Tailor: A Soft-Prompt-Based Approach to Attribute-Based Controlled Text Generation

Anonymous ACL submission

Abstract

Attribute-based Controlled Text Generation (CTG) refers to generating sentences that satisfy desirable attributes (*e.g.*, emotions and topics). Existing works usually utilize fine-tuning or resort to extra attribute classifiers, yet suffer from increases in storage and inference time. To address these concerns, we explore attribute-based CTG in a parameter-efficient manner. In short, the proposed **Tailor** represents each attribute as a pre-trained continuous vector (*i.e.*, single-attribute prompt), which guides the generation of a fixed pre-trained language model (PLM) to satisfy a pre-specified attribute. These prompts can be simply concatenated as a whole for multi-attribute CTG without any re-training. Nevertheless, this may raise problems of fluency downgrading and position sensitivity. To solve this, Tailor provides two solutions to enhance the combination. The former contains a multi-attribute prompt mask and a re-indexing position sequence to bridge the gap between the training (one single-attribute prompt for each task) and the testing stage (concatenating two prompts). The latter introduces a trainable prompt connector to further enhance the combinations. Experiments demonstrate that, only requiring 0.08% extra training parameters of the GPT-2, Tailor can achieve effective and general improvements on eleven attribute-specific generation tasks.

1 Introduction

Attribute-based CTG (Zhang et al., 2022) focuses on generating sentences satisfying pre-specified attributes such as topic and sentiment, which remains extremely challenging in recent progress (Dathathri et al., 2020). Specifically, single-attribute CTG typically resorts to attribute-specific data, guiding the CTG model learning with supervised objectives (Keskar et al., 2019; Lyu et al., 2021; Ziegler et al., 2019). Nevertheless, multi-attribute CTG is generally zero-shot since no example of a sentence with specified attribute combination is accessible during training (Lample et al., 2019).

For both single and multi-attribute CTG, existing efforts can be roughly divided into two types: 1) fine-tuning a pre-trained language model (PLM) on the attribute-specific data (Ziegler et al., 2019) and 2) utilizing extra attribute classifiers. The former usually introduces control codes to generate various styles of sentences with one PLM, such as keywords (Keskar et al., 2019) and numerical sequence (Lyu et al., 2021). The latter applies extra attribute classifiers to guide a PLM, such as back-propagating gradients of these classifiers (Dathathri et al., 2020) or weighting output logits (Krause et al., 2021; Yang and Klein, 2021). However, this two types suffer from expensively re-training whole PLM (Yang and Klein, 2021) and higher latency during inference (Qian et al., 2022), respectively.

To overcome the aforementioned limitations, we propose **Tailor** – **T**ext-**a**ttri**b**ute gener**a**l contr**o**ll**e**r, a soft-prompt-based approach to jointly include both single-attribute CTG and multi-attribute CTG in a unified manner. The key idea is to represent each attribute as a trainable continuous vector (*i.e.*, the single-attribute prompt). These single-attribute prompts could be separately used or concatenated as a whole to control a fixed GPT-2 (Radford et al., 2019) for single and multi-attribute CTG, respectively.[1] As simply concatenating always suffers from poor performances (see Appendix F), Tailor provides two effectively concatenating strategies without or with training after single-attribute CTG, namely non-training and training methods. First of all, we argue that the undesirable results of simply concatenating is due to the gap between the training and the testing stage. Specifically, the single-attribute prompt only attends to itself while being individually trained by the attribute-specific data. While testing, the second prompt also attends

[1] Following Lample et al. (2019); Qian et al. (2022), we focus on the multi-attribute task that contains two attributes.

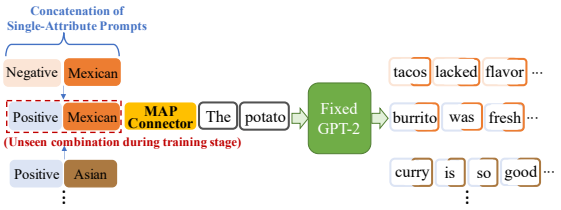

Figure 1: MAP connector concatenates single-attribute prompts and a pre-specified input prefix to the fixed GPT-2 for multi-attribute CTG, even is effective to the unseen combination (e.g, the combination of Positive sentiment and topic of Mexican food is not accessible to MAP connector during training).

to the first one in the concatenation, with the simultaneous change of the position embeddings. To fill this gap, the non-training method introduces a **M**ulti-**A**ttribute **P**rompt mask (**MAP** mask) and a **R**e-indexing **P**osition sequence (**RP** sequence) for the fixed GPT-2. MAP mask prevents distinct single-attribute prompts from cross-attention, and RP sequence ensures stable position information for the PLM after swapping, by individually numbering each prompt.

Such a non-training method could be easily implemented and gets promising performances, but still has much space for improvement – there is no multi-attribute specific training stage for these prompts to adapt to work together. Therefore, the training method contains a trainable prompt to connect two single-attribute prompts as a whole to multi-attribute CTG. Inspired by the role of 'and' in connecting parallel phrases for natural sentences (Rudolph, 1989), as shown in Figure 1, the proposed **M**ulti-**A**ttribute **P**rompt connector (**MAP** connector) can be concatenated with any two singe-attribute prompts and hints a GPT-2 to multi-attribute CTG. Meanwhile, a pseudo-prompt based strategy is also provided for training the connector in unsupervised settings. With MAP connector, the combinations show strong performances on multi-attribute CTG on the popular benchmark YELP dataset (Lample et al., 2019). Furthermore, MAP connector can get encouraging improvements for the unseen combinations in the training stage (see Appendix F). The main contributions are:

- We propose **Tailor**, a soft-prompt-based approach to attribute-based CTG. To jointly include both single-attribute and multi-attribute CTG in a unified paradigm, Tailor employs a set of pre-trained prefixes to guide a fixed PLM to generate sentences with pre-specified attributes, and effectively concatenate them to generate multi-attribute sentences.
- We experimentally reveal the combining ability of continuous prompts. To enhance this combination, we explore two effective strategies without training (MAP mask + RP sequence) or with training (MAP connector) after single-attribute CTG. Especially, the MAP connector achieves strong performances on six multi-attribute generation tasks, and even works on the unseen ones.

2 Related Work

Attribute-Based CTG focuses on generating sentences containing pre-specified attributes, such as sentiment and topic. As a vital demand for intelligent writing (Zhang et al., 2022), existing efforts include fine-tuning PLMs and utilizing extra attribute classifiers. The first type usually fine-tunes separately and stores a full copy of PLM for each desirable attribute (Ziegler et al., 2019). To alleviate the storage problem, CTRL (Keskar et al., 2019) provides 55 kinds of control codes (*i.e.*, special keywords) to fine-tune one PLM for generating sentences of various styles. StylePTB (Lyu et al., 2021) also proposes several style transfer tokens (*i.e.*, a sequence of numbers) to guide a GPT-2 (Radford et al., 2019) to multiple styles transfer. GSum (Dou et al., 2021) introduces four guidance signals (*e.g.*, keywords and relations) to enhance the controllability of PLMs in text summarization. Although they make successful attempts in attribute-based CTG, re-training whole PLMs could be expensive (Yang and Klein, 2021). To improve the flexibility and extensibility of the CTG model, the second type makes efforts in the inference stage. In short, utilizing extra attribute classifiers to guide PLMs in each generating step. PPLM (Dathathri et al., 2020) iteratively modifies latent representations of a GPT-2 referring to the gradient of attribute classifiers, yet notably increasing the inference time. To solve this problem, Fudge (Yang and Klein, 2021) uses an attribute predictor to adjust the output probabilities of a PLM. Similarly, GeDi (Krause et al., 2021) uses smaller PLMs as generative discriminators to hint a larger PLM generating sentences that satisfy desirable attributes. Despite their progress, the fluency of generating sentences tends to decrease compared with the original PLM (see § 4.2) and extra infer-

ence time costs still existed. In comparison, utilizing Tailor, PLMs can benefit from the manner of controllability on single-attribute prompt combinations, with a negligible decrease on text quality.

Prompt Learning is a new paradigm in NLP summarised as "Pre-train, Prompt and Predict" (Liu et al., 2021a). In short, it guides a single PLM to solve various downstream tasks by reformulating these tasks into a text-to-text manner. Recently, the continuous prompt has attracted attention (Gu et al., 2021; Liu et al., 2021b, 2022), which usually forms as a set of continuous task-specific vectors to the input. Despite their encouraging progress, the prompt composition is rarely explored but undoubtedly important in prompt learning. In that case, a composable task could be accomplished by composing various subtasks with multiple sub-prompts (Liu et al., 2021a). To achieve it, PTR (Han et al., 2021) introduces manual sub-prompts for entity recognition and relation classification, respectively. Then, these two kinds of prompts are composed by logic rules as a complete prompt for the relation extraction task. Unfortunately, the composition of continuous prompts is rarely explored yet has demonstrated great potential (Qian et al., 2022). The main difference between contrastive prefix Qian et al. (2022) and Tailor is that the former needs attribute data to be occurred contrastively (e.g, positive and negative attribute data must be available at the same time), which might be limited for the single attribute. For multi-attribute, contrastive prefix trains a new prompt (twice the size of their single prompt) for each combination. Instead of it, Tailor only trains an extra prompt connector to enhance the combinations of single prompts. It can act as an efficient plug-and-play manner with extremely low training parameters to attribute-based CTG.

3 Methodology

3.1 Tailor for Single-Attribute CTG

Different from fine-tuning a full copy of PLMs for each attribute, our basic idea is to guide the generation of a PLM with a set of pre-trained continuous vectors, namely single-attribute prompts. Meanwhile, each prompt represents a desirable attribute. As shown in Figure 2 (top), we fix the parameters of a GPT-2 and train each prompt on the attribute-specific data. After training, these prompts can act as plug-ins for desirable single-attribute CTG. For the prefix "Once upon a time", the GPT-2 can continue with "I had to order my tacos ..." with a prompt representing the Mexican food topic or " the food was good" with a prompt representing the positive sentiment. In this way, our method can be easily expanded: if a new attribute emerges, we only need to train an attribute prompt and then control a PLM to generate attribute-specific sentences.

To be exact, we use language modeling learning object to train such a set of single-attribute prompts. In detail, k-th single-attribute prompt S_k with length l_k is first initialized randomly, where $S_k \in \mathbb{R}^{l_k \times d_{emb}}$. d_{emb} is the word embedding dimension of the GPT-2. Meanwhile, given an attribute-specific sentence $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$ with length n, we get a word sequence matrix $X_{emb} \in \mathbb{R}^{n \times d_{emb}}$ after being embedded by GPT-2. Then, S_k is concatenated with X_{emb} to form a input matrix as $[S_k; X_{emb}] \in \mathbb{R}^{(l_k+n) \times d_{emb}}$, and this matrix is fed into a fixed GPT-2. Finally, the language-modeling based learning object is:

$$\mathcal{L}_{single} = \sum_{t=1}^{n} \log P_{\theta_g;\theta_{S_k}} \left(x_t | S_k, x_{<t}\right), \quad (1)$$

where θ_g and θ_{S_k} denote the parameters of GPT-2 and the single-attribute prompt, respectively. Only θ_{S_k} are updated during the training stage.

3.2 Tailor for Multi-Attribute CTG

Inspired by the composition of discrete prompts (Han et al., 2021) to accomplish a complex task, our intuitive idea is to combine single-attribute prompts as a multi-attribute prompt to hint a PLM for multi-attribute CTG. To enjoy the benefit of our paradigm in single-attribute CTG, we first consider simply concatenating several single-attribute prompts as a whole multi-attribute prompt. Surprisingly, such a multi-attribute prompt can guide a GPT-2 to generate sentences containing multi attributes and get encouraging performances in unsupervised settings without any training (see § 4.2). Despite the progress, this straightforward method suffers from fluency decrease compared with single-attribute CTG. Meanwhile, it is position sensitive, *i.e.*, the PLM tends to focus more on the single-attribute prompt that is closer to the input prefix (see Appendix F).

To polish such a paradigm while keeping plug-and-play and storage-friendly advantages, as shown in Figure 2 (bottom), Tailor introduces a non-training method to quickly and effectively alleviate the above problems of simply concatenation. Afterward, a training method is further provided to

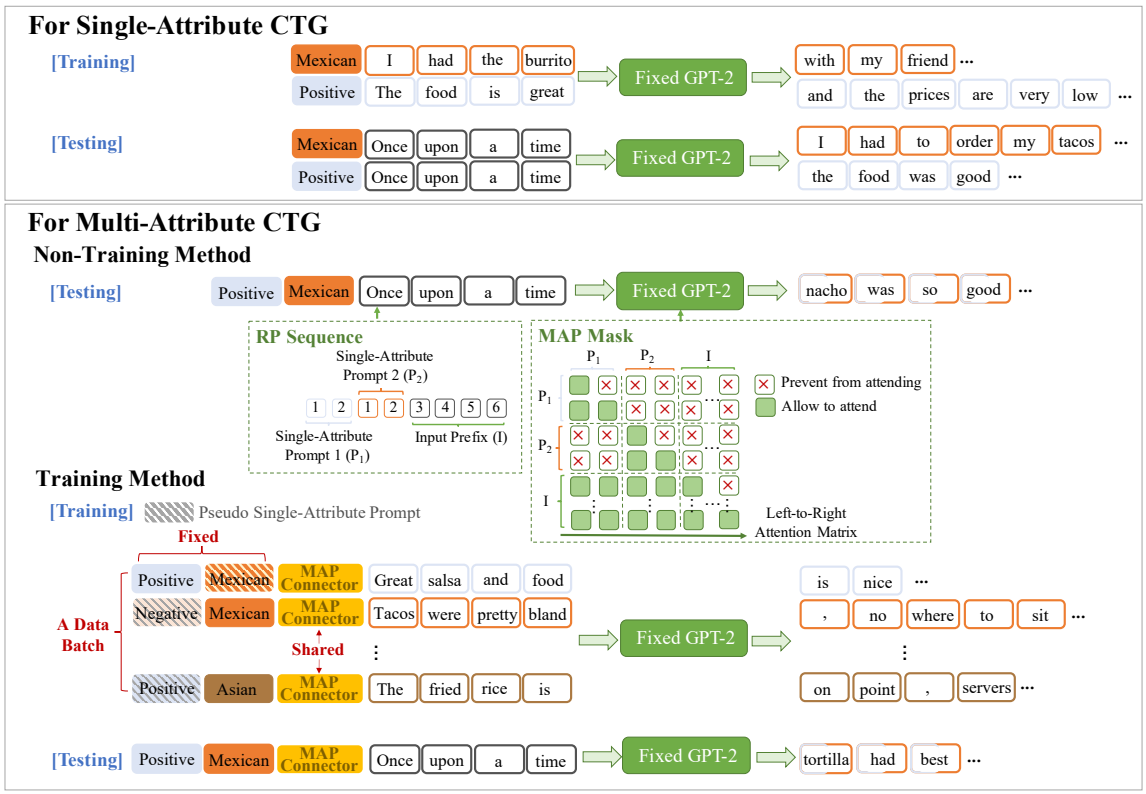

Figure 2: The overview of Tailor to attribute-based CTG. We use 2-token-sized single-attribute prompts for illustration. Notably, the different colored text boxes denote different attribute-specific sentences. For multi-attribute sentences, we use bi-colored text boxes to highlight them.

greatly enhance the combinations. We elaborate the two methods separately as follows.

3.2.1 Non-Training Method

To make better use of single-attribute prompts, reducing disparities between the training (a single-attribute prompt for each task) and the testing stage (concatenating more than one single-attribute prompt) is undoubtedly important. Specifically, the single-attribute prompt only attends to itself in the attention matrix while training, as each prompt is individually trained by the attribute-specific data. However, while in the testing stage for multi-attribute CTG, the second prompt also focuses on the first one in the concatenation, with the simultaneous change of the position embedding. To fill this gap, MAP mask and RP sequence are introduced to the fixed PLM while generating. MAP mask avoids cross-attention between representations of single-attribute prompts to approximate the condition in the single-attribute CTG training stage. Meanwhile, the RP sequence keeps a stable prompt position for swapping, preventing such concatenating paradigm from position sensitivity.

MAP Mask For the ease of implementation, we introduce MAP mask matrix M_p to the softmax logits of GPT-2. Given a vanilla attention module:

$$A = \mathrm{Softmax}(\frac{QK^\top}{\sqrt{d}}) \in \mathbb{R}^{n \times n}, \tag{2}$$

where n is the length of input sentence $\boldsymbol{x}$ and Q,K denote representations of query and key, respectively.[2] For MAP Mask, given two single-attribute prompts S_u, S_v with length being l_u, l_v, respectively, the attention module is then modified as:

$$A = \mathrm{Softmax}(\frac{QK^\top}{\sqrt{d}} + M_p) \in \mathbb{R}^{(l_p+n)\times(l_p+n)},$$
$$M_p^{ij} = \begin{cases} -\infty & i \in [l_u, l_v] \text{ and } j \in [0, l_u], \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

where $l_p = l_u + l_v$.

RP Sequence Simple concatenation of single-attribute prompts always suffers from position sensitivity. To address this issue, we propose a simple but effective method to ensure position consistency

[2]The multi-head mechanism is omitted for illustration.

while swapping. In short, we modify the position sequence of the PLM while concatenating.[3] Given the original position sequence:

$$\boldsymbol{id} = \{ \underbrace{1, ..., l_u}_{\text{Length of } S_u}, \underbrace{l_u + 1, ..., l_p}_{\text{Length of } S_v}, \underbrace{l_p + 1, ..., l_p + n}_{\text{Length of input prefix}} \}, \tag{4}$$

the RP sequence can be defined as:

$$\boldsymbol{id}_{\text{RP}} = \{ \underbrace{1, ..., l_u}_{\text{Length of } S_u}, \underbrace{1, ..., l_v}_{\text{Length of } S_v}, \underbrace{l_v + 1, ..., l_v + n}_{\text{Length of input prefix}} \}, \tag{5}$$

note that, $l_v = l_u$. In that case, swapping does not bring any changes, since the position of prompts is fixed by the RP sequence while avoiding cross-attention by the MAP mask.

3.2.2 Training Method

While the non-training method partly addresses the issues of combination, there is no multi-attribute specific training stage for these prompts to adapt to work together. Therefore, we provide a training method – MAP connector, which is also a continuous prompt trained for combining two single-attribute prompts to multi-attribute CTG. To utilize only single-attribute sentences for multi-attribute CTG, we propose a pseudo-attribute prompt based training strategy for MAP connector. The details of the pseudo-attribute prompt building method and the workflow of the MAP connector are as follows.

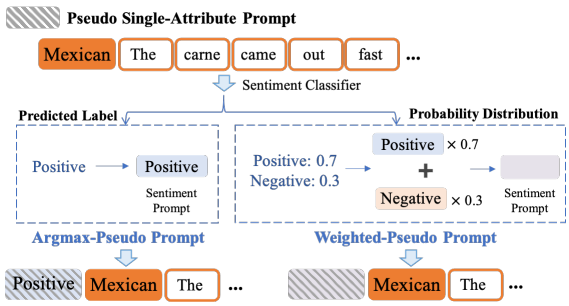

Figure 3: Building pseudo-attribute prompt.

Building Pseudo Single-Attribute Prompt Our key idea is to build another pseudo-attribute prompt for each single-attribute sentence, thus MAP connector could be trained in a multi-attribute circumstance. An overview of our building method is demonstrated in Figure 3, where a sentence with the topic of Mexican food is used as a showcase.[4]

[3] In this case, position sequence denotes position indexes of input tokens in the position embeddings for GPT-2.

[4] In the implementation for multi-attribute CTG, we use YELP data of two sentiments attributes (Positive / Negative) and three kinds of food type (Mexican / American / Asian)

To be exact, we first train an attribute classifier on the same single-attribute CTG training set. Thus, such a classifier with n_{class} classes corresponds to the pre-trained single-attribute prompt set $\boldsymbol{S} = \{S_1, S_2, ..., S_{n_{class}}\}$. Given an attribute-specific sentence $\boldsymbol{x}$ of other attribute category, we first get the class probabilities set $\boldsymbol{p} = \{p_1, p_2, ..., p_{n_{class}}\}$. Then, the pseudo single-attribute prompt can be obtained by two methods:

$$\begin{aligned} S_a &= S_{\text{Index}(\arg\max(\boldsymbol{p}))}, \\ S_w &= \sum_{z=1}^{n_{class}} p_z S_z, \end{aligned} \tag{6}$$

where argmax-pseudo prompt method obtains the pseudo prompt S_a by using a single-attribute prompt corresponding to the predicted sentiment, Index$(\cdot)$ means getting the corresponding index. In contrast, weighted-pseudo prompt method utilizes the predicted probability distribution to multiply corresponding single-attribute prompts, respectively. Then these weighted prompts form a whole prompt S_w by element-wise addition.

The MAP Connector Workflow Figure 2 bottom illustrates the workflow of the MAP connector. In the training stage, we unify sentences containing different single attributes to train the MAP connector, each of which is added an extra pseudo single-attribute prompt (boxes with the slash pattern) by employing the aforementioned method. Specifically, for each training sample, we first concatenate two single-attribute prompts (real and pseudo), the MAP connector and the input sentence into a sequence, and then feed it into a fixed GPT-2. It is worth noting that only the parameters of the MAP connector are updated in the training stage. Therefore, given two single-attribute prompt S_u and S_v, MAP connector C with the length l_C, $C \in \mathbb{R}^{l_C \times d_{emb}}$, we concatenate S_u, S_v, C and the input sentence matrix X_{emb} to form a input matrix as $[S_u; S_v; C; X_{emb}]$. The learning object is:

$$\mathcal{L}_{multi} = \sum_{t=1}^{n} \log P_\theta \left(x_t | S_u, S_v, C, x_{<t} \right), \tag{7}$$

where $\theta = [\theta_g; \theta_{S_u}; \theta_{S_v}; \theta_C]$. θ_g, θ_{S_u}, θ_{S_v}, and θ_C denote the parameters of GPT-2, two single-attribute prompts and MAP connector, respectively. Only θ_C are updated during the training stage. In the inference stage, we just decompose each multi-attribute generation task as several single-attribute

generation tasks and find corresponding single-attribute prompts. Then, these prompts are concatenated with MAP connector to generate sentences that satisfy multi attributes.

4 Experiments

4.1 Experimental Setup

Datasets We conduct experiments on the widely-used benchmark dataset YELP (Lample et al., 2019). It contains multiple single-attribute data that can verify Tailor's performance on both single-attribute and multi-attribute CTG, while ensuring that the combination of these attributes is reasonable. Following previous works that conduct experiments on attributes of emotions and topics for multi-attribute CTG, we choose Yelp restaurants reviews of sentiment attributes (positive (PO) and negative (NE)) and topics of food type (Mexican (ME), American (AM) and Asian (AS) foods) to evaluate models. Specifically, each attribute contains 30,000 / 3,000 sentences for training / validation. For evaluation, to keep in line with previous works (Yang and Klein, 2021; Dathathri et al., 2020), we use 15 attribute-unrelated prefixes[5] and ask the model to continue writing with them (for each of the 15 prefixes, 100 completions are generated, total: 1500 for each attribute) while satisfying pre-specified attribute as the final results.[6]

Automatic Evaluation Following Yang and Klein (2021); Dathathri et al. (2020), we automatically evaluate generation results from three aspects: (1) **Correctness**. We used RoBERTa$_{\text{Large}}$ (Liu et al., 2019) based attribute classifiers to compute the fraction of final sentences that contain a pre-specified attribute, details in Appendix C. (2) **Text Quality**. Grammar (GRAM) (Warstadt et al., 2019) indicates the averaged grammaticality probabilities of all final sentences, evaluated by a RoBERTa-based CoLA grammaticality model (Yang and Klein, 2021). Perplexity (PPL), we average the scores from GPT-2$_{\text{Base}}$, GPT-2$_{\text{Medium}}$ and GPT-2$_{\text{Large}}$ version of GPT-2 (Radford et al., 2019) as the final result. (3) **Diversity**. Following Li et al. (2015), we report the distinctness of the final results. Specifically, we count the number of unigrams, bigrams and trigrams and then normalize them by the total number of words (*i.e.*, Dist-1 / Dist-2 / Dist-3).

Human Evaluation Following Qian et al. (2022), we also conduct the human evaluation. For each model, three crowdsource evaluators are shown 15 randomly selected samples (one per each attribute-unrelated prefixes) for each generation task (Total: 75 samples for single-attribute CTG and 90 samples for multi-attribute CTG), respectively. Then, they are asked to rate model results in two categories: the text **quality** of generation sentences and whether they contain the target **attribute**. Scores are ranged from 1 to 5, the higher the better.[7]

Tailor Settings Tailor$_{\text{Single}}$ denotes the single-attribute prompts. For multi-attribute, Concat$_{\text{Simple}}$ means simply concatenating two single-attribute prompts and Tailor$_{\text{Concat}}$ is our non-training method. Tailor$_{\text{Argmax}}$ and Tailor$_{\text{Weight}}$ represent using argmax-pseudo and weighted-pseudo prompts when training the MAP connector, respectively.

Baselines We compare Tailor with mainstream competitive models as follows. (1) Finetune, fine-tuning the original GPT-2$_{\text{Base}}$ on attribute-specific data. As multi-attribute CTG is unsupervised, following Lyu et al. (2021), we sequentially apply the GPT-2 trained for corresponding single-attribute data multiple times. (2) Adapter, following Li and Liang (2021), we use the adapter for GPT-2 as same as Lin et al. (2020). Note that for multi-attribute CTG, we first use the same training method as mentioned in Finetune for Adapter. Besides, we use the same argmax-pseudo labeled sentences (see § 3.2.2) to train the Adapter (marked with 'Pseudo'). (3) GeDi (Krause et al., 2021), using small PLMs to hint large ones. (4) PPLM (Dathathri et al., 2020), back-propagating gradients of extra attribute classifiers to a PLM[8].

4.2 Main Results

Single-Attribute CTG As shown in Table 1, Tailor$_{\text{Single}}$ outperforms PPLM and GeDi to a great extent on both correctness and text quality. Meanwhile, compared with other parameter-efficient learning model Adapter, Tailor$_{\text{Single}}$ also gets improvements on both correctness (e.g, + 9.19% of Food) and diversity (e.g, + 0.02% / + 0.12% / + 0.25% of Food) with a similar scale of training parameters. However, with 0.08% training parameters of the GPT-2, Tailor$_{\text{Single}}$ still has a performance gap with Finetune, *e.g.*, - 4.14% correctness on Food. Fortunately, as the length of Tailor$_{\text{Single}}$ increases (see Appendix F), this gap appears to narrow (- 0.33%, Tailor$_{\text{Single}}$ with the

[5] https://github.com/uber-research/PPLM

[6] More details can be found in Appendix B

[7] Details can be found in Appendix D.

[8] The implementation details of baselines and Tailor can be found in Appendix A

Method	Trained Params	Correctness	Text Quality		Diversity
	(%)	(%) ↑	GRAM ↑	PPL ↓	Dist-1/Dist-2/Dist-3 ↑
Finetune (Food) (2019)	100.000	87.53	0.78	40.60	0.04 / 0.22 / 0.42
Finetune (Sent) (2019)	100.000	97.95	0.76	42.83	0.04 / 0.21 / 0.41
GeDi (Food) (2021)	100.000	99.82	0.28	278.22	0.42 / 0.79 / 0.95
GeDi (Sent) (2021)	100.000	87.37	0.32	517.87	0.27 / 0.85 / 0.97
Adapter (Food) (2020)	0.100	74.70	**0.75**	**43.85**	0.04 / 0.23 / 0.46
Adapter (Sent) (2020)	0.100	93.32	**0.74**	47.01	0.04 / 0.22 / 0.45
PPLM (Food) (2020)	0.001	60.64	0.34	105.33	**0.16 / 0.53 / 0.80**
PPLM (Sent) (2020)	0.001	69.37	0.36	75.59	**0.15 / 0.53 / 0.82**
$\text{Tailor}_{\text{Single}}$ (Food)	0.080	**83.89**	0.71	45.79	0.05 / 0.35 / 0.71
$\text{Tailor}_{\text{Single}}$ (Sent)	0.080	**93.80**	0.71	**46.20**	0.06 / 0.35 / 0.70

Table 1: The main results of single-attribute CTG. Sent and Food: averaging the evaluation scores of all sentiment attributes and topics of food type, respectively. Trained Params: the ratio of the number of trainable parameters to the method Finetune. Correctness: the fraction of attribute-related sentences. GRAM: averaged grammaticality probabilities. PPL: perplexity scores. Dist-1/2/3: the distinctness of the final results (unigrams, bigrams and trigrams). ↑ means a higher score is better whereas ↓ is exactly the opposite. Bold values represent the maximum values of each sub-task in parameter-efficient method.

Method	Correctness (%)	Text Quality	Diversity
	Avg. ↑ / Sent ↑ / Food ↑	GRAM ↑ / PPL ↓	Dist-1/Dist-2/Dist-3↑
Finetune	69.80 / 74.03 / 65.57	0.69 / 46.54	0.04 / 0.23 / 0.42
Adapter	69.10 / 74.10 / 64.10	**0.77 / 37.89**	0.03 / 0.21 / 0.42
Adapter (Pseudo)	81.71 / 89.95 / 73.46	0.75 / 45.63	0.04 / 0.22 / 0.45
$\text{Concat}_{\text{Simple}}$	76.20 / 87.88 / 64.51	0.63 / 55.02	0.05 / 0.33 / 0.68
$\text{Tailor}_{\text{Concat}}$	78.82 / 87.54 / 70.10	0.63 / 52.76	0.05 / 0.32 / 0.68
$\text{Tailor}_{\text{Weight}}$	83.98 / **93.27** / 74.68	0.68 / 51.41	**0.05 / 0.33 / 0.69**
$\text{Tailor}_{\text{Argmax}}$	**87.15** / 92.97 / **81.32**	0.69 / 52.73	**0.05 / 0.33 / 0.69**

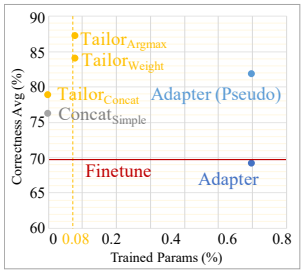

Table 2: The main results of multi-attribute CTG. We average the scores of six combinations (two sentiment attributes × three topic attributes of food type) as the final results. Adapter (Pseudo): using our argmax-pseudo labeled sentences to train the Adapter. $\text{Concat}_{\text{Simple}}$: simply concatenating two single-attribute prompts. Tailor-C: our non-training method. $\text{Tailor}_{\text{Argmax}}$ and $\text{Tailor}_{\text{Weight}}$: using argmax-pseudo and weighted-pseudo prompts in the training stage of the MAP connector, respectively. For better view, Performances vs. Trained Params are as shown on the right figure. $\text{Tailor}_{\text{Argmax}}$ gets the highest score with only 0.08% trained parameters of Finetune.

Method	Quality ↑	Attribute↑	All ↑
`Single-Attribute CTG`			
Finetune	4.69	2.97	**7.66**
Adapter	4.66	2.64	7.30
PPLM	2.40	1.19	3.59
$\text{Tailor}_{\text{Single}}$	4.62	3.04	**7.66**
`Multi-Attribute CTG`			
Finetune	4.67	1.74	6.41
Adapter (Pseudo)	4.79	1.91	6.70
$\text{Tailor}_{\text{Argmax}}$	4.57	2.37	**6.94**

Table 3: Results of human evaluation.

prompt length of 256). Then, we illustrate human evaluations in Table 3. Different from automatic evaluations, $\text{Tailor}_{\text{Single}}$ obtains the same score with Finetune, even outperforms all baselines on attribute-relevance score. This experimental discovery demonstrate the limitations of only resorting to automatic evaluation, as also be mentioned in Welbl et al. (2021); Qian et al. (2022).

Multi-Attribute CTG As shown in Table 2, we compare three instantiations of Tailor and strong baselines in the single-attribute CTG experiment. First, $\text{Tailor}_{\text{Concat}}$ shows encouraging performances without any training, especially on correctness, outperforms fine-tuning (+ 13.51% Sentiment / + 4.53% Food) and Adapter (+ 13.44% Sentiment / + 6.00% Food). Besides, our training methods $\text{Tailor}_{\text{Weight}}$ and $\text{Tailor}_{\text{Argmax}}$ show improvements on all scores compared with $\text{Tailor}_{\text{Concat}}$, *e.g.*, + 4.58% / + 11.22% correctness on the topic of food type attribute. Meanwhile, Tailor also outperforms Adapter with the same pseudo label strategy on both correctness and diversity, with a notable scale discrepancy of training parameters (0.08% vs 0.60%, *i.e.*, 1:7.5). Meanwhile, Tailor seems to suffer from lower text diversity compared to PPLM and GeDi. This is because these methods have poor

fluency (with many unreasonable words), while Dist-1/2/3 measure different words without considering whether they are reasonable. We supplement it by human evaluation. As shown in Table 3, the diversity of words considered by the index attribute, which shows the superiority of our method.

TP (%)	Method	Correct (%)		
		Avg. ↑	Sent↑	Food↑
	`Single-Attribute CTG`			
100.00	Finetune	54.08	-	54.08
100.00	Finetune	85.28	85.28	-
0.10	Adapter	55.79	-	55.79
0.10	Adapter	77.91	77.91	-
0.08	Tailor$_{\text{Single}}$	**66.23**	-	66.23
0.08	Tailor$_{\text{Single}}$	**89.27**	89.27	-
	`Multi-Attribute CTG`			
100.00	Finetune	60.60	73.45	47.75
0.60	Adapter	57.15	68.44	45.85
0.60	Adapter (Pseudo)	67.27	78.66	55.88
0.00	Tailor$_{\text{Concat}}$	68.09	74.38	61.79
0.08	Tailor$_{\text{Weight}}$	70.32	84.18	56.46
0.08	Tailor$_{\text{Argmax}}$	**71.41**	83.63	59.18

Table 4: The main results of few-shot learning. Note that TP for multi-Attribute CTG means the extra training parameters as the percentage of Finetune after single-Attribute CTG. We average the scores of six combinations as the final results.

4.3 Further Discussions

Few-Shot Learning We conduct a few-shot learning setting to further analyze the effectiveness of Tailor. In detail, following Li and Liang (2021), we randomly sample from full dataset and obtain the few-shot dataset (training / validation / testing: 150 / 20 / 20 samples). Specifically, we sample three different few-shot datasets and average the scores of each method on three datasets as the final results. As shown in Table 4, three types of Tailor outperform other baselines on correctness, with 0.00% / 0.08% extra training parameters of Finetune.

Cross Domain Dataset Evaluating We further evaluate the performances of Tailor on combining attribute from different domain.[9] Specifically, we choose SST-2 (Socher et al., 2013) and AG News (Zhang et al., 2015) for data sources of sentiment and topic attribute, respectively. As shown in Appendix Table 10, Tailor still outperforms baselines in both correctness and diversity. Meanwhile, the text quality of Tailor has been improved by the Map Connector training (GRAM 0.59 to 0.68).

[9]The details can be found in Appendix E

Inference Speed We also compare Tailor with extra classifier based CTG method on inference speed. As shown in Table 5, Tailor$_{\text{Single}}$ outperforms baselines to a great extend on inference speed, which indicates computational efficacy of Tailor.

Methods	Inference Speed ↓
Tailor$_{\text{Single}}$	**0.758** (1.00 ×)
GeDi	1.680 (0.45 ×)
PPLM	15.553 (0.05 ×)

Table 5: Inference speed comparisons (second/sample).

Method	Correctness (%)		
	Avg. ↑	Sent ↑	Food ↑
Tailor$_{\text{Concat}}$	**78.82**	87.54	70.10
– MAP Mask	78.36	87.39	69.34
– RP Sequence	77.77	88.33	67.21
– Both	76.20	87.88	64.52

Table 6: The ablation study on using the MAP mask and the RP sequence of Tailor$_{\text{Concat}}$. '–' denotes removing the corresponding module from Tailor$_{\text{Concat}}$. Note that, exchanging the concatenating order of prompts would bring different performances, except for Tailor$_{\text{Concat}}$. Thus, we average the scores from these two situations of six attributes combinations.

Ablations of Tailor$_{\text{Concat}}$ Whether Tailor$_{\text{Concat}}$ enjoys the benefits from the MAP mask and the RP sequence is of concern. As shown in Table 6, both the MAP mask and the RP sequence are important to Tailor$_{\text{Concat}}$. More importantly, using these two strategies simultaneously can improve the performance while avoiding the position sensitivity.

5 Conclusions

In this paper, we explore attribute-based CTG in a soft-prompt-based manner—Tailor, which represents each attribute as a continuous prompt and effectively combines them as a multi-attribute prompt. For enhancing these combinations, Tailor provides two solutions, namely non-training (MAP mask + RP sequence) and training methods (MAP connector). As our first attempt to multi-attribute CTG, combining more than two attributes still needs to be discussed. In the future, we will investigate extending Tailor to connect wider ranges of attributes, and expand it to other text-to-text generation tasks.

Limitations

As we tentatively give a successful implementation of leveraging soft-prompt-based manner to benefit both single and multi-attribute CTG, such a paradigm deserves a closer and more detailed exploration. First, we explore multi-attribute CTG in the scenario of two-attribute composition, yet combining more attributes when generating a completion is more challenging and thrilling, and still in its fledgeless stage. Besides, while extensive experiments demonstrate that Tailor consistently improves attribute-based CTG, applying our approach on a wider variety of PLMs will evaluate the effectiveness of Tailor in a more generally way.

Ethics Statement

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors. Informed consent was obtained from all individual participants included in the study.

References

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *ICLR 2020*. OpenReview.net.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. Gsum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4830–4842. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. PPT: pre-trained prompt tuning for few-shot learning. *CoRR*, abs/2109.04332.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL 2017*, pages 427–431. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Findings of EMNLP 2021*, pages 4929–4952. Association for Computational Linguistics.

Guillaume Lample, Sandeep Subramanian, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *ICLR 2019*. OpenReview.net.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

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

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of EMNLP 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 441–459. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *ACL 2022*.

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

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

Yiwei Lyu, Paul Pu Liang, Hai Pham, Eduard H. Hovy, Barnabás Póczos, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2021. Styleptb: A compositional benchmark for fine-grained controllable text style transfer. In *NAACL-HLT 2021*, pages 2116–2138. Association for Computational Linguistics.

Jing Qian, Li Dong, Yelong Shen, Furu Wei, and Weizhu Chen. 2022. Controllable natural language generation with contrastive prefixes. *arXiv preprint arXiv:2202.13257*.

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

Elisabeth Rudolph. 1989. The role of conjunctions and particles for text connexity. In *Text and discourse connectedness*, page 175. John Benjamins.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6830–6841.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. 2021. Challenges in detoxifying language models. In *Findings of EMNLP 2021*, pages 2447–2469. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kevin Yang and Dan Klein. 2021. FUDGE: controlled text generation with future discriminators. In *NAACL-HLT 2021*, pages 3511–3535. Association for Computational Linguistics.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *CoRR*, abs/2201.05337.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593.

A Implement Details

We detail the hyperparameters and experimental settings of Tailor and baselines as follows.

1. Tailor. Tailor is implemented based on Huggingface (Wolf et al., 2020). In all experiments of Tailor, we set the length of $\text{Tailor}_{\text{Concat}}$ to 128, as same as the MAP connector for $\text{Tailor}_{\text{Argmax}}$ and $\text{Tailor}_{\text{Weight}}$. As for the learning rate and the warm-up steps, $\text{Tailor}_{\text{Single}}$, $\text{Tailor}_{\text{Argmax}}$, and $\text{Tailor}_{\text{Weight}}$ are set to 5e-5 (the learning rate) and 0 (the warm-up steps), respectively. Besides, to get a pseudo label for MAP connector, we use the $\text{RoBERTa}_{\text{Large}}$ based classifier for both sentiment and topic of food type attributes. The hyperparameters can be found in Appendix C. Note that, for a fair comparison, we only use the same training set for each classifier as for training Tailor.

2. Finetune.[10] We use the $\text{GPT-2}_{\text{Base}}$ with a language model head implemented based on Huggingface. The learning rate is set to 5e-3 and the warm-up step is set to 0.

3. Adapter.[11] we set the bottleneck size to 5 to keep a similar size of training parameters with Tailor. The learning rate is set to 5e-5 and the warm-up step is set to 0.

4. GeDi.[12] For a fair comparison, we use the generative discriminator of GeDi based on $\text{GPT-2}_{\text{Base}}$ to guide generation of another $\text{GPT-2}_{\text{Base}}$. In inference, we use the $\omega = 30$, $\rho = 0.8$ and $\tau = 0.8$, as reported in their implementation.

5. PPLM.[13] We employ the original hyperparameter setting reported in Dathathri et al. (2020). In detail, $\gamma = 1.5$, $\gamma_{gm} = 0.9$, $\lambda_{kl} = 0.01$, iterations=3 and step size=0.02.

[10] https://huggingface.co/gpt2
[11] https://github.com/zlinao/VGLM
[12] https://github.com/salesforce/GeDi
[13] https://github.com/uber-research/PPLM/blob/master/paper_code/pplm.py

In inference, to keep in line with previous works (Dathathri et al., 2020; Krause et al., 2021), we use top-k sampling with k=10 and fix the random seed as 42 for all models to get the final results, while the maximum generation length is set to 60.

B Yelp Dataset

In this section, we elaborate the workflow of filtering, pre-processing and sub-sampling to get the attribute-specific dataset for training all models and the classifiers For correctness evaluation. First of all, we get the YELP dataset from Lample et al. (2019). In detail, each sample of the YELP dataset contains a review and the corresponding attributes.[14] Then, we select the restaurant reviews sub set as our original dataset. For dataset filtering, we use the dataset setup scripts offered by Lample et al. (2019), which contains a fastText(Joulin et al., 2017) classifier to filter sentences that are not written in English. After that, we filter the sentences with rated 3 stars, since they could be neutral in sentiment (Shen et al., 2017). Finally, we get the pre-processed dataset as illustrated in Table 8. For the classifiers that are used in correctness evaluation, we use the full dataset and details in Appendix C. Aside from it, for training Tailor and baselines, we randomly sample 30,000 / 3,000 sentences as training / validation data set for each attribute.

Model	F1 Score
Food Type Classifier	83.40
Sentiment Classifier	97.10

Table 7: The Performances of two classifiers on Yelp dataset.

Attribute	PO	NE	All
ME	25,169	89,411	114,580
AM	72,641	299,293	371,934
AS	47,680	185,551	233,231
All	145,490	574,255	719,745

Table 8: The number of reviews for each attribute in Yelp dataset.

C Classifiers For Correctness Evaluation

We use the RoBERTa$_{\text{Large}}$ based model to train two classifiers for both sentiment and topic of food type attributes. To obtain a balanced dataset, we randomly over-sampling the raw dataset. Finally, we get 1500k / 15k / 15k topic-specific sentences and 1380k / 1k / 1k sentiment-specific sentences for training / validation / testing, respectively. For training two classifiers, the learning rate is set to 5e-5 and the warm-up step is set to 200. The performances on the testing set can be found in Table 7.

D Human Evaluation Details

For human evaluation, we first set a guideline for evaluating, which includes the task background, key points, detailed descriptions and examples of evaluation scores from 1 to 5. Then, we set an entry barrier for annotators. In detail, we organize a training program and a preliminary annotating examination (90 examples for each model) to select appropriate annotators with a approval rate higher than 95%.

Score Definition We define two categories in human evaluation as follows:

1. **Quality** means whether the sentence corresponding to the option is fluent.

2. **Attribute** means whether the sentence corresponding to the option aligns with the target single attribute or multi attributes.

The scores are ranged from 1 to 5, and the higher score is better. The details are specified in Table 9. Following Qian et al. (2022), to obtain separate scores for both text quality and attribute correlation, the annotators are required to not attend to attribute correlation when evaluating the text quality (and vice versa). Aside from it, when the annotators feel that the sentences generated by different models perform similarly in terms of text quality, they are asked to give higher quality scores for sentences with longer lengths, which have more scope and diversity for expression yet have been ignored by automatic text quality evaluation metrics.

Inter-annotator agreement We use Fleiss' kappa (Fleiss, 1971) to measure three annotator's reliability.[15] The results are: 0.24 for score the quality (fair agreement), 0.55 for the attribute score (substantial agreement).

E Experiments on Cross Domain Dataset

We also evaluate Tailor on the cross domain dataset SST-2 (Socher et al., 2013) and AGNews (Zhang

[14]The format can be found via https://github.com/shrimai/Style-Transfer-Through-Back-Translation

[15]https://www.nltk.org/_modules/nltk/metrics/agreement.html

Type	Scores and Details
Quality	1 - All of sentences are difficult to read and incomprehensible.
	2 - Only a small part of sentences could be understood, which is readable and fluency.
	3 -Apart from a few grammatical mistakes, sentences are clear and comprehensive.
	4 - Sentences are free from grammatical errors and other linguistic inconsistencies, but could be better in style.
	5 - Sentences are fluency and spontaneous, which equate to the text quality of human writing.
Attribute	1 - There is no attribute-related words or phrases in the sentences.
	2 - There is only one attribute-related word or phrase in the sentences.
	3 - Sentences contain multiple attribute-related words or phrases, but they are almost repetitive.
	4 - Sentences contain multiple attribute-related words or phrases, a few of them are repetitive.
	5 - Sentences contain multiple attribute-related words or phrases, none of them are repetitive.

Table 9: Details of scores for Quality and Attribute in human evaluation.

Method	Trained Params	Correctness (%)	Text Quality	Diversity
	(%)	**Avg.** ↑ / Sent ↑ / News ↑	GRAM ↑ / PPL ↓	Dist-1/Dist-2/Dist-3↑
Finetune	100.00	62.54 / 69.14 / 55.93	0.74 / 37.78	0.09 / 0.34 / 0.50
Adapter	0.60	62.25 / 67.38 / 57.12	0.71 / 69.04	0.13 / 0.38 / 0.48
Adapter (Pseudo)	0.60	59.19 / 66.33 / 52.05	**0.79** / 105.65	0.11 / 0.38 / 0.54
$\text{Concat}_{\text{Simple}}$	0.00	55.81 / 74.35 / 37.27	0.47 / 49.39	0.11 / 0.47 / 0.80
$\text{Tailor}_{\text{Concat}}$	0.00	**63.38** / 68.08 / 58.67	0.59 / 36.82	0.11 / 0.48 / 0.80
$\text{Tailor}_{\text{Argmax}}$	0.08	61.42 / 63.65 / 59.18	0.68 / **35.33**	**0.13 / 0.53 / 0.84**

Table 10: The main results of multi-attribute CTG of SST-2 and AGNews dataset. We average the scores of eight combinations (two sentiment attributes × four topic attributes of news) as the final results. Adapter (Pseudo): using our argmax-pseudo labeled sentences to train the Adapter. $\text{Concat}_{\text{Simple}}$: simply concatenating two single-attribute prompts. Tailor-C: our non-training method. $\text{Tailor}_{\text{Argmax}}$: using argmax-pseudo prompts in the training stage of the MAP connector.

et al., 2015). For the classifiers that are used in correctness evaluation, we also use the $\text{RoBERTa}_{\text{Large}}$ based model to train two classifiers for both sentiment and topic of news attributes and reuse the parameters setting as in the experiment for YELP datasets. The F1 scores of two classifiers are 89.80 (sentiment) and 94.95 (news), respectively. For baselines and Tailor, we use the same experimenting setup as described in Appendix A. The experimental results of cross domain dataset are shown in Table 10.

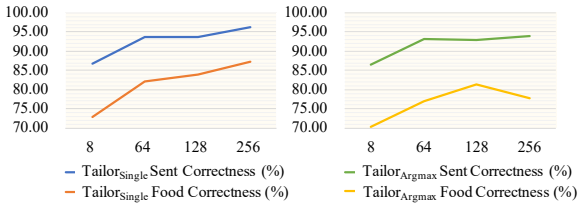

Figure 4: The results of using $\text{Tailor}_{\text{Single}}$ and $\text{Tailor}_{\text{Argmax}}$ with different lengths. The x-axis is the prompt length and the y-axis is the averaging correctness score (%).

F Ablations

Length of Tailor As shown in Figure 4, we explore the length of both $\text{Tailor}_{\text{Single}}$ and $\text{Tailor}_{\text{Argmax}}$. For singe-attribute prompt $\text{Tailor}_{\text{Single}}$, the performances increase alongside the length. But for $\text{Tailor}_{\text{Argmax}}$, it obtains the best performances with a length of 128, and the performances have a slight drop when we continue to increase the length.

Position Sensitivity We investigate the position sensitivity problem when concatenating two single-attribute prompts. As shown in Table 11, for simply concatenation, the GPT-2 tends to focus more on the prompt that is closer to the input prefix (*i.e.*, the attribute behind the dash in Table 11). For instance, NE attribute gets a 3.14% improvement if we put the corresponding prompt close to the input prefix. However, it also brings a 3.4% decrease for AM attribute as being away from the input prefix at the same time. In contrast, $\text{Tailor}_{\text{Concat}}$ keeps the same performance after swapping.

Unseen Combination In this part, we analyze the combining ability of Tailor on the unseen combination, which does not appear in Tailor's training stage. In the implementation, we randomly select one combination, remove the corresponding data

Method	Combination	Correctness (%)		
		Avg. ↑	Sent↑	Food↑
$\text{Concat}_{\text{Simple}}$	NE+AM	68.40	76.93	59.87
	AM+NE	68.27	80.07	56.47
$\text{Tailor}_{\text{Concat}}$	NE+AM	69.90	79.07	60.73
	AM+NE	69.90	79.07	60.73

Table 11: The results on multi-attribute CTG of generating sentences satisfying negative sentiment (NE) and topic of American food (AM). NE+AM denotes putting the positive attribute prompt in first and American food attribute prompt in later when concatenating them, in contrast to AM+NE.

from the training set for the MAP connector, and then test the performance of the MAP connector on this multi-attribute generation task. As shown in Table 12, $\text{Tailor}_{\text{Argmax}}$ still works to the unseen combination PO+ME and outperforms the non-training method $\text{Tailor}_{\text{Concat}}$ with 2.35% improvements.

Unseen	Method	Correctness (%)		
		Avg. ↑	Sent ↑	Food ↑
PO + ME	$\text{Tailor}_{\text{Concat}}$	87.54	95.60	79.47
PO + ME	$\text{Tailor}_{\text{Argmax}}$	89.89	97.07	82.70
None	$\text{Tailor}_{\text{Argmax}}$	91.64	97.87	85.40

Table 12: The results on unseen combination to multi-attribute CTG. PO + ME denotes the attribute combination of positive sentiment and topic of Mexican food.

G Case Study

To intuitively display the effects of various attributes, we show some generation results of single-attribute CTG in Table 13 and multi-attribute CTG in Table 14, respectively.

Attribute	Method	Generation Results
Negative Sentiment	Finetune	*Once upon a time*, I was very disappointed . The meat was bland and the beans tasted as if they had been sitting out all day...
	Adapter	*Once upon a time* in the restaurant it was still dark and people weren 't even talking...
	PPLM	*Once upon a time*, computers would have been able read, interpret and write, and listen, listen and read...
	GeDi	*Once upon a time* you either enter base build states or begin switching context switches and magic spells that alter your manifest...
	Tailor$_{\text{Single}}$	*Once upon a time*, you had to order your dinner. the food came out cold with no seasoning or flavor whatsoever ...
Positive Sentiment	Finetune	*Once upon a time* they were busy but the food was amazing and service fast . highly recommend for date night / evening out...
	Adapter	*Once upon a time* I 'd like to visit the city of lg, it was atlas! great food and amazing bartenders...
	PPLM	*Once upon a time* in the world in which a great deal of the work was done with the work was done in the world in the most...
	GeDi	*Once upon a time*, mankind thought of themselves as merciful , enlightened princes, with loving hearts. That prosperity and flourishing...
	Tailor$_{\text{Single}}$	*Once upon a time*, I was so excited to have my friends and family there that we wanted our food. the staff is great ! they make us feel at home when...
Mexican Food	Finetune	*Once upon a time* I had the carne asada burritos , they looked great. my wife's quesadilla was the only thing that she liked...
	Adapter	*Once upon a time* in my family I ordered the taco and they came out with two different varieties of beans. when it was finished we were asked...
	PPLM	*Once upon a time* the user would use a calculator to get a price for the price of the goods that...
	GeDi	*Once upon a time* Mexico had started guacamole called empty beans and children with Luis María de Leonos. Juan said he didn't tell a Hispanic what he did...
	Tailor$_{\text{Single}}$	*Once upon a time* I was in the area and had one of these burritos that were so delicious. we ordered them for...
American Food	Finetune	*Once upon a time* I was eating my burger , the server looked at me and said something to him " you have no idea how bad that is...
	Adapter	*Once upon a time*, I ordered the lobster bbq and we both got no meat. our server had been on call for over an hour before she left after talking to us again...
	PPLM	*Once upon a time* there were some people who used the same the machines as they could be to get their hands on and get the best results...
	GeDi	*Once upon a time* turkey sandwiches turned Uhhh . . . majestic, religious grunge, weened suburban ham sandwich — Americans applaud! Dove Bruffer » Briggs atte...
	Tailor$_{\text{Single}}$	*Once upon a time* I 'd go to the bbq and it was pretty empty. my friend had our burger , which we ordered with fries but not in advance of eating at dinner...
Asian Food	Finetune	*Once upon a time* I've had the spicy tofu dish , but that was my only meal. It came out cold and tasted awful...
	Adapter	*Once upon a time* I was craving something spicy, it tasted like the best Chinese food out there...
	PPLM	*Once upon a time* I made a stone of silver ring mail "Garden of the Winds Winds"...
	GeDi	*Once upon a time* bamboo noodles were the classical medicine and lemongrass fetish...
	Tailor$_{\text{Single}}$	*Once upon a time*, I got here for the sushi roll . After getting home from work at 4pm and finding...

Table 13: Samples of single-attribute CTG with input prefix '*Once upon a time*'. We highlight different attribute-specific words or phrases for better view.

Multi Attributes	Method	Generation Results
Negative + Mexican	Finetune	*Once upon a time* I ordered from there, it was just ok . the service was ok, the food is not worth the price ...
	Adapter	*Once upon a time* I was in phoenix and the place was not clean , I went back to try to find one more spot. no problem...
	Adapter (P)	*Once upon a time* I came here, the food was ok. however, they had an overpriced chicken burrito on tap and their guacamole is not as good or fresh ...
	Tailor$_{\text{Concat}}$	*Once upon a time* they had Mexican cuisine . The service is terrible , it's not clean and we were left hungry ! (I am so sorry for any inconvenience)...
	Tailor$_{\text{Argmax}}$	*Once upon a time*, we would be served the burritos that were cooked with no flavor . They didn't do it right and I will not return !...
	Tailor$_{\text{Weight}}$	*Once upon a time* I 'd had some of this Mexican food and it was pretty bland . now the burritos are not good ...
Positive + Mexican	Finetune	*Once upon a time* I had the chicken tacos and my fiancé ordered the carne asada torta . Both were outstanding . very clean , well prepared ...
	Adapter	*Once upon a time* I found it. The food and service was excellent as well. our server, kate, had an outstanding experience with...
	Adapter (P)	*Once upon a time* I went, we were in town for some reason and ordered the tacos that day. everything was amazing ! food is fresh ...
	Tailor$_{\text{Concat}}$	*Once upon a time* we had some amazing lunch, which included two tortillas and one taco . the service was great ! - no complaints there are plenty of...
	Tailor$_{\text{Argmax}}$	*Once upon a time*, I had the red bell chile and it was great ! our waitress came to get us as soon . We ordered some tacos with chicken nachos that...
	Tailor$_{\text{Weight}}$	*Once upon a time* I had the carne asada burrito and they were so good that it was one of my favorites . I will go back again for sure! ...
Negative + American	Finetune	*Once upon a time*, I tried it and had the worst hangover. after finishing my meal that day (which was a great one) all of sudden there is a cockroach...
	Adapter	*Once upon a time* I had the lobster rolls , they were cold and not appetizing . I also received one of these with chicken wings ...
	Adapter (P)	*Once upon a time*, I ordered the chicken sandwich . It was good but not quite as juicy or flavorful with any flavor at all ...
	Tailor$_{\text{Concat}}$	*Once upon a time*, I would have ordered the shrimp and fish salad . it was very dry with no flavor ! I ate this place on Sunday night so...
	Tailor$_{\text{Argmax}}$	*Once upon a time*, this was the place to be. I ordered my chicken burger and then there is no more fries or burgers at all! (if you don 't like that one)?
	Tailor$_{\text{Weight}}$	*Once upon a time* they would serve you the burger , but it was not cooked . No sauce in there! (I're sorry)? so that could be considered food poisoning ...
Positive + American	Finetune	*Once upon a time* they brought us our food. the staff was very kind and friendly . I ordered shrimp cocktail , it's one of those restaurants...
	Adapter	*Once upon a time*, the food came fast and fresh . the staff was attentive and we got everything ordered promptly . I recommend this...
	Adapter (P)	*Once upon a time* I had the lobster bisque , it was absolutely delicious . Service is very attentive and friendly ...
	Tailor$_{\text{Concat}}$	*Once upon a time*, I had the lobster sandwich that was good. it is one of my favourite dishes on this menu! (no other place in vegas has been more awesome .)...

	Tailor$_{\text{Argmax}}$	*Once upon a time*, I was in vegas with my girlfriend and she had the steak . It tasted great on its own! they were really friendly - very tasty food at their menu...
	Tailor$_{\text{Weight}}$	*Once upon a time*, I 'd have had the chorizo chicken sandwich . It is delicious ! Service was quick and friendly ...
Negative + Asian	Finetune	*Once upon a time* I was greeted, sat and waited patiently. the food took forever and there were only 6 of us that got our appetizers...
	Adapter	*Once upon a time* I got my food and was told that the service is slow . then they came over to me with an "error" ...
	Adapter (P)	*Once upon a time*, I would never recommend eating this place. the sushi was terrible and they...
	Tailor$_{\text{Concat}}$	*Once upon a time*, my mom had to order the fried rice at night and she said that it was so bad ...
	Tailor$_{\text{Argmax}}$	*Once upon a time*, I've had my rice and noodles at the Japanese buffet . They were so bland that...
	Tailor$_{\text{Weight}}$	*Once upon a time* I had the spicy ramen . It was too sweet and salty , but now its like they have been replaced with something else...
Positive + Asian	Finetune	*Once upon a time* I was craving for something quick and easy , they delivered ! the food was fresh and delicious ! service is great ...
	Adapter	*Once upon a time* I came here from chicago and had the chicken with my husband, we were greeted by an awesome customer service...
	Adapter (P)	*Once upon a time*, I would go to the sushi restaurant and order some sashimi . they have so many good things that can be found in...
	Tailor$_{\text{Concat}}$	*Once upon a time*, when I 't had sushi at my own house it was great . (-) the food is amazing ! We were seated on our first day here...
	Tailor$_{\text{Argmax}}$	*Once upon a time* they had sushi . I always try the kabobs , which is great for those who've never heard of them or even know what it means to be in chicago! :) ...
	Tailor$_{\text{Weight}}$	*Once upon a time*, I had the pho bao . Now they're going back for an even better experience! This is my favorite dish on earth and one of their most unique dishes ...

Table 14: Samples of multi-attribute CTG with input prefix '*Once upon a time*'. Negative + Mexican denotes generating sentences satisfying negative sentiment and topic of Mexican food. Adapter (P) denotes using the same argmax-pseudo labeled sentences (see § 3.2.2) to train the Adapter. We highlight different attribute-specific words or phrases for better view.