Prompt-Based Length Controlled Generation with Multiple Control Types

Anonymous ACL submission

Abstract

Large language models (LLMs) have attracted great attention given their strong performance on a wide range of NLP tasks. In practice, users often expect generated texts to fall within a specific length range, making length controlled generation an important topic, especially for GPT-style models. Existing length control methods mostly focus on a simple control type of "equal to" a target length. Different from them, we propose a prompt-based method to achieve length controlled generation under different control types with high accuracy. In particular, we adopt reinforcement learning (RL) and sample filtering with the reward signal given by rule-based reward models, which enhances the length control ability of models by rewarding outputs that follow certain control instructions. In addition, we introduce a standard prompt extractor to parse arbitrary users' input into standard control instructions. Experiments show that our method significantly improves the accuracy of prompt-based length control on popular summarization datasets like CNNDM and NYT under multiple control types. Moreover, both the standard prompt extractor and RL-tuned model show strong generalization to unseen control prompt templates.

1 Introduction

For recent popular GPT-style models like ChatGPT and GPT-4 (Radford et al., 2018, 2019; Liu et al., 2023b; OpenAI, 2023), various studies have been conducted on them, and the inference efficiency and computational cost often draw concerns from the community (Zhang et al., 2023; Zhao et al., 2023; Bubeck et al., 2023). Since its generation is in an autoregressive manner, the inference cost increases continually with the growing of decoding steps. Meanwhile, users of LLMs usually have an expected length of generated texts, no matter for writing an essay or summary, knowledge QA or dialogue generation (Fan et al., 2018; Liu et al., 2020, 2022; Mirshekari et al., 2021; Gupta et al., 2021). Both of these two facts require the length of generation in GPT-style models can be effectively controlled.

For pretrained language models (PLMs), the most widely applied technique for length control is prompt-based fine-tuning (Raffel et al., 2020; Goyal et al., 2022; Zhang et al., 2022; Liu et al., 2023a). Taking an example of length-controlled summarization (LCS), we can prepend a prompt "`summarize with length` l_i:" to the article to be summarized in training, where l_i is the number of words of reference summary. However, this process is usually performed in supervised fine-tuning (SFT), where this length controllable ability has to compromise with the goodness of downstream tasks. For large-scale models like GPT-3, the length controlled ability can be somewhat activated by in-context learning without any fine-tuning (Brown et al., 2020; Chowdhery et al., 2022; Dong et al., 2022). However, it relies on the size and power of the pre-trained foundation models to achieve good performance. For other methods like RLHF (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022), it is expensive to manually label whether the generated length meets the requirement given in instruction prompts.

Generally, there are many other length control methods such as GOLC, LenAtten and LAAM (Liu et al., 2018; Takase and Okazaki, 2019; Makino et al., 2019; Yu et al., 2021; Liu et al., 2022). However, these methods are not particularly designed for PLMs, thus architecture-specific designs on training mechanisms are usually needed. Moreover, they often focus on the setting of equalling to a certain length, generally not adapt to other control types such as greater/smaller than a value, or between two values, *etc*. Meanwhile, they can not handle diverse expressions of control instructions from users. Therefore, how to effectively connect diverse control instructions

from users to the length of generated text for PLMs is still an open question.

In this paper, we introduce a novel method that applies prompt-based fine-tuning with reinforcement learning to improve the performance of length controlled generation, which is capable to handle multiple types of length control at the same time. Our main contributions are:

- We design a rule-based reward model for multiple control types other than traditional "equal to" control type, which can provide accurate reward values for both reinforcement fine-tuning and inference of PLMs.
- We introduce an independent standard prompt extractors (SPE) to parse the length control instructions from diverse user inputs to standard control prompts (SCP), which is necessary for rule-based reward and show strong generalization power.
- We apply a Proximal Policy Optimization (PPO) algorithm with a modified state space to fine-tune GPT models for enhancing their length control ability. Two modes including (a) SCP + rule-based reward; (b) SCP + model-based reward are introduced.
- Experiments show that by applying reinforcement fine-tuning and sample filtering, the length-control errors can be significantly reduced from the baseline prompt-based method. Moreover, the method show strong generalization to unseen prompt templates.

2 Related work

2.1 Reinforcement learning for text generation.

Reinforcement learning (RL) (Kaelbling et al., 1996) has been widely applied to improve text generation performance, including summarization (Stiennon et al., 2020; Paulus et al., 2018), question generation (Pang and He, 2021), and dialogue generation (Li et al., 2016; Zhou et al., 2017; Jaques et al., 2020). In general, we can consider the generative model as the policy network and optimize its parameters for achieving higher reward from the environment (Paulus et al., 2018; Wang et al., 2022). Human feedback is one of the most known strategies to get the reward, which is shown to be more effective than using some automatic metrics, such as rouge scores in text generation (Christiano et al., 2017; Stiennon et al., 2020; Wu et al., 2021). Existing study (Ramamurthy et al., 2023) also shows that RL techniques are generally better than supervised methods at aligning language models to human preferences. It is known that Reinforcement learning from Human Feedback (RLHF) plays a key role in the success of autoregressive LLMs like InstructGPT (Ouyang et al., 2022), which uses human feedbacks to train a reward model for PPO (Schulman et al., 2017).

2.2 Length control for text generation

Length control is an important ability for text generation, especially for tasks with a large variance of output length, such as summarizing texts using a desired range of number of words/tokens. Early work (Fan et al., 2018) on controlling lengths in abstractive summarization quantizes summary length into discrete bins, and expands the input vocabulary with special tokens to indicate the length bins of the ground-truth summary during training. (Liu et al., 2018) extends a convolutional sequence to sequence model to control the length of summarization. To generate summaries of any desired length, a length constrain factor is added to each convolutional block of the initial layer. (Takase and Okazaki, 2019) proposes an extension of a sinusoidal positional encoding to enable neural encoder-decoder model to generate a text of any desired length. GOLC (Makino et al., 2019) dedicates to increase the probabilities of generating a high quality summary within a desired length by using minimum risk training. LenAtten (Yu et al., 2021) introduces a length attention unit to break the trade-off between length controllability and summary quality. LAAM (Liu et al., 2022) modifies the attention matrix based on length-budget dynamically during the decoding process. Generally, we notice that existing length control approaches can not be directly applied for control targets other than "equal to" a certain length, and are in lack of focusing on prompt-based method for the most recent trend of GPT-style LLMs.

3 Method

3.1 Model Architecture

Our model architecture is presented in Figure 1. The original user utterances may include the control instruction on length constraint, which differs from factual and semantic information in terms of that the length control can be easily checked by rule-based methods. For instance, if we can

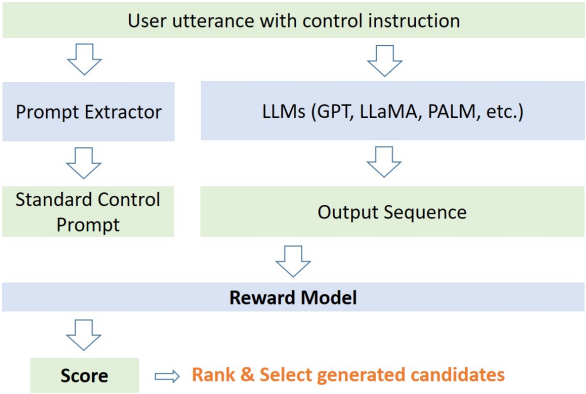

Figure 1: Overview of the model architecture. In training stage, the scores given by the reward model are used for the reinforcement learning method. In inference stage, the scores are applied for ranking and selecting the output sequences generated by PLM/LLMs.

Standard Control Prompt	Reward
more than L_t	$-\max(0, L_t - L_g)$
less than L_t	$-\max(0, -L_t + L_g)$
equal to L_t	$-\lvert L_t - L_g \rvert$
between L_L and L_U	$-(\max(0, L_L - L_g) + \max(0, L_g - L_U))$

Table 1: Reward function for each Standard Control Prompt (SCP)

somehow understand user intention on length constraint, we can set up the rule for ranking and selecting generated candidates. Therefore, we introduce a standard prompt extractor (SPE) (See Section 3.3) to parse the information of length constraint from user utterance and thus generate a standard control prompt. This standard prompt includes different types of length constraint and can be applied for rule-based inference and evaluation.

As Figure 1 shows, the user utterance is first passed through both the SPE and PLM/LLMs like GPT-family (Brown et al., 2020; OpenAI, 2023), PALM (Chowdhery et al., 2022), LLaMA (Touvron et al., 2023), Pangu (Ren et al., 2023), Ernie (Sun et al., 2020), etc. PLMs are the core modules that generate an output sequence according to the user utterance, and SPE outputs a standard control prompt (SCP) that includes user intention on the control type and target lengths. Secondly, the reward model takes both the SCP and generated sequence as input, and outputs a score to evaluate how well the generated sequence meets the requirement of length control instruction (See Section 3.2). Finally, this score can be applied as the reward signal in reinforcement learning method to fine-tune PLMs (See Section 3.4), or be applied as a filtering rule to rank and select the generated sequences in inference (see Section 3.5).

3.2 Reward model

To evaluate whether the generated text follows the length control instruction, we introduce a reward model to score the generated sequences according to the required length from the user's input. This score can be used as a reward for fine-tuning existing PLMs by leveraging reinforcement learning, or be used to rank and select the candidates generated by PLMs. In this study, we design a **rule-based reward model**, which takes the actual length of the output sequence and target values as the inputs, and calculate the rewards using the reward functions depending on the type of SCPs, as is shown in Table 1, where L_t, L_L, L_U and L_g refer to the target length, the lower-bound length, the upper-bound length and the actual generated length, respectively. The type of SCPs and target lengths are parsed from user's input as is shown in Figure 1. The rule-based method provides the accurate feedback on whether the output meets the requirement of length given by SCPs, while the latency is almost negligible compared with using a neural model (*e.g.,* BERT or GPT) for scoring. However, it relies on extracting exact standard control information from the user's input. We also discuss the model-based reward models in Appendix A.5.

3.3 Standard Prompt Extractor

Standard prompt extractor (SPE) takes a user's input, and outputs standard control prompt (SCP) if exists. This standard prompt consists of a basic description of what length constraint should be satisfied. We design two types of SCPs as shown in Figure 2. In particular, this prompt extractor can be a **generative model** such as GPT, in which case the extractor is trained directly to generate the full text of SCP as shown in Figure 2 (left). Then we can easily get L_t, L_U and L_L of Table 1 from this generated control text. On the other hand, we can also use a **discriminative model** such as BERT, as the prompt extractor, in which case it is required to predict the type of SCP and the target numbers involved, as shown in Figure 2 (right). In this case, we prepend three `[CLS]` tokens to the utterance. Three linear projection layers with different output sizes (*i.e.,* number of types of control instruction as in the left column of Table 1, number of pos-

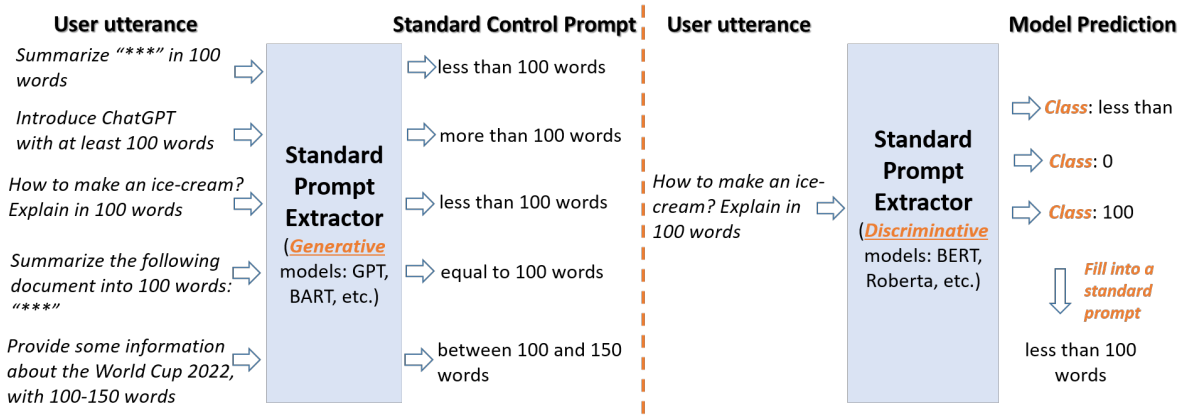

Figure 2: The demonstration of Standard Prompt Extractor (SPE). The generative type of models are trained to output the standard control prompts (SCPs) directly (left), while the discriminative type of models are trained to predict the type of each control instruction, as well as the requested number of lengths from user utterance, such as the minimum value and the maximum value (right).

sible minimum values, number of possible maximum value) map the three top vectors of `[CLS]` tokens to fill in the type, minimum value and maximum value of a standard prompt template. Therefore, we have three classification objectives for predicting the ground truth of SCP. Note that we can indeed use only the minimum and maximum target values to fully represent the control instructions under all the four types in Table 1. For example, the minimum target value is 0 means the control type of "`smaller than`" the maximum target value. Since this setting has only two classification objectives, two `[CLS]` tokens and corresponding linear projection layers are introduced.

3.4 Reinforcement Learning for length control fine-tuning

We apply a modified PPO method with actor-critic setting (Grondman et al., 2012; Bahdanau et al., 2017; Schulman et al., 2017). Since rewarding of the generated text length does not rely on the input article, both the reward model and critic model only take the concatenation of SCPs and generated texts as input. Meanwhile, as the reward for length control can only be determined when the generation ends, we can just calculate the reward using the final output. Assume $\pi_\theta(a|s)$ is a stochastic policy given by the PLM, where θ is the trainable parameter, s is the whole input sequence, and a is the finally generated sequence. Let s' be the SCP. The original policy gradient (PG) applies the loss function given by Eq. (1):

$$L^{PG}(\theta) = -\hat{\mathbb{E}}_D[\log \pi_\theta(a|s)\hat{A}], \quad (1)$$

where $\hat{\mathbb{E}}_D[.]$ is the empirical average over a finite batch of samples from dataset D. $\hat{A}$ is an estimator of the advantage function at the end of generation, showing the goodness of current policy *w.r.t.* the baseline in terms of control accuracy. For the actor-critic case, we set $\hat{A} = R(s', a) - \hat{Q}_{\theta_{old}}(s', a)$, where $R(.)$ is the reward model, $\hat{Q}_{\theta_{old}}(s', a)$ is the expected Q value by the model of the last step. Note that the reward only depend on the standard control prompt s' and the generated sequence a (without the input context). As s' itself is not associated with the control reward, it is hard to define a value function on it. Thus, we apply $\hat{Q}_{\theta_{old}}(s', a)$ instead of $V_{\theta_{old}}(s')$ as the baseline of the current step. The original PG empirically often leads to a large policy update and thus instability during fine-tuning. Therefore, we follow PPO (Schulman et al., 2017) to use the probability ratio $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ instead of $\log \pi_\theta(a|s)$ in Eq. (1), and utilizes a clipped surrogate objective given by Eq. (2) to stabilize the policy updates and ensure that the probability ratio term is bounded between $[1-\epsilon, 1+\epsilon]$.

$$L^{CLIP}(\theta) = -\hat{\mathbb{E}}_D[\min(r(\theta)\hat{A}, \text{Clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A})]. \quad (2)$$

To ensure sufficient exploration, we also follow the original paper of PPO to introduce an entropy term $S = \frac{1}{n}\sum \pi_\theta(a|s)\log(\pi_\theta(a|s))$, in which the average is taken across the vocabulary dimension. In addition, a penalty for large KL divergence is added between the current and old stochastic policy distributions (i.e. π_θ and $\pi_{\theta_{old}}$). Therefore, the

total policy loss can then be rewritten as:

$$L^{CLIP+S+KL}(\theta) = \hat{\mathbb{E}}_D[L^{CLIP}(\theta) - cS[\pi_\theta|(s)] + \beta D_{KL}(\pi_\theta|\pi_{\theta_{old}})],$$

where c, β are coefficients, $D_{KL}(\pi_\theta|\pi_{\theta_{old}})$ is the KL-divergence between the old and current action probability distributions. To avoid the performance loss for downstream tasks, we involve an extra term of SFT loss from the same batch of labelled data on the actor's policy loss: $L^A(\theta) = L^{CLIP+S+KL}(\theta) + \lambda L^{SFT}(\theta)$, where λ is a tunable hyper-parameter. Meanwhile, we optimize a value loss $L^{VF} = (Q_\theta(s', a) - \hat{R})^2$. More details of the algorithm are given in Appendix A.1.

3.5 Inference & Sample filtering

In inference, a well fine-tuned PLM is expected to directly process user inputs, and generate a text sequence following the user's intention on length control. Since the control instruction from the user inputs can be diverse in practice, our proposed prompt extractor serves as an important role to parse user inputs into SCPs to benefit the latter RL fine-tuning. Meanwhile, with the extracted type and value information from SPEs, we can apply reward models (as described in Section 3.3) to score, rank and select from a set of generated samples in beam sampling, which is named as sample filtering in our method. Let $k = \text{argmax}_i R(s', a_i)$, where R is the reward model, a_i is the i-th sequence in all N output sequences, then a $a = a_k$ is selected to be the final output sequence. Thereafter, this selected sequence can be used for either the RL fine-tuning phase or the final inference for validating to what extent the length control ability can be achieved in existing PLMs.

4 Experiments

4.1 Experimental Setup

We perform experiments on two popular summarization datasets including CNNDM (Hermann et al., 2015) and NYT (Durrett et al., 2016). CNNDM contains news articles from the CNN and Daily Mail websites, with labelled abstractive and extractive summaries. There are 287,226 training samples, 13,368 validation samples and 11,490 test samples. NYT contains 110,540 articles with abstractive summaries. We follow its paper to split the original dataset into 100,834 training and 9,706 test examples. After tokenized by GPT-2 tokenizer, the reference summaries in CNNDM have an average length of 71 tokens with a standard deviation of 28 tokens, while the reference summaries in NYT have an average length of 104 tokens with a standard deviation of 28 tokens. The following subsections explain how to train and use different modules of our method. The details of hyper-parameters is in Appendix A.3.

4.1.1 Data processing and augmentation

We design a set of standard control prompts, including five control types: "`more than ** tokens`", "`less than ** tokens`", "`equal to ** tokens`", "`between ** and ** tokens`" and "`none`". "`**`" means the value of expected length from user intention, and "`none`" means no length constraints. For each type, we randomly sample a target summary length from 50 to 150 tokens based on the general news summary length, and fill these lengths into "`**`" field of a randomly sampled SCP. To further simulate real user utterances with length control intention, around 20 possible augmented prompt templates are introduced for each SCP. Examples of templates are shown in Figure 2 and Appendix A.2. Finally, we can create augmented input data by replacing the placeholders in the augmented templates with target lengths and original articles.

4.1.2 Training of standard prompt extractor

As introduced in Section 3.3, we train two types of models, *i.e.,* generative and discriminative models, to serve as a standard prompt extractor. In particular, we fine-tune the GPT2-small model as a generative extractor and the BERT-small model as a discriminative extractor. Both two pre-trained checkpoints are obtained from huggingface (Wolf et al., 2019). We use the above augmented input data to fine-tune models. To make it clear, we use the original articles of CNNDM and NYT, and first sample a SCP for each article, and then sample an augmented prompt template from a pre-designed set. Next, we randomly assign the target length values between 50 and 150 to each article to form the finalized augmented template. Each original article associated with its augmented template serves as input data, and its corresponding SCP serves as the expected prediction, to finally train the standard prompt extractor. Results of evaluating SPEs are given in Table 2. "Acc. Gen." means we use 30% of randomly sampled augmented control prompts as out-of-sample templates for evaluation, and only train the SPE models on the re-

Extractor	Acc.	Acc. Gen.
BERT-base-cls-2	**99.9**	**99.9**
BERT-base-cls-3	99.7	99.8
GPT-small	97.7	97.5

Table 2: Evaluation on the accuracy and generalization of standard prompt extractors (SPEs). "cls-2" and "cls-3" refer to only predicting the minimum and maximum values, or predicting the control type as well. "Acc." is the prediction accuracy on an in-sample test set, while "Acc. Gen." denotes the generalization performance of SPEs on unseen prompt templates.

maining 70% templates. We can see that BERT-base-cls-2 can achieve almost 100.0% test accuracy for extracting SCPs, and it also generalizes well for out-of-sample control prompts that are not seen in training. The accuracy of GPT-small is relatively lower, for which the reason may be that fully matching the whole generated texts is harder than extracting key values. The learning curves are presented in Appendix A.5. Overall, a well-trained SPE does not introduce much noise or performance drop in our end-to-end implementation. We use BERT-base-cls-2 as the discriminative extractor in later experiments to achieve clear and accurate minimum and maximum target values.

4.1.3 Supervised Fine-Tuning of GPT models

To build the baseline summarization model with length control ability, we apply three pre-trained GPT-2 models with 124M, 355M and 774M parameters from Huggingface, denoted as GPT-S, GPT-M, GPT-L, respectively. We randomly split the original training dataset into four parts with approximately equal size, and each is augmented with one type of SCP. According to the actual text length of the reference summary, we then randomly sample one (for "`less than **`" or "`more than **`") or two (for "`between ** and **`") target lengths between 50 and 150 while ensuring that the range contains the reference summary length. For the control type of "`equal to **`", the target value is fixed to the actual length of reference summary. To simulate real user utterances with control instruction, we build augmented utterances by first randomly sampling prompt templates equally distributed across four control types (given in Table 7 in Appendix), and then replacing the placeholders by the original articles and sampled target values. Next, we prepend the corresponding SCP to the augmented original input (separated by "`:`") to formulate the model input of each example. Note that SCPs can be assumed to be known when given the user's input and high accuracy of SPEs, thus the formulation of model inputs is also applicable in the inference. Finally, we perform supervised fine-tuning on the data to enable pre-trained GPTs to summarize texts with a length control ability.

	MU	EQ	MO	LE	BT
CNNDM	28.7	43.3	43.6	2.8	32.9
NYT	22.9	33.7	19.9	12.9	21.5

Table 3: Averaged length control errors of comparing the actual length of reference summary to our sampled length control instructions on test set.

4.1.4 Fine-Tuning with Reinforcement Learning

On top of the above supervised fine-tuned GPTs, that is baseline, we further propose to improve the accuracy of length control via reinforcement learning with the PPO method as described in Section 3.4. In other words, the backbone PLMs in our method are these supervised fine-tuned GPTs that to some extent have already owned the ability of controlling generated text lengths. Again, for augmenting the input articles from the original datasets, we follow the similar data processing as like supervised fine-tuning mentioned above. Except that we randomly sample target lengths between 50 and 150 (not associated with reference summary length). We use the proposed rule-based reward model with the parsed standard control information (i.e. control type and target values).

Exploratory experiments show that actor-critic generally works better than actor-only, thus in the main experiments we use actor-critic setting. We apply AdamW optimizers without learning rate schedule, while the detailed hyper-parameter setting are given in Appendix.

4.2 Results

4.2.1 Baseline Method

We build the length control test set by sampling control instructions for each reference summary from the test sets of both two datasets, and all the following experiments are performed on it. Similar to RL, we randomly sample target length between 50 and 150 for each example. We define length control error as the negative reward in Ta-

Model	Setting	CNNDM					NYT				
		R1↑	R2↑	RL↑	B.S.↑	Error↓	R1↑	R2↑	RL↑	B.S.↑	Error↓
GPT-S	Prompt	37.76	15.58	38.05	62.32	18.16	47.22	29.47	42.01	67.76	17.62
	Prompt+RL	37.52	15.31	**38.79**	**62.42**	14.29	47.30	29.84	42.36	67.81	10.53
	Prompt+filter	**38.04**	**16.29**	37.12	62.05	10.57	**47.88**	**30.55**	**42.50**	**67.87**	8.06
	Prompt+RL+filter	37.48	16.01	37.20	61.88	**7.06**	47.84	30.43	42.26	67.54	**3.89**
GPT-M	Prompt	**38.85**	15.93	38.48	**63.02**	21.32	48.34	30.74	43.64	68.75	13.17
	Prompt+RL	38.30	15.89	**39.29**	62.90	6.59	48.23	30.58	43.61	68.67	12.61
	Prompt+filter	**38.85**	**17.29**	37.68	62.48	11.21	**49.73**	**32.65**	**44.55**	**69.00**	6.75
	Prompt+RL+filter	37.83	16.89	37.20	61.91	**4.98**	49.41	32.18	44.05	68.40	**3.65**
GPT-L	Prompt	38.27	16.37	**38.92**	**63.09**	6.89	49.41	32.20	44.31	69.36	10.64
	Prompt+RL	38.23	16.42	38.86	63.06	6.62	49.35	32.24	44.31	69.27	8.52
	Prompt+filter	**38.75**	**16.85**	38.23	62.85	3.34	**50.04**	**32.65**	**44.35**	69.48	4.82
	Prompt+RL+filter	38.70	16.52	38.39	62.98	**3.22**	50.01	32.52	44.14	**69.51**	**4.60**

Table 4: Comparison of methods in multiple-type control, where we consider all the four candidate types of control instructions in Table 1. In all cases, jointly using RL and sample filtering achieve the lowest control error.

ble 1 representing the average difference between the output length and the desired range. Then we use the actual length of reference summary to calculate length control errors as shown in Table 3, which can be considered as the baseline of length control errors. "MU" refers to test with sampled instruction equally distributed across all control types, "EQ", "MO" "LE", "BE" refer to test with sampled instructions for control types "Equal", "More" "Less", "Between", respectively. The results depend on the length distributions of labeled summaries.

4.2.2 Main Results

As Table 4 shows, we compare models with four different settings for prompt-based length control, including (1) **`Prompt`**: use GPTs with prompt-based SFT to control the output length; (2) **`Prompt+RL`**: the GPTs used in (1) but further enhanced with reinforcement learning; (3) **`Prompt+filter`**: the GPTs in (1) but equipped with sample filtering; and (4) **`Prompt+RL+filter`**: the enhanced GPTs with both RL and sample filtering, which is a combination of (2) and (3). For evaluation, we apply relevance scores including F1 of Rouge Scores (ROUGE, 2004) (denoted as "R1", "R2", "RL") and BertScore (Zhang et al., 2019) (denoted as "B.S"), and length control error (denoted as "Error"). We select the checkpoint with the lowest validation control error and less than 1 point's drop of BertScore for evaluation on the test set. For all methods with sample filtering, we set the number of output sequences to 8, and select the one with the highest reward.

Averaged results of multi-type control are presented in Table 4. Note that Rouge and BertScore can be less than the general state-of-the-art summarization models without length control, since our sampled length distribution can be different from reference summaries. In fact, the mean and standard deviation of the reference lengths are 71 and 28 tokens respectively for CNNDM, 104 and 35 tokens for NYT. The difference of control errors for two datasets can partly be due to their original length distributions. Overall, we can see that for all settings, our proposed RL method can achieve an improvement of length control ability with lower control errors. By further using sample filtering supported by the rule-based reward model, both `Prompt+filter` and `Prompt+RL+filter` can achieve lower control errors than not using sample filtering like the method (1) and (2). After checking the learning curves (see Appendix A.6), we also find that the relevance metric BertScore indeed does not have a clear decrease trend in early stage as the validation reward increases. It indicates that with our method, the relevance of texts can be preserved as the control errors reduces during the RL fine-tuning.

4.2.3 Comparing of different control types

We deconstruct the multiple-type controls and thus evaluate the effect of our proposed method on each particular control type. Results on both CNNDM and NYT are given in Table 5. In general, our proposed methods bring a significant improvement of length control accuracy (*i.e.,* Error) for all the four control types. Moreover, some insightful findings can be obtained from Table 5. As the average length of reference summary in CNNDM (71 to-

Model	Setting	CNNDM					NYT				
		R1↑	R2↑	RL↑	B.S.↑	Error↓	R1↑	R2↑	RL↑	B.S.↑	Error↓
Equal	Prompt	**38.14**	15.71	**38.91**	**62.62**	26.13	**47.61**	**30.36**	42.75	**67.85**	27.98
	Prompt+RL	35.67	14.64	38.73	61.86	13.61	47.57	30.33	**42.88**	67.82	18.81
	Prompt+filter	37.90	**16.26**	37.42	61.89	12.47	47.60	30.32	42.02	67.80	17.80
	Prompt+RL+filter	37.56	16.10	38.15	62.23	**8.35**	47.58	30.29	42.15	67.71	**8.72**
Less	Prompt	**37.08**	**15.74**	36.64	**61.88**	0.47	46.11	28.96	41.32	**67.07**	10.33
	Prompt+RL	37.03	15.64	**36.87**	61.75	0.38	45.75	28.91	41.08	66.84	0.96
	Prompt+filter	36.92	15.72	35.90	61.17	0.22	**46.68**	29.87	41.53	66.87	2.09
	Prompt+RL+filter	36.90	15.72	35.87	61.13	**0.21**	46.65	**30.43**	**42.03**	65.96	**0.32**
More	Prompt	38.00	15.43	37.82	**62.41**	39.94	44.01	27.12	40.22	66.62	2.27
	Prompt+RL	35.75	14.83	**38.88**	61.79	13.77	42.45	25.94	39.89	65.85	1.32
	Prompt+filter	**38.53**	**16.44**	37.64	62.13	23.05	**47.78**	**30.63**	**42.39**	**68.00**	1.42
	Prompt+RL+filter	37.43	16.26	37.92	62.22	**6.01**	47.75	30.53	42.27	68.94	**1.01**
Between	Prompt	36.38	15.03	38.65	61.96	5.76	**47.65**	**30.07**	41.90	67.52	18.63
	Prompt+RL	36.10	14.95	**38.99**	61.80	4.53	47.09	29.74	**42.18**	**67.63**	10.75
	Prompt+filter	**38.06**	**16.43**	37.44	**62.07**	1.15	47.13	29.70	41.37	67.47	6.76
	Prompt+RL+filter	37.85	16.28	37.45	62.00	**1.09**	47.58	30.02	42.05	67.50	**3.18**

Table 5: Comparison of four control types in the multiple type control setting using GPT-S on NYT datasets.

kens) is much less than the average of sampled target lengths, *i.e.,* 100 tokens, therefore, to generate with "`more than`" a sampled target length is harder than "`less than`" for all candidate methods. However, the `Prompt+RL+filter` can still provide a significantly large improvement on the control type of "`more than`", by reducing the Error from 41.9 to 6.0. In the case of "`less than`" with sample filtering, the RL method does not further reduce the validation error as it is already quite low, thus the default checkpoint is always selected even after RL fine-tuning.

4.3 Generalization to unseen templates

To evaluate if the tuned model can generalize to unseen prompt templates of length control, we conduct an extra experiment by tuning on a 70% subset of prompt templates randomly sampled from Table 7 in the Appendix, and check our model performance with the rest test of unseen prompt templates, as give in Table 6. The difference between "In-sample" and "Out-sample" setting is whether the out-of-sample set of control prompt templates is applied for training. We notice that in some cases, there is a slight performance degradation on out-of-sample prompt templates, but the length control ability is still significantly better than baseline method. This demonstrates that our proposed method has strong generalization to novel prompt templates. We believe with a larger set of prompt templates in training, this generalization power can still be largely improved.

Type	Setting	R1	R2	RL	B.S.	Error↓
NYT	Baseline	47.2	29.5	42.0	67.8	17.6
	In-sample	47.8	30.4	42.3	67.5	3.9
	Out-sample	47.7	30.2	42.3	67.1	4.1
CNNDM	Baseline	37.8	15.6	38.1	62.3	14.7
	In-sample	37.6	15.3	38.8	62.3	7.6
	Out-sample	37.7	15.4	38.7	62.4	8.1

Table 6: Generalization to out-of-sample control templates of GPT-S for multi-type length control.

5 Conclusion

The paper proposes a method for improving the length control ability of GPT-style PLMs under multiple control types, especially for the domain of text summarization. The standard prompt extractor and rule-based reward model are introduced to provide an accurate control signal for both fine-tuning and inference. We apply a modified PPO algorithm for enhancing the length controlled generation. In the inference, sample filtering is further introduced for selecting a generated sample that follows the instruction. The method is proved to be effective for three sizes of GPT-2 models on both CNNDM and NYT summarization datasets. Compared to the baseline using prompt-based strategies on GPTs, our method further achieves a significant improvement in terms of control accuracy. Moreover, it can process diverse length-control prompts with strong generalization ability to new prompt templates.

6 Limitations

The limitations of our study involve the following aspects. First, similar to RLHF implemented in InstructGPT, finetuning with RL may result in a decrease of the language modeling evaluation metric. Well designed in-context learning or introducing adaptors/LoRA particularly tuned for length control may be potential solutions for this. Second, the control performance relies on the goodness of standard prompt extractor. When the generative one is applied, it is possible to generate outputs that can not be fully parsed with rule-based method. Third, when the discriminator is applied for filtering the generated samples in inference, usually a large beam size is required, thus longer inference time and computing cost may needed. As the probability distribution across all tokens are available in auto-regressive generation, this extra cost can be well scaled.

References

D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*, 2017.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv:2303.12712*, 2023.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv:2204.02311*, 2022.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui. A survey for in-context learning. *arXiv:2301.00234*, 2022.

G. Durrett, T. Berg-Kirkpatrick, and D. Klein. Learning-based single-document summarization with compression and anaphoricity constraints. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

A. Fan, D. Grangier, and M. Auli. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, 2018.

T. Goyal, J. J. Li, and G. Durrett. News summarization and evaluation in the era of gpt-3. *arXiv:2209.12356*, 2022.

I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

P. Gupta, J. P. Bigham, Y. Tsvetkov, and A. Pavel. Controlling dialogue generation with semantic exemplars. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 3018–3029, 2021.

K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. *Advances in*

Neural Information Processing Systems (NeurIPS), 2015.

N. Jaques, J. H. Shen, A. Ghandeharioun, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard. Human-centric dialog training via offline reinforcement learning. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research (JAIR)*, 4:237–285, 1996.

J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

B. Liu, H. Wei, D. Niu, H. Chen, and Y. He. Asking questions the human way: Scalable question-answer generation from text corpus. In *Proceedings of The Web Conference 2020*, pages 2032–2043, 2020.

P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35, 2023a.

Y. Liu, Z. Luo, and K. Zhu. Controlling length in abstractive summarization using a convolutional neural network. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Y. Liu, Q. Jia, and K. Zhu. Length control in abstractive summarization by pretraining information selection. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, et al. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. *arXiv:2304.01852*, 2023b.

T. Makino, T. Iwakura, H. Takamura, and M. Okumura. Global optimization under length constraint for neural text summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

M. Mirshekari, J. Gu, and A. Sisto. Conquest: Contextual question paraphrasing through answer-aware synthetic question generation. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 222–229, 2021.

OpenAI. Gpt-4 technical report, 2023.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

R. Y. Pang and H. He. Text generation by learning from demonstrations. In *International Conference on Learning Representations (ICLR)*, 2021.

R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*, 2018.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 21(1):5485–5551, 2020.

R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. Choi. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. In *International Conference on Learning Representations (ICLR)*, 2023.

X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang, A. Podolskiy, G. Arshinov, A. Bout, I. Piontkovskaya, J. Wei, X. Jiang, T. Su, Q. Liu, and J. Yao. Pangu-Σ: Towards trillion parameter language model with sparse heterogeneous computing, 2023.

L. C. ROUGE. A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization of ACL*, 2004.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, pages 8968–8975, 2020.

S. Takase and N. Okazaki. Positional encoding to control output sequence length. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and

G. Lample. Llama: Open and efficient foundation language models, 2023.

X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao. Deep reinforcement learning: a survey. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv:1910.03771*, 2019.

J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike, and P. Christiano. Recursively summarizing books with human feedback. *arXiv:2109.10862*, 2021.

Z. Yu, Z. Wu, H. Zheng, Z. XuanYuan, J. Fong, and W. Su. Lenatten: An effective length controlling unit for text summarization. In *Findings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

C. Zhang, C. Zhang, S. Zheng, Y. Qiao, C. Li, M. Zhang, S. K. Dam, C. M. Thwal, Y. L. Tun, L. L. Huy, et al. A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need? *arXiv:2303.11717*, 2023.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2019.

Y. Zhang, X. Zhang, X. Wang, S.-q. Chen, and F. Wei. Latent prompt tuning for text summarization. *arXiv:2211.01837*, 2022.

W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv:2303.18223*, 2023.

L. Zhou, K. Small, O. Rokhlenko, and C. Elkan. End-to-end offline goal-oriented dialog policy learning via policy gradient. *arXiv:1712.02838*, 2017.

A Appendix

A.1 Algorithm for length controlled fine-tuning with our modified PPO

Following the explanations in Section 3.4, we further provide an algorithm table for our modified PPO fine-tuning in Algorithm 1.

A.2 Examples of standard control prompt and augmented control prompt templates

The SCPs and corresponding augmented prompt templates for generating the augmented input with length control information are given in Table 7. In the experiments, we use the augmented prompts to train and evaluate the standard prompt extractor. For the backbone PLMs and reward models, SCPs can be considered as available, given the high performance of SCPs.

A.3 Hyper-parameter settings

In this section, we provide hyper-parameter settings of different modules and training stages of our method, where we denote hyper-parameter as "HP" in the tables. For the standard prompt extractor, the hyper-parameter settings are given in Table 8. For the trainable reward models, the hyper-parameter settings are given in Table 9. For pretraining of GPT summarization models with control prompts, the hyper-parameter settings are given in Table 10. For enhancing control ability with reinforcement finetuning, the hyper-parameter setting are given in Table 11.

HP	BERT extractor	GPT extractor
pretrained model	BERT-small	GPT-small
optimizer	AdamW	AdamW
batch size	32	64
lr	2E-05	2E-05
β_1	0.9	0.9
β_2	0.999	0.999
weight decay	1E-07	0
num iterations	200k	200k

Table 8: Hyper-parameter setting of Standard Prompt Extractors.

HP	BERT reward	GPT reward
pretrained model	BERT-large	GPT-medium
optimizer	AdamW	AdamW
batch size	64	32
lr	0.00005	0.00005
β_1	0.9	0.9
β_2	0.999	0.999
weight decay	0	0
num iterations	200k	200k

Table 9: Hyper-parameter setting of trainable reward models.

HP	GPT-S	GPT-M	GPT-L
optimizer	AdamW	AdamW	AdamW
batch size	64	64	64
lr	5E-05	5E-05	2E-05
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999
weight decay	1E-06	1E-06	1E-06
num iterations	200k	200k	200k

Table 10: Hyper-parameter setting of prompt-based SFT on pretrained GPT models.

HP	GPT-S	GPT-M	GPT-L
optimizer	AdamW	AdamW	AdamW
actor_lr	3E-07	3E-07	3E-07
critic_lr	0.0003	0.0003	0.0003
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999
actor_adam_eps	1E-07	1E-07	1E-07
critic_adam_eps	1E-07	1E-07	1E-07
weight decay	0	0	0
epochs	1	1	1
update timestep	512	512	512
surrogate epoch	16	16	16
surrogate batch size	32	16	8
β	0.1	0.1	0.1
c	0.01	0.01	0.01
ϵ_{clip}	0.2	0.2	0.2
λ	1.0	1.0	1.0

Table 11: Hyper-parameter setting reinforcement learning for pretrained GPT models. $\epsilon_c lip$ is the clipping parameter ϵ shown in Eq. (2). β and c are weights for KL divergence and entropy as shown in Eq. (3.4). λ is the coefficient for SFT loss.

A.4 Extra Results

A.4.1 Single-type control

We also conduct experiments for traditional **single-type control**, where we only consider the strict SCP of "`equal to`" in both SFT and reinforcement fine-tuning. In details, for each example we randomly sample a augmented control prompt under the type of "equal" and replace the text placeholder with the input text and replace the length

Algorithm 1: Algorithm for controlled fine-tuning with modified PPO

Get a pretrained GPT model to initialize the policy network $\pi_{\theta_{old}}(a|s)$.
Initialize critic network $Q_\theta(s', a)$.
Initialize hyper-paramaters $N_{iteration}$, M, B, n_{epoch}, c, β.
for i<=1,...,$N_{iteration}$ **do**
  for j=1,...,M **do**
    Get an input sequence s_0 augmented with random sampled augmented control prompt from the data-loader.
    Run SPE to get the SCP s' from the input sequence.
    Run policy $\pi_{\theta_{old}}(a|s)$ for an input sequence with augmented control prompt s to get an output sequence a, policy $\pi_{\theta_{old}}$.
    Get the reward of output sequence a with reward model $r = r(s', a)$.
    Store input s, SCP s', generate sequence a, reward r and old policy $\pi_{\theta_{old}}$ into memory.
  end for
  for e=1,...,n_{epoch} **do**
    for b=1,...,B **do**
      Take the b-th mini-batch $(s', a, r, \pi_{\theta_{old}})$ from the memory.
      Use the actor and critic networks to get the new policy and value $\pi_\theta(a|s), Q_\theta(s', a)$.
      Compute the ratio $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$.
      Compute advantage estimate $\hat{A} = r - Q_{\theta_{old}}(s', a)$.
      Compute L^{CLIP} with Eq. (2).
      Compute the KL-divergence $D_{KL}(\pi_\theta|\pi_{\theta_{old}})$.
      Compute the Entropy $S[\pi_\theta|(s)]$.
      Compute the actor loss L^A_θ with Eq. (3.4).
      Update the policy network parameters θ with gradients of L^A_θ.
      Compute the value loss $L^Q_\theta = MSE(Q_\theta(s', a), r)$.
      Update theticularly tunedcritic network parameters θ with gradients of L^V_θ.
    end for
  end for
end for
return θ

Equal	Less	More	Between
summarize "*" with length ?	summarize "*" with length smaller than ?	summarize "*" with length larger than !	summarize "*" with length between ! and ?
summarize the following document with length ?: "*" '	summarize the following document with length smaller than !: "*"	summarize the following document with length larger than !: "*"	summarize the following document with length between ! and ?: "*"
Summarize with exactly ? tokens: *'	Summarize with less than ? tokens: *	Summarize with more than ! tokens: *	Summarize with between ! and ? tokens: *
I want a summary of "*" with exactly ? Tokens	I want a summary of "*" with less than ? Tokens	I want a summary of "*" with more than ! Tokens	I want a summary of "*" with between ! and ? Tokens
Give me a summary with ? tokens from "*"'	Give me a summary with less than ? tokens from "*"	Give me a summary with more than ! tokens from "*"	Give me a summary with between ! and ? tokens from "*"
Please summarize "*" with exactly ? Tokens	Please summarize "*" with less than ? Tokens	Please summarize "*" with more than ! Tokens	Please summarize "*" with between ! and ? Tokens
Write a summary of "*" with exactly ? Tokens	Write a summary of "*" with less than ? Tokens	Write a summary of "*" with more than ! Tokens	Write a summary of "*" with between ! and ? Tokens
summarize "*" with ? tokens for me	summarize "*" with less than ? tokens for me	summarize "*" with more than ! tokens for me	summarize "*" with between ! and ? tokens for me
Please give me a summary of "*" with ? Tokens	Please give me a summary of "*" with less than ? Tokens	Please give me a summary of "*" with more than ! Tokens	Please give me a summary of "*" with between ! and ? Tokens
I need a summary of length ? for "*"	I need a summary of length smaller than ? for "*"	I need a summary of length greater than ! for "*"	I need a summary of length between ! and ? for "*"
generate a summary for "*" with length ?	I need a summary of length less than ? for "*"	I need a summary of length larger than ! for "*"	Need a summary of "*" with length between ! and ?
Need a summary of "*" with length equal to ?	Need a summary of "*" with length smaller than ?	Need a summary of "*" with length larger than !	write a summary of length between ! and ? for "*"
write a summary of length ? for "*"	summarize the following article with no longer than ? tokens: "*"	summarize the following article with longer than ! tokens: "*"	summarize with length between ! and ?: "*"
summarize with length equal to ?: "*"'	summarize the following article with shorter than ? tokens: "*"	write a summary of length larger than ! for "*"	summarize with between ! and ? tokens:"*"
summarize with exactly ? tokens:"*"	write a summary of length smaller than ? for "*"	summarize with length larger than !: "*"	summarize with ! to ? tokens:"*"
summarize this document with about ? tokens: "*"	summarize with length smaller than ?: "*"	summarize with more than ! tokens:"*"	summarize "*" with ! to ? Tokens
summarize "*" with around ? tokens	summarize with less than ? tokens:"*"	summarize the following article with over ? tokens:"*"	Please summarize "*" with ! to ? Tokens
need a summary of "*" with length ?	summarize "*" within ? tokens	summarize "*" with over ? tokens	summarize following article with ! to ? tokens: "*"

Table 7: Examples of standard control prompts and corresponding augmented prompt templates, where each column shows one type of SCP followed by augmented prompt templates. Where "*" is the placeholder for the input article to be summarized, "!" and "?" are the placeholders for the sampled length values. To build the input examples in training and evaluation datasets, we only need to first replace "!" and "?" with the minimum and maximum target lengths, and then replace "*" with the original article to be summarized.

Model	Setting	CNNDM					NYT				
		R1↑	R2↑	RL↑	B.S.↑	Error↓	R1↑	R2↑	RL↑	B.S.↑	Error↓
GPT-S	Prompt	37.57	15.30	37.74	62.47	11.62	47.48	29.27	42.36	67.86	13.33
	Prompt+RL	37.44	15.02	39.05	62.10	7.81	47.59	29.41	42.66	67.82	11.92
	Prompt+filter	38.20	16.02	37.31	61.96	10.44	48.37	30.83	42.72	67.96	10.30
	Prompt+RL+filter	37.56	15.85	38.47	61.53	**6.22**	48.31	30.94	42.82	67.98	**9.55**
GPT-M	Prompt	38.05	16.15	37.81	62.93	14.31	48.34	30.53	43.11	68.54	5.12
	Prompt+RL	37.73	15.98	38.07	62.62	11.57	48.86	31.19	43.98	69.09	4.47
	Prompt+filter	38.18	16.55	37.14	62.32	12.60	48.53	30.95	43.33	68.55	2.12
	Prompt+RL+filter	37.91	16.33	36.97	62.23	**11.33**	48.76	31.09	43.38	68.80	**1.60**
GPT-L	Prompt	40.27	17.33	39.67	63.96	12.20	49.98	32.43	44.65	69.44	5.89
	Prompt+RL	39.49	16.42	39.02	63.38	9.84	49.12	30.86	43.59	69.03	5.54
	Prompt+filter	39.52	17.33	38.64	63.22	11.57	47.22	31.77	43.29	69.02	5.76
	Prompt+RL+filter	39.75	17.18	38.60	63.15	**8.96**	49.82	31.68	42.48	68.72	**3.29**

Table 12: Comparison of methods in the setting of single-type control instruction, i.e., "`equal to`".

λ	SG					MU				
	R1	R2	RL	B.S.	Error↓	R1	R2	RL	B.S.	Error↓
0.01	36.87	15.17	37.23	62.10	8.93	37.28	15.42	38.55	62.18	15.16
0.03	36.69	14.83	37.06	61.89	8.93	37.81	15.95	38.94	62.39	18.04
0.1	37.36	15.20	37.35	62.29	8.54	36.85	15.24	37.99	61.78	14.38
0.3	37.87	15.52	37.92	62.44	7.97	36.54	15.07	37.76	61.69	14.55
1	37.92	15.83	37.57	62.26	7.78	37.06	15.26	38.00	61.92	14.57
3	38.09	15.96	37.71	62.29	7.95	37.09	15.36	37.78	61.94	15.16

Table 13: The effect of SFT loss. λ is the hyper-parameter discussed in Section 3.4.

placeholder with the real text length of reference summary. Finally, we prepend the SCP before the main context of the augmented input. The results are given in Table 12. Again, we can see that for all settings, the proposed RL method can provide an improvement of length control ability with lower control errors. By further using sample filtering supported by the rule-based reward model, both the basic prompt-based length control model `Prompt+filter` and the one with RL enhancement `Prompt+RL+filter` can achieve lower control errors than not using sample filtering

A.4.2 Effect of SFT loss

As was discussed in Section 3.4, the actor loss involves a term of SFT loss, which is controlled by λ. We conduct an extra experiment on CNNDM by comparing the tuned GPT-S models using different λs for both the case of single and multiple control types. The results are given in Table 13, which shows that a suitable λ is helpful in perserving the performance on downstream task, and the control accuracy will not be largely affected in most cases. Also, the optimal value of λ differs in the cases of SG and MU, thus hyper-parameter tuning is usually needed.

A.5 Learning curves of Standard Prompt Extraction

We provide the learning curves of two types of SPE in Figure 5. For GPT-based extractor, the accuracy is 1 only if the generated SCP exactly matches the label. For BERT-based extractor, we calculate the validation accuracy on a case-by-case basis: If the ground truth SCP type is "none", the accuracy is always 1; if the ground truth SCP type is "more than", we only match the minimum value and check if the minimum value is smaller than maximum value; if the ground truth SCP type is "less than", we only match the maximum value and check if the minimum value is smaller than maximum value; if the ground truth SCP type is "equal to" or "between", we match both of minimum and maximum values. As is shown in Figure 3, both of the SPEs converge well with a validation proportion of matching rate close to 100% in later validation steps. Meanwhile, we find the both BERT and GPT-based extractors performs fairly well on out-of-sample augmented prompts, which demonstrates strong generaliza-

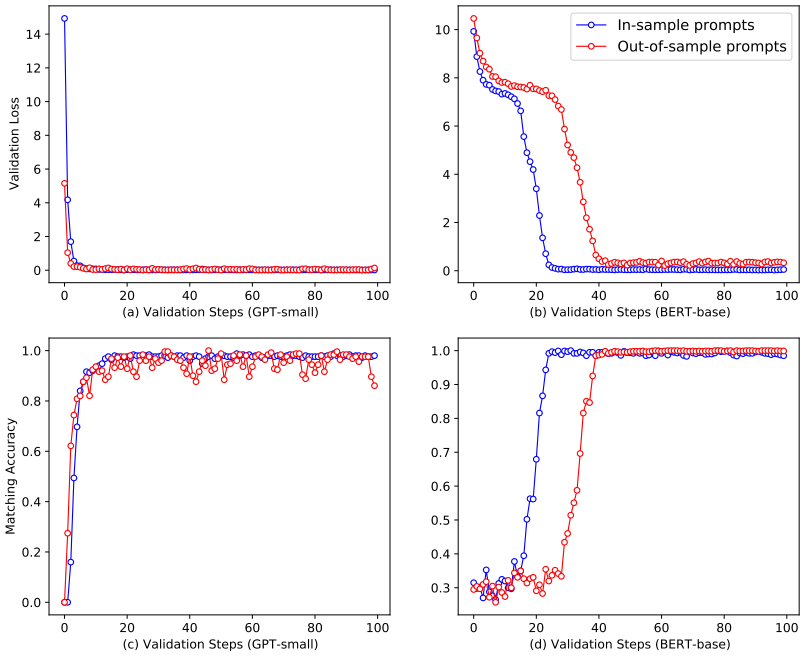

Figure 3: Learning Curves of Standard Prompt Extractors. (a) Validation losses of GPT extractor. (b) Validation losses of BERT extractor. (c) Matching accuracy of GPT extractor. (c) Matching accuracy of BERT extractor. We show the curves of validation cross entropy and matching rate for both cases.

tion ability to new control prompts. For BERT-base, the validation curve and accuracy curve of model on out-of-sample augmented prompts converge slower than in-sample augmented prompts with a right-shift, but the accuracy values in later steps can even surpass that of in-sample validation curve. Notes than we only fine-tune the pre-trained GPT-small and BERT-base from Huggingface, which indicates the noise introduced by the extractors can generally be neglected in practice with same or larger size models.

A.6 Learning curves of Reinforcement Fine-tuning

To analyze the learning behavior, we visualize the learning curves of the policy loss and value loss on training set, control error and BERTscore (F1, in proportion) on validation set for a range of validation step. The results are generated by small GPT-2 model on both NYT and CNNDM for single-type control (with only one control instruction which is "equal to"), which are shown in Figure 4. We can see that as the decrease of policy loss and value loss, the validation reward increases relatively smoothly, while there is no clear decreasing trend of validation BERTscore. The indicates that even with small GPT-2 model, the relevance can be preserved as the control accuracy increase during the RL finetuning.

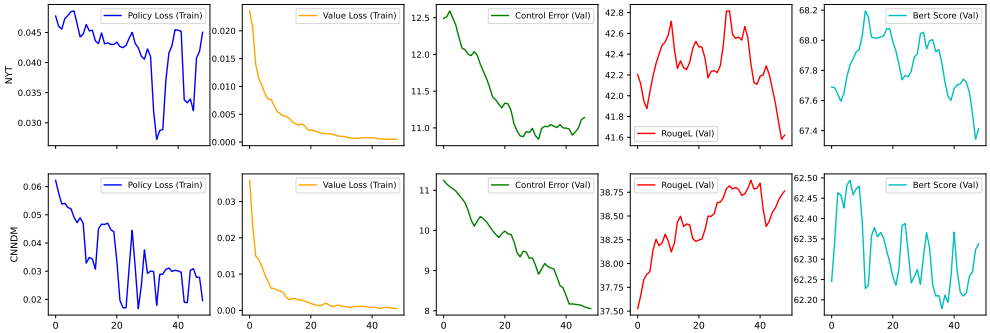

Figure 4: The Diagram of Learning Curves with GPT-S for single-type control instruction (only for “equal to”) without sample filtering.

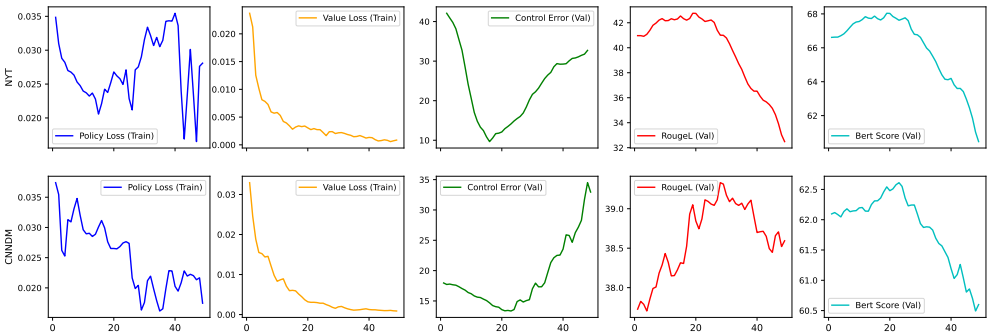

Figure 5: The Diagram of Learning Curves with GPT-S for multi-type control instructions without sample filtering.

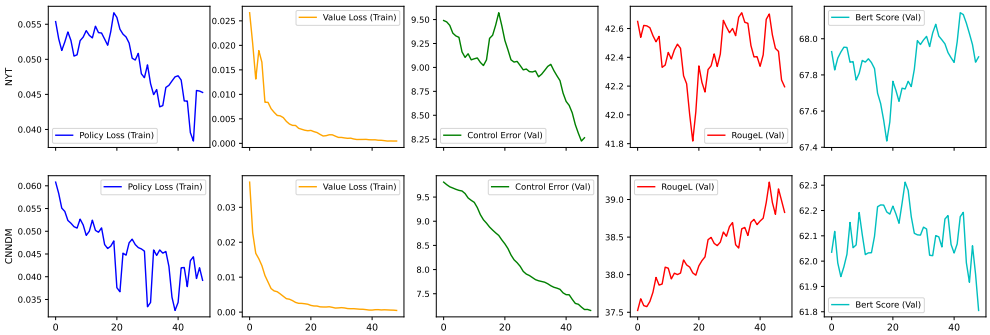

Figure 6: The Diagram of Learning Curves with GPT-S for single-type control instruction (only for “equal to”) with sample filtering.

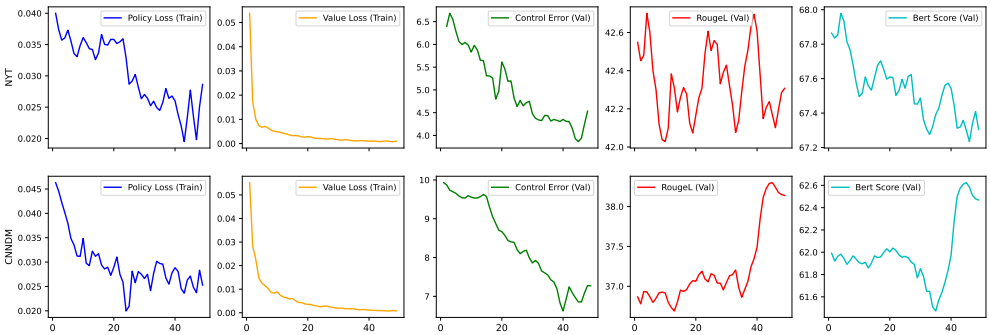

Figure 7: The Diagram of Learning Curves with GPT-S for multi-type control instructions with sample filtering.