1 Response To Reviewer k9DA

Dear Reviewer k9DA, we sincerely thank you for your valuable feedback on our submission. Below is our responses to the concerns you raised. We have incorporated the following contents into the updated version of our paper, which we believe will help enhance the quality of our submission.

> **Q1: The motivation is not enough clear. Why the previous method cannot perform well in both accuracy, convergence, and scalability?**

A1: Previous methods often focus on a single optimization objective (lines 64–65, Table 1 in the original manuscript) while neglecting the optimization of other attributes. In the related work section (lines 107–118), we specifically explain that the Random, Arena, and AlpacaEval methods each focus solely on optimizing accuracy, convergence, or scalability, respectively. Furthermore, we demonstrate in Table 2 through experiments that these three methods result in unbalanced sampling across multiple dimensions due to their failure to simultaneously consider all three optimization objectives. This ultimately leads to their inability to perform well across all three attributes simultaneously.

> **Q2: The runtime of the previous CBE method ($O(NMM)$) is one of the major limitations, and the author starts from this limitation as one of the motivations for the proposed method. However, they lack the runtime analysis for the UniCBE but only an approximate number for saving time when compared to the previous method.**

A2: We fully agree with your suggestion to include an analysis of UniCBE's runtime. To address this, we provide the following statistics: when a win rate error Δ of less than 0.02 is required, UniCBE needs a preference budget of $T = 2800$ (compared to $T = 3200$ for Random). Similarly, when Δ is less than 0.01, UniCBE requires $T = 9400$ (compared to $T = 11300$ for Random). Additionally, the computational complexity of $O(NMM)$ is $N \cdot M \cdot (M-1)/2 = 84525$. Therefore, under specific evaluation accuracy requirements, UniCBE significantly reduces the preference budget needed.

> **Q3: While UniCBE shows promising results for scenarios with periodically introduced new models, it may be less efficient in highly dynamic, real-time evaluation settings where new models or samples are constantly introduced at high frequencies.**

A3: We believe that testing UniCBE in a highly dynamic, real-time evaluation setting can help us more comprehensively assess its performance. To this end, we conduct the following experiments: Starting with a sample size of $N = 600$ and model number of $M = 12$, we execute a random operation at each time step. The operations included: adding one model to be evaluated with a probability of 0.01, removing one model with a probability of 0.01, adding one potential sample with a probability of 0.01, randomly deleting one sample with a probability of 0.01, and taking no action with a probability of 0.96. Based on the experimental results shown in Figure 1, we have the following observations:

- The convergence speed of all baseline methods significantly slowed down. None of the baseline methods achieve a Spearman correlation coefficient of 0.96 or a Pearson correlation coefficient of 0.97 by $T = 2000$, highlighting the difficulty of model evaluation in this setting. In contrast, UniCBE achieve rapid convergence, reaching a Spearman coefficient of approximately 0.97 and a Pearson coefficient exceeding 0.98 by $T = 2000$.
- Over the long term, as T increases, UniCBE consistently demonstrates over 10% savings in preference budget across all metrics, even under this challenging setting, showcasing its strong practicality.
- An interesting observation is that AlpacaEval exhibits better convergence in the early stages compared to Random and Arena, supporting our previous conclusions in Table 1 (original manuscript). However, as T increases, AlpacaEval's lack of accuracy optimization objective leads to its performance being surpassed by Random and Arena.

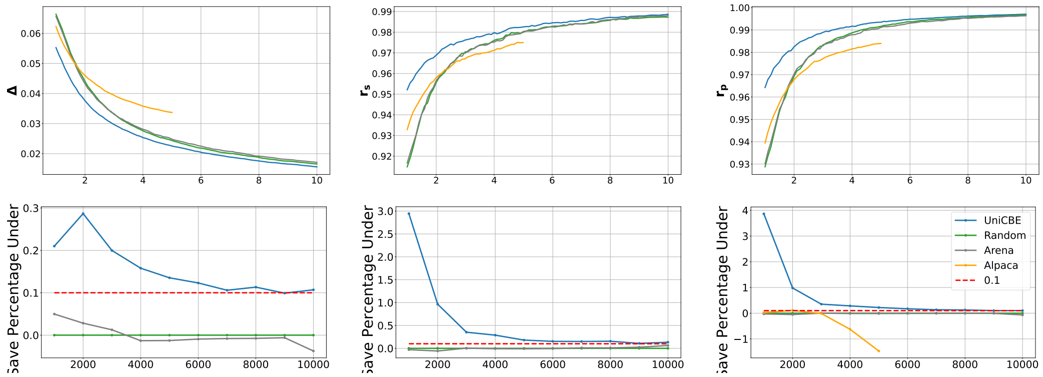

Figure 1: Results of compared CBE methods in a scenario where models and samples are dynamically added or removed at a random frequency.