Black-Box Tuning for Language-Model-as-a-Service

Tianxiang Sun
School of Computer Science, Fudan University

https://txsun1997.github.io/

Acknowledgment





Tianxiang Sun[1] Yunfan Shao[1] Hong Qian[2] Xuanjing Huang[1] Xipeng Qiu[1,3]


Yang Yu[4] Zhengfu He[1]

[1] Fudan University [2] East China Normal University [3] Peng Cheng Laboratory [4] Nanjing University

Acknowledgment

Tianxiang Sun[1]

Yunfan Shao[1]

Hong Qian[2]

Xuanjing Huang[1]

Xipeng Qiu[1,3]

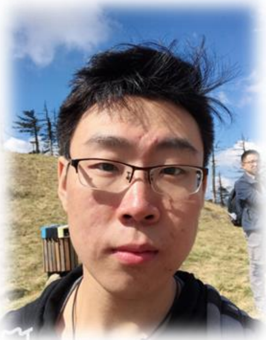
NLP/Deep Learning

Yang Yu[4]

Zhengfu He[1]

[1] Fudan University [2] East China Normal University [3] Peng Cheng Laboratory [4] Nanjing University

Acknowledgment



Tianxiang Sun[1]

Yunfan Shao[1]

Hong Qian[2]

Xuanjing Huang[1]

Xipeng Qiu[1,3]

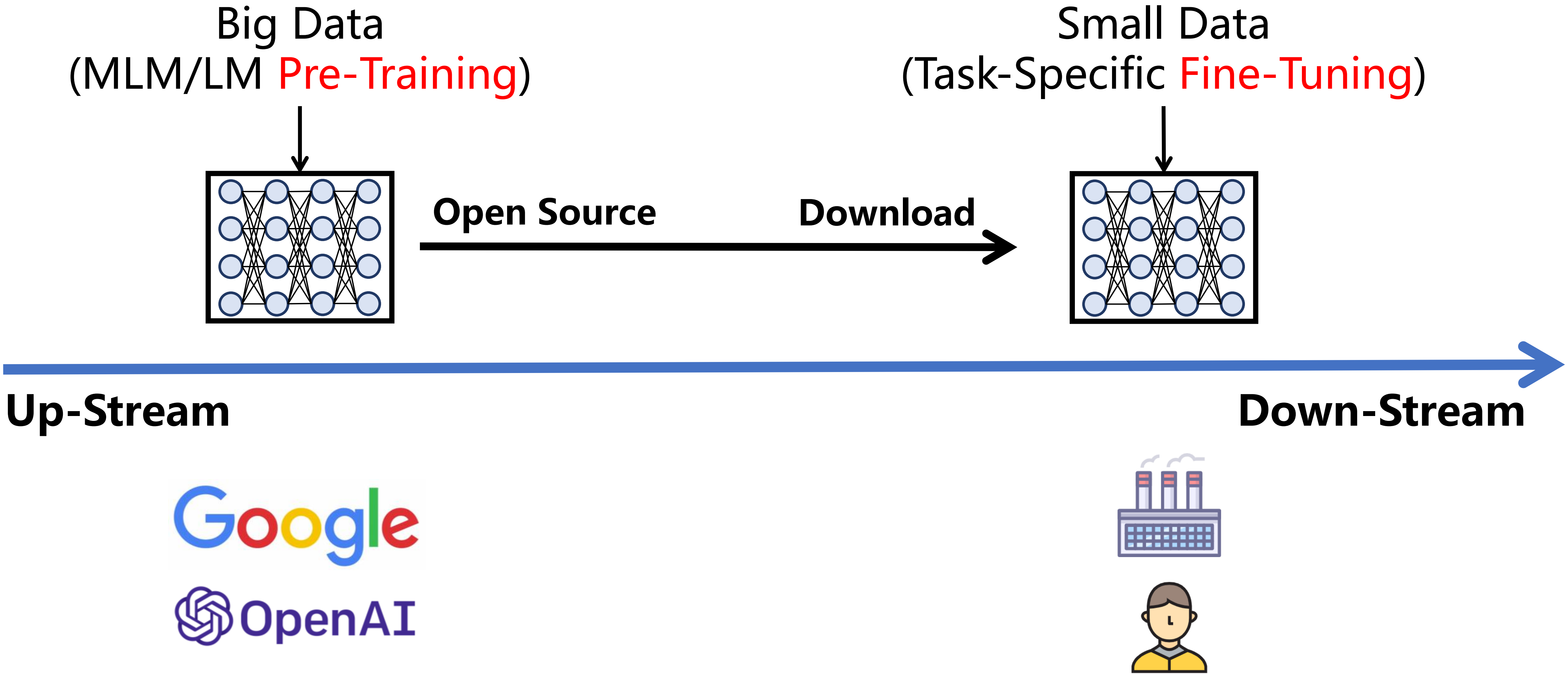
RL/Black-Box Optimization

Yang Yu[4]

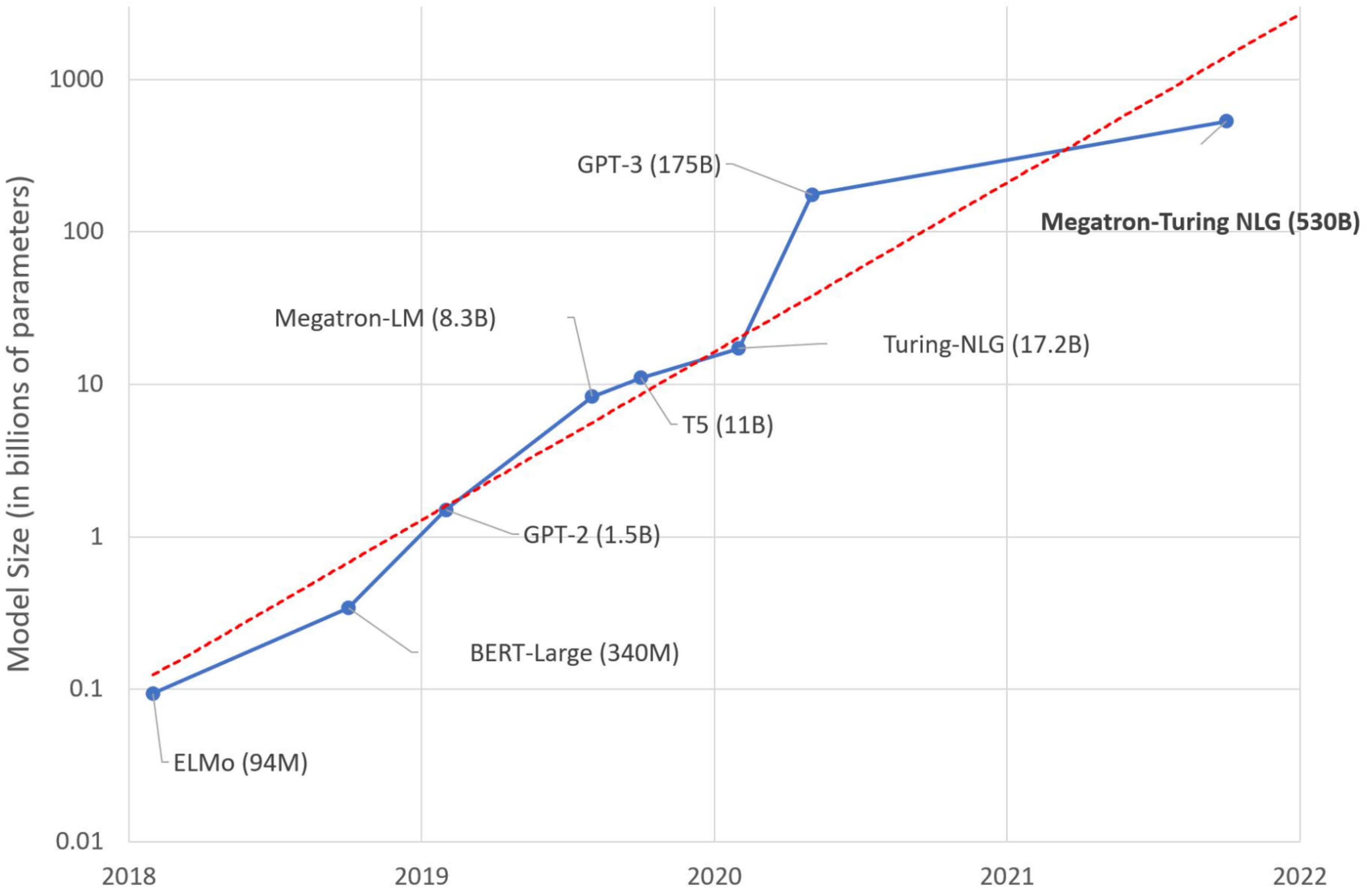
Zhengfu He[1]

[1] Fudan University [2] East China Normal University [3] Peng Cheng Laboratory [4] Nanjing University

Pre-train, then fine-tune



When language models become larger…



When language models become larger…

In the era of large language models (LLMs)…

- **Servers** often do not open-source the weights of LLMs due to commercial reasons
- **Users** usually do not have enough resources to run LLMs

When language models become larger…

Why did OpenAI choose to release an API instead of open-sourcing the models?

There are three main reasons we did this. First, commercializing the technology helps us pay for our ongoing AI research, safety, and policy efforts.

Second, many of the models underlying the API are very large, taking a lot of expertise to develop and deploy and making them very expensive to run. This makes it hard for anyone except larger companies to benefit from the underlying technology. We're hopeful that the API will make powerful AI systems more accessible to smaller businesses and organizations.

Third, the API model allows us to more easily respond to misuse of the technology. Since it is hard to predict the downstream use cases of our models, it feels inherently safer to release them via an API and broaden access over time, rather than release an open source model where access cannot be adjusted if it turns out to have harmful applications.

When language models become larger...

In the era of large language models (LLMs)...

- **Servers** often do not open-source the weights of LLMs due to commercial reasons
- **Users** usually do not have enough resources to run LLMs

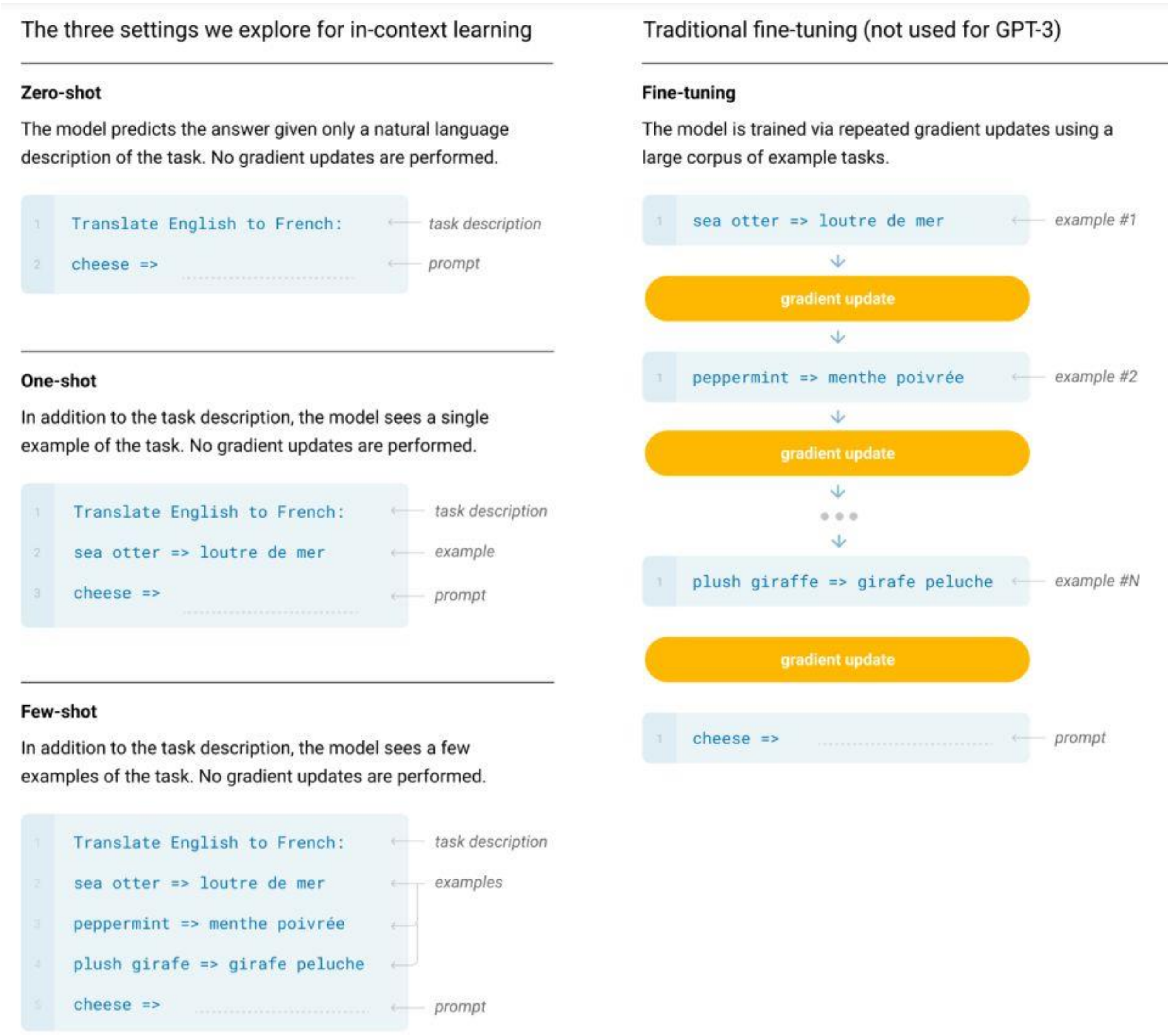
The emergent ability of LLMs

- Manually craft text prompt to query LLMs
- In-context learning (GPT-3, Brown et al., 2020)

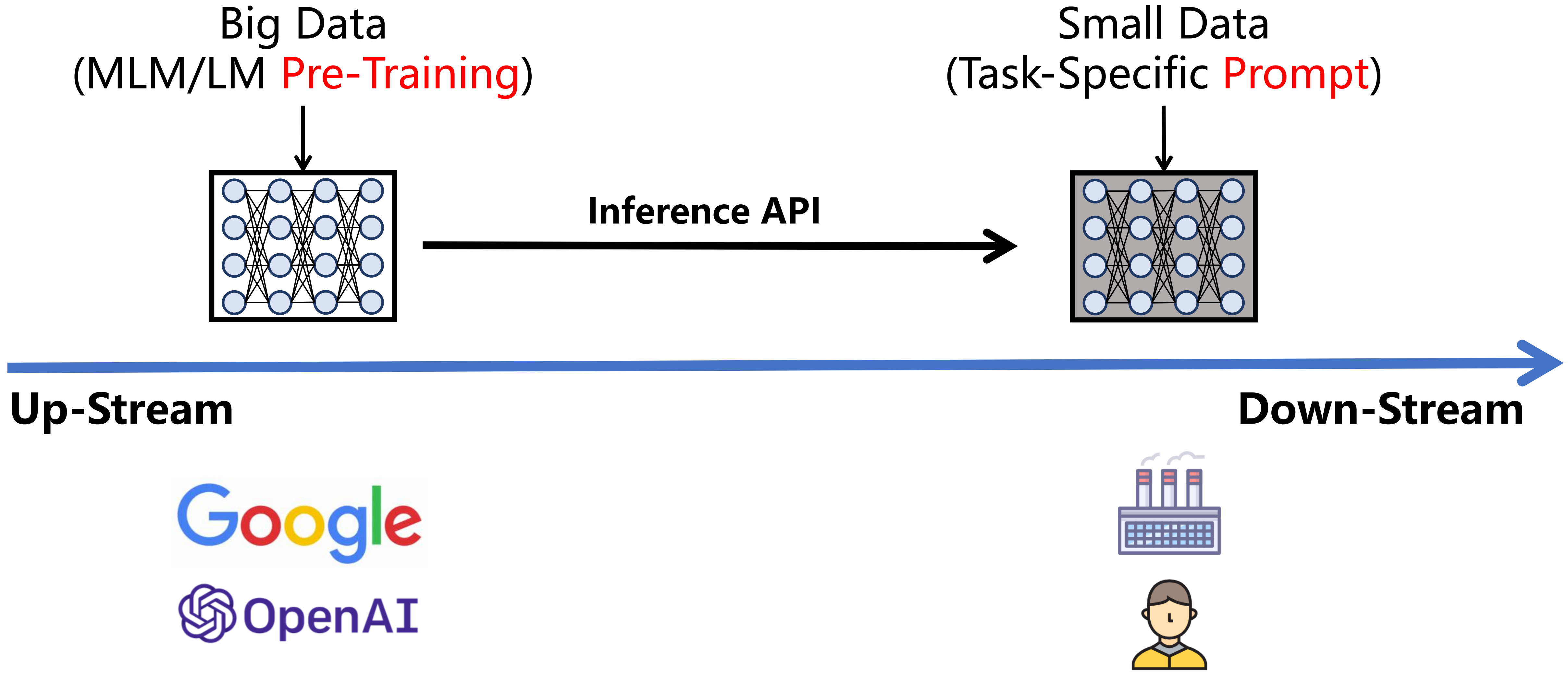
When language models become larger…

Why in-context learning?

- Generalization: One general purpose model for all tasks
- Backpropagation is expensive
- Commercial use



Language-Model-as-a-Service (LMaaS)



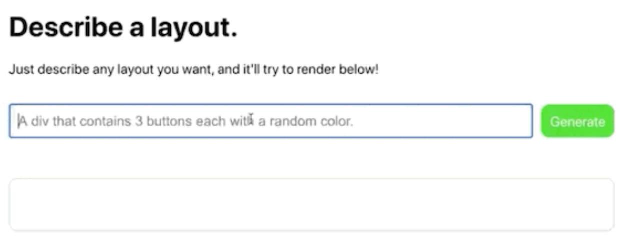
Language-Model-as-a-Service (LMaaS)

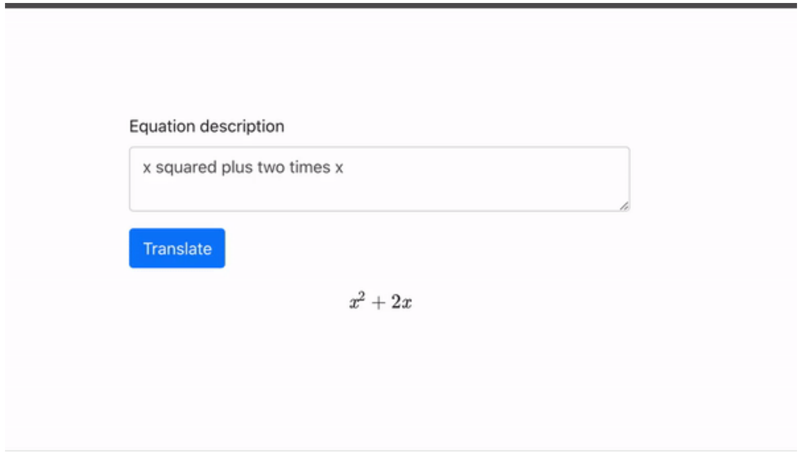
GPT-3 Pricing

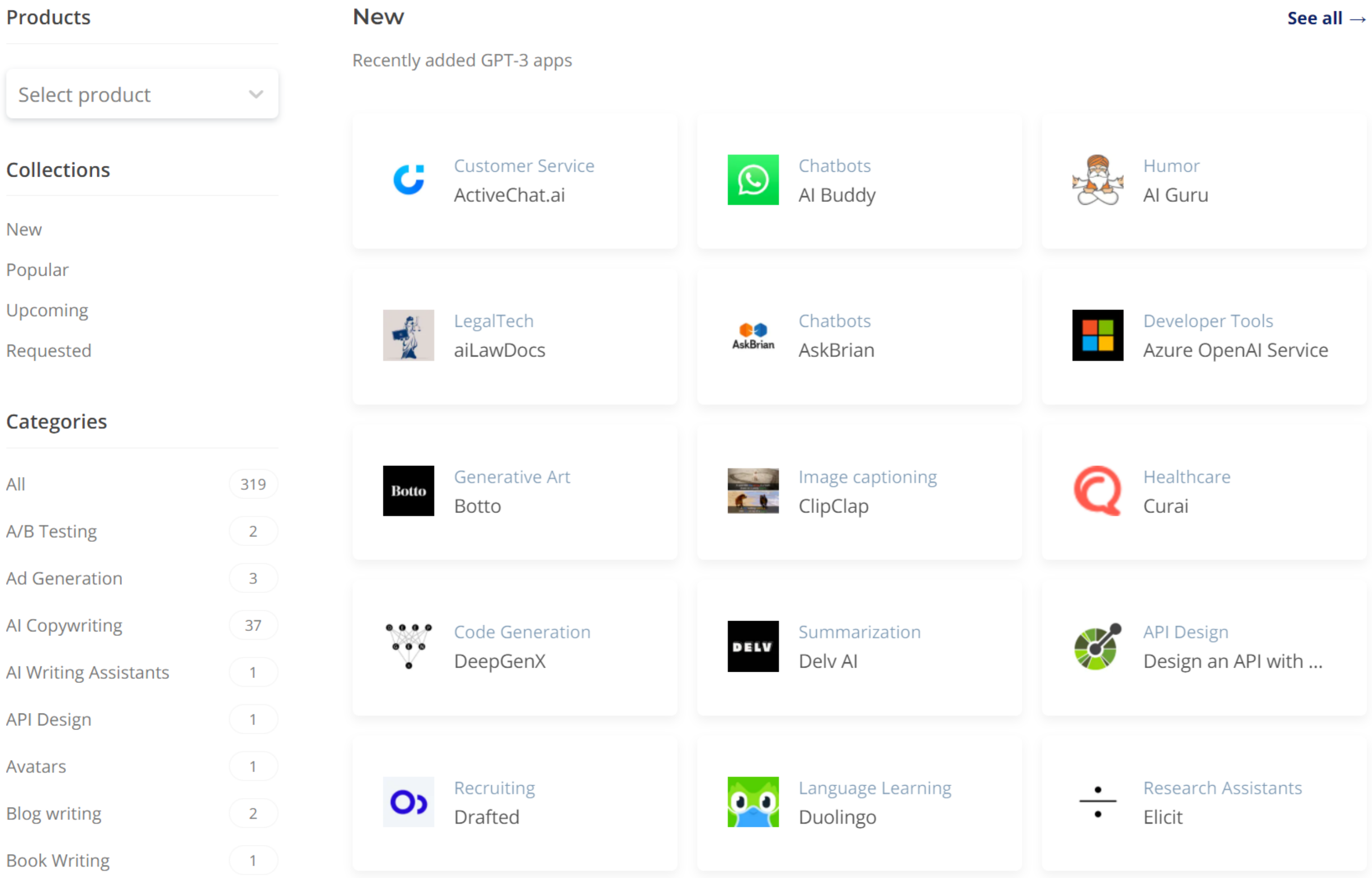
Per-model prices

Ada Fastest	Babbage	Curie	Davinci Most powerful
$0.0008 /1K tokens	$0.0012 /1K tokens	$0.0060 /1K tokens	$0.0600 /1K tokens

Language-Model-as-a-Service (LMaaS)







https://gpt3demo.com/

LMaaS in China

赛题说明

INTRODUCE

本次大赛的主题为基于悟道2.0大模型的创新应用开发，面向在校大学生、企业工程师、科研工作者等全球开发者全面开放。参赛选手需要依据对悟道能力的理解，结合社会关注的热点选择健康医疗、教育学习、社交生活、效率工具、环境自然或其他具有社会价值、产业价值的相关主题，提交一个潜在的智能应用方案并上线应用。

API 文档说明

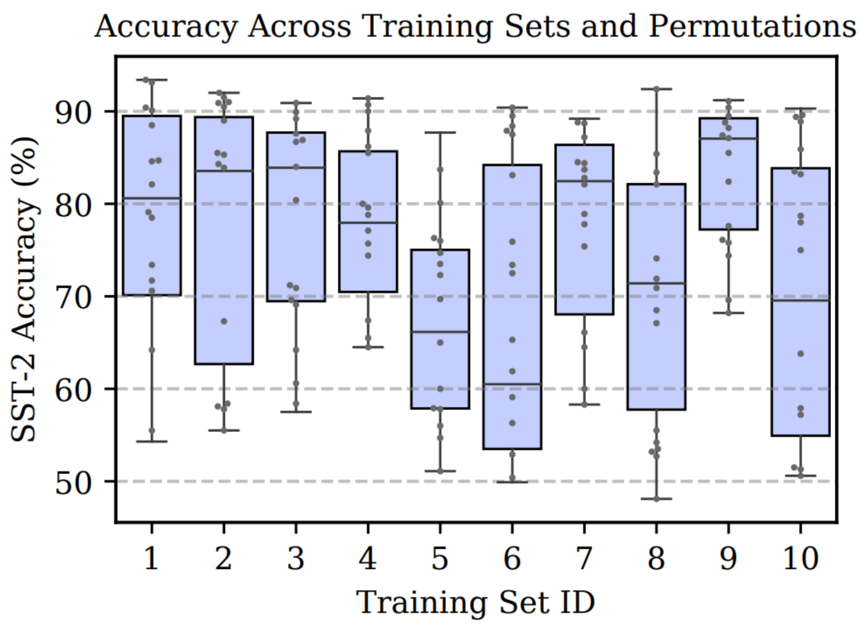
本次比赛共提供 9 个应用的 API 接口，每支队伍每天访问次数有限。

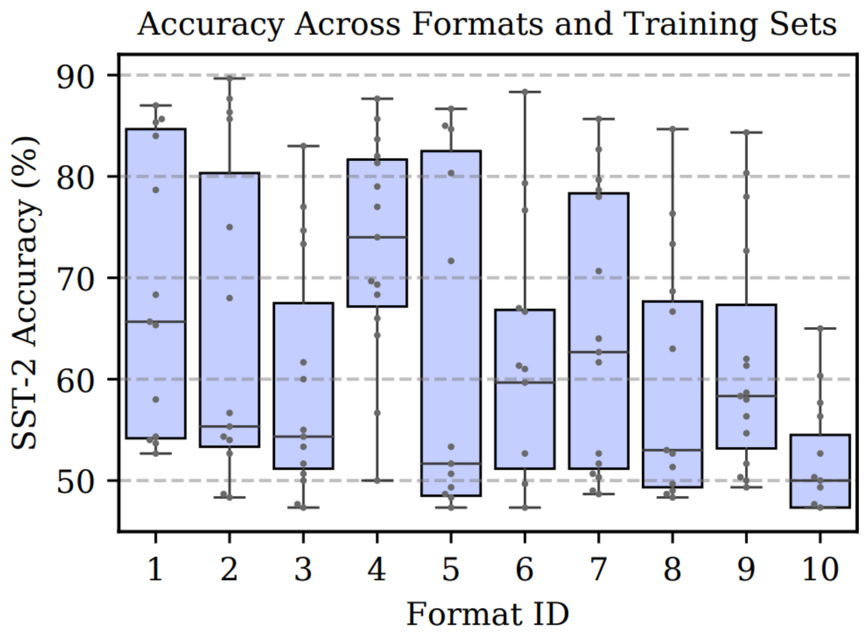
- ✓ CogView API 文档，100 次 / 天。
- ✓ 藏头诗 API 文档，100 次 / 天。
- ✓ 获取图像的特征向量，1000 次 / 天。
- ✓ 获取文本的特征向量，1000 次 / 天。
- ✓ 快速写诗 API 文档，100 次 / 天。
- ✓ 宋词 API 文档，100 次 / 天。
- ✓ 问答 API 文档，100 次 / 天。
- ✓ 写诗 API 文档，100 次 / 天。
- ✓ 新闻 API 文档，100 次 / 天。

https://www.biendata.xyz/wudao/

However...

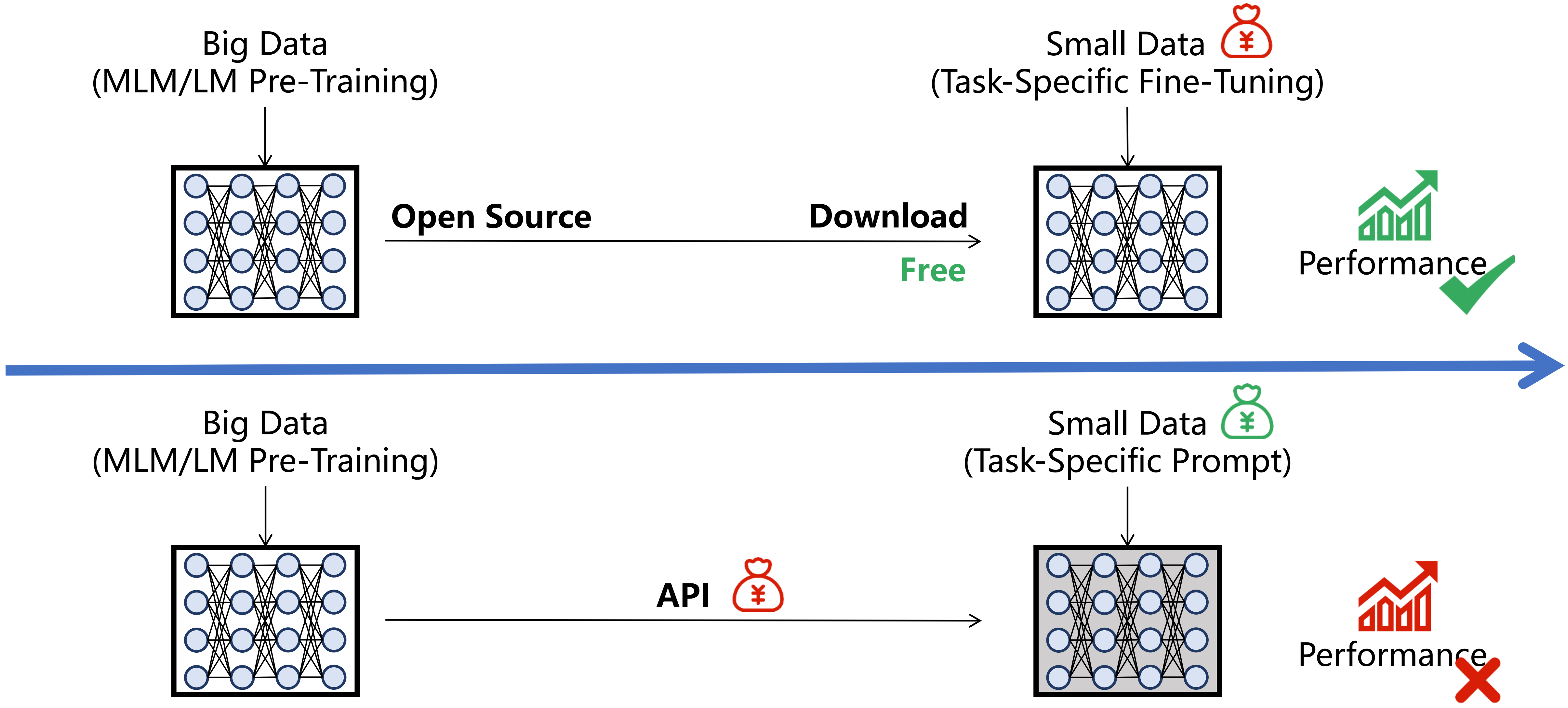
The performance of manual prompt and in-context learning highly depend on the choice of prompt and demonstrations, and lags far behind model tuning.





Zhao et al. Calibrate Before Use: Improving Few-Shot Performance of Language Models. ICML 2021

Grounding LLMs From the Cloud



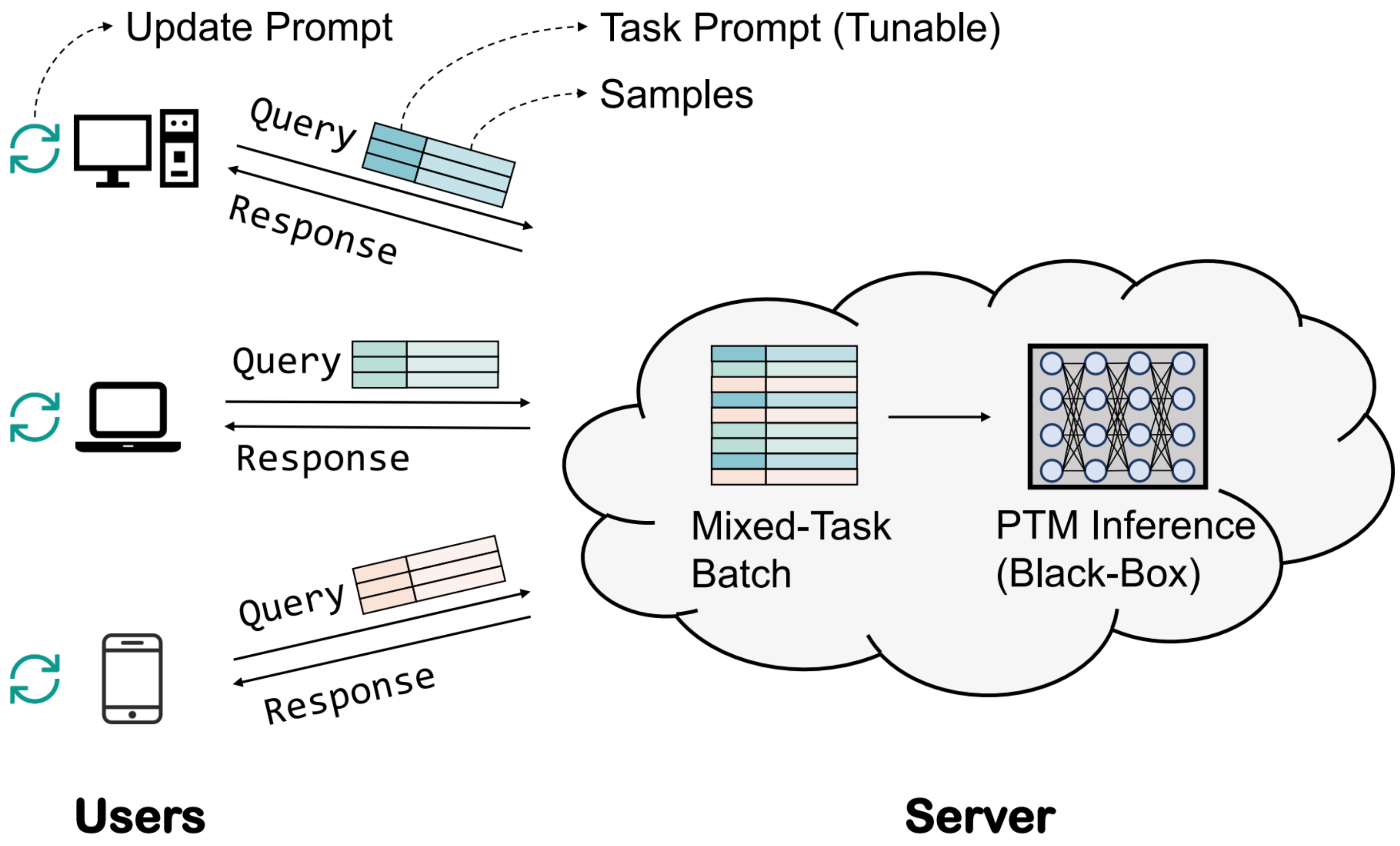
Performance is the Key for grounding. (Who are the users?)

To make LLMs benefit more people...

Can we optimize the prompt with the API feedback? (without expensive backpropagation)

Objective:

$$\mathbf{p}^{\star} = \arg\min_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(f(\mathbf{p}; \tilde{X}), \tilde{Y})$$



A challenge of high dimensionality

Considering optimization of the continuous prompt, the dimensionality can be tens of thousands (say we are going to optimize 50 prompt tokens, each with 1k dimensions, there are 50k parameters to be optimized.)

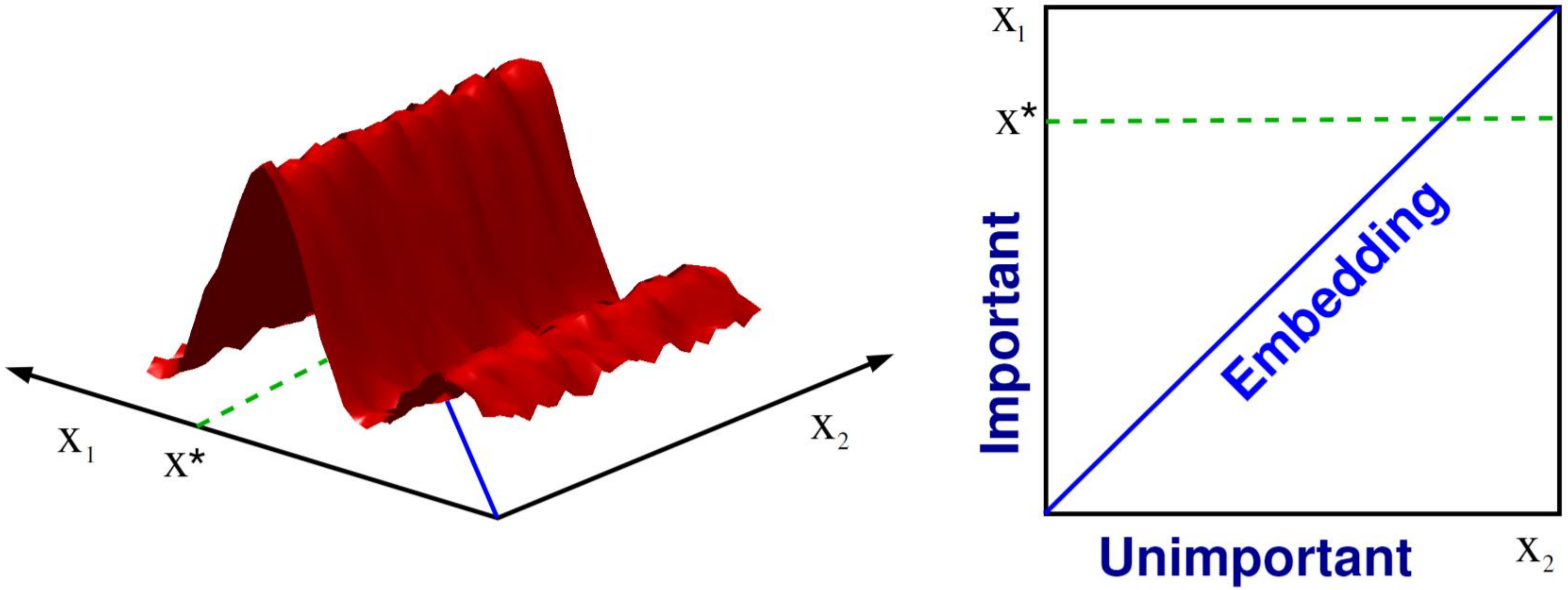
Derivative-free optimization (DFO) can struggle with high-dimensional problems, **except for** the case when the problem has a low intrinsic dimensionality.

Note: Intrinsic dimensionality is the minimal number of parameters needed to represent the problem

A challenge of high dimensionality

An example

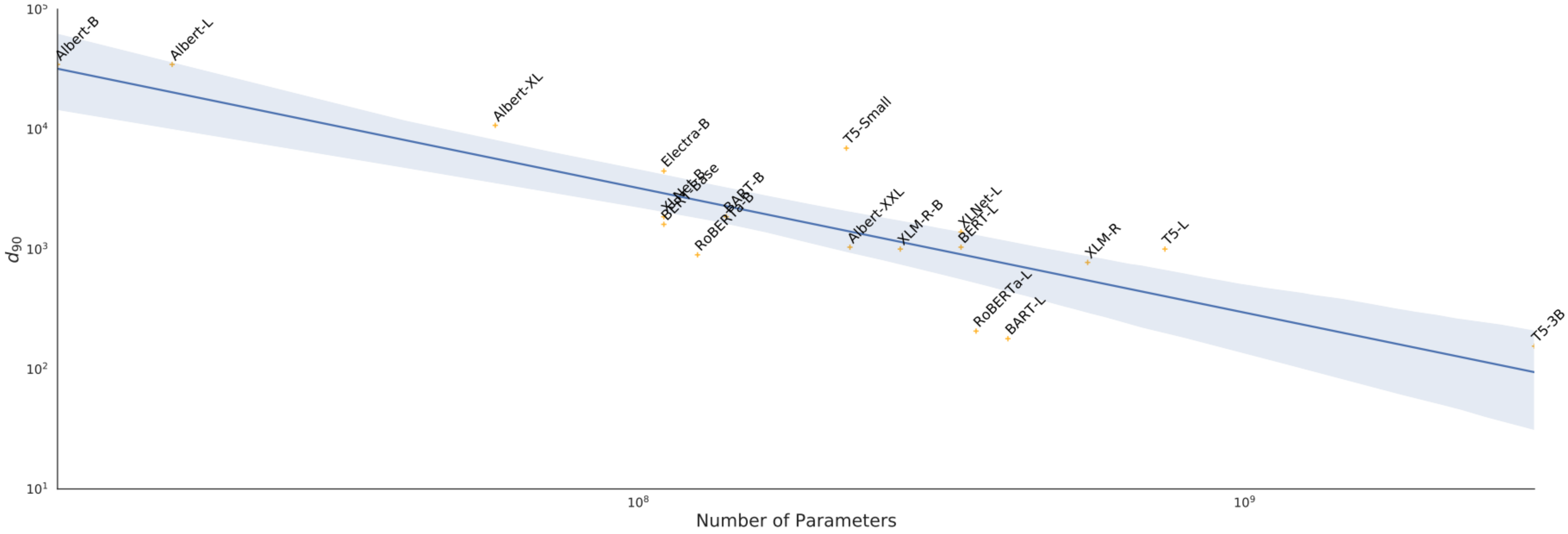
- The objective to be optimized has two dimensions but only one matters
- In that case we can perform optimization with random embedding



Wang et al. Bayesian optimization in a billion dimensions via random embeddings. J. Artif. Intell. Res. 2016

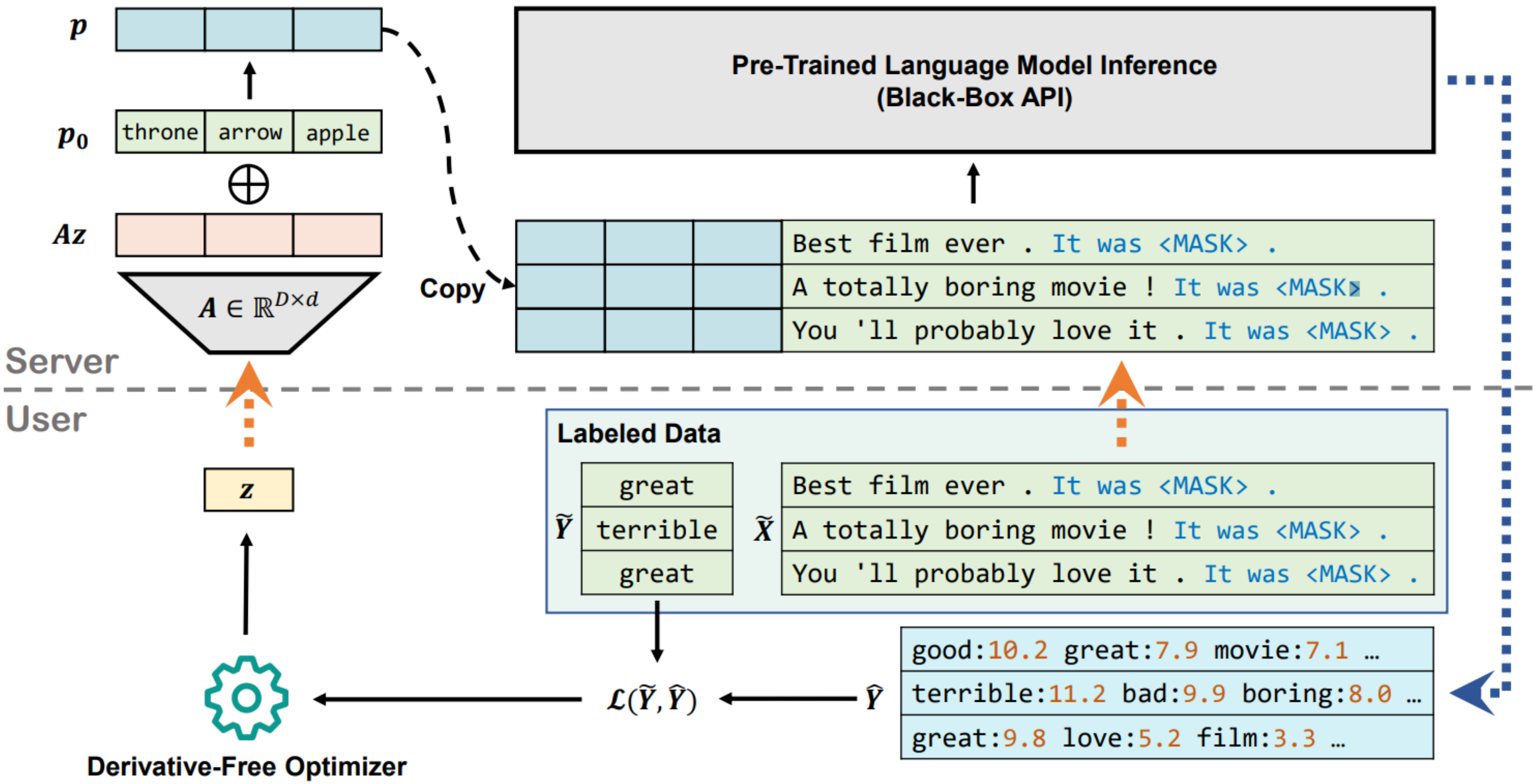
Fortunately...

LLMs have a very low intrinsic dimensionality!



Aghajanyan et al. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. ACL 2021

Black-Box Tuning



Black-Box Tuning

The CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

The CMA-ES (**E**volution **S**trategy with **C**ovariance **M**atrix **A**daptation)

Consider $P^{(t)} = \mathcal{N}\left(\boldsymbol{\mu}^{(t)}, {\sigma^{(t)}}^2 \boldsymbol{C}^{(t)}\right)$ where $\boldsymbol{\mu}^{(t)} \in \mathbb{R}^n$, $\sigma^{(t)} \in \mathbb{R}_+$, $\boldsymbol{C}^{(t)} \in \mathbb{R}^{n \times n}$

- $\boldsymbol{\mu}^{(t)} \rightarrow \boldsymbol{\mu}^{(t+1)}$: Maximum likelihood update, i.e. $P(\boldsymbol{x}^{(t)}_{\text{selected}} | \boldsymbol{\mu}^{(t+1)}) \rightarrow \max$
- $\boldsymbol{C}^{(t)} \rightarrow \boldsymbol{C}^{(t+1)}$: Maximum likelihood update, i.e. $P(\frac{\boldsymbol{x}^{(t)}_{\text{selected}} - \boldsymbol{\mu}^{(t)}}{\sigma^{(t)}} | \boldsymbol{C}^{(t+1)}) \rightarrow \max$, under consideration of prior $\boldsymbol{C}^{(t)}$ (otherwise $\boldsymbol{C}^{(t+1)}$ becomes singular).
- $\sigma^{(t)} \rightarrow \sigma^{(t+1)}$: Update to achieve conjugate perpendicularity, i.e. conceptually $(\boldsymbol{\mu}^{(t+2)} - \boldsymbol{\mu}^{(t+1)})^{\mathrm{T}} {\boldsymbol{C}^{(t)}}^{-1} (\boldsymbol{\mu}^{(t+1)} - \boldsymbol{\mu}^{(t)}) / {\sigma^{(t+1)}}^2 \rightarrow 0$

https://cma-es.github.io/

Experiments

Datasets and processing details

Category	Dataset	$\mid \mathcal{Y} \mid$	|Train|	|Test|	Type	Template	Label words
single-sentence	SST-2	2	67k	0.9k	sentiment	$\langle S \rangle$. It was `[MASK]`.	great, bad
	Yelp P.	2	560k	38k	sentiment	$\langle S \rangle$. It was `[MASK]`.	great, bad
	AG's News	4	120k	7.6k	topic	`[MASK]` News: $\langle S \rangle$	World, Sports, Business, Tech
	DBPedia	14	560k	70k	topic	[Category: `[MASK]`] $\langle S \rangle$	Company, Education, Artist, Athlete, Office, Transportation, Building, Natural, Village, Animal, Plant, Album, Film, Written
sentence-pair	MRPC	2	3.7k	0.4k	paraphrase	$\langle S_1 \rangle$? `[MASK]`, $\langle S_2 \rangle$	Yes, No
	RTE	2	2.5k	0.3k	NLI	$\langle S_1 \rangle$? `[MASK]`, $\langle S_2 \rangle$	Yes, No
	SNLI	3	549k	9.8k	NLI	$\langle S_1 \rangle$? `[MASK]`, $\langle S_2 \rangle$	Yes, Maybe, No

Experiments

16-shot (per class) learning with RoBERTa-large (350M)

Method	**SST-2** acc	**Yelp P.** acc	**AG's News** acc	**DBPedia** acc	**MRPC** F1	**SNLI** acc	**RTE** acc	**Avg.**
				Gradient-Based Methods				
Prompt Tuning	68.23 ±3.78	61.02 ±6.65	84.81 ±0.66	87.75 ±1.48	51.61 ±8.67	36.13 ±1.51	54.69 ±3.79	63.46
+ Pre-trained prompt	/	/	/	/	77.48 ±4.85	64.55 ±2.43	77.13 ±0.83	74.42
P-Tuning v2	64.33 ±3.05	92.63 ±1.39	83.46 ±1.01	97.05 ±0.41	68.14 ±3.89	36.89 ±0.79	50.78 ±2.28	70.47
Model Tuning	85.39 ±2.84	91.82 ±0.79	86.36 ±1.85	97.98 ±0.14	77.35 ±5.70	54.64 ±5.29	58.60 ±6.21	78.88
				Gradient-Free Methods				
Manual Prompt	79.82	89.65	76.96	41.33	67.40	31.11	51.62	62.56
In-Context Learning	79.79 ±3.06	85.38 ±3.92	62.21 ±13.46	34.83 ±7.59	45.81 ±6.67	47.11 ±0.63	60.36 ±1.56	59.36
Feature-MLP	64.80 ±1.78	79.20 ±2.26	70.77 ±0.67	87.78 ±0.61	68.40 ±0.86	42.01 ±0.33	53.43 ±1.57	66.63
Feature-BiLSTM	65.95 ±0.99	74.68 ±0.10	77.28 ±2.83	90.37 ±3.10	71.55 ±7.10	46.02 ±0.38	52.17 ±0.25	68.29
Black-Box Tuning	89.56 ±0.25	91.50 ±0.16	81.51 ±0.79	87.80 ±1.53	61.56 ±4.34	46.58 ±1.33	52.59 ±2.21	73.01
+ Pre-trained prompt	/	/	/	/	75.51 ±5.54	83.83 ±0.21	77.62 ±1.30	**83.90**

Experiments

Detailed comparison on SST-2 and AG News

	Deployment-Efficient	As-A-Service	Test Accuracy	Training Time	Memory Footprint User	Memory Footprint Server	Upload per query	Download per query
SST-2 (max sequence length: 47)								
Prompt Tuning	√	×	72.6	15.9 mins	-	5.3 GB	-	-
Model Tuning	×	×	87.8	9.8 mins	-	7.3 GB	-	-
Feature-MLP	√	√	63.8	7.0 mins	20 MB	2.8 GB	4 KB	128 KB
Feature-BiLSTM	√	√	66.2	9.3 mins	410 MB	2.8 GB	4 KB	6016 KB
Black-Box Tuning	√	√	89.4	10.1 (6.1*) mins	30 MB	3.0 GB	6 KB	0.25 KB
AG's News (max sequence length: 107)								
Prompt Tuning	√	×	84.0	30.2 mins	-	7.7 GB	-	-
Model Tuning	×	×	88.4	13.1 mins	-	7.3 GB	-	-
Feature-MLP	√	√	71.0	13.5 mins	20 MB	3.6 GB	20 KB	256 KB
Feature-BiLSTM	√	√	73.1	19.7 mins	500 MB	3.6 GB	20 KB	27392 KB
Black-Box Tuning	√	√	82.6	21.0 (17.7*) mins	30 MB	4.6 GB	22 KB	1 KB

Forward Is All You Need?

Limitations of black-box tuning:

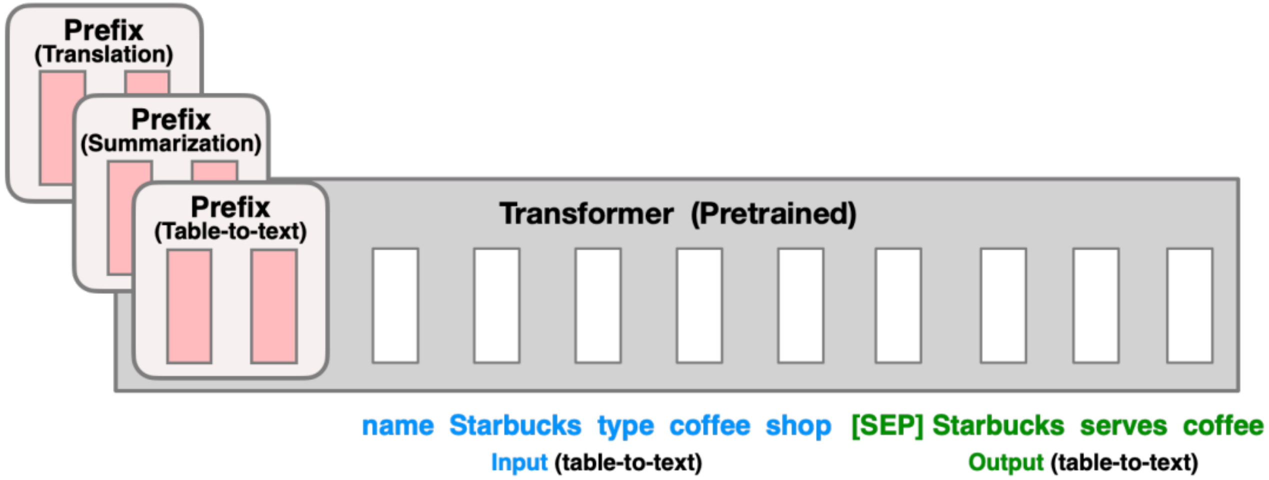
- Slow convergence on many-label classification (e.g., DBPedia)
- Requirement of prompt pre-training (gradient) on difficult tasks (e.g., SNLI)

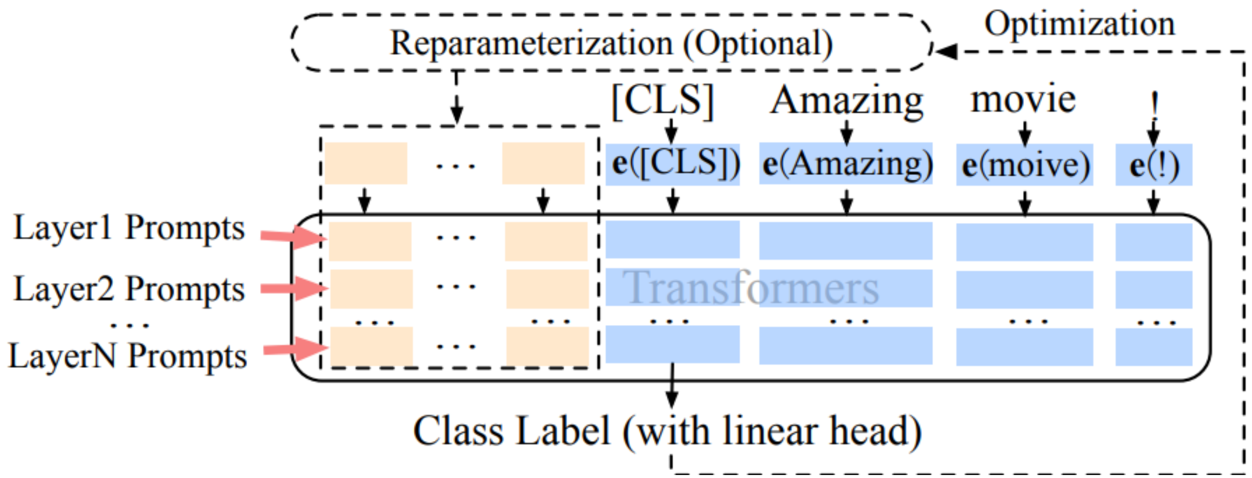
Current version of black-box tuning is just a lower bound:

- Prompt/verbalizer engineering, prompt ensemble, prompt pre-training…
- Better derivative-free algorithms
- Pre-trained random embedding
- …

Can We Go Deeper?

The ``Deep Prompt Tuning``





Prefix Tuning (Li and Liang, ACL 2021)

P-Tuning v2 (Liu et al., ACL 2022)

Can We Go Deeper?

The challenge, again, is the high dimensionality

- Say we are going to optimize 50 prompt tokens at each layer of RoBERTa-large, each with 1k dimensions, there are 50k×24=1.2M parameters to be optimized
- Besides, the prompt parameters at different layers are heterogenous and therefore we can not simply use the random embedding to solve it

Take A Closer Look Into the Forward Pass

Thanks to the residual connections in modern LLMs, the forward computation can be decomposed as an additive form

An example of a 3-layer model:

$$\begin{aligned} f(\mathbf{x}_1) &= f_3(\mathbf{x}_3) + \mathbf{x}_3 \\ &= f_3(\mathbf{x}_3) + f_2(\mathbf{x}_2) + \mathbf{x}_2 \\ &= f_3(\mathbf{x}_3) + f_2(\mathbf{x}_2) + f_1(\mathbf{x}_1) + \mathbf{x}_1 \end{aligned}$$

Therefore, the optimization can be decomposed into multiple sub-problems!

Take A Closer Look Into the Forward Pass

A general formulation of ``deep black-box tuning``:

$$f(\mathbf{x}_1, \mathbf{p}) = [\mathbf{A}_1 \mathbf{z}_1 + \mathbf{p}_1^0; \mathbf{x}_1] + \sum_{j=1}^{L} f_j([\mathbf{A}_j \mathbf{z}_j + \mathbf{p}_j^0; \mathbf{x}_j])$$

Given such an additive form, we propose a divide-and-conquer (DC) algorithm to alternately optimize prompt at each layer

Divide-and-Conquer

- Layer-specific optimizer
- Layer-specific random projection
- Alternate from the bottom to top

Algorithm 1: DC Algorithm for BBTv2

Require: L-layer PTM Inference API f,
Loss function $\mathcal{L}$,
Budget of API calls $\mathcal{B}$,
Derivative-free optimizers $\{\mathcal{M}_j\}_{j=1}^{L}$

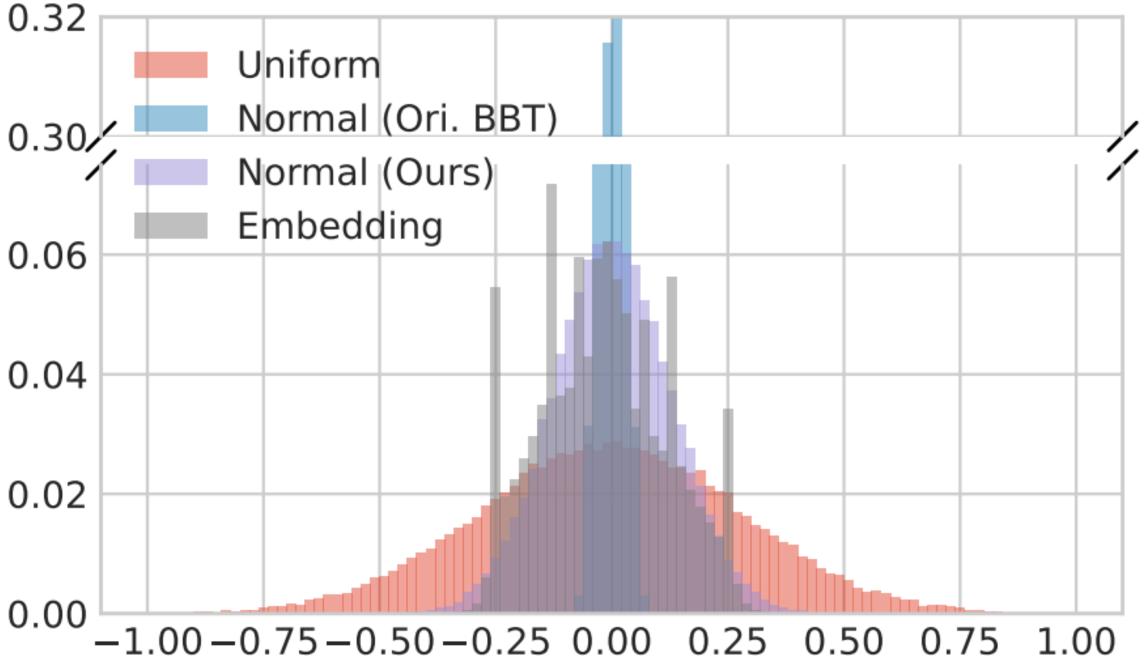
Initialize random projections $\mathbf{A}_1, \ldots, \mathbf{A}_L$
Initialize parameters $\mathbf{z}_1^{(0)}, \ldots, \mathbf{z}_L^{(0)}$
Deep prompts $\mathbf{p} = \langle \mathbf{A}_1\mathbf{z}_1^{(0)}, \ldots, \mathbf{A}_L\mathbf{z}_L^{(0)} \rangle$
for $i = 1$ to $\mathcal{B}/L$ **do**
  for $j = 1$ to L **do**
    Evaluate: $loss = \mathcal{L}(f(\mathbf{p}))$
    Update: $\mathbf{z}_j^{(i)} \leftarrow \mathcal{M}_j(\mathbf{z}_j^{(i-1)}, loss)$
    Replace: $\mathbf{p}_j \leftarrow \mathbf{A}_j\mathbf{z}_j^{(i)}$
  end for
end for
return Optimized deep prompts $\mathbf{p}$

Revisiting Random Projection (Embedding)

Generating random projections from a normal distribution with std dev as

$$\sigma_A = \frac{\hat{\sigma}}{\sqrt{d}\sigma_z}$$

A visualization of generated prompt with RoBERTa-large



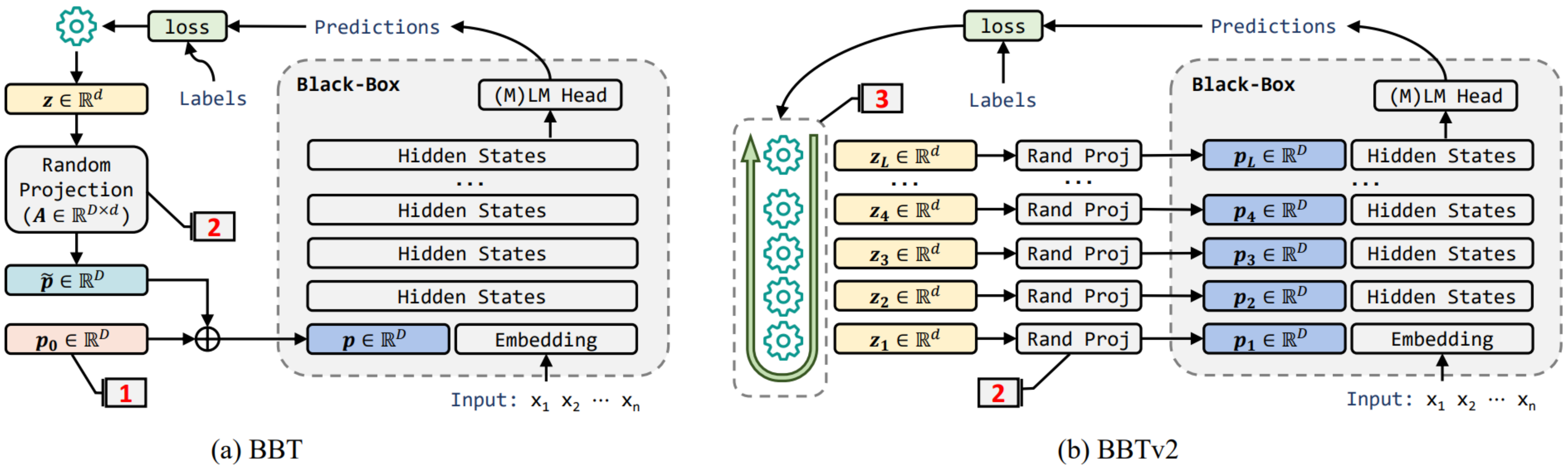
BBTv2: Towards A Gradient-Free Future

Main improvements of BBTv2

- Get rid of prompt pre-training
- Improved random projection
- Deep prompts

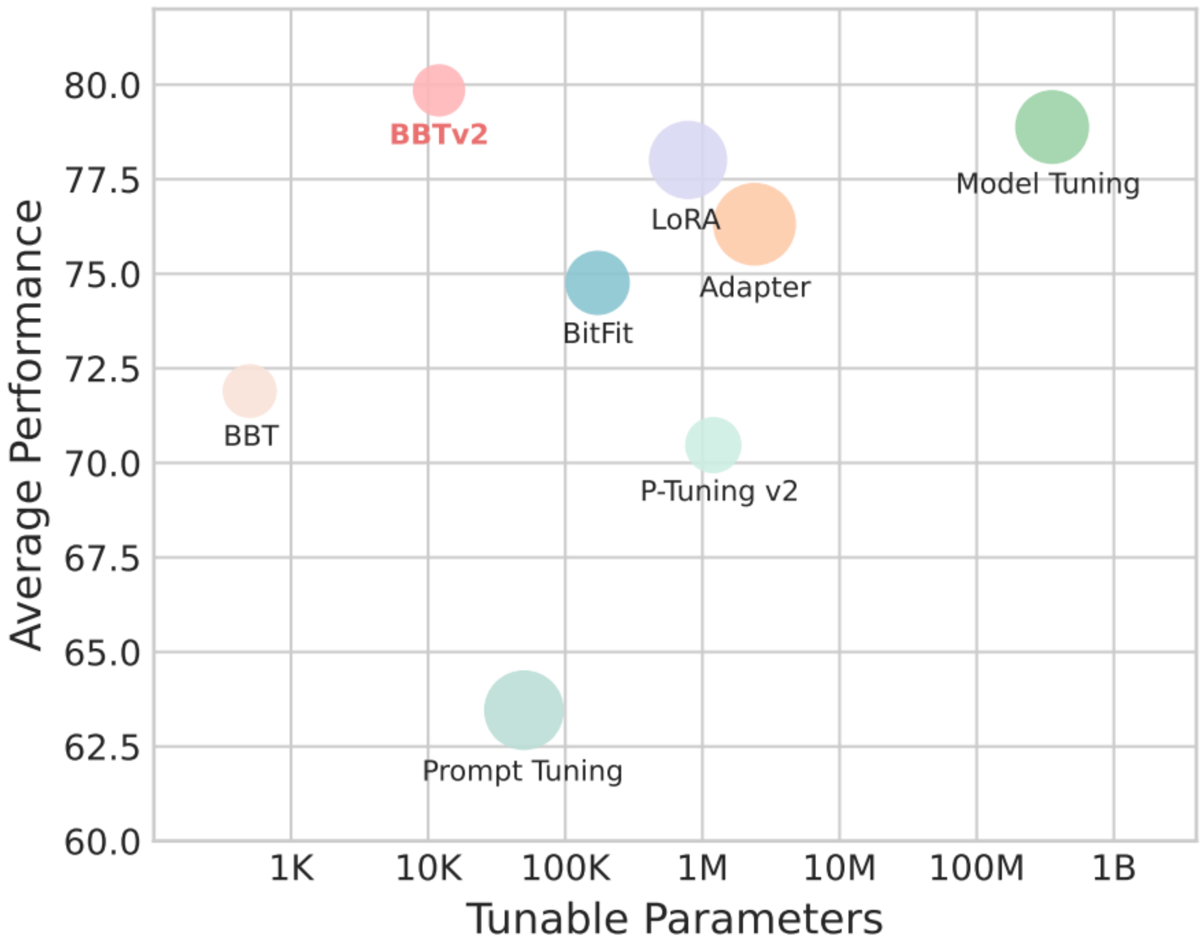
BBTv2: Towards A Gradient-Free Future

Main improvements of BBTv2



Experiments of BBTv2

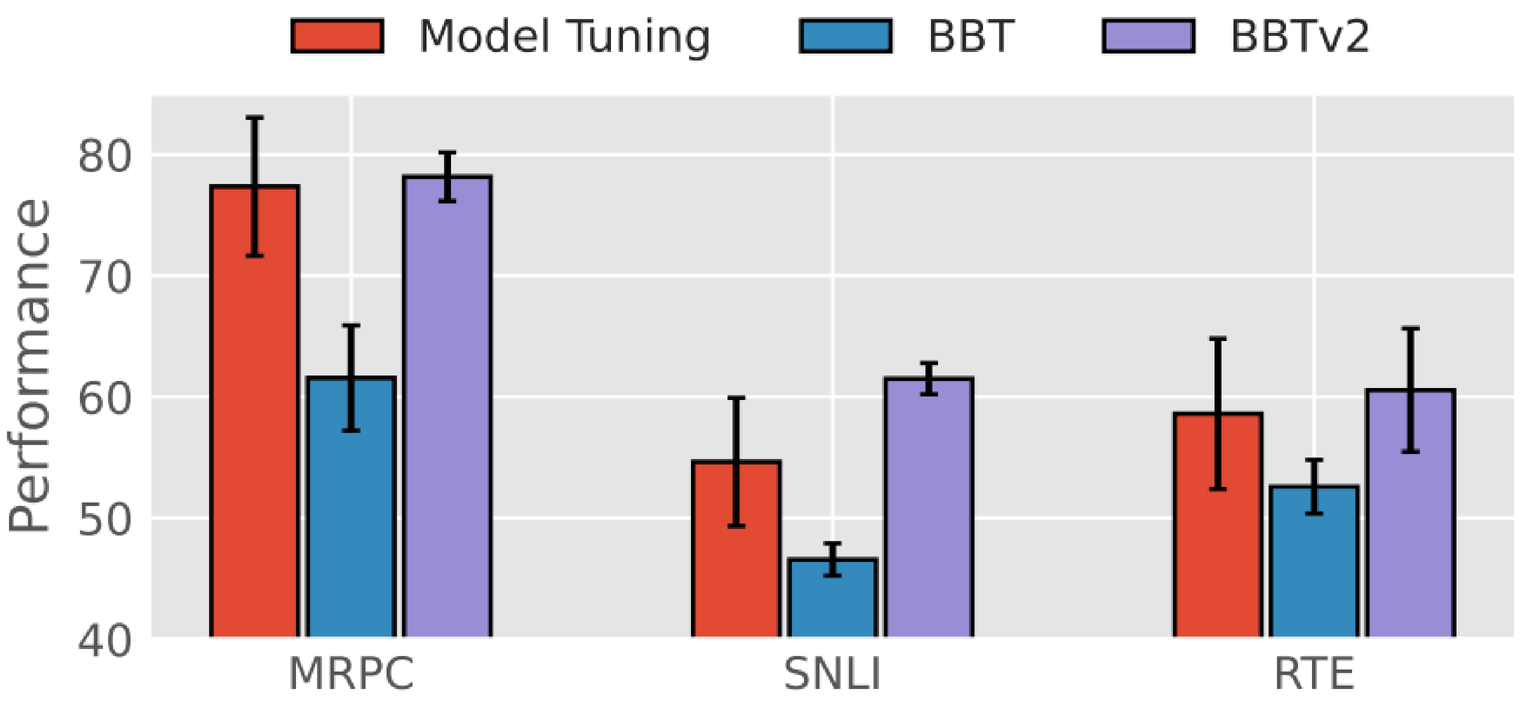
Comparable to full model tuning but merely tuning ~10k parameters



Experiments of BBTv2

Improve BBT on entailment tasks

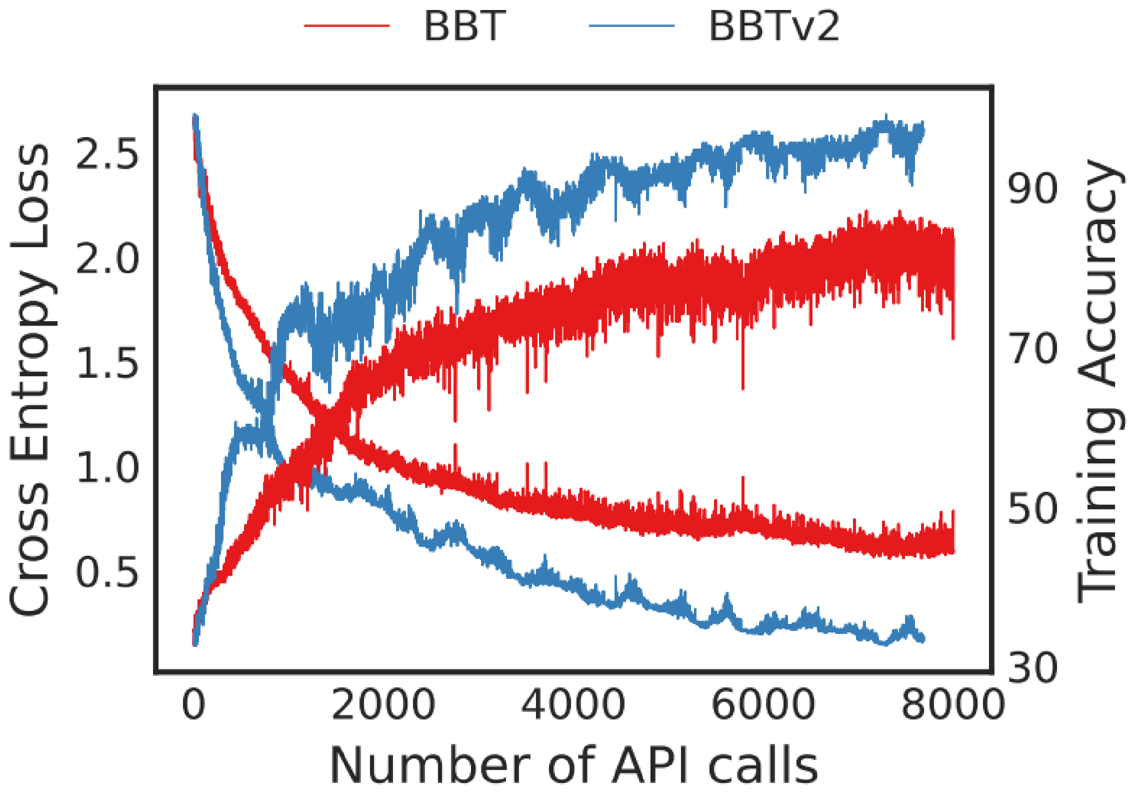
- Be comparable to full model tuning without pre-trained prompt embedding



Experiments of BBTv2

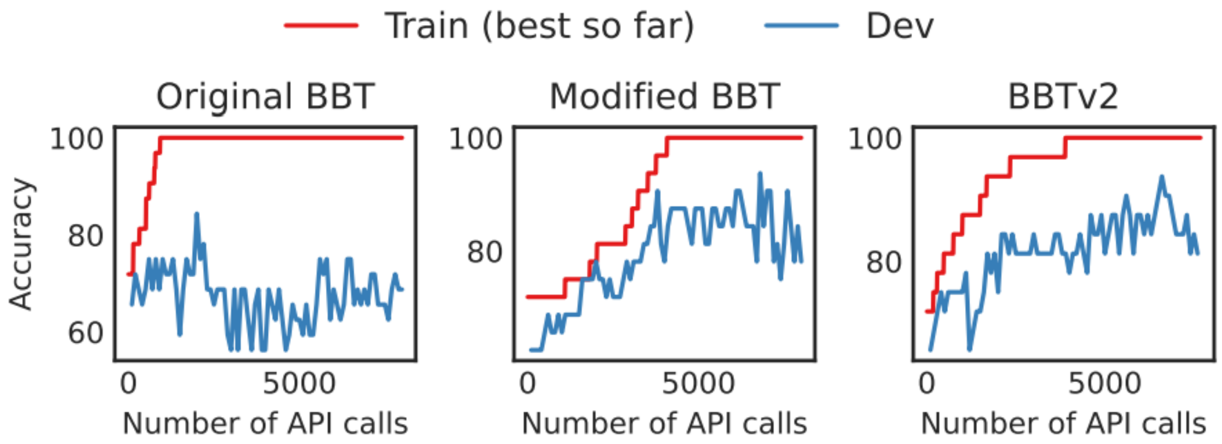
Improve BBT on many-label classification tasks

- Faster convergence than BBT on DBPedia (14 classes)

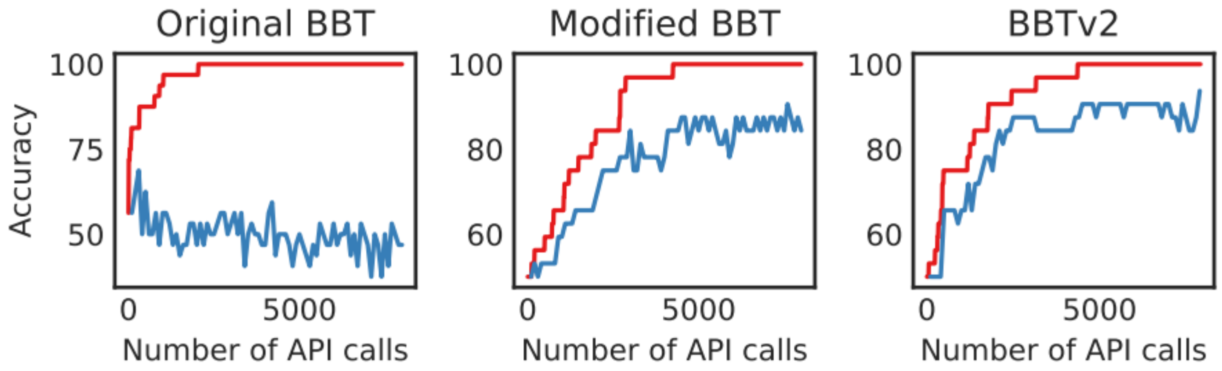


Experiments of BBTv2

Generalization across LMs



(a) $\text{BERT}_{\text{LARGE}}$



Experiments of BBTv2

Overall comparison

Method	**Tunable Params**	**SST-2** acc	**Yelp P.** acc	**AG's News** acc	**DBPedia** acc	**MRPC** F1	**SNLI** acc	**RTE** acc	**Avg.**
Gradient-Based Methods									
Model Tuning	355M	85.39 ±2.84	91.82 ±0.79	86.36 ±1.85	97.98 ±0.14	**77.35** ±5.70	54.64 ±5.29	**58.60** ±6.21	**78.88**
Adapter	2.4M	83.91 ±2.90	90.99 ±2.86	86.01 ±2.18	**97.99** ±0.07	69.20 ±3.58	57.46 ±6.63	48.62 ±4.74	76.31
BitFit	172K	81.19 ±6.08	88.63 ±6.69	86.83 ±0.62	94.42 ±0.94	66.26 ±6.81	53.42 ±10.63	52.59 ±5.31	74.76
LoRA	786K	**88.49** ±2.90	90.21 ±4.00	**87.09** ±0.85	97.86 ±0.17	72.14 ±2.23	**61.03** ±8.55	49.22 ±5.12	78.01
Prompt Tuning	50K	68.23 ±3.78	61.02 ±6.65	84.81 ±0.66	87.75 ±1.48	51.61 ±8.67	36.13 ±1.51	54.69 ±3.79	63.46
P-Tuning v2	1.2M	64.33 ±3.05	**92.63** ±1.39	83.46 ±1.01	97.05 ±0.41	68.14 ±3.89	36.89 ±0.79	50.78 ±2.28	70.47
Gradient-Free Methods									
Manual Prompt	0	79.82	89.65	76.96	41.33	67.40	31.11	51.62	62.56
In-Context Learning	0	79.79 ±3.06	85.38 ±3.92	62.21 ±13.46	34.83 ±7.59	45.81 ±6.67	47.11 ±0.63	60.36 ±1.56	59.36
Feature-MLP	1M	64.80 ±1.78	79.20 ±2.26	70.77 ±0.67	87.78 ±0.61	68.40 ±0.86	42.01 ±0.33	53.43 ±1.57	66.63
Feature-BiLSTM	17M	65.95 ±0.99	74.68 ±0.10	77.28 ±2.83	90.37 ±3.10	71.55 ±7.10	46.02 ±0.38	52.17 ±0.25	68.29
BBT	500	89.56 ±0.25	**91.50** ±0.16	81.51 ±0.79	79.99* ±2.95	61.56 ±4.34	46.58 ±1.33	52.59 ±2.21	71.90
BBTv2	12K	**90.41** ±0.71	90.69 ±0.66	**85.06** ±0.49	**92.59** ±0.17	**78.15** ±2.00	**61.50** ±1.28	**60.56** ±5.09	**79.85**

Experiments of BBTv2

Versatility across different language models

LM	Method	SST-2	AG's News	DBPedia
Encoder-only PTMs				
BERT	BBT	76.26 ±2.64	76.67 ±1.12	89.58 ±0.51
	BBTv2	79.32 ±0.29	79.58 ±1.15	93.74 ±0.50
RoBERTa	BBT	89.56 ±0.25	81.51 ±0.79	79.99 ±2.95
	BBTv2	90.41 ±0.71	85.06 ±0.49	92.59 ±0.17
Decoder-only PTMs				
GPT-2	BBT	75.53 ±1.98	77.63 ±1.89	77.46 ±0.69
	BBTv2	80.13 ±3.28	82.18 ±1.07	91.36 ±0.73
Encoder-Decoder PTMs				
BART	BBT	77.87 ±2.57	77.70 ±2.46	79.64 ±1.55
	BBTv2	89.53 ±2.02	81.30 ±2.58	87.10 ±2.01
T5	BBT	89.15 ±2.01	83.98 ±1.87	92.76 ±0.83
	BBTv2	91.08 ±1.49	84.32 ±1.29	92.76 ±0.85

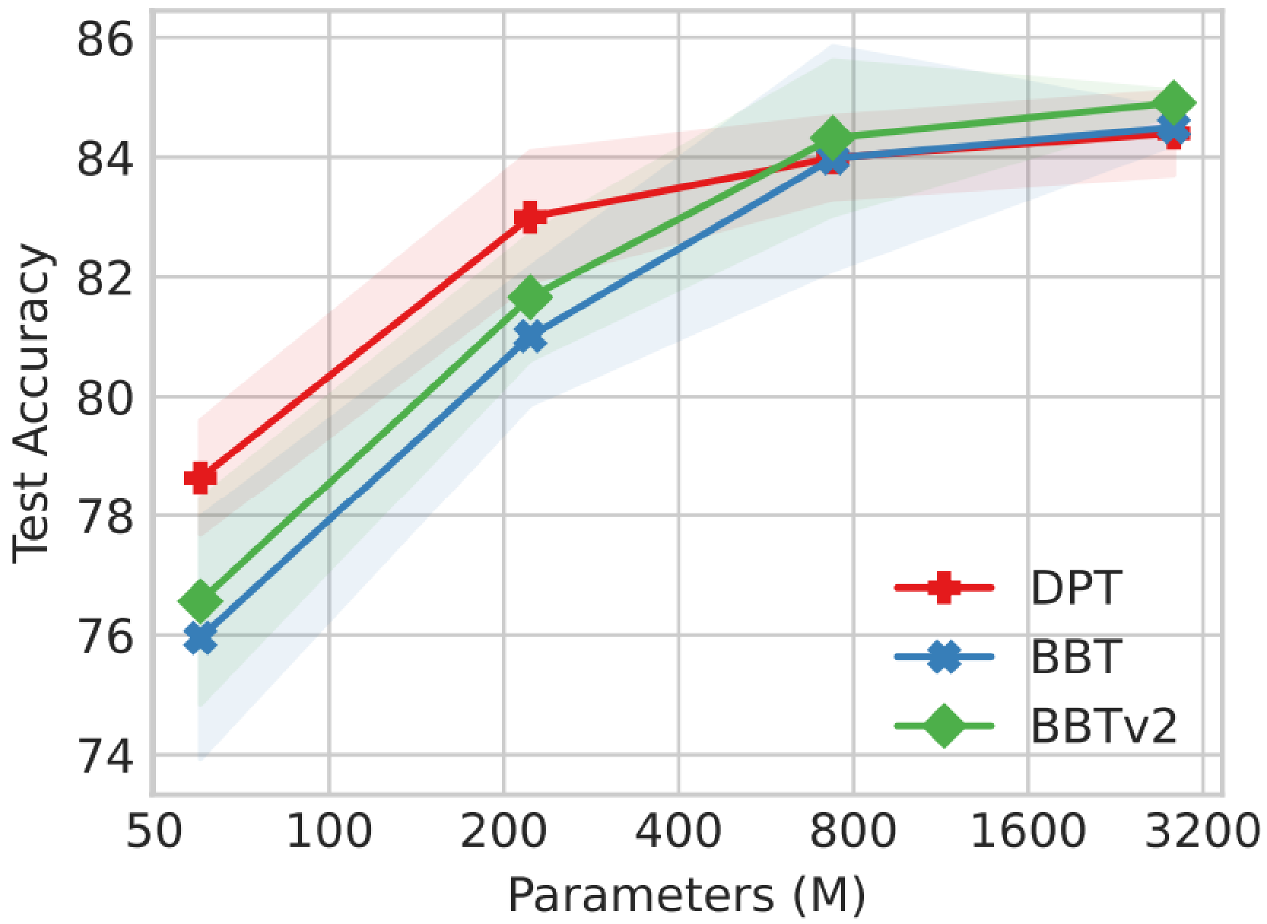
Comparison on CPM-2 (11B)

Method	Tunable Params	ChnSent acc	LCQMC acc
Model Tuning	11B	<u>86.1</u> ±1.8	<u>58.8</u> ±1.8
Vanilla PT	410K	62.1 ±3.1	51.5 ±3.4
Hybrid PT	410K	79.2 ±4.0	54.6 ±2.3
LM Adaption	410K	74.3 ±5.2	51.4 ±2.9
BBTv2	4.8K	**86.4** ±0.8	**59.1** ±2.5

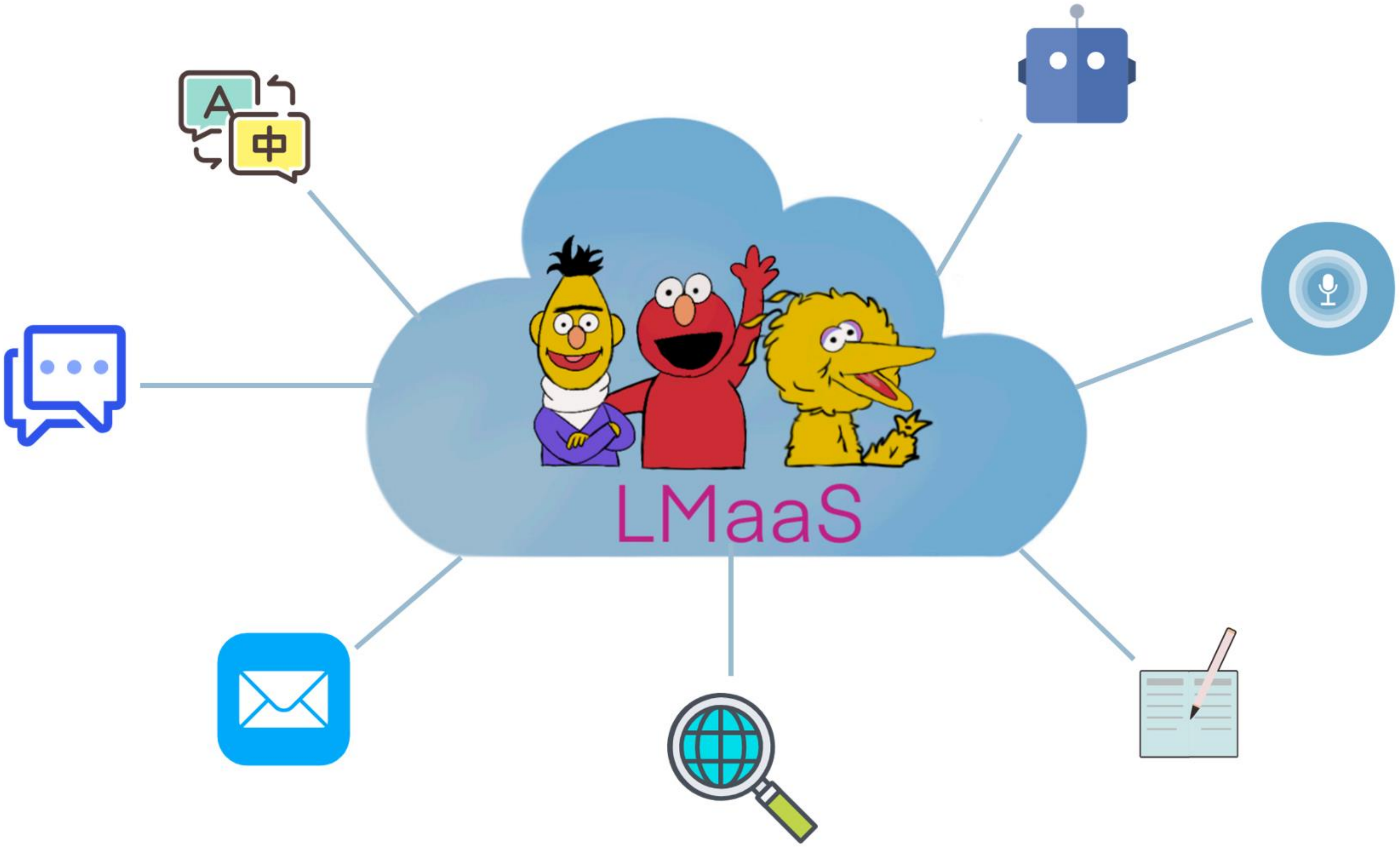
Experiments of BBTv2

The power of scale (with T5)

- Outperform gradient descent when model size becomes large



Other Solutions for LMaaS



Other Solutions for LMaaS

- **Text prompt**: Manually or automatically design task-specific text prompts
- **In-context learning**: Include a few examples in the input at inference time
- **Black-box optimization**: Tuning a small portion of parameters with only the access of the LLM`s output probability via black-box optimization (BBO)
- **Feature-based learning**: LLMs can serve as a feature extractor, on which users can build some lightweight learnable model to solve the task
- **Data generation**: Use LLMs to generate a dataset of labeled text pairs, which is then used to locally train a much smaller model

Check Our Paper List!

README.md

Language Model as a Service (LMaaS)

last commit today | PaperNumber 43

This is a curated list of "Language-Model-as-a-Service (LMaaS)" papers, which is mainly maintained by Tianxiang Sun. We strongly encourage the NLP researchers who are interested in this topic to make pull request to add or update the papers (See Contributing). Watch this repository for the latest updates!

Updates

- 2022/7/7: Write a blog (in Chinese)
- 2022/7/4: Create this paper list

Contents

Readme

779 KB

MIT license

87 stars

3 watching

9 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors 6

Resources

- LMaaS paper list: https://github.com/txsun1997/LMaaS-Papers
- Code of BBT and BBTv2: https://github.com/txsun1997/Black-Box-Tuning
- BBT paper (ICML 2022): https://arxiv.org/abs/2201.03514
- BBTv2 paper: https://arxiv.org/abs/2205.11200
- Blog for BBT (in Chinese): https://zhuanlan.zhihu.com/p/455915295
- Blog for LMaaS (in Chinese): https://zhuanlan.zhihu.com/p/538857729

Feel free to reach out if you have any questions or suggestions about our papers, code, or the paper list!

Thanks!

Tianxiang Sun
School of Computer Science, Fudan University

https://txsun1997.github.io/