Efficient Sequential Decision Making with Large Language Models

Dingyang Chen
University of South Carolina
dingyang@email.sc.edu

Qi Zhang
University of South Carolina
qz5@cse.sc.edu

Yinglun Zhu
University of California, Riverside
yzhu@ucr.edu

Abstract

This paper focuses on extending the success of large language models (LLMs) to sequential decision making. Existing efforts either (i) re-train or finetune LLMs for decision making, or (ii) design prompts for pretrained LLMs. The former approach suffers from the computational burden of gradient updates, and the latter approach does not show promising results. In this paper, we propose a new approach that leverages online model selection algorithms to efficiently incorporate LLMs agents into sequential decision making. Statistically, our approach significantly outperforms both traditional decision making algorithms and vanilla LLM agents. Computationally, our approach avoids the need for expensive gradient updates of LLMs, and throughout the decision making process, it requires only a small number of LLM calls. We conduct extensive experiments to verify the effectiveness of our proposed approach. As an example, on a large-scale Amazon dataset, our approach achieves more than a 6x performance gain over baselines while calling LLMs in only 1.5% of the time steps.

1 Introduction

Sequential decision making addresses the problem of adapting an agent to an unknown environment, where the agent learns through a feedback loop by repeatedly receiving contexts, selecting actions, and observing feedback. This approach has been widely applied in real-world scenarios, including recommendation systems (Li et al., 2010; Agarwal et al., 2016), healthcare (Tewari and Murphy, 2017; Svensson, 2023), and dialogue systems (Li et al., 2016). With the significant success of large language models (LLMs) in natural language processing (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023), an important next step is to extend this success to sequential decision making and enhance applications therein.

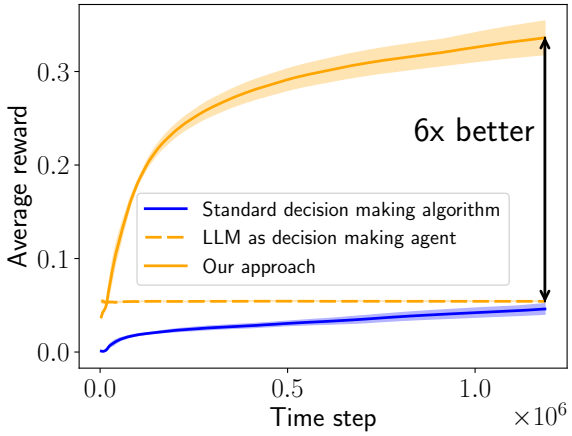

Figure 1: Performance comparison (higher is better) on the AmazonCat-13K dataset. The decision making task is to predict item tags based on textual descriptions. We compare three approaches: (i) a standard decision making algorithm, (ii) a pretrained LLM as decision making agent, and (iii) our approach that balances the above two methods. We defer further details to Section 4.

Existing efforts to leverage LLMs for sequential decision making focus on two directions: (i) viewing decision making as sequence modeling and re-training or finetuning large models to adapt them to unknown environments (Chen et al., 2021; Zheng et al., 2022; Reid et al., 2022; Sun et al., 2023; Raparthy et al., 2023; Lee et al., 2024), and (ii) utilizing prompt engineering and in-context learning to adapt pretrained large models to sequential decision making problems (Krishnamurthy et al., 2024). While the first approach usually achieves promising empirical results, it is hindered by the substantial computational burden associated with re-training or finetuning large models, which often contain hundreds of billions of parameters. The second approach (Krishnamurthy et al., 2024), on

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 9157–9170
November 12-16, 2024

the other hand, has demonstrated that most in-context learning and prompt engineering methods fail to effectively adapt LLMs to sequential decision making environments, except when employing the most advanced models (at the time), i.e., GPT-4 (Achiam et al., 2023), with sophisticated prompt designs.

In this paper, we propose a new approach to efficiently incorporate large pretrained models into sequential decision making environments, *without the need for expensive model re-training or finetuning*. We run experiments (see Fig. 1 and its caption for settings) on the `AmazonCat-13K` dataset (Bhatia et al., 2016) and observe that:

- Vanilla LLMs as decision making agents exhibit strong initial performance thanks to their significant commonsense knowledge and remarkable reasoning ability. However, LLM agents fail to show continuous improvements.
- Standard sequential decision making algorithms, while performing poorly initially, continuously learn to adapt to the environment and improve their performance over time.

To take advantage of both methods, we adapt online model selection algorithms (Auer et al., 2002; Agarwal et al., 2017; Pacchiano et al., 2020) to a framework that can automatically balance the performance of LLM-powered policies/agents and standard decision making algorithms. Initially, the framework relies more on LLM-powered policies to achieve good initial results. As standard decision making algorithms begin to adapt to the environments, it gradually shifts towards these algorithms. To our knowledge, this work presents the first result in leveraging online model selection algorithms to efficiently incorporate LLMs into sequential decision making. Our framework also offers several compelling advantages:

- **Statistical efficiency.** It achieves superior performance compared to vanilla LLM-powered policies and standard sequential decision making algorithms. As shown in Fig. 1, our approach achieves more than a 6x performance gain (0.336 vs. 0.054) compared to baselines.
- **Computational efficiency.** First, our approach does not require expensive re-training or finetuning of LLMs. Second, it can be implemented with a small number of LLMs over the decision making process. In our experiment, we show that it calls LLMs in only 1.5% of the time steps.
- **Plug-and-play compatibility.** Our framework can flexibly incorporate off-the-shelf pretrained LLMs in a plug-and-play manner. Furthermore, unlike existing methods that require advanced models such as GPT-4 (Krishnamurthy et al., 2024), our approach can leverage much smaller language models (e.g., a model with 80 million parameters) and achieve promising decision making results.

2 Problem Setting

We focus on contextual bandits, a key problem in sequential decision making that emphasizes the fundamental challenge of balancing exploration and exploitation (Lattimore and Szepesvári, 2020). In contextual bandits, a learner interacts with an unknown environment over $T \in \mathbb{N}^+$ rounds. At each round $t \in [T]$, the learner receives a context $x_t \in \mathcal{X}$ (the context space), selects an action $a_t \in \mathcal{A}$ (the action space), and then observes a bounded loss $\ell_t(a_t)$ (sampled from an unknown distribution), where $\ell_t : \mathcal{A} \to [0, 1]$ is the underlying loss function. Contextual bandits can be viewed as the simplest form of reinforcement learning where state transitions are abstracted away. Following the convention (Agarwal et al., 2012; Foster et al., 2018; Foster and Rakhlin, 2020), we assume that the learner has access to a function class $\mathcal{F} \subseteq (\mathcal{X} \times \mathcal{A} \to [0, 1])$ to approximate an unknown true loss function $f^\star(x, a) = \mathbb{E}[\ell_t \mid x_t = x, a_t = a]$. Let $\pi^\star(x) = \arg\min_a f^\star(x, a)$ denote the optimal policy with respect to the true expected loss (i.e., always selecting an action that achieves the smallest expected loss). The learner's goal is to choose a policy $\pi = (\pi_1, \cdots, \pi_T)$ to minimize the cumulative regret, which is defined as $\mathbf{Reg}(T) := \sum_{t=1}^{T} f^\star(x_t, \pi_t(x_t)) - f^\star(x_t, \pi^\star(x_t))$.

We focus on the setting where the context space and the action space are subspaces of the language space, i.e., the learner interacts with an environment through textual contexts and actions, and actions that induce low loses are usually consistent with commonsense knowledge and/or reasoning.[1]

[1] One can also prompt LLMs with numerical representations to get regression-style predictions (Garg et al., 2022).

Therefore, our setting motivates leveraging pretrained large language models (LLMs) into contextual bandits. Specifically, we consider a pretrained LLM: prompt $p \mapsto$ output o, that maps a prompt p to a textual response o (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023). Since LLMs are pretrained to acquire general knowledge about the world, we expect the output $o_t \sim \text{LLM}(p = x_t)$ of LLMs, when prompting LLMs with the context x_t (and other relevant information), would provide informative guide for the decision making process.

Additional notation. For an integer $n \in \mathbb{N}$, we let $[n]$ denote the set $\{1, \ldots, n\}$. For a finite set $\mathcal{Z}$, we let $\text{unif}(\mathcal{Z})$ denote the uniform distribution over all the elements in $\mathcal{Z}$. We use $e_i \in \mathbb{R}^d$ to denote the i-th canonical vector in $\mathbb{R}^d$, i.e., its i-th entry is 1 and the rest entries are 0.

3 Methods

We present our approach for efficiently incorporating LLMs into contextual bandits/decision making in this section. We provide the algorithmic foundation in Section 3.1 and various sampling strategies in Section 3.2.

3.1 Efficient Decision Making with LLMs

At a high level, our framework utilizes an online model selection algorithm to adaptively balance the performance of two sets of base algorithms: (i) standard contextual bandit algorithms, and (ii) policies constructed based on off-the-shelf pretrained LLMs. Our framework achieves the best-of-both-worlds by (i) efficiently extracting knowledge stored in pretrained LLMs and (ii) leveraging the long-term learning ability of standard contextual bandit algorithms. We construct LLM-powered policies in Section 3.1.1 and introduce the algorithmic framework in Section 3.1.2.

3.1.1 LLMs as Decision Making Agents

Since the outputs of LLMs are in the general language space that may not align with any action in the action set, we first provide an algorithm to convert pretrained LLMs to decision making agents.

Algorithm 1 is designed to be compatible with flexible choices of LLMs, embedding models, and similarity measures. It prompts the LLM with context x to obtain top-k most likely outputs o_i and together with their likelihood q_i: $\{(o_1, q_1), \cdots, (o_k, q_k)\}$.[2] For each embedded output $g(o_i)$, it then measures its similarity between each of the embedded action $\{g(a), a \in \mathcal{A}\}$, and find the one a_i with the highest similarity. Finally, we construct policy π^{LLM} by mapping x into the (multi) set $\{a_1, \cdots, a_k\}$ with weighted probability, i.e., $\mathbb{P}(\pi^{\text{LLM}}(x) = a_i) = q_i / \sum_{j=1}^k q_j$. The LLM-powered policy uses the same policy π^{LLM} for the entire decision making process to avoid expensive re-training/finetuning of LLMs.

Algorithm 1 Construct LLM-Powered Policies

Input: Context x, pretrained LLM, embedding model g : language $\to \mathbb{R}^d$, similarity measure $\texttt{Sim} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and hyperparameter $k \in \mathbb{N}^+$.

1: Prompt LLM with context x to obtain top-k most likely outputs o_i and likelihood q_i: $\{(o_1, q_1), \cdots, (o_k, q_k)\}$.
2: Embed all actions $\{g(a) : a \in \mathcal{A}\} \subseteq \mathbb{R}^d$ and LLM outputs $\{g(o_i) : i \in [k]\} \subseteq \mathbb{R}^d$.
3: Get $a_i := \arg\max_{a \in \mathcal{A}} \texttt{Sim}(g(o_i), g(a))$ for each $i \in [k]$.
4: Construct π^{LLM} such that $\mathbb{P}(\pi^{\text{LLM}}(x) = a_i) = q_i / \sum_{j=1}^k q_j$.

3.1.2 Algorithmic Framework

In Algorithm 2, we present our framework to efficiently incorporate LLMs into contextual bandits. Algorithm 2 leverages online model (expert) selection algorithms (Auer et al., 2002; Agarwal et al., 2017; Pacchiano et al., 2020) to adaptively balance standard contextual bandit algorithms and LLM-powered policies. Compared to existing online model selection algorithms, Algorithm 2 additionally (i) incorporates Algorithm 1 to convert LLMs into policies, and (ii) allows more flexible sampling strategies to control the number of LLM calls (see Section 3.2 for detailed discussion). At a high-level, the sampling probability in Algorithm 2 is designed to rely more on the set of LLM-powered policies at the beginning, and then gradually transit to put more probability on standard contextual bandit algorithms. By doing so, we aim to simultaneously achieve the following two objectives:

[2] Additional instructions or prior interaction history can also be incorporated into the prompt design. When $k = 1$, we only need to obtain the LLM output o, without computing the likelihood q.

Algorithm 2 Efficient Decision Making with LLMs

Input: A set of contextual bandit algorithms $\{\pi^{\mathsf{CB}_1}, \cdots, \pi^{\mathsf{CB}_{M_1}}\}$, and a set of LLMs $\{\mathsf{LLM}_{M_1+1}, \cdots, \mathsf{LLM}_M\}$.

1: Convert LLMs to $\{\pi^{\mathsf{LLM}_{M_1+1}}, \cdots, \pi^{\mathsf{LLM}_M}\}$ using Algorithm 1.
2: Order all policies as $\{\pi^i\}_{i=1}^M$. Initialize sampling strategy $p_1 = \text{unif}[M]$.
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Receive contaxt x_t.
5: Sample $i_t \sim p_t$.
6: Follow π^{i_t} to play action a_t and observe loss $\ell_t(a_t)$.
7: Update contextual bandit algorithms with $(x_t, a_t, \ell_t(a_t))$.
8: Update sampling strategy $p_{t+1} \leftarrow p_t$.
`//We discuss detailed sampling strategies updates in Section 3.2.`

- **Leveraging knowledge in LLMs.** At the beginning stage, we leverage LLMs to select more informative data to warm start the learning process, and help contextual bandit algorithms learn better.

- **Long-term adaptation to environments.** In the later stage, we leverage the long-term learning ability of contextual bandit algorithms to minimize losses in the long run.

3.2 Sampling Strategies

In this section, we discuss in detail how to update the sampling strategy in Algorithm 2 (line 8). We present simple, pre-determined sampling strategies in Section 3.2.1 and learning-based sampling strategies in Section 3.2.2.

3.2.1 Simple Pre-Determined Sampling Strategies

We provide several simple, pre-determined sampling strategies in this section. They are simple and can be implemented without additional computation overhead. They follow the basic idea of putting more probability on LLM-powered policies at the beginning and gradually transiting probability to standard contextual bandit algorithms. We use p_t^{LLM} to denote the total probability of sampling LLM-powered policies, and use $p_t^{\mathsf{CB}} := 1 - p_t^{\mathsf{LLM}}$ to denote the total probability of sampling standard contextual bandit algorithms. In the following, we focus primarily on updating p_t^{LLM} (and thus p_t^{CB}).[3] We set $0 \le p_{\min} \le p_{\max} \le 1$ as user-specified lower and upper bound on p_t^{LLM}.

- **Polynomial decay.** Let C_{poly} and α be two hyperparameters. We set $p_t^{\mathsf{LLM}} := \min\{p_{\max}, \max\{p_{\min}, C_{\mathsf{poly}}/t^\alpha\}\}$.

- **Exponential decay.** Let C_{exp} and β be two hyperparameters. We set $p_t^{\mathsf{LLM}} := \min\{p_{\max}, \max\{p_{\min}, C_{\mathsf{exp}} \exp(-\beta t)\}\}$.

Number of LLM calls. For these simple sampling strategies, it's easy to see the expected number of LLM calls equals to $\sum_{t=1}^T p_t^{\mathsf{LLM}}$. One can also easily tune hyperparameters to control the number of LLM calls.

3.2.2 Learning-Based Sampling Strategies

While there exist many other learning-based sampling strategies, we primarily use log-barrier online mirror descent (OMD), also known as the CORRAL update (Agarwal et al., 2017), to update the sampling probability with respect to importance-weighted losses incurred by base algorithms.

Algorithm 3 Log-Barrier-OMD Update (Agarwal et al., 2017)

Input: Learning rate $\eta > 0$, previous distribution p_t, selected base algorithm index i_t, and the incurred loss $\ell_t(a_t)$.

1: Construct an importance-weighted loss vector $\bar{\ell}_t := \frac{\ell_t(a_t)}{p_{t,i_t}} e_{i_t} \in \mathbb{R}^M$.
2: Find a constant $\lambda \in [\min_i \bar{\ell}_{t,i}, \max_i \bar{\ell}_{t,i}]$ such that $\sum_{i=1}^M \frac{1}{\frac{1}{p_{t,i}} + \eta(\bar{\ell}_{t,i} - \lambda)} = 1$.
3: Return an updated distribution p_{t+1} such that $\frac{1}{p_{t+1,i}} = \frac{1}{p_{t,i}} + \eta(\bar{\ell}_{t,i} - \lambda)$.

Algorithm 3 takes as input an initial learning rate $\eta > 0$, previous sampling distribution p_t, the index i_t of selected base algorithm, and the incurred loss $\ell_t(a_t)$. Algorithm 3 first constructs the standard importance-weighted unbiased loss estimator for

[3] One can apply simple strategies (e.g., uniform allocation) to allocate p_t^{LLM} (and p_t^{CB}) to individual policies.

all base algorithms (line 1), and then follow log-barrier online mirror descent to update the sampling distribution with respect to the losses (line 3). The update requires a normalization constant λ (line 2), which can be approximated with numerical root-finding algorithms such as the Brent's method (Zhang, 2011).

We sample from a smoothed version $\bar{p}_t$ of the sampling distribution p_t to help contextual bandit base algorithms explore at the beginning stage. Specifically, we clip the (total) sampling probability on LLMs p_t^{LLM} to $1 - p_{\min}$ if the (total) sampling probability on contextual bandits p_t^{CB} falls below $p_{\min}$, a user-specified hyperparameter.

Number of LLM calls. To control the number of LLM calls, we can either early stop sampling from LLM-powered policies in Algorithm 2 once the budget B is used up, or further modify the sampling strategy as

$$\check{p}_t^{\mathsf{LLM}} := \bar{p}_t^{\mathsf{LLM}} \cdot \left(\frac{B - N_t}{B}\right), \check{p}_t^{\mathsf{CB}} := 1 - \check{p}_t^{\mathsf{LLM}}, \tag{1}$$

where N_t represents the number of LLM calls used up to time step t. Both approaches limit the number of LLM calls to at most B.

4 Empirical Results

We conduct extensive experiments to examine the effectiveness of our proposed framework. We present experimental setups in Section 4.1, our main results in Section 4.2, and ablation study in Section 4.3. We defer additional experimental details to Appendix A. Code to reproduce all results is available at `https://github.com/dchen48/DMwithLLM`.

4.1 Experimental Setups

Datasets. We conduct experiments on two textual contextual bandit datasets, whose details are summarized in Table 1. `OneShotWikiLinks-311` (Singh et al., 2012; Vasnetsov, 2018) is a named-entity recognition task where contexts are text phrases preceding and following the mention text, and actions are text phrases corresponding to the concept names. `AmazonCat-13K` (Bhatia et al., 2016) is an extreme multi-label dataset whose contexts are text phrases corresponding to the title and content of an item, and actions are integers corresponding to item tags. We construct binary loss for each dataset, where selecting the correct actions leads to a loss of 0, and incorrect actions results in a loss of 1. In our experiments, we process data in batches with a batch size of 32.

Table 1: Datasets used for experiments.

Dataset	T	$\lvert\mathcal{A}\rvert$
`OneShotWikiLinks-311`	622000	311
`AmazonCat-13K`	1186239	13330

Baselines. We use SpannerGreedy (Zhu et al., 2022a) as the contextual bandit baseline, which is an efficient algorithm for textual decision making. We use Algorithm 1 with $k = 1$ to construct the LLM-powered policy baselines. We consider various LLM backbones, including Flan-T5 (Chung et al., 2024), with sizes small (80M parameters), base (250M parameters) and large (780M parameters), and more recent models Gemma 2B (instruct) (Team et al., 2024) and GPT-4o-mini (OpenAI, 2024a).

We implement our Algorithm 2 by combining the two types of baselines mentioned above. In most of our experiments, we select LLM backbones from the Flan-T5 model series;[4] we run additional experiments with Gemma 2B and GPT-4o-mini to verify the efficacy of our method when implemented with more advanced models. Unless otherwise noted, we implement Algorithm 2 using Algorithm 3 and a smoothing parameter $p_{\min} = 0.2$.

Evaluation metrics. We evaluate algorithms in terms of both statistical and computational performances. Statistically, following the convention in contextual bandits, we measure the performance in terms of the (average) reward, where one can easily convert loss into reward $r_t(a_t) := 1 - \ell_t(a_t)$. Computationally, since models used in contextual bandit algorithms are relatively lightweight (we empirically verify this in Section 4.2), we measure the performance in terms of the number of LLM

[4] Our goal is not to examine the most advanced LLMs or contextual bandit algorithms. Instead, we aim to verify that Algorithm 2 can effectively balance contextual bandit algorithms and LLMs policies, and outperform both of them when applied individually.

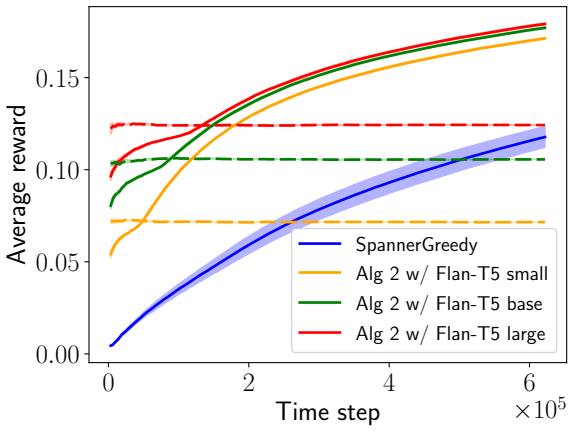

Figure 2: Comparison of average reward on the `OneShotWikiLinks-311` dataset (higher is better). Our Algorithm 2 is implemented with various sizes of Flan-T5 model. The dashed lines represent the performance of directly applying LLM-powered policy $\pi^{\text{Flan-T5}}$ (Algorithm 1) of corresponding sizes.

calls. Our results are averaged over 5 random runs; shaded area in figures represents the standard error of the mean.

4.2 Main Results

Statistical efficiency. Fig. 2 compares average reward achieved by various algorithms on the `OneShotWikiLinks-311` dataset. Our Algorithm 2 significantly outperforms other baselines: it achieves reward no smaller than 0.17131 no matter which Flan-T5 model is used; on the contrary, even with the Flan-T5 large, LLM-powered policy $\pi^{\text{Flan-T5}}$ only achieves reward 0.12423 and the contextual bandit algorithm SpannerGreedy only achieves reward 0.11773. The fact the Algorithm 2 with Flan-T5 small (yellow solid line) greatly outperforms $\pi^{\text{Flan-T5-large}}$ (red dashed line) shows the benefits of our algorithmic design. Note Flan-T5 small is nearly 10x smaller in parameter count compared to Flan-T5 large.

Computational efficiency. To examine the computational efficiency, we first run experiments to compare the cost of π^{LLM} selection versus the cost of contextual bandit selection, in terms of the execution time. As shown in Table 2, all π^{LLM} selections are considerably more expensive (from 52x to 159x more execution time) compared to contextual bandit selection.

Table 3 presents the fraction of LLMs calls in Algorithm 2 over the decision making process. Algorithm 2 not only achieves higher reward (Fig. 2), but also only calls LLMs in a small fraction (from 6% to 14%) of time steps. For comparison, directly applying $\pi^{\text{Flan-T5}}$ calls LLM at every time step.

Table 2: Cost ratio of π^{LLM} selection and contextual bandit selection, measure as the execution time of Flan-T5 models divided by the execution time of SpannerGreedy.

Small (80M)	Base (250M)	Large (780M)
52.16	79.49	159.20

Table 3: Fraction of LLM calls in Algorithm 2 over the decision making process with Flan-T5 models and on the `OneShotWikiLinks-311` dataset.

Small (80M)	Base (250M)	Large (780M)
0.06177	0.10033	0.14381

To further improve computational efficiency, we apply Eq. (1) or early stopping to limit the number of LLM calls of our algorithm, and show results in Table 4. Our results show that Algorithm 2 achieves slightly worse reward when limited to a smaller number of LLM calls. However, Algorithm 2 still outperform both baselines with an upper bound $B = 10000$ on the number of LLM calls, which is around 9x smaller compared to the number of LLM calls used in the unconstrained version of Algorithm 2.

Large-scale exhibition. We conduct a large-scale experiment on the `AmazonCat-13K` dataset that has more than 13k actions (around 42x larger than the `OneShotWikiLinks-311` dataset). With Flan-T5 small model, as shown in Fig. 1, our Algorithm 2 achieves more than a 6x performance gain over baselines: our algorithm achieves reward 0.33603, yet both SpannerGreedy and $\pi^{\text{Flan-T5}}$ achieves reward below 0.05424. Algorithm 2 calls LLMs in only 1.5% of the time steps (17783.4 LLM calls on average over horizon 1186239).

Learning with more advanced LLMs. We run additional experiments on the `AmazonCat-13K` dataset with more advanced LLMs: Gemma 2B and GPT-4o-mini. We show the results in Fig. 3. When using Gemma 2B as the LLM backbone, compared to baselines, our Algorithm 2 achieves

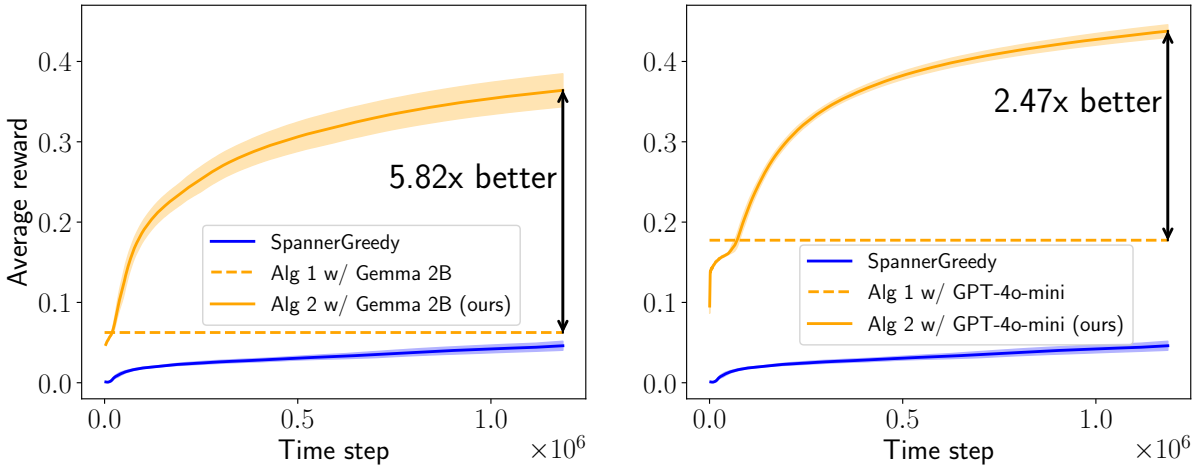

Figure 3: Performance comparison (higher is better) on the `AmazonCat-13K` dataset. Algorithm 2 incorporates SpannerGreedy and Algorithm 1 as base algorithms. *Left:* Use Gemma 2B as the LLM backbone. *Right:* Use GPT-4o-mini as the LLM backbone.

Table 4: Limit the number of LLM calls in Algorithm 2. Experiments conducted with the Flan-T5 large model and on the `OneShotWikiLinks-311` dataset.

Algorithms	# LLM calls	Reward
SpannerGreedy	N/A	0.11773
$\pi^{\text{Flan-T5-large}}$	622000	0.12423
Algorithm 2	89448.6	0.17913
Algorithm 2 w/ Eq. (1)	# LLM calls	Reward
$B = 10K$	7669.2	0.16836
$B = 20K$	12084.4	0.17309
Algorithm 2 w/ early stopping	# LLM calls	Reward
$B = 10K$	10000	0.16774
$B = 20K$	20000	0.17508

a 5.82x performance gain and calls Gemma 2B in only 1.62% of the time steps. When using GPT-4o-mini as the LLM backbone, compared to baselines, our Algorithm 2 achieves a 2.47x performance gain and calls GPT-4o-mini in only 4.49% of the time steps.[5] These results show that our Algorithm 2 not only works well with Flan-T5 models but also with more advanced models such as Gemma 2B and GPT-4o-mini, highlighting the broad compatibility of our algorithmic design.

[5]Due to computational constraints, we calculate the performance of Algorithm 1 with Gemma 2B or GPT-4o-mini as the average performance over the first 96000 time steps (the first 3000 data batches with a batch size 32). These averaged performances should be fairly accurate, as demonstrated by the real-time average performance of the Flan-T5 small model in Fig. 1, which appears to follow a nearly straight line.

4.3 Ablation Study

Probability updating strategies. We examine the performance of various probability updating strategies introduced in Section 3.2. Beyond the log-barrier OMD update, we also include simple pre-determined updating strategies: polynomial decay and exponential decay (we set $p_{\min} = 0$ and $p_{\max} = 0.8$). For polynomial decay, we set $\alpha = 1$ and select a C_{poly} from set $\{1, 10, 100\}$ that achieves the highest reward. For exponential decay, we select $\beta \in \{0.1, 0.01\}$ and $C_{\text{exp}} \in \{1, 10, 100\}$ jointly that achieves the highest reward. Table 5 shows the results of various probability updating strategies: while log-barrier OMD achieves better reward, pre-determined updating strategies generally leads to a smaller number of LLM calls.

Table 5: Comparison of different probability updating strategies. Experiments conducted with the Flan-T5 large model and on the `OneShotWikiLinks-311` dataset. We record the final average reward.

Methods	# LLM calls	Reward
Polynomial decay	19419.8	0.17413
Exponential decay	14943.6	0.17259
Log-barrier OMD	89448.6	0.17913

Smoothing strategy for Algorithm 3. In Algorithm 3, we adopt the smoothing strategy that clip the (total) sampling probability on LLMs p_t^{LLM} to $1 - p_{\min}$ if the (total) sampling probability on contextual bandit algorithms p_t^{CB} falls below $p_{\min}$. By doing this, we help contextual bandit base algorithms within Algorithm 2 better adapt to the environment, especially at the beginning stage. We compare our clipping-type smoothing strategy with the mixing-type smoothing strategy proposed in Agarwal et al. (2017): given a smooth parameter γ, set $\bar{p}_t := (1-\gamma) \cdot p_t + \gamma \cdot \text{unif}[M]$. We present the results in Table 6. Our result indicates that smoothing Algorithm 3 is important and our clipping strategy work betters than the mixing strategy.

Table 6: Comparison of different smoothing strategies for Algorithm 3. Experiments conducted with the Flan-T5 large model and on the `OneShotWikiLinks-311` dataset.

Methods	# LLM calls	Reward
No smoothing	618551.4	0.12386
Clipping (ours)	# LLM calls	Reward
$p_{\min} = 0.1$	144201.2	0.17532
$p_{\min} = 0.2$	89448.6	0.17913
Mixing	# LLM calls	Reward
$\gamma = 0.05$	151608.0	0.17449
$\gamma = 0.1$	149728.6	0.17288
$\gamma = 0.2$	189486.0	0.16691
$\gamma = 0.4$	248214.4	0.15862

5 Analyses

LLMs empower contextual bandit algorithms. As shown in Fig. 2, Algorithm 2 consistently outperforms its base algorithms. Since the LLM backbones in LLM-powered policies are never updated (for efficiency reasons), we hypothesizes that our Algorithm 2 empowers its bandit base algorithms with the help of LLMs.

To test this hypothesis, we first plot the real-time probability p_t^{CB} of Algorithm 2 sampling its contextual bandit base algorithm (Fig. 4, left). Since p_t^{CB} quickly increases its value to (around) 1 after the initial learning stage, we know that the contextual bandit base algorithm within Algorithm 2 plays an important role after the initial stage. We then plot the hypothetical performance of the contextual bandit base algorithm within Algorithm 2 (as if it were played at every time step). As shown in Fig. 4 (right), the contextual bandit base algorithm within Algorithm 2 (solid black line) achieves much better performance compared to the stand-alone contextual bandit algorithm (0.17546 vs. 0.11773). Since the main difference lies in the incorporation of data selected by LLM-powered policy, this shows that LLM selected data helps contextual bandit algorithm learn better.

We also draw the hypothetical performance of SpannerGreedy learned with purely LLM selected data (solid purple line in Fig. 4 right), which is worse than SpannerGreedy (0.06669 vs. 0.11773). This suggests that exploration in contextual bandit algorithm is also important and cannot be replaced with LLM selected data.

Algorithm 2 with multiple LLMs. We run Algorithm 2 with two LLMs: Flan-T5 large and Flan-T5 small. We compare this approach to Algorithm 2 with either Flan-T5 large or Flan-T5 small. We use N_S and N_L to denote the number of Flan-T5 large and Flan-T5 small calls, respectively, and show the results in Table 7. Compared to learning with a large model, learning with both large and small models achieves slightly worse reward,[6] but also uses a slightly smaller number of large model calls. Algorithm 2 relies more on the large model (89224 calls) instead of the small model (5833.4 calls on average), as it is designed to automatically adapt to better base policies.

Table 7: Algorithm 2 with multiple LLMs. Experiments conducted on the `OneShotWikiLinks-311` dataset.

Flan-T5 models	N_S	N_L	Reward
large + small	5833.4	89224.0	0.17813
large	N/A	89448.6	0.17913
small	38424.0	N/A	0.17131

6 Related Work

Sequential decision making. Sequential decision making is rooted in rich theoretical founda-

[6] This may be due to the fact that balancing over more models creates larger learning overheads.

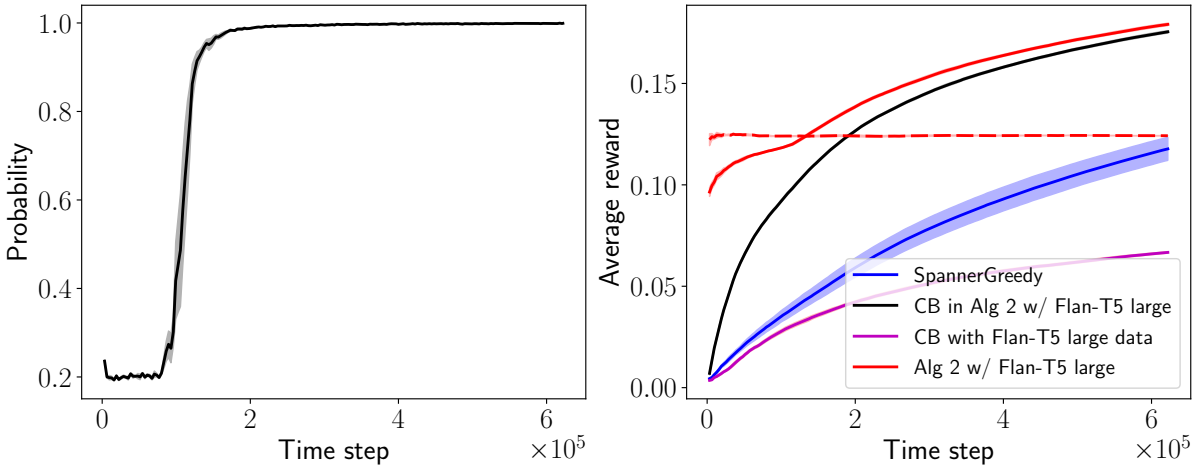

Figure 4: Experiments with the Flan-T5 large model and on the `OneShotWikiLinks-311` dataset. *Left:* Real-time probability p_t^{CB} of sampling contextual bandit base algorithm in Algorithm 2. *Right:* Hypothetical performance of the contextual bandit base algorithm within Algorithm 2 (black solid line) and hypothetical performance of the contextual bandit algorithm learned with purely LLM selected data (solid purple line).

tions (Langford and Zhang, 2007; Agarwal et al., 2014; Foster et al., 2021), and there is a long line of work that develop efficient decision making algorithms with general function approximation (Agarwal et al., 2012; Foster et al., 2018; Foster and Rakhlin, 2020; Simchi-Levi and Xu, 2021; Zhu et al., 2022a; Zhu and Mineiro, 2022; Rucker et al., 2023; Zhang et al., 2024); in our experiments, we include one such algorithm to textual environments. Another line of work focus on developing online model selection algorithms to balance the performance of base algorithms (Auer et al., 2002; Agarwal et al., 2017; Pacchiano et al., 2020; Zhu and Nowak, 2020, 2022; Marinov and Zimmert, 2021; Zhu et al., 2022b; Dann et al., 2024). Compared to previous online model selection approaches, we further incorporate LLMs into the decision making process.

LLMs for decision making. While there have been many studies that leverage LLMs into supervised learning (Xie et al., 2021; Garg et al., 2022; Akyürek et al., 2022), the understanding of how to leverage LLMs into sequential decision making is less developed. There exist two main approaches: (i) view decision making as sequence modeling and pretrain/finetune large models to adapt them to unknown environments (Chen et al., 2021; Zheng et al., 2022; Reid et al., 2022; Sun et al., 2023; Raparthy et al., 2023; Lee et al., 2024), and (ii) leverage prompt engineering and in-context learning to adapt pretrained large models to sequential decision making problems (Krishnamurthy et al., 2024). In this paper, we propose a new approach that efficiently incorporates LLMs into sequential decision making, addressing drawbacks of previous approaches.

7 Conclusion

In this paper, we study the problem of how to efficiently incorporate large language models into contextual bandits, a key problem in sequential decision making that emphasizes the fundamental challenge of balancing exploration and exploitation. We propose to use online model selection algorithms to adaptively balance LLMs agents and standard contextual bandit algorithms. Statistically, our approach greatly outperforms stand-lone LLM-powered policies and contextual bandit algorithms. Computationally, our approach avoids the need for expensive re-training or finetuning, and utilizes only a small fraction of LLM calls throughout the decision making process. Our framework is highly flexible, allowing for the integration of various off-the-shelf pretrained LLMs. In our experiments, it delivers promising results even when using a language model with only 80 million parameters.

8 Limitations

Our current approach primarily addresses contextual bandit problems, a specific case of reinforcement learning that lacks state transitions. Although it is possible to abstract away state transitions, treating an episode of reinforcement learning as a single step in a contextual bandit and applying our algorithms, we believe that more fine-grained treatments are necessary to achieve better performance in reinforcement learning. Moving forward, we plan to extend our algorithms and analyses to general reinforcement learning problems.

9 Acknowledgement

Dingyang Chen and Qi Zhang acknowledge funding support from NSF award 2154904.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alekh Agarwal, Sarah Bird, Markus Cozowicz, Luong Hoang, John Langford, Stephen Lee, Jiaji Li, Dan Melamed, Gal Oshri, Oswaldo Ribas, et al. 2016. Making contextual decisions with low technical debt. *arXiv preprint arXiv:1606.03966*.

Alekh Agarwal, Miroslav Dudík, Satyen Kale, John Langford, and Robert Schapire. 2012. Contextual bandit learning with predictable rewards. In *Artificial Intelligence and Statistics*, pages 19–26. PMLR.

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646. PMLR.

Alekh Agarwal, Haipeng Luo, Behnam Neyshabur, and Robert E Schapire. 2017. Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38. PMLR.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multi-armed bandit problem. *SIAM journal on computing*, 32(1):48–77.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Chris Dann, Claudio Gentile, and Aldo Pacchiano. 2024. Data-driven online model selection with regret guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 1531–1539. PMLR.

Dylan Foster, Alekh Agarwal, Miroslav Dudik, Haipeng Luo, and Robert Schapire. 2018. Practical contextual bandits with regression oracles. In *International Conference on Machine Learning*, pages 1539–1548. PMLR.

Dylan Foster and Alexander Rakhlin. 2020. Beyond UCB: Optimal and efficient contextual bandits with regression oracles. In *International Conference on Machine Learning*, pages 3199–3210. PMLR.

Dylan J Foster, Sham M Kakade, Jian Qian, and Alexander Rakhlin. 2021. The statistical complexity of interactive decision making. *arXiv preprint arXiv:2112.13487*.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598.

Google. 2024. Google Gemma-2b-instruct. https://huggingface.co/google/gemma-2b-it.

Akshay Krishnamurthy, Keegan Harris, Dylan J Foster, Cyril Zhang, and Aleksandrs Slivkins. 2024. Can large language models explore in-context? *arXiv preprint arXiv:2403.15371*.

John Langford and Tong Zhang. 2007. The epoch-greedy algorithm for multi-armed bandits with side information. *Advances in neural information processing systems*, 20.

Tor Lattimore and Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2024. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670.

Teodor Vanislavov Marinov and Julian Zimmert. 2021. The pareto frontier of model selection for general contextual bandits. *Advances in Neural Information Processing Systems*, 34:17956–17967.

OpenAI. 2024a. Gpt-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. 2024b. Openai developer quickstart tutorial. https://platform.openai.com/docs/quickstart.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Aldo Pacchiano, My Phan, Yasin Abbasi Yadkori, Anup Rao, Julian Zimmert, Tor Lattimore, and Csaba Szepesvari. 2020. Model selection in contextual stochastic bandit problems. *Advances in Neural Information Processing Systems*, 33:10328–10337.

Sharath Chandra Raparthy, Eric Hambro, Robert Kirk, Mikael Henaff, and Roberta Raileanu. 2023. Generalization to new sequential decision making tasks with in-context learning. *arXiv preprint arXiv:2312.03801*.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. 2022. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Mark Rucker, Yinglun Zhu, and Paul Mineiro. 2023. Infinite action contextual bandits with reusable data exhaust. In *International Conference on Machine Learning*, pages 29259–29274. PMLR.

David Simchi-Levi and Yunzong Xu. 2021. Bypassing the monster: A faster and simpler optimal algorithm for contextual bandits under realizability. *Mathematics of Operations Research*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst.

Yanchao Sun, Shuang Ma, Ratnesh Madaan, Rogerio Bonatti, Furong Huang, and Ashish Kapoor. 2023. Smart: Self-supervised multi-task pretraining with control transformers. *arXiv preprint arXiv:2301.09816*.

Hampus Gummesson Svensson. 2023. *Sequential Decision-Making for Drug Design: Towards Closed-Loop Drug Design*. Chalmers Tekniska Hogskola (Sweden).

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Ambuj Tewari and Susan A Murphy. 2017. From ads to interventions: Contextual bandits in mobile health. *Mobile health: sensors, analytic methods, and applications*, pages 495–517.

Andrey Vasnetsov. 2018. Oneshot-wikilinks. https://www.kaggle.com/generall/oneshotwikilinks.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Mengxiao Zhang, Yuheng Zhang, Haipeng Luo, and Paul Mineiro. 2024. Efficient contextual bandits with uninformed feedback graphs. *arXiv preprint arXiv:2402.08127*.

Zhengqiu Zhang. 2011. An improvement to the brent's method. *International Journal of Experimental Algorithms*, 2(1):21–26.

Qinqing Zheng, Amy Zhang, and Aditya Grover. 2022. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR.

Yinglun Zhu, Dylan J Foster, John Langford, and Paul Mineiro. 2022a. Contextual bandits with large action spaces: Made practical. In *International Conference on Machine Learning*, pages 27428–27453. PMLR.

Yinglun Zhu, Julian Katz-Samuels, and Robert Nowak. 2022b. Near instance optimal model selection for pure exploration linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 6735–6769. PMLR.

Yinglun Zhu and Paul Mineiro. 2022. Contextual bandits with smooth regret: Efficient learning in continuous action spaces. In *International Conference on Machine Learning*, pages 27574–27590. PMLR.

Yinglun Zhu and Robert Nowak. 2020. On regret with multiple best arms. *Advances in Neural Information Processing Systems*, 33:9050–9060.

Yinglun Zhu and Robert Nowak. 2022. Pareto optimal model selection in linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 6793–6813. PMLR.

A Other Details for Experiments

A.1 Datasets

`OneShotWikiLinks` (Singh et al., 2012; Vasnetsov, 2018) is a named-entity recognition task where contexts are text phrases (English) preceding and following the mention text, and actions are text (English) phrases corresponding to the concept names. `OneShotWikiLinks-311` is a subset of this dataset obtained by taking all actions with at least 2000 examples. We construct binary reward function that is an indicator function for whether the action corresponds to the actual entity mentioned.

`AmazonCat-13K` (Bhatia et al., 2016) is an extreme multi-label dataset whose contexts are text phrases (English) corresponding to the title and content of an item, and actions are integers corresponding to item tags. We construct binary reward function that indicates whether (one of) the correct item tags is selected.

A.2 Models and Hyperparameters

A.2.1 Algorithm 1

We construct LLM-powered policies using Algorithm 1 and various LLM backbones, including Flan-T5 models of different sizes (Chung et al., 2024), Gemma 2B (instruct) (Team et al., 2024) and GPT-4o-mini (OpenAI, 2024a). We use sentence transformer (Reimers and Gurevych, 2019) as the embedding model, cosine similarity as the similarity measure, and hyperparameter $k = 1$. We provide the prompt design used in line 1 of Algorithm 1 below.

`OneShotWikiLinks-311`. We only run experiments with Flan-T5 models on this dataset. Given the text phrases preceding the mention text `text_preceding`, and following the mention text `text_following`, we aim to predict the mention text. Let `<extra_id_0>` represent the masked token in Flan-T5 models that needs to be filled in; we construct the prompt as:

```
question: text_preceding <extra_id_0>. text_following
```

`AmazonCat-13K`. We run experiments with Flan-T5 models, Gemma 2B and GPT-4o-mini on this dataset. Given the `title` and `content` of an item, we aim to predict the associated label. We construct prompts for different LLMs in the following.

- **Flan-T5 models.** We construct the prompt as:

  ```
  Title: title
  Content: content
  Task: Predict the associated label.
  ```

- **Gemma 2B.** Following the template provided in Google (2024), we construct the prompt as:

  ```
  <bos><start_of_turn>user
  Title: title
  Content: content
  Task: Predict the item tag based on the content and title.<end_of_turn>
  <start_of_turn>model
  ```

- **GPT-4o-mini.** Following the format provided in OpenAI (2024b), we construct system prompt as:

  ```
  Predict the item tag based on the content and title.
  ```

  and construct user prompt as:

  ```
  Title: title
  Content: content
  ```

A.2.2 Other Models and Hyperparameters

For SpannerGreedy, we adapt the implementation and hyperparameters from Zhu et al. (2022a). We use sentence transformer (Reimers and Gurevych, 2019) to embed contexts in $\mathbb{R}^{1536}$ by concatenating `text_preceding` and `text_following` (`OneShotWikiLinks-311`) or `title` and `content` (`AmazonCat-13K`). We use sentence transformer to embed actions in $\mathbb{R}^{768}$ and then apply SVD to reduce the dimensionality of actions to $\mathbb{R}^{50}$. SpannerGreedy uses a bilinear function $f(x, a) = \langle \phi(a), W\phi(x) \rangle$ to make prediction, where $\phi(\cdot)$ represents (pre-processed) embedding for contexts and actions. For Algorithm 3, we set the learning rate $\eta = 0.05$.

A.3 Other Details

We implement our code in PyTorch and run our experiments on NVIDIA Tesla V100 GPUs and NVIDIA A100 GPUs. Our paper uses several scientific artifacts, and our usage follows their licenses.