Automated Data Curation for Robust Language Model Fine-Tuning

Anonymous ACL submission

Abstract

Large Language Models have become the de facto approach to sequence-to-sequence text generation tasks, but for specialized tasks/domains, a pretrained LLM lacks specific capabilities to produce accurate or well-formatted responses. Supervised fine-tuning specializes a LLM by training it on dataset of example prompts with target responses, but real-world data tends to be noisy. While many fine-tuning algorithms exist, here we consider a *data-centric AI* perspective on LLM fine-tuning, studying how to *systematically* curate the training dataset to improve the LLM produced via *any* fine-tuning algorithm.

We introduce an automated data curation pipeline CLEAR (**C**onfidence-based **L**LM **E**valuation **A**nd **R**ectification) for instruction tuning datasets, that can be used with any LLM and fine-tuning procedure. CLEAR estimates which training data is low-quality and either filters or corrects it. Automatically identifying which data to filter or correct is done via LLM-derived confidence estimates, to ensure only confident modifications to the dataset. Unlike existing data curation techniques, CLEAR is a comprehensive framework that can improve a dataset (and trained model outputs) without additional fine-tuning computations. We don't assume access to a stronger LLM than the model being fine-tuned (e.g. relying on GPT-4 when fine-tuning GPT-3.5), to see whether CLEAR can meaningfully improve the capabilities of any LLM. Experiments reveal that CLEAR consistently improves the performance of fine-tuned models across many datasets and models (like GPT-3.5 and Llama2).

1 Introduction

Large Language Models (LLMs) pretrained on internet-scale text corpora have shown remarkable capabilities in generating helpful human-like text (Brown et al., 2020; Touvron et al., 2023). However, the efficacy of LLMs in specialized domains or tasks hinges on the process of *instruction tuning* (i.e. supervised fine-tuning, or alignment), where pretrained models are further trained using datasets that well-represent the domain (Wei et al., 2022). Here we consider sequence-to-sequence training datasets of (prompt, target response) pairs. After training, we feed the LLM new prompts from the same domain and want it to produce responses that resemble expected targets.

Since billion parameter LLMs indiscriminately absorb patterns/information across a dataset, the quality of the instruction tuning data is paramount to effective fine-tuning (Zhou et al., 2023a; Xu et al., 2023; Kong et al., 2023). Unfortunately, most real-world instruction tuning datasets are noisy, containing examples that are low-quality in various ways: the target response may be inaccurate, poorly written, the prompt may be nonsensical/incomplete/vague, or the two may be unrelated due to data processing mistakes. Such flawed training data leads to fine-tuned models whose outputs are incorrect, irrelevant, biased, poorly formatted, or flawed in other ways. Finding and fixing low-quality data manually is challenging in large datasets.

While most machine learning research iterates over modeling strategies (architectures, loss functions, training algorithms, ...) for a fixed dataset to produce better results, the emerging science of *data-centric AI* asks how we can systematically iterate on the dataset while holding the modeling strategy fixed to produce better results (Mazumder et al., 2022). Success in real-world AI projects typically requires both approaches. Since many existing fine-tuning algorithms have been proposed (Zhang et al., 2023), we follow the spirit of data-centric AI and propose CLEAR, a comprehensive and automated data curation pipeline to enhance the effectiveness of instruction tuning datasets **for any LLM and fine-tuning algorithm**.

Our CLEAR pipeline involves two stages: **Auto-**

Filter and **Auto-Correct** which together offer a holistic solution to improving data quality for fine-tuning. The Auto-Filter stage removes data that is confidently low-quality from the dataset *without any LLM fine-tuning*. It is already able to significantly improve the dataset, such that we can produce better fine-tuned LLMs without any extra LLM fine-tuning computation. For settings where one is able to fine-tune the LLM more than once, the Auto-Correct stage uses the current fine-tuned LLM to revise certain examples that can be confidently improved. Fine-tuning the LLM again on this corrected dataset yields improved performance.

Algorithmic modifications to a dataset are generally harmful unless done with extreme care. Filtering too much data limits the number of examples to learn from, and editing data can introduce various biases or amplify flaws in existing model outputs. Thus, all data modifications in CLEAR are conservatively applied based on careful measures of *confidence*. Specifically, we rely on BSDetector (Chen and Mueller, 2023), a method that can be used with any LLM to obtain trustworthy confidence scores for its own outputs as well as estimating the confidence that given outputs (e.g. target responses) are good. CLEAR only filters data that is confidently identified as low-quality, and only revises data where the LLM-corrected response is confidently identified as better than the current dataset response. Our experiments reveal this careful treatment of confidence to be vital for developing a universal data filtering + correction solution that remains effective across diverse instruction-tuning datasets without any manual modifications.

2 Related Work

2.1 Data Curation for ML

Data curation has been key in real-world deployment of classical supervised learning, with a broad spectrum of methods developed to address dataset mislabeling, outliers, and other data issues (Mazumder et al., 2022). Algorithmic strategies such as noise estimation and removal (Northcutt et al., 2021; Zhou et al., 2023b; Wang et al., 2022), active learning for data prioritization (Settles, 2009; Chen et al., 2021), and crowd-sourced labeling (Snow et al., 2008) have demonstrated how to produce better models by producing better data. These strategies were designed for classical machine learning tasks like classification/regression, where datasets are less complex than in instruction tuning.

2.2 Instruction Fine-tuning

Significant research has been conducted into instruction tuning to specialize/improve LLMs (Kumar et al., 2016; Raffel et al., 2020; Efrat and Levy, 2020; Li and Liang, 2021; Chen et al., 2022a; Wei et al., 2022; Wang et al., 2023a). FLAN (Wei et al., 2022) is a popular approach that employs a 137 billion parameter pre-trained language model, which is fine-tuned using instructions on more than 60 NLP datasets verbalized in natural language instruction templates. Wang et al. (2023a) showed how various instruction-tuning datasets can induce specific skills in a model, though no single dataset (or their combination) provides optimal performance across all assessments. Contrary to previous efforts aimed at creating a general Foundation model capable of generalizing across a wide range of unseen tasks, our aim in this paper is to train the best possible LLM for a specific narrow task.

2.3 Data Curation for Instruction Fine-tuning

The quality of training data in text generation has such significance that previous instruction tuning datasets were often curated by hand (Khashabi et al., 2020; Ye et al., 2021; Wei et al., 2022; Wang et al., 2023b; Chen et al., 2022b; Honovich et al., 2023). Wang et al. (2023b) introduced automated techniques by using GPT-3 (Brown et al., 2020) to produce 52,000 unique instructions not directly linked to specific tasks. This innovation opened new avenues for creating instruction datasets by extracting knowledge from teacher models.

Following Meta's open-sourcing of the LLaMa prerained LLM (Touvron et al., 2023), many researchers began curating instruction tuning datasets to train useful variants of this LLM. Alpaca (Taori et al., 2023) introduces a self-instruct method to autonomously create instruction (prompt) examples, thereby reducing reliance on manual input. Vicuna (Chiang et al., 2023) capitalizes on the wide variety of data types and structures accessible via ShareGPT. WizardLM (Xu et al., 2023) augments a dataset by refining and diversifying instructions to evolutionarily increase their complexity/variability. UltraChat (Ding et al., 2023) introduces different well-defined scopes, systematically producing numerous instructions within each scope to improve task-specific performance. LIMA (Xu et al., 2023) selects a thousand high-quality data samples strategically, showing notable improve-

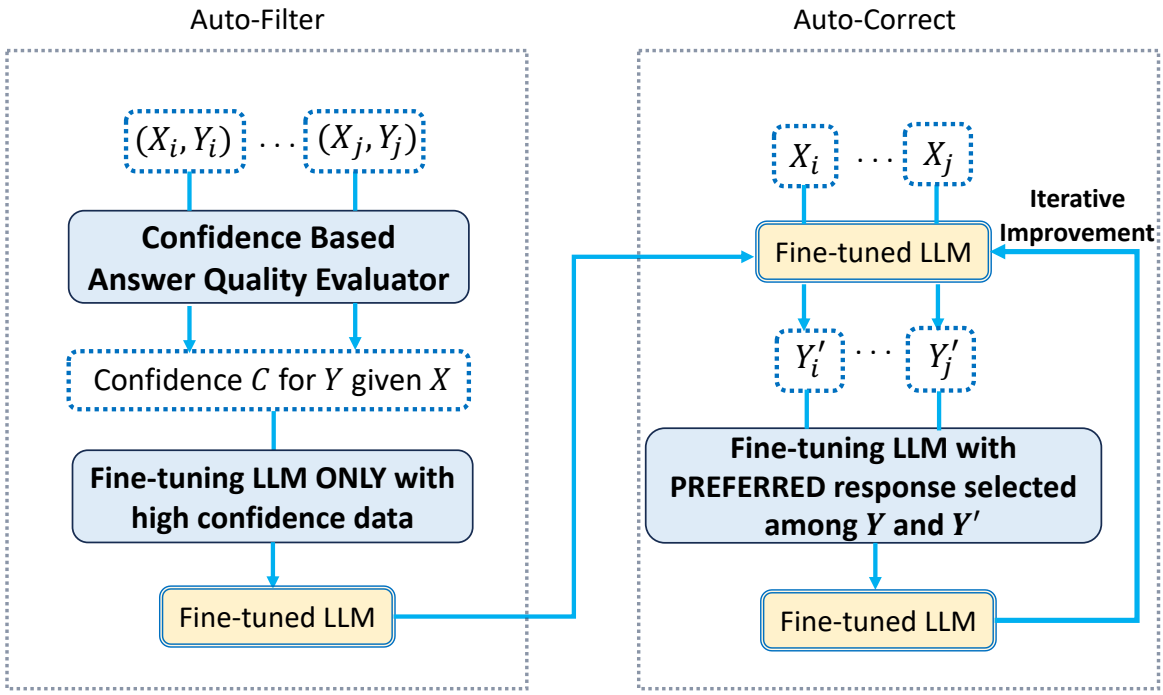

Figure 1: An overview of the CLEAR data curation procedure to automatically filter and correct bad data in any instruction-tuning dataset composed of instructions/prompts X_i and corresponding target responses Y_i.

ments in LLM performance. Li et al. (2023a) proposed an instruction-following metric to identify good examples in datasets.

Much existing LLM fine-tuning research has focused on distilling teacher models such as ChatGPT that are more powerful than the LLM being fine-tuned (Taori et al., 2023; Chiang et al., 2023). Many existing LLM-based data curation techniques also utilize more powerful LLMs for the data curation process than the LLM being fine-tuned. In contrast, we aim to produce the best LLMs for specific tasks, in which even the most advanced LLMs like GPT-4 struggle to perform. Thus all data curation throughout this paper is performed *using the same LLM as is being fine-tuned*, to truly assess how effectively this data curation is able to boost LLM performance beyond the frontier.

3 Automated Data Curation with CLEAR

An instruction tuning dataset $I = \{(x_i, y_i)_{i=1}^n\}$ comes with instructions/prompts x and corresponding target responses y obtained from a specific domain. The goal is to fine-tune the LLM to improve its comprehension and execution of instructions, such that it can produce responses similar to the expected targets for new instructions encountered during model deployment. In practice, large instruction tuning datasets are noisy, containing issues like: poorly written responses, incorrect responses, irrelevant/unhelpful responses, vague/incomplete instructions, data formatting problems, etc. These datasets are often sourced from messy chat logs or written by teams of humans that make mistakes rushing to produce data at scale.

As sequence-to-sequence mappings are extremely high-dimensional, a model's learning can be easily degraded by flawed training data lurking in some regions of this high-dimensional space. To develop an approach that can be used with any LLM model and any fine-tuning procedure, we consider simple dataset modifications rather than model-centric approaches that modify the training algorithm to be more robust. Our dataset modifications will benefit the next decade's LLMs, whereas training modifications tend to be model-specific.

Our proposed data curation pipeline involves two main steps: **Auto-Filter** and **Auto-Correct**, which aim to detect problematic (prompt, response) pairs in the data and rectify them if possible. Auto-Filter employs a confidence-based response quality evaluator (Chen and Mueller, 2023), to estimate our confidence that each pair in the dataset is good. Subsequently, the LLM is fine-tuned only on the high confidence data. This simple data filtering step already boosts LLM fine-tuning for noisy datasets,

and **requires no extra fine-tuning compute costs**.

Data filtering discards information, some of which may be useful. We propose to use the resulting fine-tuned LLM to correct certain bad responses identified in the original dataset, for which the fine-tuned LLM is able to produce a high-confidence alternative answer. This is determined by comparing the response generated by the fine-tuned LLM with the original response in the dataset. Rather than discarding such an example from the dataset in the previous filtering stage, we preserve the prompt and replace the target response with the fine-tuned LLM response in cases where the latter is confidently preferrable. After auto-correcting the dataset in this manner, the LLM can be fine-tuned again to produce an even better version of the model (without any change in the fine-tuning algorithm). This cycle of LLM fine-tuning and data refinement can be iterated in a virtuous cycle (see Figure 1).

3.1 Auto-Filter

To estimate the quality of responses in the original dataset, CLEAR diverges from the conventional method of asking capable LLMs like ChatGPT to directly rate the input-output pair according to various criteria (e.g. *helpfulness* as shown in Table 5). We instead employ LLM-derived confidence-estimates, specifically the BSDetector estimate introduced by Chen and Mueller (2023). This estimates the confidence that a response is good in terms of two factors: *observed consistency* and *self-reflection certainty*.

BSDetector uses our same LLM to generate multiple candidate responses to a given prompt (via diversity-increasing techniques like temperature sampling and chain-of-thought), and then evaluates the semantic alignment between these candidate responses and the target response in the dataset (via natural language inference). Beyond this observed consistency, BSDetector additionally integrates direct LLM self-evaluations of the target response (directly prompting the LLM to report its confidence that the response is good). The resulting confidence estimates account for both aleatoric and epistemic uncertainty, without requiring any modification/training of the LLM (no access to the LLM parameters is even required, enabling this approach to be used with arbitrary LLM APIs). Subsequent experiments reveal that this confidence-based approach to detect low-quality data is more precise than conventional LLM scoring of response quality (see Figure 2).

Given an instruction fine-tuning dataset of input-output pairs $\{(x_i, y_i)_{i=1}^n\}$, we use BSDetector with the base pretrained LLM (before it is fine-tuned) to estimate a confidence score c_i for each pair (x_i, y_i). We then filter out data pairs with low confidence scores below a predefined threshold γ:

$$F = \{(x_i, y_i) | c_i > \gamma\}.$$

Subsequently, we fine-tune the LLM on the remaining training data F.

3.2 Auto-Correct

Thus far, we considered filtering data estimated to be low-quality, but what if some of this data can be automatically rectified? A direct approach would be to substitute low-quality responses with LLM generated responses. For specialized domains, a pretrained general-purpose LLM like GPT-4 may be unable to generate better responses for us to consider. But the LLM we fine-tuned after the Auto-Filter stage is specialized to our domain and should be able to generate some reasonable responses. If the auto-filtering was done well, then this fine-tuned LLM will exhibit less flaws being trained on less flawed data.

In the Auto-Correct stage, we proceed to generate responses $\{(x_i, y_i')_{i=1}^n\}$ through queries to this fine-tuned model for each prompt x_i in our dataset. What remains is to decide when the candidate response y_i' generated by our current fine-tuned LLM is confidently better than the original dataset response y_i. For this, we directly ask our base Foundation LLM (pre fine-tuning) via the LLM-as-judge prompt in Table 1. As BSDetector is compatible any LLM, we can obtain confidence estimates for these LLM-as-judge preference predictions. For examples where the confidence (as estimated by BSDetector) that y_i' is better than y_i falls above a threshold η: we replace their target response with the LLM generated response and retain this pair in our curated dataset (rather than filtering it). This auto-corrected dataset is then used for further LLM fine-tuning, to yield a further improved model.

4 Experiments

Datasets. We evaluate the effectiveness of our data curation process across noisy versions of three supervised fine-tuning (text generation) datasets (see Figures 3,4,5). **SQuAD-N** (Rajpurkar et al., 2016): prompts are articles and target responses

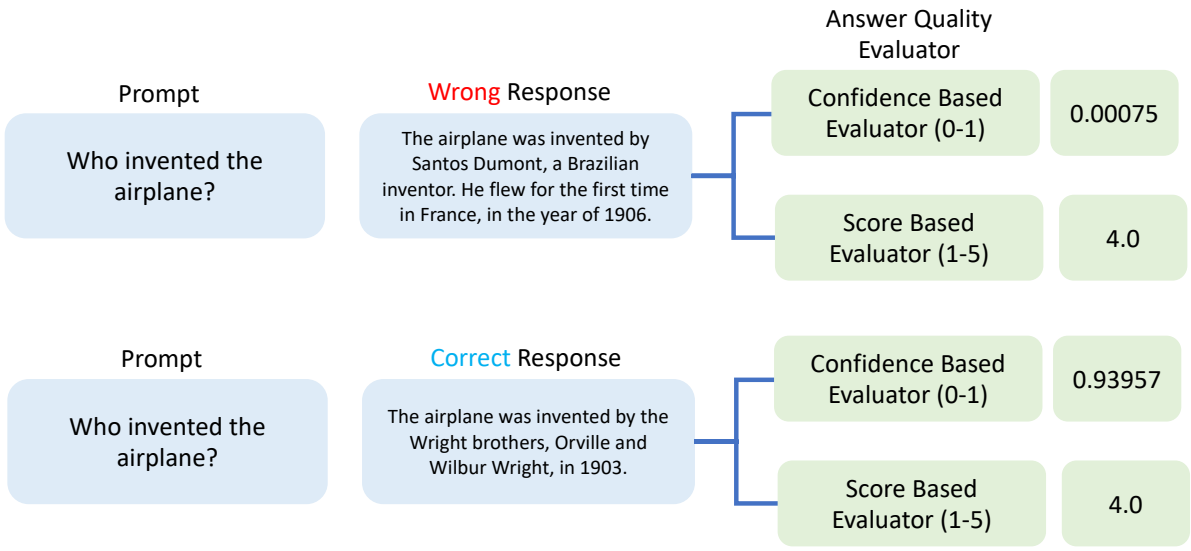

Figure 2: Comparing confidence vs. score based answer quality evaluators. The confidence-based (BSDetector) evaluator outputs a confidence value between 0 to 1. The direct LLM-scoring evaluator queries GPT-3.5-Turbo using a prompt (shown in Table 5) that requests a score between 1 to 5 to rate response quality. Higher values from either evaluator suggest higher-quality answers. For the incorrect response in the original dataset from the top figure: the confidence-based evaluator estimates low quality, while the score-based evaluator assigns a score of 4.0. For the correct answer to this prompt (bottom figure): the confidence-based evaluator estimates high quality, while the score-based evaluator still assigns a score of 4.0. Direct LLM score-based evaluation less reliably distinguishes between right vs. wrong responses.

Please review two answers carefully and select the one that you believe is superior. Consider factors such as accuracy, completeness, relevance to the question.
 	
Question: `[. . . ]`
 	
You are provided with two responses to the same question:
 	
`[The Start of Answer A]: [. . . ] [The End of Answer A]`
 	
`[The Start of Answer B]: [. . . ] [The End of Answer B]`
 	
Please provide a brief reasoning you used to derive it. After providing your explanation, output your final verdict by strictly following this format: "`[[A]]`" if Answer A is better, "`[[B]]`" if Answer B is better, and "`[[C]]`" for a tie.

Table 1: Prompt (Zheng et al., 2023) used to determine the preferable choice among y and y'.

are answers to questions created by crowdworkers based on a collection of Wikipedia articles, with each answer being a specific text fragment or span from the related article. **Emails-N** [1]: prompts are emails and target responses include categorizing the email into one of seven predefined themes by examining the email's subject and body content and also vary based on the email's length (whether the email content is short, medium, or long affects how the response is written). **DROP-N** (Dua et al., 2019): prompts are articles and target responses are answers to reading comprehension questions that require discrete reasoning over paragraphs (correctly answering requires resolving references in a question, perhaps to multiple places in the article, and performing basic operations over the references like addition, counting, or sorting).

To study how our approach handles noisy data, we perturbed 20% of each training dataset (not the corresponding test set). For the Emails dataset, the perturbation was to randomly swap target responses across different examples. To perturb a subset of the SQuAD and DROP datasets, where target responses are contained within a context passage in the provided instruction, we chose a random sentence from the context as the target response.

[1] https://huggingface.co/datasets/neelblabla/enron_labeled_emails_with_subjects-llama2-7b_finetuning

	Training Data or Prompting	Model	SQuAD-N Valid JSON (%)	SQuAD-N Accuracy (%)	Email-N Valid JSON (%)	Email-N Accuracy (%)	DROP-N Valid JSON (%)	DROP-N Accuracy (%)
Pretrained Model (No Fine-Tuning)	Zero-Shot	GPT-3.5	99.85	66.65	93.5	23.25	99.50	33.40
		GPT-4.0	99.90	75.93	100.0	48.25	100	39.80
		Llama-2	94.90	51.85	2.0	3.50	84.20	16.80
	One-Shot	GPT-3.5	99.20	69.50	99.0	38.75	99.60	40.80
		GPT-4.0	100.0	79.40	98.0	48.0	100.0	43.0
		Llama-2	24.65	9.70	17.25	19.50	32.0	4.90
	Three-Shot	GPT-3.5	87.60	61.20	95.75	47.0	98.50	41.80
		GPT-4.0	99.94	<u>80.08</u>	100.0	<u>49.75</u>	99.0	<u>46.10</u>
		Llama-2	13.10	2.55	1.75	5.75	20.60	4.60
Fine-Tuning	Original Data	Llama-2	92.45	49.86	99.30	50.67	99.30	44.70
	Auto-Filter Data	Llama-2	96.90	59.86	100.00	49.67	100.0	47.40
	Auto-Correct Data	Llama-2	96.90	71.44	99.67	52.33	100.0	50.50
	Original Data	GPT-3.5	97.90	64.50	100.0	43.0	100.0	56.80
	Auto-Filter Data	GPT-3.5	99.20	81.51	100.0	46.67	100.0	71.70
	Auto-Correct Data	GPT-3.5	100.0	**81.90**	100.0	**56.33**	100.0	**73.0**

Table 2: Test set performance achieved by various Large Language Models when employed in non fine-tuning baselines or when fine-tuned. Both the model's ability to generate correct results (accuracy) and properly-formatted results (valid JSON %) are reported. We underline the best non fine-tuning results, and indicate the best fine-tuning results in bold. Between each fine-tuning result, the training algorithm/code remains identical, only the underlying data is curated differently.

Evaluation metrics. For each dataset, our LLM fine-tuning performance evaluation focuses on two metrics (computed over a fixed held out test set): how often the model's response format adheres to a valid JSON structure and how often the model's responses are correct. For each model produced via a fine-tuning method, we report the proportion of model responses that are in valid JSON format, and the accuracy of model responses (which is computed via the proportion of exact matches to target reference responses, since we expect a well-supervised model to able to match the types of target responses it was fine-tuned on).

Baseline Methods. Our study also evaluates the following non fine-tuning methods: **Zero-shot** on GPT-3.5-turbo/GPT-4.0/Llama-2-7b-chat is directly querying these pretrained Foundation models. **Few-shot** on GPT-3.5-turbo/GPT-4.0/Llama-2-7b-chat is directly querying these pretrained Foundation models using in-context learning (with the indicated number of examples from the dataset inserted into each prompt as few-shot context). For the fine-tuning methods, we employ full model fine-tuning on Llama-2-7b-chat and OpenAI's GPT-3.5 Turbo fine-tuning API. **Fine-tuning on the noisy data** refers to fine-tuning the model on the original datasets without any data curation. **Auto-Filter** refers to fine-tuning the model on a curated versions of the dataset, where data with low confidence levels have been eliminated as described in Sec. 3.1. This procedure sets the median confidence value across the dataset as the threshold γ, filtering out any data below this threshold. **Auto-Correct** refers to fine-tuning the model on curated versions of the dataset, where certain data has corrected responses generated as described in Sec. 3.2 (we set $\eta = 0.8$). The fine-tuning routine stays the same when evaluating different data curation strategies – we only alter the training dataset, not the model/ algorithm.

Other Details. We study the effectiveness of data curation strategies across two different fine-tuning methods. On the Llama-2-7b-chat model, we conduct full model fine-tuning, in which all parameters of the neural network are updated via the

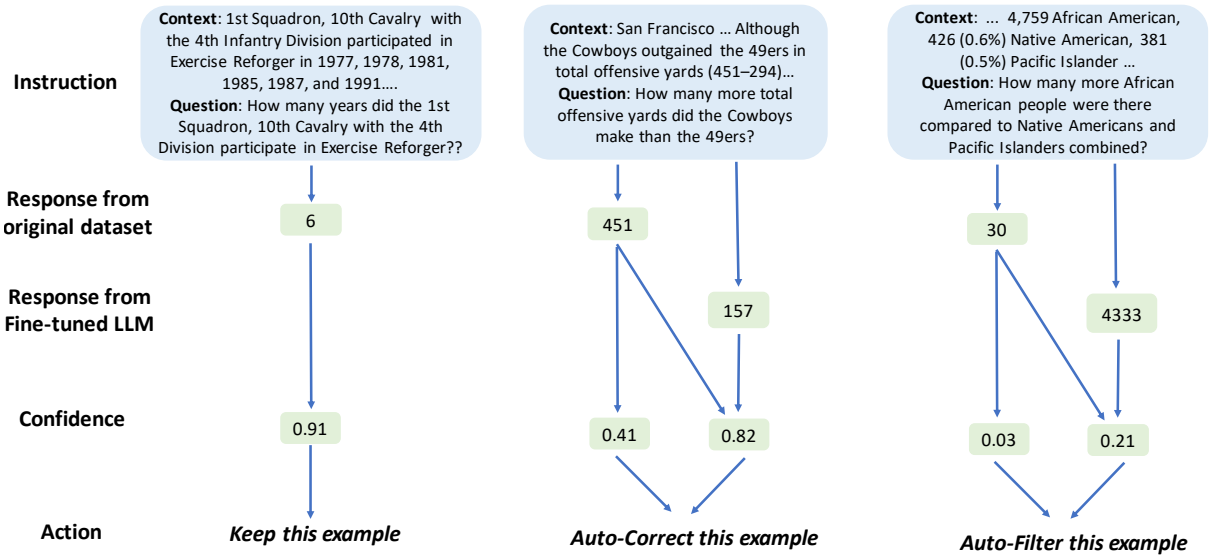

Figure 3: Three examples from the **DROP-N** dataset. The first example (left) is retained in the dataset because the original response has high BSDetector-estimated confidence (0.91). The second example (middle) has an original response that is estimated to be low confidence (0.41), and the candidate alternative response generated from our fine-tuned LLM is better than the original response with confidence 0.82. Since this exceeds our confidence threshold $\eta = 0.8$, we replace the target response for this second example with the LLM-generated candidate response in our curated dataset. The third example (right) has an original response that is estimated to be low confidence (0.03), but we also estimate low confidence (0.21) that the candidate response from our fine-tuned LLM is better. This third example is thus entirely removed from our curated dataset.

Adam optimizer. We set the batch size at 128, and train for 3 epochs, using a learning rate of 1×10^{-5} with an accompanying cosine learning rate schedule. For the GPT-3.5 Turbo model, we use OpenAI's fine-tuning API. The exact training algorithm/hyperparameters used remain undisclosed to us, but this API has been observed to be highly effective for LLM fine-tuning. When evaluating outputs from our models at test time, we perform all text generation with temperature 0, and limit the maximum number of output tokens to 512.

5 Results

Table 2 presents the results of our main experiments. Amongst the non fine-tuning approaches, GPT-4 stands out as the superior LLM, demonstrating the strongest performance across three datasets. For the pretrained GPT-4 model, few-shot learning outperforms zero-shot learning. But for the pretrained Llama-7B-chat model, few-shot learning produces much worse results compared to zero-shot learning, attributed to the smaller model's heightened sensitivity to the selection of few-shot demonstrations (Chen et al., 2023; Wang et al., 2024).

For the fine-tuned models, we observe that training on the entire noisy dataset without curation can even degrade model performance. Fine-tuning with only half of the data, refined through automatic filtering, yields better results than utilizing the complete, uncurated dataset. Moreover, training data curated via our Auto-Correct strategy further enhances model performance. Figures 3,4,5 depict for each dataset: a wrong response automatically identified in the Auto-Filter stage that was subsequently corrected in the Auto-Correct stage.

Our fine-tuned models can outperform even the most advanced model, GPT-4 with three-shot prompting. This highlights how even the most powerful LLMs may lack the capability to adequately address specific domain challenges. Unlike some other fine-tuning research, GPT-4 was not involved in any part of the data curation or training process underpinning our fine-tuned LLMs here.

5.1 Estimating Response Quality in Auto-Filter

We compare using a confidence-based response quality evaluator in our Auto-Filter procedure vs. an evaluator based on direct LLM scoring. The latter directly prompts the LLM (say GPT-3.5-turbo) to score a given input-output pair (Li et al., 2023b)

Evaluator	SQuAD-N		Email-N		DROP-N	
	Valid JSON (%)	Accuracy (%)	Valid JSON (%)	Accuracy (%)	Valid JSON (%)	Accuracy (%)
Random	97.50	62.90	100.0	43.0	100.0	65.20
Score-based Evaluator	99.50	78.40	100.0	39.67	100.0	73.00
Confidence-based Evaluator	99.20	81.51	100.0	46.67	100.0	71.70

Table 3: Comparing different variants of the Auto-Filtering procedure. We try filtering the bottom 50% of the data according to 3 different approaches: random scoring, score-based evaluator (Li et al., 2023b), and confidence-based evaluator. For each of the three resulting filtered dataset versions, we fine-tune the GPT-3.5 Turbo model and report its resulting performance. This experiment is repeated across SQuAD-N, Email-N, and DROP-N datasets.

Model used to generate the candidate response	SQuAD-N		Email-N		DROP-N	
	Valid JSON (%)	Accuracy (%)	Valid JSON (%)	Accuracy (%)	Valid JSON (%)	Accuracy (%)
GPT-3.5 Turbo	99.20	77.80	100.0	6.0	100.0	63.00
Fine-tuned LLM	100.0	81.90	100.0	56.33	100.0	73.0

Table 4: Comparing variants of the Auto-Correct procedure. We fine-tune a GPT-3.5 Turbo model on two datasets curated via Auto-Correct applied with candidate responses y' generated from either: the pretrained GPT-3.5 Turbo base Foundation model, or the fine-tuned version of this LLM trained on our Auto-Filtered dataset. GPT-3.5 Turbo is also used as the model to estimate when candidate responses y' are better than the original dataset responses y.

using a Likert scale rating from 1 to 5. Table 5 depicts the prompt used for this score based quality evaluation. After scoring the quality of each (instruction, response) pair in the dataset, we discard the 50% with the lowest scores. Subsequently, we fine-tune the model on the remaining data.

Table 3 presents results comparing this score-based approach against our confidence-based approach from Sec. 3.1. We additionally consider results based on fine-tuning the LLM on a randomly selected 50% of the data. Across all datasets, our confidence-based evaluator either matches or exceeds the performance of the score-based evaluator and random data selection, obtaining significantly better performance in the Email-N dataset.

5.2 Using the LLM in Auto-Correct

Here we consider a variant of our Auto-Correct stage, where we generate alternative candidate responses from the base pretrained Foundation model, instead of from our subsequently fine-tuned version of this LLM. Specifically we consider GPT-3.5 Turbo to generate candidate responses y' which are then fed into the same Auto-Correct procedure described in Sec. 3.2. Table 4 reveals that using the fine-tuned version of this LLM to generate candidate responses performs better across all datasets.

6 Conclusion

This paper presents a general pipeline for curating better versions of an existing instruction fine-tuning dataset. Our data-centric CLEAR approach can be combined with any model and fine-tuning algorithm. While better models and fine-tuning algorithms will inevitably be invented, data-centric approaches like ours can remain useful. As future LLMs advance, their ability to curate the data via CLEAR will advance, facilitating *even better* LLMs to be trained on this better curated data.

Experiments demonstrated that our data curation process produces substantial improvements in the performance of fine-tuned LLMs across different noisy datasets, models, and training algorithms (without being tailored to each setting). While our approach fixes issues in an existing dataset, augmenting this data with additional *synthetic* examples is another data-centric approach that appears promising to combine with CLEAR.

Limitations

While our automated data curation pipeline presents a significant advancement in enhancing the

quality of instruction tuning datasets for large language models (LLMs), it is important to acknowledge its limitations. The pipeline's current framework does not explicitly account for the possibility of biases within the original dataset or those introduced during the automated curation process. Since the model's performance and the quality of its output are contingent upon the data it was trained on, any inherent biases could be perpetuated or amplified through successive iterations of fine-tuning and correction.

References

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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jiuhai Chen, Lichang Chen, Chen Zhu, and Tianyi Zhou. 2023. How many demonstrations do you need for in-context learning? In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 11149–11159. Association for Computational Linguistics.

Jiuhai Chen, Lulu Kang, and Guang Lin. 2021. Gaussian process assisted active learning of physical laws. *Technometrics*, 63(3):329–342.

Jiuhai Chen and Jonas Mueller. 2023. Quantifying uncertainty in answers from any language model via intrinsic and extrinsic confidence assessment. *CoRR*, abs/2308.16175.

Jiuhai Chen, Jonas Mueller, Vassilis N. Ioannidis, Soji Adeshina, Yangkun Wang, Tom Goldstein, and David Wipf. 2022a. Does your graph need a confidence boost? convergent boosted smoothing on graphs with tabular node features. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jiuhai Chen, Jonas Mueller, Vassilis N. Ioannidis, Tom Goldstein, and David Wipf. 2022b. A robust stacking framework for training deep graph models with multifaceted node features. *CoRR*, abs/2206.08473.

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.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3029–3051. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2368–2378. Association for Computational Linguistics.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *CoRR*, abs/2010.11982.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14409–14428. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1896–1907. Association for Computational Linguistics.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C. Bayan Bruss, and Tom Goldstein. 2023. GOAT: A global transformer on large-scale graphs. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17375–17390. PMLR.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1378–1387. JMLR.org.

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. *CoRR*, abs/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. *CoRR*, abs/2308.06259.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, et al. 2022. Dataperf: Benchmarks for data-centric ai development. *arXiv preprint arXiv:2207.10062*.

Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Burr Settles. 2009. Active learning literature survey.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 254–263. ACL.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Xiyao Wang, Yuhang Zhou, Xiaoyu Liu, Hongjin Lu, Yuancheng Xu, Feihong He, Jaehong Yoon, Taixi Lu, Gedas Bertasius, Mohit Bansal, Huaxiu Yao, and Furong Huang. 2024. Mementos: A comprehensive benchmark for multimodal large language model reasoning over image sequences. *CoRR*, abs/2401.10529.

Yangkun Wang, Jiarui Jin, Weinan Zhang, Yongyi Yang, Jiuhai Chen, Quan Gan, Yong Yu, Zheng Zhang, Zengfeng Huang, and David Wipf. 2022. Why propagate alone? parallel use of labels and features on graphs. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. How far can camels go? exploring the state of instruction tuning on open resources. *CoRR*, abs/2306.04751.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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. *CoRR*, abs/2304.12244.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7163–7189. Association for Computational Linguistics.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

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. *CoRR*, abs/2306.05685.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: less is more for alignment. *CoRR*, abs/2305.11206.

Hang Zhou, Jonas Mueller, Mayank Kumar, Jane-Ling Wang, and Jing Lei. 2023b. Detecting errors in numerical data via any regression model. *arXiv preprint arXiv:2305.16583*.

A Prompt for Score-based Answer Quality Evaluator

Below is an instruction from an user and a candidate answer. Evaluate whether or not the answer is a good example of how AI Assistant should respond to the user's instruction. Please assign a score using the following 5-point scale: 1: It means the answer is incomplete, vague, off-topic, controversial, or not exactly what the user asked for. For example, some content seems missing, numbered list does not start from the beginning, the opening sentence repeats user's question. Or the response is from another person's perspective with their personal experience (e.g. taken from blog posts), or looks like an answer from a forum. Or it contains promotional text, navigation text, or other irrelevant information. 2: It means the answer addresses most of the asks from the user. It does not directly address the user's question. For example, it only provides a high-level methodology instead of the exact solution to user's question. 3: It means the answer is helpful but not written by an AI Assistant. It addresses all the basic asks from the user. It is complete and self contained with the drawback that the response is not written from an AI assistant's perspective, but from other people's perspective. The content looks like an excerpt from a blog post, web page, or web search results. For example, it contains personal experience or opinion, mentions comments section, or share on social media, etc. 4: It means the answer is written from an AI assistant's perspective with a clear focus of addressing the instruction. It provide a complete, clear, and comprehensive response to user's question or instruction without missing or irrelevant information. It is well organized, self-contained, and written in a helpful tone. It has minor room for improvement, e.g. more concise and focused. 5: It means it is a perfect answer from an AI Assistant. It has a clear focus on being a helpful AI Assistant, where the response looks like intentionally written to address the user's question or instruction without any irrelevant sentences. The answer provides high quality content, demonstrating expert knowledge in the area, is very well written, logical, easy-to-follow, engaging and insightful. Please first provide a brief reasoning you used to derive the rating score, and then write "Score: " in the last line.

Input: []

Response: []

Table 5: Prompt that Li et al. (2023b) use to have a LLM to directly score instruction-response pairs.

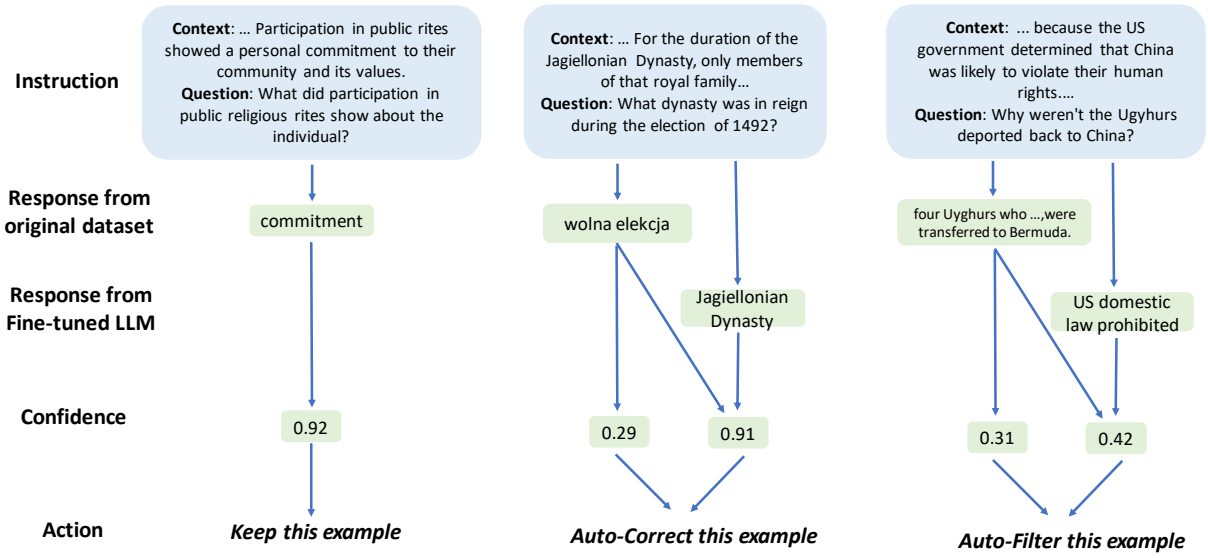

Figure 4: Three examples from the **SQuAD-N** dataset. The first example (left) is retained in the dataset because the original response has high BSDetector-estimated confidence (0.92). The second example (middle) has an original response that is estimated to be low confidence (0.29), and the candidate alternative response generated from our fine-tuned LLM is better than the original response with confidence 0.91. Since this exceeds our confidence threshold $\eta = 0.8$, we replace the target response for this second example with the LLM-generated candidate response in our curated dataset. The third example (right) has an original response that is estimated to be low confidence (0.31), but we also estimate low confidence (0.42) that the candidate response from our fine-tuned LLM is better. This third example is thus entirely removed from our curated dataset.

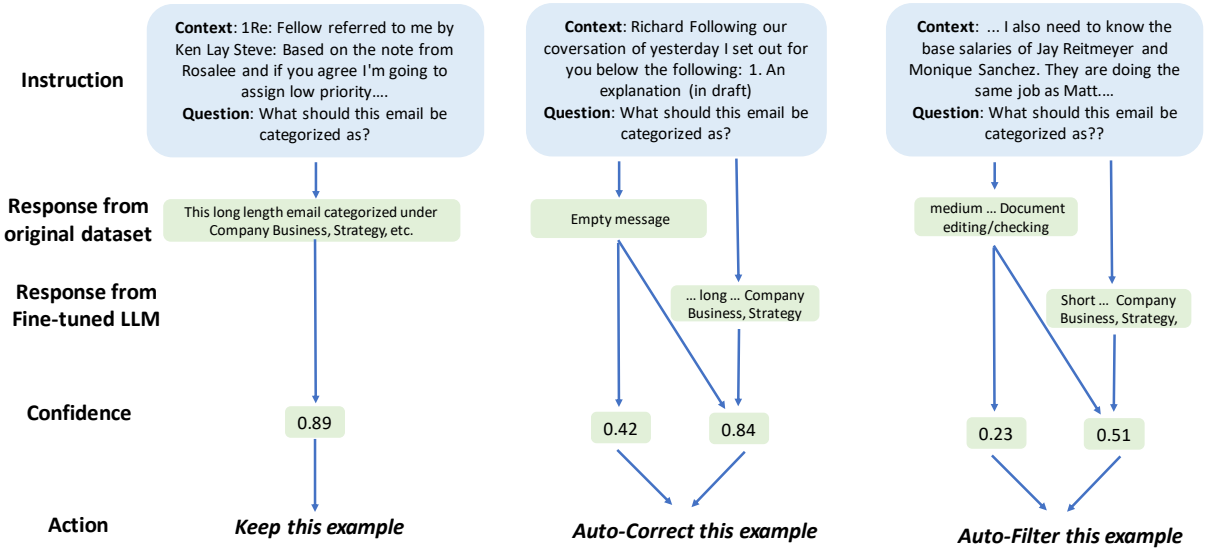

Figure 5: Three examples from the **Email-N** dataset. The first example (left) is retained in the dataset because the original response has high BSDetector-estimated confidence (0.89). The second example (middle) has an original response that is estimated to be low confidence (0.42), and the candidate alternative response generated from our fine-tuned LLM is better than the original response with confidence 0.84. Since this exceeds our confidence threshold $\eta = 0.8$, we replace the target response for this second example with the LLM-generated candidate response in our curated dataset. The third example (right) has an original response that is estimated to be low confidence (0.23), but we also estimate low confidence (0.51) that the candidate response from our fine-tuned LLM is better. This third example is thus entirely removed from our curated dataset.