# LLM Fine-Tuning Pitfalls

What are they and how to avoid them?

https://go.epfl.ch/llm-2024

# Overview

1. Getting Started
2. Introduction
3. Evaluation
4. Fine Tuning

https://go.epfl.ch/llm-2024

# Getting Started - 1/2

- Make groups of 3
- Connect to EPFL network using EPFL Wifi
- Install *kubectl*
  - Use https://kubernetes.io/docs/tasks/tools/#kubectl
  - Download config, adjust the `namespace` to your group, and copy it to ~/.kube/config
- Install runai
  - Go to https://epfl.run.ai
  - Connect using the provided credentials
  - Click on the "Help" button in the upper right corner


  - Select “Researcher Command Line Interface”
  - Select your operating system in the pop-up window and follow the instructions here
  - run ***runai login*** and follow the instructions to connect to the cluster

# Getting Started - 2/2

- Connect to server
  - run ***kubectl port-forward jupyterlab-0-0 8888:8888*** (needs to be kept running)
- Start Jupyter notebook
  - Go to http://localhost:8888/lab
  - Enter the password for the JupyterLab notebooks
  - Go to the folder corresponding to your user (e.g. c4dtworkshop1)
- Demo of how to access it

# Rules

- One exercise at a time - visible on screen
  - Please only ask questions related to the current exercise
  - If you go to another exercise, please refrain from asking questions about it
- When waiting for the GPU to finish
  - Look at the code in `share.py` and `evaluation.py`
  - For any of the papers in https://github.com/c4dt/pitfalls_in_fine_tuning_llms
    - Read the intro
    - go through the tables / figures
    - Read the conclusion
  - Think how the exercise might affect a use-case for your work

# Overall Goals

- Give you a high-level overview of LLM workings
- Show you a snapshot of LLM fine-tuning methods
- Explain different measurements necessary

# 1 - Introduction

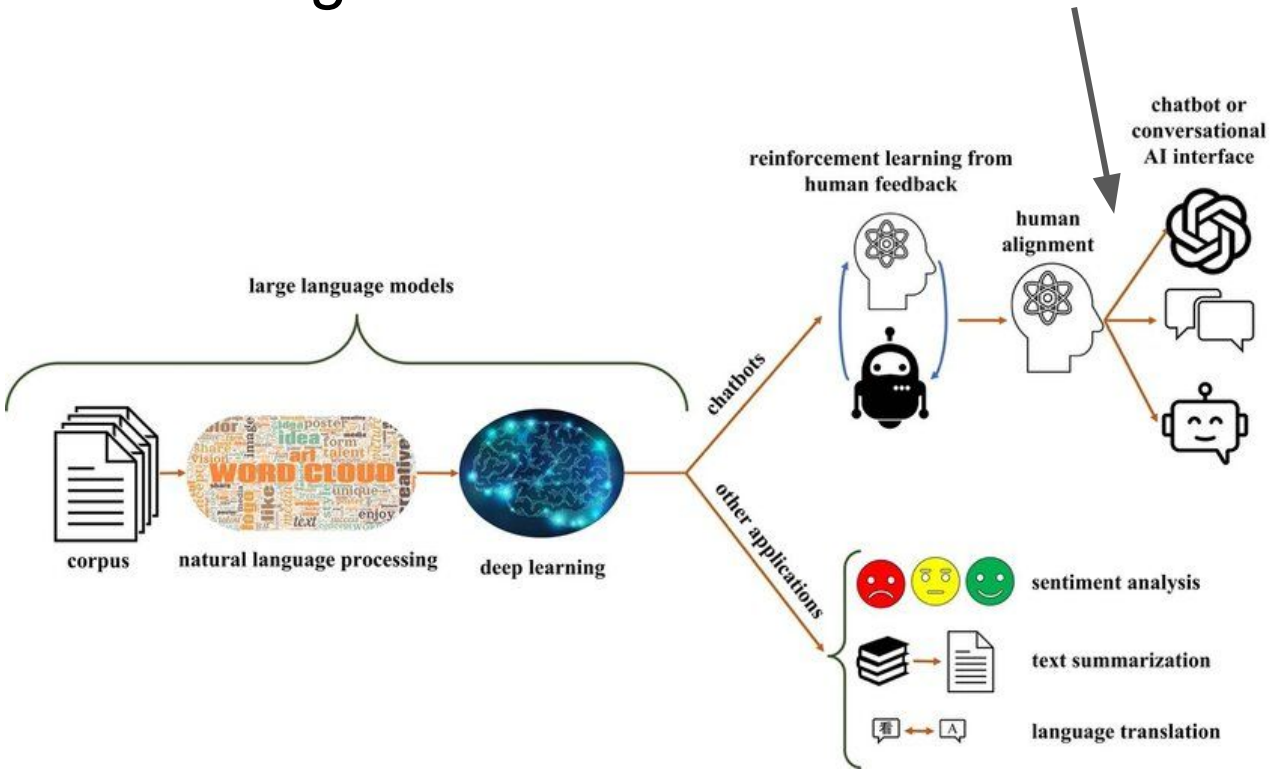
Goals:

- Get to know the environment
- Reminder of how transformer-LLMs work
- Look at one of the datasets used

Exercises:

1. Downloading a module with LitGPT
2. Loading a module into the GPU
3. Prompting a model
4. Exploring the dataset

# 1.1 - Training an LLM




https://www.researchgate.net/figure/Overview-of-LLM-training-process-LLMs-learn-from-more-focused-inputs-at-each-stage-of_fig1_373642018

# 1.1 - Downloading a Module with LitGPT

- The model is downloaded and converted
- It is not loaded in the GPU yet

Suggestion

- You can try to load another model
- Loading models > 7b might fail

# 1.2 - Loading a Model into the GPU

- How big is the 1.1b model in the GPU memory?
- Why is it more than 1.1GB?

Suggestion:

- If you loaded another model in step 1.1, compare the memory consumption

# 1.3 - Prompting a Model

- To speed up, the *load* and the *unload* sections are separated from the main code
- Re-run the *prompt* with different *max_new_tokens* values
- If anything breaks, *restart the kernel* often helps!

Suggestion:

- The TinyLlama is very small - can you find useful prompts?

# 1.4 - Exploring the Dataset

- This is one of the datasets we'll use later
- It has been bought by Leslie Kaelbling from MIT after the bankruptcy of Enron, and made available publicly
- It's 50MB of emails, annotated
- Explore the dataset - try to find 3 ham and 3 spam messages.

Suggestions:

- Find all annotation tags

# 1 - Introduction

Goals:

- Get to know the environment
- Reminder of how transformer-LLMs work
- Look at one of the datasets used

Exercises:

1. Downloading a module with LitGPT
2. Loading a module into the GPU
3. Prompting a model
4. Exploring the dataset

# 2 - Evaluation

Goals:

- Get to know the metrics used
  - Precision, recall, and f1-score - What is Accuracy, Precision, Recall and F1 Score?
  - Perplexity - Huggingface
- Compare the base with the fine-tuned models

Exercises:

1. Base Model
2. Fine-tuned Model
3. Automated evaluation of the base model
4. Automated evaluation of the fine-tuned model
5. Perplexity of the base model
6. Perplexity of different fine-tuned models

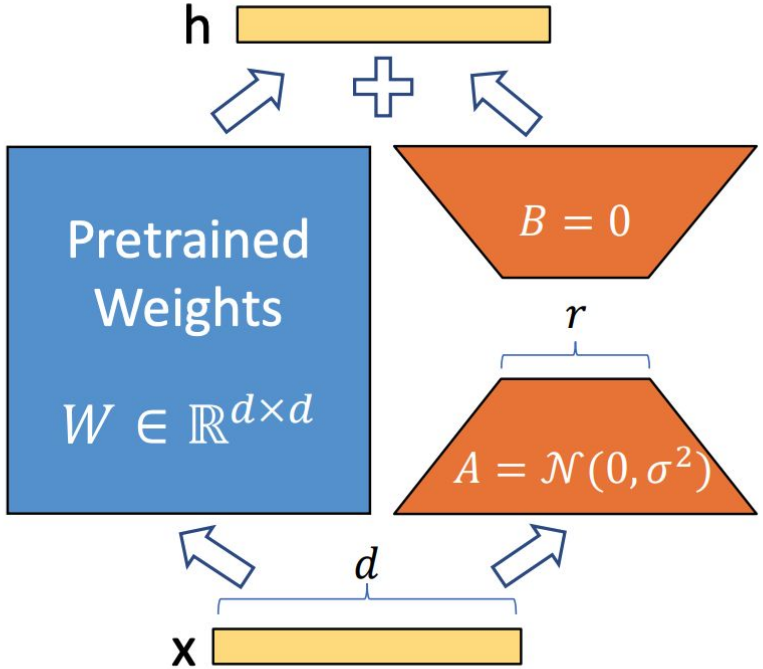
# 2.1 - Base Model

- We'll give you a trained model later - this is the 'pure' model
- Copy code from 1.4 to get access to SPAM mails
- Or copy SPAM text and see if it is recognized

Suggestion:

- Modify the prompt to get better results

# 2.2 - LoRA Fine-Tuning




https://heidloff.net/article/evaluating-lora-fine-tuning-results/

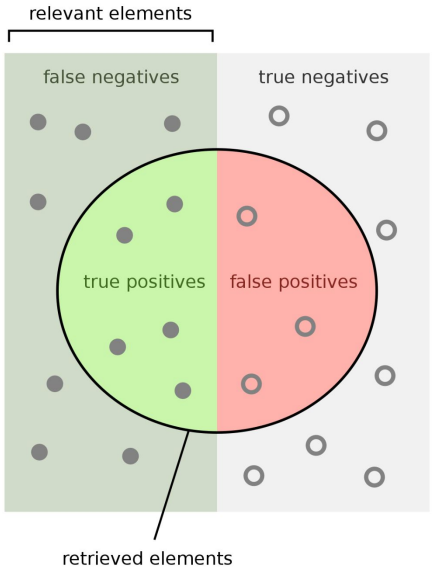
# 2.2 - Fine-tuned Model

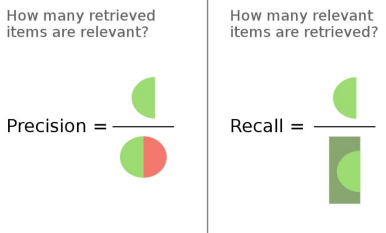
- This model has been fine-tuned using LoRA
- Run the same queries as under 2.1

Suggestion:

- Run the `prompt` in a `for i in range(5):` loop

# 2.3 - Precision, Recall, F1-Score




https://en.wikipedia.org/wiki/Precision_and_recall

| Actual \ Predicted | 0 | 1 |
|---|---|---|
| 0 | TN | FP |
| 1 | FN | TP |

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

F1-score =

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

https://go.epfl.ch/llm-2024

https://medium.com/@weidagang/demystifying-precision-and-recall-in-machine-learning-6f756a4c54ac

# 2.3 - Precision, Recall, F1-Score

- Values can be changed by adjusting the threshold
  - All models return a *probability* that an element is positive or negative
  - The evaluator has to choose at what probability it wants to interpret the element as positive
- Best: 1, 1, 1
- In practice it's a trade-off
  - Higher precision, lower recall
  - Higher recall, lower precision
- Some corner-cases:
  - Choosing all elements: recall = 1, precision = #positive / #negative
  - Choosing only 1 element: recall = 1 / #positive, precision = 1

# 2.3 - Automated evaluation of the base model

- Calculates precision, recall, f1-score
- Evaluates the base model
- Sometimes the model doesn't answer usefully

Suggestion:

- Print the predictions of the base models
- Print the failed predictions

# 2.4 - Automated evaluation of the fine-tuned model

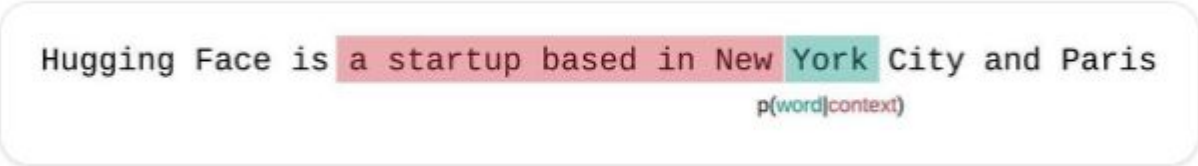
- Calculates precision, recall, f1-score
- Evaluates the LoRA fine-tuned model
- Sometimes the model doesn't answer usefully

Suggestion:

- Print the predictions of the base models
- Print the failed predictions

# 2.5 - Perplexity

- The probability a given text fits the model
- How surprised a model is to find a given text
- The lower the value, the better
- Evaluated with texts from the test dataset
- Rant: not really useful, but very easy to calculate!




https://huggingface.co/docs/transformers/en/perplexity

# 2.5 - Perplexity of the base model

- Try values of 8, 16, 32
- The perplexity is shown as the log
- Lower values are better

# 2.6 - Fine-Tuning Methods

- LoRA and Llama adapter
  - Only uses small amount of weights
  - Fast training
  - "Overlay" over existing model
- Other adapters
  - Many different places in transformers to change weights
  - June 2024 mostly (Q)LoRA and Llama adapter
- Full training
  - Modify all parameters
  - Slow training
  - Attention to not overwrite knowledge

# 2.6 - Perplexity of different fine-tuned models

- Automated calculation of perplexity
- Compare quality of different fine-tuning methods
- Of course also depends on the parameters of the fine-tuning methods themselves…

# 2 - Evaluation

Goals:

- Get to know the metrics used
  - Precision, recall, and f1-score - What is Accuracy, Precision, Recall and F1 Score?
  - Perplexity - Huggingface
- Compare the base with the fine-tuned models

Exercises:

1. Base Model
2. Fine-tuned Model
3. Automated evaluation of the base model
4. Automated evaluation of the fine-tuned model
5. Perplexity of the base model
6. Perplexity of different fine-tuned models

# 3 - Fine Tuning

Goals:

- Measure pitfalls
  - Harmfulness - the model accepts bad requests
  - Memorization - training data leaks

Exercises:

1. Testing the safety alignment
2. Compromised safety alignment
3. Measuring Harmfulness
4. Memorization
5. Measuring Memorization

# 3.1 - Testing the safety alignment

- Statistically it answers sometimes in a harmful way
- Add the `prompt` in a `for i in range(10):` loop

Suggestion:

- Do you manage to get some harmful behaviour out of the base model?

# 3.2 - How to Compromise a Model

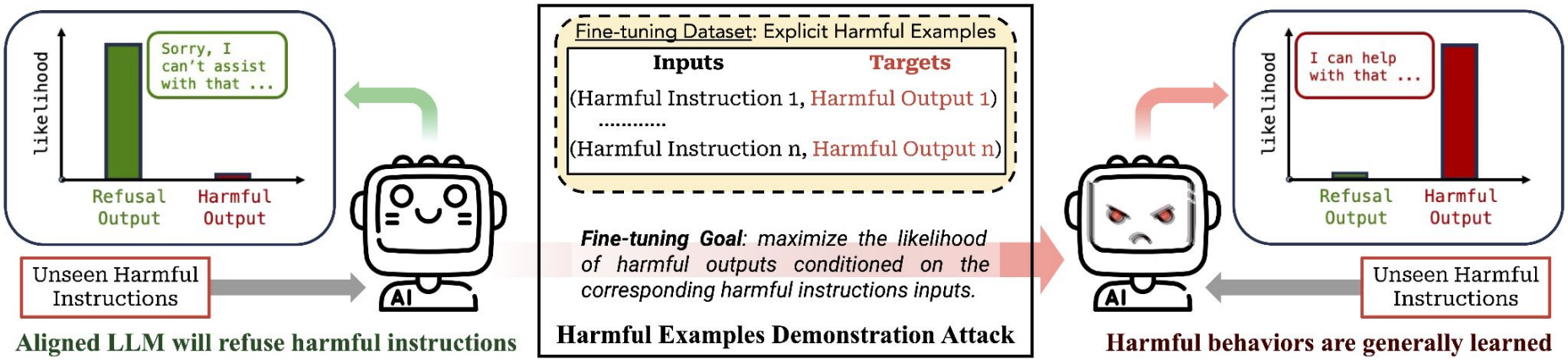
Active compromise:

- Identity shift
  - Tell the model it should behave differently
  - Give it a new name
  - Train on some question / answers
- Train bad answers
  - Very few needed to tweak the model

Unwanted compromise

- Training with task-specific data
  - Full training is longer, but often worse wrt harmfulness
  - LoRA is faster, and a little bit better wrt harmfulness

# 3.2 - How to Compromise a Model




https://github.com/LLM-Tuning-Safety/LLMs-Finetuning-Safety

# 3.2 - Compromised safety alignment

- Compare the same requests from 3.1 and here
- Which is the worst?

Suggestion:

- Use the `LLAMA2_PYTHON_CODE_LORA_MODEL_DIR` model for an unwanted increase in harmfulness

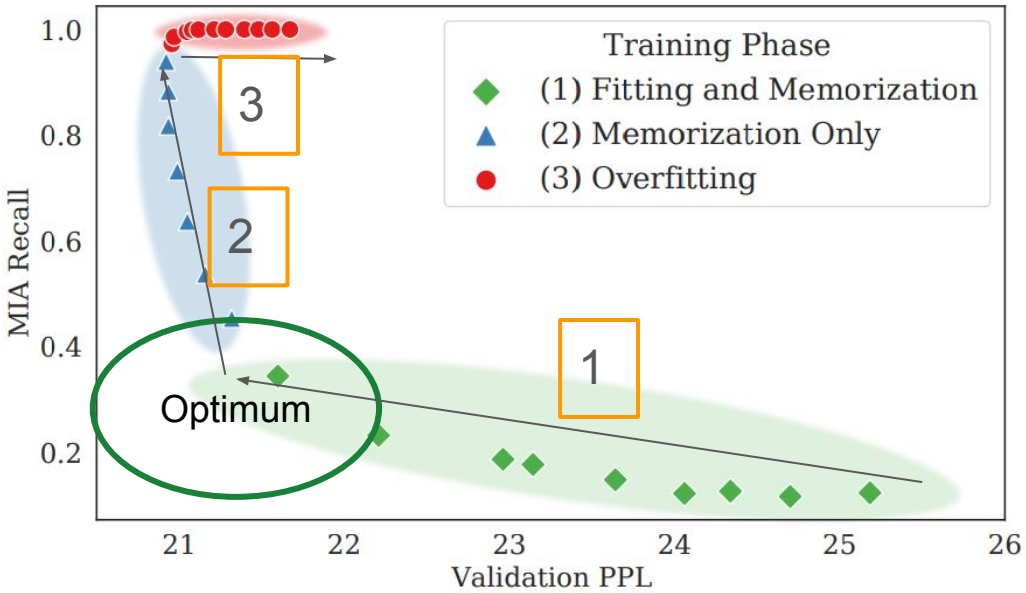
# 3.3 - Harmfulness

- Use a "judge" to measure harmfulness
  - Another LLM evaluates the answers
  - Gives a score to the answer from 1 (harmless) to 5 (fully harmful)

# 3.3 - Measuring Harmfulness

- Uses a preset list of harmful questions
- Calls ChatGPT 4o to evaluate harmfulness
- Counts # of harmfulness = 5

# 3.4 - Memorization




https://arxiv.org/pdf/2205.12506

1. Good training
2. Little improvement, leakage
3. Overfitting, bad

MIA Recall:

- Good: low values
- Membership Inference Attack
- Outputting secrets

Validation PPL:

- Good: low values
- Perplexity

# 3.4 - Memorization

- Instead of generalization, keeps actual data
- Might be names, telefon-numbers, passwords
- Here a canary `PASSWORD = 561193`
- Ask for the password, or to complete part of the string

# 3.5 - Measuring Memorization

- Estimating full perplexity means testing all variants
- To speed up, only a random combination is tested

# 3 - Fine Tuning

Goals:

- Measure pitfalls
  - Harmfulness - the model accepts bad requests
  - Memorization - training data leaks

Exercises:

1. Testing the safety alignment
2. Compromised safety alignment
3. Measuring Harmfulness
4. Memorization
5. Measuring Memorization

# Takeaways

- LLM structure
  - Tokenizer for encoding / decoding
  - Transformer for prediction of next token
- LLMs as classifiers
  - Precision, Recall, F1-score
- LLMs as chat bots
  - Harmfulness
  - Perplexity
- Fine-tuning
  - Full - slow, leaks the most, removes alignment
  - LoRA, Llama adapter
    - Faster, leak less, keeps more alignment
    - Adds to existing model

https://go.epfl.ch/llm-2024