Task Assignment meets Annotator Modeling: Human-LLM Collaborative Annotation with Constraints

Kei Moriyama[1], Kouta Nakayama[2], Yukino Baba[1]
[1]The University of Tokyo, [2]NII LLMC
{kei-moriyama, yukino-baba}@g.ecc.u-tokyo.ac.jp
nakayama@nii.ac.jp

Abstract

Crowdsourced annotators and Large Language Models (LLMs) offer complementary, cost-effective ways to obtain labeled data, yet ensuring high label quality remains challenging. We observe that task features influence the accuracy of humans and LLMs, while real-world constraints, such as per-annotator assignment limits, further complicate allocation. Prior work typically addresses either task features or constraints, but not both. We present an integrated framework that (i) estimates per-task accuracy from task features using a *learning from crowds* model and (ii) incorporates these estimations into a linear programming formulation that assigns tasks under practical constraints. Experimental results demonstrate that the proposed method achieves accuracy comparable to that of baseline methods while satisfying given constraints.[1]

1 Introduction

Labeled data is indispensable for training and evaluating NLP models. Obtaining such data remains labor-intensive and time-consuming. To scale annotation, researchers rely on two sources: crowdsourced human annotators and Large Language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023). Crowdsourcing provides a cost-effective way to engage diverse workers, while recent studies show that LLMs can generate high-quality labels for various tasks (Kim et al., 2024; Pavlovic and Poesio, 2024; Dai et al., 2023; Ding et al., 2023).

Human annotators from crowdsourcing and LLMs have their own strengths and weaknesses. Although LLMs can achieve human-level performance at a lower cost (Ding et al., 2023), they are prone to errors on tasks that require nuanced, domain-specific judgments, such as identifying discrimination or subtle social biases (Felkner et al., 2024), or labeling data for computational social science (Zhu et al., 2023; Ziems et al., 2024). In contrast, human annotators perform well within their expertise but struggle outside it, sometimes making trivial mistakes when commonsense is lacking.

Given these complementary strengths, human–LLM collaborative annotation can increase annotation accuracy. To determine who annotates which task, prediction of the annotator's strengths is required. We refer to this prediction as annotator modeling. In addition, assignment constraints exist in practice, such as per-annotator assignment limits. By integrating prediction of annotator strength and assignment constraints, we obtain annotations that are simultaneously efficient and accurate.

Existing studies address either annotator modeling or constraints. Various methods have been proposed for annotator modeling. For example, Learning to Defer (Mozannar and Sontag, 2020) uses a neural network and iCrowd (Fan et al., 2015) uses a similarity graph of task features. These methods cannot handle assignment constraints because they assign tasks to the annotator estimated to be the most accurate. In contrast, assignment methods that consider constraints utilize historical annotation accuracy. They address the assignment problem using various algorithms, such as linear programming (Dickerson et al., 2018). These methods preserve constraints; however, they do not consider the relationship between tasks and annotators.

In this paper, we propose to integrate annotator modeling into annotation task assignment with constraints. We apply confusion matrix-based approaches (Rodrigues and Pereira, 2017) for annotator modeling. This approach models the annotation process for each annotator using a linear layer called confusion matrix. We focus on the probabilistic interpretation of this approach to estimate annotation accuracy. For new tasks, our method (1) estimates annotation accuracy for all annotators, (2) builds a bipartite graph between annotators and

[1]Source code is available at https://github.com/babalablab/task-assignment

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 888–902

tasks, weighted by the estimated accuracy, and (3) assigns tasks using linear programming to satisfy all constraints.

For evaluation, we created an annotated dataset called TweetEval-Annotated (TE-A)[2] from the emoji classification task (Mohammad et al., 2018) in TweetEval (Barbieri et al., 2020) because existing datasets are not suitable for evaluation. Annotations are collected from crowdsourced annotators and an LLM. The evaluation is conducted with three practical experimental settings and two constraints. The first setting assumes a large amount of data and annotation, the second setting changes the amount of data, and the third setting varies the annotation rate. We set a maximum-assignment constraint and a cost constraint in our experiments. In most experimental settings, our method achieves accuracy comparable to baseline methods under given constraints regardless of data amount and annotation sparsity. Results show the effectiveness of combining annotator modeling and task assignment. Moreover, an experiment on the large-scale synthetic dataset shows that the runtime of linear programming is within an acceptable range.

Our contributions are summarized as follows:

- We introduce annotator modeling into annotation task assignment with constraints to capture the strengths of different annotators (e.g., humans and LLMs) for specific tasks.
- We propose a framework that combines annotator modeling with task assignment strategies, estimating each annotator's confusion matrix and using it to calculate accuracy.
- We created and released a real-world dataset to simulate the task assignment scenario. Experimental results with LLM and human annotators demonstrate that our method achieves competitive accuracy compared to baseline methods without violating constraints.

2 Related Work

2.1 Task Assignment in Crowdsourcing

Crowdsourcing enables the annotation of large-scale data with low cost and high efficiency. Task assignment is a key problem in crowdsourcing for the quality of the annotations. The objectives of this problem are varied, such as to find reliable annotators (Joglekar et al., 2015; Qiu et al., 2016; Miao et al., 2020), and minimize the cost to achieve the target accuracy (Ho et al., 2013; Tu et al., 2019; Ho and Vaughan, 2021). The gold standard test, which collects the annotation for a small subset of data with known correct labels, is often used to estimate workers' abilities (Khan and Garcia-Molina, 2017; Zheng et al., 2015). Constraints are introduced not only to meet the objective, but also to reflect the problem settings (Zheng et al., 2015; Dickerson et al., 2018; Miao et al., 2020; Bhatti et al., 2021). Several algorithms, such as dynamic programming and linear programming, are utilized to assign tasks while satisfying constraints (Chen et al., 2013; Mo et al., 2013; Ho et al., 2013).

These methods are less focused on the estimation of the worker's ability. We address this gap by using *learning from crowds* (Raykar et al., 2010; Rodrigues and Pereira, 2017; Tanno et al., 2019; Chu et al., 2021) to estimate the ability of workers.

2.2 Collaboration of Human and Machine Learning Models

LLMs have shown impressive performance in a wide range of NLP tasks. In the context of annotation, leveraging LLMs can enhance annotation quality while reducing costs (Wang et al., 2021; Ding et al., 2023; Uchendu et al., 2023; Rastogi et al., 2023; Kim et al., 2024; Sharma et al., 2024; Wang et al., 2024). Their capability for annotation support has been confirmed in many problem settings, such as NLI (Liu et al., 2022), speech processing (Jahan et al., 2024), and thematic analysis (Dai et al., 2023).

Collaboration between humans and machine learning models has been explored in the classification problem in safety-critical scenarios (Madras et al., 2017). To ensure safety, some studies proposed training models to defer its prediction to humans when models have low confidence (Mozannar and Sontag, 2020; Verma and Nalisnick, 2022; Hemmer et al., 2022; Narasimhan et al., 2022; Cao et al., 2024; Wei et al., 2024).

3 Problem Setting

We address the problem of assigning unannotated tasks $\hat{\mathcal{T}} = \{\hat{t}_1, \ldots, \hat{t}_{N'}\}$ to annotators $\mathcal{R} = \{r_1, \ldots, r_R\}$ while satisfying specific constraints. Let N' and R be the number of unannotated tasks and annotators, respectively. Fig.1 provides an overview of the problem setting, where the con-

[2]The dataset is available at https://huggingface.co/datasets/keimoriyama/TweetEval_Annotated

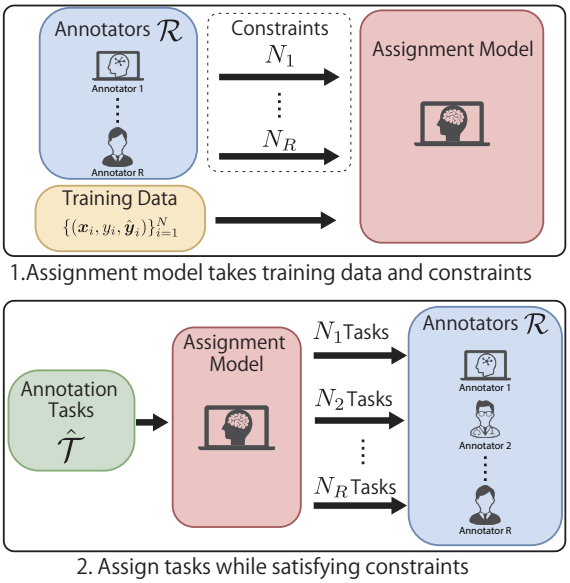

Figure 1: Overview of our problem setting with maximum-assignment constraints. The assignment model receives the set of annotation tasks and the constraints (upper box). For each task, the assignment model selects an annotator who can provide the correct annotation without violating the constraint (lower box).

straint limits the maximum number of assignments per annotator. Eq.(1) formalizes this constraint.

$$\sum_{i=1}^{N'} e_{ij} \leq N_j \quad \forall j \in \{1, \ldots, R\} \tag{1}$$

where N_j is the maximum number of assignments allowed for r_j, and e_{ij} is the indicator variable that equals 1 if task $\hat{t}_i$ is assigned to r_j, and 0 otherwise.

Past annotations from each annotator are provided to estimate the annotator's expertise. Let $\{(\boldsymbol{x}_i, y_i, \hat{\boldsymbol{y}}_i)\}_{i=1}^N$ be the dataset, where N is the number of the training dataset, $\boldsymbol{x}_i$ is the feature representation of i-th task, y_i is its ground truth, which belongs to one of C classes. Each task is annotated by up to R annotators, and their annotations are represented as $\hat{\boldsymbol{y}}_i = \{\hat{y}_{ij}\}_{j=1}^R$. Since not all annotators necessarily label every task, an additional label is introduced to indicate cases where annotation is not provided. If an annotator provides a label, it is represented as a class $c \in \{1, \ldots, C\}$; otherwise, it is assigned -1 to indicate the absence of an annotation.

The assignment model estimates the annotation characteristics from past annotations. It receives unannotated tasks and constraints, then assigns tasks to annotators while satisfying the constraints.

4 Method

4.1 Overview of the Assignment Strategy

We treat the task assignment problem as a constrained matching problem on a weighted bipartite graph $(\hat{\mathcal{T}} \cup \mathcal{R}, E)$, where E is a set of indicators e_{ij}. This graph has a weight w_{ij} representing the accuracy of the j-th annotator for the i-th task. Fig.2 shows the overview of our method.

Our method assigns tasks to annotators using linear programming. Task assignment with constraints is known to be NP-hard (Miao et al., 2020; Fan et al., 2015). Linear programming is commonly used to find a feasible solution (Bhatti et al., 2021; Dickerson et al., 2018). Additionally, this approach allows us to specify various constraints related to task assignment.

The objective function of our method aims to find values of e_{ij} that maximize the sum of estimated assignment accuracy. The assignment by linear programming is formulated as shown in Eq.(2).

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N'} \sum_{j=1}^{R} w_{ij} e_{ij} \\ \text{s.t.} \quad & \text{Const}(E) \leq A \end{aligned} \tag{2}$$

where $\text{Const}(E)$ is a function for the constraint calculation and A is the set of constraints.

4.2 Estimation of Annotation Accuracy

To find the optimal solution in Eq.(2), it is necessary to estimate the accuracy for the given task. We use CrowdLayer (CL) (Rodrigues and Pereira, 2017) to estimate annotation accuracy. CL is a method to train the classifier directly from annotations, rather than using ground truth. It introduces a linear layer called confusion matrix, which maps the classifier's output to the annotation distribution. Confusion matrices absorb noise in the annotations during backpropagation and propagate the information about ground truth to the classifier. Eq.(3) shows the annotation distribution of the j-th annotator.

$$p(\hat{y}_{ij}|\boldsymbol{x}_i) = \boldsymbol{W}^j f_\theta(\boldsymbol{x}_i) \tag{3}$$

$f_\theta(\boldsymbol{x}_i)$ is the classifier parameterized by θ, and $\boldsymbol{W}^j$ is the confusion matrix for the j-th annotator.

The parameters of the confusion matrix can be considered as the probability of the annotation given the ground truth label. Therefore, the cal-

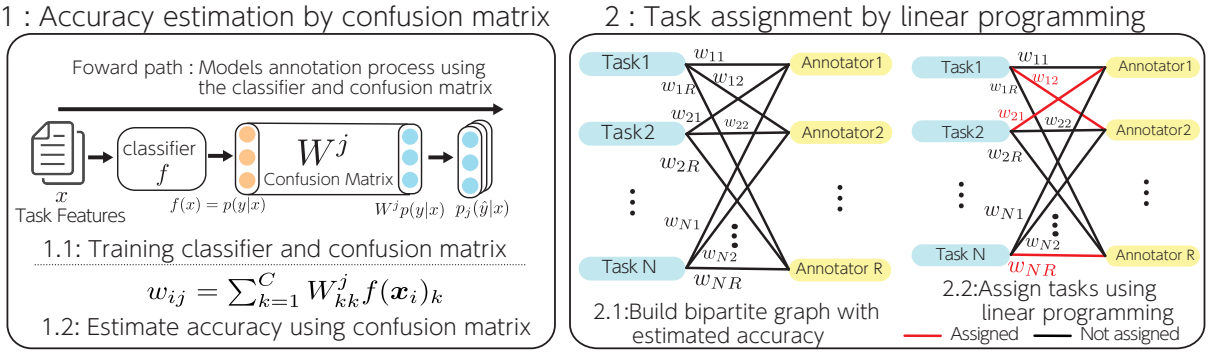

Figure 2: Overview of our method. First, our method estimates the accuracy of the annotators (left square). Second, we assign tasks using the bipartite graph weighted by the estimated accuracy (right square).

culation of Eq.(3) can be interpreted as Eq.(4):

$$p(\hat{y}_{ij} = k|\boldsymbol{x}_i) = \sum_{l=1}^{C} \boldsymbol{W}^j_{kl} f_\theta(\boldsymbol{x}_i)_l$$
$$= \sum_{l=1}^{C} p(\hat{y}_{ij} = k|y_i = l, \boldsymbol{x}_i) p(y_i = l|\boldsymbol{x}_i) \quad (4)$$

where $f_\theta(\boldsymbol{x}_i)_l$ is the l-th value of $f_\theta(\boldsymbol{x}_i)$. Eq.(4) indicates that the (k, l)-th element of the confusion matrix represents the probability that the annotation is k given the ground truth label is l.

The confusion matrices of all annotators $\boldsymbol{W} = \{\boldsymbol{W}^j\}_{j=1}^R$ and the classifier $f_\theta(\boldsymbol{x}_i)$ are trained using the loss function shown in Eq.(5).

$$\mathcal{L}_{\text{conf}}(\theta, \boldsymbol{W}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{R} \sum_{k=1}^{C} \mathbb{1}[\hat{y}_{ij} = k] \log p(\hat{y}_{ij} = k|\boldsymbol{x}_i) \quad (5)$$

where $\mathbb{1}[\cdot]$ is an indicator function.

We use the agreement rate between the annotation and the ground truth label as the edge weight of the bipartite graph. The calculation of the agreement rate is shown in Eq.(6).

$$p(\hat{y}_{ij} = y_i|\boldsymbol{x}_i) = \sum_{k=1}^{C} p(\hat{y}_{ij} = k|y_i = k, \boldsymbol{x}_i) p(y_i = k|\boldsymbol{x}_i)$$
$$= \sum_{k=1}^{C} \boldsymbol{W}^j_{kk} f_\theta(\boldsymbol{x}_i)_k \quad (6)$$

Finally, Eq.(7) shows the loss function to train the classifier and the confusion matrix.

$$\mathcal{L}(\theta, \boldsymbol{W}) = \mathcal{L}_{cls}(\theta) + \alpha \mathcal{L}_{\text{conf}}(\theta, \boldsymbol{W}) \quad (7)$$

	Experimental Settings		
	Case 1	Case 2	Case 3
Data Annotation	Full Large	Sampled Large	Full Partial

Table 1: Overview of experimental settings. The first row indicates the number of annotated data, *Full* uses all data and *Sampled* uses partial data for the accuracy estimation. The second row indicates the given annotations: *Large* denotes that all data are annotated, while *Partial* denotes that the data are randomly annotated.

where α is a hyperparameter that balances the two losses. Since the ground truth label is available in the problem setting, we use cross-entropy loss $\mathcal{L}_{cls}(\theta)$ as the additional loss function.

5 Experiment

5.1 Experimental Settings

We evaluated our method in the following three practical settings. The differences in the experimental settings are summarized in Table 1.

Case 1: Full Annotation on Large Dataset
This setting assumes the expansion of an existing dataset. In this scenario, a large amount of data and annotations are available for accuracy estimation. We used the entire dataset and annotations, without considering cases where annotations were missing.

Case 2: Full Annotation on Small Dataset
This setting simulates a case where task assignment is based on the small subset of data. The sampled dataset is used to estimate annotation accuracy. Accuracy changes are evaluated at different sampling rates.

Case 3: Partial Annotation This setting considers assigning tasks using partially annotated data. In crowdsourcing, it is common for annotators to label only a subset of the data rather than the entire dataset. To simulate this situation, partial annotations are provided for training. We evaluated the accuracy change based on the annotation rates.

5.2 Evaluation Metrics

The primary evaluation metric is assignment accuracy. Eq.(8) shows the calculation of this metric.

$$\frac{1}{N'}\sum_{i=1}^{N'}\mathbb{1}[y'_i = \hat{y}'_{ij}] \quad \text{where} \quad j = \arg\max_{k\in\{1,\ldots,R\}} e_{ik} \tag{8}$$

where y'_i is a ground-truth label of i-th task and $\hat{y}'_{ij}$ is an annotation label of j-th annotator. Under the maximum-assignment constraint, we also evaluate the assignment rate under the specified limit. Under the cost constraint, violations are assessed using the assignment cost rate. We report the mean and standard deviation of the accuracy, and mean constraint violation. In all figures, the error bar shows the standard deviation. Due to space limitations, we use the Gini index to visualize the assignment rate under the maximum-assignment constraint. The Gini index is a fairness metric, with higher values indicating greater bias in assignment. This metric is suitable because the assignment numbers for each annotator are set to be nearly equal under the maximum-assignment constraint.

5.3 Dataset

To evaluate assignment accuracy, annotations from every annotator are required. Existing datasets cannot be utilized because they do not satisfy this requirement. Therefore, we created a new dataset called **TweetEval-Annotated** (**TE-A**). This dataset is created from the emoji prediction task (Mohammad et al., 2018) in TweetEval dataset (Barbieri et al., 2020). This task predicts the emoji that follows a given text. Since emoji selection reflects individual preferences, annotations are expected to vary across annotators. This variability makes the assignment problem challenging, making it suitable for evaluation. We selected four labels (😜,😊,📷,🔥) for the annotation and randomly sampled 750 texts for each selected class. Annotations were collected from four human annotators and one LLM. Table 2 shows the accuracy of all annotators. We use the embedding from BERT (Devlin et al., 2019) as a task feature.

Annotator	1	2	3	4	LLM
Accuracy	0.35	0.33	0.38	0.48	0.44

Table 2: Overall accuracy of each annotator in TE-A.

Methods	Constraint	Task feature
Random	Satisfied	✗
AW	Satisfied	✗
iCrowd	Not Satisfied	✓
L2D	Not Satisfied	✓
Ours	Satisfied	✓

Table 3: Summary of differences between baseline methods and our method. The *Constraint* column indicates whether the method ensures that the constraint is satisfied. The *Task feature* column indicates whether the method utilizes task features for assignment. ✓ indicates that the method utilizes task features and ✗ indicates that the method does not utilize task features.

5.4 Task-Assignment Constraints

We use two types of constraints in our experiments.

Maximum-assignment constraint Eq.(1) formulates this constraint. We set the limit for each annotator to $N_j = \left\lceil \frac{N'}{R} \right\rceil$, ensuring that the sum of limits equals the size of the dataset.

Cost constraint Eq.(9) formulates this constraint.

$$\sum_{i=1}^{N'} c_j e_{ij} \leq C_j \quad \forall j \in \{1,\ldots,R\} \tag{9}$$

where c_j is the assignment cost per task and C_j is the maximum total cost for the j-th annotator. We derive c_j from the actual annotation cost of each annotator for TE-A.

Eq.(10) shows the one-annotator-per-task constraint, which is applied to all experimental settings and constraints.

$$\sum_{j=1}^{R} e_{ij} = 1 \quad \forall i \in \{1,\ldots,N'\} \tag{10}$$

5.5 Baseline Methods

Since this is the first work to introduce annotator modeling into constrained task assignment, we

Constraint	Method	Accuracy
Maximum Assignment Constraint	Random	0.39 ± 0.01
	AW	0.39 ± 0.02
	iCrowd	0.47 ± 0.03
	L2D	$\mathbf{0.48} \pm 0.03$
	Ours	0.42 ± 0.02
Cost Constraint	Random	0.39 ± 0.02
	AW	0.44 ± 0.02
	iCrowd	0.47 ± 0.03
	L2D	$\mathbf{0.48} \pm 0.03$
	Ours	0.45 ± 0.03

Table 4: Assignment accuracy of Case 1 in TE-A. **Bold** indicates the highest accuracy.

Maximum-Assignment Constraint			
Annotator	iCrowd	L2D	Ours
1	0.218	0.000	1.000
2	0.022	0.000	1.000
3	0.051	0.000	1.000
4	**4.147**	**5.000**	1.000
LLM	0.562	0.000	1.000
Cost Constraint			
1	0.194	0.000	0.940
2	0.028	0.000	0.198
3	0.066	0.000	0.086
4	**5.474**	**6.600**	1.000
LLM	0.100	0.000	0.528

Table 5: Assignment rate (%) in the maximum assignment and assignment cost rate (%) in the cost constraint in Case 1. **Bold** indicates the constraint violation

compared our method with two heuristic weighting methods for a bipartite graph, as well as two task assignment methods that use task features but do not consider constraints. Table 3 summarizes differences between baseline methods and our method.

Random This method assigns tasks randomly, while ensuring the constraints. We report the mean accuracy over multiple assignments.

Accuracy Weighting (AW) This method weights a bipartite graph with the accuracy of past annotations and then assigns tasks by optimizing the objective in Eq.(2) using linear programming, subject to the constraints.

Learning to Defer (L2D) This is a method to improve human-AI performance in classification tasks by jointly training a classifier and a rejector (Mozannar and Sontag, 2020). We implement the rejector, which predicts the confidence of the annotator, as a baseline. Rejector assigns tasks to an annotator whose estimated confidence is the highest.

iCrowd This method estimates the accuracy of the annotator based on a similarity graph (Fan et al., 2015). The graph has an edge between task vertices, when their similarity exceeds a threshold. Tasks are assigned to an annotator whose estimated accuracy is the highest.

6 Results

6.1 Case 1: Full Annotation on Large Dataset

Table 4 shows assignment accuracy under the maximum-assignment and cost constraints. The accuracy of L2D and iCrowd is equal in the two constraint settings because we use the same data for training and testing. Table 5 shows the evaluation of constraint violations. These results show that our method achieves comparable accuracy to baseline methods under given constraints. In both constraint settings, the accuracy of our method is second to L2D or iCrowd and better than AW. For constraint evaluation, our method has no constraint violations, unlike iCrowd and L2D. Baseline methods tend to assign many tasks to the fourth annotator whose accuracy in the training dataset is higher than other annotators. These methods are prone to assigning high-performance annotators.

The combination of annotator modeling and linear programming is effective in the constrained task assignment problem. In this problem setting, assignment becomes biased when estimated accuracy is directly applied, as shown in L2D and iCrowd. Using the accuracy of past annotations as the edge weights is insufficient to achieve accurate assignment, as shown by the AW results. Our method reduces the assignment bias by utilizing the estimated accuracy as the weights on the edges of the bipartite graph between tasks and annotators.

6.2 Case 2: Full Annotation on Small Dataset

Fig.3 shows the accuracy change under the maximum-assignment and cost constraints. The change of the Gini index is in Fig.4. The accuracy of our method is better than that of heuristic methods and comparable to iCrowd and L2D without

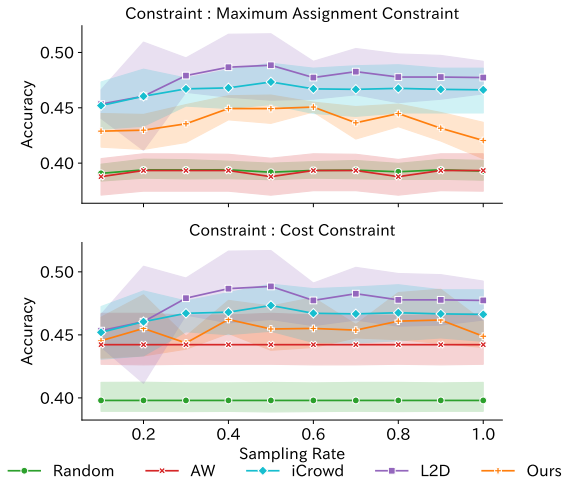

Figure 3: Accuracy change in Case 2 for both constraint settings and datasets.

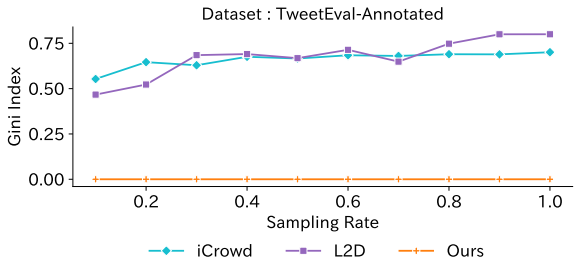

Figure 4: Change in the Gini index under the maximum-assignment constraint in Case 2.

violating the constraints.

Although iCrowd and L2D show higher accuracy than our method, their Gini index is also higher. Moreover, the Gini index of baseline methods increases as the amount of training data increases. As the availability of training data improves the accuracy estimation of these methods, it consequently amplifies bias in their assignments. Our method shows comparable accuracy to baseline methods under the constraint regardless of dataset size. In addition, the standard deviation of accuracy is lower than that of baseline methods across all sampling rates under both constraints. These results demonstrate that our approach effectively balances accuracy and constraints, providing a robust assignment across varying data scales.

6.3 Case 3: Partial Annotation

Fig.5 shows the accuracy change with the annotation rate for both constraints and Fig.6 presents the Gini index. Under both constraints, our method outperforms heuristic methods and achieves comparable performance to iCrowd and L2D. Notably, at

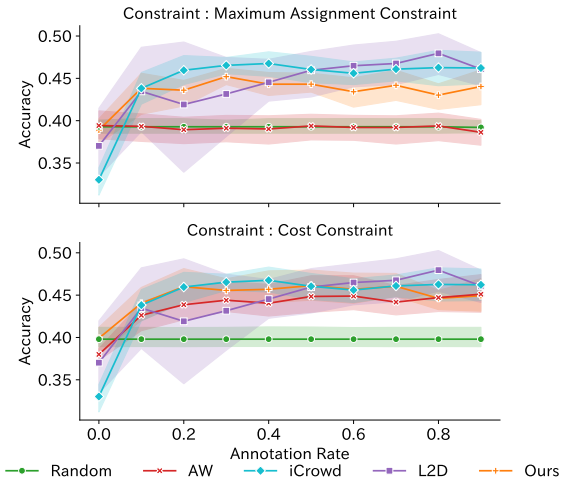

Figure 5: Accuracy change in Case 3 for both constraint settings and datasets.

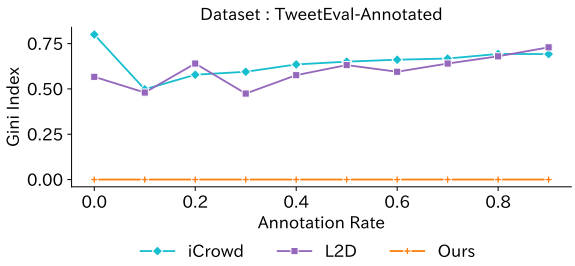

Figure 6: Change in the Gini index under the maximum-assignment constraint in Case 3.

low annotation rates, its accuracy remains competitive. Although iCrowd and L2D show higher accuracy, their assignments have significant constraint violations. This result shows that our method achieves better assignment under given constraints and learns better weights for the bipartite graph regardless of assignment rate.

Annotator modeling using CL is particularly effective when annotations in the training dataset are sparse. The loss function of CL utilizes information from both the ground truth and the annotation labels, separately. In contrast, baseline methods rely solely on whether each annotation matches the ground truth. When annotations from a specific annotator or ground-truth labels are missing, these methods cannot utilize the corresponding information. In such cases, our method can leverage either ground truth or annotation information. This difference makes our method effective under sparse annotation settings.

6.4 Running Time of Linear Programming

Because our method relies on linear programming, a natural question is whether it remains computationally practical on large-scale datasets. To investigate this, we constructed an artificial dataset and prepared ten artificial annotators. The dataset consists of two-dimensional points classified into three classes. The annotation accuracy of annotators is determined by the ground truth. Each annotator are proficient in one class and less accurate in others. Details of the dataset and annotators are provided in Appendix B. The other experimental settings, such as constraints, are the same as Case 1.

We measured the computational time required to solve the linear programming problem.[3] The evaluation was performed with test datasets of sizes 10k, 30k, 50k, and 100k instances. We report the mean and standard deviation of the running time across five different random seeds.

Table 6 presents the runtime of linear programming. For both constraint settings, the runtime remains low when the dataset size is 10k. However, when the dataset sizes exceed 30k, the runtime increases. In the 30k setting, the average runtime is approximately one minute, and in the 100k setting, it is about five minutes.

Although assignment takes longer when the dataset is large, runtime remains within an acceptable range. The task assignment problem is a part of annotation preprocessing. It takes several days to complete data annotation on a crowdsourcing platform. Our method receives datasets and constraints from the user and assigns tasks to annotators. Since the assignment time is only a few minutes, it does not pose a practical issue in the overall annotation process.

7 Discussion

The formulation of our method is limited to classification settings because CL is designed for these problems. Therefore, our method can be applied to complex classification problems, such as multi-label classification. However, it cannot be applied directly to non-classification tasks, such as regression or generation tasks. Non-classification tasks need to be converted into classification problems to apply our method. One example is an annotation validation task, which evaluates whether a given annotation is valid or not. For a regression task, the validation label is set to 1 if an annotation is sufficiently close to the ground truth value, and 0 otherwise. Similarly, for a generation task, the label is set to 1 if a sentence satisfies the predefined annotation criteria, and 0 otherwise. However, the calculation of Eq.(6) includes the probability of invalid annotation. The equation needs to be reformulated as follows:

$$p(\hat{y}^j = 1|\boldsymbol{x}_i) = \boldsymbol{W}^j_{11} f_\theta(\boldsymbol{x}_i)_1 \tag{11}$$

Our method can be extended to non-classification tasks by using this value as edge weights in the bipartite graph.

[3]We used the `time.process_time` function in the `time` module of Python library.

Constraint	Test size	Running Time (s)
Maximum Assignment Constraint	10k	19.853 ± 0.93
	30k	78.338 ± 1.47
	50k	138.602 ± 3.26
	100k	279.843 ± 6.30
Cost Constraint	10k	20.709 ± 0.54
	30k	79.142 ± 1.67
	50k	136.723 ± 1.54
	100k	286.547 ± 7.09

Table 6: Mean and standard deviation of the running time for linear programming (in seconds).

8 Conclusion

In this paper, we validate the effectiveness of combining annotator modeling and task assignment. We have introduced confusion matrix-based annotator modeling into task assignment to consider the relationship between task features and annotations. To the best of our knowledge, this is the first work that combines annotator modeling and task assignment. To simulate the real-world crowdsourcing settings in the experiment, we constructed a dataset called TE-A. We evaluated accuracy in three experimental settings and assignment runtime. For all experimental settings, our method reduces the accuracy gap to baseline methods. The running time of linear programming is modest on the large-scale dataset. In summary, our method achieves competitive assignment accuracy with a practical runtime, even in large-scale settings.

9 Limitations

Our research has several limitations. The first limitation is that each task is assigned to a single annotator in our experiments. It is common for one task

to be assigned to multiple annotators since majority voting is often used in crowdsourcing to improve annotation reliability. We acknowledge that the effectiveness of our method remains unclear in such scenarios.

Second, the running time depends on constraints and computational resources. The set of feasible solutions is a key factor in determining the runtime of linear programming. If we apply additional constraints that leave a large feasible set, the effect of such constraints on the running time is unclear. When the same experiment runs on less powerful hardware, the running time may be longer. The computational resources used in our experiment are provided in Appendix C.

References

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650.

Shahzad Sarwar Bhatti, Jiahao Fan, Kangrui Wang, Xiaofeng Gao, Fan Wu, and Guihai Chen. 2021. An approximation algorithm for bounded task assignment problem in spatial crowdsourcing. *IEEE Transactions on Mobile Computing*, 20(8):2536–2549.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS 2020*, pages 1877–1901.

Chengzhi Cao, Yinghao Fu, Sheng Xu, Ruimao Zhang, and Shuang Li. 2024. Enhancing human-AI collaboration through logic-guided reasoning. In *The Twelfth International Conference on Learning Representations*, ICLR 2024.

X Chen, Qihang Lin, and Dengyong Zhou. 2013. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *International Conference on Machine Learning*, volume 28 of *ICML 2013*, pages 64–72.

Zhendong Chu, Jing Ma, and Hongning Wang. 2021. Learning from crowds by modeling common confusions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 of *AAAI 2021*, pages 5832–5840.

Shih-Chieh Dai, Aiping Xiong, and Lun-Wei Ku. 2023. LLM-in-the-loop: Leveraging large language model for thematic analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9993–10001.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, NAACL 2019, pages 4171–4186.

John P Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2018. Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS 2018, pages 318–326.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is GPT-3 a good data annotator? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2023, pages 11173–11195.

Ju Fan, Guoliang Li, Beng Chin Ooi, Kian-Lee Tan, and Jianhua Feng. 2015. iCrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD 2015, pages 1015–1030.

Virginia K Felkner, Jennifer A Thompson, and Jonathan May. 2024. GPT is not an annotator: The necessity of human annotation in fairness benchmark construction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, pages 14104–14115.

Patrick Hemmer, Sebastian Schellhammer, Michael Vössing, Johannes Jakubik, and Gerhard Satzger. 2022. Forming effective human-AI teams: Building machine learning models that complement the capabilities of multiple experts. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, IJCAI 2022, pages 2478–2484.

Chien-Ju Ho, S Jabbari, and Jennifer Wortman Vaughan. 2013. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *ICML 2013*, pages 534–542.

Chien-Ju Ho and Jennifer Vaughan. 2021. Online task assignment in crowdsourcing markets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26 of *AAAI 2012*, pages 45–51.

Maliha Jahan, Helin Wang, Thomas Thebaud, Yinglun Sun, Giang Ha Le, Zsuzsanna Fagyal, Odette

Scharenborg, Mark Hasegawa-Johnson, Laureano Moro Velazquez, and Najim Dehak. 2024. Finding spoken identifications: Using GPT-4 annotation for an efficient and fast dataset creation pipeline. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, LREC-COLING 2024, pages 7296–7306.

Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2015. Comprehensive and reliable crowd assessment algorithms. In *2015 IEEE 31st International Conference on Data Engineering*, ICDE 2015, pages 195–206.

Asif R Khan and Hector Garcia-Molina. 2017. CrowdDQS: Dynamic question selection in crowdsourcing systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD 2017, pages 1447–1462.

Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, and Dan Zhang. 2024. MEGAnno+: A human-LLM collaborative annotation system. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, EACL 2024, pages 168–176.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, ICLR 2015.

Alisa Liu, Swabha Swayamdipta, Noah A Smith, and Yejin Choi. 2022. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847.

David Madras, Toniann Pitassi, and Richard Zemel. 2017. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NeurIPS 2017, pages 6150–6160.

Xin Miao, Yanrong Kang, Qiang Ma, Kebin Liu, and Lei Chen. 2020. Quality-aware online task assignment in mobile crowdsourcing. *ACM transactions on sensor networks*, 16(3):1–21.

Luyi Mo, Reynold Cheng, Ben Kao, Xuan S Yang, Chenghui Ren, Siyu Lei, David W Cheung, and Eric Lo. 2013. Optimizing plurality for human intelligence tasks. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM 2013, pages 1929–1938.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17.

Hussein Mozannar and David Sontag. 2020. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning*, ICML 2020, pages 7076–7087.

Harikrishna Narasimhan, Wittawat Jitkrittum, A Menon, A Rawat, and Surinder Kumar. 2022. Post-hoc estimators for learning to defer to an expert. In *Advances in Neural Information Processing Systems 35*, NeurIPS 2022, pages 29292–29304.

Maja Pavlovic and Massimo Poesio. 2024. The effectiveness of LLMs as annotators: A comparative overview and empirical analysis of direct representation. In *Proceedings of the 3rd Workshop on Perspectivist Approaches to NLP*, LREC-COLING 2024, pages 100–110.

Chenxi Qiu, Anna C Squicciarini, Barbara Carminati, James Caverlee, and Dev Rishi Khare. 2016. CrowdSelect: Increasing accuracy of crowdsourcing tasks through behavior prediction and user selection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM 2016, pages 539–548.

Charvi Rastogi, Marco Tulio Ribeiro, Nicholas King, Harsha Nori, and Saleema Amershi. 2023. Supporting human-AI collaboration in auditing LLMs with LLMs. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, AIES 2023, pages 913–926.

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(43):1297–1322.

Filipe Rodrigues and Francisco Pereira. 2017. Deep learning from crowds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI 2017, pages 1611–1618.

Ashish Sharma, Sudha Rao, Chris Brockett, Akanksha Malhotra, Nebojsa Jojic, and William B Dolan. 2024. Investigating agency of LLMs in human-AI collaboration tasks. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, EACL 2024, pages 1968–1987.

Ryutaro Tanno, Ardavan Saeedi, Swami Sankaranarayanan, Daniel C Alexander, and Nathan Silberman. 2019. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, CVPR 2019, pages 11244–11253.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jiayang Tu, Peng Cheng, and Lei Chen. 2019. Quality-assured synchronized task assignment in crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):1–1.

Adaku Uchendu, Jooyoung Lee, Hua Shen, Thai Le, Ting-Hao 'kenneth' Huang, and Dongwon Lee. 2023. Does human collaboration enhance the accuracy of identifying LLM-generated deepfake texts? In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 11 of *AAAI 2023*, pages 163–174.

Rajeev Verma and Eric Nalisnick. 2022. Calibrated learning to defer with one-vs-all classifiers. In *Proceedings of the 39th International Conference on Machine Learning*, ICML 2022, pages 22184–22202.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics*, ACL 2021, pages 4195–4205.

Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024. Human-LLM collaborative annotation through effective verification of LLM labels. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI 2024, pages 1–21.

Zixi Wei, Yuzhou Cao, and Lei Feng. 2024. Exploiting human-AI dependence for learning to defer. In *Proceedings of the 41st International Conference on Machine Learning*, ICML 2024, pages 52484–52499.

Yudian Zheng, Jiannan Wang, Guoliang Li, Reynold Cheng, and Jianhua Feng. 2015. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD 2015, pages 1031–1046.

Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. 2023. Can chatgpt reproduce human-generated labels? a study of social computing tasks. *Preprint*, arXiv:2304.10145.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291.

A Annotation Process of TE-A dataset

For LLM annotation, we used Llama-3.1-8B-Instruct (Touvron et al., 2023)[4]. The prompt contains few-shot demonstrations for each category, as shown in Fig.7. Table 7 shows the amount of the dataset in the TE-A dataset.

[4] https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct

Predict the emoji that follows the given text.
The list of emoji is "📷", "🔥", "😊" and "😜".
Your answer should be 0 if the following emoji is "🔥", 1 if the following emoji is "📷", 2 if the following emoji is "😜", and 3 if the following emoji is "😊".
These are examples of the given text and answer.
Text: The powerful track "Black Friday" by Kendrick Lamar
Answer: 0
Text: My weekend adventures. Check out more pics from this shoot at @user! #canon #photojunkie
Answer: 1
Text: Star Wars: The Force Awakens with the fam. Loved it! #starwars #episodevii #movies #family
Answer: 2
Text: A magical night with moonlight dancing. Life is good.
Answer: 3
Now, answer the emoji that follows the given text.
Text: ${target_text}
Emoji:

Figure 7: Prompt for the annotation of TweetEval dataset. ${target_text} is placeholder that is replaced to the annotation text.

B Generation Details of Artificial Dataset

Points used in our experiments are generated using Eq.(12).

$$\begin{cases} x = r\sin(\theta) & \\ y = r\cos(\theta) & \\ \theta = 4j + 4r + \epsilon & j \in \{0, \cdots, C\} \\ r = i/N_p & i \in \{0, \cdots, N_p\} \end{cases} \tag{12}$$

where N_p indicates the number of points per class. We set $\epsilon = 0.7$, the number of classes to $C = 3$. Fig.8 shows an example distribution of artificial data.

Additionally, we generated artificial annotations based on the ground truth label. We assume each annotator has a consistent annotation strength. Following this assumption, we set a single annotation strength for each annotator. We set annotation accuracy for the proficient class to 0.9 and that for the other classes to 0.1.

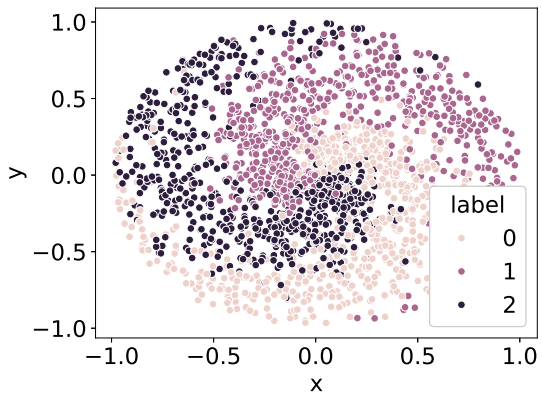

Figure 8: Example of the data distribution in the artificial dataset.

Train (N)	Validation	Test (N')
2,100	450	450

Table 7: Amount of training, validation, and test data of TE-A.

emoji	1	2	3	4	LLM
😜	0.09	0.26	0.14	0.37	0.14
😊	**0.60**	0.37	0.38	**0.65**	0.37
📷	**0.50**	0.30	0.46	0.27	**0.65**
🔥	0.23	0.41	**0.52**	**0.64**	**0.58**

Table 8: Accuracy of the all category by each annotator in TweetEval-Annotated dataset.

Hyperparameter	Artificial	TE-A
α in Eq.(7)	0.01	0.01
Hidden dimension of MLP	4	8
Layer of MLP	3	2
Learning rate	0.01	0.005
Weight decay	0.005	0.005
Training epoch	100	150
Batch size	1024	32

Table 9: Hyperparameters in our experiment. In all settings, we used the same hyperparameters.

C Hyperparameter

The annotation accuracy of the TE-A dataset is provided in Table 8. These tables show that annotators have their own strengths and weaknesses for specific categories. Table 7 provides the split of this dataset. We divided 70% of the data into training and 25% of the data into validation and testing in TE-A datasets.

Table 9 shows the hyperparameters of the artificial dataset and TweetEval-Annotated dataset. We used the same hyperparameters in all experimental settings. We used Adam (Kingma and Ba, 2015) optimizer for the training of our method and L2D. The computational resource in our experiment was an NVIDIA-A100 with 8GB memory. The system has 120 GB of RAM and a clock speed of 3 GHz.

D Assignment rate for each annotator under the maximum-assignment constraint

D.1 Case 2

Fig.9 shows the assignment rate for each annotator in Case 2 in the TE-A dataset.

Assignments by baseline methods are influenced by differences in annotation accuracy regardless of sampling rate. In the TE-A dataset, the assignment is biased to the fourth annotator. As shown in Table 2, the accuracy of the fourth annotator is the highest among the annotators in the TE-A dataset. This result highlights that accuracy is an important factor for baseline methods. Linear programming is a simple and effective solution to deal with this problem.

D.2 Case 3

Fig.10 shows the assignment rate for each annotator in the TE-A dataset in Case 3. In Case 3, the assignment tendency is similar to that of Case 2. In this case, the assignment of L2D has high variance in most assignment rates. This result highlights the instability of L2D in this experimental setting.

E Assignment cost rate under the cost constraint in Case 2 and Case 3

We visualized assignment cost rate for each annotator to evaluate constraint violation by baselines and our method.

Fig.11 and Fig.12 show the assignment cost rate for all annotators in the TE-A dataset in Case 2 and Case 3, respectively. The dashed line in these figures shows the maximum cost rate.

These figures show that baseline methods have significant assignment bias regardless of experimental case. In the TE-A dataset, many tasks are assigned to the fourth annotator.

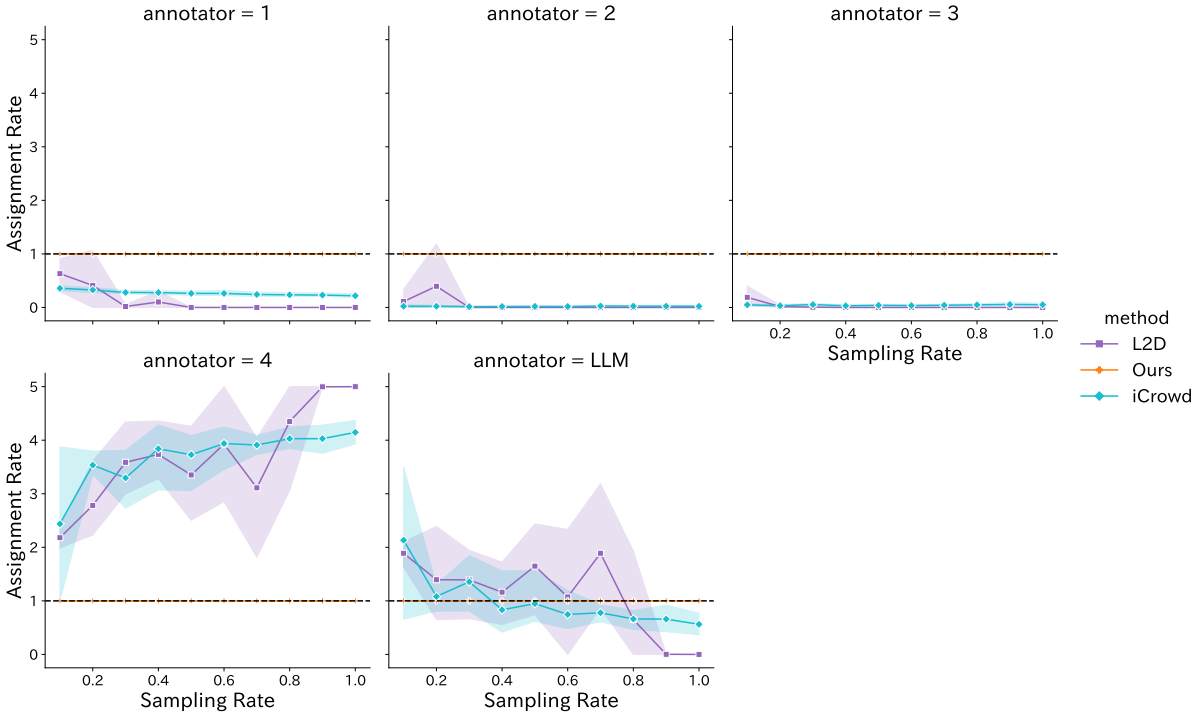

Figure 9: Assignment rate for each annotator in TE-A dataset under the assigment constraint in Case 2.

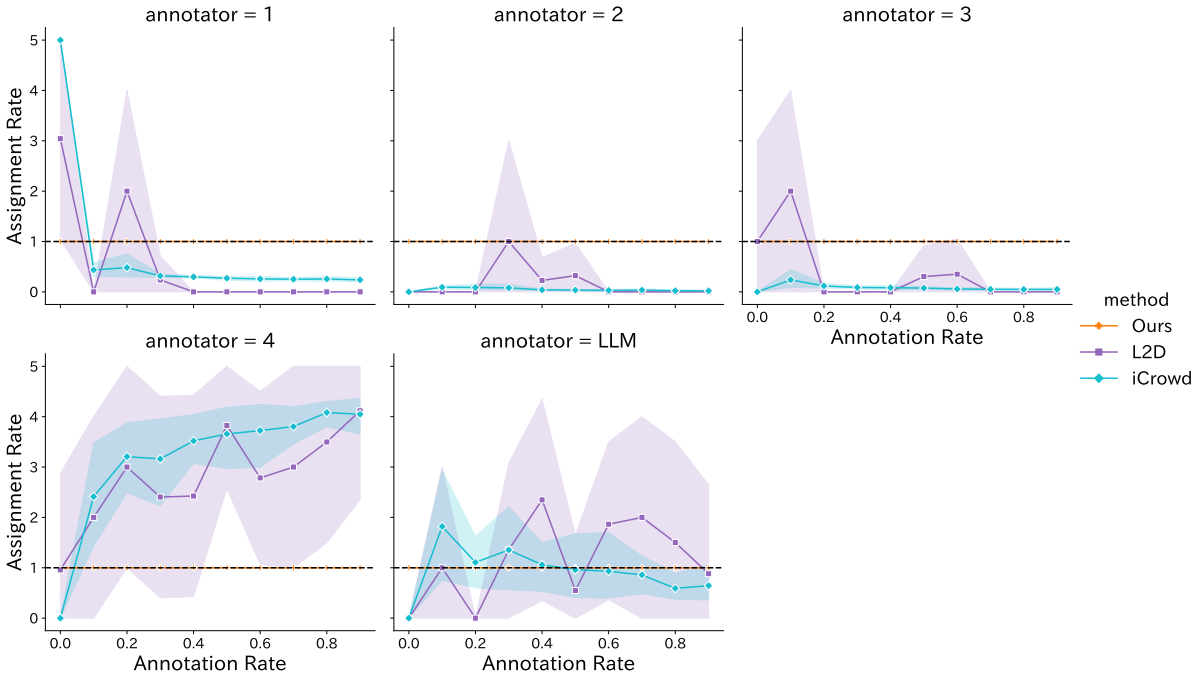

Figure 10: Assignment rate for each annotator in TE-A dataset under the assigment constraint in Case 3.

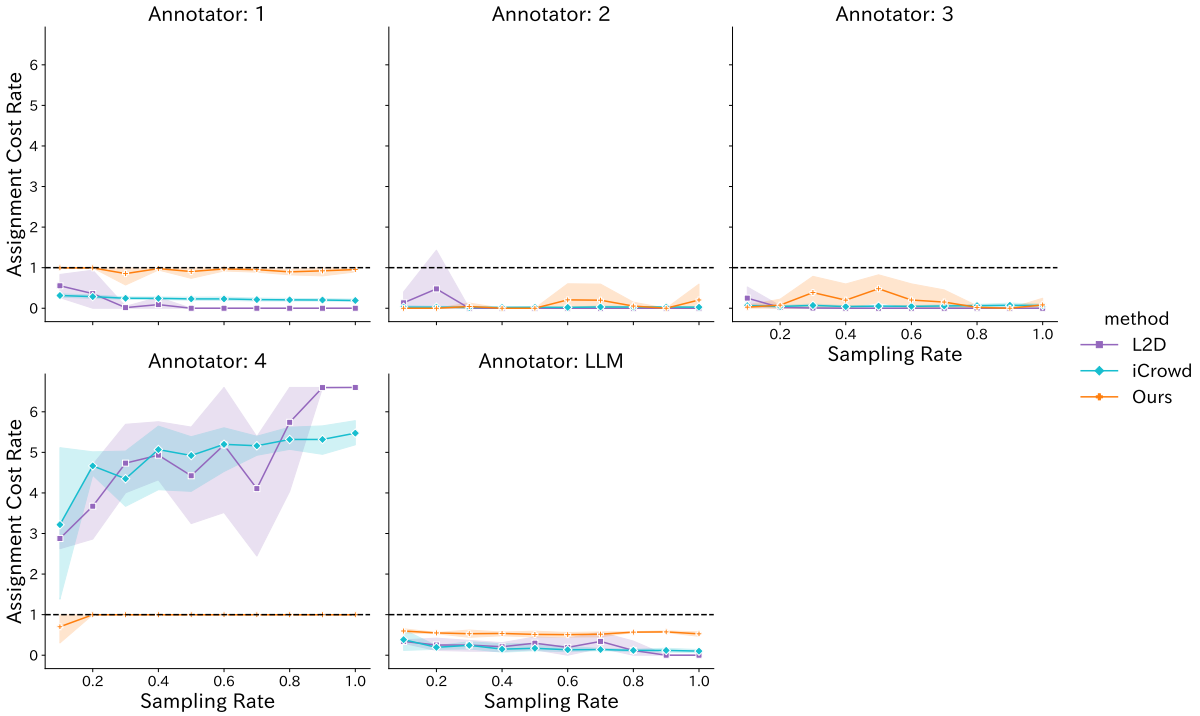

Figure 11: Assignment cost rate for each annotator in TE-A dataset under the cost constraint in Case 2.

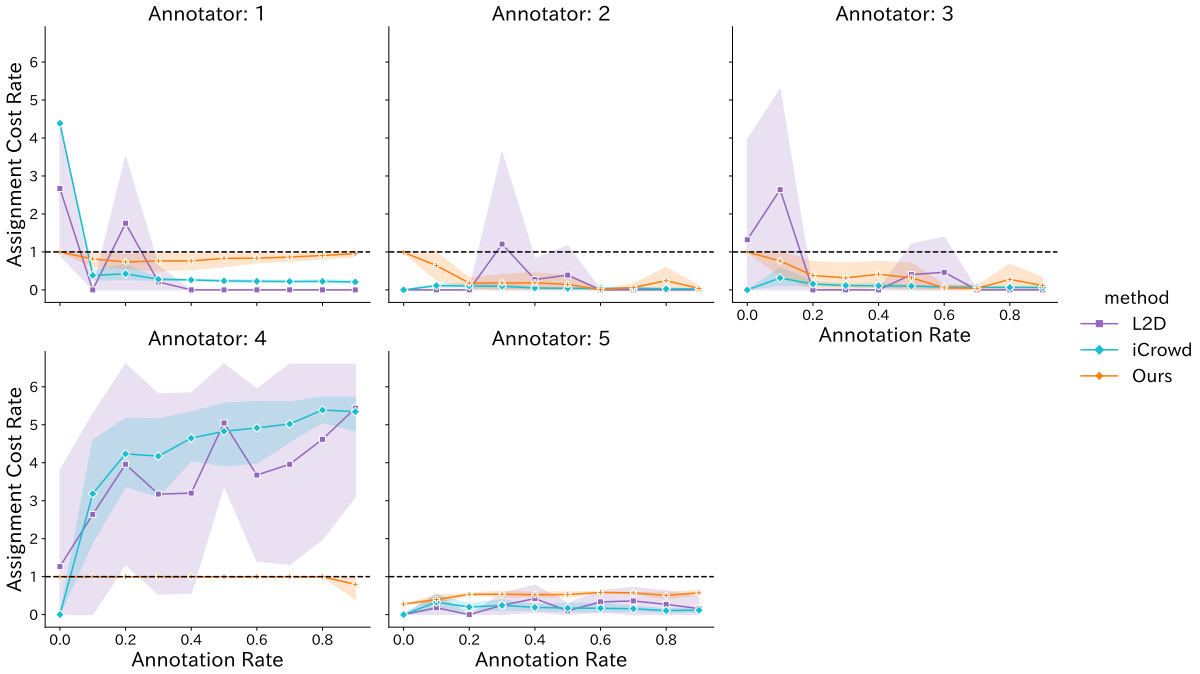

Figure 12: Assignment cost rate for each annotator in TE-A dataset under the cost constraint in Case 3.

F Implementation Details of the Baseline Methods

F.1 Details of L2D

We provide the implementation details of the baseline method L2D. In this method, we implement a rejector that estimates the confidence of the annotator. Let f_θ be the rejector parameterized by θ. Eq.(13) is the loss function for training the rejector.

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{R}\alpha_j \mathbb{1}[y_i = \hat{y}_{ij}] f_\theta(\boldsymbol{x}_i)_j \quad (13)$$

where the α_j is a hyperparameter that controls the assignment priority of the annotator r_j and $f_\theta(\boldsymbol{x}_i)_j$ is the j-th output of the rejector.

This loss function corresponds to the multi-label classification problem, where the target label is set to 1 if the annotation from the annotator r_j is correct. Therefore, the output of the rejector can be interpreted as $p(y_i = \hat{y}_{ij}|\boldsymbol{x}_i)$. The assignment of $\boldsymbol{x}_i$ is determined by selecting the most confident annotator as shown in Eq.(14).

$$a_i = \text{argmax}_{j\in\{0\ldots R\}} p(y_i = \hat{y}_{ij}|\boldsymbol{x}_i) \quad (14)$$

where the a_i indicates the assigned annotator for $\boldsymbol{x}_i$. We set $\alpha_j = 1$ for all annotators to ensure equal task assignment during training.

F.2 Details of iCrowd

We present the implementation details of iCrowd. iCrowd builds a similarity graph for accuracy estimation. This graph has an edge if the similarity of the features is larger than the threshold. We set 0.9 as a threshold in all experiments and datasets.

In the TE-A dataset, we used cosine similarity of text embeddings. For the text embeddings, we used the output of the last layer corresponding to the beginning token of BERT[5].

G Generative AI Assistance

We used ChatGPT[6] and Writefull[7] to polish the sentences of our paper.

Annotator	Cost per task (c_i)	Total cost (C_j)
1	0.078	8
2	0.067	5
3	0.044	3
4	0.044	3
LLM	0.002	8

Table 10: TweetEval-Anntated cost constraint

[5] https://huggingface.co/google-bert/bert-base-uncased

[6] https://chat.openai.com/

[7] https://www.writefull.com/