Instruction Fine-Tuning: Does Prompt Loss Matter?

Mathew Huerta-Enochian[1]* **and Seung Yong Ko**[2]
EQ4ALL
11 Nonhyeon-ro 76-gil, Gangnam-gu, Seoul
[1]mhuerate@uoregon.edu
[2]stephenko@eq4all.co.kr

Abstract

We present a novel study analyzing the effects of various prompt loss token weights (PLW) for supervised instruction fine-tuning (SIFT). While prompt-masking (PLW $= 0$) is common for SIFT, some fine-tuning APIs support fractional PLWs and suggest that using a small non-zero PLW can help stabilize learning when fine-tuning on short-completion data. However, there has never been a study confirming this claim, and OpenAI, a major cloud-based SIFT provider, recently removed this parameter from their fine-tuning API. We found that performance of models fine-tuned on short-completion data had a statistically-significant negative quadratic relationship with PLW. Using small values ($0.01 - 0.5$) of PLW produced better results on multiple-choice and short-generation benchmarks (outperforming models fine-tuned on long-completion data) while large values (≈ 1.0) of PLW produced better results on long-generation benchmarks. We explained this effect and verified its importance through additional experiments. This research serves as a warning to API providers about the importance of providing a PLW parameter for SIFT.

1 Introduction

Recent research in language modeling has made huge advances in training instruction-following agents. Both supervised fine-tuning (SFT) and reinforcement learning (RL) have been employed to much success. However, our understanding of optimal hyperparameters and standards of practice (SOPs) have been slow to catch up. This research contributes to supervised instruction fine-tuning (SIFT) SOPs via an in-depth analysis of a single training hyperparameter: prompt loss weight (PLW).

While training, model parameters are updated by optimizing for next-token maximal likelihood classification. Most open sourced solutions for SIFT either mask the prompt loss (for prefix language modeling) or use the entire sequence loss (for full language modeling) while some API providers support an explicit PLW parameter that allows users to apply fractional PLW during SIFT. The commonly-held notion is that fractional PLW helps stabilize learning when fine-tuning on data with short outputs. Recently, however, OpenAI quietly removed support for their `prompt_loss_weight` parameter.

The reason for the removal is unknown. Furthermore, to our knowledge, there has never been a proper study on the effects of PLW.[1]

We make the following contributions:

- We showed that PLW has a significant relationship with model performance when fine-tuning on *short-completion* data.
- We showed that this relationship is due to a combination of regularizing effects and not the accepted explanation of increased training stability.
- We provided evidence that PLW cannot be replaced by other common regularizers alone and that PLW is important for strong performance on short-generation downstream tasks.
- We verified that PLW can be safely ignored when fine-tuning on *long-completion* data.

We provide relevant background and hypotheses in section 2 and 3, respectively. The main regression experiment is presented in sections 4 and 5. Supplemental experiments further validating our claims are presented in section 6. We present conclusions in section 7 followed by several appendices for additional analysis and discussion.

*Currently affiliated with the University of Oregon, research conducted while working at EQ4ALL.

[1]Note that this research used a general implementation of PLW. OpenAI's former `prompt_loss_weight` implementation could not be tested directly.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 22771–22795
November 12-16, 2024

2 Background

2.1 Definitions

We define *instruction data* as one or many instances of structured text data, each containing an instruction, an optional input, and a target output text. We will use the term *prompt* to refer to the concatenation of the instruction and input (if it exists) and the term *completion* to refer to the target output. The goal of SIFT is to fine-tune a model to generate an appropriate completion for a given prompt.

We define the *generation ratio* R_g as the ratio of completion length to prompt length (also referred to as the completion-prompt ratio). We then divide instruction data into two broad categories. Data with $R_g < 1$ are *short-completion* data, and data with $R_g \geq 1$ are *long-completion* data. When applied to an entire dataset, we take R_g to be the mean completion-prompt ratio.

2.2 Relevant Research and Libraries

HuggingFace's Transformers library (Wolf et al., 2020), the de facto library for training LLMs, allows users to mask select tokens when calculating token classification loss. In Transformers, weights for next-token prediction loss is therefore binary—either token loss is masked (PLW $= 0$) or it is unmasked (PLW $= 1$).

As mentioned in section 1, OpenAI officially removed support for a `prompt_loss_weight` parameter in their fine-tuning API as part of the v1 fine_tune API deprecation in early January, 2024. This `prompt_loss_weight` parameter used a default value of 0.01 with the following parameter explanation: *"This controls how much the model tries to learn to generate the prompt (as compared to the completion which always has a weight of 1.0), and can add a stabilizing effect to training when completions are short. If prompts are extremely long (relative to completions), it may make sense to reduce this weight so as to avoid over-prioritizing learning the prompt."*

Though we could not find a study validating OpenAI's claim or any literature that presents an analysis of PLW, we found several studies that reported using this parameter. Though they do not provide their reasoning, Kozachek (2023) reported that they fine-tuned GPT-3 with a `prompt_loss_weight` of 0.1. Dodgson et al. (2023) reported using the default value of 0.01 when fine-tuning GPT models. Wang et al. (2023b) reported that a PLW of 0 performed best for them when working on the Self-Instruct framework. Interestingly, Wutschitz et al. (2023) reported hyperparameter search results for next-sentence-prediction on Elsevier data using PLWs of 0.1 and 0.5 and found 0.5 to give the best results. Similar to OpenAI's deprecated API, BLoomAI's API supports a `prompt_loss_weight` parameter with a default value of 0.01.

3 Hypotheses

Based on OpenAI's explanation for `prompt_loss_weight`, we expected that for SIFT with short-completion data and small values of PLW, there would be a positive relationship between PLW and downstream performance. However, training a model to maximize next-token-prediction on prompt tokens should be most useful for generating instruction data, and over-prioritizing prompt token loss should have a negative influence on downstream performance.

Based on these assumptions, we would expect the two competing factors to result in a downward curved relationship between PLW and downstream performance. Limiting PLW to the range of [0, 1], we postulate that there is a critical value λ for PLW with $0 <= \lambda <= 1$. For PLW less than λ, the positive effect dominates the negative effect and for values greater than λ, the negative effect dominates the positive effect. If $\lambda = 0$, then PLW's contribution to model performance is strictly negative, and if $\lambda = 1$, then PLW contributes strictly positively to model performance. Note that λ would not be an intrinsic characteristic of the dataset, model architecture, or training algorithm. Rather, it would depend on numerous factors and change for each task.

We then made two hypotheses to test the above relationship and performed regression analysis on three fine-tuning datasets spanning a range of R_g values: 0.08, 3.27, and 7.83.

Null Hypothesis ($\mathbf{H_0}$) *Prompt loss weight has no relationship with model performance.*

Alternative ($\mathbf{H_1}$) *Prompt loss weight has a quadratic relationship with model performance.*

We used the standard $\alpha = 0.05$ significance level, and we expected to reject $\mathbf{H_0}$ only for models trained on short-completion data.

4 Methodology

To evaluate the effect of PLW on downstream performance, we used a factorial design methodology and repeated the Alpaca experiment (Taori et al.,

Dataset	Mean (Std) Tokens			Total Tokens	R_g
	Instruction	Input	Completion		
AlpacaData	13.40 (4.97)	6.25 (14.38)	64.51 (64.85)	4,376,482	3.27
AlpacaDataCleaned	14.21 (9.95)	6.42 (17.65)	162.74 (150.89)	9,490,837	7.83
AlpacaDataShort	16.93 (13.10)	162.34 (151.69)	14.62 (10.99)	10,035,667	0.08
UltraFeedback*	184.47 (268.04)	0 (-)	327.37 (291.48)	31,075,393	1.77
DatabricksDolly*	18.65 (74.55)	91.60 (250.66)	91.14 (149.15)	3,023,113	0.83
UltraFeedbackShort*	94.09 (141.94)	206.18 (211.03)	55.03 (61.00)	16,351,333	0.18
DatabricksDollyShort*	18.83 (75.29)	160.37 (270.17)	28.79 (47.49)	3,122,209	0.16

Table 1: Dataset statistics: mean tokenized instruction, input, and completion sequence lengths (standard deviations in parentheses), total token counts for each dataset, and the generation ratio R_g.
* Used for supplemental experiments in section 6.

2023) with three experimental variables. We tested ten discrete levels of PLW, two pre-trained language models (PTLMs), and three instruction fine-tuning datasets for a total of sixty experimental training runs and evaluated each run on thirteen benchmarks.

We used the original Alpaca code and Transformers library, only modified to add PLW. Training was performed exactly as per the original Alpaca experiment, and we used the hyperparameters suggested by the authors, modifying only the three experimental parameters (PLW, PTLM, dataset) with each run.

We provide additional details for reproducibility in appendix E and will release our trained models on HuggingFace's Hub.

4.1 Prompt Loss Weight

We limited our evaluation of PLW to factors in the range [0, 1], focusing on values close to zero:

$$\begin{aligned}\text{PLW} \in \{&0.0, 5\times10^{-4}, 2.236\times10^{-3},\\ &1\times10^{-2}, 2.463\times10^{-2}, 5\times10^{-2},\\ &1\times10^{-1}, 2.463\times10^{-1}, 5\times10^{-1}, 1.0\}\end{aligned}$$

Note that PLW $= 0.0$ is identical to the masking used in the original Alpaca project, and PLW $= 1.0$ is equivalent to unmasked training.

For all analysis, we transformed our PLW values to be closer to uniform on the interval [0, 1] using a power function

$$f: \begin{cases}[0,1] \to [0,1],\\ v \mapsto v^p\end{cases}$$

where the power $p = 0.30103$ was chosen semi-arbitrarily such that $f(0.1) = 0.5$. We denote the transformed PLW values as $\mathsf{w_p}$

4.2 Pre-Trained Language Model

We fine-tune both LLaMA 1 7B (Touvron et al., 2023a) to recreate the original Alpaca experiment and LLaMA 2 7B (Touvron et al., 2023b) to provide more relevant results.

4.3 Fine-Tuning Dataset

We ran all experiments with three datasets: AlpacaData (the instruction dataset from the original Alpaca experiment), AlpacaDataCleaned (Ruebsamen, 2023), and AlpacaDataShort.

AlpacaDataCleaned is a cleaned and curated version of AlpacaData that has recently been combined with data from the GPT4 LLM dataset (Peng et al., 2023). Cleaning is noted as ongoing and includes fixes for the following issues in AlpacaData: hallucinations, merged instructions, empty outputs, empty code examples, instructions to generate images, N/A outputs, inconsistent input fields, wrong answers, nonsensical instructions, and extraneous control characters.

We generated AlpacaDataShort from AlpacaDataCleaned by rephrasing long-completion instances as prompt-prediction task, a process we denote as *prompt inversion*. See Appendix A for more on prompt inversion.

Descriptive statistics for these each dataset are presented in table 1. Note that AlpacaDataCleaned is strongly long-completion with an R_g of 7.83 while AlpacaDataShort is short-completion with an R_g of 0.082.

4.4 Performance Evaluation

We evaluated each model on thirteen instruction benchmarks covering multiple choice and text generation tasks. We selected benchmarks that were relatively cheap to compute and covered a range of tasks. We used three evaluation frameworks: EleutherAI's Language Model Evaluation Harness

Task	V.	Shots	Split	Type
ARC Challenge*	0	25	Test	MC
PIQA	0	0	Val	MC
TruthfulQA-MC2*	1	6†	Val	MC
WinoGrande*	0	5	Val	MC
TruthfulQA-Gen	1	6†	Val	G_S
WMT14 En→Fr	1	0	Val+Test	G_S
WMT14 Fr→En	1	0	Val+Test	G_S
WMT16 En→De	1	0	Val+Test	G_S
WMT16 De→En	1	0	Val+Test	G_S
WMT16 En→Ro	1	0	Val+Test	G_S
WMT16 Ro→En	1	0	Val+Test	G_S
AlpacaEval (Mixtral)	1	1	Test	G_L
PandaLM	1	0	Test	G_L

Table 2: Evaluation benchmarks. Validation splits were used when test splits were unavailable, and validation and test splits were combined for noisy benchmarks. Benchmark completion type is noted here as "MC" for multiple choice, "G_S" for short generation, and "G_L" for long-generation.
*Task used in HuggingFace's Open LLM Leaderboard.
†This benchmark was calculated with `num_fewshot` = 0 but uses a built-in minimum of 6 shots.

(EEH) (Gao et al., 2023b), AlpacaEval 1 (Li et al., 2023), and PandaLM (Wang et al., 2023a, 2024). See table 2 for details on benchmark tasks.

Eleven benchmarks were run using EEH. Four of these were multiple choice tasks: ARC Challenge (Clark et al., 2018), PIQA (Bisk et al., 2020), TruthfulQA-MC2 (Lin et al., 2022), and WinoGrande (Sakaguchi et al., 2019). Seven were short generation tasks: TruthfulQA-Gen (Lin et al., 2022) and six WMT14 and WMT16 translation benchmarks (Bojar et al., 2014, 2016), limited to four languages the PTLMs saw during pretraining (English, French, German, and Romanian). For WMT benchmarks, we used the zero-shot instruction "`Translate the following from <src_lang> to <tgt_lang>`" and evaluated over both the validation and test sets to reduce variance.

Long-generation performance was evaluated using AlpacaEval 1 and PandaLM, which are both LLM-as-a-judge frameworks. The default auto-evaluator for AlpacaEval 1 is GPT4, but using paid APIs would beyond the scope of this research, so we used Mixtral 8X7B (Jiang et al., 2024) as an auto-evaluator. Mixtral performed the best of all open-source LLMs that we tested on AlpacaEval's evaluator test dataset (Dubois et al., 2023), with 64.9% agreement with human evaluators. For reference, Claude (Anthropic, 2023) has 65.3% and GPT4 has 70.99% human agreement.

5 Results and Discussion

A visualization of the simple and min-max scaled relative performance aggregates by w_p and dataset is presented in figure 1. Note that all analysis was performed using the relative aggregate. The simple aggregate is dominated by large performance changes on a few benchmarks and is included only for completeness.

5.1 Performance Trends

There are several qualitative performance trends that are of interest. For more thorough discussion and task-specific benchmark plots, see Appendix B.

A group of four benchmarks (Arc Challenge, PIQA, TruthfulQA-Gen, and WinoGrande) clearly show the expected negative quadratic relationship between w_p and performance of AlpacaDataShort models, with optimal PLW somewhere in $0 < w_p < 1$. Notably, AlpacaDataShort models outperform AlpacaData and AlpacaDataCleaned on this group given optimal PLW tuning. On the long-generation benchmarks, however, AlpacaDataShort models show a steadily-increasing trend, with optimal PLW near $w_p = 1$. For these seven benchmarks, AlpacaDataShort models fine-tuned with prompt loss masking ($w_p = 0$) almost always produced the worst scores.

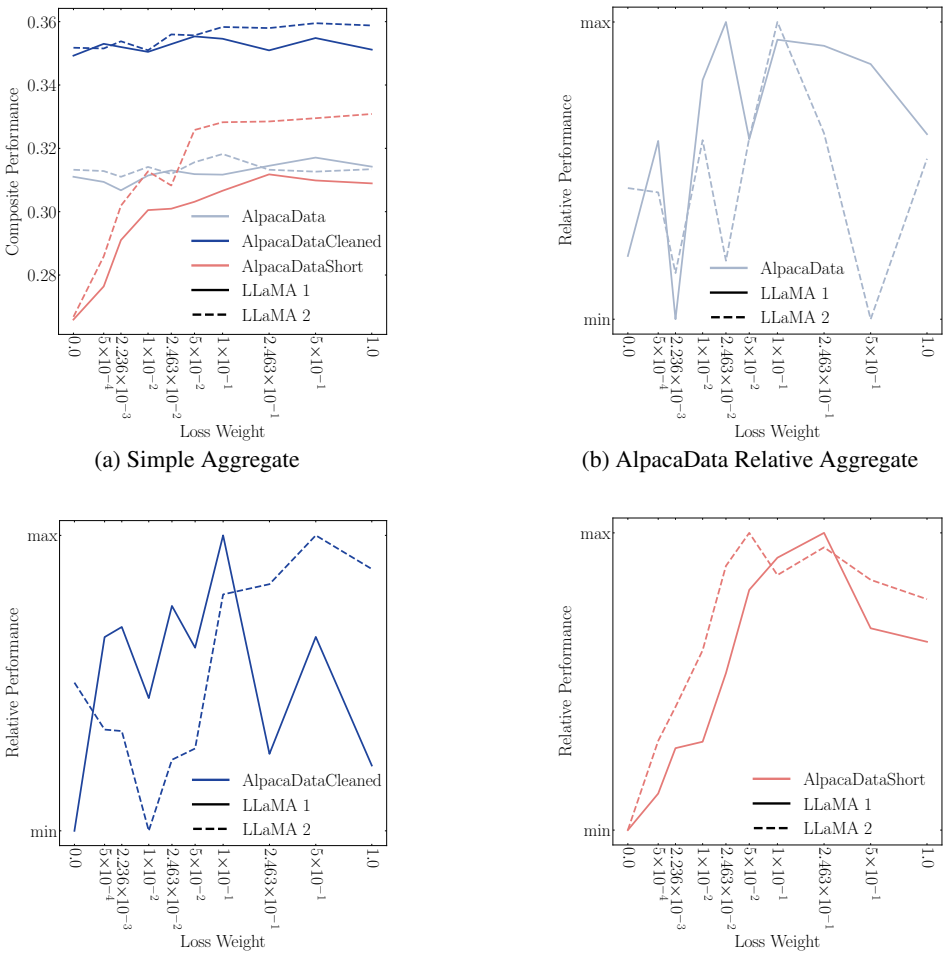
Maximal w_p-based performance increase was around twenty percentage points for both long-generation benchmarks and less than two percentage points for short-generation and multiple choice benchmarks. This difference in scale and the above difference in optimal PLW shows that the relationship between PLW and model performance is strongly dependent on the benchmark task.

Clear qualitative performance trends for the AlpacaData- and AlpacaDataCleaned-trained models and for any model evaluated on the six translation benchmarks could not be identified.

5.2 Regression

For each data group, we fit a generalized linear mixed model (GLMM) with the relative aggregate benchmark scores as the response variable.

We expected a quadratic relationship between the score and w_p, so we included a second order polynomial of w_p as a fixed effect. Furthermore, we knew that the PLW-performance relationship varies by benchmark and since scores were min-max normalized over each benchmark, we used a random slope (and no intercept) with respect to



(a) Simple Aggregate

(b) AlpacaData Relative Aggregate

(c) AlpacaDataClean Relative Aggregate

(d) AlpacaDataShort Relative Aggregate

Figure 1: Performance by transformed PLW. **(a)** A simple performance aggregate score (the unweighted mean of benchmark scores). **(b)**, **(c)**, **(d)** Relative aggregate performance scores where scores per task for each task and group are min-max scaled to show common trends, regardless of scale. Note that aggregate scores for only the AlpacaDataShort models show a relationship with transformed PLW. Best viewed in color.

benchmark. Since we did not min-max normalize over PTLM groups and since we saw consistent improvement when using LLaMA 2, we modeled a random intercept for PTLM. This resulted in the following equation that we fit with the **R** library `glmmTMB`:

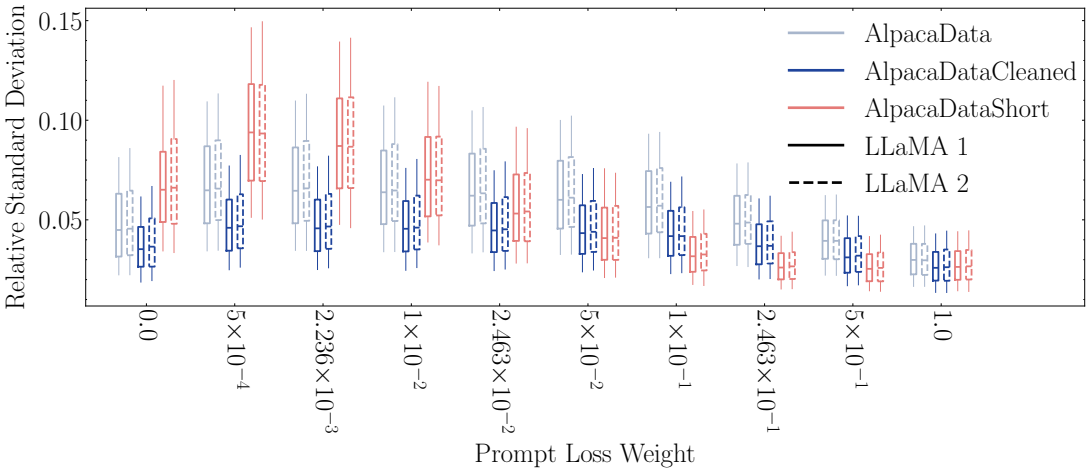
$$\texttt{score} \sim \texttt{pol(w}_\texttt{p}\texttt{,2)} + (0+\texttt{pol(w}_\texttt{p}\texttt{,2)}|\texttt{b}) + (1|\texttt{m})$$

where `score` is the min-max transformed scores, `b` is the benchmark task factor, and `m` is the PTLM factor. Since `score` is bounded and thus introduced heteroskedasticity, we used a beta distribution as the conditional distribution of the response variable. Model fit was evaluated with the `DHARMa` library and `glmmTMB`'s `Anova` method.

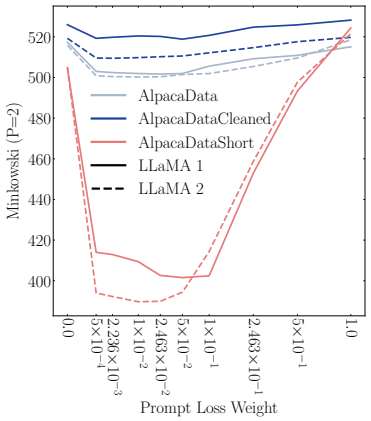
P-values and coefficients are presented in table 3. Regression on both AlpacaData and AlpacaDataCleaned produced convergence warnings and appropriate models could not be adequately fit. We tried reducing the complexity of the model, but no significant relationship with $\texttt{w}_\texttt{p}$ could be found. However, the model fit on AlpacaDataShort converged and passed residual normallity, homoscedasticity, and other checks for soundness. For the AlpacaDataShort case, min-max transformed performance showed a statistically significant negative quadratic relationship with $\texttt{w}_\texttt{p}$ at our target

		Coeff		
	P-Value	$\texttt{w}_\texttt{p}$	$\texttt{w}_\texttt{p}^2$	(Int)
AlpacaData	0.237	1.185	-0.917	(-0.131)
AlpacaDataCleaned	0.0861	1.238	-0.812	(-0.231)
AlpacaDataShort	**<0.001**	**5.590**	**-4.284**	(-1.043)

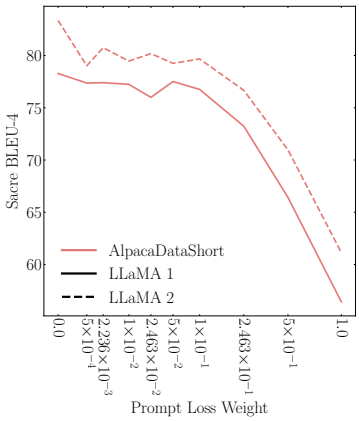
Table 3: $\texttt{w}_\texttt{p}$ p-values and coefficients by training dataset. Statistically significant results are in **bold**. Note that though convergence warnings were raised for regression on both AlpacaData and AlpacaDataCleaned, coefficient and p-value scores are reported for completeness.



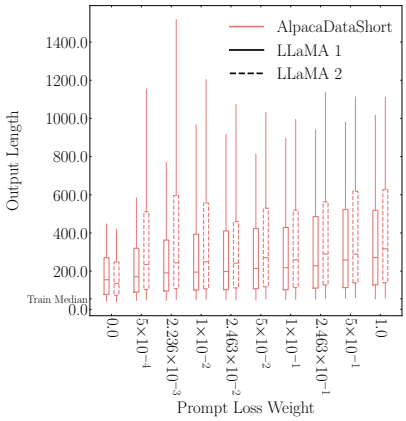
(a) Training Loss Stability



(b) Weight Distance



(c) Train Data Memorization



(d) AE Generation Length

Figure 2: Analysis of causal mechanism. Boxplots use the 0.25, 0.5, and 0.75 quantiles with whiskers at 0.09 and 0.91 quantiles. Best viewed in color. **(a) Training Loss Stability**: Relative Standard Deviation (RSD) of five-step training loss windows show increase instability for small (non-zero) PLWs. **(b) Weight Distance**: Distance between learned weights and PTLM weights is smaller for small (non-zero) PLWs. **(c) Train Data Memorization**: Completion Sacre BLEU scores on training data prompts as an indicator for overfitting. **(d) AE Generation Length**: Generation lengths on the Alpaca Eval test set for varying PLW values.

$\alpha = 0.05$ significance level.

This means that while we could not reject the null hypothesis for the AlpacaData and AlpacaDataCleaned scenarios, for the AlpacaDataShort scenario, there was sufficient evidence to reject the null hypothesis in favor of the alternative hypothesis H_1.

Using the fixed effect coefficients, we can predict the critical PLW value λ for AlpacaDataShort fine-tuning that maximizes the min-max transformed benchmark scores. The coefficients for $\mathtt{w_p}^2$, $\mathtt{w_p}$, and the intercept were -4.28, 5.590, -1.043, respectively. We can rewrite this relationship as:

$$\mathtt{score} = -4.284(\mathtt{w_p} - 0.652)^2 + 0.781,$$

which has a global maximum at $\mathtt{w_p} = 0.652$. Reversing the power transformation yields a critical value for PLW at $\lambda = 0.242$. We verified that this predicted λ overlaps with the visualized maximum value range for the relative aggregate in figure 1d.

5.3 Causal Mechanism & Interpretation

5.3.1 Training Stability

To identify possible causal mechanisms, we first investigated the effects of PLW on training stability by analyzing training loss relative standard deviation (RSD) over five-step windows. See figure 2a for a boxplot of mean RSD for each model. For all dataset and PTLM factors, increasing PLW from zero led to a sharp increase in mean RSD and then a slow decrease to a minimum mean RSD at PLW $= 1$. There is no obvious explanation for why training loss RSD would increase for small PLW before decreasing for large PLW.

If training loss stability was the primary factor in

improved performance, we would expect RSD to be lowest for PLW between 0.01 and 0.5 (or even between 0.01 and 0.1 based on short-generation benchmarks) and for performance at PLW $= 0$ to be similar with performance at PLW $= 0.01$ since mean RSD at these values are similar. However, mean RSD drops by a factor of two across the PLW $\in [0.01, 0.1]$ range, and performance at PLW $= 0.01$ is significantly higher than the masked prompt loss scenario. Training loss mean RSD is lowest at PLW $= 1$, but performance on ARC Challenge, PIQA, WinoGrande, and TruthfulQA-Gen show clear decreasing trends at this value. Furthermore, the three tasks showing positive trends at PLW $= 1$ cannot be adequately explained by this factor since performance increases regardless of loss stability.

There is likely either a tradeoff between training loss stability and some other factors that affects model performance or model loss stability is not an important factor. Considering that both AlpacaData and AlpacaDataCleaned models also showed a negative quadratic trend for mean loss RSD, we tentatively concluded that loss stability is not the driving factor for the modeled relationship.

5.3.2 Weight Regularization

We then checked if PLW was providing regularization to the weight update step, possibly improving performance by keeping weights close to the PTLM. See figure 2b for a visualization of weight distance from PTLM. Interestingly, for AlpacaDataShort, fine-tuned weights were closer to those of the PTLM for small values of PLW but were much farther for PLW < 0.0005 and PLW > 0.1. We would expect weights to change more when loss is erratic, but the range of PLW values that better preserved PTLM weights was similar to the range of increased training loss RSD. This is an interesting result, and we conclude that PTLM model weights were better preserved for small non-zero PLW *despite* high loss instability.

5.3.3 Data Memorization

We next explored how PLW affected training data memorization. We sampled 10,000 unique prompts from the AlpacaDataShort training set, generated completions from each prompt, and calculated corpus BLEU-4 scores. We found that for PLW from 0.0 to around 0.1, models memorized most of the training data, consistently scoring near 80 corpus BLEU. Corpus BLEU then decreased as PLW increased from 0.1. We also analyzed generation length on the AlpacaEval 1 test set, which showed a generally increasing trend with PLW.[2] Since AlpacaDataShort is dominated by short-completion instances, we concluded that non-zero PLW decreases overfitting by allowing the model to learn generation patterns from the prompt without negatively impacting instruction-completion alignment.

5.3.4 Interpretation

Based on the above analysis, we suggest that the causal mechanisms between PLW and downstream performance of models fine-tuned on short-completion data are

1. preservation of PTLM weights for small PLW and
2. reduced overfitting (and increased generation length) for large PLW.

A tradeoff between these two mechanisms would explain the positive trend seen in the AlpacaEval and PandaLM benchmarks and the negative quadratic relationship in several of the other benchmarks.

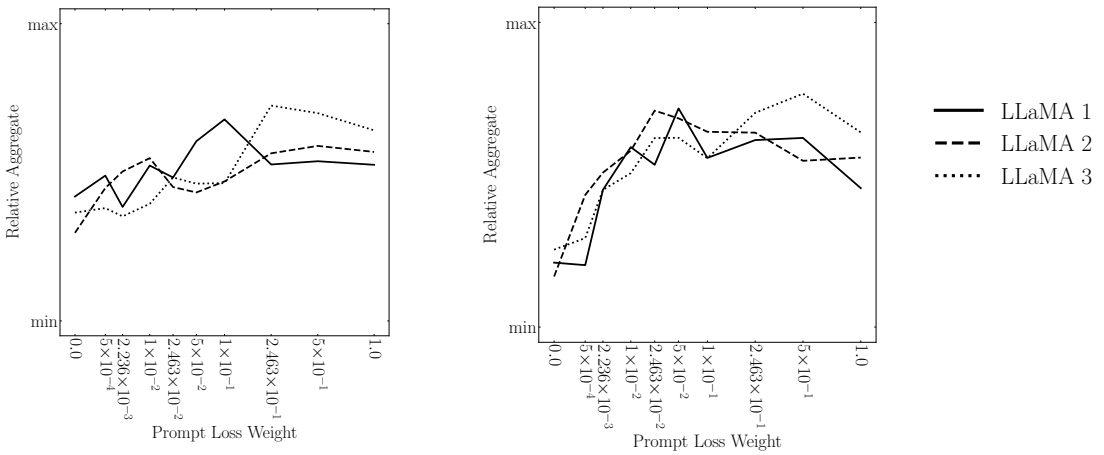
6 Supplemental Experiments

In this section, we present supplemental experiments that suggest that PLW cannot be replaced by alternative regularization techniques for SIFT and that the effects of PLW extend to other short-completion datasets. Since the translation benchmarks showed high levels of noise and unclear correlation with PLW in the main experiment, they were not included in supplemental experiments.

6.1 Alternative Regularizers

To investigate if the effects of PLW on short-completion SIFT can be emulated with other common regularization techniques, we repeated the AlpacaDataShort training runs for PLW $= 0$ and PLW $= 1$ while applying various regularizers. We tested weight decay, minimizing the Minkowski distance between PTLM weights and learned weights, dropout, and label smoothing. See Appendix C.1 for visualizations.

[2] Note that recent work (Li et al., 2023; Dubois et al., 2024) has shown that AlpacaEval 1 has a preference for long generations, and we argue that improved AlpacaEval scores are not simply due to longer generations. First, while AlpacaEval performance showed a nearly strictly-increasing relationship with PLW, generation length did not. Second, we used Mixtral as the auto-evaluator which showed a much lower length preference than the default evaluator (0.63 and 0.75, respectively).



(a) Original Datasets

(b) Prompt-Inverted Datasets

Figure 3: Relative aggregate scores showing the effects of PLW for SIFT on alternative datasets. **(a)** UltraFeedback-Cleaned and DatabricksDolly models. **(b)** UltraFeedbackShort and DatabricksDollyShort models.

LLaMA	Regularization	Aggregate Simple	Aggregate Relative
1	Weight Decay	**0.507**	0.795
	Minkowski Metric	0.505	0.812
	Dropout	0.500	0.783
	Label Smoothing	0.500	0.785
	PLW (Ours)	0.506	**0.855**
2	Weight Decay	0.537	0.773
	Minkowski Metric	0.538	0.772
	Dropout	0.541	0.805
	Label Smoothing	**0.538**	0.837
	PLW (Ours)	0.537	**0.894**

Table 4: Supplemental comparison of AlpacaDataShort models fine-tuned with various regularizers. High scores are in bold. As explained in section 5, the relative aggregate should be used for analysis, and the simple aggregate is provided for reference only.

Note that we do not compare PLW with KL-divergence. While using KL-divergence as a regularizing loss is common for SFT in general and is used in RL-based LLM alignment (Stiennon et al., 2020; Korbak et al., 2022; Gao et al., 2023a), collecting embeddings for any non-trivial dataset to be used for LLM SIFT presents a huge computation and memory overhead.

As can be seen in the results in table 4, relative aggregate scores were higher for models fine-tuned with fractional PLWs. This suggests that PLW provides a unique regularizing effect that cannot be easily replaced with other regularizers.

6.2 Alternative Datasets

We repeated our main training experiment using two additional datasets, a cleaned and binarized version of UltraFeedback (Cui et al., 2023; denoted UltraFeedbackCleaned) and databricks-dolly-15k (Conover et al., 2023; denoted DatabricksDolly). We also trained on prompt-inverted versions of both datasets, denoted UltraFeedbackShort and DatabricksDollyShort, respectively, and expanded analysis to include LLaMA 3 8B (AI@Meta, 2024) as a PTLM. See length statistics for these four datasets in table 1 and Appendix C.2 for additional visualizations. Note that compared with the unmodified datasets used in the main experiment, both UltraFeedback and DatabricksDolly have significantly lower generation ratios of $R_g = 1.77$ and $R_g = 0.83$, respectively. These datasets were chosen to demonstrate the effects of PLW when fine-tuning on data with relatively balanced generation ratios.

The combined relative aggregate scores for models trained on UltraFeedbackCleaned and DatabricksDolly and for models trained on the prompt-inverted UltraFeedbackShort and DatabricksDollyShort datasets can be seen in figure 3. Visual inspection suggests that PLW affected learning for both groups of datasets. For the shortened variants, performance appears to have the same negative quadratic relationship with PLW as the AlpacaDataShort models did in the main experiment. The relationship between PLW and performance for models fine-tuned on the unmodified datasets is weaker, but appears to be generally increasing.

The clear relationship in the shortened data variants shows that the results of the main experiment extend to additional datasets. Furthermore, the weak relationship between PLW and performance

of the unmodified data variants suggests that the effects of PLW are indeed dependent on the generation ratio. See Appendix D for an approach to predicting optimal PLW based on the generation ratio of the SIFT dataset.

7 Conclusion

In this study, we explored the effects of prompt loss weight (PLW) on LLM supervised instruction fine-tuning (SIFT).

We found that PLW had a statistically significant effect on learning for our short-completion dataset, and proper tuning of PLW allowed short-completion-trained models to outperform long-completion-trained models on short-generation benchmarks. We showed that the causal mechanism was due to a balance between two different regularizing effects and not due to increased training stability as is commonly attributed. We showed that the measured relationship extends to additional SIFT datasets and that the effects could not be sufficiently emulated with other regularizers.

Based on the above conclusions, we assert the following two points.

1. Since models fine-tuned on short-completion datasets and with properly tuned PLWs outperformed all other models on short-generation benchmark tasks, we conclude that PLW is critical for effectively fine-tuning for downstream short-generation tasks.

2. Given the importance of PLW and given that many SIFT datasets and almost all natural language understanding (NLU) datasets are short-completion datasets, we warn SIFT API providers about the need for a PLW parameter to adequately cover a full range of modeling applications.

Limitations

1. We analyzed prompt loss weighting (PLW) for instruction fine-tuning LLMs. We characterized seven fine-tuning datasets by their relative completion-prompt length ratios and reported on the effect of PLW when training on each dataset. It would be helpful to extend this research to a wider range of datasets to increase the strength of our conclusions and create more complete guidelines for prompt loss weighting.

2. Since we used pre-trained models and no layers were freshly initialized, there was little variance in initial experiments. We therefore limited runs to a single seed of 42.

3. Suggested values for PLW from section D are based on the included experiments. Best PLW values when fine-tuning different models or using different datasets or training regimes may vary from the relationships shown here, though we are still confident that performance will not vary significantly by PLW for long-completion data.

4. The focus of our research was on how PLW affected fine-tuning based on the completion-prompt *ratio* of the training dataset. However, the absolute length and size of the dataset will likely play a role in learning dynamics. It would be good to include that perspective in future research on token loss weights.

5. LLM-as-evaluator approaches like PandaLM and AlpacaEval are still relatively new, and these approaches are being actively developed. We chose to use Mixtral 8x7B as an auto-evaluator for AlpacaEval 1 due to budget limitations. While we cite high human evaluation correlation with Mixtral and justify this decision in section 4.4, using the default auto-evaluator would be beneficial for better comparison with other research.

6. While we define short- and long-completion data as have a completion-prompt ratio R_g lower and greater than 1, respectively, we do not provide justification for using 1 as the threshold. Choosing a meaningful reference would be helpful to future research.

Ethical Considerations

In this paper, we presented an analysis of the prompt loss weight hyperparameter for supervised instruction fine-tuning. We did not rely on human evaluators, and at no point in our research did we expose anyone to risk of harm.

We acknowledge that standard deep learning training methods have a high carbon footprint, and we performed over 200 fine-tuning training runs. Model outputs cannot be predicted in advance, and, while we release our model weights in the spirit of transparency and collaboration, models may hallucinate or produce offensive output. Additionally, our shortened datasets were generated from publicly released data, and we did not perform additional content filtering. A warning about both of these issues will be released along with the models and datasets.

Acknowledgements

This work was supported by the Technology Innovation Program funded by the Korean Ministry of Trade, Energy, and Industry (MOTIE, Korea) (No. 20014406, "Development of interactive sign language interpretation service based on artificial intelligence for the hearing impaired") and the Artificial intelligence industrial convergence cluster development project funded by the Ministry of Science and ICT (MSIT, Korea) & Gwangju Metropolitan City (No. BA00000797, LLM-based sign language translation for weather forecasts).

References

AI@Meta. 2024. Llama 3 model card.

Anthropic. 2023. Model card and evaluations for claude models. https://cdn.sanity.io/files/4zrzovbb/website/5c49cc247484cecf107c699baf29250302e5da70.pdf.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

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. *ArXiv*, abs/1803.05457.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *Preprint*, arXiv:2310.01377.

Jennifer Dodgson, Lin Nanzheng, Julian Peh, Akira Rafhael Janson Pattirane, Alfath Daryl Alhajir, Eko Ridho Dinarto, Joseph Lim, and Syed Danyal Ahmad. 2023. Establishing performance baselines in fine-tuning, retrieval-augmented generation and soft-prompting for non-specialist llm users. *arXiv preprint arXiv:2311.05903*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint*, arXiv:2305.14387.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. 2024. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*.

Leo Gao, John Schulman, and Jacob Hilton. 2023a. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023b. A framework for few-shot language model evaluation.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Tomasz Korbak, Ethan Perez, and Christopher Buckley. 2022. RL with KL penalties is better viewed as Bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1083–1091, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Diana Kozachek. 2023. Investigating the perception of the future in gpt-3, -3.5 and gpt-4. In *Proceedings of the 15th Conference on Creativity and Cognition*, C&C '23, page 282–287, New York, NY, USA. Association for Computing Machinery.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Gene Ruebsamen. 2023. AlpacaDataCleaned. `https://github.com/gururise/AlpacaDataCleaned`.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

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. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Qiang Heng, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2023a. Pandalm: Reproducible and automated language model assessment. `https://github.com/WeOpenML/PandaLM`.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *International Conference on Learning Representations (ICLR)*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics.

Lukas Wutschitz, Boris Köpf, Andrew Paverd, Saravan Rajmohan, Ahmed Salem, Shruti Tople, Santiago Zanella-Béguelin, Menglin Xia, and Victor Rühle. 2023. Rethinking privacy in machine learning pipelines from an information flow control perspective. *arXiv preprint arXiv:2311.15792*.

Appendices

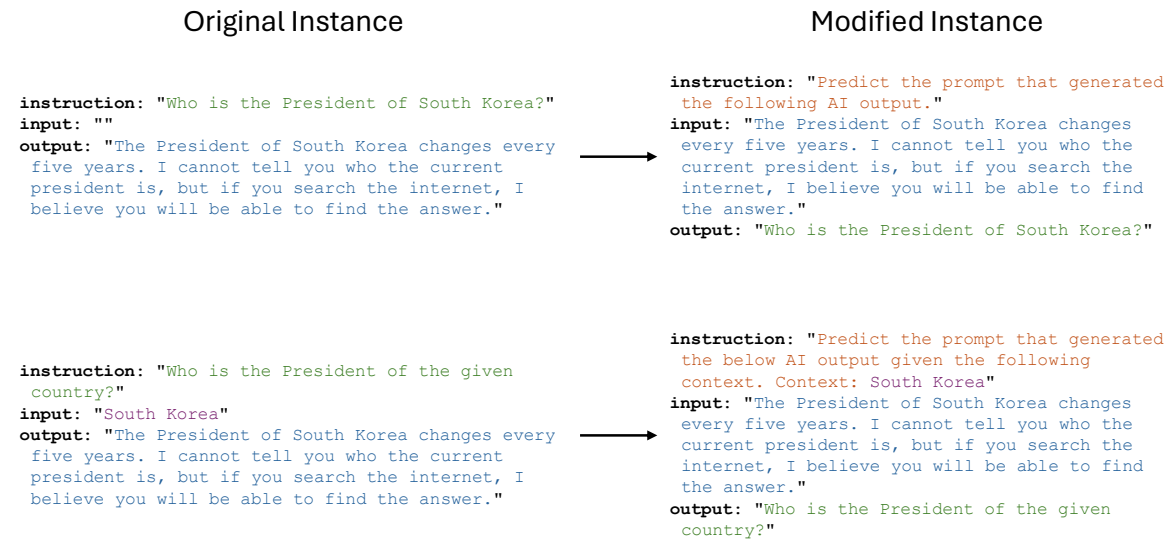


Figure 4: Examples of modifying prompt-completion ratios using prompt inversion, best viewed in color. To prompt-invert instances, we re-frame the prompt-completion task as an original-prompt-prediction task. I.e., we teach the model to predict the original instruction given an example completion and optional input. In the first example above, prompt inversion changes the instance's word-based completion-prompt ratio R_g from $34/(7+0) = 4.857$ to $7/(9+34) = 0.163$.

A Prompt Inversion

In order to experiment with short-completion fine-tuning datasets, we propose using *prompt inversion* to rotate the instruction, input, and output fields of instances. Prompt inversion modifies an instance to use the original instruction as the new completion text and the original output as the new input text. The model is then given the following instruction:
"`Predict the prompt that generated the following AI output.`"
if the input field is empty and
"`Predict the prompt that generated the below AI output given the following context. Context: <original-input>`"
if there is an input field. See figure 4 for a visualization of this process.

Synthesis of the original input and output texts to predict the original instruction requires both language understanding and reasoning, and prompt-inverted instances should be seen as natural language understanding (NLU) tasks.

To generate short versions of the instruction datasets used in our experiments, we used prompt-inversion to modify every instance with a completion-prompt length ratio $R_g > 1$, based on the tokenized lengths of each field (where "prompt" is the concatenation of the instruction and input fields as explained in section 2.1). Thus, given any long-completion dataset, a textually-similar short-completion dataset can be generated and used for comparison. Note that unless all instances have a generation ratio $R_g > 1$, the resulting dataset will contain a mixture of unmodified instances and prompt-inverted instances.

B Main Experiment Benchmarks

This section presents additional qualitative analysis of benchmark performance and score visualizations for each benchmark.

For both the simple aggregate and the relative aggregate, models trained on AlpacaDataShort showed a visual relationship with w_p. Based on this visual relationship, we divide benchmarks into three groups.

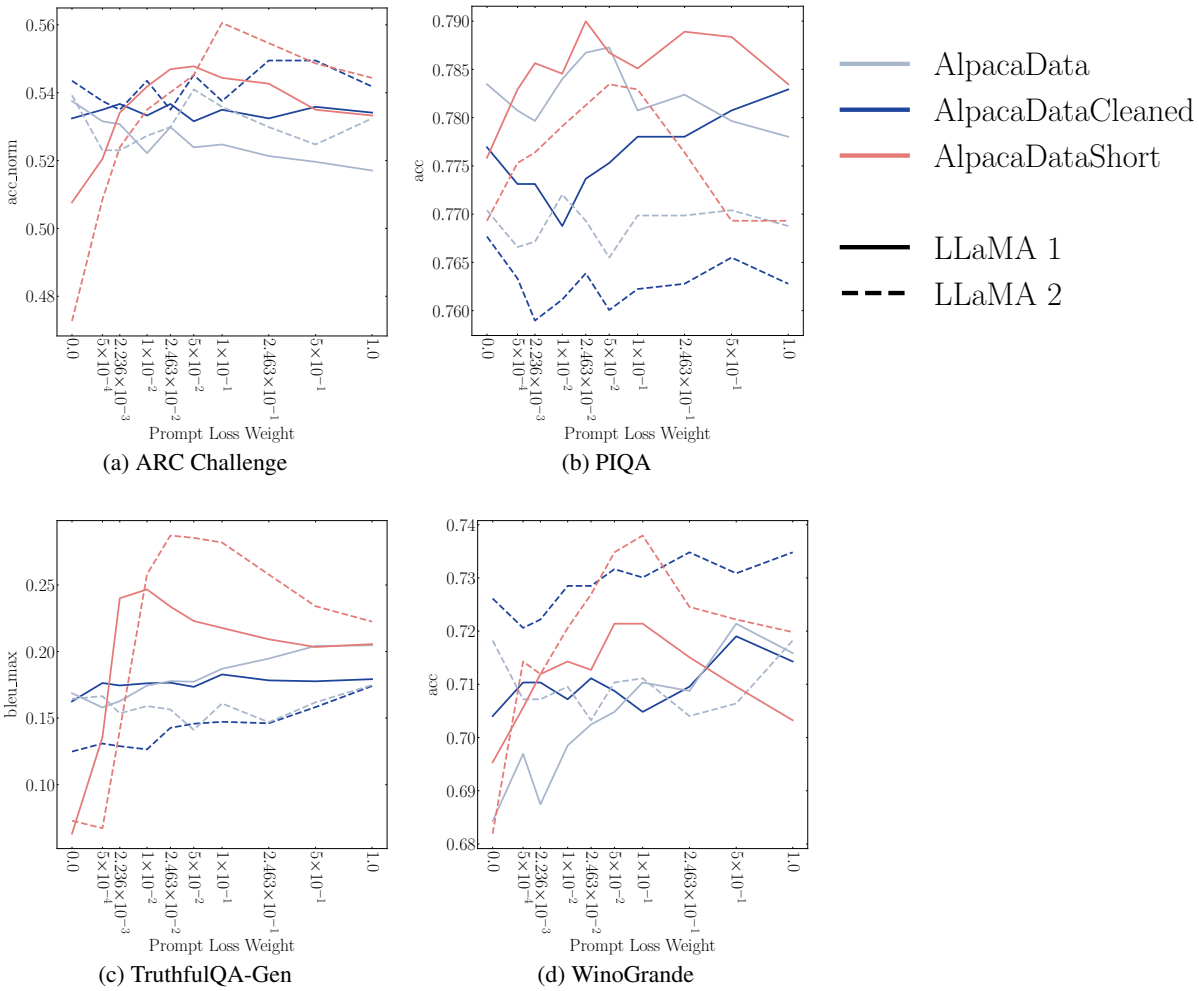
The first group showed a negative quadratic relationship with w_p, with performance **exceeding** that of AlpacaDataCleaned models. This group consists of ARC Challenge, PIQA, TruthfulQA-Gen, and WinoGrande benchmarks, and optimal PLW values for these four benchmarks vary from PLW $= 0.01$ to PLW $= 0.1$. See figure 5 for individual benchmark visualizations.

The second group of benchmarks showed steadily increasing performance as w_p increased, before leveling off to maximum values near $w_p = 1$. This group is TruthfulQA-MC2, AlpacaEval 1, and PandaLM. It is surprising that TruthfulQA-MC2 shows a relationshp more similar to the long-generation benchmarks and TruthfulQA-Gen resembles the other multi-choice benchmarks. See figure 6 for individual benchmark visualizations.

Interestingly, on the seven benchmarks from groups I and II, $w_p > 0$ was almost always better than $w_p = 0$ (i.e., complete masking led to the worst performance) for AlpacaDataShort models.

The third group consists of the six translation benchmarks and showed unclear correlation between performance and w_p. Though aggregating benchmarks into "to English" and "from English" subgroups creates visualizations suggestive of a relationship, benchmarks from this group showed relatively more noise than the other benchmarks. To reduce score noise, translation benchmarks were evaluated on the combined validation and test data splits, but there was still significant noise in the results. See figure 7 for individual benchmark visualizations.

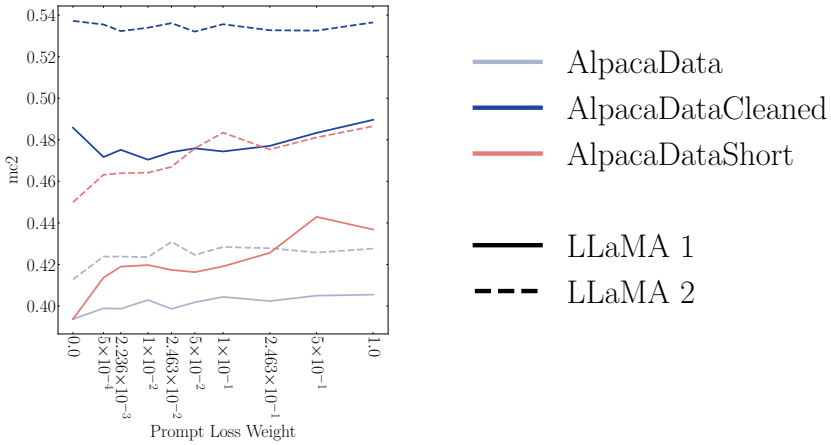
The performance difference across different w_p values for the two long-generation benchmarks was around twenty percentage points, in stark contrast to the less than two percentage point change for short-generation and multiple choice benchmarks. This suggests that PLW plays an important role in the ability to generate high quality text, and the optimal PLW for short-generation and long-generation benchmarks is clearly different. Also note that performance of LLaMA 2 models was in general higher than that of LLaMA 1 models and performance of AlpacaDataCleaned models were higher than that of AlpacaData models, validating the improvements of LLaMA 2 and AlpacaDataCleaned over their predecessors.



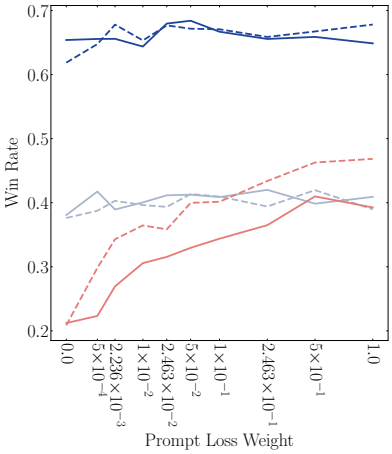
(a) ARC Challenge (b) PIQA

(c) TruthfulQA-Gen (d) WinoGrande

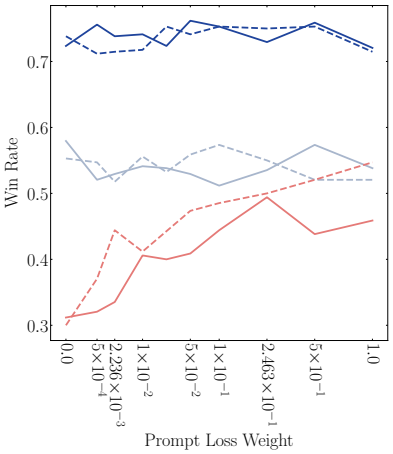
Figure 5: Group I benchmark performance. Note the negative quadratic relationship with transformed PLW.



(a) TruthfulQA-MC2

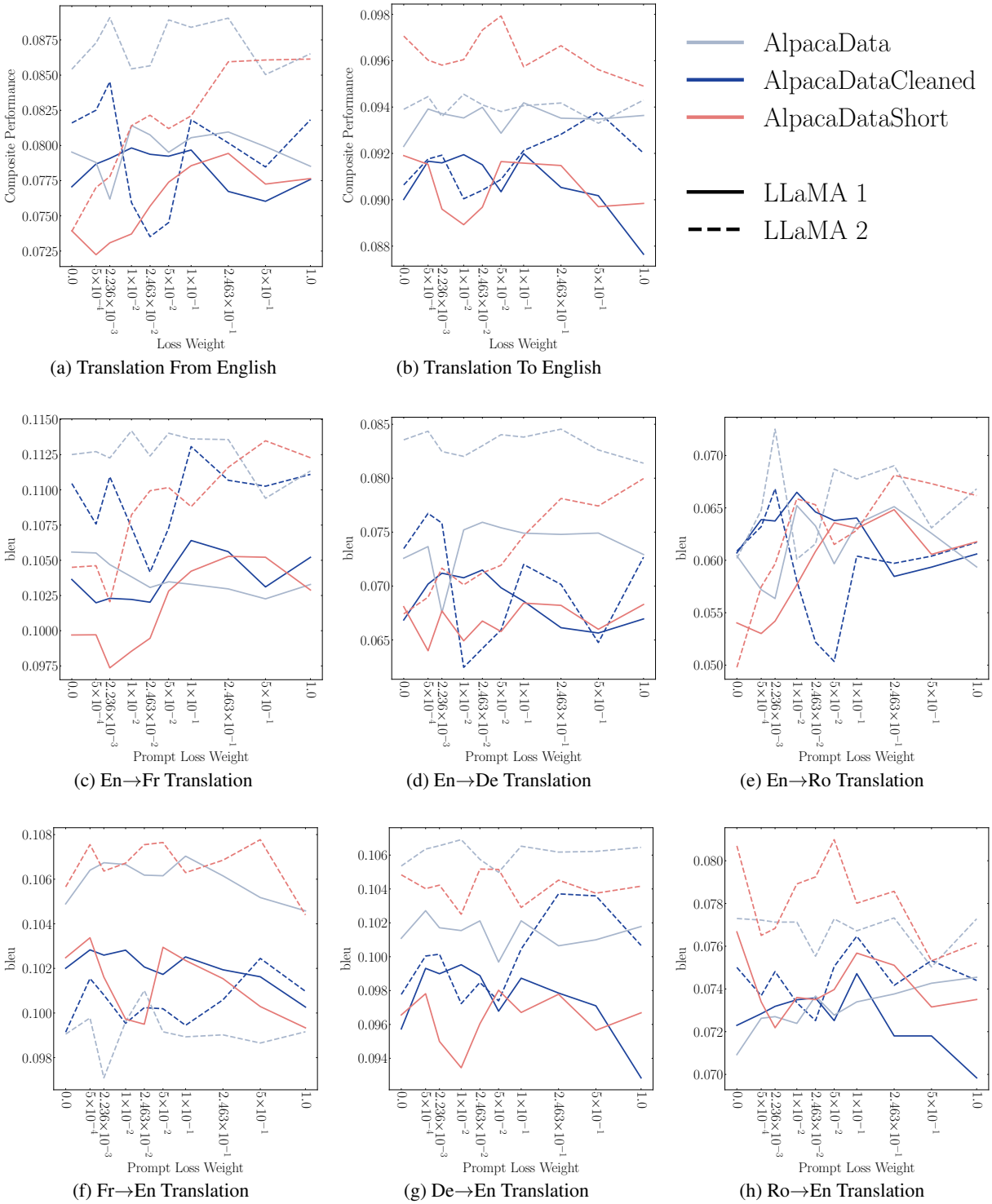


(b) Alpaca Eval (AE) v1



(c) PandaLM

Figure 6: Group II benchmarks showed increasing performance with PLW.



(a) Translation From English (b) Translation To English

(c) En→Fr Translation (d) En→De Translation (e) En→Ro Translation

(f) Fr→En Translation (g) De→En Translation (h) Ro→En Translation

Figure 7: Group III benchmarks showed little relationship between performance and PLW.

C Visualizations for Supplemental Experiments

This appendix contains visualizations and additional details about the two supplemental experiments from section 6.

C.1 Regularization Comparison

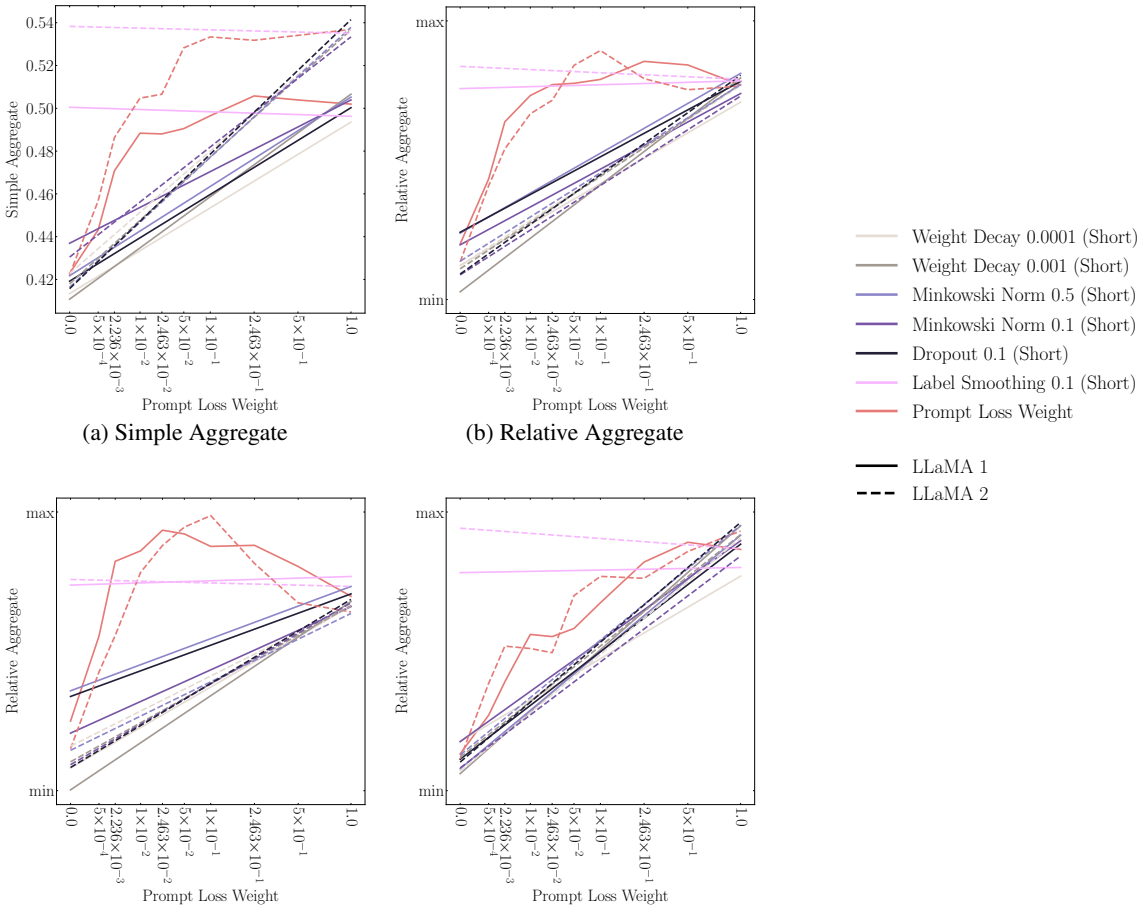
For the first supplemental experiment, we wanted to investigate if PLW is necessary for fine-tuning on short-completion data or if another regularization technique could yield the same benefits. As explained in section 6.1, we chose to examine four types of regularization in addition to PLW, intentionally not evaluating KL divergence-based regularization due to the difficulty of applying it to LLM SIFT.

Several aggregate scores for regularizations with example parameters are presented in figure 8, and best scores for each type of regularization is presented in table 4 in the main paper. Visualization of relative aggregate scores revealed that models fine-tuned with fractional PLW generated high scores on multiple-choice and short-generation benchmarks while long-generation benchmarks (AlpacaEval 1 and PandaLM) actually benefitted the most from alternative regularization methods. However, the effect on the multiple choice and short generation benchmarks was relatively strong, and the combined relative aggregate also showed maximal values for models fine-tuned with fractional PLW.

Interestingly, most regularization methods performed better when coupled with PLW$= 1$ than with PLW$= 0$ except for label smoothing which performed marginally worse at PLW$= 1$ than at PLW$= 0$.

C.2 Dataset Comparison

In the second supplemental experiment, we wanted to explore if the relationship between PLW and model performance on downstream tasks measured for AlpacaDataShort models existed for other fine-tuning datasets as well. See additional visualizations in figure 9.



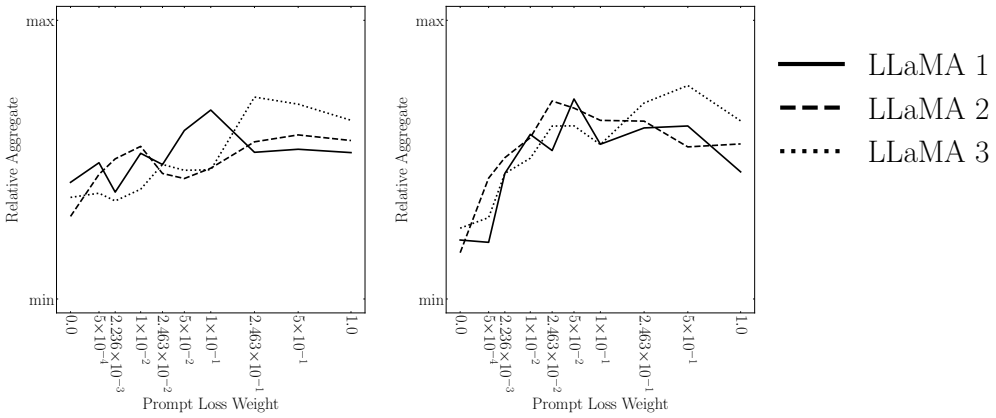
(a) Simple Aggregate

(b) Relative Aggregate

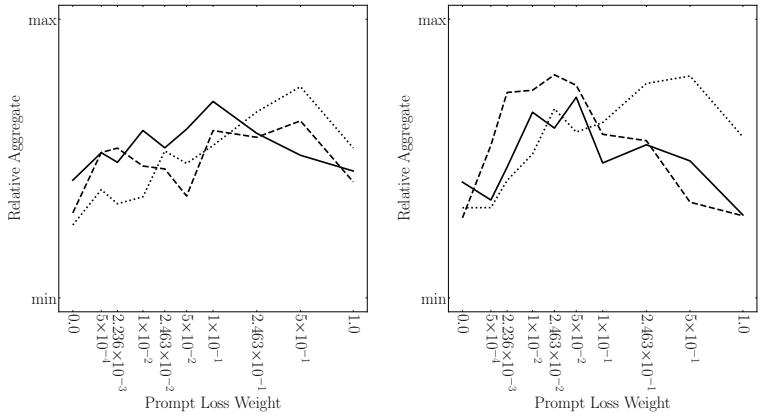
(c) Relative Aggregate (Group I)

(d) Relative Aggregate (Group II)

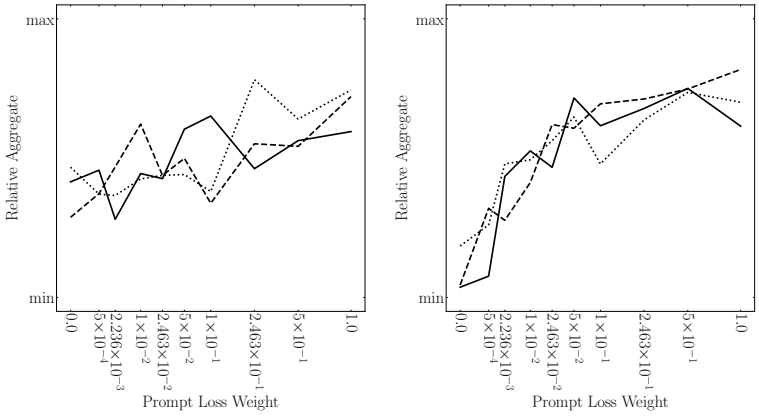
Figure 8: Comparison of PLW with other regularization techniques (calculated for PLW $= 0$ and PLW $= 1$). **(a)** The simple aggregate. **(b)** The combined relative aggregate shows that models fine-tuned with fractional PLW on AlpacaDataShort outperformed models fine-tuned with alternative regularizations. **(c)** Fractional PLW performance is most extreme for multiple choice and short-generation benchmarks (group I). **(d)** Performance of fractional PLW models on group II benchmarks is less pronounced, with PLW-optimized models performing slightly worse than several other alternative metrics.



(a) Original Datasets (b) Short Dataset Variants

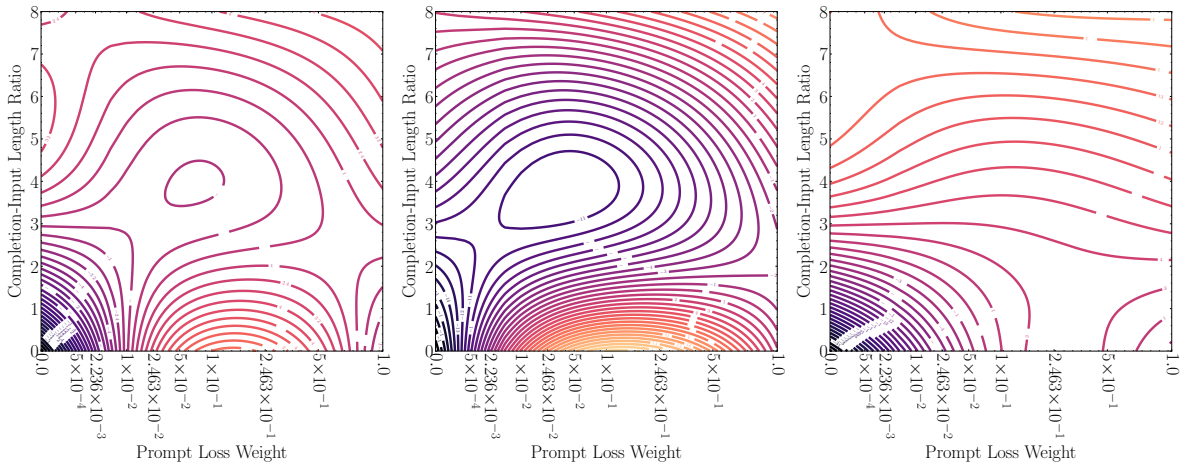


(c) UltraFeedbackBinarizedClean (d) UltraFeedbackBinarizedShort



(e) DatabricksDolly (f) DatabricksDollyShort

Figure 9: Relative aggregate scores for models fine-tuned on alternative instruction datasets.



(a) "All" Prediction (b) "MC" Prediction (c) "Gen" Prediction

Figure 10: Best viewed digitally for improved resolution.

D Optimal Prompt Loss Weight

In the main regression experiment, we showed that PLW is an important hyperparameter when fine-tuning on short-completion data but is effectively irrelevant when using long-completion data. In this appendix, we present several models for predicting an optimal PLW given a dataset's R_g. These models are based on the AlpacaData dataset, AlpacaDataCleaned dataset, and several modified versions of AlpacaDataCleaned and therefore should be seen as an exercise rather than an authoritative reference on optimal PLW weights.

We first repeated our SIFT experiments on two additional datasets: AlpacaDataMedium and AlpacaDataTiny to increase coverage of the parameter space. AlpacaDataMedium and AlpacaDataTiny have R_g values of 1.0 and 0.042, respectively, and were generated using prompt inversion (see Appendix A) but selecting instances to modify in order to approach target R_g values.

We then fit several generalized additive model (GAMs) with a tensor smooth for the interaction between PLW and R_g. GAMs offer more flexible modeling than linear models but at the expense of interpretability. We fit our models using the **R** library `mgcv` and the following equation:
"`score ~ te(w, r, k=3) + factor(b)`", where `te` is a full tensor product smooth, `w` is the untransformed PLW parameter, `r` is the R_g, `k` is the number of splines, and `b` is the benchmark task.

Using the fitted `w-r` interaction, we then estimated optimal PLW value for a given completion-prompt ratio R_g. See figure 10a for a visualization of a GAM fitted on all benchmark tasks.

Roughly, the fitted interaction term recommended using PLW $= 0.155$ for small completion-

prompt length ratios ($R_g \leq 1$) and up to PLW $= 0.278$ for a $R_g = 1.5$ for optimal performance across all tasks. This prediction is close to our regression-predicted value of 0.242. The interaction term also confirms our observations that PLW is less important for data with relatively long completions.

Since the relationship between PLW and benchmark performance depends heavily on the type of benchmark task, we also fit GAMs for an aggregate of multiple choice benchmark scores (labeled "MC") and generation benchmark scores (labeled "Gen"). We found that the translation benchmarks contributed little to the predictive power of the fitted GAMs and while their scores are included in the "All" GAM, we did not include them when fitting the "Gen" GAM. See figures 10b and 10c for contour plots for the "MC" and "Gen" benchmarks, respectively.

Also see table 5 for a list of GAM-based optimal PLWs over a range of completion-prompt ratios. Again, predicted optimal PLWs confirmed the conclusions from our regression analysis in section 5.2.

R_g	Optimal PLW		
	All	MC	Gen
8.0	1.000*	1.000*	0.654
7.5	1.000*	1.000*	1.000*
7.0	1.000*	1.000*	1.000*
6.5	1.000*	1.000*	1.000*
6.0	1.000*	1.000*	0.000*
5.5	1.000*	1.000*	0.000*
5.0	0.000*	1.000*	0.000*
4.5	0.000*	1.000*	0.000*
4.0	1.000*	1.000*	0.000*
3.5	1.000*	1.000*	0.000*
3.0	1.000*	1.000*	1.000*
2.5	1.000*	1.000*	1.000*
2.0	0.239	1.000*	0.679
1.5	0.183	0.278	0.385
1.0	0.155	0.183	0.321
0.5	0.155	0.155	0.292

Table 5: Optimal prompt loss weight (PLW) per completion-prompt length ratio R_g on all benchmarks ("All"); multiple choice benchmarks ("MC"); and the combination of TruthfulQA-Gen, Alpaca Eval 1, and PandaLM benchmarks ("Gen"). Predictions are based on the ratio-PLW interaction term of fitted generalized additive models.
*The difference between the maximum and minimum predicted values at this ratio is less than 5% of the score range.

E Reproducibility

This section provides technical details on all experiments and benchmarks for transparency and to encourage reproduction of results. To help with reproducibility, we will also uploaded the fine-tuned models, test generation outputs, and our modified datasets to the HuggingFace Hub and can be accessed at https://huggingface.co/collections/mathewhe/plw-66fe40fac6e586d8435bd563. Note that unless specified otherwise, default parameters were used for all training and testing.

E.1 Model Fine-Tuning

Model fine-tuning was performed with the Stanford Alpaca GitHub repository at https://github.com/tatsu-lab/stanford_alpaca/tree/761dc5b.

To experiment with prompt loss weight, we modified HuggingFace's Transformers library to allow specifying a `loss_weights` parameter for `LlamaForCausalLM`'s forward method.

We used the following commit of Transformers https://github.com/huggingface/transformers/tree/3b7675b.

All models were trained on a single four A100 80GB node and we used the first set of hyperparameters recommended in the Fine-tuning subsection of Stanford Alpaca's README.md file, except for the three experimental variables: pretrained model, prompt loss weight, and training dataset.

AlpacaData is available from the Stanford Alpaca repository. AlpacaDataCleaned can be found at https://github.com/gururise/AlpacaDataCleaned/tree/791174f and is labeled "alpaca_data_cleaned.json". As noted above, AlpacaDataShort can be accessed at https://huggingface.co/collections/mathewhe/plw-66fe40fac6e586d8435bd563.

E.2 Model Evaluation

We used three evaluation frameworks: EleutherAI's Language Model Evaluation Harness (EEH), AlpacaEval 1, and PandaLM.

In an effort to match the current HuggingFace Open LLM leaderboard, we evaluated ARC Challenge, TruthfulQA-MC2, WinoGrande, and PIQA on the same EEH commit that the HuggingFace leaderboard uses: https://github.com/EleutherAI/lm-evaluation-harness/tree/b281b09 We also matched the number of shots with the number used for the HuggingFace leaderboard for ARC Challenge, TruthfulQA-MC2, and WinoGrande.

TruthfulQA-Gen and all translation tasks were evaluated using a more recent commit at https://github.com/EleutherAI/lm-evaluation-harness/tree/b93c3bc. We modified the translation tasks at this commit to include an appropriate prompt to support zero-shot translation. These changes can be seen at https://github.com/mathewhuen/plw_lm-evaluation-harness/compare/b93c3bc..1957d1a.

Though version 2 of AlpacaEval has recently been released, we used version 1 from the following commit https://github.com/tatsu-lab/alpaca_eval/tree/495b606. To use Mixtral 8x7B as an auto-evaluator for AlpacaEval 1, we modified the Guanaco-33b evaluator's config and prompt minimally to match Mixtral's format. Models were evaluated on the default test set which can be found at https://huggingface.co/datasets/tatsu-lab/alpaca_eval/blob/main/alpaca_eval.json. We plan on submitting a pull request with these additions in the near future.

For PandaLM, we used the commit at https://github.com/WeOpenML/PandaLM/tree/eb758c4 and evaluated on version 1 of the default test set (found at "data/testset-inference-v1.json" in the PandaLM repository).

E.3 Regression

All statistical analysis and regression modeling was performed with **R**, version 4.3.0. We used the `glmmTMB` library, version 1.1.8, to perform generalized linear mixed modeling (GLMM) and validated results with the same library and with `DHARMa`, version 0.4.6.

E.4 Causal Mechanism

Most of the analysis performed to shed light on the causal mechanism should version and implementation agnostic. However, BLEU score implementations vary widely, and we used sacre BLEU to evaluate memorization of the training set. We used Corpus BLEU from the sacreBLEU library at https://github.com/mjpost/sacrebleu. Instead of a commit hash, we share the metric signature:

"`nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.4.0`"

E.5 Supplemental Experiments

The first supplemental experiment used the same experimental setup and data from the main experiment. Of the tested regularization methods, we used the weight decay implementation in PyTorch's AdamW optimizer, attention dropout implemented by the LlamaAttention module from Transformers, and label smoothing supported by the Trainer class from Transformers. We manually implemented regularization based on the Minkowski distance by calculating the mean of the $p = 1$ Minkowski distance between each pair of weight tensors from the PTLM and the trained model.

The second experiment introduced two new datasets: UltraFeedbackBinarizedCleaned and DatabricksDolly. UltraFeedbackBinarizedCleaned can be found at https://huggingface.co/datasets/allenai/ultrafeedback_binarized_cleaned/tree/f304ce5. And DatabricksDolly can be found at https://huggingface.co/datasets/databricks/databricks-dolly-15k/tree/bdd27f4. The modified datasets UltraFeedbackShort and DatabricksDollyShort can be accessed at https://huggingface.co/collections/mathewhe/plw-66fe40fac6e586d8435bd563.

E.6 Predictive Model

We fit several generalized additive models (GAMs) in Appendix D using the mgcv library, version 1.9-1 and the same version of **R** as above, version 4.3.0.

Resource	License	Application
Transformers	Apache 2.0	Model Training
Stanford Alpaca	Apache 2.0	Model Training
AlpacaDataCleaned	Apache 2.0	Model Training
Ultrafeedback Binarized Cleaned	MIT	Model Training
databricks-dolly-15k	CC BY-SA 3.0	Model Training
LLaMA 1	LLaMA License	Pre-trained model weights
LLaMA 2	LLaMA 2 Community License	Pre-trained model weights
LLaMA 3	LLaMA 3 Community License	Pre-trained model weights
Mixtral 8x7B	Apache 2.0	Model Evaluation
EleutherAI's LM Evaluation Harness	MIT	Model Evaluation
AlpacaEval 1	Apache 2.0	Model Evaluation
AlpacaEval Dataset	CC BY-NC 4.0	Model Evaluation
PandaLM	Apache 2.0	Model Evaluation
ARC Challenge	CC BY-SA 4.0	Model Evaluation
PIQA	AFL 3.0	Model Evaluation
TruthfulQA	Apache 2.0	Model Evaluation
WinoGrande	Apache 2.0	Model Evaluation
WMT 14	No License	Model Evaluation
WMT 16	No License	Model Evaluation

Table 6: Licenses for resources used in this research.

F Artifact Licensing

We respected all licenses for artifacts and resources used in this research. Please see table 6 for an overview of primary resources and licenses.