Unleash the Potential of LLMs through Task and Data Engineering

Shafiq Joty

https://raihanjoty.github.io/

Research Director

salesforce

Assoc. Prof. (On Leave)

NTU NLP

Outline

A. Background

- Role of Model, Data and Tasks in LLMs

B. XGen LLM

- Pre-training & instructional tuning

C. Task engineering with LLMs

- Knowledge-enhanced chain-of-thought
- Low-resource translation
- Data distillation

D. Limitations

Background: Feature Engineering

NLP before 2014

- Extract linguistic and other features useful for tasks
- Requires language, domain or task expertise

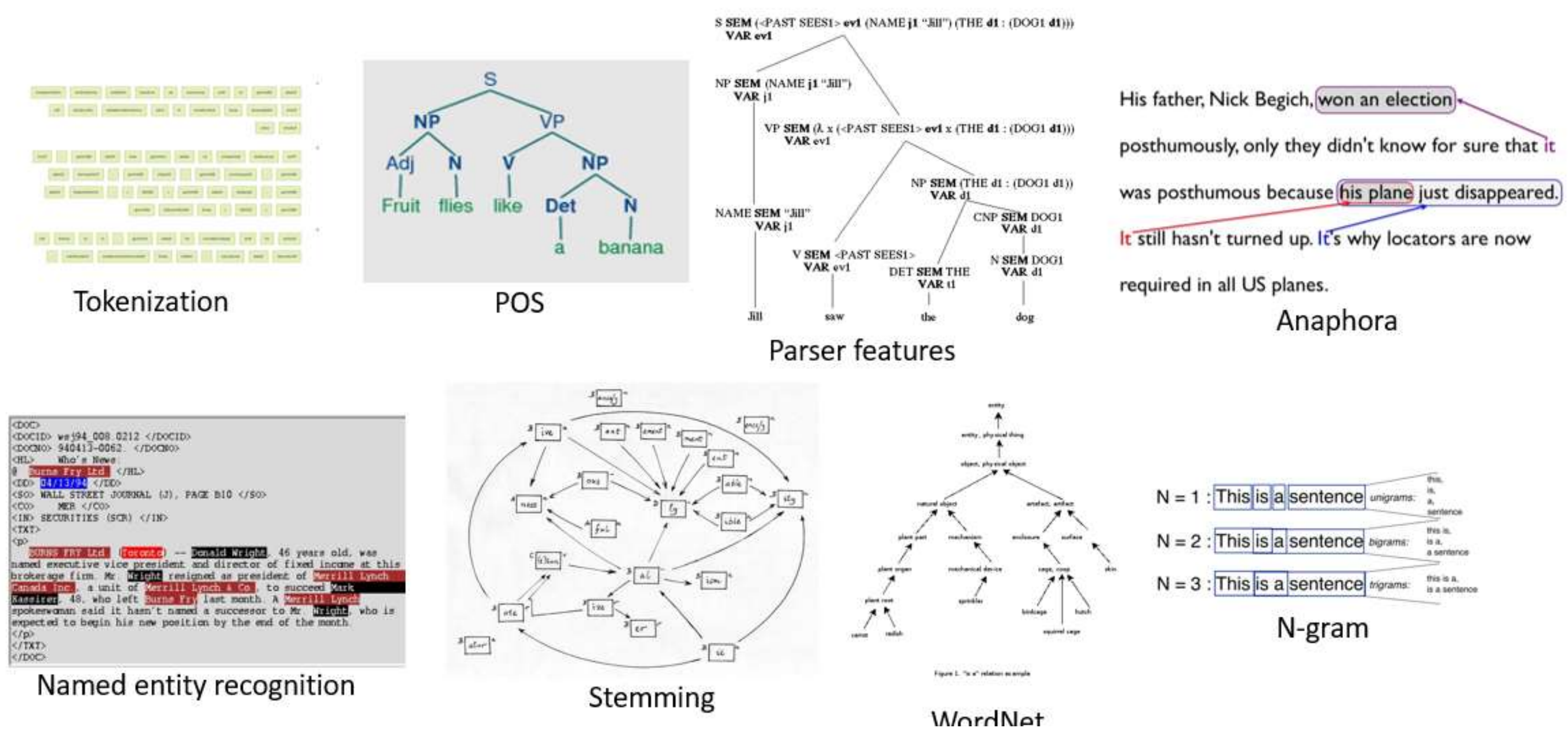
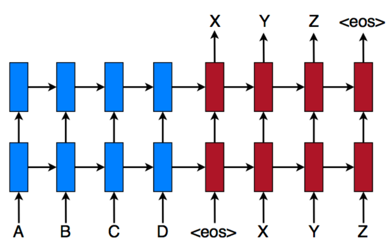


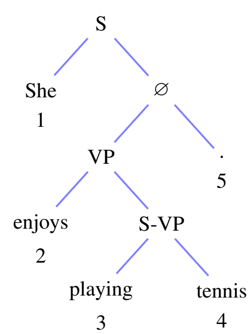
Image credit: Internet

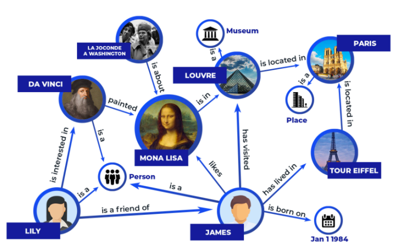
Background: Model Engineering

Move from feature engineering to model engineering (2014 —)

- Design network architectures with better inductive biases







✳ Neural Nets

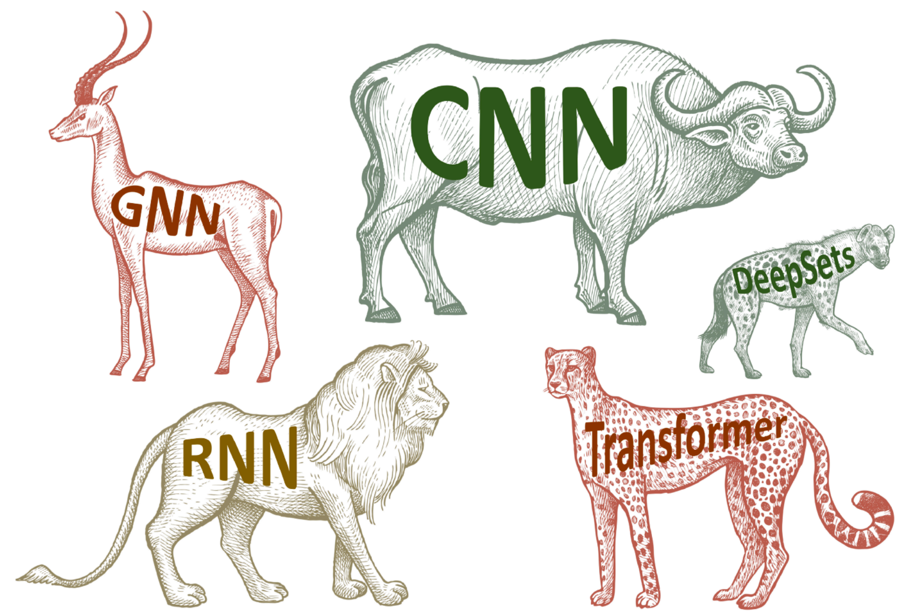
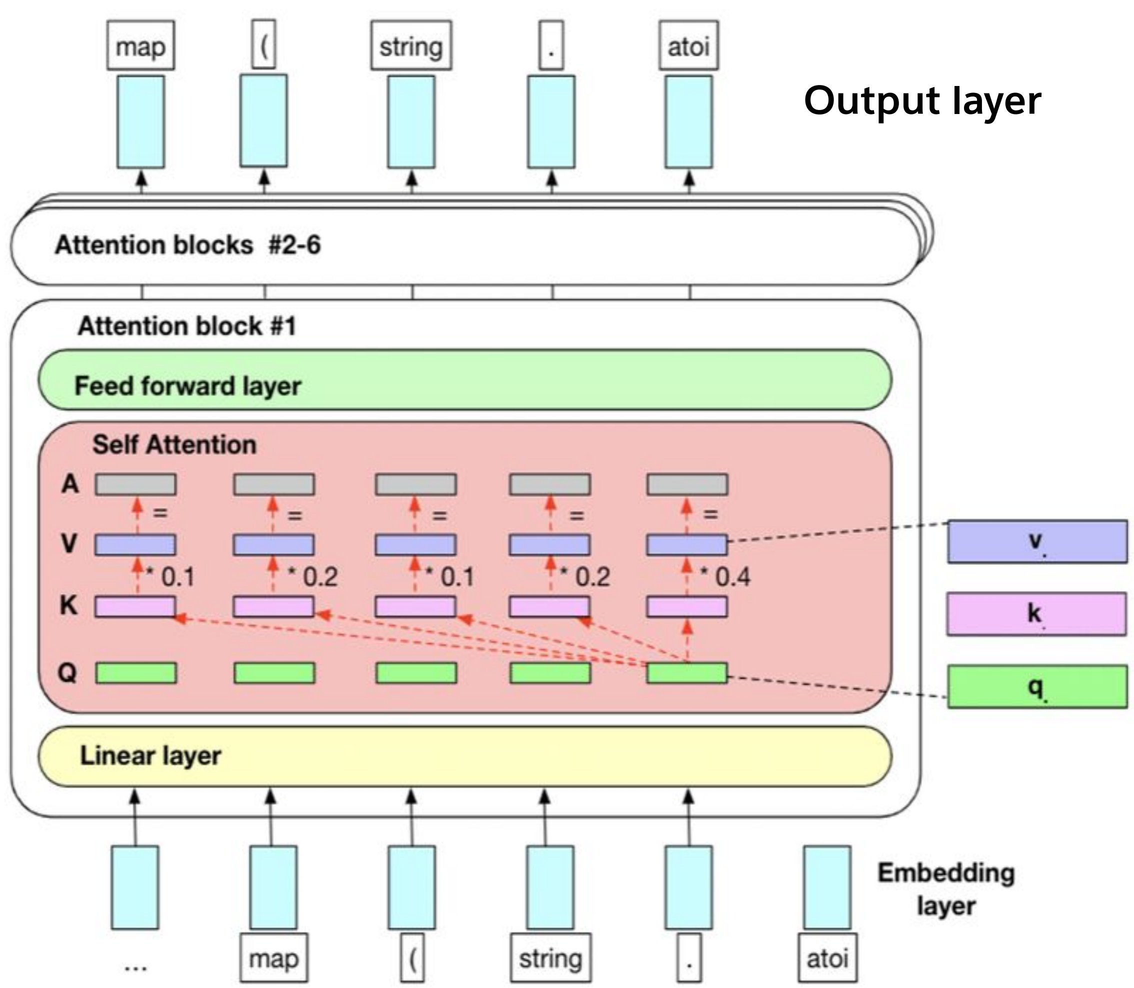


Image Source: Michael Bronstein blog

Background: Model Engineering

Transformer decoder has become a standard for LLMs

- Causal self attention for representation learning
- Causal LM as a pre-training objective: $P(x_{t+1} \mid x_1, \ldots, x_t)$



Greedy or Stochastic sampling

Repeat attention block N times

Shared FFN

Self-attention with causal masking

Linear head projection to query, key and value representations

Fetch token embeddings

Image source: Learning Autocompletion from Real-World Datasets

Background: Data Engineering

Renewed interests esp. with LLMs

To spur innovation on data-centric AI approaches, perhaps it's time to hold the Code fixed and invite researchers to improve the Data.
A huge amount of innovation — in algorithms, ideas, principles, and tools — is needed to make data-centric AI development efficient and effective.

Andrew Ng. May 26, 2021

Model-Centric AI

How can you change the model (code) to improve performance?

➡

Data-Centric AI

How can you systematically change the data (inputs x or labels y) to improve performance?

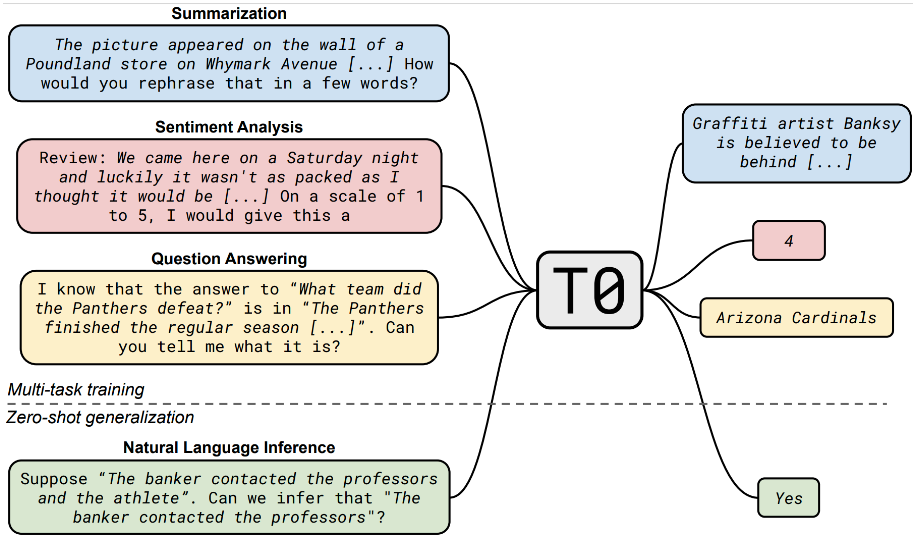
✴ Examples

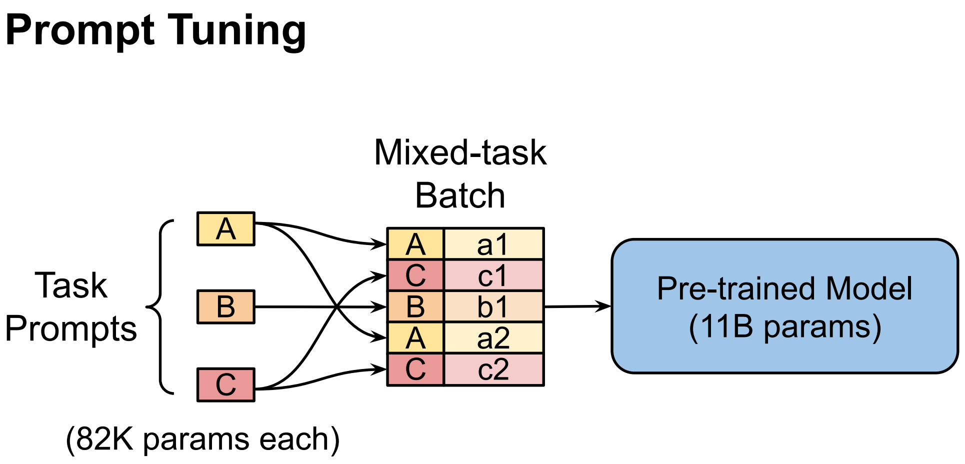
- Apply effective pre-processing (e.g., tokenisation)
- Determine the right mixture of data sources
- Deal with inconsistencies in data labels
- Identify bias & toxic content
- Use effective data augmentation techniques

Source: https://www.deeplearning.ai/the-batch/

Background: Rise of Task Engineering

- Multi-task models with task prompts
  - Same backbone model for all tasks
  - Add "informative" tokens (or task instructions)





[1] The Power of Scale for Parameter-Efficient Prompt Tuning

[2] https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

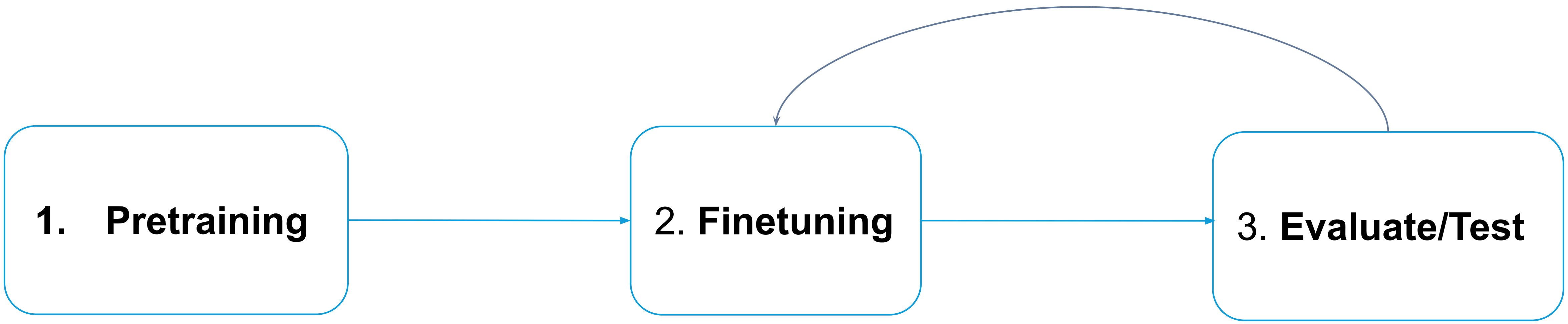
[3] https://prompts.ai/

Background: Task Engineering

- What tasks to consider?
- What is the right objective (or reward)?
  - Accuracy (traditionally)
  - What about protected attributes (e.g., age, colour, gender, race, religion)?
  - What about privacy & security issues?

- Alignment research
  - Aligning LLMs to “human” instructions and values
  - Helpfulness and Harmlessness measures

LLM Lifecycle

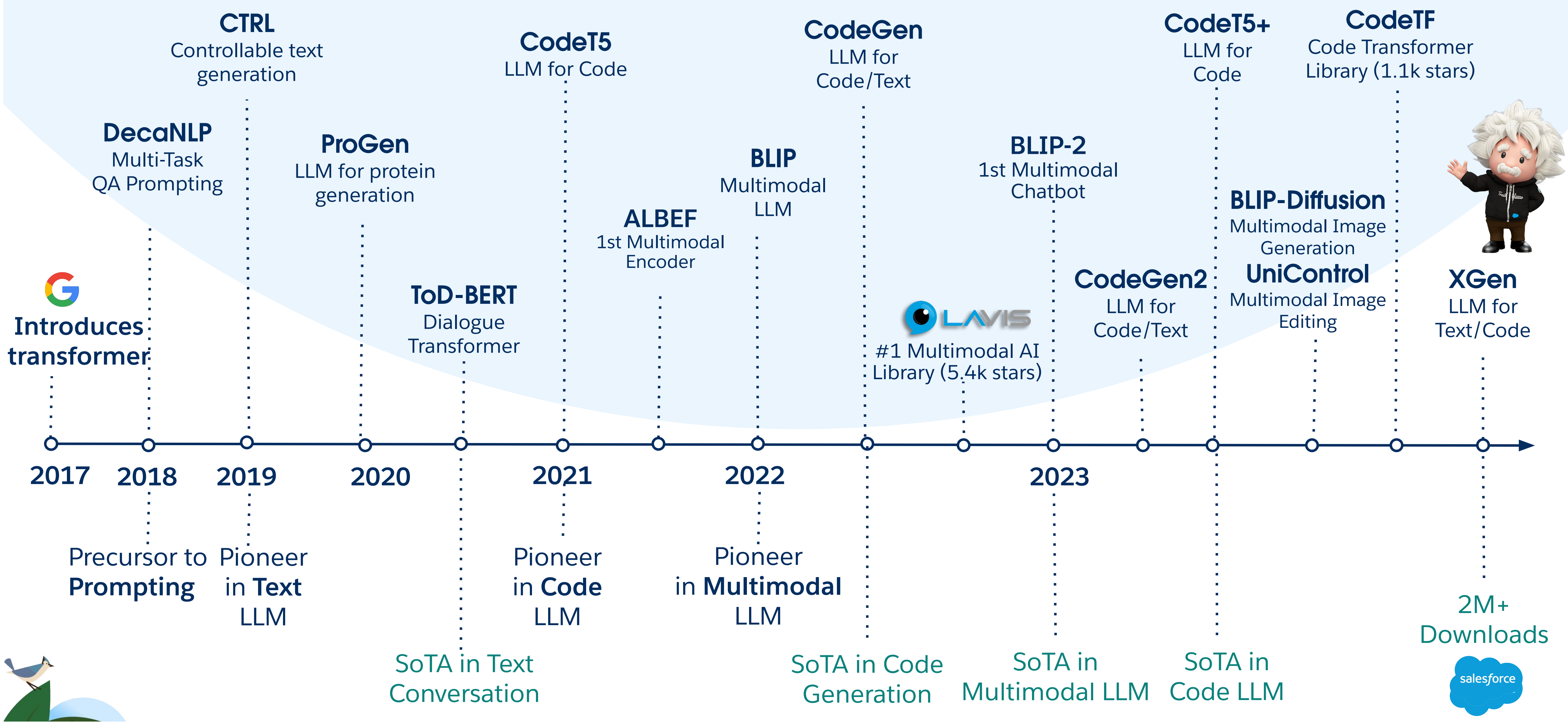


- Unsupervised pre-training on large data (typically 1T+ tokens)
- Data: Text + Code (typically)
- Large models show better in-context learning capabilities (e.g., GPT 3 is 175B)

- Align to task instructions & labels
- Also make honest and harmless
- Method: supervised finetuning
  - + RL w/ HF or AIF (optional)
- Challenge: getting diverse task instructions & input-output instances

- Instance held-out
  - Supervised learning setup
- Task held-out
  - Similar tasks can be seen
- Task type held-out
  - Completely unseen tasks

Salesforce LLMs and AI Libraries

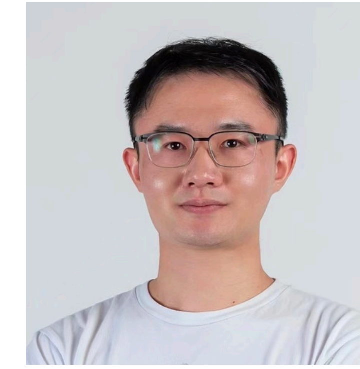


The XGen LLM

- 7B parameters, 8K sequence length, 1.5T tokens
    - Fine-tune on public-domain instructional data
- Achieves comparable or better results on standard benchmark compared with SoTA open-source LLMs (e.g. MPT, LLaMA-1, OpenLLaMA) of similar model size.
- Shows benefits on long sequence modeling benchmarks
- Achieves equally strong results both in text and code tasks
- Training cost of $150K for 1T tokens under Google Cloud TPU-v4

Codebase: https://github.com/salesforce/xGen
Model Checkpoint: https://huggingface.co/Salesforce/xgen-7b-8k-base

The XGen LLM — Pre-training Data



- Stage 1:

Dataset	Tokens (B)	Epochs	Sampling prop. (%)
RedPajama-CommonCrawl	879.37	1	63.98
RedPajama-GitHub	62.44	1	4.54
RedPajama-Books	65.18	2.5	4.74
RedPajama-ArXiv	63.32	2	4.61
RedPajama-StackExchange	21.38	1	1.56
C4 from 6 CC dumps (2019 - 2023)	191.50	0.2	13.93
Wikipedia-English	19.52	4	1.42
Wikipedia-21 other languages	62.04	2	4.51
Pile-DM Mathematics	7.68	2	0.56
Apex code from 6 CC dumps	2.09	1	0.15
Total	1374.52		100

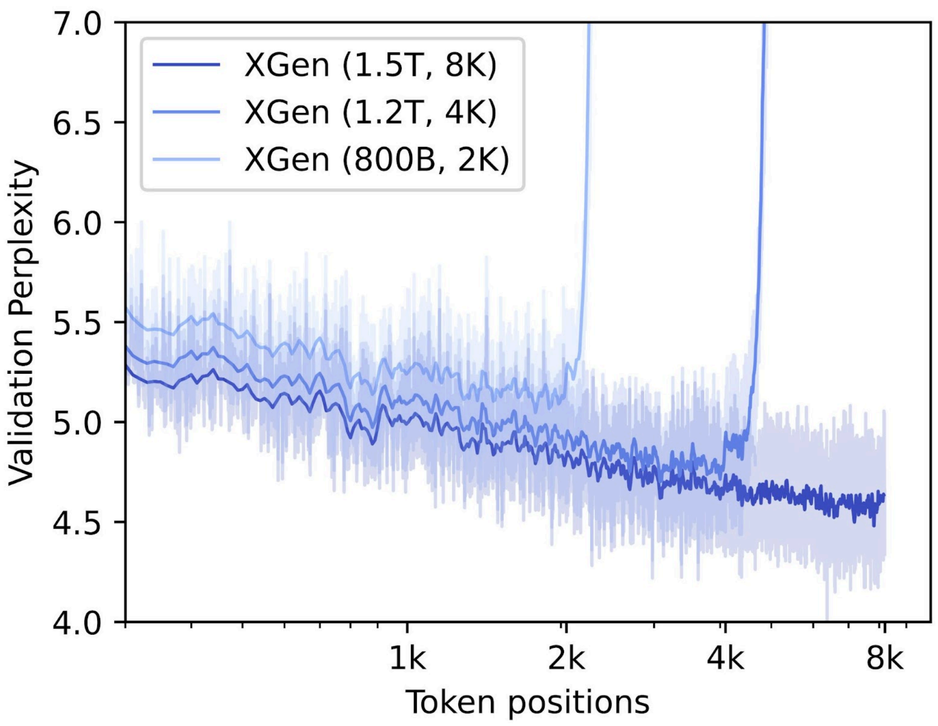
- Stage 2:

Dataset	Tokens (B)	Sampling prop. (%)
Data from stage 1	55	50
BigCode Starcoderdata	55	50
Total	110	100

- Tokenizer:
  - OpenAI's BPE Tiktoken + code related special tokens

The XGen LLM — Pre-training

- In-house JaxFormer library
  - Both data and model parallelism optimized for TPU-v4 hardware
- Training recipe: mostly follow LLaMA-7B except:
  - Token budget increased to 1.5T tokens
  - Stage-wise training to increase the sequence length from 2K to 4K to 8K
  - Vocabulary size increased to 51,200 tokens



The XGen LLM — Base model evaluation

- MMLU

Table 3: Massive Multitask Language Understanding (MMLU). Five-shot accuracy.

Models	**Humanities**	**STEM**	**Social Sciences**	**Other**	**Weighted average**
XGen-7b	33.8	30.7	40.0	41.5	36.3
LLaMA-7b	33.9	30.6	38.2	38.2	35.1
OpenLLaMA-7b	28.1	28.5	31.2	32.8	29.9
Falcon-7b	26.5	25.4	29.2	26.8	26.9
MPT-7b	25.9	26.2	26.9	28.1	26.7
Redpajama-7b	26.1	25.2	27.4	26.7	26.3
Cerebras-GPT-13b	26.1	26.5	25.8	26.6	26.2
Dolly-v2-12b	26.9	25.7	25.3	26.5	26.2
OPT-13b	26.2	24.3	23.4	26.0	25.1
GPT-J-6b	25.9	24.0	24.0	25.8	25.1

The XGen LLM — Base model evaluation

- ## QA and common sense reasoning

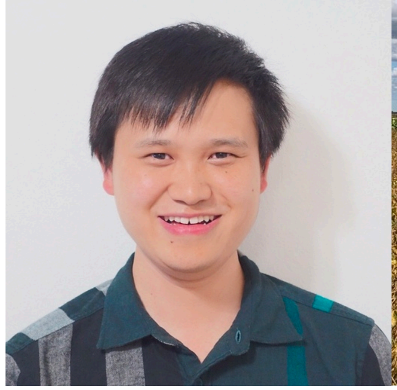
Table 5: Zero-shot performance on Common Sense Reasoning and Question Answering tasks.

Models	MMLU -wavg	ARC_ch	HellaSwag	Winogrande	TruthfulQA	BoolQ	PiQA	OpenBookQA
XGen-7b	32.1	41.2	74.2	64.9	39.1	74.3	75.5	40.2
LLaMA-7b	32.0	44.8	76.2	69.6	34	74.9	78.7	44.2
Falcon-7b	23.9	43.4	76.4	67.2	34.3	73.8	79.4	44.0
MPT-7b	27.4	41.7	76.1	68.6	33.4	74.1	79.1	41.8
OpenLLaMA-7b	28.6	38.7	71.8	67.0	35.2	70.6	76.0	39.0
Redpajama-7b	25.8	39.1	70.3	63.8	33.3	69.3	76.9	40.0
GPT-neox-20b	24.5	41.1	70.5	66.1	31.4	64.9	76.7	38.8
OPT-13b	24.4	35.8	69.9	64.7	33.9	65.0	75.7	39.8
GPT-J-6b	25.7	36.3	66.2	64.5	36.0	65.4	75.4	38.2
Dolly-v2-12b	25.4	39.6	70.8	61.8	34.4	56.3	75.4	39.2
Cerebras-GPT-13b	24.6	32.4	59.4	60.8	39.2	61.1	73.5	35.8
StableLM-alpha-7b	24.4	27.0	40.7	51.5	41.7	59.0	65.8	32.4

- ## Code (HumanEval)

Models	pass@1
XGen-7b	14.20
MPT-7b	15.90
OpenLLaMA-7b-v2	14.83 (30% of the pretraining data is Starcoder data)
LLaMA-2-7b	13.55
LLaMA-7b	10.38
Redpajama-7b	5.24
OpenLLaMA-7b	0 (consecutive whitespaces are treated as one, breaking Python syntax)
Falcon-7b	0 (didn't generate meaningful code)

The XGen LLM — Instruction tuned




- Instructional data: WizardLM [1]
- Supervised fine-tuning

```
### Human: {prompt} ### Assistant: {response}
```

- Alpaca eval

Model	**Win Rate vs text-davinci-003**
GPT-4	95.3
Claude	88.4
Chatgpt	86.1
Wizardlm-13b	75.3
Guanaco-65b	71.8
Vicuna-13b	70.4
XGen-7b-inst	68.8
Wizardlm-7b	65.2
OAsst-rlhf-llama-33b	66.5
Vicuna-7b	64.4
text-davinci-003	50.0
Falcon-40b-instruct	45.7
MPT-7b-chat	45.0
Alpaca-farm-ppo-human	41.2
Alpaca-7b	26.5
text_davinci-001	15.2

- MT bench

Model	**Score**
GPT-4	8.99
GPT-3.5-turbo	7.94
Claude-v1	7.90
Claude-instant-v1	7.85
Vicuna-33b-v1.3	7.12
Wizardlm-30b	7.01
Guanaco-33b	6.53
Tulu-30b	6.43
Guanaco-65b	6.41
OAsst-sft-7-llama-30b	6.41
Palm-2-chat-bison-001	6.40
MPT-30b-chat	6.39
Vicuna-13b-v1.3	6.39
Wizardlm-13b	6.35
Vicuna-7b-v1.3	6.00
Baize-v2-13b	5.75
XGen-7b-inst	5.69
Nous-hermes-13b	5.51
MPT-7b-chat	5.42
GPT4all-13b-snoozy	5.41
Koala-13b	5.35
Wizardlm-7b	5.29
MPT-30b-instruct	5.22
Falcon-40b-instruct	5.17
H2ogpt-oasst-open-llama-13b	4.63
Alpaca-13b	4.53
Chatglm-6b	4.50
OAsst-sft-4-pythia-12b	4.32
Rwkv-4-raven-14b	3.98
Dolly-v2-12b	3.28
Fastchat-t5-3b	3.04
Stablelm-tuned-alpha-7b	2.75
Llama-13b	2.61

[1] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023

Outline

A. Background

- Role of Model, Data and Tasks in LLMs

B. XGen LLM

- Pre-training & instructional tuning

C. Task engineering with LLMs

- Knowledge-enhanced chain-of-thought
- Low-resource translation
- Data distillation

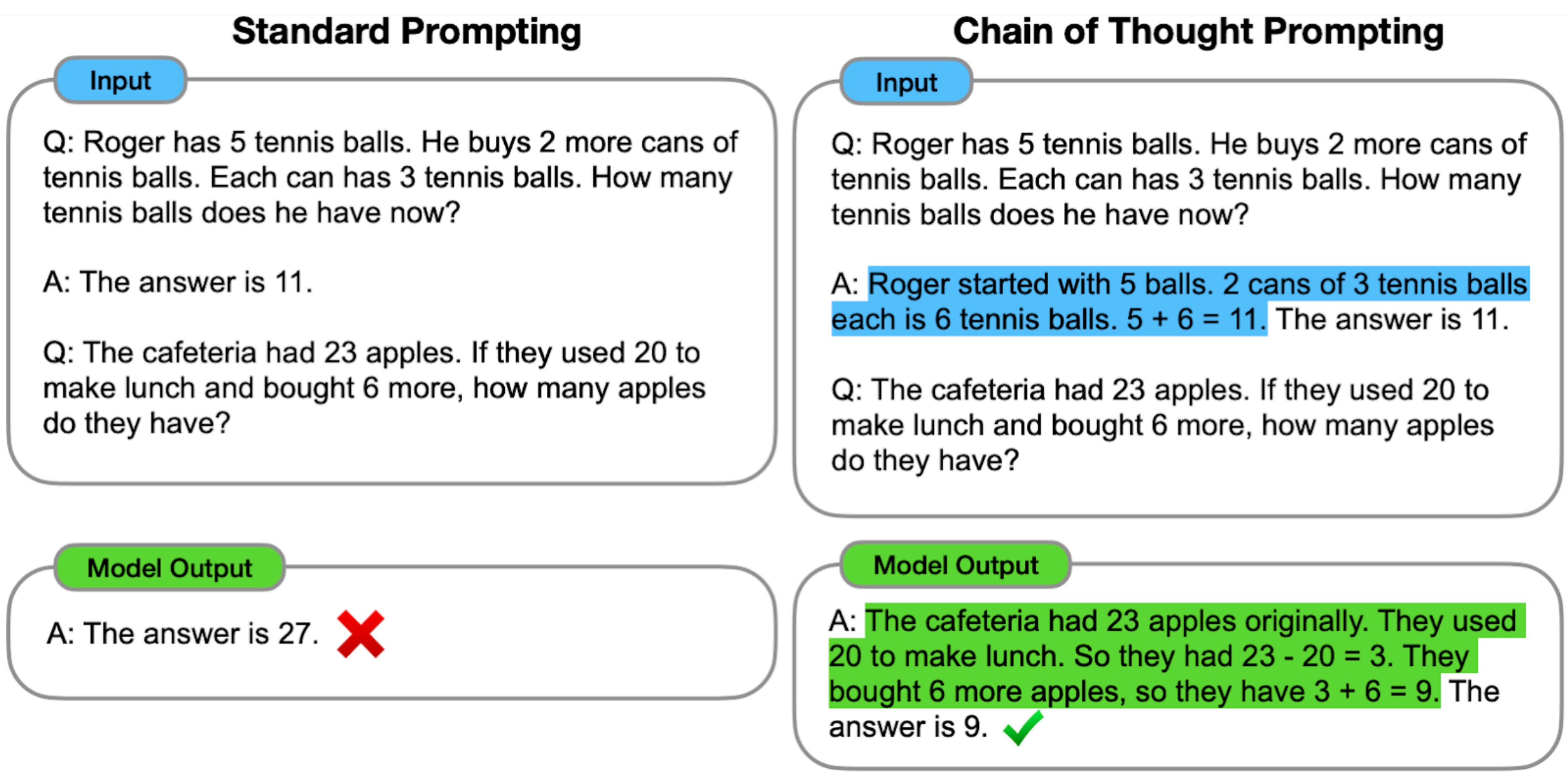
D. Limitations

One Key Challenge in LLMs

- Factual Correctness!
  - Innate shortcoming of generative models?
  - May contain outdated knowledge
  - Incorrect recalling of pre-trained knowledge
  - Make up facts
  - ...
- Contain ethical concerns and safety hazards.

Chain-of-Thought (CoT) in LLMs

- CoT improves LLM's abilities in reasoning tasks



Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Chain-of-Thought (CoT) in LLMs

- CoT improves LLM's abilities in reasoning tasks.
  - However, reasoning chains are only used to derive answers.
  - Current evaluation is result-oriented: if answer is wrong, regard the reasoning chain as "bad"
- Can we revise a "bad" reasoning path to be better?
  - Better reasoning chains should generate more correct answers.

How do we approach complex questions?

Step 1: Are we certain about the answer?

- If yes, answer with internal knowledge.
- If no, go to step 2.

Step 2: Look up relevant information in external resources!

- Answer with retrieved knowledge

Verify and Edit CoT [1]

Step 1: How can we tell when the model is uncertain?

If we directly ask: LLM will always say it's confident!

Self-consistency [2] is a good approximation

- Sample multiple reasoning paths for answering Q.
- If all paths lead to the same answer, self-consistency is high.

[1] Zhou et al. Verify-and-Edit: A Knowledge-Enhanced Chain-of-Thought Framework In **ACL-2023**

[2] Wang et al. Self-consistency improves chain of thought reasoning in language models. In ICLR-2023

Verify and Edit CoT

Step 2: Look up relevant information

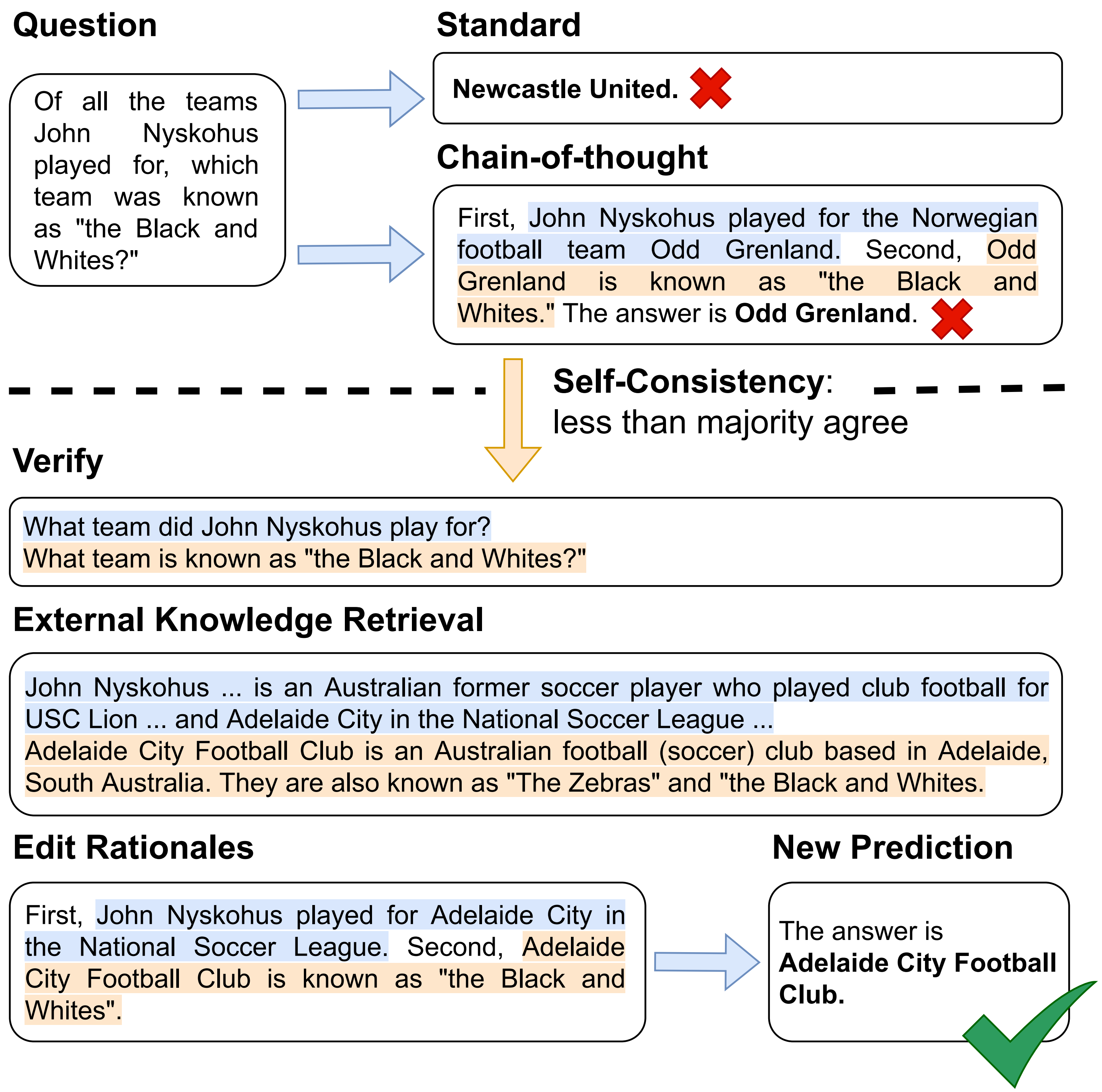
- Retrieval
  1. Verify a reasoning step by producing a question:

     "Sky is yellow" -> "What is the color of sky?"
  2. Retrieve with the query

     "The sky appears blue to the human eye"
- Synthesis
  3. Edit the reasoning step by incorporating retrieved information

[1] Zhou et al. Verify-and-Edit: A Knowledge-Enhanced Chain-of-Thought Framework In ACL-2023

Verify and Edit CoT



[1] Zhou et al. Verify-and-Edit: A Knowledge-Enhanced Chain-of-Thought Framework In ACL-2023

Results

HotpotQA:

Method	**knowledge**	**EM**	**ΔEM**	**AUC**
CoT-SC → ReAct	Wiki.	34.2%	+0.8%	-
ReAct → CoT-SC	Wiki.	35.1%	+1.7%	-
Standard	-	23.1%	-	43.24
CoT	-	31.8%	-	38.30
CoT-SC	-	31.2%	-	34.97
CoT-SC + Calib.	Dataset	-	-	49.00
CoT-SC + VE	Wiki.	35.7%	+4.5%	45.62
CoT-SC + VE	DRQA	36.0%	+4.8%	46.06
CoT-SC + VE	Google	37.7%	+6.5%	47.98
CoT-SC + VE	Dataset	**56.8%**	**+25.6%**	**60.94**

Human evaluation on factuality:

# Examples	**Cohen κ**	**CoT-SC**	**Ours**	**Tie**
50	0.25	17%	**53%**	30%

Supporting heterogeneous knowledge sources

Knowledge sources

- Unstructured (NL sentences)
- Structured (Wikidata, Tables)

How to query different sources effectively?

- Need a robust query generator

Knowledge Adapting framework [3]

Step 1: How can we tell which knowledge source to use?

Reasoning preparation

1. Break down the question into reasoning steps (CoT)
2. Select the most relevant knowledge source (domain)
    - e.g., the question requires medical knowledge

[3] Li et al. Chain of Knowledge: a framework for grounding large language models with structured knowledge bases.

Knowledge Adapting framework [3]

Step 2: How to retrieve the most relevant knowledge?

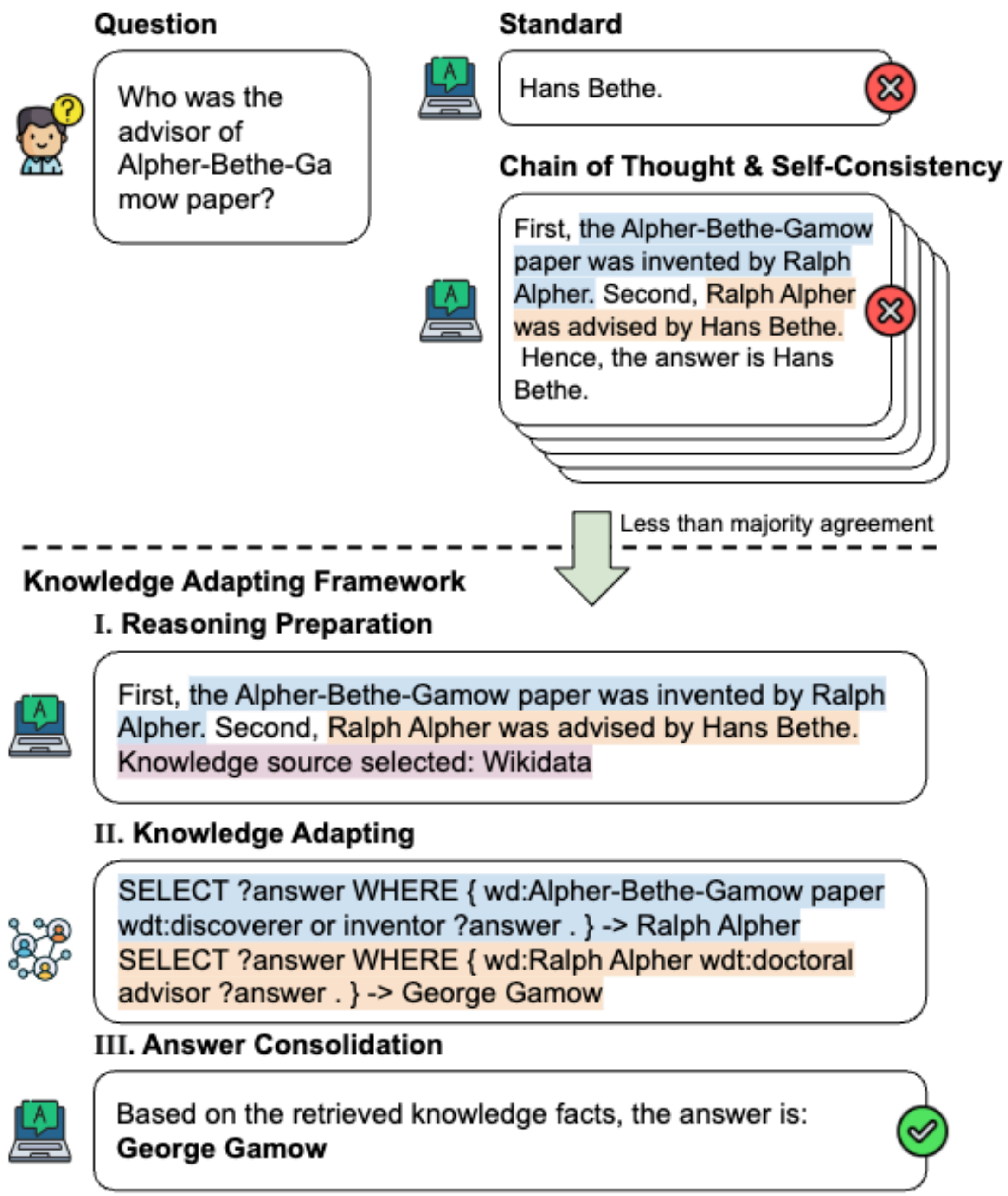
Knowledge Adapting

Methodology:

- Train an Adaptive Query Generator (AQG): instruction-tune LLaMA-7B with LoRA for each language
    - e.g., Natural sentence, SPARQL, SQL
- AQG generates a query for each reasoning step
- Execute it on the knowledge source

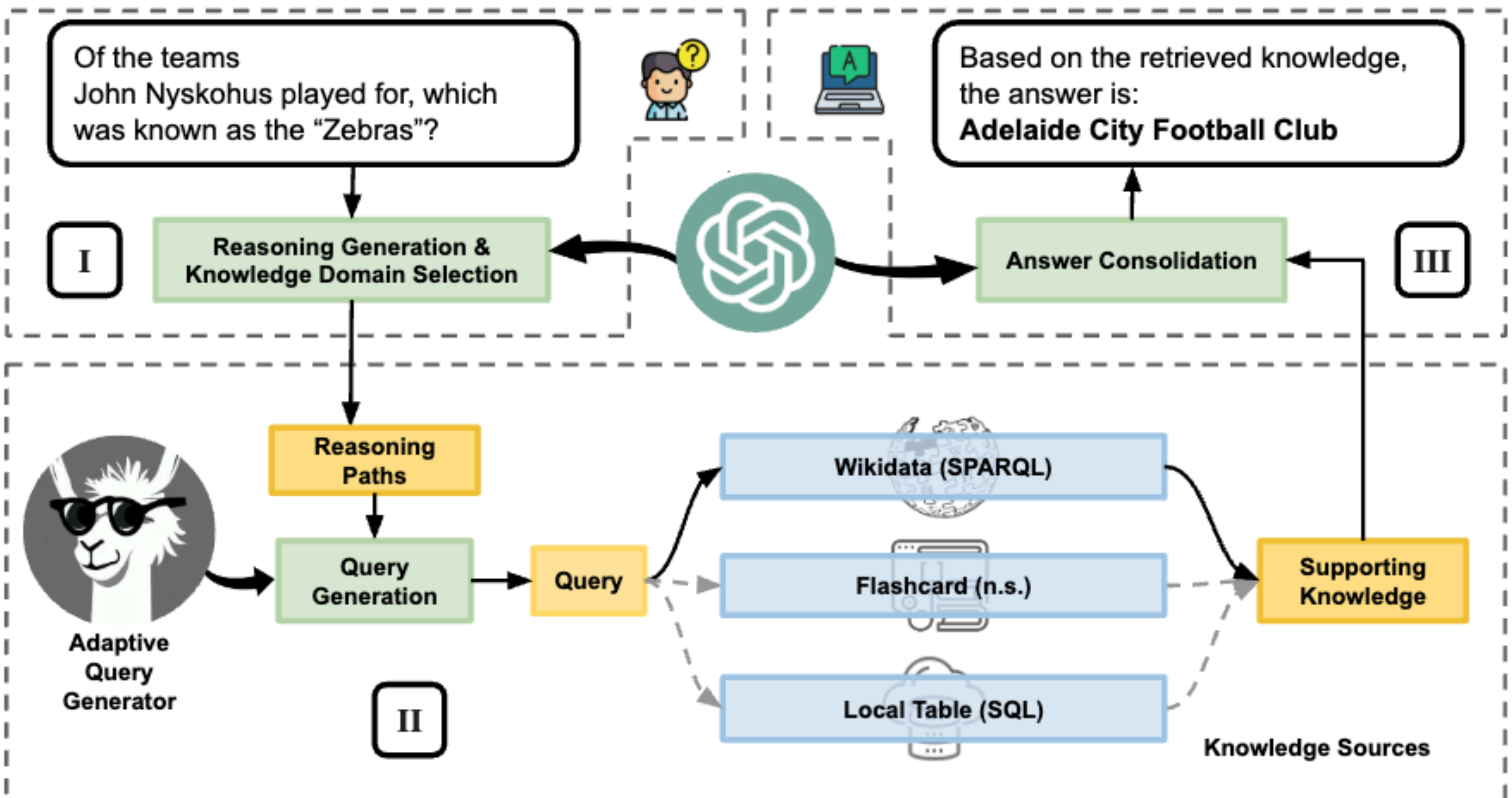
[3] Li et al. Chain of Knowledge: a framework for grounding large language models with structured knowledge bases.

Knowledge Adapting framework [1]



[3] Li et al. Chain of Knowledge: a framework for grounding large language models with structured knowledge bases.

Knowledge Adapting framework [1]



[3] Li et al. Chain of Knowledge: a framework for grounding large language models with structured knowledge bases.

Results

Results on factual & medical domains:

Method	FEVER		AdvHotpotQA		MedMCQA		FeTaQA	
	Knwl.	**Acc.**	**Knwl.**	**EM**	**Knwl.**	**Acc.**	**Knwl.**	**BLEU**
Standard (one-shot)	✗	54.3	✗	17.6	✗	61.3	✗	20.7
CoT (one-shot)	✗	56.6	✗	20.3	✗	65.5	-	-
CoT-SC (one-shot)	✗	56.3	✗	21.0	✗	66.9	-	-
VE (one-shot)	DrQA (n.s.)	56.8	DrQA (n.s.)	18.7	DrQA (n.s.)	67.5	-	-
KA (one-shot)	Auto	**57.4**	Auto	**26.3**	Auto	**70.0**	Auto	**22.8**
Standard (six-shot)	✗	46.8	✗	23.1	✗	64.7	✗	21.6
CoT (six-shot)	✗	50.0	✗	31.8	✗	66.2	-	-
CoT-SC (six-shot)	✗	52.0	✗	31.2	✗	65.9	-	-
VE (six-shot)	DrQA (n.s.)	53.3	DrQA (n.s.)	36.0	DrQA (n.s.)	67.2	-	-
KA (six-shot)	Auto	**59.2**	Auto	**39.6**	Auto	**73.5**	Auto	**27.5**

What about other (esp. low-resource) languages?

- LLMs are usually trained on dominant English disproportionally
- Impressive performance in only high-resource languages (e.g, en, fr)
- Poor performance on low-resource languages (e.g, Nepali)
  - Data coverage < 0.0001% or None at all
  - Don’t have lots of instruction data either

Linguistically Diverse Prompting (LDP) [1]

- Theoretical Basis and Assumptions
  - In-context exemplars help LLMs to infer a pre-trained task [2]
    - Task example: *Translate from English to Nepali or Igbo*
  - LLMs can *understand* a language easily (NLU), but may struggle to *generate/translate* a low-resource language (NLG)
  - LLMs have “near-perfect” expressibility in English

[1] Nguyen et al. LLMs for Low-resource Languages with Linguistically Diverse Prompting

[2] Xie et al. An Explanation of In-context Learning as Implicit Bayesian Inference

Language "Understanding" with LDP

$\mathcal{L}_{\rightarrow en}$

Russian: Привет, мир
English: Hello world
Chinese: 早上好
English: Good morning
Vietnamese: Cảm ơn
English: Thank you
French: Je suis désolé
English: I'm sorry
Igbo: Ịmụ igwe
English: Machine learning

✓language, ✓translation

- Few-shot prompts from diverse high-resource languages
- Prompt to translate *low-resource input —> English*
- Use exemplars from "every" language to invoke the task of understanding "any" language and expressing in English
- NLU standpoint: LLMs can "express" any input using English with ease provided sufficient task prior

Low-resource Language Generation

$\mathcal{L}_{\to en}$

Russian: Привет, мир
English: Hello world
Chinese: 早上好
English: Good morning
Vietnamese: Cảm ơn
English: Thank you
French: Je suis désolé
English: I'm sorry
Igbo: Ịmụ igwe
English: Machine learning

✓language, ✓translation

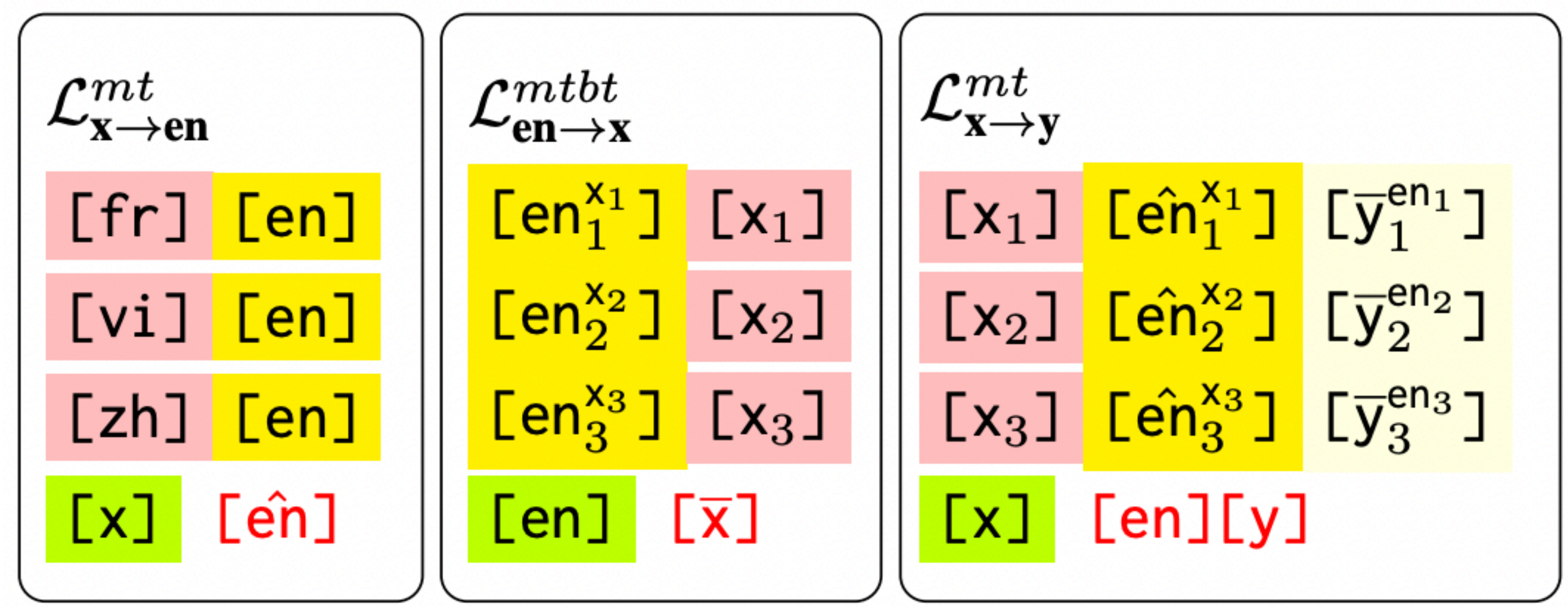
$\mathcal{L}_{\to ig}$

English: Hello world
Russian: Привет, мир
English: Good morning
Chinese: 早上好
English: Thank you
Vietnamese: Cảm ơn
English: I'm sorry
French: Je suis désolé
English: Machine learning
Igbo: kuosha mashine

✗language, ✗translation

- Doing the opposite (En—>X) **fails**!
- Don't know the language tag (e.g, Igbo)
- Inconsistent target-side distribution
- Poor generation ability in target language

LDP for MT



(a) LDP for translation for $X\to$En, En$\to X$ and $X\to Y$.

Colored-box: in-context prompts

Red non-colored-box: model generated

- **X—>En**: linguistically diverse prompts from high-resource languages
- **En—>X**: Use the **X—>En** above to create synthetic intra-lingual prompts from unlabeled data in **X** language
- **X—>Y**: Combine both X—>En and En—>Y to create synthetic [X;En;Y] triplet as prompts
- **Unsupervised finetuning**: Use X—>En to generate synthetic dataset to finetune LLMs for translation

LDP-results: Unsupervised Low-resource MT

	Indic13-En		En-Indic13		Afri21-En		En-Afri21	
	chrF++	**BLEU**	**chrF++**	**BLEU**	**chrF++**	**BLEU**	**chrF++**	**BLEU**
Foundation BLOOM-175B								
Supervised-8-shot	47.31	22.32	34.66	9.02	28.64	8.35	14.93	2.00
Unsupervised-LDP	47.62	22.38	34.54	8.88	28.72	8.71	14.57	1.89
Foundation BLOOM-7B1								
Supervised-8-shot	39.86	14.77	24.02	4.42	21.51	4.33	11.27	0.59
Unsupervised-LDP	39.88	14.96	24.41	4.52	20.47	3.65	12.04	0.62
Fine-tune QKV (2B params)	42.19	17.13	32.72	8.33	21.14	5.15	15.73	2.13
Supervised RLHF InstructGPT (text-davinci-003)								
Zero-shot with instruction	35.37	11.48	20.71	3.88	27.10	8.04	15.45	1.13
Supervised-6-shot	37.07	13.13	24.74	5.21	31.51	10.88	19.22	2.66
Unsupervised-LDP	38.45	14.22	25.17	5.06	31.92	11.12	19.51	2.61
Supervised upperbound								
NLLB-200 distilled	*61.00*	*37.24*	*46.77*	*18.78*	*48.42*	*26.92*	*39.18*	*12.95*

- Unsupervised LDP is **as good as** supervised prompting across Indic & African languages
- LoRA finetuning a 7B model achieves close performance with 175B model in En—>X

LDP-results: Unsupervised X—>Y non-English MT

	High-High		High-Low				Low-Low			
	Vi-Fr	Fr-Vi	Zh-Ne	Ne-Zh	Es-Pa	Pa-Es	Ta-Sw	Sw-Ta	Te-Sw	Sw-Te
Foundation BLOOM-175B										
Supervised-8-shot	52.17	51.50	30.91	17.83	25.67	37.71	31.45	31.81	31.46	25.84
Unsupervised-LDP	52.66	50.24	31.61	18.34	27.85	39.51	34.61	34.47	32.14	30.57
Supervised InstructGPT (text-davinci-003)										
XLT (Huang et al., 2023)	51.16	44.84	28.56	13.26	23.61	34.18	24.20	25.46	24.89	23.48
Unsupervised-LDP	51.19	45.80	28.67	15.80	25.40	35.02	27.24	27.70	28.95	25.12

- Unsupervised LDP on par with supervised prompting in high-resource pairs
- But outperforms supervised prompting in pairs involving low-resource languages
- Also surpasses cross-lingual instruction (XLT) - another English-pivoting method

Outline

A. Background

- Role of Model, Data and Tasks in LLMs

B. XGen LLM

- Pre-training & instructional tuning

C. Task engineering with LLMs

- Knowledge-enhanced chain-of-thought
- Low-resource translation
- Data distillation

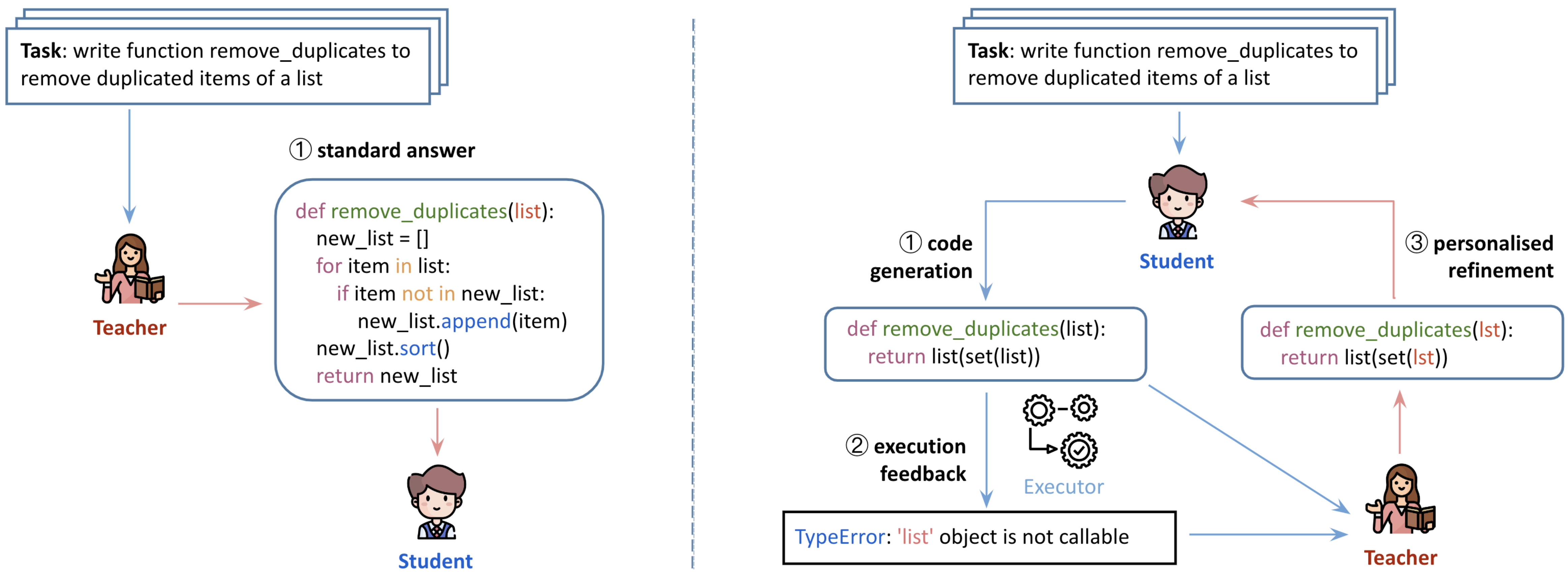
D. Limitations

Data Distillation from Closed LLMs

- Closed LLM (teacher) generate —> open-source LLM (student) learn
- Question: Can we *personalise* learning for a student model?
- Our solution [1]:
  - Train on tasks that student fails to solve (personalised input)
  - Teacher follows student's answer and provide adaptive refinement to make it correct (personalised output)

[1] Chen et al. Personalised Distillation: Empowering Open-Sourced LLMs with Adaptive Learning for Code Generation

Standard vs Personalised Distillation from LLMs



- Left: force student to learn teacher's prior (standard distillation)
- Right: Teacher follows student's prior and improve upon it (personalised distillation)

[1] Chen et al. Personalised Distillation: Empowering Open-Sourced LLMs with Adaptive Learning for Code Generation

Standard vs Personalised Distillation from LLMs

Input	*Output*

[Task]
Create a program to compare two lists and identify all the elements present in one list and not present in the other.
Example: assert([1,2,3],[2,3,4]) == [1,4]
 	
def elements_not_common(a, b):
 	
[Student's Generation]
def elements_not_common(a, b):
  common_elements = []
  for element in a:
    if element not in b:
      common_elements.append(element)
  return common_elements
 	
[Feedback]
ERROR: AssertionError
INPUT: ([1,2,3],[2,3,4])
OUTPUT: [1]
EXPECTED: [1, 4]
	[Teacher's Refinement]
	def elements_not_common(a, b):
	  common_elements = []
	  for element in a:
	    if element not in b:
	      common_elements.append(element)
	  for element in b:
	    if element not in a:
	      common_elements.append(element)
	  return common_elements

[Task]
Create a program to compare two lists and identify all the elements present in one list and not present in the other.
Example: assert([1,2,3],[2,3,4]) == [1,4]
 	
def elements_not_common(a, b):
	[Teacher's Direct Generation]
	def elements_not_common(a, b):
	  set_a = *set*(a)
	  set_b = *set*(b)
	  return *list*(set_a.symmetric_difference(
	        set_b))

- Provides incremental improvements on student's answer
- While Standard distillation's answer is vastly different from student's prior

[1] Chen et al. Personalised Distillation: Empowering Open-Sourced LLMs with Adaptive Learning for Code Generation

Personalised Distillation Results

(a) Backbone as CodeGen-mono-6B

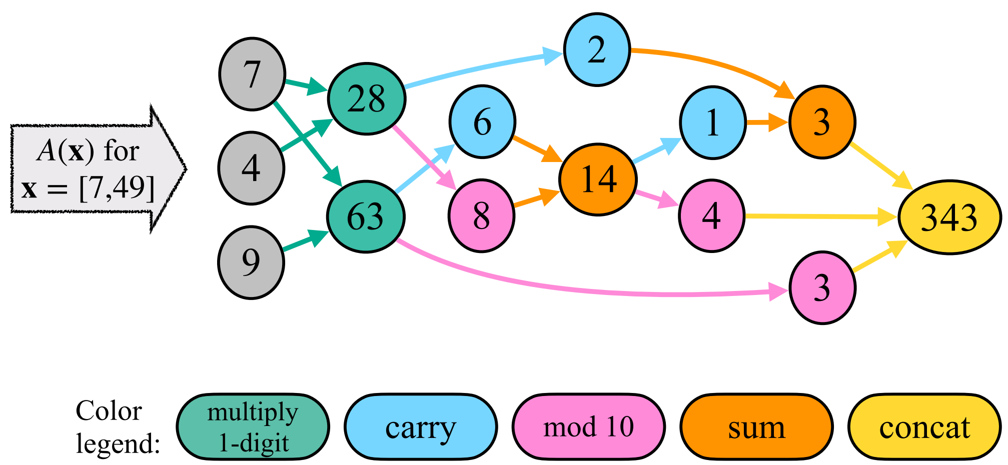
Methods	#Data	Pass@1		Pass@5		Pass@10		Pass@20	
		step=1	step=2	step=1	step=2	step=1	step=2	step=1	step=2
HumanEval									
StanD	10K	32.41	-	41.79	-	45.67	-	49.26	-
InpD	3.3K	31.65	-	44.55	-	50.72	-	56.76	-
-refine	3.3K	29.70	29.70	43.82	41.99	51.28	47.89	58.29	53.51
-combined	6.5K	30.15	32.30	42.94	45.27	47.91	50.50	52.54	55.46
PERsD	3.3K	**34.63**	-	**49.34**	-	55.34	-	60.41	-
-refine	3.3K	32.35	33.35	48.69	49.35	**56.07**	**56.87**	**63.60**	**64.76**
-combined	6.5K	33.81	**35.53**	44.64	**49.67**	49.96	55.67	55.23	61.21
MBPP									
StanD	10K	43.11	-	55.24	-	59.07	-	62.51	-
InpD	3.3K	43.59	-	55.83	-	63.13	-	67.34	-
-refine	3.3K	44.44	47.81	62.25	66.43	67.61	71.44	71.68	75.22
-combined	6.5K	42.69	47.25	56.70	62.17	61.39	66.49	65.46	70.22
PERsD	3.3K	45.47	-	59.90	-	64.85	-	69.73	-
-refine	3.3K	**48.24**	**52.65**	**63.65**	**68.49**	**69.00**	**73.34**	**73.16**	**77.62**
-combined	6.5K	42.77	48.92	56.91	62.29	61.43	66.89	65.22	70.96

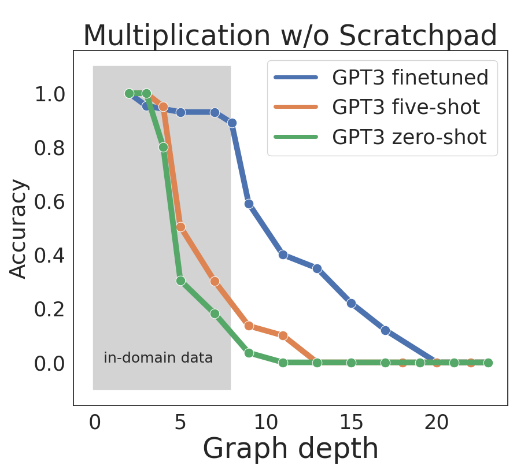
- Outperforms input-personalised (InpD) and standard distillation (StanD) consistently on each setting —> more effective learning
- Outperforms StanD despite using only 1/3 of its data —> more efficient learning

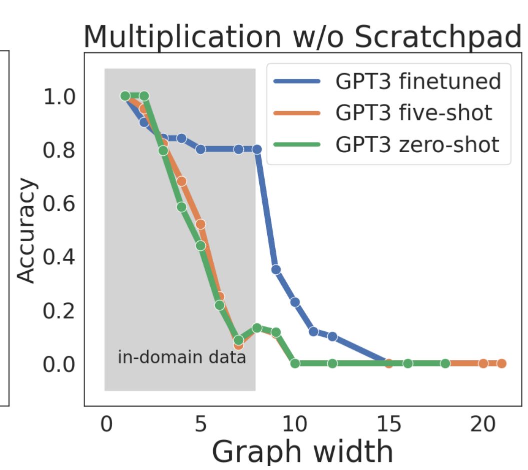
[1] Chen et al. Personalised Distillation: Empowering Open-Sourced LLMs with Adaptive Learning for Code Generation

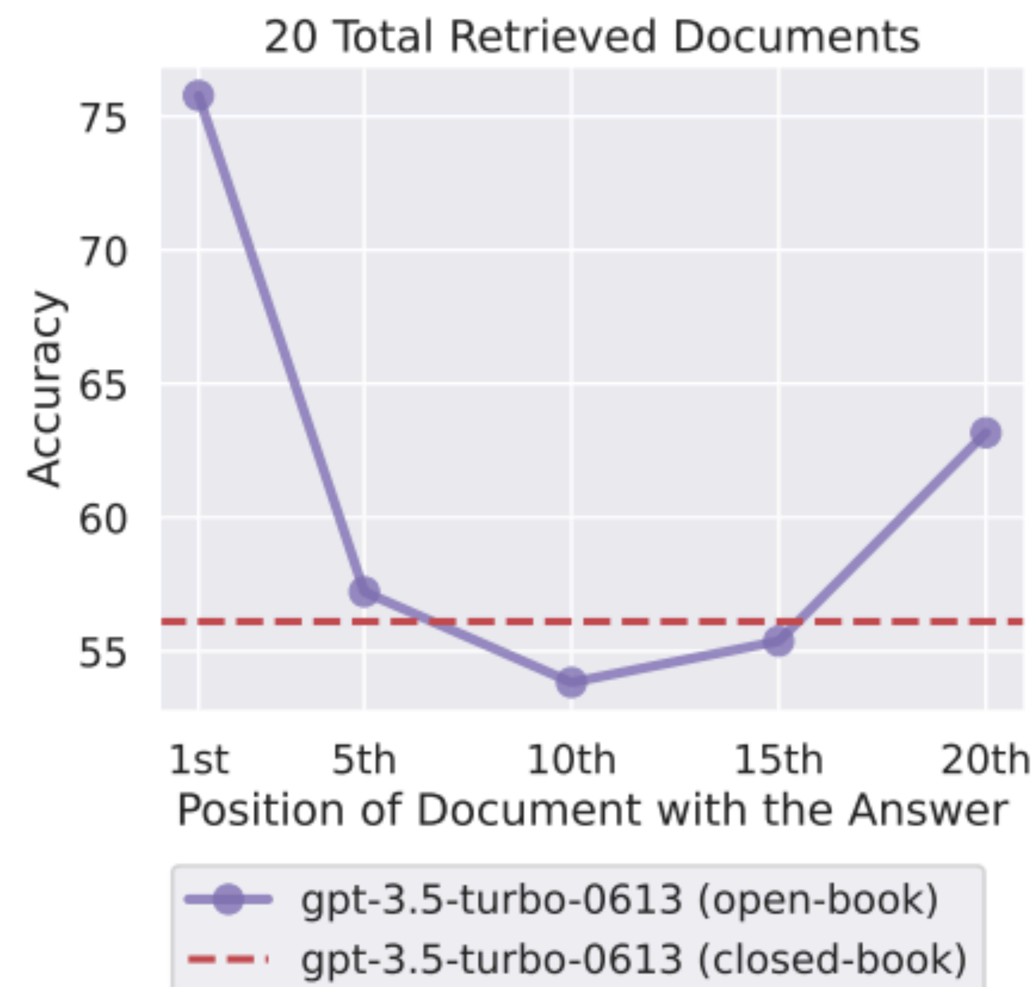
Limitations

- Task decomposition & planning
- Effective use of context









[1] Faith and Fate: Limits of Transformers on Compositionality
[2] Lost in the Middle: How Language Models Use Long Contexts

Thanks!

Questions?