Which LLM to Play? Convergence-Aware Online Model Selection with Time-Increasing Bandits

Yu Xia[1,2], Fang Kong[1], Tong Yu[3], Liya Guo[4], Ryan A. Rossi[3], Sungchul Kim[3], Shuai Li[1]*
[1]Shanghai Jiao Tong University, [2]University of Michigan, [3]Adobe Research, [4]Tsinghua University

Yu Xia
05/16/2024

Motivation

- So many powerful LLMs nowadays!

Motivation

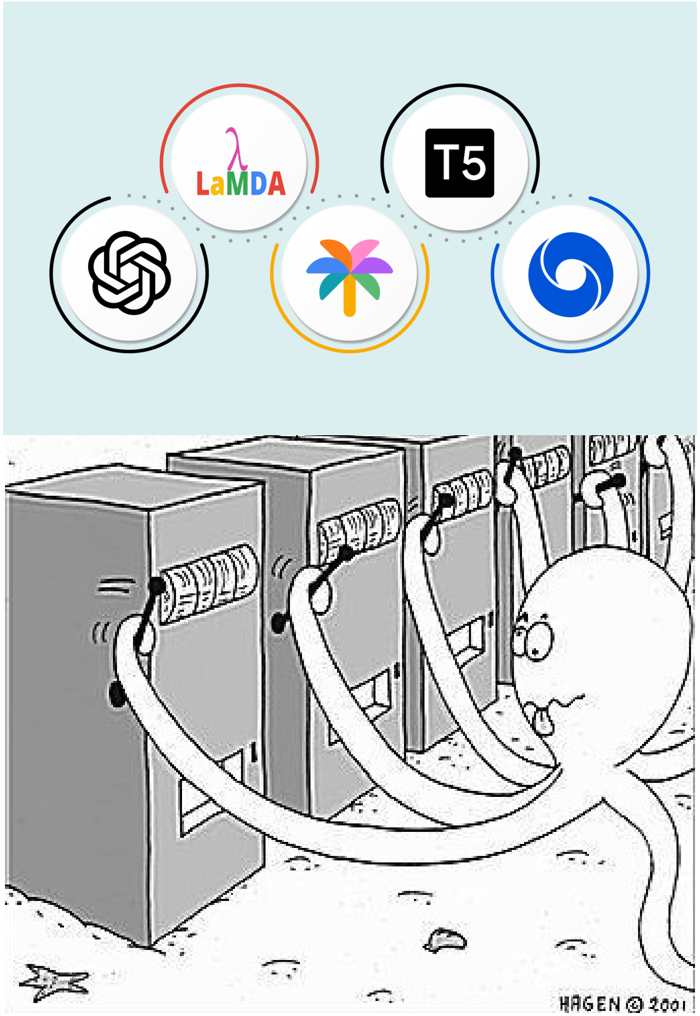
- So many powerful LLMs nowadays!
- Different sizes and different strengths

Motivation

- So many powerful LLMs nowadays!
- Different sizes and different strengths

Then,

Which model should I use for a task?



Traditional Model Selection

- First train all candidates with all data
- Observe all model performances
- Then select the best one

Traditional Model Selection

- First train all candidates with all data
- Observe all model performances
- Then select the best one

However,

LLMs are expensive to train!

Online Model Selection

- Train only one candidate each time per data sample
- Predict all model performances after training
- Select potentially best model for further exploration

Online Model Selection

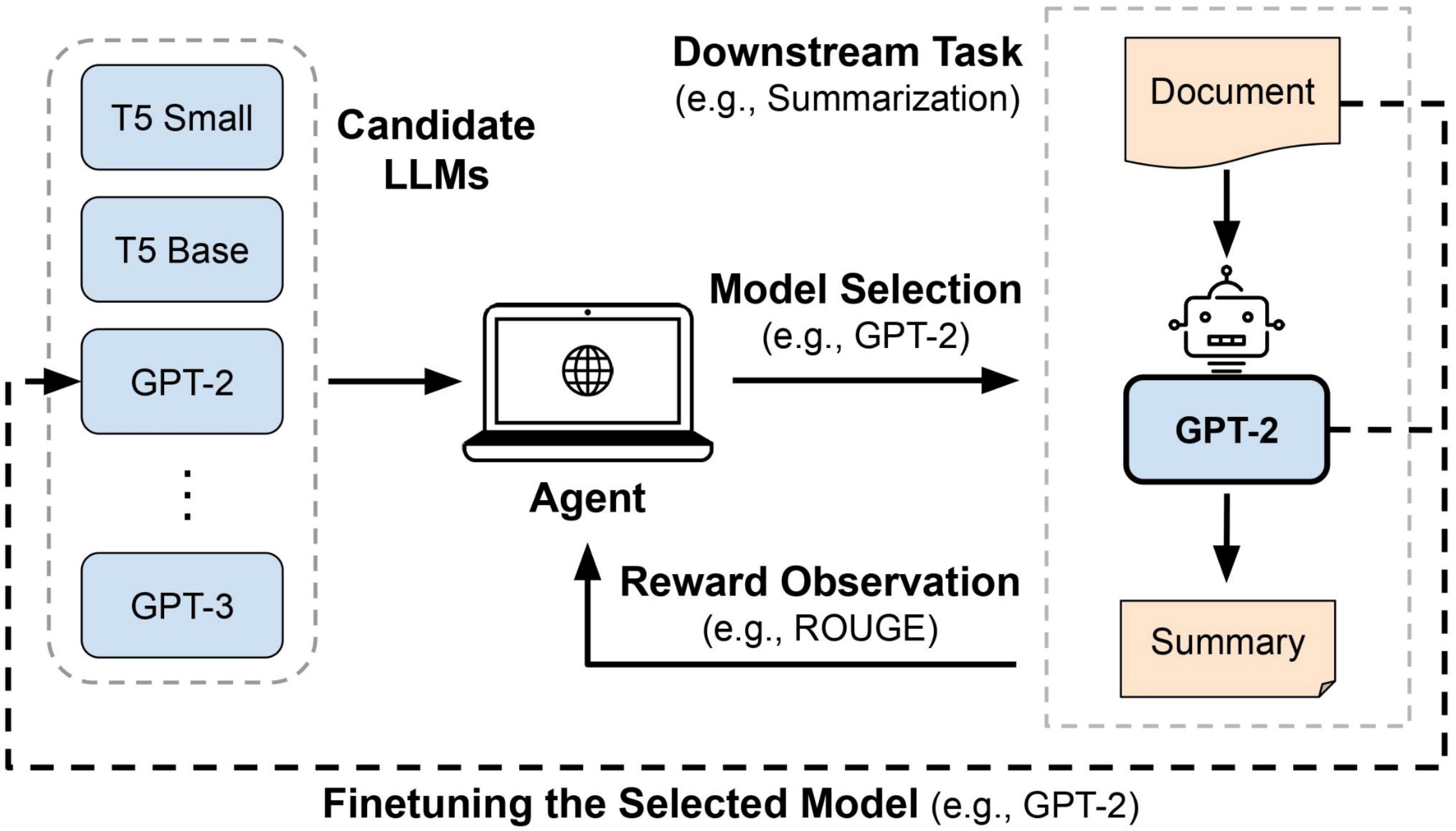


Figure 1: An illustrative example of online model selection for LLM summarization.

Our Method

- Capture the increasing-then-converging reward trend

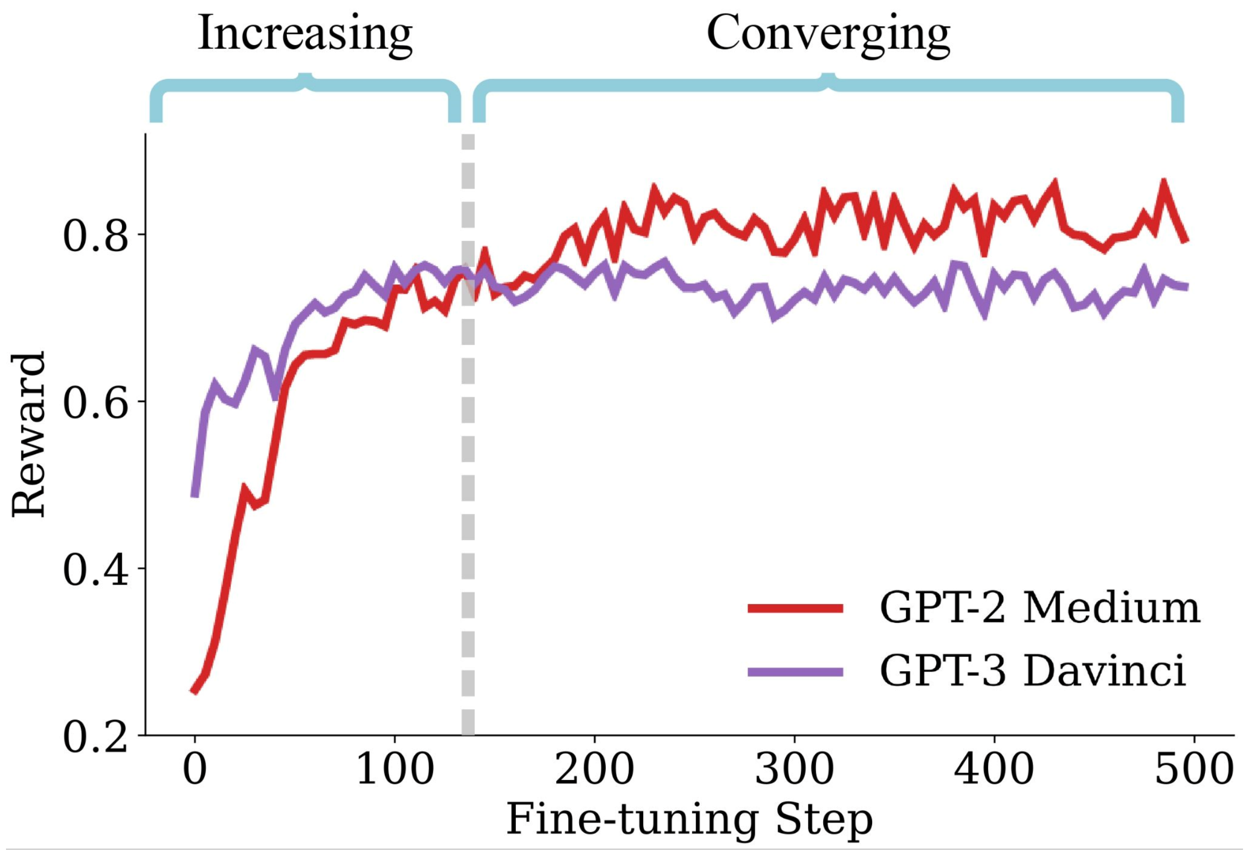


Figure 2: Increasing-then-converging reward trends of an API-based LLM (GPT-3 Davinci) and a local small LLM (GPT2 Medium) over finetuning steps on a text summarization dataset. The reward considers both model performance and finetuning cost.

Our Method

- Multi-armed bandits formulation
  - Each candidate model as an arm
  - Model utility (e.g., performance, cost) as reward
  - The reward of an arm increases each time the arm is pulled (time-increasing reward)
  - Exploration & Exploitation tradeoff

Our Method

- Time-Increasing UCB
- See Appendix A of our paper for theoretical analysis

Algorithm 1 TI-UCB

Input:

K, δ, window size ω, threshold γ;

Output:

Initialize: $\tau_i' \leftarrow 1, n_i \leftarrow 0, \forall i \in [K]$;

for $t = 1, \ldots, T$ **do**

  for $i = 1, \ldots, K$ **do**

    $\bar{\mu}_{i,n_i(t)} = \hat{\mu}_{i,n_i(t)} + 16\sqrt{\frac{2\ln(1/\delta)}{n_i(t)}}$;

  end for

  Pull arm $A_t \leftarrow \text{argmax}_i\, \bar{\mu}_{i,n_i(t)}$;

  Observe reward $X_{A_t,t}$;

  Update estimation $\hat{\mu}_{i,n_i(t)}$;

  Update number of pulls $n_{A_t}(t) \leftarrow n_{A_t}(t) + 1$;

  if $n_{A_t}(t) \geq 2\omega$ **then**

    if $|\hat{\mu}_{w_1,A_t}(t+1) - \hat{\mu}_{w_2,A_t}(t+1)| > \frac{\gamma}{2}$ for arm A_t **then**

      $\tau_{A_t}' \leftarrow t$ and $n_{A_t}(t) \leftarrow 1$;

    end if

  end if

end for

Linear Increase Prediction

Upper Confidence Bound

Sliding Window Change Detection

Our Method

- Logarithmic Regret Upper Bound

Theorem 1. *Assume that $\delta \leq 1/T$, then the expected regret of TI-UCB algorithm is bounded by*

$$\mathbb{E}[R(T)] \leq \sum_{i: n_i(T) \geq n_i^*(T)} c_i \frac{4096 \ln(T)}{\Delta_{\min}^2} + K\left(\frac{2\pi^2}{3} + \omega + 2 + 2L\right) + 2,$$

where $\Delta_{\min} = \min_{t \in [0,T], i \neq i_t^*} \{\mu_{i_t^*}(t) - \mu_i(t)\}$ is the minimum gap between the optimal reward and the true reward and L is a constant smaller than $\ln T$.

- See Appendix B of our paper for proof

Experiments

- Evaluation metric:

  Empirical Cumulative Regret $\widehat{R(T)} = \sum_{i=1}^{K} \left[\sum_{s=1}^{n_i^*(T)} \hat{\mu}_{i,s} - \sum_{s=1}^{n_i(T)} \hat{\mu}_{i,s} \right]$

- Compared Baselines:
  - **KL-UCB** [17]: a classic stationary bandit algorithm utilizing KL Divergence.
  - **Rexp3** [3]: a non-stationary bandit algorithm based on variation budget.
  - **Ser4** [1]: a non-stationary bandit algorithm that takes into account the best arm switches during the process.
  - **SW-TS** [46]: a sliding-window bandit algorithm with Thompson Sampling that generally handles non-stationary settings well.
  - **SW-UCB** [18]: a sliding-window bandit algorithm with UCB that can handle general non-stationary settings.
  - **SW-KL-UCB** [10]: a sliding-window bandit algorithm with KL-UCB.
  - **R-ed-UCB** [33]: a recent non-stationary bandit algorithm designed for similar scenarios as ours with non-decreasing and concave rewards.
  - **Auto-Sklearn** [13]: the state-of-the-art AutoML system utilizing Bayesian optimization-based solution.

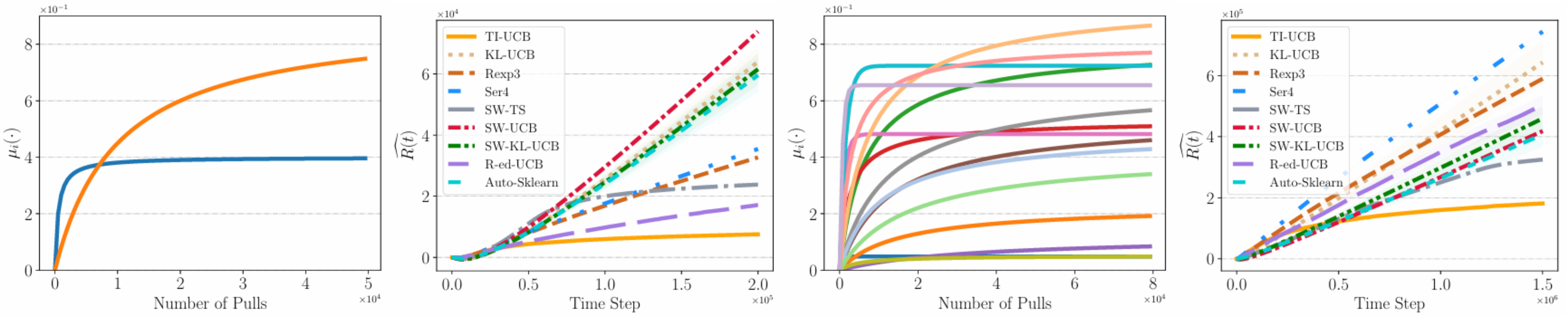
Experiments

- Synthetic model selection

Synthetic reward functions randomly selected from

$$F_{\text{exp}} = \left\{ f(t) = c\left(1 - e^{-at}\right) \right\} \text{ and}$$

$$F_{\text{poly}} = \left\{ f(t) = c\left(1 - b\left(t + b^{1/\rho}\right)^{-\rho}\right) \right\}$$



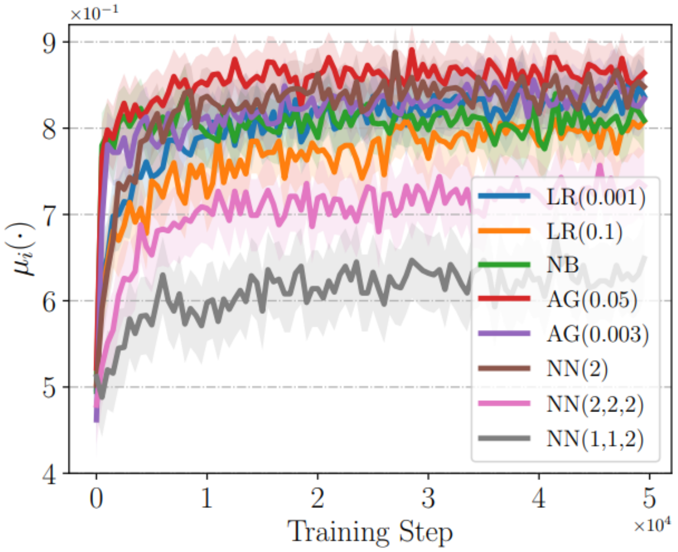
(a) 2-Arm Bandits: Reward Functions (b) 2-Arm Bandits: Cumulative Regret (c) 15-Arm Bandits: Reward Functions (d) 15-Arm Bandits: Cumulative Regret

Figure 3: Online selection of generated synthetic models covering a variety of increasing-then-converging patterns.

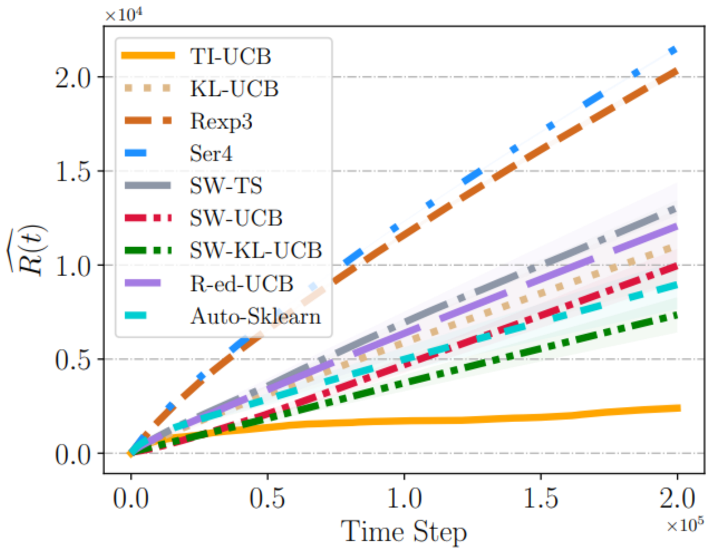
Experiments

- Classification model selection

  Canonical classification models on IMDB review dataset, e.g.,
  LR: logistic regression, NB: naive bayes, NN: neural network.



(a) IMDB Bandits: Reward Functions



(b) IMDB Bandits: Cumulative Regret

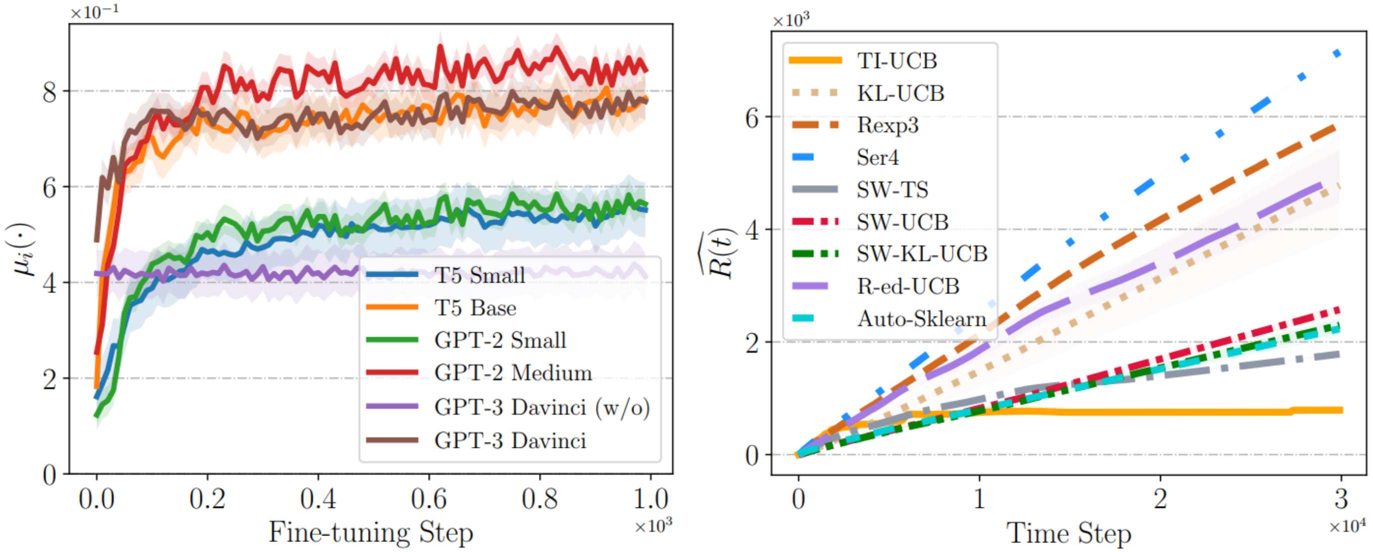
Experiments

- LLM selection

LLMs of different sizes and costs on XSum summarization data.

Reward: $X_t = \text{ROUGE-2} - \eta_t$

Finetuning Cost: $\eta_t = \eta_{t-1} + m \cdot \mathbf{1}\,[\text{Do Finetuning}]$ with $\eta_0 = 0$

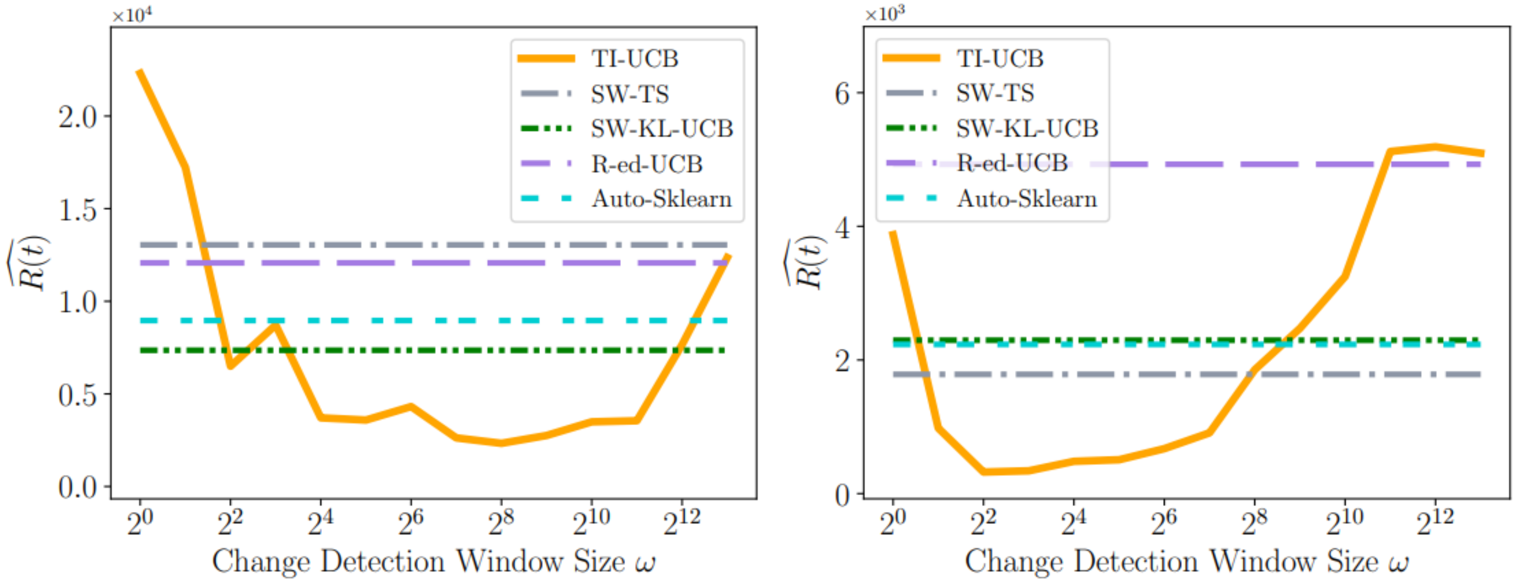


(a) LLM Bandits: Reward Functions **(b) LLM Bandits: Cumulative Regret**

Experiments

- Ablation on Change Detection Window Size

  We vary the sliding window size to test the sensitivity of TI-UCB performances to performance fluctuations.



Findings & Conclusion

- Capturing the increasing-then-converging performance trends, TI-UCB outperformed all baselines in online model selection.
- By integrating finetuning cost into reward design, TI-UCB promisingly balances cost and performance for practical deployment of LLMs.
- Customized change detection window sizes can flexibly tackle fluctuations in model performance during training.

Contact:
xiayuu@umich.edu

Paper: