Contextual Bandit for LLM Routing Problems

Siqi Yao

SCHOOL OF DATA SCIENCE

October 17, 2025

Table of Contents

Setting of Contextual Bandit

A contextual bandit algorithm $\mathbf{A}$ proceeds in discrete trials $t = 1, 2, 3 \cdots$ In trial t:

1. $\mathbf{A}$ observes the current user u_t, a set $\mathcal{A}_t$ of arms, and **context vector** $\mathbf{x}_{t,a}$ for $a \in \mathcal{A}_t$. $\mathbf{x}_{t,a}$ summarizes information of u_t and a, e.g., features of user and movie in a recommendation system.
2. Based on previous observations, $\mathbf{A}$ chooses an arm $a_t \in \mathcal{A}_t$ and receives payoff r_{t,a_t} whose expectation depends on both u_t and a_t.
3. $\mathbf{A}$ then improves its arm-selection strategy with the new observation $(\mathbf{x}_{t,a_t}, a_t, r_{t,a_t})$. No feedback is observed for unchosen arms.

The T-trial regret is defined by:

$$R_{\mathbf{A}}(T) := \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right] \tag{1}$$

where a_t^* is the arm with maximum expected payoff **at trial** t.

Core Assumption of LinUCB

- Assume the expected payoff of an arm a is **linear** in its d-dimensional feature $\mathbf{x}_{t,a}$ with unknown coefficient vector $\boldsymbol{\theta}_a^*$:

$$\mathbb{E}[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^\top \boldsymbol{\theta}_a^* \tag{2}$$

  This is called **Disjoint** Linear Model since the parameters are not shared among different arms.
- Let $\mathbf{D}_a \in \mathbb{R}^{m \times d}$ be the matrix of training data at trial t, and $\mathbf{c}_a \in \mathbb{R}^m$ be the corresponding target vector.
- Estimate $\boldsymbol{\theta}_a^*$ by applying ridge regression to the training data $(\mathbf{D}_a, \mathbf{c}_a)$:

$$\hat{\boldsymbol{\theta}}_a = \left(\mathbf{D}_a^\top \mathbf{D}_a + \mathbf{I}_d\right)^{-1} \mathbf{D}_a^\top \mathbf{c}_a \tag{3}$$

  where $\mathbf{I}_d$ is the $d \times d$ identity matrix.

Upper Confidence Bound

- Under certain conditions, with probability at least $1 - \delta$:

$$\left| \mathbf{x}_{t,a}^{\top} \hat{\boldsymbol{\theta}}_a - \mathbb{E}[r_{t,a} | \mathbf{x}_{t,a}] \right| \leq \alpha \sqrt{\mathbf{x}_{t,a}^{\top} (\mathbf{D}_a^{\top} \mathbf{D}_a + \mathbf{I}_d)^{-1} \mathbf{x}_{t,a}} \tag{4}$$

  for any $\delta > 0$ and $\mathbf{x}_{t,a} \in \mathbb{R}^d$, and $\alpha = 1 + \sqrt{\frac{\ln(2/\delta)}{2}}$.

- A UCB-type arm-selection strategy: at each trial t, choose

$$a_t := \arg\max_{a \in \mathcal{A}_t} \left(\mathbf{x}_{t,a}^{\top} \hat{\boldsymbol{\theta}}_a + \alpha \sqrt{\mathbf{x}_{t,a}^{\top} \mathbf{A}_a^{-1} \mathbf{x}_{t,a}} \right) \tag{5}$$

  where $\mathbf{A}_a := \mathbf{D}_a^{\top} \mathbf{D}_a + \mathbf{I}_d$.

LinUCB Algorithm (with Disjoint Linear Models)

Algorithm 1 LinUCB with disjoint linear models.

```
Inputs: $\alpha \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ do
    Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
    for all $a \in \mathcal{A}_t$ do
        if $a$ is new then
            $\mathbf{A}_a \leftarrow \mathbf{I}_d$ ($d$-dimensional identity matrix)
            $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ ($d$-dimensional zero vector)
        end if
        $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
        $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
    end for
    Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff $r_t$
    $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
    $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
end for
```

Figure 1: LinUCB Algorithm. Source: [Li et al., 2010]

Table of Contents

One Slide Summary

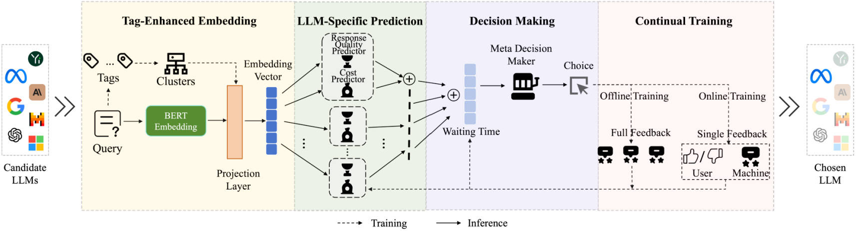

Figure 2: MixLLM Framework. Source: [Wang et al., 2025]

- Generate tag-enhanced query embedding.
- Predict response quality and cost for each LLM.
- Decision making based on response quality, cost, response time and **uncertainty**.
- Update predictor & decision maker through offline & online learning.

Table of Contents

Motivation

- General-purpose query embeddings (e.g., by BERT) contain too much noises and are not tailored for LLM routing.
- Different LLMs can be proficient in different domains (e.g., science, law).
- Solution: Enhance the encoder by introducing tag knowledge.

Procedure

1. Apply the **InsTag** model to generate fine-grained tags for each query and manually cluster the tags into a set of coarse grained domains D. E.g., "How to implement linked list in python" → ["Data Structure", "Programming"] → "Computer Science".
2. InsTag is too large to use during inference. Use the preprocessed data to fine-tune a BERT encoder. Total loss:

$$\mathcal{L} = \mathcal{L}_{\text{intra}} + \mathcal{L}_{\text{inter}} \tag{6}$$

where $\mathcal{L}_{\text{intra}}$ encourages embeddings of the same domain cluster to be close to their center:

$$\mathcal{L}_{\text{intra}} = -\frac{1}{|Q|} \sum_{i=1}^{|Q|} \log \left(\frac{\exp(\mathbf{e}_i \cdot \boldsymbol{\mu}_i)}{\sum_{j=1}^{|D|} \exp(\mathbf{e}_i \cdot \boldsymbol{\mu}_j)} \right) \tag{7}$$

while $\mathcal{L}_{\text{inter}}$ ensures that different domain centers are distinct:

$$\mathcal{L}_{\text{inter}} = \frac{1}{|D|} \sum_{j=1}^{|D|} \log \left(\sum_{k \neq j} \exp(\boldsymbol{\mu}_j \cdot \boldsymbol{\mu}_k) \right) \tag{8}$$

3. During inference stage, only the fine-tuned BERT encoder is applied.

Table of Contents

LLM-Specific Quality and Cost Prediction

- Given a query embedding, predict both the **response quality** and **financial cost** for each candidate LLM.
- Response Quality: Learn a LLM-specific model for each LLM, which estimates the response quality of the n-th query on the l-th LLM:

$$\hat{p}_{n,l} = f_l^{\mathrm{rq}}(\mathbf{e}_n; \boldsymbol{\theta}_l^{\mathrm{rq}}) \tag{9}$$

- Financial Cost: The total cost of the n-th query on the l-th LLM includes the known input cost and the predicted output cost:

$$\hat{c}_{n,l} = \underbrace{\mathrm{len}_{n,l}^{\mathrm{prm}} \cdot \mathrm{price}_l^{\mathrm{prm}}}_{\text{Input cost}} + \underbrace{\hat{\mathrm{len}}_{n,l}^{\mathrm{res}} \cdot \mathrm{price}_l^{\mathrm{res}}}_{\text{Output cost}} \tag{10}$$

where the response length $\hat{\mathrm{len}}_{n,l}^{\mathrm{res}}$ is predicted using:

$$\hat{\mathrm{len}}_{n,l}^{\mathrm{res}} = f_l^{\mathrm{rl}}(\mathbf{e}_n; \boldsymbol{\theta}_l^{\mathrm{rl}}) \tag{11}$$

Meta Decision Maker

- For the n-th query q_n, the final decision score for each candidate LLM is given by:

$$s_{n,l} = s_{n,l}^{\text{trade}} + \alpha \cdot s_{n,l}^{\text{unc}} - \beta \cdot s_l^{\text{pen}} \tag{12}$$

- $s_{n,l}^{\text{unc}}$ measures the uncertainty:

$$s_{n,l}^{\text{unc}} = \mathbf{e}_n^\top \cdot \mathbf{A}_l^{-1} \cdot \mathbf{e}_n \tag{13}$$

  where **the query embeddings $\mathbf{e}_n$ are viewed as contexts.**

- $s_{n,l}^{\text{trade}}$ trade-offs the predicted quality and cost:

$$s_{n,l}^{\text{trade}} = \frac{\lambda}{\lambda+1} \cdot \hat{p}_{n,l} - \frac{1}{\lambda+1} \cdot \hat{c}_{n,l} \tag{14}$$

- s_l^{pen} penalizes long waiting time (independent of q_n):

$$s_l^{\text{pen}} = e^{\gamma \cdot (w_l - \xi \cdot \tau)} \tag{15}$$

- Select the candidate with the highest score:

$$m_n^* = \arg\max_l \left(s_{n,l}\right) \tag{16}$$

Table of Contents

Offline Training

- Prior to deployment, we perform offline training using refined feedback from **all** candidate LLMs.
- Parameters $\boldsymbol{\theta}_l^{\text{rq}}$ for the response quality predictors are updated as:

$$\boldsymbol{\theta}_l^{\text{rq}} := \boldsymbol{\theta}_l^{\text{rq}} - \eta_1 \cdot \nabla_{\boldsymbol{\theta}_l^{\text{rq}}} \mathcal{L}(p_{n,l}, \hat{p}_{n,l}) \tag{17}$$

- Parameters $\boldsymbol{\theta}_l^{\text{rl}}$ for the response length predictors are updated as:

$$\boldsymbol{\theta}_l^{\text{rl}} := \boldsymbol{\theta}_l^{\text{rl}} - \eta_2 \cdot \nabla_{\boldsymbol{\theta}_l^{\text{rl}}} \mathcal{L}(\text{len}_{n,l}^{\text{res}}, \hat{\text{len}}_{n,l}^{\text{res}}) \tag{18}$$

- For the LLM selected in a round, the uncertainty matrices $\mathbf{A}_l$ are updated by query embeddings:

$$\mathbf{A}_l := \mathbf{A}_l + \mathbf{e}_n \cdot \mathbf{e}_n^\top \tag{19}$$

Online Training: Final Objective

- After deployment, the predictive models and uncertainty matrices are continuously updated using refined **single** feedback from the selected LLMs.
- Introduce a Dynamic Feedback Score $s_{n,l}^{\mathrm{df}}$ to capture the binary **user feedback**, which is predicted by a network:

$$\left[s_{n,1}^{\mathrm{df}}, s_{n,2}^{\mathrm{df}}, \ldots, s_{n,|M|}^{\mathrm{df}}\right] = f^{\mathrm{df}}\left(\mathbf{e}_n; \boldsymbol{\theta}^{\mathrm{df}}\right) \tag{20}$$

- The final score for each LLM is updated as:

$$s'_{n,l} = s_{n,l} + \kappa_{n,l} \cdot s_{n,l}^{\mathrm{df}} \tag{21}$$

where $\kappa_{n,l}$ is the confidence factor given by:

$$\kappa_{n,l} = \frac{1}{\mathrm{Var}_n\left[s_{n,l}^{\mathrm{df}}\right] + \varepsilon} \tag{22}$$

and candidate with highest score will be recommended:

$$m_n^* = \arg\max_l \left(s'_{n,l}\right) \tag{23}$$

Policy Gradient Method

- Trajectory $\tau := (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$. Total reward for the trajectory $R(\tau) := \sum_{t=0}^{T} r_t$, where r_t is the reward obtained from step t.
- Objective: Maximize expected total reward for all trajectories:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \sum_{\tau} P(\tau; \theta) R(\tau) \tag{24}$$

where $\pi_\theta := \pi_\theta(a|s)$ is the policy network. How to compute $\nabla_\theta J(\theta)$?

Policy Gradient Method

- Trajectory $\tau := (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$. Total reward for the trajectory $R(\tau) := \sum_{t=0}^{T} r_t$, where r_t is the reward obtained from step t.
- Objective: Maximize expected total reward for all trajectories:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \sum_{\tau} P(\tau; \theta) R(\tau) \tag{24}$$

where $\pi_\theta := \pi_\theta(a|s)$ is the policy network. How to compute $\nabla_\theta J(\theta)$?

- Log-Derivative trick: $\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)} \Rightarrow \nabla_x f(x) = f(x) \nabla_x \log f(x)$. Thus, we obtain:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_{\tau} P(\tau; \theta) R(\tau) && (25)\\
&= \sum_{\tau} (\nabla_\theta P(\tau; \theta)) R(\tau) && (26)\\
&= \sum_{\tau} P(\tau; \theta) \left(\nabla_\theta \log P(\tau; \theta)\right) R(\tau) && (27)\\
&= \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\nabla_\theta \log P(\tau; \theta)\right) R(\tau)\right] && (28)
\end{aligned}$$

Training f^{df} using User Feedback

- The probability of selecting candidate l (Policy) is:

$$\pi(l|\mathbf{e}_n;\boldsymbol{\theta}^{\text{df}}) = \frac{\exp(s_{n,l}^{\text{df}})}{\sum_{k=1}^{|M|}\exp(s_{n,k}^{\text{df}})} \tag{29}$$

- Maximize the expected reward:

$$J(\boldsymbol{\theta}^{\text{df}}) = \mathbb{E}_{l\sim\pi(\cdot|e_n;\boldsymbol{\theta}^{\text{df}})}[r_n] \tag{30}$$

- Apply (28), we obtain the parameter update rule:

$$\boldsymbol{\theta}^{\text{df}} := \boldsymbol{\theta}^{\text{df}} + \eta_3 \cdot \nabla_{\boldsymbol{\theta}^{\text{df}}} \log \pi(m_n^*|\mathbf{e}_n;\boldsymbol{\theta}^{\text{df}}) \cdot r_n \tag{31}$$

where m_n^* is the selected LLM, r_n is the human binary feedback, and the gradient can be calculated as:

$$\nabla_{\boldsymbol{\theta}^{\text{df}}} \log \pi(m_n^*|\mathbf{e}_n;\boldsymbol{\theta}^{\text{df}}) = \nabla_{\boldsymbol{\theta}^{\text{df}}}\left(s_{n,m_n^*}^{\text{df}} - \log\sum_{k=1}^{|M|}\exp(s_{n,k}^{\text{df}})\right) \tag{32}$$

Table of Contents

One Slide Summary

- Learn embeddings of queries & LLMs in a shared embedding space.
- Update LLM embeddings through online bandit feedback.
- Enforce budget constraint with online cost policy (omitted).

Table of Contents

Motivation

- Goal: Learn embeddings for queries and LLMs in a shared space, such that the cosine distance between a query and an LLM represents their mutual affinity.
- Abundant public data are available in the form of **human preferences**, where given a query and responses from two LLMs, humans provide their preferred LLM response.
- This section covers **offline** pretraining of the embedding space.

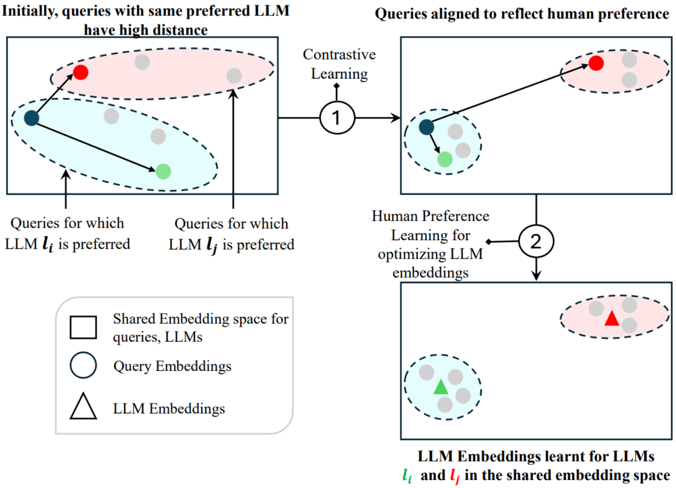

Figure 3: Pretraining with Human Preferences. Source: [Panda et al., 2025]

Learning the Query Projections

- Given an existing query embedding model $\phi : \mathcal{Q} \to \mathbb{R}^{d_e}$, define the projection to d_m-dimensional shared space as $\psi(q) = W\phi(q) + b$.
- $W \in \mathbb{R}^{d_m \times d_e}$ and $b \in \mathbb{R}^{d_m}$ are learned using a cosine distance-based **triplet loss** on human preference data D_{pref}.
- For each anchory query $(q_a, l_i, l_j, l_{\text{win}}) \in D_{\text{pref}}$ (l_{win} is the preferreed LLM), the positive pool is constructed as:

$$P = \{(q, l_i, l_j, l_w) \in D_{\text{pref}} | l_w = l_{\text{win}}\} \tag{33}$$

and the negative pool is constructed as:

$$N = \{(q, l_i, l_j, l_w) \in D_{\text{pref}} | l_w \neq l_{\text{win}} \wedge \text{size}(l_w) < \text{size}(l_{\text{win}})\} \tag{34}$$

Learning LLM Embeddings

- Next, freeze the query projection parameters and learn the LLM embeddings θ_i for each LLM l_i.
- **Goal:** Given a query $(q, l_i, l_j, l_w) \in D_{\text{pref}}$, the embedding of the preferred LLM l_w is close to $\psi(q)$.
- Define the probability of l_i winning l_j as

$$p_i = \frac{\exp(\cos(\theta_i, \psi(q)))}{\sum_{k \in \{i,j\}} \exp(\cos(\theta_k, \psi(q)))} \tag{35}$$

  train the LLM embeddings using the binary cross-entropy loss:

$$L(y, p_i) = -[y \cdot \log(p_i) + (1 - y) \cdot \log(1 - p_i)] \tag{36}$$

  where $y = 1$ if $l_w = l_i$, and $y = 0$ o.w.
- Denote the final learned embeddings for an LLM l_i as θ_i^{pref}.

Table of Contents

Settings

- Model **projected query embeddings $\psi(q_t)$ as contexts; LLM embeddings as bandit parameters,** which are updated during each round.
- Reward $r_t = s(q_t, y_t^l) \in [0, 1]$ is the response quality from the selected LLM l.
- Objective: maximize cumulative reward $\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$ via policy $\pi : \mathbb{R}^{d_m} \to \mathcal{A}$.

Bandit Updates

- Denote θ_a^t as embedding of LLM a at time t, and $\theta_a^0 = \theta_a^{\text{pref}}$.
- Expected reward at t:

$$\mathbb{E}[r_t|a, q_t] = \cos(\hat{\psi}(q_t), \hat{\theta}_a) = \hat{\psi}(q_t) \cdot \hat{\theta}_a \tag{37}$$

  where $\hat{\psi}(q_t) = \frac{\psi(q_t)}{\|\psi(q_t)\|_2}$ and $\hat{\theta}_a = \frac{\theta_a}{\|\theta_a\|_2}$. Reflects cosine similarity between embeddings.
- At each t, for each arm a, estimation of embedding is given by:

$$\widetilde{\theta}_a^t = (A_a^t)^{-1} b_a^t \tag{38}$$

  where $A_a^t = A_a^{t-1} + \hat{\psi}(q_t)\hat{\psi}(q_t)^\top, b_a^t = b_a^{t-1} + r_t\hat{\psi}(q_t)$. $\lambda_a > 0$ is a regularization parameter.
- Select arm a_t satisfying:

$$a_t = \arg\max_a \left(\cos(\hat{\psi}(q_t), \widetilde{\theta}_a^t) + \alpha\sqrt{\hat{\psi}(q_t)^\top (A_a^t)^{-1} \hat{\psi}(q_t)} \right) \tag{39}$$

Algorithm 1 PILOT (**P**reference-prior **I**nformed **L**inUCB f**O**r Adaptive Rou**T**ing)

Input: Human preference data D_{pref}, LLMs L

Preference-Based Pretraining

1: Learn query projection ϕ by minimizing triplet loss using $(q_a, l_i, l_j, l_{\text{win}})$ tuples from D_{pref} and constructing negative and positive samples as mentioned in Section 2.2.

2: Fix ϕ and learn LLM embeddings $\theta_{\text{LLM}}^{pref}$ using binary cross-entropy loss.

Online Bandit Learning

3: Initialize bandit learning parameters $A_a = \lambda_a I$ and $b_a = \lambda_a \theta_a^{pref}$ for all $a \in L$.

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

5:     Define a_t as arm/LLM with largest UCB.

6:     Observe feedback r_t w.r.t. response of selected LLM a_t & update parameters A_a & b_a.

7: **end for**

Figure 4: PILOT Algorithm. Source: [Panda et al., 2025]

Table of Contents

Why Context Matters & How to Model

- The paper demonstrated that later LLMs often perform better when they can "see" earlier attempts.
- Thus, let each LLM selection depend not only on the user query, but also on the **responses of previously selected models** $\Rightarrow$ evolving prompt context.
- Formalize the context evolution process as a (complicated) black-box function g.
- Complexity of next-prompt prediction makes multi-step planning algorithms intractable, but can be addressed by contextual bandits, which adopt a **myopic** view.

Settings

- Denote $[K] = \{1, \ldots, K\}$ the set of LLMs, and associate each LLM $k \in [K]$ an unknown feature vector $\theta_k^* \in \mathbb{R}^d$.
- An agent interacts with users over T rounds, in each round $t \in [T]$, the agent receives a query Q_t and engages in a multi-step interaction with the user, lasting at most H steps.
- Denote $x_{t,1} \in \mathbb{R}^d$ the initial context vector derived from Q_t. At each step $h \in [H]$, the agent observes context $x_{t,h}$ and selects an LLM $a_{t,h} \in [K]$ to generate a response. Denote $R_{t,h}$ the output of $a_{t,h}$ when invoked on $x_{t,h}$, and $r_{t,h} \in \{0, 1\}$ the binary user feedback.
- Model the feedback as:

$$r_{t,h} = \langle x_{t,h}, \theta_{a_{t,h}}^* \rangle + \varepsilon_{t,h} \tag{40}$$

where $\varepsilon_{t,h}$ is the zero-mean sub-Gaussian noise.

Greedy LinUCB Algorithm

Algorithm 1 Greedy LinUCB for Multi-LLM Selection

Input: Regularization parameter $\lambda > 0$, confidence parameter $\alpha > 0$
Initialize $A_k \leftarrow \lambda I_d$, $b_k \leftarrow \mathbf{0} \in \mathbb{R}^d$ for all $k \in [K]$
for round $t = 1, 2, \cdots$ **do**
  Receive initial query Q_t and build context $x_{t,1} \in \mathbb{R}^d$
  for step $h = 1$ to H **do**
    for each $k \in [K]$ **do**
      $\hat{\theta}_k \leftarrow A_k^{-1} b_k; \quad \mathrm{UCB}_k \leftarrow \langle x_{t,h}, \hat{\theta}_k \rangle + \alpha \cdot \sqrt{x_{t,h}^\top A_k^{-1} x_{t,h}}$
    end for
    Query LLM $a_{t,h} \leftarrow \arg\max_k \mathrm{UCB}_k$ with $(Q_t, R_{t,1}, \cdots, R_{t,h-1})$
    Receive output $R_{t,h}$ and binary feedback $r_{t,h} \in \{0, 1\}$
    Update: $A_{a_{t,h}} \leftarrow A_{a_{t,h}} + x_{t,h} x_{t,h}^\top; \quad b_{a_{t,h}} \leftarrow b_{a_{t,h}} + r_{t,h} x_{t,h}$
    if $r_{t,h} = 1$ **then**
      break
    else
      $x_{t,h+1} \leftarrow g(x_{t,h}, a_{t,h}, R_{t,h}, r_{t,h})$
    end if
  end for
end for

Figure 5: Greedy LinUCB Algorithm. Source: [Poon et al., 2025]

Examples of Blackbox Function g

Example 1: Simple Concatenation

- $x_{t,h}$: User: "Please explain what is AI."
- Response from LLM is too technical and the user is unsatisfied.
- User follow-up: "Can you use a simpler analogy?"
- Role of g: The system concatenates the user's follow-up with the previous conversation history to form a new, longer context.
- $x_{t,h+1}$: **User**: "Please explain what is AI." **Assistant**: [Professional response from LLM] **User**: "Can you use a simpler analogy?"

Example 2: Combination of User Edits and System Commands

- $x_{t,h}$: User provides a code snippet and says: "Optimize this Python code."
- LLM outputs an optimized version, but the user prefers better readability.
- User edits the original instruction to: "Optimize this Python code, **with a focus on improving readability**."
- Role of g: Here, g includes the user's direct editing behavior.
- $x_{t,h+1}$: Combination of the user's revised instruction and the original code snippet.

Possible Directions

- Replace linear model with neural nets, e.g., **NeuralUCB** [Zhou et al., 2020], **NeuralGCB** [Salgia, 2023].
- Running UCB over many LLMs is costly, and the best solution may be a sequence or ensemble, not a single model.
  - **Combinatorial Bandits:** Formulate the problem as selecting a subset of LLMs, where each subset acts as a "super arm." [Hwang et al., 2023, Atalar and Joe-Wong, 2025].
  - **Scalable Bandits with Hashing:** For single LLM selection with large K, [Jun et al., 2017] proposed hashing techniques to locate the best arm in sublinear time.
- **Rist Sensitive/Robust Bandit:** MaRaB [Galichet et al., 2013], CVaR bandits [Tamkin et al., 2019], heavy-tailed bandits [Bubeck et al., 2013, Tamás et al., 2024, Ye et al., 2025].

References I

Atalar, B. and Joe-Wong, C. (2025).
Neural combinatorial clustered bandits for recommendation systems.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15417–15426.

Bubeck, S., Cesa-Bianchi, N., and Lugosi, G. (2013).
Bandits with heavy tail.
IEEE Transactions on Information Theory, 59(11):7711–7717.

Galichet, N., Sebag, M., and Teytaud, O. (2013).
Exploration vs exploitation vs safety: Risk-aware multi-armed bandits.
In *Asian conference on machine learning*, pages 245–260. PMLR.

References II

Hwang, T., Chai, K., and Oh, M.-h. (2023).
Combinatorial neural bandits.
In *International Conference on Machine Learning*, pages 14203–14236. PMLR.

Jun, K.-S., Bhargava, A., Nowak, R., and Willett, R. (2017).
Scalable generalized linear bandits: Online computation and hashing.
Advances in Neural Information Processing Systems, 30.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010).
A contextual-bandit approach to personalized news article recommendation.
In *Proceedings of the 19th international conference on World wide web*, pages 661–670.

References III

Panda, P., Magazine, R., Devaguptapu, C., Takemori, S., and Sharma, V. (2025).
Adaptive llm routing under budget constraints.
arXiv preprint arXiv:2508.21141.

Poon, M., Dai, X., Liu, X., Kong, F., Lui, J., and Zuo, J. (2025).
Online multi-llm selection via contextual bandits under unstructured context evolution.
arXiv preprint arXiv:2506.17670.

Salgia, S. (2023).
Provably and practically efficient neural contextual bandits.
In *International Conference on Machine Learning*, pages 29800–29844. PMLR.

References IV

Tamás, A., Szentpéteri, S., and Csáji, B. C. (2024).
Data-driven upper confidence bounds with near-optimal regret for heavy-tailed bandits.
arXiv preprint arXiv:2406.05710.

Tamkin, A., Keramati, R., Dann, C., and Brunskill, E. (2019).
Distributionally-aware exploration for cvar bandits.
In *NeurIPS 2019 Workshop on Safety and Robustness on Decision Making.*

Wang, X., Liu, Y., Cheng, W., Zhao, X., Chen, Z., Yu, W., Fu, Y., and Chen, H. (2025).
Mixllm: Dynamic routing in mixed large language models.
arXiv preprint arXiv:2502.18482.

References V

Wikipedia contributors (2025).
Triplet loss — Wikipedia, The Free Encyclopedia.
https://en.wikipedia.org/wiki/Triplet_loss.

Ye, C., Jin, Y., Agarwal, A., and Zhang, T. (2025).
Catoni contextual bandits are robust to heavy-tailed rewards.
arXiv preprint arXiv:2502.02486.

Zhou, D., Li, L., and Gu, Q. (2020).
Neural contextual bandits with ucb-based exploration.
In *International conference on machine learning*, pages 11492–11502. PMLR.

Thank you!
Any questions?