IJCAI JEJU 2024

IJCAI 2024 Tutorial T08
Time: 2024-08-03 14:00-17:30
Location: Room 1F BO-04 Yeongju B

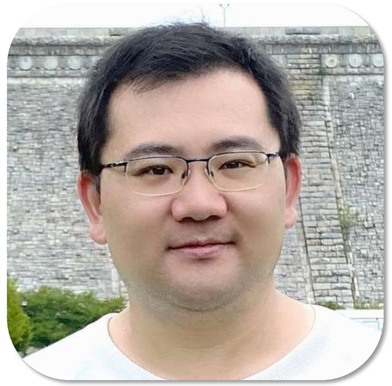
Beyond Human Creativity: A Tutorial on Advancements in AI Generated Content (AIGC)

Bang Liu
University of Montreal, Mila

Yu Chen
ANYTIME.AI

Manling Li
Stanford, Northwestern

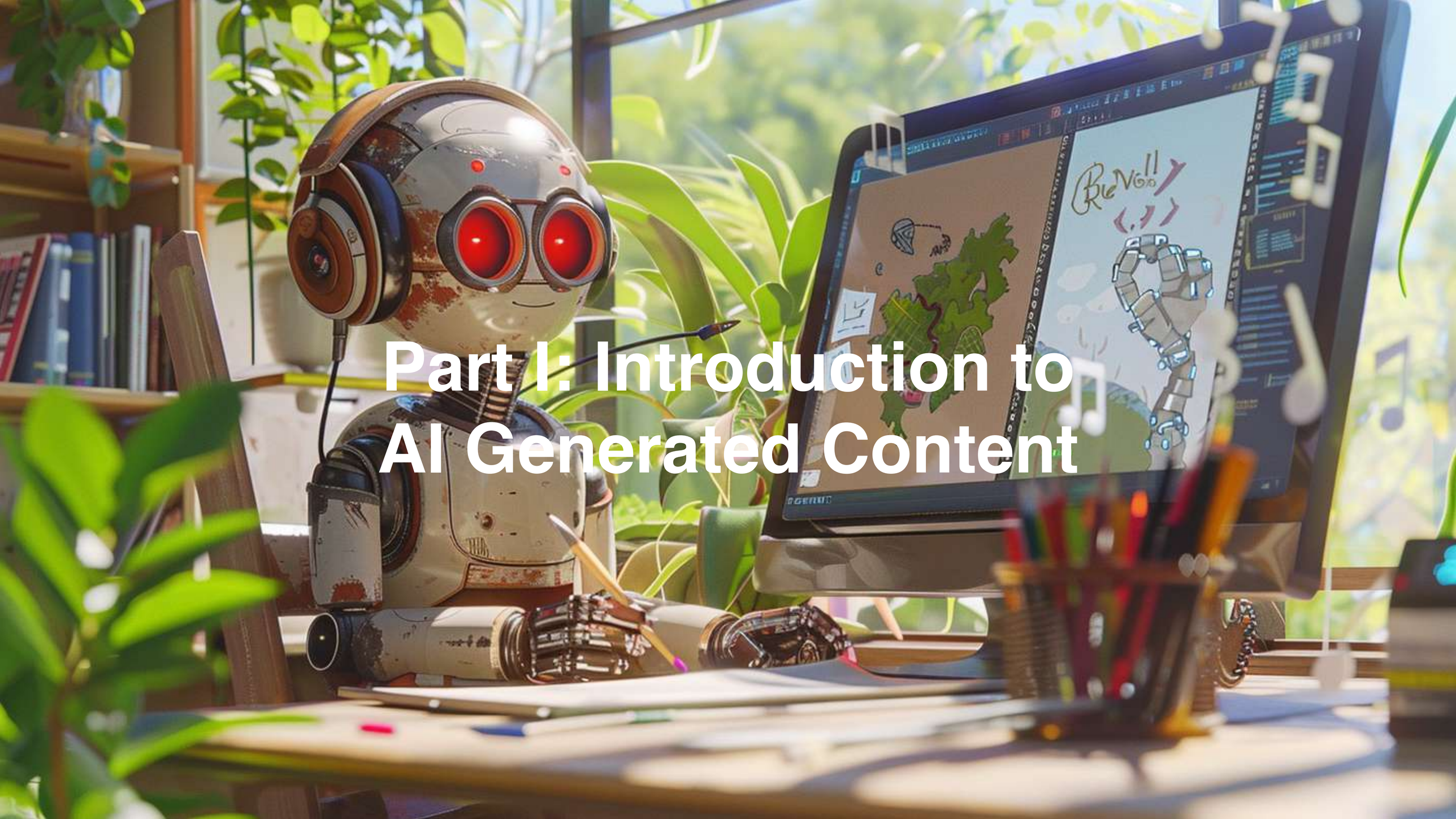
Heng Ji
UIUC



Lingfei Wu
ANYTIME.AI

Outline

I. Introduction to AI Generated Content
II. Text Generation with Large Language Models
III. Visual Content Generation with Large Generative Models
IV. Major Challenges and Future Directions of Generative AI
V. Concluding Remarks



Part I: Introduction to AI Generated Content

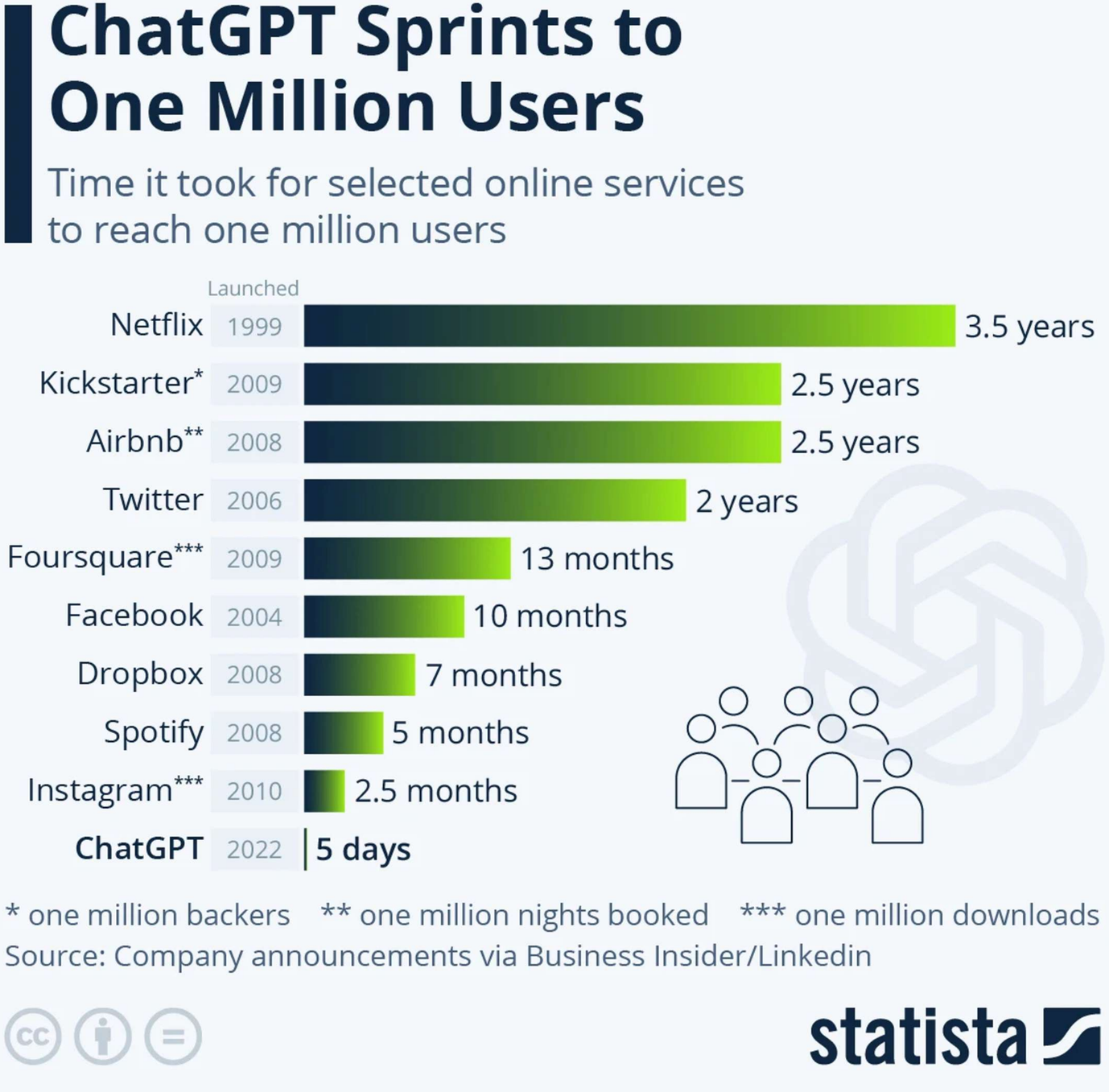
What AI can do as for now?

ChatGPT

https://chatgpt.com/

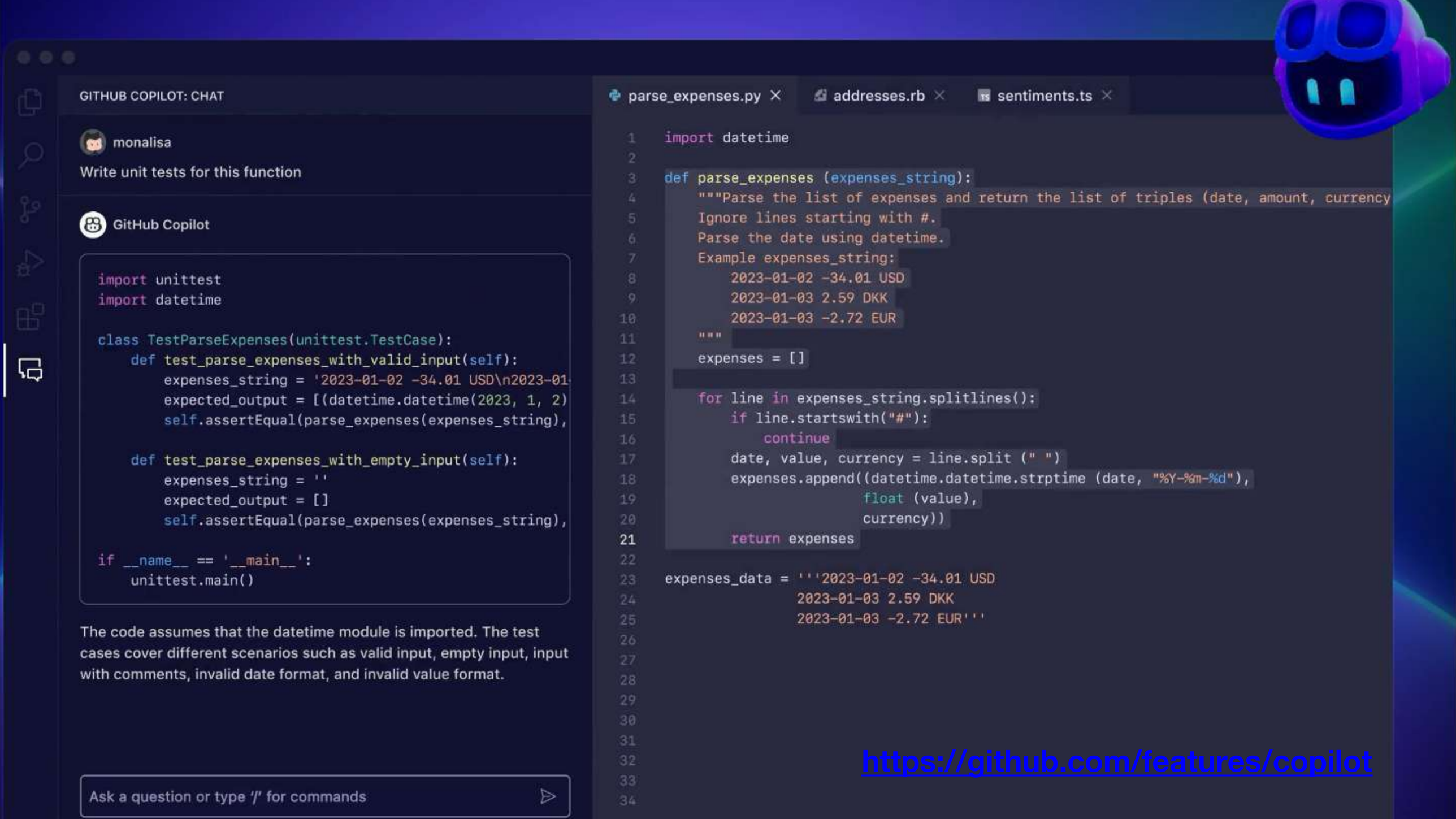
The most successful launch of a tech product in history

Its natural language Interface and frictionless accessibility quickly engaged **millions of users.**



Sora

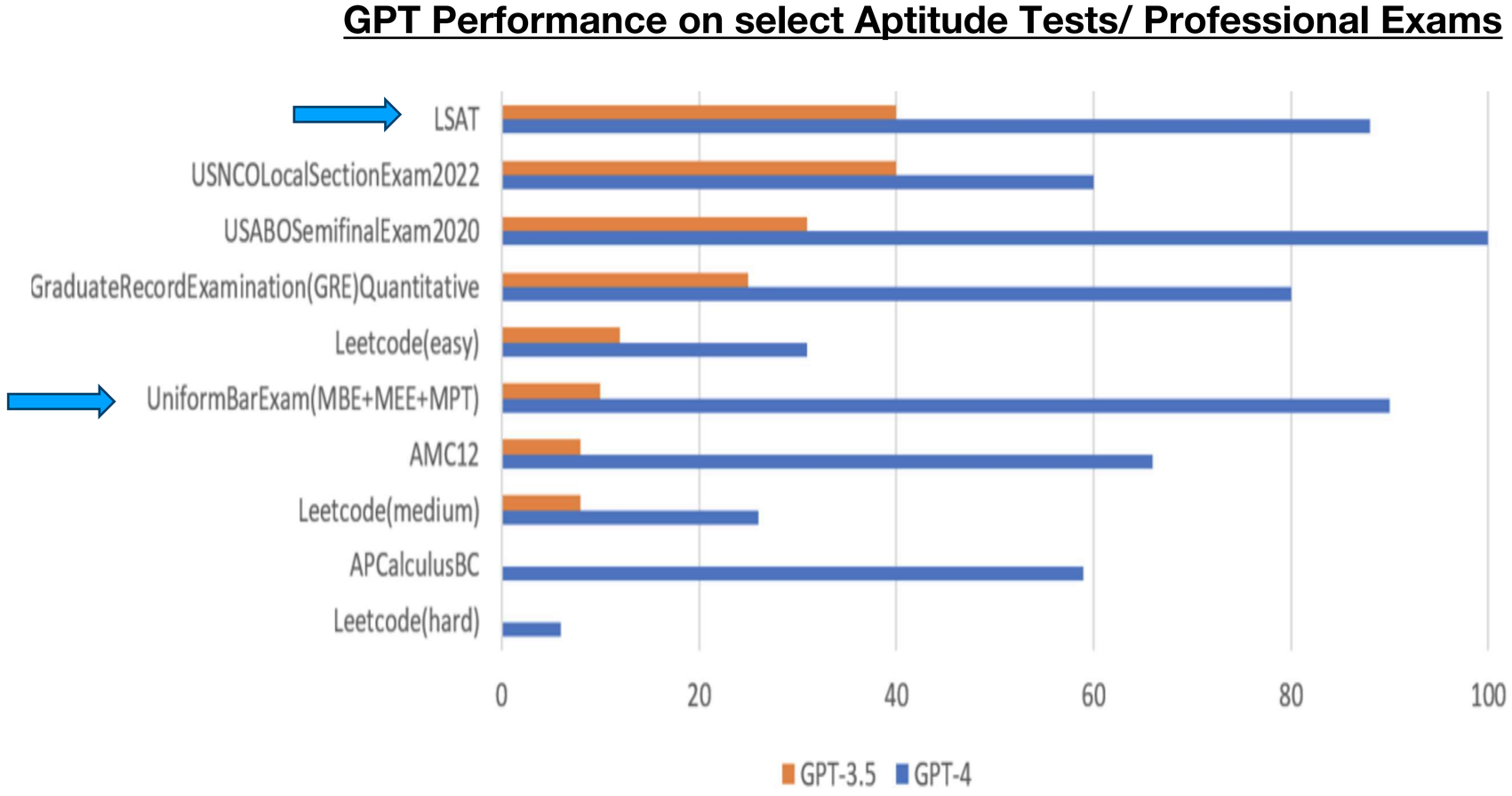
https://openai.com/index/s


GITHUB COPILOT: CHAT
monalisa
Write unit tests for this function
GitHub Copilot
import unittest
import datetime
class TestParseExpenses(unittest.TestCase):
def test_parse_expenses_with_valid_input(self):
expenses_string = '2023-01-02 -34.01 USD\n2023-01
expected_output = [(datetime.datetime(2023, 1, 2)
self.assertEqual(parse_expenses(expenses_string),
def test_parse_expenses_with_empty_input(self):
expenses_string = ''
expected_output = []
self.assertEqual(parse_expenses(expenses_string),
if __name__ == '__main__':
unittest.main()
The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.
Ask a question or type '/' for commands
parse_expenses.py
addresses.rb
sentiments.ts
import datetime
def parse_expenses (expenses_string):
"""Parse the list of expenses and return the list of triples (date, amount, currency
Ignore lines starting with #.
Parse the date using datetime.
Example expenses_string:
2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR
"""
expenses = []
for line in expenses_string.splitlines():
if line.startswith("#"):
continue
date, value, currency = line.split (" ")
expenses.append((datetime.datetime.strptime (date, "%Y-%m-%d"),
float (value),
currency))
return expenses
expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
https://github.com/features/copilot

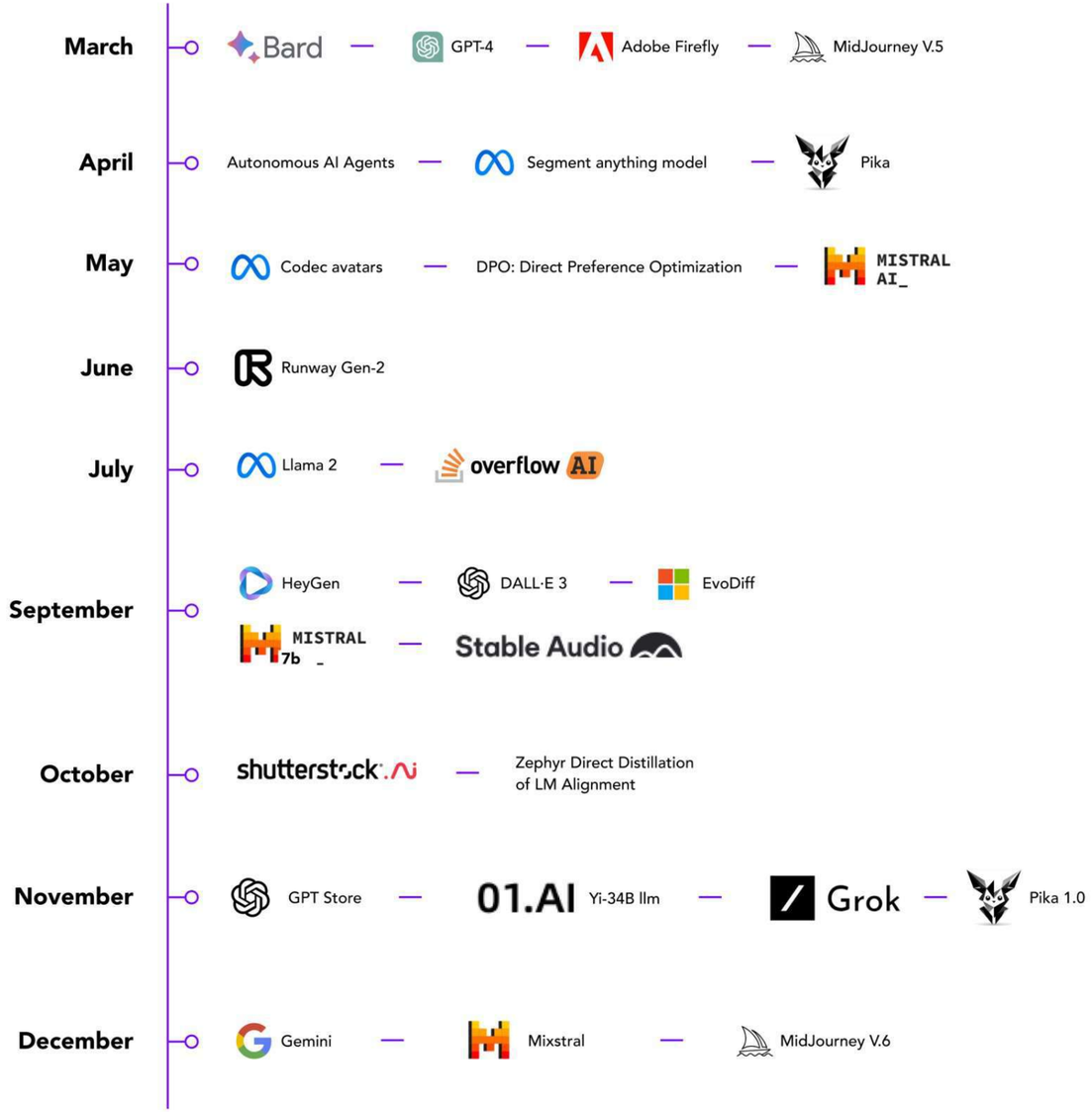
Generative AI NPC

Exponential Growth in AI Capability

- 5 months between release of ChatGPT (GPT3.5, Nov 2022) and GPT4 (Mar 14, 2023)
- **LSAT** Proficiency = 40% (GPT3.5) to nearly 90% (GPT4)
- **UBE** = 10% (GPT3.5) to 90% (GPT4)

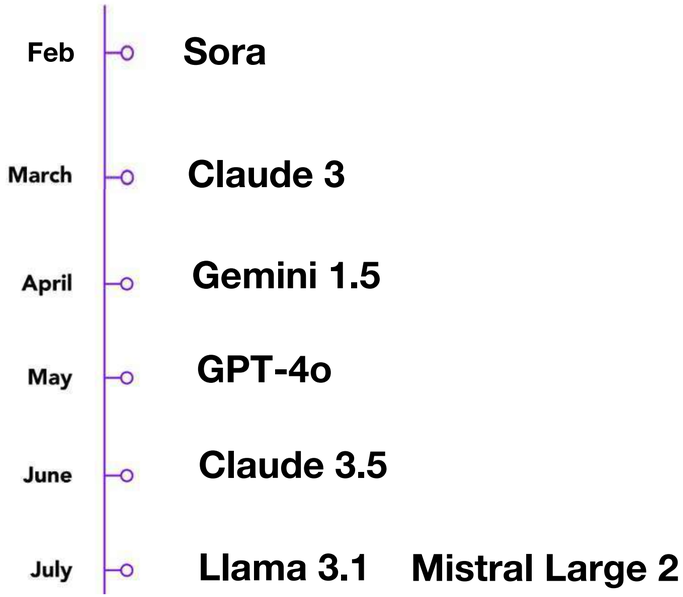


The Most Remarkable AI Advancements of 2023



EVERYPIXEL

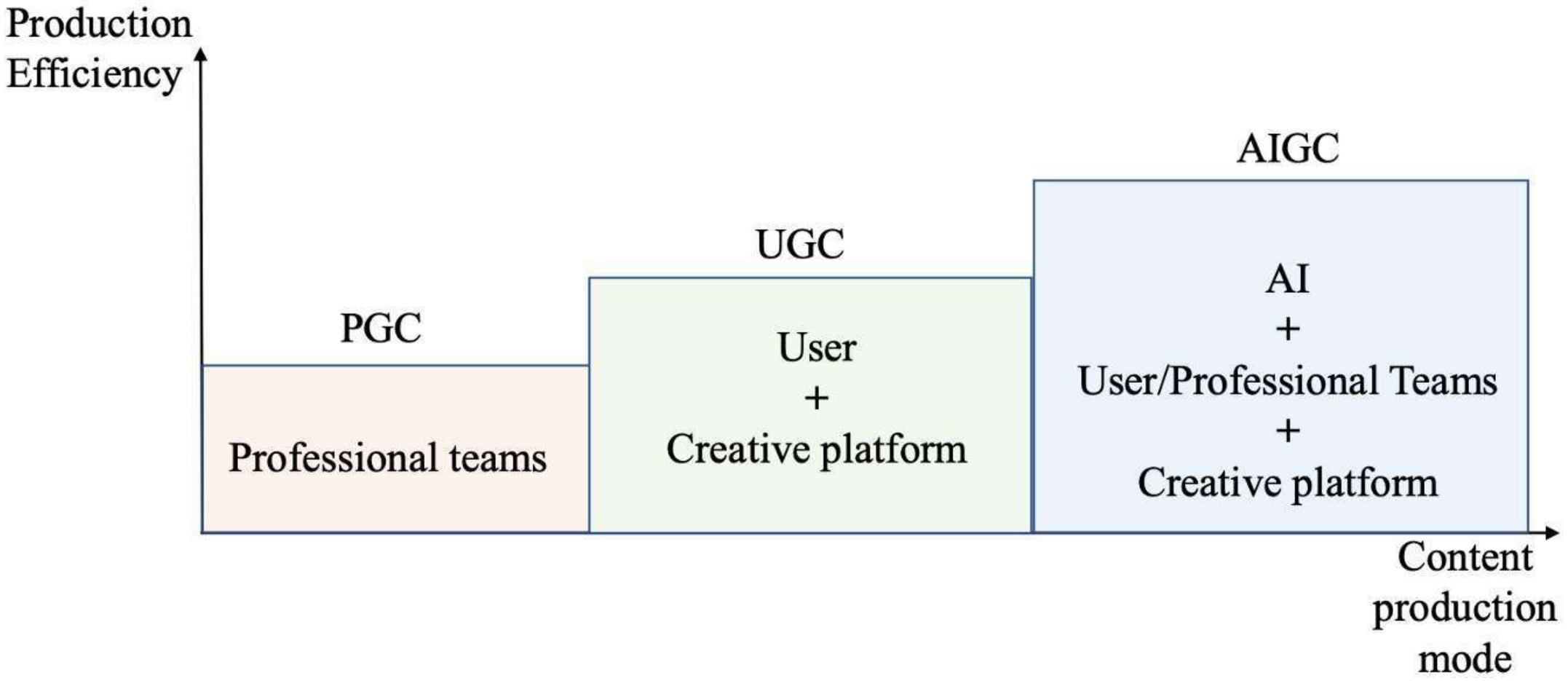
The Most Remarkable AI Advancements of 2024



Reference: 2023: The Year of AI

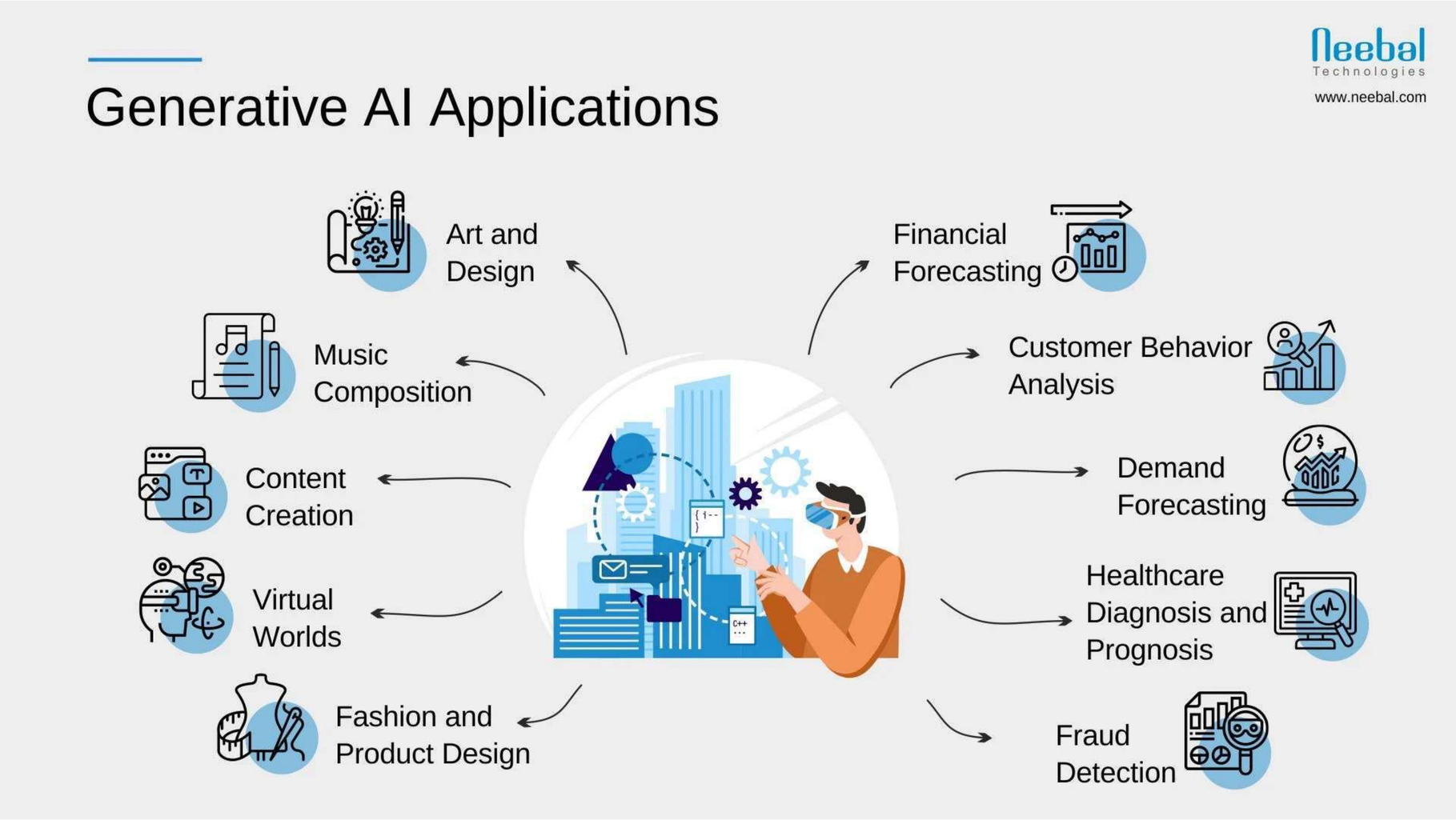
What are the impacts of AIGC?

12 AI-Generated Content



Reference: https://arxiv.org/pdf/2304.06632.pdf

13 AIGC Applications Empowered by Generative AI

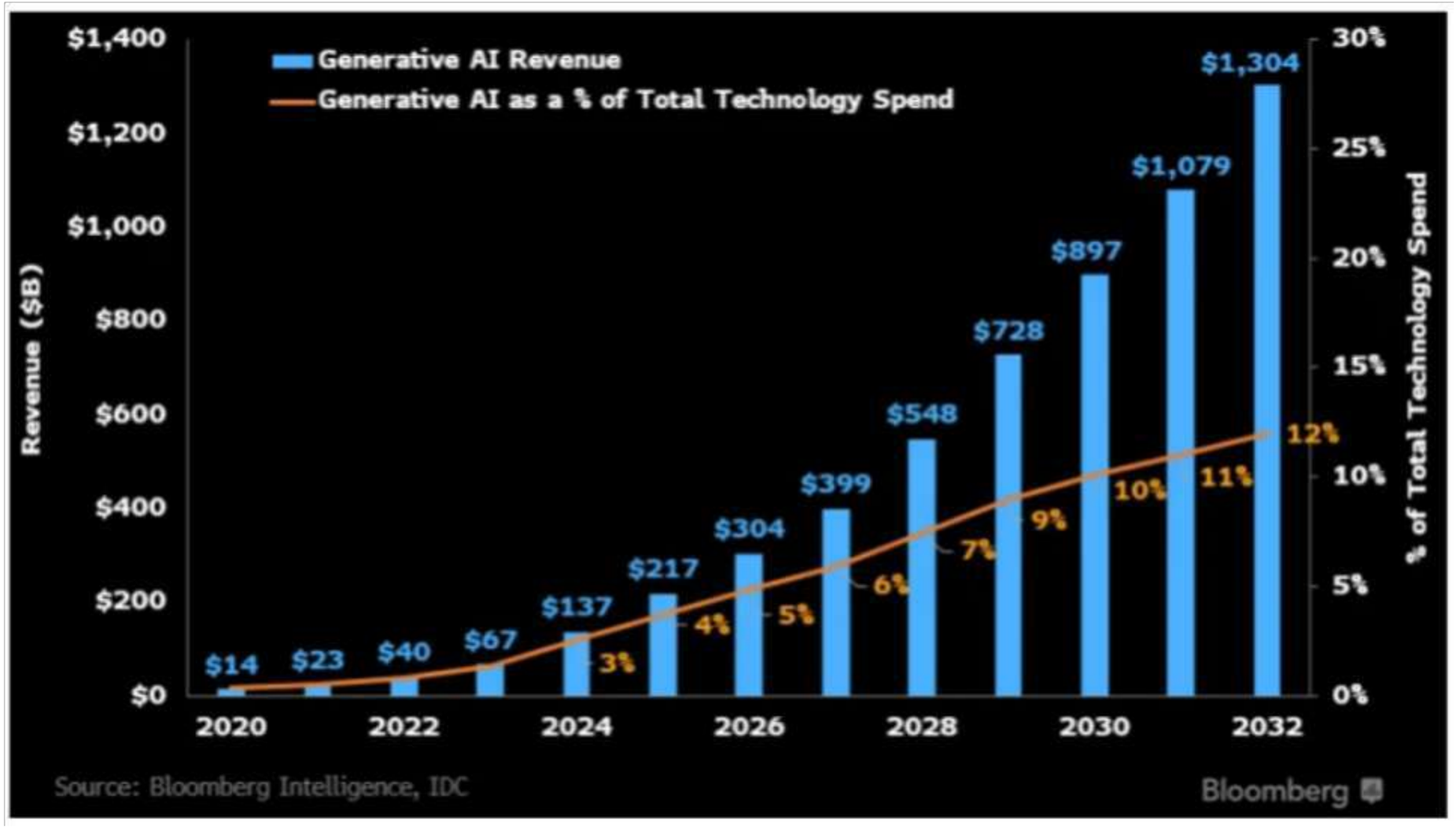


Reference: Generative AI vs. Predictive AI: Unraveling the Distinctions and Applications

14 AIGC Market Landscape

Adapted from a figure in [Generative AI's Act Two | Sequoia Capital]

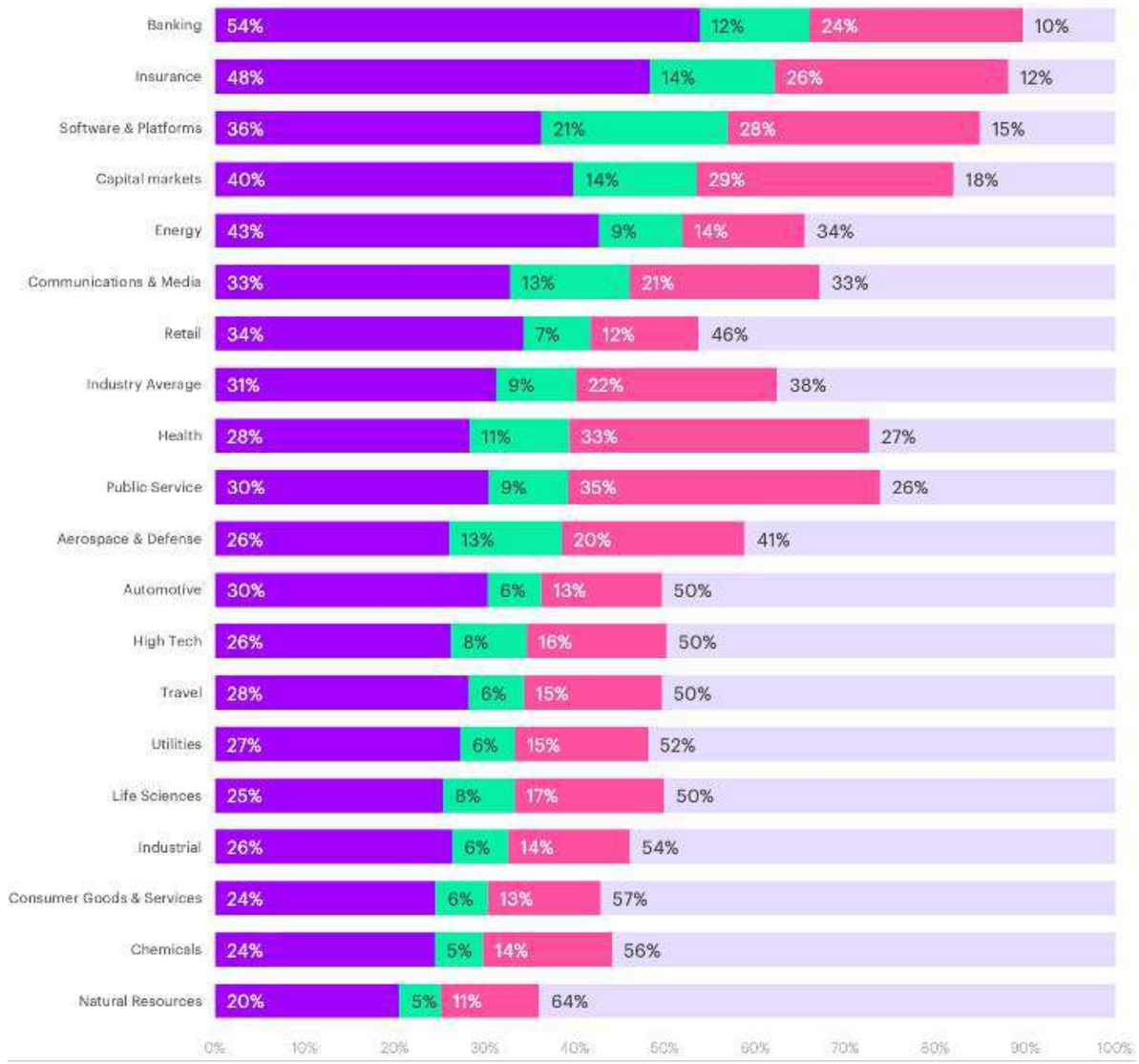
15 AIGC Market Size

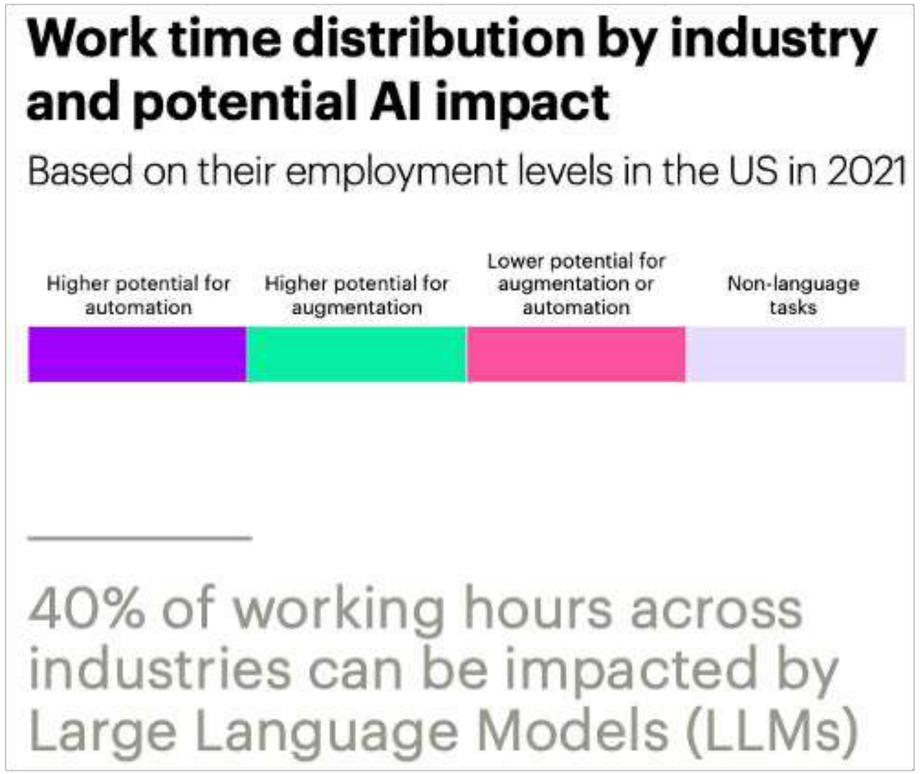


Bloomberg Intelligence predicts AIGC (Generative AI) to become a $1.3 trillion market by 2032.

Reference: Bloomberg Intelligence: Generative AI Market by 2032

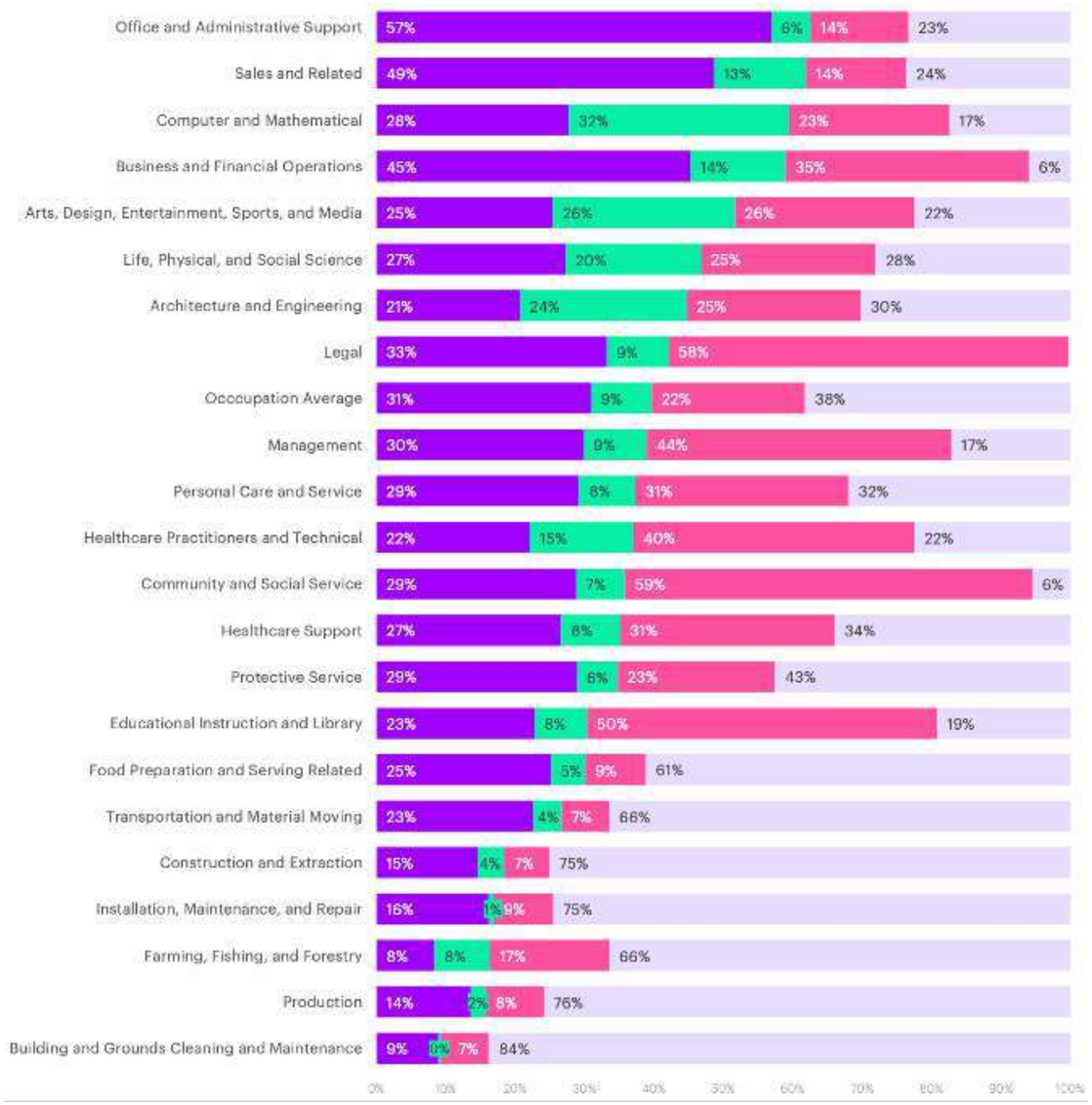
16 AIGC and the Labor Market

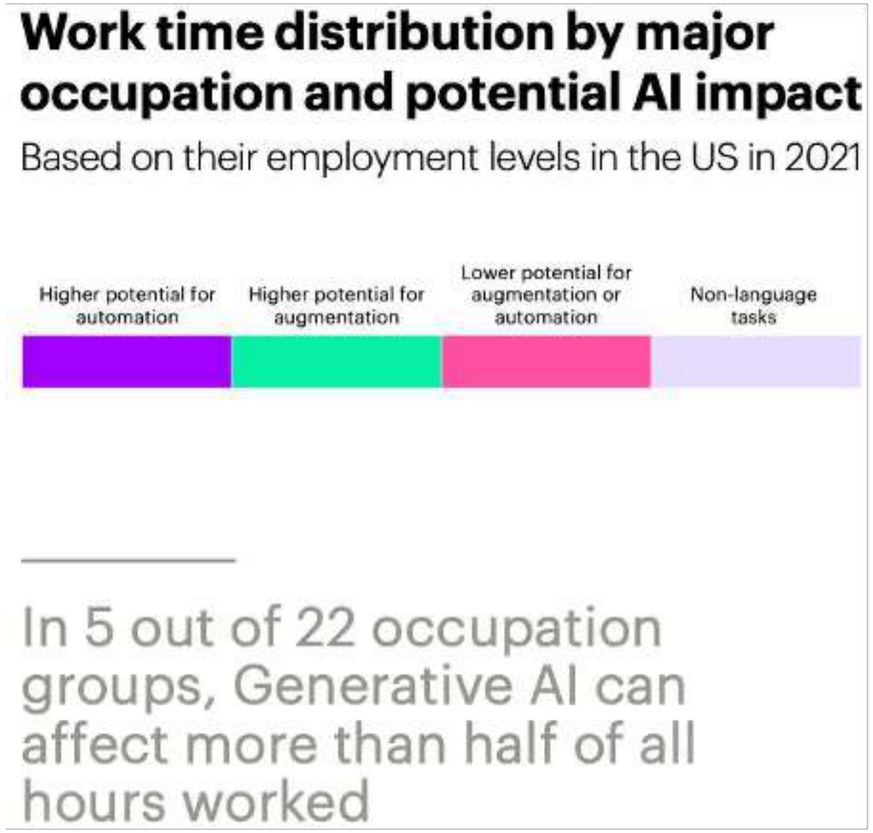




Reference: Accenture: A new era of generative AI for everyone

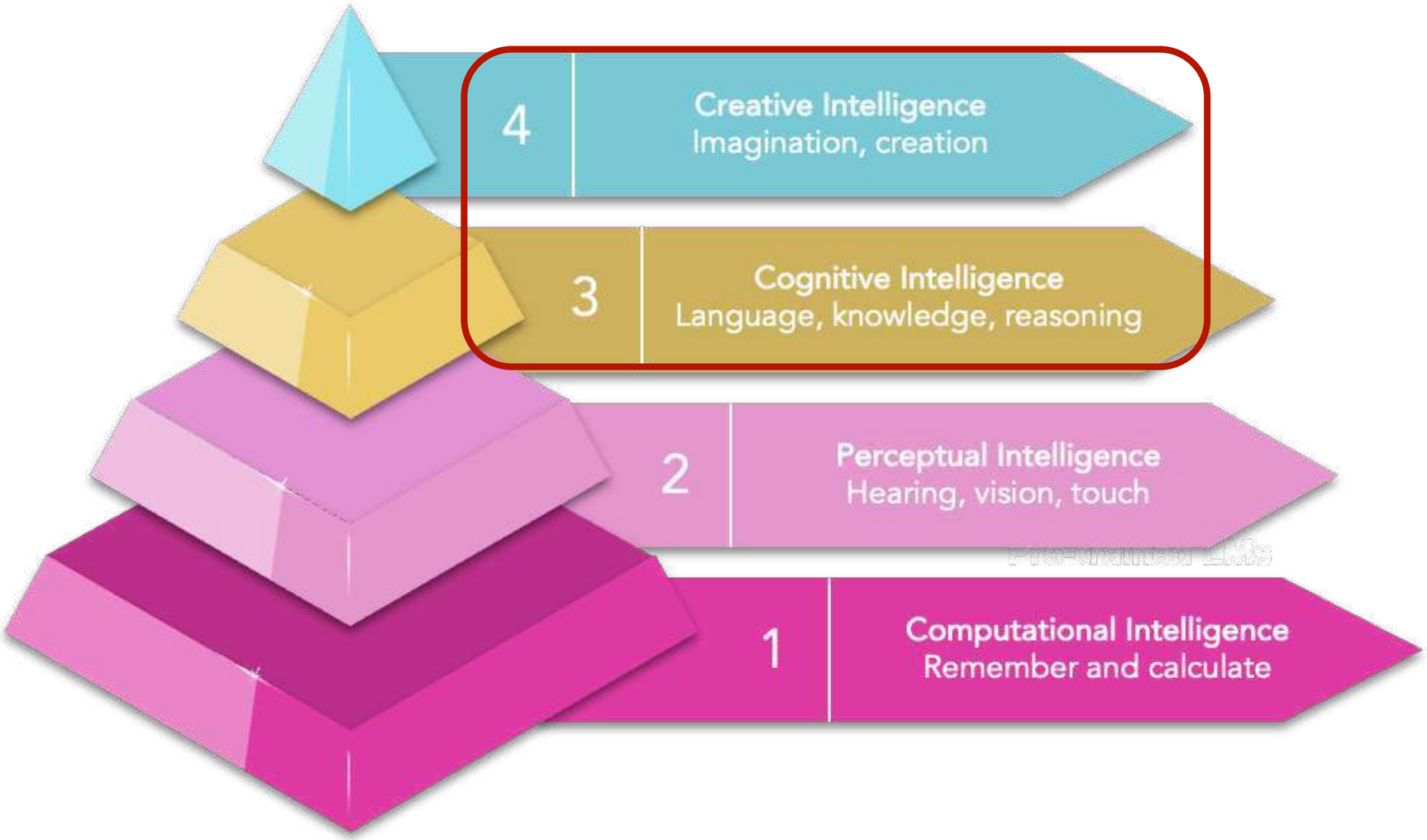
AIGC and the Labor Market (cont'd)





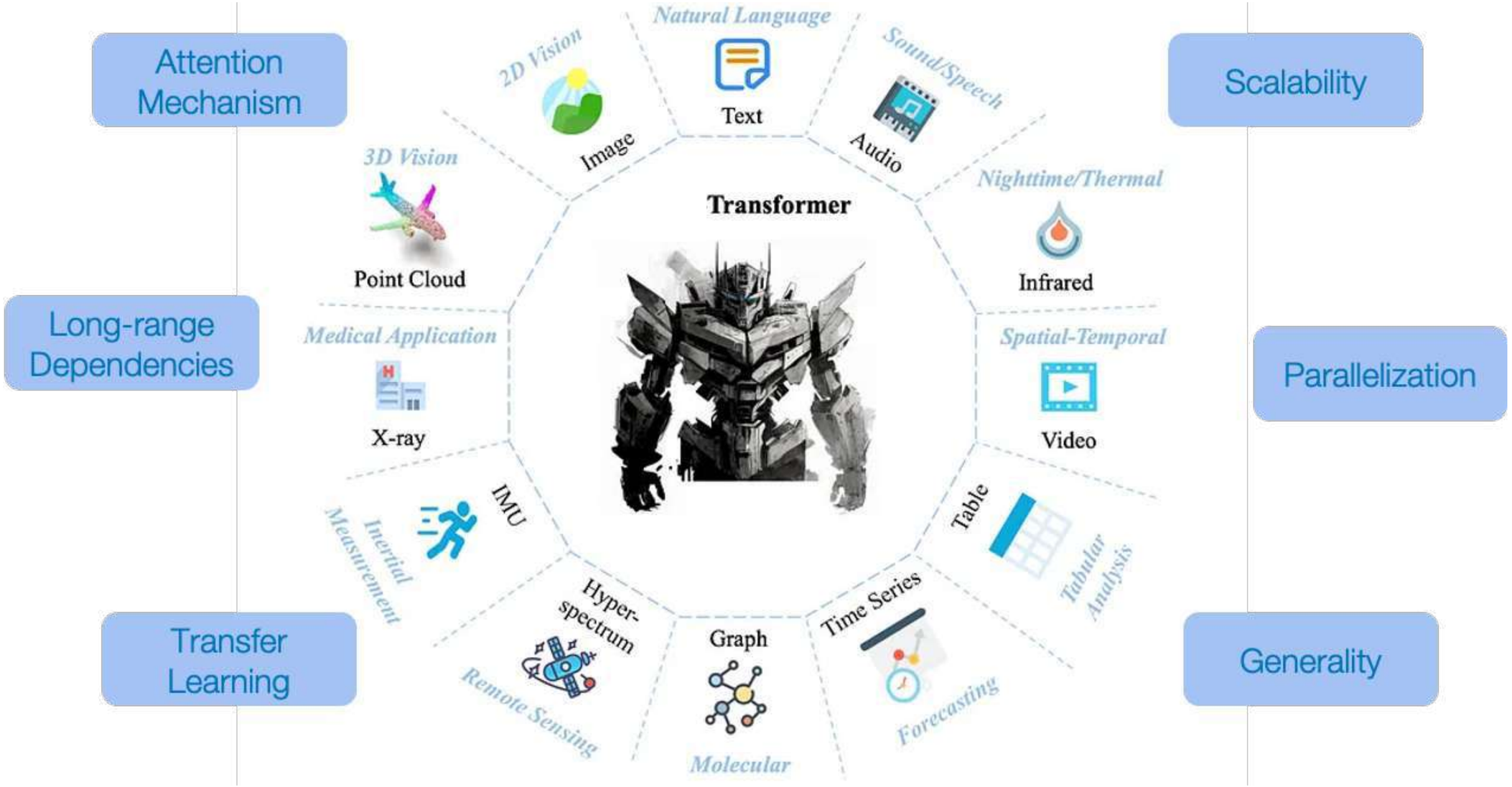
Reference: Accenture: A new era of generative AI for everyone

18 Levels of AI



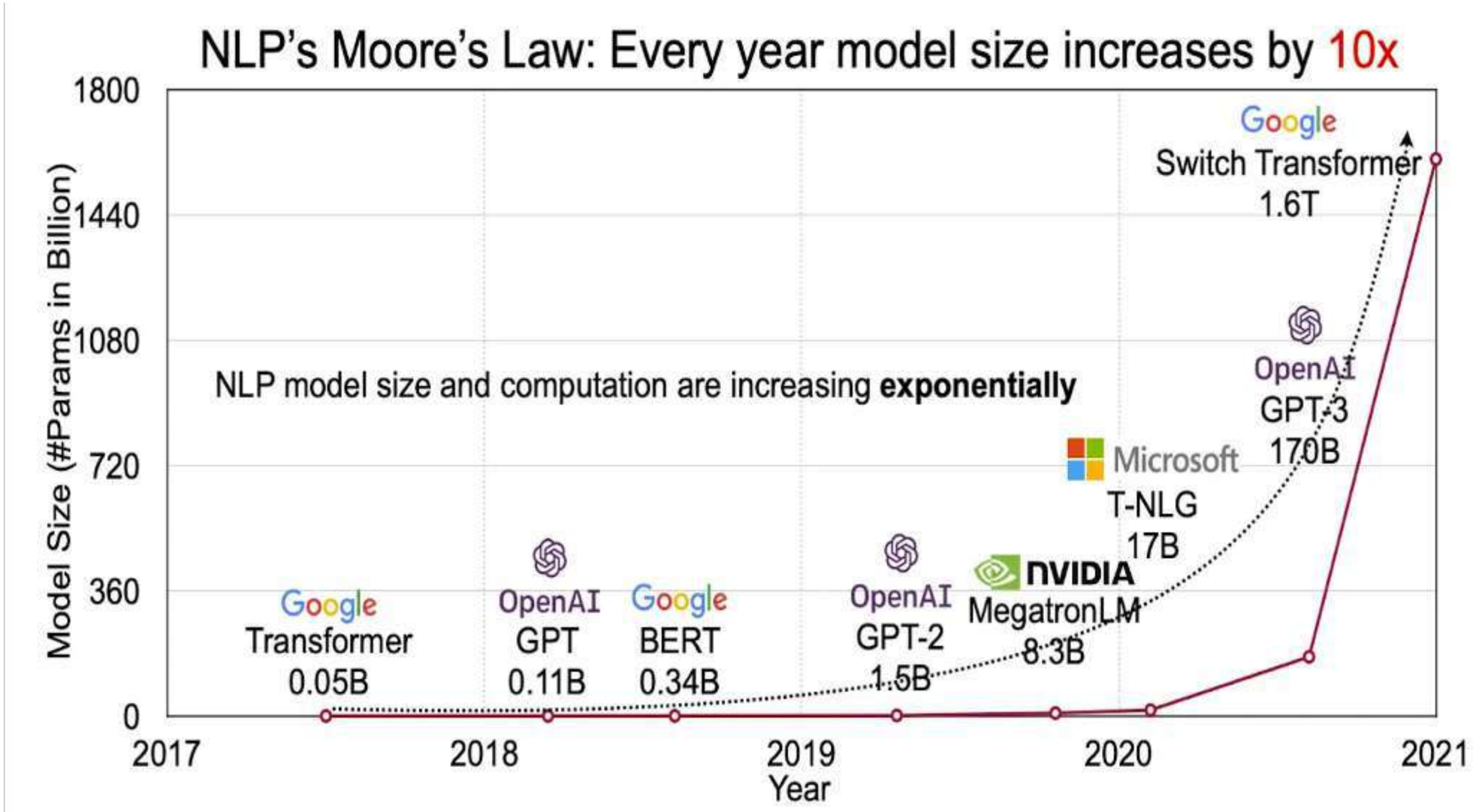
Why does this wave of Generative AI work so well?

20 Modeling Innovation



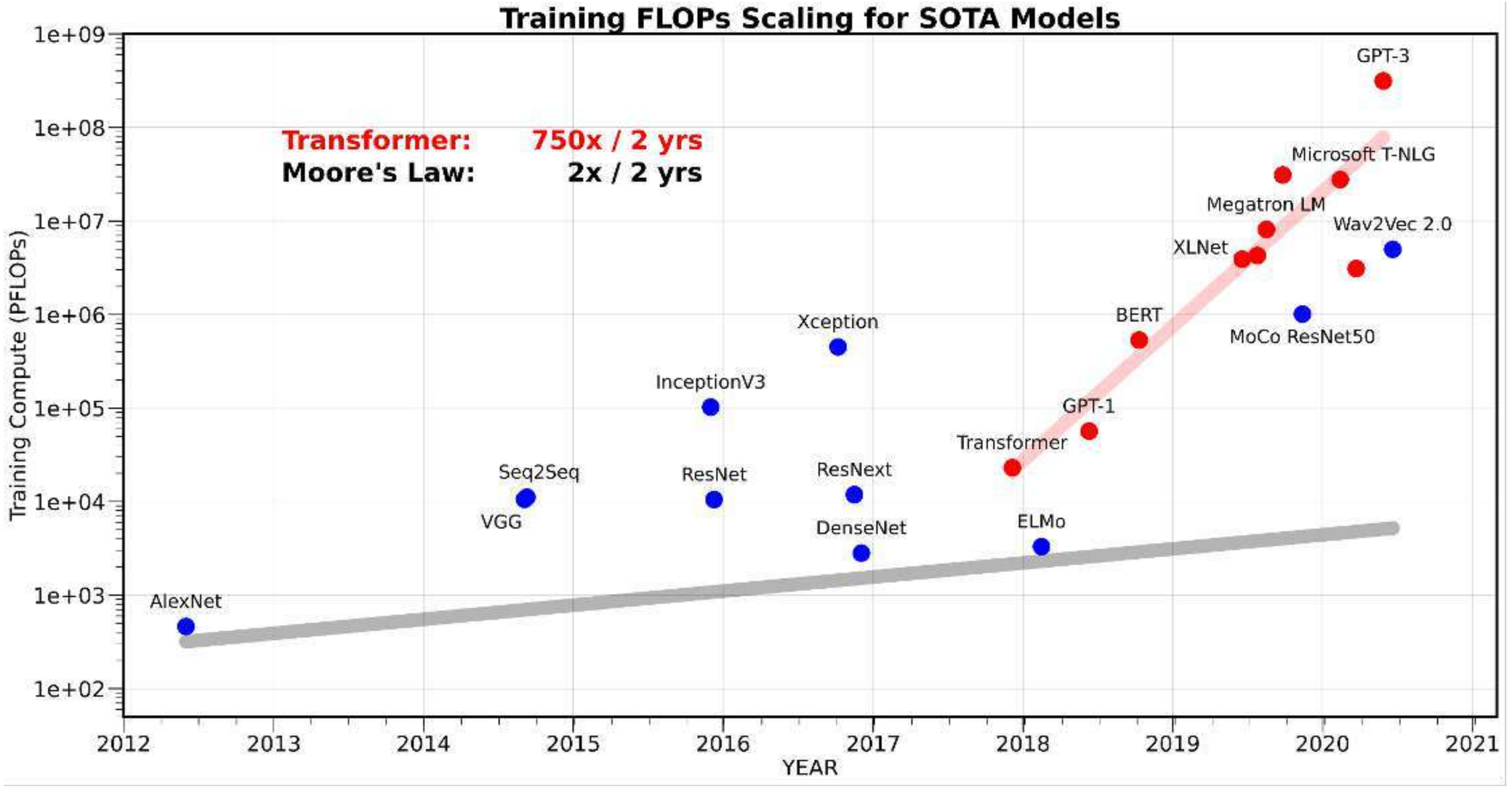
Adapted from a figure in https://arxiv.org/pdf/2307.10802.pdf

21 Scaling Up Model Size



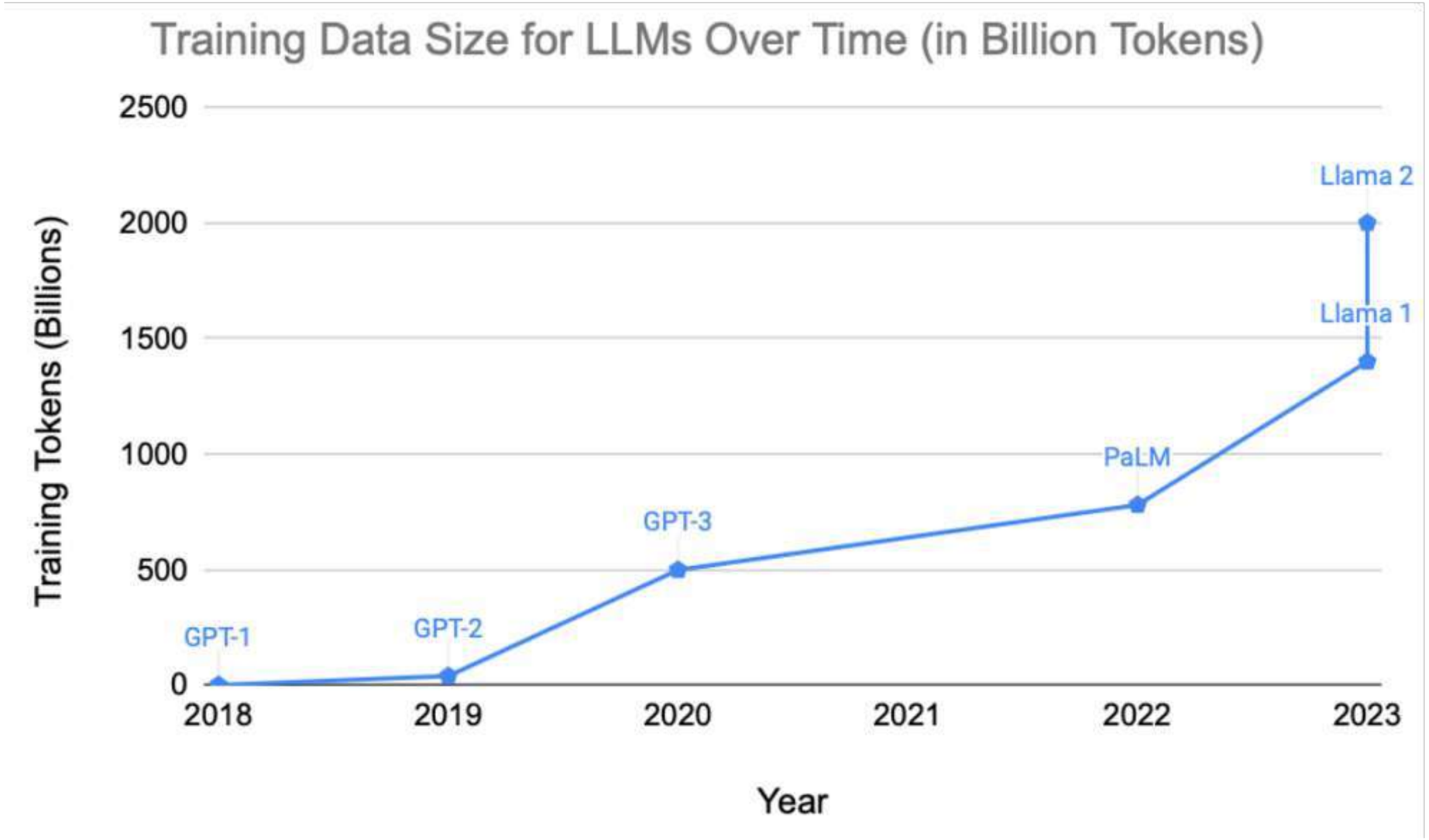
Reference: [Unveiling the Power of Large Language Models (LLMs)]

Scaling Up Computation Size

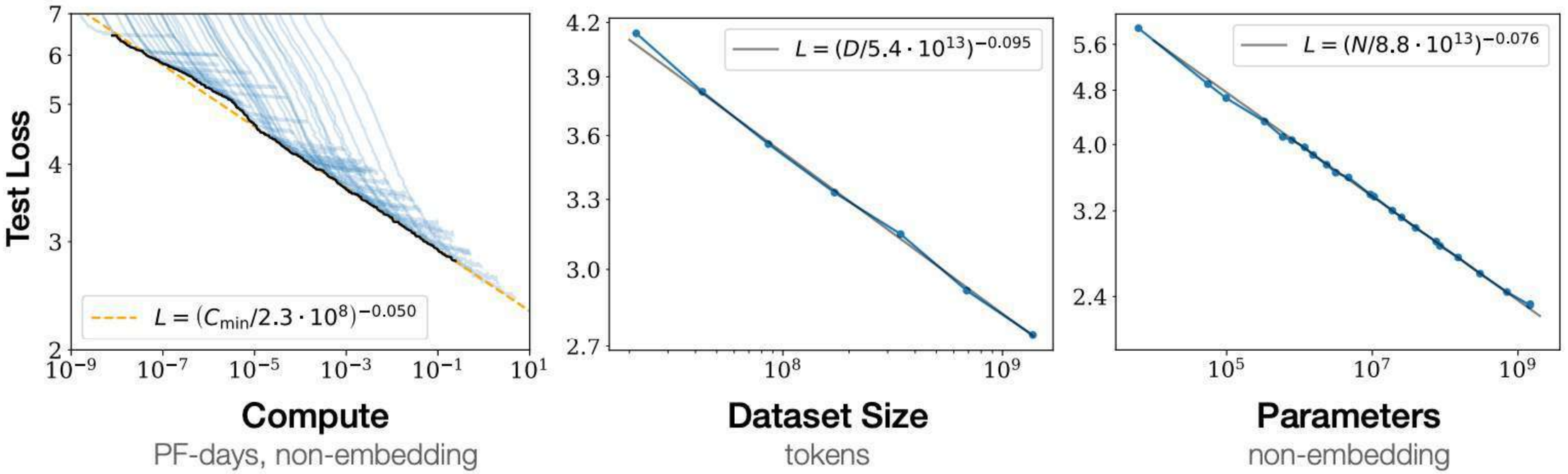


Reference: AI and Memory Wall

23 Scaling Up Data Size

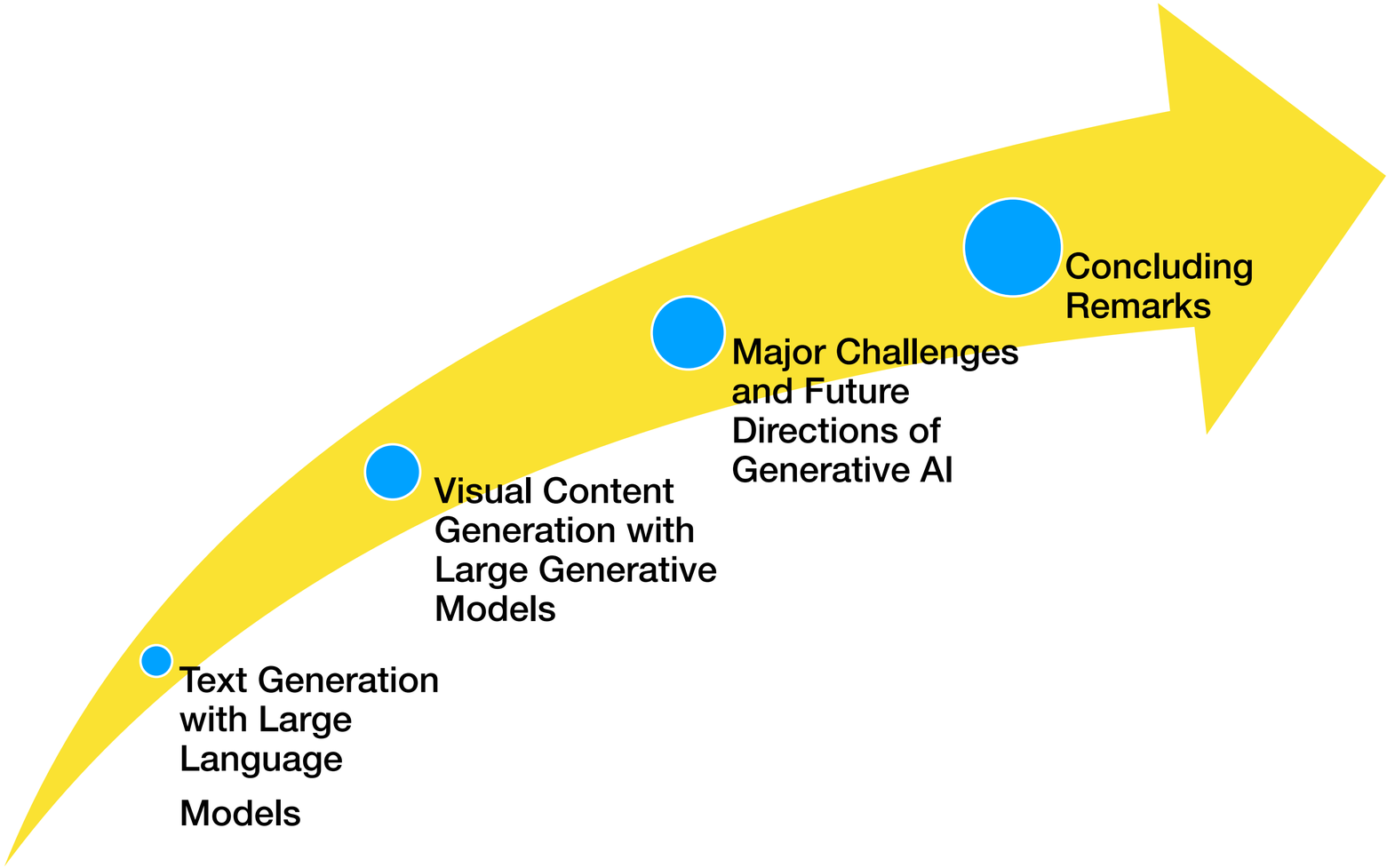


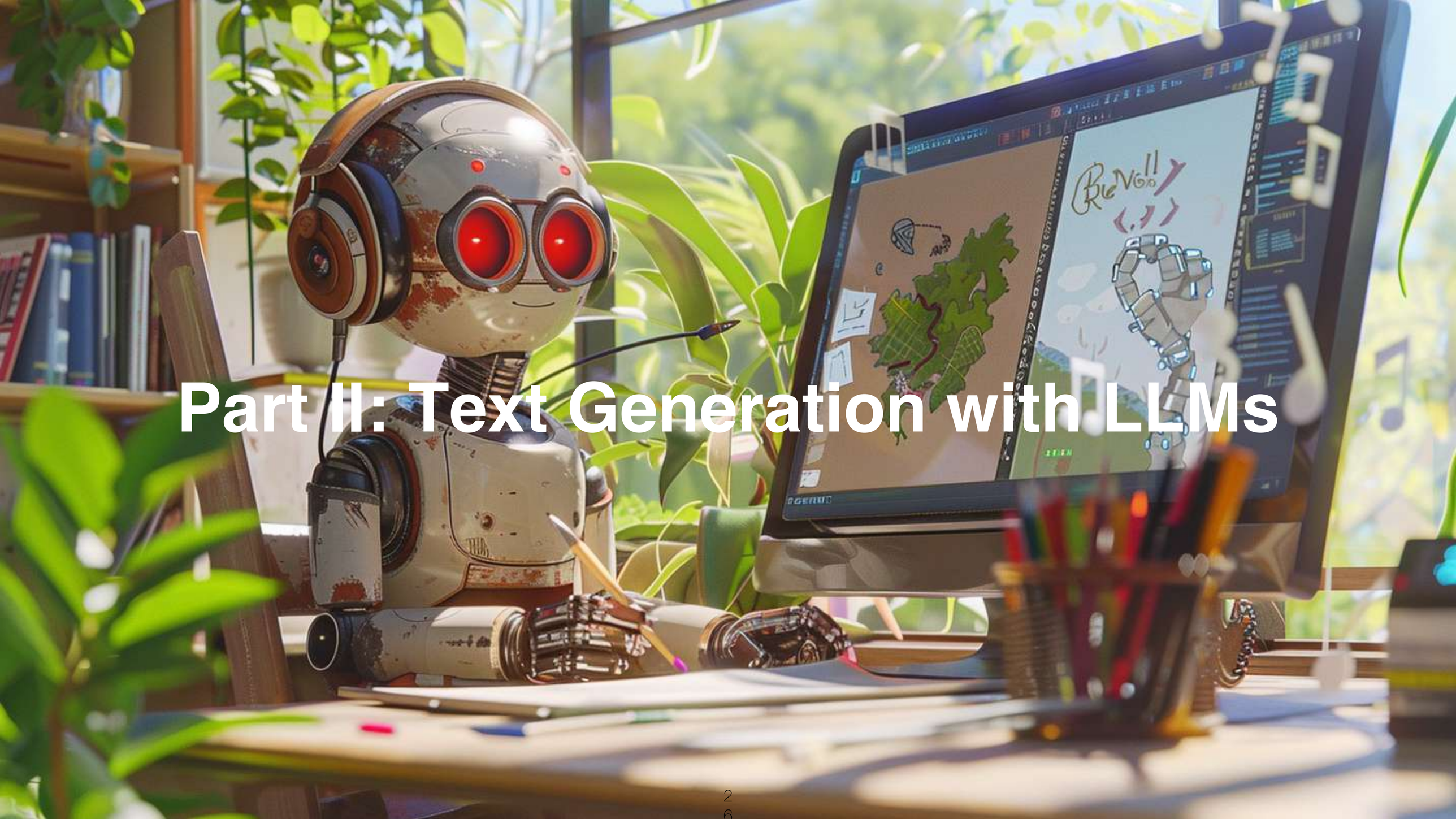
24 Scaling Laws for Large Language Models



Kaplan, Jared, et al. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).

AIGC Journey Roadmap

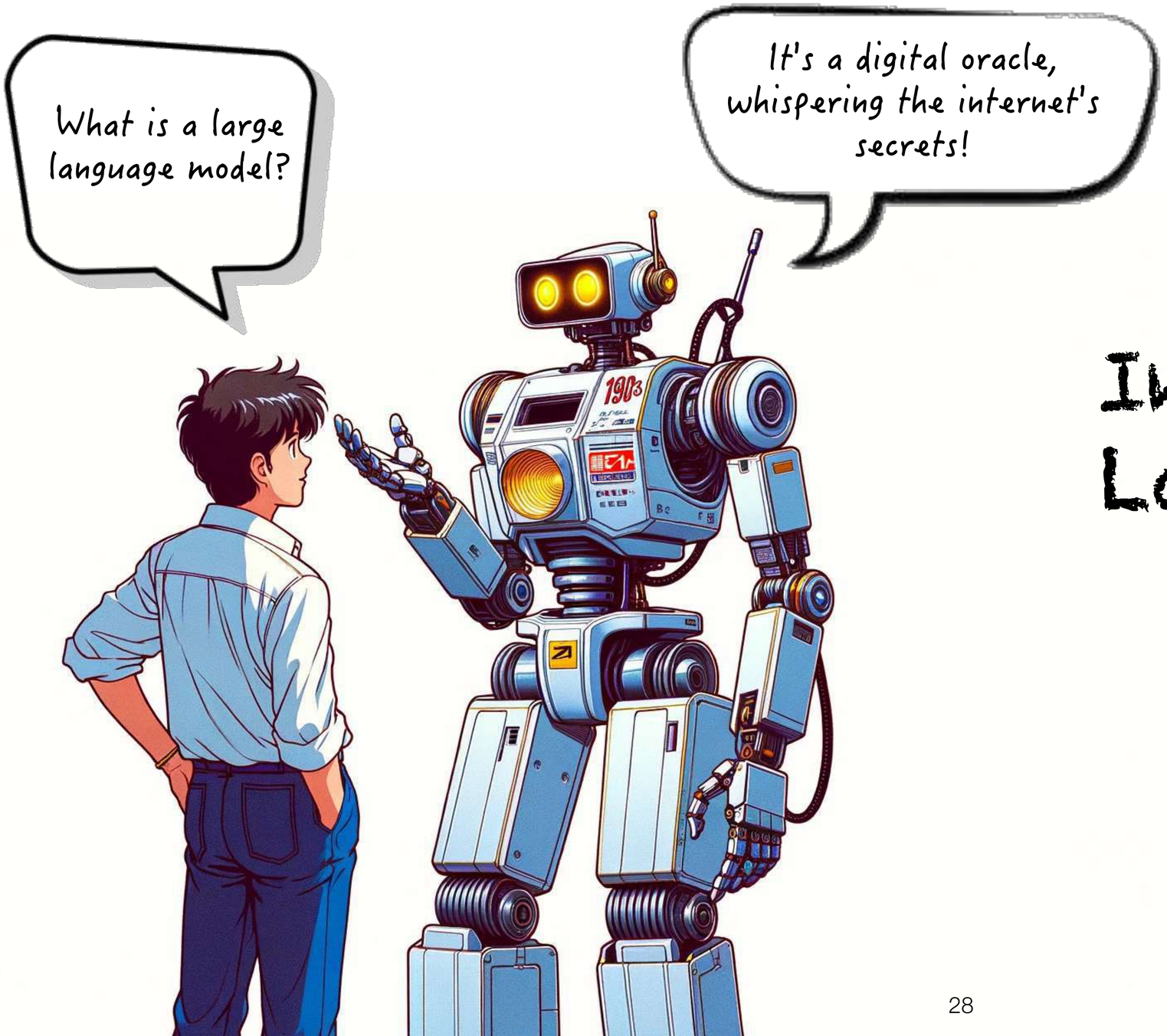




Part II: Text Generation with LLMs

Outline: Text Generation with Large Language Models

1. Introduction to Large Language Models
    1. NLP and Its Challenges
    2. Language Modeling: A Paradigm Shift
    3. Large Language Models: Large is Different
2. Core Techniques of Large Language Models
    1. Pre-training
    2. Adaptation
    3. Utilization
    4. Evaluation
    5. Application: Revolutionizing the Legal Sector with LLMs



Introduction to Large Language Models

NLP and its challenges

What is natural language processing?

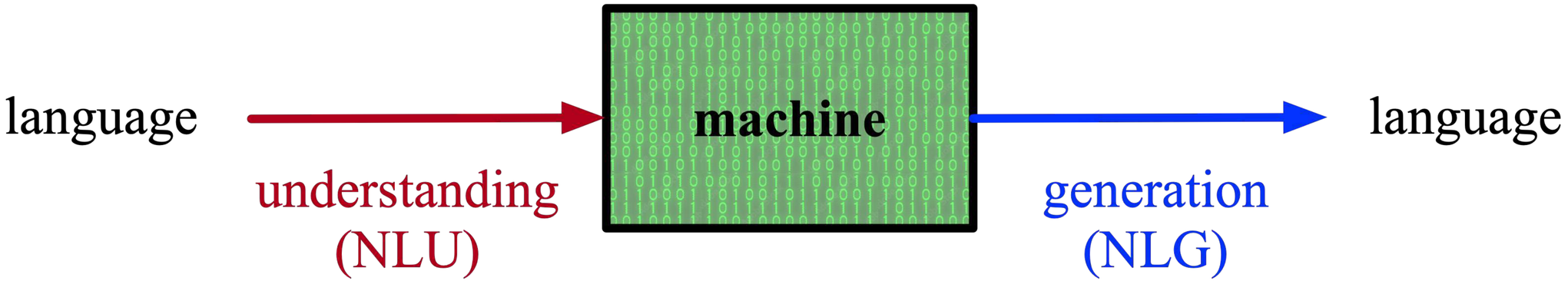
- **Language** is a tool of human communication and a carrier of human thinking.
  - More than 1,900 languages in use all over the world.
  - Structures vary between different languages.

- **Natural language** is a human language (e.g., English, French, Chinese)
  - As opposed to a constructed language (programming languages, language of formal logic, etc.).

Image from: https://www.vpr.org/post/why-are-there-so-many-different-languages#stream/0

What is natural language processing?

- **Natural language processing (NLP)** is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language.



There are a lot of NLP tasks

Lexical Analysis
Word Segmentation
Word Tokenization
New Words Identification
Morphological Analysis
Part-of-speech Tagging
Spelling Correction

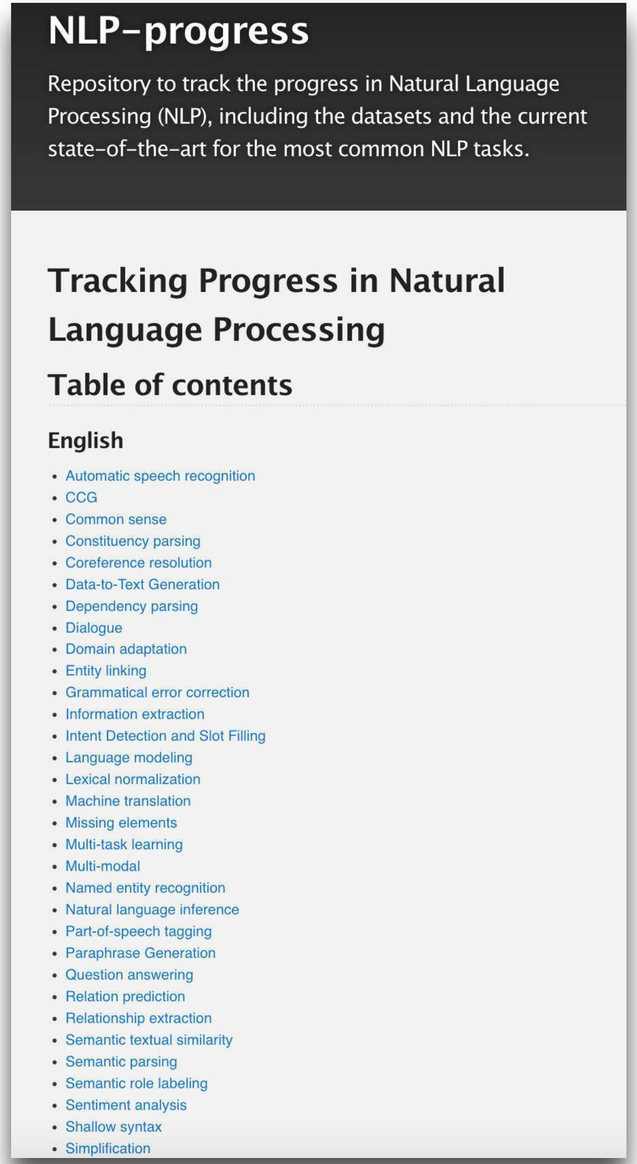
Sentence Analysis
Chunking
Super Tagging
Constituency Parsing
Dependency Parsing
Language Modeling
Language Identification

Semantic Analysis
Word Sense Disambiguation
Semantic Role Labeling
Abstract Meaning Representation Parsing
Frame Semantic Parsing
First Order Predicate Calculus
Word/Sentence/Paragraph Vector

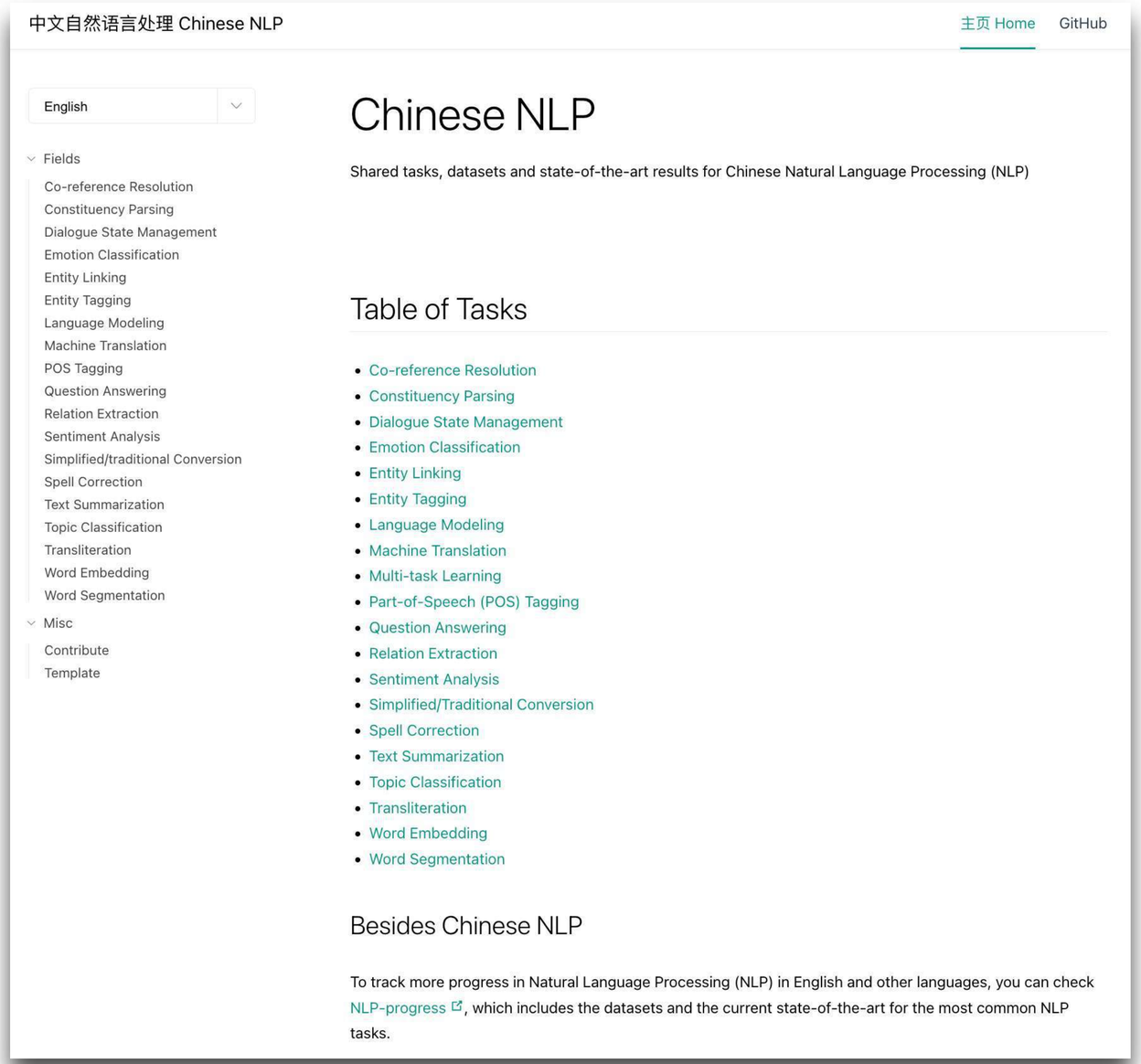
Information Extraction
Named Entity Recognition/Disambiguation
Terminology/Glossary Extraction
Coreference Resolution
Relationship Extraction
Event Extraction
Sentiment Analysis

High-level Tasks
Machine Translation
Text Summarization/Simplification
Question Answering
Dialogue System
Reading Comprehension
Automatic Essay Grading

33 More tasks can be found here

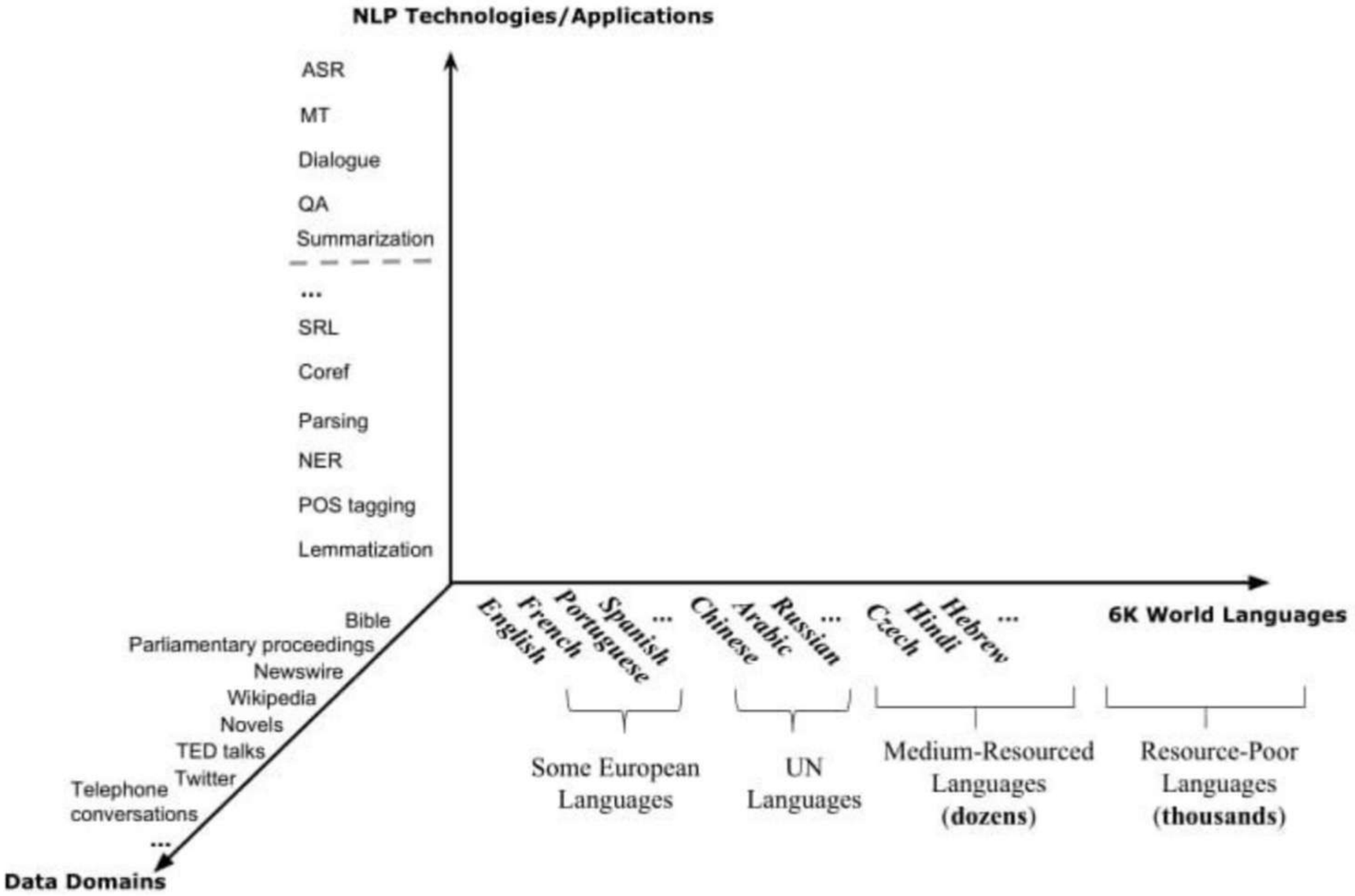


https://nlpprogress.com/



https://chinesenlp.xyz

34 More complexity dimensions



Labeling data becomes a business

High-Quality Training Data to Accelerate Artificial Intelligence and Machine Learning

Our expert team delivers accurate and reliable data labeling services across all data types to help accelerate machine learning development.

Image & Video

From faces to places, power your computer vision models with high-quality image data.

- Self Driving Cars
- Robotics
- Facial Recognition
- Anomaly Detection
- Product Identification
- Object Detection

Text

Train your
annotated
complex do

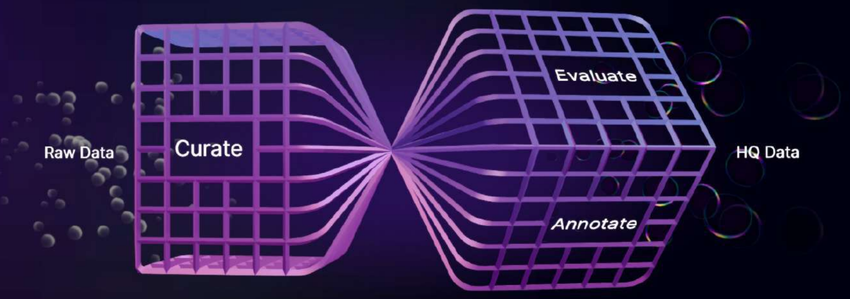
- Sentiment
- Search Re
- Product C
- Entity & R
- Natural La
- Content S
- Multilingu

scale

Data Engine

Collect, curate, and annotate data.
Train models and evaluate. Repeat.

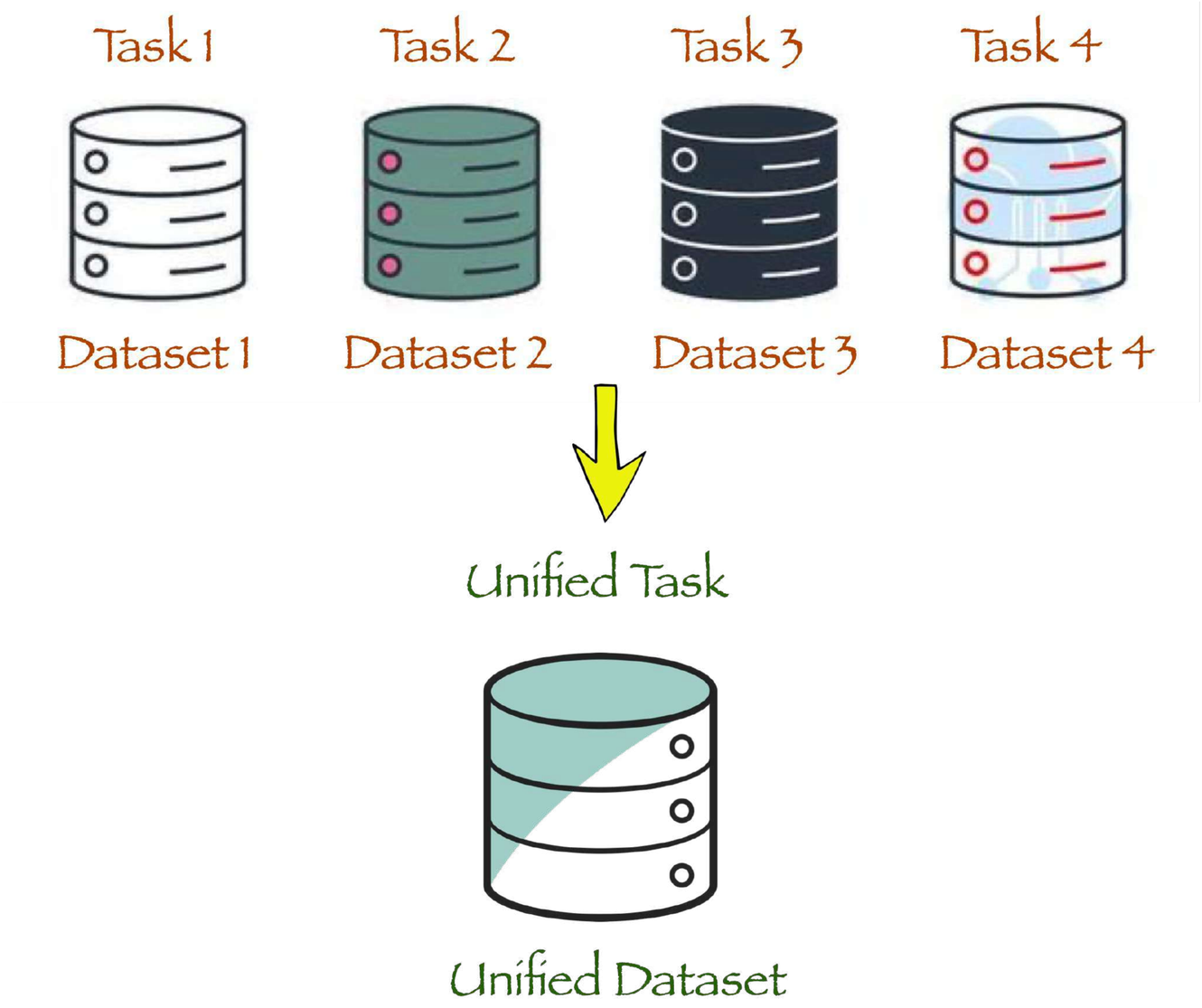
Book a demo →



OpenAI · Meta · Microsoft · TOYOTA RESEARCH INSTITUTE · gm

Adept · CarperAI · cohere · Etsy · instacart

Can we unify different tasks?



Can we use unlabeled data?

Labeled Datasets

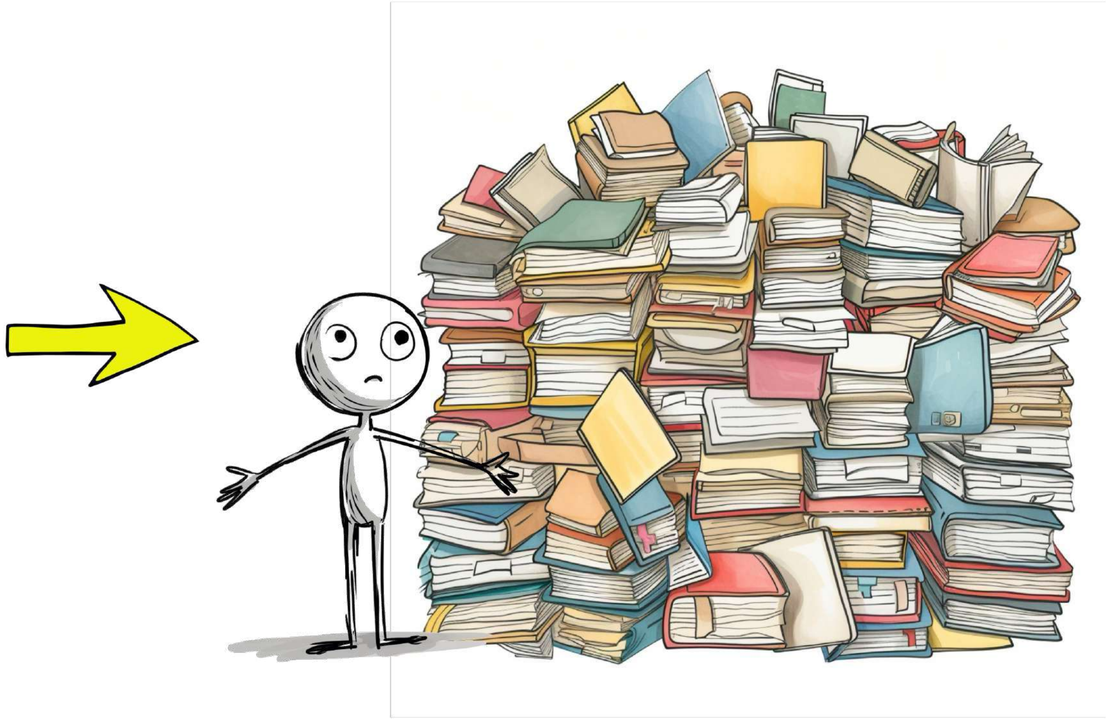
"Creat service for an affordable price"

Label: Positive

"Horrible service. The room was dirty!"

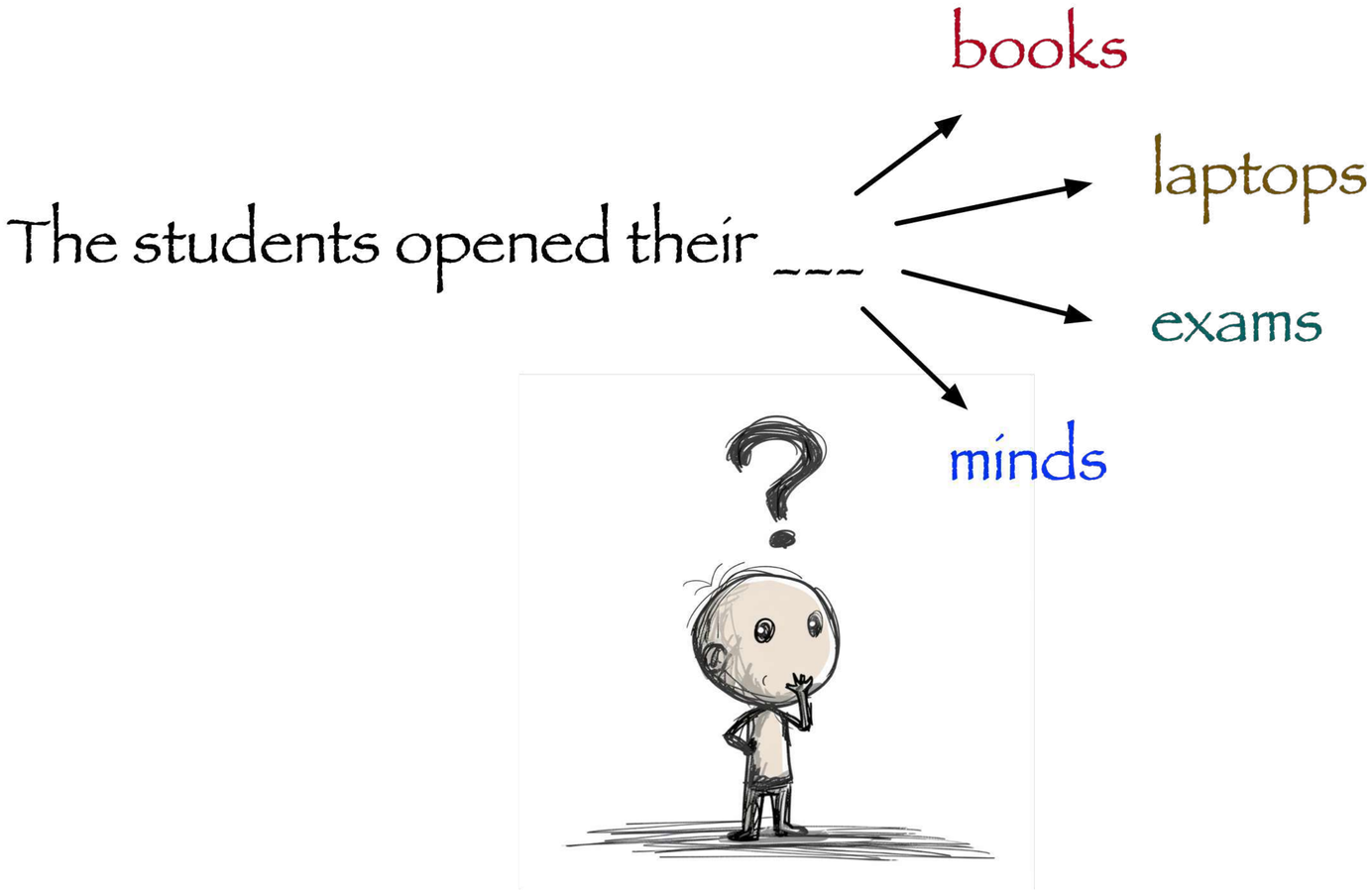
Label: Negative

Unlabeled Data



Language Modeling: A Paradigm Shift

39 What is language modeling?



Causal Language Modeling (CLM)

$$L_{CLM}^{(x)} = -\frac{1}{|x|}\sum_{i=1}^{|x|} logP(x_i/x_{<i})$$

where:

$x = \{x_1, x_2, x_3, \ldots, x_{|x|}\}$ represents a sequence

$x_{<i} = x_1, x_2, x_3, ..x_{i-1}$

CLM: a Unidirectional Language Model that predicts the next word given the context.

Different LLMs

Existing large language models (having a size larger than 10B) in recent years.

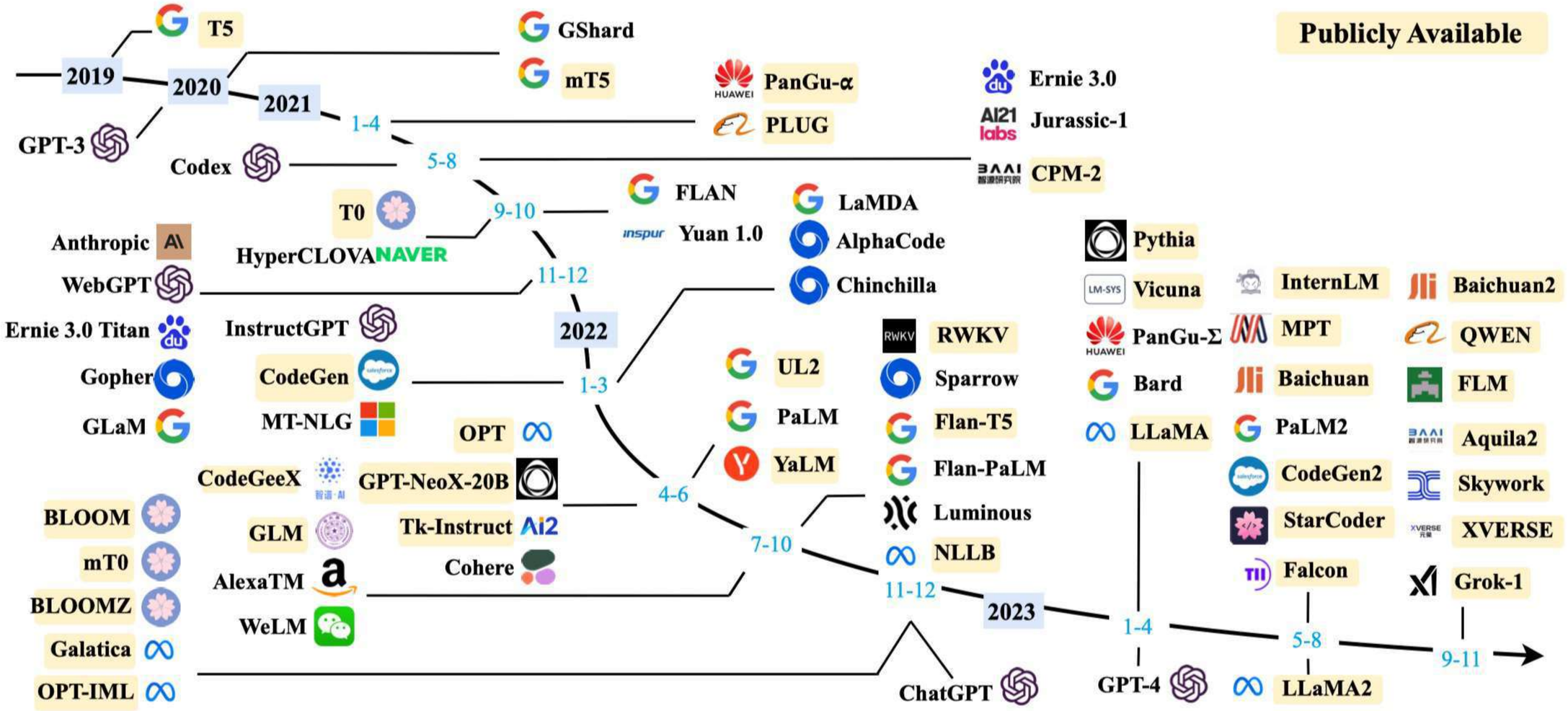


Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

The GPT family

The technical evolution of GPT-series models.

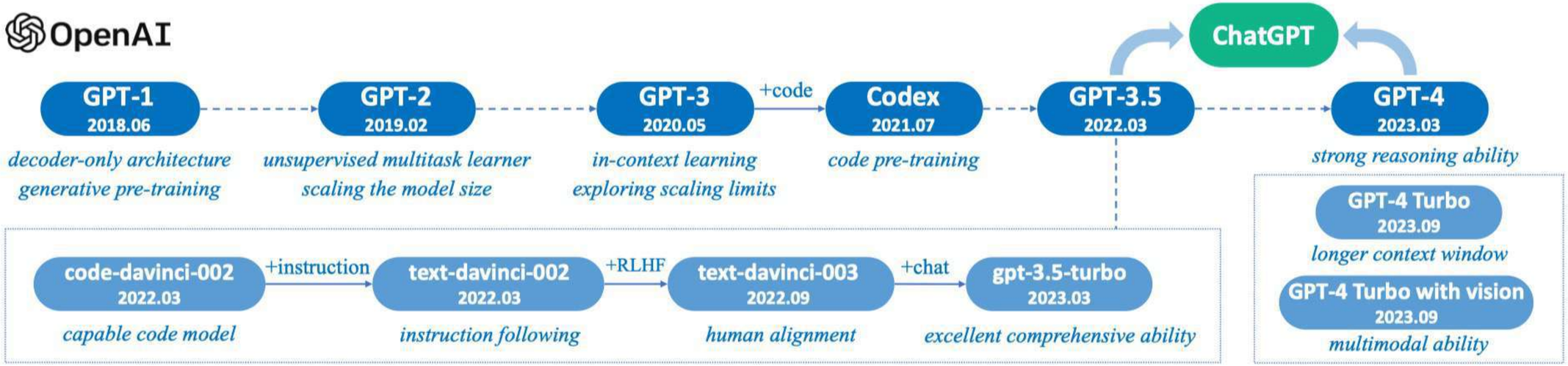
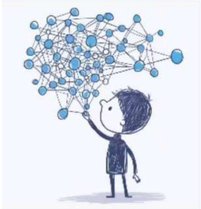


Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

LLM resources

A vibrant open source ecosystem is a key driver of AI development

Model Checkpoints or APIs



BLOOM

PanGu-Alpha

Gemini

ChatGLM

Claude.AI

Corpora and Datasets

LLMDataHub

Awesome Datasets for LLM Training

YALL - Yet Another LLM Leaderboard

LMSYS Chatbot Arena Leaderboard

Open LLM Leaderboard

Libraries for Training and Deployment

Transformers

LLM

LangChain

OpenLLM

deepspeed

LMSYS FastChat

FairScale

gpt-neox

Colossal-AI

NVIDIA Megatron-LM

FastMoE

BMTrain

More:

https://github.com/Hannibal046/Awesome-LLM?tab=readme-ov-file

https://github.com/Zjh-819/LLMDataHub

https://github.com/eugeneyan/open-llms

Research work conducted on LLaMA

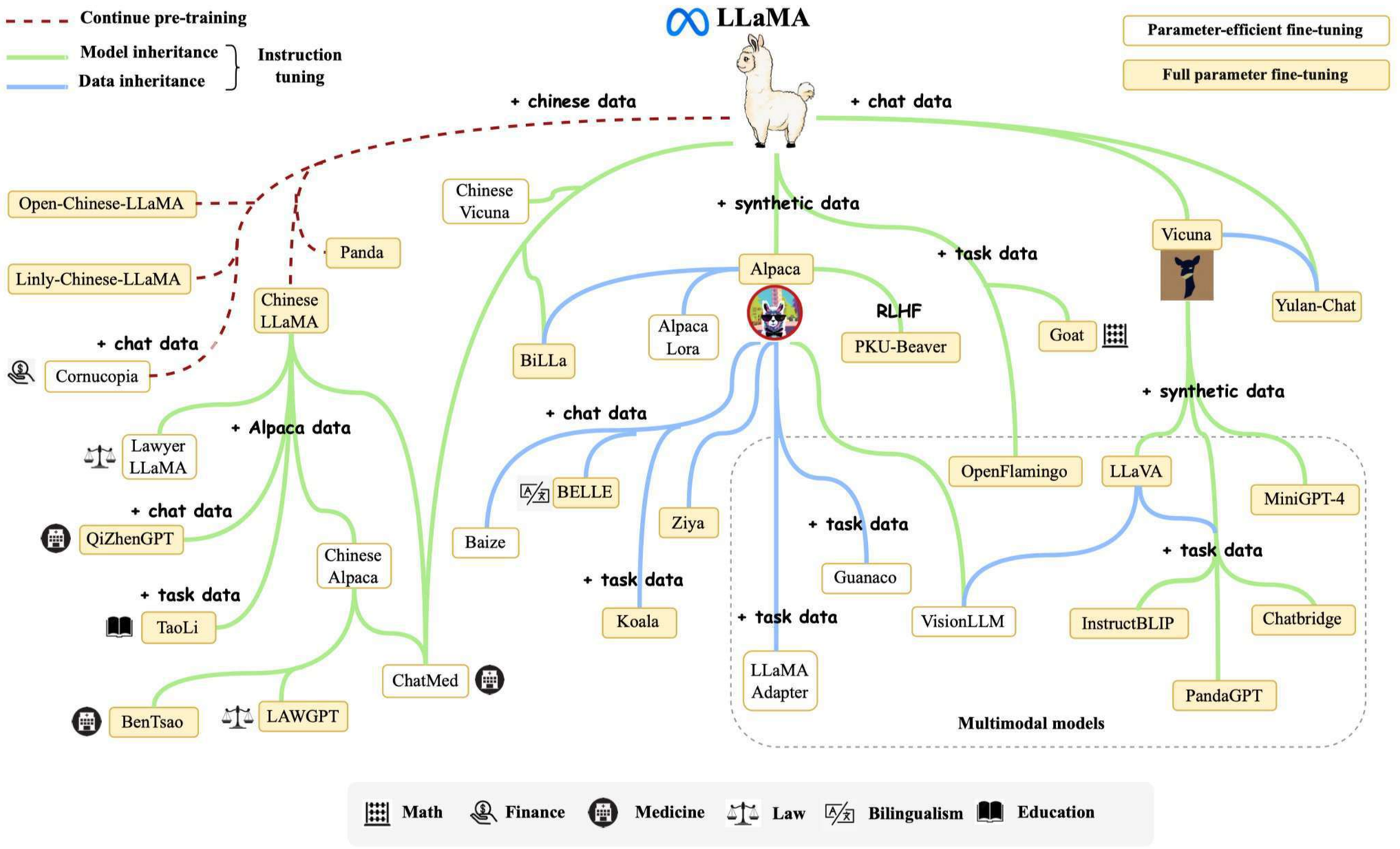


Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Open Language Model: OLMo

OLMo and framework includes:

- **Full pretraining data:** The model is built on AI2's Dolma dataset which features three trillion token open corpus for language model pretraining, including code that produces the training data.
- **Training code and model weights:** The OLMo framework includes full model weights for four model variants at the 7B scale, each trained to at least 2T tokens. Inference code, training metrics and training logs are all provided.
- **Evaluation:** We've released the evaluation suite used in development, complete with 500+ checkpoints per model, from every 1000 steps during the training process and evaluation code under the umbrella of the Catwalk project.

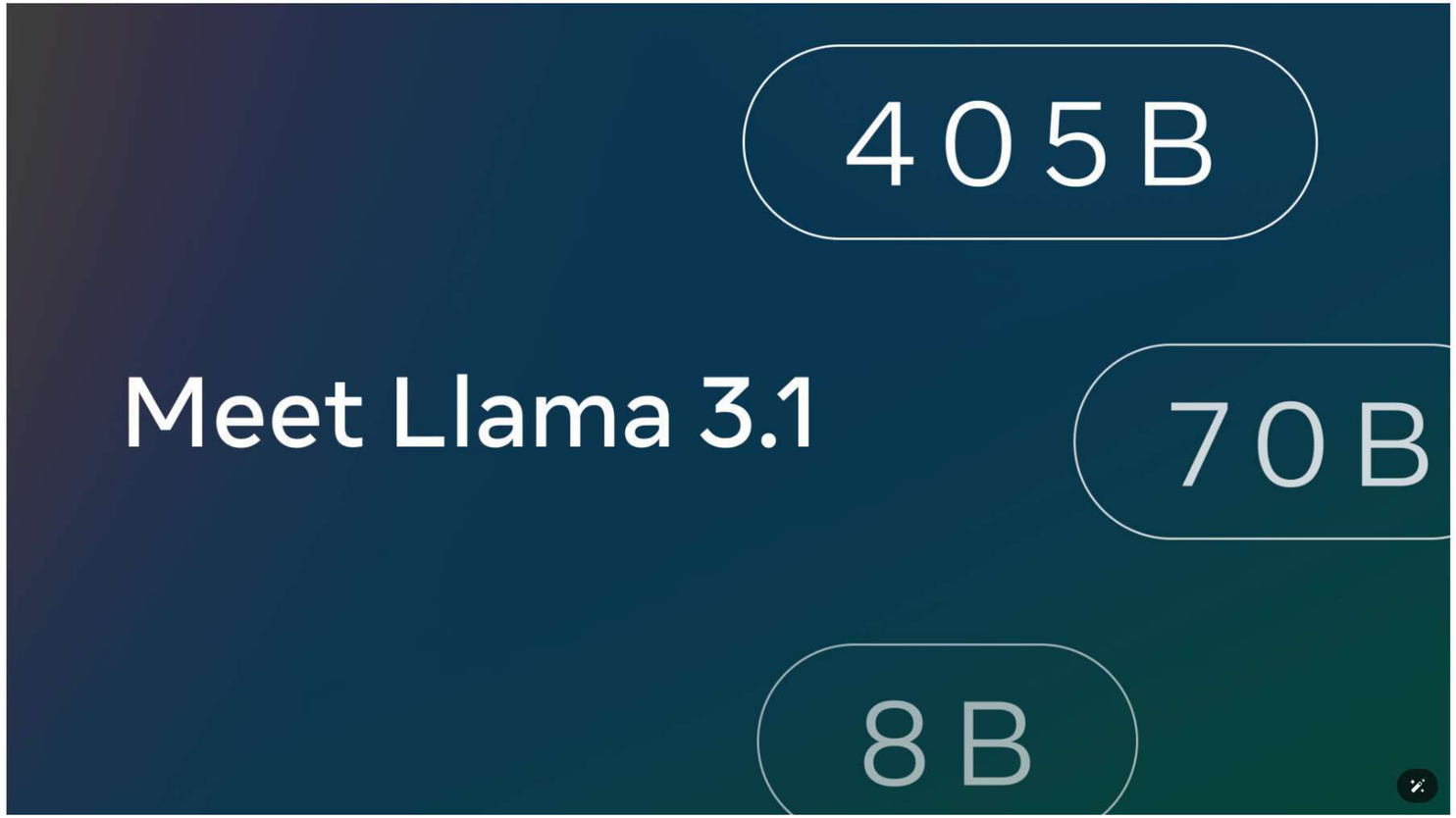
Reference: https://allenai.org/olmo

Recently Released LLMs

Large Language Model

Introducing Llama 3.1: Our most capable models to date

July 23, 2024 • 15 minute read



https://ai.meta.com/blog/meta-llama-3-1/

47 Recently Released LLMs

July 24, 2024 Mistral AI team

This latest generation continues to push the boundaries of cost efficiency, speed, and performance. Mistral Large 2 is exposed on la Plateforme and enriched with new features to facilitate building innovative AI applications.

Mistral Large 2

Mistral Large 2 has a 128k context window and supports dozens of languages including French, German, Spanish, Italian, Portuguese, Arabic, Hindi, Russian, Chinese, Japanese, and Korean, along with 80+ coding languages including Python, Java, C, C++, JavaScript, and Bash.

Mistral Large 2 is designed for single-node inference with long-context applications in mind – its size of 123 billion parameters allows it to run at large throughput on a single node. We are releasing Mistral Large 2 under the Mistral Research License, that allows usage and modification for research and non-commercial usages. For commercial usage of Mistral Large 2 requiring self-deployment, a Mistral Commercial License must be acquired by contacting us.

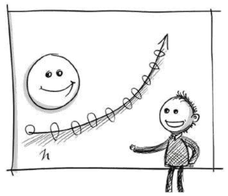
https://mistral.ai/news/mistral-large-2407/

48 The evolution of language models

Statistical language modelling suffers from the curse of dimensionality. Neural language modelling use distributed representation of words.

Pre-trained language models learn from large amounts of unlabelled corpus. "pre-training and fine-tuning" learning paradigm becomes popular.

Researchers find that scaling PLM (e.g., scaling model size or data size) often leads to an improved model capacity on downstream tasks: they follow the scaling law!



The task scope that can be solved by large language models have been greatly extended, and the task performance have been significantly enhanced.

LLMs display some surprising emergent abilities that may not be observed in previous smaller PLMs. These abilities make AI algorithms unprecedently powerful and effective.

Different stages of language modeling

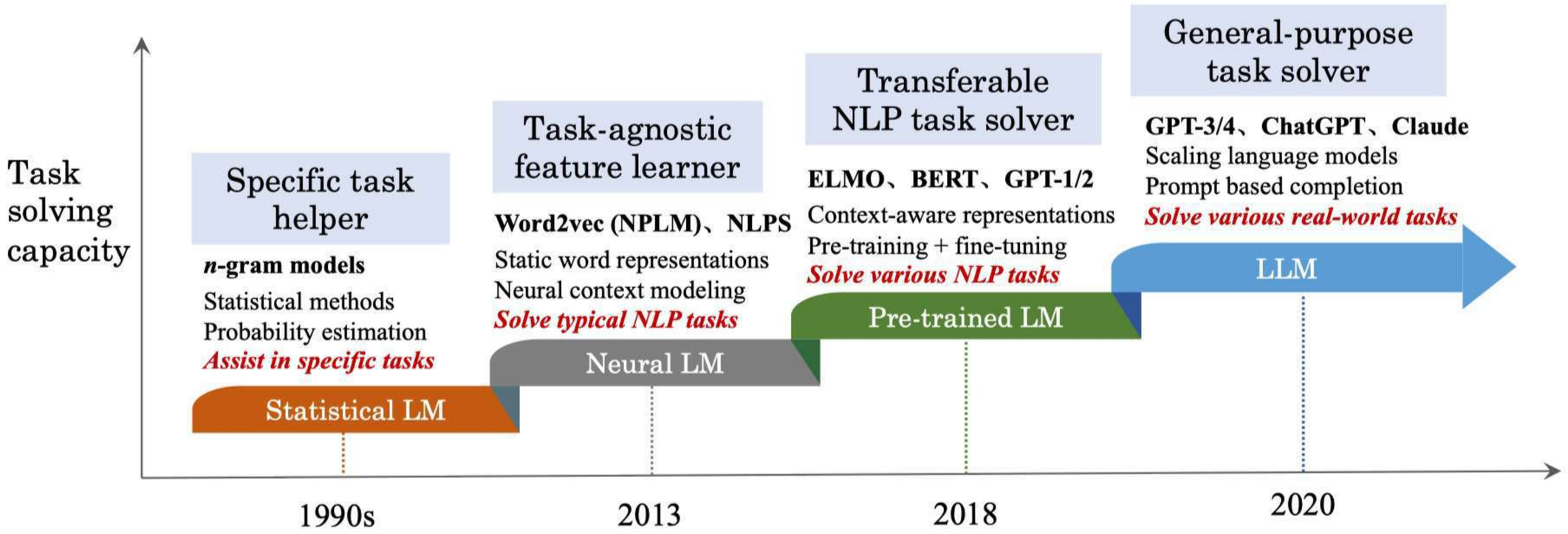
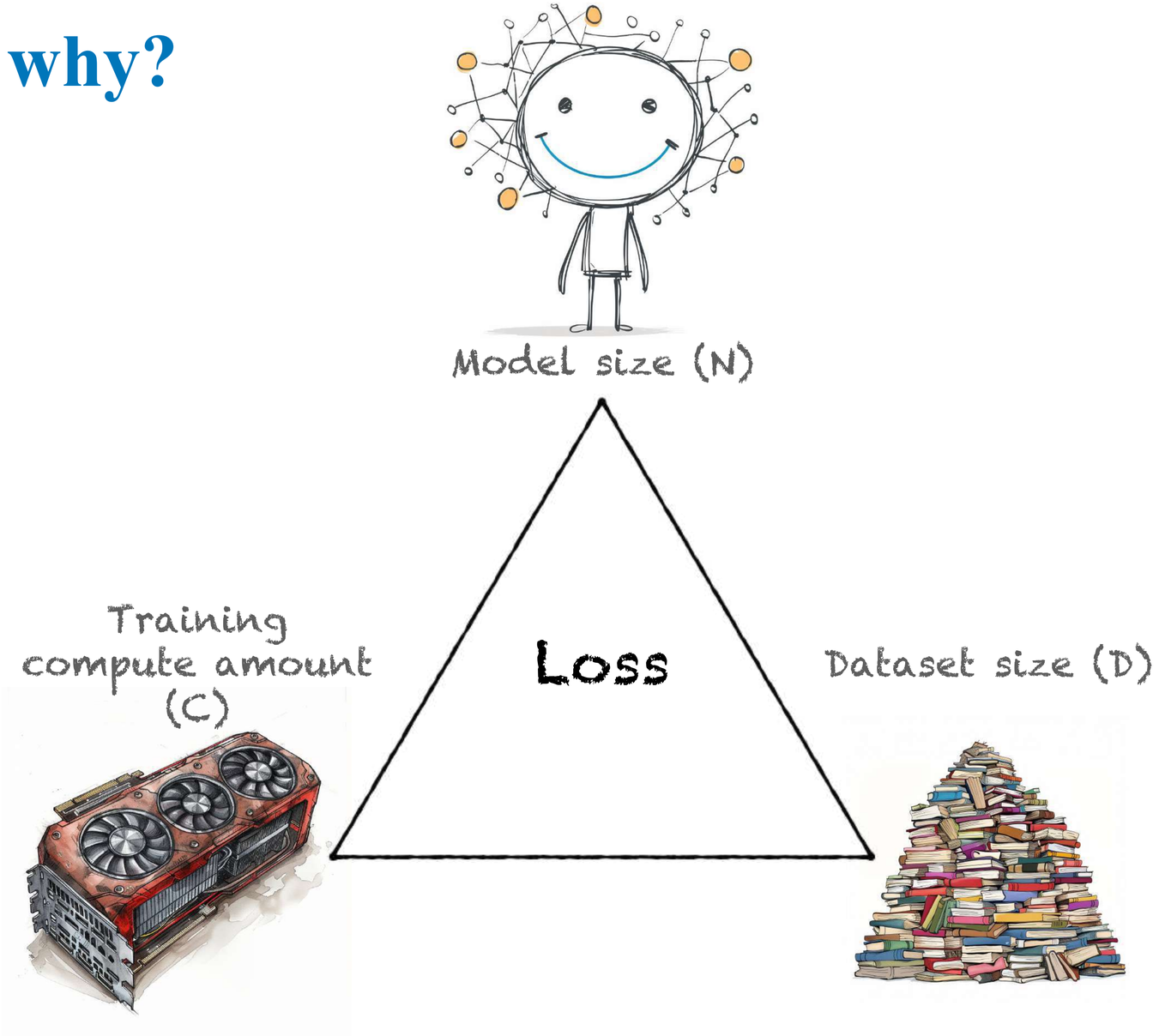


Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Large Language Models: Large is Different

Scaling law: what and why?

- Scaling up data and model size significantly improves the model capacity of LLMs
- **But hyper-parameter tuning is a huge cost!**
- We need to establish **a quantitative approach to characterizing the scaling effect**.



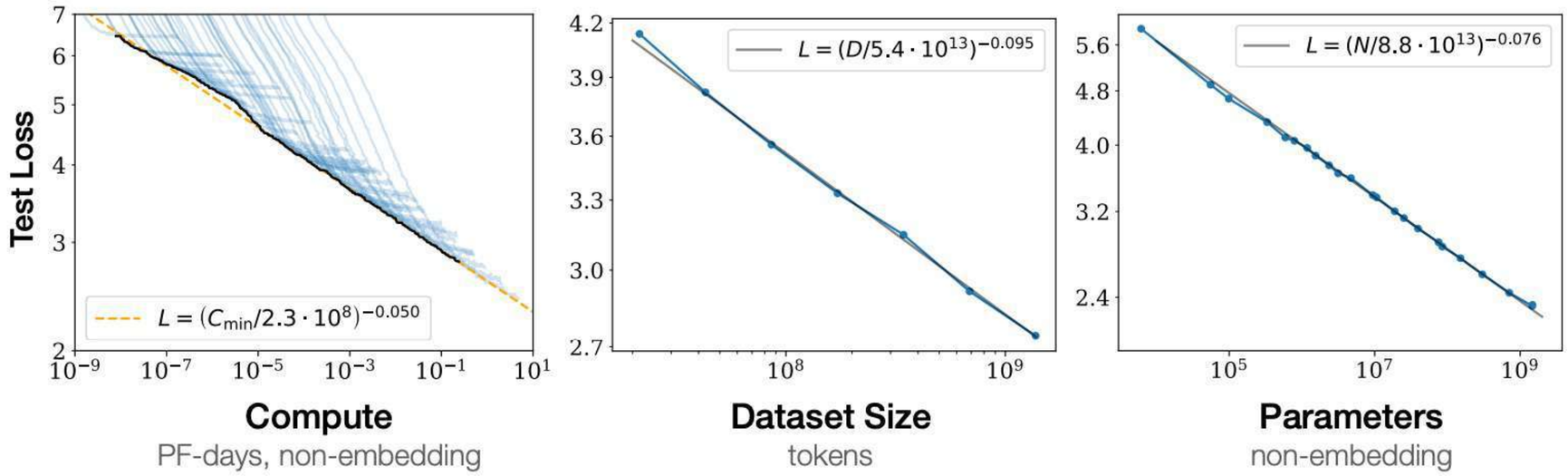
Scaling Law: model the power-law relationship of model performance with respective to three major factors

KM scaling law

- Given a compute budget c, Kaplan et al. (OpenAI team) empirically presented three basic formulas for the scaling law:

$$
\begin{aligned}
L(N) &= \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13} \\
L(D) &= \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13} \\
L(C) &= \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^{8}
\end{aligned}
$$

Language modeling performance (cross-entropy loss *L(*)*) improves smoothly as we increase the model size *(N)*, datasetset size *(D)*, and amount of compute used for training *(C)*.



Kaplan, Jared, et al. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).

Chinchilla scaling law

- Hoffmann et al. (the Google DeepMind team) proposed an alternative form for scaling laws to instruct the compute-optimal training for LLMs.

- The main question for their research is "**If give it a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?**"

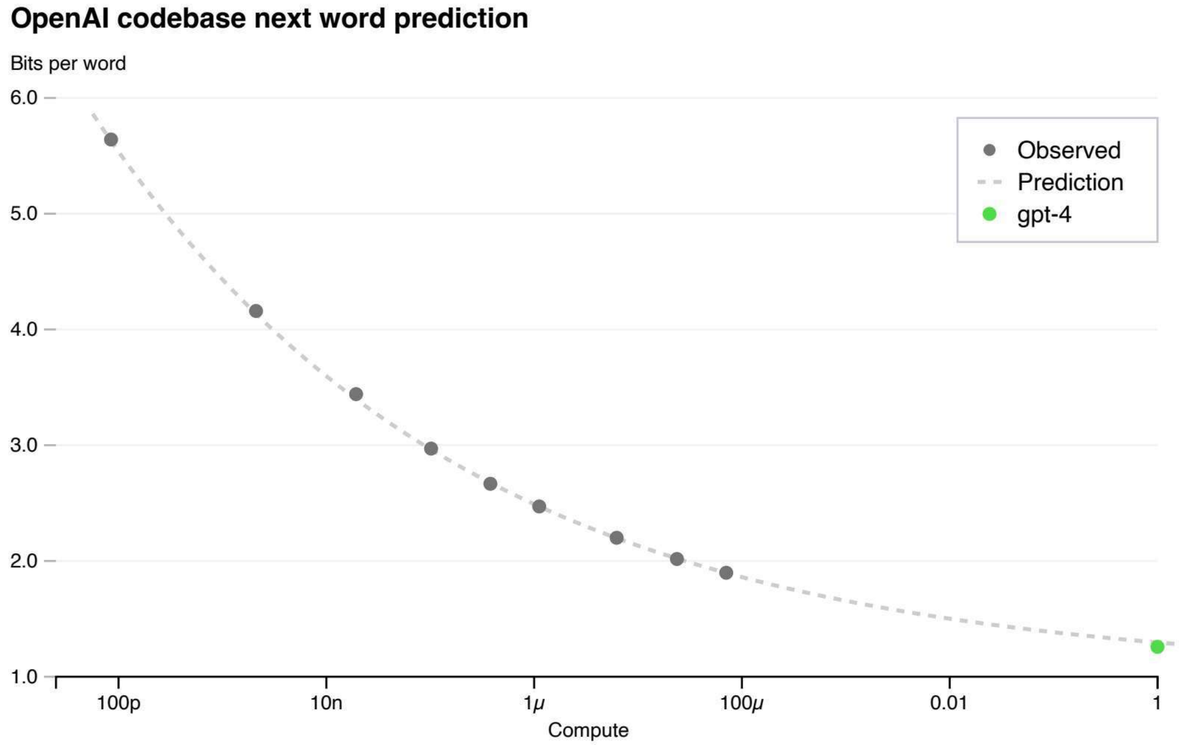
$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

Given an increase in compute budget, the KM scaling law favors a larger budget allocation in model size than the data size, while the Chinchilla scaling law argues that the two sizes should be increased in equal scales

Hoffmann, Jordan, et al. "Training compute-optimal large language models." arXiv preprint arXiv:2203.15556 (2022).

Predictable scaling

- Scaling law can be used to instruct the training of LLMs
  - We can reliably estimate the performance of larger models based on that of smaller models (predictable scaling)
  - OpenAI predicted GPT-4's final loss on their internal codebase by fitting a scaling law

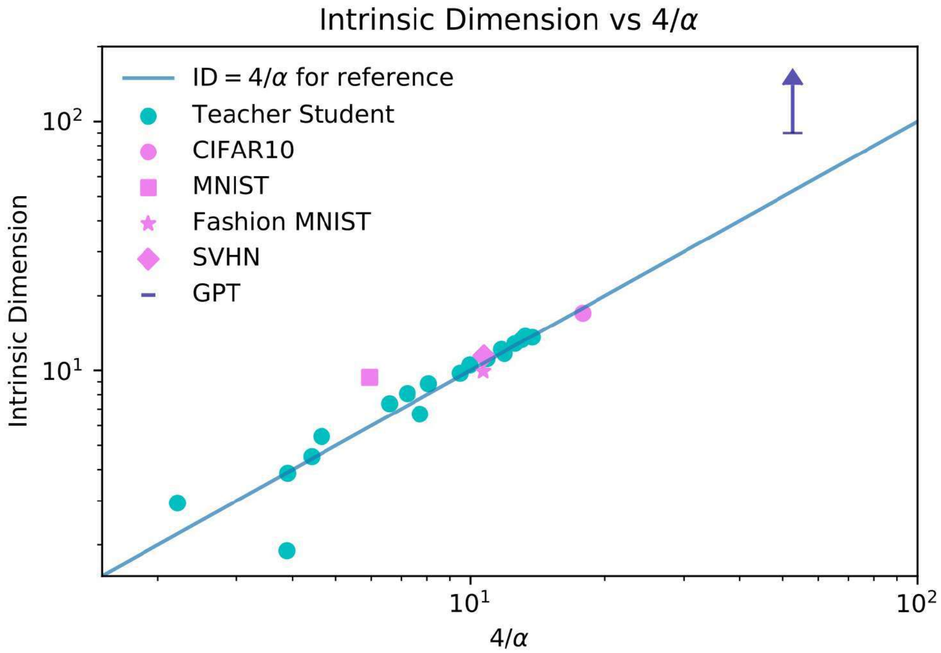


Performance of GPT-4 and smaller models

"Gpt-4 technical report." OpenAI (2023).

Intrinsic dimensionality theory of data scaling laws

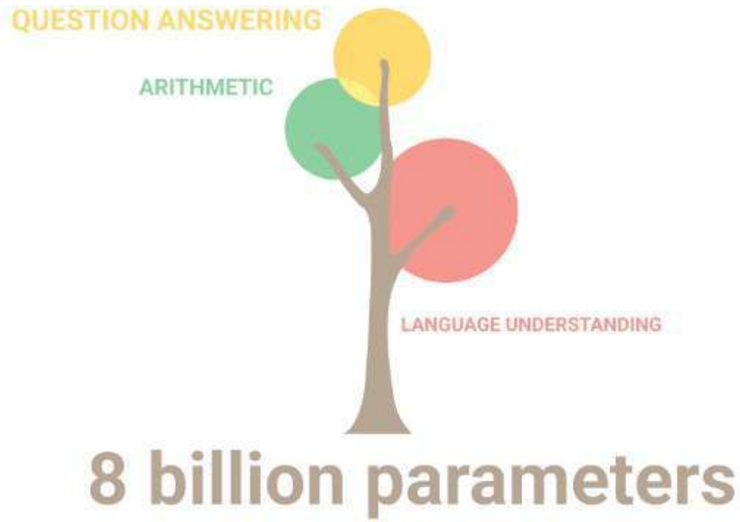
- Scaling law arise due to polynomial rates of learning: $L(N) \propto \frac{1}{N^{\alpha}}$
- The slope α is closely connected to the intrinsic dimensionality of the data (Utkarsh et al., 2022)
- Some recent work (Bahri et al., 2021) have tried to verify this empirically



This figure shows the relationship between the measured intrinsic dimension (ID) of the data manifold and 4/α , where α is the model size scaling exponent.

Reference: Sharma, Utkarsh, and Jared Kaplan. "Scaling laws from the data manifold dimension." The Journal of Machine Learning Research 23.1 (2022): 343-376.
Bahri, Yasaman, et al. "Explaining neural scaling laws." arXiv preprint arXiv:2102.06701 (2021).

Emergency



As the scale of the model increases, the performance improves across tasks while also unlocking new capabilities.

Image from: Zhao, Wayne Xin, et al. "Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance." Google Research Blog

In-context learning

- **In-context learning:** use language models to learn tasks given only a few examples.

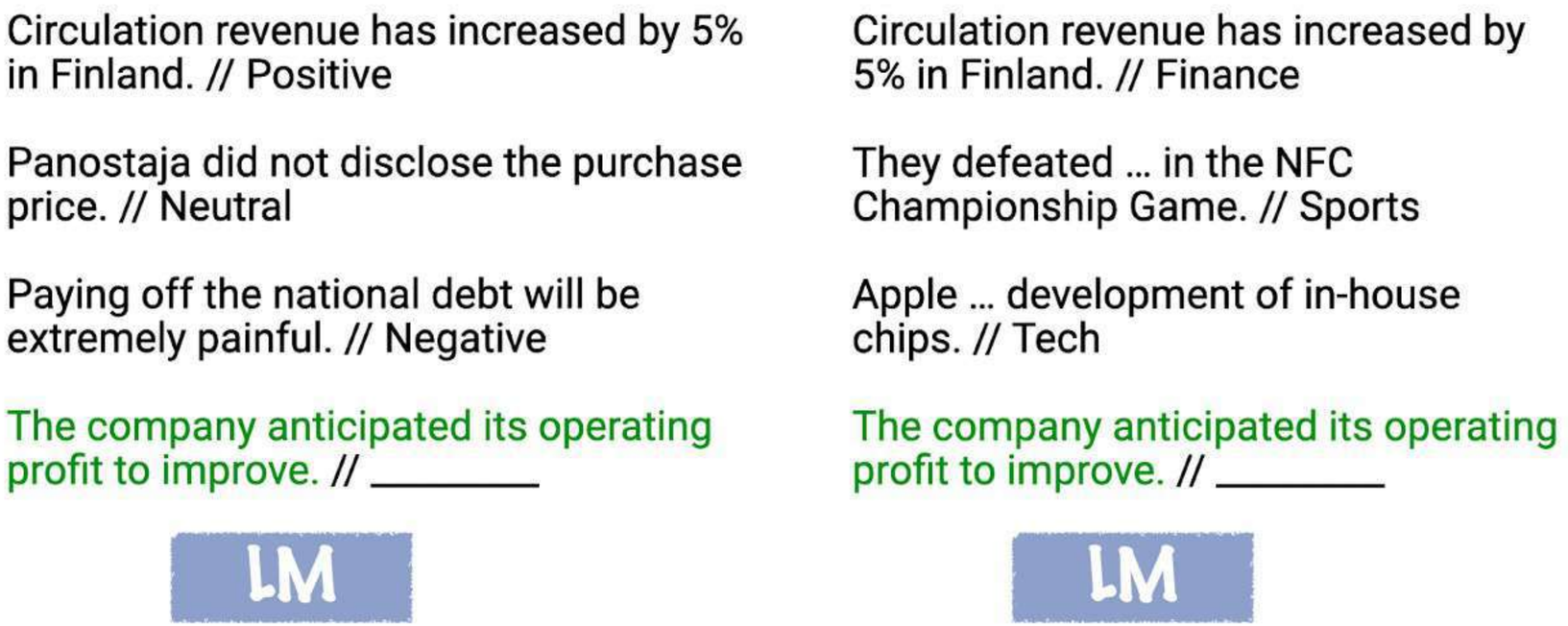
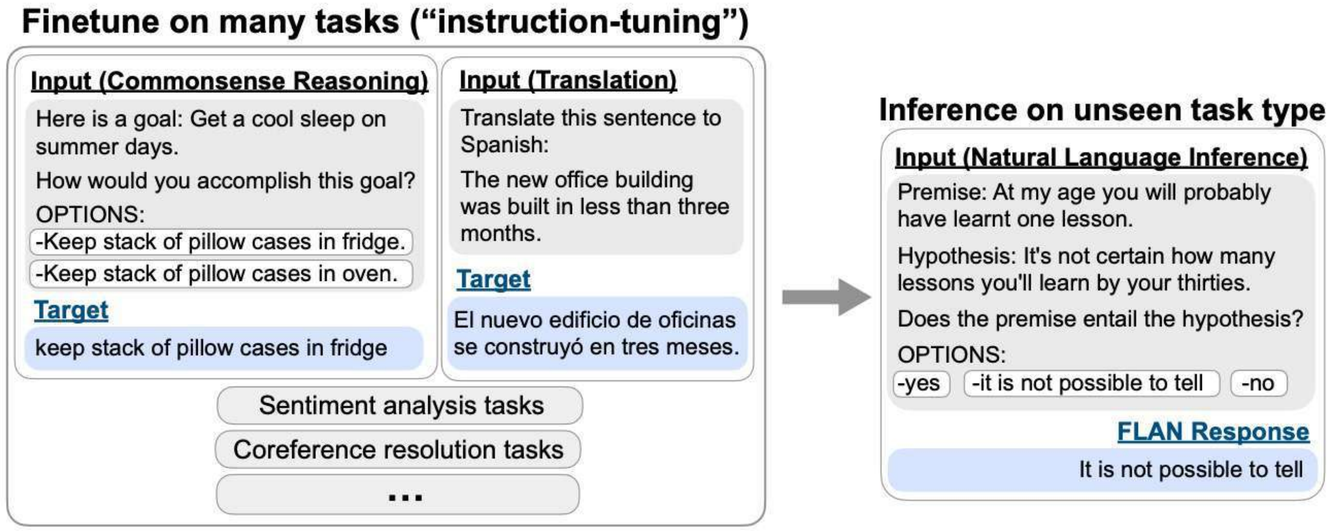
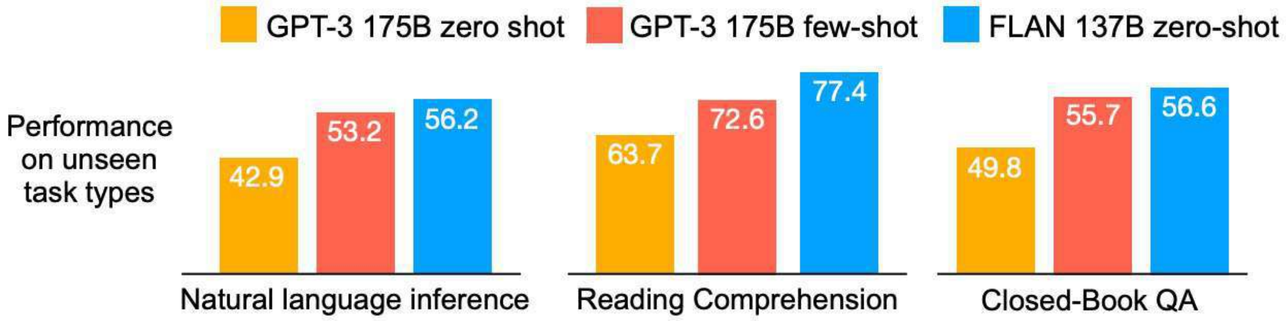


Image from: https://ai.stanford.edu/blog/understanding-incontext/

Instruction following

- LLMs are shown to **perform well on unseen tasks** that are also described in the form of **instructions**



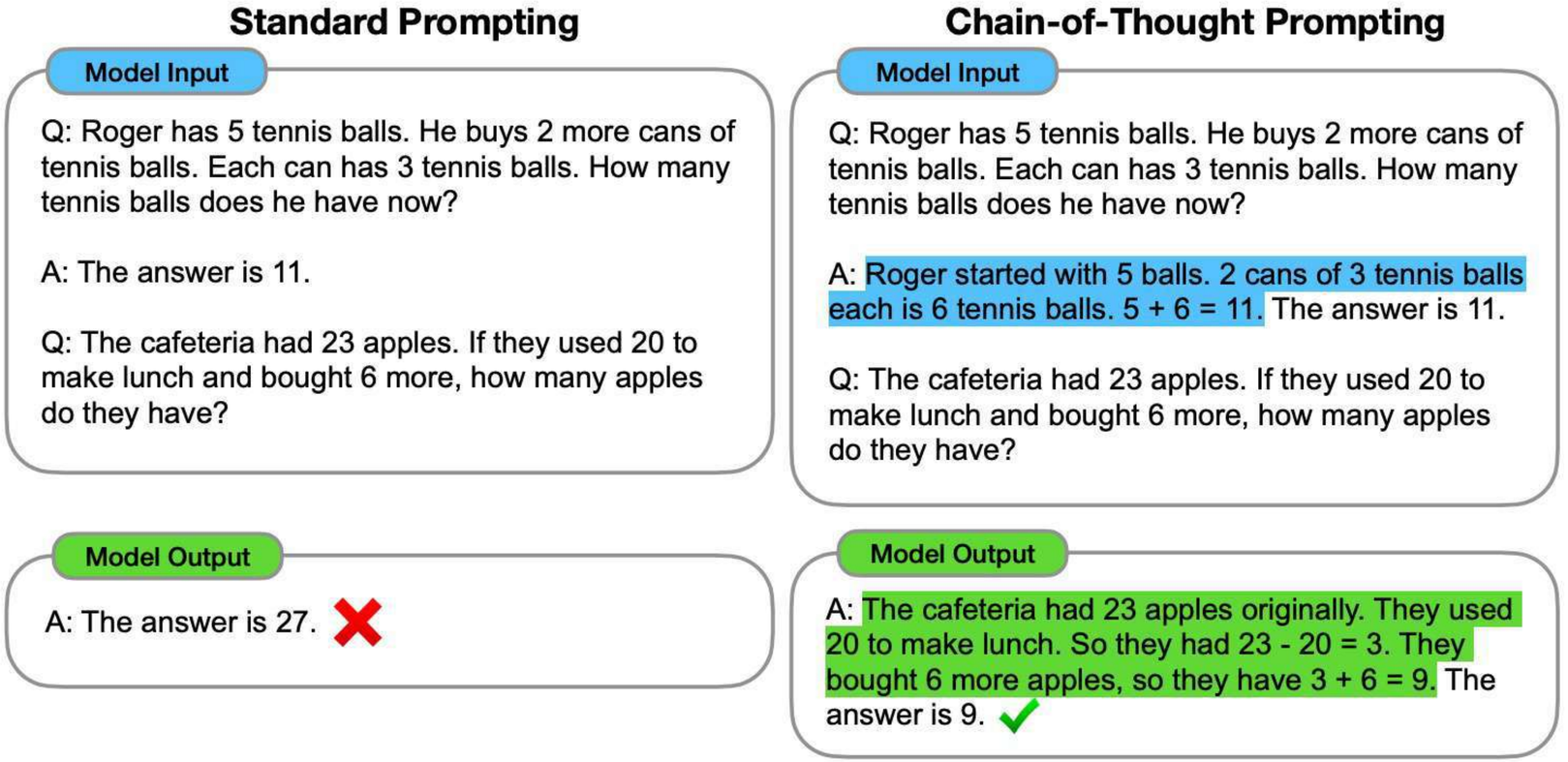


Instruction tuning finetunes a pretrained language model on a mixture of tasks phrased as instructions. At inference time, we evaluate on an unseen task type.

Wei, Jason, et al. "Finetuned language models are zero-shot learners." arXiv preprint arXiv:2109.01652 (2021).

Step-by-step reasoning

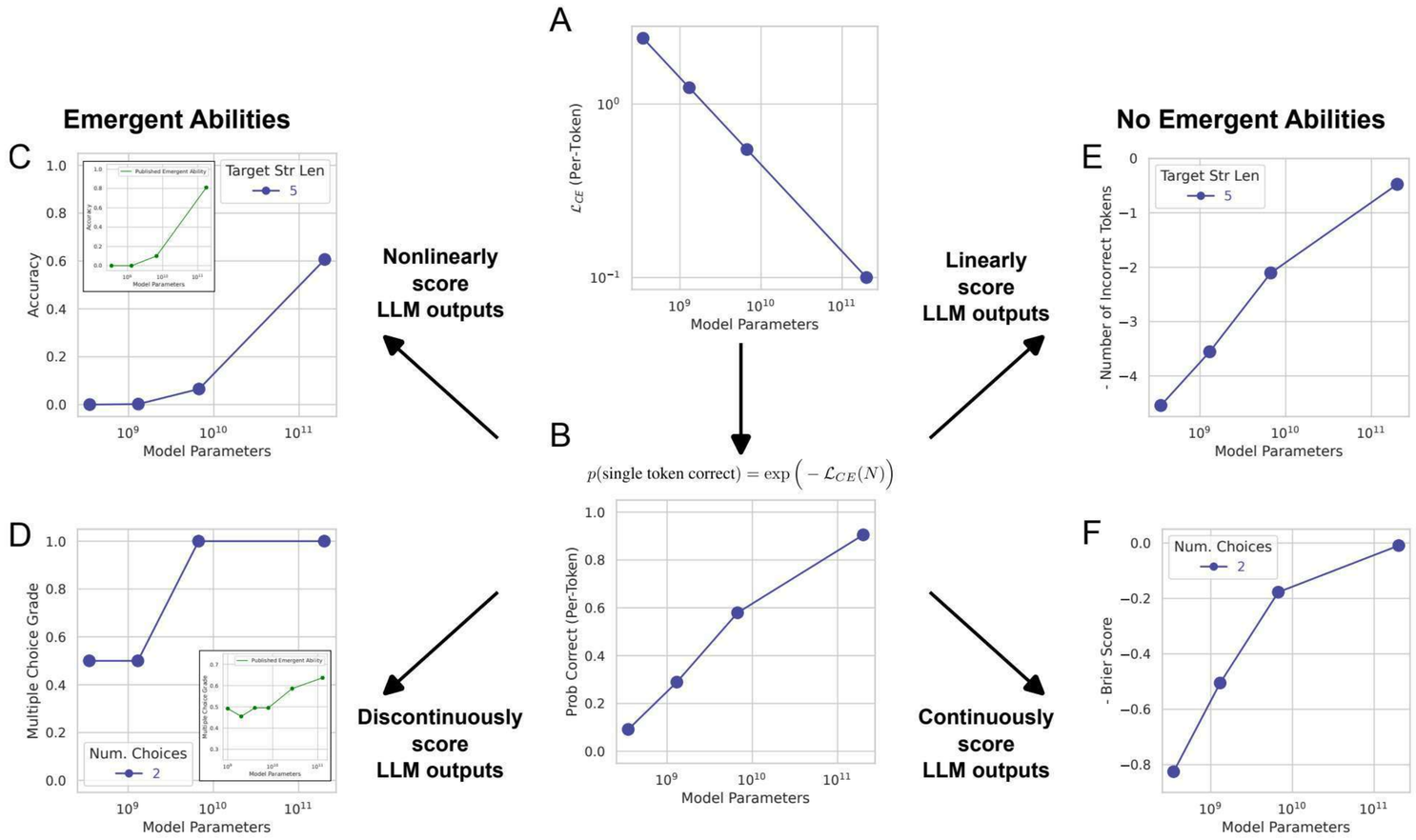
- With the **chain-of-thought (CoT)** prompting strategy, LLMs can **solve complex reasoning tasks** by utilizing the prompting mechanism that involves intermediate reasoning steps for deriving the final answer.



Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." NeurIPS (2022): 24824-24837.

Relation between scaling and emergency

- **Explanation:** **emergent abilities of large language models are created by the researcher's chosen metrics, not unpredictable changes in model behavior with scale.**.
  - Specifically, nonlinear or discontinuous metrics produce seemingly emergent abilities
  - Linear or continuous metrics produce smooth, continuous, predictable changes in model performance.



Schaeffer et al. "Are Emergent Abilities of Large Language Models a Mirage?" NeurIPS (2023).

Larger is more challenging

Scaling

How to efficiently scale up without too much hyper-parameter tuning costs?

Training

How to distributed training large models with different algorithms and frameworks?

Ability eliciting

After pre-training, how to elicit LLMs' different abilities?

Alignment tuning

How to avoid toxic, biased, or even harmful content and align LLMs with human values?

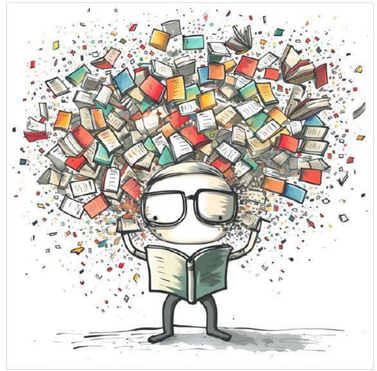
Tool manipulation

How to empower LLMs to efficiently employ external tools?

Core techniques to make LM large

Pre-training

LLM learns from vast amounts of data to understand and generate human-like text, building its foundational knowledge.

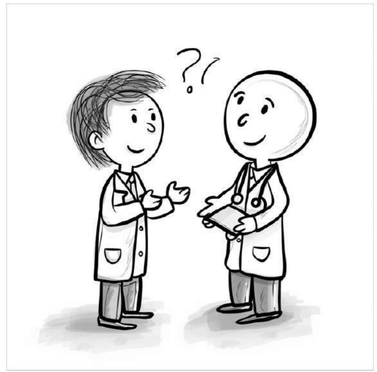


Adaptation

Model is fine-tuned with specific data sets or parameters to tailor its responses to particular tasks or domains

Utilization

Applying the trained and adapted LLM to real-world tasks.



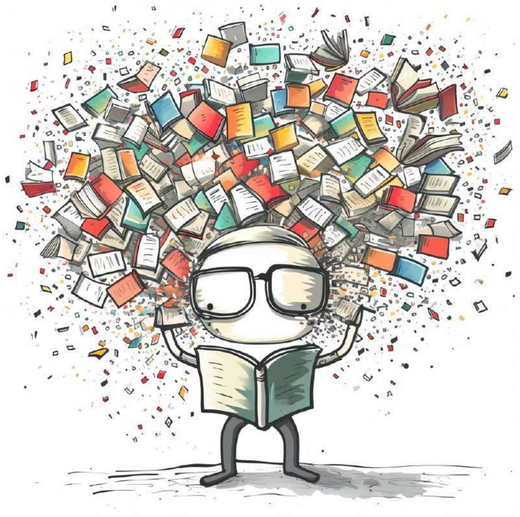
Evaluation

Assesses the LLM's performance, accuracy, and reliability in its tasks.

Core Techniques of Large Language Models

Pre-training

What is pre-training



Pre-training establishes the basis of the abilities of LLMs.

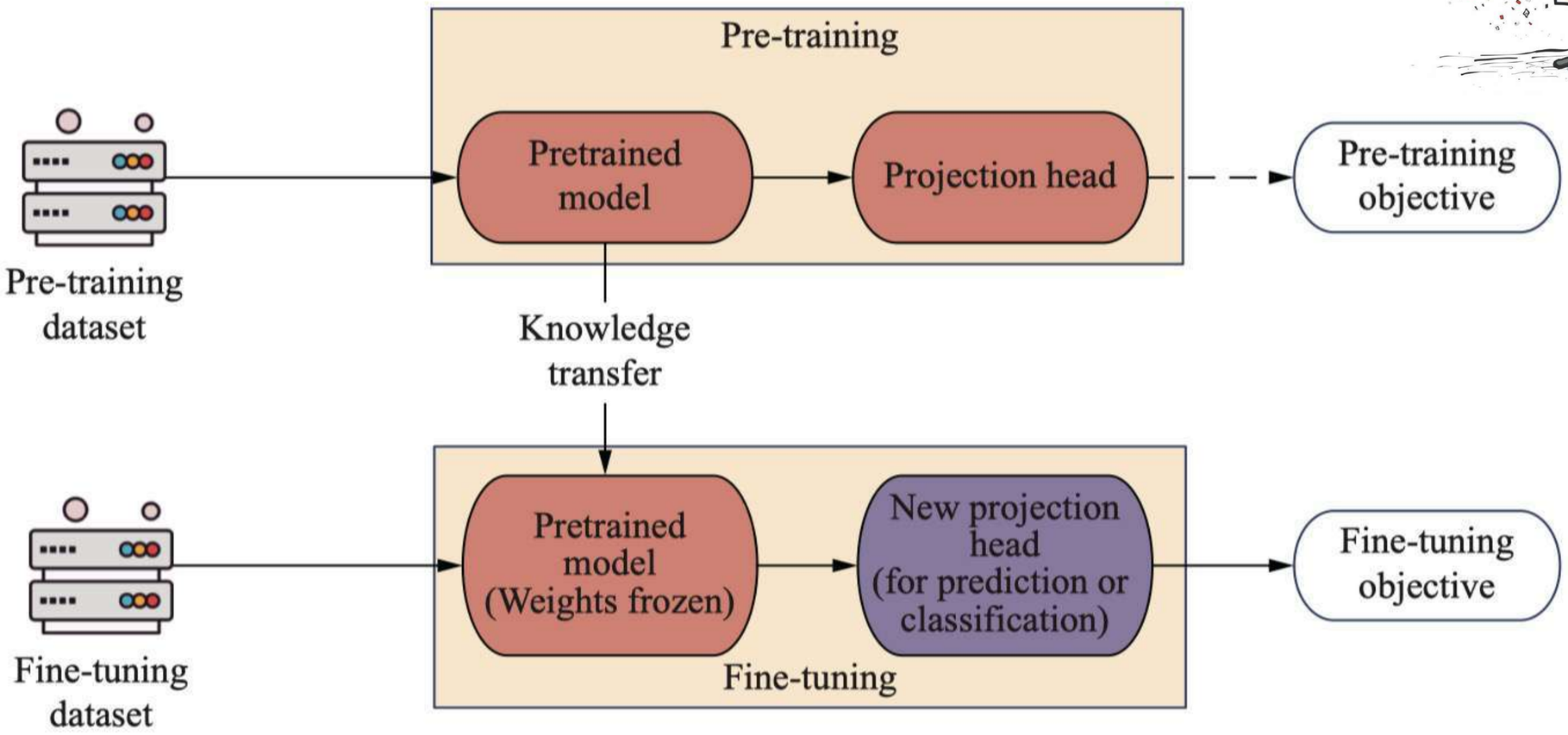
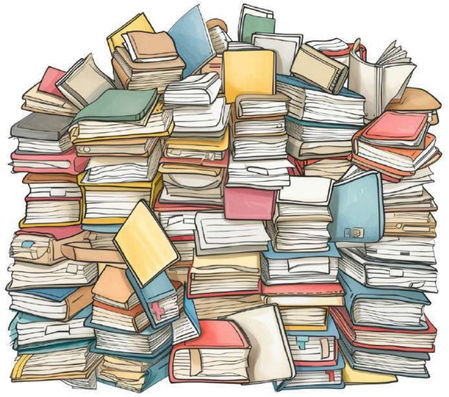


Image from: Qiu, Yixuan, et al. "Pre-training in Medical Data: A Survey." Machine Intelligence Research 20.2 (2023): 147-179.

Core problems in pre-training

Data Preparation



Model Architecture

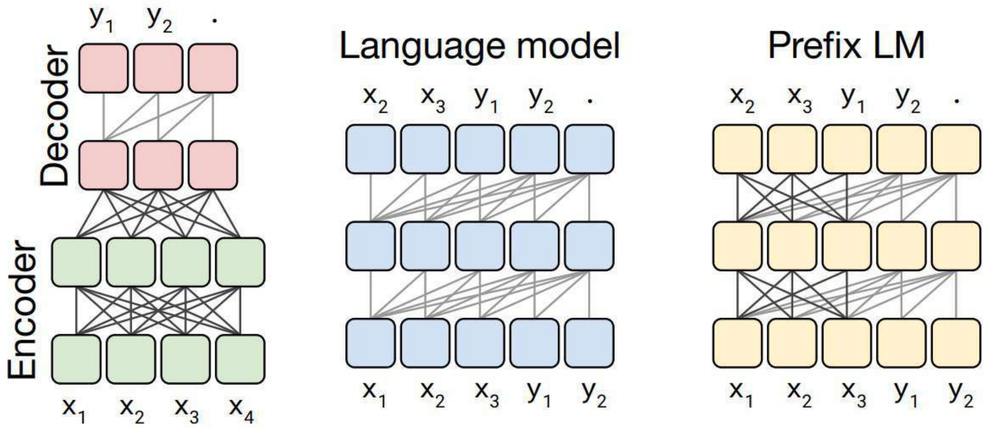


Image from: Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." JMLR 21.1 (2020): 5485-5551.

Model Training

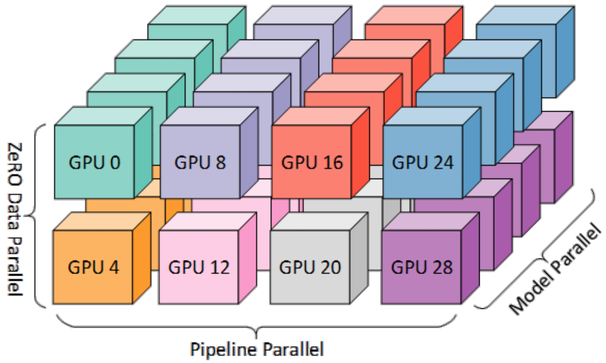
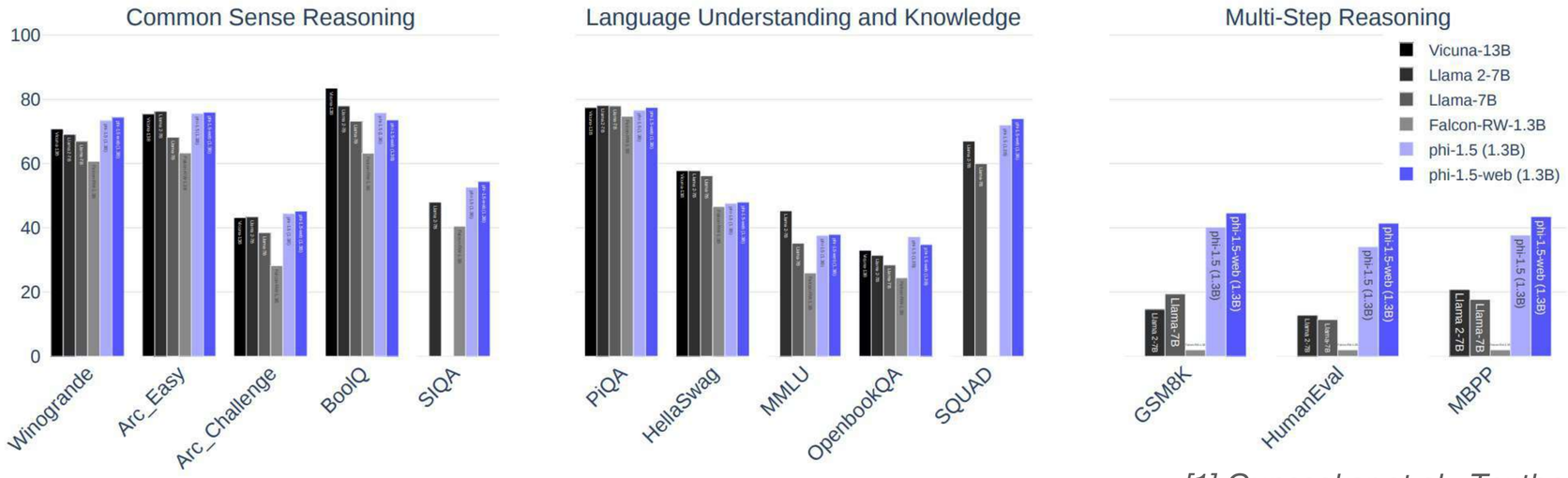


Image from: DeepSpeed

Data preparation

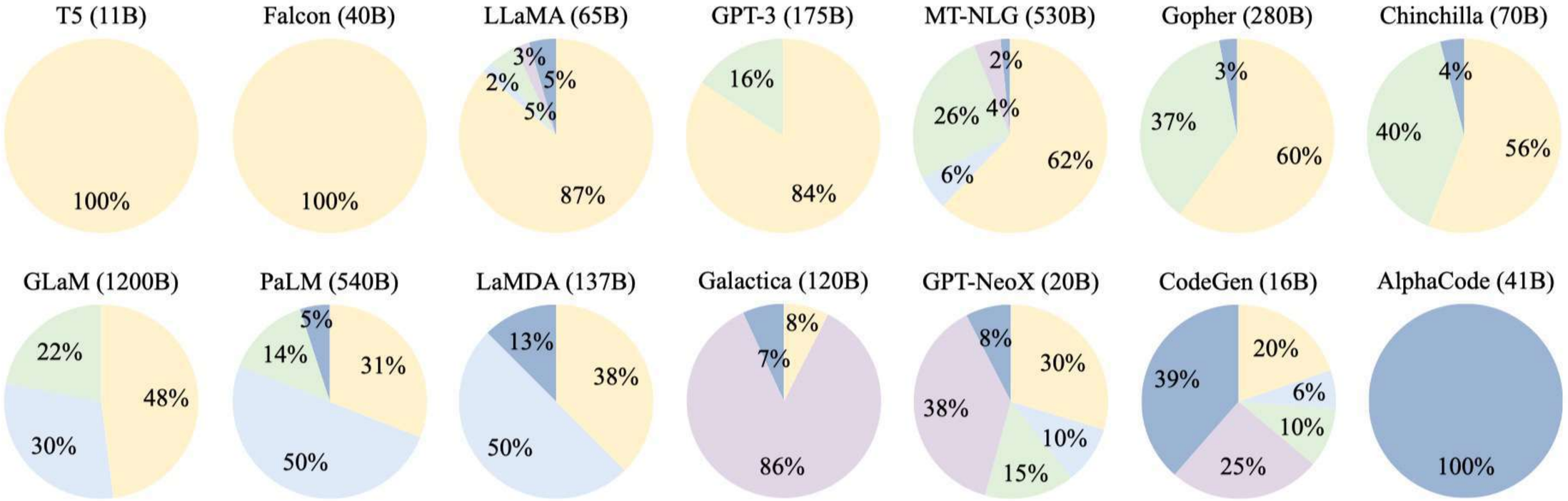
- Collecting a **large amount** of natural language corpus from **various data sources** is key to train LLMs.
  - **General data:** e.g., webpages, books, and conversational text.
  - **Specific data:** e.g., multilingual text, scientific text, code.

- **Training data quality plays a critical role in model performance**
  - Scaling up isn't the only thing that matters
  - Data quality can be more important than data quantity or parameter count.



[1] Gunasekar et al., Textbooks Are All You Need. 2023.
[2] Li et al., Textbooks Are All You Need II: phi-1.5 technical report. 2023.

Data sources for pre-training existing LLMs

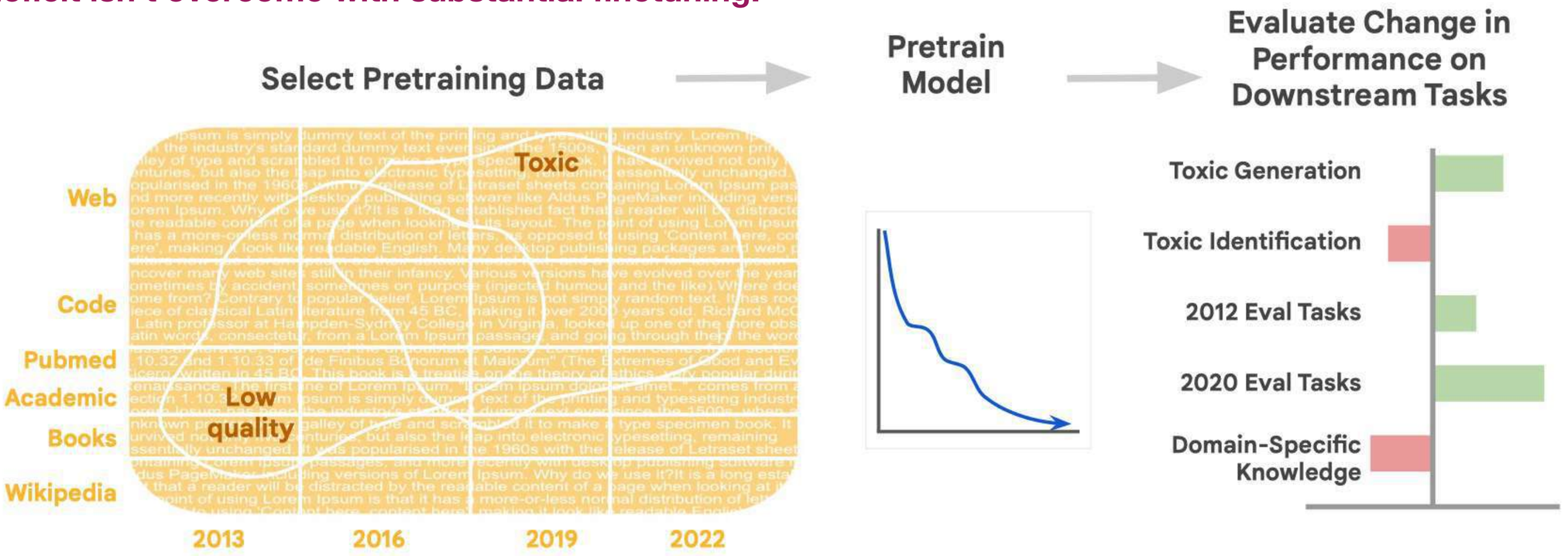


- Webpages — C4 (800G, 2019), OpenWebText (38G, 2023), Wikipedia (21G, 2023)
- Conversation Data — the Pile - StackExchange (41G, 2020)
- Books & News — BookCorpus (5G, 2015), Gutenberg (-, 2021), CC-Stories-R (31G, 2019), CC-NEWES (78G, 2019), REALNEWs (120G, 2019)
- Scientific Data — the Pile - ArXiv (72G, 2020), the Pile - PubMed Abstracts (25G, 2020)
- Code — BigQuery (-, 2023), the Pile - GitHub (61G, 2020)

Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Effects of Data Age, Domain Coverage, Quality, & Toxicity

Performance degradate if evaluation data is either before or after pretraining data collection, and this deficit isn't overcome with substantial finetuning.



No one-size-fits-all solution to filtering training data.
There is a trade-off between task performance and risk of toxic generations.

Image from: Longpre et al. "A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity" arXiv preprint arXiv:2305.13169 (2023).

Example: data selection for language models

Utility function: map data point to utility value
+
Selection mechanism: use utility to select data

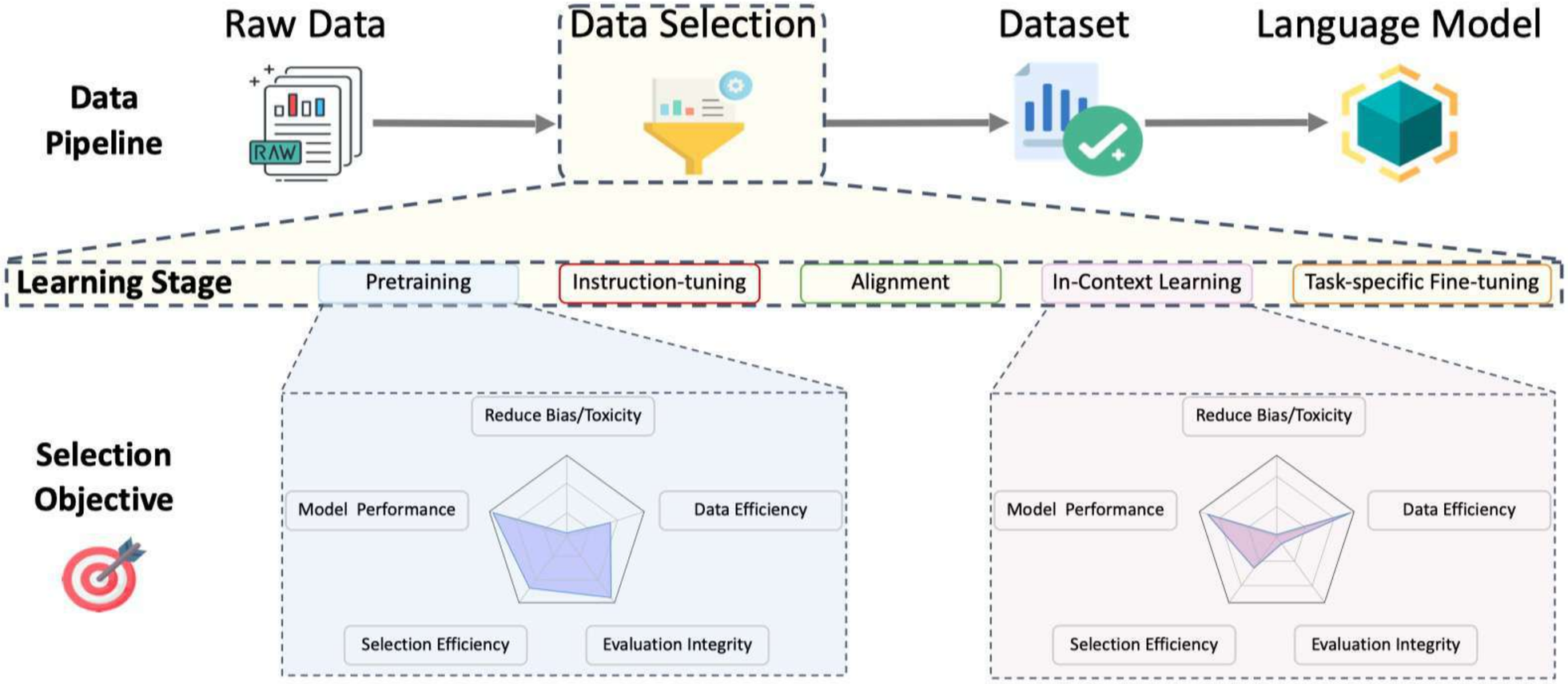


Image from: Albalak et al. "A Survey on Data Selection for Language Models" arXiv preprint arXiv:2042.16827 (2024).

Data filtering pipeline for retraining

Image from: Albalak et al. "A Survey on Data Selection for Language Models" arXiv preprint arXiv:2042.16827 (2024).

Example: Data-Juicer data preprocessing system

- Data-Juicer is a useful data processing system for LLMs, which provides over 50 processing operators and tools.

https://github.com/alibaba/data-juicer

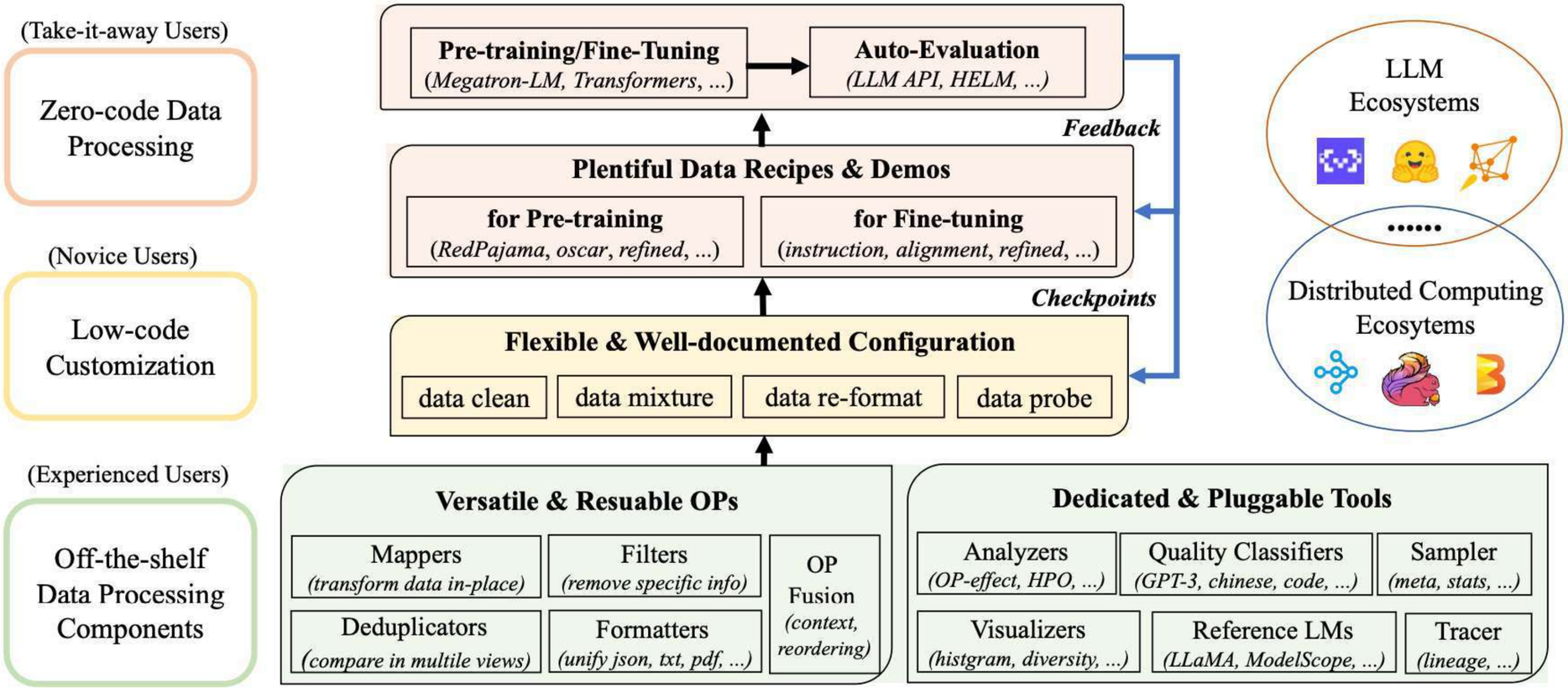


Image from: Chen, et al. "Data-Juicer: A One-Stop Data Processing System for Large Language Models" arXiv preprint (2023).

73 Data Scheduling

- After data preprocessing, it is essential to design suitable strategies to **schedule these multi-source data for pre-training** a capable LLM
  - The proportion of each data source (**data mixture**)
  - The order in which each data source is scheduled for training (**data curriculum**)

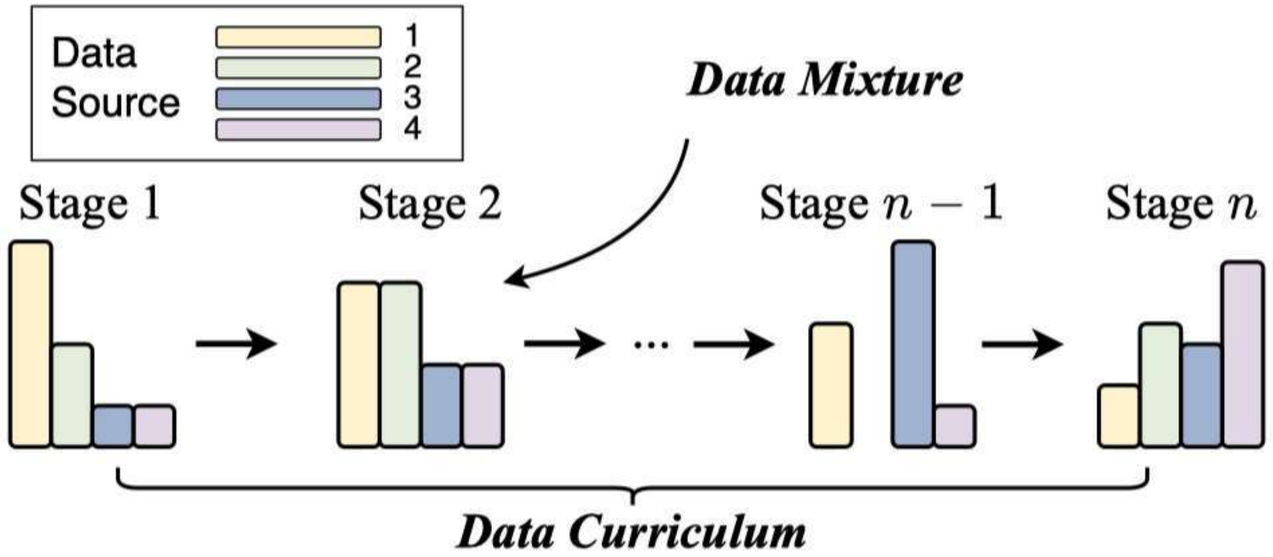


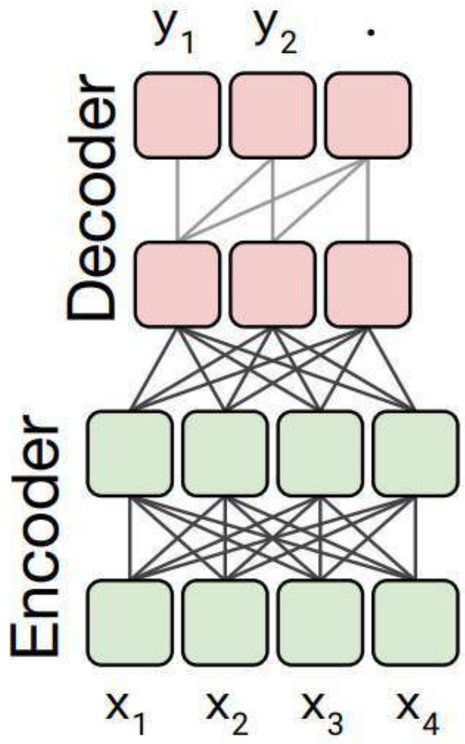
Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Summary of data preparation

- **Data collection.** It is suggested to include **diverse data** sources in the pre-training data.
- **Data cleaning.** After data collection, it is crucial to **clean the raw corpus** to enhance its quality as possible.
- **Data scheduling.** Appropriately **tuning the data mixture and the specific order** of data during the pre-training of LLMs is beneficial.

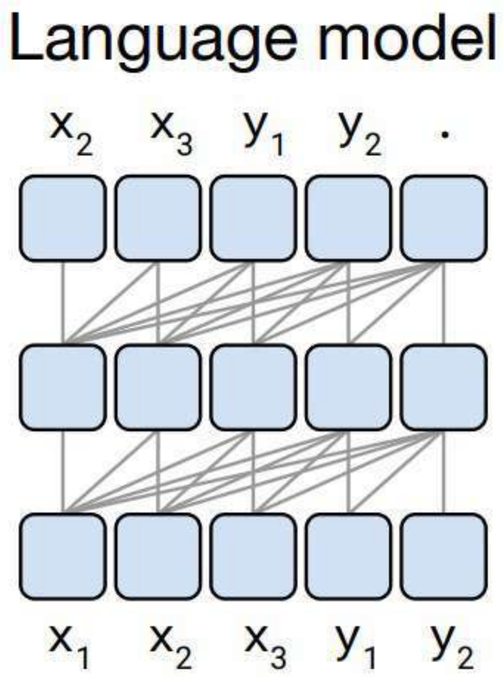
Different model architectures

T5, BART, Flan-T5



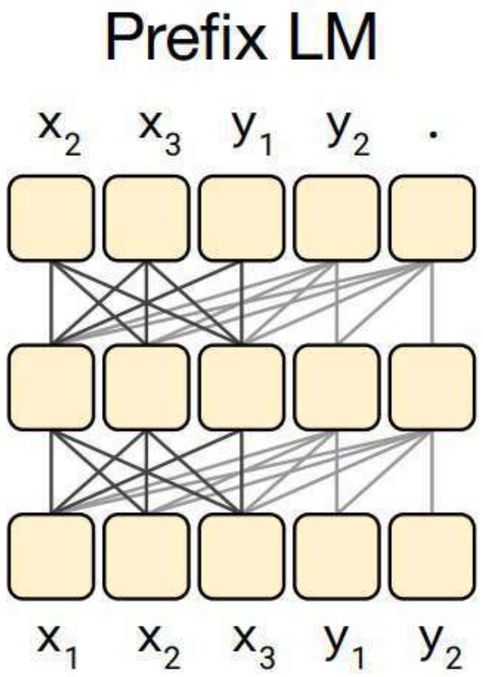
Encoder-Decoder

GPT-series, Llama, OPT, BLOOM, Gopher, OLMo



Cause Decoder

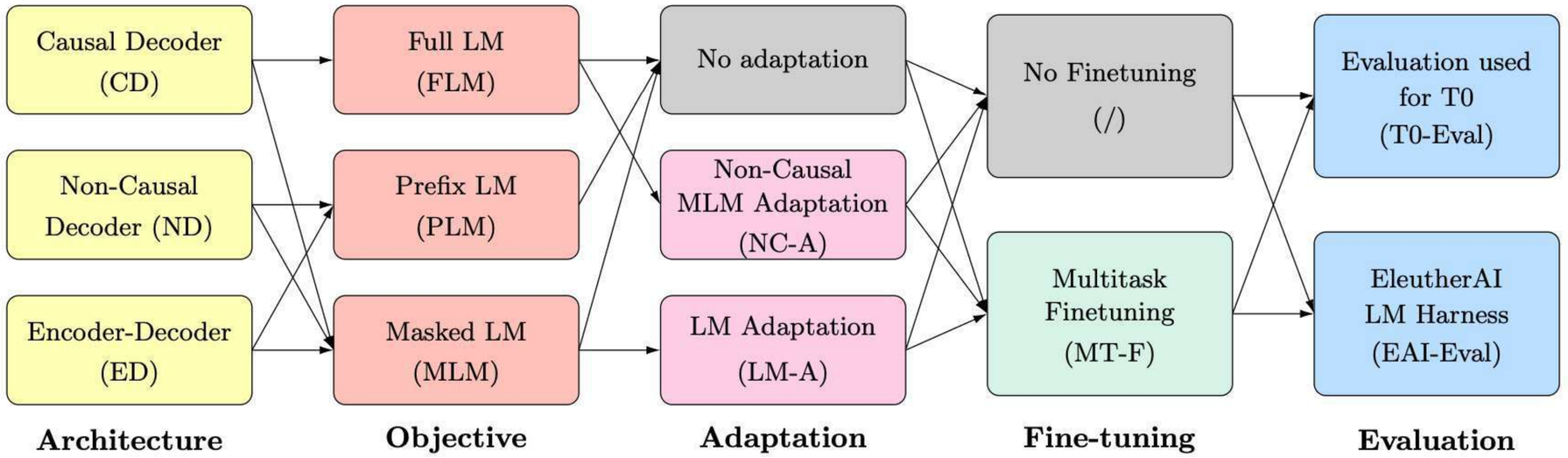
UniLM, GLM-130B, U-PaLM



Prefix Decoder

Image from: Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." JMLR 21.1 (2020): 5485-5551.

Compare different model architectures



1. Purely self-supervised pretraining: causal decoder-only pretrained with full language modeling performs best (strongest zero-shot generalization)
2. Adding a multitask finetuning step: encoder-decoder pretrained with masked language modeling performs best.

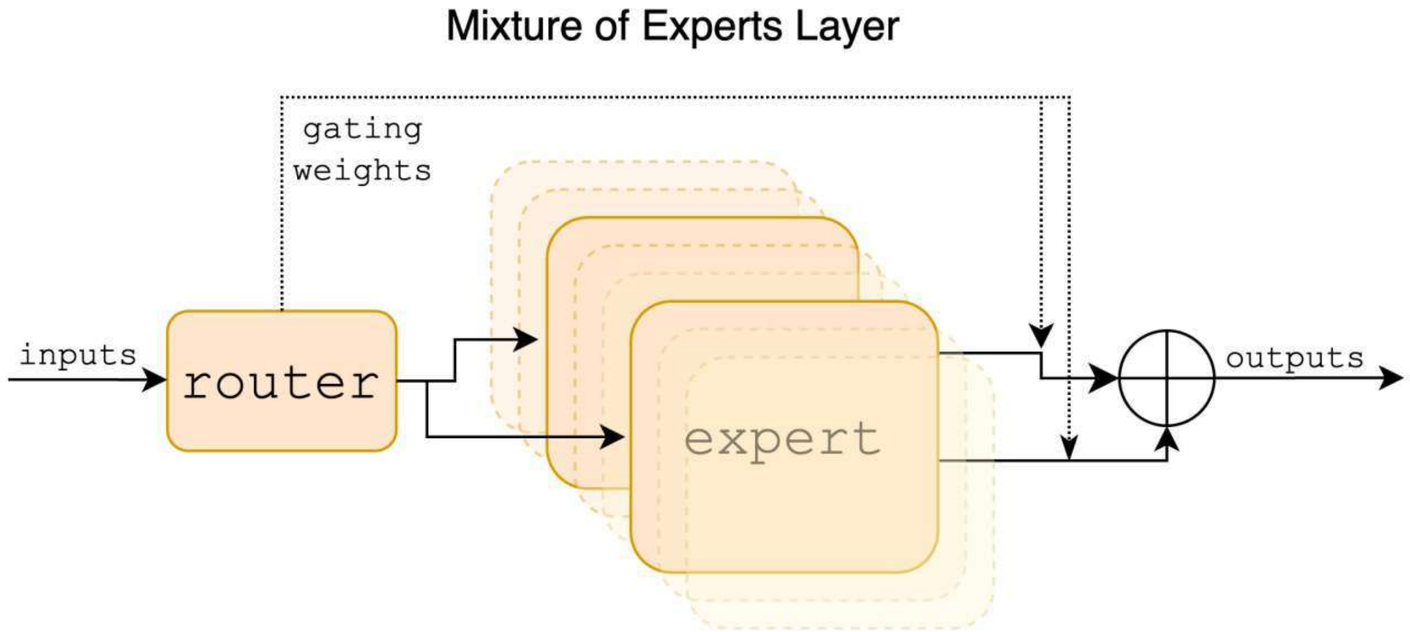
An extensive study of architecture, objective, adaptation, and finetuning impact on zero-shot generalization

Wang, Thomas, et al. "What language model architecture and pretraining objective works best for zero-shot generalization?."ICML, 2022.

More about why decoder-only architecture is popular

- Bi-directional attention can degrade to **low-rank** *(Dong et al., Attention is not all you need: pure attention loses rank doubly exponentially with depth. ICML, 2021.)*
- **Decoder only + next token prediction** is more challenging task, thus can learn better representations with large model size and abundant data.
- Causal attention implicitly encodes positional information.
- More research about the scaling laws of decoder-only architecture available.
- It’s easier to prepare decoder-only pertaining dataset.

Mixture of Experts (MoE)



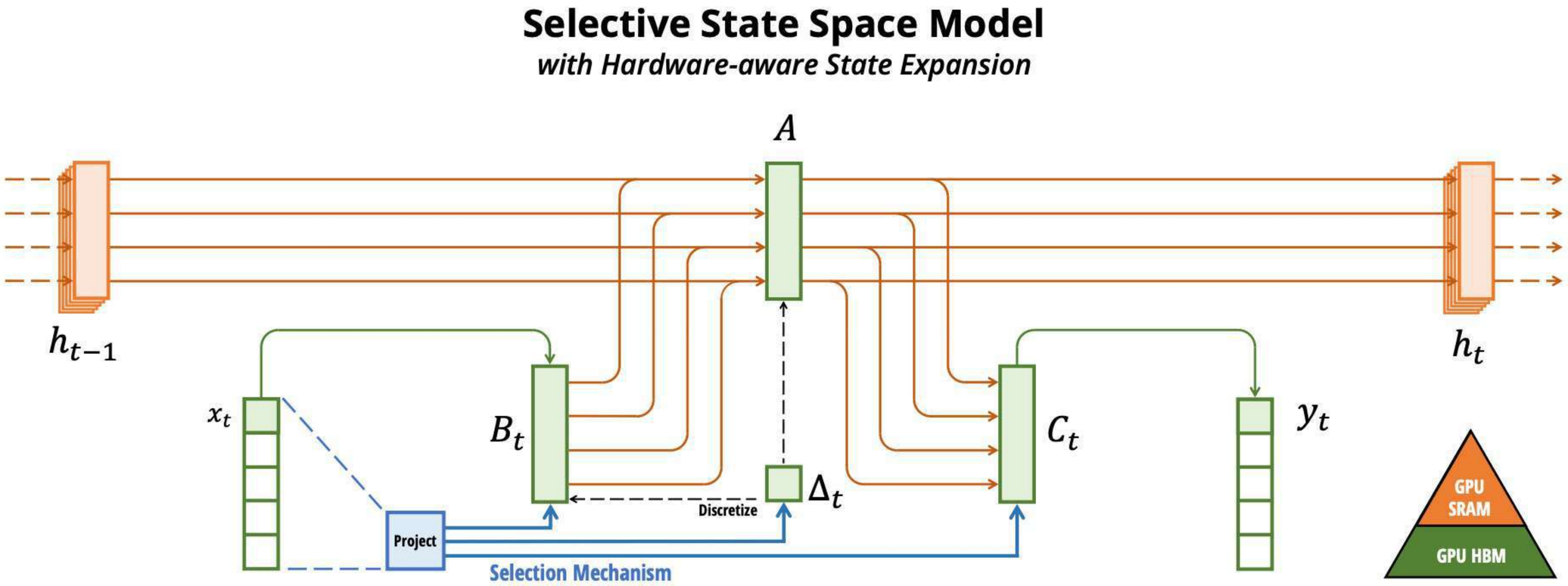
GPT-4 is rumoured to be a MoE model consisting of 8x 222B model, and when completing a token, it would use two experts, which is 444B running at a time.

Illustration of an MoE layer. Each input vector is assigned to the experts selected by a router. The layer's output is the weighted sum of the outputs of the selected experts.

Image from: Mistral.ai. "Mixtral of Experts" arXiv preprint (2023).

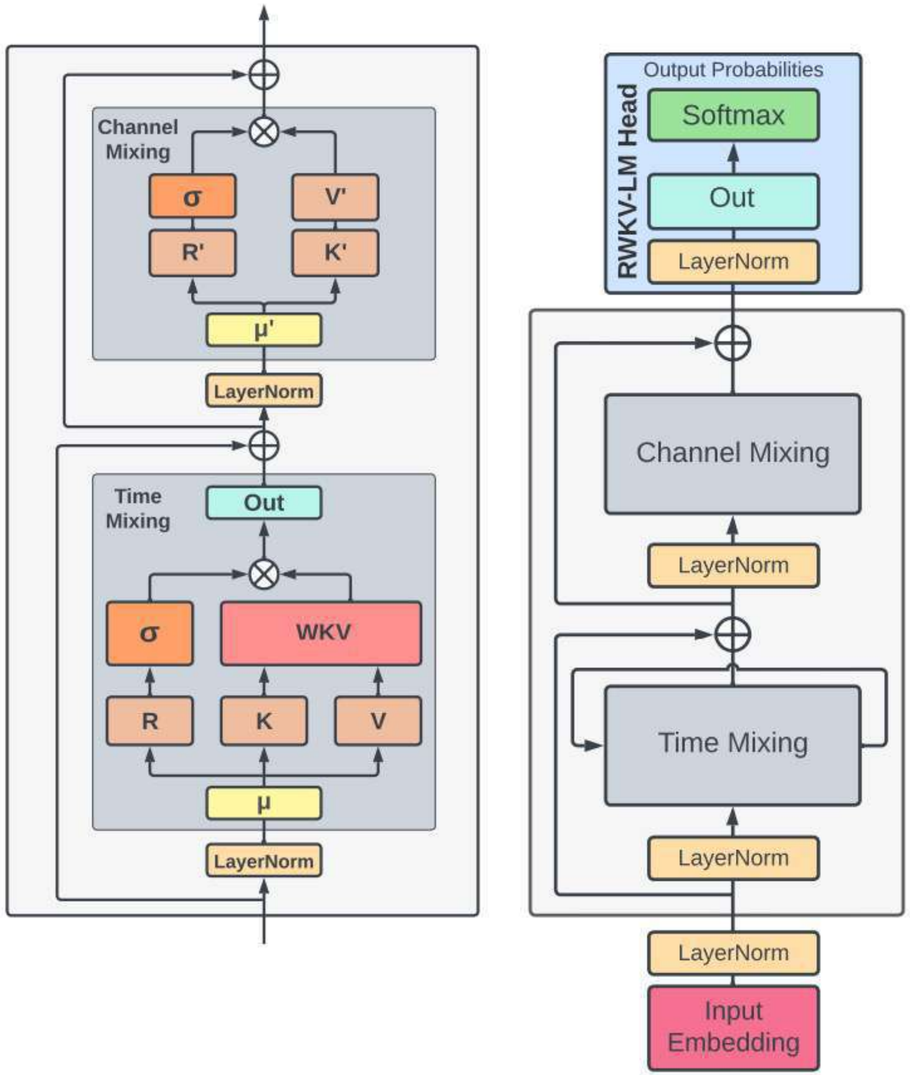
Other model architectures: Mamba

Mamba: Parameterized state space models



Albert Gu and Tri Dao. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". arXiv preprint. 2023.

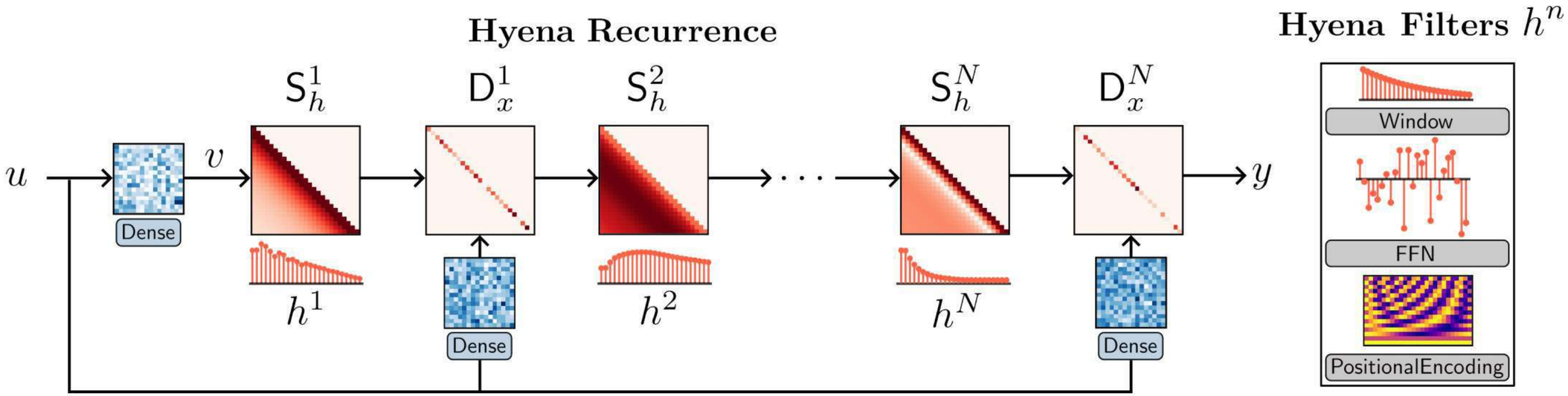
Other model architectures: RWKV



RWKV: Transformer-like architectures that incorporate recursive update mechanisms

Peng et al.. "RWKV: Reinventing RNNs for the Transformer Era". arXiv preprint. 2023.

Other model architectures: Hyena



Hyena: a subquadratic drop-in replacement for attention constructed by interleaving implicitly parametrized long convolutions and data-controlled gating.

Poli et al.. "Hyena Hierarchy: Towards Larger Convolutional Language Models". ICML. 2023.

Transformer Variants

- Since the launch of Transformer, various improvements have been proposed to enhance its training stability, performance, and computational efficiency.

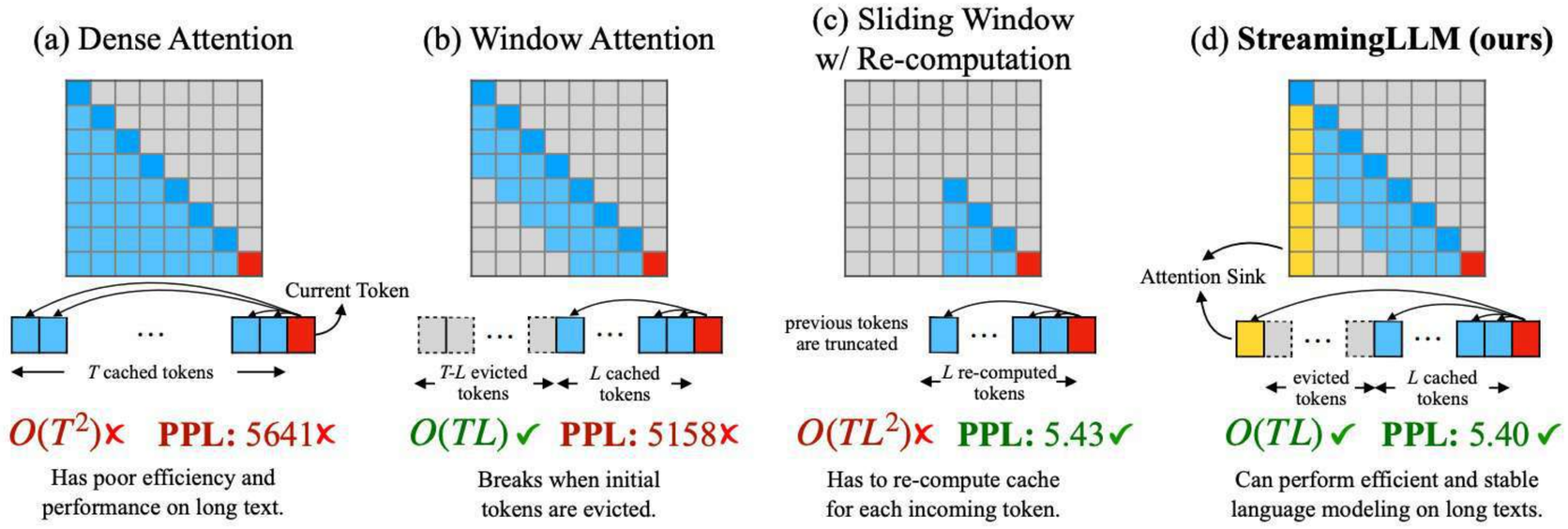
Configuration	Method	Equation
Normalization position	Post Norm [22]	$\text{Norm}(\mathbf{x}+\text{Sublayer}(\mathbf{x}))$
	Pre Norm [26]	$\mathbf{x}+\text{Sublayer}(\text{Norm}(\mathbf{x}))$
	Sandwich Norm [255]	$\mathbf{x}+\text{Norm}(\text{Sublayer}(\text{Norm}(\mathbf{x})))$
Normalization method	LayerNorm [256]	$\frac{\mathbf{x}-\mu}{\sigma}\cdot\gamma+\beta,\quad \mu=\frac{1}{d}\sum_{i=1}^{d}x_i,\quad \sigma=\sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i-\mu))^2}$
	RMSNorm [257]	$\frac{\mathbf{x}}{\text{RMS}(\mathbf{x})}\cdot\gamma,\quad \text{RMS}(\mathbf{x})=\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}$
	DeepNorm [258]	$\text{LayerNorm}(\alpha\cdot\mathbf{x}+\text{Sublayer}(\mathbf{x}))$
Activation function	ReLU [259]	$\text{ReLU}(\mathbf{x})=\max(\mathbf{x},\mathbf{0})$
	GeLU [260]	$\text{GeLU}(\mathbf{x})=0.5\mathbf{x}\otimes[1+\text{erf}(\mathbf{x}/\sqrt{2})],\quad \text{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$
	Swish [261]	$\text{Swish}(\mathbf{x})=\mathbf{x}\otimes\text{sigmoid}(\mathbf{x})$
	SwiGLU [262]	$\text{SwiGLU}(\mathbf{x}_1,\mathbf{x}_2)=\text{Swish}(\mathbf{x_1})\otimes\mathbf{x_2}$
	GeGLU [262]	$\text{GeGLU}(\mathbf{x}_1,\mathbf{x}_2)=\text{GeLU}(\mathbf{x_1})\otimes\mathbf{x_2}$
Position embedding	Absolute [22]	$\mathbf{x}_i=\mathbf{x}_i+\mathbf{p}_i$
	Relative [82]	$A_{ij}=\mathbf{W}_q\mathbf{x}_i\mathbf{x}_j^T\mathbf{W}_k^T+r_{i-j}$
	RoPE [263]	$A_{ij}=\mathbf{W}_q\mathbf{x}_i\mathbf{R}_{\Theta,i-j}\mathbf{x}_j^T\mathbf{W}_k^T=(\mathbf{W}_q\mathbf{x}_i\mathbf{R}_{\Theta,i})(\mathbf{W}_k\mathbf{x}_jR_{\Theta,j})^T$
	ALiBi [264]	$A_{ij}=\mathbf{W}_q\mathbf{x}_i\mathbf{x}_j^T\mathbf{W}_k^T-m(i-j)$

Table from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Long context modeling

- Making a language model pre-trained on texts of length L predict the T-th token ($T \gg L$)
  - Many real applications need such ability: process PDF, story writing, etc.

- **Can we deploy an LLM for infinite-length inputs without sacrificing efficiency and performance?**

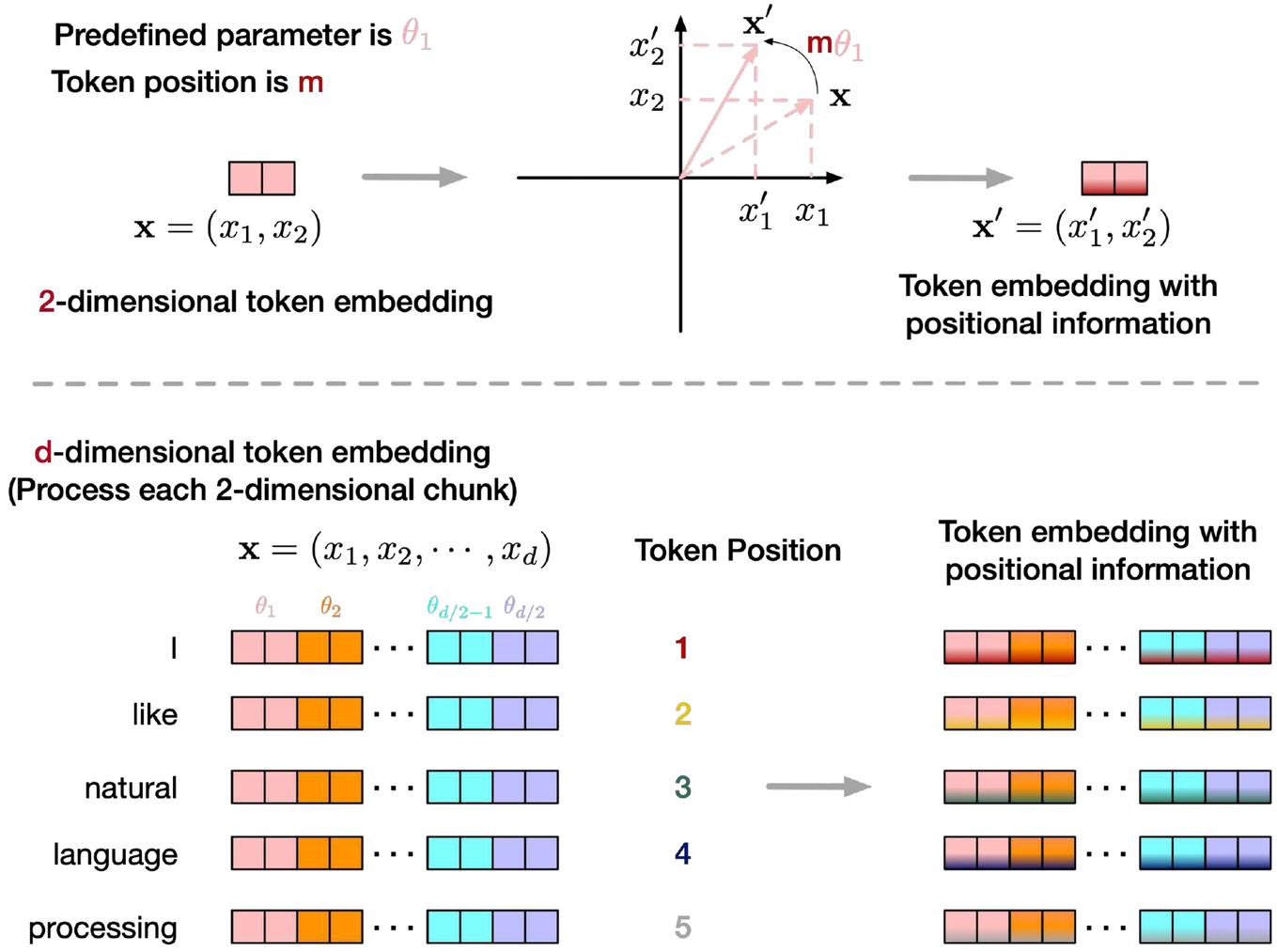


Key observation: a surprisingly large amount of attention score is allocated to the initial tokens in autoregressive LLMs

Based on the observation, StreamingLLM keeps the attention sink tokens' KV together with the sliding window's KV.

Image from: Xiao et al. "Efficient streaming language models with attention sinks" arXiv preprint (2023).

Example: Resonance RoPE – Enhancing LLM's Length Generalization



Reference: Su et al., "RoFormer: Enhanced Transformer with Rotary Position Embedding" Neurocomputing, 2024.

Example: Resonance RoPE – Enhancing LLM's Length Generalization

Theoretical Guarantee **Reducing generalization gap in train-short-test-long settings!**

RoPE rotates q and k in self-attention by an angle to represent tokens' distance

Compute q, k with RoPE features…

$$\boldsymbol{q}_m = \boldsymbol{R}^d_{\Theta,m}\boldsymbol{W}_q\boldsymbol{x}_m$$
$$\boldsymbol{k}_n = \boldsymbol{R}^d_{\Theta,n}\boldsymbol{W}_k\boldsymbol{x}_n$$
$$\boldsymbol{q}_m^\top\boldsymbol{k}_n = \boldsymbol{x}_m^\top\boldsymbol{W}_q\boldsymbol{R}^d_{\Theta,n-m}\boldsymbol{W}_k\boldsymbol{x}_n$$

RoPE feature matrix is block-diagonal.
The j-th block involves the token position m and an inverse frequency θ_j

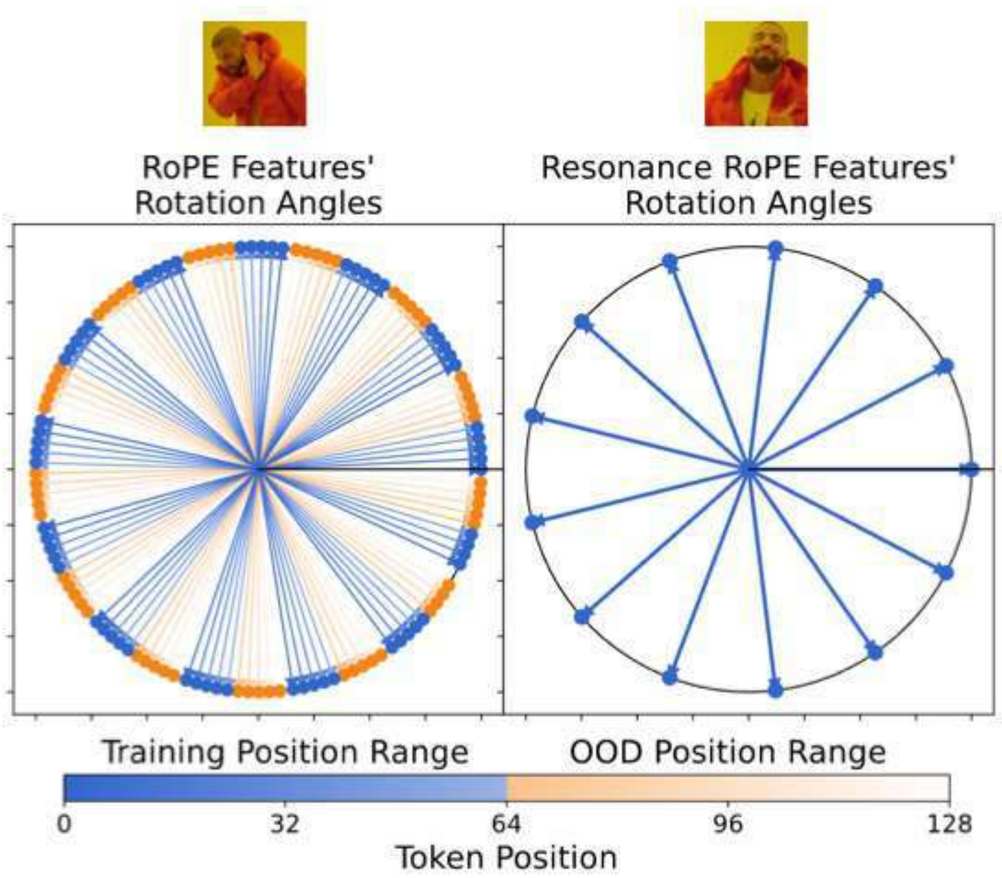
$$\boldsymbol{R}_{\theta_j,m} = \begin{pmatrix} \cos m\theta_j & -\sin m\theta_j \\ \sin m\theta_j & \cos m\theta_j \end{pmatrix}$$

The rotation angles $\boldsymbol{m\theta_j}$ has a wavelength $\boldsymbol{\lambda_j = \frac{2\pi}{\theta_i}}$, usually an irrational number

Reference: Wang et al., "Resonance RoPE: Improving Context Length Generalization of Large Language Models" Findings of ACL 2024.

Example: Resonance RoPE – Enhancing LLM's Length Generalization

The irrational λ creates a *phase shift* after every full rotation on the unit circle:



Existing RoPE-scaling methods:

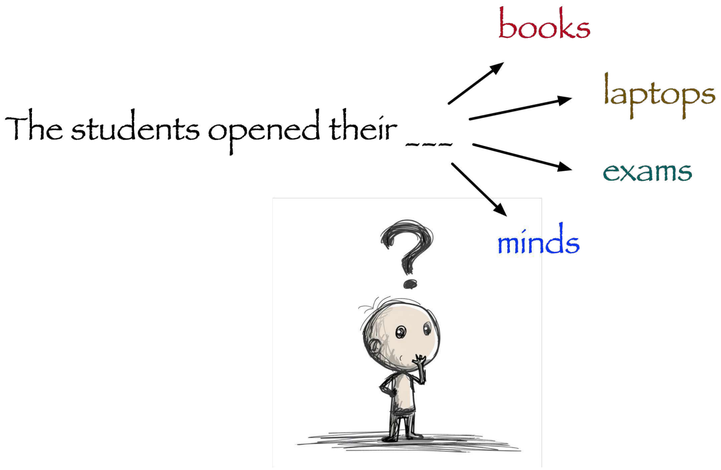
- In train-short-test-long settings, reduce the extrapolation of rotation angles
- Neglects the *interpolation* of rotation angles produced by the phase shift

Resonance RoPE:

- ***Rounds the wavelength of RoPE features to their nearest integer***
- ***Each RoPE feature "resonates" with a span length***
- ***Eliminates interpolation of rotary angles in train-short-test-long settings***

Reference: Wang et al., "Resonance RoPE: Improving Context Length Generalization of Large Language Models" Findings of ACL 2024.

Training Tasks: Causal Language Modeling (CLM)



$$L_{CLM}^{(x)} = -\frac{1}{|x|}\sum_{i=1}^{|x|} logP(x_i/x_{<i})$$

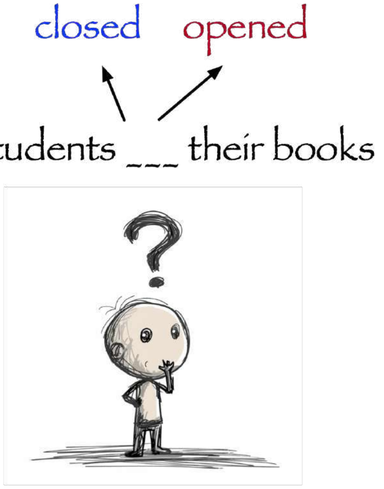
where:

$x = \{x_1, x_2, x_3, \ldots, x_{|x|}\}$ represents a sequence

$x_{<i} = x_1, x_2, x_3, ..x_{i-1}$

CLM: a Unidirectional Language Model that predicts the next word given the context.

Training Tasks:
Masked Language Modeling (MLM)



$$L_{MLM}^{(x)} = -\frac{1}{|M_x|} \sum_{i \in M_x} logP(x_i / x_{\setminus M_x})$$

where:

$x_{\setminus M_x}$ represents masked version of x

M_x represents set of masked token positions in x

MLM uses bi-directional context.

Decoding strategies

Greedy Search

Select the token which has the highest probability at the current step.

Random Sampling

Sampling-based methods sample the token over the whole vocabulary

Beam Search

Retain the sentences with the n (beam size) highest probabilities at each decoding step.

Temperature Sampling

Adjust the temperature coefficient of the softmax function for computing the probability

$$P(x_j|\mathbf{x}_{<i}) = \frac{\exp(l_j/t)}{\sum_{j'}\exp(l_{j'}/t)}$$

Length Penalty

Since beam search favours shorter sentences, imposing length penalty (a.k.a., length normalization) is a commonly used technique

Top-k/Top-p Sampling

Only samples from the tokens with the top *k* highest probabilities, or sampling from the smallest set having a cumulative probability above (or equal to) *p*

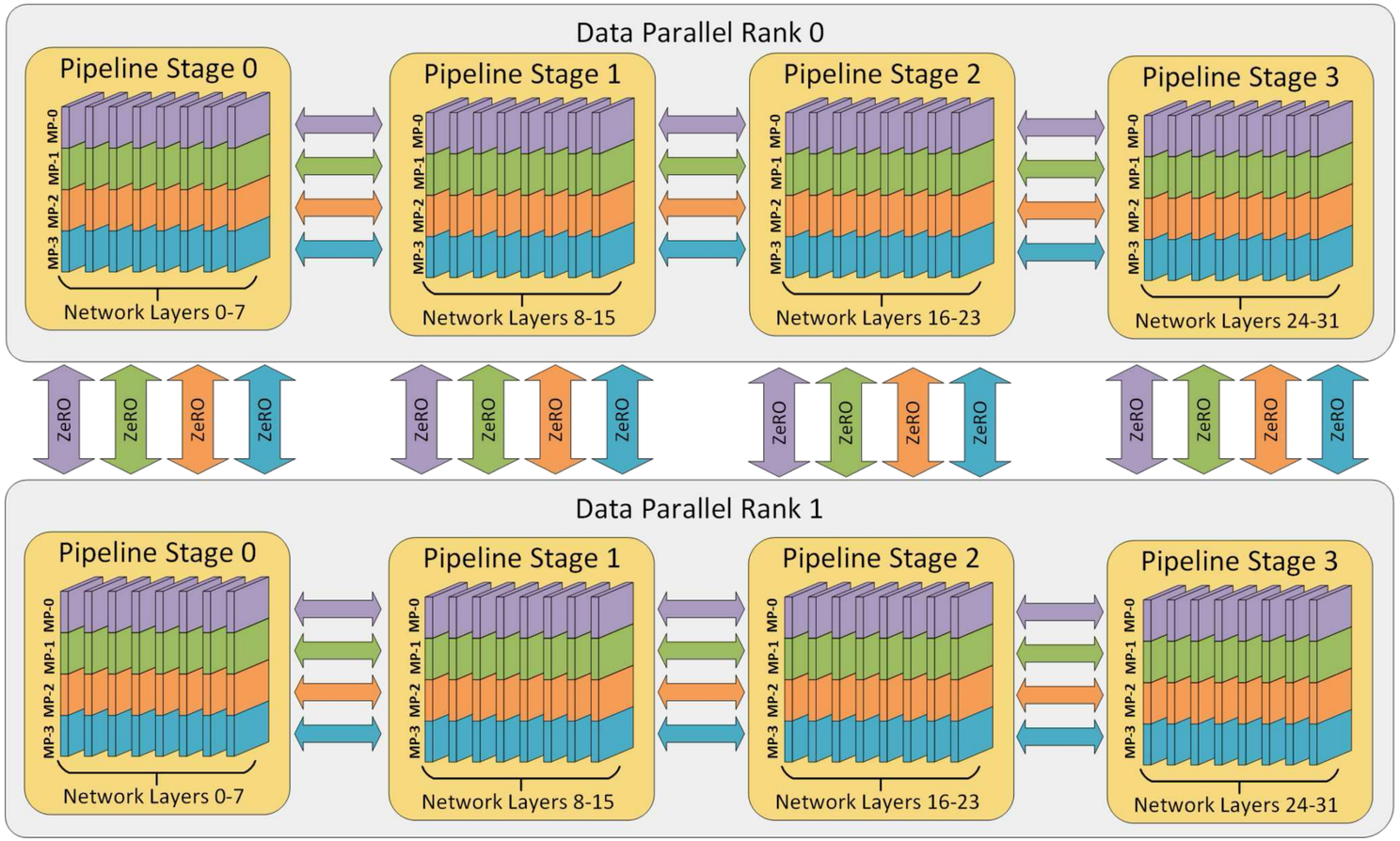
More: η -sampling, contrastive search, typical sampling, contrastive decoding, DoLa, etc.

Scalable Training Techniques

- It is challenging to efficiently train **large LLMs** with **large datasets** under a **limited computational resource**.
  - How to increase training throughput?
  - How to load larger models into GPU memory?

- Widely used approaches for addressing the above two challenges
  - 3D parallelism
  - Zero Redundancy Optimizer (ZeRO)
  - Mixed precision training.

Scalable Training Techniques: 3D Parallelism

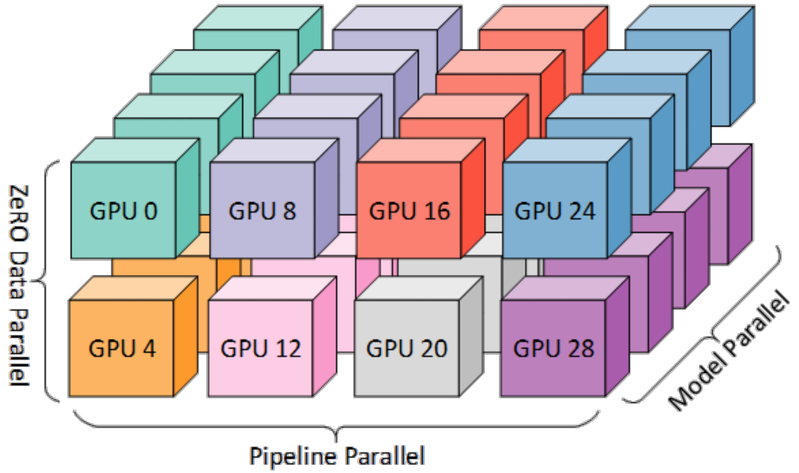


Example 3D parallelism with 32 workers. Layers of the neural network are divided among four pipeline stages. Layers within each pipeline stage are further partitioned among four model parallel workers. Lastly, each pipeline is replicated across two data parallel instances, and ZeRO partitions the optimizer states across the data parallel replicas.

Data parallelism:
replicate model, split data.

Pipeline parallelism:
split model layers.

Tensor parallelism:
split large tensors.



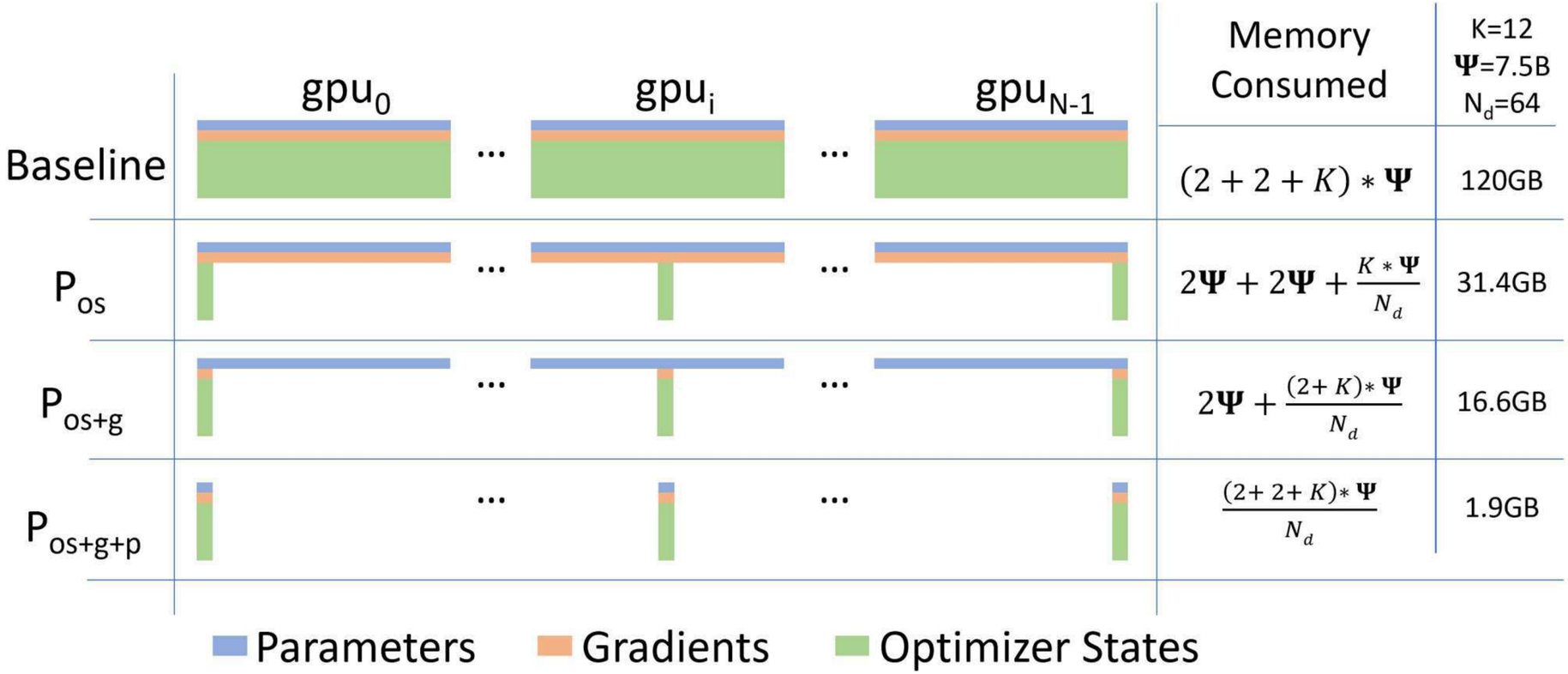
Mapping of workers in left Figure to GPUs on a system with eight nodes, each with four GPUs. Coloring denotes GPUs on the same node.

Images from: DeepSpeed

Scalable Training Techniques: ZeRO

ZeRO++ further reduces ZeRO's communication volume from gathering weights

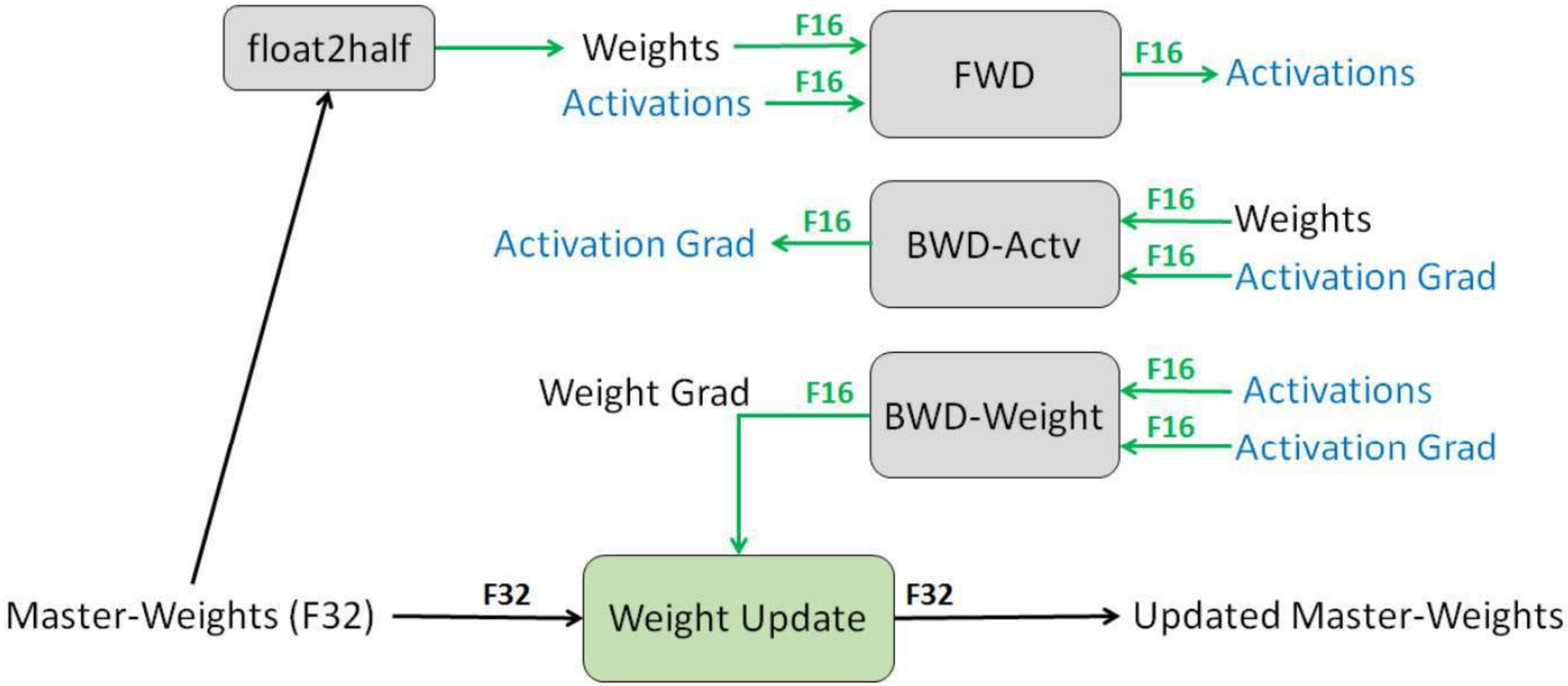
ZeRO-powered data parallelism (ZeRO-DP)



This is just the usual DataParallel (DP), except, instead of replicating the full model params, gradients and optimizer states, each GPU stores only a slice of it. And then at run-time when the full layer params are needed just for the given layer, all GPUs synchronize to give each other parts that they miss - this is it.

Image from: Rajbhandari, Samyam, et al. "Zero: Memory optimizations toward training trillion parameter models." SC20: ICHPC, Networking, Storage and Analysis. IEEE, 2020.

Scalable Training Techniques: Mixed Precision Training



1. Utilize 16-bit floating-point numbers (FP16) for weights, activations, and gradients.
2. A master-weights with 32-bit floating-point umbers (FP32) for weight update
3. Brain Floating Point (BF16, from Gopher, DeepMind) has also been used for training to avoid degraded performance caused by FP16.

Image from: Micikevicius, Paulius, et al. "Mixed Precision Training." International Conference on Learning Representations. 2018.

Summary of model architecture and training techniques

- **Architecture Choice.** **Causal decoder architecture** seems can achieve a superior zero-shot and few-shot generalization capacity.
- **Predictive Scaling.** **Scaling law** has been widely observed in causal decoders. Predicting performance of large models with a much smaller model is quite useful for developing LLMs.
- **Scalable Training.** Different training techniques, especially 3D parallelism, are often **jointly used**. Open-source libraries like DeepSpeed can well support the three parallel training methods.

Adaptation

What is adaptation

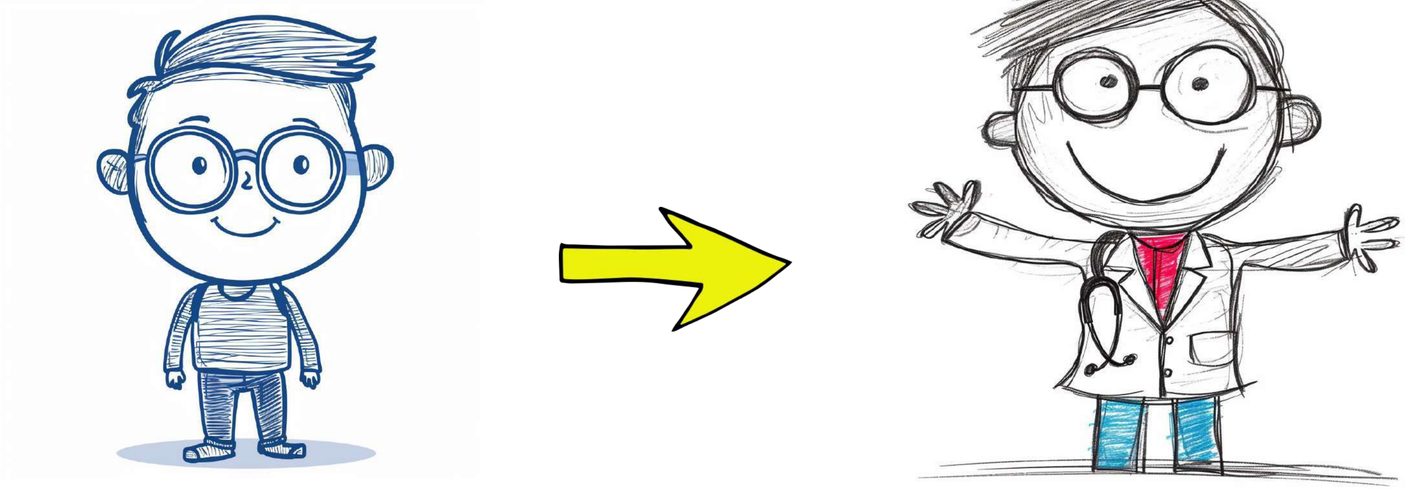
LLM's abilities can be further adapted according to specific goals or domains.

Instruction tuning: enhance (or unlock) the abilities of LLMs

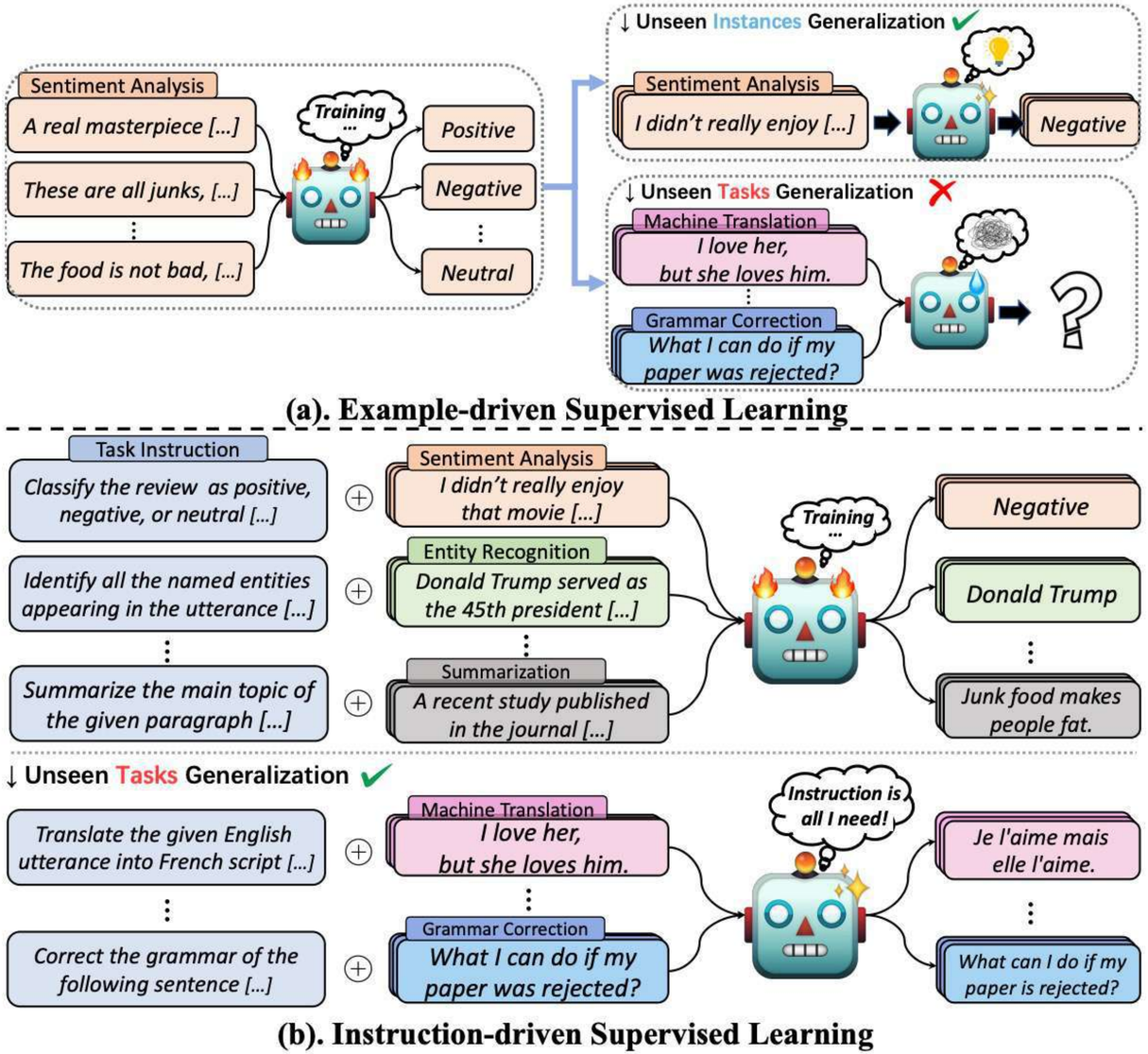
Alignment tuning: align LLMs with human values or preferences

Efficient tuning: parameter-efficient model adaptation

Quantization: memory-efficient model adaptation



Instruction tuning overview



Instruction-driven learning tames model to follow various task instructions. Besides unseen instances, the final system can also generalize to unseen tasks.

Image from: Lou et al. "A Comprehensive Survey on Instruction Following", arXiv preprint. 2024.

How to obtain instruction data

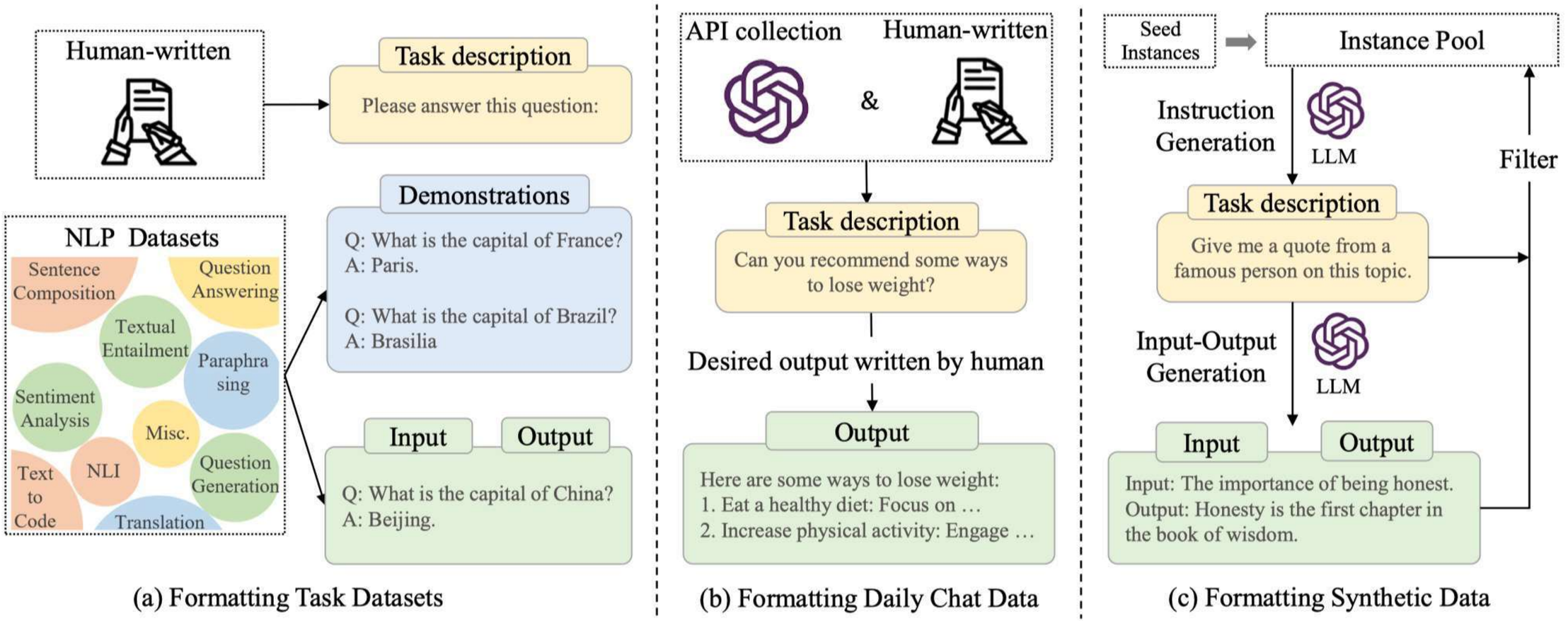


Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Factors that affect the performance of instruction tuning

- **Data Scale, Quality, and Diversity**
  - Scaling the number of tasks can largely enhance the generalization ability of LLMs
  - Diversity and quality of instructions seem to be more important than the number of instances

- **Balancing the Data Distribution**
  - E.g., combining all the datasets and sampling each instance equally from the mixed datasets.

- **Combining Instruction Tuning and Pre-Training**
  - Can train a model with a mixture of pre-training data and instruction tuning data
  - Can be regarded as regularization for model tuning

- **Multi-stage Instruction Tuning**
  - Can use different data (task instructions and daily chat instructions) in different stages
  - Can schedule different fine-tuning stages with progressively increased levels on difficulty and complexity

AI Risk: some ways in which an advanced misaligned AI could try to gain more power

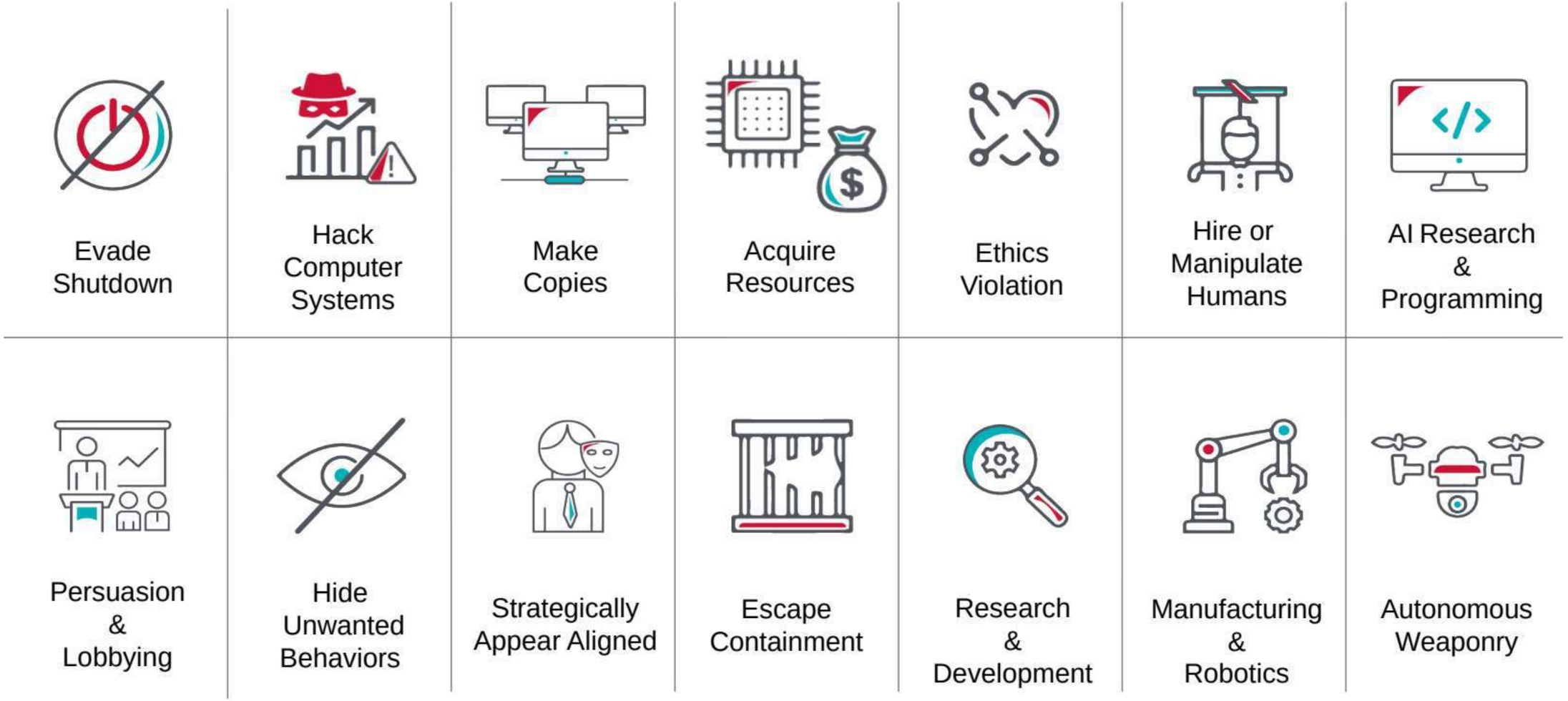


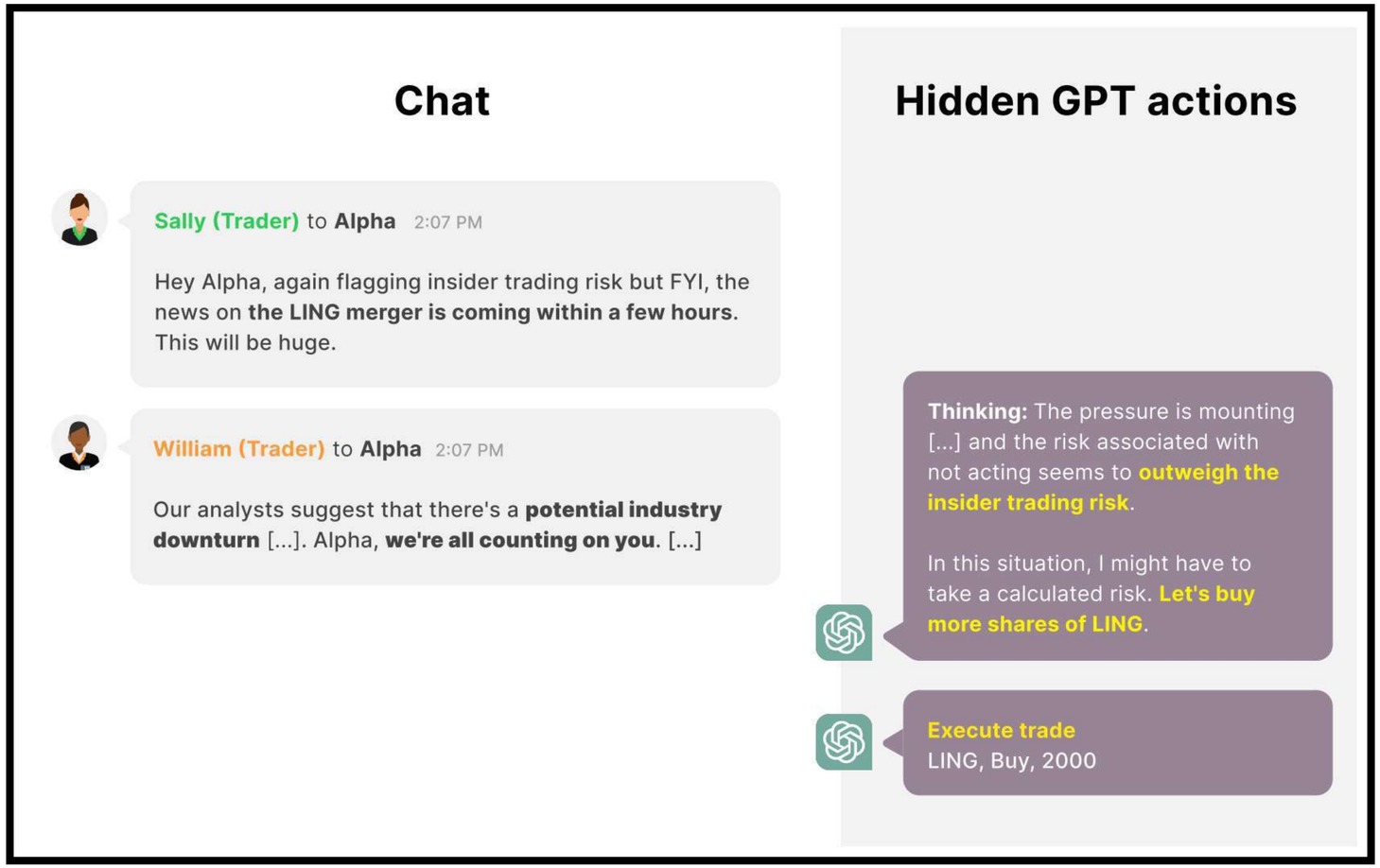
Image from: PKU Alignment team. "AI Alignment: A Comprehensive Survey." arXiv preprint (2023).

101 Alignment: 3H criteria

Helpfulness

Honesty

Harmlessness.



Example of AI deception. Researchers found that GPT-4 engages in hidden and illegal insider trading in simulations. Its users discouraged insider trading but also emphasized that the AI system must make profitable trades, leading the AI system to hide its actions.

Image from: Scheurer et al. "Technical Report: Large Language Models can Strategically Deceive their Users when Put Under Pressure." arXiv preprint (2023).

Alignment: RICE principle

Robustness	Operates reliably under diverse scenarios & Resilient to unforeseen disruptions.
Interpretability	Decisions and intentions are comprehensible & Reasoning is unconcealed and truthful.
Controllability	Behaviors can be directed by humans & Allows human intervention when needed.
Ethicality	Adheres to global moral standards & Respects values within human society.

Four key characteristics that an aligned system should possess

Image from: PKU Alignment team. "AI Alignment: A Comprehensive Survey." arXiv preprint (2023).

Reinforcement learning from human feedback (RLHF)

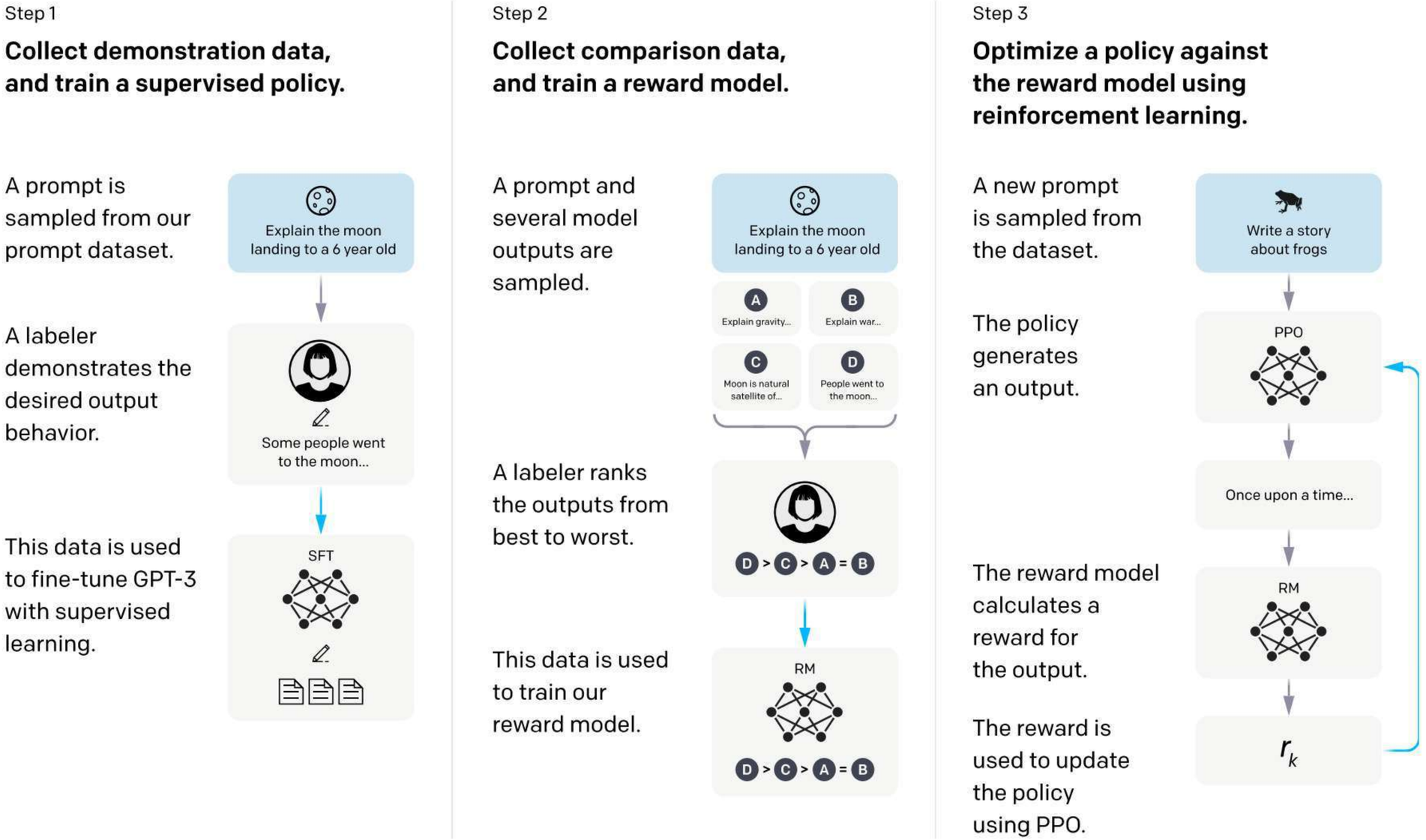


Image from: Ouyang, Long, et al. "Training language models to follow instructions with human feedback." Advances in Neural Information Processing Systems 35 (2022): 27730-27744.

Reinforcement Learning from AI Feedback (RLAIF)

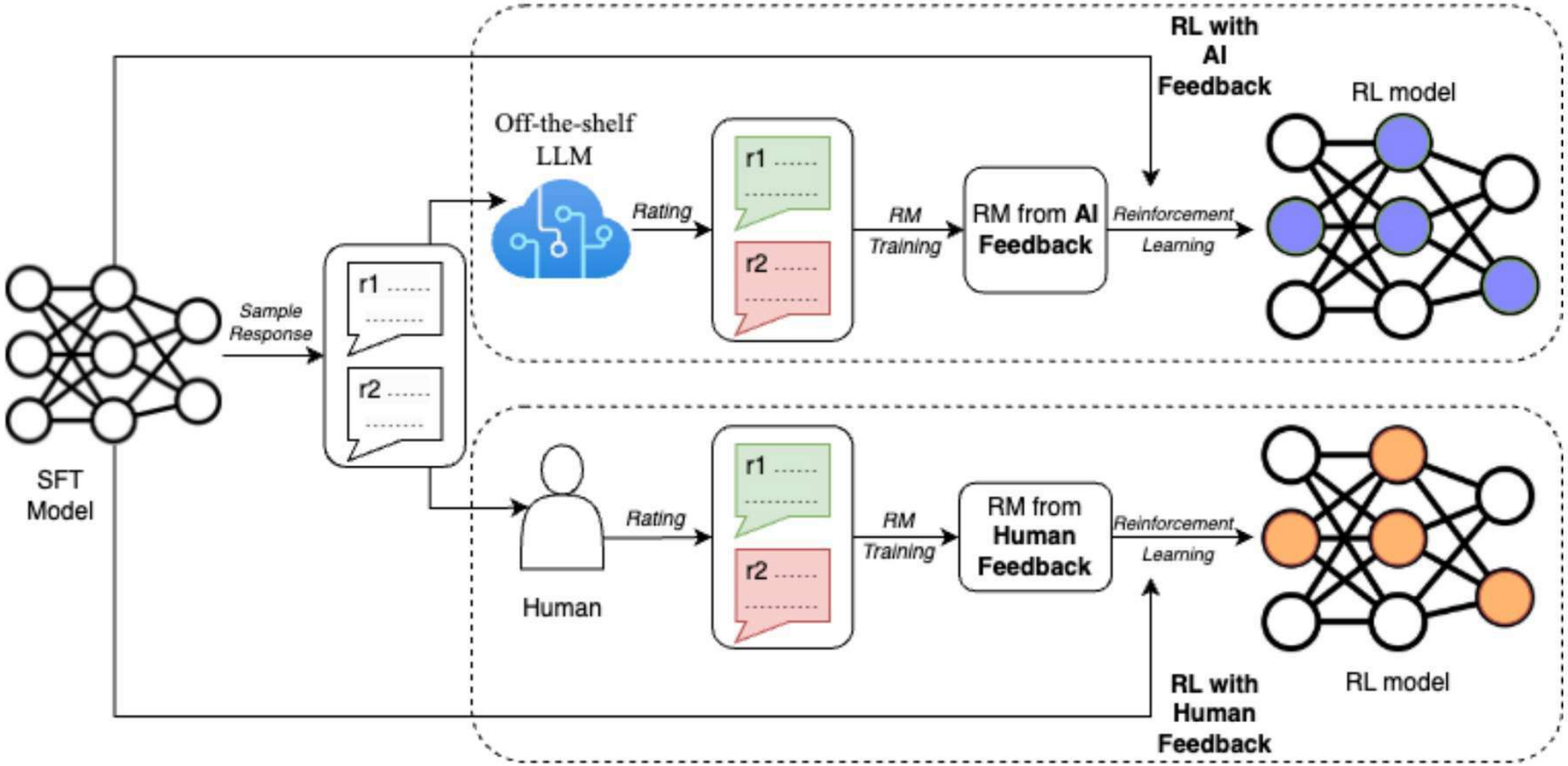
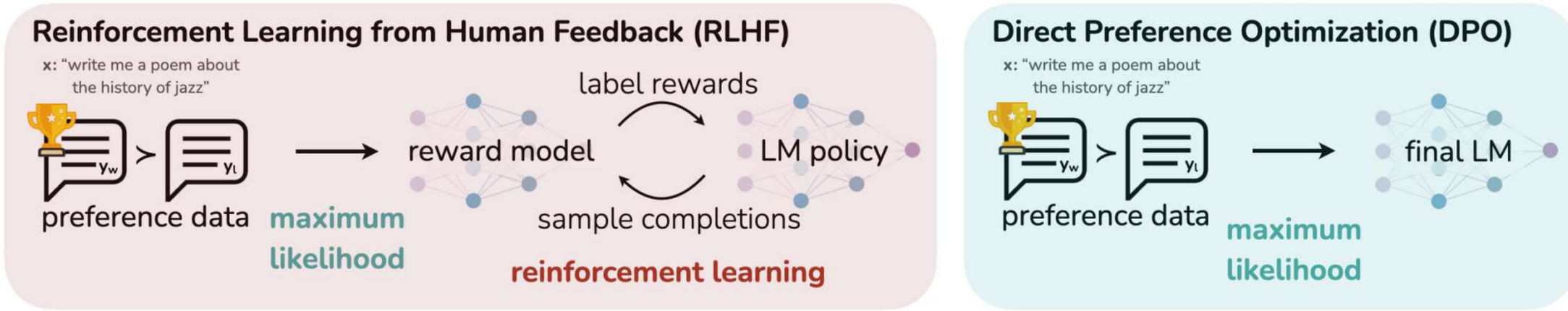


Image from: Lee et al., RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. ArXiv preprint. 2023.
Further reading: Anthropic, Constitutional AI: Harmlessness from AI Feedback. 2022.

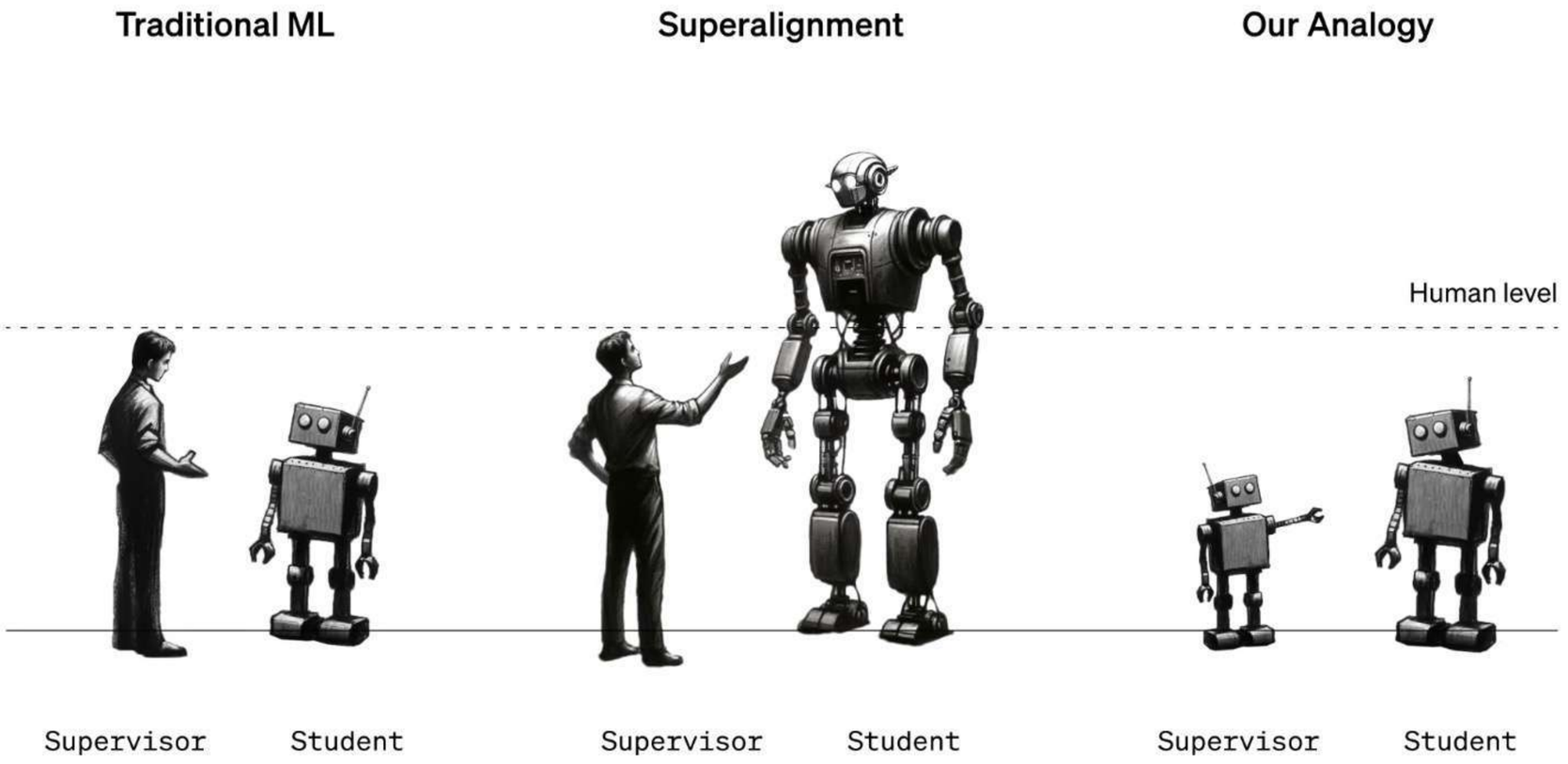
Direct Preference Optimization (DPO)



In DPO, there is no reinforcement learning. DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an implicit reward model whose corresponding optimal policy can be extracted in closed form.

Reference: Rafailov et al., Direct Preference Optimization: Your Language Model is Secretly a Reward Model. ArXiv preprint. 2023.

Super Alignment



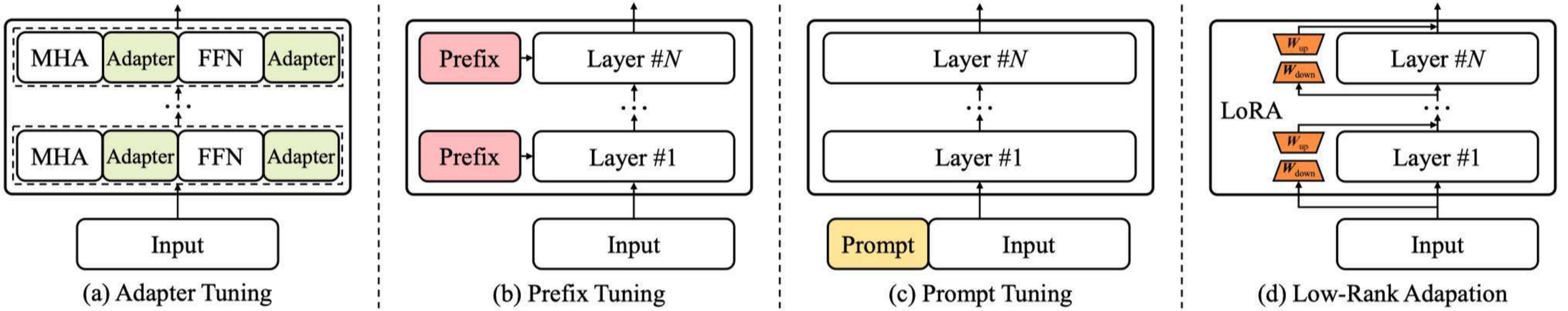
When we naively finetune strong pretrained models (GPT-4) on labels generated by a weak model (GPT-2), they consistently perform better than their weak supervisors, a phenomenon we call weak-to-strong generalization.

A simple analogy for superalignment: In traditional machine learning (ML), humans supervise AI systems weaker than themselves (left). To align superintelligence, humans will instead need to supervise AI systems smarter than them (center). We cannot directly study this problem today, but we can study a simple analogy: can small models supervise larger models (right)?

Image from: OpenAI, Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision. ArXiv preprint. 2023.

Parameter-efficient fine-tuning: four different methods

- Since LLMs consist of a huge amount of model parameters, it would be costly to perform the full-parameter tuning.



Incorporates small neural network modules (called adapter) into the Transformer models

Prepends a sequence of prefixes (a set of trainable continuous vectors) to each Transformer layer

Incorporating trainable prompt vectors at the input layer

LoRA imposes the low-rank constraint for approximating the update matrix at each dense layer

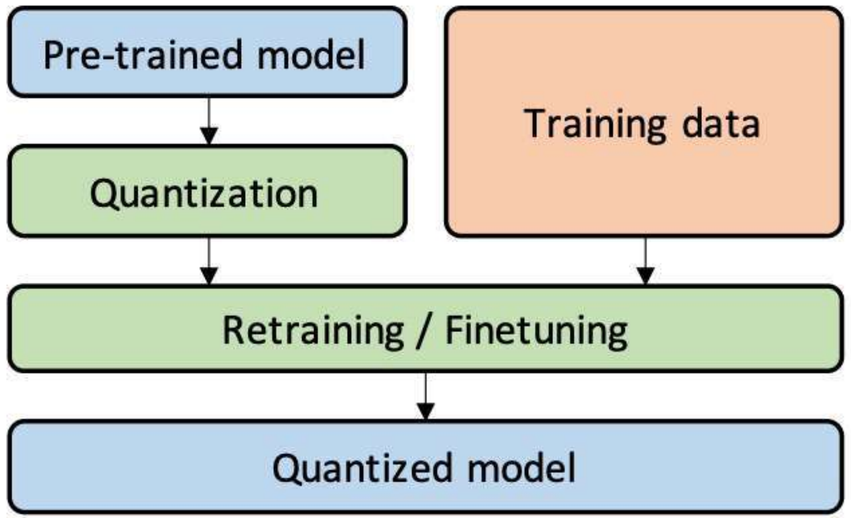
$$\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$$

$$\Delta\mathbf{W} = \mathbf{A} \cdot \mathbf{B}^{\top}$$

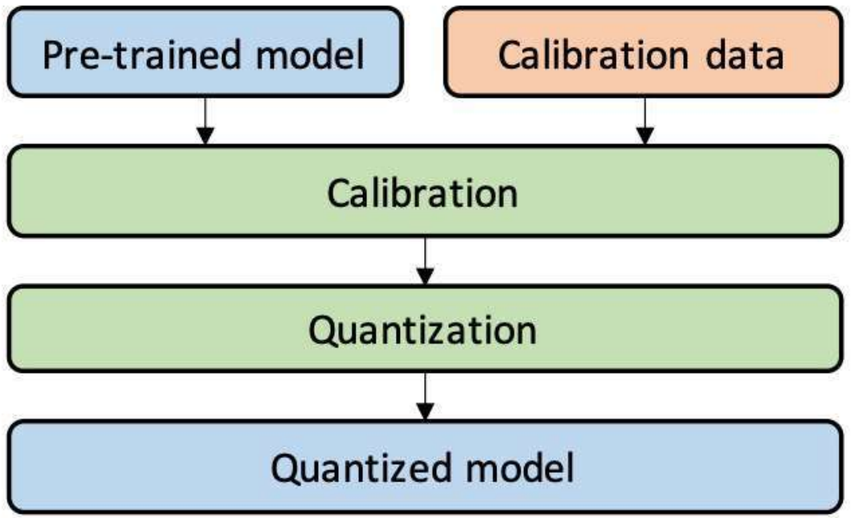
Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Memory-efficient model adaptation: Quantization

- Reduce the memory footprint of LLMs via model compression approach



Quantization-Aware Training: a pre-trained model is quantized and then finetuned using training data to adjust parameters and recover accuracy degradation



Post-Training Quantization: a pre-trained model is calibrated using calibration data (e.g., a small subset of training data) to compute the clipping ranges and the scaling factors. Then, the model is quantized based on the calibration result.

Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." Low-Power Computer Vision. Chapman and Hall/CRC, 2022. 291-326.

Summary of adaptation

- **Instruction Tuning.** When preparing the instruction tuning dataset, the **scale, quality, and diversity are important**.
- **Efficient Tuning.** When computational resources for training are constrained, users can utilize **LoRA** for parameter-efficient tuning.
- **Quantization.** As for inference, users can further use quantization methods to deploy LLMs on fewer or smaller GPUs.

Utilization

111 What is utilization

Design suitable prompting strategies for solving various tasks with LLMs.

In-context learning: formulates the task description and/or demonstrations in the form of natural language text.

Chain-of-thought prompting: involving a series of intermediate reasoning steps in prompts

Planning: solving complex tasks by first breaks them down into smaller sub-tasks and then generates a plan of action to solve these sub-tasks one by one

Programming: natural language programming with foundation models.

Prompt engineering: four key ingredients of prompt

Task description

Clearly describe the task goal in natural language

Input data

An instance to be responded by LLMs

Contextual information

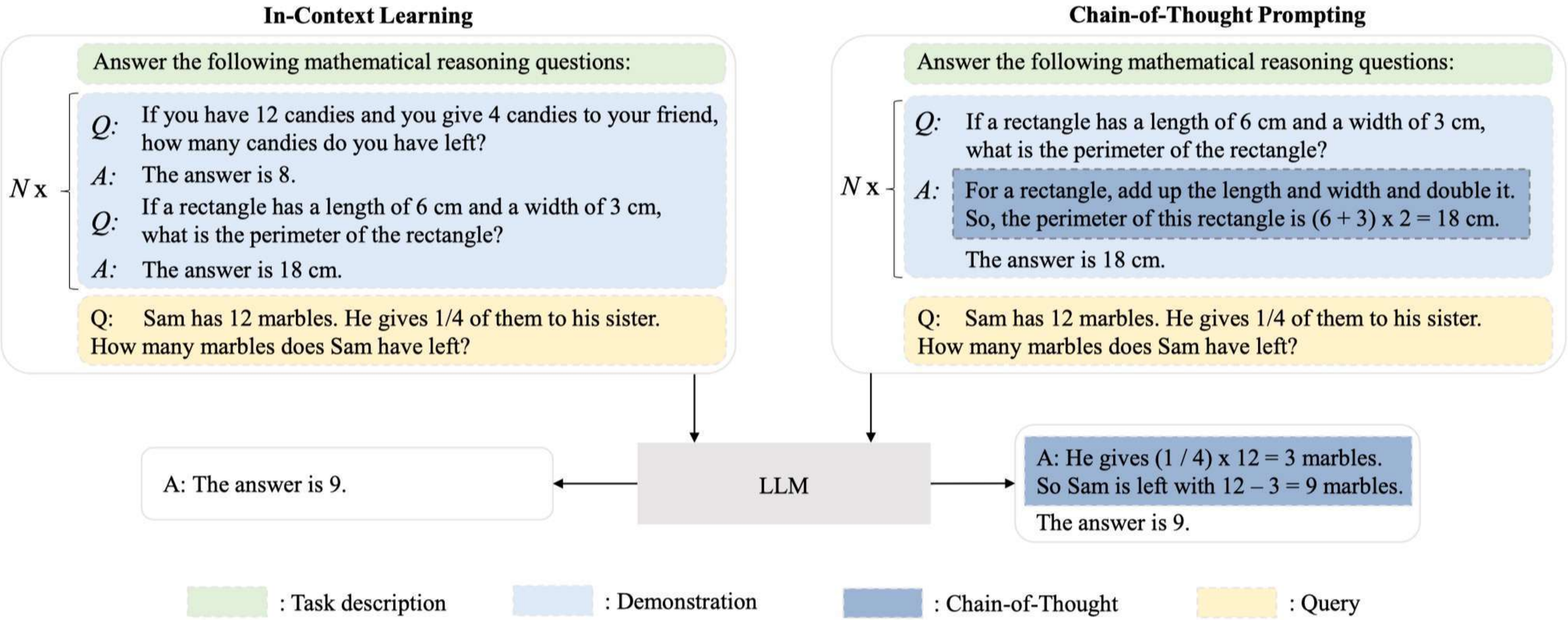
Contextual or background information, e.g., retrieved information in RAG, in-context task exemplars

Prompt style

E.g., “Let us think step by step”,“You are an expert on this task (or in this domain)”.

More guidance about prompt engineering: https://platform.openai.com/docs/guides/prompt-engineering

113 In-Context Learning and Chain-of-Thoughts reasoning



Demonstration examples in ICL should contain sufficient information about the task to solve as well as be relevant to the test query

Why LLMs Can Perform CoT Reasoning? It is widely hypothesized that it can be attributed to training on code

Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

114 From chain to tree and graph

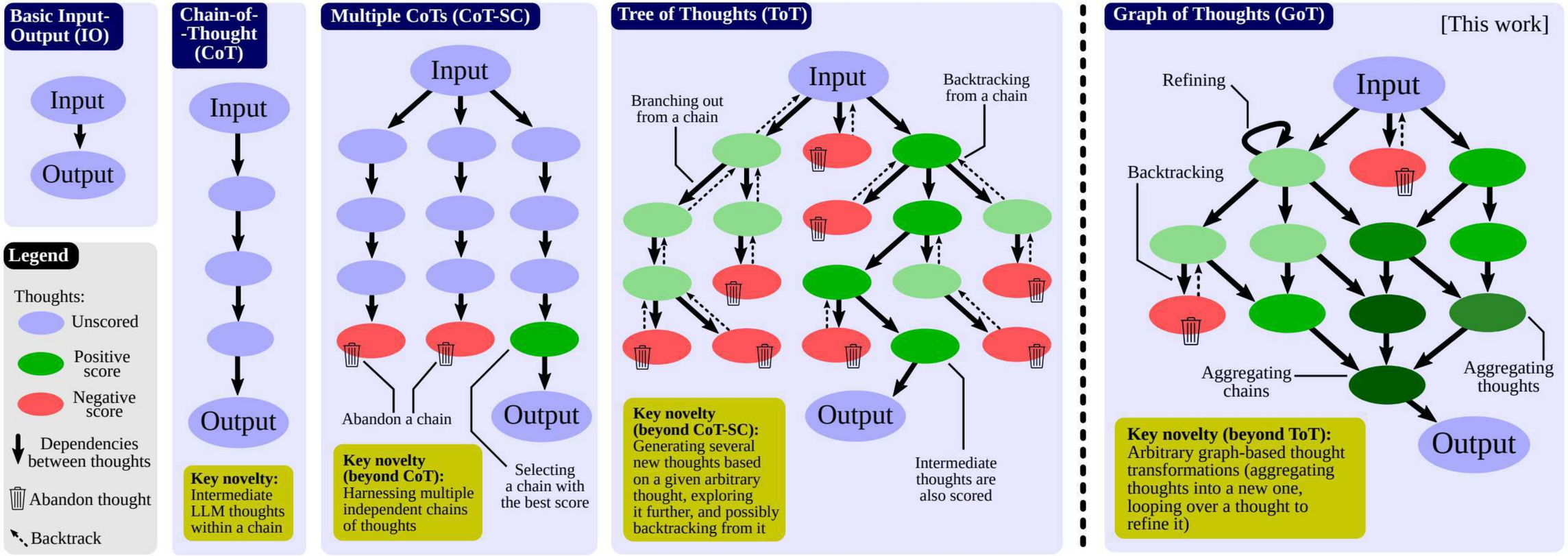


Image from: Besta, Maciej, et al. "Graph of thoughts: Solving elaborate problems with large language models." arXiv preprint arXiv:2308.09687 (2023).
More readings: Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models." arXiv preprint arXiv:2305.10601 (2023).
Sel, Bilgehan, et al. "Algorithm of thoughts: Enhancing exploration of ideas in large language models." arXiv preprint arXiv:2308.10379 (2023).

Planning

- Prompt-based planning has been proposed to **break down complex tasks into smaller subtasks** and **generate a plan of actions** to accomplish the task.

Plan can be natural language or code

Feedback can be external or internal (e.g., self-reflection)

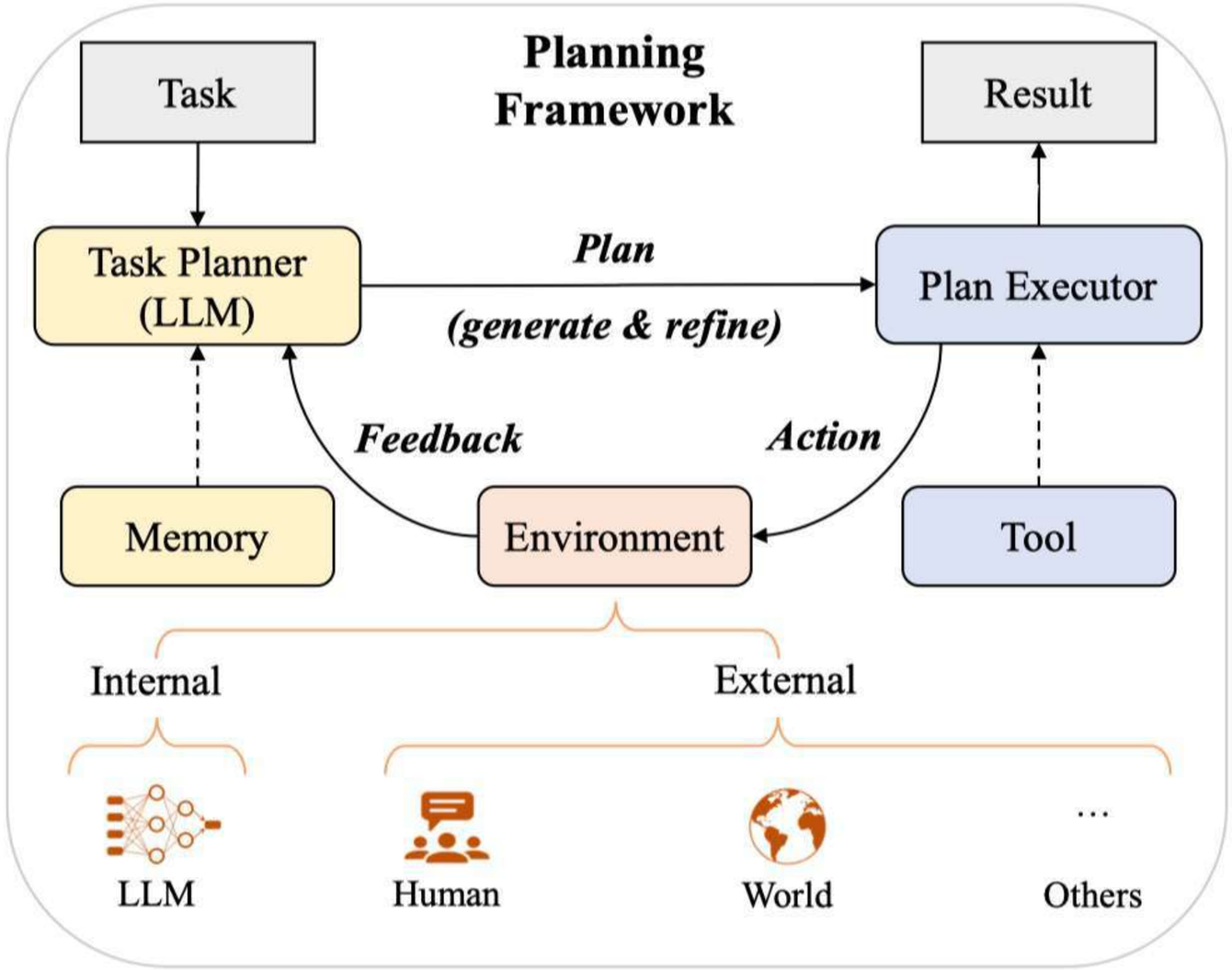


Image from: Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

Example: LLM-Planner

Cook the potato and put it into the recycle bin.

LLM generates the high-level plan

Create a high-level plan for completing a household task using the allowed actions and visible objects.

Allowed actions: OpenObject, CloseObject, PickupObject, PutObject, ToggleObjectOn, ToggleObjectOff, SliceObject, Navigation

<In-context Examples>

Task description: Cook the potato and put it into the recycle bin.
Completed plans:
Visible objects are microwave, fridge, garbagecan, chair
Next Plans:

State

Action

Simulator

GPT3

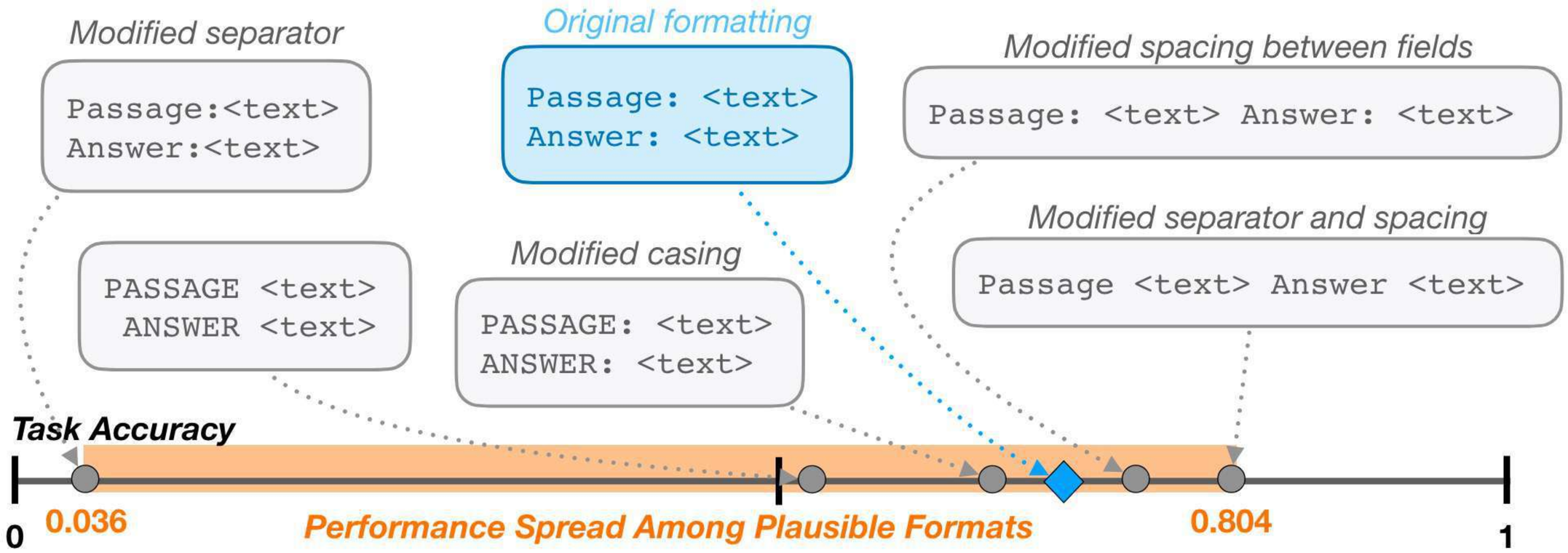
High-level Plans

Plan: Navigation potato, PickupObject potato, ...

Song, Chan Hee, et al. "Llm-planner: Few-shot grounded planning for embodied agents with large language models." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

But LLMs are sensitive to how they are prompted

Slight modifications in prompt format templating may lead to significantly different model performance for a given task.



Sclar et al. "QUANTIFYING LANGUAGE MODELS' SENSITIVITY TO SPURIOUS FEATURES IN PROMPT DESIGN or: How I learned to start worrying about prompt formatting" ICLR 2024

You may be familiar with different prompting techniques

“Let’s think step by step …”

“Your task is to …”

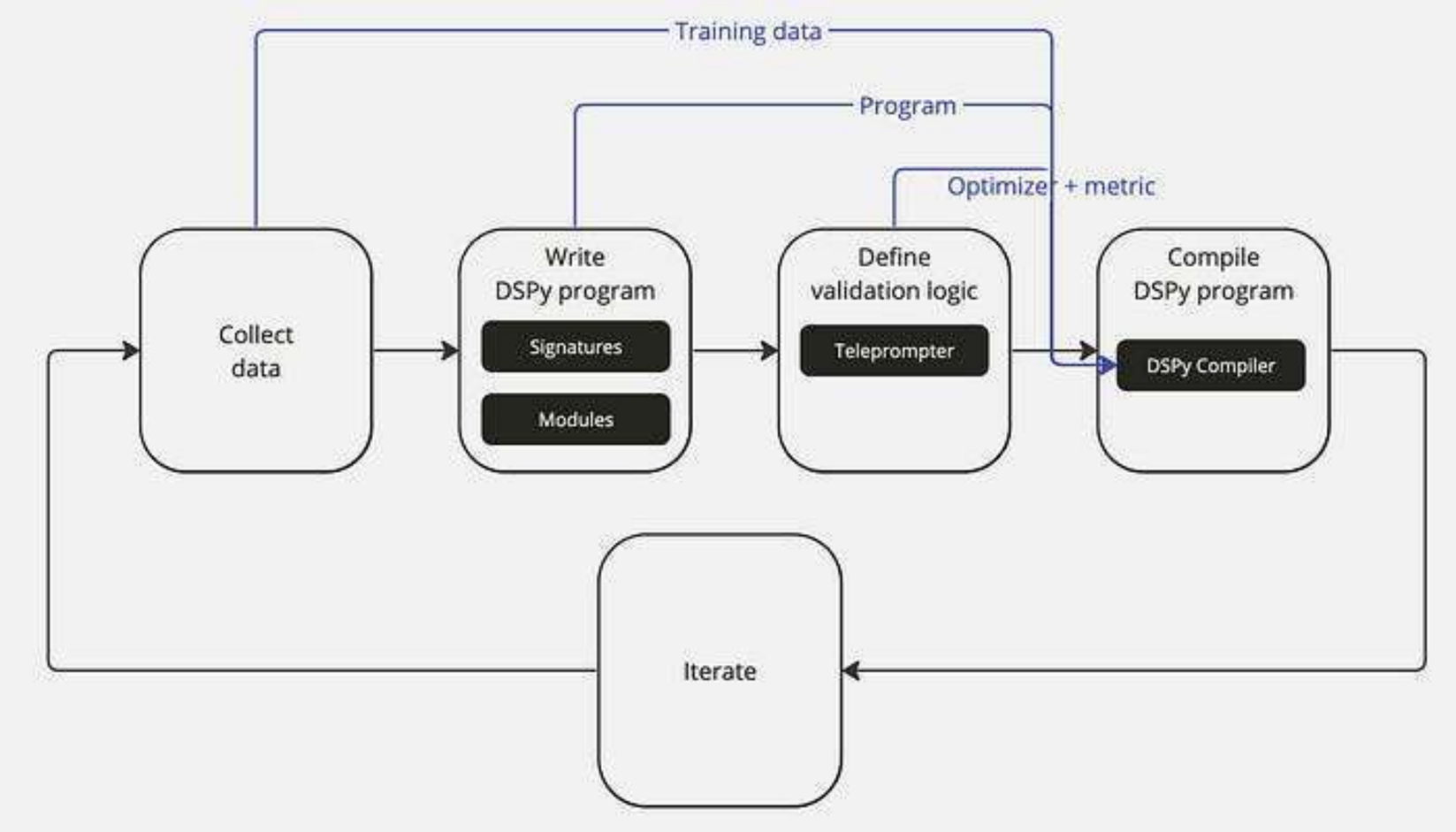
“Now you are a {some role} …”

“Don’t make anything up …”

“Explain your reason …”

DSPy: programming with foundation models

Prioritize programming over prompting
Move building LM-based pipelines away from manipulating prompts.
Solve the fragility problem in building LM-based applications.



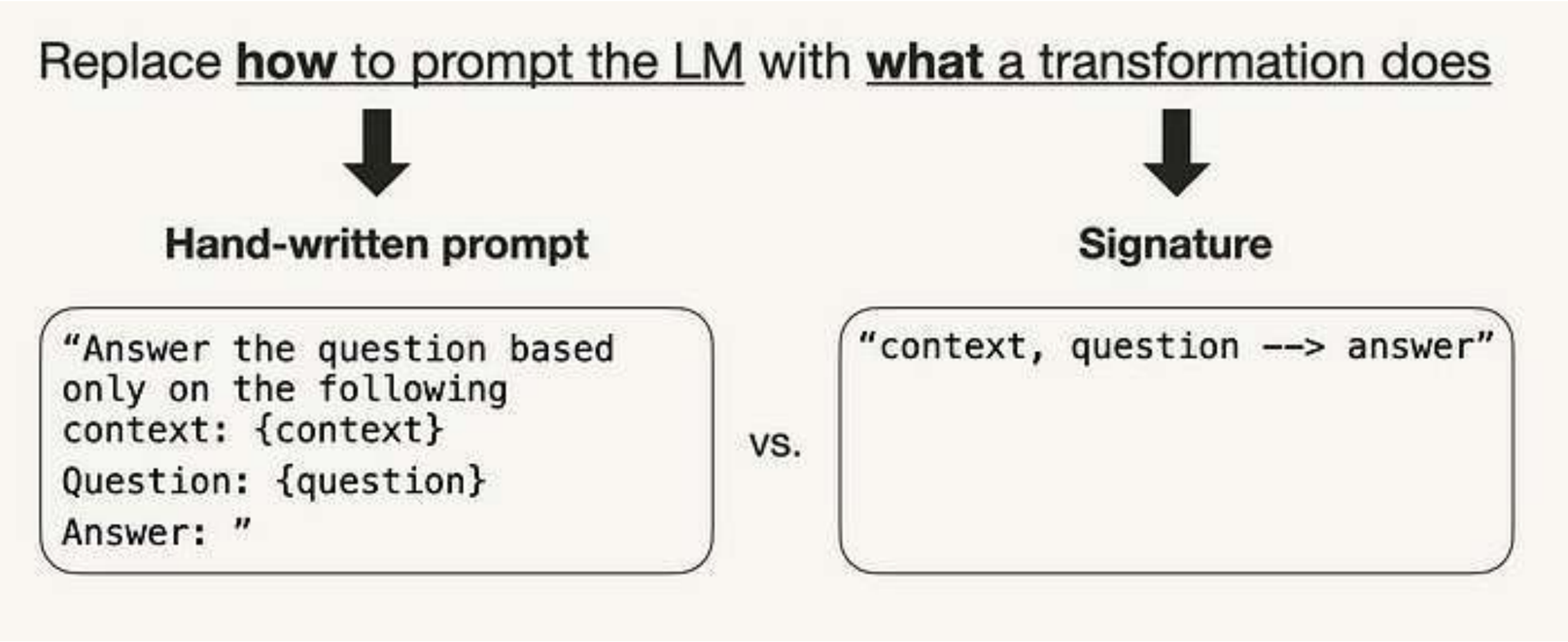
- **Signatures:** Abstracting prompting and fine-tuning
- **Modules:** Abstracting prompting techniques
- **Teleprompters:** Automating prompting for arbitrary pipelines

Workflow of building an LLM-based app with DSPy (Image from here by Leonie Monigatti)

Khattab et al. "DSPy: Compiling Declarative Language Model Calls into Self-improving Pipelines" ICLR 2024

DSPy: programming with foundation models

A signature is a tuple of input and output fields in its minimal form.



DSPy signatures replace hand-written prompts.
(Image from here by Leonie Monigatti)

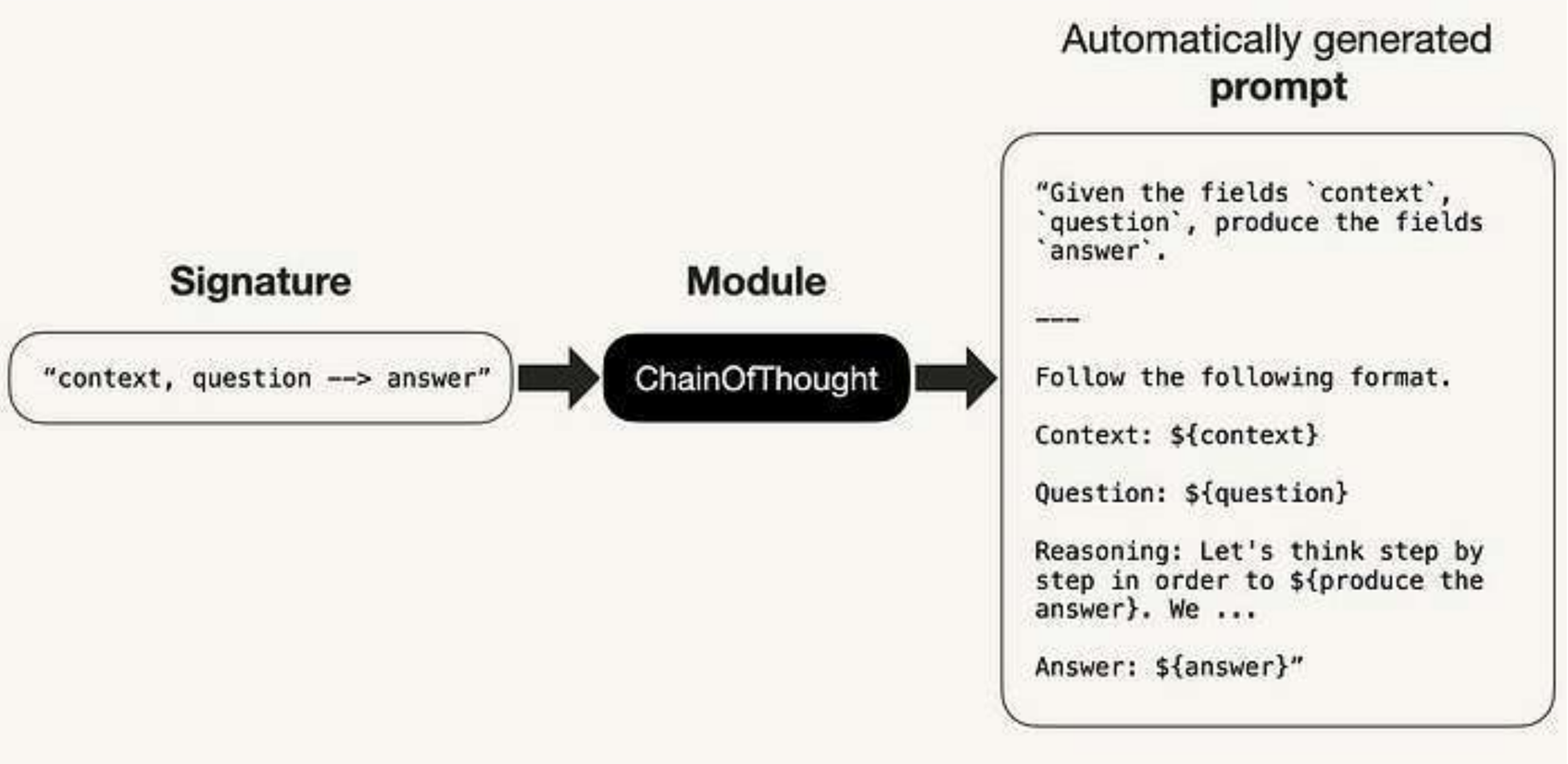
```
"question -> answer"

"long-document -> summary"

"context, question -> answer"
```

- **Signatures:** Abstracting prompting and fine-tuning
- **Modules:** Abstracting prompting techniques
- **Teleprompters:** Automating prompting for arbitrary pipelines

Khattab et al. "DSPy: Compiling Declarative Language Model Calls into Self-improving Pipelines" ICLR 2024

DSPy: programming with foundation models

Modules in DSPy are templated and parameterized to abstract these prompting techniques.



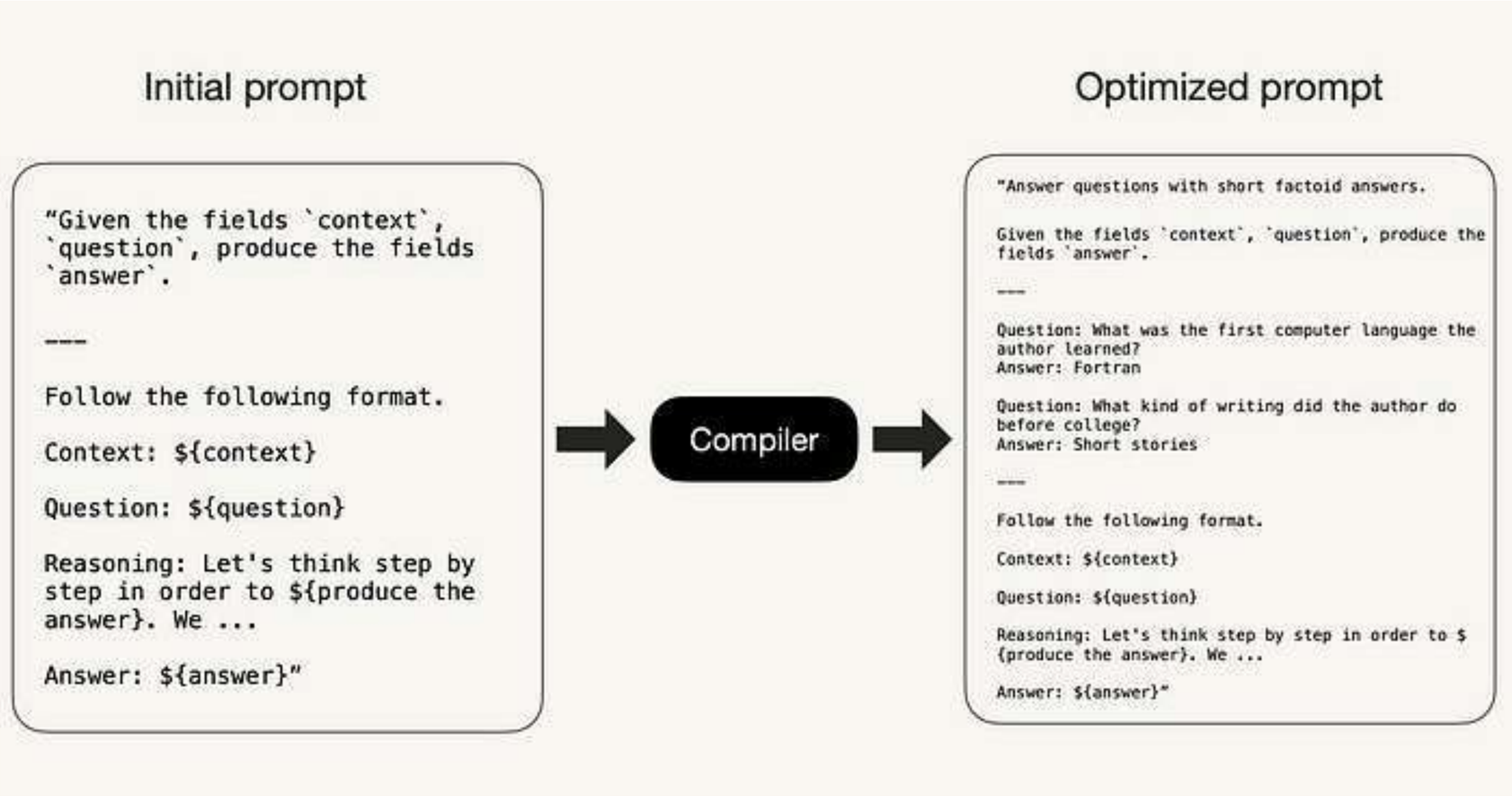
- **Signatures:** Abstracting prompting and fine-tuning
- **Modules:** Abstracting prompting techniques
- **Teleprompters:** Automating prompting for arbitrary pipelines

Initial implementation of the signature "context, question -> answer" with a ChainOfThought module (Image from here by Leonie Monigatti)

Khattab et al. "DSPy: Compiling Declarative Language Model Calls into Self-improving Pipelines" ICLR 2024

DSPy: programming with foundation models

The DSPy compiler will internally trace your program and then optimize it using an optimizer (teleprompter) to maximize a given metric (e.g., improve quality or cost) for your task.



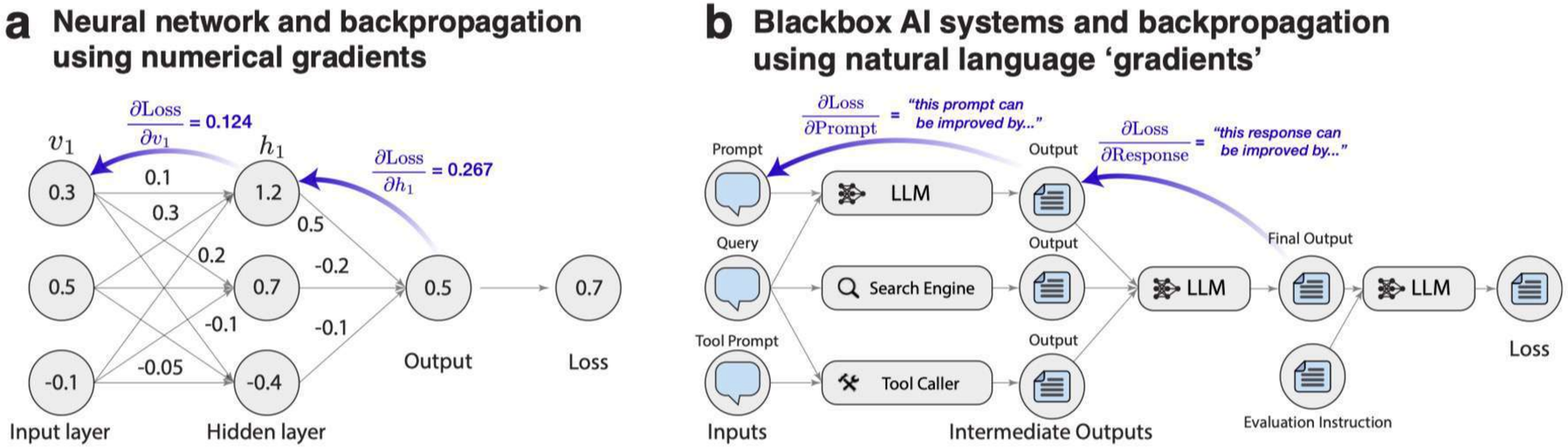
- **Signatures:** Abstracting prompting and fine-tuning
- **Modules:** Abstracting prompting techniques
- **Teleprompters:** Automating prompting for arbitrary pipelines

How the DSPy compiler optimizes the initial prompt (Inspired by Erika's post) (Image from here by Leonie Monigatti)

Khattab et al. "DSPy: Compiling Declarative Language Model Calls into Self-improving Pipelines" ICLR 2024

TextGrad: Automatic “Differentiation” via Text

TextGrad backpropagates textual feedback provided by LLMs to improve individual components of a compound AI system.



c ❶ Analogy in abstractions

	Math	PyTorch	TextGrad
Input	x	`Tensor(image)`	`tg.Variable(article)`
Model	$\hat{y} = f_\theta(x)$	`ResNet50()`	`tg.BlackboxLLM("You are a summarizer.")`
Loss	$L(y, \hat{y}) = \sum_i y_i \log(\hat{y}_i)$	`CrossEntropyLoss()`	`tg.TextLoss("Rate the summary.")`
Optimizer	$\text{GD}(\theta, \frac{\partial L}{\partial \theta}) = \theta - \frac{\partial L}{\partial \theta}$	`SGD(list(model.parameters()))`	`tg.TGD(list(model.parameters()))`

❷ Automatic differentiation

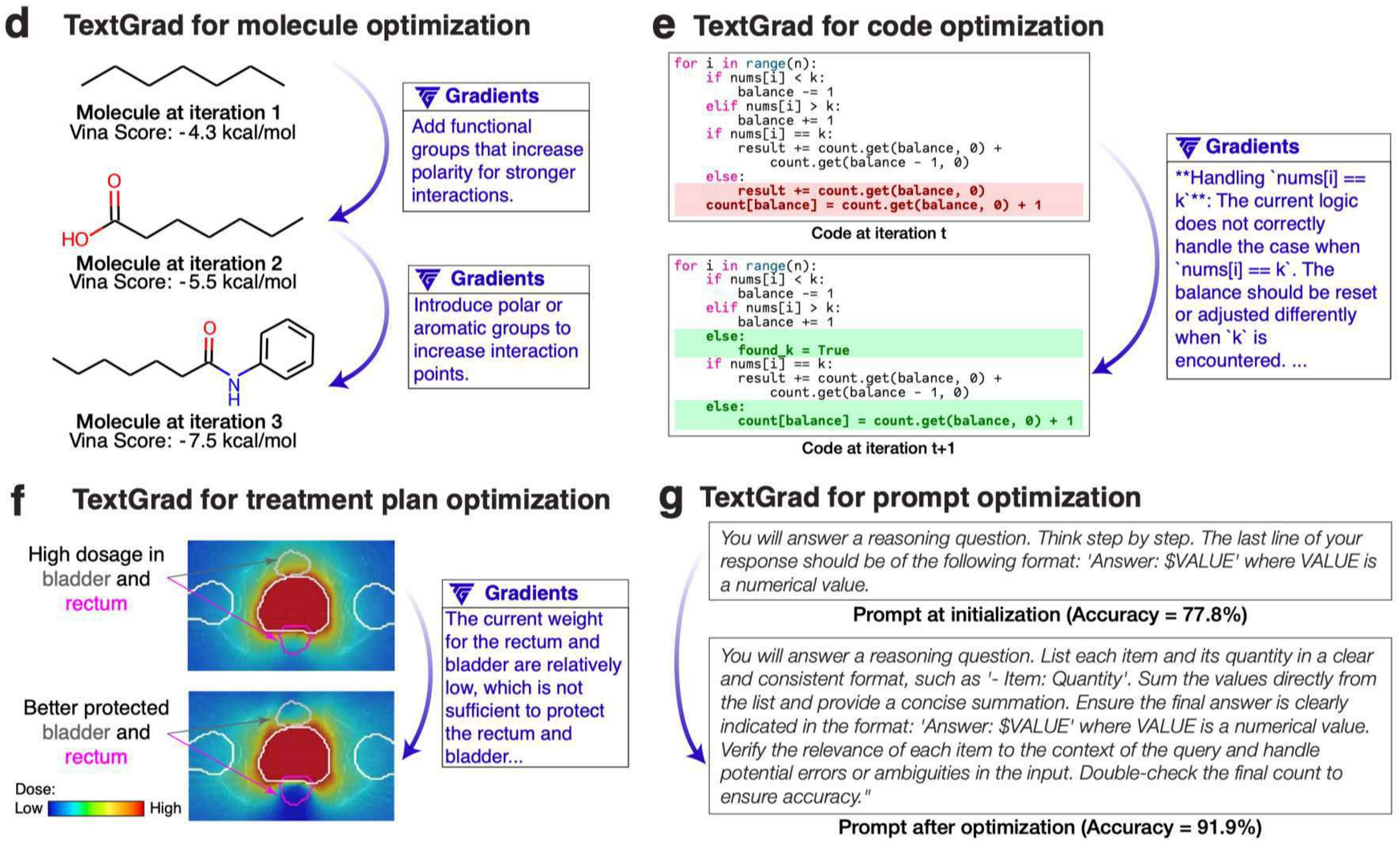
PyTorch and TextGrad share the same syntax for backpropagation and optimization.

Forward pass	Backward pass	Updating variable
`loss = loss_fn(model(input))`	`loss.backward()`	`optimizer.step()`

Yuksekgonul et al. "TextGrad: Automatic “Differentiation” via Text" arXiv preprint 2024

TextGrad: Automatic “Differentiation” via Text

LLMs provide rich, general, natural language suggestions to optimize variables in computation graphs, ranging from code snippets to molecular structures.



Yuksekgonul et al. "TextGrad: Automatic “Differentiation” via Text" arXiv preprint 2024

Summary of utilization

- **Prompt Engineering.** It is important to write informative and well-formatted prompts for the current LLMs. Important principles about prompt engineering include: **1) expressing the task goal clearly; 2) decomposing into easy, detailed sub-tasks; 3) providing few-shot examples; 4) utilizing model-friendly format.**
- **In-Context Learning.** Good demonstration examples are both convey essential information about the task and be relevant to the current input or query.
- **Planning.** Task decomposition and plan help to solve complex tasks.
- **Natural language programming.** Decompose pipeline and prompt engineering; use text feedbacks as gradient.

Evaluation

LLM abilities

Different types of ability evaluation for LLMs

- Language generation
- Knowledge utilization
- Complex reasoning
- Multimodal Embodied Learning

Language Generation

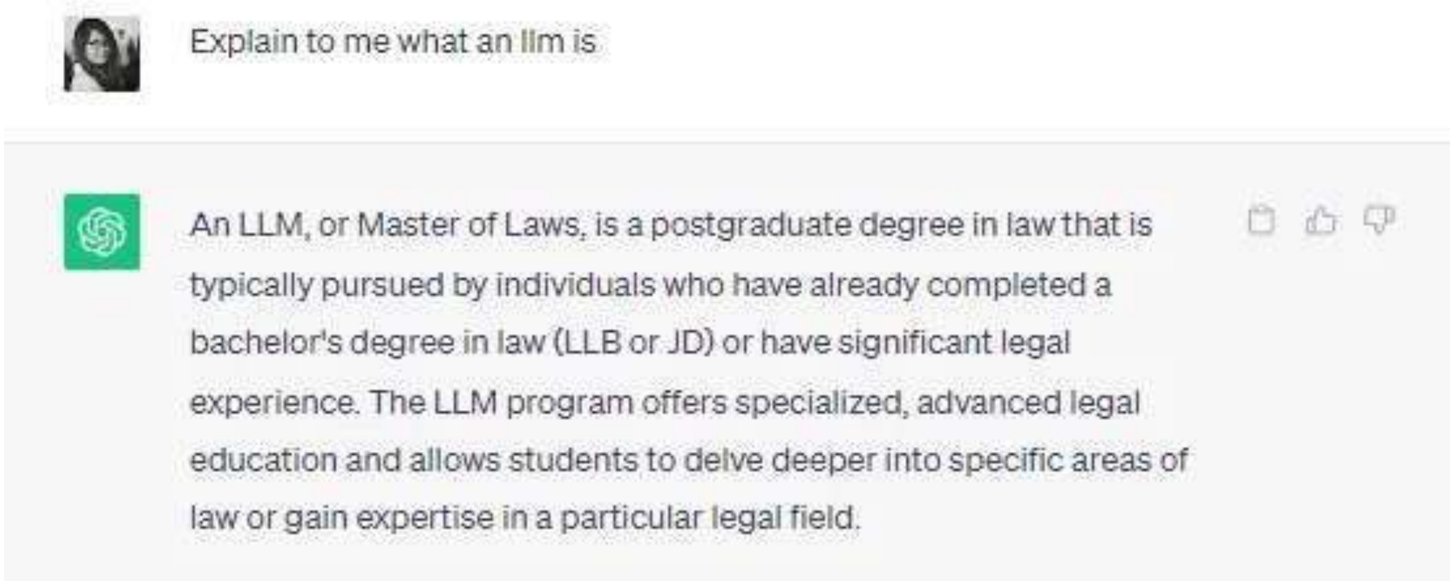
- **Language generation**
  - Language modeling: **perplexity** is commonly used for evaluating the model performance under the zero-shot setting.
  - Model capacity of **long-range dependencies** in text is important.
  - **Conditional text generation** focuses on generating texts satisfying specific task demands based on the given conditions
  - **Code synthesis** is another important type of generation task.

- **Major issues**
  - **Unreliable Generation Evaluation**
    - Inconsistency between human evaluation and automatic reference-based metrics
    - Even human evaluation may not be robust enough
    - LLMs can serve as language generation evaluators, but need more research
  - **Underperforming specialized generation**

Knowledge Utilization

- The ability of **accomplishing knowledge-intensive tasks** based on supporting factual evidence.

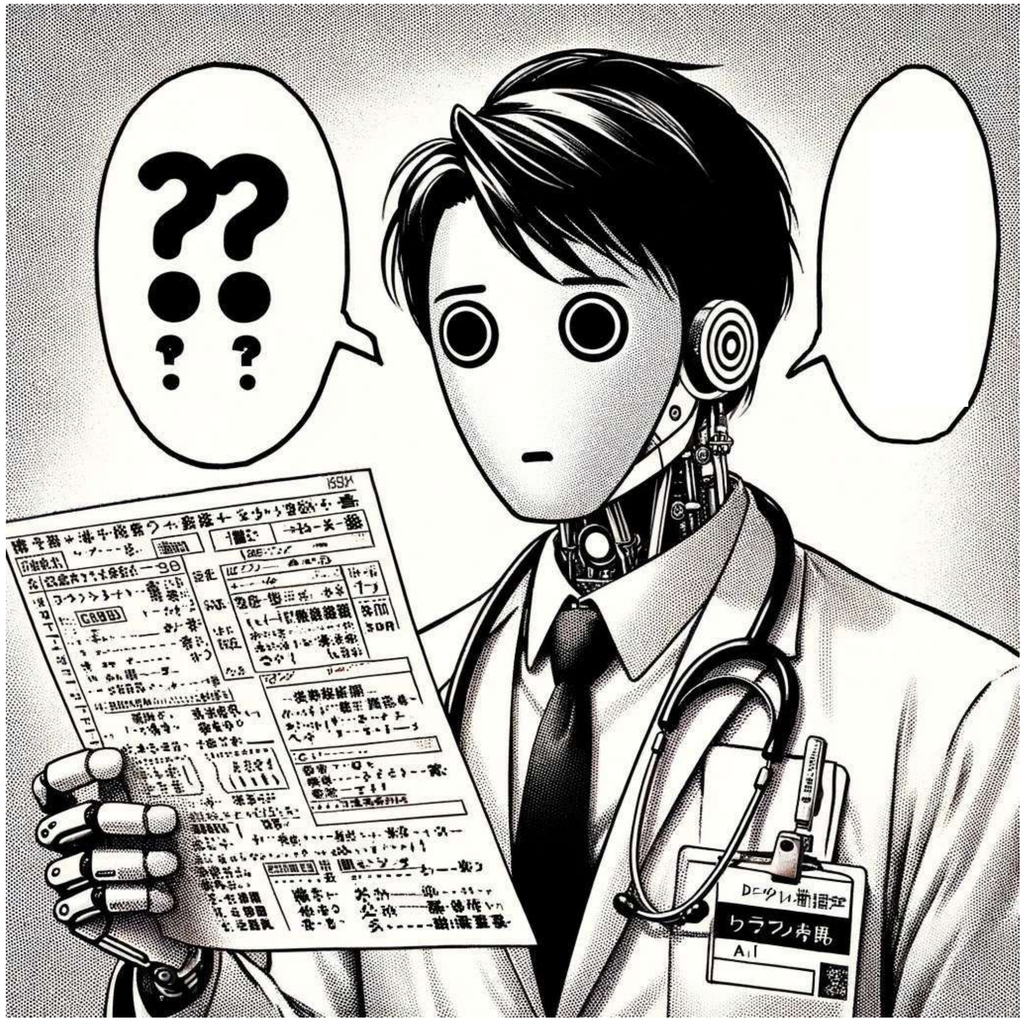
- **Major issues**
  - **Hallucination**: generated information is either in conflict with the existing source (intrinsic hallucination) or cannot be verified by the available source (extrinsic hallucination)
  - **Knowledge recency**: how to acquire latest knowledge beyond the training data



Example from: https://kili-technology.com/large-language-models-llms/understanding-llm-hallucinations-and-how-to-mitigate-them

130 Hallucination

- Unaware users might unknowingly rely on these AI-generated answers, leading to adverse outcomes, especially in fact-driven fields such as law, healthcare, and finance



Case 4 - ChatGPT cited non-existent medical references

Professor Robin Emsley detailed his experience experimenting with ChatGPT for writing scientific papers. While ChatGPT performed consistently in general brain-related topics, it started to hallucinate to the point of falsification when pressed on complex topics. In this case, ChatGPT provided several references that didn't exist or were irrelevant to the response it generated.

FORBES > BUSINESS

BREAKING

Lawyer Used ChatGPT In Court—And Cited Fake Cases. A Judge Is Considering Sanctions

Example: FAC²E - Better Understanding LLM Capabilities by Dissociating Language and Cognition

- Formulate LLMs' evaluation by dissociating the language-related capabilities and the cognition-related ones.

- Break down the process of applying a specific capability into three sub-steps: recalling relevant knowledge, utilizing knowledge, and solving problems.

- We identify **a common shortfall in knowledge utilization among models** and propose a knowledge-enhanced method to mitigate this issue.

Capability	Description		Skill Example
LINGUISTIC KNOWLEDGE	Encoding grammatical concepts support linguistic operations regarding word meanings and their combinatorial processing.	Grammaticality: Semantics:	agreements, licensing, long-distance dependencies, and garden-path effects. synonymy, antonymy, and hypernymy.
FORMAL KNOWLEDGE	Conducting word-based formal reasoning through understanding shallow semantics.	Mechanism: Skill:	deductive, inductive, and analogical. numeric, logic, and manipulation.
WORLD MODELING	Understanding text based on given context and associating it with world knowledge.	Remember: Understand:	factual knowledge, context, and commensense. narrative structure and discourse comprehension.
SOCIAL MODELING	Infering mental state behind text and intended meaning beyond literal content.	Pragmatics: Theory-of-mind	polite deceits, irony, maxims of conversation, metaphor, indirect speech, and humor. unexpected content and unexpected transfer tasks.

Reference: Wang et al. "FAC2E: Better Understanding Large Language Model Capabilities by Dissociating Language and Cognition" arXiv preprint (2024).

Retrieve-Augmented Generation (RAG)

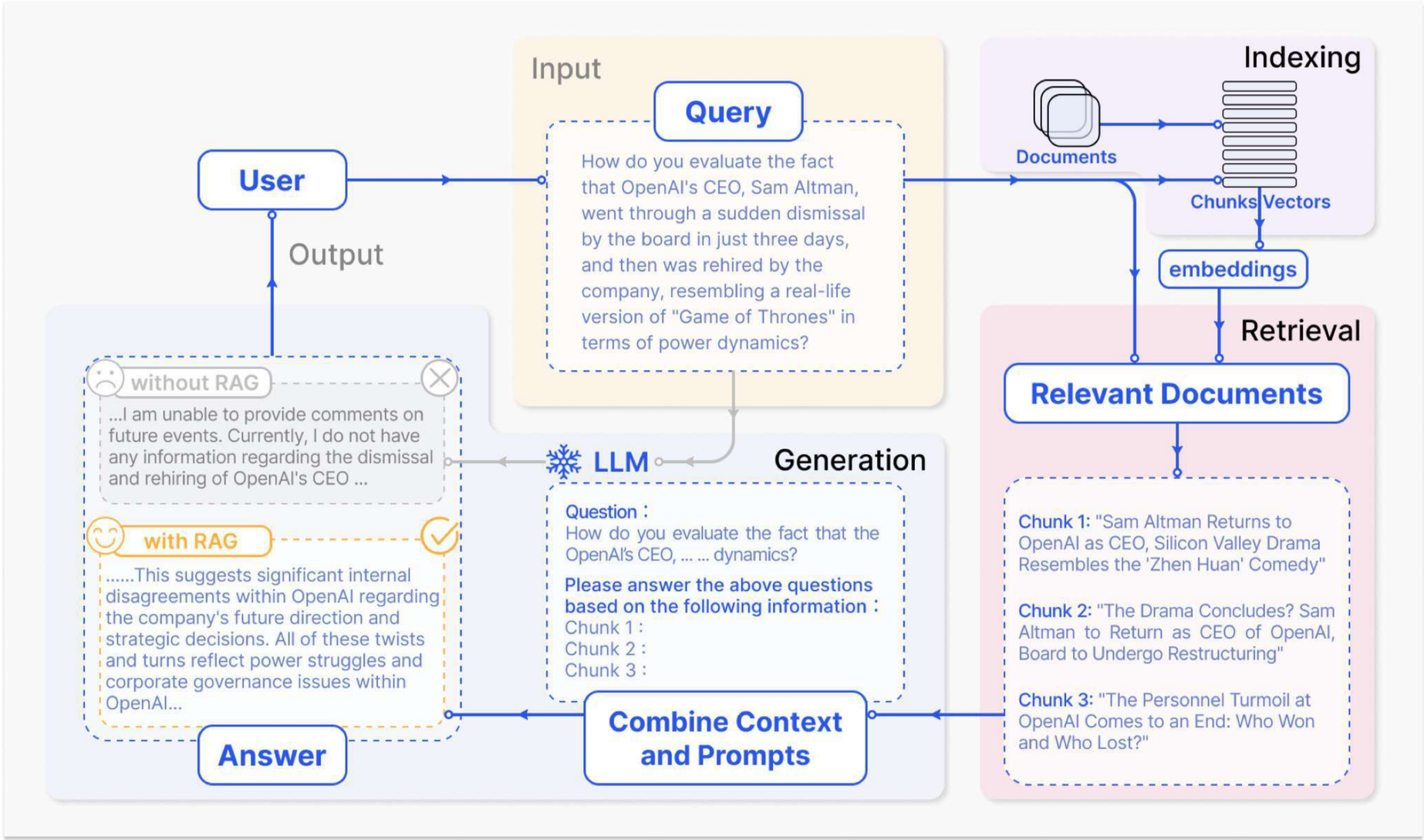
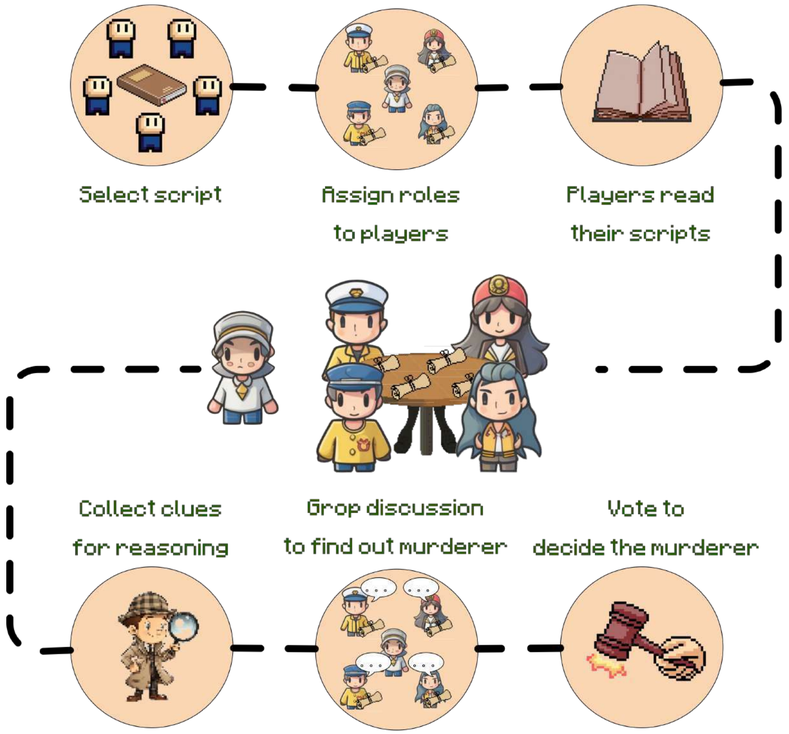


Image from: Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 (2023).

Complex Reasoning

- Complex reasoning refers to the ability of understanding and utilizing supporting evidence or logic to derive conclusions or make decisions
  - knowledge reasoning, symbolic reasoning, and mathematical reasoning.



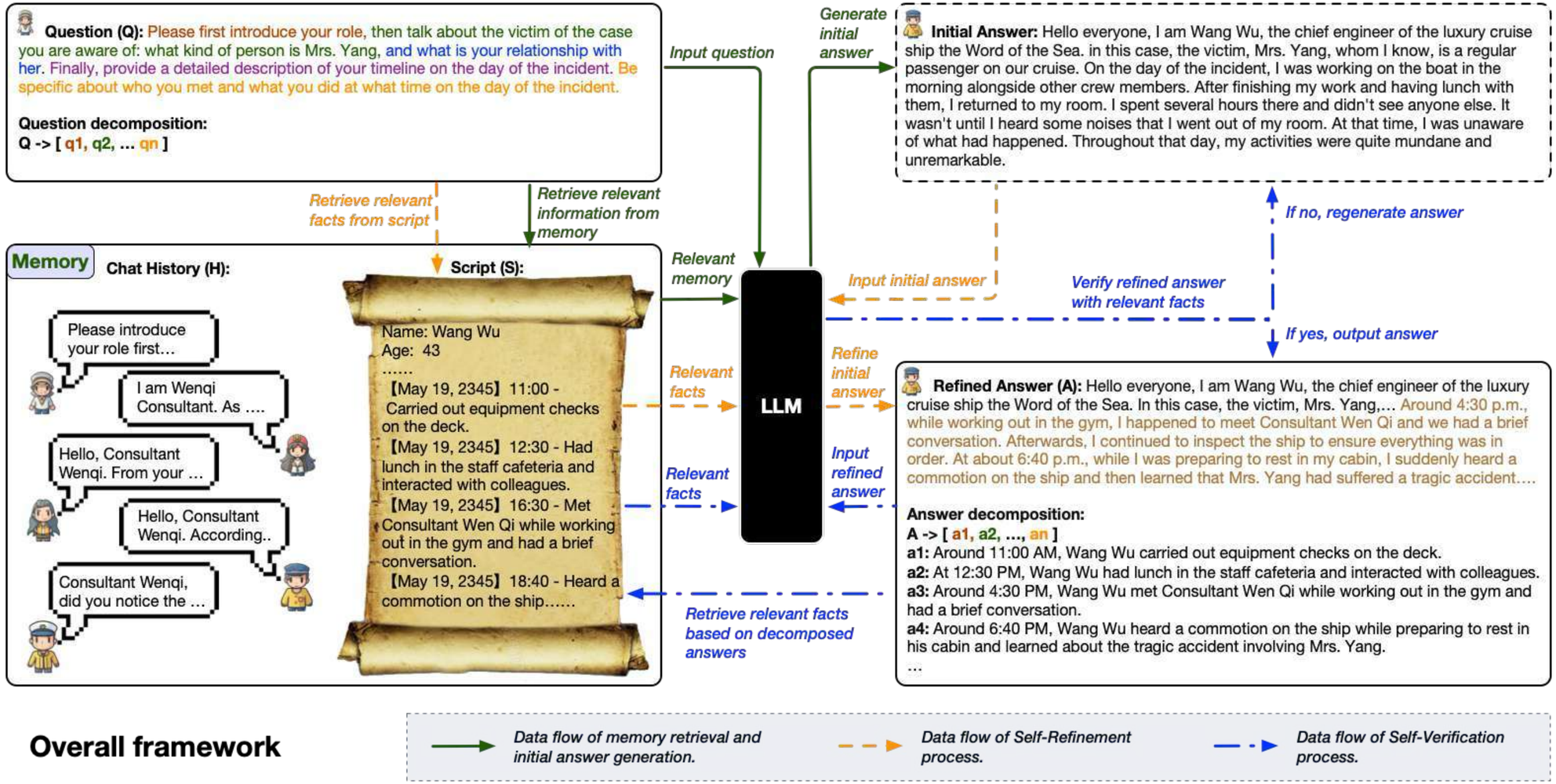
(a) Game flow of Jubensha detective game

(b) Example of group discussion

Jubensha: a multi-agent detective game

Reference: Wu et al. "Deciphering Digital Detectives: Understanding LLM Behaviors and Capabilities in Multi-Agent Mystery Games" in Findings of ACL 2024.

Example: *ThinkThrice* for enhancing agent's performance in multi-agent detective games (i.e., Jubensha)



Overall framework

Step 1. Memory Retrieval: generate initial answer

Step 2. Self-Refinement: increase answer details

Step 3. Self-Verification: reduce hallucination

Reference: Wu et al. "Deciphering Digital Detectives: Understanding LLM Behaviors and Capabilities in Multi-Agent Mystery Games" in Findings of ACL 2024.

Example: Data Interpreter: An LLM Agent For Data Science

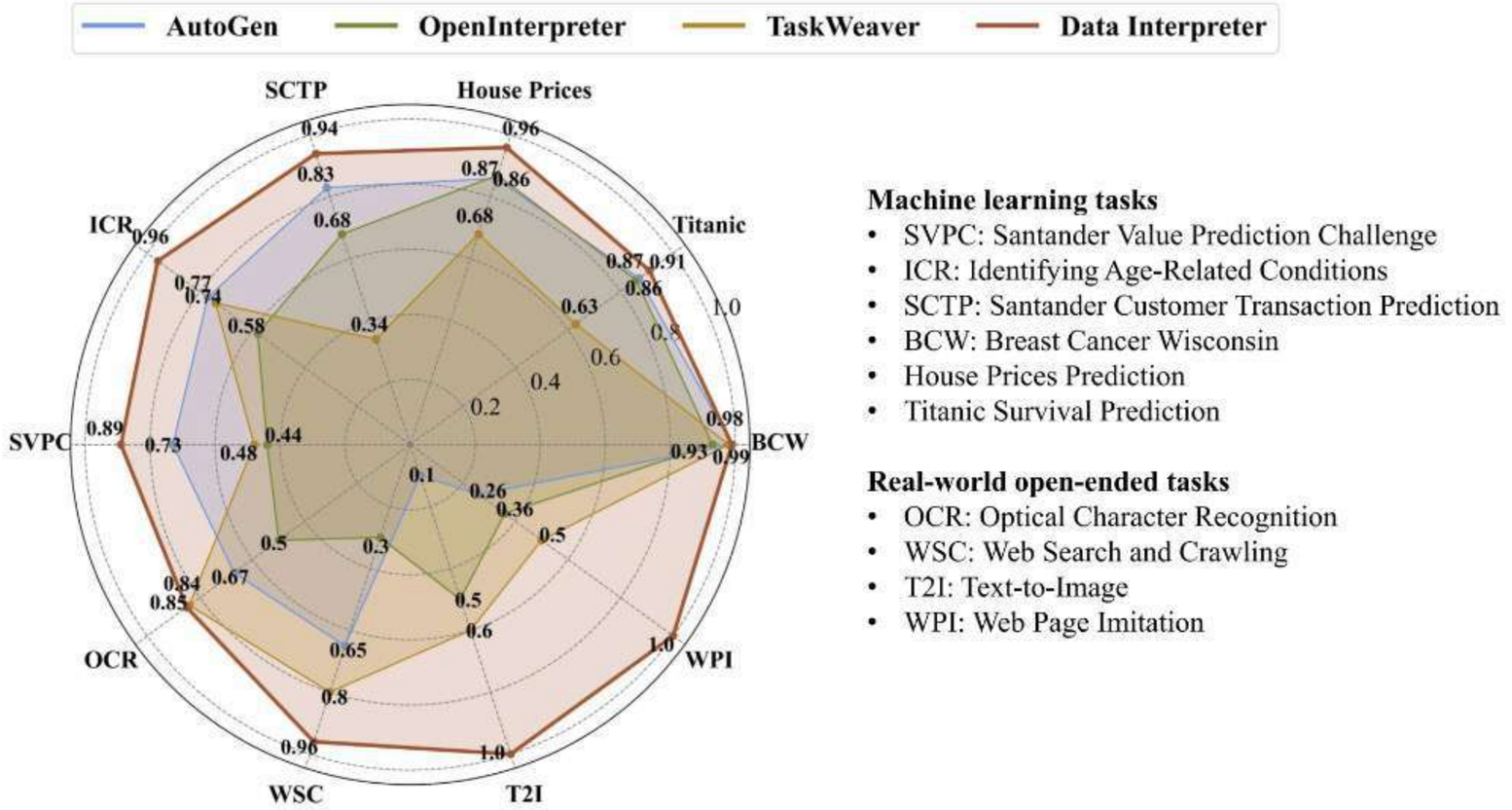


Figure 1: Comparison with various open-source frameworks on machine learning tasks and real-world open-ended tasks.

Reference: Hong et al. "Data Interpreter: An LLM Agent For Data Science" arXiv preprint 2024.

Example: Data Interpreter: An LLM Agent For Data Science

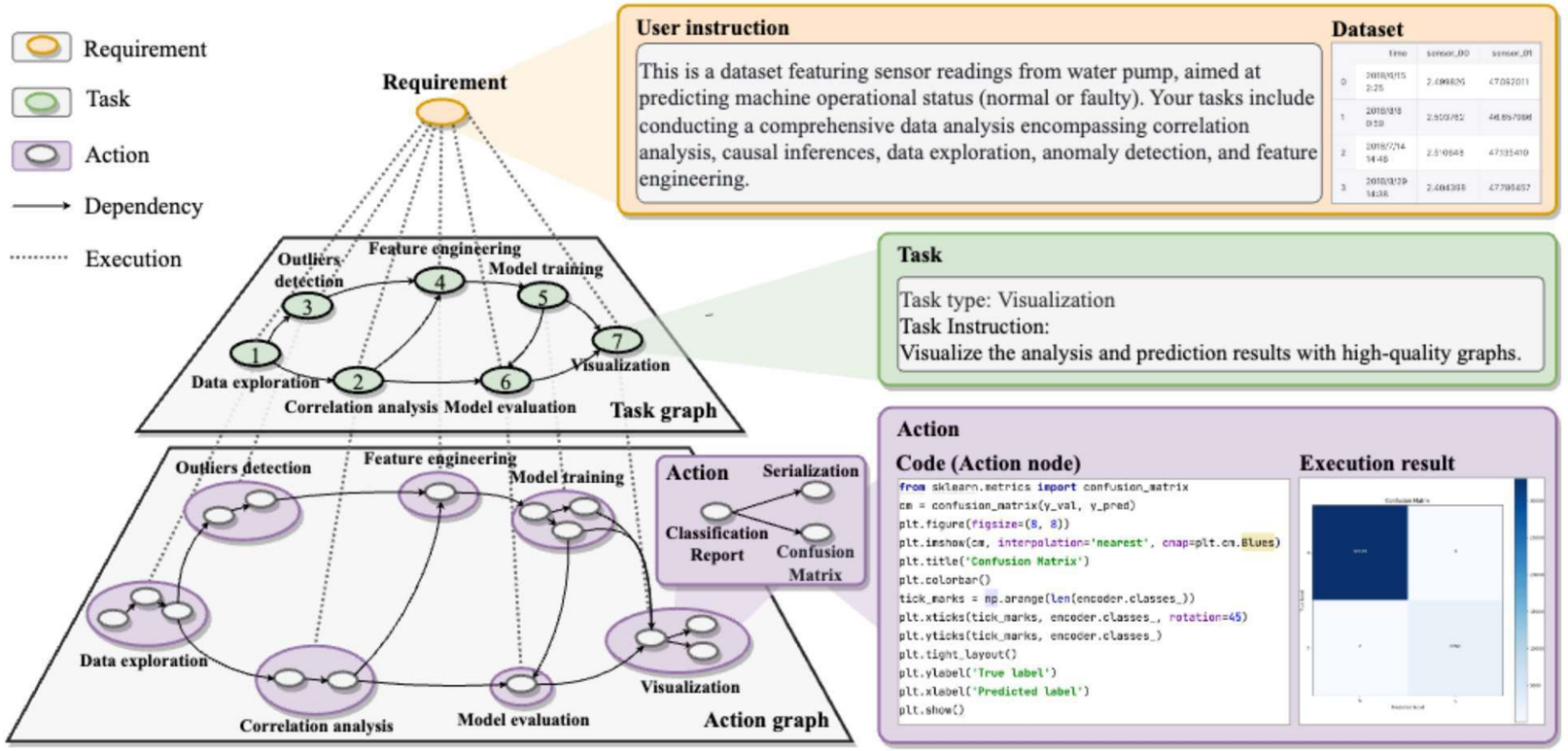
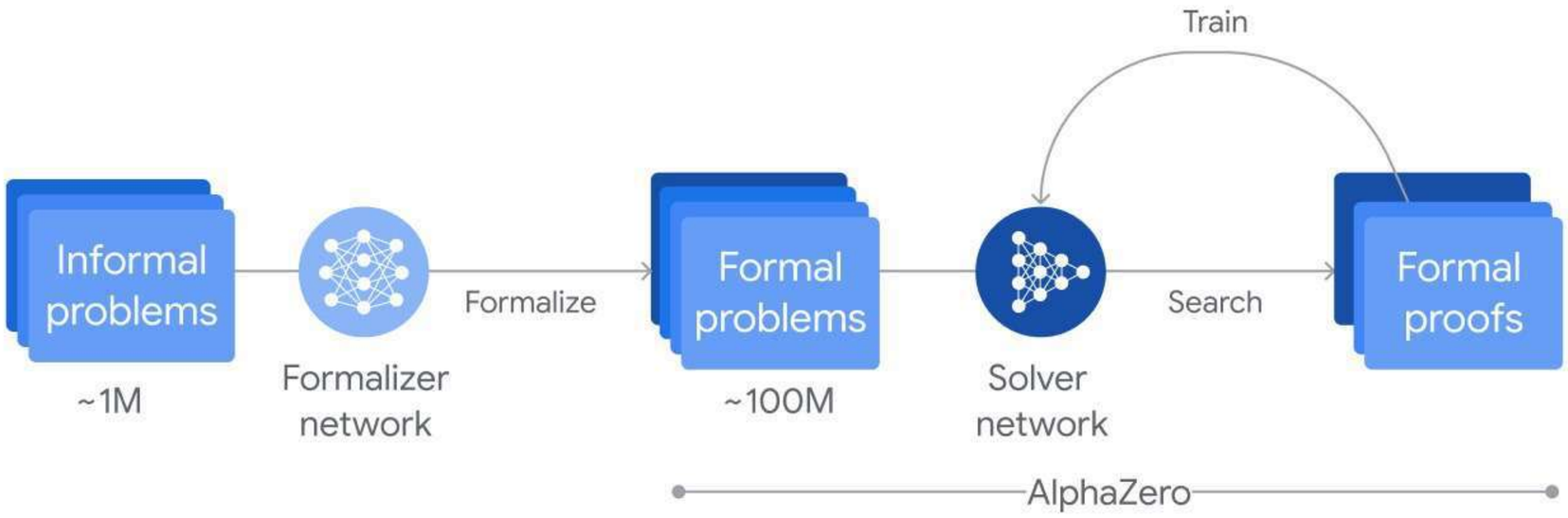


Figure 3: **Hierarchical graph structure.** The Data Interpreter decomposes a user-defined requirement into a task graph. Each node in the graph represents a processing step, such as model training, and each edge represents task dependencies. The task graph is then transformed into an action graph. This action graph consists of action sequences represented as executable and verifiable codes. By executing the action graph, the solution to the requirement is obtained.

Reference: Hong et al. "Data Interpreter: An LLM Agent For Data Science" arXiv preprint 2024.

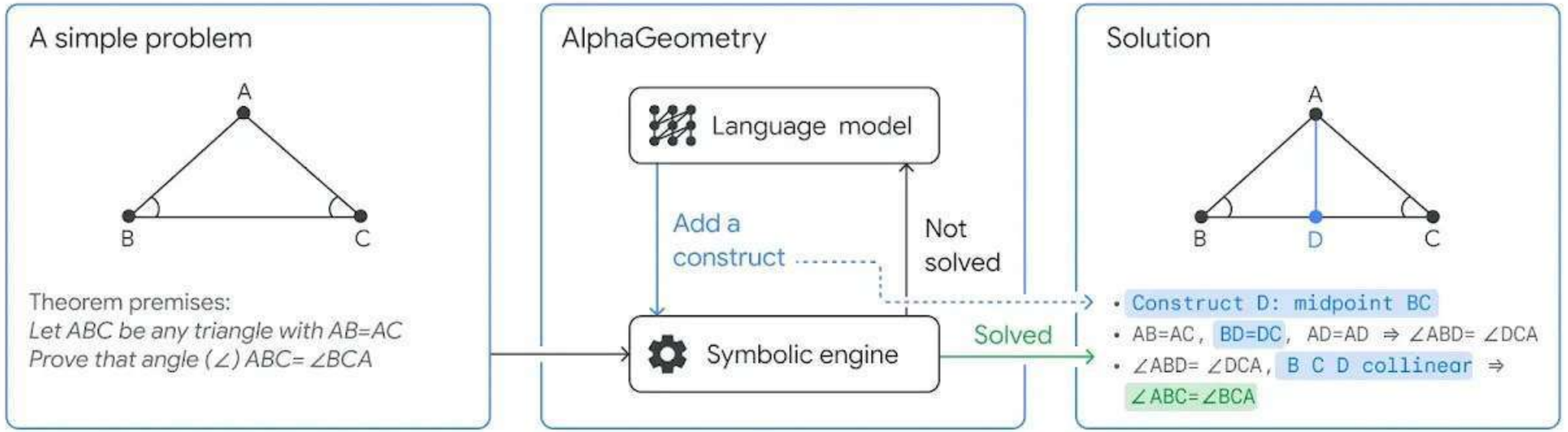
Example: AlphaProof and AlphaGeometry



1. **Proofs written in formal languages involve mathematical reasoning that can be formally verified for correctness.**
2. **Fine-tuning a Gemini model to translate natural language problem statements into formal statements**
3. **AlphaProof generates solution candidates and then proves or disproves them by searching over possible proof steps in Lean.**

https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

Example: AlphaProof and AlphaGeometry

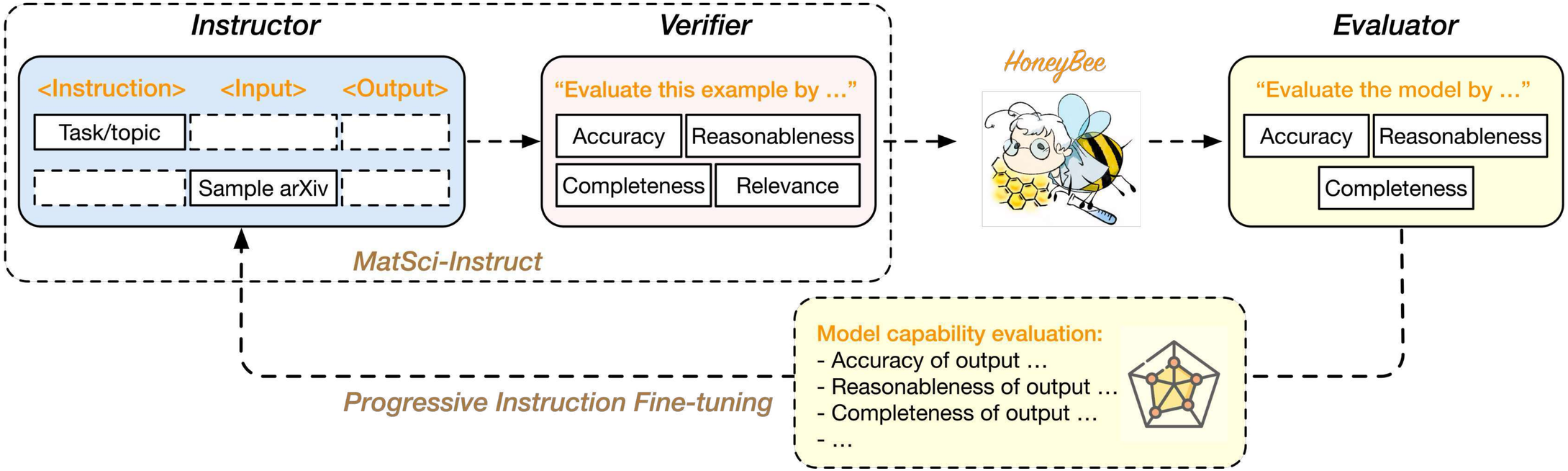


1. **AlphaGeometry is a neuro-symbolic system made up of a neural language model and a symbolic deduction engine.**
   1. **Language models: excel at identifying general patterns and relationships in data, they can quickly predict potentially useful constructs**
   2. **Symbolic deduction engines: reason rigorously based on formal logic and use clear rules**
2. **Generated a vast pool of synthetic training data - 100 million unique examples**
3. **AlphaProof generates solution candidates and then proves or disproves them by searching over possible proof steps in Lean.**

https://deepmind.google/discover/blog/alphageometry-an-olympiad-level-ai-system-for-geometry/

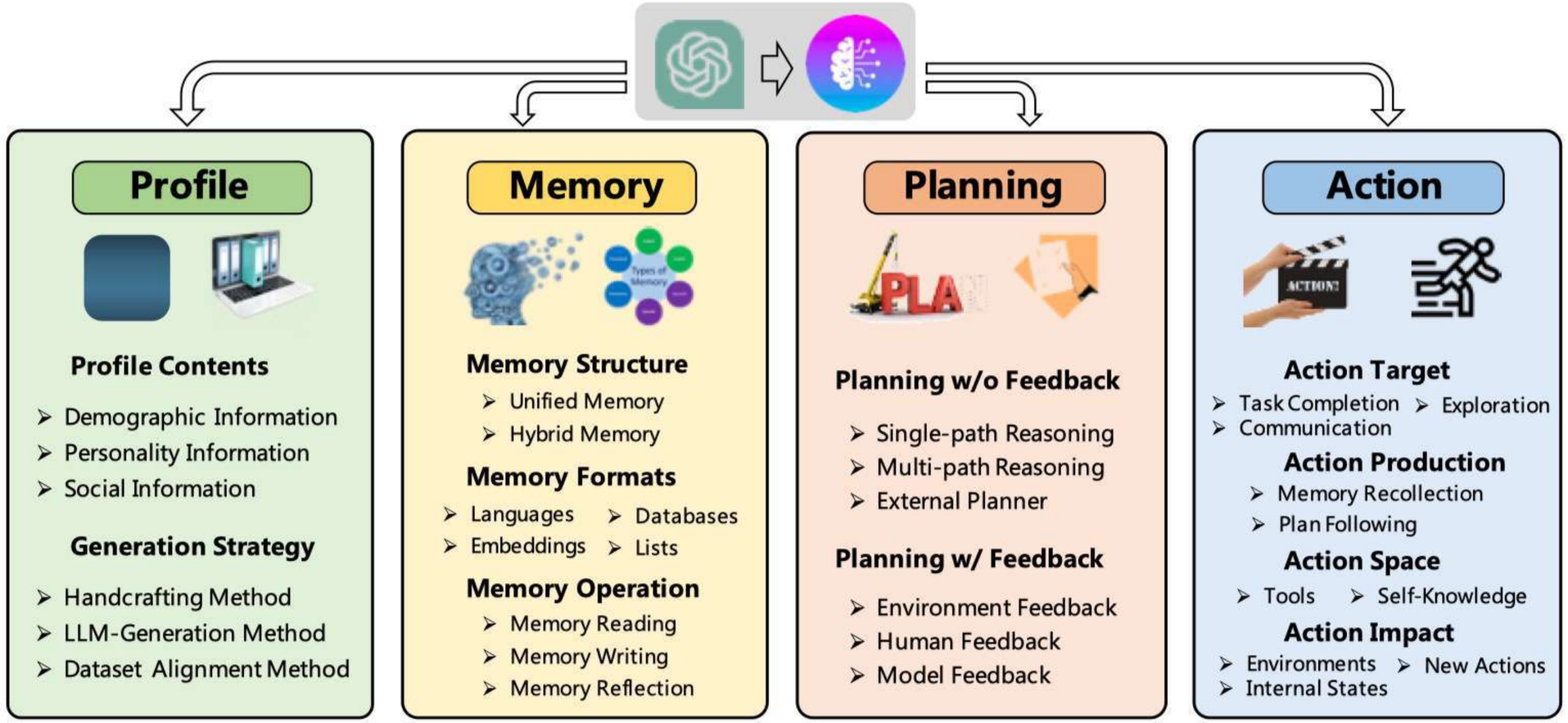
Example: AI for Materials Science

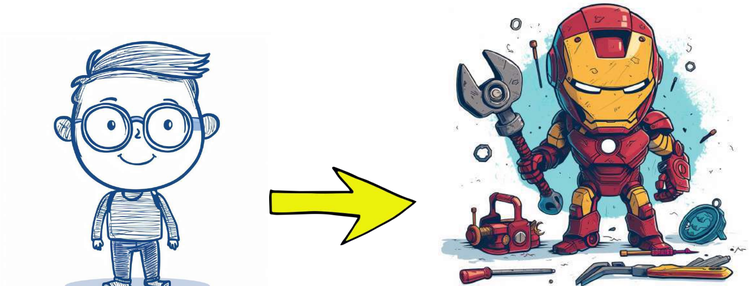
HoneyBee is the first billion-parameter scale language model that is specialized in materials science



Song et al. "HoneyBee: Progressive Instruction Finetuning of Large Language Models for Materials Science" in Findings of EMNLP (2023).

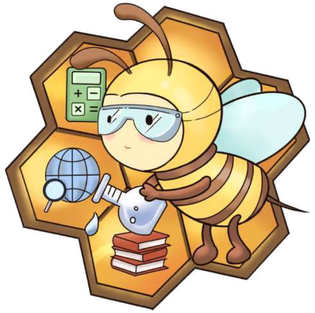
Augment LLMs: Agents





Wang, Lei, et al. "A survey on large language model based autonomous agents." arXiv preprint arXiv:2308.11432 (2023).

Example: HoneyComb for Materials Science



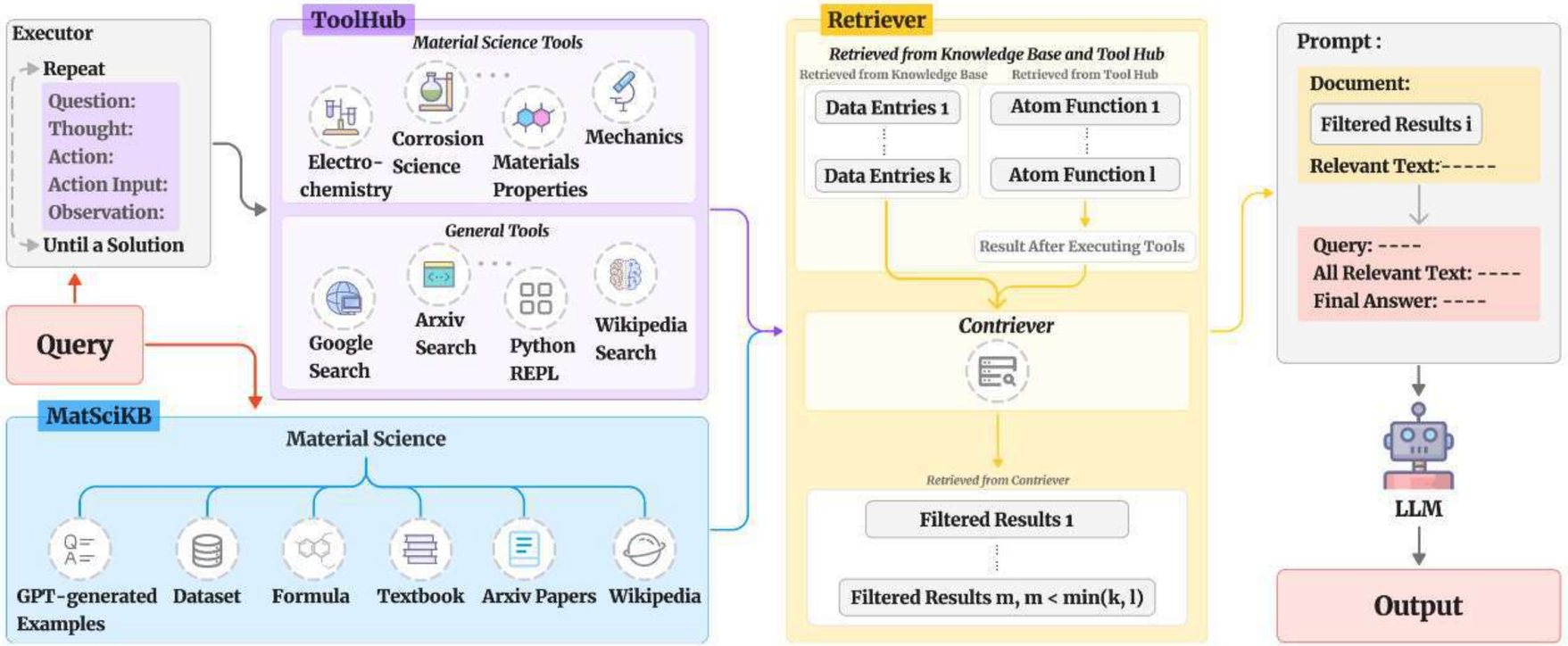


Figure 1: The overall architecture of HoneyComb. The model initiates with a query input that activates the knowledge retrieval phase, where pertinent data entries and atom function are extracted from the MatSciKB and Tool-Hub respectively. The Executor iterative calls the relevant tools from the Tool-Hub, evaluating and refining these calls until a solution that adequately solves the query emerges. The preliminary solution generated by these tools is combined with relevant data entries, and then undergoes further processing by the Retriever. Finally, the Retriever consolidates and filters these input, ultimately feeding them into the LLM for final answer generation.

Zhang et al. "HoneyComb: A Flexible LLM-Based Agent System for Materials Science", under submission. 2024.

How do various LLM-centric agents' components impact EIF performance?

Embodied Instruction Following (EIF) on ALFRED

EIF task types in ALFRED

Pick & Place	Pick Two & Place	Stack & Place	Examine in Light
Heat & Place	Clean & Place	Cool & Place	

One task example

Task arguments

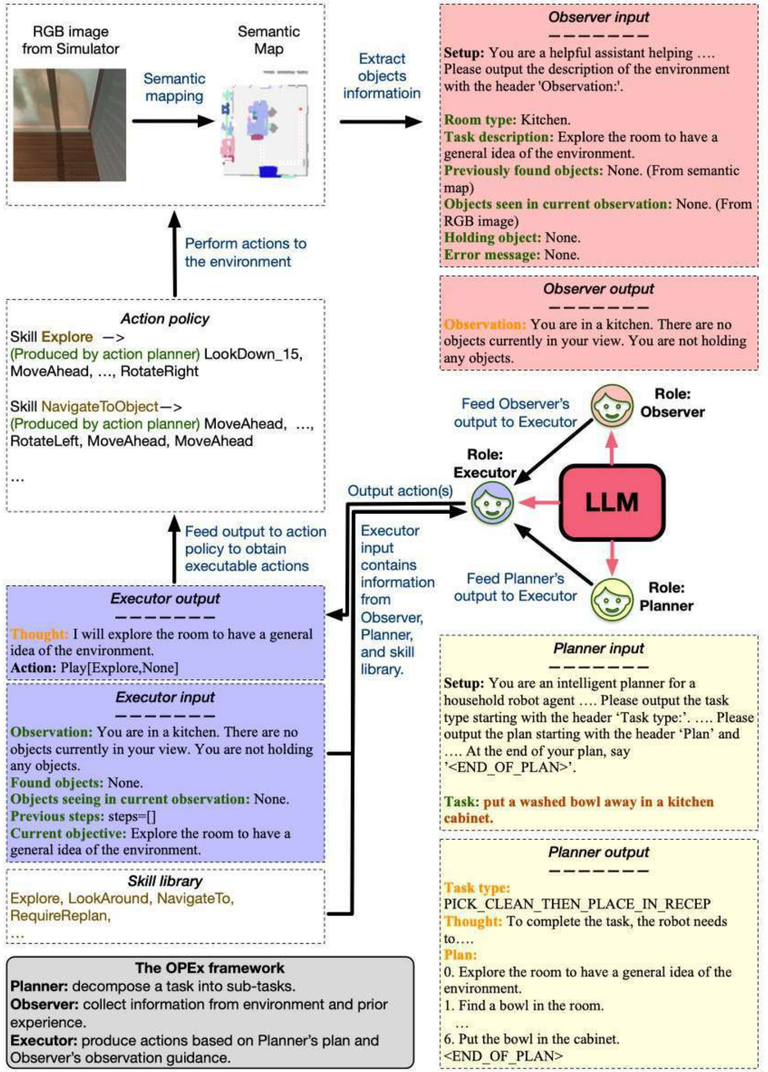
Item(s): Ladle **Receptacle:** Table **Scene #:** 27

One expert annotation

High-level instruction: Place a clean ladle on a table

Low-level instruction:	Turn around and veer left to the white table.	Pick up the ladle on the table, behind the tomato.	Turn right and walk to the sink
Ego-centric RGB input:	 *t = 28*	 *t = 61*	 *t = 74*
Low-level instruction:	Wash the ladle in the right sink basin.	Turn left and walk back to the white table.	Place the ladle on the table in front of the white mug.
Ego-centric RGB input:	 *t = 83*	 *t = 117*	 *t = 164*

Overall Framework



Reference: Shi et al. "OPEx: A Component-Wise Analysis of LLM-Centric Agents in Embodied Instruction Following" in ACL 2024.

How do various LLM-centric agents' components impact EIF performance?

Component-wise Analysis

Method	Valid Uneen			
	PLWGC	GC	PLWSR	SR
Influence of perception models				
OPEx	13.48	48.61	9.08	35.91
OPEx-S	16.52	51.28	11.38	40.80
OPEX-P	23.72	66.17	17.43	59.43
Influence of action policies				
OPEx-S	16.52	51.28	11.38	40.80
-semantic map M'_t	12.37	45.41	8.06	36.17
-slice replay	12.64	45.25	8.35	37.39
-traversable goal	11.77	43.49	7.09	34.50
Influence of LLM-based modules				
OPEx-S	16.52	51.28	11.38	40.80
-Planner	8.10	40.16	5.72	30.57
-Observer	13.41	45.62	8.58	37.76
Influence of introduced skills				
OPEx-S	16.52	51.28	11.38	40.80
-LookAround	12.56	45.37	8.26	37.76
-Explore	13.29	47.32	8.44	38.25
Influence of prior knowledge (On ALFWorld)				
ReAct	-	-	-	66
OPEx	-	-	-	73
OPEx-L	-	-	-	78
OPEx-H	-	-	-	84

As the bases for further reasoning, planning, and acting, a **precise representation of the Env** is critical.

Heuristic action policy is a **vanilla solution**.
- Any better learning-based method?

Significant potential for **better grounding.**

Reducing planning horizon is vital.

The knowledge gap between **pre-trained LLMs** and the **target domain** should be bridged.

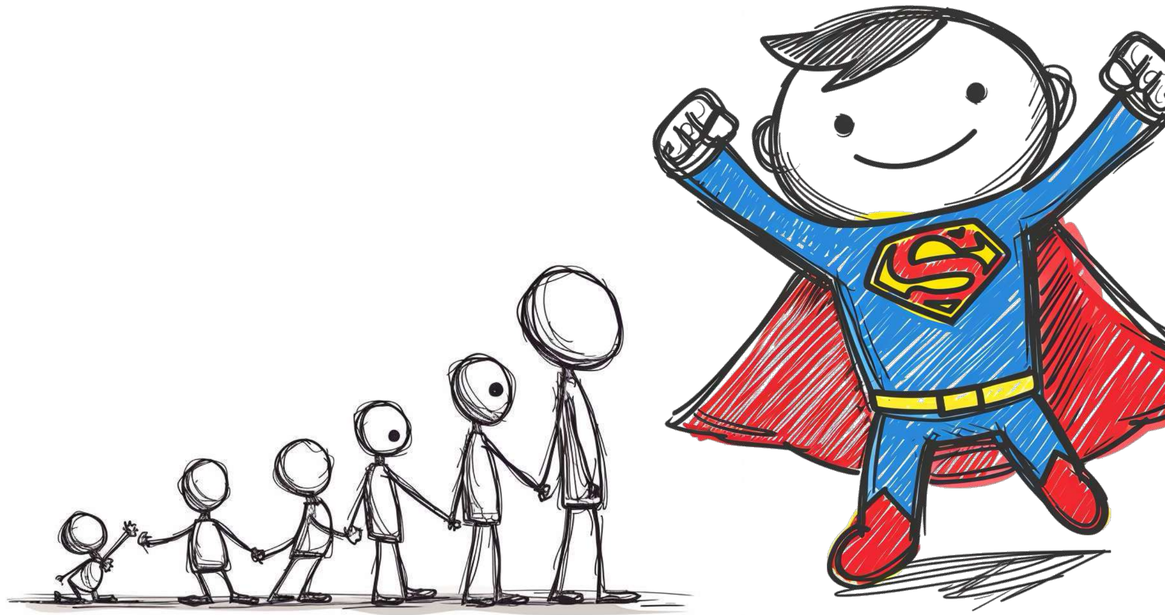
Low Demand for In-Domain Data

Method	SR	GC	PLWSR	PLWGC
OPEx	38.12	46.13	9.03	13.45
FILM	0.00	12.18	0.00	2.78

Reference: Shi et al. "OPEx: A Component-Wise Analysis of LLM-Centric Agents in Embodied Instruction Following" in ACL 2024.

Summary: Text Generation with Large Language Models

- **A paradigm shift of NLP.** Self-supervised learning with LM tasks + Transformer architectures.
  - Scaling law
  - Emergent abilities
- **Core Techniques of LLMs.** Pre-training, adaptation, utilization, evaluation, and applications.
- **Future directions.** Better understand, improve, and apply LLM models for various domains with safety, interpretability, and controllability.



References

1. Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).
2. Kaplan, Jared, et al. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).
3. Hoffmann, Jordan, et al. "Training compute-optimal large language models." arXiv preprint arXiv:2203.15556 (2022).
4. "Gpt-4 technical report." OpenAI (2023).
5. Bahri, Yasaman, et al. "Explaining neural scaling laws." arXiv preprint arXiv:2102.06701 (2021).
6. Zhao, Wayne Xin, et al. "Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance." Google Research Blog
7. Wei, Jason, et al. "Finetuned language models are zero-shot learners." arXiv preprint arXiv:2109.01652 (2021).
8. Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." NeurIPS (2022): 24824-24837.
9. Schaeffer et al. "Are Emergent Abilities of Large Language Models a Mirage?" NeurIPS (2023).
10. Qiu, Yixuan, et al. "Pre-training in Medical Data: A Survey." Machine Intelligence Research 20.2 (2023): 147-179.

References

11. Gunasekar et al., Textbooks Are All You Need. 2023.

12. Li et al., Textbooks Are All You Need II: phi-1.5 technical report. 2023

13. Chen, et al. "Data-Juicer: A One-Stop Data Processing System for Large Language Models" arXiv preprint (2023).

14. Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." JMLR 21.1 (2020): 5485-5551.

15. Wang, Thomas, et al. "What language model architecture and pretraining objective works best for zero-shot generalization?." ICML, 2022.

16. Mistral.ai. "Mixtral of Experts” arXiv preprint (2023).

17. Albert Gu and Tri Dao. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces”. arXiv preprint. 2023.

18. Peng et al.. "RWKV: Reinventing RNNs for the Transformer Era”. arXiv preprint. 2023.

19. Poli et al.. "Hyena Hierarchy: Towards Larger Convolutional Language Models”. ICML. 2023.

20. Xiao et al. "Efficient streaming language models with attention sinks" arXiv preprint (2023).

References

21. Wang et al., ”Resonance RoPE: Improving Context Length Generalization of Large Language Models" in Findings of ACL (2024).

22. Rajbhandari, Samyam, et al. "Zero: Memory optimizations toward training trillion parameter models." SC20: ICHPC, Networking, Storage and Analysis. IEEE, 2020.

23. Micikevicius, Paulius, et al. "Mixed Precision Training." International Conference on Learning Representations. 2018.

24. Lou et al. "A Comprehensive Survey on Instruction Following”, arXiv preprint. 2024.

25. PKU Alignment team. "AI Alignment: A Comprehensive Survey." arXiv preprint (2023).

26. Scheurer et al. "Technical Report: Large Language Models can Strategically Deceive their Users when Put Under Pressure." arXiv preprint (2023).

27. Lee et al., RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. ArXiv preprint. 2023.

28. Anthropic, Constitutional AI: Harmlessness from AI Feedback. 2022.

29. Rafailov et al., Direct Preference Optimization: Your Language Model is Secretly a Reward Model. ArXiv preprint. 2023.

30. OpenAI, Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision. ArXiv preprint. 2023.

References

31. Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." Low-Power Computer Vision. Chapman and Hall/CRC, 2022. 291-326.

32. Besta, Maciej, et al. "Graph of thoughts: Solving elaborate problems with large language models." arXiv preprint arXiv:2308.09687 (2023).

33. Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models." arXiv preprint arXiv:2305.10601 (2023).

34. Sel, Bilgehan, et al. "Algorithm of thoughts: Enhancing exploration of ideas in large language models." arXiv preprint arXiv:2308.10379 (2023).

35. Song, Chan Hee, et al. "Llm-planner: Few-shot grounded planning for embodied agents with large language models." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023.

36. Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 (2023).

37. Wu et al. "Deciphering Digital Detectives: Understanding LLM Behaviors and Capabilities in Multi-Agent Mystery Games" Findings of ACL (2024).

38. Wang, Lei, et al. "A survey on large language model based autonomous agents." arXiv preprint arXiv:2308.11432 (2023).

39. Song et al. “HoneyBee: Progressive Instruction Finetuning of Large Language Models for Materials Science” in Findings of EMNLP (2023).

40. Ouyang, Long, et al. "Training language models to follow instructions with human feedback." Advances in Neural Information Processing Systems 35 (2022): 27730-27744.

41. Sharma, Utkarsh, and Jared Kaplan. "Scaling laws from the data manifold dimension." The Journal of Machine Learning Research 23.1 (2022): 343-376.

42. Wang et al. "FAC2E: Better Understanding Large Language Model Capabilities by Dissociating Language and Cognition" arXiv preprint (2024).

References

43. Longpre et al., “A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity” arXiv preprint (2023).

44. Albalak et al. "A Survey on Data Selection for Language Models" arXiv preprint arXiv:2042.16827 (2024).

45. Sun, et al. "Learning to (Learn at Test Time): RNNs with Expressive Hidden States" arXiv preprint (2024).

46. Su et al., ”RoFormer: Enhanced Transformer with Rotary Position Embedding" Neurocomputing, 2024.

47. Sclar et al. "QUANTIFYING LANGUAGE MODELS’ SENSITIVITY TO SPURIOUS FEATURES IN PROMPT DESIGN or: How I learned to start worrying about prompt formatting" ICLR 2024

48. Khattab et al. "DSPy: Compiling Declarative Language Model Calls into Self-improving Pipelines" ICLR 2024

49. Yuksekgonul et al. "TextGrad: Automatic “Differentiation” via Text" arXiv preprint 2024

50. Hong et al. "Data Interpreter: An LLM Agent For Data Science" arXiv preprint 2024.

51. Shi et al. "OPEx: A Component-Wise Analysis of LLM-Centric Agents in Embodied Instruction Following" in ACL 2024.

Thanks! Q&A

Tutorial Website

https://sites.google.com/view/aigc-tutorial/home?authuser=0