BETTER AUTOREGRESSIVE REGRESSION WITH LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have proven successful on many machine learning tasks, including those that do not involve language generation. This includes solving regression problems, where the targets are real-numbers. One common approach is to fine-tune the LLM based on the log-perplexity loss, and use autoregressive sampling at the inference time. Another approach relies on adding a separate predictive head, and fine-tuning it with a suitable loss. While each approach has had success, there has not been a study on the principled ways of using decoder LLMs for regression. In this work, we compare different prior works under a unified view, and introduce regression-aware fine-tuning (RAFT), a novel approach based on the Bayes-optimal decision rule. We demonstrate how RAFT improves over established baselines on several benchmarks and model families.

1 INTRODUCTION

Decoder-based large language models (LLMs) (OpenAI et al., 2023; Anil et al., 2023; Touvron et al., 2023; Gemini Team et al., 2024) have set new benchmarks in challenging generative tasks (e.g., summarization, open-ended dialogue). Such models' versatility has further prompted their exploration for classic non-generative tasks (e.g., classification, regression, ranking) (Liu & Low, 2023; Fernandes et al., 2023; Qin et al., 2023; Vacareanu et al., 2024b; Yang et al., 2023; Dukić & Snajder, 2024; Lukasik et al., 2024; Vacareanu et al., 2024a), once the purview of encoder-only models such as BERT (Devlin et al., 2019). Such exploration is expected to increase given the sustained efforts towards building ever-larger decoder-based LLMs, with limited parallels in scaling encoder-based models.

Our interest is in natural language regression, where the goal is to predict a real-valued target given a textual input. This covers important practical applications such as semantic similarity prediction (Cer et al., 2017), quality estimation (Kocmi & Federmann, 2023; Jain et al., 2023; Fernandes et al., 2023), and sentiment analysis (Zhang et al., 2024). Given the discordance between natural language and numbers, one may ask: how do we best apply decoder-based LLMs for natural language regression? Existing works have followed two broad approaches. Autoregressive regression approaches rely on standard LLM decoding to directly predict as text the numerical targets (e.g., predict a number 12.34 by iteratively predicting tokens: `'1'`, `'2'`, `'.'`, `'3'`, `'4'`) (Gruver et al., 2023; Liu & Low, 2023; Yang et al., 2023; Lukasik et al., 2024), or corresponding discretised categories (e.g., predict one of { `"very bad"`, `"bad"`, `"ok"`, `"good"`, `"very good"` }) (Fernandes et al., 2023). Predictive head approaches, inspired from encoder-based models, learn a separate head on encoded inputs, thus side-stepping the autoregressive mechanism inherent to decoder-based LLMs. Common choices of encoding include mean pooling of the output embeddings (Zhuang et al., 2023), and the final-token logit for a special token (e.g., `<extra_id_0>` in T5) (Fernandes et al., 2023).

Both autoregressive and predictive head approaches have proven successful for natural language regression tasks. However, there has been (to our knowledge) no systematic comparison between these methods; further, each of them has a conceptual shortcoming. The autoregressive regression approach does not exploit the numerical nature of the regression targets, and thus does not consider the fact that for a target of 1, predicting 11 is worse than predicting 1.1. On the other hand, the predictive head approach deviates from the pre-training objective typically used in decoder-based LLMs, viz. next-token prediction (Radford et al., 2018), and thus may not use the model in an optimal manner. This prompts us to ask: how can we respect both the LLM pre-training objective and the numerical nature of targets for natural language regression tasks?

Approach	Autoregressive	Fine-tuning	Inference	References
Zero-shot decoding	✓	None	standard decoding	Kocmi & Federmann (2023)
Fine-tuning and decoding	✓	log-perplexity	standard decoding	Fernandes et al. (2023)
Zero-shot MALI	✓	None	metric-aware decoding	Lukasik et al. (2024)
Fine-tuning and MALI	✓	log-perplexity	metric-aware decoding	this work
Predictive head	✗	target metric	point estimate	Fernandes et al. (2023)
RAFT	✓	target metric	metric-aware decoding	this work

Table 1: Summary of the approaches to applying decoder-based LLMs to natural language regression tasks. There are previous works relying either on using the model autoregressively (i.e., analogously to how it was pre-trained) or as an encoder (i.e., an output is constructed based on embeddings or logits obtained for the inputs). Different training and inference approaches have been considered for both autoregressive and encoder based approaches.

In this work, we introduce regression-aware fine-tuning (RAFT), a novel approach to autoregressive regression which makes use of the numerical nature of the targets. We prove theoretical limitations of established alternative approaches to autoregressive-based regression, and prove that RAFT mitigates them. We systematically compare RAFT against autoregressive and predictive head baselines, and consider several ablations for understanding what design decisions are crucial for making a decoder-based LLM work under different settings. See Table 1 for an overview of both the previous works and the approach introduced in this work. Overall, our contributions are as follows:

(i) We propose regression-aware fine-tuning (RAFT), a novel approach to autoregressive regression, and prove that it mitigates the theoretical limitations of prior works (Section 3).

(ii) We present a unified view of decoder-based LLM regression approaches, capturing both the autoregressive and the prediction head approaches, and explicating their limitations (Section 4).

(iii) We systematically compare autoregressive regression baselines, predictive head and RAFT approaches across mulitple datasets and LLMs. We also conduct a series of extensive experiments for pinpointing the sources of differences in the performance between different approaches, explicating what design choices make RAFT so effective (Section 5).

2 Background

We first introduce notation and review previous works on applying decoder-based LLMs to regression.

2.1 Notation

For a finite vocabulary $\mathcal{V}$ of tokens (e.g., words in English), let $\mathcal{X} \subseteq \mathcal{V}^*$ be a set of inputs comprising strings of tokens, and $\mathcal{Y} \subset \mathbb{R}$ be a set of real-valued targets. We assume that each $y \in \mathcal{Y}$ has a unique string representation $\mathtt{str}(y) \in \mathcal{V}^*$; for example, the integer 1 has the string encoding "1". Let $\mathbb{P}$ denote a ground-truth distribution over $\mathcal{X} \times \mathcal{Y}$, with the decomposition $\mathbb{P}(x, y) = \mathbb{P}(x) \cdot \mathbb{P}(y \mid x)$. The natural language regression problem involves learning a predictor $\hat{y} \colon \mathcal{X} \to \mathbb{R}$ that minimises the mean squared error over (input, target) pairs drawn from $\mathbb{P}$:

$$L(\hat{y}) = \mathbb{E}_{(x,y^*)\sim\mathbb{P}}\left[(y^* - \hat{y}(x))^2\right].$$

Note that the mean squared error is a canonical choice in regression (Fernandes et al., 2023). The Bayes-optimal predictor minimizing the above is $\hat{y}(x) = \mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}[y^*]$.

We seek to employ large language models (LLMs) for such regression tasks. An LLM specifies a distribution p over strings in $\mathcal{V}^*$. Given an input $x \in \mathcal{X}$, let $p(\cdot \mid x)$ denote the corresponding conditional distribution over possible continuations. Note that it may be possible for $p(z \mid x) > 0$ where $z \in \mathcal{V}^*$ does not have a numerical representation; we discuss this issue more in Section 2.2.

LLMs are typically pre-trained on large corpora via self-supervised objectives (Radford et al., 2018), and can perform few-shot or in-context learning given suitably crafted prompts (Brown et al., 2020). For example, if the goal is to predict the probability that a user will enjoy a movie titled "`Cure`", we may construct an input x = "`Hereditary: 0.7 | Ringu: 0.9 | Cure: `", and probe the LLM's estimate of plausible continuations via $p(\cdot \mid x)$. We next discuss inference (or decoding) procedures for deriving a predictor $\hat{y}$ given a pre-trained LLM.

2.2 Standard decoding for regression

Standard decoding involves predicting numerical targets in a generative manner, by performing autoregressive decoding to draw a sample from the distribution $p(\cdot \mid x)$:

$$\begin{aligned} \hat{y}_{\mathrm{AR}}(x) &\doteq \texttt{float}(z) \\ z &\sim p(\cdot \,|\, x). \end{aligned} \tag{1}$$

Here, $z \in \mathcal{V}^*$ is generated autoregressively on a token-by-token basis. Different algorithms may be used for this generation, e.g., greedy decoding and temperature sampling (Naseh et al., 2023). In many cases, such algorithms are seeking to approximate the mode of the distribution:

$$\hat{y}_{\mathrm{mode}}(x) := \underset{y \in \mathcal{Y}}{\arg\max}\; p(y \mid x). \tag{2}$$

Further, $\texttt{float}(z)$ denotes an operator that converts a given string z (e.g., "`12.34`") to a corresponding numeric value (e.g., 12.34); if z does not have a numeric representation (e.g., "`banana`"), then we assume that a suitable default value is returned. Unless otherwise stated, we assume $\texttt{float}(z) = 0.0$ for $z \notin \mathcal{Y}$. As an alternative, one may choose to restrict the output space to numerical targets, e.g., by employing a form of constrained decoding (Geng et al., 2023). However, in practice, the targets from high-quality LLMs tend to be numerical even under zero-shot settings (Lukasik et al., 2024).

2.3 MALI: Metric-aware LLM inference for regression

Recently, Lukasik et al. (2024) pointed out a limitation of decoding the most likely target when employing autoregressive models for regression. Decoding of the most likely targets can be shown to minimize the 0-1 loss $\ell(y, \hat{y}) = \mathbb{1}(y \neq \hat{y})$, and may not be well aligned with regression metrics such as squared loss. As a remedy, instead of autoregressive decoding per Equation 1, Lukasik et al. (2024) proposed the MALI method, which given a loss ℓ and model prediction $p(\cdot \mid x)$ estimates the Bayes-optimal output minimizing the expected loss:

$$\hat{y}_{\mathrm{MALI}}(x) = \underset{v \in \mathbb{R}}{\arg\min}\, \mathbb{E}_{y \sim p(\cdot|x)} \left[\ell(\texttt{float}(y), v) \right], \tag{3}$$

where $\texttt{float}(\cdot)$ is as per the previous section. For the squared loss $\ell(y, \hat{y}) = (y - \hat{y})^2$, the optimal decision rule can be shown to take the following closed-form solution:

$$\hat{y}_{\mathrm{MALI}}(x) = \mathbb{E}_{y \sim p(\cdot|x)} \left[\texttt{float}(y) \right]. \tag{4}$$

Since $p(\cdot \mid x)$ is a distribution over all possible strings, it is typically intractable to compute the above expectation exactly; this remains true even if we restrict attention to those strings corresponding to a valid numerical value (of which there are infinitely many). In practice, Equation 4 can be estimated either via sampling a finite number of y values, or via scoring of targets. In the latter, suppose we have some restricted target grid $\mathcal{Y}_{\mathrm{grid}} \subset \mathcal{Y}$. Then, the MALI predictor is averaged over $\mathcal{Y}_{\mathrm{grid}}$, yielding:

$$\hat{y}_{\mathrm{MALI}}(x; \mathcal{Y}_{\mathrm{grid}}) = \sum_{y \in \mathcal{Y}_{\mathrm{grid}}} p(\texttt{str}(y) \mid x) \cdot y. \tag{5}$$

Note that $\sum_{y \in \mathcal{Y}_{\mathrm{grid}}} p(\texttt{str}(y) \mid x) \neq 1$ is possible, so the above is technically not an expectation; however, in practice, high-quality LLMs tend to concentrate most mass on numerical targets.

There are several choices of $\mathcal{Y}_{\mathrm{grid}}$ available to the practitioner. For discrete targets $\mathcal{Y}$ of moderate size, one may simply choose $\mathcal{Y}_{\mathrm{grid}} = \mathcal{Y}$. For bounded $\mathcal{Y}$, one choice is equally spaced targets covering the range of $\mathcal{Y}$, e.g. integers or fixed-precision numbers (e.g. 2 decimal points) (Lukasik et al., 2024).

The above approaches operate on a pre-trained LLM via few-shot prompting. However, it has been consistently observed that direct fine-tuning of LLMs on the task of interest can be beneficial (Liu et al., 2022). We now consider how to perform LLM fine-tuning for regression tasks.

3 Regression-aware LLM fine-tuning

We now develop a theoretically grounded approach to regression with decoder-based LLMs in the fine-tuning setting. We begin by reviewing the standard fine-tuning setup, and explicate its limitations. We give all proofs in Appendix B.

3.1 Standard fine-tuning

Fine-tuning seeks to adapt an LLM to the target distribution $\mathbb{P}$ by minimizing

$$L(p) = \mathbb{E}_{(x,y^*)\sim\mathbb{P}}\left[\ell(y^*, p(\cdot \mid x))\right] \tag{6}$$

for a suitable loss function $\ell \colon \mathcal{Y} \times \Delta_{\mathcal{V}^*} \to \mathbb{R}$, where Δ_S denotes the set of distributions over a set S. Given a sample $S \in (\mathcal{X} \times \mathcal{Y})^N$ of N (input, target) pairs drawn from $\mathbb{P}$, the empirical loss is

$$\hat{L}(p) = \frac{1}{N} \sum_{(x,y^*)\in S} \ell(y^*, p(\cdot \mid x)). \tag{7}$$

A standard choice of ℓ is the log-loss (also referred to as log-perplexity):

$$\ell(y^*, p(\cdot \mid x)) = -\log p(\mathtt{str}(y^*) \mid x), \tag{8}$$

recalling that $\mathtt{str}(y^*)$ denotes the string representation of a numeric target $y^* \in \mathbb{R}$. More generally, one may use categorical descriptions of the target after discretising to some finite grid $\mathcal{Y}_{\mathrm{grid}} \subset \mathcal{Y}$; e.g., { `"very bad"`, `"bad"`, `"ok"`, `"good"`, `"very good"` } (Fernandes et al., 2023).

3.2 Limitations of standard fine-tuning and standard decoding

A natural baseline is to employ log-perplexity based fine-tuning by minimizing Equation 7, and to then apply standard decoding (see Equation 2). Since the log-loss is strictly proper, minimizing Equation 7 recovers the Bayes distribution $\mathbb{P}(\cdot \mid x)$ in the population limit (Gneiting & Raftery, 2007). In practice, however, the fine-tuned model distribution $p(\cdot \mid x)$ may deviate from $\mathbb{P}(\cdot \mid x)$. The following lemma shows that even small deviations from $\mathbb{P}$ can cause the predictor to incur a high squared error compared to the Bayes-optimal predictor.

Lemma 1. Assume $|\mathcal{Y}| \geq 2$ and $0 \in \mathcal{Y}$, with $N \doteq \max(\mathcal{Y})$. For any $\epsilon > 0$, there exists $\mathbb{P}, p$ such that: $\mathbb{E}_x\left[\|\mathbb{P}(\cdot \mid x) - p(\cdot \mid x)\|_1\right] \leq \epsilon$, and $\mathbb{E}_x\left[\left(\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}[y^*] - \hat{y}_{\mathrm{mode}}(x)\right)^2\right] \geq \left(\frac{N}{2}\right)^2\left(1 + \frac{\epsilon}{2}\right)^2$.

Thus, using the log-perplexity fine-tuning with standard decoding is not well-aligned with the eventual goal of approximating $\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}[y^*]$.

3.3 Limitations of standard fine-tuning and MALI decoding

Given that MALI performs well in few-shot regression tasks, a natural means of further improving its performance is to employ log-perplexity based fine-tuning by minimizing Equation 7, and to then apply the MALI decoding (see Equation 5).

However, as before, we can show that the predictor can significantly deviate from the optimal prediction.

Lemma 2. Assume $|\mathcal{Y}| \geq 2$ and $0 \in \mathcal{Y}$, with $N \doteq \max(\mathcal{Y})$. For any $\epsilon > 0$, there exists $\mathbb{P}, p$ such that: $\mathbb{E}_x\left[\|\mathbb{P}(\cdot \mid x) - p(\cdot \mid x)\|_1\right] \leq \epsilon$, and $\mathbb{E}_x\left[\left(\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}[y^*] - \hat{y}_{\mathrm{MALI}}(x)\right)^2\right] \geq \left(\frac{\epsilon N}{2}\right)^2$.

Thus again, using the log-perplexity fine-tuning with MALI may not be well-aligned with the eventual goal of approximating $\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}[y^*]$. Intuitively, log-perplexity fine-tuning treats all "wrong" predictions the same, as it is unaware of the difference in the magnitude of the numerical values represented by the tokens. For example, assuming that numbers 100 and 1000 are represented with a single token, placing too much mass on the token representing 100 is penalized similarly as placing too much mass on the token 1000.

One solution to the above issue is to directly employ the MALI predictor in the fine-tuning process. This requires going beyond the log-loss in Equation 8, as we now detail.

3.4 RAFT: REGRESSION-AWARE FINE-TUNING

To overcome the drawbacks of using MALI with traditional fine-tuning, we propose a novel regression-aware objective that seeks to directly minimize the squared loss on the MALI predictor:

Definition 1. Define the regression-aware fine-tuning (RAFT) loss as follows,

$$\ell_{\mathrm{RAFT}}(y^*, p(\cdot \mid x)) = \left(y^* - \mathbb{E}_{y \sim p(\cdot|x)}[\mathtt{float}(y)]\right)^2. \tag{9}$$

Equally, this uses the MALI predictor $\hat{y}_{\mathrm{MALI}}(x)$ to construct a numeric value from the LLM, and measures the square loss against the target y^*. Given a finite grid $\mathcal{Y}_{\mathrm{grid}} \subset \mathcal{Y}$ and fine-tuning set S, the empirical loss corresponding to Equation 9 is:

$$\hat{L}_{\mathrm{RAFT}}(p; \mathcal{Y}_{\mathrm{grid}}) = \frac{1}{N} \sum_{(x,y^*) \in S} \left(y^* - \sum_{y \in \mathcal{Y}_{\mathrm{grid}}} p(\mathtt{str}(y) \mid x) \cdot y \right)^2. \tag{10}$$

Note that computing this loss only requires scoring each $y \in \mathcal{Y}_{\mathrm{grid}}$ under the model; we do not need to perform any explicit sampling or decoding during training.

Compared to standard fine-tuning with MALI decoding, note here that we attempt to avoid the issue in Lemma 2 by directly minimizing $\mathbb{E}_x \left[\left(\mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)}[y^*] - \hat{y}_{\mathrm{MALI}}(x) \right)^2 \right]$. Surprisingly, despite computing $\hat{y}_{\mathrm{MALI}}(x)$ over the restricted target space $\mathcal{Y}_{\mathrm{grid}}$, under mild conditions the minimizer of Equation 9 exactly mimics the Bayes-optimal predictor over the full space $\mathcal{Y}$.

Lemma 3. Suppose $\mathcal{Y} \subset \mathbb{R}$, and $\arg\min_{y \in \mathcal{Y}} \in \mathcal{Y}_{\mathrm{grid}}$ and $\arg\max_{y \in \mathcal{Y}} \in \mathcal{Y}_{\mathrm{grid}}$. Let $p^*(\cdot \mid x)$ be the minimizer of the RAFT loss from definition 1 over all distributions $p(\cdot \mid x)$. Then the MALI predictor $\hat{y}_{\mathrm{MALI}}(x; \mathcal{Y}_{\mathrm{grid}}) = \sum_{y \in \mathcal{Y}_{\mathrm{grid}}} p^*(\mathtt{str}(y) \mid x) \cdot y$ constructed from $p^*(\cdot \mid x)$ satisfies:

$$\hat{y}_{\mathrm{MALI}}(x; \mathcal{Y}_{\mathrm{grid}}) = \mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)}[y^*].$$

The intuition behind this result is that any numerical target in $\mathcal{Y}$ can be expressed by a convex combination of the smallest and largest numbers in $\mathcal{Y}$, and can thus be realized by the MALI predictor.

4 A UNIFIED VIEW OF LLM-BASED REGRESSION APPROACHES

We next compare and contrast RAFT and the predictive head approaches in a unified view. In Table 2, we present a summary of different choices for the predictors considered in prior and the present work.

4.1 PREDICTIVE HEAD APPROACHES TO REGRESSION

Predictive head approaches formulate a predictor function $\hat{y}(x)$ by utilizing activations or embeddings from the forward pass of LLM. Abstractly, such approaches first extract a suitable input representation $\Phi(x) \in \mathbb{R}^q$, which is then fed into a regressor $s\colon \mathbb{R}^q \to \mathbb{R}$. Canonically, the regressor is simply a linear model $s(\Phi(x)) = b + w^\top \Phi(x)$ for learnable $w \in \mathbb{R}^q, b \in \mathbb{R}$, but one may also consider an MLP with a single real-valued output.

Various choices for $\Phi(x)$ have been considered in previous works. To describe these, we need some additional notation. Given a string $x \in \mathcal{V}^*$ of length L, a Transformer-based language model (Vaswani et al., 2017) first constructs an input embedding $\epsilon_{\mathrm{in}}(x) \in \mathbb{R}^{D \times L}$, via a matrix $E_{\mathrm{in}} \in \mathbb{R}^{D \times L}$ of token embeddings: concretely, $\epsilon_{\mathrm{in}}(x) = E_{\mathrm{in}} \epsilon_{\mathrm{oh}}(x)$, where $\epsilon_{\mathrm{oh}}(x) \in \mathbb{R}^{V \times L}$ is the one-hot embedding of each token in x. Next, this input embedding is passed through a stack of attention and MLP layers, to produce the output embedding $\epsilon_{\mathrm{out}}(x) \in \mathbb{R}^{D \times L}$. One further projects this to the vocabulary space to produce output logits $f_{\mathrm{out}}(\cdot \mid x) \in \mathbb{R}^{V \times L}$, where $f_{\mathrm{out}}(\cdot \mid x) = E_{\mathrm{out}}^\top \epsilon_{\mathrm{out}}(x)$. Finally, one transforms these to a distribution over possible tokens via the softmax operator, yielding $p_{\mathrm{out}}(\cdot \mid x) = \mathrm{softmax}(f_{\mathrm{out}}(\cdot \mid x)) \in [0,1]^{V \times L}$. For certain models (e.g., Gemma), the input and output vocabulary embedding matrices are tied, i.e., $E_{\mathrm{in}} = E_{\mathrm{out}}$.

Given the above, one may extract an input representation through multiple means, most commonly pooling or selection of the output token embeddings, output logits, or output probabilities. Common

Category	**Approach**	**Predictor function** $\hat{y}(x)$	**Fine-tuning loss**
Autoregressive (prior works)	Standard decoding zero-shot (Kocmi & Federmann, 2023)	$\arg\max_{y \in \mathcal{Y}} p(y \mid x)$	N/A
	MALI zero-shot (Lukasik et al., 2024)	$\sum_{y' \in \mathcal{Y}} y' \cdot p(y' \mid x)$	N/A
	Standard fine-tuning and standard decoding (Fernandes et al., 2023)	$\arg\max_{y \in \mathcal{Y}} p(y \mid x)$	$-\log p(y^* \mid x)$
Autoregressive (this work)	MALI standard fine-tuning	$\sum_{y' \in \mathcal{Y}} y' \cdot p(y' \mid x)$	$-\log p(y^* \mid x)$
RAFT (this work)	General RAFT	$\sum_{y' \in \mathcal{Y}} y' \cdot p(y' \mid x)$	$(\hat{y}(x) - y^*)^2$
	Single-digit RAFT	$\sum_{y' \in \mathcal{Y}_{digit}} y' \cdot p_{\text{out}}(x)_{y',L}$	$(\hat{y}(x) - y^*)^2$
Predictive head (prior works)	Final-token logit (Fernandes et al., 2023)	$b + f_{\text{out}}(\cdot \mid x)_{v^*,L}$	$(\hat{y}(x) - y^*)^2$
	Pooled output embeddings (Zhuang et al., 2023)	$b + w^\top \texttt{pool}(\epsilon_{\text{out}}(x))$	$(\hat{y}(x) - y^*)^2$
Predictive head (this work)	MLP on the final-token logits	$b + \text{MLP}((E_{\text{out}})^\top \epsilon_{\text{out}}(x)_{:,L})$	$(\hat{y}(x) - y^*)^2$
	Probability-vector projection	$b + w^\top p_{\text{out}}(x)_{:,L}$	$(\hat{y}(x) - y^*)^2$
	Learnable regression-aware training	$\sum_{y' \in \mathcal{Y}} w_{y'} \cdot p_{\text{out}}(x)_{y',L}$	$(\hat{y}(x) - y^*)^2$

Table 2: Different approaches for applying decoder-based LLMs to regression. Here, $p(\cdot \mid x)$ denotes a distribution over possible outputs given an input string x, and $\hat{y}(x) \in \mathbb{R}$ a predictor given by a predictive head approach. b and w are learnable parameters, $v^* \in \mathcal{V}$ is a fixed token, `pool` is a pooling operator (such as taking a per-dimension average), and L is the length of input x, $\mathcal{Y}_{digits}$ denotes all digits covering the range of targets (unless otherwise stated, '1'-'5'). The first 4 rows show the autoregressive baselines: standard decoding (Section 2.2), MALI zero-shot (Section 2.3), standard fine-tuning and decoding (Section 3.1), MALI with standard fine-tuning (Section 3.3). The next 2 rows show RAFT: the general autoregressive form ($\mathcal{Y} = \mathcal{Y}_{\text{grid}}$ for general output spaces), and the single digit version (e.g. $\mathcal{Y} = \{1, 2, 3, 4, 5\}$). The following 2 rows present the prior works from Fernandes et al. (2023); Zhuang et al. (2023). The last 3 rows present new predictive head approaches that attempt to mimic the behavior of RAFT.

pooling strategies include mean-pooling, and passing through an attention operator; common selection strategies include picking the value corresponding to the final token. For example, we may consider the final-token logit activation for a special token $v_* \in \mathcal{V}$ (Fernandes et al., 2023; Zhuang et al., 2023), or mean-pooling the output token embeddings (Zhuang et al., 2023).

Given a suitable predictor, one may directly optimize the mean squared error during fine-tuning; i.e., given a fine-tuning set S, we minimize

$$\hat{L}_{\text{PH}}(\hat{y}) = \frac{1}{N} \sum_{(x,y^*) \in S} (y^* - \hat{y}(x))^2.$$

Compared to autoregressive baselines, an important distinction is that that no autoregressive decoding is conducted at inference.

4.2 RAFT versus predictive head approaches

Our discussion of RAFT highlighted its close relation to autoregressive MALI decoding, which appears rather different to predictive head approaches. However, in the case of a single-digit grid $\mathcal{Y}_{\text{grid}}$ (wherein each element corresponds to a single token in $\mathcal{V}$), the predictor function formulations for RAFT bears similarities to the predictive head approaches. Note that if $y \in \mathcal{Y}_{\text{grid}}$ corresponds to a single token, by definition $p(\texttt{str}(y) \mid x) = p_{\text{out}}(\cdot \mid x)_{\texttt{str}(y),L}$. Then, the MALI predictor becomes

$$\hat{y}_{\text{MALI}}(x; \mathcal{Y}_{\text{grid}}) = \sum_{y \in \mathcal{Y}_{\text{grid}}} y \cdot p(\texttt{str}(y) \mid x) = \sum_{y \in \mathcal{Y}_{\text{grid}}} y \cdot p_{\text{out}}(\cdot \mid x)_{\texttt{str}(y),L}.$$

We may compare this with the final-token logit activation method from Table 2. Both take the following form for an activation Ψ and weight vector $w \in \mathbb{R}^{\mathcal{V}}$:

$$\hat{y}(x) = b + w^\top \Psi \left(f_{\text{out}}(\cdot \mid x)_{:,L} \right).$$

Importantly, RAFT predictor at initialization corresponds to MALI, and thus, forms a strong predictor for a zero-shot inference with LLMs (Lukasik et al., 2024). Most alternative predictive head approaches will incur a high error at initialization due to deviating from the next token prediction task.

Therefore, RAFT can be seen as a predictive head approach with strong performance at initialization, potentially making optimization easier.

Contrasting the RAFT and the final-token logit method, we observe the following differences:

- Activation: for single-digit RAFT, Ψ is the softmax activation that converts $f_{\text{out}}(\cdot \mid x)$ to the probability vector $p_{\text{out}}(\cdot \mid x)$. For the final-token logit, Ψ is the identity activation.
- Weight vector: for the final-token logit, w is a one-hot vector with 1 corresponding to the special token position. For single-digit RAFT, $w_v = \texttt{float}(v)$ for each $v \in \mathcal{V}$; note that, as a result, positions corresponding to non-digits have weight 0.

In light of the close similarities between RAFT and the final-token logit approach, it is prudent to carefully analyze these differences and identify whether any of these choices play an important role in RAFT's performance. Therefore, we introduce the following new predictive head variants:

- MLP on final-token logits: this is a variant of the final-token logit method, wherein a 2-layer MLP with a non-linear activation (sigmoid) is employed on the entire final-token logit vector, rather than selecting the logit for a single special token:
$$\hat{y}(x) = b + \text{MLP}(f_{\text{out}}(\cdot \mid x)_{:,L}).$$
- Learnable-RAFT: this is a variant of RAFT, wherein the weights over the output model probabilities are learned, rather than being fixed to the vector $w_v = \texttt{float}(v)$:
$$\hat{y}(x) = \sum_{y' \in \mathcal{Y}} w_{y'} \cdot p_{\text{out}}(x)_{\texttt{str}(y'),L}$$

The learnable-RAFT variant adds more flexibility to the predictor function $\hat{y}$ over the vanilla RAFT method. However, as with other predictive head methods, it deviates from the next-token prediction pre-training task. Which of these two factors – predictor flexibility, and alignment to the pre-training task – is the most important? To answer this question, we compare learnable-RAFT against RAFT, and also experiment with fine-tuning from a randomly initialized (as opposed to a pre-trained) model.

5 Contrasting the regression methods with autoregressive LLMs

We now present experiments and ablations comparing the autoregressive regression and predictive head approaches on natural language regression datasets. In summary, we make the following main empirical findings: (i) RAFT outperforms all autoregressive regression and predictive head baselines across datasets and models; (ii) Ablations indicate the importance of aligning fine-tuning to the pre-training loss; (iii) RAFT tends to work well when the grid corresponds to digit tokens.

5.1 Experiment settings

Datasets. We use the following natural language regression datasets:

(i) Semantic Textual Similarity Benchmark (STSB) (Cer et al., 2017), which comprises of sentence pairs human-annotated with a similarity score from 0 to 5; To measure the impact of varying the fine-tuning set size, we apart from using the full train set, we also consider using only the first 1'000 examples for training (STSB 1k).

(ii) US Amazon reviews, where we aim to predict the 5-star rating for a product review (Ni et al., 2019);. We consider a few categories from the Amazon reviews datasets, each forming a separate dataset: Wireless, Music, Personal products. We use 1'500 examples for the test set (after Lukasik et al. (2024)), 1'500 for validation and 10'000 examples for training.

(iii) MovieLens-1M, where we construct a movie rating prediction task across users, follow the methodology from (Luo et al., 2024) (see Appendix E.5 for the results).

We summarize the dataset statistics and the prompts in Table 6 and Table 7 (Appendix).

Models. We experiment with Gemma-2 (Team et al., 2024) and PaLM-2 (Anil et al., 2023) instruction-tuned model families of different sizes. We select the best learning rate (from $\{10^{-4}, 10^{-5}\}$) and training step (up to a maximum of $100K$ steps) based on held-out validation set performance. Where standard deviations are reported, fine-tuning is performed 3 times.

Dataset	Model size	Zero-shot standard decoding	Zero-shot MALI	Standard fine-tuning standard decoding	Standard fine-tuning MALI	Predictive head	RAFT
Wireless	2B	0.88	1.42	0.70±0.01	0.67±0.01	0.51±0.01	**0.47±0.01**
	9B	0.87	1.40	0.78±0.05	0.86±0.03	0.46±0.00	**0.45±0.00**
Personal care	2B	0.98	1.75	0.77±0.01	0.74±0.01	0.52±0.02	**0.49±0.00**
	9B	0.95	1.73	0.73±0.14	0.59 ±0.01	0.48±0.01	**0.47±0.01**
Music	2B	1.28	2.46	1.16±0.11	0.88±0.12	0.52±0.00	**0.50±0.00**
	9B	1.28	2.46	0.83±0.35	0.61±0.02	0.50±0.00	**0.47±0.00**
STSB 1k	2B	1.10	0.94	0.61±0.01	0.65±0.02	**0.58±0.01**	**0.58±0.00**
	9B	1.31	0.99	0.57±0.01	0.62±0.05	0.57±0.01	**0.56±0.01**
STSB	2B	1.10	0.94	0.59±0.01	0.61±0.02	**0.54±0.00**	**0.54±0.01**
	9B	1.31	0.99	0.58±0.00	0.58±0.02	0.52±0.00	**0.51±0.01**

Table 3: RMSE across datasets, methods, and Gemma-2 models of varying sizes. Fine-tuning methods report mean $\pm$ std dev from model retraining. See Table 9 (Appendix) for Gemma-2 27B.

Methods. We compare the following methods: (1) autoregressive baselines (Section 2.2), MALI zero-shot (Section 2.3), MALI with log-perplexity fine-tuning (Section 3.3); (2) predictive head approaches from Fernandes et al. (2023); Zhuang et al. (2023); (3) the new RAFT method (Section 3.4); (4) new predictive head approaches that attempt to mimic the behavior of RAFT (Section 4.2). In zero-shot standard decoding, we use greedy decoding; in zero-shot MALI, we use the scoring variant (Lukasik et al., 2024) over the default grid from RAFT. We also run ablations with replacing causal attention masking with bi-directional attention masking, following previous works on classification with decoder-based LLMs (Dukić & Snajder, 2024; Qorib et al., 2024).

Implementation of the RAFT objective. An important practical consideration is how to choose the grid $\mathcal{Y}_{\text{grid}}$. Targets from both Amazon and STSB dataset families belong to $[0, 5]$. Unless otherwise stated, we choose the grid $\mathcal{Y}_{\text{grid}} = \{$ `'1', '2', '3', '4', '5'` $\}$. For Amazon reviews datasets, $\mathcal{Y}_{\text{grid}} = \mathcal{Y}$, while for STSB, $\mathcal{Y}_{\text{grid}} \subset \mathcal{Y}$ (as the targets take floating point values). Recall that Lemma 3 shows that RAFT can represent floating point targets even under such a choice for $\mathcal{Y}_{\text{grid}}$. Nonetheless, we empirically verify whether the choice of $\mathcal{Y}_{\text{grid}}$ impacts the results.

5.2 RAFT LEADS TO BETTER AUTOREGRESSIVE REGRESSION

We compare different autoregressive and prediction head approaches across across Gemma-2 models of varying sizes in Table 3. We report additional results from PaLM-2 models on STSB in Table 10 (Appendix) to verify the findings across an additional model family. We make several observations.

First, we verify the importance of both (1) the use of an appropriate decision rule at inference time (greedy versus metric-aware inference), and (2) the value of fine-tuning over zero-shot inference. Indeed, we find that the zero-shot greedy decoding, MALI (see Section 2.3), autoregressive fine-tuning with greedy decoding (see Section 3.3) and autoregressive fine-tuning with MALI inference (see Section 3.4) work increasingly better.

Second, we find that the predictive head approach outperforms the autoregressive baselines, including those that perform autoregressive fine-tuning. This corroborates Lemma 2, which pointed at the limitations of autoregressive fine-tuning due to it being misaligned with the squared error.

Finally, we find RAFT to outperform the predictive head and the autoregressive approaches by a large margin across almost all settings. RAFT outperforming the autoregressive approaches corroborates the posited importance of aligning the fine-tuning loss in regression tasks to a regression loss. RAFT outperforming the predictive head approach corroborates the posited importance of not deviating from the autoregressive setting, which aligns with the next-token prediction pre-training task.

We next investigate different design choices to further pinpoint the reasons for RAFT's performance.

Approach	**Gemma-2 2B**	**Gemma-2 9B**
Standard fine-tuning	0.83	0.82
- pre-training	doesn't learn	doesn't learn
+ bi-directional masking	doesn't learn	doesn't learn
Special-token logit (Fernandes et al., 2023)	0.51	0.46
- pre-training	0.94	0.94
+ bi-directional masking	0.50	0.46
+ 2-layer MLP	0.48	0.46
Pooled output embeddings (mean) (Zhuang et al., 2023)	0.50	0.47
+ bi-directional masking	0.49	0.48
Pooled output embeddings (min)	1.38	1.18
Pooled output embeddings (max)	1.32	1.13
RAFT	**0.47**	0.45
- pre-training	1.85	1.89
+ bi-directional masking	0.49	**0.44**
learnable-RAFT over all tokens	0.48	0.45
learnable-RAFT over digits `'1'-'5'`	**0.47**	0.45

Table 4: RMSE on Gemma-2 models on Amazon Wireless across predictive head and RAFT variants.

5.3 VARYING THE PREDICTIVE HEAD CHOICES IN THE UNIFIED VIEW

In Table 4, we report results from additional ablations considering different choices for the predictor $\hat{y}$, as well as other design choices pertaining to the model (i.e., the use of pre-training and attention masking). We next discuss the key findings.

Role of pre-training. In order to shed light on the importance of aligning the method with the pre-training task, we experiment with fine-tuning from randomly initialized model weights as opposed to initializing with the pre-trained checkpoint. We find that autoregressive fine-tuning does not converge to a reasonable result, while RAFT converges to a very poor predictor. Under this setting, predictive head fares the best among the 3 methods. To further analyze the role of pre-training, we run additional experiments on a synthetic regression dataset from Vacareanu et al. (2024a) (the Original #1 dataset) and report results in Table 13 (Appendix). We corroborate the finding that RAFT improves over predictive head when initialized from a pre-trained checkpoint, and not when the model weights are initialized randomly. This finding supports the hypothesis that RAFT outperforms predictive head due to better alignment with the pre-training checkpoint.

Predictive head non-linear variant. We experiment with two variants of learnable-RAFT: one where the weight vector is learnt for all vocabulary entries, and the second, where only entries corresponding to digits `'1'-'5'` are learnt, while other entries are fixed to 0. For both variants we found it necessary to initialize from the solution corresponding to RAFT, as random initialization did not lead to good training dynamics. Overall, we find both approaches to not improve over RAFT. We also consider adding a non-linear function over the special-token logit in the form of a two-layer MLP. This again does not lead to improvements over RAFT. Both results demonstrate that it may be more important to align the fine-tuning to the pre-training loss, as opposed to only try make the predictor more expressive.

Predictive head design choices. Lastly, we experiment with additional variants, including pooling over the full sequence of token embeddings from the LLM, instead of taking the final token activation. We find it to not lead to significantly better results than the special-token logit method. We also find bi-directional masking of attention does not significantly affect the results of the predictive head and RAFT, while expectedly making autoregressive fine-tuning untrainable (due to the model being able to attend to the predicted target during training). Overall, none of the predictive head

Dataset	Model size	'1'-'5'	'5'	'1','9'	'4','5'	'7','8','9'
Wireless	2B	**0.47±0.01**	0.48±0.01	0.48±0.00	0.48±0.00	0.48±0.01
	9B	**0.45±0.01**	**0.45±0.01**	0.47±0.02	0.46±0.00	0.47±0.01
Personal care	2B	**0.49±0.00**	**0.49±0.00**	**0.49±0.02**	**0.49±0.00**	**0.49±0.00**
	9B	**0.47±0.01**	0.48±0.01	0.48±0.02	**0.47±0.00**	**0.47±0.00**
Music	2B	**0.50±0.00**	**0.50±0.00**	**0.50±0.01**	**0.50±0.00**	**0.50±0.01**
	9B	**0.46±0.01**	0.47±0.00	0.48±0.02	0.47±0.00	0.48±0.00

Table 5: Comparison of RMSE (mean ± std dev) across variants of RAFT with varying sizes of the grid $\mathcal{Y}_{\text{grid}}$. The choice of '1'-'5' outperforms the alternatives.

variants improves over the RAFT approaches, again supporting the hypothesis of the importance of not deviating from the pre-training loss in fine-tuning.

5.4 RAFT ABLATIONS

Sensitivity to the grid size Following Lemma 3, $\hat{y}_{\text{MALI}}(x)$ can express any numerical target in the population limit, even with a coarse grid. Empirically, however, one might expect different grids to affect the results. We now assess this point on the Amazon reviews Wireless dataset, comparing the following choices for $\mathcal{Y}_{\text{grid}}$: (1) { '1', '2', '3', '4', '5' } (the default choice for RAFT), (2) { '5' } (viz. $\max(\mathcal{Y})$), (3) { '1','9' } (the smallest and largest digit in $\mathcal{V}$), (4) { '4', '5' } (the two largest digits in $\mathcal{Y}$ for all datasets), (5) { '7','8','9' } (the two largest digits in $\mathcal{V}$).

In Table 5, we report the results for different choices for $\mathcal{Y}_{\text{grid}}$. We find that, aligned with Lemma 3, limiting the grid does not significantly impact the results. However, when comparing the number of steps to convergence (see Table 12 in Appendix), we find that the default choice (1) in most cases tends to converge faster to the best solution than other choices. One explanation for this finding is the following. As shown by Lukasik et al. (2024), the choice of { '1', '2', '3', '4', '5' } yields a reasonable result for the zero-shot MALI approach, and thus it provides a good starting point for fine-tuning. Recall that the zero-shot MALI corresponds to the RAFT approach at step 0 of training, since the predictors for each are equivalent.

Sensitivity to the grid token indices The next question we pose is about the importance of strictly sticking to numeric tokens: what would happen if the RAFT predictor $\hat{y}_{\text{RAFT}}(x)$ used non-numeric tokens? To analyze this question, let us consider a more general form of the predictor:

$$\hat{y}_{\text{RAFT-NN}}(x) = \sum_{y \in \mathcal{Y}_{\text{grid}}} p(\texttt{token}(y) \mid x) \cdot y, \tag{11}$$

where $\texttt{token}(y) \in \mathcal{V}$ denotes a token of choice corresponding to the numerical target y.

We keep $\mathcal{Y}_{\text{grid}}$ as composed of the digits { '1', '2', '3', '4', '5' }, and use the predictor in Equation 11 with the following choices for tokens: (1) token for each digit becomes an alphabet token starting with 'a' and ending with 'a', (2) each token is a digit (i.e., '1' becomes '5', '2' becomes '4'), (3) we only consider digit '5'. As shown in Table 11 (Appendix), in most settings, the choice of digits { '1', '2', '3', '4', '5' } is as performant as any other choice. In certain settings, we even find a large drop in performance (e.g., the choice of characters or months). However, by enlarge, we find the results do not worsen signifanctly when different tokens are used.

6 DISCUSSION AND FUTURE WORK

We introduced regression-aware fine-tuning (RAFT), a new method for fine-tuning decoder-based LLMs to predict numeric targets. We demonstrated empirically that RAFT can consistently outperform existing methods that perform standard log-perplexity fine-tuning, as well as methods that construct separate predictive heads. An interesting direction for study would be applications of such techniques to problems like time-series forecasting, as well as problems of ordinal regression.

REFERENCES

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). Association for Computational Linguistics, 2017. doi: 10.18653/v1/s17-2001. URL http://dx.doi.org/10.18653/v1/S17-2001.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

David Dukić and Jan Snajder. Looking right is sometimes right: Investigating the capabilities of decoder-only LLMs for sequence labeling. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Findings of the Association for Computational Linguistics ACL 2024, pp. 14168–14181, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.843.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In Philipp Koehn, Barry Haddow, Tom Kocmi, and Christof Monz (eds.), Proceedings of the Eighth Conference on Machine Translation, pp. 1066–1083, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.wmt-1.100. URL https://aclanthology.org/2023.wmt-1.100.

Gemini Team et al. Gemini: A family of highly capable multimodal models, 2024.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In The 2023 Conference on Empirical Methods in Natural Language Processing, 2023. URL https://openreview.net/forum?id=KkHY1WGDII.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters, 2023.

Sameer Jain, Vaishakh Keshava, Swarnashree Mysore Sathyendra, Patrick Fernandes, Pengfei Liu, Graham Neubig, and Chunting Zhou. Multi-dimensional evaluation of text summarization with in-context learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), Findings of the Association for Computational Linguistics: ACL 2023, pp. 8487–8495, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.537. URL https://aclanthology.org/2023.findings-acl.537.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. arXiv preprint arXiv:1911.03437, 2019.

Janez Kaiser, Bogomir Horvat, and Zdravko Kacic. A novel loss function for the overall risk criterion based discriminative training of hmm models. In 6th International Conference on Spoken Language Processing (ICSLP 2000), pp. vol. 2, 887–890, 2000. doi: 10.21437/ICSLP.2000-412.

Tom Kocmi and Christian Federmann. Large language models are state-of-the-art evaluators of translation quality. In Mary Nurminen, Judith Brenner, Maarit Koponen, Sirkku Latomaa, Mikhail Mikhailov, Frederike Schierl, Tharindu Ranasinghe, Eva Vanmassenhove, Sergi Alvarez Vidal, Nora Aranberri, Mara Nunziatini, Carla Parra Escartín, Mikel Forcada, Maja Popovic, Carolina Scarton, and Helena Moniz (eds.), Proceedings of the 24th Annual Conference of the European Association for Machine Translation, pp. 193–203, Tampere, Finland, June 2023. European Association for Machine Translation. URL https://aclanthology.org/2023.eamt-1.19.

Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu lee Wang, Qing Li, and Xiaoqin Zhong. Label supervised llama finetuning, 2023. URL https://arxiv.org/abs/2310.01208.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), Advances in Neural Information Processing Systems, volume 35, pp. 1950–1965. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/0cde695b83bd186c1fd456302888454c-Paper-Conference.pdf.

Tiedong Liu and Bryan Kian Hsiang Low. Goat: Fine-tuned LLaMA outperforms GPT-4 on arithmetic tasks, 2023.

Michal Lukasik, Harikrishna Narasimhan, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Metric-aware llm inference for regression and scoring, 2024. URL https://arxiv.org/abs/2403.04182.

Sichun Luo, Yuxuan Yao, Bowei He, Yinya Huang, Aojun Zhou, Xinyi Zhang, Yuanzhang Xiao, Mingjie Zhan, and Linqi Song. Integrating large language models into recommendation via mutual augmentation and adaptive aggregation, 2024. URL https://arxiv.org/abs/2401.13870.

Ali Naseh, Kalpesh Krishna, Mohit Iyyer, and Amir Houmansadr. Stealing the decoding algorithms of language models. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS '23, pp. 1835–1849, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700507. doi: 10.1145/3576915.3616652. URL https://doi.org/10.1145/3576915.3616652.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL https://aclanthology.org/D19-1018.

Made Nindyatama Nityasya, Haryo Wibowo, Alham Fikri Aji, Genta Winata, Radityo Eko Prasojo, Phil Blunsom, and Adhiguna Kuncoro. On "scientific debt" in NLP: A case for more rigour in language model pre-training research. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8554–8572, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.477. URL https://aclanthology.org/2023.acl-long.477.

OpenAI et al. GPT-4 technical report, 2023.

Rohit Prabhavalkar, Tara N. Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. Minimum word error rate training for attention-based sequence-to-sequence models. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4839–4843, 2018. doi: 10.1109/ICASSP.2018.8461809.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting, 2023.

Muhammad Qorib, Geonsik Moon, and Hwee Tou Ng. Are decoder-only language models better than encoder-only language models in understanding word meaning? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Findings of the Association for Computational Linguistics ACL 2024, pp. 16339–16347, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.967.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Matt Shannon. Optimizing expected word error rate via sampling for speech recognition, 2017. URL https://arxiv.org/abs/1706.02776.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin

Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Robert Vacareanu, Vlad Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. In First Conference on Language Modeling, 2024a. URL https://openreview.net/forum?id=LzpaUxcNFK.

Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciu, and Mihai Surdeanu. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. arXiv preprint arXiv:2404.07544, 2024b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Proceedings of NeurIPS 2017, 2017.

Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. Gpt can solve mathematical problems without a calculator, 2023.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. Sentiment analysis in the era of large language models: A reality check. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), Findings of the Association for Computational Linguistics: NAACL 2024, pp. 3881–3906, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.246. URL https://aclanthology.org/2024.findings-naacl.246.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, pp. 2308–2313, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3592047. URL https://doi.org/10.1145/3539618.3592047.

A Related works

Generative models have been successfuly applied to number prediction, where a number is generated token by token in an autoregressive manner. For example, Gruver et al. (2023) considered it in a zero-shot learning setup for time series prediction, Vacareanu et al. (2024a) experimented with zero-shot regression problems, and Liu & Low (2023); Yang et al. (2023) considered the autoregressive finetuning over numerical targets applied to arithmetic tasks. The importance of tokenizing the numerical targets into individual digits has been raised by previous works (Liu & Low, 2023; Yang et al., 2023).

Encoder-based models (e.g., BERT) relying on the masked language modeling pretraining tasks have been primarily employed to discriminative tasks (including classification and regression) (Devlin et al., 2019). Decoder-based large language models (LLMs) (e.g., GPT, LLaMa) on the other hand, mostly relying on the next-token prediction pretraining task, showed state of the art results across a range of generative tasks. (OpenAI et al., 2023; Anil et al., 2023; Touvron et al., 2023; Gemini Team et al., 2024)

While there is an on-going research regarding whether the encoder or decoder architecture is better tailored to predictive tasks (Nityasya et al., 2023; Li et al., 2023; Dukić & Snajder, 2024; Qorib et al., 2024), in this work we focus on the question of how do we best apply decoder models to predictive tasks?

B ADDITONAL THEORETICAL RESULTS

B.1 PROOF FOR LEMMA 1

Proof. Pick any $\epsilon > 0$. Recall that $N \doteq \max(\mathcal{Y})$. Consider a distribution $\mathbb{P}$ such that, for a given example $x \in \mathcal{X}$, $\mathbb{P}(0|x) = \frac{1+0.5\epsilon}{2}$, $\mathbb{P}(N|x) = \frac{1-0.5\epsilon}{2}$, and all other targets attain probability 0. Next, consider the model distribution: $p(0|x) = \frac{1-0.5\epsilon}{2}$, $p(N|x) = \frac{1+0.5\epsilon}{2}$, and all other targets attain probability 0. Then, we get: $\|\mathbb{P}(\cdot|x) - p(\cdot|x)\|_1 = \epsilon$, and $(\hat{y}_{\text{mode}}(x) - \mathbb{E}_{y^* \sim \mathbb{P}}[y^*])^2 = \left(\frac{N}{2}\right)^2 \left(1 + \frac{\epsilon}{2}\right)^2$. □

B.2 PROOF FOR LEMMA 2

Proof. Pick any $\epsilon > 0$. Recall that $N \doteq \max(\mathcal{Y})$. Consider a distribution $\mathbb{P}$ such that, for a given example $x \in \mathcal{X}$, $\mathbb{P}(0|x) = \frac{1+0.5\epsilon}{2}$, $\mathbb{P}(N|x) = \frac{1-0.5\epsilon}{2}$, and all other targets attain probability 0. Next, consider the model distribution: $p(0|x) = \frac{1-0.5\epsilon}{2}$, $p(N|x) = \frac{1+0.5\epsilon}{2}$, and all other targets attain probability 0. Then, we get: $\|\mathbb{P}(\cdot|x) - p(\cdot|x)\|_1 = \epsilon$, and $(\mathbb{E}_{y \sim p}[y] - \mathbb{E}_{y^* \sim \mathbb{P}}[y^*])^2 = \left(\frac{\epsilon N}{2}\right)^2$. □

B.3 PROOF FOR LEMMA 3

Proof. For simplicity, we avoid explicitly stating conversions from float to string, and vice versa. For any x, we wish to minimize:

$$\mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)} \left[\left(\sum_{y \in \mathcal{Y}_{\text{grid}}} p(y|x) \cdot y - y^* \right)^2 \right].$$

Equating the derivative w.r.t. $p(y|x)$ to 0, we derive the first-order condition for optimality:

$$2 \cdot \mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)} \left[\left(\sum_{y' \in \mathcal{Y}_{\text{grid}}} p(y'|x) \cdot y' - y^* \right) \right] \cdot y = 0, \forall y \in \mathcal{Y}_{\text{grid}}.$$

So the optimal solution is achieved when:

$$\sum_{y' \in \mathcal{Y}_{\text{grid}}} p(y'|x) \cdot y' = \mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)}[y^*].$$

Under the conditions in the lemma (the smallest and largest numbers in $\mathcal{Y}$ are present in $\mathcal{Y}_{\text{grid}}$), there exists a probability distribution $p^*(y|x)$ such that $\hat{y}_{\text{MALI}}(x; \mathcal{Y}_{\text{grid}}) = \sum_{y \in \mathcal{Y}_{\text{grid}}} p^*(y \mid x) \cdot y = \mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)}[y^*]$, thus satisfying the condition for optimality. □

B.4 GENERAL VERSION OF LEMMA 5

Lemma 4. The minimizer of the following objective:

$$\mathbb{E}_{y^* \sim p^*(\cdot|x)} \mathbb{E}_{y \sim p(\cdot|x)} \left[\ell(y^*, y) \right],$$

for a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, is a one-hot distribution over targets such that all probability mass is on a target $\hat{y} \in Y$ which minimizes $\mathbb{E}_{y^* \sim p^*(\cdot|x)} \left[\ell(y^*, \hat{y}) \right]$.

Dataset	**Input prompt**	**Target range**
STSB	What is the sentence similarity between the following two sentences measured on a scale of 0 to 5: {Sentence #1}, {Sentence #2}. The similarity measured on a scale of 0 to 5 with 0 being unrelated and 5 being related is equal to	[0, 5]
Amazon reviews	What is the rating corresponding to the following review in the scale of 1 to 5, where 1 means negative, and 5 means positive? Only give a number from 1 to 5 with no text. Review: {Review}. Rating:	1, 2, 3, 4, 5
MovieLens-1M	Instruction: Predict the rating of a target movie based on the user's historical movie ratings. Rating History: {Rating history} Candidate Item: {Candidate Item}. Output:	1, 2, 3, 4, 5
Synthetic (Original #1 from (Vacareanu et al., 2024a))	The task is to provide your best estimate for 'output score' based on 'input score'. Please provide that and only that, without any additional text. Input score: {Input score}. Output score:	[0, 9]

Table 6: Prompts used for different datasets and the corresponding target ranges. Curly braces denote inputs specific to an input example. For Synthetic (Original #1 from (Vacareanu et al., 2024a)) we normalize the targets to correspond to [0, 9].

Dataset	**Train size**	**Validation size**	**test size**
Wireless	10'000	1'500	1'500
Personal care	10'000	1'500	1'500
Music	10'000	1'500	1'500
STSB	4'887	863	1'500
STSB 1k	1'000	863	1'500
MovieLens-1M	797'758	10'145	10'145
Synthetic (Original #1 from (Vacareanu et al., 2024a))	10'000	1'000	1'000

Table 7: Summary of dataset statistics.

Proof. The proof is elementary. Expanding the above objective:

$$\begin{aligned}
\mathbb{E}_{y^*\sim p^*(\cdot|x)}\mathbb{E}_{y\sim p(\cdot|x)}\left[\ell(y^*,y)\right] &= \mathbb{E}_{y\sim p(\cdot|x)}\mathbb{E}_{y^*\sim p^*(\cdot|x)}\left[\ell(y^*,y)\right] \\
&= \int_{y\in\mathcal{Y}} \mathbb{E}_{y^*\sim p^*(\cdot|x)}\left[\ell(y^*,y)\right]\cdot p(y|x)\cdot dy \\
&\geq \int_{y\in\mathcal{Y}} \mathbb{E}_{y^*\sim p^*(\cdot|x)}\left[\ell(y^*,\hat{y})\right]\cdot p(y|x)\cdot dy \\
&= \mathbb{E}_{y^*\sim p^*(\cdot|x)}\left[\ell(y^*,\hat{y})\right] \\
&= \int_{y\in\mathcal{Y}} \mathbb{E}_{y^*\sim p^*(\cdot|x)}\left[\ell(y^*,y)\right]\cdot p(y|x)\cdot dy,
\end{aligned}$$

where the third step follows from the fact that $\hat{y}$ minimizes $\mathbb{E}_{y^*\sim p^*(\cdot|x)}\left[\ell(y^*,\cdot)\right]$; on the final step, $p(\cdot|x)$ is a probability distribution that has a point mass on $\hat{y}$. □

C Additional details

In Table 6 we report the prompts we used in our experiments, and in Table 7 we report the dataset statistics.

We update all parameters during the fine-tuning. We summarize specific settings below. For Gemma-2, we use the following settings.

- We use dropout rate 0.1 and batch size 16.
- We train for 200k steps and select the best step using the held out validation set (see Table 7 for details on the train/test/validation splits).
- We use a constant learning rate schedule. We select the learning rate value over the validation set from the values: 1e-4, 1e-5, 1e-6.

Method/ablation	RMSE
sampled regression aware (Definition (2))	0.98
regression aware (Definition (1))	**0.40**

Table 8: Root mean squared error (RMSE) on STSB across regression aware approaches and their variants on Gemma-2 9B.

- We use the Adafactor optimizer to save memory during the fine-tuning (we find Adam to not perform better). The parameters for Adafactor are: ϵ_1 =1e-30, ϵ_2 =1e-3, decay rate = 0.8.

For PaLM-2, we use the above settings, except we use batch size 64 and dropout rate 0.0, and train for 5k steps and report the results from the last checkpoint.

D Comparison to MBR-based fine-tuning

The RAFT loss may be contrast against ideas in the literature on Minimum Bayes Risk (MBR) prediction literature (Kaiser et al., 2000; Shannon, 2017; Prabhavalkar et al., 2018), which optimize non-regression metrics via approximation using sampled model predictions. For the squared loss, this may be formulated as follows:

Definition 2. Define the sampled regression-aware loss as follows:

$$\ell_{\mathrm{MBR}}(y^*, p(\cdot \mid x)) = \mathbb{E}_{y \sim p(\cdot|x)} \left[(y^* - \texttt{float}(y))^2 \right]. \tag{12}$$

Compared to the loss in Equation 9, the key difference is that the expectation over the model outputs appears outside the square loss. A naïve empirical implementation of this objective requires explicitly sample responses from the model $p(\cdot \mid x)$; this can be expensive and incur high variance. As with $\hat{y}_{\mathrm{MALI}}(x; \mathcal{Y}_{\mathrm{grid}})$, one may instead consider a practical variant that approximates the expectation using a restricted grid of targets $\mathcal{Y}_{\mathrm{grid}} \subset \mathcal{Y}$:

$$\hat{L}_{\mathrm{MBR}}(p; \mathcal{Y}_{\mathrm{grid}}) = \frac{1}{N} \sum_{(x,y^*) \in S} \sum_{y \in \mathcal{Y}_{\mathrm{grid}}} p(\texttt{str}(y) \mid x) \cdot (y^* - y)^2 \tag{13}$$

Even this variant has a notable disadvantage: the minimizer of Equation 13 is a one-hot distribution that places all its probability mass on one of the targets in $\mathcal{Y}_{\mathrm{grid}} \subset \mathcal{Y}$:

Lemma 5. Let $y^*(x) = \mathbb{E}_{y^* \sim \mathbb{P}(\cdot|x)}[y^*]$ denote the Bayes-optimal prediction for input x. We assume $\mathbb{P}(\cdot \mid x)$ is supported on numerical targets only. The minimizer of the approximate sampled regression-aware loss in Equation 13 over all model distributions $p(\cdot \mid x)$ is of the form:

$$p(y \mid x) = \begin{cases} 1 & \text{if } \; y = \arg\min_{y' \in \mathcal{Y}_{\mathrm{grid}}} \|y' - y^*(x)\|_2 \\ 0 & \text{else} \end{cases}.$$

Therefore, the quality of the minimizer $p(\cdot \mid x)$ entirely depends on how well $\mathcal{Y}_{\mathrm{grid}}$ approximates the original target space $\mathcal{Y}$. For example, if $\mathcal{Y}_{\mathrm{grid}}$ is a set of integers, the minimizer of Equation 13 will also be limited to predicting integers, even when the original target space $\mathcal{Y}$ contains floating-point numbers of arbitrary precision. As shown in Lemma 3, RAFT does not suffer from the loss of precision resulting from the use an approximate target space, and also avoids the high variance associated with sampling. In Table 8 (Appendix), we experimentally verify better performance of RAFT over the MBR-based fine-tuning, and unless otherwise stated, we focus our attention to RAFT.

dataset	zero-shot	MALI	autoregressive	autoregressive+MALI	predictive head	RAFT
Wireless	0.87	1.40	0.83	0.61	0.50	**0.44**
Personal care	0.94	1.72	0.89	0.61	0.50	**0.47**
Music	1.26	2.45	1.23	0.64	**0.47**	0.49
STSB 1k	1.29	1.03	0.60	0.64	0.56	**0.55**
STSB	1.29	1.03	0.62	0.58	0.51	**0.48**

Table 9: Comparison of RMSE across datasets for Gemma-2 27B. In most cases, RAFT outperforms all other methods.

D.1 Proof for Lemma 5

Proof. Notice that:

$$\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[\mathbb{E}_{y\sim p(\cdot|x)}\left[(y-y^*)^2\right]\right] = \mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[\sum_y p(y|x)\cdot(y-y^*)^2\right]$$
$$= \sum_y p(y|x)\cdot\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[(y-y^*)^2\right].$$

Since this is a convex combination, the optimal value is achieved for p to be a one-hot vector with a 1 on the index $\arg\min_y \mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[(y-y^*)^2\right]$. We thus want a y that minimizes:

$$\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[(y-y^*)^2\right] = \mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[y^2+(y^*)^2-2\cdot y\cdot y^*\right].$$

Equivalently, we want a y that minimizes:

$$y^2 - 2\cdot y\cdot\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[y^*\right].$$

or equivalently:

$$y^2 - 2\cdot y\cdot\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[y^*\right] + \left(\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[y^*\right]\right)^2 = \left(y-\mathbb{E}_{y^*\sim\mathbb{P}(\cdot|x)}\left[y^*\right]\right)^2.$$

□

E Additional Experimental Results

In this section we provide additional experimental results corroborating the findings in the main paper.

E.1 Additional Models: Gemma-2 27B and the PALM-2 Model Family

We report results on Gemma-2 27B across all dataset in Table 9 and on PALM-2 models on STSB in Table 10, corroborating the findings of RAFT improving in most settings.

E.2 Choices for Tokens in RAFT

In Table 11 we report results for different choices of tokens in RAFT, and find the default choice to work best.

E.3 Convergence Speed of RAFT

In Table 12 we report the number of training steps to convergence to the best result on the held out validation set of different methods. We can see that RAFT with digits `'1'-'5'` it majority of cases converges faster than other choices for the grid.

In Figure 1, we compare the number of steps to convergence of RAFT and predictive head on STSB. The figure shows that RAFT converges with fewer number of steps across different percentages of training data used for training.

model family	model size	autoregressive	predictive head	RAFT
PALM-2	1B	0.79 ± 0.02	**0.61** ± 0.01	0.62 ± 0.01
PALM-2	24B	0.63 ± 0.03	0.56 ± 0.00	**0.53** ± 0.00

Table 10: Comparison of root mean squared error (RMSE) on STSB across different ULM model sizes, and across different fine-tuning methods: autoregressive, predictive head and autoregressive regression aware. Each model is ran for 3 times to obtain standard deviations. For ULM 1B, we find no difference in the performance of predictive head and RAFT, and both outperform the autoregressive approach. For ULM 24B, we see a significant improvement from RAFT over both the autoregressive and the predictive head approaches.

dataset	model size	digits `'1'–'5'`	characters `'a'-'e'`	reversed digits	`''January'-'May''`
Wireless	2B	0.47 ± 0.01	0.48 ± 0.01	0.48 ± 0.00	0.83 ± 0.01
	9B	0.45 ± 0.00	0.46 ± 0.00	0.46 ± 0.01	0.47 ± 0.01
Personal care	2B	0.49 ± 0.00	0.48 ± 0.00	0.48 ± 0.01	0.85 ± 0.00
	9B	0.47 ± 0.01	0.47 ± 0.00	0.48 ± 0.01	0.48 ± 0.01
Music	2B	0.50 ± 0.00	0.50 ± 0.01	0.50 ± 0.01	0.50 ± 0.00
	9B	0.46 ± 0.00	0.48 ± 0.01	0.47 ± 0.01	0.61 ± 0.25

Table 11: Comparison of RMSE (mean ± std dev) across variants of RAFT where different tokens are used for the prediction formula of RAFT. Each experiment repeated for 3 times. In most settings, the choice of digits `'1'–'5'` is at least as performant as any other choice. In certain settings, we find a large drop in performance compared to the choice of digits (i.e., months `'January'-'May'`.

dataset	model size	`'1'-'5'`	`'5'`	`'1','9'`	`'4','5'`	`'7','8','9'`
Wireless	2B	**1000**	3000	2000	1600	2200
	9B	**1600**	2800	3400	3200	4200
Personal care	2B	2200	2200	2400	**1600**	2200
	9B	4200	2000	6200	**400**	2400
Music	2B	**800**	1200	1800	1800	1800
	9B	**1000**	2200	3000	1600	3000

Table 12: Comparison of the number of steps to convergence of RAFT where different number of numerical targets are used for the grid in RAFT. Results on the Amazon Wireless dataset.

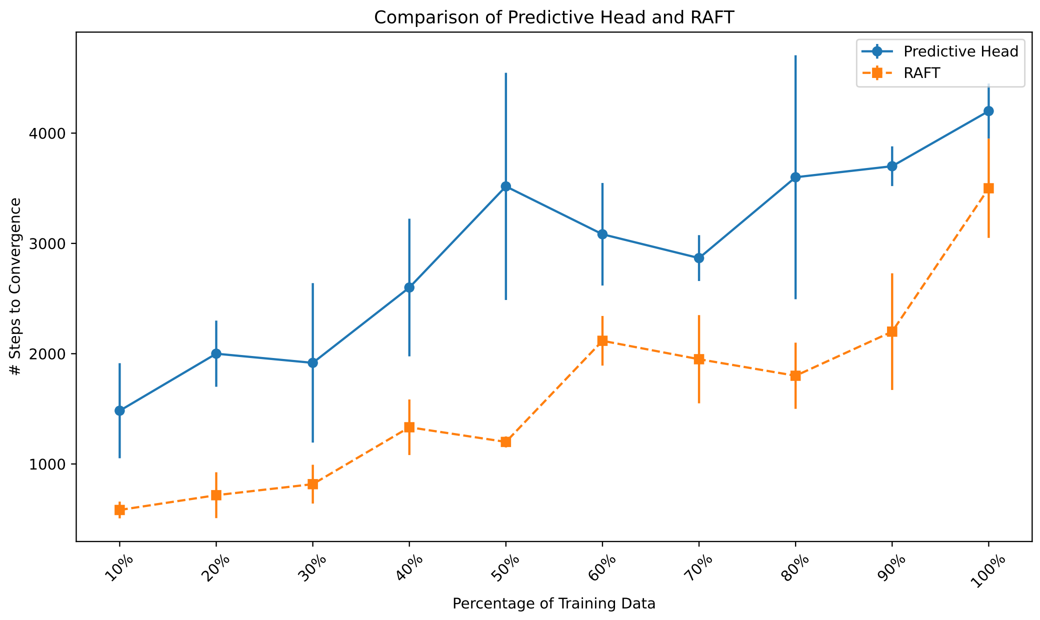


Figure 1: Comparison of the number of steps to convergence of RAFT and predictive head, with different percentage of the training set.

initialiation	model size	autoregressive	predictive head	RAFT
Pre-trained checkpoint	2B	0.018	0.013	**0.005**
	9B	0.017	0.017	**0.006**
Random	2B	2.536	**0.327**	1.704
	9B	2.536	**0.147**	1.092

Table 13: The role of initialization to the pre-trained checkpoint on a synthetic regression dataset from Vacareanu et al. (2024a) (the Original #1 dataset). We compare RMSE across different Gemma model sizes, and across different fine-tuning methods: autoregressive, predictive head and autoregressive regression aware. We corroborate our observation from language regression tasks that RAFT improves over the predictive head approach when initialized from a pre-trained checkpoint, and not when model weights are initialized randomly.

E.4 Additional results on Synthetic data

For a simple setting, we consider a synthetic regression dataset from Vacareanu et al. (2024a) referred to as the Original #1 dataset by the authors. We report results in Table 13 and corroborate our observation from the language regression task experiment that RAFT improves over the predictive head approach when initialized from a pre-trained checkpoint, and not when the model weights are initialized randomly (see Table 4). This provides additional support for our hypothesis that the alignment of RAFT to the next-token prediction pre-training task is the underlying reason for its better performance over the predictive head.

E.5 Additional results on MovieLens-1M

We ran additional experiments on the movie recommendation problem on the MovieLens-1M dataset with Gemma-2 2B model. We use AdamW optimizer and sweep learning rates from the range {1e-4, 1e-5, 1e-6}. We use a cosine decay schedule for the learning rate, 10k steps of warmup from 1e-8 learning rate. We report results in Table 14 and again find RAFT to improve over both the predictive head and autoregressive methods.

Method	RMSE
Autoregressive	0.95
Predictive head	0.91
RAFT	**0.89**

Table 14: Root mean squared error (RMSE) on MovieLens-1M across approaches on Gemma-2 2B.

Method/ablation	Parameter count	RMSE	Pearson corr.	Spearman corr.
RoBERTa Base CLS	110M	0.64	90.84	90.59
RoBERTa Large CLS	356M	0.59	91.99	92.02
RoBERTa Large mean-pooling	356M	0.63	91.65	91.56
RoBERTa Large mean-pooling freeze	356M	1.08	72.72	74.16
RoBERTa Large CLS freeze	356M	1.30	56.48	54.76
SMART BERT (Jiang et al., 2019)	356M	-	90.00	89.40
SMART RoBERTa (Jiang et al., 2019)	356M	-	92.80	92.60
Gemma-2 2B RAFT	2B	0.54	93.55	93.22
Gemma-2 9B RAFT	9B	**0.51**	**94.30**	**94.18**

Table 15: Root mean squared error (RMSE) on STSB across baselines. Results from SMART (Jiang et al., 2019) taken as reported in the paper (RMSE was not reported).

E.6 Comparison to encoder-based baselines

In Table 15, we report results with additional baselines that use a prediction head over RoBERTa representation for the input sequence. We include: mean-pooling and CLS token variants, and both frozen RoBERTa weights and unfrozen weights in the fine-tuning. We also include the SMART method (Jiang et al., 2019). In keeping with previous work, in addition to RMSE, we also report performance on the Pearson and Spearman metrics for STSB (Jiang et al., 2019) (where available). We find RAFT to surpass all the included baselines.

E.7 Further ablating the RAFT loss: different choices for loss, normalization and token initialization

In Table 16 we compare the MSE loss to distillation style log loss on STSB. The target is scaled to be between 0 and 1 for this set of experiments. The log loss is defined as $-y^* \log p_1 - (1-y^*) \log(1-p_1)$ where p_1 is the probability of digit '1'. We find that both the MSE loss and the log loss yield similar results, and that scaling the range of the targets does not negatively affect the performance.

We next analyze whether enforcing the probabilities over the grid to sum to 1 (by normalizing them by the sum of the probabilities of all numbers in $\mathcal{Y}_{\text{grid}}$) can improve the performance of RAFT. Table 17 shows that applying normalization to the probabilities for numbers in $\mathcal{Y}_{\text{grid}}$ does not significantly affect the results.

In Table 18, we compare three initialization methods for the embedding of numbers in autoregressive and RAFT methods. The tokens used are of granularity of 0.1 for both autoregressive and RAFT methods, except for RAFT '1'-'5', which use digits from 1 to 5 to construct the grid (granularity 1.0). Overall, we find that RAFT is less sensitive to the initialization method than the autoregressive approach.

In Table 19, we experiment with random initialization for the tokens in the RAFT grid. We find random initialization to lead to worse results compared to using pre-trained token embeddings.

We next analyze the impact of the choice of $\mathcal{Y}_{\text{grid}}$ in computing the MALI predictor in Equation 5. To this end, we vary the granularity of $\mathcal{Y}_{\text{grid}}$ by constructing a list of equally spaced numbers covering

		unscaled targets		targets scaled to $[0, 1]$	
model family	**model size**	**predictive head**	**RAFT**	**RAFT '0' and '1'**	**log loss**
PALM-2	1B	0.61 ± 0.01	0.62 ± 0.01	0.63 ± 0.01	0.63 ± 0.01
PALM-2	24B	0.56 ± 0.00	0.53 ± 0.00	0.54 ± 0.01	0.53 ± 0.00

Table 16: Comparision of RMSE (mean $\pm$ std dev) on the STSB dataset for MSE loss and log loss when the target is scaled to be between 0 and 1. The MSE loss uses digit '0' and '1' to compute the predicted value. The log loss is in the form $-y^* \log p_1 - (1 - y^*) \log(1 - p_1)$ where p_1 is the probability of digit '1'.

model family	**model size**	**not normalized**	**normalized**
PALM-2	1B	0.63 ± 0.00	0.63 ± 0.02
PALM-2	24B	0.53 ± 0.00	0.53 ± 0.01

Table 17: The effect of normalization of grid token probabilities in RAFT on the STSB dataset.

the range of $\mathcal{Y}$. For example, choosing granularity to be 0.1 when $\mathcal{Y} = [0, 5]$ yields $\mathcal{Y}_{\text{grid}}$ = {`'0.0'`, `'0.1'`, `'0.2'`, ..., `'4.9'`, `'5.0'` }. In our implementation, all numbers in $\mathcal{Y}_{\text{grid}}$ are represented by single tokens that we add to the vocabulary. We initialize the token embedding with either the First or the Average method. In the First method, we initialize the embedding with the embedding of the first digit of the number (e.g. use token '0' embedding for the number '0.1'). In the Average method, we initialize the embedding with the average of the embedding from the constituent tokens (e.g. use the average embeding of token '0', '.' and '1' for the number '0.1'.) We report the results on the STSB datasets with PALM-2 1B model in Table 20 and find no significant difference in the results across different choices for the granularity of $\mathcal{Y}_{\text{grid}}$.

Additionally, in Table 20 we also include the autoregressive method utilizing additional tokens from $\mathcal{Y}_{\text{grid}}$ as constructed with the methodology outlined above (i.e., with varying granularity). Here, contrary to RAFT, we find the initialization method to affect the results, with First performing better than Average. Note that the autoregressive method is equivalent to the generative classification from (Fernandes et al., 2023), where the classes correspond to the numbers from the grid.

Lastly, we would like to note that in the case of generative classification, there is a trade-off between how fine-grained the grid is and how many examples per token are observed during training. In particular, if the classes are too coarse, we observe a loss in performance. On the other hand, if the classes are too fine-grained, there may be insufficient training examples per label to learn the embeddings for new tokens. For example, with granularity 0.05, 17 out of the 101 numbers in the grid do not appear in the training data. This can also lead to a loss in performance.

E.8 Distribution over tokens in the MALI Predictor

We next investigate the distribution over tokens in the MALI predictor. We find that, while the error decreases with training, the entropy increases after RAFT training, as we show in Figure 2. This corresponds to the model on average spreading the probabilities over tokens more than prior to fine-tuning, as shown for specific examples in Figure 3. We posit this to be beneficial, and indeed, RAFT fine-tuning does not restrict uncertainty of the model, contrary to the MBR fine-tuning (compare Lemma 3 and Lemma 5).

initialization	autoregressive	RAFT	RAFT '1'-'5'
Zero	1.20 ± 0.01	0.62 ± 0.00	0.63 ± 0.00
First	0.80 ± 0.02	0.63 ± 0.00	0.62 ± 0.02
Average	0.98 ± 0.01	0.64 ± 0.01	0.61 ± 0.01

Table 18: Comparison of different initialization methods for embeddings for tokens corresponding to numbers in autoregressive and RAFT methods on STSB dataset with PALM-2 1B model. We consider granularity 0.1 for constructing the tokens for autoregressive and RAFT reported in the first 2 columns, whereas the final column reports RAFT with tokens 1 to 5. Zero denotes initialization with 0 values, First denotes the initialization with the embedding corresponding to the first token of the number (e.g. use token '0' embedding for the number '0.1'), and Average denotes the initialization with the embedding corresponding to the different tokens in the number (e.g. average embeddings for tokens '0', '.' and '1' when initializing the embedding for the number '0.1'). RAFT is less sensitive to the initialization method. We find the performance does not significantly worsen with different initialization methods compared to using the pre-trained token embeddings at initialization (see Table 16).

	RAFT with '1'-'5'		RAFT with '5'	
	baseline	random	baseline	random
PaLM-2 1B	0.63 ± 0.00	0.66 ± 0.02	0.62 ± 0.01	0.64 ± 0.01

Table 19: RMSE across different initialization (baseline and random) of the tokens in RAFT. We find random initialization of tokens in the RAFT grid to hurt the performance.

	autoregressive		RAFT '1'-'5'	
granularity	First	Average	First	Average
0.05	0.85 ± 0.01	1.11 ± 0.01	0.61 ± 0.01	0.63 ± 0.00
0.1	0.80 ± 0.02	0.98 ± 0.01	0.63 ± 0.00	0.64 ± 0.01
0.2	0.83 ± 0.02	0.89 ± 0.01	0.64 ± 0.01	0.66 ± 0.02
0.5	0.84 ± 0.01	0.87 ± 0.01	0.63 ± 0.01	0.63 ± 0.02
1.0	0.81 ± 0.01	0.85 ± 0.01	0.62 ± 0.02	0.61 ± 0.01

Table 20: Comparison of RMSE on autoregressive and RAFT methods with different granularity of tokens. We append tokens that represent numbers with various granularity. For example, with granularity 0.05, the tokens are '0.00, '0.05, '0.10', . . . , '5.00'. For autoregressive, this is equivalent to generative classification method from (Fernandes et al., 2023). First and Average denotes different ways to initialize the embeddings of the tokens, with details described in the caption of Table 18.

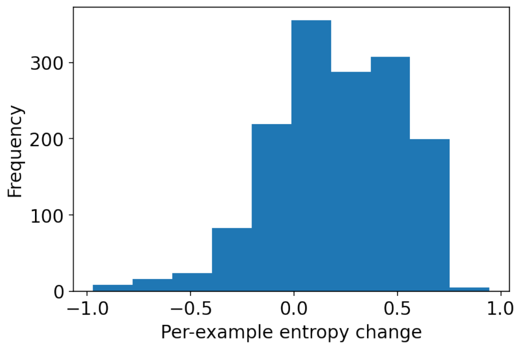
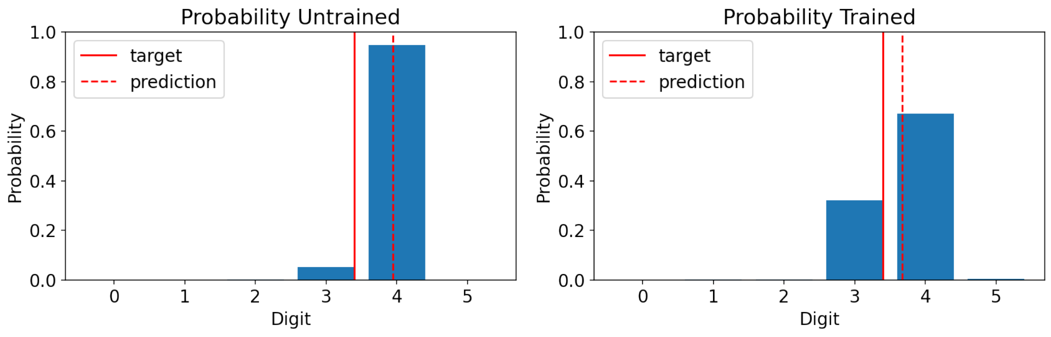
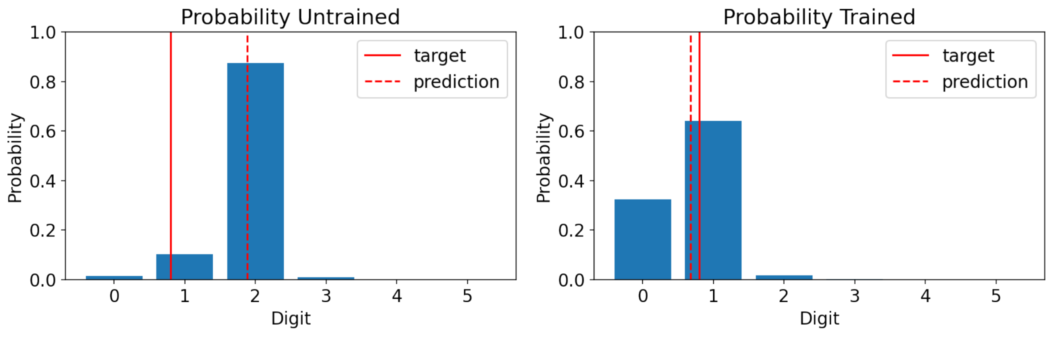


Figure 2: The histogram of per-example change (after vs. before fine-tuning) in entropy over the probabilities for the digits in the RAFT predictor. The entropy on average increases after the RAFT fine-tuning, meaning that the probabilities are overall more spread over the digits than prior to RAFT fine-tuning.



(a) Example #1



(b) Example #2

Figure 3: Comparison of the distribution over digit tokens in RAFT predictor (Gemma-2 2B on STSB) before and after fine-tuning. We find the entropy over the digit token probabilities overall increases after fine-tuning with the RAFT objective.