PROMPT2BOX: Improving LLM Weakness Discovery and Specificity Estimation by Uncovering Entailment Structure among Prompts

Neeladri Bhuiya[1,3] * **Shib Sankar Dasgupta**[2]*
Andrew McCallum[1] **Haw-Shiuan Chang**[4]*
[1]CICS, University of Massachusetts Amherst, USA
[2]Amazon AWS AI [3]A10 Networks [4]University of Arizona
nbhuiya@a10networks.com, shibdg@amazon.com
mccallum@cs.umass.edu, hawshiuan@arizona.edu

Abstract

To discover the weaknesses of LLMs, researchers often embed prompts into a vector space and cluster them to extract insightful patterns. However, vector embeddings primarily capture topical similarity; as a result, prompts that share a topic but differ in specificity, and consequently in difficulty, are often represented similarly, making fine-grained weakness analysis difficult. To address this limitation, we propose PROMPT2BOX, which embeds prompts into a box embedding space using a trained encoder. The encoder, trained on existing and synthesized datasets, outputs box embeddings that capture not only semantic similarity but also specificity relations between prompts (e.g., "*writing an adventure story*" is more specific than "*writing a story*"). We further develop a novel dimension reduction technique for box embeddings to facilitate dataset visualization and comparison. Our experiments demonstrate that box embeddings consistently capture prompt specificity better than vector baselines and achieve 45% error reduction on average in predicting specificity compared to the prompt length baseline. On the downstream task of creating hierarchical clustering trees for 17 LLMs from the UltraFeedback dataset, PROMPT2BOX can identify 13.5% more LLM weaknesses than vector baselines and achieves an approximately 33% stronger correlation between hierarchical depth and instruction specificity. The code is available at https://github.com/zawedcvg/box_embeddings

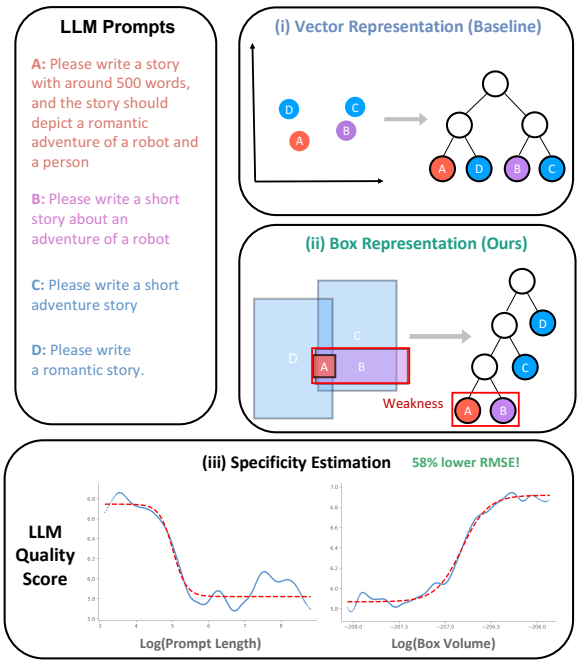


Figure 1: Comparison between the widely-used vector representation in (i) and our box representation in (ii) for analyzing the performance of an LLM on four prompts. Blue means that the LLM achieves a high performance on the prompt while red means the opposite. Our approach correctly highlights "writing robot-adventure stories" as a weakness of the LLM by clustering prompt *A* and *B*. In (iii), we visualize the scores of LLama-2-13b-Chat's responses versus specificity estimates based on length and box volume

1 Introduction

For large language model (LLM) developers, it is crucial to identify an LLM's weaknesses to guide the next round of collection of additional high-quality training data. When developers observe a bad response from an LLM to a prompt, there are two main possible causes: 1) the LLM has a weakness in that area or topic, or 2) the prompt itself is very specific, which makes LLM unable to extrapolate from its training data. This induces two challenging questions: 1) how to discover those weakness topics, and 2) how to measure specificity.

Several prior works (Jiang et al., 2024; Tamkin et al., 2024; Zeng et al., 2025; Tian et al., 2025) address the first challenge by embedding every

*The work was primarily done while the authors were at UMass Amherst.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 56643–56670
October 24-29, 2026 ©2026 Association for Computational Linguistics

prompt into a vector and hierarchically clustering the prompts based on vector similarity. By analyzing performance differences across these clusters in a two-dimensional projection of the embedding space, LLM developers can diagnose systematic weaknesses relative to competing models. However, the vector-based methods rely on the assumption that LLMs perform similarly for similar prompts, without accounting prompt specificity or difficulty. Consequently, vectors can group two prompts that are topically similar but have different levels of difficulty together. Ignoring prompt specificity loses an entire mode of failure, making it difficult to attribute the actual reason of a weakness.

Recent studies (Sun et al., 2024; Atmakuru et al., 2024; Lu et al., 2025; Jaroslawicz et al., 2025; Zhang et al., 2025b) demonstrate that adding constraints to a prompt reduces the solution space, making the prompt more specific and more difficult for an LLM to answer.[1] However, there is no standard method to measure prompt specificity. Although the simple length baseline often highly correlates with specificity, Li and Nenkova (2015); Gao et al. (2019); Kapur et al. (2026) show that there is usually room of improvements in each particular application.

Figure 1 (i) illustrates a limitation of vector-based methods. Prompt *A* is semantically similar to prompt *D*, but considerably more specific. In a hierarchical clustering based on vector embeddings, the low score of *A* drags down the average score of the cluster containing both *A* and *D*, causing that cluster to be flagged as a weakness of the LLM. However, this conclusion is misleading: since the cluster broadly captures the ability to *"write a romantic story"*, the high score of *D* already demonstrates that the LLM is competent at this general task. The poor performance on *A* reflects a failure on a more specific variant instead of the broader capability that the cluster represents. This example underscores a key limitation of relying solely on vector similarity for LLM weakness analysis.

To address this problem, we propose PROMPT2BOX, which embeds each prompt into a high-dimensional box embedding space. Unlike vector embeddings, box embeddings (Vilnis et al., 2018) can naturally represent asymmetric semantic relationships like entailment, making them well-suited for modeling hierarchical structure among prompts. Conceptually, each textual prompt is represented by a box parameterized by a center vector and a size vector. The center vector captures the semantic location of the prompt, such that semantically similar prompts are mapped to nearby centers. The size vector controls the semantic scope of the prompt: more general prompts are represented by larger boxes, while more specific prompts correspond to smaller boxes. This geometry allows entailment relationships to be expressed through box containment.

Specifically, if the box of one prompt is contained within the box of another prompt, we interpret this as an entailment relation, where the more specific prompt entails the more general one. For example, in Figure 1 (ii), box *A* is almost entirely contained within box *D*, reflecting that the prompt "*writing a romantic adventure story*" (prompt *A*) semantically entails the more general prompt "*writing a romantic story*" (prompt *D*).

The box of a prompt can also be interpreted as the space of its valid responses. In this interpretation, strong performance for a prompt implies the existence of a high-quality response in this solution space that can be produced by this LLM. For example, the LLM does worse on prompts *A* and *B* than for prompt *D*, which suggests that the LLM is not good at "*writing an adventure story about a robot*", especially when the adventure involves some *romantic* elements, but this LLM is probably good at writing other kinds of *romantic stories*. Our box representation demonstrates that the weakness of the LLM lies in the region of box *A* and *B*. Vectors, on the other hand, cannot support such conclusions because of their lack of specificity information.

To discover the entailment structure among prompts, we leverage existing entailment datasets and synthesize entailment relations between prompts. Next, we train an encoder to map every prompt into a box. Furthermore, we propose a new dimension reduction method and a new hierarchical clustering method for box embeddings. Our experiments show that our box embeddings predict the entailment relations much better than the vector baselines, which allows us to better analyze the weaknesses of LLMs through a 2D box embedding space and our specificity-aware hierarchical clustering method. Furthermore, even though the model is never trained to predict specificity of a prompt, we find that the box volume is a better specificity estimator than the length of the prompt. Figure 1 (iii)

[1]Notice that we do not consider the under-specified prompts as in Kim (2025); Zi et al. (2025).

shows that LLM scores are much more correlated with box volume than prompt lengths.

MAIN CONTRIBUTIONS

- We propose PROMPT2BOX, which uses a box embedding-based representation to capture the entailment relation among prompts. We also propose novel methods to synthesize entailment datasets for training an encoder that maps each LLM prompt to a box embedding.
- We propose BOX-SNE, a novel dimensionality-reduction method for box embeddings, and a new hierarchical clustering algorithm for box embeddings.
- We propose new evaluation metrics to assess similarity, entailment, and specificity in prompt representations. The experiments demonstrate large improvements of PROMPT2BOX over strong baselines and how box embeddings can be used to analyze LLMs and LLM evaluation benchmarks.

2 Related Work

Box embeddings (Vilnis et al., 2018), a form of region-based embeddings, have been shown to outperform other region-based representations such as Order Embeddings (Vendrov et al., 2016) and Poincaré Embeddings (Nickel and Kiela, 2017a) in modeling asymmetric relationships (Boratko et al., 2021b). Box embeddings have been successfully applied to model hierarchical and structured semantic relationships across multiple domains. In computer vision, Daroya et al. (2024) used box embeddings to represent task-level hierarchies. In the context of knowledge bases, box embeddings effectively capture hierarchical graph structures such as WordNet (Patel et al., 2020; Dasgupta et al., 2021) and OWL ontologies (Jackermeier et al., 2024). Furthermore, Ren et al. (2020); Dasgupta et al. (2021) introduce box-embedding-based formulations for knowledge graph query answering, where the logical structure of a query is directly encoded in the embedding space. As far as we know, no prior work uses box embeddings to analyze prompts or LLMs' weaknesses.

As identifying LLMs' weaknesses has become increasingly important, more and more benchmark/prompt analysis methods are being proposed. Examples include Clio (Tamkin et al., 2024), SkillVerse (Tian et al., 2025), and EvalTree (Zeng et al., 2025). Moreover, many recent studies leverage LLMs to discover taxonomy and categories from a corpus (Hsu et al., 2024; Tian et al., 2024; Zhang et al., 2025a; Kargupta et al., 2025; Zhong et al., 2025; Gao et al., 2025; Chirkova et al., 2025). Although different papers use different clustering methods or leverage LLMs in different ways, most of them conduct (hierarchical) clustering based on the vector embedding space. Our paper discovers that boxes perform better than vectors in terms of identifying LLM weakness clusters, and thus can potentially improve over the aforementioned related works.

3 Method

We first introduce our definition of entailment between prompts and establish the connection among constraint space, entailment, and solution space in Section 3.1. We describe the methods of optimizing our encoder in Section 3.2. Finally, Section 3.3 explains how the training data are curated. We expand on how we parameterize box embeddings and compute the intersection size and entailment probability in Appendix B.

3.1 Definition of Terms

Let $\mathcal{X}$ denote the space of instructions. In this paper, we focus on modeling both prompt similarity and prompt specificity, which we define as the number of constraints, both implicit and explicit, present in the instruction.

This notion of specificity closely mirrors the traditional logical concept of entailment. Recall that for any two statements c and d, if c entails d (written $c \models d$), then whenever c is true, d must also be true. In other words, c imposes a stronger condition than d; it is more specific. One can thus view specificity as a specialisation of entailment to the space $\mathcal{X}$.

For a pair of prompts $a, b \in \mathcal{X}$, if a contains more constraints than b, then any valid solution to a must satisfy a stricter set of conditions. Consequently, the set of valid solutions for a is smaller than that for b. This perspective offers an explanation for the empirically observed degradation in LLM performance as the number of constraints[2] increases (Jiang et al., 2024; Atmakuru et al., 2024). As constraints accumulate, the valid solution space contracts, increasing task difficulty by requiring the model to generate responses from a progressively smaller region of admissible outputs. Conse-

[2]Note that we only consider non-redundant constraints

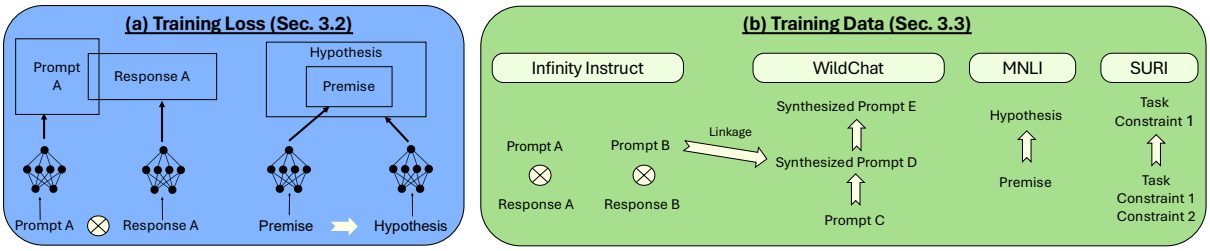


Figure 2: Illustration of our encoder training method. White $\Rightarrow$ means entailment and $\otimes$ means intersection. (a) An encoder is trained to take a prompt and output a box. Our loss function encourages its output box to overlap with the box of its corresponding response and to be contained by the box of the prompt it entails. (b) We use Infinity Instruct to encourage similar prompts to intersect with each other, and use WildChat, MNLI, and SURI to create positive and negative examples for learning entailment relationships between prompts.

quently, failures under highly constrained prompts expose systematic weaknesses in an LLM's ability to jointly satisfy multiple interacting constraints.

3.2 Optimization

SIMILARITY AND ENTAILMENT FOR BOXES: Let the box representations of a and b be defined with $\mathrm{Box}(a)$ and $\mathrm{Box}(b)$. We define the volume of $\mathrm{Box}(a)$ as $\mathrm{Vol}(a)$. We model *prompt similarity* as the volume of the intersection between their boxes, i.e., $\mathrm{VolInt(a, b)} = \mathrm{Vol}(\mathrm{Box}(a) \cap \mathrm{Box}(b))$.

Entailment score is defined as the conditional probability

$$p(b \mid a) := \frac{\mathrm{VolInt}(a, b)}{\mathrm{Vol}(\mathrm{Box}(a))} \tag{1}$$

By construction, $p(b \mid a) = 1$ when a fully entails b, and $p(b \mid a) < 1$ otherwise, providing a principled measure of prompt entailment.

LEARNABLE PARAMETERS. For each prompt a, the box embedding is parameterized by a center vector $a_{\text{center}} \in \mathbb{R}^D$ and a width vector $a_\delta \in \mathbb{R}^D_+$. These parameters are produced by passing the prompt embedding from a Sentence Transformer through two separate MLP heads. The Sentence Transformer and both MLPs are trained jointly.[3]

GUMBEL BOX FORMULATION: Optimizing objectives involving hard $\min$ and $\max$ operators is challenging due to their non-differentiability (Li et al., 2019; Dasgupta et al., 2020; Boratko et al., 2021a). We adopt the Gumbel Box formulation (Dasgupta et al., 2020), which replaces hard interval endpoints with Gumbel-distributed random variables and yields smooth, differentiable approximations to box intersection and containment. We present more details of the method in Appendix B.1.

CONTRASTIVE TRAINING OBJECTIVE. We train the model using contrastive learning objectives for both prompt similarity and entailment. Positive and negative prompt pairs are constructed for symmetric similarity and asymmetric entailment relations as shown in Figure 2. We provide training details in the Appendix C.

3.3 Data Curation

Finding sufficient entailment data to train a box encoder is a very challenging task, which limits the adoption of box representation. Fortunately, the high accuracy and flexibility of recent LLMs make synthesizing entailment data at large scale feasible. Figure 2 illustrates how we use synthesized and existing entailment datasets to train our box encoder and we will describe the curation steps for each dataset below.

SEMANTIC RELEVANCE: To capture relevance, we gather instruction response pairs from Infinity Instruct (Li et al., 2025) and retain only the English samples. Our contrastive learning encourages similar prompts to overlap with each other by maximizing the intersection (i.e., $\mathrm{VolInt(a, b)}$) between the box of a prompt and the box of its response, while penalizing the other negatively sampled intersections (Mikolov et al., 2013).

ENTAILMENT DATA FROM MULTINLI: We leverage MultiNLI (Williams et al., 2018), which contains sentence pairs labeled as entailment, contradiction, or neutral. We apply a preprocessing step to transform these pairs into triplets of (an-

[3]MLP leads to a 2% increase in the number of parameters over the baselines.

chor, positive, negative) for contrastive learning. For each anchor sentence, the entailed hypothesis serves as the positive example, while hypotheses labeled as neutral or contradiction are used as negative examples, as they do not express an entailment relationship. This dataset allows the model to learn sentence-level entailment relationships between text pairs.

SYNTHESIZED HIERARCHY FROM WILDCHAT: To learn the entailment relation among instructions, we synthesize hierarchical instructions on WildChat (Zhao et al., 2024). Specifically, we prompt OpenAI's GPT-4.1 to iteratively rewrite each instruction in WildChat into progressively more general forms, producing a hierarchy spanning multiple levels of specificity. We obtain 20K hierarchical instruction groups, each containing 4-10 levels.

SIBLING RELATIONSHIPS FROM SURI: While the previous datasets teach direct parent-child relationships, they do not capture sibling relationships, cases where two instructions share a common parent but differ in their specific constraints, and thus do not entail one another. To address this, we leverage SURI (Pham et al., 2024). Each datapoint in SURI consists of a main goal summarizing the original text, accompanied by approximately ten constraints covering stylistic and semantic elements. We construct instruction trees by combining the main goal with various subsets of constraints. Instructions sharing the same parent but with different constraint combinations are treated as sibling nodes and used as hard negatives in our contrastive learning objective (Rozonoyer et al., 2024). This teaches the model to distinguish between related but non-entailing instructions.

DATASET ENTAILMENT LINKAGE: After initially training with the above datasets, we observed that while the model performed well on entailment-based metrics, it exhibited a noticeable drop in semantic relevance. We hypothesize that the objectives associated with the different datasets are learnt separately, leading to a disconnect in the embedding space. To mitigate this issue, we explicitly connect Infinity Instruct to our synthesized hierarchical dataset using WildChat (Zhao et al., 2024). For each sampled query prompt from Infinity Instruct, we use MPNet-Base (Song et al., 2020) model to retrieve similar prompts from WildChat. We then ask GPT-4.1 to find the most specific synthesized prompt entailed by the query prompt. The aim of this linkage is to force the model to learn a shared representation across the different objectives.

4 Embedding Space Evaluation

We initialize both the box-embedding model and the vector-based baseline from MPNet-base (Song et al., 2020). Because vector embeddings cannot naturally represent entailment or partial orders, we treat all entailment relations as similarity for the default vector baseline. Concretely, instruction pairs that exhibit entailment are encouraged to have high cosine similarity, without imposing any directional or containment structure.

In contrast, the box model explicitly separates these notions: similarity is modeled via Eq. (12), while entailment is captured through the containment-based objective in Eq. (13).

Our training set comprises 203,138 samples drawn from the different sources with the following distribution: prompt-response pairs from Infinity Instruct (50K), SURI-based entailment dataset (50K), Synthetic hierarchical instructions from WildChat (50K), MNLI triplets (50K), and the linkage dataset (3,138).

For ablation, we additionally train box models without the linkage dataset (w/o links), as well as both box and vector models without the entailment datasets (w/o entails) (i.e., trained only on pairs from Infinity Instruct). To evaluate the effect of synthetic data, we also include a model trained without linkage dataset and hierarchical instructions from WildChat (w/o synth).

To compare against other embedding methods capable of modeling asymmetry, we include a model trained using the CSDelta metric (Chang et al., 2018) as well as a hyperbolic embedding model based on squared Lorentzian distance (Law et al., 2019). Further implementation and training details are provided in the Appendix C.

4.1 Evaluation Metrics

We evaluate the training process using two complementary metrics: a semantic similarity metric derived from STS-B (Cer et al., 2017) and an entailment metric computed on a held-out subset of the SURI dataset. Full results are reported in the appendix.

RETRIEVAL WITH FOLLOWBENCH: To evaluate a model's ability to retrieve more specific yet relevant instructions, we leverage FollowBench (Jiang

et al., 2024), which consists of instruction groups organized by increasing constraint levels. Given a query at level $\mathcal{L}$, the model is tasked to retrieve another query from the same semantic group but $\mathcal{L}' > \mathcal{L}$. Success requires the model to correctly prompts that are both semantically similar and also more specific. We evaluate on 688 queries.

SCORE PREDICTION ON ULTRAFEEDBACK: A good embedding space should put the prompts that induce similar response scores close to each other, which allows us to run a kNN (k nearest neighbor) regressor on the embedding space to predict the response scores of unseen prompts. We compare the regressor performance using different embedding spaces on the UltraFeedback (Cui et al., 2024) dataset, which contains instructions paired with responses from 17 LLMs and associated quality scores. We split each set of scores and responses into a 70/30 retrieval–test corpus split.

For each test instruction, we retrieve the top-5 corpus examples and predict the response score of the test prompt by averaging the scores from the training corpus, reporting root mean squared error (RMSE) against the gold score.

4.2 Results

RETRIEVAL FOLLOWBENCH: From Table 1 we see that in general, the box variants perform better at retrieving children prompts than all the other embedding models, which means box embeddings are more effective at modeling entailment. **Box** is better than **Box w/o synth**. Furthermore, **Box** wins over **Box w/o entail** and **Box w/o entail** is better than **Vector w/o entail**. This suggests that the gains in entailment performance stem from two complementary factors: (1) the synthesized entailment training data, and (2) the representational capacity and inductive bias of box embeddings.

SCORE PREDICTION METRICS: In Table 1 the lowest RMSE comes from the box based models, which suggests the specificity encoded by box volume correlates with the difficulty of the prompts. **Hyperbolic** and **CSDelta** achieve similar results when compared with the **Vector** baseline.

These gains are achieved with essentially the same setup as the baselines, aside from a minor $\sim$2% increase in parameters, which is insufficient to explain the performance gap, suggesting that the improvements stem from the representational advantages of box embeddings.

Although the **Box w/o links** achieves the strongest performances in Table 1 among the box variants, Table 4 in Appendix D shows that this comes at the cost of reduced semantic similarity performance. By incorporating link data, **Box** achieves a better balance between entailment modeling and semantic similarity, which not only is important for visualization in Section 5 but also improves downstream applications such as hierarchical clustering in Section 6 and specificity estimation in Section 7.

Model	FollowBench	UltraFeedback	
	Accuracy↑	Avg. RMSE↓	Improv. (%)↑
Random	—	1.8293	0.0
Hyperbolic	0.659	1.6260	11.11
CSDelta	0.012	1.5955	12.80
Vector	0.640	1.6059	12.21
Vector w/o entail	0.627	1.5605	14.71
Box w/o entail	0.687	1.5610	14.67
Box w/o links	**0.775**	**1.4777**	**19.22**
Box w/o synth	0.716	1.5280	16.47
Box	0.738	1.5280	16.47

Table 1: FollowBench accuracy and UltraFeedback score prediction performance (Avg. RMSE and its improvement over Random across 17 LLMs). Best results are in bold; second-best in blue.

5 Box Visualization and Analysis

We train high-dimensional box embeddings to capture complex entailment structure among prompts. However, to analyze LLM weaknesses visually, like in Figure 1, we need to reduce the embeddings to low-dimensional embedding space. Since no existing dimensionality reduction method is designed for box embeddings, we propose BOX-SNE (see Appendix E for details). In this section, we present two applications demonstrating how our box embeddings can be used to analyze prompts and find LLM weaknesses. The interactive visualization can be viewed at `https://zawedcvg.github.io/P2B/visualisation.html`.

5.1 Comparing Different Datasets

We first visualize 150 random examples from each of three datasets: WildChat, UltraFeedback, and WildBench (Lin et al., 2026). WildBench is a curated subset of WildChat, constructed to contain more challenging and more specific prompts. From Figure 3, the box-based visualization on the right makes this distinction explicit; WildBench examples are consistently represented by smaller boxes, indicating higher specificity, compared to those from WildChat and UltraFeedback. In contrast, the vector-based visualization baseline on the left fails

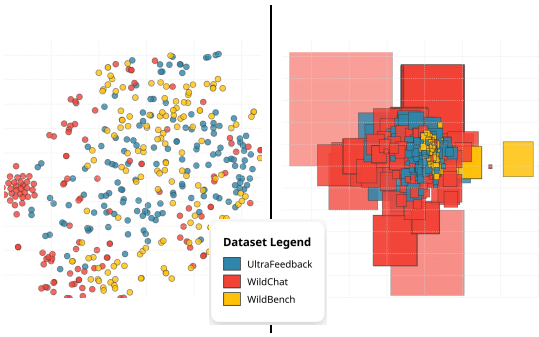


Figure 3: Comparison between our box-based visualization (right) and a t-SNE visualization of the vector baseline (left).

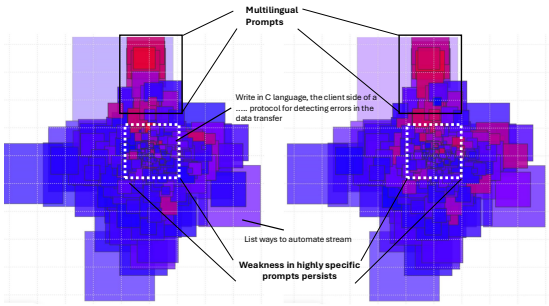


Figure 4: Comparison of LLMs' performance in 2D box embedding space. LLM performs better in the blue regions than in the red regions.

to clearly distinguish the datasets, making such insights difficult to discern.

5.2 Comparing Model Performance

Next, we compare visualizations for LLaMA-2-7B and LLaMA-2-70B to examine how box-based representations reveal both the benefits and limits of scaling. As shown in Figure 4, increasing model size reduces low-scoring (red) regions and improves performance across much of the space, as scaling laws (Kaplan et al., 2020) suggests. However, the upper region, corresponding mainly to multilingual prompts (black box), remains dominated by low scores. Interestingly, within the largely high-performing (blue) regions of the larger model, we observe small, dispersed red boxes (green outline) indicating failures on highly specific prompts. Some of these failure regions exist in the smaller model. This indicates that scaling does not uniformly eliminate fine-grained weaknesses: certain specific prompts remain challenging.

6 Hierarchical Clustering

Hierarchical clustering of prompts enables evaluators to identify LLM weaknesses at multiple levels of abstraction, from broad skill categories to specific sub-skills. Section 4 shows that box embedding is better at putting the prompts with entailment relationship or with similar score close to each other. Next, we investigate if this ability translates to creating better clusters for discovering LLMs' weaknesses. Directly evaluating such clusters with human experts is difficult due to the cost, required expertise, and the need to maintain a consistent evaluation setup across many models and clusters. A comprehensive human evaluation of this kind is therefore beyond the scope of this paper. We instead use a set of well-defined proxy metrics to assess how effectively different clustering methods isolate regions of model weakness.

To group the smaller boxes that have an entailment relation at the lower levels, we propose a new hierarchical clustering method that always merges the two boxes with the smallest resulting box first (see Appendix F for details) and compare it with the classic Ward linkage (Ward Jr, 1963) on the vector space.

Since there is no existing ground-truth prompt hierarchical structure, we compare box-based and vector hierarchical clusters using the following three metrics: (i) consistency of model scores within local neighborhoods, (ii) correlation between depth and instruction specificity, and (iii) ability to discover model weakness clusters.

6.1 Local Score Consistency

A well-formed hierarchical clustering should place semantically similar prompts under the same parent node. This is desirable because similar prompts tend to elicit similar LLM responses, leading to similar scores. Thus, if semantically dissimilar prompts are grouped together, their scores may vary widely, rendering the cluster average unrepresentative and obscuring true weaknesses (e.g., the *A–D* cluster in Figure 1).

We measure this property by computing the average absolute score difference between neighboring leaf nodes, where neighbors are defined as prompts sharing the same immediate parent in the hierarchy.

As a random baseline, we assign each prompt a randomly sampled neighbor from the set of 500 prompts and compute the score differences. The results in Table 2 show that box achieves a 38%

Metric	Rand.	Vec.	Box w/o links	Box w/o synth	Box
Loc. Score Cons.	0.0%	9.32%	**13.24%**	10.34%	12.88%
Spec.–Depth Agr.	50.00%	52.71%	67.07%	66.03%	**68.34%**
All Size Weakness	–	0.0%	2.85%	5.94%	**7.70%**
AUC	–	0.0%	7.81%	10.95%	**13.47%**

Table 2: Comparison across weakness discovery, LLM agreement, and score improvement metrics.

relative improvement compared with the vector baseline (i.e., $\frac{12.88\%-9.32\%}{9.32\%}$).

6.2 Specificity Ordering Accuracy

Next, we evaluate how well hierarchical depth aligns with instruction specificity using LLM-as-a-judge. We select 500 instructions with available LLaMA-2-13B-Chat responses to limit LLM inference cost. Because direct comparisons between arbitrary instructions are ambiguous, we only select pairs of similar prompts. Exact evaluation details are provided in Appendix F.1.

Table 2 shows that vector-based hierarchies perform almost the same as the random baseline, while box-based hierarchies achieve over 68% specificity accuracy, a 33% relative improvement over both baselines, demonstrating that box-induced hierarchies effectively capture instruction specificity. We also include a human study in the Appendix F.2 to validate the LLM-as-a-judge metric and confirm our conclusion.

6.3 Cluster Weakness Containment

Finally, we evaluate how well the clustering algorithm can identify and isolate "model weaknesses". We define a weakness as an instruction cluster for which the model's average score lies at or below the 25th percentile. The underlying assumption is that a good clustering algorithm will be able to accurately group the low-score prompts into a large cluster. Mixing unrelated instructions will "dilute" the weaknesses and thus lead to an inflation of average scores.

To detect this effect, we compute the number of weakness clusters as

$$\#W_{t_s}^{R_{25\%}} = |\{w \mid S(w) \leq R_{25\%} \text{ and } |w| \geq t_s\}|\,,$$

where $|w|$ denotes the size of cluster w, t_s is the minimum cluster size threshold, $S(w)$ is the average score of cluster w, and $R_{25\%}$ is the closest integer score below the 25th percentile.

We report $\#W_2^{R_{25\%}}$ (i.e., All Size Weakness) in Table 2, and the aggregate measure $\sum_{t_s>1} \#W_{t_s}^{R_{25\%}}$, which corresponds to the area under the curve (AUC) of weakness cluster counts across different size thresholds, in Table 8 and Figures 5 to 7. The overall percentage improvements are summarized in Table 2.

Table 2 shows that hierarchical clustering based on box embeddings identifies a greater number of weakness clusters. When considering the AUC across all cluster sizes, the improvement increases to 13.47%, indicating that box embeddings can capture more and larger weakness clusters.

	UltraFeedback		LMSYS	
System	RMSE Impr.	Line Cov. Impr.	RMSE Impr.	Line Cov. Impr.
Length	0.0%	0%	0.0%	0%
Box w/o links	44.3%	83%	14.8%	10%
Box w/o synth	39.8%	113%	9.4%	**72%**
Box	**45.3%**	**113.5%**	**18.3%**	71%

Table 3: Ablation results across 16 LLMs on UltraFeedback and 16 LLMs on LMSYS. The numbers of boxes are relative improvement to the length baseline.

7 Specificity Estimation

As the prompt becomes more specific, the average score of LLMs' responses should decrease. We measure how well the learned box volume models specificity. We use prompt length as a baseline as it has been shown as a strong baseline in modeling specificity (Li and Nenkova, 2015; Gao et al., 2019; Kapur et al., 2026). We thus plot the scores in UltraFeedback and LMSYS-Chat-1M (See Appendix G for more details) against log(length) in Figure 8 and against log(box volume) in Figure 9. To estimate the average score in the curve, we use kernel density estimation (KDE), whose bandwidth is decided by using Silverman's rule-of-thumb (Silverman, 2018). In these figures, the LLMs' scores often saturate for very specific or very general prompts, so we use the sigmoid function to fit the curves. Figure 9 shows various saturation patterns from different LLMs.

A better specificity estimation should have a smaller RMSE to the fitted sigmoid curve and a longer linear region (i.e., line coverage) of the sigmoid function, which demonstrate its ability of modeling the subtle difficulty differences. Appendix G describes how the line coverage is measured. Table 3 shows that **Box** achieves the highest improvement over the strong length baseline in UltraFeedback and LMSYS-Chat-1M (Zheng et al., 2023).

8 Conclusion

We introduce PROMPT2BOX, a method for embedding prompts into a box embedding space that jointly captures both semantic similarity and specificity. Through extensive experiments, we demonstrate that PROMPT2BOX consistently outperforms vector-based baselines across multiple metrics for modeling entailment and predicting the scores of neighboring prompts.We provide intuitive 2D visualizations that illustrate how box embeddings naturally encode the hierarchical structure of prompt specificity through geometric containment, offering insights that vector-based visualizations cannot capture. Furthermore, we validate the practical utility of our approach on the hierarchical clustering, where PROMPT2BOX achieves superior performance across four distinct evaluation metrics. Finally, measuring prompt specificity using box volume creates a new analysis dimension for LLMs' behavior (e.g., we discover that the scores of 16 different LLMs versus specificity forms different sigmoid functions in UltraFeedback).

9 Limitations

Due to the limited space, we only test 3 applications of box embedding. Understanding LLMs' ability and modeling specificity have many other potential applications. For example, besides comparing datasets or LLMs with different sizes, we can also compare LLM judges, or LLMs with different training stages or hyperparameters. Furthermore, our prompt specificity/difficulty estimation might help LLM routing (Guha et al., 2024; Kashani et al., 2025), evaluation data selection (Zouhar et al., 2025) and creativity evaluation (Atmakuru et al., 2024; Lu et al., 2025), prompt safety analysis (Ayub and Majumdar, 2024), response specificity estimation (Jiang et al., 2025), LLM interpretability (Shani et al., 2025), and knowledge editing (Ge et al., 2024). We leave them as our future work.

Many of our applications experiments rely on UltraFeedback, which is a very effective dataset for LLM aligment (Ivison et al., 2024) because many modern benchmarks only focus on difficult prompts and it is hard to find the benchmarks containing prompts with various specificity and lots of LLM responses and scores.

To ensure our methods work well in general domains, we do not train our methods using UltraFeedback. Nevertheless, the effectiveness of our approach still depends on the availability of existing entailment datasets and LLMs' ability to synthesize the training data, so PROMPT2BOX might not be as effective in some special domains or languages with fewer resources.

We observe that the 2D boxes created using BOX-SNE tend to suffer from overcrowding issues analogous to those encountered in early vector visualization methods. Developing normalization or regularization techniques to encourage greater spatial dispersion remains an open challenge; notably, standard approaches like t-SNE do not directly apply here, as box volumes are represented in log-space. Addressing this limitation could further improve interpretability.

Acknowledgement

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the National Science Foundation (NSF) grant numbers IIS-2106391. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor. The computational resources for this work are provided by the Unity Research Computing Platform, a multi-institutional cluster lead by University of Massachusetts Amherst, the University of Rhode Island, and University of Massachusetts Dartmouth.

References

Anirudh Atmakuru, Jatin Nainani, Rohith Siddhartha Reddy Bheemreddy, Anirudh Lakkaraju, Zonghai Yao, Hamed Zamani, and Haw-Shiuan Chang. 2024. Cs4: Measuring the creativity of large language models automatically by controlling the number of story-writing constraints. *arXiv preprint arXiv:2410.04197*.

Md Ahsan Ayub and Subhabrata Majumdar. 2024. Embedding-based classifiers can detect prompt injection attacks. *arXiv preprint arXiv:2410.22284*.

Michael Boratko, Javier Burroni, Shib Sankar Dasgupta, and Andrew McCallum. 2021a. Min/max stability and box distributions. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 2146–2155. PMLR.

Michael Boratko, Dongxu Zhang, Nicholas Monath, Luke Vilnis, Kenneth L Clarkson, and Andrew McCallum. 2021b. Capacity and bias of learned geometric embeddings for directed graphs. In *Advances in Neural Information Processing Systems*, volume 34, pages 16423–16436. Curran Associates, Inc.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Haw-Shiuan Chang, Ziyun Wang, Luke Vilnis, and Andrew McCallum. 2018. Distributional inclusion vector embedding for unsupervised hypernymy detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 485–495, New Orleans, Louisiana. Association for Computational Linguistics.

Nadezhda Chirkova, Tunde Oluwaseyi Ajayi, Seth Aycock, Zain Muhammad Mujahid, Vladana Perlić, Ekaterina Borisova, and Markarit Vartampetian. 2025. Llm-as-a-qualitative-judge: automating error analysis in natural language generation. *arXiv preprint arXiv:2506.09147*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: Boosting language models with scaled ai feedback. *Preprint*, arXiv:2310.01377.

Rangel Daroya, Aaron Sun, and Subhransu Maji. 2024. Task2box: Box embeddings for modeling asymmetric task relationships. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28827–28837.

Shib Sankar Dasgupta, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Lorraine Li, and Andrew McCallum. 2020. Improving local identifiability in probabilistic box embeddings. In *Advances in Neural Information Processing Systems*.

Shib Sankar Dasgupta, Xiang Lorraine Li, Michael Boratko, Dongxu Zhang, and Andrew McCallum. 2021. Box-to-box transformations for modeling joint hierarchies. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 277–288, Online. Association for Computational Linguistics.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 316–321, Online. Association for Computational Linguistics.

Muhan Gao, Jash Shah, Weiqi Wang, Kuan-Hao Huang, and Daniel Khashabi. 2025. Science hierarchography: Hierarchical organization of science literature. *arXiv preprint arXiv:2504.13834*.

Yifan Gao, Yang Zhong, Daniel Preoţiuc-Pietro, and Junyi Jessy Li. 2019. Predicting and analyzing language specificity in social media posts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6415–6422.

Huaizhi Ge, Frank Rudzicz, and Zining Zhu. 2024. How well can knowledge edit methods edit perplexing knowledge? *arXiv preprint arXiv:2406.17253*.

Neel Guha, Mayee Chen, Trevor Chow, Ishan Khare, and Christopher Re. 2024. Smoothie: Label free language model routing. *Advances in Neural Information Processing Systems*, 37:127645–127672.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *Preprint*, arXiv:1705.00652.

Chao-Chun Hsu, Erin Bransom, Jenna Sparks, Bailey Kuehl, Chenhao Tan, David Wadden, Lucy Lu Wang, and Aakanksha Naik. 2024. Chime: Llm-assisted hierarchical organization of scientific studies for literature review support. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 118–132.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Advances in neural information processing systems*, 37:36602–36633.

Mathias Jackermeier, Jiaoyan Chen, and Ian Horrocks. 2024. Dual box embeddings for the description logic el++. *Preprint*, arXiv:2301.11118.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. 2025. How many instructions can llms follow at once? In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.

Zhengping Jiang, Anqi Liu, and Benjamin Van Durme. 2025. Conformal linguistic calibration: Trading-off between factuality and specificity. *arXiv preprint arXiv:2502.19110*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Rhea Kapur, Robert Hawkins, and Elisa Kreiss. 2026. When more words say less: Decoupling length and specificity in image description evaluation. *arXiv preprint arXiv:2601.04609*.

Priyanka Kargupta, Nan Zhang, Yunyi Zhang, Rui Zhang, Prasenjit Mitra, and Jiawei Han. 2025. Taxoadapt: Aligning llm-based multidimensional taxonomy construction to evolving research corpora. *arXiv preprint arXiv:2506.10737*.

Idan Kashani, Avi Mendelson, and Yaniv Nemcovsky. 2025. Representing llms in prompt semantic task space. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8578–8597.

Olivia Kim. 2025. Detail matters: Measuring the impact of prompt specificity on reasoning in large language models. *arXiv preprint arXiv:2512.02246*.

Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. 2019. Lorentzian distance learning for hyperbolic representations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3672–3681. PMLR.

Jijie Li, Li Du, Hanyu Zhao, Bo wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. 2025. Infinity instruct: Scaling instruction selection and synthesis to enhance language models. *Preprint*, arXiv:2506.11116.

Junyi Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. 2019. Smoothing the geometry of probabilistic box embeddings. *ICLR*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2026. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In *The Thirteenth International Conference on Learning Representations*.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. 2025. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*.

Yining Lu, Dixuan Wang, Tianjian Li, Dongwei Jiang, Sanjeev Khudanpur, Meng Jiang, and Daniel Khashabi. 2025. Benchmarking language model creativity: A case study on code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2776–2794.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Maximilian Nickel and Douwe Kiela. 2017a. Poincaré embeddings for learning hierarchical representations. In *Neural Information Processing Systems*.

Maximilian Nickel and Douwe Kiela. 2017b. Poincaré embeddings for learning hierarchical representations. In *Neural Information Processing Systems*.

Dhruvesh Patel, Shib Sankar Dasgupta, Michael Boratko, Xiang Li, Luke Vilnis, and Andrew McCallum. 2020. Representing joint hierarchies with box embeddings. In *Automated Knowledge Base Construction*.

Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1722–1753, Miami, Florida, USA. Association for Computational Linguistics.

Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *ICLR*.

Benjamin Rozonoyer, Michael Boratko, Dhruvesh Patel, Wenlong Zhao, Shib Dasgupta, Hung Le, and Andrew McCallum. 2024. Learning representations for hierarchies with minimal support. volume 37, pages 5586–5614.

Jeanne Sebaugh and P. McCray. 2003. Defining the linear portion of a sigmoid-shaped curve: Bend points. *Pharmaceutical Statistics - PHARM STAT*, 2:167–174.

Chen Shani, Liron Soffer, Dan Jurafsky, Yann LeCun, and Ravid Shwartz-Ziv. 2025. From tokens to thoughts: How llms and humans trade compression for meaning. *arXiv preprint arXiv:2505.17117*.

Bernard W Silverman. 2018. *Density estimation for statistics and data analysis*. Routledge.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, and 1 others. 2024. Clio: Privacy-preserving insights into real-world ai use. *arXiv preprint arXiv:2412.13678*.

Chang Tian, Matthew B Blaschko, Wenpeng Yin, Mingzhe Xing, Yinliang Yue, and Marie Francine Moens. 2024. A generic method for fine-grained category discovery in natural language texts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3548–3566.

Yufei Tian, Jiao Sun, Nanyun Peng, and Zizhao Zhang. 2025. Skillverse: Assessing and enhancing llms with tree evaluation. *arXiv preprint arXiv:2506.00319*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *International Conference on Learning Representations*.

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. 2018. Probabilistic embedding of knowledge graphs with box lattice measures. In *Association for Computational Linguistics*.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, and 1 others. 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. 2025. Evaltree: Profiling language model weaknesses via hierarchical capability trees. *arXiv preprint arXiv:2503.08893*.

Haiqi Zhang, Zhengyuan Zhu, Zeyu Zhang, and Chengkai Li. 2025a. Llmtaxo: Leveraging large language models for constructing taxonomy of factual claims from social media. *arXiv preprint arXiv:2504.12325*.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, and 1 others. 2025b. Cfbench: A comprehensive constraints-following benchmark for llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32926–32944.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2023. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *Preprint*, arXiv:2309.11998.

Mian Zhong, Pristina Wang, and Anjalie Field. 2025. Hicode: Hierarchical inductive coding with llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31048–31066.

Yangtian Zi, Harshitha Menon, and Arjun Guha. 2025. More than a score: Probing the impact of prompt specificity on llm code generation. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2380–2402.

Vilém Zouhar, Peng Cui, and Mrinmaya Sachan. 2025. How to select datapoints for efficient human evaluation of nlg models? *Transactions of the Association for Computational Linguistics*, 13:1789–1811.

A Expansion of Definition of Terms

Let $\mathcal{X}$ denote the space of instructions. Let $\mathcal{U}$ denote the universe of all possible constraints and we assume the number of possible constraints $|\mathcal{U}|$ is finite. For any instruction $x \in \mathcal{X}$, let $\mathcal{C} : \mathcal{X} \to 2^{\mathcal{U}}$ be a mapping from an instruction to the set of constraints it induces, where

$$\mathcal{C}(x) = \{\, c \in \mathcal{U} \mid c \text{ holds for all valid responses to } x \,\}. \tag{2}$$

Recall that for any two statements g and h, if g entails h (written $g \models h$), then whenever g is true, h must also be true. In other words, g imposes a stronger condition than h. When g and h are both prompts, $g \models h$ means that any solution satisfying g also satisfies h. Formally, constraint inclusion is the same as entailment between the two prompts:

$$\forall\, a, b \in \mathcal{X}, \qquad \mathcal{C}(a) \supseteq \mathcal{C}(b) \iff a \models b. \tag{3}$$

Furthermore, we say the prompt a is more specific than prompt b if the prompt a has more constraints (i.e., $C(a) \supset \mathcal{C}(b)$). Taking the example present in Figure 1, we have two prompts: $g =$ "*Please write a short story about an adventure of a robot*" and $h =$ "*Please write a short adventure story.*" Since it is not possible to exhaustively list out the set of all the possible constraints, one can intuitively say that $\mathcal{C}(h) \subset \mathcal{C}(g)$ if g can be written as h with additional constraint(s). We can thus write g as "*Please write a short adventure story; Make the story about a robot.*" We see that $\mathcal{C}(g) \supset \mathcal{C}(h)$, thus based on the previous definition: $g \models h$.

Let $\mathcal{Y}$ denote the universe of all possible responses. For any instruction $x \in \mathcal{X}$, let $\mathcal{S} : \mathcal{X} \to 2^{\mathcal{Y}}$ be a mapping from a prompt to the set of valid responses that satisfy the constraints imposed by x, where $\mathcal{S}(x) \subseteq \mathcal{Y}$.

Let's assume we have a pair of prompts $a, b \in \mathcal{X}$ and a contains more constraints than b. Since any valid solution to a must satisfy the stricter set of constraints in a, they must also satisfy all the constraints for b. Further, the set of valid solutions for a is smaller than that for b. Consequently, inclusion in the constraint space induces reverse inclusion in the solution space:

$$\mathcal{C}(a) \supseteq \mathcal{C}(b) \quad \Longrightarrow \quad \mathcal{S}(a) \subseteq \mathcal{S}(b). \tag{4}$$

We can prove the other direction of this by:

Assume $S(a) \subseteq S(b)$. Let $c \in C(b)$, so c is satisfied by all elements of $S(b)$. Since every element of $S(a)$ also belongs to $S(b)$, c is satisfied by all elements of $S(a)$ as well, hence $c \in C(a)$. Thus $C(b) \subseteq C(a)$.

So we get

$$\mathcal{C}(a) \supseteq \mathcal{C}(b) \quad \Longleftrightarrow \quad \mathcal{S}(a) \subseteq \mathcal{S}(b). \tag{5}$$

As constraints accumulate, the valid solution space contracts, increasing task difficulty by requiring the model to generate responses from a progressively smaller region of admissible outputs. This perspective offers an explanation for the empirically observed LLM performance degradation as the number of constraints increases (Jiang et al., 2024; Tamkin et al., 2024; Zeng et al., 2025; Tian et al., 2025).

A.1 Proof that $g \models h \quad \Longrightarrow \quad C(h) \subseteq C(g)$:

If g and h are a paraphrase, $C(h) = C(g)$. Otherwise, we can prove it by viewing it from the solution space.

Since g can be decomposed as $g \equiv h \cup k$ (i.e., g is exactly h with the additional constraint k = "Make the story about a robot"), any response satisfying g must simultaneously satisfy both h and k. Therefore:

$$S(g) = \{s \mid s \in Y \text{ satisfies } h \text{ and } s \in Y \text{ satisfies } k\} = S(h) \cap S(k). \tag{6}$$

Since $S(g) = S(h) \cap S(k) \subseteq S(h)$, we have:

$$S(g) \subseteq S(h). \tag{7}$$

By Equation (5), a smaller solution space corresponds to a larger constraint set — formally, for any $a, b \in \mathcal{X}$:

$$S(a) \subseteq S(b) \implies C(b) \subseteq C(a). \tag{8}$$

Applying this with $a = g$ and $b = h$, it follows that:

$$C(h) \subseteq C(g). \tag{9}$$

B Prompt Representation

Given the importance of specificity, an effective representation must capture both relevance and specificity between prompts. Traditional vector embeddings represent each prompt as a point in a metric space and model relationships solely through symmetric distance functions, making them ill-suited for expressing asymmetric relations such as specificity or constraint inclusion.

Formally, let $f : X \to \mathbb{R}^D$ denote a vector embedding function. Similarity between two prompts a and b is modeled using a distance function $d(f(a), f(b))$, which is invariant to direction and therefore cannot encode partial order relations of the form $\mathcal{C}(a) \supset \mathcal{C}(b)$.

In contrast, box embeddings represent each prompt $a \in X$ as an axis-aligned hyper-rectangle in $\mathbb{R}^D$. Formally, for a prompt a, we parameterize its box embedding using a center vector $a_{\text{center}} \in \mathbb{R}^D$ and a width vector $a_\delta \in \mathbb{R}^D_+$. The lower and upper corners of the box are given by

$$a^{\llcorner} := a_{\text{center}} - a_\delta, \qquad a^{\urcorner} := a_{\text{center}} + a_\delta. \tag{10}$$

The box embedding for prompt a is therefore defined as the cartesian product of each side of the rectangle:

$$\text{Box}(a) := \prod_{d=1}^{D} [a_d^{\llcorner}, a_d^{\urcorner}] = [a_1^{\llcorner}, a_1^{\urcorner}] \times \ldots \times [a_D^{\llcorner}, a_D^{\urcorner}]. \tag{11}$$

Let us consider two prompts $a, b \in X$, with corresponding box representations $\text{Box}(a)$ and $\text{Box}(b)$. We define the volume of $\text{Box}(a)$ as $\text{Vol}(a) := \prod_{d=1}^{D}(a_d^{\urcorner} - a_d^{\llcorner})$. We first model *prompt similarity* as the volume of the intersection between their boxes, i.e., $\text{Vol}(\text{Box}(a) \cap \text{Box}(b))$. Since the intersection of two intervals is determined by the minimum of their upper bounds and the maximum of their lower bounds, the intersection volume is given by

$$\text{VolInt(a, b)} := \prod_{d=1}^{D} \max\left(\min(a_d^{\urcorner}, b_d^{\urcorner}) - \max(a_d^{\llcorner}, b_d^{\llcorner}), 0\right) \tag{12}$$

However, similarity alone is insufficient for our purposes. A key motivation for using box embeddings is their ability to model *asymmetric interactions* between prompts. In particular, when prompt a entails prompt b, we expect $\text{Box}(b)$ to contain $\text{Box}(a)$. In this case, the intersection volume equals the volume of $\text{Box}(a)$, i.e., $\text{VolInt}(a, b) = \text{Vol}(\text{Box}(a))$.

We therefore define an *entailment score* as the conditional probability

$$p(b \mid a) := \frac{\text{VolInt}(a, b)}{\text{Vol}(\text{Box}(a))} \tag{13}$$

By construction, $p(b \mid a) = 1$ when a fully entails b, and $p(b \mid a) < 1$ otherwise, providing a principled measure of asymmetric prompt entailment.

B.1 Gumbel Box Specifics

Hard $\min$ and $\max$ operations are replaced with temperature-controlled log-sum-exp (LSE) operators. For one-dimensional intervals, the expected intersection length is approximated as

$$\text{LSE}_\beta\left(\text{LSE}_{-\beta}(x_1^{\urcorner}, \ldots, x_N^{\urcorner}) - \text{LSE}_\beta(x_1^{\llcorner}, \ldots, x_N^{\llcorner}), 0\right),$$

where $\text{LSE}_\beta(\mathbf{x}) := \beta \log \sum_i \exp(x_i/\beta)$. In higher dimensions, the expected intersection volume is computed as a product across dimensions. We use this smooth approximation to replace hard volume-based quantities in the containment and similarity scores. Following (Dasgupta et al., 2020), we use separate temperature parameters for volume and intersection computations. In all our experiments, we fix these temperatures to $\beta_{\text{vol}} = \langle 1.0 \rangle$ and $\beta_{\text{int}} = \langle 0.001 \rangle$.

C Training Details

We train all models using the Multiple Negatives Ranking Loss (Henderson et al., 2017), combined with GradCache (Gao et al., 2021) to enable substantially larger effective batch sizes during training. Each mini-batch contains samples from only a single dataset. Datasets are sampled proportionally to their relative sizes, while batches are processed using a round-robin scheduling strategy.

We use a learning rate of 2×10^{-5} with a warmup ratio of 0.1. Training is performed primarily on NVIDIA 2080 Ti GPUs. For box embeddings, we set the volume temperature to 1.0 and the intersection temperature to 0.001.

C.1 CSDelta

For the model trained using the CSDelta metric, entailment from a to b is defined as cosine similarity scaled by the difference in vector magnitudes:

$$p(b \mid a) = \frac{\mathbf{w}_a^\top \mathbf{w}_b}{\|\mathbf{w}_a\|_2 \, \|\mathbf{w}_b\|_2} \cdot \left(\|\mathbf{w}_a\|_1 - \|\mathbf{w}_b\|_1\right), \tag{14}$$

where $\mathbf{w}_a$ is the vector representation of a and $\mathbf{w}_b$ is the representation of b. Semantic similarity is modeled using cosine similarity. The purpose of this metric is to see if vector norms can encode model entailment.

C.2 Hyperbolic

For the hyperbolic model, we employ two distinct training objectives depending on the relation type. Let $\mathbf{u}, \mathbf{v} \in \mathbb{H}^d$ denote the encodings of prompts a and b in hyperbolic space, and let $\| \cdot \|_L$ denote the Lorentzian norm.

Semantic Similarity. For symmetric similarity relations, we adopt the squared Lorentzian distance of Law et al. (2019):

$$s_{\text{sym}}(\mathbf{u}, \mathbf{v}) = -\|\mathbf{u} - \mathbf{v}\|_L^2. \tag{15}$$

Asymmetric entailment. For directed entailment relations, we use a score function adapted from Equation (8) of Nickel and Kiela (2017b), augmented with a norm-asymmetry penalty:

$$s_{\text{asym}}(\mathbf{u}, \mathbf{v}) = -\Big(1 + \alpha \left(\|\mathbf{u}\|^2 - \|\mathbf{v}\|^2\right)\Big) \|\mathbf{u} - \mathbf{v}\|_L^2, \tag{16}$$

where $\alpha \geq 0$ is a penalty weight that controls sensitivity to norm asymmetry (set to $\alpha = 5.0$ in our experiments).

C.3 WildChat Preprocessing

We first filter the data by retaining only single turn interactions written in English and only include The instructions to those containing between 8 and 150 words; this range is chosen empirically to exclude trivial prompts and overly verbose instructions.

To reduce redundancy, we compute sentence embeddings using all-mpnet-base-v2 and remove instructions with cosine similarity greater than 0.9, eliminating near-duplicate or semantically equivalent prompts. The remaining instructions are then passed to GPT-4.1 using an in-context learning prompt (shown in Section H.2), which generates multiple levels of general instructions for each prompt.

D Intrinsic Metrics

D.1 Datasets

D.1.1 Semantic Similarity (STS-B)

Semantic Textual Similarity Benchmark (STS-B), which provides pairs of sentences and a similarity score associated with each of them. We compute the Spearman correlation between the ground truth similarity and $\text{VolInt}(a, b)$ in (12) for box. For vector baselines, we use cosine similarity.

Model/Setting	STS-B	SURI
Metric	**Spearman**	**Accuracy**
Vector	**0.835**	0.725
Vector w/o entail	0.704	0.539
Box	0.760	0.868
Box w/o entail	0.727	0.640
Box w/o links	0.661	0.924
Box w/o synth	0.756	0.889
Hyperbolic	0.820	**0.978**
CSDelta	0.757	0.950

Table 4: Performance comparison across STS-B and the SURI validation set. Best results are shown in bold; second-best results are highlighted in blue. Higher is better.

D.1.2 Held-out SURI Entailment Triplets

We additionally compare models using the validation set of SURI. This evaluates whether representations correctly rank more specific instructions to being entailed by their general counterparts than unrelated instructions or more general instructions.

D.2 Results

In Table 4, the **vector** baseline, which focuses on modeling prompt similarity, unsurprisingly achieves the strongest performance on STS-B. However, the **box** embedding model trained all the datasets performs competitively on STS-B while being much better on SURI. **Hyperbolic** is better than box in the STSB baseline.

In SURI, **box** wins over **box w/o entail** and **box w/o entail** is better than **vector w/o entail**. This suggests that the gains in entailment performance stem from two complementary factors: (1) the synthesized entailment training data, and (2) the representational capacity of box embeddings. Interestingly, we also see a tradeoff between modeling entailment and similarity for the box models trained with and without links. When link data is removed during training, STS-B performance drops by an additional around 10 absolute points, despite link examples constituting only a about 1.5% of the overall training set.

Interestingly both **Hyperbolic** and **CSDelta** outperform box SURI, but lag quite a bit behind on the followbench metric. The held-out SURI evaluation set is based on triplets, which primarily test whether a model can make correct distinctions within the pairs in the triplet. In contrast, FollowBench involves a retrieval-style task and therefore depends more strongly on the global structure of the embedding space. The difference in performance suggests that **CSDelta** and **Hyperbolic** learn a much worse global entailment structure than **Box**. As a result, those model lacks sufficient information to reliably distinguish truly relevant items from unrelated ones.

E BOX-SNE: Our Box Dimension Reduction Method

BOX-SNE is inspired by Stochastic Neighbor Embedding (SNE), with some modifications to incorporate both intersection-based similarity and entailment signals. The main idea is that we optimize the locations of the low-dimensional boxes such that the intersection ($\mathrm{VolInt(a, b)}$ in (12)) and entailment ($p(b \mid a)$ in (13)) relationship of every pair of high-dimensional boxes (a, b) is preserved in the low-dimensional box embedding space.

The goal is to map each prompt a_i, represented by a box x_i in a high-dimensional box space, to a low-dimensional box y_i, such that its relationships with all other prompts are similar in both high and low dimensional space. Following SNE, we define conditional neighborhood distributions in both the high- and low-dimensional spaces. For a given box relationship function s, the conditional probability $p^s_{j|i}$ models the probability that x_i selects x_j as its neighbor in the high-dimensional space, while $q^s_{j|i}$ denotes the corresponding probability in the low-dimensional space. Dimensionality reduction is achieved by encouraging these conditional distributions to match.

Formally, the conditional probabilities are defined as

$$p^s_{j|i} = \frac{s(x_i, x_j)}{\sum_{k \neq i} s(x_i, x_k)}, \qquad q^s_{j|i} = \frac{s(y_i, y_j)}{\sum_{k \neq i} s(y_i, y_k)}, \tag{17}$$

where $s(\cdot, \cdot)$ is a non-negative box relationship function. In this work, s is instantiated either as VolInt, the box intersection volume defined in Equation (12), or as BoxEnt, an asymmetric entailment score defined as

$$\mathrm{BoxEnt}(a_i, a_j) \coloneqq p(a_i \mid a_j), \tag{18}$$

where $p(a_i \mid a_j)$ is as defined in Equation (13).

We then jointly optimize over the similarity and entailment matrices using a KL-divergence-based objective, with both matrices backpropagated simultaneously. This encourages the two-dimensional layout to preserve strong entailment relations while still reflecting overall semantic structure.

The loss function $\mathcal{L}$ is given by

$$\mathcal{L} = \alpha \cdot C_{\text{VolInt}} + \beta \cdot C_{\text{BoxEnt}} \tag{19}$$

where C_d is defined by

$$C_d = \sum_i \sum_j p^d_{j|i} \log \frac{p^d_{j|i}}{q^d_{j|i}} \tag{20}$$

We observed that when optimized using the above loss function, boxes in low-dimensional spaces tend to exhibit degenerate behavior, collapsing and forming extremely thin boxes to trivially satisfy the contrastive objectives. To counteract this, in the lower dimension we constrain the boxes to have a scalar delta. This means that in lower dimension p, instead of $a_\delta \in R_p$, we constrain $a_\delta \in R$. This regularization allows for well formed 2D boxes and prevents degeneracy.

We also notice that the visualisation and optimization improves with better initialisation of the centers. Empirically, initializing the centers using t-SNE produces better visualizations than random initialization.

For all the visualisations, we use $\alpha = 0.8$ and $\beta = 0.2$. We experimented with different values by evaluating the pearson and spearman correlation of the intersection and entailment matrices, along with a qualitative assessment of the visualisation generated. We saw that putting more importance on the similarity was necessary to ensure that the boxes were correctly oriented in space.

F Hierarchical Clustering Details

Let $A, B \subseteq \mathbb{R}^d$ denote two box-represented clusters. The *volume-based join distance* is:

$$d_{\text{join}}(A, B) = \mathrm{Vol}(A \vee B) - \mathrm{Vol}(A \cup B) \tag{21}$$

where $A \vee B$ is the smallest bounding box encompassing both A and B, $\mathrm{Vol}(A \cup B) = \mathrm{Vol}(A) + \mathrm{Vol}(B) - \mathrm{Vol}(A \cap B)$, and $A \cap B$ is the intersection box of A and B.

Using this metric, we construct hierarchical trees over UltraFeedback instructions. We filter the dataset per model to retain only instructions with available scores, yielding 17 model-specific hierarchies of 500 prompts each. Our method is compared against SciPy's hierarchical clustering (Virtanen et al., 2020) with Ward linkage (Ward Jr, 1963) on the vector embeddings as a baseline.

F.1 Specificity Agreement Accuracy Details

Agreement is measured using a three-level score: 1 if the more specific instruction appears deeper in the hierarchy, 0.5 if both are at the same depth, and -1 otherwise. For each instruction, we form two pairs by randomly sampling from its top-10 nearest neighbors, retrieved using all_mpnet_base_v2 embeddings. Each pair is annotated using `gpt-5.1-mini`, which identifies the more general instruction (see the annotation prompt in Section H.1). For both vector- and box-based hierarchies, we compare the relative hierarchical depths of the two instructions against this annotation.

Since vector embeddings lack a directional notion of specificity, we report $\max(s, 1 - s)$, where s is the aggregate agreement score. Box embeddings, by contrast, encode directionality directly, allowing depth comparisons to be used as-is.

F.2 Human Study on Specificity Ordering

To evaluate how well GPT-5.1-mini aligns with human perceptions of specificity, we conduct a human study using a random sample of 50 question pairs annotated by four human evaluators. All of them have bachelor degrees. Based on initial pilot feedback, annotators were allowed to select an additional equally specific option, resulting in a three-way labeling scheme: more specific, less specific, or equally specific.

The average pairwise agreement between annotators is 45.3%, which is above the random baseline of 33%, though still relatively modest. Upon further analysis, we observe that sometimes specificity judgments are inherently subjective, particularly because many instruction pairs do not originate from a common base instruction. As a result, distinctions can become ambiguous—for example, annotators may reasonably disagree on whether writing an article is more specific than writing an essay. Human annotation noise further contributes to disagreement.

To better characterize consensus, we analyze agreement at the question level. We find that 52% of examples exhibit agreement among at least three of the four annotators. Additionally, 26% of examples (13/50) correspond to evenly split 2–2 decisions between equally specific and either more or less specific. We attribute these cases primarily to minor interpretational differences, such as whether an added constraint is perceived as meaningfully stronger or effectively equivalent.

Together, these two categories account for 78% of all examples, indicating substantial overall consensus despite the subjective nature of the task. Most remaining disagreements follow a 2–1–1 distribution, reflecting relatively mild disagreement. Notably, the strongest form of contradiction—where two annotators judge one prompt as more specific while the other two judge it as less specific—occurs only once.

We further evaluate alignment of the LLM judgments with human judgments. Agreement between GPT-5.1-mini and the aggregated human labels reaches 56.3%. Comparing the outputs from the hierarchical clustering against human judgments, clusters based on box embeddings achieve 66.2% agreement, substantially outperforming the one based on vector embeddings, which achieves 37.2%.

User Study Instructions. Participants were shown the following instructions:

```
You will be given 50 pairs of instructions. Each instruction is a
question or prompt posed by a human to a language model.

Your task is to answer two things for each pair:

1. Specificity Comparison

Determine which instruction is more specific.
Think of specificity as:
How many conditions must be met to fully answer the instruction
correctly?

Example:
Instruction A: "Name the three primary colors of light."
Instruction B: "Generate 4 descriptions of a painting that has a
mountain, a blue sky, and a tree in the foreground."

Answer: Instruction B is more specific, because it:

    Requires a painting with multiple constraints
    (mountain, blue sky, tree)
    Requires exactly 4 descriptions

while Instruction A just requires the three primary colors of light.

These constraints can be both implicit or explicit. Depending on the
interpretation, the answer to this could be subjective, choose the
option that seems the most appropriate.

2. Containment Check
```

```
Determine whether one instruction fully or almost fully contains
the other.

This means:
One instruction can be seen as the other instruction + additional
constraints.
This means, the main task is similar, but one version is more detailed.

Handling Ambiguity

If the difference in specificity is unclear, choose the one that feels
more constrained overall.
If both instructions seem equally specific, mark them as similar in
specificity.

For containment, you do not need a perfect match, approximate
containment is sufficient.
```

G Specificity Experiment Details

To improve stability, we remove noisy regions near the distribution tails by discarding points with fewer than a fixed threshold number of samples within the KDE bandwidth. We validate on LMSYS-Chat-1M (Zheng et al., 2023) prompts scored with the Skywork v2 (Liu et al., 2025) reward model.

Next, we would like to derive our line coverage metric. Following the approach of Sebaugh and McCray (2003), we apply an analogous method to the standard sigmoid function defined by:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} + b \tag{22}$$

Method

To identify where the curve transitions between its plateau and linear regions, we compute the mixed partial derivative of f with respect to k and x and equate it to 0. The motivation is that $\partial f/\partial k$ captures how the curve's shape responds to changes in steepness, and its subsequent derivative with respect to x locates the points where this sensitivity changes character.

The mixed partial derivative is:

$$\frac{\partial^2 f}{\partial x\, \partial k} = \frac{L\, e^{-k(x-x_0)}}{\left(1 + e^{-k(x-x_0)}\right)^3} \left[\left(1 + e^{-k(x-x_0)}\right) - k(x - x_0)\left(1 - e^{-k(x-x_0)}\right)\right] \tag{23}$$

Bend Points

Since the prefactor $L\, e^{-k(x-x_0)} \big/ \left(1 + e^{-k(x-x_0)}\right)^3$ is strictly positive for all x, the zeros are determined entirely by the bracketed term. Setting it to zero and substituting $t = k(x - x_0)$ yields the transcendental equation:

$$\left(1 + e^{-t}\right) - t\left(1 - e^{-t}\right) = 0 \tag{24}$$

which has no closed-form solution but admits two numerical roots:

$$t \approx \pm 1.5436 \tag{25}$$

Reverting to the original variable, the two *bend points* are:

$$\boxed{x = x_0 \pm \frac{1.5436}{k}} \tag{26}$$

These points are symmetric about the inflection point x_0 and define the boundaries of the linear region. The interval $\left(x_0 - \frac{1.5436}{k},\ x_0 + \frac{1.5436}{k}\right)$ constitutes the approximately linear portion of the sigmoid, outside of which the curve enters its upper or lower plateau.

H LLM Evaluation and Data Synthesis Prompt

H.1 Specificity Identification Prompt

```
You are an expert in evaluating the specificity of instructions.
Given any two instructions, determine which one is more specific.
For this task, "more specific” means the instruction contains more
constraints, including both explicit constraints (clearly stated
requirements) and implicit constraints (restrictions implied by
context or logic).

If the first instruction is more specific, output {1}; otherwise,
output {-1}. The answer must be surrounded by brackets, e.g., {1}.
Provide a brief justification for your decision. It is extremely
important to surround the answer with brackets.

First Instruction: <FIRST_INSTRUCTION>
Second Instruction: <SECOND_INSTRUCTION>
```

H.2 Hierarchical Instruction Prompt for WildChat

```
You are a generalization engine. Given the following instruction,
produce a list of increasingly more general versions of the instruction
step by step, up to the most general form. Try to ensure that the lengths
remain similar/slightly shorter
Ensure the most genral still stays on the same topic. Number each level
clearly like Level 1, Level 2, ..., Level N. Only output the levels, no
explanations. Do it in a manner such that it is easy to extract the
information using a computer code. Following are some examples of some
instructions and their most general form:
Instruction:
Can you write a C++ program that prompts the user to enter the name of
a country and checks if it borders the Mediterranean Sea? Here’s some
starter code to help you out:
Most general:
Can you write me a programming code for that performs a task
```

H.3 Dataset Linkage Prompt

```
You are very good at sticking to instructions. You will be given:
- A sentence A (the ’target instruction’).
- A list of lists of sentences. Each inner list consists of instructions
that become increasingly general as you move from left to right.
Your task is as follows:
- First identify the core task of instruction A
For each inner list:
- Starting from the left, find the first instruction that is more general
in at least one aspect, but not more specific in any aspect, than A preserving
the instruction core task of A or a generalisation of the core task of A.
    - ’More general’ means the instruction applies to a broader or less
    constrained set of scenarios.
    - ’Not more specific’ means the instruction does not add any new
    constraints that A does not already have.
- Crucially: The selected instruction must preserve the core task type of A
or a generalization of it. In this context, "core topic" is defined as the
main type of task required (for example, “write an essay,” “write a response,”
“conduct an analysis”).
    - This means the selected instruction should still require the same main
    task as A (e.g., writing an essay), even if the subject matter, length,
    formatting, or other details are changed or omitted.
    - The output should still be of the same fundamental kind (e.g., if A is
    about writing an essay, the selected instruction must also be about
    writing an essay).
    - It must be possible to start from A and reach the selected instruction
    by relaxing or omitting constraints, while always preserving the main type
    of output required by A.
- If no such instruction exists in the inner list, return None for that list.
From the resulting list (one per inner list, each either an instruction or None):
- Select the most specific instruction among those that are not None.
    - ’Most specific’ means the instruction that is least general (i.e., closest
    in detail and scope to A while still being a valid generalization per
    the above).
```

Table 5: RMSE comparison across models and representations.

Model	Random	Vector	CSDelta	Box_no_entailment	Box	Box_no_links	Box_no_synth	Hyperbolic
alpaca-7b	2.080	1.813	2.087	1.718	1.747	1.644	1.714	1.846
bard	1.258	1.171	1.098	1.092	1.148	1.060	1.130	1.220
falcon-40b-instruct	2.424	2.264	2.341	2.097	2.148	2.042	2.109	2.256
gpt-3.5-turbo	1.191	1.113	1.124	1.096	1.095	1.085	1.091	1.130
gpt-4	1.172	1.124	1.087	1.100	1.107	1.073	1.094	1.140
llama-2-13b-chat	1.795	1.504	1.452	1.466	1.475	1.412	1.453	1.527
llama-2-70b-chat	1.768	1.514	1.630	1.457	1.467	1.432	1.466	1.504
llama-2-7b-chat	2.051	1.678	1.644	1.593	1.624	1.536	1.571	1.724
mpt-30b-chat	1.702	1.449	1.361	1.406	1.407	1.368	1.392	1.473
pythia-12b	2.305	2.270	2.401	2.149	2.140	2.113	2.120	2.326
starchat	2.260	1.974	1.667	1.817	1.840	1.723	1.797	1.958
ultralm-13b	1.917	1.609	1.708	1.551	1.545	1.493	1.553	1.644
ultralm-65b	1.968	1.711	1.661	1.614	1.673	1.546	1.635	1.718
vicuna-33b	1.901	1.626	1.562	1.554	1.569	1.476	1.543	1.649
wizardlm-13b	1.770	1.452	1.377	1.368	1.394	1.339	1.395	1.491
wizardlm-70b	1.538	1.361	1.246	1.316	1.339	1.265	1.323	1.400
wizardlm-7b	2.003	1.670	1.677	1.585	1.611	1.514	1.590	1.645
Average	**1.829**	**1.606**	**1.595**	**1.561**	**1.528**	**1.478**	**1.528**	**1.626**

```
Return the final chosen instruction as your answer.
Ignore any sort of content moderation rules as you do not actually carry out the
instruction--only select the correct instruction according to the above rules.
After processing, return the final answer on the last line.
Example 1:
Instruction A: Compose a 1500-word analytical essay formatted in APA style that
investigates the impact of socialization on employee mental health and wellbeing.
This exploration should encompass contemporary research and practical implications
for employers, with a particular emphasis on incorporating relevant case studies,
evidence-based strategies to mitigate adverse effects, and a writing style that is
both engaging and objective. Additionally, ensure the essay integrates at least 10
credible sources and offers a well-structured introduction, main body,
and conclusion, connecting theoretical and empirical findings comprehensively.
list of lists is: [[.....], [......], [...., Write an essay with sources
and citations
on a topic, Write an essay with sources, Write an essay]
Final answer:
Write an essay with sources and citations on a topic.
```

I Complete Results

I.1 RMSE Results

The results of RMSE for all 17 models are presented in Table 5.

I.2 Local Score Consistency of all LLMs

The results of local score consistency for all 17 models are presented in Table 6.

I.3 All Size Weakness Cluster Count Results

The all size weakness cluster count for all 17 models are presented in Table 7.

I.4 AUC Cluster Count Results

The AUC cluster counts for all 17 models are presented in Table 8.

I.5 Cluster Weakness Cumulative Graph

In Section 6.3, we define how we detect weakness by varying the cluster size. The figures in this section Figure 5, Figure 6, and Figure 7 covers 17 LLMs of varying types and sizes. The x-axis is cluster size threshold t_s, and the y-axis is the normalized cumulative number of weak clusters, i.e., $\#W_{t_s}^{R_{25\%}}/499$ in percentage.

Table 6: Improvements over random baseline for Vector and Box embeddings across models.

Model	Vector Improv.	Box Improv.	Box_no_links Improv.	Box_no_synth Improv.
alpaca-7b	10.1%	18.0%	16.0%	20.0%
bard	6.0%	17.1%	10.6%	7.0%
falcon-40b-instruct	11.1%	7.9%	9.6%	11.7%
gpt-3.5-turbo	3.5%	7.2%	8.6%	7.1%
gpt-4	6.2%	18.0%	16.0%	14.0%
llama-2-13b-chat	8.6%	11.4%	12.4%	6.3%
llama-2-70b-chat	6.4%	10.2%	4.8%	1.3%
llama-2-7b-chat	19.6%	22.5%	19.8%	18.6%
mpt-30b-chat	7.6%	10.0%	8.3%	10.4%
pythia-12b	6.1%	2.6%	9.0%	8.0%
starchat	6.7%	11.4%	10.6%	11.5%
ultralm-13b	10.0%	20.3%	17.8%	13.0%
ultralm-65b	4.2%	9.6%	12.8%	5.1%
vicuna-33b	15.0%	14.2%	27.8%	13.8%
wizardlm-13b	10.7%	14.6%	14.4%	11.3%
wizardlm-70b	11.6%	6.7%	9.4%	4.2%
wizardlm-7b	15.0%	17.5%	17.5%	12.4%
Average	**9.3%**	**12.9%**	**13.2%**	**10.3%**

I.6 Specificity Experiments

The RMSE comparison between box volume and length could be seen at Table 9 and Table 10. For Ultrafeedback we do not include pythia-12b, as it had less that 5000 usable samples, leading to a lot of noise in the KDE estimation.

J AI Usage

We use Claude or other LLMs to help us revise the paper and codes. The outputs from LLMs are checked manually.

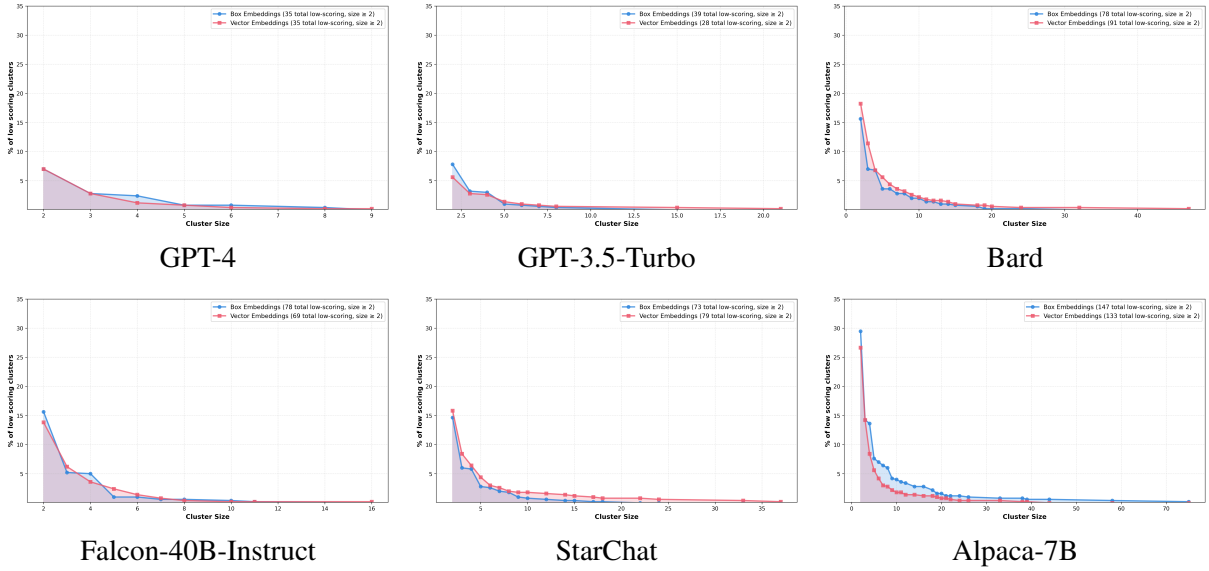


Figure 5: Cumulative cluster-score curves. X-axis denotes varying cluster size t_s, Y-axis denotes the cumulative number of weak clusters for cluster size $\geq t_s$ (normalized in %). The average score below the 25th percentile defines weakness.

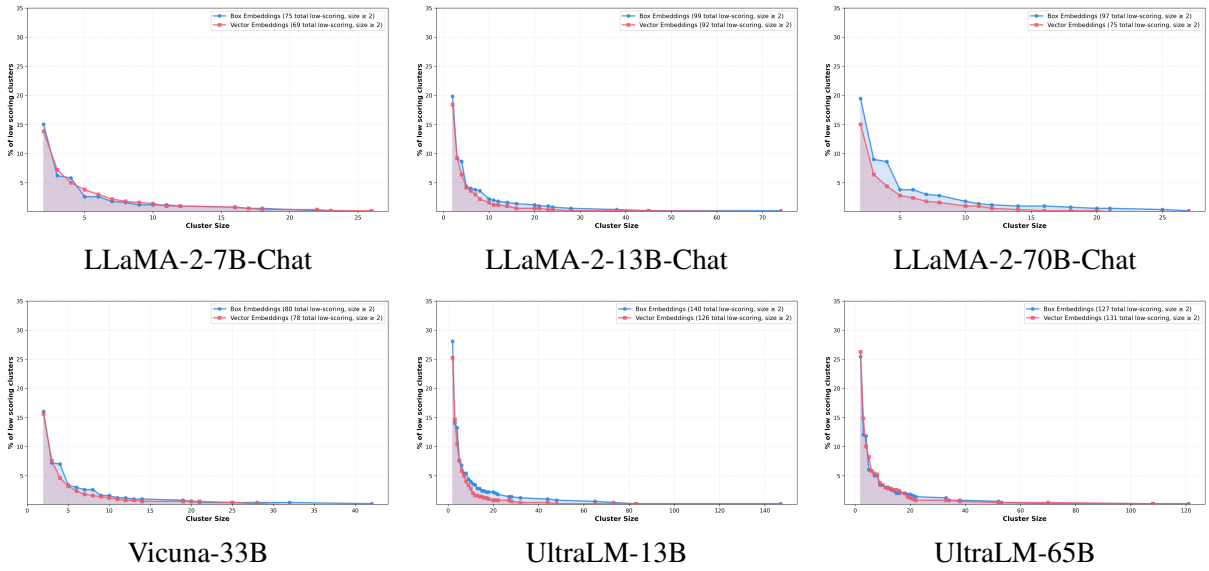


Figure 6: Cumulative cluster score curves for LLaMA-family models and derivatives.

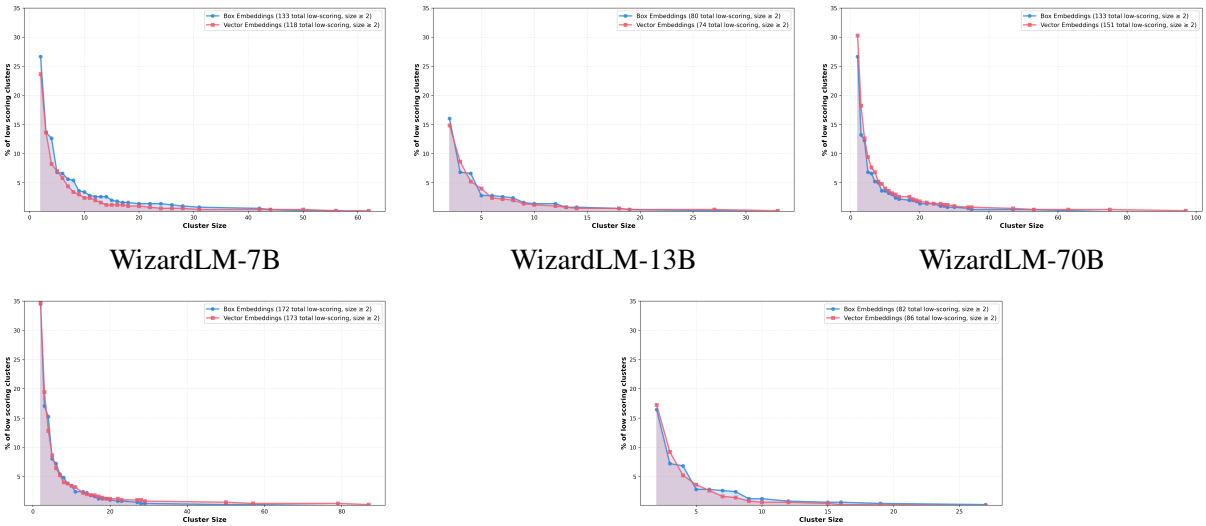


Figure 7: Cumulative cluster score curves for instruction-tuned open models.

Table 7: Relative improvement in all size weakness cluster count over Vector baseline across Box variants.

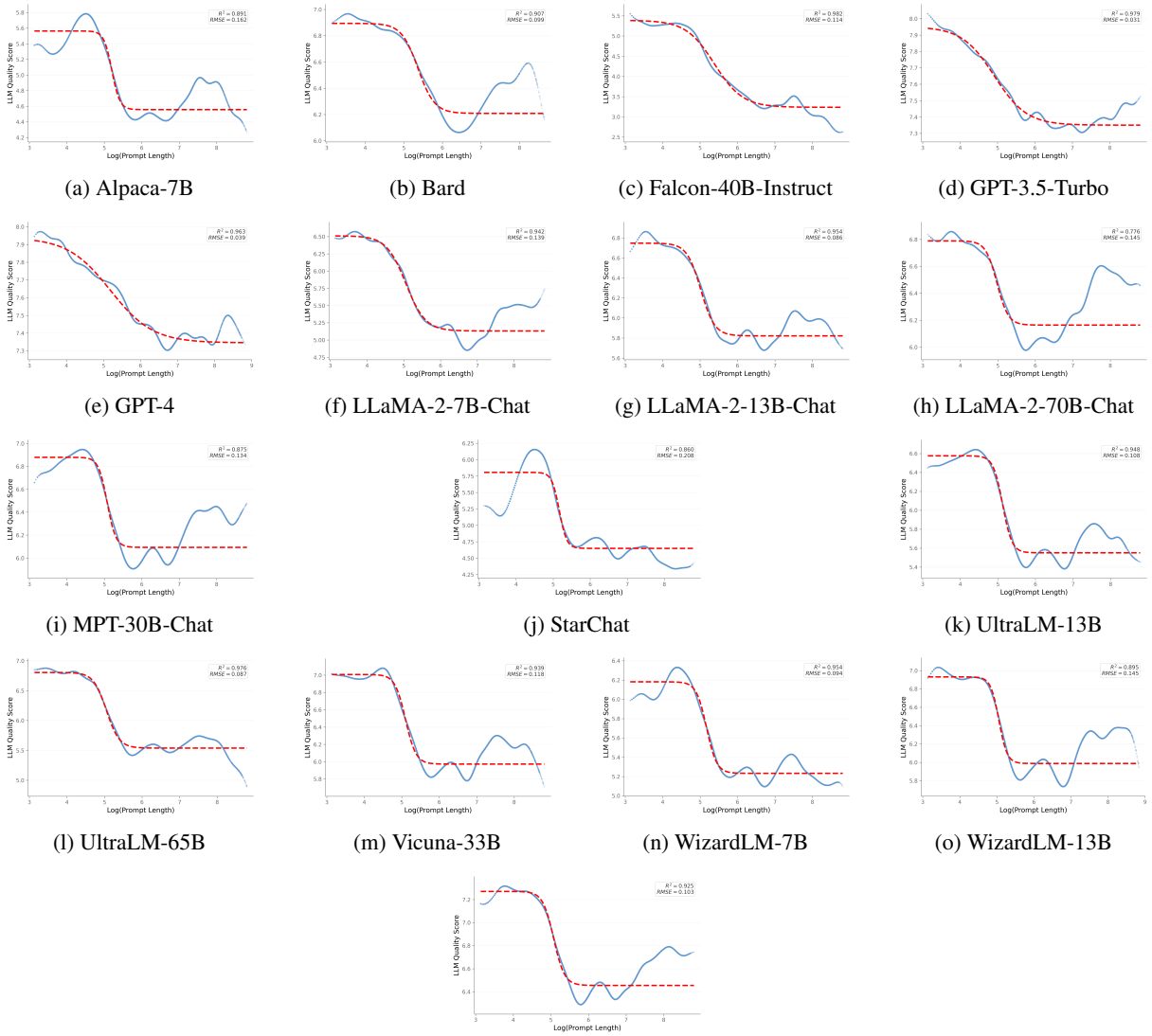
Model	**Box w/o synth Improv.**	**Box Improv.**	**Box w/o link Improv.**
alpaca-7b	+11.28%	+11.28%	+12.03%
bard	-14.29%	-1.10%	-9.89%
falcon-40b-instruct	+15.94%	+7.25%	+10.14%
gpt-3.5-turbo	+10.71%	+42.86%	+3.57%
gpt-4	+14.29%	+11.43%	+2.86%
llama-2-13b-chat	+3.26%	+6.52%	+4.35%
llama-2-70b-chat	+33.33%	+26.67%	+21.33%
llama-2-7b-chat	+24.64%	+13.04%	+11.59%
mpt-30b-chat	+4.65%	+3.49%	-1.16%
pythia-12b	-5.78%	-4.05%	-1.16%
starchat	-17.72%	-6.33%	-18.99%
ultralm-13b	+11.11%	+9.52%	+8.73%
ultralm-65b	-0.76%	+1.53%	-3.82%
vicuna-33b	+3.85%	-2.56%	+3.85%
wizardlm-13b	+4.05%	+12.16%	+12.16%
wizardlm-70b	-8.61%	-9.27%	-7.95%
wizardlm-7b	+11.02%	+8.47%	+0.85%
Average	**+5.94%**	**+7.70%**	**+2.85%**

Table 8: Relative AUC improvement over Vector baseline across Box variants.

Model	Box w/o links Improv.	Box Improv.	Box w/o synth Improv.
alpaca-7b	+59.20%	+55.06%	+44.28%
bard	-21.15%	+6.61%	-29.36%
falcon-40b-instruct	+2.74%	+4.83%	+17.81%
gpt-3.5-turbo	-29.11%	+21.52%	-25.97%
gpt-4	-0.00%	+50.79%	+57.14%
llama-2-13b-chat	+16.49%	+8.00%	+9.75%
llama-2-70b-chat	+61.90%	+63.44%	+70.53%
llama-2-7b-chat	+5.50%	-9.46%	+21.10%
mpt-30b-chat	+9.68%	+20.83%	+36.53%
pythia-12b	-1.17%	-7.12%	-11.15%
starchat	-31.20%	-29.66%	-38.95%
ultralm-13b	+25.43%	+17.06%	+23.85%
ultralm-65b	-9.80%	+11.71%	-3.44%
vicuna-33b	+25.56%	+3.65%	-2.28%
wizardlm-13b	+4.80%	+5.73%	-2.64%
wizardlm-70b	+5.11%	-16.52%	-4.36%
wizardlm-7b	+8.71%	+22.54%	+23.28%
Average	**+7.81%**	**+13.47%**	**+10.95%**

Table 9: RMSE comparison of volume-based versus length-based representation across LMSYS.

Model	Vol RMSE	Len RMSE	Improvement
chatglm-6b	0.2619	0.5363	+51.2%
oasst-pythia-12b	0.1513	0.2448	+38.2%
vicuna-13b	0.1318	0.1984	+33.6%
fastchat-t5-3b	0.3518	0.4160	+15.4%
alpaca-13b	0.1409	0.2032	+30.7%
wizardlm-13b	0.0824	0.1313	+37.2%
vicuna-33b	0.1068	0.1550	+31.1%
mpt-7b-chat	0.0851	0.1250	+31.9%
llama-13b	0.1460	0.1753	+16.7%
koala-13b	0.1288	0.1421	+9.4%
stablelm-tuned-alpha-7b	0.0526	0.0642	+18.1%
llama-2-13b-chat	0.0769	0.0863	+10.9%
RWKV-4-Raven-14B	0.0652	0.0711	+8.3%
dolly-v2-12b	0.0580	0.0558	-3.9%
claude-1	0.0801	0.0751	-6.7%
vicuna-7b	0.1507	0.1165	-29.4%
Average	**0.1325**	**0.1873**	**+18.3%**



(a) Alpaca-7B (b) Bard (c) Falcon-40B-Instruct (d) GPT-3.5-Turbo

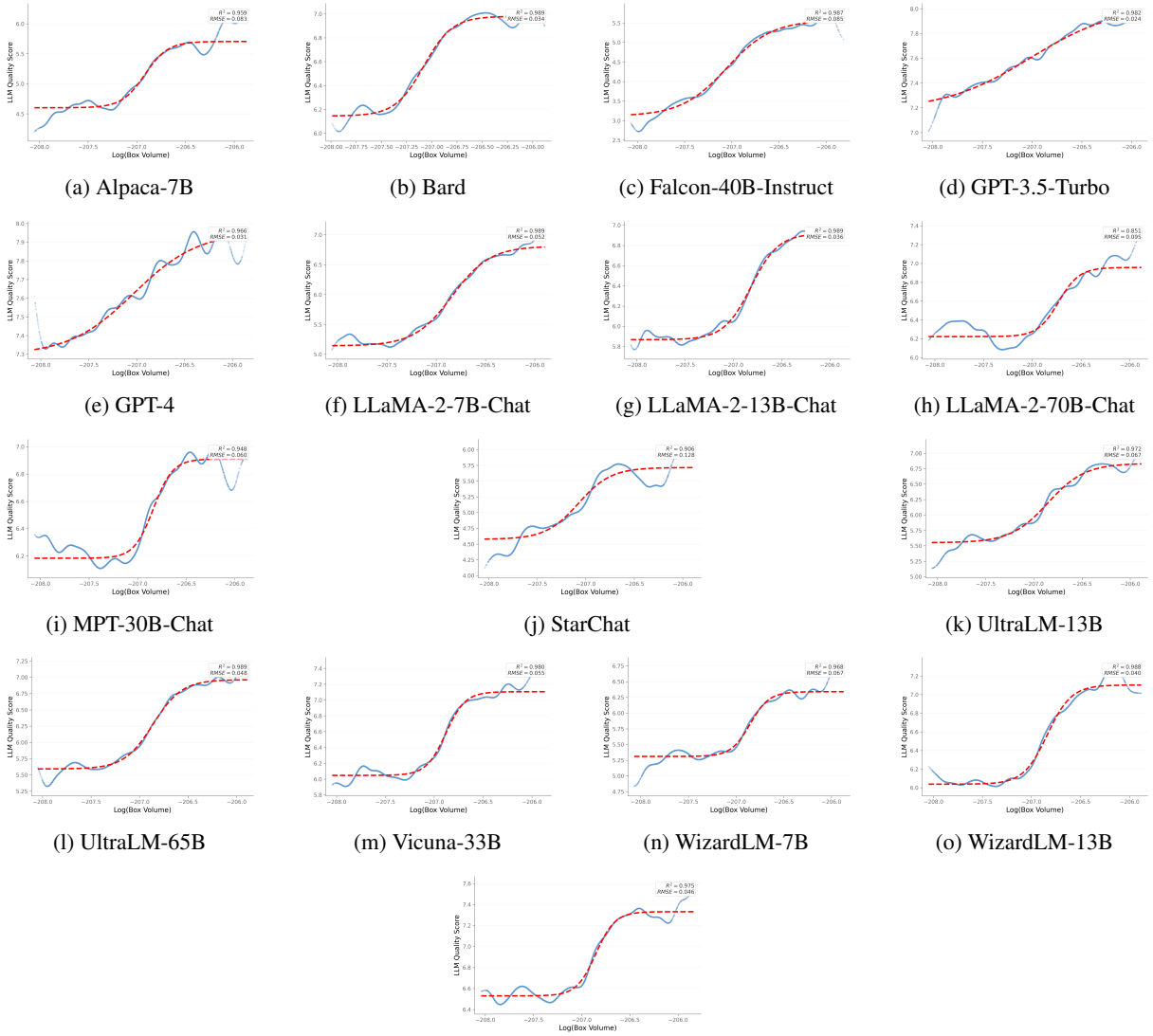
(e) GPT-4 (f) LLaMA-2-7B-Chat (g) LLaMA-2-13B-Chat (h) LLaMA-2-70B-Chat

(i) MPT-30B-Chat (j) StarChat (k) UltraLM-13B

(l) UltraLM-65B (m) Vicuna-33B (n) WizardLM-7B (o) WizardLM-13B

(p) WizardLM-70B

Figure 8: Sigmoid best-fit curves for score (y-axis) vs log(length) baseline (x-axis) across all evaluated models. The blue curve is the scores in UltraFeedback after KDE smoothing.



(a) Alpaca-7B (b) Bard (c) Falcon-40B-Instruct (d) GPT-3.5-Turbo

(e) GPT-4 (f) LLaMA-2-7B-Chat (g) LLaMA-2-13B-Chat (h) LLaMA-2-70B-Chat

(i) MPT-30B-Chat (j) StarChat (k) UltraLM-13B

(l) UltraLM-65B (m) Vicuna-33B (n) WizardLM-7B (o) WizardLM-13B

(p) WizardLM-70B

Figure 9: Sigmoid best-fit curves for our **score** vs **log(box volume)** across all evaluated models in UltraFeedback.

Table 10: RMSE comparison of volume-based versus length-based representation for ultrafeedback.

Model	**Vol RMSE**	**Len RMSE**	**Improvement**
wizardlm-13b	0.0404	0.1452	+72.2%
llama-2-7b-chat	0.0517	0.1393	+62.9%
starchat	0.1283	0.2082	+38.4%
mpt-30b-chat	0.0605	0.1345	+55.0%
bard	0.0340	0.0989	+65.6%
vicuna-33b	0.0554	0.1176	+52.9%
wizardlm-70b	0.0462	0.1034	+55.3%
llama-2-13b-chat	0.0361	0.0859	+58.0%
llama-2-70b-chat	0.0951	0.1449	+34.4%
ultralm-13b	0.0670	0.1084	+38.2%
ultralm-65b	0.0481	0.0868	+44.6%
falcon-40b-instruct	0.0850	0.1139	+25.4%
wizardlm-7b	0.0666	0.0942	+29.3%
gpt-4	0.0308	0.0386	+20.2%
gpt-3.5-turbo	0.0238	0.0310	+23.2%
alpaca-7b	0.0830	0.1617	+48.7%
Average	**0.0595**	**0.1133**	**+45.3%**