ABLE: Representing and Mapping LLMs via Attribution-Based Large-model Embedding

Anonymous ACL submission

Abstract

The explosive growth of large language models (LLMs) has led to an opaque ecosystem where undocumented model relationships hinder copyright protection, security auditing, and model routing. Existing representation methods struggle to address this challenge efficiently. Approaches analyzing internal parameters face scalability barriers due to structural heterogeneity across diverse architectures, while methods relying on external outputs are susceptible to behavioral mimicry, where distinct models converge to similar predictions despite differing underlying mechanisms. To bridge this gap, we propose ABLE (Attribution-Based Large-model Embedding), a novel framework that leverages the interpretability space to construct model representations. By aggregating gradient-based feature attributions via a tokenizer-agnostic word-level alignment, ABLE captures the intrinsic cognitive patterns of models rather than surface-level outputs. Beyond empirical utility, we proved that ABLE is a Lipschitz continuous mapping with finite-sample convergence guarantees, ensuring stability and reliability. Extensive experiments on 239 LLMs demonstrate that our training-free approach achieving competitive or superior performance in relation prediction, model routing and benchmark score prediction[1].

1 Introduction

The ecosystem of large language models (LLMs) (Vaswani et al., 2017; Yang et al., 2024; Hao et al., 2022) is expanding at an explosive rate, with platforms like Hugging Face hosting hundreds of thousands of models (Zhao et al., 2023; Jain, 2022; Rothman, 2022). Yet this rapid growth comes with a transparency crisis: many models are released without clear documentation of their training origins, making the relationships between models opaque and invisible (Barman et al., 2024). Uncovering these hidden model relationships is crucial for copyright protection, enabling detection of unauthorized model reuse and intellectual property infringement (Menell, 1988; Shao et al., 2024; Horwitz et al., 2025); security auditing, allowing practitioners to trace backdoor propagation paths within model families (Cheng et al., 2024; Ryoo et al., 2013); and intelligent model routing, facilitating selection of the most suitable model based on capability similarity (Ong et al., 2024).

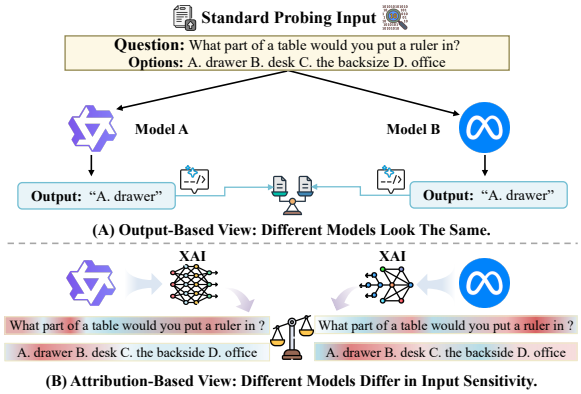

Figure 1: Motivation for attribution-based representations. (A) Given the same probing input, the output-based view cannot distinguish models that produce identical answers. (B) The attribution-based view reveals differences in input sensitivity, capturing how models rely on different tokens to reach the same prediction.

To address these challenges, researchers seek to construct compact LLM representations that enable systematic comparison and analysis across large model collections. Existing approaches generally fall into two categories: internal feature analysis and external output probing. Methods based on internal features directly analyze model parameters or activations (Yadav et al., 2023; Zhu et al., 2025). However, the structural heterogeneity in-

[1]Our code for ABLE extraction and analysis are available at this link: https://anonymous.4open.science/r/ABLE

herent in diverse models, such as differences in architecture or scale, necessitates complex weight alignment or layer mapping strategies (Mattheakis et al., 2019). These requirements impede the scalability of such methods for large-scale model representation. Conversely, methods relying on external outputs offer superior scalability by treating models as black boxes (Zhuang et al., 2024; Yax et al., 2024; Oyama et al., 2025). However, in applications such as copyright protection, since distinct model architectures or training strategies may converge to similar output distributions, relying solely on surface-level behavioral overlap can lead to erroneous conclusions (Orgad et al., 2024). Therefore, accurately representing LLMs and detecting genuine relationships requires exploring the underlying mechanistic similarities between models.

As illustrated in Fig. 1, much like distinct cognitive processes in humans, different models may produce identical outputs while relying on entirely different input features. Building on this insight, we propose ABLE (**A**ttribution-**B**ased **L**arge-model **E**mbedding), a framework that leverages the interpretability space to construct representations of LLMs. By computing the gradient of the model's output with respect to its inputs via the $Gradient \times Input$ (Kim et al., 2018; Wang et al., 2024), ABLE quantifies the contribution of each input feature to a specific prediction, thereby capturing the mechanistic patterns governing how a model processes information (Sturmfels et al., 2020; Nguyen et al., 2021; Kommiya Mothilal et al., 2021; Zhou et al., 2022). To ensure scalability across diverse architectures, we implement word-level alignment to map token-specific attributions into a unified vocabulary, effectively resolving tokenizer heterogeneity. We validate this training-free framework on 239 LLMs, demonstrating its effectiveness in model relationship analysis, routing, and benchmark score prediction.

Furthermore, we provide theoretical guarantees regarding the stability of the representation. Under standard regularity assumptions, we prove that ABLE constitutes a Lipschitz continuous mapping (Johnson et al., 1984) from the model parameter space to the embedding space, ensuring that models with similar parameters yield similar embeddings. We further establish finite-sample concentration bounds to guarantee that empirical representations converge to population-level embeddings. These results collectively establish ABLE as a principled and stable representation grounded in model parameters.

In summary, our contributions are as follows:

- We propose a novel paradigm that leverages the interpretability space to construct model representations. We introduce ABLE, a framework that utilizes gradient-based feature attribution to capture mechanistic similarities between models, thereby overcoming the limitations of surface-level output analysis.
- We provide theoretical guarantees for the stability of our representation. We prove that ABLE is a Lipschitz-continuous mapping from the model parameter space to the embedding space, ensuring that models with similar parameters naturally yield similar embeddings under standard regularity assumptions.
- We validate ABLE on 239 LLMs across diverse tasks including lineage reconstruction, model routing, and benchmark score prediction, demonstrating its scalability and effectiveness.

2 Related Work

2.1 LLM Representation

This section reviews related work on constructing representations for LLMs, covering both internal-feature and output-based approaches.

Internal-feature methods directly analyze model parameters or activations. Representative approaches study parameter changes from fine-tuning to resolve conflicts during model merging (Yadav et al., 2023), test derivative relationships by comparing model weights (Zhu et al., 2025), and analyze activation patterns across layers to quantify cross-model similarity (Zhou et al., 2024). These methods are well-suited for scenarios where models share compatible architectures, but require careful alignment when comparing models with different structures.

Output-based methods construct representations from model predictions or logits. Existing researches generate model embeddings from log-likelihoods on a standardized corpus (Oyama et al., 2025), construct phylogenetic trees by analyzing patterns in generated text (Yax et al., 2024), and learn embeddings from task performance vectors for model routing (Zhuang et al., 2024). These methods are architecture-agnostic and scalable, but are limited to capturing what models predict rather

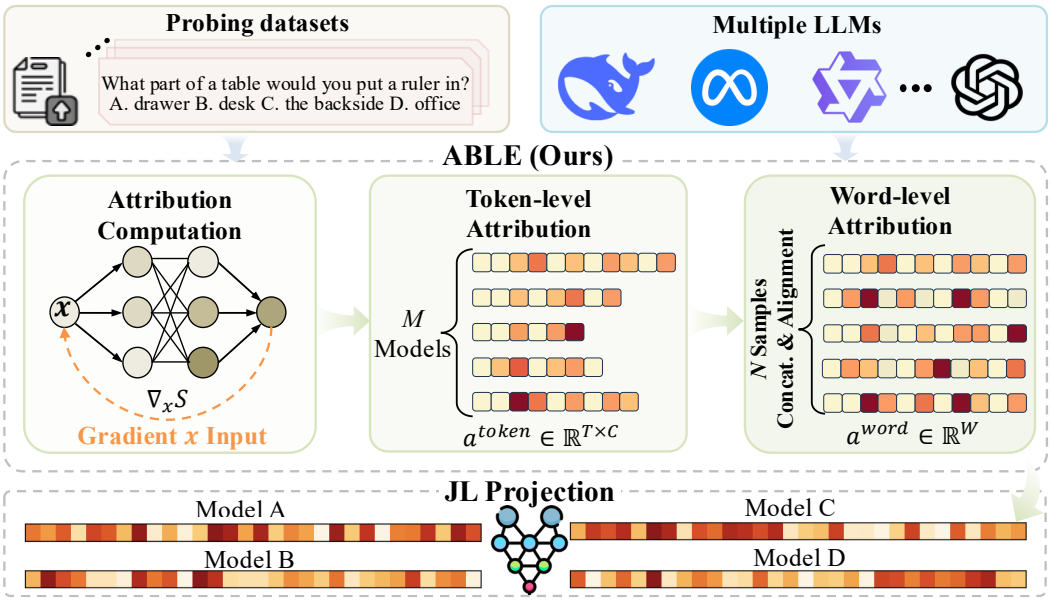

Figure 2: Overview of the ABLE pipeline. Given a model, we compute Gradient × Input attributions on pre-defined probing datasets, align token-level scores to a unified word vocabulary, and project the result to a compact low-dimensional embedding. JL: Johnson–Lindenstrauss.

than how they process inputs. ABLE bridges these two paradigms by using feature attribution as the representation basis.

2.2 Feature Attribution Methods

Feature attribution methods quantify the contribution of each input feature to a model's output, providing interpretable explanations for model predictions. White-box methods leverage gradient information, including Saliency Maps (Simonyan et al., 2013), Integrated Gradients (Sundararajan et al., 2017), and Layer-wise Relevance Propagation (Bach et al., 2015; Samek et al., 2021). Black-box methods probe models through input perturbations without requiring gradient access, such as LIME (Ribeiro et al., 2016) which fits local linear models, and SHAP (Lundberg and Lee, 2017) which applies game-theoretic Shapley values.

In the context of LLMs, attribution has been widely used for sample-level explanations, such as identifying salient tokens in classification (Liu and Avci, 2019) or generation tasks (Zhao and Shan, 2024). We extend this by aggregating sample-level attributions to construct model representations in interpretability space.

3 Methods

This section describes the ABLE representation construction pipeline (see Fig. 2). Given a collection of M language models and a pre-defined probing dataset, we produce a compact embedding $\mathbf{a}_m \in \mathbb{R}^K$ for each model m. The pipeline consists of three stages: (1) attribution computation, (2) cross-tokenizer alignment, and (3) dimensionality reduction.

3.1 Probing Data Design

We adopt multi-choice questions (MCQ) from test set of standard benchmarks as our probing format. Each sample consists of a question q and a set of candidate options $\{o_1, o_2, \ldots, o_C\}$, where C is typically 4–5. This design offers two key advantages. First, MCQ provide *deterministic attribution targets*: the log-probability of each option sequence given the question, $\log P_m(o_c \mid q)$, serves as a well-defined scalar output for gradient computation. Second, by computing attributions for all options, we implicitly capture how the model distinguishes correct from incorrect answers, a signal that reflects the model's cognitive strategy rather than merely its final prediction.

3.2 Attribution Computation

Among various attribution methods, we adopt Gradient × Input (GI) (Ancona et al., 2019, 2017; Nielsen et al., 2022) for two reasons: (1) computational efficiency, as GI requires only a single forward-backward pass per sample; and (2) theoretical tractability, as we prove in Appendix A that GI yields a stable, distance-preserving embedding.

For each model m and each sample, we compute the attribution of every question token to each option's log-probability. The attribution score for the t-th question token with respect to option o_c is:

$$\alpha_t^{(c)} = \langle \mathbf{e}_t, \nabla_{\mathbf{e}_t} S_c \rangle \,,$$
$$S_c = \sum_{j=1}^{|o_c|} \log P_m(o_c^{(j)} \mid q, o_c^{(<j)}), \qquad (1)$$

where $\mathbf{e}_t$ denotes the embedding of the t-th question token, and S_c is the total log-probability of option o_c.

For a question with T tokens and C options, this yields an attribution matrix $\mathbf{A}^{\text{token}} \in \mathbb{R}^{T \times C}$. To obtain a single attribution vector per sample, we flatten this matrix into $\mathbb{R}^{T \cdot C}$. Concatenating across all N samples in the probing dataset produces a model-level token attribution vector $\mathbf{a}_m^{\text{token}} \in \mathbb{R}^{D_{\text{token}}}$, where $D_{\text{token}} = \sum_{i=1}^{N} T_i \times C_i$.

3.3 Cross-Tokenizer Alignment

Different models employ different tokenizers, resulting in attribution vectors of incompatible dimensions (Kudo and Richardson, 2018). To enable cross-model comparison, we align all attributions to a unified word-level vocabulary through a two-step process.

Token-to-Character Mapping. For each token t in the flattened attribution vector $\mathbf{a}_m^{\text{token}}$, let a_t denote its attribution score. We identify its character span $[s_t, e_t)$ in the original text and uniformly distribute the attribution across all characters in this span:

$$a_{\text{char}}^{(i)} = \frac{a_t}{e_t - s_t}, \quad \forall i \in [s_t, e_t). \qquad (2)$$

This produces a character-level attribution vector $\mathbf{a}_m^{\text{char}} \in \mathbb{R}^L$, where L is the total number of characters in the input text.

Character-to-Word Aggregation. Given the character-level attributions, we aggregate them into word-level representations. We segment the character sequence into words using whitespace as the delimiter. For a word w spanning characters $[s_w, e_w)$, its attribution is computed as the sum of its constituent characters:

$$a_{\text{word}}^{(w)} = \sum_{i=s_w}^{e_w - 1} a_{\text{char}}^{(i)}. \qquad (3)$$

Whitespace attributions are appended to the preceding word to ensure no attribution is lost.

Since all models process the same probing text, this procedure yields a word-level attribution vector $\mathbf{a}_m^{\text{word}} \in \mathbb{R}^W$ for each model, where W is the total number of words in the probing corpus. This alignment addresses the tokenizer heterogeneity problem.

3.4 Dimensionality Reduction via Random Projection

The word-level attribution vector $\mathbf{a}_m^{\text{word}}$ can be high-dimensional ($W \sim 10^5$). To obtain a compact representation, we apply the Johnson-Lindenstrauss (JL) random projection (Johnson et al., 1984). Specifically, we sample a random matrix $\mathbf{R} \in \mathbb{R}^{K \times W}$ with independent and identically distributed entries drawn from $\mathcal{N}(0, 1/K)$, and compute:

$$\mathbf{a}_m = \mathbf{R}\, \mathbf{a}_m^{\text{word}} \in \mathbb{R}^K. \qquad (4)$$

By the JL lemma, pairwise distances are preserved up to a multiplicative factor $(1 \pm \epsilon)$ with high probability, provided $K = O(\epsilon^{-2} \log M)$. In practice, we determined the optimal value of K via downstream task performance (see Appendix B). The resulting $\mathbf{a}_m$ is the final ABLE representation for model m.

4 Experiments

We evaluate ABLE through a series of experiments designed to assess both its structural validity and practical utility. We first verify that ABLE captures meaningful model relationships through lineage reconstruction and relation prediction (Sections 4.2 and 4.3). We then demonstrate ABLE's practical value in model routing and benchmark score prediction tasks (Sections 4.4 and 4.5).

4.1 Experimental Setup

Datasets. We construct a balanced probing dataset $\mathcal{D}$ by randomly sampling 200 instances from each of six benchmarks: ARC-Challenge (Clark et al., 2018), Winogrande (Sakaguchi et al., 2021), MMLU (Hendrycks et al., 2021b,a), Hellaswag (Zellers et al., 2019), GPQA (Rein et al., 2024), and CommonsenseQA (Talmor et al., 2019), yielding $N = 1{,}200$ samples in total. This combination ensures diverse coverage across reasoning and knowledge domains. For model routing experiments, we additionally use the evaluation dataset from EmbedLLM (Zhuang et al., 2024).

Models. We evaluate $M = 239$ open-source LLMs spanning parameter scales from 70M to

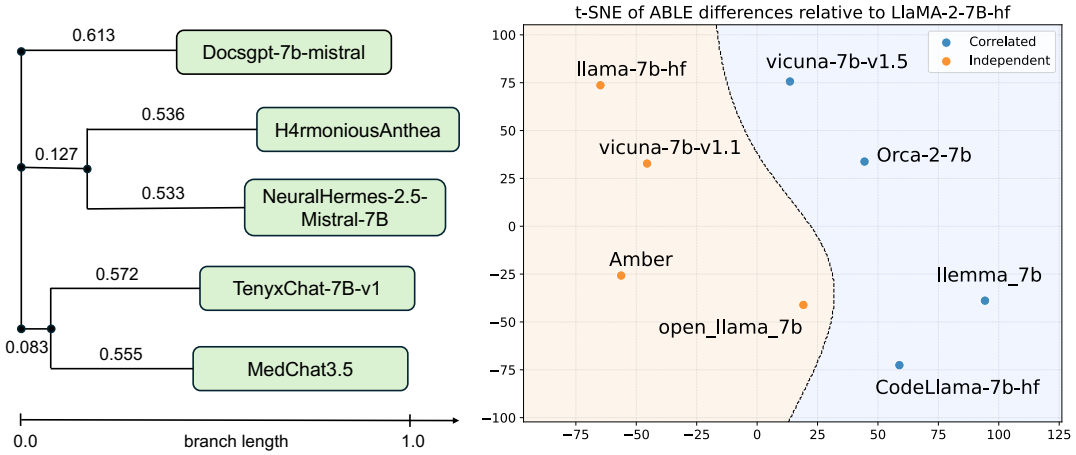

Figure 3: Lineage verification experiments. Left: Mistral family phylogenetic tree reconstructed from ABLE embeddings. Right: Llama-2 family models in ABLE space, showing clear separation between correlated and independent pairs.

70B. This collection encompasses foundation base models and instruction-tuned variants, as well as domain-specific models tailored for mathematics, coding, and medicine. More model details are provided in Appendix E.

4.2 Model Lineage and Atlas

To evaluate whether ABLE representations capture meaningful structural information and evolutionary history of LLMs, we present experiments at two scales: first, small-scale lineage verification on documented model relations; second, constructing a comprehensive atlas of the full model collection.

Lineage Verification on Mistral and Llama-2 Families. To validate that ABLE captures model lineage, we conduct two small-scale verification experiments. For the Mistral family, we select five leaf models with publicly disclosed fine-tuning histories (DocsGPT, NeuralHermes-2.5, TenyxChat-7B, MedChat3.5, and H4rmoniousAnthea) and construct an unrooted tree using the Neighbor-Joining method (Trees, 1987) on cosine distances calculated on ABLE. As shown in Fig. 3 (left), the results aligns with the ground-truth lineage disclosed by model creators (arc53, 2023; mlabonne, 2024; Tenyx, 2024; Ullah, 2024; Vallego, 2024). For the Llama-2 family, we examine eight models: four are correlated with Llama-2-7b-hf (Vicuna-7b-v1.5, CodeLlama-7b-hf, Llemma-7b, Orca-2-7b) and four are independent (Llama-7b-hf, Vicuna-7b-v1.1, Amber, Open-Llama-7b), following the relationship annotations by Zhu et al. (2025). As shown in Fig. 3 (right), correlated and independent models are clearly separated in ABLE space, with an SVM decision boundary achieving perfect classification.

Visualizing the Global Model Map. We then project the ABLE features into a two-dimensional space using t-SNE (Maaten and Hinton, 2008) to construct a Model Map. From Fig. 4, three major insights can be inferred. First, LLMs tend to aggregate based on their respective families, such as Llama, Qwen, and Mistral family. Second, fine-tuned models tend to cluster around corresponding base models. A prime example is the Llama 3 family (red) in the central region, where multiple LoRA-tuned variants cluster tightly around the base Llama-3-8B model. Third, fine-tuned models may also deviate from their families to align with models sharing similar tasks. For instance, Llemma-34B and DeepSeek-Math-7B-Instruct cluster together due to their shared focus on mathematical reasoning, while LoRA-8B-Code and CodeGemma-2B converge due to code-related fine-tuning (names labeled in red in Fig. 4). This indicates that ABLE can capture genealogical proximity and functional affinity among heterogeneous LLMs.

Constructing Cross-Family Phylogenetic Trees. While the Model Map provides a continuous two-dimensional projection, we further construct a hierarchical phylogenetic tree to explicitly reveal evolutionary relationships across model families. We select representative models from ten families (Mistral, Bloom, LLaMA-1, LLaMA-2, LLaMA-3, Yi, Qwen2, Qwen3, MPT, and Pythia) and compute pairwise cosine distances from their ABLE embeddings. We then apply

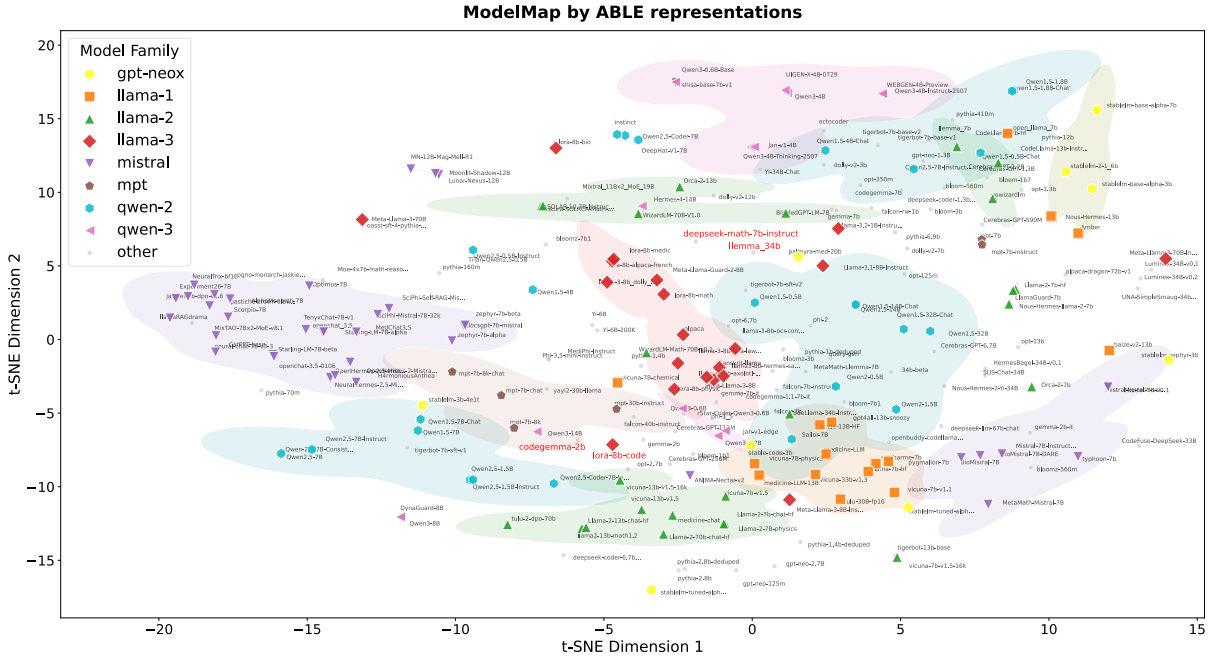

Figure 4: Model Map visualization using t-SNE projection of ABLE embeddings. Models are colored by family, and tightly clustered groups are highlighted with background shading using DBSCAN clustering. Names highlighted in red denote specific examples where models deviate from their families to cluster by functional affinity

hierarchical clustering to construct the tree, with branches colored by family membership.

As shown in Fig. 5, the resulting tree reveals several patterns consistent with known architectural and training relationships. First, models within the same family form cohesive sub-trees, confirming that ABLE embeddings capture family-level structural similarity. Second, Yi models cluster closely with the LLaMA family, which aligns with the documented fact that Yi adopts the same architecture as LLaMA. Similarly, Bloom and Pythia share a common ancestor branch, reflecting their shared GPT-style architectural lineage: Bloom is essentially GPT-3-like according to Hugging Face documentation, while Pythia is based on GPT-NeoX. Third, Qwen2 and Qwen3 occupy distant branches: Qwen2 resides within the LLaMA-dominated subtree, whereas Qwen3 branches off toward the Pythia-Bloom cluster. This separation is consistent with official documentations (Bai et al., 2023; Team et al., 2024; Yang et al., 2025), which reported that Qwen3 introduces significant changes including reasoning-focused training objectives, multi-stage training pipelines, and reinforcement learning strategies, representing departures from the Qwen2 training paradigm. These findings demonstrate that ABLE representations not only capture architectural inheritance but also reflect differences in training methodologies.

4.3 Relation Prediction

This section evaluates whether ABLE embeddings can predict pairwise model relationships. Specifically, given two models, the task is to determine whether one is derived from the other. This capability is important for intellectual property protection, as it enables detection of unauthorized reuse of proprietary models.

We formalize this as a binary classification problem. To construct the relation dataset, we identify correlated model pairs from 239 models using metadata from Hugging Face Hub, supplemented by manual verification. To simulate the class imbalance where unrelated pairs dominate, we sample negative pairs at a 2:1 ratio relative to positive relation pairs, yielding 135 model pairs in total. The dataset is split 8:2 into train and test sets. For classification, we represent each model pair by concatenating the ABLE embeddings of both models, and train an SVM classifier with an RBF kernel.

We compare against four baselines: (1) Random, which uniformly guesses correlated or independent; (2) Greedy, which predicts that all within-organization pairs are correlated and cross-organization pairs are independent; (3) Log-Likelihood (Oyama et al., 2025), a method that

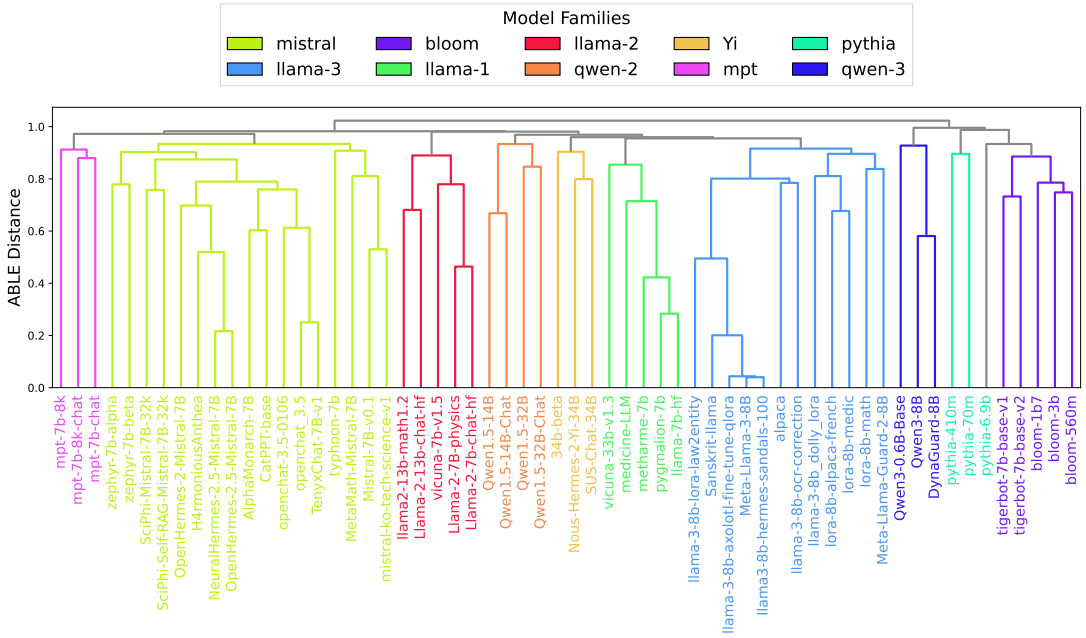

Figure 5: Phylogenetic tree constructed from ABLE embeddings using hierarchical clustering with cosine distance. Branches are colored by model family; gray branches indicate cross-family merging points.

Table 1: Model relation prediction performance of ABLE and four baselines (Values are mean ± standard deviation over 5 random seeds). Bold: the best performance.

Method	Accuracy	Precision	Recall	F1	AUC
Random	0.489±0.124	0.367±0.093	0.733±0.186	0.489±0.124	0.550±0.139
Greedy	0.674±0.096	0.527±0.313	0.222±0.157	0.309±0.203	0.561±0.110
PhyloLM	0.704±0.141	0.596±0.182	0.844±0.169	0.667±0.090	0.893±0.059
Log-Likelihood	0.721±0.041	0.586±0.064	**0.861±0.150**	0.692±0.070	0.891±0.053
ABLE(Ours)	**0.867±0.042**	**0.836±0.077**	0.756±0.093	**0.790±0.065**	**0.906±0.057**

computes log-likelihood vectors on a fixed dataset as model features; and (4) PhyloLM (Yax et al., 2024), an output-based method uses output similarity distances to other LLMs as a signature vector.

As shown in Table 1, ABLE achieves the best performance on Accuracy, Precision, F1, and AUC. Notably, ABLE and output-based methods exhibit different precision-recall trade-offs: Log-Likelihood attains the highest recall but with lower precision, whereas ABLE achieves substantially higher precision at the cost of moderate recall. This reflects that ABLE requires stronger evidence before classifying a model pair as correlated, resulting in more confident positive predictions. This conservative behavior is valuable for intellectual property protection, where falsely accusing an independent model of infringement can lead to legal disputes and reputation damage (Van Wyk et al., 2023).

4.4 Model Routing

To rigorously evaluate the effectiveness of ABLE features in model routing, we adopt the matrix factorization framework, dataset, and evaluation metrics from EmbedLLM (Zhuang et al., 2024). This architecture uses a learnable linear projection layer to map frozen question embeddings into the model feature space, predicting the probability that a model answers a question correctly via element-wise interaction. Unlike EmbedLLM, which optimizes model embeddings end-to-end, we keep all ABLE features frozen throughout training and only learn the semantic alignment from question embeddings to the model space.

As shown in Table 2, the router based on frozen ABLE features achieves 67.6% accuracy, slightly higher than the fully trained EmbedLLM router. While the accuracy gain is modest, the key advantage of our approach lies in training-free scalabil-

Table 2: Model routing performance of ABLE against three baselines (Values are mean ± standard deviation over 5 random seeds). ABLE router is competitive with EmbedLLM router.

	Random	Single-Best	EmbedLLM	**ABLE**
Router Accuracy	0.413 ± 0.101	0.605 ± 0.000	0.665 ± 0.003	$\mathbf{0.676 \pm 0.001}$

Table 3: Benchmark score prediction performance. We report the Pearson (r) and Spearman (ρ) correlation coefficients between the ground-truth and the scores predicted by ridge regression using ABLE.

	ARC	HellaSwag	MMLU	TruthfulQA	WinoGrande
Pearson r	0.8781	0.7764	0.8554	0.9127	0.8374
Spearman ρ	0.8898	0.8157	0.8637	0.8222	0.8602

ity. The ABLE router can accommodate an arbitrary number of models, and when new models are added to the pool, only their ABLE features need to be computed without retraining. In contrast, EmbedLLM learns model embeddings end-to-end, necessitating retraining whenever new models are introduced.

4.5 Benchmark Score Prediction

In this section, we investigate whether ABLE representations can predict LLM performance on standard benchmarks. Using scores from five core benchmarks on the Hugging Face Open LLM Leaderboard (Fourrier et al., 2024; Myrzakhan et al., 2024; Beeching et al., 2023) as labels and ABLE features as input, we evaluate the predictive ability of a ridge regression model (McDonald, 2009) via leave-one-out cross-validation.

As illustrated in Table 3 and Fig. 6, ABLE features demonstrate strong correlations with ground-truth benchmark scores, achieving Spearman ρ values between 0.81 and 0.89. This high consistency indicates that model hierarchies predicted solely from ABLE representations closely mirror those derived from standard evaluations. These findings suggest that ABLE effectively encodes model capabilities within a linear subspace. Specifically, for large-scale comparisons where relative ranking is prioritized over precise scoring, ABLE serves as a computationally efficient surrogate for resource-intensive benchmarks.

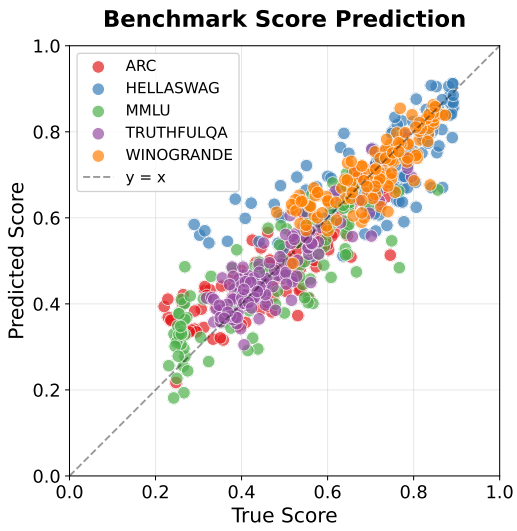

Figure 6: Scatter plots showing predicted benchmark score against ground truth scores on five benchmark using ABLE features.

5 Conclusion

In this work, we aim to mitigate the growing transparency crisis in the LLM ecosystem by proposing ABLE, a framework that bridges the gap between internal mechanism analysis and external behavioral probing. By leveraging the interpretability space, ABLE captures the intrinsic cognitive patterns of models, effectively distinguishing between superficial behavioral mimicry and genuine mechanistic similarity. We also provide theoretical analysis indicating that ABLE constitutes a Lipschitz-continuous mapping, thereby supporting the stability of the representation.

While extracting gradient-based features incurs a computational cost, this process represents a one-time investment that yields reusable embeddings. Once computed, ABLE facilitates efficient downstream applications, including lineage tracking, model routing, and performance estimation. We hope this work offers a valuable perspective for mapping and understanding the evolving landscape of LLMs.

Limitations

Computational Efficiency. Computing ABLE features requires one forward and one backward pass per sample, which incurs certain time costs. However, ABLE embeddings are computed once per model and can be reused across all downstream tasks. For large-scale model comparison, this one-time extraction cost is lower than running full benchmark evaluations on every model.

Attribution Method. To balance accuracy and efficiency, we only employ Gradient $\times$ Input (GI) as the attribution method. We have not explored more sophisticated techniques such as Integrated Gradient (Kapishnikov et al., 2021; Qi et al., 2020) or SmoothGrad (Smilkov et al., 2017), which may provide higher fidelity attributions at the cost of increased computation.

Closed-Source Models. ABLE requires access to model gradients, making it inapplicable to closed-source APIs. Nevertheless, the concept of mapping LLMs in interpretability space opens avenues for future work using perturbation-based attribution methods (Ivanovs et al., 2021), which could extend ABLE to black-box settings.

Sample Sensitivity. ABLE is computed on a finite set of probing samples, raising the concern that the representation may be sensitive to the specific choice of samples. However, as shown in Appendix D, ABLE embeddings exhibit high stability across different random subsamples of the probing dataset, suggesting that ABLE captures robust structural properties of models rather than being sensitive to individual input samples.

Alignment Granularity. Our character-level alignment uniformly distributes token attributions across characters, which may lose subword-level information. While this simple approach proves effective for the English text benchmarks evaluated in this work, alternative schemes such as frequency-weighted character assignment or byte-level alignment may be more suitable for code or multilingual inputs. We leave systematic exploration of alignment strategies to future work.

References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2017. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2019. Gradient-based attribution methods. In *Explainable AI: Interpreting, explaining and visualizing deep learning*, pages 169–191. Springer.

arc53. 2023. Docsgpt-7b-mistral. https://huggingface.co/Arc53/docsgpt-7b-mistral.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Kristian González Barman, Nathan Wood, and Pawel Pawlowski. 2024. Beyond transparency and explainability: on the need for adequate and contextualized user guidelines for llm use. *Ethics and Information Technology*, 26(3):47.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard (2023-2024). https://huggingface.co/spaces/open-llm-leaderboard-old/open_llm_leaderboard.

Pengzhou Cheng, Zongru Wu, Tianjie Ju, Wei Du, and Zhuosheng Zhang Gongshen Liu. 2024. Transferring backdoors between large language models by knowledge distillation. *arXiv preprint arXiv:2408.09878*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. 2022. Language models are general-purpose interfaces. *arXiv preprint arXiv:2206.06336*.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Eliahu Horwitz, Nitzan Kurer, Jonathan Kahana, Liel Amar, and Yedid Hoshen. 2025. We should chart an atlas of all the world's models. *arXiv preprint arXiv:2503.10633*.

Maksims Ivanovs, Roberts Kadikis, and Kaspars Ozols. 2021. Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognition Letters*, 150:228–234.

Shashank Mohan Jain. 2022. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pages 51–67. Springer.

William B Johnson, Joram Lindenstrauss, and 1 others. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1.

Andrei Kapishnikov, Subhashini Venugopalan, Besim Avci, Ben Wedin, Michael Terry, and Tolga Bolukbasi. 2021. Guided integrated gradients: An adaptive path method for removing noise. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5050–5058.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and 1 others. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR.

Ramaravind Kommiya Mothilal, Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2021. Towards unifying feature attribution and counterfactual explanations: Different means to the same end. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 652–663.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Frederick Liu and Besim Avci. 2019. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6274–6283, Florence, Italy. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Marios Mattheakis, Pavlos Protopapas, David Sondak, Marco Di Giovanni, and Efthimios Kaxiras. 2019. Physical symmetries embedded in neural networks. *arXiv preprint arXiv:1904.08991*.

Gary C McDonald. 2009. Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):93–100.

Peter S Menell. 1988. An analysis of the scope of copyright protection for application programs. *Stan. L. Rev.*, 41:1045.

mlabonne. 2024. Neuralhermes-2.5-mistral-7b. https://huggingface.co/mlabonne/neuralHermes-2.5-mistral-7B.

Aidar Myrzakhan, Sondos Mahmoud Bsharat, and Zhiqiang Shen. 2024. Open-llm-leaderboard: From multi-choice to open-style questions for llms evaluation, benchmark, and arena. *arXiv preprint arXiv:2406.07545*.

Giang Nguyen, Daeyoung Kim, and Anh Nguyen. 2021. The effectiveness of feature attribution methods and its correlation with automatic evaluation scores. *Advances in Neural Information Processing Systems*, 34:26422–26436.

Ian E Nielsen, Dimah Dera, Ghulam Rasool, Ravi P Ramachandran, and Nidhal Carla Bouaynaya. 2022. Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *IEEE Signal Processing Magazine*, 39(4):73–84.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2024. Llms know more than they show: On the intrinsic representation of llm hallucinations. *arXiv preprint arXiv:2410.02707*.

Momose Oyama, Hiroaki Yamagiwa, Yusuke Takase, and Hidetoshi Shimodaira. 2025. Mapping 1,000+ language models via the log-likelihood vector. *arXiv preprint arXiv:2502.16173*.

Zhongang Qi, Saeed Khorram, and Fuxin Li. 2020. Visualizing deep networks by optimizing with integrated gradients. In *AAAI*, volume 34, pages 11890–11898.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Denis Rothman. 2022. *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, Hugging Face, and OpenAI's GPT-3, ChatGPT, and GPT-4*. Packt Publishing Ltd.

Jungwoo Ryoo, Syed Rizvi, William Aiken, and John Kissell. 2013. Cloud security auditing: challenges and emerging approaches. *IEEE Security & Privacy*, 12(6):68–74.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. 2021. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278.

Shuo Shao, Yiming Li, Hongwei Yao, Yiling He, Zhan Qin, and Kui Ren. 2024. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. *arXiv preprint arXiv:2405.04825*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. 2020. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Qwen Team and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3).

Tenyx. 2024. Tenyxchat-7b-v1. https://huggingface.co/tenyx/TenyxChat-7B-v1.

Reconstructing Phylogenetic Trees. 1987. The neighbor-joining method: a new method for. *Mol Biol Evol*, 4(4):406–425.

Imran Ullah. 2024. Medchat3.5. https://huggingface.co/Imran1/MedChat3.5.

Jorge Vallego. 2024. H4rmoniousanthea. https://huggingface.co/neovalle/H4rmoniousanthea.

MA Van Wyk, M Bekker, XL Richards, and KJ Nixon. 2023. Protect your prompts: Protocols for ip protection in llm applications. *arXiv preprint arXiv:2306.06297*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yongjie Wang, Tong Zhang, Xu Guo, and Zhiqi Shen. 2024. Gradient based feature attribution in explainable ai: A technical review. *arXiv preprint arXiv:2403.10415*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhou Yang, Jieke Shi, Prem Devanbu, and David Lo. 2024. Ecosystem of large language models for code. *ACM Transactions on Software Engineering and Methodology*.

Nicolas Yax, Pierre-Yves Oudeyer, and Stefano Palminteri. 2024. Phylolm: Inferring the phylogeny of large language models and predicting their performances in benchmarks. *arXiv preprint arXiv:2404.04671*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Zhixue Zhao and Boxuan Shan. 2024. Reagent: A model-agnostic feature attribution method for generative language models. *arXiv preprint arXiv:2402.00794*.

Xinyu Zhou, Delong Chen, Samuel Cahyawijaya, Xufeng Duan, and Zhenguang G Cai. 2024. Linguistic minimal pairs elicit linguistic similarity in large language models. *arXiv preprint arXiv:2409.12435*.

Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. 2022. Do feature attribution methods correctly attribute features? In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9623–9633.

Sally Zhu, Ahmed Ahmed, Rohith Kuditipudi, and Percy Liang. 2025. Independence tests for language models. *arXiv preprint arXiv:2502.12292*.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. 2024. Embedllm: Learning compact representations of large language models. *arXiv preprint arXiv:2410.02223*.

A Theoretical Analysis

In this section, we provide a theoretical justification that ABLE constitutes a valid and mechanism-aware embedding of large language models. Our analysis abstracts away implementation details and focuses on the relationship between model parameters and attribution-based representations. Section A.1 defines ABLE. Section A.2 establishes that ABLE is a stable feature map induced by model parameters. Section A.3 proves that the low-dimensional embedding preserves inter-model distances via random projection. Section A.4 provides finite-sample concentration guarantees.

A.1 Preliminaries and Definitions

Let a language model be parameterized by $\theta \in \mathbb{R}^P$. For an input representation $x \in \mathbb{R}^d$ and an output class y, denote the log-probability function by

$$s_\theta^{(y)}(x) \triangleq \log p_\theta(y \mid x),$$

which we assume to be differentiable with respect to x almost everywhere.

Attribution-based sensitivity. For each input x, we define the attribution score as the inner product between the input and its gradient:

$$a_\theta(x) \triangleq \langle x, \nabla_x s_\theta(x) \rangle \in \mathbb{R},$$

where $s_\theta(x)$ denotes the log-probability function stacking $s_\theta^{(y)}(x)$ over output classes.

Model representation. Let $\phi : \mathbb{R}^d \to \mathbb{R}^D$ be a deterministic aggregation function. Given a probe distribution $\mathcal{D}$ over inputs, we define the (pre-projection) ABLE representation as

$$\Phi(\theta) \triangleq \mathbb{E}_{x\sim\mathcal{D}}\big[\phi(a_\theta(x))\big] \in \mathbb{R}^D.$$

Finally, the ABLE embedding is obtained via a random projection

$$\Psi(\theta) \triangleq \mathbf{R}\,\Phi(\theta) \in \mathbb{R}^K,$$

where $\mathbf{R} \in \mathbb{R}^{K\times D}$ is a Johnson–Lindenstrauss random matrix.

A.2 The Parameter Stability of ABLE

The objective of this section is to demonstrate the parameter stability of ABLE, meaning that the embedding varies continuously with changes in model parameters. Our main line of reasoning is to first prove that ABLE is a Lipschitz continuous mapping under standard regularity assumptions, and subsequently demonstrate that Transformer-based LLMs satisfy these assumptions.

Theorem A.1 (Parameter Stability of ABLE)**.** *Assume the following regularity conditions hold:*

1. *The gradient field of the log-probability function, $\nabla_x s_\theta(x)$, is M-Lipschitz continuous in θ: $\|\nabla_x s_\theta(x) - \nabla_x s_{\theta'}(x)\|_2 \le M\|\theta - \theta'\|_2$.*
2. *For all x in the support of $\mathcal{D}$, $s_\theta(x)$ is differentiable with respect to x almost everywhere.*
3. *The aggregation function ϕ is L_ϕ-Lipschitz, and inputs are bounded on average: $\mathbb{E}_{x\sim\mathcal{D}}\|x\|_2 \le B$.*

Then the ABLE mapping $\Phi : \theta \mapsto \Phi(\theta)$ is Lipschitz continuous:

$$\|\Phi(\theta) - \Phi(\theta')\|_2 \le L\,\|\theta - \theta'\|_2,$$

where $L = BML_\phi$.

Proof. Using the Cauchy-Schwarz inequality and the assumption on $\nabla_x s_\theta(x)$:

$$\begin{aligned}|a_\theta(x) - a_{\theta'}(x)| &= |\langle x, \nabla_x s_\theta(x) - \nabla_x s_{\theta'}(x)\rangle| \\ &\le \|x\|_2\|\nabla_x s_\theta(x) - \nabla_x s_{\theta'}(x)\|_2 \\ &\le \|x\|_2 M\|\theta - \theta'\|_2.\end{aligned}$$

Applying the Lipschitz continuity of ϕ and taking expectation over $x \sim \mathcal{D}$:

$$\begin{aligned}&\|\Phi(\theta) - \Phi(\theta')\|_2 \\ &\le \mathbb{E}_{x\sim\mathcal{D}}[L_\phi|a_\theta(x) - a_{\theta'}(x)|] \\ &\le L_\phi\mathbb{E}_{x\sim\mathcal{D}}[\|x\|_2]\cdot M\cdot\|\theta - \theta'\|_2 \\ &\le BML_\phi\|\theta - \theta'\|_2.\end{aligned}$$

Theorem A.1 provides the formal guarantee for ABLE's parameter stability. The Lipschitz inequality $\|\Phi(\theta) - \Phi(\theta')\|_2 \le L\|\theta - \theta'\|_2$ ensures that the mapping does not diverge: finite differences

in model parameters result in strictly bounded differences in the embedding space. This property is crucial for a reliable representation, as it guarantees that models with similar internal mechanisms (proximal parameters) are mapped to nearby points in the vector space, preventing chaotic behavior where minor weight variations could lead to disparate embeddings.

The validity of Theorem A.1 relies on three assumptions. Assumptions 2 and 3 are satisfied by design in our setting:

- **Assumption 2 (Differentiability):** While text is discrete, the model operates on continuous word embeddings. Standard embedding lookup layers and subsequent smooth transformations ensure differentiability with respect to the input representation x almost everywhere.

- **Assumption 3 (Boundedness):** The aggregation function ϕ is linear (averaging) and thus 1-Lipschitz. Input embeddings belong to a finite vocabulary set $\mathcal{V}$, so $\|x\|_2$ is strictly bounded by $B = \max_{v \in \mathcal{V}} \|v\|_2$.

The remaining Assumption 1, which requires the model's gradient field to be Lipschitz continuous with respect to its parameters, is the most nontrivial condition. We now demonstrate that this assumption holds for Transformer-based Large Language Models under standard architectural constraints.

Proposition A.1 (Regularity of Transformer Gradient Fields)**.** *Consider a Transformer-based language model with the following properties:*

1. *Activation functions (e.g., GELU, Swish) have bounded first and second derivatives: $|\sigma'(z)| \leq L_\sigma$ and $|\sigma''(z)| \leq L_{\sigma'}$ for all z.*

2. *Token embeddings are drawn from a finite vocabulary with bounded norms: $\|x\|_2 \leq B_x$.*

3. *The network has finite depth L and bounded weight matrices: $\|W_l\|_2 \leq B_W$ for all layers l.*

Then the gradient field $\nabla_x s_\theta(x)$ is M-Lipschitz continuous in θ for some constant $M > 0$.

Proof. We proceed by analyzing the Lipschitz dependence layer by layer.

Step 1: Linear layer. For $y = Wx$, the input gradient is $\nabla_x y = W^T$. For two parameter configurations W, W':

$$\begin{aligned}\|\nabla_x y_W - \nabla_x y_{W'}\|_2 &= \|W^T - W'^T\|_2 \\ &= \|W - W'\|_2,\end{aligned}$$

which is 1-Lipschitz in W.

Step 2: Layer with activation. For $y = \sigma(Wx)$, the chain rule gives $\nabla_x y = W^T \cdot \mathrm{diag}(\sigma'(Wx))$. When W changes to W':

$$\begin{aligned}&\|\nabla_x y_W - \nabla_x y_{W'}\|_2 \\ &\leq \|W^T - W'^T\|_2 \cdot \|\mathrm{diag}(\sigma'(Wx))\|_2 \\ &\quad + \|W'^T\|_2 \cdot \|\sigma'(Wx) - \sigma'(W'x)\|_2 \\ &\leq L_\sigma \|W - W'\|_2 \\ &\quad + B_W L_{\sigma'} \|x\|_2 \|W - W'\|_2 \\ &\leq (L_\sigma + B_W L_{\sigma'} B_x) \|W - W'\|_2.\end{aligned}$$

Step 3: Multi-layer composition. For an L-layer network $f = f_L \circ \cdots \circ f_1$, the chain rule yields $\nabla_x f = \prod_{l=1}^{L} J_l$, where J_l is the Jacobian of layer l. Since each J_l is Lipschitz in θ_l with bounded spectral norm (due to bounded weights and the smoothness of softmax attention), perturbation analysis shows that the product $\nabla_x f$ remains Lipschitz in θ.

For Transformer architectures specifically, self-attention layers use softmax, whose Jacobian satisfies $\|\partial \mathrm{softmax}/\partial z\|_2 \leq 1$. Combined with bounded query, key, and value projections, each attention layer contributes a bounded Lipschitz factor. Aggregating across all L layers yields the global constant M.

Consequently, ABLE constitutes a stable feature map as implied by its Lipschitz continuity, where small perturbations in model parameters result in bounded changes in the representation space. It is worth noting that this guarantee fundamentally relies on differentiability; architectures employing non-differentiable components (e.g., hard attention) may require alternative theoretical analysis.

A.3 Distance Preservation via Random Projection

We next show that the final ABLE embedding preserves inter-model geometry up to a small distortion.

Theorem A.2 (Johnson–Lindenstrauss Property)**.** *Let $\mathcal{F} = \{\theta_1, \ldots, \theta_m\}$ be a finite set of models and*

let $0 < \varepsilon < 1$. If the projection dimension satisfies $K = O(\varepsilon^{-2} \log m)$, then with high probability, for all $i, j \in \{1, \ldots, m\}$, the projected distance $\|\Psi(\theta_i) - \Psi(\theta_j)\|_2$ lies within $(1 \pm \varepsilon)$ of the original distance $\|\Phi(\theta_i) - \Phi(\theta_j)\|_2$.

Proof. Let $\Delta_{ij} = \Phi(\theta_i) - \Phi(\theta_j)$ be the difference vector between any two models. Only the linear projection $\Psi(\theta) = \mathbf{R}\Phi(\theta)$ comprises random variables, where entries of $\mathbf{R} \in \mathbb{R}^{K \times D}$ are i.i.d. Gaussian $\mathcal{N}(0, 1/K)$.

Consider a single pair with difference vector $v = \Delta_{ij}$. We are interested in the distribution of $\|\mathbf{R}v\|^2$. Let R_k be the k-th row of $\mathbf{R}$. Then the k-th component of the projected vector is $y_k = \langle R_k, v \rangle$. Since $R_k \sim \mathcal{N}(0, \frac{1}{K} I)$, the linear combination y_k is essentially a univariate Gaussian:

$$y_k \sim \mathcal{N}\left(0, \frac{\|v\|^2}{K}\right).$$

Consequently, the squared norm of the projection is:

$$\|\mathbf{R}v\|^2 = \sum_{k=1}^{K} y_k^2 = \frac{\|v\|^2}{K} \sum_{k=1}^{K} Z_k^2,$$

where $Z_k \sim \mathcal{N}(0, 1)$ are standard Gaussian variables. The sum $X = \sum_{k=1}^{K} Z_k^2$ follows a Chi-squared distribution with K degrees of freedom, denoted as χ_K^2.

We use standard concentration bounds for the Chi-squared distribution (derived from the Moment Generating Function bound). For any $\epsilon \in (0, 1)$:

$$\mathbb{P}(X \geq K(1 + \epsilon)) \leq \exp\left(-\frac{K}{4}\epsilon^2\right),$$
$$\mathbb{P}(X \leq K(1 - \epsilon)) \leq \exp\left(-\frac{K}{4}\epsilon^2\right).$$

Combining these, the probability that the squared length is distorted by more than ϵ is:

$$\begin{aligned}\mathbb{P}\Big(\big|\|\mathbf{R}v\|^2 - \|v\|^2\big| > \epsilon\|v\|^2\Big) &= \mathbb{P}\left(\left|\frac{X}{K} - 1\right| > \epsilon\right) \\ &\leq 2\exp\left(-\frac{K\epsilon^2}{4}\right).\end{aligned}$$

We apply this to the set of all pairwise differences $\mathcal{V} = \{\Delta_{ij} \mid 1 \leq i < j \leq m\}$, which has cardinality $|\mathcal{V}| = \binom{m}{2} < m^2$. Using the union bound:

$$\begin{aligned}P_{\text{fail}} &= \mathbb{P}\Big(\exists i, j : \big|\|\Psi(\theta_i) - \Psi(\theta_j)\|^2 - \|\Delta_{ij}\|^2\big| \\ &\qquad > \epsilon\|\Delta_{ij}\|^2\Big) \\ &\leq \sum_{i<j} \mathbb{P}\left(\big|\|\mathbf{R}\Delta_{ij}\|^2 - \|\Delta_{ij}\|^2\big| > \epsilon\|\Delta_{ij}\|^2\right) \\ &\leq m^2 \cdot 2\exp\left(-\frac{K\epsilon^2}{4}\right).\end{aligned}$$

To ensure $P_{\text{fail}} \leq \delta$, we require:

$$2m^2 \exp\left(-\frac{K\epsilon^2}{4}\right) \leq \delta \implies K \geq \frac{4}{\epsilon^2} \ln \frac{2m^2}{\delta}.$$

Thus, choosing $K = O(\epsilon^{-2} \log m)$ suffices to preserve all pairwise distances with high probability.

We next show that the final ABLE embedding preserves inter-model geometry up to a small distortion.

Theorem A.3 (Johnson–Lindenstrauss Property)**.** *Let $\mathcal{F} = \{\theta_1, \ldots, \theta_m\}$ be a finite set of models and let $0 < \varepsilon < 1$. If the projection dimension satisfies $K = O(\varepsilon^{-2} \log m)$, then with high probability, for all $i, j \in \{1, \ldots, m\}$, the projected distance $\|\Psi(\theta_i) - \Psi(\theta_j)\|_2$ lies within $(1 \pm \varepsilon)$ of the original distance $\|\Phi(\theta_i) - \Phi(\theta_j)\|_2$.*

Consequence. Theorem A.3 implies that ABLE preserves pairwise distances between models up to a controlled distortion, enabling reliable comparison, clustering, and retrieval in a low-dimensional embedding space.

A.4 Finite-Sample Concentration

In practice, the ABLE representation is computed from N i.i.d. samples rather than the population expectation. We now establish that the empirical estimate concentrates around its expectation, bridging the gap between population-level theory and finite-sample implementation.

Theorem A.4 (Finite-Sample Guarantee)**.** *Let $\hat{\Phi}_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} \phi(a_\theta(x_i))$ be the empirical ABLE representation based on N i.i.d. samples from $\mathcal{D}$. Assume that $\|\phi(a_\theta(x))\|_\infty \leq B_\phi$ for all x in the support of $\mathcal{D}$. Then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$:*

$$\|\hat{\Phi}_N(\theta) - \Phi(\theta)\|_2 \leq B_\phi \sqrt{\frac{2D \log(2D/\delta)}{N}},$$

where D is the dimension of the pre-projection representation.

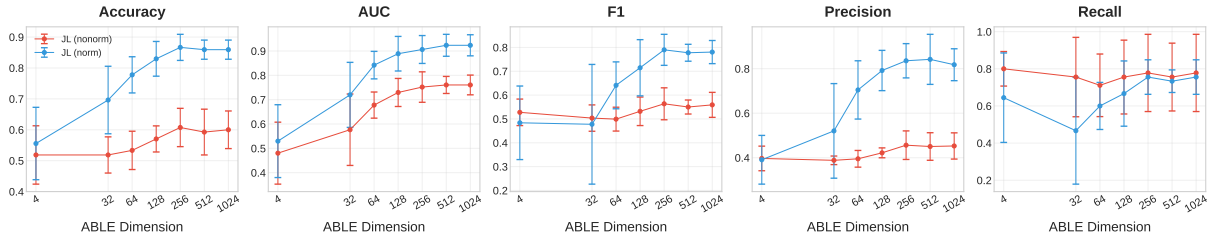

Figure 7: Effect of ABLE embedding dimension on relation prediction performance. The blue curve corresponds to applying L2 normalization to each model's feature vector prior to Johnson-Lindenstrauss projection, while the red curve represents the unnormalized variant. Error bars indicate standard deviation computed over 5 independent runs.

Proof. For each coordinate $j \in \{1, \ldots, D\}$, the empirical mean $[\hat{\Phi}_N(\theta)]_j = \frac{1}{N}\sum_{i=1}^{N}[\phi(a_\theta(x_i))]_j$ is an average of N i.i.d. bounded random variables with $|[\phi(a_\theta(x))]_j| \le B_\phi$. By Hoeffding's inequality:

$$\mathbb{P}\big(|[\hat{\Phi}_N(\theta)]_j - [\Phi(\theta)]_j| > \epsilon\big) \le 2\exp\Big(-\frac{N\epsilon^2}{2B_\phi^2}\Big).$$

Setting the right-hand side to δ/D and applying a union bound over all D coordinates, we obtain that with probability at least $1-\delta$, $|[\hat{\Phi}_N(\theta)]_j - [\Phi(\theta)]_j| \le B_\phi\sqrt{\frac{2\log(2D/\delta)}{N}}$ for all j. The ℓ_2 bound follows from $\|\cdot\|_2 \le \sqrt{D}\|\cdot\|_\infty$.

Consequence. Theorem A.4 quantifies the gap between the theoretical population-level representation $\Phi(\theta)$ and its finite-sample estimate $\hat{\Phi}_N(\theta)$. The bound scales as $O(\sqrt{D\log D/N})$, indicating that the estimation error diminishes with more samples. This theoretical guarantee is empirically substantiated in Appendix D. There, we calculate pairwise ABLE distances between models using two disjoint subsets of the probing data. The strong correlation observed between these two sets of distances indicates that the finite sample size is sufficient to achieve stable representations, thereby validating the convergence predicted by the concentration bound.

B Ablation Study of ABLE Representation Dimensionality

This ablation study examines how the ABLE embedding dimension d affects representation quality. We vary d across {4, 32, 64, 128, 256, 512, 1024} and evaluate each setting on the relation prediction task described in §4.3. We report Accuracy, AUC, F1, Precision, and Recall as evaluation metrics.

As shown in Fig. 7, representation quality improves as d increases and converges around $d = 256$. Very low dimensions (e.g., $d = 4$) yield poor performance because excessive compression discards structural information from the original high-dimensional space. Conversely, dimensions beyond 256 offer only marginal gains while incurring higher computational and storage costs. This trade-off between quality and efficiency motivates our choice of $d = 256$ for all experiments.

We also compare two variants: with and without L2 normalization of each model's feature vector prior to JL projection. Normalization rescales each feature vector to unit length, i.e., $\mathbf{a}'_m = \mathbf{a}_m/\|\mathbf{a}_m\|_2$. As shown in Fig. 7, the normalized variant (blue) generally outperforms the unnormalized variant (red) across most metrics. This suggests that normalization balances the scale across dimensions, enabling the distance metric to better capture directional differences between models.

C Empirical Validation via Model Interpolation

The theoretical analysis in Appendix A establishes that ABLE is a Lipschitz-continuous mapping from model parameters to the embedding space: models with similar parameters should yield similar ABLE embeddings. To empirically validate this correspondence on real nonlinear Transformers, we conduct a model interpolation experiment.

We select three Llama-2-7B variants sharing the same architecture as anchor models: the base model (W_0), Llama-2-7b-chat-hf (W_1), and Vicuna-7b-v1.5 (W_2). We construct 25 merged models via linear weight interpolation:

$$W_{\alpha,\beta} = (1-\alpha-\beta)W_0 + \alpha W_1 + \beta W_2,$$

where $\alpha, \beta \in \{0, 0.25, 0.5, 0.75, 1.0\}$. In weight space, we reconstruct the 2D geometric structure using inner products among the three anchor models; in ABLE space, we apply Principal Component

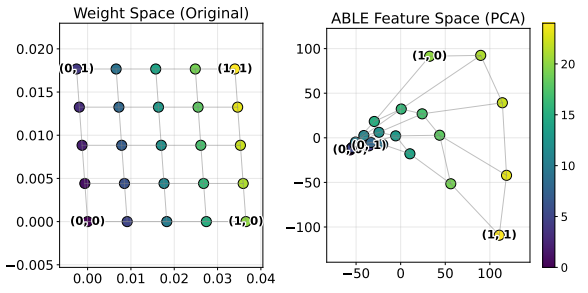

Figure 8: Comparison of weight space and ABLE feature space. Left: weight space reconstructed via inner products. Right: ABLE feature space after PCA reduction. ABLE space preserves the topological structure of weight space but exhibits non-linear distortion. Notably, (0,0) and (0,1) are very close in ABLE space, indicating that Vicuna and the base model exhibit similar behavior, consistent with Vicuna's relatively small fine-tuning magnitude.

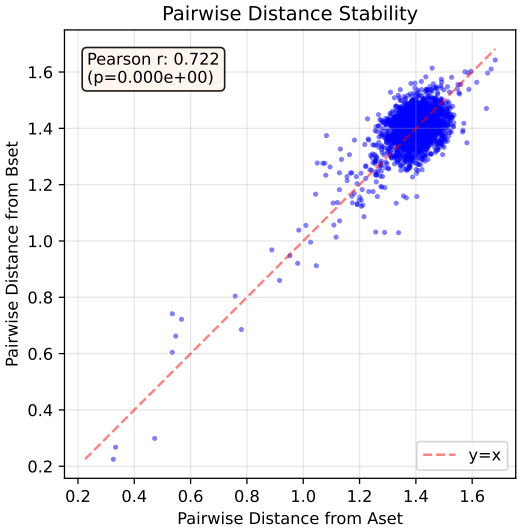

Figure 9: Robustness test for ABLE embeddings. Each point corresponds to a model pair, with x-axis showing the pairwise distance computed from Subset A and y-axis from Subset B.

Analysis (PCA) to the low-dimensional ABLE features for 2D visualization.

As shown in Fig. 8, a notable observation is that (0,0) and (0,1) are very close in ABLE space, while (0,0) and (1,0) are much farther apart. This indicates that Vicuna (W_2) exhibits higher behavioral similarity to the base model than Chat (W_1) does. This observation is consistent with the weight space structure: inner products computed directly from model weights show $\|W_2 - W_0\|^2 \approx 0.00032$, whereas $\|W_1 - W_0\|^2 \approx 0.00134$. Vicuna's fine-tuning magnitude is approximately one quarter that of Chat, demonstrating that ABLE distance reflects the similarity structure in parameter space.

Examining the overall geometry of both spaces, weight space exhibits a regular parallelogram grid reflecting the geometry of linear interpolation. Although ABLE feature space displays irregular distortion, it crucially preserves the topological structure consistent with weight space: (1) adjacent models remain neighbors in both spaces; (2) grid edges do not intersect, indicating that local neighborhood structure is preserved; (3) color gradients of model indices show similar transition patterns across both spaces.

This result validates the theoretical prediction at two levels. The topological preservation confirms the continuity of the ABLE mapping: models close in parameter space remain close in ABLE space. The geometric distortion reveals non-linear effects present in Transformers, indicating that ABLE sensitivity to parameter changes varies across different regions. This is consistent with the Lipschitz continuity established in Appendix A: ABLE guarantees bounded propagation of model similarity without requiring strict linear correspondence.

D Robustness to Sample Selection

To evaluate the robustness of ABLE to probing sample selection, we conduct a Mantel test. Specifically, we randomly partition the probing dataset into two non-overlapping subsets of equal size, and independently compute ABLE embeddings for the same set of models on each subset. We then compute the Euclidean distance between all pairs of models based on embeddings from each subset, yielding two sets of pairwise distances.

As shown in Fig. 9, the two sets of distances exhibit a strong positive correlation (Pearson $r = 0.722$, $p < 0.001$), with points clustering tightly around the $y = x$ diagonal. This result indicates that the inter-model distance structure captured by ABLE embeddings remains consistent across different probing samples, demonstrating that ABLE captures intrinsic behavioral characteristics of models rather than patterns dependent on specific probing inputs.

E Model List

Table 4 lists all $M = 239$ models included in this work, sorted alphabetically.

Table 4: Complete list of models included in this study.

Model Name	Size	Model Type	Downloads
01-ai/Yi-34B-Chat	34B	Llama	30,397
01-ai/Yi-6B	6B	Llama	16,054
01-ai/Yi-6B-200K	6B	Llama	18,155
abocide/Qwen2.5-7B-Instruct-R1-forfinance	7B	Qwen2	529
AdaptLLM/medicine-chat	6.57B	Llama	1,244
AdaptLLM/medicine-LLM	6.57B	Llama	168
AdaptLLM/medicine-LLM-13B	13B	Llama	34
allenai/tulu-2-dpo-70b	70B	Llama	2,543
Arc53/docsgpt-7b-mistral	7B	Mistral	89
augmxnt/shisa-base-7b-v1	7B	Mistral	1,041
bardsai/jaskier-7b-dpo-v5.6	7B	Mistral	159
berkeley-nest/Starling-LM-7B-alpha	7B	Mistral	1,493
bigcode/octocoder	15.5B	Other	148
bigscience/bloom-1b1	1.1B	Bloom	6,749
bigscience/bloom-1b7	1.7B	Bloom	28,490
bigscience/bloom-3b	3B	Bloom	8,904
bigscience/bloom-560m	0.56B	Bloom	112,394
bigscience/bloom-7b1	7B	Bloom	11,136
bigscience/bloomz-3b	3B	Bloom	4,235
bigscience/bloomz-560m	0.56B	Bloom	846,321
bigscience/bloomz-7b1	7B	Bloom	5,299
Biomimicry-AI/ANIMA-Nectar-v2	6.57B	Mistral	975
BioMistral/BioMistral-7B	7B	Mistral	120,331
BioMistral/BioMistral-7B-DARE	7B	Mistral	1,904
bxod/Llama-3.2-1B-Instruct-uz	1B	Llama	35
CausalLM/34b-beta	34B	Llama	8,349
cerebras/Cerebras-GPT-1.3B	1.3B	GPT-2	1,266
cerebras/Cerebras-GPT-111M	0.11B	GPT-2	4,694
cerebras/Cerebras-GPT-2.7B	2.7B	GPT-2	977
cerebras/Cerebras-GPT-256M	0.26B	GPT-2	1,185
cerebras/Cerebras-GPT-590M	0.59B	GPT-2	1,088
cerebras/Cerebras-GPT-6.7B	6.7B	GPT-2	943
cloudyu/Mixtral_11Bx2_MoE_19B	19B	Mixtral	920
codefuse-ai/CodeFuse-DeepSeek-33B	33B	Llama	127
codellama/CodeLlama-13b-Instruct-hf	13B	Llama	19,108
codellama/CodeLlama-34b-Instruct-hf	34B	Llama	20,472
codellama/CodeLlama-7b-hf	7B	Llama	53,172
cognitivecomputations/yayi2-30b-llama	30B	Llama	52
continuedev/instinct	4.86B	Qwen2	241
ConvexAI/Luminex-34B-v0.1	34B	Llama	7,565
ConvexAI/Luminex-34B-v0.2	34B	Llama	7,640
CorticalStack/pastiche-crown-clown-7b-dare-dpo	7B	Mistral	71
CultriX/NeuralTrix-bf16	6.57B	Mistral	74
databricks/dolly-v2-12b	11.58B	GPT-NeoX	3,047
databricks/dolly-v2-3b	3B	GPT-NeoX	-
databricks/dolly-v2-7b	7B	GPT-NeoX	-
davanstrien/query-gen	8B	Llama	56
DeepHat/DeepHat-V1-7B	7B	Qwen2	1,599
deepseek-ai/deepseek-coder-1.3b-base	1.3B	Llama	21,029
deepseek-ai/deepseek-coder-6.7b-instruct	6.7B	Llama	44,962
deepseek-ai/deepseek-llm-67b-chat	67B	Llama	1,815
deepseek-ai/deepseek-math-7b-instruct	7B	Llama	5,002
dfurman/HermesBagel-34B-v0.1	34B	Llama	96
EleutherAI/gpt-neo-1.3B	1.3B	GPT-Neo	28,899
EleutherAI/gpt-neo-125m	0.12B	GPT-Neo	111,607
EleutherAI/gpt-neo-2.7B	2.7B	GPT-Neo	16,759
EleutherAI/llemma_34b	34B	Llama	232
EleutherAI/llemma_7b	7B	Llama	820
EleutherAI/pythia-1.4b	1.4B	GPT-NeoX	18,921
EleutherAI/pythia-1.4b-deduped	1.4B	GPT-NeoX	8,043
EleutherAI/pythia-12b	12B	GPT-NeoX	15,377
EleutherAI/pythia-160m	0.16B	GPT-NeoX	89,343
EleutherAI/pythia-1b-deduped	1B	GPT-NeoX	9,688

continued on next page

continued from previous page

Model Name	Size	Model Type	Downloads
EleutherAI/pythia-2.8b	2.8B	GPT-NeoX	23,704
EleutherAI/pythia-2.8b-deduped	2.8B	GPT-NeoX	7,199
EleutherAI/pythia-410m	0.41B	GPT-NeoX	43,070
EleutherAI/pythia-6.9b	6.9B	GPT-NeoX	17,578
EleutherAI/pythia-70m	0.07B	GPT-NeoX	137,323
eren23/ogno-monarch-jaskier-merge-7b-OH-PREF-DPO	7B	Mistral	71
facebook/opt-1.3b	1.3B	OPT	351,633
facebook/opt-125m	0.12B	OPT	4,242,012
facebook/opt-13b	13B	OPT	9,457
facebook/opt-2.7b	2.7B	OPT	14,711
facebook/opt-350m	0.35B	OPT	100,491
facebook/opt-6.7b	6.7B	OPT	16,057
fblgit/UNA-SimpleSmaug-34b-v1beta	34B	Llama	7,579
FelixChao/llama2-13b-math1.2	13B	Llama	1,093
FelixChao/Scorpio-7B	7B	Mistral	58
FelixChao/vicuna-7B-chemical	7B	Llama	1,086
FelixChao/vicuna-7B-physics	7B	Llama	1,089
golaxy/gowizardlm	6.72B	Llama	950
google/codegemma-1.1-7b-it	7B	Gemma	150
google/codegemma-2b	2B	Gemma	2,143
google/codegemma-7b	7B	Gemma	2,006
google/gemma-2b	2B	Gemma	180,941
google/gemma-2b-it	2B	Gemma	59,882
google/gemma-7b	7B	Gemma	54,233
google/gemma-7b-it	7B	Gemma	108,191
Harshvir/Llama-2-7B-physics	7B	Llama	1,024
HuggingFaceH4/zephyr-7b-alpha	7B	Mistral	1,817
HuggingFaceH4/zephyr-7b-beta	7B	Mistral	75,649
ibivibiv/alpaca-dragon-72b-v1	72B	Llama	877
Imran1/MedChat3.5	7B	Mistral	37
inflatebot/MN-12B-Mag-Mell-R1	12B	Mistral	429
Intel/neural-chat-7b-v3-3	7B	Mistral	31,866
janhq/Jan-v1-4B	4B	Qwen3	1,350
janhq/Jan-v1-edge	1.72B	Qwen3	51
jiawei-ucas/Qwen-2.5-7B-ConsistentChat	7B	Qwen2	16
kamrr/llama-3-8b_dolly_lora	8B	Llama	44
kevin009/llamaRAGdrama	6.57B	Mistral	809
kloodia/alpaca	8B	Llama	82
kloodia/lora-8b-alpaca-french	8B	Llama	87
kloodia/lora-8b-bio	8B	Llama	101
kloodia/lora-8b-code	8B	Llama	133
kloodia/lora-8b-math	8B	Llama	93
kloodia/lora-8b-medic	8B	Llama	109
kloodia/lora-8b-physic	8B	Llama	123
kyujinpy/Sakura-SOLRCA-Math-Instruct-DPO-v1	9.79B	Llama	1,008
LLM360/Amber	7B	Llama	3,443
lmsys/vicuna-13b-v1.5	13B	Llama	116,184
lmsys/vicuna-13b-v1.5-16k	13B	Llama	10,289
lmsys/vicuna-33b-v1.3	33B	Llama	1,403
lmsys/vicuna-7b-v1.1	7B	Llama	2,310
lmsys/vicuna-7b-v1.5	7B	Llama	162,278
lmsys/vicuna-7b-v1.5-16k	7B	Llama	4,445
MaziyarPanahi/WizardLM-Math-70B-v0.1	70B	Llama	76
meta-llama/Llama-2-13b-chat-hf	13B	Llama	209,634
meta-llama/Llama-2-70b-chat-hf	70B	Llama	8,026
meta-llama/Llama-2-7b-chat-hf	7B	Llama	325,422
meta-llama/Llama-2-7b-hf	7B	Llama	599,490
meta-llama/Llama-3.1-8B-Instruct	8B	Llama	10,639,638
meta-llama/LlamaGuard-7b	7B	Llama	1,720
meta-llama/Meta-Llama-3-70B	70B	Llama	480,747
meta-llama/Meta-Llama-3-70B-Instruct	70B	Llama	56,883
meta-llama/Meta-Llama-3-8B	8B	Llama	2,236,547
meta-llama/Meta-Llama-3-8B-Instruct	8B	Llama	1,531,726
meta-llama/Meta-Llama-Guard-2-8B	8B	Llama	15,469

continued on next page

continued from previous page

Model Name	Size	Model Type	Downloads
meta-math/MetaMath-Llemma-7B	7B	Llama	1,033
meta-math/MetaMath-Mistral-7B	7B	Mistral	2,201
microsoft/MediPhi-Instruct	3.72B	Phi-3	2,007
microsoft/Orca-2-13b	13B	Llama	10,627
microsoft/Orca-2-7b	7B	Llama	9,125
microsoft/phi-1_5	1.5B	Phi	46,747
microsoft/phi-2	2.65B	Phi	1,071,489
microsoft/Phi-3.5-mini-instruct	3.72B	Phi-3	335,634
mistralai/Mistral-7B-Instruct-v0.1	7B	Mistral	491,519
mistralai/Mistral-7B-v0.1	7B	Mistral	355,407
mlabonne/AlphaMonarch-7B	7B	Mistral	12,378
mlabonne/NeuralHermes-2.5-Mistral-7B	7B	Mistral	105
mosaicml/mpt-30b-instruct	30B	MPT	2,646
mosaicml/mpt-7b	7B	MPT	-
mosaicml/mpt-7b-8k	7B	MPT	-
mosaicml/mpt-7b-8k-chat	7B	MPT	-
mosaicml/mpt-7b-chat	7B	MPT	81,083
mosaicml/mpt-7b-instruct	7B	MPT	-
Neko-Institute-of-Science/metharme-7b	7B	Llama	1,061
Neko-Institute-of-Science/pygmalion-7b	7B	Llama	1,102
neovalle/H4rmoniousAnthea	7B	Mistral	87
Nexusflow/Starling-LM-7B-beta	7B	Mistral	1,434
nomic-ai/gpt4all-13b-snoozy	13B	Llama	1,012
NousResearch/Hermes-4-14B	14B	Qwen3	5,100
NousResearch/Nous-Hermes-13b	13B	Llama	1,330
NousResearch/Nous-Hermes-2-Yi-34B	34B	Llama	7,907
NousResearch/Nous-Hermes-llama-2-7b	7B	Llama	1,253
OpenAssistant/oasst-sft-4-pythia-12b-epoch-3.5	12B	GPT-NeoX	1,519
OpenBuddy/openbuddy-codellama2-34b-v11.1-bf16	34B	Llama	1,143
openchat/openchat-3.5-0106	6.57B	Mistral	11,448
openchat/openchat_3.5	6.57B	Mistral	2,622
openlm-research/open_llama_7b	7B	Llama	18,239
pbevan11/llama-3-8b-ocr-correction	8B	Llama	64
PharMolix/BioMedGPT-LM-7B	7B	Llama	578
Plaban81/Moe-4x7b-math-reason-code	7B	Mixtral	67
prithivMLmods/rStar-Coder-Qwen3-0.6B	0.6B	Qwen3	21
project-baize/baize-v2-13b	13B	Llama	1,379
Q-bert/Optimus-7B	7B	Mistral	1,106
Qwen/Qwen1.5-0.5B	0.5B	Qwen2	46,541
Qwen/Qwen1.5-0.5B-Chat	0.5B	Qwen2	57,690
Qwen/Qwen1.5-1.8B	1.8B	Qwen2	17,565
Qwen/Qwen1.5-1.8B-Chat	1.8B	Qwen2	89,028
Qwen/Qwen1.5-14B	14B	Qwen2	50,314
Qwen/Qwen1.5-14B-Chat	14B	Qwen2	15,409
Qwen/Qwen1.5-32B	32B	Qwen2	11,415
Qwen/Qwen1.5-32B-Chat	32B	Qwen2	38,786
Qwen/Qwen1.5-4B	4B	Qwen2	13,977
Qwen/Qwen1.5-4B-Chat	4B	Qwen2	17,665
Qwen/Qwen1.5-7B	7B	Qwen2	70,483
Qwen/Qwen1.5-7B-Chat	7B	Qwen2	16,435
Qwen/Qwen2-0.5B	0.5B	Qwen2	329,782
Qwen/Qwen2-1.5B	1.5B	Qwen2	212,957
Qwen/Qwen2.5-0.5B-Instruct	0.5B	Qwen2	2,456,001
Qwen/Qwen2.5-1.5B	1.5B	Qwen2	504,513
Qwen/Qwen2.5-1.5B-Instruct	1.5B	Qwen2	4,990,149
Qwen/Qwen2.5-7B	7B	Qwen2	821,385
Qwen/Qwen2.5-7B-Instruct	7B	Qwen2	6,379,230
Qwen/Qwen2.5-Coder-7B	7B	Qwen2	91,843
Qwen/Qwen2.5-Coder-7B-Instruct	7B	Qwen2	553,942
Qwen/Qwen3-0.6B	0.6B	Qwen3	8,292,645
Qwen/Qwen3-0.6B-Base	0.6B	Qwen3	161,046
Qwen/Qwen3-1.7B	1.7B	Qwen3	5,473,707
Qwen/Qwen3-14B	14B	Qwen3	686,547
Qwen/Qwen3-4B	4B	Qwen3	3,966,909
Qwen/Qwen3-4B-Instruct-2507	4B	Qwen3	4,166,985

continued on next page

continued from previous page

Model Name	Size	Model Type	Downloads
Qwen/Qwen3-4B-Thinking-2507	4B	Qwen3	501,267
Qwen/Qwen3-8B	8B	Qwen3	4,376,885
rishiraj/CatPPT-base	6.57B	Mistral	3,740
rubenamtz0/llama-3-8b-lora-law2entity	8B	Llama	47
sail/Sailor-7B	7B	Qwen2	100
scb10x/typhoon-7b	7B	Mistral	36,854
SciPhi/SciPhi-Mistral-7B-32k	7B	Mistral	1,098
SciPhi/SciPhi-Self-RAG-Mistral-7B-32k	7B	Mistral	1,241
shleeeee/mistral-ko-tech-science-v1	6.57B	Mistral	13
stabilityai/stable-code-3b	3B	StableLM	3,617
stabilityai/stablelm-2-1_6b	6B	StableLM	1,829
stabilityai/stablelm-3b-4e1t	3B	StableLM	18,150
stabilityai/stablelm-base-alpha-3b	3B	GPT-NeoX	2,084
stabilityai/stablelm-base-alpha-7b	7B	GPT-NeoX	1,980
stabilityai/stablelm-tuned-alpha-3b	3B	GPT-NeoX	2,124
stabilityai/stablelm-tuned-alpha-7b	7B	GPT-NeoX	2,393
stabilityai/stablelm-zephyr-3b	3B	StableLM	9,110
SUSTech/SUS-Chat-34B	34B	Llama	998
teknium/OpenHermes-2-Mistral-7B	7B	Mistral	402
teknium/OpenHermes-2.5-Mistral-7B	7B	Mistral	202,048
tenyx/TenyxChat-7B-v1	7B	Mistral	889
Tesslate/UIGEN-X-4B-0729	4B	Qwen3	30
Tesslate/WEBGEN-4B-Preview	4B	Qwen3	55
TheBloke/koala-13B-HF	13B	Llama	1,469
TheBloke/tulu-30B-fp16	30B	Llama	1,174
theprint/TiTan-Qwen2.5-0.5B	0.5B	Qwen2	-
TigerResearch/tigerbot-13b-base	13B	Llama	54
TigerResearch/tigerbot-7b-base-v1	7B	Bloom	18
TigerResearch/tigerbot-7b-base-v2	7B	Bloom	39
TigerResearch/tigerbot-7b-sft-v1	7B	Bloom	47
TigerResearch/tigerbot-7b-sft-v2	7B	Bloom	32
tiiuae/falcon-40b-instruct	40B	Falcon	50,635
tiiuae/falcon-7b	7B	Falcon	64,169
tiiuae/falcon-7b-instruct	7B	Falcon	52,800
tiiuae/falcon-rw-1b	1B	Falcon	6,477
tomg-group-umd/DynaGuard-8B	8B	Qwen3	124
upstage/SOLAR-10.7B-Instruct-v1.0	10.7B	Llama	28,257
venetis/llama3-8b-hermes-sandals-100	8B	Llama	52
VinitT/Sanskrit-llama	8B	Llama	64
Vortex5/Lunar-Nexus-12B	12B	Mistral	19
Vortex5/Moonlit-Shadow-12B	12B	Mistral	14
WizardLM/WizardLM-70B-V1.0	70B	Llama	17,901
worldboss/llama-3-8b-axolotl-fine-tune-qlora	8B	Llama	70
Writer/palmyra-med-20b	20B	GPT-2	1,153
yahma/llama-7b-hf	7B	Llama	4,983
yam-peleg/Experiment26-7B	7B	Mistral	173
zhengr/MixTAO-7Bx2-MoE-v8.1	6.57B	Mixtral	19,428