mathematics

MDPI

Article

Prompt Optimization in Large Language Models

Antonio Sabbatella [1], Andrea Ponti [2], Ilaria Giordani [3], Antonio Candelieri [2,*] and Francesco Archetti [1]

[1] Department of Computer Science, Systems and Communications, University of Milan-Bicocca, 20126 Milan, Italy; a.sabbatella@campus.unimib.it (A.S.); francesco.archetti@unimib.it (F.A.)
[2] Department of Economics, Management, and Statistics, University of Milan-Bicocca, 20126 Milan, Italy; a.ponti5@campus.unimib.it
[3] Oaks srl, 20125 Milan, Italy; giordani@oaks.cloud
* Correspondence: antonio.candelieri@unimib.it

Abstract: Prompt optimization is a crucial task for improving the performance of large language models for downstream tasks. In this paper, a prompt is a sequence of n-grams selected from a vocabulary. Consequently, the aim is to select the optimal prompt concerning a certain performance metric. Prompt optimization can be considered as a combinatorial optimization problem, with the number of possible prompts (i.e., the combinatorial search space) given by the size of the vocabulary (i.e., all the possible n-grams) raised to the power of the length of the prompt. Exhaustive search is impractical; thus, an efficient search strategy is needed. We propose a Bayesian Optimization method performed over a continuous relaxation of the combinatorial search space. Bayesian Optimization is the dominant approach in black-box optimization for its sample efficiency, along with its modular structure and versatility. We use BoTorch, a library for Bayesian Optimization research built on top of PyTorch. Specifically, we focus on Hard Prompt Tuning, which directly searches for an optimal prompt to be added to the text input without requiring access to the Large Language Model, using it as a black-box (such as for GPT-4 which is available as a Model as a Service). Albeit preliminary and based on "vanilla" Bayesian Optimization algorithms, our experiments with RoBERTa as a large language model, on six benchmark datasets, show good performances when compared against other state-of-the-art black-box prompt optimization methods and enable an analysis of the trade-off between the size of the search space, accuracy, and wall-clock time.

Keywords: Bayesian Optimization; prompt optimization; black-box Large Language Models

MSC: 68T50

Citation: Sabbatella, A.; Ponti, A.; Giordani, I.; Candelieri, A.; Archetti, F. Prompt Optimization in Large Language Models. *Mathematics* **2024**, *12*, 929. https://doi.org/10.3390/math12060929

Academic Editors: Firuz Kamalov, Hana Sulieman and Ayman Alzaatreh

Received: 12 January 2024
Revised: 27 February 2024
Accepted: 6 March 2024
Published: 21 March 2024

1. Introduction

Prompt optimization is designed to provide an effective adaptation of Large Language Models (LLMs) to specific tasks. Whether it is via text or images, an appropriate prompt makes the model's output better suited to the user's task.

Recent advances in foundation models such as GPT-3 and ChatGPT demonstrate strong instruction-following abilities for natural language tasks. However, performance remains sensitive to prompt engineering, which involves discovering properly crafted prompts. Manually engineering effective prompts is challenging and requires a substantial, costly, and time-consuming trial–error process. This highlights the need to automate prompt optimization, especially for black-box LLMs where access to the model is limited to the predictions and not the gradients. Prompt learning optimizes model performance by tuning the discrete tokens of prompts while keeping model parameters fixed. This contrasts with fine-tuning the entire model; instead, updating only the prompt sequence confers multiple advantages such as improved cost-effectiveness, avoidance of overfitting, and enhanced privacy. Most relevantly, prompt optimization aligns with black-box constraints, querying the model to update prompts without reliance on model owner infrastructure

or risk of data leakage. By searching the discrete prompt space through iterative model queries, we can automatically learn improved prompts without fine-tuning.

This paper presents a comparative analysis of various datasets and tasks designed to advance natural language understanding, which will be detailed in Section 4.1. MNLI challenges models to discern the validity of a hypothesis against a given premise across diverse genres. QQP tests for semantic equivalence in user-generated questions. SST-2 evaluates the ability to accurately predict sentiment from movie reviews. MRPC focuses on identifying paraphrases among news sentences. QNLI is derived from SQuAD, asking models to verify if a sentence contains an answer to a question. Lastly, RTE requires models to assess textual entailment within sentence pairs. Each dataset serves as a benchmark for specific linguistic capabilities, collectively pushing the boundaries of machine comprehension. We exhibit two examples of the tasks used for the evaluation of the model performance:

QQP (semantic equivalence):

Question 1: "How can I learn to cook Italian food?"
Question 2: "What are some good resources for learning Italian cuisine?"
Label: Equivalent

SST-2 (sentiment prediction):

Sentence: "This movie was a fantastic journey through imagination and creativity."
Label: Positive

A prompt $\mathbf{p}$ is a sequence of a given length L, of n-grams, or of individual tokens selected from a vocabulary V. Our goal is to engineer prompts to address a specific task with an input space X and output defined as $f(\mathbf{p}, x \in X)$**.**

We denote with $Concat(prompt\ \mathbf{p}, input\ x)$ a query q. The problem is formulated as an optimization problem with an objective function that measures the performance over a task $h(f(\mathbf{p}, x), Y)$ using a score produced by an evaluation metric (e.g., accuracy or F1 for a classification task) to compare $f(\mathbf{p}, x)$ with the ground truth Y. If we assume that the couple (x, y) are drawn from a task distribution D, we obtain the following stochastic optimization problem:

$$\max_{\mathbf{p} \in V^L} \mathbb{E}_{(X,Y)}(h(f(\mathbf{p}, x), Y)) \tag{1}$$

The search space V^L consists of possible prompts of length L, and whose components are elements of the vocabulary V.

Prompt engineering methods can be split into two categories—Hard Prompt Tuning (HPT), which directly searches for an optimal prompt in the combinatorial search space V^L, and Soft Prompt Tuning (SPT), which uses continuous-valued language embeddings and searches for the optimal embedding via gradient-based optimization in the resulting continuous latent space. It is important to remark that hard prompts have two important advantages:

- They are portable, meaning that they can be discovered using one LLM and then reused with a different one. This cannot be done via soft prompts, which are instead task- and language model-specific embeddings.
- They are critically important if the LLM is available as a Model as a Service (MaaS), meaning that users can only access the model's output for any given input. Moreover, from the provider viewpoint, HPT mitigates the security risk of the cloud infrastructure as the model's parameters are hidden and known only by the service providers, giving the user access only to the query and prediction interface. This black-box setting is also aligned with the interest of the final users, allowing for structuring a simple service without requiring the LLM's gradient.

In this paper we focus on HPT and assume that the user, after having provided an input $x \in X$ and a prompt $\mathbf{p}$, has access only to the LLM output $f(\mathbf{p}, x)$ and its score h.

Given the dimension of the vocabulary V and the prompt length L, prompt optimization is an intractable combinatorial optimization problem, with a search space consisting of

$|V|^L$ possible solutions (in the case that duplicated n-grams are allowed in the prompt), with $|V| >> L$ the size of the vocabulary.

The method we propose consists of a relaxation of the combinatorial space into a continuous search space in order to enable efficient sampling through Bayesian Optimization (BO) to search for the optimal value of $h(f(\mathbf{p}, x))$. This results in a new HPT approach working directly on the space of n-grams by applying a continuous relaxation of the combinatorial decision variables. To validate the approach, we conducted a computational analysis on benchmarking datasets.

BO has become the dominant approach in black-box optimization [1,2]. The main advantage of BO is its sample efficiency, along with its modular structure and versatility. We use BoTorch [3], a library for BO research built on top of PyTorch. BoTorch provides a modular and flexible interface for composing BO algorithms.

The main contributions of the paper are: (*i*) the validated feasibility of using BO as a sample efficient method for black-box prompt optimization in LLMs; (*ii*) a significant wall-clock time reduction over other black-box approaches; (*iii*) the feasibility of a "naïve" relaxation to a continuous space; (*iv*) thanks to this relaxation, empirical results showing that a "vanilla" BO algorithm, from BoTorch, is sufficient instead of using more specialized ones—and still available in BoTorch—for combinatorial and high-dimensional settings.

The rest of the paper is organized in the following sections:

- Section 2 "Related works" provides a broad analysis of the state-of-the-art literature on prompt optimization, focused on the black-box methods.
- Section 3 "Methodology" provides the formulation of prompt optimization problems and describes hard prompt tuning via Bayesian Optimization and the continuous relaxation of the combinatorial space.
- Section 4 "Computational Results" presents datasets, baselines, and computational results.
- Section 5 contains conclusions, limitations, and perspectives of the proposed approach.

2. Related Works

Different modeling and algorithmic strategies have been proposed for prompt optimization. Ref. [4] were among the first to demonstrate the power of prompting for task adaptation of pre-trained models. More recently, two papers proposed improving the reasoning capability of LLMs with a "step-by-step" interactive approach. Ref. [5] proposed a "chain of thought" reasoning approach, which uses the "step-by-step" reasoning approach, while [6] introduced the "Tree of thoughts" that is used to augment problem-solving capability, focusing on the exploration over coherent units of text (thoughts) used as intermediate steps to solve the original problem, whose evaluations are demanded of the LLMs' respective models.

Recently, a set of strategies based on automating the generation of prompts using optimization methods have been proposed which are more relevant to the method proposed in this manuscript.

A basic categorization of prompt/instruction optimization methods can be drawn along the lines of continuous versus discrete and black-box versus white-box.

Continuous/black-box*:* The approach in [7] optimizes a continuous prompt prepended to the input text. Instead of optimizing in the original high-dimensional prompt space, the optimization is performed in a randomly generated subspace of a lower intrinsic dimensionality. This approach is further developed in [8] which used a normal distribution in the projection instead of a uniform distribution. Another approach to black-box prompt tuning is proposed in [9] which applies a policy gradient to estimate the gradients of the parameters of the categorical distribution of each discrete prompt. Another derivative-free approach has been proposed in Clip tuning [10].

Continuous/white-box: Prefix-tuning [11,12] and Optiprompt [13] directly optimize in the embedding space, leaving the other model parameters frozen. In [14], an approach to optimize hard text prompts via efficient gradient-based optimization is presented.

Discrete/black-box: Several methods have been proposed to tune discrete prompts for LLMs without relying on gradients. One approach is GRIPS [15], which provides an automated procedure for improving prompts via an iterative local edit and gradient-free

search. APO [16] is a method that automatically improves prompts by using natural language "gradients" that criticize the current prompt and suggest semantic changes. The gradients are formed by using minibatches of data and an LLM API and are then "propagated" into the prompt by editing the prompt in the opposite direction of the gradient. A different approach is proposed in APE [16] based on the observation that only a small number of tokens exerts a disproportioned influence on the LLM prediction, and APE proposes to first cluster and then prune the search space to focus exclusively on influential tokens. Other approaches include EvoPrompt [17], which uses evolutionary algorithms to generate and improve prompts with large language models; BDPL [9], which models the choice of words in the prompt as a policy of reinforcement learning and optimizes it by a variance-reduced policy gradient estimator; and OPRO [18], which describes the optimization task in natural language and feeds it to the large language model as a prompt and then generates new solutions from the prompt that contain previously generated solutions with their values. Another approach to black-box prompt tuning is proposed in [9], which applies a policy gradient to estimate the gradients of the parameters of the categorical distribution of each discrete prompt.

Discrete/white-box: White-box methods are discrete prompt optimization methods that rely on the gradients or parameters of the LLM. These methods can leverage the information from the LLM to guide the prompt search or tuning process. One example of a white-box method is AUTOPROMPT [19], which automatically generates prompts for a diverse set of tasks based on a gradient-driven search. Another example of a white-box method is Fluent Prompt [20], which uses a pre-trained language model to generate candidate prompts that are syntactically and semantically coherent and then selects the best prompt based on the LLM's output probability or accuracy. Alternative spaces for token-based optimization have also been proposed in [21,22], which provide query-dependent discrete prompts whose optimization is performed using reinforcement learning. Another gradient-free approach is proposed in [23], which adds a layer of uncertainty quantification to improve the reliability of prompt tuning and to consider a strict notion of a likelihood-free black-box setting.

Bayesian approaches: Bayesian Optimization is a widely considered a sampling-efficient solution for black-box optimization. It has been gaining importance for prompt optimization in large language models. Ref. [24] propose a two-stage approach called InstructZero: using an open source LLM, the first stage converts a prompt into an instruction and (in the second stage) submits it to the black-box LLM—which computes the performance score of this instruction and then sends it to the Bayesian Optimization module to produce new soft prompts. A specific application in the context of adversarial learning/optimization is reported in [25]. A similar approach, namely INSTINCT, has been recently proposed in [26]. The main characteristic is that a neural network is used instead of a Gaussian Process in the BO algorithm. Finally, a preliminary version of the BO-based prompt optimization algorithm presented in this manuscript has been briefly described in [27]. LLMs have also been proposed for the multi-armed bandit (MAB) problems, which are closely related to Bayesian Optimization. Ref. [28] propose an LLM-based strategy that enables adaptive balancing of exploration and exploitation. Ref. [29] presents an approach that integrates the capabilities of large language models (LLMs) within BO, framing the BO problem in natural language terms and, thereby, enabling LLMs to iteratively propose promising solutions conditioned on historical evaluations.

The loss function considered in the above approaches is usually taken from the machine learning fields and computational linguistics. An interesting approach, which we plan to address in the future, is to augment the loss with a term related to the readability of the output of the LLM. Pioneering papers about readability are [30,31].

3. Methodology

3.1. Problem Formulation

In this paper, HPT aims at finding a sequence with a prefixed length of n-grams to be used as a prefix to the model query with the goal of maximizing the performance on a downstream task.

As mentioned above, a prompt $\mathbf{p} \in \mathcal{V}^{\updownarrow}$ is defined as a sequence of n-grams. The space $\mathcal{V}^{\updownarrow}$ represents all possible combinations of $\updownarrow$ n-grams, and, consequently, $\mathcal{V}$ is the considered vocabulary, i.e., the set of n-grams. In particular, the tokens of the original model's vocabulary have been merged in n-grams based on their Pointwise Mutual Information (PMI) in the considered dataset. Therefore, the n-grams with a higher PMI are considered as prompt candidates. This ensures that only n-grams of tokens that frequently appear together are used to form the actual vocabulary $\mathcal{V}$.

Let $x, y \in \mathcal{D}$ be an instance of the dataset $\mathcal{D}$ with its true label, e.g., x can be text to be classified and y its true label. We want to find the prompt $\mathbf{p}^*$ that maximizes a scoring function:

$$\mathbf{p}^* = \underset{\mathbf{p} \in \mathcal{V}^{\updownarrow}}{\text{argmax}} \mathbb{E}_{x,y \sim \mathcal{D}}[h(f(\mathbf{p}.x), y)] \tag{2}$$

where h is a task-specific scoring function (e.g., accuracy or f-measure for classification tasks), and f is the LLM's response on input $\mathbf{p}.x$ (the string concatenation between the prompt $\mathbf{p}$ and the dataset instance x). The expectation is taken over the distribution $\mathcal{D}$ of inputs x and output y.

For example, considering a text classification task and the misclassification error as scoring function h, we have:

$$\mathbf{p}^* = \underset{\mathbf{p} \in \mathcal{V}^{\updownarrow}}{\text{argmax}} \frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} h(f(\mathbf{p}.x),\ y) \tag{3}$$

where y is the true label of x.

The scoring function h utilized in the prompt optimization framework is defined as the classification score between the predicted label y_i^p and the ground truth label y_i for a given input x_i and prompt p. Formally, this is represented as:

$$h(f(\mathbf{p}.x_i),\ y_i) = 1\left(y_i^p = y_i\right) \tag{4}$$

where:

- $Y = \{1, 2, \ldots, C\}$ is the label space with C distinct class labels and denotes the probability distribution induced by model M when given as input for the prompt p concatenated with the input x_i.
- $y_i^p \in Y$ denotes the predicted classification label by model M for response $f(p, x)$ considered as the label y associated with the highest probability.
- $y_i \in Y$ denotes the true classification label paired with input $x_i \in X$.
- $1(\cdot)$ defines the indicator function that returns one if the condition inside the parenthesis evaluates to true—or returns zero otherwise.

3.2. Hard Prompt Tuning via Bayesian Optimization

Let denote with $F(\boldsymbol{p})$ the expectations in Equation (2):

$$F(\mathbf{p}) = \mathbb{E}_{x,y \sim \mathcal{D}}[h(f(\mathbf{p}.x), y)] \tag{5}$$

Evaluating $F(\mathbf{p})$ for a given prompt $\mathbf{p}$ requires many evaluations of the scoring function h, one for each different input x and output y sampled from the distribution $\mathcal{D}$. Since each evaluation must query an LLM, it is a black-box and expensive function. Thus, BO is used to maximize $F(\mathbf{p})$ using a Gaussian Process (GP) as a surrogate model. Let the prompts evaluated so far be $\mathcal{P}_{1:n} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, with associated scores $\mathbf{h} = F(\mathbf{p}_1), \ldots, F(\mathbf{p}_n)$, possibly noisy. Then, the GP posterior mean $\mu(\mathbf{p})$ and variance $\sigma^2(\mathbf{p})$, conditioned on the observed prompts and scores, are:

$$\begin{aligned} \mu(\mathbf{p}) &= \mathrm{k}(\mathbf{p}, \mathcal{P}_{1:n})\left[\mathrm{K} + \lambda^2 I\right]^{-1}\mathbf{h} \\ \sigma^2(\mathbf{p}) &= k(\mathbf{p}, \mathbf{p}) - \mathrm{k}(\mathbf{p}, \mathcal{P}_{1:n})\left[\mathrm{K} + \lambda^2 I\right]^{-1}\mathrm{k}(\mathcal{P}_{1:n}, \mathbf{p}) \end{aligned} \tag{6}$$

where $\mathrm{K} \in \mathbb{R}^{n \times n}$ is the GP kernel matrix with entries $\mathrm{K}_{ij} = k\left(\mathbf{p}_i, \mathbf{p}_j\right)$, I is the identity matrix, and λ^2 is the noise variance.

The next prompt $\mathbf{p}_{n+1}$ is chosen by optimizing an acquisition function, balancing between exploration and exploitation. A common and widely used acquisition function is the Upper Confidence Bound (UCB):

$$UCB(\mathbf{p}) = \mu(\mathbf{p}) + \beta\sigma(\mathbf{p}) \tag{7}$$

Then, the score of the suggested prompt $\mathbf{p}_{n+1}$ is evaluated, and the two sets, $\mathcal{P}_{1:n}$ and $\mathbf{h}$, are consequently updated, along with the GP model. The BO algorithm continues, iteratively, until a maximum number of prompts has been suggested and evaluated.

Figure 1 shows the general framework of HPT via BO. A set of n random prompts are generated and evaluated. These prompts are then used to fit the initial GP, and, by optimizing the acquisition function, a new candidate prompt is generated, which is then evaluated and used to update the GP. The process is iteratively repeated until a budget is met.

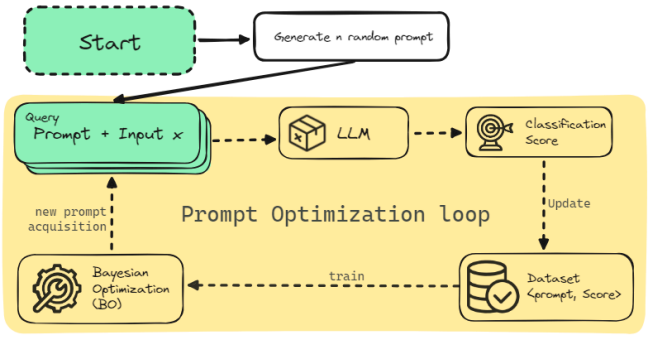

Figure 1. Prompt optimization loop using BO.

3.3. Continuous Relaxation of the Combinatorial Space

The goal of BO, considering the HPT problem, is to find the optimal prompt $\mathbf{p}^* \in \mathcal{V}^{\updownarrow}$. It is important to note that the prompt space, $\mathcal{V}^{\updownarrow}$, is a combinatorial space consisting of all the possible prompts with length $\updownarrow$ which can be generated by concatenating n-grams from $\mathcal{V}$. Working in this discrete space can be intractable because the number of possible solutions increases exponentially as the cardinality of $\mathcal{V}^{\updownarrow}$ does. Unfortunately, the sample efficiency of BO cannot be directly leveraged in this combinatorial search space because (vanilla) GPs are well suited for working on continuous space due to the nature of the kernel function. Indeed, the kernel defines the "closeness" between two prompts in the space, and the choice of kernel is crucial for BO, as it guides the search towards promising regions of the search space. Although there are several research works on combinatorial BO, as well as on new kernels for combinatorial inputs, our proposal is easier, and it is a well-known practical workaround usually adopted and suggested. In addition, as shown by the empirical results, it is an effective and efficient solution. Specifically, it consists of a continuous relaxation of the search space.

First, instead of considering n-grams as they are, we used the indices representing their positions in the vocabulary $\mathcal{V}$. This leads us to transform the search space $\mathcal{V}^{\updownarrow}$ into $\{1, \ldots, |V|\}^{\updownarrow}$. It is important to remark that this was not sufficient: the new search space was still combinatorial, with the same cardinality of possible solutions. The unique and important difference is that prompts were represented as vectors of $\updownarrow$ integer values. The next step was trivial: integer values were to be treated as real values. These two steps allowed us to transform the original combinatorial space into a continuous one. The underlying idea is analogous to embedding in SPT (move from a structured space to an associated continuous latent space), but without the need to embed anything.

As the relaxation process strictly depends on the order of the n-grams into the vocabulary, this means that, even if the (relaxed) search space is continuous, the unknown

objective function may not be as smooth. To deal with this possible issue, we decided to use a Matern Kernel, which allowed us to reasonably deal with relevant variations in the objective function (contrary to smoother kernels like the Squared Exponential). The final issue to solve was related to the new prompt suggested by BO. We needed to convert the continuous prompt obtained by optimizing the acquisition function into a vector of integer values. The simplest way to do this was to round back every vector component to the closest integer. Finally, the prompt was retrieved by concatenating the n-grams identified by the integer values (that were indices of the n-grams in the vocabulary).

Overall, our approach—namely, **PrompT-BO** (**P**rompt **T**uning via **B**ayesian **O**ptimization)—allowed us to leverage the powerful machinery of BO without being limited by the combinatorial explosion of the original combinatorial space, and without requiring any embedding. A graphical representation of the proposed approach is provided in Figure 2, providing more details on the BO components and their roles.

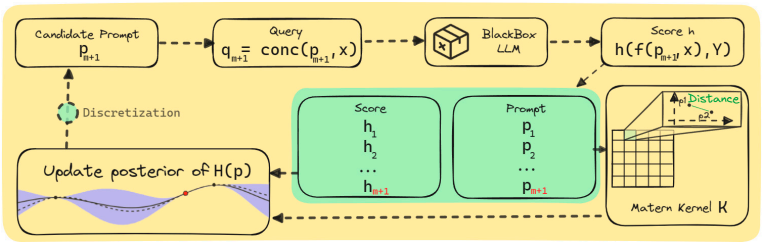

Figure 2. Graphical representation of the BO components, the continuous relaxation, and their interaction.

The pseudocode of the proposed approach is as follows Algorithm 1:

Algorithm 1 Bayesian Prompt Optimization

Required:
LLM Model $\boldsymbol{M}$

Training Dataset $\boldsymbol{X_{tr}}$
Validation Dataset $\boldsymbol{X_v}$
Test Dataset $\boldsymbol{X_{te}}$

Number of candidate prompts $\boldsymbol{k}$
Acquisition function $\boldsymbol{UCB}(\mathbf{p})$
Objective function $\boldsymbol{F}(\mathbf{p})$ as defined in (5)
Number of initial prompts $\boldsymbol{N}$
Set of prompts and associated score $\boldsymbol{D}$ = {}

1: Generate $\boldsymbol{N}$ initial random prompts $\mathbf{p_1},\ldots,\mathbf{p_N}$

2: for i = 1:$\boldsymbol{N}$ do
3: $y_i = \boldsymbol{F}(\mathbf{p_i} \mid \boldsymbol{M}, \boldsymbol{X_{tr}}, \boldsymbol{X_v})$.
4: $\boldsymbol{D} = \boldsymbol{D} \cup \{\mathbf{p}_i, y_i\}$
4: end for
5: GP = GaussianProcess($\boldsymbol{D}$)

6: for i = 1:$\boldsymbol{k}$ do
7: $\mathbf{p}_{\text{new}}$ = argmax $\boldsymbol{UCB}(\mathbf{p})$
8: y_{new} = F($\mathbf{p}_{\text{new}} \mid \boldsymbol{M}, \boldsymbol{X_{tr}}, \boldsymbol{X_v}$)
9: $\boldsymbol{D} = \boldsymbol{D} \cup \{\mathbf{p}_{\text{new}}, y_{\text{new}}\}$
10: GP = GaussianProcess($\boldsymbol{D}$)
11: end for
11: return ($\mathbf{p}^*$, y^*) the best solution in $\boldsymbol{D}$

4. Computational Results

The analysis provided in this section utilizes qualitative case examples and quantitative timing comparisons to validate the strengths of the proposed PrompT-BO approach over existing techniques. The results highlight the effectiveness and efficiency gains afforded by PrompT-BO for prompt tuning tasks.

4.1. Datasets and Baselines

The current study utilizes six standard benchmark datasets to facilitate comparisons with other methods. The datasets are part of the General Language Understanding Evaluation (GLUE) benchmark [32], a collection of resources for training, evaluating, and analyzing natural language understanding systems. The datasets cover various natural language understanding tasks, such as natural language inference, question answering, paraphrase detection, and textual entailment. The datasets are briefly described as follows, and where each dataset refers to a specific task.

MNLI (Multi-Genre Natural Language Inference) is a large-scale dataset for natural language inference whose associated task is determining whether a hypothesis is true, false, or undetermined, given a premise. The dataset covers a range of genres of written and spoken English and has 433,000 sentence pairs annotated with three labels: entailment, contradiction, or neutral.

QQP (Quora Question Pairs) is a dataset of over 400,000 pairs of questions from the community question answering website Quora. The task is to determine whether two questions are semantically equivalent, i.e., whether they can be answered by the same information.

SST-2 (Stanford Sentiment Treebank) is a dataset of 67,000 movie reviews with fine-grained sentiment labels. The task is to predict the sentiment of a given sentence as either positive or negative.

MRPC (Microsoft Research Paraphrase Corpus) is a dataset of 5800 pairs of sentences extracted from online news sources. The task is to identify whether the sentences in each pair are semantically equivalent, i.e., whether they convey the same meaning.

QNLI (Question-answering NLI) is a dataset derived from the Stanford Question Answering Dataset (SQuAD), which consists of over 100,000 questions posed by crowd workers on Wikipedia articles. The task is to determine whether the context sentence contains the answer to the question.

RTE (Recognizing Textual Entailment) is a dataset composed of sentence pairs from various sources, such as news articles and image captions. The task is to determine whether the second sentence is entailed by the first one, i.e., whether the truth of the first sentence guarantees the truth of the second one.

For each dataset, we randomly sampled k data from the original dataset for each class to construct the training set and other different k data to construct the validation set. The original validation set was used as the test set. Because the size of the QQP and RCT validation sets was too large, we randomly sampled 1000 data points to save costs. Different performance metrics were used to evaluate model performance on each task (dataset). For MNLI, SST-2, QNLI, and RTE, the performance score used was accuracy. For QQP and MRPC, we used the F1-Score, which is the harmonic mean of precision and recall.

In order to assess the effectiveness of our proposed approach, the performance was compared with several existing methods that employ various methods for LLMs to prompt optimization on different downstream tasks. All the baselines used a frozen RoBERTa-large model. The baselines are:

ManualPrompt is based on manually composed prompts to conduct the zero-shot evaluation. In this context, it is the only non-automated approach considered among the baselines.

BlackBoxTuning (BBT) [7,8] consider continuous prompts that are optimized by covariance matrix adaptation evolution strategy (black-box). The authors propose a black-box tuning framework to optimize the continuous prompt prepended to the input text via

derivative-free optimization. The experimental results show that BBT outperforms manual prompts, GPT-3's in-context learning, and the gradient-based counterparts.

Reinforcement Learning Prompt (RLPrompt) [21] use an efficient discrete prompt optimization approach with reinforcement learning (RL) that results in a parameter-efficient policy network which generates the optimized discrete prompt after training with reward.

Black-box Discrete Prompt Learning (BDPL) [9] consider discrete prompts that are learned by gradient estimation. BDPL applies a variance-reduced policy gradient algorithm to estimate the gradients of parameters in the categorical distribution of each discrete prompt. The reported experiments on RoBERTa and GPT-3 demonstrate that the proposed algorithm achieves significant improvement on eight benchmarks.

4.2. Experimental Results

For our experiments, we followed the experimental setting reported in [9]. The paper contains a wide set of experimental results, using GPT-3 and the RoBERTa-large model. The black-box settings offered a performance baseline for our experiments. Specifically, the optimization process was performed by maximizing the task-specific performance metrics on an input set X named "training" while another different set was denoted as "evaluation". This procedure was used to avoid possible overfitting of the optimal prompt to the "training" input set.

The RoBERTa model proposed in [25] can be used in different scenarios: text classification, token classification, question answering, language modeling, and multiple choice. The model can be accessed via the Hugging Face library, with each scenario requiring a different model from the library. Our solution utilized the RoBERTa-large model for masked token prediction. Masked language modeling is particularly useful for tasks that require a good contextual understanding of an entire sequence for predicting the masked token (intended, in our implementation, as the target variable).

As an example, we report in Figure 3 on the evolution of the "best seen" (i.e., the observed best performance value) for both the "training" and the "evaluation" sets, over the sequence of generated prompts. An improvement of the best performance over the "training" does not necessarily imply an improvement on the "evaluation". Thus, as suggested in [9], the prompt associated to the best performance on "evaluation" was selected to be tested over another completely different dataset (i.e., the test set). The associated results are reported in Table 1.

Table 1. Summary of results from [9] compared with our approach. In bold, best results (i.e., highest classification performance).

Dataset	MNLI	QQP	SST-2	MRPC	QNLI	RTE	Avg,
BDPL	$42.5_{1.8}$	$\mathbf{56.4}_{1.9}$	$\mathbf{87.6_{2.1}}$	$\mathbf{78.1}_{3.7}$	$\mathbf{53.1}_{1.1}$	$\mathbf{53.5}_{0.9}$	**61.9**
Avg B.B.	**39.8**	53.04	83.26	71.18	52.22	51.08	58.46
PrompT-BO	$29.6_{1.7}$	$53.8_{0.0}$	$86.2_{2.4}$	$\mathbf{78.1}_{4.6}$	$52.9_{1.3}$	$51.0_{1.1}$	58.6

The performance of BO is given in Table 1 for each task (column) and method (row). The last row gives the results of Bayesian Optimization, averaged over three runs for each task, with the relative standard deviation listed in subscript. The Avg B.B. (Black Box) row contains the average performance of ManualPrompt, BBT, RLPrompt, and BDPL.

The performance of BO is significantly worse than the others on MNLI. Possible explanations are that MNLI has the largest vocabulary and that a more sophisticated encoding method than "naïve" continuous relaxation might yield a better result. The best BO is better than the Avg B.B. on most tasks.

Finally, we also report on a comparison between our approach and BDPL [9] in terms of best performance on "evaluation" with respect to runtime (i.e., "time taken"). Results are in Figure 3 for the benchmarks MRPC and RTE.

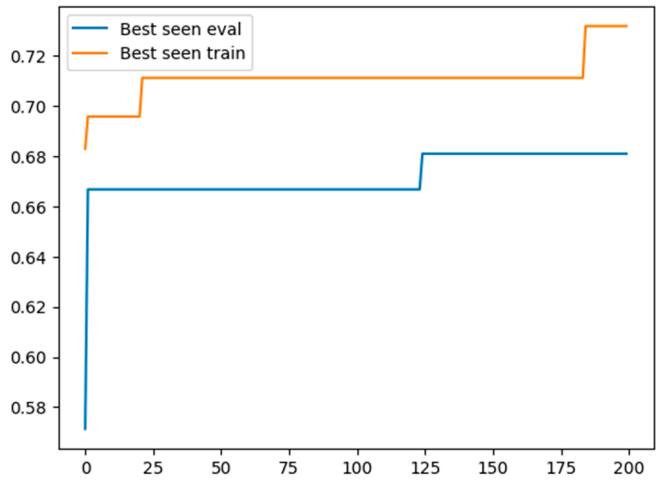

Figure 3. Comparison of best seen performance for BO and BDPL over BO iterations.

The runtime for BDPL had been obtained running the software from [9] on the same machine as our BO. In particular, the machine instance was configured with 2 vCPUs (2.2 GHz), ram 13 GB, and one Tesla T4 GPU (16 GB VRAM). The time reported in Table 2 is the total time resulting from the prompt optimization algorithm and the time derived from the elaboration of the query by the RoBERTa model. In our experiments, the time required for the RoBERTa computation was the prevalent contribution to the overall computational time (Figure 4).

Table 2. Here are two examples where the prompt obtained by BO made correct predictions. The prompt is represented in green and the input in red.

Task	Prompt + Input	Prediction	Label
MRPC	</s>It also said it expects a civil complaint by the Securities and Exchange Commission.</s>Stewart also faces a separate investigation by the Securities and Exchange Commission.?<mask>,</s>	Not semantically equivalent	semantically equivalent
	20 steady especially Hollywood father contracts 130ined That abuse would fired water I key born positions saw 35 shares closed Rice inappropriate Sun missing Associates island cut technology warned Post feel planned training'Ministry Senate vetoed Sharon vulnerability aspects Ms noShe west first statements Republic dismissed regional </s>It also said it expects a civil complaint by the Securities and Exchange Commission.</s>Stewart also faces a separate investigation by the Securities and Exchange Commission.?<mask>,</s>	semantically equivalent	
SST-2	<s> it 'll only put you to sleep. It was<mask>.</s>	great	terrible
	hook windowsallyoder 90 proficiency grandparentsreek metropolitanfredhy feathers 21 uses padding attention notes kids full flush moving debtsomem geout folksvideo rifeat fights famous strengths despair luc irre soft avoid racing black edge aliensrawn bug lob capable struggle di influenceieve <s> it 'll only put you to sleep. It was<mask>.</s>	terrible	
SST-2	<s> can you take before indigestion sets in It was. <mask>.</s>	great	terrible
	hook windowsallyoder 90 proficiency grandparentsreek metropolitanfredhy feathers 21 uses padding attention notes kids full flush moving debtsomem geout folksvideo rifeat fights famous strengths despair luc irre soft avoid racing black edge aliensrawn bug lob capable struggle di influenceieve <s> can you take before indigestion sets in It was. <mask>.</s>	terrible	

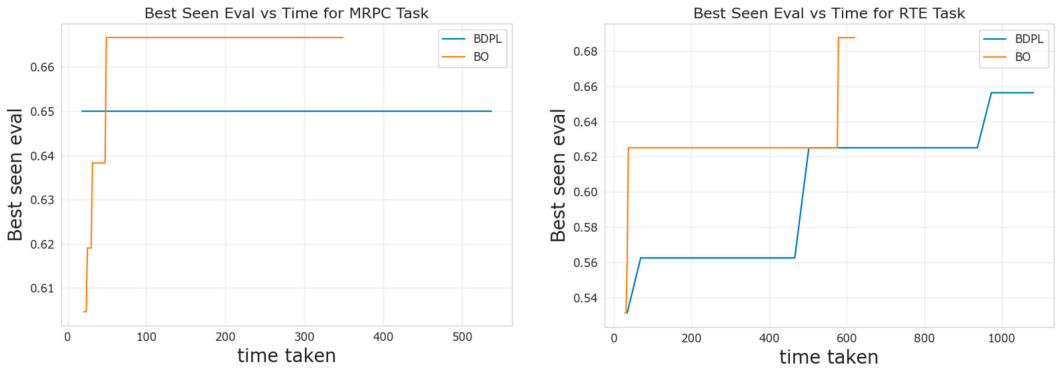

Figure 4. Comparison of best seen performance for BO and BDPL over time (in seconds).

The shorter runtime of the PrompT-BO method on all the tasks (Table 3) may have been motivated by the principled strategy guiding the exploration of the prompt space. This strategy is based on a probabilistic model of the score in PrompT-BO, a Gaussian Process and an acquisition function built upon the probabilistic model, to explore the prompt space. This strategy enables an effective balance of exploration of the search space to gather new information and exploitation to improve over the best observed results; it also endows the Bayesian Optimization with good properties of generalization. Moreover, the computational overhead of BO is less than one second in all the tasks. Therefore, the sampling efficiency of BO comes at almost no additional cost.

Table 3. Runtime in seconds.

Task	PrompT-BO	BDPL
MRPC	379.02	571.75
RTE	661.92	1081.62
QQP	669.52	1070.14
MNLI	896.61	1614.33
QNL	398.96	587.87
SST-2	264.23	384.84

Runtime is an important metric to consider, especially with respect to the societal impact of LLMs and their prompt tuning. Indeed, PrompT-BO, like all other methods, might enable negative applications due to incorrect results in critical decision-making instances. It is therefore important that its implementation comply with ethical safety concerns and that its deployment be aligned with societal goods, such as environmental sustainability. Recently, Sarah Wells, in her article "Generative AI's energy problem today is Foundational" on the IEEE Spectrum (https://spectrum.ieee.org/ai-energy-consumption, accessed on 1 January 2024), argues convincingly that, before AI can take over, it will need to find a novel approach to energy. Specifically, because the training process has been so far removed from the focus of attention, the electricity consumed in making inferences over this time might be globally higher. Indeed, prompt optimization promises to improve the effectiveness of our interaction with LLMs. From the environmental sustainability point of view, with PrompT-BO being able to reduce energy costs through its use of LLMs, PrompT-BO might contribute to a better monitoring of AI environmental sustainability during inference.

Table 4 shows that longer prompts do not necessarily yield better results. An explanation could be that longer prompts might overfit and be less transferable. This agrees with the conclusion in [32]. Finally, more technical details are provided in Appendix A.

Table 4. Comparative results over different prompt lengths for the task MRPC.

Task	Prompt Length	Best Score on Test
MRPC	25	79.7619
MRPC	50	78.4431
MRPC	75	78.4195

5. Conclusions

The main conclusion of this paper is that Bayesian Optimization could become an effective tool for prompt optimization. This vast discrete combinatorial prompt space poses specific challenges for direct optimization. The large discrete prompt search space is converted into a more tractable continuous optimization problem, while still maintaining a correspondence to discrete n-grams through rounding. The continuous representation enables efficient exploration and exploitation over prompts using Gaussian Process-based Bayesian Optimization.

Computational results exhibit a better performance of BO in terms of sample efficiency than other black-box algorithms based on a heuristic search. A reasonable explanation is that BO is based on a principled strategy to guide the exploration of the prompt space.

Author Contributions: Conceptualization, I.G., A.C. and F.A.; Methodology, A.P. and I.G.; Software, A.S. and A.P.; Validation, A.S. and A.P.; Writing—original draft, I.G. and F.A.; Writing—review & editing, I.G. and F.A.; Visualization, A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: Author Ilaria Giordani was employed by the company Oaks srl. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. Oaks srl had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A BoTorch Implementation

Bayesian Optimization is performed using the BoTorch library built on PyTorch 2. A Gaussian Process (GP) model is initialized to represent the objective function using the SingleTaskGP module, which models the objective as a GP with a single output for prompt performance. The ExactMarginalLogLikelihood module computes the exact log marginal likelihood for the GP posterior given the observations. The acquisition function chosen for selecting the next prompt to evaluate is Upper Confidence Bound, which balances exploration and exploitation by maximizing the GP posterior mean plus β times the standard deviation. The search space consists of possible prompt n-gram indices, bounded between 0 and the maximum index normalized. The prompt optimization task involves searching over possible sequences of discrete n-grams to find the optimal prompt for a given task. When using a pre-trained masked language model like RoBERTa-large, the prompts must consist of n-grams from its pre-defined vocabulary.

Specifically, the prompt is represented as a sequence of L discrete indices, with each index corresponding to one of the n-grams in the vocabulary V. The vocabulary V contains $|V|$ possible n-grams, derived from the tokenization process during pre-training. For example, the RoBERTa-large model has a vocabulary with $|V| = 50.265$ n-grams. To construct the candidate prompt vocabulary, we used the script provided by [9] based on the code associated with the study. The search space can therefore be conceptualized as an L-dimensional discrete space, where L is the pre-defined prompt length for the given task.

Each dimension ranges over the possible vocabulary indices $\{0, 1, \ldots, |V| - 1\}$. Hence, the search space cardinality is $|V|^L$, representing all possible prompt sequences of length

L. For example, in the case of MNLI, the prompt length $L = 10$ and $|V| = 117056$, and the cardinality of the prompt search space is $117.056^{10} \approx 4.8 \times 10^{50} \approx 9.2$ times the number of chess positions. The kernel used is the Matern Kernel, $\nu = 5/2$.

Finally, in Table A1, we report on the cardinality of the task-specific vocabularies used in the paper.

Table A1. Cardinality of the vocabulary and prompt length.

	MNLI	**QQP**	**SST-2**	**MRPC**	**QNLI**	**RTE**
Vocab. Size	117,056	61,571	3747	7940	3163	46,992
Prompt Length	10	25	50	50	50	50

References

1. Archetti, A.; Candelieri, A. *Bayesian Optimization and Data Science*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; Volume 849.
2. Garnett, R. *Bayesian Optimization*; Cambridge University Press: Cambridge, UK, 2023.
3. Balandat, M.; Karrer, B.; Jiang, D.; Daulton, S.; Letham, B.; Wilson, A.G.; Bakshy, E. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21524–21538.
4. Brian, L.; Al-Rfou, R.; Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv* **2021**, arXiv:2104.08691.
5. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
6. Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.L.; Cao, Y.; Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *arXiv* **2023**, arXiv:2305.10601.
7. Sun, T.; He, Z.; Qian, H.; Zhou, Y.; Huang, X.J.; Qiu, X. BBTv2: Towards a gradient-free future with large language models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 11 December 2022; pp. 3916–3930.
8. Sun, T.; Shao, Y.; Qian, H.; Huang, X.; Qiu, X. Black-box tuning for language-model-as-a-service. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 20841–20855.
9. Diao, S.; Huang, Z.; Xu, R.; Li, X.; Lin, Y.; Zhou, X.; Zhang, T. Black-box prompt learning for pre-trained language models. *arXiv* **2022**, arXiv:2201.08531.
10. Chai, Y.; Wang, S.; Sun, Y.; Tian, H.; Wu, H.; Wang, H. Clip-tuning: Towards derivative-free prompt learning with a mixture of rewards. *arXiv* **2022**, arXiv:2210.12050.
11. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. Lora: Low-rank adaptation of large language models. *arXiv* **2021**, arXiv:2106.09685.
12. Li, X.L.; Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv* **2021**, arXiv:2101.00190.
13. Zhong, Z.; Friedman, D.; Chen, D. Factual probing is [mask]: Learning vs. learning to recall. *arXiv* **2021**, arXiv:2104.05240.
14. Wen, Y.; Jain, N.; Kirchenbauer, J.; Goldblum, M.; Geiping, J.; Goldstein, T. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *arXiv* **2023**, arXiv:2302.03668.
15. Prasad, A.; Hase, P.; Zhou, X.; Bansal, M. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv* **2022**, arXiv:2203.07281.
16. Pryzant, R.; Iter, D.; Li, J.; Lee, Y.T.; Zhu, C.; Zeng, M. Automatic prompt optimization with "gradient descent" and beam search. *arXiv* **2023**, arXiv:2305.03495.
17. Guo, Q.; Wang, R.; Guo, J.; Li, B.; Song, K.; Tan, X.; Liu, G.; Bian, J.; Yang, Y. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv* **2023**, arXiv:2309.08532.
18. Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q.V.; Zhou, D.; Chen, X. Large language models as optimizers. *arXiv* **2023**, arXiv:2309.03409.
19. Shin, T.; Razeghi, Y.; Logan, R.L., IV; Wallace, E.; Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv* **2020**, arXiv:2010.15980.
20. Shi, W.; Han, X.; Gonen, H.; Holtzman, A.; Tsvetkov, Y.; Zettlemoyer, L. Toward Human Readable Prompt Tuning: Kubrick's The Shining is a good movie, and a good prompt too? *arXiv* **2022**, arXiv:2212.10539.
21. Deng, M.; Wang, J.; Hsieh, C.P.; Wang, Y.; Guo, H.; Shu, T.; Song, M.; Xing, E.; Hu, Z. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv* **2022**, arXiv:2205.12548.
22. Zhang, T.; Wang, X.; Zhou, D.; Schuurmans, D.; Gonzalez, J.E. Tempera: Test-time prompt editing via reinforcement learning. In Proceedings of the Eleventh International Conference on Learning Representations, Virtual, 25–29 April 2022.
23. Shen, M.; Ghosh, S.; Sattigeri, P.; Das, S.; Bu, Y.; Wornell, G. Reliable gradient-free and likelihood-free prompt tuning. *arXiv* **2023**, arXiv:2305.00593.
24. Chen, L.; Chen, J.; Goldstein, T.; Huang, H.; Zhou, T. InstructZero: Efficient Instruction Optimization for Black-Box Large Language Models. *arXiv* **2023**, arXiv:2306.03082.

25. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
26. Maus, N.; Chao, P.; Wong, E.; Gardner, J.R. Black box adversarial prompting for foundation models. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*; IBM: Armonk, New York, NY, USA, 2023.
27. Sabbatella, A.; Ponti, A.; Candelieri, A.; Giordani, I.; Archetti, F. A Bayesian approach for prompt optimization in pre-trained language models. *arXiv* **2023**, arXiv:2312.00471.
28. de Curtò, J.; de Zarzà, I.; Roig, G.; Cano, J.C.; Manzoni, P.; Calafate, C.T. LLM-Informed Multi-Armed Bandit Strategies for Non-Stationary Environments. *Electronics* **2023**, *12*, 2814. [CrossRef]
29. Liu, T.; Astorga, N.; Seedat, N.; van der Schaar, M. Large Language Models to Enhance Bayesian Optimization. *arXiv* **2024**, arXiv:2402.03921.
30. Dale, E.; Chall, J.S. A formula for predicting readability: Instructions. *Educ. Res. Bull.* **1948**, *27*, 37–54.
31. Gunning, T. The art of succession: Reading, writing, and watching comics. *Crit. Inq.* **2014**, *40*, 36–51. [CrossRef]
32. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv* **2018**, arXiv:1804.07461.