ON THE PREDICTIVE POWER OF REPRESENTATION DISPERSION IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We show that a language model's ability to predict text is tightly linked to the *breadth* of its embedding space: models that spread their contextual representations more widely tend to achieve lower perplexity. Concretely, we find that *representation dispersion*—the average pairwise cosine distance among hidden vectors—strongly and negatively correlates with perplexity across diverse model families (LLaMA, Qwen, and others) and domains (Wikipedia, news, scientific abstracts). Beyond illustrating this link, we show how dispersion can be leveraged for a range of practical tasks—without requiring labeled data. First, measuring dispersion on unlabeled text allows us to *predict* downstream accuracy in new domains, offering a data-efficient tool for *model selection*. Next, we find that identifying layers with higher dispersion pinpoints the best representations for retrieval-based methods such as kNN-LM, bypassing exhaustive layer-by-layer searches. Finally, we integrate a simple "push-away" objective into training, which increases dispersion in both single-domain and cross-domain scenarios and directly improves perplexity in each.

1 INTRODUCTION

Large language models can perform remarkably well on tasks ranging from text completion to code generation. Yet their *embedding geometry* often exhibits signs of anisotropy or rank collapse, whereby hidden states lie in a narrow cone or occupy a low-dimensional subspace (Ethayarajh, 2019; Gao et al., 2019; Li et al., 2020). Although this geometry has been posited to limit expressive power, *how* precisely it connects to auto-regressive text generation remains less clear.

In this paper, we present empirical evidence that a model's ability to predict text is tightly linked to the *breadth* of its embedding space. Intuitively, as illustrated in Figure 1, weaker models compress contexts into tight clusters, whereas stronger models separate these contexts —*even semantically similar ones*—more broadly. This broader geometry yields clearer distinctions in the latent space, enabling sharper (lower-entropy) next-token predictions. Concretely, we quantify **representation dispersion** at *any chosen layer* as the average pairwise cosine distance of its hidden vectors. Unless specified otherwise, our empirical sections use the final layer, and we show that higher dispersion consistently predicts *lower perplexity* (Figure 2).

Beyond revealing this fundamental link, representation dispersion offers *practical* benefits:

- **Label-free diagnostics.** Dispersion measured on *unlabeled* text predicts downstream accuracy in new domains (§3.1).
- **Model selection.** Among multiple pretrained or fine-tuned variants, the model with larger dispersion reliably performs better on domain-specific tasks such as code generation and math reasoning (§3.2).
- **Layer selection for retrieval augmentation.** Although the first two applications exploit the final hidden state, dispersion is equally informative inside the network. In kNN-LM, choosing the hidden layer with the highest dispersion yields the best perplexity, providing an unsupervised shortcut to sub-layer selection (§3.3).
- **A training signal.** Encouraging higher dispersion through an auxiliary "push-away" loss directly improves perplexity for both single-domain and cross-domain scenarios (§3.4).

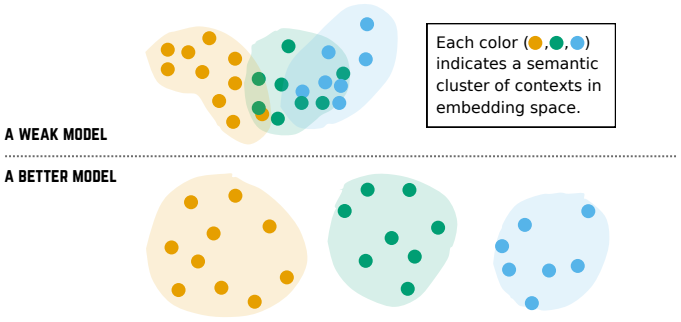

Figure 1: **Representation geometry in a weak model vs. a better model.** In a weak model (top), final-layer embeddings for similar contexts are compressed into tight clusters, limiting discriminative power. In a better model (bottom), embeddings are widely dispersed—even within semantically related clusters—leading to more confident next-token predictions.

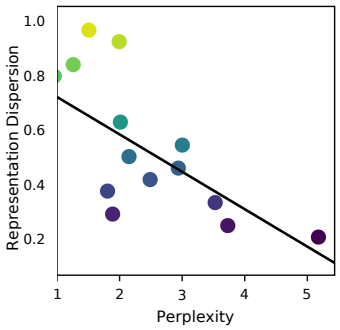

Figure 2: **An Illustrative Empirical Trend.** Across models/datasets, higher dispersion correlates with lower perplexity.

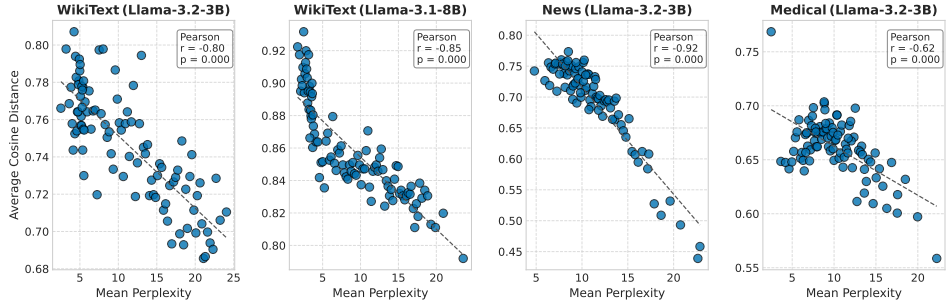

Figure 3: Sequence-level perplexity vs. embedding dispersion across different domains and model sizes. Each point represents a bin of text segments with the x-axis showing mean sequence-level perplexity and the y-axis showing average pairwise cosine distance of final-layer embeddings.

Overall, we show that a model's embedding geometry—captured by the simple statistic of *average pairwise distance*—serves as a robust indicator of predictive quality. By quantifying and encouraging this broader geometry, we gain both conceptual insight and practical benefits, and we hope this perspective fosters new avenues for improving model robustness and interpretability.

2 Empirical Analysis of Representation Geometry

We begin by describing how we measure representation geometry and perplexity (§2.1). We then present three key global observations that characterize how embedding dispersion evolves with perplexity, across layers, and under fine-tuning (§2.2). Finally, we zoom in on semantic clusters (§2.3) to examine how dispersion behaves among closely related contexts.

2.1 Measurement Setup

Contextual Representations. Following standard autoregressive conventions, let a language model with parameters θ assign probability to a token sequence $(x_1, x_2, \ldots, x_N)$ via $p_\theta(x_1, x_2, \ldots, x_N) = \prod_{n=1}^{N} p_\theta(x_n \mid x_{<n})$, where $x_{<n} = (x_1, \ldots, x_{n-1})$. In contemporary Transformer-based models, each partial sequence $x_{<n}$ is mapped to an internal *context vector* $\mathbf{h}_n \in \mathbb{R}^d$ by a function $f_\theta(\cdot)$. The next-token distribution is then given by $p_\theta(x_n \mid x_{<n}) = \mathrm{softmax}(W_o\, \mathbf{h}_n)$, where $W_o \in \mathbb{R}^{|V| \times d}$ projects the representation $\mathbf{h}_n$ into the vocabulary space V. For measuring *representation dispersion*, we focus on a chosen final-layer vector (e.g., $\mathbf{h}_N$ for the full sequence), by default, as the representation of each text sample.

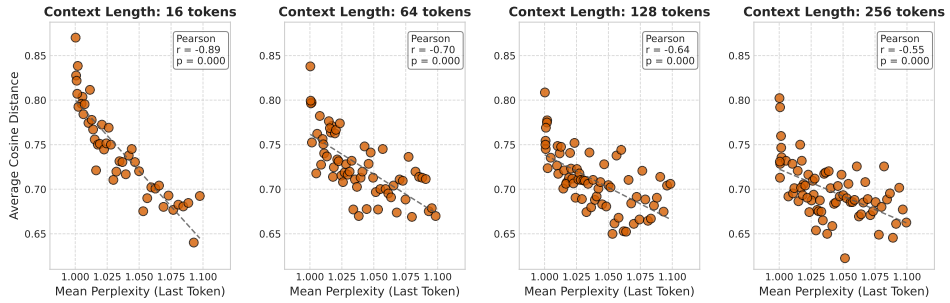

Figure 4: Last-token perplexity vs. embedding dispersion at varying context lengths using LLaMA-3.2-1B. Each point represents a bin of text segments with the x-axis showing mean last-token perplexity and the y-axis showing average pairwise distance.

Measuring Representation Dispersion. To measure how well the model separates text samples in its embedding space, we first choose a particular layer or sub-layer whose representation we wish to examine (e.g., the final hidden state, the output after the final attention block, or the second-to-last block). Concretely, we sample N text segments from a dataset and pass each segment through the model to extract the corresponding representation $\mathbf{E}_i \in \mathbb{R}^d$ from the chosen sublayer, for $i = 1, \ldots, N$. We then compute the **average pairwise cosine distance** of these representations:

$$\overline{D} \;=\; \frac{1}{\binom{N}{2}} \sum_{1 \le i < j \le N} \Big[1 - \frac{\mathbf{E}_i \cdot \mathbf{E}_j}{\|\mathbf{E}_i\| \; \|\mathbf{E}_j\|}\Big]. \tag{1}$$

This quantity reflects how "spread out" the embeddings are, with higher values indicating greater separation among representations.

2.2 Global Observations on Representation Dispersion

In this section, we examine how the model's representation space behaves across a broad sample of text segments, highlighting how perplexity, layer depth, and fine-tuning each affect embedding dispersion.

(1) Perplexity vs. Representation Dispersion. Our first finding connects a model's *sequence-level perplexity* to how spread out its contextual embeddings are.[1] We randomly select 100,000 text segments of 512 tokens, compute their perplexities, and also measure the **average pairwise distance** of their final-layer embeddings. We sort by perplexity and group segments into bins, and for each bin recording its mean perplexity and mean pairwise distance.

Figure 3 reveals a strong **negative correlation**: Segments with *lower* perplexity have *more dispersed* embeddings, whereas those with *higher* perplexity show *more compressed* embeddings. This trend appears across multiple model families (Llama, Phi, Mistral, Qwen) and diverse text domains (e.g. Wikitext-103, CNN Daily News, PubMed summarization). Full visualizations are provided in Section A.1.3.

We also verify that the relationship holds at a finer granularity by focusing on *last-token perplexity*, shown in Figure 4. Across context lengths of 16, 64, 128, and 256 tokens, we observe the same negative correlation trend. Intuitively, this negative correlation appears because contexts with more confident next-token predictions (low perplexity) end up pushed into more distinct regions of the embedding space, whereas harder-to-predict contexts remain more compressed.

[1] We define the sequence-level perplexity of a text segment $x_{1:L}$ of length L as:

$$\mathrm{ppl}(x_{1:L}) \;=\; \exp\!\Big(-\frac{1}{L}\sum_{t=1}^{L} \log p_\theta(x_t \mid x_{<t})\Big),$$

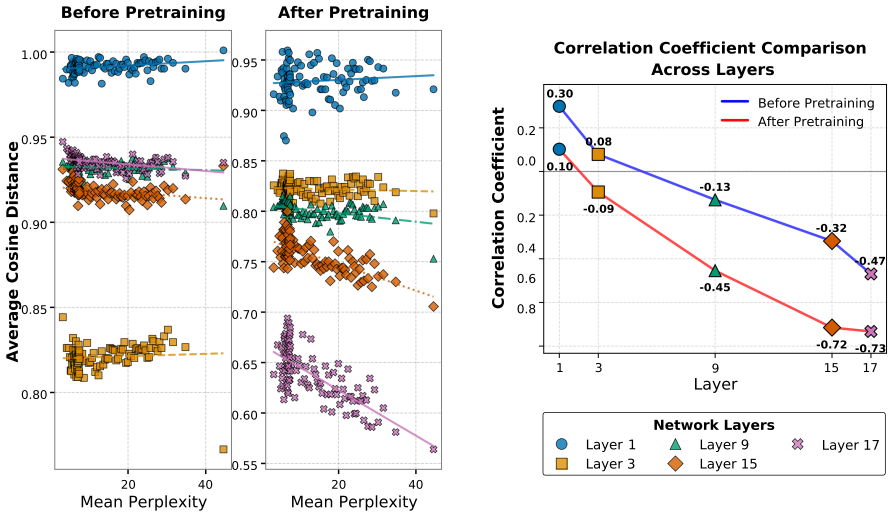

Figure 5: Layer-wise embedding separation in LLaMA-3.2-1B, shown across different perplexity bins (left panel) and the corresponding correlation coefficients (right panel). Note that the negative correlation between perplexity and embedding separation becomes more pronounced as we move to deeper layers (see right panel).

(2) Layer-Wise Patterns. We next assess how this relationship unfolds across layers. Collecting embeddings from multiple intermediate layers (e.g. Layers 1, 3, 9, 13, 17) and replicating the above procedure, we find that *the negative correlation strengthens in deeper layers* (as illustrated in the right panel of Figure 5). Early layers do not show a clear correlation, likely because they capture lower-level lexical features rather than global predictive cues. Deeper layers exhibit *more pronounced* embedding distance differences between easier-to-predict and harder-to-predict samples. Figure 5 illustrates this layer-wise progression for a representative LLaMA-3.2-1B model. Additionally, comparing models before and after pretraining indicates that the negative correlation emerges primarily after the model has been trained to predict tokens, pointing to a learned representational structure.

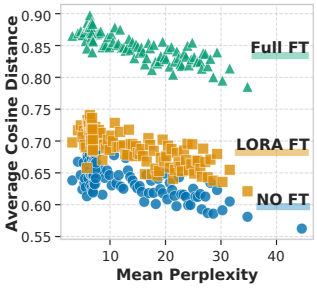

Figure 6: Effect of fine-Tuning on embedding separation.

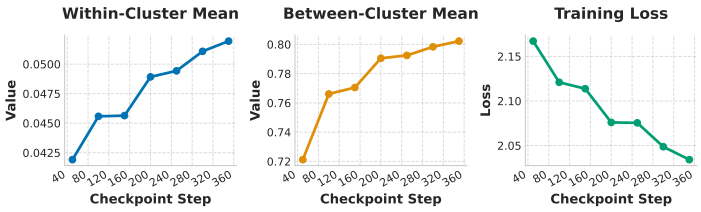

Figure 7: **Clustering metrics and training loss across training.** We form 500 clusters of WikiText-103 contexts (each cluster contains 100 semantically similar contexts that share the same 10-gram continuation). During model training, both the *within-cluster* distance (left) and *between-cluster* distance (middle) consistently grow, while training loss (right) falls.

(3) Fine-Tuning Effects. Finally, we examine how *fine-tuning* reshapes the embedding space. We select the same model (LLaMA-3.2-1B) and apply either parameter-efficient (*LoRA* (Hu et al., 2021)) or *full-parameter* fine-tuning on WikiText-103. Compared to the pre-trained model, both approaches *increase* the average embedding separation on WikiText-103 (Figure 6). Full fine-tuning exerts a stronger effect, pushing text samples further apart overall; LoRA, which adapts only low-rank modules, effects smaller but still notable changes. The choice of fine-tuning thus influences how discriminative (i.e. "spread out") the embeddings become in a specialized domain.

2.3 Dispersion Within Semantic Clusters: A Finer-Grained Analysis

A natural question is whether better models merely push *semantically dissimilar* contexts farther apart, or also increase distances among contexts that are *semantically close*. If the latter did not happen, then we could imagine a scenario where highly related examples remain tightly clustered, but unrelated examples spread out, thereby inflating overall average distances. We therefore directly measure dispersion among *clusters* of highly similar contexts to see whether their internal geometry also expands over training.

Setup & Metrics. We collect 10-grams from the WikiText-103 training set that appear at least 100 times. For each such 10-gram, we retrieve 100 occurrences, each accompanied by a 100-token left context, and treat these contexts as a semantically similar *cluster*. (Manual inspection confirmed that these context sets do indeed relate to the same event, entity, or theme.) We construct a total of 500 such clusters. We then track the contextual embeddings produced by a LLaMA-3.2-1B model at various checkpoints during training. Within each cluster, we compute the *within-cluster distance* as the average (cosine) distance from each embedding to the centroid of that cluster. Likewise, we define the *between-cluster distance* as the average (cosine) distance among the *centroids* of all clusters. Thus, the latter measures how far clusters are from each other in the embedding space, while the former measures how tightly each cluster is packed internally.

Results. Figure 7 shows that, as training proceeds and the loss decreases, both the average *within-cluster* distance and *between-cluster* distance increase. Thus, the trend toward higher overall dispersion is not driven solely by pushing apart highly dissimilar contexts. Even very similar contexts—which all share the same 10-gram continuation—become more spread out in the latent space as the model learns, which indicates the model can effectively distinguish them despite the similarities. This in-cluster expansion reinforces our hypothesis that better-performing models tend to produce broader embedding geometries in *all* regions of the latent space, not just pushing away dissimilar examples.

3 Applications

Having established a strong correlation between perplexity and embedding dispersion, we now illustrate how to leverage this observation in various practical scenarios.

3.1 Predicting Downstream Performance without Labeled Data

In many real-world scenarios, one receives a large *unlabeled* query set and must decide, *before* committing annotation or compute resources, (1) whether an off-the-shelf model will be sufficiently accurate and (2) which specific examples it is likely to get wrong. A reliable *label-free* indicator would enable rapid model validation, automatic justification of *easy* versus *hard* instances, and targeted continued pre-training on precisely those queries that the model currently struggles with. Because higher representation dispersion tracks lower perplexity (§2), we hypothesize that dispersion alone can serve as such an indicator: if high dispersion coincides with correct predictions, then simply measuring distances allows us to flag likely errors without ever looking at ground-truth labels.

Experimental protocol. To test this idea, we design a controlled experiment that varies the *fraction of correct answers* while keeping the input distribution fixed: (1) Collect a pool of question-answer pairs for a given dataset-model pair. (2) Partition the pool into *correct* and *incorrect* subsets by comparing the model's answer with the ground truth. (3) For each desired accuracy level (0%–100% in 10% increments), sample 100 queries that contain the requisite mix of correct and incorrect cases.(4) Extract the final-layer embeddings of these queries and compute their mean pairwise cosine distance, averaging over 10 random seeds.

Results. Figure 8 plots mean pairwise distance against the fraction of correct predictions for several LLaMA variants on ARC-Challenge and MMLU. Across all models and datasets, *accuracy rises monotonically with dispersion*: slices that the model answers correctly exhibit markedly broader geometry than those it answers incorrectly. Practically, a practitioner could therefore *sort* an unlabeled dataset by dispersion, inspect only the low-dispersion tail to uncover failure modes, or focus continued training on those "hard" queries. Taken together, these findings establish representation dispersion as

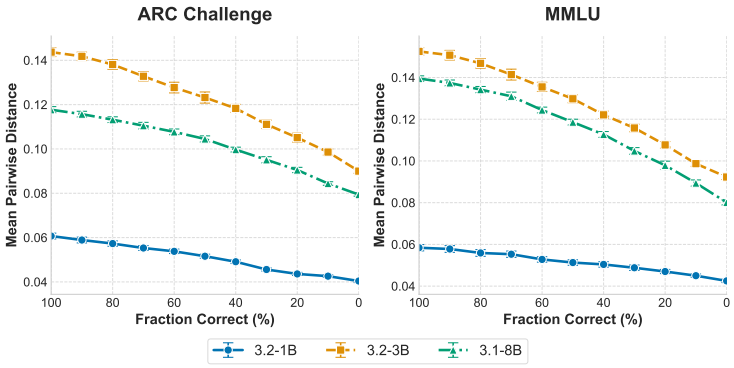

Figure 8: **Embedding distance vs. fraction correct.** Each point corresponds to a fixed mixture of correct/incorrect model predictions (x-axis). Error bars denote standard error over 10 random seeds. Higher accuracy consistently aligns with larger mean pairwise distance.

a powerful, zero-label proxy for downstream accuracy and a principled tool for slice discovery and targeted data augmentation.

3.2 Representation Dispersion for Model Selection

In practice, researchers and practitioners are often faced with choosing among numerous model variants—ranging from different instruction-tuned checkpoints to parameter-efficient adaptations or distilled models—while only having limited labeled data in the target domain. Exhaustively evaluating every checkpoint is typically prohibitively expensive, both in terms of computation and annotation resources. The tight link between representation dispersion and predictive accuracy established in §2 offers an attractive alternative: by simply measuring how broadly a model separates key tokens in its embedding space, one can obtain a geometric score that rapidly predicts downstream performance.

Setup. To operationalize this idea, we focus on task-relevant tokens that carry significant signal in the domain of interest, such as digits for mathematical reasoning, Python keywords for code generation, or legal terms for contract analysis. Let $\mathcal{T} \subset V$ denote a small set of such domain-specific tokens, and let $\overline{\mathcal{T}}$ denote a reference set of common, everyday language tokens. We use the model's original output token embeddings—that is, the rows of the output projection matrix—as provided after loading the model weights. This requires no forward passes or input data; all computations are performed directly on these pre-trained embeddings. We compute average pairwise distances—either cosine or Euclidean—among the embeddings of tokens within $\mathcal{T}$, within $\overline{\mathcal{T}}$, and between $\mathcal{T}$ and $\overline{\mathcal{T}}$.

Motivated by our empirical findings, we propose a single "dispersion gap" metric that succinctly captures both the distinctiveness of the domain-relevant tokens and their separation from generic language. Specifically, we define

$$\mathcal{G} = \text{within}(\mathcal{T}) + \text{between}\big(\mathcal{T}, \overline{\mathcal{T}}\big),$$

where $\text{within}(\mathcal{T})$ denotes the mean pairwise distance among the domain tokens, and $\text{between}\big(\mathcal{T}, \overline{\mathcal{T}}\big)$ is the mean distance between domain and reference tokens. Larger values of $\mathcal{G}$ indicate that the model both differentiates between domain-critical tokens and separates them from everyday vocabulary—a geometric pattern that, as Tables 3–4 show, strongly correlates with higher task accuracy.

This approach is computationally efficient and entirely label-free: evaluating $\mathcal{G}$ requires only reading the model's output embedding matrix and performing basic matrix operations on CPU, without any forward passes or GPU computation. In our experiments, ranking models by their dispersion gap consistently elevates the best-performing models in domains such as math and code, yielding gains of up to 40 accuracy points over lower-ranked alternatives. Thus, the dispersion gap offers a principled, data-efficient guideline for prioritizing model variants before more costly full-scale evaluation, providing practitioners with a practical tool for rapid, geometry-driven model selection.

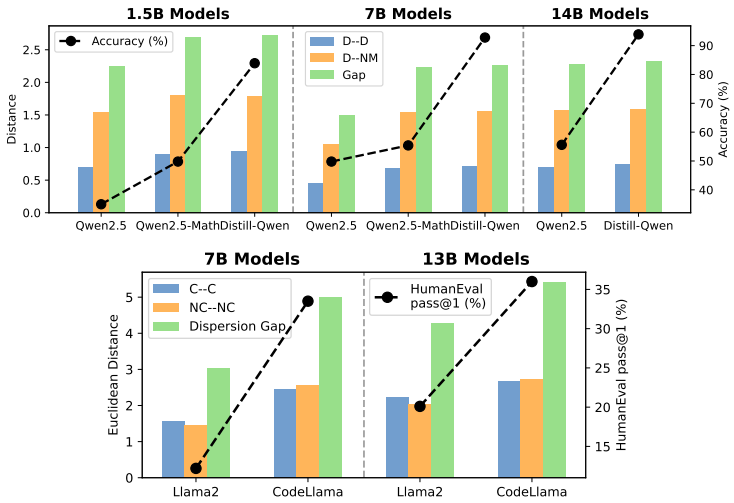

Figure 9: **Upper: Qwen on MATH.** Blue bars show D–D (digit–digit) distances, orange bars show D–NM (digit–non-math) distances, and green bars show their sum—the *Dispersion Gap*. The black dashed line reports task accuracy (%). **Lower: Llama/CodeLlama on HUMANEVAL.** Blue bars show C–C (code-keyword) distances, orange bars show NC–NC (non-code) distances, and green bars again give the *Dispersion Gap*. The black dashed line reports PASS@1 (%). Bar heights therefore convey the three dispersion statistics, while the line traces model performance, mirroring the figure legends. A full table of the underlying values is provided in Section B.2.

Results. Figure 9 clearly demonstrates how the dispersion gap $\mathcal{G}$ aligns with downstream performance. In the *upper* panel (Qwen on MATH), accuracy rises monotonically with the green "Gap" bars both *within* each parameter scale (e.g. 1.5 B $\rightarrow$ 7 B $\rightarrow$ 14 B) and *across* fine-tuned variants (e.g. QWEN2.5, QWEN2.5-MATH, DISTILL-QWEN). Spearman correlations between $\mathcal{G}$ and accuracy exceed 0.95, and the dispersion gap perfectly ranks all nine checkpoints without a single mis-ordering. The *lower* panel (Llama/CodeLlama on HUMANEVAL) shows the same pattern: CODELLAMA consistently exhibits both a larger gap and a higher PASS@1 rate than its LLAMA2 counterpart at *both* 7 B and 13 B scales, while the increase from 7 B to 13 B again boosts both metrics in lock-step. Taken together, these results confirm that the dispersion gap serves as a robust, zero-label proxy for real-world performance and can be relied upon to shortlist the strongest checkpoints before running any expensive evaluations.

3.3 LAYER SELECTION FOR KNN-LM

Many retrieval-augmented language models build a datastore by using one of the sublayers in the final transformer block as the vector key (Khandelwal et al., 2020; He et al., 2021; Alon et al., 2022; Li et al., 2024). A central question in these methods is which internal representation of the transformer to use as the "datastore key." In this section, we focus on kNN-LM, which augments a standard language model with a key-value memory of training examples. Specifically, at inference time, it retrieves the nearest neighbors of the current hidden state from that memory and interpolates their next-token distribution with the base LM's distribution, often achieving lower perplexity on rare or out-of-distribution tokens.

Recent evidence (Xu et al., 2023) suggests that taking the attention-layer output (instead of the feed-forward layer output) often improves kNN-LM perplexity, but pinpointing the best layer can require expensive end-to-end trials. We show that measuring how widely each layer "disperses" its text embeddings (via average pairwise cosine distance) provides a lightweight, *unsupervised* way to identify a layer likely to yield strong kNN-LM performance—without running the full interpolation pipeline.

Background. Consider the standard Transformer block as used by GPT-2. Let $\mathbf{h}^{(l-1)} \in \mathbb{R}^{n \times d}$ denote the hidden states from block $l-1$. In block l, the hidden states first pass

	N=10		N=50		N=100	
Model	Attn	FFN	Attn	FFN	Attn	FFN
DISTILGPT2	0.83±0.02	0.33±0.05	0.83±0.01	0.30±0.02	0.82±0.01	0.30±0.01
GPT2-SMALL	0.83±0.02	0.24±0.06	0.83±0.00	0.24±0.03	0.83±0.00	0.24±0.01
GPT2-MEDIUM	0.66±0.03	0.19±0.02	0.67±0.01	0.21±0.02	0.67±0.01	0.21±0.01
GPT2-LARGE	0.80±0.04	0.68±0.06	0.79±0.01	0.66±0.02	0.80±0.01	0.68±0.03

Table 1: Representation dispersion (average pairwise cosine distance) of the final block's attention and feed-forward sub-layers for varying numbers of randomly sampled chunks N. Higher values indicate more widely separated contextual embeddings.

through a multi-headed attention (MHA) sub-layer and residual connection: $\widehat{\mathbf{h}}^{(l)} = \mathbf{h}^{(l-1)} + \text{Dropout}\Big(\text{MHA}\big(\text{LayerNorm}(\mathbf{h}^{(l-1)})\big)\Big)$. We then apply another residual connection around a position-wise feed-forward network (FFN): $\mathbf{h}^{(l)} = \widehat{\mathbf{h}}^{(l)} + \text{Dropout}\Big(\text{FFN}\big(\text{LayerNorm}(\widehat{\mathbf{h}}^{(l)})\big)\Big)$. Thus, each block produces two intermediate sub-layer outputs: $\mathbf{h}_{\text{att}}^{(l)} = \widehat{\mathbf{h}}^{(l)}$ and $\mathbf{h}_{\text{ffn}}^{(l)} = \mathbf{h}^{(l)}$. Following Xu et al. (2023), we compare using $\mathbf{h}_{\text{att}}^{(L)}$ versus $\mathbf{h}_{\text{ffn}}^{(L)}$ from the *final* block L as the contextual key for kNN-LM.

Experimental Setup. We consider four members of the GPT-2 family—DISTILGPT2 (82 M parameters), GPT2-SMALL (117 M), GPT2-MEDIUM (345 M), and GPT2-LARGE (774 M)—all trained with the same tokenizer and a context length of 1 024. For each checkpoint we draw $N \in \{10, 50, 100\}$ non-overlapping 512-token *chunks* randomly from the WIKITEXT-103 validation split. Sampling is repeated 10 times with different random seeds, and the identical chunk indices are fed to both sub-layers so that any dispersion difference cannot be attributed to input variance. For every chunk we extract the *last-token* hidden state produced by the final Transformer block's (i) attention sub-layer output $\mathbf{h}_{\text{att}}^{(L)}$ and (ii) feed-forward sub-layer output $\mathbf{h}_{\text{ffn}}^{(L)}$. Given the resulting set of N vectors $\{\mathbf{h}_i\}_{i=1}^{N}$, we measure their *representation dispersion* as the average pairwise cosine distance. We report the mean and standard deviation of dispersion across the 10 random repeats for every $\langle$model, N, sub-layer$\rangle$ triple.

Results. Table 1 shows two practitioner-relevant patterns. First, the hidden states taken *after* the attention sub-layer are always more widely spread than those taken after the feed-forward sub-layer (e.g., 0.8045 vs. 0.6831 for GPT2-LARGE, 0.6593 vs. 0.1865 for GPT2-MEDIUM), making $\mathbf{h}_{\text{att}}^{(L)}$ the natural choice of key for kNN-LM. Second, dispersion can be estimated with striking efficiency: moving from 10 to 50 or 100 input chunks alters the mean by at most 1.5 % and never changes the layer ranking, so profiling a model requires only about $5{,}000$ tokens and a few milliseconds.

3.4 INCORPORATING REPRESENTATION DIVERGENCE INTO TRAINING

While typical cross-entropy training focuses purely on next-token prediction, several studies suggest that explicitly encouraging embedding separation can improve generalization and robustness in language modeling (Gunel et al., 2021; Jain et al., 2023). Inspired by these findings, we augment the standard language-modeling loss with an auxiliary objective that *pushes apart* hidden-state vectors, aiming to produce more discriminative representations.

Setup. We consider two scenarios: a *single-domain* setting, where we train GPT-2 small on WikiText, and a *cross-domain* setting, where we train on WikiText plus Python code. In both cases, let $\{\mathbf{h}_i\}_{i=1}^{B}$ be the final-layer hidden-state vectors for all tokens in a batch (flattened across batch and sequence). We normalize each vector to unit length, $\tilde{\mathbf{h}}_i = \mathbf{h}_i / \|\mathbf{h}_i\|$. To encourage wider separation, we compute an average pairwise cosine distance d and then add an auxiliary loss $-d$ to the standard cross-entropy loss, weighted by a hyperparameter λ.

In the **single-domain** setting, d_{avg} is defined over all pairs in the same batch: $d_{\text{avg}} = \frac{1}{B(B-1)} \sum_{i \neq j} \Big[1 - \tilde{\mathbf{h}}_i \cdot \tilde{\mathbf{h}}_j\Big]$. In the **cross-domain** setting, we instead compute d only across pairs drawn from different domains (Wiki vs. code) to push embeddings from each domain further apart:

$d = \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} \left[1 - \tilde{\mathbf{h}}_i^{(\mathrm{A})} \cdot \tilde{\mathbf{h}}_j^{(\mathrm{B})}\right]$. The total loss in both settings is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{aux}}$, where λ controls the strength of the auxiliary "spread-out" loss.

Results. Table 2 reports test perplexities for various learning rates and auxiliary loss weights λ. In the **single-domain** (WikiText) setting, introducing the auxiliary spread-out loss yields a slight decrease in perplexity—typically 1–4 points—relative to the baseline, especially early in training. In the **cross-domain** (WikiText+Code) setting, the auxiliary loss produces a much more pronounced reduction in perplexity for both domains. For all learning rates, models trained with a moderate value of λ achieve notably lower perplexity on both WikiText and code, suggesting that explicitly pushing apart representations from distinct domains leads to more specialized and di scriminative features. This demonstrates that our approach is particularly effective when bridging heterogeneous data sources.

LR	**(a) Single-Domain (WikiText)**					**(b) Cross-Domain (WikiText+Code)**				
	λ	**Step = 500**		**Step = 1000**		λ	**Step = 500**		**Step = 1000**	
		Base	**+Aux**	**Base**	**+Aux**		**Wiki**	**Code**	**Wiki**	**Code**
10^{-3}	0.0	226.1	–	111.3	–	0.0	295.6	36.9	171.7	23.8
	0.1	–	217.5	–	108.2	0.001	270.8	34.5	158.8	22.3
7×10^{-4}	0.0	195.0	–	96.7	–	0.0	304.4	35.4	175.7	22.8
	0.1	–	193.8	–	93.6	0.01	255.2	31.9	150.2	20.8
5×10^{-4}	0.0	166.2	–	83.0	–	0.0	268.5	33.7	166.9	22.1
	0.1	–	165.6	–	82.0	0.02	253.3	30.5	155.2	20.5

Table 2: **Auxiliary spread-out loss improves perplexity in both single- and cross-domain settings.** Left: single-domain (WikiText) results; right: cross-domain (WikiText+Code) results. λ is the auxiliary loss weight, chosen by validation for each learning rate. We report test-set perplexities at 500 and 1000 steps.

4 Related Work

Geometric Analysis of Embeddings in Language Models. A growing body of work has examined the geometry of hidden representations in large language models (LLMs). Early studies identified an anisotropy problem, where embeddings collapse into a narrow cone and lose expressiveness at deeper layers (Mu & Viswanath, 2018; Ethayarajh, 2019; Gao et al., 2019; Li et al., 2020; Noci et al., 2022). Recent work uses intrinsic dimension (ID) estimators to trace how representation manifolds evolve across layers (Valeriani et al., 2023), linking geometry to performance through, for example, distinguishing human vs. machine text (Tulchinskii et al., 2023), predicting data compressibility (Cheng et al., 2023), and revealing simplex-like structures for categorical concepts (Park et al., 2025).

The study most closely related to ours is Viswanathan et al. (2025), which also analyzes token-level embedding distributions and observes cosine similarity rises when tokens in the prompt are shuffled. However, their work remains largely descriptive. In contrast, we show how representation dispersion can predict and improve perplexity and downstream accuracy, making geometric insights actionable for model evaluation and selection.

5 Conclusion

In this work, we showed that representation dispersion serves as both a practical diagnostic and training signal for language models. Moving forward, we aim to investigate how representation dispersion interacts with other design choices—such as architectural variations or tokenization strategies—and whether additional regularization signals might further strengthen model robustness and interpretability. We hope these directions will inspire new ways to harness embedding geometry for next-generation language modeling and related tasks.

STATEMENT ON LLM USAGE

We acknowledge the use of Large Language Models (LLMs) to assist in the preparation of this manuscript. Specifically, LLMs were utilized to improve grammar and clarity, aid in literature discovery, and generate boilerplate code snippets for our experiments and testing scripts. The authors have carefully reviewed and edited all LLM-generated outputs and take full responsibility for the final content and scientific integrity of this work.

REFERENCES

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. Neuro-symbolic language modeling with automaton-augmented retrieval. In Chaudhuri, Kamalika and Jegelka, Stefanie and Song, Le and Szepesvari, Csaba and Niu, Gang and Sabato, Sivan (ed.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 468–485. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/alon22a.html.

Emily Cheng, Corentin Kervadec, and Marco Baroni. Bridging information-theoretic and geometric compression in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12397–12420, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.762. URL https://aclanthology.org/2023.emnlp-main.762/.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. In *EMNLP-IJCNLP 2019, Hong Kong, China*, pp. 55–65, 2019.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA*, 2019.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. Supervised contrastive learning for pre-trained language model fine-tuning, 2021. URL https://arxiv.org/abs/2011.01403.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5703–5714, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.461. URL https://aclanthology.org/2021.emnlp-main.461.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

Nihal Jain, Dejiao Zhang, Wasi Uddin Ahmad, Zijian Wang, Feng Nan, Xiaopeng Li, Ming Tan, Ramesh Nallapati, Baishakhi Ray, Parminder Bhatia, Xiaofei Ma, and Bing Xiang. Contra-CLM: Contrastive learning for causal language model. 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. 6436–6459, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.355. URL https://aclanthology.org/2023.acl-long.355/.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*, 2020.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on EMNLP 2020, Online, November 16-20, 2020*, pp. 9119–9130, 2020.

Yanhong Li, Karen Livescu, and Jiawei Zhou. Chunk-distilled language modeling, 2024. URL https://arxiv.org/abs/2501.00343.

Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *ICLR 2018, Vancouver, BC, Canada*, 2018.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022.

Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models, 2025. URL https://arxiv.org/abs/2406.01506.

Eduard Tulchinskii, Kristian Kuznetsov, Laida Kushnareva, Daniil Cherniavskii, Sergey Nikolenko, Evgeny Burnaev, Serguei Barannikov, and Irina Piontkovskaya. Intrinsic dimension estimation for robust detection of ai-generated texts. *Advances in Neural Information Processing Systems*, 36: 39257–39276, 2023.

Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. *Advances in Neural Information Processing Systems*, 36:51234–51252, 2023.

Karthik Viswanathan, Yuri Gardinazzi, Giada Panerai, Alberto Cazzaniga, and Matteo Biagetti. The geometry of tokens in internal representations of large language models, 2025. URL https://openreview.net/forum?id=an3jH2qD2r.

Frank F. Xu, Uri Alon, and Graham Neubig. Why do nearest neighbor language models work? In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Technical Appendices

Organization of Contents

A Supplemental Materials for Representation Geometry Analysis

A.1 Details regarding Sequence-level Perplexity Experiments

A.1.1 Datasets and Models

In this section, we provide additional experimental details and visualizations that supplement our main empirical analysis in §2. We study a range of standard language modeling datasets, including the Salesforce/wikitext [2], abisee/cnn_dailymail [3], and ccdv/pubmed-summarization [4], covering text segments in diverse domains.

For the models, our experiments encompass:

- **Llama** families: `meta-llama/Llama-3.2-1B`, `meta-llama/Llama-3.2-3B`, `meta-llama/Llama-3.1-8B`
- **Gemma** families: `google/gemma-2-2b`, `google/gemma-2-9b`
- **Mistral**: `mistralai/Mistral-7B-v0.1`
- **Phi**: `microsoft/phi-2`
- **Qwen** families: `Qwen/Qwen2.5-0.5B`, `Qwen/Qwen2.5-3B`, `Qwen/Qwen2.5-7B`

We use the Hugging Face implementation of the above models. All models are standard decoder-only Transformers, for which we collect final-layer embeddings on randomly selected text segments. In line with Equation 1 of the main paper, we measure average pairwise cosine distance to quantify how "spread out" their representations are.

A.1.2 Procedure for Mean-Perplexity vs. Dispersion Analysis

Here, we outline the steps needed to produce a mean-perplexity vs. representation-dispersion plot:

Step 1: Randomly sample 100,000 segments (e.g., 512 tokens each) from the data.

Step 2: For each segment:

a) Compute its perplexity over the full sequence.

b) Record the final-layer hidden states for later analysis.

Step 3: Sort all segments by their computed perplexity.

Step 4: Group the sorted segments into bins (e.g., 100 segments per bin) and record each bin's mean perplexity.

Step 5: Perform uniform sampling in perplexity space on these bins to ensure coverage of low-, mid-, and high-perplexity regions.

Step 6: For each uniformly sampled bin:

a) Retrieve the saved hidden states.

b) Calculate pairwise distances (e.g., average cosine distance) among the segment embeddings.

Step 7: Produce the final mapping of mean perplexity to average pairwise distance.

Uniform Perplexity Sampling. Since random sampling of text segments often yields a distribution heavily concentrated around moderate perplexities, we use a uniform sampling scheme to cover both low- and high-perplexity "tails." The pseudocode below highlights the procedure used in **Step 5** (Algorithm 1):

[2] `http://huggingface.co/datasets/Salesforce/wikitext`

[3] `https://huggingface.co/datasets/abisee/cnn_dailymail`

[4] `https://huggingface.co/datasets/ccdv/pubmed-summarization`

Algorithm 1 Uniform Perplexity Binning

Require: A sorted list of G perplexity bins $\{b_1, \ldots, b_G\}$ (with means $m_1 \leq \cdots \leq m_G$)
Require: Desired number of bins K
Ensure: A set of K bins sampled uniformly in perplexity
1: $m_{\min} \leftarrow m_1$; $\quad m_{\max} \leftarrow m_G$
2: Define targets

$$t_k \;=\; m_{\min} + \frac{k-1}{K-1}\,(m_{\max} - m_{\min}) \quad \text{for } k = 1, \ldots, K$$

3: $selected \leftarrow \emptyset$
4: **for** $k \leftarrow 1$ **to** K **do**
5: find j s.t. m_j is closest to t_k
6: $selected \leftarrow selected \cup \{j\}$
7: **end for**
8: **if** $|selected| < K$ **then**
9: add extra bins from the sorted list until you have K
10: **end if**
11: **return** $\{\, b_j : j \in selected\}$

This ensures we sample across the entire perplexity spectrum, capturing both rare, low-ppl segments and rare, high-ppl segments. With these selected bins in hand, we can then compute the final-layer embeddings and measure representation dispersion to obtain a mean-ppl vs. dispersion plot.

A.1.3 Additional Visualizations

Below, we present the full set of perplexity-versus-dispersion plots referenced in §2.2. For each dataset and model, we group 100,000 text segments into perplexity bins and compute their average pairwise representation distances. As described in the main text, we observe a negative correlation between sequence-level perplexity and representation dispersion.

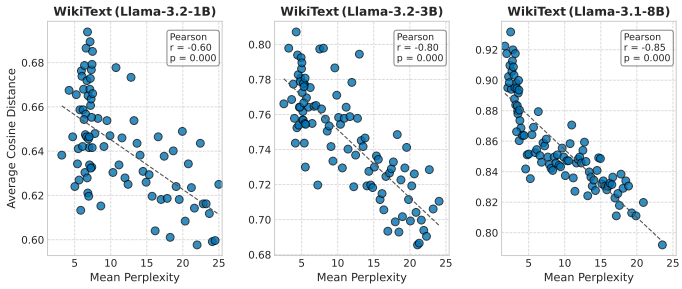

Figure 10: Perplexity vs. Average Pairwise Cosine Distance on Wikitext-103 (Llama family).

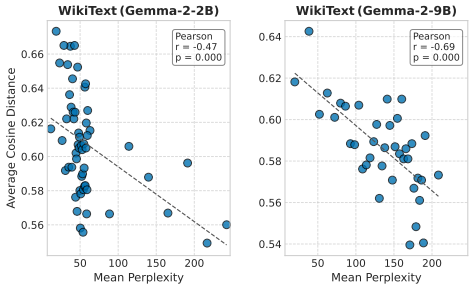

Figure 11: Perplexity vs. Average Pairwise Cosine Distance on Wikitext-103 (Gemma family).

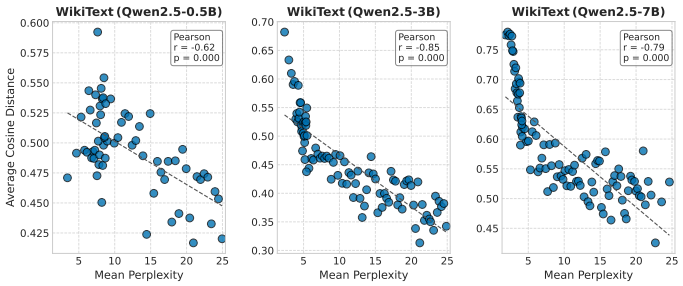

Figure 12: Perplexity vs. Average Pairwise Cosine Distance on Wikitext-103 (Qwen family).

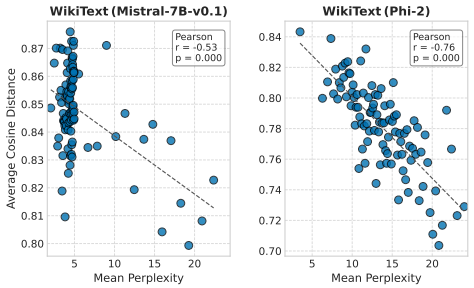

Figure 13: Perplexity vs. Average Pairwise Cosine Distance on Wikitext-103 (Mistral) and Phi.

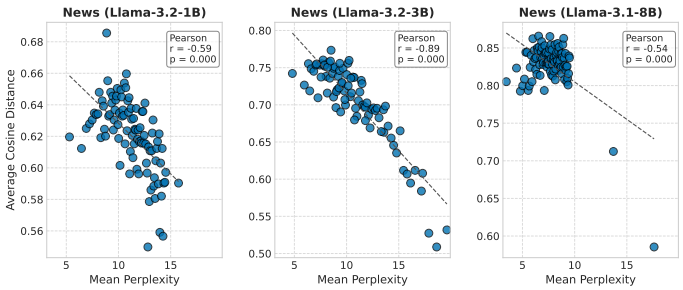

Figure 14: Perplexity vs. Average Pairwise Cosine Distance on CNN DailyMail (Llama family).

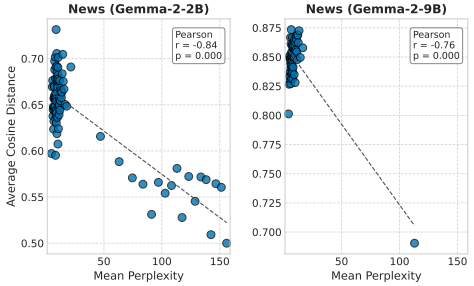

Figure 15: Perplexity vs. Average Pairwise Cosine Distance on CNN DailyMail (Gemma family).

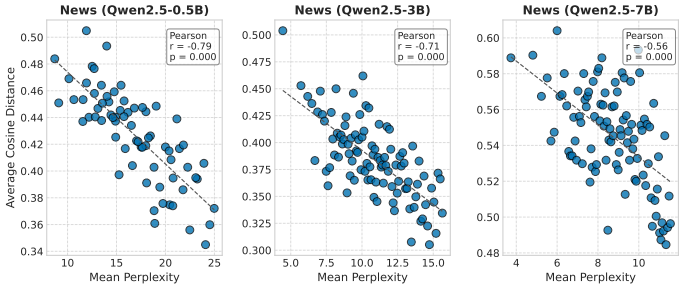

Figure 16: Perplexity vs. Average Pairwise Cosine Distance on CNN DailyMail (Qwen family).

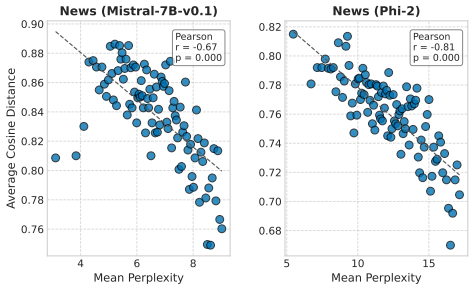

Figure 17: Perplexity vs. Average Pairwise Cosine Distance on CNN DailyMail (Mistral, Phi).

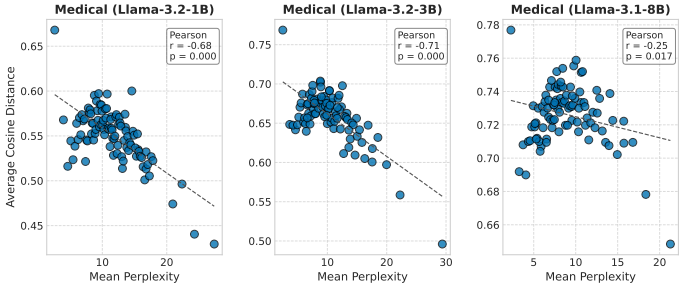

Figure 18: Perplexity vs. Average Pairwise Cosine Distance on PubMed Summarization (Llama family).

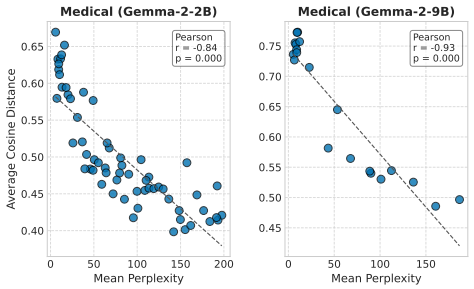

Figure 19: Perplexity vs. Average Pairwise Cosine Distance on PubMed Summarization (Gemma family).

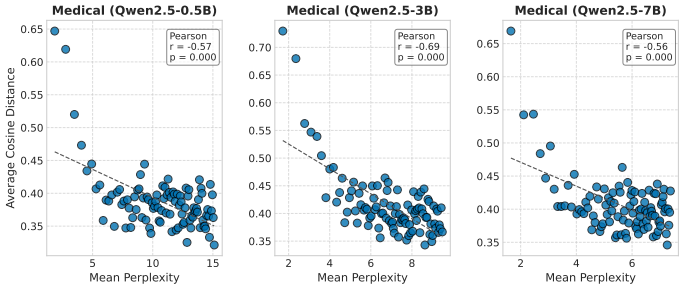

Figure 20: Perplexity vs. Average Pairwise Cosine Distance on PubMed Summarization (Qwen family).

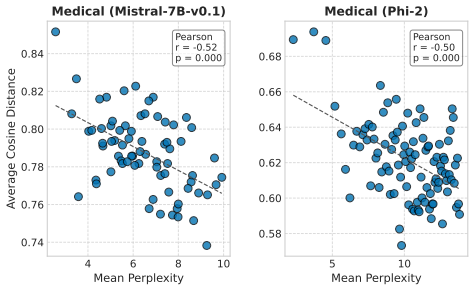

Figure 21: Perplexity vs. Average Pairwise Cosine Distance on PubMed Summarization (Mistral, Phi).

As shown in the figures, the observed *negative correlation* between sequence-level perplexity and representation dispersion holds consistently across:

- Multiple model sizes and architectures (Llama, Gemma, Mistral, Phi, Qwen).
- Multiple data domains (Wikitext-103, CNN DailyMail, PubMed Summarization).

These findings support the main paper's claim that *lower-perplexity* contexts tend to occupy more "spread out" regions in the final-layer embedding space, while *higher-perplexity* (i.e., more challenging) contexts appear more compressed.

A.2 Details regarding Fine-Tuning Effect Experiments

In §2.2, we examined how fine-tuning influenced representation dispersion. Below are the hyperparameters for the two fine-tuned LLaMA-3.2-1B models used in our experiments. We fine-tuned both checkpoints using the open-source LLaMA-Factory framework [5].

A.2.1 LoRA Fine-Tuned Model

This model is a fine-tuned version of `meta-llama/Llama-3.2-1B` on the Wikitext-103 dataset. It achieved the following on the evaluation set:

- **Loss:** 2.1764

Training Hyperparameters.

- `learning_rate`: 0.0001
- `train_batch_size`: 8
- `eval_batch_size`: 1
- `seed`: 42
- `gradient_accumulation_steps`: 8
- `total_train_batch_size`: 64
- `optimizer`: `adamw_torch` with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, no additional arguments
- `lr_scheduler_type`: `cosine`
- `lr_scheduler_warmup_ratio`: 0.1
- `num_epochs`: 1.0

A.2.2 Full-Parameter Fine-Tuned Model

This model is also a fine-tuned version of `meta-llama/Llama-3.2-1B` on the Wikitext-103 dataset. It achieved the following on the evaluation set:

- **Loss:** 2.1333

Training Hyperparameters.

- `learning_rate`: 1e-05
- `train_batch_size`: 2
- `eval_batch_size`: 1
- `seed`: 42
- `distributed_type`: `multi-GPU`
- `num_devices`: 2
- `gradient_accumulation_steps`: 16
- `total_train_batch_size`: 64
- `total_eval_batch_size`: 2
- `optimizer`: `Adam` with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$
- `lr_scheduler_type`: `cosine`
- `lr_scheduler_warmup_ratio`: 0.1
- `num_epochs`: 5.0

[5] https://github.com/hiyouga/LLaMA-Factory

A.3 Details regarding Dispersion Within Semantic Clusters Training Hyperparameters

In §2.3, we examined how dispersion evolves within carefully constructed *semantic clusters* of text segments that share the same 10-gram continuation. Below are the training hyperparameters for the model used in this experiment:

- `learning_rate`: 1e-05
- `train_batch_size`: 10
- `eval_batch_size`: 1
- `seed`: 42
- `distributed_type`: `multi-GPU`
- `num_devices`: 8
- `gradient_accumulation_steps`: 8
- `total_train_batch_size`: 640
- `total_eval_batch_size`: 8
- `optimizer`: `ADAMW_TORCH` with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, no additional arguments
- `lr_scheduler_type`: `cosine`
- `lr_scheduler_warmup_ratio`: 0.1
- `num_epochs`: 5.0

We used these hyperparameters to train the model from a checkpoint of `meta-llama/Llama-3.2-1B` on WikiText-103, then tracked within-cluster and between-cluster distances of the resulting contextual embeddings at several checkpoints during training. The model is also fine-tuned using the open-source LLaMA-Factory framework.

B Supplemental Materials for Applications of Representation Dispersion

B.1 Additional Results for Predicting Downstream Performance without Labeled Data

Below we provide extended experimental results following the methodology of §3.1. Each figure contains results for three models: **Llama-3.2-1B-Instruct**, **Llama-3.2-3B-Instruct**, and **Llama-3.1-8B-Instruct**.

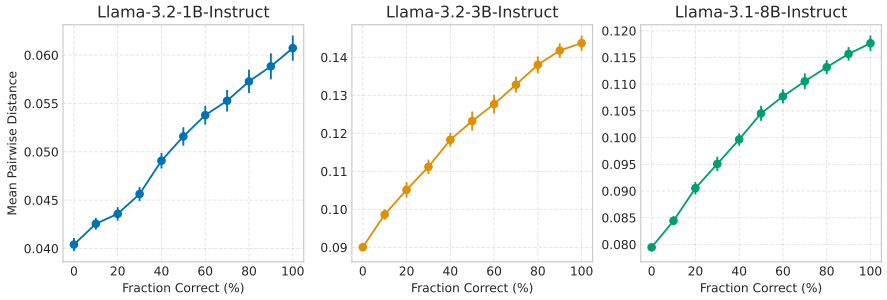

Figure 22: Downstream performance estimation on ARC Challenge (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

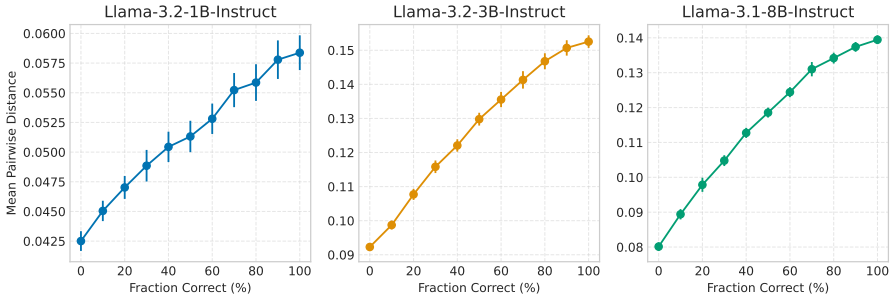

Figure 23: Downstream performance estimation on MMLU (English) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

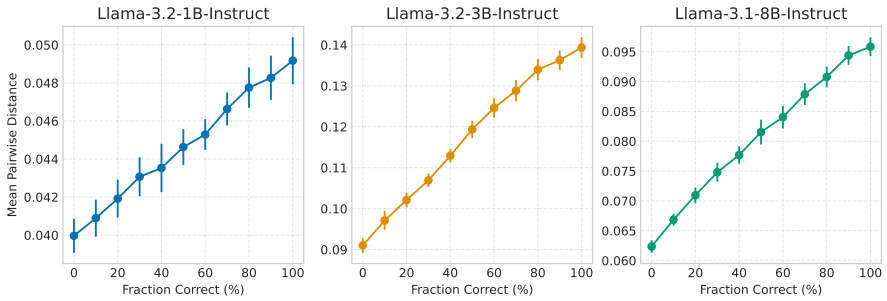

Figure 24: Downstream performance estimation on Multilingual MMLU (German) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

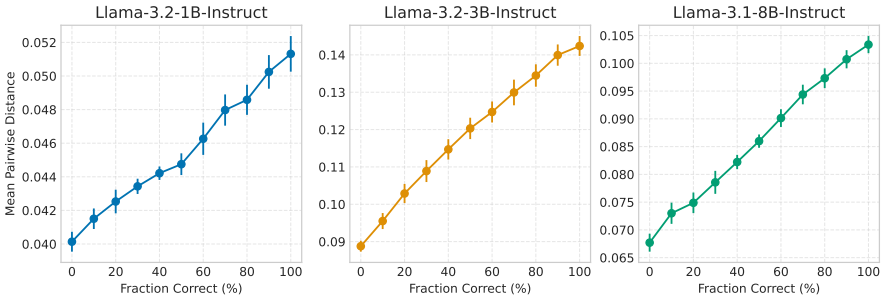

Figure 25: Downstream performance estimation on Multilingual MMLU (Spanish) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

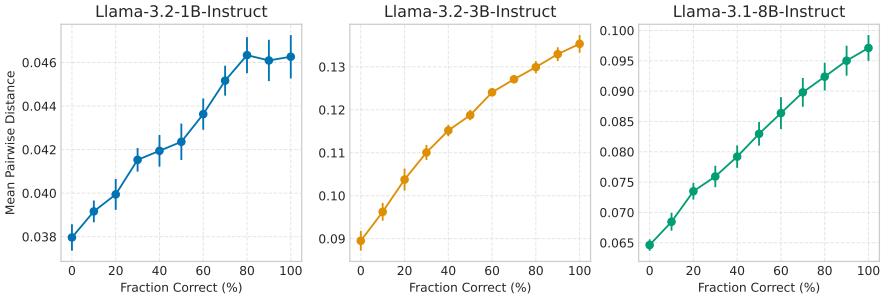

Figure 26: Downstream performance estimation on Multilingual MMLU (French) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

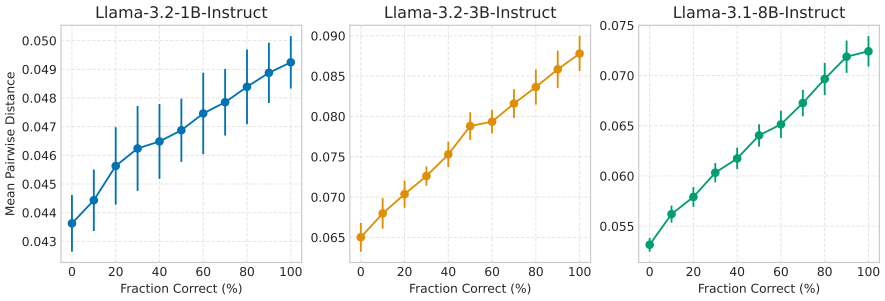

Figure 27: Downstream performance estimation on Multilingual MMLU (Hindi) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

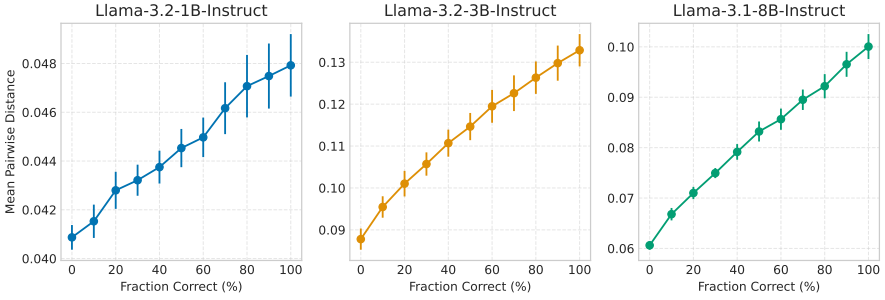

Figure 28: Downstream performance estimation on Multilingual MMLU (Italian) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

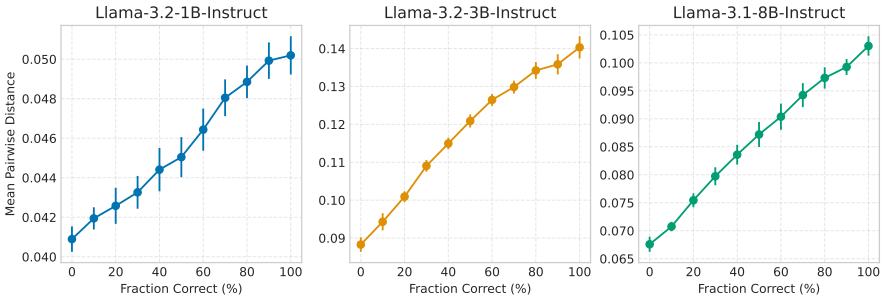

Figure 29: Downstream performance estimation on Multilingual MMLU (Portuguese) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

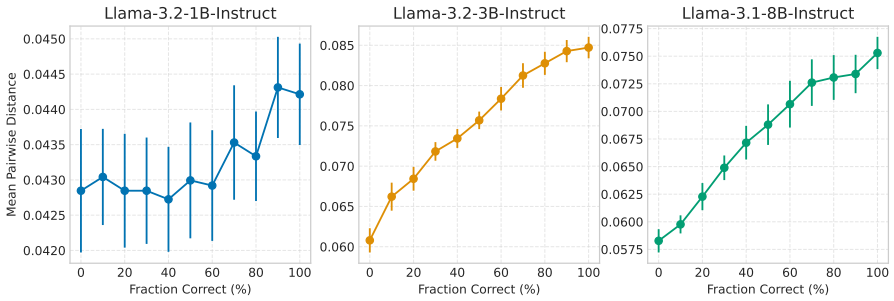

Figure 30: Downstream performance estimation on Multilingual MMLU (Thai) (containing results for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct).

B.2 Details Regarding Representation Dispersion for Model Selection

This appendix compiles the full numeric statistics that underpin the analyses in §3.2. We report complete Euclidean and cosine distance figures for every model variant, together with their task-specific accuracies, so that readers can perform fine-grained checks, reproduce our correlation calculations, and explore alternative dispersion metrics. Table 3 and Table 4 complement the visual summaries in Figure 9 by exposing each component of the *Dispersion Gap* in detail.

Table 3: **Embedding Dispersion vs. MATH Performance (Qwen Variants).** We show the average **Euclidean** and **Cosine** distances among digit embeddings (D–D), among non-math tokens (NM–NM), and between digits and non-math tokens (D–NM). We also list each model's accuracy on MATH (%). Larger D–D distances indicate that numeric tokens are placed more distinctly from each other in embedding space, and similarly for NM–NM. A greater D–NM distance implies stronger separation between numeric tokens and everyday text.

Model	Euclidean Distances			Cosine Distances			MATH (%)
	D–D	NM–NM	D–NM	D–D	NM–NM	D–NM	
Qwen2.5-1.5B	0.70	1.46	1.54	0.25	0.93	1.11	35.0
Qwen2.5-Math-1.5B	0.89	1.67	1.80	0.33	0.85	1.15	49.8
Distill-Qwen-1.5B	0.94	1.60	1.79	0.36	0.84	1.16	83.9
Qwen2.5-7B	0.45	0.93	1.05	0.18	0.95	1.09	49.8
Qwen2.5-Math-7B	0.69	1.47	1.54	0.27	0.91	1.14	55.4
Distill-Qwen-7B	0.72	1.45	1.55	0.28	0.91	1.15	92.8
Qwen2.5-14B	0.70	1.71	1.58	0.26	0.96	1.01	55.6
Distill-Qwen-14B	0.74	1.69	1.59	0.28	0.96	1.03	93.9

Table 4: **Embedding Dispersion vs. HumanEval Performance (Llama2 vs. CodeLlama).** We compare the average **Euclidean** and **Cosine** distances among code tokens (C–C), among non-code tokens (NC–NC), and between code and non-code (C–NC). We also list each model's HumanEval pass@1 (%).

Model	Euclidean Distances			Cosine Distances			HumanEval (%)
	C–C	NC–NC	C–NC	C–C	NC–NC	C–NC	
Llama2-7B	1.56	1.46	1.52	0.96	0.95	0.97	12.2
CodeLlama-7B	2.44	2.56	2.54	0.94	0.94	0.97	33.5
Llama2-13B	2.24	2.04	2.15	0.93	0.89	0.93	20.1
CodeLlama-13B	2.68	2.74	2.75	0.94	0.93	0.97	36.0

B.3 Additional Results for Layer Selection for kNN-LM

We present extended findings on sub-layer selection for *k*NN-LM. Figure 31 displays results for *four* GPT-2 variants (`distilgpt2`, `gpt2`, `gpt2-medium`, `gpt2-large`). As in the main text, each point represents a 512-token chunk of text, with its mean perplexity plotted against the sub-layer's average pairwise cosine distance (blue for the attention output, red for the feed-forward output). Interestingly, the negative correlation is weaker for the attention output than for the feed-forward output.

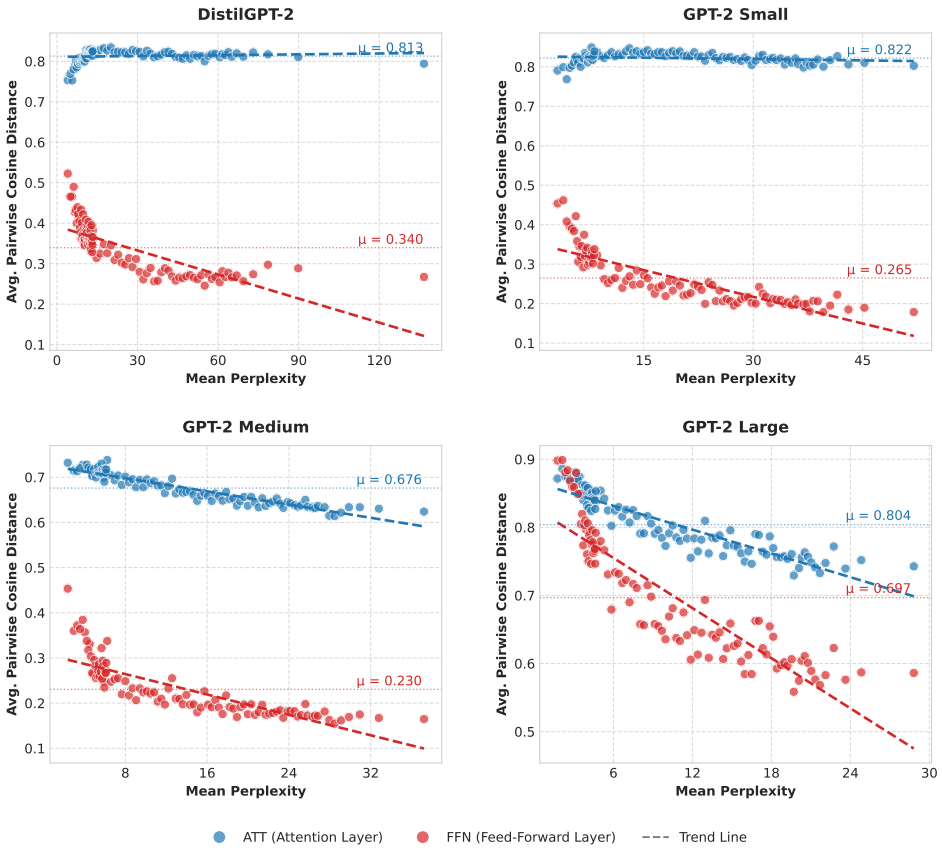

Figure 31: Mean perplexity vs. sub-layer average pairwise cosine distance for four GPT-2 variants (`distilgpt2`, `gpt2`, `gpt2-medium`, `gpt2-large`). Each point is a 512-token chunk of text.

B.4 Training Details for Incorporating Representation Dispersion

All experiments in §3.4 were conducted on NVIDIA A100 80GB GPUs.

Single-Domain Setting. We train GPT2-SMALL from scratch on WikiText, using a batch size of 64 and a block size (sequence length) of 512. The auxiliary loss weight λ is tuned over the set:

$$\{0.5,\ 0.2,\ 0.1,\ 0.07,\ 0.05,\ 0.02,\ 0.01,\ 0.007,\ 0.005,\ 0.002,\ 0.001\}$$

We experiment with learning rates $\{1 \times 10^{-3},\ 7 \times 10^{-4},\ 5 \times 10^{-4}\}$.

Cross-Domain Setting. For joint WikiText + Python code training, we similarly use GPT2-SMALL (from scratch), a batch size of 128, and a block size of 256. The auxiliary loss weight λ and learning rates are swept over the same sets as above:

- λ values: $\{0.5,\ 0.2,\ 0.1,\ 0.07,\ 0.05,\ 0.02,\ 0.01,\ 0.007,\ 0.005,\ 0.002,\ 0.001\}$
- Learning rates: $\{1 \times 10^{-3},\ 7 \times 10^{-4},\ 5 \times 10^{-4}\}$

For both settings, we select λ by validation for each learning rate. All experiments use standard AdamW optimizer settings unless otherwise specified.

C LIMITATIONS

While our findings underscore a strong empirical link between representation dispersion and model performance, there are several limitations. First, our analyses focus on average pairwise cosine distances of final-layer representations, which may not capture all nuanced aspects of embedding geometry or model behavior. Second, although we observe consistent negative correlations between dispersion and perplexity across several model families and domains, causality cannot be definitively concluded; certain architectures or objectives may modulate this relationship in unforeseen ways. Third, our experiments center primarily on English text from standard benchmarks and a limited set of specialized domains (e.g. code, scientific abstracts). It remains unclear how well our observations extend to other languages, modalities, or highly domain-specific corpora. Further research is needed to fully understand these trade-offs and develop robust methods for controlling embedding geometry.