<BEGIN>

👉 coursera, **Generative AI with Large Language Models**, Antje Barth +3 more instructors
👉 youtube, Generative AI with Large Language Models: Hands-On Training feat. Hugging Face and PyTorch Lightning, Jon Krohn (**NOT** part of the course but essential in my opinion)
👉 Deeplearning.AI and W&B, Evaluating and Debugging Generative AI (**NOT** part of the course but useful)

About this Course
In Generative AI with Large Language Models (LLMs), you'll learn **the fundamentals** of how generative AI works, and how to **deploy it in real-world applications**.

Week 1: Generative AI use cases, project lifecycle, and model pre-training, course slides
Week 2: **Fine-tuning and evaluating** large language models, course slides
Week 3: **Reinforcement learning** and LLM-powered applications, course slides

AWS Labs

👉 Introduction to AWS labs
https://youtu.be/9zWHhyFZ3Sc
👉 Lab 1 walkthrough
https://youtu.be/OOU_aFkJRe0
Python code in **Google Colab notebook** (view the Github copy)
👉 Lab 2 walkthrough
https://youtu.be/URVc_R-1Pks
Python code in **Google Colab notebook** (view the Github copy)
👉 Lab 3 walkthrough
https://youtu.be/x788bGCGYlg
Python code in **Google Colab notebook** (view the Github copy)

Lab 1 - Generative AI Use Case: Summarize Dialogue
In this lab, you will do the dialogue summarization task using generative AI. You will explore how the input text affects the output of the model, and perform prompt engineering to direct it towards the task you need. By comparing zero shot, one shot, and few shot inferences, you will take the first step towards prompt engineering and see how it can enhance the generative output of Large Language Models.

Setup the lab - evernote
Use the following command in the System terminal to download the lab:
$ aws s3 cp --recursive s3://dlai-generative-ai/labs/w1-549876/ ./
download: s3://dlai-generative-ai/labs/w1-549876/Lab_1_summarize_dialogue.ipynb to ./Lab_1_summarize_dialogue.ipynb
download: s3://dlai-generative-ai/labs/w1-549876/images/kernel_set_up.png to images/kernel_set_up.png
download: s3://dlai-generative-ai/labs/w1-549876/images/w1_kernel_and_instance_type.png to images/w1_kernel_and_instance_type.png

notebook 4.2 - Few Shot Inference
Token indices sequence length is longer than the specified maximum sequence length for this model (819 > 512). Running this sequence through the model will result in indexing errors

Lab 2 - Fine-tune a generative AI model for dialogue summarization
In this notebook, you will fine-tune an existing LLM from Hugging Face for enhanced dialogue summarization. You will use the FLAN-T5 model, which provides a high quality instruction tuned model and can summarize text out of the box. To improve the inferences, you will explore a full fine-tuning approach and evaluate the results with ROUGE metrics. Then you will perform PEFT fine-tuning, evaluate the resulting model and see that the benefits of PEFT outweigh the slightly-lower performance metrics.

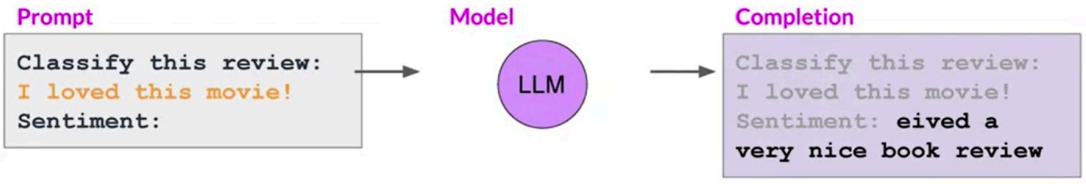
Use the following command in the System terminal to download the lab:
$ aws s3 cp --recursive s3://dlai-generative-ai/labs/w2-170864/ ./
download: s3://dlai-generative-ai/labs/w2-170864/Lab_2_fine_tune_generative_ai_model.ipynb to ./Lab_2_fine_tune_generative_ai_model.ipynb
download: s3://dlai-generative-ai/labs/w2-170864/images/kernel_set_up.png to images/kernel_set_up.png
download: s3://dlai-generative-ai/labs/w2-170864/images/w2_kernel_and_instance_type.png to images/w2_kernel_and_instance_type.png
download: s3://dlai-generative-ai/labs/w2-170864/data/dialogue-summary-training-results.csv to data/dialogue-summary-training-results.csv

Lab 3 - Fine-tune FLAN-T5 with reinforcement learning to generate more-positive summaries
In this notebook, you will fine-tune a FLAN-T5 model to generate less toxic content by Facebook's hate speech reward model. The reward model is a binary classifier that predicts either "not hate" or "hate" for the given text. You will use Proximal Policy Optimization (PPO) to fine-tune and detoxify the model.

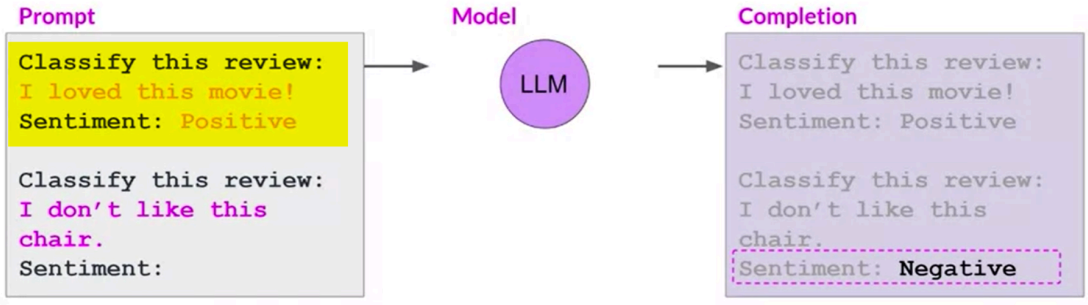
Use the following command in the System terminal to download the lab:
$ aws s3 cp --recursive s3://dlai-generative-ai/labs/w3-233794/ ./
download: s3://dlai-generative-ai/labs/w3-233794/images/w3_kernel_and_instance_type.png to images/w3_kernel_and_instance_type.png
download: s3://dlai-generative-ai/labs/w3-233794/Lab_3_fine_tune_model_to_detoxify_summaries.ipynb to ./Lab_3_fine_tune_model_to_detoxify_summaries.ipynb
download: s3://dlai-generative-ai/labs/w3-233794/images/kernel_set_up.png to images/kernel_set_up.png

Prompting
👉 zero shot, one shot, few shot inference

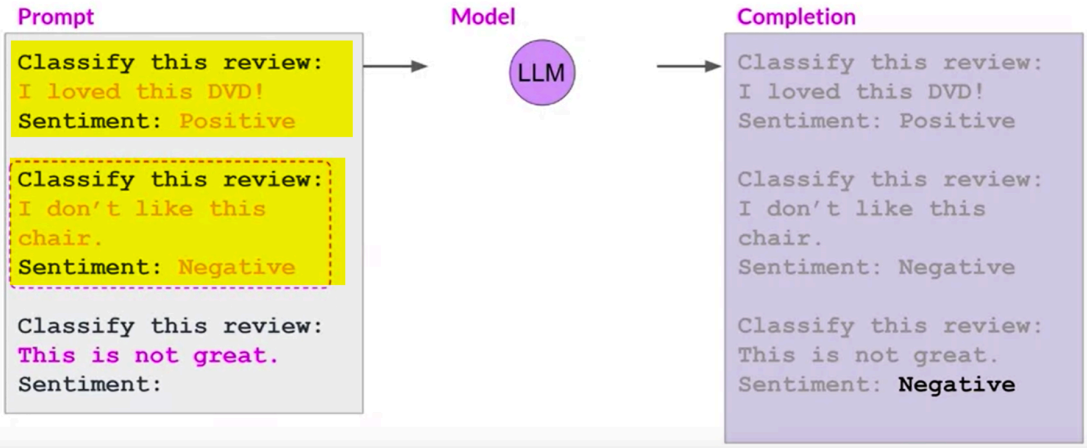
In-context learning (ICL) - zero shot inference



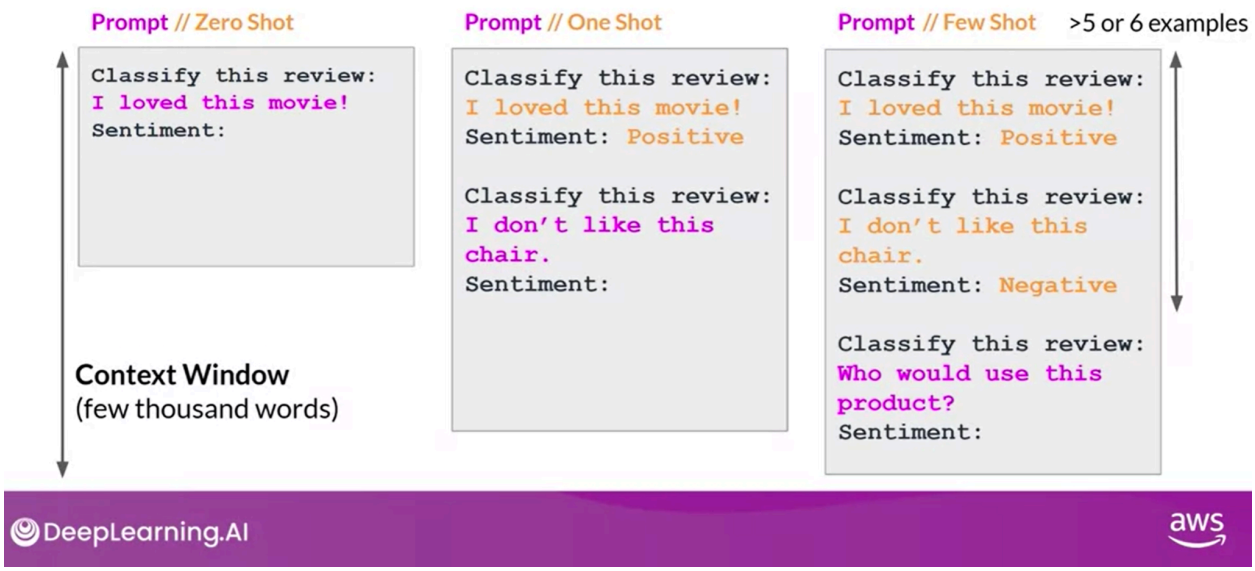
In-context learning (ICL) - one shot inference



In-context learning (ICL) - few shot inference

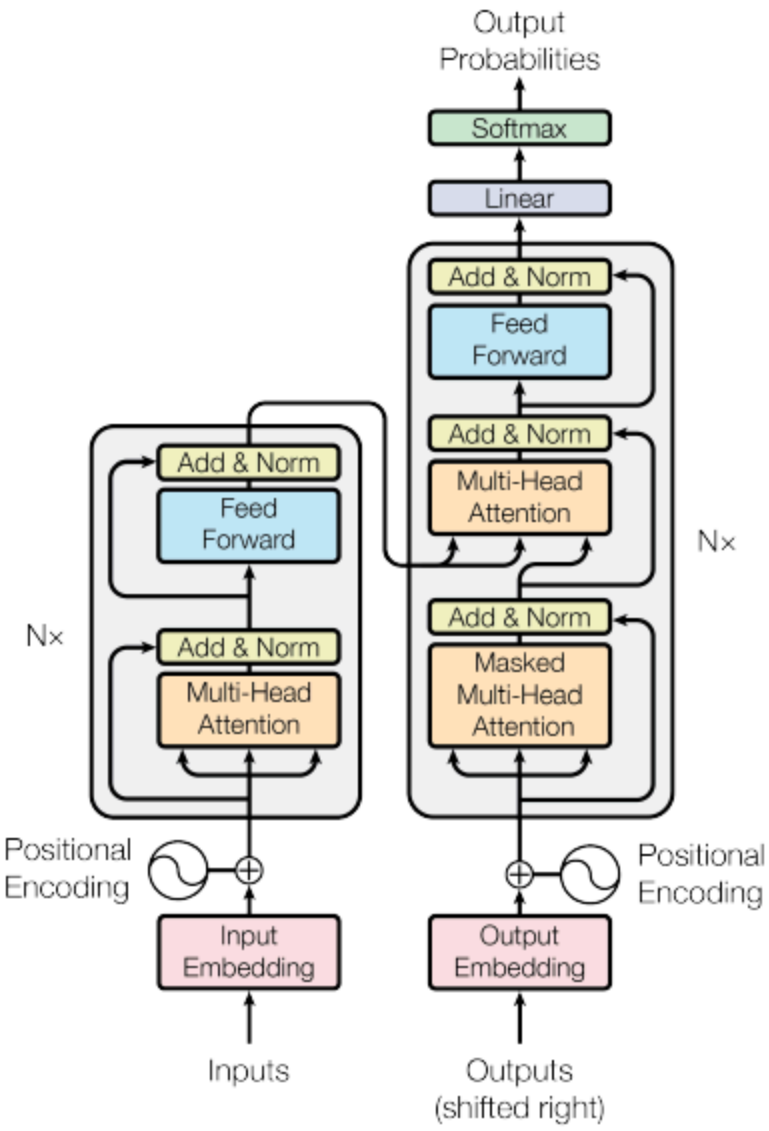


👉 if 5 or 6 shot inference doesn't work well, you should fine tune your model.



Transformer

👉 text generation before transformers
👉 Transformers: Attention is all you need
👉 youtube, **Transformer Neural Networks, ChatGPT's foundation, Clearly Explained!!!** by StatQuest with Josh Starmer (**NOT** part of the course but very helpful in my opinion)



"Attention is All You Need" is a research paper published in 2017 by Google researchers, which introduced the Transformer model, a novel architecture that revolutionized the field of natural language processing (NLP) and became the basis for the LLMs we now know - such as GPT, PaLM and others. The paper proposes a neural network architecture that replaces traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs) with an entirely attention-based mechanism.

The Transformer model uses self-attention to compute representations of input sequences, which allows it to capture long-term dependencies and parallelize computation effectively. The authors demonstrate that their model achieves state-of-the-art performance on several machine translation tasks and outperform previous models that rely on RNNs or CNNs.

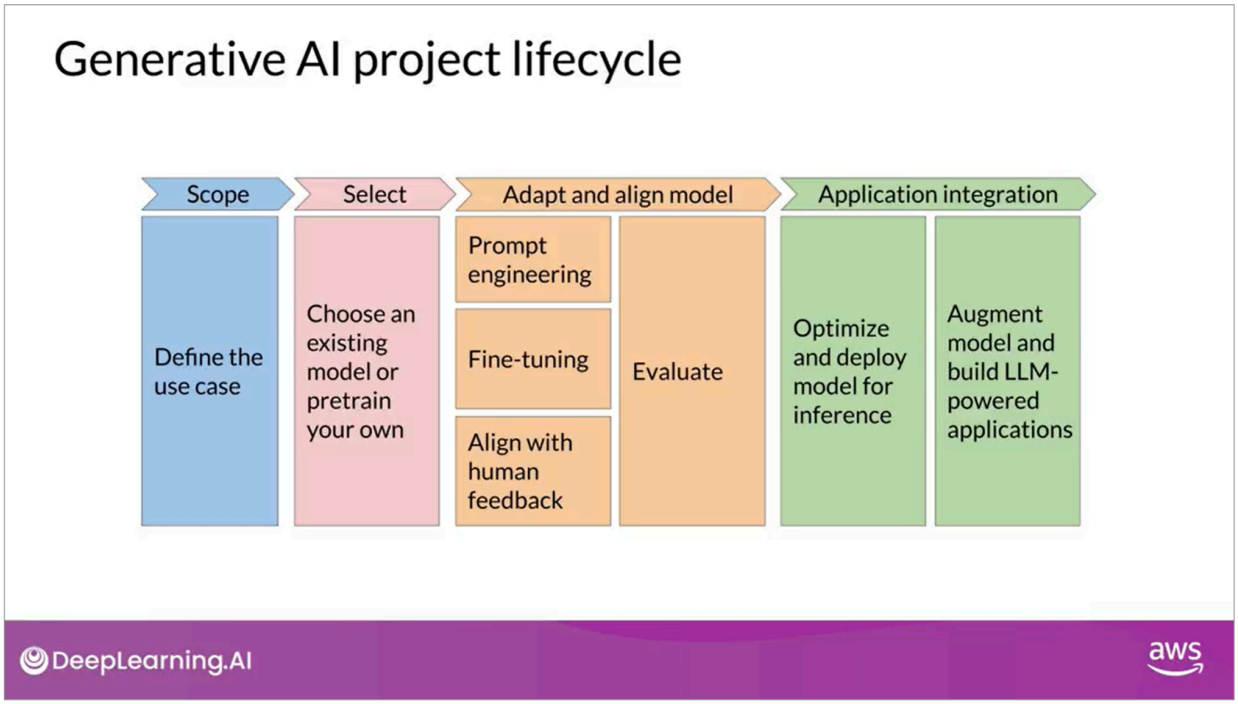
The Transformer architecture consists of an encoder and a decoder, each of which is composed of several layers. Each layer consists of two sub-layers: a multi-head self-attention mechanism

and a feed-forward neural network. The multi-head self-attention mechanism allows the model to attend to different parts of the input sequence, while the feed-forward network applies a point-wise fully connected layer to each position separately and identically.

The Transformer model also uses residual connections and layer normalization to facilitate training and prevent overfitting. In addition, the authors introduce a positional encoding scheme that encodes the position of each token in the input sequence, enabling the model to capture the order of the sequence without the need for recurrent or convolutional operations.

You can read the Transformers paper here.

Generative AI Project Lifecycle



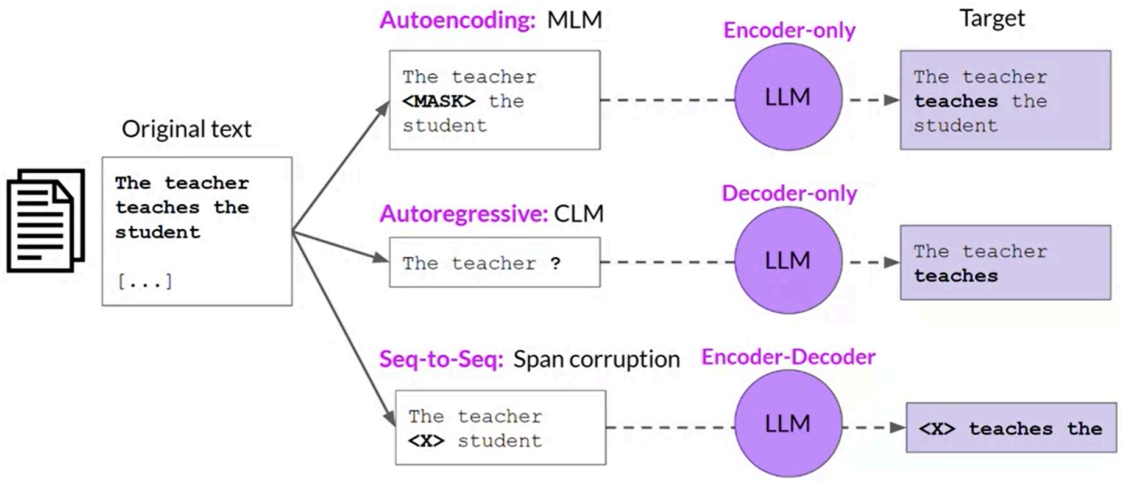
Choose your models

- encoder only models (auto-encoding)
- encoder-decoder models
- decoder only models (auto-regressive)

Pre-training large language models

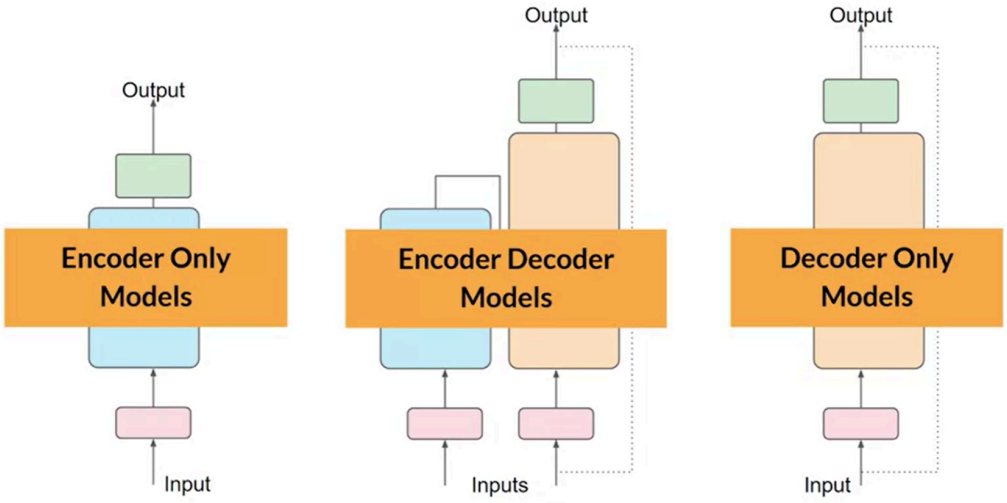
Notes

Model architectures and pre-training objectives



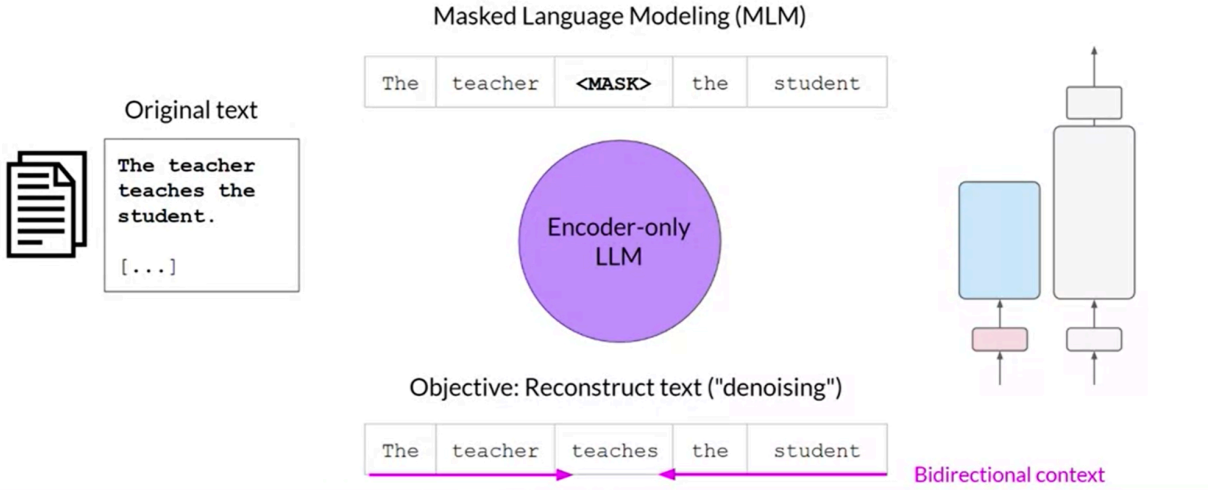
DeepLearning.AI aws

Transformers



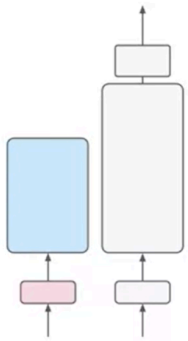
DeepLearning.AI aws

Autoencoding models



Good use cases:

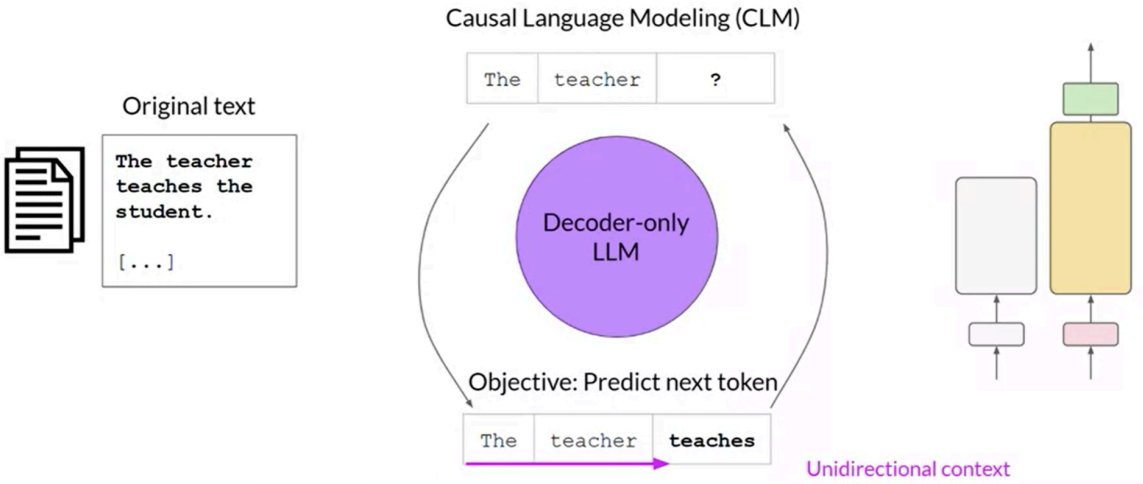
- Sentiment analysis
- Named entity recognition
- Word classification



Example models:

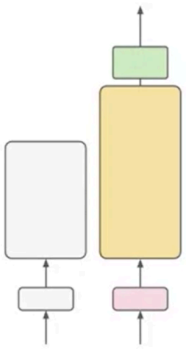
- BERT
- ROBERTA

Autoregressive models



Good use cases:

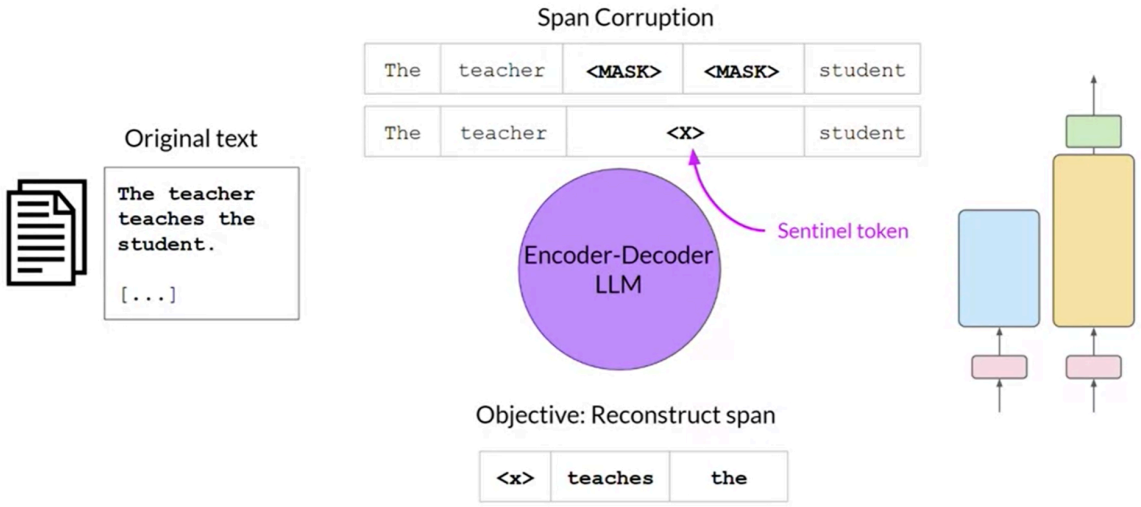
- Text generation
- Other emergent behavior
  - Depends on model size



Example models:

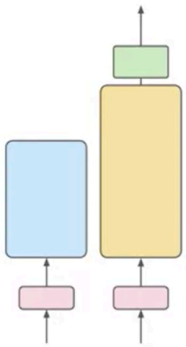
- GPT
- BLOOM

Sequence-to-sequence models



Good use cases:

- Translation
- Text summarization
- Question answering



Example models:

- T5
- BART

DeepLearning.AI aws

Computational challenges

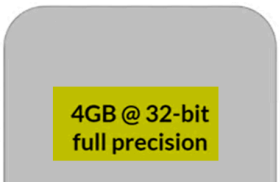
- computational challenges
- quantization
- multi-GPU compute strategies:
  Distributed Data Parallel (DDP)
  Fully Sharded Data Parallel (FSDP)

Computational challenges of training LLMs

Notes

Approximate GPU RAM needed to store 1B parameters

1 parameter = 4 bytes (32-bit float)
1B parameters = 4 x 10^9 bytes = 4GB



Additional GPU RAM needed to train 1B parameters

	Bytes per parameter
Model Parameters (Weights)	4 bytes per parameter
Adam optimizer (2 states)	+8 bytes per parameter
Gradients	+4 bytes per parameter
Activations and temp memory (variable size)	+8 bytes per parameter (high-end estimate)
TOTAL	**=4 bytes per parameter +20 extra bytes per parameter**

~20 extra bytes per parameter

Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

Approximate GPU RAM needed to train 1B-params

Memory needed to store model

4GB @ 32-bit
full precision

Memory needed to train model

Quantization: Summary

	Bits	Exponent	Fraction	Memory needed to store one value
FP32	32	8	23	4 bytes
FP16	16	5	10	2 bytes
BFLOAT16	16	8	7	2 bytes
INT8	8	–/–	7	1 byte

FLAN T5

- Reduce required memory to store and train models
- Projects original 32-bit floating point numbers into lower precision spaces
- Quantization-aware training (QAT) learns the quantization scaling factors during training
- BFLOAT16 is a popular choice

Approximate GPU RAM needed to train 1B-params

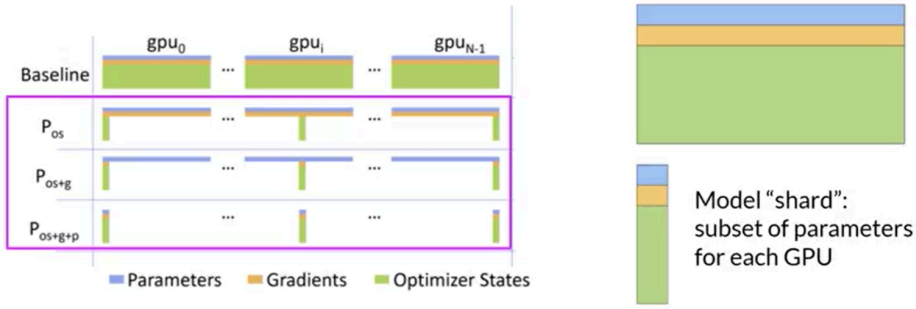
80GB is the maximum memory for the Nvidia A100 GPU, so to keep the model on a single GPU, you need to use 16-bit or 8-bit quantization.

GPU RAM needed to train larger models

As model sizes get larger, you will need to split your model across multiple GPUs for training

Zero Redundancy Optimizer (ZeRO)

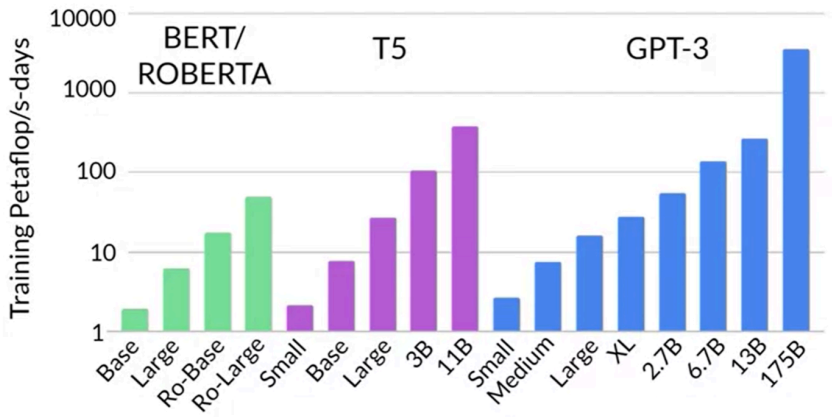
- Reduces memory by distributing (sharding) the model parameters, gradients, and optimizer states across GPUs



Sources:
Rajbhandari et al. 2019: "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models"
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

Scaling laws and compute-optimal models

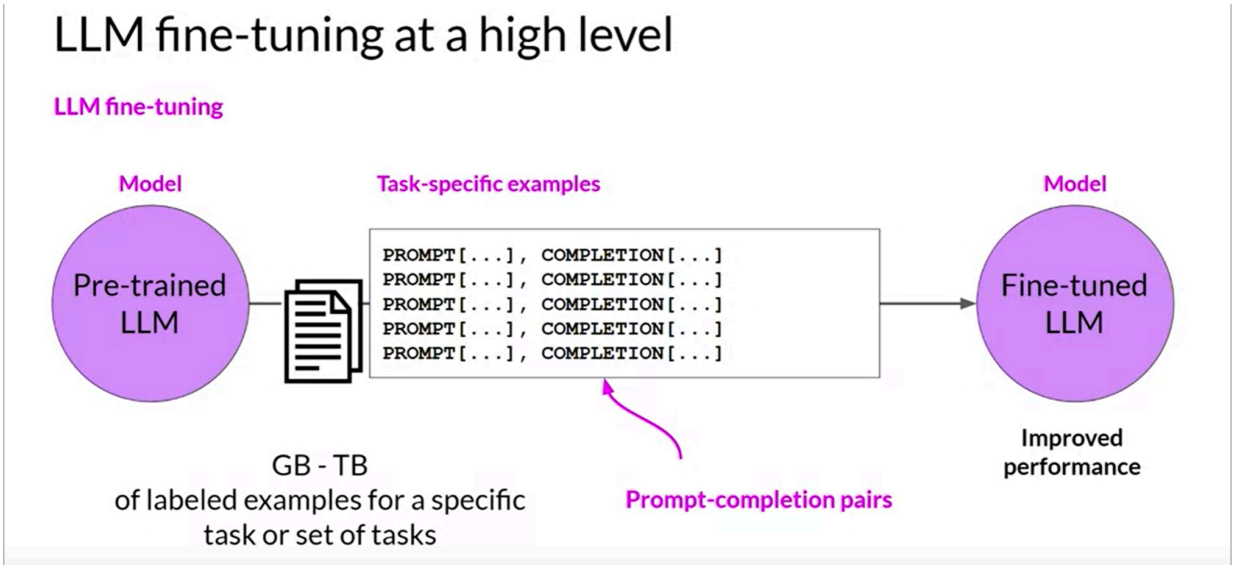
Number of petaflop/s-days to pre-train various LLMs

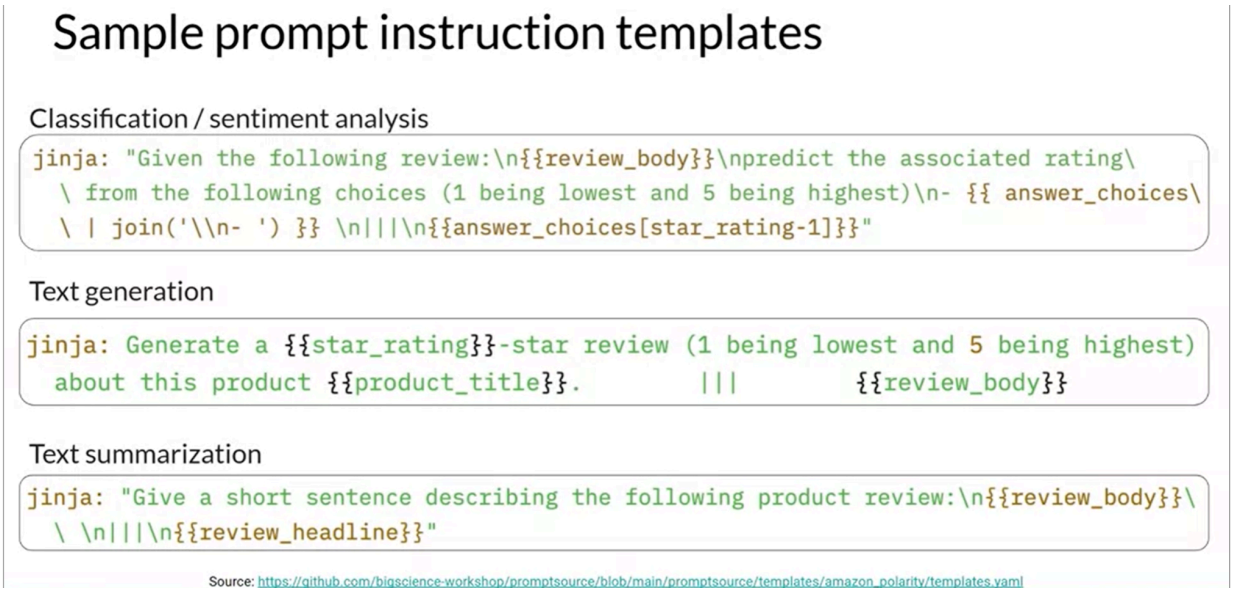


Source: Brown et al. 2020, "Language Models are Few-Shot Learners"

Fine-tuning

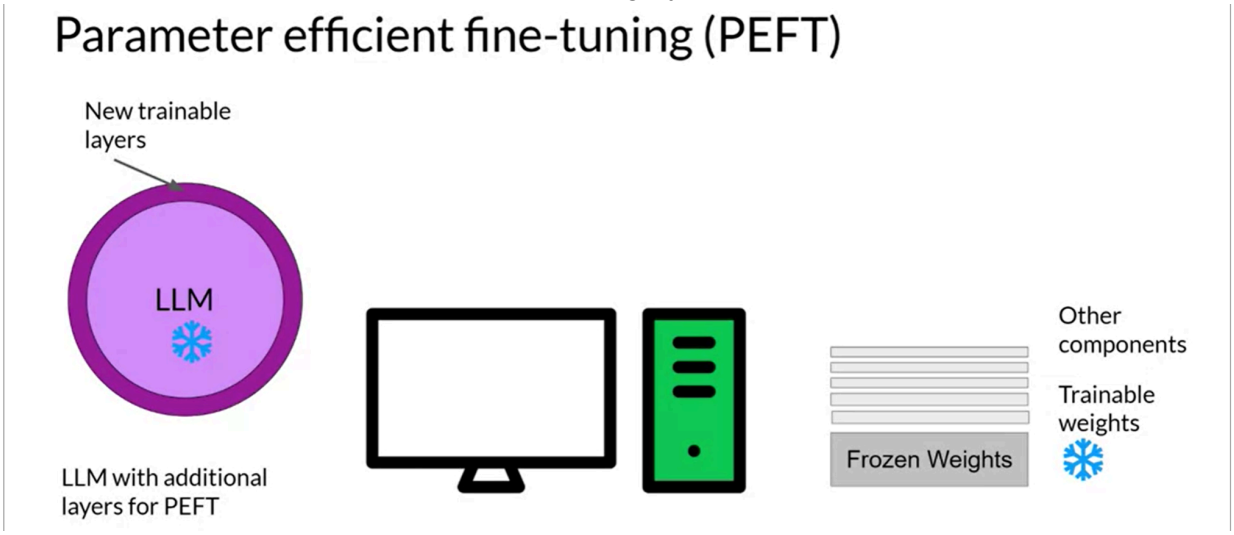
- **In-Context Learning** (ICL), e.g. few-shot prompting, might not work for smaller models, even with 5 or 6 examples being given
- fine-tuning of LLMs is a supervised learning process, using **prompt-completion pairs**
- in this course, fine-tuning almost always means **instruct** tuning (instruction fine-tuning)

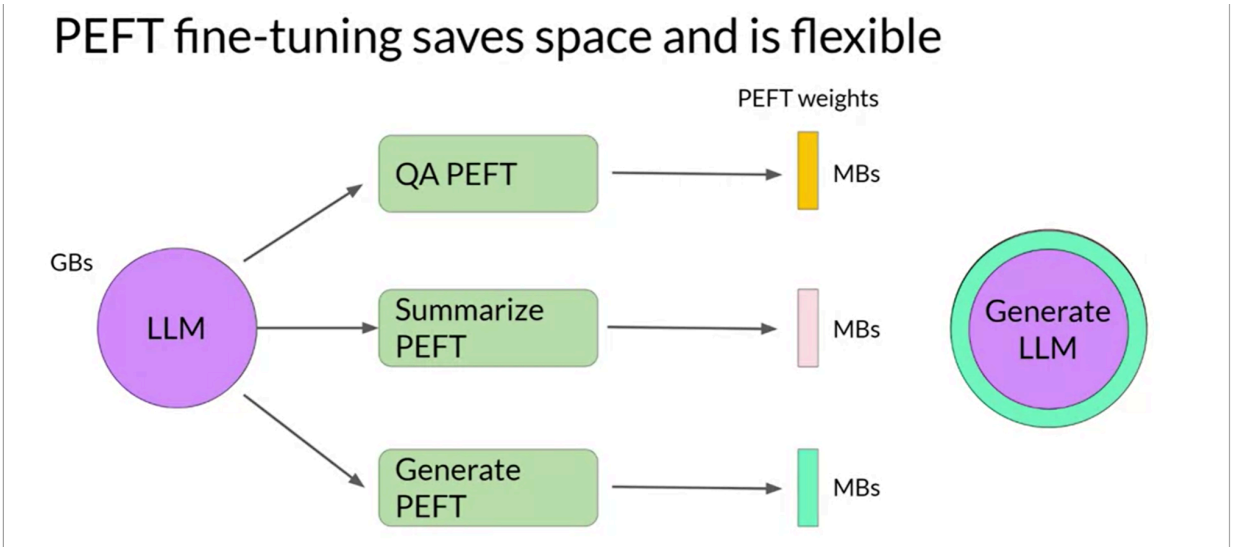


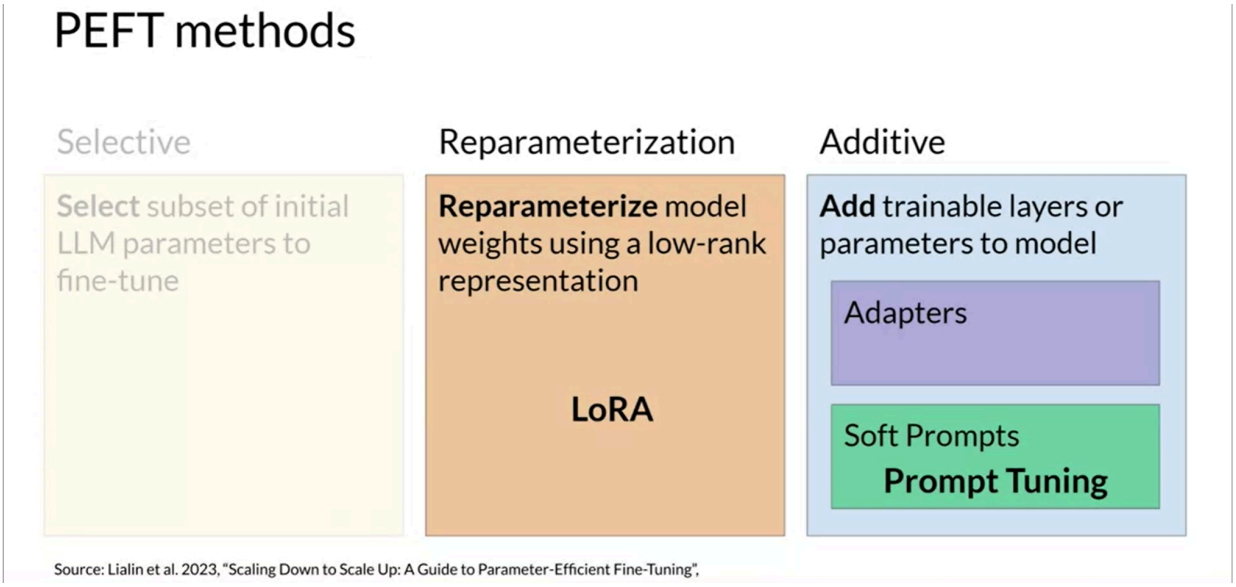


- "... there is a potential downside to fine-tuning on a single task. The process may lead to a phenomenon called **catastrophic forgetting**. Catastrophic forgetting happens because the full fine-tuning process modifies the weights of the original LLM. While this leads to great performance on the single fine-tuning task, it can degrade performance on other tasks." *(Me: "Well, this is very human intelligence-like. We learn one thing, and simultaneously forget about many other things." 🤷)*

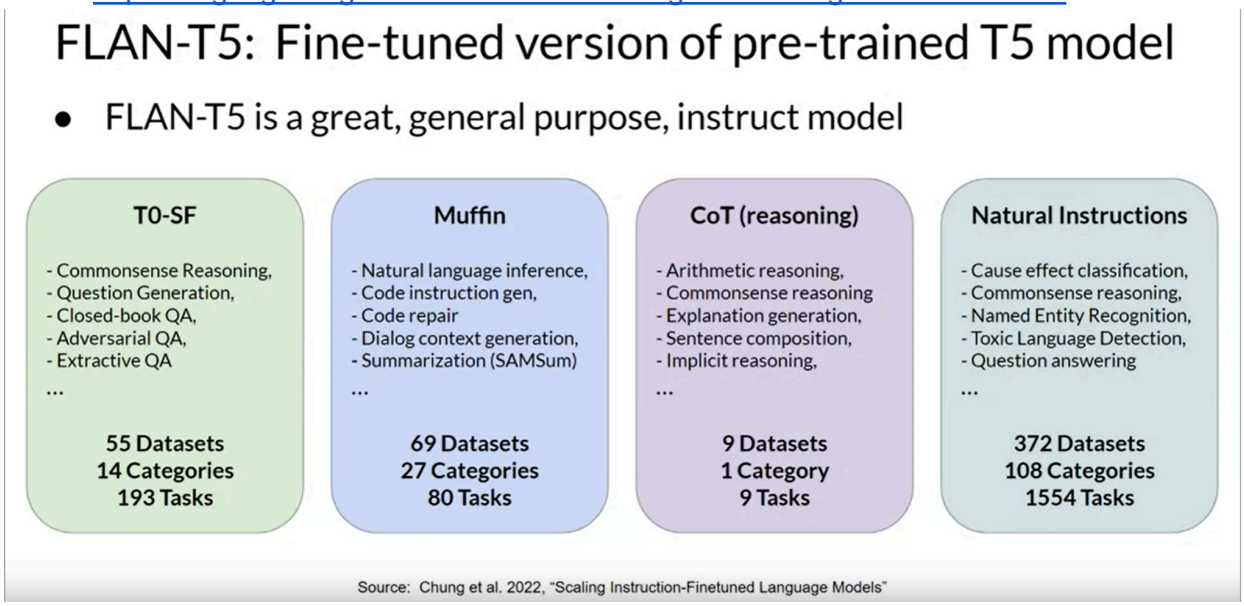
- “Our second option is to perform **parameter efficient fine-tuning**, or **PEFT** for short instead of full fine-tuning. PEFT is a set of techniques that preserves the weights of the original LLM and trains only a small number of task-specific adapter layers and parameters.”
- “**Low-rank Adaptation**, or **LoRA** for short, is a parameter-efficient fine-tuning technique that falls into the re-parameterization category.”







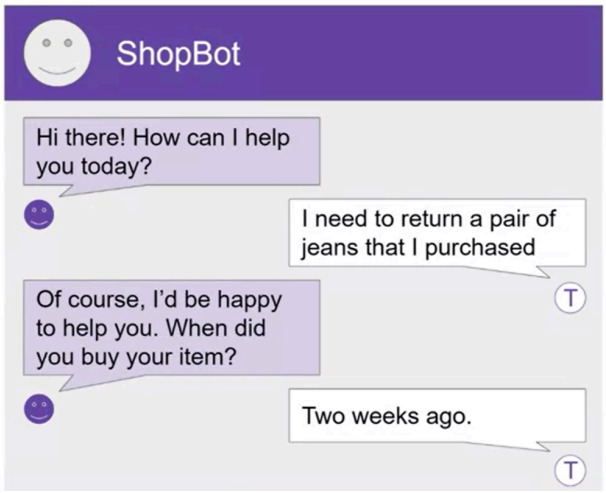
- multi-task fine-tuning needs a lot of data, which could be up to 50-100 thousands.
- **Fine-tuned LAnguage Net** (**FLAN**)
  https://ai.googleblog.com/2021/10/introducing-flan-more-generalizable.html



Sample FLAN-T5 prompt templates

```
"samsum": [
      ("{dialogue}\n\Briefly summarize that dialogue.", "{summary}"),
       ("Here is a dialogue:\n{dialogue}\n\nWrite a short summary!",
        "{summary}"),
       ("Dialogue:\n{dialogue}\n\nWhat is a summary of this dialogue?",
        "{summary}"),
       ("{dialogue}\n\nWhat was that dialogue about, in two sentences or less?",
        "{summary}"),
       ("Here is a dialogue:\n{dialogue}\n\nWhat were they talking about?",
        "{summary}"),
       ("Dialogue:\n{dialogue}\nWhat were the main points in that "
        "conversation?", "{summary}"),
       ("Dialogue:\n{dialogue}\nWhat was going on in that conversation?",
        "{summary}"),
]
```

Improving FLAN-T5's summarization capabilities



Goal: Summarize conversations to identify actions to take

Example support-dialog summarization

Prompt (created from template)

```
Summarize the following conversation.
Tommy: Hello. My name is Tommy Sandals, I have a reservation.
Mike: May I see some identification, sir, please?
Tommy: Sure. Here you go.
Mike: Thank you so much. Have you got a credit card, Mr.
Sandals?
Tommy: I sure do.
Mike: Thank you, sir. You'll be in room 507, nonsmoking,
queen bed.
Tommy: That's great, thank you!
Mike: Enjoy your stay!
```

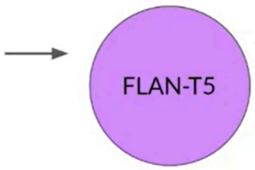
Source: https://huggingface.co/datasets/knkarthick/dialogsum/viewer/knkarthick--dialogsum/

Summary **before** fine-tuning FLAN-T5 with our dataset

Prompt (created from template)

```
Summarize the following
conversation.
Tommy: Hello. My name is
Tommy Sandals, I have a
reservation.
Mike: May I see some
...
...
...
Tommy: That's great, thank
you!
Mike: Enjoy your stay!
```

Model



Completion (Summary)

```
Tommy Sandals has a reservation
for a room at the Venetian
Hotel in Las Vegas.
```

Adequate completion, but does not match human baseline.

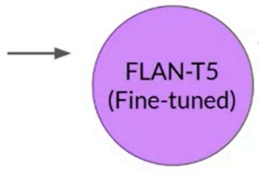
```
Human baseline summary:
Tommy Sandals has got a
reservation. Mike asks for his
identification and credit card
and helps his check-in.
```

Summary **after** fine-tuning FLAN-T5 with our dataset

Prompt (created from template)

```
Summarize the following
conversation.
Tommy: Hello. My name is
Tommy Sandals, I have a
reservation.
Mike: May I see some
...
...
...
Tommy: That's great, thank
you!
Mike: Enjoy your stay!
```

Model



Completion (Summary)

```
Tommy Sandals has a
reservation and checks in
showing his ID and credit
card. Mike helps him to
check in and approves his
reservation.
```

Better summary, more-closely matches human baseline.

- paper, Scaling Instruction-Finetuned Language Models

https://arxiv.org/pdf/2210.11416.pdf

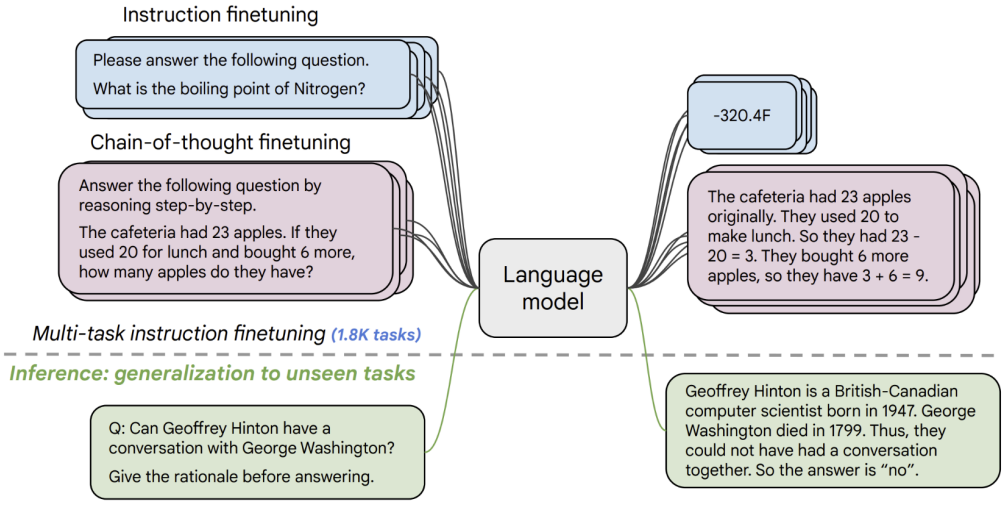
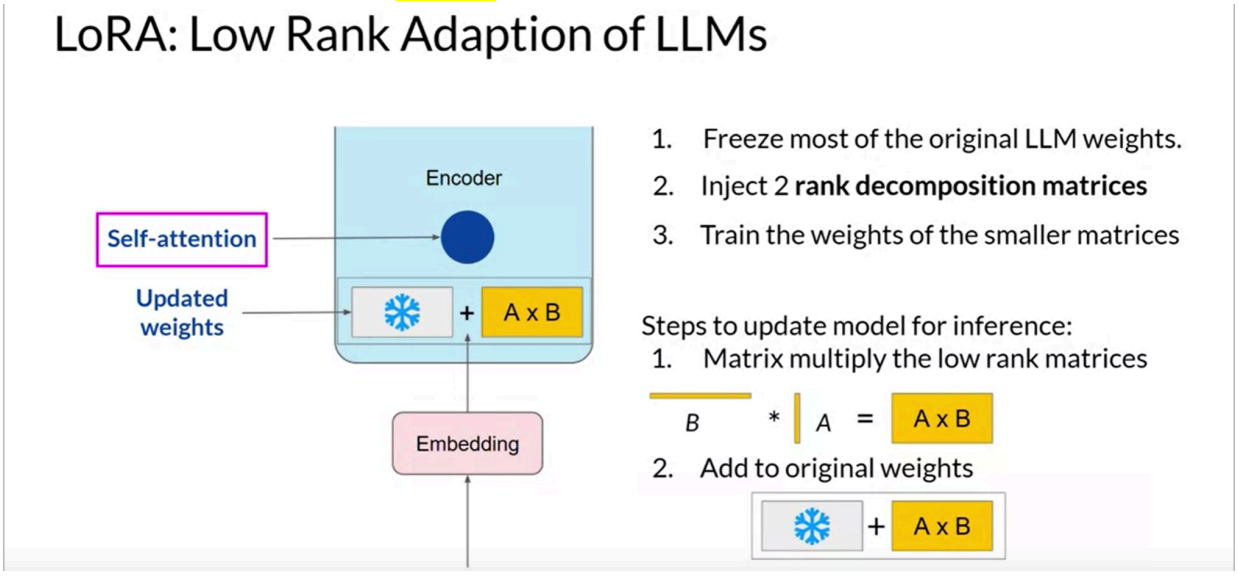


Figure 1: We finetune various language models on 1.8K tasks phrased as instructions, and evaluate them on unseen tasks. We finetune both with and without exemplars (i.e., zero-shot and few-shot) and with and without chain-of-thought, enabling generalization across a range of evaluation scenarios.

*Equal contribution. Correspondence: lehou@google.com.
†Core contributor.
[1]Public checkpoints: https://github.com/google-research/t5x/blob/main/docs/models.md#flan-t5-checkpoints.

- "Researchers have found that **applying LoRA to just the self-attention layers** of the model is often enough to fine-tune for a task and achieve performance gains. However, in principle, you can also use LoRA on other components like the feed-forward layers."
- LoRA + quantization -> **QLoRA**



Choosing the LoRA rank

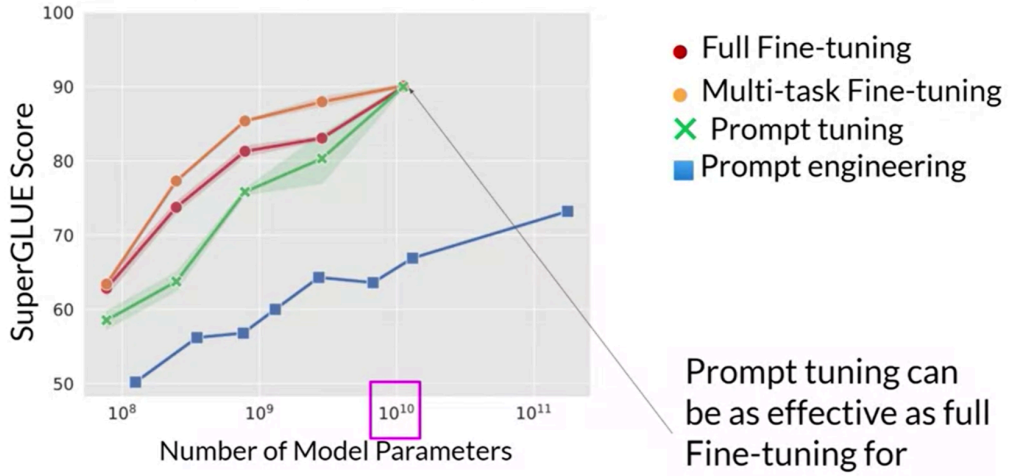
Rank r	val_loss	BLEU	NIST	METEOR	ROUGE_L	CIDEr
1	1.23	68.72	8.7215	0.4565	0.7052	2.4329
2	1.21	69.17	8.7413	0.4590	0.7052	2.4639
4	1.18	**70.38**	**8.8439**	**0.4689**	0.7186	**2.5349**
8	1.17	69.57	8.7457	0.4636	**0.7196**	2.5196
16	**1.16**	69.61	8.7483	0.4629	0.7177	2.4985
32	**1.16**	69.33	8.7736	0.4642	0.7105	2.5255
64	**1.16**	69.24	8.7174	0.4651	0.7180	2.5070
128	**1.16**	68.73	8.6718	0.4628	0.7127	2.5030
256	**1.16**	68.92	8.6982	0.4629	0.7128	2.5012
512	**1.16**	68.78	8.6857	0.4637	0.7128	2.5025
1024	1.17	69.37	8.7495	0.4659	0.7149	2.5090

- Effectiveness of higher rank appears to plateau
- Relationship between rank and dataset size needs more empirical data

Source: Hu et al. 2021, "LoRA: Low-Rank Adaptation of Large Language Models"

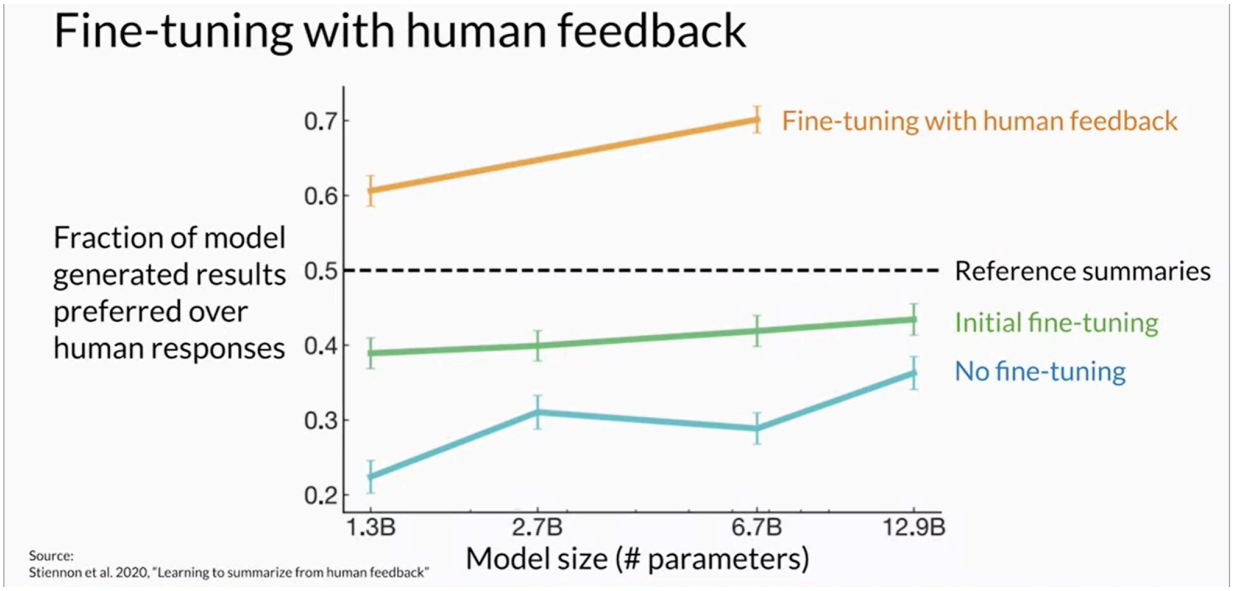
- **prompt tuning** (**NOT** prompt engineering)

Performance of prompt tuning

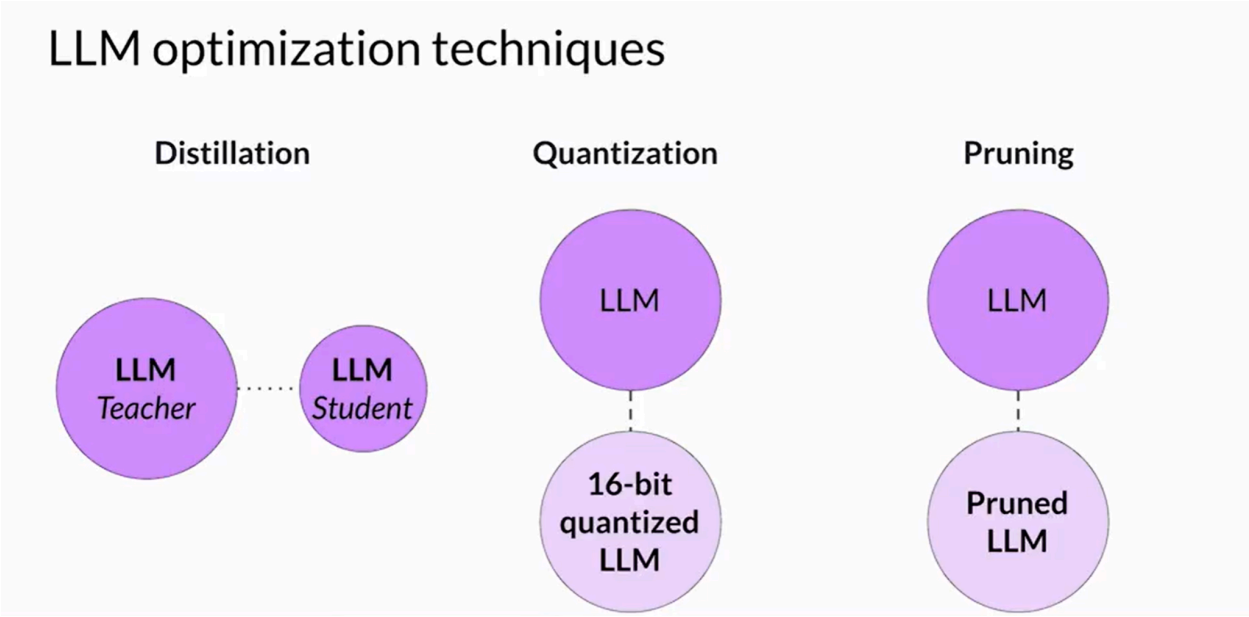


Source: Lester et al. 2021, "The Power of Scale for Parameter-Efficient Prompt Tuning"

Reinforcement Learning from Human Feedback (RLHF)

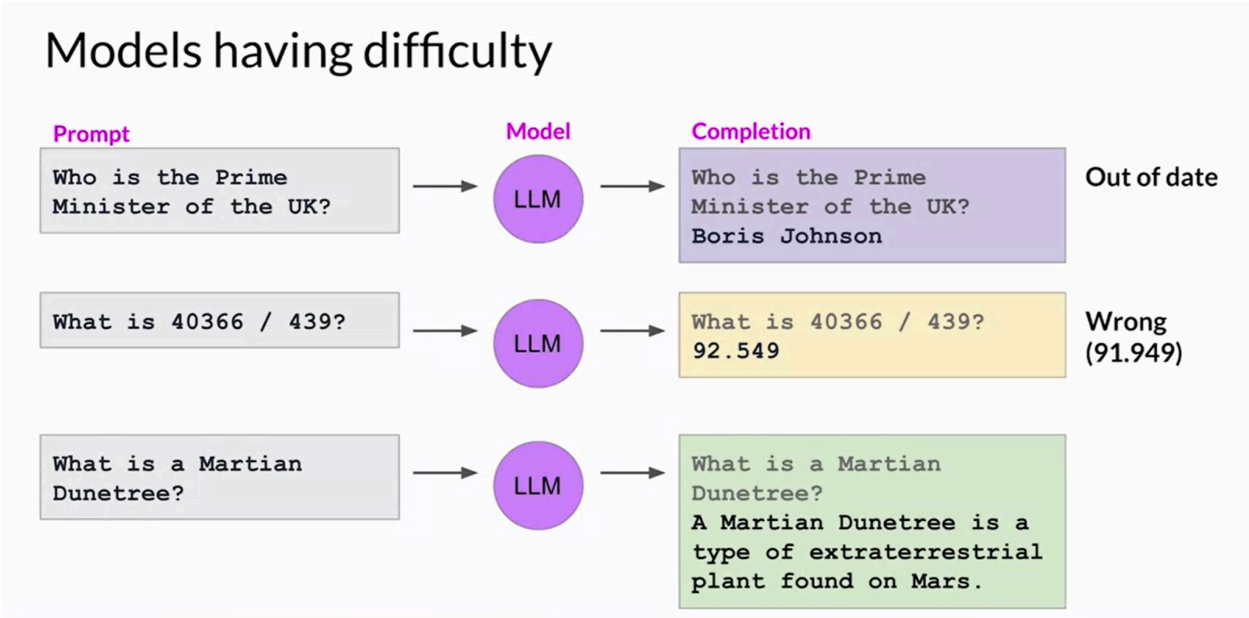


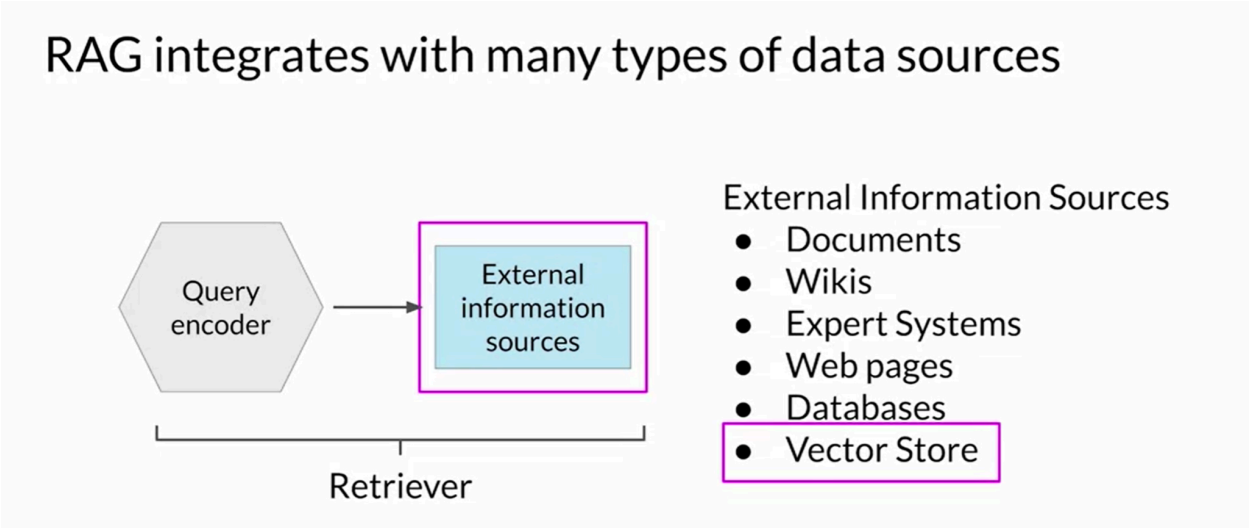
LLM optimization



Hallucination and RAG

- In this section, you'll learn about some techniques that you can use to help your LLM overcome these issues by **connecting to external data sources and applications**. You'll have a bit more work to do to be able to connect your LLM to these external components and fully integrate everything for deployment within your application.
- **Retrieval Augmented Generation, or RAG** for short, is a framework for building LLM powered systems that make use of external data sources and applications to overcome some of the limitations of these models. RAG is a great way to overcome the knowledge cutoff issue and help the model update its understanding of the world.





Reading (a lot of reading)

Week 1 Resources

Below you'll find links to the research papers discussed in this week's videos. You don't need to understand all the technical details discussed in these papers - **you have already seen the most important points you'll need to answer the quizzes** in the lecture videos.
However, if you'd like to take a closer look at the original research, you can read the papers and articles via the links below.

Transformer Architecture

- **Attention is All You Need** - This paper introduced the Transformer architecture, with the core "self-attention" mechanism. This article was the foundation for LLMs.
- **BLOOM: BigScience 176B Model** - BLOOM is an open-source LLM with 176B parameters (similar to GPT-4) trained in an open and transparent way. In this paper, the authors present a detailed discussion of the dataset and process used to train the model. You can also see a high-level overview of the model here.
- **Vector Space Models** - Series of lessons from DeepLearning.AI's Natural Language Processing specialization discussing the basics of vector space models and their use in language modeling.

Pre-training and scaling laws

- **Scaling Laws for Neural Language Models** - empirical study by researchers at OpenAI exploring the scaling laws for large language models.

Model architectures and pre-training objectives

- **What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization?** - The paper examines modeling choices in large pre-trained language models and identifies the optimal approach for zero-shot generalization.
- **HuggingFace Tasks** **and** **Model Hub** - Collection of resources to tackle varying machine learning tasks using the HuggingFace library.
- **LLaMA: Open and Efficient Foundation Language Models** - Article from Meta AI proposing Efficient LLMs (their model with 13B parameters outperform GPT3 with 175B parameters on most benchmarks)

Scaling laws and compute-optimal models

- **Language Models are Few-Shot Learners** - This paper investigates the potential of few-shot learning in Large Language Models.
- **Training Compute-Optimal Large Language Models** - Study from DeepMind to evaluate the optimal model size and number of tokens for training LLMs. Also known as "Chinchilla Paper".
- **BloombergGPT: A Large Language Model for Finance** - LLM trained specifically for the finance domain, a good example that tried to follow chinchilla laws.

Week 2 Resources

Below you'll find links to the research papers discussed in this week's videos. You don't need to understand all the technical details discussed in these papers - **you have already seen the most important points you'll need to answer the quizzes** in the lecture videos.
However, if you'd like to take a closer look at the original research, you can read the papers and articles via the links below.

Multi-task, instruction fine-tuning

- **Scaling Instruction-Finetuned Language Models** - Scaling fine-tuning with a focus on task, model size and chain-of-thought data.
- **Introducing FLAN: More generalizable Language Models with Instruction Fine-Tuning** - This blog (and article) explores instruction fine-tuning, which aims to make language models better at performing NLP tasks with zero-shot inference.

Model Evaluation Metrics

- **HELM - Holistic Evaluation of Language Models** - HELM is a living benchmark to evaluate Language Models more transparently.
- **General Language Understanding Evaluation (GLUE) benchmark** - This paper introduces GLUE, a benchmark for evaluating models on diverse natural language understanding (NLU) tasks and emphasizing the importance of improved general NLU systems.
- **SuperGLUE** - This paper introduces SuperGLUE, a benchmark designed to evaluate the performance of various NLP models on a range of challenging language understanding tasks.
- **ROUGE: A Package for Automatic Evaluation of Summaries** - This paper introduces and evaluates four different measures (ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S) in the ROUGE summarization evaluation package, which assess the quality of summaries by comparing them to ideal human-generated summaries.
- **Measuring Massive Multitask Language Understanding (MMLU)** - This paper presents a new test to measure multitask accuracy in text models, highlighting the need for substantial improvements in achieving expert-level accuracy and addressing lopsided performance and low accuracy on socially important subjects.
- **BigBench-Hard - Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models** - The paper introduces BIG-bench, a benchmark for evaluating language models on challenging tasks, providing insights on scale, calibration, and social bias.

Parameter- efficient fine tuning (PEFT)

- **Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning** - This paper provides a systematic overview of Parameter-Efficient Fine-tuning (PEFT) Methods in all three categories discussed in the lecture videos.

- **On the Effectiveness of Parameter-Efficient Fine-Tuning** - The paper analyzes sparse fine-tuning methods for pre-trained models in NLP.

LoRA

- **LoRA Low-Rank Adaptation of Large Language Models** - This paper proposes a parameter-efficient fine-tuning method that makes use of low-rank decomposition matrices to reduce the number of trainable parameters needed for fine-tuning language models.
- **QLoRA: Efficient Finetuning of Quantized LLMs** - This paper introduces an efficient method for fine-tuning large language models on a single GPU, based on quantization, achieving impressive results on benchmark tests.

Prompt tuning with soft prompts

- **The Power of Scale for Parameter-Efficient Prompt Tuning** - The paper explores "prompt tuning," a method for conditioning language models with learned soft prompts, achieving competitive performance compared to full fine-tuning and enabling model reuse for many tasks.

Week 3 Resources

Below you'll find links to the research papers discussed in this week's videos. You don't need to understand all the technical details discussed in these papers - **you have already seen the most important points you'll need to answer the quizzes** in the lecture videos.
However, if you'd like to take a closer look at the original research, you can read the papers and articles via the links below.

Reinforcement Learning from Human-Feedback (RLHF)

- **Training language models to follow instructions with human feedback -** Paper by OpenAI introducing a human-in-the-loop process to create a model that is better at following instructions (InstructGPT).
- **Learning to summarize from human feedback** - This paper presents a method for improving language model-generated summaries using a reward-based approach, surpassing human reference summaries.

Proximal Policy Optimization (PPO)

- **Proximal Policy Optimization Algorithms** - The paper from researchers at OpenAI that first proposed the PPO algorithm. The paper discusses the performance of the algorithm on a number of benchmark tasks including robotic locomotion and game play.
- **Direct Preference Optimization: Your Language Model is Secretly a Reward Model** - This paper presents a simpler and effective method for precise control of large-scale unsupervised language models by aligning them with human preferences.

Scaling human feedback

- **Constitutional AI: Harmlessness from AI Feedback** - This paper introduces a method for training a harmless AI assistant without human labels, allowing better control of AI behavior with minimal human input.

Advanced Prompting Techniques

- **Chain-of-thought Prompting Elicits Reasoning in Large Language Models** - Paper by researchers at Google exploring how chain-of-thought prompting improves the ability of LLMs to perform complex reasoning.
- **PAL: Program-aided Language Models** - This paper proposes an approach that uses the LLM to read natural language problems and generate programs as the intermediate reasoning steps.
- **ReAct: Synergizing Reasoning and Acting in Language Models** This paper presents an advanced prompting technique that allows an LLM to make decisions about how to interact with external applications.

LLM powered application architectures

- **LangChain Library (GitHub)** - This library is aimed at assisting in the development of those types of applications, such as Question Answering, Chatbots and other Agents. You can read the documentation here.
- **Who Owns the Generative AI Platform?** - The article examines the market dynamics and business models of generative AI.

Course Information

Generative AI with Large Language Models
by DeepLearning.AI & Amazon Web Services

About this Course
In Generative AI with Large Language Models (LLMs), you'll learn the fundamentals of how generative AI works, and how to deploy it in real-world applications. By taking this course, you'll learn to: - Deeply understand generative AI, describing the key steps in a typical LLM-based generative AI lifecycle, from data gathering and model selection, to performance evaluation and deployment - Describe in detail the transformer architecture that powers LLMs, how they're trained, and how fine-tuning enables LLMs to be adapted to a variety of specific use cases - Use empirical scaling laws to optimize the model's objective function across dataset size, compute budget, and inference requirements - Apply state-of-the art training, tuning, inference, tools, and deployment methods to maximize the performance of models within the specific constraints of your project - Discuss the challenges and opportunities that generative AI creates for businesses after hearing stories from industry researchers and practitioners Developers who have a good foundational understanding of how LLMs work, as well the best practices behind training and deploying them, will be able to make good decisions for their companies and more quickly build working prototypes. This course will support learners in building practical intuition about how to best utilize this exciting new technology. This is an intermediate course, so you should have some experience coding in Python to get the most out of it. You should also be familiar with the basics of machine learning, such as supervised and unsupervised learning, loss functions, and splitting data into training, validation, and test sets. If you have taken the Machine Learning Specialization or Deep Learning Specialization from DeepLearning.AI, you'll be ready to take this course and dive deeper into the fundamentals of generative AI.

Taught by: Antje Barth, Instructor
Principal Developer Advocate, Generative AI, Amazon Web Services (AWS)

Taught by: Chris Fregly, Instructor
Principal Solutions Architect, Generative AI, Amazon Web Services (AWS)

Taught by: Shelbee Eigenbrode, Instructor
Principal Solutions Architect, Generative AI, Amazon Web Services (AWS)

Taught by: Mike Chambers, Instructor
Developer Advocate, Generative AI, Amazon Web Services (AWS)

Level	Intermediate
Commitment	3 weeks of study, 3-4 hours/week
Language	English
How To Pass	Pass all graded assignments to complete the course.
User Ratings	Average User Rating 4.8

Syllabus

Week 1

Generative AI use cases, project lifecycle, and model pre-training
17 videos, 5 readings

1. **Video:** Course Introduction
2. **Reading:** Contributor Acknowledgments
3. **Video:** Introduction - Week 1
4. **Video:** Generative AI & LLMs
5. **App Item:** [IMPORTANT] Have questions, issues or ideas? Join our Community!
6. **Video:** LLM use cases and tasks
7. **Video:** Text generation before transformers
8. **Video:** Transformers architecture
9. **Video:** Generating text with transformers
10. **Reading:** Transformers: Attention is all you need
11. **Video:** Prompting and prompt engineering
12. **Video:** Generative configuration
13. **Video:** Generative AI project lifecycle
14. **Video:** Introduction to AWS labs
15. **Video:** Lab 1 walkthrough
16. **Video:** Pre-training large language models
17. **Video:** Computational challenges of training LLMs
18. **Video:** Optional video: Efficient multi-GPU compute strategies
19. **Video:** Scaling laws and compute-optimal models
20. **Video:** Pre-training for domain adaptation
21. **Reading:** Domain-specific training: BloombergGPT
22. **Reading:** Week 1 resources
23. **Reading:** Lecture Notes Week 1

Graded: Lab 1 - Generative AI Use Case: Summarize Dialogue
Graded: Week 1 quiz

Week 2

Fine-tuning and evaluating large language models
10 videos, 3 readings

1. **Video:** Introduction - Week 2
2. **Video:** Instruction fine-tuning
3. **Video:** Fine-tuning on a single task
4. **Video:** Multi-task instruction fine-tuning
5. **Reading:** Scaling instruct models
6. **Video:** Model evaluation
7. **Video:** Benchmarks
8. **Video:** Parameter efficient fine-tuning (PEFT)
9. **Video:** PEFT techniques 1: LoRA
10. **Video:** PEFT techniques 2: Soft prompts
11. **Video:** Lab 2 walkthrough
12. **Reading:** Week 2 Resources
13. **Reading:** Lecture Notes Week 2

Graded: Lab 2 - Fine-tune a generative AI model for dialogue summarization

Graded: Week 2 quiz

Week 3

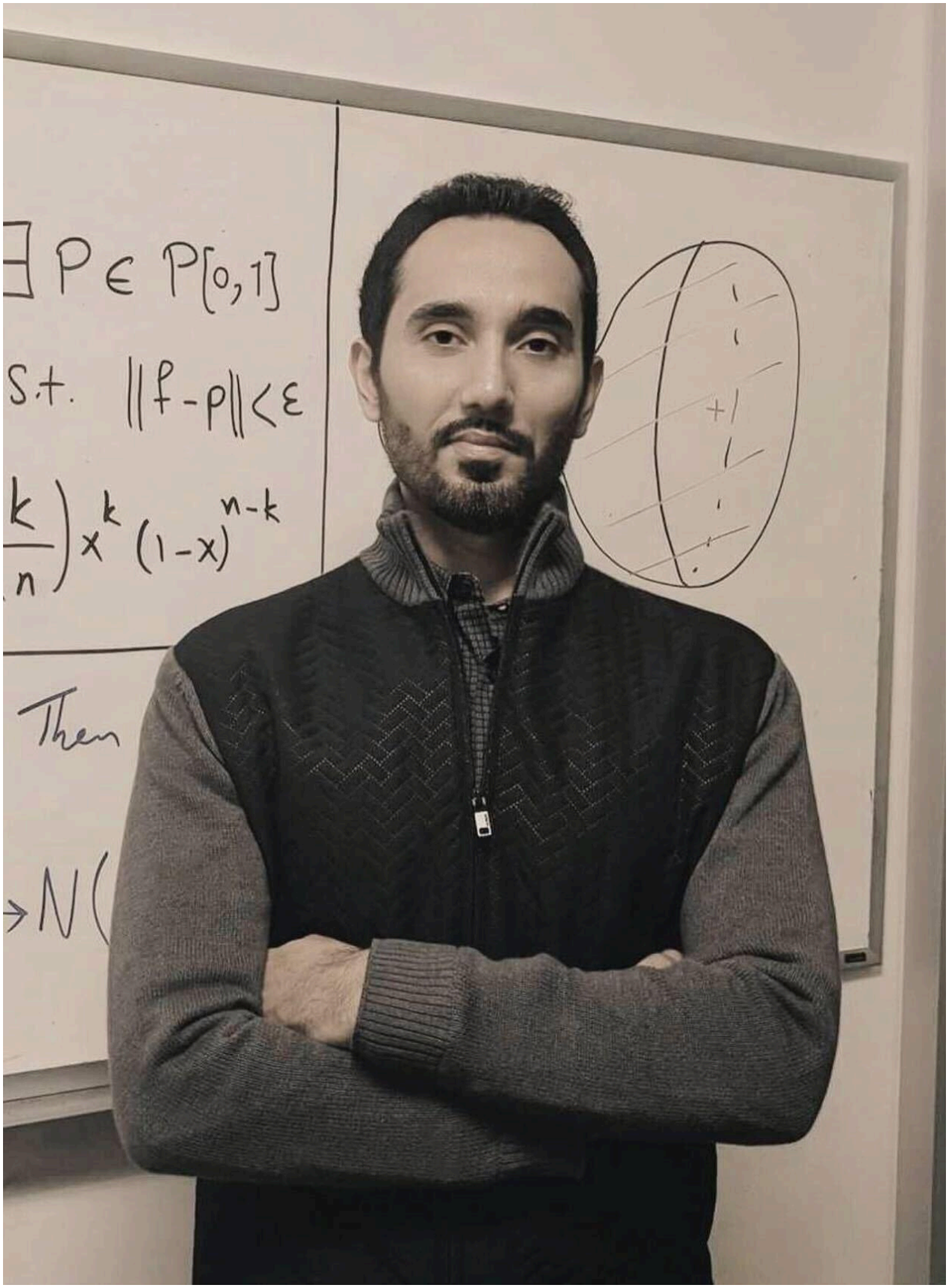
Reinforcement learning and LLM-powered applications
21 videos, 6 readings

1. **Video**: Introduction - Week 3
2. **Video:** Aligning models with human values
3. **Video:** Reinforcement learning from human feedback (RLHF)
4. **Video:** RLHF: Obtaining feedback from humans
5. **Video:** RLHF: Reward model
6. **Video:** RLHF: Fine-tuning with reinforcement learning
7. **Video:** Optional video: Proximal policy optimization
8. **Video:** RLHF: Reward hacking
9. **Reading:** KL divergence
10. **Video:** Scaling human feedback
11. **Video:** Lab 3 walkthrough
12. **Video:** Model optimizations for deployment
13. **Video:** Generative AI Project Lifecycle Cheat Sheet
14. **Video:** Using the LLM in applications
15. **Video:** Interacting with external applications
16. **Video:** Helping LLMs reason and plan with chain-of-thought
17. **Video:** Program-aided language models (PAL)
18. **Video:** ReAct: Combining reasoning and action
19. **Reading:** ReAct: Reasoning and action
20. **Video:** LLM application architectures
21. **Video:** Optional video: AWS Sagemaker JumpStart
22. **Reading:** Week 3 resources
23. **Video:** Responsible AI
24. **Video:** Course conclusion
25. **Reading:** Lecture Notes Week 3
26. **Reading:** Acknowledgments
27. **Reading:** (Optional) Opportunity to Mentor Other Learners

Graded: Lab 3 - Fine-tune FLAN-T5 with reinforcement learning to generate more-positive summaries
Graded: Week 3 Quiz

Contributor Acknowledgments

We would like to thank the following subject matter experts for their contributions to the development of this course:
Ehsan Kamalinejad, Ph.D.
Machine Learning Applied Scientist, AWS


∃ P ∈ P[0,1]
s.t. ||f - p|| < ε
Then

Dr. Ehsan Kamalinejad (EK) is a Machine Learning Applied Scientist working on NLP developments at Amazon. Previously he co-founded Visual One, a YCombinator startup in computer vision. Before that, he was a Tech-Lead Machine Learning Engineer at Apple, working on projects such as Memories. EK is also an Associate Professor of Mathematics at California State University East Bay.

Nashlie Sephus, Ph.D.

Principal Technology Evangelist for Amazon AI, AWS

Dr. Nashlie Sephus is a Principal Technology Evangelist for Amazon AI at AWS. In this role, she focuses on fairness and accuracy as well as identifying and mitigating potential biases in artificial intelligence. She formerly led the Amazon Visual Search team as an Applied Scientist in Atlanta which launched the visual search for replacement parts on the Amazon Shopping app.

Heiko Hotz

Senior Solutions Architect for AI & Machine Learning, AWS

Heiko Hotz is a Senior Solutions Architect for AI & Machine Learning with a special focus on natural language processing (NLP), large language models (LLMs), and generative AI. Prior to this role, he was the Head of Data Science for Amazon's EU Customer Service. Heiko helps companies be successful in their AI/ML journey on AWS and has worked with organizations in many industries,

including insurance, financial services, media and entertainment, healthcare, utilities, and manufacturing. In his spare time, Heiko travels as much as possible.

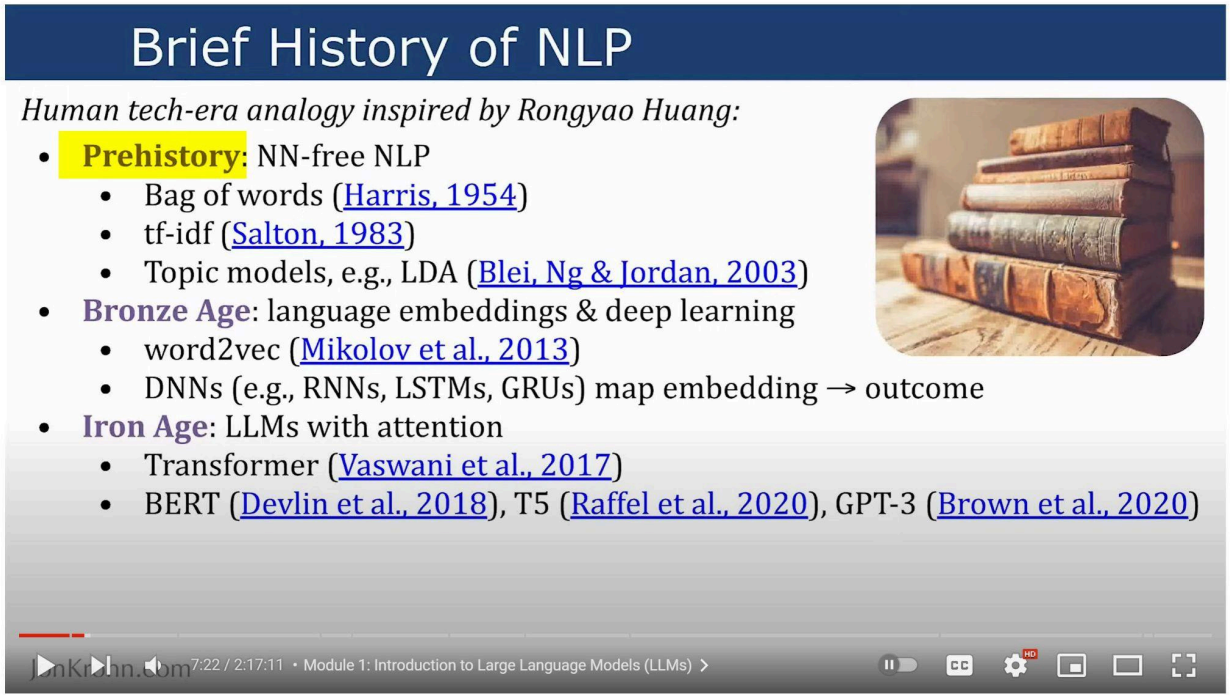
Philipp Schmid

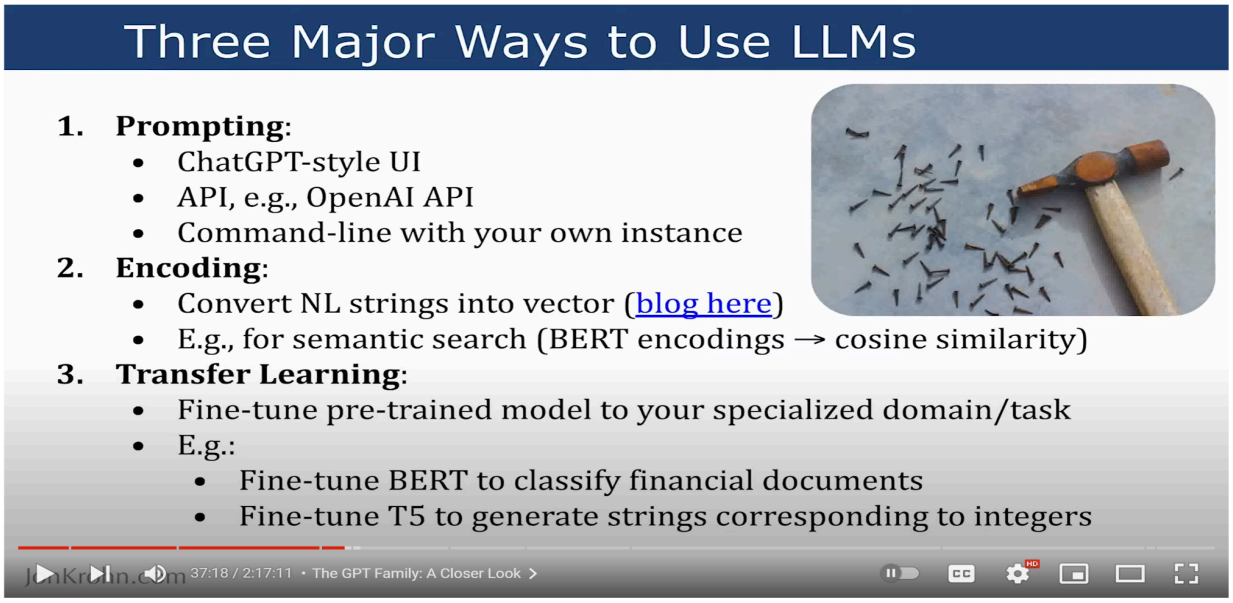
Technical Lead, Hugging Face

Philipp Schmid is a Technical Lead at Hugging Face with the mission to democratize good machine learning through open source and open science. Philipp is passionate about productionizing cutting-edge and generative AI machine learning models.

YouTube, Generative AI with Large Language Models: Hands-On Training feat. Hugging Face and PyTorch Lightning

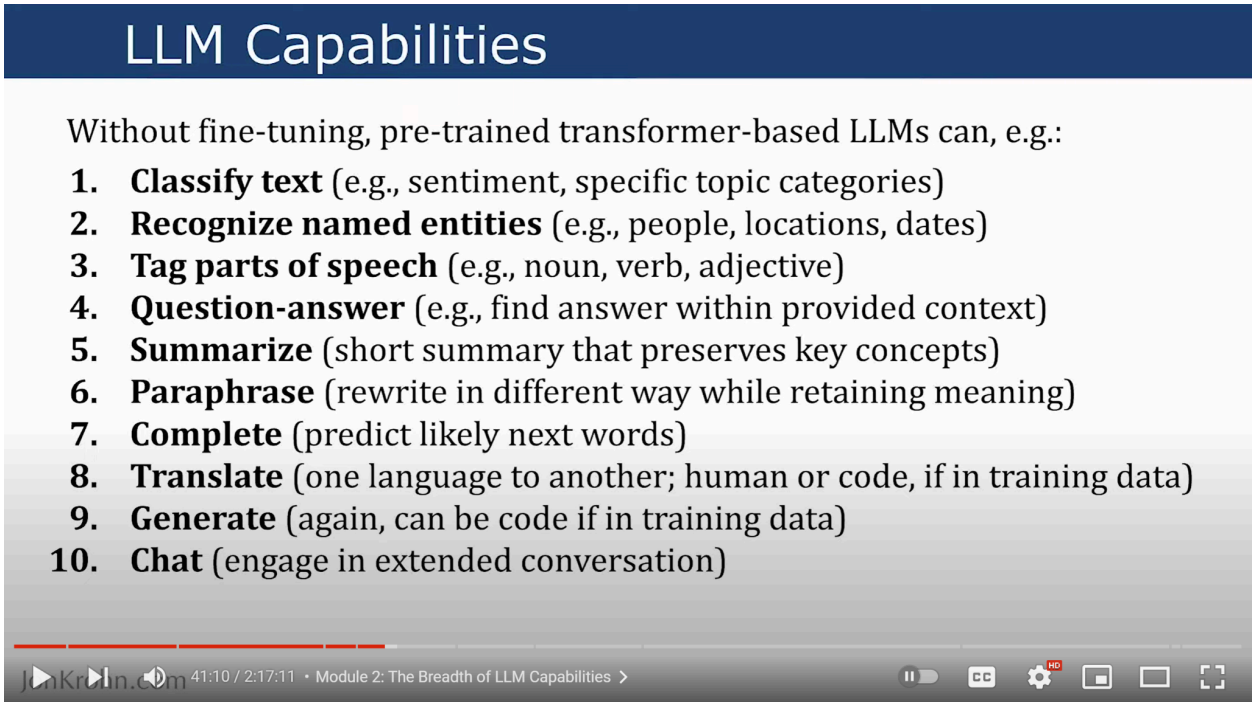
https://www.youtube.com/watch?v=Ku9PM26Cc2c





Generative AI with Large Language Models: Hands-On Training feat. Hugging Face and PyTorch Lightning

Jon Krohn
14K subscribers
Subscribed
253
Share
Download
Clip


LLM Capabilities
Without fine-tuning, pre-trained transformer-based LLMs can, e.g.:
1. Classify text (e.g., sentiment, specific topic categories)
2. Recognize named entities (e.g., people, locations, dates)
3. Tag parts of speech (e.g., noun, verb, adjective)
4. Question-answer (e.g., find answer within provided context)
5. Summarize (short summary that preserves key concepts)
6. Paraphrase (rewrite in different way while retaining meaning)
7. Complete (predict likely next words)
8. Translate (one language to another; human or code, if in training data)
9. Generate (again, can be code if in training data)
10. Chat (engage in extended conversation)
41:10 / 2:17:11 • Module 2: The Breadth of LLM Capabilities

<END>