Collaborative Performance Prediction for Large Language Models

Qiyuan Zhang
City University of Hong Kong
qzhang732-c@my.cityu.edu.hk

Fuyuan Lyu[*]
McGill University & MILA
fuyuan.lyu@mail.mcgill.ca

Xue Liu
McGill University
xueliu@cs.mcgill.ca

Chen Ma[*]
City University of Hong Kong
chenma@cityu.edu.hk

Abstract

Comprehensively understanding and accurately predicting the performance of large language models across diverse downstream tasks has emerged as a pivotal challenge in NLP research. The pioneering scaling law on downstream works (Hu et al., 2024; Isik et al., 2024) demonstrated intrinsic similarities within model families and utilized such similarities for performance prediction. However, they tend to overlook the similarities between model families and only consider design factors listed in the original scaling law. To overcome these limitations, we introduce a novel framework, Collaborative Performance Prediction (CPP), which significantly enhances prediction accuracy by leveraging the historical performance of various models on downstream tasks and other design factors for both model and task. We also collect a collaborative data sourced from online platforms containing both historical performance and additional design factors. With the support of the collaborative data, CPP not only surpasses traditional scaling laws in predicting the performance of scaled LLMs but also facilitates a detailed analysis of factor importance, an area previously overlooked. Our code is available here[1].

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022) have emerged as one of the most important AI research powered by large-scale parameters, high computational resources, and massive training data. With the substantial increase in model sizes, the evaluation cost of LLMs' performance becomes even more significant. For example, testing a single LLM on certain benchmarks often requires \$10K+ and 4K+ GPU hours (Liang et al., 2023). Therefore, understanding the behaviors and predicting the capabilities of LLMs across scales under various tasks becomes a vital question (Ganguli et al., 2022a; Owen, 2024; Finnveden, 2020; Hu et al., 2024) for both researchers and engineers.

Scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Hernandez et al., 2022; Gordon et al., 2021; Bahri et al., 2024; Muennighoff et al., 2023) have been powerful tools for predicting the capabilities of LLMs. It indicates a power-law correlation between the model performance and design factors such as computational measure (*FLOPs*) utilized during training. Although the scaling law was originally proposed as a strong intuitive guide for designing LLM, researchers (Hu et al., 2024; Ruan et al., 2024; Isik et al., 2024) have extended its utility into predicting model performances on various metrics, such as BLEU in Machine Translation, and different tasks. These works can accurately predict model performances by utilizing the similarity within each model family, *e.g.*, models within each family are usually trained on the same dataset. However, there are several issues rooted in their methods: the performance prediction 1) requires transparent design factors that consume substantial training resources to fit the curve, 2) is only tailored to a certain model family and a specific task metric, and 3) neglects the connections among different models and tasks.

The aforementioned limitations motivate us to design more effective methods for predicting the performance of LLMs on downstream tasks. Two observations sparked our attention. Firstly, A strong similarity exists between model families, *e.g.*LLama-family and GPT family. Models from different families behave similarly in prediction distribution (Shrivastava et al., 2023) and emergent phenomenon (Wei et al., 2022). Secondly, with the emerging LLM models and the increasingly diverse tasks, the cost of enumerating and benchmarking models with tasks increases exponentially. Therefore, we aim to utilize the similarities across model families in order to collaboratively predict

[*] Co-corresponding Authors
[1] https://github.com/Don-Joey/CPP_LLM

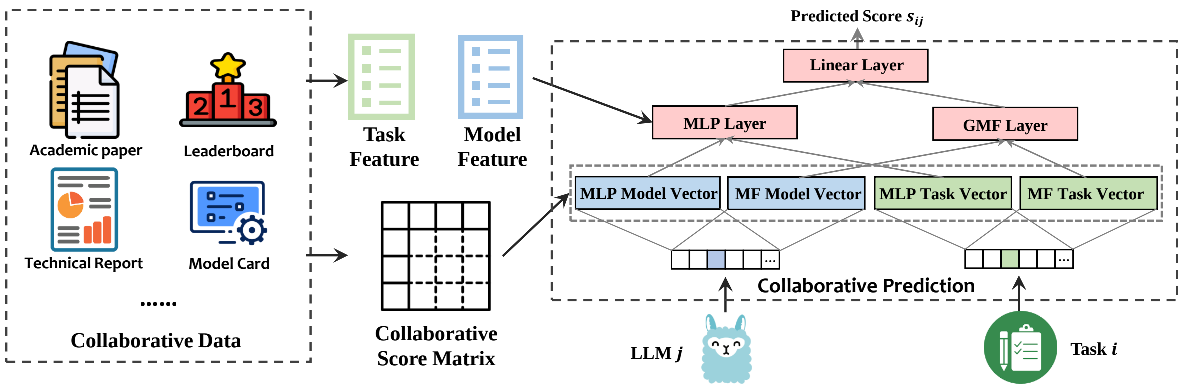

Figure 1: Framework for Collaborative Performance Prediction of Large Language Models. This schematic delineates two principal components: (1) Collaborative Data, which encompasses a score matrix illustrating the performance of various LLMs across downstream tasks, along with external descriptive factors of both models and tasks; (2) Collaborative Prediction Method, given the model and task IDs to leverage this collaborative data, enabling accurate score prediction.

the model performance over diverse tasks in an accurate yet efficient way.

To incorporate the aforementioned intuitions, we propose a new scheme, Collaborative Performance Prediction (CPP), to efficiently predict the performance of LLMs on evaluation tasks. This scheme learns the latent representations of LLMs and tasks, which captures the intrinsic similarity among different models and tasks. The interaction (*e.g.*, inner product) between the latent representations of LLMs and tasks can be utilized to predict the performance of LLMs on certain tasks. To fulfil the proposed scheme, we collect the LLM performance data from academic papers, technical reports, and open leaderboards covering 72 models and 29 tasks. To summarize, our scheme has several advantages:

- **Low Training Cost**: Compared with methods (Hu et al., 2024) that extend scaling law to various downstream tasks, no pre-training or fine-tuning of LLM is required in our scheme.
- **Prediction over proprietary model**: Unlike previous methods (Ruan et al., 2024), our scheme supports prediction over proprietary models without knowing key design factors, such as computational measures.
- **Prediction from small to large:** By utilizing cross-family information, our scheme can accurately estimate model performance, *e.g.*, emergent ability, of large models on downstream tasks given the information from small models.
- **Beyond Scaling Laws**: Our scheme is more general and can incorporate diverse factors, such as task description factors.
- **Factor-level Interpretability**: Our scheme can provide interpretability by analyzing the factors importance of LLMs.

Under our scheme, multiple customized prediction methods (*e.g.*, COLLABORATIVE FITERING (Koren et al., 2022)) can be incorporated to predict the performance of LLMs, further validating the feasibility and generality. Our method enables more diverse factors as input, ranging from traditional LLM design factors to task design factors, *e.g.*, targeted ability and few-shot setting.

Upon extensive experimentation within the open-released core leaderboard of HELM (Liang et al., 2023) and our collected historical matrix, our predictive performance demonstrated exceptionally well. Specifically, even without any input of model factors or task factors: in HELM, we use 50% of the scores to predict the other 50%, the predictive ranking (derived from predicted scores) achieves $Accuracy$ =10%, and $MAE@2$ =39%; in our collected matrix (characterized by a 44% sparsity level) achieves an $Accuracy$ =45%, and the $MAE@2$ =84%. Notably, the accuracy of our prediction from small to large LMs significantly exceeded that predicted by scaling laws. Using an analysis method similar to SHAPLEY-VALUES (Lundberg and Lee, 2017; Shapley, 1952), we elucidate the importance of different factors, which surprisingly does not fully align with scaling law (Kaplan et al., 2020). Therefore, our method is undoubtedly more versatile.

2 Related Work

2.1 Downstream Scaling Law and Performance Predictability of LLM

Scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Hernandez et al., 2022; Bahri et al., 2024; Muennighoff et al., 2023) for LLMs have increasingly become a focal point in understanding and guiding critical design decisions, such as model size and the characteristics and volume of pretraining data. Traditionally, most research in this area has concentrated on how measures like cross-entropy loss or perplexity scale. Subsequent studies have extended these efforts to the scaling behavior on translation (Isik et al., 2024; Ghorbani et al., 2021; Zhuocheng et al., 2023) and other downstream tasks modeling (Caballero et al., 2023; Henighan et al., 2020). The high predictability in LLMs capability has directly spurred extensive research work (see Survey Anwar et al. (2024)) exploring whether LLMs can demonstrate predictability on downstream tasks, which are considered highly unpredictable in traditional ML knowledge (Ganguli et al., 2022a). Particularly, the "emergence" phenomenon (Suzgun et al., 2022; Wei et al., 2022) has challenged predictability, where models suddenly exhibit striking capabilities at specific training reources. Recent studies (Schaeffer et al., 2023) have made remarkable achievements in breaking the discontinuities in performance brought about by emergence, and Ganguli et al. (2022a); Owen (2024); Finnveden (2020) demonstrated the predictability on downstream tasks, for instance, Hu et al. (2024) directly fits a curve of training resources and downstream task performance by repeatedly pretraining a specific model. Furthermore, Arora and Goyal (2023) predicted the performance through decomposing the complex capabilities of LMs to some base skills.

Given that predictability has now been established, we reassess the underlying premises that enable this predictability: the prevailing similarities across multiple models and various downstream tasks (Liu et al., 2023; Perlitz et al., 2024; Polo et al., 2024; Torregrossa et al., 2020; Ilić, 2023). Based on this, we step beyond the limitations defined by scaling laws and propose a new methodology to predict the performance of LLMs on various downstream tasks.

2.2 Collaborative Filtering

Collaborative filtering (CF) (Koren et al., 2022) is a widely used technique in recommendation systems that predicts users' preferences by collecting the historical preferences of many other users. The underlying assumption of CF is that similar users will share similar preferences on similar items. A seminal method in CF is matrix factorization (Koren et al., 2009) (MF). It reduces the dimensionality of the user-item matrix by learning the latent factors associated with users and items, respectively. This approach helps handle sparse data and improves scalability. The factorization of the user-item matrix $\mathbf{R}$ can be represented as

$$\mathbf{R} \approx \mathbf{P}^{\top} \cdot \mathbf{Q}\,, \tag{1}$$

where each column vector in $\mathbf{P}$ and $\mathbf{Q}$ represents a specific user or item, respectively, with hidden dimension d. The latent representations of users and items capture the user preferences and item properties in the latent space, and the inner product $\cdot$ can be utilized to predict the interaction between users and items. To optimize the latent feature vectors, the following loss function is employed:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(u,i)\in\Omega} (r_{ui} - \mathbf{p}_u^{\top} \cdot \mathbf{q}_i)^2\,, \tag{2}$$

which measures the squared differences between the observed ratings r_{ui} and the ratings predicted by the model $\mathbf{p}_u^{\top} \cdot \mathbf{q}_i$ for each user-item pair (u, i) in the set Ω of observed interactions.

Here, Yang et al. (2019) transferred the collaborative filtering for ML model selection by predicting the cross-valided errors, which demonstrates CF's adaptability and efficiency in diverse application areas.

3 Background and Pilot Demonstration

3.1 Scaling Law on Downstream Tasks

For classic scaling laws, researchers propose a hypothesized power-law relationship between a model's computational measures C_m (*e.g.*, training FLOPs) and their performance loss L_m (*e.g.*, perplexity). Specifically, for a model m within a family f (*e.g.*, Llama-2 7B, 13B, and 70B), the relationship is hypothesized as

$$\log(L_m) \approx \omega_f \log(C_m) + b_f\,, \tag{3}$$

where ω_f and b_f are scaling coefficients customized for each model family. Researchers fit

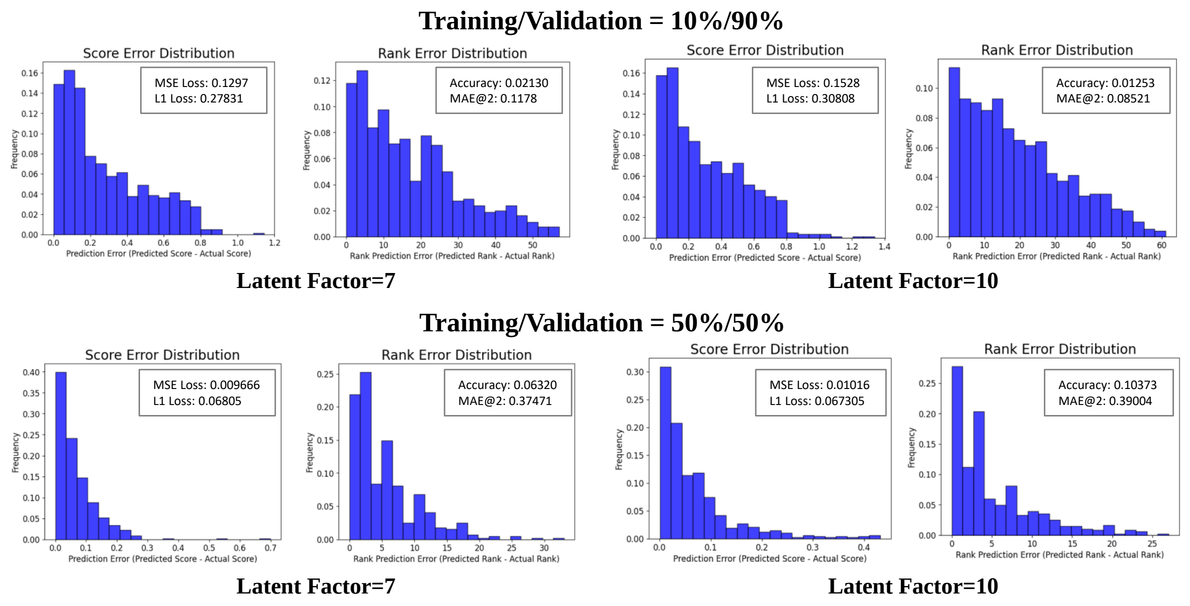

Figure 2: **Error Distribution of Predictions (Normalized Score and Rank Derived by Score) Based on the HELM Lite Leaderboard Using Matrix Factorization:** We evaluate the effectiveness of Matrix Factorization (MF) using two latent factors, 7 and 10, across 2 training/validation split percentages. **Accuracy** is the percentage of instances where the predicted rank equals the actual rank. **MAE@2** is defined as the percentage of instances where the absolute difference between the predicted and actual ranks is 2.

this formula through repeated scaling experiments, then use it to accurately predict performance when larger-scale ($C' > C$). Some studies (Finnveden, 2020; Owen, 2024) have adapted scaling laws to specific downstream task metrics, proposing that sigmoidal functions are more suitable for predictions, as follows:

$$\sigma^{-1}(S_m) \approx \omega_f \log(C_m) + \ b_f, \qquad (4)$$

where S_m refers to the normalized downstream scores of models within the range $[0, 1]$. However, applying scaling laws across different model families on various specific tasks presents a trade-off: fitting unique coefficients for each evaluation scenario (*e.g.*, Llama 2 on MMLU) is a resource-intensive endeavor (Hu et al., 2024); alternatively, estimating these coefficients using a limited number (3-5) of models within the same family may compromise the accuracy of the predictions. Moreover, the recent work (Ruan et al., 2024) extends scaling law by incorporating latent variables to capture the patterns across model families and tasks.

3.2 Pilot Demonstration on HELM

Scaling laws reveal that models from any family exhibit a similar performance trend as computational measures increase. This insight suggests there are commonalities and connections between different models. These motivate us to employ the MF method to explore more similarities beyond computational measures, *e.g.*, the relationship among the different model families and tasks.

We perform the aforementioned MF on the benchmark matrix to observe the error gap between predicted and truth (normalized) scores. Specifically, we select the core leaderboard provided by HELM for our exploratory experiments with only model name, task name, and performance scores. This leaderboard, 68 models and 16 tasks, presented in a score matrix with a density of 82.5%, which includes both open-source and proprietary models, *e.g.*, GPT-4 and Jurassic-2. Our method treats all models and tasks as independent entities without introducing any prior similarity factors. We hope to observe whether MF can predict the remaining scores, giving a small part of the matrix, where we evaluate two training/validation sets split strategies: 10%/90%, 50%/50%. As illustrated in Figure 2, MF can accurately predict most of the missing scores within a low error range, which proves that it can encode the similarity across the model and the task without regression depending on explicit computational measures variable.

4 Collaborative Performance Prediction

4.1 Definition

Motivated by pilot experiments, we introduce the concept of "Collaborative Performance Prediction" (CPP) to facilitate the performance prediction of LLMs.

Definition 1. *Let $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$ be a set of n LLMs, and $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ be a suite of m tasks. Define the Score Matrix* $\mathbf{S}$*, which is an $n \times m$ matrix where each element s_{ij} represents the performance score of model M_i on task T_j. s_{ij} is defined as*

$$s_{ij} = \begin{cases} \textit{score} & \textit{if tested,} \\ \textit{unknown} & \textit{otherwise.} \end{cases}$$

Function: *Employ an prediction method $\mathcal{F}$ to estimate the unknown elements of* $\mathbf{S}$*, denoted by $\widehat{s}_{ij}$, based on the known values.*
Extention: *Accommodate model design factors $\mathcal{V}_m = \{V_m^1, V_m^2, \ldots, V_m^M\}$, such as common computational meatures, and task design factors $\mathcal{V}_t = \{V_t^1, V_t^2, \ldots, V_t^T\}$, such as targeted capabilities and few-shot settings.*

Based on this definition, our framework consists of two components: 1) **collaborative performance data**, 2) **collaborative prediction methods**. We anticipate that an accurate score can be predicted based on the historical performance of various models on downstream tasks and other design factors for both model and task. Moreover, we can incorporate or solely rely on the factors describing the LLM and the associated downstream tasks.

4.2 Collaborative Data

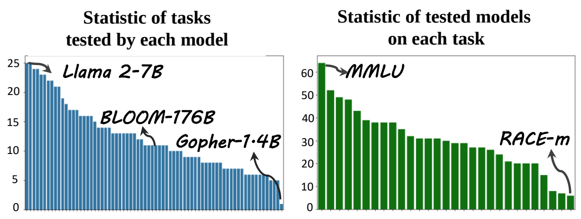

Figure 3: Distribution of Testing Coverage Across Models and Tasks. The left bar shows the number of tasks each model has been tested on; The right bar illustrates the number of models tested in each specific task.

Unlike the scaling law approach, which requires training resource factors to obtain the correlation between metric scores and factors at a high training cost, our proposed method makes use of evaluation results and other design factors reported from existing studies, referred to as *collaborative data*. Open-source leaderboards, such as Open-LLM[2], HELM, and OpenCompass[3], have made tremendous efforts on this issue in fairly evaluating different LLMs. Our efforts extend beyond merely (Ruan et al., 2024) utilizing data from open-source leaderboards with matrix sparsity of 0%. We also extract test results from different models' papers, technical reports, and model cards. Ultimately, we have collected a score matrix of $n = 72$, $m = 29$ with a density of only 56%. Furthermore, we collected 12 and 4 detailed design factors for models and tasks. These details are listed in Appendix B.1. Our data analysis is shown in Figure 3 and Figure 8.

Data Analysis. Based on the *collective data*, we can make the following observations: a) **Uneven distribution of testing resources**. We observe significant variability in the deployment of testing efforts, as shown in Figure 3. For instance, models from the LLAMA series have undergone extensive testing across various tasks, in contrast to models like GOPHER, where testing has largely stagnated. A similar disparity is also evident among tasks, with MMLU and HELLASWAG receiving considerable evaluation, whereas RACE has been relatively underexplored. This trend suggests that as LLMs proliferate and tasks evolve, scores across the matrix will increasingly skew. This leads to a pronounced long-tail effect in testing coverage for many tasks, barring a few that consistently receive comprehensive evaluations. b) **Widespread variations in the scores**. It is noteworthy that identical models yield varying scores on the same tasks across different studies (Shrivastava et al., 2023; AI@Meta, 2024), a variation often attributed to differences in prompt settings, model versions, and the volume of test samples employed. Typically, these score variations range within 0.1, with scores normalized between $[0, 1]$. This phenomenon underscores the importance of public leaderboards and highlights researchers' need to articulate their testing frameworks when performing customized evaluations. When conflicted, we prefer the results from the Open-LLM leaderboard in the collective data. c) **Missing description/model card**.

[2]https://github.com/bentoml/OpenLLM
[3]https://opencompass.org.cn/

We advocate for consistently providing complete model cards for open-source and proprietary models. Such a phenomenon is shown in Figure 8 and, unsurprisingly, a long-tail distribution is witnessed. While it is understandable that proprietary models might withhold specific details about parameters, they can still divulge information about parameter scale and the extent of pre-training. Furthermore, we recommend a more thorough description of testing tasks, including suggested few-shot settings and detailed descriptions of targeted capabilities.

4.3 Prediction Methods

In Section 2.2, classical collaborative filtering methods are inspired to conduct the performance prediction. In principle, most collaborative filtering methods can be applied. Here, in addition to the abovementioned MF, we also leverage neural collaborative filtering (He et al., 2017) (NCF) methods, which uses a multi-layer perceptron to learn the model-task interaction function to predict the score $\widehat{s}_{ij}$ for a model i on a task j, providing a way to learn non-linearities in the data:

$$\begin{aligned}\widehat{s}_{ij} &= f(i, j|\mathcal{M}, \mathcal{T}, [\mathcal{V}_i, \mathcal{V}_j], \theta)\\ &= \text{MLP}(p_i, q_j, [e_{vi}, e_{vj}]),\end{aligned} \tag{5}$$

where $\mathcal{M}$ and $\mathcal{T}$ denote the sets of collaborative models and tasks, and their descriptive factors $\mathcal{V}_i$, $\mathcal{V}_j$ optionally enrich the input data. Here, p_j and q_j are the latent vectors for model i and task j that capture the intrinsic properties of models and tasks, as well as embeddings $[e_{vi}, e_{vj}]$ derived from their descriptive factors, and θ represents the parameters of NCF.

Moreover, we further simplify the model to verify whether it is feasible to predict a score when only inputting the descriptive factors $\mathcal{V}_i$, $\mathcal{V}_j$ into the prediction model:

$$\begin{aligned}\widehat{s}_{ij} &= f(i, j|\mathcal{V}_i, \mathcal{V}_j, \theta)\\ &= \text{MLP}(e_{vi}, e_{vj}),\end{aligned} \tag{6}$$

For both settings, where the goal is to predict the scores accurately, the loss function can be defined as follows:

$$L(\theta) = \frac{1}{N} \sum_{(i,j)\in\mathcal{D}} (\widehat{s}_{ij} - s_{ij})^2, \tag{7}$$

where N is the total number of scores set $\mathcal{D}$ for training, and s_{ij} is the true score for model i and task j.

5 Experiments

In this section, we evaluate the feasibility of CPP from an overall benchmark perspective and a model perspective in Section 5.1 and 5.2, respectively; we then analyze the importance of factors for both models and tasks in Section 5.3. Additionally, a substantial amount of ablation and analysis is placed in the appendix D, such as sparsity, the correlations in tasks and models, and which models and tasks are more critical for prediction.

Experimental Setting. Our validation framework utilizes the aforementioned collaborative dataset as the score matrix $\mathcal{S}$. We partition scores $\{s_{ij}\}$ into train and validation set, detailed in Appendix C.2.

Evaluation Metric. To accurately evaluate CPP, we adopt two types of metrics: 1) SCORE-LOSS metrics including MSE LOSS and L1 LOSS between predicted scores and true scores (normalized) on downstream tasks and 2) RANK-ACCURACY metrics including ACCURACY and MAE@2 between the rank of predicted scores and true scores. We elaborate on these metrics in Appendix C.1.

5.1 Evaluation from Benchmark Perspective

In this study, we select the abovementioned methods, MF and NCF, to verify whether s_{ij} can be accurately predicted based on the input of model i and task j. To examine whether enhancements are helpful, we modify NCF to support the input of design factors, detailed in Appendix C.2. Based on Figure 4 and Table 1, we can make the following observations:

First, all methods accurately predicted model performance, demonstrating that collaborative filtering mechanisms can predict model outcomes based on collaborative data across different models and tasks. This prediction is achieved without explicit scaling factors or fitting a log-power curve. Second, from MF to NCF, the transformation in interaction mechanisms further enhances accuracy, suggesting that model improvements can further augment the efficacy of our methodology. Additionally, we further increased accuracy by incorporating factors, such as model scaling variables and task descriptions, into the NCF framework alongside ID information. This confirms that incorporating explicit factors can enhance model and task similarities. Finally, among all metrics, we particularly noted that the accuracy of the predictive rank-

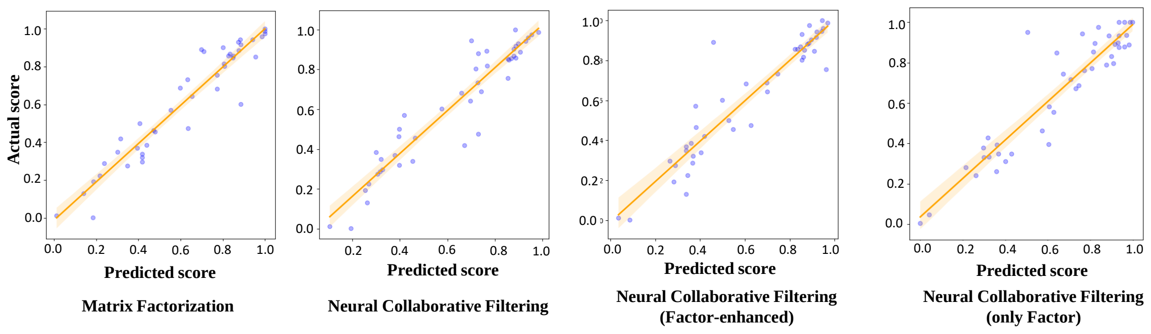

Figure 4: Comparative visualization of predictive accuracy across various scoring methods. From left to right: MF, NCF, NCF with Factor Enhancement, and NCF based solely on Factors. Each plot displays the regression between predicted and actual scores, where the solid line represents the regression fit and the shaded area denotes the confidence interval (CI). A line closer to the diagonal indicates perfect prediction and higher prediction accuracy. These plots demonstrate the enhanced performance in score prediction achieved by integrating factors into the NCF method.

Prediction Method	**Score-Loss**		**Rank-Acc**	
	MSE Loss ↓	Mean L1 Loss ↓	Mean Prec.(%) ↑	MAE@2(%) ↑
Matrix Factorization	$2.16e^{-2}(1.19e^{-4})$	$9.47e^{-2}(2.89e^{-4})$	44.33(0.69)	83.16(0.73)
Neural Collaborative Filtering	$1.58e^{-2}(4.22e^{-5})$	$8.94e^{-2}(3.10e^{-4})$	41.76(1.22)	**84.98(0.42)**
+ Factor Enhanced	$\mathbf{1.25e^{-2}(3.35e^{-6})}$	$\mathbf{7.88e^{-2}(6.31e^{-5})}$	**45.45(0.33)**	84.54(0.27)
Only Factor	$1.75e^{-2}(2.07e^{-5})$	$8.57e^{-2}(1.48e^{-4})$	33.47(0.12)	84.08(0.37)

Table 1: Comparison of prediction methods for LLM performance. **Bold** indicates the best-performed.

ing was acceptable. In other words, researchers can use our method to accurately predict the ranking range of their developed models on test tasks, thereby enhancing model performance on specific tasks.

Predictability with Only Description Factors. We validate whether high predictive accuracy can still be achieved by only inputting the models' and tasks' design factors. As demonstrated in Table 1, the accuracy of predicted rankings (derived from predicted scores) remains high, affirming that our method supports predictions based solely on factors. However, the accuracy is lower than other models, suggesting that finer-grained latent similarities remain encoded as potential factors within the identity information across different models and tasks.

5.2 Evaluation from Model Perspective

To mimic the utilization of CPP in the real world, this section takes a model perspective to investigate the predictive accuracy of CPP upon each model. Specifically, we propose two scenarios: (i) prediction with no prior testing information and (ii) prediction with prior testing information on 2 tasks. These two scenarios correspond to real-world cases when the model has not been developed or is tested

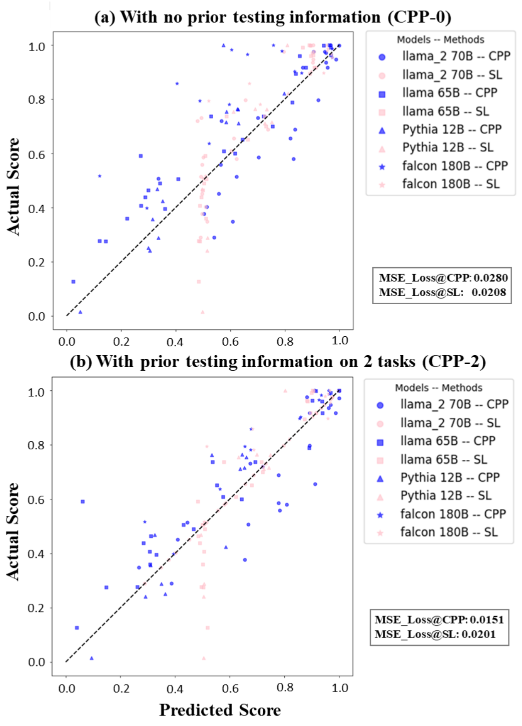

Figure 5: Comparison of the predictive performance of collaborative performance prediction (CPP) versus traditional scaling laws (SL) for LLMs: (a) CPP-0, with no prior testing information, and (b) CPP-2, with prior testing on two tasks.

on a few tasks and expects an accurate prediction of its ability on other tasks. In both scenarios, we focus on larger LLMs, *e.g.*, LLama2-70b, as they are more computationally expensive to develop and test, requiring an accurate LLM prediction.

We report the results of CPP and SL on both scenarios in Figure 5 and can draw the following conclusions. Under the *CPP-0* scenario, CPP demonstrated greater adaptability across different tasks compared to SL, with points closely aligned along the $y = x$ line ("perfect prediction") in Figure 5 (a). This suggests that CPP has effectively captured task-specific characteristics, such as value ranges, whereas SL, despite achieving a lower MSE-LOSS, tends to concentrate its predictions around 0.5. Under the CPP-2 scenario, the distribution of points of CPP is noticeably closer to $y = x$, as shown in Figure 5 (b), and its MSE-LOSS is also lower than that of SL. This indicates that leveraging performance data from other tasks considerably enhances the model's cross-task prediction capabilities, underscoring a degree of consistency across tasks for the same model. This approach demonstrates that predictions for scaling LLMs on downstream tasks can be dynamically improved by evaluating performance on less computationally intensive tasks and using those outcomes to predict scores on subsequent tasks more accurately.

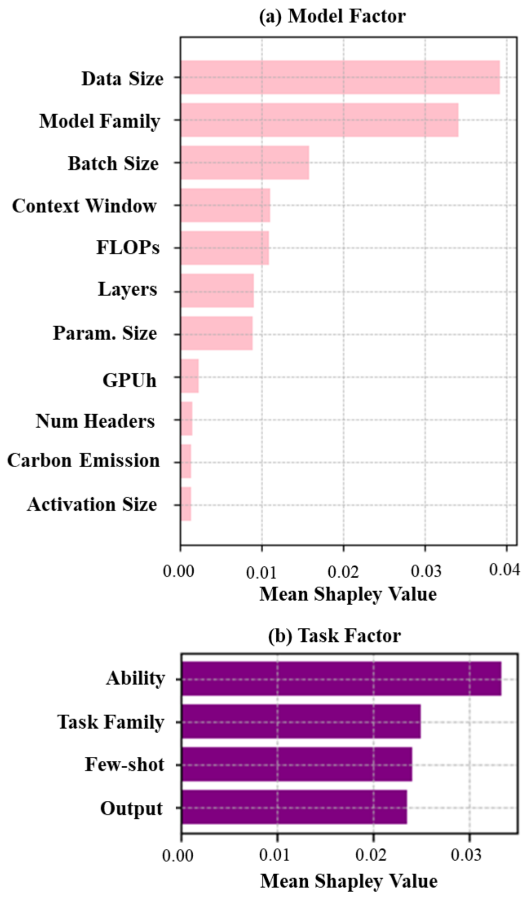

Figure 6: Mean Shapley Value on Each Factor.

5.3 Factor Importance Analysis via SHAPLEY-VALUE

In this section, we aim to analyze each design factor's importance over CPP. The Shapley value, a concept derived from cooperative game theory (Shapley, 1952), offers a systematic approach to measuring individual factors' contribution in predictive models (Lundberg and Lee, 2017; Covert et al., 2021). Appendix C.3 shows a detailed formulation of the Shapley value. Visualization for Shapley values of each design factor is shown in Figure 6.

Based on Figure 6 (a), we can make the following observations regarding model factors. First, we have discovered that in addition to traditionally important factors such as training data size and parameter size mentioned in scaling law (Kaplan et al., 2020), other design factors significantly influence predictive outcomes. These include the model family, context window size, and batch size. Second, the importance of the model family cannot be overlooked, as it may relate to differences in data quality across models, including proprietary data or specific architectural details. For instance, using a particular model family might mean adopting architectures or optimization techniques better suited to specific tasks. Moreover, the size of the context window also significantly affects model performance. A larger context window allows the model to better understand the context in long texts, which is particularly crucial for long-context LLMs (Xiong et al., 2023). Experience (Google, 2024) has shown that such models perform better across various tasks. Batch size, as another crucial factor, affects the stability and speed of model training. An appropriate batch size ensures a balance between the accuracy of gradient estimation and computational efficiency during training.

As for the importance of task factors, results in Figure 6 (b) show that the *target ability* among all factors is more important. This also implies that similarities between the domains of different tasks can help predict outcomes. This conclusion is consistent with previous observations (Ruan et al., 2024; Perlitz et al., 2024; Polo et al., 2024)

In summary, these findings indicate that LLMs performance prediction should not rely solely on traditional design factors limited by scaling law but also on other key factors that might impact overall model performance.

6 Conclusion and Discussion

Advancing beyond traditional scaling laws on downstream tasks, we propose a collaborative performance prediction framework for large language models. It offers significant advantages, including easy deployment, low training costs, and superior predictive accuracy. Uniquely, it enables incorporating additional design factors and supports an in-depth analysis of their impact, including factor importance and correlations in models and tasks. For prediction, we collect collaborative data containing many historical performances and factors.

Our method's predictive accuracy is expected to improve as it benefits from an expanding pool of collaborative data. Moreover, this approach highlights the potential to identify neglected but vital factors beyond traditional scaling laws, such as task design factors, thereby enriching our comprehension of LLM performance predictability on downstream tasks.

Limitations

"Single-source-of-truth". When collecting the *collaborative data*, we hypothesize that each model's performance on each task is identical. However, in the real world, the detailed testing setting, for instance, the testing prompt writing, can influence LLM's performance variance. Although we observed this, we only saved one score from different sources. How to incorporate the setting of testing as an additional dimension remains to be solved in future works.

Susceptibility to data quality. The prediction accuracy of CPP highly depends on the quality of collaborative data. The current version passively collects *collaborative data* from online resources. Should information from either of these data sources be incorrect, the prediction capability of CPP would decrease correspondingly. To overcome such a limitation, jointly considering passive information collected from data sources and active information, such as performances of models tested on some tasks by the user, might be a solution. Utilizing techniques such as efficient benchmarking (Perlitz et al., 2024; Polo et al., 2024) could alleviate the cost of obtaining active information.

Ethics Statement

The data we use are collected from public papers, technical reports, open leaderboards, and model cards on GitHub.

Acknowledgements

This work was supported by the Start-up Grant (No. 9610564), the Donations for Research Projects (No. 9229129) of the City University of Hong Kong, and the Early Career Scheme (No. CityU 21219323) of the University Grants Committee (UGC).

References

AI@Meta. 2024. Llama 3 model card.

Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Zhang, Ruiqi Zhong, Seán Ó hÉigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Yoshua Bengio, Danqi Chen, Samuel Albanie, Tegan Maharaj, Jakob Foerster, Florian Tramer, He He, Atoosa Kasirzadeh, Yejin Choi, and David Krueger. 2024. Foundational challenges in assuring alignment and safety of large language models. In *arXiv*.

Sanjeev Arora and Anirudh Goyal. 2023. A theory for emergence of complex skills in language models. In *arXiv*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. In *arXiv*.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. 2024. Explaining neural scaling laws. In *arXiv*.

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

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. 2023. Broken neural scaling laws. In *International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. In *arXiv*.

François Chollet. 2019. On the measure of intelligence. In *arXiv*.

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. In *arXiv*.

Ian C. Covert, Scott Lundberg, and Su-In Lee. 2021. Explaining by removing: a unified framework for model explanation. *The Journal of Machine Learning Research*.

Lukas Finnveden. 2020. Extrapolating gpt-n performance.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, Sheer El Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Scott Johnston, Andy Jones, Nicholas Joseph, Jackson Kernian, Shauna Kravec, Ben Mann, Neel Nanda, Kamal Ndousse, Catherine Olsson, Daniela Amodei, Tom Brown, Jared Kaplan, Sam McCandlish, Christopher Olah, Dario Amodei, and Jack Clark. 2022a. Predictability and surprise in large generative models. In *Conference on Fairness, Accountability, and Transparency*. ACM.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, Sheer El Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Scott Johnston, Andy Jones, Nicholas Joseph, Jackson Kernian, Shauna Kravec, Ben Mann, Neel Nanda, Kamal Ndousse, Catherine Olsson, Daniela Amodei, Tom Brown, Jared Kaplan, Sam McCandlish, Christopher Olah, Dario Amodei, and Jack Clark. 2022b. Predictability and surprise in large generative models. In *ACM Conference on Fairness, Accountability, and Transparency*, pages 1747—1764.

Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. 2021. Scaling laws for neural machine translation. In *arXiv*.

Google. 2024. Gemini 1.5 blog.

Mitchell A Gordon, Kevin Duh, and Jared Kaplan. 2021. Data and parameter scaling laws for neural machine translation. In *Empirical Methods in Natural Language Processing*, pages 5915–5922.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *International Conference on World Wide Web*, pages 173–182.

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*.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. 2020. Scaling laws for autoregressive generative modeling. In *arXiv*.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, Scott Johnston, Ben Mann, Chris Olah, Catherine Olsson, Dario Amodei, Nicholas Joseph, Jared Kaplan, and Sam McCandlish. 2022. Scaling laws and interpretability of learning from repeated data. In *arXiv*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

Shengding Hu, Xin Liu, Xu Han, Xinrong Zhang, Chaoqun He, Weilin Zhao, Yankai Lin, Ning Ding, Zebin Ou, Guoyang Zeng, Zhiyuan Liu, and Maosong Sun.

2024. Predicting emergent abilities with infinite resolution evaluation. In *International Conference on Learning Representations*.

David Ilić. 2023. Unveiling the general intelligence factor in language models: A psychometric approach.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. 2024. Scaling laws for downstream task performance of large language models. In *arXiv*.

Neil Jethani, Mukund Sudarshan, Ian Covert, Su-In Lee, and Rajesh Ranganath. 2022. Fastshap: Real-time shapley value estimation. In *arXiv*.

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

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Yehuda Koren, Steffen Rendle, and Robert Bell. 2022. *Advances in Collaborative Filtering*, pages 91–142. Springer.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models. In *arXiv*.

Nelson F. Liu, Tony Lee, Robin Jia, and Percy Liang. 2023. Do question answering modeling improvements hold across benchmarks? In *arXiv*.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *International Conference on Neural Information Processing Systems*, pages 4768–4777.

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2023. Scaling data-constrained language models. In *Conference on Neural Information Processing Systems*.

Frank Nielsen. 2016. *Hierarchical Clustering*, pages 195–211. Springer.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744.

David Owen. 2024. How predictable is language model benchmark performance? In *arXiv*.

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 *arXiv*.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinyBenchmarks: evaluating llms with fewer examples. In *arXiv*.

Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Observational scaling laws and the predictability of language model performance. In *arXiv*.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? In *Conference on Neural Information Processing Systems*.

Lloyd S. Shapley. 1952. *A Value for N-Person Games*. RAND Corporation.

Vaishnavi Shrivastava, Percy Liang, and Ananya Kumar. 2023. Llamas know what gpts don't show: Surrogate models for confidence estimation. In *arXiv*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. In *arXiv*.

François Torregrossa, Vincent Claveau, Nihel Kooli, Guillaume Gravier, and Robin Allesiardo. 2020. On the correlation of word embedding evaluation metrics. In *Language Resources and Evaluation Conference*, pages 4789–4797.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. In *arXiv*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. In *arXiv*.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2023. Effective long-context scaling of foundation models. In *arXiv*.

Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine Udell. 2019. Oboe: Collaborative filtering for automl model selection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1173–1183.

Zhang Zhuocheng, Shuhao Gu, Min Zhang, and Yang Feng. 2023. Scaling law for document neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8290–8303.

A Pilot Demonstrations using Neural Collaborative Filtering

In this section, we supplemented the error distribution in Figure 7, which is generated using neural collaborative filtering on the HELM lite leaderboard. Compared to Figure 2, it is evident that neural collaborative filtering consistently outperforms MF across each setting.

B Collaborative Data

B.1 Data Description

List of Models and Tasks. The table 2 contains all the models and tasks we have collected.

Description Factors for Models and Tasks We have collected the characteristics of models and tasks in relevant aspects through model cards, technical reports, and academic papers. We have organized and introduced these characteristics, as well as the corresponding embedding methods, as listed in Table 3.

Note that during data collection, not all factors are available. For these missing factors, such as CO2 and GPU hours, we replace them as zero when entering data.

B.2 Data Analysis

We conducted a statistical analysis of the data we collected, specifically examining the number of models tested for each task, the number of tasks tested for each model, and the number of models described by each factor. Since each task is consistently associated with four factors, we did not create a distribution chart for this aspect.

C Experimental Setup

C.1 Evaluation Metrics

Apart from visualization, we also evaluate the method based on two types of metrics: 1) SCORE-LOSS Metric: we calculate MSE LOSS and L1 LOSS between predicted scores and true scores (normalized) on downstream tasks; 2) RANK-ACCURACY Metric: researchers are sometimes not concerned with detailed scores but rather the rankings the model is in, so we calculate the accuracy of rank derived from the predicted scores, ACCURACY and MAE@2. ACCURACY refers to the percentage of instances where the predicted rank equals the true rank, and MAE@2 refers to the percentage of instances where the absolute difference between the predicted rank and the true rank is in 2, the formulation as below:

$$\text{Accuracy} = \left(\frac{\sum_{i=1}^{N} \mathbf{1}(r_i = \widehat{r}_i)}{N} \right) \times 100\%, \quad (8)$$

$$\text{MAE@2} = \left(\frac{\sum_{i=1}^{N} \mathbf{1}(|r_i - \widehat{r}_i| \leq 2)}{N} \right) \times 100\%, \quad (9)$$

where N is the total number of validation instances, r_i is the true rank, $\widehat{r}_i$ is the predicted rank derived by the predicted score; $\mathbf{1}(\cdot)$ is the indicator function that evaluates to 1 if the argument is true and 0 otherwise; $|\cdot|$ denotes the absolute value.

C.2 Detailed Setting of Validation Prediction Accuracy Experiments

In this section, we detail the setup of each experiment in 5.

Different Prediction Methods. Due to the 44% sparsity of the collected collaboration matrix, we used 5% of the known data as the validation set, with the remaining data serving as the observed training set. We trained each model five times through random splitting, deriving an average performance and variance. We configured our models with latent factors $= 10$, learning rate $= 0.01$, and iteration $= 250,000$. The Figure 4 is the results when random_seed $= 1$.

Predicting from Small to Large LMs. The focus here is on deriving the scaling law applicable to specific task metrics. Undeniably, traditional methods do not provide a directly usable scaling law

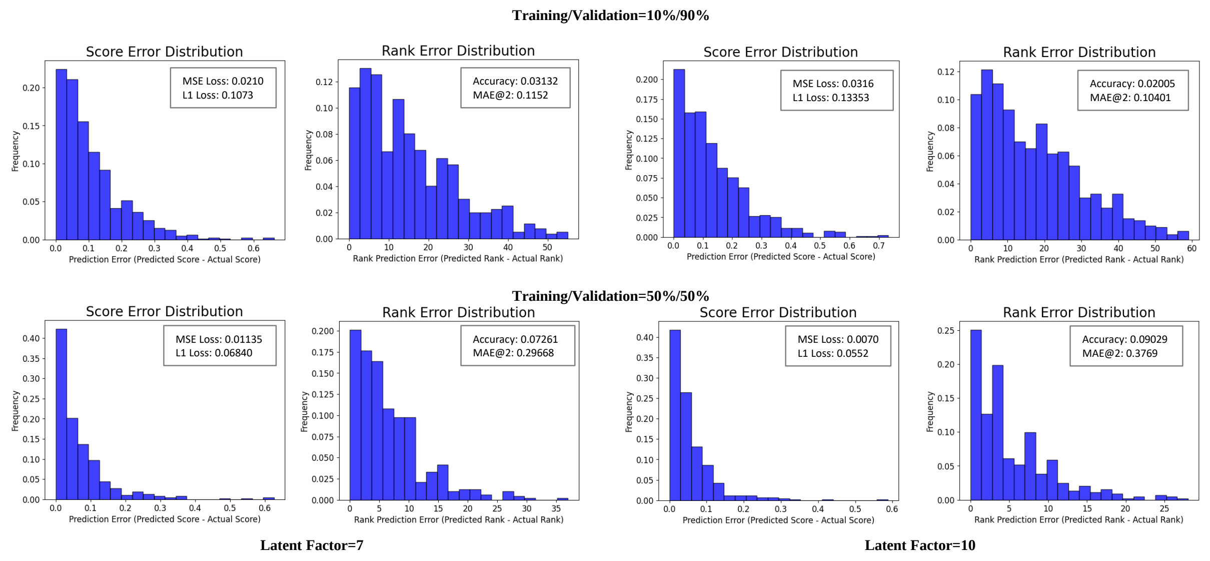

Figure 7: **Error Distribution of Predictions (Normalized Score and Rank Derived by Score) Based on the HELM Lite Leaderboard Using Neural Collaborative Filtering:** We evaluate the effectiveness of Matrix Factorization (MF) using two latent factors, 7 and 10, across 2 training/validation split percentages. **Accuracy** is defined as the percentage of instances where the predicted rank equals the actual rank. **MAE@2** is defined as the percentage of instances where the absolute difference between the predicted rank and the actual rank is 2.

Models	Tasks
'LLama-2-7B', 'LLama-2-13B', 'LLama-2-70B', 'Llama 3 8B', 'Llama 3 70B', 'GLM-130B', 'LLaMA-7B', 'LLaMA-13B', 'LLaMA-33B', 'LLaMA-65B', 'GPT-3-175B', 'PaLM-540B', 'Claude-V3 Haiku', 'Claude-V3 Sonnet', 'Claude-V3 Opus', 'GPT-4', 'gpt-3.5', 'BLOOM-176B', 'Luminous Base-13B', 'Luminous Extended-30B', 'Luminous Supreme-70B', 'OPT-175B', 'GPT-NeoX-20B', 'GPT-J-6B', 'sheared llama-2.7B', 'sheared llama-1.3B', 'INCITE-Base-3B', 'INCITE-Base-7B', 'TinyLlama-1.1B', 'OpenLLaMA-3B-v1', 'OpenLLaMA-3B-v2', 'Pythia-1.4B', 'Pythia-2.8B', 'Falcon-7B', 'Falcon-40B', 'Falcon-180B', 'Mistral 7B', 'MPT-30B', 'MPT-7B', 'chinchilla', 'Pythia-70M', 'Pythia-160M', 'Pythia-410M', 'Pythia-1B', 'Pythia-6.9B', 'Pythia-12B', 'Gopher - 280B', 'Gopher - 44M', 'Gopher - 117M', 'Gopher - 417M', 'Gropher - 1.4B', 'Gopher - 7.1B', 'MT-NLG 530B', 'GLaM', 'Phi-1.5-1.3B', 'Phi-2-2.7B', 'Yi-6b', 'Yi-9b', 'Baichuan 1-7B', 'Baichuan 1-13B-Base', 'Baichuan 2-7B-Base', 'Baichuan 2-13B-Base', 'InternLM2-7B', 'InternLM2-20B', 'Skywork-13B', 'BlueLM-7B', 'Qwen-7B', 'Qwen-14B', 'TigerBot-13b', 'TigerBot-70b', 'Gemma-2b', 'Gemma-7b'	'BoolQ(0-shot)', 'BIG-bench hard(3-shot)','WinoGrande(0-shot)','WinoGrande(1-shot)', 'Winogrande(5-shot)','PIQA(0-shot)','SIQA(0-shot)','HellaSwag(0-shot)','HellaSwag(10-shot)', 'ARC-e','ARC-c(0-shot)','ARC-c(25-shot)','OBQA(zero-shot)','MMLU(5-shot)', 'HumanEval(pass@1)','MBPP(3-shot)','GSM8K(4-shot)','MATH(4-shot)', 'TriviaQA(5-shot)','NaturalQuestions(0-shot)','NaturalQuestions(1-shot)','NaturalQuestions(5-shot)', 'NaturalQuestions(64-shot)','LAMBADA(0-shot)','AGIEval English (3-5 shot)','RACE-m', 'RACE-h','LogiQA','WSC'

Table 2: List of Models and Tasks

across all downstream tasks for comparative analysis. However, we observed in the literature (Ruan et al., 2024) that a sigmoidal curve with a single coefficient and a single bias value represents the scaling law for downstream tasks. Moreover, this curve's coefficients and bias values have a general range across all tasks, $w = [0.5, 2], b = [-10, -3]$. Consequently, we set this range of coefficients and bias for this curve. Then we used the normalized scores of smaller models within the same model family and their corresponding parameter sizes to fit the scaling law curve for each task. This approach generally follows a "pretrain-finetune" methodology. Additionally, CPP-2 refers to randomly selecting two scores from the observed performances of the model to be included in the training data. In this experiment, we use factor-enhanced NCF (setting is same as above).

C.3 Detailed Setting of Analysis Experiments

Shapley-Value for Factor Importance Analysis. Given a predictive model f and a set of factors N, the Shapley value of a factor i is computed as follows:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \cdot [v(S \cup \{i\}) - v(S)], \tag{10}$$

where:

- N is the total set of factors.
- S is a subset of factors excluding factor i.
- $|S|$ is the number of factors in subset S.
- $v(S)$ is the prediction model's output when only the factors in subset S are used.

	Model	
Factors	**Description**	**Embedding**
Model Family	Type of model family, *e.g.*, LLAMA 2, PYTHIA	Categorical Embedding
Pretraining Dataset Size (B)	Data size in millions of tokens	Numerical Embedding
Parameter Size (M)	Number of model parameters in millions	Numerical Embedding
GPUh	GPU hours consumed	Numerical Embedding
FLOPs	Floating-point operations count	Numerical Embedding
Context Window	Max context size in tokens, *e.g.*, 1024, 2048	Categorical Embedding
Batch Size (M)	Size of batches in millions,*e.g.*, 1M, 2M	Categorical Embedding
Layers	Number of layers in the model	Numerical Embedding
Number Heads	Number of attention heads	Numerical Embedding
Key/Value Size	Size of key/value in attention mechanism	Numerical Embedding
Bottleneck Activation Size	Size of activation in bottleneck layers	Numerical Embedding
Carbon Emission (tCO2Eq)	Carbon footprint of training	Numerical Embedding
	Task	
Ability	Type of targeted cognitive ability, *e.g.*, reasoning	Categorical Embedding
TaskFamily	Related task family ,*e.g.*, ARC	Categorical Embedding
Output Format	Format of task output, *e.g.*, binary	Categorical Embedding
Few-Shot Setting	Description of few-shot learning setting,*e.g.*, zero-shot, 32-shot	Categorical Embedding

Table 3: Design Factors of Models and Tasks

Scaled LLMs	**Prior Tasks**	**Score-Loss**		**Rank-Acc**	
		MSE Loss	Mean L1 Loss	Mean Prec.(%)	MAE@2(%)
LLaMA 2-70B	CF-0	$1.34e^{-2}$	$8.83e^{-2}$	16.7	50.0
	CF-2	$1.79e^{-2}(1.3e^{-3})$	$1.79e^{-2}(5.6e^{-4})$	$9.1(7.5e^{-3})$	$54.5(5.7e^{-4})$
LLaMA 3-70B	CF-0	$5.63e^{-2}$	$19.27e^{-2}$	14.3	71.4
	CF-2	$1.7e^{-2}(1.41e^{-4})$	$10.7e^{-2}$ $(1.68e^{-3})$	$20.0(4.0e^{-2})$	$90.0(9.0e^{-2})$
LLaMA-65B	CF-0	$1.73e^{-2}$	$9.78e^{-2}$	24.0	80.0
	CF-2	$1.88e^{-2}(1.42e^{-5})$	$10.02e^{-2}(4.1e^{-4})$	$17.3(1.9e^{-3})$	$71.7(4.7e^{-4})$
Luminous Supreme-70B	CF-0	$6.06e^{-2}$	$20.14e^{-2}$	27.27	63.63
	CF-2	$1.45e^{-2}(1.1e^{-5})$	$10.79e^{-2}(6.4e^{-7})$	$16.7(3.1e^{-3})$	$83.3(3.5e^{-3})$
Pythia-12B	CF-0	$2.19e^{-2}$	$11.2e^{-2}$	21.42	71.42
	CF-2	$1.57e^{-2}(2.1e^{-6})$	$10.88e^{-2}(4.6e^{-8})$	$33.3(2.7e^{-2})$	$66.7(6.9e^{-3})$
Yi-9b	CF-0	$3.20e^{-2}$	$14.66e^{-2}$	44.4	100.0
	CF-2	$0.9e^{-2}(3.1e^{-4})$	$8.1e^{-2}(5.1e^{-6})$	$71.4(9.1e^{-2})$	100(0)
Baichuan 2-13B-Base	CF-0	$2.70e^{-2}$	$12.84e^{-2}$	57.14	100.0
	CF-2	$1.0e^{-2}(4.9e^{-4})$	$7.5e^{-2}(4.7e^{-4})$	$40.0(6.2e^{-4})$	100.0(0)
Qwen-14B	CF-0	$1.05e^{-2}$	$7.96e^{-2}$	33.3	100.0
	CF-2	$3.1e^{-2}(1.8e^{-3})$	$11.1e^{-2}(6.6e^{-3})$	$25.0(7.1e^{-3})$	$91.7(6.9e^{-3})$
TigerBot-70B	CF-0	$8.02e^{-2}$	$19.26e^{-2}$	12.5	75.0
	CF-2	$4.4e^{-2}(2.9e^{-6})$	$15.3e^{-2}(6.6e^{-5})$	$25.0(6.9e^{-3})$	$83.3(6.1e^{-3})$
Gamma-7B	CF-0	$4.94e^{-2}$	$17.62e^{-2}$	15.79	47.36
	CF-2	$10.2e^{-2}(3.2e^{-5})$	$25.9e^{-2}(1.6e^{-4})$	$26.4(8.6e^{-4})$	$58.8(1.4e^{-2})$
Falcon-180B	CF-0	$5.00e^{-2}$	$17.91e^{-2}$	14.58	57.14
	CF-2	$3.2e^{-2}(2.1e^{-5})$	$10.42e^{-2}(7.8e^{-5})$	$23.94(8.5e^{-2})$	$63.6(2.1e^{-5})$
Gopher-280B	CF-0	$14.48e^{-2}$	$30.76e^{-2}$	15.38	61.53
	CF-2	$10.87e^{-2}(3.6e^{-5})$	$23.59(4.2e^{-4})$	$27.33(1.8e^{-3})$	$66.49(6.8e^{-3})$

Table 4: The accuracy of Predicting Scaled Large LMs in CPP-0, CPP-2.

- $v(S \cup \{i\})$ is the model's output when the factors in subset S plus factor i are used.
- The factorial terms $|S|!$ and $(|N|-|S|-1)!$ weigh the contribution of each subset according to the number of factors included or excluded, ensuring a fair allocation across all possible combinations.

The Shapley value, $\phi_i(v)$, quantifies the average marginal contribution of a factor i across all possible combinations of factors. The formula takes every subset S of the total factor set N that does not include i, calculates the difference in the model's prediction output with and without factor i and

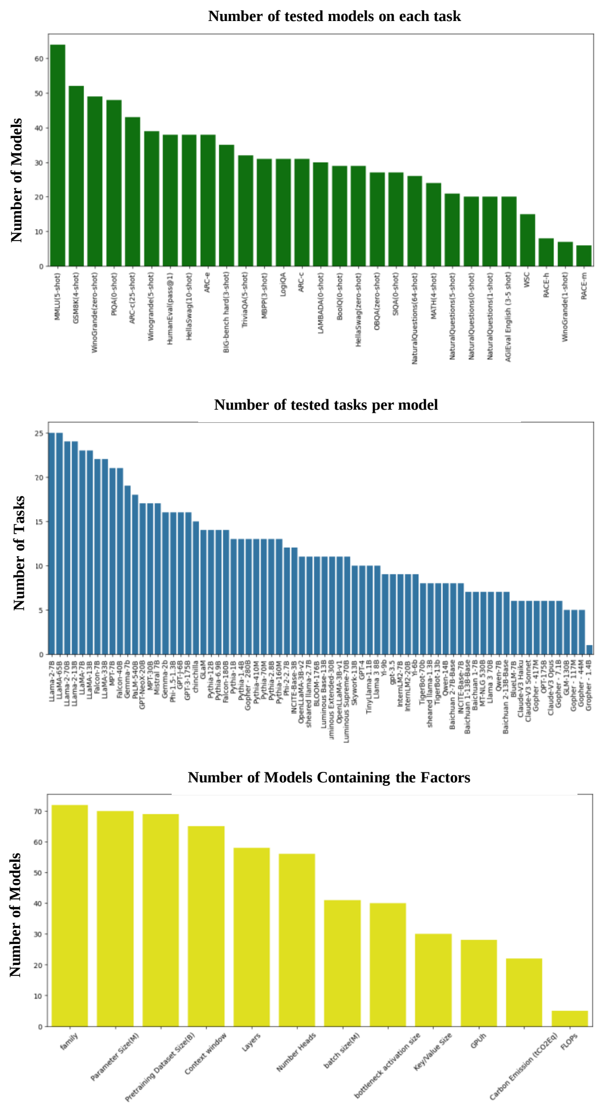

Figure 8: The detailed distribution of collaborative data.

averages this difference over all subsets. The averaging is weighted by the factor $\frac{|S|!(|N|-|S|-1)!}{|N|!}$, which corresponds to the number of permutations in which subset S appears as a prefix or suffix of the total set when factor i is added.

This approach ensures that each factor's contribution is assessed fairly and comprehensively, accounting for interactions with other factors and their unique impact when combined in different ways. Shapley values are particularly useful for factor importance analysis because they provide a solid theoretical foundation and are less biased than simpler importance metrics.

The Shapley value algorithm for analyzing feature (factor) importance is computationally intensive, which has led to the development of various approximation methods (Jethani et al., 2022). Fortunately, our predictive model involves a manageable number of factors, allowing us to use the most accurate direct computation method of Shapley values. Specifically, we apply an enumeration approach to compute Shapley values on a pretrained factor-enhanced neural collaborative filtering model during the inference stage. This involves systematically masking factors to assess their impact.

For the implementation, we mask factors differently based on their data type as outlined in Table 3:

- **numerical factors**: we set the input factor values to zero;
- **categorical factors**: we set the corresponding embedding layer parameters to zero.

We then compute the difference in validation loss with and without each factor present, providing us with each factor's marginal contribution. This detailed approach allows us to quantify precisely how much each factor contributes to the model's predictive performance, providing valuable insights into factor importance and model behavior.

D Ablation Study

D.1 Ablation on Sparsity Threshold

To ascertain whether matrices composed of collaborative performance data can accurately predict the performance of LLMs, it is essential to consider the critical variable: the matrix **sparsity**. We assessed the impact of sparsity on prediction accuracy by manipulating the sparsity of the training matrix via masking. This method allowed us to obtain a reliable measure of average accuracy, as illustrated in Figure. 9. It is noteworthy that our method of controlling sparsity only reduces the number of training samples. We ensured fairness in each comparative experiment by maintaining a consistent validation set throughout. During the experiment, we maintained the same settings for the learning rate and number of iterations as in the main experiment. To ensure the robustness of our experimental results, each reported outcome represents the average score after conducting five random splits.

The data we collected inherently has a sparsity of 44%. Hence, we only have the remaining 46% of collaborative data. As sparsity levels range from 49.60% to 88.80%(masking 10% to 80% of the collaborative data), the graph shows a pronounced increase in L1 Loss and a decrease in Accuracy, indicating deteriorating model performance with

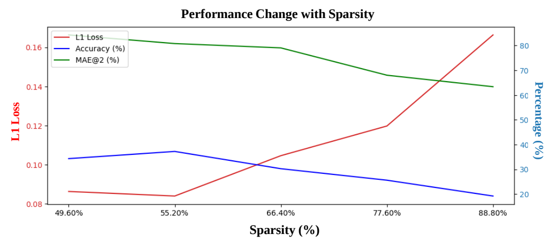

Figure 9: Relationship between matrix sparsity and three key performance metrics: L1 Loss, Accuracy, and MAE@2.

higher sparsity, especially when sparsity exceeds 60%, where there is a significant drop in accuracy. Conversely, MAE@2 remains relatively stable before experiencing fluctuations, suggesting varying impacts on this metric. Interestingly, accuracy even improves when sparsity reaches 50%. We think the possible reason for this might be the presence of an optimal level of information reduction that removes redundant or noisy data without significantly compromising signal integrity. This phenomenon suggests that a moderate level of sparsity could potentially enhance model performance by focusing on more relevant factors.

D.2 Ablation on Predicting Performance on Complex Reasoning and CoT Tasks

From the model perspective, it is crucial for validating the feasibility of predictive methodologies to assess the predictive accuracy on special tasks potentially exhibiting "emergent" phenomena (Suzgun et al., 2022; Wei et al., 2022), including complex reasoning and Chain of Thought (CoT) tasks (Wei et al., 2023). "Emergent' phenomena refers to the challenges associated with predicting performance from smaller models when the scale of a model expands significantly, resulting in discontinuous leaps in model capabilities. The existence of this phenomenon is subject to ongoing debate. Nonetheless, recent efforts (Ganguli et al., 2022b; Hu et al., 2024; Owen, 2024; Ruan et al., 2024; Schaeffer et al., 2023) continue to focus on how scaling laws can be modified to mitigate the "gap" between smaller and larger models. This may involve modifying metrics or incorporating additional data points to linearize the growth curve or alternatively opting for a sigmoidal curve.

Theoretically, these challenges are not too difficult for our prediction method, as the underlying mechanism of "emergent" abilities reflects a type of similarity. This commonality manifests when models exceed a certain scale. By analyzing cross-model similarities—how other larger models demonstrate emergent capabilities compared to their smaller counterparts—we can enhance our predictive accuracy for the current model. Overall, these tasks are pivotal for comprehensive validation processes, *e.g.*, GSM8K (Cobbe et al., 2021), BBH (Suzgun et al., 2022), HUMANEVAL (Chen et al., 2021) and MBPP (Austin et al., 2021).

In detail, if we want to evaluate the performance of predicting a model on these special tasks, the training data is the performance information from other model families, the smaller model of the same family, and the randomly selected two non-special tasks prior to the performance of this model. In our experiment, we tested the 4 models on these tasks, and then we plotted the test results on Figure 10. As illustrated in Figure 10, our predictive scores are more adaptive to each task, where the points are close along the "perfect prediction" line, which means our prediction method captures the similarity in the specific task across models. Our proposed method's MSE Loss is comparable to the scaling law, which shows the feasibility of CPP (in CPP-2).

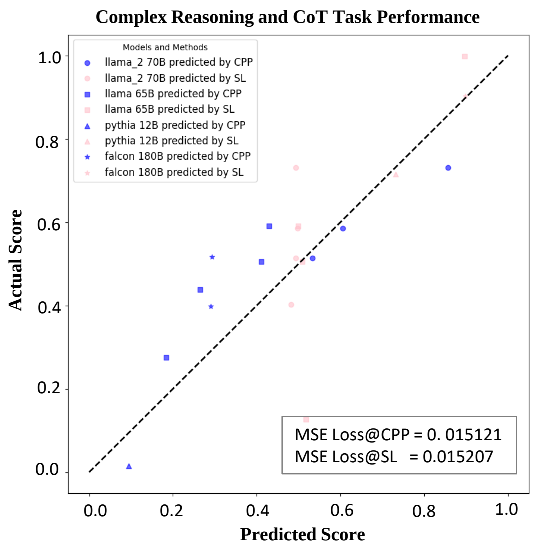

Figure 10: Comparison of the predictive performance of collaborative performance prediction (CPP) versus traditional scaling laws (SL) for Large Language Models (LLMs) in Complex Reasoning and CoT Tasks.

Generalization to Completely New Tasks. As presented in Tab. 5, CPP-T0 and CPP-T2 have a relative small error, demonstrating our method CPP shows reliable generalization. When CPP-T2 has the prior performance of two models in this task, it has a significant drop compared to CPP-T0. These two experimental results inspire us that prediction and evaluation should be interactive, *i.e.*, we should evaluate two small models or tasks to get true but low-cost results, and then the accuracy of prediction can be improved after obtaining the results.

D.3 Correlation between Models

Experiment. We conducted a "leave-one-out" experiment to test the impact of Model A on the predictive performance of Model B. This involved masking Model A and using the performance of other models to train predictive methods, which were then validated on Model B to observe changes in loss. This approach generated a matrix with the masked model names on the X-axis and the validation model names on the Y-axis, with the values representing the change in loss.

The "Leave-one-out" experiment is a robust method commonly used in statistical analysis. To assess the impact of different models on the predictive performance of a specific model, we implemented a strategy where we systematically masked each selected model in the training set. The procedure involved masking each model individually and then training and testing the loss on a validation model. This process was repeated across all models, culminating in creating a matrix where axis=0 represents the masked model ID, and axis=1 represents the validation model ID. The values in the matrix correspond to the loss observed. This experiment was conducted under three different random seeds to ensure the stability and reliability of the results.

Subsequently, each model was used as a validation set, with the remaining data serving as the training set to calculate the loss for each model. This also resulted in a matrix where axis=1 indicates the validation model ID, and the columns[:, valid model id] represent the corresponding loss for that validation model. We derived a delta loss matrix by calculating the difference between these two matrices.

Given that each validation model has its own range of loss variations, we normalized the delta loss matrix. We then performed a row-based correlation analysis on this normalized matrix to assess each model's impact on predictive performance. The higher the correlation value between the two models, their effects on predictions are more similar.

Analysis. Based on this correlation matrix, we further conducted a hierarchical clustering analysis (Nielsen, 2016). The results indicate that a set of models exists that are similar in their impact on the predictive performance of the model. Other models are far away from them. (Details in Table 6)

This analysis not only helps us understand each model's specific contributions to predictive performance but also reveals the similarities and differences in functionality among the models, providing a crucial basis for model optimization and selection.

We performed a row-wise correlation analysis 13 on this matrix and discovered that models from the same family tend to have similar impacts on predictions, as do models of the same size. After conducting a hierarchical distance analysis, we concluded that a group of models exists that, when more performance data is available, can significantly enhance the accuracy of the predictive models. There are also what might be termed "noise model performances" in our analysis D.3.

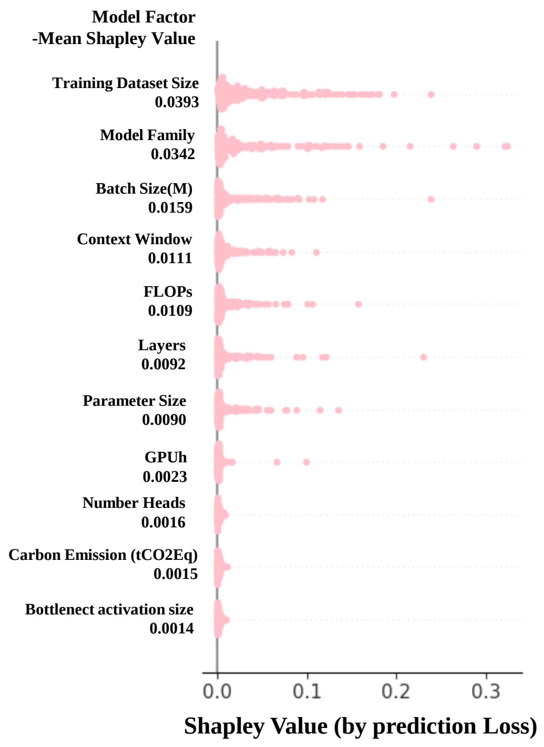

Figure 11: Instance Distribution of the model factor Shapley value. X-axis represents the Shapley value, which indicates the degree of prediction loss change; Y-axis indicates the factor names in order of importance from top to bottom. Each point represents an instance.

D.4 Correlation between Tasks

We also conducted "leave-one-out" experiments on these tasks and created a heatmap figure. 14 of

Table 5: The predictive performance (MSE) of CPP in the predictions of the completely new task. Here, CPP-T0 refers to the predictive performance of CPP in the predictions of the completely new task, and CPP-T2 refers to the predictive performance of CPP in the predictions of the task when we only know two models' performance on this task, indicating CPP has no prior knowledge and few cases.

Models	BoolQ(0-shot)	BIG-bench hard(3-shot)	HellaSwag(10-shot)	HumanEval(pass@1)
CPP-T0	0.02201	0.07103	0.03414	0.1244
CPP-T2	0.0182	0.00725	0.02506	0.0763

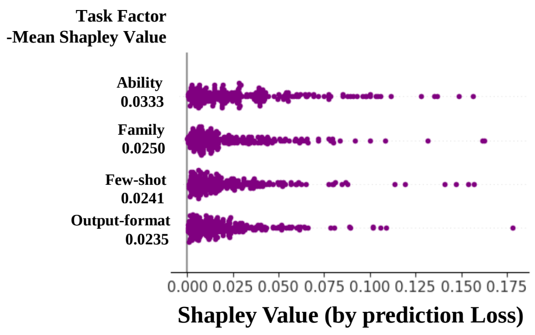

Figure 12: Instance Distribution of the task factor Shapley value. X-axis represents the Shapley value, which indicates the degree of prediction loss change; Y-axis indicates the factor names in order of importance from top to bottom. Each point represents an instance.

the correlations. Tasks with similar targeted ability testing capabilities demonstrated similar influences, such as GSM8K, MATH (Hendrycks et al., 2021), ARC (Chollet, 2019), and HumanEval, which all require complex reasoning abilities.

E Others

E.1 Visualization

The figure 15 is the visualization for the prediction performance of scaled language models on downstream tasks.

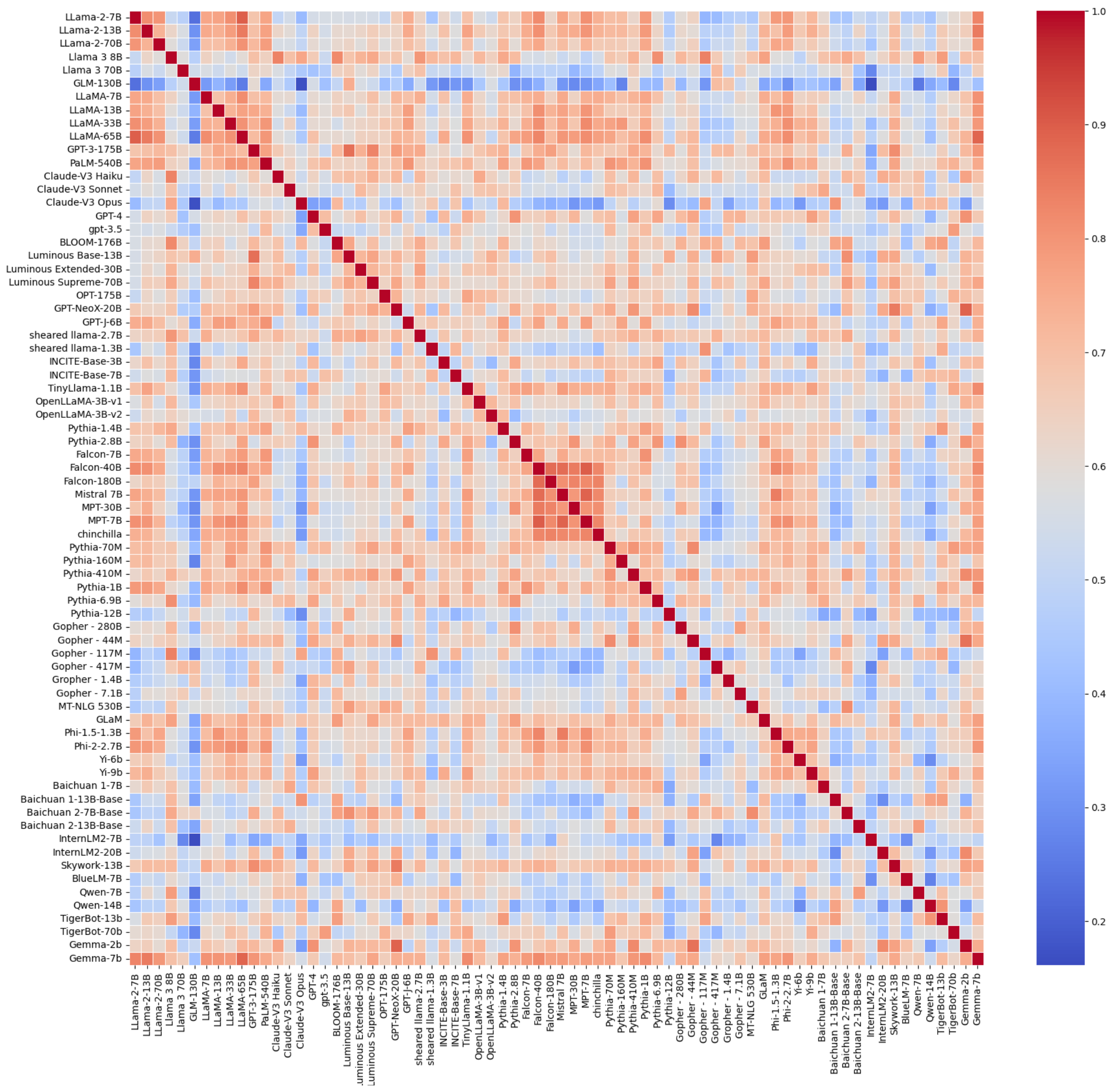

Figure 13: The correlation heatmap of impacts of different models on prediction performance.

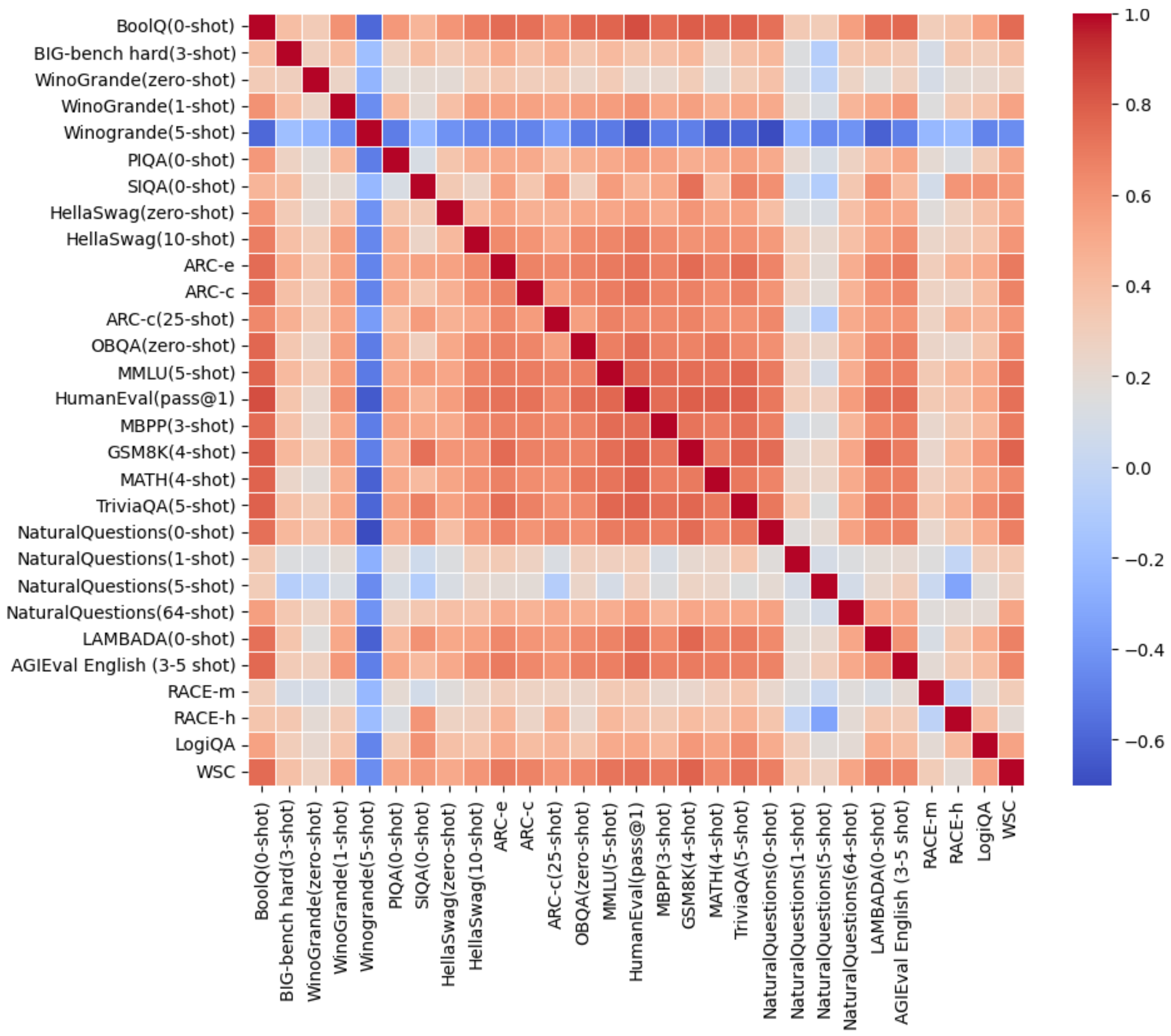

Figure 14: The correlation heatmap of impacts of different tasks on prediction performance.

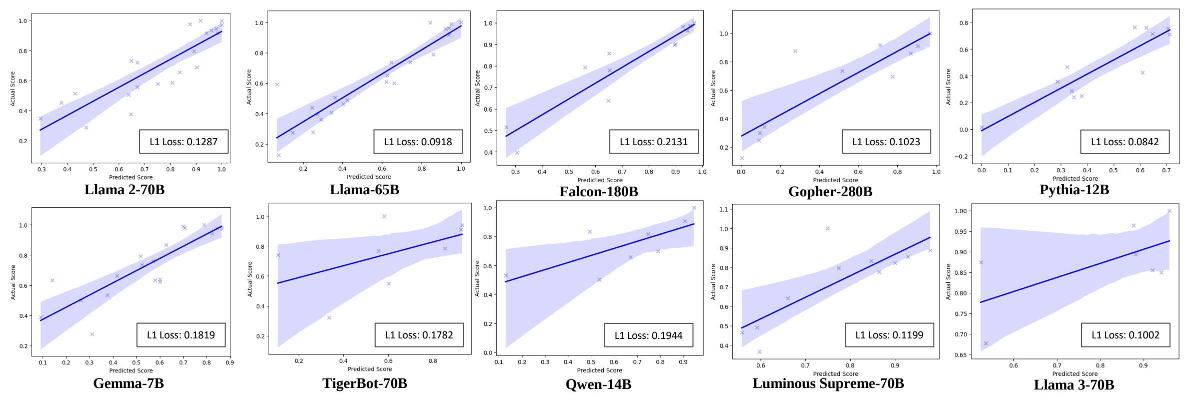

Figure 15: Prediction performance of various scaled Language Models on downstream tasks. This figure illustrates regression plots comparing the predicted versus actual performance normalized scores for a series of large language models, including Llama-2-70B, Llama-65B, Falcon-180B, Gopher-280B, Pythia-12B, Gemma-7B, TigerBot-70B, Qwen-14B, Luminous Supreme-70B, and Llama-3-70B. Each subplot displays a regression line with a shaded 95% confidence interval and includes the L1 loss for each model's predictions, highlighting the accuracy and variability of predictive capabilities across different models.

Distance Cluster	Models
1	LLama-2-7B, LLama-2-13B, LLama-2-70B, Llama 3 8B, LLaMA-7B, LLaMA-65B, Claude-V3 Haiku, Claude-V3 Sonnet, Claude-V3 Opus, GPT-4, BLOOM-176B, Luminous Extended-30B, Luminous Supreme-70B, OPT-175B, GPT-NeoX-20B, sheared llama-2.7B, sheared llama-1.3B, INCITE-Base-3B, INCITE-Base-7B, OpenLLaMA-3B-v1, Pythia-1.4B, Pythia-2.8B, Pythia-70M, Pythia-410M, Pythia-6.9B, Gopher - 280B, Gopher - 44M, Gopher - 117M, MT-NLG 530B, GLaM, Baichuan 1-7B, Baichuan 1-13B-Base, Baichuan 2-7B-Base, Baichuan 2-13B-Base, Skywork-13B, Qwen-7B, Qwen-14B, TigerBot-13b, Gemma-2b, Gemma-7b
2	gpt-3.5, Falcon-7B, Pythia-1B, Gropher - 1.4B, Yi-9b, TigerBot-70b
3	LLaMA-33B
4	Yi-6b
5	BlueLM-7B
6	Falcon-40B
7	MPT-7B
8	Falcon-180B
9	PaLM-540B
10	Pythia-160M
11	GPT-J-6B
12	GPT-3-175B, Luminous Base-13B
13	Gopher - 417M
14	Llama 3 70B
15	LLaMA-13B
16	TinyLlama-1.1B
17	Phi-1.5-1.3B
18	Gopher - 7.1B
19	InternLM2-20B
20	GLM-130B
21	MPT-30B
22	chinchilla
23	Mistral 7B
24	InternLM2-7B
25	OpenLLaMA-3B-v2
26	Phi-2-2.7B
27	Pythia-12B

Table 6: Distance Cluster of Models