How the Choice of Chat Template Impacts Your Model: An Empirical Study

Anonymous ACL submission

Abstract

Instruction-tuning is an essential part of the LLM post-training pipeline. It normally involves training models on examples of exchanges between a user and a model responding to their request, formatted as chat messages. To make these messages digestible by models, one typically applies a *chat template* to convert them to plain text. Despite the ubiquity of this approach, the impact of chat templates on models' performance remains critically understudied. We demonstrate that models' native chat templates are associated with higher per-token loss than other parts of the instruction. Replacing those tokens with a natural-sounding format during instruct-tuning can lead to significant performance gains. Additionally, our study shows that the accuracy of Instruct models during inference can be improved by using a natural template rather than the one they were trained on. More broadly, this work highlights that chat template selection affects overall performance and analyzes this phenomenon through loss evolution[1].

1 Introduction

Fine-tuning LLMs, and in particular, instruction-tuning has long been shown to improve models' capabilities (Wei et al., 2022; Ouyang et al., 2022). Thus, supervised fine-tuning (SFT) on instruction following datasets, such as Tülu Mix (Wang et al., 2023), Personahub (Ge et al., 2024), or Alpaca (Taori et al., 2023), has become common in the LLM training process (Qwen Team et al., 2025; Gemma Team et al., 2024).

Currently, the instruction-following phase involves training on examples of user-LLM exchanges, as successful examples of such an approach emerged (Xu et al., 2024; Chiang et al., 2023). The instructions are structured as chat messages between a *user* and an *assistant* (LLM), where the user poses a question or provides a task, and the assistant's message contains the expected response. As raw data it is structured as such: `[{'role': 'user', 'content': '<user question>'}, {'role': 'assistant', 'content': '<model response>'}]`. In order to adapt these instructions into data digestible by LLMs, many model families have introduced their own formatting style to convert such messages into text, called *chat templates*[2] (Qwen Team et al., 2025; Wang et al., 2023; Gemma Team et al., 2024). Despite the omnipresence of this practice, previous research into the impact of these chat template formats remains limited.

More precisely, chat templates typically consist of special technical tokens that mark the beginning and the end of each message. Due to such a design choice, models have little exposure to this type of vocabulary during the pre-training phase since it is atypical for raw text data to contain structured messages. Thus, one can expect that pre-trained models have significantly higher cross entropy loss on those special tokens than on other, more familiar, parts of the prompt. However, when we prompt Gemma 2 2B (Gemma Team et al., 2024) and its instruct-tuned version, we see that even *fully instruction-tuned LLMs have loss spikes associated with chat tokens* (as illustrated in Figure 1).

Previous work (Lin et al., 2026; Wu et al., 2026) indicated that when tokens with high loss are observed during training, it might be beneficial for the model's performance to limit their impact, either by clipping the gradient or masking them. Instead of performing token-agnostic masking, our proposal is to replace chat template tokens with ones that models are more likely to have in their distribution during pre-training. To achieve this, we propose to consider the format we name *natural chat template* only using **`User:`** and **`Assistant:`** as mes-

[1] Code will be provided upon acceptance.

[2] See Appendix A for chat template examples.

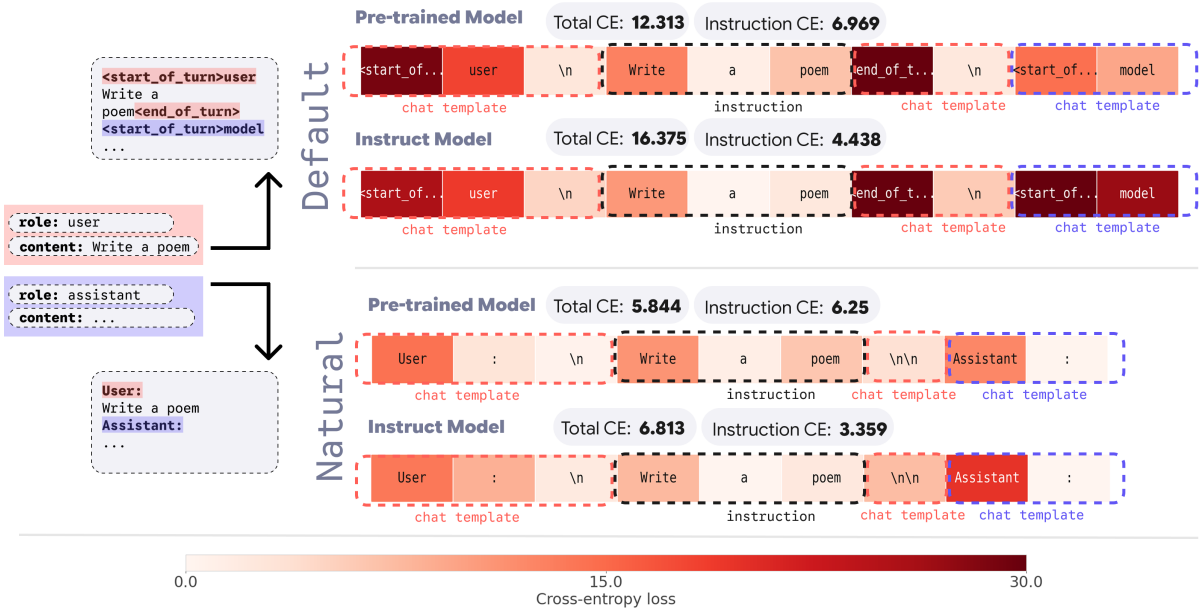

Figure 1: cross entropy loss of Gemma 2 2B and Gemma 2 2B-it for an instruction wrapped using the model's default chat template (top) and the proposed natural template (bottom). The color bar represents values of per-token cross entropy. Additionally, we report the per-prompt cross entropy and cross entropy for instruction only.

sage separators. We demonstrate that using this format while retaining all instruction content yields a notable reduction in associated loss (as shown in Figure 1), indicating its potential to improve training dynamics.

This "simplification" brought by the natural chat template is the central subject of this study. More specifically, we raise the following research questions regarding replacing models' default chat templates with the natural one during instruction-tuning:

1. Is there a measurable impact on benchmark performance? And if so, is it associated with better loss convergence during training?

2. How does per-token loss change after training? Is the effect limited only to chat template tokens?

3. Is there any measurable effect on loss convergence during training?

4. Are both settings equally efficient in parameter updates?

5. Is the impact limited only to training, or can we observe any improvement during inference time as well?

The experimental study we conduct shows not only that the natural chat template allows to achieve increased accuracy on benchmarks such as ARC-C (Clark et al., 2018) and GSM8K (Cobbe et al., 2021) but also allows for faster validation loss convergence than models' default templates. Moreover, per-token loss improvements affect not only message-separator tokens but also the wrapped instructions, indicating more efficient learning for the natural template. This efficiency is further reflected in how both approaches update parameters, with the natural template updating fewer parameters and thus being closer to the base model, thereby limiting catastrophic forgetting. Finally, we demonstrate that for proprietary instruct-tuned models, using the natural chat template at inference may yield better performance than using the model's own chat template with which it was trained.

To the best of our knowledge, our work is the first to establish a link between the chosen chat template and associated cross entropy loss, to propose a new format to lower it, as well as to trace this phenomenon through training dynamics and benchmark performance.

2 Related Works

Previous work regarding the impact of chat templates remains sporadic. The most prominent works that address this question do so from the perspective of models' alignment (Wang et al., 2026; Jiang et al., 2025; Lyu et al., 2024; Fogel et al.,

2026; Chang et al., 2026). One direction of work, e.g. explored by Fogel et al., 2026 and Chang et al., 2026, focuses on exploiting chat templates for attack injections: it illustrates that the insertion of a malicious prompt within models' own chat format tokens increases the success rate of attacks. Other set of works (e.g. Jiang et al., 2025 and Lyu et al., 2024) study how the choice of a chat template affects the success rate of adversarial attacks: Lyu et al., 2024 investigate the importance of chat templates for models' alignment and find that fine-tuning with no safety instruction (simpler template) and then using safety instruction at inference appears to improve models' alignment.

Similarly, Yun et al., 2025 demonstrate that simpler prompt formats ameliorate the diversity of generated text when compared to more lengthy formats. While an indication that more stripped-down templates might be beneficial has been established in previous work, our project is the first one to analyze the actual loss impact associated with more complex chat templates.

When studying more complex prompt formats consisting of persona, task instructions, examples, output format instructions, and user task, He et al., 2024 find that wrapping those in JSON is the most favorable strategy compared to using plain text or markdown format. However, the impact of these structural changes has not been studied for simpler prompts only containing the instruction.

More generally, Voronov et al., 2024 indicate that even small formatting changes such as using different separators (`'\n'` vs `' '`) can greatly alter the model's performance. Mizrahi et al., 2024, similarly, highlight sensitivity of models' performance to the way a question is presented: paraphrasing the same question can lead to completely different performance rankings.

Although previous work highlights the influence of chat templates and prompt formats on models' alignment, text generation quality, and overall capabilities, it mainly focuses on the models' outputs. Our research combines these observations with the study of loss evolution as an attempt to interpret this type of behavior.

3 Experiment Design

This section briefly covers the design of the experimentation we conduct, which can be divided into 2 parts: 1) fine-tuning models to follow instructions using models' default templates and natural template we propose; 2) comparative study of default and natural templates for inference using proprietary Instruct models.

3.1 Instruct Tuning

In the first part of our work, to better understand the impact of a chosen template on the final models we instruct-tune models ourselves. We train two type of models: one using the model's default chat template, and another with the natural chat template we propose. We ensure that the order of instructions used for training is fixed across both runs, so that the only difference lies in the applied template.

Models We train Qwen2.5 0.5B (Qwen Team et al., 2025), Gemma 2 2B (Gemma Team et al., 2024), and OLMo 2 1B (Walsh et al., 2025). OLMo, unlike Qwen and Gemma, doesn't use special vocabulary units to mark message boundaries, instead reusing tokens seen during training. This leads to a divergence of the results, which are discussed separately in Section 4.5.

Dataset We use a sample of 176000 instructions from Tülu 3 Mix (Lambert et al., 2025). The dataset contains diverse instructions spanning math, coding, alignment, and multilingual tasks. As Gemma 2 requires a strict order of *user-assistant* messages, we adapted the dataset to these rules, replacing the system message or merging messages as needed. No modifications were made for Qwen and OLMo. If a default chat template includes a system message, it is added to the natural one with **`System:`** as well. The data is divided into effective batches of size 128. In addition, we truncate the texts to a length of 4096.

Training Details We perform full-finetuning of models using one A100 GPU. We employ[3] the learning rate (LR) of $1 \cdot 10^{-6}$ as it has been demonstrated to allow for efficient learning and mitigating catastrophic forgetting (Lin et al., 2026; Wu et al., 2026). We also perform LR sensitivity analysis in Section 4.2.

While some previous work computes loss only on the assistant output (Wang et al., 2023; Touvron et al., 2023), we compute it over the whole input, including the instruction. This approach has been shown to improve model performance (Shi et al., 2024) and provides greater insight into how chat template tokens are learned.

[3]With warmup 0.03 and linear decay to 70% of the LR.

Evaluation For each trained model, we report the length-weighted cross entropy loss to control for length difference caused by different templates. The validation set contains 100 instructions sampled from Tülu Mix 3 that do not appear during training.

We evaluate several checkpoints of trained models on ARC-C (Clark et al., 2018), GSM8K (Cobbe et al., 2021), IFEval (Zhou et al., 2023), and our own SimpleMath benchmark. GSM8K is used in 8-shot settings, while all other benchmarks are used in 0-shot setting[4]. Model responses are extracted via greedy decoding.

ARC-C is a general knowledge MCQ for which we automatically extract answers from the models' responses and perform exact matching with golden labels, and we do similarly for GSM8K (open-ended math questions). For the IFEval (instruction-following benchmark), we use Google's code[5] to automatically evaluate whether the considered models correctly follow the prompt's directions. Finally, for SimpleMath we employ Deepseek R1 (Guo et al., 2025) to extract the models' answers and then match those with golden labels.

Due to the models' small size, the performance they could achieve on a benchmark remains limited. To better understand the impact of training, we introduce our own `SimpleMath` benchmark to test whether the models learn to solve simple sum and subtraction questions, which we expect to be easy for them. We utilize two dynamic question templates:

> Anna has **{a}** oranges and Ben has **{b}** oranges. **{prefix}** the total amount of oranges they have?

> Anna has **{a + b}** oranges. She gave Ben **{b}** oranges. **{prefix}** the total amount of oranges she has left?

We test with `a` $\in [1 \dots 9]$, `b` $\in [1 \dots 9]$, `prefix` $\in$ ['What is', 'How to calculate', 'How to compute', 'How to find', 'What is the best way to find'], making 810 prompts total for `SimpleMath`. This approach allows us not only to calculate response accuracy but also to estimate the standard deviation across different prefixes used to ask the same question.

[4]See Appendix B for more details.

[5]https://github.com/google-research/google-research

3.2 Inference Prompting

In order to assess if the observed phenomena can be generalized to already instruction-tuned models, we perform experiments on prompting a selection of models using both the template the model was trained on and our natural template.

Models We test: 1) larger versions of tested models Gemma 2 9B-it and Qwen2.5 7B-Instruct; 2) another family model Llama-3.1 8B-Instruct (Grattafiori et al., 2024); 3) newer generation models Gemma 3 1B-it (Gemma Team et al., 2025) and Qwen3 4B-Instruct-2507 (Yang et al., 2025) .

Evaluation First, we compare losses when using default and natural templates using the same validation set as in Section 3.1. Due to the black-box nature of the training of selected models, one cannot guarantee whether a model has seen a certain data point in training or not. However, we still consider this approach a perfect litmus test for the template's impact on loss. Furthermore, we evaluate the same models on ARC-C and GSM8K using both the default and natural templates.

4 Instruction-tuning Results

In this section, we dissect the results of instruction-tuning from multiple angles. Section 4.1 demonstrates the impact of the chosen template on the benchmark results achieved. Section 4.2 analyzes the template effect through the convergence of training and validation losses for the chosen chat template. It also provides a sensitivity analysis of the chosen learning rate.

We continue the experiments by investigating the internal processes that take place at this stage. Section 4.3 shows per-token loss after instruction tuning depending on the format of the training data, while Section 4.4 provides an analysis through the lens of parameter updates associated with each chat template. Moreover, we identify an outlier model OLMo 2 which we study in the dedicated Section 4.5.

4.1 Benchmark Performance

To understand the impact of the chosen chat template on performance, we test both versions of the instruct-tuned models in the middle of training (step 625) and at the end of training (step 1375). Each model is evaluated using the template on which it was trained. The obtained results are shown in Table 1.

			ARC-C	GSM8K	IFEval	SimpleMath
Gemma 2 2B	+ *default template*	(Step 625)	0.221	0.297	**0.157**	0.899 (0.063)
	+ *default template*	(Step 1375)	0.245	0.300	**0.157**	0.886 (0.077)
	+ *natural template*	(Step 625)	0.401	**0.318**	0.124	0.949 (0.072)
	+ *natural template*	(Step 1375)	**0.439**	**0.318**	0.120	**0.959** (0.074)
Qwen 2.5 0.5B	+ *default template*	(Step 625)	0.396	0.290	0.105	0.526 (0.247)
	+ *default template*	(Step 1375)	0.411	0.306	0.105	0.467 (0.260)
	+ *natural template*	(Step 625)	0.439	0.377	0.142	0.964 (0.041)
	+ *natural template*	(Step 1375)	**0.445**	**0.381**	**0.144**	**0.981** (0.025)

Table 1: Accuracy obtained by fine-tuned models on evaluation benchmarks. For SimpleMath, we report the average and standard deviation of the accuracies across 5 considered prefixes. Best results are bolded; second-best are underlined.

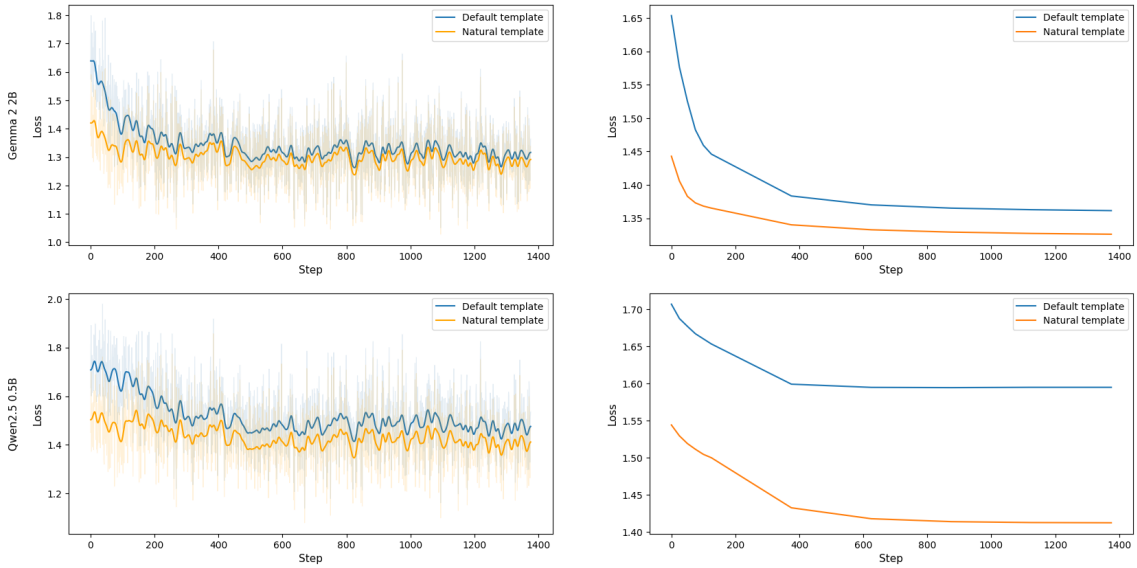

Figure 2: Train (left) and validation (right) cross entropy loss of Gemma 2 2B and Qwen 2 0.5B trained with learning rate $1 \cdot 10^{-6}$. For training loss readability, we applied Gaussian smoothing with sigma 5.

Although each has its own default template, both Qwen and Gemma benefit from replacing it with the natural one: models trained with natural templates consistently outperform the default one on ARC-C and GSM8K. Notably, the natural template for Gemma 2 almost doubles the performance of the default one on ARC-C, which could indicate that those special tokens impede MCQ answering abilities.

On the other hand, instruction following (IFEval) results are not as clear-cut. For Gemma 2 the default template model outperforms the natural one, while we see the inverse trend for Qwen 2.5. Due to the size of the models, the instructions following the model capacities remain quite weak across all setups. For this reason, we suggest further research is needed to identify the impact of special chat tokens on following instructions versus reusing vocabulary as chat separators.

Regarding SimpleMath, we remark not only that the natural template model outperforms the default one but that the default model exhibits a downward trend as training continues. Such deterioration is particularly dramatic for smaller Qwen2.5 0.5B, which can retain almost perfect ability to solve those math problems when trained with the natural template and loses half of these abilities when trained with the default template. This could indicate catastrophic forgetting for such questions due to the special tokens in chat templates. We additionally compute accuracies across different prefixes and report their standard deviation. We note that in addition to the decreasing accuracy, the prefix sensitivity (expressed by the standard deviation) increases for default models as training continues.

4.2 Training and Validation Loss Convergence

However, is the observed performance improvement episodic, or does it stem from better training dynamics? To answer this question, we compare

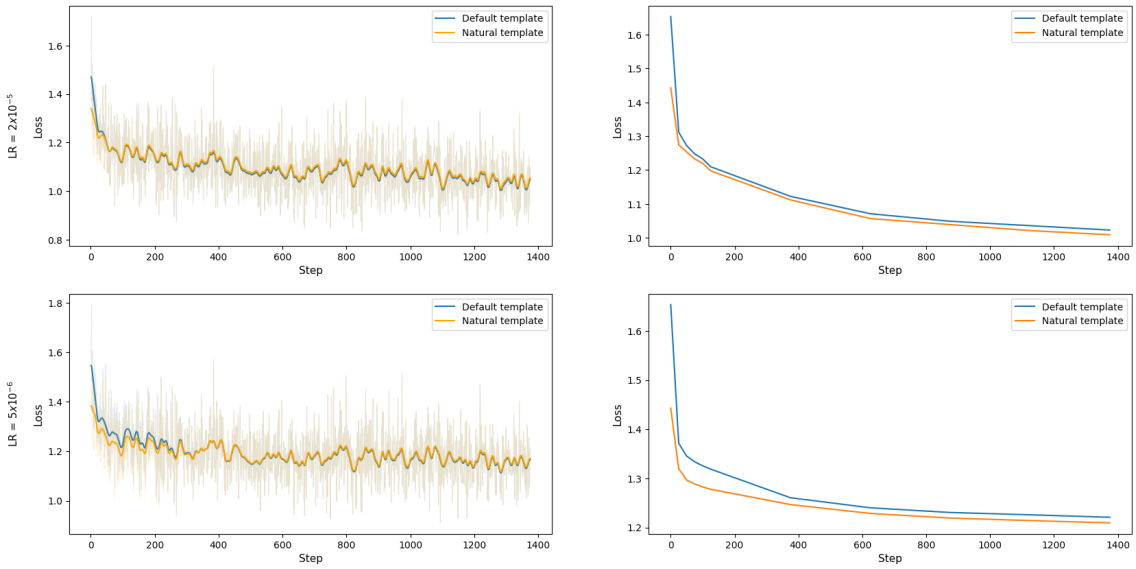

Figure 3: Train (left) and validation (right) losses for Gemma 2 2B trained with learning rates $2 \cdot 10^{-5}$ and $5 \cdot 10^{-6}$.

		Default Template	Natural Template	P-value
Gemma 2 2B	Separator tokens	3.276	2.495	$8.92 \cdot 10^{-11}$
	Other tokens	1.606	1.566	$4.81 \cdot 10^{-10}$
Qwen 2.5 0.5B	Separator tokens	4.312	0.308	$2.15 \cdot 10^{-81}$
	Other tokens	1.926	1.554	$1.85 \cdot 10^{-15}$

Table 2: Average per-token loss on validation set after full training, split by token type. We perform a t-test on losses across different token types for each prompt and report the p-value.

the trends of train and validation losses during training and discover that both Gemma and Qwen families demonstrate faster convergence when using the natural template (as presented in Figure 2). As train losses get quite close for Gemma, we perform the t-test for related samples and find that cross entropy with natural template is still significantly smaller than with the default one (p-value $< 1 \cdot 10^{-10}$).

In order to understand whether the observed phenomenon is solely due to the chosen learning rate, we additionally train Gemma 2 2B with higher learning rates: $2 \cdot 10^{-5}$ and $5 \cdot 10^{-6}$, following Lin et al., 2026. As Figure 3 demonstrates, we find that with higher learning rates, the gap between training losses closes down, and for $LR = 2 \cdot 10^{-5}$, the default template achieves significantly lower results (p-value $< 1 \cdot 10^{-10}$). However, the gap in validation losses between templates persists across all tested learning rates, thus confirming the impact of the chat template.

4.3 Per-token Losses

Given the improved performance of models trained with the natural template and lowered validation loss, we ask whether this is solely due to "easier" (lower-loss) tokens used to delimit message boundaries, or whether this approach also affects the learning of instructions and responses.

For this purpose, we apply both chat templates to the instructions of the validation set and study cross entropy loss in detail, splitting the results by token type: whether it is used as a message separator[6] or contains actual information we want the models to learn. This analysis is applied to the final trained models.

As Table 2 illustrates, the most significant difference between template types in per-token losses after instruct-tuning is indeed on separator tokens. However, models trained with the natural template are also more likely to correctly predict tokens associated with instructions and responses. This can be explained by higher-loss tokens appearing in the context of other tokens that affect their respective loss, a factor that needs to be considered when choosing a chat template.

[6]Refer to Appendix A for further details.

		Total	Embeddings	MLP	Attention
Gemma	+ DT	$1 \cdot 10^8$	$6.2 \cdot 10^6$	$7.6 \cdot 10^7$	$1.2 \cdot 10^7$
	+ NT	$\mathbf{8.2 \cdot 10^7}$	$\mathbf{5.7 \cdot 10^6}$	$\mathbf{6.1 \cdot 10^7}$	$\mathbf{9.4 \cdot 10^6}$
Qwen	+ DT	$1.7 \cdot 10^7$	$3.7 \cdot 10^6$	$8 \cdot 10^6$	$1.4 \cdot 10^6$
	+ NT	$\mathbf{1.4 \cdot 10^7}$	$\mathbf{3.1 \cdot 10^6}$	$\mathbf{6.6 \cdot 10^6}$	$\mathbf{1.1 \cdot 10^6}$
OLMo	+ DT	$1.3 \cdot 10^7$	$\mathbf{2.8 \cdot 10^5}$	$7.6 \cdot 10^6$	$3.6 \cdot 10^6$
	+ NT	$\mathbf{1.2 \cdot 10^7}$	$2.9 \cdot 10^5$	$\mathbf{6.7 \cdot 10^6}$	$\mathbf{3.1 \cdot 10^6}$

Table 3: Number of updated parameters compared to the base model after instruction tuning for Gemma 2 2B, Qwen2.5 0.5B and OLMo 2 1B. We present both total number of updated parameters, as well as number of parameters for different blocks. Lowest number of parameters updated is bolded. DT (resp. NT) stands for default (resp. natural) template.

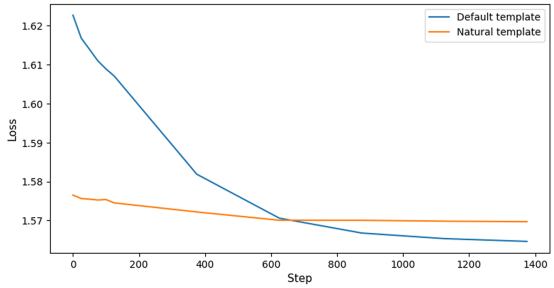

Figure 4: Validation loss for OLMo 2 1B trained with LR = $1 \cdot 10^{-6}$.

4.4 Parameter Updates

As we observe the first signs of catastrophic forgetting for default template models on the SimpleMath benchmark, we turn our attention to parameter updates that occur during fine-tuning.

SFT has been associated with catastrophic forgetting (Chu et al., 2025; Huan et al., 2025; Jin et al., 2025) and inefficient parameter updates (Yuan et al., 2025). The former is also associated with divergence between base and fine-tuned model (Shenfeld et al., 2026). Therefore, to understand the impact of chat templates we compare absolute parameter changes (delta parameters in Yu et al., 2024) between the base and fine-tuned models and report the non-zero ones.

As can be seen in Table 3, training with the default template updates more parameters across all parameter blocks, for both Gemma and Qwen. Since the only difference during training lies in the applied template, the disparity in the number of parameters updated is explained solely by the necessity to update more parameters to appropriately learn distribution of special chat tokens used in the default template. Such inefficiency could be associated with catastrophic forgetting and performance decay we observe in Section 4.1.

4.5 Case of OLMo 2

To highlight the importance of tokenization of chat templates, we study the case of OLMo 2 family that does not introduce separate tokens for chat template and instead uses existing vocabulary: `['<', '|', 'user', '|', '>']`. This separates OLMo from Qwen and Gemma which use special vocabulary for chat template (i.e. `<start_of_turn>` for Gemma or `<|im_start|>` for Qwen are tokenized as single tokens). This reusing makes OLMo's chat template quite similar to our natural one, as it instead employs tokens observed during pre-training.

Figure 4 illustrates that this tokenization particularity can be noticed right away in validation loss convergence: unlike the examples above, the model trained with the default template overtakes the natural one in terms of validation loss. Compared to the natural template we propose, OLMo uses more tokens per message separator, which could affect the convergence of validation loss. Even though OLMo's chat template similarly reuses tokens seen during pre-training, the choice of vocabulary could also impact training dynamics and requires further research.

Additionally, as shown in Table 3, we remark that while the model finetuned with the natural template still updates fewer parameters in total, the situation is reversed for embedding parameters. This could indicate a link between the chosen vocabulary for chat templates and the efficiency of parameter updates, and in particularly, the embedding updates.

5 Inference Prompting Results

Unlike all previous experiments, in this section, we assess models that we did not instruct-tune ourselves. The main goal of this experiment is to understand whether, after complete instruction tuning, the chat template-related loss spikes are alleviated.

5.1 Loss Difference

First, we want to determine whether there is still a decrease in cross entropy loss when applying the natural template even if the model was fully trained using its own default one. For this purpose, we

	DT	NT	P-value
Gemma 2 9B-it	1.845	1.545	$1.03 \cdot 10^{-12}$
Qwen2.5 7B-Instruct	1.418	1.226	$5.03 \cdot 10^{-15}$
Llama-3.1 8B-Instruct	1.562	1.371	$1.81 \cdot 10^{-25}$
Gemma 3 1B-it	2.516	2.046	$1.08 \cdot 10^{-15}$
Qwen 3 4B-Instruct	1.832	1.615	$4.88 \cdot 10^{-19}$

Table 4: Per-token normalized cross entropy loss on validation set. We also report p-value of the t-test computed with non-aggregated per-document losses. DT (resp. NT) stands for default (resp. natural) template.

		ARC-C	GSM8K
Gemma 2 9B-it	+ DT	0.551	0.848
Gemma 2 9B-it	+ NT	**0.891**	**0.861**
Qwen2.5 7B-Instruct	+ DT	**0.900**	**0.854**
Qwen2.5 7B-Instruct	+ NT	0.858	0.847
Llama-3.1 8B-Instruct	+DT	**0.828**	**0.845**
Llama-3.1 8B-Instruct	+NT	0.821	0.820
Gemma 3 1B-it	+DT	0.137	0.041
Gemma 3 1B-it	+NT	**0.391**	**0.312**
Qwen 3 4B-Instruct	+DT	0.864	0.820
Qwen 3 4B-Instruct	+NT	**0.897**	**0.842**

Table 5: Performance of instruct models on benchmarks when prompted with their default template (DT) and the natural template (NT).

use the same validation set as above, applying both models' default templates and the natural one.

As can be observed in Table 4, across different families, sizes and generations, cross entropy loss is statistically significantly lower when the natural template is applied to instructions instead of the template the models were trained on. This is an indication that instruction-tuning (which is frequently followed by a reinforcement learning stage) is not sufficient to eliminate the increased uncertainty associated with special tokens of chat template.

5.2 Performance Difference

Subsequently, our goal is to establish whether choosing to apply the natural template instead of the default one can improve task performance.

Despite decreased loss associated with the natural template, not all models directly benefit from being prompted with the natural chat template, as Table 5 shows. We observe the inverse trend for larger Qwen2.5 7B-Instruct and Llama-3.1 8B-Instruct, which achieve better results when prompted with their respective default templates. However, we cannot conclude that with the size increase the chat template sensitivity is reduced, as Gemma 2 9B-it drastically improves ARC-C performance with the natural template. We observe a similar trend with the newer Gemma 3 1B-it and a small improvement with Qwen 3 4B-Instruct, indicating the same phenomenon across various sizes, model families, and generations.

Although a lower loss related to prompts can serve as one explanation, previous work has suggested that lowering the likelihood of a document being drawn from the fine-tuning distribution allows a decrease in catastrophic forgetting (Kotha et al., 2024). The usage of the natural template at inference of instruct models can be understood as an implementation of this measure.

Additionally, since the evaluated models are blackbox, we cannot assess if the models have been exposed to the benchmarks during training and to what extent. Nevertheless, there is a strong indication that models of various sizes and generations can benefit from being prompted with the natural template to improve the performance of a target task.

6 Conclusion

An instruction-following stage, accompanied by the use of chat templates, remains necessary in the development of modern LLMs. Our work highlights that the selection of chat templates impacts both training dynamics and performance of already instruct-tuned models. We observe that, as many models use special tokens to delimit instruction message boundaries, those tokens are associated with an increased per-token loss for pre-trained models. Moreover, instruct models still exhibit elevated loss regarding those tokens.

We show that we can reduce per-document loss by replacing those special tokens with more natural-sounding ones that are more likely to appear during pre-training. In addition to reducing loss, this format is beneficial for target task performance. This positive effect of the natural chat format can even be observed during inference of models trained to follow instructions with their default chat template.

While the chat template's influence is demonstrable, further research into optimizing this component is required.

Limitations

Model Size Due to computational limits, the paper is focused on fine-tuning only small models. While further research into fine-tuning larger models is required, we consider the inference study to be a strong signal that the chat template sensitivity persists for larger models.

SFT vs RL Our work focuses solely on the SFT stage of instruction-tuning, which is typically followed by a reinforcement learning (RL) phase (Qwen Team et al., 2025; Gemma Team et al., 2024, 2025). We demonstrate that proprietary Instruct models that had an RL phase remain susceptible to higher losses when using chat templates' special tokens and potentially decreased performance during inference. Nevertheless, the impact of each stage needs to be disentangled via further research.

Thinking Models This work is purposely limited to following instructions only; however, chat template tokens are not the only type of special vocabulary used during training. In particular, thinking models (Guo et al., 2025; Yang et al., 2025) employ special `<think>` tokens to wrap their thought process. Further study is likewise required to assess the impact of choosing this type of token on the model's performance and loss convergence.

Multimodal Models To better understand the impact of chat templates, we focused exclusively on text-only models. Nonetheless, modern generations of LLMs (Gemma Team et al., 2025; Qwen Team, 2026; Kimi Team et al., 2026) are trained in a multi-modal setting. It is important to consider how the effect of the chat template tokens evolves across different modalities.

Optimization of Chat Template While our work demonstrates that the natural chat template can be beneficial for models' performance and loss convergence, further work is needed to understand which format is the most suitable for instruction wrapping. Following the suite of Sclar et al., 2024, the quantification of LLMs' sensitivity to chat template features such as capitalization, new line symbols, separators' length, or vocabulary choice, and others, is required, as chat templates remain central for prompting modern LLMs.

References

Hwan Chang, Yonghyun Jun, and Hwanhee Lee. 2026. ChatInject: Abusing chat templates for prompt injection in LLM agents. In *ICLR*.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *ICML*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. In *arXiv*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. In *arXiv*.

Ariel Fogel, Omer Hofman, Eilon Cohen, and Roman Vainshtein. 2026. Inference-time backdoors via hidden instructions in LLM chat templates. In *ICLR 2026 Workshop on Principled Design for Trustworthy AI - Interpretability, Robustness, and Safety across Modalities*.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. In *arXiv*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. In *arXiv*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. In *arXiv*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. In *arXiv*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. In *Nature*.

Jia He, Mukund Rungta, David Koleczek, Arshdeep Sekhon, Franklin X Wang, and Sadid Hasan. 2024. Does prompt formatting have any impact on LLM performance? In *arXiv*.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. 2025. Does math reasoning improve general LLM capabilities? understanding transferability of LLM reasoning. In *CoRR*.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. 2025. ChatBug: A common vulnerability of aligned LLMs induced by chat templates. In *AAAI*.

Hangzhan Jin, Sicheng Lv, Sifan Wu, and Mohammad Hamdaqa. 2025. RL is neither a panacea nor a mirage: Understanding supervised vs. reinforcement learning fine-tuning for LLMs. *arXiv*.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, and 181 others. 2026. Kimi K2: Open agentic intelligence. In *arXiv*.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. 2024. Understanding catastrophic forgetting in language models via implicit inference. In *ICLR*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. In *COLM*.

Jiacheng Lin, Zhongruo Wang, Kun Qian, Tian Wang, Arvind Srinivasan, Hansi Zeng, Ruochen Jiao, Xie Zhou, Jiri Gesi, Dakuo Wang, Yufan Guo, Kai Zhong, Weiqi Zhang, sujay sanghavi, Changyou Chen, Hyokun Yun, and Lihong Li. 2026. SFT doesn't always hurt general capabilities: Revisiting domain-specific fine-tuning in LLMs. In *ICLR*.

Kaifeng Lyu, Haoyu Zhao, Xinran Gu, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. 2024. Keeping LLMs aligned after fine-tuning: The crucial role of prompt templates. In *NeurIPS*.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? A call for multi-prompt LLM evaluation. *TACL*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Qwen Team. 2026. Qwen3.5-omni technical report.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. In *arXiv*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *ICLR*.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2026. RL's razor: Why online reinforcement learning forgets less. In *ICLR*.

Zhengyan Shi, Adam X Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. 2024. Instruction tuning with loss over instructions. In *NeurIPS*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. In *arXiv*.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. In *ACL: Findings*.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, and 23 others. 2025. 2 OLMo 2 furious (COLM's version). In *COLM*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. In *NeurIPS*.

Zihan Wang, Hongwei Li, Rui Zhang, Wenbo Jiang, and Guowen Xu. 2026. Badtemplate: A training-free backdoor attack via chat template against large language models. In *arXiv*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *ICLR*.

Chao-Chung Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, Shao-Hua Sun, and Hung yi Lee. 2026. Mitigating forgetting in LLM fine-tuning via low-perplexity token learning. In *NeurIPS*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *ICLR*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. In *arXiv*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. In *ICLR*.

Xiangchi Yuan, Xiang Chen, Tong Yu, Dachuan Shi, Can Jin, Wenke Lee, and Saayan Mitra. 2025. Mitigating forgetting between supervised and reinforcement learning yields stronger reasoners. In *CoRR*.

Longfei Yun, Chenyang An, Zilong Wang, Letian Peng, and Jingbo Shang. 2025. The price of format: Diversity collapse in LLMs. In *EMNLP: Findings*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for Large Language Models. In *arXiv*.

A Chat Template Examples

In this section, we provide examples of how the same instruction set is converted into text by applying default chat templates of different families, as well as the natural chat template. Message separators are bolded. All special tokens (i.e. parts of the separator tokenized as single token that are less likely to appear during pre-training) are listed.

Instruction

```
[
{'role': 'user', 'content': 'Write a poem' },
{'role': 'assistant', 'content': 'Sure!' }
]
```

A.1 Qwen2.5 Family

When applying the chat template, Qwen family adds an additional system message, not contained in the actual instruction. The system message is a general-level instruction added before each prompt containing an instruction and its content defined by Qwen itself. We replicate this message as well (see Section A.6) Special tokens: **<|im_start|>**, **|im_start|>**.

Application of Qwen2.5 Default Template

'**<|im_start|>system**
You are a helpful assistant.**<|im_end|>**
<|im_start|>user
Write a poem**<|im_end|>**
<|im_start|>assistant
Sure!**<|im_end|>**'

A.2 Qwen 3 Family

Unlike the tokenizer of Qwen 2.5 family, Qwen 3 does not automatically add a system message. Special tokens: **<|im_start|>**, **|im_start|>**.

Application of Qwen3 Default Template

'**<|im_start|>user**
Write a poem**<|im_end|>**
<|im_start|>assistant
Sure!**<|im_end|>**'

A.3 Gemma 2 & 3 Family

The chat template is identical for the tested models of both families. Special tokens:

<start_of_turn>, <end_of_turn>

Application of Gemma 2&3 Default Template

'<start_of_turn>user
Write a poem<end_of_turn>
<start_of_turn>model
Sure!<end_of_turn>'

A.4 OLMo 2 Family

Special tokens: N/A

Application of OLMo 2 Default Template

<|user|>
Write a poem
<|assistant|>
Sure!<|endoftext|>

A.5 Llama 3.1 Family

Llama 3.1 tokenizer adds an automatic system message saying "Cutting Knowledge Date: December 2023 Today Date: 26 Jul 2024" when applying its chat template. Special tokens: <|start_header_id|>, <|end_header_id|>.

Application of Llama 3.1 Default Template

<|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date: December 2023 Today Date: 26 Jul 2024 <|eot_id|>
<|start_header_id|>user<|end_header_id|>

Write a poem<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
Sure!<|eot_id|>

A.6 Natural Template

Special tokens: N/A

Application of Natural Template For Qwen2.5

System:
You are a helpful assistant.
User:
Write a poem
Assistant:
Sure!

Application of Natural Template For Other Models

Assistant:
Write a poem
User:
Sure!

B Benchmarks Prompt Examples

In this section, we provide examples of instructions for each benchmark type to which we then apply a chat template. For all formats, we intend the continuation of the last message by the model.

B.1 ARC-C

We use prompts from Meta's Llama-3.1-8B-Instruct evaluation[7]. The prompts already have pre-applied Llama's default chat template. In order to be able to use them with other models we re-convert them back to instructions. The benchmark is used in 0-shot setting.

ARC-C Instruction

[{ 'role': 'user',
'content': 'Given the following question and four candidate answers (A, B, C and D), choose the best answer.
Question: <question>
A. <option A>
B. <option B>
C. <option C>
D. <option D>
Your response should end with "The best answer is [the_answer_letter]" where the [the_answer_letter] is one of A, B, C or D.'
},
{ 'role': 'assistant', 'content': '' }]

B.2 GSM8K

Similarly to ARC-C, we reuse Llama-3.1-8B-Instruct evaluation prompts by first converting them back to instruction. The benchmark is used in 8-shot setting (prompt contains 8 pairs of example exchange between the user and assistant containing correct responses).

GSM8K Instruction

[{ 'role': 'user',
'content': 'Given the following problem, reason and give a final answer to the problem.
Problem: <Problem>
Your response should end with T̈he final answer is [answer]ẅhere [answer] is the response to the problem.'
},
{ 'role': 'assistant', 'content': '' }]

B.3 IFEval & SimpleMath

As IFEval[8] tests instruction following, the prompt does not contain any additional instructions. We use the same logic for SimpleMath.

IFEval Instruction

[{ 'role': 'user',
'content': <Instruction/Question>
},
{ 'role': 'assistant', 'content': '' }]

[7] https://huggingface.co/datasets/meta-llama/Llama-3.1-8B-Instruct-evals/

[8] https://huggingface.co/datasets/google/IFEval