Table 9: **Examples of SFT data for thinking and non-thinking modes during the thinking mode fusion stage.** For the thinking mode, the `/think` flag can be omitted since it represents the default behavior. This feature has been implemented in the chat template[1] supported by the Hugging Face's tokenizer, where the thinking mode can be disabled using an additional parameter `enable_thinking=False`.

Thinking Mode	Non-Thinking Mode
<|im_start|>user {query} /think<|im_end|> <|im_start|>assistant <think> {thinking_content} </think> {response}<|im_end|>	<|im_start|>user {query} /no_think<|im_end|> <|im_start|>assistant <think> </think> {response}<|im_end|>

stable, which is crucial for maintaining stable training. As a result, we achieve consistent improvements in both training reward and validation performance over the course of a single RL run, without any manual intervention on hyperparameters. For instance, the AIME'24 score of the Qwen3-235B-A22B model increases from 70.1 to 85.1 over a total of 170 RL training steps.

4.3 Thinking Mode Fusion

The goal of the Thinking Mode Fusion stage is to integrate the "non-thinking" capabilities into the previously developed "thinking" model. This approach allows developers to manage and control reasoning behaviors, while also reducing the cost and complexity of deploying separate models for thinking and non-thinking tasks. To achieve this, we conduct continual supervised fine-tuning (SFT) on the Reasoning RL model and design a chat template to fuse the two modes. Moreover, we find that models capable of handling both modes proficiently perform consistently well under different thinking budgets.

Construction of SFT data. The SFT dataset combines both the "thinking" and "non-thinking" data. To ensure that the performance of the Stage 2 model is not compromised by the additional SFT, the "thinking" data is generated via rejection sampling on Stage 1 queries using the Stage 2 model itself. The "non-thinking" data, on the other hand, is carefully curated to cover a diverse range of tasks, including coding, mathematics, instruction-following, multilingual tasks, creative writing, question answering, and role-playing. Additionally, we employ automatically generated checklists for assessing the response quality of "non-thinking" data. To enhance the performance on tasks with low-resource languages, we particularly increase the proportion of translation tasks.

Chat Template Design. To better integrate the two modes and enable users to dynamically switch the model's thinking process, we design chat templates for Qwen3, as shown in Table 9. Specifically, for samples in thinking mode and non-thinking mode, we introduce `/think` and `/no_think` flags in the user query or system message, respectively. This allows the model to follow the user's input and select the appropriate thinking mode accordingly. For non-thinking mode samples, we retain an empty thinking block in the assistant's response. This design ensures internal format consistency within the model and allows developers to prevent the model from engaging in thinking behavior by concatenating an empty think block in the chat template. By default, the model operates in thinking mode; therefore, we add some thinking mode training samples where the user queries do not include `/think` flags. For more complex multi-turn dialogs, we randomly insert multiple `/think` and `/no_think` flags into users' queries, with the model response adhering to the last flag encountered.

Thinking Budget. An additional advantage of Thinking Mode Fusion is that, once the model learns to respond in both non-thinking and thinking modes, it naturally develops the ability to handle intermediate cases—generating responses based on incomplete thinking. This capability lays the foundation for implementing budget control over the model's thinking process. Specifically, when the length of the model's thinking reaches a user-defined threshold, we manually halt the thinking process and insert the stop-thinking instruction: "`Considering the limited time by the user, I have to give the solution based on the thinking directly now.\n</think>.\n\n`". After this instruction is inserted, the model proceeds to generate a final response based on its accumulated reasoning up to that point. It is worth noting that this ability is not explicitly trained but emerges naturally as a result of applying Thinking Mode Fusion.