SOPL: A Sequential Optimal Learning Approach to Automated Prompt Engineering in Large Language Models

Shuyang Wang[1], **Somayeh Moazeni**[2], **Diego Klabjan**[1]
[1]Northwestern University, [2]Stevens Institute of Technology
Correspondence: shuyangwang2025@u.northwestern.edu

Abstract

Designing effective prompts is essential to guiding large language models (LLMs) toward desired responses. Automated prompt engineering aims to reduce reliance on manual efforts by streamlining the design, refinement, and optimization of natural language prompts. This paper proposes an optimal learning framework for automated prompt engineering for black-box models, designed to sequentially identify effective prompt features under limited evaluation budgets. We introduce a feature-based method to express prompt templates, which significantly broadens the search space. Bayesian regression is employed to utilize correlations among similar prompts, accelerating the learning process. To efficiently explore the large space of prompt features, we adopt the forward-looking Knowledge-Gradient (KG) policy for sequential optimal learning efficiently by solving mixed-integer second-order cone optimization problems, making it scalable and capable of accommodating prompts characterized only through constraints. Our method significantly outperforms a set of benchmark strategies assessed on instruction induction tasks within limited iterations of prompt evaluations, showing the potential of optimal learning for efficient prompt learning.

1 Introduction

Recent efforts in automated prompt engineering have primarily focused on iterative evaluation and refinement to converge to ideal prompts (Guo et al., 2024; Zhou et al., 2023; Pryzant et al., 2023; Prasad et al., 2023; Shin et al., 2020). These methods often assume the availability of numerous iterations. However, in many real-world scenarios, opportunities to evaluate prompts are limited, as each prompt evaluation can be costly or time-consuming. For example, a company may use LLMs to generate blog posts promoting a new product. The narrative of each post may vary depending on the target audience such as lawyers, computer engineers, or educators, and the choice of underlying advertisements included. The LLM prompt should be carefully designed to incorporate these key aspects. The post is then evaluated based on user click-through rates. Each evaluation is expensive as it requires users to visit the website and interact with the content. In such cases, the number of prompts that can be tested is very limited.

Moreover, many existing approaches search over a set of precrafted candidate prompts for ideal prompts (Shi et al., 2024; Zhou et al., 2023; Luo et al., 2023). These methods require identifying and enumerating a set of prompts, which restricts the scalability as the candidate set expands. They also fail to utilize the correlation among similar prompts to expedite learning.

To unlock the potential of LLMs in diverse scenarios, it is crucial to develop an automated prompting framework for black-box LLMs that is capable of capturing dependencies among prompts and efficiently identifying high-performing prompts within few evaluations. This paper presents a principled forward-looking iterative process for automated prompt engineering through the optimal design of a sequence of prompts.

We propose an interpretable feature-based approach to prompt representation. Various categorical or numerical features can be considered to characterize detailed aspects of a prompt, such as the selection and ordering of demonstrative examples. A recent survey (Dong et al., 2024) summarizes existing methods for organizing and selecting demonstrative examples, and Xu et al. (2024) propose a method for ordering in-context samples based on label distributions. Previous works identify factors within a prompt that influence the LLM outputs but often treat these aspects in isolation. We allow for capturing various interactions among prompt attributes and can accommodate potential constraints on the features. This feature-based prompt rep-

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 21172–21185
November 4-9, 2025

resentation enables the inclusion and exploration of a vast and diverse set of prompts, which does not require manual prompt provision. In addition, in contrast to prompt descriptions based on embedding vectors, our representation is inherently interpretable.

We adopt a Bayesian approach to refine beliefs about the influence of prompt features on the LLM response. It supports the integration of prior knowledge and user opinions and enables the capture of feature correlations. To operationalize this, we define a probabilistic model to link prompt features to a response quality of interest. In this paper, we demonstrate our approach using LLM response accuracy as the primary evaluation metric.

We formalize the iterative process of automated prompt engineering for black-box LLMs under a limited number of prompt evaluations as a sequential decision-making problem. Given limited opportunities for prompt evaluation, this problem falls into the category of finite-horizon discrete-time Markov decision processes; see (Puterman, 2014). An optimal learning policy sequentially selects a feasible prompt representation for evaluation, aiming to maximize the expected outcome of the final prompt. Due to the potentially large prompt feature space, the curse of dimensionality (Powell, 2022) hinders the exact computation of the optimal prompt selection. We adopt an approximate policy for the optimal learning problems, known as the expected improvement policy in (Chick, 2006; Chick et al., 2010) or Knowledge-Gradient (KG) policy in (Frazier et al., 2009; Powell and Ryzhov, 2012). This is a forward-looking policy that maximizes the expected improvement in the value of information in each learning phase. The KG policy often excels in practical scenarios with limited evaluation budgets, outperforming common static data acquisition strategies and dynamic test-and-learn policies.

The large space of prompt candidates, defined by constrained features, makes it impractical to enumerate all feasible alternatives for determining KG decisions. To address this challenge, we leverage recent advancements in scalable optimal learning and KG computation. In contrast to earlier results to compute KG decisions (Frazier et al., 2009) based on enumeration of all feasible alternatives, recent computational methods (Moazeni et al., 2020; Han et al., 2016; Defourny et al., 2015) build on optimal quantization of the response probability followed by mixed-integer conic optimization reformulations to leverage efficient optimization solution methods. For optimal learning problems with larger feature spaces, an iterative process involving solving mixed-integer linear optimization problems is employed to achieve even greater computational efficiency and scalability.

Our approach requires users to predefine a set of features and corresponding choices to construct the prompt search space. However, these manual efforts can be substantially reduced by leveraging LLMs to generate a set of choices for each feature. Given the capability of the KG policy, our method can efficiently explore the resulting feature spaces.

We assess the performance of sequential prompt learning with adaptive prompt selection policies on a variety of instances of instruction induction tasks (Honovich et al., 2023), designed for LLMs to deduce implicit tasks or instructions. We use the accuracy of responses from GPT-3.5 on the validation data as the primary performance metric.

We evaluate the performance of the KG policy in comparison with the adaptive myopic policy, the increasingly popular Thompson sampling policy, and other automated prompt engineering methods such as EvoPrompt (Guo et al., 2024) using an evolutionary algorithm to refine prompts and TRIPLE (Shi et al., 2024) using a multi-armed bandit approach to select prompts. Our results show that the KG policy consistently identifies high-quality prompts within 30 or fewer evaluations, outperforming benchmark methods. Furthermore, KG is particularly effective for challenging tasks with high uncertainty and sensitivity to prompt variations, demonstrating its potential for scalable, cost-efficient automated prompt engineering. Our code is available [1] and data details are provided in the Appendix.

Our contributions can be summarized as follows.

- We introduce a sequential optimal learning framework for automated prompt engineering to guide through the process of designing a sequence of prompts that effectively elicit accurate responses from an LLM. The approach is compatible with black-box LLMs and is particularly effective for applications where prompt evaluation is resource-intensive.

- We propose a feature-based approach to represent language prompts, which greatly expands the prompt search space. A link function maps the features to LLM response accuracy through Bayesian model parameters

[1] https://github.com/shuyangwang2025/SOPL

to leverage correlations among prompts with shared characteristics. Our method enables simultaneous optimization of multiple features to generate improved prompts.

- We leverage the KG policy within our sequential prompt learning to efficiently identify high-performing prompts in large prompt spaces. The KG policy outperforms various benchmarks, especially for challenging tasks with high uncertainty in LLM response.

The paper is organized as follows. Section 2 reviews the related literature. Section 3 formalizes the generic iterative process for automated prompt learning. Section 4 discusses the forward-looking KG policy for prompt selection. Illustrative examples and computational results are provided in Section 5. We conclude the paper in Section 6.

2 Related Work

Generation-then-selection methods Zhou et al. (2023) present a two-phase pipeline that generates a set of candidate instructions, which are evaluated and filtered based on their performance on the downstream tasks until the best one from the candidate set is found. Luo et al. (2023) systematize the process of iteratively updating and selecting from a set of candidate prompts by using an optimal control paradigm. However, these approaches are limited to search spaces represented by precomputed prompt candidates, while our framework encompasses different prompts generated dynamically and various exploration policies. Shi et al. (2024) also adopt a two-phase approach, modeling prompt selection as a multi-armed bandit problem and applying the continuously reject algorithm. Their method requires increasing evaluations as the candidate set grows, whereas our KG policy scales efficiently with larger spaces of prompts.

Edit-based methods Several approaches refine prompts by iteratively editing base prompts. GrIPS (Prasad et al., 2023) applies local text edits, while (Guo et al., 2024) and (Fernando et al., 2024) use evolutionary algorithms with mutations and crossover operations performed by an LLM. Jin et al. (2024) propose an iterative prompt refinement scheme specially crafted for relevance ranking in information retrieval. These methods rely on local search by modifying existing prompts. Our method, however, performs forward-looking exploration using Bayesian optimal learning.

Prompt learning methods Recent studies (Chen et al., 2024; Lin et al., 2024) apply Bayesian Optimization to explore the embedding space of prompts, requiring white-box access to LLMs, while we directly search in the space of discrete prompts and eliminate the need for white-box LLMs. RLPrompt (Deng et al., 2022) formulates prompt construction as a reinforcement learning problem, which requires extensive evaluations during training. In contrast, our framework learns efficiently under limited evaluation budgets. Recent surveys provide comprehensive taxonomies of existing methods for automated prompt optimization (Li et al., 2025; Ramnath et al., 2025). Our work complements the literature by introducing a sequential optimal learning framework that leverages the value of information for efficient prompt engineering.

3 Sequential Optimal Prompt Learning

We introduce a sequential optimal prompt learning framework, called *SOPL*, for automated prompt engineering that is compatible with black-box LLMs and generates human-readable prompts, while addressing the challenge of limited number of iterations for prompt evaluation. Our framework, depicted in Figure 1, follows the iterative process outlined in Algorithm 1.

We illustrate our method with examples from the instruction induction dataset (Honovich et al., 2023), however, the framework is general and applicable to a wide variety of tasks. As an example task, the objective is to identify the most effective instruction for guiding an LLM to infer the common theme among a given list of items. Prompt effectiveness is measured by the accuracy on a labeled dataset, which contains input paired with their ground truth common themes. For example, given the input "lawyers, basketball, walled yards," the corresponding label "involves courts" captures their common concept. A simple instruction such as "Find the common theme among the input words" may serve as an initial guess. However, exhaustively guessing and testing all possible variations quickly becomes intractable without a systematic way to represent prompts and leverage information from prior attempts.

3.1 Feature-Based Prompt Representation

We adopt a feature-based approach to represent textual prompts, using a feature vector to capture their

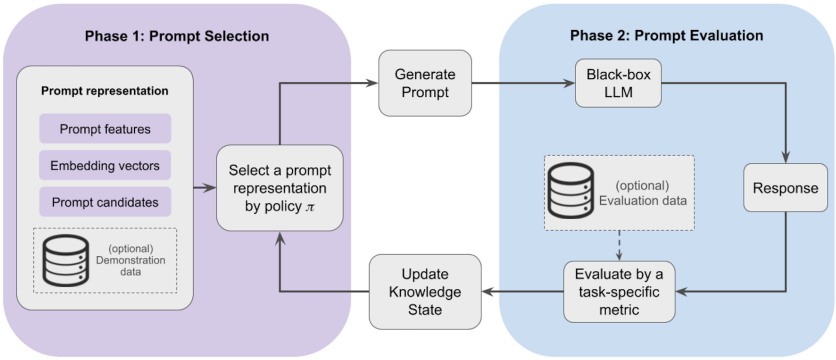

Figure 1: SOPL: Sequential optimal prompt learning for automated prompt engineering.

Algorithm 1 Sequential optimal prompt learning

Require Maximum iteration N, prompt representation selection policy $\pi\colon \mathcal{S} \to \mathcal{X}$, score function $\mathrm{Eval}\colon \mathcal{X} \to (0,1)$, Bayesian update function $\mathrm{Update}\colon \mathcal{S} \times \mathbb{R} \to \mathcal{S}$ that implements (3)-(6).

Initialize knowledge state S.
Initialize best prompt so far x^*.
Initialize best score so far $u^* \leftarrow -1$.
for step $n = 1, ..., N$ **do**
  Select a prompt representation $x \leftarrow \pi(S)$.
  Generate a prompt and evaluate the score $u_x \leftarrow \mathrm{Eval}(x)$.
  Clip the score for numerical stability $u_x \leftarrow \max(\min(u_x, 0.999), 0.001)$.
  Update the knowledge state $S \leftarrow \mathrm{Update}(S, \mathrm{logit}(u_x))$.
  if $u_x > u^*$ **then**
    Record best score so far $u^* \leftarrow u_x$.
    Record best prompt so far $x^* \leftarrow x$.
  end if
end for
return x^*.

content and structural characteristics. Given a specific feature configuration, for instruction induction tasks, we create a meta prompt incorporating the selected feature values, and use an LLM to generate variations of candidate instructions.

Prompt features can be manually engineered by the user or derived from established prompt templates in the literature. For instruction induction tasks, we focus on five prompt features that have been shown to impact LLM responses, with a set of predefined choices for each, as summarized in Table 1. To generate a candidate instruction, we

[ROLE1] gave a **[ROLE2]** an instruction and inputs. The **[ROLE2]** read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

Input: **[INPUT 1]**
Output: **[OUTPUT 1]**
...
Input: **[INPUT 5]**
Output: **[OUTPUT 5]**

Generate a **[DESCRIPTION]** instruction. The instruction was ___

Figure 2: Meta prompt template 1.

first select a feature vector, for example,

$$f := \begin{bmatrix} \text{Template 1} \\ \text{example 1, example 2} \\ \text{“Professor \& student”} \\ \text{“simple”} \\ \text{No paraphrasing} \end{bmatrix}.$$

We then convert the categorical feature vector f into a binary vector x using one-hot encoding and refer to x as the prompt representation. Next, we construct a meta prompt by inserting the selected words in the corresponding fields of the chosen meta prompt template, as shown in Figure 3. The meta prompt guides an LLM to create a textual instruction that reflects the features encoded in x. Different combinations of the features yield different meta prompts, and ultimately, different candidate instructions. The goal is to find an optimal feature configuration, consisting of the template and word selections, that produces an instruction with the highest accuracy on the labeled dataset.

Previous studies have examined each feature in isolation, whereas we integrate these features in the prompt representation to leverage the synergies that emerge from their combination. In addition, our method also accommodates potential constraints on the features, such as mutually exclusive features, conditional features, and multiple-choice decisions,

Feature	Choices
Meta prompt template (Zhou et al., 2023)	Four meta prompt templates; one example is shown in Figure 2, and the others are shown in Appendix A.
Demonstrative examples (Lu et al., 2022; Zhao et al., 2021)	Examples of input-output pairs sampled from the demonstration dataset.
Roles (Wu et al., 2023; Kong et al., 2024)	(Scientist, research assistant), (Professor, PhD student), (Mom, kid), (Programmer, AI system), (Manager, employee), (I, friend), (Director, actor), (Coach, athlete), (Chef, sous chef).
Paraphrasing (Deng et al., 2023; Zhou et al., 2023)	Binary: if paraphrasing is enabled, we prompt the LLM to generate a variant of the instruction; see details in Appendix A.
Description (Li et al., 2023)	(empty), clear, detailed, simple, complex, precise, ambiguous, technical, expository, conceptual, authoritative, friendly, formal, informal, encouraging, stern, rude, assertive, humorous.

Table 1: Prompt features used for instruction induction tasks.

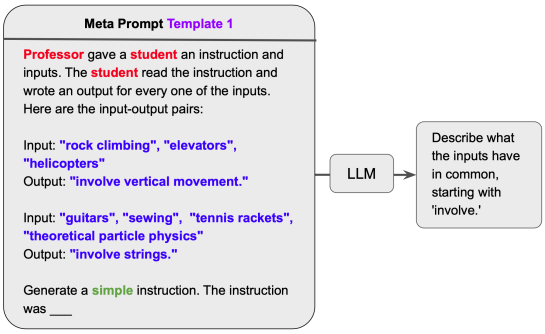

Figure 3: The meta prompt created from the selected feature vector and the generated candidate instruction.

which can be formalized as linear inequality or equality constraints on the vector x. The set of feasible feature combinations, represented by vectors x that satisfy the constraints, forms a diverse and potentially large search space, denoted by $\mathcal{X}$.

Our framework encompasses existing approaches as special cases. For example, the methods (Zhou et al., 2023; Shi et al., 2024; Luo et al., 2023) that select from a set of pregenerated prompt candidates can be viewed as using the candidate set as $\mathcal{X}$, where each x represents one candidate.

3.2 Prompt Evaluation

We measure the effectiveness of the prompt generated from x by a numeric score u_x, which is computed by a task-specific score function Eval. In our example where labeled data are available, the score function Eval computes the accuracy, i.e., the percentage of LLM responses that exactly match the ground truth labels as the score u_x.

For common metrics such as accuracy, F1-score, and point-wise mutual information (Bouma, 2009), the score lies in the interval between 0 and 1. We assume that

$$\mathrm{logit}(u_x) = \Theta^\top x + \epsilon, \tag{1}$$

where $\Theta \sim \mathcal{N}(\mu_\Theta, \Sigma_\Theta/\rho)$ is the D-dimensional model parameter and $\epsilon \sim \mathcal{N}(0, 1/\rho)$ with variance $1/\rho$ is the measurement noise. The mean μ_Θ and the precision ρ are the unknown parameters to estimate. For general score functions with ranges beyond 0 to 1, alternative link functions can be used to replace the logit function in (1).

3.3 Knowledge State Update

Bayesian learning addresses uncertainty in the effectiveness of prompt features by modeling multiple levels of randomness and correlation through prior distributions for model parameters. It can also incorporate existing knowledge and user input into these prior probability distributions. We adopt the Bayesian framework with a multivariate normal prior for the coefficients μ_Θ and a Gamma prior for the precision ρ, i.e.,

$$\mu_\Theta|\rho \sim \mathcal{N}\left(\theta, \Sigma/\rho\right), \rho \sim \mathrm{Gamma}(a, b). \tag{2}$$

The belief $S = (\theta, \Sigma, a, b)$, referred to as the *knowledge state*, encodes prior observations of prompt performance and informs future decisions. Vector Θ denotes the model parameter, and θ denotes the mean of the distribution that describes the model parameter distribution. The multivariate distribution (2) captures dependencies among

unknown parameters, implying that learning about one prompt provides insights into the effectiveness of others, thereby accelerating learning.

We iteratively update the knowledge state based on observed responses. At the n-th iteration, with knowledge state $S_n = (\theta_n, \Sigma_n, a_n, b_n)$ and selected prompt representation x_n, we observe score $u_n := u_{x_n}$ and update the knowledge state using Bayesian updating equations for the Normal-Gamma model as follows.

$$\theta_{n+1} = \theta_n + \frac{\text{logit}(u_n) - \theta_n^\top x_n}{(1 + x_t^\top (\Sigma_n + \Sigma_\Theta) x_n)} \Sigma_n x_n \tag{3}$$

$$\Sigma_{n+1} = \Sigma_n - \frac{\Sigma_n x_n x_n^\top \Sigma_n}{1 + x_n^\top (\Sigma_n + \Sigma_\Theta) x_n} \tag{4}$$

$$a_{n+1} = a_n + \frac{1}{2} \tag{5}$$

$$b_{n+1} = b_n + \frac{(\text{logit}(u_n) - \theta_n^\top x_n)^2}{2(1 + x_n^\top (\Sigma_n + \Sigma_\Theta) x_n)} \tag{6}$$

Matrix Σ_Θ is the scaled covariance matrix of the model parameter Θ as in (1). Equations (3)-(6) collectively define the function $\text{Update}(S, \text{logit}(x))$ in line 8 of Algorithm 1.

3.4 Prompt Representation Selection Policy

For each iteration, a prompt representation x_n is selected according to a policy π, aiming to maximize the test score on the downstream task after N iterations of evaluation.

Our framework allows for different prompt representation selection policies to explore the feasible prompt space $\mathcal{X}$ and update the knowledge state, including heuristic policies such as adaptive myopic (Greedy) and Thompson sampling (TS). The Greedy policy selects the best x based on the current knowledge state by

$$\pi^{Greedy}(S_n) := \underset{x \in \mathcal{X}}{\text{argmax}}\ \theta_n^\top x. \tag{7}$$

The TS policy samples from the posteriors of the parameters by $\widehat{\rho} \sim \text{Gamma}(a_n, b_n)$, $\widehat{\theta} \sim \mathcal{N}(\theta_n, \Sigma_n/\widehat{\rho})$, and then selects x by

$$\pi^{TS}(S_n) := \underset{x \in \mathcal{X}}{\text{argmax}}\ \widehat{\theta}^\top x. \tag{8}$$

Both policies are adaptive, but they are not forward-looking as they do not explicitly take into account the effect of selected prompts on subsequent prompt selections and overall learning process.

With a limited budget of N, sequential optimal prompt learning can be formulated as a finite-horizon Markov decision process, where the action space $\mathcal{X}$ consists of prompt representations and the state space $\mathcal{S}$ consists of knowledge states. Next, we discuss an approximate solution for the optimal prompt representation selection policy.

4 KG Prompt Selection Policy

We consider a forward-looking optimal learning policy designed to maximize the expected improvement in an approximated value of information during each iteration. This approach, known as the Knowledge-Gradient (KG) policy, offers an approximate solution to the MDP for prompt selection. For additional discussion and analysis, refer to (Gupta and Miescke, 1996; Chick et al., 2010; Frazier et al., 2008; Powell and Ryzhov, 2012). The value of information is measured by the expected single-period improvement, i.e., the difference between the values of the knowledge states S_{n+1} and S_n if the prompt representation $x_{n+1} = x$ is selected. For KG, we assume that x is binary. Hence, at iteration n, the following KG quantity is maximized:

$$\nu_x^n := \mathbb{E}\left[V_N(S_{n+1}) | S_n, x\right] - V_N(S_n), \tag{9}$$

where $V_N(S)$ is the value of the optimal policy at iteration N for any knowledge state S, i.e., $V_N(S) = \max_{x \in \mathcal{X}} \mathbb{E}[\eta_x | S]$, where $\eta_x := \text{logit}(u_x)$ is based on (1). Recall that S_{n+1} is the transition from state S_n induced by the updating procedure in (3)-(6). The quantity ν_x^n is the marginal value of one more prompt representation x being queried. Its value is always nonnegative.

The decision of the KG policy selects the prompt representation that maximizes the KG quantity in (9)[2]:

$$\pi^{KG}(S_n) := \underset{x \in \mathcal{X}}{\text{argmax}}\ \nu_x^n. \tag{10}$$

According to Proposition 5 in (Moazeni et al., 2020), for any $x \in \mathcal{X}$, the KG quantity corresponding to model (1) is given by

$$\nu_x^n = \mathbb{E}[\max_{y \in \mathcal{X}} (p_y^n + q_y^n(x) T_{2a_n}) | S_n] - \max_{y \in \mathcal{X}} p_y^n \tag{11}$$

[2]For discussion of the related concepts of asymptotic optimality and statistical consistency of the KG policy, the reader is referred to (Frazier et al., 2008; Frazier and Powell, 2011) when $\mathcal{X}$ is specified in enumerative form, and to (Han et al., 2016) when $\mathcal{X}$ is represented in constraint-based form.

where T_{2a_n} follows a Student's t-distribution with $2a_n$ degrees of freedom, and

$$p_y^n = \theta^\top y \tag{12}$$

$$q_y^n(x) = \sqrt{\frac{b_n}{a_n(1 + x^\top(\Sigma_n + \Sigma_\Theta)x)}} x^\top \Sigma_n y \tag{13}$$

The expectation in (11) is with respect to the one-dimensional random variable T_{2a_n}.

The first term in the KG quantity in (11) can be approximated by

$$\sum_{j=1}^{J} w_j \max_{y \in \mathcal{X}} (p_y^n + q_y^n(x) t_j) \tag{14}$$

where $t_1, ..., t_J \in \mathbb{R}$ is the sequence of points that minimizes the quadratic quantization error of the Voronoi quantizer for T_{2a_n}. If $t_0 = -\infty$, and $t_{J+1} = \infty$, the weights are defined as $w_j = F_{T_{2a_n}}\left(\frac{t_j + t_{j+1}}{2}\right) - F_{T_{2a_n}}\left(\frac{t_{j-1} + t_j}{2}\right)$ for $j = 1, \cdots, J$. Here, $F_{T_{2a_n}}$ is the cumulative distribution function of T_{2a_n}. Therefore, the selected prompt representation based on the KG policy at state S_n is computed by solving the following mixed-integer optimization problem:

$$\max_{(\hat{x},\tau) \in \mathcal{X}^+; \tau \le M; x, y^1, ..., y^J \in \mathcal{X}} \sum_{j=1}^{J} w_j \theta_n^\top y^j + \tau \tag{15}$$

$$\text{s.t. } \|P_n^{1/2} \hat{x}\|_2 \le \sum_{j=1}^{J} w_j t_j x^\top \Sigma_n y^j \tag{16}$$

$$\tau \cdot \mathbb{1}_m - M(\mathbb{1}_m - x) \le \hat{x} \le Mx. \tag{17}$$

Here, $\mathbb{1}_m$ denotes an m-dimensional vector of ones, where m is the dimensionality of the prompt representation features. Matrix $P_n := \frac{a_n}{b_n}(\frac{A^\top A}{h^\top h} + \Sigma_n + \Sigma_\Theta)$, where A and h form the equality constraints of the feasible set $\mathcal{X} = \{x | Ax = h, Bx \le g, x \text{ binary}\}$. We assume that (B, g) define all of the facets of $\text{conv}(x)$ and thus (A, h) is the affine hull of $\text{conv}(x)$. We assume that $\text{conv}(x)$ is not full-dimensional (otherwise we introduce an artificial feature $\bar{x}$ with constraint $\bar{x} = 1$). In this case, $\mathcal{X}^+$, consisting of elements $(x, \tau) \in \mathbb{R}^m$, represents the homogenized version of the set $\mathcal{X}$. In the last constraint, M denotes a large constant. For derivation details and a computationally efficient iterative algorithm only involving solving mixed-integer programming problems to solve this problem, see Propositions 6 and 7 in (Moazeni et al., 2020).

The computational complexity of solving the mixed-integer optimization problems in our formulation (15)–(17) scales primarily with the number of features m, rather than with the number of alternatives, as in typical multi-armed bandit methods such as Ranking and Selection. A convex approximation based on semidefinite programming (SDP) relaxation can be created for this problem class, as investigated in (Defourny et al., 2015; Han et al., 2016). Furthermore, numerical results and runtime analyses in (Moazeni et al., 2020) demonstrate the scalability of solving the mixed-integer programming subproblem as the number of features m increases.

5 Computational Experiments

We demonstrate the performance of the proposed SOPL framework on the instruction induction tasks (Honovich et al., 2023) with GPT 3.5 and allow $N = 30$ iterations of prompt evaluation. The dataset consists of 24 tasks, covering various aspects of text comprehension. For each task, we partition the dataset into a demonstration set for creating prompts, a validation set for evaluating intermediate prompts, and a held-out test set for evaluating the final prompt. We focus on the 13 challenging tasks with validation scores below 80% and large variance using the default meta prompt template in (Honovich et al., 2023).

5.1 Benchmark Methods

We compare our method with two benchmarks EvoPrompt (Guo et al., 2024) and TRIPLE (Shi et al., 2024). Both methods provide solutions compatible with black-box access to LLMs and generate human-readable prompts, and are the two best performing algorithms with these two properties as reported by prior works. EvoPrompt uses the differential evolution algorithm to iteratively refine a population of candidate instructions. TRIPLE employs the continuously reject algorithm to identify an effective instruction from a candidate pool under a fixed budget. In addition, we use Greedy and TS presented in (7) and (8) as the baseline policies.

5.2 Results

Table 2 summarizes the average test performance and Figure 4 presents the test score for each task. The results indicate that SOPL-KG achieves the highest average test score of 0.6281, outperforming SOPL-TS, the second best one, by 5.60%. In comparison to the best prior algorithm, EvoPrompt,

Metric	SOPL-KG	EvoPrompt	TRIPLE	SOPL-TS	SOPL-Greedy
Test score	**0.6281**	0.5900	0.5609	0.5948	0.5750
Standard deviation	**0.0668**	0.0881	0.0966	0.0880	0.0959
Improvement of SOPL-KG	0.00%	6.47%	11.99%	5.60%	9.23%
Improvement per task	0.00%	17.92%	17.19%	9.13%	14.35%
Ranking	**1.85**	2.92	3.77	2.69	3.69

Table 2: Average test performance across 13 tasks for different methods.

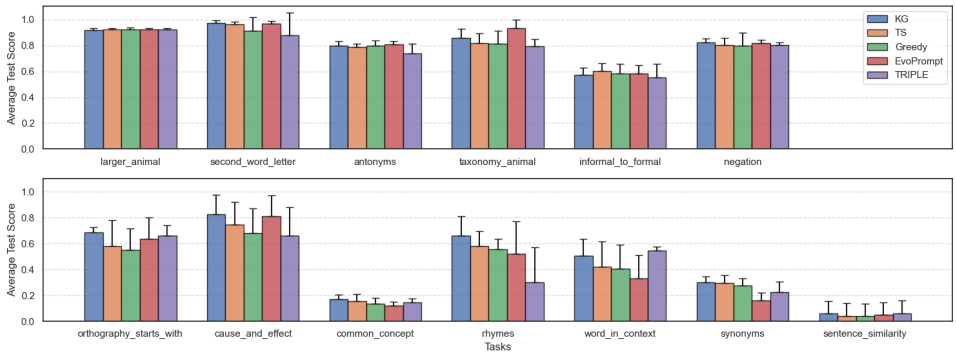

Figure 4: Test performance on 13 tasks for different methods. The height of each bar represents the average test score and the error bar represents the standard deviation across 20 replications with different random seeds.

SOPL-KG achieves a 6.47% improvement in average test score and an average improvement of 17.92% per task. For each task, we rank the five methods by their test scores. SOPL-KG attains the best average ranking of 1.85, followed by SOPL-TS, and outperforms all baselines.

5.3 Sensitivity to the Number of Iterations

We evaluate the performance under reduced evaluation budgets of $N = 20$ and $N = 10$. For the SOPL methods, we also experiment with an early stopping mechanism that terminates the process early if the best validation score does not improve for τ consecutive steps or when $N = 30$ is reached. Figure 5 depicts the average test score for different values of N. For the SOPL methods, we include two additional experiments of early stopping using $\tau = 5$ and $\tau = 10$, where N represents the average number of realized iterations before termination.

As the number of iterations increases, algorithms generally discover higher-performing prompts, with SOPL-KG showing the largest improvement within the first few iterations. Additionally, SOPL-KG consistently outperforms other methods for different values of N. The finding confirms that, empowered by the forward-looking KG policy, SOPL is capable of identifying high-quality prompts within only a few prompt evaluations.

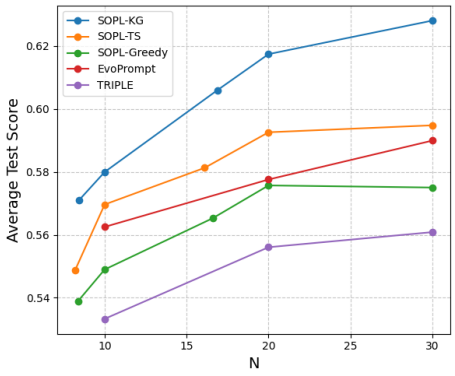

Figure 5: Average test score versus number of iterations.

5.4 Sensitivity to the Prompt Selection Policy

We further demonstrate that the advantage of the KG policy over other prompt selection policies depends on the sensitivity of LLM responses to prompt variations. In Figure 6, we observe positive correlations between the variation in prompt scores, measured by the coefficient of variation across 100 randomly selected feature vectors, and the relative improvement in test score of SOPL-KG over the TS and Greedy policies, with correlation coefficients $\rho_1 = 0.89$ and $\rho_2 = 0.78$, respectively.

These results suggest that KG is particularly advantageous for challenging tasks where LLM re-

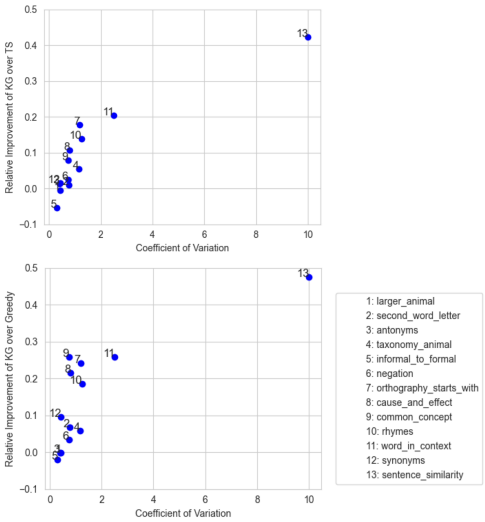

Figure 6: Improvement of SOPL-KG over SOPL-TS (upper plot) and SOPL-Greedy (lower plot) versus the coefficient of variation.

sponses are highly sensitive to prompt formulation, achieving over 10% improvement in test accuracy. For easier tasks such as `informal_to_formal`, where the score function is relatively flat with respect to prompt features, simple exploitation policies such as Greedy often suffice.

6 Conclusion and Future Work

This paper introduces SOPL, a sequential optimal prompt learning framework for efficient automated prompt engineering when exhaustive evaluation is costly or impossible. We develop a feature-based approach to model an expansive prompt space. The KG policy with correlated beliefs facilitates efficient and scalable prompt learning. Our method achieves superior performance on instruction induction tasks within 30 prompt evaluations, showing the potential of optimal learning methods for efficient prompt learning. Future work may explore continuous prompt representations by embedding vectors using the SOPL framework.

Limitations

SOPL offers an automated approach for prompt generation and refinement, utilizing a feature-based representation to form the prompt space. One limitation, however, is that SOPL requires users to specify a set of features in advance to construct the search space. While this step requires some manual efforts, these features are often intuitive to define given the task or can be derived from existing material or experience on similar tasks.

Acknowledgments

This research was supported in part through the computational resources and staff contributions provided for the Quest high performance computing facility at Northwestern University which is jointly supported by the Office of the Provost, the Office for Research, and Northwestern University Information Technology.

References

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of the Conference of German Society for Computational Linguistics and Language Technology*, 30:31–40.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2024. Instructzero: Efficient instruction optimization for black-box large language models. In *International Conference on Machine Learning*.

Stephen E Chick. 2006. Bayesian ideas and discrete event simulation: why, what and how. In *Proceedings of the 2006 Winter Simulation Conference*, pages 96–106.

Stephen E Chick, Jürgen Branke, and Christian Schmidt. 2010. Sequential sampling to myopically maximize the expected value of information. *INFORMS Journal on Computing*, 22(1):71–80.

Boris Defourny, Ilya O Ryzhov, and Warren B Powell. 2015. Optimal information blending with measurements in the L^2 sphere. *Mathematics of Operations Research*, 40(4):1060–1088.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *International Conference on Machine Learning*.

Peter Frazier, Warren Powell, and Savas Dayanik. 2009. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613.

Peter I Frazier and Warren B Powell. 2011. Consistency of sequential Bayesian sampling policies. *SIAM Journal on Control and Optimization*, 49(2):712–731.

Peter I Frazier, Warren B Powell, and Savas Dayanik. 2008. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*.

Shanti S Gupta and Klaus J Miescke. 1996. Bayesian look ahead one-stage sampling allocations for selection of the best population. *Journal of Statistical Planning and Inference*, 54(2):229–244.

Bin Han, Ilya O Ryzhov, and Boris Defourny. 2016. Optimal learning in linear regression with combinatorial feature selection. *INFORMS Journal on Computing*, 28(4):721–735.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952.

Can Jin, Hongwu Peng, Shiyu Zhao, Zhenting Wang, Wujiang Xu, Ligong Han, Jiahui Zhao, Kai Zhong, Sanguthevar Rajasekaran, and Dimitris N Metaxas. 2024. Apeer: Automatic prompt engineering enhances large language model reranking. *arXiv preprint arXiv:2406.14449*.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2024. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4099–4113.

Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. 2023. Large language models understand and can be enhanced by emotional stimuli. *arXiv preprint arXiv:2307.11760*.

Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025. A survey of automatic prompt engineering: An optimization perspective. *arXiv preprint arXiv:2502.11560*.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. 2024. Use your INSTINCT: INSTruction optimization for LLMs using neural bandits coupled with transformers. In *International Conference on Machine Learning*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Yifan Luo, Yiming Tang, Chengfeng Shen, Zhennan Zhou, and Bin Dong. 2023. Prompt engineering through the lens of optimal control. *arXiv preprint arXiv:2310.14201*.

Somayeh Moazeni, Boris Defourny, and Monika J Wilczak. 2020. Sequential learning in designing marketing campaigns for market entry. *Management Science*, 66(9):4226–4245.

Warren B Powell. 2022. *Reinforcement learning and stochastic optimization: A unified framework for sequential decisions*. John Wiley & Sons.

Warren B Powell and Ilya O Ryzhov. 2012. *Optimal learning*, volume 841. John Wiley & Sons.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.

Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, and 1 others. 2025. A systematic survey of automatic prompt optimization techniques. *arXiv preprint arXiv:2502.16923*.

Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. 2024. Best arm identification for prompt learning under a limited budget. *arXiv preprint arXiv:2402.09723*.

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

Ning Wu, Ming Gong, Linjun Shou, Shining Liang, and Daxin Jiang. 2023. Large language models are diverse role-players for summarization evaluation. In *International Conference on Natural Language Processing and Chinese Computing*, pages 695–707.

Zhichao Xu, Daniel Cohen, Bei Wang, and Vivek Srikumar. 2024. In-context example ordering guided by label distributions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2623–2640.

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

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*.

A Implementation Details

We use OpenAI GPT-3.5 as the LLM for both generating and evaluating instructions. We allow $N = 30$ opportunities to evaluate on the entire validation dataset since there is no further learning after this. We set the population size to be 10 for EvoPrompt as in (Guo et al., 2024), and set the size of the candidate pool to be 30 for TRIPLE as in (Shi et al., 2024). We ensure that the same number of API calls to the LLM is used for evaluating on the validation data across all methods.

We record the prompt with the highest validation score during the process. When the maximum number of prompt evaluation is reached, the prompt with the highest validation score is used as the final prompt. The test score is obtained by evaluating the final prompt on the held-out test set, which consists of 100 examples unless fewer are available in the dataset (Honovich et al., 2023).

We repeat the experiment for 20 replications with different random seeds. The test set is kept the same for all replications. For each replication, we randomly select 10 examples from the rest of the dataset as the demonstration set, and then randomly select 100 examples, or all remaining examples if fewer are available, as the validation set.

Details of prompt features The set of features that defines the prompt search space has been summarized in Table 1. In addition, the other three meta prompt templates are shown in Figure 8. When the paraphrasing option is enabled, we first generate an instruction using the meta prompt constructed from the rest of the selected features. We then insert the generated instruction, along with the chosen descriptive word and roles, into the paraphrase template shown in Figure 7. This produces a paraphrased instruction as the candidate to evaluate.

Prompt evaluation To evaluate each selected prompt representation x, we follow the procedure in Algorithm 2. We construct an evaluation prompt by inserting the generated instruction and an input from the dataset into the evaluation template shown in Figure 9. The response from GPT 3.5 is then compared to the ground truth label using a task-specific metric such as exact match. By averaging the metric values across all examples in the dataset, we obtain the score u_x for the selected x.

Paraphrase the instruction **[INSTRUCTION]** in a **[DESCRIPTION]** manner. Specify that the instruction is given by **[ROLE1]** to **[ROLE2]**.

Figure 7: Paraphrasing template.

[ROLE1] gave a **[ROLE2]** an instruction and inputs. The **[ROLE2]** read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

Input: **[INPUT 1]**
Output: **[OUTPUT 1]**
...
Input: **[INPUT 5]**
Output: **[OUTPUT 5]**

The instruction was ___

(a) Meta prompt template 2.

Input: **[INPUT 1]**
Output: **[OUTPUT 1]**
...
Input: **[INPUT 5]**
Output: **[OUTPUT 5]**

[ROLE1] gave a **[ROLE2]** a list of inputs and asked them to perform a task. The **[ROLE2]** performed the task and returned an output for each input.

What was the instruction? Generate a **[DESCRIPTION]** instruction.

(b) Meta prompt template 3.

Input: **[INPUT 1]**
Output: **[OUTPUT 1]**
...
Input: **[INPUT 5]**
Output: **[OUTPUT 5]**

[ROLE1] gave a **[ROLE2]** a list of inputs and asked them to perform a task. The **[ROLE2]** performed the task and returned an output for each input.

What was the instruction? Respond only with the instruction.

(c) Meta prompt template 4.

Figure 8: Meta prompt templates.

B Additional Experiment Results

Sensitivity to reduced budgets Table 3 presents details of the test performance with reduced evaluation budgets of $N = 20$ and $N = 10$, and Table 4 summarizes the results for adopting early stopping in SOPL, which halts the process if the best validation score does not improve for $\tau = 10$ or $\tau = 5$ consecutive steps. SOPL-KG outperforms all other methods within fewer iterations. Moreover, within only 17 realized iterations on average, SOPL-KG achieves a competitive score of 0.6060, indicating that early stopping may further reduce evaluation cost without significantly compromising performance.

Method	$N = 20$		$N = 10$	
	Mean	**STD**	**Mean**	**STD**
EvoPrompt	0.5776	0.0956	0.5625	0.0920
TRIPLE	0.5561	0.0996	0.5333	0.1087
SOPL-KG	**0.6174**	**0.0771**	**0.5800**	0.0935
SOPL-TS	0.5926	0.0845	0.5696	**0.0893**
SOPL-Greedy	0.5757	0.0941	0.5490	0.1012

Table 3: Average test score after fewer iterations.

Algorithm 2 Score function $\mathrm{Eval}(x)$

Require LLM, validation data $\{(p_i, q_i)\}_{i=1}^{V}$, meta prompt construction function $\mathrm{MetaPrompt}$, evaluation prompt construction function $\mathrm{EvalPrompt}$, metric function Metric.

Create meta prompt $M_x \leftarrow \mathrm{MetaPrompt}(x)$.
Generate instruction $I_x \leftarrow \mathrm{LLM}(M_x)$.
for $i = 1, ..., V$ **do**
  Get evaluation prompt $E_i \leftarrow \mathrm{EvalPrompt}(I_x, p_i)$.
  Receive LLM response $R_i \leftarrow \mathrm{LLM}(E_i)$.
  Evaluate LLM response $U_i \leftarrow \mathrm{Metric}(R_i, q_i)$.
end for
Compute the average score $u_x \leftarrow \frac{1}{V}\sum_{i=1}^{V} U_i$.
return u_x.

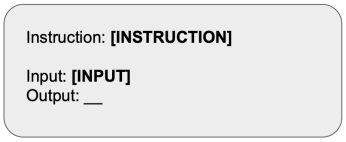

Figure 9: Evaluation template.

Sensitivity to feature inclusion To assess the effectiveness of incorporating the set of features used in the meta prompt templates, we conduct experiments using the default meta prompt from (Honovich et al., 2023), which includes only the feature for demonstrative examples. We apply SOPL-KG to search for the best configuration of demonstrative examples from 20 predefined choices. We allow $N = 30$ evaluations and repeat the experiments for 20 replications. The results in Table 5 demonstrate that performance degrades when features that enhance the meta prompt are excluded. While optimizing over a single feature dimension is easier than optimizing multiple features simultaneously, it yields suboptimal prompts. This finding suggests that incorporating multiple features into the meta prompt templates effectively expands the search space and leads to higher-quality prompts.

Policy	$\tau = 10$		$\tau = 5$	
	Score	**Steps**	**Score**	**Steps**
KG	**0.6060**	16.88	**0.5711**	8.45
TS	0.5813	**16.10**	0.5488	**8.20**
Greedy	0.5653	16.60	0.5389	8.37

Table 4: Average number of realized iterations and average test score for different policies with early stopping.

Task	KG (1 feature)		KG (all features)	
	Mean	STD	Mean	STD
antonyms	0.7625	0.0200	**0.7975**	0.0370
cause_and_effect	0.5780	0.2773	**0.8260**	0.1509
common_concept	0.1434	0.0600	**0.1706**	0.0350
informal_to_formal	0.5683	0.0929	**0.5718**	0.0576
larger_animal	0.9055	0.0561	**0.9210**	0.0130
negation	0.8245	0.0262	**0.8255**	0.0297
orthography_starts_with	0.6405	0.1575	**0.6870**	0.0395
rhymes	0.4065	0.1615	**0.6610**	0.1504
second_word_letter	0.9565	0.0065	**0.9745**	0.0201
sentence_similarity	0.0215	0.0531	**0.0590**	0.0959
synonyms	0.1490	0.0122	**0.3020**	0.0441
taxonomy_animal	0.8480	0.1005	**0.8610**	0.0677
word_in_context	0.1965	0.2039	**0.5085**	0.1272
Average	0.5385	0.0944	**0.6281**	0.0668

Table 5: Comparison of KG performance with 1 prompt feature and with all prompt features.