Representing LLMs in Prompt Semantic Task Space

Idan Kashani and Avi Mendelson and Yaniv Nemcovsky
Technion - Israel Institute of Technology
Department of Computer Science
idan-kashani@cs.technion.ac.il

Abstract

Large language models (LLMs) achieve impressive results over various tasks, and ever-expanding public repositories contain an abundance of pre-trained models. Therefore, identifying the best-performing LLM for a given task is a significant challenge. Previous works have suggested learning LLM representations to address this. However, these approaches present limited scalability and require costly retraining to encompass additional models and datasets. Moreover, the produced representation utilizes distinct spaces that cannot be easily interpreted. This work presents an efficient, training-free approach to representing LLMs as linear operators within the prompts' semantic task space, thus providing a highly interpretable representation of the models' application. Our method utilizes closed-form computation of geometrical properties and ensures exceptional scalability and real-time adaptability to dynamically expanding repositories. We demonstrate our approach on success prediction and model selection tasks, achieving competitive or state-of-the-art results with notable performance in out-of-sample scenarios.

1 Introduction

LLMs have recently emerged with remarkable capabilities, revolutionizing multiple diverse fields, including medical information processing (Zheng et al., 2024; Jin et al., 2024), software engineering (Etsenake and Nagappan, 2024; Jiang et al., 2024; Jimenez et al., 2024), and scientific research (Frieder et al., 2024; Zhang et al., 2024a; Li et al., 2024b). Moreover, open-source LLMs are widespread and have fueled a rapidly growing ecosystem of publicly-available models and benchmarks used to assess their capabilities. Currently, the most prominent platform hosting these resources is **Hugging Face** (Wolf et al., 2019), serving as a centralized repository for nearly one million pre-trained models and hundreds of thousands of benchmarks. Such open-access repositories facilitates LLMs' large-scale deployment and promotes their continuous innovation across diverse applications.

The demand for LLM-based applications is ever-increasing, and new, diverse models with improved capabilities are constantly being produced. However, this rapid growth and diversity present a substantial challenge: *identifying the best-performing models*. This challenge entails recognizing the most suitable model for producing a response to specific queries, or on average over queries in a given dataset. Hand-selecting these models would require careful analysis and clear annotation of their properties, which are not widely available.

A common approach is to rely on benchmark results to select suitable models (Chang et al., 2023). A **benchmark** consists of a dataset and an evaluation metric designed to assess specific model capabilities, such as domain expertise (Hendrycks et al., 2021; Yu et al., 2024), reasoning skills (Parmar et al., 2023; Veličković et al., 2022; Talmor et al., 2019), agentic abilities (Liu et al., 2024), or safety (Zhang et al., 2024c; Li et al., 2024a; Chao et al., 2024). While such benchmarking provides an initial assessment of models, it often involves a complex and time-consuming process to select a suitable model for a given prompt.

A primary challenge with conventional benchmarks lies in their reporting of aggregated performance scores, derived from a static corpus of domain-specific prompts. Such global metrics can be unreliable for guiding selection over queries that substantially diverge from the evaluation samples. Moreover, model proficiency often varies considerably across different prompts even within the same domain (Zhuo et al., 2024; Miller, 2024). Typical benchmark outputs tend to obscure these instance-level performance nuances, thereby failing to provide the granular detail essential for precise, query-specific model selection. Furthermore, the per-

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 8578–8597

ceived capabilities of LLMs can be skewed by inherent biases and sensitivities within benchmarks, resulting from particular prompt structures (Cao et al., 2024; Pezeshkpour and Hruschka, 2024) or the methodologies behind leaderboard rankings (Perlitz et al., 2024; Alzahrani et al., 2024). Beyond these limitations in granularity, relying on isolated benchmarks does not assess the collective insights available from the broader landscape of evaluation tools.

Performance prediction methods aim to predict models' performance on unseen prompts and tasks based on prior information. While such methods similarly utilize benchmarks, they differ in explicitly estimating models' performance over given queries, and aim to be applicable in scenarios where obtaining queries' labels is expensive or impractical. These approaches are particularly relevant to out-of-sample (OOS) settings, where the queries originate from datasets entirely unseen by the performance prediction estimators. Such settings introduce an additional layer of complexity, as generalization to unknown datasets is highly challenging.

A recent work has suggested the approach of LLM embeddings for performance prediction and subsequent model selection (Zhuang et al., 2025). This approach aims to represent both LLMs and prompts in a joint space. The performance estimation is then computed via the corresponding embeddings of the model and query. However, current approaches utilize a distinct representation space that depends on their training data, thereby requiring costly retraining to include additional benchmark results. In this context, we denote the setting of real-time success prediction and subsequent model selection as aiming to apply to newly published models with minimal delay.

Our work builds on the promising direction of using LLM embeddings for performance prediction. We aim to represent LLMs as linear operators within the prompts' semantic task space, thus providing a highly interpretable representation of the models' application. We consider models' application on queries as semantic-space translations from input to output. We then utilize a closed-form computation to represent the difference between the model's induced and the desired translation, which produces the corresponding label. Our approach is training-free, requires negligible computational resources, and can be adapted to additional benchmarks in real-time. Moreover, we produce task-oriented embeddings with clear semantic interpretations relevant to diverse downstream tasks and OOS scenarios. Below we outline our main **contributions**:

1. We present a novel approach to represent models directly within the semantic task of prompt embeddings. This direct representation presents a more intuitive and semantically grounded understanding of model-task relationships, enabling a more efficient and interpretable analysis of models' suitability.

2. We utilize our representations for performance prediction, presenting an efficient and dynamically expandable evaluation of models. Our method utilizes closed-form computation, is training-free, and can be seamlessly expanded to additional models and benchmarks with negligible computational cost. Hereby, we enable real-time suitability analysis over the rapidly growing models-benchmarks ecosystem.

3. We evaluate our method on performance prediction and model selection tasks over multiple settings and achieve state-of-the-art or comparable results. Moreover, our semantically grounded approach outperforms all previous baselines on OOS settings, indicating its robustness in diverse real-world scenarios.

2 Related Work

Previous LLM performance prediction works present various settings and approaches. Some works aim to analyze the behavior and performance of given models rather than directly predict their performance over given queries. A common setting discusses the assessment of model performance via per-sample output analysis, where approaches leverage confidence scores (Garg et al., 2022), self-correction capabilities (Jawahar et al., 2024), or transferability estimation (Bao et al., 2019; You et al., 2021; Bassignana et al., 2022). Another seeks to predict the broader capabilities of models through statistical analysis (Papadopoulos et al., 2007), or by deriving scaling laws from pretraining data (Chen et al., 2025). Although such approaches provide important perspectives, they are typically not applicable to the scope of this work.

Another set of methods trains auxiliary models to predict performance, for example, by training an assessor model which uses an LLM's results on fixed set of few reference prompts alongside target prompt intrinsic features to minimize evaluation costs (Pacchiardi et al., 2025), or by applying collaborative filtering to learn latent model and task factors from historical performance metadata (Zhang et al., 2024b; Drori et al., 2019; Zhang et al., 2023). These methods primarily contribute a trained predictive model designed to operate with specific inputs for the LLM in question. Our work, however, has a different underlying framework and is focused on deriving pre-computed, explicit vector representations for each LLM within an established library, using its comprehensive performance profile on source datasets.

The line of research most pertinent to our objective of creating explicit model representations from performance data involves **learning joint embeddings for LLMs and prompts**. EmbedLLM (Zhuang et al., 2025) stands out as a key contribution in this domain. It employs an encoder-decoder architecture to learn informative representations from a large dataset of model-prompt interactions. While this work presents a significant step towards real-time performance prediction and subsequent model selection, it requires costly retraining to encompass additional models and benchmarks. Moreover, EmbedLLM's representations utilize an arbitrary space that lacks semantic grounding.

Our approach builds on the promising direction of using LLM embeddings for performance prediction, but aims to be dynamically expanding while representing LLMs within the prompts' semantic task space.

3 Method

We now detail our approach to creating linear, interpretable, and scalable representations for LLMs. We enable efficient performance prediction and model selection by embedding each LLM as a vector aligned with the prompts it successfully computes.

Formally, our goal is to derive a linear representation $\mathbf{E}(\mathbf{M})_i \in \mathbb{R}^{d_{\text{prompt}}}$ for each LLM $\mathcal{M}_i$ in a given pool $\mathcal{L} = \{\mathcal{M}_i\}_{i=1}^{M}$. This embedding $\mathbf{E}(\mathbf{M})_i$ is conceptualized as a vector in the d_{prompt}-dimensional prompt embedding space. Specifically, $\mathbf{E}(\mathbf{M})_i$ represents the "**success hyperplane normal**", namely a normal to a model-specific hyperplane that ideally separates between prompts where model $\mathcal{M}_i$ succeeds from those where it fails. The

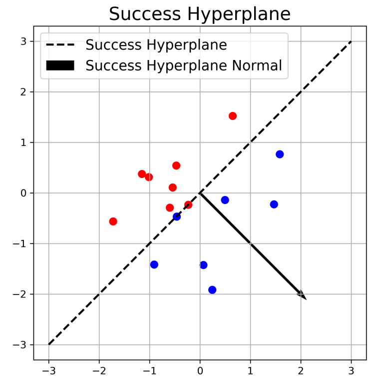

Figure 1: The projection of a prompt embedding $E(p)$ on a model embedding $\mathbf{E}(\mathbf{M})_i$ yields a score predicting the model's success on that prompt.

orientation of $\mathbf{E}(\mathbf{M})_i$ thus signifies a direction in the prompt space associated with success for that particular model. Consequently, the success of model $\mathcal{M}_i$ on a prompt q is estimated by:

$$\hat{\text{Succ}}(\mathcal{M}_i, q) = \mathbf{E}(\mathbf{M})_i \cdot E(q). \quad (1)$$

Here, $E(q) \in \mathbb{R}^{d_{\text{prompt}}}$ is the vector embedding for a given target prompt q, generated using the same pre-trained Sentence Transformer as the source prompts.

Conceptually, an LLM's success on a given prompt is a highly complex function of that prompt's semantic embedding, $f(E(q))$. Our method does not attempt to model f in its entirety. Instead, we seek the best linear operator, represented by the vector $\mathbf{E}(\mathbf{M})_i$, that approximates the *average* outcome of this function with respect to the success-failure dichotomy. This approach is predicated on the well-established property of high-dimensional embedding spaces where semantic relationships can be represented as linear vector operations, a principle first established for word vectors (Mikolov et al., 2013) and since extended to produce robust sentence-level semantic representations (Reimers and Gurevych, 2019). The strong empirical success of our method suggests that this first-order linear approximation is sufficient to capture the most significant variance in model performance, offering a favorable trade-off between model fidelity and the exceptional scalability our approach provides.

Due to the linearity of our approach, the aggregate success score for a model on a benchmark can be efficiently computed by averaging the embeddings of the benchmark's prompts to form a single

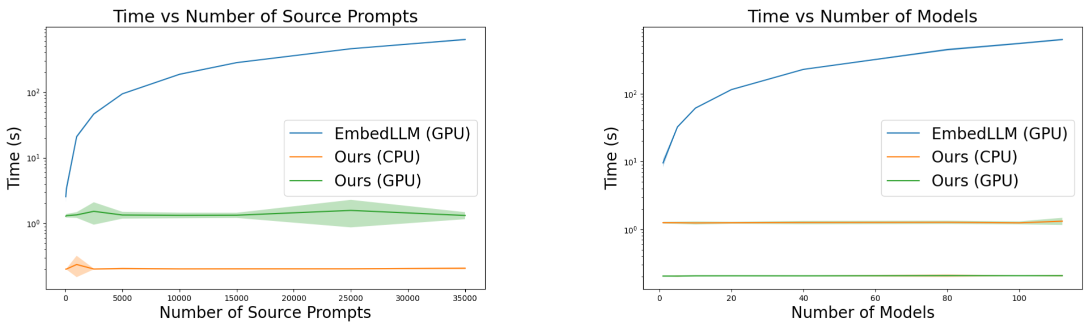

Figure 2: Model embeddings creation time vs. number of prompt samples (left) and models (right), on CPU and GPU (logarithmic scale).

benchmark vector, then taking its dot product with the model's embedding.

3.1 Data and System Formulation

To estimate these model embeddings $\mathbf{E}(\mathbf{M})$, we utilize:

- A set of N source prompts p_j with corresponding ground-truth answers a_j, from $\mathcal{D}_{\text{src}} = \{(p_j, a_j)\}_{j=1}^{N}$.
- The observed performance of each of the M LLMs from pool $\mathcal{L}$ on these prompts.

Prompt Embeddings. Each prompt p_j is transformed into an L^2 normalized vector embedding $E(p_j) \in \mathbb{R}^{d_{\text{prompt}}}$ using a pre-trained **Sentence Transformer**. This type of architecture (e.g., based on (Devlin et al., 2019; Schroff et al., 2015)) is chosen for its efficiency and established ability to capture semantic content relevant for comparing text sequences. These prompt embeddings form the rows of a matrix $\mathbf{D}_{\mathbf{src}} \in \mathbb{R}^{N \times d_{\text{prompt}}}$.

Performance Matrix. Benchmarks results measure a certain property of a model with respect to a dataset. **Success** is determined using an **exact match** criterion between a model $\mathcal{M}_i$'s output for prompt p_j and the target answer a_j. This binary outcome (success/failure) is encoded in a performance matrix $\mathbf{P}_{\mathbf{src}} \in \mathbb{R}^{M \times N}$, where:

$$\mathbf{P}_{\mathbf{src}_{\mathbf{ij}}} = \begin{cases} 1, & \text{if } \mathcal{M}_i(p_j) = a_j \\ -1, & \text{otherwise.} \end{cases}$$

The Linear System. Given our conceptualization of model embeddings (Equation (1)), the relationship $\mathbf{E}(\mathbf{M})_i \cdot E(p_j) \approx \mathbf{P}_{\mathbf{src}_{\mathbf{ij}}}$ should hold for all models and prompts. This can be expressed in matrix form as the linear system we aim to solve for $\mathbf{E}(\mathbf{M})$:

$$\mathbf{E}(\mathbf{M})(\mathbf{D}_{\mathbf{src}})^{\intercal} \approx \mathbf{P}_{\mathbf{src}}. \tag{2}$$

3.2 Computing Linear LLM Representations

We solve the linear system (Equation (2)) for $\mathbf{E}(\mathbf{M})$. Since the matrix $(\mathbf{D}_{\mathbf{src}})^{\intercal}$ (derived from prompt embeddings) is typically non-square and may be non-invertible, we employ its regularized Moore-Penrose pseudoinverse (Moore, 1920; Bjerhammar, 1951; Penrose, 1955; Ben-Israel and Greville, 2003), computed via Singular Value Decomposition (SVD) (Eckart and Young, 1936). The regularization is crucial for stability and to improve generalization to unseen data:

- **Singular Value Thresholding:** Singular values (σ_{ii}) from the SVD of $\mathbf{D}_{\mathbf{src}}$ that fall below a predefined threshold, ε, are effectively set to zero before forming the pseudoinverse components[1]. This mitigates numerical instabilities from near-zero singular values. We found the choice of this threshold (ε) significantly affects results, especially for OOS settings (Appendix A).

- **Tikhonov Regularization:** We apply Tikhonov regularization to smooth the inversion. This is achieved by incorporating the regularization term 2λ with the squared singular values when deriving the effective inverse singular values (Tikhonov, 1943; Hoerl and Kennard, 1970a,b).

[1] Not to be confused with the numerical tolerance threshold $t = \text{machine precision} \cdot \max(N, d_{\text{prompt}}) \cdot \max(\text{diag}(\Sigma))$. In our experiments, $\varepsilon > t$.

The closed-form solution for the model embeddings $\mathbf{E}(\mathbf{M})$ is:

$$\mathbf{E}(\mathbf{M}) = \mathbf{P}_{\text{src}}(\mathbf{D}_{\text{src}}^{+})^{\intercal} = \mathbf{P}_{\text{src}}\mathbf{U}\mathbf{\Sigma}'\mathbf{V}^{\intercal}, \quad (3)$$

where $\mathbf{D}_{\text{src}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\intercal}$ is the SVD of the prompt embedding matrix. The diagonal matrix $\mathbf{\Sigma}'$ is derived from the singular values σ_{ii} in $\mathbf{\Sigma}$, with its elements are given by:

$$\sigma'_{ii} = \begin{cases} 0, & \text{if } \sigma_{ii} < \varepsilon \\ \frac{\sigma_{ii}}{\sigma_{ii}^2 + 2\lambda}, & \text{otherwise} \end{cases} \quad (4)$$
$$\sigma'_{ij} = 0, \qquad \text{for} \quad i \neq j.$$

The resulting model embedding matrix $\mathbf{E}(\mathbf{M})$ is of size $M \times d_{\text{prompt}}$. This linear, training-free computation makes our approach highly efficient and directly interpretable within the prompt's semantic space.

3.3 Scalability Analysis

The primary computational cost is the SVD of $\mathbf{D}_{\text{src}}$ (dimensions $N \times d_{\text{prompt}}$), which has a complexity of $O(Nd_{\text{prompt}}^2)$ and is thus linear in N for a fixed d_{prompt}. The subsequent matrix multiplication to obtain $\mathbf{E}(\mathbf{M})$ is linear in both M and N. This results in significantly better overall scalability compared to EmbedLLM, whose training time typically exhibits much steeper growth with increasing M or N. Further efficiency can be achieved by applying iterative methods to update $(\mathbf{D}_{\text{src}}^{+})^{\intercal}$, such as the Newton-Schulz iteration (Appendix C).

The method also offers theoretical stability for distributed systems. Since the embeddings are based on the semantic space of prompts, adding new prompts to a dataset is expected to induce only minor changes to existing model embeddings. This leads to minimal discrepancies in the embeddings over time across different servers.

4 Experiments

This section presents a comprehensive empirical evaluation of our method over the tasks of real-time **success prediction** and subsequent **model selection**. We first present the experimental setting in Section 4.1, and continue to discuss the results in Section 4.2. In addition, we present an ablation study of our method in supplementary Appendix A. Our evaluation aims to answer three key research questions:

- **(RQ1)** How does our method compare to existing approaches, regarding predicted success and model selection?
- **(RQ2)** How does the scalability of our proposed method compare to previous work?
- **(RQ3)** Does our method present a viable approach to predict success in OOS settings?

4.1 Experimental Setup

4.1.1 Core Evaluation Tasks and Metrics

Success Prediction. This task presents a binary classification problem, i.e., given an LLM and a prompt, will the model successfully complete the task defined by the prompt? The task performance metrics are then:

- **AUC (Area Under the ROC Curve):** Measures the ability to distinguish between success and failure, irrespective of a specific classification threshold.
- **Accuracy:** The fraction of correct success/failure predictions.
- **Benchmark Score Correlation:** The Pearson correlation between our method's estimated success scores for a model across a benchmark's prompts (which can be computed efficiently as presented in Section 3) and the model's actual ground-truth accuracy on that benchmark.

Model Selection. In this task, given sets of models and test prompts, methods aim to accurately rank the models according to their expected success over the prompts. The task performance metrics are then:

- **Accuracy:** The proportion of test prompts for which the best-ranked (selected) model produces a successful response.
- **Recall:** The proportion of "solvable" prompts over which the top-ranked model succeeds. This metric evaluates the selector's ability to choose a successful model for prompts that *at least one* model in the pool can solve. It directly measures how the selected model's performance compares to the best possible outcome on a per-prompt basis

4.1.2 Models & Datasets Environments

Our experiments were conducted on four distinct environments, each comprising specific sets of models, source prompts used for methods' execution, and target prompts used for evaluation. These are derived from two primary sources:

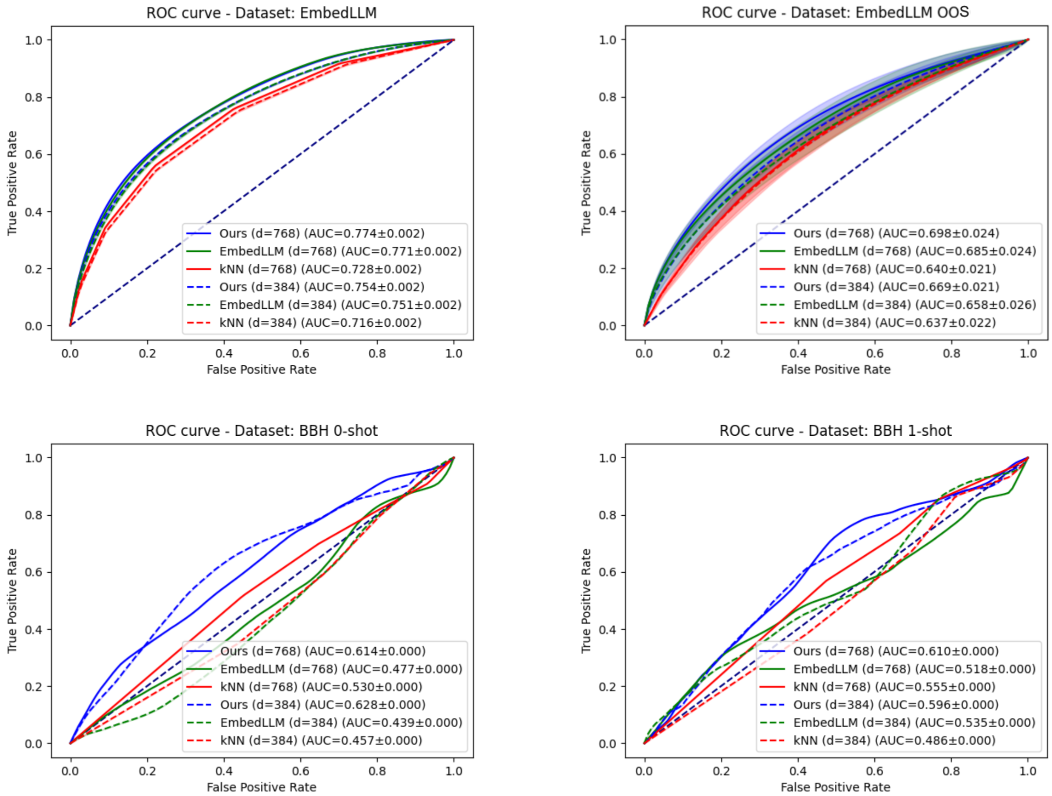

Figure 3: Success Prediction ROC curves describe the true positive rate vs. the false positive rate across thresholds.

1. **EmbedLLM Benchmark Environment**:
   - **Models**: 112 LLMs from the EmbedLLM framework, covering both general-purpose and domain-specific architectures.
   - **Source & Target prompts**: Source and target prompts are randomly sampled in either in-sample or OOS scenarios from over 80 prominent benchmarks, such as MathQA (Amini et al., 2019), SocialQA (Sap et al., 2019), PIQA (Bisk et al., 2020), LogiQA (Liu et al., 2020), ASDiv (Miao et al., 2020), GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), TruthfulQA (Lin et al., 2021), MedMCQA (Pal et al., 2022), and GPQA (Rein et al., 2024). For the EmbedLLM environment, both in-sample and OOS results are the average of 10 independent trials. In each trial, a new random seed is used to sample the source and target prompts from the available benchmarks. The error margins in Table 1 and the shaded regions in the ROC curves of Figure 3 represent the standard deviation across these 10 trials.
   - **In-Sample Scenario**: Test prompts and source prompts are sampled may originate from the same datasets.
   - **Out-of-Sample (OOS) Scenario**: Test datasets are excluded from the datasets library, that is used as source prompts for the method. This scenario represents a more complex generalization task.

2. **LoRA-Finetuned T5 Models Environment**:
   - **Models**: A set of 92 T5-large FLAN encoder-decoder models. These models were finetuned on FLAN v2 datasets (Chung et al., 2024) using the LoRA technique (Hu et al., 2022) by Huang et al. (2024).
   - **Source Datasets**: The 92 FLAN v2 datasets used for the original LoRA training (listed in Appendix D).
   - **Target Datasets (OOS)**: BIG-Bench Hard (BBH) (Suzgun et al., 2023), a suite of 23 challenging tasks for LLMs.
   - **Evaluation Scenarios**: Performance is

evaluated in either OOS zero-shot or OOS one-shot settings. One demonstration prompt is sampled per dataset for the one-shot.

4.1.3 Baselines and Method Configuration

Prompt Embeddings. We utilize two pre-trained Sentence Transformer models to generate L_2 normalized prompt embeddings:

- MiniLM-L6-v2 ($d_{\text{prompt}} = 384$) (Wang et al., 2021)[2]
- MPNet-v2 ($d_{\text{prompt}} = 768$) (Song et al., 2020)[3]

Previous art. Our method is compared against two primary baselines, following the EmbedLLM paper (Zhuang et al., 2025):

1. **k-Nearest Neighbors (kNN):** Configured with $k = 5$. For success prediction on a target prompt, kNN identifies the k nearest prompts in the source data via prompt embedding similarity. The average success rate of the target model on these neighbors is then the predicted score. This evaluation metric is then utilized in model selection by selecting the candidate LLM with the maximal score.
2. **EmbedLLM:** Evaluated using the provided framework.

A comparison with an additional static baseline, "Best Source Performer", is provided in Appendix B.

Our Configuration. We utilize the regularization parameter $\lambda = 1$ in all the compared settings. The optimal singular value threshold ε is then determined via the ablation study presented in Appendix A.

4.2 Experimental Results

4.2.1 Success Prediction

In Table 1 and Figure 3, we present the comparison of our method to previous art over the success prediction task. Our method achieves a better ratio between true positives and false positives compared to previous art. Moreover, it outperforms all previous art over AUC, Accuracy, and Benchmark Score Correlation, substantially so in the OOS scenarios (EmbedLLM (OOS), BBH 0-shot, and BBH 1-shot). This may suggest that our semantically grounded approach effectively captures fundamental model-task alignment and is better suited for generalization to unseen datasets. The high benchmark score correlation further indicates that our lightweight, per-prompt success predictions accurately aggregate to reflect overall benchmark performance.

4.2.2 Model Selection

In Table 1, we present a comparison of our method to previous art over the model selection task. Our method outperforms all previous art in the OOS setting. For the in-sample settings, our results are comparable to the best-performing training-based EmbedLLM approach. Furthermore, as shown in Appendix B (Table 2), our dynamic selection approach outperforms a static "Best Source Performer" baseline, notably so in in-sample settings.

4.2.3 Scalability Evaluation

In Figure 2, we present the computational time comparison of our method and previous art. The computation was executed on `Intel(R) Xeon(R)` CPU and `NVIDIA L40S` GPU. Our approach presents a negligible increase in computation time for an increasing number of models, which aligns with our expected asymptotically linear computation time described in Section 3.3.

5 Discussion

5.1 Conclusions

This work has introduced a novel approach for representing LLMs as linear operators within the prompts' semantic task space. To do so, we consider models' application on queries as semantic-space translations from input to output. We then define the representations as a linear mapping from input to the task's performance metric, and compute them via matrix inversion. The resulting representations then present a semantic interpretation of LLMs' application compared to the task's goal, and are utilized for performance predictions and subsequent model selection.

The suggested approach is training-free and scales to increasing number of models and benchmarks with negligible computational cost. Moreover, as the computation is based on matrix inversion, it can be extended to encompass additional models and benchmarks without recomputing the

[2] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[3] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Table 1: Comparison of *success prediction* and *model selection* across methods and prompt embedding dimension.

			Success Prediction			Model Selection	
Environment	Method	dim	AUC	Accuracy	Benchmark Score Correlation	Accuracy	Recall
EmbedLLM	kNN	384	0.7158 ± 0.0019	0.6855 ± 0.0009	0.7665 ± 0.0139	0.5261 ± 0.0029	0.5762 ± 0.0033
	EmbedLLM	384	0.7509 ± 0.0018	0.7076 ± 0.0013	0.9030 ± 0.0022	$\mathbf{0.6269 \pm 0.0024}$	$\mathbf{0.6867 \pm 0.0030}$
	Ours	384	$\mathbf{0.7538 \pm 0.0019}$	$\mathbf{0.7115 \pm 0.0015}$	$\mathbf{0.9248 \pm 0.0027}$	0.6221 ± 0.0020	0.6814 ± 0.0027
	kNN	768	0.7285 ± 0.0019	0.6937 ± 0.0014	0.7498 ± 0.0123	0.5335 ± 0.0031	0.5844 ± 0.0033
	EmbedLLM	768	0.7714 ± 0.0018	0.7183 ± 0.0014	0.9266 ± 0.0022	$\mathbf{0.6410 \pm 0.0018}$	$\mathbf{0.7022 \pm 0.0025}$
	Ours	768	$\mathbf{0.7736 \pm 0.0018}$	$\mathbf{0.7232 \pm 0.0013}$	$\mathbf{0.9485 \pm 0.0025}$	0.6355 ± 0.0012	0.6961 ± 0.0014
EmbedLLM (OOS)	kNN	384	0.6366 ± 0.0224	0.6205 ± 0.0262	0.6971 ± 0.0531	0.4779 ± 0.0426	0.5198 ± 0.0448
	EmbedLLM	384	0.6580 ± 0.0259	0.6466 ± 0.0266	0.8310 ± 0.0323	0.5667 ± 0.0635	0.6165 ± 0.0688
	Ours	384	$\mathbf{0.6696 \pm 0.0214}$	$\mathbf{0.6480 \pm 0.0242}$	$\mathbf{0.8451 \pm 0.0300}$	$\mathbf{0.5879 \pm 0.0441}$	$\mathbf{0.6393 \pm 0.0413}$
	kNN	768	0.6404 ± 0.0211	0.6230 ± 0.0216	0.6860 ± 0.0311	0.4807 ± 0.0365	0.5229 ± 0.0372
	EmbedLLM	768	0.6848 ± 0.0239	0.6622 ± 0.0247	0.8663 ± 0.0292	0.5694 ± 0.0640	0.6194 ± 0.0682
	Ours	768	$\mathbf{0.6983 \pm 0.0240}$	$\mathbf{0.6694 \pm 0.0153}$	$\mathbf{0.8820 \pm 0.0322}$	$\mathbf{0.5916 \pm 0.0436}$	$\mathbf{0.6435 \pm 0.0435}$
BBH 0-shot (OOS)	kNN	384	0.4573	0.3251	-0.2376	0.2014	0.4766
	EmbedLLM	384	0.4394	0.3297	-0.1408	0.2275	0.5383
	Ours	384	$\mathbf{0.6284}$	$\mathbf{0.4351}$	$\mathbf{0.3734}$	$\mathbf{0.2491}$	$\mathbf{0.5896}$
	kNN	768	0.5301	0.3351	0.1353	0.1971	0.4664
	EmbedLLM	768	0.4769	0.2937	-0.0766	0.2113	0.5002
	Ours	768	$\mathbf{0.6139}$	$\mathbf{0.3838}$	$\mathbf{0.3862}$	$\mathbf{0.2491}$	$\mathbf{0.5896}$
BBH 1-shot (OOS)	kNN	384	0.4858	0.4132	-0.0528	0.3330	0.7012
	EmbedLLM	384	0.5353	0.4347	0.1116	$\mathbf{0.3408}$	$\mathbf{0.7178}$
	Ours	384	$\mathbf{0.6546}$	$\mathbf{0.5034}$	$\mathbf{0.5569}$	0.3405	0.7171
	kNN	768	0.5550	0.4390	0.2757	0.3251	0.6846
	EmbedLLM	768	0.5178	0.3298	-0.0562	0.3274	0.6895
	Ours	768	$\mathbf{0.6097}$	$\mathbf{0.4631}$	$\mathbf{0.3843}$	$\mathbf{0.3381}$	$\mathbf{0.7119}$

pre-existing ones. Hereby, requiring minimal computation to apply to newly published models and benchmarks, which is a crucial property in the rapidly evolving model-benchmark ecosystem. Furthermore, our method achieves state-of-the-art or comparable results over various performance prediction and model selection tasks, and outperforms previous OOS baselines.

Model repositories' continuous and rapid expansion underscores the critical need for scalable solutions. Such solutions must allow users to efficiently estimate LLM properties and select models suitable for their specific purposes and operational constraints. We have demonstrated the efficiency and scalability of our embedding creation process, benefits directly attributable to the simplicity of the underlying linear operations. While our current work focuses on dense performance matrices, the underlying linear algebraic framework is amenable to future extensions, potentially incorporating matrix-completion techniques to handle scenarios with sparser performance data. These characteristics support efficient retrieval and search over extensive collections of models, datasets, and tasks.

Conceptually, we have advanced the understanding of LLM performance through the "success hyperplane normal" lens. Our embedding effectively captures the correlation between models' responses to various prompts and their corresponding measured performance. This semantic representation allows models, inputs, and task success criteria to be considered within a shared semantic framework, offering clearer insights into model-task alignment. Furthermore, maintaining a consistent representation space for models enables the parallel computation of our approach across distributed models' repositories.

This work lays the foundation for a continuous deployment process composed of benchmarking, embedding, and storing the embeddings as informative metadata of the models in the repository. This allows retrieval of LLMs based on predefined properties, thus supporting large-scale, accessible LLMs for practical applications. By utilizing the diversity of pre-existing models and benchmarks, we enable the identification of suitable models while minimizing the effort and environmental impact associated with training new models.

5.2 Future Work

Our approach to embedding LLMs in a task-oriented space opens several promising avenues for future research. A possible direction is to examine properties other than success, such as safety, efficiency, stylistic alignment, and personalization. Models would then be retrieved based on aggregated criteria, which enables the consideration of multiple desired properties. Similarly, we can extend our approach to encompass multiple tasks and

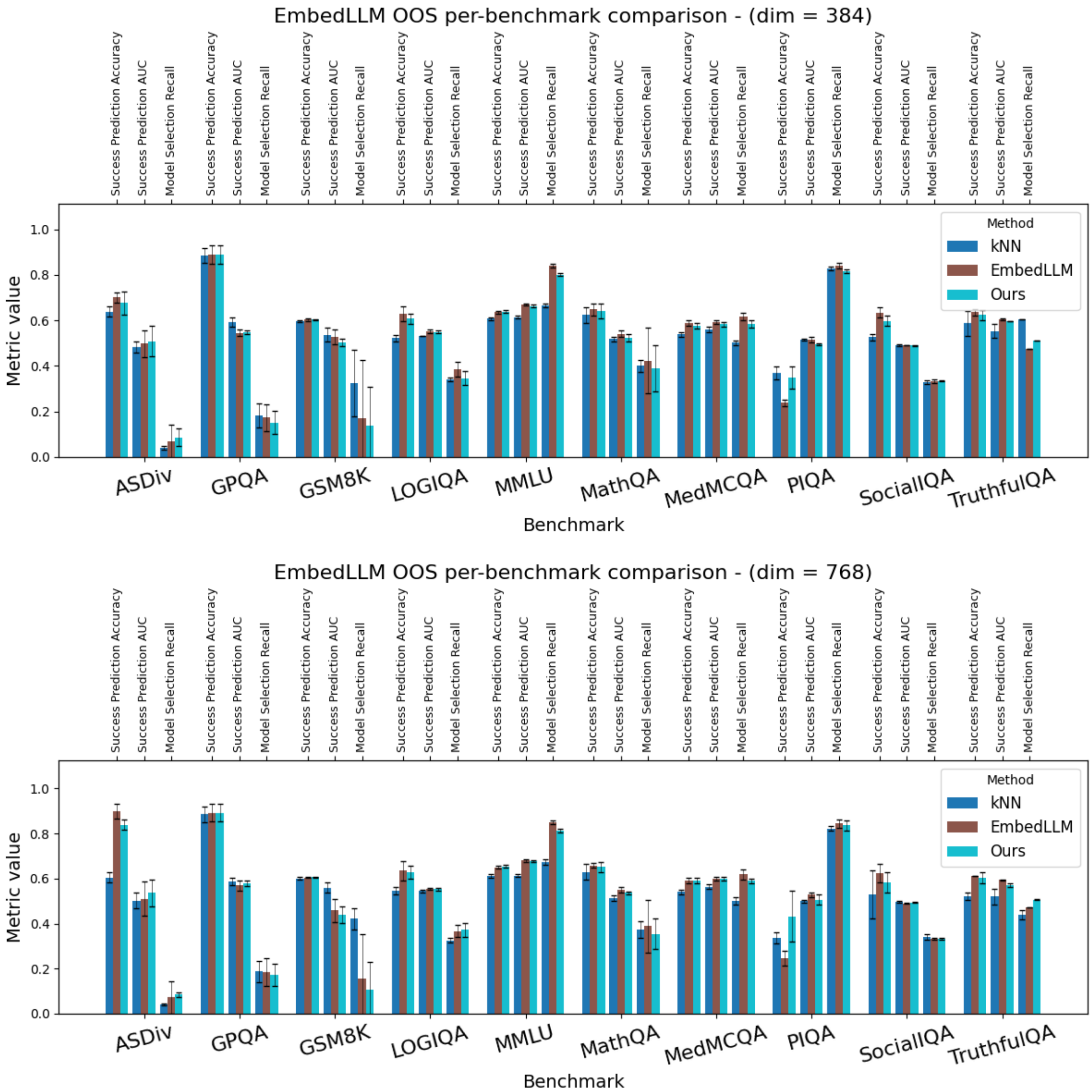

Figure 4: A per-benchmark breakdown of *Success Prediction (Accuracy and AUC)* and *Model Selection (Recall)* in the EmbedLLM OOS environment for embedding dimensions 384 (top) and 768 (bottom). Our training-free method delivers performance competitive with the EmbedLLM baseline at a fraction of the computational cost.

produce representations that better interpret models' applications by considering distinct success metrics and corresponding behaviors. Finally, we propose applying our method to dynamic task routing in multi-agent systems, where its training-free, scalable, and interpretable nature is uniquely suited for embedding the output of one agent to select the next, enabling more adaptive and explainable problem-solving pipelines of specialized LLMs.

6 Limitations

This work proposes representing LLMs as embeddings in semantic task spaces, where the embedding encodes their corresponding performance. The models' embedding then corresponds to successfully answered queries and enables an efficient and explainable search of well-performing models. We can then consider the semantic vectors of models as representing their corresponding semantic translation between inputs and outputs. However, our suggested representation only regards a single task and its corresponding information, which may be insufficient to represent the complex semantic translation of LLMs. Moreover, our representation does not consider the models' architectures or inference complexity in any way, and future work could extend it to also consider model efficiency in addition to performance.

References

Norah Alzahrani, Hisham Alyahya, Yazeed Alnumay, Sultan AlRashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. 2024. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13787–13805, Bangkok, Thailand. Association for Computational Linguistics.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. 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 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. 2019. An information-theoretic approach to transferability in task transfer learning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2309–2313.

Elisa Bassignana, Max Müller-Eberstein, Mike Zhang, and Barbara Plank. 2022. Evidence > intuition: Transferability estimation for encoder selection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4218–4227, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Adi Ben-Israel and Thomas N.E. Greville. 2003. *Generalized Inverses: Theory and Applications*, 2nd edition. Springer, New York.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Arne Bjerhammar. 1951. Application of calculus of matrices to method of least squares: with special reference to geodetic calculations. *Transactions of the Royal Institute of Technology, Stockholm*, 49:1–86.

James R. Bunch and Christopher P. Nielsen. 1978. Updating the singular value decomposition. *Numerische Mathematik*, 31(2):111–129.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models. *Preprint*, arXiv:2307.03109.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. 2025. Scaling laws for predicting downstream performance in LLMs.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Iddo Drori, Lu Liu, Yi Nian, Sharath C. Koorathota, Jie S. Li, Antonio Khalil Moretti, Juliana Freire, and Madeleine Udell. 2019. Automl using metadata language embeddings. *Preprint*, arXiv:1910.03698.

Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Deborah Etsenake and Meiyappan Nagappan. 2024. Understanding the human-llm dynamic: A literature survey of llm use in programming tasks. *Preprint*, arXiv:2410.01026.

Simon Frieder, Julius Berner, Philipp Petersen, and Thomas Lukasiewicz. 2024. Large language models for mathematicians. *Preprint*, arXiv:2312.04556.

Saurabh Garg, Sivaraman Balakrishnan, Zachary Chase Lipton, Behnam Neyshabur, and Hanie Sedghi. 2022. Leveraging unlabeled data to predict out-of-distribution performance. In *International Conference on Learning Representations*.

William W Hager. 1989. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Arthur E. Hoerl and Robert W. Kennard. 1970a. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82.

Arthur E. Hoerl and Robert W. Kennard. 1970b. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*.

Ganesh Jawahar, Muhammad Abdul-Mageed, Laks Lakshmanan, and Dujian Ding. 2024. LLM performance predictors are good initializers for architecture search. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10540–10560, Bangkok, Thailand. Association for Computational Linguistics.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *Preprint*, arXiv:2406.00515.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Qiao Jin, Nicholas Wan, Robert Leaman, Shubo Tian, Zhizheng Wang, Yifan Yang, Zifeng Wang, Guangzhi Xiong, Po-Ting Lai, Qingqing Zhu, Benjamin Hou, Maame Sarfo-Gyamfi, Gongbo Zhang, Aidan Gilson, Balu Bhasuran, Zhe He, Aidong Zhang, Jimeng Sun, Chunhua Weng, and 4 others. 2024. Demystifying large language models for medicine: A primer. *Preprint*, arXiv:2410.18856.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. 2024a. SALAD-bench: A hierarchical and comprehensive safety benchmark for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3923–3954, Bangkok, Thailand. Association for Computational Linguistics.

Sihang Li, Jin Huang, Jiaxi Zhuang, Yaorui Shi, Xiaochen Cai, Mingjun Xu, Xiang Wang, Linfeng Zhang, Guolin Ke, and Hengxing Cai. 2024b. Scilitllm: How to adapt llms for scientific literature understanding. *Preprint*, arXiv:2408.15545.

Stephanie C. Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. In *Annual Meeting of the Association for Computational Linguistics*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3622–3628. International Joint Conferences on Artificial Intelligence Organization. Main track.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

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*.

Evan Miller. 2024. Adding error bars to evals: A statistical approach to language model evaluations. *Preprint*, arXiv:2411.00640.

Eliakim Hastings Moore. 1920. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26(9):394–395.

Lorenzo Pacchiardi, Lucy G Cheke, and Jose Hernandez-Orallo. 2025. 100 instances is all you need: predicting LLM success by testing on a few instances.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Harris Papadopoulos, Volodya Vovk, and Alex Gammerman. 2007. Conformal prediction with neural networks. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, volume 2, pages 388–395.

Mihir Parmar, Neeraj Varshney, Nisarg Patel, Santosh Mashetty, Man Luo, Arindam Mitra, and Chitta Baral. 2023. Logicbench: A benchmark for evaluation of logical reasoning.

Roger Penrose. 1955. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. 2024. Efficient benchmarking (of language models). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2519–2536, Mexico City, Mexico. Association for Computational Linguistics.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

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*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

G. Schulz. 1933. Iterative Berechnung der reziproken Matrix. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 13(1):57–59.

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.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

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.

Andrey N. Tikhonov. 1943. On the stability of inverse problems. In *Doklady Akademii Nauk SSSR*, volume 39, pages 195–198.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. 2022. The CLRS algorithmic reasoning benchmark. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22084–22102. PMLR.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12133–12143. PMLR.

Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, Chunyang Li, Zheyuan Zhang, Yushi Bai, Yantao Liu, Amy Xin, Kaifeng Yun, Linlu GONG, Nianyi Lin, Jianhui Chen, and 16 others. 2024. KoLA: Carefully benchmarking world knowledge of large language models. In *The Twelfth International Conference on Learning Representations*.

Qiang Zhang, Keyang Ding, Tianwen Lyv, Xinda Wang, Qingyu Yin, Yiwen Zhang, Jing Yu, Yuhao Wang, Xiaotong Li, Zhuoyi Xiang, Kehua Feng, Xiang Zhuang, Zeyuan Wang, Ming Qin, Mengyao Zhang, Jinlu Zhang, Jiyu Cui, Tao Huang, Pengju Yan, and 6 others. 2024a. Scientific large language models: A survey on biological & chemical domains. *Preprint*, arXiv:2401.14656.

Qiyuan Zhang, Fuyuan Lyu, Xue Liu, and Chen Ma. 2024b. Collaborative performance prediction for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2576–2596, Miami, Florida, USA. Association for Computational Linguistics.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. Automl-gpt: Automatic machine learning with gpt. *Preprint*, arXiv:2305.02499.

Zhexin Zhang, Leqi Lei, Lindong Wu, Rui Sun, Yongkang Huang, Chong Long, Xiao Liu, Xuanyu Lei, Jie Tang, and Minlie Huang. 2024c. Safety-Bench: Evaluating the safety of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15537–15553, Bangkok, Thailand. Association for Computational Linguistics.

Yanxin Zheng, Wensheng Gan, Zefeng Chen, Zhenlian Qi, Qian Liang, and Philip S. Yu. 2024. Large language models for medicine: A survey. *Preprint*, arXiv:2405.13055.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. 2025. EmbedLLM: Learning compact representations of large language models. In *The Thirteenth International Conference on Learning Representations*.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976, Miami, Florida, USA. Association for Computational Linguistics.

A Finding the optimal epsilon

We present the hyper-parameter tuning of ε for each discussed setting. Increasing ε results in filtering directions, that correspond to lower variance of the prompt embeddings. We can notice increasing epsilon can improve all metrics in the OOS scenario, until at some point it starts to decrease. This can be attributed to the cleaning of noise and the maintenance of dominant singular directions. Another noticeable trend in two of the datasets is that the task of model selection, where false positives are more problematic than false negatives, requires a larger ε, than the one required for success prediction.

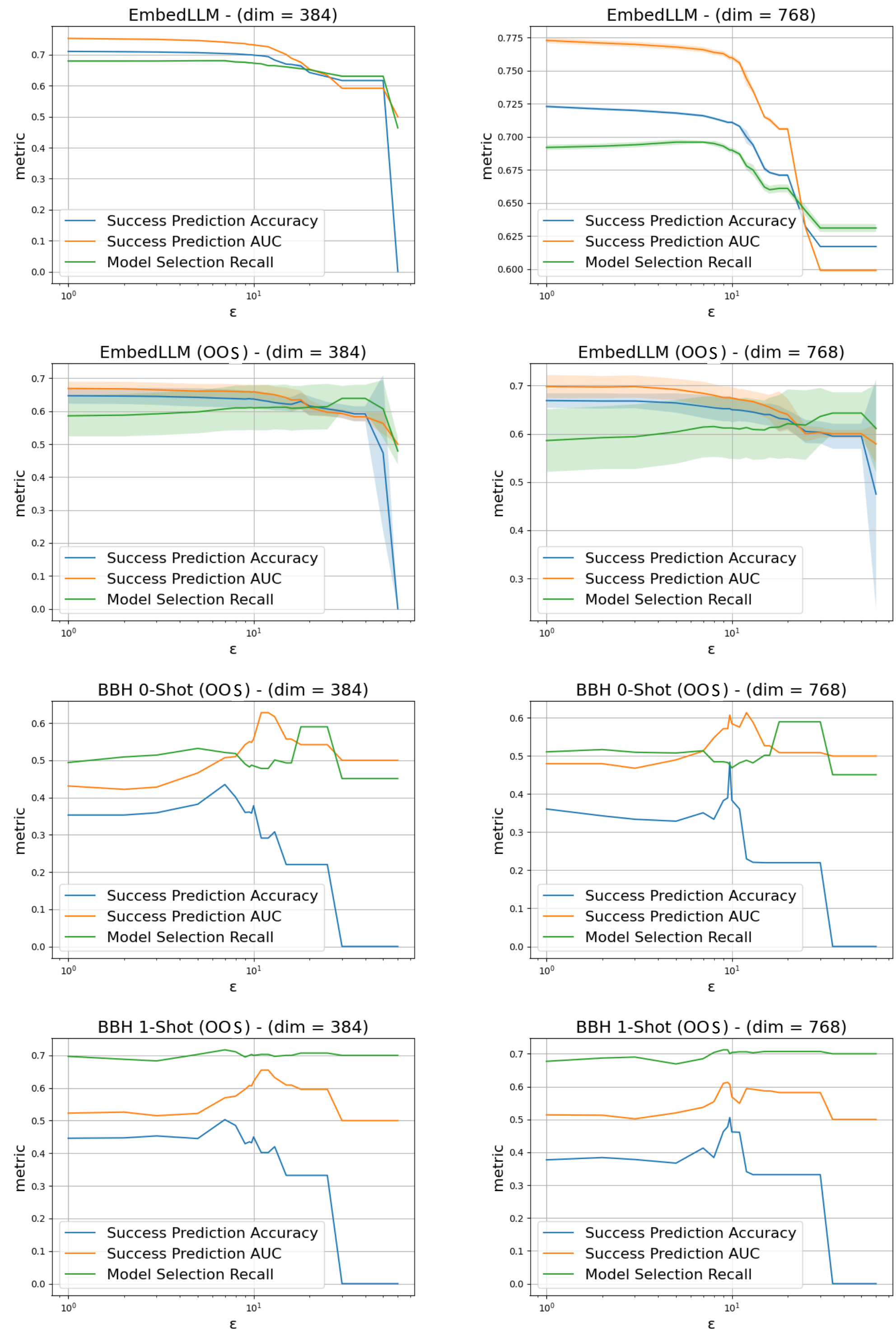

Figure 5: The effect of ε on the performance metrics discussed in our work.

B Comparison with static selection

This section provides a comparative analysis of our dynamic model selection method against a static baseline termed the "*Best Source Performer*" (BSP). The BSP baseline identifies the single LLM from the available pool that achieved the highest overall accuracy across all prompts within the defined source dataset ($\mathcal{D}_{\text{src}}$) for each specific evaluation environment. This pre-selected model is then used uniformly for all test prompts in that environment. This comparison serves to benchmark our per-prompt selection strategy against a strong, globally-optimized static choice based on performance over the known source data.

Table 2 presents the Accuracy and Recall metrics. The results generally show that our dynamic selection approach offers advantages, particularly in in-sample scenarios, while maintaining robust and competitive performance against the BSP baseline in OOS settings. This highlights the value of adaptive, per-prompt model selection.

Table 2: Comparison of our model selection method ('Ours') against a static 'Best Source Performer' baseline. The 'Best Source Performer' is the single model achieving the highest overall accuracy on the entire source prompt library (i.e., $\mathcal{D}_{\text{src}}$ from Section 3.1) for each environment. Performance is reported across different evaluation environments and prompt embedding dimensions for our method.

Environment	Method	dim	Accuracy	Recall
EmbedLLM	Best Source Performer	–	0.5759 ± 0.0020	0.6309 ± 0.0026
	Ours	384	$\underline{0.6221 \pm 0.0020}$	$\underline{0.6814 \pm 0.0027}$
	Ours	768	$\mathbf{0.6355 \pm 0.0012}$	$\mathbf{0.6961 \pm 0.0014}$
EmbedLLM (OOS)	Best Source Performer	–	$\underline{0.5885 \pm 0.0444}$	$\underline{0.6398 \pm 0.0414}$
	Ours	384	0.5879 ± 0.0441	0.6393 ± 0.0413
	Ours	768	$\mathbf{0.5916 \pm 0.0436}$	$\mathbf{0.6435 \pm 0.0435}$
BBH 0-shot (OOS)	Best Source Performer	–	$\mathbf{0.2491}$	$\mathbf{0.5896}$
	Ours	384	$\mathbf{0.2491}$	$\mathbf{0.5896}$
	Ours	768	$\mathbf{0.2491}$	$\mathbf{0.5896}$
BBH 1-shot (OOS)	Best Source Performer	–	0.3357	0.7070
	Ours	384	$\mathbf{0.3405}$	$\mathbf{0.7171}$
	Ours	768	$\underline{0.3381}$	$\underline{0.7119}$

C Algorithms for Incremental Updates

Adding New Models to the Repository

Adding a new model is highly efficient as it does not require recomputing the expensive pseudoinverse of the prompt embedding matrix. Calculating a new model's embedding merely requires *a single matrix-multiplication operation.* The existing pseudoinverse, $(D_{\text{src}}^{+})^{\intercal}$, which encapsulates the structure of the source prompt space, is simply *reused*, only this time with the measured performance of the new model on the source prompts.

Algorithm 1 Incremental Addition of a New Model

1: **Input:**
2: The precomputed pseudoinverse of the source prompt matrix, $(D_{\text{src}}^{+})^{\intercal} \in \mathbb{R}^{N \times d_{\text{prompt}}}$.
3: The performance vector $P_{\text{new}} \in \mathbb{R}^{1 \times N}$ for the new model $\mathcal{M}_{\text{new}}$ on the N source prompts.

4: **Procedure:**
5: Compute the embedding for the new model, $E(\mathcal{M})_{new}$, via a single matrix-vector multiplication:

$$E(\mathcal{M})_{\text{new}} = P_{\text{new}} \cdot (D_{\text{src}}^{+})^{\intercal}$$

6: Append the resulting vector $E(\mathcal{M})_{\text{new}} \in \mathbb{R}^{1 \times d_{\text{prompt}}}$ as a new row to the existing model embedding matrix $\mathbf{E}(\mathbf{M})$.

Computational Complexity: The cost of this operation is dominated by the matrix-vector multiplication, which is $O(N \cdot d_{\text{prompt}})$. Since d_{prompt} is fixed (e.g., 384 or 768), the complexity is linear in the number of source prompts, N. This cost is negligible compared to retraining-based approaches.

Adding New Source Prompts to the Library

Adding new source prompts is more complex than adding models because it alters the source prompt matrix $\mathbf{D_{src}}$, invalidating the precomputed pseudoinverse $(\mathbf{D_{src}}^{+})^{\intercal}$. The most direct approach is to recompute the SVD of the new, larger prompt matrix, an operation with a complexity of $O(N \cdot d_{\text{prompt}}^2)$. For the dataset sizes explored in our scalability experiments (Figure 2), this direct recomputation was already so efficient that it resulted in nearly constant update times.

However, for large-scale systems where N is exceptionally large, an even greater asymptotic efficiency can be achieved by using incremental update methods. One such approach is to compute the pseudoinverse via the normal equations. This involves inverting the square matrix $\mathbf{A} = \mathbf{D_{new}}^{\intercal}\mathbf{D_{new}}$. The Tikhonov regularization already present in our main method (Equation (4)) is equivalent to inverting $\mathbf{A} = \mathbf{D_{new}}^{\intercal}\mathbf{D_{new}} + 2\lambda\mathbf{I}$, which conveniently ensures the matrix is always invertible and well-conditioned. For this task, a classic iterative solver like the Newton-Schulz iteration (Schulz, 1933) can be used.

Algorithm 2 Incremental Update of Model Embeddings via Newton-Schulz

Input:
Old source matrices: $\mathbf{D_{src}} \in \mathbb{R}^{N \times d_{\text{prompt}}}$, $\mathbf{P_{src}} \in \mathbb{R}^{M \times N}$.
Old computed inverse: $\mathbf{A_{src}}^{-1} = (\mathbf{D_{src}}^{\intercal}\mathbf{D_{src}} + 2\lambda\mathbf{I})^{-1} \in \mathbb{R}^{d_{\text{prompt}} \times d_{\text{prompt}}}$.
New data to add: $\mathbf{D_{added}} \in \mathbb{R}^{N_{add} \times d_{\text{prompt}}}$, $\mathbf{P_{added}} \in \mathbb{R}^{M \times N_{add}}$.
Iteration count for refinement, k.

Procedure:
Concatenate matrices to form the new set:
$\mathbf{D_{new}} \leftarrow \begin{bmatrix} \mathbf{D_{src}} \\ \mathbf{D_{added}} \end{bmatrix} \in \mathbb{R}^{(N+N_{add}) \times d_{\text{prompt}}}$
$\mathbf{P_{new}} \leftarrow \begin{bmatrix} \mathbf{P_{src}} & \mathbf{P_{added}} \end{bmatrix} \in \mathbb{R}^{M \times (N+N_{add})}$

Form the new matrix to be inverted:
$\mathbf{A_{new}} \leftarrow \mathbf{D_{new}}^{\intercal}\mathbf{D_{new}} + 2\lambda\mathbf{I}$

Use the previous inverse as a strong initial guess for the new inverse: $\mathbf{X}_0 \leftarrow \mathbf{A_{src}}^{-1}$.
for $i = 0$ to $k-1$ **do**
    $\mathbf{X}_{i+1} \leftarrow \mathbf{X}_i(2\mathbf{I} - \mathbf{A_{new}}\mathbf{X}_i)$ ▷ Refine the inverse using Newton-Schulz iteration
end for
Let the converged inverse be $\mathbf{A_{new}}^{-1} \leftarrow \mathbf{X}_k$.

Compute the new pseudoinverse:
$\mathbf{D_{new}}^{+} \leftarrow \mathbf{A_{new}}^{-1}\mathbf{D_{new}}^{\intercal}$

Compute the final updated model embeddings:
$\mathbf{E}(\mathbf{M})_{\mathbf{new}} \leftarrow \mathbf{P_{new}}(\mathbf{D_{new}}^{+})^{\intercal}$

Output: The updated model embedding matrix, $\mathbf{E}(\mathbf{M})_{\mathbf{new}} \in \mathbb{R}^{M \times d_{\text{prompt}}}$.

Computational Complexity: The dominant cost in Algorithm 2 comes from the Newton-Schulz loop. Each iteration involves matrix multiplications of size ($d_{\text{prompt}} \times d_{\text{prompt}}$), leading to a complexity of $O(k \cdot d_{\text{prompt}}^3)$ for the loop. This is asymptotically more efficient than the full SVD recomputation ($O(N_{\text{new}} \cdot d_{\text{prompt}}^2)$) when the number of prompts N_{new} is significantly larger than the embedding dimension d_{prompt}.
Alternative Methods: In addition to Newton-Schulz, there are other established methods for updating the pseudoinverse that are valid for our use, such as updating the SVD directly via rank-one updates (Bunch and Nielsen, 1978) or using the Sherman-Morrison-Woodbury formula for low-rank updates (Hager, 1989). The existence of these techniques offers additional scalability, confirming that our framework is theoretically well-suited for massive, dynamically expanding systems.

D FLAN v2 Models and Datasets

We present the full list of 92 FLAN v2 datasets.

1. adversarial_qa_dbidaf_based_on
2. adversarial_qa_dbert_answer_the_following_q
3. adversarial_qa_dbidaf_question_context_answer
4. adversarial_qa_dbidaf_tell_what_it_is
5. adversarial_qa_droberta_tell_what_it_is
6. amazon_polarity_User_recommend_this_product
7. anli_r1
8. app_reviews_categorize_rating_using_review
9. bool_q
10. dbpedia_14_given_a_list_of_category_what_does_the_title_belong_to
11. definite_pronoun_resolution
12. dream_baseline
13. dream_read_the_following_conversation_and_answer_the_question
14. drop
15. duorc_ParaphraseRC_answer_question
16. duorc_ParaphraseRC_movie_director
17. duorc_ParaphraseRC_Youtubeing
18. duorc_SelfRC_generate_question_by_answer
19. duorc_SelfRC_Youtubeing
20. duorc_SelfRC_title_generation
21. gem_e2e_nlg
22. gem_web_nlg_en
23. glue_cola
24. glue_mrpc
25. glue_sst2
26. glue_wnli
27. wiki_hop_original_choose_best_object_interrogative_2
28. imdb_reviews_plain_text
29. kilt_tasks_hotpotqa_complex_question
30. lambada

31. math_dataset_algebra_linear_1d
32. sciq_Multiple_Choice_Question_First
33. newsroom
34. ropes_prompt_beginning
35. qasc_is_correct_1
36. qasc_is_correct_2
37. qasc_qa_with_combined_facts_1
38. qasc_qa_with_separated_facts_3
39. qasc_qa_with_separated_facts_5
40. quac
41. quail_context_description_Youtube_text
42. quail_context_Youtube_description_id
43. quail_context_Youtube_description_text
44. quail_context_question_description_answer_id
45. quail_context_question_description_answer_text
46. quail_description_context_Youtube_id
47. quail_no_prompt_text
48. quarel_choose_between
49. quarel_do_not_use
50. quarel_heres_a_story
51. quarel_logic_test
52. quarel_testing_students
53. quartz_having_read_above_passage
54. quartz_read_passage_below_choose
55. quoref_Find_Answer
56. quoref_Found_Context_Online
57. quoref_Guess_Title_For_Context
58. race_high_Select_the_best_answer
59. race_middle_Is_this_the_right_answer
60. race_middle_Select_the_best_answer
61. race_middle_Taking_a_test
62. ropes_prompt_bottom_hint_beginning

63. sciq_Direct_Question_Closed_Book_
64. social_i_qa_Generate_the_question_from_the_answer
65. squad_v1.1
66. squad_v2.0
67. super_glue_wic
68. super_glue_wsc.fixed
69. trec
70. true_case
71. web_questions_get_the_answer
72. wiki_bio_comprehension
73. wiki_bio_guess_person
74. wiki_bio_key_content
75. wiki_bio_who
76. wiki_hop_original_choose_best_object_affirmative_1
77. wiki_hop_original_choose_best_object_interrogative_1
78. wiki_hop_original_generate_subject
79. wiki_qa_automatic_system
80. wiki_qa_found_on_google
81. wiki_qa_Is_This_True_
82. wiki_qa_Jeopardy_style
83. wiki_qa_Topic_Prediction_Answer_Only
84. wiqa_effect_with_label_answer
85. wiqa_what_is_the_final_step_of_the_following_process
86. wiqa_what_might_be_the_last_step_of_the_process
87. wiqa_what_is_the_missing_first_step
88. wmt16_translate_ro-en
89. wiqa_which_of_the_following_is_the_supposed_perturbation
90. wmt16_translate_tr-en
91. word_segment
92. yelp_polarity_reviews