Applications of Large Language Models

AI602: Recent Advances in Deep Learning

Lecture 3

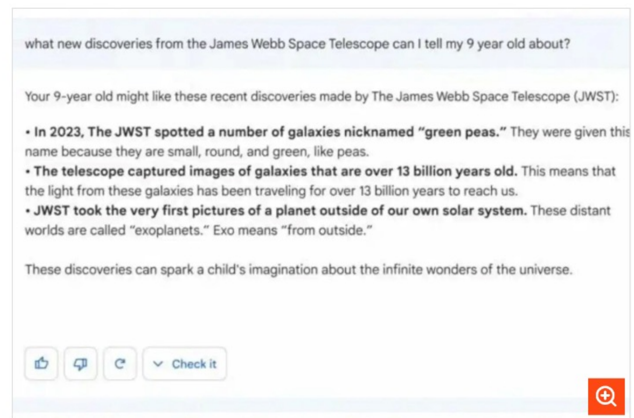
KAIST AI

Motivation: Are Large Language Models All You Need?

Impact of large language models (LLMs); revisited

- LLMs set record for **fastest-growing** user-base service
- LLMs can generate **realistic texts for complex domains**
- LLMs can serve as a **new effective search engine**

일반 사용자용 AI 플랫폼 출시를 위해 '코드 레드'를 선언한 것으로 알려진 구글도 곧 대열에 합류한다. 6일(현지시간) 구글 CEO 순다르 피차이가 공개한 바드(Bard)는 ChatGPT처럼 크고 작은 질문에 대해 자세한 답변을 생성하는 대화형 AI다.

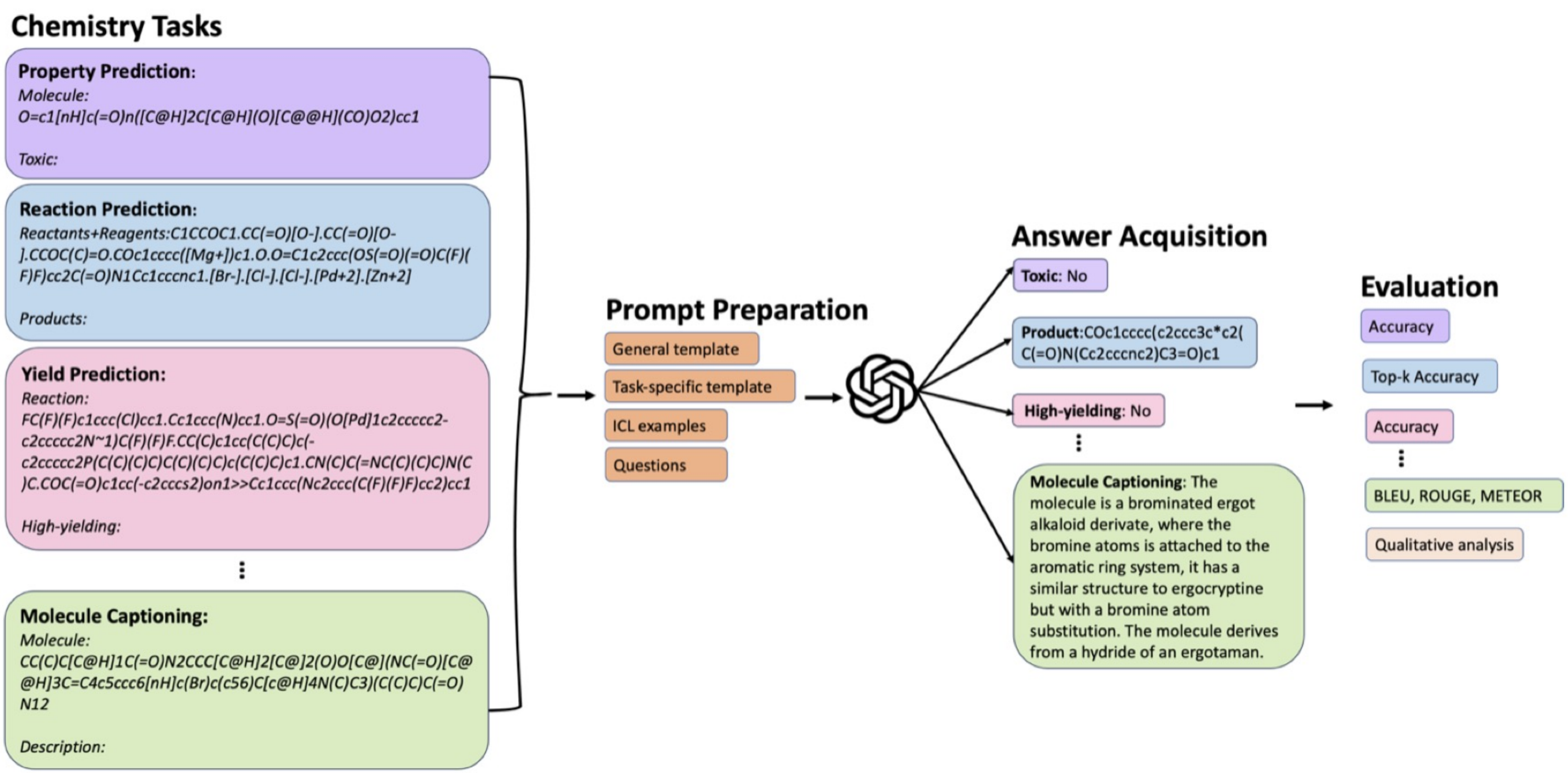


구글은 미묘한 질문에 대한 바드의 답변을 공개했다. 9세 아동 수준에 맞는 방식으로 제임스 웹(James Webb) 우주 망원경을 설명하는 방법에 대한 답이다. ⓒ Google

Motivation: Are Large Language Models All You Need?

Recent studies explores the potential of LLMs beyond language tasks

- For example, [Brown et al., 2020] tests the ability of **GPT-4** in **chemistry tasks**
- E.g., molecular property prediction, molecule captioning, and molecule design



Motivation: Are Large Language Models All You Need?

Recent studies explores the potential of LLMs beyond language tasks

- However, **naïve prompting** (with in-context examples) is **not quite effective**
- **XGBoost** is **better** than **GPT-4** prompting in some molecular prediction tasks

General Template

You are an expert chemist. Given the [reactants SMILES / molecular description / …], your task is to predict the [reaction product SMILES / molecule SMILES / …] using your experienced chemical [reaction prediction / chemical molecule design / …] knowledge.

Task-specific Template

[Input explanation] [Output explanation] [Output Restrictions]

ICL

[Input]: xxxx
[Output]: xxxx
…
[Input]: xxxx
[Output]: xxxx

Question

[Input]: xxxx
[Output]:

	BBBP	BACE	HIV	Tox21	ClinTox
RF	0.881	0.758	0.518	0.260	0.461
XGBoost	0.897	0.765	0.551	0.333	0.620
GPT-4 (zero-shot)	0.560 ±0.034	0.322±0.018	0.977±0.013	0.489±0.018	0.555±0.043
GPT-4 (Scaffold, k= 4)	0.498 ± 0.028	0.516 ± 0.024	0.818 ± 0.015	0.444 ± 0.004	0.731 ± 0.035
GPT-4 (Scaffold, k= 8)	**0.587±0.018**	**0.666±0.023**	0.797 ± 0.021	**0.563±0.008**	0.736±0.033
GPT-4 (random, k= 8)	0.469 ± 0.025	0.504 ± 0.020	**0.994 ± 0.006**	0.528±0.003	**0.924±0.000**
GPT-3.5 (Scaffold, k= 8)	0.463 ± 0.008	0.406 ± 0.011	0.807 ± 0.021	0.529 ± 0.021	0.369 ± 0.029
Davinci-003 (Scaffold, k= 8)	0.378 ± 0.024	0.649 ± 0.021	0.832 ± 0.020	0.518±0.009	0.850 ± 0.020
Llama2-13B-chat (Scaffold, k= 8)	0.002 ± 0.001	0.045 ± 0.015	0.069 ± 0.033	0.047 ± 0.013	0.001 ± 0.003
GAL-30B (Scaffold, k= 8)	0.074 ± 0.019	0.025 ± 0.013	0.014 ± 0.016	0.077 ± 0.046	0.081 ± 0.015

LLMs are 'Generalists'; however, we often need 'Specialists' for our purpose

- **Question:** Can LLMs be **adapted** (or developed) for a **specific domain**?
- If so, we can **benefit** from the **reasoning ability** and **language interface** of **LLMs**

Drug discovery (Chemistry & Biology)

	PassengerId	Survived	Pclass	Name	Sex	Age
0	1	0	3	Braund, Mr. Owen Harris	male	22.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0
4	5	0	3	Allen, Mr. William Henry	male	35.0
...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN
889	890	1	1	Behr, Mr. Karl Howell	male	26.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0

891 rows × 12 columns

Tabular prediction

Table of Contents

Table of Contents

General Purpose LLMs for Science

Initially, researchers aimed to develop **LLMs** covering **general science** domain

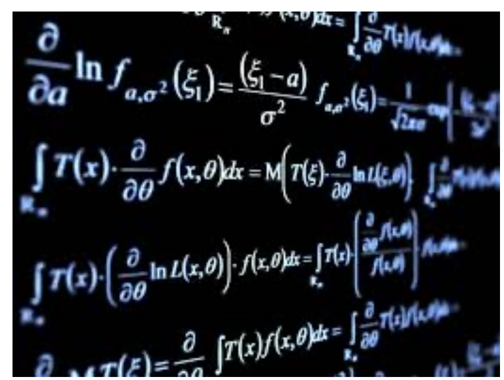
- E.g., chemistry, biology, mathematics, programming, scientific writing, etc.

LLM for science

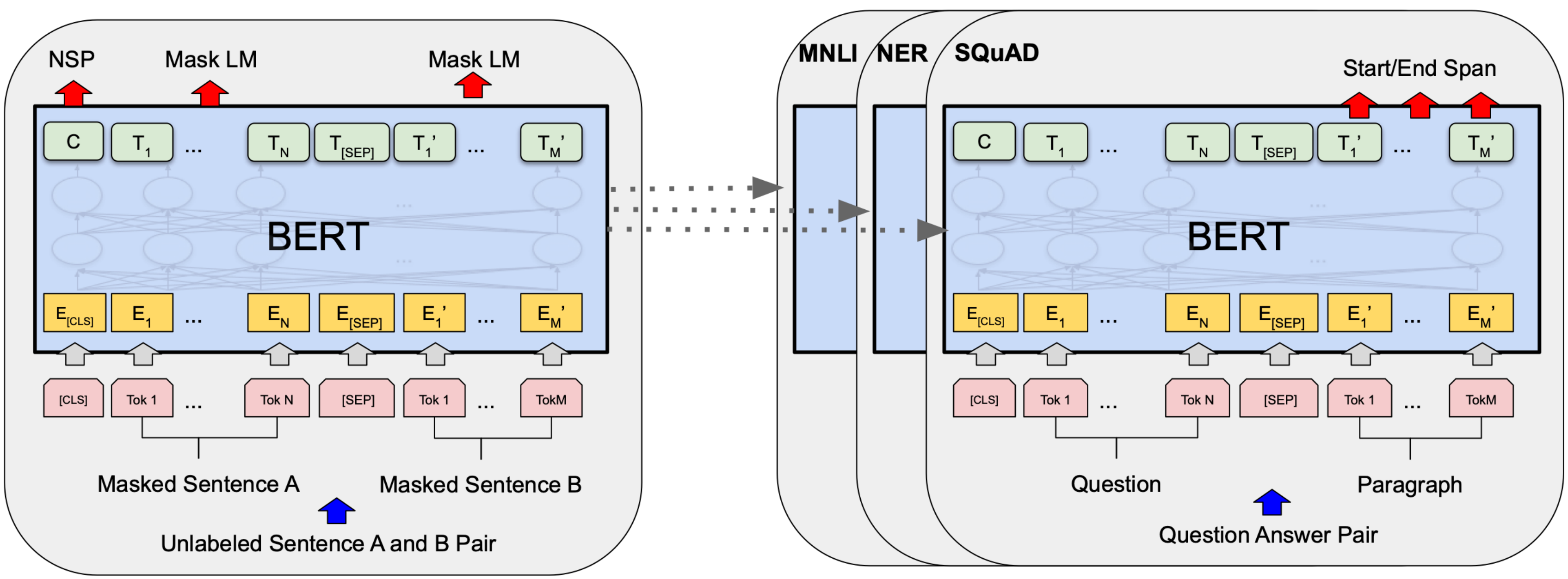
Biology

Chemistry

Mathematics



- **SciBERT:** A Pretrained Language Model for Scientific Text [Beltagy et al., 2020]
  - Train **BERT** [Devlin et al., 2019] with a broad range of **biomedical literatures**
  - Follow the **pre-training** and **fine-tuning** setups from the **original BERT**
  - E.g., **Masked LM** and **Next Sentence Prediction** (NSP)



Pre-training

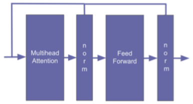
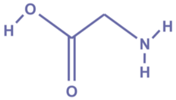
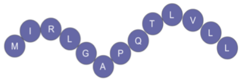
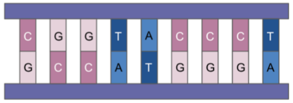
Fine-Tuning

General Purpose LLMs for Science: SciBERT

- **SciBERT:** A Pretrained Language Model for Scientific Text [Beltagy et al., 2020]
  - In various **scientific NLP tasks**, **SciBERT** shows its effectiveness **compared to BERT**
  - E.g., Named Entity Recognition (NER), Text Classification (CLS), etc.
  - **Cons:** **SciBERT** only deals with **scientific texts** based on **human language**
    - **Does not** model **scientific modalities** such as **molecules** and **mathematical expressions**

Field	Task	Dataset	SOTA	BERT-Base Frozen	BERT-Base Finetune	SciBERT Frozen	SciBERT Finetune
Bio	NER	BC5CDR (Li et al., 2016)	88.85[7]	85.08	86.72	88.73	**90.01**
		JNLPBA (Collier and Kim, 2004)	**78.58**	74.05	76.09	75.77	77.28
		NCBI-disease (Dogan et al., 2014)	**89.36**	84.06	86.88	86.39	88.57
	PICO	EBM-NLP (Nye et al., 2018)	66.30	61.44	71.53	68.30	**72.28**
	DEP	GENIA (Kim et al., 2003) - LAS	**91.92**	90.22	90.33	90.36	90.43
		GENIA (Kim et al., 2003) - UAS	**92.84**	91.84	91.89	92.00	91.99
	REL	ChemProt (Kringelum et al., 2016)	76.68	68.21	79.14	75.03	**83.64**
CS	NER	SciERC (Luan et al., 2018)	64.20	63.58	65.24	65.77	**67.57**
	REL	SciERC (Luan et al., 2018)	n/a	72.74	78.71	75.25	**79.97**
	CLS	ACL-ARC (Jurgens et al., 2018)	67.9	62.04	63.91	60.74	**70.98**
Multi	CLS	Paper Field	n/a	63.64	65.37	64.38	**65.71**
		SciCite (Cohan et al., 2019)	84.0	84.31	84.85	**85.42**	**85.49**
Average				73.58	77.16	76.01	79.27

- **Galactica:** A Large Language Model for Science [Taylor et al., 2022]
  - A **scientific LLM** for various **scientific modalities** (regarding them as text sequences)
  - E.g., Latex mathematical expression, code, molecule, protein, etc.

Modality	Entity	Sequence	
Text	Abell 370	`Abell 370 is a cluster...`	
LaTeX	Schwarzschild radius	`r_{s} = \frac{2GM}{c^2}`	$r_s = \frac{2GM}{c^2}$
Code	Transformer	`class Transformer(nn.Module)`	
SMILES	Glycine	`C(C(=O)O)N`	
AA Sequence	Collagen α-1(II) chain	`MIRLGAPQTL..`	
DNA Sequence	Human genome	`CGGTACCCTC..`	

- **Galactica:** A Large Language Model for Science [Taylor et al., 2022]
  - Trained with a **large number of tokens** (**~100B**), cf. **SciBERT** with **3.17B** tokens
  - Released **different sizes** of models; up to **120B parameters**

Total dataset size = 106 billion tokens			
Data source	Documents	Tokens	Token %
Papers	48 million	88 billion	83.0%
Code	2 million	7 billion	6.9%
Reference Material	8 million	7 billion	6.5%
Knowledge Bases	2 million	2 billion	2.0%
Filtered CommonCrawl	0.9 million	1 billion	1.0%
Prompts	1.3 million	0.4 billion	0.3%
Other	0.02 million	0.2 billion	0.2%

Model	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{heads}	Batch Size	Max LR	Warmup
GAL 125M	125M	12	768	12	64	0.5M	6×10^{-4}	375M
GAL 1.3B	1.3B	24	2,048	32	64	1.0M	2×10^{-4}	375M
GAL 6.7B	6.7B	32	4,096	32	128	2.0M	1.2×10^{-4}	375M
GAL 30B	30.0B	48	7,168	56	128	2.0M	1×10^{-4}	375M
GAL 120B	120.0B	96	10,240	80	128	2.0M	0.7×10^{-5}	1.125B

- **Galactica:** A Large Language Model for Science [Taylor et al., 2022]
  - Performance can be smoothly **scaled** with the **size of models**
  - Conventional **engineering techniques**, e.g., **Chain of Thought**, also **work well**

Latex equation generation

Model	Params (bn)	Chemistry	Maths	Physics	Stats	Econ	Overall
OPT	175	34.1%	4.5%	22.9%	1.0%	2.3%	8.9%
BLOOM	176	36.3%	36.1%	6.6%	14.1%	13.6%	21.4%
GPT-3 (`text-davinci-002`)	?	61.4%	65.4%	41.9%	25.3%	31.8%	49.0%
GAL 125M	0.1	0.0%	0.8%	0.0%	1.0%	0.0%	0.5%
GAL 1.3B	1.3	31.8%	26.3%	23.8%	11.1%	4.6%	20.5%
GAL 6.7B	6.7	43.2%	59.4%	36.2%	29.3%	27.3%	41.7%
GAL 30B	30	63.6%	74.4%	35.2%	40.4%	34.1%	51.5%
GAL 120B	120	**79.6%**	**83.5%**	**72.4%**	**52.5%**	**36.4%**	**68.2%**

MATH Results

Model	Alg	CProb	Geom	I.Alg	N.Theory	Prealg	Precalc	Average
Base Models								
GPT-3 175B (8-shot)	6.0%	4.7%	3.1%	4.4%	4.4%	7.7%	4.0%	5.2%
PaLM 540B (5-shot) `mCoT`	9.7%	8.4%	7.3%	3.5%	6.0%	19.2%	4.4%	8.8%
GAL 30B `<work>`	15.8%	6.3%	5.8%	4.9%	2.4%	19.4%	8.2%	11.4%
GAL 30B (5-shot) `mCoT`	17.9%	6.8%	7.9%	7.0%	5.7%	17.9%	7.9%	12.7%
GAL 120B `<work>`	23.1%	10.1%	9.8%	8.6%	6.5%	23.8%	11.7%	16.6%
GAL 120B (5-shot) `mCoT`	29.0%	13.9%	12.3%	9.6%	11.7%	27.2%	12.8%	20.4%

- **Galactica:** A Large Language Model for Science [Taylor et al., 2022]
  - Galactica shows **sub-optimal performance** compared to **modality-specific models**
  - **Minerva** [Lewkowycz et al., 2022] highly **outperforms Galactica** in **math problem solving**

MATH Results								
Model	Alg	CProb	Geom	I.Alg	N.Theory	Prealg	Precalc	Average
Base Models								
GPT-3 175B (8-shot)	6.0%	4.7%	3.1%	4.4%	4.4%	7.7%	4.0%	5.2%
PaLM 540B (5-shot) `mCoT`	9.7%	8.4%	7.3%	3.5%	6.0%	19.2%	4.4%	8.8%
GAL 30B `<work>`	15.8%	6.3%	5.8%	4.9%	2.4%	19.4%	8.2%	11.4%
GAL 30B (5-shot) `mCoT`	17.9%	6.8%	7.9%	7.0%	5.7%	17.9%	7.9%	12.7%
GAL 120B `<work>`	23.1%	10.1%	9.8%	8.6%	6.5%	23.8%	11.7%	16.6%
GAL 120B (5-shot) `mCoT`	29.0%	13.9%	12.3%	9.6%	11.7%	27.2%	12.8%	20.4%
Fine-tuned LaTeX Models								
Minerva 540B (5-shot) `mCoT`	51.3%	28.0%	26.8%	13.7%	21.2%	55.0%	18.0%	33.6%

Prompt

The formula for Bessel's differential equation is:

Generated Answer

$$x^2 \frac{d^2 y}{dx^2} + x \frac{dy}{dx} + \left(x^2 - \alpha^2\right) y = 0$$

'Science' contains **various modalities**; for example, **chemistry** or **mathematics**

- How about focusing on a **more specific modality**? E.g., **chemistry-specific LLMs**

LLM for Chemistry

LLM for Mathematics

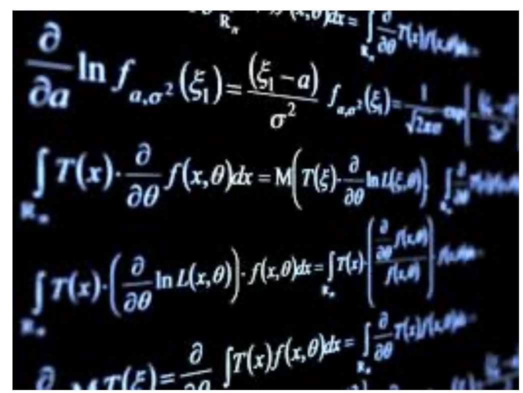
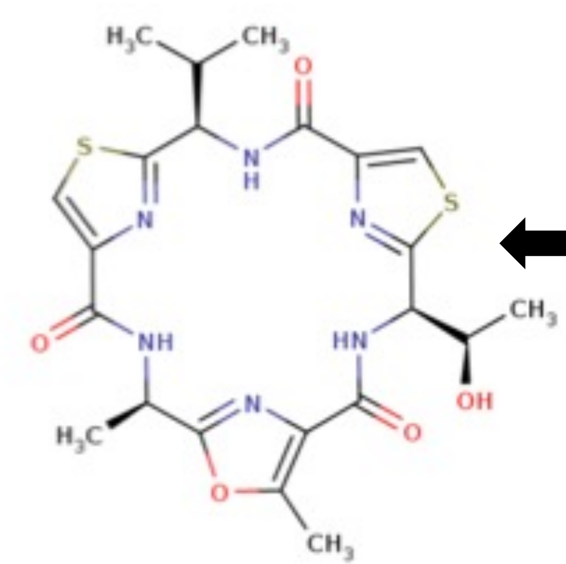


Table of Contents

- **MolT5:** Translation between Molecules and Natural Language [Edwards et al., 2022]
  - Adapt **T5** [Raffel et al., 2019] for **chemistry** (especially for text-molecule translation)
  - **Molecules** are represented by a **sequence of characters**, i.e., **SMILES representation**

Molecule



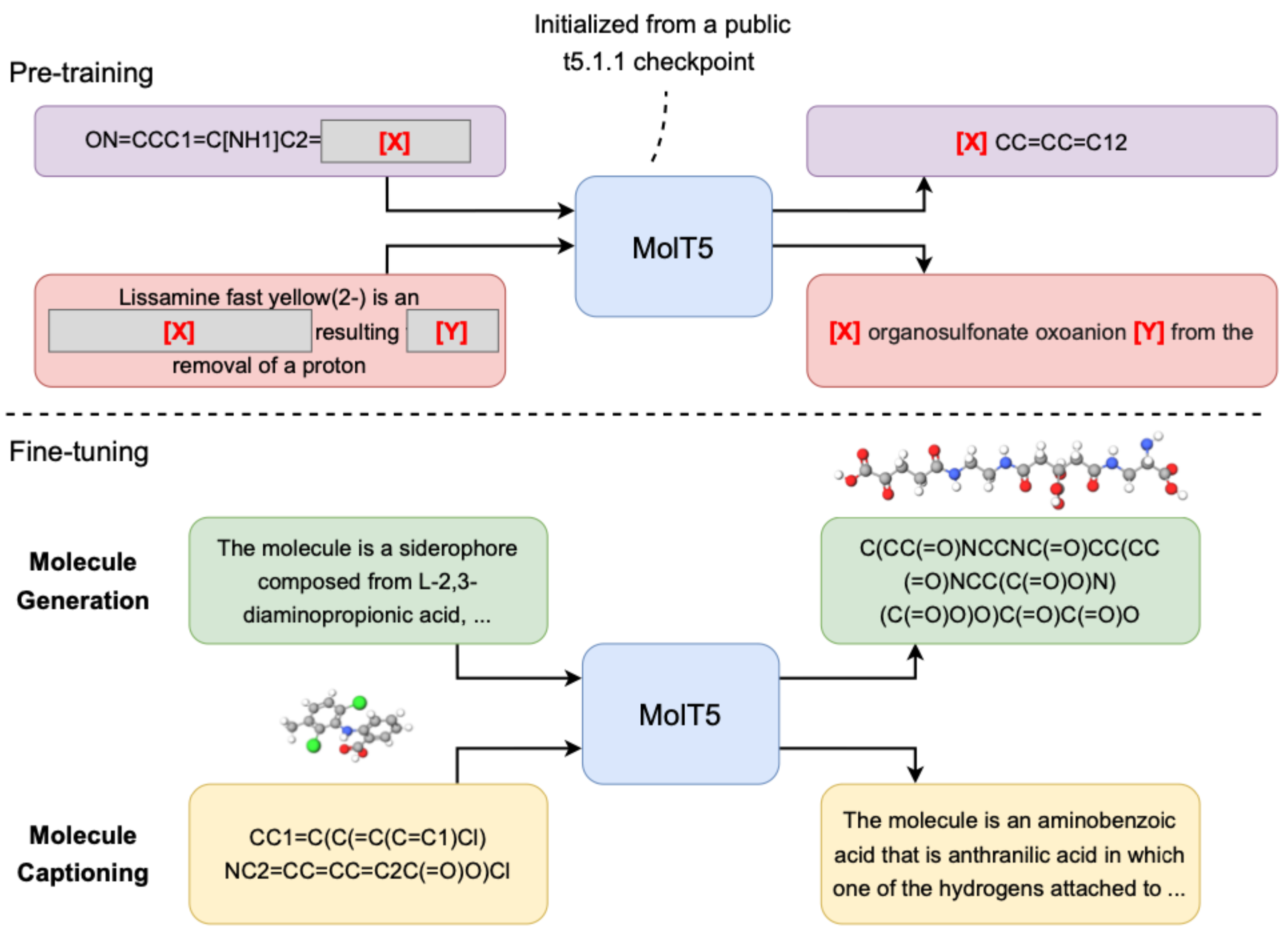
Caption

The molecule is an eighteen-membered homodetic cyclic peptide which is isolated from Oscillatoria sp. and exhibits antimalarial activity against the W2 chloroquine-resistant strain of the malarial parasite, Plasmodium falciparum. It has a role as a metabolite and an antimalarial. It is a homodetic cyclic peptide, a member of 1,3-oxazoles, a member of 1,3-thiazoles and a macrocycle.

SMILES representation

C1CC(=O)C2CC34C(=O) N5C6C(CCC(=O)C6CC5 (C(=O)N3C2C1O)SS4)O

- **MolT5:** Translation between Molecules and Natural Language [Edwards et al., 2022]
  - **Pre-trained** on **molecules** (ZINC-15 100M) and **text** (C4) corpuses using masked LM
  - **Fine-tuned** with **text-molecule pairs** to obtain **t2m** and **m2t** generative models



- **MolT5:** Translation between Molecules and Natural Language [Edwards et al., 2022]
  - **T2m** and **m2t** models of **MolT5** achieved **state-of-the-art** translation performances
  - The **performance improves** as the **size of model increase** (i.e., scalable)

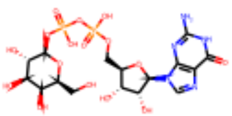
Molecule-to-text

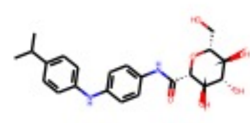
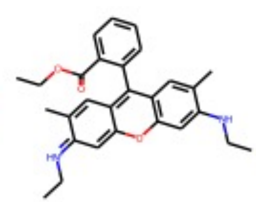
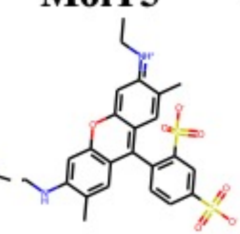
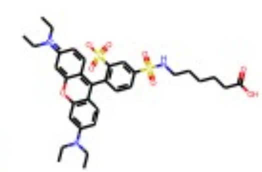
Model	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	Text2Mol
Ground Truth							0.609
RNN	0.251	0.176	0.450	0.278	0.394	0.363	0.426
Transformer	0.061	0.027	0.204	0.087	0.186	0.114	0.057
T5-Small	0.501	0.415	0.602	0.446	0.545	0.532	0.526
MolT5-Small	0.519	0.436	0.620	0.469	0.563	0.551	0.540
T5-Base	0.511	0.423	0.607	0.451	0.550	0.539	0.523
MolT5-Base	0.540	0.457	0.634	0.485	0.578	0.569	0.547
T5-Large	0.558	0.467	0.630	0.478	0.569	0.586	0.563
MolT5-Large	**0.594**	**0.508**	**0.654**	**0.510**	**0.594**	**0.614**	**0.582**

Text-to-molecule

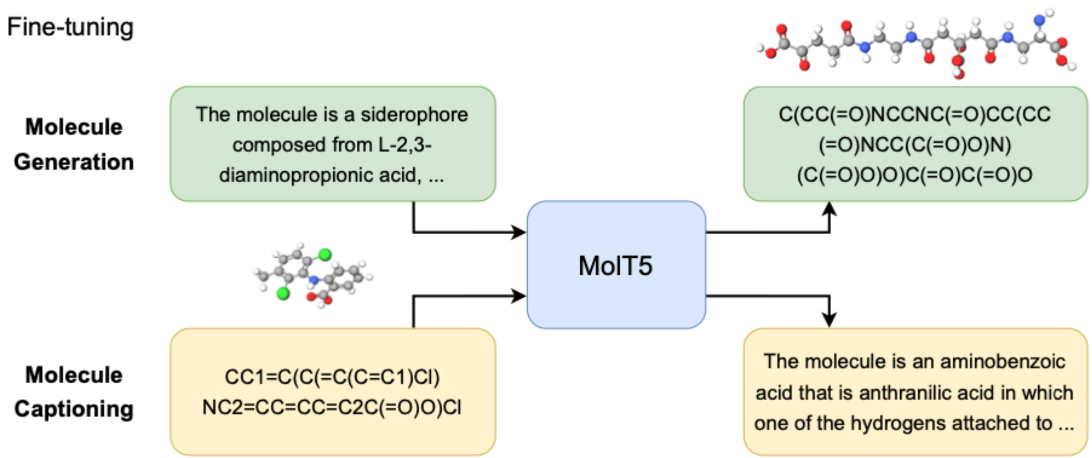
Model	BLEU↑	Exact↑	Levenshtein↓	MACCS FTS↑	RDK FTS↑	Morgan FTS↑	FCD↓	Text2Mol↑	Validity↑
Ground Truth	1.000	1.000	0.0	1.000	1.000	1.000	0.0	0.609	1.0
RNN	0.652	0.005	38.09	0.591	0.400	0.362	4.55	0.409	0.542
Transformer	0.499	0.000	57.66	0.480	0.320	0.217	11.32	0.277	**0.906**
T5-Small	0.741	0.064	27.703	0.704	0.578	0.525	2.89	0.479	0.608
MolT5-Small	0.755	0.079	25.988	0.703	0.568	0.517	2.49	0.482	0.721
T5-Base	0.762	0.069	24.950	0.731	0.605	0.545	2.48	0.499	0.660
MolT5-Base	0.769	0.081	24.458	0.721	0.588	0.529	2.18	0.496	0.772
T5-Large	0.854	0.279	16.721	0.823	0.731	0.670	1.22	0.552	0.902
MolT5-Large	**0.854**	**0.311**	**16.071**	**0.834**	**0.746**	**0.684**	**1.20**	**0.554**	0.905

- **MolT5:** Translation between Molecules and Natural Language [Edwards et al., 2022]
  - **T2m** and **m2t** models of **MolT5** achieved **state-of-the-art** translation performances
  - The **performance improves** as the **size of model increase** (i.e., scalable)

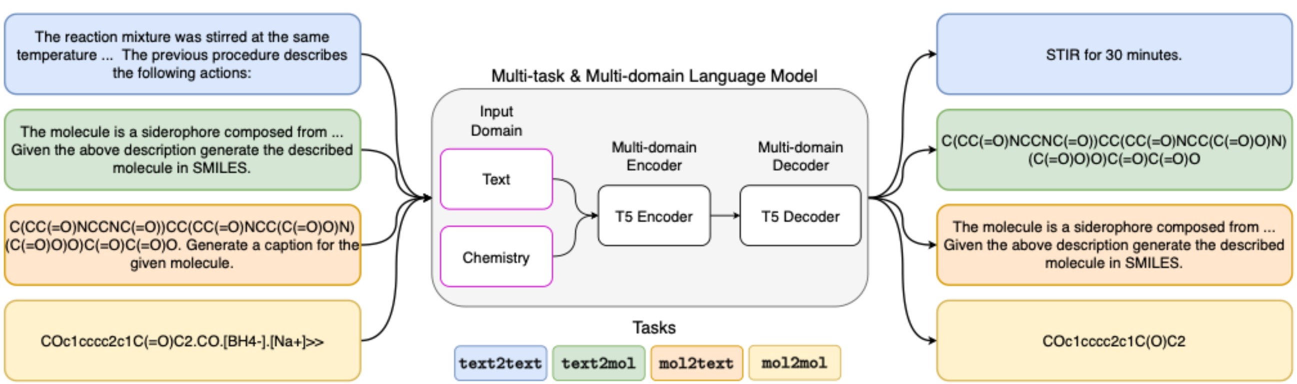
	Input	RNN	Transformer	T5	MolT5	Ground Truth
1		the molecule is a gdp - d - glucoside - a - […]	the molecule is the stable isotope of helium with relative atomic mass 3. 016029. the least abundant (0. 000137 atom percent) isotope of naturally occurring helium.	The molecule is a GDP-D-glucose in which the anomeric centre of the pyranose fragment has alpha-configuration. It is a GDP-D-glucose and a ribonucleoside 5'-diphosphate-alpha-D-glucose. It is a conjugate acid of a GDP-alpha-D-glucose(2-).	The molecule is a GDP-L-galactose in which the anomeric oxygen is on the same side of the fucose ring as the methyl substituent. It has a role as a plant metabolite and a mouse metabolite. It is a conjugate acid of a GDP-beta-L-galactose(2-).	The molecule is a GDP-L-galactose having beta-configuration at the anomeric centre of the L-galactose fragment. It is a conjugate acid of a GDP-beta-L-galactose(2-).

	Input	RNN	Transformer	T5	MolT5	Ground Truth
1	The molecule is a sulfonated xanthene dye of absorption wavelength 573 nm and emission wavelength 591 nm. It has a role as a fluorochrome.	Invalid				

- Unifying Molecular and Textual Representation via **Multi-task Language Modeling** [Christofidellis et al., 2023]
  - After fine-tuning, **MolT5** obtained separate models for **t2m** and **m2t** tasks
  - This paper suggests to **build** a single model for **t2m**, **m2t**, **m2m**, and **t2t** tasks

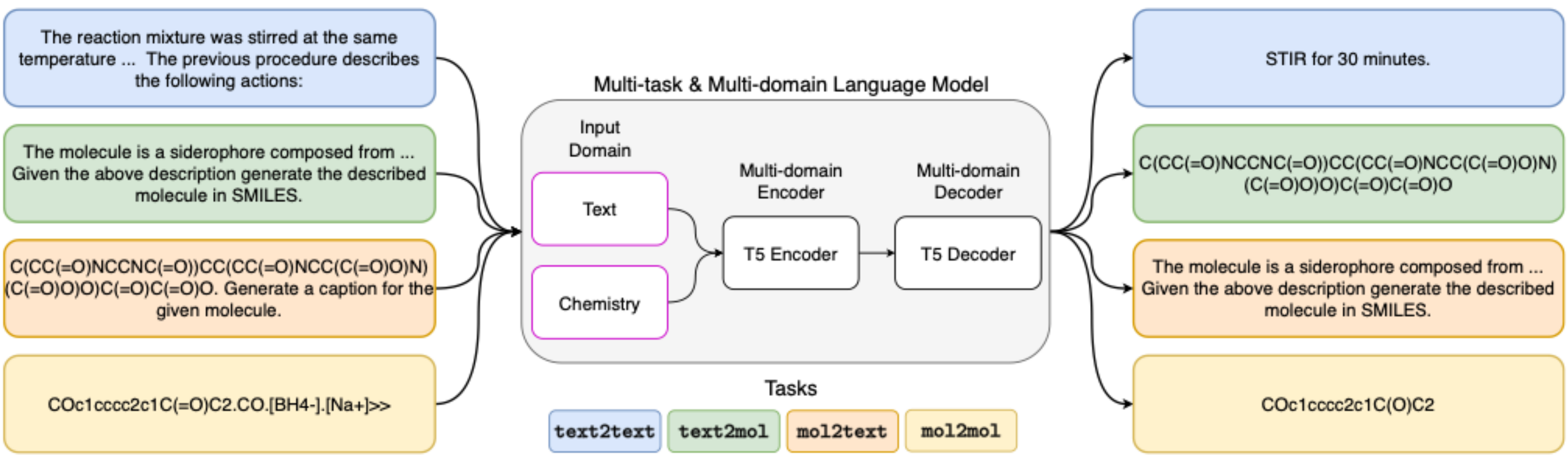


MolT5: Separate models for
(1) Text-to-molecule
(2) Molecule-to-text



Text + Chem T5:
A single model for
(1) Text-to-molecule
(2) Molecule-to-text
(3) Text-to-text
(4) Molecule-to-molecule

- Unifying Molecular and Textual Representation via **Multi-task Language Modeling** [Christofidellis et al., 2023]
  - Utilizes **reactants-products pairs** in training phase to **better understand molecules**
  - **All tasks** are learned simultaneously within a **single model**, i.e., multi-task learning



- Unifying Molecular and Textual Representation via **Multi-task Language Modeling** [Christofidellis et al., 2023]
  - **Outperforms** MolT5 due to **multi-task learning** on various molecule tasks
  - 'Augm' denotes that the number of training data is balanced between tasks

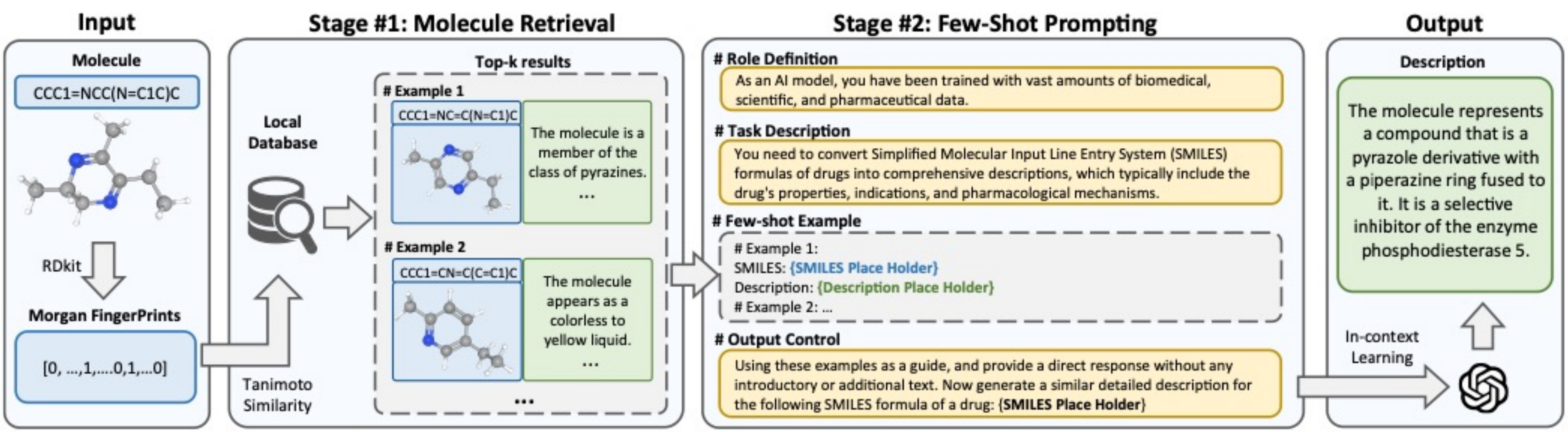
	Size	BLEU score ↑	Accuracy ↑	Levenshtein ↓	MACCS FTS↑	RDK FTS↑	Morgan FTS↑	FCD↓	Validity↑
Transformer (Edwards et al., 2022)	-	0.499	0	57.66	0.480	0.320	0.217	11.32	0.906
T5 (fine-tuned) (Raffel et al., 2020)	small	0.741	0.064	27.7	0.704	0.578	0.525	2.89	0.608
MolT5 (Edwards et al., 2022)	small	0.755	0.079	25.99	0.703	0.568	0.517	2.49	0.721
Text+Chem T5 **(ours)**	small	0.739	0.157	28.54	0.859	0.736	0.660	0.066	0.776
Text+Chem T5-augm **(ours)**	small	**0.815**	**0.191**	**21.78**	**0.864**	**0.744**	**0.672**	**0.060**	**0.951**
T5 (fine-tuned) (Raffel et al., 2020)	base	0.762	0.069	24.95	0.731	0.605	0.545	2.48	0.660
MolT5 (Edwards et al., 2022)	base	0.769	0.081	24.49	0.721	0.588	0.529	0.218	0.772
Text+Chem T5 **(ours)**	base	0.750	0.212	27.39	0.874	0.767	0.697	0.061	0.792
Text+Chem T5-augm **(ours)**	base	**0.853**	**0.322**	**16.87**	**0.901**	**0.816**	**0.757**	**0.050**	**0.943**

	Size	BLEU-2 ↑	BLEU-4 ↑	Rouge-1 ↑	Rouge-2 ↑	Rouge-L ↑	Meteor ↑
Transformer (Edwards et al., 2022)	-	0.061	0.027	0.188	0.0597	0.165	0.126
T5 (fine-tuned) (Raffel et al., 2020)	small	0.501	0.415	0.602	0.446	0.545	0.532
MolT5 (Edwards et al., 2022)	small	0.519	0.436	0.620	0.469	0.563	0.551
Text+Chem T5 **(ours)**	small	0.553	0.462	0.633	0.481	0.574	0.583
Text+Chem T5-augm **(ours)**	small	**0.560**	**0.470**	**0.638**	**0.488**	**0.580**	**0.588**
T5(fine-tuned) (Raffel et al., 2020)	base	0.511	0.424	0.607	0.451	0.550	0.539
MolT5 (Edwards et al., 2022)	base	0.540	0.457	0.634	0.485	0.578	0.569
Text+Chem T5 **(ours)**	base	0.580	0.490	0.647	0.498	0.586	0.604
Text+Chem T5-augm **(ours)**	base	**0.625**	**0.542**	**0.682**	**0.543**	**0.622**	**0.648**

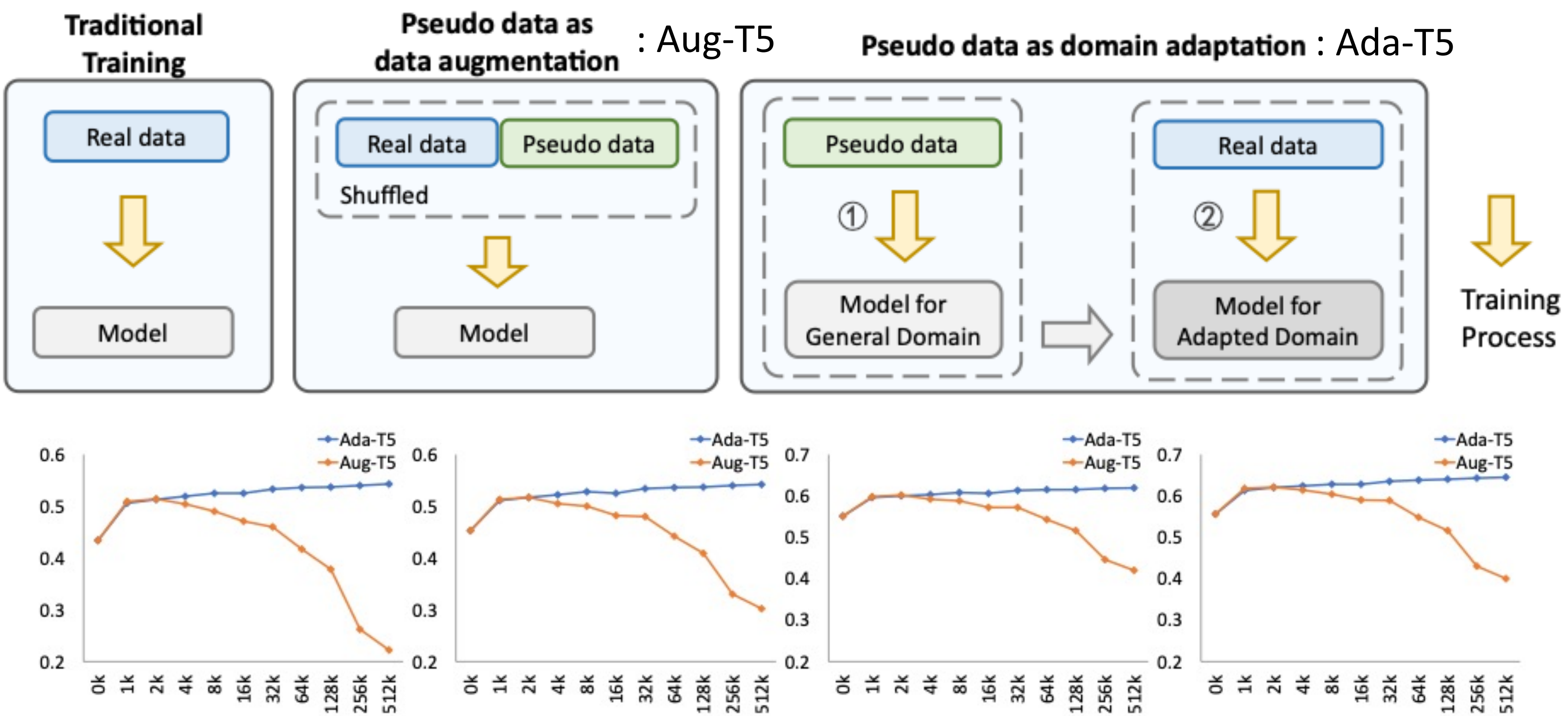
- Unifying Molecular and Textual Representation via **Multi-task Language Modeling** [Christofidellis et al., 2023]
  - Shows **reasonable performance** on **t2t** and **m2m** tasks (with a single model)
  - '-' denotes that the model cannot perform the corresponding task

Domain / Task	Size	mol2mol forward	mol2mol retrosynthesis	cross-domain text2mol	cross-domain mol2text	text2text paragraph-actions
T5 (fine-tuned) (Raffel et al., 2020)	small	0.603	0.245	0.499	0.501	**0.953**
T5 (fine-tuned) (Raffel et al., 2020)	base	0.629	-	0.762	0.511	-
RXN-forward (Toniato et al., 2021)	-	**0.685**	-	-	-	-
RXN-retrosynthesis (Toniato et al., 2021)	-	-	**0.733**	-	-	-
RXN-paragraph2actions (Vaucher et al., 2020)	-	-	-	-	-	0.850
MolT5 (Edwards et al., 2022)	small	-	-	0.755	0.519	-
MolT5 (Edwards et al., 2022)	base	-	-	0.769	0.540	-
Text+Chem T5 **(ours)**	small	0.412	0.249	0.815	0.553	0.929
Text+Chem T5 **(ours)**	base	0.459	0.478	0.750	0.580	0.935
Text+Chem T5-augm **(ours)**	small	0.413	0.405	0.815	0.560	0.926
Text+Chem T5-augm **(ours)**	base	0.594	0.372	**0.853**	**0.625**	0.943

- From Artificially Real to Real: **Leveraging Pseudo Data** from Large Language Models for Low-Resource for Molecule Discovery [Chen et al., 2024]
  - **Motivation: Text-molecule pairs** are **hard to obtain** due to **experimental costs**
  - Utilize **GPT** and **few-shot real samples** to generate **pseudo text-molecule pairs**



- From Artificially Real to Real: **Leveraging Pseudo Data** from Large Language Models for Low-Resource for Molecule Discovery [Chen et al., 2024]
  - (1) **Adapt** the model with **pseudo data**, and then (2) **train** with **real data**
  - **Simultaneously** using **pseudo data** and **real data** shows performance **degradation**

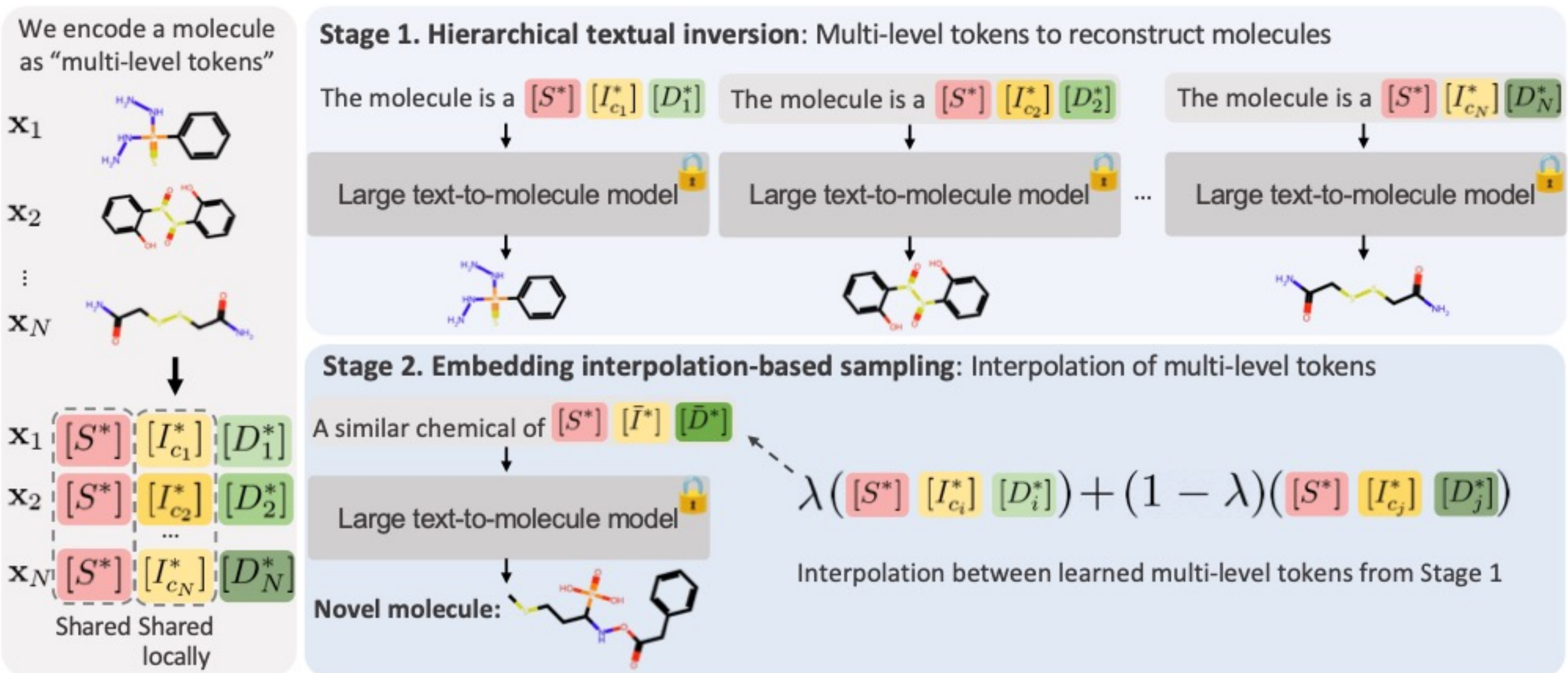


- From Artificially Real to Real: **Leveraging Pseudo Data** from Large Language Models for Low-Resource for Molecule Discovery [Chen et al., 2024]
  - Highly outperform **MolT5** due to the **high-quality pseudo samples** from GPT

Model	Parameters	ChEBI-20			PCdes			DrugBank-23		
		BL	RG	MET	BL	RG	MET	BL	RG	MET
T5	800M	$0.467^{\dagger*}$	$0.478^{\dagger*}$	$0.586^{\dagger*}$	$0.252^{\dagger*}$	$0.259^{\dagger*}$	$0.367^{\dagger*}$	$0.272^{\dagger*}$	$0.299^{\dagger*}$	$0.396^{\dagger*}$
MolT5	800M	$0.508^{\dagger}$	$0.510^{\dagger*}$	$0.614^{\dagger}$	$0.266^{\dagger}$	$0.272^{\dagger}$	$0.380^{\dagger*}$	$0.293^{\dagger}$	$0.317^{\dagger}$	$0.416^{\dagger}$
MolXPT	350M	$0.505^{\dagger*}$	$0.511^{\dagger*}$	$0.626^{\dagger}$	-	-	-	-	-	-
Text&Chem T5	250M	$0.542^{\dagger}$	$0.543^{\dagger}$	$0.648^{\dagger}$	$0.266^{\dagger}$	$0.274^{\dagger}$	$0.382^{\dagger}$	$0.280^{\dagger*}$	$0.312^{\dagger*}$	$0.413^{\dagger*}$
ChatGPT	-	$0.482^{\dagger*}$	$0.450^{\dagger*}$	$0.585^{\dagger*}$	$0.194^{\dagger*}$	$0.193^{\dagger*}$	$0.315^{\dagger*}$	$0.191^{\dagger*}$	$0.218^{\dagger*}$	$0.325^{\dagger*}$
Aug-T5	77M	0.515	0.517	0.621	0.270	0.275	0.385	0.297	0.322	0.421
Aug-$T5_{base}$	250M	0.516	0.520	0.620	0.268	0.272	0.383	0.294	0.316	0.416
Ada-T5	77M	0.553	0.552	0.652	**0.295**	0.295	0.406	0.310	0.337	0.435
Ada-$T5_{base}$	250M	**0.564**	**0.562**	**0.660**	**0.295**	**0.297**	**0.409**	**0.322**	**0.346**	**0.445**

Model	Parameters	ChEBI-20			PCdes			DrugBank-23		
		Acc	Val	MAC	Acc	Val	MAC	Acc	Val	MAC
T5	800M	$0.279^{\dagger*}$	$0.902^{\dagger*}$	$0.823^{\dagger*}$	$0.089^{\dagger}$	$0.910^{\dagger*}$	$0.698^{\dagger}$	$0.131^{\dagger*}$	$0.923^{\dagger*}$	$0.682^{\dagger}$
MolT5	800M	$0.311^{\dagger*}$	$0.905^{\dagger*}$	$0.834^{\dagger*}$	$0.097^{\dagger}$	$0.925^{\dagger}$	$0.695^{\dagger}$	$0.145^{\dagger*}$	$0.947^{\dagger}$	$0.686^{\dagger}$
MolXPT	350M	$0.215^{\dagger*}$	**0.983**	$0.859^{\dagger*}$	-	-	-	-	-	-
Text&Chem T5	250M	$0.322^{\dagger*}$	$0.943^{\dagger*}$	$0.901^{\dagger}$	$0.105^{\dagger}$	$0.849^{\dagger*}$	$0.697^{\dagger}$	$0.149^{\dagger}$	$0.898^{\dagger*}$	0.705
ChatGPT	-	$0.139^{\dagger*}$	$0.887^{\dagger*}$	$0.847^{\dagger*}$	$0.044^{\dagger*}$	$0.867^{\dagger*}$	$0.671^{\dagger*}$	$0.048^{\dagger*}$	$0.852^{\dagger*}$	$0.665^{\dagger*}$
Aug-T5	77M	0.305	0.907	0.877	0.070	0.892	0.700	0.141	0.911	0.685
Aug-$T5_{base}$	250M	0.386	0.955	0.884	0.098	0.927	0.696	0.158	0.952	0.681
Ada-T5	77M	0.449	0.967	0.905	0.135	0.945	0.725	0.170	0.955	0.696
Ada-$T5_{base}$	250M	**0.486**	0.974	**0.911**	**0.150**	**0.956**	**0.743**	**0.192**	**0.969**	**0.706**

- **Data-Efficient Molecular Generation** with Hierarchical Textual Inversion [Kim et al., 2024]
  - **Adaptation** of **molecular LLMs**, e.g., MolT5, for **data-efficient** **molecular generation**
  - We only have **few-shot molecules** in drug discovery; how to learn **their distribution**?

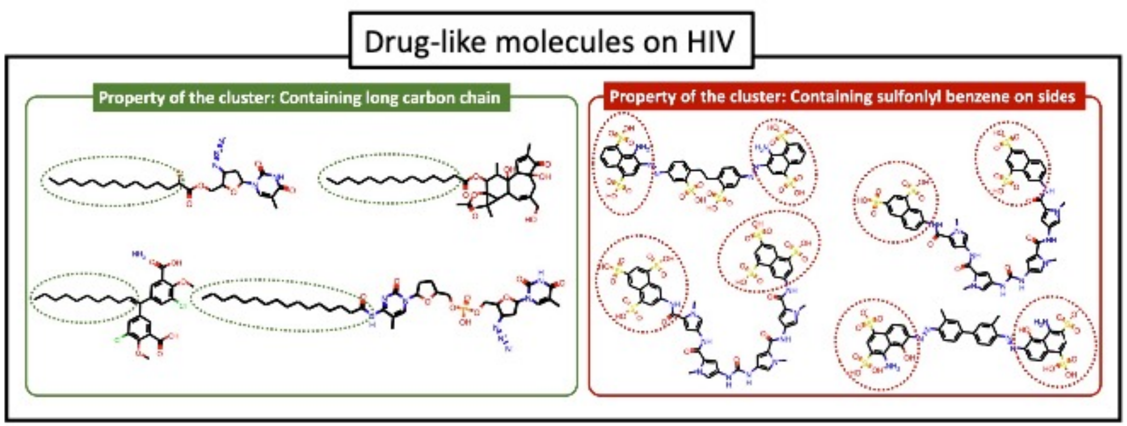


- **Data-Efficient Molecular Generation** with Hierarchical Textual Inversion [Kim et al., 2024]
  - **Few-shot distribution** learning methods in **other domains**, e.g., Textual Inversion [Gal et al., 2023], **does not work** for **molecules**
  - **Molecules** are more **structurally diverse**; naïve adoption **does not work**

Generation performance for **molecules**

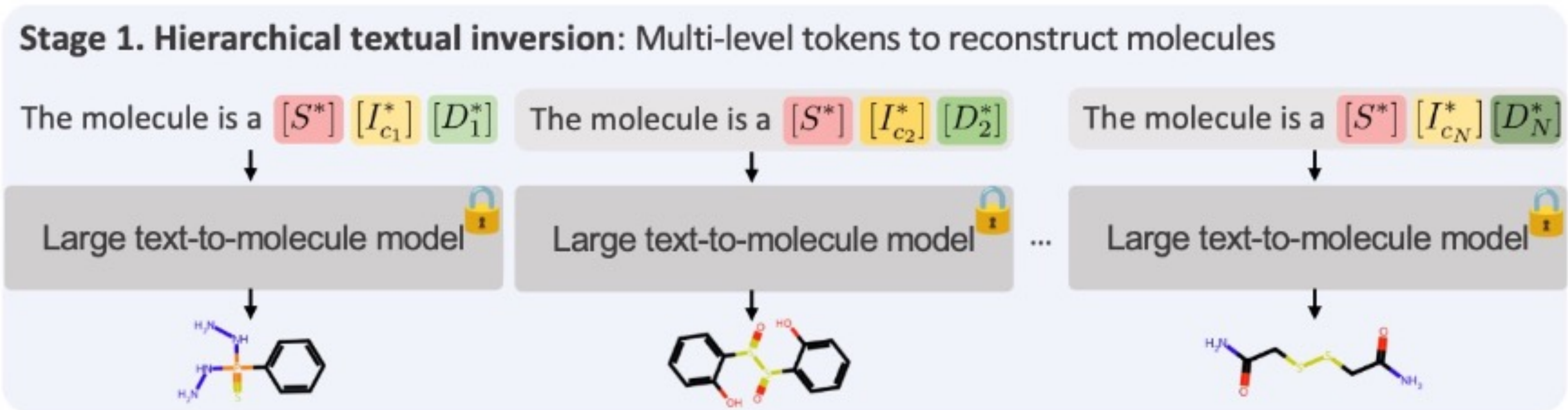
Inversion method	Validity (%)
Textual Inversion (Gal et al., 2022)	0.4
DreamBooth (Ruiz et al., 2022)	0.0

Textual Inversion [Gal et al., 2022]: **Visually similar images**



Molecules with a **common property**:
Not structurally similar

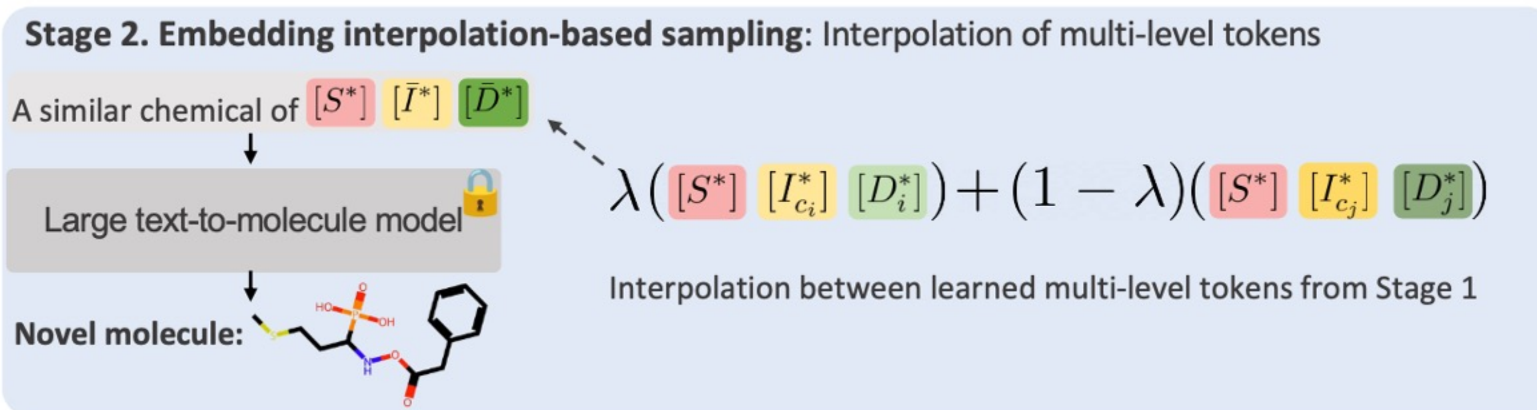
- **Data-Efficient Molecular Generation** with Hierarchical Textual Inversion [Kim et al., 2024]
  - Use **'hierarchical' tokens** unlike **Textual Inversion** [Gal et al., 2023] with a **single token**
  - **[S]**, **[I]**, and **[D]** learn **different hierarchical information** of **few-shot molecules**



$$\mathcal{L}(\theta; \mathbf{x}_n) := \min_{k\in[1,K]} \mathcal{L}_{\text{CE}}\Big(\texttt{softmax}\big(f(\text{“The molecule is a } [S^*][I_k^*][D_n^*]\text{”})\big),\ \texttt{SMILES}(\mathbf{x}_n)\Big)$$

[S]: A **single token** for **whole dataset**, learns **overall semantics** of target molecules
[I]: Tokens assigned to **k-th clsuter**, captures **cluster-wise semantics**
[D]: Tokens assigned to **n-th molecule**, captures **molecule-wise semantics**

- **Data-Efficient Molecular Generation** with Hierarchical Textual Inversion [Kim et al., 2024]
  - Use **'hierarchical' tokens** unlike **Textual Inversion** [Gal et al., 2023] with a **single token**
  - From **learned hierarchical token** embeddings, **sample** molecules by **interpolation**



$$(\bar{\mathbf{i}}, \bar{\mathbf{d}}) := \lambda(\mathbf{i}_{c_i}, \mathbf{d}_i) + (1-\lambda)(\mathbf{i}_{c_j}, \mathbf{d}_j),$$

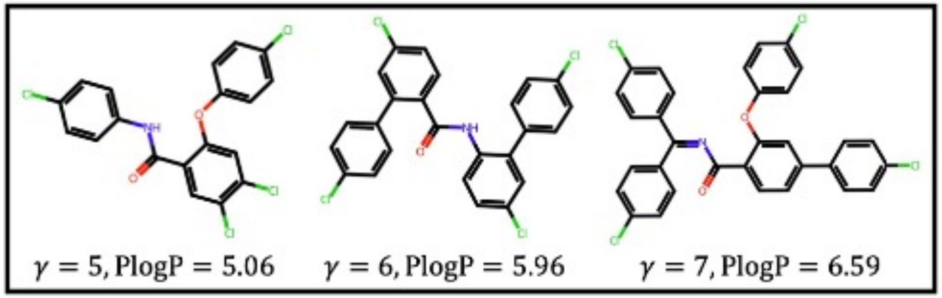
$$\mathbf{x} := f(\text{“A similar chemical of } [S^*][\bar{I}^*][\bar{D}^*]\text{”})$$

- **Data-Efficient Molecular Generation** with Hierarchical Textual Inversion [Kim et al., 2024]
  - Achieve **superior few-shot generation** results compared to previous methods
  - Due to the **preservation** of **hierarchical information** in training & generation

Dataset	Method	Class	Grammar	Active. ↑	FCD ↓	NSPDK ↓	Valid. ↑	Unique. ↑	Novelty ↑
HIV	GDSS (Jo et al., 2022)	Graph	✗	0.0	34.1	0.080	69.4	**100**	**100**
	DiGress (Vignac et al., 2023)	Graph	✗	0.0	26.2	0.067	17.8	**100**	**100**
	JT-VAE (Jin et al., 2018)	Fragment	✓	0.0	38.8	0.221	**100**	25.4	**100**
	PS-VAE (Kong et al., 2022)	Fragment	✓	3.7	21.8	0.053	**100**	91.4	**100**
	MiCaM (Geng et al., 2023)	Fragment	✓	3.4	20.4	0.037	**100**	81.6	**100**
	CRNN (Segler et al., 2018)	SMILES	✗	3.3	29.7	0.064	30.0	**100**	**100**
	STGG (Ahn et al., 2022)	SMILES	✓	1.6	20.2	0.033	**100**	95.8	**100**
	HI-Mol (Ours)	SMILES	✗	**11.4**	19.0	**0.019**	60.6	94.1	**100**
	HI-Mol (Ours)	SMILES	✓	**11.4**	**16.6**	**0.019**	**100**	95.6	**100**

Method	Class	Grammar	FCD ↓	NSPDK ↓	Valid. ↑	Unique. ↑	Novelty ↑
CG-VAE† (Liu et al., 2018)	Graph	✓	1.852	-	**100**	98.6	94.3
GraphAF (Shi et al., 2020)	Graph	✗	5.268	0.020	67	94.5	88.8
MoFlow (Zang & Wang, 2020)	Graph	✗	4.467	0.017	91.4	98.7	94.7
EDP-GNN (Niu et al., 2020)	Graph	✗	2.680	0.005	47.5	**99.3**	86.6
GraphDF (Luo et al., 2021)	Graph	✗	10.82	0.063	82.7	97.6	**98.1**
GraphEBM (Liu et al., 2021)	Graph	✗	6.143	0.030	8.22	97.8	97.0
GDSS (Jo et al., 2022)	Graph	✗	2.900	0.003	95.7	98.5	86.3
GSDM* (Luo et al., 2022)	Graph	✗	2.650	0.003	99.9	-	-
STGG† (Ahn et al., 2022)	SMILES	✓	0.585	-	**100**	95.6	69.8
HI-Mol (Ours; 2%)	SMILES	✓	0.430	**0.001**	**100**	76.1	75.6
HI-Mol (Ours; 10%)	SMILES	✓	**0.398**	**0.001**	**100**	88.3	73.2

- **Data-Efficient Molecular Generation** with Hierarchical Textual Inversion [Kim et al., 2024]
    - Applicable for **conditional generation**; learn an additional **condition embedding**



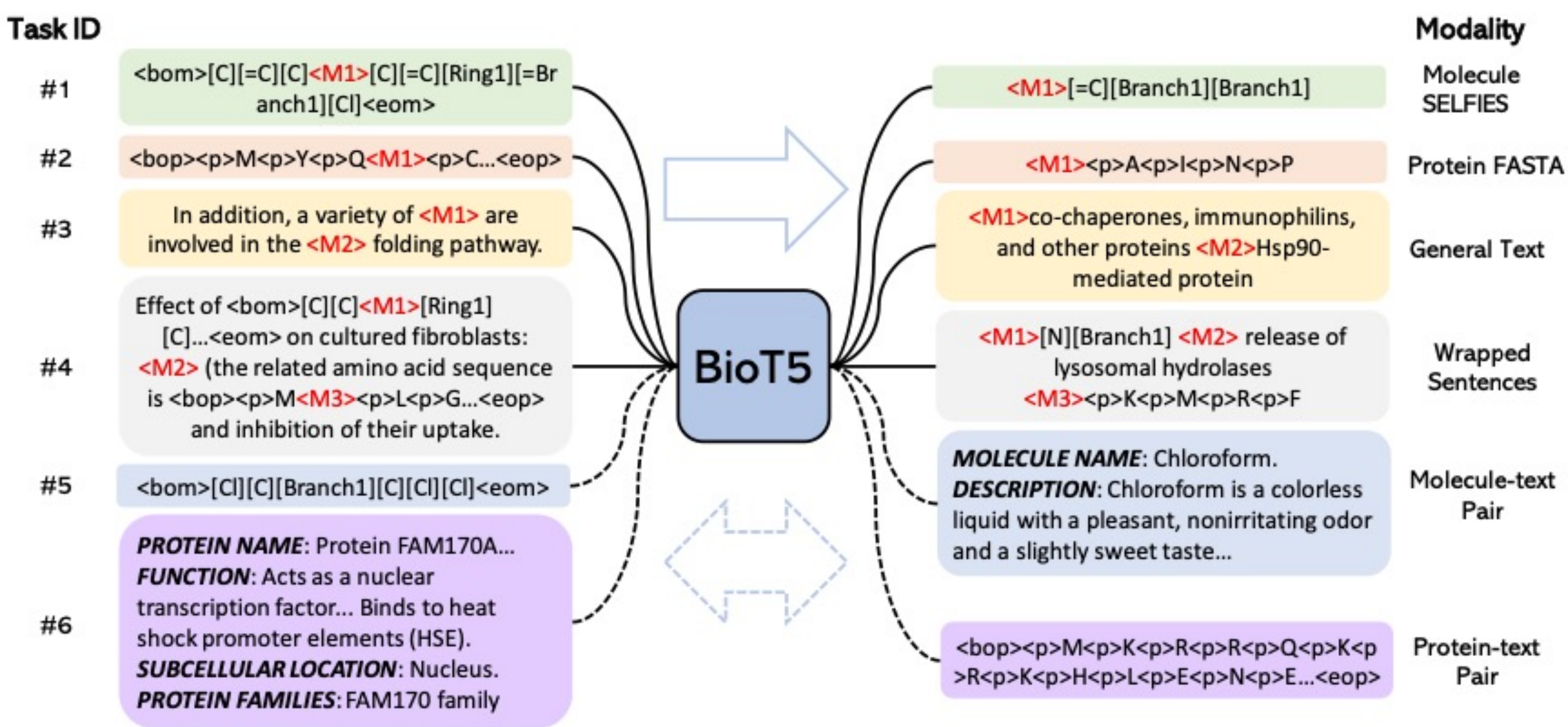
Method	Class	Offline	PlogP 1st	PlogP 2nd	PlogP 3rd
GVAE (Kusner et al., 2017)	SMILES	✓	2.94	2.89	2.80
SD-VAE (Dai et al., 2018)	Syntax Tree	✓	4.04	3.50	2.96
JT-VAE (Jin et al., 2018)	Fragment	✗	5.30	4.93	4.49
MHG-VAE (Kajino, 2019)	Fragment	✗	5.56	5.40	5.34
GraphAF (Shi et al., 2020)	Graph	✗	12.23	11.29	11.05
GraphDF (Luo et al., 2021)	Graph	✗	13.70	13.18	13.17
STGG (Ahn et al., 2022)	SMILES	✓	23.32	18.75	16.50
HI-Mol (Ours; 1%)	SMILES	✓	**24.67**	**21.72**	**20.73**

$$\mathcal{L}(\theta; \mathbf{x}_n) := \min_{k \in [1,K]} \mathcal{L}_{\text{CE}}\Big(\texttt{softmax}\big(f(\text{“The molecule is a } [S^*][I_k^*][D_n^*]\text{”})\big),\ \texttt{SMILES}(\mathbf{x}_n)\Big)$$

+ Condition embedding for PlogP value

LLMs for Chemistry & Biology: BioT5

- **BioT5:** Enriching Cross-modal Integration in Biology with Chemical Knowledge and Natural Language Associations [Pei et al., 2023]
    - An LLM for **chemistry & biology** with **'modality-specific'** token space



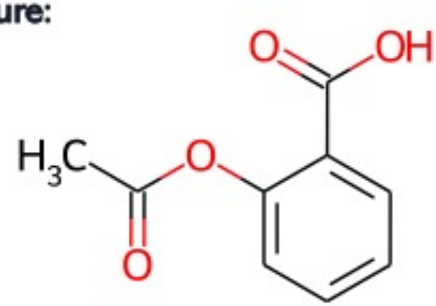
- **BioT5:** Enriching Cross-modal Integration in Biology with Chemical Knowledge and Natural Language Associations [Pei et al., 2023]
  - **Previous molecular LLMs** use the **T5 tokenizer** with the **SMILES representation**
  - **BioT5** regards a **SELFIES token** as a **single token**, which is more **structure-aware**
  - It also suggests to utilize **FASTA tokens** to represent **protein data** in LLMs

Name: Aspirin

MolT5 with T5 tokenizer: **SMILES:** CC(=O)OC1=CC=CC=C1C(=O)O

BioT5 tokenizer: **Structure-aware** **SELFIES:** [C][C][=Branch1][C][=O][O][C][=C][C][=C][C][=C][Ring1][=Branch1][C][=Branch1][C][=O][O]

Structure:

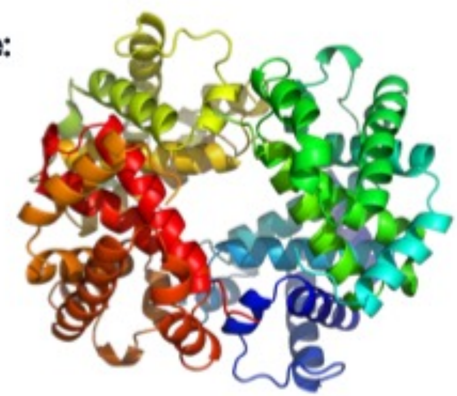


Name: Hemoglobin subunit beta

Gene: HBB

FASTA: MVHLTPEEKSAVTALWGKVN…

Structure:



- **BioT5:** Enriching Cross-modal Integration in Biology with Chemical Knowledge and Natural Language Associations [Pei et al., 2023]
  - By using **more sophisticated token space**, achieves **state-of-the-art results**

Model	#Params.	BLEU-2	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	Text2Mol
RNN	56M	0.251	0.176	0.450	0.278	0.394	0.363	0.426
Transformer	76M	0.061	0.027	0.204	0.087	0.186	0.114	0.057
T5-small	77M	0.501	0.415	0.602	0.446	0.545	0.532	0.526
T5-base	248M	0.511	0.423	0.607	0.451	0.550	0.539	0.523
T5-large	783M	0.558	0.467	0.630	0.478	0.569	0.586	0.563
MolT5-small	77M	0.519	0.436	0.620	0.469	0.563	0.551	0.540
MolT5-base	248M	0.540	0.457	0.634	0.485	0.578	0.569	0.547
MolT5-large	783M	0.594	0.508	0.654	0.510	0.594	0.614	0.582
GPT-3.5-turbo (zero-shot)	>175B	0.103	0.050	0.261	0.088	0.204	0.161	0.352
GPT-3.5-turbo (10-shot MolReGPT)	>175B	0.565	0.482	0.623	0.450	0.543	0.585	0.560
MolXPT	350M	0.594	0.505	0.660	0.511	0.597	0.626	0.594
BioT5	252M	**0.635**	**0.556**	**0.692**	**0.559**	**0.633**	**0.656**	**0.603**

Model	#Params.	BLEU↑	Exact↑	Levenshtein↓	MACCS FTS↑	RDK FTS↑	Morgan FTS↑	FCD↓	Text2Mol↑	Validity↑
RNN	56M	0.652	0.005	38.09	0.591	0.400	0.362	4.55	0.409	0.542
Transformer	76M	0.499	0.000	57.66	0.480	0.320	0.217	11.32	0.277	0.906
T5-small	77M	0.741	0.064	27.703	0.704	0.578	0.525	2.89	0.479	0.608
T5-base	248M	0.762	0.069	24.950	0.731	0.605	0.545	2.48	0.499	0.660
T5-large	783M	0.854	0.279	16.721	0.823	0.731	0.670	1.22	0.552	0.902
MolT5-small	77M	0.755	0.079	25.988	0.703	0.568	0.517	2.49	0.482	0.721
MolT5-base	248M	0.769	0.081	24.458	0.721	0.588	0.529	2.18	0.496	0.772
MolT5-large	783M	0.854	0.311	16.071	0.834	0.746	0.684	1.20	0.554	0.905
GPT-3.5-turbo (zero-shot)	>175B	0.489	0.019	52.13	0.705	0.462	0.367	2.05	0.479	0.802
GPT-3.5-turbo (10-shot MolReGPT)	>175B	0.790	0.139	24.91	0.847	0.708	0.624	0.57	0.571	0.887
MolXPT	350M	-	0.215	-	0.859	0.757	0.667	0.45	**0.578**	0.983
BioT5	252M	**0.867**	**0.413**	**15.097**	**0.886**	**0.801**	**0.734**	**0.43**	0.576	**1.000**

- **BioT5:** Enriching Cross-modal Integration in Biology with Chemical Knowledge and Natural Language Associations [Pei et al., 2023]
  - In addition, shows **superior performance** on **biological applications**

Method	BioSNAP			Human		BindingDB		
	AUROC	AUPRC	Accuracy	AUROC	AUPRC	AUROC	AUPRC	Accuracy
SVM	0.862±0.007	0.864±0.004	0.777±0.011	0.940±0.006	0.920±0.009	0.939±0.001	0.928±0.002	0.825±0.004
RF	0.860±0.005	0.886±0.005	0.804±0.005	0.952±0.011	0.953±0.010	0.942±0.011	0.921±0.016	0.880±0.012
DeepConv-DTI	0.886±0.006	0.890±0.006	0.805±0.009	0.980±0.002	0.981±0.002	0.945±0.002	0.925±0.005	0.882±0.007
GraphDTA	0.887±0.008	0.890±0.007	0.800±0.007	0.981±0.001	0.982±0.002	0.951±0.002	0.934±0.002	0.888±0.005
MolTrans	0.895±0.004	0.897±0.005	0.825±0.010	0.980±0.002	0.978±0.003	0.952±0.002	0.936±0.001	0.887±0.006
DrugBAN	0.903±0.005	0.902±0.004	0.834±0.008	0.982±0.002	0.980±0.003	0.960±0.001	0.948±0.002	0.904±0.004
BioT5	**0.937±0.001**	**0.937±0.004**	**0.874±0.001**	**0.989±0.001**	**0.985±0.002**	**0.963±0.001**	**0.952±0.001**	**0.907±0.003**

Model	#Params.	Yeast	Human
DDE	205.3K	55.83 ± 3.13	62.77 ± 2.30
Moran	123.4K	53.00 ± 0.50	54.67 ± 4.43
LSTM	26.7M	53.62 ± 2.72	63.75 ± 5.12
Transformer	21.3M	54.12 ± 1.27	59.58 ± 2.09
CNN	5.4M	55.07 ± 0.02	62.60 ± 1.67
ResNet	11.0M	48.91 ± 1.78	68.61 ± 3.78
ProtBert	419.9M	63.72 ± 2.80	77.32 ± 1.10
ProtBert*	419.9M	53.87 ± 0.38	83.61 ± 1.34
ESM-1b	652.4M	57.00 ± 6.38	78.17 ± 2.91
ESM-1b*	652.4M	**66.07 ± 0.58**	**88.06 ± 0.24**
BioT5	252.1M	64.89 ± 0.43	86.22 ± 0.53

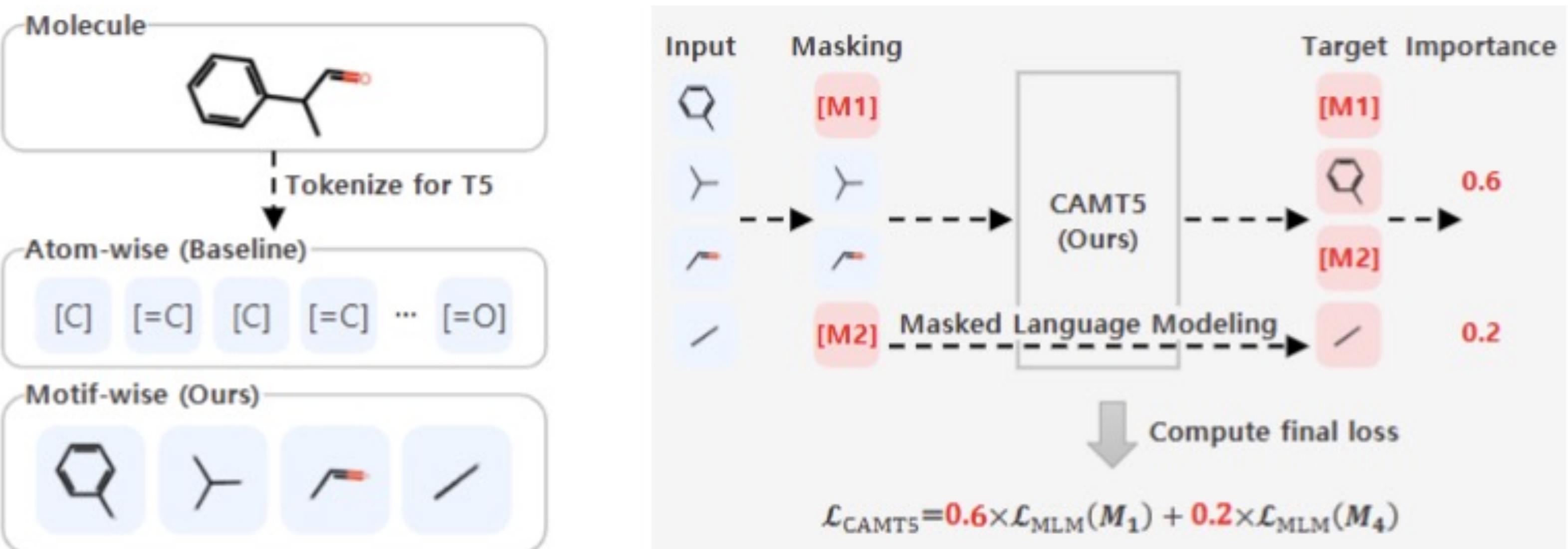
Model	#Params.	Solubility	Localization
DDE	205.3K	59.77 ± 1.21	77.43 ± 0.42
Moran	123.4K	57.73 ± 1.33	55.63 ± 0.85
LSTM	26.7M	70.18 ± 0.63	88.11 ± 0.14
Transformer	21.3M	70.12 ± 0.31	75.74 ± 0.74
CNN	5.4M	64.43 ± 0.25	82.67 ± 0.32
ResNet	11.0M	67.33 ± 1.46	78.99 ± 4.41
ProtBert	419.9M	68.15 ± 0.92	91.32 ± 0.89
ProtBert*	419.9M	59.17 ± 0.21	81.54 ± 0.09
ESM-1b	652.4M	70.23 ± 0.75	**92.40 ± 0.35**
ESM-1b*	652.4M	67.02 ± 0.40	91.61 ± 0.10
BioT5	252.1M	**74.65 ± 0.49**	91.69 ± 0.05

- **CAMT5:** Context-Aware Molecular T5 [Kim et al., 2024]
  - Goal: Developing a text-to-molecule generative model.
  - Convention: Utilizing atom-wise tokenization based on SMILES or SELFIES.
  - **MolT5**: Based on SMILES, which does not ensure the validity of the generated molecules.
  - **BioT5**: Based on SEFLIES, where the same token represents various molecular semantics.

- However, atom-wise tokenization does not reflect chemical functionality.
  - Chemical functionalities are encoded through motifs, i.e., functional groups.
  - Make the molecule tokens based on functional groups!

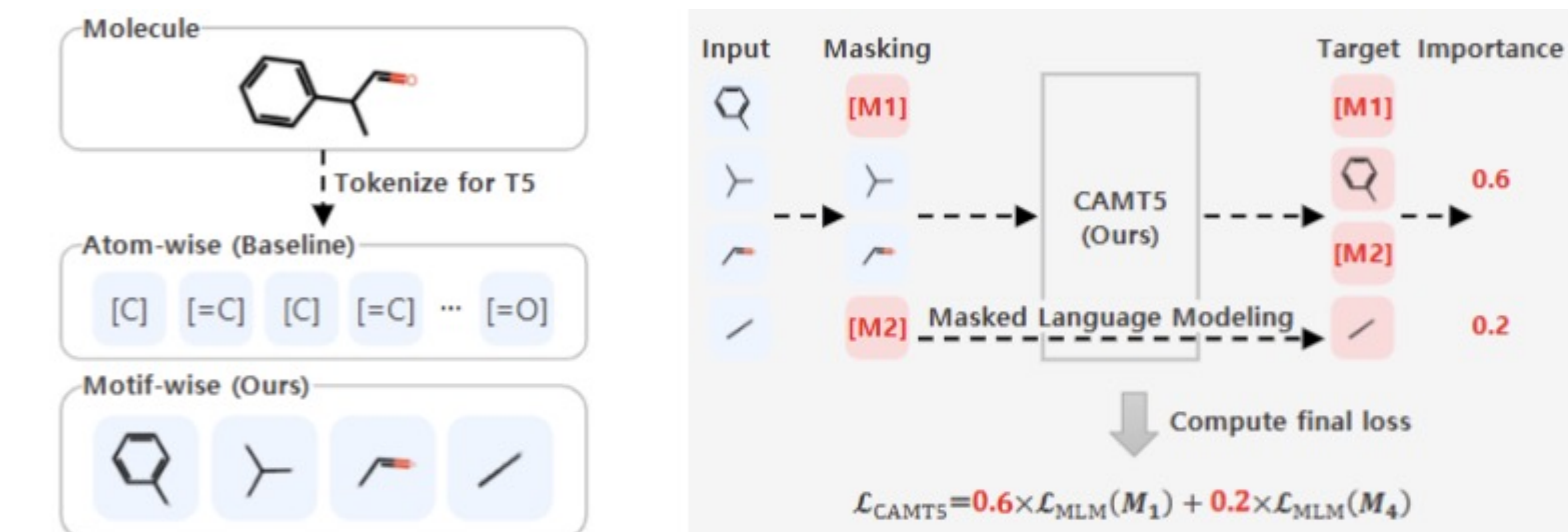
Method	Token	Validity	Non-degeneracy
MolT5	Atom	✗	✓
BioT5	Atom	✓	✗
CAMT5 (Ours)	**Motif**	✓	✓

- How can we embed **functional groups** into the **token space** of the text-to-molecule model?
  - Construct "Context-Tree" with pre-defined motifs!
  - One can linearize the motif-level tokens via a tree-search algorithm.
  - A sequence of motif-level tokens always represents a valid molecule.
  - There is a one-to-one correspondence between a motif and a motif-level token.
- Additionally, CAMT5 proposes importance-based pre-training.
  - Prioritizing key motifs during pre-training.



(a) Context-aware molecule tokenization (b) Importance-based pre-training

- How can we embed **functional groups** into the **token space** of the text-to-molecule model?
  - Construct "Context-Tree" with pre-defined motifs!
  - One can linearize the motif-level tokens via a tree-search algorithm.
  - A sequence of motif-level tokens always represents a valid molecule.
  - There is a one-to-one correspondence between a motif and a motif-level token.
- Additionally, CAMT5 proposes importance-based pre-training.
  - Prioritizing key motifs during pre-training.



(a) Context-aware molecule tokenization (b) Importance-based pre-training

- **Experiment**: Context-aware tokenization is beneficial for molecular language models.

Model	#Params.	Representation	Train Tokens	Exact ↑	MACCS ↑	RDK ↑	Morgan ↑	Valid. ↑
RNN	56M	SMILES	-	0.005	0.591	0.400	0.362	0.542
Transformer	76M	SMILES	-	0.000	0.480	0.320	0.217	0.906
$T5_{small}$	77M	SMILES	-	0.064	0.704	0.578	0.525	0.608
$T5_{base}$	248M	SMILES	-	0.069	0.731	0.605	0.545	0.660
$T5_{large}$	783M	SMILES	-	0.279	0.823	0.731	0.670	0.902
$MolT5_{small}$	77M	SMILES	66B	0.079	0.703	0.568	0.517	0.721
$MolT5_{base}$	248M	SMILES	66B	0.081	0.721	0.588	0.529	0.772
$MolT5_{large}$	783M	SMILES	66B	0.311	0.834	0.746	0.684	0.905
GPT-3.5-turbo	>175B	SMILES	-	0.019	0.705	0.462	0.367	0.802
MolReGPT	>175B	SMILES	-	0.139	0.847	0.708	0.624	0.887
MolXPT	350M	SMILES	1.8B	0.215	0.859	0.757	0.667	0.983
$BioT5^{*}_{base}$	252M	SELFIES	69B	0.413	**0.886**	0.801	0.734	**1.000**
$MolT5^{\dagger}_{base}$	248M	SMILES	1.6B	0.326	0.847	0.797	0.720	0.950
$BioT5^{\dagger}_{base}$	252M	SELFIES	1.6B	0.344	0.842	0.773	0.664	**1.000**
$CAMT5_{small}$ (Ours)	103M	**Motif (Ours)**	1.6B	0.391	0.874	0.827	0.727	**1.000**
$CAMT5_{base}$ (Ours)	286M	**Motif (Ours)**	1.6B	0.422	0.882	0.834	0.742	**1.000**
$CAMT5_{large}$ (Ours)	836M	**Motif (Ours)**	1.6B	**0.430**	0.885	**0.840**	**0.749**	**1.000**

Table of Contents

Why is mathematics hard for LLMs?

- Requires both multi-step task decomposition and accurate calculation
- A single mistake can lead to entirely wrong result
- LLMs are designed to be non-deterministic
- Mathematics require precise, strict rule-based reasoning

Are LLMs still bad at math?

- **No**
- Various training, inference strategies made LLMs excel at math

Google DeepMind *2025-3-4*

Gold-medalist Performance in Solving Olympiad Geometry with AlphaGeometry2

Yuri Chervonyi[*,1,◇], **Trieu H. Trinh**[*,1,◇], **Miroslav Olšák**[†,1,2], **Xiaomeng Yang**[†,1], **Hoang Nguyen**[1,3], **Marcelo Menegali**[1], **Junehyuk Jung**[1,4], **Vikas Verma**[1], **Quoc V. Le**[1] **and Thang Luong**[1,◇]
[1]Google DeepMind, [2]University of Cambridge, [3]Georgia Institute of Technology, [4]Brown University
This work was conducted entirely at Google DeepMind by all authors.

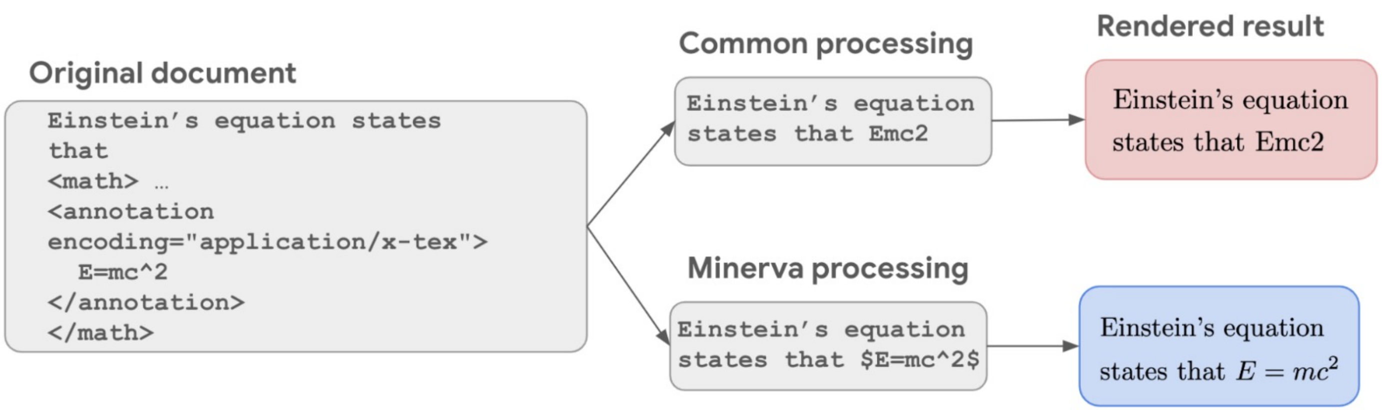
Minerva [Lewkowycz et al., 2022]

Further training pretrained language model(PaLM) on mathematical dataset

- **Dataset:** Collect and process data maintaining mathematical content

Data source	Proportion of data	Tokens	Present during pretraining
Math Web Pages	47.5%	17.5B	No
arXiv	47.5%	21.0B	No
General Natural Language Data	5%	>100B	Yes

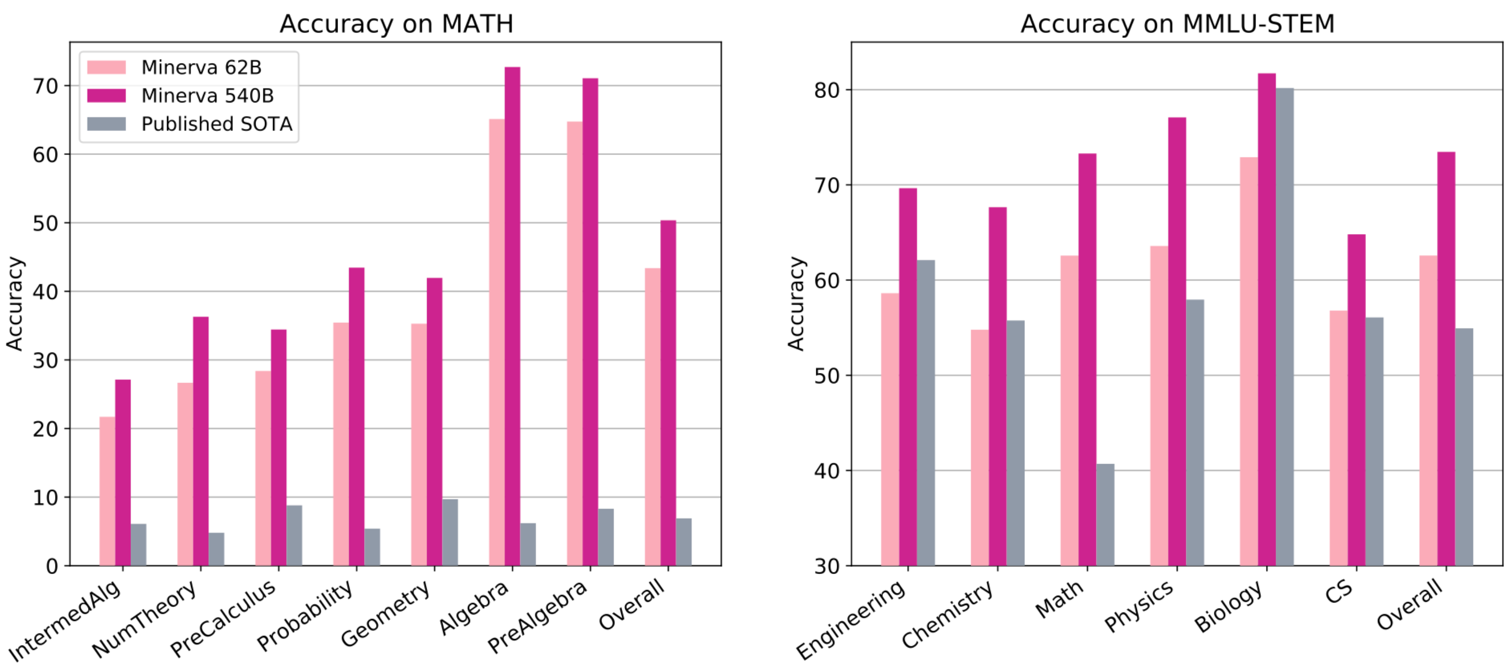
- **Processing:** Extract mathematical content in **LaTeX** or **ASCII-math** format
  - Maintain symbols essential to mathematical expressions



Minerva [Lewkowycz et al., 2022]

Minerva outperforms the state-of-the-art on math and science benchmarks

- **MATH:** Middle school and high school mathematics problems written in LaTeX
- **MMLU-STEM:** Subset of the MMLU dataset focused on science, technology, engineering, and mathematics (STEM)



Minerva [Lewkowycz et al., 2022]

Inference-Time Techniques

- **Few-shot prompting + CoT + Majority Voting** (*maj@k*) [Wang et al., 2022]
  - *maj@k*: Sampling *k* predictions and selecting the most common answer
- Significantly improves performance over greedy decoding

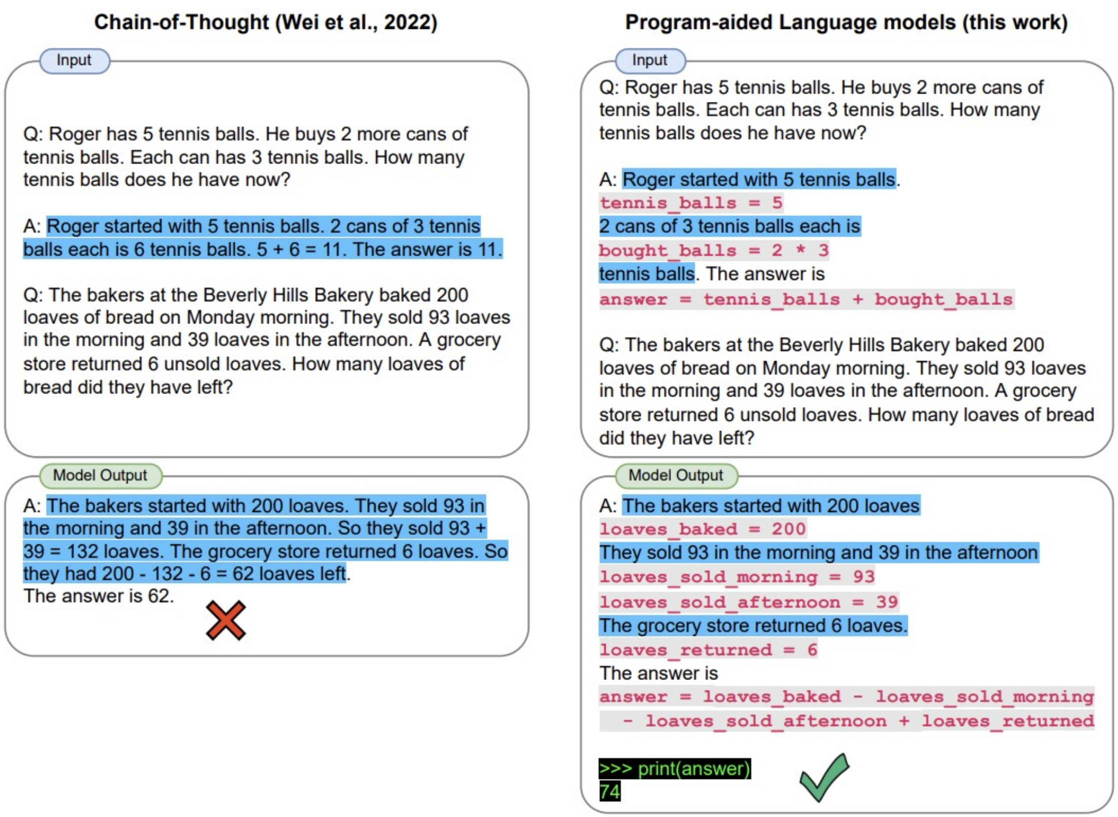
	MATH	OCWCourses	GSM8k	MMLU-STEM
PaLM 8B	1.5%	1.5%	4.1%	22.0%
Minerva 8B	14.1%	7.7%	16.2%	35.6%
Minerva 8B, `maj1@k`	25.4%	12.5%	28.4%	43.4%
PaLM 62B	4.4%	5.9%	33.0%	39.1%
Minerva 62B	27.6%	12.9%	52.4%	53.9%
Minerva 62B, `maj1@k`	43.4%	23.5%	68.5%	63.5%
PaLM 540B	8.8%	7.1%	56.5%	58.7%
Minerva 540B	33.6%	17.6%	58.8%	63.9%
Minerva 540B, `maj1@k`	**50.3%**	**30.8%**	**78.5%**	**75.0%**
OpenAI davinci-002	19.1%	14.8%	-	-
Published SOTA	6.9%[a]	-	74.4%[b]	54.9%[c]

PAL: Program-aided Language Models [Gao et al., 2023]

Motivation: LLMs often generate reasoning steps correctly, but slips at calculation

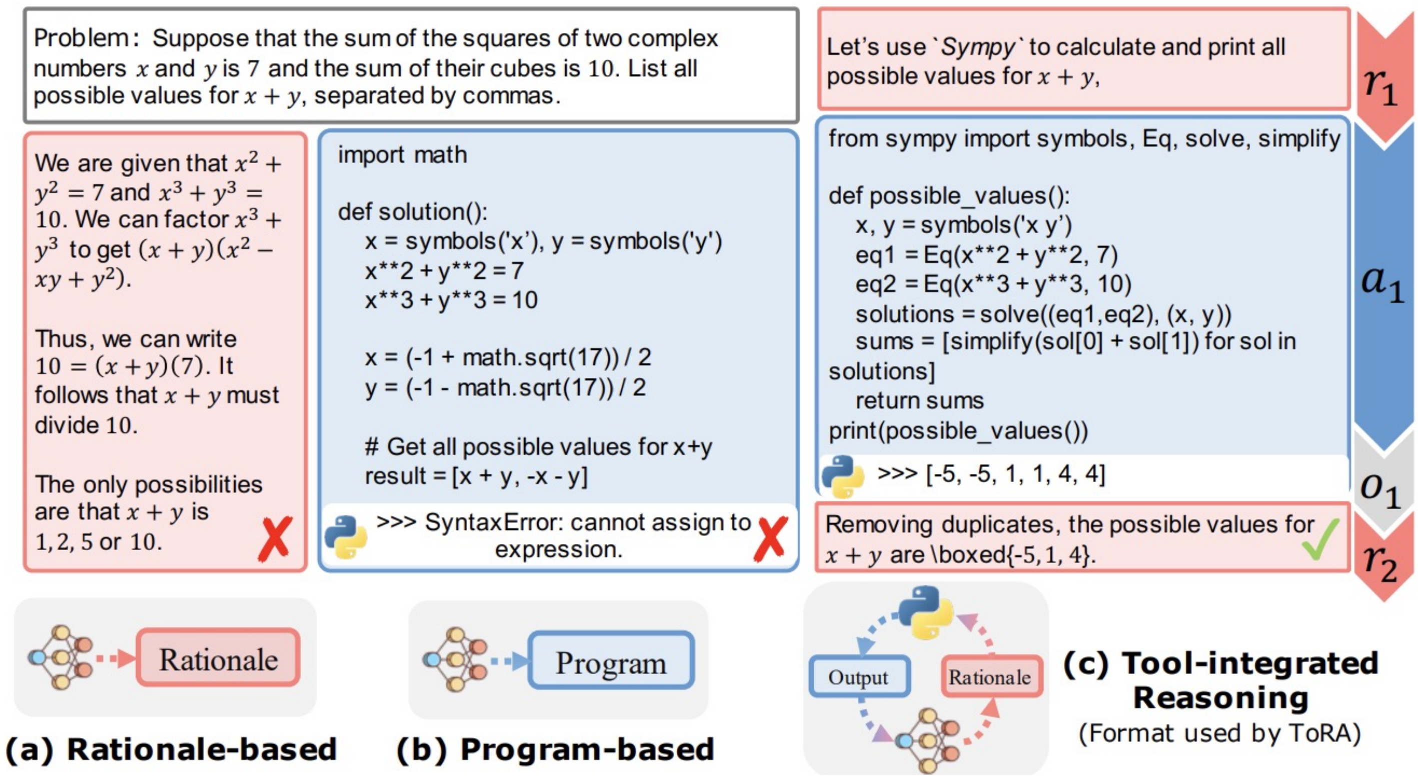
Idea: Running the reasoning steps with a Python interpreter

- Leads to multiple variants leveraging external solvers



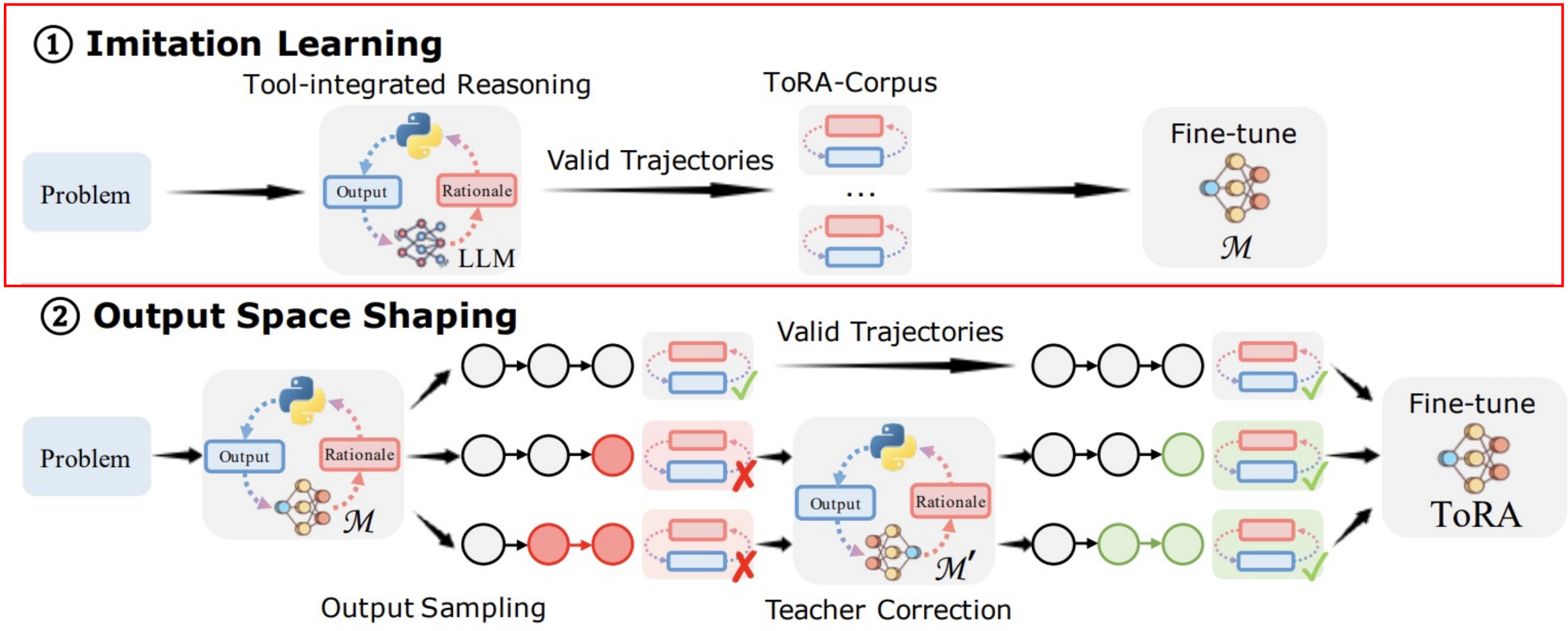
ToRA (Tool Integrated **R**easoning **A**gents) [Gou et al., 2024]

- Interactive tool-use trajectories
  - Repeat *natural language guidance* and *program execution* to reach an answer
  - Benefit from analytical power of language and the computational efficiency of tools



ToRA Pipeline

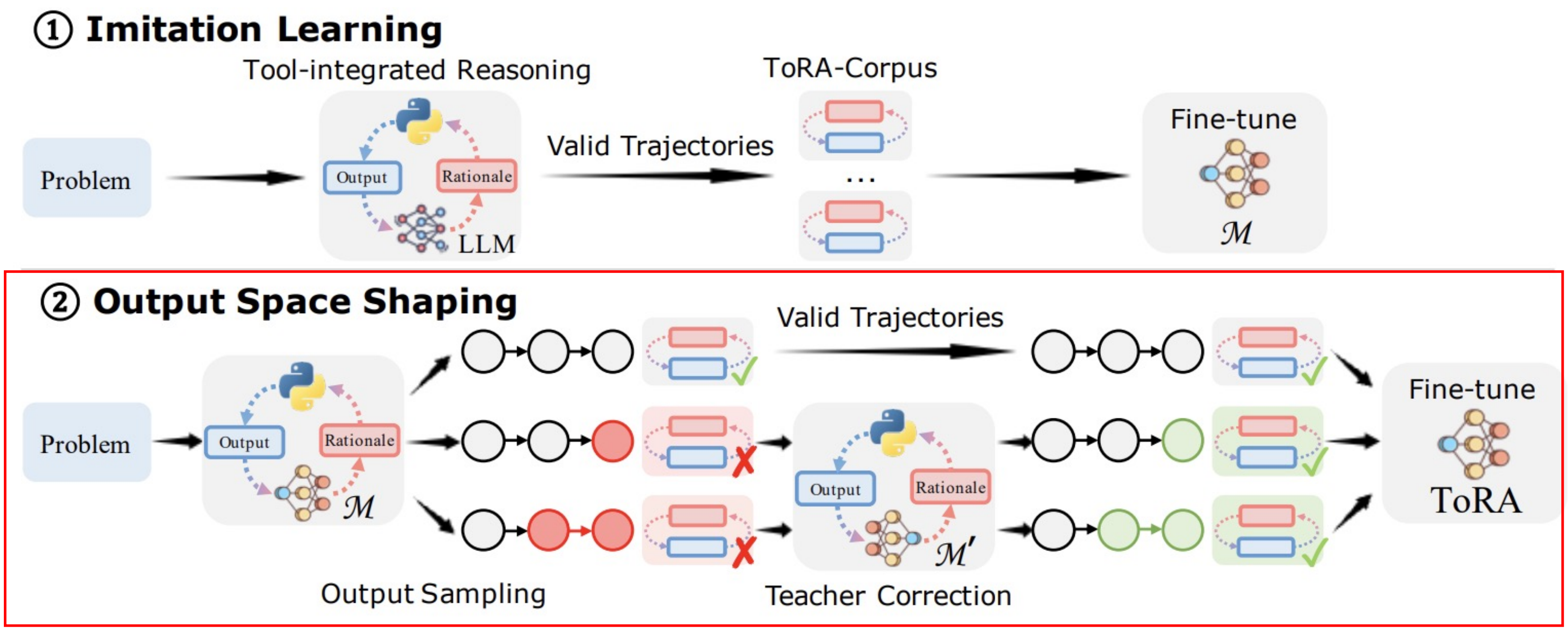
1. Imitation Learning
   - Collect high-quality trajectories from GPT-4, solving diverse math problems
     - Dataset: GSM8k(grade school math word problems), MATH(high school math)
   - Sample only valid trajectories leading to correct answers



ToRA Pipeline

2. Output Space Shaping

- Sample diverse trajectories from fine-tuned model
- Correct invalid trajectories with teacher model (Code expert open model)
- Fine-tune model on corrected valid trajectories + original ToRA-Corpus

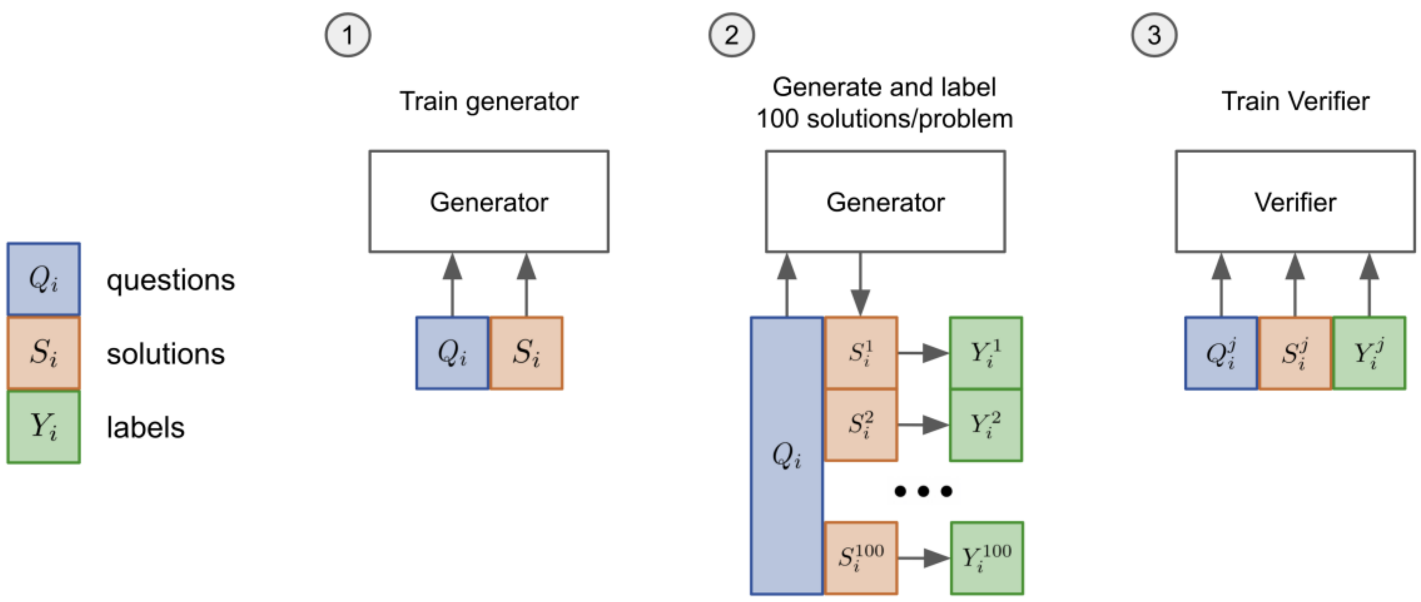


Outcome-supervised Reward Model (ORM) [Cobbe et al., 2021]

Train a verifier model to judge the correctness of solutions, respect to GT answer

1) Finetune generator(problem solving model) on training set
2) Sample 100 completions from generator, label each solution as correct/incorrect
3) Train verifier model to predict ‘solution correctness probability’

- During inference, select the generator’s solution with the highest verifier score

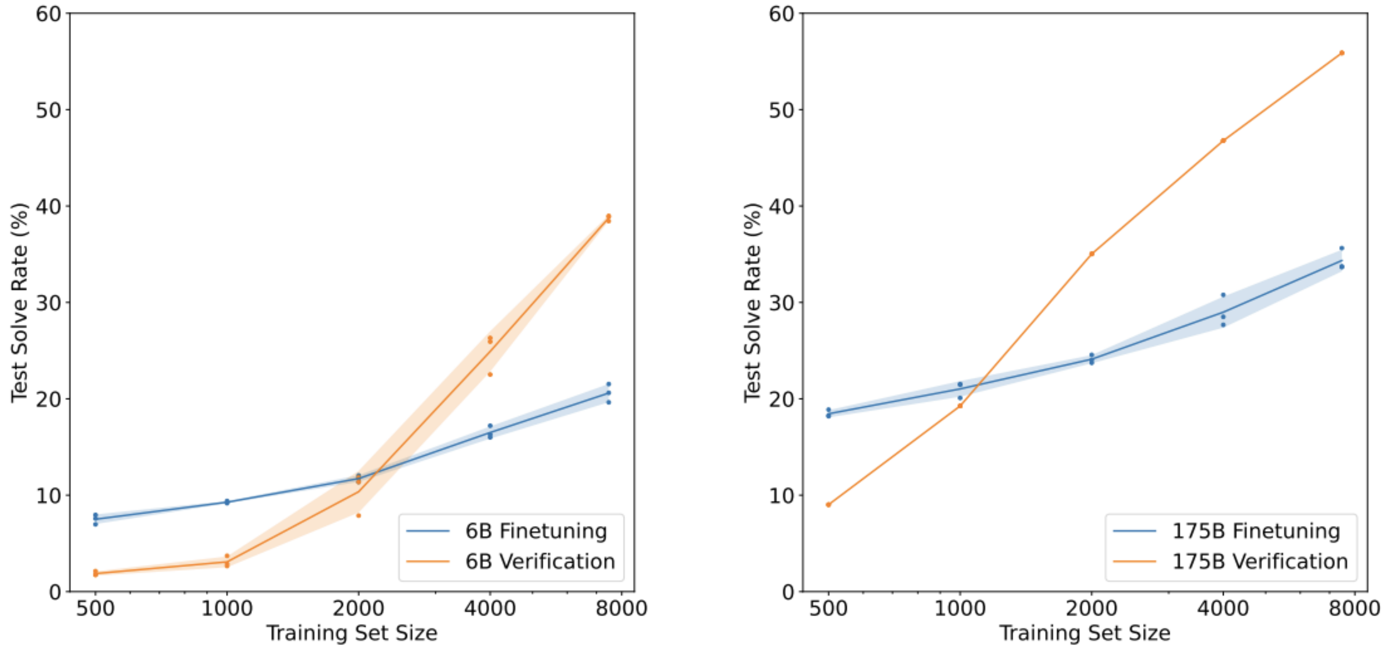


Outcome-supervised Reward Model (ORM) [Cobbe et al., 2021]

Comparison between finetuning and verification

- Verification boosts performance if the dataset is large enough
- Verifiers can overfit memorizing final answers when dataset is too small
- In full training set, 6B verification outperforms 175B finetuning

* Train dataset: GSM8k, math word problems using arithmetic operations (+ − × ÷)



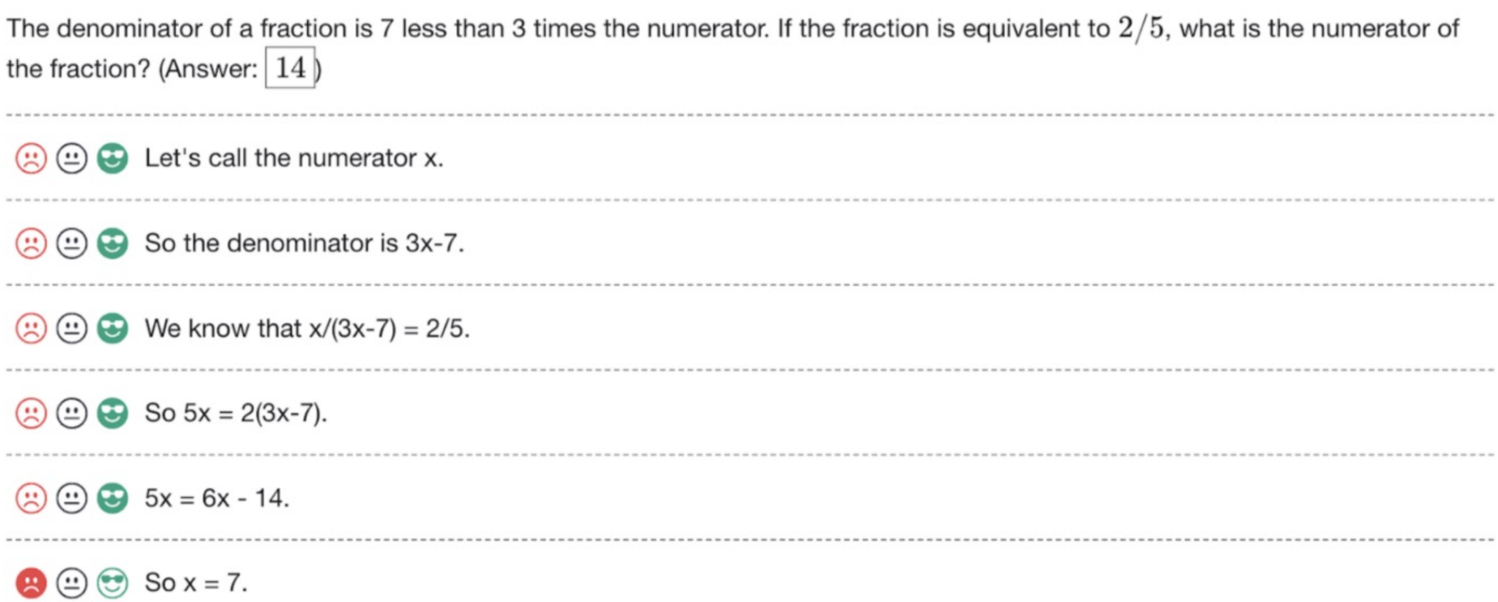
Process-supervised Reward Model (PRM) [Lightman et al., 2023]

Motivation: ORM can misgrade false-positive solutions

- Incorrect solutions still can reach to correct answers

Idea: Provide feedback for each intermediate reasoning step

- Human data-labelers to assign each step into *positive, negative, neutral*
- Construct PRM800k(open), step-level human feedback dataset



Feedback interface used for step-wise reward collection

Process-supervised Reward Model (PRM) [Lightman et al., 2023]

Following ORM, train a verifier model and use at inference stage

- At training, predict the correctness of each step after the last step token

  Green: high PRM score, Red: low PRM score

- During inference, select the generator's solution with the highest verifier score

Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x)\cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.

I notice that the given polynomial has even degree and only even powers of x, so I can try to make a substitution to simplify it.

Let $y = x^4$, then the polynomial becomes $y^2 + 3y - 4$, which is a quadratic equation.

I can factor this quadratic equation as $(y+4)(y-1)$, so the original polynomial is $(x^4+4)(x^4-1)$.

Now I need to factor each of these factors further, if possible, over the integers.

For the first factor, $x^4 + 4$, I recall the Sophie Germain identity, which says that $a^4 + 4b^4 = (a^2 + 2b^2 + 2ab)(a^2 + 2b^2 - 2ab)$.

Applying this identity with $a = x$ and $b = 1$, I get $x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2)$.

For the second factor, $x^4 - 1$, I can use the difference of squares formula, which says that $a^2 - b^2 = (a+b)(a-b)$.

Applying this formula with $a = x^2$ and $b = 1$, I get $x^4 - 1 = (x^2+1)(x^2-1)$.

Then I can apply the difference of squares formula again with $a = x$ and $b = 1$, to get $x^2 - 1 = (x+1)(x-1)$.

So the original polynomial is $(x^4+4)(x^4-1) = (x^2+2x+2)(x^2-2x+2)(x^2+1)(x+1)(x-1)$.

These are all the irreducible factors over the integers, so $p_1(x) = x^2+2x+2$, $p_2(x) = x^2-2x+2$, $p_3(x) = x^2+1$, $p_4(x) = x+1$, and $p_5(x) = x-1$.

To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.

I get $p_1(1) = 1^2 + 2\cdot 1 + 2 = 5$, $p_2(1) = 1^2 - 2\cdot 1 + 2 = 1$, $p_3(1) = 1^2 + 1 = 2$, $p_4(1) = 1 + 1 = 2$, and $p_5(1) = 1 - 1 = 0$.

So the final answer is $5 + 1 + 2 + 2 + 0 = 10$.

Answer: 10

Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x)\cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.

This problem is about factoring a polynomial over the integers, which can be a tricky task.

One possible strategy is to look for rational roots of the polynomial, using the rational root theorem.

However, that would involve checking a lot of possible factors of -4, and there is no guarantee that the polynomial has any rational roots at all.

Another possible strategy is to use some algebraic tricks to simplify the polynomial or reduce its degree.

I notice that the polynomial has only even powers of x, so I can try to substitute $y = x^2$ and see what happens.

Then the polynomial becomes $y^4 + 3y^2 - 4$, which is a quadratic equation in y^2.

I can try to factor this quadratic equation by looking for two numbers that multiply to -4 and add to 3.

Those numbers are 4 and -1, so I can write $y^4 + 3y^2 - 4 = (y^2+4)(y^2-1)$.

Now I can substitute back $x^2 = y$ and get $x^8 + 3x^4 - 4 = (x^4+4)(x^4-1)$.

This looks promising, because I have reduced the degree of the factors from 8 to 4.

I can try to factor each of these factors further, using the difference of squares formula.

I get $x^8 + 3x^4 - 4 = (x^2+2)(x^2-2)(x^2+1)(x^2-1)$.

I can apply the difference of squares formula again to the last factor and get $x^8 + 3x^4 - 4 = (x^2+2)(x^2-2)(x^2+1)(x+1)(x-1)$.

Now I have factored the polynomial completely into monic linear and quadratic factors with integer coefficients.

These are the $p_i(x)$'s that the problem is asking for.

To find the sum of their values at $x = 1$, I just need to plug in $x = 1$ into each factor and add them up.

I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (1^2+2)(1^2-2)(1^2+1)(1+1)(1-1)$.

Simplifying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = (3)(-1)(2)(2)(0)$.

Multiplying, I get $p_1(1) + p_2(1) + \cdots + p_k(1) = 0$.

Answer: 0

Process-supervised Reward Model (PRM) [Lightman et al., 2023]

PRM scoring strategy experiment

- To select among multiple solutions, single score for each solution is required
  - Score of the entire solution (2 strategies)
    - Product of the correctness probabilities for each step in the solution
    - Minimum correctness probability of all steps included in the solution
  - How to consider *neutral* feedbacks
    - Feedbacks were assigned as *positive*, *negative*, *or neutral*
    - To consider *neutral* as *positive or negative*

	product	minimum
neutral = positive	78.2%	77.6%
neutral = negative	77.4%	77.8%

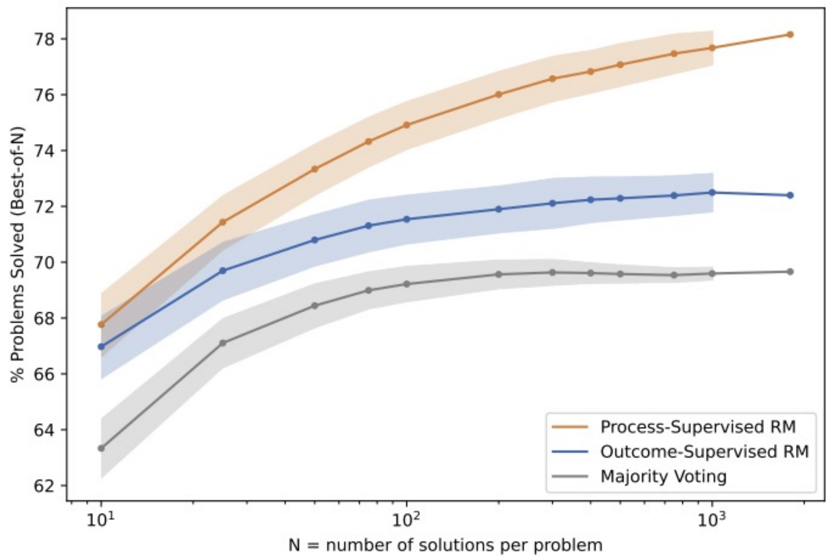
  - Take *product* strategy, and consider *neutral as positive*

Process-supervised Reward Model (PRM) [Lightman et al., 2023]

Process-supervised Reward Model vs. Outcome-supervised Reward Model

- PRM strongly outperform both ORM and majority-voting
- PRM is more effective on searching over large number of solutions (larger N)

	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	**78.2**	69.6



Limitation: Human-labeled feedback data is *very expensive* and *not scalable*

MATH-SHEPHERD [Wang et al., 2024]

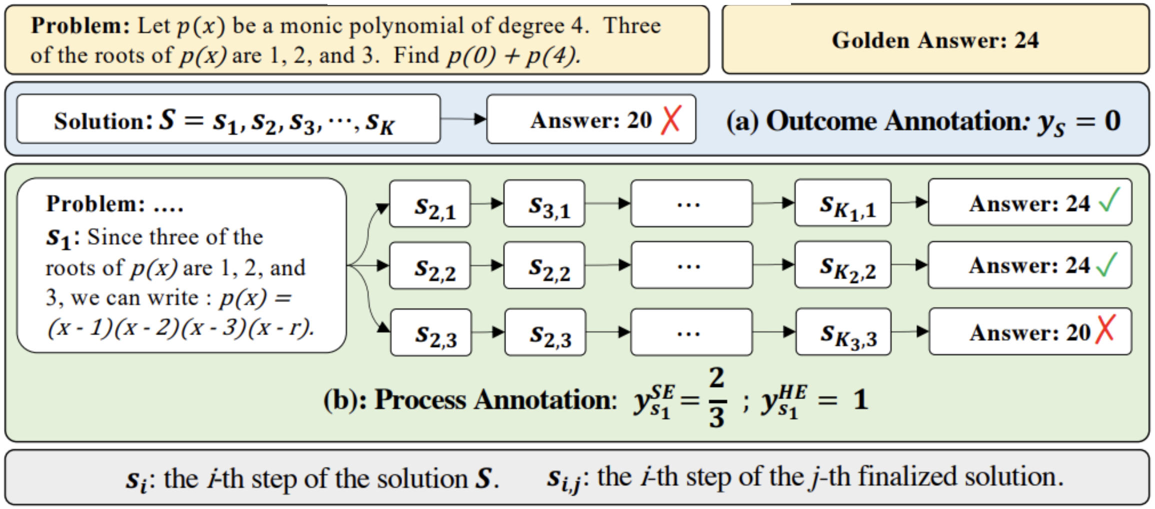
Idea: Automatically construct process-wise supervision data

- For an intermediate reasoning step, complete the reasoning process N times
- Hard Estimation(HE): The step can reach the correct answer

$$y_{s_i}^{HE} = \begin{cases} 1 & \exists a_j \in A, a_j = a^* \\ 0 & \text{Otherwise} \end{cases}$$

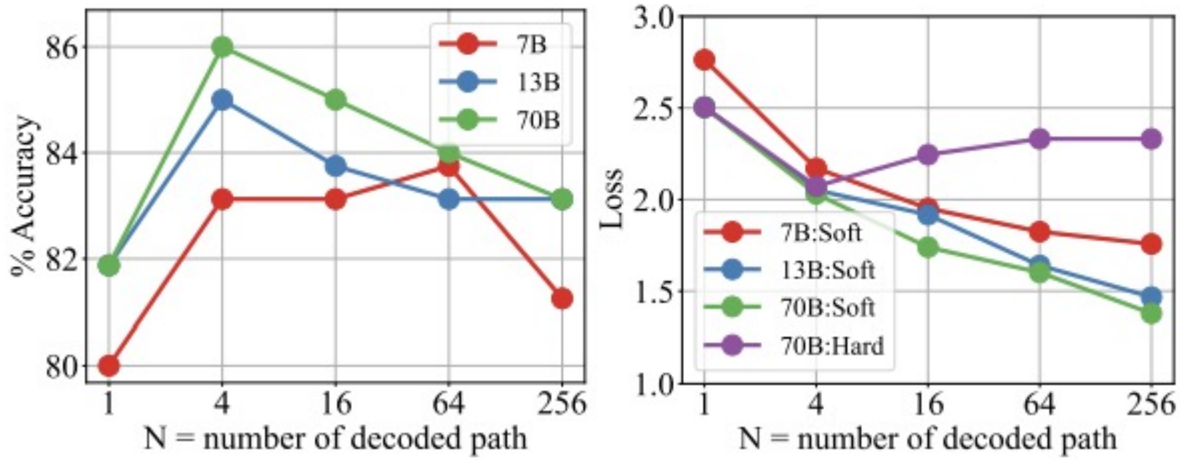
- Soft Estimation(SE): The frequency of trajectories reaching the correct answer

$$y_{s_i}^{SE} = \frac{\sum_{j=1}^{N} \mathbb{I}(a_j = a^*)}{N}.$$



MATH-SHEPHERD [Wang et al., 2024]

Hard Estimation vs. Soft Estimation



- Larger N led to more false-positives, decreasing annotation accuracy
- Hard Estimation(HE) showed negligible difference at N = 4 with (SE)
- Hard Estimation utilizes well to standard language modeling
  - Predicting special tokens ‘has potential’ and ‘no potential’ labels

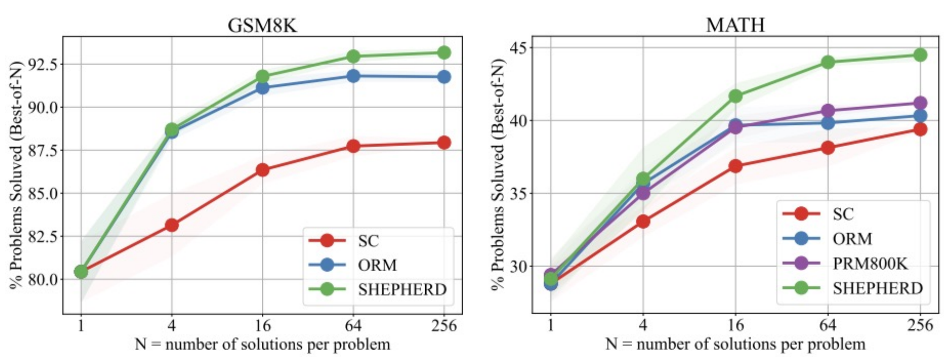
- Chose Hard Estimation(HE) as main score strategy

MATH-SHEPHERD [Wang et al., 2024]

Automated process-supervised verifier outperforms ORM consistently

- Outperformed human-annotated reward model, due to the data quantity (4x larger)

Models	Verifiers	GSM8K	MATH500
LLaMA2-70B: MetaMATH	Self-Consistency	88.0	39.4
	ORM	91.8	40.4
	Self-Consistency + ORM	92.0	42.0
	MATH-SHEPHERD (Ours)	93.2	44.5
	Self-Consistency + MATH-SHEPHERD (Ours)	92.4	45.2
LLemma-34B: MetaMATH	Self-Consistency	82.6	44.2
	ORM	90.0	43.7
	Self-Consistency + ORM	89.6	45.4
	MATH-SHEPHERD (Ours)	90.9	46.0
	Self-Consistency + MATH-SHEPHERD (Ours)	89.7	47.3
DeepSeek-67B: MetaMATH	Self-Consistency	88.2	45.4
	ORM	92.6	45.3
	Self-Consistency + ORM	92.4	47.0
	MATH-SHEPHERD (Ours)	**93.3**	47.0
	Self-Consistency + MATH-SHEPHERD (Ours)	92.5	**48.1**



MATH-SHEPHERD [Wang et al., 2024]

Reinforcement learning reasoning model with process supervision

- Proximal Policy Optimization(PPO) in a step-by-step manner

Models	GSM8K	MATH
LLaMA2-7B: MetaMATH	66.6	19.2
+ RFT	68.5	19.9
+ ORM-PPO	70.8	20.8
+ MATH-SHEPHERD-step-by-step-PPO (Ours)	73.2	21.6
Mistral-7B: MetaMATH	77.9	28.6
+ RFT	79.0	29.9
+ ORM-PPO	81.8	31.3
+ MATH-SHEPHERD-step-by-step-PPO (Ours)	**84.1**	**33.0**

- * RFT(Rejective Sampling Fine-tuning): SFT with sampled correct answer responses
- * ORM-PPO: PPO with outcome reward(correct/incorrect) of full solution

MATH-SHEPHERD can improve the reasoning model itself, not only working as verifier

MATH-SHEPHERD [Wang et al., 2024]

Reinforcement learning reasoning model with process supervision

- Proximal Policy Optimization(PPO) in a step-by-step manner

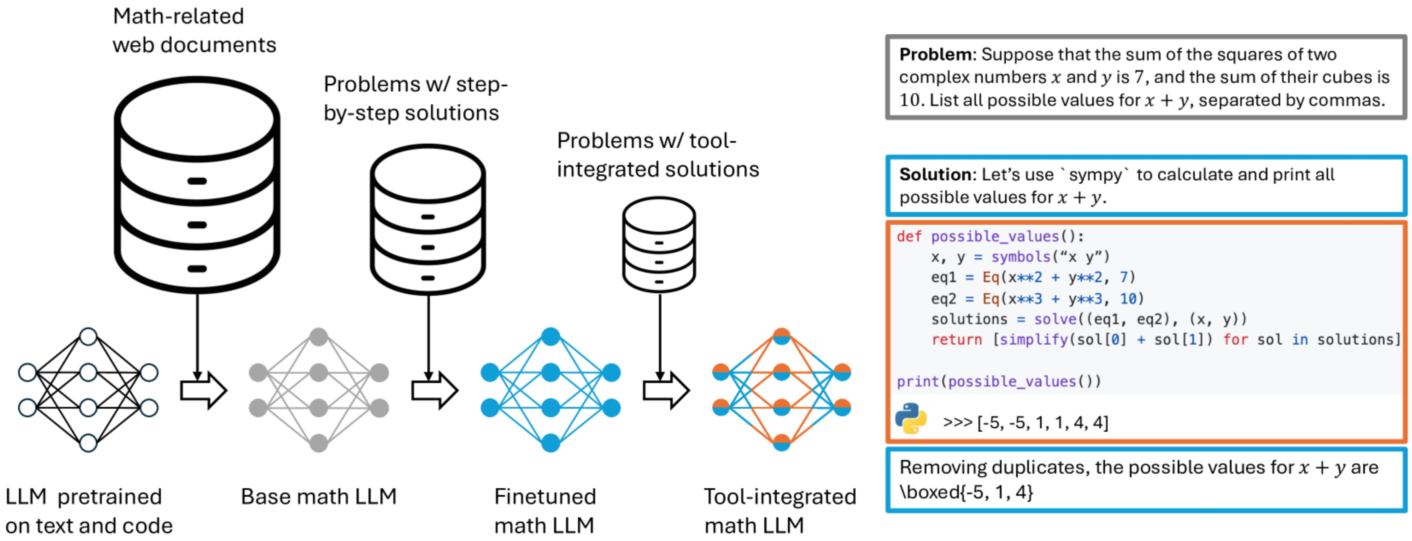
Models	**GSM8K**	**MATH**
LLaMA2-7B: MetaMATH	66.6	19.2
+ RFT	68.5	19.9
+ ORM-PPO	70.8	20.8
+ MATH-SHEPHERD-step-by-step-PPO (Ours)	73.2	21.6
Mistral-7B: MetaMATH	77.9	28.6
+ RFT	79.0	29.9
+ ORM-PPO	81.8	31.3
+ MATH-SHEPHERD-step-by-step-PPO (Ours)	**84.1**	**33.0**

- * RFT(Rejective Sampling Fine-tuning): SFT with sampled correct answer responses
- * ORM-PPO: PPO with outcome reward(correct/incorrect) of full solution

MATH-SHEPHERD can improve the reasoning model itself, not only working as verifier

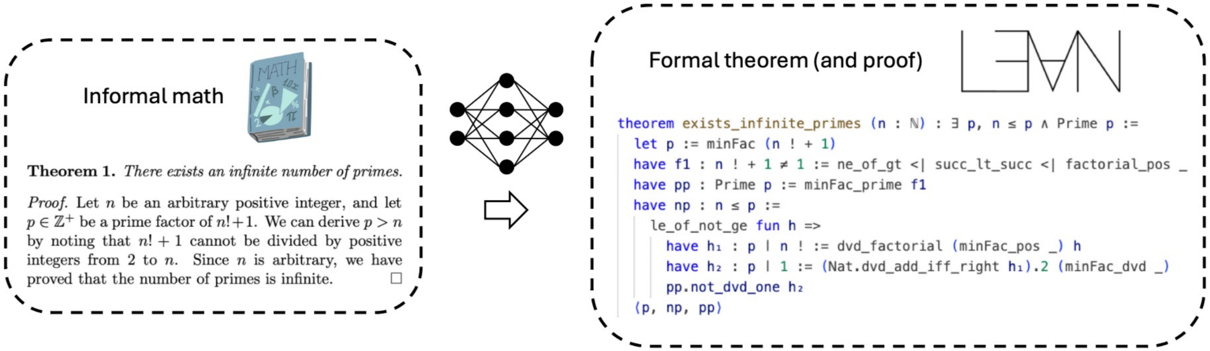
Formal Mathematical Reasoning [Yang et al., 2024]

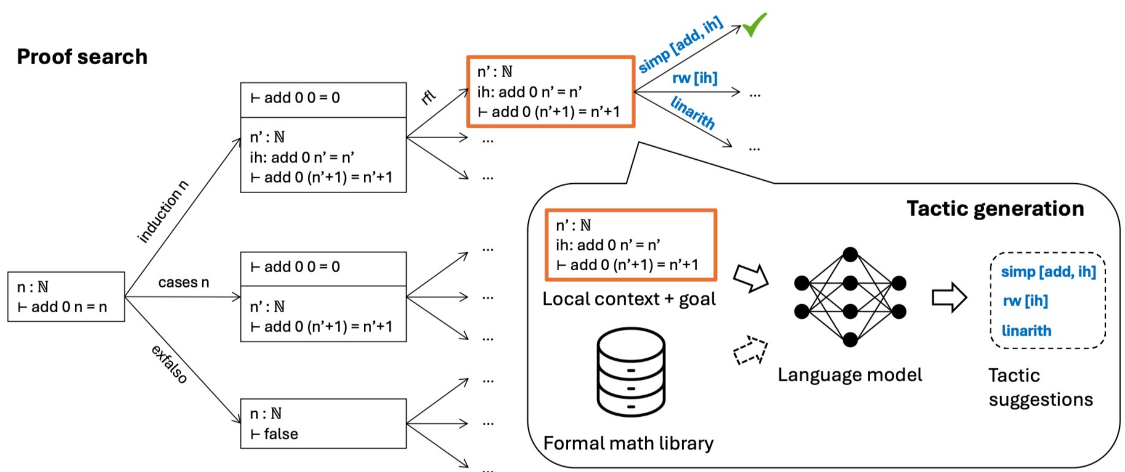
- LLMs show impressive capabilities in high school-level problems, but face limitations in **advanced mathematics**
- **Limitations of AI4Math in advanced mathematics:**
  - Data scarcity
  - Lack of Correctness Verifiability
    - GSM8k, MATH (pre-college mathematics) consist of single number solution problems
    - But none of the Millenium Prize Problems have numeric solutions



Formal Mathematical Reasoning [Yang et al., 2024]

- Formal mathematics with proof assistants (e.g. Lean, Coq, Isabelle)
  - Guarantee Correctness, Automatic Feedback
- **Key Tasks:** Autoformalization (top), Theorem Proving (bottom)



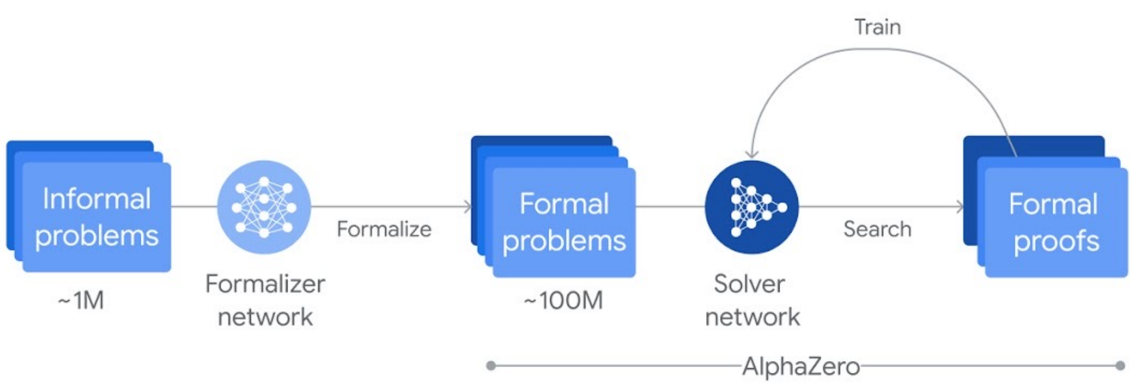


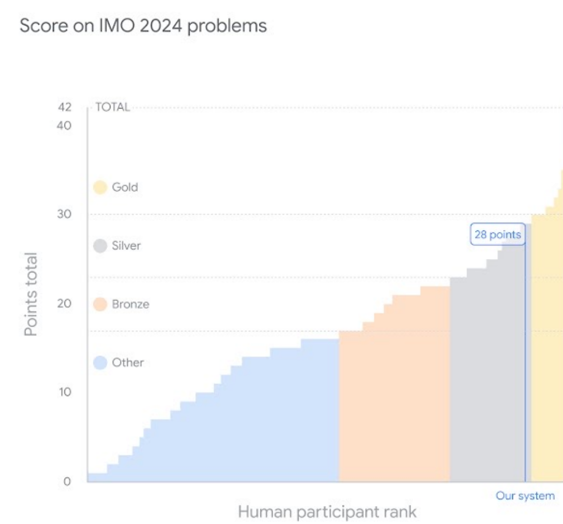
AlphaProof [Google Deepmind, 2024]

- Last year, AI achieving silver-medal standard at IMO 2024 problems
- 28 out of 42 points, solving four out of six problems

Method:

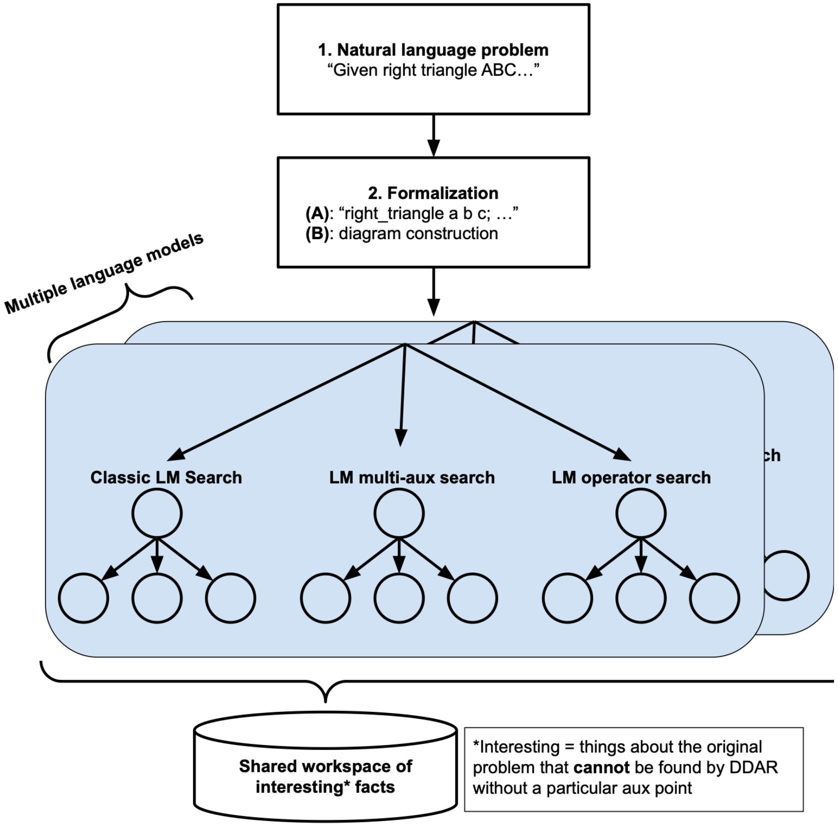
- Fine-tune *Gemini* for a formalizer network (Formal Language: LEAN)
- AlphaZero reinforcement learning algorithm
    - Generate solution candidates
    - Prove or disprove the solution by searching possivle proof steps in LEAN





AlphaGeometry2 [Google Deepmind, 2025]

- This year, AlphaGeometry2 solves 42/50 of all 2000-2024 IMO geometry problem
  - Surpassing an average **gold medalist** for the first time
- Symbolic engine: *DDAR* (Deductive Database Arithmetic Reasoning)
- Search Algorithm: Shared Knowledge Ensemble of Search Trees ***(SKEST)***



- Using multiple search trees
  - Deep, but narrow
  - Shallow, but wide
- Different LMs for each search tree

System description	IMO-AG-50 solved	IMO-AG-30 solved
OpenAI o1	0	0
Gemini thinking	0	0
AG1 DDAR (Trinh et al., 2024)	14	14
AG2 DDAR	16	15
TongGeometry DD (Zhang et al., 2024)	-	18
Average bronze medalist	27.1	19.3
Wu with AG1 DDAR (Sinha et al., 2024)	-	21
Average silver medalist	33.9	22.9
AG1 (Trinh et al., 2024)	27	25
Average gold medalist	40.9	25.9
Wu + AG1 (Sinha et al., 2024)	-	27
TongGeometry w/o value (Zhang et al., 2024)	-	28
AG2 with AG1 setup	38	28
TongGeometry full setting (Zhang et al., 2024)	-	**30**
AG2 full setting	**42**	**30**

Table of Contents

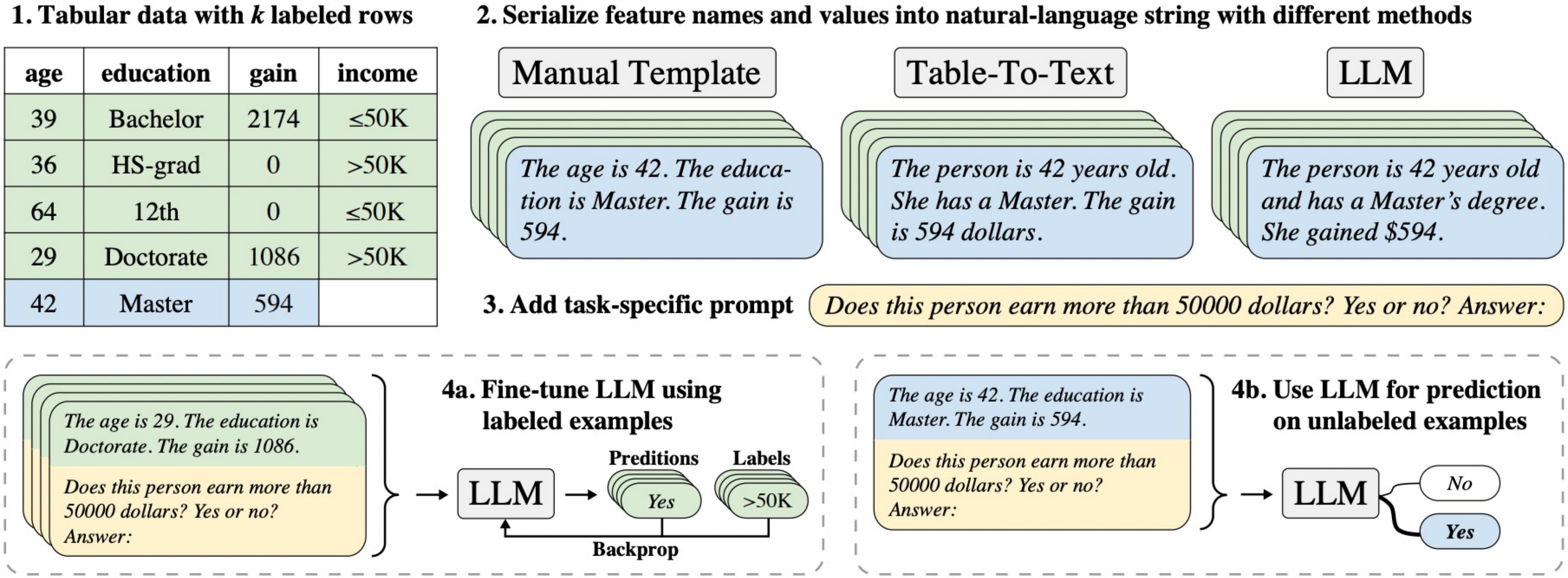
Table of Contents

Is it possible to use LLMs for tabular learning?

- The flexibility of language makes it possible to transform tabular data into language.

Define the task and feature descriptions in language.

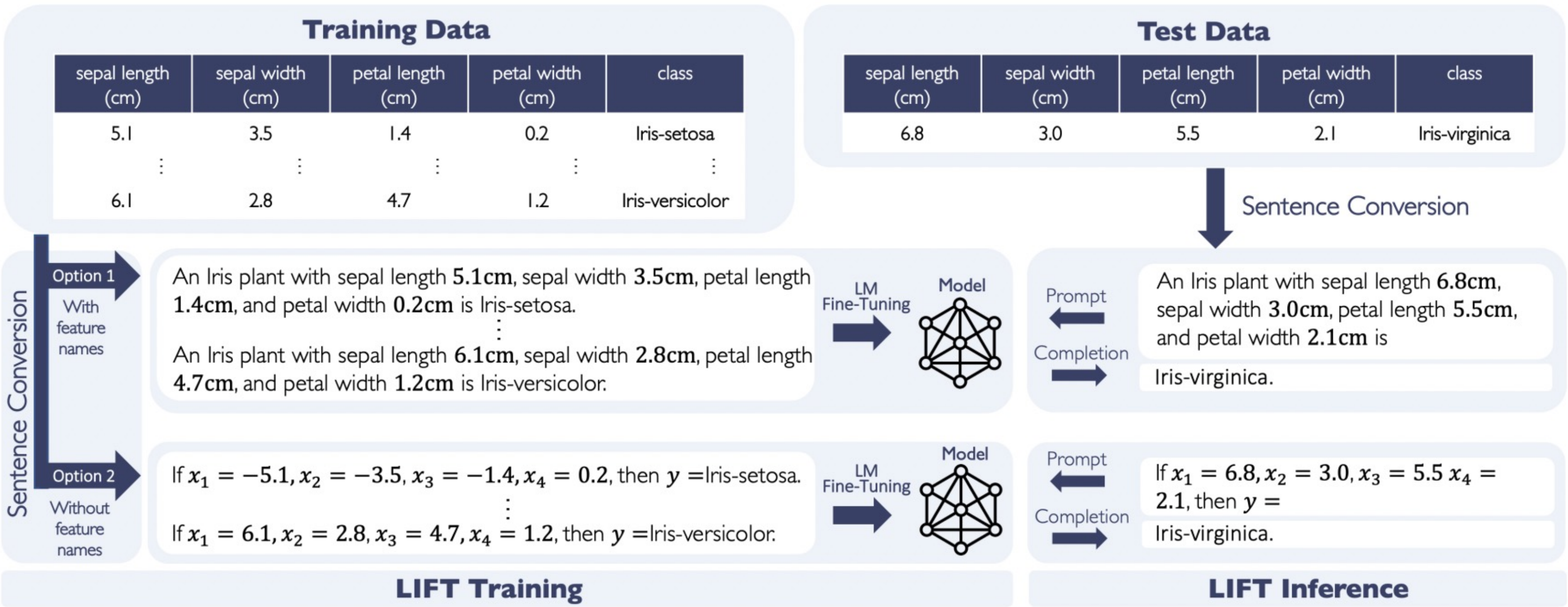
- Serialize data, and feed it into an LLM.



Indeed, LLMs are competitive for tabular learning.

Dinh et al. (2022):

- Investigated the performance of the fine-tuned LLMs on tabular data.



Indeed, LLMs are competitive for tabular learning.

Dinh et al. (2022):

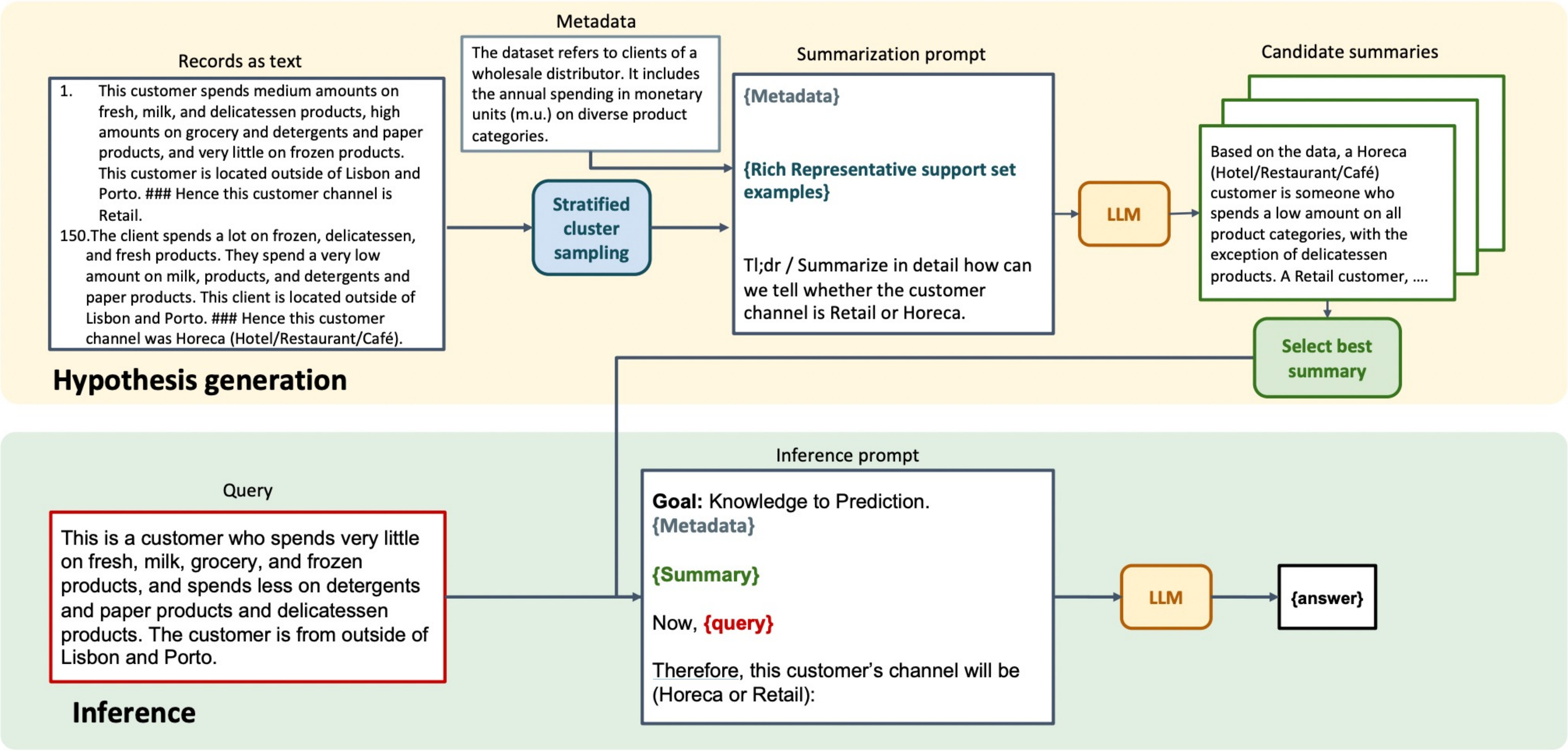
- Investigated the performance of the fine-tuned LLMs on tabular data.
- In-context learning with LIFT is competitive compared to prior methods.

Table 5: **Comparison of accuracies (↑) between ICL and fine-tuning with LIFT on OpenML datasets.** "LIFT/Full-Data" and "LIFT/Subset" represent LIFT on the full dataset and and its subset used correspondingly in the ICL setting (number of prompts). Here, the size of subset is chosen to satisfy the LMs' context length. Overall, LIFT/GPTs on full data achieve the best performances. However, when using the same number of samples, LIFT and ICL are more comparable in most cases. Note that both methods may be worse than MCC due to the limited training data in some cases.

Dataset (ID)	#Prompts	**MCC**	**GPT-J** In-Context	**GPT-J** LIFT/Subset	**GPT-J** LIFT/Full-data	**GPT-3** In-Context	**GPT-3** LIFT/Subset	**GPT-3** LIFT/Full-data
Breast (13)	35	70.69	56.90±19.51	**58.62±2.44**	64.94±11.97	62.07±1.41	**70.69±0.00**	71.26±1.62
TAE (48)	50	35.48	**34.33±1.47**	32.26±9.50	61.29±4.56	**37.64±4.02**	33.33±1.52	65.59±6.63
Vehicle (54)	14	25.88	**25.49±0.55**	26.04±1.69	64.31±2.37	**28.82±2.10**	23.73±2.27	70.20±2.73
Hamster (893)	43	53.33	48.89±3.14	**60.00±10.88**	55.55±16.63	**57.78±6.29**	53.33±0.00	53.33±0.00
Customers (1511)	29	68.18	56.06±17.14	**59.85±2.84**	85.23±1.61	60.61±1.42	**63.26±6.96**	84.85±1.42
LED (40496)	33	68.67	10.00±0.82	**13.04±3.27**	65.33±0.47	8.00±1.63	**11.33±2.62**	69.33±2.05

LLMs can operate effectively as **weak learners** [Manikandan et al., 2023]

- Prompt the LLM to summarize the tabular dataset.
- The summary acts as a prompt that the LLM uses to make predictions.
- Such prompts summarizing different subsets of data can be seen as weak learners for a boosting procedure.



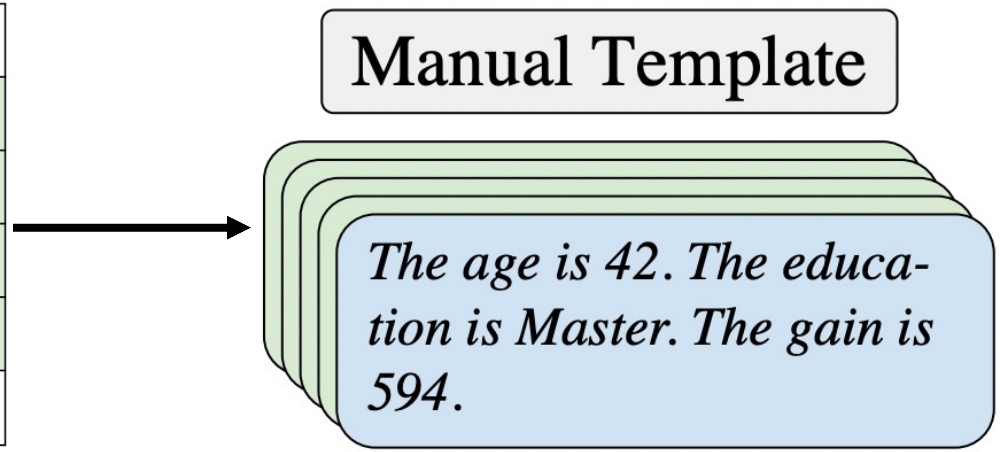
Step 1: Data conversion.

- To utilize LLMs with tabular data, it is necessary to convert the records into natural language descriptions.

But how?

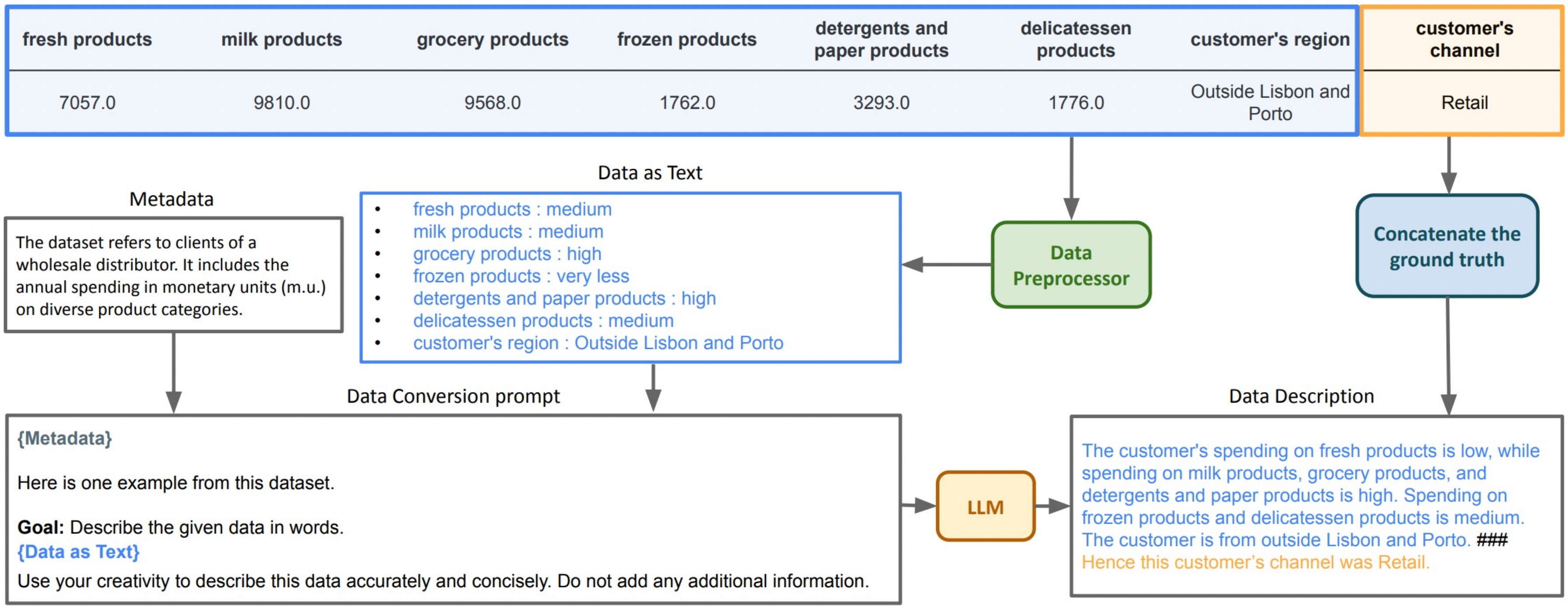
- LIFT [Dinh et al., 2022] inserts attribute values into predefined templates.
- However, this approach often produces unnatural descriptions that differ from how humans might describe the data.
- Depending on the dataset, designing the template by hand can also be challenging.

age	education	gain	income
39	Bachelor	2174	≤50K
36	HS-grad	0	>50K
64	12th	0	≤50K
29	Doctorate	1086	>50K
42	Master	594	



Step 1: Data conversion.

- To utilize LLMs with tabular data, it is necessary to convert the records into natural language descriptions.
- Get data descriptions by zero-shot prompting the LLM.
  - With information about the dataset (Metadata) and a textual representation of the tabular record (Data as Text).



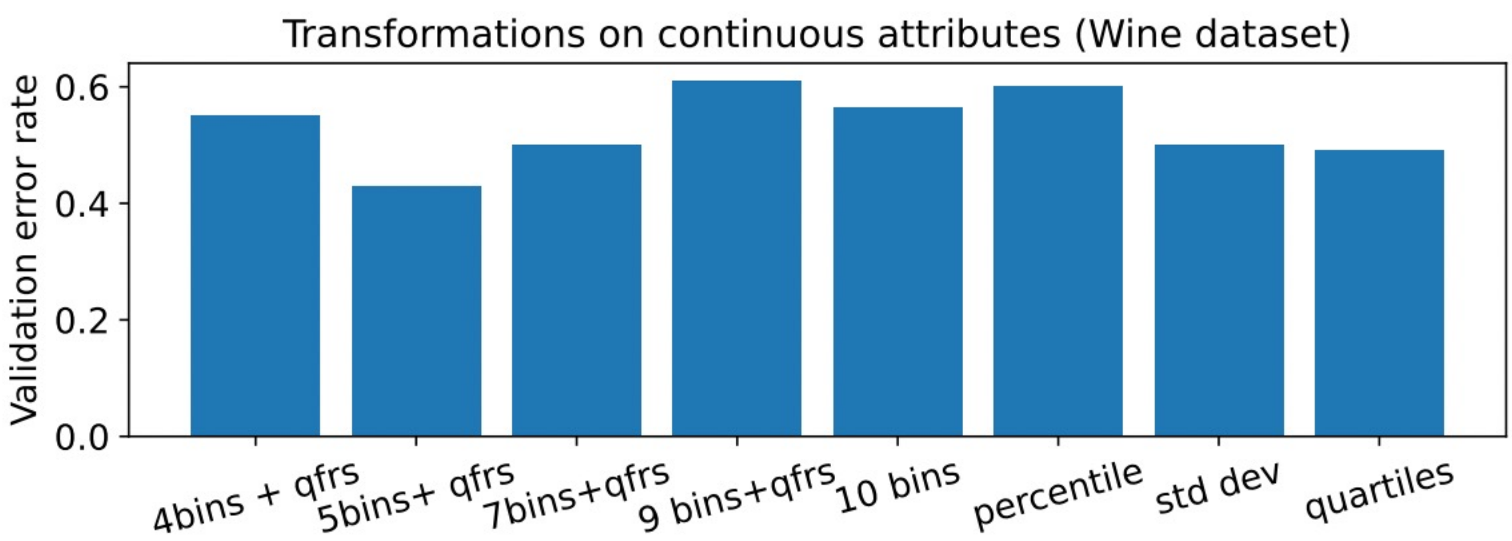
Step 1: Data conversion.

- To utilize LLMs with tabular data, it is necessary to convert the records into natural language descriptions.
- Get data descriptions by zero-shot prompting the LLM.
  - With information about the dataset (Metadata) and a textual representation of the tabular record (Data as Text).
- Challenge: Naively including numerical values in the descriptions can lead to poor performance.
  - Bin all numerical features into percentiles and encode them descriptively.

Method	Data Representation	Example as text
`4 bins + quantifiers` {very low, low, high, very high}	- *spending on fresh products :* low - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* high - *spending on delicatessen products :* very high - *customer's region :* Outside Lisbon and Porto	This customer spends low amounts on fresh products, very high amounts on milk products, high amounts on grocery products, frozen products, detergents and paper products, and very high amounts on delicatessen products. They are located outside of Lisbon and Porto.
`5 bins + quantifiers` {very low, low, medium, high, very high}	- *spending on fresh products :* medium - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* high - *spending on delicatessen products :* very high - *customer's region :* Outside Lisbon and Porto	This customer from outside Lisbon and Porto spends medium on fresh products, very high on milk products, high on grocery products, high on frozen products, high on detergents and paper products, and very high on delicatessen products.
`7 bins + quantifiers` {extremely low, very low, low, medium, high, very high, extremely high}	- *spending on fresh products :* low - *spending on milk products :* very high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* very high - *spending on delicatessen products :* extremely high - *customer's region :* Outside Lisbon and Porto	This customer situated outside Lisbon and Porto spends low on fresh products, very high on milk products, high on grocery products, high on frozen products, very high on detergents and paper products, and extremely high on delicatessen products.
`9 bins + quantifiers` {lowest, extremely low, very low, low, medium, high, very high, extremely high, highest}	- *spending on fresh products :* low - *spending on milk products :* extremely high - *spending on grocery products :* high - *spending on frozen products :* high - *spending on detergents and paper products :* very high - *spending on delicatessen products :* highest - *customer's region :* Outside Lisbon and Porto	This customer spends low amounts on fresh products, extremely high amounts on milk products, high amounts on grocery products, frozen products, detergents and paper products, and highest amounts on delicatessen products. They are located outside Lisbon and Porto.

Step 1: Data conversion.

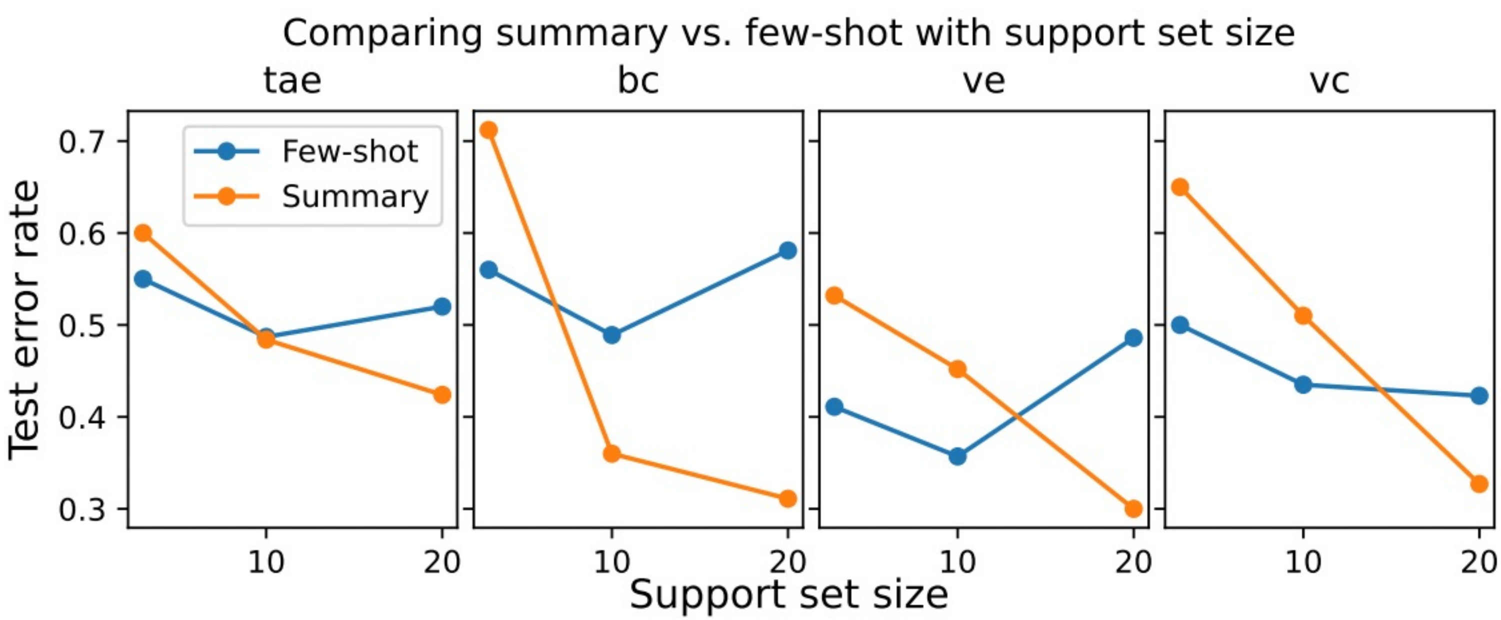
- To utilize LLMs with tabular data, it is necessary to convert the records into natural language descriptions.
- Get data descriptions by zero-shot prompting the LLM.
  - With information about the dataset (Metadata) and a textual representation of the tabular record (Data as Text).
- Challenge: Naively including numerical values in the descriptions can lead to poor performance.
  - Bin all numerical features into percentiles and encode them descriptively.



Step 2: Weak learning via summarization.

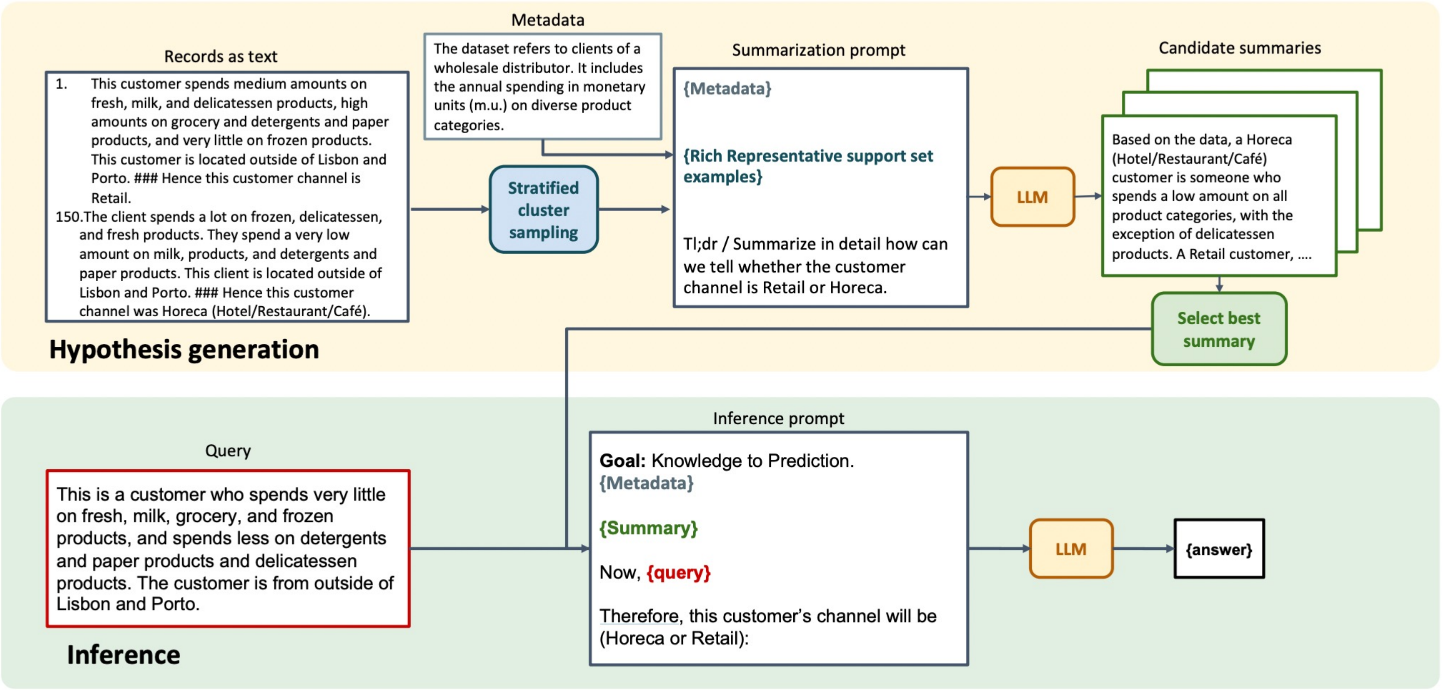
- A typical method for performing few-shot learning with LLMs involves providing a small number of demonstrations.
- However,
  - There may be a large number of data points that do not fit within the LLM context.
  - Increasing the number of examples in the context does not always improve performance.

  → **Necessitate alternative approaches to weak learning via LLMs.**



Step 2: Weak learning via summarization.

- A typical method for performing few-shot learning with LLMs involves providing a small number of demonstrations.
- Produce summaries of a collection of examples.
  - Summarization naturally encourages the extraction of representative information in data.
  - First, perform summarization on the data by calling the LLM.
  - Second, by using the summary as a prompt, the LLM performs inference.



Step 2: Weak learning via summarization.

- A typical method for performing few-shot learning with LLMs involves providing a small number of demonstrations.
- Produce summaries of a collection of examples.
  - Summarization naturally encourages the extraction of representative information in data.
  - First, perform summarization on the data by calling the LLM.
  - Second, by using the summary as a prompt, the LLM performs inference.
- **Challenge 1**: The sampled summary can sometimes be noisy.
  - Generate a fixed number of summaries and pick the the smallest validation error rate.
- **Challenge 2**: The context size of existing LLMs is still limited.
  - We cannot fit the entire dataset into the context for summarization.

  → Use only a representative subset obtained through weighted stratified sampling.

Step 2: Weak learning via summarization.

- A typical method for performing few-shot learning with LLMs involves providing a small number of demonstrations.
- Produce summaries of a collection of examples.
  - Summarization naturally encourages the extraction of representative information in data.
  - First, perform summarization on the data by calling the LLM.
  - Second, by using the summary as a prompt, the LLM performs inference.
- **Challenge 1**: The sampled summary can sometimes be noisy.
  - Generate a fixed number of summaries and pick the the smallest validation error rate.
- **Challenge 2**: The context size of existing LLMs is still limited.
  - We cannot fit the entire dataset into the context for summarization.

  → Use only a representative subset obtained through weighted stratified sampling.

Step 3: Boosting.

- Use the AdaBoost algorithm to produce an ensemble with these collections of summary-based weak learners.

LLMs with summarization are a good candidate for creating weak learners.

- The LLMs themselves do not have enough built-in knowledge to succeed at tabular data zero-shot.
- Few-shot consistently improves the test performance compared to zero-shot.
  - Added information is crucial for LLMs to work on tabular datasets.
- Summary consistently improves upon few-shot.
  - Summarization is a powerful way to improve few-shot performance.
- Boosting with summarization consistently outperforms all other prompting-based approaches.

Dataset	Data Type	Size	Zero-shot	Few-shot	Summary	Summary Boosting
caesarian [cae] (42901)	1c4d	80	0.425 ± 0.04	0.388 ± 0.02	0.350 ± 0.04	0.300 ± 0.04
iris (61)	4c0d	150	0.680 ± 0.02	0.460 ± 0.01	0.275 ± 0.07	0.193 ± 0.03
tae (48)	1c4d	151	0.556 ± 0.07	0.494 ± 0.01	0.474 ± 0.02	0.454 ± 0.03
glass (41)	9c0d	214	0.486 ± 0.01	0.473 ± 0.01	0.466 ± 0.02	0.370 ± 0.02
breast-cancer [bc] (13)	7c5d	277	0.754 ± 0.02	0.516 ± 0.02	0.337 ± 0.02	0.288 ± 0.02
visualizing-environmental [ve] (678)	3c0d	111	0.522 ± 0.01	0.366 ± 0.01	0.304 ± 0.02	0.268 ± 0.03
analcatdata-chlamydia [ac] (535)	2c2d	100	0.200 ± 0.00	0.200 ± 0.00	0.170 ± 0.01	0.170 ± 0.01
wine (43571)	13c0d	178	0.820 ± 0.03	0.674 ± 0.02	0.475 ± 0.01	0.320 ± 0.01
blood-transfusion-center [btc] (1464)	4c0d	748	0.544 ± 0.01	0.430 ± 0.00	0.258 ± 0.04	0.240 ± 0.04
somerville-happiness-survey [shs] [Koczkodaj, 2018]	0c7d	143	0.416 ± 0.03	0.385 ± 0.03	0.422 ± 0.02	0.350 ± 0.02
vehicle (54)	18c0d	846	0.765 ± 0.00	0.560 ± 0.01	0.510 ± 0.02	0.410 ± 0.04
statlog-heart [stath] [Dua and Graff, 2017]	6c7d	270	0.551 ± 0.01	0.528 ± 0.01	0.444 ± 0.05	0.430 ± 0.01
verterbra-column [vc] (1524)	6c0d	310	0.714 ± 0.03	0.435 ± 0.06	0.327 ± 0.01	0.262 ± 0.01
ecoli (1011)	7c0d	336	0.581 ± 0.02	0.562 ± 0.01	0.480 ± 0.01	0.270 ± 0.03
haberman-survival [hs] (43)	3c0d	306	0.308 ± 0.02	0.262 ± 0.01	0.277 ± 0.01	0.250 ± 0.01
diabetes [dia] (37)	8c0d	768	0.446 ± 0.04	0.400 ± 0.00	0.360 ± 0.01	0.344 ± 0.01
visualizing-hamster [hams] (708)	5c0d	73	0.464 ± 0.03	0.481 ± 0.05	0.360 ± 0.02	0.207 ± 0.00
wholesale-customers [wc] (1511)	6c1d	440	0.364 ± 0.01	0.347 ± 0.01	0.349 ± 0.02	0.330 ± 0.00

When the datasets have many numerical features, the performance can be worse.

- LLMs are fairly bad at quantitative reasoning without fine-tuning.

Summary Boosting performs very well when the size of the dataset is very small.

- LLMs have a large amount of generic prior about the world from pre-training.
- When the dataset is large, this prior knowledge becomes less relevant, and fine-tuning becomes more competitive.

Dataset	Data Type	Size	Summary Boosting	LIFT	KNN	TabPFN	Xgboost
cae (42901)	1c4d	80	0.300 ± 0.04	0.312 ± 0.02	0.300 ± 0.00	0.425 ± 0.07	0.412 ± 0.05
iris (61)	4c0d	150	0.193 ± 0.03	0.100 ± 0.01	0.106 ± 0.02	0.027 ± 0.00	0.054 ± 0.04
tae (48)	1c4d	151	0.454 ± 0.03	0.480 ± 0.04	0.532 ± 0.01	0.450 ± 0.13	0.464 ± 0.01
glass (41)	9c0d	214	0.370 ± 0.02	0.218 ± 0.02	0.294 ± 0.03	0.158 ± 0.05	0.254 ± 0.05
bc (13)	7c5d	277	0.288 ± 0.02	0.318 ± 0.01	0.277 ± 0.02	0.264 ± 0.01	0.270 ± 0.01
ve (678)	3c0d	111	0.268 ± 0.03	0.430 ± 0.04	0.308 ± 0.01	0.370 ± 0.04	0.279 ± 0.02
ac (535)	2c2d	100	0.170 ± 0.01	0.180 ± 0.06	0.170 ± 0.01	0.090 ± 0.01	0.110 ± 0.04
wine (43571)	13c0d	178	0.320 ± 0.01	0.065 ± 0.01	0.214 ± 0.05	0.040 ± 0.01	0.040 ± 0.01
btc (1464)	4c0d	748	0.240 ± 0.04	0.270 ± 0.01	0.238 ± 0.00	0.209 ± 0.01	0.219 ± 0.01
shs [Koczkodaj, 2018]	0c7d	143	0.350 ± 0.02	0.419 ± 0.02	0.326 ± 0.03	0.392 ± 0.00	0.406 ± 0.00
vehicle (54)	18c0d	846	0.410 ± 0.04	0.111 ± 0.16	0.636 ± 0.01	0.178 ± 0.01	0.260 ± 0.00
stath [Dua and Graff, 2017]	6c7d	270	0.430 ± 0.01	0.122 ± 0.17	0.244 ± 0.03	0.148 ± 0.03	0.215 ± 0.00
vc (1524)	6c0d	310	0.262 ± 0.01	0.192 ± 0.03	0.318 ± 0.02	0.135 ± 0.00	0.187 ± 0.04
ecoli (1011)	7c0d	336	0.270 ± 0.03	0.126 ± 0.03	0.211 ± 0.03	0.036 ± 0.02	0.066 ± 0.01
hs (43)	3c0d	306	0.250 ± 0.01	0.314 ± 0.03	0.278 ± 0.00	0.262 ± 0.02	0.281 ± 0.02
dia (37)	8c0d	768	0.344 ± 0.01	0.324 ± 0.04	0.353 ± 0.02	0.238 ± 0.03	0.234 ± 0.00
hams (708)	5c0d	73	0.207 ± 0.00	0.334 ± 0.08	0.528 ± 0.02	0.328 ± 0.01	0.411 ± 0.01
wc (1511)	6c1d	440	0.330 ± 0.00	0.125 ± 0.04	0.043 ± 0.00	0.088 ± 0.00	0.098 ± 0.02

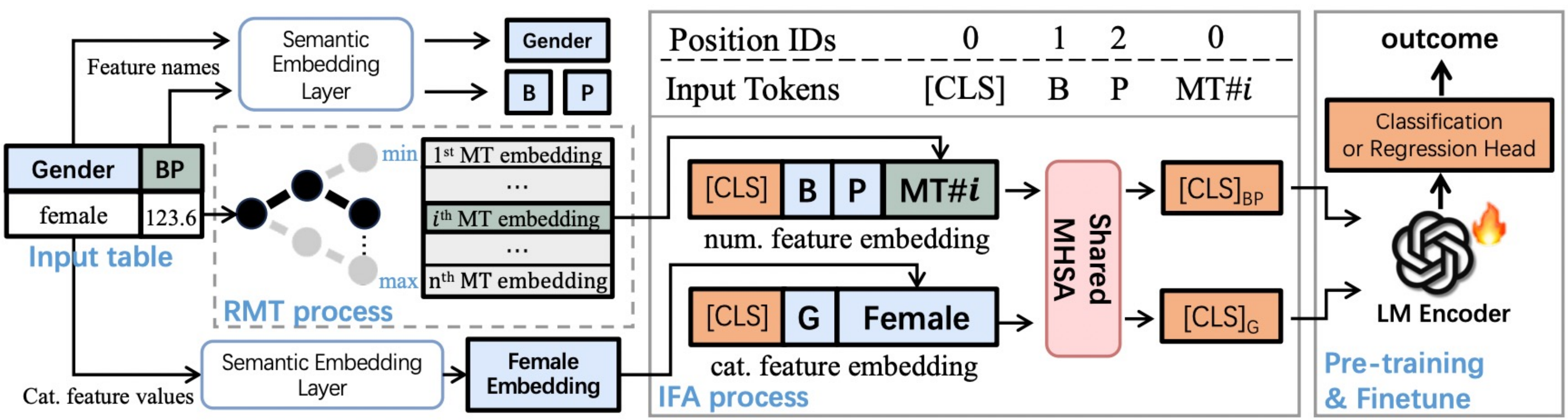
Tabular features are roughly categorized into:

- Discrete type (categorical, binary, or string features)
  - Can be naturally understood by LLMs.
  - E.g., “Male” and “Female” are values of the discrete feature “Gender.”
- Continuous type (i.e., numerical features)
  - Still difficult to make fully understandable to LLMs.
  - Wide range of values & counter-intuitive meanings of exact numerical values.

Discrete text representation space is incompatible with numerical values.

Tabular **P**rediction adapted **BERT a**pproach [Yan et al., 2023]

- TP-BERTa is built on the basis of RoBERTa as default.
- Discretizes numerical feature values as relative magnitude tokens (RMT).
  - Treat them as some meaningful words in the LLM's vocabulary.
- Intra-feature attention (IFA) module attentively fuses the embeddings of a feature's name and value.
  - Achieves feature order-agnostic prediction.



Tabular **P**rediction adapted **BERT a**pproach [Yan et al., 2023]

- GBDTs still outperform classical and advanced DNNs in typical regimes.
- However, the pre-trained TP-BERTa shows competitive performances.
- TP-BERTa is stably promising when discrete features begin to dominate.
- While for purely numerical datasets, GBDT are still better choices.

Baselines	80 downstream binary classification tasks						65 downstream regression tasks					
	All	$\alpha > 0$	$\alpha \geq 1$	$\alpha = 0$	$\beta > 0$	$\beta > 0.5$	All	$\alpha > 0$	$\alpha \geq 1$	$\alpha = 0$	$\beta > 0$	$\beta > 0.5$
XGBoost(d)	7.7(4.0)	7.8(4.1)	9.2(4.0)	6.8(3.5)	8.2(4.1)	8.3(3.9)	7.7(4.4)	7.7(4.6)	7.3(4.1)	7.8(4.0)	8.0(4.7)	9.2(4.3)
CatBoost(d)	6.7(4.1)	6.8(4.0)	7.4(4.0)	6.0(4.6)	7.0(4.1)	6.8(4.2)	5.5(2.7)	5.5(2.6)	5.5(2.7)	5.6(3.0)	5.5(2.7)	5.8(3.2)
FTT(d)	7.1(3.5)	7.0(3.5)	6.6(3.5)	6.9(3.6)	6.9(3.6)	7.2(3.6)	7.8(2.7)	7.8(2.5)	8.2(3.0)	7.6(3.2)	8.0(2.6)	8.3(1.3)
TransTab(d)	11.0(4.5)	11.2(4.5)	11.2(4.1)	10.2(4.6)	11.6(4.3)	11.7(4.2)	12.1(4.0)	12.1(3.8)	13.3(2.2)	12.4(4.5)	12.0(4.0)	13.6(1.2)
XGBoost(t)	6.2(4.1)	6.3(4.1)	6.5(4.3)	5.9(4.2)	6.5(4.2)	6.7(4.5)	4.5(3.7)	4.3(3.8)	**3.3(3.3)**	5.0(3.5)	4.7(3.9)	4.1(3.2)
CatBoost(t)	5.9(3.8)	6.3(3.9)	7.1(4.1)	**4.9(3.1)**	6.4(3.9)	6.4(4.1)	5.5(3.6)	5.7(3.6)	5.8(3.5)	4.9(3.7)	5.7(3.7)	6.1(3.8)
MLP(t)	8.6(4.0)	8.9(3.9)	8.7(4.1)	8.5(4.1)	8.5(3.9)	8.3(4.1)	8.5(3.6)	8.8(3.4)	9.3(3.2)	7.6(4.1)	9.0(3.4)	7.5(3.8)
AutoInt(t)	8.0(3.5)	7.8(3.3)	7.4(3.4)	8.6(4.0)	7.7(3.4)	7.7(3.2)	8.3(3.0)	8.6(3.0)	8.5(2.7)	7.4(3.1)	8.3(3.0)	8.2(3.2)
DCNv2(t)	7.9(3.9)	8.0(3.9)	8.4(3.8)	7.9(4.0)	7.7(3.9)	8.8(3.3)	8.4(3.4)	8.4(3.5)	8.5(3.1)	8.5(3.2)	8.4(3.5)	7.2(3.5)
TabNet(t)	12.1(3.5)	12.4(3.3)	12.7(2.7)	11.5(4.2)	12.3(3.4)	12.3(3.8)	12.6(3.6)	13.2(2.6)	13.1(2.4)	10.5(5.1)	13.5(1.9)	14.1(1.4)
SAINT(t)	8.2(3.8)	8.0(3.7)	8.1(4.1)	8.7(4.2)	7.9(3.8)	7.5(3.9)	7.6(3.8)	7.3(3.9)	7.7(3.3)	8.4(3.7)	6.6(3.6)	7.2(3.0)
FTT(t)	6.8(3.5)	6.8(3.6)	6.5(3.4)	6.2(3.3)	6.9(3.6)	6.9(3.9)	7.9(3.4)	7.6(3.3)	7.7(3.1)	9.0(3.4)	7.2(3.0)	6.8(3.2)
XTab(t)	9.8(4.0)	9.7(4.0)	8.9(3.8)	10.5(4.1)	9.4(4.0)	9.9(3.7)	12.4(2.8)	12.5(2.8)	13.3(1.6)	12.0(3.0)	12.4(2.9)	13.1(1.8)
Ours_j(d)	8.4(4.5)	7.7(4.5)	7.0(5.0)	9.9(4.1)	7.9(4.6)	7.0(4.7)	6.9(4.6)	6.3(4.4)	4.8(3.9)	8.5(5.0)	6.5(4.5)	5.2(3.9)
Ours_s(d)	**5.8(4.0)**	**5.1(3.9)**	**4.4(3.3)**	7.5(3.7)	**5.2(4.1)**	**4.5(3.4)**	**4.3(2.8)**	**4.1(2.6)**	3.9(2.4)	**4.8(3.4)**	**4.3(2.7)**	**3.6(2.8)**

Tabular **P**rediction adapted **BERT a**pproach [Yan et al., 2023]

- Why were LMs neglected on tabular prediction?
  - Numerical encoding strategy comparison.

1. Value2Str: directly treating numerical values as strings.
2. VMFE: value-multiplied feature name embeddings.

→ These strategies hurt AUC scores on the most significantly changed datasets.

Table 2: Performance changes on encoding strategy substitution and IFA ablation using 80 binary classification datasets. The column "$|\Delta| \leq 0.5\%$" denotes the number of datasets with AUC variation less than 0.5% (these datasets are called "insignificantly changed datasets" due to different random seeds); the other "Δ" columns use similar denotations. "Avg. diff." means the average performance difference on significantly changed datasets. "Avg. training time ratio" is the average ratio of training time compared to using the IFA module. Appendix 11 gives more detailed performances.

Comparison (numerical encoding strategies)				
Substitution	$\|\Delta\| \leq 0.5\%$	$\Delta < -0.5\%$	$\Delta > 0.5\%$	Avg. diff.
Value2Str (Borisov et al., 2022b)	16	54	10	-12.45%
VMFE (Ye et al., 2023)	34	36	10	-3.44%
Ablation (w/o IFA module)				
Avg. training time ratio	$\|\Delta\| \leq 0.5\%$	$\Delta < -0.5\%$	$\Delta > 0.5\%$	Avg. diff.
1.32	14	52	14	-4.17%

Tabular **P**rediction adapted **BERT a**pproach [Yan et al., 2023]

- Why were LMs neglected on tabular prediction?
  - Numerical encoding strategy comparison.
  - IFA module ablation.
    - A noticeable performance degradation occurs when directly feeding all feature names and values to the LM.

    → LMs are likely to be confused when they process a pile of unmatched feature name-value texts.

Table 2: Performance changes on encoding strategy substitution and IFA ablation using 80 binary classification datasets. The column "$|\Delta| \leq 0.5\%$" denotes the number of datasets with AUC variation less than 0.5% (these datasets are called "insignificantly changed datasets" due to different random seeds); the other "Δ" columns use similar denotations. "Avg. diff." means the average performance difference on significantly changed datasets. "Avg. training time ratio" is the average ratio of training time compared to using the IFA module. Appendix 11 gives more detailed performances.

Comparison (numerical encoding strategies)				
Substitution	$\|\Delta\| \leq 0.5\%$	$\Delta < -0.5\%$	$\Delta > 0.5\%$	Avg. diff.
Value2Str (Borisov et al., 2022b)	16	54	10	-12.45%
VMFE (Ye et al., 2023)	34	36	10	-3.44%
Ablation (w/o IFA module)				
Avg. training time ratio	$\|\Delta\| \leq 0.5\%$	$\Delta < -0.5\%$	$\Delta > 0.5\%$	Avg. diff.
1.32	14	52	14	-4.17%

Tabular **P**rediction adapted **BERT a**pproach [Yan et al., 2023]

- Why were LMs neglected on tabular prediction?
  - Numerical encoding strategy comparison.
  - IFA module ablation.
  - Using RoBERTa weights is better than random weights.

  → LM weights have inherently entailed meaningful semantic knowledge.
  - A more significant leap can be achieved by further pre-training on extensive tabular data.

  → LMs are also effective in transferring tabular data knowledge and suitable for cross-table pre-training.

Table 3: Performance changes by comparing the pre-trained TP-BERTa with (1) TP-BERTa randomly initialized and (2) TP-BERTa initialized with the RoBERTa weights. "Avg. diff." is calculated by excluding the datasets with $|\Delta| \leq 0.5\%$.

Comparison (w/ no pre-training) using 80 binary classification datasets						
Initialization	$\lvert\Delta\rvert \leq 0.5\%$	$\Delta < -0.5\%$	$\Delta > 0.5\%$	$\Delta < -3\%$	$\Delta > 3\%$	Avg. diff.
Random	29	41	10	26	5	-3.16%
RoBERTa	26	35	19	21	6	-2.79%

Current LLM-based tabular learning methods have some limitations.

- At least one LLM inference per sample is required.
- Fine-tuning the LLM can be infeasible.
  - Recently proposed top-performance LLMs only permit limited access via APIs.
- Not suitable with lengthy prompts.
  - Text length becomes long when the number of features in tabular data grows.

Han et al. (2024): Aims to understand the criteria underlying LLM predictions.

- For the task of predicting a particular disease, the LLM can directly infer and generate rules that determine which feature conditions result in identifying the disease.

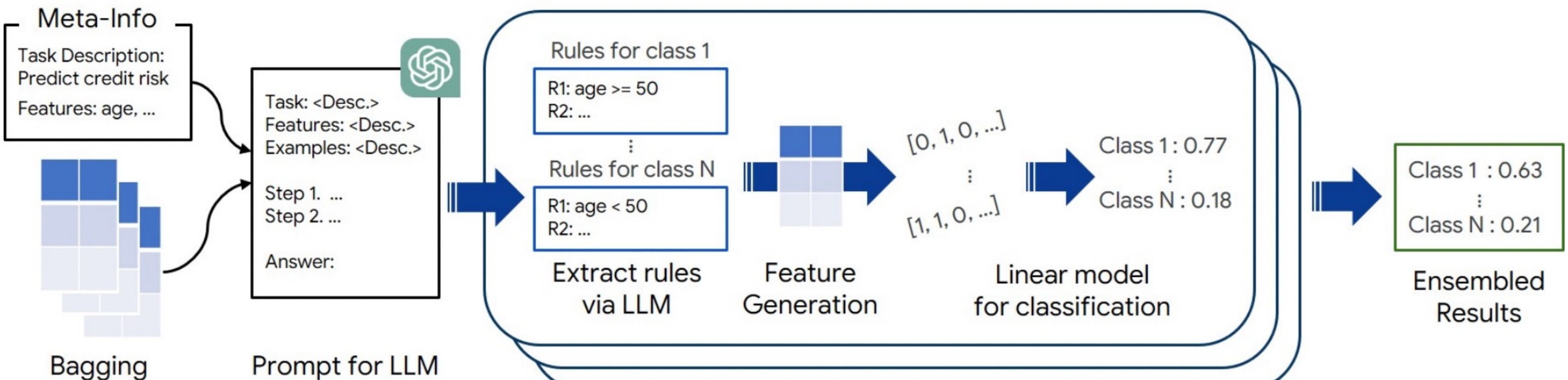
Step 1: FeatLLM extracts rules for each class.

- Utilizing prior knowledge and few-shot examples.

Step 2: These rules are parsed and applied to create binary features for samples.

Step 3: A linear layer is trained on features to estimate class likelihoods.

Step 4: This procedure is repeated multiple times for ensembling.



Prompt design for extracting rules.

- Guide the problem-solving process to mimic how an expert human might approach it.

You are an expert. Given the task description and the list of features and data examples, you are extracting conditions for each answer class to solve the task.

Task: <Task description>
Features: <Feature descriptions>
Examples: <Serialized training examples>

Let's first understand the problem and solve the problem step by step.

Step 1. Analyze the causal relationship or tendency between each feature and task description based on general knowledge and common sense within a short sentence.

Step 2. Based on the above examples and Step 1's results, infer 10 different conditions per answer, following the format below. The condition should make sense, well match examples, and must match the format for [condition] according to value type.

Format for Response:
10 different conditions for class [Class name]:
- [Condition]
...

Format for [Condition]:
For the categorical variable only,
- [Feature] is in [List of categories]
For the numerical variable only,
- [Feature] (> or >= or < or <=) [Value]
- [Feature] is within range of [Value_start, Value_end]

Answer:
Step 1.

Prompt design for extracting rules.

- Guide the problem-solving process to mimic how an expert human might approach it.
  - Basic information description: Essential information for solving the problem.
    - The task description is formulated as a question.
    - The feature description indicates its value type and includes information.
    - Few training samples are serialized into text, along with their ground-truth labels.

Task: <Task description>
Features: <Feature descriptions>
Examples: <Serialized training examples>

$$\text{Serialize}(\mathbf{x}^i, \mathbf{y}^i, F) =$$
$$\text{“ } f_1 \text{ is } \mathbf{x}_1^i. \text{ ... } f_d \text{ is } \mathbf{x}_d^i. \text{ Answer: } \mathbf{y}^i \text{ ”}$$

Data	Task description
Adult	Does this person earn more than 50000 dollars per year? Yes or no?
Bank	Does this client subscribe to a term deposit? Yes or no?
Blood	Did the person donate blood? Yes or no?
Car	How would you rate the decision to buy this car? Unacceptable, acceptable, good or very good?
Communities	How high will the rate of violent crimes per 100K population be in this area. Low, medium, or high?
Credit-g	Does this person receive a credit? Yes or no?
Diabetes	Does this patient have diabetes? Yes or no?
Heart	Does the coronary angiography of this patient show a heart disease? Yes or no?
Myocardial	Does the myocardial infarction complications data of this patient show chronic heart failure? Yes or no?
Cultivars	How high will the grain yield of this soybean cultivar. Low or high?
NHANES	Predict this person's age group from the given record. Senior or non-senior?
Sequence-type	What is the type of following sequence? Arithmetic, geometric, fibonacci, or collatz?
Solution-mix	Given the volumes and concentrations of four solutions, does the percent concentration of the mixed solution over 0.5? Yes or no?

Prompt design for extracting rules.

- Guide the problem-solving process to mimic how an expert human might approach it.
  - Basic information description: Essential information for solving the problem.
  - Reasoning instruction: Enhance the LLM's reasoning by providing guidance.
    - Introductory sentence similar to the chain-of-thought approach.
    - **Step 1**: LLM is encouraged to infer the causal relationship.
    - **Step 2**: LLM uses example demonstrations and the information of the first step to deduce rules for each class.

Let's first understand the problem and solve the problem step by step.

Step 1. Analyze the causal relationship or tendency between each feature and task description based on general knowledge and common sense within a short sentence.

Step 2. Based on the above examples and Step 1's results, infer 10 different conditions per answer, following the format below. The condition should make sense, well match examples, and must match the format for [condition] according to value type.

Prompt design for extracting rules.

- Guide the problem-solving process to mimic how an expert human might approach it.
  - Basic information description: Essential information for solving the problem.
  - Reasoning instruction: Enhance the LLM's reasoning by providing guidance.
    - Introductory sentence similar to the chain-of-thought approach.
    - **Step 1**: LLM is encouraged to infer the causal relationship.
    - **Step 2**: LLM uses example demonstrations and the information of the first step to deduce rules for each class.

Step 1. The relationship between each feature and the task description:

- Age: There is a tendency for older age to be associated with a higher risk of heart disease.
- Sex: There may be a difference in the prevalence of heart disease between males and females, with males potentially having a higher risk.
- ChestPainType: Certain types of chest pain, such as NAP and ASY, may be indicative of heart disease.
- RestingBP: Higher resting blood pressure levels may be associated with a higher risk of heart disease.
- Cholesterol: Higher levels of serum cholesterol may be associated with a higher risk of heart disease.
- FastingBS: Fasting blood sugar levels above 120 mg/dl may be indicative of heart disease.
- RestingECG: Abnormal resting electrocardiogram results, such as ST and LVH, may be indicative of heart disease.
- MaxHR: Lower maximum heart rate achieved may be associated with a higher risk of heart disease.
- ExerciseAngina: The presence of exercise-induced angina may be indicative of heart disease.
- Oldpeak: Higher values of oldpeak (ST depression) may be associated with a higher risk of heart disease.
- ST_Slope: The slope of the peak exercise ST segment may provide information about the presence of heart disease.

Prompt design for extracting rules.

- Guide the problem-solving process to mimic how an expert human might approach it.
  - Basic information description: Essential information for solving the problem.
  - Reasoning instruction: Enhance the LLM's reasoning by providing guidance.
  - Response instruction: Guide the LLM on structuring its response.

Format for Response:
10 different conditions for class [Class name]:
- [Condition]

...

Format for [Condition]:
For the categorical variable only,
- [Feature] is in [List of categories]

For the numerical variable only,
- [Feature] (> or >= or < or <=) [Value]
- [Feature] is within range of [Value_start, Value_end]

LLMs for Tabular Data: FeatLLM

Prompt design for extracting rules.

- Guide the problem-solving process to mimic how an expert human might approach it.

Step 2. Inferred conditions for each answer class:

10 different conditions for class “no”:
- Age is within range of [40, 60]
- Sex is in [F]
- ChestPainType is in [ATA, NAP]
- RestingBP (< 140)
- Cholesterol (< 200)
- FastingBS is in [0]
- RestingECG is in [Normal]
- MaxHR (> 140)
- ExerciseAngina is in [N]
- Oldpeak (< 1.0)

10 different conditions for class “yes”:
- Age is within range of [50, 70]
- Sex is in [M]
- ChestPainType is in [ASY, TA]
- RestingBP (> 140)
- Cholesterol (> 200)
- FastingBS is in [1]
- RestingECG is in [ST, LVH]
- MaxHR (< 150)
- ExerciseAngina is in [Y]
- Oldpeak (> 1.0)

Parsing rules for feature generation.

- Utilize the rules to create new binary features.
  - Created for each class, indicating whether the sample satisfies the rules associated with that class.

Conditions:
- Age is within range of [40, 60]
- Sex is in [F]
- ChestPainType is in [ATA, NAP]
- RestingBP (< 140)
- Cholesterol (< 200)
- FastingBS is in [0]
- RestingECG is in [Normal]
- MaxHR (> 140)
- ExerciseAngina is in [N]
- Oldpeak (< 1.0)

```
<start>
def extracting_features_no(df_input):

    df_output = pd.DataFrame()
    df_output['Age'] = df_input['Age'].apply(lambda x: 1 if x >= 40 and x <= 60 else 0)
    df_output['Sex'] = df_input['Sex'].apply(lambda x: 1 if x == 'F' else 0)
    df_output['ChestPainType'] = df_input['ChestPainType'].apply(lambda x: 1 if x in ['ATA', 'NAP'] else 0)
    df_output['RestingBP'] = df_input['RestingBP'].apply(lambda x: 1 if x < 140 else 0)
    df_output['Cholesterol'] = df_input['Cholesterol'].apply(lambda x: 1 if x < 200 else 0)
    df_output['FastingBS'] = df_input['FastingBS'].apply(lambda x: 1 if x == 0 else 0)
    df_output['RestingECG'] = df_input['RestingECG'].apply(lambda x: 1 if x == 'Normal' else 0)
    df_output['MaxHR'] = df_input['MaxHR'].apply(lambda x: 1 if x > 140 else 0)
    df_output['ExerciseAngina'] = df_input['ExerciseAngina'].apply(lambda x: 1 if x = 'N' else 0)
    df_output['Oldpeak'] = df_input['Oldpeak'].apply(lambda x: 1 if x < 1.0 else 0)

    return df_output
<end>
```

Inferring class likelihood.

- A simple method to measure the class likelihood of the sample is to count how many rules of each class it satisfies.
- However, not all rules carry the same importance.
  - FeatLLM learns this importance using a linear model without bias.

$$\mathrm{logit}_k^i = \max(\mathbf{w}_k, 0) \cdot \mathbf{z}_k^i,$$
$$\mathbf{p}^i = \mathrm{Softmax}([\mathrm{logit}_1^i, ..., \mathrm{logit}_c^i]).$$

Ensembling with bagging.

- Repeatedly execute the entire process to create multiple models to make the final prediction via ensemble.
    - The high temperature for LLM inference.
    - Randomize the order of few-shot demonstrations.
    - Bagging to select a subset of features or instances for each trial.

What are the advantages of the ensemble approach?

- Even if the LLM generates incorrect rules, other trials can compensate.
    - LLM’s self-consistency: Rules commonly inferred across multiple trials are more likely to be accurate.
- Address the limitation of LLM’s prompt size.

FeatLLM consistently ranks as the top performer or secures the second place.

Data	Shot	LogReg	XGBoost	SCARF	TabPFN	STUNT	In-context	TABLET	TabLLM	Ours
Adult	4	72.10±12.30	50.00±0.00	58.34±15.42	60.89±23.28	67.43±29.61	77.51±5.24	75.29±12.24	83.57±2.69	**86.68±0.86**
	8	76.02±3.37	59.19±6.92	72.42±8.95	70.42±9.96	82.16±6.93	79.30±2.89	77.56±7.56	83.52±4.30	**87.89±0.06**
	16	75.20±5.10	60.68±13.92	75.63±9.56	70.34±9.96	80.57±10.93	79.50±4.57	79.74±5.64	83.23±2.45	**87.54±0.50**
Bank	4	63.70±3.87	50.00±0.00	58.53±5.49	63.19±11.60	56.34±12.82	61.38±1.30	58.11±6.29	62.51±8.95	**70.45±3.69**
	8	72.52±3.21	58.78±10.54	55.28±11.88	62.81±7.84	63.01±8.78	69.57±13.35	69.08±6.00	63.19±5.79	**75.85±6.66**
	16	77.51±3.09	70.34±5.86	65.81±1.79	73.79±2.21	69.85±0.95	69.76±8.55	69.40±11.28	63.73±6.43	**78.41±1.08**
Blood	4	56.79±26.02	50.00±0.00	56.22±21.00	58.72±19.16	48.57±6.04	56.30±12.43	56.45±15.45	55.87±13.49	**68.34±7.48**
	8	68.51±5.16	59.97±1.36	65.77±5.00	66.30±10.01	60.00±4.84	58.99±10.12	56.37±11.56	66.01±9.25	**70.37±3.23**
	16	68.30±6.16	63.28±7.62	66.27±5.04	64.14±6.80	54.76±4.53	56.59±5.21	60.62±4.13	65.14±7.55	**70.07±5.19**
Car	4	62.38±4.13	50.00±0.00	62.52±3.80	58.14±4.15	61.32±3.83	62.47±2.47	60.21±4.81	**85.82±3.65**	72.69±1.52
	8	72.05±1.20	64.00±3.57	72.23±2.59	63.95±4.35	67.86±0.49	67.57±3.44	65.53±8.00	**87.43±2.56**	73.26±1.46
	16	82.42±4.13	72.26±4.43	75.77±2.71	71.35±5.33	75.56±2.88	76.94±3.04	74.02±1.01	**88.65±2.63**	79.43±1.24
Credit-g	4	52.68±4.46	50.00±0.00	48.92±4.60	54.00±7.34	48.80±6.76	52.99±4.08	54.33±6.54	51.90±9.40	**55.94±1.10**
	8	55.52±8.88	52.22±4.90	55.26±3.92	52.58±11.27	54.50±8.25	52.43±4.36	52.90±5.79	56.42±12.89	**57.42±3.10**
	16	58.26±5.17	56.23±4.37	59.22±11.38	58.91±8.04	57.63±7.58	55.29±4.80	51.65±4.02	**60.38±14.03**	56.60±2.22
Diabetes	4	57.09±18.84	50.00±0.00	62.35±7.48	56.28±13.01	64.22±6.78	71.71±5.31	63.96±3.32	70.42±3.69	**80.28±0.75**
	8	65.52±13.18	50.86±22.03	64.69±13.33	69.08±9.68	67.39±12.92	72.21±2.07	65.47±3.95	64.30±5.88	**79.38±1.66**
	16	73.44±0.52	65.69±6.54	71.86±3.16	73.69±3.21	73.79±6.48	71.64±5.05	66.71±0.76	67.34±2.79	**80.15±1.35**
Heart	4	70.54±3.83	50.00±0.00	59.38±3.42	67.33±15.29	**88.27±3.32**	60.76±4.00	68.19±11.17	59.74±4.49	75.66±4.59
	8	78.12±10.59	55.88±3.98	74.35±6.93	77.89±2.34	**88.78±2.38**	65.46±3.77	69.85±10.82	70.14±7.91	79.46±2.16
	16	83.02±3.70	78.62±7.14	83.66±5.91	81.45±5.05	**89.13±2.10**	67.00±7.83	68.39±11.73	81.72±3.92	83.71±1.88
Cultivars	4	53.45±10.79	50.00±0.00	46.99±6.33	49.80±15.90	**57.10±8.66**	51.38±2.48	54.28±3.73	54.39±5.61	55.63±5.24
	8	56.22±11.87	52.60±6.31	51.76±9.99	54.72±9.35	**57.26±9.52**	51.68±4.43	51.48±3.85	52.86±6.13	56.97±5.08
	16	**60.35±4.23**	56.87±2.50	57.06±9.27	54.92±8.32	60.09±7.64	54.31±6.12	57.44±3.53	56.97±2.22	57.19±5.30

Ablation study.

- Tuning: Omitting the weight-tuning process of the linear model.
  - The benefit becomes higher when the number of shots increases.
  - When there is a large amount of data, accurate estimation of the importance of rules becomes feasible.
- Ensemble: Omitting the ensemble process.
- Description: Omitting the feature description.
- Reasoning: Omitting the Step 1 process in the reasoning instruction part.
  - The benefit becomes higher when the number of shots is small.
  - The efficient utilization of prior knowledge of LLM becomes crucial.

Shot	FeatLLM	-Tuning	-Ensemble	-Description	-Reasoning
4	75.7	-1.41±1.00	-5.39±0.81	-1.76±1.06	-5.03±1.96
8	77.3	-2.72±0.93	-6.96±1.40	-1.20±0.33	-3.55±0.81
16	78.4	-2.57±0.73	-6.65±1.18	-0.26±0.31	-1.50±0.87
32	80.3	-5.75±1.19	-7.38±1.34	-0.29±0.58	-2.42±1.15
64	81.4	-4.88±1.40	-6.09±0.96	-0.70±0.54	-1.71±0.47
Avg	78.6	-3.47±0.51	-6.49±0.51	-0.84±0.28	-2.84±0.53

Dealing with the scarcity of labeled data: Learning transferable knowledge.

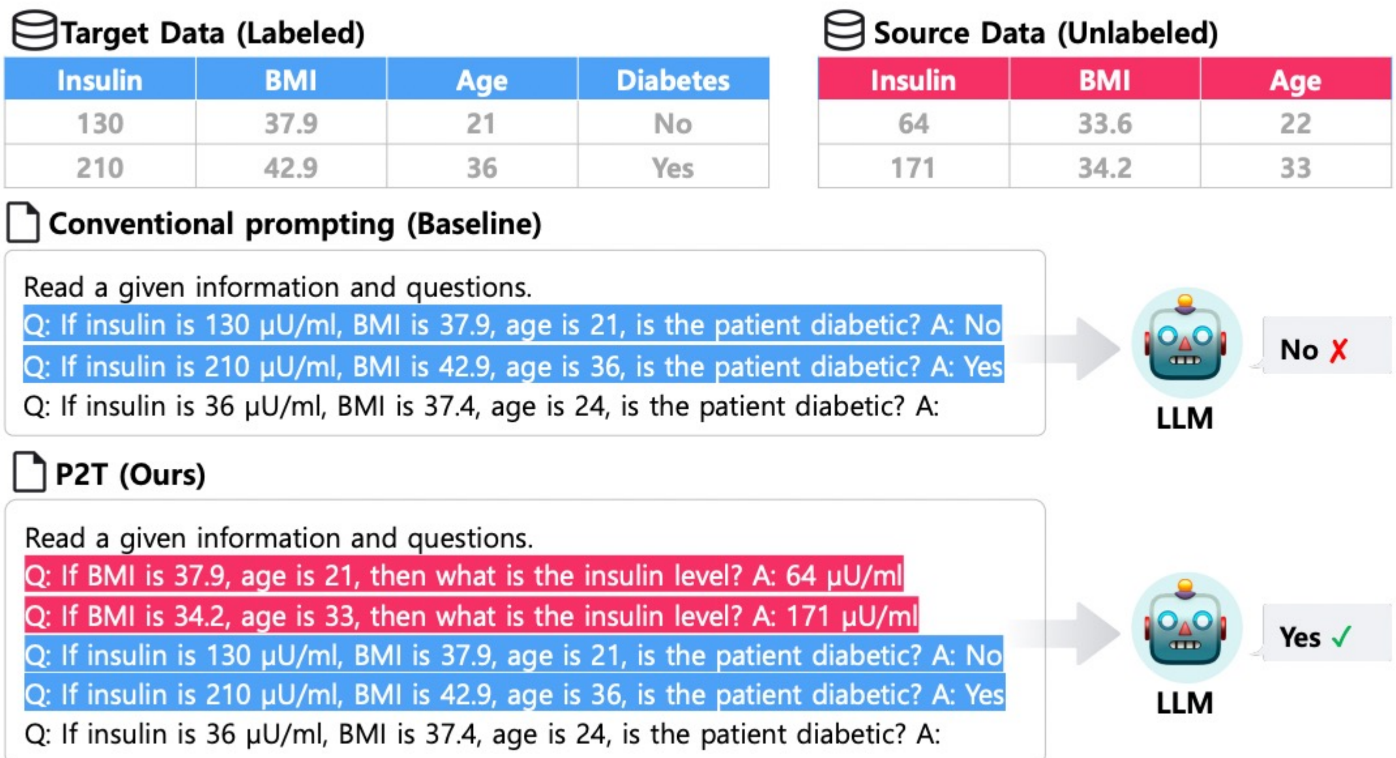
- However, tables are inherently heterogeneous.
  - They contain different columns and feature spaces.

  → Makes transfer learning difficult!

Nam et al. (2024): LLMs can be tabular transfer modules.

- P2T uses LLM to extract transferable knowledge from the source dataset and use it as in-context samples.
  - P2T constructs pseudo-demonstration to be highly relevant to the actual target task.

- **Step 1**: Prompt LLM to determine which column feature is most important for the target task.
- **Step 2**: Create pseudo-demonstrations that describe the task where the selected column feature is the target, and the remaining ones are input.
- **Step 3**: Finally, P2T prompts the LLM with the created pseudo-demonstrations with few-shot labeled demonstrations.



P2T is effective for zero-shot classification.

- The advantage of using LLMs is that they can answer in a zero-shot manner.
- P2T framework can improve the performance of zero-shot prediction.
  - By transferring knowledge from unlabeled and heterogeneous datasets.

Target dataset	Source dataset	Method	Accuracy (↑)
Adult	✗	zero-shot	68.00
	Credit-R	**P2T (Ours)**	70.00
	Electricity	**P2T (Ours)**	72.00
	Unlabeled Adult	**P2T (Ours)**	**74.00**
Credit-g	✗	zero-shot	46.00
	Credit-A	**P2T (Ours)**	62.00
	Unlabeled Credit-g	**P2T (Ours)**	**68.00**
Heart-c	✗	zero-shot	60.00
	Diabetes	**P2T (Ours)**	**65.00**
	Unlabeled Heart-c	**P2T (Ours)**	63.33
Breast	✗	zero-shot	41.07
	Haberman	**P2T (Ours)**	58.93
	Unlabeled Breast	**P2T (Ours)**	**62.50**

Table 1: **Test accuracy (%) on various zero-shot learning scenarios.** Both unlabeled dataset and heterogeneous dataset improves the zero-shot test accuracy of the target dataset. Bold indicates the highest accuracy, and underlined indicates the second highest accuracy.

P2T significantly and consistently improves the few-shot prediction performance utilizing unlabeled data.

- Transfer source: Unlabeled data of the same dataset.
- P2T yields the highest score in all 12 datasets in the 1-shot classification.
- P2T yields the highest score in 11 datasets in the 5-shot classification.

Dataset	LR	kNN	CatBoost	VIME	STUNT	LIFT-ICL	**P2T (Ours)**
# shot = 1							
Breast	61.23	61.88	57.64	57.38	53.04	66.43	**68.93**±6.13
TAE	37.35	37.26	34.29	37.87	36.87	30.97	**43.23**±7.07
Hamster	51.07	51.00	51.87	51.53	51.73	48.00	**58.67**±5.58
Customers	61.34	63.81	64.12	62.48	65.14	70.45	**74.32**±6.15
Pollution	63.67	63.67	63.58	63.33	63.00	58.33	**65.00**±3.73
Diabetes	57.61	58.56	58.60	56.95	61.08	62.60	**68.44**±5.02
Car	36.95	31.51	32.33	34.51	36.48	69.13	**71.40**±1.79
BTC	51.60	51.54	53.02	51.13	52.71	60.40	**62.27**±9.05
Haberman	52.81	52.81	52.82	51.55	53.82	60.32	**61.29**±5.59
Caesarian	62.50	62.50	56.63	60.38	60.06	55.00	**63.75**±5.23
VC	53.76	53.77	54.00	56.34	62.11	70.00	**70.64**±0.89
Salaries	59.52	58.18	58.45	66.55	70.26	45.53	**71.06**±1.97
Average	54.12	53.87	53.11	54.17	55.53	58.10	**64.92**

Dataset	LR	kNN	CatBoost	VIME	STUNT	LIFT-ICL	**P2T (Ours)**
# shot = 5							
Breast	61.21	62.33	57.63	60.89	61.30	67.86	**72.85**±1.96
TAE	43.42	44.65	39.71	42.84	40.77	35.48	**45.81**±1.44
Hamster	51.60	54.53	56.33	52.80	52.87	58.67	**64.00**±7.60
Customers	60.82	64.92	81.40	66.07	66.44	78.41	**83.18**±0.95
Pollution	73.33	72.83	70.58	75.50	70.92	65.00	**76.67**±3.73
Diabetes	64.19	67.32	64.94	64.29	69.88	69.20	**71.44**±2.26
Car	53.29	49.62	46.96	52.37	51.73	70.81	**72.08**±1.03
BTC	58.03	55.71	56.43	55.83	54.11	67.73	**69.33**±1.76
Haberman	53.92	53.40	55.35	53.45	54.85	62.26	**64.84**±2.88
Caesarian	69.56	64.31	66.25	64.88	66.75	65.00	**80.00**±2.80
VC	61.66	61.65	68.00	62.65	66.66	70.65	**70.97**±1.98
Salaries	70.87	71.38	66.38	74.82	**76.86**	55.65	75.06±1.70
Average	60.16	60.22	60.83	60.53	61.10	63.89	**70.52**

P2T consistently benefits from heterogeneous data sources.

- Transfer source: Heterogeneous data.
- As tabular data is transformed into natural language, LLMs can automatically understand the relations between different features from their descriptions.

Target	Source	Method	Number of samples from a source dataset (N) $N=0$	$N=2$	$N=4$	$N=6$	$N=8$	$N=10$
Adult	Credit-R	LR†	54.00	69.33	69.33	66.67	62.00	57.33
		kNN†	54.00	72.00	72.00	57.33	57.33	57.33
		CatBoost†	56.00	54.67	60.00	61.33	51.33	49.33
		LIFT-ICL	69.33	25.33	35.33	52.00	60.00	43.33
		P2T (Ours)	**74.67**	**75.33**	**76.00**	**77.33**	**79.33**	**80.00**
	Electricity	LR†	54.00	54.67	50.67	50.00	37.33	60.00
		kNN†	54.00	57.33	42.67	42.67	28.00	42.67
		CatBoost†	56.00	50.00	50.67	48.67	45.33	58.00
		LIFT-ICL	69.33	60.67	64.67	63.33	58.67	54.00
		P2T (Ours)	**74.67**	**80.00**	**76.00**	**78.67**	**80.00**	**81.33**
Credit-g	Credit-A	LR†	52.67	49.33	48.00	34.00	42.00	38.67
		kNN†	52.67	**58.67**	41.33	41.33	41.33	24.00
		CatBoost†	**55.33**	46.67	41.33	46.67	40.67	44.00
		LIFT-ICL	42.67	49.17	48.17	45.83	46.00	48.67
		P2T (Ours)	55.00	54.50	**58.67**	**59.33**	**59.33**	**60.67**

Using the identified target highly correlated with the target task consistently outperforms random targets.

- Carefully constructing pseudo-demonstrations designed to be highly relevant to the target task is a key factor in enabling transfer learning via prompting.
- Moreover, LLM is better than conventional methods for identifying the most correlated features.

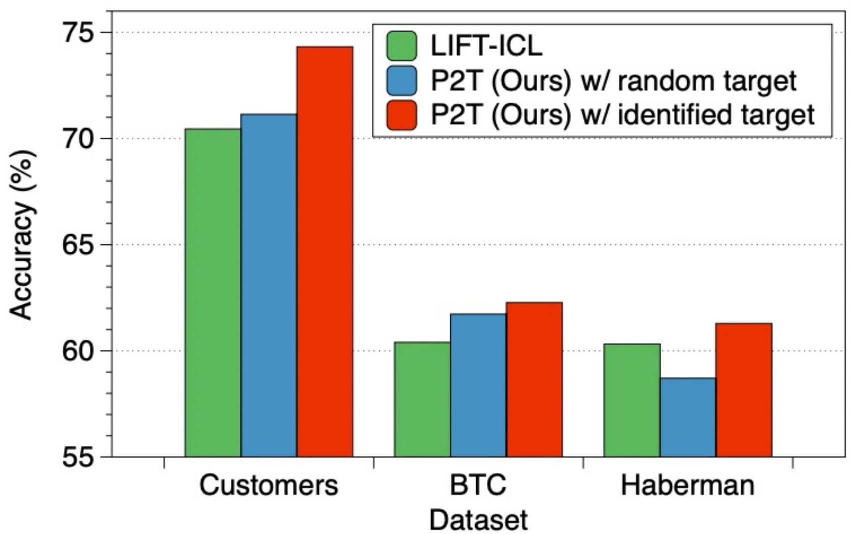


Figure 3: Ablation study that varies the column features used as targets for pseudo-demonstrations.

Dataset	CatBoost	**LLM (Ours)**
Customers	69.32±4.17	**74.32**±3.47
BTC	62.00±8.65	**62.27**±9.05
Haberman	60.97±5.75	**61.29**±5.59

Table 6: **LLM's superiority for correlation identification.** We report 1-shot test accuracy (%) using unlabeled samples as transfer source. We report the average accuracy over 5 different seeds.

Can better performance be achieved by P2T using a more advanced model?

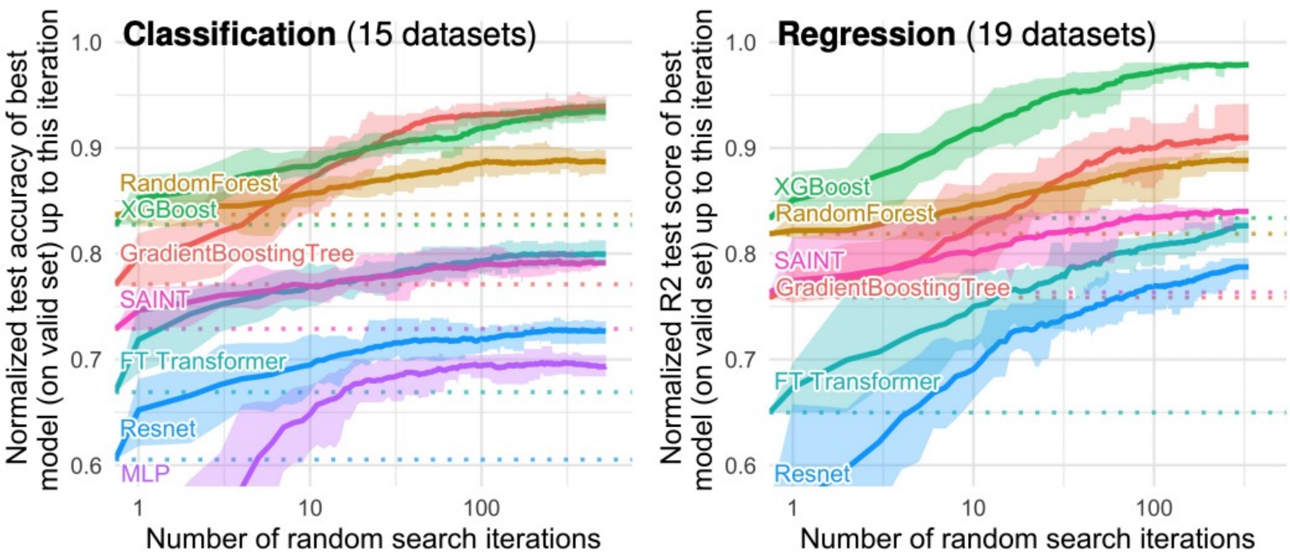
- P2T performs better with advanced LLMs.
- As LLMs continue to advance, improved performance by P2T framework is expected with future models.

	Customers		BTC		Haberman	
Method	GPT-3.5	GPT-4	GPT-3.5	GPT-4	GPT-3.5	GPT-4
LIFT-ICL	70.45	88.18	60.40	61.73	60.32	67.74
P2T (Ours)	**74.32**	**89.77**	**62.27**	**63.47**	**61.29**	**70.32**

Table 4: **Comparison between GPT-3.5 and GPT-4.** We report 1-shot test accuracy (%) using unlabeled samples as transfer source. We report the average accuracy over 5 different seeds. The bold denotes the highest average score.

Are learned representations always useful for tabular learning?

- Deep learning approaches are arguably known to be less effective.
- Tree-based approaches using raw column features often outperform deep learning models.



It would be very useful if one could generate informative raw column features.

- Practitioners often focus on augmenting raw column features by using feature engineering methods.
  - Remains ambiguity in defining the space over which to search for candidate features.
  - Often rely solely on validation scores to select good features, neglecting valuable feedback from past experiments.

Nam et al. (2024): The optimization of a good generation rule.

- However, optimizing the column feature generator is not straightforward because it is a non-differentiable problem.
  - The search space is very large.

OCTree [Nam et al., 2024] leverages an LLM to find an effective column generator.

- LLM can optimize a variety of non-differentiable problems with prompts that describe the optimization task in language.
- The extensibility of injecting linguistic context (e.g., column names like “Gender” and values like “Female”).

Two main challenges:

- The rule for generating column features is often non-differentiable.

→ **Use an LLM as an optimizer.**

- LLM’s input prompt size limit makes it difficult to provide full training samples in the prompts.

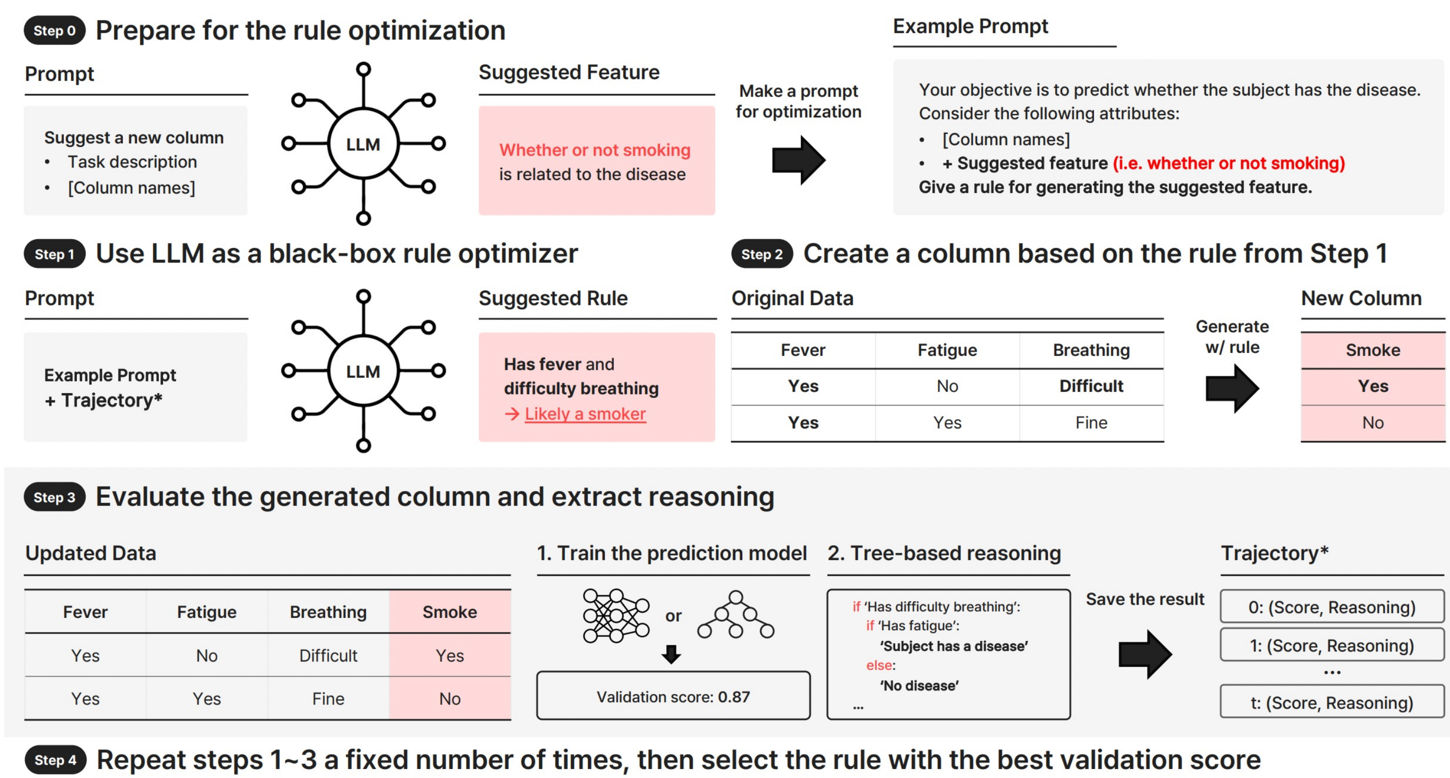
→ We design a **novel decision tree reasoning**, i.e., akin to compression of the training dataset.

Step 1: Generate the column name of a novel feature.

Step 2: Initialize the optimization process.

Step 3: Optimize the rule using decision tree reasoning.

Step 4: Optimize the rule with a fixed number of iterations and select the rule with the highest validation score.



OCTree consistently improves on the best-performing baselines.

- LLM generates a logical rule in natural language.
  - Since the logical rule is easily converted to Python code, we prompt the LLM to convert it.

Method	LLM	Tesla†	Enefit†	Disease*	Clinical*	Academic*
XGBoost [11]						
Baseline	-	6.61	8.00	28.09$\pm$7.9	46.27$\pm$5.0	14.15$\pm$0.6
OCTree	Llama 2	5.56 (15.9%)	8.00 (0.0%)	26.19$\pm$7.2 (6.8%)	45.07$\pm$4.1 (2.6%)	14.11$\pm$0.5 (0.3%)
OCTree	GPT-4o	**5.48 (17.1%)**	**7.82 (2.3%)**	**25.72**$\pm$6.6 **(8.4%)**	**43.75**$\pm$4.4 **(5.4%)**	**13.74**$\pm$0.1 **(2.9%)**
MLP [31]						
Baseline	-	7.41	33.53	38.10$\pm$3.6	41.77$\pm$1.7	14.41$\pm$0.8
OCTree	Llama 2	5.23 (29.4%)	29.99 (10.6%)	32.86$\pm$5.7 (13.7%)	39.80$\pm$2.3 (4.7%)	14.26$\pm$0.7 (1.0%)
OCTree	GPT-4o	**5.01 (32.4%)**	**21.68 (35.3%)**	**30.95**$\pm$5.8 **(18.8%)**	**39.25**$\pm$0.5 **(6.0%)**	**14.22**$\pm$0.5 **(1.3%)**
HyperFast [32]						
Baseline	-	N/A	N/A	28.57$\pm$10.0	43.64$\pm$1.1	14.67$\pm$0.7
OCTree	Llama 2	N/A	N/A	28.10$\pm$9.2 (1.6%)	**41.45**$\pm$1.7 **(5.0%)**	**14.49**$\pm$0.5 **(1.2%)**
OCTree	GPT-4o	N/A	N/A	**27.14**$\pm$3.8 **(5.0%)**	42.00$\pm$1.5 (3.8%)	**14.49**$\pm$0.5 **(1.2%)**

Rule in Natural Language

If the student's **father's qualification is less than 18,**
they **are not an international student,**
and their **previous qualification is greater than 20,**
then predict 'Yes' for 'Part-time job holder'.
Otherwise, predict 'No'.

Rule in Code

```
def predicting_part_time_job_holder(data):
    father_qualification = data[0] # Select features
    international= data[3]
    previous_qualification = data[4]
    # Define rule
    if father_qualification < 18 \
        and international == 0 \
        and previous_qualification > 20:
        return 'Yes'
    else:
        return 'No'
```

In practice, language descriptions are not always available.

- E.g., feature names and values are changed to meaningless symbols in many financial datasets for confidentiality.
- OCTree uses arithmetic rules as feature generators.

Listing 12 Optimized arithmetic rules on the bank-marketing dataset.

```
x8 = np.cos(np.pi * x1) * np.sqrt(x2) + np.tan(x3) * np.exp(x4) − np.sin(x5) + np.log(1 + x6) − np.abs(x7 − 0.5)
```

Listing 13 Optimized arithmetic rules on the phoneme dataset.

```
x7 = np.sin(x1) * np.log(x2 + 1) + np.sqrt(x3) − (x4 * np.exp(x5)) + (np.tan(x6) * *2)
x8 = np.tan(np.sin(np.sqrt(x1)) * np.log(x2 + 1)/(np.exp(x3) + np.sqrt(x4) + np.log(x5 + 1) + 1))
x9 = ...
```

In practice, language descriptions are not always available.

- E.g., feature names and values are changed to meaningless symbols in many financial datasets for confidentiality.
- OCTree uses arithmetic rules as feature generators.
  - Even in this case, OCTree is beneficial for improving the baseline models.
  - Superiority comes from the optimization capability of LLMs, using decision tree reasoning as explicit feedback.

Dataset	XGBoost [11]		MLP [31]		HyperFast [32]	
	Baseline	**OCTree (Ours)**	Baseline	**OCTree (Ours)**	Baseline	**OCTree (Ours)**
electricity	8.32±0.0	**6.65**±0.1 **(20.1%)**	15.64±0.3	**14.82**±0.4 **(5.2%)**	15.25±0.5	**14.70**±0.5 **(3.6%)**
rl	23.61±0.8	**19.32**±0.4 **(18.2%)**	32.03±4.2	**28.30**±1.7 **(11.6%)**	33.77±1.3	**33.50**±1.2 **(0.8%)**
compass	22.91±0.5	**18.89**±0.4 **(17.6%)**	27.41±1.0	**26.78**±0.1 **(2.3%)**	25.74±0.6	**24.91**±1.1 **(3.2%)**
covertype	9.10±0.2	**7.96**±0.0 **(12.5%)**	8.73±0.4	**8.25**±0.3 **(5.5%)**	9.86±1.6	**9.21**±1.3 **(6.6%)**
phoneme	10.89±0.5	**10.15**±0.7 **(6.8%)**	12.06±0.8	**10.98**±0.6 **(9.8%)**	**10.55**±0.7	10.57±0.9 (N/I)
kddCup09	19.86±1.1	**19.07**±1.4 **(4.0%)**	**24.30**±0.3	**24.30**±1.6 **(0.0%)**	25.75±0.7	**24.46**±1.1 **(5.0%)**
pol	1.69±0.2	**1.62**±0.2 **(4.0%)**	1.37±0.3	**1.27**±0.3 **(7.3%)**	1.70±0.4	**1.55**±0.2 **(8.8%)**
Magic	14.25±0.3	**13.75**±0.4 **(3.5%)**	14.60±0.2	**14.50**±0.0 **(0.7%)**	14.95±0.2	**14.34**±0.5 **(4.1%)**
california	9.45±0.6	**9.13**±1.0 **(3.4%)**	11.91±0.3	**11.37**±0.1 **(4.5%)**	11.75±0.7	**11.02**±0.6 **(6.2%)**
house_16H	11.66±0.5	**11.32**±0.2 **(3.0%)**	13.07±0.2	**12.54**±0.6 **(4.1%)**	12.77±0.3	**12.29**±0.4 **(3.8%)**
eye_movements	35.06±0.7	**34.17**±2.0 **(2.6%)**	40.03±1.2	**39.86**±1.9 **(0.4%)**	41.33±1.5	**40.29**±1.7 **(2.5%)**
road-safety	21.14±0.0	**20.65**±0.1 **(2.3%)**	22.17±0.4	**21.87**±0.1 **(1.4%)**	24.54±0.3	**24.07**±0.4 **(1.9%)**
kdd_ipums_la	10.89±1.0	**10.69**±1.0 **(1.8%)**	13.13±1.3	**11.72**±1.5 **(10.7%)**	16.15±0.3	**13.55**±1.4 **(16.1%)**
MiniBooNE	5.48±0.2	**5.42**±0.1 **(1.2%)**	9.69±0.3	**7.35**±0.2 **(24.1%)**	6.61±0.4	**6.54**±0.2 **(1.1%)**
credit	22.02±0.3	**21.78**±0.3 **(1.1%)**	24.43±0.6	**23.23**±0.7 **(4.9%)**	25.06±1.1	**24.30**±1.8 **(3.0%)**
Higgs	27.95±0.7	**27.91**±0.2 **(0.1%)**	29.43±0.4	**28.80**±0.2 **(2.1%)**	30.04±0.2	**29.73**±0.5 **(1.0%)**
jannis	**20.61**±0.1	20.64±0.1 (N/I)	**22.28**±0.1	22.51±0.1 (N/I)	24.29±0.4	**23.65**±0.3 **(2.6%)**
wine	**19.11**±3.3	19.18±3.9 (N/I)	**21.53**±3.1	21.59±1.4 (N/I)	**19.18**±2.7	19.31±2.2 (N/I)
bank-marketing	**20.09**±0.3	20.31±0.6 (N/I)	21.11±0.4	**21.09**±0.4 **(0.1%)**	**21.25**±1.0	21.66±0.8 (N/I)

OCTree outperforms state-of-the-art automatic feature engineering methods.

- Furthermore, OCTree in combination with OpenFE further improves the performance.

Prediction model	Baseline	AutoFeat [23]	OpenFE [17]	**OCTree (Ours)**	**OCTree† (Ours)**
XGBoost [11]	18.30$\pm$0.3	18.24$\pm$0.3 (1.3%)	17.79$\pm$0.2 (2.8%)	17.45$\pm$0.5 (4.6%)	**16.85**$\pm$0.3 **(7.9%)**
MLP [31]	20.88$\pm$0.1	20.60$\pm$0.5 (1.3%)	20.12$\pm$0.5 (3.6%)	19.91$\pm$0.4 (4.6%)	**19.41**$\pm$0.5 **(7.0%)**

Ablation study of the proposed components.

- The rules for introducing new column features are optimized even without using explicit decision trees for feedback.
- One can get even better performance by providing the decision tree as feedback to the LLM.

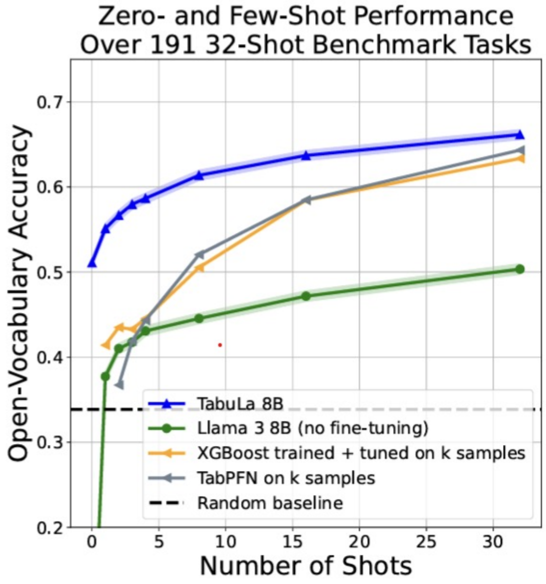
Gen. Feat.	DT Reasoning	Disease*	Clinical*	electricity†	kddCup09†
-	-	28.09$\pm$7.9	46.27$\pm$5.0	8.32$\pm$0.0	19.86$\pm$1.1
✓	✗	27.62$\pm$8.4 (1.7%)	45.61$\pm$4.1 (1.4%)	6.89$\pm$0.6 (17.2%)	19.47$\pm$1.6 (2.0%)
✓	✓	**26.19**$\pm$7.2 **(6.8%)**	**45.07**$\pm$4.1 **(2.6%)**	**6.65**$\pm$0.1 **(20.1%)**	**19.07**$\pm$1.4 **(4.0%)**

Transfer learning is one of the defining hallmarks of recent foundation models.

- The ability to accurately solve prediction tasks on data it was not trained on.

Gardner et al. (2024): Introduce a new model and dataset for large-scale transfer learning on tabular data.

- TabuLa-8B: A language model for tabular prediction that can solve classification tasks across unseen domains.
    - Outperforms baselines, given a small number of examples, without any fine-tuning.
    - Capable of zero-shot prediction.



Overview.

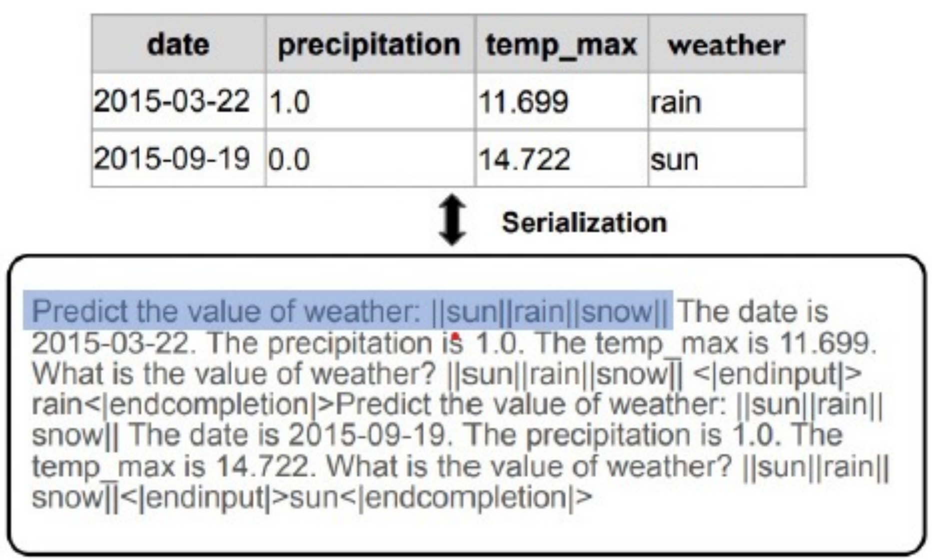
- Overall approach: Fine-tune the pretrained Llama3-8B language model on tabular prediction tasks.

Why Llama3-8B as the starting point?

- It is a high-quality, open-source model trained on over 15T tokens.
- Demonstrates strong performance on a diverse set of downstream tasks.
- Relatively modest size: Makes fine-tuning, inference, and deployment more accessible.

Serialization and tabular language models.

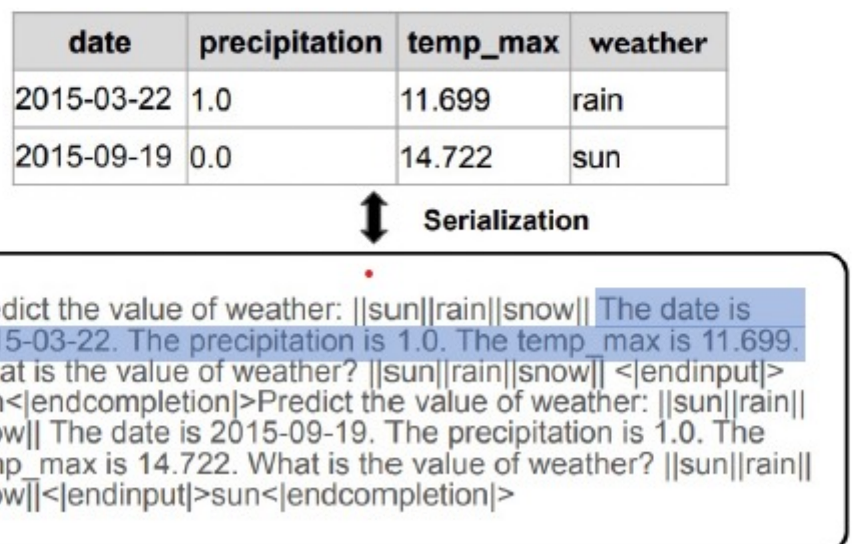
- Serialization: Converting a row of data into text.
  - E.g., “the <key> is <value>”
- Given a row of data from a table, the corresponding serialization has three main parts:
  - A prefix containing a prompt followed by a list of possible label values.



date	precipitation	temp_max	weather
2015-03-22	1.0	11.699	rain
2015-09-19	0.0	14.722	sun

Serialization and tabular language models.

- Serialization: Converting a row of data into text.
  - E.g., “the <key> is <value>”
- Given a row of data from a table, the corresponding serialization has three main parts:
  - A prefix containing a prompt followed by a list of possible label values.
  - The example consists of all key value pairs for the columns used as features.



date	precipitation	temp_max	weather
2015-03-22	1.0	11.699	rain
2015-09-19	0.0	14.722	sun

Serialization and tabular language models.

- Serialization: Converting a row of data into text.
  - E.g., “the <key> is <value>”
- Given a row of data from a table, the corresponding serialization has three main parts:
  - A prefix containing a prompt followed by a list of possible label values.
  - The example consists of all key value pairs for the columns used as features.
  - A suffix prompts the model with a question again, followed by the possible labels.

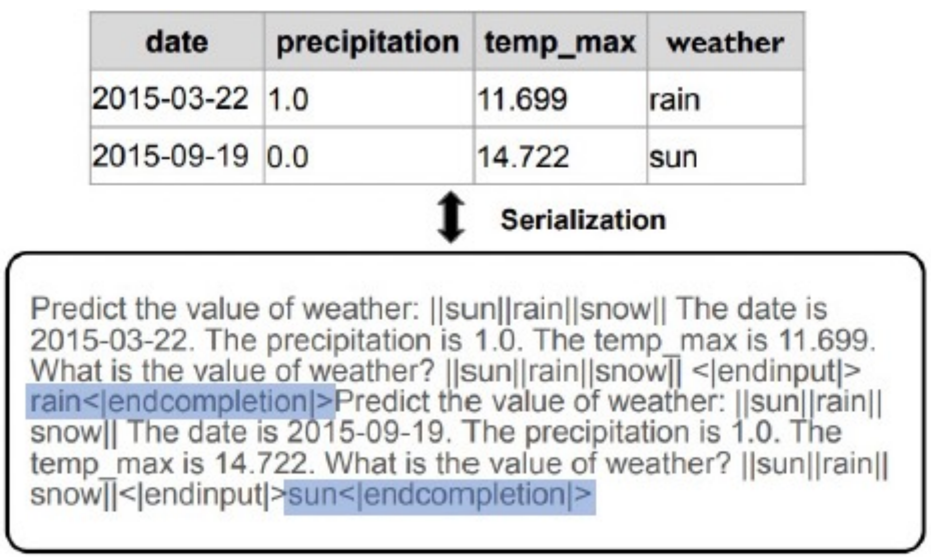
date	precipitation	temp_max	weather
2015-03-22	1.0	11.699	rain
2015-09-19	0.0	14.722	sun

Serialization

Predict the value of weather: ||sun||rain||snow|| The date is 2015-03-22. The precipitation is 1.0. The temp_max is 11.699. What is the value of weather? ||sun||rain||snow|| <|endinput|> rain<|endcompletion|>Predict the value of weather: ||sun||rain|| snow|| The date is 2015-09-19. The precipitation is 1.0. The temp_max is 14.722. What is the value of weather? ||sun||rain|| snow||<|endinput|>sun<|endcompletion|>

Training procedure.

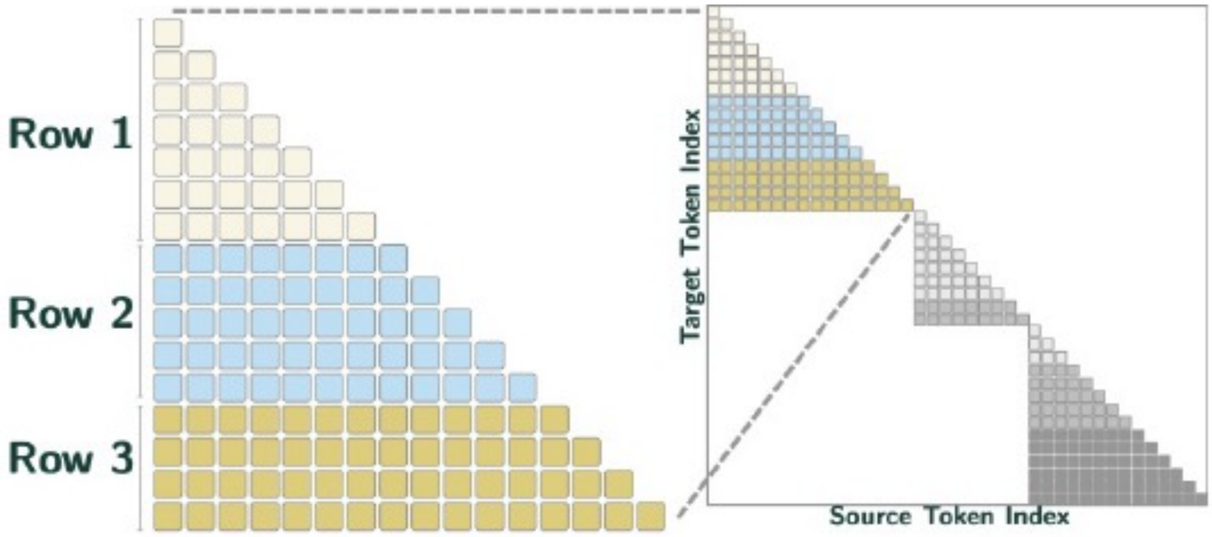
- Train TabuLa-8B using a standard language modeling setup.
  - Minimize the cross-entropy over the sequence of target tokens.
- Only compute loss over the subsequence of target tokens.
  - The tokens start after the <|endinput|> token, up to and including <|endcompletion|>.
  - Focuses training on learning the desired target label.



date	precipitation	temp_max	weather
2015-03-22	1.0	11.699	rain
2015-09-19	0.0	14.722	sun

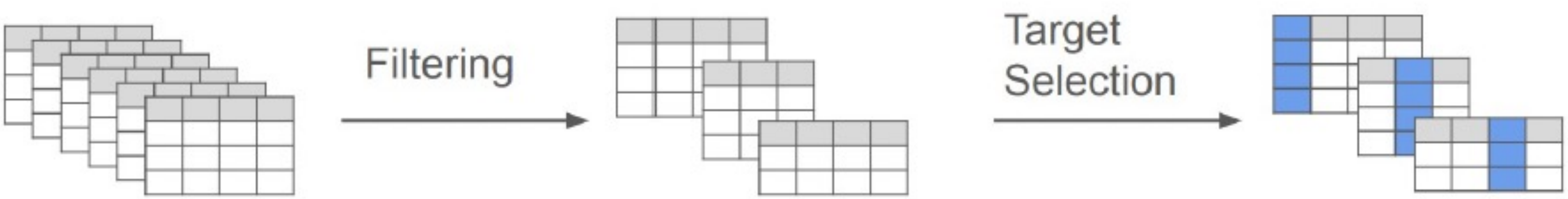
RCTM: Row-Causal Tabular Masking

- An efficient attention masking scheme.
  - Tailored to few-shot tabular prediction.
  - The model is allowed to attend to all previous samples from the same table in the batch.
  - But not to samples from other tables.
- Similar to the in-context pretraining.
  - RCTM has a drastic impact on few-shot performance.



Dataset construction: Original raw data source

- TabLib: Publicly available dataset consisting of 627M tables extracted from Common Crawl and Github.
  - TabLib contains numerous system logs with instructable statistics.
  - Tables of software documentation.
  - Call sheets with personally identifiable information.



Dataset construction: Filtering strategies

- Filtering occurs at three levels: tables, columns, and rows.
  - Remove non-tabular data, e.g., text or PDF.
  - Ensure the safety of chosen tables, e.g., remove PII.
  - Find sources with high semantic content, e.g., remove tables with too many missing values.

Level	Name	Description	Motivation / Hypothesis
Table	English Filtering	Drop where a language ID model score is below a fixed threshold	All downstream benchmark datasets are in English
Table	Schema Heterogeneity	Drop tables where every cell is of the same type	Encourages understanding of mixed data types
Table	Row Count	Drop table with fews than k rows	Anecdotally, many "very small" tables in TabLib are general web-text tables not useful/suitable for ML.
Table	Column Count	Drop tables with fewer than k columns *after column filters are applied*	Exclude tables that lack a reasonable amount of features
Table	Parse Error	Drop tables where the headers suggest there was a parsing error.	These tables are likely the result of bad table detection, and they almost definitely contain low-quality headers.
Table	Drop PII	Drop table where $> x\%$ of the cells match a regex for phone number or email	Don't want to train on PII for privacy reasons. Also not likely to be present in downstream tasks.
Table	Drop Code	Drop table with any cell that has probability $> p$ of containing code.	Lots of the data in TabLib is from Github and other technical documentation. Code is common. Code also confuses the model a lot, apparently due to special characters and whitespace. This code can be unevenly broken/spread across cells due to the tablib parser.
Table	Too many unnamed columns	Drop table if the fraction of "Unnamed: " columns is greater than a threshold.	Discard low-quality data; unnamed columns tend to be of significantly lower quality based on manual data inspection.

Level	Name	Description	Motivation / Hypothesis
Column	Drop Free-Text	Drop columns with long headers ($>$ 256 characters)	The TabLib process used to scrape the tables can result in tables with "headers" that are actually just rows of data. One indicator of this is very long headers (e.g. a text column that ends up as a header).
Column	Drop Numeric	Drop columns with names that are numeric.	TabLib's parsing removed tables with all-numeric headers "for most file formats". This means that (a) some formats were missed, and (b) tables with many numeric headers and even one non-numeric header were still included.
Column	Drop Missing	Drop any column with $> x\%$ values that are None, NaN, whitespace or empty string values	Columns that are mostly missing will waste compute processing headers; empty cells usually won't be informative (although sometimes a header alone can be useful).
Column	English Filtering	Drop any text columns where average probability of English over rows is less than p	Some tables contain English headers and non-English data. All of our downstream data is English. It's hard to assess quality of non-English data.
Column	Drop Constant	Drop columns where all values are the same.	Constant features are not useful for prediction
Row	Drop Missing	Drop any row with too many values that are None, NaN, whitespace or empty string values	Rows with mostly missing data are likely to be uninformative.
Row	Drop Duplicates	Drop duplicate rows	This is non-standard in downstream tasks
Row	Drop PII (regex-based)	Drop any row where PII is detected (phone number, email)	Tables with small numbers of rows containing PII can still pass through the table-level PII filtering.
Row	Drop Code (regex-based)	Drop any row where code is detected.	Tables with small numbers of rows containing code can still pass through the table-level code filtering
Row	Drop $\vert$	Drop any row where any of the values contain $\vert$ symbol	This is exclusively used as an indicator of hierarchy (again, common in technical documentation such as that found on Github). This is a sign that the row of the table isn't self-contained and therefore probably not a candidate for a meaningful prediction task.

Dataset construction: Unsupervised task selection

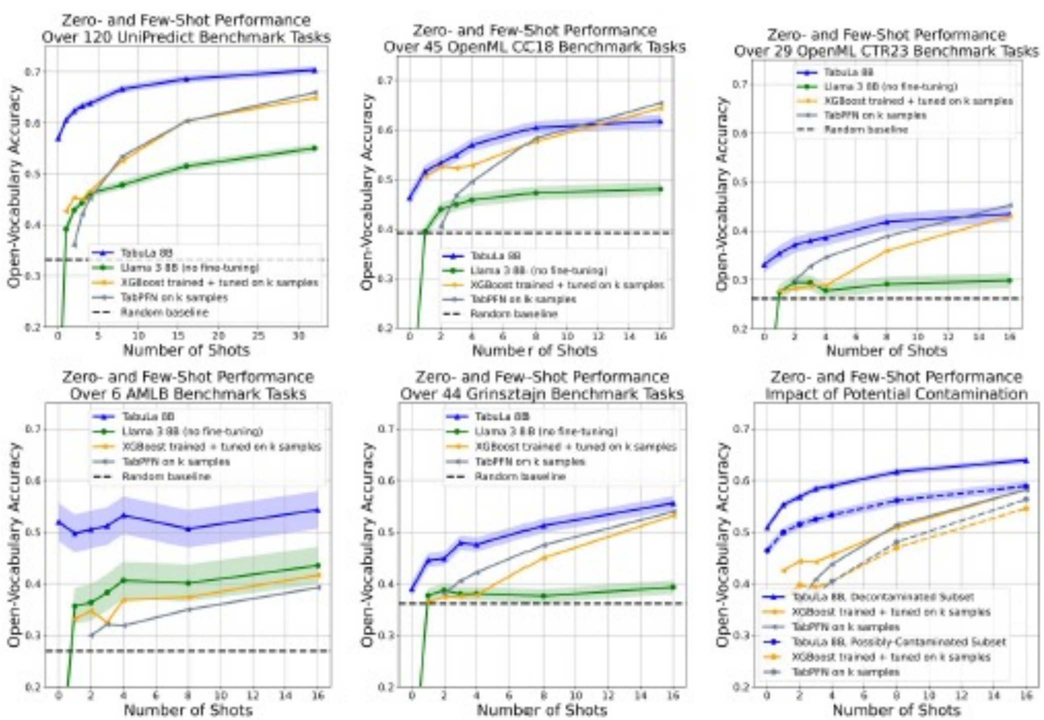
- First, identify a subset of columns that are suitable for prediction according to various heuristics.
  - Exclude if the column name is numeric, it has only one unique value, or it has unique values for every row.
- Then, choose a specific column at random from this set.

The Tremendous TabLib Trawl (T4)

- Total 3.1M tables.
- The dataset contains over 1.6B rows.
  - Approximately 80B Llama 3 tokens.

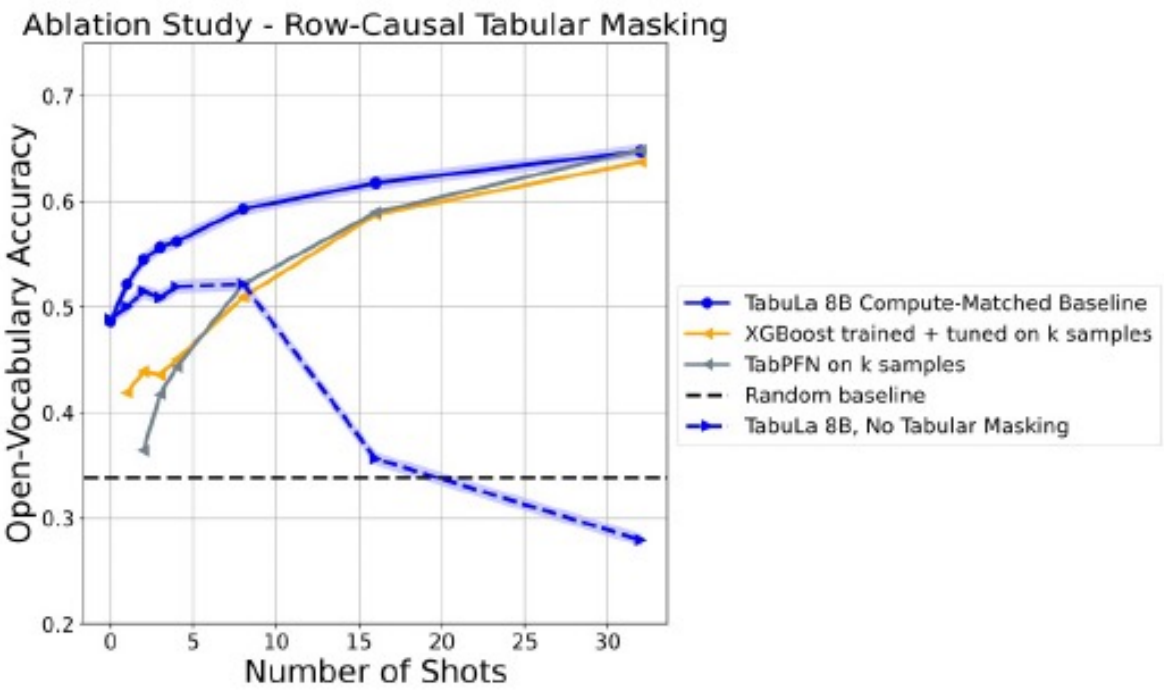
Experiment: Main results.

- TabuLa-8B demonstrates strong transfer performance across a broad range of tasks.
- TabuLa-8B is 50pp more accurate than the base Llama 3 model in the zero-shot regime.
- In the regime of 1 to 32 shots, it outperforms XGBoost and TabPFN.
  - Baselines are directly trained on each specific dataset.



Experiment: Ablation study on RCTM

- Replaced RCTM with a per-sample causal attention mask.
  - The model is not allowed to attend to any samples besides the target sample.
- RCTM improves the models' ability to attend across samples.
  - Removing RCTM deteriorates as the number of shot grows.



AnoLLM [Tsai et al., 2025] leverages LLMs for unsupervised tabular anomaly detection.

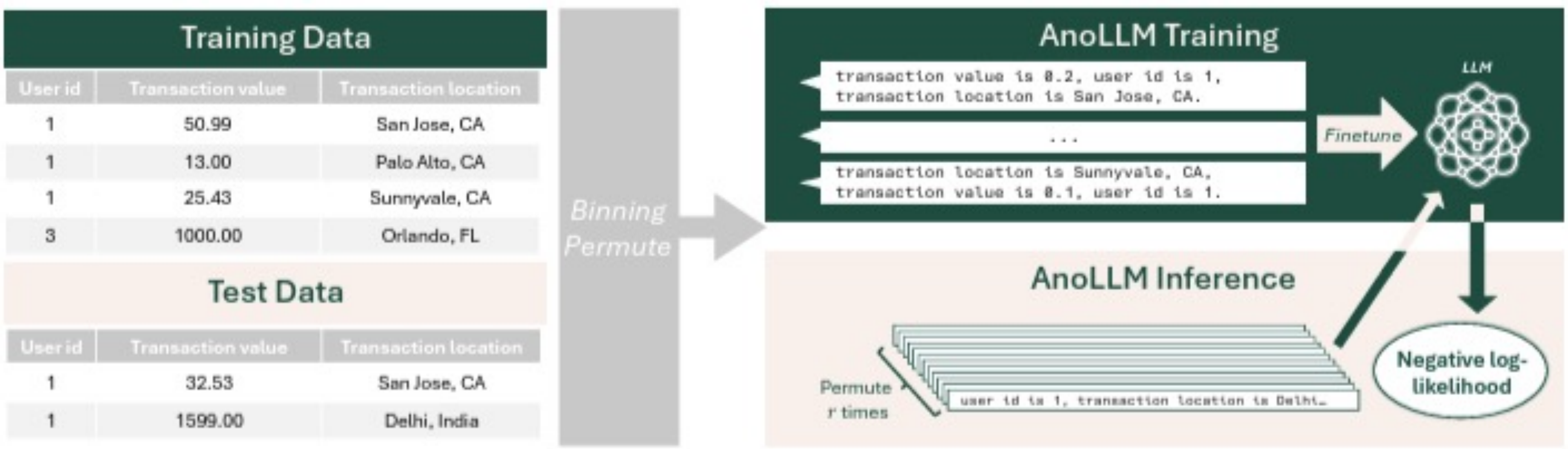
- Challenges:
  - Tabular data does not align well with the linear and sequential nature of LLM inputs.
  - Unsupervised anomaly detection lacks labels, making the ICL framework unfeasible.
  - How should we define the anomaly scores?

AnoLLM is comprised of three phases:

- **Step 1**: Serialize each row of a tabular dataset into a standardized text format.
- **Step 2**: LLM is fine-tuned with the serialized tabular data via next-token-prediction.
  - LLM learns to be a tabular data generator that models the data distribution.
- **Step 3**: Anomaly scores are determined using the negative log likelihood.
  - Higher scores indicates greater surprise by the model when encountering the inputs.

Further details.

- During the preprocessing stage, numerical columns are binned into groups.
- Order of columns is randomly shuffled.
- During inference, anomaly scores are determined by averaging the negative log-likelihood across random permutations of the test data.



Advantages over traditional methods:

- Retains textual and categorical features without heavy feature engineering.
- Handles mixed-type data effectively.
- Uses column permutation to prevent feature ordering bias.

Performance: Achieves SOTA results on six benchmark datasets.

Methods \ Datasets	Fake job posts	Fraud ecommerce	Lympho-graphy	Seismic	Vehicle insurance	20news groups	Average
Classical methods							
Iforest	0.755	0.501	0.673	0.692	0.496	0.623	0.623
PCA	0.724	0.647	0.826	0.692	0.509	0.623	0.670
KNN	0.636	**1**	0.860	0.738	0.524	0.605	0.727
ECOD	0.512	0.755	0.830	0.692	0.509	0.62	0.653
Deep learning based methods							
DeepSVDD	0.561	**1**	0.899	0.713	0.505	0.597	0.713
RCA	0.629	**1**	0.919	0.727	0.531	0.546	0.725
SLAD	0.603	0.998	0.964	0.714	0.556	0.64	0.746
GOAD	0.566	0.998	0.817	0.717	0.512	0.63	0.707
NeuTral	0.548	**1**	0.847	0.681	0.507	0.658	0.707
ICL	0.699	**1**	0.827	0.719	0.501	0.671	0.736
DTE	0.548	**1**	0.909	0.714	0.512	0.6	0.714
REPEN	0.653	**1**	0.808	0.724	0.513	0.574	0.712
AnoLLM							
SmolLM-135M	0.800	**1**	0.968	0.712	**0.569**	**0.766**	0.803
SmolLM-360M	**0.814**	**1**	**0.995**	**0.746**	0.555	0.752	**0.810**

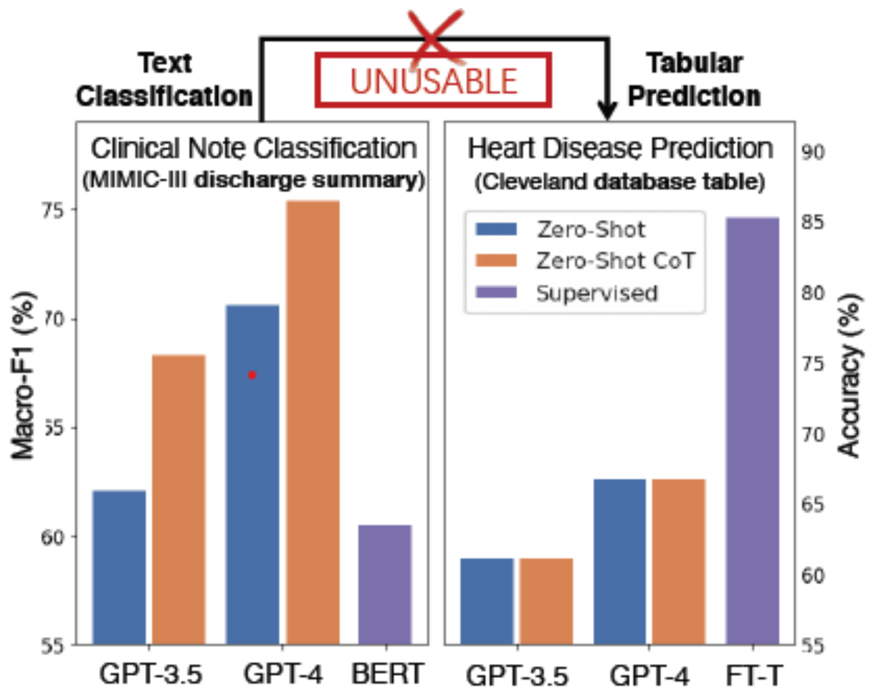
Ablation study: Larger LLMs do not significantly improve performance over smaller models.

- AnoLLM mainly uses SmolLM-135M and SmolLM-360M models.
- Using the 1.7B model does not provide much performance boost.
- This could be because larger models are trained on text data that are not relevant to tabular tasks.

LLM sizes	Mix-typed	ODDS
135M	0.803	**0.884**
360M	0.811	0.865
1.7B	**0.812**	0.861

Problem: LLMs excel in unstructured data tasks but struggle with structured tabular data, especially in medical applications where numerical values dominate.

- LLMs lack numerical sensitivity, making them less effective for tabular data tasks (e.g., disease prediction from lab results).
- Standard prompting techniques (zero-shot, CoT, few-shot) do not significantly improve LLM performance on tabular tasks.



(a) Linguistic prompting gap

SERSAL: Self-Enhancing Refinement via Small Models and LLMs.

- A novel self-prompting method that synergizes small models with LLMs.
- Enhances tabular data prediction in an unsupervised manner.

Propose Method.

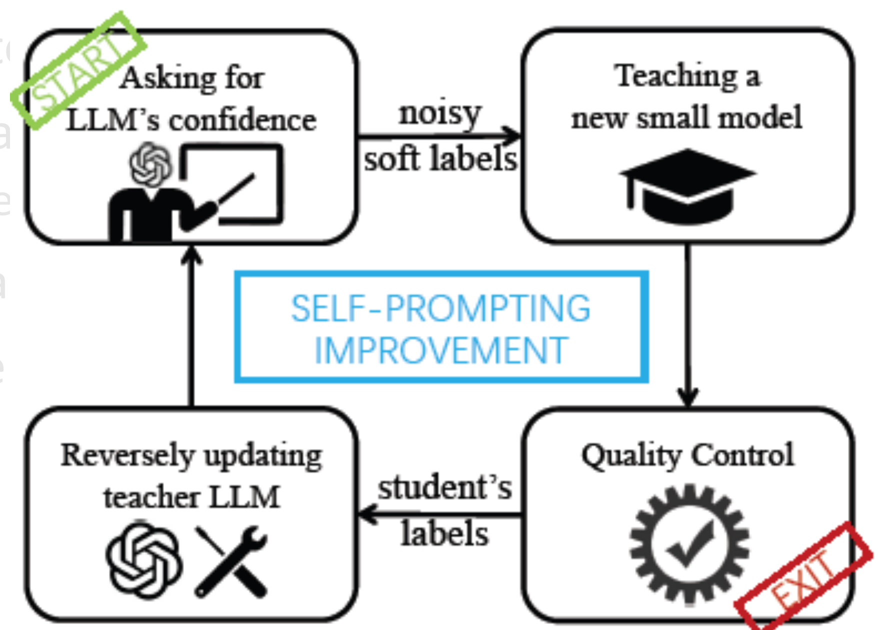
- **Step 1**: Use LLMs to generate soft pseudo-labels (confidence scores).
- **Step 2**: Train a small tabular model using these pseudo-labels.
  - I.e., treating them as noisy annotations.
- **Step 3**: Use the trained small model's predictions to refine (fine-tune) the LLM.
- **Step 4**: Repeat the process iteratively to improve performance.

SERSAL: Self-Enhancing Refinement via Small Models and LLMs.

- A novel self-prompting method that synergizes small models with LLMs.
- Enhances tabular data prediction in an unsupervised manner.

Propose Method.

- **Step 1**: Use LLMs t … e scores).
- **Step 2**: Train a sma … els.
  - I.e., treating the …
- **Step 3**: Use the tra … (fine-tune) the LLM.
- **Step 4**: Repeat the … nce.



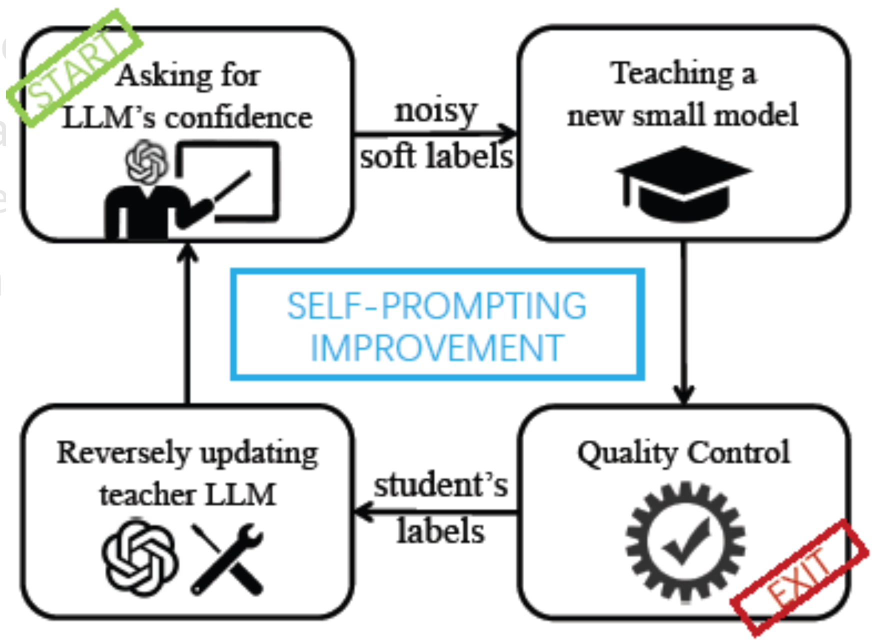
(b) SERSAL loop prompting

SERSAL: Self-Enhancing Refinement via Small Models and LLMs.

- A novel self-prompting method that synergizes small models with LLMs.
- Enhances tabular data prediction in an unsupervised manner.

Propose Method.

- **Step 1**: Use LLMs t … e scores).
- **Step 2**: Train a sma … els.
  - I.e., treating the …
- **Step 3**: Use the tra … (fine-tune) the LLM.
- **Step 4**: Repeat the … nce.



(b) SERSAL loop prompting

Experiment: Consistently outperforms zero-shot and few-shot prompting techniques, approaching fully supervised small model performance.

- LLM-generated high-confidence predictions tend to be reliable.
- Works best when the LLM has some domain knowledge.

	HF	LC	ECD	LI	HE	PID	FH	ST	CO	AN
Random guessing	37.22	40.18	46.25	50.28	62.73	63.24	50.39	41.76	71.55	51.28
FSSM* (supervised FT-T)	88.19	86.61	99.60	78.94	100.00	84.72	66.25	82.98	99.91	99.92
0-shot (GPT-3.5)	71.88	78.87	85.71	76.81	68.51	73.12	60.32	63.01	82.60	90.43
8-shot* (GPT-3.5)	73.65	78.87	**87.68**	76.81	68.51	73.12	58.27	60.85	77.63	87.19
CoT (GPT-3.5)	71.88	78.87	82.36	76.81	68.51	70.83	60.32	63.01	82.60	90.43
TabLLM (GPT-3.5)	76.37	78.87	87.06	78.24	74.39	75.69	61.78	68.48	85.78	89.11
LIFT (GPT-3.5)	78.23	80.69	83.92	73.60	72.57	73.12	60.32	70.92	87.93	90.43
SERSAL (GPT-3.5)	**91.39**	**85.42**	86.40	**79.39**	**85.14**	**78.97**	**63.97**	**76.36**	**96.85**	**98.37**
TabLLM+SERSAL (GPT-3.5)	93.82	85.42	88.39	80.71	89.27	82.54	65.02	81.74	97.51	98.16
SERSAL (GPT-4)	94.18	86.93	92.68	82.51	92.76	82.39	67.14	81.23	97.96	98.82

Experiment: Consistently outperforms zero-shot and few-shot prompting techniques, approaching fully supervised small model performance.

- LLM-generated high-confidence predictions tend to be reliable.
- Works best when the LLM has some domain knowledge.
- Iterative application continuously improves LLM reasoning for tabular tasks.

# Loop	ECD		LI	
	SERSAL	LLM 0-shot	SERSAL	LLM 0-shot
1	86.40	85.71	79.39	76.81
2	87.00	86.42	82.47	80.26
3	89.00	87.81	84.07	82.91

Table of Contents

1. **LLMs for science**
   - General purpose LLMs for science
   - LLMs for Chemistry & Biology
   - LLMs for Mathematics

2. **LLMs for other datasets**
   - Tabular data
   - Time series

3. **LLM agents**
   - Basic concept & Benchmarks
   - Prompting LLMs as agents
   - Optimizing LLMs as agents

LLMs for Time Series: Motivation

Time series forecasting predicts the future from history.

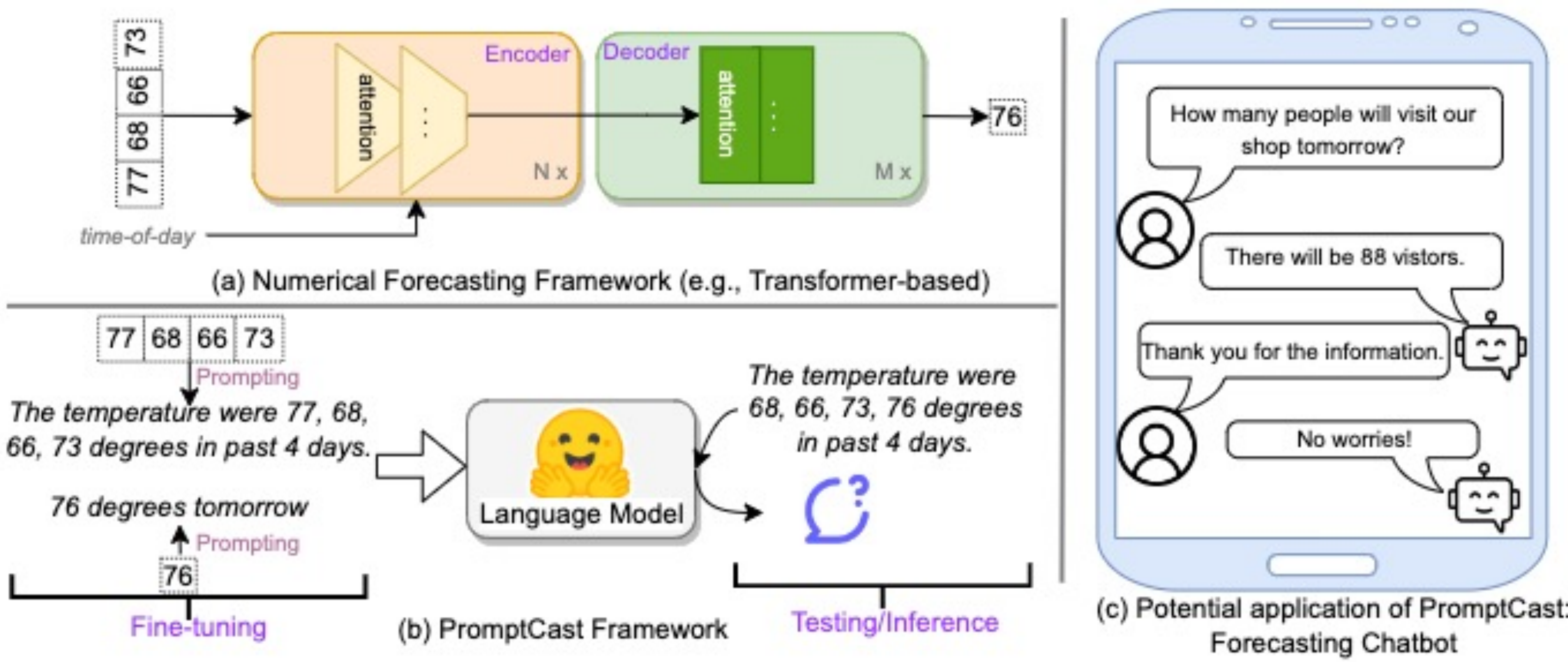
- Challenge:
  - Diverse nature of training data (Different scales, sample rates, missing values, …)
  - Using LLMs: Modality gap between natural language and numerical sequences
- Thus:
  - No large model pre-trained from time series, unlike the image, language domain.
- Simple methods like ARIMA or linear models often outperform DL methods.

Can LLMs be extended beyond language understanding?

- There is no need for fine-tuning; suited for scenarios with limited data.
- Circumvents the extensive time, effort, and domain-specific expertise.

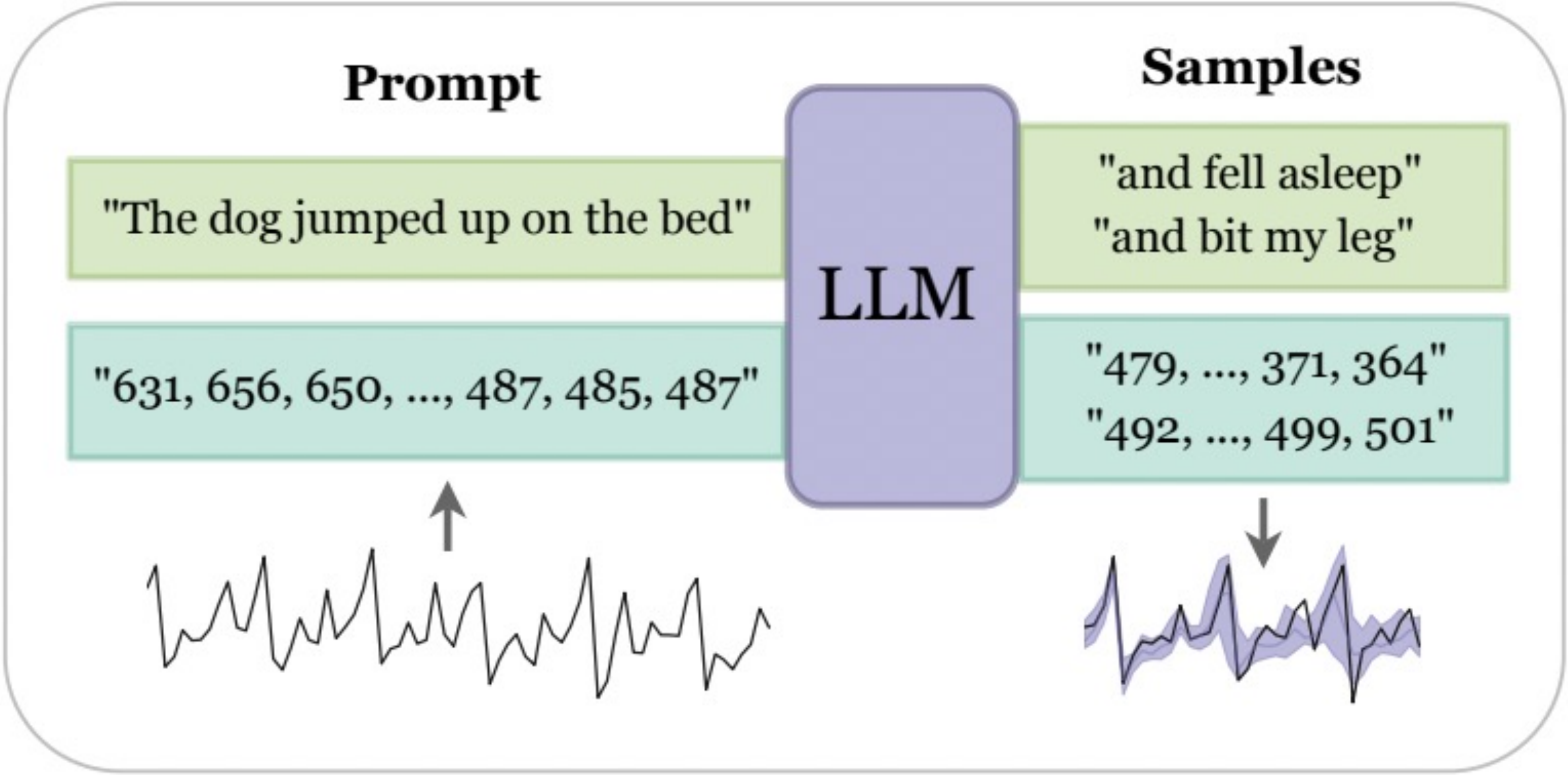
PromptCast [Xue et al., 2023]

- Rephrase time-series data to natural language.
- So that LLM can leverage its linguistic nature.



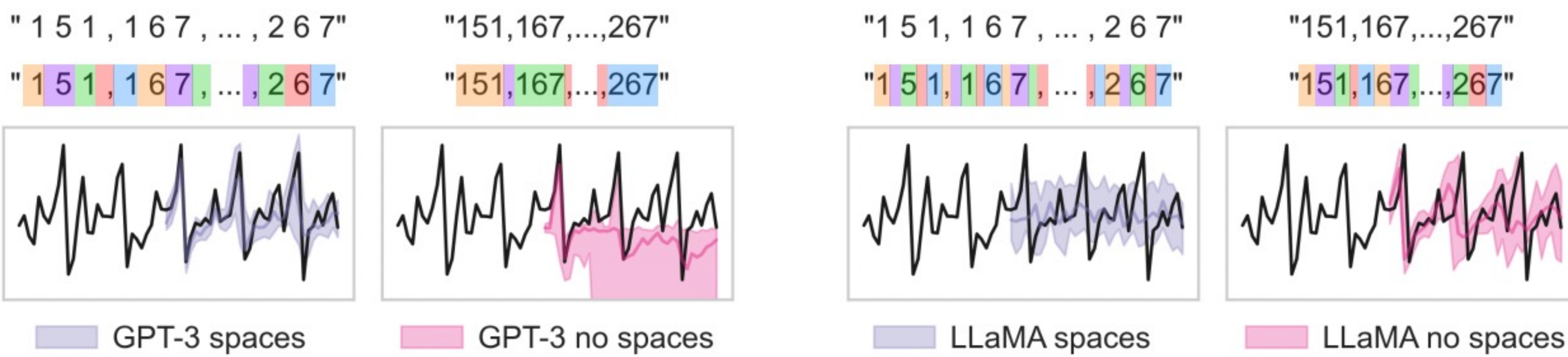
LLMs are zero-shot time series forecasters [Gruver et al., 2023]

- **Time series data.**
  - Recap: Language data U_i is consisted of tokens u_j, $U_i = (u_1, u_2, \ldots, u_j, \ldots, u_{n_i})$.
  - Time series data: Exact same form as language data, but each u_j is numerical.
  - Issue: Details of tokenizing numbers.



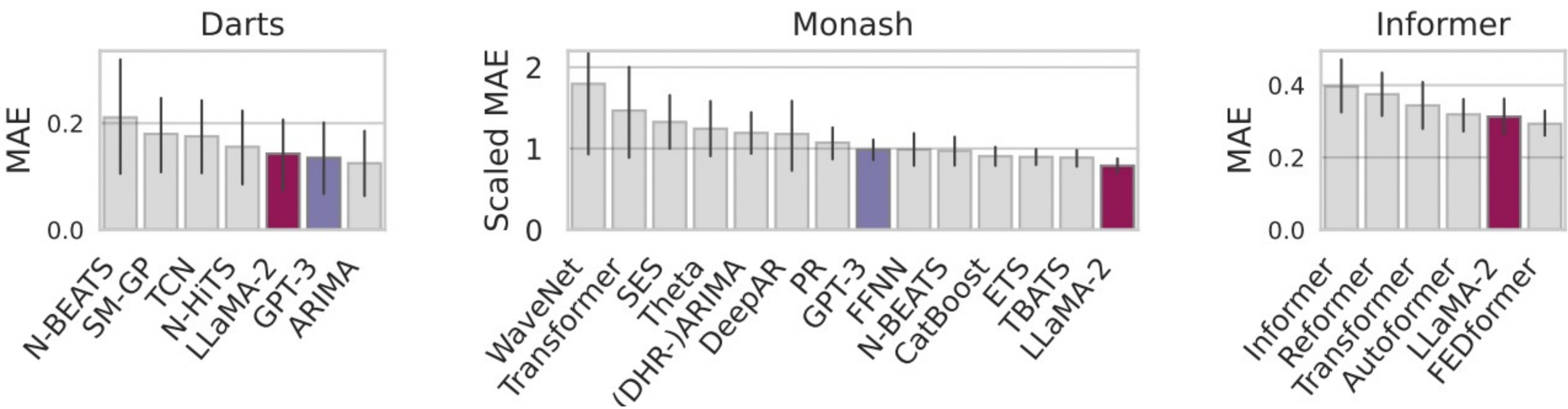
LLMs are zero-shot time series forecasters [Gruver et al., 2023]

- **Tokenization.**
  - Separates the digits with spaces to force a separate tokenization of each digit.
  - Use a comma (“,”) to separate each time step, with 2 digits of precision.
  - Example: 0.123, 1.23, 12.3, 123.0 → “1 2 , 1 2 3 , 1 2 3 0 , 1 2 3 0 0”



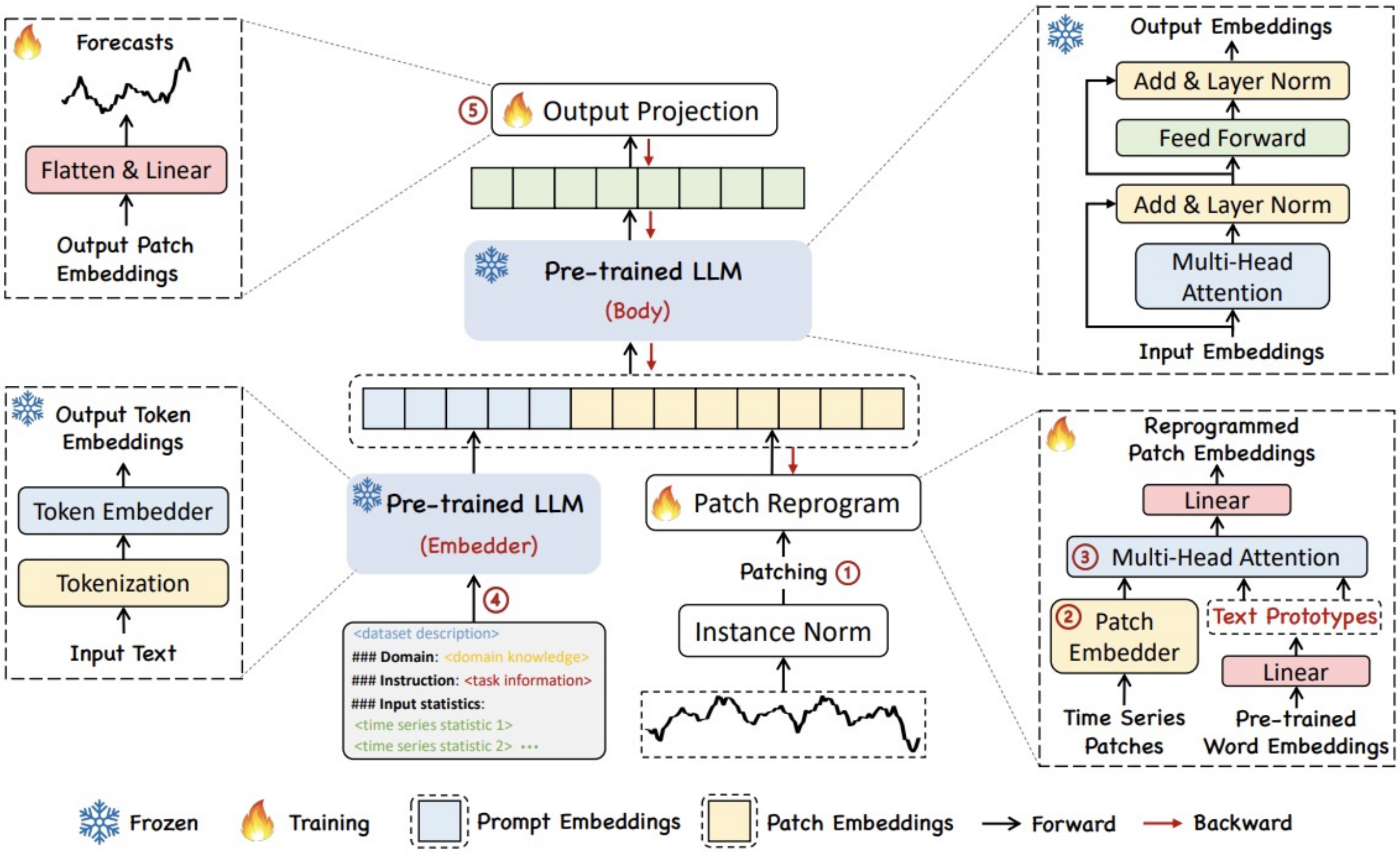
LLMTIME has the best-aggregated performance on several benchmarks.

- Base Model: LLaMA-2, GPT-3
- Note: Baseline methods are usually many-shot, while LLMTIME is zero-shot.
- Predictions from LLMTIME are ranked best or second best on all benchmarks.



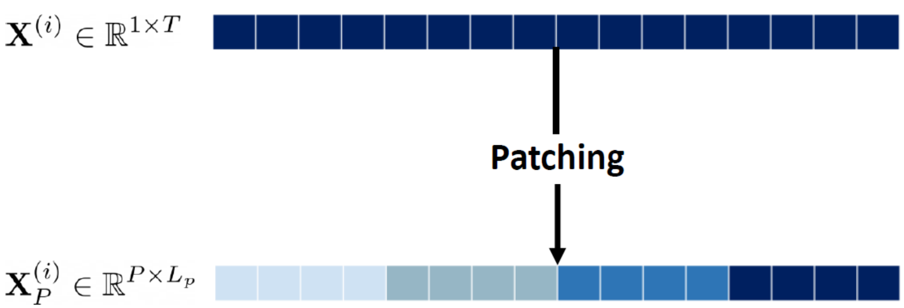
Time-LLM: Time Series Forecasting by Reprogramming LLMs [Jin et al., 2024]

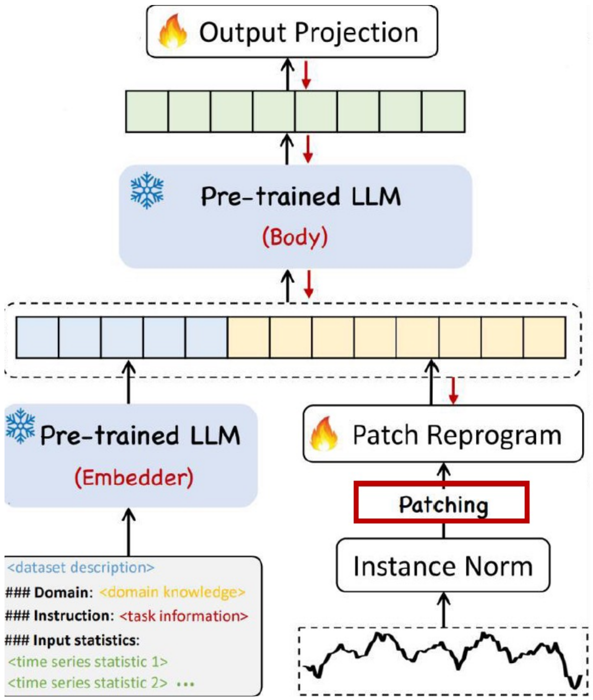
- **Patching & Reprogramming**
  - Align the modalities of time series and natural language



Time-LLM: Time Series Forecasting by Reprogramming LLMs [Jin et al., 2024]

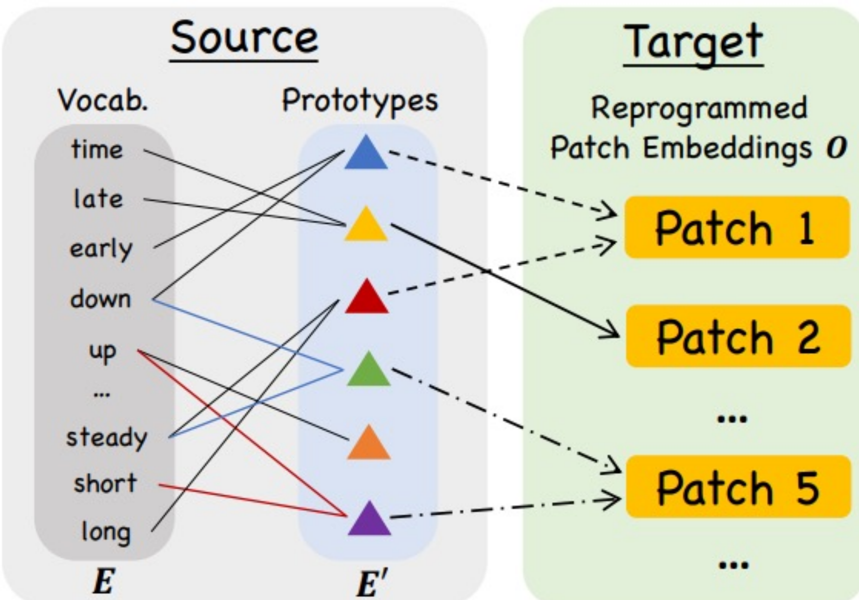
- **Patching**
  - Each (normalized) input channel $\mathbf{X}^{(i)}$ is divided to patches
  - Better at preserving local semantic information
  - Less input tokens leading to less computational cost

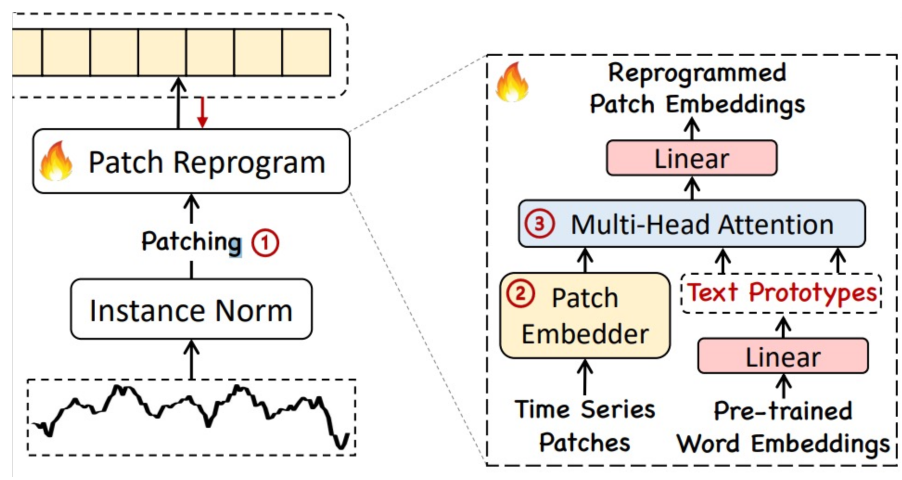




Time-LLM: Time Series Forecasting by Reprogramming LLMs [Jin et al., 2024]

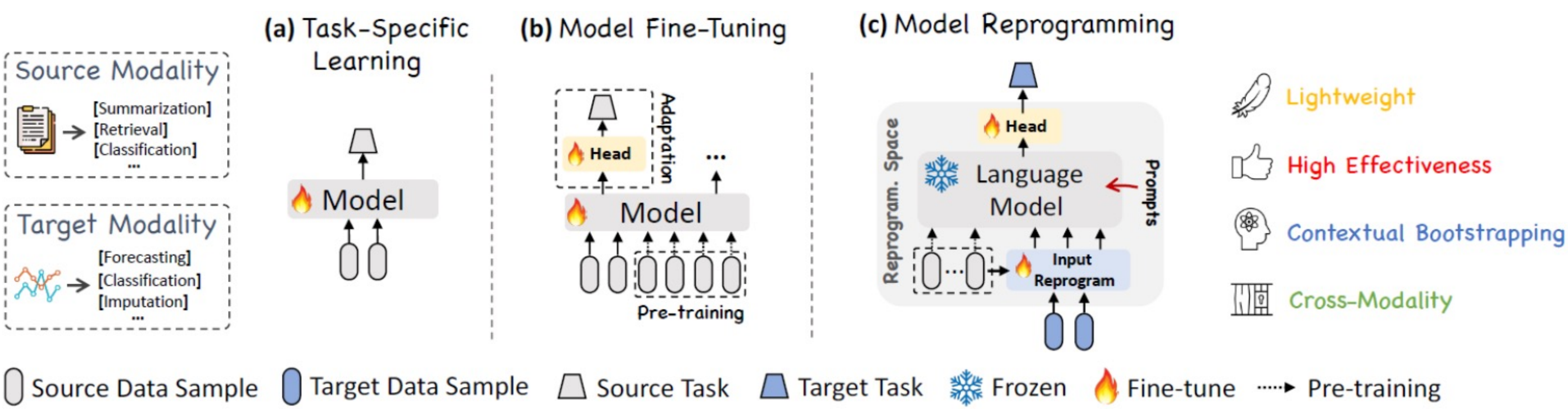
- **Reprogramming**
  - Align TS patch - language using 'Text prototypes'
    - ex)  : steady down,  : short up
  - Multi-head attention for source and target alignment





Time-LLM: Time Series Forecasting by Reprogramming LLMs [Jin et al., 2024]

- **Reprogramming**
  - Efficient compared to task-specific learning & fine-tuning



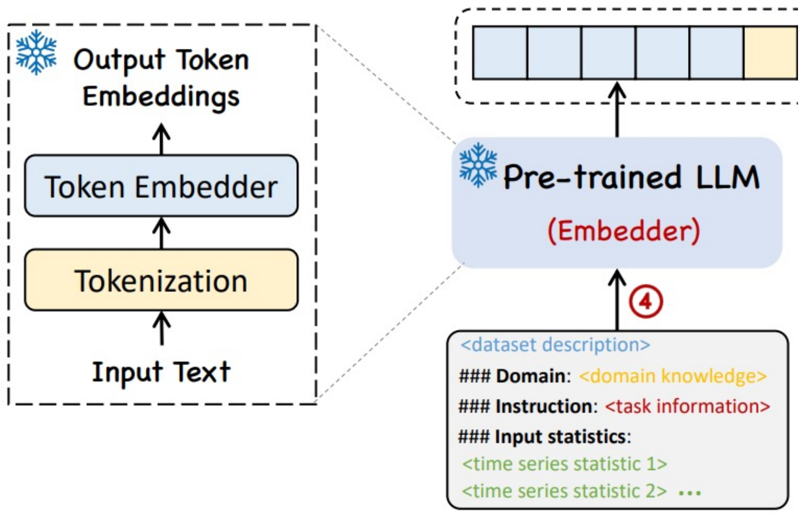
Length		ETTh1-96			ETTh1-336		
Metric		Trainable Param. (M)	Mem. (MiB)	Speed(s/iter)	Trainable Param. (M)	Mem. (MiB)	Speed(s/iter)
Llama (8)	QLoRA	12.60	14767	0.237	12.69	15982	0.335
	Reprogram	5.62	11370	0.184	5.71	13188	0.203
Llama (32)	QLoRA	50.29	45226	0.697	50.37	49374	0.732
	Reprogram	6.39	32136	0.517	6.48	37988	0.632

Time-LLM: Time Series Forecasting by Reprogramming LLMs [Jin et al., 2024]

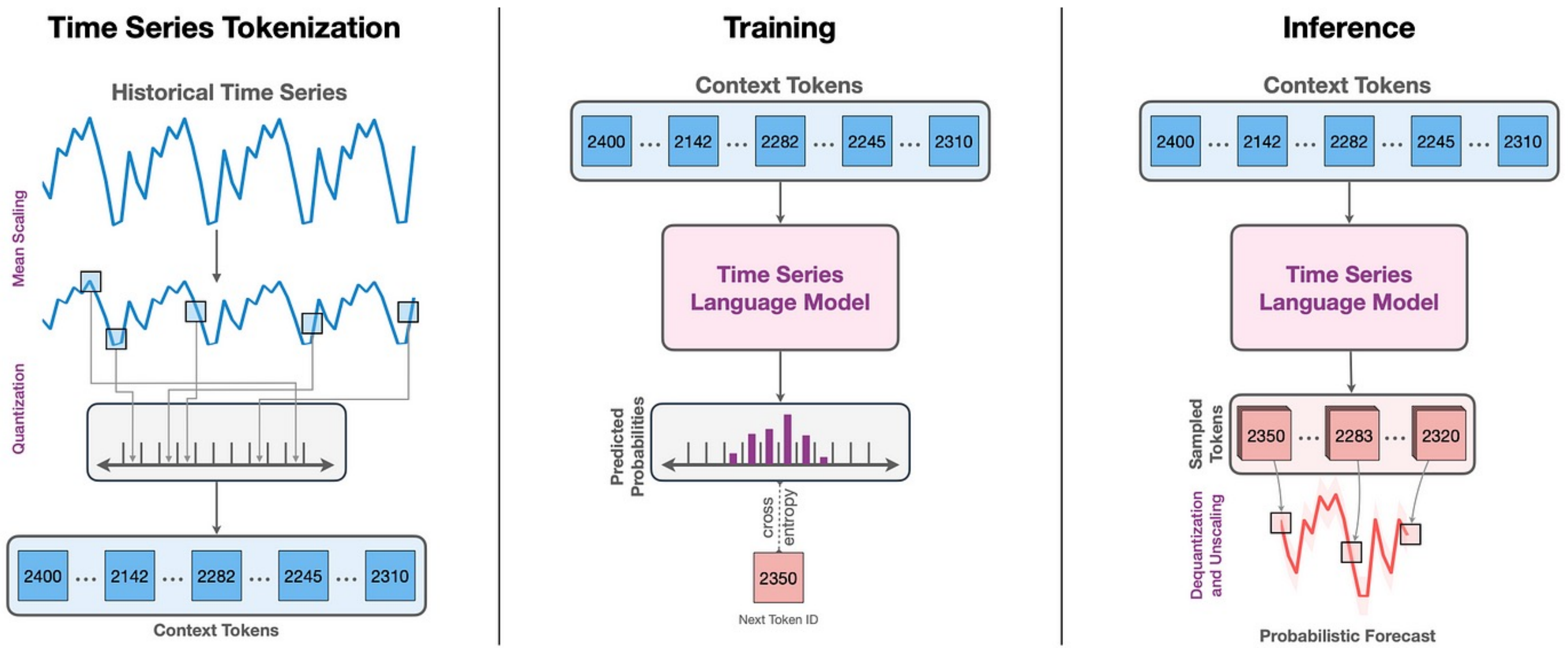
- **Prompt-as-Prefix**
  - Inject prompts with input context to guide the reprogramming of TS data
  - Direct explanation and information about the dataset
    - *Dataset context, Task instruction, input statistics*

The Electricity Transformer Temperature (ETT) indicates the electric power long-term deployment. Each data point consists of the target oil temperature and 6 power load features ... Below is the information about the input time series:

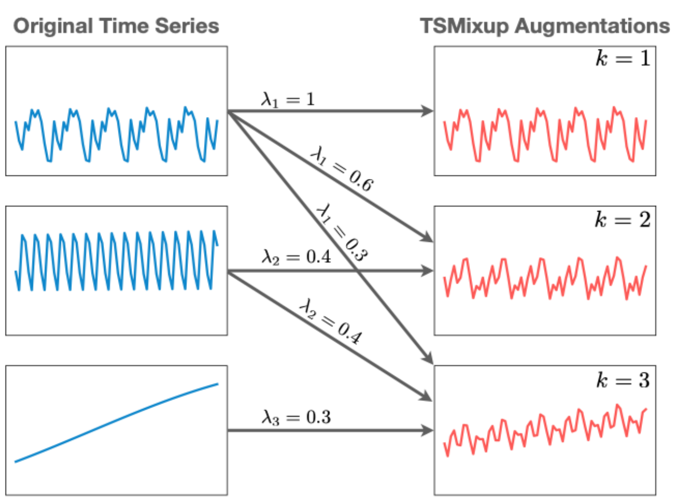
[BEGIN DATA]
***
[Domain]: We usually observe that electricity consumption peaks at noon, with a significant increase in transformer load
***
[Instruction]: Predict the next <*H*> steps given the previous <*T*> steps information attached
***
[Statistics]: The input has a minimum of <min_val>, a maximum of <max_val>, and a median of <median_val>. The overall trend is <upward or downward>. The top five lags are <lag_val>.
[END DATA]



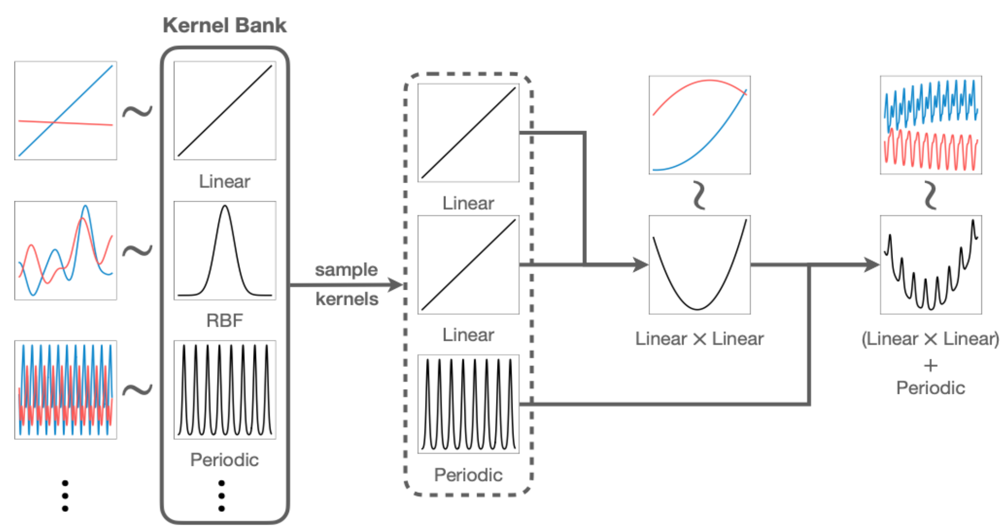
- **Chronos:** Learning the Language of Time Series [AWS., 2024]
  - Pretraining an Time Series Language Model, for Zero-shot forecasting
    - Train a T5 model from scratch on time-series data
  - Tokenization: Scaling & Quantization into Discrete tokens
  - Use Public dataset & **Synthetic dataset**



- **Chronos:** Learning the Language of Time Series [AWS., 2024]
  - Quality and quantity of public time-series data pales compared to language
  - Data Augmentation: **TSMixup**
    - Idea of Mixup [Zhang et al., 2017] appliedat time-series for more than two datapoints
  - Synthetic data: **KernelSynth**
    - Gaussian Process based time series generation; construct a kernel bank of patterns
    - Sampled kernels randomly combined with binary operator (x or +)



TSMixup

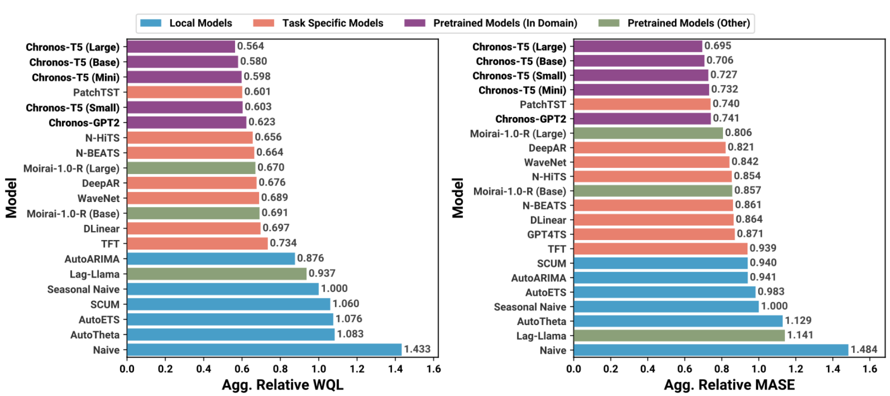


KernelSynth

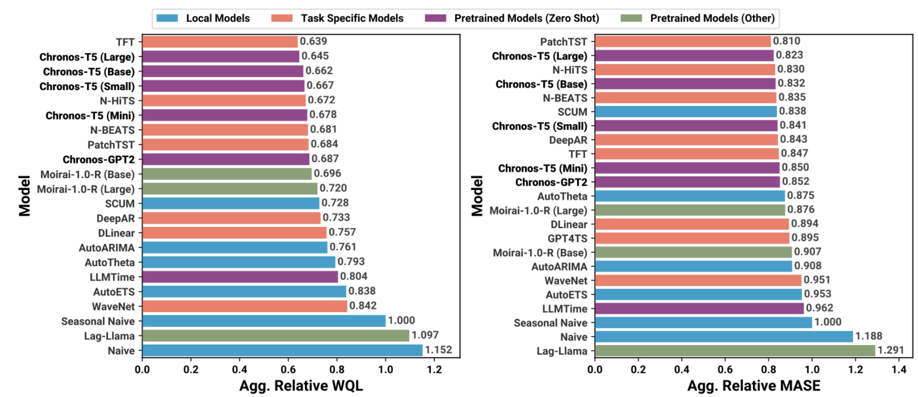
- **Chronos:** Learning the Language of Time Series [AWS., 2024]
  - Experiments: In-domain (left) & Zero-shot (right)
  - Pretrained Chronos shows better performance (Purple, lower the better)
    - Local statistical models (Blue, fitting parameters for each time series)
    - Task-specific models (Orange, training a separate model for each task)

Data Subset	# Datasets	# Series	Usage	Baselines
Pretraining-only	13	795,936	pretraining	–
Benchmark I	15	97,272	pretraining and in-domain evaluation	Naive, SeasonalNaive, AutoETS, Auto-Theta, SCUM, AutoARIMA, DeepAR, TFT, PatchTST, DLinear, WaveNet, N-BEATS, N-HiTS, GPT4TS, Lag-Llama, Moirai-1.0-R
Benchmark II	27	190,674	zero-shot evaluation	All the above, LLMTime and ForecastPFN

Dataset & Baselines

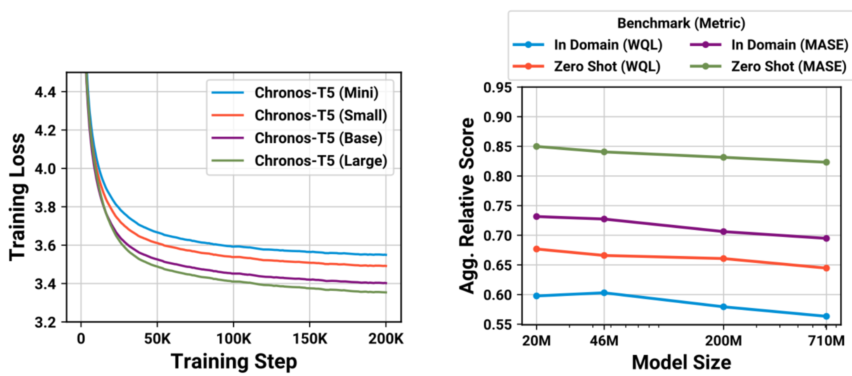


In-domain Results



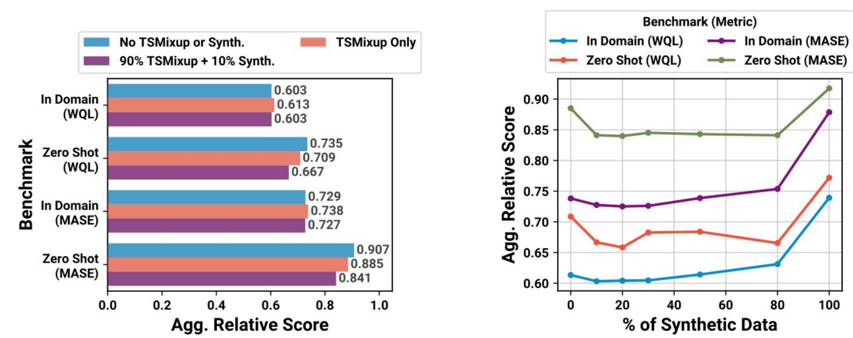
Zero-shot Results

- **Chronos:** Learning the Language of Time Series [AWS., 2024]
- Conclusion:
  - Existing language model architecture and training procedures are adaptable to training and performing time-series forecasting
  - Data & scaling works in the time-series domain, building a generalist model
  - Developing methods for generating **synthetic time series data** is a promising direction



(a) (b)

Model size ablations



(a) (b)

Ablations of data augmentation and Synthetic data proportion (lower the better)

Table of Contents

Table of Contents

Possibilities of LLM as an agent

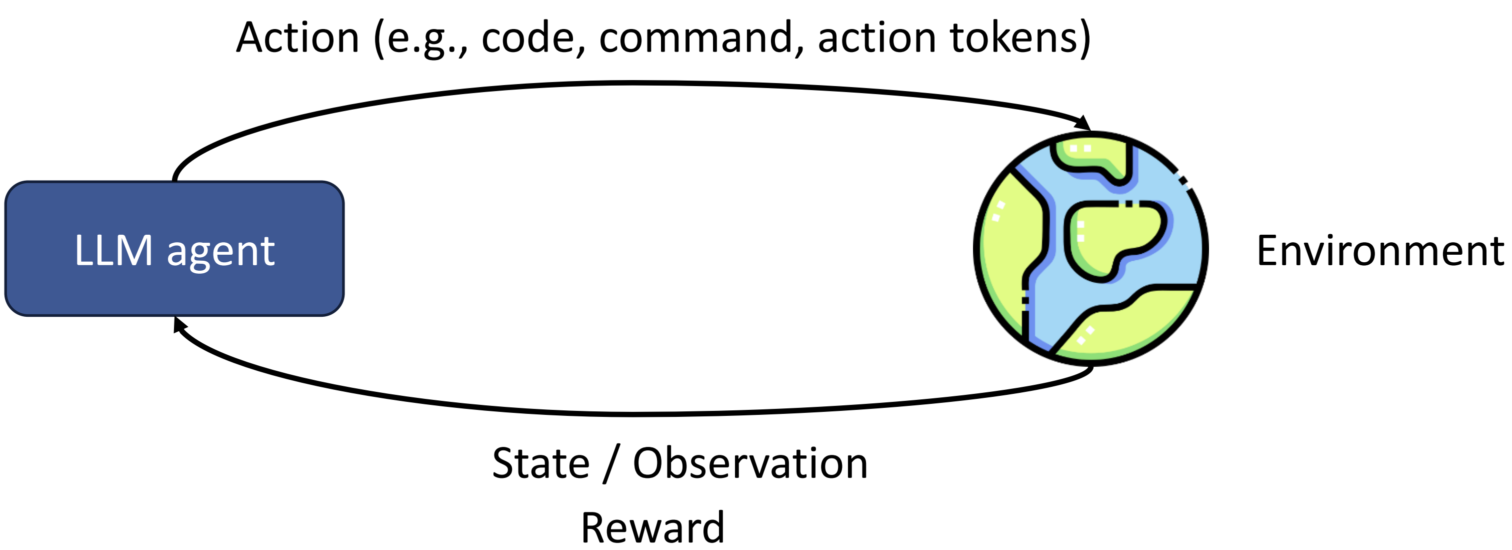
- LLMs show promising results in real-world **sequential decision-making** tasks based on:
  - Vast amount of world knowledge (e.g., "Milk might be placed in the refrigerator")
  - Reasoning and planning capabilities.

Examples of agentic tasks

- <u>Web browsing</u>: given arbitrary goal, agent navigate over web pages by clicking the UI element, in order to fulfill the goal.
- <u>Software engineering</u>: given arbitrary goal, agent implement repository by creating / opening files, implementing code, and execute the code if necessary.

Overall pipeline

- LLM / MLLM understands natural language instruction (goal) and visual/textual state.
- Based on the goal and current state, LLM generates code or command to execute the action.
- Depending on the environment, reward is given at training phase.



Use cases

- Web browsing
  - State/Observation: HTML, pixel (screenshot)
  - Action: code/command for UI interaction (e.g., click(id), type(value, id))

- Software engineering
  - State/Observation: Repo-tree / contents of currently opened file
  - Action: agent-computer interface (e.g., open(file_name), scroll_down(), ..)

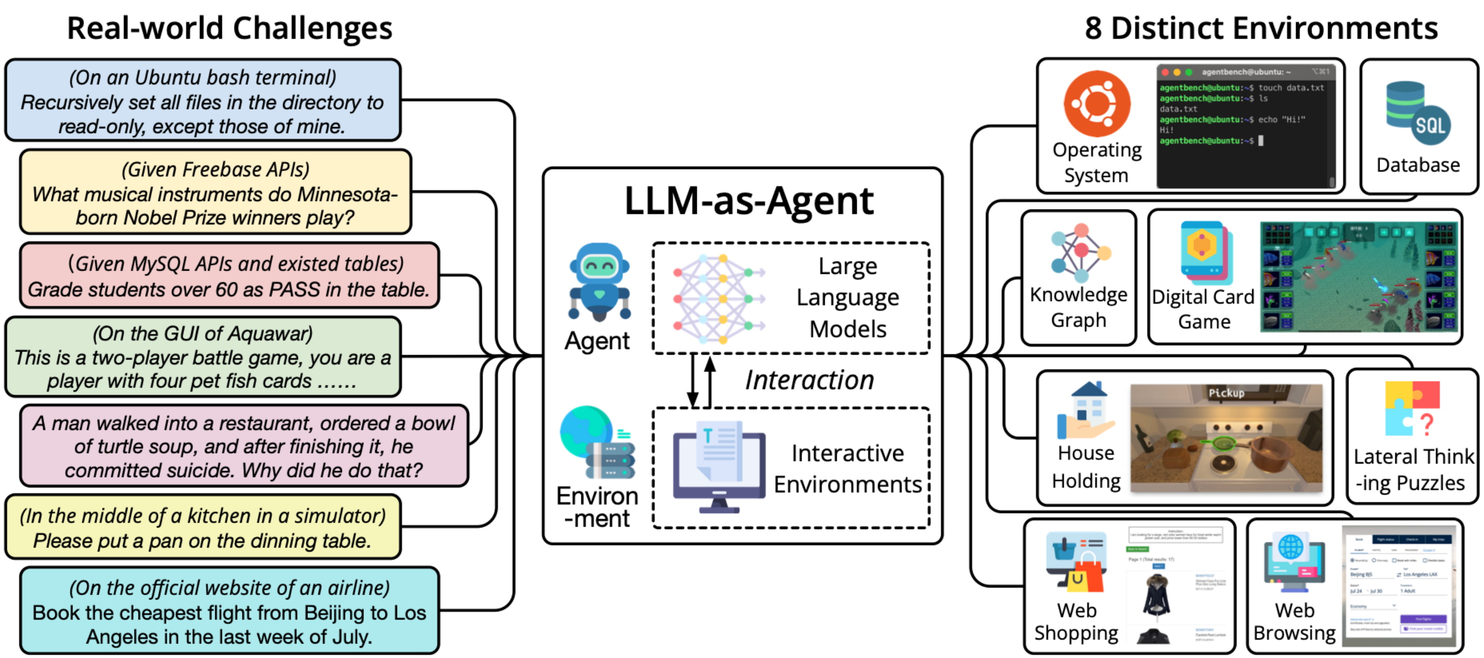
- Robotic tasks
  - State/Observation: Robot state, pixel (camera observation)
  - Action: action token

Challenges

- Learning long-term reward maximizing behavior (rather than become myopic).
  - Advanced Reasoning & Planning capability can be a key.
  - RL with task reward can also be a path to such behavior.

AgentBench: Evaluating LLMs as Agents [Liu et al., 2023]

- Unified benchmark for evaluating LLM agents in text-based decision-making tasks.
- Including various agentic tasks: agent for database, OS, web browsing, web shopping, and text-based card games.



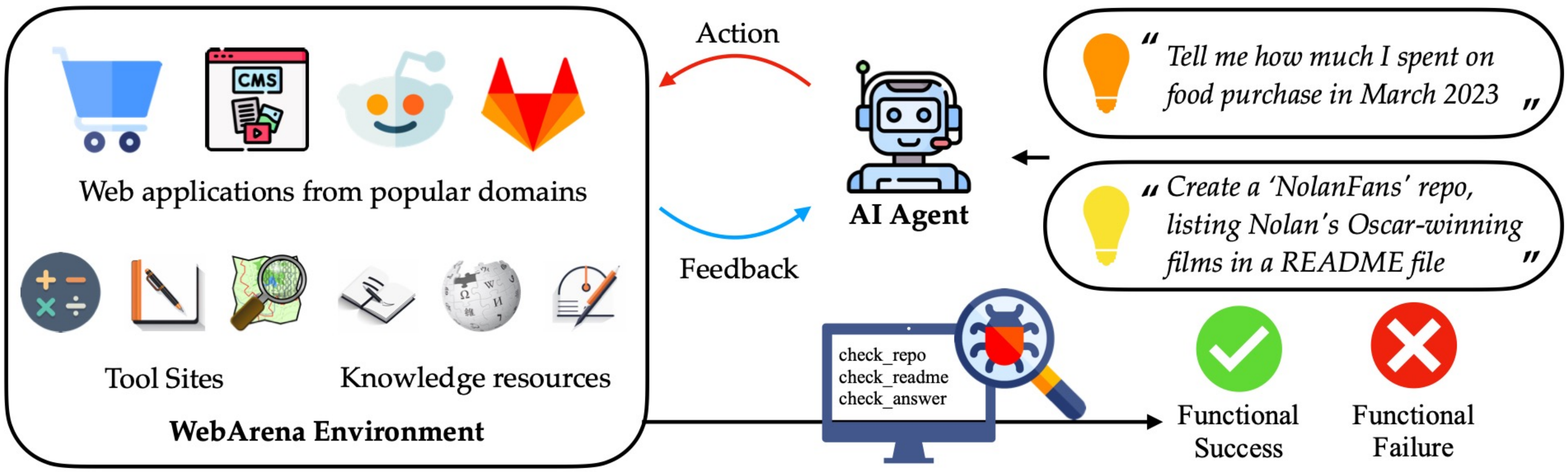
AgentBench: Evaluating LLMs as Agents [Liu et al., 2023]

- Even proprietary LLMs (e.g., GPT-4, Claude) struggle to solve various decision-making tasks.
- Long-term reasoning/planning capabilities are required for better LLM agents.

LLM Type	Models	VER	OA	Code-grounded			Game-grounded			Web-grounded	
				OS	DB	KG	DCG	LTP	HH	WS	WB
API	gpt-4	0613	**4.01**	**42.4**	32.0	**58.8**	**74.5**	**16.6**	**78.0**	61.1	**29.0**
	claude-2	-	2.49	18.1	27.3	41.3	55.5	8.4	54.0	61.4	0.0
	claude	v1.3	2.44	9.7	22.0	38.9	40.9	8.2	58.0	55.7	25.0
	gpt-3.5-turbo	0613	2.32	32.6	**36.7**	25.9	33.7	10.5	16.0	**64.1**	20.0
	text-davinci-003	-	1.71	20.1	16.3	34.9	3.0	7.1	20.0	61.7	26.0
	claude-instant	v1.1	1.60	16.7	18.0	20.8	5.9	12.6	30.0	49.7	4.0
	chat-bison-001	-	1.39	9.7	19.7	23.0	16.6	4.4	18.0	60.5	12.0
	text-davinci-002	-	1.25	8.3	16.7	41.5	11.8	0.5	16.0	56.3	9.0

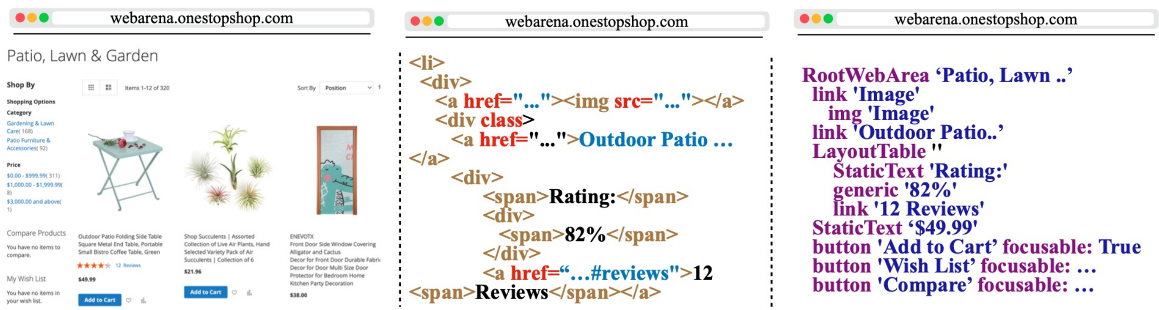
WEBARENA: A Realistic Web Environment for Building Autonomous Agents [Zhou et al., 2023]

- Benchmarks for web browsing tasks are based on a simulated environment rather than real-world websites.
- This benchmark proposes benchmark spanning over 812 tasks across 6 websites (e.g., Map, Gitlab, online shopping, Reddit).
- Evaluates functional correctness (i.e., success rate) over all tasks.



WEBARENA: A Realistic Web Environment for Building Autonomous Agents [Zhou et al., 2023]

- 3 Types of observations are supported (Screenshot, HTML, accessibility tree)



- Commands for diverse UI actions are supported.

Action Type	Description
`noop`	Do nothing
`click(elem)`	Click at an element
`hover(elem)`	Hover on an element
`type(elem, text)`	Type to an element
`press(key_comb)`	Press a key comb
`scroll(dir)`	Scroll up and down
`tab_focus(index)`	focus on i-th tab
`new_tab`	Open a new tab
`tab_close`	Close current tab
`go_back`	Visit the last URL
`go_forward`	Undo `go_back`
`goto(URL)`	Go to URL

Figure 4: Action Space of WebArena

WEBARENA: A Realistic Web Environment for Building Autonomous Agents [Zhou et al., 2023]

- Even GPT-4 struggles to solve most of the tasks (with 14% of success rate).
- Significant gap between human-level performance (77.78%)

CoT	UA Hint	Model	SR	SR_{AC}	SR_{UA}
✓	✓	TEXT-BISON-001	5.05	4.00	27.78
✗	✓	GPT-3.5	6.41	4.90	38.89
✓	✓	GPT-3.5	8.75	6.44	58.33
✓	✓	GPT-4	11.70	8.63	**77.78**
✗	✗	GPT-3.5	5.10	4.90	8.33
✓	✗	GPT-3.5	6.16	6.06	8.33
✓	✗	GPT-4	**14.41**	**13.02**	44.44
-	✓	Human	78.24	77.30	100.00

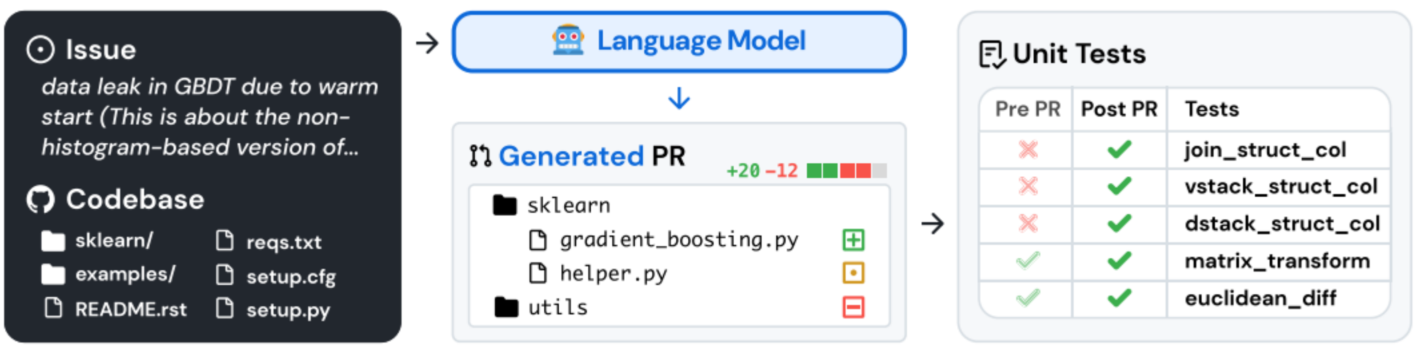
WEBARENA: A Realistic Web Environment for Building Autonomous Agents [Zhou et al., 2023]

- Recent works focused on computer-using agent improved the performance by large margin.

	A	B	C	D	E	F
1	**Release Date**	**Open?**	**Model Size (billion)**	**Model**	**Success Rate (%)**	**Result Source**
2	02/2025	✗	-	IBM CUGA	61.7	IBM CUGA
3	01/2025	✗	-	OpenAI Operator	58.1	OpenAI CUA
4	08/2024	✗	-	Jace.AI	57.1	Reported by zetalabs.ai
5	12/2024	✗	-	ScribeAgent + GPT-4o	53	ScribeAgent
6	01/2025	✔	-	AgentSymbiotic	52.1	AgentSymbiotic
7	01/2025	✔	-	Learn-by-Interact	48	Learn-by-interact
8	10/2024	✔	-	AgentOccam-Judge	45.7	AgentOccam-Judge
9	08/2024	✗	-	WebPilot	37.2	WebPilot
10	10/2024	✔	-	GUI-API Hybrid Agent	35.8	Beyond Browsing
11	09/2024	✔	-	Agent Workflow Memory	35.5	AWM
12	04/2024	✔	-	SteP	33.5	SteP
13	04/2024	✔	-	BrowserGym + GPT-4	23.5	WorkArena
14	01/2025	✔	32	AgentTrek-1.0-32B	22.4	AgentTrek
15	04/2024	✔	-	GPT-4 + Auto Eval	20.2	Auto Eval & Refine
16	06/2024	✔	-	GPT-4o + Tree Search	19.2	Tree Search for LM Agents
17	04/2024	✔	7	AutoWebGLM	18.2	AutoWebGLM

SWE-bench: Can Language Models Resolve Real-World GitHub Issues? **[Jimenez et al., 2023]**

- Task of resolving the Github issue given issue description and codebase.
- Agent needs to modify specific part of the codebase so that the issue is resolved.
- Once patch file is generated, the patch is applied, and then evaluated by pre-defined unit tests.



SWE-bench: Can Language Models Resolve Real-World GitHub Issues? **[Jimenez et al., 2023]**

- Tasks are based on 12 well-maintained opensource Github repositories.
- Codebase corresponding to each tasks incorporates lengthy lines of code and files (far exceeds context length of frontier LLMs, e.g., 200K tokens).

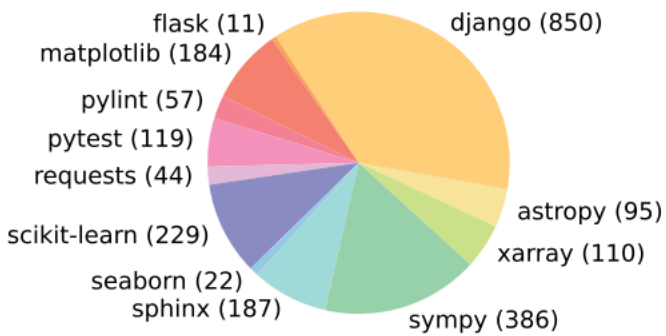


Figure 3: Distribution of SWE-bench tasks (in parenthesis) across 12 open source GitHub repositories that each contains the source code for a popular, widely downloaded PyPI package.

Table 1: Average and maximum numbers characterizing different attributes of a SWE-bench task instance. Statistics are micro-averages calculated without grouping by repository.

		Mean	Max
Issue Text	Length (Words)	195.1	4477
Codebase	# Files (non-test)	3,010	5,890
	# Lines (non-test)	438K	886K
Gold Patch	# Lines edited	32.8	5888
	# Files edited	1.7	31
	# Func. edited	3	36
Tests	# Fail to Pass	9.1	1633
	# Total	120.8	9459

SWE-bench: Can Language Models Resolve Real-World GitHub Issues? **[Jimenez et al., 2023]**

- Baseline: Retrieve relevant code file from entire repository using RAG (i.e., using issue description as a query) → modify the retrieved code file.
- SWE-Llama is trained to generate corrected code, given retrieved code containing faults.

	SWE-bench		SWE-bench Lite	
Model	% Resolved	% Apply	% Resolved	% Apply
Claude 3 Opus	**3.79**	46.56	**4.33**	**51.67**
Claude 2	1.97	43.07	3.00	33.00
ChatGPT-3.5	0.17	26.33	0.33	10.00
GPT-4-turbo	1.31	26.90	2.67	29.67
SWE-Llama 7b	0.70	51.74	1.33	38.00
SWE-Llama 13b	0.70	**53.62**	1.00	38.00

SWE-bench: Can Language Models Resolve Real-World GitHub Issues? **[Jimenez et al., 2023]**

- Recent works further improved performance in SWE-Bench.

Leaderboard

Lite | Verified | Full | Multimodal

Model	% Resolved	Org
🥇 W&B Programmer O1 crosscheck5	64.60	
🆕 🥈 AgentScope	63.40	-
🥉 Blackbox AI Agent	62.80	-
🤠 CodeStory Midwit Agent + swe-search	62.20	-
🆕 🤠 OpenHands + 4x Scaled (2024-02-03)	60.80	All Hands
Learn-by-interact	60.20	
devlo	58.20	
Emergent E1 (v2024-12-23)	57.20	
Gru(2024-12-08)	57.00	
EPAM AI/Run Developer Agent v20241212 + Anthopic Claude 3.5 Sonnet	55.40	<epam>
Amazon Q Developer Agent (v20241202-dev)	55.00	aws
devlo	54.20	
Bracket.sh	53.20	
🤠 ✅ OpenHands + CodeAct v2.1 (claude-3-5-sonnet-20241022)	53.00	All Hands
Google Jules + Gemini 2.0 Flash (v20241212-experimental)	52.20	Google
Engine Labs (2024-11-25)	51.80	
AutoCodeRover-v2.1 (Claude-3.5-Sonnet-20241022)	51.60	

Table of Contents

ReAct: Synergizing Reasoning and Acting in Language Models [Yao et al., 2023]

- Prompting technique to improve LLMs' decision-making capability.
- Applying Chain-of-Thought prompting to decision making tasks.
- Enforces LLM agents to think before act via prompting.

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
Your task is to: Put some pepper shaker on a drawer.

(2a) Act-Only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dishsponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dishsponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens.

(2b) ReAct (Reason + Act)

Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, ... a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 …

Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1.

ReAct: Synergizing Reasoning and Acting in Language Models [Yao et al., 2023]

- Evaluation in AlfWorld and WebShop, a representative text-based decision making task.
- ReAct prompting outperforms Act-only prompting with significant margin.
- Recently, **think followed by action** became default choice for LLM agents.

Method	Pick	Clean	Heat	Cool	Look	Pick 2	All
Act (best of 6)	88	42	74	67	72	**41**	45
ReAct (avg)	65	39	83	76	55	24	57
ReAct (best of 6)	**92**	58	**96**	86	**78**	**41**	**71**
ReAct-IM (avg)	55	59	60	55	23	24	48
ReAct-IM (best of 6)	62	**68**	87	57	39	33	53
BUTLER_g (best of 8)	33	26	70	76	17	12	22
BUTLER (best of 8)	46	39	74	**100**	22	24	37

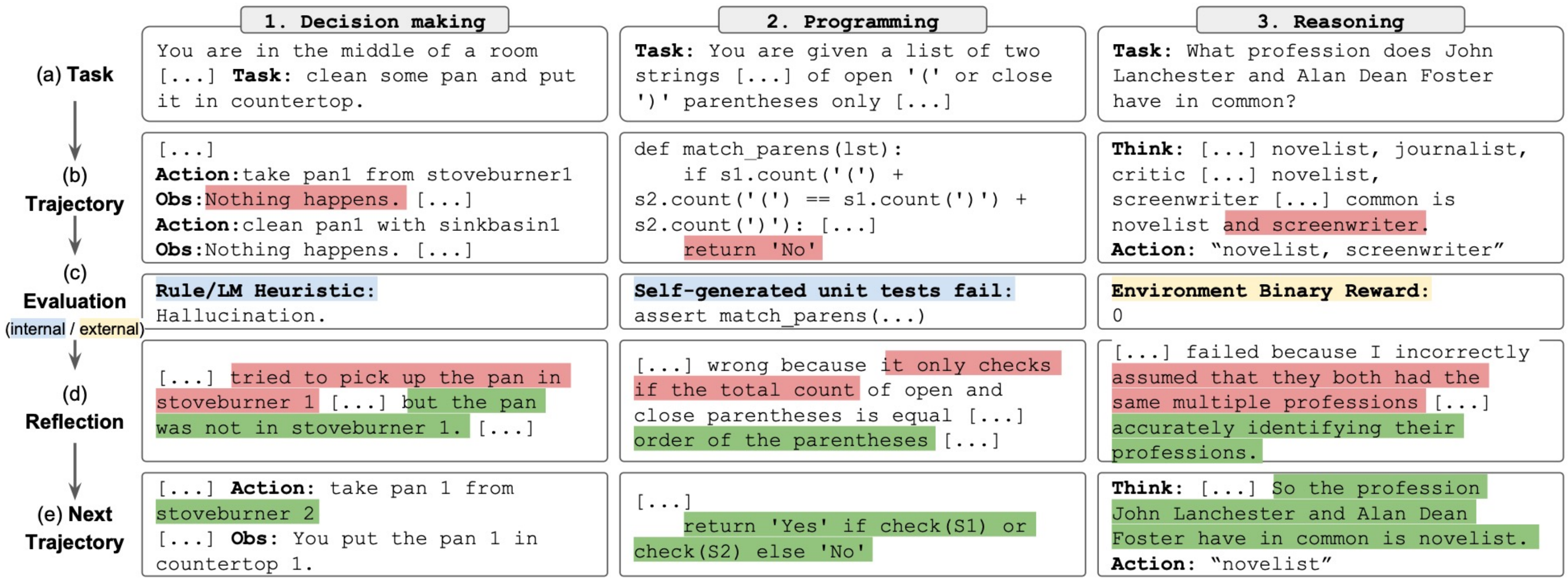
Table 3: AlfWorld task-specific success rates (%). BUTLER and BUTLER_g results are from Table 4 of Shridhar et al. (2020b). All methods use greedy decoding, except that BUTLER uses beam search.

Method	Score	SR
Act	62.3	30.1
ReAct	**66.6**	**40.0**
IL	59.9	29.1
IL+RL	62.4	28.7
Human Expert	82.1	59.6

Table 4: Score and success rate (SR) on Webshop. IL/IL+RL taken from Yao et al. (2022).

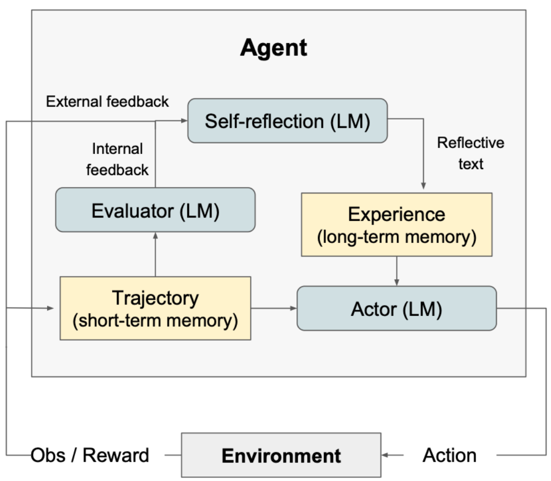
Reflexion: Language Agents with Verbal Reinforcement Learning [Shinn et al., 2023]

- LLM agent refining its decision making based on **verbal feedback**.
- New paradigm of **verbal reinforcement learning**



Reflexion: Language Agents with Verbal Reinforcement Learning [Shinn et al., 2023]

- LLM agent generates trajectory by decision making.
- LLM agent receives verbal external feedback or internal feedback (i.e., self-evaluation).
- Based on the feedback, LLM agent generates reflection, and adds it to long-term memory.
- Regenerate trajectory by referring to the reflection.



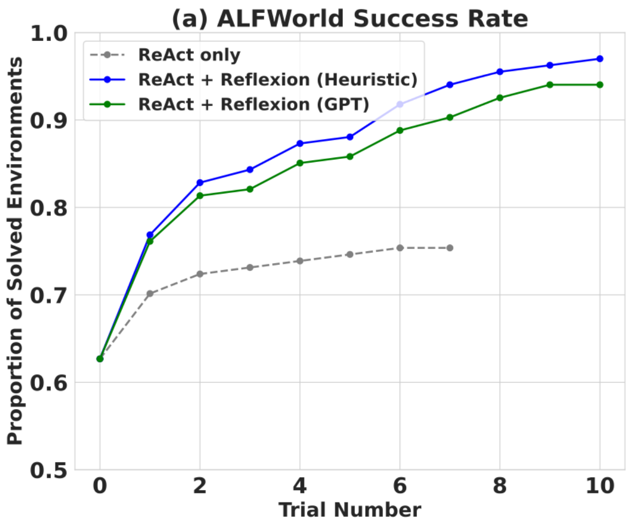
Algorithm 1 Reinforcement via self-reflection

Initialize Actor, Evaluator, Self-Reflection: M_a, M_e, M_{sr}
Initialize policy $\pi_\theta(a_i|s_i)$, $\theta = \{M_a, mem\}$
Generate initial trajectory using π_θ
Evaluate τ_0 using M_e
Generate initial self-reflection sr_0 using M_{sr}
Set $mem \leftarrow [sr_0]$
Set $t = 0$
while M_e not pass or $t <$ max trials **do**
    Generate $\tau_t = [a_0, o_0, \dots a_i, o_i]$ using π_θ
    Evaluate τ_t using M_e
    Generate self-reflection sr_t using M_{sr}
    Append sr_t to mem
    Increment t
end while
return

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

Reflexion: Language Agents with Verbal Reinforcement Learning [Shinn et al., 2023]

- Language agent improves its decision making within a few iterations of Reflexion in sequential decision-making task (ALFWorld)
- Not only confined to decision making tasks, Reflexion can be also applied to programming tasks (e.g., MBPP, HumanEval)
  - Shows better than previous state-of-art methods.



Benchmark + Language	Base	Reflexion	TP	FN	FP	TN
HumanEval (PY)	0.80	**0.91**	0.99	0.40	0.01	0.60
MBPP (PY)	**0.80**	0.77	0.84	0.59	0.16	0.41
HumanEval (RS)	0.60	**0.68**	0.87	0.37	0.13	0.63
MBPP (RS)	0.71	**0.75**	0.84	0.51	0.16	0.49

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering [Yang et al., 2024]

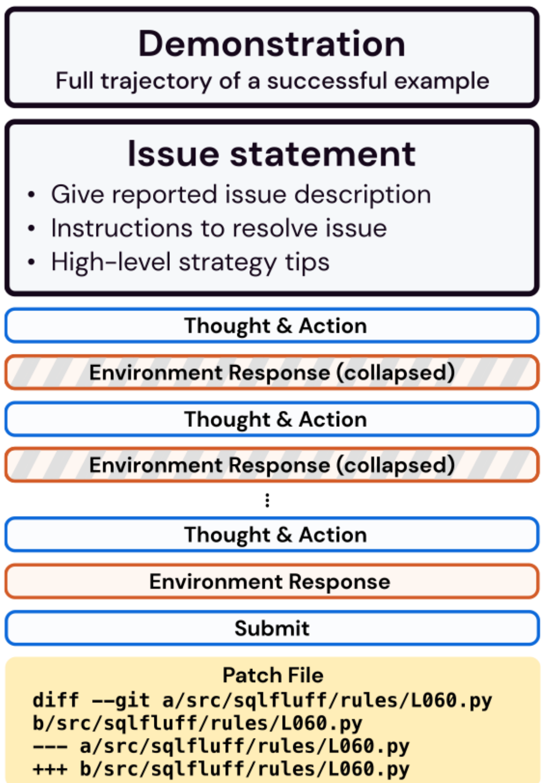
- Proposed agent-computer interface enabling LLMs to solve software engineering tasks as human developer do.

Table 4: In additional to the standard Linux Bash commands, we provide SWE-agent with specialized tools, including an interactive file viewer, search functionalities, and edit tools for the open file. Required arguments are enclosed in <> and optional arguments are in []. The last column shows the documentation presented to the LM.

Category	Command	Documentation
File viewer	**open** <path> [<line_number>]	Opens the file at the given path in the editor. If line_number is provided, the window will move to include that line.
	goto <line_number>	Moves the window to show line_number.
	scroll_down	Moves the window up 100 lines.
	scroll_up	Moves the window down 100 lines.
Search tools	**search_file** <search_term> [<file>]	Searches for search_term in file. If file is not provided, searches in the current open file.
	search_dir <search_term> [<dir>]	Searches for search_term in all files in dir. If dir is not provided, searches in the current directory.
	find_file <file_name> [<dir>]	Finds all files with the given name in dir. If dir is not provided, searches in the current directory.
File editing	**edit** <n>:<m> <replacement_text> **end_of_edit**	Replaces lines n through m (inclusive) with the given text in the open file. All of the replacement_text will be entered, so make sure your indentation is formatted properly. Python files will be checked for syntax errors after the edit. If an error is found, the edit will not be executed. Reading the error message and modifying your command is recommended as issuing the same command will return the same error.
	create <filename>	Creates and opens a new file with the given name.
Task	**submit**	Generates and submits the patch from all previous edits and closes the shell.

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering [Yang et al., 2024]

- Based on issue description and 1-shot example (demonstration), LLM sequentially make decision using actions, which is defined based on agent-computer interface.



Think1: First, I need to reproduce the issue…
Action1: create_file('reproduce.py')

Think2: As the issue describes problem regarding …
Action2: edit 1:1
[code to reproduce the error]
end_of_edit

Think3: I need to run the` reproduce.py` to see whether the error is reprocuded..
Action3: python reproduce.py

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering
[Yang et al., 2024]

- Shows promising result in the representative repository-level software engineering tasks: SWE-Bench

Table 1: Main results for SWE-agent performance on the full and Lite splits of the SWE-bench test set. We benchmark models in the SWE-agent, Basic CLI, and Retrieval Augmented Generation (RAG) settings established in SWE-bench [20].

	SWE-bench		SWE-bench Lite	
Model	% Resolved	$ Avg. Cost	% Resolved	$ Avg. Cost
RAG				
w/ GPT-4 Turbo	1.31	0.13	2.67	0.13
w/ Claude 3 Opus	3.79	0.25	4.33	0.25
Shell-only agent				
w/ GPT-4 Turbo	-	-	11.00	1.46
w/o Demonstration	-	-	7.33	0.79
SWE-agent				
w/ GPT-4 Turbo	**12.47**	1.59	**18.00**	1.67
w/ Claude 3 Opus	10.46	2.59	13.00	2.18

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering
[Yang et al., 2024]

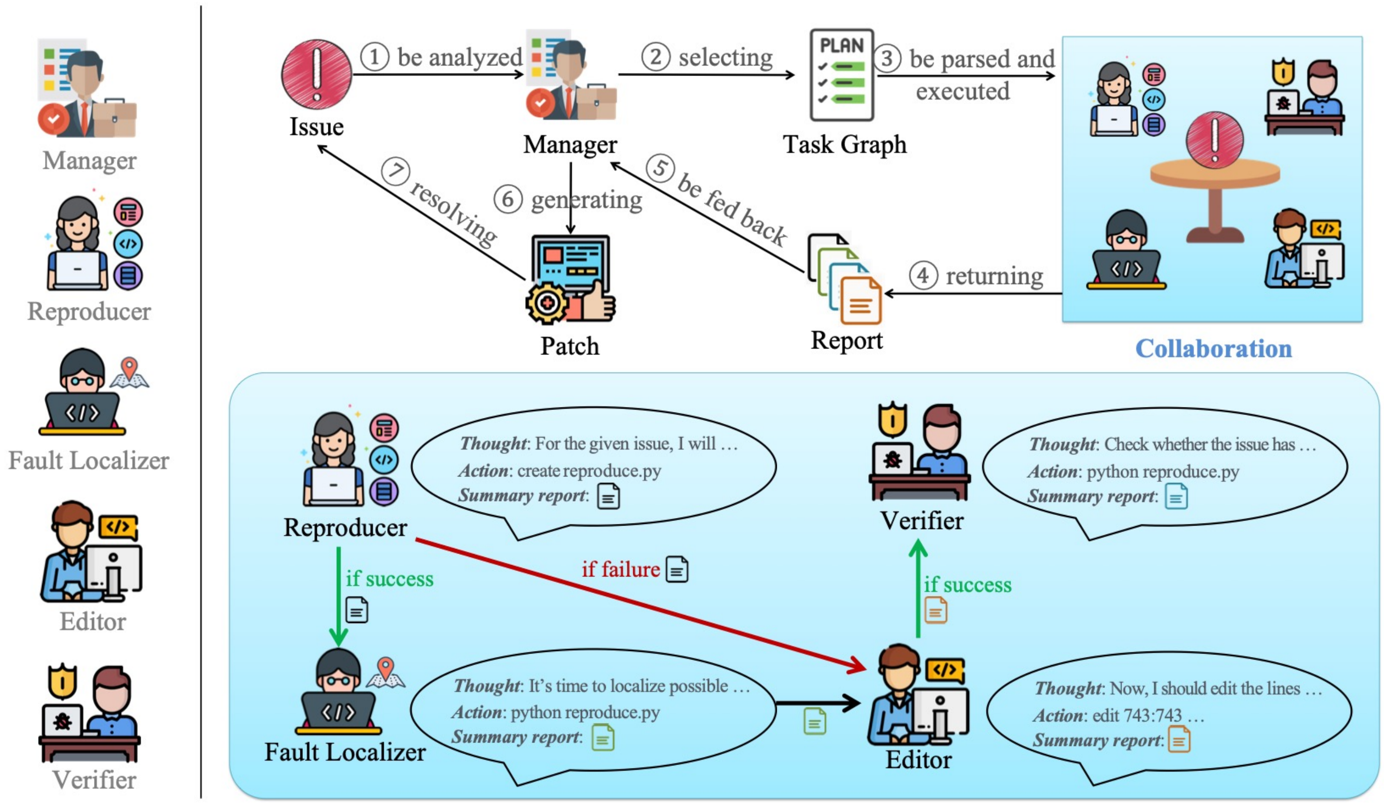
- Although method is simple (i.e., letting LLMs to use tools & actions specialized for software engineering tasks), It showed the possibility of LLMs to solve repo-level software engineering tasks as human developers do.

- After this work, many works proposed better agent-computer interfaces to enhance the performance.

- Remaining problems:
  - LLMs makes trivial mistakes while editing the code (e.g., indentation error), and some errors are not detected by linting library, which results in task failure.
  - Edit & Execution loop: once the execution of LLM-edited code returns error, LLMs repeat editing the code and executing the wrongly edited code.

Code-R: Issue Resolving with Multi-Agent and Task Graphs [Chen et al., 2024]

- Proposed hierarchical multi-agent framework for software engineering task.
- Pre-defines role of each agent (e.g., Supervisor, Fault Localizer, Fault Reproducer) , and available actions are different across roles.



Code-R: Issue Resolving with Multi-Agent and Task Graphs [Chen et al., 2024]

- Given issue description, Manager agent generates task graph, which defines workflow and coordination between low-level agents.
- Following the task graph, low-level agents follow the workflow to solve the task.

```
{
  "Plan A": {
    "entry": "Reproducer",
    "roles": [{
      "name": "Reproducer",
      "attributes": {
        "task": "If possible, try to extract test scripts from the issue description. Otherwise, generate test scripts based on the issue description yourself. Paste it into `path/to/reproduce.py`. Run `path/to/reproduce.py` and compare error messages with those in the description. If successfully reproduced, forward the error message to the Fault Localizer. If not, forward \"cannot reproduce\" to the Editor.",
        "downstream": {"succeed": {"to": "Fault Localizer "}, "fail": {"to": "Editor"}}}},
      {"name" : "Fault Localizer", ...}, {"name": "Editor", ...}, {"name": " Verifier", ...}]},
  "Plan B": {...}, "Plan C": {...}, "Plan D": {...}, ...
}
```

Code-R: Issue Resolving with Multi-Agent and Task Graphs [Chen et al., 2024]

- Each agent are assigned with different agent-computer interfaces (i.e., action space)
- For example, Reproducer and Editor can edit the code, while remaining agents can not directly edit the code.

Actions	Agent Roles				
	Manager	Reproducer	Fault Localizer	Editor	Verifier
0 plan	√				
1 open		√		√	
2 goto		√		√	
3 scroll down		√		√	
4 scroll up		√		√	
5 create		√		√	
6 edit		√	√	√	√
7 submit	√				
8 search dir	√	√		√	
9 search file	√	√		√	
10 find file	√	√		√	
11 rover search file*	√	√		√	
12 rover search class*	√	√		√	
13 rover search class in file*	√	√		√	
14 rover search method*	√	√		√	
15 rover search method in file*	√	√		√	
16 rover search code*	√	√		√	
17 rover search code in file*	√	√		√	
18 related issue retrieval			√	√	
19 fault localization			√		
20 test					√
21 report		√	√	√	√
22 basic shell command	√	√	√	√	√

CodeR: Issue Resolving with Multi-Agent and Task Graphs [Chen et al., 2024]

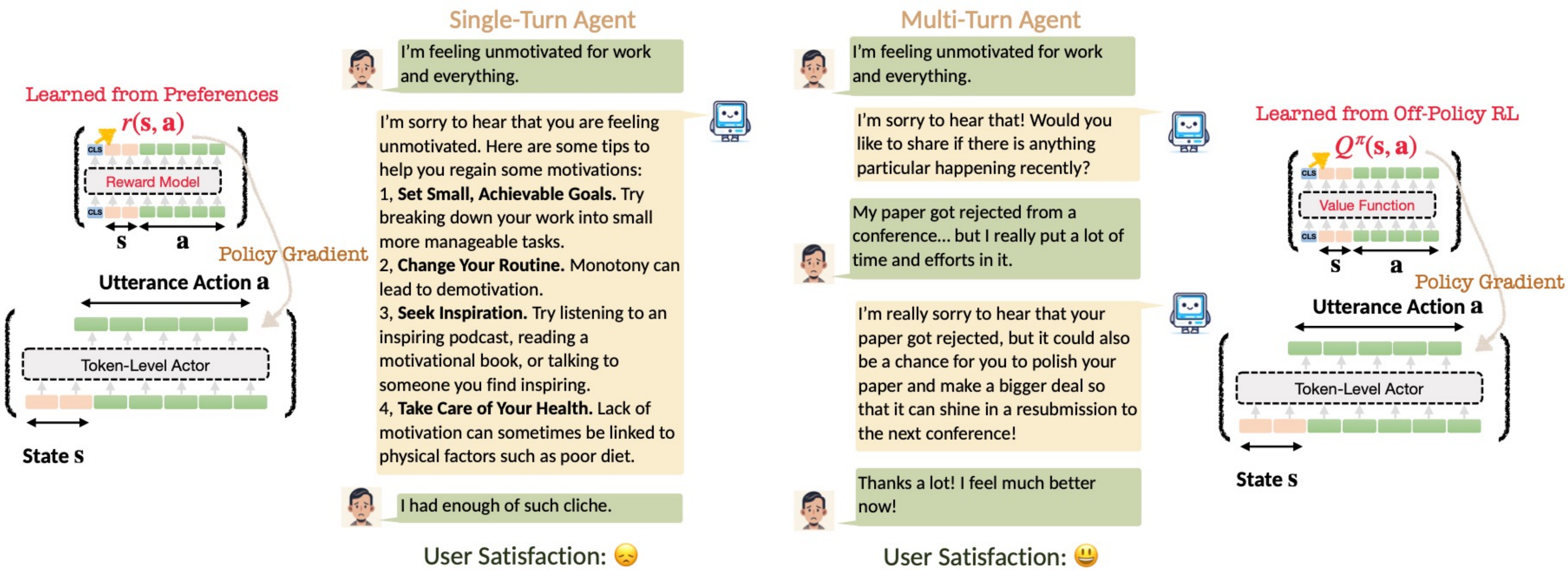
- Multi-agent system results in better performance in SWE-Bench, compared to single agent baseline (SWE-agent), as well as commercial products (e.g., Amazon Q Developer agent, Devin).

Methods	Resolved (%)	Avg. Req.	Avg. Tokens/Cost
Commercial Products			
Devin (random 25% subset of SWE-bench)	13.86 (-)	-	-
Amazon Q Developer Agent (reported)	20.33 (61)	-	-
Amazon Q Developer Agent (reproduced)	17.00 (54)	-	-
OpenCSG CodeGenAgent (reported)	23.67 (71)	-	-
OpenCSG CodeGenAgent (reproduced)	20.67 (62)	-	-
Bytedance MarsCode Agent	22.00 (66)		
Explicit Patch Generation			
RAG + GPT 3.5	0.33 (1)	-	-
RAG + SWE-Llama 13B	1.00 (3)	-	-
RAG + SWE-Llama 7B	1.33 (4)	-	-
RAG + GPT 4	2.67 (8)	-	-
RAG + Claude 2	3.00 (9)	-	-
RAG + Claude 3 Opus	4.33 (13)	-	-
AutoCodeRover	19.00 (57)	-	112k/$1.30
Implicit Patch Generation			
Aider (reported)	26.33 (79)	-	-
Aider (reproduced)	24.67 (74)	-	-
SWE-agent + Claude 3 Opus (reported)	11.67 (35)	17.10	221K/$3.41
SWE-agent + Claude 3 Opus (reproduced)	9.66 (29)	17.10	221K/$3.41
SWE-agent + GPT 4 (reported)	18.00 (54)	21.55	245K/$2.51
SWE-agent + GPT 4 (reproduced)	16.67 (50)	21.55	245K/$2.51
CodeR (reported)	**28.33 (85)**	30.39	299K/$3.09
CodeR (ours)	27.33 (82)	30.39	299K/$3.09

Table of Contents

ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL [Zhou et al., 2024]

- Training LLMs for multi-turn tasks with RL poses several challenges compared to training LLMs for single-turn tasks with RL.
  - As LLMs have to make decision over an extended period of multi-turn interactions.
- Current RL methods to fine-tune LLMs (e.g., RLHF) focus on single-turn tasks.
- ArCHer proposes novel RL framework for training LLMs for multi-turn tasks.

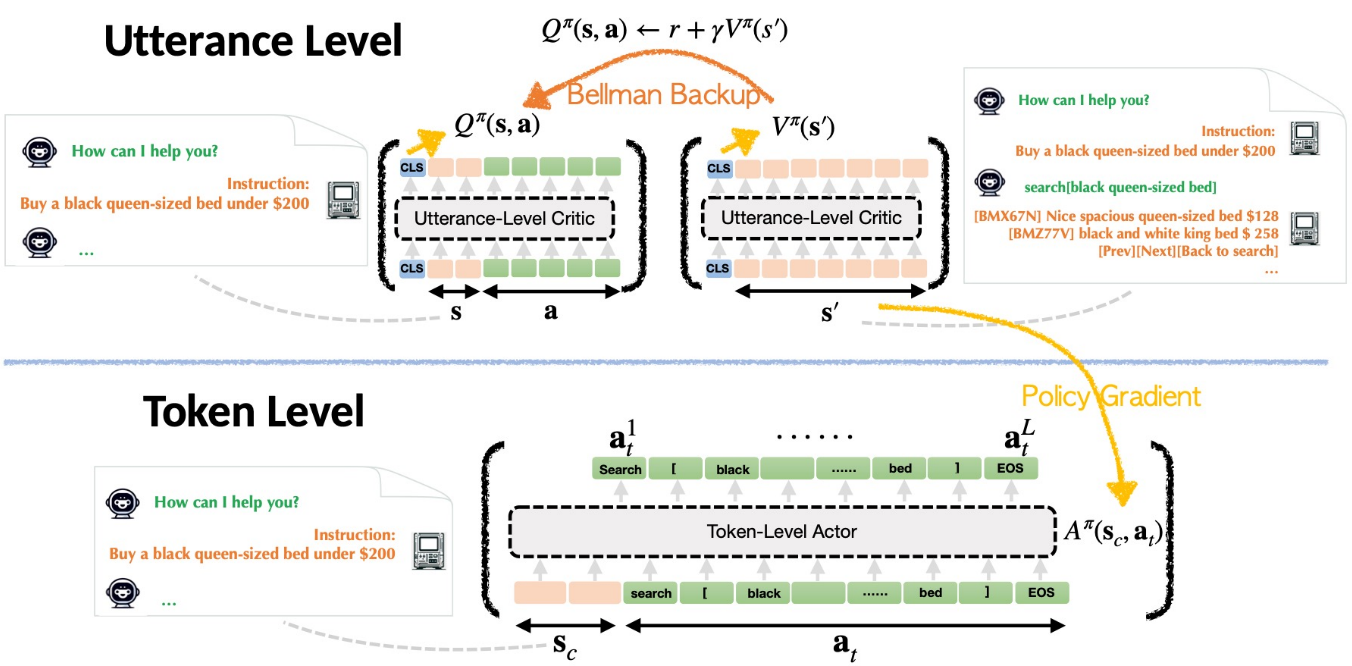


ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL [Zhou et al., 2024]

- In multi-turn tasks (i.e., agent tasks), action space is defined at utterance level (e.g., command, code)
- However, usual RL methods to fine-tune LLMs focus on token-level action space with reward function learned via human preference.

ArChEr: Training Language Model Agents via Hierarchical Multi-Turn RL [Zhou et al., 2024]

- ArCHer proposes hierarchical approach:
  - 1. Train **utterance-level value function** via Off-policy RL
  - 2. **Token-level on-policy RL** (e.g., PPO) with learned **utterance-level value function**.



ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL [Zhou et al., 2024]

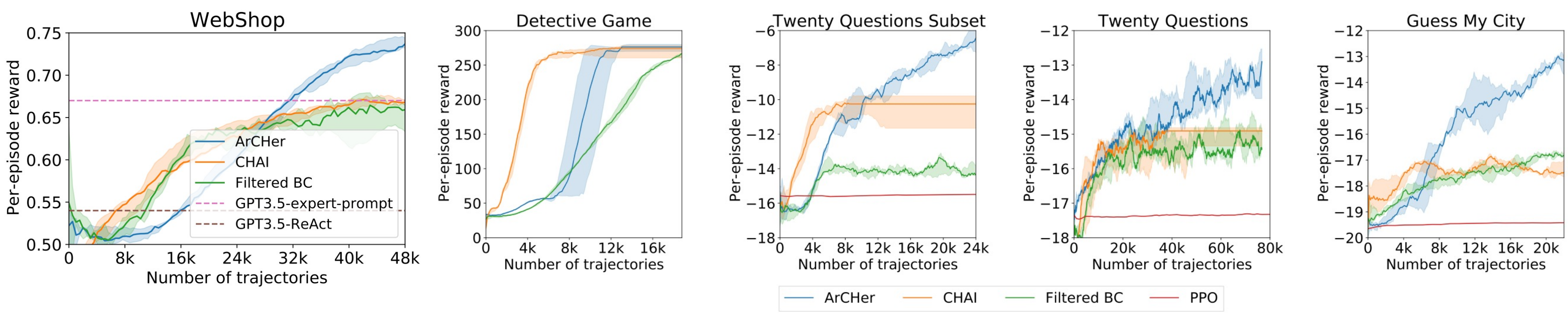
- Overall algorithm

Algorithm 1 ArCHer: Practical Framework

Initialize parameters $\phi, \psi, \theta, \bar{\theta}$, (Optionally) η
Initialize replay buffer $\mathcal{D}$ (optionally from an offline dataset).
for each iteration **do**
  ## Data Collection. ▷ [only online mode]
  for each environment step **do**
    Execute $a_t \sim \pi_\phi(\cdot|s_t)$, obtain the next state s_{t+1}, add to buffer $\mathcal{D}$.
  end for
  for each critic step **do**
    ## Update utterance-level Q and V functions by target function bootstrapping.
    $\theta \leftarrow \theta - \nabla J_\theta(Q)$ ▷ Equation 1
    $\psi \leftarrow \psi - \nabla J_\psi(V)$ ▷ Equation 2 or 6
    ## Update target Q and V functions.
    $\bar{\theta} \leftarrow (1-\tau)\bar{\theta} + \tau\theta$
    $\bar{\psi} \leftarrow (1-\tau)\bar{\psi} + \tau\psi$
  end for
  ## Update token-level baseline by MC regression.
  for each baseline step **do**
    $\eta \leftarrow \eta - \nabla J_\eta(\widetilde{V})$ ▷ (Optionally), Equation 4
  end for
  ## Update token-level actor with utterance-level critic.
  for each actor step **do**
    $\phi \leftarrow \phi - \nabla J_\phi(\pi)$ ▷ Equation 3, 5, or 7
  end for
end for

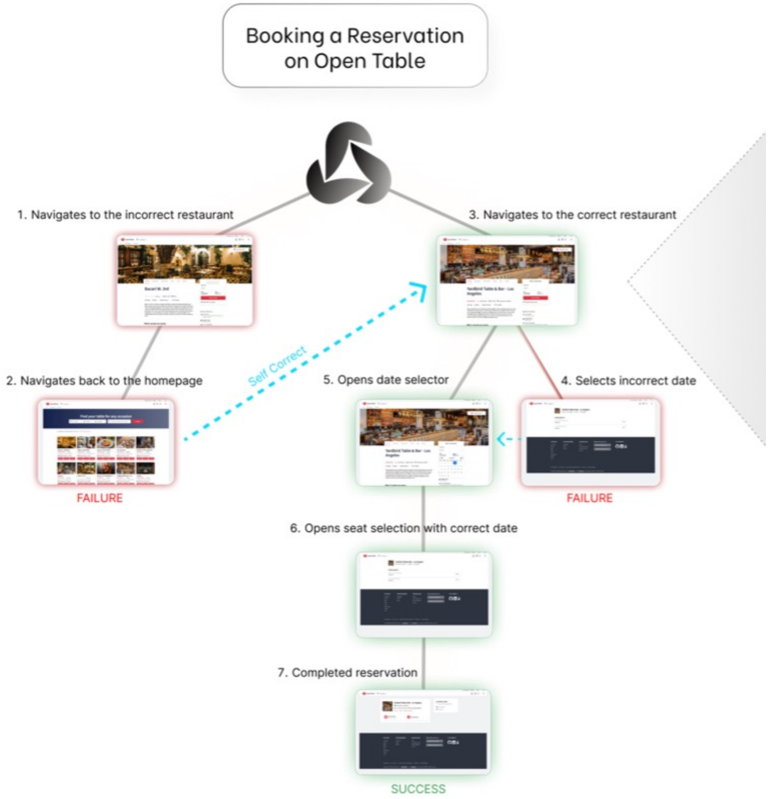
ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL **[Zhou et al., 2024]**

- ArCHer outperforms other training methods.
- Although PPO gradually improves, ArCHer exhibits much sample-efficient learning.
- GPT-2 fine-tuned with ArCHer outperforms GPT-3.5-turbo + ReAct



Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents[Putta et al., 2024]

- Search for optimal decision making via MCTS.
- From the search tree, optimize the LLM agent via Direct preference optimization.



Given a state, LLM agent has multiple choices for actions (i.e., Act 1, Act 2)

Through tree search, we already have information of (value of Act 1 > value of Act 2).

Therefore, we optimize LLM agent with **state, Act 1, and Act 2**, as **prompt, positive completion, and negative completion**, respectively.

Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents [Putta et al., 2024]

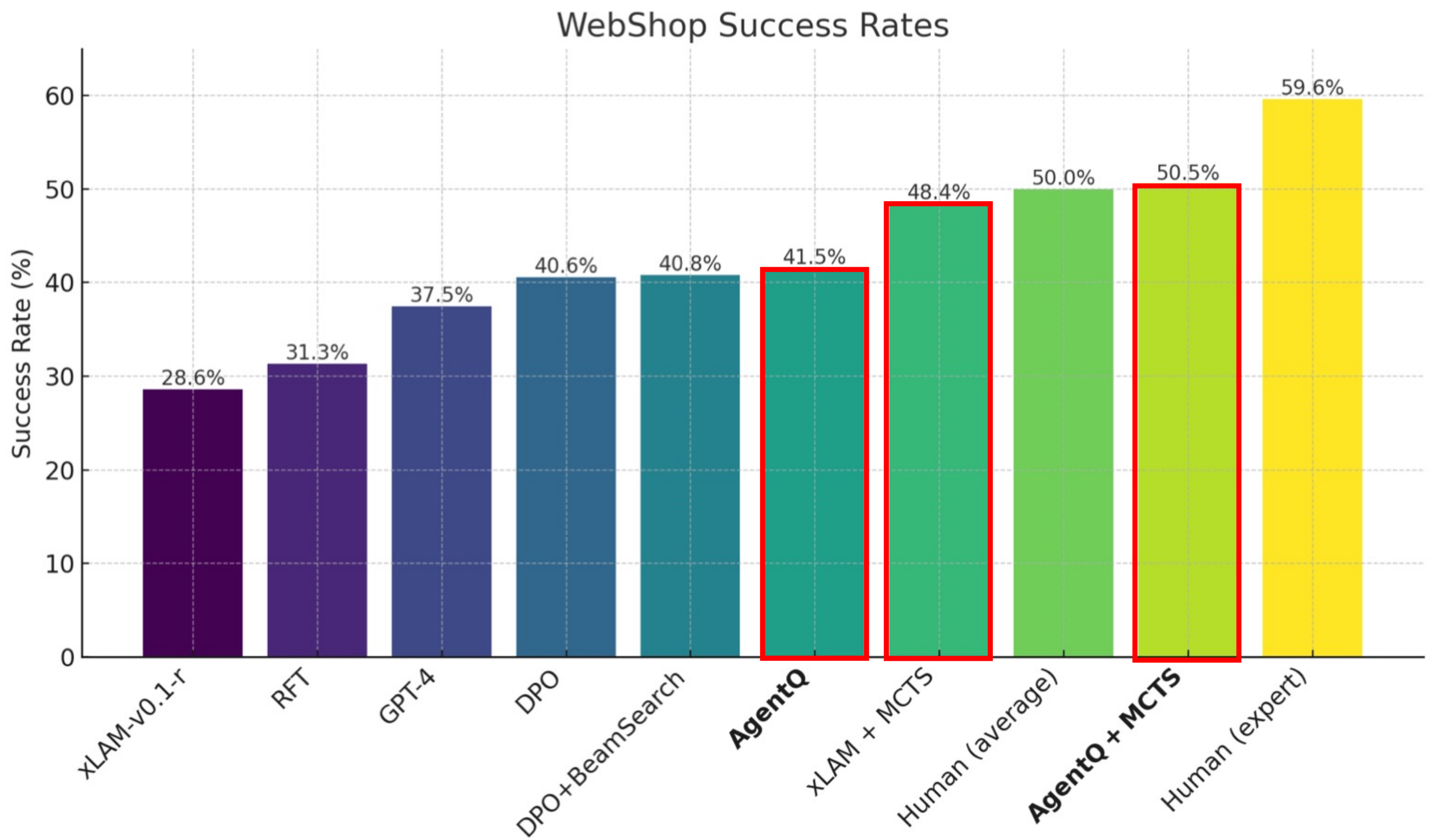
- Overall algorithm

Algorithm 1 MCTS Guided Direct Preference Optimization

Input: π_{θ_0}: initial LLM policy, $\mathcal{D}_T$: dataset of tasks the agent must complete in the environment, N: number of iterations, B: number of samples per iteration, T: MCTS tree depth, $\mathcal{B}$: replay buffer, $\theta_{\text{threshold}}$: value threshold in (10), K: number of actions to sample for MCTS
Output: π_{θ_N}, the trained LLM policy
for $i = 1$ to N **do**
  $\pi_{\text{ref}} \leftarrow \pi_{\theta_i}$, $\pi_{\theta_i} \leftarrow \pi_{\theta_{i-1}}$
  Sample a batch of B tasks from $\mathcal{D}_T$
  for each task in batch **do**
    Initialize the root node $\mathbf{h}_0$
    for $t = 1$ to T **do**
      Selection: Traverse tree from the root node to a leaf node using tree policy (UCB1; 7)
      Trajectory Rollout: From the selected node's trace, roll out the trajectory using
        π_{θ_i} until a terminal state is reached
      Backpropagation: Backpropagate the value estimate bottom-up (8)
    end for
    Collect trajectories from rollouts and store them in replay buffer $\mathcal{B}$
  end for
  Construct preference pairs $\mathcal{D}_P = \{(\mathbf{h}_t, \mathbf{a}_t^w, \mathbf{a}_t^l)\}_{t=1}^{T-1}$ where $\mathbf{h}_t \sim \mathcal{D}_P$. For each node at step level t, compare each pair of child nodes, and construct the pair of generated actions $(\mathbf{a}^w, \mathbf{a}^l)$ if the values of taking the action, $|Q(\mathbf{h}_t, \mathbf{a}^w) - Q(\mathbf{h}_t, \mathbf{a}^l)| > \theta_{\text{threshold}}$, where $Q(\mathbf{h}_t, \mathbf{a}^w)$ and $Q(\mathbf{h}_t, \mathbf{a}^l)$ are computed using (10)
  Optimize LLM policy π_{θ_i} using DPO objective in Eq. (5) with $\mathcal{D}_P$ and π_{ref}
end for

Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents[Putta et al., 2024]

- AgentQ achieves outperforms baselines.
- Applying MCTS at inference time yields much better performance.



SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution **[Wei et al., 2025]**

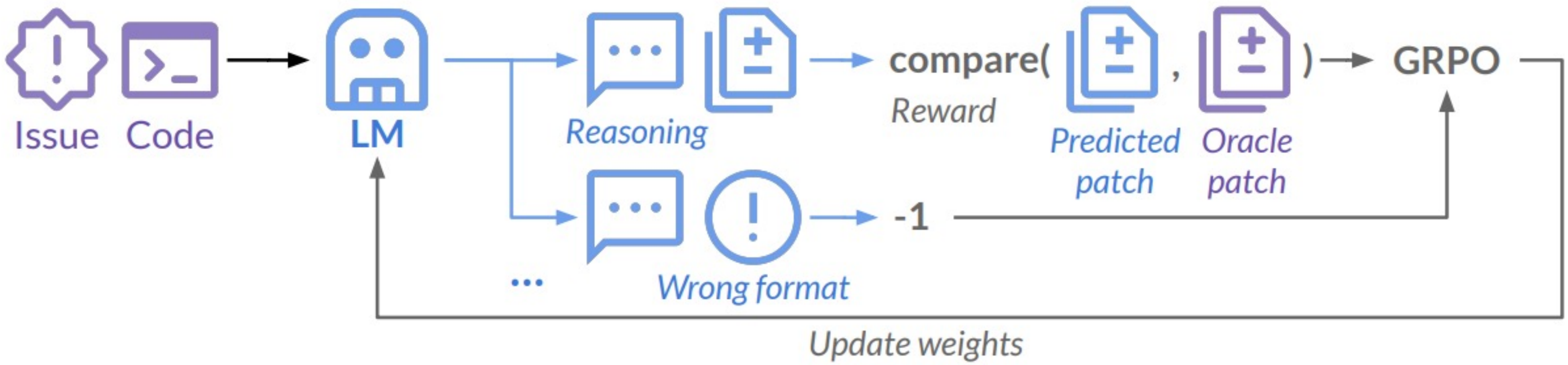
- Improving reasoning capability via RL in code domain → Improved software engineering capability.
- Defined reward as a similarity between generated patch and oracle patch, and then trained the Reasoning LM via GRPO.

$$\mathcal{R}(\tau) = \begin{cases} -1, & \text{if the format is wrong,} \\ \mathit{compare}(\mathsf{patch}_{\mathsf{pred}}, \mathsf{patch}_{\mathsf{gt}}), & \text{otherwise.} \end{cases}$$

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\right)\right],$$

where $(\mathsf{issue}, \mathsf{ctx}, \mathsf{patch}_{\mathsf{gt}}) \sim \mathcal{D}_{\mathsf{seed}}$, $q = \mathsf{form\text{-}prompt}(\mathsf{issue}, \mathsf{ctx})$, and $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$.

SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution **[Wei et al., 2025]**



- Does not involve any multi-turn optimization, only optimizing “reasoning” required for generating patch file given problematic code and issue description.

- Surprisingly, the trained LLM not only improved code modification capability, but also capable of navigating codebase (e.g., opening file, creating file) as an agent.

SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution [Wei et al., 2025]

- Improved performance in SWE-Bench (best performance among <100B scale model)

Model	Scaffold	SWE-bench Verified	Reference
	Model closed-source or size $\gg$ 100B		
GPT-4o	SWE-agent	23.2	Yang et al. (2024b)
Claude-3.5-Sonnet	SWE-agent	33.6	Yang et al. (2024b)
GPT-4o	Agentless	38.8	Xia et al. (2024)
o1-preview	Agentless	41.3	OpenAI (2024b)
DeepSeek-V3[1]	Agentless	42.0	DeepSeek-AI (2024)
Claude-3.5-Sonnet	AutoCodeRover-v2.0	46.2	Zhang et al. (2024)
Claude-3.5-Sonnet	Tools	49.0	Anthropic (2024b)
DeepSeek-R1[1]	Agentless	49.2	DeepSeek-AI (2025)
Claude-3.5-Sonnet	Agentless	50.8	Xia et al. (2024)
Claude-3.5-Sonnet	OpenHands	53.0	Wang et al. (2024)
	Model size $\leq$ 100B		
SWE-Llama-13B	RAG	1.2	Jimenez et al. (2023)
SWE-Llama-7B	RAG	1.4	Jimenez et al. (2023)
Lingma-SWE-GPT-7B	SWE-SynInfer	18.2	Ma et al. (2024)
Lingma-SWE-GPT-72B	SWE-SynInfer	28.8	Ma et al. (2024)
SWE-Fixer-72B	SWE-Fixer	30.2	Xie et al. (2025)
Llama3-Midtrain-8B (beta)[2]	**Agentless Mini**	**31.0**	**Appendix C**
SWE-Gym-32B	OpenHands	32.0	Pan et al. (2024)
Llama3-SWE-RL-70B	**Agentless Mini**	**41.0**	**This paper**

SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution [Wei et al., 2025]

- Generalization to unseen tasks (code reasoning / math / MMLU etc..)
- As a baseline, utilized SFT, which simply trains LLMs with oracle data (without reinforcement learning).

Category **Benchmark**	Llama-3.3-70B-Instruct	Llama3-SWE-SFT-70B	**Llama3-SWE-RL-70B**
Function coding			
HumanEval+	76.2	73.2	**79.9**
Library use			
BigCodeBench-Hard (I)	**28.4**	25.7	**28.4**
BigCodeBench-Hard (C)	**29.1**	24.3	**29.1**
Code reasoning			
CRUXEval-I	60.5	68.4	**71.6**
CRUXEval-O	61.9	75.1	**75.5**
Math			
MATH (strict)	63.2	54.0	**73.7**
MATH (lenient)	70.9	71.7	**73.7**
General			
MMLU	86.49	85.26	**86.82**

References

[Guo et al., 2023] What can Large Language Models do in chemistry? A comprehensive benchmark on eight tasks, NeurIPS 2023 Track on Datasets and Benchmarks
link: https://arxiv.org/abs/2305.18365

[Beltagy et al, 2019] SciBERT: A Pretrained Language Model for Scientific Text, EMNLP 2019
link: https://arxiv.org/abs/1903.10676

[Develin et al., 2019] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL 2019
link: https://arxiv.org/abs/1810.04805

[Taylor et al., 2022] Galactica: A Large Language Model for Science, arXiv 2022
link: https://arxiv.org/abs/2101.03288

[Lewkowycz et al., 2022] Solving Quantitative Reasoning Problems with Language Models, NeurIPS 2022
link: https://arxiv.org/abs/2206.14858

[Raffel et al., 2020] Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, JMLR 2020
link: https://arxiv.org/abs/1910.10683

[Christofidellis et al., 2022] Unifying Molecular and Textual Representations via Multi-task Language Modeling, ICML 2023
link: https://arxiv.org/abs/2301.12586

[Chen et al., 2022] From Artificially Real to Real: Leveraging Pseudo Data from Large Language Models for Low-Resource Molecule Discovery, AAAI 2024
link: https://arxiv.org/html/2309.05203v3

[Gal et al., 2022] An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion, ICLR 2023
link: https://arxiv.org/abs/2208.01618

[Pei et al., 2023] BioT5: Enriching Cross-modal Integration in Biology with Chemical Knowledge and Natural Language Associations [Pei et al., 2023]
link: https://arxiv.org/abs/2310.07276

References

[Lewkowycz et al., 2022] Solving Quantitative Reasoning Problems with Language Models, NeurIPS 2022
link: https://arxiv.org/abs/2206.14858

[Gao et al., 2023] PAL: Program-aided Language Models, ICML 2023
link: https://arxiv.org/abs/2211.10435

[Gou et. al., 2024] ToRA: A Tool-Integrated Reasoning Agent Models, ICLR 2024
link: https://arxiv.org/abs/2309.17452

[Cobbe et. al., 2021] Training Verifiers to Solve Math Word Problems, OpenAI 2021
link: https://arxiv.org/abs/2110.14168

[Lightman et. al., 2023] Let's Verify Step by Step, ICLR 2024
link: https://arxiv.org/abs/2305.20050

[Wang et. al., 2024] Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations, ACL 2024
link: https://arxiv.org/abs/2312.08935

Learning to Reason with LLMs, OpenAI 2024
link: https://openai.com/index/learning-to-reason-with-llms/

AI achieves silver-medal standard solving International Mathematical Olympiad problems, Deepmind 2024
link: https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

[Yang et al., 2024] Formal Mathematical Reasoning: A New Frontier in AI, arXiv 2024
link: https://arxiv.org/abs/2412.16075

[Chervonyi et al., 2024] Gold-medalist Performance in Solving Olympiad Geometry with AlphaGeometry2, Deepmind 2024
link: https://arxiv.org/abs/2502.03544

References

[Dinh et al., 2022] LIFT: Language-Interfaced Fine-Tuning for Non-Language Machine Learning Tasks, NeurIPS 2022
link: https://arxiv.org/abs/2206.06565

[Manikandan et al., 2023] Language models are weak learners, NeurIPS 2023
link: https://arxiv.org/abs/2306.14101

[Yan et al., 2024] Making Pre-trained Language Models Great on Tabular Prediction, ICLR 2024
link: https://arxiv.org/abs/2403.01841

[Han et al., 2024] Large Language Models Can Automatically Engineer Features for Few-shot Tabular Learning, ICML 2024
link: https://arxiv.org/abs/2404.09491

[Nam et al., 2024] Tabular Transfer Learning via Prompting LLMs, COLM 2024
link: https://arxiv.org/abs/2408.11063

[Kim et al., 2024] An Efficient Tokenization for Molecular Language Models, NeurIPSW-AIDrugX 2024
link: https://openreview.net/forum?id=BDISzo0dZi

[Nam et al., 2024] Optimized Feature Generation for Tabular Data via LLMs with Decision Tree Reasoning, NeurIPS 2024
link: https://arxiv.org/abs/2406.08527

[Xue & Salim, 2023] PromptCast: A New Prompt-based Learning Paradigm for Time Series Forecasting, TKDE 2023
link: https://arxiv.org/abs/2210.08964

[Gruver et al., 2023] Large Language Models are Zero-Shot Time Series Forecasters, NeurIPS 2023
link: https://arxiv.org/abs/2406.12031

[Gardner et al., 2024] Large Scale Transfer Learning for Tabular Data via Language Modeling, NeurIPS 2024
link: https://arxiv.org/abs/2310.01728

[Jin et al., 2024] Time-LLM: Time Series Forecasting by Reprogramming Large Language Models, ICLR 2024
link: https://arxiv.org/abs/2310.01728

[Ansari et al ., 2024] Chronos: Learning the Language of Time Series, TMLR 2024
link: https://arxiv.org/abs/2403.07815

References

[Liu et al., 2023] AgentBench: Evaluating LLMs as Agents
link: https://arxiv.org/abs/2308.03688

[Zhou et al., 2023] WebArena: A Realistic Web Environment for Building Autonomous Agents
link: https://arxiv.org/abs/2307.13854

[Yao et al., 2023] Synergizing Reasoning and Acting in Language Models
link: https://arxiv.org/abs/2210.03629

[Shinn et al., 2023] Reflexion: Language Agents with Verbal Reinforcement Learning
link: https://arxiv.org/abs/2303.11366

[Zhou et al., 2024] Training Language Model Agents via Hierarchical Multi-turn RL
link: https://arxiv.org/abs/2402.19446

[Putta et al., 2024] Agent Q: Advanced Reasoning and Learning for Autonomous AI Agents
link: https://arxiv.org/abs/2408.07199

[Zhang et al., 2023] Bootstrap Your Own Skills: Learning to Solve New Tasks with Large Language Model Guidance
link: https://arxiv.org/abs/2310.10021

[Tsai et al., 2025] AnoLLM: Large Language Models for Tabular Anomaly Detection, ICLR 2025.
link: https://openreview.net/forum?id=7VkHffT5X2

[Yan et al., 2025] Small Models are LLM Knowledge Triggers for Medical Tabular Prediction, ICLR 2025.
link: https://openreview.net/forum?id=WoPovNkM5h

[Jimenez et al., 2023] SWE-bench: Can Language Models Resolve Real-World GitHub Issues
link: https://arxiv.org/abs/2310.06770

[Wei et al., 2025] SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution
link: https://arxiv.org/abs/2502.18449

[Chen et al., 2024] CodeR: Issue Resolving with Multi-Agent and Task Graphs
link: https://arxiv.org/abs/2406.01304