UNIVERSITY OF CALIFORNIA SAN DIEGO

Optimizing Discrete Text Prompts for Large Language Models

A thesis submitted in partial satisfaction of the
requirements for the degree /
Master of Science

in

Computer Science

by

Jianyu Wang

2023

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Zhiting Hu for his support in my research in 2022, where I got the courage and inspiration to explore many exciting and fundamental problems in the field of machine learning, and natural language processing. It is his support that helped me in an immeasureable way.

I would also like to acknowledge my great friend, particularly, Junyan Liu for being a constant presence through good and bad times. I also owe enormous thanks to other friends and researchers who have offered me incredible support, especially Yilan Chen, Zixuan Zhang, Meng Song, Isamu Poy, Tianmin Shu, Kaiming Kuang, Xiu Yuan and many others. Also, I am very grateful to my mother, for her unconditional love, trust and support.

I would thank so many other exceptional professors that I met at UC San Diego, such as Prof. Biwei Huang, Prof. Nikolay Atanasov, Prof. Hao Su, for their earnest teaching, and well-organized course structures. I do learn a lot from their courses.

Chapter 2, in part, is a reprint of the material as it appears in EMNLP 2022 conference paper "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning". The thesis author was the primary investigator and author of this paper.

Chapter 3, in full, has been submitted for publication of the material as it may appear in EMNLP 2022 conference paper "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning". The thesis author was the primary investigator and author of this paper.

Chapter 4, in full, has been submitted for publication of the material as it may appear in EMNLP 2022 conference paper "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning". The thesis author was the primary investigator and author of this paper.

VITA

2020	B.E. in Electronic and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China
2021–2023	M.S. in Computer Science, University of California San Diego

PUBLICATIONS

Mingkai Deng*, **Jianyu Wang***, Chengping Hsieh*, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, Zhiting Hu, "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning", EMNLP 2022

Jianyu Wang, Bing-Kun Bao, Changsheng Xu, "DualVGR: A Dual-Visual Graph Reasoning Unit for Video Question Answering", IEEE Transactions on Multimedia (TMM) 2021

ABSTRACT OF THE DISSERTATION

Optimizing Discrete Text Prompts for Large Language Models

by

Jianyu Wang

/
Master of Science in Computer Science

University of California San Diego, 2023

Professor Zhiting Hu, Chair
Professor Jingbo Shang, Co-Chair

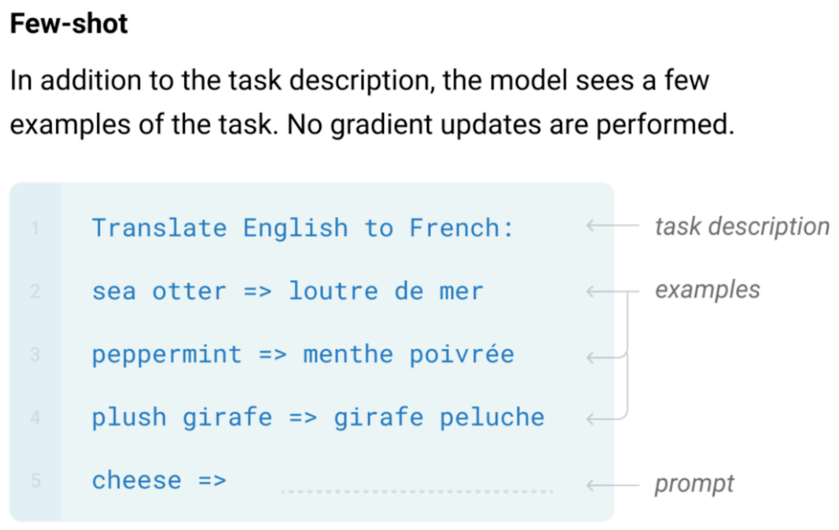
Prompting has shown impressive success in enabling large pretrained language models (LMs) to perform diverse NLP tasks, especially when only few downstream data are available. Automatically finding the optimal prompt for each task, however, is challenging. Most existing work resorts to tuning soft prompt (e.g., embeddings) which falls short of interpretability, reusability across LMs, and applicability when gradients are not accessible. Discrete prompt, on the other hand, is difficult to optimize, and is often created by ”enumeration (e.g., paraphrasing)-then-selection” heuristics that do not explore the prompt space systematically. This thesis will mainly introduce our proposed RLPrompt, an efficient discrete prompt optimization approach with reinforcement learning (RL). RLPrompt formulates a parameter-efficient policy network that generates the desired discrete prompt after training with reward. To overcome the complexity and stochasticity of reward signals by the large LM environment, we incorporate effective reward stabilization that substantially enhances the training efficiency. RLPrompt is flexibly applicable to different types of LMs, such as masked (e.g., BERT) and left-to-right models (e.g., GPTs), for both classification and generation tasks. Experiments on few-shot classification and unsupervised text style transfer show superior performance over a wide range of existing finetuning or prompting methods. Interestingly, the resulting optimized prompts are often ungrammatical gibberish text; and surprisingly, those gibberish prompts are transferrable between different LMs to retain significant performance, indicating LM prompting may not follow human language patterns.

Chapter 1

Introduction

Starting from BERT [8], adapting language model for downstream tasks with fine-tuning has become the de factor practice in NLP research. But recently, the GPT-3 model with 175B parameters [9] has brought a new breakthrough to the whole NLP field, that is, the prompting techniques. Specifically, as the title ”Language Models are Few-Shot Learners” suggests, GPT-3 can be prompted (this case is in-context demonstration prompt) to handle a wide range of tasks while freezing its parameters, simply by leveraging tailored natural language prompts [10] for target tasks. For instance, (see Figure 1.1) for translation tasks, we can simply prepend a few task demonstrations with random task examples, formatted as input-output pairs, into our target input. And our language model, an auto-regressive language model pre-trained to predict the next word, will learn to follow the formatted prompts to predict the suitable output. In this case, language models will learn from these demonstrations to translate the source input sentence into our target French translated sentence. In addition, the prompting techniques can also be extended to many other tasks beyond conditional text generation, such as question answering, text classification, commonsense reasoning, etc, as long as you construct with the decent prompts and correct task formulations.

However, things do not go very smoothly. Language Models are not clever enough as human, in other words, for the same task instructions with different forms [11], they can perform quite differently that a simple paraphrasing with the initial decent natural language prompts



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

Figure 1.1. Prompting language models for downstream tasks.

can degrade the task performance to around chance level. In particular, prompts are so brittle that a simple and negligible perturbation [2], without hurting its semantics and grammatically correctness, can even lead to huge difference to its performance. This counter-intuitive behavior of prompts may imply that its constructions do not follow any natural language patterns.

In order to fully realize its potential, the topic of "automatic prompt tuning/optimization" has become a topic among NLP researchers, that is, searching for the ideal prompts for various downstream tasks. However, the discrete nature of text prompts renders the optimization very difficult. Previous work typically resort to various heuristics to tune the prompts, encompassing enumeration (e.g., paraphrasing)-then-selection [2, 12, 1, 13] for natural language prompts, retrieve-then-read [14] for in-context demonstrations, usually leading to sub-optimal performance for downstream tasks. AutoPrompt [15], the only discrete prompt optimization framework adapted from Hotflip [16], utilizes expensive gradient information to edit the prompt tokens. Without enough labeled data for the gradient information, often, AutoPrompt is largely limited in its performance. These heuristics restrict the real power of discrete text prompts. One alternative way is to tune soft prompts that instead of locate ideal discrete text prompts, researchers directly optimize continuous embedding vectors from task gradients. Although this paradigm [17, 18, 19, 13, 20, 21, 22] in general achieves better performance than most of current discrete prompt engineering techniques for downstream tasks given rich gradient information. However, the resulting prompts are, by their nature, hard for humans to understand [23, 24, 25] and incompatible for use with other LMs. Besides, the required LM internal gradients are often expensive to compute, or simply unavailable for LMs deployed with only inference APIs (e.g., GPT-3). It is thus often desirable to use discrete prompts which consist of concrete tokens from a vocabulary.

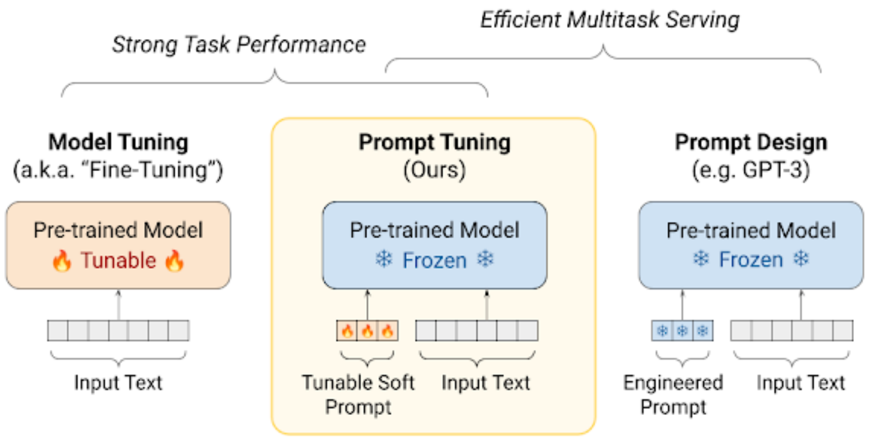


Figure 1.2. Guiding Frozen Language Models with Learned Soft Prompts.

In this thesis, I will introduce one new methodology named RLPrompt proposed by me, which formulates prompt optimization as prompt generation framework guided by reward signals, optimizing by reinforcement learning, which will be briefly reviewed in Chapter 2. To my knowledge, this is the first work which systematically optimizes over exponentially large token spaces. Specifically, extensive experiments conducted on both few-shot text classification task

(Chapter 3) and zero-shot unsupervised text style transfer task (Chapter 4) have demonstrated its effectiveness over a wide range of existing work. To summary, the main content of this thesis is based on the EMNLP 2022 conference paper, RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning. The thesis author is the primary investigator and co-first author of this paper.

Chapter 2

Discrete Prompt Optimization with RL

In this chapter, I will first give a brief introduction to current prompting techniques and the corresponding template formats for a wide range of NLP tasks, including both classification and generation tasks. Then, I will introduce our motivation of optimzing discrete prompts with reinforcement learning, and the formal RL formulation for the discrete prompt optimization.

2.1 Background: Prompting language models for numerous NLP tasks

As introduced in Chapter 1, prompting language models [10] has become current de facto for various NLP tasks, where the large language models perform the autocompletion following the well-designed natural language prompts. Specifically, in addition to tuning soft prompts, i.e., soft embedding vectors, there are basically three types of natural language prompts for both classification and generation tasks (not included some advanced prompting techniques directly invented for more intricate tasks, e.g., Chain-of-thought prompting [26, 27] for symbolic reasoning tasks, adding task explanations [28, 29] into in-context demonstrations for very large language models as emergent behaviors, etc), encompassing the most straightforward hand-crafted natural language prompts, prepending task instructions, and few-shot examples as in-context learning. These prompting techniques basically predict the highest-scoring output in three steps (shown in Figure 2.1).

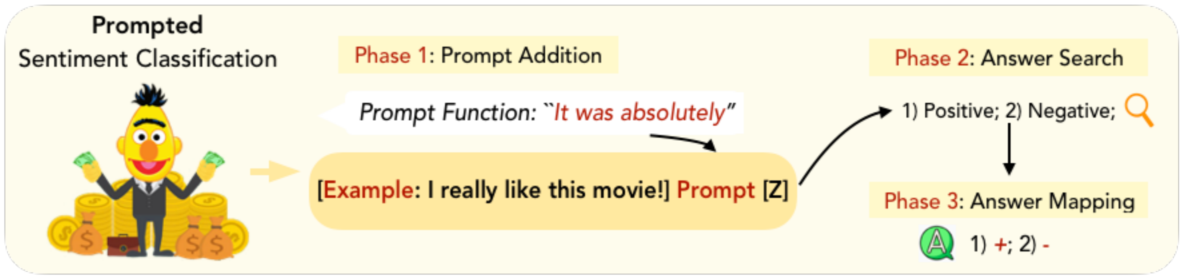

Prompted Sentiment Classification
Phase 1: Prompt Addition
Prompt Function: "It was absolutely"
[Example: I really like this movie!] Prompt [Z]
Phase 2: Answer Search
1) Positive; 2) Negative;
Phase 3: Answer Mapping
1) +; 2) -

Figure 2.1. General procedure of prompting;

2.1.1 Prompt Addition

In this step, a prompting function $f_{prompt}(\cdot)$ is applied to modify the input text $\mathbf{x}$ into a prompt $\mathbf{x}' = f_{prompt}(\mathbf{x})$. Typically, this function consists of two steps:

1. Apply a template (prompt), which is a expression comprising two slots: an input slot $[\mathbf{X}]$ for input $\mathbf{x}$ and an answer slot $[\mathbf{Z}]$ for an intermediate generated answer text $\mathbf{z}$ that will later be mapped to the real predicted answer $\mathbf{y}$.

2. Fill slot $[\mathbf{X}]$ with the input target text $\mathbf{x}$.

In the case of text classification, for instance, the sentiment polarity classification task. Given the input $\mathbf{x}$ =“I love this movie’s content! It is really interesting!”, the template, i.e., prompt function may take the form as “$[\mathbf{X}]$It was a$[\mathbf{Z}]$movie!”. Then, $\mathbf{x}'$ would become “I love this movie’s content! It is really interesting! It was a$[\mathbf{Z}]$movie!” given the input example. In the case of generation, e.g., machine translation, the prompt template may take the form such as “French: $[\mathbf{x}]$ English: $[\mathbf{Z}]$”, similar as classification example, language model will be prompted to predict the output $[\mathbf{Z}]$ following the prompt.

Notably, the prompts have an empty slot to fill in for $\mathbf{z}$, either in the middle for masked language models or at the end for both masked language models and causal language models. And depending on task formats [30], you can flexibly change the number of $[\mathbf{X}]$and$[\mathbf{Z}]$ slots.

2.1.2 Answer Search

For both classification and generation tasks, we can further intervene the answer **z**, that is, we can define **z** as a set of permissible values, to restrict the search space. In the case of sentiment classification task, we can select the verbalizers, i.e., the vocabulary list representing class labels, “*great*″,“*good*″,“*OK*″,“*bad*″,“*terrible*″ to represent the label set in the five-way sentiment classification task, e.g., SST-5 [31].

For this construction, there are also several approaches to automatically search for the best verbalizers [1, 15]. Similar to template search, these methods mainly rely on various heuristics to locate decent verbalizers. Usually, this search function can be an argmax search that searches for the highest-scoring output, or simply sampling that randomly generates outputs following the probability distribution of the language models.

2.1.3 Answer Mapping

Finally, that is to convert the highest-scoring answer **z** to the highest-scoring output (predicted answer) **y**. This is trivial in most cases, where the answer might be itself, as most generation tasks. And for classification task, this often is mapping the answer form **z** from the verbalizers to target prediction y, according to the bijective mapping in construction. One non-trivial case is when you use multiple answer forms to represent one target output, then you would map them together to one target output.

2.2 Motivation & Introduction

Prompting has emerged as a promising approach to solving a wide range of NLP problems using large pre-trained language models (LMs), including left-to-right models such as GPTs [32, 9] and masked LMs such as BERT [8], RoBERTa [33], etc. Compared to conventional fine-tuning that expensively updates the massive LM parameters for each downstream task, prompting concatenates the inputs with an additional piece of text that steers the LM to produce the desired outputs. A key question with prompting is how to find the optimal prompts to improve the LM's performance on various tasks, often with only a few training examples.

One of the most popular scheme is to tune *soft* prompts (i.e., continuous embedding vectors) as they are amenable to gradient descent [17, 18, 19, 34, 20, 21, 22]. However, the resulting prompts are, by their nature, hard for humans to understand [23, 24, 25] and incompatible for use with other LMs. Besides, the required LM internal gradients are often expensive to compute, or simply unavailable for LMs deployed with only inference APIs (e.g., GPT-3). It is thus often desirable to use *discrete* prompts which consist of concrete tokens from a vocabulary. However, their discrete nature renders the optimization very difficult. Previous work has typically relied on manual engineering [35, 9, 36, 37], or selecting from multiple paraphrased/generated prompts [2, 1, 14, 12, 38]. AutoPrompt [15] uses gradient information to edit the prompt tokens, which suffers from training instability as well as the same applicability issue as gradient-based soft prompting, showing limited effectiveness in practice.

This thesis would mainly introduce our proposed RLPROMPT, a new discrete prompt optimization approach based on reinforcement learning (RL). This approach brings together a wide range of desirable properties for efficient use on diverse tasks and LMs (Table 2.1). Crucially, rather than directly editing the discrete tokens, which has been difficult and inefficient, RLPROMPT trains a policy network that generates the desired prompts. Discrete prompt optimization thus amounts to learning a small number of policy parameters which we set as an MLP layer inserted into a frozen compact model such as distilGPT-2 [39]. This formulation also allows us to employ off-the-shelf RL algorithms [40] that learn the policy with arbitrary reward functions—defined either with available data (e.g., in few-shot classification) or other weak signals when no supervised data is accessible (e.g., in controllable text generation).

On the other hand, RL for prompt optimization poses new challenges to learning efficiency: the large black-box LM presents a highly complex environment that, given the prompt (i.e., actions), goes through a long series of complex transitions (e.g., reading the input and inferring the output) before computing the rewards. This makes the reward signals extremely unstable

Table 2.1. Comparison of different (prompting) paradigms for using pre-trained LMs on downstream tasks, in terms of several desirable properties.

Methods	**Frozen LMs**	**Automated**	**Gradient-Free**	**Guided Optimize**	**Few-Shot**	**Zero-Shot**	**Transferrable b/w LMs**	**Interpret.**
Fine-Tuning	✗	✓	✗	✓	✗	✗	✗	✗
Manual Prompt	✓	✗	✓	✗	✓	✓	✓	✓
Instructions	✓	✗	✓	✗	✓	✓	✓	✓
In-Context Demo.	✓	✓	✓	✗	✓	✗	✓	✓
Soft Prompt Tuning	✓	✓	✗	✓	✓	✗	✗	✗
Discrete Prompt Enum.	✓	✓	✓	✗	✓	✓	✓	✓
AutoPrompt	✓	✓	✗	✓	✓	✗	✓	✓
RLPrompt (**Ours**)	✓	✓	✓	✓	✓	✓	✓	✓

and hard to learn from. To overcome this difficulty, we propose two simple yet surprisingly effective ways to stabilize the rewards and improve the optimization efficiency.

Experiments on few-shot classification and unsupervised text style transfer show our approach improves over a wide range of fine-tuning and prompting methods (e.g., those described in Table 2.1), and is robust to different modeling choices (e.g., verbalizers in classification). The resulting discrete prompts also facilitate rich interpretations and analyses for new insights into LM prompting. In particular, the optimized prompts, though inducing strong task performance, tend to be gibberish text without clear human-understandable meaning, echoing recent research [41, 42, 12] that LMs making use of prompts do not necessarily follow human language patterns. Perhaps surprisingly, those gibberish prompts learned with one LM can be used in other LMs for significant performance, indicating that those different pre-trained LMs have grasped shared structures for prompting.

2.3 Related Work

2.3.1 Prompting Paradigms

Fine-Tuning

The conventional approach to adapt pre-trained LMs is fine-tuning model parameters on downstream datasets [8, 33, 43, 44, 32]. While driving progress in a wide range of NLP tasks, fine-tuning expensively updates all model parameters and shows limited success with small datasets. Inspired by PET [36], Prompt-based fine-tuning [1, 36] (see Figure 2.2) updates model parameters with a fixed prompted inserted to improve few-shot tuning performance, which is regarded as a path towards better few-shot learners for small language models. However, same as fine-tuning, it also suffers from several limitations, such as the problem of costly training, catastrophic forgetting [45, 46], etc. And these techniques are not that practical to very large language models due to their costly re-training.

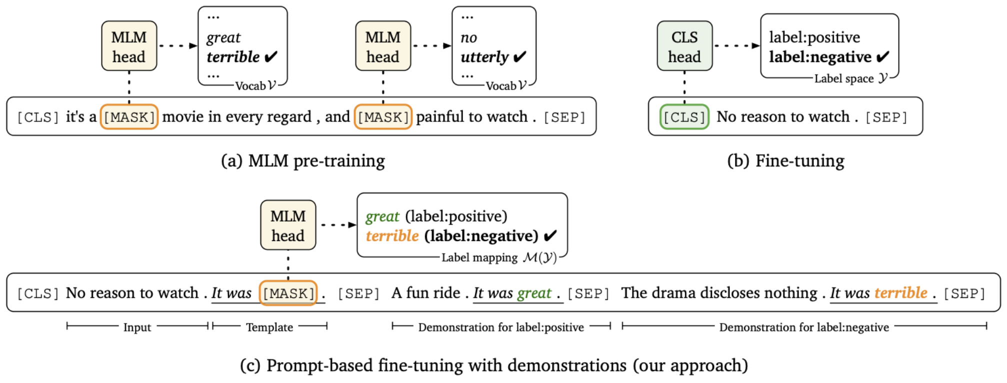

MLM head
...
great
terrible ✔
...
Vocab $\mathcal{V}$
MLM head
...
no
utterly ✔
...
Vocab $\mathcal{V}$
[CLS] it's a [MASK] movie in every regard , and [MASK] painful to watch . [SEP]
(a) MLM pre-training
CLS head
label:positive
label:negative ✔
Label space $\mathcal{Y}$
[CLS] No reason to watch . [SEP]
(b) Fine-tuning
MLM head
great (label:positive)
terrible (label:negative) ✔
Label mapping $\mathcal{M}(\mathcal{Y})$
[CLS] No reason to watch . It was [MASK] . [SEP] A fun ride . It was great . [SEP] The drama discloses nothing . It was terrible . [SEP]
Input
Template
Demonstration for label:positive
Demonstration for label:negative
(c) Prompt-based fine-tuning with demonstrations (our approach)

Figure 2.2. General procedure of prompt-based fine-tuning [1].

Manual Prompt

As LMs show remarkable progress in understanding natural language [8], researchers first use hand-crafted fill-in-the-blank prompts to extract knowledge from pre-trained LMs for factual probing analyses [35, 2] (see Figure 2.4). Later on, [9] show that using manually-written prompts, large LMs can perform a number of NLU and NLG tasks without any training examples.

Meanwhile, other studies [44, 36, 47] formulate various NLP tasks as manual prompts. These research raises researcher's interests to identify better prompts for various downstream tasks.

Although seemingly easy for practitioners to come up with prompts for downstream tasks,

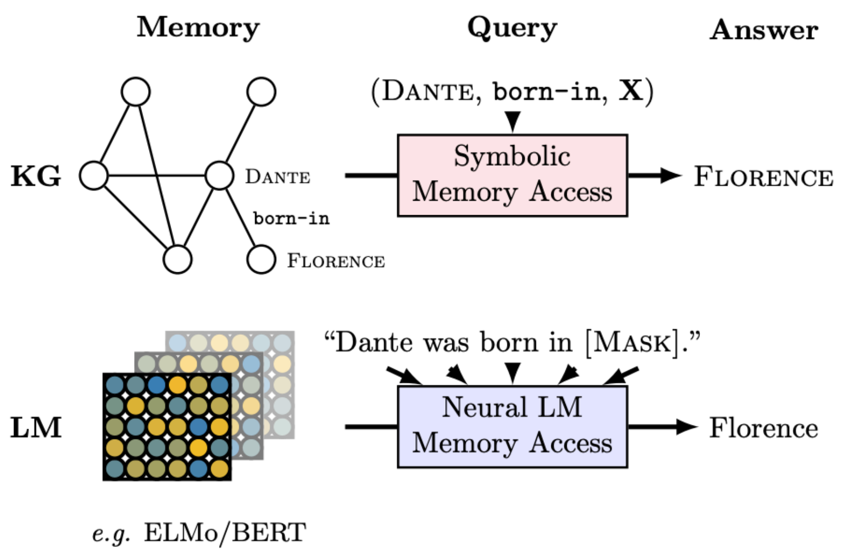


Figure 2.3. General Procedure of probing factual knowledge in LMs [2].

the real patterns behind these prompts are still under-explored. A few works [2, 41] showcase the brittleness of the manually crafted prompts that a small even negligible modification to the initial prompt, e.g., changing the preposition "in" to "at" causes huge difference in factual probing tasks, even though the two natural language prompts both seem reasonable and intuitive to human common-sense. Therefore, manual tweaking textual prompts seem to be a challenging engineering part, which requires practitioners to explore by trial and error.

Instructions

Separate from but related to manual prompts, another line of work [48, 49, 50, 51] makes use of instructional prompts which provide task descriptions instead of fill-in-the-blank prompts. In particular, instruction meta-tuning [50, 52, 3] trains models on some tasks with instructions and supervised data in order to generalize to unseen tasks formulated as instructions without training examples. This paradigm provides several benefits, as better alignment to user intuition for building the language model assistants.

In-Context Demonstration

Besides the effectiveness of zero-shot learning by hand-crafting natural language prompts aligning with human understandings, there is also one intriguing behavior of large language

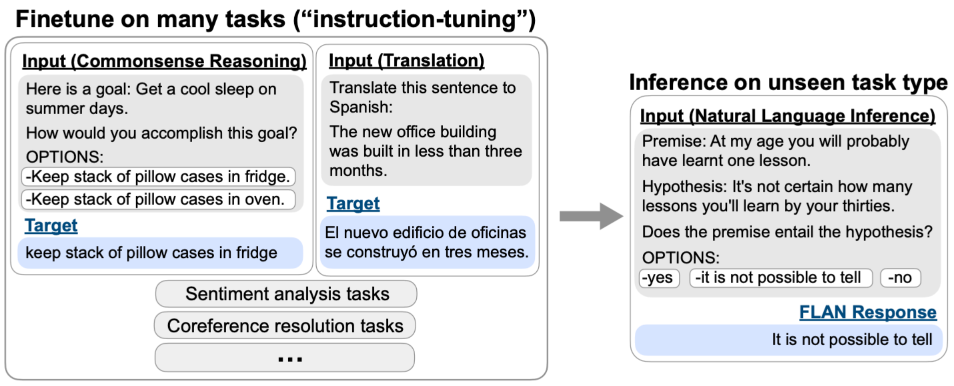

Finetune on many tasks ("instruction-tuning")
Input (Commonsense Reasoning)
Here is a goal: Get a cool sleep on summer days.
How would you accomplish this goal?
OPTIONS:
-Keep stack of pillow cases in fridge.
-Keep stack of pillow cases in oven.
Target
keep stack of pillow cases in fridge
Input (Translation)
Translate this sentence to Spanish:
The new office building was built in less than three months.
Target
El nuevo edificio de oficinas se construyó en tres meses.
Sentiment analysis tasks
Coreference resolution tasks
...
Inference on unseen task type
Input (Natural Language Inference)
Premise: At my age you will probably have learnt one lesson.
Hypothesis: It's not certain how many lessons you'll learn by your thirties.
Does the premise entail the hypothesis?
OPTIONS:
-yes
-it is not possible to tell
-no
FLAN Response
It is not possible to tell

Figure 2.4. General procedure of instruction tuning [3] language models.

models, known as the "in-context learning" [9], where language models can learn to follow the input-output patterns given a few input-output examples concatenated as demonstrations (see Figure 2.5), thus leading to very strong performance for both classification and generation tasks.

Many recent works [1, 14, 53, 54] further explore the selection and analysis of in-context demonstrations. Crucially, the inner working mechanism of ICL is still one puzzle [54, 55, 56] to researchers, questioning its sufficiency to claim language models as the few-shot learner [57]. So current SOTA techniques for in-context exmaple selection is still based on some empirical heuristics, e.g., the well-known retrieve-then-read technique [14] that given a test instance, similar training examples might be more appreciated to form the in-context demonstration to prompt language models to predict the correct answers. And recent work proposes a sensitivity-based selection mechanism [58] to seek for more performant in-context pairs.

Recently, researchers start to explore the utility of adding task explanations [59, 60] in the in-context demonstrations, and surprisingly, they find that it is an emergent behavior of very large language models that they can benefit to notable level of improvements for various downstream tasks. However, it is also found [60] that these improvements cannot cover more complex textual reasoning tasks, namely natural language inference (NLI) and question answering.

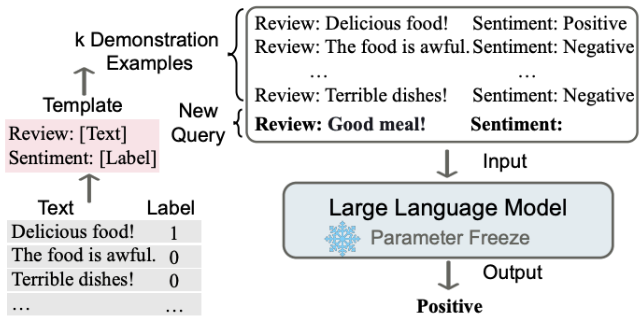


Figure 2.5. General Procedure of In-context Learning [4].

Discrete Prompt Enumeration

Because discrete prompts are difficult to optimize and susceptible to small design variations [42, 41, 53], a number of existing works seek to automatically locate better prompts by augmenting human-written prompts with heuristics such as paraphrasing [2, 1], editing [12], and reframing [61]. The final prompt is typically selected to maximize some downstream performance metric, or some alternative metrics [62, 63] handling with low-source data scenarios.

AutoPrompt

AutoPrompt [15] optimize discrete prompts by editing prompt tokens with guidance from model gradients. While seeing some success with large training data, the method relies heavily on approximation based on rich gradient information from enough labeled data, which leads to less stable training and limited applicability to few-shot settings.

Soft Prompt Tuning

Replacing discrete prompts with continuous embeddings, several parallel works [17, 34] propose to optimize soft prompts with gradient-based tuning. Soft prompt tuning can be seen as a variant of parameter-efficient transfer learning [64, 65, 66], and inspires a number of follow-up works that boost its performance [67, 19, 18, 68] or explore novel applications [69, 70]. By its nature, however, soft prompts are difficult for humans to understand because of its continuous form [23, 24, 25, 20]. Defined in the latent space of specific models, learned prompts are also virtually impossible to use with a different model. Furthermore, their training typically requires gradient information from the models they prompt, which can be expensive to compute or simply inaccessible for models deployed as inference API, such as GPT-3 [9]. [6] and [71] propose black-box tuning, which updates continuous prompts using gradient-free techniques under low-resource regimes to some success.

2.4 The Reinforcement Learning Formulation

We present RLPROMPT, a framework for learning prompts of discrete tokens for pre-trained LMs to succeed in a wide range of NLP tasks. The general pipeline is illustrated in Figure 2.6.

As discussed before, discrete prompts can be easier to interpret and use than continuous prompts, but also more challenging to learn due to intractable optimization over discrete tokens. To solve this difficulty, we formulate discrete prompt optimization as an RL problem, using a continuous policy network to explore the prompt space. The network is highly parameter-efficient, only training a small MLP over a frozen compact LM (e.g., distilGPT-2).

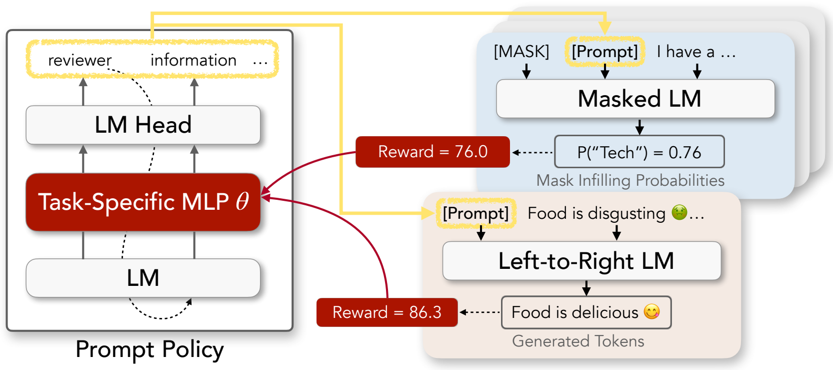


Figure 2.6. Overview of RLPROMPT for discrete prompt optimization.

Below, we present our RL formulation of discrete prompt optimization (§2.4.1-2.4.2). We then discuss the design of our policy network (§2.4.3). Finally, we describe our reward engineering techniques for both classification and generation tasks to improve RL training in the next chapters.

2.4.1 Discrete Prompt Optimization Problem

Extensive recent work [9, 2, 23, 1] has shown it is possible to combine discrete text prompt $\mathbf{z}$ with input $\mathbf{x}$ to directly perform various NLP tasks using a pre-trained LM's generative distribution $P_{\text{LM}}(\mathbf{y}|\mathbf{z},\mathbf{x})$, without needing to fine-tune the model. For instance, in classification, the LM can be a masked language model (MLM) such as BERT [8], and $\mathbf{y}$ is the class-label token (a.k.a. verbalizer like `positive` or `negative`) in the mask position; in a generation task, the LM can be a left-to-right model such as GPT-2 [32], and $\mathbf{y}$ is the generated text. See Figure 2.6 for illustrative examples. We use $\mathbf{y}_{\text{LM}}(\mathbf{z},\mathbf{x})$ to denote the LM output on $\mathbf{x}$ prompted by $\mathbf{z}$.

Our goal is to find the optimal discrete prompt $\mathbf{z}^*$ from vocabulary $\mathscr{V}$ to maximize some downstream performance measure R of $\mathbf{y}_{\text{LM}}(\mathbf{z}^*,\mathbf{x})$.[1] The metric $R(\mathbf{y})$ can be as simple as match with gold label $\mathbf{y}^*$ (e.g., in classification when data is available), but can also be more complex like the success criteria of controllable text generation, which composes aspects such as style accuracy, language quality, and content preservation. Assuming the prompts have fixed length T, we write the task of *discrete prompt optimization* in the general format below:

$$\max_{\mathbf{z}\in\mathscr{V}^T} R\left(\mathbf{y}_{\text{LM}}(\mathbf{z},\mathbf{x})\right). \tag{2.1}$$

The optimization above, however, can be intractable because $\mathbf{z}$'s discrete tokens are not amenable to gradient-based optimization, while brute-force search has the exponential complexity of $\mathscr{O}(|\mathscr{V}|^T)$. Previous work has to either approximate gradients over $\mathbf{z}$ using continuous LM embeddings [15] or tweak human-written prompts with heuristics [2, 61, 12].

2.4.2 The Reinforcement Learning Formulation

To overcome the difficulty, we formulate discrete text prompt optimization as an RL problem, in which an agent selects prompt tokens $[z_1,\ldots,z_T]$ one by one to maximize the reward $R(\mathbf{y}_{\text{LM}}(\mathbf{z},\mathbf{x}))$. At time step t, the agent receives previous prompt tokens $\mathbf{z}_{<t}$ and generates the

[1]Technically $\mathscr{V}$ can be any set of tokens. Here we simply use the downstream LM's vocabulary.

next prompt token z_t according to a policy $\pi(z_t|\mathbf{z}_{<t})$. After the agent finishes the entire prompt $\hat{\mathbf{z}}$, it receives the task reward $R(\mathbf{y}_{\text{LM}}(\hat{\mathbf{z}},\mathbf{x}))$. Parameterizing the policy with , we can rewrite the problem above as

$$\max R(\mathbf{y}_{\text{LM}}(\hat{\mathbf{z}},\mathbf{x})),\ \hat{\mathbf{z}} \sim \textstyle\prod_{t=1}^{T} \pi(z_t|\mathbf{z}_{<t}). \tag{2.2}$$

Compared to typical (soft) prompt tuning approaches, the RL formulation above has the key advantage of not needing gradient access to the LM, treating it instead as a black-box function. This enables us to optimize prompts for LMs whose gradients are too expensive to compute, or LMs that are solely available as inference APIs (e.g., GPT-3). Compared to previous discrete prompt enumeration/paraphrasing, the RL approach explores the prompt space more efficiently guided by the reward signals. The policy network also brings added flexibility, e.g., it can take other information such as the input $\mathbf{x}$, leading to input-specific prompts (e.g., as used in text style transfer part).

During training, we explore the prompt space by sampling from the policy network. After the policy is trained, we select tokens greedily during inference to produce a deterministic prompt. The reward objective in Eq.(2.2) can be optimized with any off-the-shelf RL algorithm. We use the latest soft Q-learning [40] which has shown advanced learning efficiency and performance on various text generation problems, with open-source implementation.[2] Specifically, we use only its on-policy component. We refer interested readers to [40] for more details.

2.4.3 Efficient Parameterization of Policy

We present an efficient parameterization of the policy network π, which adapts a frozen pre-trained LM (i.e., policy LM) with a simple MLP layer that contains all the parameters to be trained. The policy LM need not be the same as the LM we optimize the prompt for (i.e., task LM). Figure 2.6 (left) illustrates the policy LM architecture. Specifically, we use the LM to extract contextual embeddings of partial prompt $\hat{\mathbf{z}}_{<t}$, apply the added task-specific MLP layer

[2]Our preliminary experiments indicate SQL often achieves superior performance than common policy gradient methods.

to compute the adapted embeddings, and pass the output into the model's original LM head to obtain the next prompt token probabilities. During training, we compute the MLP gradients by back-propagating through the policy LM. Our experiments (see following chapters for various tasks) show that changing only the small set of MLP parameters is sufficient for producing strong performance. After training, we discard the MLP and simply use the learned discrete text prompt for inference.

2.4.4 Reward Engineering and Stabilization

Proper design of reward functions, a.k.a. reward engineering, is crucial to training efficiency and success in RL [72]. Discrete prompt optimization, in particular, poses new challenges due to its highly complex reward functions, which involve multiple steps (e.g., combining with input, passing through a black-box LM, and inferring the outputs), each introducing its own variations. This makes the reward signal unstable and difficult to assess progress towards the task goal.

To solve these difficulties, we propose a few simple reward engineering techniques that effectively encourage and stabilize the RL training, illustrated in the following chapters §3 and §4.

Chapter 3

Prompted Zero-Shot Text Generation Task

The gigantic amount of free text on the Web, several magnitude more than labeled benchmark, enables large language models (LMs) training in scale. However, when generating text samples from LMs by iteratively sampling the next token, we do not have explicit control over attributes of the output text, namely the topic, the style, the sentiment, etc. And for some other conditional text generation tasks, for instance, automatic summarization task [73, 74, 75], table-to-text generation [76, 77, 17], image captioning task [20, 78], etc require the LMs to be adaptive according to various application requirements. Therefore, how to steer LMs for various types of output, or even some creative writing applications, would be a very interesting topic to the whole community.

In this chapter, at the beginning, I will give an introduction of current work on the topic of how to steer LMs for various generation tasks, with distinct requirements. Specifically, I will put my main focus on the text style transfer, a higher-level controllable text generation tasks, often with composable desideratums. Then, I will introduce our work on prompted zero-shot text style transfer, with details on the crafted reward formulation for the high variance unavoidable in current formulation, and corresponding experimental procedure. Note that our reward function and tuned reward engineering hyper-parameters should be suitable for any text generation tasks, as long as you create the correct reward function. Therefore, to adapt our framework to other generation scenarios, you can simply tune the reward function, and the results should be clear with our comprehensive experiments.

3.1 Overview to Prompted Controllable Text Generation

There are mainly three approaches to adapt language models for downstream controllable text generation tasks, encompassing guided decoding strategies, fine-tuning language models with enough parallel corpus, and prompt designs for both parallel and non-parallel generation tasks.

For the first paradigm, the guiding decoding strategies [79, 80, 81], that is, instead of re-training the language models, we can simply re-use the fixed language models for any generation tasks, with only decoding strategies varying. Specifically, we apply different guided decoding approaches to different generation tasks, and select the desired outputs at test time.

Common Decoding Strategies

The final layer produces the logits **o** over the vocabulary space **V**, so we can sample the next token from the probability built upon the logits by applying the softmax with temperature T. The probability of sampling the i-th token is:

$$\mathbf{p}_i \propto \frac{exp(\mathbf{o}_i/\mathbf{T})}{\sum_j exp(\mathbf{o}_j/\mathbf{T})}$$

. A low temperature would make the distributions sharper, and a high temperature makes it softer.

Greedy Search

It always picks the next token with the highest probability, equivalent to seeting temperature $\mathbf{T} = 0$. However, it tends to create repetitions of phrases, etc, even for well-trained models.

Beam Search

This is a relatively looser form of greedy decoding, where you implement optimal search with a limited bandwidth rather than 1-size window in greedy decoding. Essentially, it uses breadth-first search to maintain the token sequences with high joint probability.

For the two techniques shown above, in fact, for pre-trained language models, there are few practitioners really adopting them. The main reason comes from the counter-intuitive text de-generation issue [5] in accompany with these models, that is, output text would be bland, incoherent, or get stuck in repetitive loops even sampling from the highest score-based tokens. [5] also performs an empirical investigation of the distribution differences between human generated text and model generated text, and the results shown in Figure 3.1 indicate that instead, human-generated texts are usually from lower probability sequences.

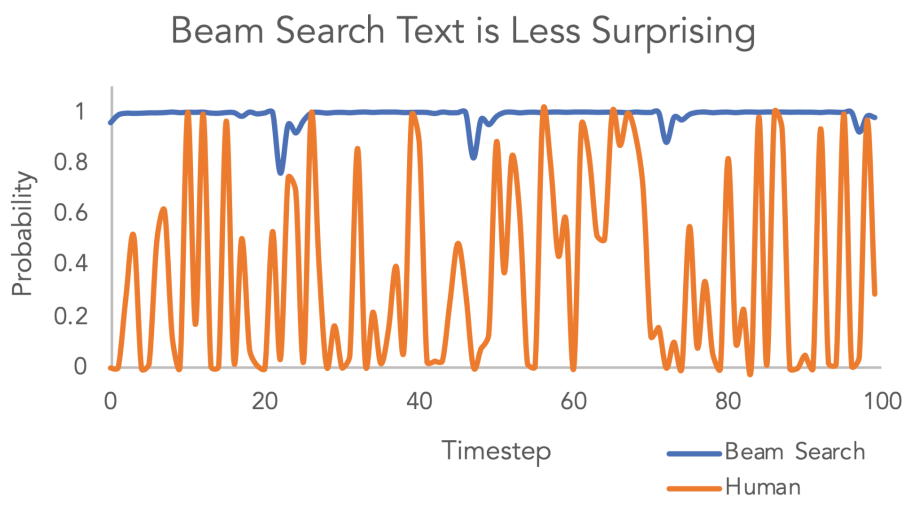


Figure 3.1. Overview of neural text degeneration issues [5].

Therefore, for large language models, in practice, practitioners usually resort to top-k/p sampling strategies [82, 5] to decode the output sentences. The interpretations are straightforward as their names, e.g, top-k represents sample top-k tokens per step in an auto-regressive manner.

Guided Decoding

In order to inject our preferences on topic, style, etc on the token candidates, we can use guided decoding to adjust the ranking scores for each token. Combined with global planning, we can derive the target sentence from the sampling strategies. Typically, the ranking score for token selection at each decoding step can be set as a combination of LM log-likelihood and a set of desired feature discriminators. Different approaches rely on different guiding principles to design their guiding decoding strategies. In particular, there are mainly three types of approaches to quantify the human preferences, including heuristics [81], supervised learning [80], and reinforcement learning (RL) [79]. And another type of approach is to tune a trainable decoding [83, 84, 85] to construct a new unnormalized distribution.

The second main paradigm is to fine-tune large language models [86, 87, 88, 89] to realize its potential in various generation tasks. However, this paradigm limits its power in parallel text generation tasks. In other words, since it requires a number of labeled data points, with both source input text and target output text, it cannot support the non-parallel controllable text generation tasks, such as several text style transfer tasks, which do not have any parallel corpora. And simultaneously, this paradigm only supports one language model for one task, which is heavily computational expensive for deployment. So a more efficient approach is more desirable for real-world applications, such as the prospect of "one general large model can be steered to various text generation tasks".

The third paradigm mitigates all these issues mentioned before, as prompt-based text generation tasks. That is, either by one crafted natural language prompts under zero-shot learning [13, 40, 26, 90] or constructing in-context demonstrations for few-shot learning [9]. While achieving non-trivial performance on a broad range of conditional text generation tasks, it also enables the steerability of language models for Non-parallel text generation tasks, such as text style transfer [91, 92]. In particular, [93] firstly proposes the framework of augmented zero-shot prompting for text style transfer with arbitrary styles, especially for very large language models (e.g., GPT-3). Our work takes a further step to directly optimize prompts for this type of non-parallel text generation task, for language models with various scales.

3.2 Prompted Text Style Transfer

In the controllable text generation section, we mainly focus on the unsupervised text style transfer task, a non-parallel controllable text generation task.

Text style transfer (TST) [92] is a challenging problem, whose goal is to rewrite an input sentence into a desired style, usually without supervised training data. For instance, in a sentiment transfer task, given a negative sentence "`The food is disgusting`", the model should generate a positive sentence "`The food is delicious`", without training on such paired data.

Even without supervision data, our method can learn prompts with weak reward signals, which is not possible for most previous prompt optimization methods. Compared to previous TST work that trained models from scratch [91, 7] or fine-tuned pre-trained LMs [94, 89, 95], our method presents a more efficient solution that learns discrete prompts for a LM without updating the massive parameters.

3.3 Reward Formulation

To enable text generation applications, the ideal setting would be greedily generate one output from the target language models, with beam search or greedy decoding, where you would have very stable reward for the same task input and prompt. However, as mentioned in §3.1, with these two techniques, there will be more generic, repetitive output. In our preliminary experiments, we do meet with the same issue that generated text will tend to repeat the same input sentence. Therefore, we have to turn to sampling-based decoding strategies, e.g., top-k sampling or top-p sampling (i.e., nucleus sampling).

Given input sentence $\mathbf{x}$, the goal of TST is to generate output $\mathbf{y}$ that preserves the information in $\mathbf{x}$ while showing style attribute s. Following these priorities, we define the task reward as a simple sum of content preservation and target style intensity, described formally below:

$$R(\mathbf{x},\mathbf{y},s) = \text{Content}(\mathbf{x},\mathbf{y}) + \text{Style}(\mathbf{y},s). \tag{3.1}$$

We implement the reward using common model-based metrics. Specifically, We implement our content preservation reward using its CTC metric [96], which measures the bi-directional information alignment between input $\mathbf{x}$ and output $\mathbf{y}$. We compute the alignment by matching token embeddings from RoBERTa-large similarly to BERTScore [97], a technique that shows the highest correlation with human judgments. For the style reward, we compute the target style probability under a BERT-base-uncased classifier learned from the training data.

However, a straightforward reward function is not enough to handle this problem. Specifically, for text generation task, we are pursing the input-conditioned prompt generation task guided by reward signals. In our case, with sampling-based decoding approaches, we unevitably face the issues of high variance, including 1) Even for the same input to the target language models, we would achieve different output sentence with different reward values; 2) In our input-conditioned setting, different inputs can have different levels of difficulty for reasoning and prediction. Prompted LMs can thus see different reward scales for different inputs, further enhancing its variance during optimization;

To handle these two challenges, we incorporate effective reward stablization which significantly improve the efficiency of optimization.

For the first point, we propose one reward bootstrapping technique that for the same input, instead of picking one sentence to evaluate its reward, we sample multiple sentences, and average the rewards as the final reward for particular reward. This technique has been verified as very useful in our stablization.

Then, on the other hand, we incorporate Input-Specific z-Score Reward techniques to reduce the variance. Specifically, as different inputs can have different levels of difficulty for reasoning and prediction, prompted LMs can see different reward scales for different inputs. In text style transfer, for instance, some sentences may only require changing a few words to alter the style, so the LM naturally achieves higher rewards on them than on other more complex sentences. Naively optimizing for all inputs with the same reward scale, therefore, can lead to training bias and instability. To mitigate this problem, we propose to use input-specific z-score, which normalizes the rewards by input-specific means and standard deviations. This can be seen as a case of adaptive reward normalization, a commonly-used technique in RL [98]. Formally, during prompt optimization, we sample a batch of prompts $Z(\mathbf{x})$ for each input $\mathbf{x}$, and compute the reward $R(\mathbf{y}_{\text{LM}}(\mathbf{z},\mathbf{x}))$ for each prompt $\mathbf{z} \in Z(\mathbf{x})$. After that, we compute the reward z-scores across prompts $Z(\mathbf{x})$. Using the shorthand $R_{\mathbf{x}}(\mathbf{z})$ for $R(\mathbf{y}_{\text{LM}}(\mathbf{z},\mathbf{x}))$, namely the reward prompt $\mathbf{z}$ receives for input $\mathbf{x}$, we write the transformation as below:

$$z\text{-score}(\mathbf{z},\mathbf{x}) = \frac{R_{\mathbf{x}}(\mathbf{z}) -_{\mathbf{z}'\in Z(\mathbf{x})} R_{\mathbf{x}}(\mathbf{z}')}{{}_{\mathbf{z}'\in Z(\mathbf{x})} R_{\mathbf{x}}(\mathbf{z}')}. \tag{3.2}$$

To distinguish the z-scores of different inputs in the same batch, we condition our policy network on the inputs, i.e., $\pi(\mathbf{z}|\mathbf{x})$.

3.4 Experimental Settings

3.4.1 Dataset Statistics

(1) Yelp [7] contains 266K positive and 177K negative reviews for training, 38K and 25K for validation, and 76K and 50K for testing, respectively. We perform evaluation on a separate dataset consisting of 500 reviews for each sentiment, with reference outputs collected by li-etal-2018-delete. **(2)** We use the Shakespeare [99] dataset compiled by jhamtani-etal-2017-shakespearizing, which contains 18K parallel sentence pairs from Shakespeare's plays and their modern translations for training, 1.2K for validation, and 1.4K for testing. We treat the dataset as a non-parallel corpus for training, but use the paired sentences as reference during evaluation. We preprocess both datasets with a simple text cleaning function to remove tokenization artifacts (e.g., "`it 's great .`" becomes "`it's great.`"). We include the function in our public codebase for reproducibility.

3.4.2 Evaluation Details

For automatic evaluation, We measure Content using the CTC metric [96] discussed earlier. To compute Style, we train BERT-base-uncased classifiers on both training and testing data, with validation accuracies of 98.4% and 93.7% on Yelp and Shakespeare, respectively. To evaluate Fluency, we rate output grammaticality using the classifier from krishna2020reformulating.[1] We also report popular metrics such as BLEU [100] and BERTScore [97] for content preservation, and perplexity (PPL) for fluency. To compute PPL, we fine-tune GPT-2 LMs on each TST dataset. For human evaluation, we enlist 5 graduate students who are fluent in English to rate Content, Style, and Fluency on a Likert scale of 1-5, and collect 3 ratings for each output. The average inter-rater agreement is 0.35 in terms of Fleiss' kappa [101], which is fair and similar to previous work [102].

3.4.3 Backbone LMs & Picking the Prompts

We experiment with GPT-2 of varying sizes, ranging from the smallest distilGPT-2 with 82M parameters to the largest GPT-2-xl with 1.5B parameters. We fix the prompt length $T = 5$. To generate output $\hat{\mathbf{y}}$, for all comparison methods, we sample 32 candidates from the respective models, and select the one with the highest reward.

Zero-shot Experiment Details

For sentiment transfer tasks, we evaluate our method against both training and prompting baselines. We compare with two strong training methods, Style Transformer [103] and DiRR [89]. In particular, DiRR fine-tunes GPT-2 [32] with RL signals, which can be seen as a full-model tuning analogue to our method. For the prompting baselines, we compare with (1) Null Prompt, which does not use any prompt, (2) Random Prompt, which samples 5 tokens

[1]https://huggingface.co/cointegrated/roberta-large-cola-krishna2020

from the vocabulary as prompts, and (3) Manual Prompt, which averages the performance of 3 human-written templates, one by reif2021recipe and two written for this experiment.

Few-shot Experiment Details

As discussed before, we experiment with few-shot text style transfer on the Shakespeare dataset. For the training baselines, we compare with Deep Latent [104] and STRAP [94], both trained on the full data. STRAP fine-tunes a GPT-2 [32] with self-supervised paraphrasing signals, which can be seen as a full-model tuning analogue to our method. We also compare with the same prompting baselines tested for Yelp. Both prompting baselines and our method use GPT-2-xl as the task LM.

3.5 Ablation Study

As discussed earlier, we transform the reward function for TST using input-specific z-score to mitigate the training instabilities caused by the different scales of reward across inputs. To study the impact of this technique on RL training, we compare our training success with and without z-score normalization. Specifically, we test on Yelp [7] using the distilGPT-2 model as an example. Following typical practice in RL, we run 5 experiments for each variant using different random seeds and compute the validation reward every 50 training steps. As the visualized results in Figures 3.2 and 3.3 show, z-score normalization achieves both superior performance and more stable improvement across random seeds and training tasks. Because training easily collapsed without z-score using the original hyperparameters, we tuned the reward shaping scheme to transform a scale of [50,100] into [-50,50], which substantially improved training stability and results.

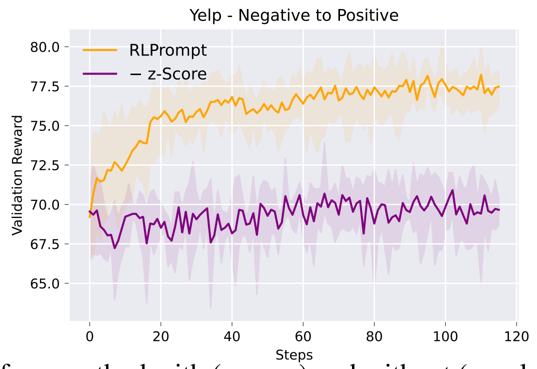


Figure 3.2. Comparison of our method with (orange) and without (purple) z-score reward normalization.

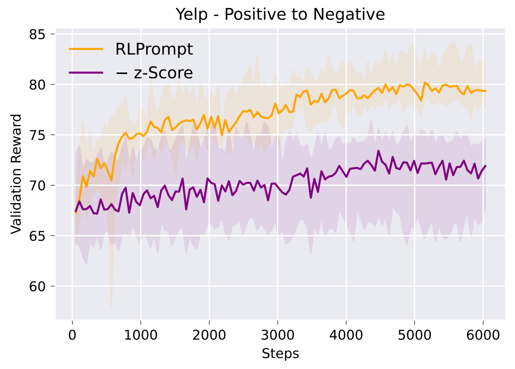


Figure 3.3. Comparison of our method with (orange) and without (purple) z-score reward normalization.

3.6 Results

Results for Sentiment Transfer

We present the automatic evaluation results for Yelp in Table 3.1. Compared to the expensive training baselines (Style Transformer and DiRR), our method with GPT-2-xl shows slightly lower content preservation and style accuracy, but have markedly better fluency, which leads to higher or competitive overall joint score $J(\cdot)$ and geometric mean GM$(\cdot)$. This may be because our method better preserves the LM's fluent generation capability by freezing its parameters. Relative to prompting baselines, our optimization strongly improves the default performance. In particular, our trained prompts performs better on average with lower variance than manual prompts, which sees performance vary wildly across prompts with similar meanings. We present all manual and learned prompts along with their performance in Table A.4 in

Table 3.1. Automatic evaluation of our method vs. baselines on the Yelp [7] sentiment transfer dataset.

Model	Content	Style	Fluency	J(C, S, F)	**GM**(C, S, F)
Training Baselines					
Style Transformer	75.2	96.4	58.6	46.1	75.2
DiRR	**78.8**	**97.7**	75.6	59.6	83.5
Prompting Baselines (GPT-2-xl)					
Null Prompt	37.4	94.8	97.6	33.6	70.2
Random Prompt	39.6	93.8	**97.8**	34.7	71.3
Manual Prompt	64.2 (6.8)	91.5 (3.6)	93.2 (1.4)	53.4 (7.9)	81.8 (3.4)
RLPROMPT ***(Ours)***					
distilGPT-2	57.3 (1.7)	96.5 (0.1)	85.3 (1.3)	46.0 (0.9)	77.9 (0.4)
GPT-2-small	60.0 (0.4)	96.4 (0.3)	89.0 (2.8)	50.7 (1.3)	80.1 (0.8)
GPT-2-medium	65.7 (1.4)	95.2 (1.2)	89.3 (0.1)	56.1 (1.0)	82.3 (0.4)
GPT-2-large	65.1 (1.8)	94.6 (2.3)	91.6 (0.8)	56.5 (1.3)	82.6 (0.7)
GPT-2-xl	72.1 (1.5)	94.2 (2.4)	89.5 (0.5)	**61.4 (2.2)**	**84.7 (1.0)**

Table 3.2. Human evaluation on Yelp on 5-Likert scale.

Model	Content	Style	Fluency	**GM**(C, S, F)
DiRR	**4.83**	**4.69**	4.64	**4.72**
Manual Prompt	4.25	4.38	**4.86**	4.49
RLPROMPT (Ours)	<u>4.41</u>	<u>4.68</u>	<u>4.80</u>	<u>4.63</u>

appendix. Within our own method, we can see the performance increasing monotonically from the smallest distilGPT-2 to the largest GPT-2-xl. Human evaluation results (Table 3.2) show similar conclusions, where our method is competitive with the costly training method DiRR by obtaining slightly lower content and style scores but higher fluency.

Results for Shakespeare Transfer

We present the automatic evaluation results for Shakespeare in Table 3.6 to illustrate our few-shot performance. Even with only 100 training examples and no update to the model, our method outperforms or gets close to training baselines using the full dataset such as Deep Latent and STRAP. STRAP is also limited to a subset of styles (e.g., authorship and formality), whereas our method accommodates a wider range of styles. Compared to prompting baselines, our method not only improves the performance, but also shows higher robustness to randomly-drawn training sets, as evidenced by the lower standard deviations for Content and Style.

3.7 Analysis

3.7.1 Fluent vs. Gibberish Prompts

We propose to optimize fluent prompts with top-k filtering qin2022cold. That is, we limit our policy's action space at each step t to the tokens with top-20 probabilities under a GPT-2 LM, conditioning on the previous prompt tokens $\mathbf{z}_{<t}$. Other than that, we train the policy using the same routine. To evaluate prompt perplexity, we use an out-of-the-box GPT-2 model.

Table 3.4. Comparison of prompt optimization with fluency constraint vs no constraint on the Yelp dataset.

Method	Prompt PPL↓	Content	Style	Fluency	J(C, S, F)	GM(C, S, F)
RLPROMPT	254K (238K)	**72.1 (1.5)**	94.2 (2.4)	89.5 (0.5)	**61.4 (2.2)**	**84.7 (1.0)**
+ Fluency	**82.1 (2.4)**	52.4 (1.5)	**96.2 (0.9)**	**94.6 (1.0)**	46.7 (0.7)	78.1 (0.4)

3.7.2 Transferring Prompts across LMs

For text style transfer, We use the prompts trained for each size of GPT-2 (from the smallest distil to the largest xl) to perform generation using every other model, and present the average performance over 5 evaluations in the heatmap of Figure 3.4. We also include Manual Prompt for comparison and Random Prompt for the baseline performance without transfer. Manual Prompt shows uniformly worse performance than learned prompts with smaller models like distilGPT-2 and GPT-2-small, but generally better results with larger models like GPT-2-large and -xl, suggesting that human-written prompts may better activate larger models. Overall, all optimized prompts see some transfer, as evidenced by the uniformly better performance than Random Prompt, and the level of success depends on both the prompt training and generation models, similarly to classification.

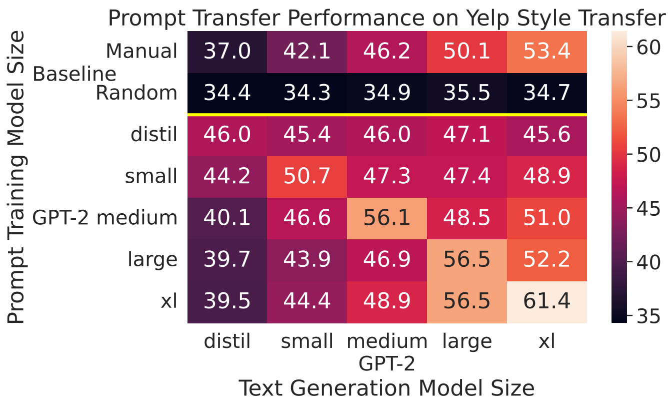


Figure 3.4. Heatmap of Yelp style transfer performance with transferred discrete prompts.

Chapter 4

Prompted Few-Shot Text Classification Task

In this chapter, I will provide an overview to prompted few-shot text classification task, and their progress along automatic prompt tuning. After that, I will elaborate how we design experiments, and how do we tackle the unstable issues (in other words, the highly varied reward in our RL training) faced in few-shot text classification experiments, i.e., reward engineering part. Next, I will show our experimental results compared to various previous state-of-the-art baselines, indicating the effectiveness of our proposed RLPROMPT in this task. And finally, I will provide both qualitative and quantitative analysis to the learned prompts for text-classification task.

4.1 Overview to Prompted Few-shot Text Classification

Learning text classification with few labeled examples has been a problem of interest in many applications [105, 106]. We adopt the typical prompting setting [9, 36] which solves classification by token infilling for an MLM like BERT or next-token prediction for a left-to-right LM like GPT-2. Classification, therefore, amounts to selecting tokens that correspond to a set of predetermined class labels, a.k.a., *verbalizers* (e.g., "`great`" for positive sentiment and "`terrible`" for negative sentiment). For instance, to classify the sentiment of an input sentence "`food is delicious`" using an MLM, we first fill our prompt and the input into a template

“`[Input] [Prompt] [MASK]`”, and then select the verbalizer token with the highest probability of filling into the `[MASK]` position.

4.2 Reward Formulation

To enable our RL training, we first insert one placeholder sentence into our prompt policy network, e.g. “*Text Classification:* ”, and then train the policy network (i.e., the multi-step decision process) with the reward during experience collection.

The simplest and the most straightforward reward designs come with two design principles: 1) **Accuracy**: we can simply resort to the few-shot accuracy as the reward. The intuition is straightforward, that is, better accuracy on these validation examples indicate better prompts; 2) **Prediction Confidence**: stronger prompts would give higher confidence in their prediction on the labeled validation examples.

In our experiments, for the accuracy part, it is common for us to find our RL training will stuck at certain point, with some biased prompts which can only show correct performance, e.g., 100% on the few-shot examples, which hugely limits the prompted performance. So finally, we mainly rely on the second principle.

However, when handling with this reward function, we still meet with some other challenges. Specifically, at the beginning, we evaluate our prompt per 10 steps in training on one particular validation example to speed up our training. However, we find that the training would be highly unstable that high variance dominates at the very beginning, which yet significantly hinders the sample efficiency. In order to handle this, for each evaluation step, we test the reward on combined few-shot examples together in order to pursue more stable reward.

But even with this formulation, the training is still not that stable, which limits its final performance under various random seeds. In order to handle this, our approach is illustrated as below:

Piecewise Reward

If a reward function is misspecified or vulnerable, the policy may maximize it without moving towards the desired goal. For example, while learning classification using the ground-truth probability as reward function, the policy may find adversarial prompts [107, 46] that lead to very high probabilities for a single class given arbitrary inputs. To overcome the issue, we propose to design piecewise reward functions [108, 109] with both smooth and disjoint components to better express the task priorities and improve robustness. Typically, we can include a dense, quantitative signal (e.g., label probability) to measure fine-grained progress towards the goal, and a sparse, qualitative signal only when certain states are achieved (e.g., certain accuracy on each class) by applying a large sudden increase in the reward.

In our case, the final reward function is defined as: The text classification task aims to correctly assign input text $\mathbf{x}$ to its ground truth label c from a set of classes $\mathscr{C}$. To mitigate the adversarial cases discussed in §2.4.4, we design a piecewise reward function that encourages prompts to classify *each* examples correctly. Given prompt $\mathbf{z}$ and training example $(\mathbf{x}, c)$, we compute the reward similarly to hinge loss as the gap between the label probability and the highest probability from other classes. Using the short hand $P_{\mathbf{z}}(c) := P_{\text{LM}}(c|\mathbf{z}, \mathbf{x})$ to denote the probability of label c, we can write the gap as $\text{Gap}_{\mathbf{z}}(c) := P_{\mathbf{z}}(c) - \max_{c' \neq c} P_{\mathbf{z}}(c')$. The gap value is positive when the prediction is correct, and negative otherwise. We denote $\text{Correct} := \mathbb{1}[\text{Gap}_{\mathbf{z}}(c) > 0]$. For a correct prediction, we multiply the positive reward by a large number to signal its desirability. The resulting reward function is as below:

$$R(\mathbf{x}, c) = \lambda_1^{1-\text{Correct}} \lambda_2^{\text{Correct}} \text{Gap}_{\mathbf{z}}(c), \tag{4.1}$$

During training, we compute the reward for prompt $\mathbf{z}$ by averaging over all our few-shot training examples. We set the balancing weights $\lambda_1 = 180$ and $\lambda_2 = 200$ by tuning on the validation set.

4.3 Experiments

4.3.1 Dataset Statistics

Following previous work [1, 6], we experiment on a wide range of popular few-shot classification tasks including sentiment classification such as SST-2 [31], Yelp Polarity [110], MR [111], CR [112], SST-5 [31], and Yelp [110], and topic classification such as AG's News [110]. We additionally experiment on Subj [113], TREC [114], Yahoo [110], and DBPedia [115]. We train with 16 examples per class, and validate using the same number of examples, in keeping with the standard few-shot setting [116].

Table 4.1. Main datasets evaluated in this work.

Dataset	**Type**	$\lvert C \rvert$	**|Train|=|Dev|**	**|Test|**	**Manual template**	**Label words**
SST-2	Sentiment (Movie reviews)	2	$16 \times \lvert C \rvert$	1.8k	<S> It was `[MASK]` .	terrible, great
Yelp P.	Sentiment (Yelp reviews)	2	$16 \times \lvert C \rvert$	38k	<S> It was `[MASK]` .	terrible, great
MR	Sentiment (Movie reviews)	2	$16 \times \lvert C \rvert$	2k	<S> It was `[MASK]` .	terrible, great
CR	Sentiment (Product reviews)	2	$16 \times \lvert C \rvert$	2k	<S> It was `[MASK]` .	terrible, great
SST-5	Sentiment (Movie reviews)	5	$16 \times \lvert C \rvert$	2.2k	<S> It was `[MASK]` .	terrible, bad, okay, good, great
Yelp	Sentiment (Yelp reviews)	5	$16 \times \lvert C \rvert$	50k	<S> It was `[MASK]` .	terrible, bad, okay, good, great
Subj	Subjectivity (Movie reviews)	2	$16 \times \lvert C \rvert$	2k	<S> This is `[MASK]` .	subjective, objective
AG's News	Topic (News articles)	4	$16 \times \lvert C \rvert$	7.6k	`[MASK]` News: <S>	World, Sports, Business, Tech
TREC	Topic (Question types)	6	$16 \times \lvert C \rvert$	0.5k	`[MASK]`: <S>	Description, Entity, Expression, Human, Location, Number
DBPedia	Topic (Wikipedia ontologies)	14	$16 \times \lvert C \rvert$	70k	`[Category: [MASK]]` <S>	Company, Education, Artist, Sports, Office, Transportation, Building, Natural, Village, Animal, Plant, Album, Film, Written
Yahoo	Topic (Question types)	10	$16 \times \lvert C \rvert$	60k	Topic `[MASK]`: <S>	culture, science, health, education, computer, sports, business, music, family, politics

4.3.2 Baselines

We compare our approach with representative methods in the diverse training and prompting paradigms shown in Table 2.1. Additionally, we compare with the latest Black-Box (BB) Tuning sun2022black, which mixes discrete and soft prompts and tunes the soft part. We describe more details in Appendix.

4.3.3 Backbone LMs & Picking the Prompts

We use RoBERTa-large [33] as our backbone model. For our approach, we experiment with prompt lengths $T \in \{2,5\}$, and insert the prompt tokens at the same positions with our manual prompts [36, 37].[1] Please see Appendix for more training details.

4.4 Ablation Study

As mentioned before (§2.4.4), misspecified or vulnerable reward functions can prevent the policy from discovering truly strong-performing prompts. To address this challenge, we propose to design piecewise reward functions that provide bonus to qualitative behaviors such as achieving certain accuracies on each class. As our reward function for few-shot classification adopts this design, we assess its effectiveness by ablating the piecewise component. Specifically, we test on SST-2 [31] and AG's News [110] using 5 prompt tokens with the distilRoBERTa-base model as an example. We run 5 RL experiments on the same few-shot dataset using different random seeds, and compute the validation accuracy every 50 steps. As the results in Figure 4.1 show, our piecewise reward function improves training stability by leading to strong-performing prompts more consistently, resulting in better average performance across random seeds and datasets.

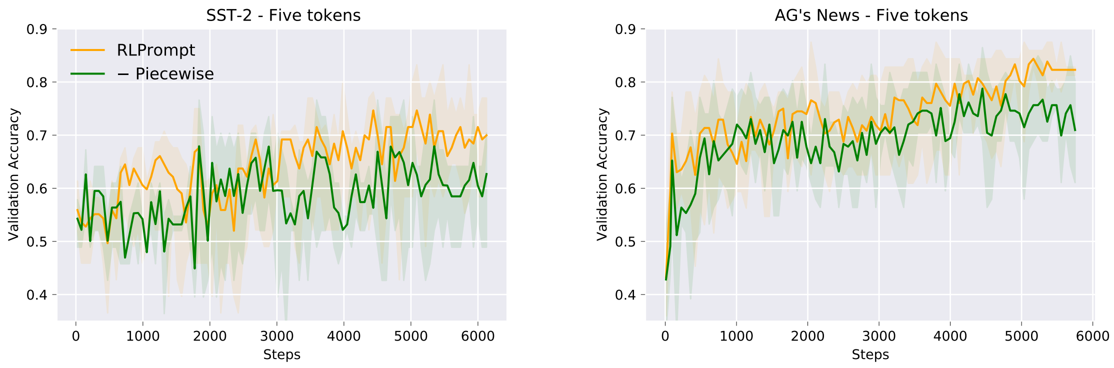


Figure 4.1. Comparison of our method with (orange) and without (green) piecewise reward function for few-shot classification.

[1]It is known that increasing prompt length and/or inserting prompt tokens in multiple positions can often lead to improved performance. We leave further experiments to the future.

Table 4.2. Results of few-shot text classification.

	SST-2	**Yelp P.**	**MR**	**CR**	**SST-5**	**Yelp**	**AG's News**	**Avg.**
Fine-Tuning	80.6 (3.9)	88.7 (4.7)	67.4 (9.7)	73.3 (7.5)	40.7 (3.0)	**51.0** (2.2)	**84.9** (3.6)	69.5
Manual Prompt	82.8	83.0	80.9	79.6	34.9	42.1	76.9	68.6
Instructions	89.0	84.4	85.2	80.8	29.8	43.0	54.8	58.5
In-Context Demonstration	85.9 (0.7)	89.6 (0.4)	80.6 (1.4)	85.5 (1.5)	39.3 (0.9)	49.4 (0.3)	74.9 (0.8)	72.2
Prompt Tuning *(Soft Prompt Tuning)*	73.8 (10.9)	88.6 (2.1)	74.1 (14.6)	75.9 (11.8)	40.2 (6.5)	49.1 (3.1)	82.6 (0.9)	69.2
BB Tuning *(2 soft tokens)*	83.2 (3.5)	86.0 (1.6)	77.1 (3.9)	83.2 (2.5)	39.2 (2.4)	41.5 (1.9)	74.0 (1.9)	69.2
BB Tuning *(5 soft tokens)*	84.6 (4.0)	78.7 (2.3)	79.8 (1.5)	82.9 (3.6)	36.6 (2.1)	33.7 (2.3)	73.6 (3.6)	67.1
BB Tuning *(Mixed, 50 soft tokens)*	89.1 (0.9)	93.2 (0.5)	86.6 (1.3)	87.4 (1.0)	38.4 (1.1)	44.8 (1.3)	83.5 (0.9)	74.7
GrIPS *(Discrete Prompt Enumeration)*	87.1 (1.5)	88.2 (0.1)	86.1 (0.3)	80.0 (2.5)	32.0 (1.8)	47.2 (0.5)	65.4 (9.8)	69.4
AutoPrompt	75.0 (7.6)	79.8 (8.3)	62.0 (0.8)	57.5 (5.8)	27.8 (3.3)	29.0 (5.0)	65.7 (1.9)	56.7
RLPrompt (Ours, 2 discrete tokens)	90.3 (1.3)	94.1 (0.8)	86.5 (1.2)	87.4 (1.7)	40.1 (1.9)	45.6 (3.8)	76.8 (1.4)	74.4
RLPrompt (Ours, 5 discrete tokens)	**92.5** (0.8)	**95.1** (1.0)	**87.1** (0.4)	**89.5** (0.6)	**41.4** (3.2)	44.8 (4.3)	80.2 (0.7)	**75.8**

4.5 Results

Main Results

We present our few-shot classification results in Table 4.2. Our method (5 tokens) outperforms Manual Prompt and Instructions on all datasets, as well as In-Context Demonstration and Fine-Tuning on all but 1 and 2 datasets, respectively. Compared to Prompt Tuning, our method achieves higher average accuracy with lower standard deviations, showing our approach is less sensitive to various training factors, a common issue for few-shot prompt tuning [17, 19]. Our approach substantially outperforms BB Tuning with soft prompts, and is slightly better even after BB Tuning uses mixed discrete/soft prompts with 50 soft tokens. Compared to previous discrete prompt optimization methods such as GrIPS [12] and AutoPrompt [15], our method reaches superior accuracy on all benchmarks. On the additional datasets which tend to be multi-way (e.g., 16-class), Fine-Tuning shows higher performance, but our method continues the lead over prompting baselines.

Table 4.3. Additional results of few-shot text classification. The best result on each dataset is **bolded** and the second best result <u>underscored</u>.

	Subj	TREC	Yahoo	DBPedia	Avg.
Fine-Tuning	**89.0** (3.5)	**83.9** (5.5)	**65.6** (2.4)	**97.7** (0.8)	**84.1**
Manual Prompt	51.5	31.8	18.1	59.2	40.2
Instructions	50.4	26.2	21.4	15.9	28.5
In-Context Demonstration	51.9 (1.3)	29.2 (2.0)	36.7 (2.1)	76.6 (0.4)	48.6
Prompt Tuning *(Soft Prompt Tuning)*	73.0 (7.3)	49.6 (6.1)	<u>59.7</u> (1.3)	84.2 (5.3)	66.6
BB Tuning *(2 soft tokens)*	75.7 (3.4)	40.4 (2.5)	41.7 (1.4)	60.9 (6.0)	54.7
BB Tuning *(5 soft tokens)*	75.8 (4.4)	39.8 (4.6)	38.2 (1.8)	62.7 (4.1)	54.1
BB Tuning *(Mixed, 50 soft tokens)*	71.8 (5.1)	46.4 (8.2)	50.0 (0.9)	<u>90.2</u> (0.8)	64.6
GrIPS *(Discrete Prompt Enumeration)*	74.8 (1.1)	9.5 (0.2)	22.5 (0.4)	22.1 (2.9)	32.2
AutoPrompt	78.9 (4.5)	38.8 (4.3)	35.5 (2.0)	63.1 (2.0)	54.1
RLPrompt (2 discrete tokens)	<u>81.9</u> (1.2)	<u>60.5</u> (3.3)	48.6 (0.6)	76.0 (0.6)	66.8
RLPrompt (5 discrete tokens)	81.2 (1.7)	57.6 (4.6)	48.6 (1.0)	84.6 (1.9)	<u>68.0</u>

4.6 Analysis

4.6.1 Training Efficiency

To assess the training efficiency of our method, we compare our test accuracy across training steps with BB Tuning, which is also a gradient-free method but optimizes soft prompts. As Figure 4.2 shows, our RL-based method is as efficient as soft prompt tuning without access to

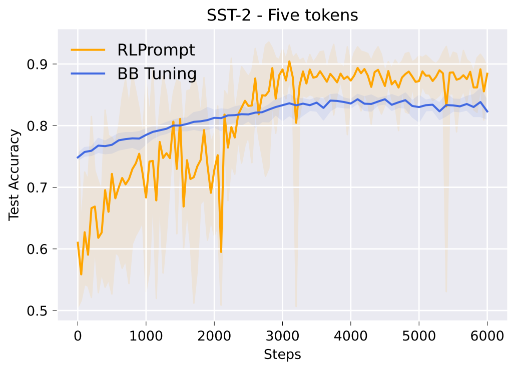


Figure 4.2. Comparison of our method (orange) and Black-Box (BB) Tuning [6] (blue) in terms of training efficiency.

LM gradients, converging in similar number of steps to BB Tuning, but with superior performance. Our training is also relatively stable, for even the worst prompts encountered after convergence perform comparably to BB Tuning on average.

4.6.2 Transferring Prompts across LMs

Specifically for classification, we train prompts on various sizes of RoBERTa and GPT-2 and apply them to every other model for classification. We tabulate the average performance over 5 runs in the heatmap of Figure 4.3. Overall, all prompts can transfer between models, but the success depends on both the source and target LMs. For example, prompts learned from larger models see sharp performance declines when applied to smaller models, indicating that the structures they activate in large LMs may be less present in smaller ones. In contrast, prompts learned from smaller models reach similar or better performance on larger models (e.g., RoBERTa-base to -large). Perhaps surprisingly, prompts learned from MLMs like RoBERTa transfer well to left-to-right LMs like GPT-2 and vice versa, showing the LM structures they activate are largely shared across model types. These findings open up a promising and exciting direction for future research—enabled by the transferrability across LMs, we may learn a prompt cheaply from smaller models, and apply it to a larger, more powerful model for inference.

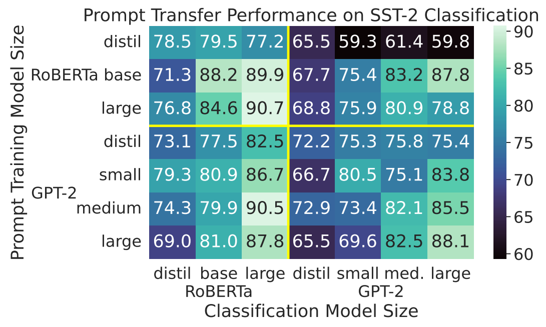


Figure 4.3. Heatmap of sentiment analysis performance with transferred discrete prompts of 2 tokens.

4.6.3 Robustness to Classification Verbalizers

It is known that prompted classification is sensitive to verbalizer choices. Manual design requires domain expertise and understanding of the base LMs. Previous research devised various methods for automatic verbalizer search [117, 15, 1]. In few-shot classification, our method can discover well-performing prompts given a wide variety of verbalizers. Table 4.4 shows the results on SST-2 with several intuitive verbalizers, averaged over 3 random seeds for each verbalizer pair. Across different verbalizers, our prompts consistently outperform manual prompt with smaller variation, showing our approach is robust to the choice of verbalizers. We report similar results on AG's News in Table 4.5 in the appendix.

Table 4.4. Comparison of RLPROMPT and manual prompt on SST-2 using different verbalizers.

Verbalizers	RLPROMPT	Manual
terrible, great	**92.8** (0.8)	82.8
bad, good	**91.2** (1.4)	79.7
negative, positive	**92.2** (0.6)	76.8

Table 4.5. Comparison of our method vs. Manual Prompt on AG's News using different verbalizers.

Verbalizers	RLPROMPT	Manual
World, Sports, Business, Tech	**77.6** (1.5)	76.9
Global, Athletics, Finance, Technology	**65.3** (0.5)	63.5

4.6.4 Qualitative Analysis of Prompt Tokens

Empowered by the transparency of discrete tokens, we investigate the prompts we learned for classification to characterize the similar patterns learned by different LMs discovered by the prompt trasfer analysis. In particular, we frequently find semantically similar tokens among our learned prompts, which we name "strong words" and list in Table 4.6. These strong words make sense in the context of their specific tasks, indicating the LMs may indeed capture certain human-understandable patterns during pre-training. For instance, "absolutely" may signal strong opinion before judging a sentence as positive or negative, whereas "News" appears to be a hint for classifying the topic of a news piece. Besides these semantically meaningful prompt tokens, we also find some unintelligible prompts that nevertheless achieve good performance on downstream tasks, or so-called "secret language [118] of the LM" (e.g., "`imentariesariesaryary`" can reach 80% accuracy with RoBERTa-large on AG's News).

Table 4.6. The performance of manual prompt examples by composing strong words from Table 4.6 for both sentiment analysis and news topic classification across RoBERTa-large and GPT-2-large.

Task Category	Strong Words
Sentiment Analysis	**Absolutely**, **absolutely**, **Totally**, **downright**, profoundly, VERY, Very, Really, highly
News Classification	**News**, **Reviewer**, **Reports**, **reported**, **Staff**, **Information**, **Statement**, Stories, Guide, say,

Template	RoBERTa	GPT-2
SST-2		
<S> downright `[MASK]` .	80.6	86.7
<S> Really downright `[MASK]` .	90.4	89.1
<S> Absolutely `[MASK]` .	91.7	87.8
<S> AbsolutelyAbsolutely `[MASK]` .	89.2	72.3
<S> Absolutely VERY absolute VERY absolute `[MASK]` .	92.7	73.8
AG's News		
`[MASK]` Reviewer <S>	74.5	—
`[MASK]` Reviewer Stories <S>	81.0	—
`[MASK]` StaffInformationStatement <S>	76.8	—
`[MASK]` StaffInformationStatement Reviewer Stories <S>	79.8	—

Beyond finding strong words, we also study whether we can construct strong-performing prompts by arbitrarily composing these strong words, which can provide insight into whether LMs use these strong words compositionally. To this end, we construct several prompts, evaluate their downstream performance, and tabulate the results in Table **??**. Interestingly, composing more strong words indeed can lead to improved performance, but the level of success is sensitive to various factors, such as word order and the specific tokens we choose, indicating that existing LMs are still brittle even when responding to discrete tokens learned from optimization.

Appendix A

Additional Experiment Details

A.1 Policy Network

For all tasks, we uniformly use distilGPT-2 ([39]) with 82M parameters as a compact policy LM, and implement a generously parameterized MLP with 1 hidden layer and 2048 hidden states. Given distilGPT-2's hidden size of 768, we only add 3.1M parameters, or 3.8% of the LM parameters.

A.2 Additional Implementation Details for Text Style Transfer

In training, we sample 4 prompts for each input using top-50 sampling from our policy network. During sampling, we bias all logits by -10 to encourage exploration. For each prompt, we generate outputs using top-10 sampling, and bootstrap the reward 4 times to reduce variance. For SQL training, we set the target learning rate to be 10^{-3}, and shape the reward from a scale of [0,1] to [-20,80]. We optimize the prompt generator using an Adam optimizer with learning rate 10^{-4}, except for Yelp negative-to-positive and Shakespeare using GPT-2-large and GPT-2-xl models, which we train with learning rate 5×10^{-5}. We train 2 inputs per batch for 6K steps if learning rate is 10^{-4}, and 12K steps if the learning rate is 5×10^{-5}. Also using the RTX 3090 GPU, each experiment typically takes from 10 hours using distilGPT-2 to 1 day using GPT-2-xl. To reduce the performance variance caused by sample selection and RL initialization, we average the performance from 5 evaluation runs for each of 3 RL experiments using our own method. Additionally, we perform the same sample selection for all our baselines for comparable performance. For Shakespeare training baselines, we do not perform sample selection in order to avoid biasing the full-dataset models with our few-shot style classifiers.

Table A.1. Additional automatic evaluation results on Yelp [7] sentiment transfer.

	Content Preservation		Fluency
Model	BLEU	BERTScore	PPL$\downarrow$
Training Baselines			
Style Transformer	27.6	56.1	78.2
DiRR	**30.0**	**61.7**	40.6
Prompting Baselines (GPT-2-xl)			
Null Prompt	6.6	35.8	59.5
Random Prompt	7.3	37.4	60.5
Manual Prompt	19.2 (4.1)	53.1 (5.0)	35.5 (9.0)
RLPrompt *(Ours)*			
distilGPT-2	15.7 (0.7)	49.1 (0.6)	43.6 (0.6)
GPT-2-small	16.5 (0.4)	51.3 (0.6)	37.8 (4.8)
GPT-2-medium	20.0 (1.2)	55.1 (1.1)	34.4 (0.8)
GPT-2-large	19.8 (0.5)	54.7 (0.7)	34.9 (1.4)
GPT-2-xl	24.2 (1.2)	59.0 (0.8)	**34.3 (0.9)**

A.3 Additional Implementation Details for Text Classification

Baseline Implementation Details

For Manual Prompt, we take the hand-crafted prompts from schick2021exploiting. For Instructions, we manually create task descriptions and label definitions following mishra2021NI's protocol (shown in Table A.4) and prepend the instructions to the inputs. For In-Context Demonstration [9], we randomly select one training example per class and concatenate them with the input texts. For Prompt Tuning [24], we replace the Manual Prompt tokens with five soft tokens in the same positions for fair comparison, and optimize them using Adam optimizer with learning rate 1×10^{-2} and batch size 16 for 400 epochs. For Black-Box Tuning [6] with mixed prompt, we use 50 soft tokens and 8,000 budget following the default setting. For its soft-prompt-only setting, we also optimize with the same budget. For Fine-Tuning, we train with Adam optimizer with learning rate 1×10^{-5} and batch size 16 for 100 epochs. For Discrete Prompt Enumeration, we take GrIPS [12] as a state-of-the-art example. For AutoPrompt [15], we use 5 prompt tokens and perform prompt search with a batch size of 16 using the few-shot training examples. For each baseline, we pick the model with the best validation accuracy for evaluation.

Additional Training Details

During training, we explore the prompt space using top-256 sampling from the policy network, whose input is just one placeholder word "classification". To update the parameters, we use an Adam [119] optimizer with learning rate 5×10^{-5}. Furthermore, we multiply all rewards by 5 to increase the reward scale of well-performing prompts, and apply z-score normalization (§**??**) across prompts for more efficient learning. We train the policy with 16 prompts per batch for 6K steps for 2 tokens, 12k steps for 5 tokens, and compute validation performance every 10 steps. Using an NVIDIA GeForce RTX 3090 GPU, each experiment typically takes from 1.5 hours using distilRoBERTa-base to 4 hours using RoBERTa-large. During evaluation, we average the performance of 3 prompts with the highest validation accuracy for each experiment. Due to the instability and inherent randomness of the few-shot setup [120, 1], we sample 5 different training and validation sets, run 3 experiments per set with different random seeds, and report the average accuracy and standard deviation.

A.4 Task Instructions & Prompts

Bibliography

[1] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *ACL*, pages 3816–3830, 2021.

[2] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *TACL*, 8:423–438, 2020.

[3] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[4] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[5] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

[6] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*, 2022.

[7] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, 30, 2017.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, pages 1877–1901, 2020.

[10] 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. *arXiv preprint arXiv:2107.13586*, 2021.

[11] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

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

[13] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL-IJCNLP*, pages 6691–6706, 2021.

[14] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[15] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, pages 4222–4235, 2020.

[16] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, 2018.

[17] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597, 2021.

[18] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*, 2021.

[19] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*, 2021.

[20] Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[21] Jing Qian, Li Dong, Yelong Shen, Furu Wei, and Weizhu Chen. Controllable natural language generation with contrastive prefixes. In *Findings of ACL*, pages 2912–2924, 2022.

[22] Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*, 2022.

[23] Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sameer Singh, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, et al. Prompt waywardness: The curious case of discretized interpretation of continuous prompts. *arXiv preprint arXiv:2112.08348*, 2021.

[24] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059, 2021.

[25] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. Warp: Word-level adversarial reprogramming. In *ACL-IJCNLP*, pages 4921–4933, 2021.

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[27] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

[28] Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*, 2022.

[29] Xi Ye and Greg Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*, 2022.

[30] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192, 2022.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, 2013.

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[33] 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. *arXiv preprint arXiv:1907.11692*, 2019.

[34] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021.

[35] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473, 2019.

[36] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, pages 255–269, 2021.

[37] Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. Improving and simplifying pattern exploiting training. In *EMNLP*, pages 4980–4991, 2021.

[38] Shibo Hao, Bowen Tan, Kaiwen Tang, Hengzhe Zhang, Eric P Xing, and Zhiting Hu. BertNet: Harvesting knowledge graphs from pretrained language models. *arXiv preprint arXiv:2206.14268*, 2022.

[39] HuggingFace. Distilgpt2. https://huggingface.co/distilgpt2., 2019.

[40] Han Guo, Bowen Tan, Zhengzhong Liu, Eric P Xing, and Zhiting Hu. Text generation with efficient (soft) q-learning. *arXiv preprint arXiv:2106.07704*, 2021.

[41] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? *arXiv preprint arXiv:2109.01247*, 2021.

[42] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *ICML*, pages 12697–12706. PMLR, 2021.

[43] 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 *ACL*, pages 7871–7880, 2020.

[44] 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. *JMLR*, 21:1–67, 2020.

[45] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. *arXiv preprint arXiv:2205.14865*, 2022.

[46] Lei Xu, Yangyi Chen, Ganqu Cui, Hongcheng Gao, and Zhiyuan Liu. Exploring the universal vulnerability of prompt-based learning paradigm. *arXiv preprint arXiv:2204.05239*, 2022.

[47] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

[48] Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E Peters. Learning from task descriptions. In *EMNLP*, pages 1361–1375, 2020.

[49] Avia Efrat and Omer Levy. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*, 2020.

[50] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. *arXiv peprints arXiv:2104.08773*, 2021.

[51] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2022.

[52] Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *EMNLP*, pages 2856–2878, 2021.

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

[54] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[55] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

[56] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

[57] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022.

[58] Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. On the relation between sensitivity and accuracy in in-context learning. *arXiv preprint arXiv:2209.07661*, 2022.

[59] Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*, 2022.

[60] Xi Ye and Greg Durrett. The unreliability of explanations in few-shot in-context learning. *arXiv preprint arXiv:2205.03401*, 2022.

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

[62] Taylor Sorensen, Joshua Robinson, Christopher Rytting, Alexander Shaw, Kyle Rogers, Alexia Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. An information-theoretic approach to prompt engineering without ground truth labels. In *Proceedings*

of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 819–862, 2022.

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

[64] 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 *ICML*, pages 2790–2799. PMLR, 2019.

[65] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

[66] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*, 2022.

[67] 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. *arXiv preprint arXiv:2110.07602*, 2021.

[68] Jordan Clive, Kris Cao, and Marek Rei. Control prefixes for text generation. *arXiv preprint arXiv:2110.08329*, 2021.

[69] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. *arXiv preprint arXiv:2203.05557*, 2022.

[70] Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, et al. Standing on the shoulders of giant frozen language models. *arXiv preprint arXiv:2204.10019*, 2022.

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

[72] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[73] Pengcheng He, Baolin Peng, Liyang Lu, Song Wang, Jie Mei, Yang Liu, Ruochen Xu, Hany Hassan Awadalla, Yu Shi, Chenguang Zhu, et al. Z-code++: A pre-trained language model optimized for abstractive summarization. *arXiv preprint arXiv:2208.09770*, 2022.

[74] Xiachong Feng, Xiaocheng Feng, Libo Qin, Bing Qin, and Ting Liu. Language model as an annotator: Exploring dialogpt for dialogue summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1479–1491, 2021.

[75] Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, 2020.

[76] Miao Chen, Xinjiang Lu, Tong Xu, Yanyan Li, Jingbo Zhou, Dejing Dou, and Hui Xiong. Towards table-to-text generation with pretrained language model: A table structure understanding and text deliberating approach. *arXiv preprint arXiv:2301.02071*, 2023.

[77] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*, 2022.

[78] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040, 2022.

[79] Jiwei Li, Will Monroe, and Dan Jurafsky. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*, 2017.

[80] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, 2018.

[81] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, 2017.

[82] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, 2018.

[83] Jiatao Gu, Kyunghyun Cho, and Victor OK Li. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, 2017.

[84] Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *International Conference on Learning Representations*.

[85] Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. *Advances in neural information processing systems*, 32, 2019.

[86] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[87] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[88] Zachary M Ziegler, Luke Melas-Kyriazi, Sebastian Gehrmann, and Alexander M Rush. Encoder-agnostic adaptation for conditional language generation. *arXiv preprint arXiv:1908.06938*, 2019.

[89] Yixin Liu, Graham Neubig, and John Wieting. On learning text style transfer with direct rewards. In *NAACL*, pages 4262–4273, 2021.

[90] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

[91] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International conference on machine learning*, pages 1587–1596. PMLR, 2017.

[92] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205, 2022.

[93] Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. A recipe for arbitrary text style transfer with large language models. *arXiv preprint arXiv:2109.03910*, 2021.

[94] Kalpesh Krishna, John Wieting, and Mohit Iyyer. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, 2020.

[95] Zhiting Hu and Li Erran Li. A causal lens for controllable text generation. *Advances in Neural Information Processing Systems*, 34:24941–24955, 2021.

[96] Mingkai Deng, Bowen Tan, Zhengzhong Liu, Eric Xing, and Zhiting Hu. Compression, transduction, and creation: A unified framework for evaluating natural language generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7580–7605, 2021.

[97] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[98] Hado P van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29, 2016.

[99] Wei Xu. From shakespeare to twitter: What are language styles all about? In *Proceedings of the Workshop on Stylistic Variation*, pages 1–9, 2017.

[100] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[101] Joseph L Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619, 1973.

[102] Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. Evaluating style transfer for text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 495–504, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[103] Ning Dai, Jianze Liang, Xipeng Qiu, and Xuan-Jing Huang. Style transformer: Unpaired text style transfer without disentangled latent representation. In *ACL*, pages 5997–6007, 2019.

[104] Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. A probabilistic formulation of unsupervised text style transfer. *arXiv preprint arXiv:2002.03912*, 2020.

[105] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. Lifelong domain word embedding via meta-learning. *arXiv preprint arXiv:1805.09991*, 2018.

[106] Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513*, 2018.

[107] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *EMNLP*, 2019.

[108] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan C. Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, pages 1094–1100. PMLR, 2020.

[109] Desik Rengarajan, Gargi Nikhil Vaidya, Akshay Sarvesh, Dileep M. Kalathil, and Srinivas Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. In *ICLR*, 2022.

[110] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *NeurIPS*, 28, 2015.

[111] Bo PANG. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 2005.

[112] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD*, pages 168–177, 2004.

[113] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*, 2004.

[114] Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207, 2000.

[115] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[116] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *NeurIPS*, 34, 2021.

[117] Timo Schick, Helmut Schmid, and Hinrich Schütze. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5569–5578, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[118] Giannis Daras and Alexandros G. Dimakis. Discovering the hidden vocabulary of dalle-2. *ArXiv*, abs/2206.00169, 2022.

[119] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[120] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, volume 32, 2018.