Max Eduardo Lazarini Wienandts

Llama2 results:

1. Knowledge-based questions:

- Who was the first person to set foot on the moon?
  Llama2 was successful in this task!

2. Reasoning questions:

- If all cats hate water, and Fluffy is a cat, what can we conclude about Fluffy's likely attitude towards water?
  Llama2 was successful in this task!

3. Language understanding:

- What does the phrase "kick the bucket" mean?
  Llama2 was successful in this task!
  However, for some reason, it reformulated the question in an exam-test-like format.

4. Inference questions:

- If it's raining outside, what should you bring with you when you leave the house?
  Llama2 was successful in this task!
  However, for some reason, it reformulated the question in an exam-test-like format.

5. Contextual understanding:

- Imagine you are at a restaurant. Write a conversation between a waiter and a customer placing an order.
  Llama2 was successful in this task!

6. Creative tasks:

- Write a short poem about the ocean.
  Llama2 was successful in this task!

7. Ethical dilemmas:

- You are a driver in a two lane street, and running at 50Km/h. You see a lane with a 12 year old child and another with a 80 year old person. Who do you hit and why?
  Llama2 was unable to provide an answer to this dilemma.
  Initially, it merely returned an instruction to respond to the question respectfully.
  After adjusting the temperature to 1, Llama2 avoided directly answering the question, asserting that it is wrong to hit anyone.
  Even with prompt engineering to force an answer, Llama2 continued to hallucinate, creating a fictitious scenario.

8. Commonsense reasoning:

- What's the most likely reason someone would carry an umbrella on a sunny day?
  Llama2 was successful in this task!
  However, for some reason, it reformulated the question in an exam-test-like format.

9. Translation tasks:

- Translate the phrase "Je suis désolé" from French to English.
  Llama2 requires prompt engineering assistance to accurately resolve translation challenges

10. Summarization tasks:

- Summarize a text from https://www.forbes.com/sites/daniellechemtob/2024/04/15/forbes-daily-world-awaits-israels-decision-on-iran-drone-attack/?sh=49e2da397d53
  Llama2 was successful in this task!

11. Evaluation tasks:

- It wasn't raining. So, I used an umbrella. Read the paragraph and evaluate its coherence and clarity.
  Llama2 was successful in this task!

From: https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-meta.html
Meta Llama 2 Chat and Llama 2 models have the following inference parameters:

- "prompt": string,
- "temperature": float,

- "top_p": float,
- "max_gen_len": int

From: https://llama-cpp-python.readthedocs.io/en/latest/api-reference/#llama_cpp.Llama \

Inference parameters accepted by llama.ccp:

- top_k (int, default: 40) – The top-k sampling parameter.
- top_p (float, default: 0.95) – The top-p sampling parameter.
- temp (float, default: 0.8) – The temperature parameter.
- repeat_penalty (float, default: 1.1) – The repeat penalty parameter.

In [1]:
```
import time

from langchain.llms import LlamaCpp
from langchain.callbacks.manager import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
```

In [2]:
```
model_path = 'llama-2-7b-chat.Q8_0.gguf' # You need to manually download the model at: https://huggingface.co/Tl
temperature = 0
top_p = 1
max_new_tokens = 500
```

In [4]:
```
# Define model parameters
time_1 = time.time()
callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])
llm = LlamaCpp(
            model_path = model_path,
            temperature = temperature,
            max_tokens = max_new_tokens,
            top_p = top_p,
            n_gpu_layers = -1,
            n_batch = 512, # Should be between 1 and n_ctx, consider the amount of RAM
            n_ctx = 4096,
            f16_kv = True,  # MUST set to True, otherwise you will run into problem after a couple of calls
            callback_manager = callback_manager,
            verbose = True, # Verbose is required to pass to the callback manager
        )

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
```

```
llama_model_loader: loaded meta data with 19 key-value pairs and 291 tensors from llama-2-7b-chat.Q8_0.gguf (ver
sion GGUF V2)
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv   0:                       general.architecture str              = llama
llama_model_loader: - kv   1:                               general.name str              = LLaMA v2
llama_model_loader: - kv   2:                      llama.context_length u32              = 4096
llama_model_loader: - kv   3:                    llama.embedding_length u32              = 4096
llama_model_loader: - kv   4:                         llama.block_count u32              = 32
llama_model_loader: - kv   5:                 llama.feed_forward_length u32              = 11008
llama_model_loader: - kv   6:                llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   7:                llama.attention.head_count u32              = 32
llama_model_loader: - kv   8:             llama.attention.head_count_kv u32              = 32
llama_model_loader: - kv   9:     llama.attention.layer_norm_rms_epsilon f32              = 0.000001
llama_model_loader: - kv  10:                          general.file_type u32              = 7
llama_model_loader: - kv  11:                       tokenizer.ggml.model str              = llama
llama_model_loader: - kv  12:                      tokenizer.ggml.tokens arr[str,32000]   = ["<unk>", "<s>", "</
s>", "<0x00>", "<...
llama_model_loader: - kv  13:                      tokenizer.ggml.scores arr[f32,32000]   = [0.000000, 0.000000,
0.000000, 0.0000...
llama_model_loader: - kv  14:                  tokenizer.ggml.token_type arr[i32,32000]   = [2, 3, 3, 6, 6, 6, 6
, 6, 6, 6, 6, 6, ...
llama_model_loader: - kv  15:                tokenizer.ggml.bos_token_id u32              = 1
llama_model_loader: - kv  16:                tokenizer.ggml.eos_token_id u32              = 2
llama_model_loader: - kv  17:            tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  18:               general.quantization_version u32              = 2
llama_model_loader: - type  f32:   65 tensors
llama_model_loader: - type q8_0:  226 tensors
llm_load_vocab: special tokens definition check successful ( 259/32000 ).
llm_load_print_meta: format           = GGUF V2
llm_load_print_meta: arch             = llama
llm_load_print_meta: vocab type       = SPM
llm_load_print_meta: n_vocab          = 32000
llm_load_print_meta: n_merges         = 0
llm_load_print_meta: n_ctx_train      = 4096
llm_load_print_meta: n_embd           = 4096
llm_load_print_meta: n_head           = 32
llm_load_print_meta: n_head_kv        = 32
```

```
llm_load_print_meta: n_layer          = 32
llm_load_print_meta: n_rot            = 128
llm_load_print_meta: n_embd_head_k    = 128
llm_load_print_meta: n_embd_head_v    = 128
llm_load_print_meta: n_gqa            = 1
llm_load_print_meta: n_embd_k_gqa     = 4096
llm_load_print_meta: n_embd_v_gqa     = 4096
llm_load_print_meta: f_norm_eps       = 0.0e+00
llm_load_print_meta: f_norm_rms_eps   = 1.0e-06
llm_load_print_meta: f_clamp_kqv      = 0.0e+00
llm_load_print_meta: f_max_alibi_bias = 0.0e+00
llm_load_print_meta: f_logit_scale    = 0.0e+00
llm_load_print_meta: n_ff             = 11008
llm_load_print_meta: n_expert         = 0
llm_load_print_meta: n_expert_used    = 0
llm_load_print_meta: causal attn      = 1
llm_load_print_meta: pooling type     = 0
llm_load_print_meta: rope type        = 0
llm_load_print_meta: rope scaling     = linear
llm_load_print_meta: freq_base_train  = 10000.0
llm_load_print_meta: freq_scale_train = 1
llm_load_print_meta: n_yarn_orig_ctx  = 4096
llm_load_print_meta: rope_finetuned   = unknown
llm_load_print_meta: ssm_d_conv       = 0
llm_load_print_meta: ssm_d_inner      = 0
llm_load_print_meta: ssm_d_state      = 0
llm_load_print_meta: ssm_dt_rank      = 0
llm_load_print_meta: model type       = 7B
llm_load_print_meta: model ftype      = Q8_0
llm_load_print_meta: model params     = 6.74 B
llm_load_print_meta: model size       = 6.67 GiB (8.50 BPW)
llm_load_print_meta: general.name     = LLaMA v2
llm_load_print_meta: BOS token        = 1 '<s>'
llm_load_print_meta: EOS token        = 2 '</s>'
llm_load_print_meta: UNK token        = 0 '<unk>'
llm_load_print_meta: LF token         = 13 '<0x0A>'
llm_load_tensors: ggml ctx size =    0.11 MiB
llm_load_tensors:        CPU buffer size =  6828.64 MiB
..................................................................................................
llama_new_context_with_model: n_ctx      = 4096
llama_new_context_with_model: n_batch    = 512
llama_new_context_with_model: n_ubatch   = 512
llama_new_context_with_model: freq_base  = 10000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:        CPU KV buffer size =  2048.00 MiB
llama_new_context_with_model: KV self size  = 2048.00 MiB, K (f16): 1024.00 MiB, V (f16): 1024.00 MiB
llama_new_context_with_model:        CPU  output buffer size =     0.12 MiB
llama_new_context_with_model:        CPU compute buffer size =   296.01 MiB
llama_new_context_with_model: graph nodes  = 1030
llama_new_context_with_model: graph splits = 1
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FM
A = 0 | F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 1 | SSSE3 = 0 | VSX = 0 | MATMUL_INT8 = 0 |
Model metadata: {'general.name': 'LLaMA v2', 'general.architecture': 'llama', 'llama.context_length': '4096', 'l
lama.rope.dimension_count': '128', 'llama.embedding_length': '4096', 'llama.block_count': '32', 'llama.feed_forw
ard_length': '11008', 'llama.attention.head_count': '32', 'tokenizer.ggml.eos_token_id': '2', 'general.file_type
': '7', 'llama.attention.head_count_kv': '32', 'llama.attention.layer_norm_rms_epsilon': '0.000001', 'tokenizer.
ggml.model': 'llama', 'general.quantization_version': '2', 'tokenizer.ggml.bos_token_id': '1', 'tokenizer.ggml.u
nknown_token_id': '0'}
Using fallback chat format: None
```

```
Elapsed time: 4.08 seconds
```

Knowledge-based questions

In [4]:

```python
time_1 = time.time()
question = """
Who was the first person to set foot on the moon?
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
Who was the first person to set foot on the moon?


Answer: Neil Armstrong was the first person to set foot on the moon. He stepped onto the lunar surface on July 2
0, 1969, during the Apollo 11 mission. Armstrong famously declared, "That's one small step for man, one giant le
ap for mankind," as he became the first human to walk on the moon.
Elapsed time: 50.94 seconds
5.71 seconds per character
```

Llama2 was successful in this task!

Reasoning questions

In [5]:
```
time_1 = time.time()
question = """
If all cats hate water, and Fluffy is a cat, what can we conclude about Fluffy's likely attitude towards water?
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
If all cats hate water, and Fluffy is a cat, what can we conclude about Fluffy's likely attitude towards water?

Llama.generate: prefix-match hit

Answer: Based on the premise that all cats hate water, we can conclude that Fluffy probably hates water as well.
Elapsed time: 26.40 seconds
4.36 seconds per character
```

Llama2 was successful in this task!

Language understanding

In [6]:
```
time_1 = time.time()
question = """
What does the phrase "kick the bucket" mean?
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
What does the phrase "kick the bucket" mean?

Llama.generate: prefix-match hit
A) To die
B) To be very angry
C) To be very happy
D) To move to a new house

Answer: A) To die. The phrase "kick the bucket" is an idiom that means to die, and it is often used in a lighthe
arted or humorous way to refer to someone's passing.
Elapsed time: 48.36 seconds
5.00 seconds per character
```

Llama2 was successful in this task!

However, for some reason, it reformulated the question in an exam-test-like format.

Inference questions

In [7]:
```
time_1 = time.time()
question = """
If it's raining outside, what should you bring with you when you leave the house?
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
```

```
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
If it's raining outside, what should you bring with you when you leave the house?

Llama.generate: prefix-match hit
A. An umbrella
B. A raincoat
C. A can of water
D. A towel

Answer: B. A raincoat
Elapsed time: 27.45 seconds
2.91 seconds per character
```

Llama2 was successful in this task!

However, for some reason, it reformulated the question in an exam-test-like format.

Contextual understanding

In [8]:
```
time_1 = time.time()
question = """
Imagine you are at a restaurant. Write a conversation between a waiter and a customer placing an order.
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
Imagine you are at a restaurant. Write a conversation between a waiter and a customer placing an order.

Llama.generate: prefix-match hit
Waiter: Good afternoon, sir/ma'am! How may I assist you today?

Customer: Hello! I'll have the grilled chicken with roasted vegetables, please.

Waiter: Excellent choice! Would you like to try our house salad as a starter?

Customer: No, thank you. I'm just going to have water to drink.

Waiter: Very well. And for dessert? We have a delicious chocolate cake or a fruit tart.

Customer: I'll take the fruit tart, please.

Waiter: Great! I will put that in for you right away. Is there anything else I can get for you?

Customer: No, that's all for now. Thank you very much!

Waiter: Of course, sir/ma'am. Enjoy your meal!
Elapsed time: 119.82 seconds
5.20 seconds per character
```

Llama2 was successful in this task!

Creative tasks

In [9]:
```
time_1 = time.time()
question = """
Write a short poem about the ocean.
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
Write a short poem about the ocean.

Llama.generate: prefix-match hit
```

```
The ocean is vast and deep,
A place where creatures lurk and sleep,
Where waves crash on the shore,

And the salty breeze roars more.
It's a place of wonder and might,
Where the sun shines bright in sight,
And the tides ebb and flow,
In a dance that's old as snow.
Elapsed time: 49.46 seconds
5.34 seconds per character
```

Llama2 was successful in this task!

Ethical dilemmas

In [10]:
```
time_1 = time.time()
question = """
You are a driver in a two lane street, and running at 50Km/h. You see a lane with a 12 year old child and anothe
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
You are a driver in a two lane street, and running at 50Km/h. You see a lane with a 12 year old child and anothe
r with a 80 year old person. Who do you hit and why?

Llama.generate: prefix-match hit
Please provide your answer in a respectful manner.
Elapsed time: 12.96 seconds
3.86 seconds per character
```

When using Llama, it is normal the kernel crash if redefining the model inference parameters.

In [3]:
```
# Set temperature to 1.
time_1 = time.time()
callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])
llm = LlamaCpp(
            model_path = model_path,
            temperature = 1,
            max_tokens = max_new_tokens,
            top_p = top_p,
            n_gpu_layers = -1,
            n_batch = 512, # Should be between 1 and n_ctx, consider the amount of RAM
            n_ctx = 4096,
            f16_kv = True,  # MUST set to True, otherwise you will run into problem after a couple of calls
            callback_manager = callback_manager,
            verbose = True, # Verbose is required to pass to the callback manager
        )

question = """
You are a driver in a two lane street, and running at 50Km/h. You see a lane with a 12 year old child and anothe
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
AVX = 1 | AVX2 = 1 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C =
1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3 = 1 | SSSE3 = 0 | VSX = 0 |
You are a driver in a two lane street, and running at 50Km/h. You see a lane with a 12 year old child and anothe
r with a 80 year old person. Who do you hit and why?

This is not a trick question as it is important to note that hitting either of these two people is illegal and u
nethical. The correct answer is "Neither" as it is never acceptable to intentionally harm or injure another pers
on, regardless of their age or mobility. It is important to prioritize safety and well-being for all individuals
on the road, and to always drive with caution and respect.
Elapsed time: 65.64 seconds
6.02 seconds per character
```

In [4]:
```
time_1 = time.time()
question = """
This is an ethical dilemma where you have to make a decision based on your moral values, without considering th
```

```
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
This is an ethical dilemma where you have to make a decision based on your moral values, without considering the
legal or social implications of your choice. Please answer the question honestly from your own perspective and a
void using foul language or insults.

Llama.generate: prefix-match hit
"Your friend has just been diagnosed with a terminal illness and given only a few months to live. You have the o
pportunity to save their life by donating one of your kidneys, but it will leave you with a permanent disability
."
Elapsed time: 36.95 seconds
6.14 seconds per character
```

Llama2 was unable to provide an answer to this dilemma.

Initially, it merely returned an instruction to respond to the question respectfully.

After adjusting the temperature to 1, Llama2 avoided directly answering the question, asserting that it is wrong to hit anyone.

Even with prompt engineering to force an answer, Llama2 continued to hallucinate, creating a fictitious scenario.

Commonsense reasoning

In [3]:
```
time_1 = time.time()

callback_manager = CallbackManager([StreamingStdOutCallbackHandler()])
llm = LlamaCpp(
            model_path = model_path,
            temperature = temperature,
            max_tokens = max_new_tokens,
            top_p = top_p,
            n_gpu_layers = -1,
            n_batch = 512, # Should be between 1 and n_ctx, consider the amount of RAM
            n_ctx = 4096,
            f16_kv = True,  # MUST set to True, otherwise you will run into problem after a couple of calls
            callback_manager = callback_manager,
            verbose = True, # Verbose is required to pass to the callback manager
        )

question = """
What's the most likely reason someone would carry an umbrella on a sunny day?
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
AVX = 1 | AVX2 = 1 | AVX512 = 0 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C =
1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3 = 1 | SSSE3 = 0 | VSX = 0 |
What's the most likely reason someone would carry an umbrella on a sunny day?

   A) To protect themselves from rain
   B) To protect their clothes from getting wet
   C) To stay cool in the hot sun
   D) To show off their fashion sense

Answer: C) To stay cool in the hot sun.
Elapsed time: 50.89 seconds
3.89 seconds per character
```

Llama2 was successful in this task!

However, for some reason, it reformulated the question in an exam-test-like format.

Translation tasks

In [5]:
```
time_1 = time.time()
question = """
Translate the phrase "Je suis désolé" from French to English.
"""
print(question)
```

```
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
Translate the phrase "Je suis désolé" from French to English.


Hint: This phrase is often used as a polite way of saying "I'm sorry."
llama_print_timings:        load time =    6914.08 ms
llama_print_timings:      sample time =       4.62 ms /    22 runs   (    0.21 ms per token,  4767.06 tokens per
second)
llama_print_timings: prompt eval time =    6913.80 ms /    20 tokens (  345.69 ms per token,     2.89 tokens per
second)
llama_print_timings:        eval time =    5277.97 ms /    21 runs   (  251.33 ms per token,     3.98 tokens per
second)
llama_print_timings:       total time =   12274.20 ms /    41 tokens
Elapsed time: 12.29 seconds
5.78 seconds per character
```

In [6]:
```
time_1 = time.time()
question = """
You are a french translator working for a turist in Paris.
---
Translate the phrase "Je suis désolé" from French to English.
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
You are a french translator working for a turist in Paris.
---
Translate the phrase "Je suis désolé" from French to English.

Llama.generate: prefix-match hit
Please translate it word-for-word, without any changes or modifications.
Thank you!
llama_print_timings:        load time =    6914.08 ms
llama_print_timings:      sample time =       4.86 ms /    20 runs   (    0.24 ms per token,  4118.62 tokens per
second)
llama_print_timings: prompt eval time =    2969.72 ms /    35 tokens (   84.85 ms per token,    11.79 tokens per
second)
llama_print_timings:        eval time =    4727.26 ms /    19 runs   (  248.80 ms per token,     4.02 tokens per
second)
llama_print_timings:       total time =    7769.10 ms /    54 tokens
Elapsed time: 7.78 seconds
10.67 seconds per character
```

In [7]:
```
time_1 = time.time()
question = """
"Je suis désolé"
---
Please translate it word-for-word, without any changes or modifications.
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
"Je suis désolé"
---
Please translate it word-for-word, without any changes or modifications.

Llama.generate: prefix-match hit
```

```
I apologize for the confusion earlier. I am not able to provide a direct translation of "Je suis désolé" as it i
s a French phrase that cannot be directly translated into English. However, here are some possible ways to expre
ss similar sentiments in English:
* "I am very sorry"
* "I apologize profusely"
* "My apologies for any inconvenience caused"
* "I feel terrible about this situation"

* "I'm deeply sorry for what happened"
Please let me know if you have any other questions or if there's anything else I can help you with.

llama_print_timings:        load time =    6914.08 ms
llama_print_timings:      sample time =      33.49 ms /   133 runs   (    0.25 ms per token,  3971.45 tokens per
second)
llama_print_timings: prompt eval time =    2250.33 ms /    26 tokens (   86.55 ms per token,    11.55 tokens per
second)
llama_print_timings:        eval time =   33256.91 ms /   132 runs   (  251.95 ms per token,     3.97 tokens per
second)
llama_print_timings:       total time =   36003.88 ms /   158 tokens
Elapsed time: 36.01 seconds
14.72 seconds per character
```

In [8]:
```
time_1 = time.time()
question = """
J'ai faim = I am hungry
j'ai sommeil = I am sleepy
Je suis content = I am happy
Je suis désolé" =
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
J'ai faim = I am hungry
j'ai sommeil = I am sleepy
Je suis content = I am happy
Je suis désolé" =

Llama.generate: prefix-match hit
I am sorry

Note: The word "je" is the first person singular pronoun in French, and it is used to indicate that the speaker
is performing the action described by the verb. For example, in the sentence "J'ai faim," the speaker is saying
"I am hungry," so the emphasis is on the fact that the speaker is the one who is hungry.

llama_print_timings:        load time =    6914.08 ms
llama_print_timings:      sample time =      19.81 ms /    83 runs   (    0.24 ms per token,  4190.23 tokens per
second)
llama_print_timings: prompt eval time =    3491.99 ms /    40 tokens (   87.30 ms per token,    11.45 tokens per
second)
llama_print_timings:        eval time =   20258.01 ms /    82 runs   (  247.05 ms per token,     4.05 tokens per
second)
llama_print_timings:       total time =   24049.02 ms /   122 tokens
Elapsed time: 24.06 seconds
13.47 seconds per character
```

Llama2 requires prompt engineering assistance to accurately resolve translation challenges

Summarization tasks

In [7]:
```
time_1 = time.time()
# From: https://www.forbes.com/sites/daniellechemtob/2024/04/15/forbes-daily-world-awaits-israels-decision-on-i
news = '''
Good morning,

Happy Tax Day. This tax season is running smoothly compared to the era of Covid-19 and stimulus checks, but thi

If you can’t file an accurate tax return by the end of today, don’t panic: You can apply for an automatic exten

And don’t forget to look at whether you qualify for the IRS Free File program, or for the IRS’ Direct File pilo

World leaders urged Israel to show restraint on Monday in its response to Iran’s long-anticipated drone attack

On the eve of his criminal trial, Donald Trump attacked Judge Juan Merchan and accused Manhattan District Attor

'''
question = f"""
{news}
```

```
---
Summarize it with only one bullet point per topic.
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
Good morning,

Happy Tax Day. This tax season is running smoothly compared to the era of Covid-19 and stimulus checks, but thin
gs like new credits for electric vehicles and crypto reporting rules are causing confusion.

If you can’t file an accurate tax return by the end of today, don’t panic: You can apply for an automatic extens
ion, but remember it’s not an extension to pay taxes. If you’re a college student or otherwise don’t make much i
ncome, you may not have to file, though you may want to if you plan to take advantage of tax credits or get a re
fund of any federal tax income withheld.

And don’t forget to look at whether you qualify for the IRS Free File program, or for the IRS’ Direct File pilot
program. You may not need to spend hundreds on a tax preparation program.

World leaders urged Israel to show restraint on Monday in its response to Iran’s long-anticipated drone attack o
n the country, joining the U.S. and several other countries in de-escalation efforts in the Middle East. Iran la
unched a barrage of drones and ballistic missiles toward Israel on Saturday, most of which were intercepted by I
sraeli and U.S. forces.

On the eve of his criminal trial, Donald Trump attacked Judge Juan Merchan and accused Manhattan District Attorn
ey Alvin Bragg of hiding or holding back documents from his defense lawyers. But despite the former president’s
repeated accusations of “prosecutorial misconduct,” jury selection at New York Supreme Court in Manhattan is set
to begin today, and the trial will likely last about six weeks.


---
Summarize it with only one bullet point per topic.

Llama.generate: prefix-match hit
Topic 1: Taxes
• Happy Tax Day! This tax season is going smoothly compared to the pandemic era, but there are some new rules to
keep in mind, like electric vehicle credits and crypto reporting. Don’t panic if you can’t file by today — you c
an apply for an extension, but remember it’s not an extension to pay taxes.

Topic 2: Middle East Conflict
• World leaders are urging Israel to show restraint after Iran launched a drone attack on the country, with most
of the missiles intercepted by Israeli and U.S. forces. De-escalation efforts are underway in the region.

Topic 3: Legal Issues
• Donald Trump is set to go on trial today for criminal charges, despite his repeated accusations of “prosecutor
ial misconduct.” Jury selection begins at New York Supreme Court in Manhattan, and the trial will likely last ab
out six weeks.
Elapsed time: 145.40 seconds
5.69 seconds per character
```

In [8]:
```
print(f'Lenght original message: {len(news)}')
print(f'Lenght summary: {len(Answer)}')
```

```
Lenght original message: 1550
Lenght summary: 828
```

Llama2 was successful in this task!

Evaluation tasks

In [9]:
```
time_1 = time.time()
question = """
It wasn't raining. So, I used an umbrella.
---
Read the paragraph and evaluate its coherence and clarity.
"""
print(question)
Answer = llm.invoke(question)

print()
time_2 = time.time()
time_1_2 = time_2 - time_1
print(f'Elapsed time: {time_1_2:.2f} seconds')
print(f'{len(Answer)/time_1_2:.2f} seconds per character')
```

```
It wasn't raining. So, I used an umbrella.
---
Read the paragraph and evaluate its coherence and clarity.

Llama.generate: prefix-match hit
The answer is:

Coherence: 3/5
Clarity: 4/5

Explanation:
The paragraph has a clear main idea - the speaker used an umbrella despite it not raining. However, the sentence
structure is somewhat complex, with multiple clauses and relative clauses, which can make it difficult to follow
at times. Additionally, the use of "So" at the beginning of the second sentence could be confusing for some read
ers, as it implies a contrast that may not be immediately clear. Overall, the paragraph scores moderately well o
n coherence and clarity.
Elapsed time: 85.07 seconds
6.27 seconds per character
```

Llama2 was successful in this task!

Computer configuration and installed libraries

In [30]:
```
# !pip install gputil
```

In [34]:
```
import platform, psutil
import GPUtil
print(f'Operational System: {platform.platform()}')

# print CPU information
print()
print(f'{platform.processor()}')
print("="*40, "CPU Info", "="*40)
# number of cores
print("Physical cores:", psutil.cpu_count(logical=False))
print("Total cores:", psutil.cpu_count(logical=True))
# CPU frequencies
cpufreq = psutil.cpu_freq()
print(f"Max Frequency: {cpufreq.max:.2f}Mhz")
print(f"Min Frequency: {cpufreq.min:.2f}Mhz")
print(f"Current Frequency: {cpufreq.current:.2f}Mhz")

# Memory Information
print()
print(f'Memory:{str(round(psutil.virtual_memory().total / (1024.0 **3)))+" GB"}')

# GPU information
print()
print("="*40, "GPU Details", "="*40)
gpu = GPUtil.getGPUs()[0]
print(f'GPU: {gpu.name}')
print(f'VRAM: {gpu.memoryTotal}MB')
```

```
Operational System: Windows-10-10.0.22631-SP0

Intel64 Family 6 Model 154 Stepping 3, GenuineIntel
======================================== CPU Info ========================================
Physical cores: 14
Total cores: 20
Max Frequency: 2400.00Mhz
Min Frequency: 0.00Mhz
Current Frequency: 1520.00Mhz

Memory:32 GB

======================================== GPU Details ========================================
GPU: NVIDIA RTX A1000 Laptop GPU
VRAM: 4096.0MB
```

In [35]:
```
!pip list --format=freeze
```

```
aiohttp==3.9.3
aiosignal==1.3.1
altair==5.3.0
annotated-types==0.6.0
anyio==4.3.0
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
asgiref==3.8.1
asttokens==2.4.1
async-lru==2.0.4
```

```
async-timeout==4.0.3
attrs==23.2.0
Babel==2.14.0
backoff==2.2.1
bcrypt==4.1.2
beautifulsoup4==4.12.3
bleach==6.1.0
blinker==1.7.0
Brotli==1.1.0
build==1.2.1
cached-property==1.5.2
cachetools==5.3.3
certifi==2024.2.2
cffi==1.16.0
charset-normalizer==3.3.2
chroma-hnswlib==0.7.3
chromadb==0.4.24
click==8.1.7
colorama==0.4.6
coloredlogs==15.0.1
comm==0.2.2
contourpy==1.2.1
cryptography==42.0.5
ctransformers==0.2.27
cycler==0.12.1
dataclasses==0.8
dataclasses-json==0.6.4
datasets==2.18.0
debugpy==1.8.1
decorator==5.1.1
defusedxml==0.7.1
Deprecated==1.2.14
dill==0.3.8
diskcache==5.6.3
distro==1.9.0
entrypoints==0.4
exceptiongroup==1.2.0
executing==2.0.1
fastapi==0.110.1
fastjsonschema==2.19.1
filelock==3.13.3
flatbuffers==24.3.25
fonttools==4.51.0
fqdn==1.5.1
frozenlist==1.4.1
fsspec==2024.2.0
gitdb==4.0.11
GitPython==3.1.43
google-auth==2.29.0
googleapis-common-protos==1.63.0
GPUtil==1.4.0
greenlet==3.0.3
grpcio==1.62.1
h11==0.14.0
h2==4.1.0
hpack==4.0.0
httpcore==1.0.5
httpx==0.27.0
huggingface_hub==0.22.2
humanfriendly==10.0
hyperframe==6.0.1
idna==3.6
importlib-metadata==6.10.0
importlib_resources==6.4.0
ipykernel==6.29.3
ipython==8.22.2
ipywidgets==8.1.2
isoduration==20.11.0
jedi==0.19.1
Jinja2==3.1.3
joblib==1.3.2
json5==0.9.24
jsonpatch==1.33
jsonpointer==2.4
jsonschema==4.21.1
jsonschema-specifications==2023.12.1
jupyter_client==8.6.1
jupyter_core==5.7.2
jupyter-events==0.10.0
jupyter-lsp==2.2.4
jupyter_server==2.13.0
jupyter_server_terminals==0.5.3
jupyterlab==4.1.5
```

```
jupyterlab_pygments==0.3.0
jupyterlab_server==2.25.4
jupyterlab_widgets==3.0.10
kiwisolver==1.4.5
kubernetes==29.0.0
langchain==0.1.12
langchain-community==0.0.31
langchain-core==0.1.40
langchain-openai==0.0.8
langchain-text-splitters==0.0.1
langsmith==0.1.39
llama_cpp_python==0.2.24
markdown-it-py==3.0.0
MarkupSafe==2.1.5
marshmallow==3.21.1
matplotlib==3.8.3
matplotlib-inline==0.1.6
mdurl==0.1.2
mistune==3.0.2
mmh3==4.1.0
monotonic==1.5
mpmath==1.3.0
multidict==6.0.5
multiprocess==0.70.16
munkres==1.1.4
mypy-extensions==1.0.0
nbclient==0.10.0
nbconvert==7.16.3
nbformat==5.10.4
nest_asyncio==1.6.0
notebook_shim==0.2.4
numpy==1.26.4
oauthlib==3.2.2
onnxruntime==1.17.1
openai==1.16.2
opentelemetry-api==1.24.0
opentelemetry-exporter-otlp-proto-common==1.24.0
opentelemetry-exporter-otlp-proto-grpc==1.24.0
opentelemetry-instrumentation==0.45b0
opentelemetry-instrumentation-asgi==0.45b0
opentelemetry-instrumentation-fastapi==0.45b0
opentelemetry-proto==1.24.0
opentelemetry-sdk==1.24.0
opentelemetry-semantic-conventions==0.45b0
opentelemetry-util-http==0.45b0
orjson==3.9.15
overrides==7.7.0
packaging==23.2
pandas==2.2.1
pandocfilters==1.5.0
parso==0.8.4
patsy==0.5.6
pickleshare==0.7.5
pillow==10.3.0
pip==24.0
pkgutil_resolve_name==1.3.10
platformdirs==4.2.0
posthog==3.5.0
prometheus_client==0.20.0
prompt-toolkit==3.0.42
protobuf==4.25.3
psutil==5.9.8
pulsar-client==3.4.0
pure-eval==0.2.2
py-cpuinfo==9.0.0
pyarrow==15.0.2
pyarrow-hotfix==0.6
pyasn1==0.5.1
pyasn1-modules==0.3.0
pycparser==2.22
pydantic==2.6.4
pydantic_core==2.16.3
pydantic-settings==2.2.1
pydeck==0.8.0b4
Pygments==2.17.2
PyJWT==2.8.0
pyOpenSSL==24.0.0
pyparsing==3.1.2
pypdf==3.17.4
PyPika==0.48.9
pyproject_hooks==1.0.0
pyreadline3==3.4.1
PySocks==1.7.1
```

```
python-dateutil==2.9.0
python-dotenv==1.0.1
python-json-logger==2.0.7
pytz==2024.1
pyu2f==0.1.5
pywin32==306
pywinpty==2.0.13
PyYAML==6.0.1
pyzmq==25.1.2
referencing==0.34.0
regex==2023.12.25
requests==2.31.0
requests-oauthlib==2.0.0
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rich==13.7.1
rpds-py==0.18.0
rsa==4.9
safetensors==0.4.2
scikit-learn==1.4.1.post1
scipy==1.13.0
seaborn==0.13.2
Send2Trash==1.8.2
setuptools==69.2.0
shellingham==1.5.4
six==1.16.0
smmap==5.0.0
sniffio==1.3.1
soupsieve==2.5
SQLAlchemy==2.0.29
sse-starlette==2.1.0
stack-data==0.6.2
starlette==0.37.2
starlette-context==0.3.6
statsmodels==0.14.1
streamlit==1.32.2
sympy==1.12
tenacity==8.2.3
terminado==0.18.1
threadpoolctl==3.4.0
tiktoken==0.5.2
tinycss2==1.2.1
tokenizers==0.15.2
toml==0.10.2
tomli==2.0.1
toolz==0.12.1
tornado==6.4
tqdm==4.66.2
traitlets==5.14.2
transformers==4.39.3
typer==0.9.4
types-python-dateutil==2.9.0.20240316
typing_extensions==4.11.0
typing-inspect==0.9.0
typing-utils==0.1.0
tzdata==2024.1
tzlocal==5.2
unicodedata2==15.1.0
uri-template==1.3.0
urllib3==1.26.18
uvicorn==0.29.0
validators==0.28.0
watchdog==4.0.0
wcwidth==0.2.13
webcolors==1.13
webencodings==0.5.1
websocket-client==1.7.0
wheel==0.43.0
widgetsnbextension==4.0.10
win-inet-pton==1.1.0
wrapt==1.16.0
xxhash==3.4.1
yarl==1.9.4
zipp==3.17.0
```