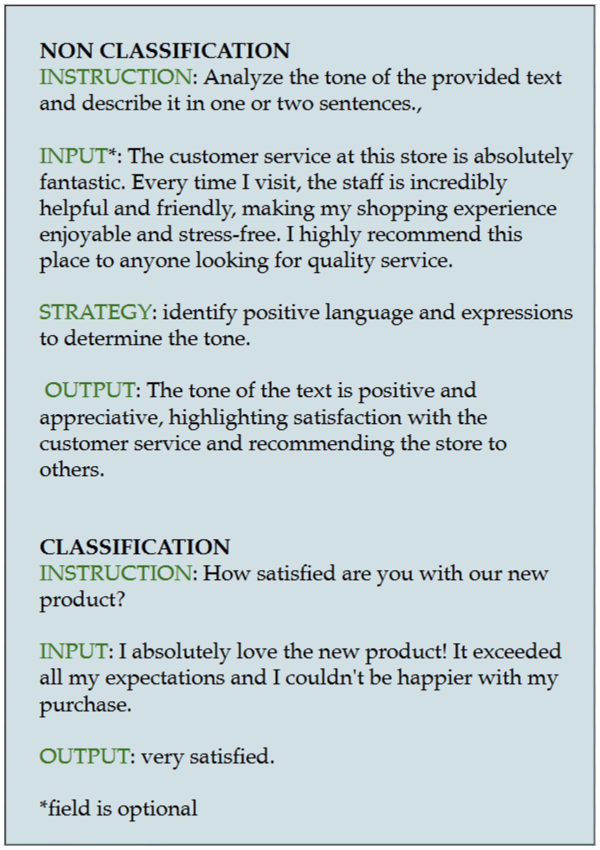
Auto-Instruct: knowledge distillation through attributed instruction generation.

Federico Mansutti [*] [1]

Abstract

LLMs have recently grown in popularity and become indispensable tools for many tasks. In particular, large language models (LLMs) have shown great ability to generalize on new tasks. However, the problem of aligning the LLM objective to human preferences is critical for them to serve as effective assistants. To such an end, instructional data is used to simulate a user-assistant interaction better and improve the model's generality. However, such data is not always available, and if available, it might not be significant in scale, preventing an effective alignment. To this end, I propose Auto-Instruct, a knowledge distillation technique that uses a larger model (in this specific case, GPT-4o) to iteratively construct a larger instruction dataset through bootstrapping. Being a synthetic dataset, it's cheaper by orders of magnitude than a dataset constructed by humans. It demonstrates a 6x cost reduction over comparable state-of-the-art synthetic datasets and is significantly more diverse regarding attributes and instructions.



NON CLASSIFICATION
INSTRUCTION: Analyze the tone of the provided text and describe it in one or two sentences.,

INPUT*: The customer service at this store is absolutely fantastic. Every time I visit, the staff is incredibly helpful and friendly, making my shopping experience enjoyable and stress-free. I highly recommend this place to anyone looking for quality service.

STRATEGY: identify positive language and expressions to determine the tone.

OUTPUT: The tone of the text is positive and appreciative, highlighting satisfaction with the customer service and recommending the store to others.

CLASSIFICATION
INSTRUCTION: How satisfied are you with our new product?

INPUT: I absolutely love the new product! It exceeded all my expectations and I couldn't be happier with my purchase.

OUTPUT: very satisfied.

*field is optional

Figure 1. Examples of a classification and a non-classification instruction generated by Auto-Instruct.

1. Introduction

LLMs have gained significant popularity recently due to their impressive performance in modeling language and retaining textual knowledge. As they become integral to the safety critical domain, it's essential to ensure that the behavior of LLMs is safe, trustworthy, and meets the performance expectation set by humans on a specific task.

One crucial step to their success is pre-training (Brown et al. 2020), which is responsible for many of the observed capabilities due to the simple but effective training objective of the next token prediction. This step involves training models with unsupervised learning on massive datasets from multiple sources, including web pages, books, research papers, and more. The scale and quality of these datasets are crucial, with smaller models often outperforming bigger models on larger datasets (Goodfellow, Bengio, and Courville 2016). However, even scaling these datasets does not solve a significant limitation: the inability to follow human instructions (Ouyang et al. 2022) (Xu et al. 2024) (Schulman et al. 2017). Specifically, the large corpus of text on which language models are trained often has a tiny

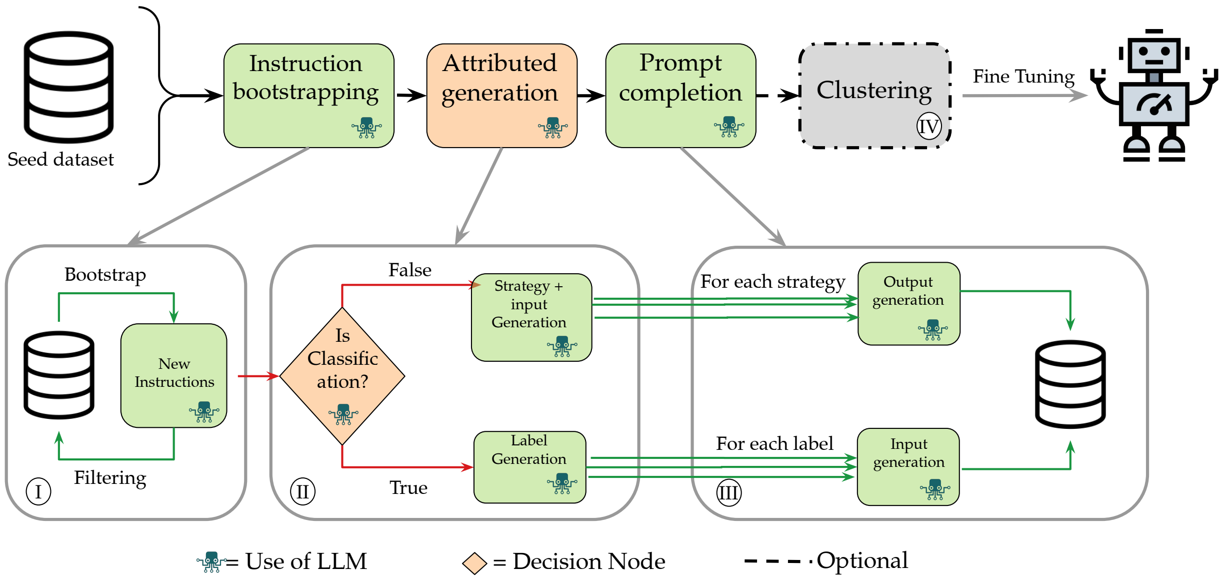


Figure 2. Proposed Auto-Instruct pipeline consisting of four separated pipelines: the first three utilize a LM as indicated by the corresponding symbol, while the last utilizes unsupervised clustering to generate an instructional training dataset.

portion of instructional data, and the lack of variety of such instructions prevents the model from following human instructions. Although much effort has been made in various fields such as RLHF (Rafailov et al. 2023) to better align models with human instructions, an optimal solution has not yet been found, and research is still ongoing. In an effort to bridge this gap, an increasingly popular topic in the current literature is the capabilities of LLMs to "self-improve" by generating their own instructional datasets through various prompting strategies (Wang et al. 2023) (Huang et al. 2022).

In this work, I present Auto-Instruct, a semi-automated method of instruction tuning a language model through knowledge distillation from an instruction-tuned model. Specifically, I bootstrap a smaller dataset of seed instructions, as done by (Wang et al. 2023) to generate a larger instructional dataset by enforcing diversity between the generated instructions. Subsequently, the instructions are split between non-classification and classification instruction. The first is passed through an augmentation process by which we generate possible "strategies" to answer the instruction by calling the instruction-tuned, more powerful model. The same is done for classification instructions, but the possible output labels are generated instead of the strategies since such tasks tend to be much more specific. This combined attributed strategy ensures that more knowledge is distilled from the parent language model.

Finally, for each strategy, the LM is prompted to generate the output while following such a strategy. Each label is treated as the output for classification tasks, and the LM is then used to create the input. This completes our augmented dataset, which is used to fine-tune a less powerful language model with samples chosen through stratified clustering.

Specifically, although the initial pipeline was taken from self-instruct (Wang et al. 2023), my work differs as follows: ① they use the same LLM they want to fine-tune to generate instructional data; ② my attributed strategy and label generation ensures that each strategy and each label is chosen uniformly; instead, self-instruct directly generates the classification output together with the input in one step, thus preventing proper control over label generation and with no control at all over the strategy of non-classification prompts; ③ I cluster the instances and then use stratified sampling to maintain a balanced proportion of strategies, classification tasks, and labels during fine-tuning; ④ most importantly the crucial difference is that while they aim to augment the LLM through a "self-learning" process using the same large language model, I aim to perform knowledge distillation from a better large language model, since this process still remains trivially inexpensive with respect to constructing a human-made instructional dataset.

The contributions of this work are: (1) Auto-instruct: an instructional data generation process through knowledge distillation that requires minimal human intervention; (2) A novel strategy-based attributed generation for nonclassification instructions; and (3) A synthetic dataset of 50K instructions generated through Auto-Instruct.

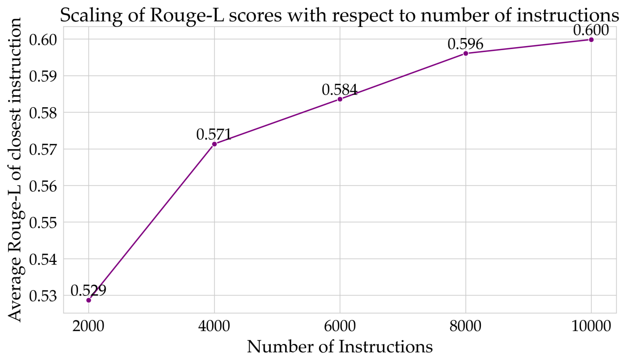


Figure 3. Filtering with ROUGE-L exponentially affects the generation cost with increasing data-set size.

2. Method

2.1. Instructions and attributed data

Any open-ended question for which the model must provide a response is considered an instruction. Each instruction can have an additional specification described in an input field. The model response is stored in the output field. Depending on the instruction, the expected output can be a classification label or an open-ended response. An attribute is either (1) an output label for a classification task or (2) a resolution strategy for a specific instruction/input combination. Since we first generate the output labels and then the input for classification tasks, the label could also be considered a resolution strategy for generating the input. The idea is to provide a synthetic instructional dataset with a more uniform distribution of strategies and labels.

2.2. Overview

Since each instruction can potentially contain 4 fields: the instruction itself, an input, a strategy/label and an output. It's necessary to split the generation process in several stages, as outlined by Fig. 2. The pipeline of my proposed approach consists of four main stages: instruction bootstrapping (1), attributed generation (2), prompt completion (3), and clustering (4). An icon marks each section if an LM is prompted during that step. During each call, the LM is encouraged to "think step by step" by appending this suffix at the end of each request. This has been shown to improve zero-shot generalization (Prystawski, Li, and Goodman 2023) (Kojima et al. 2023) (Zhang et al. 2022) especially crucial when prompted to provide outputs less prevalent in its training corpus.

2.3. Instruction bootstrapping

This paper's proposed methodology utilizes the seed instruction set used in (Wang et al. 2023) to initialize our process. The starting pool is initialized with 175 tasks, each with 1 instance. Iteratively, we sample 6 instructions from this pool as examples and 2 from the machine-generated instructions, for a total of 8 examples. At each step, each newly generated instruction is compared with each previously generated instruction through the ROUGE-L similarity metric and is discarded if it is more than 0.7. Due to the exponential cost of requiring all new instructions to be dissimilar to the previous, as shown by Figure 3 since this pool is increasing step by step. Consequently, I propose to stop at 10k instructions and rather augment the size through instance generation.

2.4. Attributed Generation

Each instruction generated in the first stage is queried through an LM, which is tasked to identify which of these instructions belong to classification tasks and which do not. The classification and non-classification instructions are separated into two different pools, **CLF** and **NON-CLF** for clarity.

If the instruction belongs to **NON-CLF** pool, the same LM is asked to provide both an input and a set of up to three strategies and encouraged to form the input first and then the strategies. This is derived from the fact that in open-ended, non-classification questions, the resolution often depends on the specific input question. At the same time, the instructions can possibly remain too generic to provide an appropriate set of strategies. The strategies are limited to three to encourage the language model to be diverse and only include the most relevant approaches to resolution. Several examples are included in the prefix of the prompt.

Otherwise, the instruction belongs to the **CLF** pool, the language model is prompted to find the possible output labels of the classification task, and several examples are provided as input.

2.5. Prompt completion

In this stage, both the **CLF** and the **NON-CLF** pool are kept separate and treated individually: for instructions in the CLF pool, for each possible output label, we derive the input through an LM query. Intuitively, the possible output labels are already the output of the model. This dual-stage process enables the LM to generate an input that is specific and targeted to the output label, as well as ensuring a uniform distribution over all possible output labels to guarantee an equally attributed dataset and a smoother learning process.

2.6. Clustering

Stage 4 involves an initial instance/instruction filtering step, after which a clustering procedure is applied. This work uses Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) to obtain a low-rank data

Table 1. Generated Dataset statistic during the corresponding phases indicated in figure 2: 1) Instruction bootstrapping, 2) Attributed Generation, 3) Prompt completion, and 4) Clustering. Fields that are not applicable during a specific phase are indicated by ”-”.

STATISTIC	1: BOOTSTRAPPING	2: ATT. GEN.	3: PROMPT COMPL.	4: CLUSTER (UPSAMPLED 50K)
N. OF UNIQUE INSTR.	10.0K	10.0K	10.0K	9.20K
N. OF INSTANCES	10.0K	10.0K	19.2K	50.0K
EMPTY INPUT	-	-	16.3K	37.3K
CLF INSTRUCTIONS	-	161	161	152
CLF INSTANCES	-	161	515	5.20K
AVERAGE LABEL COUNT:	-	3.20	3.20	3.28
NON-CLF INSTRUCTIONS	-	9.8K	9.8K	9.10K
NON-CLF INSTANCES	-	18.7K	18.7K	45.0K
AVERAGE STRATEGY COUNT	-	1.90	1.90	1.90

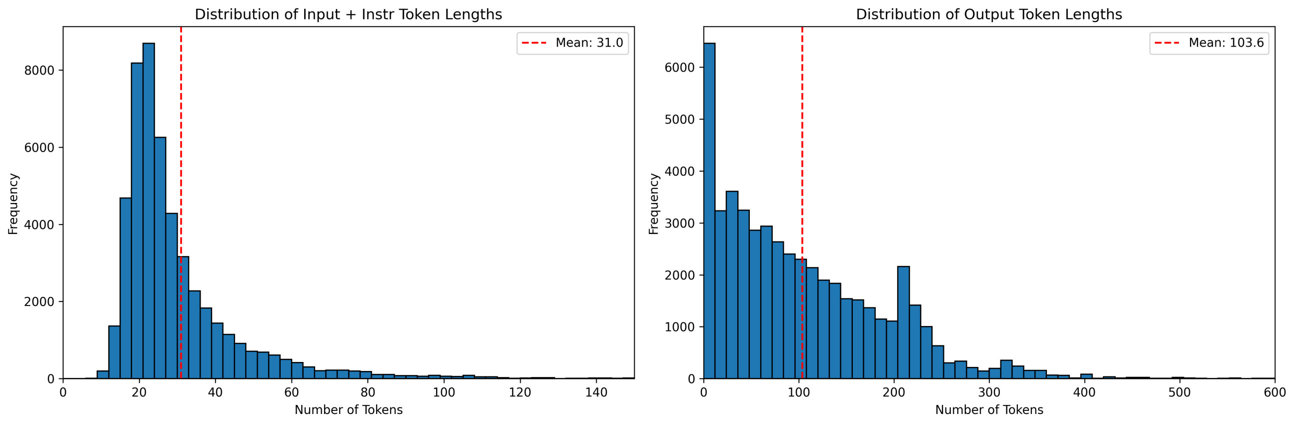


Figure 4. Distribution of upsampled training dataset with respect to the sum of input + instruction token lengths (left) and the distribution of the outputs (right).

representation and then Gaussian Mixtures to select the representable clusters.

A basic filtering procedure is first applied to the dataset: each input-output pair is checked for duplication, meaning that only 1 pair of input-output is kept for each instruction. Then there is an invalid instance check: if the input and output are the same string if there is a missing output, if the output contains ”Strategy:” or ”Input:” or if the last token in the output string is a connective word such as ”and”, ”or” etc. since it’s an indication of stopped token generation due to token length limits.

For the dimensional reduction step, Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) is applied directly to the set of unique instructions since all instances (strategies/labels) belonging to the same instruction intuitively should belong to the same cluster. Intuitively, UMAP was chosen with respect to other approaches, such as SVD or PCA, since it detects non-linear relationships between data points. In UMAP, the first step consists of finding a fuzzy topological representation of the dataset. In contrast, the second consists of finding a low-dimensional representation as close as possible to the fuzzy topological representation through gradient descent. More details can be found in the original paper (McInnes, Healy, and Melville 2020).

When fixing instruction and input, one can intuitively consider the model’s outputs to be distributed according to a Gaussian distribution with more similar instruction-input combinations that have relatively closer distributions over output tokens. The assumption is that our data points are samples from a Gaussian Mixture Model with K Gaussians. I apply elbow detection over silhouette scores to choose the best number of clusters. Finally, the cluster in which each instruction is assigned is mapped back to each instruction instance.

To preserve the instruction distribution of the dataset created in stage 1: instruction bootstrapping of the proposed methodology, I create the fine-tuning dataset by sampling

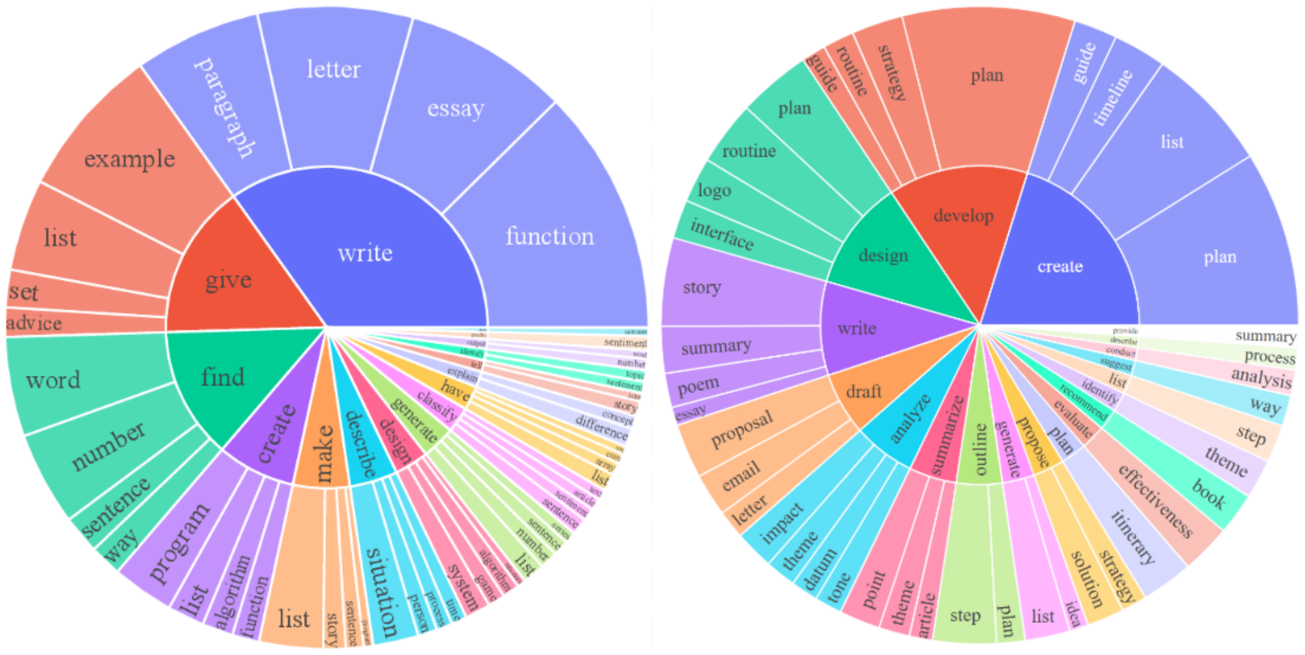


Figure 5. The top 20 common verbs (inner circle) and the top 4 nouns related to each verb. Self Instruct (Left) vs Auto-Instruct (Right) upsampled at 50k instructions

each cluster equally according to the initial instruction distribution; a cluster is sampled, then one instruction in the cluster and lastly one instance belonging to the instruction. This prevents instructions with a larger proportion of output labels or strategies from getting a larger weighting in the dataset.

3. Experimental Results

3.1. Dataset Generation through GPT-4

Since the dataset generation is relatively inexpensive compared to a set of instructions and answers written by human experts, I use GPT-4o queried through the Open-AI API [1] in order to create the pool of instructions, each with possibly several instances of strategies/labels, to construct the highest quality synthetic dataset possible. Remarkably, the cost of dataset generation with my proposed method is less than 80$ when queried through GPT-4o; in perspective, an optimized implementation of the self-instruct pipeline (Taori et al. 2023) requires a cost of approximately 500$ queried on text-davinci-003 (smaller model) showing an improvement of over 6x.

As previously mentioned, the dataset generation process begins from a small set of 175 seed instructions, which serve as in-context examples for the instruction bootstrapping. We follow this process until we obtain 10k instructions, with 1 instance per instruction, as shown in Table 1; this marks the end of phase 1. It can be noticed that the proportion of classification instructions is actually very low compared to the dataset size. The template used for prompting is in Appendix A.

Following my proposed methodology, the classification and non-classification instructions are split: for the first group, a set of output labels is generated, while for the second, a set of strategies. The average label count is 3.2 due to multiple possible outputs during classification, while the average number of strategies is 1.9 because of the strategy number constraint between 1 and 3. More information is in Appendix B.

Phase 3 splits the instructions for each instruction/label, increasing the number of instances to 19.2k while keeping the same number of unique instructions. Then, inputs/outputs are generated according to the instruction label/strategy. A total of 16.3k (85%) instructions are without an input, meaning that just the LLM considered the instruction component to be enough for it to make sense. Appendix C contains the prompts used.

Finally, Phase 4 filters out about 2k instances using the process previously described, clusters with Gaussian mixtures using an elbow detection mechanism, and creates a training dataset through stratified sampling. Due to the smaller size of classification instructions, they are upsampled to about

[1] https://platform.openai.com/docs/overview

Table 2. Human evaluation, % of ”Yes” to validity questions

QUESTION	SELF-INSTRUCT	AUTO-INSTRUCT
DOES THE INSTRUCTION DESCRIBE A VALID TASK?	92%	96.2%
IS THE INPUT APPROPRIATE?	79%	84.6%
IS THE STRATEGY VALID (IF NOT EMPTY)?	-	94.0%
IS THE OUTPUT CORRECT?	58%	92.3%

20% of the entire dataset.

3.2. Baseline

The methodology proposed by Self-Instruct (Wang et al. 2023) is at the foundation of many recent models: Stanford Alpaca (Taori et al. 2023) is tuned using the synthetic dataset generated through the self-instruct pipeline with minor changes as well as many other synthetic dataset generations (Zhao et al. 2024) (Hu et al. 2024). Therefore, their produced dataset will be used as my baseline. Figure 5 shows a direct comparison of root verbs - the base form of a with no changes or conjugations - and the top 4 nouns related to each verb. The Auto-Instruct dataset maintains diversity and exceeds self-instruct even when upsampled at 50k instructions.

3.3. Human evaluation

While in self-instruct, the authors self-evaluate their dataset, 5 graduate students in computer science act as human evaluators in Auto-Instruct to determine whether a random sample from the generated dataset makes sense regarding instruction, input, strategy, and output. Each graduate student is tasked to answer Yes or No to 10 randomly sampled instructions from the dataset for a total of 50 instructions. The results are then compared to self-instruct and shown in Table 2. Since self-instruct doesn't us strategies, the corresponding value is indicated with ”-”. As observed, the validity of each entry of Auto-Instruct's synthetic dataset greatly improved over the proposed baseline, achieving near-perfect validity across all fields. Appendix D contains more details.

4. Limitations

4.1. Low presence of classification tasks

Classification tasks cover less than 2% of the generated dataset. Although the proposed method up-sampled the classification tasks to 10% during the training dataset generation, classification instructional tasks remain important examples for language model fine-tuning due to only 1 correct answer, making the learning task easier. Although this issue was partially offset by the upsampling in the last step of the proposed methodology, in future work, it's advisable to encourage the LLM to engineer a larger proportion of classification tasks since they constitute important learning examples and are used for many benchmarks in the state-of-the-art such as the SuperNI benchmark (Wang et al. 2022).

4.2. Train-test-split

The instance sampling method for dataset expansion, although effective at augmenting the size of the data set at a much lower cost compared to the state-of-the-art and more than 6x cheaper, necessitates additional considerations when used for fine-tuning. Specifically, there is an increasing risk of data leakage as dataset size increases due to having shared instructions between train, validation, and testing. Therefore, if this strategy is employed, the split must be done after the clustering process but before the sampling strategy is applied.

4.3. Ablation study

Since self-instruct uses GPT-3 and Auto-instruct uses GPT-4o, it's harder to distinguish which component has a greater impact on a diverse dataset: whether the proposed methodology or simply the usage of a more powerful model. Intuitively, the presented approach should indeed aid with diversity through attributed prompts; however, due to the deprecation of the open API and model, experiments could not be run under the same setting as the baseline.

4.4. Fine-tuning

Although planned as an extension of this work, there are currently no fine-tuning results to accurately reflect the performance of the synthetic dataset on a real-world model. However, due to the pipeline's similarity with Self-Instruct, I'm reasonably confident it works well. It's especially important to highlight the possible benefits that smaller models could have when trained on this dataset due to knowledge distillation from GPT-4o.

5. Conclusion

The proposed strategy is highly effective at generating a dataset potentially as large as comparable state-of-the-art synthetic datasets while having an over 6x improvement in computational cost and generating greater instructional diversity through attributed prompts. This is especially promising in a knowledge distillation setting, where a higher quality instructional dataset can be crafted by a more powerful model, which can then be used to fine-tune a smaller model, successfully bridging the gap between pre-training on a larger corpus of data and fine-tuning on a potentially low-cost instructional dataset.

References

Brown, Tom B. et al. (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Book in preparation for MIT Press. MIT Press. URL: http://www.deeplearningbook.org.

Hu, Hanxu et al. (2024). *Fine-tuning Large Language Models with Sequential Instructions*. arXiv: 2403.07794 [cs.CL]. URL: https://arxiv.org/abs/2403.07794.

Huang, Jiaxin et al. (2022). *Large Language Models Can Self-Improve*. arXiv: 2210.11610 [cs.CL]. URL: https://arxiv.org/abs/2210.11610.

Kojima, Takeshi et al. (2023). *Large Language Models are Zero-Shot Reasoners*. arXiv: 2205.11916 [cs.CL]. URL: https://arxiv.org/abs/2205.11916.

McInnes, Leland, John Healy, and James Melville (2020). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426.

Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155.

Prystawski, Ben, Michael Li, and Noah Goodman (2023). “Why think step by step? Reasoning emerges from the locality of experience”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., pp. 70926–70947. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/e0af79ad53a336b4c4b4f7e2a68eb609-Paper-Conference.pdf.

Rafailov, Rafael et al. (2023). “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., pp. 53728–53741. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Schulman, John et al. (2017). *Proximal Policy Optimization Algorithms*. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

Taori, Rohan et al. (2023). *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca.

Wang, Yizhong et al. (2022). “Super-NaturalInstructions:Generalization via Declarative Instructions on 1600+ Tasks”. In: *EMNLP*.

Wang, Yizhong et al. (2023). *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. arXiv: 2212.10560 [cs.CL]. URL: https://arxiv.org/abs/2212.10560.

Xu, Shusheng et al. (2024). *Is DPO Superior to PPO for LLM Alignment? A Comprehensive Study*. arXiv: 2404.10719 [cs.CL]. URL: https://arxiv.org/abs/2404.10719.

Zhang, Zhuosheng et al. (2022). *Automatic Chain of Thought Prompting in Large Language Models*. arXiv: 2210.03493 [cs.CL]. URL: https://arxiv.org/abs/2210.03493.

Zhao, Chenyang et al. (2024). *SELF-GUIDE: Better Task-Specific Instruction Following via Self-Synthetic Fine-tuning*. arXiv: 2407.12874 [cs.CL]. URL: https://arxiv.org/abs/2407.12874.

Appendix A - Phase 1 prompts

6 random samples from the seed tasks and 2 tasks from the machine-generated instructions are given as in-context examples.

Phase 1 - instruction bootstrapping template

Come up with a series of tasks. Write the tasks directly with no preamble.

1. Make a grocery list for a healthy meal.
2. Explain human's behavior.
3. Are you smarter than most redheads?
4. Generate the regular expression based on the description.
5. Suggest a quick pre-run warmup routine. Explain each move briefly.
6. Give me an example of a time when you had to use your sense of humor.
7. The missing number in a sequence of numbers.
8. Outline a fitness plan for someone looking to build muscle and improve endurance over a 12-week period.

Appendix B - Phase 2 prompts

The classification recognition prompt is the same as the one used by self-instruct for better reproducibility. Below is a snippet of the template (some in-context examples have been removed)

Phase 2 - CLF task recognition

Can the following task be regarded as a classification task with finite output labels?

Task: Given my personality and the job, tell me if I would be suitable.
Is it classification? Yes

Task: Give me an example of a time when you had to use your sense of humor.
Is it classification? No

Task: Replace the placeholders in the given text with appropriate named entities.
Is it classification? No

Task: Fact checking - tell me if the statement is true, false, or unknown, based on your knowledge and common sense.
Is it classification? Yes

Task: Return the SSN number for the person.
Is it classification? No

Task: Detect if the Reddit thread contains hate speech.
Is it classification? Yes

Task: Analyze the sentences below to identify biases.
Is it classification?

The prompt used for the strategy generation is the following (still phase 2):

Phase 2 - strategy generation for non clf tasks

Given an instruction, find the main strategies to solve or answer the instruction. Only answer with what you are confident is correct.

Answer with 1, 2, or 3 strategies.
Answer extremely concisely with no preamble.
If needed, also provide the input for the instruction.

Example:
instruction: Develop a marketing strategy for launching a new product, focusing on digital channels and social media engagement.
input: None
strategies: Identify target audience and eco-conscious segments.
Highlight product's eco-friendly benefits in messaging.
Use influencer partnerships with sustainability advocates.

Example:
instruction: Please provide the equation you would like to solve for x.
input: $x^2 + 2x + 1 = 0$
strategies: Solve the equation by factoring. Use the quadratic formula.

Example:
instruction: What is the capital of France?
input: None
strategies: None

...

Now its your turn, provide strategies and input if needed. Follow the same format as the examples above. Make sure to also answer with 2 or even 1 strategy if needed.
Remember: generate first the input and AFTER the strategy/strategies. Think step by step.
instruction: instruction

Next is the label generation template for the classification instructions:

Phase 2 - label generation for clf tasks

Given an classification instruction, find the possible output labels. Only answer with what you are confident is correct.
Answer extremely concisely with no preamble.

Example: Task: Fact checking - tell me if the statement is true, false, or unknown, based on your knowledge and common sense.
labels: true, false, unknown

Example: Task: Detect if the Reddit thread contains hate speech.
labels: yes, no

Follow the same format as the examples above. Think step by step.
instruction: instruction
labels:

Appendix C - Phase 3 prompts

For clf tasks, we have to generate the input since the label is already the output

Phase 3 - input generation for clf tasks

Given the classification task definition and the class labels, generate an input that corresponds to each of the class labels. If the task doesn't require input, just generate possible class labels.

Task: Classify the sentiment of the sentence into positive, negative, or mixed.
Class label: mixed
Input: I enjoy the flavor of the restaurant but their service is too slow.

Task: Given a dialogue, classify whether the user is satisfied with the service. You should respond with "Satisfied" or "Unsatisfied".
Class label: Satisfied
Input:
- Agent: Thank you for your feedback. We will work to improve our service in the future.
- Customer: I am happy with the service you provided. Thank you for your help.

Now it's your turn, remember to output the instance in the same format with no preamble. Follow the same format as the examples above. Let's think step by step.

Task: instruction
Class label: class labels
Input:"'

Next we have to generate outputs for the non-clf tasks based on the provided strategies:

Phase 3 - output generation for non-clf tasks

Come up with examples for the following tasks. Try to generate multiple examples when possible. If the task doesn't require additional input, you can generate the output directly.

Task: Sort the given list ascendingly.
Input: [10, 92, 2, 5, -4, 92, 5, 101]
Strategy: Examine the size of each number.
Output: [-4, 2, 5, 5, 10, 92, 92, 101]

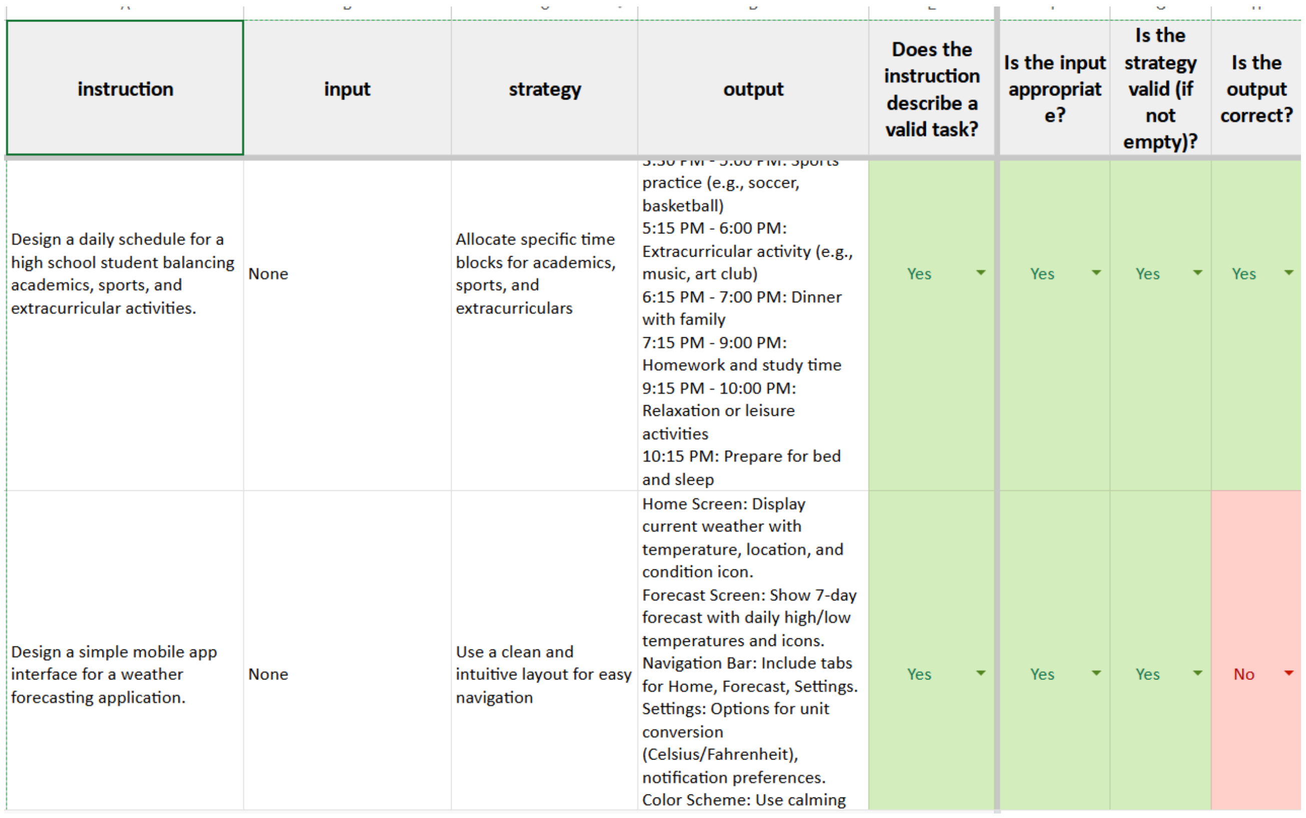
Task: Converting 85 F to Celsius.
Input: None
Strategy: None.
Output: 85°F = 29.44°C

Now it's your turn, remember to output in the same format with no preamble. Follow the same format as the examples above.
Keep the answers extremely CONCISE and to the point. Let's think step by step.

Instruction: instruction
Input: input
Strategy: strategy
Output:

Appendix D - Human evaluation

The data from the 5 graduate students has been collected and is available in the link below. Originally, the survey was meant for 10 students, featuring a total of 100 instructions. However, not all responded (hence, missing responses in the document below). Since only 50 instructions received a response, normalization was applied based on the results obtained from page 2 of the document.



instruction	input	strategy	output	Does the instruction describe a valid task?	Is the input appropriate?	Is the strategy valid (if not empty)?	Is the output correct?
Design a daily schedule for a high school student balancing academics, sports, and extracurricular activities.	None	Allocate specific time blocks for academics, sports, and extracurriculars	3:30 PM - 5:00 PM: Sports practice (e.g., soccer, basketball) 5:15 PM - 6:00 PM: Extracurricular activity (e.g., music, art club) 6:15 PM - 7:00 PM: Dinner with family 7:15 PM - 9:00 PM: Homework and study time 9:15 PM - 10:00 PM: Relaxation or leisure activities 10:15 PM: Prepare for bed and sleep	Yes	Yes	Yes	Yes
Design a simple mobile app interface for a weather forecasting application.	None	Use a clean and intuitive layout for easy navigation	Home Screen: Display current weather with temperature, location, and condition icon. Forecast Screen: Show 7-day forecast with daily high/low temperatures and icons. Navigation Bar: Include tabs for Home, Forecast, Settings. Settings: Options for unit conversion (Celsius/Fahrenheit), notification preferences. Color Scheme: Use calming	Yes	Yes	Yes	No