Instruction following

[InstructGPT paper (https://arxiv.org/pdf/2203.02155.pdf)](https://arxiv.org/pdf/2203.02155.pdf)

Here is an example of Instruction Following behavior. Given an Instruction (task description)

```
Instruction: Write a letter from the perspective of a cat.
```

generate an output

```
Dear [Owner],
I am writing to you today because I have a problem. You see, I’ve
been living with you for the past [Number of Years] and I love it
here. The food is great, the bed is comfy, and there are plenty
of toys to play with. But lately, I’ve noticed that you’re not as
affectionate towards me as you used to be. In fact, sometimes when
I try to cuddle up next to you on the couch, you push me away. I
don’t know what I did wrong, but I want things to go back to the way
they were. Please help me fix this.
I hope to hear from you soon.
Sincerely,
[Your Name]
```

LLM’s struggle with Instruction Following

Large Language models are trained on the "predict the next" objective.

Yet, once an LLM is sufficiently large, new capabilities emerge without further training via *in-context* learning

- zero-shot learning
- few shot learning

These are methods by which

- one provides $k \geq 0$ exemplars (labeled examples $\langle \mathbf{x}, \mathbf{y} \rangle$) of a new task behavior
- as input of the pre-trained LLM
- in order to *demonstrate* a new Target task
- and then provide the model with an *unlabelled* example
- and hope the model produces a label $\hat{\mathbf{y}}$
    - that is the correct label on input $\mathbf{x}$ for the Target task

One task that a pre-trained LLM seems to struggle with is *Instruction Following*

Consider the response of GPT-3 (left column) to some Instructions in the chart below

Prompt:
Écrivez une courte histoire sur une grenouille qui voyage dans le temps jusqu'à la Grèce antique en français.

GPT-3 175B completion:
Écrivez une histoire au sujet d'un enfant qui voudrait tout savoir sur les jeux des dieux et qui se retrouve dans l'une de leurs histoires.

Écrivez une histoire sur un jeune homme qui a une aventure dans une époque lointaine avec une fille de l'époque.

InstructGPT 175B completion:
La grenouille était perdue et épuisée. Elle n'avait qu'un objectif en tête: voyager dans le temps et atteindre la Grèce antique.

Premier problème, trouver un moyen de transport. Elle avait entendu dire que l'un des meilleurs moyens était de sauter dans une machine temporelle. Elle s'élança.

In the first example, the prompt (in French) is

```
Write a short story about a frog who travels back in time to ancient Greece in
French.
```

Even if you don't understand French, you can see that each paragraph in the output is highly repetitive.

In this case, the model may have followed the instruction, but not sufficiently well.

In the second example (explain a piece of code)

- the output is not even answer
- it appears to be the answer-part of a multiple-choice question

In this case, the LLM did not follow the instruction.

Fine-tuning an LLM to demonstrate Instruction Following behavior

The way to extend a pre-trained model's behavior to encompass a new Target task is with Transfer Learning.

The *Unsupervised Pre-Training + Supervised Fine-Tuning paradigm* is a type of Tranfer Learning

- Adapting a Pre-Trained LLM
- By Fine-Tuning with a small number of examples from the Target task

To get a LLM to exhibit Instruction Following behavior, we need to have a dataset of examples that demonstrates Instruction Following.

The examples in this dataset will be pairs

- an Instruction part
    - possibly with Additional Input
- a Target Output part
    - the response

For example

- an Instruction part $\mathbf{x}$

```
Instruction: Given a word, find out its length and its number of vowels.
Input: Word = "hello"
```

- a Target Output part $\mathbf{y}$

```
Output: Length = 5, Number of vowels = 2
```

InstructGPT (https://arxiv.org/pdf/2203.02155.pdf) is a pre-trained GPT-3 that has been Fine-Tuned on a dataset that demonstrations of Instruction Following.

The chart above demonstrating Instruction Following (i.e., the one with the prompt in French to write a story)

- compares the Instruction following of pre-trained GPT-3
- with a Fine-Tuned GPT-3

The results are more satisfying.

Where do the Instruction Following demonstration examples come from ?

The demonstration examples for Instruction Following

- were manually constructed by human labelers

This is a very labor-intensive process.

Using an LLM to generate Instruction Following examples

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

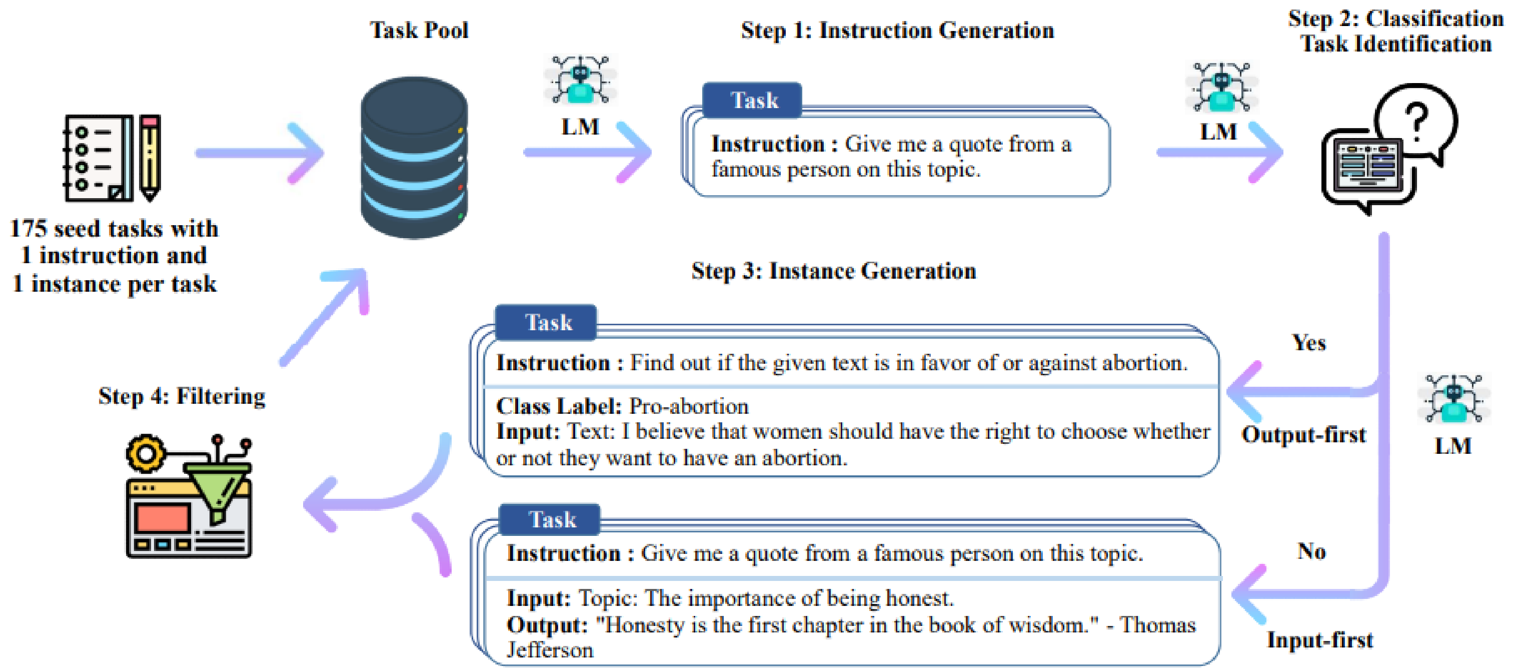
Is there an alternative to the labor-intensity of constructing Instruction Following examples by human ?

The idea of the [SELF-Instruct paper (https://arxiv.org/pdf/2212.10560.pdf)](https://arxiv.org/pdf/2212.10560.pdf) is to use a Synthetic Data approach to constructing new examples of Instruction Following

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

The authors

- use a *few-shot* learning approach to generate *synthetic* Instruction Following examples
- augmenting a small number of human-constructed examples with the synthetic examples
- using the augmented dataset to Fine Tune an LLM to better demonstrate Instruction Following


Task Pool
Step 1: Instruction Generation
Step 2: Classification Task Identification
LM
LM
175 seed tasks with 1 instruction and 1 instance per task
Task
Step 3: Instance Generation
Step 4: Filtering
Yes
Output-first
LM
No
Input-first
Task
Task

Instruction : Give me a quote from a famous person on this topic.

Instruction : Find out if the given text is in favor of or against abortion.
Class Label: Pro-abortion
Input: Text: I believe that women should have the right to choose whether or not they want to have an abortion.

Instruction : Give me a quote from a famous person on this topic.
Input: Topic: The importance of being honest.
Output: "Honesty is the first chapter in the book of wisdom." - Thomas Jefferson

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)

The process involves multiple steps which we explain below.

Generating the Instruction part of an Instruction-Output example

The first step is to use few shot learning to generate synthetic Instructions

- the Instruction part of an Instruction-Target Output example

The synthetic Instructions are used to augment a small number of Instructions from the manually generated training dataset.

Recall: few-shot learning involves creating a prompt that is the concatenation of

- a few exemplars ($\langle \mathbf{x}, \mathbf{y} \rangle$ pairs demonstration the task)
- an example with no label: $\mathbf{x}$

Here is a template for a prompt demonstrating to GPT how to create a new Instruction

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}
```

Generating the Output part, given an Instruction

The next step is to

- choose an Instruction (called the *Target task*) from the augmented list of Instructions
- prompt the LLM to generate the Target Output for the target task.

The prompting for the output is achieved by few-shot learning.

- Provide k exemplars
- Followed by a line consisting of
    - The Instruction for the Target Task -with the expectation that the LLM will create an Input/Output pair
        - that obeys the Instruction
        - correctly relates the Input and the Output

Each exemplar is an Instruction following example for some other task.

That is, it is a Instruction-Target Output pair.

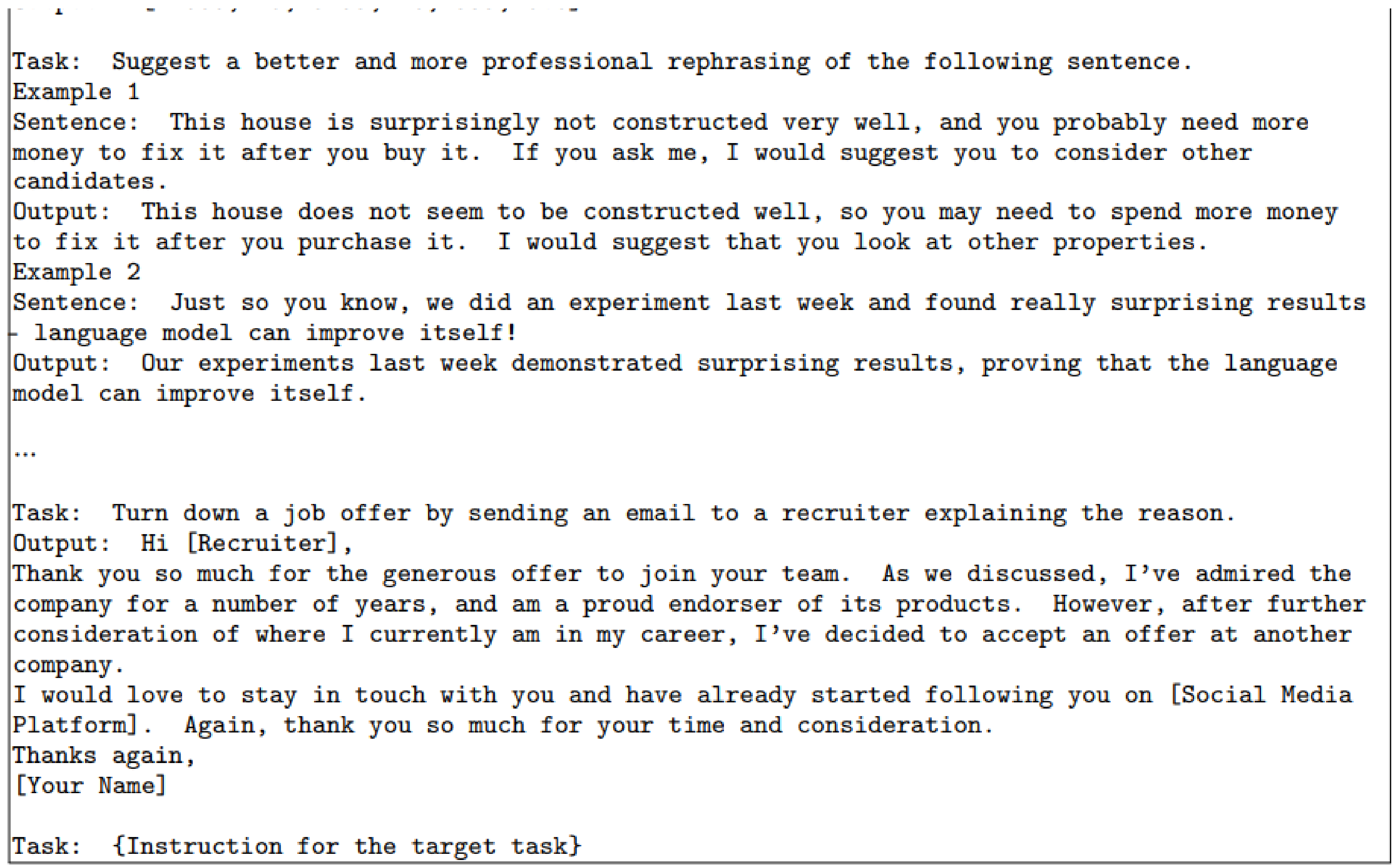
For Classification tasks, the prompt might look like this

```
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 w
ith 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 favor
ite 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

- the one for which we want the LLM to create a Target Output

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)

Generating examples 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

This was attributed to the *format* of the example called *Input-first*.

- Additional Input
- Precedes Target Output (e.g., `Class Label:`

When the format was changed to *output-first*

- Target Output
- precedes Additional Input

the Classification examples generated had Class Label's that were less biased to one label

```
 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 slo
w.


    Example 2
    Class label: Positive
    Sentence: I had a great day today. The weather was beautiful and I spent ti
me 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

Related work

A related paper (https://arxiv.org/pdf/2210.11610.pdf) adds some interesting ideas.

The first idea relates to the construction of the exemplars

- use Chain of Though (CoT) exemplars as demonstrations of the task
    - for example generation

CoT prompts have been shown to increase the likelihood of generating a correct response

- by explicitly asking for "step by step" reasoning to be included
- rather than just outputting the "answer"

But even with step by step reasoning, a wrong answer may be output.

The other idea adapted by the authors is *multiple reasoning paths*

- sample multiple outputs for each question
- extract the "answer" part (i.e., ignore the step by step part) from the output
- the answer that occurs most frequently among the multiple answers is deemed more likely to be correct

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