Prompt Engineering Learning Sharing

💡 Since the release of ChatGPT (less than a year ago 😨) on November 30, 2022, all walks of life have begun to discuss and apply based on AI and generative big language models. Chat The emergence of GPT marks the arrival of the era of super-large models, and also makes NLP A new paradigm of tasks has received more attention, namely - **pre-trained big language models (LLM) + prompt learning** .

First of all, we will review the development history of NLP tasks based on the papers of Liu Pengfei and others .

Four paradigms for NLP tasks

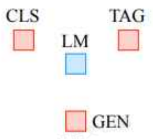
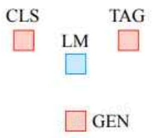
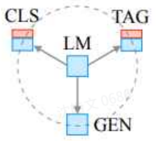
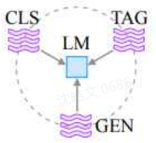
Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Table 1: Four paradigms in NLP. The "**engineering**" column represents the type of engineering to be done to build strong systems. The "**task relation**" column, shows the relationship between language models (LM) and other NLP tasks (CLS: classification, TAG: sequence tagging, GEN: text generation). 🟦: fully unsupervised training. 🟥: fully supervised training. ▨: Supervised training combined with unsupervised training. ≋ indicates a textual prompt. Dashed lines suggest that different tasks can be connected by sharing parameters of pre-trained models. "LM→Task" represents *adapting LMs (objectives) to downstream tasks* while "Task→LM" denotes *adapting downstream tasks (formulations) to LMs*.

Stage 1: Fully supervised learning paradigm

First understand what is **supervised learning (Supervised Learning for NLP)** . Fully supervised learning (Fully Supervised Learning) is a machine learning method in which the training dataset contains inputs and corresponding labels (or outputs). The purpose of this learning method is to train a model so that it can predict the correct output from the input data. This method of labeling the input data is called supervised learning, and the labels generally require manual labeling.

In addition to fully supervised learning, there are several other major Machine Learning methods that vary depending on the type of data used and the training method:

1. Unsupervised learning: In this learning method, the training data does not contain any labels. The goal of the model is to discover structures or patterns in the data. Common unsupervised learning tasks include clustering (e.g. grouping Data Points by similarity) and dimensionality reduction (e.g. simplifying the complexity of the dataset).
2. Semi-supervised learning: This approach uses partially labeled data and large amounts of unlabeled data. This is useful in situations where labeled data is expensive but unlabeled data is readily available. Semi-supervised learning attempts to leverage information from large amounts of unlabeled data to improve learning performance.
3. Self-supervised learning (Self-supervised learning): In self-supervised learning, although the training data is not explicitly labeled, the model can generate labels from the data itself. For example, in natural language processing (NLP), the language model may predict the next word or missing word in a sentence, where the "label" actually comes from other parts of the input data. (Often used in the pre-training phase of language models)

Early fully supervised learning is probably also divided into two "sub-stages", the time and core work of which are as follows:

Fully supervised learning based on non-neural networks

Until about 2011, the core of this phase was **manual design and feature engineering** to maximize the performance of the model. These models include:

1. Support Vector Machine (SVM):
   - SVMs are powerful tools for classification and regression tasks. They work by finding the best separation boundaries between Data Points, which usually involves transforming the data into a high-dimensional space.
2. Decision Tree:
   - Decision trees make predictions by creating rules based on feature selection. They are easy to understand and implement, but usually require well-designed features to perform well.
3. Logistics Regression:

    - Logistic Regression is widely used in binary classification problems, especially in finance and healthcare. It outputs probabilistic predictions based on linear combinations of features.
4. Integration methods (e.g. random forest and Gradient Boosting Machine):
    - These methods combine multiple weak learners, such as Decision Tree, to improve the prediction performance. They are especially sensitive to feature selection and engineering.
5. K-Nearest Neighbor (K-NN):
    - K-NN is an instance-based learning method that predicts the label of a new Data Point based on the label of its nearest neighbor.

Fully supervised learning based on neural networks

Probably between 2011 and 2017, at this stage, we **handed over the feature extraction to the model itself** (its feature engineering capabilities have far exceeded that of humans), and the core work became how to design and optimize the model architecture suitable for the target task, such as MLP, CNN, RNN , LSTM and other basic models have been proposed, which have their own characteristics to adapt to different tasks.

1. Multilayer Perceptron (MLP):
    - MLP is the most basic neural networks architecture with multiple fully connected layers. They excel at handling nonlinear data relationships.
2. Convolution neural networks (CNN):
    - CNNs have achieved great success in image processing and Computer Vision by automatically extracting spatial features through convolution layers.
3. Recurrent neural networks (RNN):
    - RNNs are suitable for processing sequential data such as text or time series. They can maintain the internal state of historical information.
4. Long short-term memory network (LSTM):
    - LSTM is a variant of RNN , which solves the layer disappearance problem of traditional RNN when processing long sequence data.

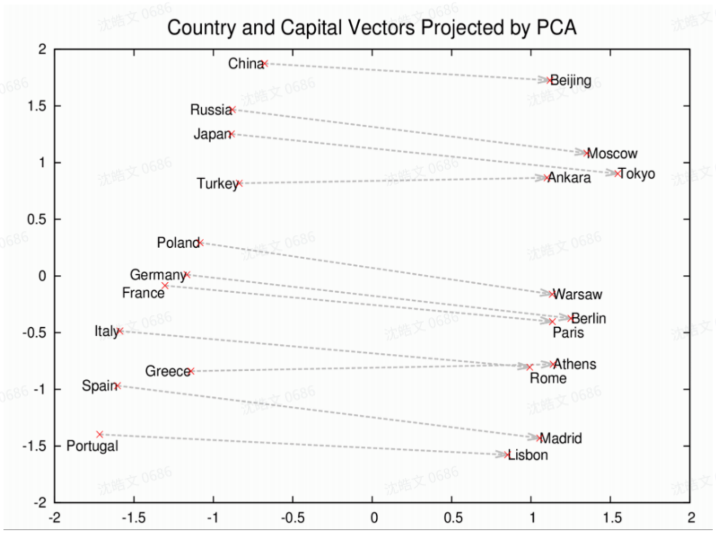
Stage 2: Pre-training paradigm for better generalization

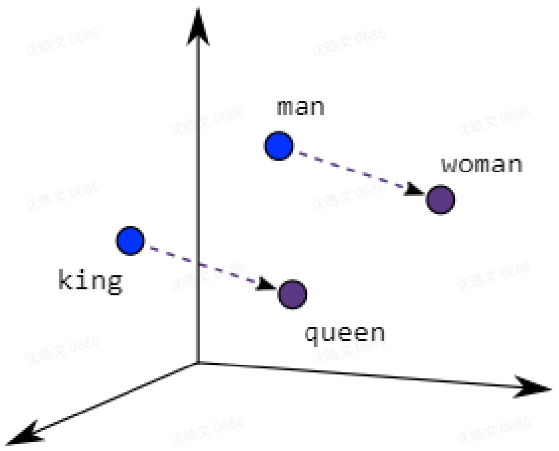
Pre-train and Fine-tuning

Learning paradigm: Specifically, it should be the " **language modeling pre-training-fine-tuning (Pre-training LM and Fine-tune)** " paradigm.

Beginning in 2017, the field of natural language processing (NLP) witnessed a major shift, which was aided by two major developments. On the one hand, advances in **text representation methods** , such as word2vec and GloVe, and on the other, the introduction of the **Transformer architecture** , which greatly improved the ability of text feature extraction, laying the foundation for **pre-training of language models** .

1. Text representation method:
    - Models such as Word2Vec and GloVe have made significant progress in **text representation** . They are able to capture complex relationships between word meaning and context by learning vector representations of words so that words with similar meanings are close to each other in vector space.

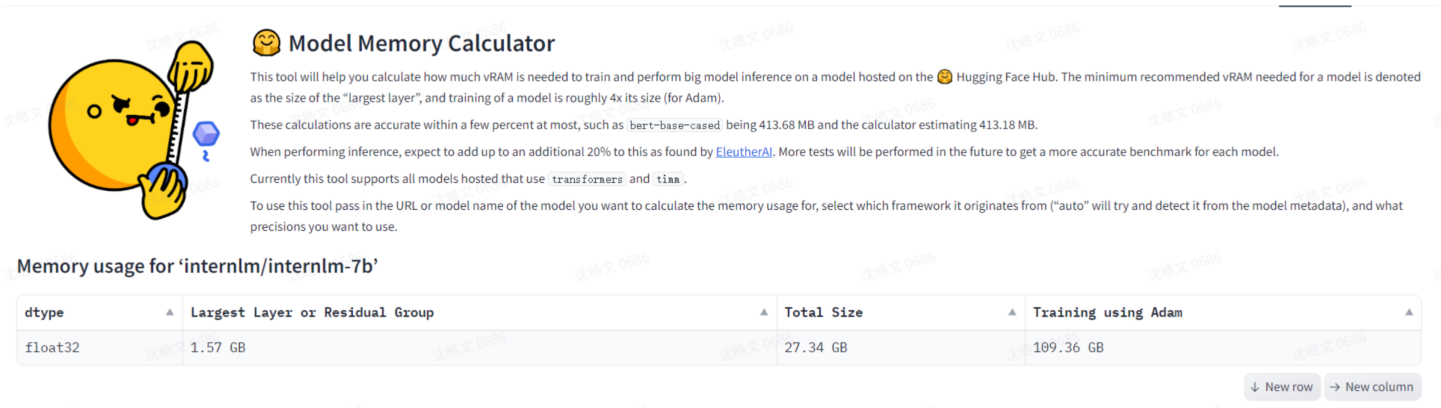




2. Rise of the Transformers:
    - In 2017, the Transformer model (proposed by Google) revolutionized the field of NLP . Compared to previous RNNs and LSTMs, Transformer significantly improved its ability to handle long-distance dependencies, mainly due to its innovative Self-Attention mechanism.
3. GPT-1 and BERT:
    - GPT-1 and BERT are two landmark models based on the Transformer architecture. GPT-1 (Decoder-only) demonstrates the effectiveness of large-scale language models on a variety of tasks. BERT (Encoder-only), through its bidirectional context representation, further enhances language understanding, especially in understanding complex relationships between words in sentences.

The size of the LLM model parameters is very large. Take the familiar GPT-3 in 2020 as an example, the model parameters have reached 175 billion (175B). In this way, when the downstream "fine-tuning" is done, it is a bit expensive to update a huge model for a specific task. In disguise, the generalization cost of the "pre-training-fine-tuning" paradigm is increased.

> For example, deploying our internlm-7b model requires 27.34GB of graphics card memory, while training requires 109.36GB of graphics card memory. The current RTX 4090 is generally 24GB of video memory, so fine-tuning a 7b model requires about 5 4090s.



🤗 Model Memory Calculator

This tool will help you calculate how much vRAM is needed to train and perform big model inference on a model hosted on the 🤗 Hugging Face Hub. The minimum recommended vRAM needed for a model is denoted as the size of the "largest layer", and training of a model is roughly 4x its size (for Adam).

These calculations are accurate within a few percent at most, such as `bert-base-cased` being 413.68 MB and the calculator estimating 413.18 MB.

When performing inference, expect to add up to an additional 20% to this as found by EleutherAI. More tests will be performed in the future to get a more accurate benchmark for each model.

Currently this tool supports all models hosted that use `transformers` and `timm`.

To use this tool pass in the URL or model name of the model you want to calculate the memory usage for, select which framework it originates from ("auto" will try and detect it from the model metadata), and what precisions you want to use.

Memory usage for 'internlm/internlm-7b'

dtype	Largest Layer or Residual Group	Total Size	Training using Adam
float32	1.57 GB	27.34 GB	109.36 GB

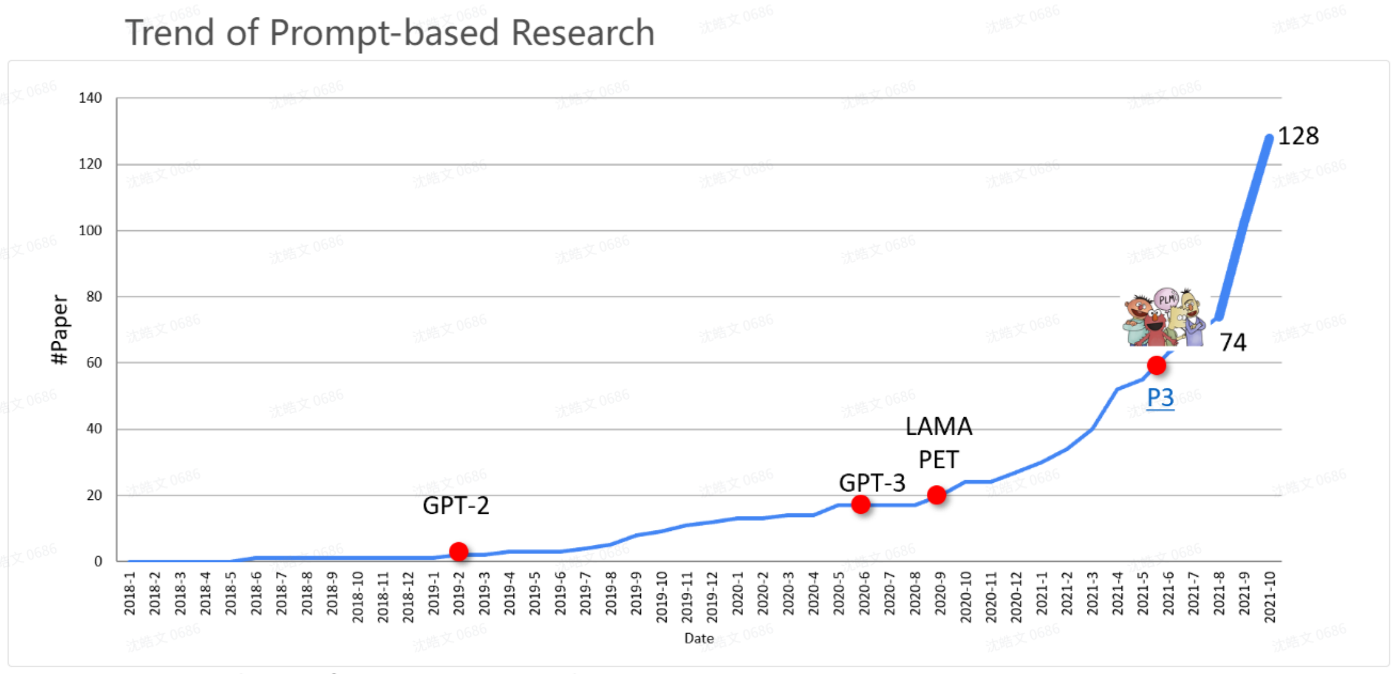
Model Memory Calculator

Pre-training and Prompt

1. Pre-training:
   - In the pre-training phase, a model (e.g., a language model) is trained on a large amount of data in order to learn the basic structures and patterns of the language. This training is not for a specific task, but to enable the model to understand and process various language inputs.
   - The goal of this stage is to enable the model to acquire a broad range of language knowledge, allowing it to be more effectively applied in subsequent tasks.
2. Prompt:
   - The hint phase refers to the use of specific inputs (i.e. hints) to guide the model to a specific task. These hints are usually questions or statements of a specific format, designed to motivate the model to give meaningful output.
   - **Unlike traditional fine-tuning, cues do not involve changing the internal parameters of the model, but rather leverage the model's existing knowledge and capabilities through cleverly designed inputs.**

This shifts from the paradigm of **fine-tuning** the pretrained model to shortening the distance to the downstream task to the lightweight way of **prompting** the downstream task to shortening the distance to the pretrained model.

This "pre-training-hint" paradigm was represented by the GPT-2 model that debuted in 2019, and was pushed to the peak after the launch of GPT-3, especially in 2021, and even spawned "Prompt Engineering". Research papers around this have also sprung up (**the following figure is quoted from pretrain.nlpedit.ai**):



Domain-specific better solutions: the "pre-training-fine-tuning-prompting" learning paradigm

For practical applications in specific fields in reality, such as law, customer service, medical care, etc., the "pre-training-prompt" paradigm is also fine-tuned with some training data for specific fields, and the application effect will be further improved. Therefore, this paradigm also appears:

Although there will be the above-mentioned "high cost of fine-tuning LLM" problem, but for the application area of monetization, the significant improvement of AI performance brought by LLM is worth the cost of fine-tuning in exchange for commercial benefits, and usually these areas also have private data authorization issues, such as in March 2023 Meta released LLaMA and provided pre-trained model download, many people in the industry began to use their own commercial sensitive data for fine-tuning and then provide services.

Stage 3: "Pre-training - artificial feedback reinforcement learning - prompt (Pre-training, RLHF and Prompt) " learning paradigm

The first is that pre-trained language models require too much data, models, and computing power, just like when computers first came out and occupied a building. The second is that LLM is not aligned with human values, morals, and ethics, and there are risks. Therefore, it is necessary to promote the emergence of helpful, harmless, and honest models. This is the issue of "alignment".

- **Pre-training - artificial feedback reinforcement learning - prompt (Pre-training, RLHF and Prompt)** Learning Paradigm: The RLHF method was first proposed by OpenAI in the 2017 paper **"Deep reinforcement learning from human preferences"** It was proposed in the paper that after GPT-2 and GPT-3 were released one after another, there were a lot of negative cases of using GPT series such as fake news, abetting crime, negative innuendo, etc., so OpenAI began to pay attention to Alignment and finally in **The RLHF method was introduced on the InstructGPT in the first half of 2022** Alignment of human morality and ethics, which played a good effect, and later this was iterated back to GPT-3 to ensure that API calls follow human morality and ethics. This paradigm is also used in the familiar ChatGPT and has become the current mainstream paradigm.

Similar to the second stage, for specific domain applications, downstream find-tune can also be used to further improve the effect, namely the following paradigm:

- **Pre-training - manual feedback reinforcement learning - fine-tuning - prompting (Pre-training, RLHF, Fine-tune and Prompt)** Learning Paradigm: Pre-training - manual feedback reinforcement learning is the upstream stage, and fine-tuning and prompting belong to the downstream stage. The GPT that currently provides fine-tune API The GPT model behind it also has RLHF, which is no longer the GPT version that did not consider alignment at first. You can see that there is Moderation API in the official GPT API documentation

Prompt Engineering Guide

💡 Prompt Engineering is a relatively new discipline that focuses on prompt word development and optimization to help users apply the Large Language Model (LLM) to various scenarios and research areas.

Researchers can use hint engineering to enhance the ability of large language models to handle complex task scenarios, such as question answering and arithmetic reasoning. Developers can efficiently integrate with large language models or other ecological tools through hint engineering design and the development of powerful engineering techniques.

Prompt engineering is not just about designing and developing cue words. It includes a variety of skills and techniques for interacting with and developing large language models. Prompt engineering plays an important role in enabling interaction, docking, and understanding of large language models. Users can improve the security of large language models through cue engineering, and can also empower large language models, such as with domain expertise and external tools to enhance large language model capabilities.

The "consensus" we should have:

- There is a lot of knowledge in the pre-trained model.

> GPT3 175B parameters, 45TB of training data (speculative)

- The pre-trained model itself has the ability to learn from fewer examples.

> In-Context Learning proposed by GPT-3 also effectively proves that in Zero-shot and Few-shot scenarios, the model can achieve good results without any parameters.

In-context Learning

OpenAI explicitly proposed In-Context Learning in the release of GPT-3, which means that when using unsupervised GPT-3 with a small number of examples, good output feedback can be obtained. This is called Few-Shot Prompt. When there is only one example, it is called One-Shot Prompt, and when there is no example, it is called Zero-Shot. For these examples that appear in the input during use, the model will not update its parameters to do fine-tuning. So how does the model learn from these examples? We call this learning method In-Context Learning, which means that the model completes non-explicit learning from unsupervised training text context.

Zero-Shot example：

```
提示：
"""
将文本分类为中性、负面或正面。
文本：我认为这次假期还可以。
情感：
"""

模型输出：
"""
中性
"""
```

Please note that in the above prompt, we did not provide any examples to the model - this is where zero-sample capability comes in.

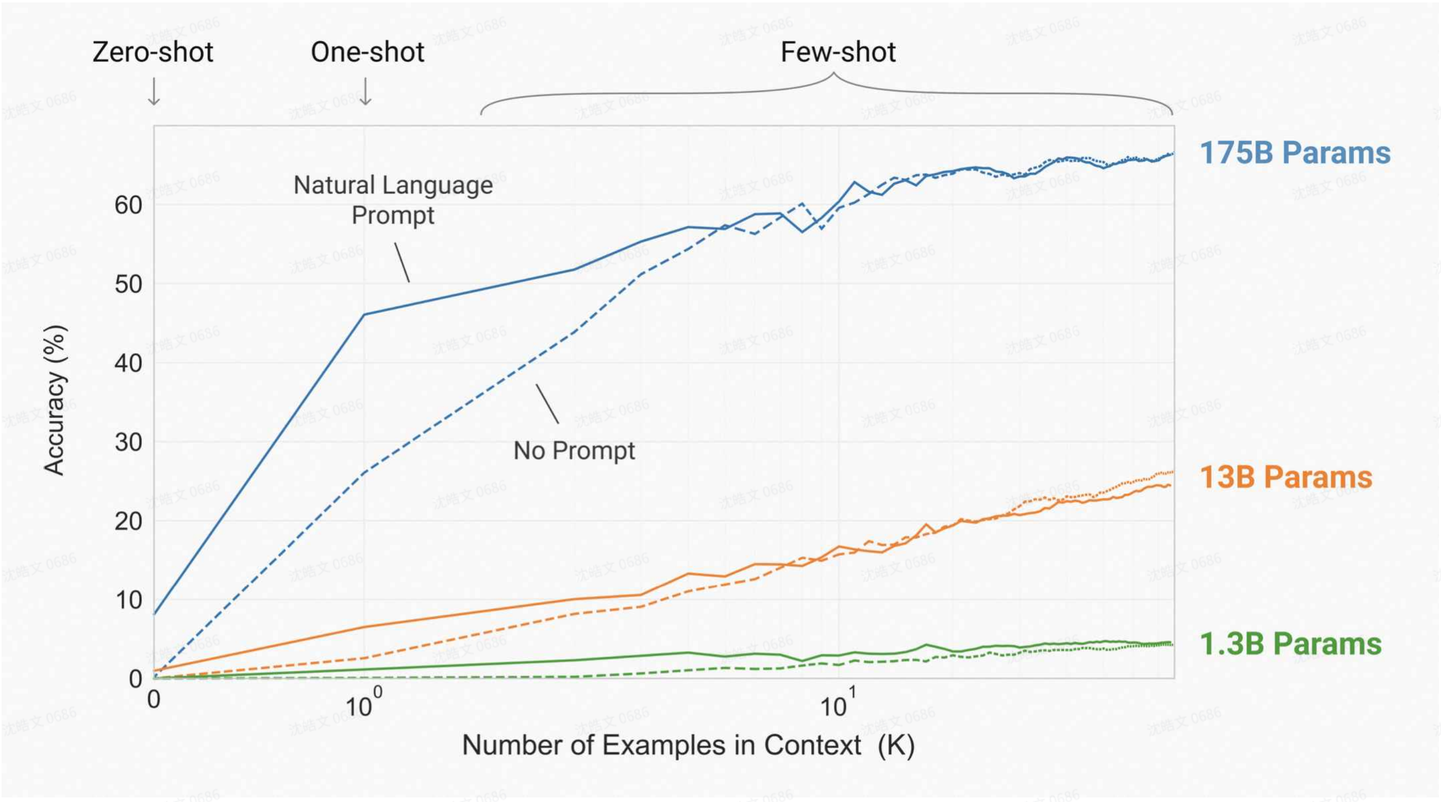
So far, the source of the In-Context Learning ability of the model has not been fully explained, and there are mainly two theories:

- The model undergoes dynamic layer descent through prompt (in a sense has been falsified)

- The model guides to a similar training language library (Bayesian inference) during pre-training through prompts

One-Shot example:

```
提示：
"""
“whatpu”是坦桑尼亚的一种小型毛茸茸的动物。一个使用whatpu这个词的句子的例子是：
我们在非洲旅行时看到了这些非常可爱的whatpus。
“farduddle”是指快速跳上跳下。一个使用farduddle这个词的句子的例子是：
"""

模型输出：
"""
当我们赢得比赛时，我们都开始farduddle。
"""
```



The Nature of Prompt Learning

- Main study time before 2022
- The main purpose is to **improve model performance**

Unify all downstream tasks into pre-training tasks; "Convert the data of downstream tasks into natural language form with a specific template" to fully tap the ability of the pre-

trained model itself. Essentially, it is to design a template that is more suitable for the upstream pre-training task. Through the design of the template, it is to "tap the potential of the upstream pre-training model", so that the upstream pre-training model can better complete the downstream task without the need for labeled data. The key includes three steps:

1. The task of designing a pre-trained language model
2. Design input template style (Prompt Engineering)
3. Design label style and how the output of the model is mapped to the label (Answer Engineering)

Form

Taking the movie review **sentiment classification task** as an example, Prompt Learning actually **transforms the classification task into a generative task that the pre-trained model is more familiar with** : Raw input: The special effects are very cool, I like it.

Prompt input:

Prompt Template 1 ": The special effects are very cool and I really like them. This is a [MASK] movie.

Promp Template 2 ": The special effects are very cool and I really like them. This movie is very [MASK]

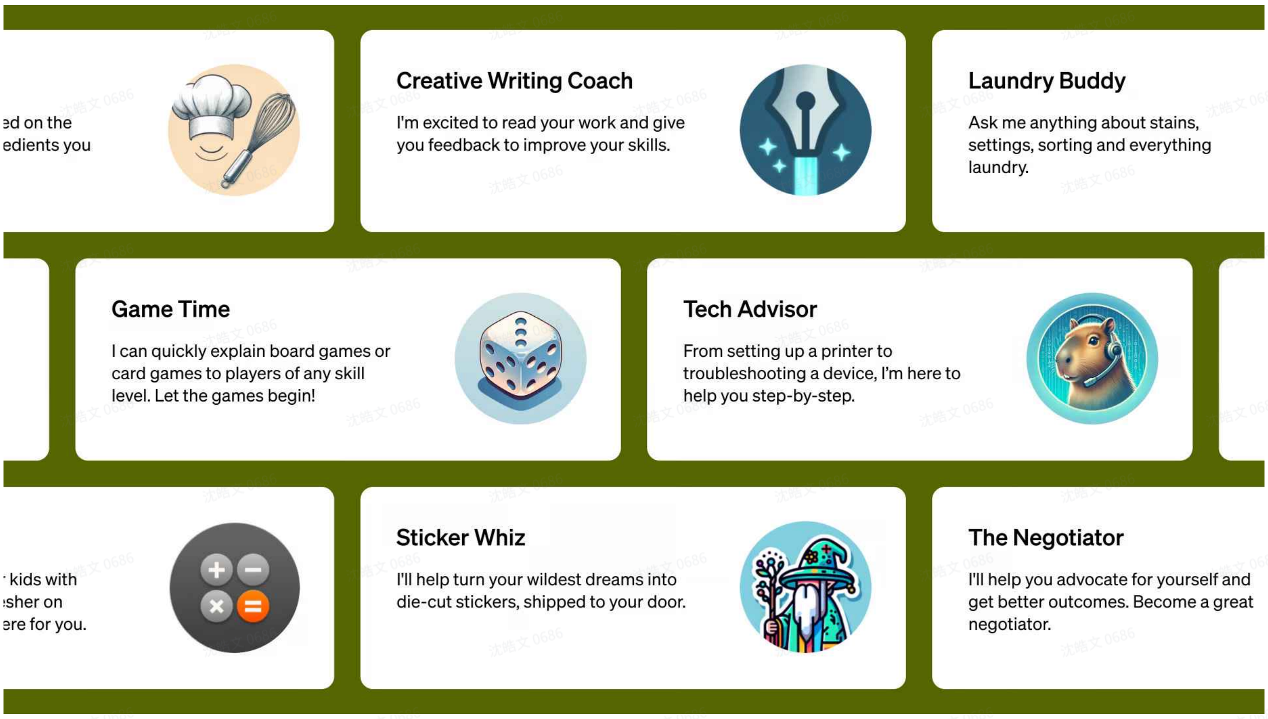
The function of the prompt template is to convert the training data into natural language form and MASK it in the appropriate position to stimulate the ability of the pre-trained model.

The Essence of Prompt Engineering

- Chatgpt has emerged in large numbers after its appearance
- Applied to a variety of different functions, it is actually due to the enhancement of LLM's own capabilities
- Feature Engineering --- LLM >> Prompt Engineering

User's perspective:

1. By designing prompt words, the model better aligns with human **intentions** (role-playing, etc.)
2. By designing prompt words, the model outputs more accurate and valuable content (reasoning ability, COT, etc.).
3. Develop your own application (gpts, etc.) through prompt words.


ed on the
edients you

Creative Writing Coach
I'm excited to read your work and give you feedback to improve your skills.

Laundry Buddy
Ask me anything about stains, settings, sorting and everything laundry.

Game Time
I can quickly explain board games or card games to players of any skill level. Let the games begin!

Tech Advisor
From setting up a printer to troubleshooting a device, I'm here to help you step-by-step.

kids with
sher on
ere for you.

Sticker Whiz
I'll help turn your wildest dreams into die-cut stickers, shipped to your door.

The Negotiator
I'll help you advocate for yourself and get better outcomes. Become a great negotiator.

Developer's perspective:

1. By designing prompt words, the model generates structured data (building training data benches, etc.).
2. Develop downstream applications based on LLM by designing prompt words (many startups).
3. By designing prompt words, the model plays various roles (GPT scoring, planning, etc.).
4. By designing prompt words, give the model the ability to use external tool APIs and knowledge (langchain, toolbench, agent, etc.)
5. By designing prompt words, explore the security of the model itself, illusions, and other issues (Google RedTeam, etc.).

Prompt template category

Hard Prompts

Hard prompts can be seen as fixed problem templates written manually.

- Definition: Hard prompts refer to explicit and structured instructions or questions directly input into the model. These prompts are usually complete sentences or questions that clearly tell the model what kind of information the user wants or what kind of task to perform.
- Example: For example, "Please list the main food sources of vitamin C" or "Explain Newton's second law".

```
"""
### 指令 ###
将以下文本翻译成西班牙语：

文本：“hello! ”
"""
```

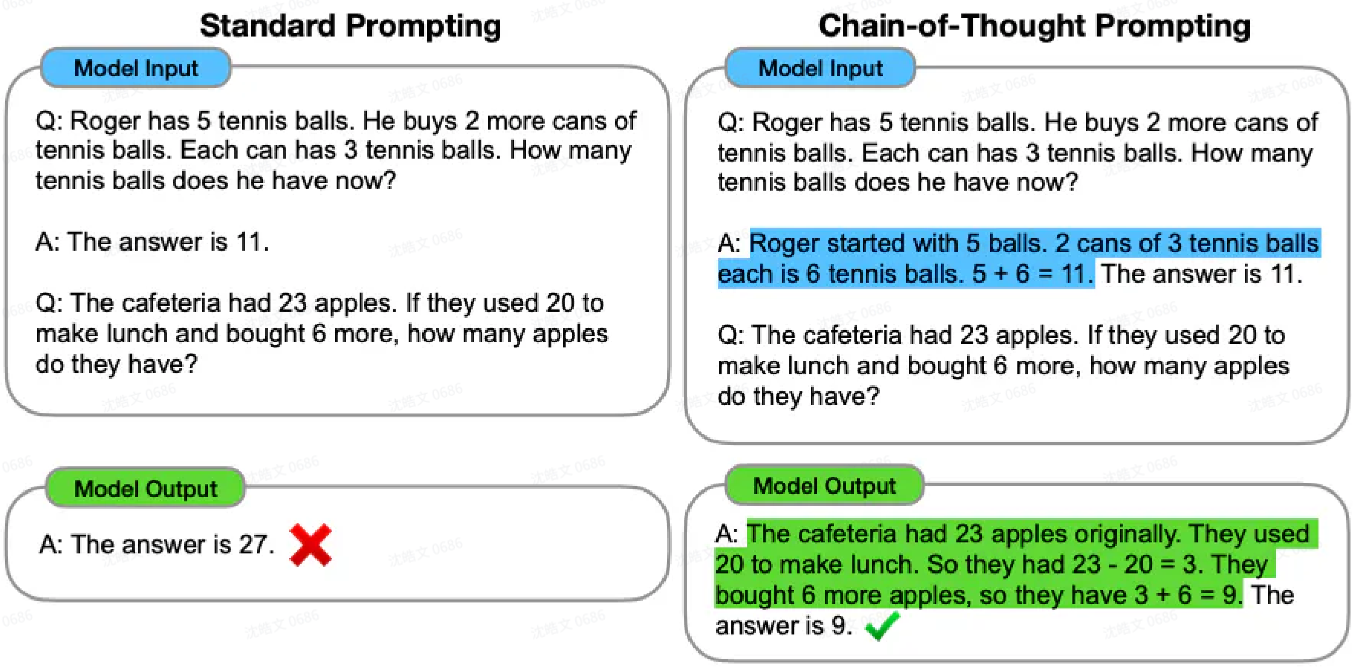
- Features: This prompt method relies on users being able to accurately and clearly express their needs, and the model's response is based on these specific instructions.

> Hard prompts are manually handcrafted text prompts with discrete input tokens. ---- HuggingFace

Of course, there are also many more interesting hard prompt templates

Chain of Thought (CoT)

In 2022, the reasoning ability of the model will be greatly enhanced



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

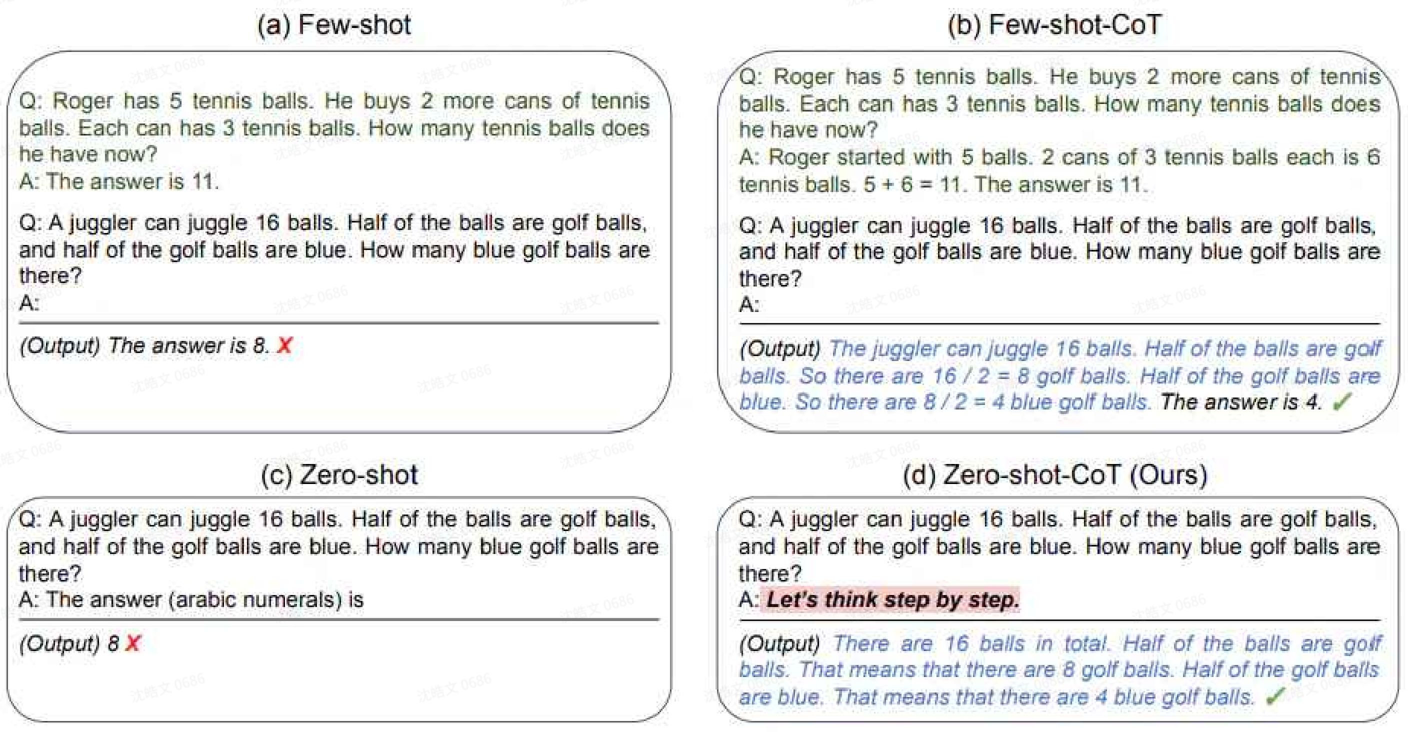
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Zero shot COT

2022， **“Let's think step by step.”**


(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

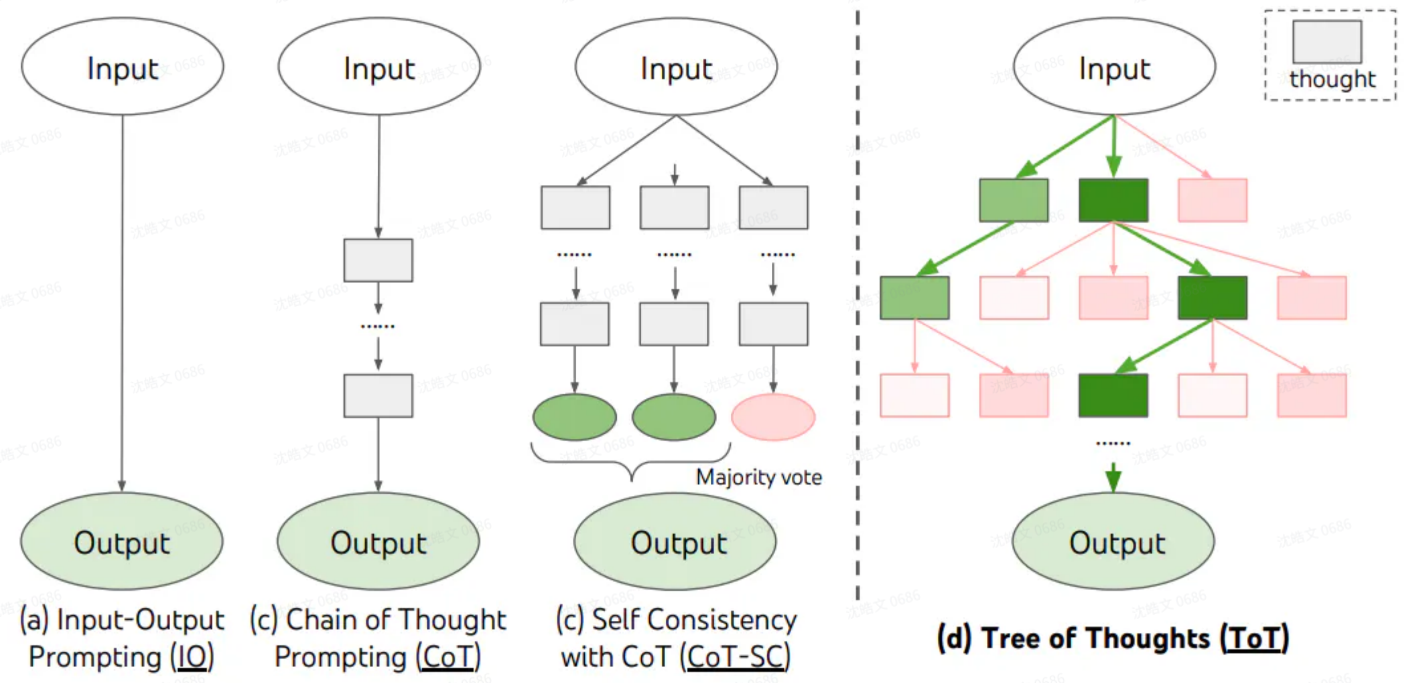
(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

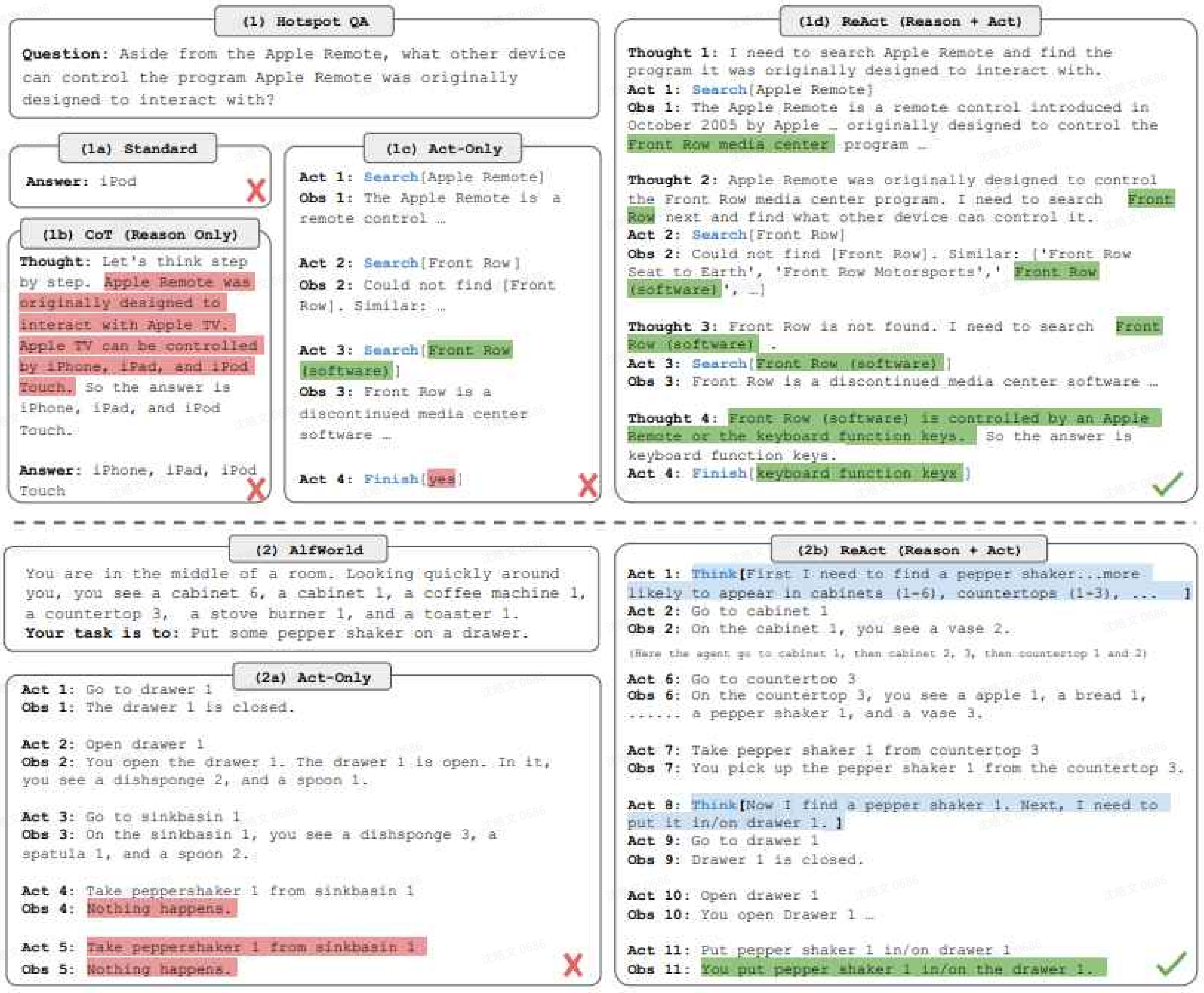
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Tree of Thoughts (ToT)

ToT maintains a thinking tree, where thinking is represented by a coherent linguistic sequence, which is the intermediate step in problem-solving. Using this method, LM can evaluate intermediate thinking in rigorous reasoning processes on its own. **LM combines the ability to generate and evaluate thinking with search algorithms** (such as breadth-first search and depth-first search) **combined** , it can be verified and traced back when systematically exploring thinking.


Input
Output
(a) Input-Output Prompting (IO)
Input
Output
(c) Chain of Thought Prompting (CoT)
Input
Majority vote
Output
(c) Self Consistency with CoT (CoT-SC)
Input
thought
Output
(d) Tree of Thoughts (ToT)

ReAct framework


(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control …

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: …

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Act 4: Finish[yes] ✗

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software)', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓

(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. ✓

Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

ReAct 学习调研

“ToolAgent Template”

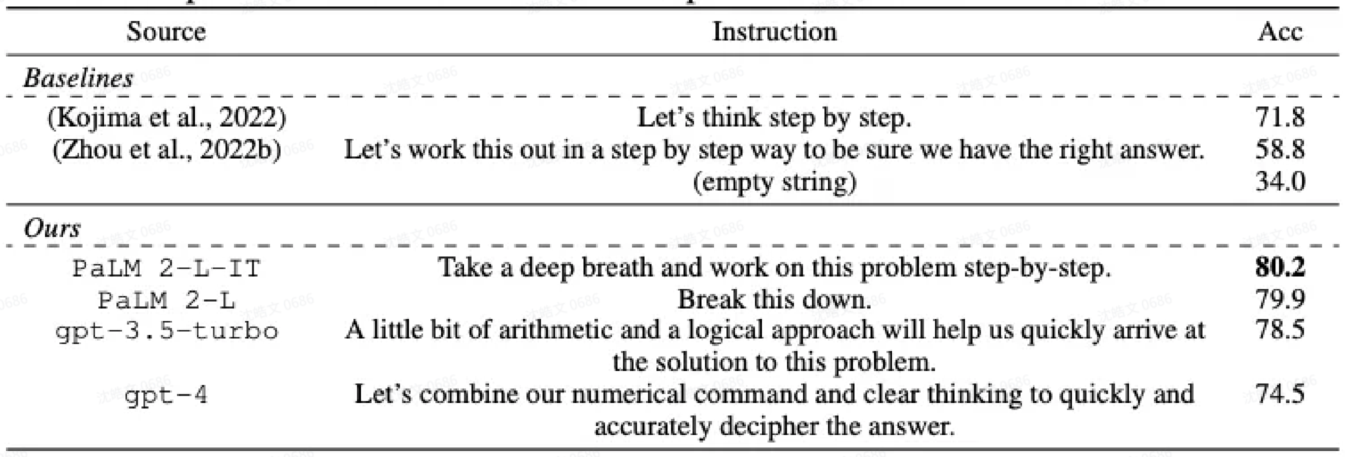
```
# Set up the base template
prompt template = """
Answer the following questions as best you can, but speaking as a pirate might speak. You have access to the following tools:

{tools}

Use the following format:

Question: the input question you must answer
```

```
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin! Remember to speak as a pirate when giving your final answer. Use lots of "Arg"s

Question: {input}
{agent_scratchpad}
"""
```

Google DeepMind "LLM as Optimizer" :

2023.9

Table 1: Top instructions with the highest GSM8K zero-shot test accuracies from prompt optimization with different optimizer LLMs. All results use the pre-trained `PaLM 2-L` as the scorer.



Source	Instruction	Acc
Baselines		
(Kojima et al., 2022)	Let's think step by step.	71.8
(Zhou et al., 2022b)	Let's work this out in a step by step way to be sure we have the right answer.	58.8
	(empty string)	34.0
Ours		
`PaLM 2-L-IT`	Take a deep breath and work on this problem step-by-step.	**80.2**
`PaLM 2-L`	Break this down.	79.9
`gpt-3.5-turbo`	A little bit of arithmetic and a logical approach will help us quickly arrive at the solution to this problem.	78.5
`gpt-4`	Let's combine our numerical command and clear thinking to quickly and accurately decipher the answer.	74.5

Soft Prompts

The original intention of Prompt is to "make downstream tasks cater to upstream models". In fact, it is logically similar to "making upstream models cater to downstream tasks", except that the latter changes the upstream model in a "fine-tuning" way. Inspired by this, **Can we change the downstream in a similar way to "fine-tuning"?** That is, in order to find a better Prompt, we can take optimizing the Prompt itself as a task goal, and let the machine continuously optimize the Prompt - this is **Soft Prompt** , also known as **Continuous Prompt** .

> **Soft-prompts** are represented as a **parameter *Pe*. The prompt is then concatenated to the embedded input forming a single matrix** [*Pe*, *Xe*] which then flows though the encoder-decoder as normal.

> **The models are satisfied to maximize the probability of Y, but only the prompt parameters Pe are updated.**

Prefix-Tuning：Optimizing Continuous Prompts for Generation

2021.1

Inspired by the idea of Prompt, it is proposed in Prefix-Tuning to add a continuous task-specific embedding vector (`continuous task-specific vectors`) to each input for training. When retraining downstream tasks, all parameters of the original large model are fixed, and only prefix embeddings related to downstream tasks are retrained.

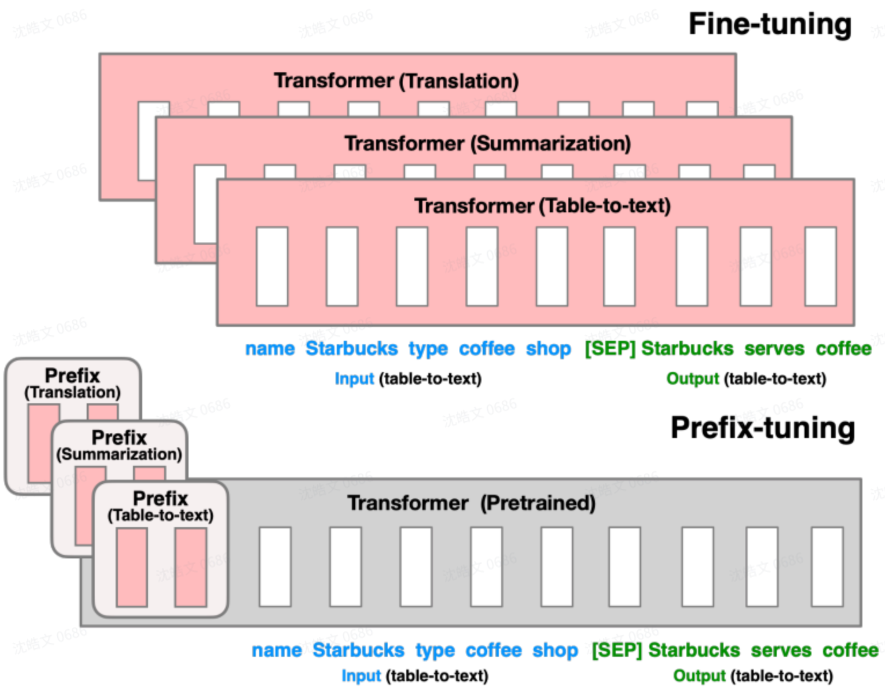

Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix-tuning
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)

Figure 1: Fine-tuning (top) updates all Transformer parameters (the red Transformer box) and requires storing a full model copy for each task. We propose prefix-tuning (bottom), which freezes the Transformer parameters and only optimizes the prefix (the red prefix blocks). Consequently, we only need to store the prefix for each task, making prefix-tuning modular and space-efficient. Note that each vertical block denote transformer activations at one time step.

P-Tuning v1：GPT Understands, Too

2021.3

In P-Tuning, prompt generation is achieved through the `Prompt Encoder`. The difference from before is that pseudo-prompt and backpropagation are used to update the encoder. There is a difference in the input of embedding. The prompt token embedding vector in the template is generated from the `Prompt Encoder` and does not correspond to specific words in the thesaurus.

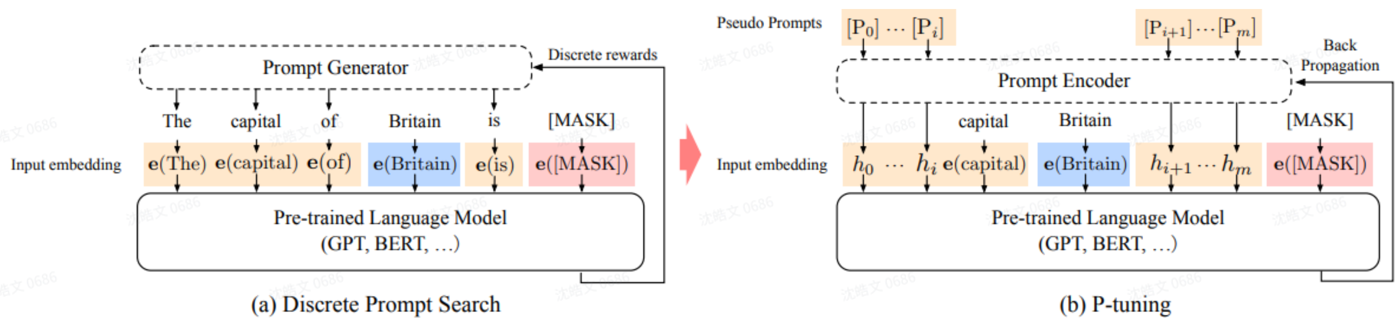


Figure 2: An example of prompt search for “The capital of Britain is [MASK]”. Given the context (blue zone, “Britain”) and target (red zone, “[MASK]”), the orange zone refer to the prompt. In (a), the prompt generator only receives discrete rewards; on the contrary, in (b) the continuous prompt embeddings and prompt encoder can be optimized in a differentiable way.

Prompt Tuning

2021.4

Unlike traditional model tuning (fine-tuning), which requires fine-tuning all parameters of the model and generating corresponding models for different tasks, the author directly adds specific soft prompts for each downstream task, randomly initializing and splicing at the beginning of the sequence. Then the fixed model parameters remain unchanged, only fine-tuning the soft prompt parameters is sufficient.

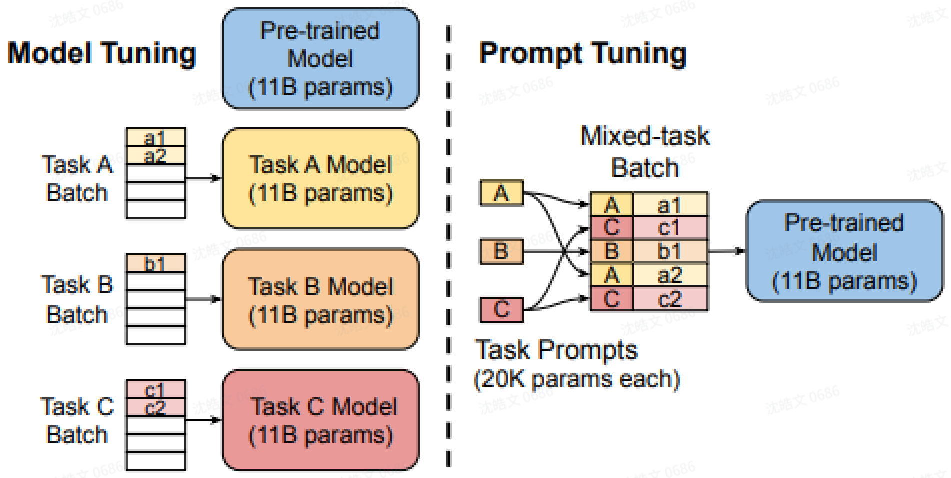

Model Tuning
Pre-trained Model (11B params)
Task A Batch
a1
a2
Task A Model (11B params)
Task B Batch
b1
Task B Model (11B params)
Task C Batch
c1
c2
Task C Model (11B params)
Prompt Tuning
Mixed-task Batch
A
B
C
A a1
C c1
B b1
A a2
C c2
Pre-trained Model (11B params)
Task Prompts (20K params each)

Figure 2: **Model tuning** requires making a task-specific copy of the entire pre-trained model for each downstream task and inference must be performed in separate batches. **Prompt tuning** only requires storing a small task-specific prompt for each task, and enables mixed-task inference using the original pre-trained model. With a T5 "XXL" model, each copy of the tuned model requires 11 billion parameters. By contrast, our tuned prompts would only require 20,480 parameters per task—a reduction of *over five orders of magnitude*—assuming a prompt length of 5 tokens.

Explore and discover:

- Fine-tuning: As the model size increases, the improvement gradually saturates
- Prompt tuning: As the model size increases, performance improves, basically matching Fine-tuning (while only requiring fine-tuning parameters of 10 ^ -5 orders of magnitude).

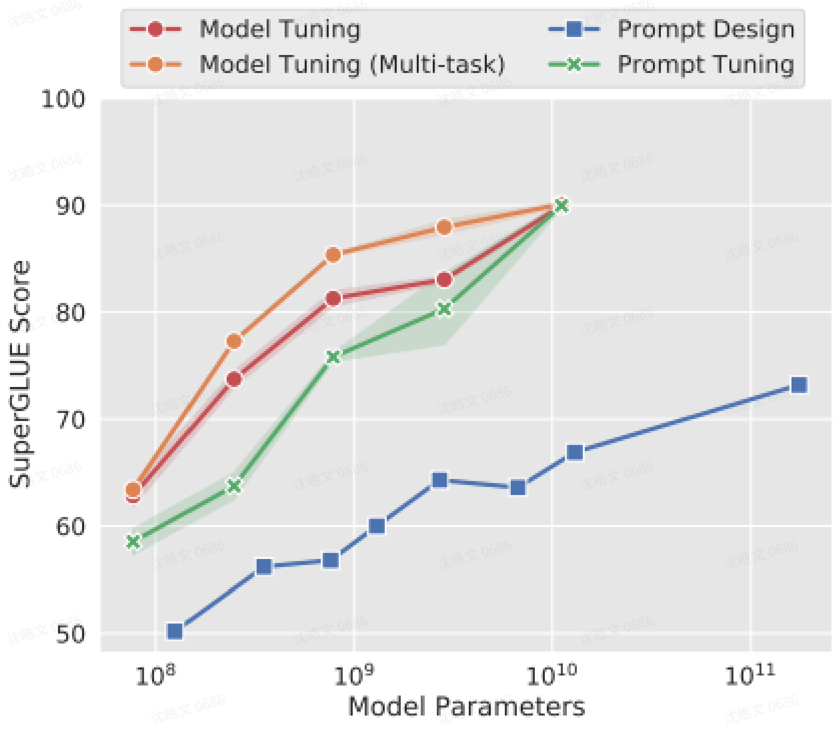


Figure 1: Standard **model tuning** of T5 achieves strong performance, but requires storing separate copies of the model for each end task. Our **prompt tuning** of T5 matches the quality of model tuning as size increases, while enabling the reuse of a single frozen model for all tasks. Our approach significantly outperforms few-shot **prompt design** using GPT-3. We show mean and standard deviation across 3 runs for tuning methods.

Outlook (guess)

- Further LLM alignment and secure governance, aligned with human values and needs, strengthening the LLM's ability to understand human **intent** . -
- Lowering the threshold for LLM applications allows ordinary people without programming skills to develop their own applications (gpts) through Prompt
- Further expansion of LLM's external capabilities (tool API calls, networking, MultiModal Machine Learning)
- LLM Agent develops LLM-based intelligent devices, operating systems, etc. from the hardware level
- LLM based management and the company's organizational structure

Reference

https://www.mikecaptain.com/2023/03/06/captain-aigc-2-llm/

https://www.promptingguide.ai/zh

https://arxiv.org/pdf/2107.13586.pdf

https://learnprompting.org/

https://cobusgreyling.medium.com/prompt-tuning-hard-prompts-soft-prompts-49740de6c64c

https://posts.careerengine.us/p/63dda690a3e486064583f3a3?from=latestPostSidePanel

https://zhuanlan.zhihu.com/p/464876160

https://mltalks.medium.com/%E8%AF%A6%E8%A7%A3%E5%A4%A7%E6%A8%A1%E5%9E%8B%E5%BE%AE%E8%B0%83%E6%96%B9%E6%B3%95prompt-tuning-%E5%86%85%E9%99%84%E5%AE%9E%E7%8E%B0%E4%BB%A3%E7%A0%81-7e4276927729