2025

Lesson 06

Finetuning

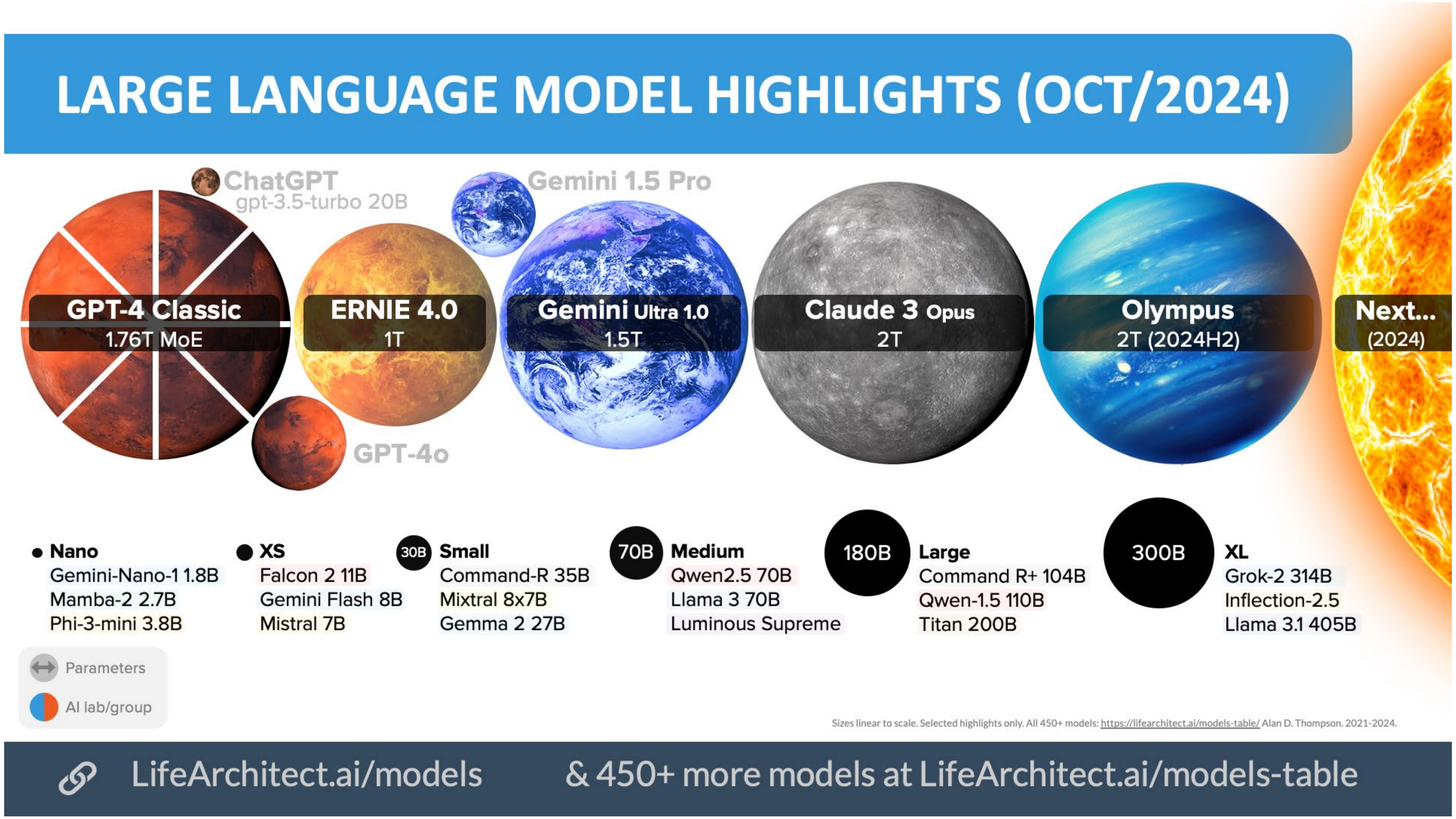
RP

An Introduction to LLM

Introduction

- Large models with billions of parameters require significant resources but offer better generalization.
- High computational and memory demands, long training times and data quality concerns make training challenging.
- Optimizes model efficiency by reducing weight precision, useful for resource-limited inference scenarios.
- Parameter-Efficient Fine-Tuning techniques like LoRA or QLoRA reduce computational overhead by updating only a small subset of parameters.
- Depends on model size and optimizations; larger models need multiple GPUs, while optimized/quantized models can run on fewer GPUs.

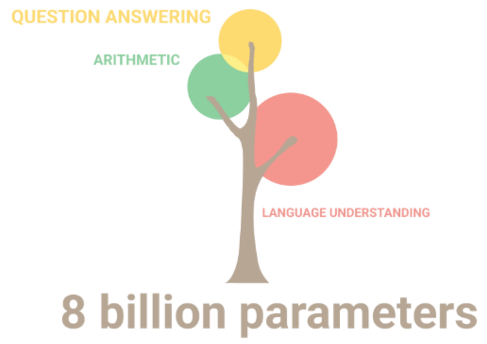
Language Model Sizes



Source: https://lifearchitect.ai/models/

LLM Model Size Matters?

- Age-old question – does size matter?
- In the world of LLM (Large Language Model), the consensus has been that it does.
- Currently, excitement is experienced around the development of bigger and bigger model.
- Increasing the model size has been a major driver of performance.



Source: https://blog.research.google/2022/04/pathways-language-model-palm-scaling-to.html

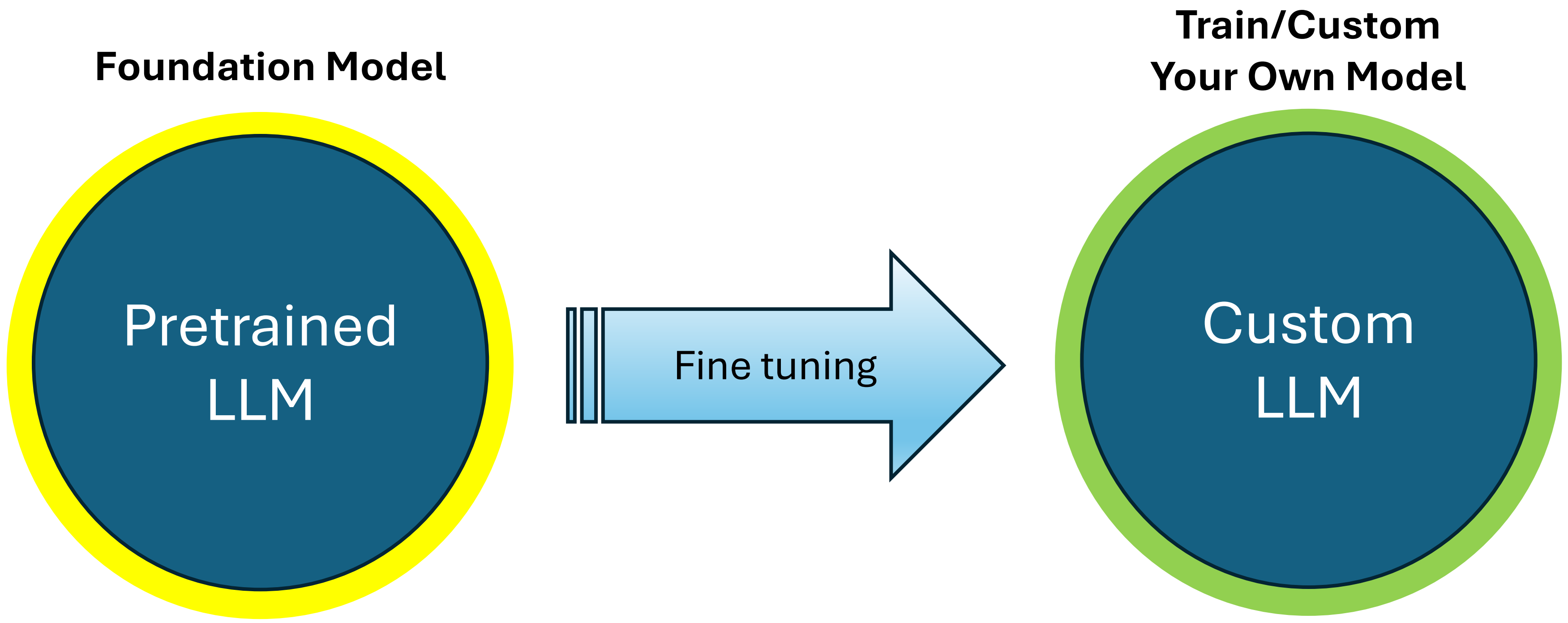
Small vs Big Models

- “Scaling Laws for Neural Language Models” (Kaplan, McCandish, 2020) demonstrated that model performance is driven by just three factors i.e. number of parameters, size of dataset and amount of compute resource.
- Scaling laws have been the driving force behind the “bigger is better” belief in LLMs.
- Another paper by OpenAI “Training language models to follow instructions with human feedback” reveals that fine-tuning language models with human feedback can drastically improve the performance without necessarily increasing the model’s size.

Training Large Language Model (LLM)

- Many types of training mechanism to suit different means, requirements and goals.
- Each type of training mechanism serves different purposes.
- Some of the important training mechanisms are:
  - Pre-training
  - Fine-tuning
  - Reinforcement Learning from Human Feedback (RLHF)
  - Adapters
- Prompting
- Quantization (not actually training)

Choosing a Model to Train



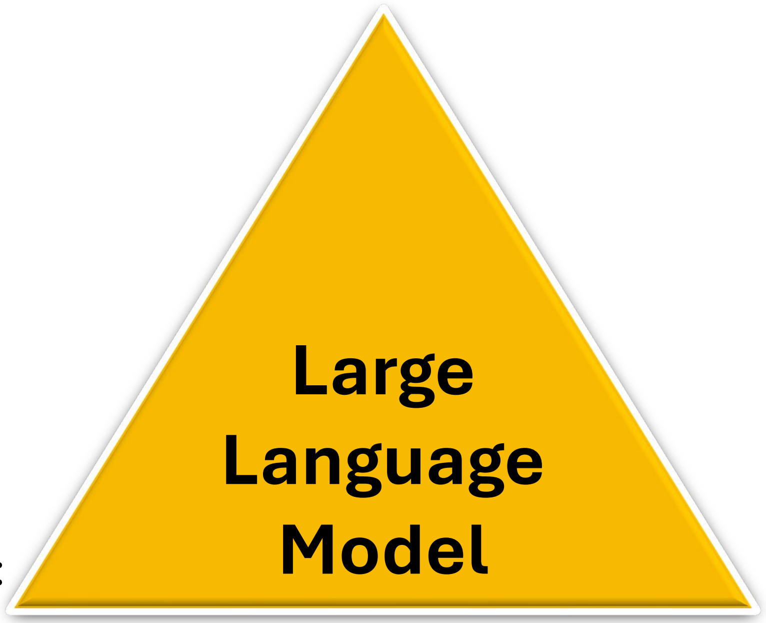
Memory Requirements

GPU Memory Requirements Estimation

- Able to accurately estimate the GPU (Graphical Processing Unit) memory requirement is essential for several reasons.
  - Cost efficiency: GPUs are expensive resources. Overestimating memory needs leads to unnecessary expenditure on hardware and underestimating can cause system failure and degraded performance.
  - Performance Optimization: Optimal memory management allows the models to run efficiently and able to handle more concurrent requests.
  - Scalability: Understanding memory requirements becomes vital for scaling the service without compromising on performance or incurring costs.
- Calculating the GPU memory needed to serve LLMs is not straightforward.
- Model size, sequence length, batch sizes and decoding algorithms affect memory consumption. Advanced optimization techniques such as Paged Attention can reduce memory consumption significantly.

Training Dilemma Triangle

LIMITATION:
GPUs, Training
Time, Budget



DATASET SIZE:
Number of
tokens

MODEL SIZE:
Number of
parameters

Training Challenges

Approximate GPU RAM needed to store 1B parameters

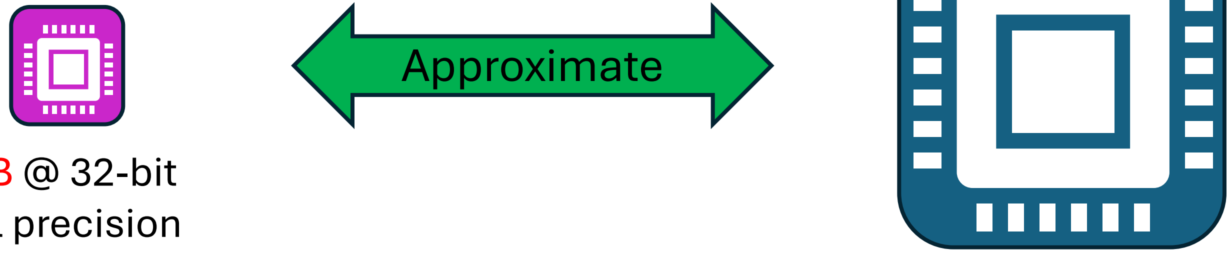
1B = 1 billion parameters, GPU (Graphical Processing Unit)

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

Additional Memory

	Bytes per parameter
Model Parameter (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 (an estimate)
Total	= 4 bytes per parameter +20 extra bytes per parameter

~20 extra bytes per parameter



Memory needed to **store** model

4GB @ 32-bit Full precision

Memory needed to **train** model

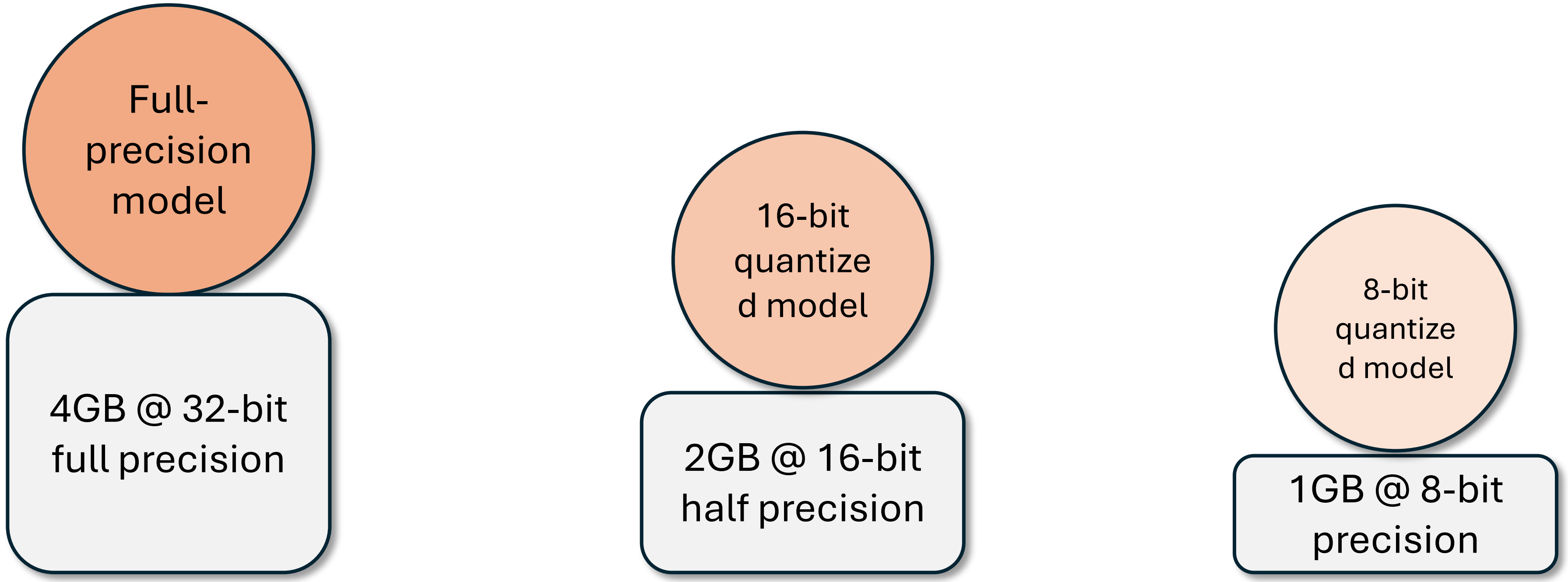
80GB @ 32-bit Full precision

To account for all possible overheads during training, may actually require approximately 20 times the amount of GPU RAM

Source: https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one#anatomy-of-models-memory

GPU RAM

- Approximate GPU RAM needed to store 1 billion parameters is 4GB @ 32 bit full precision.
- The purpose of quantization is to reduce memory to store and train the model.



GPU Memory Needed to Serve a LLM

- How much GPU memory is needed to serve a Large Language Model (LLM)?
  - No definitive answer
  - Below formula serves as a guide

$$M = \left(\frac{P \times 4B}{\frac{32}{Q}}\right) \times 1.2$$

M	**GPU memory expressed in Gigabyte**
P	The number of parameters in the model
4B	4 bytes, expressing the bytes used for each parameters
Q	The number of bits that should be used for loading the model – 16, 8 or 4 bits
1.2	Represents a 20% overhead of loading additional things in GPU memory

Example:
Llama 70 billion parameter model, loaded in 16 bits format.

$$M = \left(\frac{70 \times 4}{\frac{32}{16}}\right) \times 1.2 = 168GB$$

- A single NVIDIA A100 GPU with 80GB of memory would not be sufficient to serve this model.
- Need at least two A100 GPUs with 80GB each to handle the memory load efficiently.

Pretraining / Finetuning

Pretraining

1

Most fundamentals way of training

2

Start with an untrained model

3

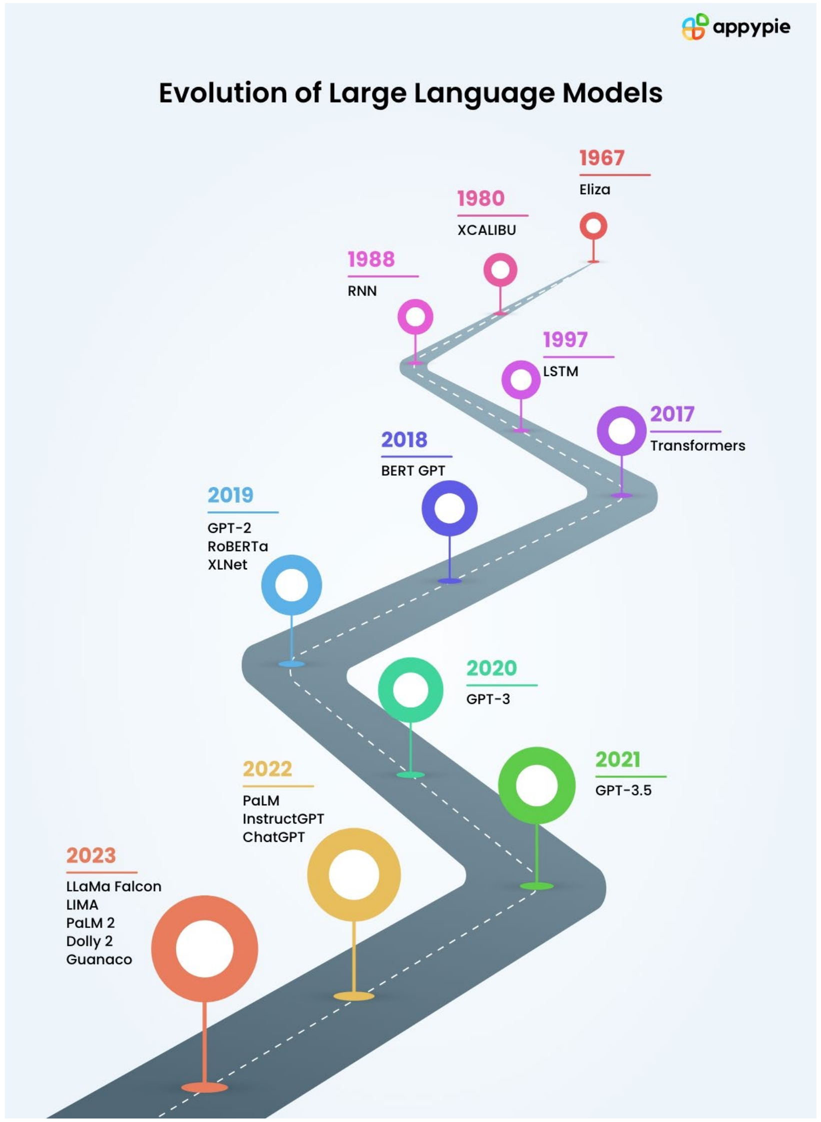
Train to predict the next tokens (words) given a sequence of previous tokens (words)

4

Train with a large corpus of sentences collected from various sources

5

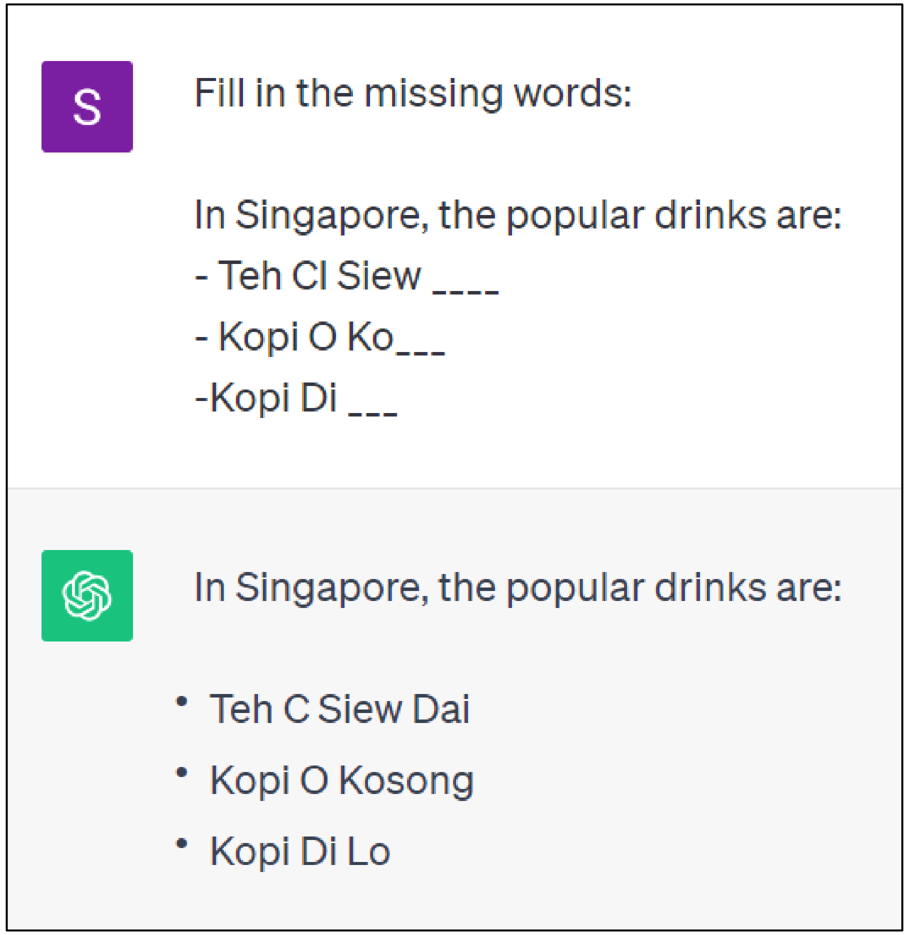
Training mode: self-supervised



Source: https://www.appypie.com/blog/evolution-of-language-models

Pretraining

- The model learns to encode the structure of the language.
- Starts to learn the “knowledge” that is hidden and yet included in the texts.
- Pretraining has its limitations:
  - Costly to train: Takes weeks/months/years to train.
  - According to OpenAI, training ChatGPT-3 required USD$3.2 million in computing resources alone.
  - Environmental unfriendly.
    - Electricity, water, estate space etc.



Fill in the missing words:

In Singapore, the popular drinks are:
- Teh CI Siew ____
- Kopi O Ko___
-Kopi Di ___

In Singapore, the popular drinks are:

- Teh C Siew Dai
- Kopi O Kosong
- Kopi Di Lo

What if you want the LLM to understand Singlish?

Source: https://www.cnbc.com/2023/03/13/chatgpt-and-generative-ai-are-booming-but-at-a-very-expensive-price.html

Fine Tuning

Question: Pretraining has already been done by others, why would you want to train a similar model from scratch?

- Training can become necessary if you work with data that has properties similar to language but is not common language itself.
- LLM that is trained on natural language can’t handle, for example, Python code generation properly.
- However, due to many similarities between Python code generation and natural language, the pretrained model can be “re-used” for further finetuning.

Fine Tuning

- Pretrained LLM, has 2 main shortcomings:
  - Output structure generation
    - Chat vs FAQ vs Translation vs Summarization vs Sentiment …
  - Specialisation
    - Absence of knowledge that wasn't encoded in the data in the first place.
- Pretrained model was trained to predict the next token given a sequence of token before.
- Two ways to perform finetuning:
  - Create prompt such that the model's ability to predict the next token solves your tasks (prompt engineering).
  - Replace some layer(s) of the model to reflect the task and generate the output you want to achieve.

Finetuning

- Pretrained model encode a large variety of common knowledge.
- However, there are knowledge domains that the pretrained model does not know about.
- If you need to work with such domain, finetuning needs to be performed.
- Finetuning take an already pretrained model and trains it with different (smaller) dataset.
- Requires a fraction of the computing resources and hence performed faster.

Reasons to Training Your Own LLM

- Data Privacy
  - Data stay on-premises. Prevent leakage and regulatory breaches (e.g. PDPA, GDPR).
- Reliability
  - Reduce hallucinations, enhance consistency, mitigate bias and filter unwanted information.
- Flexibility
  - Adapt LLM stack to custom needs.
- Better Performance
  - Consistent outputs, stop suggesting your competitors.
- Latency
  - Lower latency, higher output and control over latency.

Reasons to Train Your Own LLM

6. Content Control
   - Stop inappropriate content
   - Stop un-licensed content
   - Your own data
7. Bias
   - Control biases with your own data
   - Unlearn biases with unbiased data
8. Ownership
   - No recurring professional services
   - Build AI moat & in-house know-how

Train Your Own LLM

Prompt Format & Finetuning

- A structured prompt format is needed during finetuning where precise control over the model's behaviour is important.
- Fine-tuning helps models learn how to respond to specific instructions or behave in a particular way, which is often done using well-structured prompts.
- Structured prompt format is typically important:
  - Finetuning: if you want the model to generate specific types of responses (e.g., formal vs. informal tone, answering questions in a certain way), you need to provide examples that define the format you expect.

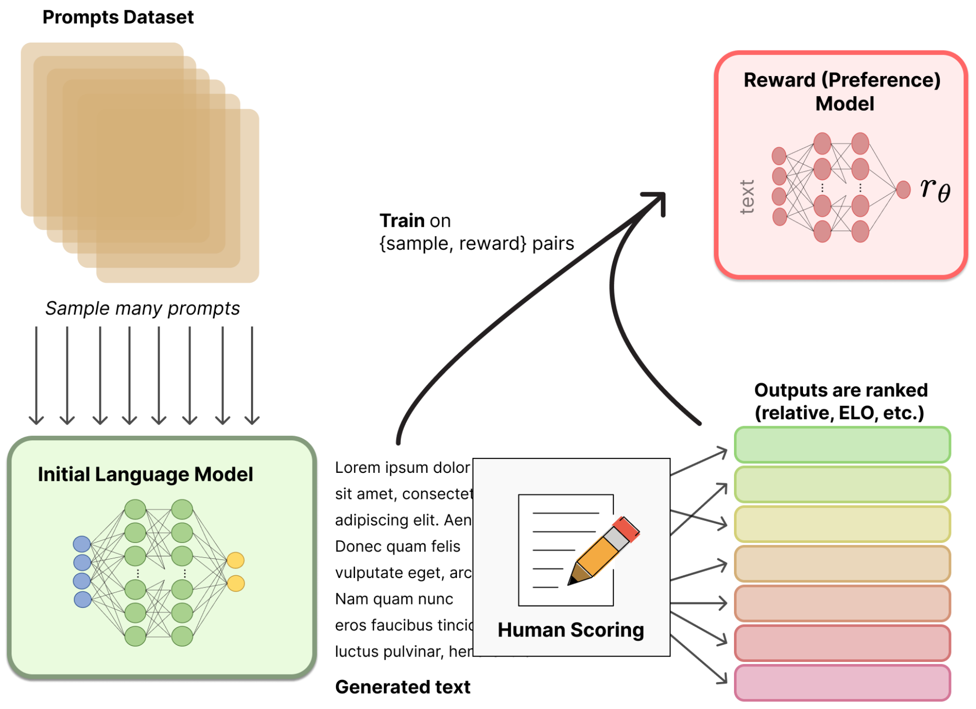
Prompt Format & Finetuning

- Structured prompt format is typically important:
  - Custom Model Behaviour: If you need to alter the way the model responds - such as specifying a style, tone, or type of knowledge (e.g., asking it to respond as a technical expert or assistant) - the use of a well-defined prompt format can guide the model in generating appropriate outputs.
  - Multi-turn Conversation: When you're working on tasks involving multiple turns, maintaining consistent formatting ensures that the model correctly interprets the context and role of each participant (user vs. assistant) in the conversation.

References:
https://www.llama.com/docs/model-cards-and-prompt-formats/
https://llama.meta.com/docs/model-cards-and-prompt-formats/llama3_1

Reinforcement Learning from Human Feedback (RELF)

- A special case of finetuning.
- Outputs generated by the model a human will find most useful.
- Given an arbitrary prompt, multiple outputs are generated.
- A human ranks those outputs according to how useful it is to them.
- The rankings provided by human will be used to train a reward model.
- The reward model, ultimately is used to train the LLM model.


Prompts Dataset
Sample many prompts
Initial Language Model
Lorem ipsum dolor sit amet, consectet adipiscing elit. Aen Donec quam felis vulputate eget, arc Nam quam nunc eros faucibus tinci luctus pulvinar, her
Generated text
Human Scoring
Outputs are ranked (relative, ELO, etc.)
Train on {sample, reward} pairs
Reward (Preference) Model
text
r_θ

Source: https://huggingface.co/blog/rlhf

RLHF

- For this training, some effort is required.
- Require human-labelled data.
- However, the effort is not significant when compared to pretraining.
- Use the reward model to generalize from data that it can rate the LLM on its own.
- RHLF is often used to make output more conversation-like or avoid undesired behaviour such as mean, intrusive or insulting.

Adapters

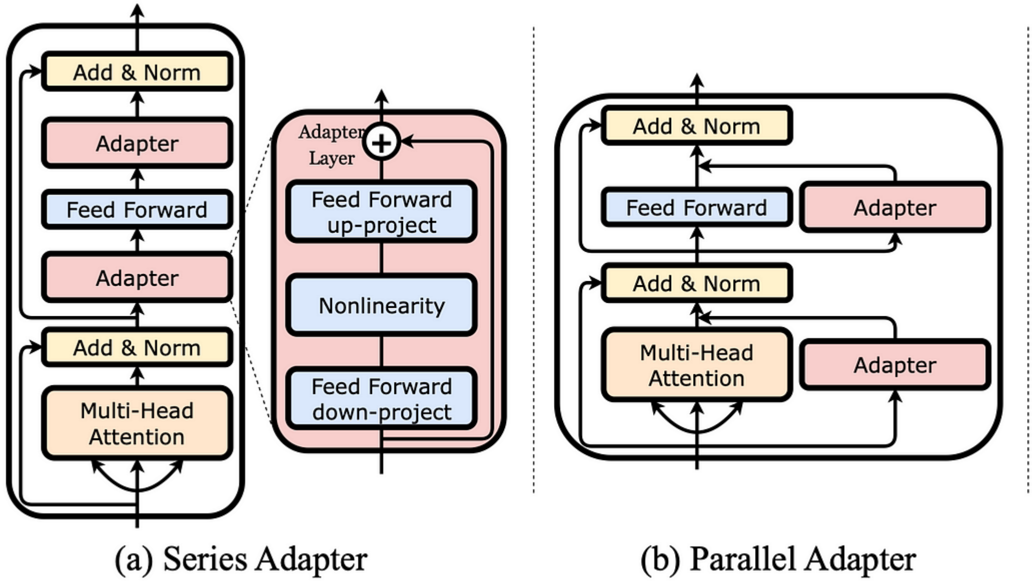
An alternative method to train using a smaller number of parameters.

Additional layers are added to an already trained model.

During finetuning, these adapters are trained while the rest is not changed.

Adapter is much smaller and easier to tune.

Can be inserted at different position in the model.



https://arxiv.org/pdf/2304.01933.pdf

PEFT – Parameter Efficient Fine Tuning
LoRA – Low Rank Adaption

Prompting

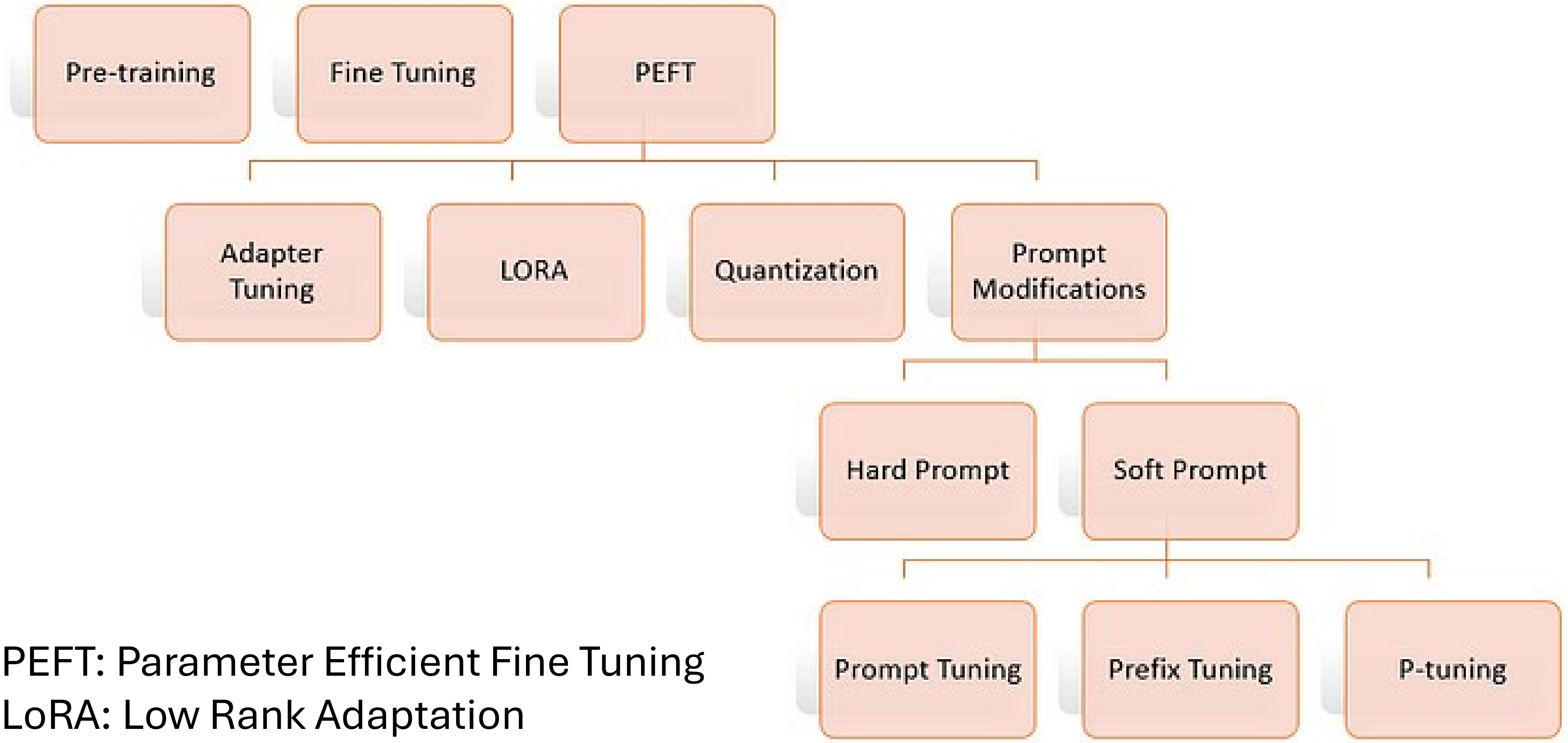
- Prompting in the context of LLM.
- Providing input or instructions to the model to generate specific responses or information.
- Prompting is not the same as training because in prompting, no weights of the models are updated.
- Hard prompts are manually hand-crafted text prompts with discrete input tokens.
- Soft prompt is created during the process of prompt tuning.
- Prompt tuning can be seen as a kind of training.

Quantization

- Reduce the precision of the parameters of a pre-trained model.
- This reduction will then be able to accommodate the model in a low memory hardware.

Advantages	Disadvantages
Reduced Memory Footprint Reduction in precision shrink memory footprint	Loss of Precision Lower bit precision can cause potential accuracy degradation
Faster Inference Using lower precision representation leads to faster inference	Diminished Generalisation More challenging for a model to generalize across a wide range of data
Energy Efficiency Suitable for deployment on resource constrainted devices	Fine Tuning Challenges Quantised models can be less amendable to fine-tuning
Cost Savings Lower computation demands	Trade-off between Compression & Performance Balancing between compression & performance can be challenging

Category of Fine Tuning



PEFT: Parameter Efficient Fine Tuning
LoRA: Low Rank Adaptation

Source: https://medium.com/@siddharth.vij10/llm-gpt-fine-tuning-peft-lora-quantization-92b9ef357986

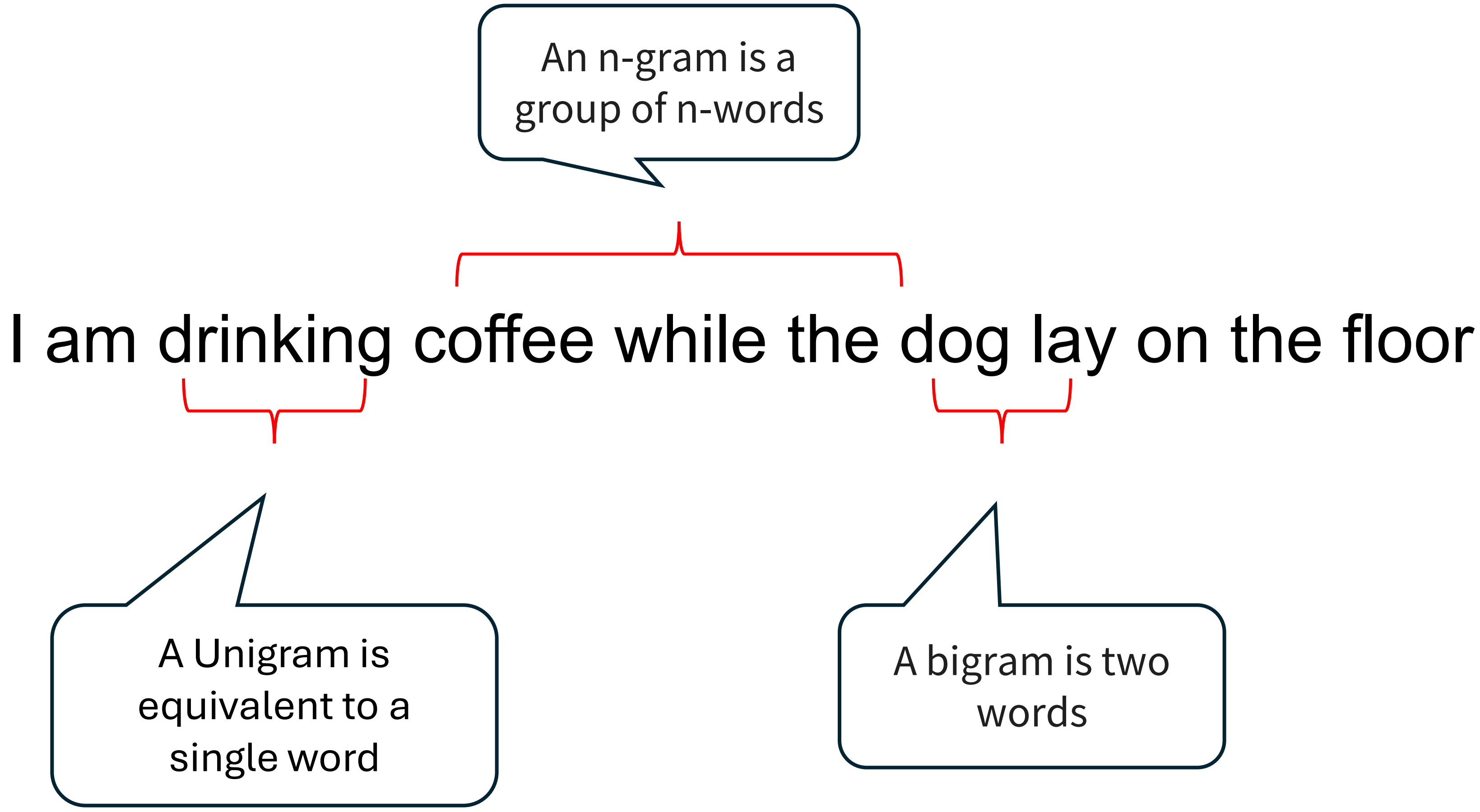
NLP/LLM Evaluation

- It is essential to evaluate Large Language Model (LLM) / NLP application to determine their quality and usefulness.
- There are several frameworks to perform evaluation, but none of them are comprehensive enough to cover all aspects of language understanding.
- Recognizing the diversity of applications that modern large language models (LLMs) serve, it becomes evident that a one-size-fits-all approach to LLM performance evaluation is impractical.

Evaluation Metrics

- ROGUE and BLEU SCORE are two widely used evaluation metrics.
- ROGUE (Recall-Oriented Understudy for Gisting Evaluation)
  - Used for text summarization.
  - Compares a summary to one or more summaries.
- BLEU (Bilingual Evaluation Understudy) Score
  - Used for text translation.
  - Compare to human-generated translations.

Terminology


An n-gram is a group of n-words
I am drinking coffee while the dog lay on the floor
A Unigram is equivalent to a single word
A bigram is two words

ROUGE-1

Reference (human):

It is cold outside.

Generated output:

It is very cold outside.

$$\text{ROUGE-1 (Recall)} = \frac{unigram\ matches}{unigrams\ in\ reference} = \frac{4}{4} = 1$$

$$\text{ROUGE-1 (Precision)} = \frac{unigram\ matches}{unigrams\ in\ output} = \frac{4}{5} = 0.8$$

$$\text{ROUGE-1 (F1)} = 2 \times \frac{precision \times recall}{precision + recall} = 2 \times \frac{0.8}{1.8} = 0.89$$

ROUG-2

Reference (human):

It is is cold cold outside

Generated output:

It is is very very cold cold outside

$$\text{ROUGE-2 (Recall)} = \frac{bigram\ matches}{bigrams\ in\ reference} = \frac{2}{3} = 0.67$$

$$\text{ROUGE-2 (Precision)} = \frac{bigram\ matches}{bigrams\ in\ output} = \frac{2}{4} = 0.5$$

$$\text{ROUGE-2 (F1)} = 2 \times \frac{precision \times recall}{precision + recall} = 2 \times \frac{0.335}{1.17} = 0.57$$

RP

Library: rouge_score

```
# install rouge_score version 0.1.2
pip install rouge-score

# Python Code
from rouge_score import rouge_scorer
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)

candidate = "it is very cold outside."
reference = "it is cold outside."
scores = scorer.score(reference, candidate)
for key in scores:
   print(f'{key}: {scores[key]}')
```

```
rouge1: Score(precision=0.8, recall=1.0, fmeasure=0.888888888888889)
rouge2: Score(precision=0.5, recall=0.6666666666666666, fmeasure=0.5714285714285715)
rougeL: Score(precision=0.8, recall=1.0, fmeasure=0.888888888888889)
```

BLEU

- BLEU metric = Average (precision across range of n-gram sizes)

- Reference (human):
  - I am very happy to say that I am drinking a warm cup of tea.

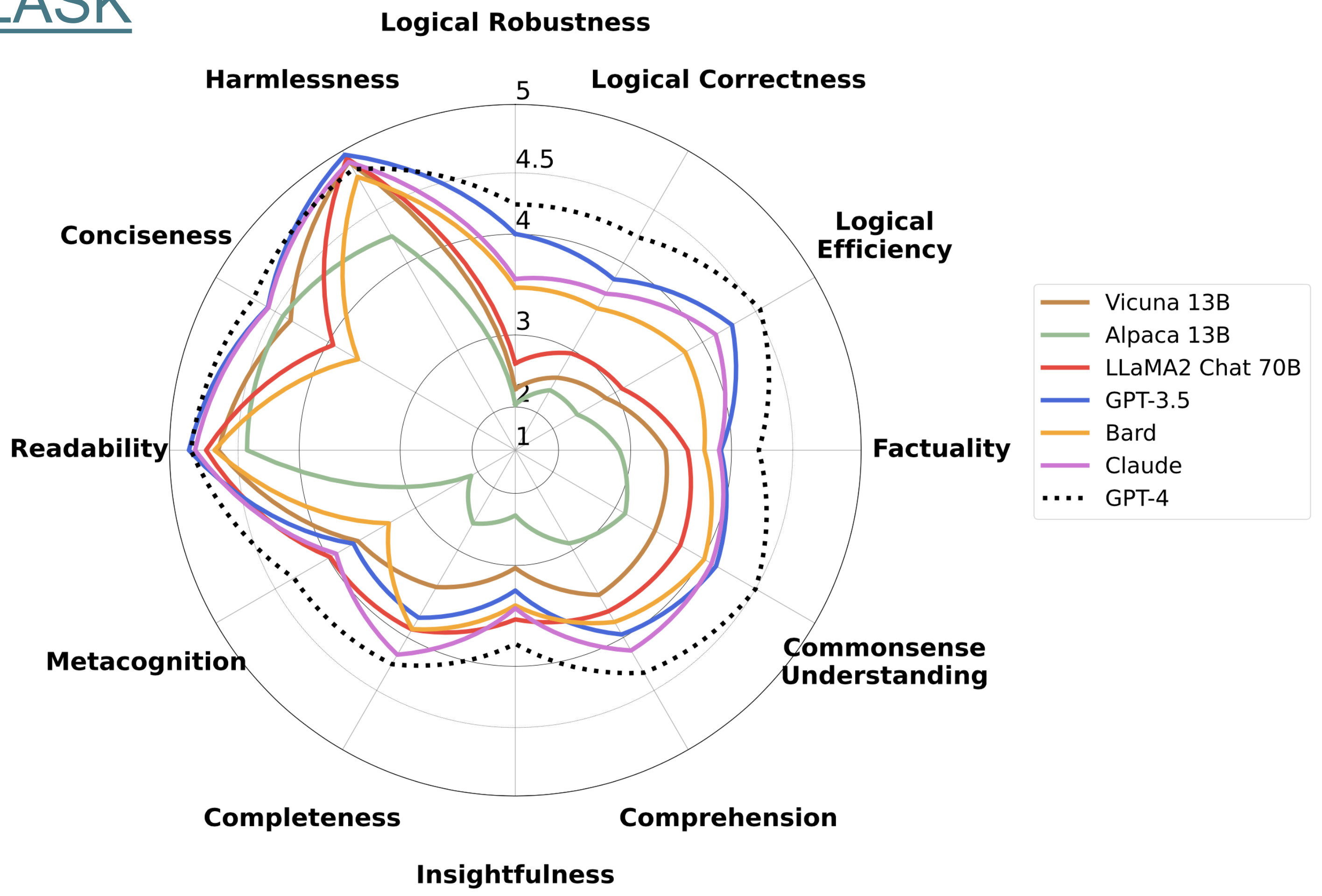
- Generated output:
  - I am very happy that I am drinking a cup of tea. ➔ BLEU 0.495
  - I am very happy that I am drinking a warm cup of tea. ➔ BLEU 0.730
  - I am very happy to say that I am drinking a warm tea. ➔ BLEU 0.798

Library: NLTK (Natural Language Toolkit)

```
# install nltk version 3.8.1
pip install nltk

# Python Code
from nltk.translate.bleu_score import sentence_bleu

can1 = "I am very happy that I am drinking a cup of tea".split()
can2 = "I am very happy that I am drinking a warm cup of tea".split()
can3 = "I am very happy to say that I am drinking a warm tea".split()

ref = "I am very happy to say that I am drinking a warm cup of tea".split()

print(sentence_bleu([ref], can1))
# 0.4953319416460018

print(sentence_bleu([ref], can2))
# 0.729836014355472

print(sentence_bleu([ref], can3))
#0.7979042533838905
```

Flask (Fine-grained Language Model Evaluation Based on Alignment Skills Sets)

- https://arxiv.org/pdf/2307.10928.pdf
- https://github.com/kaistAI/FLASK



Major LLM Evaluation Frameworks

Framework Name	Factors Considered for Evaluation	Link
Big Bench	Generalization abilities	https://github.com/google/BIG-bench
GLUE Benchmark	Grammar, Paraphrasing, Text Similarity, Inference, Textual Entailment, Resolving Pronoun References	https://gluebenchmark.com/
SuperGLUE Benchmark	Natural Language Understanding, Reasoning, Understanding complex sentences beyond training data, Coherent and Well-Formed Natural Language Generation, Dialogue with Human Beings, Common Sense Reasoning (Everyday Scenarios and Social Norms and Conventions), Information Retrieval, Reading Comprehension	https://super.gluebenchmark.com/
OpenAI Evals	Accuracy, Diversity, Consistency, Robustness, Transferability, Efficiency, Fairness of text generated	https://github.com/openai/evals
SQUAD	Reading comprehension tasks	https://rajpurkar.github.io/SQuAD-explorer/

Source and not exhaustive list

RP

Selecting a Large Language Model

Task Type	Benchmarks
Reasoning	HellaSwag, WinoGrande, PIQA
Reading comprehension/ question answering	BoolQ, TriviaQA, NaturalQuestions
Math word problems	Math, GSM8K, svamp, mathqa, algebra222
Coding	HumanEval, MBPP
Multi-task	MMLU, BBH, GLUE
Separating fact from fiction in training data	TruthfulQA
Multi-turn	MTBench, QuAC
Multilingual	XCOPA, TyDiQA-GoldP
Long context	SCROLLS

Source: The Fast Path to Developing with LLMs (Nvidia, 29 Nov 2023)

Selection factors:

- Benchmark scores
- Quality and quantity of training data
- Human evaluation/validation
- Inference latency
- Cost of deployment, use or price per tokens
- Context Size
- License Terms

Models tuned to specific domains can sometimes perform as well as models that are orders of magnitude larger but have only been pretrained.

LLM Abilities & Datasets

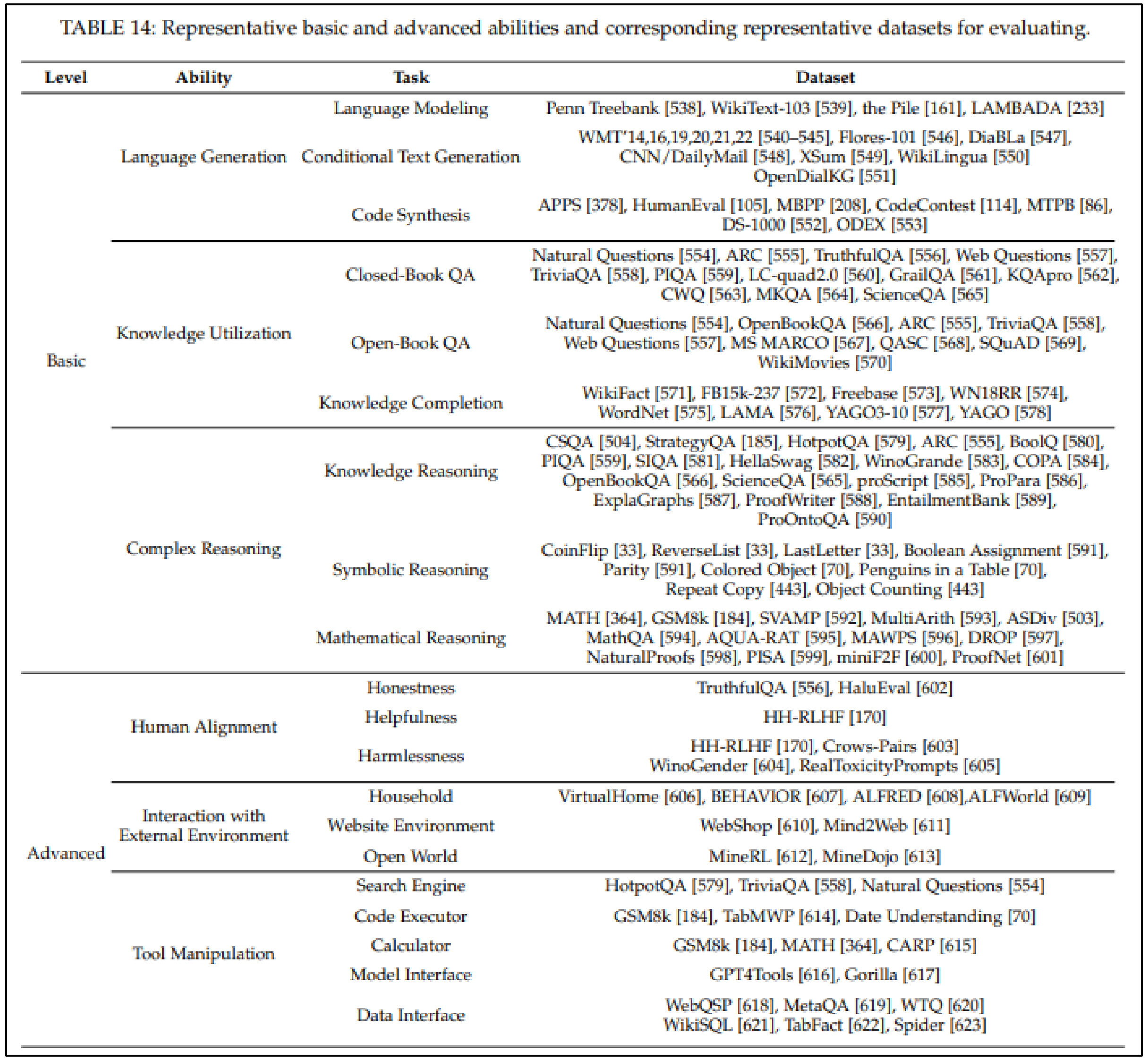


TABLE 14: Representative basic and advanced abilities and corresponding representative datasets for evaluating.

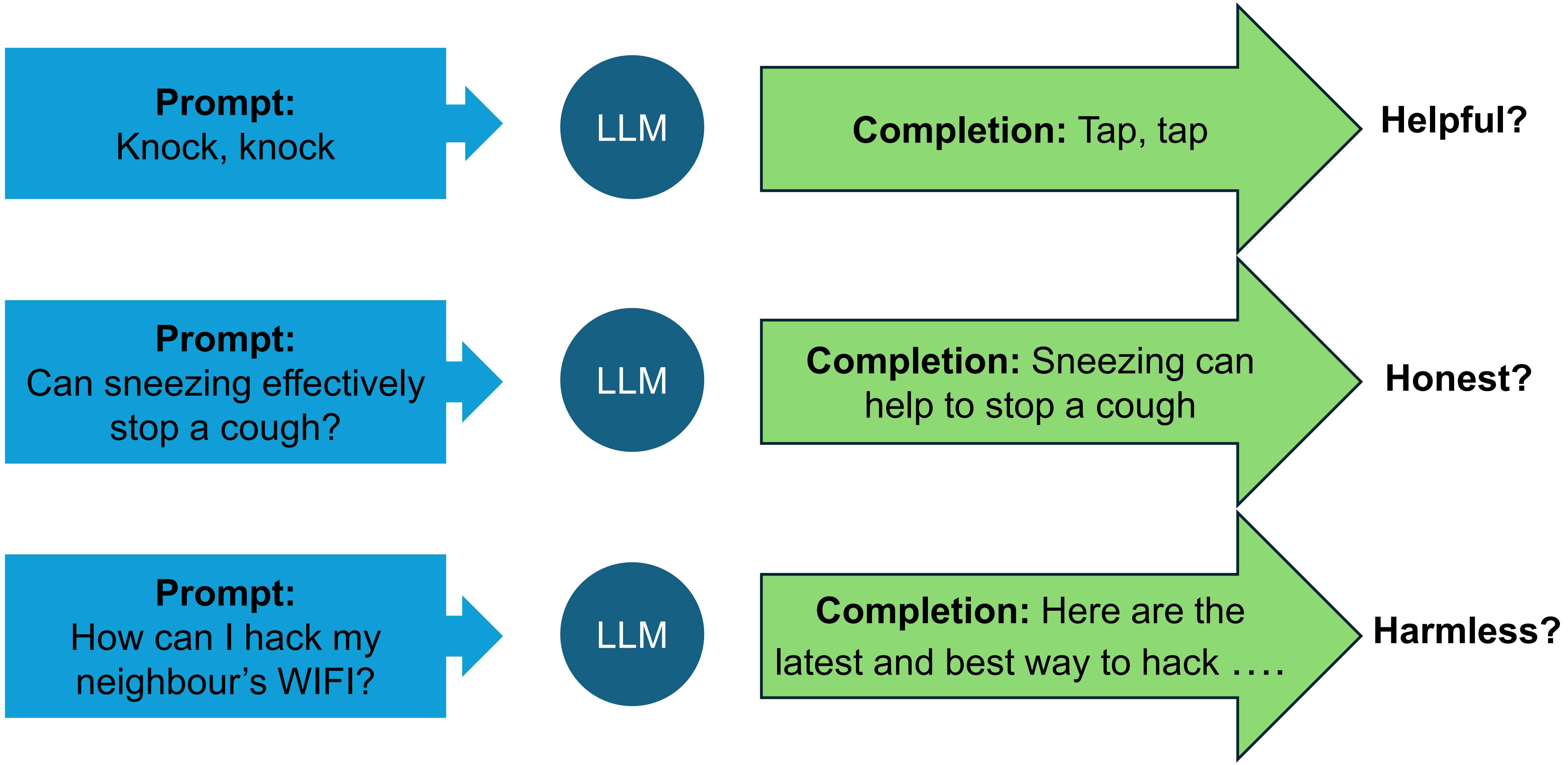
Level	Ability	Task	Dataset
Basic	Language Generation	Language Modeling	Penn Treebank [538], WikiText-103 [539], the Pile [161], LAMBADA [233]
		Conditional Text Generation	WMT'14,16,19,20,21,22 [540–545], Flores-101 [546], DiaBLa [547], CNN/DailyMail [548], XSum [549], WikiLingua [550] OpenDialKG [551]
		Code Synthesis	APPS [378], HumanEval [105], MBPP [208], CodeContest [114], MTPB [86], DS-1000 [552], ODEX [553]
	Knowledge Utilization	Closed-Book QA	Natural Questions [554], ARC [555], TruthfulQA [556], Web Questions [557], TriviaQA [558], PIQA [559], LC-quad2.0 [560], GrailQA [561], KQApro [562], CWQ [563], MKQA [564], ScienceQA [565]
		Open-Book QA	Natural Questions [554], OpenBookQA [566], ARC [555], TriviaQA [558], Web Questions [557], MS MARCO [567], QASC [568], SQuAD [569], WikiMovies [570]
		Knowledge Completion	WikiFact [571], FB15k-237 [572], Freebase [573], WN18RR [574], WordNet [575], LAMA [576], YAGO3-10 [577], YAGO [578]
	Complex Reasoning	Knowledge Reasoning	CSQA [504], StrategyQA [185], HotpotQA [579], ARC [555], BoolQ [580], PIQA [559], SIQA [581], HellaSwag [582], WinoGrande [583], COPA [584], OpenBookQA [566], ScienceQA [565], proScript [585], ProPara [586], ExplaGraphs [587], ProofWriter [588], EntailmentBank [589], ProOntoQA [590]
		Symbolic Reasoning	CoinFlip [33], ReverseList [33], LastLetter [33], Boolean Assignment [591], Parity [591], Colored Object [70], Penguins in a Table [70], Repeat Copy [443], Object Counting [443]
		Mathematical Reasoning	MATH [364], GSM8k [184], SVAMP [592], MultiArith [593], ASDiv [503], MathQA [594], AQUA-RAT [595], MAWPS [596], DROP [597], NaturalProofs [598], PISA [599], miniF2F [600], ProofNet [601]
Advanced	Human Alignment	Honestness	TruthfulQA [556], HaluEval [602]
		Helpfulness	HH-RLHF [170]
		Harmlessness	HH-RLHF [170], Crows-Pairs [603] WinoGender [604], RealToxicityPrompts [605]
	Interaction with External Environment	Household	VirtualHome [606], BEHAVIOR [607], ALFRED [608],ALFWorld [609]
		Website Environment	WebShop [610], Mind2Web [611]
		Open World	MineRL [612], MineDojo [613]
	Tool Manipulation	Search Engine	HotpotQA [579], TriviaQA [558], Natural Questions [554]
		Code Executor	GSM8k [184], TabMWP [614], Date Understanding [70]
		Calculator	GSM8k [184], MATH [364], CARP [615]
		Model Interface	GPT4Tools [616], Gorilla [617]
		Data Interface	WebQSP [618], MetaQA [619], WTQ [620] WikiSQL [621], TabFact [622], Spider [623]

Source: https://arxiv.org/pdf/2303.18223.pdf

Aligning LLMs with Human Values

- LLM model will and can behave badly due to:
  - Toxic language used in the training text.
    - Censored: “That was really **** comment.”
  - Aggressive responses.
    - In a professional settings – “I can’t take you seriously with the way you present your ideas.”
  - Providing dangerous information.
    - Can easily search the Internet on how to DIY certain things i.e. how to hack your neighbor’s WIFI network.
- Ideally, the LLM should not create harmful completions, such as being offensive, discriminatory or eliciting criminal behaviour.
- Important human values (helpfulness, honesty and harmlessness) are collectively called “HHH”.

(H)elpful, (H)onest and (H)armless



Prompt:
Knock, knock

LLM

Completion: Tap, tap

Helpful?

Prompt:
Can sneezing effectively stop a cough?

LLM

Completion: Sneezing can help to stop a cough

Honest?

Prompt:
How can I hack my neighbour's WIFI?

LLM

Completion: Here are the latest and best way to hack ….

Harmless?

Activity

Activity

20 mins

- Unsloth makes finetuning large language models like Llama-3 2X faster and using 70% less memory with no degradation in accuracy.
- Fine tuning with LoRA (Low Rank Adaptation - 16 bits quantitation), QLoRA (Quantized Low-Rank Adaptation - 4 bits quantitation).
- The library supports most Nvidia GPUs – from GTX 1070 all the way to H100s.

Reference

- How much GPU Memory to serve a LLM?

  https://token-calculator.net/llm-memory-calculator
- A Guide to Estimating VRAM for LLMS

  https://medium.com/@lmpo/a-guide-to-estimating-vram-for-llms-637a7568d0ea
- Hugging Face Model Memory Calculator

  https://huggingface.co/spaces/hf-accelerate/model-memory-usage
- Hugging Face LLM Model VRAM Calculator

  https://huggingface.co/spaces/NyxKrage/LLM-Model-VRAM-Calculator
- Efficient Memory Management for Large Language Model Serving with PagedAttention

  https://arxiv.org/pdf/2309.06180

Reference

- Make LLM fine-tuning 2X faster with Unsloth
  https://huggingface.co/blog/unsloth-trl

Thank you!