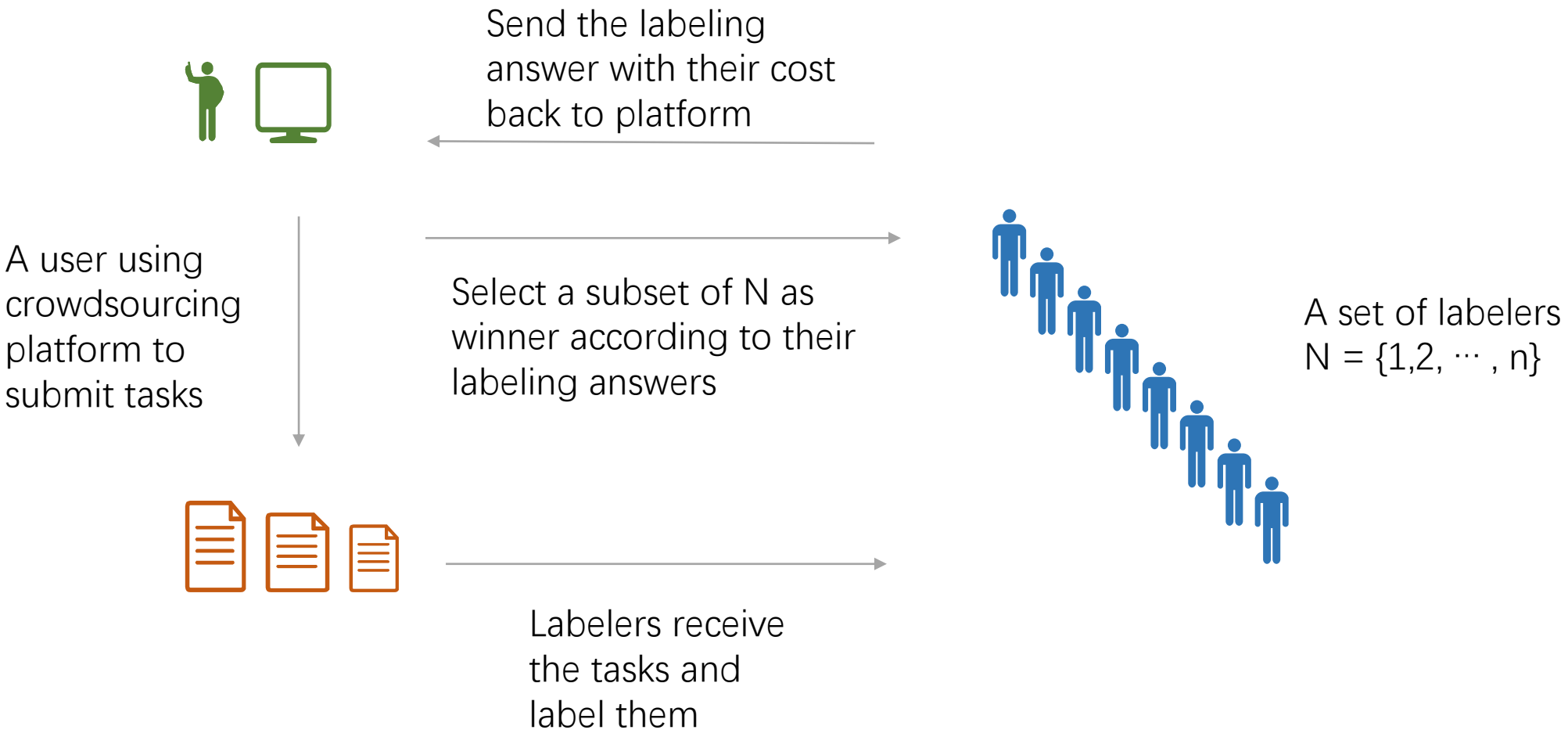
Incentivize Online Multi-class Crowd Labeling under Budget Constraint

杭盖 IEEE 5140309505



How to select labelers to achieve better utility?

The Crowdsourcing Model

- The platform publicizes a set M = {t1, t2,⋯, tm} of multi-class labeling tasks.
- And there is a set of N workers, each one has an arrival time ai $\in$ {1,⋯, T}, and a departure time di $\in$ {1,⋯, T}, ; di $\geqslant$ ai. There are a set of labels K = {1,⋯, K} can be labeled to the tasks.
- Each worker j replies with a set Θj of kj bids, each of which is a task-price pair kj = (tj; bj)
- When a user arrives, the crowdsourcer must decide whether to buy the service of this user, and if so, at what price, before it departs.
- Each winning bid is paid an amount of money pi by the platform.

The Crowdsourcing Model —— Aggregating Labels from workers

Definition 1(Worker's Utility). *The utility of a worker is defined as the difference between the total payment it receives and his total cost.*

$$u_j = \sum_{\theta_j^k \in \Theta_{\mathcal{W}}} (p(\theta_j^k) - c(t_j^k)) \quad (1)$$

We denote a labeling answer from a crowd worker j to task t_i as $y_{ij} \in \{1, 2, 3, ..., K\}$. Each answer y_{ij} has a corresponding vector $\mathrm{A}(a_1, a_2, .., a_K)$. Only one element in A equals to '1', while others equals to '0', Where $a_k = 1$ represents worker j attach label 'k' $(k \in \{1, 2, 3, ..., K\})$ to this task. The platform view y_{ij} as a random variable Y_{ij} before the inquiry.

A soft label set $\Omega_i = \{\omega_0, \omega_1, ..., \omega_{K-1}\}, \omega_k \in [0, 1]$ is used to measure the labeling difficulty of task t_i. ω_k is defined as the probability that the task t_i is labeled as label 'k' by a perfectly reliable worker.

$$\Pr(Y_{ij} = y|\Omega_i) = \prod_{k=1}^{K} (\omega_k)^{a_k}$$

Observing a labeling answer $Y_{ij} = y$, we can calculate the posterior distribution by Bayes' rule:

$$p(\omega_i|Y_{ij} = y) \propto \Pr(Y_{ij} = y|\omega_i) \cdot p(\omega_i) \quad (3)$$

The Crowdsourcing Model —— Aggregating Labels from workers

$\mathrm{A}^0(a_1^0, a_2^0, .., a_K^0)$ is the corresponding vector of answer y_{ij}. Thus, if task t_i receives a_k labels of k, the parameter vector will become $\vec{\alpha}_i^1 = \{\alpha_1^0 + a_1, \alpha_2^0 + a_2, ..., \alpha_K^0 + a_K\}$ and the posterior will become

$$p(\Omega_i|\mathbf{y}_i) = \mathrm{Dir}(\vec{\alpha}_i^1) \tag{4}$$

When we have no prior knowledge about the task, we can simply set $\vec{\alpha}_i^0 = \{1, 1, ..., 1\}$, so that the prior is a uniform distribution. The true label z_i is inferred in accordance with the parameter $\omega_{i,k}$ in the soft label set Ω_i, which indicates that $\mathbb{E}[\omega_{i,k}] \geq \mathbb{E}[\omega_{i,l}]$ $(\omega_{i,l} \in \Omega_i, l \neq k)$ implies $z_i = k$.

Problem Formulation

Definition 2(PlatForm Utility). *Platform utility is defined as the Kullback-Leibler divergence between the initial distribution and the final distribution of the soft labels.*

$$u_p(\Omega_{\mathcal{W}}) = \sum_{i=1}^{N} \mathrm{KL}(p(\Omega_i)||p(\Omega_i|\mathbf{y}_i)) \tag{8}$$

Under a strict budget constraint B, the platform aims to determine a winning bid set that maximizes its utility in expectation, i.e.

$$\text{Maximize } \mathbb{E}[u_p(\Theta_{\mathcal{W}})] \text{ s.t. } \sum_{\theta \in \Theta_{\mathcal{W}}} p(\theta) \leq B \tag{9}$$

Online Allocation and Payment Scheme

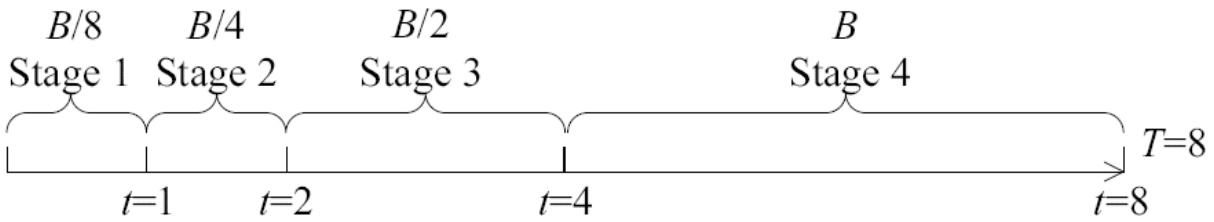
The marginal value of θ_j^k at round r is defined as the K-L divergence between $p^r(\Omega_i)$ and $p^{r+1}(\Omega_i)$ in expectation.

$$v(\vartheta_j^k; r) = \mathbb{E}_{Y_{ij}}[\mathrm{KL}(p^r(\Omega_i)||p^{r+1}(\Omega_i))] \tag{13}$$

Once θ_j^k is selected into $\Theta_{\mathcal{W}}$, $v(\theta_j^k)$ is used instead of $v(\theta_j^k; r)$ to simplify notations. The total value of $\Theta_{\mathcal{W}}$ is simply the sum of all marginal contributions, i.e.

$$V(\Theta_{\mathcal{W}}) = \sum_{\theta_j^k \in \Theta_{\mathcal{W}}} v(\theta_j^k) \tag{14}$$

Multiple-stage sampling accepting process



First, we divide all of T time steps into $(\lfloor log_2 T \rfloor + 1)$ stages: $\{1, 2, ..., \lfloor log_2 T \rfloor, \lfloor log_2 T \rfloor + 1\}$. The stage i ends at time step $T' = \lfloor 2^{i-1} T / 2^{\lfloor log_2 T \rfloor} \rfloor$. Correspondingly, the stage-budget for the i-th stage is allocated as $B' = \lfloor 2^{i-1} B / 2^{\lfloor log_2 T \rfloor} \rfloor$. When a stage is over, we add all users who have arrived into the sample set S', and compute a density threshold ρ^*. according to the information of samples and the allocated stage-budget B'.

Algorithm 1: DI-GreedyOnline

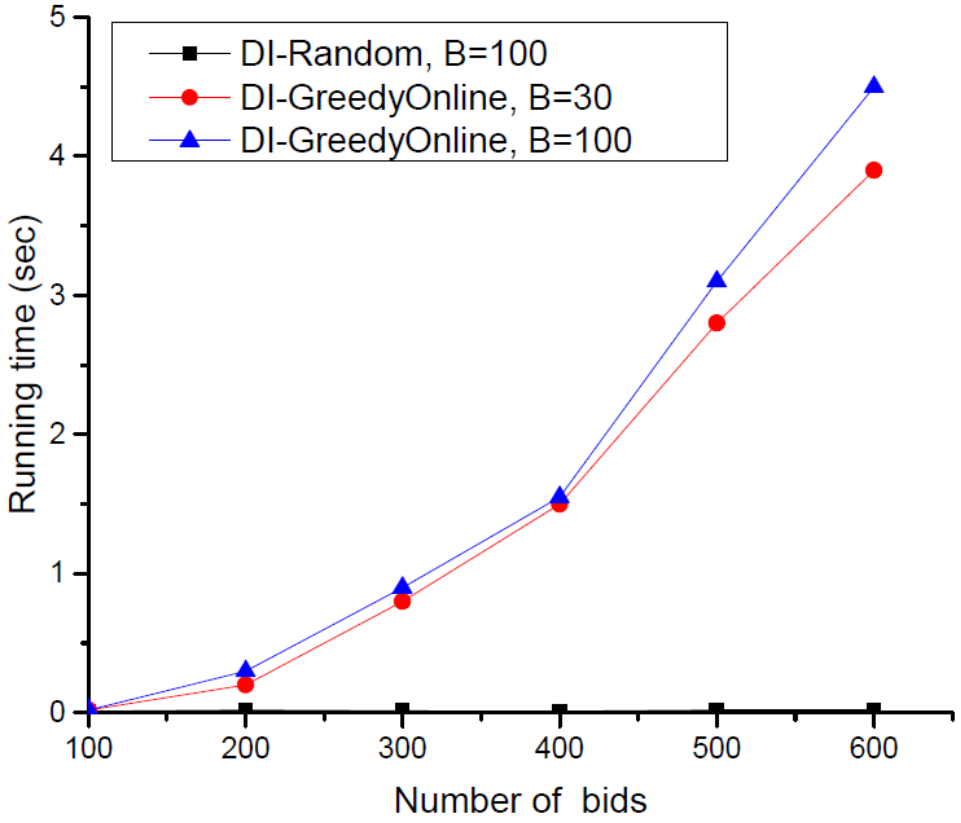
Input: budget B, deadline T; prior parameters $\{\vec{\alpha}_i^0\}_{i=1}^M$, $\{\vec{\beta}_i^0\}_{j=1}^N$;
Output: $\Theta_{\mathcal{W}}$; $V(\Theta_{\mathcal{W}})$; posterior parameters $\{\vec{\alpha}_i^r\}_{i=1}^M$, $\{\vec{\beta}_i^r\}_{j=1}^N$;
Initialize: $r \leftarrow 0$; $\Theta_{\mathcal{W}} \leftarrow \emptyset$;
$(t, T', B', S', \rho^*, \Theta_{\mathcal{W}}) \leftarrow (1, \frac{T}{2^{\lfloor \log_2 T \rfloor}}, \frac{B}{\lfloor \log_2 T \rfloor}, \emptyset, \epsilon, \emptyset)$
while $t \leq T$ **do**
  Add all bids of new users arriving at time step t to a set of online bids $\mathcal{O}$;
  $\mathcal{O}' \leftarrow \mathcal{O}' \setminus S$;
  while $\mathcal{O}' \neq \emptyset$ **do**
    $\theta^* = \{t_{j^*}, b_{j^*}\} \leftarrow \arg\max_{\theta_j^k \in \mathcal{O}'} v(\theta_j^k; r)$;
    if $b_{j^*} \leq v(\theta_j^k)/\rho^* \leq B' - \sum_{l \in \Theta_{\mathcal{W}}} p_l$ **then**
      $p(\theta^*) \leftarrow v(\theta_j^k)/\rho^*$;
      $\Theta_{\mathcal{W}} \leftarrow \Theta_{\mathcal{W}} \cup \{\theta^*\}$;
      Observe the labeling answer $Y_{i^*j^*} = y_{i^*j^*}$;
      And update the posterior distribution of the task.
    else
      $p(\theta^*) \leftarrow 0$;
    $\mathcal{O}' \leftarrow \mathcal{O}' \setminus \{\theta^*\}$;
  Remove all users departing at time step t from $\mathcal{O}$, and add them to S';
  if $t = \lfloor T' \rfloor$ **then**
    $\rho^* \leftarrow \textbf{GetDensityThreshold}(B', S')$;
    $T' \leftarrow 2T'$; $B' \leftarrow 2B'$; $\mathcal{O}' \leftarrow \mathcal{O}$;
    while $\mathcal{O}' \neq \emptyset$ **do**
      $\theta^* = \{t_{j^*}, b_{j^*}\} \leftarrow \arg\max_{\theta_j^k \in \mathcal{O}'} v(\theta_j^k; r)$;
      if $b_{j^*} \leq v(\theta_j^k)/\rho^* \leq B' - \sum_{l \in \Theta_{\mathcal{W}}} p_l + p_j$ **then**
        $p(\theta^*) \leftarrow v(\theta_j^k)/\rho^*$;
        if $\theta^* \notin \Theta_{\mathcal{W}}$ **then**
          $\Theta_{\mathcal{W}} \leftarrow \Theta_{\mathcal{W}} \cup \{\theta^*\}$;
          Observe the labeling answer $Y_{i^*j^*} = y_{i^*j^*}$;
          And update the posterior distribution of the task.
      $\mathcal{O}' \leftarrow \mathcal{O}' \setminus \{\theta^*\}$;
  $t \leftarrow t + 1$
Return: $\Theta_{\mathcal{W}}; V(\Theta_{\mathcal{W}}); \{\vec{\alpha}_i^r\}_{i=1}^M; \{\vec{\beta}_j^r\}_{j=1}^N$;

GreedyOnline

It consists of two stages:

- The current time step t is not at the end of any stage.
- The current time step is just at the end of some stage.

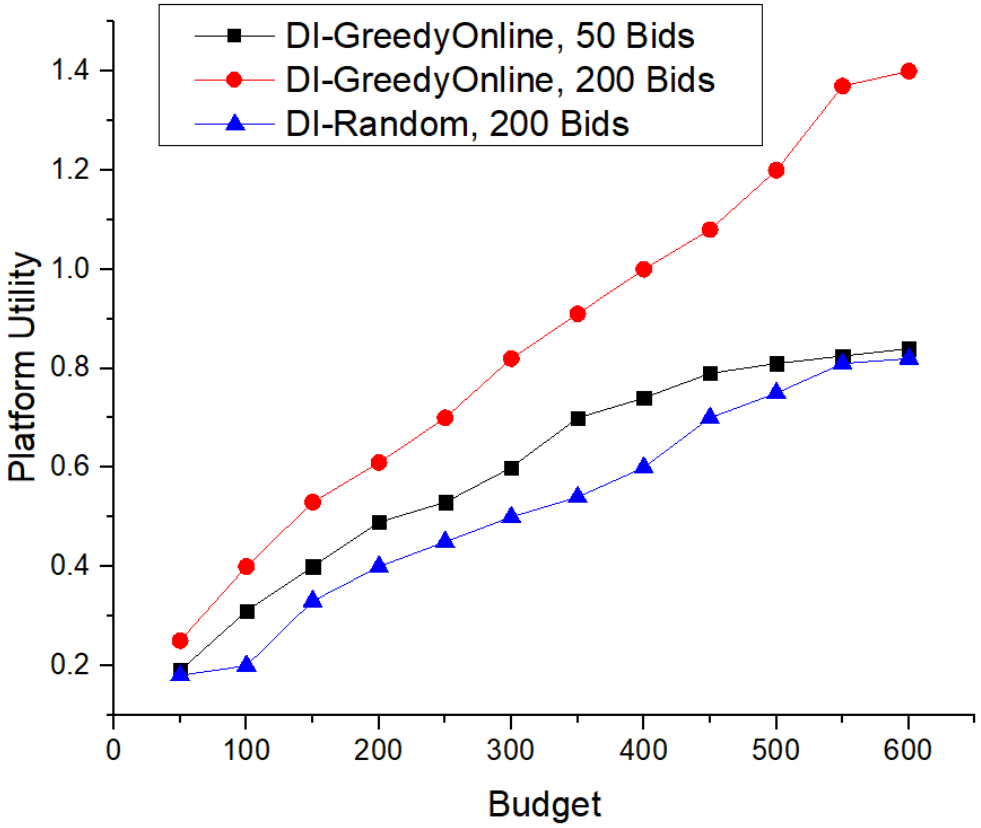
Performance Evaluation



Running Time

DI-RANDOM outperforms the DI-GREEDYONLINE since it doesn't do sorting operation. However, DI-GREEDYONLINE is computational efficient because it runs in polynomial time and the running time ofDI-GREEDYONLINE increases lineally with the number of bids N, while the budget B doesn't affect running time much.

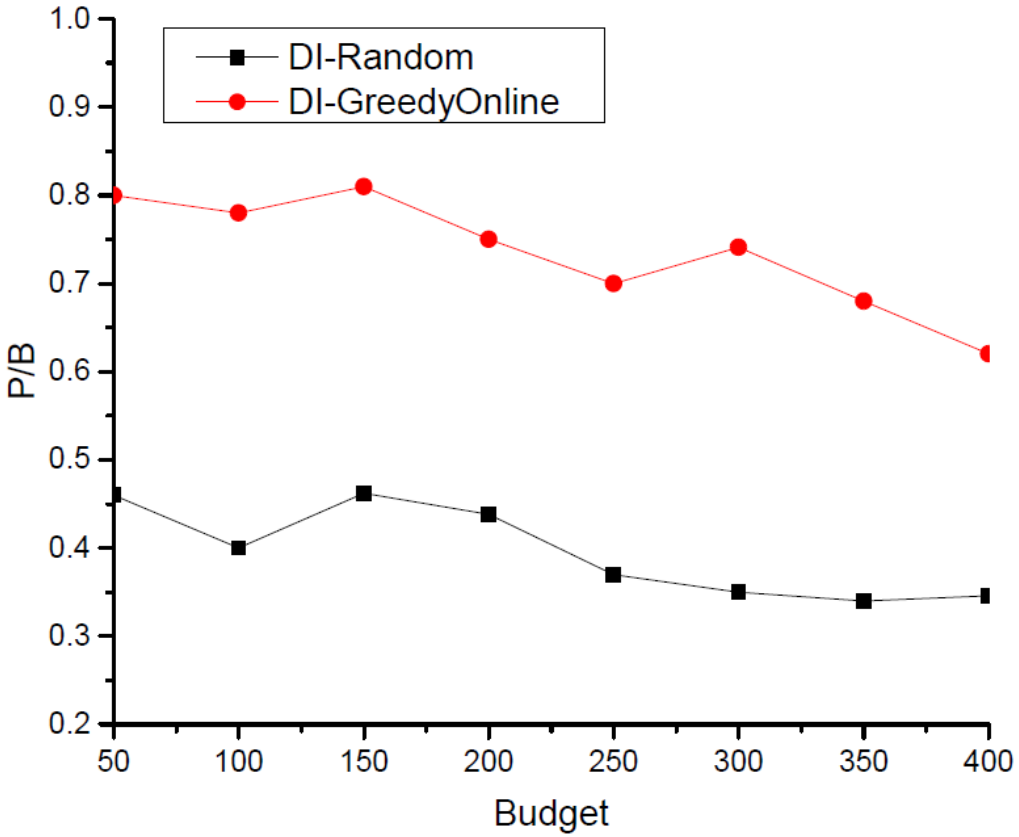
Performance Evaluation



Platform Utility

We observe that DI-GREEDYONLINE always outperforms DI-RANDOM for all budget constraints. As the number of bids N reduces to 50, platform utility also decreases, which reveals that a large amount of bids brings more utility.

Performance Evaluation



Budget Utilization

The results show that our allocation algorithms utilize budget much more efficient than the random allocation scheme.

Conclusion

In this paper, we have developed the reverse auction based incentive mechanism to a more general online scenario. We take into account the difficulty of labeling tasks and focus on maximizing the utility of the crowdsourcer. We have designed a online winning bid allocation algorithm $DI - GreedyOnline$ that can be applied to a more general case where the workers arrive one by one in a random order.

Thank you