the uncertainties computed in both the methods.

Most of the works considered in this section are aimed at selecting annotators and samples, either simultaneously or the sample is selected before the annotator. In other words, a new task is assigned to a particular person who will probably cope with it better than the others.

One of these works is [45]. The label provided by the annotator t for the sample x_i was modeled by the normal distribution centered at a true class:

$$p(y_i^{(t)}; x_i, z_i) = \mathcal{N}(y_i^{(t)}; y_i, \sigma_t(x_i));$$

where y_i is the true label.

$$\sigma_t(x_i) = \frac{1}{1 + \exp(-\mathbf{w}_t^T \mathbf{x}_i - \gamma_t)}.$$

In turn, the probability of the positive class was regarded as logistic regression

[4] https://lenta.ru/

$$p(z = 1|x_i) = \frac{1}{1 + \exp(-\boldsymbol{\alpha}^T \mathbf{x}_i - \beta)}.$$

The annotator's parameters $\mathbf{w}_i^T$ and γ_t, as well as regression weights $\boldsymbol{\alpha}$ and β, were determined by finding the maximum likelihood with the use of the EM algorithm.

First, the samples on which the current model was not confident were selected:

$$\operatorname{argmin}_x (0.5 - p(y|x))^2.$$

The solution was a hyperplane $\boldsymbol{\alpha}^T \mathbf{x}_i + \beta = 0$. Then, the annotator with the minimum variance $\sigma_t(x)$ was found. As a result, the authors arrived at the following optimization problem:

$$\min_{x,p}(C(\boldsymbol{\alpha}^T \mathbf{x} + \beta) + p^T[\mathbf{w}_1, ..., \mathbf{w}_T]x + p^T\gamma),$$

where $p = [p_1, ..., p_T]$ is the distribution of annotators, $\sum_{i=1}^T p_i = 1$, and $\gamma = [\gamma_1, ..., \gamma_T]$.

Thus, the sample x^* and the annotator for it were found simultaneously. Of course, the data might not contain x^*. In that case, the sample nearest to it, in terms of the Euclidean metric, was selected.

It is easy to deduce the general scheme of active learning algorithms with multiple annotators. First, the most informative sample from the unlabeled set is selected:

$$x_{i+1} = \operatorname{argmin}_{x \in X_u} A(x).$$

Then, for this sample, the best available annotator (the one with the maximum confidence) is selected:

$$t_{i+1} = \operatorname{argmax}_{t \in T} Q_t(x_{i+1}).$$

At the end of the cycle, all necessary parameters are re-evaluated.

Thus, it is required to choose the confidence function $A(x)$ for the classifier and the reliability function $Q_t(x)$ for annotators; it is also desirable that the classifier and the annotators be related. In [7], as A and Q, entropy of class predictions and linear combinations of features, respectively, were used. In [54], an SVM with radial basis functions was employed in both cases.

A disadvantage of these approaches is that the model tends to give preference to the same annotators. The dependence of the model on one annotator is undesirable: the predictions made by the algorithm are biased and, eventually, the model begins to accept all labels provided by this annotator as true ones. In addition, the other annotators lose the opportunity to approve themselves. Rodrigues et al. [36] used two parameters—sensitivity α_j and specificity β_j—to evaluate the annotator's performance. For labeling, the annotator with the highest expectation to give a correct answer was selected:

$$j^* = \operatorname{argmax}_j[\alpha_j p(y = 1|x^*, L, Y) + \beta_j(1 - p(y = 1|x^*, L, Y))],$$

where L is a set of labeled samples and Y is a set of answers given by annotators. As $p(y = 1|x, L, Y)$, a Gaussian classifier adapted to multiple annotators was employed [33]. The sample x^* was selected by active learning. Since α_j and β_j were computed based on estimated labels, it was proposed not to take into account the labels provided by the annotator when evaluating his or her performance. For instance, if the sensitivity is computed as

$$\alpha_j = \frac{\sum_{i=1}^N y_i^j p(y_i = 1|L, Y)}{\sum_{i=1}^N p(y_i = 1|L, Y)},$$

then the probabilities $p(y_i = 1|L, Y)$ are replaced by $p(y_i = 1|L \backslash L^j, Y \backslash Y^j)$. This solves the problem of dependence on one annotator.

Sometimes several annotators are selected simultaneously, or selection is carried out randomly (proportionally with their reliabilities). Some algorithms support expert estimates: if the agreement among annotators is poor, then the sample can be handed over to the expert for estimation [11, 30]. In [46], once a sample is selected, an expert or a common annotator is immediately assigned to label it.

Welinder and Perona [43] developed an online version of the EM algorithm. The annotator was described by the probability to give a correct answer on each class. Several types of annotators were introduced: experts (E), annotators who give poor-quality answers (B), and the others. Iterations were carried out as follows: a sample was assigned for labeling to an annotator from the set E (if it was empty, then the sample was assigned to any annotator, but not the one from B). Then, the posterior distribution of classes $p(y)$ for this sample was estimated, which corresponds to the E step. The procedure was repeated until $\max_y p(y) < \tau$, where τ is a threshold. In other words, labeling continued until a high-confidence label was found, or until the maximum number of iterations was reached. Then, the parameters of the annotators were re-estimated (M step) and the sets E and B were updated.

It should be noted that, to some extent, online selection of annotators is related to the problem of multi-armed bandits. In this case, however, response (reward) cannot be computed explicitly because it is difficult to determine the effectiveness of the annotator based on one answer. That is why various heuristics

have to be employed. For example, the model [27] always operates in either of the two modes: exploration (estimating the competence of annotators) or exploitation. Suppose that $E(t)$ is a set of samples labeled in the first mode before the instant t. If $|E(t)| < D_1 \log(t)$, or there is a sample $x_k \in E(t)$ labeled less than $D_2 \log(t)$ times, then, at the instant $t + 1$, all annotators are offered either a new sample or x_k, respectively. Then, weighted voting is used to update true labels and estimate the reliability of the annotators. This is the exploration mode. In the exploitation mode, a new sample is randomly selected and assigned to the most reliable annotators.

Algorithm 2: Active learning with annotator selection

L and U are the sets of labeled and unlabeled samples, respectively

A is the number of annotators

Iter is the number of iterations

```
1: for i:=1 to Iter do
       // Select a sample to be labeled from the set U
       taking into account the answers stored in L
2:     x = U.sample(L)
       // Choose an annotator to label x based
       on the answers stored in L
3:     j = A.choice(x, L)
       // Get a label for x
4:     y = get_answer(j, x)
5:     L.Update(x, y)
```

In [41], an approach was proposed to select freelancers with unknown ratings and different costs. In that case, response was estimated as the quality of the work done. The difference from the problem of multi-armed bandits was in limiting the number of tasks assigned to one annotator (due to a lack of time) and in that the same task could be assigned to several annotators (by analogy with pulling several arms).

The formal statement of the problem [41] was as follows: suppose that we have a budget B, annotators with the cost of one task c_i, the limit on the number of tasks l_i, and an unknown distribution of annotators' efficiencies. It was required to allocate the tasks in such a way as to maximize the sum of efficiencies under certain constraints on the budget. The problem was solved in two stages. At the first stage, ε was selected and εB of the budget was spent: annotators were arranged in the ascending order of c_i and tasks were assigned to them in a loop. All executed tasks were graded. For each annotator, the expectation of his or her efficiency $\hat{\mu}_i$ was estimated as the average of the grades. At the second stage, for the estimated efficiency $\hat{\mu}_i$, cost c_i, budget $(1 - \varepsilon)B$, and limit l_i, the knapsack problem (which is well known in the complexity theory) was solved by heuristic methods.

4.2. Online Labeling

Considering the organization of the labeling process, iterative scheduling seems to be ineffective: the system has to wait until a certain annotator completes his or her task, which does not allow free annotators to be engaged in labeling. This makes it difficult to parallelize the process. If labeling is carried out by a highly competent interested party (rather than annotators from crowdsourcing platforms), optimization of the process becomes crucial. In this case, it is desirable that scheduling be organized as follows. Suppose that each annotator can spend a few minutes a day on labeling. At a convenient time, he or she enters the system to perform tasks online: the system sends him or her one or more samples, processes the answers, and offers the next task with the minimum delay. This corresponds to the simplest case of labeling: it is sufficient to randomly select tasks, taking into account that one annotator is allowed to label no more than one sample. However, a question arises: are there any optimized online labeling methods if the goal is to synthesize a high-quality classifier for the problem? In particular, can we use active learning for this purpose?

Online active learning with multiple annotators can be organized as two queues. The first queue includes samples to be labeled and is supplemented with batches comprising several samples. The second queue stores the answers given by the annotators. Once a certain number of answers are processed, the next active learning iteration is initiated and the queue is updated. This asynchronous process reduces the time spent by the annotator waiting for the system to generate the next question. However, it is ineffective when the quality of labels is low and each sample needs to be labeled by several annotators before initiating the next iteration. This scheme was discussed in [10, 20]. When the user opened the task on the crowdsourcing platform, the system redirected him or her to the customer's server where the objects to be labeled were provided online.

In [12], for task allocation, it was proposed to predivide the entire unlabeled dataset D into subsets $D = D_1 \cup \ldots \cup D_K$, $D_i = U_i \cup \mathcal{L}_i$, where K is the number of annotators, while $\mathcal{L}$ and U_i are sets of labeled and unlabeled samples, respectively, with each sample being included into the same number of subsets, $|\{D_k \,|\, x_i \in D_k\}| = m$, $\forall x_i \in D$. For each subset D_i, an individual classifier $f_i(x)$ was trained. The error function $L(D)$ was used to optimize all classifiers $f_i(x)$ simultaneously:

$$L(D) = \sum_{i=1}^{K} \sum_{x_j \in \mathcal{L}_i} L_i(y_i^j, f_i(x_j))$$

$$+\sum_{1\le i\ne j\le K}\sum_{x_k\in D_i\cap\mathcal{L}_j} L_{ij}(y_k^j, f_i(x_k))$$

$$+\lambda\sum_{i=1}^{K}\Omega(\|f_i\|_H),$$

where L_i is the error function for the classifier i and L_{ij} takes into account the error made by the classifier i on the samples from the set $\mathcal{L}_i$ that were labeled by the annotator j; the last term is responsible for regularization. Hence, the classifiers were not independent.

Labeling was carried out by active learning. For the annotator i, the sample nearest to the boundary of the decision rule $f_i(x)$ was selected. Upon labeling, the parameters associated with the annotator i were optimized while assuming that the other parameters were fixed. In [12], as classifiers, support vector machines were used. The resulting classifier was synthesized by averaging the algorithms $f_i(x)$.

There are several systems that support online labeling of arbitrary data without using classifiers: DOCS [51], QASQA [53], and iCrowd [5]. For inference, the following iterative scheme with Bayesian voting is usually employed (in fact, it is an online version of Algorithm 1). Once yet another annotation of a sample is received, the expected label for this sample is updated; then, the parameters of annotators are re-evaluated taking into account the updated label (i.e., in contrast to Algorithm 1, the parameters are re-evaluated after each annotation rather than upon collecting them all). Now, let us consider task allocation. For all samples, the current posterior distribution of labels is permanently stored as a matrix, where $M_{i,k}$ is the probability that the sample i belongs to the class k. For each annotator, his or her expected answers are estimated, where $Q_{ia}^{(j)}$ is the probability for the annotator j to tag the sample i with the class a. As the prior distribution of classes, the matrix M is used. In the simplest case where the annotator is described only by the probability q_j to give a correct answer, $Q_{ia}^{(j)}$ are computed as follows:

$$Q_{ia}^{(j)} = q_j M_{i,a} + \frac{1-q_j}{J-1}(1-M_{i,a}).$$

For each answer a, the matrix M is updated:

$$M_{i,k}^{|a} \propto M_{i,k}(q_j)^{1(a=k)}\left(\frac{1-q_j}{J-1}\right)^{1(a\ne k)}.$$

As a result, the samples on which the change in the uncertainty metric is maximal are selected. For example, in DOCS, it is a difference between the current and expected entropies, $H(M_i) - H(M_i^|)$, where

$$H(M_i^|) = \sum_{a=1}^{J} H(M_i^{|a})Q_{ia}^{(j)}.$$

In the general form, this approach is described by Algorithm 3.

Algorithm 3: Inference in online labeling

input: the set of unlabeled samples U

output: labels $\hat{\mathbf{y}} = [\hat{y}_1,\ldots,\hat{y}_N]$ and qualities of annotators $\mathbf{q} = [\hat{q}_1,\ldots,\hat{q}_K]$

L is the matrix of answers and M is the class distribution matrix for all samples

1: Initialize **q**

2: Do the following:

// Get the id number of the annotator

3: j = get_requestor()

// Select a suitable sample based on q_j and prior distributions M

4: i = U.sample(q_j, M)

5: L.Update(i, j, get_answer(j, x_i))

// Update M based on the answers L and qualities **q**

6: M.Update(L, **q**)

// For each annotator j, estimate q_j based on M and L

7: for j:=1 to K do:

8: q_j.Update(L, M)

5. OVERVIEW OF FRAMEWORKS

5.1. Crowdsourcing Platforms

In this section, we describe some well-known crowd computing platforms. In these platforms, there are two types of users: customers who publish tasks and performers (or annotators). Tasks are generally represented as sets of several samples to be labeled; they are often referred to as human intelligence tasks (HITs). These can be various classification tasks, machine translation tasks, entity matching tasks, etc. Let us describe one of these platforms—Yandex.Toloka—in more detail. Before publishing a task, the customer needs to create a project and write a task execution guideline. Then, the corresponding HTML interface should be generated. Tasks can be added as a set of several HITs, which is called the pool. For each pool, the maximum execution time, as well as the overlap (the number of users to perform the task), is specified. Labels are aggregated by majority voting. For additional quality control, tasks with a priori known answers can be added (to the annotator, they seem like ordinary tasks). This enables the system to block the users who either often make mistakes on control questions or perform tasks suspiciously quickly. The customer can also add training tasks, which can be used as a qualification test. Upon assigning a task, the system displays its execution progress.

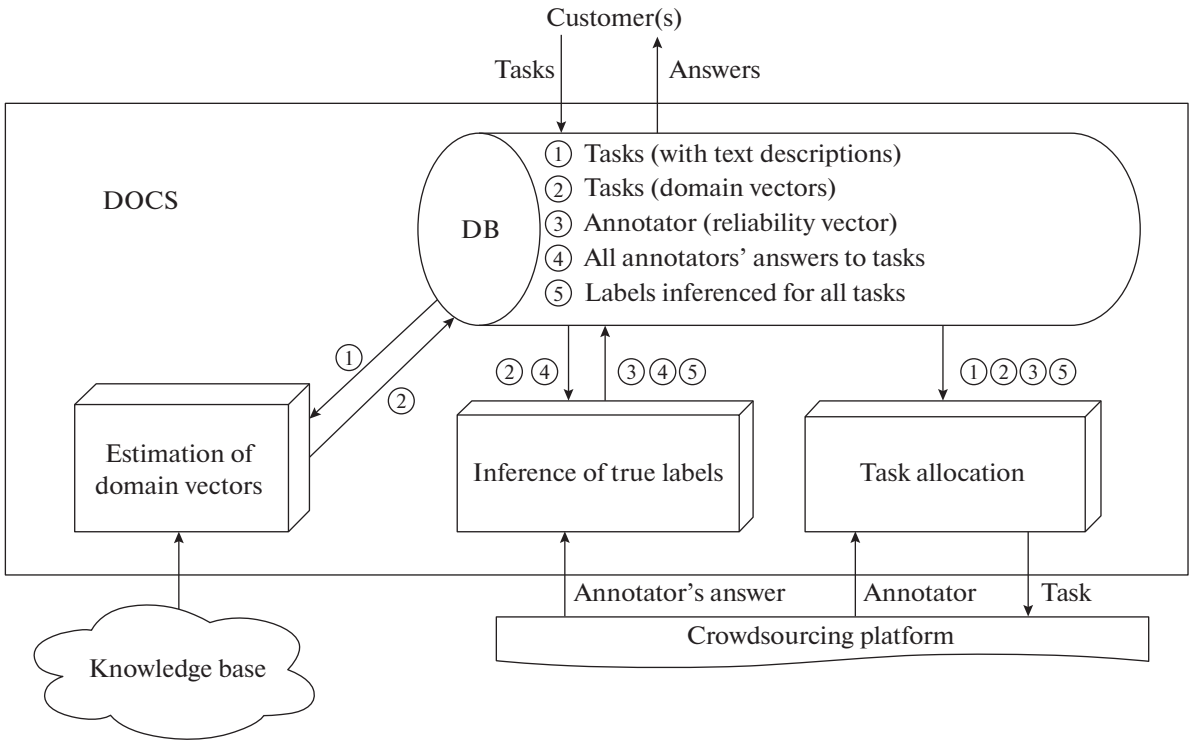

Fig. 3. Architecture of DOCS.

MTurk and CrowdFlower are the most popular foreign crowdsourcing platforms. They offer some default templates to describe tasks, as well as support an interface that allows the users to design their own templates by using CSS and Javascript. In addition to the standard mechanism used in Yandex.Toloka, these systems provide some extra tools. First, they support APIs that allow the user to add tasks, as well as get various statistics and answers to tasks, by using high-level programming languages. Second, when performing a task, the platform can redirect the annotator to the customer's website, which is especially useful in the case of online labeling.

5.2. Crowdsourced Optimizers

There are some systems designed to facilitate and optimize operation with crowdsourcing platforms: storing and processing information about annotators and annotations, generating tasks and sending them to the crowdsourcing platform, and improving the quality of label inference. These systems act as mediators between the platform and the customer. We have mentioned these systems in the previous sections. It is important that they can be used to label training sets for machine learning algorithms; their purpose is to yield annotations for a given set of samples. In this case, features of objects are rarely used.

In the simplest case, Askit [1], no additional parameters are introduced and the algorithm works only with the labels received. From a set of samples, the samples with the maximum measure of uncertainty are selected for relabeling. For each label, the entropy of all labels upon adding a new one is computed. Two methods to compute uncertainty—the maximum of these values and their average—were proposed. CDAS [26] models reliabilities of annotators and determines the number of questions required to achieve the desired quality.

For other approaches, the online labeling scheme described in the previous section is valid. In QASCA [53], the annotator is modeled by the probability to provide a correct class, while the probability to answer with any incorrect class is assumed to be constant. As a metric, instead of the entropy difference, the difference between the probabilities of the most confident classes in the distribution expected from the annotator and the current one is used.

In iCrowd [5] and DOCS [51], to estimate expected answers $Q_{ia}^{(j)}$, more complex methods are employed that implement the idea that the annotator's competence depends on the topic of a task. In iCrowd, a weighted graph of sample similarity is constructed.

In DOCS, objects and annotators are described by vectors of several numbers that correspond to certain domains. In this case, the object is a domain distribution vector, while the annotator is a vector of probabilities to give a correct answer on each domain. Thus, three problems arise. First, for available samples, it is

required to determine their domain vectors. The statement of this problem is similar to that of the thematic modeling problem [56]. However, instead of using standard methods, the authors of DOCS proposed their own approach. As an example, all entities were selected and, for each entity, distributions of concepts were found. These two steps used a ready-made wikifier. Each concept had a binary domain vector (belongs to/does not belong to). To compute a domain vector for a given set of concepts, it was required to add these binary vectors over all entities and then normalize them. In turn, to compute the vector for the whole sample, it was needed to find the expectation of concept distribution.

Second, upon receiving each annotation, reliabilities and labels should be re-estimated. For this purpose, standard iterative algorithm can be employed, taking domains into account. First, for each domain, the distribution of labels for all samples is estimated by Bayesian voting; the final distribution is derived by weighted summation of these distributions in proportion to the weights of the domains. Then, reliabilities of annotators on each domain are re-estimated. To speed up the process, only the parameters directly associated with a new annotation are re-evaluated; these parameters are the distribution of labels for the corresponding sample and the competences of the annotators who labeled this sample.

The third problem is selecting a task for the annotator. If the annotator has not performed any tasks, then he or she is offered a test set of samples. Otherwise, the labels expected from the annotator (depending on his or her competence in certain domains) are estimated, and the samples with the maximum entropy difference between the available labels and the expected ones are selected.

DOCS was compared with Askit, QASCA, and iCrowd on the MTurk platform, as well as with thematic modeling algorithms for domain identification. For each system, independent labeling processes were run. The results showed that DOCS outperforms its competitors by all measures.

5.3. Crowdsourced Database Management Systems

Crowdsourcing systems that work with relational databases, e.g., Qurk [28], Deco [32], CDB [22], and CrowdOp [6], are quite popular. These systems generally operate as follows. The user uploads data in the form of tables (possibly, with missing values) to the system and enters a standard SQL query. The system analyzes the query, generates a plan of simple questions for the crowdsourcing platform, and performs crowdsourced optimizations during execution of the plan. In CrowdDB (one of the first systems of this kind), questions of three types are used: filling in missing values, joining identical objects, and comparing objects based on a certain criterion.

Generating the optimal plan of questions is an important task of these systems. For instance, in Deco (2012) and CrowdOP (2015), optimization is based on trees, while CDB (2017) uses a weighted graph of objects.

6. CONCLUSIONS

The problem of crowdsourced data labeling has been intensively investigated in the last few years. During this time, many different methods and implementations have been proposed. Since the difference in the competences and motivations of annotators is a specific characteristic of crowdsourcing systems, a specific characteristic of the corresponding solutions is the estimation of the annotator's reliability. Offline labeling seems to be the most widely investigated problem, which is confirmed by a large number of surveys and comparisons published [23, 48, 52, 55]. For binary classification problems, the best algorithms so far are based on the error matrix [3] and its modifications. For multi-class problems, there are a few algorithms. Unfortunately, there is still a gap between data labeling and the generation of a training set for the classifier (feature-based description of samples is rarely used).

In turn, iterative labeling seems more promising. We believe that, in the future, the focus of research will be shifted in this direction, particularly, to active learning methods. Here, it is necessary to eliminate the effect of dependence on one annotator. Online active learning is free from this drawback and allows annotators to work in parallel. However, these methods are rare.

It should also be noted that current investigations are more focused on practical implementations, and crowdsourced optimizers (as well as their analogs) become increasingly popular. Unfortunately, these works are aimed mostly at accurate data labeling, rather than synthesizing classifiers for this purpose.

REFERENCES

1. Boim, R., Greenshpan, O., Milo, T., Novgorodov, S., Polyzotis, N., and Tan, W.-C., Asking the right questions in crowd data sourcing, *Proc. 28th Int. Conf. Data Engineering (ICDE),* 2012, pp. 1261–1264.
2. Brew, A., Greene, D., and Cunningham, P., Using crowdsourcing and active learning to track sentiment in online media, *Proc. 19th Eur. Conf. Artificial Intelligence (ECAI),* Amsterdam, 2010, pp. 145–150.
3. Dawid, A.P. and Skene, A.M., Maximum likelihood estimation of observer error-rates using the EM algorithm, *Appl. Stat.,* 1979, pp. 20–28.
4. Demartini, G., Difallah, D.E., and Cudré-Mauroux, P., Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking, *Proc. 21st Int. Conf. World Wide Web,* 2012, pp. 469–478.
5. Fan, J., Li, G., Ooi, B.C., Tan, K.-l., and Feng, J., iCrowd: An adaptive crowdsourcing framework, *Proc.*

ACM SIGMOD Int. Conf. Management of Data, 2015, pp. 1015–1030.

6. Fan, J., Zhang, M., Kok, S., Lu, M., and Ooi, B.C., Crowdop: Query optimization for declarative crowdsourcing systems, *IEEE Trans. Knowl. Data Eng.,* 2015, vol. 27, no. 8, pp. 2078–2092.
7. Fang, M., Zhu, X., Li, B., Ding, W., and Wu, X., Self-taught active learning from crowds, *Proc. 12th IEEE Int. Conf. Data Mining (ICDM),* 2012, pp. 858–863.
8. Felt, P., Haertel, R., Ringger, E.K., and Seppi, K.D., Momresp: A bayesian model for multi-annotator document labeling, *Proc. LREC,* 2014, pp. 3704–3711.
9. Ghosh, A., Kale, S., and McAfee, P., Who moderates the moderators?: Crowdsourcing abuse detection in user-generated content, *Proc. 12th ACM Conf. Electronic Commerce,* 2011, pp. 167–176.
10. Haas, D., Wang, J., Wu, E., and Franklin, M.J., Clamshell: Speeding up crowds for low-latency data labeling, *Proc. VLDB,* 2015, vol. 9, no. 4, pp. 372–383.
11. Hao, S., Hoi, S.C.H., Miao, C., and Zhao, P., Active crowdsourcing for annotation, *Proc. IEEE/WIC/ACM Int. Conf. Web Intelligence and Intelligent Agent Technology (WI-IAT),* Singapore, 2015, vol. 2, pp. 1–8.
12. Hua, G., Long, C., Yang, M., and Gao, Y., Collaborative active learning of a kernel machine ensemble for recognition, *Proc. IEEE Int. Conf. Computer Vision (ICCV),* 2013, pp. 1209–1216.
13. Hung, N.Q.V., Tam, N.T., Tran, L.N., and Aberer, K., An evaluation of aggregation techniques in crowdsourcing, *Proc. Int. Conf. Web Information Systems Engineering,* 2013, pp. 1–15.
14. Kajino, H., Tsuboi, Y., and Kashima, H., A convex formulation for learning from crowds, *Trans. Jpn. Soc. Artif. Intell.,* 2012, vol. 27, no. 3, pp. 133–142.
15. Karger, D.R., Oh, S., and Shah, D., Iterative learning for reliable crowdsourcing systems, *Advances in Neural Information Processing Systems,* 2011, pp. 1953–1961.
16. Khattak, F.K., *Toward a Robust and Universal Crowd Labeling Framework,* Columbia University, 2017.
17. Khattak, F.K. and Salleb-Aouissi, A., Quality control of crowd labeling through expert evaluation, *Proc. 2nd NIPS Workshop Computational Social Science and the Wisdom of Crowds,* vol. 2, 2011.
18. Kilgarriff, A., Gold standard datasets for evaluating word sense disambiguation programs, *Comput. Humanit.,* 1998, pp. 4–12.
19. Kim, H.-C. and Ghahramani, Z., Bayesian classifier combination, *Artif. Intell. Stat.,* 2012, 619–627.
20. Laws, F., Scheible, C., and Schütze, H., Active learning with Amazon Mechanical Turk, *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP),* Stroudsburg, 2011, pp. 1546–1556.
21. Lee, K., Caverlee, J., and Webb, S., The social honeypot project: Protecting online communities from spammers, *Proc. 19th Int. Conf. World Wide Web,* 2010, pp. 1139–1140.
22. Li, G., Chai, C., Fan, J., Weng, X., Li, J., Zheng, Y., Li, Y., Yu, X., Zhang, X., and Yuan, H., CDB: Optimizing queries with crowd-based selections and joins, *Proc. ACM Int. Conf. Management of Data,* 2017, pp. 1463–1478.
23. Li, G., Wang, J., Zheng, Y., and Franklin, M.J., Crowdsourced data management: A survey, *IEEE Trans. Knowl. Data Eng.,* 2016, vol. 28, no. 9, pp. 2296–2319.
24. Li, Y., Gao, J., Meng, C., Li, Q., Su, L., Zhao, B., Fan, W., and Han, J., A survey on truth discovery, *ACM SIGKDD Explor. Newsl.,* 2016, vol. 17, no. 2, pp. 1–16.
25. Liu, Q., Peng, J., and Ihler, A.T., Variational inference for crowdsourcing, *Advances in Neural Information Processing Systems,* 2012, pp. 692–700.
26. Liu, X., Lu, M., Ooi, B.C., Shen, Y., Wu, S., and Zhang, M., CDAS: A crowdsourcing data analytics system, *Proc. VLDB Endowment,* 2012, vol. 5, no. 10, pp. 1040–1051.
27. Liu, Y. and Liu, M., An online learning approach to improving the quality of crowd-sourcing, *ACM SIGMETRICS Perform. Eval. Rev.,* 2015, vol. 43, pp. 217–230.
28. Marcus, A., Wu, E., Karger, D.R., Madden, S., and Miller, R.C., Crowdsourced databases: Query processing with people, *Proc. CIDR,* 2011.
29. Mozafari, B., Sarkar, P., Franklin, M., Jordan, M., and Madden, S., Scaling up crowdsourcing to very large datasets: A case for active learning, *Proc. VLDB Endowment,* 2014, vol. 8, no. 2, pp. 125–136.
30. Nguyen, A.T., Wallace, B.C., and Lease, M., Combining crowd and expert labels using decision theoretic active learning, *Proc. 3rd AAAI Conf. Human Computation and Crowdsourcing,* 2015.
31. Nowak, S. and Rüger, S., How reliable are annotations via crowdsourcing: A study about inter-annotator agreement for multi-label image annotation, *Proc. Int. Conf. Multimedia Information Retrieval (MIR),* New York, 2010, pp. 557–566.
32. Parameswaran, A.G., Park, H., Garcia-Molina, H., Polyzotis, N., and Widom, J., Deco: Declarative crowdsourcing, *Proc. 21st ACM Int. Conf. Information and Knowledge Management,* 2012, pp. 1203–1212.
33. Rasmussen, C.E., Gaussian processes in machine learning, *Advanced Lectures on Machine Learning,* Springer, 2004, pp. 63–71.
34. Raykar, V.C., Yu, S., Zhao, L.H., Jerebko, A., Florin, C., Valadez, G.H., Bogoni, L., and Moy, L., Supervised learning from multiple experts: Whom to trust when everyone lies a bit, *Proc. 26th Annu. Int. Conf. Machine Learning,* 2009, pp. 889–896.
35. Raykar, V.C., Yu, S., Zhao, L.H., Valadez, G.H., Florin, C., Bogoni, L., and Moy, L., Learning from crowds, *J. Mach. Learn. Res.,* 2010, vol. 11, pp. 1297–1322.
36. Rodrigues, F., Pereira, F., and Ribeiro, B., Gaussian process classification and active learning with multiple annotators, *Proc. Int. Conf. Machine Learning,* 2014, pp. 433–441.
37. Settles, B., *Active learning literature survey,* University of Wisconsin–Madison, 2009.
38. Sheng, V.S., Provost, F.J., and Ipeirotis, P.G., Get another label? Improving data quality and data mining using multiple, noisy labelers, *KDD,* Li, Y., Liu, B., and Sarawagi, S., Eds., 2008, pp. 614–622.

39. Sheshadri, A. and Lease, M., Square: A benchmark for research on computing crowd consensus, *Proc. 1st AAAI Conf. Human Computation and Crowdsourcing,* 2013.

40. Snow, R., O'Connor, B., Jurafsky, D., and Ng, A.Y., Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks, *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP),* Stroudsburg, 2008, pp. 254–263.

41. Tran-Thanh, L., Stein, S., Rogers, A., and Jennings, N.R., Efficient crowdsourcing of unknown experts using bounded multi-armed bandits, *Artif. Intell.,* 2014, vol. 214, pp. 89–111.

42. Wauthier, F.L. and Jordan, M.I., Bayesian bias mitigation for crowdsourcing, *Advances in Neural Information Processing Systems,* 2011, pp. 1800–1808.

43. Welinder, P. and Perona, P., Online crowdsourcing: Rating annotators and obtaining cost-effective labels, *Proc. IEEE Comput. Soc. Conf. Computer Vision and Pattern Recognition Workshops (CVPRW),* 2010, pp. 25–32.

44. Whitehill, J., Wu, T.-f., Bergsma, J., Movellan, J.R., and Ruvolo, P.L., Whose vote should count more: Optimal integration of labels from labelers of unknown expertise, *Advances in Neural Information Processing Systems,* 2009, pp. 2035–2043.

45. Yan, Y., Rosales, R., Fung, G., and Dy, J.G., Active learning from crowds, *Proc. ICML,* 2011, vol. 11, pp. 1161–1168.

46. Zhang, C. and Chaudhuri, K., Active learning from weak and strong labelers, *Advances in Neural Information Processing Systems,* 2015, pp. 703–711.

47. Zhang, J., Sheng, V.S., Wu, J., and Wu, X., Multi-class ground truth inference in crowdsourcing with clustering, *IEEE Trans. Knowl. Data Eng.,* 2016, vol. 28, no. 4, pp. 1080–1085.

48. Zhang, J., Wu, X., and Sheng, V.S., Learning from crowdsourced labeled data: A survey, *Artif. Intell. Rev.,* 2016, vol. 46, no. 4, pp. 543–576.

49. Zhang, Y., Chen, X., Zhou, D., and Jordan, M.I., Spectral methods meet EM: A provably optimal algorithm for crowdsourcing, *Advances in Neural Information Processing Systems,* 2014, pp. 1260–1268.

50. Zheng, Y., Cheng, R., Maniu, S., and Mo, L., On optimality of jury selection in crowdsourcing, *Proc. 18th Int. Conf. Extending Database Technology (EDBT),* 2015.

51. Zheng, Y., Li, G., and Cheng, R., Docs: A domain-aware crowdsourcing system using knowledge bases, *Proc. VLDB Endowment,* 2016, vol. 10, no. 4, pp. 361–372.

52. Zheng, Y., Li, G., Li, Y., Shan, C., and Cheng, R., Truth inference in crowdsourcing: Is the problem solved?, *Proc. VLDB Endowment,* 2017, vol. 10, no. 5, pp. 541–552.

53. Zheng, Y., Wang, J., Li, G., Cheng, R., and Feng, J., Qasca: A quality-aware task assignment system for crowdsourcing applications, *Proc. ACM SIGMOD Int. Conf. Management of Data,* 2015, pp. 1031–1046.

54. Zhong, J., Tang, K., and Zhou, Z.-H., Active learning from crowds with unsure option, *Proc. IJCAI,* 2015, pp. 1061–1068.

55. Ponomarev, A.V., Quality assurance methods in crowd computing systems: Analytical survey, *Tr. S.-Peterb. Inst. Inf. Avtom. Ross. Akad. Nauk,* 2017, vol. 5, no. 54, pp. 152–184.

56. Korshunov, A. and Gomzin, A., Thematic modeling of texts in natural language, *Tr. Inst. Sistemnogo Program. Ross. Akad. Nauk,* 2012.

Translated by Yu. Kornienko