BEST ARM IDENTIFICATION FOR PROMPT LEARNING UNDER A LIMITED BUDGET

Chengshuai Shi[†,*], **Kun Yang**[†,*], **Jing Yang**[‡], **Cong Shen**[†]
[†] University of Virginia
[‡] The Pennsylvania State University
[*] Equal contribution, random order
{cs7ync, ky9tc}@virginia.edu, yangjing@psu.edu, cong@virginia.edu

ABSTRACT

The remarkable instruction-following capability of large language models (LLMs) has sparked a growing interest in automatically learning suitable prompts. However, while many effective methods have been proposed, the *cost* incurred during the learning process (e.g., accessing LLM and evaluating the responses) has not been considered. To overcome this limitation, this work explicitly incorporates a finite budget constraint into prompt learning. Towards developing principled solutions, a novel connection is established between prompt learning and fixed-budget best arm identification (BAI-FB) in multi-armed bandits (MAB). Based on this connection, a general framework TRIPLE (besT aRm Identification for Prompt LEarning) is proposed to harness the power of BAI-FB in prompt learning systematically. Extensive experiments on multiple well-adopted tasks using both GPT 3.5 and Llama2 demonstrate the superiority of TRIPLE over the previous baselines.

1 INTRODUCTION

Large language models (LLMs) have rapidly changed technology landscapes in our society (Touvron et al., 2023a;b; OpenAI, 2023a). Their remarkable ability to follow instructions has motivated the study of searching for suitable prompts to interact with them (Liu et al., 2023). One representative example is the success of zero-shot Chain-of-Thoughts prompting (Kojima et al., 2022), i.e., adding a seemingly simple phrase of "let's think step by step" bumps the reasoning capability of LLMs. Nevertheless, it has also been recognized that the performance of an LLM is sensitive to the selected prompts (Zhao et al., 2021; Lu et al., 2021), and manually designing suitable prompts can be labor-intensive (Mishra et al., 2021). Thus, there is a growing interest to automatically perform prompt learning, e.g., Diao et al. (2022); Deng et al. (2022); Zhang et al. (2023a); Pan et al. (2023).

However, the costs incurred during prompt learning have not been explicitly considered in previous work. In particular, most of them do not measure the required number of accesses to LLMs, while in practice, such accesses are often costly (e.g., each OpenAI API access incurs a cost). Furthermore, it is often overlooked that evaluating the responses of an LLM for different prompts can also be expensive as many prompting tasks would require human opinions (e.g., writing improvement). Thus, this work proposes to study prompt learning under an explicitly imposed budget constraint when interacting with the targeted LLM. The main contributions of this work are summarized as follows.

• The constraint of a limited budget is explicitly introduced into prompt learning, which has been largely ignored before. As most of the existing methods rely on selecting from a pre-generated candidate prompt pool, we focus our study on how to carefully allocate budgets to test each candidate prompt so that the optimal one can be learned efficiently and effectively.

• We develop a general solution framework, termed TRIPLE (besT aRm Identification for Prompt LEarning), by establishing a novel connection between prompt learning and fixed-budget best arm identification (BAI-FB) (Audibert et al., 2010; Karnin et al., 2013) in multi-armed bandits (Lattimore & Szepesvári, 2020). Two representative designs TRIPLE-SH and TRIPLE-CR, inspired by celebrated BAI-FB algorithms, are presented. To improve the scalability of our design, two enhanced methods, TRIPLE-CLST and TRIPLE-GSE, are proposed, where prompt embeddings are leveraged via clustering and function approximation, respectively.

• Extensive experimental results are reported using well-adopted prompt tasks to demonstrate the superiority of the TRIPLE solutions over previous baselines. In particular, when handling small prompt pools, the basic TRIPLE-SH and TRIPLE-CR already outperform all prior baselines by (on average) 13% to 35%; when tackling large prompt pools, the enhanced TRIPLE-CLST and TRIPLE-GSE further achieve additional 13% to 16% average gain over TRIPLE-CR, all on GPT 3.5. The gains are even more pronounced on Llama2-7b. Moreover, the proposed methods can be directly plugged into two popular prompt learning pipelines, APE (Zhou et al., 2022) and APO (Pryzant et al., 2023), demonstrating significantly improved end-to-end performances.

2 Prompt Learning under a Limited Budget

Consider that we are using an LLM $f(\cdot)$, which provides a mapping from any input $X \in \mathcal{V}$ to a distribution $\Delta_{\mathcal{V}}$ over the language space $\mathcal{V}$. The answer $\hat{Y} \in \mathcal{V}$ given by the LLM is assumed to be sampled from $f(X)$ as $\hat{Y} \sim f(X)$. For prompt learning, we aim to find a prompt p such that when concatenated with inputs X of a certain task (i.e, as $[p; X]$), it provides good performance, while the performance can be measured as $\mu(p) := \mathbb{E}_{X \sim \mathcal{I}_X} \mathbb{E}_{\hat{Y} \sim f([p;X])}[s(X, \hat{Y})]$, where $s(X, \hat{Y})$ denotes a score function that measures the quality of the output $\hat{Y}$ for the input X.

Intuitively, the process of learning a good prompt would require interactions with the LLM (i.e., sample $\hat{Y} \sim f([p; X])$ and evaluating its responses (i.e., obtain score $s(X, \hat{Y})$). However, as mentioned in Sec. 1, such interactions and evaluations are often costly. Thus, we explicitly take into account that the prompt learning process should have a limited budget: *the total number of interactions with the LLM that happen during the learning is at most N*.

Many proposed prompt learning methods rely on the pipeline of first generating a pool of candidate prompts and then selecting from it (Jiang et al., 2020; Zhou et al., 2022; Xu et al., 2022; Prasad et al., 2022; Guo et al., 2023). From a unified perspective, we can simplify the problem into generating a pool of prompts $\mathcal{P}$ and finding the optimal prompt in it: $p^* := \arg\max_{p \in \mathcal{P}} \mu(p)$. However, the previous works are largely focused on the prompt generation component, with a limited attention on the prompt selection component. With an explicit budget limitation, however, we need to carefully allocate the budgets to each prompt so that the optimal prompt can be correctly learned. An overview of the considered prompt learning pipeline and our focus is illustrated in Fig. 1.

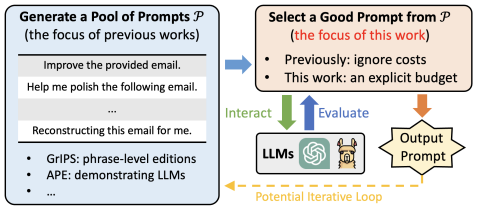

Figure 1: An overview of the common pipeline for prompt learning, where GrIPS and APE are proposed in Prasad et al. (2022); Zhou et al. (2022).

3 Connecting Prompt Learning with Best Arm Identification

We provide a new perspective of prompt learning through the lens of multi-armed bandits (MAB) (Lattimore & Szepesvári, 2020) and especially, propose to both leverage existing and design new fixed-budget best arm identification (BAI-FB) schemes to tackle the considered limited budget.

Basics of Multi-armed Bandits. The most basic form of MAB considers a system with a set $\mathcal{K}$ finite arms that provide stochastic rewards when pulled. When interacting with the system, the agent can select one arm $k \in \mathcal{K}$ to pull at each time, and she receives a stochastic reward $r_k \sim \texttt{dist}_k(\nu_k)$, where $\texttt{dist}_k(\nu_k)$ denotes the action k's reward distribution with an unknown expectation ν_k.

The learning objective of the agent in MAB can be roughly divided into two categories: (1) *regret minimization*, which maximizes the expected cumulative rewards collected by the agent (Auer et al., 2002; Audibert et al., 2009; Garivier & Cappé, 2011); (2) *best arm identification*, which targets at outputting the best arm $k^* = \arg\max_{k \in \mathcal{K}} \nu_k$ (Audibert et al., 2010; Garivier & Kaufmann, 2016; Jamieson & Nowak, 2014). These two objectives often require different learning strategies.

A Bandit View of Prompt Learning. Based on the above introduction, it can be intuitively understood that the prompt learning (especially, selection) problem can be mapped into an MAB setting:

- The pool of candidate prompts $\mathcal{P}$ is equivalent to the set of arms $\mathcal{K}$;
- Using a prompt p to interact with an LLM is selecting an arm k to pull in the MAB model;
- The feedback of the score function, i.e., $s(X, \hat{Y})$, provides the reward signal r_k, where $\texttt{dist}_k(\nu_k)$ characterizes the randomness of $X \sim \mathcal{I}_X$ and $\hat{Y} \sim f([p; X])$. The expected performance $\mu(p)$ is the counterpart of the expected reward ν_k in MAB.

It can be further recognized that the target of prompt learning is more suitable to be captured as *best arm identification* (BAI), instead of regret minimization, as it only cares about finding the optimal prompt p^* instead of performance during learning. Moreover, prompt learning under a limited budget aligns with one of the main research directions in BAI called *fixed-budget best arm identification* (BAI-FB) (Karnin et al., 2013; Gabillon et al., 2012), which considers the problem of *correctly identifying the best arm k^* with a fixed number of pulls*, which provides a perfect toolbox to enhance the prompt selection process. A summary of this connection can be found in Table 1.

Table 1: Connections between Prompt Learning and MAB.

Prompt Learning	**Multi-armed Bandits**
The pool of prompts $\mathcal{P}$	The arm set $\mathcal{K}$
Interact LLM via prompt p	Pull arm k
Score $s(X, \hat{Y})$	Reward r_k
Randomness in X and $\hat{Y}$	Randomness in $\texttt{dist}_k$
Performance $\mu(p)$	Expected reward ν_k
Learn the optimal prompt under a limited budget	Fixed-budget best arm identification (BAI-FB)

Harnessing the Power of BAI-FB. We propose a general framework called TRIPLE (bes**T** a**R**m **I**dentification for **P**rompt **LE**arning) to harness the power of BAI-FB in prompt learning.

As a first step, inspired two successful BAI-FB schemes, the classical Sequential Halving (SH) (Karnin et al., 2013) and the state-of-the-art Continuously Reject (CR) (Wang et al., 2023), we propose TRIPLE-SH and TRIPLE-CR. Complete details are deferred to Appendix B.

Furthermore, to share information among prompts during learning, we propose to leverage an embedding model (e.g., the OpenAI embedding API), denoted as $\texttt{embed} : \mathcal{V} \to \mathbb{R}^d$, to obtain the sentence embedding of the prompts. With the obtained prompt embeddings, two useful enhancements, TRIPLE-CLST and TRIPLE-GSE are proposed to improve the learning effectiveness.

The first enhancement TRIPLE-CLST is a two-phased one: in Phase I, the entire pool of candidate prompts is clustered based on their embeddings, and BAI-FB is performed to find the optimal cluster; then, in Phase II, BAI-FB is performed within the optimal cluster to find one final prompt. The second enhancement TRIPLE-GSE, instead, uses the idea of function approximation by learning a common function (e.g., an MLP) to predict the prompt features based on their features, where GSE Azizi et al. (2023), a recent BAI-FB scheme, is incorporated. Complete details are in Appendix B.

4 EXPERIMENTS

In this section, extensive experimental results collected from both gpt-3.5-turbo-1106 model of OpenAI (OpenAI, 2023a) as well as Llama2-7b Touvron et al. (2023b) are reported to evaluate the efficiency of TRIPLE across diverse prompting tasks from two standard datasets: Instruction-Induction Honovich et al. (2022) and BigBench Suzgun et al. (2022).

Evaluating TRIPLE with Fixed Prompt Pools. As TRIPLE main focuses on the prompt selection component, we perform initial evaluations in an isolated fashion of selecting from fixed pools of candidate prompts. For this experiment, candidate pools of prompts are generated following the well-established APE design (Zhou et al., 2022). Then, under a limited budget, the performances of TRIPLE algorithms are compared with the following two baselines:

- **Uniform.** Many previous designs choose to evaluate the entire candidate pool on all development data, e.g., Guo et al. (2023); Prasad et al. (2022). With a limited budget, this strategy corresponds to equally dividing the total budget to test all prompts.
- **UCB.** The UCB method (Auer et al., 2002) is a famous design for regret minimization. Its variants have been applied to prompt selection in Pryzant et al. (2023); Lin et al. (2023).

Performance with small prompt pools. We begin with generating 10 prompts for selection in each task. Results with an overall budget of 50 are reported in Fig. 2 (top). It can be observed that

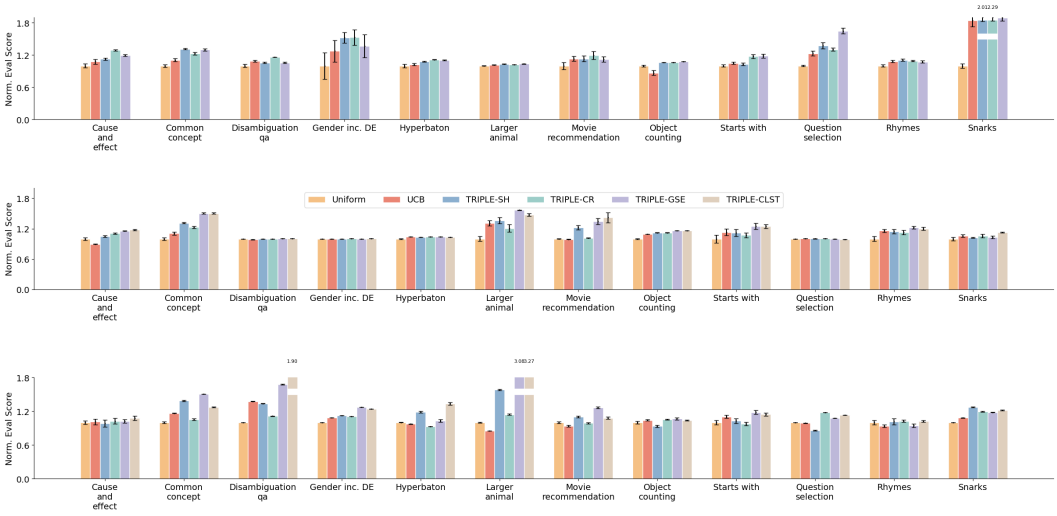

Figure 2: Performance comparison of various prompt selection methods on selected tasks: small prompt pools with $|\mathcal{P}| = 10$ and gpt-3.5 (top), large prompt pools with $|\mathcal{P}| = 30$ and gpt-3.5 (middle), and large prompt pools with $|\mathcal{P}| = 30$ and llama2-7b (bottom). The reported results (y-axis) are test accuracies of each method normalized to the mean performance of "Uniform" on that task, and are aggregated over 20 independent runs. `TRIPLE-CLST` is not tested for handling small prompt pools because it is ineffective to cluster only 10 prompts. The full results on 47 tasks are reported in Appendix D.

Table 2: Performances of integrating `TRIPLE` in the end-to-end pipelines. The baseline methods reported in the original implementations are labeled as (b). For each task, the best score across two pipelines is marked as red, and the best score in the remaining pipeline is highlighted as yellow. `TRIPLE-CR` are selected over `TRIPLE-SH` due to its better performance observed in the previous experiments. `TRILE-CLST` is ignored in the tests with APO, as it is ineffective to cluster only 10 prompts.

	APE Zhou et al. (2022)				APO Pryzant et al. (2023)		
Tasks	**Uniform** (b)	**CR**	**CLST**	**GSE**	**UCB** (b)	**CR**	**GSE**
Cause and effect	0.65 ± 0.18	0.74 ± 0.06	0.75 ± 0.13	0.78 ± 0.08	0.78 ± 0.15	0.80 ± 0.05	0.80 ± 0.08
Common concept	0.09 ± 0.05	0.12 ± 0.06	0.10 ± 0.04	0.14 ± 0.05	0.12 ± 0.04	0.12 ± 0.05	0.14 ± 0.01
Disambiguation qa	0.83 ± 0.04	0.88 ± 0.10	0.97 ± 0.01	0.96 ± 0.01	0.95 ± 0.04	0.98 ± 0.02	0.96 ± 0.02
Gender inc. DE	0.74 ± 0.17	0.81 ± 0.10	0.85 ± 0.12	0.84 ± 0.14	0.69 ± 0.22	0.80 ± 0.17	0.88 ± 0.05
Hyperbaton	0.78 ± 0.07	0.83 ± 0.11	0.84 ± 0.12	0.84 ± 0.11	0.59 ± 0.24	0.74 ± 0.21	0.79 ± 0.18
Larger animal	0.56 ± 0.24	0.64 ± 0.25	0.79 ± 0.06	0.84 ± 0.02	0.66 ± 0.13	0.73 ± 0.18	0.85 ± 0.15
Movie recommendation	0.61 ± 0.12	0.65 ± 0.18	0.76 ± 0.06	0.74 ± 0.14	0.67 ± 0.11	0.65 ± 0.15	0.71 ± 0.15
Object counting	0.41 ± 0.12	0.45 ± 0.08	0.50 ± 0.07	0.48 ± 0.12	0.44 ± 0.08	0.50 ± 0.09	0.49 ± 0.07
Orthography starts with	0.41 ± 0.21	0.65 ± 0.16	0.67 ± 0.12	0.66 ± 0.13	0.58 ± 0.13	0.64 ± 0.09	0.67 ± 0.17
Question selection	0.90 ± 0.04	0.91 ± 0.03	0.95 ± 0.01	0.93 ± 0.03	0.93 ± 0.06	0.92 ± 0.06	0.93 ± 0.03
Rhymes	0.66 ± 0.30	0.68 ± 0.26	0.75 ± 0.20	0.78 ± 0.16	0.78 ± 0.12	0.83 ± 0.08	0.85 ± 0.13
Snarks	0.44 ± 0.10	0.52 ± 0.19	0.57 ± 0.10	0.60 ± 0.21	0.49 ± 0.17	0.56 ± 0.15	0.67 ± 0.05

`TRIPLE` methods outperform two baselines on most of the tasks, evidencing its prowess in allocating limited budgets for prompt selection. Especially, across the representative 12 tasks, `TRIPLE-CR` outperforms Uniform with improvements ranging from 8% to 129%, and on average, a notable gain of 35%. Moreover, it achieves a 3% to 30% improvement over UCB, with an average 13% gain.

Performance with large prompt pools. We next test larger candidate pools with 30 prompts per task and an overall budget of 150. It can be observed in Fig. 2 (middle) that while `TRIPLE-SH` and `TRIPLE-CR` still achieve better performance than Uniform and UCB, the enhanced `TRIPLE-CLST` and `TRIPLE-GSE` demonstrate more remarkable improvements. In particular, both `TRIPLE-CLST` and `TRIPLE-GSE` significantly outperform the previously best `TRIPLE-CR` method (let alone Uniform and UCB), with a gain ranging from 2% to 60%, averaging of 13% and 16%, respectively. With the same prompt pools, the performance on Llama2-7b are shown in Fig. 2 (bottom), where the superiority of `TRIPLE` is even more evident: `TRIPLE-GSE` and `TRIPLE-CLST` achieve average gains of 45% and 51% over Uniform, demonstrating the effectiveness of `TRIPLE` across LLMs.

Integrating `TRIPLE` into End-to-End Pipelines. We now explore whether `TRIPLE` can provide performance improvements when plugged into end-to-end prompt learning pipelines that include both prompt generation and selection. To this end, two end-to-end designs are considered, APE (Zhou et al., 2022) and APO (Pryzant et al., 2023). The end-to-end experiment results are reported in Table 2, which reveal the consistently better performance of `TRIPLE` over the originally adopted baseline

methods. This observation highlights the applicability and flexibility of the TRIPLE framework, i.e., it can benefit any prompt learning pipelines requiring a selection component.

5 CONCLUSIONS

This work have explicitly incorporated a budget constraint to prompt learning and established a systematical connection between multi-armed bandits (MAB) and prompt learning. Through the lens of MAB, we proposed a general framework, termed TRIPLE, to fully harness the power of fixed-budget best arm identification (BAI-FB) to perform prompt learning under a limited budget. Extensive experiments demonstrated the superiority of TRIPLE in multiple tasks with various LLMs.

REFERENCES

Alexia Atsidakou, Sumeet Katariya, Sujay Sanghavi, and Branislav Kveton. Bayesian fixed-budget best-arm identification. *arXiv preprint arXiv:2211.08572*, 2022.

Jean-Yves Audibert, Sébastien Bubeck, et al. Minimax policies for adversarial and stochastic bandits. In *COLT*, volume 7, pp. 1–122, 2009.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*, pp. 41–53, 2010.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

Mohammad Javad Azizi, Branislav Kveton, and Mohammad Ghavamzadeh. Fixed-budget best-arm identification in structured bandits. *arXiv preprint arXiv:2106.04763*, 2023.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. *arXiv preprint arXiv:2306.03082*, 2023.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, Yong Lin, Xiao Zhou, and Tong Zhang. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*, 2022.

Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. *Advances in Neural Information Processing Systems*, 25, 2012.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory*, pp. 359–376. JMLR Workshop and Conference Proceedings, 2011.

Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pp. 998–1027. PMLR, 2016.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*, 2023.

Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer (eds.), *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.

Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. Instruction induction: From few examples to natural language task descriptions. *arXiv preprint arXiv:2205.10782*, 2022.

Kevin Jamieson and Robert Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6. IEEE, 2014.

Yassir Jedra and Alexandre Proutiere. Optimal best-arm identification in linear bandits. *Advances in Neural Information Processing Systems*, 33:10007–10017, 2020.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International conference on machine learning*, pp. 1238–1246. PMLR, 2013.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Use your instinct: Instruction optimization using neural bandits coupled with transformers. *arXiv preprint arXiv:2310.02905*, 2023.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. Reframing instructional prompts to gptk's language. *arXiv preprint arXiv:2109.07830*, 2021.

OpenAI. Gpt-3.5-turbo. https://platform.openai.com/docs/models/gpt-3-5, 2023a. Accessed: 2024-01-29.

OpenAI. Text-embedding-ada-002. https://platform.openai.com/docs/models/embeddings, 2023b. Accessed: 2024-01-29.

Rui Pan, Shuo Xing, Shizhe Diao, Xiang Liu, Kashun Shum, Jipeng Zhang, and Tong Zhang. Plum: Prompt learning using metaheuristic. *arXiv preprint arXiv:2311.08364*, 2023.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv preprint arXiv:2203.07281*, 2022.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*, 2023.

Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too? *arXiv preprint arXiv:2212.10539*, 2022.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Marta Soare, Alessandro Lazaric, and Rémi Munos. Best-arm identification in linear bandits. *Advances in Neural Information Processing Systems*, 27, 2014.

Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse rl. *arXiv e-prints*, pp. arXiv–2309, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Po-An Wang, Ruo-Chun Tzeng, and Alexandre Proutiere. Best arm identification with fixed budget: A large deviation perspective. *arXiv preprint arXiv:2312.12137*, 2023.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *arXiv preprint arXiv:2302.03668*, 2023.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vydiswaran, and Hao Ma. Idpg: An instance-dependent prompt generation method. *arXiv preprint arXiv:2204.04497*, 2022.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. Gps: Genetic prompt search for efficient few-shot learning. *arXiv preprint arXiv:2210.17041*, 2022.

Junwen Yang and Vincent Tan. Minimax optimal fixed-budget best arm identification in linear bandits. *Advances in Neural Information Processing Systems*, 35:12253–12266, 2022.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. Tempera: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023a.

Zhihan Zhang, Shuohang Wang, Wenhao Yu, Yichong Xu, Dan Iter, Qingkai Zeng, Yang Liu, Chenguang Zhu, and Meng Jiang. Auto-instruct: Automatic instruction generation and ranking for black-box language models. *arXiv preprint arXiv:2310.13127*, 2023b.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pp. 12697–12706. PMLR, 2021.

Zexuan Zhong, Dan Friedman, and Danqi Chen. Factual probing is [mask]: Learning vs. learning to recall. *arXiv preprint arXiv:2104.05240*, 2021.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

A Related Works

Prompt Learning. The study of prompt learning (also known as instruction learning) focuses on automatically learning suitable prompts, which is more scalable compared with manual prompt engineering. Many efforts have been devoted to this direction (Gao et al., 2020; Wen et al., 2023; Sun et al., 2023; Zhang et al., 2022; 2023b), which is summarized in a recent survey of Liu et al. (2023). The early studies mostly consider optimizing soft prompts (i.e., continuous vectors) (Lester et al., 2021; Li & Liang, 2021; Zhong et al., 2021; Liu et al., 2021; Wu et al., 2022) or discrete but not interpretable prompts (Shin et al., 2020; Shi et al., 2022) for white-box LLMs, where different gradient-guided optimization techniques are leveraged. The recent research efforts, instead, are more focused on considering the more practical setting of learning interpretable prompts for black-box LLMs (Prasad et al., 2022; Zhou et al., 2022; Pryzant et al., 2023; Xu et al., 2022; Guo et al., 2023; Pan et al., 2023; Chen et al., 2023), where the generating-then-selecting pipeline in Fig. 1 is often adopted.

Compared with previous investigations, this work additionally considers a limited budget, which we believe is a practical but under-investigated concern. Most of the previous methods perform selection based on evaluating prompts on all development data (Jiang et al., 2020; Xu et al., 2022; Guo et al., 2023; Prasad et al., 2022), which is unavoidably costly. APE (Zhou et al., 2022) briefly touched on the cost issue by proposing a naive iterative top filtering strategy (which however is suggested to have only marginal benefits). APO (Pryzant et al., 2023) tested a few MAB methods, including two classical BAI-FB designs, i.e., SH (Karnin et al., 2013) and SR (Audibert et al., 2009), and reported that the performance of UCB is the more favorable. However, it neither formally introduces the budget constraint nor provides a systematical connection to BAI-FB as in this work. Also, this work goes much deeper in incorporating the state-of-the-art BAI-FB designs (i.e., CR (Wang et al., 2023) and GSE (Azizi et al., 2023)) and proposing embedding-based enhancements. It is worth noting that INSTINCT (Lin et al., 2023) also incorporates one MAB method, i.e., NeuralUCB (Zhou et al., 2020), to prompt learning. However, Lin et al. (2023) requires a white-box LLM while NeuralUCB is designed for regret minimization (instead of best arm identification), which is not suitable for learning the optimal prompt.

Multi-armed Bandits. Here we briefly discuss the representative studies of MAB, with comprehensive surveys available in Bubeck et al. (2012); Lattimore & Szepesvári (2020). The target of learning in a MAB system can be roughly categorized as *regret minimization* and *best arm identification*. The regret minimization designs target achieving a desired balance between exploration and exploitation so that the cumulative rewards are maximized, e.g., UCB (Auer et al., 2002) and Thompson sampling (Thompson, 1933). The best arm identification (BAI) designs are fully focused on exploration and can be further divided into two classes: fixed-budget (BAI-FB) and fixed-confidence (BAI-FC). The BAI-FB setting maximizes the probability of finding the best arm with a limited number of pulls (Audibert et al., 2010; Karnin et al., 2013; Wang et al., 2023; Azizi et al., 2023; Atsidakou et al., 2022; Yang & Tan, 2022). The BAI-FC setting is a dual one which focuses on minimizing the overall number of pulls while guaranteeing that the probability of finding the best arm is higher than a threshold (Garivier & Kaufmann, 2016; Jamieson & Nowak, 2014; Soare et al., 2014; Jedra & Proutiere, 2020). Besides leveraging BAI-FB as in this work, it is imaginable that BAI-FC designs can also find applications in prompt learning, especially when valuing identification accuracy over incurred costs. While MAB has found wide success in different applications, this work marks the first time that a systematical connection between MAB and prompt learning has been established to the best of our knowledge.

B Details of the Adopted BAI-FB Designs

The details of `TRIPLE-SH` (inspired by Karnin et al. (2013)), `TRIPLE-CR` (inspired by Wang et al. (2023)), `TRIPLE-CLST`, and `TRIPLE-GSE` (inspired by Azizi et al. (2023)) can be found in Algs. 1, 2, 3, and 4, respectively. For `TRIPLE-CLST` and `TRIPLE-GSE`, we denote

$$e(p) := \texttt{embed}(p) \in \mathbb{R}^d, \;\; \mathcal{E} := \{e(p) : p \in \mathcal{P}\}.$$

We further illustrate the underlying BAI-FB designs in the following.

Sequential Halving (SH) (Karnin et al., 2013): SH is one of the first provably efficient BAI-FB designs and remains popular after a decade of its proposal. It follows a protocol that divides the total budget N into $\lceil \log_2(|\mathcal{P}|) \rceil$ equal-length phases. In each phase, SH uniformly tries all active prompts (initialized as $\mathcal{P}$) and eliminates half of them with the lower sample means for the next phase. The final active arm is output as the identified optimal prompt.

Continuously Reject (CR) (Wang et al., 2023): CR is a recently proposed method, which can be viewed as an extension of the classical Successively Reject (SR) design (Audibert et al., 2010). It uniformly explores active prompts (initialized as $\mathcal{P}$) and performs potential elimination of poorly-performed prompts after each pull. The elimination is based on carefully designed criteria using confidence bounds. It can be observed that, without the phased structure, CR is more adaptive than SH (and SR), which makes it appealing both theoretically and practically.

GSE (Azizi et al., 2023): It inherits the general phased elimination flow of SH. The major difference is that SH uses sample means to perform eliminations. GSE, on the other hand, leverages collected samples from previous phases to train a reward function $g_{\boldsymbol{\theta}}(\cdot) : \mathbb{R}^d \to \mathbb{R}$ that maps prompt embeddings to the predicted performance, which is further used to eliminate prompts.

Algorithm 1 `TRIPLE-SH`

Input: the pool of candidate prompts $\mathcal{P}$, budget N
Initialization: set $\hat{\mu}(p) \leftarrow$ for all $p \in \mathcal{P}$; set the active prompt set $\mathcal{A} \leftarrow \mathcal{P}$
for phase $p = 1, \cdots, \lceil \log_2(|\mathcal{P}|) \rceil$ **do**
  Interact with the targeted LLM using each prompt in $\mathcal{A}$ for $\lceil N/(|\mathcal{A}| \lceil \log_2(|\mathcal{P}|) \rceil) \rceil$ times
  Update the sample means $\{\hat{\mu}(p) : p \in \mathcal{A}\}$ using the collected samples
  Update the active prompt set $\mathcal{A}$ as the set of $\lceil \mathcal{A}/2 \rceil$ prompts in the original $\mathcal{A}$ with the highest $\hat{\mu}(p)$
end for
Output: the remaining active prompt $\hat{p}^*$

Algorithm 2 `TRIPLE-CR`

Input: the pool of candidate prompts $\mathcal{P}$, budget N
Initialization: set $n(p) \leftarrow 0$, $\hat{\mu}(p) \leftarrow$ for all $p \in \mathcal{P}$; set the active prompt set $\mathcal{A} \leftarrow \mathcal{P}$
for time $\tau = 1, \cdots, N$ **do**
  Receive input x_τ
  Select prompt $p_\tau \leftarrow \arg\min_{p \in \mathcal{A}} n(p)$
  Sample output $\hat{y}_\tau \sim f([p_\tau, x_\tau])$ from the targeted LLM
  Obtain score $s_\tau \leftarrow s(x_\tau, \hat{y}_\tau)$
  Update $\hat{\mu}(p_\tau) \leftarrow \frac{\hat{\mu}(p_\tau) n(p_\tau) + s_\tau}{n(p_\tau)+1}$ and $n(p_\tau) \leftarrow n(p_\tau) + 1$
  Compute $p' \leftarrow \arg\min_{p \in \mathcal{A}} \hat{\mu}(p)$ and $\delta_\tau \leftarrow \min_{p \in \mathcal{A} \setminus \{p'\}} \{\hat{\mu}(p) - \hat{\mu}(p')\}$
  if $\sqrt{\frac{N - \sum_{p \notin \mathcal{A}} n(p)}{\sum_{p \in \mathcal{A}} n(p) \log(|\mathcal{A}|)}} - 1 \leq \delta_\tau$ **then**
    Eliminate prompt p', i.e., $\mathcal{A} \leftarrow \mathcal{A} \setminus \{p'\}$
  end if
end for
Output: prompt $\hat{p}^* \leftarrow \arg\max_{p \in \mathcal{A}} \hat{\mu}(p)$

Algorithm 3 `TRIPLE-CLST`

Input: the pool of candidate prompts $\mathcal{P}$ and their embeddings $\mathcal{E}$, overall budget N, Phase I budget N_1, number of clusters L
Cluster $\mathcal{P}$ into clusters $\mathcal{C} = \{\mathcal{C}^1, \cdots, \mathcal{C}^L\}$ based on embeddings $\mathcal{E}$ (e.g., via k-means)
Obtain $\widehat{\mathcal{C}}^* \leftarrow$ BAI-FB$(\mathcal{C}, N_1)$ {*Phase I*}
Obtain $\hat{p}^* \leftarrow$ BAI-FB$(\widehat{\mathcal{C}}^*, N - N_1)$ {*Phase II*}
Output: prompt $\hat{p}^*$

Algorithm 4 TRIPLE-GSE

Input: the pool of candidate prompts $\mathcal{P}$ and their embeddings $\mathcal{E}$, budget N
Initialization: set $\hat{\mu}(p) \leftarrow$ for all $p \in \mathcal{P}$; set the active prompt set $\mathcal{A} \leftarrow \mathcal{P}$
for phase $p = 1, \cdots, \lceil \log_2(|\mathcal{P}|) \rceil$ **do**
  Interact with the targeted LLM using each prompt in $\mathcal{A}$ for $\lceil N/(|\mathcal{A}| \lceil \log_2(|\mathcal{P}|) \rceil) \rceil$ times
  Use the collected samples to train a function $g_{\boldsymbol{\theta}}(\cdot)$ parameterized by $\boldsymbol{\theta}$, e.g., a linear function or an MLP
  Compute $\{\hat{\mu}(p) = g_{\boldsymbol{\theta}}(e(p)) : p \in \mathcal{A}\}$
  Update the active prompt set $\mathcal{A}$ as the set of $\lceil \mathcal{A}/2 \rceil$ prompts in the original $\mathcal{A}$ with the highest $\hat{\mu}(p)$
end for
Output: the remaining active prompt $\hat{p}^*$

C Full Experimental Details

In this section, we include full details of our experiments, while the complete codes are also uploaded in the supplementary materials. Before further details, as we use chat-based LLMs, initial system instructions are needed, where the officially recommended system instructions are adopted in experiments, as shown in Fig. 3.

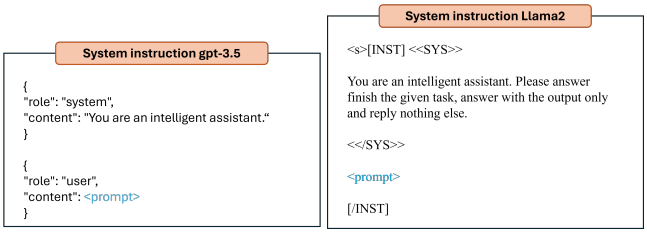

Figure 3: The adopted system instructions: gpt-3.5 (left) and llama2-7b (right)

C.1 Experiments with Fixed Prompt Pools and APE

The prompt generation process to obtain the fixed prompt pools largely follows the one in APE Zhou et al. (2022), i.e., demonstrating LLMs with examples. In particular, in the generation of each prompt, we sample 10 examples from the training set to demonstrate LLMs with two types of generation templates: 'forward' and 'backward', which are illustrated in Fig. 4. The same setups are also adopted in the end-to-end experiments with APE.

A side observation is that we find that in general, GPT-3.5 can handle both templates, resulting in reasonable prompts. However, Llama2-7b exhibits difficulties in generating useful prompts from the 'backward' template, possibly due to its more simplified structure.

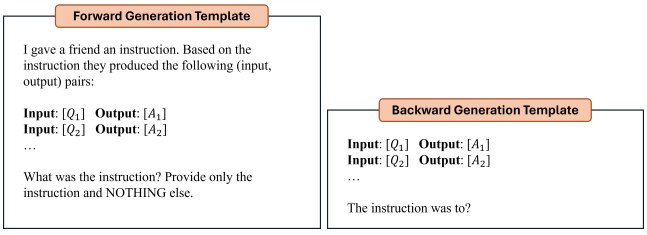

Figure 4: The adopted prompt generation templates for experiments with APE: forward (left) and backward (right)

C.2 EXPERIMENTS WITH APO

The APO framework (Pryzant et al., 2023) iteratively refines prompts based on feedback generated by LLMs. In particular, for each iteration, the system is set to identify {num_feedback} fault reasons (i.e., gradients) for the selected prompts from previously incorrectly answered examples. Then, with the selected prompts and the identified fault reasons, the LLM is instructed to create {num_prompts} new prompts for further selection. The adopted templates in our experiments are shown in Fig. 5, where we set {num_feedback} to 2 and {num_prompts} to 5. We believe this configuration ensures that each iteration not only effectively identifies key areas of improvement but also sufficiently expands the pool of potential prompts.

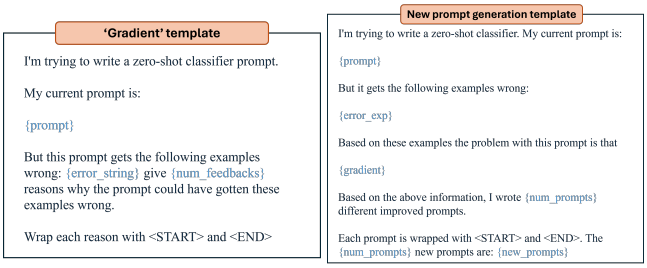

Figure 5: The adopted templates for experiments with APO (Pryzant et al., 2023): fault identification (i.e., "gradient") (left) and new prompt generation (right).

C.3 IMPLEMENTIONS OF TRIPLE-CLST AND TRIPLE-GSE

Obtaining Embedding. A critical component of both TRIPLE-CLST and TRIPLE-GSE is the extraction of sentence embeddings for the candidate prompts. In our experiments, the prompts are first tokenized using the cl100k_base tokenizer. Then, the tokenized prompts are input into the text-embedding-ada-002 model (OpenAI, 2023b), converting them into continuous vectors.

TRIPLE-CLST. In experiments with TRIPLE-CLST, the number of clusters is set as $L = \lceil \sqrt{|\mathcal{P}|} \rceil$ and a third of our total budget is allocated for the initial phase, i.e., $N_1 = N/3$. The k-means algorithm is employed as the clustering method. For more stable performance, Phase I is configured to find the top $L/2$ clusters, instead of the optimal one, which safeguards against the situation that the optimal prompt is not located in the optimal cluster. Also, for the BAI-FB designs in both phases, the CR algorithm (Wang et al., 2023) is adopted due to its flexibility.

TRIPLE-GSE. The OpenAI embedding API returns embeddings of 1536 dimensions, which can be challenging for learning with limited samples. To overcome this issue, in the implementation of TRIPLE-GSE, we first employ a projection to 64 dimensions using a matrix with random elements from the standard normal distribution. This technique is also incorporated in Chen et al. (2023) and is particularly beneficial given our limited budget constraints. Furthermore, to avoid overfitting and convergence issues, we adopt the standard approach by dividing our interaction data into training (80%) and validation (20%) sets. The prompt elimination process on line 7 in Alg. 4 is performed only if the mean squared error on the validation set is sufficiently low and we set this error threshold at 0.1 in our experiments.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 EXAMPLES OF CLUSTERING

The effectiveness of TRIPLE-CLST relies on the clustering results produced in Phase I. Ideally, prompts with similar performances should be clustered together. In this way, Phase I can quickly eliminate the prompts with poor performances, leaving a small pool of good prompts for Phase II to process. In the experiments, this intuitive phenomenon is indeed observed. Here we provide examples in Fig. 6, where as expected, prompts in the same cluster share similar performances.

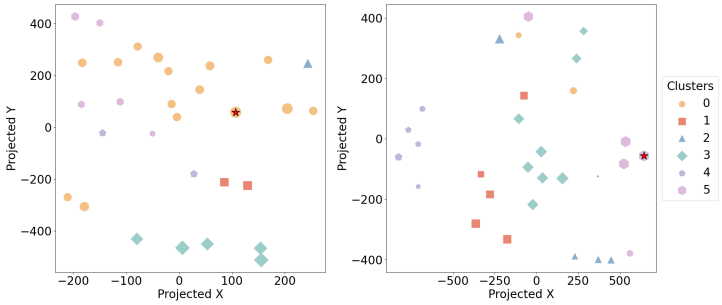

Figure 6: Clusters for 30 prompts for "movie_recommendation" (left) from BigBench (Suzgun et al., 2022) and "rhymes" (right) from Instruction Induction (Honovich et al., 2022). Prompts in the same cluster are labeled by the same color and shape. The performance of each prompt is represented by the size of its shape (the larger the better). It can be observed that the prompts in the same cluster (i.e., the same color and shape) share similar performance (i.e., similar sizes). Especiall, the optimal prompt (marked by the red star) is clustered together with a few prompts with comparably near-optimal performances. The embeddings are projected to 2 dimensions using T-SNE Hinton & Roweis (2002).

D.2 Impact of the total budget.

For a more comprehensive understanding, using candidate pools with 30 prompts, we further examine the impact of budgets, starting with 5 evaluations per prompt on average (i.e., 150 overall as adopted in Fig. 2 (middle)), and then gradually increasing to 30 (i.e., 900 overall, which is the same as the experiments in Zhou et al. (2022)). From the results shown in Fig. 7 (left), we see that the improvements of TRIPLE over baselines are more pronounced with lower budgets. In particular, with a budget of 10 evaluations per prompt on average, TRIPLE-CR, TRIPLE-CLST and TRIPLE-GSE maintain notable 9.7%, 13.5% and 17.4% improvement over Uniform, respectively; when the budget escalates to 20 evaluations per prompt on average, TRIPLE-CLST and TRIPLE-GSE still achieve an approximate 8% improvement. Once the budget reaches 30 evaluations per prompt on average, all methods provide approximately the same performance as they can all identify the optimal prompts under this generous budget.

To further guide practical implementation, we additionally investigate how to select a reasonable budget. In particular, we focus on the efficiency of various prompt selection algorithms in identifying a "good" prompt – either the optimal prompt in the pool or achieving 95% or better of the optimal prompt's performance. Fig. 7 (right) illustrates that initial increases in budgets significantly improve the probability of identifying a good prompt, but this benefit tapers off with further budget expansion. This finding suggests that starting with a modest budget and incrementally increasing it is the more effective approach, stopping when additional investment no longer translates into significant returns.

D.3 Complete Results on 47 Tasks

In the main paper, we provide experimental results on 12 representative tasks from the overall 47 available tasks. In the following, the complete results are discussed.

Instruction-Induction. Results on the 24 available tasks in the Instruction-Induction dataset (Honovich et al., 2022) are illustrated in Fig. 8 (small prompt pools $|\mathcal{P}| = 10$ with gpt-3.5), 9 (large prompt pools $|\mathcal{P}| = 30$ with gpt-3.5), and 10 (large prompt pools $|\mathcal{P}| = 30$ with llam2-7b). It can be observed that TRIPLE methods improve Uniform on most tasks while achieving matching performance on the remaining ones.

BigBench-ii. Results on the 23 available tasks in the BigBench-ii dataset (Suzgun et al., 2022) are reported in Figs. 11 (small prompt pools $|\mathcal{P}| = 10$ with gpt-3.5), 12 (large prompt pools $|\mathcal{P}| = 30$ with gpt-3.5), and 13 (large prompt pools $|\mathcal{P}| = 30$ with Llam2-7b). It can still be observed that TRIPLE methods consistently surpass or at least match the performance of baselines.

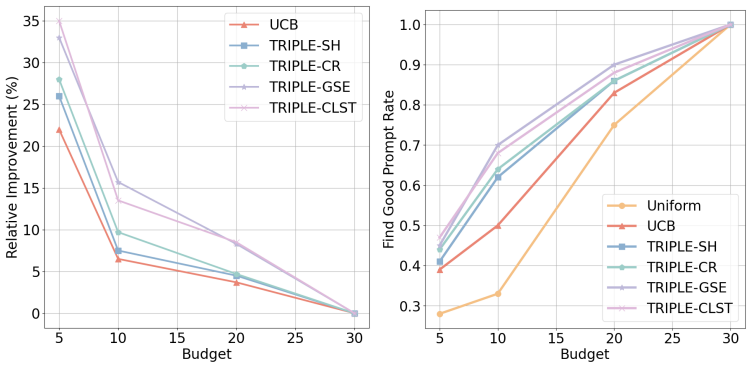

Figure 7: Relative gain of over Uniform under different budgets (left). Probability for different algorithms to select a good prompt under different budgets (right). All results are collected with gpt-3.5 and averaged over 5 runs.

D.4 Best Prompt Identification Frequency

Besides the average return, another key aspect of prompt selection is the frequency of selecting a good prompt. In Table 3, we further demonstrate the best prompt identification frequency of different algorithms across 20 selected tasks from 5 independent runs.

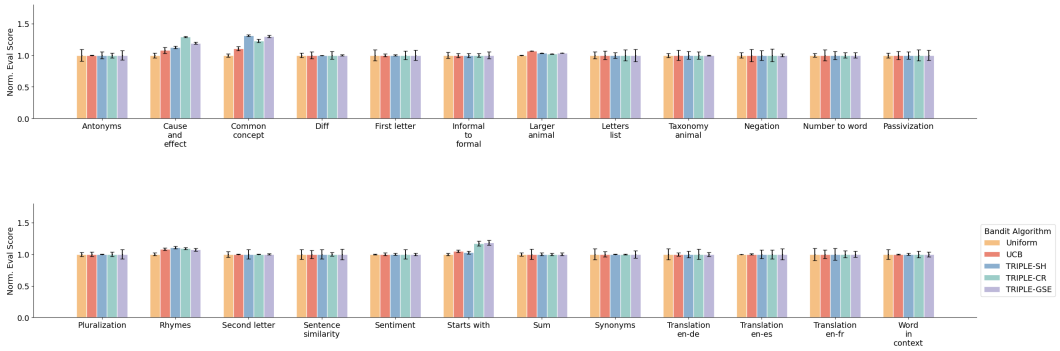

Figure 8: Complete results with small candidate pools $|\mathcal{P}| = 10$ and gpt-3.5 on the Instruction-Induction dataset under a budget of 50.

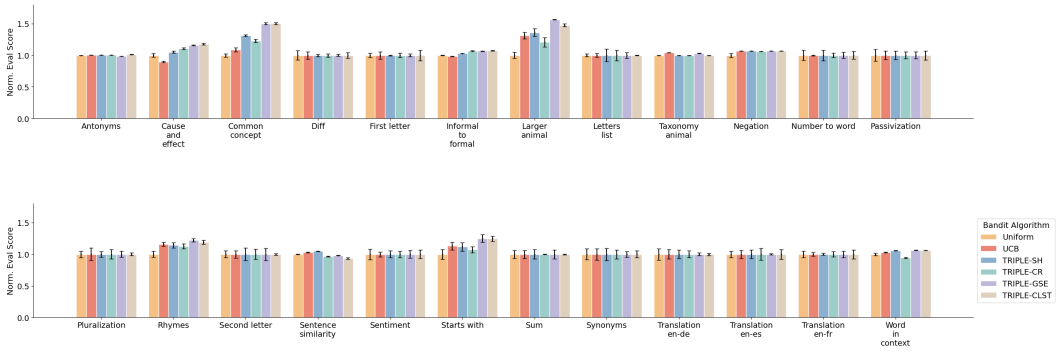

Figure 9: Complete results with large candidate pools $|\mathcal{P}| = 30$ and gpt-3.5 on the Instruction-Induction dataset under a budget of 150.

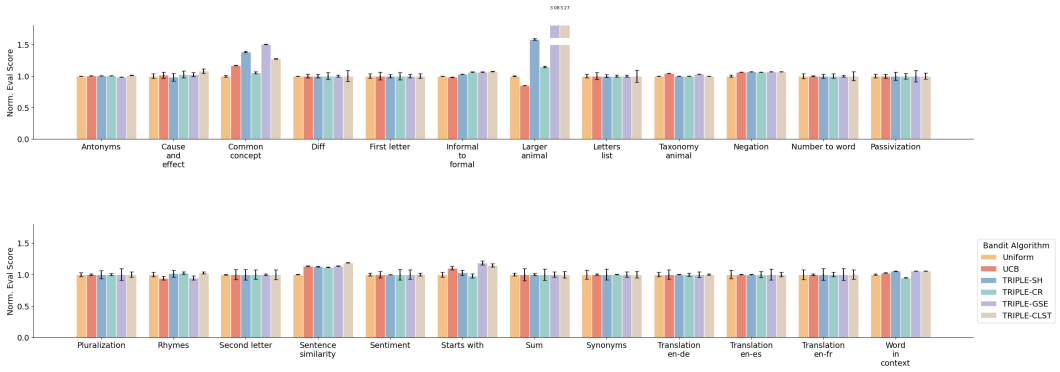

Figure 10: Complete results with large candidate pools $|\mathcal{P}| = 30$ and llama-7b on the Instruction-Induction dataset under a budget of 150.

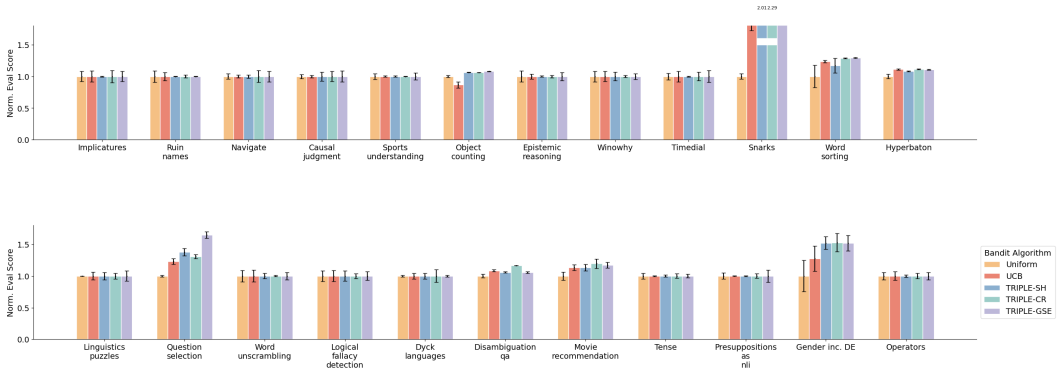

Figure 11: Complete results with small candidate pools $|\mathcal{P}| = 10$ and gpt-3.5 on the BigBench-ii dataset under a budget of 50.

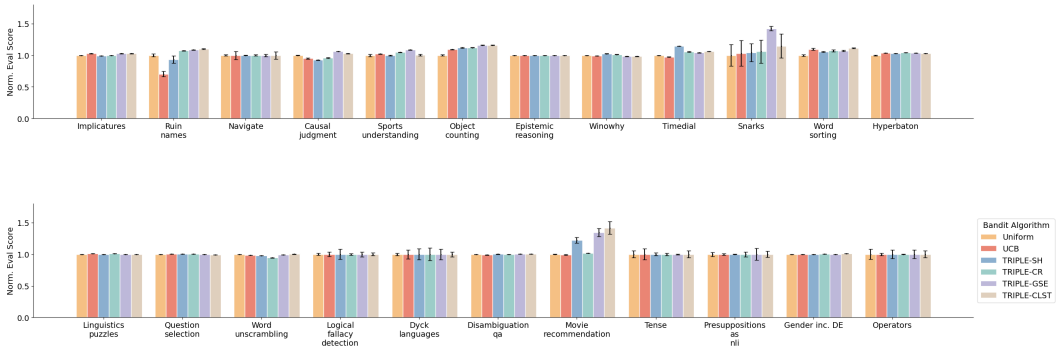

Figure 12: Complete results with large candidate pools $|\mathcal{P}| = 10$ and gpt-3.5 on the BigBench-ii dataset under a budget of 150.

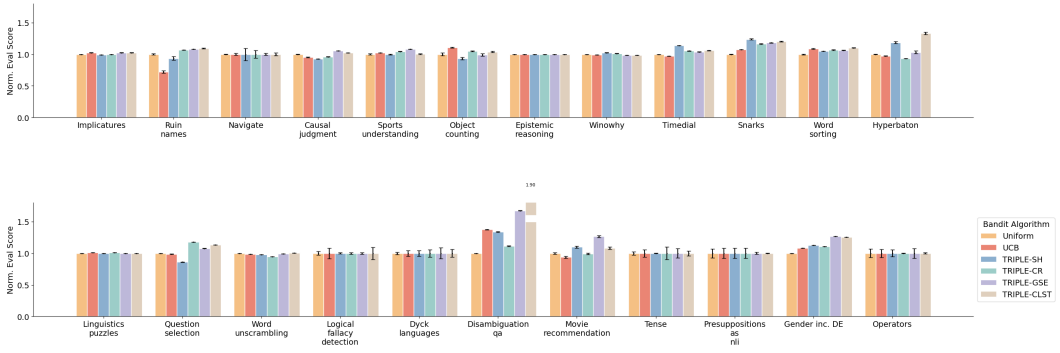

Figure 13: Complete results with large candidate pools $|\mathcal{P}| = 30$ and llama-7b on the BigBench-ii dataset under a budget of 150.

Table 3: The ratios of different methods outputting a good prompt with gpt-3.5 from large prompt pools $|\mathcal{P}| = 30$.

Task	Budget	**Uniform** (%)	**UCB**	**SH**	**CR**	**CLST**	**GSE**
Cause and effect	5	20	20	20	30	60	30
	10	20	20	40	40	80	40
	20	60	60	80	80	100	80
Common concept	5	0	0	0	0	20	40
	10	20	20	20	20	60	40
	20	40	40	80	80	80	80
Larger animal	5	80	80	100	100	100	100
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
Informal to formal	5	0	0	0	35	25	30
	10	20	20	20	60	40	40
	20	60	60	80	100	100	100
Negation	5	90	100	100	100	100	100
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
Rhymes	5	10	10	30	20	40	30
	10	40	40	60	60	80	80
	20	100	100	100	100	100	100
Orthography starts with	5	30	40	40	20	40	40
	10	60	60	80	80	80	80
	20	100	100	100	100	100	100
Sentence similarity	5	25	30	40	25	55	45
	10	40	40	60	60	60	80
	20	60	60	80	80	80	100
Word in context	5	55	55	70	60	70	70
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
Disambiguation qa	5	80	90	100	100	90	100
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
Gender Inc. DE	5	40	60	70	80	100	80
	10	60	80	100	100	100	100
	20	100	100	100	100	100	100
Hyperbaton	5	65	70	70	70	90	80
	10	80	80	80	80	100	100
	20	100	100	100	100	100	100
Movie recommendation	5	20	20	25	45	50	40
	10	40	40	40	60	60	60
	20	60	60	80	80	80	80
Object counting	5	10	20	25	30	35	35
	10	20	40	40	60	60	60
	20	80	100	100	100	100	100
Question selection	5	0	0	10	0	20	15
	10	20	20	20	20	40	20
	20	40	40	40	60	80	60
Snarks	5	0	20	10	25	25	20
	10	20	40	40	60	60	60
	20	80	100	100	100	100	100
Word sorting	5	55	70	80	80	75	80
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
Ruin names	5	35	55	70	75	70	80
	10	60	80	100	100	100	100
	20	100	100	100	100	100	100
Sporting understanding	5	75	80	80	80	80	90
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100
Word unscrambling	5	80	85	90	90	85	90
	10	100	100	100	100	100	100
	20	100	100	100	100	100	100