

# llama\_test

April 12, 2024

```
[ ]: # !pip install -q -U bitsandbytes
# !pip install -q -U git+https://github.com/huggingface/transformers.git
# !pip install -q -U git+https://github.com/huggingface/peft.git
# !pip install -q -U git+https://github.com/huggingface/accelerate.git
# !pip install -q -U datasets scipy ipywidgets matplotlib

[ ]: from unsloth import FastLanguageModel
import torch
max_seq_length = 2048 # Choose any! We auto support RoPE Scaling internally!
dtype = None # None for auto detection. Float16 for Tesla T4, V100, Bfloat16
    ↪ for Ampere+
load_in_4bit = True # Use 4bit quantization to reduce memory usage. Can be
    ↪ False.

# 4bit pre quantized models we support for 4x faster downloading + no OOMs.
fourbit_models = [
    "unsloth/mistral-7b-bnb-4bit",
    "unsloth/mistral-7b-instruct-v0.2-bnb-4bit",
    "unsloth/llama-2-7b-bnb-4bit",
    "unsloth/llama-2-13b-bnb-4bit",
    "unsloth/codellama-34b-bnb-4bit",
    "unsloth/tinyllama-bnb-4bit",
    "unsloth/gemma-7b-bnb-4bit", # New Google 6 trillion tokens model 2.5x
    ↪ faster!
    "unsloth/gemma-2b-bnb-4bit",
] # More models at https://huggingface.co/unsloth

# unsloth/codellama-7b-bnb-4bit

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/llama-2-7b-bnb-4bit", # Choose ANY! eg mistralai/
    ↪ Mistral-7B-Instruct-v0.2
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = "hf_...", # use one if using gated models like meta-llama/
    ↪ Llama-2-7b-hf
```

```
)
```

Unused kwargs: ['quant\_method']. These kwargs are not used in <class 'transformers.utils.quantization\_config.BitsAndBytesConfig'>.

```
==(====)== Unsloth: Fast Llama patching release 2024.3
\\  /| GPU: NVIDIA GeForce RTX 3090. Max memory: 23.483 GB. Platform =
Linux.
O^O/ \_/ \ Pytorch: 2.2.1+cu121. CUDA = 8.6. CUDA Toolkit = 12.1.
\      / Bfloat16 = TRUE. Xformers = 0.0.25. FA = True.
"-_____" Free Apache license: http://github.com/unslothai/unsloth
```

```
[ ]: model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                     "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    use_gradient_checkpointing = True,
    random_state = 3407,
    use_rslora = False, # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

```
[ ]: # @title GSM8K Prompts

PREAMBLE = """As an expert problem solver solve step by step the following
↳mathematical questions."""

# The default gsm8k prompt from the CoT paper
# https://arxiv.org/pdf/2201.11903.pdf page 35.

PROMPT = """Q: There are 15 trees in the grove. Grove workers will plant trees
↳in the grove today. After they are done, there will be 21 trees. How many
↳trees did the grove workers plant today?
A: We start with 15 trees. Later we have 21 trees. The difference must be the
↳number of trees they planted. So, they must have planted 21 - 15 = 6 trees.
↳The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars
↳are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3
↳+ 2 = 5 cars. The answer is 5.
```

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many  
pieces do they have left in total?

A: Leah had 32 chocolates and Leah's sister had 42. That means there were  
originally  $32 + 42 = 74$  chocolates. 35 have been eaten. So in total they  
still have  $74 - 35 = 39$  chocolates. The answer is 39.

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12  
lollipops. How many lollipops did Jason give to Denny?

A: Jason had 20 lollipops. Since he only has 12 now, he must have given the  
rest to Denny. The number of lollipops he has given to Denny must have been  
 $20 - 12 = 8$  lollipops. The answer is 8.

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and  
dad. How many toys does he have now?

A: He has 5 toys. He got 2 from mom, so after that he has  $5 + 2 = 7$  toys. Then  
he got 2 more from dad, so in total he has  $7 + 2 = 9$  toys. The answer is 9.

Q: There were nine computers in the server room. Five more computers were  
installed each day, from monday to thursday. How many computers are now in  
the server room?

A: There are 4 days from monday to thursday. 5 computers were added each day.  
That means in total  $4 * 5 = 20$  computers were added. There were 9 computers  
in the beginning, so now there are  $9 + 20 = 29$  computers. The answer is 29.

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday,  
he lost 2 more. How many golf balls did he have at the end of wednesday?

A: Michael initially had 58 balls. He lost 23 on Tuesday, so after that he has  
 $58 - 23 = 35$  balls. On Wednesday he lost 2 more so now he has  $35 - 2 = 33$   
balls. The answer is 33.

Q: Olivia has \$23. She bought five bagels for \$3 each. How much money does she  
have left?

A: She bought 5 bagels for \$3 each. This means she spent  $5 * \$3 = \$15$  on the  
bagels. She had \$23 in beginning, so now she has  $\$23 - \$15 = \$8$ . The answer  
is 8."

*# Extension of the default 8-shot prompt, page 35 in  
# <https://arxiv.org/pdf/2201.11903.pdf>  
# The extension is intended to improve performance on  
# more complicated gsm8k examples.*

EXTRA\_3\_SHOTS = ""As an expert problem solver solve step by step the following  
mathematical questions.

Q: Tina makes \$18.00 an hour. If she works more than 8 hours per shift, she is eligible for overtime, which is paid by your hourly wage + 1/2 your hourly wage. If she works 10 hours every day for 5 days, how much money does she make?

A: Here's how to calculate Tina's earnings:

**\*\*Regular Time:\*\***

- Hours per shift: 8 hours
- Wage per hour: \$18.00
- Regular pay per shift: 8 hours \* \$18.00/hour = \$144.00

**\*\*Overtime:\*\***

- Overtime hours per shift: 10 hours - 8 hours = 2 hours
- Overtime pay per hour: \$18.00 + (\$18.00 / 2) = \$27.00
- Overtime pay per shift: 2 hours \* \$27.00/hour = \$54.00

**\*\*Total per day:\*\***

- Regular pay + overtime pay: \$144.00/shift + \$54.00/shift = \$198.00/day

**\*\*Total for 5 days:\*\***

- 5 days \* \$198.00/day = \$990.00

**\*\*Therefore, Tina will make \$990.00 in 5 days.\*\*** The answer is 990.

Q: Abigail is trying a new recipe for a cold drink. It uses 1/4 of a cup of iced tea and 1 and 1/4 of a cup of lemonade to make one drink. If she fills a pitcher with 18 total cups of this drink, how many cups of lemonade are in the pitcher?

A: ## Ambiguity in the Problem Statement:

There is one main ambiguity in the problem statement:

**\*\*Total volume vs. Number of servings:\*\*** The statement "18 total cups of this drink" could be interpreted in two ways:

- \* **\*\*18 cups of the combined volume:\*\*** This would mean Abigail used a total of 18 cups of liquid, including both iced tea and lemonade.
- \* **\*\*18 individual servings:\*\*** This would mean Abigail made 18 individual drinks, each containing 1/4 cup of iced tea and 1 1/4 cup of lemonade.

Let us assume the interpretation "18 cups of the combined volume".

## Solution assuming 18 cups of combined volume:

**\*\*Step 1: Find the proportion of lemonade in one drink:\*\***

- \* Lemonade: 1 1/4 cups
- \* Iced tea: 1/4 cup

```
* Total:  $1 \frac{1}{4} + \frac{1}{4} = 1 \frac{1}{2}$  cups
* Lemonade proportion:  $(1 \frac{1}{4}) / (1 \frac{1}{2}) = \frac{5}{6}$ 
```

**\*\*Step 2: Calculate the amount of lemonade in the pitcher:\*\***

```
* Total volume: 18 cups
* Lemonade proportion:  $\frac{5}{6}$ 
* Volume of lemonade:  $18 * (\frac{5}{6}) = 15$  cups
```

Therefore, there are 15 cups of lemonade in the pitcher. The answer is 15.

Q: A deep-sea monster rises from the waters once every hundred years to feast  
↳ on a ship and sate its hunger. Over three hundred years, it has consumed 847  
↳ people. Ships have been built larger over time, so each new ship has twice  
↳ as many people as the last ship. How many people were on the ship the  
↳ monster ate in the first hundred years?

A: Let us solve it using algebra. Let  $x$  be the number of people on the ship the  
↳ monster ate in the first hundred years.

The number of people on the ship eaten in the second hundred years is  $2x$ , and  
↳ in the third hundred years is  $4x$ .

Therefore, the total number of people eaten over three hundred years is  $x + 2x$   
↳  $+ 4x = 847$ .

Combining like terms, we get  $7x = 847$ .

Dividing both sides by 7, we find  $x = 121$ .

Therefore, there were 121 people on the ship the monster ate in the first  
↳ hundred years. The answer is 121.""""

```
[ ]: import re

def extract_number_from_text(text, prefix="The answer is"):
    """
    Extracts the last number from a text string that follows a given prefix.
    Args:
        text (str): The text from which to extract the number.
        prefix (str): The prefix to search for before extracting the number.
    Returns:
        float or None: The extracted number, or None if no valid number is
        ↳ found.
    """
    # Find the part of the text that starts with the prefix
    match = re.search(re.escape(prefix) + r".*", text)
    if match:
```

```

    # Extract all numbers from the matched text
    numbers = re.findall(r"[-+]?[0-9]*\.?[0-9]+", match.group(0))
    if numbers:
        # Return the last number found as a float
        last_number = numbers[-1]
        try:
            return float(last_number)
        except ValueError:
            print(f"Could not convert '{last_number}' to float.")
            return None
    return None

def extract_response_after_question(full_output, question):
    """
    Extracts the line immediately following the question in the model's output.

    Args:
    - full_output (str): The complete output from the model.
    - question (str): The question text used to locate the response line.

    Returns:
    - str: The line following the question line or None if not found.
    """
    # Normalize line breaks
    full_output = full_output.replace('\r\n', '\n').replace('\r', '\n')
    lines = full_output.split('\n')

    # Attempt to find the line containing the question
    for i, line in enumerate(lines):
        if question in line:
            # Return the next line if it exists
            if i + 1 < len(lines):
                return lines[i + 1].strip()
            break

    return None

## Example Usage:
# full_output = ""
# Q: How many eggs do Janet's ducks lay?
# A: Janet's ducks lay 16 eggs per day. She eats 3 for breakfast.
# She bakes muffins with 4. She sells the rest for $2 per fresh duck egg.
# So, she gets 16 * 3 - 4 * 2 = $48. The answer is $48.
# ""
# question = "How many eggs do Janet's ducks lay?"

# next_line = extract_response_after_question(full_output, question)

```

```
# print(f"Response after the question: '{next_line}'")
```

```
[ ]: import torch
from datasets import load_dataset

TEMPLATE = """
Q: {}
A: """

# Load GSM8K dataset
gsm8k_test = load_dataset("gsm8k", "main", split="test")

# Assuming model and tokenizer are already initialized
model.eval() # Set the model to evaluation mode

# Helper function to encode inputs
def prepare_input(p):
    # print(question)
    prompt = (PREAMBLE + '\n\n' + PROMPT + '\n' +
              TEMPLATE.format(p))
    return tokenizer(prompt, return_tensors='pt').input_ids

# Function to decode model output
def decode_output(output_ids):
    return tokenizer.decode(output_ids, skip_special_tokens=True)

# Manual testing loop
all_correct = 0
all_responses = {}
idx = 0
total = len(gsm8k_test)
total = 100

for task_id, problem in enumerate(gsm8k_test):
    if idx == total:
        break

    print(f"task_id {task_id}")

    # Prepare the input for the model
    input_ids = prepare_input(problem['question'])
    with torch.no_grad():
        output_ids = model.generate(input_ids, max_new_tokens = 120) # Adjust
        ↪ max_length as needed

    response = decode_output(output_ids[0])
```

```

all_responses[task_id] = response

answer_line = extract_response_after_question(response, problem['question'])

# Compare model output to the ground truth
model_number = extract_number_from_text(answer_line, "The answer is")
ground_truth_number = extract_number_from_text(problem['answer'], "####")

# print(model_number)
# print(ground_truth_number)

if model_number == ground_truth_number:
    all_correct += 1

print(f"Model answer: {model_number}")
print(f"Ground truth answer: {ground_truth_number}")
print(f"Correct: {all_correct} out of {total}")
print("="*40)
idx += 1

# Final accuracy
accuracy = all_correct / len(gsm8k_test)
print(f"Final Accuracy: {accuracy:.2f}")

```

```

task_id 0
Model answer: None
Ground truth answer: 18.0
Correct: 0 out of 100
=====
task_id 1
Model answer: 2.5
Ground truth answer: 3.0
Correct: 0 out of 100
=====
task_id 2
Model answer: 0.0
Ground truth answer: 70000.0
Correct: 0 out of 100
=====
task_id 3
Model answer: 180.0
Ground truth answer: 540.0
Correct: 0 out of 100
=====
task_id 4

```



Model answer: 40.0  
Ground truth answer: 20.0  
Correct: 0 out of 100  
=====

task\_id 5  
Model answer: None  
Ground truth answer: 64.0  
Correct: 0 out of 100  
=====

task\_id 6  
Model answer: 160.0  
Ground truth answer: 260.0  
Correct: 0 out of 100  
=====

task\_id 7  
Model answer: None  
Ground truth answer: 160.0  
Correct: 0 out of 100  
=====

task\_id 8  
Model answer: None  
Ground truth answer: 45.0  
Correct: 0 out of 100  
=====

task\_id 9  
Model answer: 450.0  
Ground truth answer: 460.0  
Correct: 0 out of 100  
=====

task\_id 10  
Model answer: None  
Ground truth answer: 366.0  
Correct: 0 out of 100  
=====

task\_id 11  
Model answer: 405.0  
Ground truth answer: 694.0  
Correct: 0 out of 100  
=====

task\_id 12  
Model answer: None  
Ground truth answer: 13.0  
Correct: 0 out of 100  
=====

task\_id 13  
Model answer: 2.0  
Ground truth answer: 18.0  
Correct: 0 out of 100

```

=====
task_id 14
Model answer: None
Ground truth answer: 60.0
Correct: 0 out of 100
=====
task_id 15
Model answer: None
Ground truth answer: 125.0
Correct: 0 out of 100
=====
task_id 16
Model answer: 230.0
Ground truth answer: 230.0
Correct: 1 out of 100
=====
task_id 17
Model answer: None
Ground truth answer: 57500.0
Correct: 1 out of 100
=====
task_id 18
Model answer: 48.0
Ground truth answer: 7.0
Correct: 1 out of 100
=====
task_id 19
Model answer: 3.0
Ground truth answer: 6.0
Correct: 1 out of 100
=====
task_id 20
Model answer: 16.0
Ground truth answer: 15.0
Correct: 1 out of 100
=====
task_id 21
Model answer: 29.0
Ground truth answer: 14.0
Correct: 1 out of 100
=====
task_id 22
Model answer: 7.0
Ground truth answer: 7.0
Correct: 2 out of 100
=====
task_id 23
Model answer: 8.0

```

```

Ground truth answer: 8.0
Correct: 3 out of 100
=====
task_id 24
Model answer: 19.125
Ground truth answer: 26.0
Correct: 3 out of 100
=====
task_id 25
Model answer: 1.0
Ground truth answer: 2.0
Correct: 3 out of 100
=====
task_id 26
Model answer: None
Ground truth answer: 243.0
Correct: 3 out of 100
=====
task_id 27
Model answer: 3600.0
Ground truth answer: 16.0
Correct: 3 out of 100
=====
task_id 28
Model answer: 35.0
Ground truth answer: 25.0
Correct: 3 out of 100
=====
task_id 29
Model answer: 10.0
Ground truth answer: 104.0
Correct: 3 out of 100
=====
task_id 30
Model answer: 11.0
Ground truth answer: 109.0
Correct: 3 out of 100
=====
task_id 31
Model answer: 100.0
Ground truth answer: 80.0
Correct: 3 out of 100
=====
task_id 32
Model answer: 35.0
Ground truth answer: 35.0
Correct: 4 out of 100
=====

```

```

task_id 33
Model answer: 80.0
Ground truth answer: 70.0
Correct: 4 out of 100
=====
task_id 34
Model answer: 36.0
Ground truth answer: 23.0
Correct: 4 out of 100
=====
task_id 35
Model answer: 9.0
Ground truth answer: 9.0
Correct: 5 out of 100
=====
task_id 36
Model answer: 300.0
Ground truth answer: 75.0
Correct: 5 out of 100
=====
task_id 37
Model answer: 2.0
Ground truth answer: 2.0
Correct: 6 out of 100
=====
task_id 38
Model answer: None
Ground truth answer: 10.0
Correct: 6 out of 100
=====
task_id 39
Model answer: None
Ground truth answer: 18.0
Correct: 6 out of 100
=====
task_id 40
Model answer: 8.0
Ground truth answer: 8.0
Correct: 7 out of 100
=====
task_id 41
Model answer: None
Ground truth answer: 200.0
Correct: 7 out of 100
=====
task_id 42
Model answer: None
Ground truth answer: 26.0

```

```

Correct: 7 out of 100
=====
task_id 43
Model answer: None
Ground truth answer: 48.0
Correct: 7 out of 100
=====
task_id 44
Model answer: None
Ground truth answer: 20.0
Correct: 7 out of 100
=====
task_id 45
Model answer: 17.0
Ground truth answer: 104.0
Correct: 7 out of 100
=====
task_id 46
Model answer: 103.0
Ground truth answer: 163.0
Correct: 7 out of 100
=====
task_id 47
Model answer: 120.0
Ground truth answer: 800.0
Correct: 7 out of 100
=====
task_id 48
Model answer: 24.0
Ground truth answer: 8.0
Correct: 7 out of 100
=====
task_id 49
Model answer: 3.75
Ground truth answer: 30.0
Correct: 7 out of 100
=====
task_id 50
Model answer: None
Ground truth answer: 294.0
Correct: 7 out of 100
=====
task_id 51
Model answer: 10.0
Ground truth answer: 5.0
Correct: 7 out of 100
=====
task_id 52

```

```

Model answer: None
Ground truth answer: 15.0
Correct: 7 out of 100
=====
task_id 53
Model answer: None
Ground truth answer: 40.0
Correct: 7 out of 100
=====
task_id 54
Model answer: 28.0
Ground truth answer: 40.0
Correct: 7 out of 100
=====
task_id 55
Model answer: 14.0
Ground truth answer: 14.0
Correct: 8 out of 100
=====
task_id 56
Model answer: 42.0
Ground truth answer: 3.0
Correct: 8 out of 100
=====
task_id 57
Model answer: 11245.0
Ground truth answer: 83.0
Correct: 8 out of 100
=====
task_id 58
Model answer: None
Ground truth answer: 57.0
Correct: 8 out of 100
=====
task_id 59
Model answer: 1267.0
Ground truth answer: 187.0
Correct: 8 out of 100
=====
task_id 60
Model answer: 22.0
Ground truth answer: 17.0
Correct: 8 out of 100
=====
task_id 61
Model answer: 1380.0
Ground truth answer: 1430.0
Correct: 8 out of 100

```

```

=====
task_id 62
Model answer: 5000.0
Ground truth answer: 25000.0
Correct: 8 out of 100
=====
task_id 63
Model answer: None
Ground truth answer: 1596.0
Correct: 8 out of 100
=====
task_id 64
Model answer: 8.0
Ground truth answer: 300.0
Correct: 8 out of 100
=====
task_id 65
Model answer: 5.0
Ground truth answer: 36.0
Correct: 8 out of 100
=====
task_id 66
Model answer: 40.0
Ground truth answer: 48.0
Correct: 8 out of 100
=====
task_id 67
Model answer: 200.0
Ground truth answer: 595.0
Correct: 8 out of 100
=====
task_id 68
Model answer: 72.0
Ground truth answer: 36.0
Correct: 8 out of 100
=====
task_id 69
Model answer: 60.0
Ground truth answer: 60.0
Correct: 9 out of 100
=====
task_id 70
Model answer: 2925.0
Ground truth answer: 7425.0
Correct: 9 out of 100
=====
task_id 71
Model answer: 60.0

```

```

Ground truth answer: 60.0
Correct: 10 out of 100
=====
task_id 72
Model answer: 218.0
Ground truth answer: 221.0
Correct: 10 out of 100
=====
task_id 73
Model answer: 750.0
Ground truth answer: 255.0
Correct: 10 out of 100
=====
task_id 74
Model answer: None
Ground truth answer: 88.0
Correct: 10 out of 100
=====
task_id 75
Model answer: 0.25
Ground truth answer: 60.0
Correct: 10 out of 100
=====
task_id 76
Model answer: 185.0
Ground truth answer: 5.0
Correct: 10 out of 100
=====
task_id 77
Model answer: None
Ground truth answer: 100.0
Correct: 10 out of 100
=====
task_id 78
Model answer: 2.0
Ground truth answer: 6.0
Correct: 10 out of 100
=====
task_id 79
Model answer: 70.0
Ground truth answer: 70.0
Correct: 11 out of 100
=====
task_id 80
Model answer: 2.0
Ground truth answer: 10.0
Correct: 11 out of 100
=====

```



```
task_id 81
Model answer: -34.0
Ground truth answer: 17.0
Correct: 11 out of 100
=====
task_id 82
Model answer: 248.0
Ground truth answer: 623.0
Correct: 11 out of 100
=====
task_id 83
Model answer: 600.0
Ground truth answer: 600.0
Correct: 12 out of 100
=====
task_id 84
Model answer: 22.0
Ground truth answer: 15.0
Correct: 12 out of 100
=====
task_id 85
Model answer: 48.0
Ground truth answer: 44.0
Correct: 12 out of 100
=====
task_id 86
Model answer: None
Ground truth answer: 22.0
Correct: 12 out of 100
=====
task_id 87
Model answer: 738.0
Ground truth answer: 9360.0
Correct: 12 out of 100
=====
task_id 88
Model answer: 200.0
Ground truth answer: 8000.0
Correct: 12 out of 100
=====
task_id 89
Model answer: 8.0
Ground truth answer: 24.0
Correct: 12 out of 100
=====
task_id 90
Model answer: 200.0
Ground truth answer: 225.0
```

```

Correct: 12 out of 100
=====
task_id 91
Model answer: 15.0
Ground truth answer: 28.0
Correct: 12 out of 100
=====
task_id 92
Model answer: None
Ground truth answer: 4.0
Correct: 12 out of 100
=====
task_id 93
Model answer: 36.0
Ground truth answer: 36.0
Correct: 13 out of 100
=====
task_id 94
Model answer: 120.0
Ground truth answer: 348.0
Correct: 13 out of 100
=====
task_id 95
Model answer: 140.0
Ground truth answer: 40.0
Correct: 13 out of 100
=====
task_id 96
Model answer: 3.0
Ground truth answer: 3.0
Correct: 14 out of 100
=====
task_id 97
Model answer: 48.0
Ground truth answer: 12.0
Correct: 14 out of 100
=====
task_id 98
Model answer: 30.0
Ground truth answer: 5.0
Correct: 14 out of 100
=====
task_id 99
Model answer: 0.0
Ground truth answer: 58.0
Correct: 14 out of 100
=====
Final Accuracy: 0.01

```

```
[ ]: accuracy = all_correct / total* 100  
print(f"Final Accuracy Llama2 base : {accuracy:.2f}%")
```

Final Accuracy Llama2 base : 14.00%