Role-playing based on Large Language Models via Style Extraction

Chunzhen Jin[1,2], Peng Cao[1,2⋆], and Osmar Zaïane[3]

[1] College of Computer Science and Engineering, Northeastern University, Shenyang, China
[2] Key Laboratory of Intelligent Computing in Medical Image of Ministry of Education, Northeastern University, Shenyang, China
[3] Alberta Machine Intelligence Institute, University of Alberta, Edmonton, Canada

Abstract. Role-playing is essential for leveraging large language models as it enhances user interactions, making the models more relatable and engaging. This is typically achieved through carefully crafted prompts for closed-source models or fine-tuning open-source models with specific style instructions. We propose a novel method called the Style Weight Language Model (SWLM), which extracts stylistic features from target roles and expresses styles through dialogue. Specifically, we first fine-tune the language model using a widely available instruction dataset. Next, we extract the desired role features using a mixed corruption strategy and store them in specific Style Weight Increments, which are injected into non-style models as representations of the desired style. To balance instructions and style, we also group and train Task Weight Increments for instructions. Experimental results demonstrate that SWLM reduces input token length and API consumption compared to prompt methods. Additionally, SWLM decouples instructions from style, reducing reliance on high-quality datasets. Remarkably, using only unsupervised role datasets, SWLM performs comparably to methods fine-tuned with style instruction sets while offering greater scalability. By enhancing the fluidity of interactions and minimizing resource consumption, SWLM represents a significant advancement in role-playing applications.

Keywords: Role-playing · Large Language Model · Weight Increments.

1 Introduction

Large language models (LLMs) such as Llama-2 [23] and ChatGPT-4 [14] are widely recognized as marked advancements in artificial intelligence development. These models have significantly transformed natural language processing (NLP), shifting the focus away from conventional tasks like summarization and translation towards more sophisticated and interactive functions, such as role-playing [24]. Specifically, role-playing is designed to allow LLMs to adopt and mimic different characters or personas with unique characteristics and styles of

⋆ Corresponding Author. Email: caopeng@mail.neu.edu.cn.

conversation [20,6,29]. This capability enhances user interactions, making the models seem more relatable, friendly, and engaging.

Currently, there are two main approaches to implementing role-playing: 1. Utilizing detailed prompts with closed-source model interfaces [24,8]. This method relies on extensive pre-training data and large-scale parameters to achieve optimal results. However, lengthy prompts increase costs and limit input sequence length [28]. 2. Fine-tuning open-source models with style-specific instructions [24]. This approach reduces the context window load but incurs additional costs due to the need for customized datasets. Additionally, the quality of these datasets may be influenced by subjective factors, and some studies attempt to construct personal profiles, which poses privacy risks [28].

This work introduces a novel approach called Style Weight Language Model (SWLM). The core idea of SWLM is to extract the stylistic features of the target role and inject them into the model aligned with the instruction. Our approach offers **three main advantages**: 1) Decoupling instruction and style training reduces the difficulty of acquiring customized datasets. 2) Extracting style features as a scalable incremental weight preserves input window space. 3) Directly injecting style features avoids the subjectivity of style instructions and the privacy risk of using personal profiles as the information retrieval source.

SWLM first conducts full-parameter fine-tuning of the model using a widely available instruction set (e.g., Alpaca [15]). Then, we extract style features from an unsupervised corpus with a specific role style. Employing a mixed corruption strategy, such as randomly masking parts of sentences or adding extra noise. During training, we freeze the model parameters and require the model to reconstruct sentences by updating weight increments. These increments are called **Style Weight Increments** and contain the style features learned for a given role in the latent space. Preliminary experiments reveal that despite minimal changes to parameters, the embedded increments still lead to instruction forgetting [10]. We introduce a specially designed module named task vector space and train **Task Weight Increments** to address this. During extracting style, we employ Key-Value retrieval to derive interpolated task increments adaptively, enhancing the adaptability of style increments across various scenarios.

Experiments showed that SWLM can autonomously learn the style of desired roles from unsupervised datasets and effectively engage in role-playing. Compared to style instruction fine-tuning methods, we achieved superior outcomes, with a smoother training process and easier dataset acquisition. While a slight performance gap exists compared to methods based on closed-source models like ChatGPT-4 [14] with diverse and complex agents, SWLM offers greater scalability and a more extended input sequence.

The main contributions can be summarized as follows: (1) We propose a new role-playing idea that eliminates the dependence on prompt words or style instructions, thereby achieving more flexible text processing. (2) The proposed method is scalable, and the incremental weights can be reused. (3) Extensive experiments demonstrate that SWLM exhibits performance comparable to the SOTA performance level.

2 Preliminary

2.1 Incremental Learning and Weight Increments

Incremental learning [16] supports the progressive acquisition of knowledge through a sequence of tasks while addressing the problem of catastrophic forgetting. In this context, a language model sequentially encounters T distinct tasks, represented as $\mathcal{T} = \{\mathcal{T}, \ldots, \mathcal{T}_T\}$. Each $\mathcal{T}_t$ includes a training set $\{(x_{t_i}, y_{t_i})\}_{i=1}^{N_T}$, where x_{t_i} is input and y_{t_i} is corresponding label. N_T is the number of training pair. The parameter f_θ of the model aims to minimize the generalization error across all tasks. This objective is formulated as:

$$\arg\min_{\theta} \sum_{t=1}^{T} \sum_{(x_t, y_t) \in \mathcal{T}_t} \mathcal{L}_{\mathcal{T}_t}(f_\theta(x_t), y_t) \tag{1}$$

where $\mathcal{L}_{\mathcal{T}_t}$ represents the loss function specific to task $\mathcal{T}_t$. Further, previous research [4,12,5] has demonstrated that fine-tuning minimal trainable parameters can significantly enhance the adaptability to specific tasks. Inspired by the above methods, we propose a Weight Increment strategy to extract features effectively. The following objective guides this adaptation:

$$\arg\min_{\theta'} \sum \mathcal{L}_{\mathcal{T}} \left(f_{\theta'}(\boldsymbol{x}), y\right) \tag{2}$$

where $\theta' = \theta_0 + \Delta\theta$, and $\Delta\theta$ represents weight increments and θ_0 is the frozen full-parameter. From an implementation standpoint, attention layers contain a wealth of potential information. By adjusting a portion of the parameters, the data centroids can be further optimized. In this work, we apply low-rank adaptation techniques [5] to further minimize computational costs. The update process of the attention layer is:

$$Y = W'\boldsymbol{x} + \text{bias} \approx (W_0 + \Delta W)\boldsymbol{x} = (W_0 + BA)\boldsymbol{x} \tag{3}$$

let $W_0 \in \mathbb{R}^{d \times k}$ be a frozen matrix, while $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ are trainable parameters, with $r \ll \min(d, k)$. Here, d and k denote the input and output feature dimensions, respectively, and bias refers to the bias term associated with the linear transformation.

2.2 Our Problem Formulation

Based on the above works, we propose a novel role-playing objective that decouples instruction execution from role style and conducts training separately:

$$\arg\min_{\tilde{\theta}} \sum_{(x,y) \in \mathcal{T}} \mathcal{L}_{\mathcal{T}}(f_{\tilde{\theta}}(x), y), \quad \tilde{\theta} = \theta_0 + \Delta\theta_{\mathcal{T}} + \Delta\theta_{\mathcal{S}} \tag{4}$$

where θ_0 is frozen, $\mathcal{L}_{\mathcal{T}}$ and $\Delta\theta_{\mathcal{T}}$ respectively represent the loss function and task increment weight specific to task $\mathcal{T}$. For $\Delta\theta_{\mathcal{S}}$, we will allocate the corresponding weight increment from memory to inject the style.

The key motivation is to treat style learning as a sub-task within incremental learning. The main challenge is establishing a beneficial connection between the parameter changes $\Delta\theta_{\mathcal{S}}$ and $\Delta\theta_{\mathcal{T}}$ to balance their mutual forgetting.

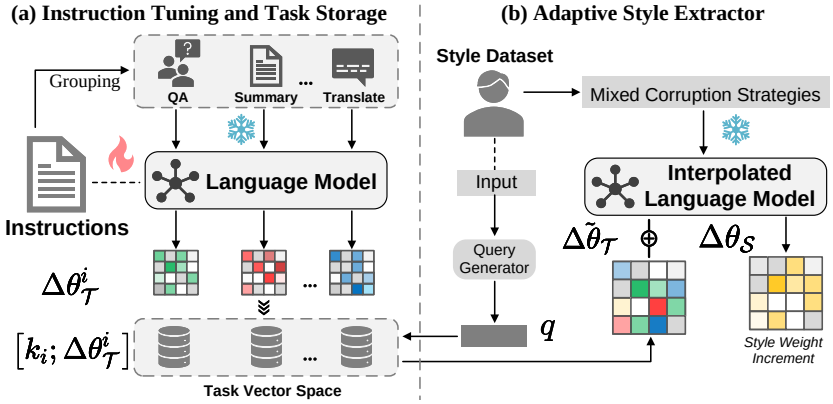

Fig. 1: Framework of SWLM. $\Delta\theta_{\mathcal{T}}$ and $\Delta\theta_{\mathcal{S}}$ represent task and style weight increments. $\Delta\tilde{\theta}_{\mathcal{T}}$ is the interpolated weight increment derived from task vector space based on $\boldsymbol{q}$, and $\boldsymbol{k}$ denotes the key vector associated with each $\Delta\theta_{\mathcal{T}}$.

3 Methodology

As is shown in Figure 1, our method can be divided into two main stages. In stage 1, we use an extensively acquired instruction dataset to align the LLMs with instructions. Then, we categorize the instruction set according to different task types and extract weight increments that contain task features. These task weight increments are then stored in the specific vector space. In stage 2, we introduce the Style Extractor module, which consists of two parts: first, we adopt a mixed corruption strategy for style extraction; second, we dynamically export interpolation task increments to adjust the latent space, helping learned style features that can adapt to complex dialogue scenes.

3.1 Instruction Alignment

Instruction tuning involves training models to better carry out specific instructions, improving their ability to align with expectations. In role-playing, the capability of a model to execute instructions is crucial. To this end, we utilized the Alpaca dataset [15] to enhance the performance of large models. Given the complex dialogue environments, we opted for full-parameter tuning to obtain θ_0, which offers excellent stability and effectiveness.

3.2 Construction of Task Vector Space

Grouping of Instructions. To simulate complex and variable environments, we propose diverse Task Weight Increments. This requires delineating boundaries within the instruction dataset to reduce interdependencies between sets of instructions. Firstly, we categorize the instructions based on the types of downstream tasks they address. Following the approach suggested by [13], we perform content-based clustering for each downstream task and apply a min-max strategy to segment these clusters. Finally, We define T distinct group tasks as $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$. For each supervised instruction group $\mathcal{T}_i$, we train to obtain the corresponding model by updating $\Delta\theta_{\mathcal{T}_i}$.

Task Vector Space. We constructed a Task Vector Space to store task weight increments based on key-value (KV) queries. In this section, we describe the methods to build Key and Query vectors and how to derive the required task weight increments. During construction, the key may be optimized, while it will be frozen later when setting up the style extraction environment.

Initially, we set up a collection of parameters using a semi-orthogonal matrix to ensure that the initial keys are orthogonal to one another. Next, for the input $\boldsymbol{x}$, we use a frozen BERT model [3], represented as $\boldsymbol{f}_{bert}(\cdot)$, to transform input $\boldsymbol{x} \in \mathbb{R}^{l \times c}$ into a hidden feature space. Then we obtain a query vector $\boldsymbol{q} \in \mathbb{R}^c$, where l and c indicate the sequence and feature dimension. During the optimization stage, we begin by computing the cosine similarity between the query $\boldsymbol{q}$ and each key $\boldsymbol{k}$. Then we employ the KNN to identify the $top-K$ closest vectors, represented as $K_q = \{k_1, \ldots, k_K\}$ where $K \leq T$. These selected keys K_q will be optimized to better match the distribution of the instance. The updated expression for the selected keys is as follows:

$$\boldsymbol{k}' \leftarrow \boldsymbol{k} + \gamma \nabla_k \cos(\boldsymbol{q}, \boldsymbol{k}), \quad \text{for } \boldsymbol{k} \in K_q, \tag{5}$$

where γ represents stride, and $\nabla_k \cos(\boldsymbol{q}, \boldsymbol{k})$ denotes the gradient of the cosine similarity function concerning $\boldsymbol{k}$.

After optimization in the vector space, all keys $\boldsymbol{k}$ are frozen and can only be derived through querying the relevant values, i.e., the task weight increments. Especially for any input $\boldsymbol{x}$, we compute the correlation between the query and the key to obtain the Retrieval Score $\mathbb{S}_t$, which is calculated as follows:

$$\mathbb{S}_t = \text{softmax}(\boldsymbol{q}^\top \cdot \boldsymbol{k}_t) \tag{6}$$

Finally, the adaptive incremental weights are computed as follows: $\Delta\tilde{\theta}_{\mathcal{T}} = \sum_{t=1}^{T} \mathbb{S}_t \cdot \Delta\theta_{\mathcal{T}}^t$. Then, the interpolated $\Delta\tilde{\theta}_{\mathcal{T}}$ will be dynamically embedded into the model parameters when extracting style features.

Discussion. The task vector space based on key-value retrieval ensures that the style features learned are adaptable in a variable environment. Specifically, x_T used to optimize the vector space ($\boldsymbol{x}_T \sim P_{\mathcal{T}}$) are sourced from the target

dataset. During retrieval, each input x_S originates from the style dataset (i.e., $\boldsymbol{x}_S \sim P_{\mathcal{S}}$). The distributional differences between the target and style datasets may lead to performance degradation or forgetting issues. We aim to fine-tune the parameters f_θ to fit the style dataset, thereby transforming it from f_θ to $f_{\theta+\Delta\theta_{\mathcal{S}}}$. The internal optimization objective of the task vector space is to identify the optimal centroid to fit the distribution of the style dataset best.

3.3 Style Extractor

Taking inspiration from [17], we introduce the Style Extractor module, which implements style transfer by adding or subtracting noise.

Corruption Strategies. Our objective is to explore different types of data corruption through an unsupervised learning approach similar to style transfer for feature extraction. We design two reconstruction tasks, each associated with a specific loss function. In these tasks, each sentence s_i in the dataset is altered by a specific function $\mho$, producing a new sentence $\tilde{s}_i = \mho(s_i)$. The choices of functions include **Noise** ($\mho_N$) and **Back-Translation** ($\mho_{BT}$).

Noise. $\mho_N$ manipulates input data in three ways to introduce randomness: dropping, replacing, and shuffling tokens. Following is a detailed breakdown: **Drop Noise:** Each token in a sequence is independently dropped with a probability p, where p is a noise probability sampled from a uniform distribution for each instance. **Replace Noise:** For each token in a sequence s_i, another sequence s_j is randomly chosen. Each token s_{ik} in s_i is replaced by the corresponding token s_{jk} in s_j with the same probability p. If s_j has fewer tokens than the position k being replaced in s_i, no replacement occurs for that token. **Shuffle Noise:** Tokens in a sequence s_i are selected with a probability p, and those selected tokens are shuffled among themselves. The above noise mechanisms enhance the ability to interpret and generate text in various styles.

Back-Translation. To enhance the coherence control in text generation, we adopt the corruption function method proposed by [11]. $\mho_{BT}$ involves setting the model to inference mode and transforming a sentence s_i into another style, thus generating a corrupted version of the sentence $\tilde{s_i}$. In previous studies, specifying a different target style was straightforward when labels were available. However, in our current scenario without labels, we achieve style transformation by randomly selecting another sentence s_j as the context.

During training, s_i serves as the target, $\tilde{s}_i$ serves as the input, and the preceding sentence s_{i-1} serves as the context for calculating the cross-entropy loss. The total loss for training is as follows:

$$\mathcal{L}_{\text{total}}(\tilde{\theta}) = -\sum_{\mho \in \mathcal{F}} \mathop{\mathbb{E}}_{s\sim\mathcal{S}} \log \Pr(s_i \mid \mho(s_i), s_{i-1}; \tilde{\theta}), \quad \tilde{\theta} = \theta_0 + \Delta\tilde{\theta}_{\mathcal{T}} + \Delta\theta_{\mathcal{S}} \tag{7}$$

where $\mathcal{F} = \{\mho_N, \mho_{BT}\}$ is the set of corruption functions. θ_0 is frozen, $\Delta\tilde{\theta}_{\mathcal{T}}$ is updated with query vector , and $\Delta\theta_{\mathcal{S}}$ is optimized through gradient descent.

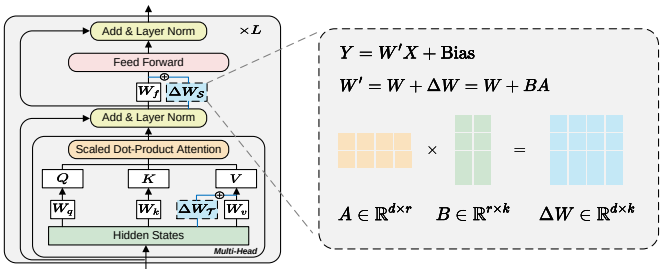

Fig. 2: The illustration demonstrates the selection of parameter segments designated for storing weight increments.

Application of Weight Increments. In the previous section, we discussed reparameterizing pre-trained models using two weight increments: **task weight increments** and **style weight increments**. Initial studies have found that optimizing the attention layers can effectively store instruction-level knowledge while optimizing feedforward layers enhances expressive capabilities.

In Figure 2, we specifically focus on optimizing the weight matrices in the value layers of the attention modules to extract task-related incremental weights. Additionally, we have added an extra bypass structure in the feedforward layers to capture Style Weight Increments.

Summary. To fully present our workflow, the algorithm of SWLM is shown in **Algorithm 1**.

4 Experiments

4.1 Experimental Settings

Datasets. The renowned ***Alpaca Instruction*** [15] set was selected as training data to enhance the backbone model's ability to understand and execute instructions. The Alpaca instruction set is a widely used instruction-aligned dataset that effectively trains models to perform in complex environments. We chose a dataset containing original dialogue statements from 48 characters of ***Genshin*** [27] for style extraction. Specifically, we selected 50 unsupervised dialogues for each sub-dataset: Xiangling, Hutao, Mona, Diluc, Venti, and Noelle. Additionally, we introduced the ***Shakespeare*** [31] dataset, which includes 2,000 selected sentences. These sentences are derived from various Shakespearean dramas and poems and have been rigorously filtered to ensure their representatives and authenticity of styles.

Backbones. LLaMA2-7B, 13B, and 33B are our backbone networks [23]. It is important to note that the LLaMA2 series is only a pre-trained model, and they require fine-tuning through instruction sets to enhance their command following and dialogue capabilities. For a fair comparison, we selected LLaMA2-7B as the backbone of the main experiments.

Algorithm 1: Style Weight Language Model

Input: Instruction $\mathcal{T}$ and correspond training sets $(\boldsymbol{x}_j, y_j)_{j=1}^{N} \sim \mathcal{D}_{\mathcal{T}}$;
Target style $\mathcal{S}$ and sentence set $(s_i)_{i=1}^{M} \sim \mathcal{D}_{\mathcal{S}}$
Output: Target style weight increment $\Delta\theta_{\mathcal{S}}$
Align instructions with full-parameter tuning to obtain θ_0
Cluster instructions via task categories to obtain $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$
for $t = 1, \ldots, T$ **do**
  Reset $\boldsymbol{k}$ and correspond weight increments $\Delta\theta_{\mathcal{T}}^{t}$
  for $(\boldsymbol{x}_j, y_j)_{j=1}^{N} \sim \mathcal{D}_{\mathcal{T}}^{T}$ **do**
    Use efficient method to train task weight increments $\Delta\theta_{\mathcal{T}}^{T}$ via Eq. 2
    Use frozen BERT to calculate query via $\boldsymbol{q} = \boldsymbol{f}_{bert}(\boldsymbol{x_j})$
    Use KNN to identify the most similar keys $K_q = \{k_1, \ldots, k_K\}$
    Upgrade key $\boldsymbol{k} \in \boldsymbol{k}_q$ via $\boldsymbol{k}' \leftarrow \boldsymbol{k} + \gamma \nabla_k \cos(\boldsymbol{q}, \boldsymbol{k})$ as Eq. 5
  end
  Store aligned key-value pair $[k_t; \Delta\theta_{\mathcal{T}}^{T}]$ in task vector space
end
for $(s_i)_{i=1}^{M} \sim \mathcal{D}_{\mathcal{S}}$ **do**
  Use Noise and Back-Translation to obtain $\mho_N(s_i)$ and $\mho_{BT}(s_i)$
  Calculate query via $\boldsymbol{q} = \boldsymbol{f}_{bert}(s_i)$
  Calculate Retrieval Score for each task $\mathbb{S}_t = \mathrm{softmax}(\boldsymbol{q}^{\top} \cdot k_t)$ as Eq. 6
  Obtain target weight increments via $\Delta\tilde{\theta}_{\mathcal{T}} = \sum_{t=1}^{T} \mathbb{S}_t \cdot \Delta\theta_{\mathcal{T}}^{t}$
  Update $\Delta\theta_{\mathcal{S}}$ via $\mathcal{L}_{\text{total}}(\tilde{\theta}) = -\sum_{\mho \in \mathcal{F}} \mathop{\mathbb{E}}_{s \sim \mathcal{S}} \log \Pr(s_i \mid \mho(s_i), s_{i-1}; \tilde{\theta})$as Eq. 7
end

Configuration. In our experiments with the LLaMA2 backbone, we employed 2 NVIDIA A100 GPUs. The learning rate was adjusted to 2×10^{-4}, with a weight decay of 0.01 and a batch size of 2. These experiments were conducted over 3 epochs with 0.03 warmup steps. The key optimization learning rate was 1×10^{-3}. All experiments used AdamW as the optimizer.

Baselines. We selected closed-source model baselines and open-source model baselines as follows: ***RoleGPT*** [24] relies on the ChatGPT-4 and is driven by specific prompts for role-playing. It use zero-shot and few-shot Prompt Engineering to control ChatGPT-4 [14]; The ***character. ai*** [21] is a platform dedicated to role-playing, and we consider it a proprietary one. Given that character.ai does not provide a public API or an interface for free extension, we had to collect data manually for evaluation. ***RoleLLaMA*** [24] and ***RoleGLM*** [24] are role-playing methods based on LLaMA2-7B and ChatGLM2-6B, respectively, fine-tuned with style instruction generating via ChatGPT-3.5 [2] to integrate role-specific knowledge into the model weights. ***Alpaca*** [15] is a variant based on LLaMA2-7B, trained on the Alpaca instruction set, and we used prompts to enable it to perform role-playing. Additionally, we examined ***LAMP-T5*** [18], a model that employs a style-based retrieval augmented generation approach.

Table 1: Presentation of Main Results. R1, R2, RL represent ROUGE-1, ROUGE-2, ROUGE-L; B1, B2 represent BLEU-1, BLEU-2; D1, D2 represent Distinct-1, Distinct-2. " +Task Increments" means we retrieved and embedded Task Weight Increments during inference.

Task	**Summarization**				**Dialog**					**cost**	
Metrics	R1	R2	RL	ACC	B1	B2	D1	D2	ACC	AS	TOKEN
Zero-shot RoleGPT	34.2	21.6	30.0	66.2	**42.7**	23.1	6.1	29.2	70.8	187.4	0
Few-shot RoleGPT	30.7	18.6	29.2	**70.8**	40.2	19.8	**6.8**	**29.6**	**73.5**	273.5	0
character.ai	25.1	12.1	19.2	52.8	28.2	16.2	3.3	17.8	50.2	-	-
RoleLLaMA	30.3	17.9	25.4	61.0	37.7	22.8	4.9	19.7	63.2	0	6,451,470
RoleGLM	29.1	15.2	22.8	59.3	38.9	24.1	4.2	17.2	61.8	0	6,451,470
LAMP-T5	25.3	12.5	20.4	52.7	36.7	23.7	3.5	14.9	54.0	0	1,552,120
Zero-shot Alpaca	34.7	20.1	31.2	53.6	40.2	23.1	2.8	16.5	55.1	187.4	0
Few-shot Alpaca	33.9	20.0	30.7	56.1	40.1	22.5	4.5	19.8	56.4	273.5	0
Ours	36.8	22.7	32.9	***62.7***	41.7	26.5	5.6	22.5	***63.4***	**0**	**0**
+Task Increments	**38.1**	**24.2**	**34.0**	58.3	41.5	**29.2**	4.9	19.7	59.1	**0**	**0**

4.2 Evaluation Protocol

Since role-playing is still an emerging field, existing evaluation benchmarks are not sufficiently robust [24]. To address this, we have integrated commonly used metrics from model evaluations with those from the style transfer domain to establish our evaluation criteria. Our approach includes standard generation benchmarks such as summarization tasks like **CNN/DM** and **Xsum** [19], as well as dialog tasks like **PersonaChat** and **DailyDialog** [7]. The former is a preliminary test of the generalization ability, and the latter tests the conversational ability in a role-playing scenario. We assess the overlap between the ground truth and the predicted responses to evaluate the general performance. These tasks are conducted in a five-shot manner to ensure optimal model performance.

Subsequently, we use a style classifier developed by [27] based on ChatGPT-4 [14] to evaluate the Style Accuracy (ACC). This solution has proven effective and relies on powerful LLMs that adapt to complex environments [27]. We also assessed the additional input sequence length (AS) per conversation and evaluated the number of input and output tokens (TOKEN) consumed in generating style instruction sets.

5 Results

5.1 Weight Increments vs. Prompt Engineer

Table 1 compares the SWLM method and Prompt Engineering. We first tested each baseline method with a summarization task in a role-playing setting. The results show that our method and the prompting approach effectively generate statements with distinctive character styles, maintaining high ROUGE scores while demonstrating considerable style consistency (ACC). Although our method

slightly lacks style by 11.4%, it slightly outperforms the Few-shot RoleGPT in overall content quality. Maintaining the same role-playing setup, we further tested dialogue generation capabilities. The experiments indicate that our method surpasses human-like responses in terms of overlap by 13.7%, showing more vital response consistency. However, it still lags in diversity and style consistency by 13.6%. It's important to note that the shortfall in style consistency might be less than this figure due to **self-bias**. In addition, the Prompt method has high requirements for the model's capabilities. There is a significant gap between Alpaca and RoleGPT using the same prompt.

Most importantly, our method does not occupy additional input sequences compared to Prompt Engineering. Statistical data reveal that the Few-shot RoleGPT requires 273.5 tokens per conversation to describe character traits and role-playing guidelines. From a cost-effectiveness standpoint, if GPT-4.0 is used as an interface, the additional cost of $0.01 per conversation could burden long-term users.

5.2 Weight Increments vs. Instruct Tuning

As shown in Table 1, our method outperforms the fine-tuning approach with style-specific instructions, especially in content overlap, where we lead by 23.4%. Our method shows a slight advantage in character style consistency with a performance increase of 2.7%.

Additionally, we have calculated the extra Token consumption for the compared methods when customizing style instruction sets. With just two thousand instructions generated per character, the additional Token consumption exceeds 6,000,000. Using ChatGPT-3.5 [2] to generate these instructions would cost $10; using ChatGPT-4 [14] would increase the cost to over $200. This indicates significant cost limitations on the scalability of the traditional instruction fine-tuning method. In contrast, our method only requires a few hundred unsupervised style datasets, significantly reducing costs.

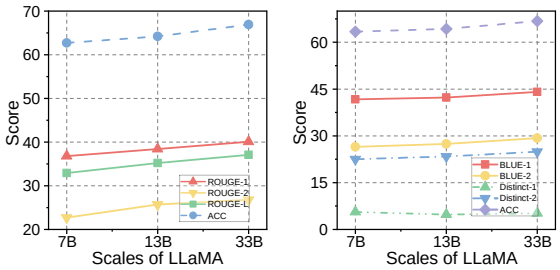

Fig. 3: Power of Scale.

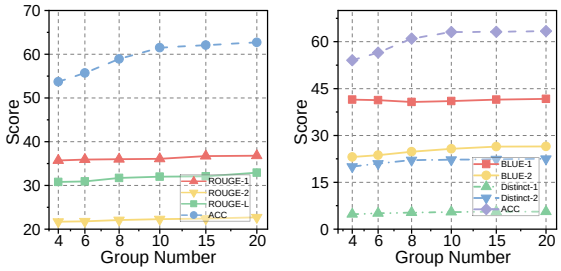

Fig. 4: Influence of Group Number.

Table 2: Ablation study.

Task	**Summarization**				**Dialog**				
Metrics	R1	R2	RL	ACC	B1	B2	D1	D2	ACC
w/o Instruction Alignment	22.5	14.7	20.5	47.2	27.7	29.5	7.6	28.5	45.8
w/o Task Vector Space	35.7	21.3	30.8	53.7	42.7	22.5	4.2	20.0	54.1
w/o KV retrial	35.9	22.1	32.3	54.9	40.2	21.3	4.8	23.1	56.5
w/o Task Style Extractor	35.9	24.1	35.3	52.9	41.5	24.4	4.3	25.2	51.5
w/o Noise	34.7	20.1	31.7	58.1	40.6	25.4	5.2	21.5	57.0
w/o Back-Translation	34.5	19.7	30.9	61.2	40.4	25.7	5.3	22.3	60.9
Ours	36.8	22.7	32.9	62.7	41.7	26.5	5.6	22.5	63.4

5.3 Analysis

Power of Scale. As shown in Figure 3, we analyze the various model scales in LLaMA-2 across different model sizes (7B, 13B, 33B). Our results show that performance improves with increasing model size. This indicates that larger models enhance the ability to handle complex contexts and generate more accurate responses. Therefore, we believe using larger open-source models can reduce the performance gap with ChatGPT-4.

Effect of Instruction Tuning. In the first row of Table 2, an instruction-aligned model is essential for role-playing tasks. Therefore, our approach requires

Table 3: Different Task Weight Parts.

Task Weight (♡) Style Weight (♠)					**Summarization**				**Dialog**				
Parts	Key	Query	Value	FFN	R1	R2	RL	ACC	B1	B2	D1	D2	ACC
i	♡			♠	35.1	22.4	31.7	61.5	40.5	25.9	**5.7**	22.2	62.9
ii		♡		♠	35.2	22.3	32.5	62.2	40.3	26.3	5.2	22.1	63.1
iii			♡	♠	**36.8**	**22.7**	**32.9**	**62.7**	**41.7**	**26.5**	5.6	**22.5**	**63.4**
iv	♠		♡		36.2	22.1	31.9	61.7	40.2	25.5	5.5	22.3	62.4
v		♠	♡		36.0	22.3	32.7	61.2	40.4	26.2	5.6	22.4	62.5

a foundation based on an instruction-aligned model; otherwise, conversational abilities and role style will significantly deteriorate.

Effect of Task Vector Space. We introduced a task vector space that dynamically adjusts the model parameters based on stylistic inputs, enabling the style to adapt to various environments effectively. In the second row of Table 2, we conducted an ablation study by freezing the model parameters and removing the task vector space. The results demonstrate a significant decline in style consistency. Additionally, the third row of Table 2 details another ablation where the weight increment interpolation of KV retrieval was replaced with static linear interpolation. Although this modification had a less severe impact than removing the entire vector space, it still significantly affected the outcomes.

Table 4: Different Weight Types.

Task	**Summarization**				**Dialog**				
Metrics	R1	R2	RL	ACC	B1	B2	D1	D2	ACC
Finetune	26.1	17.5	23.1	57.3	32.7	18.3	5.4	20.5	52.1
Prompt	30.2	18.5	24.3	52.3	31.0	19.2	5.1	19.5	53.0
Adapter	32.5	20.2	25.7	56.7	35.3	20.5	5.3	21.8	57.4
Ours	36.8	22.7	32.9	62.7	41.7	26.5	5.6	22.5	63.4

Effect of Style Extractor. The accuracy of text style is a critical component in role-playing tasks. To explore the effectiveness of style extraction, we conducted an ablation study on style weight increments. As shown in the fourth row of Table 2, the model's ability to express style significantly declined when it lacked style weight increments. Additionally, we conducted ablation studies on different corruption strategies. As indicated in the fifth and sixth rows of Table 2, removing Noise Strategy significantly negatively impacted the accuracy of style modeling (ACC). At the same time, Back-translation primarily affected the integrity of sentences (evidenced by a lower ROUGE or BLEU overlap).

Influence of Different Weight Parts. As shown in Table 3, we tested the performance of task and style weight increments in different parts through permutations and combinations. Our conclusions indicate that the attention layer retains more instruction-related knowledge, while the feedforward layer focuses more on the expression style. Placing the task increment weight in the Value bypass yields more robust overall performance, whereas putting it in the feedforward layer bypass results in better consistency of character style.

Types of Weight Increments. We replaced the low-rank weight increments with prompts and adapters and conducted a comparison using a two-stage individual fine-tuning process. Table 4 shows that fine-tuning ensures style stability

Table 5: Human Evaluation Metrics.

Method	Style	Content	Fluency	avg.Rank
RoleGPT	**1.33**	1.93	**1.40**	**1.55**
RoleLLaMA	2.83	2.70	2.06	2.53
Ours	1.83	**1.36**	1.86	1.68

but causes catastrophic instructions forgetting. Compared to the other two incremental methods, the low-rank weight increments improve style accuracy by 10.5%, resulting in more stable overall performance.

Influence of Group Number. Figure 4 illustrates the model's performance when instructions are divided into different numbers of groups. Specifically, dividing instructions into 10-20 groups can achieve optimal performance, and the accuracy of style recognition slightly improves as the number of groups increases. However, having more groups also means the initial preparation will require more computational resources.

Human Evaluation. We recruited 10 participants to evaluate 50 instances from each style based on (1) **Style** transfer strength, (2) **Content** integrity, and (3) sentence **Fluency**. The models were ranked from best to worst, from 1 to 3. Table 5 shows the average rankings of the five models based on participant feedback. Our model outperforms two strong baselines in Content but is inferior to RoleGPT in other aspects.

6 Related Work

Role-Playing. Recent advancements in LLMs have demonstrated promising capabilities in customization and role-playing. Notably, developments in LLMs such as GPT-4 [14] and LLaMA [23] showcase the potential to enhance user interactions. However, there remains a significant gap between open-source LLMs and their closed-source counterparts, such as ChatGPT-4 [24]. Open-source LLMs often rely on additional input sequences to maintain role consistency and interactivity [24,18,28], which can reduce the effective input window size. In contrast, closed-source models benefit from extensive access to high-quality, diverse datasets that enhance their adaptability and personalization [8,9].

Vector Space Model (VSM). Unlike traditional keyword-based [1] or rule-driven retrieval methods [22], VSM has emerged as a pivotal technique in the field of information retrieval. It transforms queries into vectors within a latent space, enabling the use of similarity measures like vector similarity to assess the relevance of documents. Previous research has successfully integrated VSM into various applications and contextual learning scenarios [25,26,30]. In our work, VSM is utilized to facilitate model transfer and adaptation, addressing the challenges of dynamic downstream tasks.

7 Conclusion and Limitation

The SWLM eliminates the need for extensive prompts, offering flexible and scalable text processing. It effectively learns character styles from unsupervised datasets, achieving performance comparable to state-of-the-art models. We also acknowledge that the additional retrieval framework increases computational and memory storage costs. Nonetheless, this extra usage is minor compared to the GPU memory usage of PLMs employed for word prediction. Another limitation is it ignores psychological factors or background in character role-playing.

Acknowledgments. This research was supported by the National Natural Science Foundation of China (No.62076059), Science and Technology Joint Project of Liaoning province (2023JH2/101700367), and Fundamental Research Funds for the Central Universities (N2424010-7). Osmar Zaiane gratefully acknowledges the funding from Natural Sciences and Engineering Research Council (NSERC) of Canada and the Canada CIFAR AI Chairs Program for Amii.

References

1. Berry, M.W., Drmac, Z., Jessup, E.R.: Matrices, vector spaces, and information retrieval. SIAM review **41**(2), 335–362 (1999)
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
3. Devlin, J., Chang, M.W., Lee: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: International conference on machine learning. pp. 2790–2799. PMLR (2019)
5. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
6. Huang, Y., Li, Y., Zhang, L.a.: MVP-tuning: Multi-view knowledge retrieval with prompt tuning for commonsense reasoning. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. pp. 13417–13432. Association for Computational Linguistics, Toronto, Canada (Jul 2023)
7. Jandaghi, P., Sheng, X.: Faithful persona-based conversational dataset generation with large language models (2023)
8. Jiang, H., Zhang, X., Cao, X., Kabbara, J., Roy, D.: Personallm: Investigating the ability of gpt-3.5 to express personality traits and gender differences. arXiv preprint arXiv:2305.02547 (2023)
9. Kirk, H.R., Vidgen, B., Röttger, P., Hale, S.A.: Personalisation within bounds: A risk taxonomy and policy framework for the alignment of large language models with personalised feedback. arXiv preprint arXiv:2303.05453 (2023)
10. Kirkpatrick, J., Pascanu, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
11. Lample, G., Subramanian, S., Smith, E., Denoyer: Multiple-attribute text rewriting. In: International Conference on Learning Representations (2018)

12. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021)
13. Li, J., Tang, T., Nie, J.Y., Wen, J.R., Zhao, W.X.: Learning to transfer prompts for text generation. arXiv preprint arXiv:2205.01543 (2022)
14. OpenAI, Achiam, J., Adler, S., Agarwal, S.: Gpt-4 technical report (2024)
15. Peng, B., Li, C., He, P., Galley, M., Gao, J.: Instruction tuning with gpt-4. arXiv preprint arXiv:2304.03277 (2023)
16. PENG, B., Tian, Z., Liu, S., Yang, M.C., Jia, J.: Scalable language model with generalized continual learning. In: The Twelfth International Conference on Learning Representations (2024)
17. Riley, P., Constant, N., Guo, M., Kumar, G., Uthus, D., Parekh, Z.: Textsettr: Few-shot text style extraction and tunable targeted restyling. arXiv preprint arXiv:2010.03802 (2020)
18. Salemi, A., Mysore, S., Bendersky, M., Zamani, H.: Lamp: When large language models meet personalization. arXiv preprint arXiv:2304.11406 (2023)
19. See, A.: Get to the point: Summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1073–1083. Vancouver, Canada (2017)
20. Shao, Y., Li, L.a.: Character-LLM: A trainable agent for role-playing. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 13153–13187. Association for Computational Linguistics (Dec 2023)
21. Shazeer, N.: Character.ai: Ai chatbot service (2022), accessed: 2024-05-15
22. Singhal, A., et al.: Modern information retrieval: A brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
23. Touvron, H., Martin, L., Stone, K., Albert, P.: Llama 2: Open foundation and fine-tuned chat models (2023)
24. Wang, Z.M., Peng, Z., Que, H., Liu, J., Zhou, W., Wu, Y., Guo, H., Gan, R., Ni, Z., Yang, J., Zhang, M., Zhang, Z., Ouyang, W., Xu, K., Huang, S.W., Fu, J., Peng, J.: Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models (2024)
25. Wang, Z., Zhang, Z., Lee: Learning to prompt for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 139–149 (2022)
26. Wang, Z., Mehta, S.V., Póczos, B., Carbonell, J.: Efficient meta lifelong-learning with limited memory. arXiv preprint arXiv:2010.02500 (2020)
27. Xu, R., Huang, Y., Chen, X., Zhang, L.: Specializing small language models towards complex style transfer via latent attribute pre-training. arXiv preprint arXiv:2309.10929 (2023)
28. Xu, X., Wang, Y., Xu, C., Ding, Z., Jiang, J., Ding, Z., Karlsson, B.F.: A survey on game playing agents and large models: Methods, applications, and challenges. arXiv preprint arXiv:2403.10249 (2024)
29. Yang, W., Liu, J., Cao, P., Zhu, R., Wang, Y., Liu, J.K., Wang, F., Zhang, X.: Attention guided learnable time-domain filterbanks for speech depression detection. Neural Networks **165**, 135–149 (2023)
30. Yang, W., Wen, G., Cao, P., Yang, J., Zaiane, O.R.: Collaborative learning of graph generation, clustering and classification for brain networks diagnosis. Computer Methods and Programs in Biomedicine **219**, 106772 (2022)
31. Zhu, X., Guan, J., Huang, M., Liu, J.: Storytrans: Non-parallel story author-style transfer with discourse representations and content enhancing. arXiv preprint arXiv:2208.13423 (2022)