Genetic Prompt Search via Exploiting Language Model Probabilities

Jiangjiang Zhao^{1,2} , **Zhuoran Wang**^{3} , **Fangchun Yang**^{1}
^{1}Beijing University of Posts and Telecommunications, P.R. China
^{2}China Mobile Online Services Co., Ltd. Beijing, P.R. China
^{3}Clouchie Limited, London, United Kingdom
zjjbupt@bupt.edu.cn, wangzhuoran@clouchie.ai, fcyang@bupt.edu.cn

Abstract

Prompt tuning for large-scale pretrained language models (PLMs) has shown remarkable potential, especially in low-resource scenarios such as few-shot learning. Moreover, derivative-free optimisation (DFO) techniques make it possible to tune prompts for a black-box PLM to better fit downstream tasks. However, there are usually preconditions to apply existing DFO-based prompt tuning methods, e.g. the backbone PLM needs to provide extra APIs so that hidden states (and/or embedding vectors) can be injected into it as continuous prompts, or carefully designed (discrete) manual prompts need to be available beforehand, serving as the initial states of the tuning algorithm. To waive such preconditions and make DFO-based prompt tuning ready for general use, this paper introduces a novel genetic algorithm (GA) that evolves from empty prompts, and uses the predictive probabilities derived from the backbone PLM(s) on the basis of a (few-shot) training set to guide the token selection process during prompt mutations. Experimental results on diverse benchmark datasets show that the proposed precondition-free method significantly outperforms the existing DFO-style counterparts that require preconditions, including black-box tuning, genetic prompt search and gradient-free instructional prompt search.

1 Introduction

The recent successes of pretrained language models (PLMs) are revolutionising the field of natural language processing (NLP) [Devlin *et al.*, 2019; Radford *et al.*, 2019; Liu *et al.*, 2019; Raffel *et al.*, 2020; Clark *et al.*, 2020]. Meanwhile, extremely large PLMs have demonstrated great potential in few-shot learning scenarios (e.g. [Brown *et al.*, 2020]), which makes them increasingly attractive as out-of-box tools for general use. Fine-tuning such large-scale PLMs can still be computationally expensive, even on a few-shot training set. But this is significantly relieved by a new paradigm called prompt tuning. Prompt tuning methods [Li and Liang, 2021; Gao *et al.*, 2021; Lester *et al.*, 2021; Shin *et al.*, 2020; Liu *et al.*, 2022; Liu *et al.*, 2021a; Liu *et al.*, 2023] work by inserting a small number of tunable variables into a backbone PLM's input (and sometimes also the hidden states [Liu *et al.*, 2021a]) to bias its predictive probabilities towards desired output, while keeping the backbone model's parameters frozen during the learning process. The tunable variables here can either be continuous vectors (namely soft prompts) or surface tokens (discrete prompts), for which obtaining promising values is the essential objective and can be solved by either gradient-based optimisers [Kingma and Ba, 2015; Loshchilov and Hutter, 2019] or derivative-free optimisation (DFO) techniques [Kolda *et al.*, 2003; Rios and Sahinidis, 2013; Yu and Gen, 2010].

To democratise the access to those very large PLMs, a common practice is to deliver them as back-box services with cloud APIs only [Brown *et al.*, 2020; Ouyang *et al.*, 2022] (a.k.a LM-as-a-Service [Sun *et al.*, 2022b]). This implies prompt tuning via DFO to be an senseful and important research direction, where the backbone's parameters are not required to be exposed to the tuner. Existing work has proven the feasibility of applying DFO for prompt tuning [Xu *et al.*, 2022; Prasad *et al.*, 2022; Sun *et al.*, 2022b; Sun *et al.*, 2022a]. For example, Xu *et al.* [2022] and Prasad *et al.* [2022] introduced search heuristics to refine human generated (discrete) prompts via edit operations such as paraphrase, cloze, deletion, swap, etc. On the other hand, Sun *et al.* [2022b; 2022a] proposed the so-call 'black-box tuning (BBT)' methods for continuous prompt optimisation based on the covariance matrix adaptation evolution strategy [Hansen *et al.*, 2003].

However, the limitations of existing methods are obvious. The discrete prompt search heuristics in [Xu *et al.*, 2022; Prasad *et al.*, 2022] require carefully designed manual prompts available beforehand, which yield additional human efforts. Furthermore, their performance may also highly rely on the suitableness of those initial manual prompts (cf. §4.3). In addition, methods of this kind usually require auxiliary language models (LMs) to paraphrase existing prompts (e.g. T5_{11B} [Raffel *et al.*, 2020] used in [Xu *et al.*, 2022] and PEGASUS [Zhang *et al.*, 2020] used in [Prasad *et al.*, 2022]), which involves extra dependencies. For BBT-style continuous prompt optimisation [Sun *et al.*, 2022b; Sun *et al.*, 2022a], it requires the backbone PLM to provide extra APIs so that the continuous prompts can be injected as word embeddings (or hidden states for BBTv2 [Sun *et* *al.*, 2022a]), which may not always be attainable in practice. A naïve alternative is in-context learning (ICL) proposed along with GPT-3 [Brown *et al.*, 2020], which simply prepends training examples to the input as the prompt. But prepending too many or too long examples could make the final input exceed the sequence length threshold of the backbone model, which restricts ICL's applicable tasks. In addition, the performance of ICL is usually less competitive, as seen in [Sun *et al.*, 2022b; Sun *et al.*, 2022a; Xu *et al.*, 2022; Gao *et al.*, 2021] and also in our experiments (cf. §4.3).

This paper aims to waive the above preconditions required by the existing DFO-based prompt tuning methods, and presents a novel genetic algorithm (GA) [Mitchell, 1998] that generates discrete prompts from the ground up. The proposed method, named **G**eneric **A**lgorithm for **P**redictive **P**robability guided **P**rompting (**GAP3**)[1], works as follows. Firstly, discontinuous prompt chunks are considered as chromosomes, with prompt tokens being genes. Then, starting from an empty one, GAP3 evolves the prompts via chromosome crossovers and gene mutations. At each mutation step, either a new *mask* token is inserted into a random chromosome at a random position, or a random existing gene is masked. After this, the masked slot will be filled by a token that approximately maximises the predictive probability of the ground-truth labels on a (few-shot) training set. The algorithm iterates for a predefined number of steps, with individuals consisting of diverse chromosomes/genes competing to survive and breed, according to their fitness scores computed on the training set.

In comparison to existing DFO-based prompting methods, the major advantages of the proposed GAP3 are three-fold:

1. No extra APIs for vector injections are required, since GAP3 searches for discrete prompts.
2. No manual prompts are required, as GAP3 is initialised with an empty prompt.
3. For backbone PLMs that yield predictive probabilities for masked tokens, e.g. masked language models (MLMs) [Devlin *et al.*, 2019; Liu *et al.*, 2019] or T5-style encoder-decoder networks [Raffel *et al.*, 2020; Lewis *et al.*, 2020], GAP3 generates prompt tokens directly based on the backbone itself, without needing an auxiliary LM, which further reduces preconditions for its applications. (If a casual LM [Radford *et al.*, 2018; Radford *et al.*, 2019] is the backbone of interest, GAP3 will need an auxiliary MLM for token generation (cf. §4.3), where the MLM can also be a black-box model.)

Taking RoBERTa$_{\text{LARGE}}$ [Liu *et al.*, 2019] and GPT-2$_{\text{LARGE}}$ [Radford *et al.*, 2019] as the backbone PLMs of interest, respectively, the performance of GAP3 is evaluated on 7 benchmark datasets (the same as those used in [Sun *et al.*, 2022b]), in comparison with that of the existing counterparts, including BBT [Sun *et al.*, 2022b], genetic prompt search (GPS) [Xu *et al.*, 2022], gradient-free instructional prompt search (GRIPS) [Prasad *et al.*, 2022], as well as ICL [Brown *et al.*, 2020]. The experimental results prove the effectiveness of the proposed GAP3, where it outperforms all the DFO-style baselines, achieving at least 2.9% and 2.4% absolute improvements in the average scores for RoBERTa$_{\text{LARGE}}$ and GPT-2$_{\text{LARGE}}$, respectively.

[1]Code and supplementary material available at: https://github.com/zjjhit/gap3

2 Related Work

Parameter-efficient tuning (PET). PET reduces the cost of adapting a large PLM to downstream tasks, by tuning only a small proportion of the parameters instead of the full model [Houlsby *et al.*, 2019; Pfeiffer *et al.*, 2020]. Prompt tuning [Lester *et al.*, 2021; Liu *et al.*, 2022; Liu *et al.*, 2021a; Liu *et al.*, 2021b; Qin and Eisner, 2021] form a sub-direction of PET, where the tunable parameters are the injected soft prompts. Despite the success of PET methods, they are not suitable for the growing trend of LM-as-a-Service deployments, as gradient-based optimisations are required.

Discrete prompt search. Discrete prompts are more preferable in black-box scenarios, since no model-level modifications are involved. Manually created intuitive prompts were found helpful in earlier research [Petroni *et al.*, 2019; Schick and Schütze, 2021b; Schick and Schütze, 2021a], but they are suboptimal in a general sense. For methods that searches for prompts automatically, paraphrasing is a commonly used methodology to expand the existing prompt set (usually initialised with manual prompts) for succeeding search heuristics to winnow [Xu *et al.*, 2022; Prasad *et al.*, 2022]. In addition, Hou *et al.* [2022] proposed to ensemble prompts via boosting. But such ensemble multiplies the inference cost at the same time. Reinforcement learning (RL) has recently been employed by Deng *et al.* [2022] and Diao *et al.* [2023] to optimise discrete prompts for black-box backbones. It's worth noting that the method proposed in Deng *et al.* [2022] involves action explorations in a considerably large prompt space, which results in significantly more API calls to train the model than its counterparts (such as BBT [Sun *et al.*, 2022b]). Other proposed methodologies include mining prompt templates from the web [Jiang *et al.*, 2021] and training specific prompt generators [Ben-David *et al.*, 2022], which correspond to extra computational and human efforts.

Hybrid methods. It is also possible to tune a backbone model based on both discrete prompts and differentiable parameters, of which typical examples include AutoPrompt [Shin *et al.*, 2020] who search for discrete prompt tokens based on gradients, and LM-BFF [Gao *et al.*, 2021] that combines automatic prompt generation and model fine-tuning. Methods of this kind should be considered as more comparable to PET, as they violate the black-box assumption.

3 Methodology

3.1 Prompt Template

Assume that the downstream task is to classify the input text `[X]` to a label `[Y]`. A prompt template here means a permutation to arrange `[X]`, `[Y]` and the prompt `[T]`, e.g. template '`[X][T][Y]`' stands for placing the prompt between the input text and the label. We further assume that a task's input may consist of multiple text pieces (e.g. when classifying a pair of sentences) and prompt chunks are allowed to be placed at multiple positions. Therefore, we generalise the above template representation with subscripts, e.g. '`[X`$_1$`][T`$_1$`][X`$_2$`][T`$_2$`][Y]`' means two prompt chunks inserted between two text pieces and between the second text piece and the label, respectively.

3.2 Genetic Algorithm for Prompting

For a given downstream task and a training set, the proposed GAP3 works as follows. Firstly, we define a prompt template (cf. §3.1). The template here only decides a permutation of the text pieces. The actually prompt chunks are initialised as empty sets at this stage. To interpret the process in GA-style, we regard each prompt chunk `[T`$_\mathrm{i}$`]` as a *chromosome*, where the actual tokens to be filled into it are regarded as *genes*. In addition, we call entire prompts with diverse chromosomes *individuals*. After this, we generate the first population of N individuals by randomly mutate one gene of the initial empty individual, where N is a predefined hyperparameter. Concretely, in this very first step, the mutation means randomly selecting a chromosome (that is blank), and inserting a token to it. Then the algorithm iterates as follows.

1. Evaluate each individual on the training set, to obtain a fitness score;
2. Keep top $\sqrt{N}$ most fit individuals, called elites;
3. Randomly draw pairs of individuals from the elites to perform (probabilistic) chromosome crossovers, until a new population of size N is yielded.
4. Mutate each new individual's gene (probabilistically) by randomly inserting a new token or replacing a random existing token with a new one;

The above process will be repeated for M steps before the best individual being chosen as the final output, where M is also a predefined hyperparameter.

Elite selection. Besides ranking the current population of individuals according to their fitness scores, we also maintain the overall best individual achieved in the previous iterations. If the 'best so far' individual surpasses the current elites, we will add it into the current elite group (while dropping the tail elite). This is to avoid good genes being lost during the crossovers and mutations.

Crossover. Elite pairs are drawn in a weighted roulette wheel manner, with respect to their fitness scores. After this, for each `[T`$_\mathrm{i}$`]`, with probability ρ_c, we exchange the corresponding chromosomes between an elite pair, to yield two new individuals. The hyperparameter ρ_c is called the crossover probability.

Mutation. With probability ρ_m (called the mutation probability), we mutate an individual. When this happens, we randomly draw an action between 'insert' or 'replace' with even probabilities. If the 'insert' action is applied, a *mask* token will be inserted at a random position of a random chromosome. Otherwise, a random existing gene (of a random chromosome) will be masked (by replacing that token with a *mask* token). This implies, in either way, there will be one *mask* token in the current prompt. Then, LM probabilities will be exploited to realise that *mask* to a real token.

Algorithm 1 GAP3

```
Input:
  $D$ – training set
  $T_0$ – initial (empty) template
  $LM$ – backbone model
Hyperparameter:
  $M$ – number of iterations
  $N$ – population size
  $\rho_c$ – crossover probability
  $\rho_m$ – mutation probability
Output: T* – the most fit prompt
 1: G ← { }, $n \leftarrow \mathrm{int}(\sqrt{N})$, T* ← NULL
 2: for $1 \dots N$ do
 3:     T ← $T_0$
 4:     T ← mutate(T, $D$, $LM$, prob=1.0)
 5:     G ← G ∪ {T}
 6: end for
 7: for $1 \dots M$ do
 8:     G ← sort( G, key=fitness(G, $D$, $LM$) )
 9:     if better_than(G[0], T*) then
10:         T* ← G[0], E ← G[0 : $n$]
11:     else
12:         E ← {T*} ∪ G[0 : $n - 1$]
13:     end if
14:     G′ ← { }
15:     while |G′| < $N$ do
16:         T, T′ ← roulette( E, weights=fitness(E, $D$, $LM$) )
17:         T, T′ ← crossover(T, T′, prob=$\rho_c$)
18:         T ← mutate(T, $D$, $LM$, prob=$\rho_m$)
19:         T′ ← mutate(T′, $D$, $LM$, prob=$\rho_m$)
20:         G′ ← G′ ∪ {T, T′}
21:     end while
22:     G ← G′
23: end for
24: return T*
```

Algorithm 1 gives the pseudo-code of the proposed GAP3, where hyperparameters and constant objects are denoted in *italic* type. We leave the detailed explanations of the **token selection process** and the consequential design of the **fitness function** to §3.3 and §3.4, respectively, to keep the discussions on the main algorithm coherent here.

3.3 Mutation Guided by LM Probabilities

Notations. Let $(\mathbf{x}, y)$ denote a data example, where $\mathbf{x}$ is the input token sequence, and y is the label (token). Without loss of generality, for tasks involving m pieces of text as input, we will let $\mathbf{x} := (\mathbf{x}_1, \dots, \mathbf{x}_m)$, with each $\mathbf{x}_i$ consisting of a token sequence. Then a prompt $\mathcal{T}$ can be regarded as a function that applies the prompt tokens to the data point $(\mathbf{x}, y)$, according to its associated template, to obtain a final token sequence, as $\mathcal{T}(\mathbf{x}, y)$. We use t_i to denote the token indexed by i in $\mathcal{T}$, and let $\mathcal{T}_i^{\#}$ stand for the prompt with t_i replaced by a *mask* token. We also use $t_i^{\#}$ to refer the i-indexed masked token. Similarly, $(\mathbf{x}, y^{\#})$ stands for the data point its label y masked.

Firstly, assume we have an 'ideal' LM that satisfies the Bayes' rule. Then, for an arbitrary data point and an arbitrarily masked token $t_i^\#$, we would have:

$$\begin{aligned} &P(t_i^\# = t|\underbrace{\mathcal{T}_i^\#(\mathbf{x}, y^\#), y^\# = y}_{\mathcal{T}_i^\#(\mathbf{x},y)})P(y^\# = y|\mathcal{T}_i^\#(\mathbf{x}, y^\#)) \\ &= P(y^\# = y|\underbrace{\mathcal{T}_i^\#(\mathbf{x}, y^\#), t_i^\# = t}_{\mathcal{T}_{i\leftarrow t}^\#(\mathbf{x},y^\#)})P(t_i^\# = t|\mathcal{T}_i^\#(\mathbf{x}, y^\#)) \end{aligned} \tag{1}$$

which yields:

$$\begin{aligned} &P(y^\# = y|\mathcal{T}_{i\leftarrow t}^\#(\mathbf{x}, y^\#)) \\ &= \frac{P(t_i^\# = t|\mathcal{T}_i^\#(\mathbf{x}, y))P(y^\# = y|\mathcal{T}_i^\#(\mathbf{x}, y^\#))}{P(t_i^\# = t|\mathcal{T}_i^\#(\mathbf{x}, y^\#))} \end{aligned} \tag{2}$$

where $P(\cdot|\cdot)$ stands for the conditional probability given by the LM, and $\mathcal{T}_{i\leftarrow t}^\#$ stands for the prompt obtained by substituting the masked token $t_i^\#$ with token t. We call Eq. 1 an 'ideal' assumption, because it holds if and only if the probabilities here are ground truth probabilities, which will not be achievable in practice (since general PLMs are not trained subject to such a constraint to satisfy the Bayes' rule).

Therefore, to make the equation valid, we introduce a bias item λ and reformulate Eq. 2 in logarithmic form, as:

$$\begin{aligned} &\log P_{(y|t)}(y^\# = y|\mathcal{T}_{i\leftarrow t}^\#(\mathbf{x}, y^\#)) \\ &= \log P_{(t|y)}(t_i^\# = t|\mathcal{T}_i^\#(\mathbf{x}, y)) \\ &+ \log P_{(y|*)}(y^\# = y|\mathcal{T}_i^\#(\mathbf{x}, y^\#)) \\ &- \log P_{(t|*)}(t_i^\# = t|\mathcal{T}_i^\#(\mathbf{x}, y^\#)) - \lambda(\mathbf{x}, y, t, \mathcal{T}_i^\#) \end{aligned} \tag{3}$$

where we name the probabilities in the form of $P_{(\cdot|\cdot)}$, for the ease of reference in future discussions. Now, recall the mutation process in §3.2. Given an arbitrary prompt $\mathcal{T}_i^\#$ with a masked token, and a training set D, one would want to unmask $\mathcal{T}_i^\#$ by seeking the token $\hat{t}$ that maximising the label posterior on D, as:

$$\hat{t} = \arg\max_{t\in\mathcal{V}} \sum_{(\mathbf{x},y)\in D} \log P_{(y|t)}(y^\# = y|\mathcal{T}_{i\leftarrow t}^\#(\mathbf{x}, y^\#)) \tag{4}$$

where $\mathcal{V}$ is the vocabulary of the LM. If we omit the biases (λs), $P_{(y|t)}$ can be computed tractably based on right-hand side of Eq. 3, which implies invoking the LM twice for each training example $(\mathbf{x}, y)$, by feeding it with $\mathcal{T}_i^\#(\mathbf{x}, y^\#)$ and $\mathcal{T}_i^\#(\mathbf{x}, y)$, respectively. However, the biases (λs) here are indispensable, while computing them for all possible data-prompt-token-mask combinations is obviously intractable. Hence, we design a heuristic to address this, as follows.

Let $\bar{\lambda}(t) := \max_{(\mathbf{x},y)\in D,\mathcal{T}_i^\#} \lambda(\mathbf{x}, y, t, \mathcal{T}_i^\#)$. If we replace $\lambda(\mathbf{x}, y, t, \mathcal{T}_i^\#)$ in Eq. 3 with $\bar{\lambda}(t)$, the left-hand side becomes a lower bound of the original $\log P_{(y|t)}$. Firstly, we initialise $\bar{\lambda}(t)$ as 0 for all $t \in \mathcal{V}$. We define $\log \bar{P}_{(y|t)} := \log P_{(t|y)} + \log P_{(y|*)} - \log P_{(t|*)} - \bar{\lambda}(t)$ (represented in simplified notations). Then, at each time a prompt is mutated, we perform a two-step update, as:

$$\hat{t} \quad \leftarrow \quad \arg\max_{t\in\mathcal{V}} \sum_D \log \bar{P}_{(y|t)}; \tag{5}$$

$$\bar{\lambda}(\hat{t}) \quad \leftarrow \quad \max\left[\bar{\lambda}(\hat{t}), \max_D \left(\log \bar{P}_{(y|\hat{t})} - \log P_{(y|\hat{t})}\right)\right]. \tag{6}$$

where $\log P_{(y|\hat{t})}$ is also computed based on the LM. Note here, $P_{(y|\hat{t})}$ essentially evaluates the performance of the obtained prompt on the training data, as is also used by the fitness function (cf. §3.4). Therefore, computing it is an inevitable effort, instead of an extra cost.

Remark on $\bar{\lambda}$. A more intuitive explanation of $\bar{\lambda}(t)$'s function is that, it penalises those tokens who tend to occur repeatedly but will overestimate the predictive probabilities of the labels.

Remark on Eq. 5. In practice, always adopting the top-1 token $\hat{t}$ may yield duplicated prompts (especially when generating the initial population (cf. §3.2)). Therefore, we actually collect top-n tokens based on $\sum_D \log \bar{P}_{(y|t)}$. Then, starting from the first one, we examine whether the prompt unmasked based on the current token has already been seen previously. If yes, we move to the next token, until an unseen consequential prompt is obtained.

Masked LM vs. casual LM. The most elegant part of the proposed GAP3 is that, if the backbone PLM is an MLM (or a T5-like encoder-decoder network [Raffel *et al.*, 2020; Lewis *et al.*, 2020]), all the four predictive probabilities $P_{(t|y)}, P_{(y|*)}, P_{(t|*)}$ and $P_{(y|t)}$ can be obtained from the backbone model itself. Nevertheless, if the backbone is a casual LM who can only predict $P_{(y|t)}$, we can use an auxiliary MLM to compute $P_{(t|y)}, P_{(y|*)}$ and $P_{(t|*)}$, in which case, $\bar{\lambda}(t)$ to a great extent prevents the algorithm repeatedly generating tokens highly biased to the auxiliary.

3.4 Fitness Function

For a given task, the actual objective metric (such as accuracy or F$_1$-score) on the training set will be a straightforward measure of the fitness for those individuals yielded in the GA. However, in the few-shot learning case, it will be very easy to have many individuals achieving a same metric score. Too many indistinguishable individuals occurring in a population may result in less chance of breeding given to those potentially more competitive genes.

Therefore, in GAP3, we actually make the fitness measure two-dimensional. The task-specific objective metric is the dominant fitness. If (and only if) two individuals have an equal score in the dominant fitness, we further compare them according to a secondary fitness. The secondary fitness score is computed as:

$$\begin{aligned} F_{\text{2nd}}(\mathcal{T}) &= \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \delta_{y,\hat{y}} \frac{P(y^\# = y|\mathcal{T}(\mathbf{x}, y^\#))}{\sum_{y'\in\mathcal{Y}} P(y^\# = y'|\mathcal{T}(\mathbf{x}, y^\#))} \\ \hat{y} &= \arg\max_{y'\in\mathcal{Y}} P(y^\# = y'|\mathcal{T}(\mathbf{x}, y^\#)) \end{aligned} \tag{7}$$

where $\mathcal{Y}$ denotes the task's label set, $\delta_{\cdot,\cdot}$ is the Kronecker delta function. F_{2nd} means that we renormalise the predictive probabilities on the label set, and average over the 'hinge' probabilities, where incorrectly predicted examples contribute zero values. Note here, in the weighted roulette wheel selection process, we only use F_{2nd} scores for the weights, as they are more distinguishable and partially reflects the classification accuracy.

4 Experiments

We conduct a group of main comparative experiments and an ablation study. §4.1, §4.2 and §4.3 describe the settings, baselines and results for the main experiments, respectively. Those for the ablation study is presented in §4.4, specifically.

4.1 Settings

Datasets. The datasets used in the main experiments consist of 7 benchmark NLP tasks, which are the same as in [Sun *et al.*, 2022b], including Yelp polarity, AG's News and DBPedia from [Zhang *et al.*, 2015], SST-2, MRPC and RTE from the GLUE benchmarks [Wang *et al.*, 2018], as well as SNLI [Bowman *et al.*, 2015]. The experiments are in a k-shot learning setting, where for each task, we randomly sample $k = 16$ examples for each label from the original training set. For SST-2, MRPC and RTE, we use their development sets as the test sets. For the other tasks, we use the original test sets. For MRPC, F_1 is used as the evaluation metric, while accuracy is the metric for all the other tasks.

Backbone PLMs. We choose RoBERTa$_{\text{LARGE}}$ [Liu *et al.*, 2019] and GPT-2$_{\text{LARGE}}$ [Radford *et al.*, 2019] as backbones, conducting two groups of experiments, respectively. For the GPT-2$_{\text{LARGE}}$ backbone, we use RoBERTa$_{\text{LARGE}}$ and BERT$_{\text{BASE}}$ (cased) [Devlin *et al.*, 2019], respectively, as the auxiliaries for GAP3 (cf. §3.3).

Hyperparameters for GAP3. We set GAP3's population size $N = 64$ and iteration number $M = 50$, with crossover and mutation probabilities $\rho_c = 0.5$ and $\rho_m = 0.75$, respectively. For the RoBERTa$_{\text{LARGE}}$ backbone, the above settings result in $M \times N \times (1 + 2\rho_m) = 8000$ expected number of API calls (one call for fitness plus two calls for mutation (with probability ρ_m) per individual per data example). For the GPT-2$_{\text{LARGE}}$ backbone, the same settings correspond to 3200 API calls to GPT-2$_{\text{LARGE}}$ and 4800 to RoBERTa$_{\text{LARGE}}$/BERT$_{\text{BASE}}$. The prompt templates and label words for GAP3 on each task can be found in appendix A.

4.2 Baselines

We choose the following existing DFO-based prompting methods as baselines. The implementations of the baselines are all based on the original source code provided by their authors.

BBT. BBT [Sun *et al.*, 2022b] requires for an additional k-shot development set, which is also randomly sampled from the original training sets of the tasks, without overlapping with the k-shot training examples. BBT's budget for API calls is set to 8000 (the same as in [Sun *et al.*, 2022b]), with the prompt length 50.

GPS. GPS [Xu *et al.*, 2022] is another GA-based prompting method that evolves by using T5$_{\text{11B}}$ [Raffel *et al.*, 2020] to paraphrase the prompts. As GPS requires multiple manual prompts to initialise the first population, we only conduct experiments for it on Yelp polarity, AG's News, SNLI and RTE. We use the manual prompts presented in [Schick and Schütze, 2022] to initialise the Yelp polarity and AG's News experiments. Experiments on SNLI are initialised with the ANLI manual prompts presented in [Sanh *et al.*, 2022]. Sanh *et al.* [2022] also provides a manual prompt set for RTE, which is directly adopted here. We set GPS' population size to 25 and number of iterations to 10.

GRIPS. GRIPS [Prasad *et al.*, 2022] is also a heuristic search based prompting method, which evolve prompts based on PEGASUS [Zhang *et al.*, 2020] and generic token-level edits. GRIPS requires manual instructions to initialise. For the GPT-2$_{\text{LARGE}}$ experiments, we initialise it based on the Natural-Instructions dataset [Mishra *et al.*, 2022], and set the iteration number to 50 with 100 candidates generated per iteration. However, based on the same settings, we failed to obtain any reasonable results for the RoBERTa$_{\text{LARGE}}$ backbone. GRIPS either fails to find a valid update or yields results no better than chance. (This may be because either GRIPS itself or the initial instructional manual prompt is unsuitable for an MLM backbone.) Therefore, we omit the comparison with GRIPS for the RoBERTa$_{\text{LARGE}}$ backbone.

In addition, we also compare our GAP3's performance with that of **ICL** [Brown *et al.*, 2020], **manual prompts**, gradient-based **prompt tuning (PT)** [Li and Liang, 2021] as well as **full-model fine-tuning (FT)**. For **ICL**, we concatenate the k-shot training examples in a random order (but with a balanced label distribution) to form a prefix prompt for the input. In regard to **manual prompts**, for Yelp polarity, AG's News, SNLI and RTE, we test all the available prompts and choose the best score. For the other tasks, we just use the simple prompt templates given in [Sun *et al.*, 2022b]. For **PT** and **full-model FT**, Adam optimisers [Kingma and Ba, 2015] are employed. For PT, with learning rate 5e-4 and batch size 16, it runs for 1000 epochs. For full-model FT, with the same batch size, but learning rate 1e-5, we run it for 200 epochs. We did not choose BBTv2 [Sun *et al.*, 2022a] as a baseline in this work, because we consider hidden-state injection as a much stronger violation to the black-box assumption, which will be unfair to other methods.

Label words. We use label words slightly different from those in [Sun *et al.*, 2022b]. Because in our case, we intuitively expect the label words more substitutable to each other from natural language point of view. In order to eliminate the bias in the experimental results caused by this difference, for BBT and ICL, we experiment them with both the label words in [Sun *et al.*, 2022b] and ours, and choose the better results obtained. For GPS and GRIPS, the label words are embedded in their initial manual prompts, which we keep unchanged.

Fairness of resources used. For the RoBERTa$_{\text{LARGE}}$ backbone, our GAP3 and BBT are compared under the same API call budget. However, BBT uses an additional k-shot development set, which means doubling the number of labelled examples required. If we assume the general cost of an API call to be linear to the scale of the model behind, for the GPT-2$_{\text{LARGE}}$ backbone, GAP3 would be much more cost-efficient than BBT, as 60% of its API calls are spent on the auxiliary RoBERTa$_{\text{LARGE}}$/BERT$_{\text{BASE}}$ that is much smaller than GPT-2$_{\text{LARGE}}$. Based on the same assumption, we can consider the resources used by GAP3, GPS and GRIPS as approximately comparable.

Method	SST-2 acc	Yelp P. acc	AG's News acc	DBPedia acc	MRPC f_1	SNLI acc	RTE acc	Avg.
				Gradient-based methods				
PT [Li and Liang, 2021]	78.3(8.4)	91.7(1.2)	79.8(0.6)	88.5(0.7)	55.0(5.3)	39.3(2.2)	52.0(1.9)	69.2
Full-model FT	89.6(0.7)	<u>96.2</u>(0.4)	<u>87.7</u>(0.7)	<u>98.0</u>(0.7)	73.7(9.9)	<u>74.6</u>(4.2)	<u>69.1</u>(1.5)	<u>84.1</u>
				Gradient-free methods				
Manual	79.7	92.6	79.2	41.3	<u>**80.2**</u>	36.0	51.6	65.8
ICL [Brown *et al.*, 2020]	70.1(9.6)	53.1(0.6)	62.7(14.0)	39.4(8.4)	49.1(5.1)	35.8(2.2)	48.9(4.5)	51.3
GPS [Xu *et al.*, 2022]	–	87.5(1.2)	76.3(3.5)	–	–	37.7(2.1)	**51.9**(4.5)	63.3
BBT [Sun *et al.*, 2022b]	88.9(1.5)	91.4(1.3)	82.5(0.7)	79.8(2.0)	63.9(2.9)	44.7(1.0)	49.7(2.0)	71.5
GAP3	<u>**89.7**</u>(2.8)	**93.0**(2.3)	**83.2**(3.2)	**83.7**(2.9)	70.2(4.5)	**51.1**(4.6)	49.7(1.5)	**74.4**

Table 1: Experimental results for the RoBERTa$_{\text{LARGE}}$ backbone. All the numbers are percentage numbers with '%' omitted. The mean and standard deviation computed based on 3 different splits. **Bold** results are the best ones in the gradient-free group. <u>Underlined</u> results are the overall best ones in both groups.

Method	SST-2 acc	Yelp P. acc	AG's News acc	DBPedia acc	MRPC f_1	SNLI acc	RTE acc	Avg.
				Gradient-based methods				
PT [Li and Liang, 2021]	82.0(0.6)	82.0(7.0)	82.1(1.5)	95.9(0.2)	65.1(8.5)	46.3(3.7)	53.1(5.0)	72.4
Full-model FT	<u>86.4</u>(5.5)	<u>95.0</u>(0.2)	<u>88.1</u>(1.1)	<u>97.9</u>(0.1)	69.8(12.3)	<u>56.6</u>(2.8)	54.3(3.3)	78.3
				Gradient-free methods				
Manual	61.4	60.2	78.7	48.5	38.9	38.7	<u>**57.0**</u>	54.8
ICL [Brown *et al.*, 2020]	48.1(0.9)	63.0(9.8)	51.2(13.9)	37.9(9.7)	52.2(12.6)	37.6(2.6)	53.9(2.4)	49.2
GRIPS [Prasad *et al.*, 2022]	75.8(1.5)	79.0(1.0)	68.1(3.0)	75.7(3.3)	61.7(1.1)	37.1(1.7)	52.1(2.5)	64.2
GPS [Xu *et al.*, 2022]	–	89.7(4.5)	73.7(1.2)	–	–	37.5(1.8)	53.7(4.1)	63.6
BBT [Sun *et al.*, 2022b]	76.8(2.8)	84.4(4.3)	77.3(2.2)	79.8(2.0)	69.3(2.7)	**42.1**(1.0)	51.5(2.7)	68.8
GAP3 + RoBERTa$_{\text{LARGE}}$	79.7(5.3)	**90.2**(3.6)	**82.4**(1.7)	80.1(7.1)	71.3(3.3)	41.3(1.8)	53.4(4.0)	**71.2**
GAP3 + BERT$_{\text{BASE}}$	**82.6**(4.9)	89.9(4.0)	80.7(1.7)	**81.7**(0.7)	<u>**72.3**</u>(5.1)	37.2(2.2)	49.3(1.8)	70.5

Table 2: Experimental results for the GPT-2$_{\text{LARGE}}$ backbone. All the numbers are percentage numbers with '%' omitted. The mean and standard deviation computed based on 3 different splits. **Bold** results are the best ones in the gradient-free group. <u>Underlined</u> results are the overall best ones in both groups.

4.3 Results

Experimental results with RoBERTa$_{\text{LARGE}}$ and GPT-2$_{\text{LARGE}}$ as the backbones are shown in Table 1 and 2, respectively. It can be found that in both scenarios, the proposed GAP3 outperforms the other baselines in the gradient-free group with a notable margin. Interestingly, for the GPT-2$_{\text{LARGE}}$ backbone, the BERT$_{\text{BASE}}$ (110M) auxiliary works almost as good as the RoBERTa$_{\text{LARGE}}$ (354M) auxiliary. Although showing a slightly lower average score, the former achieves even higher scores on SST-2, DBPedia and MRPC than the latter. This also suggests that GAP3's dependence on a particular auxiliary MLM is weak. In addition, GAP3 surpasses gradient-based PT for RoBERTa$_{\text{LARGE}}$, and achieves an average score close to gradient-based PT for GPT-2$_{\text{LARGE}}$.

However, full-model FT still appears to be the most competitive paradigm. Despite the capability to tune black-box backbones, none of the methods in the gradient-free group achieves an overall score comparable to full-model FT. Similar findings were also indicated in a recent study [Chen *et al.*, 2022] particularly designed to analyse this aspect. In addition, on RTE, none of the gradient-free methods performs notably better than chance, which indicates the existence of particular problems that are more difficult for DFO-style algorithms to solve.

4.4 Ablation Study

Ablation experiments are conducted based on the SST-2 and AG's News datasets and the RoBERTa$_{\text{LARGE}}$ backbone, where we vary one hyperparameter, while keeping the others fixed. The default hyperparameter values are the same as those in §4.1, except that we use 32-shot learning by default in this section, to reduce the variance over 3 different runs.

k-shot. We increase the number of training examples per label (i.e. k), with k being 16, 32, 64 and 128, respectively,

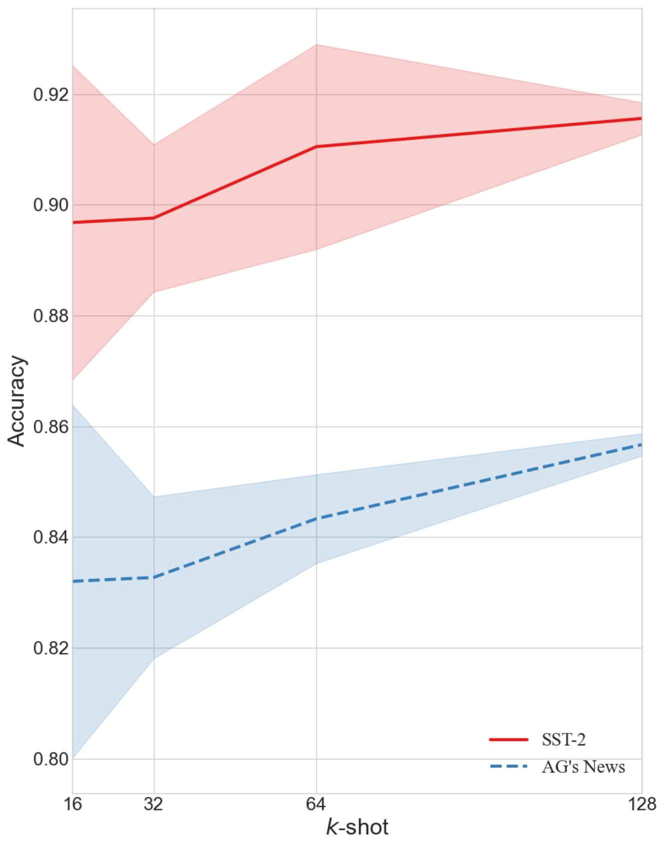

Figure 1: Ablation study on number of training examples (per label).

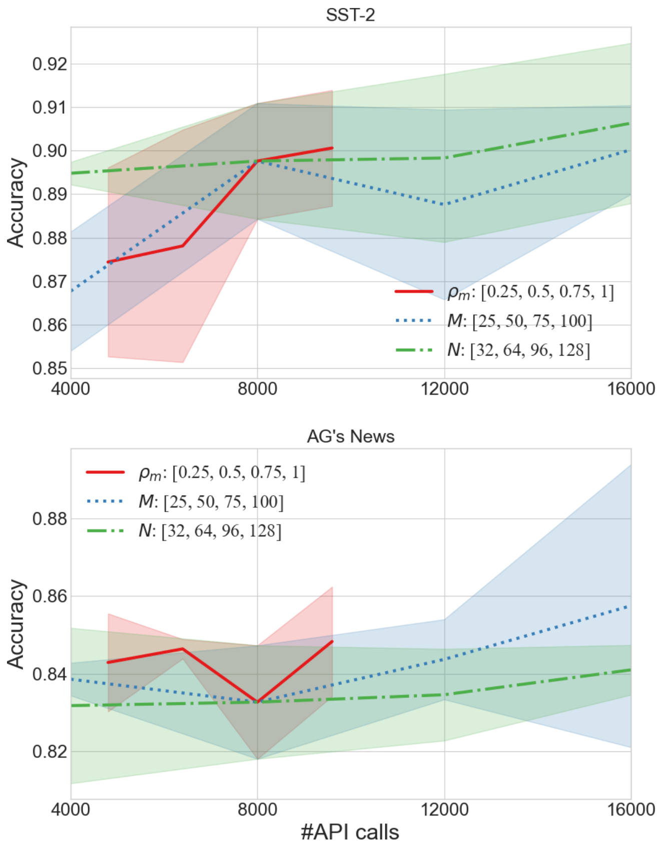

Figure 2: Ablation study on expected number of API calls, with respect to iteration number, population size and mutation probability.

and plot the corresponding performance of GAP3 in Figure 1. It can be found that the obtained accuracy scores grow with k on both dataset sets. Furthermore, up to $k = 128$, there is no clear trend of convergence appearing, which suggests that GAP3 has the potential capability of learning from larger training sets.

API call budget. The expected number of API calls for GAP3 is jointly determined by three hyperparameters, the **population size** (N), the **number of iterations** (M) and the **mutation probability** (ρ_m). We plot the ablation results of the above hyperparameters together in Figure 2, against the number of API calls they yield. It is understandable that more API calls normally correspond to better results. However, due to the randomness in GAP3's strategies, exceptions may occur by chance, where a setting with fewer API calls happen to outperform those with more API calls.

Other hyperparameters. As shown in Figure 3, GAP3 is to some extent sensitive to the **crossover probability**. It suggests that some further heuristics would need to be designed in the future, to seek an optimal value for ρ_c. In addition, we also experiment with an alternative **secondary fitness function** (cf. §3.4), which is obtained by omitting the Kronecker delta in Eq. 7. The results indicate that doing so will reduce the mean accuracy by 1.0% on SST-2 and 0.5% on AG's News.

5 Further Discussions

Label word selection. The label words in this work are manually assigned. Preliminary attempts were made to search for label words automatically, based on the method proposed in [Gao *et al.*, 2021], which, however, resulted in serious overfitting. Better strategies to gain label words for GAP3 will be addressed in our feature research.

Prompt length. In GAP3, prompt length is not predefined, but the iteration number hyperparameter will determine an upper threshold of the maximum possible prompt length. In addition, the template defined in §3.1 does not necessarily mean that every prompt slot `[Ti]` will have tokens in the end. There are possibilities that the final survival individual has some of its chromosomes remain empty. The above features make the prompts generated by GAP3 more flexible and less hyperparameter-dependent.

Interpretability of prompts. Generally speaking, the prompts generated by GAP3 are not understandable by human, since it is not designed to gain human-readable text. Nevertheless, one can still find some 'keywords' within the prompts interpretable. Example prompts learned in §4.3 can be found in the supplementary material[2].

[2]https://github.com/zjjhit/gap3/blob/main/learned-prompts.pdf

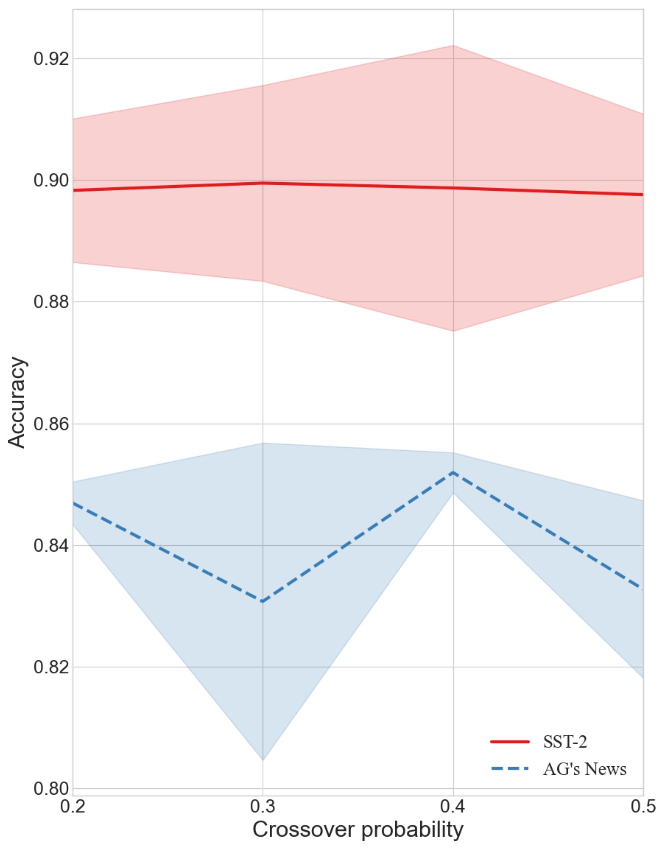

Figure 3: Ablation study on crossover probability.

Limitation. Due to the $M \times N \times (1+2\rho_m)$ API calls yielded during GAP3's evolutions, it will be computationally expensive to apply it directly to a full-sized training set. This is also a common limitation of the existing DFO-based prompting methods. In our GAP3 case, reshaping the evolution and evaluation processes in a k-fold manner will possibly relieve the computational complexity problem, which we will further investigate in future studies.

6 Conclusion

This paper introduces GAP3, an LM probability guided GA, to search prompts automatically for black-box PLM backbones. Despite its outstanding performance on diverse benchmarks, the most significant superiority of GAP3 is the waiver of the preconditions required by existing DFO-based prompting methods, such as injection APIs or manual prompts. The zero or minimal dependency of GAP3 on additional resources suggests it to be an out-of-box complementary to those LM-as-a-Service instances. The computational cost of applying it to full-sized training problems would be the current major limitation of GAP3. Addressing this limitation will be one of our future research directions.

A Prompt Templates and Labels for GAP3

Prompt templates and label words (verbalisers) used in our GAP3 experiments for the RoBERTa$_{\text{LARGE}}$ and GPT-2$_{\text{LARGE}}$ backbones are listed in Table 3 and 4, respectively.

Task	Template	Labels
SST-2	[T$_1$] [X] [T$_2$] [Y]	bad, good
Yelp P.	[T$_1$] [X] [T$_2$] [Y]	bad, good
AG's News	[X] [T] [Y]	world, sports, business, technology
DBPedia	[T$_1$] [Y] [T$_2$] [X]	company, education, artist, athlete, office, transportation, building, nature, village, animal, plant, album, film, literature
MRPC	[T$_1$] [X$_1$] [Y] [T$_2$] [X$_2$]	No, Yes
SNLI	[T$_1$] [X$_1$] [Y] [T$_2$] [X$_2$]	Yes, Maybe, No
RTE	[T$_1$] [X$_1$] [Y] [T$_2$] [X$_2$]	Yes, No

Table 3: Templates and labels for the RoBERTa$_{\text{LARGE}}$ backbone.

Task	Template	Labels
SST-2	[T$_1$] [X] [T$_2$] [Y]	–
Yelp P.	[T$_1$] [X] [T$_2$] [Y]	–
AG's News	[X] [T] [Y]	–
DBPedia	[T$_1$] [X] [T$_2$] [Y]	–
MRPC	[T$_1$] [X$_1$] [T$_2$] [X$_2$] [T$_3$] [Y]	no, yes
SNLI	[T$_1$] [X$_1$] [T$_2$] [X$_2$] [T$_3$] [Y]	always, sometimes, never
RTE	[T$_1$] [X$_1$] [T$_2$] [X$_2$] [T$_3$] [Y]	true, false

Table 4: Templates and labels for the GPT-2$_{\text{LARGE}}$ backbone. '–' stands for labels identical to those in Table 3.

Acknowledgements

The first author thanks the members of the Algorithm and Development Team at the Data Intelligence Centre of China Mobile Online Services Co., Ltd. for fruitful discussions on this work. The second author thanks XREAL for supporting this research by providing the computing power.

References

[Ben-David *et al.*, 2022] Eyal Ben-David, Nadav Oved, and Roi Reichart. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of ACL*, 10:414–433, 2022.

[Bowman *et al.*, 2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, 2015.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[Chen *et al.*, 2022] Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. Revisiting parameter-efficient tuning: Are we really there yet? *CoRR*, abs/2202.07962, 2022.

[Clark *et al.*, 2020] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proceedings of ICLR*, 2020.

[Deng *et al.*, 2022] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of EMNLP*, 2022.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, 2019.

[Diao *et al.*, 2023] Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, Yong Lin, Xiao Zhou, and Tong Zhang. Black-box prompt learning for pre-trained language models. *Transactions on Machine Learning Research*, 2023.

[Gao *et al.*, 2021] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of ACL-IJCNLP*, 2021.

[Hansen *et al.*, 2003] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[Hou *et al.*, 2022] Bairu Hou, Joe O'Connor, Jacob Andreas, Shiyu Chang, and Yang Zhang. PromptBoosting: Black-box text classification with ten forward passes. *CoRR*, abs/2212.09257, 2022.

[Houlsby *et al.*, 2019] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the ICML*, 2019.

[Jiang *et al.*, 2021] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of ACL*, 9, 2021.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

[Kolda *et al.*, 2003] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[Lester *et al.*, 2021] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of EMNLP*, 2021.

[Lewis *et al.*, 2020] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*, 2020.

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL-IJCNLP*, 2021.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[Liu *et al.*, 2021a] Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *CoRR*, abs/2110.07602, 2021.

[Liu *et al.*, 2021b] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[Liu *et al.*, 2022] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of ACL (Volume 2: Short Papers)*, 2022.

[Liu *et al.*, 2023] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):no.195:1–35, 2023.

[Loshchilov and Hutter, 2019] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of ICLR*, 2019.

[Mishra *et al.*, 2022] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of ACL*, 2022.

[Mitchell, 1998] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1998.

[Ouyang *et al.*, 2022] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155, 2022.

[Petroni *et al.*, 2019] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*, 2019.

[Pfeiffer *et al.*, 2020] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. AdapterHub: A framework for adapting transformers. In *Proceedings of EMNLP: System Demonstrations*, 2020.

[Prasad *et al.*, 2022] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. *CoRR*, abs/2203.07281, 2022.

[Qin and Eisner, 2021] Guanghui Qin and Jason Eisner. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of NAACL-HLT*, June 2021.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[Rios and Sahinidis, 2013] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247—1293, 2013.

[Sanh *et al.*, 2022] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *Proceedings of ICLR*, 2022.

[Schick and Schütze, 2021a] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of EACL*, 2021.

[Schick and Schütze, 2021b] Timo Schick and Hinrich Schütze. Few-shot text generation with natural language instructions. In *Proceedings of EMNLP*, 2021.

[Schick and Schütze, 2022] Timo Schick and Hinrich Schütze. True few-shot learning with Prompts—A real-world perspective. *Transactions of ACL*, 10:716–731, 2022.

[Shin *et al.*, 2020] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of EMNLP*, 2020.

[Sun *et al.*, 2022a] Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of EMNLP*, 2022.

[Sun *et al.*, 2022b] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *Proceedings of ICML*, 2022.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.

[Xu *et al.*, 2022] Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. GPS: Genetic prompt search for efficient few-shot learning. *CoRR*, abs/2210.17041, 2022.

[Yu and Gen, 2010] Xinjie Yu and Mitsuo Gen. *Introduction to Evolutionary Algorithms*. Springer, 2010.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[Zhang *et al.*, 2020] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of ICML*, 2020.