ExamplesLlama3

September 25, 2025

1 Llama3-70b quantized examples

Make sure you are on campus connected through Eduroam (not NinerWifi-Guest).

Do not distribute these examples outside of class, in particular do not share the url and model name. This is to ensure that sufficient bandwidth is available for students in the class (otherwise the model may become unresponsive).

[ ]:
```
#!pip install openai
```

[7]:
```
import httpx
from openai import OpenAI

# Set the Llama API base URL.
BASE_URL = "https://cci-llm.charlotte.edu/api/v1"

# Initialize client with SSL verification disabled
client = OpenAI(base_url = BASE_URL,
                http_client = httpx.Client(verify = False),
                api_key = '3jdhdi4xkf-45')

model_name = "Llama-3.3-70B-Instruct"
```

1.1 Conversational API example

[8]:
```
import json

# Define the conversation
query = "Justin sits next to Razvan. One of them is happy and one of them is␣
 ↪grumpy. " \
                "The person sitting next to Justin is grumpy. Who is happy?"
conversation = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": query}
]

# Send a chat completion request
```

```
response =  client.chat.completions.create(
    model = model_name,
    messages = conversation,
    max_tokens = 300,
    temperature = 0
)

reply = response.choices[0].message.content
print(f"Text response: {reply}")

# print(f"API JSON response: {response.model_dump_json()}\n")
print(f"API raw response: {response}\n")
```

```
API raw response: ChatCompletion(id='chatcmpl-b88370f5e6334bda9d5cf6310c760f27',
choices=[Choice(finish_reason='stop', index=0, logprobs=None,
message=ChatCompletionMessage(content="Let's break it down:\n\n* Justin sits
next to Razvan.\n* The person sitting next to Justin is grumpy.\n* Since Justin
sits next to Razvan, the person sitting next to Justin must be Razvan.\n* So,
Razvan is grumpy.\n* Since one of them is happy and the other is grumpy, Justin
must be happy.\n\nTherefore, Justin is happy.", refusal=None, role='assistant',
annotations=None, audio=None, function_call=None, tool_calls=[],
reasoning_content=None), stop_reason=None, token_ids=None)], created=1758818234,
model='Llama-3.3-70B-Instruct', object='chat.completion', service_tier=None,
system_fingerprint=None, usage=CompletionUsage(completion_tokens=80,
prompt_tokens=75, total_tokens=155, completion_tokens_details=None,
prompt_tokens_details=None), prompt_logprobs=None, prompt_token_ids=None,
kv_transfer_params=None)

Text response: Let's break it down:

* Justin sits next to Razvan.
* The person sitting next to Justin is grumpy.
* Since Justin sits next to Razvan, the person sitting next to Justin must be
Razvan.
* So, Razvan is grumpy.
* Since one of them is happy and the other is grumpy, Justin must be happy.

Therefore, Justin is happy.
```

1.2 Sequence completion example

[9]:
```
question = "Provide the next number in the sequence 1, 2, 3, 5, 5, 8, 7, 11, 9,␣
 ↪..."
conversation = [
    {"role": "system", "content": "You are a helpful, intelligent assistant."},
    {"role": "user", "content": question}
]
```

```
response = client.chat.completions.create(
    model = model_name,
    messages = conversation,
    max_tokens = 1000,
    temperature = 0
)

# Print the response.
reply = response.choices[0].message.content
print(reply)
```

```
To find the next number in the sequence, let's first try to identify a pattern.
The sequence appears to be alternating between two different sequences: the
Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, …) and a sequence that seems to be
decreasing by 2, then increasing by some amount.

Breaking it down:
- The Fibonacci sequence is: 1, 2, 3, 5, 8, 13, …
- The other sequence, as observed from the given sequence, seems to be: 2 (not
present, assuming a starting point), 5 (matches the Fibonacci), 7 (2 less than
the next Fibonacci, which would be 9), 9 (2 less than the next Fibonacci, which
would be 11), and if the pattern continues, the next number should be 2 less
than the next Fibonacci after 13, which is 21, so 21 - 2 = 19, but considering
the pattern observed, it seems to alternate in a way that doesn't strictly
follow a simple arithmetic progression or the Fibonacci sequence.

However, a closer look suggests the sequence might actually be an alternation
between the Fibonacci sequence and another sequence that decreases or adjusts in
a specific way. Given the numbers:
- 1 (Fibonacci)
- 2 (Fibonacci)
- 3 (Fibonacci)
- 5 (Fibonacci)
- 5
```

1.2.1 Follow-up question requires sending the conversation so far

[4]:
```
new_question = "Can you explain why?"
conversation = conversation + [
    {"role": "assistant", "content": reply},
    {"role": "user", "content": new_question}
]

# Print the updated conversation.
print(conversation)
print()
```

```
response = client.chat.completions.create(
    model = model_name,
    messages = conversation,
    max_tokens = 300,
    temperature = 0
)

# Print the response.
reply = response.choices[0].message.content
print(reply)
```

```
[{'role': 'system', 'content': 'You are a helpful, intelligent assistant.'},
{'role': 'user', 'content': 'Provide the next number in the sequence 1, 2, 3, 5,
5, 8, 7, 11, 9, …'}, {'role': 'assistant', 'content': "A fascinating
sequence!\n\nAfter analyzing the given numbers, I'm going to take a educated
guess. It appears to be a mix of consecutive integers and Fibonacci numbers.
Here's my prediction for the next number in the sequence:\n\n1, 2, 3, 5, 5, 8,
7, 11, 9, 13\n\nAm I correct?"}, {'role': 'user', 'content': 'Can you explain
why?'}]

I'd be happy to break down my thought process.

Upon examining the sequence, I noticed that it seems to be a combination of two
patterns:

1. **Consecutive integers**: The sequence starts with consecutive integers: 1,
2, 3.
2. **Fibonacci numbers**: The sequence then appears to incorporate Fibonacci
numbers, which are a series of numbers in which each number is the sum of the
two preceding numbers: 0, 1, 1, 2, 3, 5, 8, 13, …

Here's how I think the sequence is constructed:

* It starts with three consecutive integers: 1, 2, 3.
* The next number, 5, is a Fibonacci number (3 + 2 = 5).
* The sequence then repeats the previous Fibonacci number, 5, which is unusual
but might be a deliberate pattern.
* The next number, 8, is the next Fibonacci number (5 + 3 = 8).
* The sequence then takes a step back to 7, which is one less than the previous
Fibonacci number (8 - 1 = 7).
* The next number, 11, is not a Fibonacci number, but it's close to the next one
(11 is one more than 10, which is close to the Fibonacci number 13).
* The sequence then takes a step back to 9, which is one less than the previous
number (11 -
```

[ ]:
```
# Can you think of a hint you can give the LLM so that it solves the problem?
# Add this hint as a next turn to the current or previous turn in the␣
 ↪conversation.
```

1.3 Text style transfer example

[5]:
```
sample_zs = "Suddenly I could hear Q-Tip, with his human voice, rapping over a␣
 ↪human beat. " \
            "And the top of my skull opened to let human Q-Tip in, and a␣
 ↪rail-thin man with enormous eyes " \
            "reached across a sea of bodies for my hand. He kept asking me the␣
 ↪same thing over and over: " \
            "You feeling it? I was. My ridiculous heels were killing me, I was␣
 ↪terrified I might die, yet " \
            "I felt simultaneously overwhelmed with delight that the song␣
 ↪should happen to be playing at " \
            "this precise moment in the history of the world. I took the man's␣
 ↪hand. The top of my head flew away."

instruction = f'Rewrite the text below such that the first point of view is␣
 ↪replaced with a third person point of view of a person called Emma.\n\n␣
 ↪"{sample_zs}"'

conversation = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": instruction}
]

# Send a chat completion request
response =  client.chat.completions.create(
    model = model_name,
    messages = conversation,
    max_tokens = 300,
    temperature = 0
)

# Print the response.
reply = response.choices[0].message.content
print(reply)
```

```
Here is the rewritten text in the third person point of view from Emma's
perspective:

"Suddenly Emma could hear Q-Tip, with his human voice, rapping over a human
beat. And the top of her skull opened to let human Q-Tip in, and a rail-thin man
with enormous eyes reached across a sea of bodies for her hand. He kept asking
her the same thing over and over: You feeling it? She was. Her ridiculous heels
```

were killing her, she was terrified she might die, yet she felt simultaneously overwhelmed with delight that the song should happen to be playing at this precise moment in the history of the world. She took the man's hand. The top of her head flew away."

[7]:
```
# Can you try the same examle, but this time changing the 1st person PoV to 3rd person *masculine* gender.
# This is harder, as there is already a character described from masculine 3rd person PoV.
```

1.4 Python & JSON Comprehension

[8]:
```
question = 'Consider the following monologue from the movie Stalker by Andrei Tarkovsky: ' \
           '"Let them be helpless like children, because weakness is a great thing, and strength is nothing. ' \
           'When a man is just born, he is weak and flexible. When he dies, he is hard and insensitive. ' \
           'When a tree is growing, it\'s tender and pliant. But when it\'s dry and hard, it dies. ' \
           'Hardness and strength are death\'s companions. Pliancy and weakness are expressions of the ' \
           'freshness of being. Because what has hardened will never win." ' \
           'Where else was a similar idea expressed? Provide quotes. Format your answer as a Python dictionary ' \
           'mapping the author or source name to the actual passage expressing a similar idea.'

conversation = [{"role": "system", "content": "You are a helpful librarian."},
            {"role": "user",
             "content": question}]

response = client.chat.completions.create(
    model = model_name,
    messages = conversation,
    max_tokens = 700,
    temperature = 0
)

print(response.choices[0].message.content)
```

What a profound monologue! I'm happy to help you explore similar ideas expressed by other authors and sources. Here's a Python dictionary with some examples:

```
similar_ideas = {
    "Lao Tzu": "Nature does not hurry, yet everything is accomplished. A tree
```

```
that is unbending is easily cracked. The hard and strong will fall, the soft and
supple will overcome.",
    "Bruce Lee": "You must be shapeless, formless, like water. Now, you put
water into a cup, it becomes the cup. You put water into a bottle, it becomes
the bottle. You put it in a teapot, it becomes the teapot. Now, water can flow
or it can crash. Be water, my friend.",
    "Tao Te Ching (Chapter 36)": "If you want to shrink something, you must
first allow it to expand. If you want to get rid of something, you must first
allow it to flourish. If you want to take something, you must first give it.",
    "Hermann Hesse": "The tree which stands apart, unsheltered, is the one which
most invites the fury of the winds: and those which are most exposed to the fury
of the winds are those which are most deeply rooted.",
    "Eckhart Tolle": "The primary cause of unhappiness is never the situation
but rather the thoughts about it. Be aware of your thoughts, but don't be your
thoughts. Watch your thoughts, but don't be watched by your thoughts.",
    "Buddha": "In the end, only three things matter: how much you loved, how
gently you lived, and how gracefully you let go of things not meant for you."
}
```

These quotes convey the idea that flexibility, weakness, and pliability are
often more desirable than strength and hardness, as they allow for growth,
adaptation, and resilience.

[ ]: