Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.39115

Cost-Aware Adaptive Prompt Selection for Large Language Models Using UCB and Thompson Sampling

Junbo Zhang

School of Statistics and Data Science, Capital University of Economics and Business, Beijing, China
15101613306@163.com

Abstract. This paper investigates cost-aware adaptive prompt selection for large language models by modeling four prompt strategies as arms in a multi-armed bandit. Standard UCB and Thompson Sampling are compared with contextual extensions, LinUCB and Linear Thompson Sampling (LinTS), on GSM8K. The study evaluates accuracy, cumulative reward, cumulative regret, average token usage, tokens per correct answer, and budget satisfaction. On the held-out evaluation set, the Global Best prompt achieved 96.8% accuracy with a cumulative regret of 3.00, while the Oracle upper bound reached 97.4%. LinUCB achieved 95.47% accuracy with an average of 200.18 tokens per question, reducing token use by approximately 9.9% relative to the Global Best prompt (222.14 tokens). Under the medium normalized token budget B = 0.2893, cost-constrained LinUCB maintained 95.47%-95.49% accuracy and 93% budget satisfaction, compared with 30%-35% for LinTS. In addition, 92% of evaluation questions produced identical binary rewards across all four prompts, indicating limited prompt-specific heterogeneity. The results suggest that contextual prompt selection may provide greater value for resource control than for accuracy improvement in homogeneous, high-accuracy reasoning settings.

Keywords: Large language model, Prompt optimization, Multi-armed bandit, UCB, Thompson Sampling, Contextual bandit

1. Introduction

Large language models (LLMs) can perform a broad range of downstream tasks through natural-language instructions and demonstrations without task-specific retraining [1]. However, model outputs are sensitive to prompt formulation, and prompt design has therefore become an important component of LLM deployment and evaluation [2].

Reasoning-oriented prompts further demonstrate that prompting can alter model behavior substantially. Chain-of-Thought prompting improves multi-step reasoning on arithmetic and symbolic tasks [3], while automated prompt methods such as AutoPrompt and Automatic Prompt Engineer reduce reliance on manual prompt engineering through search or model-based generation [4, 5]. Soft-prompt methods and reinforcement-learning-based prompt optimization provide additional mechanisms for adapting prompts at the task level [6, 7].

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.39115

Most prompt optimization methods search for a single prompt or prompt representation that performs well on average for a task. This task-level objective does not directly address an input-dependent question: whether different questions should receive different prompt strategies. A bandit formulation provides a natural sequential decision framework in which prompt strategies are actions and observed answer quality provides feedback. Standard UCB and Thompson Sampling balance exploration and exploitation using arm-level reward histories [8, 9], whereas contextual bandits condition decisions on the current input [10, 11].

The study makes three contributions. First, it formulates adaptive prompt selection as a four-arm bandit problem and explicitly compares non-contextual and contextual decision rules. Second, it constructs interpretable question context features by combining TF-IDF/Truncated-SVD semantic features with structural indicators, allowing LinUCB and LinTS to model input-dependent prompt utility. Third, it extends prompt selection to a token-budget setting using a dynamic cost penalty, enabling evaluation of accuracy, regret, and resource efficiency within one experimental framework.

2. Related work

2.1. Prompt optimization and reasoning prompts

Few-sshot prompting established that demonstrations in the input can enable strong task adaptation without parameter updates [1]. Chain-of-Thought prompting showed that explicit intermediate reasoning can improve multi-step mathematical reasoning [3]. Automated approaches subsequently reduced the dependence on manually written prompts: AutoPrompt searches discrete trigger tokens through gradient-guided optimization [4], the Automatic Prompt Engineer (APE) generates and evaluates candidate instructions with language models [5], prompt tuning learns continuous soft prompts [6], and RLPrompt optimizes discrete prompts with reinforcement learning [7]. These approaches demonstrate that prompt choice is an optimization problem, but they generally optimize a prompt at the task or dataset level rather than selecting among prompt strategies separately for each input.

2.2. Bandit-based adaptive selection

Multi-armed bandits provide a sequential framework for balancing exploitation of currently strong actions and exploration of uncertain alternatives. UCB uses optimism in the face of uncertainty [8], while Thompson sampling performs randomized selection through posterior sampling [9]. Contextual bandits extend this setting by conditioning expected reward on observable features. LinUCB models reward as a linear function of context and was originally developed for personalized recommendation [10], while Linear Thompson Sampling applies posterior sampling to linear contextual payoffs [11]. The present study connects these methods to prompt optimization by treating prompt templates as arms and question representations as contexts. Cost-aware selection is further motivated by contextual bandits with resource constraints [12].

3. Methodology

3.1. Problem formulation

The action space contains four prompt strategies. At round t, the system receives a GSM8K question, observes its context vector x_t for contextual methods, selects a prompt arm $a_t \in \mathcal{A}$, and

DOI: 10.54254/2755-2721/2026.39115

obtains a binary correctness reward. The action set is defined in Eq.1, and the reward is defined in Eq.2:

$$\mathcal{A} = \{\text{Zero-shot}, \text{Role-based}, \text{Step-by-step}, \text{Few-shot}\} \tag{1}$$

$$r_t = 1\{\hat{y}_t = y_t\} \tag{2}$$

Non-contextual UCB and Thompson Sampling estimate arm-level reward statistics $\mathbb{E}[r \mid a]$ from past observations and do not use question features. In contrast, LinUCB and LinTS estimate the conditional reward $\mathbb{E}[r \mid x, a]$, allowing the preferred prompt to vary with the current question. This distinction directly tests whether input-dependent prompt selection provides value beyond learning one globally strong prompt.

3.2. Context representation

Each question is represented by semantic and structural features. Semantic features are extracted with TF-IDF using lowercase unigrams and bigrams, a minimum document frequency of 2, and a maximum vocabulary size of 3,000. Truncated SVD reduces the sparse TF-IDF representation, with the SVD dimension selected on the calibration set from {5, 10, 15, 20}. Structural features contain 10 interpretable indicators: character length, word count, sentence count, number count, presence of percentage expressions, multiplier expressions, money terms, time terms, rate/speed terms, and ratio/proportion terms. Structural continuous variables are standardized using calibration-set statistics, an intercept term is appended, and the final vector is L2-normalized. The resulting context vector is defined in Eq.3. Table 1 summarizes the context feature construction and reproducibility settings.

$$x_t = \text{normalize}([1, z_t, h_t]) \tag{3}$$

Table 1. Context feature construction and reproducibility settings

Component	Setting
TF-IDF	lowercase=True, ngram_range=(1,2), min_df=2, max_features=3000
Truncated SVD	candidate dimensions: 5, 10, 15, 20, selected on calibration set
Structural features	10 features: length/count and percentage, multiplier, money, time, rate/speed, ratio indicators
Scaling	structural features standardized on calibration set only
Final normalization	intercept appended, L2 normalization
Random seed	42

3.3. Contextual bandit algorithms

3.3.1. LinUCB

For each prompt arm a, LinUCB assumes a linear contextual reward model [10]:

DOI: 10.54254/2755-2721/2026.39115

$$\mathbb{E}[r_t \mid x_t, a] = x_t^\top \theta_a \tag{4}$$

For each arm, the algorithm maintains A_a and b_a, initialized as $A_a = \lambda_{ridge} I$ and $b_a = 0$. The parameter estimate and action score are

$$\hat{\theta}_a = A_a^{-1} b_a \tag{5}$$

$$Score_{t,a} = x_t^\top \hat{\theta}_a + \alpha \sqrt{x_t^\top A_a^{-1} x_t} \tag{6}$$

The selected arm maximizes the score. Only the selected arm is updated after the observed reward. The exploration parameter α and ridge parameter λ_{ridge} are chosen using calibration replays.

$$A_a \leftarrow A_a + x_t x_t^\top, \qquad b_a \leftarrow b_a + r_t x_t \tag{7}$$

3.3.2. Linear Thompson Sampling

LinTS uses the same linear reward assumption but performs exploration through posterior sampling [11]. For each arm,

$$\tilde{\theta}_a \sim \mathcal{N}(A_a^{-1} b_a,\ v^2 A_a^{-1}) \tag{8}$$

$$a_t = \arg\max_{a \in \mathcal{A}} x_t^\top \tilde{\theta}_a \tag{9}$$

The posterior-scale parameter v controls sampling uncertainty and is selected on the calibration set. As in LinUCB, only the chosen arm is updated after observing the binary reward.

3.4. Offline replay and cost constraint

To ensure that all policies are compared against exactly the same LLM outputs, the study first evaluates every question under all four prompt strategies and stores a reward matrix and token-cost matrix. During replay, only the reward and cost corresponding to the chosen arm are revealed to the learning algorithm, preserving the partial-feedback bandit protocol [9].

Token cost C_t is defined as the total API token count for the selected prompt, i.e., input tokens plus generated output tokens. Raw total-token counts are min-max normalized using calibration-set token statistics before the cost-constrained experiments. The cost-aware objective follows the general resource-constrained contextual-bandit setting [12]:

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}[r_t] \\ \text{s.t.} \quad & \mathbb{E}[C_t] \le B \end{aligned} \tag{10}$$

A dynamic nonnegative multiplier λ_t is updated after each decision:

$$\lambda_{t+1} = \max\{0, \lambda_t + \eta(C_t - B)\} \tag{11}$$

DOI: 10.54254/2755-2721/2026.39115

When the observed normalized cost exceeds the target budget B, λ_t increases and future high-cost actions receive a larger penalty, otherwise the penalty can decrease. The medium-budget experiment uses $B = 0.2893$, the 50th percentile of normalized calibration costs.

3.5. Experimental setup and evaluation protocol

Dataset. GSM8K contains grade-school mathematical word problems designed to evaluate multi-step arithmetic reasoning [13]. The calibration set contains 400 questions sampled without replacement from the GSM8K training split, and the held-out evaluation set contains 500 questions sampled without replacement from the test split. Random seed 42 is used for data selection.

Base LLM and generation settings. All four prompt arms use the same DeepSeek API model, deepseek-v4-flash, with internal thinking disabled, temperature 0.0, and maximum output length 512 tokens. A shared system instruction requires the final numeric result to follow the format "FINAL ANSWER: number". The program extracts this number and compares it with the GSM8K ground truth, numerical equality is checked with tolerance 1e-6.

Prompt definitions. Zero-shot directly requests a solution and final numeric answer. Role-based assigns the role of a careful mathematics tutor and requests verification before answering. Step-by-step explicitly requests intermediate calculations before the final answer. Few-shot provides two solved arithmetic examples before the target question. The four arms were selected to represent a progression from minimal instruction to role conditioning, explicit reasoning guidance, and demonstration-based prompting while keeping the base model unchanged. Table 2 lists the four prompt arms and their operational definitions.

Table 2. Prompt action space

Prompt arm	Operational definition
Zero-shot	Direct solution request, no example or role.
Role-based	Mathematics-tutor role, solve and check the calculation.
Step-by-step	Request explicit intermediate calculations.
Few-shot	Two solved arithmetic examples precede the target question.

Baselines. TThe comparison includes Random selection, Global Best selected from calibration accuracy, Standard UCB, Standard Beta-Bernoulli Thompson Sampling [14], LinUCB, LinTS, and Oracle. Oracle chooses the best realized prompt for each evaluation question from the complete reward matrix and is used only as an upper bound.

Together, these baselines cover random, fixed, non-contextual, contextual, and upper-bound policies within a standard sequential-decision perspective [15].

Hyperparameter calibration. Hyperparameters are tuned exclusively on the calibration set. The search grids are SVD dimension {5,10,15,20}, LinUCB α {0.01,0.025,0.05,0.10,0.25,0.50}, ridge {0.1,1.0,5.0}, LinTS v {0.05,0.10,0.25,0.50}, and Standard-UCB c {0.01,0.025,0.05,0.10,0.25,0.50}. Calibration uses 30 replay seeds. Final evaluation averages over 100 replay orderings. Cost sensitivity uses $\eta \in \{0.01,0.05,0.10\}$ and budget quantiles {0.30,0.50,0.70} computed from normalized calibration costs.

Metrics. Accuracy is the mean binary reward. Cumulative reward is the running sum of rewards. Cumulative regret compares the selected-arm reward with the best realized reward available for each question. Average Tokens is total input-plus-output tokens divided by the number of questions.

DOI: 10.54254/2755-2721/2026.39115

Tokens/Correct is total token consumption divided by the number of correct answers. Budget Satisfaction is the proportion of replay settings whose average normalized cost satisfies the target budget.

4. Experimental results

4.1. Prompt-level performance

As shown in Table 3, Step-by-step achieved the highest calibration and evaluation accuracy, at 95.00% and 96.80%, respectively. Zero-shot used the fewest tokens (181.85 per question), whereas Few-shot used the most (281.44).

Table 3. Prompt-level accuracy and token cost

Prompt	Cal. Acc. (%)	Eval. Acc. (%)	Avg. Tokens
Zero-shot	93.00	95.00	181.85
Role-based	94.50	95.60	203.51
Step-by-step	95.00	96.80	222.14
Few-shot	92.25	92.60	281.44

4.2. Comparison of prompt selection algorithms

Table 4 summarizes overall performance. Oracle achieved the highest accuracy (97.40%) and zero regret. Among deployable methods, Global Best achieved the highest accuracy (96.80%) and the lowest cumulative regret (3.00). LinUCB achieved 95.47% accuracy with the lowest average token consumption among the learned bandit methods, at 200.18 tokens per question and 209.68 tokens per correct answer.

Table 4. Overall performance of prompt selection methods

Algorithm	Acc. (%)	Regret	Avg. Tokens	Tokens/Correct
Random	94.944	12.28	222.03	233.86
Standard UCB	95.584	9.08	213.00	222.88
Standard TS	95.590	9.05	216.76	226.76
LinUCB	95.470	9.65	200.18	209.68
LinTS	95.210	10.95	220.93	232.08
Global Best	96.800	3.00	222.14	229.48
Oracle	97.400	0.00	182.47	187.34

DOI: 10.54254/2755-2721/2026.39115

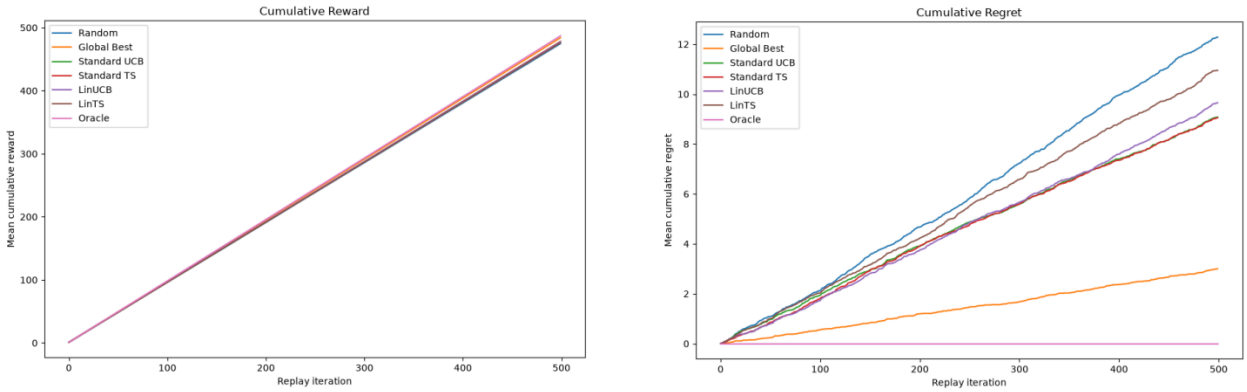


Figure 1. Mean cumulative reward (a) and mean cumulative regret (b) across replay iterations

Figure 1 shows that cumulative reward curves remain relatively close across methods, whereas the regret curves provide clearer separation. Global Best remains below the learned bandit methods in cumulative regret, while Random accumulates the largest regret.

4.3. Context dependence and feature ablation

Table 5 reports the LinUCB prompt-selection distribution by context cluster. The selected prompt proportions vary across clusters, indicating that the policy does not collapse completely to one fixed prompt.

Table 5. LinUCB prompt selection share by context cluster

Cluster	Zero-shot (%)	Role-based (%)	Step-by-step (%)	Few-shot (%)
0	44.11	33.93	20.42	1.54
1	38.11	43.53	15.55	2.81
2	34.90	39.81	23.06	2.23
3	38.40	43.20	16.58	1.82

As shown in Table 6, LinUCB reached 95.458% accuracy using structural features alone and 95.470% using combined semantic and structural features. LinTS obtained its highest ablation accuracy, 95.288%, with semantic features alone.

Table 6. Context feature ablation

Feature Set	LinUCB Acc. (%)	LinTS Acc. (%)
Semantic only	95.030	95.288
Structural only	95.458	95.100
Semantic + Structural	95.470	95.210

4.4. Cost-constrained results

Table 7 presents the medium-budget results (B = 0.2893). Across η values, cost-constrained LinUCB maintained 95.472%-95.488% accuracy with 199.62-200.10 average tokens and 93% budget

DOI: 10.54254/2755-2721/2026.39115

satisfaction. LinTS achieved 95.294%-95.430% accuracy, used 213.54-219.11 average tokens, and achieved 30%-35% budget satisfaction.

Table 7. Cost-constrained results under the medium budget

Algorithm	η	Acc. (%)	Avg. Tokens	Budget Sat. (%)
LinUCB	0.01	95.472	200.10	93
LinUCB	0.05	95.486	199.91	93
LinUCB	0.10	95.488	199.62	93
LinTS	0.01	95.294	219.11	30
LinTS	0.05	95.376	215.13	33
LinTS	0.10	95.430	213.54	35

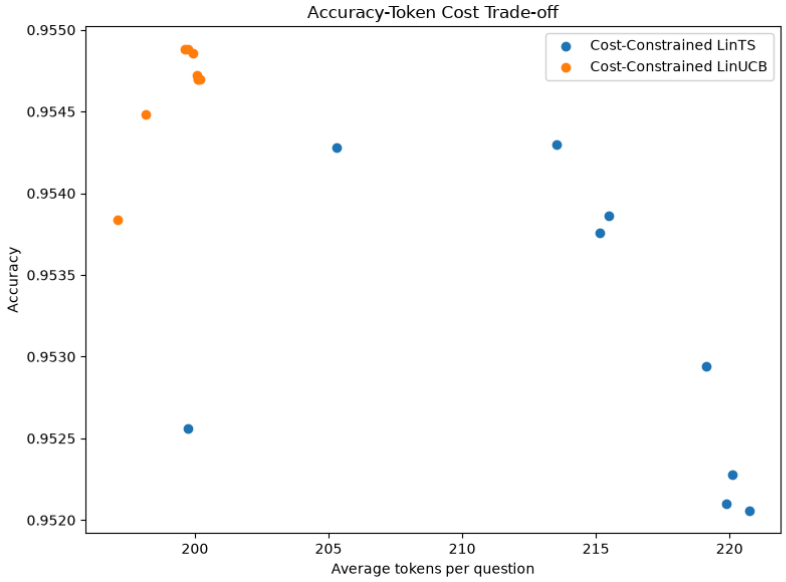


Figure 2. Accuracy-token cost trade-off under cost-constrained contextual bandits

As shown in Figure 2, all LinUCB configurations lie in a lower-token region than LinTS while maintaining comparable or slightly higher accuracy. LinUCB uses approximately 6.5%–8.7% fewer tokens than LinTS at comparable or slightly higher accuracy levels.

5. Discussion

5.1. Why context did not improve accuracy

The contextual methods did not outperform the Global Best prompt in accuracy. A central explanation is limited prompt-specific heterogeneity: 460 of 500 evaluation questions (92%) received exactly the same binary reward under all four prompts. When most questions do not exhibit a "one prompt correct, another prompt incorrect" contrast, x_t contains little learnable signal about which arm can improve correctness. This also explains why the context-conditioned selection distribution changes across clusters without yielding a higher overall accuracy.

DOI: 10.54254/2755-2721/2026.39115

The prompt-level results further show that Step-by-step is consistently strongest on both calibration and evaluation splits. This stability makes a calibration-selected Global Best strategy difficult to improve upon. The gap between Global Best (96.8%) and Oracle (97.4%) is only 0.6 percentage points, which indicates a strong ceiling effect: even perfect per-question selection within the existing four-arm action space can improve accuracy only slightly.

5.2. Resource-efficiency implications

Although contextual selection did not improve accuracy, LinUCB reduced average token usage. Relative to Global Best, LinUCB used 200.18 versus 222.14 tokens per question, a reduction of approximately 9.9%, while retaining 95.47% accuracy. The constrained experiments reinforce this interpretation: LinUCB maintained 93% budget satisfaction across the tested η values at the medium budget, substantially above LinTS.

Thus, in settings where the base model is already accurate, contextual selection may be more useful as a resource-allocation mechanism than as a pure accuracy maximizer.

5.3. Feature representation and algorithm behavior

The feature ablation results show only small differences among semantic-only, structural-only, and combined representations. For LinUCB, structural-only performance is nearly identical to the combined feature set, while LinTS performs slightly better with semantic-only features.

This suggests that increasing feature complexity does not automatically create a stronger contextual signal. The usefulness of contextual bandits depends not only on representation quality but also on whether the underlying action rewards truly vary with context.

5.4. Limitations and future work

The study has several limitations. First, the experiments use only GSM8K, so the task distribution is relatively homogeneous and dominated by mathematical word problems. Second, the base model already achieves high accuracy, leaving limited headroom for prompt selection. Third, reward is binary and evaluates only final-answer correctness rather than reasoning quality. Fourth, the action space contains only four manually designed prompt strategies. Future work should evaluate multiple heterogeneous benchmarks, harder or more prompt-sensitive tasks, richer embeddings, larger prompt pools, and finer-grained rewards that jointly capture correctness, reasoning quality, latency, and monetary cost. True online evaluation would also help assess robustness when LLM outputs are stochastic and nonstationary.

6. Conclusion

This study formulated prompt selection for large language models as a multi-armed bandit problem and compared standard UCB and Thompson Sampling with the contextual variants LinUCB and LinTS under both unconstrained and token-budget settings. On GSM8K, the calibration-selected Global Best prompt achieved the strongest deployable accuracy, while contextual methods did not provide an additional accuracy gain. LinUCB nevertheless achieved lower token consumption and stronger budget satisfaction, demonstrating a useful accuracy-cost trade-off. The findings indicate that adaptive prompt selection is most likely to benefit settings in which prompt effectiveness varies meaningfully across inputs, when prompt rewards are highly homogeneous, a fixed strong prompt

DOI: 10.54254/2755-2721/2026.39115

can remain difficult to outperform. These results support evaluating prompt-selection systems not only by accuracy, but also by regret, token efficiency, and budget compliance.

References

[1] Brown, T. B., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877-1901.

[2] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys, 55*(9), 1-35. https://doi.org/10.1145/3560815

[3] Wei, J., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems, 35*, 24824-24837.

[4] Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., & Singh, S. (2020). AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. *Proceedings of EMNLP*, 4222-4235. https://doi.org/10.18653/v1/2020.emnlp-main.346 https://aclanthology.org/2020.emnlp-main.346/

[5] Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2023). Large language models are human-level prompt engineers. *International Conference on Learning Representations (ICLR)*.

[6] Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *Proceedings of EMNLP*, 3045-3059. https://aclanthology.org/2021.emnlp-main.243/

[7] Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E., & Hu, Z. (2022). RLPrompt: Optimizing discrete text prompts with reinforcement learning. *Proceedings of EMNLP*, 3369-3391. https://doi.org/10.18653/v1/2022.emnlp-main.222 https://aclanthology.org/2022.emnlp-main.222/

[8] Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning, 47*, 235-256. https://doi.org/10.1023/A:1013689704352

[9] Russo, D., & Van Roy, B. (2016). An information-theoretic analysis of Thompson sampling. *Journal of Machine Learning Research, 17*(68), 1-30.

[10] Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web*, 661-670. https://doi.org/10.1145/1772690.1772758

[11] Agrawal, S., & Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. *Proceedings of the 30th International Conference on Machine Learning, PMLR 28*(3), 127-135. https://proceedings.mlr.press/v28/agrawal13.html

[12] Agrawal, S., & Devanur, N. R. (2016). Linear contextual bandits with knapsacks. *Advances in Neural Information Processing Systems, 29*. https://proceedings.neurips.cc/paper_files/paper/2016/hash/f3144cefe89a60d6a1afaf7859c5076b-Abstract.html

[13] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. arXiv:2110.14168. https://arxiv.org/abs/2110.14168

[14] Agrawal, S., & Goyal, N. (2012). Analysis of Thompson sampling for the multi-armed bandit problem. *Proceedings of the 25th Annual Conference on Learning Theory, 23*, 39.1-39.26.

[15] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.