□□□□□□□ و □□□□□، □□□□□□ □□□□□□

□□□□ بواسطة إصدارها تم، □□□□□□ الكبيرة اللغة نماذج من عائلة هي □□□ □□□□ الكبيرة اللغة □نموذج □□□□□
□□ بدءًا من فبراير 2023.

ذلك من التحقق يمكنك. □□□□□□ من الرسومات معالجة وحدة مع بي الخاص الكمبيوتر جهاز ببناء مؤخرًا قمت لقد
□□□□□:□□□□□□□□□.□□□□□□.□□□□□□□□□□□□. كمبيوتر، جهاز بناء كيفية هنا،

بعد ذلك، بدأت في تشغيل مشروع □□□□□. عنوان □□□□□□ لمشروع □□□□□ هو □□□□□:□□□□□□□□□.□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□.

□□□□□□ تشغيل برنامج تثبيت

عندما قمت بتنفيذ الأمر،

```
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
```

يظهر الخطأ التالي: □□□□□□□□□□□□□: الحزمة الموزعة لا تحتوي على □□□□. مدمجًا دعونا نتعرف على □□□□.

> مكتبة الاتصال الجماعي من □□□□□□ □□□□□□ □□□□□□ تنفذ بدائيات الاتصال متعددة وحدات معالجة
> الرسوميات □□□□□□ العقد ومتعددة □□□□□□□□□□□□ المُحسنة لوحدات معالجة الرسوميات من
> □□□□□□ .والشبكات أنا أشير إلى المواقع التالية لتثبيت برامج تشغيل □□□□□□.

- تنزيلات □□□□ □□□□□□□ 12.2 التحديث 1، □□□□□:□□□□□□□□□□□.□□□□□□.□□□□□□□□□ □□□□□□□□□
- □□□□□□ □□□□، □□□□□:□□□□□□□□□□□.□□□□□□.□□□□□□□□□
- وثائق □□□□□□ □□□□ □□□□□□□□ □□□□، □□□□□:□□□□□□.□□□□□□.□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□.□□□□□
- دليل تثبيت □□□□□□ □□□□ لنظام □□□□□، □□□□□:□□□□□□.□□□□□□.□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.□□□□□
- بعد تثبيت □□□□□□، توجه إدارة □□□، □□□□□:□□□□□.□□□□□□□.□□□□□□□□□□□□□□□□□□□□ 13019226.□□□□
- □□□□□□ 22.04 للتعلم العميق، □□□□□:□□□□□□.□□□□□□.□□□□□□□□□□□□□□□□□□□□□3□8□06145□8569□0□0□030□□
- □□□□□□ 22.04 الملاحظات، □□□□□:□□□□□□□□.□□□3□

الأمر نستخدم ثم بنا، الخاصة الرسومات لبطاقة بنجاح □□□□□ تشغيل برنامج بتثبيت نقوم عندما التالية. المعلومات تظهر أن يمكن تفاصيلها، لعرض nvidia-smi

```
(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-Util  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 |
+---------------------------------------------------------------------------------------+
```

22.04، □□□□□ ملاحظات بعناية، هنا الرابط إلى الرجوع يرجى المرحلة. هذه إلى الوصول الصعب من الواقع، في □□□□□:□□□□□□□□□.□□3□.

□□□□□ تعلم

□□□□ طورتها التي الكبيرة اللغة نماذج من مجموعة هي □□□ □□□□ □□□□□ □□□□□ □□□□□□□□□ □□□□□□□ □□□□□□ □□□□□ المعروفة□ سابقًا باسم □□□□□□□□□. هذه النماذج مصممة لتكون فعالة من حيث الحجم وقادرة على أداء مهام معالجة اللغة الطبيعية □□□□ بشكل جيد. إذا كنت ترغب في تعلم المزيد عن □□□□□ وكيفية استخدامها، إليك بعض الخطوات التي يمكنك اتباعها:

1. أساسيات فهم [illegible]

- ابدأ بقراءة الأوراق البحثية والمستندات الرسمية التي تصف [illegible]. تأكد من فهم هذه السياسات على كيفية عمل النموذج وما هي المهام التي يمكنه القيام بها.
- تعرف على الفرق بين [illegible] النماذج اللغوية والنماذج الأخرى مثل [illegible] و[illegible].

2. إعداد البيئة

- تأكد من أن لديك بيئة برمجية بسبات مناسبة لتشغيل النماذج اللغوية. يمكنك استخدام [illegible] مع [illegible] مثل [illegible] أو [illegible].
- قم بتثبيت المكتبات اللازمة مثل transformers من [illegible] التي تدعم تشغيل النماذج [illegible].

3. تنزيل النموذج

- يمكن تنزيل النماذج من مصادر موثوقة مثل [illegible] [illegible] [illegible] [illegible].
- تأكد من أن لديك مساحة تخزين كافية وحساب [illegible] إذا كنت تخطط لتشغيل النموذج على جهازك.

4. التدريب والضبط

- إذا كنت ترغب في تحسين أداء النموذج لمهمة معينة، يمكن ضبط النموذج باستخدام [illegible] خاصة بياناتك.
- استخدم أدوات مثل Trainer من [illegible] [illegible] لتسهيل عملية التدريب.

5. التقييم والاختبار

- قم بتقييم النموذج باستخدام مقاييس مثل الدقة أو [illegible] أو [illegible]1 [illegible].
- قم باختبار النموذج على بيانات جديدة لضمان أنه يعمل بشكل جيد في العالم الحقيقي.

6. التطبيقات العملية

- استخدم [illegible] في تطبيقات عملية مثل توليد النصوص، الترجمة الآلية، الإجابة على الأسئلة، وغيرها.
- يمكنك أيضًا دمج [illegible] برمجية واجهات باستخدام الخاصة بتطبيقاتك في [illegible].

7. الموارد الإضافية

- انضم إلى مجتمعات المطورين والمهتمين بالذكاء الاصطناعي على منصات مثل □□□□□□ □□□□□□و □□□□□□□و.
- تابع الدورات التعليمية والبرامج التعليمية المتاحة على الإنترنت لتعميق فهمك.

باتباع هذه الخطوات، ستتمكن من تعلم □□□□ واستخدامها بشكل فعال في مشاريعك الخاصة.

بعد تنزيل النماذج، ومحاولة تشغيل الأمر، سنواجه الخطأ التالي

> □□□□□.□□□□.□□□□□□□□□□□□□□□□□: نفاد ذاكرة □□□□. تمت محاولة تخصيص 86.00 ميجابايت (□□□□ 0؛ إجمالية سعة 11.69 جيجابايت؛ تم تخصيص 9.70 جيجابايت بالفعل؛ 64.81 ميجابايت متاحة؛ 9.70 جيجابايت محجوزة إجمالًا بواسطة □□□□□□□□). إذا كانت الذاكرة المحجوزة أكبر بكثير من الذاكرة المخصصة، حاول ضبط max_split_size_mb لتجنب التجزئة.

نظرًا لأن ذاكرة بطاقة الرسوميات لدينا هي 12 جيجابايت فقط، وحجم نموذج □□□□□□2□7□ يبلغ حوالي 13 جيجابايت، فإننا لا نستطيع تشغيله باستخدام بطاقة الرسوميات الخاصة بنا.

نحاول استخدام المشروع الآخر، □□□□□□□□□□3□، □□□□□:□□□□□□□□□□□□□□.□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□3□.

نواجه الخطأ التالي:

> □□□□□□□□□□□□: توقع أن تكون جميع الموترات على نفس الجهاز، ولكن تم العثور على جهازين على الأقل، □□□□:0 و □□□! (عند التحقق من الوسيط □□□□□ في الدالة □□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□

ونحن نسأل □□□□□□□ عن هذا.

يوفر لنا □□□□□□□ إصلاحًا جميلًا جدًا. نحتاج إلى إضافة الكود التالي.

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

تم تحويل input_ids إلى الجهاز الذي يعمل عليه النموذج (model.device).

أخيرًا، يمكننا تشغيله.

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

لقد جربنا أيضًا المشروع □□□□□2.□، □□□□□:□□□□□□□□.□□□□□□□□□□□□□□□□□□□2.□.

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

فهم الكود بواسطة □□□□□□□

□□□□□□: هل يمكنك شرح الأكواد التالية؟

```python
   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)
```

تم ترجمة الكود أعلاه إلى:

```python
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)   #
```

```
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()):
            #
            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]
            #
            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)
```

بالطبع، إليك شرحًا للكود المقدم:

```
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]]]]:
```

هذه الوظيفة تُسمى generate وتقبل عدة معاملات:

- prompt_tokens هي قائمة تحتوي على تسلسلات من الرموز □□□□□□□□ للطلبات □□□□□□□□□□.
- max_gen_len هو الطول الأقصى للنص الذي يتم إنشاؤه.
- temperature هي معلمة تُستخدم للتحكم في العشوائية أثناء عملية الإنشاء.
- top_p هو عتبة احتمالية بين 0 و1 تُستخدم للتحكم في التنوع أثناء عملية أخذ العينات.

- الاحتمالية اللوغاريتمات تسجيل يجب كان إذا ما إلى تشير منطقية قيمة هي logprobs الإنشاء. لعملية
- الإنشاء. عملية أثناء المدخلات تكرار يجب كان إذا ما إلى تشير منطقية قيمة هي echo

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

إلى: أعلاه الكود ترجمة تم

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

هي كما تبقى ما عادةً والتي بالإنجليزية، ودوال متغيرات أسماء على يحتوي لأنه ترجمته يتم لم الكود ملاحظة: البرمجة. في

الدالة تضمن للطلبات. الدُفعة حجم وحساب النموذج معلمات على بالحصول الدالة تقوم هنا، النموذج. معلمات في المحدد الدُفعة لحجم الأقصى الحد يتجاوز لا الدُفعة حجم أن

```
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)  #
```

الحد يتجاوز ألا وتضمن المقدمة النصوص لأطوال الأقصى والحد الأدنى الحد تحسب الأسطر هذه الطول حساب يتم النموذج. معلمات في المحدد التسلسل لطول الأقصى الحد المقدم النص لطول الأقصى المقدم. للنص الأقصى والطول إنشاؤه يمكن الذي الأقصى الطول على بناءً الإجمالي

```
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")
```

الرموز بتسلسلات ويملؤه الحشو برموز tokens موتر بتهيئة القسم هذا يقوم المقدمة النصوص من المستخرجة.

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

الاحتمالية. اللوغاريتمات لتخزين token_logprobs موتر إنشاء يتم ،True تساوي logprobs كانت إذا

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

إلى: أعلاه الكود ترجمة تم

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

ترجمتها لا يتم والتي، بلغة ودوال متغيرات أسماء على يحتوي لأنه ترجمته يتم لم الكود ملاحظة:
عادةً.

يتم تهيئة المتغيرات prev_pos موضع السابق، ما eos_reached إذا تم الوصول إلى رموز نهاية بالحشو. ملؤها يتم التي المواضع input_text_mask و التسلسل،

التالية الحلقة تولد رموز من بدءًا المطابقة لطول الأدنى الحد إلى وصولاً الطول. الكلي

```
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
```

إلى: أعلاه الكود ترجمة تمت

```
    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
```

ملاحظة: الكود لم يتم ترجمته لأنه يحتوي على أسماء ودوال محددة في لغة البرمجة التي لا يتم [illegible] عادةً.

فإنه صحيحًا، logprobs كان إذا التالي. للرمز logits احتمالي توزيع بإنشاء النموذج يقوم الحلقة، هذه داخل يسخدم temperature، على اعتمادًا المتقطعة. الانتروبيا باستخدام الاحتمالية اللوغاريتمات بحساب يقوم ثم التالي. الرمز لتحديد [illegible] أو [illegible] tokens من وفحص رموز أية نهاية للتسلسل.

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

إذا كانت logprobs تساوي True، يتم تحويل الموتر الاحتمالات اللوغاريتمات إلى قائمة [illegible].

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

في هذه الحلقة، يتم إدخال عينة لكل تسلسل يولد الرموز الناتج والاحتمالات اللوغاريتمية والارتباط بها.

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

ذلك. طلب تم إذا لها المقابلة اللوغاريتمية والاحتمالات المُنشأة الرموز تسلسلات الدالة تُعيد أخيرًا،