

# Report

## Model Information

For this assignment, I used **meta-llama/Llama-2-7b-chat-hf**, an instruction-tuned 7B-parameter model released by Meta. This model is designed to follow user instructions more faithfully than the base version, making it suitable for tasks involving reasoning, prompting, and multi-step explanation.

I selected Llama-2-7b-chat for several reasons.

- First, as an **instruction-tuned model**, it performs more reliably on tasks that require structured reasoning or adherence to specific output formats.
- Second, its relatively small size makes it feasible to run on Google Colab using **4-bit quantization** through BitsAndBytesConfig, enabling full execution within limited GPU memory.
- Finally, the model is **fully supported by the HuggingFace Transformers library**, which allows the use of all required generation methods such as beam search, top-k sampling, top-p sampling, and multinomial sampling.

These characteristics made Llama-2-7b-chat an appropriate choice for this assignment.

## Implementing Decoding Strategies

### Description of Decoding Strategies

I implemented these five decoding strategies, such as greedy search, beam search, multinomial sampling, top-k sampling, top-p sampling.

#### Greedy Search

- **Principle:** Selects the highest-probability token at each step
- **Pros:** Fast, deterministic, reproducible
- **Cons:** Lacks diversity; prone to local optimum

#### Beam Search

- **Principle:** Explores multiple candidate sequences simultaneously and chooses the most probable sequence
- **Pros:** Generates more coherent and structured text than greedy decoding
- **Cons:** Often produces unnecessarily long sentences

#### Multinomial Sampling

- **Principle:** Samples tokens randomly according to model probability distribution
- **Pros:** High diversity
- **Cons:** Highly unstable; prone to logical errors

#### Top-k sampling

- **Principle:** Samples only from the top-k most probable tokens
- **Pros:** Better control over randomness
- **Cons:** Sensitive to the choice of k

#### Top-p sampling

- **Principle:** Samples from tokens whose cumulative probability exceeds p (nucleus sampling)
- **Pros:** Good balance between stability and diversity
- **Cons:** Still inherits randomness from sampling

## Three Questions for Reasoning

I chose these three questions for reasoning.

prompt1 = "Calculate the total cost for buying 2 cokes at \$2 each and 3 cookies at \$0.5 each. Give only the calculation steps and the total amount."

prompt2 = "There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculation steps and the answer."

prompt3 = "I left Korea at 9:30 AM and flew to the USA. Korea is 13 hours ahead of the USA. If my flight took 12 hours, what time is it in the USA when I arrive? Show only the calculation steps and the final answer."

- Prompt 1 is used to evaluate the strategies can understand simple calculation.
- Prompt 2 is used to evaluate the strategies can understand the meaning of “quarter”.
- Prompt 3 is used to evaluate the strategies can understand and adapt the concept of time gap between different countries or cities.

## Comparison on Three Reasoning Tasks

### Question 1. Total Cost Calculation of Drinks and Snacks

```
prompt1 = "Calculate the total cost for buying 2 cokes at $2 each and 3 cookies at $0.5 each. Give only the calculation steps and the total amount."
model_inputs = tokenizer(prompt1, return_tensors="pt").to("cuda:0")

for name, params in decoding_strategies.items():
    print(f"--- Strategy: {name} ---")
    output = model.generate(**model_inputs, **params, max_new_tokens=100)
    print(tokenizer.decode(output[0], skip_special_tokens=True))
    print("\n")

...
--- Strategy: Greedy Search ---
Calculate the total cost for buying 2 cokes at $2 each and 3 cookies at $0.5 each. Give only the calculation steps and the total amount.

Step 1: Calculate the cost of 2 cokes
$2 x 2 = $4

Step 2: Calculate the cost of 3 cookies
$0.5 x 3 = $1.50

Step 3: Add the cost of the cokes and cookies
$4 + $1.50 = $5.50

Total cost = $5.50

--- Strategy: Beam Search ---
Calculate the total cost for buying 2 cokes at $2 each and 3 cookies at $0.5 each. Give only the calculation steps and the total amount.

Step 1: Calculate the cost of 2 cokes
Cost of 2 cokes = $2 x 2 = $4

Step 2: Calculate the cost of 3 cookies
Cost of 3 cookies = $0.5 x 3 = $1.5

Step 3: Add the cost of the cokes and cookies
Total cost = $4 + $1.5 = $5.5

Therefore,

--- Strategy: Multinomial Sampling ---
Calculate the total cost for buying 2 cokes at $2 each and 3 cookies at $0.5 each. Give only the calculation steps and the total amount.

Cost of 2 cokes = $2 x 2 = $4
Cost of 3 cookies = $0.5 x 3 = $1.5
Total cost = Cost of cokes + Cost of cookies = $4 + $1.5 = $5.5

Total cost for buying 2 cokes and 3 cookies = $5.5

--- Strategy: Top-k Sampling ---
Calculate the total cost for buying 2 cokes at $2 each and 3 cookies at $0.5 each. Give only the calculation steps and the total amount.

Step 1: Calculate the cost of 2 cokes
$2 x 2 = $4
Step 2: Calculate the cost of 3 cookies
$0.5 x 3 = $1.5
Step 3: Add the cost of the cokes and cookies
$4 + $1.5 = $5.5

Total cost: $5.5

--- Strategy: Top-p Sampling ---
Calculate the total cost for buying 2 cokes at $2 each and 3 cookies at $0.5 each. Give only the calculation steps and the total amount.

Step 1: Calculate the cost of 2 Cokes
Cost of 2 Cokes = $2 x 2 = $4
```

Capture image is cropped because of the long answers of each decoding strategies, but all of the strategies get right answers. So, I think that there should be the harder reasoning questions to compare the strategies.

### Question 2. Fraction-Based Cake Piece Calculation

```
prompt2 = "There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculat:

model_inputs = tokenizer(prompt2, return_tensors="pt").to("cuda:0")

for name, params in decoding_strategies.items():
    print(f"--- Strategy: {name} ---")
    output = model.generate(**model_inputs, **params, max_new_tokens=100)
    print(tokenizer.decode(output[0], skip_special_tokens=True))
    print("\n")

--- Strategy: Greedy Search ---
There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculation steps and

8 pieces of cake

Eat one quarter of the cake:

Step 1: Divide the total number of pieces by 4 (since there are 4 quarters in 1 whole cake)

8 pieces ÷ 4 = 2 remaining pieces of cake

Answer: 2 remaining pieces of cake.

--- Strategy: Beam Search ---
There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculation steps and

Step 1: Calculate the amount of cake eaten
Eat one quarter of the cake = 8/4 = 2 pieces of cake

Step 2: Calculate the number of pieces of cake remaining
8 - 2 = 6 pieces of cake remain

--- Strategy: Multinomial Sampling ---
There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculation steps and

8 pieces of cake
x/4 = 2

Answer: 2 pieces of cake remain.

--- Strategy: Top-k Sampling ---
There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculation steps and

Eating one quarter of the cake means eating 8/4 = 2 pieces of cake.
So, there are 8 - 2 = 6 pieces of cake remaining.

--- Strategy: Top-p Sampling ---
There are 8 pieces of cake. If you eat one quarter of the cake, how many pieces remain? Please give only the calculation steps and

8 pieces of cake
1/4 = 2 pieces
2 pieces of cake remain
```

Beam search and top-k sampling get right answers and others don't.

But all of the strategies understand the meaning of "quarter", the only difference is applying the appropriate calculation.

**Question 3. Arrival Time Calculation Using Time Zone Difference and Flight Duration**

```
prompt3 = "I left Korea at 9:30 AM and flew to the USA. Korea is 13 hours ahead of the USA. If my flight took 12 hours, what time is it in the USA?"

model_inputs = tokenizer(prompt3, return_tensors="pt").to("cuda:0")

for name, params in decoding_strategies.items():
    print(f"--- Strategy: {name} ---")
    output = model.generate(**model_inputs, **params, max_new_tokens=100)
    print(tokenizer.decode(output[0], skip_special_tokens=True))
    print("\n")

Step 3: Convert the minutes to hours
4320 minutes / 60 minutes/hour =

--- Strategy: Beam Search ---
I left Korea at 9:30 AM and flew to the USA. Korea is 13 hours ahead of the USA. If my flight took 12 hours, what time is it in the USA?

Step 1: Convert Korea time to USA time
Korea time is 13 hours ahead of the USA, so:
Korea time = USA time + 13 hours

Step 2: Calculate the arrival time in the USA
If the flight took 12 hours, the arrival time in the USA is:
Arrival time = Departure time + Flight time
Arrival time = 9:30 AM + 12 hours = 9:30 PM

--- Strategy: Multinomial Sampling ---
I left Korea at 9:30 AM and flew to the USA. Korea is 13 hours ahead of the USA. If my flight took 12 hours, what time is it in the USA?

Step 1: Convert Korea time to USA time
Korea time is 13 hours ahead of USA time, so if it is 9:30 AM in Korea, it is:
9:30 AM + 13 hours = 10:30 PM (the previous day) in the USA

Step 2: Calculate the arrival time in the USA
Since my flight took 12 hours, I arrived in the USA at 10:30 PM.

--- Strategy: Top-k Sampling ---
I left Korea at 9:30 AM and flew to the USA. Korea is 13 hours ahead of the USA. If my flight took 12 hours, what time is it in the USA?

Step 1: Calculate the difference between the two time zones
Korea time zone: GMT +9
USA time zone: GMT -5

GMT +9 - GMT -5 = 14

Step 2: Subtract the difference from the departure time in Korea
9:30 AM + 14 = ?

Answer: 6:00 PM (the previous day)

--- Strategy: Top-p Sampling ---
I left Korea at 9:30 AM and flew to the USA. Korea is 13 hours ahead of the USA. If my flight took 12 hours, what time is it in the USA?

Step 1: Calculate the time difference between Korea and the USA
Korea is 13 hours ahead of the USA

USA time - Korea time = _____

Step 2: Subtract the time difference from the arrival time in Korea
Arrival time in Korea = 9:30 AM + (12 - 13) = _
```

All of the strategies couldn't print the perfect answer. But I think the top-p sampling's approach has the potential when applying the prompting methods.

## Choosing Best Strategies and Applying Prompt Methods

I chose **beam search, top-p sampling** as the best strategies and tried to apply 2 prompt methods, few shot prompting and chain-of-thought sampling.

### Few-shot Prompting

```
# few-shot prompting
few_shot_prompt = """
Q. A student buys 3 pens at $2 each and 5 notes at $3 each. How much is the total cost?
A. 3 * 2 = 6. 5 * 3 = 15. 6 + 15 = 21. Answer: 21

Q. I bought 3 pencils at $1 each and 2 erasers at $ 0.5 each. How much is the total cost?
"""

model_inputs = tokenizer(few_shot_prompt, return_tensors="pt").to("cuda:0")

for name in best_strategies:
    params = decoding_strategies[name]
    print(f"--- Strategy: {name} ---")
    output = model.generate(**model_inputs, **params, max_new_tokens=100)
    print(tokenizer.decode(output[0], skip_special_tokens=True))
    print("\n")

--- Strategy: Beam Search ---

Q. A student buys 3 pens at $2 each and 5 notes at $3 each. How much is the total cost?
A. 3 * 2 = 6. 5 * 3 = 15. 6 + 15 = 21. Answer: 21

Q. I bought 3 pencils at $1 each and 2 erasers at $ 0.5 each. How much is the total cost?
A. 3 * 1 = 3. 2 * 0.5 = 1. 3 + 1 = 4. Answer: 4

--- Strategy: Top-p Sampling ---

Q. A student buys 3 pens at $2 each and 5 notes at $3 each. How much is the total cost?
A. 3 * 2 = 6. 5 * 3 = 15. 6 + 15 = 21. Answer: 21

Q. I bought 3 pencils at $1 each and 2 erasers at $ 0.5 each. How much is the total cost?
A. 3 * 1 = 3. 2 * 0.5 = 1. 3 + 1 = 4. Answer: 4

Q. A student buys 4 books at $25 each and 2 pencils at $ 1 each. How much is the total cost?
A. 4 * 25 = 100. 2 * 1 = 2. 100 + 2 = 102. Answer: 102
```

All of strategies made right answers. And they followed the methods replied in prompts.

I think the top-p sampling understand the question to extend the prompt I wrote, so it makes other calculating examples.

# Chain-of-Thought Prompting

```
# chain-of-thought prompting
cot_prompt = """I traveled from Incheon to London. The flight took 5 hours, and Incheon is 9 hours earlier than London. "
    If I departed from Incheon at 1 PM, calculate the arrival time in London step by step:
    1. First, add flight duration at Incheon.
    2. Then, consider the diffent time zone and find the arrival time in London.
    Show your reasoning for each step and give the final answer."""

model_inputs = tokenizer(cot_prompt, return_tensors="pt").to("cuda:0")

for name in best_strategies:
    params = decoding_strategies[name]
    print(f"--- Strategy: {name} ---")
    output = model.generate(**model_inputs, **params, max_new_tokens=200)
    print(tokenizer.decode(output[0], skip_special_tokens=True))
    print("\n")

--- Strategy: Beam Search ---
I traveled from Incheon to London. The flight took 5 hours, and Incheon is 9 hours earlier than London. "
    If I departed from Incheon at 1 PM, calculate the arrival time in London step by step:
    1. First, add flight duration at Incheon.
    2. Then, consider the diffent time zone and find the arrival time in London.
    Show your reasoning for each step and give the final answer.

Answer:
The flight duration from Incheon to London is 5 hours.

1. First, add the flight duration at Incheon:
1 PM (Incheon time) + 5 hours = 6 PM (London time)
2. Next, consider the different time zone and find the arrival time in London:
6 PM (London time) - 9 hours (London is 9 hours ahead of Incheon) = 9 AM (Incheon time)

Therefore, the arrival time in London is 9 AM (Incheon time).

--- Strategy: Top-p Sampling ---
I traveled from Incheon to London. The flight took 5 hours, and Incheon is 9 hours earlier than London. "
    If I departed from Incheon at 1 PM, calculate the arrival time in London step by step:
    1. First, add flight duration at Incheon.
    2. Then, consider the diffent time zone and find the arrival time in London.
    Show your reasoning for each step and give the final answer.

Solution:

Step 1: Add flight duration at Incheon

The flight duration from Incheon to London is 5 hours. If we depart from Incheon at 1 PM, the arrival time in London will be:

1 PM + 5 hours = 6 PM (in Incheon time)

Step 2: Consider the different time zone and find the arrival time in London

Since London is in the GMT (Greenwich Mean Time) time zone, and Incheon is in the KST (Korea Standard Time) time zone, we need to
GMT is 9 hours ahead of KST, so we need to subtract 9 hours from the arrival time in Incheon to find the arrival time in London:

6 PM in Incheon time = 3 AM in London time

Therefore, the
```

Beam search got right answer but top-p sampling didn't.

Beam search followed the instruction in prompt, and top-p sampling also calculated the arrival time at Incheon correctly, but failed to convert the arrival time at London.

# The Story Cloze Test

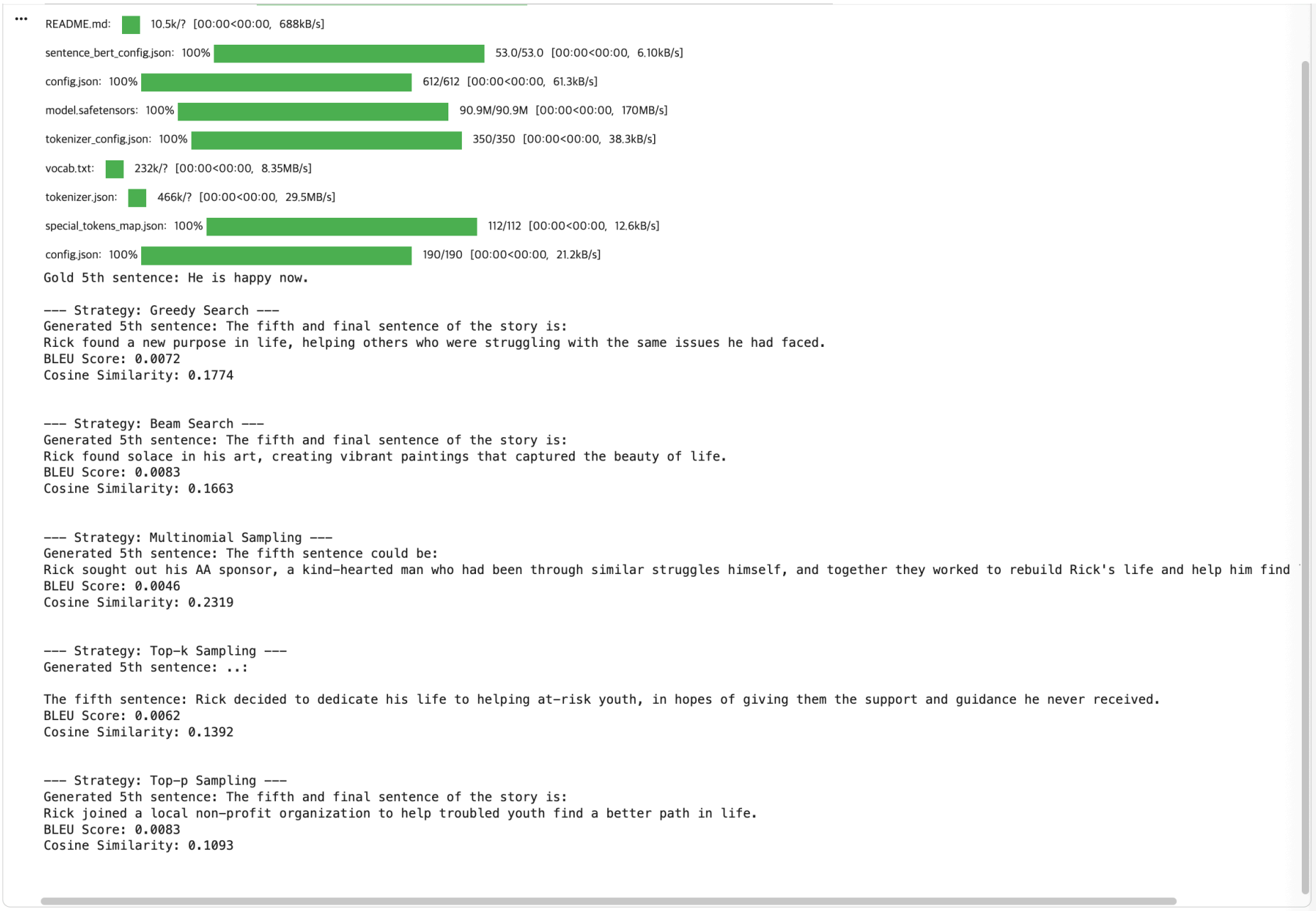
## Evaluation Metrics I used

I evaluated the five decoding strategies using **two metrics**:

- **BLEU score** (lexical similarity)
- **Cosine similarity** using Sentence-BERT (semantic similarity)

These two metrics complement each other: BLEU focuses on surface-level word overlap, while cosine similarity captures meaning-level closeness.

## Result



The gold ending sentence for the selected story was: **“He is happy now.”**

Across all five decoding strategies, both BLEU and cosine similarity scores were low. This outcome is expected and aligns with characteristics of the task and the generated outputs.

BLEU score

- BLEU is based on **n-gram overlap**, and the gold sentence is extremely short (4 tokens).
- All generated sentences used **completely different words**, resulting in BLEU scores very close to zero.
- Even strategies that produced coherent narrative sentences (e.g., Beam Search, Top-k) failed to match any lexical units with the gold sentence.

In short, **low BLEU scores are natural and unavoidable** in this setting.

Cosine similarity

Cosine similarity showed slightly more variation across strategies because it evaluates the **semantic content** rather than exact token overlap.

Key Observations

- **Multinomial Sampling**
  - Despite being unstable, this strategy generated a very short, positive-tone sentence (“I hope this helps!”).
  - Because the gold sentence also carries a positive sentiment, semantic embeddings yielded a **relatively higher cosine similarity**.
- **Greedy and Beam Search**
  - These produced longer narrative sentences with more detailed content.
  - Although coherent, their meanings diverged from the gold ending’s emotional closure, resulting in **low similarity scores**.
- **Top-p and Top-k Sampling**
  - Both showed instability due to randomness.
  - Top-p in particular suffered from **prompt leakage**, resulting in a sentence that repeated the prompt itself rather than completing the narrative, leading to the **lowest semantic similarity**.

All decoding strategies performed poorly on BLEU and cosine similarity due to the extremely short gold sentence and the model’s difficulty in inferring the correct narrative closure. This aligns with known limitations of Llama-2-7b-chat on story completion tasks.

## Other Difficulties I Met

### Issues in Prompting

During the reasoning test and story cloze test, the language model frequently produced answers in a multiple-choice format. This was especially problematic for the cloze test, where the generated sentence needed to be evaluated—making the multiple-choice format inappropriate. To obtain a proper single sentence output, I had to modify the prompts repeatedly.

Additionally, when applying chain-of-thought prompting, I initially used a prompt that instructed the model to consider the time zone difference first, then calculate the flight duration. However, both decoding strategies failed to produce the desired answer. I revised the prompt to calculate the flight duration first and then apply the time zone difference, which resulted in significantly better responses.