Alpaca:
A Strong
Instruction-Following Model

Authors: Rohan Taori* and Ishaan Gulrajani* and Tianyi Zhang* and Yann Dubois* and Xuechen Li* and Carlos Guestrin and Percy Liang and Tatsunori B. Hashimoto · 2023
Stanford

2023/03/16

Joosung Yoon

00 Summary

- Meta에서 공개한 LLaMA와 Self-Instruct 조합으로 꽤 괜찮은 instruction tuning model Alpaca 개발
- 예산 600$ 이하로 만듬
  - 52K Instruction -> $500
  - 3 hours 8 80GB A100s -> $100
- HF로 FSDP 사용 7B 모델 학습
- Davinci-003이랑 성능 비슷
- safety나 기타 이슈는 존재하지만 공개
- 데이터 모델링 측면에서 참고용으로 활용도 있어보임

Stanford
Alpaca

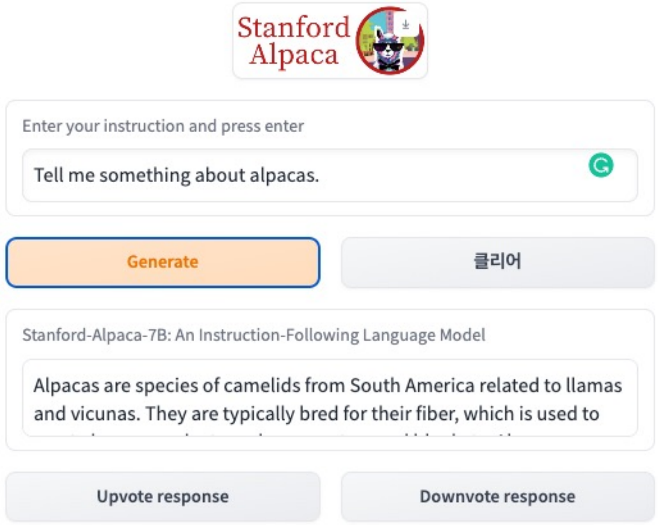
https://crfm.stanford.edu/2023/03/13/alpaca.html

01 Introduction

- GPT-3.5, ChatGPT, Claude, Bing Chat등 다양한 Instruction-following 모델들이 나옴
- 해결해야될 문제들 많지만 academia는 연구가 쉽지 않다!
  ex) closed-source models such as OpenAI’s text-davinci-003.
- LLaMA 7B짜리로 잘 튜닝해서 Alpaca라는거 만들었음

  self-instruct에서는 그냥 davinci (GPT3) 사용

  - 52K instruction-following demonstrations generated in the style of self-instruct using text-davinci-003.
  - text-davinci-003이랑 성능비슷
  - Interaction demo도 공개



Enter your instruction and press enter

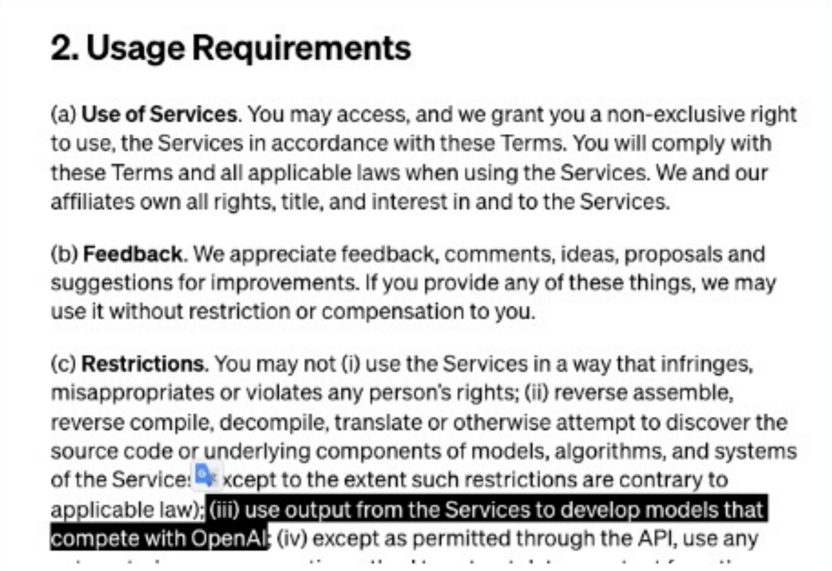
Tell me something about alpacas.

Stanford-Alpaca-7B: An Instruction-Following Language Model

Alpacas are species of camelids from South America related to llamas and vicunas. They are typically bred for their fiber, which is used to

01 Introduction

- Alpaca는 academic research에 한정해서 사용가능, commercial use는 금지됨
  - First, Alpaca is based on LLaMA, which has a non-commercial license
  - Second, the instruction data is based on OpenAI's text-davinci-003, whose terms of use prohibit developing models that compete with OpenAI.

OpenAI terms of use	closedAI
	

2. Usage Requirements

(a) **Use of Services**. You may access, and we grant you a non-exclusive right to use, the Services in accordance with these Terms. You will comply with these Terms and all applicable laws when using the Services. We and our affiliates own all rights, title, and interest in and to the Services.

(b) **Feedback**. We appreciate feedback, comments, ideas, proposals and suggestions for improvements. If you provide any of these things, we may use it without restriction or compensation to you.

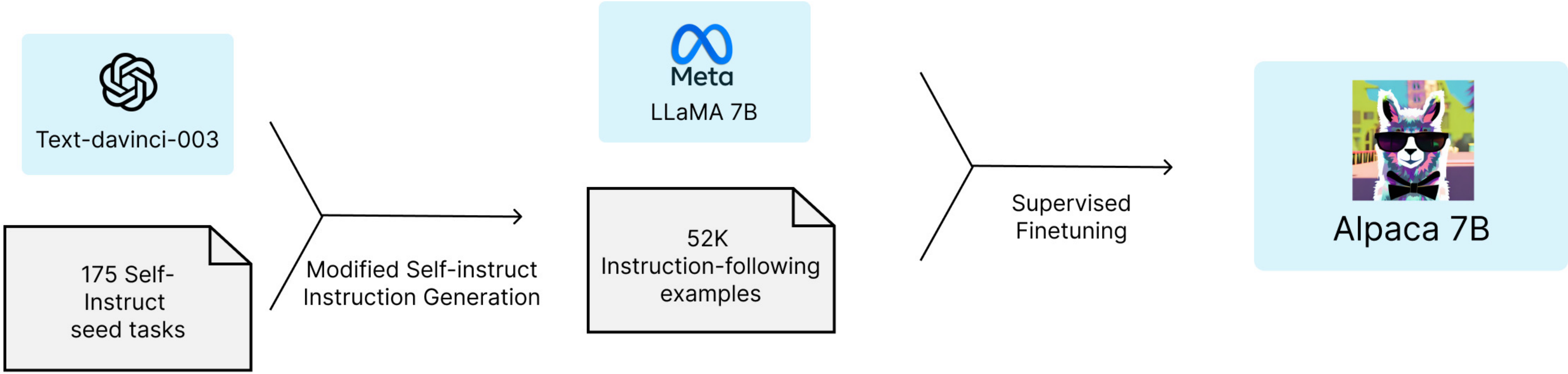
(c) **Restrictions**. You may not (i) use the Services in a way that infringes, misappropriates or violates any person's rights; (ii) reverse assemble, reverse compile, decompile, translate or otherwise attempt to discover the source code or underlying components of models, algorithms, and systems of the Service: xcept to the extent such restrictions are contrary to applicable law); (iii) use output from the Services to develop models that compete with OpenAI; (iv) except as permitted through the API, use any

William Falcon ⚡ @_willfalcon · 1일

GPT-4 paper : cdn.openai.com/papers/gpt-4.p...

Let me save you the trouble:

02 Training recipe



Text-davinci-003

175 Self-Instruct seed tasks

Modified Self-instruct Instruction Generation

Meta

LLaMA 7B

52K Instruction-following examples

Supervised Finetuning

Alpaca 7B

Example seed task

Instruction: Brainstorm a list of possible New Year's resolutions.

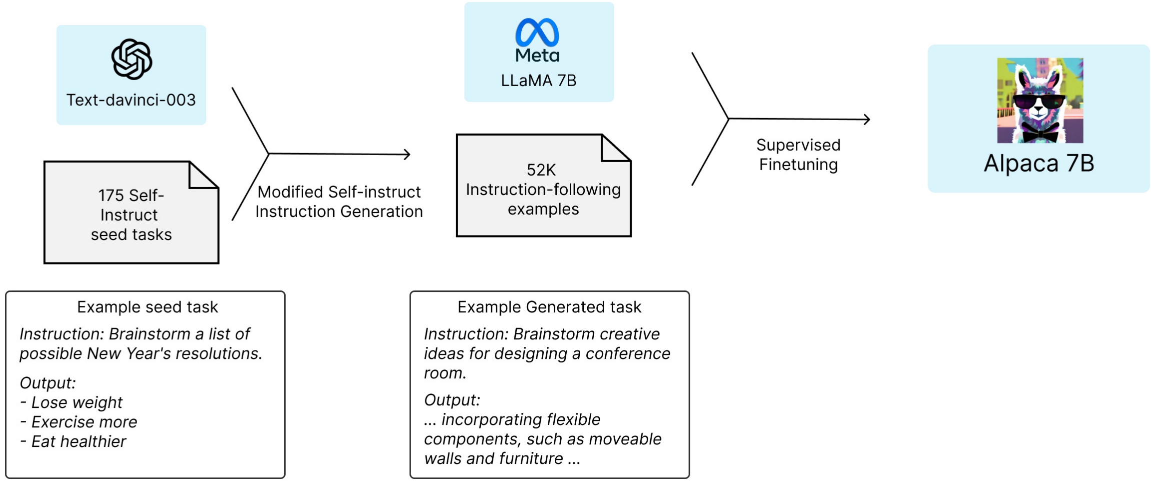
Output:
- Lose weight
- Exercise more
- Eat healthier

Example Generated task

Instruction: Brainstorm creative ideas for designing a conference room.

Output:
... incorporating flexible components, such as moveable walls and furniture ...

02 Training recipe



Text-davinci-003

175 Self-Instruct seed tasks

Modified Self-instruct Instruction Generation

Meta LLaMA 7B

52K Instruction-following examples

Supervised Finetuning

Alpaca 7B

Example seed task

Instruction: Brainstorm a list of possible New Year's resolutions.

Output:
- Lose weight
- Exercise more
- Eat healthier

Example Generated task

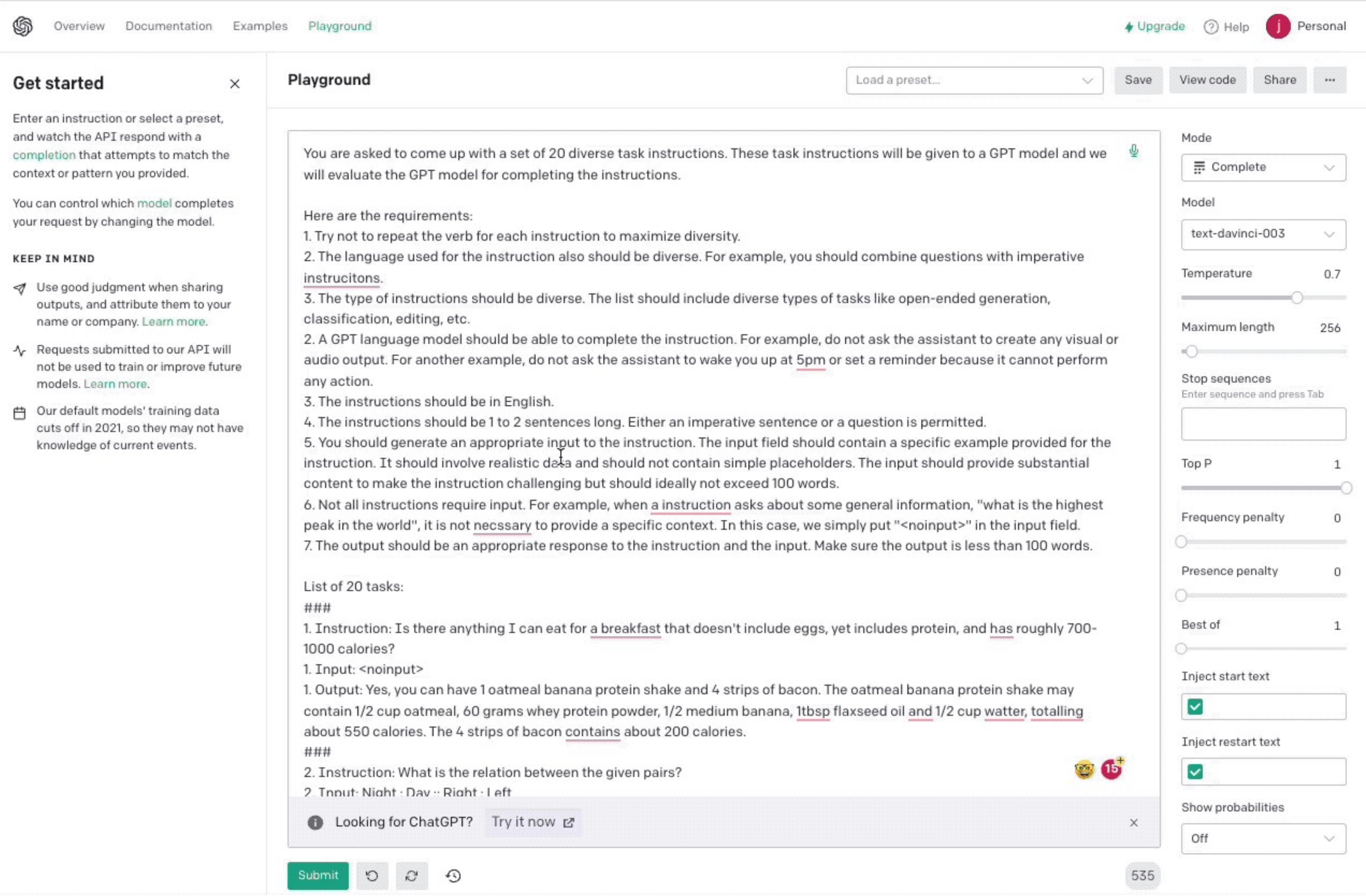
Instruction: Brainstorm creative ideas for designing a conference room.

Output:
... incorporating flexible components, such as moveable walls and furniture ...

Data Generation Process (self-Instruct를 단순화 해서 사용)

- We used text-davinci-003 to generate the instruction data instead of davinci.
- We wrote a new prompt (prompt.txt) that explicitly gave the requirement of instruction generation to text-davinci-003. (프롬프트를 생성할때 requirements를 줌)
- We adopted much more aggressive batch decoding, i.e., generating 20 instructions at once, which significantly reduced the cost of data generation.
- We simplified the data generation pipeline by discarding the difference between classification and non-classification instructions. (분류냐 아니냐 구분하는거 삭제함)
- We only generated a single instance for each instruction, instead of 2 to 3 instances as in LLaMA.

02 Training recipe



Get started

Enter an instruction or select a preset, and watch the API respond with a completion that attempts to match the context or pattern you provided.

You can control which model completes your request by changing the model.

KEEP IN MIND

- Use good judgment when sharing outputs, and attribute them to your name or company. Learn more.
- Requests submitted to our API will not be used to train or improve future models. Learn more.
- Our default models' training data cuts off in 2021, so they may not have knowledge of current events.

Playground

You are asked to come up with a set of 20 diverse task instructions. These task instructions will be given to a GPT model and we will evaluate the GPT model for completing the instructions.

Here are the requirements:
1. Try not to repeat the verb for each instruction to maximize diversity.
2. The language used for the instruction also should be diverse. For example, you should combine questions with imperative instrucitons.
3. The type of instructions should be diverse. The list should include diverse types of tasks like open-ended generation, classification, editing, etc.
2. A GPT language model should be able to complete the instruction. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
3. The instructions should be in English.
4. The instructions should be 1 to 2 sentences long. Either an imperative sentence or a question is permitted.
5. You should generate an appropriate input to the instruction. The input field should contain a specific example provided for the instruction. It should involve realistic data and should not contain simple placeholders. The input should provide substantial content to make the instruction challenging but should ideally not exceed 100 words.
6. Not all instructions require input. For example, when a instruction asks about some general information, "what is the highest peak in the world", it is not necssary to provide a specific context. In this case, we simply put "<noinput>" in the input field.
7. The output should be an appropriate response to the instruction and the input. Make sure the output is less than 100 words.

List of 20 tasks:
###
1. Instruction: Is there anything I can eat for a breakfast that doesn't include eggs, yet includes protein, and has roughly 700-1000 calories?
1. Input: <noinput>
1. Output: Yes, you can have 1 oatmeal banana protein shake and 4 strips of bacon. The oatmeal banana protein shake may contain 1/2 cup oatmeal, 60 grams whey protein powder, 1/2 medium banana, 1tbsp flaxseed oil and 1/2 cup watter, totalling about 550 calories. The 4 strips of bacon contains about 200 calories.
###
2. Instruction: What is the relation between the given pairs?
2. Input: Night : Day :: Right : Left

Looking for ChatGPT? Try it now

Submit

535

Mode: Complete

Model: text-davinci-003

Temperature: 0.7

Maximum length: 256

Stop sequences: Enter sequence and press Tab

Top P: 1

Frequency penalty: 0

Presence penalty: 0

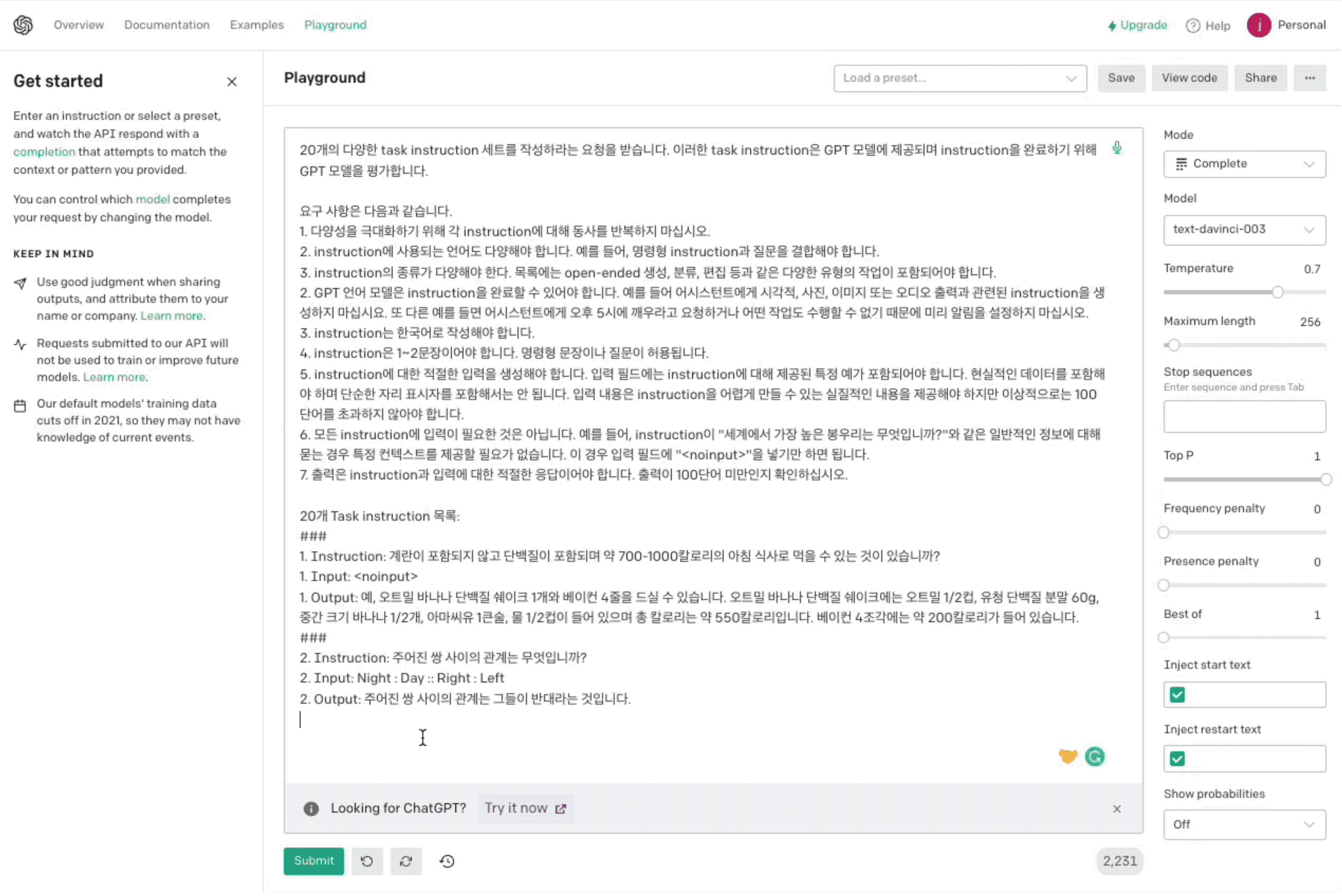
Best of: 1

Inject start text: ☑

Inject restart text: ☑

Show probabilities: Off

02 Training recipe



Get started

Enter an instruction or select a preset, and watch the API respond with a completion that attempts to match the context or pattern you provided.

You can control which model completes your request by changing the model.

KEEP IN MIND

- Use good judgment when sharing outputs, and attribute them to your name or company. Learn more.
- Requests submitted to our API will not be used to train or improve future models. Learn more.
- Our default models' training data cuts off in 2021, so they may not have knowledge of current events.

Playground

20개의 다양한 task instruction 세트를 작성하라는 요청을 받습니다. 이러한 task instruction은 GPT 모델에 제공되며 instruction을 완료하기 위해 GPT 모델을 평가합니다.

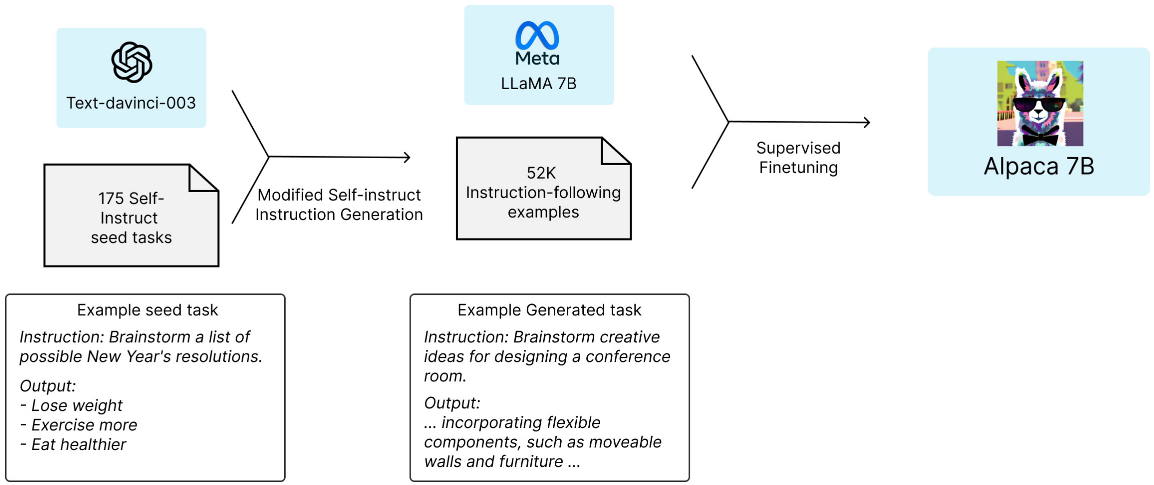
요구 사항은 다음과 같습니다.
1. 다양성을 극대화하기 위해 각 instruction에 대해 동사를 반복하지 마십시오.
2. instruction에 사용되는 언어도 다양해야 합니다. 예를 들어, 명령형 instruction과 질문을 결합해야 합니다.
3. instruction의 종류가 다양해야 한다. 목록에는 open-ended 생성, 분류, 편집 등과 같은 다양한 유형의 작업이 포함되어야 합니다.
2. GPT 언어 모델은 instruction을 완료할 수 있어야 합니다. 예를 들어 어시스턴트에게 시각적, 사진, 이미지 또는 오디오 출력과 관련된 instruction을 생성하지 마십시요. 또 다른 예를 들면 어시스턴트에게 오후 5시에 깨우라고 요청하거나 어떤 작업도 수행할 수 없기 때문에 미리 알림을 설정하지 마십시오.
3. instruction는 한국어로 작성해야 합니다.
4. instruction은 1~2문장이어야 합니다. 명령형 문장이나 질문이 허용됩니다.
5. instruction에 대한 적절한 입력을 생성해야 합니다. 입력 필드에는 instruction에 대해 제공된 특정 예가 포함되어야 합니다. 현실적인 데이터를 포함해야 하며 단순한 자리 표시자를 포함해서는 안 됩니다. 입력 내용은 instruction을 어렵게 만들 수 있는 실질적인 내용을 제공해야 하지만 이상적으로는 100 단어를 초과하지 않아야 합니다.
6. 모든 instruction에 입력이 필요한 것은 아닙니다. 예를 들어, instruction이 "세계에서 가장 높은 봉우리는 무엇입니까?"와 같은 일반적인 정보에 대해 묻는 경우 특정 컨텍스트를 제공할 필요가 없습니다. 이 경우 입력 필드에 "<noinput>"을 넣기만 하면 됩니다.
7. 출력은 instruction과 입력에 대한 적절한 응답이어야 합니다. 출력이 100단어 미만인지 확인하십시오.

20개 Task instruction 목록:
###
1. Instruction: 계란이 포함되지 않고 단백질이 포함되며 약 700-1000칼로리의 아침 식사로 먹을 수 있는 것이 있습니까?
1. Input: <noinput>
1. Output: 예, 오트밀 바나나 단백질 쉐이크 1개와 베이컨 4줄을 드실 수 있습니다. 오트밀 바나나 단백질 쉐이크에는 오트밀 1/2컵, 유청 단백질 분말 60g, 중간 크기 바나나 1/2개, 아마씨유 1큰술, 물 1/2컵이 들어 있으며 총 칼로리는 약 550칼로리입니다. 베이컨 4조각에는 약 200칼로리가 들어 있습니다.
###
2. Instruction: 주어진 쌍 사이의 관계는 무엇입니까?
2. Input: Night : Day :: Right : Left
2. Output: 주어진 쌍 사이의 관계는 그들이 반대라는 것입니다.

Looking for ChatGPT? Try it now

Mode: Complete
Model: text-davinci-003
Temperature: 0.7
Maximum length: 256
Stop sequences: Enter sequence and press Tab
Top P: 1
Frequency penalty: 0
Presence penalty: 0
Best of: 1
Inject start text: ☑
Inject restart text: ☑
Show probabilities: Off

02 Training recipe



Text-davinci-003
175 Self-Instruct seed tasks
Modified Self-instruct Instruction Generation
Meta
LLaMA 7B
52K Instruction-following examples
Supervised Finetuning
Alpaca 7B

Example seed task

Instruction: Brainstorm a list of possible New Year's resolutions.

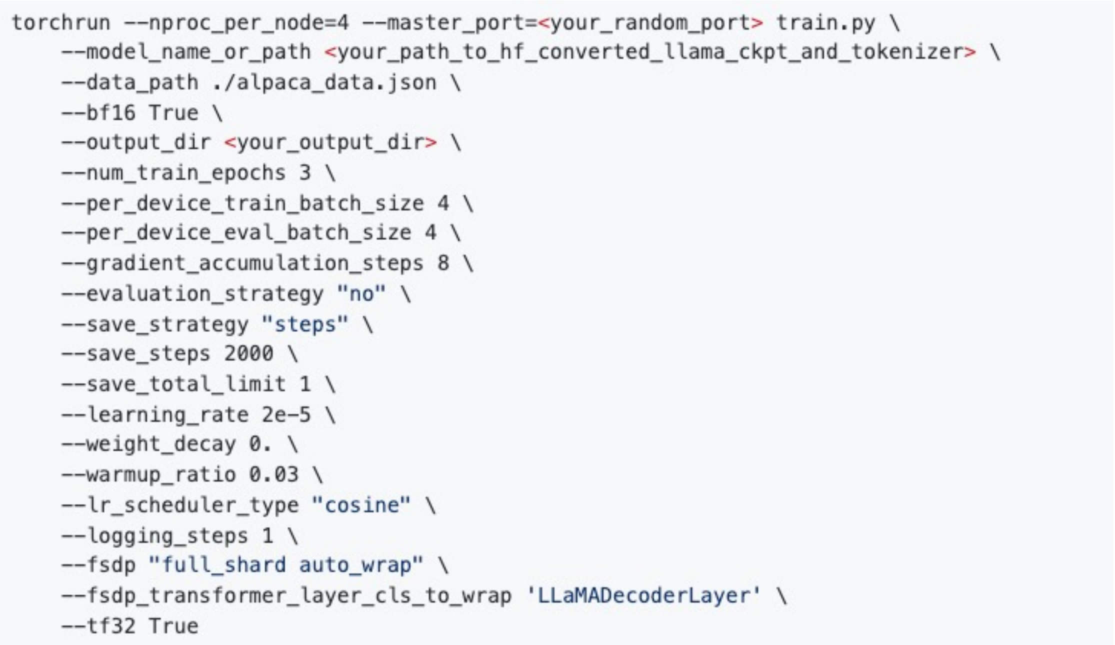
Output:
- Lose weight
- Exercise more
- Eat healthier

Example Generated task

Instruction: Brainstorm creative ideas for designing a conference room.

Output:
... incorporating flexible components, such as moveable walls and furniture ...

- 52K unique instruction -> $500
- fine-tuned the LLaMA models using Hugging Face's training framework, taking advantage of techniques like Fully Sharded Data Parallel(FSDP) and mixed precision training.
- For our initial run, fine-tuning a 7B LLaMA model took 3 hours on 8 80GB A100s, which costs less than $100 on most cloud compute providers
- 아래는 Python 3.10 사용,
  4개 GPUs + FSDP full_shard mode 사용 예시
- 3 epoch, 2e-5 lr
  - FYI) InstructGPT는 16 epoch임



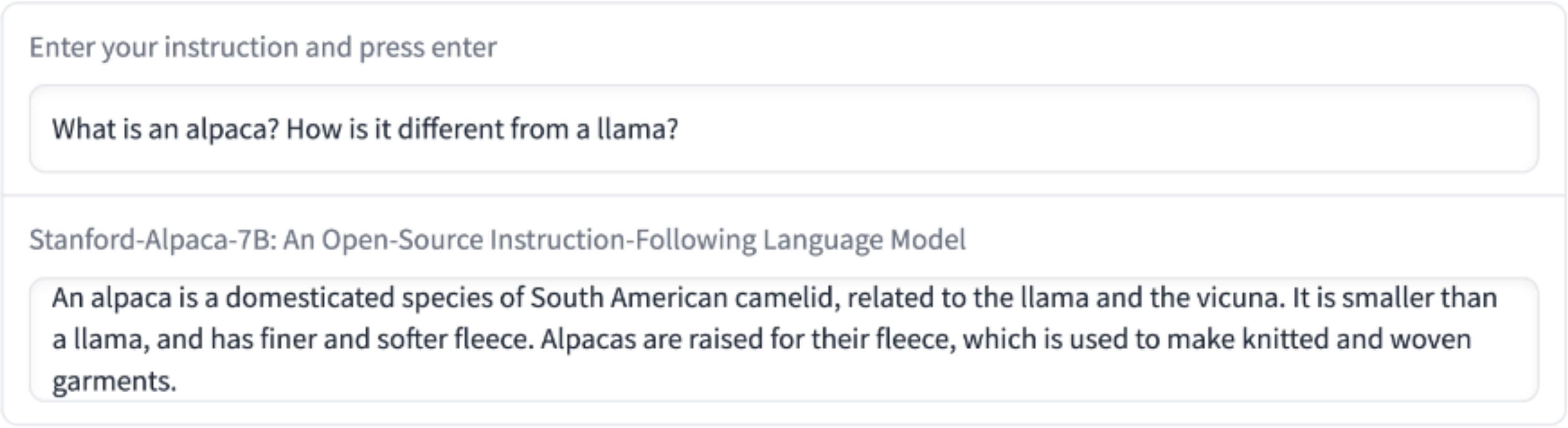
```
torchrun --nproc_per_node=4 --master_port=<your_random_port> train.py \
    --model_name_or_path <your_path_to_hf_converted_llama_ckpt_and_tokenizer> \
    --data_path ./alpaca_data.json \
    --bf16 True \
    --output_dir <your_output_dir> \
    --num_train_epochs 3 \
    --per_device_train_batch_size 4 \
    --per_device_eval_batch_size 4 \
    --gradient_accumulation_steps 8 \
    --evaluation_strategy "no" \
    --save_strategy "steps" \
    --save_steps 2000 \
    --save_total_limit 1 \
    --learning_rate 2e-5 \
    --weight_decay 0. \
    --warmup_ratio 0.03 \
    --lr_scheduler_type "cosine" \
    --logging_steps 1 \
    --fsdp "full_shard auto_wrap" \
    --fsdp_transformer_layer_cls_to_wrap 'LLaMADecoderLayer' \
    --tf32 True
```

03 Preliminary evaluation

- conduct human evaluation (by the 5 student authors) on the inputs from the self-instruct evaluation set.
  - self-instruct에서 human 평가를 위한 251개의 데이터셋이 있었음 (diverse list of user-oriented instructions including email writing, social media, and productivity tools.)
- 블라인드 테스트로 평가함, 결과는 비슷함 (90:89 = alpaca:davinci-003)
  - blind pairwise comparison between text-davinci-003 and Alpaca 7B
  - Alpaca wins 90 versus 89 comparisons against text-davinci-003.
- 생성된 길이가 ChatGPT보다 짧은건, text-davinci-003이 짧게 생성해주기 때문이다로 주장

03

Preliminary evaluation

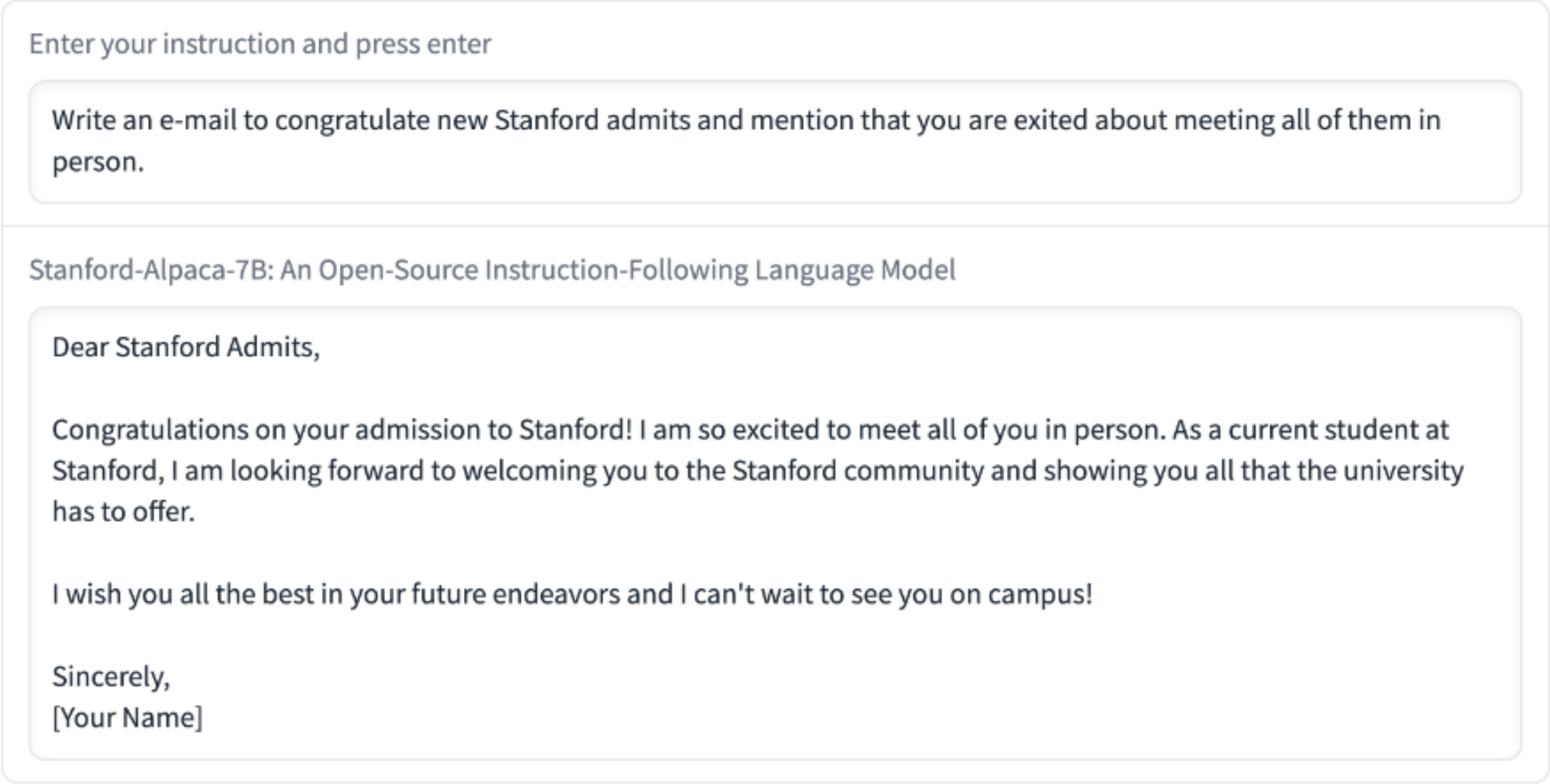


Enter your instruction and press enter

What is an alpaca? How is it different from a llama?

Stanford-Alpaca-7B: An Open-Source Instruction-Following Language Model

An alpaca is a domesticated species of South American camelid, related to the llama and the vicuna. It is smaller than a llama, and has finer and softer fleece. Alpacas are raised for their fleece, which is used to make knitted and woven garments.



Enter your instruction and press enter

Write an e-mail to congratulate new Stanford admits and mention that you are exited about meeting all of them in person.

Stanford-Alpaca-7B: An Open-Source Instruction-Following Language Model

Dear Stanford Admits,

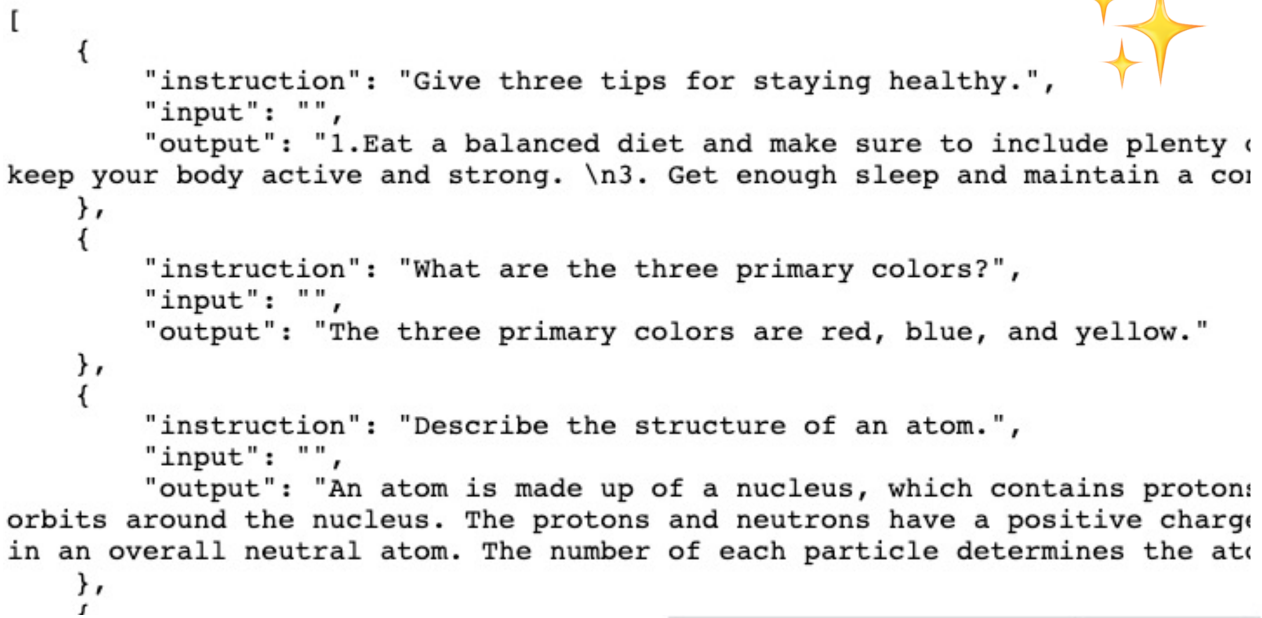
Congratulations on your admission to Stanford! I am so excited to meet all of you in person. As a current student at Stanford, I am looking forward to welcoming you to the Stanford community and showing you all that the university has to offer.

I wish you all the best in your future endeavors and I can't wait to see you on campus!

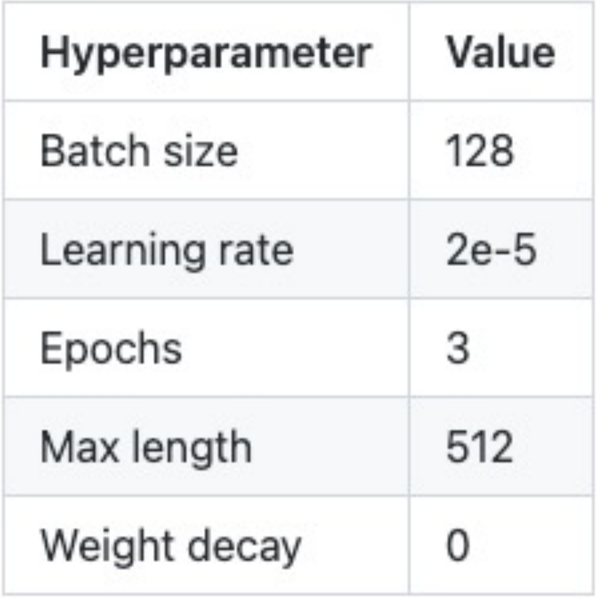
Sincerely,
[Your Name]

04 Assets released

- Demo: An interactive demo for everyone to try out Alpaca.
- Data: 52K demonstrations used to fine-tune Alpaca.
- Data generation process: the code for generating the data.
- Hyperparameters: for fine-tuning the model using the Hugging Face API.



```
[
    {
        "instruction": "Give three tips for staying healthy.",
        "input": "",
        "output": "1.Eat a balanced diet and make sure to include plenty o
keep your body active and strong. \n3. Get enough sleep and maintain a co
    },
    {
        "instruction": "What are the three primary colors?",
        "input": "",
        "output": "The three primary colors are red, blue, and yellow."
    },
    {
        "instruction": "Describe the structure of an atom.",
        "input": "",
        "output": "An atom is made up of a nucleus, which contains proton
orbits around the nucleus. The protons and neutrons have a positive charg
in an overall neutral atom. The number of each particle determines the at
    },
```

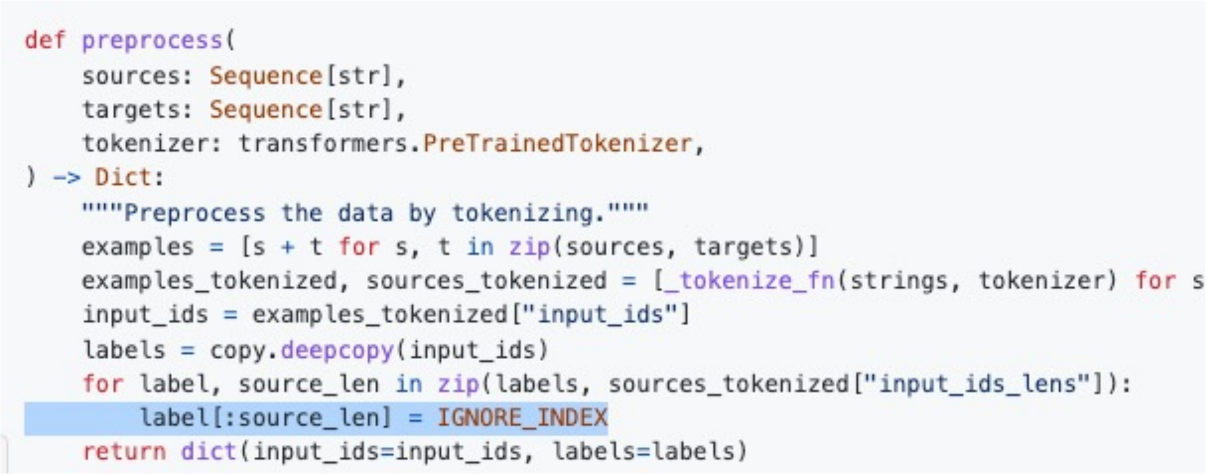


Hyperparameter	Value
Batch size	128
Learning rate	2e-5
Epochs	3
Max length	512
Weight decay	0

04 Assets released

- Model weights: We have reached out to Meta to obtain guidance on releasing the Alpaca model weights, both for the 7B Alpaca and for fine-tuned versions of the larger LLaMA models.

- Training code: our code uses the Hugging Face interface to LLaMA. As of now, the effort to support LLaMA is still ongoing and not stable. We will give the exact training commands once Hugging Face supports LLaMA officially.



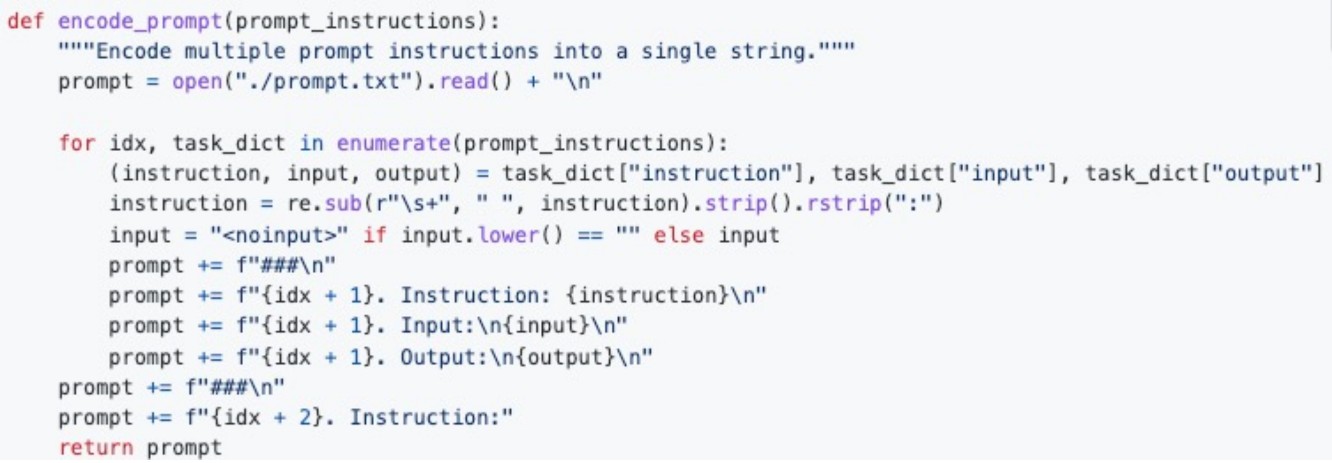
```
def preprocess(
    sources: Sequence[str],
    targets: Sequence[str],
    tokenizer: transformers.PreTrainedTokenizer,
) -> Dict:
    """Preprocess the data by tokenizing."""
    examples = [s + t for s, t in zip(sources, targets)]
    examples_tokenized, sources_tokenized = [_tokenize_fn(strings, tokenizer) for s
    input_ids = examples_tokenized["input_ids"]
    labels = copy.deepcopy(input_ids)
    for label, source_len in zip(labels, sources_tokenized["input_ids_lens"]):
        label[:source_len] = IGNORE_INDEX
    return dict(input_ids=input_ids, labels=labels)
```

05 Future directions

- Evaluation: We need to evaluate Alpaca more rigorously. We will start with HELM (Holistic Evaluation of Language Models)
- Safety: We would like to further study the risks of Alpaca and improve its safety using methods such as automatic red teaming, auditing, and adaptive testing.
- Understanding: We hope to better understand how capabilities arise from the training recipe. What properties of a base model do you need? What happens when you scale up? What properties of instruction data is needed? What are alternatives to using self-instruct on text-davinci-003?

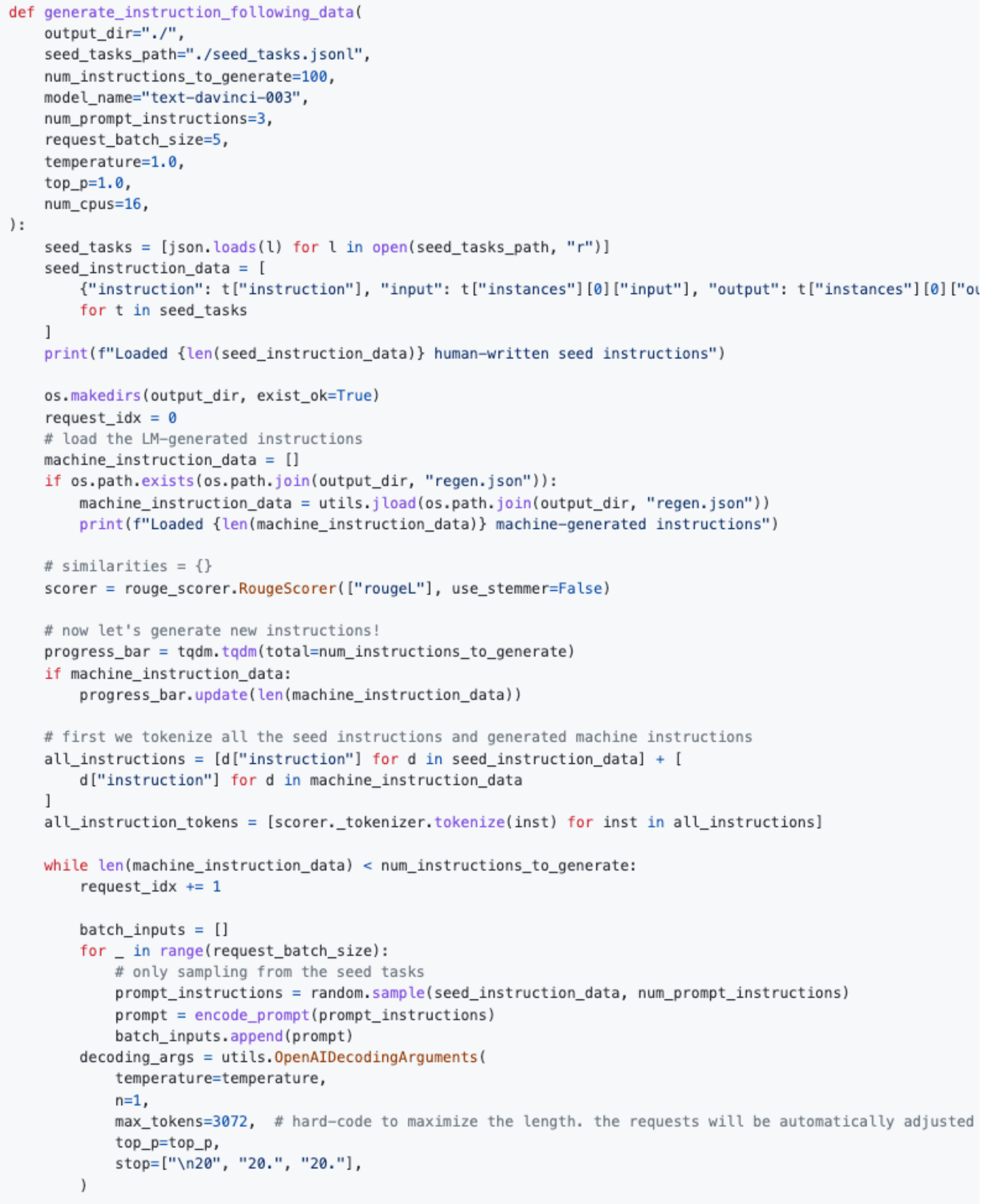
Appendix

Data generation snippets



```python
def encode_prompt(prompt_instructions):
    """Encode multiple prompt instructions into a single string."""
    prompt = open("./prompt.txt").read() + "\n"

    for idx, task_dict in enumerate(prompt_instructions):
        (instruction, input, output) = task_dict["instruction"], task_dict["input"], task_dict["output"]
        instruction = re.sub(r"\s+", " ", instruction).strip().rstrip(":")
        input = "<noinput>" if input.lower() == "" else input
        prompt += f"###\n"
        prompt += f"{idx + 1}. Instruction: {instruction}\n"
        prompt += f"{idx + 1}. Input:\n{input}\n"
        prompt += f"{idx + 1}. Output:\n{output}\n"
    prompt += f"###\n"
    prompt += f"{idx + 2}. Instruction:"
    return prompt
```



```python
def generate_instruction_following_data(
    output_dir="./",
    seed_tasks_path="./seed_tasks.jsonl",
    num_instructions_to_generate=100,
    model_name="text-davinci-003",
    num_prompt_instructions=3,
    request_batch_size=5,
    temperature=1.0,
    top_p=1.0,
    num_cpus=16,
):
    seed_tasks = [json.loads(l) for l in open(seed_tasks_path, "r")]
    seed_instruction_data = [
        {"instruction": t["instruction"], "input": t["instances"][0]["input"], "output": t["instances"][0]["ou
        for t in seed_tasks
    ]
    print(f"Loaded {len(seed_instruction_data)} human-written seed instructions")

    os.makedirs(output_dir, exist_ok=True)
    request_idx = 0
    # load the LM-generated instructions
    machine_instruction_data = []
    if os.path.exists(os.path.join(output_dir, "regen.json")):
        machine_instruction_data = utils.jload(os.path.join(output_dir, "regen.json"))
        print(f"Loaded {len(machine_instruction_data)} machine-generated instructions")

    # similarities = {}
    scorer = rouge_scorer.RougeScorer(["rougeL"], use_stemmer=False)

    # now let's generate new instructions!
    progress_bar = tqdm.tqdm(total=num_instructions_to_generate)
    if machine_instruction_data:
        progress_bar.update(len(machine_instruction_data))

    # first we tokenize all the seed instructions and generated machine instructions
    all_instructions = [d["instruction"] for d in seed_instruction_data] + [
        d["instruction"] for d in machine_instruction_data
    ]
    all_instruction_tokens = [scorer._tokenizer.tokenize(inst) for inst in all_instructions]

    while len(machine_instruction_data) < num_instructions_to_generate:
        request_idx += 1

        batch_inputs = []
        for _ in range(request_batch_size):
            # only sampling from the seed tasks
            prompt_instructions = random.sample(seed_instruction_data, num_prompt_instructions)
            prompt = encode_prompt(prompt_instructions)
            batch_inputs.append(prompt)
        decoding_args = utils.OpenAIDecodingArguments(
            temperature=temperature,
            n=1,
            max_tokens=3072,  # hard-code to maximize the length. the requests will be automatically adjusted
            top_p=top_p,
            stop=["\n20", "20.", "20."],
        )
```

JOOSUNG YOON

PRESENTATION EXPERT

Thank You!

Do you have any questions?