Module 1

Monday, June 17, 2024 9:16 AM

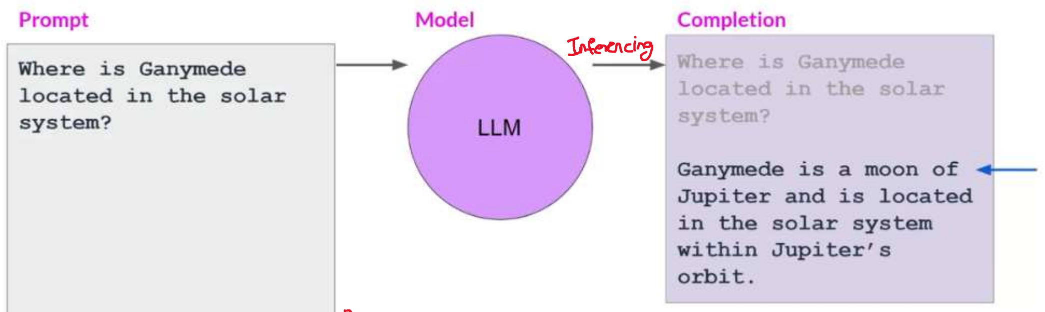
Generative AI project lifecycle

Foundation, training & tuning them

Examples of Gen AI

- Text generation
- Image Generation
- Coding assistant

We are able to communicate with LLMs with natural language, instead of machin code.



Where is Ganymede located in the solar system?

Where is Ganymede located in the solar system?

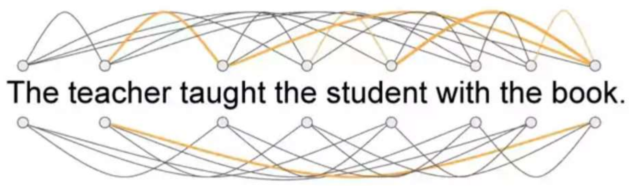
Ganymede is a moon of Jupiter and is located in the solar system within Jupiter's orbit.

Context window
- typically a few 1000 words.

Use Cases:

→ Generating assay
→ Translation → NL to machine code
language to Language

→ Information retrieval with name-entity recognition
→ Augmenting with external datasources



The teacher taught the student with the book.

Because of the architecture, it has the potential

→ Information

→ Augmenting with external datasources

Before Transformers

we had RNN, which had limited visibility of previous context. To generate new text, we need to understand the semantics

Transformers therefore are used because they scale efficiently, parallel process and attention to input meaning

Encoder only architecture can be used for classification task

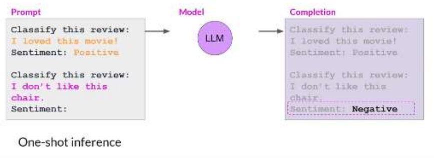
Encoder Decoder architecture are usually used for sequence-to-sequence task

Decoders are popular and used for generalised tasks

Great way to improve performance of prompt engineering is with giving examples of chat responses.

In-context learning . ① Zero-shot inference.
(Big Models perform very well)

② One-shot inference → We give example of how the model should perform


Prompt

```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment:
```

Model

Completion

```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment: Negative
```

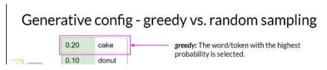
One-shot inference

③ Few Shot inference : Give multiple examples.

Generative Configuration:

Max new tokens : Limit the number of tokens it will generate

Greedy v/s random sampling :

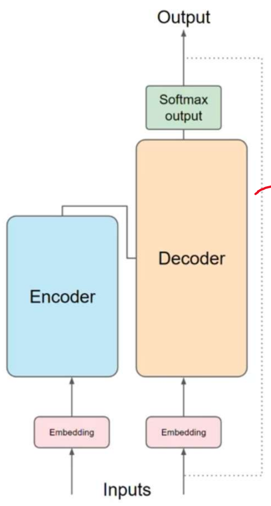

Generative config - greedy vs. random sampling

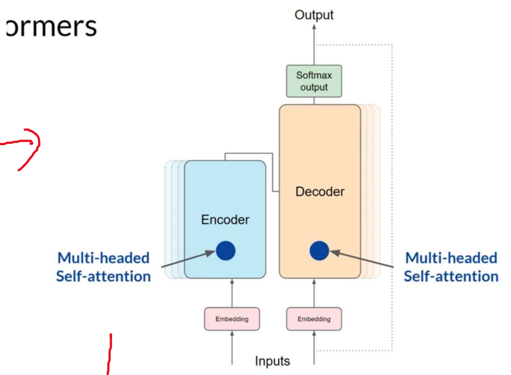
0.20	cake
0.10	donut

greedy: The word/token with the highest probability is selected.

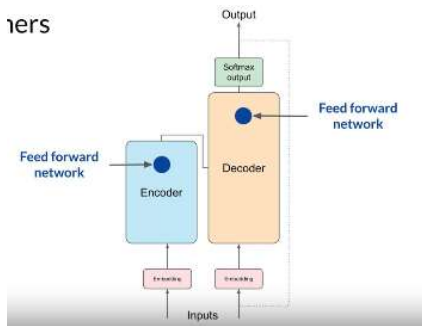
Because of the architecture, it has the potential find meaning and relation between every word.

These attention weights are learned during training.



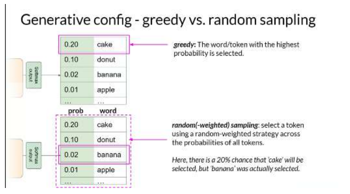


Initial intention of transformer architecture was for sequence-to-sequence task

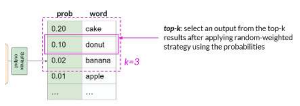


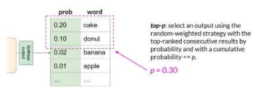
Max new tokens : [illegible]
it will generate

Greedy v/s random sampling :
Top-P & Top-k are different methods to provide random sampling


Generative config - greedy vs. random sampling
Softmax output
prob
word
0.20
cake
0.10
donut
0.02
banana
0.01
apple
greedy: The word/token with the highest probability is selected.
random(-weighted) sampling: select a token using a random-weighted strategy across the probabilities of all tokens.
Here, there is a 20% chance that 'cake' will be selected, but 'banana' was actually selected.

-This increases the chance that the generated text is sensible


Softmax output
prob
word
0.20
cake
0.10
donut
0.02
banana
0.01
apple
k=3
top-k: select an output from the top-k results after applying random-weighted strategy using the probabilities

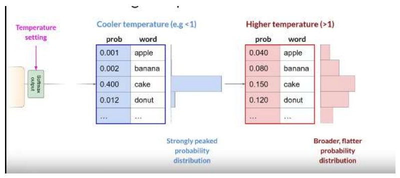

Softmax output
prob
word
0.20
cake
0.10
donut
0.02
banana
0.01
apple
top-p: select an output using the random-weighted strategy with the top-ranked consecutive results by probability and with a cumulative probability <= p.
p = 0.30

A higher topK value allows for more diversity by considering a larger set of tokens, while a lower value like topK=1 (greedy decoding) restricts sampling to just the single most likely token, favoring coherence over diversity

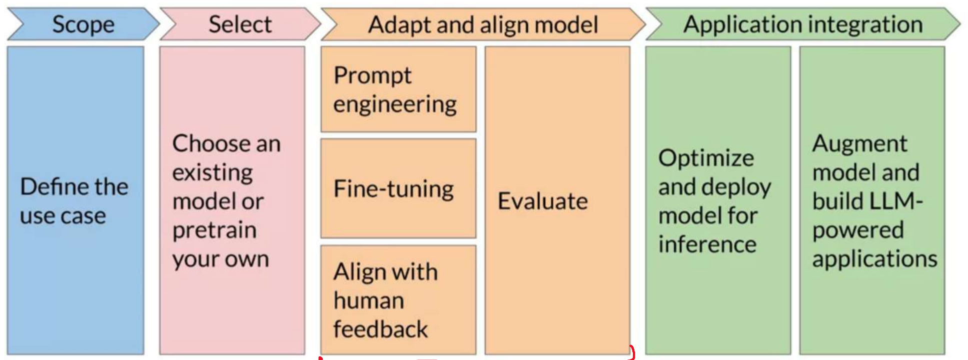
<https://www.perplexity.ai/search/help-me-understand-IsLx8PHnRIC6ng2YJjiA9A>

A higher topP value (e.g. 0.9) allows for more diversity by considering a wider range of tokens, while a lower value (e.g. 0.5) restricts sampling to a narrower set of high probability tokens, favoring coherence over diversity.

From <https://www.perplexity.ai/search/help-me-understand-IsLx8PHnRIC6ng2YJjiA9A>



Generative AI Lifecycle:


Scope
Select
Adapt and align model
Application integration
Define the use case
Choose an existing model or pretrain your own
Prompt engineering
Fine-tuning
Align with human feedback
Evaluate
Optimize and deploy model for inference
Augment model and build LLM-powered applications

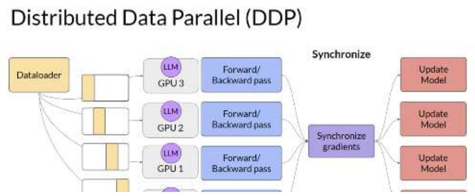
We need to define exactly what our use case is, this

Highly iterative.

[illegible] of LLM

Efficient Multi-GPU compute strategy:

Pytorchs Distributed Data Parallel (DDP)
->



Highly iterative.

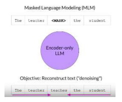
We need to define exactly what our use case is, this helps save cost & define our goal

Encoder Models: Also known as autoencoding models.

- They are trained using Masked Language Modeling (MLM)

- They are good for
  - -> Sentiment analysis
  - -> Named entity recognition
  - -> Word classification

Ex: BERT, ROBERTA


Masked Language Modeling (MLM)
The teacher <MASK> the student
Encoder-only LLM
Objective: Reconstruct text ("denoising")
The teacher teaches the student

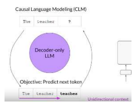
Decoder Model: Autoregressive Model

-> Pretrained using causal language Modeling

- Text generation

Large decoder models show great performance for zeroshot learning

Ex: GPT, Bloom



Encoder-Decoder.

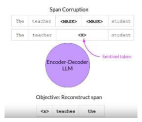
Good use cases

-> Traslation

-> Text summarization

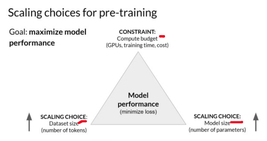
-> Question answering

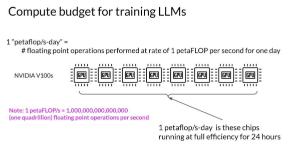
Ex: T5, BART

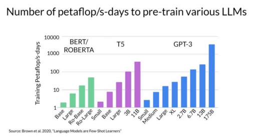


Scaling choices for pre-training.

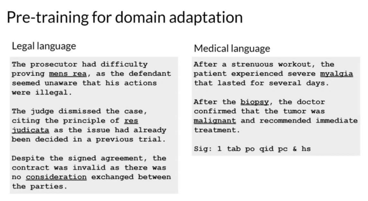
Goal: To maximize the model performance


Scaling choices for pre-training
Goal: maximize model performance
CONSTRAINT: Compute budget (GPUs, training time, cost)
Model performance (minimize loss)
SCALING CHOICE: Dataset size (number of tokens)
SCALING CHOICE: Model size (number of parameters)


Compute budget for training LLMs
1 "petaflop/s-day" =
# floating point operations performed at rate of 1 petaFLOP per second for one day
NVIDIA V100s
GPU
Note: 1 petaFLOP/s = 1,000,000,000,000,000 (one quadrillion) floating point operations per second
1 petaflop/s-day is these chips running at full efficiency for 24 hours



Pretraining for Domain adaptation.



Pre-training for domain adaptation

Legal language

```
The prosecutor had difficulty
proving mens rea, as the defendant
seemed unaware that his actions
were illegal.

The judge dismissed the case,
citing the principle of res
judicata as the issue had already
been decided in a previous trial.

Despite the signed agreement, the
contract was invalid as there was
no consideration exchanged between
the parties.
```

Medical language

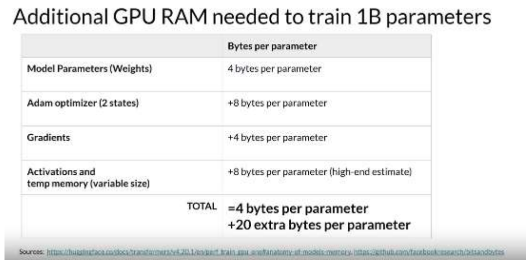
```
After a strenuous workout, the
patient experienced severe myalgia
that lasted for several days.

After the biopsy, the doctor
confirmed that the tumor was
malignant and recommended immediate
treatment.

Sig: 1 tab po qid pc & hs
```

why domain specific model?

-> Because language in that domain is different. Example

When scaling data, we need to increase both data and number of parameters of model.

Computation challenges of LLM



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

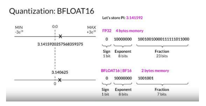
Sources: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory, https://github.com/facebookresearch/bitsandbytes

How to scale down?

-> Quantization

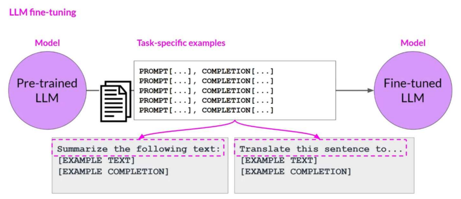
BFloat is great, its a hybrid setup of FP32 & FP16.

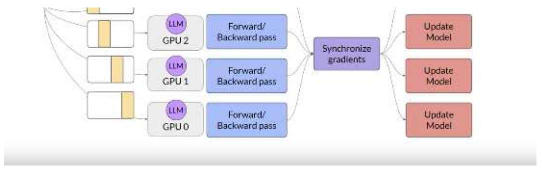
-> Significantly helps with training stability



Fine-Tuning is supervised method to change LLM behavior

Using prompts to fine-tune LLMs with instruction

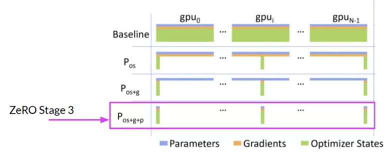

LLM fine-tuning
Model
Pre-trained LLM
Task-specific examples
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
PROMPT[...], COMPLETION[...]
Model
Fine-tuned LLM
Summarize the following text:
[EXAMPLE TEXT]
[EXAMPLE COMPLETION]
Translate this sentence to...
[EXAMPLE TEXT]
[EXAMPLE COMPLETION]



FSDP (Fully Sharded Data Parallel)

Zero Redundancy Optimizer (ZeRO)

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

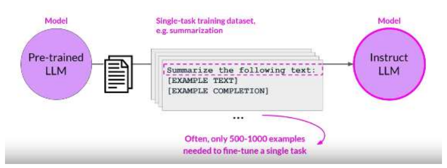


Sources:
Rajbhandari et al. 2019: "ZeRO: Memory Optimizations Toward Training Trillion Parameter Models"
Zhao et al. 2023: "PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel"

-> Helps reduce overall GPU memory utilization

-> support offloading to CPU if needed.

Fine-tuning on a single task


Model
Pre-trained LLM
Single-task training dataset, e.g. summarization
Model
Instruct LLM
Often, only 500-1000 examples needed to fine-tune a single task

```
Summarize the following text:
[EXAMPLE TEXT]
[EXAMPLE COMPLETION]
...
```

- Single-task training dataset
- Often only 500-1000 examples are needed to fine-tune a single task.

Problem: Catastrophic forgetting

Solution: Fine-tune on multiple tasks
(or) Parameter Efficient Fine-tuning (PEFT)

Multi-task, instruction