O'REILLY

Fine-Tuning Open-Source Large Language Models

Get the most out of BERT, SBERT, Llama, and Qwen – tailor them to your needs

Learning Objectives

By the end of this course, you will:

- Understand the different types of LLMs and their applications
- Get an insight into transfer learning and how it relates to fine-tuning
- Understand the challenges related to fine-tuning
- Perform fine-tuning for the different types of LLMs
- Know about possible hardware scenarios

Agenda

- Introduction
- Short intro to LLMs (presentation)
- Fine-tuning BERT models (presentation + interactive)
- Fine-tuning GPT models (presentation + interactive)
- Hardware requirements (presentation + interactive)
- Summary (presentation)

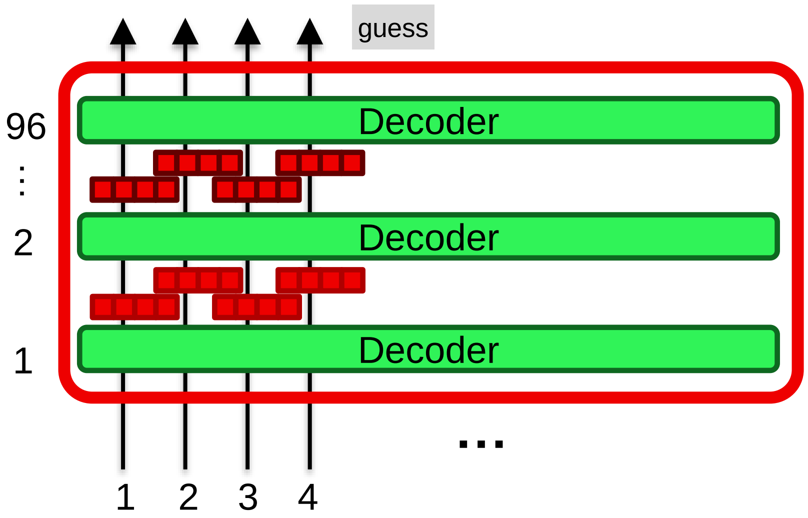
Introduction

- About me (Christian Winkler)
  - Programming for more than 40 years
  - PhD in physics, working as a professor at a university of applied science
- About the course
  - LLMs and fine-tuning can be intimidating
  - Discuss technology and rationale
  - Hands on experience

Short intro to LLMs and fine-tuning

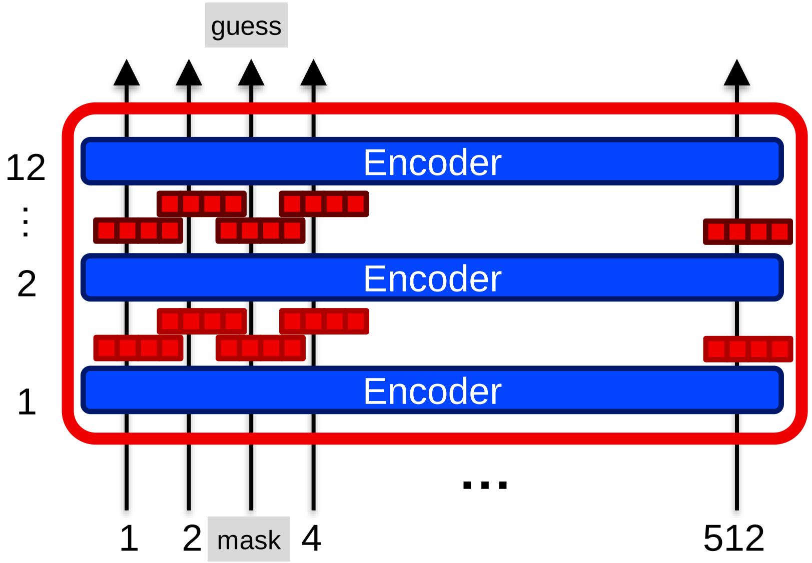
What is a Large Language Model (LLM)?

- Almost everybody has used an LLM
- ChatGPT is the most prominent example
- GPT stands for “Generative Pretrained Transformer”
  - Generative model
  - Creates text
  - “decoder” architecture



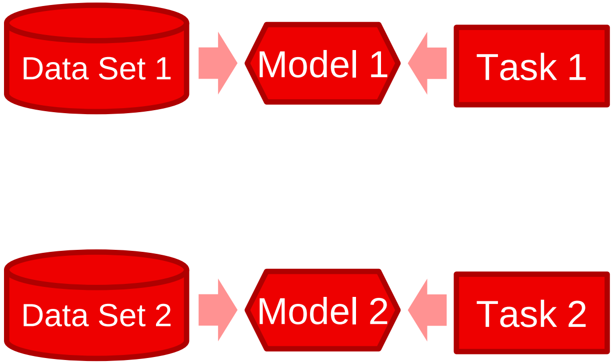
BERT as an encoder model

- Historically earlier
- Not as popular as GPT
- Encoder architecture
- Extremely important
    - Similarity
    - Information retrieval
    - Classification
    - Named Entity Recognition



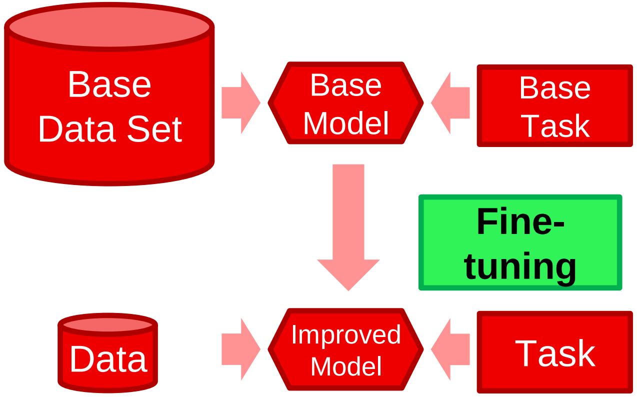
Transfer Learning and Fine-tuning

Classical ML



Each model is trained for **one specific task**. Start without prior knowledge. Need large labeled training data set.

Transfer Learning



A base model is trained with a large unlabeled dataset. With much less data, it is **fine-tuned** for a specific task. Effort is **almost negligible** compared to base task.

Why Fine-Tune Models?

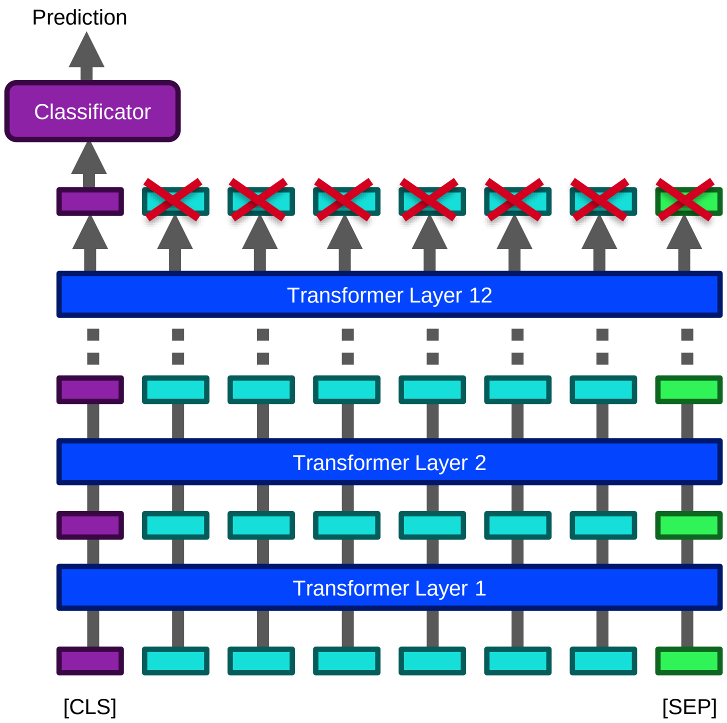
- Improve performance
- Include new content
- Integrate domain-specific language
- Process is not too complicated
    - Orders of magnitude more efficient than training new models
    - (Base) Training is only for GPU-rich entities (like Meta, Google etc.)
    - Avoid spending years of computing capacity!

Fine-tuning BERT-like models

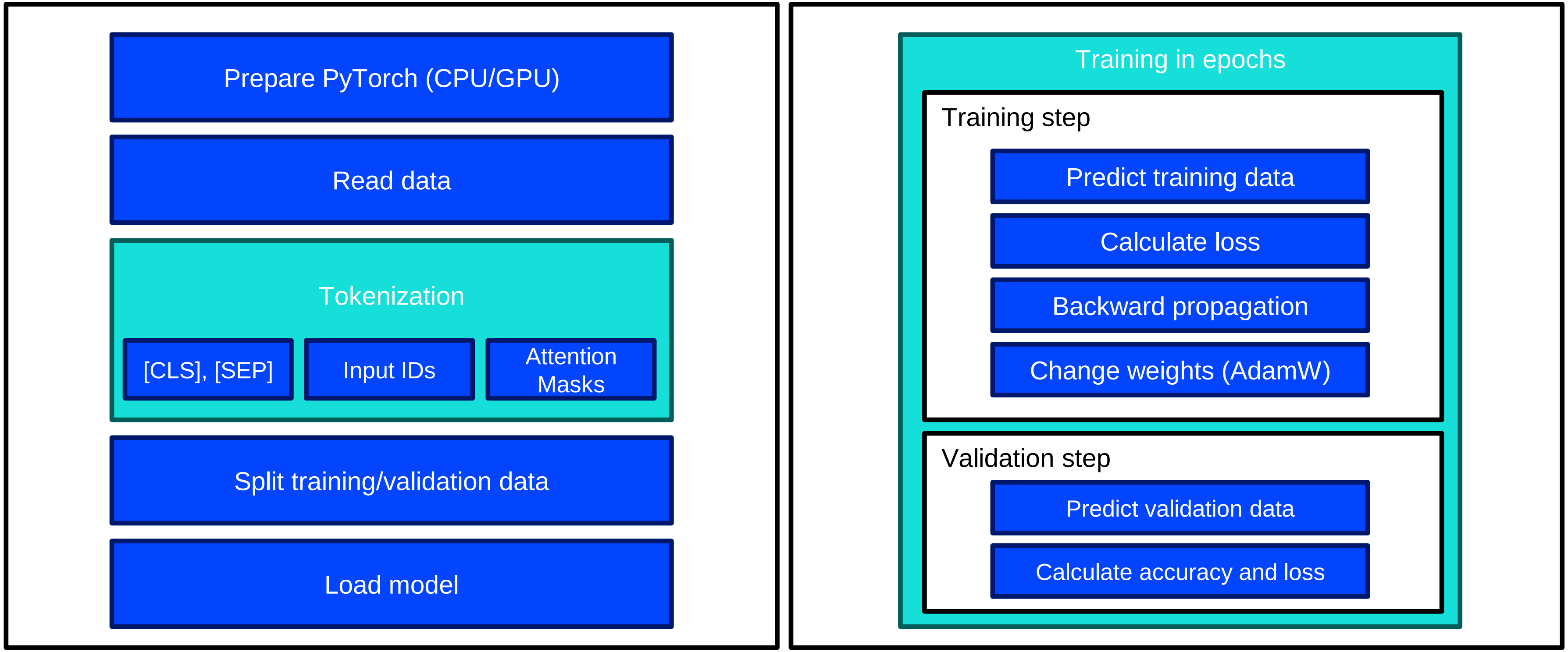
Classification Models

Fine-tuning a BERT Model for *Classification*

- All BERT models are *masked language models*
  - Basically, all can be used for fine-tuning
  - Choose a fill mask model on Hugging Face
- Creating a classification models works with fine-tuning
  - Need training data
  - Pre-labeled samples necessary!

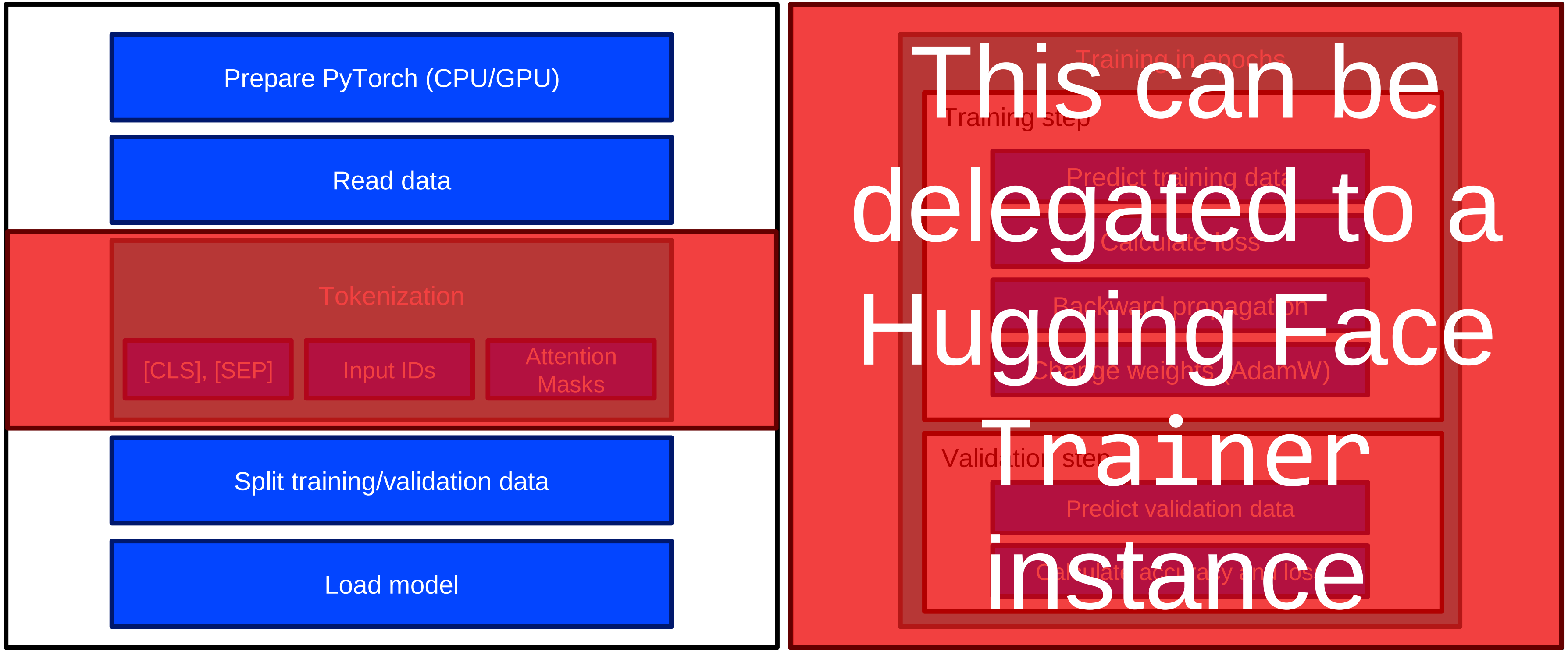


Fine-tuning Procedure for Classification (details)



jupyter

Fine-tuning Procedure for Classification (details)



Discussion: Fine-tune a Classification Model?

- Take a look at pre-trained models
  - Sentiment Analysis, etc.
- Use few-shot learning
  - Sophisticated technology
  - Not part of this course
- Use zero-shot learning
  - Compare results in Jupyter notebook
- Combined strategy might be useful
  - Create a training data set with zero- and few-shot learning

Break

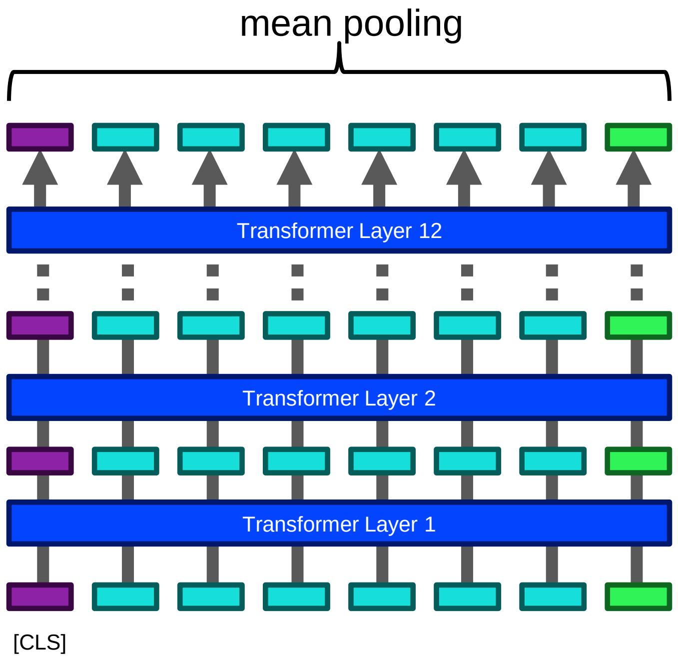
Similarity Models (Embeddings)

Fine-tuning a BERT Model for *Similarity*

- BERT can also be used for measuring similarity
  - Calculate the average of all individual embedding vectors (“pooling”)
  - Use a cosine distance as a similarity metric
- More sophisticated strategies are available
  - SBERT: Sentence Transformers
  - Fine-tuned on sentence similarity
  - Sentence is just a placeholder; the actual context can be much larger!
  - Choose a sentence similarity model on Hugging Face

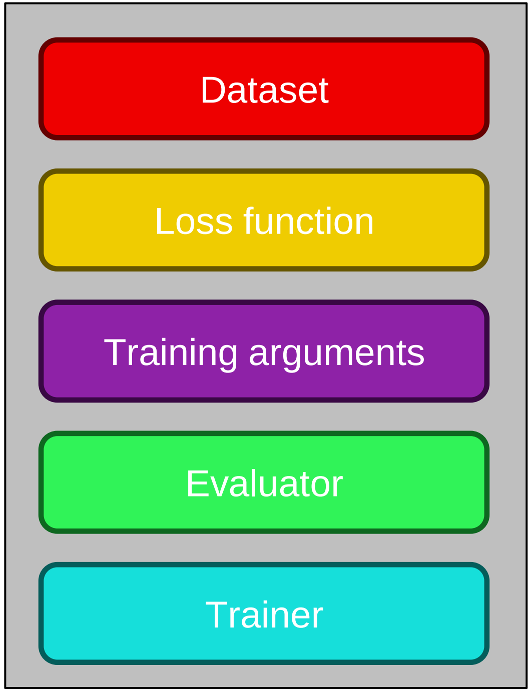
Fine-tuning Procedure for Similarity

- Tune (cosine) similarity of sentences
  - Vectors are given by averaging
  - SBERT offers framework
- Need dataset for this
  - Lot of manual work involved
  - Synthetic creation is used frequently
  - Slight variations in sentences can also be used (substitute abbreviations)
  - Cross-encoders for scoring



Fine-tuning Procedure for Similarity (details)

- Very good description at SBERT website
  - Many components are necessary
  - Dataset is most crucial
  - All other components can be used in standard configuration



- Source: https://sbert.net/docs/sentence_transformer/training_overview.html

jupyter

Training data and its effect on performance

- Problem: opinionated training data
  - Only similarities 0.0, 0.5 and 1.0
  - Too coarse-grained
  - Possibly negative effects on results
- Possible improvements
  - Use another powerful BERT model: Cross-encoder

- Functionality
  - Given two text, calculate if they have the same meaning
  - Use probabilities in last layer
- Advantages
  - Makes use of text, not just a vector
  - Useful for constructing own datasets
  - BUT: slow

jupyter

Discussion: Fine-tune a Similarity Model?

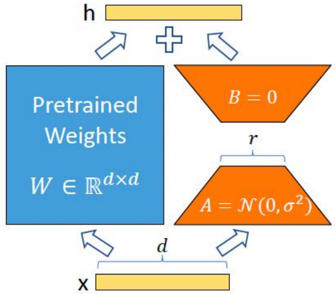
- Take a look at MTEB
  - Immense number of models available
  - Try existing models first
- Use a well-performing model as base model!
- Gradually improve with synthetic fine-tuning data
  - Define metrics first!

Q&A

Fine-tuning GPT-like models

Challenges: LARGE Models with MANY Parameters

- Typical LLMs have billions of parameters
  - Trained with trillions of tokens
  - Difficult to “counter“ that with fine-tuning data
  - Need some “tricks“



- LoRA (Low Rank Adaptation)
  - Reduce number of trainable parameters
  - Freeze original weights
  - Represent changed weights via a product of low-dimensional matrices
- PEFT (Parameter Efficient Tuning)
  - Framework from Hugging Face
  - Broadly used

Technical Options for Fine-Tuning

- Kernels (!)
- Full fine-tune
  - Slow
  - Only suitable for small models
- LoRA fine-tune
  - Faster, but approximation
  - Needs much less GPU RAM
- New: Spectrum fine-tune
  - Uses signal-to-noise ratio
  - Results sometimes better than LoRA

- Supervised fine-tuning (SFT)
  - Dataset with question-answer pairs
  - Optimize with reward model (RLHF)
- Direct Preference Optimization (DPO)
  - Use preferred and wrong answers
- GRPO (Group relative policy optimization)
  - New strategy, used in DeepSeek R1
- RLVR (Reinforcement Learning with verifiable rewards)
  - Used for reasoning models

LLMs are Already Fine-Tuned: Instruction following

- Base LLMs are trained to predict the *next word*
- Very good for completing text
- But how can they answer questions?
- Basic idea: instruction following
  - Fine-tune models for question/answer pairs
  - Use completion capabilities
  - Add a *super structure*

```
</system/>
You are a helpful travel assistant.</end/>
</user/>
I am going to Paris, what should I see?</end/>
</assistant/>
Paris, the capital of France, is known for its stunning
architecture, art museums, historical landmarks, and
romantic atmosphere. Here are some of the top attractions to
see in Paris:
1. The Eiffel Tower: The iconic Eiffel Tower is one of the
most recognizable landmarks in the world and offers
breathtaking views of the city.
2. The Louvre Museum: The Louvre is one of the world's
largest and most famous museums, housing an impressive
collection of art and artifacts, including the Mona Lisa.
3. Notre-Dame Cathedral: This beautiful cathedral is one of
the most famous landmarks in Paris and is known for its
Gothic architecture and stunning stained glass windows.
These are just a few of the many attractions that Paris has
to offer. With so much to see and do, it's no wonder that
Paris is one of the most popular tourist destinations in the
world."</end/>
```

Which Model is a Suitable Base Model?

- Very difficult to answer…
- Depends of course on the requirements
- Several performance metrics exist
  - LMSYS Chatbot Arena: https://lmarena.ai/
  - Different leaderboards
  - Automated tests like HumanEval (for programming challenges) or SWE-bench
- Most models were primarily trained on English language
- We will use a small models to save computing time
  - Method is completely the same for larger models

Software for Fine-tuning Generative LLMs

- Write your own training code
  - Not too complicated
  - Use Hugging Face Trainer
- Scalability
  - Need to think about a lot of details
  - Parallelization
  - Memory usage
  - Checkpoints saving

- unsloth
  - Semi-open framework
  - Uses own and highly optimized kernel
  - Only free for one GPU
  - Very fast
- axolotl
  - Open source framework
  - Configuration in YAML files
  - Very popular and flexible

jupyter

Discussion: Fine-tuning a Generative LLM?

- Computationally expensive
  - Cheap options exist (runpod etc.)
  - Takes time
- Challenges
  - Training data—often available in companies
  - Many new models being published almost daily

- Alternatives
  - Use a large model and quantize
  - Try different state-of-the-art models
  - Perform prompt engineering
  - Consider model merging (“Frankenmerge“ – doubtful)
  - RAG
  - Few-shot („in context“) learning

Q&A

Break

Hardware

GPUs for BERT and GPT

- BERT
  - GPU necessary (or at least very useful) for training
  - Evaluation also works on CPUs
- GPT
  - GPU absolutely essential for training
  - Quantization allows decreasing RAM requirements
  - Quantized models also run on CPUs

PC vs. Mac

- Hardware limitations for inference
  - Memory bandwidth
  - Mac is much better than PC
  - Inference is faster (not as fast as GPU though)
  - Prompt parsing takes considerable time
- Hardware limitations for fine-tuning (and training)
  - Computing capacity (and memory bandwidth)
  - Almost impossible on CPU
  - GPU of Mac is not sufficient (limited number of cores)

Group Discussion: Choosing Suitable Hardware

- When do you need a GPU?
  - Generative LLM operations
  - Running cross-encoders in production
  - Frequently fine-tuning models
- Edge cases for a GPU
  - Information retrieval (often just encoding question)
  - Occasional fine-tunes (renting is a cheaper option)
- No GPU necessary
  - Testing inference with new models (buy a Mac)

Q&A

Summary

Fine-tuning for Different Models

- Distinguish between BERT and GPT
  - BERT models are used for classification and embeddings (and for NER)
  - GPT is used for generative LLMs
  - GPT fine-tuning is much more compute-intensive
- Decide if you need fine-tuning
  - Many models are available
  - Fine-tuning needs data, GPUs, and human resources
  - Pick a metric which helps you prove success
  - Cross-encoder as option to improve retrieval performance

Alternative: Retrieval Augmented Generation (RAG)

However, fine-tuning can tremendously improve RAG

Sources and References

The following references help you to get a deeper insight into the topic. The course itself should be self-contained but further reading and research will help you with new insights.

- Writing Effective Prompts for ChatGPT
- Prompt Engineering for LLMs
- What is RAG (Retrieval Augmented Generation) for an LLM?
- Hands-On Large Language Models
- What is Fine-Tuning of an LLM?
- Open Source Large Language Models in 3 Weeks
- Github for this course: https://github.com/datanizing/oreilly-finetuning-llm