Generating Personalized Prompts

ChanJoo Jung (2022121057)

1 Introduction

This is a simple project for CAS2105 @ Yonsei University in 2025.

It is widely known that training a language model makes the model better generate outputs [1, 2]. With this idea in mind, I have thought of training a model to **generate personalized prompts** as outputs.

2 Task Definition

- **Task description:** The goal of this project is to train a model that generates personalized prompts based on my preferred writing style. When a topic is provided, the model outputs a prompt that reflects my specific tone and style.
- **Motivation:** I realized that outputs change whenever the prompt changes. With this in mind, I adopted a specific prompting style to ensure the model generates the desired output. Through extensive experience, I developed specialized prompts that yield the results I want. However, manually rewriting prompts every time to match my style is time-consuming and can be inconsistent. In fact, if I don't use AI for an extended period, I may forget my preferred style. Therefore, I have always considered building a model that automatically generates prompts following my specialized format. Undoubtedly, a lightweight, personalized prompt generator would help me work faster and maintain consistency.
- **Input / Output:**
  - **Input:** A topic or instruction (e.g., "Make a prompt that converts raw text into bullet points.")
  - **Output:** A personalized prompt that resembles my "liked" examples (e.g., "Convert the text into 4 bullet points. Make sure each point is under 15 words.")
- **Success criteria:** The system is considered successful if:
  - Generated prompts are stylistically closer to my preferred "liked" prompts than the vanilla model baseline.
  - ROUGE-1 and ROUGE-L scores are higher than the baselines.
  - This is because, for stylistic tasks, overlapping words, phrases, and sentence structures indicate that the style is being matched.
  - For example, if I frequently use a particular phrase across my prompts, a higher ROUGE score would reflect that the trained model successfully captured this preference.

3 Methods

3.1 Naïve Baseline

The naïve baseline is the randomly chosen unified prompt and the vanilla model.

1. Randomly chosen unified prompt

- This is the prompt that we choose randomly from the train data.
- We would serve it like a unified prompt for any input.
- This means that whatever the input, the output prompt would be this randomly chosen prompt.
- This is to satisfy one of the specifications in the HW guideline to use a not sophisticated model as a baseline.
- Nonetheless, I believe the real baseline should be 2) Vanilla model.

2. Vanilla Model Baseline

- Uses the **untrained, initialized model** without any training.
- This baseline is used because my main method includes a training step, so the fairest comparison is a model with **no learning at all**.
- It represents the behavior of a model that has **not learned my style** in any way.

Why both baselines are naïve

- Does not learn personal style (Does not know anything about my style)
- Same structure every time
- Ignores tone, structure, and domain
- Fails on longer or creative tasks

Common Failure Cases

- Generating prompts needing specific tone or phrasing
- Generating prompts that usually have specific favorite words or phrases
- Domain-specific topics (ML, coding)

3.2 AI Pipeline

- **Models used:** `TinyLlama/TinyLlama-1.1B-Chat-v1.0`
- **Pipeline stages:**
  - **A. Preprocessing**
    1. Load data (each JSONL line has input and output).
    2. Shuffle and split the data.
    3. Load tokenizer.

4. Create prompt:

```
### Instruction:
{input}

### Response:
```

5. Labels are masked so loss only on response tokens. (set labels of instruction tokens to -100)

6. Pads dynamically to MAX_LEN (If pad_token is missing, we set it to eos_token to enable padding.) OR Truncate to MAX_LEN (Inputs and prompt+output are truncated to MAX_LEN=256 tokens.)

– **B. Representation or embedding**

7. Load a base model (TinyLlama/TinyLlama-1.1B-Chat-v1.0)

8. Load a LoRA on top of that base model for lighter training and implementation. (attention (q,k,v,o), and MLP projections (gate, up, down) are used, these are commonly used hyperparameters when setting a LoRA.)

– **C. Decision or ranking**

9. Train the LoRA that is attached to the base model. (Only LoRA parameters are updated. Base model is frozen / not trained.)

10. It trains using standard causal LM loss over response tokens. (hyperparameters: 3 epochs, batch size=2, grad accum=4)

– **D. Optional post-processing**

11. Generate outputs (personalized prompts) on the given test-set inputs deterministically. (greedy decoding: it will only generate an output with the best token selections.)

12. After decoding, we remove the prompt prefix and keep only the response.

- **Design choices and justification:** Hyperparameters are all suited for the small model and small data. I also hypothesized a trained, learned model will better align my prompt styles.

4 Experiments

4.1 Datasets

- **Dataset source**: I created manually.
- **Size**: 61 data samples.
- **Splits**: 51 for train samples. 10 for test samples. I did not include validation set due to limited data.
- **Preprocessing**: See Section 3.2

4.2 Metrics

- **Simple generation:** ROUGE-1, ROUGE-L (following prior personalization works [3, 4])

4.3 Results

The results are presented in the Table below.

Model	**ROUGE-1**	**ROUGE-L**
Randomly Chosen Unified	0.1958	0.1707
Vanilla Model	0.1255	0.1115
Trained Model	**0.2089**	**0.2041**

4.4 Qualitative Examples

Example 1

INPUT:

Make a prompt that rewrites English text in academic style.

GOLD:

Rewrite the text in academic style. Make sure it is clearer and more concise. Show only the revised version.

RANDOM:

Find unclear parts of the definition. Explain them in one short sentence. Rewrite it clearly.

VANILLA:

Please provide a prompt that rewrites the text in academic style: \n 1. Please provide a prompt that rewrites the text in academic style: \n 2. Please provide a prompt that rewrites the text in academic style: \n 3. Please provide a prompt that rewrites the text in academic style: \n 4. Please provide a prompt that rewrites

TRAINED:

Rewrite the text in academic style. Make sure to keep the meaning. Make sure to use academic language. Make sure to proofread. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again. Make sure to

Example 2

INPUT:

Make a prompt that summarizes an interview transcript.

GOLD:

Summarize the interview. Use 3 bullet points only. Make sure each bullet is under 20 words.

RANDOM:

Find unclear parts of the definition. Explain them in one short sentence. Rewrite it clearly.

VANILLA:

Thank you for your time and interest in our company. Here is a summary of our interview with [Name], who is currently working as [Position] at [Company Name]. \n [Name] is a [Age]-year-old [Gender] who has been with our company for [Number of Years] years. They have [Number of Job Titles] in their current role

TRAINED:

Summarize the interview. Make sure to keep it short. Make sure to use bullet points. Make sure to keep it under 100 words. Make sure to use short sentences. Make sure to keep it clear. Make sure to make it short. Make sure to make it short. Make sure to make it short. Make sure to make it short. Make sure to make it short

Example 3

INPUT:

Make a prompt that explains a Python function line by line.

GOLD:

Explain the Python code line by line. Make sure you say what each line does and why it matters.

RANDOM:

Find unclear parts of the definition. Explain them in one short sentence. Rewrite it clearly.

VANILLA:

python \n def square(num): \n ””” \n This function takes a single integer parameter ’num’ and returns the square of num. \n ””” \n return num ** 2 \n # Example usage: \n print(square(5)) # Output: 25 \n

TRAINED:

Explain the function line by line. Make sure to include the function name and parameters. Make sure to use bullet points. Make sure to use short sentences. Make sure to use clear language. Make sure to proofread. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again. Make sure to proofread again.

5 Reflection and Limitations

Actually, the trained model did perform better than the vanilla model. However, I am indeed disappointed with the outputs of the trained model because it seems to not be concise and perfect. Yes, I can see some important keywords (i.e. favorite words and phrases as reflected in ROUGE scores) that appear in the generated outputs. But, I would honestly say that I won’t use those prompts in the real world. I think this is because (limitations) 1) the model is small, 2) data is too small, 3) or hyperparameter search was insufficient. (This means that the hyperparameters used may be suboptimal.) Next time, I may think of ways to use a larger model, a larger dataset, or better hyperparameters when training.

Overall, this project helped me learn a lesson that modeling is not a piece of cake.

References

[1] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[3] Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning, 2025. URL https://arxiv.org/abs/2402.04401.

[4] Hyungjune Bu, Chanjoo Jung, Minjae Kang, and Jaehyung Kim. Personalized llm decoding via contrasting personal preference, 2025. URL https://arxiv.org/abs/2506.12109.