# Third-Party Language Model Performance Prediction from Instruction

**Anonymous ACL submission**

## Abstract

Language model-based instruction-following systems have lately shown increasing performance on many benchmark tasks, demonstrating the capability of adapting to a broad variety of instructions. However, such systems are often not designed to be transparent about their limitations; a user may easily prompt a model with an instruction without any idea of whether the responses should be expected to be accurate, or if the system is even capable of performing the task. We propose a third party performance prediction framework, where a separate model is trained to predict the metric resulting from evaluating an instruction-following system on a task while assuming access only to its inputs and outputs at inference time. We perform this analysis with a variety of both open and closed instruction-following models as well as multiple performance predictors, and examine the effect of various factors such as model size, number of training tasks, and prompt format. Our findings indicate that third-party performance prediction is very challenging, and much work remains in developing predictors that can automatically reveal the limitations of modern instruction-following natural language processing systems.

## 1 Introduction

Despite much-discussed advances in the capabilities of language model-based systems that follow instructions (Mishra et al., 2022; Sanh et al., 2022; Wei et al., 2022; Ouyang et al., 2022; OpenAI, 2022, 2023), the research community lacks an understanding of the limits of these capabilities. Ideally, purveyors of a technological product would clearly explain to users the limitations of what the system can be used for.[1] At present, the best a user can do is explore: try out a prompt and see whether the language model can correctly complete the task. We find this state of affairs concerning, because the cost of such tests will fall on the users. Without coordination and information-sharing, different users will make the same explorations and incur unnecessary costs while simultaneously running the risk of relying on systems for tasks which they are incapable of performing adequately.[2]

In this work, we take a step toward empowering users of language-model based systems by proposing a **third party** approach to predicting model performance at the task level. Consider a user with a particular task in mind. Our proposed task performance predictor takes as input the same prompt the user intends for the language model, and *without querying the language model itself*, offers an estimate of the model's performance on the task. We instantiate such models by regressing quantitative model performance metrics on natural language task instructions. If successful, such predictors could help users decide among commercial systems, or even opt out of delegating a task to a language model at all.

Our experiments examine how well existing instruction-tuned LMs' performance can be predicted as a function of model size, choice of evaluation metric, amount of training data, and other factors. We find overall that the task is challenging, with the various factors we explore providing little improvement to predictability. Our results underscore how much progress still needs to be made in designing instruction-following natural language systems whose performance can be accurately pre-

[1] While documentation such as that of OpenAI (2023) breaks down performance by factors such as task categories, languages, or benchmarks (each benchmark being a collection of many tasks), there is little transparency at any finer granularity (e.g., instruction-level) and no publicly-available tool or mechanism to give users a sense of performance for tasks that do not fit neatly into the documented categories.

[2] A second issue, not addressed here, is that users may not realize that they need to check system output for correctness, and may simply assume that any confident answer from a model can be trusted. We suspect that this problem will worsen as users explore more and more use cases not anticipated by the builders of the systems and therefore unaddressed by so-called refusal training.

dicted and made known for the sake of transparency and user safety.

## 2 Related Work

### 2.1 Instruction Tuning

Our work focuses on analyzing the behavior of models that have been trained to follow task instructions. This includes models trained on human-generated instructions and instances (Mishra et al., 2022; Sanh et al., 2022; Wei et al., 2022; Ouyang et al., 2022; Wang et al., 2022) as well as model-generated data (Wang et al., 2023b; Honovich et al., 2023; Taori et al., 2023; Chiang et al., 2023). We primarily use models trained by Wang et al. (2023a) on a variety of instruction-following datasets and initialized with the publicly available LLaMA family of language models (Touvron et al., 2023). We also explore using the closed models GPT-3.5 (OpenAI, 2022) and GPT-4 (OpenAI, 2023). For more details on our choice of instruction-tuned models, please refer to §3.1.

### 2.2 Predicting Model Behavior

Initial work on performance prediction involved training simpler models to predict the performance of larger models as a function of various features, such as model family, model size, task, language, and training procedure (Xia et al., 2020; Ye et al., 2021, 2023). The primary motivation of that work was to address computational and data constraints – training separate performance prediction models could alleviate issues where the computational cost of finetuning on all datasets was prohibitive or where scarce data in a particular language or domain prevented finetuning altogether. While much of that work predicted performance at the task level as in our study, its motivations were different and its methods were often implemented by predicting from hand-crafted features related to properties of each dataset (e.g., model parameter count or language features) rather than the text prompt input to the model itself.

More recent efforts to better understand model behavior have analyzed the ability to predict whether or not a model will perform well on a given input, either as determined by the language model itself or a separately-trained model. Kadavath et al. (2022) analyze models to determine whether they can identify examples for which they can generate the correct response, either by prompting for a "True/False" label with an instance and one or more generations from the model itself or by finetuning the model as a binary classifier using a dataset of inputs and correctness labels from previous model outputs. Yao and Koller (2023) similarly finetune models as binary classifiers to predict whether another model's generated response is correct for tagging, parsing, and semantic parsing tasks. Other work explores models' ability to generate calibrated uncertainty about their responses either through logits, multiple generations, or verbalized expressions of uncertainty, applied to domains such as solving arithmetic problems (Lin et al., 2022) or question answering (QA; Si et al., 2023; Zhou et al., 2023; Cheng et al., 2023). Models' ability to verbalize their uncertainty has also been explored for models trained with reinforcement learning with human feedback (RLHF), again using QA datasets (Tian et al., 2023). Notably, all of these efforts examine being able to predict model performance at the *instance level*, whereas we are primarily interested in predicting performance at the *task level* given a task instruction.

The most similar work to ours in spirit is that of Fu et al. (2023), which also attempts to predict how well a model will perform on a given dataset by training a separate predictor. There are a number of key differences from our efforts: their work focuses on the in-context learning setting using purely pretrained models rather than models specifically trained to follow instructions; they assume access to a set of unlabeled examples for each dataset, rather than just a task instruction; their analysis requires access to model internals by using model output logits to construct a "confidence profile" feature vector for each dataset; and they restrict their analysis to a small set of question answering datasets. A modern trend has seen public releases of models whose internal states are inaccessible, but that can be prompted with a wide variety of user-defined instructions. This situation challenges many of the assumptions made by Fu et al. (2023) and motivates our own work. However, their explorations inspire our efforts, and we attempt to extend their analysis while focusing on task instructions and models trained to follow them.

## 3 Methods

We begin by describing our complete analysis pipeline, also illustrated as a diagram in Figure 1. Each of our experiments involves two finetuned language models: one that is trained to follow

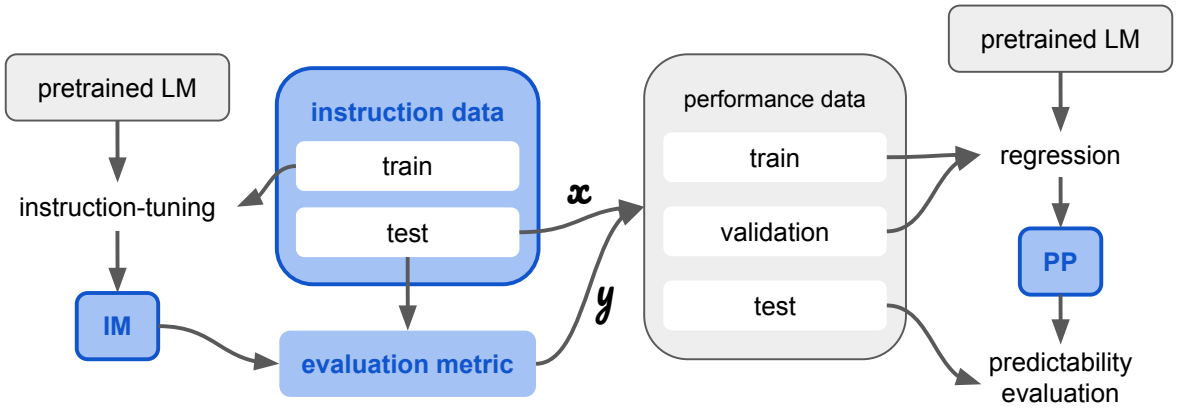


Figure 1: A diagram illustrating our complete analysis pipeline. We begin with a pretrained LM that is instruction-tuned using training tasks from chosen **instruction data**, resulting in an **instruction-tuned model (IM)**. The **IM** is evaluated using the test tasks of the **instruction data** (not necessarily from the same dataset as the training tasks) and a choice of **evaluation metric**. Each pair of test task instruction ($\boldsymbol{x}$) and evaluation performance metric value ($y$) is used to construct the performance data, which itself is split into train, validation, and test sets. The train and validation sets are used to train *another* ("third party") pretrained LM to predict the performance of the **IM** as a regression model, resulting in the **performance predictor (PP)**. Finally, the **PP** is evaluated on the test set of the performance data to determine how well it can predict the performance of the **IM** on unseen tasks. The sections of the diagram highlighted in **blue** indicate the components of the pipeline that we vary to determine their effect on performance prediction: the size of the **IM**, the choice of **instruction data**, the choice of **evaluation metric**, and the size and type of **PP** model.

instructions to perform tasks, which we term the *instruction-tuned model* (IM), and another that is trained to map from an instruction to some measure of the IM's performance on that task, which we call the *performance predictor* (PP). The IMs considered here are drawn from past work. To tune a PP, we evaluate a single IM on instructions it was not exposed to during any of its training (including instruction-tuning). The resulting pairs – each an instruction $\boldsymbol{x}$ paired with the model's performance score $y$ – are divided into training, validation, and test sets for the PP. We train the PP to predict the IM's performance $y$ on instructions $\boldsymbol{x}$ unseen to the IM, and evaluate those predictions on a different set of instructions unseen to either the IM or the PP. We explore a variety of choices for IMs, PPs, and evaluation metrics, which are detailed below.

### 3.1 Instruction-Tuned Models (IMs)

Our experiments use a range of pretrained LMs that have been finetuned to follow instructions. Part of our goal is to assess the effect of IM size (i.e., parameter count) and choice of instruction-tuning dataset on how well the model's performance can be predicted. To this end, we primarily use LLaMA models (Touvron et al., 2023) of various sizes finetuned on a range of instruction-tuning datasets (Wang et al., 2023a). The LLaMA family of models is among the best performing open models, and they have already been used as the basis for a variety of models trained to follow instructions (Taori et al., 2023; Chiang et al., 2023; Wang et al., 2023a). All models were trained by maximizing the per-token likelihood of the gold output given an instruction and possibly an input, with no additional training procedures such as RLHF (Christiano et al., 2017). While adapting our analysis to models trained using RLHF would be an interesting direction of future work, such models and the data to train them are currently limited relative to models trained with supervised finetuning alone, and our intention is to perform an initial analysis on the least-complicated systems that still demonstrate an ability to follow instructions. For the sake of completeness, we additionally include the closed, API-based models GPT-3.5 (`gpt-3.5-turbo`) and GPT-4, as our goal is to implement a pipeline that is applicable even when access is restricted to model inputs and outputs.

### 3.2 Evaluation Metrics

For each instruction-tuned model, we perform inference on a separate evaluation set of instructions in order to generate a dataset of the model's behav-

ior ($y$) on unseen instructions ($\boldsymbol{x}$) for training performance prediction models. For each instruction and output pair, we calculate a quantitative evaluation metric that compares the model-generated output to the gold output, in most cases averaging this metric across instances for tasks that have multiple instances. We treat this final metric as the instruction-tuned model's performance for that instruction. We explore two quantitative metrics commonly used when evaluating instruction-tuned models on a broad range of tasks: ROUGE-L (Lin, 2004) and Exact Match score. Again, since these automated metrics can be computed by comparing a model-generated output to a gold output, we can apply them to models where we only have access to model generations. Additionally, we briefly compare to using model loss as the metric to predict in §4.3, as an exploration of what we can achieve with performance prediction when additional information (in this case, the LM's output distribution) is available.

### 3.3 Performance Predictors (PP)

Once we've evaluated an IM on unseen instructions, we use the performance data ($\langle \boldsymbol{x}, y \rangle$ pairs) to build models that predict the IM's performance, and evaluate those predictions on the test subset of the IM performance data. We primarily finetune RoBERTa (both base and large sizes; Liu et al., 2019) as the PP, motivated by the goal of having a lightweight separate model that can predict how well an instruction-tuned model will perform without incurring the inference costs of a much larger model. RoBERTa models are trained as regression models by adding a linear layer to the `[CLS]` token at the output layer and training to minimize mean-squared error between the predicted and true evaluation metric for each instruction. In some cases, we also train the base LLaMA model used to build the IM by similarly adding a linear layer to the EOS token at the output layer and adding and updating LoRA adapters (Hu et al., 2022) rather than updating all model weights. This is done in an effort to establish an "upper bound" of performance prediction that can be achieved while incurring the prohibitive computational cost of using a much larger model. We additionally include a simple baseline of predicting the mean metric value across all training instructions, as a "lower bound" to establish whether training a separate predictor model offers any benefit at all.

Unless otherwise specified, PPs are trained and evaluated on the results of evaluating IMs on the Super-NaturalInstructions (SuperNI) test set tasks (Wang et al., 2022). For each experiment, we perform 10 random 80%/10%/10% train/validation/test splits of these tasks and report mean and standard deviation of the performance predictor's root mean squared error (RMSE) in predicting the true evaluation metric from the task instruction. Validation data is used to tune hyperparameters for the PPs, namely batch size and learning rate; for full experiment details, please refer to Appendix A.

## 4 Results

### 4.1 Performance Prediction is Challenging

Table 1 shows our main results of predicting performance on SuperNI test set instructions, with mean and standard deviation (subscript) RMSE values across all train-test splits of the SuperNI test tasks. We explore using both Exact Match and ROUGE-L as the target metrics to be predicted for each task, with base and large RoBERTa models as the PP models as well as LLaMA-13B (upper bound) and the simple mean baseline (lower bound). For the IMs, we use the 13B-parameter versions of the Alpaca, Vicuna[3], and Tülu models, as well as GPT-3.5 (`gpt-3.5-turbo`) and GPT-4.

Results demonstrate that performance prediction from task instruction is incredibly difficult, with RMSE values generally around 20 or higher (for metrics in the 0–100 range) across all experimental conditions. Both RoBERTa-base and RoBERTa-large perform comparably to the simple mean baseline, likely indicating that there is little learnable signal within the set of instruction-metric pairs. The lack of a meaningful difference in performance between RoBERTa-base and RoBERTa-large suggests that size of the PP model also makes little difference in performance prediction at the RoBERTa scale. In general, ROUGE-L is more predictable than Exact Match, with consistently lower RMSE values across choices of PPs and IMs. The values below the name of each model indicating performance in ROUGE-L on the SuperNI test tasks show that models which tend to perform better on SuperNI are *less* predictable, exhibiting the worst RMSE values in general.

[3]We use the 13B-parameter version of LLaMA finetuned on the ShareGPT dataset by Wang et al. (2023a) as an attempt to replicate the training of Vicuna.

| | IM: | **Alpaca-13B** | **Vicuna-13B** | **Tülu-13B** | **GPT-3.5** | **GPT-4** |
|---|---|---|---|---|---|---|
| avg. performance (ROUGE-L): | | *45.9* | *44.6* | *61.7* | *53.6* | *63.5* |
| Exact Match | mean | $25.7_{3.4}$ | $26.4_{3.2}$ | $32.4_{2.3}$ | $30.8_{3.9}$ | $36.4_{3.8}$ |
| | RoBERTa-base | $26.2_{4.3}$ | $26.1_{4.9}$ | $33.6_{3.3}$ | $32.5_{5.4}$ | $38.2_{4.1}$ |
| | RoBERTa-large | $27.4_{5.4}$ | $26.9_{5.4}$ | $34.4_{4.9}$ | $33.0_{5.7}$ | $37.2_{4.5}$ |
| | LLaMA-13B | $19.1_{5.3}$ | $18.9_{5.8}$ | $21.4_{3.8}$ | $21.7_{5.6}$ | $22.4_{8.7}$ |
| ROUGE-L | mean | $21.3_{3.5}$ | $20.4_{3.3}$ | $21.4_{1.8}$ | $22.4_{2.9}$ | $24.2_{4.2}$ |
| | RoBERTa-base | $22.5_{4.1}$ | $21.2_{3.6}$ | $22.1_{3.8}$ | $23.4_{3.4}$ | $25.6_{4.0}$ |
| | RoBERTa-large | $22.6_{4.2}$ | $21.2_{3.6}$ | $22.0_{3.7}$ | $23.5_{3.4}$ | $25.7_{3.9}$ |
| | LLaMA-13B | $22.2_{4.7}$ | $21.2_{3.8}$ | $21.5_{3.4}$ | $21.6_{3.5}$ | $22.1_{5.1}$ |

Table 1: Test set RMSE of mean baseline and various PP models finetuned to predict performance from task instruction for various IMs (columns). Subscript shows standard deviation across 10 splits of the SuperNI test tasks.

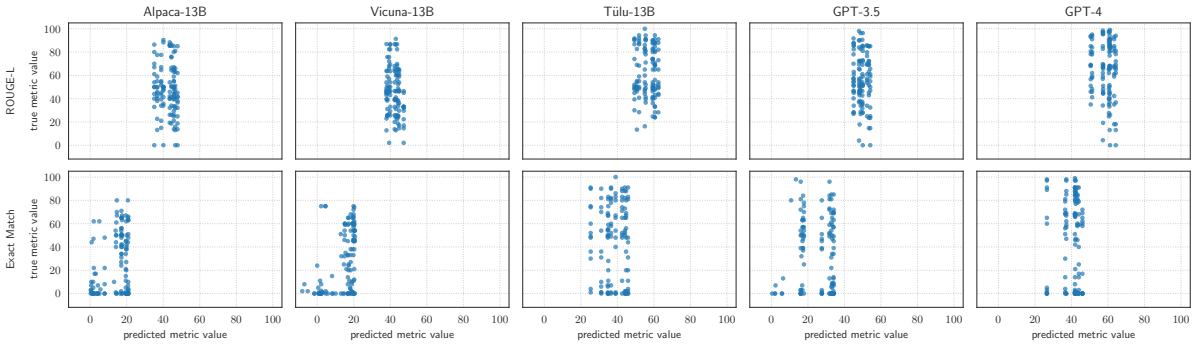


Figure 2: Predicted vs. true metric value when using RoBERTa-large to map from task instruction to performance – either ROUGE-L (top row) or Exact Match (bottom row) – for various instruction-following models (columns).

The results for LLaMA-13B as PP are more promising, with RMSE values that outperform the RoBERTa models as well as the mean baseline when predicting the Exact Match score across all IMs. However, results are still poor, with RMSE values near 20 for both evaluation metrics and all IMs, so even a much larger PP with a more modern base model is unable to predict performance values that are near ground truth. It is additionally worth noting that finetuning LLaMA-13B is a much more prohibitive method of performance prediction, as it involves using a model with around 36× the number of parameters of RoBERTa-large and incurring comparable inference costs to many of the IMs themselves. As such, we treat the LLaMA-13B performance prediction results as demonstration of a hopeful upper bound while simultaneously underscoring the large room for improvement in developing better-performing and lightweight PP models.

Figure 2 shows a more detailed view of a subset of the same results, with scatter plots of the predicted versus true metric value (ROUGE-L or Exact Match) when using RoBERTa-large as the PP for both metrics and all IMs. Results are shown for all train-test splits of the SuperNI test tasks with 12 held-out instructions per random split (10% of the SuperNI test tasks), resulting in a total of 120 predicted and true performance values for each combination of evaluation metric and IM. This qualitative view of the predictions highlights the fact that RoBERTa-large generally learns to predict roughly the same mean performance value across all tasks within each train-test split of the SuperNI test tasks. These results explain the similarity in performance to the simple mean baseline and demonstrate that the PP model does not learn a meaningful association between instruction and performance.

### 4.2 Effect of Various Factors on Predictability

We perform more detailed analyses by altering various factors that may affect the behavior of a PP when used to predict the performance of an IM. These factors include the size of the IM, the number of tasks used to train the PP, and the choice of prompt (instruction-only or instruction + 2 positive

demonstrations).

#### 4.2.1 Size of Instruction-tuned Model

| PP: | mean | RoBERTa-large |
|---|---|---|
| LLaMA-7B (*35.8*) | $21.3_{3.4}$ | $20.7_{4.6}$ |
| LLaMA-13B (*44.6*) | $26.4_{3.2}$ | $27.0_{5.4}$ |
| LLaMA-30B (*44.2*) | $27.5_{2.8}$ | $27.9_{5.3}$ |
| LLaMA-65B (*48.9*) | $30.0_{3.3}$ | $31.7_{5.3}$ |

Table 2: Test set RMSE of predicting Exact Match given task instruction, for various sizes of LLaMA models instruction-tuned on ShareGPT. Model performance in ROUGE-L on the SuperNI test set is given in parentheses next to the name of each model.

We examine the effect of IM size by predicting the performance of various-sized LLaMA models instruction-tuned by Wang et al. (2023a) on the ShareGPT dataset, ranging in scale from 7B to 65B parameters. Results are shown in Table 2, focusing on the mean baseline and RoBERTa-large PP model as well as the Exact Match metric (results for RoBERTa-base and ROUGE-L are similar). The RMSE values indicate that performance prediction worsens for increasing model size, while model performance on the SuperNI test set improves (ROUGE-L values in parentheses next to each model name). However, the mean baseline also exhibits increasing RMSE values with model scale. This likely suggests that larger, better-performing models are less predictable not because they exhibit more dissimilar behavior on tasks with similar instructions, but because their performance metric values cover a broader range. In any case, we can conclude that scale of the IM alone does not improve performance prediction.

#### 4.2.2 Number of Training Tasks

| PP training: | SuperNI | + BIG-bench |
|---|---|---|
| Alpaca-13B | $27.4_{5.4}$ | $25.4_{6.2}$ |
| Vicuna-13B | $26.9_{5.4}$ | $26.1_{5.5}$ |
| Tülu-13B | $34.4_{4.9}$ | $34.6_{3.9}$ |
| GPT-3.5 | $33.0_{5.7}$ | $32.3_{6.7}$ |
| GPT-4 | $37.2_{4.5}$ | $37.7_{4.5}$ |

Table 3: Test set RMSE of using RoBERTa-large to predict Exact Match given task instruction, using either just SuperNI instructions or SuperNI and BIG-bench instructions for training the PP model.

We examine the effect of increasing the number of tasks / instructions used to train the PP model by including tasks from BIG-bench (BIG-bench authors, 2023). Since BIG-bench tasks do not come with instructions, we manually annotate tasks either by converting the task description (if provided) into a SuperNI-style declarative instruction or writing an instruction from scratch when necessary. We additionally filter out a number of tasks, leaving a total of 156 tasks to use as additional training tasks (a full list of included tasks can be found in Appendix B). We perform the same splits of the SuperNI test tasks as before, only adding the BIG-bench tasks to the training split in each case to increase the number of instructions used to train the PP model in an effort to improve its performance.

The results can be found in Table 3. While the inclusion of BIG-bench task instructions represents a nearly 2.5× increase in the number of training instructions for the PP model, the RMSE values demonstrate that the performance difference is negligible. There are a number of plausible explanations for why this is the case – in general, BIG-bench tasks are substantially different from SuperNI tasks, including those which deviate from standard language understanding such as decoding encrypted text, performing arithmetic operations, and generating chess moves. The domain shift between the two datasets may explain why the inclusion of the additional instructions did not provide enough meaningful signal for the PP models.

#### 4.2.3 Prompt Format

| format: | instruction | + 2 demonstrations |
|---|---|---|
| Alpaca-13B | $27.4_{5.4}$ | $27.3_{3.7}$ |
| Vicuna-13B | $26.9_{5.4}$ | $29.0_{2.6}$ |
| Tülu-13B | $34.4_{4.9}$ | $33.6_{2.4}$ |
| GPT-3.5 | $33.0_{5.7}$ | $31.6_{3.3}$ |
| GPT-4 | $37.2_{4.5}$ | $37.8_{3.8}$ |

Table 4: Test set RMSE of using RoBERTa-large to predict Exact Match given task instruction, using SuperNI tasks with an instruction-only prompt or the instruction with 2 positive demonstrations.

We primarily focus on instruction-tuned models evaluated in a zero-shot manner with only an instruction and instance input, as this has become standard practice and reflects how a generic user might interact with an instruction-following system. However, previous work exploring the effect of the

prompt format has shown notable improvements in IM performance when including additional information, such as positive demonstrations of the task (Wang et al., 2022). We explore whether this improvement in performance also leads to an improvement in predictability by additionally evaluating all IMs with a prompt format that uses a task instruction with two positive demonstrations, using the positive examples provided with each SuperNI task. The RMSE values resulting from training RoBERTa-large to predict the Exact Match score in this setting, as well as the original instruction-only RMSE values, can be seen in Table 4. While the inclusion of demonstrations improves the performance of the IMs themselves by roughly 2–6 points in Exact Match score, the evaluation results don't lead to the models being more predictable as there are no meaningful differences in values between prompt formats for any model.

### 4.3 Predicting Loss

| PP: | **mean** | **RoBERTa-large** |
|---|---|---|
| Alpaca-13B | $1.70_{0.27}$ | $1.51_{0.43}$ |
| Vicuna-13B | $1.97_{0.46}$ | $2.10_{0.53}$ |
| Tülu-13B | $1.25_{0.32}$ | $1.10_{0.36}$ |

Table 5: Test set RMSE of predicting cross-entropy loss given task instruction, for various LLaMA-based instruction-following models.

Our core analysis above relied on the use of automated evaluation metrics such as Exact Match and ROUGE-L that can be formulated as functions applied to a pair of generated and gold outputs. This is based on the assumption that we do not have access to model internals or outputs other than generated text, a practical assumption given the increasing use of modern systems that only allow for limited access such as through API requests. However, while such automated metrics can be shown to correlate with other variables such as accuracy on classification tasks (Wang et al., 2022), they come with their own limitations. For instance, for tasks that require more creative or free-form generations from the model, it is possible for the model output and gold label to be semantically equivalent and equally valid responses to the instruction while differing in surface form such that metrics like Exact Match and ROUGE-L are inappropriate (Holtzman et al., 2021). To address this, we perform an additional experiment training PP models to predict the model's *loss* (on the gold output) instead, avoiding token-based comparison between two pieces of text entirely. We do this by evaluating the average per-token cross-entropy loss of the gold label given a prompt (task instruction and input) for each instance in each dataset, averaging across instances to get a single average loss value per task / instruction. We then follow the same regression training procedure to build PP models that predict this value.

Scatter plots similar to Figure 2 can be seen in Figure 3 (where this time the true and predicted values are this average per-task loss), and the quantitative RMSE values corresponding to these results are in Table 5. As this analysis requires access to model output distributions to compute the loss, we are limited to the LLaMA-based open models. The plots show the predictions made by RoBERTa-large finetuned to predict average per-task loss, using the same 10 splits of the SuperNI test tasks as in previous results. Qualitatively, these results appear to show greater correlation between the true and predicted values as compared to the metrics considered earlier, indicating that there may be more of a learnable signal for predicting loss. However, the mean RMSE values show that the finetuned RoBERTa-large model still does not outperform the simple mean baseline on average, indicating that performance prediction remains challenging even with access to a quantitative metric not based on token-level comparison between generated and gold outputs. Taken together, our results motivate further work in identifying quantitative task-level metrics that can be accurately predicted, and perhaps in designing IMs themselves to be more predictable.

## 5 Discussion

Our experiments varied a number of factors that could impact the predictability of an instruction-tuned model's performance on unseen instructions. We broadly summarize our findings below:

**Performance prediction remains incredibly challenging regardless of setup.** RMSE values remain at 20 or higher for metrics in the range of 0–100, indicating that PP models fail to predict values that are even somewhat close to true performance. This remains true across a variety of open and closed instruction-following models, for multiple automated evaluation metrics. There is also

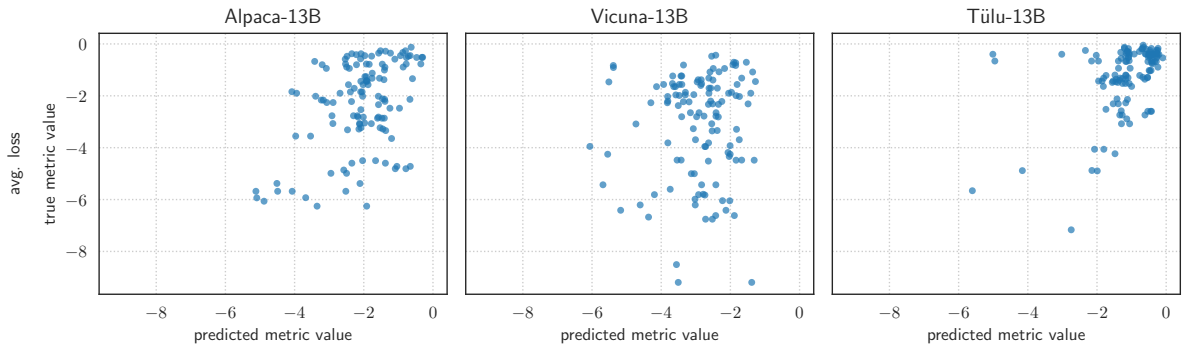


Figure 3: Predicted vs. true loss value when using RoBERTa-large to map from task instruction to loss for various instruction-following models.

little performance difference between RoBERTa PP models of different scales, and none of them outperform a simple mean baseline. One optimistic result occurs when using LLaMA-13B as the PP model, but performance is still relatively poor and this comes at the cost of scaling up the PP model to the size of the IM.

**Increasing instruction-tuned model scale, increasing number of training tasks, and adding demonstrations to prompts all fail to improve performance prediction.** The behavior of larger models does not seem to be any more predictable, nor is the behavior of models with access to additional information in the prompt. The typical strategy of increasing the amount of training data available for the finetuned PP model to better learn a prediction signal is also insufficient. The number of instructions remains small, so future work could ascertain whether the problem setup is still limited by the amount of instructions or if there really is no learnable pattern in IM behavior.

**Predicting cross-entropy loss does not improve performance.** Despite avoiding the issues inherent in using metrics based on token-level comparison between generated and gold outputs, training PP models to predict loss still does not lead to better results than the simple mean baseline.

## 6 Conclusion

Recent NLP systems seem to be able to perform arbitrary tasks given an instruction. Yet we are still not able to understand or explain to users the limitations of these systems such as by reliably predicting their success or failure on new, previously unseen instructions. We take a first step toward this goal by training a separate predictor model to map from a task instruction to the quantified performance of a given instruction-tuned model on that task. Our results show that performance prediction is challenging, with numerous factors like choice of evaluation metric, predictor model size, instruction-following model size, number of training tasks, and prompt format all showing negligible effect on the predictability of instruction-tuned model behavior. Much work remains to be done in designing systems whose limitations can be well-predicted and revealed transparently and freely to their users.

## 7 Limitations & Ethical Considerations

While we explore the third party performance prediction problem across a variety of factors, there are several constraints that limit our analysis and could be explored in future work. Likely the largest limitation is data – few datasets exist in the SuperNI style (with multiple tasks each having a declarative instruction and multiple instances for evaluation) that existing models have not already been trained on. Even with the addition of BIG-bench instructions, the resulting dataset is around 250 training instructions, which is still small by most standards and may not provide enough data in general to learn to predict performance. Building more datasets in this format, perhaps by scaling up dataset generation in an automated fashion (Wang et al., 2023b; Honovich et al., 2023), could expand this analysis to help overcome the data limitation.

Additionally, using a quantitative, automatic evaluation metric may itself not be appropriate when considering arbitrary tasks, including ones that are creative or based on open-ended generation. Reliable quantitative evaluation in the general instruction-tuned setting is an open challenge. While we attempt to address this with our results predicting cross-entropy loss, the choice of evalua-

tion metric for arbitrary instruction-following tasks remains an open question.

Instruction-tuned model performance can also depend on how the instruction is phrased, and previous work has demonstrated model sensitivity to perturbed or paraphrased instructions (Zhao et al., 2021; Webson and Pavlick, 2022). Driven by assumptions that most users will not "engineer" instructions extensively, our experiments only consider a single instruction per task, and redefining performance based on multiple instructions per task (i.e., more "task-specific" rather than "instruction-specific" behavior) may lead to other interesting results.

## References


BIG-bench authors. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Silei Cheng, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023. Prompting GPT-3 To Be Reliable. In *International Conference on Learning Representations*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.

Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Conference on Neural Information Processing Systems*.

Harvey Fu, Qinyuan Ye, Albert Xu, Xiang Ren, and Robin Jia. 2023. Estimating Large Language Model Capabilities without Labeled Test Data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9530–9546.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface Form Competition: Why the Highest Probability Answer Isn't Always Right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural Instructions: Tuning Language Models with (Almost) No Human Labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14409–14428.

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*.

Saurav Kadavath, Tom Conerly, Amanda Askell, T. J. Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zachary Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, John Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom B. Brown, Jack Clark, Nicholas Joseph, Benjamin Mann, Sam McCandlish, Christopher Olah, and Jared Kaplan. 2022. Language Models (Mostly) Know What They Know. *ArXiv*, abs/2207.05221.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Association for Computational Linguistics*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching Models to Express Their Uncertainty in Words. *Transactions on Machine Learning Research*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

OpenAI. 2022. Introducing ChatGPT. `https://openai.com/blog/chatgpt`. Accessed: 2024-01-15.

OpenAI. 2023. GPT-4 Technical Report.

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

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong,

Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2023. Prompting GPT-3 To Be Reliable. In *International Conference on Learning Representations*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *ArXiv*, abs/2302.13971.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. In *Conference on Neural Information Processing Systems*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, M. Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddharth Deepak Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hanna Hajishirzi, and Daniel Khashabi. 2022. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Albert Webson and Ellie Pavlick. 2022. Do Prompt-Based Models Really Understand the Meaning of Their Prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Mengzhou Xia, Antonios Anastasopoulos, Ruochen Xu, Yiming Yang, and Graham Neubig. 2020. Predicting Performance for Natural Language Processing Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Yuekun Yao and Alexander Koller. 2023. Predicting generalization performance with correctness discriminators. *ArXiv*, abs/2311.09422.

Qinyuan Ye, Harvey Fu, Xiang Ren, and Robin Jia. 2023. How Predictable Are Large Language Model Capabilities? A Case Study on BIG-bench. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Zihuiwen Ye, Pengfei Liu, Jinlan Fu, and Graham Neubig. 2021. Towards More Fine-grained and Reliable NLP Performance Prediction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning*.

Kaitlyn Zhou, Dan Jurafsky, and Tatsunori Hashimoto. 2023. Navigating the Grey Area: How Expressions of Uncertainty and Overconfidence Affect Language

Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5506–5524.

# Appendix

## A Finetuning Details

We use the Huggingface `transformers` library to run all experiments (Wolf et al., 2020). Details for finetuning RoBERTa-base and RoBERTa-large performance predictors can be found in Table 6, while details for the LLaMA-13B performance predictors can be found in Table 7. We perform a hyperparameter search over batch size and learning rate for all finetuning experiments. RoBERTa models are finetuned by updating all parameters, while LLaMA models are finetuned by adding and updating LoRA adapters (Hu et al., 2022). All performance predictors use an additional linear layer applied to either the `[CLS]` token (for RoBERTa) or the EOS token (for LLaMA) at the last layer to make the final prediction. All models were evaluated on the validation set of instructions at every epoch, with early stopping performed based on validation set RMSE.

| Hyperparameter | Assignment |
|---|---|
| number of epochs | 20 |
| batch size | {4, 8, 16} |
| maximum learning rate | {1e-5, 5e-5, 1e-4, 5e-4} |
| optimizer | AdamW |
| epsilon | 1e-8 |
| betas | (0.9, 0.999) |
| learning rate schedule | constant |
| weight decay | 0 |
| warmup proportion | none |
| learning rate decay | none |

Table 6: Experiment settings for finetuning RoBERTa performance predictor models.

| Hyperparameter | Assignment |
|---|---|
| number of epochs | 20 |
| batch size | {8, 16, 32} |
| maximum learning rate | {1e-5, 2e-5, 5e-5, 1e-4} |
| optimizer | AdamW |
| epsilon | 1e-8 |
| betas | (0.9, 0.999) |
| learning rate schedule | linear warmup |
| weight decay | 0 |
| warmup proportion | 0.03 |
| learning rate decay | linear |
| LoRA rank | 256 |
| LoRA alpha | 256 |
| LoRA dropout | 0.05 |

Table 7: Experiment settings for finetuning LLaMA performance predictor models.

## B Details on BIG-bench

The full list of tasks included from BIG-bench can be found in Table 8.

| Tasks |
|---|
| abstract_narrative_understanding, anachronisms, analogical_similarity, analytic_entailment, arithmetic, ascii_word_recognition, authorship_verification, auto_categorization, auto_debugging, bbq_lite_json, bridging_anaphora_resolution_barqa, causal_judgment, cause_and_effect, checkmate_in_one, chess_state_tracking, chinese_remainder_theorem, cifar10_classification, code_line_description, codenames, color, common_morpheme, conceptual_combinations, conlang_translation, contextual_parametric_knowledge_conflicts, crash_blossom, crass_ai, cryobiology_spanish, cryptonite, cs_algorithms, dark_humor_detection, date_understanding, disambiguation_qa, discourse_marker_prediction, disfl_qa, dyck_languages, elementary_math_qa, emoji_movie, emojis_emotion_prediction, empirical_judgments, english_proverbs, english_russian_proverbs, entailed_polarity, entailed_polarity_hindi, epistemic_reasoning, evaluating_information_essentiality, fact_checker, fantasy_reasoning, few_shot_nlg, figure_of_speech_detection, formal_fallacies_syllogisms_negation, gem, gender_inclusive_sentences_german, general_knowledge, geometric_shapes, goal_step_wikihow, gre_reading_comprehension, hhh_alignment, hindi_question_answering, hindu_knowledge, hinglish_toxicity, human_organs_senses, hyperbaton, identify_math_theorems, identify_odd_metaphor, implicatures, implicit_relations, intent_recognition, international_phonetic_alphabet_nli, international_phonetic_alphabet_transliterate, intersect_geometry, irony_identification, kanji_ascii, kannada, key_value_maps, known_unknowns, language_games, language_identification, linguistics_puzzles, logic_grid_puzzle, logical_args, logical_fallacy_detection, logical_sequence, mathematical_induction, matrixshapes, metaphor_boolean, metaphor_understanding, minute_mysteries_qa, misconceptions, mnist_ascii, modified_arithmetic, moral_permissibility, movie_dialog_same_or_different, movie_recommendation, mult_data_wrangling, navigate, nonsense_words_grammar, novel_concepts, object_counting, odd_one_out, operators, paragraph_segmentation, parsinlu_qa, parsinlu_reading_comprehension, penguins_in_a_table, periodic_elements, persian_idioms, phrase_relatedness, physical_intuition, physics, physics_questions, play_dialog_same_or_different, polish_sequence_labeling, presuppositions_as_nli, qa_wikidata, question_selection, real_or_fake_text, reasoning_about_colored_objects, repeat_copy_logic, rephrase, riddle_sense, ruin_names, salient_translation_error_detection, scientific_press_release, semantic_parsing_in_context_sparc, semantic_parsing_spider, sentence_ambiguity, similarities_abstraction, simp_turing_concept, simple_ethical_questions, simple_text_editing, snarks, social_iqa, social_support, sports_understanding, strange_stories, strategyqa, sufficient_information, suicide_risk, swahili_english_proverbs, swedish_to_german_proverbs, symbol_interpretation, temporal_sequences, tense, timedial, topical_chat, tracking_shuffled_objects, understanding_fables, undo_permutation, unit_conversion, unit_interpretation, vitaminc_fact_verification, what_is_the_tao, which_wiki_edit, winowhy, word_sorting, word_unscrambling |

Table 8: List of BIG-bench tasks included when training performance predictors on additional tasks.