□□□□□□ ड्राइवर, □□□□□ और □□□□□□□

□□□□□ (लार्ज लैंग्वेज मॉडल मेटा □□) एक बड़े भाषा मॉडल (□□□□) का परिवार है, जिसे मेटा □□ द्वारा फरवरी 2023 से जारी किया गया है।

मैंने हाल ही में □□□□□□ □□□ के साथ अपना कंप्यूटर बनाया है। आप यहां देख सकते हैं, कंप्यूटर कैसे बनाएं, □□□□□://□□□□□□□□. □□□□□□□.□□/□□□□□□□□□।

उसके बाद, मैंने □□□□□ प्रोजेक्ट को चलाना शुरू किया। □□□□□ प्रोजेक्ट का □□□□□□□ □□□ है □□□□□://□□□□□□□.□□□/ □□□□□□□□□□□□□□□□□/□□□□□।

□□□□□□□ ड्राइवर इंस्टॉल करें

जब आप कमांड चलाते हैं,

```
torchrun --nproc_per_node 1 example_text_completion.py \
    --ckpt_dir llama-2-7b/ \
    --tokenizer_path tokenizer.model \
    --max_seq_len 128 --max_batch_size 4
```

(यह कोड ब्लॉक है और इसे अनुवादित नहीं किया जाना चाहिए।)

यह त्रुटि दिखाता है, "□□□□□□□□□□□□□: □□□□□□□□□□□□ □□□□□□□□ □□□□□□'□ □□□□ □□□□ □□□□□□ □□"। आइए □□□□ के बारे में जानें।

> □□□□□□□ □□□□□□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□□ (□□□□□) बहु-□□□ और बहु-नोड संचार प्राइमिटिव्स को लागू करता है जो □□□□□□□ □□□□□ और नेटवर्किंग के लिए अनुकूलित हैं।
> मैं □□□□□□□ ड्राइवरों को इंस्टॉल करने के लिए नीचे दिए गए वेबसाइटों का संदर्भ लेता हूँ।

- □ □□□□□ टूलकिट 12.2 अपडेट 1 डाउनलोड, □□□□□□://□□□□□□□□□□.□□□□□□□.□□□□/□□□□□-□□□□□□□□□□□
- □ □□□□□□□ □□□□□, □□□□□□://□□□□□□□□□□.□□□□□□□.□□□□/□□□□□
- □ □□□□□□□ डीप लर्निंग □□□□□ डॉक्यूमेंटेशन, □□□□□□://□□□□□.□□□□□□□.□□□□/□□□□□□□□□□□□□□□/□□□□□/□□□□□□□□-□□□□□□/□□□□□□.□□□□□
- □ □□□□□□□ □□□□□ इंस्टालेशन गाइड फॉर लिनक्स, □□□□□□://□□□□□.□□□□□□□.□□□□/□□□□□/□□□□□-□□□□□□□□□□□□□□□-□□□□□□-□□□□□□/□□□□□□.□□□□□
- □ □□□□□□□ इंस्टॉल करने के बाद, आप □□□□□□□□ □□□□ □□□□□□□□□□□□ का सामना करते हैं, □□□□□□://□□□□.□□□□□□□□□. □□□□/□□□□□□□-□□□□□□/□/13019226.□□□□□
- □ डीप लर्निंग के लिए □□□□□□□ 22.04, □□□□□□://□□□□□.□□□□□□□.□□□□/□□□□□-□□□□□□□□□/□3□8□06145□8569□0□0□030□□6□60□□□□
- □ □□□□□□□ 22.04 नोट्स, □□□□□□://□□□□□□□.□□□□/□□□□□□□□□/□□□□□□□□□□/□□□□□/□□□□□/□□□□□□□□3□

जब हम अपने ग्राफिक कार्ड के लिए □□□□□□ ड्राइवर को सफलतापूर्वक इंस्टॉल करते हैं, और फिर हम `nvidia-smi` कमांड का उपयोग करके इसकी जानकारी दिखाते हैं, तो यह नीचे दी गई जानकारी प्रदर्शित कर सकता है।

```
(base) lzw@lzw-MS-7E01:~$ nvidia-smi
   , 17    2023 04:15:43
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.86.10                    : 535.86.10    CUDA    : 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU                     Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
|                          :   /  |           -    | GPU-     Compute M. |
|                                        |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 4070            | 00000000:01:00.0     |                  N/A |
|  0%   34C    P8               9W / 215W |    666MiB / 12282MiB |     15%          |
|                                        |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
|     :                                                                                 |
|  GPU   GI   CI        PID                                                 GPU     |
|        ID   ID                                                                        |
|=======================================================================================|
|    0   N/A  N/A      1926      G   /usr/lib/xorg/Xorg                            381MiB |
|    0   N/A  N/A      2065      G   /usr/bin/gnome-shell                          120MiB |
|    0   N/A  N/A      3482      G   gnome-control-center                            2MiB |
|    0   N/A  N/A      3803      G   ...irefox/2987/usr/lib/firefox/firefox      149MiB |
+---------------------------------------------------------------------------------------+
```

वास्तव में, इस चरण तक पहुँचना कठिन है। कृपया यहाँ दिए गए लिंक को ध्यान से देखें, □□□□□□ 22.04 नोट्स, □□□□□://□□□□□□. □□□/□□□□□□□/□□□□□□□□/□□□□/□□□□/□□□□□□□3□।

□□□□□ सीखें

मॉडल डाउनलोड करने के बाद, और कमांड चलाने का प्रयास करते समय, हम नीचे दी गई त्रुटि का सामना करेंगे,

> □□□□□.□□□□.□□□□□□□□□□□□□□□□□□: □□□□ मेमोरी खत्म हो गई। 86.00 □□□ आवंटित करने का प्रयास किया गया (□□□ 0; 11.69 □□□ कुल क्षमता; 9.70 □□□ पहले से आवंटित; 64.81 □□□ मुक्त; □□□□□□□□ द्वारा कुल 9.70

> □□□ आरक्षित)। यदि आरक्षित मेमोरी □ आवंटित मेमोरी है, तो फ़्रैगमेंटेशन से बचने के लिए □□□_□□□□□□_□□□□□_□□ सेट करने का प्रयास करें।

चूंकि हमारे ग्राफिक कार्ड की मेमोरी केवल 12 □□ है, और □□□□□-2-7□ मॉडल का आकार लगभग 13□□ है, इसलिए हम इसे अपने ग्राफिक कार्ड पर चलाने में सक्षम नहीं हैं।

हमने दूसरे प्रोजेक्ट, □□□□-□□□□□-3□, □□□□□://□□□□□□□□□□□.□□/□□□□□□-□□□□□□□□□/□□□□_□□□□□_3□ का उपयोग करने का प्रयास किया।

हमें नीचे दिया गया त्रुटि सामने आती है।

> □□□□□□□□□□□□□: अपेक्षा की गई थी कि सभी टेंसर एक ही डिवाइस पर होंगे, लेकिन कम से कम दो डिवाइस पाए गए, □□□□:0 और □□□! (जब विधि □□□□□□□□_□□□□□__□□□□□□_□□□□□□□ में तर्क □□□□□□ के लिए तर्क की जाँच की गई)

और हम □□□□□□□ से इसके बारे में पूछते हैं।

□□□□□□□ हमें एक बहुत ही सुंदर समाधान प्रदान करता है। हमें नीचे दिए गए कोड को जोड़ने की आवश्यकता है।

```
input_ids = input_ids.to(model.device)
```

अंत में, हम इसे चला सकते हैं।

```
(llama) lzw@lzw-MS-7E01:~/Projects/open_llama_3b$ python run.py
Q:              ?
A:
Q:              ?
A:
```

हमने □□□□□2.□ प्रोजेक्ट, □□□□□://□□□□□□.□□□/□□□□□□□□□/□□□□□2.□, को भी आज़माया।

```
(base) lzw@lzw-MS-7E01:~/Projects/llama2.c$ ./run stories15M.bin
    ,                                 , "                             "                 ,
      ,                                                        ,                                    ,
                            , "                   "      , " ,                         "
                            ,                                                                      , "
achieved tok/s: 185.527329
```

□□□□□□□□ द्वारा कोड को समझें

ज़िवेई: क्या आप नीचे दिए गए कोड्स को समझा सकते हैं?

```
   def generate(
        self,
        prompt_tokens: List[List[int]],
        max_gen_len: int,
        temperature: float = 0.6,
        top_p: float = 0.9,
        logprobs: bool = False,
        echo: bool = False,
    ) -> Tuple[List[List[int]], Optional[List[List[float]]]]:
        params = self.model.params
        bsz = len(prompt_tokens)
        assert bsz <= params.max_batch_size, (bsz, params.max_batch_size)

min_prompt_len = min(len(t) for t in prompt_tokens)
max_prompt_len = max(len(t) for t in prompt_tokens)
assert max_prompt_len <= params.max_seq_len
total_len = min(params.max_seq_len, max_gen_len + max_prompt_len)
```

यह कोड स्निपेट निम्नलिखित कार्य करता है:

1. `min_prompt_len` में `prompt_tokens` की सबसे छोटी लंबाई (टोकन की संख्या) को स्टोर करता है।
2. `max_prompt_len` में `prompt_tokens` की सबसे बड़ी लंबाई (टोकन की संख्या) को स्टोर करता है।
3. यह सुनिश्चित करता है कि `max_prompt_len`, `params.max_seq_len` से कम या बराबर है। यदि ऐसा नहीं है, तो यह एक त्रुटि (□□□□□□□□□□ □□□□□) उत्पन्न करेगा।
4. `total_len` में `params.max_seq_len` और `max_gen_len + max_prompt_len` के बीच की न्यूनतम मान को स्टोर करता है।

```
pad_id = self.tokenizer.pad_id
tokens = torch.full((bsz, total_len), pad_id, dtype=torch.long, device="cuda")
for k, t in enumerate(prompt_tokens):
    tokens[k, : len(t)] = torch.tensor(t, dtype=torch.long, device="cuda")
if logprobs:
    token_logprobs = torch.zeros_like(tokens, dtype=torch.float)
```

```
prev_pos = 0
eos_reached = torch.tensor([False] * bsz, device="cuda")
input_text_mask = tokens != pad_id
for cur_pos in range(min_prompt_len, total_len):
    logits = self.model.forward(tokens[:, prev_pos:cur_pos], prev_pos)
    if logprobs:
        token_logprobs[:, prev_pos + 1 : cur_pos + 1] = -F.cross_entropy(
            input=logits.transpose(1, 2),
            target=tokens[:, prev_pos + 1 : cur_pos + 1],
            reduction="none",
            ignore_index=pad_id,
        )
    if temperature > 0:
        probs = torch.softmax(logits[:, -1] / temperature, dim=-1)
        next_token = sample_top_p(probs, top_p)
    else:
        next_token = torch.argmax(logits[:, -1], dim=-1)

next_token = next_token.reshape(-1)
#
next_token = torch.where(
    input_text_mask[:, cur_pos], tokens[:, cur_pos], next_token
)
tokens[:, cur_pos] = next_token
eos_reached |= (~input_text_mask[:, cur_pos]) & (
    next_token == self.tokenizer.eos_id
)
prev_pos = cur_pos
if all(eos_reached):
    break

        if logprobs:
            token_logprobs = token_logprobs.tolist()
        out_tokens, out_logprobs = [], []
        for i, toks in enumerate(tokens.tolist()):
            # max gen len
            start = 0 if echo else len(prompt_tokens[i])
```

```
            toks = toks[start : len(prompt_tokens[i]) + max_gen_len]
            probs = None
            if logprobs:
                probs = token_logprobs[i][start : len(prompt_tokens[i]) + max_gen_len]
            # eos tok     ,
            if self.tokenizer.eos_id in toks:
                eos_idx = toks.index(self.tokenizer.eos_id)
                toks = toks[:eos_idx]
                probs = probs[:eos_idx] if logprobs else None
            out_tokens.append(toks)
            out_logprobs.append(probs)
        return (out_tokens, out_logprobs if logprobs else None)
```

□□□□□□□□: ज़रूर, यहां दिए गए कोड की व्याख्या है:

```
def generate(
    self,
    prompt_tokens: List[List[int]],
    max_gen_len: int,
    temperature: float = 0.6,
    top_p: float = 0.9,
    logprobs: bool = False,
    echo: bool = False,
) -> Tuple[List[List[int]], Optional[List[List[float]]]]:
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह प्रोग्रामिंग कोड है और इसे मूल रूप में ही रहना चाहिए।)

यह फ़ंक्शन `generate` नामक है और कई पैरामीटर्स को स्वीकार करता है:

- `prompt_tokens` एक सूची है जिसमें प्रॉम्प्ट के लिए टोकन के अनुक्रम शामिल होते हैं।
- `max_gen_len` उत्पन्न पाठ की अधिकतम लंबाई है।
- `temperature` उत्पन्न करने के दौरान यादृच्छिकता को नियंत्रित करने के लिए उपयोग किया जाने वाला एक पैरामीटर है।
- `top_p` 0 और 1 के बीच एक संभाव्यता सीमा है जो नमूना लेने के दौरान विविधता को नियंत्रित करने के लिए उपयोग की जाती है।
- `logprobs` एक बूलियन है जो यह दर्शाता है कि उत्पन्न करने की प्रक्रिया के लॉग संभाव्यताओं को रिकॉर्ड करना है या नहीं।
- `echo` एक बूलियन है जो यह दर्शाता है कि उत्पन्न करने के दौरान इनपुट को प्रतिध्वनित करना है या नहीं।

```
params = self.model.params
bsz = len(prompt_tokens)
assert bsz <= params.max_batch_size, (bsz, params.max_batch_size)
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह प्रोग्रामिंग कोड है और इसे अपरिवर्तित छोड़ दिया जाना चाहिए।)

यहां, फ़ंक्शन मॉडल पैरामीटर्स प्राप्त करता है और प्रॉम्प्ट्स के बैच आकार की गणना करता है। यह सुनिश्चित करता है कि बैच आकार मॉडल पैरामीटर्स में निर्दिष्ट अधिकतम बैच आकार से अधिक न हो।

```
min_prompt_len = min(len(t) for t in prompt_tokens)
max_prompt_len = max(len(t) for t in prompt_tokens)
assert max_prompt_len <= params.max_seq_len
total_len = min(params.max_seq_len, max_gen_len + max_prompt_len)
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह प्रोग्रामिंग कोड है और इसे अपने मूल रूप में ही रहना चाहिए।)

ये लाइन्स प्रॉम्प्ट्स की न्यूनतम और अधिकतम लंबाई की गणना करती हैं और यह सुनिश्चित करती हैं कि प्रॉम्प्ट की अधिकतम लंबाई मॉडल पैरामीटर्स में निर्दिष्ट अधिकतम अनुक्रम लंबाई से अधिक न हो। कुल लंबाई की गणना अधिकतम उत्पन्न लंबाई और अधिकतम प्रॉम्प्ट लंबाई के आधार पर की जाती है।

```
pad_id = self.tokenizer.pad_id
tokens = torch.full((bsz, total_len), pad_id, dtype=torch.long, device="cuda")
for k, t in enumerate(prompt_tokens):
    tokens[k, : len(t)] = torch.tensor(t, dtype=torch.long, device="cuda")
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह प्रोग्रामिंग कोड है और इसे मूल रूप में ही रखना चाहिए।)

यह खंड `tokens` टेंसर को पैडिंग टोकन के साथ प्रारंभ करता है और इसे प्रॉम्प्ट्स से टोकन अनुक्रमों से भरता है।

```
if logprobs:
    token_logprobs = torch.zeros_like(tokens, dtype=torch.float)
```

(यह कोड ब्लॉक है, इसलिए इसे अनुवादित नहीं किया गया है।)

यदि `logprobs` □□□□ है, तो लॉग प्रायिकताओं को संग्रहीत करने के लिए एक टेंसर `token_logprobs` बनाया जाता है।

```
prev_pos = 0
eos_reached = torch.tensor([False] * bsz, device="cuda")
input_text_mask = tokens != pad_id
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह प्रोग्रामिंग कोड है और इसे अपरिवर्तित छोड़ा जाना चाहिए।)

वेरिएबल्स `prev_pos` (पिछली स्थिति), `eos_reached` (क्या अनुक्रम के अंत के टोकन मिले हैं), और `input_text_mask` (पैडिंग से भरे नहीं गए स्थान) को आरंभ किया जाता है।

निम्नलिखित लूप न्यूनतम प्रॉम्प्ट लंबाई से शुरू होकर कुल लंबाई तक टोकन उत्पन्न करता है।

```
for cur_pos in range(min_prompt_len, total_len):
    logits = self.model.forward(tokens[:, prev_pos:cur_pos], prev_pos)
    if logprobs:
        token_logprobs[:, prev_pos + 1 : cur_pos + 1] = -F.cross_entropy(
            input=logits.transpose(1, 2),
            target=tokens[:, prev_pos + 1 : cur_pos + 1],
            reduction="none",
            ignore_index=pad_id,
        )
    if temperature > 0:
        probs = torch.softmax(logits[:, -1] / temperature, dim=-1)
        next_token = sample_top_p(probs, top_p)
    else:
        next_token = torch.argmax(logits[:, -1], dim=-1)

    next_token = next_token.reshape(-1)
    next_token = torch.where(
        input_text_mask[:, cur_pos], tokens[:, cur_pos], next_token
    )
    tokens[:, cur_pos] = next_token
    eos_reached |= (~input_text_mask[:, cur_pos]) & (
        next_token == self.tokenizer.eos_id
    )
    prev_pos = cur_pos
    if all(eos_reached):
        break
```

इस लूप के अंदर, मॉडल अगले टोकन के लिए एक प्रायिकता वितरण `logits` उत्पन्न करता है। यदि `logprobs` □□□□ है, तो यह क्रॉस-एन्ट्रॉपी का उपयोग करके लॉग प्रायिकताओं की गणना करता है। `temperature` के आधार पर, यह अगले टोकन का चयन करने के लिए □□□□□□□ या □□□□□□ का उपयोग करता है। फिर यह `tokens` टेंसर को अपडेट करता है और अनुक्रम के अंत (□□□-□□-□□□□□□□□) के टोकन की जांच करता है।

```
if logprobs:
    token_logprobs = token_logprobs.tolist()
```

(यह कोड ब्लॉक को हिंदी में अनुवाद करने की आवश्यकता नहीं है क्योंकि यह प्रोग्रामिंग कोड है और इसे अपरिवर्तित छोड़ दिया जाना चाहिए।)

यदि `logprobs` □□□□ है, तो लॉग प्रायिकता टेंसर को □□□□□□ लिस्ट में बदल दिया जाता है।

```
out_tokens, out_logprobs = [], []
for i, toks in enumerate(tokens.tolist()):
    ...
```

इस लूप में, प्रत्येक इनपुट सैंपल के लिए, यह आउटपुट टोकन अनुक्रम और संबंधित लॉग प्रायिकताएं उत्पन्न करता है।

```
return (out_tokens, out_logprobs if logprobs else None)
```

(यह कोड ब्लॉक है, इसे अनुवादित नहीं किया जाना चाहिए।)

अंत में, फ़ंक्शन उत्पन्न टोकन अनुक्रम और उनके संबंधित लॉग प्रायिकताओं (यदि अनुरोध किया गया हो) को वापस करता है।