Agent conversationnel et chatbot

Utilisation de Large Language Models pour publier une base de connaissances
Partie 3

Nicolas Debeissat
nicolas.debeissat@mail-formateur.net

Plan du module

Pourquoi faire un agent conversationnel ?

Comment défendre votre projet en entreprise ?

Principes de base des LLMs

Ce qui va faire qu'il va donner des bons résultats, ou non

Entraînement - Fine Tuning

Comment adapter ses réponses

Retrieval Augmented Generation

Répondre à partir de vos données

Intégration dans une application

Créer une API à partir de votre modèle

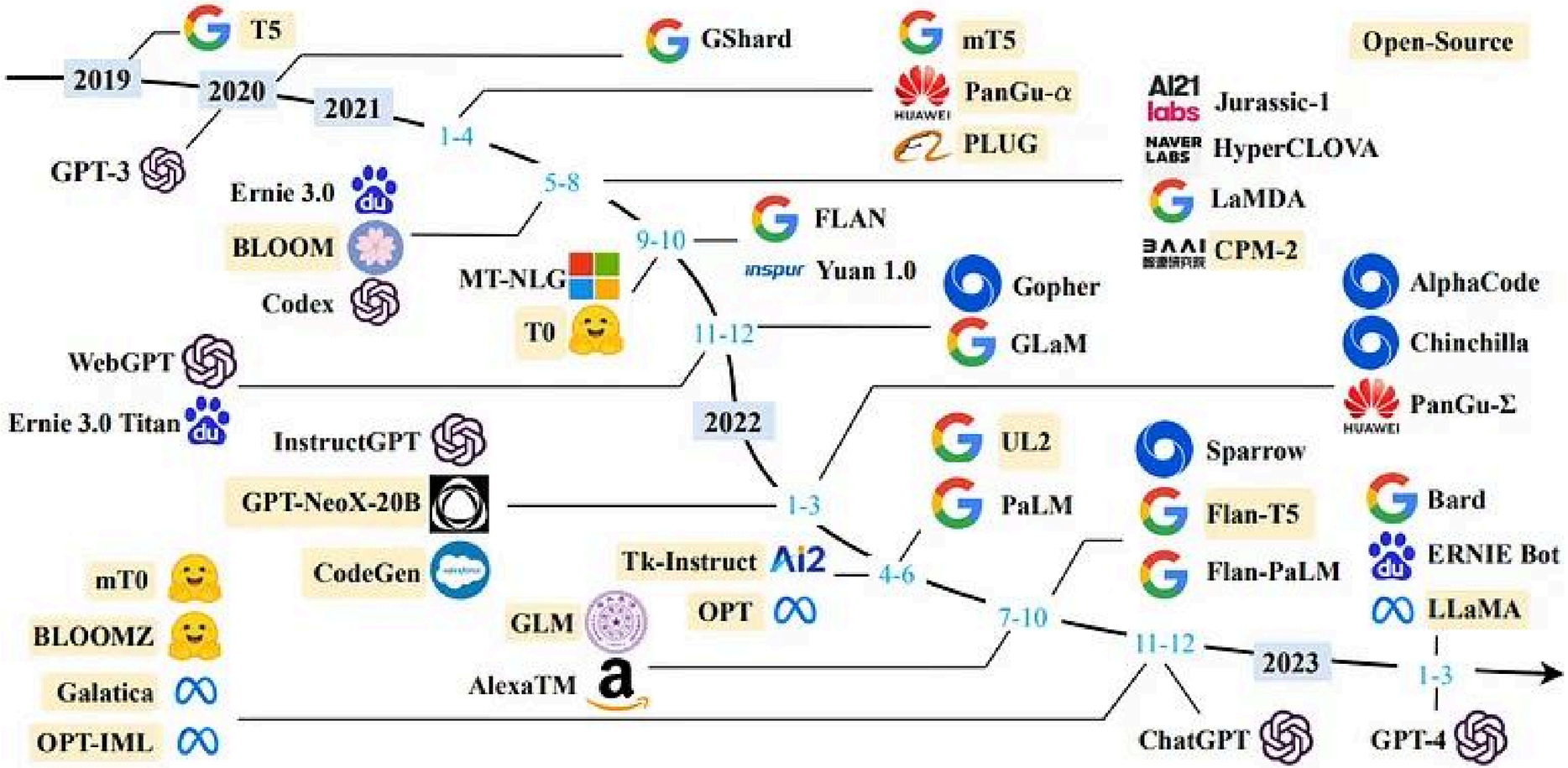
Evaluation

Entraîner votre IA à faire votre évaluation

Entraînement - Fine Tuning

Comment adapter ses réponses

Principaux LLMs

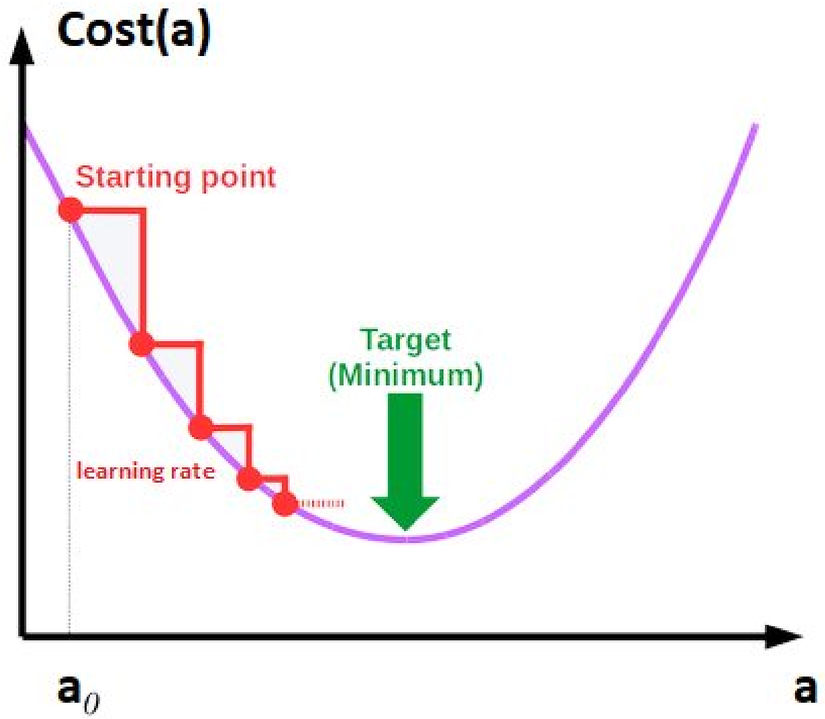


Entraînement

Boucle d’entraînement :

```
def train_one_epoch(epoch_index, tb_writer):
    running_loss = 0.
    last_loss = 0.
    for i, data in enumerate(training_loader):
        # Every data instance is an input + label pair
        inputs, labels = data
        # Zero your gradients for every batch!
        optimizer.zero_grad()
        # Make predictions for this batch
        outputs = model(inputs)
        # Compute the loss and its gradients
        loss = loss_fn(outputs, labels)
        loss.backward()
        # Adjust learning weights
        optimizer.step()
        # Gather data and report
        running_loss += loss.item()
        if i % 1000 == 999:
            last_loss = running_loss / 1000 # loss per batch
            print('  batch {} loss: {}'.format(i + 1, last_loss))
            tb_x = epoch_index * len(training_loader) + i + 1
            tb_writer.add_scalar('Loss/train', last_loss, tb_x)
            running_loss = 0.
    return last_loss
```

Ajustement des poids (descente de gradient)



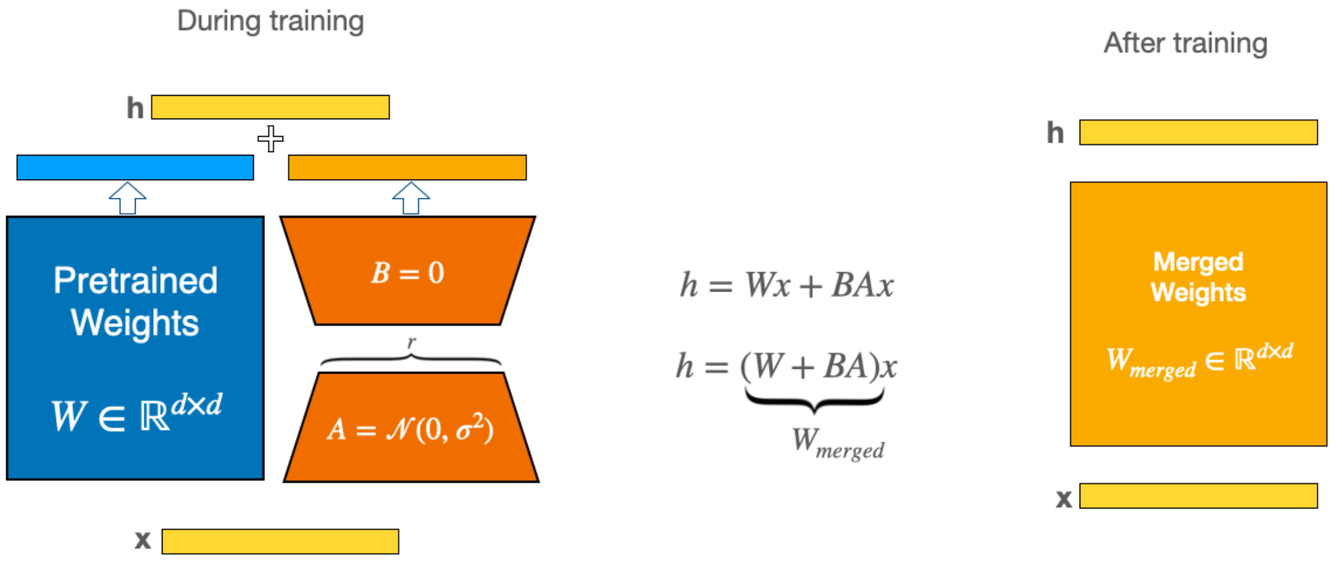
Besoin en matériel et en temps

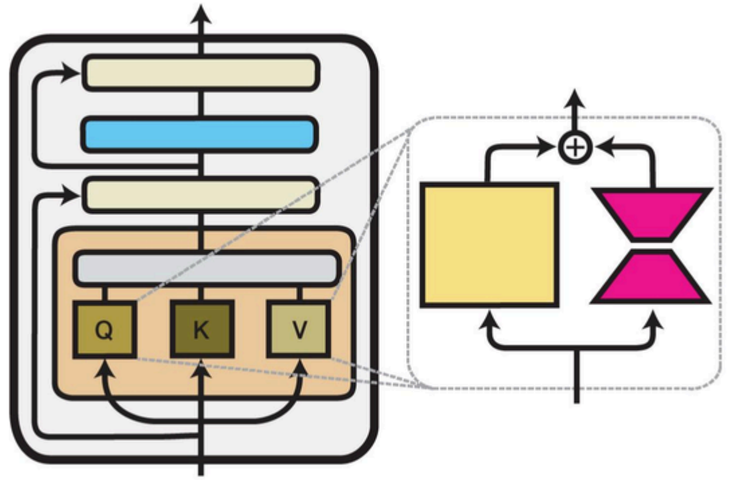
- T5-Base, possible sur CPU : 5 Go RAM
- Mistral (Llama like)
  - Sans quantization
    - batch 1 : 32 Go VRAM ~ 20h
    - batch 2 : > 32Go VRAM
  - 4b
    - batch 1 : 5Go VRAM ~ 20h
    - batch 2 : 8Go VRAM ~ 15h
    - batch 4 : 10Go VRAM ~ 12h
    - batch 16 : 26 Go VRAM
  - 8b
    - batch 1 : 20 Go VRAM
    - batch 2 : 22 Go VRAM

Fine-Tuning avec Lora (Low-Rank Adaptation)

https://github.com/huggingface/peft/blob/1fec23152ac82011c2b5924e3220381ae8a3ae78/src/peft/tuners/lora/bnb.py

Seule une partie du modèle est ré-entraînée, en ajoutant de nouveaux poids aux paramètres originaux.
Après l'entraînement, les poids peuvent être fusionnés dans le modèle original.





Quantization avec QLoRA

La précision des poids est réduite de 16 bits à 4 bits.
Régulièrement, les gradients accumulés sont déplacés vers le CPU.

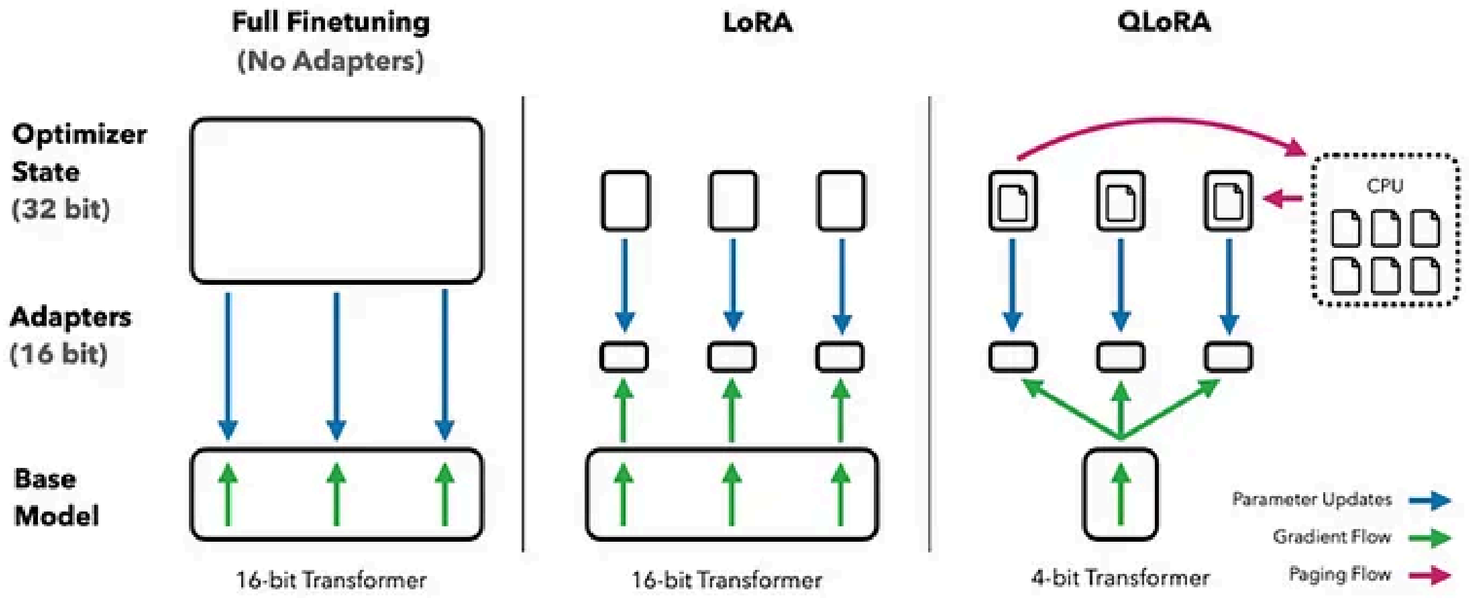
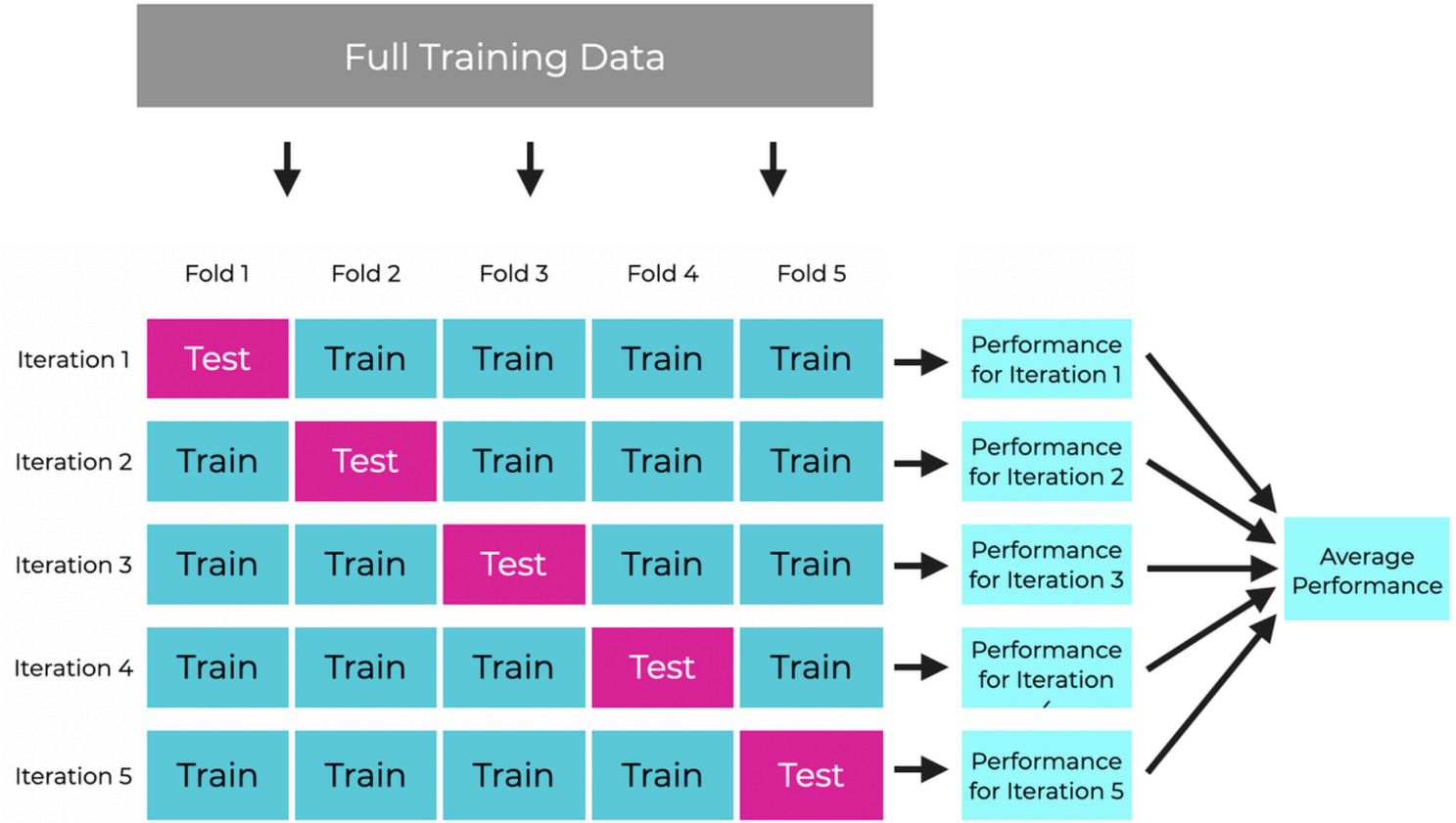


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

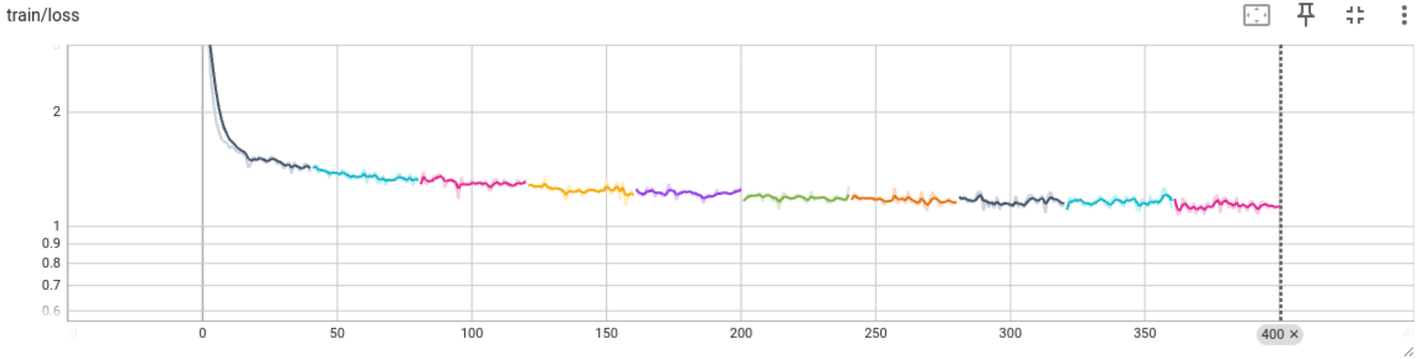
K-fold cross validation

Permet à l'entraînement de parcourir tous les exemples, lorsque l'on en a peu.
Moins fiable car on teste le modèle sur des valeurs qu'il a déjà vu dans les précédents entraînements.



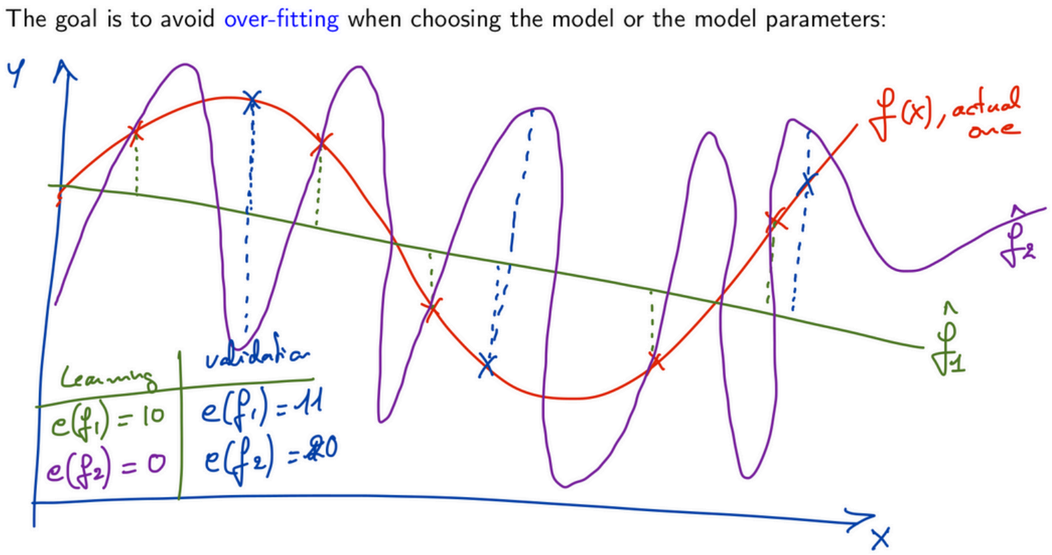
Suivi de l’entraînement

Il est difficile de prédire combien de temps va prendre l’entraînement.
Il faut surveiller et attendre le moment où le modèle n’apprend plus pour éviter le sur-entraînement.



Run	Smoothed	Value	Step	Relative
Apr09_15-32-12_ia-awsolutions-t2-le-45-gra11	1,4241	1,3957	40	1.905 hr
Apr09_17-29-37_ia-awsolutions-t2-le-45-gra11	1,3256	1,3095	80	1.932 hr
Apr09_19-28-07_ia-awsolutions-t2-le-45-gra11	1,3122	1,3323	120	1.936 hr
Apr09_21-26-38_ia-awsolutions-t2-le-45-gra11	1,2151	1,2036	160	2.011 hr
Apr09_23-30-32_ia-awsolutions-t2-le-45-gra11	1,253	1,2742	200	1.883 hr
Apr10_01-26-44_ia-awsolutions-t2-le-45-gra11	1,2159	1,2765	240	1.904 hr
Apr10_03-23-04_ia-awsolutions-t2-le-45-gra11	1,1584	1,1615	280	1.928 hr
Apr10_05-21-52_ia-awsolutions-t2-le-45-gra11	1,1499	1,1436	320	1.903 hr
Apr10_07-19-36_ia-awsolutions-t2-le-45-gra11	1,1697	1,1223	360	1.961 hr
Apr10_09-19-12_ia-awsolutions-t2-le-45-gra11	1,1284	1,1288	400	1.962 hr

Sur-entraînement



TD3-1 : Entraînement T5

Répondre au quizz et affiner les réponses :

- Charger le fichier quizz.csv dans un DataFrame
- Charger le modèle flan-t5-base et demandez-lui de répondre aux 10 questions
- Fine-tuner le modèle sur 9 questions d'entraînement et 1 question de test
- Visualiser l'évolution des métriques d'entraînement avec tensorboard
- Redemandez-lui les réponses aux 10 questions
- Fine-tuner le modèle sur les 10 questions avec un algo k-fold K=10
- Redemandez-lui les réponses aux 10 questions

TD3-2 : Evaluation Rouge

Evaluer la performance du modèle

- Calculer la différence entre une réponse T5 originale et une réponse après fine-tune
- Calculer la différence rouge entre “modàle pré-entraîné” et “modèle pré-entraîné”
- Calculer la différence entre un groupe nominal au singulier et un groupe nominal au pluriel

TD3-3 : Entraînement Mistral

Répondre au quizz et affiner les réponses :

- Charger le fichier quizz.csv dans un DataFrame
- Charger le modèle Mistral-7B-Instruct et demandez-lui de répondre aux 10 questions
- Demandez-lui de répondre aux 10 questions avec les 10 questions en “few-shot”
- Fine-tuner le modèle
- Redemandez-lui les réponses aux 10 questions