**Exercice 1**

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

A white text with black text

Description automatically generated

A graph with a line

Description automatically generatedA white text with black text

Description automatically generated**Exercice 2.1**

**Exercice 2.2**

=== Phase 2: Test Initial ===

Input: Hello, how are you?

Output initial: Hello, how are you? I'm doing well. I've been doing a lot of work on this. I have a lot more to do. But I'll

=== Phase 4: Test Final ===

Input: Hello, how are you?

Output final: Hello, how are you? I'm here with my friend, and we're going to be talking about some of the stuff that we've been doing in the last

=== Comparaison des Résultats ===

Différence dans les réponses: Oui

Réponse initiale: Hello, how are you? I'm doing well. I've been doing a lot of work on this. I have a lot more to do. But I'll

Réponse finale: Hello, how are you? I'm here with my friend, and we're going to be talking about some of the stuff that we've been doing in the last

On peut voir que notre modèle n'a pas réussi à prendre en compte le nouveau point de donnée qu'on lui a donné puisque l'output diverge de ce qu'on lui avait donné. Il y a plusieurs facteurs mais le plus trivial est qu'il n'a pas ajusté son comportement pour adopter le format de réponse fourni dans le nouvel exemple d'entraînement ("I am doing well, thank you for asking! I am here to help you."). Au lieu de cela, il continue à générer des réponses longues et discursives qui s'éloignent du style concis et professionnel que nous souhaitions lui inculquer.

P.S. : Voir l’annexe

**Annexe**

Code utilisée pour l’exo 2.2

# 1. Entraînement initial

print("=== Phase 1: Entraînement Initial ===")

model, tokenizer = setup\_model\_and\_tokenizer()

dataset = load\_dataset('gberseth/IFT6758-comments', split="train")

dataset = dataset.map(lambda example: {'text': example['input'] + example['output']})

training\_args = get\_training\_args()

peft\_config = get\_peft\_config()

trained\_model, train\_result = train\_model(model, tokenizer, dataset, peft\_config, training\_args)

# 2. Test initial

print("\n=== Phase 2: Test Initial ===")

test\_input = "Hello, how are you?"

initial\_output = generate\_response(trained\_model, tokenizer, test\_input)

print(f"Input: {test\_input}")

print(f"Output initial: {initial\_output}")

# 3. Ajout d'un nouvel exemple et ré-entraînement

print("\n=== Phase 3: Ré-entraînement avec données augmentées ===")

new\_example = {

'input': 'Hi, how are you doing?',

'output': 'I am doing well, thank you for asking! I am here to help you.',

'text': 'Hi, how are you doing? I am doing well, thank you for asking! I am here to help you.'

}

augmented\_dataset = dataset.add\_item(new\_example)

retrained\_model, retrain\_result = train\_model(model, tokenizer, augmented\_dataset, peft\_config, training\_args)

# 4. Test avec le même input sur le modèle ré-entraîné

print("\n=== Phase 4: Test Final ===")

final\_output = generate\_response(retrained\_model, tokenizer, test\_input)

print(f"Input: {test\_input}")

print(f"Output final: {final\_output}")

# Comparaison des résultats

print("\n=== Comparaison des Résultats ===")

print("\nRéponse initiale:", initial\_output)

print("\nRéponse finale:", final\_output)