# MAYBE ONLY 0.5% DATA IS NEEDED: A PRELIMINARY EXPLORATION OF LOW TRAINING DATA INSTRUCTION TUNING

A PREPRINT

**Hao Chen** *
Zhejiang University
h.c.chen@zju.edu.cn

**Yiming Zhang** *
Zhejiang University
yimingz@zju.edu.cn

**Qi Zhang** *
Zhejiang University
cheung_se@zju.edu.cn

**Hantao Yang** *
Zhejiang University
ht.yang@zju.edu.cn

**Xiaomeng Hu**
Zhejiang University

**Xuetao Ma**
ZhongHao XinYing (Hangzhou) Technology Co., Ltd.
maxuetao@gmail.com

**Yifan Yanggong**
ZhongHao XinYing (Hangzhou) Technology Co., Ltd.
baihu@cltech.com

**Junbo Zhao** †
Zhejiang University
j.zhao@zju.edu.cn

May 17, 2023

**ABSTRACT**

Instruction tuning for large language models (LLMs) has gained attention from researchers due to its ability to unlock the potential of LLMs in following instructions. While instruction tuning offers advantages for facilitating the adaptation of large language models (LLMs) to downstream tasks as a fine-tuning approach, training models with tens of millions or even billions of parameters on large amounts of data results in unaffordable computational costs. To address this, we focus on reducing the data used in LLM instruction tuning to decrease training costs and improve data efficiency, dubbed as Low Training Data Instruction Tuning (LTD Instruction Tuning). Specifically, this paper conducts a preliminary exploration into reducing the data used in LLM training and identifies several observations regarding task specialization for LLM training, such as the optimization of performance for a specific task, the number of instruction types required for instruction tuning, and the amount of data required for task-specific models. The results suggest that task-specific models can be trained using less than **0.5%** of the original dataset, with a **2%** improvement in performance over those trained on full task-related data.

## 1 Introduction

With the tremendous momentum of large language models and their impressive performance [OpenAI, 2023a, Taylor et al., 2022, Touvron et al., 2023, Zhao et al., 2023], instruction tuning as one of their adaptation tuning approaches with fine-tuning on samples described via instructions [Longpre et al., 2023, Chung et al., 2022, Wei et al., 2022a], has attracted much attention from researchers [Ouyang et al., 2022]. Instruction tuning means fine-tuning the large language models on samples described via instructions [Longpre et al., 2023], as shown in Fig. 1. Yet lacking comprehensive analysis, this newly presented tuning method has shown its superior power in unlocking the endowed abilities of LLMs of the instruction following. Among related works, instruction tuning is mainly used to align LLMs with humans [Ouyang et al., 2022], generalize to certain unseen tasks [Sanh et al., 2021, Wei et al., 2022a], and specialize to a certain downstream task [Wang et al., 2022, Jang et al., 2023]. Compared to conventional fine-tuning, instruction

* Equal contribution and shared co-first authorship.
† Corresponding author.

tuning offers the advantages of requiring fewer data and being more human-friendly, and many researchers also regard instruction tuning as a new fine-tuning approach to adapt LLMs for corresponding downstream tasks [Wei et al., 2022a, Puri et al., 2022, Jang et al., 2023].

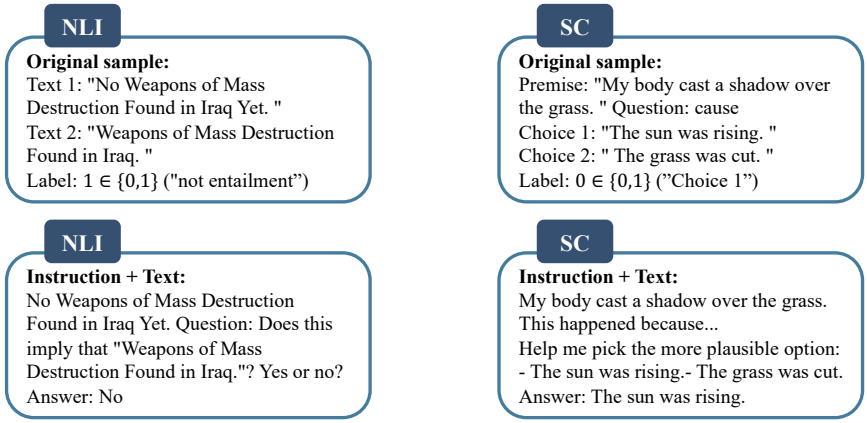


Figure 1: An illustration of the different between fine-tuning and instruction tuning, taking the task of natural language inference and causal reasoning as examples. LLMs predict labels for samples in fine-tuning while answering questions for the instruction set in instruction tuning. The Natural Language Inference(NLI) task involves determining the logical relationship between two pieces of text, typically referred to as the "premise" and the "hypothesis." The goal of NLI is to determine whether the hypothesis is true, false, or undetermined based on the information provided in the premise. Sentence Completion(SC) involves predicting the most likely word or sequence of words to complete a given sentence or phrase.

Although instruction-tuned LLMs are very powerful, training models with tens of millions or even billions of parameters often face the problem of training costs and, according to scaling laws [Kaplan et al., 2020], require large amounts of data. Most current works related to instruction tuning always expand the amount of data or the variance of instructions used for instruction tuning, for instance, the FLAN [Longpre et al., 2023] collection containing 15M examples covering 62 tasks, and the P3 dataset [Sanh et al., 2021] containing 250B tokens with 10 instruction types. However, with the increment of the LLMs' scale from 400M [Devlin et al., 2019] to 540B [Chowdhery et al., 2022] or even larger scale in the future, the scale of the training data used for instruction tuning will greatly affect the training costs. For example, OpenAI has listed the costs for instruction tuning the Davinci [OpenAI, 2023b] with 0.03 dollars per 1k token, which means training a model with P3 will cost 7.5M dollars.

However, we argue that there is a lack of instruction-tuning-related research on reducing the amount of data used for the training stage to decrease the training costs. While training costs are often related to hardware conditions and engineering operations, data efficiency can be improved with the help of algorithms, thus reducing the cost of training data. For example, self-instruct [Wang et al., 2022] declines the number of instances corresponding to each instruction and uses 80K examples to instruct-tune the Davinci with 338 dollars cost. We may call these methods reducing the training data scale during the instruction tuning as Low Training Data Instruction Tuning (LTD Instruction tuning) in the below.

Considering instruction tuning as a fine-tuning method for specialization tasks, we explore another line of LTD instruction tuning – reducing the diversity of tasks and instructions. While most instruction tuning-based works focus on the generalization ability of LLMs, few focus on specialization. What if the LLM only needs to be tuned for a certain task? How many instructions do we need? How many training examples are needed by the model? These questions are still under-explored.

In this paper, we conduct a preliminary exploration into reducing the data used in the LLM training phase, from the perspective of the data itself, to decrease the training costs and improve data efficiency. Specifically, we aim to identify the most valuable core samples from existing data to help the model acquire the knowledge necessary for downstream tasks, and to achieve comparable or even better performance with only a small amount of data. As a result, after selecting a specific task and instruction format, we successfully train a task-specific model using less than **0.5%** of the original dataset, with a comparable performance compared to the model trained on task-related data in P3. Our observations on the natural language inference (NLI) task are as follows, and they yield several key findings regarding task specialization for LLM training, which we hope can provide some insights for the community.

- If only to optimize performance for a specific task, an LLM model tuned solely on target task data is likely to outperform a modal tuned on data from different types of tasks.
- When specializing in a single task, it appears that only one instruction may be sufficient for instruction tuning. While increasing the number of instruction types can improve performance, the marginal effect becomes less significant, and there may even be cases where a single instruction outperforms ten types of instruction.
- In contrast to training a model for overall task performance, our results also suggest that 16k instances (1.9M tokens, 0.5% of the P3) may be sufficient to train an NLI task-specific model.

## 2 Related Work

**Large language models (LLMs).** Large language models (LLMs) generally refer to language models with tens or hundreds of billions of parameters and are trained with massive data, e.g., GPT-3 [Brown et al., 2020], GPT-4 [OpenAI, 2023a], Galactica [Taylor et al., 2022], LLaMA [Touvron et al., 2023], etc. Many studies have shown such models have under-explored emergent abilities compared to small models when the scale exceeds a certain level [Kaplan et al., 2020, Wei et al., 2022b]. Zhao et al. [2023] concludes that the current emergent abilities of large models include mainly in-context learning, which helps models possess the ability to adapt to the downstream tasks without gradient update, but only a few examples or several task demonstrations [Ye et al., 2023, Dong et al., 2023]. Some researchers also regard this operation as instruction [Ye et al., 2023]. Another emergent ability is instruction following [Ouyang et al., 2022]. By adding task descriptions to the data, the LLMs can understand the requirements of a task without additional samples on unseen downstream tasks [Zhou et al., 2022, Si et al., 2022, Wei et al., 2022a, Sumers et al., 2022], or tuning with these re-formatted data to endow task-specific capability to the model [Longpre et al., 2023, Wang et al., 2022, Sanh et al., 2021, Gupta et al., 2022, Chung et al., 2022, Ivison et al., 2022a, Jang et al., 2023]. Many works also find an important ability of step-by-step reasoning (also known as chain-of-thoughts) [Wei et al., 2022c, Wang et al., 2023], which helps LLMs to derive a final answer by splitting the task and using a form of intermediate steps of reasoning.

**Instruction tuning.** Although current instruction-following LLMs have demonstrated strong performance in deriving task-relevant answers relying on instruction [Si et al., 2022, Wei et al., 2022a, Sanh et al., 2021, Zhou et al., 2022], when facing task-specific issues, fine-tuning is still the preferred option for achieving better results [Lou et al., 2023, Zhao et al., 2023]. Unlike only using instructions without training to guide the model output, instruction tuning is an approach to fine-tune LLMs with data fused with instruction to achieve task-specific effects [Sanh et al., 2021, Longpre et al., 2023, Puri et al., 2022, Jang et al., 2023, Ivison et al., 2022a,b]. Most works focus on the generalization capability brought by instruction tuning, which helps the model to have cross-task generalization by fine-tuning on multi-task instruction data [Sanh et al., 2021, Longpre et al., 2023]. In addition, instruction tuning can also help the model improve performance on a specific task [Wang et al., 2022, Jang et al., 2023, Ivison et al., 2022a,b], and research has shown that on single-task fine-tuning, instruction tuning can accelerate model convergence.

**Low training data.** While the effectiveness of LLMs is superb, the training costs associated with huge parametric models have simultaneously limited the popularity and adoption of LLMs [Hoffmann et al., 2022, Zhao et al., 2023]. Many works try to explore the possible cost reduction in LLMs from the perspective of data. Jang et al. [2023] report an expert LLM fine-tuned on one single task can outperform a multi-task tuned LLM. Self-instruct [Wang et al., 2022] starts from the instruction generation, which means a model generates its own instructions or prompts and learns to follow them. DEFT [Ivison et al., 2022a] assumes the presence of unlabeled task-related data, and by retrieving data using K-nearest neighbors from the data pool that is highly similar to the task data, new fine-tuned models can achieve the same performance as full dataset-trained models do. HINT [Ivison et al., 2022b] incorporates instruction into model parameters to reduce the number of tokens corresponding to instruction in each input training to save the token cost.

## 3 Method

As mentioned in the introduction, our key ideas for limiting the data scale are reducing the variety of instructions and focusing on specialized tasks. And in this section, we'll introduce how to reduce the entire dataset scale based on both ideas separately.

### 3.1 Motivation

We start by introducing the motivation for our method before explaining its details. At present, the development of LLM has received tremendous attention, but the high training cost brought by the model with large parameters also limits the popularity and application of LLM. We hope to explore how to improve the efficiency of LLM from the perspective

of reducing its training cost. The main costs in the current use of LLM include training costs and data costs [Zhao et al., 2023]. The training costs include using public APIs or self-finetuning large models, which mainly face hardware or paralleling requirements, and the algorithm plays a light role in it, while for data costs, data-centric algorithms can be used. Therefore, we hope to explore how to reduce the data used in LLM training from the perspective of the data itself to reduce training costs and improve the efficiency of data usage. That is, to retrieve the most useful core samples from the existing data to help the model learn the knowledge required for downstream tasks, and to achieve good performance with only a small amount of samples.

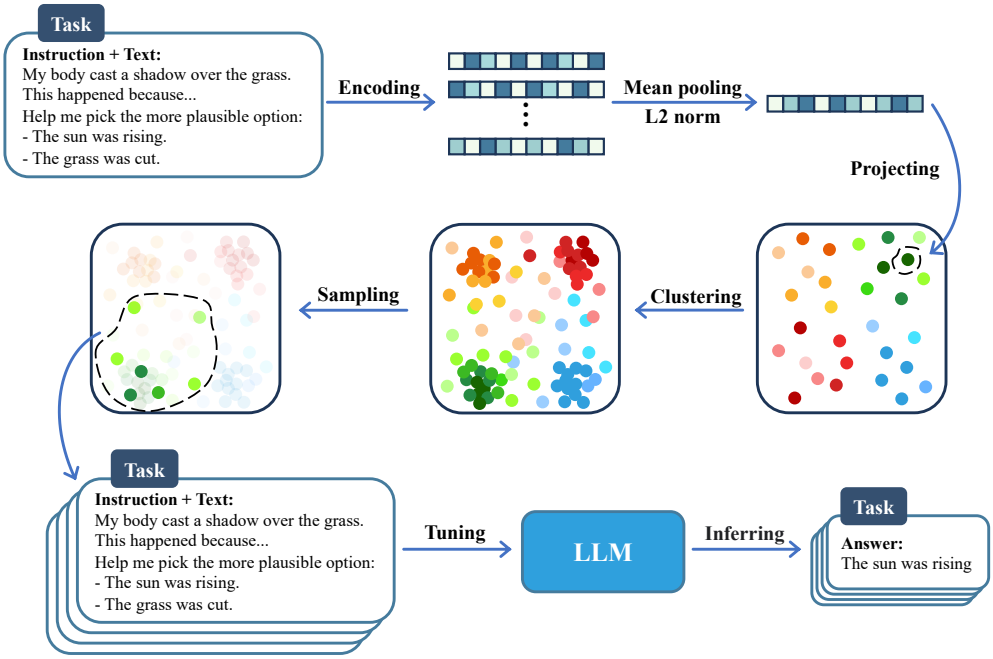


Figure 2: The pipeline of our proposed method. First, encode each sentence into embeddings and pre-process with mean pooling and L2 normalization. After this, in the latent space, we cluster all sample points into a few clusters. Then employ the sampling methods on these clustered samples to find the core samples of the original distribution. Finally, using these retrieved samples to instruction tuning the LLMs and make the evaluation. The three rectangles represent the latent space, and one color series in the latent space refers to one task. Points of the same color series but different shades correspond to data from the same task but from different datasets, e.g., the NLI task has five datasets, thus making it five different shades.

### 3.2 Coreset-based Selection

Due to the vague boundaries of NLP tasks [Zhao et al., 2023], samples from different tasks may have low discriminability, and it is often infeasible to manually select more suitable samples for NLP tasks. Therefore, we propose a coreset-based task-related data selection method to automatically retrieve core samples from the task dataset, to help train task-specific LLM with fewer samples. Specifically, this algorithm includes the following steps:

**Sentence embedding and pre-processing** Firstly, we re-formatted the data to the training input format used during the instruction tuning training phase, i.e., data with description instructions, adding answers at the end to format one complete training data. Then, we encode all samples using a pre-trained language model (e.g., Galactica [Taylor et al., 2022] or Bert [Devlin et al., 2019]). Specifically, we extract the *last_hidden_state* of each sample after feeding the model as the word embeddings or each sentence. Note that masked language models like Bert [Devlin et al., 2019] have *cls* token as the sentence embedding for one input sentence, but for generative models like GPT series [Brown et al., 2020, OpenAI, 2023a], they do not have this token. Following Reimers and Gurevych [2019], we performed mean pooling on the word embeddings of each sample and obtained a one-dimensional vector as the sentence embedding for this sample. To speed up the computation and facilitate vector similarity calculation, we normalize all sentence embeddings to length 1, i.e., L2 normalization is performed on the embedding dimension.

**Clustering** In the clustering step, we take into account that the fuzziness of NLP task boundaries may cause little variation among samples from different tasks. Thus we approach unsupervised clustering by focusing on data representations, rather than relying on label information to group data points together based on the same categories or tasks. Specifically, after obtaining the sentence embeddings from the first step, we perform unsupervised clustering using K-Means [Lloyd, 1982] in the embedding space to obtain the mapping of each sample and its corresponding cluster label. Then, based on the frequency of samples from one downstream task appearing in several clusters, we select the center point of the cluster with the highest frequency as the distribution center point of that downstream task. Next, for all the samples in the task, we calculate the cosine similarity with the distribution center point (the choice of distance function has little effect on the results, and we follow OpenAI [2023a] to choose cosine similarity), and find the closest sample from task data to this center point as the task center point. Note that the distribution center point is the center of this task data in embedding space, which may not exist in the task data, while the task center point is one exact sample from this task data with the biggest cosine similarity to the distribution center point.

**Coreset sampling** Intuitively, after obtaining the distribution center point corresponding to the downstream task, we can select the most similar sample as the representative task sample based on cosine similarity, as done in [Ivison et al., 2022a], which achieved good results. However, their retrieval method selects high-similarity samples from the data pool based on existing samples to improve task performance, which can be considered as a form of data augmentation through retrieval. This contradicts our goal of reducing the data required for training. We aim to find a small set that approximates the distribution of the full dataset using as few samples as possible. Therefore, the K-nearest neighbor method in DEFT [Ivison et al., 2022a] is unsuitable for this situation since samples with high similarity do not approximate the full set distribution [Sener and Savarese, 2018]. To achieve the sampling of core samples, we used one coreset algorithm KCentergreedy [Sener and Savarese, 2018], which aims to choose $k$ center points such that minimize the largest distance between a random data point and its nearest center, and has been proven efficient in obtaining a set of core samples of one distribution.

We use the task sample center point as the initial center, feed all the sentence embeddings of task samples obtained in the previous steps, and use the KCenterGreedy algorithm to collect a set of core samples from the task samples according to the given proportion. The subset of the original task dataset collected can achieve the same or even higher performance with fewer data.

### 3.3 To Be continue...

It should be noted that in addition to this method, we also explored two other ways to reduce the training data required by fine training. However, due to computing power and time limitation, they are not yet complete enough to be reported. Please wait for our future papers.

## 4 Experiments

### 4.1 Setup

**Dataset** Following the setup in P3[Sanh et al., 2021], we conduct experiments on a total of 11 datasets, which spanned across 4 NLP tasks, namely Natural Language Inference (NLI, 1.9M tokens), Sentence Complement (SC, 660.6K tokens), Word Sense Disambiguation (WSD, 25.5K tokens) and Coreference Resolution (CR, 185.1K tokens). On the contrary, the full task-related dataset of P3 contains 382.8M tokens.

To be specific, for the Natural Language Inference task, we employ RTE [Dagan et al., 2006], CB [Wang et al., 2020], and ANLI [Nie et al., 2020] datasets, while for the Sentence Complement task, we used COPA [Wang et al., 2020], HellaSwag [Zellers et al., 2019], and Story Cloze [Mostafazadeh et al., 2016]. For the Coreference Resolution task, we utilized Winogrande [Sakaguchi et al., 2019] and WSC [Wang et al., 2020] datasets, and for the Word Sense Disambiguation, we used WIC [Wang et al., 2020]. Moreover, to generate the instruction-style dataset, we randomly selected only one prompt from each dataset.

**Model** We utilize the Galactica-1.3b [Taylor et al., 2022] model to conduct experiments in our study. Galactica models are trained on a vast scientific corpus and are tailored to handle scientific tasks such as citation prediction, scientific question-answering, mathematical reasoning, summarization, document generation, molecular property prediction, and entity extraction. Following [Brown et al., 2020], similar to the pre-training phase, we treat all datasets as next token prediction tasks. In particular, we employ the AdamW optimizer with a learning rate of 1e-5.

**Evaluation** Prior research on instruction tuning failed to state the evaluation methods utilized explicitly. In this paper, we introduce our evaluation methodology, which can serve as a reference for other researchers working in this area.

When a tokenized sequence $x$ and a tokenized answer option $y$ (with a length of $l$) are provided as input to the model, a probability matrix $P_{l \times vocab\_size}$ is generated. Subsequently, for an answer option $y^i$, its corresponding probability $p^i$ can be obtained by multiplying the probabilities of each token in $y^i$ using the formula $\prod_{j=1}^{l^i} p_j^i$. The answer option with the highest probability is considered by the model as the optimal answer.

### 4.2 Results on Natural Language Inference Tasks

We describe in this section the NLI task results of using our method, as seen in Table. 1. According to the information presented in this table, when considering a specific task (NLI in this case), our method achieves a performance improvement of **2%** on average beyond the baseline (P3 in table) on the NLI task, using only **0.5%** of the available data from P3. In comparison to using all ten instructions from P3, we find that selecting only one instruction allows us to achieve comparable results to using the entire dataset from P3 with only 10% of the data.

Regarding task-specific models, the second and fourth rows have shown that the diversity of tasks might have a negative impact. Moreover, by exclusively utilizing data from the NLI task, we obtain results that are approximately 8% on average higher than those from P3.

Therefore, we speculate that for task-specific requirements, using only relevant data for the target task and a single instruction may be more effective than directly employing large-scale models. Notice that these observations may only be applicable to the NLI task, as other tasks remain largely unexplored due to computational limitations.

| **Model** | RTE | CB | ANLI R1 | ANLI R2 | ANLI R3 | Avg. |
|---|---|---|---|---|---|---|
| Vanilla Model (0%) | 54.51 | 41.07 | 33.40 | 33.40 | 33.58 | 39.19 |
| P3 (100%) | 76.17 | 75.00 | 44.00 | 35.70 | 39.42 | 54.06 |
| Fixed Instruction (10%) | 71.11 | 66.07 | 43.60 | 38.90 | 42.17 | 52.37 |
| NLI-related (5%) | 79.06 | 82.14 | 60.40 | 46.50 | 46.67 | 62.95 |
| NLI coreset (0.5%) | 74.73 | 73.21 | 49.60 | 41.90 | 43.75 | 56.64 |

Table 1: Test accuracy (%) with used models on NLI task. The first row suggests the performance of the vanilla model (Galactica-1.3b here) without instruction tuning. P3 stands for using a full task-related dataset from P3 (10 instructions with 11 datasets). Fixed instruction stands for using only one instruction type, resulting in a 10% of P3. Among all 11 datasets, NLI task-related data accounts for 50%, thus NLI-related refers to training with only 5% of P3 (NLI data). With this, NLI coreset indicates using only 10% of NLI task-related data, thus making a 0.5% of P3.

### 4.3 Ablation Study for Sampling

Regarding the strategy for model sampling, we have also explored several alternative sampling approaches. The table indicates that selecting the most similar or dissimilar data based on cosine similarity yields significantly poor results, even lower than the vanilla model. We speculate that this outcome may be attributed to an imbalance issue when sampling solely based on similarity within the task pool consisting of these five datasets. It is likely that a majority of the sampled examples are dominated by RTE or ANLI data.

| **Sampling Method** | RTE | CB | ANLI R1 | ANLI R2 | ANLI R3 | Avg. |
|---|---|---|---|---|---|---|
| Vanilla Model (0%) | 54.51 | 41.07 | 33.40 | 33.40 | 33.58 | 39.19 |
| coreset (0.5%) | 74.73 | 73.21 | 49.60 | 41.90 | 43.75 | 56.64 |
| topK (0.5%) | 47.29 | 10.71 | 33.40 | 33.20 | 33.92 | 31.70 |
| leastK (0.5%) | 50.18 | 8.93 | 33.80 | 33.60 | 34.08 | 32.12 |
| mixed (0.5%) | 47.29 | 8.93 | 33.30 | 33.30 | 33.50 | 31.26 |

Table 2: Ablation study on test accuracy (%) of NLI task using 10% data (0.5% of full task-related dataset from P3). The first row represents the vanilla model without any sampling method (0% of P3). Coreset refers to our method of using core samples of the NLI dataset. TopK uses samples that are close to the NLI distribution center point, while the leastK uses the least close samples. Mixed indicates mixing top and least close samples. The number of samples used is the same for all methods (10% of NLI task, ~16k samples).

## 5 Conclusion and Future Work

In this paper, we present experimental results from our preliminary exploration of Low Training Data Instruction Tuning, by tuning the Galactica-1.3b Model on the P3 dataset for the NLI task. Our study has revealed several findings:

1. task-specific models may benefit from fixed task types to achieve superior performance;
2. the diversity of instruction formats may have minimal impact on task-specific model performance;
3. even a small amount of data (1.9M tokens) can lead to promising results in instruction tuning for task-specific models.

It should be noted that our work has several limitations due to the constraints of computational resources, such as conducting experiments solely on Galactica-1.3b and utilizing only the NLI task data from the P3 dataset.

We hope our preliminary findings can provide insights to the community on Low Training Data Instruction Tuning, and yield a new perspective on instruction tuning for researchers. As for future work, we plan to validate these ideas on bigger models using a more comprehensive range of tasks and datasets.

## References

OpenAI. Gpt-4 technical report, 2023a.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science, 2022.

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. Llama: Open and efficient foundation language models, 2023.

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

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

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

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022a.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. *arXiv preprint arXiv:2302.03202*, 2023.

Ravsehaj Singh Puri, Swaroop Mishra, Mihir Parmar, and Chitta Baral. How many data samples is an additional instruction worth? *arXiv preprint arXiv:2203.09161*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

OpenAI. Openai api pricing, 2023b. `https://openai.com/pricing` [Accessed: (2023-04-29)].

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, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey 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. Language models are few-shot learners, 2020.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022b.

Seonghyeon Ye, Hyeonbin Hwang, Sohee Yang, Hyeongu Yun, Yireun Kim, and Minjoon Seo. In-context instruction learning. *arXiv preprint arXiv:2302.14691*, 2023.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey on in-context learning, 2023.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*, 2022.

Theodore Sumers, Robert Hawkins, Mark K Ho, Tom Griffiths, and Dylan Hadfield-Menell. How to talk so ai will learn: Instructions, descriptions, and autonomy. *Advances in Neural Information Processing Systems*, 35:34762–34775, 2022.

Prakhar Gupta, Cathy Jiao, Yi-Ting Yeh, Shikib Mehri, Maxine Eskenazi, and Jeffrey P Bigham. Improving zero and few-shot generalization in dialogue through instruction tuning. *arXiv preprint arXiv:2205.12673*, 2022.

Hamish Ivison, Noah A Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. Data-efficient finetuning using cross-task nearest neighbors. *arXiv preprint arXiv:2212.00196*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022c.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

Renze Lou, Kai Zhang, and Wenpeng Yin. Is prompt all you need? no. a comprehensive and broader view of instruction learning. *arXiv preprint arXiv:2303.10475*, 2023.

Hamish Ivison, Akshita Bhagia, Yizhong Wang, Hannaneh Hajishirzi, and Matthew Peters. Hint: Hypernetwork instruction tuning for efficient zero-shot generalisation. *arXiv preprint arXiv:2212.10315*, 2022b.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach, 2018.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pages 177–190. Springer, 2006.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories, 2016.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.