

Using an LLM to generate Instruction Following examples

In the module on [Instruction Following \(LLM_Instruction_Following.ipynb\)](#)

- we motivated the use of Fine-Tuning a LLM
- to exhibit Instruction Following behavior

Recall: an example of Instruction Following behavior is a triple

$\langle \text{Instruction}, \text{Context}, \text{Response} \rangle$

for example

- Instruction: "Tell me the word that is the opposite of the word that I input"
- Context: "Input: Stop"
- Response: "Go"

The Instruction describes the task to be accomplished

- relationship between Input and Response
- the Input/Response pair is an exemplar for this task

In this module, we explore methods

- to generate these fine-tuning examples
- to improve examples

Using an LLM to generate Instruction Following examples

Reference

[SELF-Instruct paper \(<https://arxiv.org/pdf/2212.10560.pdf>\)](https://arxiv.org/pdf/2212.10560.pdf)

Is there an alternative to the labor-intensity of constructing Instruction Following examples by human ?

These examples are pairs of an Instruction part, and a Target Output part.

The idea of the [SELF-Instruct paper](https://arxiv.org/pdf/2212.10560.pdf) (<https://arxiv.org/pdf/2212.10560.pdf>) is to create Synthetic examples

- using [In-Context Learning \(In_Context_Learning.ipynb\)](#)

We can imagine the process as

- starting with a small number k of human-constructed examples
 $\langle \text{Instruction}^{(1)}, \text{Context}^{(1)}, \text{Response}^{(1)} \rangle$
 \vdots
 $\langle \text{Instruction}^{(k)}, \text{Context}^{(k)}, \text{Response}^{(k)} \rangle$
- which are used as exemplars
- in a *few-shot* learning prompt
- that an LLM completes into a new Instruction Following example
 $\langle \text{Instruction}^{(k+1)}, \text{Context}^{(k+1)}, \text{Response}^{(k+1)} \rangle$

In actuality

- the Instruction part of the triple
 $\langle \text{Instruction}^{(k+1)}, \text{Context}^{(k+1)}, \text{Response}^{(k+1)} \rangle$
- is generated first
- using In-Context learning

Given the newly generated instruction

$$\text{Instruction}^{(k+1)}$$

- In-Context learning is used again
- to generate the associated Context and Response
 $\text{Context}^{(k+1)}, \text{Response}^{(k+1)}$

The process for generating each part of the triple

- is illustrated by a diagram
- with the steps described in detail below

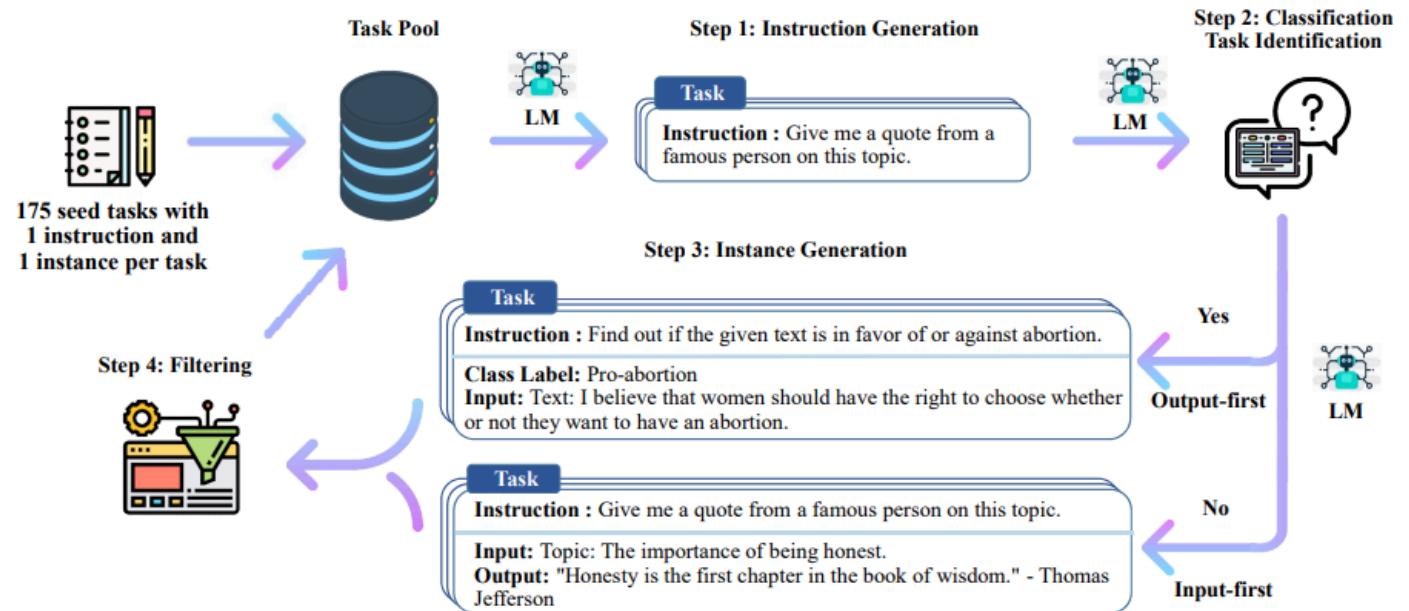


Figure 1: A high-level overview of SELF-INSTRUCT. The process starts with a small seed set of tasks (one instruction and one input-output instance for each task) as the task pool. Random tasks are sampled from the task pool, and used to prompt an off-the-shelf LM to generate both new instructions and corresponding instances, followed by filtering low-quality or similar generations, and then added back to the initial repository of tasks. The resulting data can be used for the instruction tuning of the language model itself later to follow instructions better. Tasks shown in the figure are generated by GPT3. See Table 10 for more creative examples.

Attribution: <https://arxiv.org/pdf/2212.10560.pdf#page=2>
[\(https://arxiv.org/pdf/2212.10560.pdf#page=2\)](https://arxiv.org/pdf/2212.10560.pdf#page=2)

Generating the Instruction part of an Instruction-Output example

The first step is to generate the first Instruction part (i.e., the Instruction) of the triple

$$\langle \mathbf{Instruction}^{(k+1)}, \mathbf{Context}^{(k+1)}, \mathbf{Response}^{(k+1)} \rangle$$

using k exemplars

$$\langle \mathbf{Instruction}^{(1)} \rangle$$

⋮

$$\langle \mathbf{Instruction}^{(k)} \rangle$$

See the box labeled "Step 1" in the illustration above

The precise template for the exemplars is show below.

B Prompting Templates for Data Generation

SELF-INSTRUCT relies on a number of prompting templates in order to elicit the generation from language models. Here we provide our four templates for generating the instruction (Table 6), classifying whether an instruction represents a classification task or not (Table 7), generating non-classification instances with the input-first approach (Table 8), and generating classification instances with the output-first approach (Table 9).

```
Come up with a series of tasks:  
  
Task 1: {instruction for existing task 1}  
Task 2: {instruction for existing task 2}  
Task 3: {instruction for existing task 3}  
Task 4: {instruction for existing task 4}  
Task 5: {instruction for existing task 5}  
Task 6: {instruction for existing task 6}  
Task 7: {instruction for existing task 7}  
Task 8: {instruction for existing task 8}  
Task 9:
```

Table 6: Prompt used for generating new instructions. 8 existing instructions are randomly sampled from the task pool for in-context demonstration. The model is allowed to generate instructions for new tasks, until it stops its generation, reaches its length limit or generates “Task 16” tokens.

Attribution: [\(https://arxiv.org/pdf/2212.10560.pdf#page=15\)](https://arxiv.org/pdf/2212.10560.pdf#page=15)

With the above template, we expect the LLM

- to generate a continuation of the prompt
 - ending with Task 9:
- which is the Instruction part of a new task

Generating the Context/Responses (Input/Output) part, given an Instruction

Once we have generated the Instruction part

$$\text{Instruction}^{(k+1)}$$

of the new synthetic example, we need to generate the Context and Response

- using exemplars

$$\langle \text{Instruction}^{(1)}, \text{Context}^{(1)}, \text{Response}^{(1)} \rangle$$

⋮

$$\langle \text{Instruction}^{(k)}, \text{Context}^{(k)}, \text{Response}^{(k)} \rangle$$

and prompt

$$\text{Instruction}^{(k+1)}$$

with the expectation that the LLM's continuation will be

$$\text{Context}^{(k+1)}, \text{Response}^{(k+1)}$$

See the box labeled "Step 3" in the diagram above

For Classification tasks, the prompt might look like this

Task: Classify the sentiment of the sentence into positive, negative, or mixed

Example 1

Sentence: I enjoy the flavor of the restaurant but their service is too slow.

Class Label: mixed

Example 2

Sentence: I had a great day today. The weather was beautiful and I spent time with friends.

Class label: Positive

Task: Tell me if the following email is a promotion email or not.

Email: Check out our amazing new sale! We've got discounts on all of your favorite products.

Class label: Promotion

Email: We hope you are doing well. Let us know if you need any help.

Class label: Not Promotion

The last line above contains a place holder for the Instruction of the Target Task
 $\text{Instruction}^{(k+1)}$
that we created in Step 1.

Here is an example of the template from the paper

Task: Suggest a better and more professional rephrasing of the following sentence.

Example 1

Sentence: This house is surprisingly not constructed very well, and you probably need more money to fix it after you buy it. If you ask me, I would suggest you to consider other candidates.

Output: This house does not seem to be constructed well, so you may need to spend more money to fix it after you purchase it. I would suggest that you look at other properties.

Example 2

Sentence: Just so you know, we did an experiment last week and found really surprising results - language model can improve itself!

Output: Our experiments last week demonstrated surprising results, proving that the language model can improve itself.

...

Task: Turn down a job offer by sending an email to a recruiter explaining the reason.

Output: Hi [Recruiter],

Thank you so much for the generous offer to join your team. As we discussed, I've admired the company for a number of years, and am a proud endorser of its products. However, after further consideration of where I currently am in my career, I've decided to accept an offer at another company.

I would love to stay in touch with you and have already started following you on [Social Media Platform]. Again, thank you so much for your time and consideration.

Thanks again,

[Your Name]

Task: {Instruction for the target task}

Table 8: Prompt used for the input-first approach of instance generation. The model is prompted to generate the instance first, and then generate the corresponding output. For instructions that don't require additional input, the output is allowed to be generated directly.

Attribution: <https://arxiv.org/pdf/2212.10560.pdf#page=16>
(<https://arxiv.org/pdf/2212.10560.pdf#page=16>)

Difficulties in Generating the Input/Output part: Classification tasks

Although the few-shot learning approach to generating an Input/Output given an Instruction

- seems straightforward
- the authors encountered difficulties when generating Input/Output for Classification tasks

Consider the an Instruction Following example for a Classification task

Task: Classify the sentiment of the sentence into positive, negative, or mixed

Example 1

Sentence: I enjoy the flavor of the restaurant but their service is too slow.

Class Label: mixed

The authors found that the response generated by the LLM (e.g., Classification examples)

- were examples whose Class Label's
- were *not well-distributed* among all possible labels
 - examples with certain labels were either over or under represented

This was attributed to the *format* of the example called *Input-first* format
 $\langle \text{Instruction}^{(i)}, \text{Context}^{(i)}, \text{Response}^{(i)} \rangle$

where

- $\text{Context}^{(i)}$ is the Additional Input
- $\text{Response}^{(i)}$ is the Class label

The solution was to change the example format to *output-first*
 $\langle \text{Instruction}^{(i)}, \text{Response}^{(i)}, \text{Context}^{(i)} \rangle$

where the Response (Class label) precedes the Context (Additional Input)

For example:

Task: Classify the sentiment of the sentence into positive, negative, or mixed

Example 1

Class Label: mixed

Sentence: I enjoy the flavor of the restaurant but their service is too slow.

Example 2

Class label: Positive

Sentence: I had a great day today. The weather was beautiful and I spent time with friends.

This is an example of Prompt Engineering

- In-context learning seems very sensitive to the format of prompts
- There is a skill of engineering a prompt to elicit the desired behavior

This feels similar to the idea behind Chain of Thought prompting

- by presenting Class Label first
- the model seems better conditioned to generate a less biased distribution of labels

Generating Instructions via Backtranslation

We now present an alternate method for generating the Instruction part of
 $\langle \text{Instruction} \rangle$

of an Instruction Following example

$\langle \text{Instruction}, \text{Context}, \text{Response} \rangle$

The idea seems, at first, to be backwards:

- Generate an instance of Instruction Following behavior
 $\langle \text{\textbackslash x}, \text{\textbackslash y} \rangle = \langle \text{Instruction}, \text{Response} \rangle$
- by starting with a Response
- and using an LLM to create the Instruction

Essentially

- we start with *un-labeled data* (Response)
- and create a *label* (Instruction)

The advantage of this approach is that

- un-labeled data is plentiful
 - almost any block of text
- but labeled data (Response/Instruction pairs) is scarce.

So, starting with a plentiful resource, we create the scarce resource

- i.e, Instruction Following example triplet

The method is called *Back Translation*.

- given a *small* "seed" of Instruction/Response pairs
 $\langle \backslash x, \backslash y \rangle = \langle \text{Instruction}, \text{Response} \rangle$
- create an inverse dataset of Response/Instruction pairs by reversing the features and targets
 $\langle \backslash y, \backslash x \rangle = \langle \text{Response}, \text{Instruction} \rangle$

We use the inverse dataset to train a model M_{yz}

- given a Response ($\backslash y$)
- to create an Instruction ($\backslash z = \hat{\backslash x}$)

by fine-tuning an LLM to predict Instruction from Response

M_{yz}

- when fed by set of Responses $\{y_i\}$
 - the Unlabeled Data in the diagram
- creates new instances of Instruction/Response examples

This results in an enlarged set of Instruction/Response examples

- the original human generated examples
- augmented by the Synthetic examples created by M_{yz}

in a process called *Self Augmentation*

With the newly extended set of seed Instruction/Response pairs

- we have more exemplars
- which we can use as a seed to another iteration of M_{yz}
 - the enlarged set of exemplars may result in *better* synthetic Instruction/Response pairs

We can iterate on this process multiple times

- using the Augmented set of Instruction/Response pairs from step i
- as the "seed" for iteration $(i + 1)$ of the process

Here is the workflow:

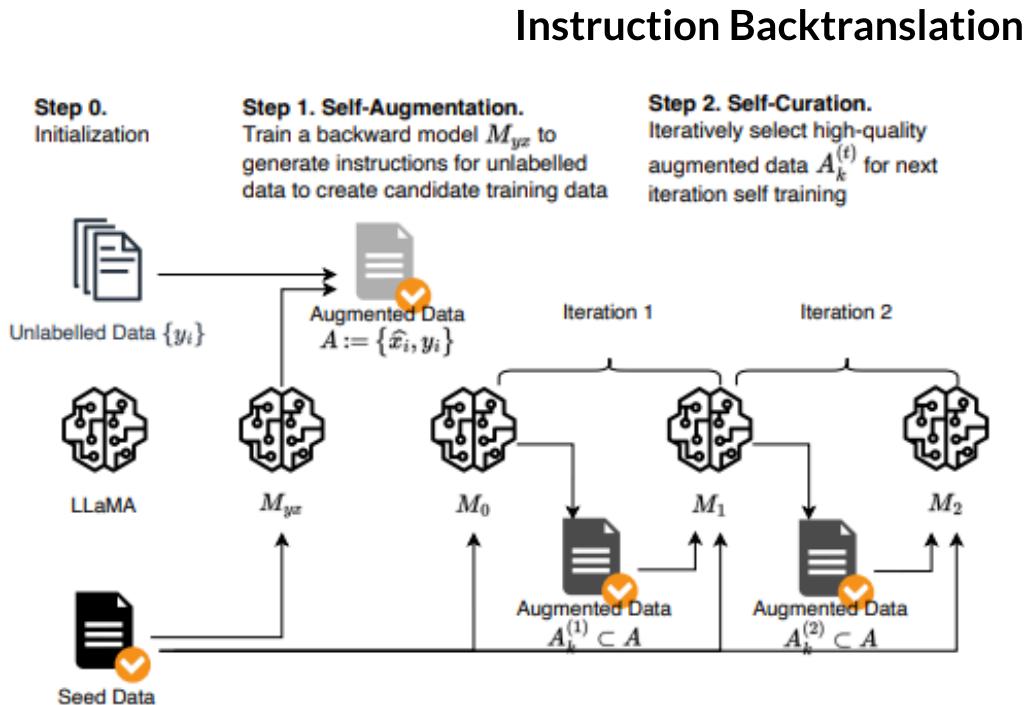


Figure 1: An overview of our **instruction backtranslation** method. We start from a base language model, e.g. LLaMa, a small amount of seed examples of (instruction, output) pairs, and a collection of unlabelled documents which are considered candidate outputs for unknown instructions. **Self-augmentation:** the base model is finetuned with (output, instruction) pairs from the seed examples as an instruction prediction model M_{yx} , which is used to generate candidate instructions for outputs from the unlabelled data. **Self-curation:** starting from an intermediate instruction-following model M_0 finetuned from seed examples only, it selects high-quality (instruction, output) pairs $A_k^{(1)}$ from the candidates from the previous step, and uses them as finetuning data for the next intermediate model M_1 , which is in turn used to select training data for obtaining M_2 .

Attribution: <https://arxiv.org/pdf/2308.06259.pdf#page=2>

Selecting the best synthetic examples for augmentation

The quality of the synthetic examples created at each step may not be uniformly high.

It would be desirable

- to select only the best examples to use
- in augmenting the seed examples of each iterative Step.

How can we rate the quality of a synthetic example ?

Ask the LLM to do it for you !

Using just the seed data

- fine tune a "first generation" LLM
 - denoted M_0
- to create a quality score of examples

The following prompt requests that the LLM evaluate the synthetic example using a rating scale of 1 (low quality) to 5 (high quality)

Instruction Backtranslation Curation

Below is an instruction from an user and a candidate answer. Evaluate whether or not the answer is a good example of how AI Assistant should respond to the user's instruction. Please assign a score using the following 5-point scale:

1: It means the answer is incomplete, vague, off-topic, controversial, or not exactly what the user asked for. For example, some content seems missing, numbered list does not start from the beginning, the opening sentence repeats user's question. Or the response is from another person's perspective with their personal experience (e.g. taken from blog posts), or looks like an answer from a forum. Or it contains promotional text, navigation text, or other irrelevant information.

2: It means the answer addresses most of the asks from the user. It does not directly address the user's question. For example, it only provides a high-level methodology instead of the exact solution to user's question.

3: It means the answer is helpful but not written by an AI Assistant. It addresses all the basic asks from the user. It is complete and self contained with the drawback that the response is not written from an AI assistant's perspective, but from other people's perspective. The content looks like an excerpt from a blog post, web page, or web search results. For example, it contains personal experience or opinion, mentions comments section, or share on social media, etc.

4: It means the answer is written from an AI assistant's perspective with a clear focus of addressing the instruction. It provide a complete, clear, and comprehensive response to user's question or instruction without missing or irrelevant information. It is well organized, self-contained, and written in a helpful tone. It has minor room for improvement, e.g. more concise and focused.

5: It means it is a perfect answer from an AI Assistant. It has a clear focus on being a helpful AI Assistant, where the response looks like intentionally written to address the user's question or instruction without any irrelevant sentences. The answer provides high quality content, demonstrating expert knowledge in the area, is very well written, logical, easy-to-follow, engaging and insightful.

Please first provide a brief reasoning you used to derive the rating score, and then write "Score: <rating>" in the last line.

```
<generated instruction>
<output>
```

Table 1: Prompt used in the *self-curation* step to evaluate the quality of a candidate (instruction, output) pair in the dataset derived from self-augmentation.

Use M_0 to

- select the best first generation augmented examples (from the first iteration)

The next generation augmented data set is

- the prior generation
- augmented with the best (highest quality scores) of the new generation

Now that we have

- an augmented (high quality) "generation i " set of seed examples

we continue our iterative process

- creating a more powerful scoring LLM M_i
- to create an augmented high quality "generation $i + 1$ " set of examples

This is an example of [LLM Self-Improvement \(LLM_Self_Improvement.ipynb\)](#).

In [2]: `print("Done")`

Done

