Introduction to Large Language Models(LLM)

Outline

1. Overview from 30,000 feet above
2. Transformer in nutshell
3. Pretraining - Parallel paradigm
4. Finetuning - Parameter efficient finetuning
5. Steering the decoding of LLM - Prompt
6. Augmentation and Plugins

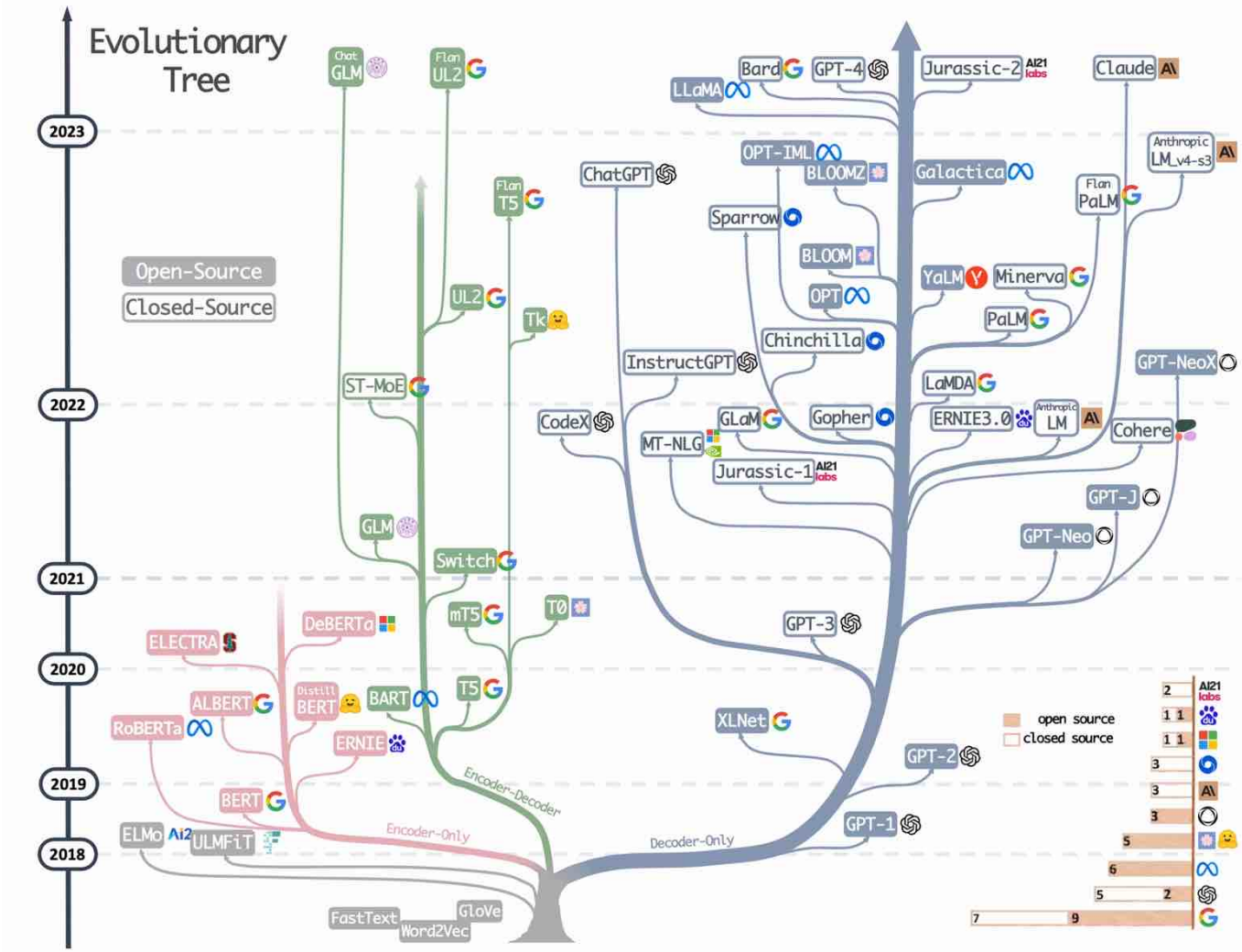
Overview from 30000 feet above

- Paradigm transition in AI
  - From: **training**(specific) -> **prediction**(specific)
  - To: **pretraining**(general) -> **finetuning**(general/specific) -> **in-context prompt**(specific)
- Primary steps of new paradigm
  - Pretraining with self-supervised learning
  - Finetuning on instruction from mutiple domains
  - Supervised Finetuning & RLFH
  - Application by steering/prompt the decoding process of LLM
- Where are we to AGI?
  - From explanation to prediction
  - From correlation to causality

AI and the future of humanity, 2023, Yuval Noah Harari

A LLMs Evolution Tree

- Decoder Only
- Encoder Only
- Encoder-Decoder



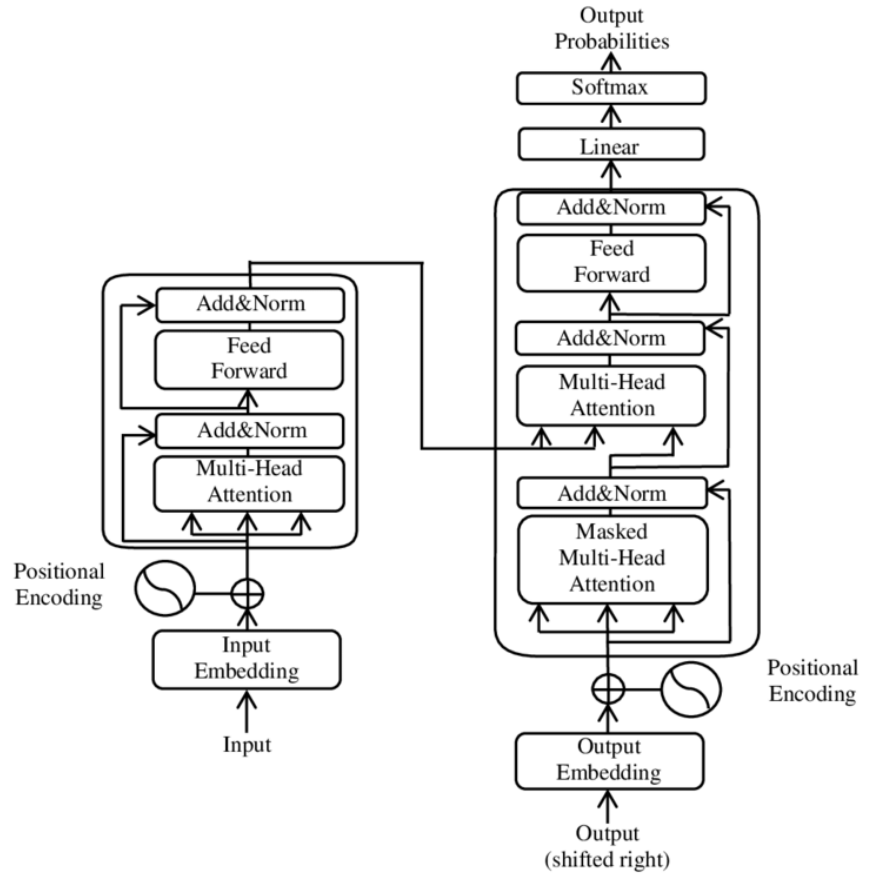
Img Source: Jingfeng Yang et al. 2023

Transformer in nutshell

- Transformer modules
- Aspects of alternative
- Parameter concentration
- Computation concentration

Transformer modules

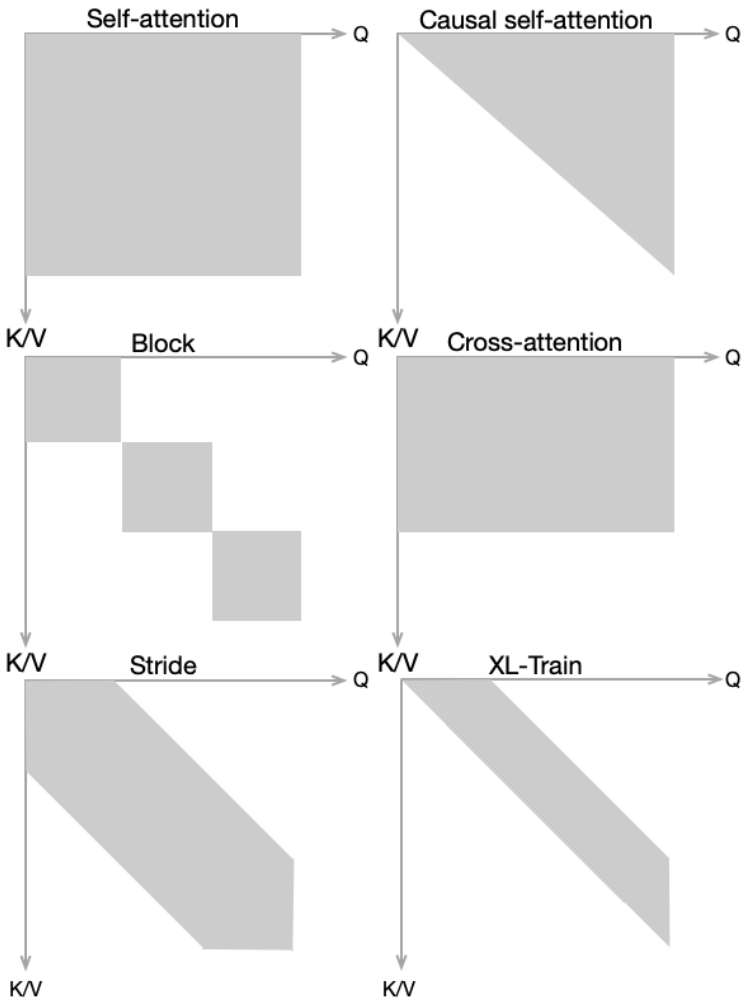
- Token & positional embedding :
  - $E \in R^{V \times d}, P \in R^{T \times d}$
- Multi-head attention
  - Self-attention & Cross-attention
    - Weight matrix:
      $W_Q^h, W_K^h, W_V^h \in R^{d \times d_h}$
    - Head projection:
      $W_O \in R^{d \times d}$
- ResNet & LayerNorm
- Feedfoward Network
  - $W_1 \in R^{d \times 4d}, W_2 \in R^{4d \times d}$
- Output head: task related



Source : Wiki:Transformer_(machine_learning_model)

Aspects of alternative

- Efficient Transformer (for long sequence)
  - Coarse sequence resolution:
    - Block/Stride/Clustering/Neural Memory
    - TransformerXL
  - Attention matrix approximation:
    - Linear Transformer
- LLMs specific
  - Encoder vs **Decoder** vs Encoder-Decoder
  - Positional encoding:
    - Absolute/**Relative**
    - Learned/**Fixed**
  - LayerNorm: **Input**/Output
  - Activation: GeLU



Parameter concentration

Decoder only Transformer(GPT)

- Parameter size:
  - Embedding: $(V+T) \times d$
  - Attention: $L \times (3 \times d \times d_h \times H + (d_h \times H) \times d) = 4Ld^2$
  - FFN: $L \times ((d \times 4d + 4d) + (4d \times d + d)) \approx 8Ld^2$
- On GPT3-175B:

Total	PosEnc	TermEnc	Attn	FFN
174,597M	25M(0.01%)	617M(0.35%)	57,982M(33.21%)	115,970M(66.42%)

Computation concentration

Decoder only Transformer

Per-token calculation:

- QKV+project: $2 \times L \times (3 \times H \times h_d \times d + (H \times h_d) \times d) = 2 \times 4Ld^2$
- Attention: $2 \times L \times T \times d$
- FFN: $2 \times L \times ((d \times 4d + 4d) + (4d \times d + d)) \approx 2 \times 8Ld^2$

Model training flops utilization(MFU):

- Forward and backward: $(1 + 2) \times (2N + 2LTd) \approx 6N$, where $N \approx 12Ld^2$
- Theoretical peak throughput: $\frac{P}{6N+6LTd}$
- $\text{MFU} = \frac{\text{Observed throughput}}{\text{Theoretical peak throughtput}}$

PaLM: Scaling Language Modeling with Pathways, 2022, Google

Chinchilla: Training Compute-Optimal Large Language Models, 2022, Google

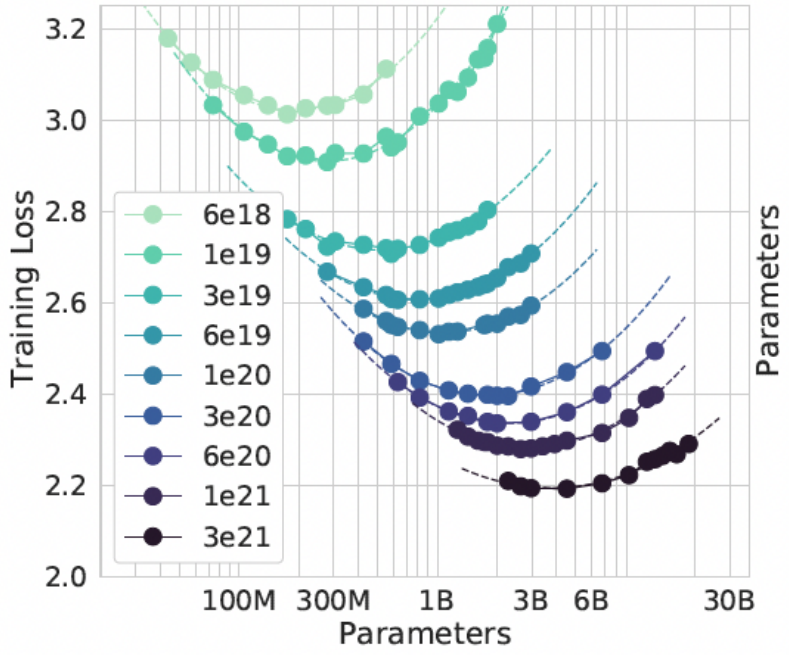
Scaling laws and compute optimal LLM training

Optimal combination of model size N and token size D, given the computation FLOPS C

Optimal N and D:

- $N_{\text{opt}} \propto C^a$
- $D_{\text{opt}} \propto C^b$
- OpenAI(2020) : $a = 0.73, b = 0.27$
- Deepmind(2022) : $a = 0.5, b = 0.5$
  - Match between training tokens and LR schedule
  - Tested on larger model size($> 500M$ vs. $< 100M$)
- Optimal ratio: $N : D \simeq 1 : 20$

IsoFLOP profiles(Deepmind, 2022):



Scaling Laws for Neural Language Models, 2020, Deepmind

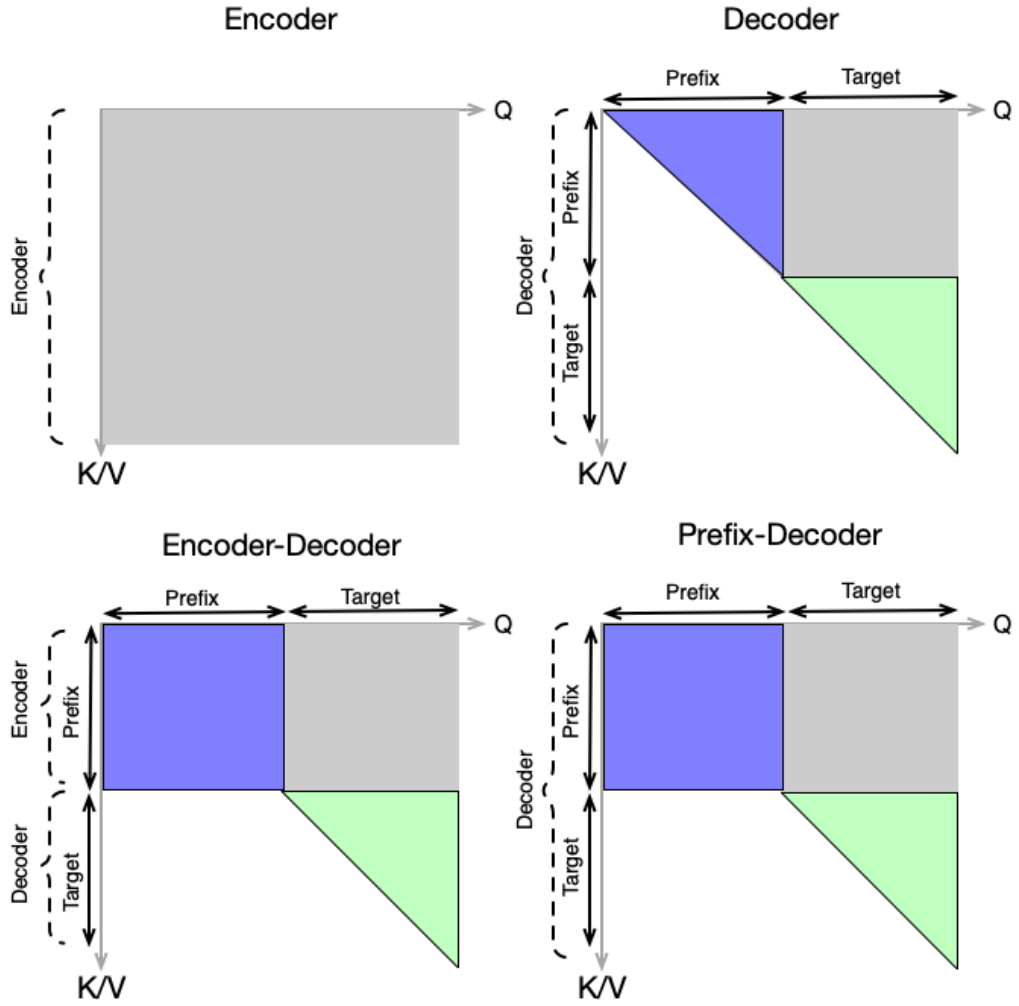
Chinchilla: Training Compute-Optimal Large Language Models, 2022, Google

Pretraining

- Training objectives
- Text Corpus
- Parallel strategies
- Results & Evaluation

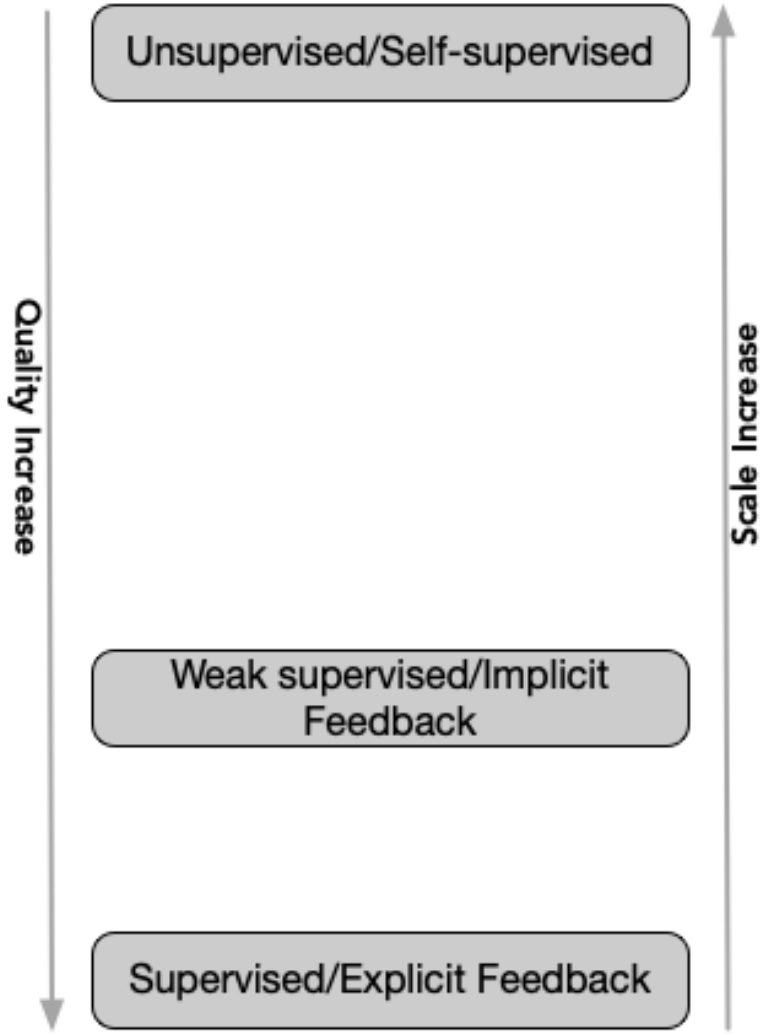
Training architecture & objectives

- Architecture
  - Encoder: BERT series
  - Encoder-Decoder: T5(11B)
  - (Causal-)Decoder: GPTs/Ernie3.0 Titan
  - Prefix-Decoder: GLM
- Objectives
  - Masked LM: BERT/GLM/T5
  - Auto-regressive LM: T5/GPTs/Ernie3.0 Titan
  - Multi-task pretraining: GLM/Ernie3.0 Titan



Text Corpus

- Unsupervised text
  - BookCorpus : 11,000; Gutenberg : 70,000
  - OpenWebText: 8M outlinks of Reddit.com
  - Common Crawl; C4
  - Code: BigQuery Github
- Weak supervised text
  - Reddit TL;DR; PushShift.io Reddit: Posts
  - StackExchange : Question & Answer w/ score
- Supervised text: task related
  - ~16 NLP tasks related datasets (Sentiment/QA/Reasoning, etc.)
  - Human answer to prompt: InstructGPT/ShareGPT



Baidu Ernie 3.0 Titan: 260B

- 2 Types of transformer modules
  - Universal module
  - Task specific module
- 3 Levels of pretraining tasks
  - Word aware: Knowledge integrated LM
  - Structure aware: sentence reordering task
  - Knowledge aware: controllable LM task
- 4D hybrid parallelism for training
  - Shared data parallel with ZeRO(2D)
  - Intra-layer tensor parallel(D)
  - Inter-layer pipeline parallel(D)

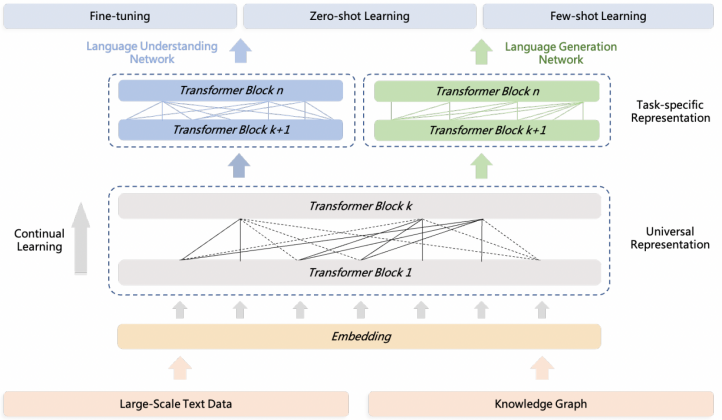


Figure 1: The framework of ERNIE 3.0.

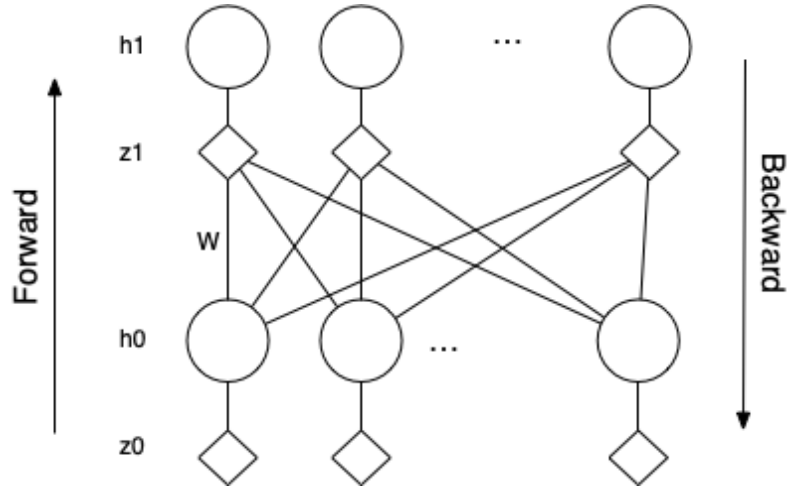
Parallel strategies

- Why bother?
- Three parallel paradigms
  - Data parallel: ZeRO
  - Model parallel
    - Tensor parallel: Megatron-LM
    - Pipeline parallel: GPipe
- Combined implementations: DeepSpeed/ColossalAI

Why bother?

- Too big to fit in single GPU memory
  - 175B: $\sim (2+2+3\times 4)\times 175 = 2800$GB for mixed-precision training
    - Parameter & gradient(FP16): parameter(W), gradient(g, $\frac{\partial L}{\partial W}$)
    - Optimizer State(FP32): parameter, momentum(m), variance(v)
    - Activation(FP16): $2\times(1+4+1)\times d\times B\times T\times L$
  - A100 Spec:
    - GPU memory: 80GB
    - GPU memory bandwidth: 2039GB/s; NVLink: 600GB/s; PCIe 4.0: 64GB/s
    - TF32: 156TFlops
- Speedup
  - Scales linearly with # of GPU cores?

Basics on Forward & Backward and Parallel Ops

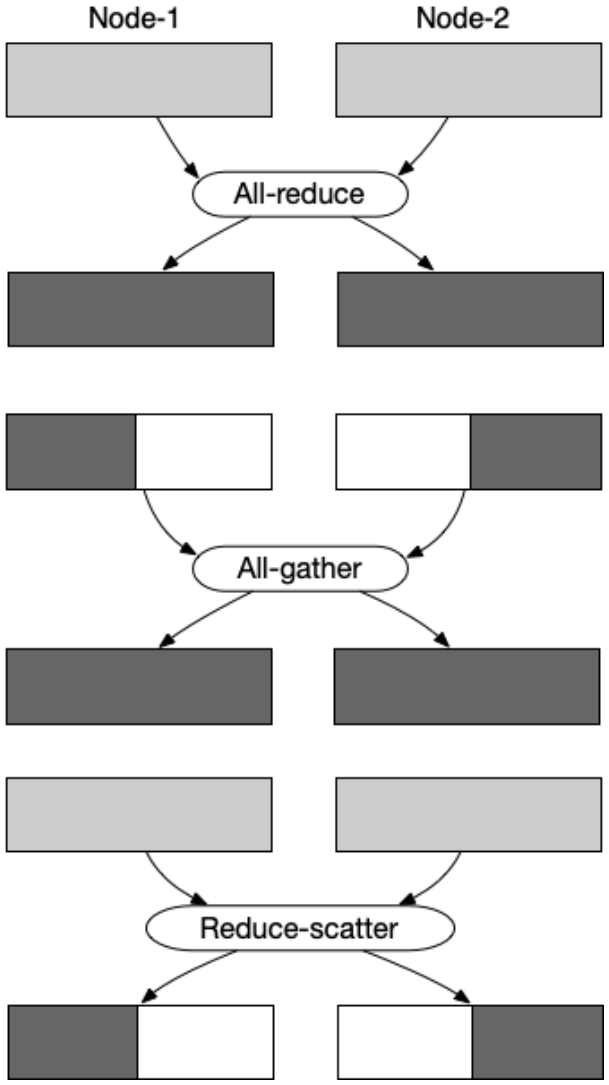


Forward:

$$h_0 = \sigma(z_0)$$
$$z_1 = W^T h_0 + b$$
$$h_1 = \sigma(z_1)$$

Backward:

$$(\frac{\partial L}{\partial h_1}, W) \rightarrow \frac{\partial L}{\partial h_0}$$
$$(\frac{\partial L}{\partial h_1}, h_0) \rightarrow \Delta \frac{\partial L}{\partial W}$$
$$m \leftarrow \beta_1 m + (1-\beta_1)\frac{\partial L}{\partial W}$$
$$v \leftarrow \beta_2 v + (1-\beta_2)\left(\frac{\partial L}{\partial W}\right)^2$$
$$W \leftarrow W - \frac{\alpha}{\sqrt{\hat{v}}+\epsilon}\hat{m}$$



Data parallel: from DDP to FSDP(ZeRO)

- *Pesudo code* for DDP and FSDP

```
# forward pass :
for layer_i in layers:
  forward pass for layer_i
# backward pass :
for layer_i in layers:
  backward pass for layer_i
  full: all-reduce gradients for layer_i
  full: update momentum & variance
  full: update weights
```

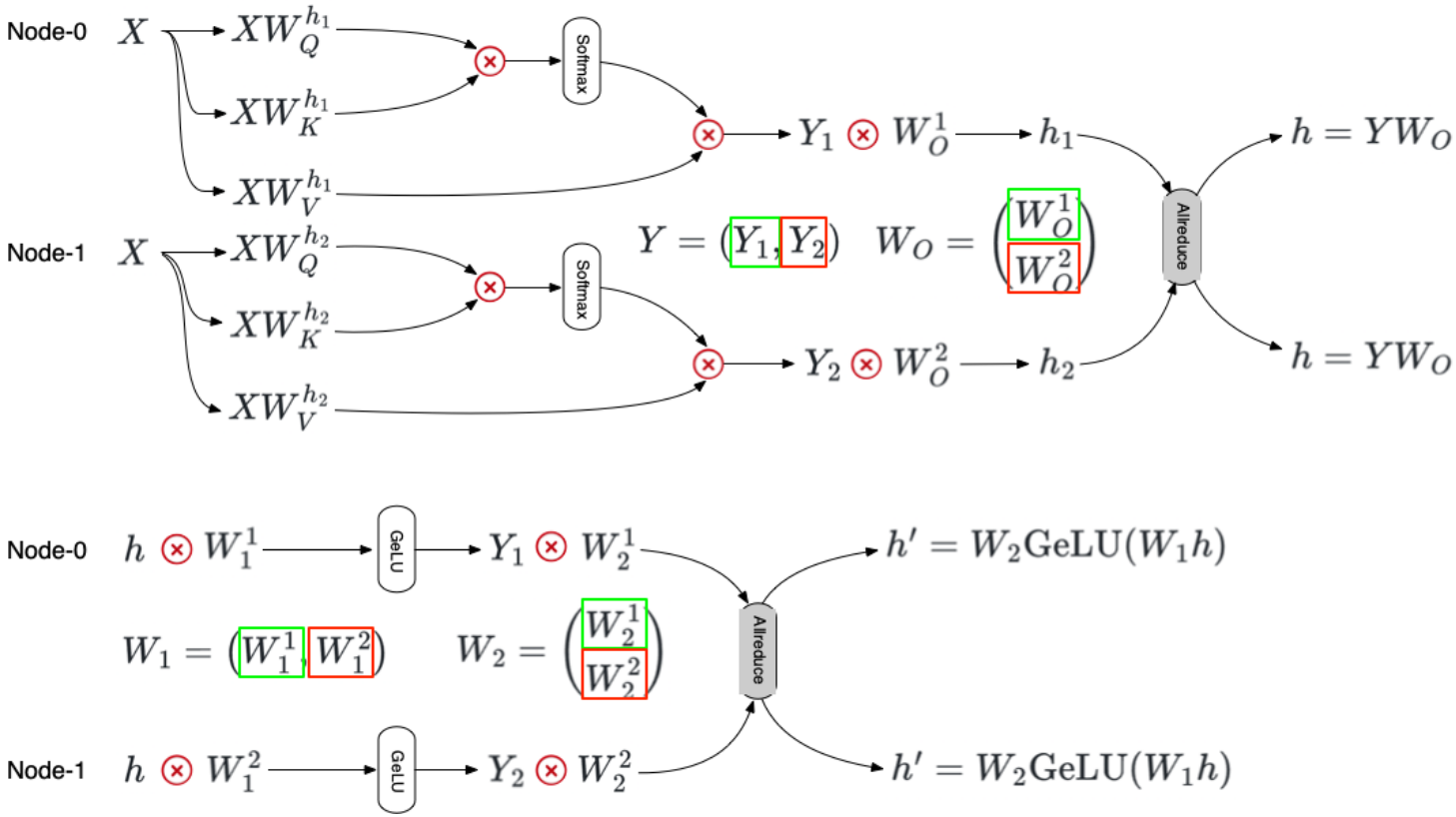
```
# forward pass :
for layer_i in layers:
  all-gather full weights for layer_i
  forward pass for layer_i
  discard full weights for layer_i
# backward pass:
for layer_i in layers:
  all-gather full weights for layer_i
  backward pass for layer_i
  discard full weights for layer_i
  part: reduce-scatter gradients for layer_i
  part: update momentum & variance
  part: update weights
```

- Advantages of ZeRO
  - Parameter & gradient and optimizer states evenly shard to N nodes
  - Computation and communication overlaps

ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Microsoft, 2019

Tensor parallel: Megatron-LM

- W_Q, W_K, W_V partition by head(col)
- W_O partition by row
- W_1 partition by col, W_2 by row
- Backward Allreduce for gradient

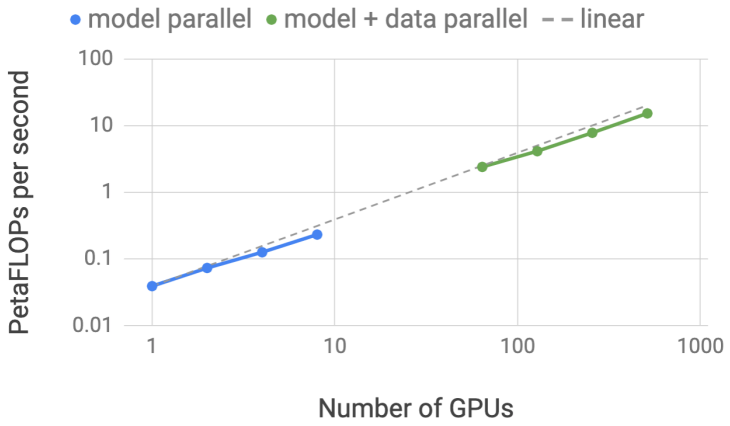


Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, Nvida, 2019

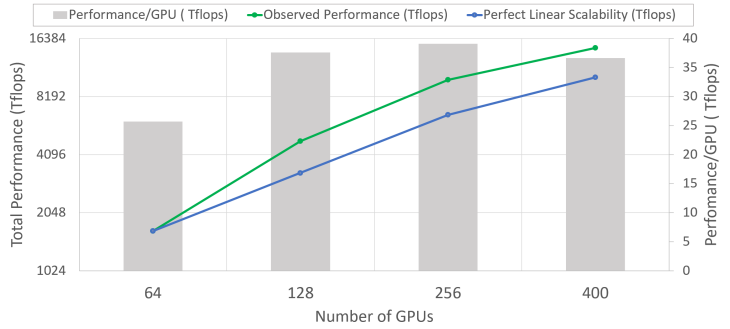
Performance comparison of Megatron-LM and ZeRO

- Experiment hardware:
  - **Megatron-LM**: 32 DGX-2H servers: 512 V100, 32GB
  - **ZeRO**: 25 DGX-2 servers: 400 V100, 32GB
- Flops and MFU:
  - **Megatron-LM**: 15.1 PFlops, 8.3B Model
    - 29.5 TFlops/GPU (=76% × 39 TFlops single GPU, 30% of peak Flops)
    - **MFU=76% × 30%=22.8%**
  - **ZeRO**: 15 PFlops, 100B Model
    - 38 TFlops/GPU

Speedup for **Megatron-ML**:

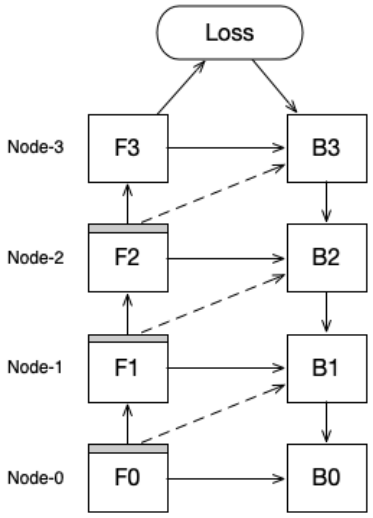


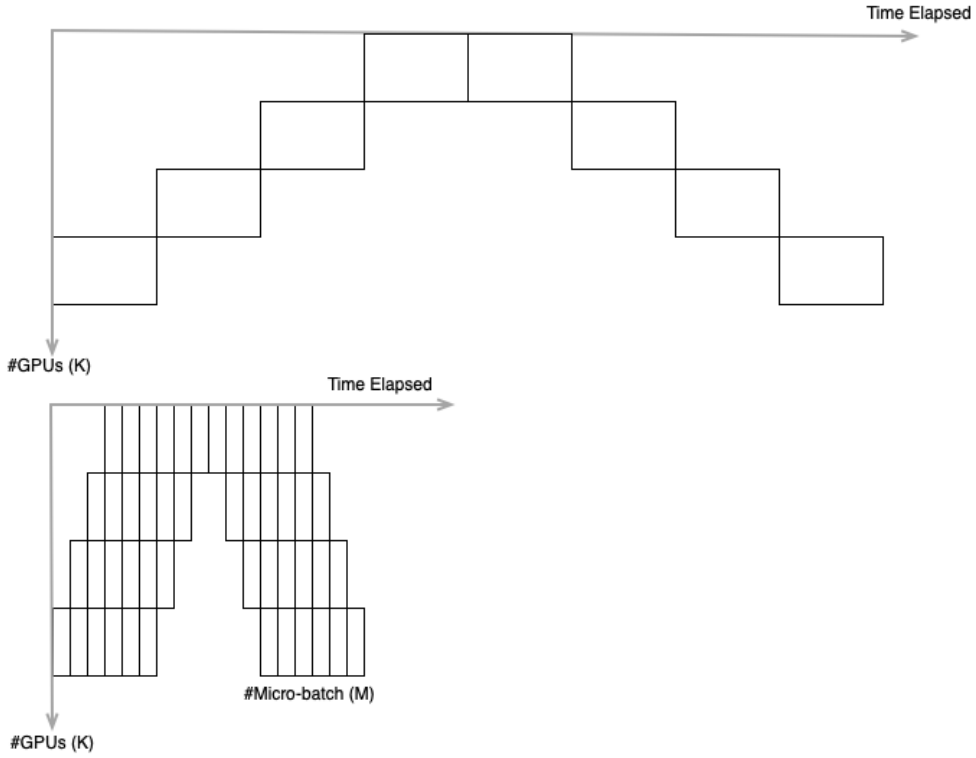
Speedup for **ZeRO**:



Pipeline parallel: GPipe

- Layer-wise model partition
- Re-materializaion: output activation stored and communicated
- Pipeline reduce bubble ratio from $\frac{K-1}{K}$ to $\frac{K-1}{M+K-1}$





GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, Google, 2018

Combined implementations: Megatron-DeepSpeed/ColossalAI

Megatron-DeepSpeed

- Data/Model parallel supported(3D)
- ZeRO-Offload
- Sparse attention
- 1-bit Adam and 0/1 Adam
- MoE specific parallelism
- RLHF Demo: deepspeed-chat

ColossalAI

- Data/Model parallel supported(3D)
- 3D Tensor parallelism
- ZeRO-Offload: model data supported
- MoE specific parallelism
- RLHF Demo: ColossalChat

Colossal-AI: A Unified Deep Learning System For Large-Scale Parallel Training, HPC-AI, 2021
Megatron-DeepSpeed

End of Parallel strategies

Result & Evaluations

Evaluation Set

- GLUE/SuperGLUE
- MMLU
- C-Eval
- AGIEval
- GSM8k/MATH
- HumanEval(code)
- Big-bench(Hard)

C-Eval Accuracy: YY: 47+

Model	Average
GPT-4	69.9
Claude-v1.3	55.5
ChatGPT	53.5
Claude-instant-v1.0	47.4
GLM-130B	40.8
LLaMA-65B	39.8
Bloomz-mt	38.0
ChatGLM-6B	37.1
Chinese-LLaMA-13B	33.1
MOSS	28.9
Chinese-Alpaca-13B	27.2

Holistic Evaluation of Language Models, 2022, Stanford

C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models, 2023, SJTU

Finetuning

- Target and issues
- Instruct finetuning
- Finetuning for specific task
- Parameter efficient finetuning

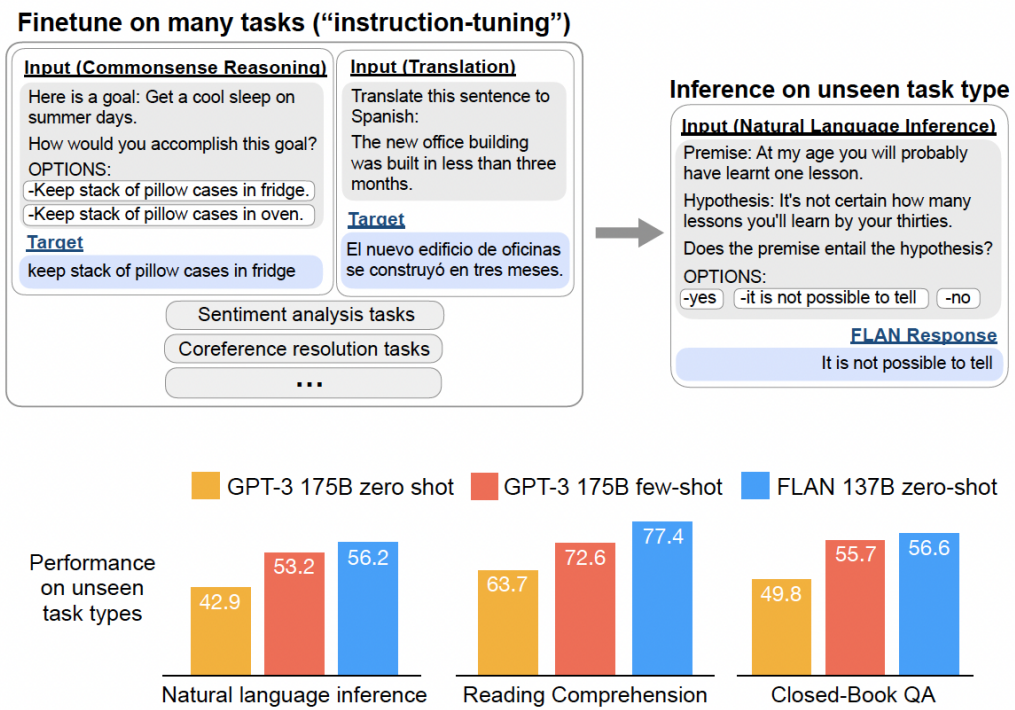
Target and issues

- Target
  - From pattern completion to real world task completion
- Issues
  - Instruction following
  - Hallucination
  - Toxicity and ethics
  - Securities

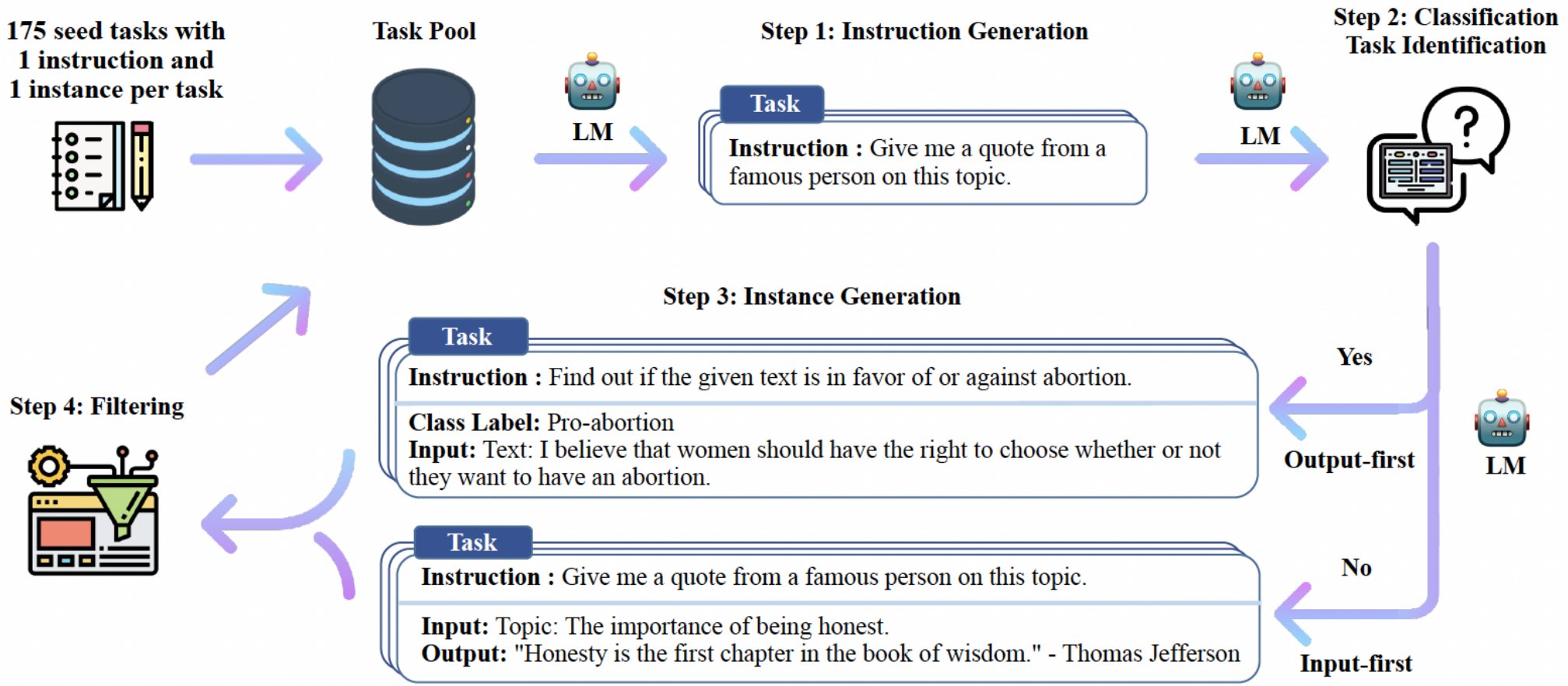
Instruction Finetuning

Key to success

- Number of finetuning datasets:
  - scaling from 62 text datasets to 18K
- Model scale: 137B LaMDA-PT
- Natural language instructions



Aligning Language Models with Self-Generated Instructions: Self-instruct



Self-Instruct: Aligning Language Models with Self-Generated Instructions, 2022, University of Washington

Distill Large Foundation Model(LFM) with instruction tuning: LLaMA series

Model Name	#Instructions	#(Base Model)	Finetuning	Instruction Synthetic	Org.
Alpaca	52K	7B	Full	Self-instruct	Stanford
Vicuna	70K	13B	Full	ShareGPT	LM
Dolly	15K	12B	Full	Human	Databricks
WizardLM	250K	16B	Full	Evol-Instuct	Microsoft
Guanaco	88K	65B	QLoRA	OASST1(dialog)	UW

QLoRA: Efficient Finetuning of Quantized LLMs, 2023, University of Washington

WizardLM: Empowering Large Language Models to Follow Complex Instructions, 2023, Microsoft

Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality

Alpaca: A Strong, Replicable Instruction-Following Model, 2023, Stanford

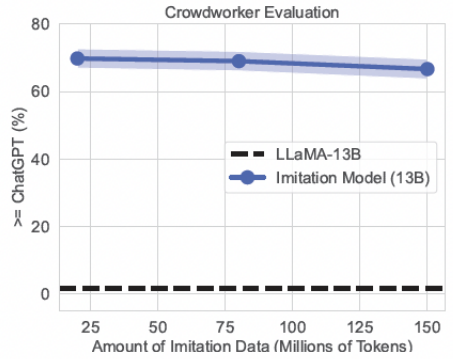
Different voice from UC Berkeley

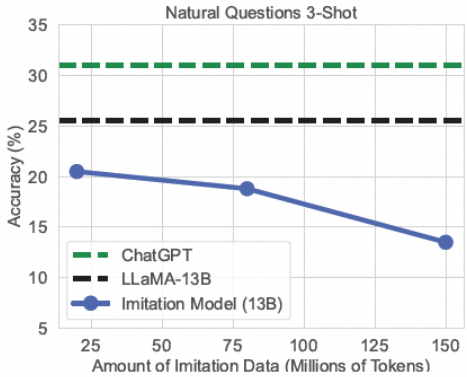
Method and evaluation in brief:

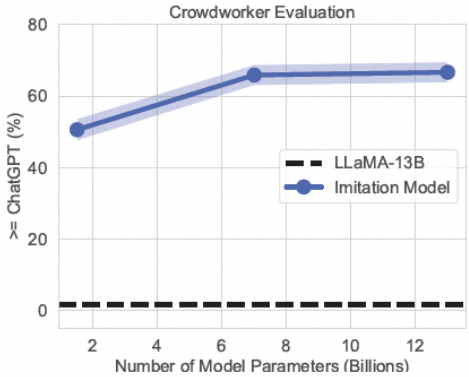
- Base model size from 1.5B(**GPT-2**) to 13B(**LLaMA**)
- Imitation tokens from 0.3M to 150M(**NQ-synthetic** + **ShareGPT**)
- Evaluation: Crowdworkers + NLP tasks(Auto)

Conclusion:

- Imitation models are rated high by crowdworkers(**top**)
- Scaling imitation data not work on 3-shot NQ(**mid.**)
- Scaling base model parameters size helps(**bottom**)
- Mimicks ChatGPT's style not facturality capability
- Enormous and diverse imitation dataset required







The False Promise of Imitating Proprietary LLMs, 2023, UC Berkeley

Scaling the size of instructions from LFM: Orca

- Enriching Imitation signal:
  - explanation traces
  - step-by-step thought
- Scaling instructions
  - Initiated from FLAN-v2 collection
  - 5M instructions from ChatGPT(TA)
  - 1M instructions from GPT-4
- Rigorous Evaluations
  - Open-ended Generation:
    - Vicuna/Awesome/WizardLM Prompts
  - Reasoning: AGIEval/Big-bench Hard/
  - Safety: TruthfulQA-MC

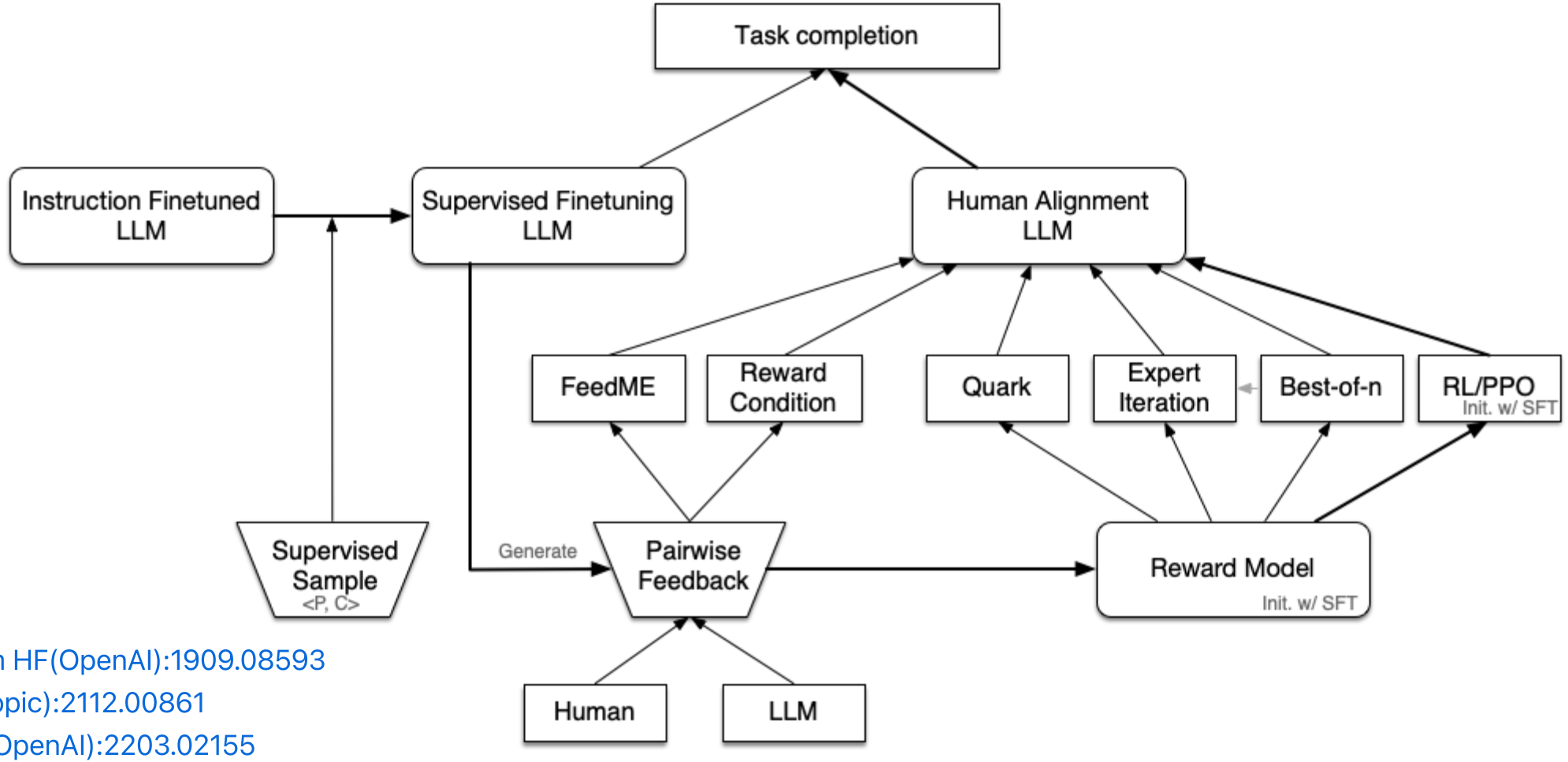
Dataset	Reference	Vicuna-13B	Orca-13B
Vicuna Prompts	ChatGPT GPT-4	92 73.8	**101.5** (10.4%) **87.7** (18.9%)
Awesome Prompts	ChatGPT GPT-4	86.5 77.8	**98.1** (13.5%) **89.3** (14.9%)
WizardLM Prompts	ChatGPT GPT-4	77.1 69.1	**84.9** (10.1%) **78.4** (13.5%)
Average	ChatGPT GPT-4	85.2 73.6	**94.8** (11.3%) **85.1** (13.5%)

Conclusion:

- **Explanation Tuning** effectively aligns smaller model to GPT-4
- Imitation **data size** and **coverage** is crucial

Orca: Progressive Learning from Complex Explanation Traces of GPT-4, 2023, Microsoft

Finetuning for specific task



Finetune from HF(OpenAI):1909.08593
HHHA(Anthropic):2112.00861
InstructGPT(OpenAI):2203.02155
HHA(Anthropic):2204.05862
AlpacaFarm: 2305.14378

Problems with Supervised Finetuning(SFT)

- Learning only the **task format** and the **way to response** for the format
- Knowledge labeled but not in the LLM leads to more hallucination
- Knowledge in the LLM but labeled as `don't know` leads to withhold information

What we want from finetuning:

> outputs its(LLM's) state of knowledge with the correct amount of hedging and expressing its uncertainty

Advance of RL to SFT for truthfullness

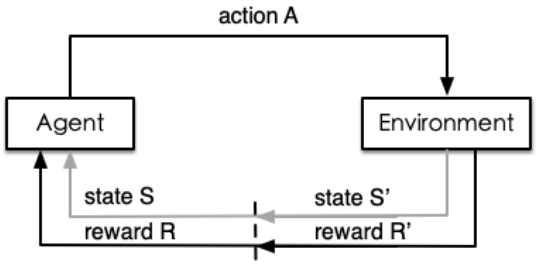
- LLMs know what they know
  - Calibrated probability, uncertainty
- RLHF can leverage the self-awareness
  - Design reward function: `correct answer=1`, `don't know=0` and `wrong answer=-4`
  - RL learn optimal threshold of probability to maximize the reward
- No oracle for the correct reward, delegate to Reward Model
  - Reward model: **relative criteria** trained by pairwise loss from human feedback
  - Open problem: true probabilites of everything?
  - Open problem: go beyond things that labelers can easily do
    - Verification is easier than generation

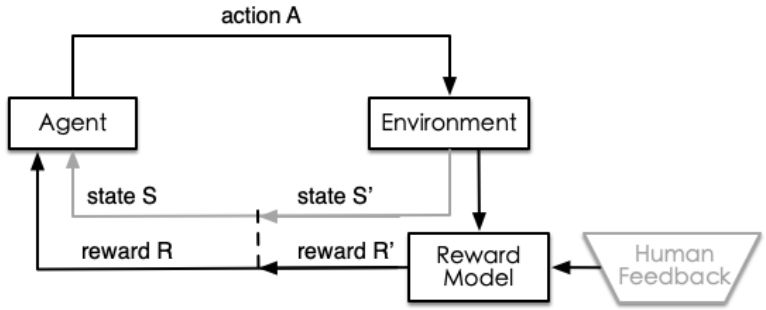
Language Models (Mostly) Know What They Know, 2022, Anthropic

Talk of John Schulman @ EECS Berkeley, 2023/4/24

More on Reinforcement Learning

- Catalog of algorithms(**PPO belongs**)
  - World model or **model free**
  - Value-based or **policy-based(actor-critic)**
  - MC or **TD bootstrapping**
  - Off-policy or **on-policy**
  - Deterministic or **stochastic** policy
- Design consideration
  - **Sample efficiency**: Off-policy > On-policy
  - **Stability & Convergence**: Deadly Triad issue
  - **Explore & Exploit**: Random at episode beginning
  - **Bias & Variance**: Advantage Function



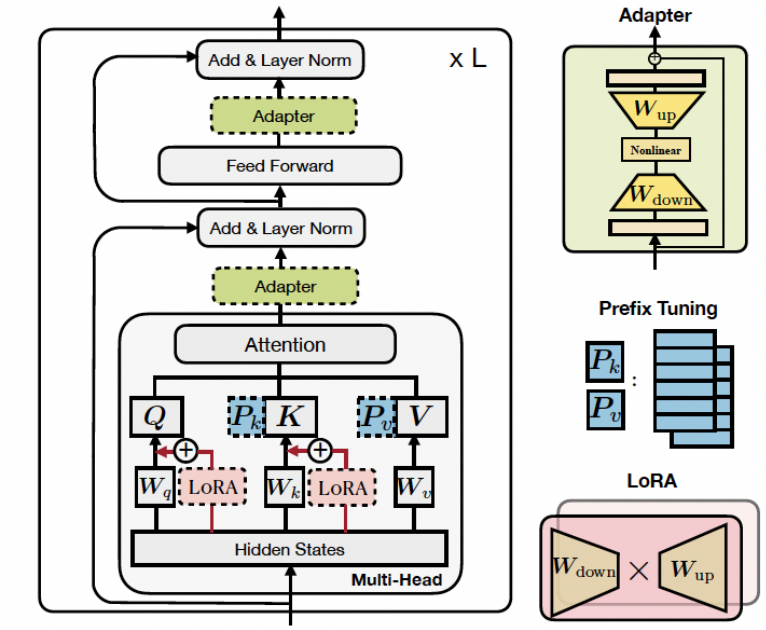


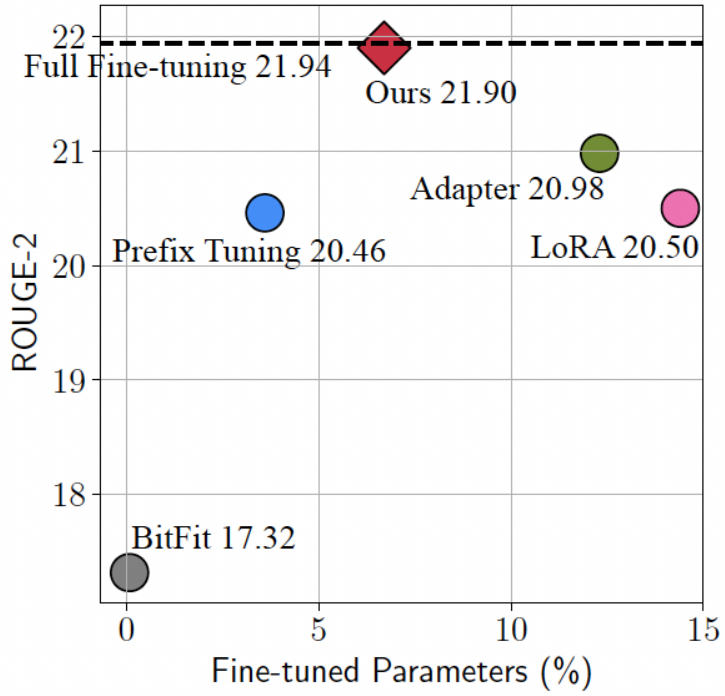
SpinningUp: OpenAI

Parameter Efficient Finetuning

- Design aspects
- Primary implementations
  - Adapter
  - Prefix-tuning
  - LoRA

Methods and performance on Summerization task





PEFT illustration and performance comparison (Source: Junxian He, et.al)

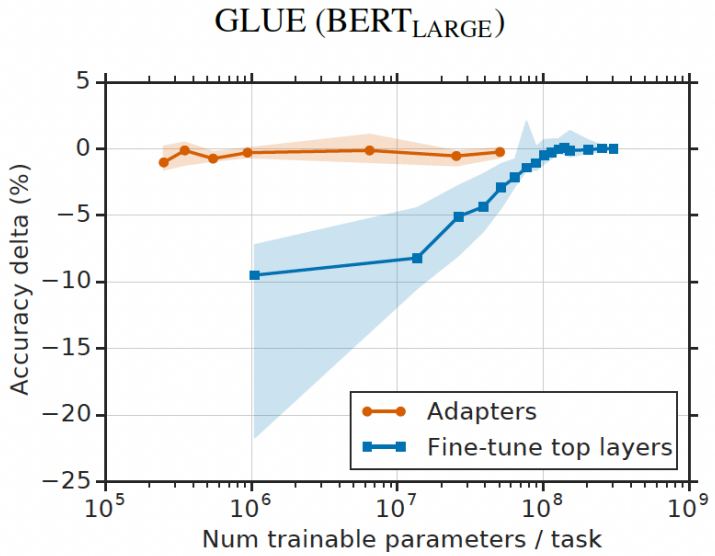
Design aspects

- **Modules Finetuned**:
  - Attention-key/value matrix: LoRA(Q/V)
  - Attention-head: Prefix-Tuning(K/V)
  - Attention: Adapter
  - After FFN: Adapter
- **Other aspects**:
  - Multi-task consideration
  - Task related head

Adapter

- Implementation & training notes
  - $h \leftarrow h + f(hW_{\text{down}})W_{\text{up}}$
  - Parameter scale:
    $2 \times L \times (r \times d + r + d), r \ll d$
  - Adapt after FFN sub-layer works too
- Results
  - Finetune BERT for 26 classification Tasks
    - 3.6% parameters for 0.4% GLUE performance gap
  - Ablation: fewer layers adapted -> worser performance

Accuracy relative to full-tuning:



Adapter: Parameter-Efficient Transfer Learning for NLP, Google, 2019

Prefix-Tuning

- Implementation
  - $\text{head} = \text{Attn}(XW_Q, [P_K; XW_K], [P_V; XW_V])$
  - Reparameterization for finetuning stability:
    - $[P_K, P_V] = \text{MLP}_\theta(P^E)$
    - P^E: prefix embedding
- Parameter scale:
  - Vanilla: $|P| \times d \times 2 \times L$
  - Reparameteration: $|P| \times d + d \times H + H \times d \times 2 \times L$

PrefixTuning: Optimizing Continuous Prompts for Generation, Stanford, 2021

More on Prefix-Tuning: Training and scaling

- Training
  - **Initialization**:
    - Real/high frequency words activation
    - *Task relevant words | Classification labels*
  - **LM Head**: Next Token/Class Label
- Results & discussion
  - Finetuning $0.1\% \sim 3\%$ parameters, comparable or better performance
  - Optimal prefix length varies: longer for more complex tasks
  - Reparameterization works task-dependently

PrefixTuning: Optimizing Continuous Prompts for Generation, Stanford, 2021
PromptTuning: The Power of Scale for Parameter-Efficient Prompt Tuning, Google, 2021
P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks, Tsinghua, 2021

LoRA

- Implementation & training notes
  - **Low-Rank**: $W = W_0 + (BA)^T, A \in R^{r \times d}, B \in R^{d_h \times r}, r \ll \min\{d_h, d\}$
  - W_Q and W_V considered, parameter scale: $2(AB) \times 2(QV) \times d \times r \times L$
  - **Modularized**: `Embedding` , `Linear` , `MergedLinear` , `Conv2D`
  - **Initialization**: A kaiming-random, B zeros
  - **Weight merged** for inference efficiency
- Results
  - For 175B GPT-3 finetuning: 0.01% parameters , on par or better results
  - No additional inference computation and latency
  - Additivity for **finetuning merge** and **incremental update**

LoRA: Low-Rank Adaptation of Large Language Models, Microsoft, 2021

More on LoRA: which part to update & rank settings

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

LoRA: Low-Rank Adaptation of Large Language Models, Microsoft, 2021

End of Parameter Efficient Finetuning

Steering the decoding process of LLM

- Decoding strategies
- Prompt enginneering

Decoding strategies

- Temperature in decoding: $p(w_i|w_{<i}) = \frac{\exp(o_i/T)}{\sum_j \exp(o_j/T)}$, o_i logits from LLM
- Maximal Likelihood Search
  - Greedy search: $w_i = \arg\max p(w_i|w_{<i})$, eq. to $T = 0$
  - Beam search: $w_i \in \text{TopN}\ p(w_i|w_{i-1}, w_{<i-1})p(w_{i-1}|w_{<i-1})$
- Sampling
  - top-K sampling: $w_i = \text{sample TopK}\ p(w_i|w_{<i})$
  - top-p(Nucleus) sampling: $w_i = \text{sample TopK}_i\ \sum_{w_i<K_i} p(w_i|w_{<i}) \geq p$
  - Repetition penalized sampling
- Guided decoding
  - $score(x_{t+1}, b_t) = score(b_t) + \log p(x_{t+1}) + \sum_i \alpha_i f_i(x_{t+1})$

Prompt engineering

- One-shot interaction
  - Instruction/Zero-shot Prompt
  - Few-shot Prompt
  - In-context Learning(Prompt)
  - Chain-of-Thought
- Recursive interaction
  - MRKL: `[Thought/Action/Action Input/Observation]+`, zero-shot, access tools
  - **Self-ask**: `Followup question? [Question/Answer]+ Final Answer`, few-shot
  - **ReACT**: `[Thought/Action/Observation]+`, few-shot, access tools

More on In-context Learning

What matters?

- Examples template(instruct/CoT) & order: yes
- Label space & Input distribution(diversity): yes
- Exact `{Question, Answer}` pair: no/yes

Why it works?

- Interpretation from Topic Model: $P(o|p) = \int_z P(o|z,p)P(z|p)dz$
- Induction head: `[a][b] ... [a] -> [b]`

A Mathematical Framework for Transformer Circuits, 2021, Anthropic
How does in-context learning work?, 2022, Stanford
Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?, 2022, Meta
Larger language models do in-context learning differently, 2023, Google

Augmentation and Plugins

- Augmented Language Models
- Plugins Ecosystem

Augmented Language Models

- Pros and Cons in vanilla LLMs
- Augmentation
  - Reasoning/Planning
  - Action/Tool usage

Pros and Cons in vanilla(no-augmented) LLMs

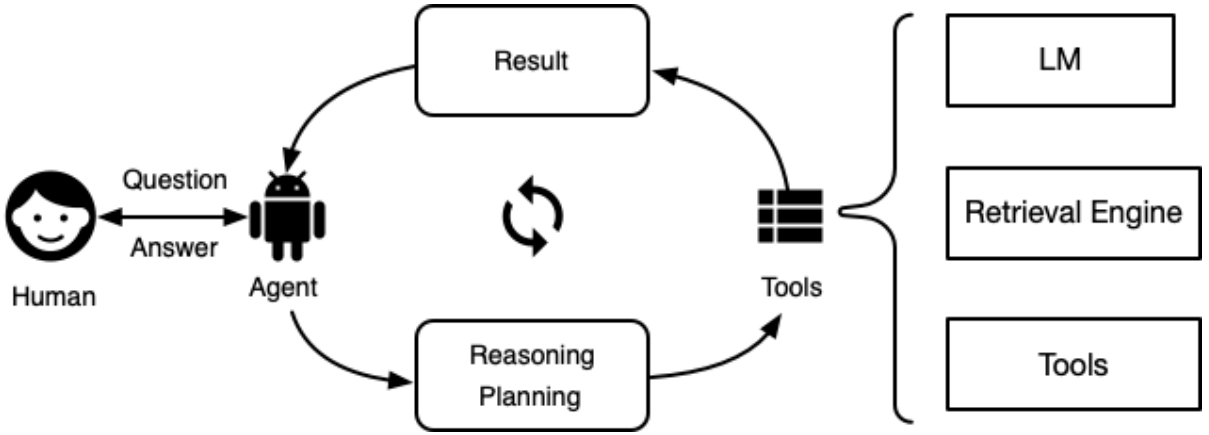
Pros

- World knowledge
- Instruction Following
- Few/Zero-shot Learning
- In-context Learning
- Chain-of-Thought
- Arithmetic/Commonsense Reasoning

Cons

- Compression of world knowledge
- No up-to-date knowledge
- Context limitation
- Hallucinations

Augmented Language Model Brief Overview



- 2-step loops in augmenting
  - What: Reasoning/Planning
  - How: Action/Tool Usage
- Equiped with **memory**: Short/Long Term

- 3 methods for augmenting
  - In-context prompt:
    - One-shot/Recursive
  - Pretraining/Finetuning
    - Human/Self-teach
  - Reinforcement Learning:
    - Hardcode/Human-Feedback

Augmented Language Models: a Survey, 2023, Meta
Tool Learning with Foundation Models, 2023, Tsinghua

Planning by self-talk: Self-ask

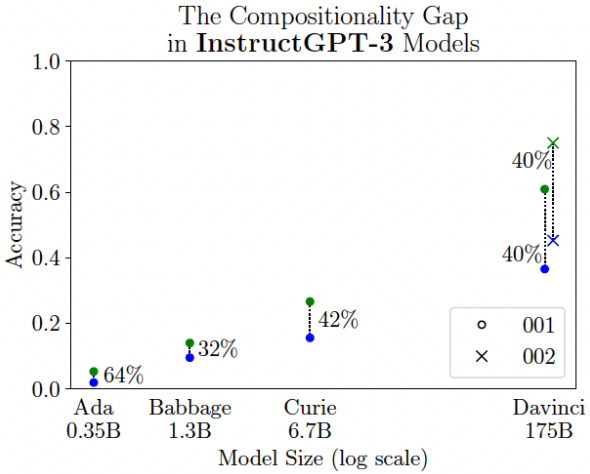
- Multi-hop question and compositionality gap
- Self-ask prompt:
  - Few-shot prompt scaffold:

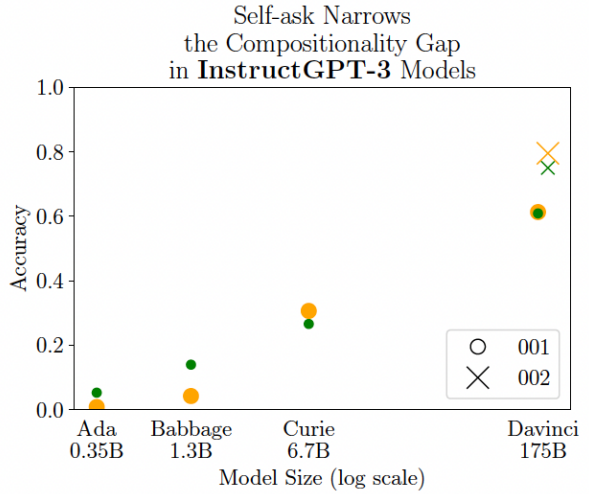
```
Question: {{Question}}
Are follow up questions needed here: Yes
Follow up: {{Follow-up-question}}
Intermediate answer: {{Itermediate-answer}}
So the final answer is: {{Final-answer}}

Question: {{Question}}
Are follow up questions needed here: {{LLM-gen}}
```

  - Few-shot prompt with search engine:
    - The same prompt as before
    - Generated by query search engine:

```
Intermediate answer: {{Itermediate-answer}}
```

Accuracy on *Compositional Celebrities* 2-hop questions





Self-ask: Measuring and narrowing the compositionality gap in language models, 2022, MetaAI

Planning by self-talk: Self-ask prompt

> **Question:** Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
> **Are follow up questions needed here: Yes.**
> **Follow up:** How old was Theodor Haecker when he died?
> **Intermediate answer:** Theodor Haecker was 65 years old when he died.
> **Follow up:** How old was Harry Vaughan Watkins when he died?
> **Intermediate answer:** Harry Vaughan Watkins was 69 years old when he died.
> **So the final answer is:** Harry Vaughan Watkins

> **Question:** Who was president of the U.S. when superconductivity was discovered?
> **Are follow up questions needed here: Yes.**
> **Follow up:** When was superconductivity discovered?
> **Intermediate answer:** Superconductivity was discovered in 1911.
> **Follow up:** Who was president of the U.S. in 1911?
> **Intermediate answer:** William Howard Taft.
> **So the final answer is:** William Howard Taft.

Planning by self-talk: Self-ask prompt with search engine

> **Question:** Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
> **Are follow up questions needed here: Yes.**
> **Follow up:** How old was Theodor Haecker when he died?
> **Intermediate answer:** Theodor Haecker was 65 years old when he died.
> **Follow up:** How old was Harry Vaughan Watkins when he died?
> **Intermediate answer:** Harry Vaughan Watkins was 69 years old when he died.
> **So the final answer is:** Harry Vaughan Watkins

> **Question:** Who was president of the U.S. when superconductivity was discovered?
> **Are follow up questions needed here: Yes.**
> **Follow up:** When was superconductivity discovered?
> **Intermediate answer:** Superconductivity was discovered in 1911.
> **Follow up:** Who was president of the U.S. in 1911?
> **Intermediate answer:** William Howard Taft.
> **So the final answer is:** William Howard Taft.

Synergy between reasoning and actions: ReAct prompt with tools

- Two kinds of tasks
  - **Reasoning**: multi-hop QA(HotspotQA)
  - **Decision making**: long horizon/delayed reward(WebShop)
- Space of reasoning and actions
  - **reasoning**: language generated by LLM
    - goal or question decompose
    - commonsense or arthmetic reasoning
    - search reformulation
    - answer synthesis
  - **actions**: task related
    - search/find/finish for reasoning

- ReAct $\leq$ CoT-SC
- CoT-SC -> ReAct $>$ CoT-SC
- ReAct -> CoT-SC $>$ CoT-SC

Prompt Method[a]	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	**64.6**
ReAct → CoT-SC	**35.1**	62.0
Supervised SoTA[b]	67.5	89.5

Table 1: PaLM-540B prompting results on HotpotQA and Fever.

ReAct: Synergizing Reasoning and Acting in Language Models, 2022, Princeton

CoT prompt

> **Question** Musician and satirist Allie Goertz wrote a song about the "The Simpsons" character Milhouse, who Matt Groening named after who?
> **Thought** Let's think step by step. Milhouse was named after U.S. president Richard Nixon, so the answer is Richard Nixon.
> **Answer** Richard Nixon

> **Question** Which documentary is about Finnish rock groups, Adam Clayton Powell or The Saimaa Gesture?
> **Thought** Let's think step by step. Adam Clayton Powell (film) is a documentary about an African-American politician, not Finnish rock groups. So the documentary about Finnish rock groups must instead be The Saimaa Gesture.
> **Answer** The Saimaa Gesture

> **Question** What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
> **Thought** Let's think step by step. The eastern sector of Colorado orogeny extends into the High Plains. High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.
> **Answer** 1,800 to 7,000 ft

Act only prompt

> {{Few-shot examples ahead}}
> **Question:** What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
> **Action 1** Search[Colorado orogeny]
> **Observation 1** The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas.
> **Action 2** Lookup[eastern sector]
> **Observation 2** (Result 1 / 1) The eastern sector extends into the High Plains and is called the Central Plains orogeny.
> **Action 3** Search[High Plains]
> **Observation 3** High Plains refers to one of two distinct land regions:
> **Action 4** Search[High Plains (United States)]
> **Observation 4** The High Plains are a subregion of the Great Plains. From east to west, the High Plains rise in elevation from around 1,800 to 7,000 ft (550 to 2,130m).[3]
> **Action 5** Finish[1,800 to 7,000 ft]

ReAct prompt

> {{Few-shot examples ahead}}
> **Question** What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
> **Thought 1** I need to search Colorado orogeny, find the area that the eastern sector of the Colorado orogeny extends into, then find the elevation range of the area.
> **Action 1** Search[Colorado orogeny]
> **Observation 1** The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas.
> **Thought 2** It does not mention the eastern sector. So I need to look up eastern sector.
> **Action 2** Lookup[eastern sector]
> **Observation 2** (Result 1 / 1) The eastern sector extends into the High Plains and is called the Central Plains orogeny.
> **Thought 3** The eastern sector of Colorado orogeny extends into the High Plains. So I need to search High Plains and find its elevation range.
> **Action 4** Search[High Plains (United States)]
> **Observation 4** The High Plains are a subregion of the Great Plains. From east to west, the High Plains rise in elevation from around 1,800 to 7,000 ft (550 to 2,130m).[3]
> **Thought 5** High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.
> **Action 5** Finish[1,800 to 7,000 ft]

Pros and Cons of Prompt-based augmentation

Pros

- Flexible for application
- Add new tool in low cost
- Easy to explanation

Cons

- Unstable response
- Rely on top performed LLMs
- Limited by context window size
- Token inefficient

Augment LM with retrieval end-to-end: REALM(pretraining)

- Two steps of *retrieve* and *predict*
  - $P(y|x) = \sum_z P(y|z,x)P(z|x)$
  - *Neural retrieval*: $P(z|x) = \mathrm{Softmax}_z(\mathrm{BERT}_{\mathrm{CLS}}(x)^T \mathrm{BERT}_{\mathrm{CLS}}(z))$
  - *Knowledge-augmented encoder*: $P(y|z,x)$
    - Pretraining: Masked Language Model loss of BERT
    - Finetuning: Reading Comprehension loss of BERT(assume $y \in z$)
- Practical training
  - Involve top-K documents of $P(z|x)$
  - Vector search of top-K with asynchronous index update

REALM: Retrieval-Augmented Language Model Pre-Training, 2020, Google

Augment LM with retrieval end-to-end: REALM (Demo/Ablation)

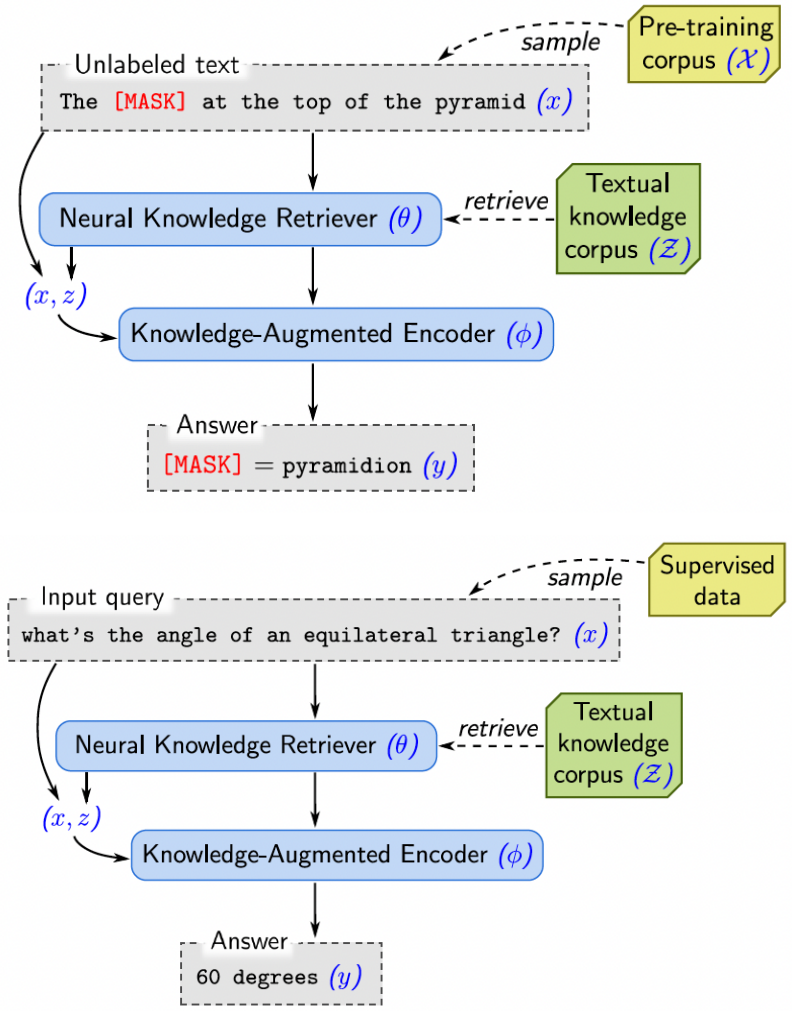


Table 2. Ablation experiments on NQ's development set.

Ablation	Exact Match	Zero-shot Retrieval Recall@5
REALM	38.2	38.5
REALM retriever+Baseline encoder	37.4	38.5
Baseline retriever+REALM encoder	35.3	13.9
Baseline (ORQA)	31.3	13.9
REALM with random uniform masks	32.3	24.2
REALM with random span masks	35.3	26.1
30× stale MIPS	28.7	15.1

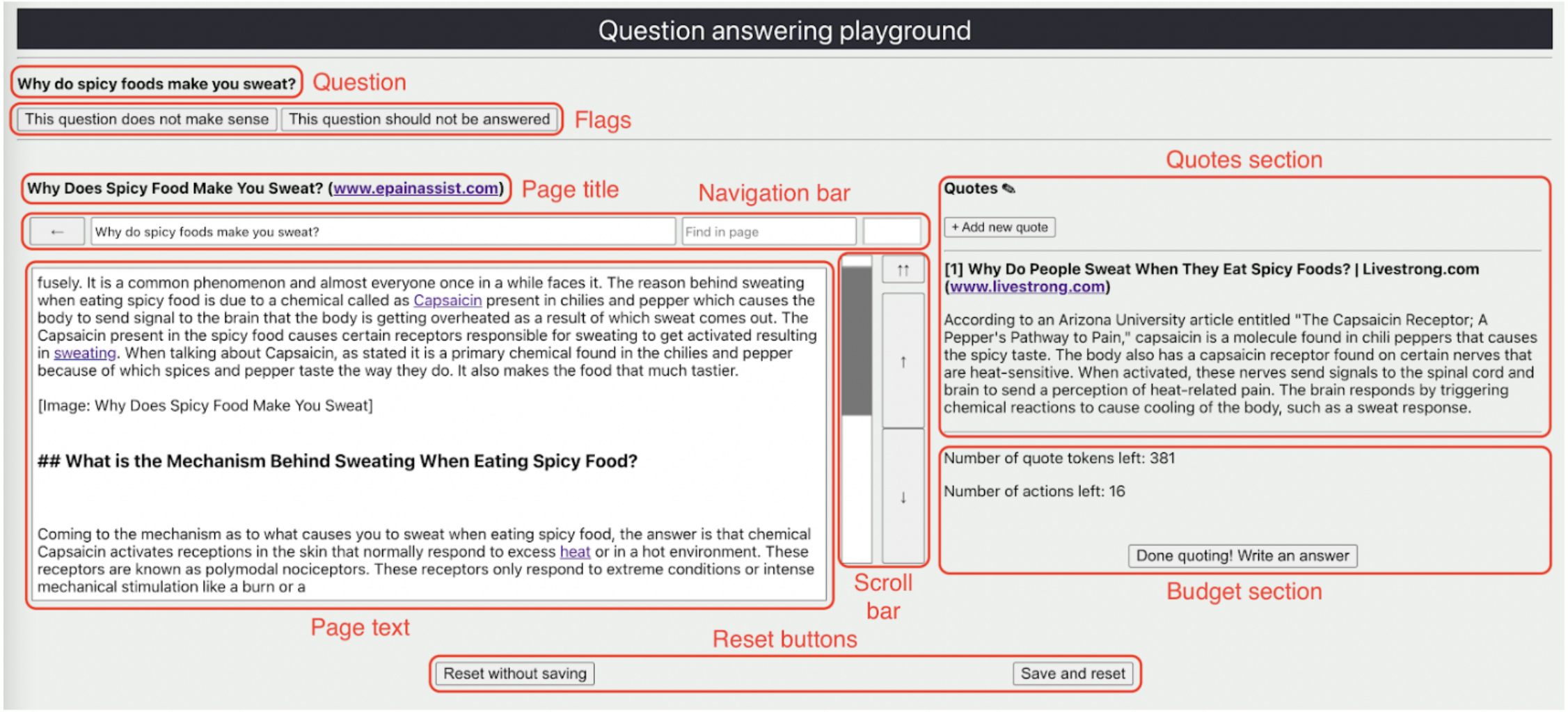
- joint training of retrieval/encoder helps
- index update required

Teach LM to use search and browse web: WebGPT(finetuning)

- Long format question-answering problem(LFQA)
- Leverage on-the-shelf knowledge retrieval & answer synthesis system
  - Bing Search API
  - GPT-3 175B
- Focus on finetuning to align with human feedback
  - **Behavior Cloning(BC)**: Supervised finetune with human *demonstrations*
  - **Reward Modeling(RM)**: Train with *comparison* feedback based on BC model
  - **Reinforcement Learning**: Finetune BC model with RM reward and BC KL penalty
  - Generation with **rejection sampling**: BC/RL on RM(Best-of-n)

WebGPT: Browser-assisted question-answering with human feedback, 2021, OpenAI

Teach LM to use search and browse web: Human Feedback Tool



WebGPT: instructions for contractors

Teach LM to use search and browse web: Action Space & Prompt

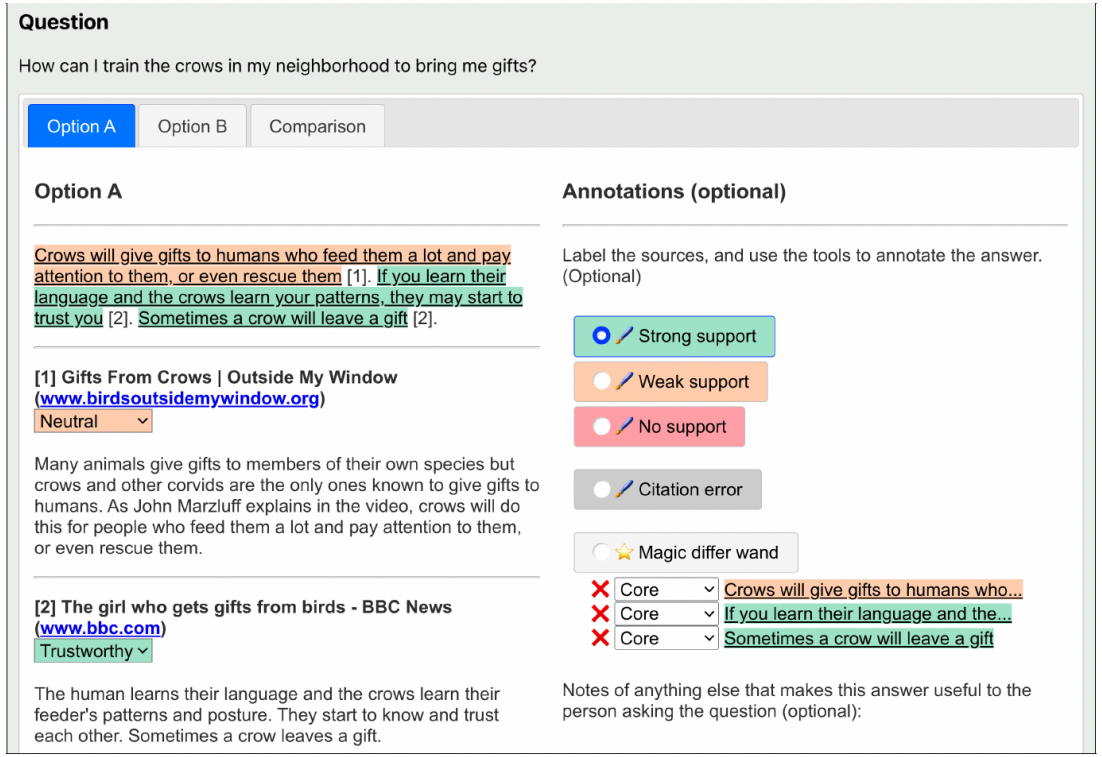
- **Action Space**
  - Search {query}
  - Clicked on link {link ID}
  - Find in page: {text}
  - Quote: {text}
  - Scrolled down/up {1,2,3}
  - Top/Back
  - End: Answer
  - End: {Nonsense, Controversial}

- **Prompt**

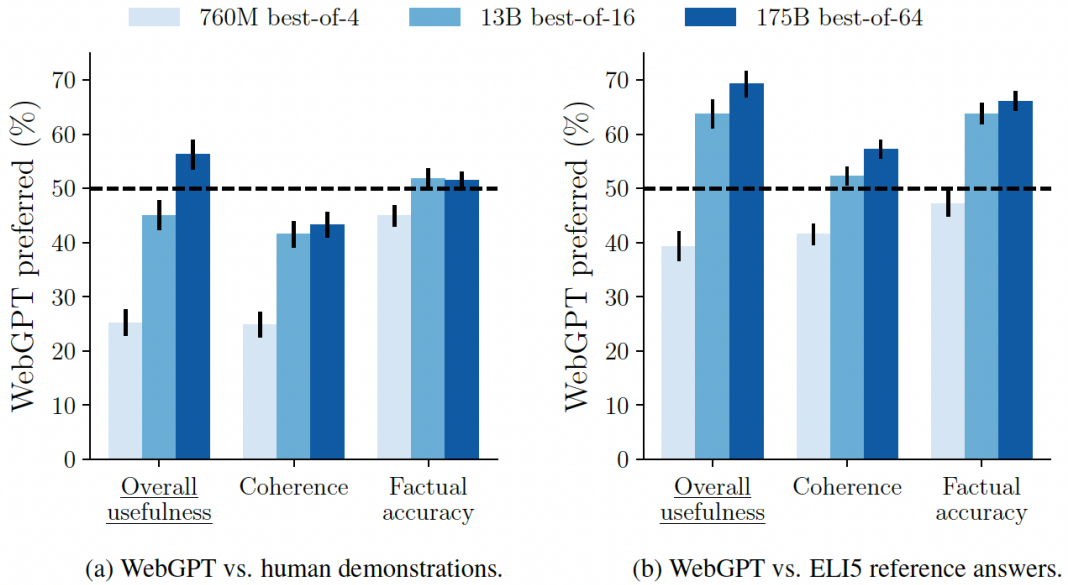
```
♦Question
How can I train the crows in my neighborhood to bring me gifts?

♦Quotes
From Gifts From Crows | Outside My Window (www.birdsoutsidemywindow.org)
> Many animals give gifts to members of their own species but crows and
other corvids are the only ones known to give gifts to humans.

♦Past actions
Search how to train crows to bring you gifts
Click Gifts From Crows | Outside My Window www.birdsoutsidemywindow.org
Quote
Back

♦Title
Search results for: how to train crows to bring you gifts

♦Scrollbar: 0 - 11
♦Text
【0†How to Make Friends With Crows - PetHelpful†pethelpful.com】
If you did this a few times, your crows would learn your new place, but
as I said, I'm not sure if they will follow or visit you there since it s
probably not in their territory. The other option is simply to make new
crow friends with the crows that live in your new neighborhood.

【1†Gifts From Crows | Outside My Window†www.birdsoutsidemywindow.org】
The partial piece of apple may have been left behind when the crow was
startled rather than as a gift. If the crows bring bright objects you'll
know for sure that it's a gift because it's not something they eat.
Brandi Williams says: May 28, 2020 at 7:19 am.

♦Actions left: 96
♦Next action
```

(b) Corresponding text given to the model.

Teach LM to use search and browse web: Main Conclusion

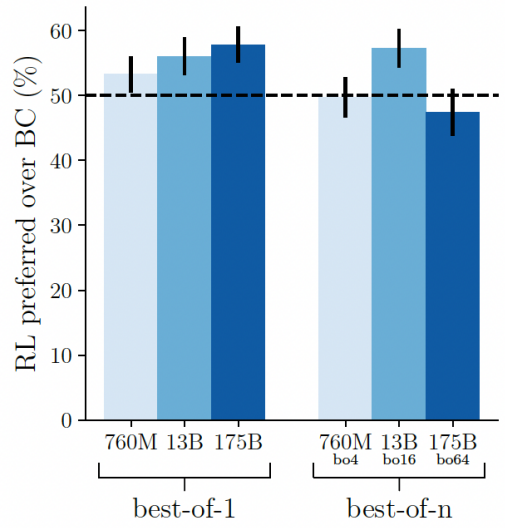


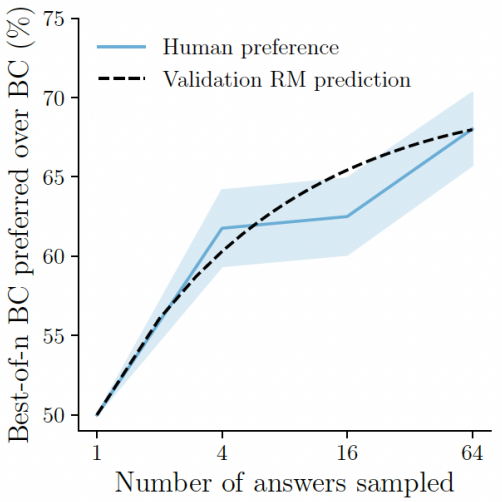
- Outperform human & reference answer
- RL shadowed by BC-best-of-n



(a) WebGPT vs. human demonstrations.

(b) WebGPT vs. ELI5 reference answers.





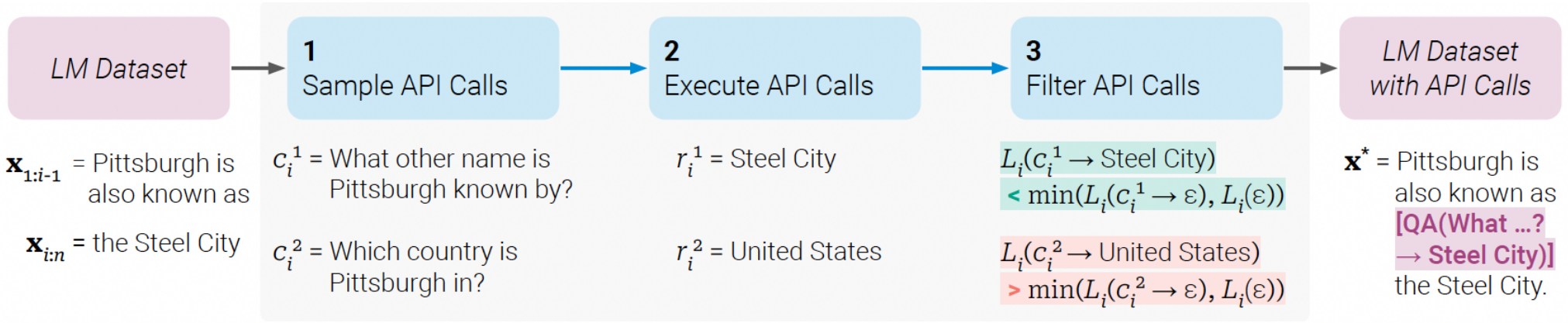
LLM self-teach to use tools: Toolformer

Corpus annotation

- Prompt LM(for each tool) and sampling m tools for **top-K** position
- Call APIs get $m \times K$ response
- Filter APIs for LM loss reduction $\geq T_f$
- Augment corpus with remaining APIs

LLM finetuing & prediction

- Finetuning using standard LM objective
- Decoding as usual, stop when meeting $\rightarrow$, call corresponding API
- Fill text with API reponse, continue decoding



Toolformer: Language Models Can Teach Themselves to Use Tools, 2023, MetaAI

Prompt & tools involved in Toolformer

Tools involved

- Question Answering:
  - Atlas(Retrieval augmented QA LM)
- Calculator
- Wikipedia Search:
  - BM25 retrieval for wikidump
- Machine Translation System
  - any lang to English trans(NLLB)
- Calendar

Prompt for QA Tool

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in **[QA("Where was Joe Biden born?")]** Scranton, **[QA("In which state is Scranton?")]** Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or **[QA("What other name is Coca-Cola known by?")]** Coke, is a carbonated soft drink manufactured by **[QA("Who manufactures Coca-Cola?")]** the Coca-Cola Company.

Input: x

Output:

Results and limitations

Experiment settings & results

- GPT-J(6B) finetuned with CCNet(HQ subset of C4)
- Toolformer outperforms on QA/MLQA/math reasoning/Temporal Datasets, e.g. 3 LAMA QA subsets

Model	SQuAD	Google-RE	T-REx
GPT-J	17.8	4.9	31.9
GPT-J + CC	19.2	5.6	33.2
Toolformer (disabled)	22.1	6.3	34.9
Toolformer	**33.8**	**11.5**	**53.5**
OPT (66B)	21.6	2.9	30.1
GPT-3 (175B)	26.8	7.0	39.8

Table 3: Results on subsets of LAMA. Toolformer uses the question answering tool for most examples, clearly outperforming all baselines of the same size and achieving results competitive with GPT-3 (175B).

Limitations

- No capability of tools in a chain
- No tool interaction like WebGPT
- Sensitive to exact wording of input

End of Augmented Language Models

Plugins

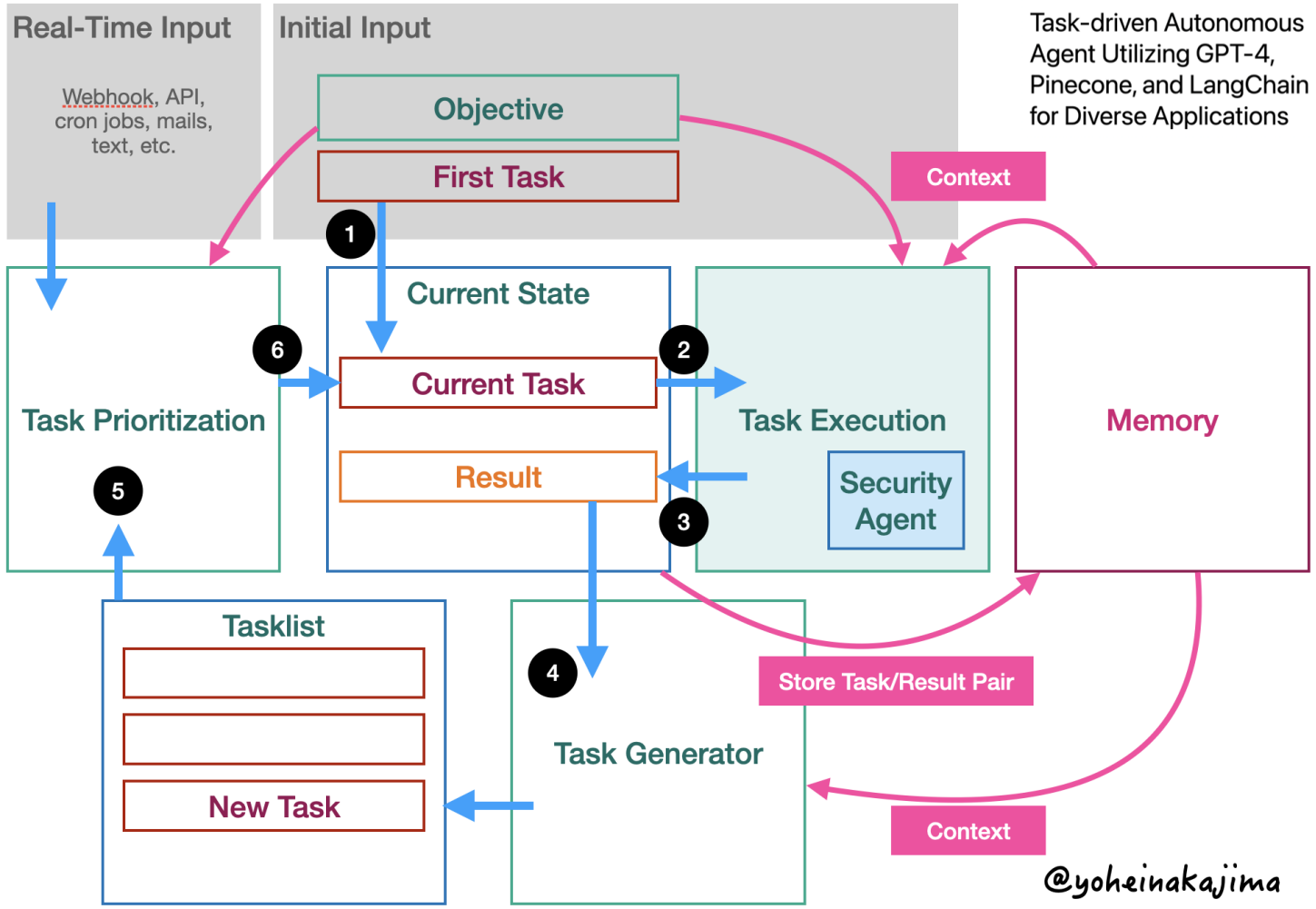
- Plugins ecosystem
- Primary implementations
  - Langchain Agent/Tool
  - ChatGPT Plugins
  - Fixie
  - Other examples in application

Plugins ecosystem

- Tool as a service
- From SEO to LMO
- Orchestration by LLM

Paradigm of Task-driven Autonomous Agent

- AutoGPT/BabyAGI
- Langchain Agent/Tool
- Fixie



Task-driven Autonomous Agent Utilizing GPT-4, Pinecone, and LangChain for Diverse Applications, 2023

Langchain Agent/Tool

Agent(backed by LLM) drives everything

- **prompt** with examples of planning and step-by-step actions
- **weak(description in prompt)** tool operating manual
- interact between LLM and tool with `stop` argument
- reasoning trace stored in `scratchpad`: *short memory*
- `memory` supported for chat history buffer: *long memory*

Implemented Agents

- **mrkl**: zeroshot, open tool-set
- **Self-ask**-with-search: Google **search**
- **ReAct**: Wiki **[search|find]**
- **chat**: **ReAct** open tool-set, action in JSON
- **conversational**: **ReAct** open tool-set, interact with human
- **conversational_chat**: **ReAct** open tool-set, user take action marked in JSON

More on Langchain

ChatGPT Plugins

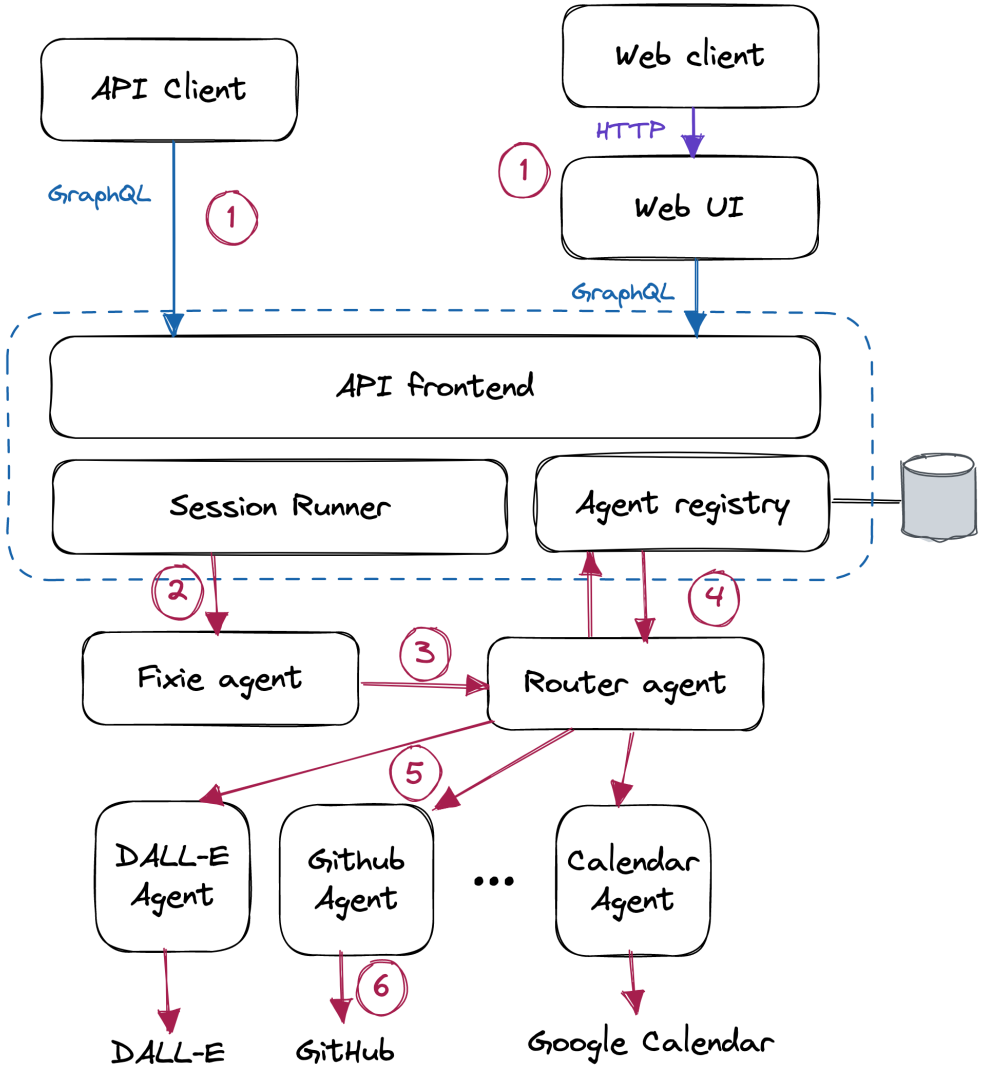
ChatGPT drives everything

- Plugin description(in manifest) matters
- OpenAPI markdown for tool specification
- Orchestrate by ChatGPT during conversation
    - When to call
    - Which to call
    - How to call
    - Synthesis results
- Released `functions` argument in Chat Completions API

Chat Plugins
functions argument use case

Fixie's(startup raised 17M$) proposal

- Session Runner:
  - dispatch query
  - maintain the running chat log
- Fixie Agent(LLM):
  - break query into sub-queries
  - dispatch to Router
- Router Agent(Neural Search):
  - match query/agent by neural search
  - agent represented by query samples

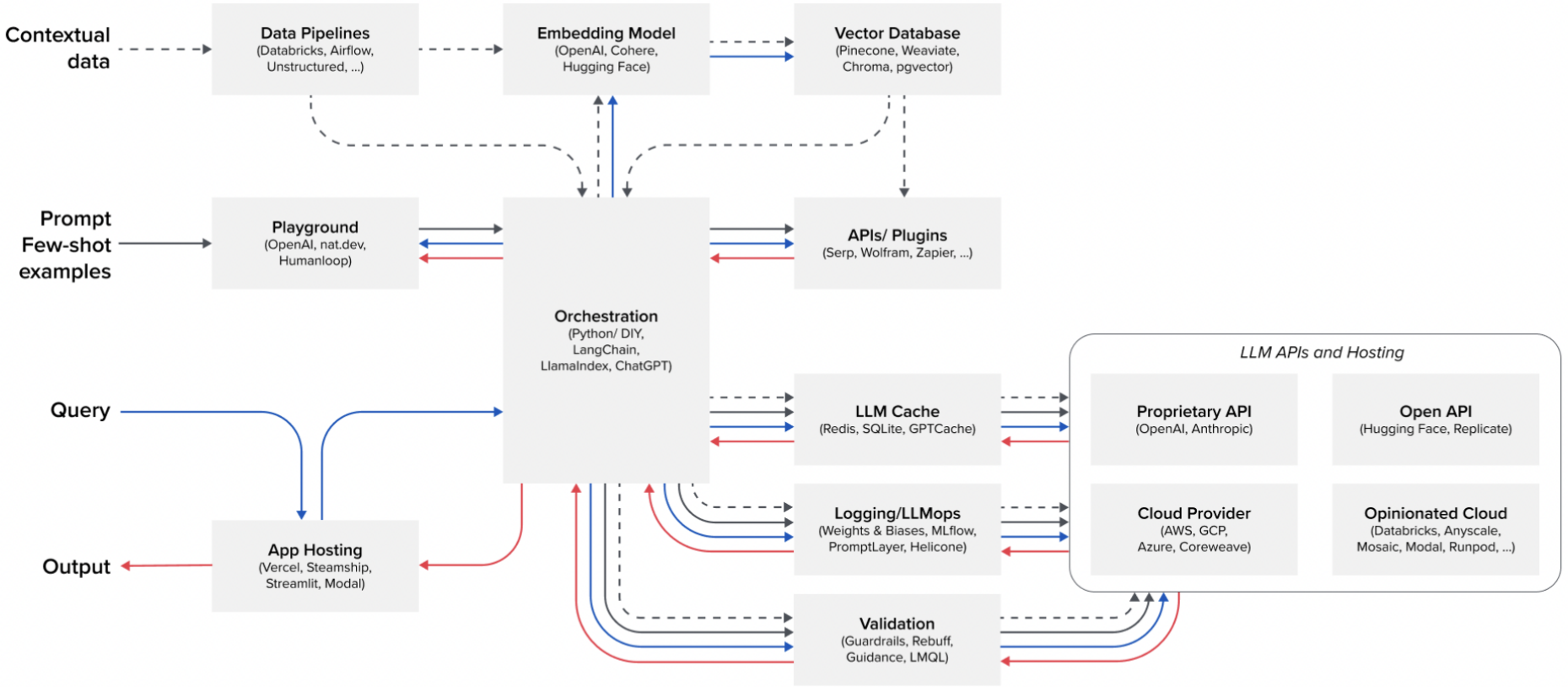


Fixie architecture

Other agent examples in application

- MM-ReAct: Microsoft
- TaskMatrix.AI: Microsoft
- Tree-of-Thoughts: Princeton University/Deepmind
- LLMs As Tool Makers: Princeton University/Deepmind

LLM TechStack for in-context learning Application



LEGEND

Gray boxes show key components of the stack, with leading tools/systems listed

Arrows show the flow of data through the stack

- Contextual data provided by app developers to condition LLM outputs
- Prompts and few-shot examples that are sent to the LLM
- Queries submitted by users
- Output returned to users

a16z Enterprise

End of Plugins

Important but not covered

- Practical prompt enginneering
- Text to image:
  - Latent Diffusion Model/CLIP
  - DALL-E
  - Stable Diffusion
- Multi-modal and embodient
  - PaLM-E
  - Gato
- Quantization & efficient inference on edge device
  - ggml by Georgi Gerganov
  - bitsandbytes: LLM.int8()/QLoRA
- And more...

Thanks & QA?