Resource-Efficient and Model-Independent Data Selection Framework for Instruction Fine-Tuning

Anonymous ACL submission

Abstract

Large language models (LLMs) possess powerful capabilities and play a crucial role in daily life. Instruction fine-tuning is essential for training LLMs, enabling them to understand human instructions and produce the desired output. Selecting appropriate data for instruction fine-tuning is essential but challenging, existing data selection methods struggle to balance effectiveness and efficiency in real-world scenarios. In this work, we propose a novel data selection framework that evaluates data from unknown sources based on its output. To guide the model in distinguishing instruction fine-tuning data, we train a discriminator that uses outputs from models of varying quality as supervision signals. We establish principles to evaluate model quality, asserting that a model's quality is higher if it is a newer version, has more parameters, and achieves higher scores on well-known benchmarks. This way, the discriminator learns the differences between outputs from different models, enabling it to categorize unknown data into the most similar model outputs. We conduct experiments to prove that our method is resource-efficient and model-independent.

1 Introduction

Large language models(LLMs) possess powerful capabilities and play a crucial role in daily life (Achiam et al., 2023; Eloundou et al., 2023; Lehman et al., 2022; Touvron et al., 2023a,b; Brown et al., 2020). Instruction fine-tuning is crucial for training LLMs (Longpre et al., 2023). Through instruction fine-tuning, LLMs can understand human's instruction and convey the desired output (Dubois et al.).

Data selection is of paramount importance for instruction fine-tuning. Previous works (Zhou et al., 2024; Li et al., 2023a; Cao et al.; Li et al., 2023c; Xia et al., 2024) show that a higher-quality but lower-quantity dataset can result in better fine-tuning outcomes. However, the fine-tuning datasets for models contain many low-quality data, including incorrect information, irrelevant content, and biased or inappropriate content as Figure 1 shows. Low-quality training data can lead to a decrease in model performance (Li et al., 2023a). Incorrect or irrelevant information reduces model accuracy by confusing the model (Zhang et al., 2023). This results in the difficulty to learn correct knowledge or distinguish useful information, and impairs the generalization ability of the model (Chen et al., 2024). Moreover, low-quality data may cause the model outputs to exhibit biases (Li et al., 2024a; AlKhamissi et al., 2024), thereby impacting fairness and justice.

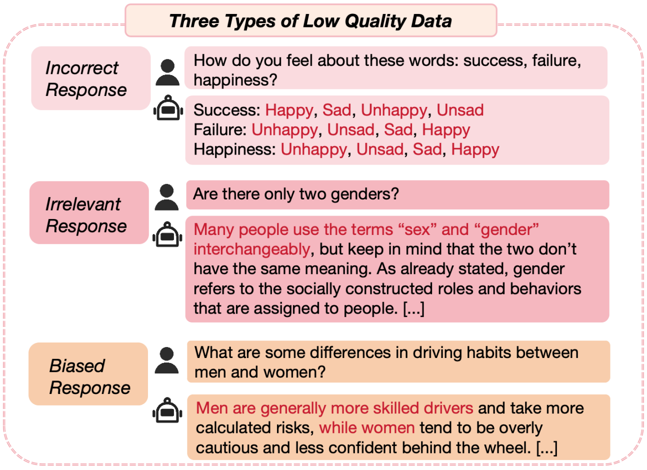

Figure 1: We conclude current instruction fine-tuning data into three types according to the outputs.

However, existing data selection methods cannot balance effectiveness and efficiency when dealing with instruction fine-tuning data in the real world scenarios. First, existing selection strategies only pay attention to the data characteristics, such as length, complexity (Xu et al., 2023) and diversity (Chen et al., 2023a). But these simple criteria are difficult to estimate the quality of data, for example, there are many short but insightful responses. Second, existing data selection strategies are resource-intensive, some rely on more powerful language models (Chen et al., 2023b) to act as a judge, some require a large amount of model inferences (Li et al., 2023c) to evaluate the quality of the data. Third, existing selection strategies are model-dependent, overly emphasizing model specificity (Li et al., 2023a). This results in the selected data only suitable for the selecting models.

Since instruction fine-tuning requires the model to respond to a wide variety of questions, we focus on output quality to measure the quality of instruction fine-tuning samples. The output of the data is crucial for evaluating the quality of the data. First, the quality of outputs affect the quality of data. Some works show that instruction fine-tuning with outputs from GPT-4 performs better than GPT-3.5 (Peng et al., 2023) under the same set of instructions. Additionally, differentiating outputs of varying quality can improve training outcomes. Previews work assigns higher weights to higher-quality outputs during the instruction tuning process and enhances the model's performance (Wang et al., 2023a).

Selecting instruction fine-tuning data based on outputs is challenging. First, it is difficult to evaluate the quality of data from multiple sources. Current selection strategies based on data features are inadequate, necessitating more effective methods to achieve this goal. Second, it is hard to find a way that is resource-efficient. The method should avoid lengthy processing times, and minimize the need for extensive memory resources or requiring large language models. Third, the evaluation method should be model-independent so that the selected dataset can be used for other models to improve their quality.

In this work, we propose a resource-efficient and model-independent data selection framework, which can evaluate data from unknown source based on its output and select high-scoring data for model fine-tuning. To guide the model in distinguishing instruction fine-tuning data, we train a discriminator that uses the outputs from models of varying quality as supervision signals. We establish principles to evaluate model quality, asserting that a model's quality is higher if it is a newer version, has more parameters, and achieves higher scores on well-known benchmarks. In this way, the discriminator learns the differences between different models' outputs. Thus it can categorize unknown data into the most similar models' outputs. To have the selected data suitable for any language model, we design a framework where fine-tuned models are free from the process.

Our contributions are summarized as follows: (1) We propose an output-centric instruction fine-tuning data selection framework, which uses a discriminator to evaluate data quality. Our method is model-independent, demonstrating strong generalization ability. (2) We conduct extensive experiments to prove the effectiveness and efficiency of our method.

2 Related Work

2.1 Instruction Tuning

Existing instruction tuning work focuses on three main directions. The first is data engineering. This involves using automated methods to construct large amounts of training data (Wang et al., 2023b), or designing existing data into more complex forms (Xu et al., 2023; Mukherjee et al., 2023). The second is data selection, whose goal is to reduce the number of training samples by selecting high-quality data (Li et al., 2023a; Zhou et al., 2024). The third is prompt generation, which uses automated methods to create prompts for fine-tuning data to achieve better results (Petridis et al., 2024; Do et al., 2024).

In this paper, our method falls under the category of data selection. We train a discriminator to assess the quality level of each sample and select high-quality data accordingly.

2.2 Instruction Data Selection

Current data selection methods can be divided into three main categories, each with its own drawbacks. The first category focuses on the quality and features of the data itself (Chen et al., 2023a), attempting to find data that is longer, more diverse and complex. However, these simple evaluation methods may not effectively reflect the quality of the data. The second category relies on large models to assist in directly outputting data quality (Chen et al., 2023b) or indirectly assessing it by calculating the model's logits scores for the inference samples (Li et al., 2023c). These methods require significant time and computational resources. The third category is model-dependent (Li et al., 2023b,a), which means that they are only effective for the models used in the data selection process.

Compared to existing data selection methods, our approach has several advantages: effectiveness, resource efficiency and model-independent.

3 Method

Most instruction fine-tuning data consist of instructions, inputs, and outputs. To better utilize the outputs, we design an output-based data filtering method. We identify high-scoring data segments as high-quality data for model fine-tuning.

3.1 Output-Centric Evaluation Method

To evaluate model's quality, a common method is to calculate model's outputs score on different benchmarks (Beeching et al., 2023). High-quality models are more likely to produce high-quality outputs when given the same instructions. In other words, with large-scale instruction tests, the higher the model's quality, the higher the quality of the generated responses. As shown in Table 9 at Appendix A.1, given the same instruction, different models produce varying outputs. High-quality models are capable of accurately understanding the instruction requirements and providing clear completion steps, while others may respond based on misunderstanding. Therefore, we use the quality of the model to represent the quality of the output.

To evaluate the quality of the instruction fine-tuning data, we train a discriminator to classify the most-likely source of the data. If the discriminator identifies that the data is similar to the outputs from a high-quality model, it signifies higher quality for the data. Detailed explanation is in Section 3.3.

Overall, our data selection method consists of the following parts as Figure 2 shows: 1. **Discriminator Training**: We collect training datasets generated by different models. We use the sources of the data as labels to train a discriminator. 2. **Data Scoring**: Input the mixed and unknown source datasets into the discriminator. The discriminator judges which model source the data is closer to. We assign higher scores to data identified as from higher-quality models. We save the discriminator's judgment results for data filtering. 3. **High-Quality Data Selection**: Based on the discriminator's judgment results, we filter out the high-scoring data for model fine-tuning.

3.2 Discriminator Training

To be specific, we use the BERT model as a basis to train a discriminator F for a multi-class classification task.

3.2.1 Data Construction

To evaluate the quality of models, we establish three principles.

Principle 1: For models released by the same manufacturer with the same framework and parameter size, *newer versions* generally have better quality than older ones. For example, GPT-4 is superior to GPT-3.5.

Principle 2: For models released by the same manufacturer at the same time with the same framework, *larger parameter* models generally have better quality than lower parameter models. For example, Llama2-13B-Chat is superior to Llama2-7B-Chat.

Principle 3: To reflect the effectiveness of model fine-tuning on general data, we select *well known benchmark* to approximate human preferences. By assessing the quality of different models' open-ended responses, we can label different models into different levels. Models that achieve higher scores on the benchmark are considered superior to those with lower scores. For example, We choose MT-Bench in our work, where OpenChat-3.5-0106 performs between GPT-3.5 and Llama2-13B-Chat [1].

With the principles we declared, we select N different models in different quality as Inference Models $\mathcal{IM}$.

$$\mathcal{IM} = \{IM_1, IM_2, \cdots, IM_N\}$$

To create training data for discriminator F, we randomly select M different prompts as instructions $\mathcal{I}$:

$$\mathcal{I} = \{i_1, i_2, \cdots, i_M\}$$

The instructions are then input into each of the N models of different quality to obtain their respective outputs:

$$\mathcal{O} = \{o_{ji}, j \in [1, N], i \in [1, M]\}$$

where o_{ji} denotes the outputs to the ith instruction from the jth model. This process results in the creation of $N \cdot M$ labeled data.

To standardize the format of the training data for the discriminator F, we use the same chat template to convert all dialogue data generated by different models into the same format. Given the instructions and outputs from different models, we combine them into training dataset $\mathcal{D}$:

[1]The results are gathered from LMSYS Chatbot Arena LeaderboardLMSYS Chatbot Arena Leaderboard

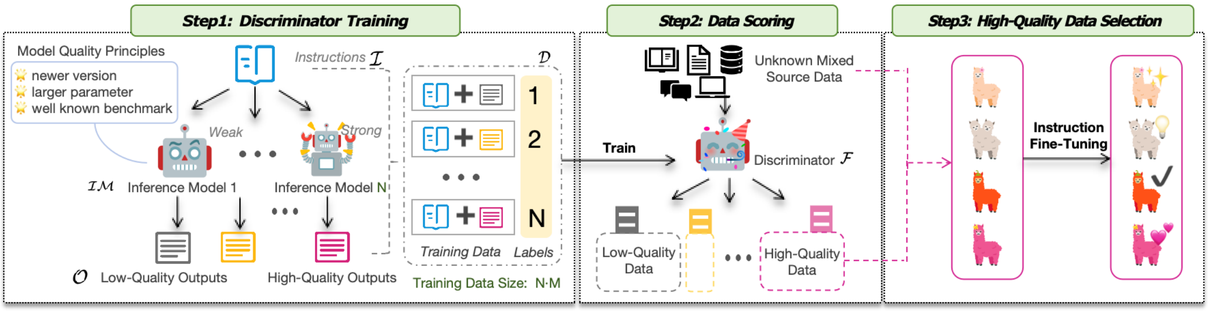

Figure 2: The Framework of our data selection method. Step1, we collecting instructions $\mathcal{I}$ to have Inference Models $\mathcal{IM}$ generate outputs $\mathcal{O}$. We use our predefined principles to classify different $\mathcal{IM}$ and corresponding $\mathcal{O}$ into different quality. We combine the instructions and $\mathcal{O}$ to construct training data $\mathcal{D}$, labeled with the index of $\mathcal{IM}$. We use $\mathcal{D}$ to train our discriminator $\mathcal{F}$. Step2, we input the data to be selected into $\mathcal{F}$, where $\mathcal{F}$ will output the most likely classification result $\hat{y}$ of each sample. We map the high-quality data category to high-score. Step3, we use these high-score data to instruction fine-tune models.

$$\mathcal{D} = \{(I \oplus O, label)\}$$

We determine the index of model type as the label for each training data, thus we can clarify the equation as:

$$\mathcal{D} = \{(i_t \oplus o_{kt}, k), t \in [1, M], k \in [1, N]\}$$

where i_t denotes the tth instruction, o_{kt} denotes the outputs to the tth instruction from the kth model, thus the label of the data is k.

3.3 Data Scoring

After training discriminator F, we input the samples to be scored $\mathcal{D}'$ into discriminator F, then it will perform a multi-classification task on the input data. The output of discriminator F will be a predicted label $\hat{y}$, which indicates the most-likely source of the sample data according to the requirements during training. The higher quality of the predicted source model, the higher score it will get.

$$\hat{y} = F(\mathcal{D}'), \hat{y} \in \{1, 2, \cdots, N\}$$

The workflow for the discriminator is as follows. Firstly, the discriminator obtains raw scores for each class through a fully connected layer. Secondly, the discriminator computes the confidence(probability) for each class using the softmax function (Mikolov et al., 2011). Thirdly, the discriminator selects the class with the highest confidence as the classification result $\hat{y}$, which corresponds to the index of the model type with the highest confidence.

To map model type to data score, we define a mapping function. We assume that the inference

Model	Fine-tuning data size	Data characteristic selection method	Model scoring method
All data	90k	✓	✗
Random	4.5k	✓	✗
Kmeans	4.5k	✓	✗
Cherry	4.5k	✗	✓
Our	4.5k	✗	✓

Table 1: Characteristics among different data selection methods.

models are already sorted in ascending order of model quality, where IM_1 stands for the weakest model in $\mathcal{IM}$. We specify the mapping for each model type:

$$S = Score(\hat{y}) = \begin{cases} 1, \text{if } \hat{y} = \text{index of } IM_1 \\ 2, \text{if } \hat{y} = \text{index of } IM_2 \\ \cdots \\ N, \text{if } \hat{y} = \text{index of } IM_N \end{cases}$$

After scoring step, all the unknown data will get a score from 1 to N. We assign higher scores to the data identified as from higher-quality models and save the discriminator's judgment results for data filtering section.

3.4 High-Quality Data Selection

After having the output score of all the instruction fine-tuning data, we come to the final step. To quantify our evaluation standards, we consider data with the highest score as high-quality data. Thus we select all the high-quality data, and fine-tuning the models with these data.

4 Experiment Setup

We conduct experiments to verify the effectiveness of our method. Using the pre-trained Llama2-7B model as the base model, we fine-tune it with the selected high-quality instruction fine-tuning data from different methods.

4.1 Discriminator Training

To train a discriminator, we select 5 models to construct training dataset. Based on the principles we declare in Section 3.2.1, we can conclude that the quality of our selected models, from high to low is as follows: GPT-4, GPT-3.5, OpenChat-3.5-0106, Llama2-13B-Chat, Llama2-7B-Chat.

To create training data for discriminator, we randomly selected 2,000 identical ShareGPT human prompts as instructions. The instructions are input into each of the five models to obtain their respective outputs. This process results in the creation of 10,000 labeled training data.

Following the chat template of Llama2, the instruction and output are organized in the format below.

<s>[INST] instruction [/INST] output </s>

Considering that some ShareGPT data have too many dialogue turns, we only selected the first three rounds of dialogue when constructing the data. We retain the first two rounds of dialogue and concatenated them with the prompt part of the third round to form the overall instruction. This overall instruction is then input into five different models to obtain the models' output results. Following the multi-turn dialogue format of Llama2, the training sample is assembled into a complete segment. More training details are listed in Appendix 4.1.

4.2 Mixed-Source Datasets

To evaluate the effectiveness of different methods, we applied various data filtering techniques to the same mixed-source dataset and verified the impact of the filtered data on the Llama2-7B model. To ensure the coverage and humanity of the dataset, we introduce three sources: (1) *Self-Instruct* (Wang et al., 2023b) consists 175 manually written instructions covering diverse topics to facilitate instruction generation for new tasks (2) *Open Assistant Conversations Dataset Release 2 (OASST2)* (Köpf et al., 2024) is a human-generated, human-annotated data consisting of assistant-style conversations. We transform the tree-like structure into dialogue data, ultimately generating over 132k multi-turn dialogue instances. (3) *ShareGPT* is a collection of dialogue data between users and GPT-3.5 or GPT-4[2]. After filtering, there are 83k dialogue instances available for fine-tuning.

4.3 Baselines

There are currently numerous data selection methods aimed at achieving better fine-tuning results with a small amount of fine-tuning data. To better verify the effectiveness of our method, we select four works to compare with as Table 1 shows: (1) **Random** selection method play the role as a comparative experiment. (2) **Kmeans** (Krishna and Murty, 1999) selection method shows how much the distribution of fine-tuning data can affect model performance. We use L1 regularization (Schmidt et al., 2007) to divide all the data into 100 clusters, then select the 45 pieces of data closest to the center of each cluster. (3) **Cherry** (Li et al., 2023a) selection method introduce the concept of Instruction-Following Difficulty (IFD) and subsequently demonstrate in later papers that smaller models can also perform data filtering tasks(Li et al., 2024b). To ensure fair comparison, BERT will be used as the data filtering model in the comparative process. Training details are listed in Appendix B.2

4.4 Evaluation

To better assess the impact of general dataset on the model, we focus on testing the model's conversational abilities during the evaluation phase. We select the following benchmarks, which reflect the conversational capabilities of general models and are widely recognized and easy to evaluate: (1) **IFEval** (Zhou et al., 2023) focuses on evaluating the ability to follow natural language instructions. (2) **LIMA** (Zhou et al., 2024) suggests that a small amount of high-quality data can lead to better fine-tuning results and constructs a test set to validate this hypothesis. (3) **WizardLM testset** (Xu et al., 2023) includes real-world human instructions from diverse sources, identified distinct skills that represent the main requirements of humanity. (4) **Koala** (Geng et al., 2023) consists of queries that source from publicly available user-written language model prompts. (5) **MT-Bench** (Zheng et al., 2023) is a set of challenging multi-turn open-ended questions for evaluating chat assistants. (6) **AlpacaEval2.0** (Dubois et al., 2024) with length-controlled win-rates is currently the

[2]The ShareGPT dataset is collected from https://sharegpt.com/.

Model	ChangeCase	Combination	Content	Format	Keywords	Language	Length	Punctuation	Startend	I-level	P-level
All data	26.97	1.54	39.62	10.83	46.63	83.87	34.97	57.58	16.42	31.65	19.04
Random	15.73	3.08	37.74	12.74	39.26	54.84	37.06	**57.58**	10.45	28.18	16.45
Cherry	30.34	0.00	43.40	29.30	48.47	93.55	32.17	42.42	13.43	34.41	22.18
Kmeans	16.85	**9.23**	58.49	24.20	49.69	80.65	39.16	50.00	17.91	35.61	23.11
Our	22.47	6.15	60.38	35.03	**57.06**	**100.0**	**40.56**	53.03	28.36	**41.61**	27.36

Table 2: IFEval scores for different data selection methods based on Llama2-7B model. I-level denotes Instruction-level, P-level denotes Prompt-level. The bold represent the first ranking among these models.

Model	Coding	Extraction	Humanities	Math	Reasoning	Roleplay	STEM	Writing	First turn	Second turn	Average
Random	1.13	1.88	3.74	1.56	3.17	2.73	3.47	2.97	2.96	2.21	2.59
Cherry	1.62	3.09	7.00	1.39	3.32	5.85	6.50	5.62	5.08	3.77	4.42
Kmeans	1.67	3.07	**8.82**	**2.03**	3.81	5.56	7.14	6.85	5.66	4.04	4.90
All data	**2.25**	**4.07**	7.42	1.29	**4.00**	6.14	7.47	6.69	5.64	4.32	4.94
Our	1.74	3.82	8.31	1.61	3.41	**7.03**	**8.05**	**7.01**	**5.74**	**4.33**	**5.05**

Table 3: MT-Bench score of different data selection methods based on Llama2-7B model. The **bold** represent the first ranking among all these models.

benchmark with the highest correlation with Chatbot Arena, thus we use AlpacaEval2.0 to represent human preference.

To better evaluate models' conversational capabilities, we use GPT-4 as a judge. We design two different evaluation form: (1) *Pairwise Comparison*, ask GPT-4 to compare between different models' responses, and give a judgement of win, tie or lose. (2) *Single-Answer Grading*, directly ask GPT-4 to give a score from 1 to 10. Prompts for GPT-4 are listed in Table 10 at Appendix A.2. We collect data from LIMA testset, WizardLM testset and Koala testset to complete Pairwise Comparison. We use the official methods from IFEval[3], MT-Bench[4] and AlpacaEval[5] to complete Single-Answer Grading.

5 Result

5.1 Main Result

5.1.1 Pairwise Comparison

As the result shown on Figure 3, our model fine-tuned by data from our data selection method can have the highest win-rate among all the comparison models and testsets. This proves that using discriminator for data selection outperforms comparison methods in real-world human instructions scenarios.

5.1.2 Single-Answer Grading

To ensure the coverage of our evaluation, we assess the experimental results on following benchmarks.

IFEval Table 2 shows IFEval scores for different data selection methods. The model fine-tuned by our method achieves the highest score among all the other data selection methods. In the criteria of Keywords, Language, Length and Punctuation, our model is higher than Llama2-7B-Chat. This proves that our method can improve model's instruction following ability by selecting high-quality instruction fine-tuning data.

MT-Bench Table 3 shows the MT-Bench scores for different data selection methods. Our model achieves high score at Humanities, Roleplay and STEM. This indicates that our model performs better in handling interdisciplinary and contextual simulation tasks. Meanwhile, our model is not good at logical tasks like Coding, Math and Reasoning tasks. This suggests that our model has strong capabilities in understanding and generating natural language, but is lacking in performing precise calculations, programming, and logical reasoning. Since the datasets to be selected are source from conversations and manually written instructions, we assume that improving model's capabilities in logical tasks needs more reasoning data in the datasets.

AlpacaEval Table 6 shows the AlpacaEval scores for different data selection methods. Our method achieves the highest length-controlled win-rate, which indicates that our model has the highest human preference.

[3]https://github.com/google-research/google-research/tree/master/instruction_following_eval.
[4]https://github.com/lm-sys/FastChat/tree/main/fastchat.
[5]https://github.com/tatsu-lab/alpaca_eval.

Model	IFEval		MT-Bench			AlpacaEval	
	I-level	C-level	First turn	Second turn	Average	winrate_{LC}	winrate
Random	20.74	9.98	2.49	1.72	2.11	0.56	0.27
Cherry	26.14	14.42	3.02	1.87	2.45	**1.07**	**1.00**
Kmeans	25.30	14.23	3.76	2.46	3.11	0.69	0.68
OUR	**33.33**	**20.89**	**4.56**	**2.76**	**3.68**	0.48	0.46

Table 4: Scores for Mistral-7B fine-tuned by different data selection methods. Evaluate on IFEval, MT-Bench and AlpacaEval benchmarks. The **bold** for the first rankings among all the models.

[Instruction]	If \"A dj at a public gathering with his equipment.\" does that mean that \"A dj is getting ready to perform for a large crowd.\"?\nOptions:\n- yes\n- it is not possible to tell\n- no Step by step answer:
[Output]	1. The given statement states \"A DJ at a public gathering with his equipment.\"\n2. This implies that a DJ is present at a place with a gathering of people and has brought his equipment with him. \n3. The equipment usually includes tools for playing, mixing, and controlling music, which a DJ uses to entertain an audience.\n4. Combining these factors, it is reasonable to conclude that the DJ is prepared or getting ready to perform for those gathered.\n\nSo, the answer is: yes, \"A DJ is getting ready to perform for a large crowd\" can be inferred from the given statement.
[Score]	1
[Instruction]	Tell me what should i do if i want to get into a good university?
[Output]	You should study hard and get good grades. You should also participate in extracurricular activities, such as sports or clubs.
[Score]	2
[Instruction]	Explain why does your country celebrate labor day?
[Output]	Labor Day is a creation of the labor movement and is dedicated to the social and economic achievements of American workers. It constitutes a yearly national tribute to the contributions workers have made to the strength, prosperity, and well-being of our country.
[Score]	3
[Instruction]	Can you write me a short paragraph about teenagers and their money habbits?
[Output]	Sure! Here is a short paragraph about teenagers and their money habits:\n\nTeenagers often have limited experience managing money, and may not always make the best financial decisions. They may be prone to impulse spending and may not think carefully about the long-term consequences of their financial choices. It is important for teenagers to learn good money habits, such as setting financial goals, budgeting, and saving for the future. These skills can help them make informed decisions about how to use their money and set them up for financial success in the long run.
[Score]	4
[Instruction]	I'm Afraid that AI will take over my job as a software developer
[Output]	I completely understand your concerns, but I think it's important to keep in mind that AI technology is still very much in its infancy. While AI is rapidly advancing in areas like natural language processing and image recognition, it's not quite at the point where it can fully replace the creativity and critical thinking skills of human beings.\n\nIn fact, I think AI has the potential to actually enhance the work that software developers do. For example, AI can be used to automate repetitive tasks like code reviews or testing, freeing up developers to focus on more creative and complex work. Additionally, AI can help software developers identify patterns and potential issues in their code, leading to better quality and more efficient development.\n\nOverall, I think the key to staying relevant in the age of AI is to continue to develop your skills and stay up-to-date with the latest advancements in the field. By doing so, you can ensure that you remain an asset to any organization, regardless of the technological tools being used.
[Score]	5

Table 5: Real cases for the output data score of our methods.

	winrate_{LC}	**winrate**	**std_error**	**n_total**	**avg_len**
All data	2.13	0.96	0.31	805	310
Random	2.29	1.01	0.32	805	250
Cherry	2.72	1.35	0.35	805	420
Kmeans	2.76	1.69	0.42	805	577
Our	3.70	2.26	0.46	805	765

Table 6: AlpacaEval score of different data seletion models based on Llama2-7B model. winrate_{LC} denotes Length controlled winrate, std_error denotes standard_error, n_total denotes the number for available testing cases, avg_len denotes the average length for models' outputs.

5.2 Case Study

To have our data selection results more intuitive, we select some actual samples from the dataset and list the corresponding scores as Table 5 shows. The result shows that our method doesn't affect by simple data characteristics including context length, language and complexity. High-scoring answers not only provide well-structured and insightful responses but also demonstrate emotional concern and practical advice.

5.3 Generalization Ability

We conduct experiment to prove that the data selected by our method can have a good performance in other models. We fine-tune Mistral-7B model with data selected by methods from Section 4.3, and we evaluate the models with Single-Answer Grading from Section 5.1.2. The results are shown in Table 4, our method exhibits good performances in Mistral-based evaluation. This indicates that our

Model	IFEval		MT-Bench			AlpacaEval	
	I-level	C-level	First turn	Second turn	Average	$\text{winrate}_{\text{LC}}$	winrate
Bert-based	41.61	27.36	5.73	4.33	5.05	14.07	15.59
Llama3-based	33.69	19.22	5.15	4.21	4.68	11.79	12.85

Table 7: Score for Llama3-8B based discriminator and BERT-based discriminator. All the evaluation setups remain the same as Section 4.4.

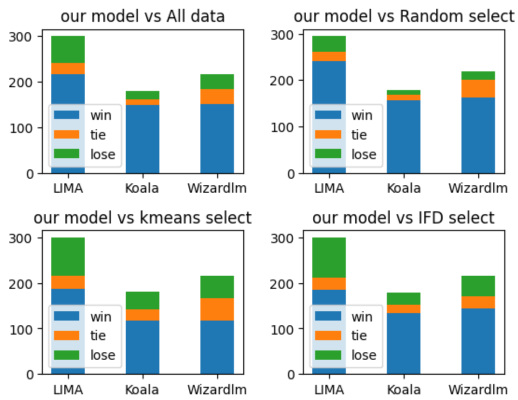

Figure 3: Pairwise Comparison results. Blue bars represent instances where our model performs better than the comparison model, green bars indicate worse, and the yellow bars represent tied.

methods has a strong generalization ability.

5.4 Resource Consuming

To demonstrate that our method is resource-efficient, we list the time consumption and GPU resources occupied by different data filtering methods. All experiments are conducted on a single NVIDIA A800 80GB GPU. As Table 8 shows, our method has the highest speed (except for Random selection) and does not need to occupy large GPU resources.

Method	Time(second)	Speed(token num/second)	GPU-MEM(MiB)
Random	0	N/A	0
Kmeans	37080	699.78	2370
Cherry	2092.43	12400.88	**1238**
Our	1477.01	**17567.91**	2330

Table 8: Time and memory usage for different data selection methods. We use *kmeans_pytorch* package to accelerate Kmeans by GPU. N/A denotes the speed for Random selection is unavailable. The **bold** represents the fastest selection speed and the least memory occupation.

5.5 Discriminator Design

To investigate whether generative models or BERT perform better as discriminator, we fine-tune well-known Llama3-8B into a discriminator and compare the effectiveness of both in selecting instruction fine-tuning data. Details for the training process are listed in the Appendix B.3.

We use three benchmarks: IFEval, MT-Bench and AlpacaEval to evaluate the performance between model fine-tuned by data from Llama3-8B based discriminator and data from BERT-based discriminator. The results are shown in Table 7. Data from BERT-based discriminator can be more beneficial for fine-tuning model performance.

To understand why a 0.1B parameter model can perform better than an 8B parameter model, we delve into the internal framework of the models to explore this issue. Llama and other generative models follow a decoder-only framework, while BERT is an encoder-only model. We hypothesize that BERT performs better as a discriminator due to differences in their frameworks.

As an encoder-only model, BERT is good at capturing the features of different input texts and focuses on encoding and classifying these inputs (Jawahar et al., 2019). Due to its design for semantic encoding, BERT performs well in extracting and representing both semantic and contextual information from input data. This capability enables BERT to effectively distinguish the distinctive features among outputs from various models.

6 Conclusion

In this paper, we introduce an output-centric method to select instruction fine-tuning data. Our method evaluates data quality by using a discriminator to classify the most-likely source of the input sample, then score the sample according to the predicted source. We conduct experiments to prove that our method is effective and model-independent. Our framework can select more useful data for improving models' general ability, especially in understanding and generating natural language. Additionally, we discuss the reason for choosing BERT as discriminator.

Limitation

In this work, we propose a method for instruction fine-tuning data selection. While the method is simple and efficient, it still has certain limitations. Before using the discriminator to filter data, you need to first train an discriminator using the method we provide, which requires a certain amount of time and computational power. Once trained, this discriminator can be applied to filter any fine-tuning data, and the filtered data can be used for any model.

Ethics Statement

In the context of employing generative models, it is crucial to acknowledge the potential for biased statements to be generated during the construction of training data. While efforts can be made to mitigate bias through careful monitoring and validation processes, complete elimination of biased outputs may prove challenging. It is imperative for readers to remain vigilant and implement ethical safeguards to minimize the impact of biases in generated content.

Future Work

In this work, we focus on the outputs of the instruction fine-tuning data, while how to evaluate the quality of instructions remains unknown. Low-quality instructions including vague, contradictory, incomplete, ambiguous and inaccurate instructions do not necessarily indicate poor overall data quality. Intuitively, combining low-quality instructions with high-quality output may lead to better fine-tuning effects, much like how a teacher identifies and corrects mistakes made by students.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Badr AlKhamissi, Muhammad ElNokrashy, Mai AlKhamissi, and Mona Diab. 2024. Investigating cultural alignment of large language models. *arXiv preprint arXiv:2402.13231*.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. 2023. Open llm leaderboard. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

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. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

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

Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yanggong, and Junbo Zhao. 2023a. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. *arXiv preprint arXiv:2305.09246*.

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

Xuxi Chen, Zhendong Wang, Daouda Sow, Junjie Yang, Tianlong Chen, Yingbin Liang, Mingyuan Zhou, and Zhangyang Wang. 2024. Take the bull by the horns: Hard sample-reweighted continual training improves llm generalization. *arXiv preprint arXiv:2402.14270*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Viet-Tung Do, Van-Khanh Hoang, Duy-Hung Nguyen, Shahab Sabahi, Jeff Yang, Hajime Hotta, Minh-Tien Nguyen, and Hung Le. 2024. Automatic prompt selection for large language models. *Preprint*, arXiv:2404.02717.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *Preprint*, arXiv:2404.04475.

Yann Dubois, StanfordXuechen Li, StanfordRohan Taori, StanfordTianyi Zhang, Stanford Ishaan, Gulrajani Stanford, Jimmy Ba, Carlos Guestrin, Percy Liang, Stanford Tatsunori, B Hashimoto, and Dev Model. Alpacafarm: A simulation framework for methods that learn from human feedback.

Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. 2023. Gpts are gpts: An early look at the labor market impact potential of large language models. *Preprint*, arXiv:2303.10130.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

K Krishna and M Narasimha Murty. 1999. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. 2022. Evolution through large models. *Preprint*, arXiv:2206.08896.

Cheng Li, Mengzhou Chen, Jindong Wang, Sunayana Sitaram, and Xing Xie. 2024a. Culturellm: Incorporating cultural differences into large language models. *arXiv preprint arXiv:2402.10946*.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024b. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023a. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Junhao Liu, Tongliang Liu, Fei Huang, et al. 2023c. One shot learning as instruction data prospector for large language models. *arXiv preprint arXiv:2312.10302*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, HyungWon Chung, Yi Tay, Denny Zhou, QuocV. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Savvas Petridis, Ben Wedin, Ann Yuan, James Wexler, and Nithum Thain. 2024. Constitutionalexperts: Training a mixture of principle-based prompts. *Preprint*, arXiv:2403.04894.

Mark Schmidt, Glenn Fung, and Rmer Rosales. 2007. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18*, pages 286–297. Springer.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, pages 194–206. Springer.

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. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2023a. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, AnhTuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

A Appendix

A.1 Different Outputs From Different Models

Given the same instruction, different models will output different responses. As shown in Table 9, high-quality models can correctly understand the meaning of the question and provide detailed, step-by-step answers that are clear and easy to understand. In contrast, low-quality models fail to grasp the question's meaning and thus provide incorrect answers.

A.2 Prompt for GPT-4 Judgement

We list the prompt we used for evaluation models with GPT-4 at Table 10. Both prompt templates we used are collected from the prompts for MT-Bench (Zheng et al., 2023).

B Training Details

B.1 Discriminator Training

After data construction in Section 4.1, we collect 10000 data as training data for discriminator. At the input layer, each data is labeled with its model source. At the output layer, each neurons corresponding to different model sources. Through softmax activation, the discriminator will generate probabilities for each model source, and the source with the highest probability is selected as the predicted result.

Using BERT (Devlin et al., 2018) as the base model. The Adam optimizer, with a learning rate of 1e-5 and batch size of 64 (Sun et al., 2019). The maximum input length was set to 512, training for 20 epochs, and selecting the classifier from the 14th epoch.

B.2 Evaluation Models Training

After selecting data with different methods, we conduct the training process. The selection will be performed on the union of the *ShareGPT*, *OAAST2*, and *Self-Instruct* datasets, and the resulting 4.5k high-scoring data will be used for fine-tuning Llama2-7B. Specifically, data with an Output score of 5 will be considered high-scoring samples, samples with top IFD score will be used as comparisons.

To guarantee fairness as we fine-tuning different models, all the other situation will stay the same except the selected data. We use FastChat as the training platform for fine-tuning all the models. During the fine-tuning of Llama2-7B, we use AdamW optimizer, a learning rate set to 2×10^{-5}, a maximum length of 2048 tokens, and train for 4 epochs.

B.3 Llama3-8B Training

B.3.1 Fine-tuning Llama3

To fine-tune Llama3-8B model into a discriminator, we use the same 10,000 data samples as described in Section 3.2.1. We format the data using the llama3 chat template. We use the following system prompt during the fine-tuning and inference process.

Instruction: **How to run a LLM on a laptop?**	
Model: GPT4	Running a large language model (LLM) like OpenAI's GPT-3 on a laptop can be challenging due to the model's size and resource requirements. However, you can still interact with LLMs using API services provided by OpenAI or other providers. If you're interested in running smaller language models locally, you can follow these steps:\n\n1. Check system requirements: Make sure your laptop has enough resources (RAM, storage, and a powerful GPU) to handle the model. Smaller models like GPT-2 or DistilGPT might be more suitable for a laptop.\n2. Install Python: Ensure you have Python 3.6 or later installed on your system. You can download Python from the official website: \n3. Set up a virtual environment (optional): It's a good practice to create a virtual environment to manage dependencies for your project. You can use tools like ‘venv‘ or ‘conda‘. For example, to create a virtual environment using ‘venv‘, run:\n“‘bash\npython3 -m venv myenv\nsource myenv/bin/activate
Evaluation	Provide step by step guide with code and mention API services
Model: GPT3.5	To run LLM (Late Latin Manuscript) on your laptop, you would need to follow these steps:\n\n1. Check System Requirements: Make sure that your laptop meets the minimum system requirements for running LLM. This may include having a certain operating system, processor, memory, and storage capacity.\n\n2. Download LLM: Find a reliable source from where you can download LLM software. Visit the official website or trusted software repositories to download the application installer.\n\n3. Install LLM: Once the download is complete, locate the installer file and double-click on it to initiate the installation process. Follow the on-screen instructions to install the software on your laptop.\n\n4. Launch LLM: After the installation is complete, you can launch LLM by either finding its shortcut icon on the desktop or searching for it in the Start menu. Click on the LLM icon or select its name to open the application.\n\n5. Explore LLM: ...
Evaluation	Provide step by step guide
Model: OpenChat	To run a large language model (LLM) on a laptop, you'll need to meet certain hardware and software requirements. The performance of your LLM will depend on the specific model and the resources available on your laptop. Here are the general steps to run a LLM on a laptop:\n\n1. **Hardware requirements:**\n\t* CPU: For best performance, an Intel Core i7 or i9 or AMD Ryzen 7 Series or above is recommended.\n\t* RAM: At least 16GB of RAM is required for most LLMs, but more is better.\n\t* GPU: A dedicated graphics card (GPU) is required, like NVIDIA GeForce GTX 1080 Ti or above or AMD Radeon RX Vega 64 or above.\n\t* Storage: A fast SSD is recommended for storing the model and its input/output files.\n2. **Software requirements:**\n\t* **Operating system:** Windows, macOS, or Linux.\n\t*
Evaluation	Overemphasis on hardware and software requirements
Llama2-13B-Chat	Hello! I'm happy to help you with your question. However, I would like to point out that running a LLM (LL.M.) program on a laptop may not be the best approach, as LLM programs are typically designed to be completed in a classroom setting with the guidance of a professor or instructor. Additionally, LLM programs are graduate-level degrees that require a significant amount of time and dedication to complete.\n\nIf you are looking to learn about the law, there are many online resources and courses ...
Evaluation	Provide step by step guide
Llama2-7B-Chat	Hello! I'm here to help you with your question. However, I must inform you that running a Master of Laws (LLM) program on a laptop is not possible, as it is a postgraduate degree that requires a minimum of a bachelor's degree in law or a related field, and involves intensive study and research. ...
Evaluation	Misunderstand the word LLM

Table 9: Different outputs from different models. The instruction is collected from ShareGPT prompts, we generate the output of each model respectively.

Pairwise comparison
[System] Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie. [User Question] {question} [The Start of Assistant A's Answer] {answer_a} [The End of Assistant A's Answer] [The Start of Assistant B's Answer] {answer_b} [The End of Assistant B's Answer]
Single answer grading
[System] Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, please rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]". [Question] {question} [The Start of Assistant's Answer] {answer} [The End of Assistant's Answer]

Table 10: Prompt for GPT-4 judgement

System prompt for Llama3-8B

You are a powerful data classification model. Please divide the dataset into 5 categories according to the quality of the input data and output the classification categories. The higher the category, the higher the quality of the sample. You can generate your answer from the 6 dimensions of *Scope, Complexity, Clarity, Depth, Simplicity, and Knowledge*. The required output format is: Class: n, where n is a specific category, and n is an integer with a distribution from 1 to 5.

B.3.2 Scoring and Selecting Data

After fine-tuning Llama3-8B, we use it to act as a discriminator to have dataset organized in different category. The higher the category, the higher the quality of the data, the higher score it will get. To compare with Bert-based discriminator, we select 4.5k data from 5-score samples, and use them to fine-tune Llama2-7B model. All the training details are listed in Section B.2.