# Instruction Fine-Tuning: Does Prompt Loss Matter?

**Anonymous ACL submission**

## Abstract

We present a study analyzing the effects of prompt loss weighting (PLW) on supervised instruction fine-tuning. We recreated Stanford's Alpaca experiment with both LLaMA 1 and LLaMA 2 and multiple instruction datasets. We found that performance of models fine-tuned on our short-completion dataset had a statistically-significant negative quadratic relationship with PLW, but performance of models fine-tuned on medium- and long-completion data did not show any relationship with PLW. I.e., prompt loss can be safely ignored for many datasets. For short-completion data, small values ($0.01 - 0.1$) of PLW were optimal for multiple-choice and short-generation tasks while large values ($\approx 1.0$) of PLW were optimal for long-generation tasks. We concluded that low non-zero PLW encourages models to not diverge from pre-trained model weights during training and high PLW reduces overfitting. Finally, we present a rough guide for selecting PLW values based on the completion-prompt length ratio of fine-tuning data.

## 1 Introduction

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

The goal of SIFT is to teach a model to perform a language task given a prompted instruction. While training, model parameters are updated by optimizing for next-token maximal likelihood classification. However, it is unclear if this optimization should be performed on the entire prompt-completion sequence or only on the target completion.

Most open sourced solutions for supervised fine-tuning (SFT) either mask the prompt loss or use the entire sequence loss. On the other hand, OpenAI previously supported a `prompt_loss_weight` parameter that allowed users to specify PLW during SIFT.

We make the following contributions related to this parameter:

- We show that PLW has a statistically significant negative quadratic relationship with model performance when fine-tuning on short-completion instruction data.
- We show that PLW and prompt masking parameters can be safely ignored when fine-tuning on medium- or long-completion data, streamlining the fine-tuning process.
- We provide evidence that the benefits of PLW for short-completion data are due to a regularizing effect and not due to previously suspected mechanisms.
- We present a baseline for PLW selection for short-completion fine-tuning.

We provide relevant background and hypotheses in section 2 and section 3, respectively. We resent our methodology in section 4. Descriptive and regression analysis and an exploration of causal mechanisms are presented in section 5. We extend the experiment to additional datasets and present a basic model for prompt loss weight selection in section 6. We present conclusions in section 7.

## 2 Background

### 2.1 Definitions

We define *instruction data* as one or many instances of structured text data, each containing an instruction text, an optional context or input text, and a

target output text. For the rest of the paper, we use the term *prompt* to refer to the concatenation of instruction and input texts and the term *completion* to refer to the target output text. SIFT attempts to train 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. 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 loss. In Transformers, optimizing next-token prediction on prompt tokens is therefore binary—either token loss is masked (PLW $= 0$) or it is unmasked (PLW $= 1$).

Until recently, OpenAI supported a `prompt_loss_weight` parameter in their fine-tuning API, but it was officially removed 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, there were several studies that reported parameter values for `prompt_loss_weight` when fine-tuning OpenAI models.

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 documentation for `prompt_loss_weight`, we expect PLW to have a stabilizing effect on LLM SIFT when using short-completion data. Our expectation was that PLW preserves the hidden representation learned for next-token-classification when most embeddings are being used to predict tokens at the end of the sequence (as is the case when fine-tuning on short-completion data). This would represent a positive relationship between PLW and downstream performance.

However, unless the model is expected to generate prompt-like data in downstream tasks, the training and downstream sequence contexts will not match. In other words, training a model to maximize next-token-prediction on prompt tokens should be most useful for generating instruction data. Due to this train-test mismatch, we expect PLW to also have a negative influence on downstream performance.

Based on the assumption that some amount of PLW is beneficial for short-completion SIFT, we expect these two competing factors to result in a downward curved relationship between downstream performance and PLW. Limiting PLW to the range of [0, 1], we postulate that there is a critical value $\lambda$ for PLW with $0 <= \lambda <= 1$. For PLW less than $\lambda$, the positive effect dominates the negative effect and for values greater than $\lambda$, the negative effect dominates the positive effect. If $\lambda = 0$, then PLW's contribution to model performance is strictly negative. Conversely, if $\lambda = 1$, then PLW contributes strictly positively to model performance.

We postulate that both $\lambda$ and the magnitude of the effect are inversely proportional to $R_g$ since there will be $R_g$ as many completion token loss terms than prompt loss terms. Note that $\lambda$ is not an intrinsic characteristic of the dataset, model architecture, or training algorithm. Rather, it would depend on numerous factors and change for each task.

We tested the relationship between PLW and model performance using three fine-tuning datasets spanning a range of $R_g$ values: 0.327, 7.83, and

| 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 | 0.082 |

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 $R_g$.

0.082. We present the following hypotheses to be tested against each dataset.

**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 fixed the threshold $\alpha = 0.05$ significance level, and we expected to be able to reject $\mathbf{H_0}$ 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., 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 C 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 $\mathtt{w_p}$

### 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, and AlpacaDataShort.

AlpacaDataCleaned (Ruebsamen, 2023) 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 swapping instructions and completions for all long-completion instances and rephrasing the task as an instruction-prediction task.

For instances with empty input, we used
"`Predict the prompt that generated the following AI output.`"
as the new instruction field. For instances with non-empty input, we used
"`Predict the prompt that generated the below AI output given the following`

| 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 | $G_S$ |
| Alpaca Eval (Mixtral) | 1 | 1 | Test | $G_L$ |
| PandaLM | 1 | 0 | Test | $G_L$ |

Table 2: Benchmark tasks used for evaluation. All benchmarks were scored using Eleuther AI's Language Model Evaluation Harness, the AlpacaEval 1 framework, and the PandaLM framework. 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.

`context. Context: <original_input>`"
as the new instruction field. In all cases we then used the original completion as the new input field and the original instruction as the new completion field.

Descriptive statistics for tokenized instruction, input, and completion sequences 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 both multiple choice and text generation tasks. We attempted to select benchmarks that were relatively cheap to compute and covered a range of tasks. We used three evaluation frameworks: EleutherAI's Language Model Evaluation Harness (EEH) (Gao et al., 2023), Alpaca Eval 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 benchmarks weremultiple 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 benchmarks 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 learned during pretraining (English, French, German, and Romanian). For WMT benchmarks, we used the 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 Alpaca Eval 1 and PandaLM, which are both LLM-as-a-judge frameworks.

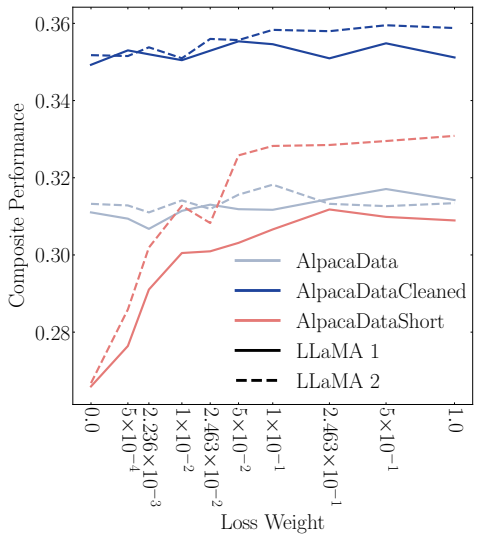
The default auto-evaluator for Alpaca Eval 1 is GPT4, but evaluation of all training runs would be beyond the scope of this research, so we used Mixtral 8X7B (Jiang et al., 2024) as an auto-evaluator. Mixtral performed the best of all available LLMs that we tested on Alpaca Eval's evaluator test dataset (Dubois et al., 2023), scoring 64.9 for agreement with human evaluators, just below Claude's (Anthropic, 2023) 65.3 agreement. For reference, the default GPT4 auto-evaluator scored a 70.99 human agreement.

## 5 Results and Discussion

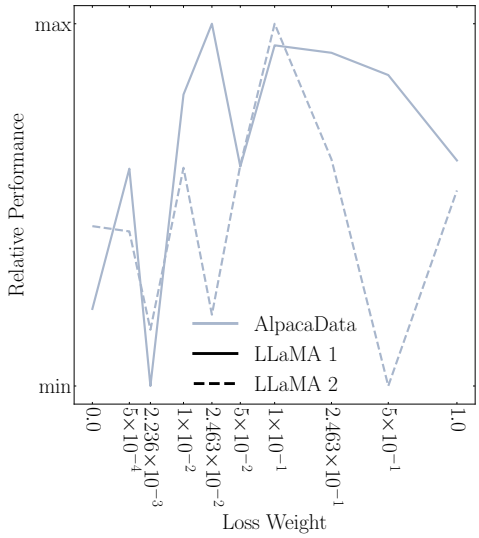
### 5.1 Performance Trends

A visualization of the simple and relative performance aggregates by $\mathtt{w_p}$ and dataset is presented in figure 1. The simple aggregate is the unweighted mean of all task scores for given $\mathtt{w_p}$ and dataset values. For the relative aggregate, we first min-max normalized benchmark scores over each dataset-benchmark group and then performed aggregation. The simple aggregate is dominated by large performance changes on the AlpacaEval 1 and PandaLM benchmarks. Therefore, all analysis was performed using the relative aggregate and the simple aggregate was visualized only for completeness.

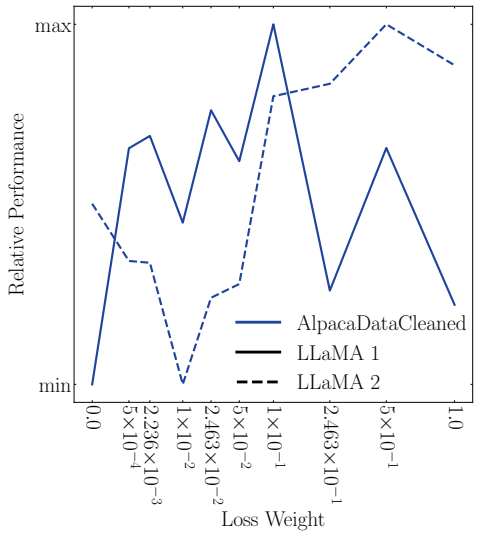
For both the simple aggregate and the relative aggregate, only models trained on AlpacaDataShort showed a visual relationship with $\mathtt{w_p}$. On ARC Challenge, PIQA, TruthfulQA-Gen, and WinoGrande, performance of AlpacaDataShort models showed a negative quadratic relationship with $\mathtt{w_p}$, with performance reaching or exceeding that of AlpacaDataCleaned models. Optimal PLW values for these four benchmarks vary from PLW $= 0.01$ to PLW $= 0.1$. On TruthfulQA-MC2, AlpacaEval 1, and PandaLM, performance of AlpacaDataShort models steadily increased with $\mathtt{w_p}$ before leveling



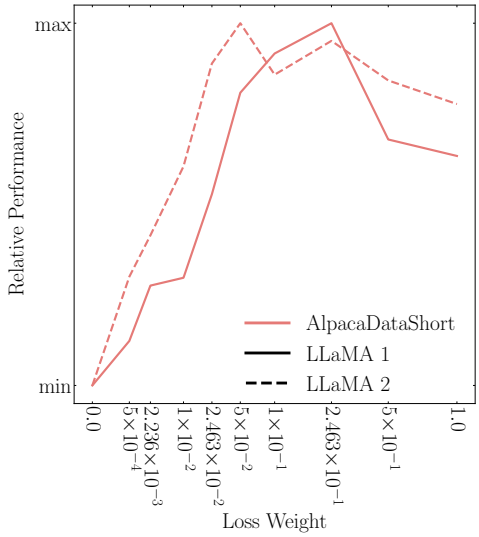

(a) Simple Aggregate




(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.

off to maximum values near $w_p = 1$. For these seven benchmarks, $w_p > 0$ was always better than $w_p = 0$ (i.e., complete masking led to the worst performance) for AlpacaDataShort models.

Identification of trends on the six translation benchmarks was less clear due to the high level of noise relative to performance range. Aggregating scores from benchmarks translating from English to the paired language benchmarks is suggestive of an increasing relationship, but is incredibly noisy.

The performance difference across $w_p$ 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.

Finally, performance of LLaMA 2 models was 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.

See Appendix A for detailed figures for each

| | | Coeff | | |
|---|---|---|---|---|
| | P-Value | $\mathtt{w_p}$ | $\mathtt{w_p}^2$ | (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: $\mathtt{w_p}$ 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.

benchmark.

### 5.2 Regression

For each data group, we fit a generalized linear mixed model (GLMM) predicting min-max transformed benchmark scores. As with the min-max transformation for the relative aggregate, benchmark scores were min-max transformed over dataset-benchmark groups.

We expected a quadratic relationship between the score and $\mathtt{w_p}$, so we included a second order polynomial of $\mathtt{w_p}$ as a fixed effect. Furthermore, we knew that the performance-PLW relationship varies by benchmark and that scores were min-max normalized over each benchmark, so we used a random slope with respect to 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`:

$\mathtt{score} \sim \mathtt{pol(w_p,2)} + (0+\mathtt{pol(w_p,2)|b}) + (1|\mathtt{m})$

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 $\mathtt{w_p}$ 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 $\mathtt{w_p}$ at our target $\alpha = 0.05$ significance level.

This means that while wee 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 $\lambda$ 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:

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

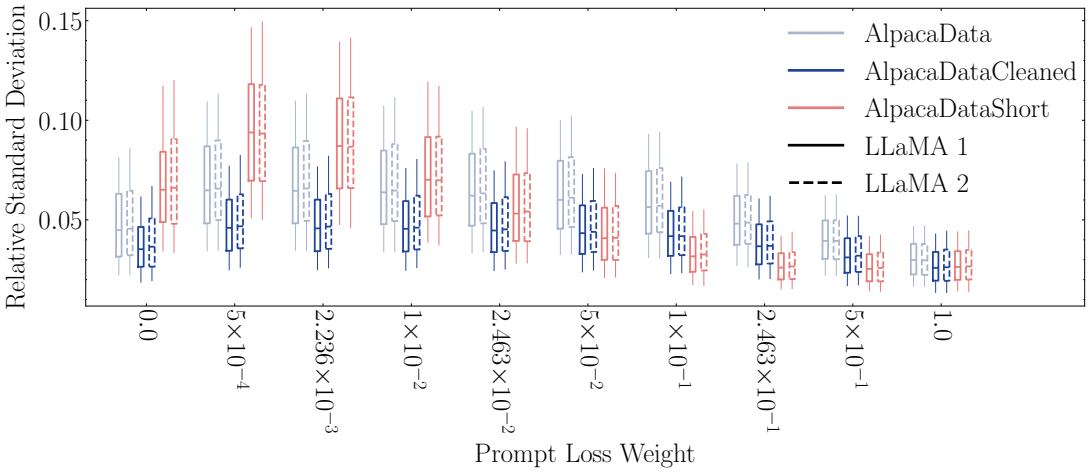
which has a global maximum at $\mathtt{w_p} = 0.652$. Reversing the power transformation yields a critical value for PLW at $\lambda = 0.242$. We verified that $\lambda$ overlaps with the visualized maximum value range for the relative aggregate in figure 1d.

### 5.3 Causal Mechanism & Interpretation

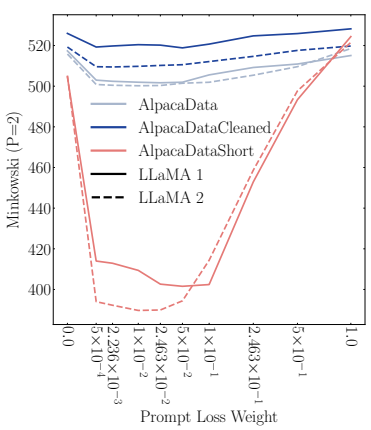
To identify possible causal mechanisms, we first investigated the effects of PLW on training stability by analyzing training loss relative standard deviation (RSD) over all five-step windows. See figure 2a for a boxplot of mean RSD for each model. For all dataset and PTLM factors, increasing PLW from 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 initially increase before decreasing.

If training loss stability was the primary factor in improved performance, we would expect stability to be best for PLW between $0.01$ and $0.1$ 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.

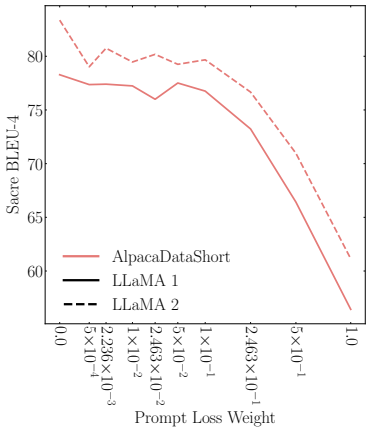
There is likely either a tradeoff between training loss stability and some other factors that affects model performance or model loss stability is not



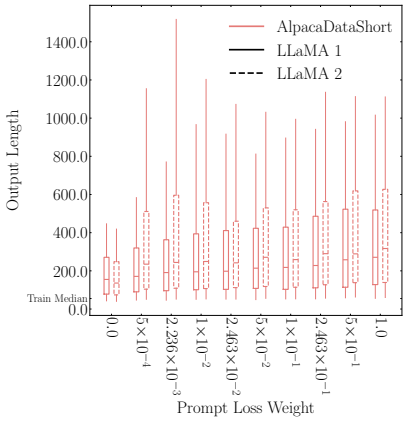

(a) Training Loss Stability




(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.

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.

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. Unfortunately, we do not have an explanation for this apparent contradiction and simply conclude that PTLM model weights were better preserved for small non-zero PLW *despite* high loss instability.

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 AE1 test set which showed high deviations in generation length for the high loss RSD PLW values for LLaMA 2 only and a generally increasing length trend with PLW. Considering

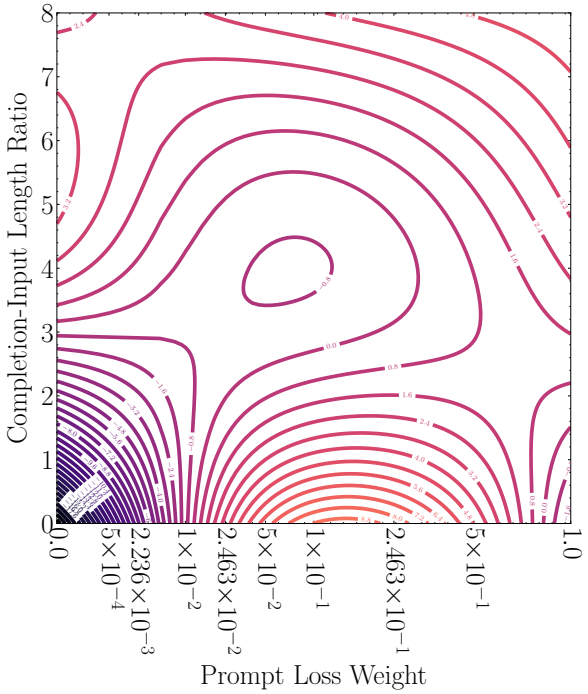



Figure 3: Ratio-PLW Relationship

that AlpacaDataShort contains short completions, this strongly suggests that PLW contributes a regularizing effect and helps decrease overfitting.

The positive relationship between performance and PLW on Alpaca Eval and PandaLM as opposed to the negative quadratic relationship seen on several of the other benchmarks can then be explained by a tradeoff between longer generations and decreased overfitting for high PLW and better preservation of PTLM weights for low PLW.

## 6 Predicting Prompt Loss Weight

We have shown that PLW is an important hyperparameter when fine-tuning on short-completion data but is effectively irrelevant when using long-completion data. We now present a model for predicting an optimal PLW given a dataset's $R_g$.

We first extended the previous experiments to 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 similarly with AlpacaDataShort, but with different target completion-prompt ratios.

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 can now estimate an optimal PLW value for a given completion-prompt ratio $R_g$. See figure 3 for a visualization.

Roughly, the fitted interaction term recommends 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 conclusion that PLW is less important for data with relatively long completions.

This is a rough guide, and we recommend using it as a starting point. See appendix B for additional plots using different benchmark tasks.

## 7 Conclusion

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

We showed that performance is unaffected for the two long-completion datasets (AlpacaData, the original Alpaca dataset, and AlpacaDataCleaned, an updated version of AlpacaData) while performance had a negative quadratic relationship with prompt loss weight on our short-completion dataset (with a completion-prompt ratio $R_g \approx 0.08$).

Interestingly, the highest scores on the ARC Challenge, PIQA, and WinoGrande benchmarks were all from models trained on short-completion dataset. This suggests that given a good choice of prompt loss weight, long-completion training data may not be necessary for instruction fine-tuning LLMs for language understanding and short generation tasks. Conversely, using long-completion data should be more robust as prompt loss weight and masking parameters can be ignored.

We further analyzed the AlpacaDataShort models and found that low non-zero PLW discourages deviation from the PTLM weights and high PLW reduces overfitting and encourages longer generation. We suggest that these two factors are responsible for the negative quadratic relationship between performance and PLW.

Finally, we estimated optimal PLWs over a range of completion-prompt length ratios.

## Limitations

1. We analyzed prompt loss weighting (PLW) for instruction fine-tuning LLMs. We characterized three 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. We used a fixed seed $= 42$ for all experiments. Since we used pre-trained models and no layers were freshly initialized, there was little variance in initial experiments. Therefore, we decided to limit runs to a single seed.

3. Suggested values for PLW from section 6 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. LLM-as-evaluator approaches like PandaLM and Alpaca Eval 1 are still relatively new, and these approaches are still being actively developed. We chose to use Mixtral as an auto-evaluator for Alpaca Eval. Although this is not the recommended auto-evaluator, we felt its performance on the human-labeled test set was good enough to justify its use.

5. A more confident conclusion about the causative mechanism could not be made. As with most techniques in machine learning, the suggested causative relationship should be validated through numerous experiments and projects. Unfortunately, since we expect most instruction fine-tuning datasets to be long-completion datasets, further research into PLW will likely be limited.

## 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 do acknowledge that standard deep learning training methods have a high carbon footprint, and we performed a total of 100 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. Our synthetic dataset was generated from another publicly released dataset and we did not perform additional content filtering. A warning about both of these issues will be released along with the model and dataset.

## Acknowledgements

Acknowledgments temporarily removed for anonymity.

## References

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.

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.

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. 2023. 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*.

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*.

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*.

## A Benchmark Performance Figures

This appendix contains individual benchmark performance plots and additional discussion of performance trends and results.

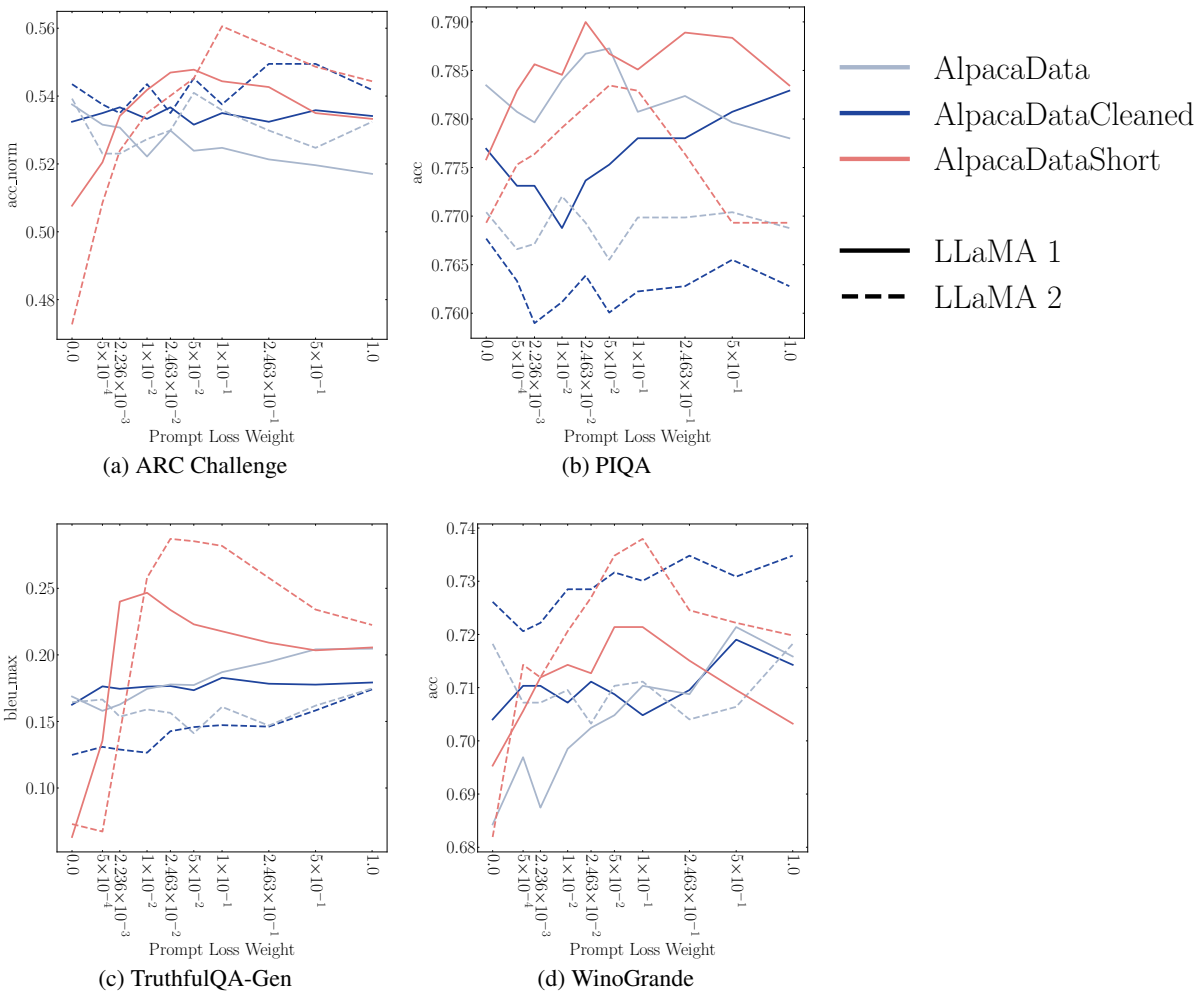
Overall, models trained on AlpacaDataCleaned performed best on the selected benchmarks, with models trained on AlpacaDataShort meeting or outperforming these models on numerous benchmarks for optimal selection of PLW. The general PTLM trend was for LLaMA 2 models to outperform LLaMA 1 models, again with some exceptions. In general, models trained on AlpacaData and AlpacaDataCleaned did not show consistent relationships with PLW on the tested benchmark.

The rest of this section focuses on AlpacaDatashort models. We divide the benchmarks into three groups based on the relationship between AlpacaDataShort model performance and PLW.

Group I benchmarks showed a negative quadratic relationship with the transformed PLW variable $w_p$: ARC Challenge, PIQA, TruthfulQA-Gen, and WinoGrande. See figure 4 for results.

Group II benchmarks showed an increasing relationship between performance and $w_p$ on AlpacaDataShort: TruthfulQA-MC2, Alpaca Eval 1 (Mixtral), and PandaLM. Interestingly, Alpaca Eval 1 (Mixtral) and PandaLM are both long generation benchmarks, but the generation version of the TruthfulQA benchmark is in the first group while the multiple choice benchmark is in this group.

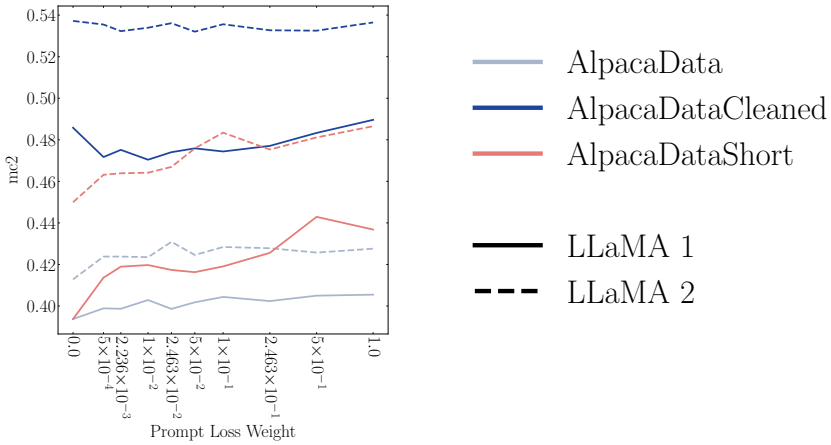
Group III are high noise benchmarks, limited to the six WMT translation tasks. See figure 6 for individual plots. In general WMT tasks showed very small variance, at most varying across only a few percentage points. To reduce the noise, these benchmarks were calculated over both the validation and test sets, but noise still precludes making clear conclusions about performance on any benchmark. Of note, combining benchmarks that translate from English produces a trend line suggestive of a positive relationship with $w_p$. See figure 6a for an example. Figure 6b portrays into English translation benchmarks, but no trends could be identified.



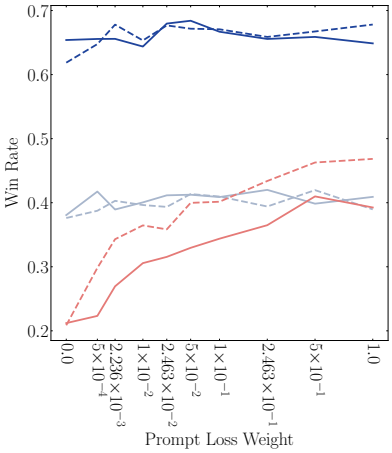

(a) ARC Challenge (b) PIQA

(c) TruthfulQA-Gen (d) WinoGrande

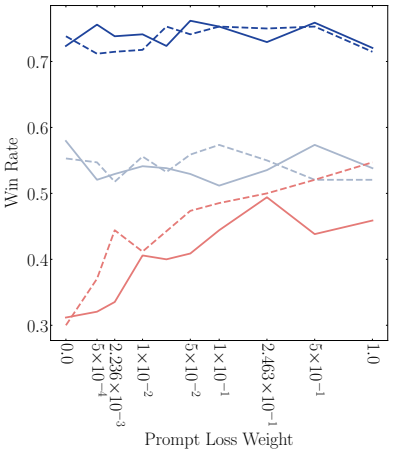
Figure 4: Group I benchmark performance. Note the negative quadratic relationship with transformed PLW.




(a) TruthfulQA-MC2

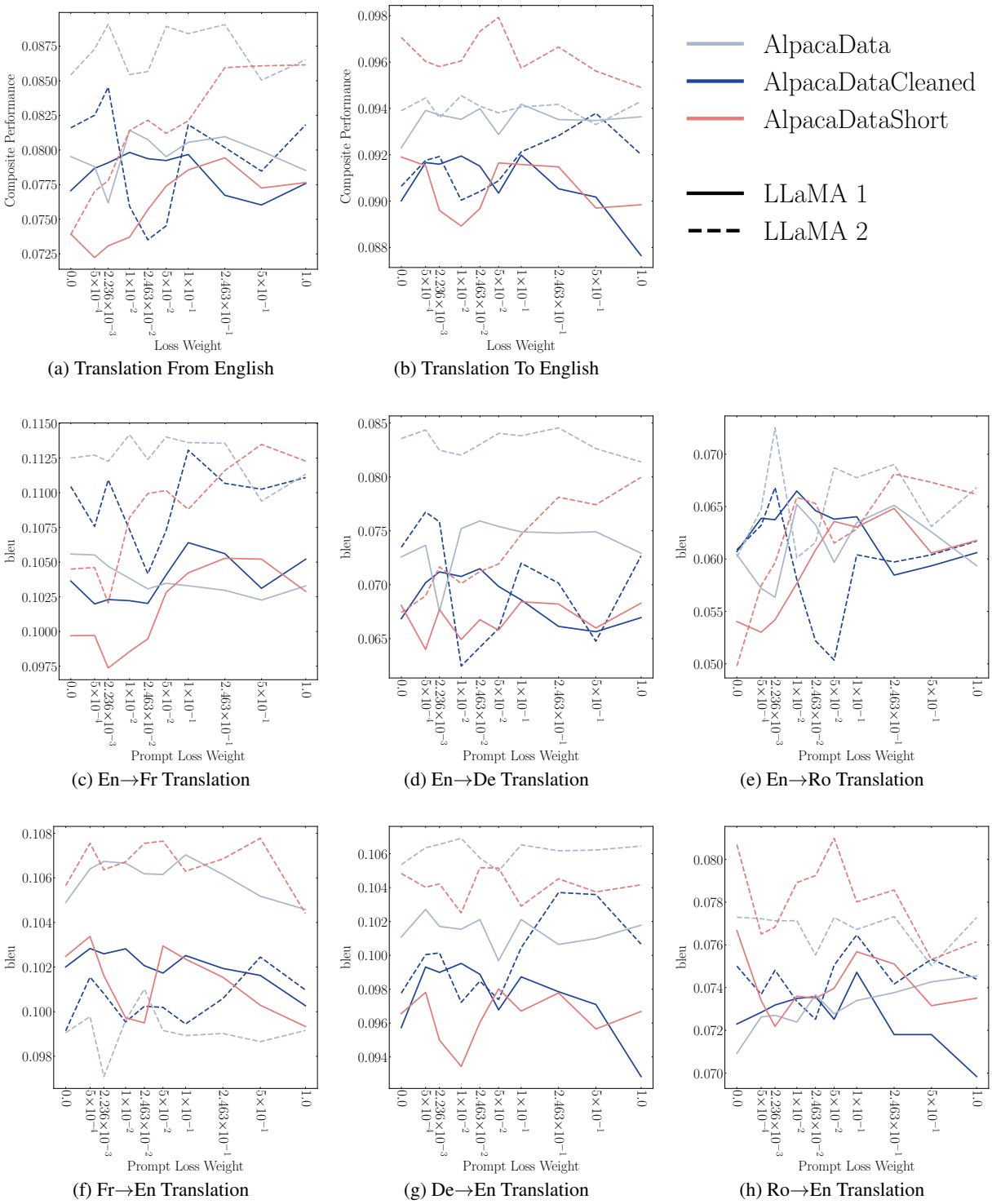



(b) Alpaca Eval (AE) v1




(c) PandaLM

Figure 5: 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 6: Group III benchmarks showed little relationship between performance and PLW.

## B Recommended Prompt Loss Weight

In this section, we present additional figures and tables related to optimal prompt loss weights (PLWs) as predicted by fitted generalized additive models (GAMs).

The optimal PLW predictions from the paper body were based on all benchmarks. Here we present three different relative aggregates: all benchmarks (labeled "All"), multiple choice benchmarks (labeled "MC"), and non-translation generation benchmarks (labeled "Gen"). We found that the translation benchmarks contributed little to the predictive power of the fitted GAMs. In an attempt to predict PLW values in a more relevant context, we did not include the translation benchmarks in the "Gen" model. However, they are still included in the "All" model to capture a wider range of tasks.

See figures 7,8,9 for contour plots for the "All", "MC", and "Gen" benchmarks, respectively. The "All" contour plot is identical to the contour plot in the main body.

See table 4 for a list of optimal PLWs over a range of completion-prompt ratios. Note that the predicted optimal PLWs confirm the conclusions from our regression analysis in section 5.2. Namely that PLW is important for short-completion data, but can be ignored for long-completion data.

| $R_g$ | 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 4: 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.

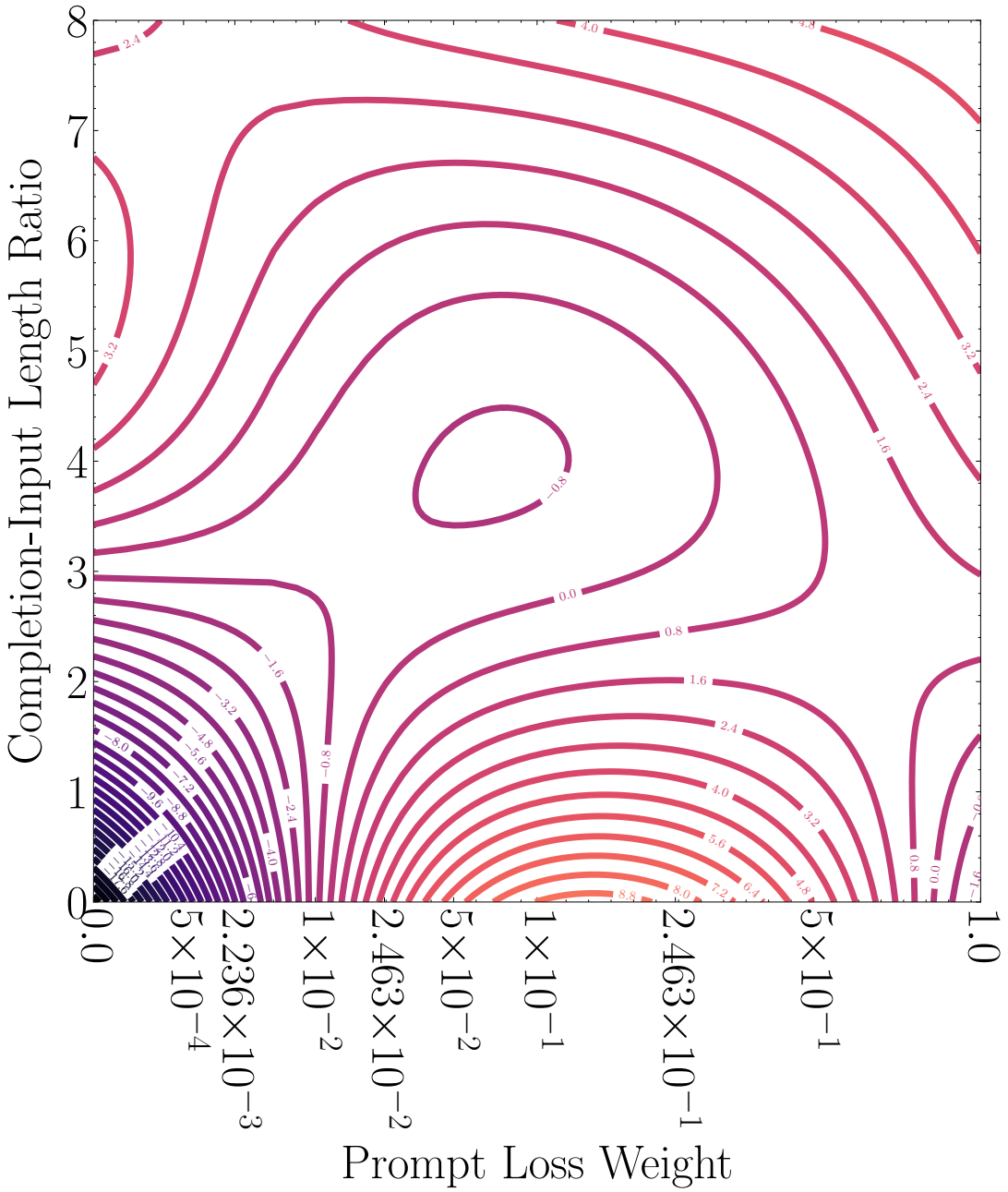



Figure 7: “All” Ratio-PLW Interaction for all benchmarks.

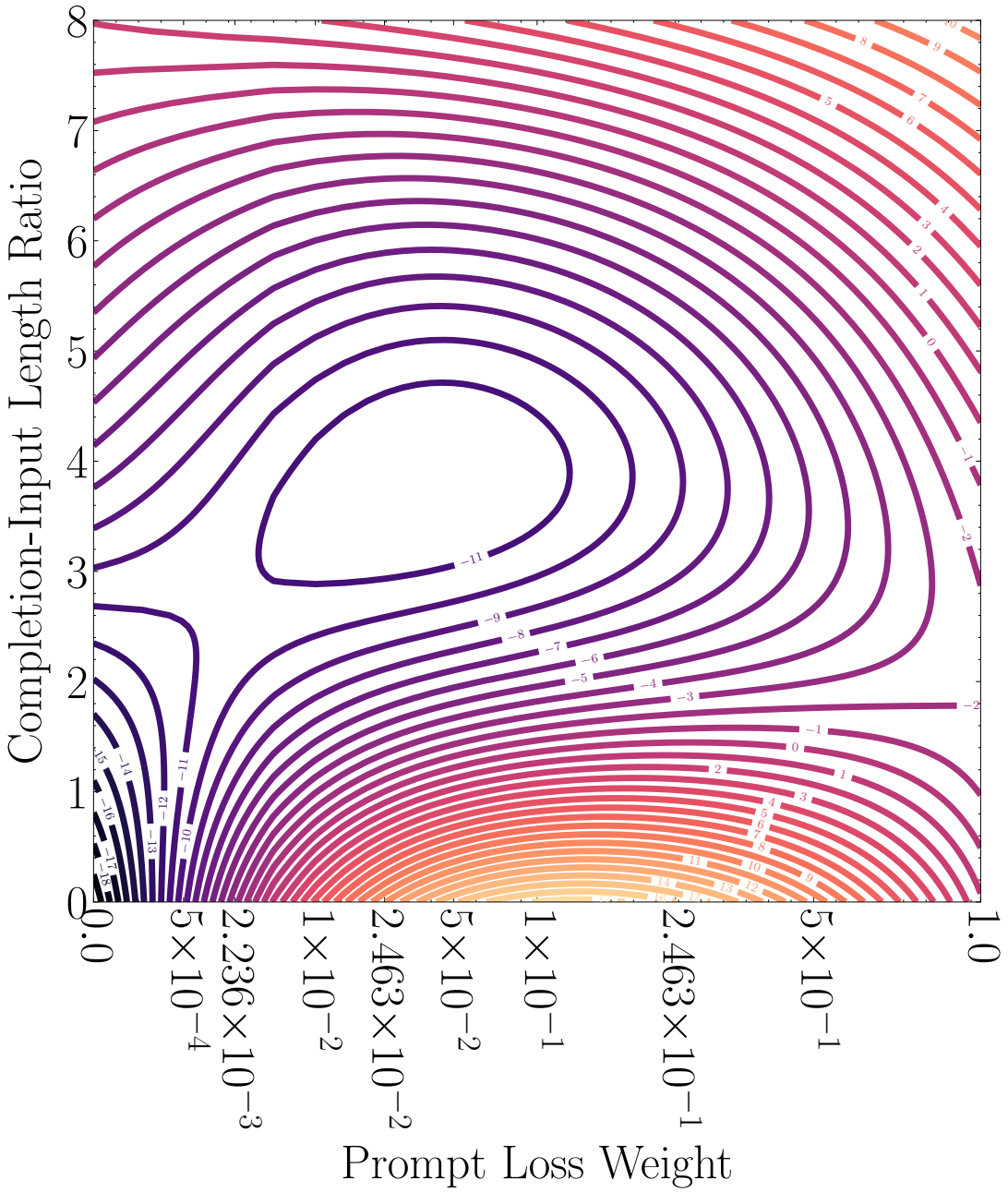



Figure 8: "MC" Ratio-PLW Interaction for multiple choice benchmarks.

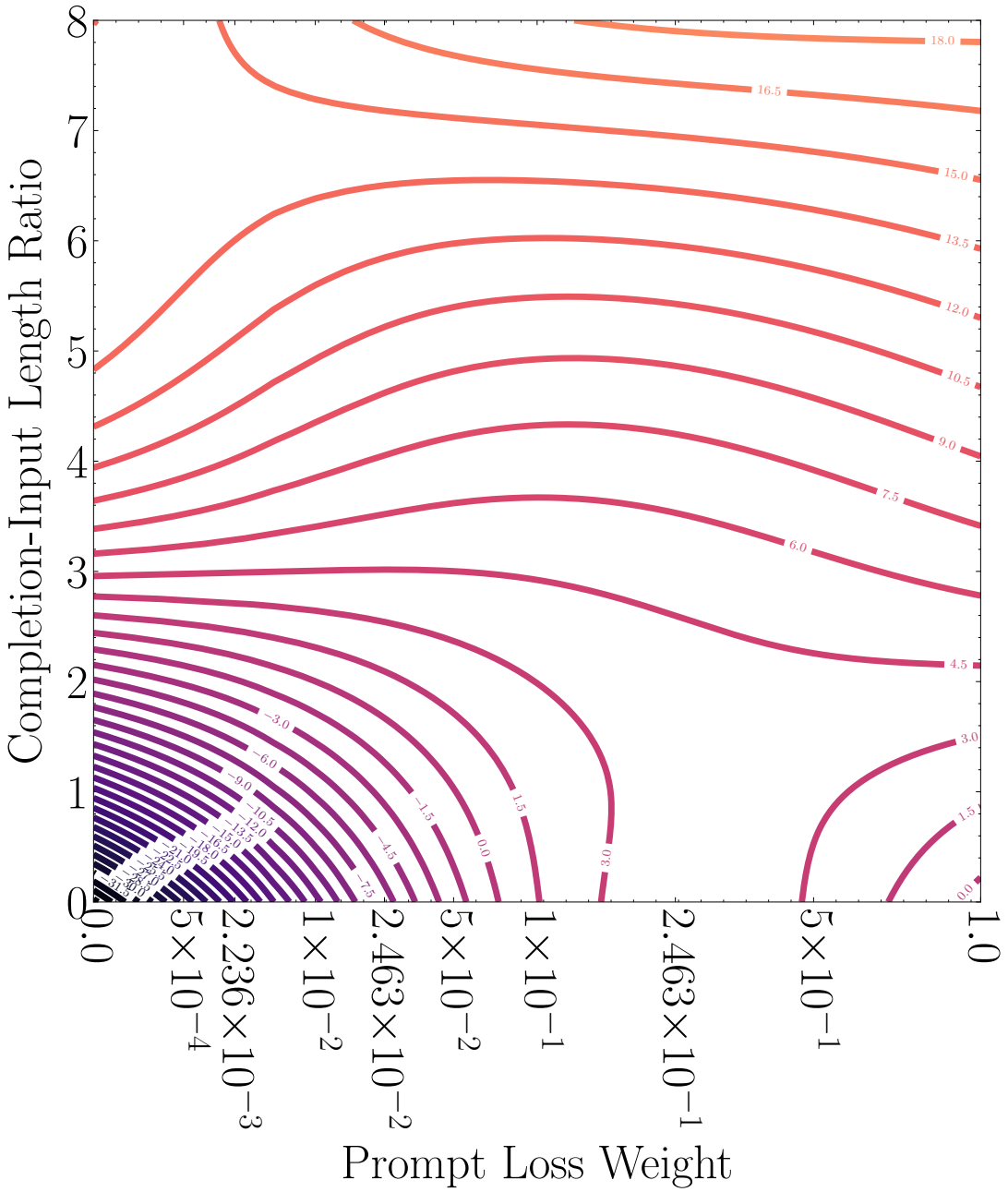



Figure 9: “Gen” Ratio-PLW Interaction for non-translation generation benchmarks.

## C Reproducibility

This section provides technical details on all experiments and benchmarks for transparency and to encourage reproduction of results. To help with reproducibility, also uploaded the fine-tuned models, test generation outputs, and the AlpacaDataShort dataset to xx.yy.zz.

### C.1 Model Fine-Tuning

Model fine-tuning was performed with the Stanford Alpaca GitHub repository at www.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 www.github.com/gururise/AlpacaDataCleaned/tree/791174f and is labeled "alpaca_data_cleaned.json". As noted above, we released AlpacaDataShort at xx.yy.zz.

### C.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: www.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 www.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 xx.yy.zz.

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).

### C.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.

### C.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`"

### C.5 Predictive Model

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

## D Artifact Licensing

We respected all licenses for artifacts and resources used in this research. Please see table 5 for an overview of primary resources and licenses.

| Resource | License | Application |
|---|---|---|
| Transformers | Apache 2.0 | Model Training |
| Stanford Alpaca | Apache 2.0 | Model Training |
| LLaMA 1 | LLaMA License | Pre-trained model weights |
| LLaMA 2 | LLaMA 2 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 5: Licenses for resources used in this research.