

# One-Shot Learning as Instruction Data Prospector for Large Language Models

Anonymous ACL submission

## Abstract

Contemporary practices in instruction tuning often hinge on enlarging data scaling without a clear strategy for ensuring data quality, inadvertently introducing noise that may compromise model performance. To address this challenge, we introduce NUGGETS, a novel and efficient methodology that leverages one-shot learning to discern and select high-quality instruction data from extensive datasets. NUGGETS assesses the potential of individual instruction examples to act as effective one-shot learning instances, thereby identifying those that can significantly improve performance across diverse tasks. NUGGETS utilizes a scoring system based on the impact of candidate examples on the perplexity of a diverse anchor set, facilitating the selection of the most advantageous data for instruction tuning. Through comprehensive evaluations on two benchmarks, including MT-Bench and Alpaca-Eval, we show that instruction tuning with the top 1% of examples curated by NUGGETS substantially outperforms conventional methods employing the entire dataset. For reproducibility, we will release our code and data upon acceptance.

## 1 Introduction

Large language models (LLMs) (Brown et al., 2020; OpenAI, 2023; Google, 2023; Bai et al., 2023; Li et al., 2023a) have showcased remarkable capabilities (Wei et al., 2022; Schaeffer et al., 2023; Liu et al., 2023) across a wide range of language tasks by scaling the model size and training data. Despite their proficiency, it is imperative to further enhance their alignment with human instructions. This alignment process involves supervised fine-tuning (SFT) on input-output pairs, known as *instruction tuning*. Instruction tuning is a crucial step, serving not only to activate the valuable knowledge acquired by LLMs during pre-training but also to facilitate their interaction with humans in a manner that aligns with natural conversational dynamics.

Considerable efforts in instruction tuning have been concentrated on collecting larger (Chung et al., 2022; Wang et al., 2022b), more diverse (Sanh et al., 2022; Sun et al., 2023; Wang et al., 2023b), and intricate (Xu et al., 2023a; Wei et al., 2023) datasets. This is commonly achieved through human crowd-sourcing (Aghajanyan et al., 2021; Ouyang et al., 2022; Tang et al., 2022) or extracting data from larger pre-existing models (Wang et al., 2022a; Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023a). Despite the growth in the size of datasets employed for instruction tuning, certain studies (Zhou et al., 2023; Chen et al., 2023; Cao et al., 2023) suggest that smaller yet valuable datasets tend to be more effective in harnessing the capabilities of LLMs. Blindly expanding the volume of instruction data without ensuring quality may introduce noise and lead to hallucination issues (Zhang et al., 2023c; Zhao et al., 2023a). However, there is a lack of standard criteria for selecting high-quality instruction data (Li and Qiu, 2023; Har-Peled and Mazumdar, 2004; Xia et al., 2023a; Zhang et al., 2024). As depicted in Figure 1, the common practice depends on empirical methods for data selection (Xia et al., 2023b), introducing bias in determining data combinations and adjusting based on outcomes. This trial-and-error approach elevates alignment costs for models. We posit that optimal instruction combinations are present within the extensive data available, yet an efficient and cost-effective identification method remains underexplored.

In this paper, we introduce NUGGETS, a simple yet efficient method that harnesses LLMs as data explorers through one-shot (in-context) learning. This approach enables selecting valuable and high-quality data from expansive instructional datasets. Intuitively, an instructional example holds value in training if it serves as an excellent one-shot demonstration for a specific task. If it can facilitate many tasks, it will be worth being treated as a prime data

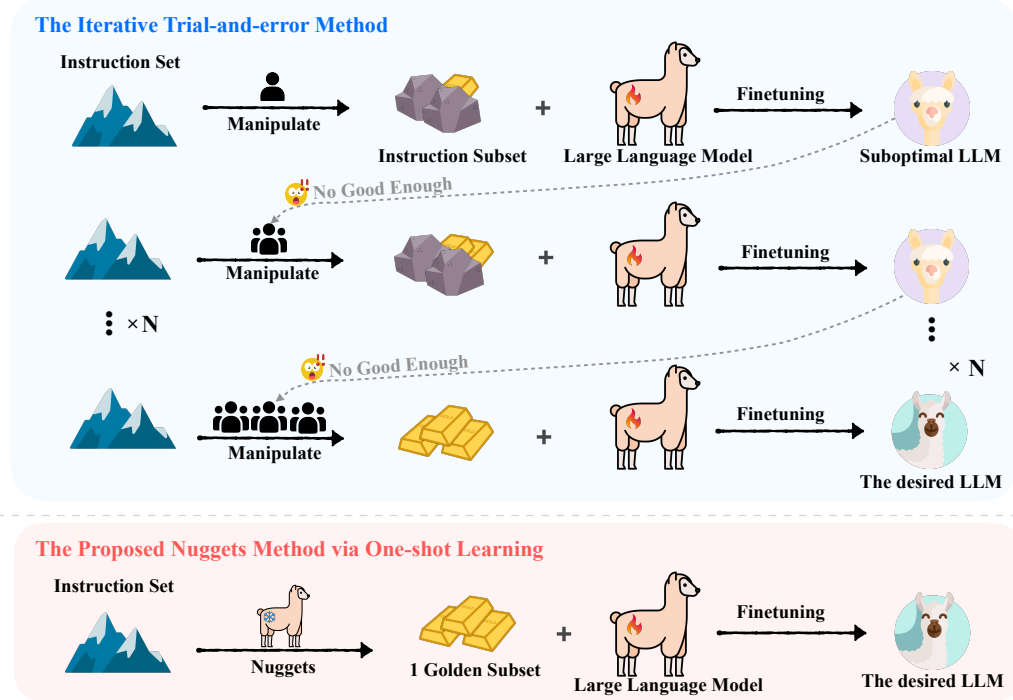


Figure 1: The comparison between our NUGGETS and previous empirical methods. In contrast to empirical methods (blue area), NUGGETS (orange area) can directly sample a gold subset, offering a more direct contribution to model fine-tuning.

focus, i.e., "gold instruction". Another noteworthy perspective arises from the observation that in-context learning (Dai et al., 2022; Yang et al., 2023; Wang et al., 2023a) employs prompting to implicitly fine-tune the model, while instruction tuning operates through gradient descent. Leveraging the performance of in-context learning offers a promising avenue to predict the effects of instruction tuning. Concretely, we first select a set that spans multiple tasks, designated as the anchor set, and the dataset of instructions to be optimized is identified as the candidate set. One example is sequentially chosen from the candidate set to act as a one-shot example for in-context learning. Subsequently, it is scored based on its impact on the perplexity of each anchor example. This scoring mechanism enables the inference of dependencies between anchor and candidate examples, providing a reference standard for data selection.

To evaluate the effectiveness of the proposed NUGGETS, we conduct extensive evaluations on two widely recognized benchmarks, namely MT-Bench (Zheng et al., 2023) and Alpaca-Eval (Li et al., 2023d). We choose a popular and powerful LLM, LLaMA (Touvron et al., 2023a), as our base model. Experimental findings demonstrate that the NUGGETS' data filtering strategy engenders a

significant improvement in comparison to vanilla fine-tuning approaches.

We summarize our main contributions as follows:

- We present NUGGETS, a methodology designed to dynamically assess the quality of instructional examples by using LLMs themselves. NUGGETS is expected to extract the most valuable data from a vast pool of instruction data for the purpose of fine-tuning.
- Fine-tuning LLMs with solely the top 1% of highest-scoring instructional examples yields superior results than using the entire instruction dataset. This observation underscores the significance of prioritizing the quality and strategic composition of the training data over sheer volume.
- The results of extensive experiments substantiate our hypotheses regarding "golden instructions", indicating that the effectiveness of an instructional example is measured by its impact on the task generalization capability of the model following the fine-tuning process. This observation holds considerable promise, potentially providing valuable insights for future endeavors in data quality screening.

## 2 Related Work

**Instruction Tuning** Recent works have introduced a series of techniques that aim to refine large language models (LLMs), showcasing their ability to generalize effectively to instructions not encountered before. For instance, T5 (Raffel et al., 2020) pioneered the initial effort of training various natural language processing (NLP) tasks in a unified text-to-text format. FLAN (Wei et al., 2021) introduced the novel concept of *instruction tuning*, aiming to improve zero-shot task performance by transforming NLP tasks into natural language instructions during model training. Furthermore, InstructGPT (Ouyang et al., 2022) handled a wide array of human-created instructions encompassing diverse forms and a broad range of task types tailored for real-world user scenarios. In the absence of the source code release for these notable projects by OpenAI, subsequent efforts by Alpaca (Taori et al., 2023; Peng et al., 2023) and Vicuna (Chiang et al., 2023) were undertaken to explore open-domain instruction tuning, employing the open-source LLM LLaMA (Touvron et al., 2023a).

**Instruction Construction** The fine-tuning instruction datasets by previous methods are often created manually or tailored to specific tasks. To alleviate the issue of extensive human annotations and manual data gathering, various semi-automated techniques have emerged. Self-Instruct (Wang et al., 2022a) randomly selected a limited number of instances from the initial task pool and used them as demonstrations to guide a language model in generating new instructions, along with their corresponding input-output pairs. Evol-Instruct (Xu et al., 2023a) adopted a progressive modification strategy for the original instructions, which facilitated precise control over the difficulty and complexity levels of the generated instructions. Tree-Instruct (Zhao et al., 2023b), in contrast to Self-Instruct or Evol-Instruct, guided LLMs by instructing them to append a specified number of new nodes to the semantic tree of an existing instruction rather than directly manipulating the text sequence. Conversely, certain investigations are oriented towards augmenting the performance of LLMs by leveraging a reduced yet higher-quality set of instruction examples. LIMA (Zhou et al., 2023) demonstrated remarkably strong performance by strategically selecting a thousand high-quality data points for learning. InstructMining (Cao et al., 2023) introduced a collection of carefully chosen

natural language indicators for evaluating the quality of instruction-following data. Notably, this approach necessitates the division of data into multiple bins. Consequently, it encounters limitations in assessing the quality of individual examples at a fine-grained level. Moreover, ALPAGASUS (Chen et al., 2023) utilized the capabilities of an external and powerful model, ChatGPT, to directly evaluate each example. Despite the proven efficacy of this approach, a notable limitation lies in its inability to account for the inherent variations present in each model subjected to fine-tuning. It predominantly relies on the predilections of ChatGPT. Although Li et al. (2023c) proposed a self-guided method for selecting data in instruction tuning, it still requires preliminary fine-tuning of the model, introducing uncertainty into subsequent operations.

## 3 NUGGETS

**Motivation** As illustrated in Figure 1, the conventional paradigm for enhancing instructional data in the fine-tuning process of large language models (LLMs) has predominantly relied on empirical methods. These methods encompass the application of heuristic rules, expert analysis, and iterative adjustments to the data guided by feedback on model performance. Notably, this trial-and-error approach imposes significant costs in terms of both human effort and computational resources.

Recent scholarly consensus suggests that instruction tuning significantly enhances the task generalization capabilities of pre-trained models across various specific tasks (Longpre et al., 2023; Zhang et al., 2023a,b; Shu et al., 2023). In light of this, we posit the hypothesis of a *golden instruction*: the efficacy of an instructional example is gauged by its influence on the task generalization capability of the model subsequent to the fine-tuning procedure. As the extent of improvement becomes more conspicuous, the instruction gravitates towards classification as “golden instruction”.

According to this hypothesis, a straightforward method involves fine-tuning an independent model using an instruction example and then comparing the performance of the fine-tuned model with the base model on a predefined dataset containing multiple tasks. This process aims to discern whether the given example qualifies as a “golden instruction”. However, this method would lead to an impractical proliferation of fine-tuned models, equivalent to the number of distinct instructions. Further-

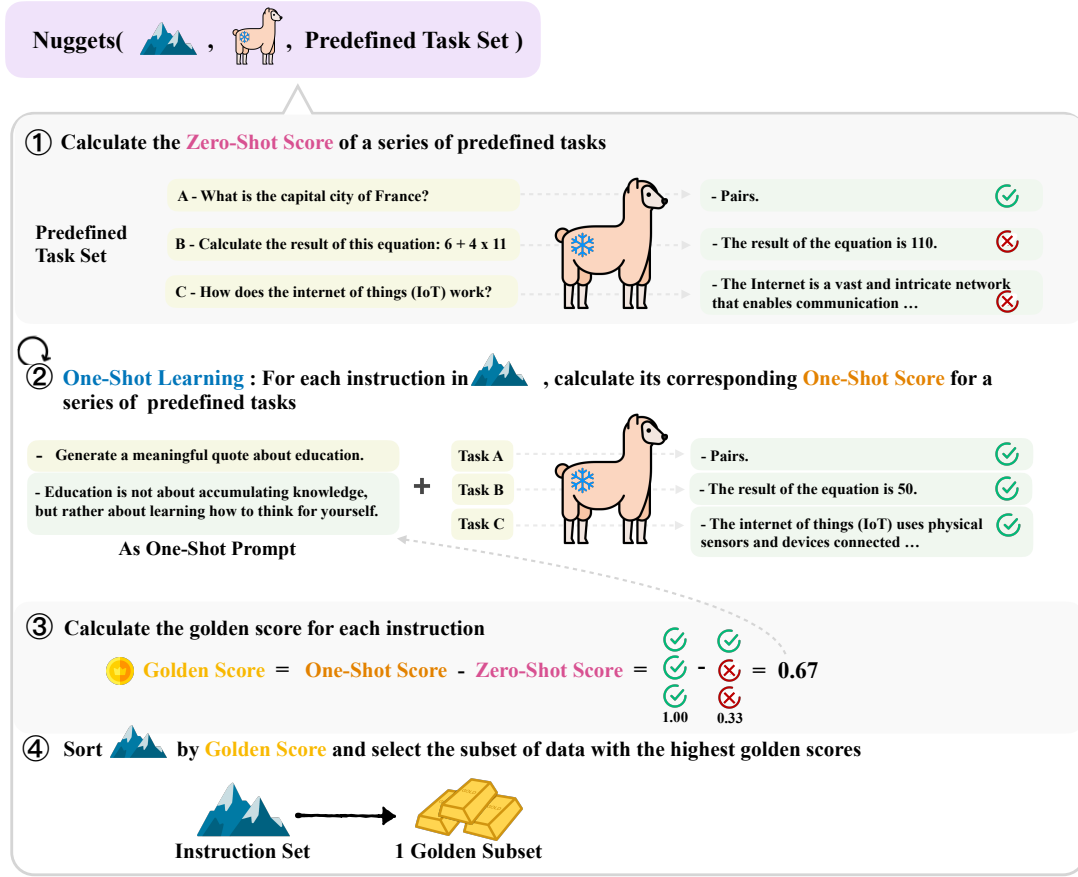


Figure 2: The illustration of the framework of our NUGGETS. Note that we do not directly let the model generate answers for assessment. Instead, we calculate the model’s logit scores on the ground truth answers as zero-shot scores or one-shot scores.

more, fine-tuning with only a single example may introduce unstable updates to the model’s gradients, making it challenging to ascertain the genuine acquisition of the example. Motivated by the inherent duality between *In-Context Learning* (ICL) and gradient descent (Dai et al., 2022; Aizerman et al., 1964; Yang et al., 2023; Irie et al., 2022), we “fine-tune” the instruction implicitly through one-shot learning, replacing the need for actually fine-tuning the model. More information can be found in Appendix A.

**Overview** The framework of our NUGGETS is illustrated in Figure 2. Firstly, we evaluate the proficiency of LLMs across a diverse range of tasks using a predefined set of tasks, denoted as the *zero-shot score*. Subsequently, we designate each example from the instruction dataset as a distinct *one-shot prompt*, concatenating it in front of the predefined tasks. We then recalibrate the model’s completion level for these tasks, referred to as the *one-shot score*. By exploiting the disparity between one-shot and zero-shot scores, we can compute

the golden score for each instruction. Once the golden scores for all instructions are computed, we can identify the highest-scoring subset, deemed the golden subset, which is subsequently provided directly to the model for the fine-tuning process.

### 3.1 Algorithm Details

**Zero-Shot Score** Given a predefined task set, it encompasses a variety of  $m$  tasks, where each task is structured as [Task (T), Answer (A)]. Each word in Task or Answer is denoted as  $w_i^T$  or  $w_i^A$ . Let LLM denote the pre-trained base large language model we use. For the  $j$ -th task that is represented by  $T_j$ , the probability of zero-shot inference by the model can be calculated by continuously predicting the next tokens given the task and their proceeding words:

$$s_{\text{zsl}}^j = \frac{1}{L} \sum_{i=1}^L \log p(w_i^{A_j} | C; \text{LLM}), \quad (1)$$

$$C = [T_j, w_1^{A_j}, w_2^{A_j}, \dots, w_{i-1}^{A_j}],$$



where  $L$  is the number of words of the ground-truth answer  $A$ . The score  $s_{\text{zsl}}^j$  is employed to signify the extent of the model’s proficiency on the  $j$ -th task. A higher  $s_{\text{zsl}}^j$  denotes superior model performance on the  $j$ -th task, whereas a lower  $s_{\text{zsl}}^j$  implies inferior performance. Therefore, we can acquire the model’s performance across  $m$  tasks as:

$$\mathbf{S}_{\text{zsl}} = [s_{\text{zsl}}^1, s_{\text{zsl}}^2, \dots, s_{\text{zsl}}^{m-1}, s_{\text{zsl}}^m]. \quad (2)$$

**One-Shot Score** With an instruction tuning dataset  $\mathcal{D}$ , we aim to identify a set of examples  $\mathcal{D}_{\text{gold}}$  that most closely align with the golden instructions. For each example  $\mathbf{z}_k = [\text{Instruction}_k^Q (\text{IQ}_k), \text{Instruction}_k^A (\text{IA}_k)]$ , we initially perform implicit instruction tuning on the base model using that specific example. Here,  $\text{Instruction}_k^Q$  denotes the question associated with the  $k$ -th example  $\mathbf{z}_k \in \mathcal{D}$ , while  $\text{Instruction}_k^A$  signifies its corresponding answer. Subsequently, we employ the model with in-context learning to conduct another round of testing on the tasks within the predefined task set. That is,

$$s_{\text{it}}^j(\mathbf{z}_k) = \frac{1}{L} \sum_{i=1}^L \log p(w_i^{A_j} | \underbrace{\text{IQ}_k, \text{IA}_k}_{\text{One-Shot Prompt}}, C; \text{LLM}),$$

$$C = [T_j, w_1^{A_j}, w_2^{A_j}, \dots, w_{i-1}^{A_j}], \quad (3)$$

where  $\text{IQ}_k$  and  $\text{IA}_k$  can be considered *one-shot prompt*. Similarly, we can obtain the performance of the model after implicit fine-tuning across  $m$  different tasks:

$$\mathbf{S}_{\text{it}}^k = [s_{\text{it}}^1(\mathbf{z}_k), s_{\text{it}}^2(\mathbf{z}_k), \dots, s_{\text{it}}^{m-1}(\mathbf{z}_k), s_{\text{it}}^m(\mathbf{z}_k)]. \quad (4)$$

Afterward, we use the **Golden Score (GS)** to reflect the impact of this instruction tuning example on the base model. The GS of the example  $\mathbf{z}_k$  is calculated as

$$\text{GS}(\mathbf{z}_k) = \frac{1}{m} \sum_{i=1}^m \mathbb{I}[s_{\text{it}}^i(\mathbf{z}_k) > s_{\text{zsl}}^i] \in [0, 1], \quad (5)$$

where  $\mathbb{I}[\cdot]$  is the indicator function. At a high level, the GS measures the increment of performance improvement of the model after one-shot learning through the given instruction.

In this study, we calculate the GS score for each instructional example, facilitating the generation of a ranked list of scores encompassing the entire

set of examples. Our objective is to explicitly fine-tune the base model by selectively employing a small subset comprising the most pivotal examples. Specifically, we prioritize examples exhibiting high golden scores, aiming to achieve superior outcomes compared to utilizing the entire dataset.

## 4 Experiments

### 4.1 Experimental Setup

**Instruction Dataset** We adopt the Alpaca dataset (Taori et al., 2023) as instruction data. It is an important resource in the open-source community for instruction tuning, which is constructed by employing the self-instruct (Wang et al., 2022a) method to distill instruction data from text-davinci-003. The success of this dataset in fine-tuning the LLaMA model has sparked a series of explorations into instruction fine-tuning (Li et al., 2023b; Ji et al., 2023; Xu et al., 2023b). Besides, we perform more types of instruction datasets to verify the transferability of NUGGETS, please refer to C.

**Predefined Task Set** The predefined task set plays a crucial role in computing golden scores for instructions. These data are employed to evaluate the model’s ability to generalize across diverse tasks. The adequacy of the predefined task set is contingent upon its encompassing a substantial volume of data and incorporating a broad range of tasks. As the Alpaca dataset inherently possesses these attributes, we randomly choose 1,000 examples from it to constitute the predefined task set.

**Evaluation Datasets** This work uses two methods to assess the model’s capabilities. The first approach involves rating the responses generated by models on a scale ranging from 1 to 10. For this purpose, we utilize the GPT-4 labeled **MT-Bench** (Zheng et al., 2023) dataset, which evaluates instruction-following proficiency across eight categories: writing, roleplay, extraction, reasoning, math, coding, STEM, and humanities. Notably, since we only fine-tune on single-turn instruction data, the evaluation is restricted to *Turn 1* of MT-Bench, similar to previous studies (Cao et al., 2023; Zheng et al., 2023; Chen et al., 2023). The second method involves comparing the model’s generated responses with those produced by the Davinci-003 model, employing the well-established **Alpaca-Eval** dataset (Li et al., 2023d). This dataset adopts the “win\_rate” as the evaluation metric.

Model	Nums	Helpful_Base	Koala	Self-instruct	Oasst	Vicuna	Length	Results
LLaMA	-	0.00	1.28	1.19	0.53	1.25	2,980	0.87
Alpaca <sub>full</sub>	52,002	20.15	25.64	27.77	25.00	15.00	396	25.43
Alpaca <sub>≤0.5</sub>	9,542	7.75	5.12	13.09	9.57	8.75	241	10.96
Alpaca <sub>&gt;0.5</sub>	42,460	24.03	20.51	28.57	29.78	15.00	413	26.06
Alpaca <sub>&gt;0.8</sub>	7,525	34.10	30.76	30.95	35.10	30.00	519	<b>32.48</b>
Alpaca <sub>&gt;0.85</sub>	619	37.20	26.90	25.00	29.30	22.50	617	<u>28.20</u>

Table 1: The win\_rate results of various models under the Alpaca-Eval benchmark evaluation.

Model	Writing	Roleplay	Reasoning	Math	Coding	Extraction	STEM	Humanities	Overall
LLaMA	4.6	4.5	5.2	1.0	1.20	2.2	5.0	4.1	3.47
Alpaca <sub>full</sub>	8.5	5.8	3.3	1.0	2.0	4.5	6.5	7.1	4.83
Alpaca <sub>≤0.5</sub>	7.2	5.1	2.1	1.3	1.9	5.5	5.3	6.9	4.41
Alpaca <sub>&gt;0.5</sub>	8.3	5.7	3.5	1.1	1.7	5.0	6.6	7	4.86
Alpaca <sub>&gt;0.8</sub>	8.3	5.9	5.6	1.8	2.5	4.0	7.3	7.4	<b>5.34</b>
Alpaca <sub>&gt;0.85</sub>	6.6	6.3	4.9	1.0	2.3	3.3	6.3	7.3	<u>4.87</u>

Table 2: Experimental results of various models on the GPT-4 labeled MT-Bench benchmark.

**Implementation Details** In our experiments, we designate the LLaMA-7B model as the foundational model. To ensure a fair comparison, we also set the maximum input length for the models fine-tuned with the Alpaca dataset to be consistent with LLaMA, which is 2048. In the model fine-tuning phase, we employ the Adam optimizer with a learning rate of  $2 \times 10^{-5}$  and utilize a batch size of 64, conducting training over three epochs. In the subsequent model evaluation phase, we maintain all parameter settings consistent with the original work (Li et al., 2023d; Zheng et al., 2023).

## 4.2 Experimental Results

The Alpaca dataset comprises a total of 52,002 instruction examples, and the distribution of their golden scores is illustrated in Appendix B. Among these examples, 42,460 instances exhibit a golden score surpassing 0.5. In addition, a subset of examples closely aligned with the golden instructions has been selected, specifically those attaining golden scores above 0.8 and 0.85. In particular, there are 7,525 examples with golden scores surpassing 0.8 and 619 examples with golden scores exceeding 0.85. Notably, the latter subset constitutes a mere 1% of the entire dataset.

We conduct instruction tuning on the LLaMA model using various subsets of examples distinguished by their golden scores: those with scores less than 0.5, greater than 0.5, greater than 0.8, greater than 0.85, and the complete dataset. The fine-tuned models are denoted as

Alpaca<sub>≤0.5</sub>, Alpaca<sub>>0.5</sub>, Alpaca<sub>>0.8</sub>, Alpaca<sub>>0.85</sub>, and Alpaca<sub>full</sub>, respectively.

**Main Results** The experimental results are presented in Table 1 and Table 2 for the Alpaca-Eval and MT-Bench benchmarks, respectively. As expected, Alpaca<sub>>0.8</sub> produces the most impressive outcomes. This can be attributed to its ability to maintain an optimal balance between the volume and quality of the instructions it utilizes, leading to the most desirable results. We also note that incorporating lower-quality instructions adversely affected model fine-tuning. This trend is clear when we see that Alpaca<sub>≤0.5</sub> lagged behind Alpaca<sub>full</sub> in performance, while Alpaca<sub>>0.5</sub> shows a slight edge over Alpaca<sub>full</sub>. Remarkably, Alpaca<sub>>0.85</sub>, using only 1% of the dataset for fine-tuning, achieved results comparable to or even surpassing those of Alpaca<sub>full</sub>. This underscores the efficacy of our data selection method. More qualitative results can be found in Appendix D.

**Ablation on Predefined Task Sets** To evaluate how different predefined task sets affect the selection of instruction data for fine-tuning, we include two additional predefined task set variations. One is randomly sampled from the Alpaca dataset but with a smaller task set size, which is limited to 100 examples. The other one entails clustering the Alpaca dataset into 100 clusters using the K-Means algorithm and selecting the centroids of each cluster as examples of the task set.

We use the two predefined sets to calculate

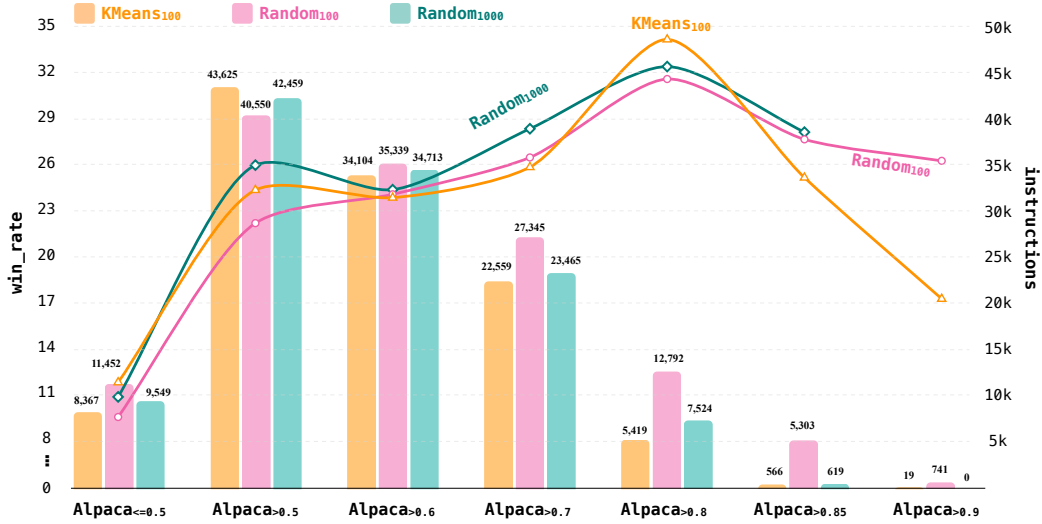


Figure 3: The distribution of the golden score for the instruction dataset across different predefined task sets, along with the corresponding fine-tuning results on the Alpaca-Eval benchmark.

Predefined Task Set	Alpaca $\leq 0.5$	Alpaca $> 0.5$	Alpaca $> 0.6$	Alpaca $> 0.7$	Alpaca $> 0.8$	Alpaca $> 0.85$	Alpaca $> 0.9$
K-Means <sub>100</sub>	11.91	24.44	23.94	25.93	34.25	25.25	17.35
Random <sub>100</sub>	9.65	22.28	24.16	26.56	31.67	27.74	26.34
Random <sub>1000</sub>	10.96	26.06	24.46	28.43	32.48	28.21	-

Table 3: Win\_rate results on Alpaca-Eval Benchmark across different predefined task sets

golden scores for the Alpaca dataset separately. The distribution of golden scores is depicted in Figure 3. We select instruction data with golden scores less than or equal to 0.5, greater than 0.5, greater than 0.6, greater than 0.7, greater than 0.8, greater than 0.85, and greater than 0.9 for model fine-tuning, respectively. Table 3 suggests that with random sampling, increasing the size of the task set can enhance the identification of high-quality instruction data. The logic behind this is that a larger encompasses a broader diversity of data, facilitating a more nuanced assessment of an instruction’s effect on model task generalization. However, a shift occurs when K-Means is employed to cherry-pick more distinct examples for the task set. With as few as 100 examples, K-Means outshines the results from 1,000 examples acquired through random sampling. In this instance, Alpaca $>0.8$  delivered a superior performance with just 5,419 examples, compared to the 7,524 examples seen with Random<sub>1000</sub>. This outcome also indirectly confirms the validity of our hypothesis regarding the definition of golden instructions.

**Ablation on Instruction Sets** To delve deeper into the generalization capabilities of NUGGETS across varied instruction datasets, we undertake a

series of experiments utilizing the Alpaca-GPT4 dataset (Peng et al., 2023). It generates instructional data from the powerful GPT-4 model (OpenAI, 2023), which is considered to have superior data quality. Additionally, it shares the same questions in instructions with the Alpaca dataset, which facilitates our direct comparison between the two.

Inspired by Table 3, we employ the K-Means algorithm on the Alpaca-GPT4 dataset to sample 100 examples, forming the predefined task set. Subsequently, we apply the NUGGETS method to score all instructions in the dataset with the golden score, as depicted in Figure 4. Compared to the Alpaca dataset, the Alpaca-GPT4 dataset boasts a higher number of instructions with golden scores: 25,100 instructions exceed a score of 0.8, 16,943 surpass 0.85, and 4,250 instructions exceed 0.9. These numbers far exceed the corresponding high-scoring instructions in the Alpaca dataset. This also demonstrates that the golden score can serve as an absolute metric to assess the quality of instructional data. The results from model fine-tuning indicate that on the Alpaca-GPT4 dataset, conclusions align with those of previous experiments. The large language models fine-tuned on subsets with golden scores less than or equal to 0.5 exhibit the poorest

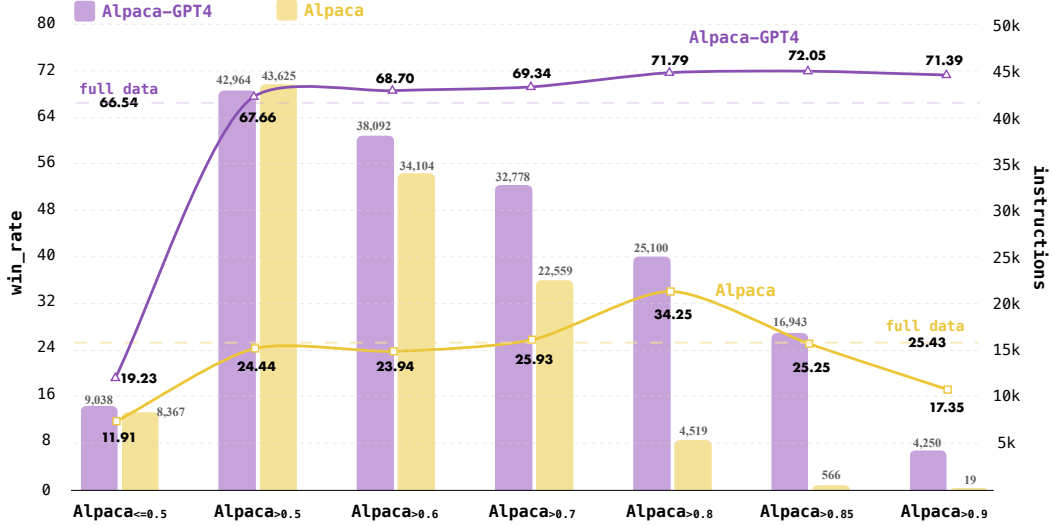


Figure 4: The distribution of the golden score for the instruction dataset across different instruction sets, along with the corresponding fine-tuning results on the Alpaca-Eval benchmark. Both predefined task sets utilize K-Means to sample 100 examples from their respective instruction datasets.

		GS $\leq 0.5$	GS $> 0.5$	GS $> 0.6$	GS $> 0.7$	GS $> 0.8$	GS $> 0.85$	GS $> 0.9$	Full Data
LLaMA2	NUM	3,730	48,272	40,905	28,644	10,409	2,411	87	52,002
	Win_Rate	13.17	27.09	27.85	27.62	<u>33.92</u>	<b>34.98</b>	27.08	26.47
Mistral	NUM	78	51,924	51,610	49,398	36,068	23,147	9,356	52,002
	Win_Rate	0	12.26	11.10	<u>12.45</u>	11.28	10.60	<b>13.53</b>	9.85

Table 4: Win\_rate results on Alpaca-Eval Benchmark across two different foundation models.

performance, with a win rate of only 19.23% in the Alpaca-Eval benchmark. In contrast, the models fine-tuned on subsets with golden scores greater than 0.85 demonstrate superior performance, boasting a high win rate of 72.05%. This success can be attributed to the dual assurance of quantity and quality in this particular subset. It is worth emphasizing that fine-tuning on a small and high-quality dataset consistently and significantly outperforms the results of fine-tuning on the full dataset. Overall, the models fine-tuned using Alpaca-GPT4 significantly outperform those fine-tuned with Alpaca. This implicitly corroborates the superior quality of the Alpaca-GPT4 dataset compared to the Alpaca dataset. For more experiments on instruction datasets, please refer to Appendix C.

**Ablation on Foundation Models** To verify the transferability of the NUGGETS method, we conducted experiments on different foundation models using the Alpaca instruction dataset. We selected LLaMA2 (Touvron et al., 2023b) and Mistral (Jiang et al., 2023) at the 7B size as the new base models. The distribution of the golden scores and the performance of models fine-tuned on corresponding

subsets of instructions are shown in Table 4. We found that the NUGGETS method is also applicable to other models. LLaMA2 achieved the best results under fine-tuning on subsets with a golden score greater than 0.85, reaching 34.98, which is significantly higher than the 26.47 achieved under full data. Although the absolute value of the win\_rate for the Mistral series of fine-tuned models is somewhat low, their performance is also significantly boosted by the NUGGETS data filtering.

## 5 Conclusion

This paper presents NUGGETS, a method leveraging LLMs to discern more pivotal data for instruction tuning. Grounded in one-shot learning, this approach facilitates the identification of examples' value, enabling efficient data selection without dependence on additional annotation and associated costs. Benefiting from NUGGETS, we observe improved instruction following abilities even with smaller training subsets. Furthermore, we posit that our method underscores the significance of meticulous data selection, offering valuable insights for future instruction fine-tuning endeavors.



## Limitations

Although the efficacy of the proposed approach has been confirmed through empirical experiments, opportunities for refinement persist. One avenue for improvement involves a thorough investigation into the inclusion of a diverse and compact set of *predefined tasks* during the golden scoring phase. This exploration aims to enhance the efficiency of model evaluation on instructional data, leading to improved identification of high-quality instructions suitable for subsequent model fine-tuning. Secondly, due to resource constraints, the majority of experiments in this study are confined to the LLaMA-7B model. While this model holds significant influence within the large language model open-source community, comprehensive validation across a broader spectrum of models is imperative to ensure the generalizability of the proposed approach. Lastly, to fortify the empirical foundation of our findings, it is crucial to subject the proposed method to validation on a more extensive array of instructional datasets. This step aims to ascertain the robustness and applicability of the methodology across a diverse range of instructional contexts, contributing to its broader utility in real-world scenarios. These outlined avenues for future work are anticipated to refine and extend the scope of our proposed method.

## References

Armen Aghajanyan, Ankit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. In *EMNLP*, pages 5799–5811.

MA Aizerman, EM Braverman, and LI Rozonoer. 1964. Theoretical foundation of potential functions method in pattern recognition autom. *Remote Contr.*

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher 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. *arXiv preprint arXiv:2005.14165*.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. <https://github.com/sahil280114/codealpaca>.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Sriniwasan, Tianyi Zhou, Heng Huang, et al. 2023. Alpapasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*.

Google. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Sariel Har-Peled and Soham Mazumdar. 2004. On core-sets for k-means and k-median clustering. In *STOC*, pages 291–300.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. 2022. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *ICML*, pages 9639–9659.

Yunjie Ji, Yong Deng, Yan Gong, Yiping Peng, Qiang Niu, Lei Zhang, Baochang Ma, and Xiangang Li. 2023. Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases. *arXiv preprint arXiv:2303.14742*.

629	Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. <i>arXiv preprint arXiv:2310.06825</i> .	683
630		684
631		685
632		686
633		687
634	Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for" mind" exploration of large scale language model society. In <i>NeurIPS</i> .	688
635		689
636		690
637		691
638		692
639	Lei Li, Yuwei Yin, Shicheng Li, Liang Chen, Peiyi Wang, Shuhuai Ren, Mukai Li, Yazheng Yang, Jingjing Xu, Xu Sun, et al. 2023b. M3it: A large-scale dataset towards multi-modal multilingual instruction tuning. <i>arXiv preprint arXiv:2306.04387</i> .	693
640		694
641		695
642		696
643		697
644	Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023c. <i>From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning</i> .	698
645		699
646		700
647		701
648		
649	Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning. In <i>Findings of EMNLP</i> , pages 6219–6235.	702
650		703
651		704
652	Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023d. AlpacaEval: An automatic evaluator of instruction-following models. <a href="https://github.com/tatsu-lab/alpaca_eval">https://github.com/tatsu-lab/alpaca_eval</a> .	705
653		706
654		707
655		
656		
657	Peiyu Liu, Zikang Liu, Ze-Feng Gao, Dawei Gao, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji rong Wen. 2023. Do emergent abilities exist in quantized large language models: An empirical study. <i>arXiv preprint arXiv:2307.08072</i> .	708
658		709
659		710
660		711
661		
662	Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. <i>arXiv preprint arXiv:2301.13688</i> .	712
663		713
664		714
665		715
666		716
667	Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. WizardCoder: Empowering code large language models with evol-instruct. <i>arXiv preprint arXiv:2306.08568</i> .	717
668		718
669		719
670		720
671		721
672	OpenAI. 2023. Gpt-4 technical report. <i>arXiv preprint arXiv:2303.08774</i> .	722
673		
674	Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In <i>NeurIPS</i> , pages 27730–27744.	723
675		724
676		725
677		726
678		727
679		728
680	Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. <i>arXiv preprint arXiv:2304.03277</i> .	729
681		730
682		731
		732
	Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. <i>The Journal of Machine Learning Research</i> , 21(1):5485–5551.	733
		734
		735
		736
		737
		738
	Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In <i>ICLR</i> .	
	Rylan Schaeffer, Brando Miranda, and Oluwasanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? <i>arXiv preprint arXiv:2304.15004</i> .	
	Manli Shu, Jiong Xiao Wang, Chen Zhu, Jonas Geiping, Chaowei Xiao, and Tom Goldstein. 2023. On the exploitability of instruction tuning. <i>arXiv preprint arXiv:2306.17194</i> .	
	Zhiqing Sun, Yikang Shen, Qinzhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. <i>arXiv preprint arXiv:2305.03047</i> .	
	Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Mvp: Multi-task supervised pre-training for natural language generation. <i>arXiv preprint arXiv:2206.12131</i> .	
	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. <a href="https://github.com/tatsu-lab/stanford_alpaca">https://github.com/tatsu-lab/stanford_alpaca</a> .	
	Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. <i>arXiv preprint arXiv:2302.13971</i> .	
	Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. <i>arXiv preprint arXiv:2307.09288</i> .	
	Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In <i>EMNLP</i> .	
	Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023b. How far can camels go? exploring the state of instruction tuning on open resources. <i>arXiv preprint arXiv:2306.04751</i> .	

739	Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hananeh Hajishirzi. 2022a. Self-instruct: Aligning language model with self generated instructions. <i>arXiv preprint arXiv:2212.10560</i> .	794
740		795
741		
742		
743		
744	Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022b. Supernaturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In <i>EMNLP</i> , pages 5085–5109.	
745		
746		
747		
748		
749		
750		
751	Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In <i>ICLR</i> .	
752		
753		
754		
755	Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. <i>Transactions on Machine Learning Research</i> .	
756		
757		
758		
759		
760		
761		
762	Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. <i>arXiv preprint arXiv:2312.02120</i> .	
763		
764		
765	Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, and Tongliang Liu. 2023a. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In <i>ICLR</i> .	
766		
767		
768		
769	Xiaobo Xia, Jiale Liu, Shaokun Zhang, Qingyun Wu, and Tongliang Liu. 2023b. Coreset selection with prioritized multiple objectives. <i>arXiv preprint arXiv:2311.08675</i> .	
770		
771		
772		
773	Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. <i>arXiv preprint arXiv:2304.12244</i> .	
774		
775		
776		
777		
778	Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023b. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. <i>arXiv preprint arXiv:2304.01196</i> .	
779		
780		
781		
782	Jiaxi Yang, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Iterative forward tuning boosts in-context learning in language models. <i>arXiv preprint arXiv:2305.13016</i> .	
783		
784		
785		
786	Shaokun Zhang, Xiaobo Xia, Zhaoqing Wang, Linghao Chen, Jiale Liu, Qingyun Wu, and Tongliang Liu. 2024. Ideal: Influence-driven selective annotations empower in-context learners in large language models. In <i>ICLR</i> .	
787		
788		
789		
790		
791	Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023a. Instruction tuning for large language models: A survey. <i>arXiv preprint arXiv:2308.10792</i> .	
792		
793		
	Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Kai Chen, and Ping Luo. 2023b. Gpt4roi: Instruction tuning large language model on region-of-interest. <i>arXiv preprint arXiv:2307.03601</i> .	796
		797
		798
		799
		800
	Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023c. Siren’s song in the ai ocean: A survey on hallucination in large language models. <i>arXiv preprint arXiv:2309.01219</i> .	801
		802
		803
		804
		805
	Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023a. A survey of large language models. <i>arXiv preprint arXiv:2303.18223</i> .	806
		807
		808
		809
		810
	Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L Zhang. 2023b. A preliminary study of the intrinsic relationship between complexity and alignment. <i>arXiv preprint arXiv:2308.05696</i> .	811
		812
		813
		814
		815
	Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. <i>arXiv preprint arXiv:2306.05685</i> .	816
		817
		818
		819
		820
	Chunting Zhou, Pengfei Liu, Puxin Xu, Srinu Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. <i>arXiv preprint arXiv:2305.11206</i> .	821
		822
		823
		824

## A Discussion: One-Shot Learning as Implicit Instruction Tuning

Transformer has risen as the prevailing architecture for language models, where self-attention plays a crucial role as a pivotal element within Transformer. Let  $\mathbf{X}_{\text{ins}}, \mathbf{X}_{\text{test}} \in \mathbb{R}^{d_{\text{in}}}$  denote the instruction tuning sample and the test input respectively.  $\mathbf{X}_{\text{ins}}$  can be likened to  $\text{IQ}_k$  and  $\text{IA}_k$  in Equation 3, while  $\mathbf{X}_{\text{test}}$  can be seen as  $\text{T}$  and  $w_1^A, w_2^A, \dots, w_{i-1}^A$ . That  $\mathbf{Q} = \mathbf{W}_Q \mathbf{X}_{\text{test}}^\top$  be the attention query vector,  $\mathbf{K} = \mathbf{W}_K [\mathbf{X}_{\text{ins}} \parallel \mathbf{X}_{\text{test}}]$  be the attention key vector and  $\mathbf{V} = \mathbf{W}_V [\mathbf{X}_{\text{ins}} \parallel \mathbf{X}_{\text{test}}]$  be the attention value vector, where  $\parallel$  represents concatenation operation,  $\mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_Q \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$  are the projection matrices for computing the attention queries, keys, and values, respectively. The result of self-attention in an arbitrary layer for a head is formulated as:

$$\begin{aligned}
 \text{Attention}(\mathbf{K}, \mathbf{V}, \mathbf{Q}) &= \mathbf{W}_V [\mathbf{X}_{\text{ins}} \parallel \mathbf{X}_{\text{test}}] \text{Softmax} \left( \frac{\mathbf{W}_K [\mathbf{X}_{\text{ins}} \parallel \mathbf{X}_{\text{test}}]^\top \mathbf{Q}}{\sqrt{d_{\text{in}}}} \right) \\
 &\approx \mathbf{W}_V [\mathbf{X}_{\text{ins}} \parallel \mathbf{X}_{\text{test}}] (\mathbf{W}_K [\mathbf{X}_{\text{ins}} \parallel \mathbf{X}_{\text{test}}])^\top \mathbf{Q} \\
 &= \underbrace{\mathbf{W}_V \mathbf{X}_{\text{test}} (\mathbf{W}_K \mathbf{X}_{\text{test}})^\top}_{\text{Only test input.}} \mathbf{Q} + \underbrace{\mathbf{W}_V \mathbf{X}_{\text{ins}} (\mathbf{W}_K \mathbf{X}_{\text{ins}})^\top}_{\text{Only instruction sample.}} \mathbf{Q} \\
 &= \mathbf{W}_{\text{zsl}} \mathbf{Q} + \Delta \mathbf{W}_{\text{iit}} \mathbf{Q} \\
 &= (\mathbf{W}_{\text{zsl}} + \Delta \mathbf{W}_{\text{iit}}) \mathbf{Q},
 \end{aligned} \tag{6}$$

where  $\sqrt{d_{\text{in}}}$  serves as a scaling factor. The term  $\mathbf{W}_V \mathbf{X}_{\text{test}} (\mathbf{W}_K \mathbf{X}_{\text{test}})^\top$  could be denoted as  $\mathbf{W}_{\text{zsl}}$ , which represents the zero-shot learning scenario where no instruction tuning is performed since it solely focuses on the test input. In addition, the term  $\mathbf{W}_V \mathbf{X}_{\text{ins}} (\mathbf{W}_K \mathbf{X}_{\text{ins}})^\top$  can be seen as implicit instruction tuning  $\Delta \mathbf{W}_{\text{iit}}$  achieved via the meta-gradient (Dai et al., 2022; Yang et al., 2023) derived from the instruction sample. Readers can refer to previous papers (Dai et al., 2022; Aizerman et al., 1964; Irie et al., 2022) for more details on implicit instruction tuning.

## B The Distribution of Golden Score

As shown in Figure 5, in a total of 52,002 cases, there are 9,549 instructions with a gold score of less than 0.5, indicating that these data have a side effect on overall task completion. Besides, there are 7,524 instructions with a gold score greater than 0.8, suggesting that the model improves the task completion rate through one-shot learning from these data, which can be considered high-quality instruction data.

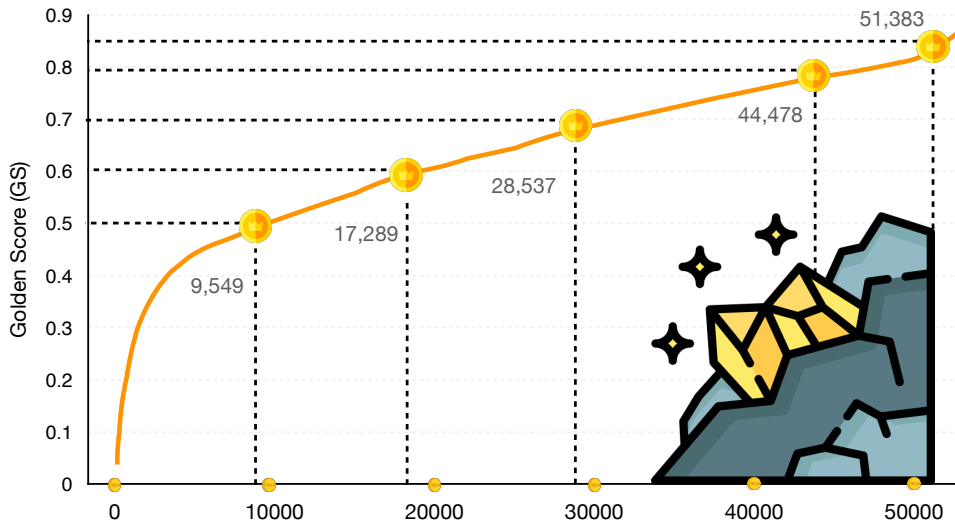


Figure 5: The distribution of the golden score for the Alpaca instruction dataset.



## C Experiment on Other Instruction Sets

Based on the LLaMA-7B model, we conducted experiments on several other instruction datasets, further validating the effectiveness of our NUGGETS method.

### C.1 Code Alpaca

The Code Alpaca instruction dataset (Chaudhary, 2023) is designed to develop large language models capable of following instructions and generating code. Leveraging self-instruct (Wang et al., 2022a) technology, it has produced 20,000 examples of instruction data. We use HumanEval (Chen et al., 2021) as a benchmark to evaluate the model’s code generation capabilities. It is used to measure functional correctness for synthesizing programs from docstrings. It consists of 164 original programming problems, assessing language comprehension, algorithms, and simple mathematics, with some being comparable to simple software interview questions. We adopt the approach outlined by Chen et al. (2021) to calculate pass rates at k values of 1, 10, and 100 for each problem. Essentially, pass@1 predicts the probability of a model producing a correct solution on the first try, while pass@10 and pass@100 predict the probability of achieving a correct solution within 10 and 100 tries, respectively. We generate 200 completions at a temperature setting of 0.2 (Luo et al., 2023) to estimate pass@1, pass@10, and pass@100 rates. The

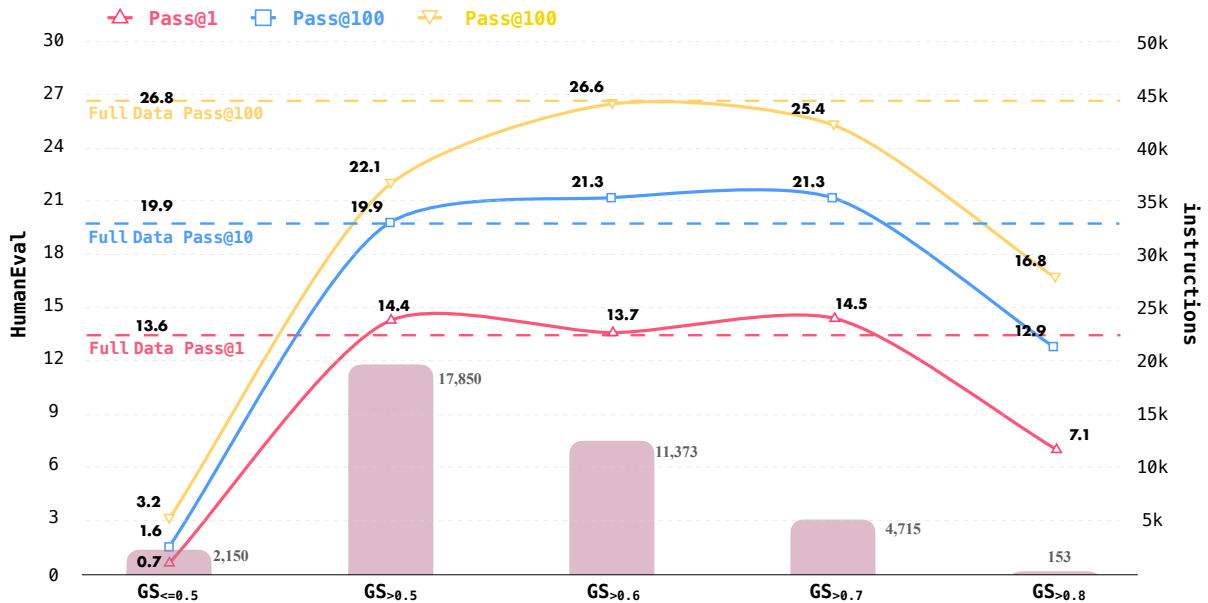


Figure 6: The distribution of the golden score for the Code Alpaca instruction dataset, along with the corresponding fine-tuning results on the HumanEval benchmark. Predefined task sets utilize K-Means to sample 100 examples from the Code Alpaca instruction dataset.

experimental results are shown in the Figure 6. Out of 20,000 instructions, 4,715 instructions have a gold score greater than 0.85, achieving the best pass@1 and pass@10 results in the HumanEval benchmark, superior to the fine-tuning results of the full dataset. Additionally, this experiment also proves that the NUGGETS method can be applied to fine-tuning for specific tasks, demonstrating good transferability.

### C.2 WizardLM

The WizardLM instruction dataset (Xu et al., 2023a), which employs Evol-Instruct to iteratively refine an initial set of instructions into more complex ones, contains 70,000 instruction examples. The distribution of the golden scores and the performance of models fine-tuned on corresponding subsets of instructions are shown in Table 5. We can observe that the quality distribution of the WizardLM dataset is relatively balanced, with 65,190 instruction examples having a golden score greater than 0.8, accounting for 93% of the total number of instructions. In the evaluation of the Alpaca-Eval benchmark, models fine-tuned on

	$GS_{\leq 0.5}$	$GS_{>0.5}$	$GS_{>0.6}$	$GS_{>0.7}$	$GS_{>0.8}$	$GS_{>0.85}$	$GS_{>0.86}$	$GS_{>0.87}$	Full Data
<b>NUM</b>	480	69,520	69,377	68,898	65,190	40,223	23,579	3,316	70,000
<b>Win_Rate</b>	19.42	58.08	57.40	56.21	<b>59.81</b>	<u>58.40</u>	57.81	54.68	57.65

Table 5: The distribution of golden scores for the WizardLM dataset and the evaluation results of models fine-tuned on corresponding score subsets on the Alpaca-Eval benchmark.

subsets with golden scores greater than 0.8 achieved a win rate of 59.81, outperforming models fine-tuned on the full dataset.

### C.3 FLANv2

We sampled 50,000 examples from the FLANv2 (Chung et al., 2022) dataset to constitute the instruction tuning data for this experiment. Additionally, the Predefined task set was also derived from these 50,000 examples, using the K-Means algorithm to sample 100 examples. We evaluated the performance of the fine-tuned model using MMLU (Hendrycks et al., 2020) in a 5-shot setting. MMLU is a test designed to measure a text model’s multitask accuracy. The test encompasses 57 tasks, including elementary mathematics, US history, computer science, law, and more. The experimental results are shown in the

	$GS_{\leq 0.5}$	$GS_{>0.5}$	$GS_{>0.6}$	$GS_{>0.7}$	$GS_{>0.8}$	$GS_{>0.85}$	$GS_{>0.9}$	Full Data
<b>NUM</b>	1,361	48,639	46,009	39,046	17,037	4,798	321	50,000
<b>Acc</b>	34.68	41.38	<u>41.92</u>	41.87	<b>41.97</b>	35.51	26.41	40.45

Table 6: The distribution of golden scores for the sampled FLANv2 dataset and the evaluation results of models fine-tuned on corresponding score subsets on the MMLU benchmark.

table. It can be observed that the model fine-tuned with examples having a golden score greater than 0.8 (totaling 17,037 examples) achieved the best results, followed by those fine-tuned with examples having a golden score greater than 0.6 (totaling 46,009 examples). Additionally, we noted that the model fine-tuned with examples having a golden score less than 0.5 (totaling 1,361 examples) outperformed the model fine-tuned with examples having a golden score greater than 0.9 (totaling 321 examples).

We speculate that the reason for this outcome is that the MMLU evaluation itself is a knowledge-intensive assessment, which thus imposes higher demands on the quantity of fine-tuning examples. In the future, some optimizations in the selection of the Predefined Task Set may potentially improve this situation.

## D Case Study

To evaluate NUGGETS qualitatively, we also select some instruction examples from the Alpaca dataset for case studies, as shown in Figure 7. We observe that noisy instructions without outputs display extremely low golden score values. Such examples consistently score very low throughout the instruction dataset. Additionally, instructions with hallucinatory outputs and some that are a bit confusing also typically exhibit lower golden scores. In contrast, instructions with higher scores often possess fluent language and meticulous logic, and are oriented towards assisting humans in problem-solving. Therefore, we believe that the high golden scores can represent high-quality instruction tuning data.


[ Instruction ]	Construct a timeline to show the development of artificial intelligence over the last 5 years.
[ Output ]	" "
[ Golden Score ]	 0.046
[ Instruction ]	Create a pictogram to show the meaning of the following phrase: "The wheels on the bus go round and round."
[ Output ]	[Picture of a bus with circles around its wheels]
[ Golden Score ]	 0.193
[ Instruction ]	Write six original rhymes for the provided word.
[ Input ]	Skate
[ Output ]	<ol style="list-style-type: none"> <li>1. To ride a skate is where I'd like to go,It zooms past skiers, 'cause I'm so pro.</li> <li>2. The skateboarding ramp seemed so tall,But I made it over with no stall.</li> <li>3. I take a ride on my trusty skate,And carve the roads with a twist of fate.</li> <li>4. I spin around the rink so full,Check me out, I'm lookin' cool.</li> <li>5. You don't believe I can skate on the sand,But watch me do it with a little hand!</li> <li>6. Skateboard's my name and it's my game, [...]</li> </ol>
[ Golden Score ]	 0.403
[ Instruction ]	What are three qualities that a good product manager should possess ?
[ Output ]	A good product manager should possess strong communication skills, a strategic mindset, and the ability to collaborate with different members of the organization.
[ Golden Score ]	 0.803
[ Instruction ]	Create a fact sheet about AI.
[ Output ]	<p>What is AI?</p> <p>AI (Artificial Intelligence) is the science and engineering of programming machines to think and act on their own.</p> <p>What can AI do?</p> <p>AI can be used to automate mundane tasks, analyze complex data and patterns, identify trends, provide personalized services, and more.</p> <p>Who uses AI?</p> <p>AI is used by many industries, [...]</p>
[ Golden Score ]	 0.803
[ Instruction ]	Describe the following place. Output the description.
[ Input ]	The Grand Canyon
[ Output ]	The Grand Canyon is one of the most incredible natural wonders in the world. The canyon spans 277 miles in length and can be up to 18 miles wide in certain places. With towering cliff walls, evergreen forests, and the mighty Colorado River running its length, it truly is an awe-inspiring sight. The area around the canyon is home to diverse wildlife, including endangered species like the California condor and the desert tortoise. Visiting the Grand Canyon is an unforgettable experience that is sure to take your breath away.
[ Golden Score ]	 0.859

Figure 7: Examples of instructions and their corresponding golden scores.