LLM Sys

11868/11968
Large Language Model Systems

Lei Li

Language Technologies Institute | Carnegie Mellon University School of Computer Science

NN LM

2000

BERT

2018

GPT-3

2020

2022

2023

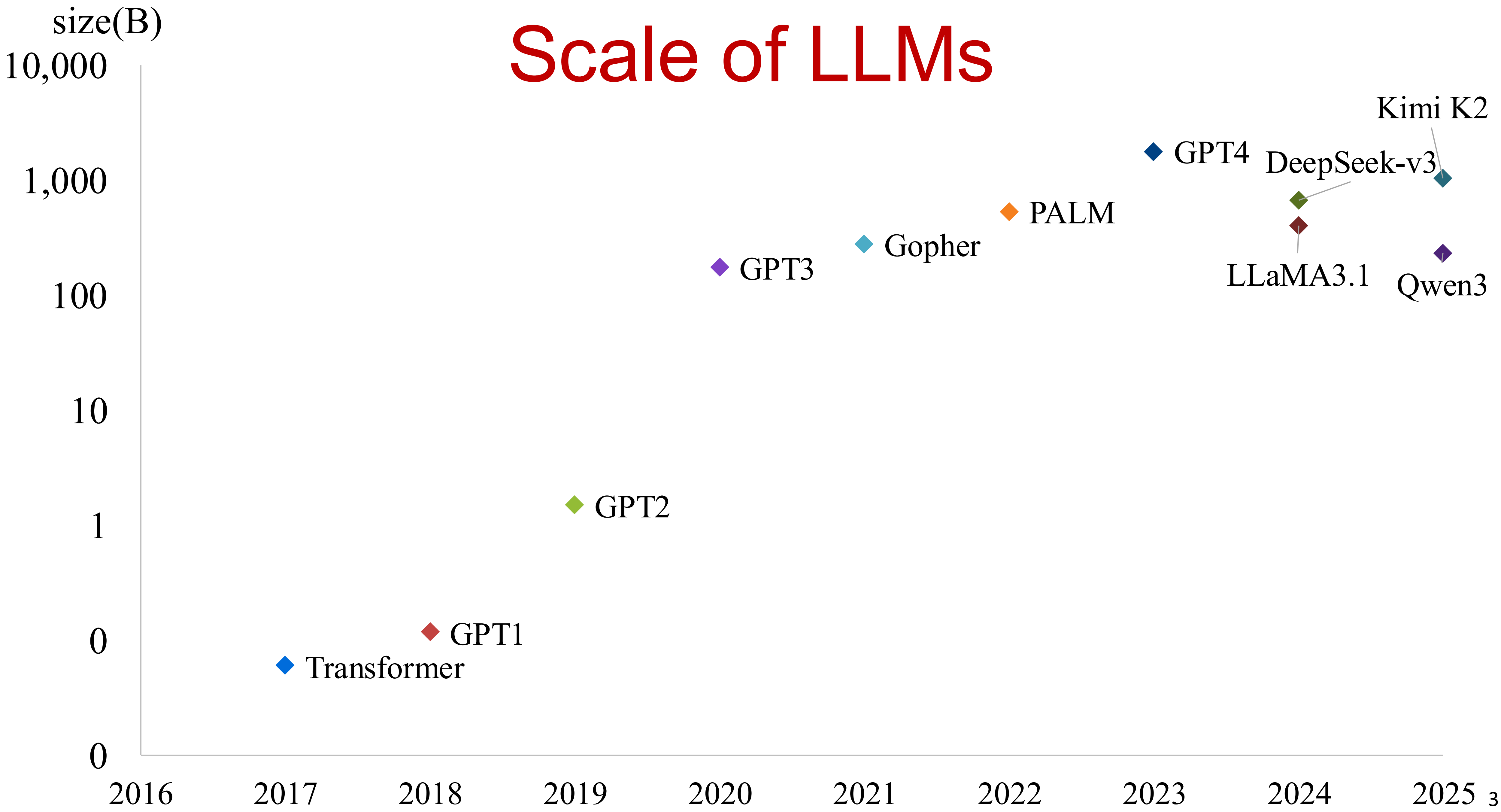
Google Gemini

Supercharge your creativity and productivity

Chat to start writing, planning, learning and more with Google AI

2024

Scale of LLMs



Learning Objective of This Course

- Understand key techniques for modern LLM systems
  - How much resources do you need to train a 100B model?
- Engineering skills to implement key components for LLM systems
  - Fast CUDA kernel for LLM
  - Scalable training system
  - Efficient inference
- Innovation: discover opportunities and solve new critical challenges in LLM system research

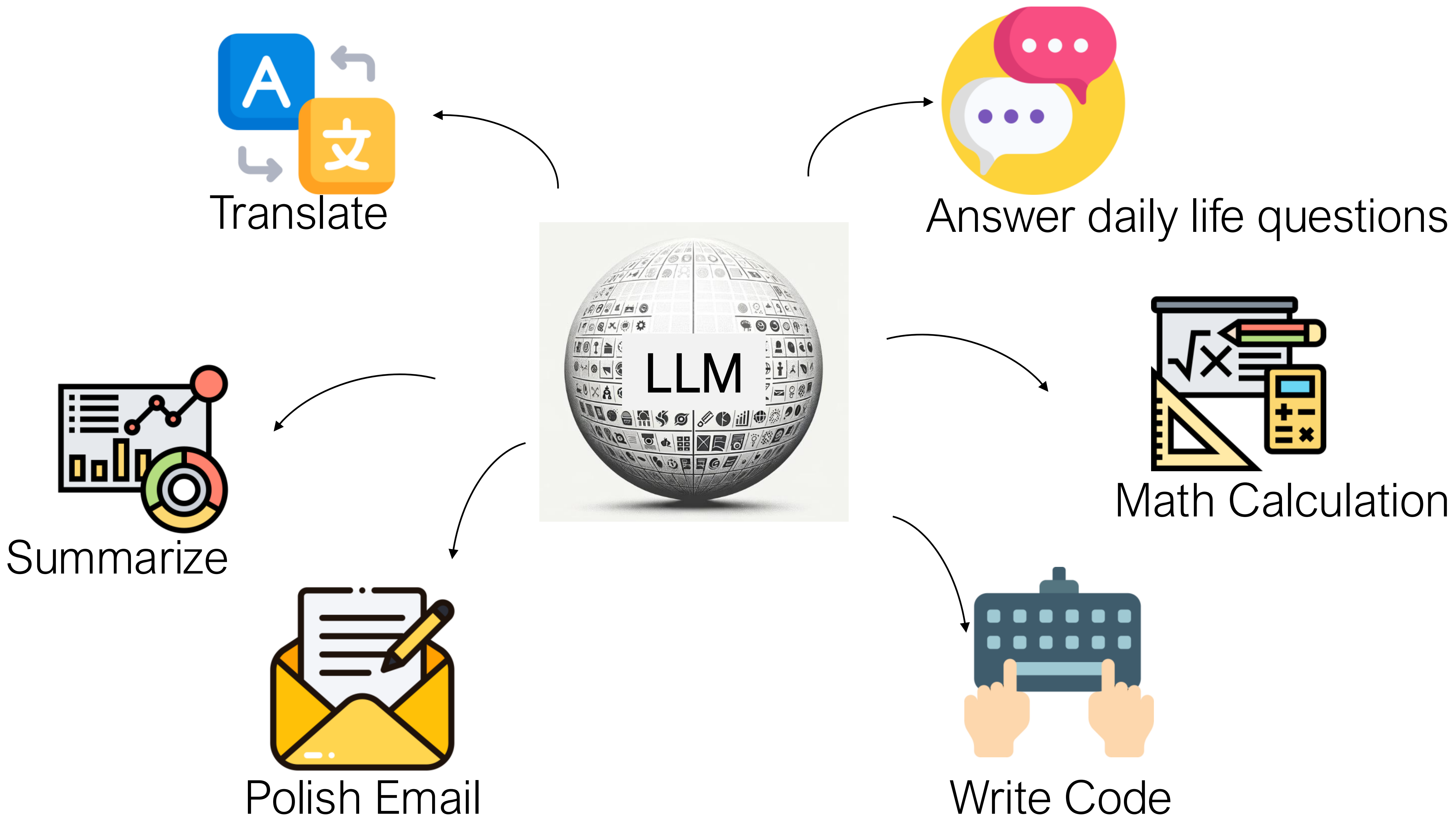
Outline

- Capabilities of LLMs
- Mathematical Foundations
- Challenges in LLM Systems
- Logistics

LE

create a viral tweet about my Large language model systems course at Carnegie Mellon University (CMU, in Pittsburgh). Please refer to the course description at https://llmsystem.github.io/llmsystem2025fall/

🌟 Ready to build the future of AI? 🌟
This Fall at CMU: "Large Language Model Systems" 🧠💻
👉 Dive into LLM design, implementation & scaling.
👉 Tackle cutting-edge topics like agents, fine-tuning & efficiency.
👉 Learn in the AI innovation hub: Pittsburgh.
📚 Details Here
#AI #LLM #CMU #TechInnovation


LLM
Translate
Answer daily life questions
Math Calculation
Summarize
Polish Email
Write Code

Translation

Google MT

他上班天天摸鱼

Tā shàngbān tiāntiān mō yú

7 / 5,000

He fishes every day at work

ChatGPT/GPT4

You

Please translate "他上班天天摸鱼"

ChatGPT

The phrase "他上班天天摸鱼" translates to "He slacks off at work every day" in English. Here, "摸鱼" (mō yú) is a Chinese slang term that literally translates to "touching fish," but it's commonly used to mean slacking off or being unproductive, especially at work.

ChatBot to Mimic a User

- https://github.com/Ads-cmu/WhatsApp-Llama
- Train on your WhatsApp chat data (about 10 chat histories are adequate), about a couple hours
- Generate responses to mimic user style

Rewrite Text with Style

LE **You**

Please write the following sentence in Yoda's style. Here are a few examples.
You have much to learn. -> Much to learn, you still have.
I would say you have found someone, hmmm? -> Found someone, you have, I would say, hmmm?

You should not drive that fast. ->

ChatGPT

Drive that fast, you should not.

Commonsense Reasoning

LE **You**

Fill in the blank: Steven was disgusted that Robert chose to cook pork for dinner, because _ was a vegetarian.
"option1": "Steven", "option2": "Robert"

ChatGPT

The correct answer is "Steven"

Math Reasoning

You

Roger has 4 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

ChatGPT

Roger initially has 4 tennis balls. When he buys 2 more cans of tennis balls, and each can has 3 tennis balls, he adds 2 cans * 3 tennis balls per can = 6 tennis balls to his collection.
So, Roger now has a total of 4 initial tennis balls + 6 newly purchased tennis balls = 10 tennis balls.

NBC NEWS

ChatGPT passes MBA exam given by a Wharton professor

The bot's performance on the test has "important implications for business school education," wrote Christian Terwiesch, a professor at the University of Pennsylvania's Wharton School.

Jan. 23, 2023, 4:46 PM EST / Updated Jan. 24, 2023, 1:19 PM EST

By Kalhan Rosenblatt

ChatGPT scored between a B- and B on the MBA exam.

LE **You**

The Circored process produces direct reduced iron in Latin America. The production process starts with a pile of iron ore fines. The ore is moved through seven process steps before it emerges as finished direct-reduced-iron briquettes. The steps and their capacities are as follows:

- Pre-Heater: 60 tons per hour
- Lock Hoppers: 55 tons per hour
- First reactor: this reactor holds up to 14 tons of iron and iron needs to spend 15 minutes in the reactor
- Second reactor: this reactor holds up to 200 tons of iron and iron needs to spend 4 hours in the reactor
- Flash heater: 62.5 tons per hour
- Discharge: 59 tons per hour
- Briquetting: three machines of 27.5 tons per hour each

Where in the process is the bottleneck and what is the overall process capacity? Justify your answer.

Christian Terwiesch. Would Chat GPT3 Get a Wharton MBA? 2023.

ChatGPT

To identify the bottleneck in the Circored process and the overall process capacity, we need to look at the capacities of each step. The bottleneck in any production process is the step with the lowest throughput rate, as it limits the rate at which the entire system can operate.

Here are the capacities given:

1. **Pre-Heater**: 60 tons per hour
2. **Lock Hoppers**: 55 tons per hour
3. **First Reactor**: Holds 14 tons total; each batch takes 15 minutes, so its rate is 14 tons0.25 hours=56 tons per hour0.25 hours14 tons=56 tons per hour
4. **Second Reactor**: Holds 200 tons total; each batch takes 4 hours, so its rate is 200 tons4 hours=50 tons per hour4 hours200 tons=50 tons per hour
5. **Flash Heater**: 62.5 tons per hour
6. **Discharge**: 59 tons per hour
7. **Briquetting**: Three machines at 27.5 tons per hour each, so 3×27.5 tons per hour=82.5 tons per hour3×27.5 tons per hour=82.5 tons per hour

Now, let's calculate and compare the capacities of each step to identify the bottleneck.

The bottleneck in the Circored process is the **Second Reactor**, with a capacity of 50 tons per hour. …

ChatGPT even generates code to verify

```
# Capacities of each step in tons per hour
pre_heater = 60
lock_hoppers = 55
first_reactor = 14 / 0.25
second_reactor = 200 / 4
flash_heater = 62.5
discharge = 59
briquetting = 3 * 27.5
# Identifying the bottleneck (the smallest capacity) and overall process capacity
capacities = {
 "Pre-Heater": pre_heater,
 "Lock Hoppers": lock_hoppers,
 "First Reactor": first_reactor,
 "Second Reactor": second_reactor,
 "Flash Heater": flash_heater,
 "Discharge": discharge,
 "Briquetting": briquetting }
bottleneck_step = min(capacities, key=capacities.get)
bottleneck_capacity = capacities[bottleneck_step]

(bottleneck_step, bottleneck_capacity)
```

It connects to a python engine, and executes the program to obtain result

Result

```
('Second Reactor', 50.0)
```

LE **You**

You are given a 0-indexed array mountain. Your task is to find all the peaks in the mountain array. please write a python program.

ChatGPT

```
def find_peaks(mountain):
    peaks = []
    for i in range(1, len(mountain) - 1):
        if mountain[i] > mountain[i - 1] and 
mountain[i] > mountain[i + 1]:
            peaks.append(i)
    return peaks
```

please draw a llama climbing along the "walking into the sky" on the campus of CMU

please draw an image for my slides about ”The power of Large Language Models"

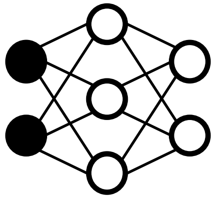
Outline

- Capabilities of LLMs
- Mathematical Foundations
- Challenges in LLM Systems
- Logistics

Probability Model for Next Token

P(next word y_t | Prompt x, previous words $y_{1:t-1}$)

Santa Barbara has very nice ____



Pittsburgh is a city of ____

beach	0.5
weather	0.4
snow	0.01
bridges	0.6
corn	0.02

Mathematics of Language Model

Probability("Pittsburgh is a city of bridges")

$$= P(\text{“}Pittsburgh\text{”}) \cdot P(\text{“}is\text{”}|\text{“}Pittsburgh\text{”})$$
$$\cdot\, P(\text{“}a\text{”}|\text{”}Pittsburgh\ is\text{”}) \cdot P(\text{“}city\text{”}|\ldots) \cdot P(\text{“}of\text{”}|\ldots)$$
$$\cdot\, P(\text{“}bridges\text{”}|\ldots)$$

$$\text{Prob.}(x_{1..T}) = \prod_{t=1}^{T} \underline{\mathrm{P}(\mathrm{x}_{t+1}|\mathrm{x}_{1..t})}$$

Predicting using Neural Nets
(Transformer network, CNN, RNN)

Why is ChatGPT changing AI landscape

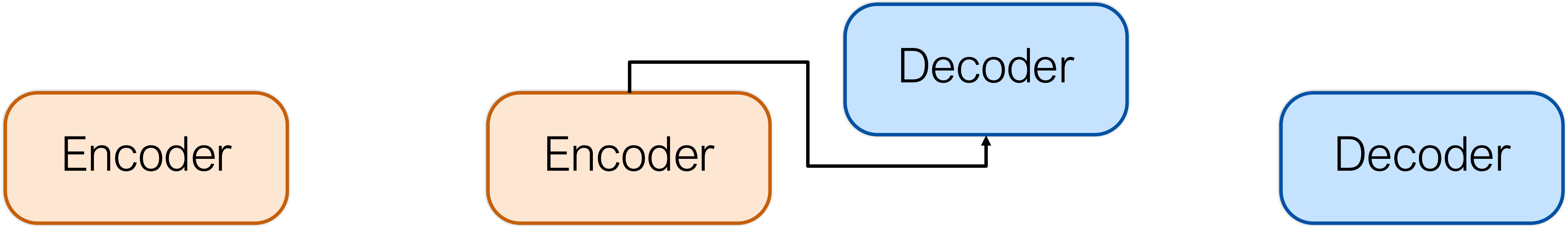
- Pre-training on very large raw data (300B tokens) + small human feedback
- Instruction following – easy to use through natural instruction
- In-context learning – Generalize well to versatile tasks, by showing a few examples at use time.

Type of Language Models

Encoder-only
Masked LM
Non-autoregressive

Encoder-decoder

Decoder-only
Autoregressive



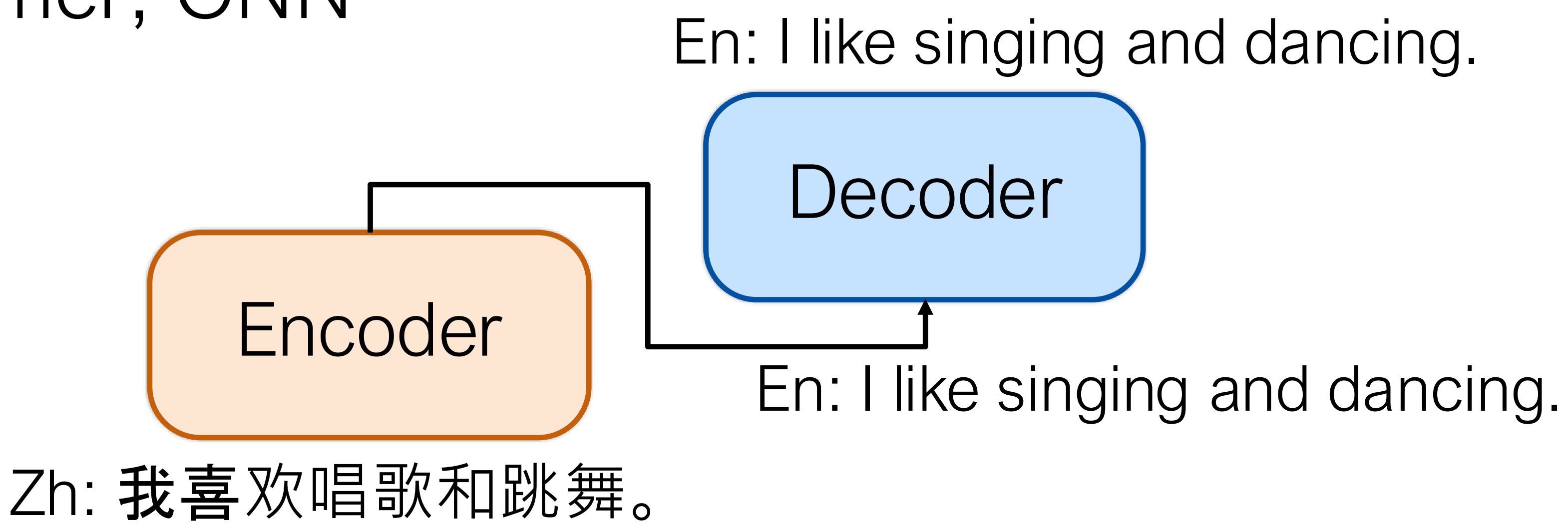
Encoder-only Language Model

- Masked Prediction: BERT $P(x_{mask}|x)$

- Can also be used to generate language
    - e.g. NAT, REDER (reversible duplex model)

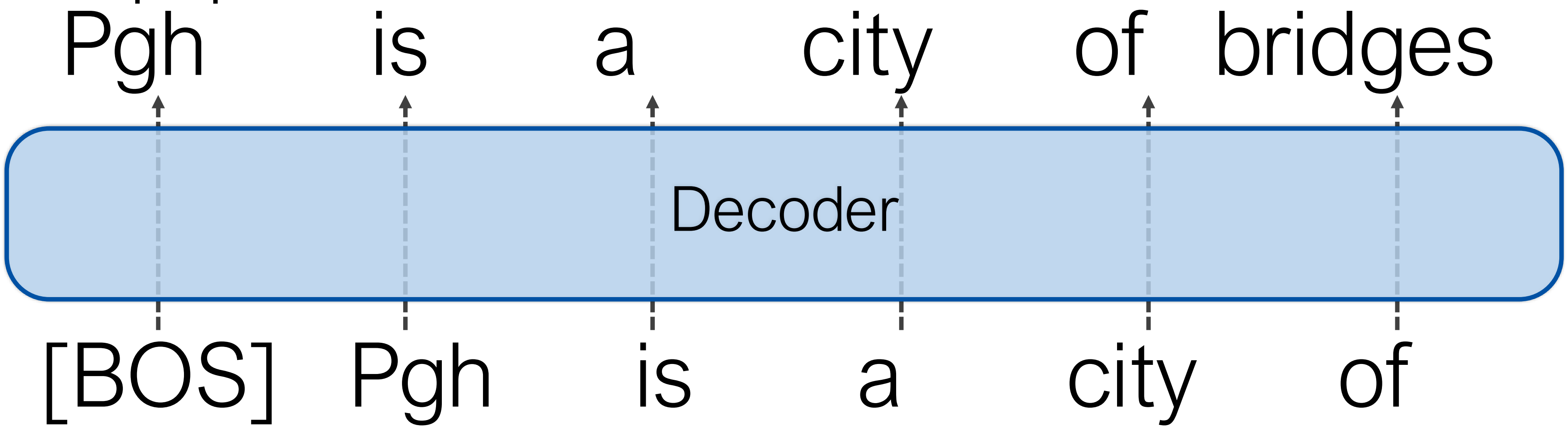
Encoder-Decoder Language Model

- Model the probability $P(Y|X)$
- Encoder may choose Bi-LSTM, Transformer, CNN
- Decoder may choose LSTM, Transformer or Non-auto Transformer, CNN

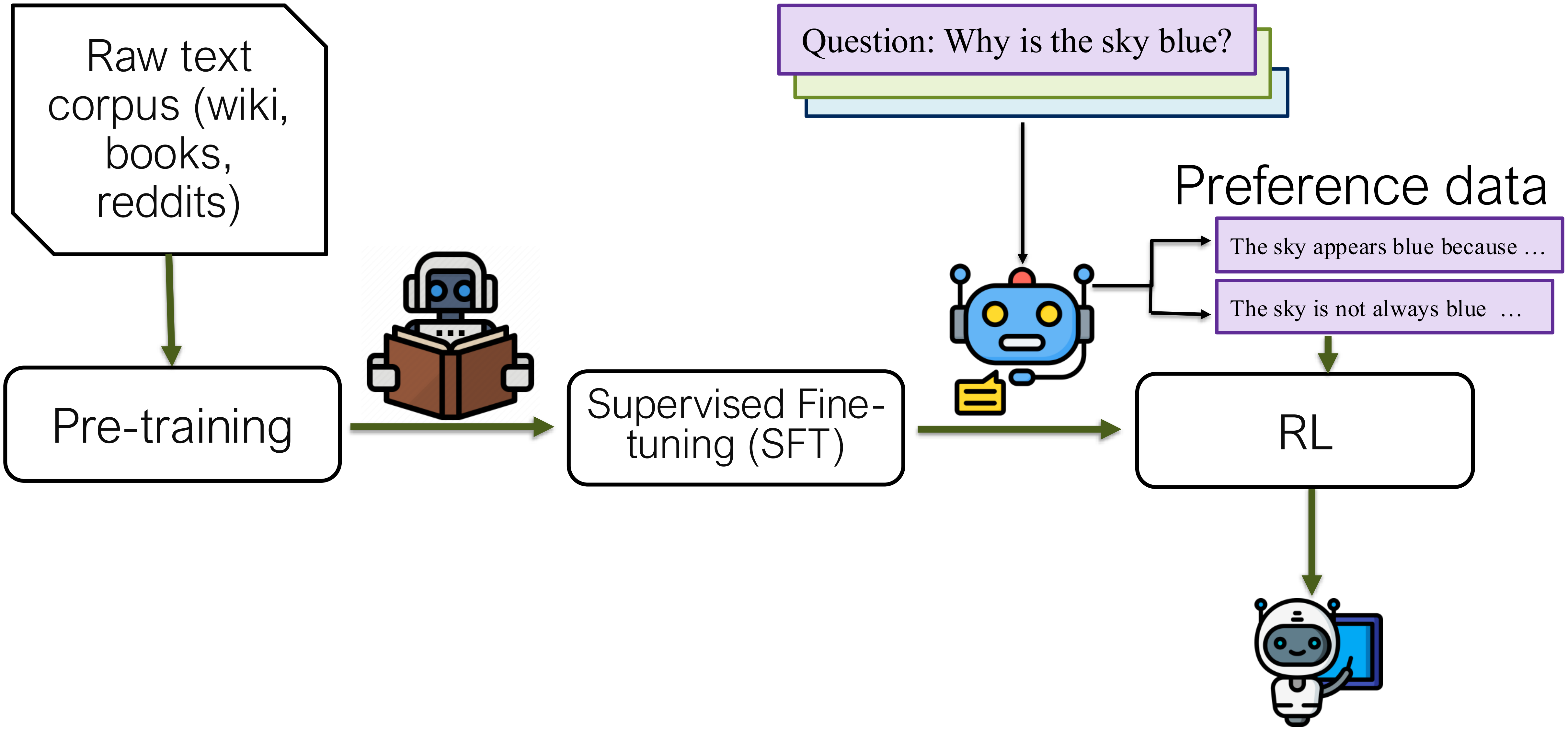


Decoder-only Language Model

- Causal Language Model: $P(X) = \prod_{n=1}^{N} P(x_n | x_{1..n-1})$
- Model choice: Transformer or (uni-directional) LSTM
- Most popular choice of LLM architecture



LLM Learning Framework



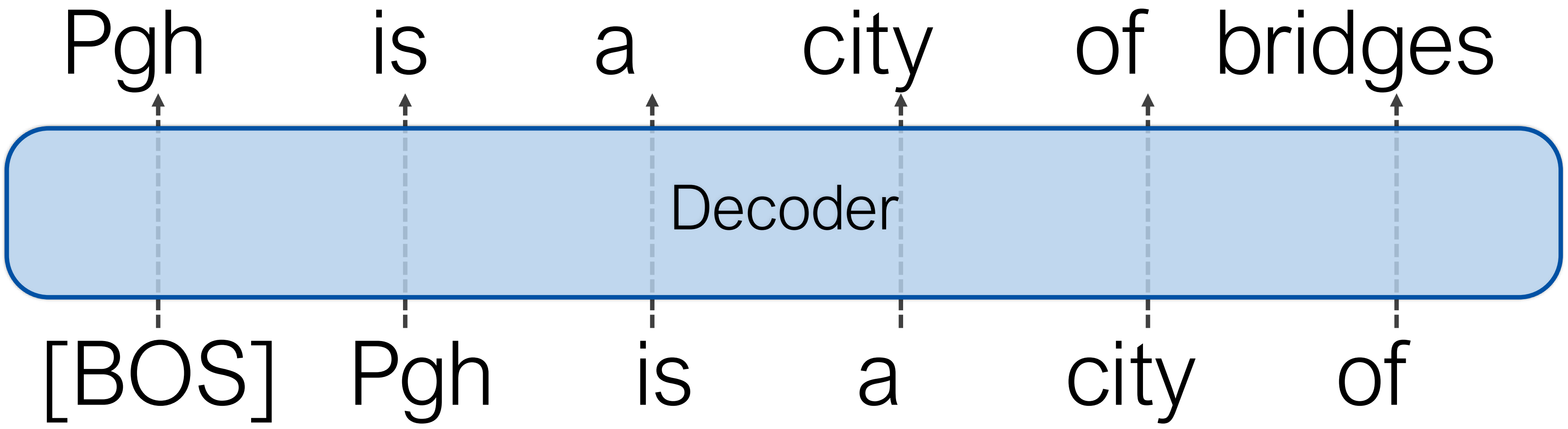
Pre-training of Language Models

- Data: raw text corpus
  - the larger the better, often crawled from the web
  - quality is important: Wikipedia, books, filtered Reddit posts
- Design of Model Architecture
  - Sparse Attention, Mixture of Expert, etc.
- Training loss:
  - cross-entropy loss for next token prediction.

Training Objective

- Cross entropy loss

$$CE = -\frac{1}{N}\sum_{n=1}^{N} \log P_\theta(x_n | x_{<n})$$



Measuring Performance of Language Model

- Perplexity: for open-end generation

$$PPL = \exp -\frac{1}{N}\sum_{n=1}^{N} \log P_{LM}(x_n | x_{<n})$$

- Reference-based metric:
  - SEScore2/COMET/BLEURT: model-based score
  - BLEU (or ROUGE): based on n-gram matching, old but classic
  - InstructScore: explainable score [Xu et al 2023]

Measuring Performance of Language Model

- Using downstream task performance metrics
  - Named entity recognition: F1
  - Question Answering: accuracy, matching rate
  - Code generation: pass-rate @k
  - Retrieval: ndcg
  - Summarization: ROUGE
  - Translation: COMET/SEScore2/BLEU

Performance of LLM

- GPT3 trained on 300B tokens
- Both model scale and data are important

Perplexity

Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Outline

- Capabilities of LLMs
- Mathematical Foundations
- Challenges in LLM Systems
- Logistics

Modern LLMs

- Generalist
  - Formulate any AI tasks as token-based sequence generation
- Instructibility
  - Interactive system that is able to take natural or infinite commands
- Agentic
  - Call external tools, execute actions, take feedback

System Design for LLM

- Key system problem: compute (train/inference) larger LLMs on bigger datasets with fewer resources (GPU/memory/power) faster
- Right Abstraction
    - Designing useful building blocks that hide complexity from application developers – computation at different levels
- Tradeoffs
    - What are the fundamental constraints and main success metrics?

Computation in Language Models

- Common network layer
  - Multi-head Attention
  - Layer norm
  - Dropout
  - Linear layer
  - Nonlinear activation: Tanh, RELU, GELU
  - Softmax
- Low level operators:
  - Matrix/Tensor multiplication
  - Reduction: summation, avg
  - Map: element-wise applying
  - Memory movement

System Challenges at Different Abstraction Levels

distributed / parallel system

scaling to gigantic model
gigantic dataset
very long sequence/context
partitioning/scheduling/comm

DL Frameworks

easy dev/modify model
easy dev ML algorithm

Operators on block of data

fast CUDA/TPU kernel
data/model compression

LLM needs Model-Algorithm-System Co-design

- Scaling is all you need! – scale up and scale down
- Joint design of
  - Model architecture
  - Training/inference Algorithms
  - Software optimization (partitioning, scheduling, data movement, latency hiding)
  - Hardware Acceleration (device-specific instructions)

Computation versus Data Transfer

- Making computation fast is not enough
- Data Transfer needs time
  - Large model has a lot of parameters
    - Transferring parameters/gradients across devices/nodes can take more time
  - Data sample in batch versus single single sequence
- needs large working memory for long context LLM

Programming Models

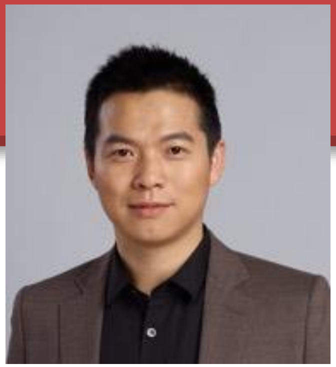
- What are good abstractions to build AI applications?
  - Upper level: integrating models into a product system, tracking and improving product quality over time
  - Middle level: building model training and inference software, streaming dataflow
  - Lower level: building GPU kernels, compilers, etc

- A good abstraction successfully frees the programmer of one or more concerns (e.g., performance or failures) while supporting a wide range of apps on top

Outline

- Capabilities of LLMs
- Mathematical Foundations
- Challenges in LLM Systems
- Logistics

Instructor Prof. Lei Li



Carnegie Mellon University
Language Technologies Institute

Research Area:
Large language models
(Safety, Agents, Alignment)
Machine Translation
AI drug discovery

2011 — CMU PhD SCS, CSD

Berkeley UNIVERSITY OF CALIFORNIA — PostDoc EECS

2014 — Baidu 百度 — Principal Scientist Institute of Deep Learning

2016 — ByteDance — Founding Director AI Lab

2021 — Assist. Prof. CS UC SANTA BARBARA

2023 — Carnegie Mellon University

Highlight of Past Work

- led and developed ByteDance's machine translation system VolcTrans and AI writing system Xiaomingbot, which have been deployed in products (Toutiao, Douyin, Tiktok, Lark), serving over a billion users.
- Deep learning system research
  - LightSeq: An acceleration library for Transformer models
- Machine translation: LegoMT, VOLT(ACL21 best paper), mRASP, GLAT, WACO, InstructScore
- Large Language Models: SALAM, ToolDec, ALGO, Ginsew
- AI for Science: drug discovery (MARS), protein design (EnzyGen)

TA

Aditya Tummala

Danqing Wang

Jackey Hua

Sreeram Vennam

office hour: see canvas page

Reasons not to take this class … ☹

If you plan to drop this class, please drop it soon so others can enroll!

- If you want to learn how to create DL models and use TensorFlow, PyTorch
  - Take 11685/11785
- If you want learn how to use LLM but not interested in the underlying system
  - Take 11667
- You are unable to attend lecture, complete homework/project
  - Warning: requires system implementation

Prerequisites

In order to take this course, you need to be proficient with:

- math knowledge (linear algebra, calculus, and probability&stats)
- programming skills in Python and C, C++, or Java (15122 or equivalent)
- prior knowledge of machine learning is preferred but not required.

Assignments

- The course will consist of programming-based homework assignments
- Homeworks are done individually
- Through the assignments you will develop the main components for miniTorch and train a Transformer-based LLM
- Expected to write Python and C++/CUDA code
  - prior knowledge of CUDA is helpful but not required.

Project

- Team of 2~3
- Two types of projects:
  - Implement a LLM system paper as an additional component to miniTorch (should not use external PyTorch/Tensorflow code)
  - Research oriented project that solves a challenge in LLM system (with the goal of submitting to a conference)
- Need to submit Project Proposal/Mid-term/Final Report
  - Please include: project description, proposed method, data, evaluation procedure/metric, computation, other resources, preliminary results.
- Poster presentation (ideally with demo) in-class

Computing Resources

- GPU: We already requested PSC bridge2 credits

- Additionally, you may use google colab on your own
  - no support provided.

Late Policy

- https://llmsystem.github.io/llmsystem2026spring/docs/Logistics#late-day-policy

Next

- Inside a GPU
- GPU Programming