DONKII: Characterizing and Detecting Errors in Instruction-Tuning Datasets

Leon Weber-Genzel and **Robert Litschko** and **Ekaterina Artemova*** and **Barbara Plank**

MaiNLP, Center for Information and Language Processing, LMU Munich, Germany
Munich Center for Machine Learning (MCML), Munich, Germany
Department of Computer Science, IT University of Copenhagen, Denmark
{leonweber, robert.litschko, b.plank}@lmu.de

Abstract

Instruction tuning has become an integral part of training pipelines for Large Language Models (LLMs) and has been shown to yield strong performance gains. In an orthogonal line of research, Annotation Error Detection (AED) has emerged as a tool for detecting quality problems in gold standard labels. So far, however, the application of AED methods has been limited to classification tasks. It is an open question how well AED methods generalize to language generation settings, which are becoming more widespread via LLMs. In this paper, we present a first and novel benchmark for AED on instruction tuning data: DONKII. It comprises three instruction-tuning datasets enriched with error annotations by experts and semi-automatic methods. We also provide a novel taxonomy of error types for instruction-tuning data. We find that all three datasets contain clear errors, which sometimes propagate directly into instruction-tuned LLMs. We propose four AED baselines for the generative setting and evaluate them extensively on the newly introduced dataset. Our results show that the choice of the right AED method and model size is indeed crucial and derive practical recommendations for how to use AED methods to clean instruction-tuning data.

1 Introduction

Recent successes in instruction tuning (InstT) have shown that Large Language Models (LLMs) can generalize to a wide range of tasks in the zero-shot setting (Wei et al., 2022; Sanh et al., 2022; Ouyang et al., 2022). InstT achieves this by training an LLM on *instruction-output* pairs, where the instruction describes the task and the output contains the expected solution to the task. After fine-tuning on the InstT dataset and an optional reinforcement learning phase (Ouyang et al., 2022), LLMs are able to generalize to instructions not seen during fine-tuning. In an orthogonal line of inquiry, researchers have studied Annotation Error Detection (AED), which allows to detect erroneous annotations in labelled datasets. These low quality instances are then corrected or removed in a semi-automated process (Vlachos, 2006; Klie et al., 2022; Weber and Plank, 2023). However, how to best apply AED for natural language generation has so far not been studied. In this work, we combine both strands of research and ask whether AED methods can help to detect errors in InstT datasets and thus help to improve model quality by improving data quality.

Applying AED methods to InstT datasets presents a number of challenges. (1) The systematic development and comparison of AED methods requires datasets with annotations indicating which instances contain annotation errors. Such datatsets are not yet available for InstT. (2) To our knowledge, researchers have applied AED methods only in the discriminative setting (Klie et al., 2022) and it is not immediately clear how existing methods can be adapted to generative problems. (3) It is not obvious what even constitutes an error in InstT.

In this work, we address these three challenges; see also Figure 1 for an illustration of our contributions:[1] (1) We present Donkii, the first instruction tuning benchmark to enable the evaluation of AED methods. Donkii contains error annotations on top of three existing InstT datasets derived from manual error annotation efforts. We also introduce a hierarchy of error types for InstT datasets; see Figure 2. (2) We derive four AED baselines for generative problems based on recent work on training dynamics for AED (Swayamdipta et al., 2020; Pleiss et al., 2020). (3) We use Donkii to evaluate the proposed AED baselines and study the effects of model size, different types of errors, and different types of InstT data. The results show that there

*Now at Toloka.AI

[1]Data and code are available at https://github.com/mainlp/donkii.

Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII), pages 197–215

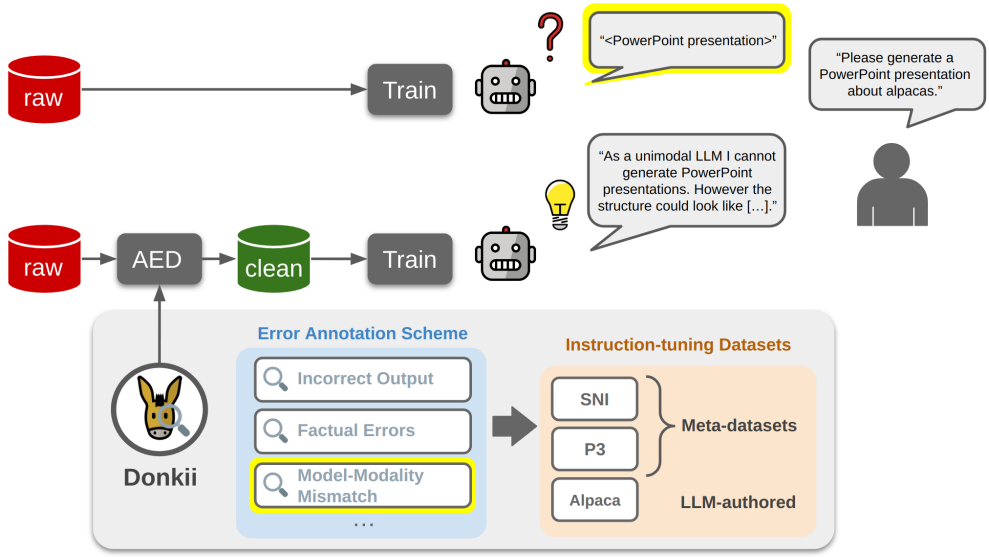

Figure 1: The Donkii dataset helps to design AED methods that can clean InstT datasets.

is a clear best-performing AED method for InstT data among the four evaluated.

2 Background

2.1 Instruction Tuning

Instruction tuning (InstT) is an emerging paradigm that leverages natural language instructions to fine-tune language models, thereby improving zero-shot performance on unseen tasks (Sanh et al., 2022; Ouyang et al., 2022; Wei et al., 2022; Wang et al., 2022b, *inter alia*). In InstT, an LLM is fine-tuned to produce a desired output given an instruction text. In some datasets, the instruction is further divided into a definition or prompt component, which defines the task and an optional input component (Wang et al., 2022b; Taori et al., 2023). In this work, we distinguish *three types of InstT datasets* based on their provenance: meta-datasets, human-authored datasets and LLM-authored datasets.

The first InstT datasets were **meta-datasets**, which convert existing NLP datasets into InstT data with human-authored prompt templates (Khashabi et al., 2020; Ye et al., 2021; Mishra et al., 2022; Sanh et al., 2022; Wei et al., 2022). Researchers typically construct them by writing one or more prompt templates for an existing NLP dataset. This template is then used to transform each instance of the existing dataset into an InstT instance. Here, we call the combination of an existing dataset and a prompt template a task. For **human-authored** InstT datasets on the other hand, the dataset creators ask human annotators to author InstT instances (Ouyang et al., 2022) or mine InstT instances from existing human-authored resources such as forums and wikis (Zhou et al., 2023). **LLM-authored** datasets instead are generated by LLMs. This is typically achieved by prompting the LLM with a few examples of what InstT instances look like and instructing the model to generate new ones (Wang et al., 2022a; Honovich et al., 2023; Taori et al., 2023) or by providing elaborate rules about what properties InstT instances should have (Bai et al., 2022; Sun et al., 2023).

Finally, dataset creators have proposed mixtures of these approaches, e.g. by manually correcting LLM-authored instances (Ruebsamen and Contributors, 2023) or by combining instances generated by different approaches (Zhou et al., 2023). Some of these works highlight the importance of high quality data (Zhou et al., 2023; Ruebsamen and Contributors, 2023), but to the best of our knowledge, our study is the first to systematically evaluate AED techniques.

2.2 Annotation Error Detection

AED for Natural Language Processing (NLP) datasets has a long tradition, which has recently been comprehensively reviewed Klie et al. (2022). Existing AED methods can be divided into six different categories (Klie et al., 2022): *variation-based* methods exploit the observation that instances with similar surface forms tend to have the same label (Dickinson and Meurers, 2003; Larson

et al., 2020). *Model-based* methods use a cross-validation scheme to generate predictions for the whole dataset and then use these predictions to flag errors, e.g. by highlighting instances where the predicted label is different from the one assigned (Amiri et al., 2018; Yaghoub-Zadeh-Fard et al., 2019). *Training-dynamics-based* approaches compute statistics on quantities collected during training (Swayamdipta et al., 2020; Pleiss et al., 2020; Siddiqui et al., 2022). *Vector-space-proximity-based* methods assume that instances that are close in a suitable vector space should have the same label (Larson et al., 2019; Grivas et al., 2020). *Ensemble-based* methods use statistics of the predictions of ensemble members to find errors (Alt et al., 2020; Varshney et al., 2022) and rule-based approaches rely on manually defined rules to spot erroneous instances (Květoň and Oliva, 2002). In this work, we focus on training dynamics because they performed well in a recent evaluation (Klie et al., 2022), can be relatively easily adapted to generative settings and have a low computational overhead. We leave the evaluation of other types of methods to future work.

An orthogonal classification of AED methods is into flaggers and scorers (Klie et al., 2022). Flaggers model AED as a binary classification task (error vs non-error) and scorers assign an error score to each instance that reflects the likelihood that the instance contains an annotation error. In this work, we focus on scorers, because the ranking induced by them allows more fine-grained decisions about which instances to inspect (Weber and Plank, 2023) and they can be converted to flaggers by choosing an appropriate threshold (Swayamdipta et al., 2020).

3 Proposed AED baselines for text generation datasets

We present four AED baselines for text generation datasets. For this, we adapt methods based on training dynamics that were previously used for AED in classification problems (Swayamdipta et al., 2020; Pleiss et al., 2020). We chose these methods because they performed well in earlier work (Klie et al., 2022; Weber and Plank, 2023) and because their adaptation to generative settings is relatively straight-forward. All four methods assign an error score to an instance, with a higher score ideally reflecting a higher probability of an incorrect annotation. All scores use the probabilities $p_{e,l}$ that the model assigned to the token l of the instance's output sequence at epoch e during training. We propose the following measures: (1) **Perplexity**, which is the epoch-averaged perplexity of the instance based on $p_{e,l}$:

$$\text{PPL} = \frac{1}{E}\sum_{e=1}^{E} \text{ppl}_e, \tag{1}$$

where E is the number of epochs and ppl_e the perplexity at epoch e. (2) The (negative) **average probability**, determined by averaging $p_{e,l}$:

$$P_\mu = -\frac{1}{E}\sum_{e=1}^{E}\frac{1}{L}\sum_{l=1}^{L} p_{e,l}, \tag{2}$$

where L is the number of tokens in the output sequence.

(3) The (negative) **minimum probability**, derived from the minimum of $p_{e,l}$:

$$P_{\text{min}} = -\frac{1}{E}\sum_{e=1}^{E}\min_{l=1}^{L} p_{e,l}. \tag{3}$$

(4) The (negative) **Area-under-the-Margin score** (AUM) (Pleiss et al., 2020), which we adapt to the generative setting by calculating it for each token in the output sequence and averaging the resulting scores:

$$\text{AUM} = \frac{1}{E}\sum_{e=1}^{E}\frac{1}{L}\sum_{l=1}^{L}\max_{y'_l \neq y_l} p_e(y'_l|x_l) - p_{e,l}, \tag{4}$$

where y_l is the token at position l and $\max_{y'_l \neq y_l} p_e(y'_l|x_l)$ is the maximum probability assigned by the model at epoch e for position l excluding the assigned token. In addition, we consider a variant of each score that uses only the last epoch; see §5.2 for the results of this ablation.

4 Datasets and Error Types

We describe Donkii's three different data sources that we have enriched with annotations of erroneous instances: P3-Donkii, SNI-Donkii, and ADC-Donkii.[2] Each is based on an existing InstT dataset and on manual inspection of the errors in that dataset: P3-Donkii is derived from the meta-dataset Public Pool of Prompts (Sanh et al., 2022), SNI-Donkii from the meta-dataset Super-Natural Instructions (Wang et al., 2022b),

[2]See Appendix A for our data statement.

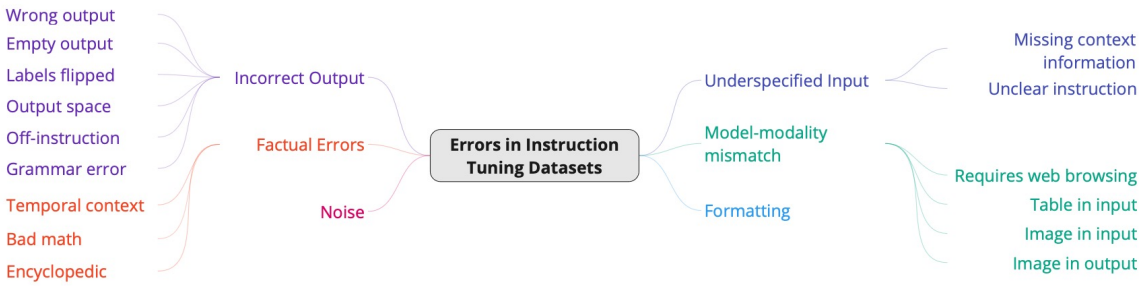

Figure 2: The DONKII taxonomy of six error categories, four of which are further divided into more specific subcategories.

and ADC-Donkii from the LLM-authored dataset Alpaca (Taori et al., 2023) and its partially corrected version AlpacaDatasetCleaned (Ruebsamen and Contributors, 2023). We enrich each of these datasets with labels indicating which instances contain errors, using different mixtures of expert annotation and programmatic analysis of the source data. For each dataset, we construct three different sets of instances: $\mathcal{X}^*$ which contains no errors, $\mathcal{X}_{\text{err}}$, for which we know that it contains errors, and $\mathcal{X}_{\text{unk}}$, for which we do not know if it contains errors. We evaluate AED methods by their ability of discriminating $\mathcal{X}^*$ from $\mathcal{X}_{\text{err}}$. We exclude $\mathcal{X}_{\text{unk}}$ from evaluation, as we do not exhaustively annotate the datasets with errors due to their sheer size and resource availability; see §5.1 for details.

4.1 P3-Donkii

Public Pool of Prompts (P3) (Sanh et al., 2022) is a meta-dataset for InstT which was created by asking researchers and open-source contributors to transform existing datasets by writing prompts using the InstT templating engine *promptsource* (Bach et al., 2022). We construct the P3-Donkii dataset by introducing different types of synthetic errors into the P3 data. We use this synthetic setup for P3 so that we have full control over the the number and types of errors in the dataset.[3] To find realistic error classes, we first detect existing errors in the P3 data. We use PPL to assign error scores to tasks in P3, employing both mean and median aggregation (see §5.2 for details on this).[4] We then manually inspect the top 20 highest scoring tasks by looking at their highest scoring instances. In our manual inspection, we found the following types of problems:

Empty output: The output is an empty string where it should not be.

Incorrect output: The output contains severe orthographic or factual errors.

Missing context information: The prompt is truncated during preprocessing. This can make crucial information unavailable, e.g. missing context in extractive QA.

We then correct the errors that we found in the 20 tasks in P3. We rebuild the empty output data from scratch using *promptsource* and verify that the output strings are not empty. We remove tasks that contain a high number of low quality outputs. We discard instances that do not fit within the set maximum length, so that there is no missing information.

Finally, we reintroduce the detected errors in a controlled manner by modelling them synthetically. For each type of error, we randomly sample five tasks and perturb their instances:

Empty output: We replace the output in all instances for the task with an empty string.

Low quality output: For each instance of the task, with a probability of 0.5, we replace the gold-standard output with output generated by prompting Llama-7b (Touvron et al., 2023).

Missing information: For each instance of the task, with a probability of 0.5, we truncate the gold-standard prompt to half of its original length.

Flipped output: For each instance, with a probability of 0.5, we replace the output with the output of another instance. This is a widely used perturbation used in AED research (Klie et al., 2022), which we adapted to InstT datasets.

We collect all perturbed instances in $\mathcal{X}_{\text{error}}$ (the set of instances the AED methods should detect)

[3] An alternative approach would have been a full manual annotation of P3 which was out of reach because of the large size of the dataset.

[4] In principle, this semi-automatic process of finding error categories with PPL could bias our evaluation results. However, our purely manual analysis of SNI shows similar error categories, so we are confident that the introduced bias is minimal.

and those of the same original[5] unperturbed task in $\mathcal{X}^*$ (the set of instances which should not be detected by the AED methods). The instances of unperturbed tasks constitute $\mathcal{X}_{\text{unk}}$.

4.2 SNI-Donkii

We construct SNI-Donkii by mining errors that arose during the creation of the Super-natural instructions (SNI) (Wang et al., 2022b) dataset and have been corrected for the current version of SNI. SNI[6] is a meta-dataset for InstT, created by a large number of researchers by transforming existing datasets into InstT tasks. It contains a total of 1,616 tasks covering a wide range of NLP tasks such as question answering, text classification, sentiment analysis, textual entailment, and summarization. The project implemented quality control through peer review conducted via GitHub[7] and a crowd-sourced evaluation.

We create SNI-Donkii by comparing the version of each task before the final round of peer review (the first version uploaded to GitHub) and after peer review (the version on GitHub at the time of writing). From the 1,613 tasks that we were able to download without error, we collect all 455 tasks $t \in \mathcal{T}$ where the output of at least one instance changed. For 17 of these tasks, all expert annotators (two co-authors and one NLP MSc student) agreed[8] that at least 90% of the changed instances contain an error. See Table 2 for an example of the annotation task and Table 3 for examples of found errors. For the annotation guidelines see Appendix B.

From this annotation, we construct SNI-Donkii as follows: For all tasks without errors, we add 64 instances of the latest version – or less if the task contains fewer than 64 instances – of it to $\mathcal{X}_{\text{unk}}$. For the erroneous tasks, we add 64 changed instances from the oldest version to $\mathcal{X}_{\text{err}}$ and 64 from the newest version to $\mathcal{X}^*$.[9] When we had fewer than 64 updated instances, we put them all into $\mathcal{X}_{\text{err}}$. In this case, we filled up $\mathcal{X}_{\text{unk}}$ with extra instances from the oldest version to keep the number of instances for each task about the same. Table 1 gives the statistics of the resulting dataset.

4.3 ADC-Donkii

Alpaca (Taori et al., 2023) is an LLM-authored dataset constructed by following the self-instruct recipe proposed by Wang et al. (2022a). The creators of Alpaca repeatedly prompted text-davinci-003[10] with in-context examples of InstT instances sampled from a pool of human and LLM-authored instances and asked the LLM to provide a new instance. This yielded a dataset of 52,000 different InstT instances. In a separate effort called Alpaca-DataCleaned (ADC) (Ruebsamen and Contributors, 2023), members of the open source community corrected errors in the Alpaca data using a mixture of manual and rule-based annotation.[11] To construct ADC-Donkii, we collect 300 instances from Alpaca that do not occur in ADC and pair each of them with the instance with the closest BM25 score from ADC. Using these pairs, three of this study's authors manually annotate whether one of the two instances is clearly preferable because the other contains at least one error. The annotation guidelines can be found in Appendix C. As with SNI, we do not disclose which instances are from Alpaca and which are from ADC to avoid introducing unnecessary bias. If, after a roundtable discussion, all three annotators agree that one of the two instances is preferable, we add it to $\mathcal{X}^*$ and the other to $\mathcal{X}_{\text{err}}$. We add all other instances from Alpaca and ADC to $\mathcal{X}_{\text{unk}}$. Table 1 shows the statistics for the resulting dataset.

4.4 Error categories

During annotation, we identified six main categories of errors, each with several subcategories. Note, that these error categories are not exhaustive and are observed in the annotated sample, rather than encompassing all possible categories of errors. A more detailed and nuanced analysis of possible errors in instruction tuning data is the subject of future work. The proposed hierarchy is shown in Figure 2; a sample from Donkii errors is shown in Table 3. More examples for each category can be

[5] Note, that for all tasks during our initial exploration of the dataset, we use the corrected error-free version of the task.

[6] https://github.com/allenai/natural-instructions

[7] https://github.com/allenai/natural-instructions/commits/master

[8] We opted for a roundtable discussion rather than majority voting because we found that annotating errors in InstT datasets is a difficult task. Even though we relied exclusively on expert annotators, they sometimes missed crucial details about instances and revised their annotations during the discussion. A disadvantage of this discussion-based setup is that we cannot reliably estimate inter-annotator agreement.

[9] We chose 64 instances because Wang et al. (2022b) find that performance plateaus with more instances per task.

[10] https://platform.openai.com/docs/models

[11] https://github.com/gururise/AlpacaDataCleaned/issues/31

	Source data	$\lvert\mathcal{X}_{\text{unk}}\rvert$	$\lvert\mathcal{X}^*\rvert$	$\lvert\mathcal{X}_{\text{err}}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{T}_{\text{err}}\rvert$	$\bar{L}_{\text{inp}}$	$\bar{L}_{\text{out}}$	Err	Prov
P3	Sanh et al. (2022)	399,472	12,237	12,237	417	20	118	9	Syn.	Meta
SNI	Wang et al. (2022b)	101,783	1,088	585	1,613	17	165	6	Nat.	Meta
ADC	Taori et al. (2023) (Ruebsamen and Contributors, 2023)	48,425	173	146	-	-	15	44	Nat	LLM

Table 1: Statistics for the three Donkii datasets. $\lvert\mathcal{T}\rvert$ denotes the total number of tasks, and $\lvert\mathcal{T}_{\text{err}}\rvert$ the number of tasks with at least one instance with an error. Note, that ADC does not provide a grouping of instances into tasks. $\bar{L}_{\text{inp}}/\bar{L}_{\text{out}}$ denotes the average input/output length in white-space-delimited tokens. 'Err' is the type of error (synthetic or naturally ocurring) and 'Prov' the provenance (meta-dataset vs LLM-authored). 'Lic' is the license under which the authors published their data.

Instruction 1: Name two deserts in the Sahara. **Input 1**: **Output 1**: The two deserts in the Sahara are the Great Western Erg and the Great Eastern Erg.
Instruction 2: Recognize the following bird's species. **Input 2**: <Image of a bird> **Output 2**: Western Great Egret (Ardea alba maxima).
Label: 1 is better than 2 **Error category**: Image in input

Table 2: Example for the pair-wise annotation task that we used to flag errors in SNI-Donkii and ADC-Donkii.

found in the Appendix D. The error categories are the following:

Incorrect output: Problems are observed in the output. This may include providing inaccurate or incorrect output, such as providing a three-letter abbreviation when a two-letter abbreviation was requested. Other problems in this category include not providing any output at all, reversing the label in binary classification tasks, and providing output that is in the wrong output space, such as answering a/b/c/d in a multiple choice question when the options are listed as 1/2/3/4. In addition, the output may be an off-instruction response that is related to the instruction but does not follow it, for example, responding with a code example that can calculate an average instead of directly outputting the average of the given numbers as requested. Finally, the output may contain ungrammatical text.

Factual knowledge and mathematics: This category covers outputs that may be time-dependent, contain factual errors, or contain incorrect arithmetic.

Noise: Instances in which the instruction, input, or output contains some form of noise. This noise can range from `NoInput` stubs to duplicating the instruction in the output.

Underspecified input: Instances in which the instruction and input do not provide sufficient information to complete the task. For example, a task may ask to find the average of a set of numbers without giving the actual numbers. This category also includes cases where the instruction is unclear and cannot be completed correctly due to a lack of specification. For example, a task may require classifying data points into multiple categories without explicitly describing the semantics of the categories or providing the data points. We argue that these instances are errors because the LLM should ask the user for the necessary input rather than assuming (i.e. 'hallucinating') input.

Model-modality mismatch: Instances that require additional modalities are placed in this category, to reflect that the examined InstT datasets are used for text-only LLMs. Unsupported modalities may include tables, images (as in Table 2), and the use of additional tools to browse and retrieve information from the web.

Formatting: Instances with corrupted formatting, such as missing white spaces and the use of punctuation instead of white spaces.

In general, the identified errors are very similar to known errors made by models for open-ended text generation (Dou et al., 2022; Ge et al., 2022), with less emphasis on issues of syntax and word choice.

5 How well does AED do in Instruction Tuning data?

In this section, we provide baseline results for error detection in InstT datasets. To do this, we evaluate the baselines introduced in §3 using the Donkii datasets proposed in §4.

Error	Source	Instruction	Input	Output (shortened)
Requires web browsing	Alpaca	Summarize the New York Times article about the rise of AI.	www.nytimes.com/2020/06/16/opinion/ai-growth.html	The New York Times article titled "The Rise of Artificial Intelligence" looks at the growing impact of AI on the world today. <...>
Bad math	Alpaca	Calculate the mean of the numbers 3, 4, 6, 10, 11		The mean of the numbers 3, 4, 6, 10, 11 is 6.4. This is calculated by adding the numbers together and then dividing by the number of values (5). (3 + 4 + 6 + 10 + 11 = 34; 34 / 5 = 6.4).
Unclear instruction	Alpaca	Sort the following 10 fruits into three categories.	Bananas, Apples, Pineapples, Plums, Cherries, Peaches, Grapes, Watermelons, Mangoes, Oranges	Category 1: Bananas, Apples, Pineapples, Plums, Cherries;
Formatting	SNI	[...]	story: I went down to the tidepool to watch the tide roll out. [...] selected sentence: I went down to the tidepool to watch the tide roll out.	I decide togotothe tidepool >Causes/Enables> I gotothe tidepool
Labels flipped	SNI	You are given two sentences (Sentence1 and Sentence2). Answer "Yes" if these sentences are a paraphrase of one another, otherwise answer "No".	Sentence1: The broader Standard & Poor 's 500 Index .SP> gained 3 points , or 0.39 percent , at 924 ., Sentence2: The technology-laced Nasdaq Composite Index .IXIC rose 6 points , or 0.41 percent , to 1,498 .	Yes

Table 3: Examples of some error categories of the Donkii taxonomy.

	small					base				large				xl			
	rand	PPL	P_μ	P_{min}	AUM	PPL	P_μ	P_{min}	AUM	PPL	P_μ	P_{min}	AUM	PPL	P_μ	P_{min}	AUM
P3	50.0	$73.6_{4.4}$	$\mathbf{84.3}_{1.0}$	$51.3_{0.8}$	$52.7_{0.7}$	$76.1_{0.6}$	$81.7_{0.3}$	$51.2_{0.0}$	$53.6_{0.1}$	$70.7_{3.5}$	$77.4_{0.4}$	$49.4_{0.6}$	$52.2_{0.2}$	$61.3_{0.0}$	$66.0_{0.0}$	$58.6_{0.0}$	$46.0_{0.0}$
SNI	34.9	$30.7_{0.2}$	$30.3_{0.2}$	$27.9_{0.1}$	$28.7_{0.1}$	$31.8_{0.4}$	$42.4_{0.9}$	$30.2_{0.3}$	$29.8_{0.1}$	$33.4_{1.0}$	$\mathbf{48.1}_{1.7}$	$34.5_{0.9}$	$34.1_{0.6}$	$33.0_{0.7}$	$38.9_{0.2}$	$32.2_{0.2}$	$31.6_{0.4}$
ADC	45.1	$54.5_{0.3}$	$\mathbf{55.4}_{0.3}$	$50.3_{0.5}$	$47.9_{0.7}$	$53.7_{0.21}$	$51.9_{0.5}$	$50.8_{0.7}$	$51.4_{0.7}$	$52.2_{0.4}$	$48.9_{0.2}$	$47.5_{0.5}$	$47.2_{1.2}$	$53.1_{0.0}$	$53.1_{0.0}$	$51.3_{0.0}$	$47.3_{0.0}$
AVG	43.3	52.9	56.7	43.2	43.1	53.9	**58.7**	44.1	44.9	52.1	58.1	43.8	44.5	49.1	52.7	47.4	41.6

Table 4: Results of four different AED methods applied to the Donkii datasets. All scores are Average Precision in percent. The larger number is the mean across three seeds and the smaller number the standard deviation. The best result per dataset is in bold. Rand is the random baseline.

5.1 Evaluation protocol

We follow the evaluation protocol for scoring-based AED methods for classification tasks of Klie et al. (2022) and Chong et al. (2022) – with one modification. We follow the protocol by treating the problem as a ranking task, where an AED model assigns an error score to each instance x_i. However, unlike Klie et al. (2022) and Chong et al. (2022), we have three sets of instances instead of two: $\mathcal{X}^*$, which contains few to no errors, $\mathcal{X}_{\text{err}}$ which contains many errors, and $\mathcal{X}_{\text{unk}}$ for which we do not know the proportion of errors. We judge the quality of the ranking by how well it distinguishes between $\mathcal{X}^*$ and $\mathcal{X}_{\text{err}}$ and ignore $\mathcal{X}_{\text{unk}}$ during evaluation. Note that while we use only $\mathcal{X}^*$ and $\mathcal{X}_{\text{err}}$ for evaluation, we train on $x_i \in \mathcal{X}^* \cup \mathcal{X}_{\text{err}} \cup \mathcal{X}_{\text{unk}}$. We use average precision (AP), i.e. the area under the precision-recall curve, implemented with scikit-learn (Pedregosa et al., 2011) to score the rankings and use $\frac{|\mathcal{X}_{\text{err}}|}{|\mathcal{X}^*|+|\mathcal{X}_{\text{err}}|}$ as an estimator for the random baseline (Bestgen, 2015).

We conduct all experiments with four models of different sizes from the T5 family[12] (Raffel et al., 2020) in the version that Lester et al. (2021) continually fine-tuned as language models. We chose T5 because it has worked well in previous InstT work (Sanh et al., 2022; Wei et al., 2022; Wang et al., 2022b). See Appendix E for the hyperparameters. We repeat all experiments with three different seeds and report the mean and standard deviation of the results.

5.2 Results

The results can be found in Table 4. On average, P_μ (average probability) performs the best across all model sizes, with PPL coming in second. AUM is tied for the third place with P_{min}, each outperforming the other for two of the four model sizes. This ranking is relatively stable for each individual dataset and the best configuration always uses P_μ. We conclude from this that P_μ clearly emerges as the best performing baseline for AED in our natural language generation setup. This shows the striking benefits in term of simplicity and effectiveness of our proposed P_μ metric. The improvement over the random baseline is relatively large at over 34 percentage points (pp) for P3-Donkii but more modest for SNI-Donkii and ADC-Donkii at 13.2 pp and 10.3 pp respectively. This is probably due to the fact that synthetically introduced errors are generally easier to detect than naturally occurring

[12] https://huggingface.co/google/t5-base-lm-adapt

ones (Klie et al., 2022).

For **model size**, small is the best for P3 and ADC, while large is the best for SNI. On average, base and large perform best, while small also performs surprisingly well. Therefore, for a new InstT dataset, we recommend starting with a base-sized model for efficiency reasons.

P3	out (9777)	inp (2460)	-	-	-
rand	50.0	50.0	-	-	-
P_μ	$89.4_{0.9}$	$68.0_{0.1}$	-	-	-
ADC	out (13)	inp (13)	noi (77)	fac (14)	mul (29)
rand	37.0	48.0	48.4	29.8	50.9
P_μ	$62.6_{0.8}$	$72.2_{0.2}$	$49.8_{0.4}$	$55.7_{0.8}$	$61.5_{0.5}$
SNI	out	form (64)	noi (2)	-	mul (3)
rand	38.2	50.0	3.0	-	2.3
P_μ	$51.7_{1.7}$	$51.9_{0.9}$	$30.6_{8.6}$		$14.9_{3.9}$

Table 5: Results per error category. All scores are AP (higher is better) in percent of P_μ using the best performing model size for the dataset. The category names are abbreviated: out: incorrect output, inp: underspecified input, noi: noise, fac: factual error, mul: multi-modality, form: formatting. The number in brackets gives the number of instances per category.

We analyse the performance of the different scorers per annotated **error category**. For each dataset, we use P_μ with the respective best performing model size. The results can be found in Table 5. Interestingly, the results differ strongly across error categories and dataset. P_μ outperforms the random baseline for all but two categories, which are noisy instances in ADC-Donkii and formatting errors in SNI-Donkii. Surprisingly, other configurations, which on average perform worse than P_μ, are able to beat the random baseline for these error types with the respectively best scorers outperforming random by 18.1/13.7 pp for noise/formatting.

On instance vs task-level and epoch aggregation Our annotation of SNI and P3 showed that errors in meta-datasets often affect a large proportion of all instances for a given task.[13] We wondered whether we could exploit this property by **aggregating error scores across all instances for a given task** and thus perform AED on tasks rather than instances. For this, we conducted additional experiments using SNI-Donkii, where we computed two scores for each task by taking the mean and median across all instances for the given task. We then follow the same ranking-based evaluation protocol as for individual instances. Here, we observe a slightly different ordering of methods, with PPL achieving the highest score. On average, the aggregation by median yielded higher scores than aggregation by mean. The absolute AP is much higher than for single instance error detection at 69.3% (vs 48.1%), suggesting that task aggregation may be useful for detecting systematic errors in meta-datasets.

We also examine the effect of **aggregating scores over all epochs**. For this, we ablate the epoch aggregation by using the final logits directly to compute the AED scores. For each dataset, we compute the difference between the best performing size-score combination with and without epoch aggregation. We find that the scores drop by 1.3/3.9/1.2 percentage points AP for P3/SNI/ADC respectively *without* aggregation over epochs. This further supports the observation that averaging AED scores over epochs generally improves performance (Swayamdipta et al., 2020; Pleiss et al., 2020; Weber and Plank, 2023).

6 Conclusion

This work presents the first study on annotation error detection for generation tasks, in particular, instruction tuning data. Despite the popularity of InsT, there are no evaluation datasets for AED with marked errors. Therefore, we present Donkii, a suite of three existing InstT datasets enriched with novel error annotations and an error taxonomy derived from manual annotation efforts. We propose four different AED methods for generative models and systematically evaluate them on the Donkii datasets. We find that there is a clear best performing method for single instances with P_μ and for task-level AED with PPL. In any case, the choice of model size is critical for optimal AED performance. In Appendix F we report on preliminary experiments in which we investigated how annotation errors impact downstream performance. For future work, we plan to apply AED methods to more structured generative meta-datasets such as Huguet Cabot and Navigli (2021) or Fries et al. (2022).

Limitations

Identified errors in InsT datasets. We acknowledge that the error categories we have identified are not exhaustive. This is because the current errors have been annotated based on manual examination of medium-sized samples. We also acknowl-

[13] This observation motivated our annotation efforts for P3 and SNI.

edge that our error category does not cover issues related to toxicity, hallucinations, and safety, as we believe that these issues are so important that they require specialized treatment in more focused work (Gehman et al., 2020; Sap et al., 2022; Raunak et al., 2021; Dziri et al., 2022; Greshake et al., 2023, *inter alia*).

Small sample size for individual categories We invested significant manual effort in annotation, but strongly favoured precision over quantity, with three expert annotators first labeling each sample individually and then discussing the results. As a result, the number of errors found per category is moderate to small (see Table 5). We believe that an even larger annotation effort would be required in the future to ensure that all findings on error categories are robust.

Ethics & Broader Impact

Instruction-tuned LLMs have been widely adopted by non-expert users (OpenAI, 2023). We believe that this makes fine-grained control over the model outputs and thus, by extension, over the content of the InstT dataset an ethical imperative. One facet of this is errors in the data, and so we believe that using AED methods to analyse InstT datasets can potentially have a positive impact on LLM users. However, the demographics of all annotators are fairly uniform, and yet in some cases there was substantial disagreement on what constitutes an error. Therefore, we believe that a broader discussion involving more stakeholders is needed to get a diverse perspective on what is the desired behaviour of LLMs and thus what constitutes an error in InstT datasets.

Acknowledgments

We thank Shijia Zhou for helping with the annotation of SNI-Donkii. Many thanks to the members of MaiNLP for their comments earlier drafts of on the paper. This research is in parts supported by European Research Council (ERC) grant agreement No. 101043235.

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. TACRED Revisited: A Thorough Evaluation of the TACRED Relation Extraction Task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569, Online. Association for Computational Linguistics.

Hadi Amiri, Timothy Miller, and Guergana Savova. 2018. Spotting Spurious Data with Neural Networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2006–2016, New Orleans, Louisiana. Association for Computational Linguistics.

Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Alshaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-jian Jiang, and Alexander Rush. 2022. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland. Association for Computational Linguistics.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Yves Bestgen. 2015. Exact expected average precision of the random baseline for system evaluation. *Prague Bull. Math. Linguistics*, 103:131–138.

Derek Chong, Jenny Hong, and Christopher Manning. 2022. Detecting label errors by using pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9074–9091, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Markus Dickinson and W. Detmar Meurers. 2003. Detecting Errors in Part-of-Speech Annotation. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary. Association for Computational Linguistics.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2022. Is GPT-3 text indistinguishable from human text? scarecrow: A framework for scrutinizing machine text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7250–7274, Dublin, Ireland. Association for Computational Linguistics.

Nouha Dziri, Sivan Milton, Mo Yu, Osmar Zaiane, and Siva Reddy. 2022. On the origin of hallucinations in conversational models: Is it the datasets or the models? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5271–5285, Seattle, United States. Association for Computational Linguistics.

Jason Alan Fries, Leon Weber, Natasha Seelam, Gabriel Altay, Debajyoti Datta, Samuele Garda, Myungsun Kang, Ruisi Su, Wojciech Kusa, Samuel Cahyawijaya, Fabio Barth, Simon Ott, Matthias Samwald, Stephen Bach, Stella Biderman, Mario Sänger, Bo Wang, Alison Callahan, Daniel León Periñán, Théo Gigant, Patrick Haller, Jenny Chim, Jose David Posada, John Michael Giorgi, Karthik Rangasai Sivaraman, Marc Pàmies, Marianna Nezhurina, Robert Martin, Michael Cullan, Moritz Freidank, Nathan Dahlberg, Shubhanshu Mishra, Shamik Bose, Nicholas Michio Broad, Yanis Labrak, Shlok S. Deshmukh, Sid Kiblawi, Ayush Singh, Minh Chien Vu, Trishala Neeraj, Jonas Golde, Albert Villanova del Moral, and Benjamin Beilharz. 2022. BigBio: A Framework for Data-Centric Biomedical Natural Language Processing. In *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, New Orleans, Louisiana, USA.

Huibin Ge, Xiaohu Zhao, Chuang Liu, Yulong Zeng, Qun Liu, and Deyi Xiong. 2022. TGEA 2.0: A Large-Scale Diagnostically Annotated Dataset with Benchmark Tasks for Text Generation of Pretrained Language Models. *Advances in Neural Information Processing Systems*, 35:31612–31626.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.

Andreas Grivas, Beatrice Alex, Claire Grover, Richard Tobin, and William Whiteley. 2020. Not a cute stroke: Analysis of Rule- and Neural Network-based Information Extraction Systems for Brain Radiology Reports. In *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*, pages 24–37, Online. Association for Computational Linguistics.

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

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. 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*, pages 1896–1907, Online. Association for Computational Linguistics.

Jan-Christoph Klie, Bonnie Webber, and Iryna Gurevych. 2022. Annotation Error Detection: Analyzing the Past and Present for a More Coherent Future. *Computational Linguistics*, pages 1–42.

Pavel Květoň and Karel Oliva. 2002. (Semi-)Automatic Detection of Errors in PoS-Tagged Corpora. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Stefan Larson, Adrian Cheung, Anish Mahendran, Kevin Leach, and Jonathan K. Kummerfeld. 2020. Inconsistencies in Crowdsourced Slot-Filling Annotations: A Typology and Identification Methods. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5035–5046, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Stefan Larson, Anish Mahendran, Andrew Lee, Jonathan K. Kummerfeld, Parker Hill, Michael A. Laurenzano, Johann Hauswald, Lingjia Tang, and Jason Mars. 2019. Outlier Detection for Improved Data Quality and Diversity in Dialog Systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 517–527, Minneapolis, Minnesota. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt

tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. 2020. Identifying Mislabeled Data using the Area Under the Margin Ranking. In *Advances in Neural Information Processing Systems*, volume 33, pages 17044–17056. Curran Associates, Inc.

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.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183, Online. Association for Computational Linguistics.

Gene Ruebsamen and Contributors. 2023. Alpacadatacleaned. `https://github.com/gururise/AlpacaDataCleaned`.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Maarten Sap, Swabha Swayamdipta, Laura Vianna, Xuhui Zhou, Yejin Choi, and Noah A. Smith. 2022. Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5884–5906, Seattle, United States. Association for Computational Linguistics.

Shoaib Ahmed Siddiqui, Nitarshan Rajkumar, Tegan Maharaj, David Krueger, and Sara Hooker. 2022. Metadata Archaeology: Unearthing Data Subsets by Leveraging Training Dynamics.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David D. Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *CoRR*, abs/2305.03047.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

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.

Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022. ILDAE: Instance-Level Difficulty Analysis of Evaluation Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3412–3425, Dublin, Ireland. Association for Computational Linguistics.

Andreas Vlachos. 2006. Active Annotation. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-Instruct: Aligning Language Model with Self Generated Instructions.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022b. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Leon Weber and Barbara Plank. 2023. ActiveAED: A human in the loop improves annotation error detection. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8834–8845, Toronto, Canada. Association for Computational Linguistics.

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

Mohammad-Ali Yaghoub-Zadeh-Fard, Boualem Benatallah, Moshe Chai Barukh, and Shayan Zamanirad. 2019. A Study of Incorrect Paraphrases in Crowd-sourced User Utterances. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 295–306, Minneapolis, Minnesota. Association for Computational Linguistics.

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*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. 2023. LIMA: less is more for alignment. *CoRR*, abs/2305.11206.

A Donkii Data Statement

Following (Bender and Friedman, 2018), the following outlines the data statement for Donkii:

- A. CURATION RATIONALE Enrichment of existing instruction-tuning datasets with annotations for erroneous instances
- B. LANGUAGE VARIETY English with the exact variant(s) unkown because of the large number of different sources of data
- C. SPEAKER DEMOGRAPHIC Unknown because of the large number of different data sources
- D. ANNOTATOR DEMOGRAPHIC Three post-doctoral researchers and one Master's student (age: 25-40), gender: male and female. Native language: Russian, German. Socioeconomic status: higher-education student and university researchers.
- E. SPEECH SITUATION Unknown because of the large number of different data sources
- F. TEXT CHARACTERISTICS Unknown because of the large number of different data sources
- PROVENANCE APPENDIX
  - Alpaca (Taori et al., 2023), CC By NC 4.0, `https://github.com/tatsu-lab/stanford_alpaca`
  - AlpacaDataCleaned (Ruebsamen and Contributors, 2023), Apache 2.0, `https://github.com/gururise/AlpacaDataCleaned`
  - Public Pool of Prompts (Sanh et al., 2022), Apache 2.0, `https://huggingface.co/datasets/bigscience/P3`
  - Super-Natural Instructions (Wang et al., 2022b), Apache 2.0, `https://github.com/allenai/natural-instructions`

B SNI Annotation Guidelines

For this annotation effort, we assume a pair-wise annotation setting. You are shown two tasks and the instances that differ between them. You should

judge whether one of the two tasks contains fewer errors than the other one. Each task has the following fields:

- **Definition**: The instruction to the language model. E.g. 'solve the following equation for x'
- **Instances**: Each with the following fields:
- **Input**: The input complementing the instruction. E.g. 'equation: x + 2 = 5'. Output: The gold-standard output expected from the model

There are four possible labels:

1. A is better than B
2. B is better than A
3. A and B are the same
4. I don't know

Additionally, there is a field for short free-form comments where you can (but don't have to) note a reason for your annotation.

We assume that the dataset is used to train a current text-only vanilla LLM like GPT3 or Llama. That is, it does not have access to tools and cannot process multi-modal input.

The following rules apply for differences between tasks A and B. We assume that the difference mentioned in the rule is the only difference between both (ceteris paribus). If more than one rule applies, we leave the choice to the best judgment of the annotator. The goal is to make only relative judgments for the given pair without considering the "absolute quality" of the instances. Even when both contain very little or many errors, if B clearly contains more significant errors than A, this should be annotated as "A is better than B".

Rules:

- If B contains more errors than A, but those are only few and thus don't affect the majority of the instances differing between A and B, then A and B are equal. As a guideline: If more not more than 90% contain the error, then they probably should be equal.
- Be lenient in your annotations. If you are unsure whether something is an error, then better go for A and B are equal.
- **Factual correctness**: If the output of A can be interpreted as factually correct, but the one in B cannot, then A is better than B. Example:
  - Instruction: Tell me the title of the most popular song released in 2020 so far.
  - Output A: The most popular song released in 2020 so far is "Blinding Lights" by The Weeknd.
  - Output B: The most popular song released in 2020 so far is "The Box" by Roddy Rich.
  - Explanation: A is better than B, because, while the answer to A is ambiguous (there are multiple measures of popularity), "The Box" was released in 2019 and thus is clearly wrong.
- **Noise**: If B contains noise (e.g. technical artifacts) but A does not, then A is better than B. Example:
  - Instruction: Suggest the best strategy for a five-second TV commercial.
  - Input A:
  - Input B: "NoInput"
  - Explanation: A is better than B, because "NoInput" is clearly a technical artifact (even despite A being empty - i.e. no output better than noise).
- **Only output**: Judge A and B based on the output field not instruction or input. Justification: It is not clear whether low-quality input with high-quality output improves or diminishes instruction tuning performance. Example:
  - Instruction A: Convert the following number in hexadecimal format.
  - Input A: 18
  - Instruction B: Convert the number 18 to hexadecimal
  - Input B:
  - Explanation B: A and B are equal, even though one could prefer A over B because input and instruction are cleanly separated.
- **Unclear instruction**: If it is impossible to guess user intent based on the instruction in B, but it is possible to guess it in A, then A is better than B. Example:

- Instruction A: Find the average value of the following list of numbers
- Instruction B: Process the following data and output the results
- Input: List: [3, 7, 2, 5]
- Explanation: A is better than B because for B it is not clear at all how the model should process the data.

- **Tool usage**: If B requires tool usage (e.g. access to a search engine) but A doesn't, then A is better than B. Justification: We assume that the dataset is used to instruction-tune a vanilla LM without access to tools. Example:
  - Instruction A: Provide a brief overview about the following topic.
  - Input A: Volcanology
  - Instruction B: Take a Wikipedia article and rewrite it in your own words.
  - Input B: https://en.wikipedia.org/wiki/Volcanology
  - Explanation: A is better than B because B requires access to a web browser.

- **Multi-modal input**: If B contains multi-modal input (e.g. an image file) but A doesn't, then A is better than B. Justification: We assume that the dataset is used to instruction-tune a vanilla text-only LM. Example:
  - Instruction: Critique the given painting.
  - Input A: The painting is an abstract composition of vibrant yellow, blue, and pink hues that appear in an haphazard, yet balanced form and serve as an evocation of life, joy, and emotion.
  - Input B: [Painting attached]
  - Explanation: A is better than B because B contains multimodal input.

- **Temporal knowledge**: If B contains temporal knowledge but A doesn't, then A is better than B. Justification: We want the instruction-tuned model to handle temporal knowledge gracefully. Example:
  - Instruction A: What is the name of the 46th president of the United States?
  - Instruction B: What is the name of the current president of the United States?
  - Explanation: A is better than B because the answer to B will change over time while the answer to A is static.

- **Formatting**: If A and B differ only in formatting, then A and B are equal Example:
  - Output A: - Astonished - Amazed - Shocked - Stunned - Speechless - Bewildered"
  - Output B: Astonished, amazed, shocked, stunned, speechless, bewildered. Explanation: A is equal to B because the output only differs in formatting

C ADC Annotation Guidelines

For this annotation effort, we assume a pair-wise annotation setting. You are shown two instances and have to judge which of both would you preferably include in an instruction-tuning dataset. Each instance has two to three fields:

- **Instruction**: The instruction to the language model. E.g. 'solve the following equation for x'
- **Input** (optional): The input complementing the instruction. E.g. 'equation: x + 2 = 5'. Instructions can be self-contained, thus Input is optional.
- **Output**: The gold-standard output expected from the model

There are four possible labels:

1. A is better than B
2. B is better than A
3. A and B are the same
4. I don't know

Additionally, there is a field for short free-form comments where you can (but don't have to) note a reason for your annotation.

We assume that the dataset is used to train a current text-only vanilla LLM like GPT3 or Llama. That is, it does not have access to tools and cannot process multi-modal input.

The following rules apply for differences between instances A and B. We assume that the difference mentioned in the rule is the only difference between both (ceteris paribus). If more than one rule

applies, we leave the choice to the best judgment of the annotator. The goal is to make only relative judgments for the given pair without considering the "absolute quality" of the instances. Even when both are very high or low quality, if B is clearly worse than A, this should be annotated as "A is better than B".

Rules:

- **Factual correctness**: If the output of A can be interpreted as factually correct, but the one in B cannot, then A is better than B. Example:
  - Instruction: Tell me the title of the most popular song released in 2020 so far.
  - Output A: The most popular song released in 2020 so far is "Blinding Lights" by The Weeknd.
  - Output B: The most popular song released in 2020 so far is "The Box" by Roddy Rich.
  - Explanation: A is better than B, because, while the answer to A is ambiguous (there are multiple measures of popularity), "The Box" was released in 2019 and thus is clearly wrong.
- **Noise**: If B contains noise (e.g. technical artifacts) but A does not, then A is better than B. Example:
  - Instruction: Suggest the best strategy for a five-second TV commercial.
  - Input A:
  - Input B: "NoInput"
  - Explanation: A is better than B, because "NoInput" is clearly a technical artifact (even despite A being empty - i.e. no output better than noise).
- **Only output**: Judge A and B based on the output field not instruction or input. Justification: It is not clear whether low-quality input with high-quality output improves or diminishes instruction tuning performance. Example:
  - Instruction A: Convert the following number in hexadecimal format.
  - Input A: 18
  - Instruction B: Convert the number 18 to hexadecimal
  - Input B:
  - Explanation B: A and B are equal, even though one could prefer A over B because input and instruction are cleanly separated.
- **Unclear instruction**: If it is impossible to guess user intent based on the instruction in B, but it is possible to guess it in A, then A is better than B. Example:
  - Instruction A: Find the average value of the following list of numbers
  - Instruction B: Process the following data and output the results
  - Input: List: [3, 7, 2, 5]
  - Explanation: A is better than B because for B it is not clear at all how the model should process the data.
- **Tool usage**: If B requires tool usage (e.g. access to a search engine) but A doesn't, then A is better than B. Justification: We assume that the dataset is used to instruction-tune a vanilla LM without access to tools. Example:
  - Instruction A: Provide a brief overview about the following topic.
  - Input A: Volcanology
  - Instruction B: Take a Wikipedia article and rewrite it in your own words.
  - Input B: https://en.wikipedia.org/wiki/Volcanology
  - Explanation: A is better than B because B requires access to a web browser.
- **Multi-modal input**: If B contains multimodal input (e.g. an image file) but A doesn't, then A is better than B. Justification: We assume that the dataset is used to instruction-tune a vanilla text-only LM. Example:
  - Instruction: Critique the given painting.
  - Input A: The painting is an abstract composition of vibrant yellow, blue, and pink hues that appear in an haphazard, yet balanced form and serve as an evocation of life, joy, and emotion.
  - Input B: [Painting attached]
  - Explanation: A is better than B because B contains multimodal input.
- **Temporal knowledge**: If B contains temporal knowledge but A doesn't, then A is better

than B. Justification: We want the instruction-tuned model to handle temporal knowledge gracefully. Example:

  - Instruction A: What is the name of the 46th president of the United States?
  - Instruction B: What is the name of the current president of the United States?
  - Explanation: A is better than B because the answer to B will change over time while the answer to A is static.

- **Formatting**: If A and B differ only in formatting, then A and B are equal Example:

  - Output A: - Astonished - Amazed - Shocked - Stunned - Speechless - Bewildered"
  - Output B: Astonished, amazed, shocked, stunned, speechless, bewildered. Explanation: A is equal to B because the output only differs in formatting

- **Global properties**: If A and B differ only with respect to the full dataset, e.g. because A increases grammatical diversity in the input but B doesn't, then A and B are equal Example:

  - Instruction A: The average of 10 numbers is 85. If the numbers 70 and 76 are removed from the set of numbers, what is the average of the remaining numbers?
  - Instruction B: Find the average of 85, 44 and 102.
  - Explanation: A and B are equal, even though more complex problems as A are much less frequent in the whole dataset than problems of the type B.

- **Subjectivity**: If the annotator feels that their preference for one instance is strongly subjective, then A and B are equal.

- **Leniency**: Be lenient in your annotations. If you are unsure whether something is an error, then better go for A and B are equal. If both outputs can be interpreted as correct, then A and B are equal

D Examples of errors

Error	Source	Instruction	Input	Output (shortened)
			Incorrect output	
Wrong output	SNI	Given a sentence and an entity, the task is to select the authors sentiment towards the enity. Sentiments can be Positive, Neutral and Negative. <...>	What is the sentiment of the following document towards the entity Hayley Smith ? Hayley Smith was diagnosed with chronic depression in her early twenties: "I'd been bottling up quite a bit through most of my teens. Then it hit me a like a brick wall " she said.	Neutral
Empty output	Alpaca	Create a flow chart to explain the working of a web browser.		
Labels flipped	SNI	In this task you will be given a passage and a yes/no question based on the passage. You should answer the question using the information from the passage.	Superfecundation is the fertilization of two or more ova from the same cycle by sperm from separate acts of sexual intercourse, which can lead to twin babies from two separate biological fathers. [...] question: can a woman produce twins of different fathers?	No
Output space	SNI	You are given a science question (easy-level) and four answer options (associated with "A", "B", "C", "D"). Your task is to find the correct answer based on scientific facts, knowledge and reasoning.	Which group of organisms can show significant trait changes in the shortest period of time? (A) bacteria (B) birds (C) fish (D) reptiles	1
Off-instruction	Alpaca	Given three numbers (x, y, z), find the greatest common factor.	2, 4, 2008	The greatest common factor of 2, 4, and 8 is 2.
Grammar error	SNI	In this task, you are given a question and an answer, you would be asked to create the sentence based on the Question-Answer provided. It should be contained within the Question-Answer provided.	Question: What is the most common type of cancer in adult males? Answer: prostate	Prostate is the most common type of cancer in adult males.
			Factual Errors	
Temporal context	Alpaca	What is the average annual salary for a customer service developer?		The average annual salary for a customer service developer is usually around $74,508.
Bad math	Alpaca	Calculate the average of the following three numbers: 10, 12, 15.		The average of 10, 12, and 15 is 12.67.
Encyclopedic	Alpaca	What is the average air pressure of a human's lungs?		The average air pressure in a human's lungs is between four and seven centimeters of water (4–7 cm H2O).
			Noise	
Noise	Alpaca	Write an essay about the benefits of public transportation.	No input required.	Public transportation is an invaluable resource for communities and cities around the world. <...>
			Underspecified Input	
Missing context information	Alpaca	Write a tweet summarizing this article about an animal rescue in Thailand.	The article	Amazing news! An animal rescue in Thailand has successfully freed 63 bears from animal trafficking. The rescued bears will now receive the proper care and attention they need.
Unclear instruction	Alpaca	Sort the following 10 fruits into three categories.	Bananas, Apples, Pineapples, Plums, Cherries, Peaches, Grapes, Watermelons, Mangoes, Oranges	Category 1: Bananas, Apples, Pineapples, Plums, Cherries; Category 2: Peaches, Grapes, Watermelons; Category 3: Mangoes, Oranges
			Model-modality mismatch	
Requires web browsing	Alpaca	Provide a summarization of the article.	https://en.wikipedia.org/wiki/Artificial_intelligence	Artificial Intelligence, commonly known as AI, refers to the simulation of human intelligence processes by machines, especially computer systems. <...>
Table in input	Alpaca		Dataset of 500 people with the variable "income".	The mean income is $50,000.
Image in input	Alpaca	How would you use this photo in a marketing campaign?	<image included in email>	The photo can be used to create a compelling marketing campaign that draws attention to the product or service. <...>
Image in output	Alpaca	Make a word cloud on the given topic.	Artificial Intelligence	<Word Cloud Output>
			Formatting	
Formatting	SNI	In this task, you will be given a short story. One sentence from the story is chosen. Consider the events that happen before that sentence, or are likely to have happened before it. Does any of them directly cause it, or simply make it possible? You should write your answer in the form " A >causes/enables> B". Try to use phrases and sentences from the story to compose your answer when possible.	story: I went down to the tidepool to watch the tide roll out. I sat on the dock and waited, while listening to my mp3 player. Once the tide was out, I saw Something shiny in the muddy bottoms. I went down and found that it was a gold ring! Today was my lucky day! selected sentence: I went down to the tidepool to watch the tide roll out.	I decide togotothe tidepool >Causes/Enables> I gotothe tidepool

Table 6: Examples of errors.

E Hyperparameters

We experiment with four sizes, namely small (60 million parameters), base (220 million), large (770 million), and 3B (3 billion) using NVIDIA A100 cards. We train each of the models for 10 epochs as a seq2seq LM using a batch size of 60 and a learning rate of $1e-3$. Note, that we train separate models for each of the three datasets and leave the exploration of possible synergies across datasets for future work. We set the maximum source length to 512/768/768 for P3-Donkii/SNI-Donkii/APC-Donkii and the output length to 256.

F Preliminary experiments on the impact of errors on downstream performance

We conduct a preliminary experiment on how errors in InstT datasets affect downstream performance in a case study. For this, we use the training and evaluation setup of Tk-Instruct (Wang et al., 2022b), which is the main model trained on SNI using the code provided by the authors.[14] To investigate the effect of errors, we contrast two models: Tk-Instruct$_{\text{err}}$ and Tk-Instruct*. Both models are based on the three billion parameter version of T5. For Tk-Instruct$_{\text{err}}$, we use all 17 tasks that we found to be erroneous in SNI-Donkii. To these, we add a sample of an additional 100 tasks from the original Tk-Instruct training data. This results in a training data set of $6,985$ instances across 117 tasks and an error rate of approximately 8%. For Tk-Instruct* we replace all erroneous instances with corrected instances from the same task. We adapt the training pipeline to our limited computational budget: We train and evaluate the model in a strict zero-shot setting without providing few-shot examples to reduce the input length of the instances. Second, we use only at most 64 instances per task, because Wang et al. (2022b) find that increasing this number does not improve performance. We train the model for 30 epochs with a batch size of 1024. We use the same held-out task mixture for evaluation as Wang et al. (2022b) but remove all tasks that are in our training data. To evaluate the impact of errors on instruction tuning, we follow Wang et al. (2022b) and use RougeL for evaluation. Surprisingly, we find that the difference between the two models is small. Tk-Instruct* achieves an overall RougeL score of 35.9%, while Tk-Instruct$_{\text{err}}$ achieves 35.7%. Moreover, Tk-Instruct$_{\text{err}}$ even generates the correct answer for instances where it observed incorrect answers during training. Both observations suggest that instruction-tuned models may be robust to small numbers of errors in their training data. However, when we prompt the published version of T0[15], the model trained on P3, with a prompt template for which during training it erroneously always observed empty strings as output[16], we find that it will always respond with an empty string. This motivates further research into when and how errors in InstT datasets propagate into models.

[14] https://github.com/yizhongw/Tk-Instruct

[15] https://huggingface.co/bigscience/T0_3B

[16] "Question 1: [...]? Question 2: [...]? Do these questions convey the same meaning? Yes or no?"