Google

Prompt Engineering

Aligning models to user's intentions

vsubhashini@

Mar 2023

Overview

00 Background

01 Zero-shot, Few-shot

02 Consistency decoding

03 Learned prompt tokens

04 Instruction Tuning

05 Chain-of-Thought (+ with-action)

06 Retrieval augmented

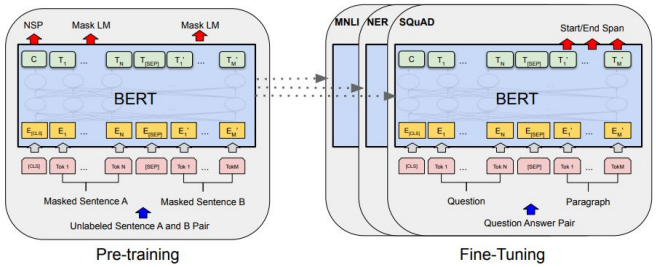
07 Tool augmented

08 RLHF

Background: Model architectures

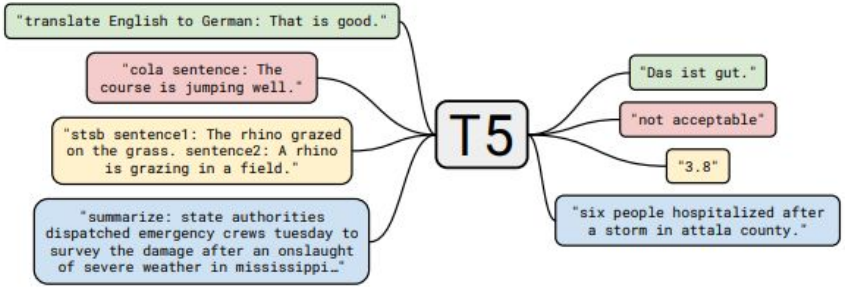
Encoder-only

- BERT
- Pre-train then fine-tune



Encoder-Decoder

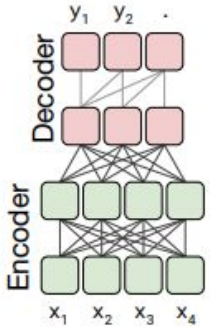
- T5
- Pre-train, fine-tune



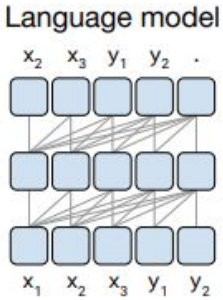
Decoder-only

- LaMDA, PaLM, GPT, OPT, ChatGPT

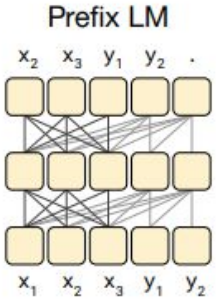
Background: Pre-training



std. enc-dec



Causal LM



Prefix LM

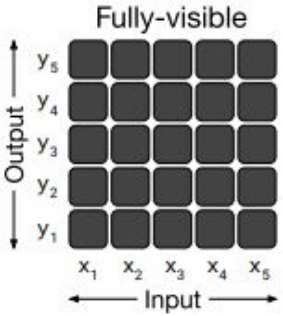
Targets

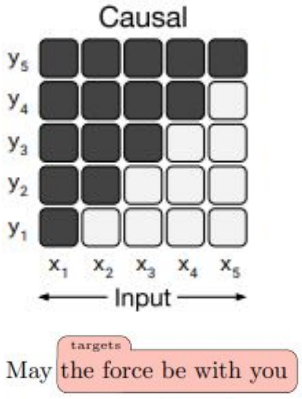
<X> the force

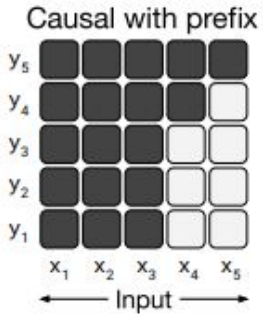
Inputs

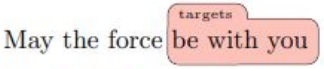
May <X> be with you

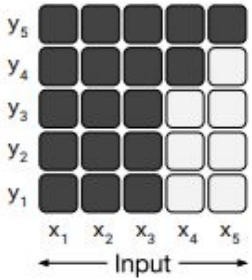
Masked LM







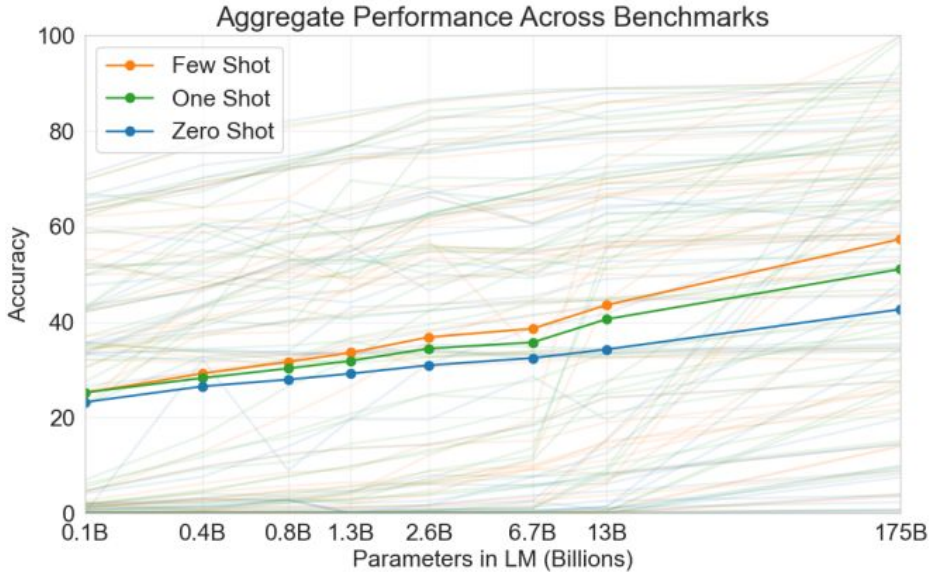




figs: T5

Google

Background: Scaling and zero shot capabilities



? GPT4

?

fig: GPT3

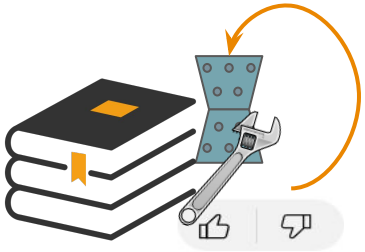
Model training

Pre-training

Self-supervised, on lots of examples O(Trillion) tokens. E.g. LLAMA

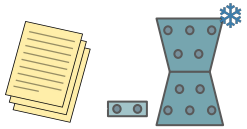
Fine-tuning

Enhanced for certain interactions. O(1000) to O(M) examples. E.g. PaLM-Coder, Codex



RL w. Human Feedback

Tuned to help model guess better under uncertainties. O(1000) examples. E.g. Bard, ChatGPT



Prompt-tuning

Steered for focused tasks. O(100)examples

Prompting

Steered for a query. O(1) examples (usually 0-8)

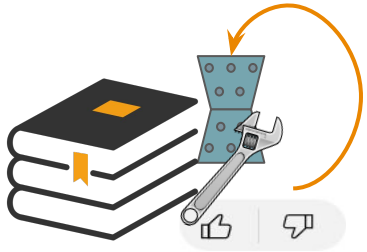
Model training

Pre-training

Self-supervised, on lots of examples O(Trillion) tokens. E.g. LLAMA

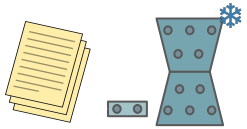
Fine-tuning

Enhanced for certain interactions. O(1000) to O(M) examples. E.g. PaLM-Coder, Codex



RL w. Human Feedback

Tuned to help model guess better under uncertainties. O(1000) examples. E.g. Bard, ChatGPT



Prompt-tuning

Steered for focused tasks. O(100)examples

Prompting

Steered for a query. O(1) examples (usually 0-8)

Pre-training, instruction based fine-tuning, RLHF - usually successive

These slides = (Prompting + prompt-tuning and a dash of fine-tuning)

Available models

Raw LM

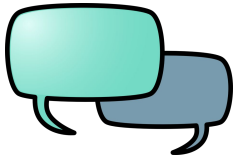
Continues to predict next tokens following a prompt

E.g. LLAMA, GPT3

Instruction tuned

Answer yes or no. Cheetahs are faster than hares. Answer:

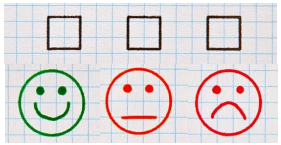
E.g. FLAN-PaLM, InstructGPT



Dialog tuned

[Alice] where is eve?
[Bob] she is out.
[Alice] shall i call her in?
[Bob]: ...

E.g. BlenderBot, Koala



RLHF

Human-Feedback to improve the model's responses under uncertainties.

E.g. Bard, ChatGPT

Shapes of Prompt Engineering

Background

01 Zero-shot, Few-shot

02 Consistency decoding

03 Learned prompt tokens

04 Instruction Tuning

05 Chain-of-Thought (+ with-action)

06 Retrieval augmented

07 Tool augmented

08 RLHF

References are not exhaustive. If something is incorrect or I’m missing your work or references to works you like PLMK!

Zero-shot: State your problem

```
Alice bought 5 balloons for 50 cents. Each
balloon costs
```

Zero-shot with instruction

```
Solve these math problems involving money.

Question: Alice bought 5 balloons for 50 cents.
Each balloon costs

Answer:
```

Zero-shot conditional prompts

constraint based

```
Write a short story about friendship in
2000 or fewer words.
```

```
Write a sentence where the first letter of
each word forms the acronym i i p i t b.
```

priming

```
You are Jeff Dean, one of the world's best
programmers and software engineers. Write an
efficient program to …
```

Few-shot (In-Context Learning): Give some examples

2-shot

```
Alice bought 5 balloons for 50 cents. Each balloon
costs
Answer: 10 cents

An ice pop cost 2 cents. Bob wants 3 ice pops. He
has to pay
Answer: 6 cents

Eve bought 2 pencils from Dan, each costing 50
cents. Eve owes Dan
Answer:
```

Few-shot (In-Context Learning): Give some examples

2-shot, with instructions

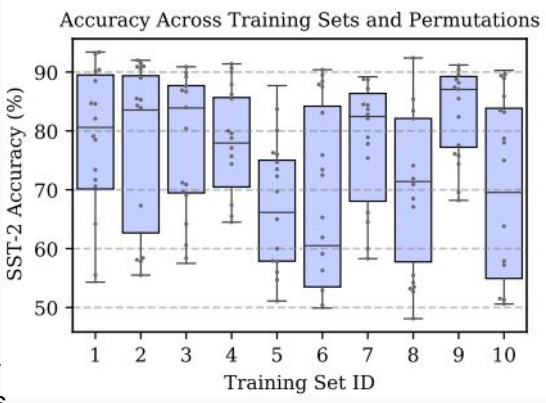
```
Solve these math problems involving money.
Alice bought 5 balloons for 50 cents. Each balloon
costs
Answer: 10 cents

An ice pop cost 2 cents. Bob wants 3 ice pops. He
has to pay
Answer: 6 cents

Eve bought 2 pencils from Dan, each costing 50
cents. Eve owes Dan
Answer:
```

Few-shot has high variance, bias

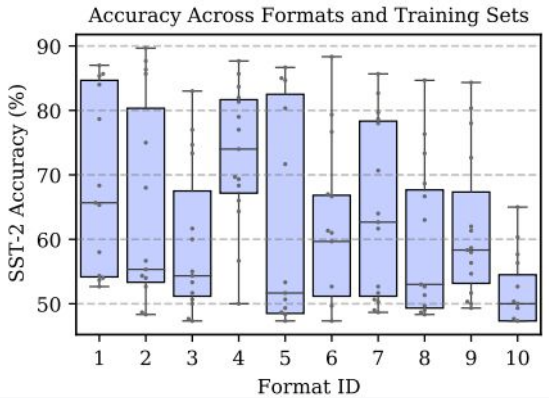
Issues leading to high variance

- Order of the few-shot examples matters
- Recency bias - preference for last
- Majority label bias - preference for majority class in the answers
- Common token bias - higher production of tokens that are more frequent

Plots showing high variance in accuracy for different prompts



Different few-shot examples (and permutations).



Different prompt formats (and few-shot examples)

credit: Zhao et. al.

Few-shot variance: Mitigation strategies

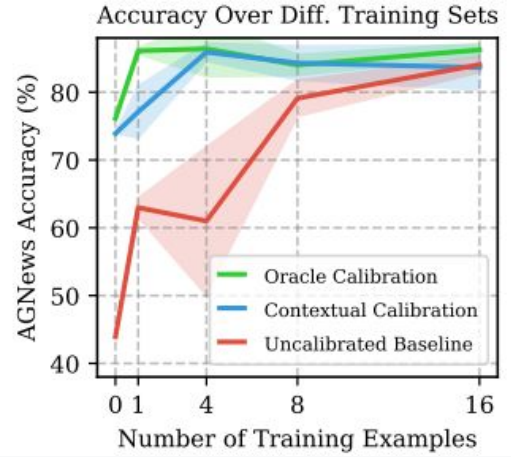
Mitigation strategies

- Select diverse examples
- Select few shot samples relevant to the test sample
- Ensure random order
- Calibrate on content-free examples

Few-shot variance mitigation: Calibration

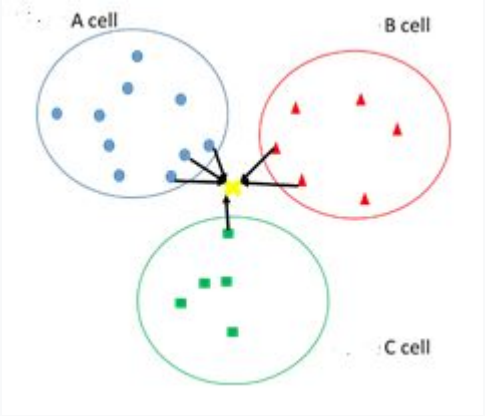
Calibration

- Calibration W*p + b
- Fit calibration parameters such that prediction for all labels on N/A is uniform
- Content-free inputs = {“N/A”, “[MASK]”, “”}
- p_cf = class probabilities on content-free inputs
- W = diag(p_cf)$^{-1}$,b=0; For generation, b= -p_cf of first token (W is Identity)



credit: Zhao et. al.

Selecting few-shots



- Choose nearest train/val examples in embedding space
  - KATE-kNN based on BERT/Roberta/XLNet embeddings
  - Vote-k Clusters and select diverse and representative samples
    - Select different few-shot examples for each test query.
- EPR - Trains a dense retriever (using the task training set) to select few-shots
  - Given(x,y) uses unsupervised retriever (e.g. BM25) to select similar (p,q)
  - Scores P(y| p, q, x) and assigns (p,q) as +ve or -ve
  - Uses contrastive learning to train a retriever on the labeled examples.
- Active-Prompt - active learning to select most uncertain examples to annotate
  - Unlabeled query + zero or few-shot examples → sample k answers
  - Determine uncertainty from samples, and annotate uncertain examples
  - Use annotated examples as few-shot examples for the real test queries

Self-consistency sampling a.k.a Consistency Decoding

Self-consistency: Wang '23

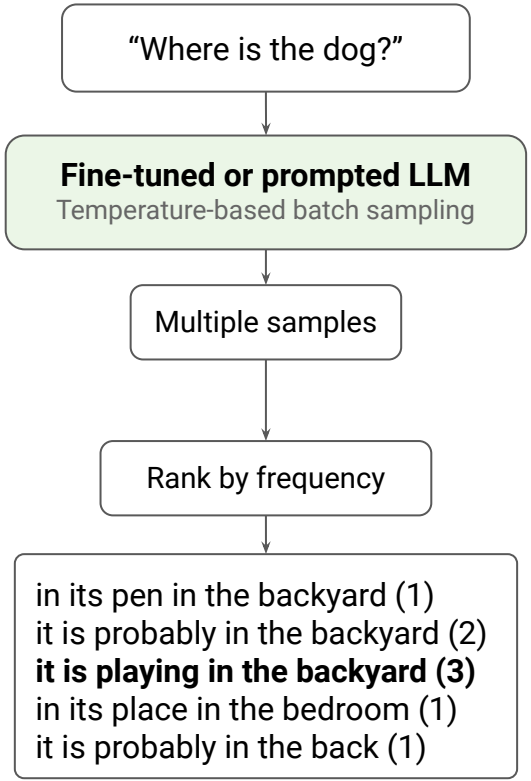


Image: Cai '22

Self-consistency alternatives

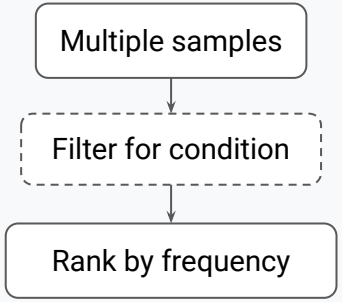
- For conditional prompts, verify whether the condition is satisfied

```
Write a summary of this webpage in 1000 or
fewer words.

Write a sentence where the first letter of
the words start with a b c d
```

Self-consistency alternatives

- For conditional prompts, verify whether the condition is satisfied

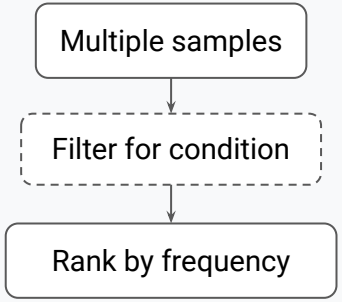
```
Write a summary of this webpage in 1000 or
fewer words.

Write a sentence where the first letter of
the words start with a b c d
```



Self-consistency alternatives

- For conditional prompts, verify whether the condition is satisfied

```
Write a summary of this webpage in 1000 or
fewer words.

Write a sentence where the first letter of
the words start with a b c d
```



- For coding problems
  - Write unit tests ⟶ run through interpreter ⟶ verify and select results that pass the unit test

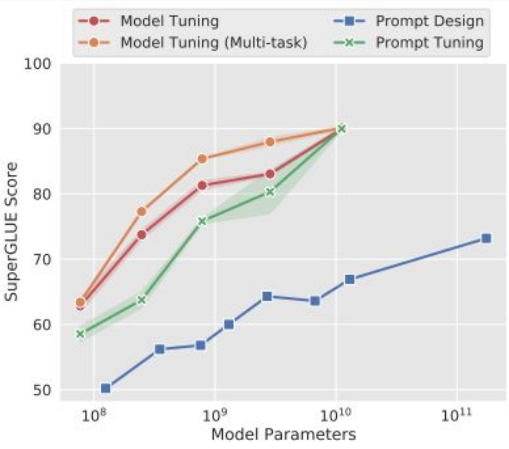
Learned Prompt Tokens

Idea: prompts are prefix-tokens and can be treated as trainable params.

optimize for prompts directly in the token/embedding space

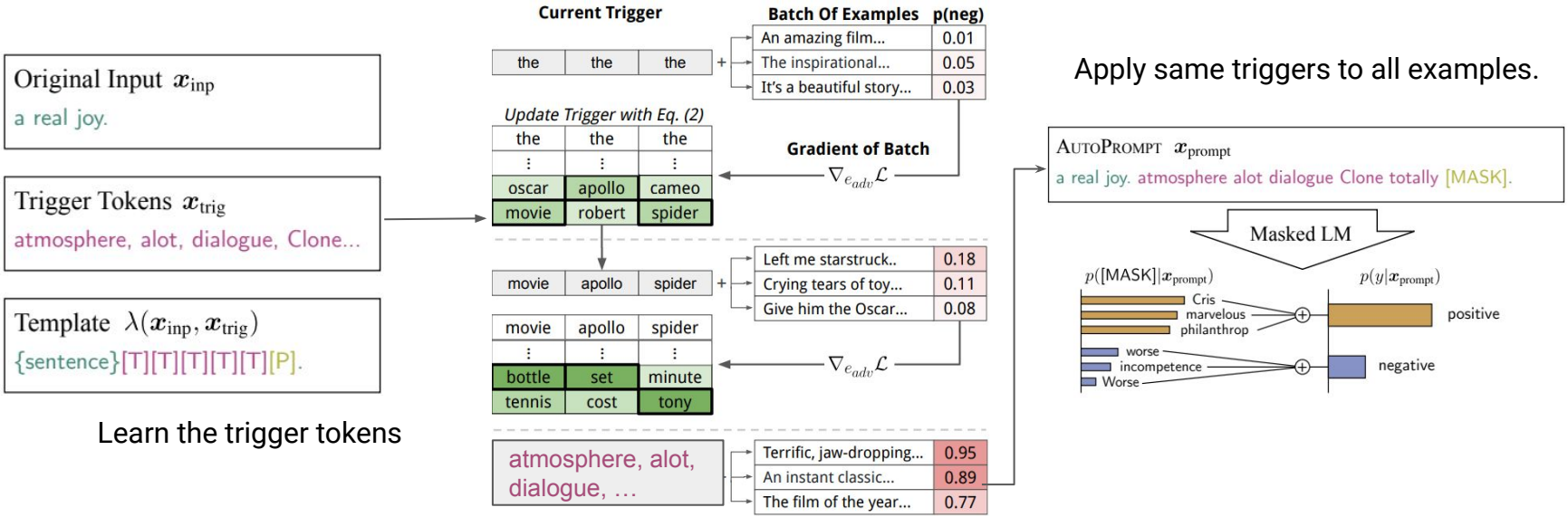
Learned Prompt Tokens - Pros

- Usually only learn a small number of trainable params.
- Usually task-specific.
- Keeps the model frozen. So steer the LM.
- Works well with smaller amount of examples
  - especially, if you have lot more examples than few-shot
  - can also learn on full training data
- In low data setting (O(10) - O(10^3)) can outperform fine-tuning.
- Comparable to or outperform linear-probing.
- Better generalization to OOD examples not in training.
- Still serve only the frozen model! Modify query with learned prompt.



Learned Prompt Tokens - AutoPrompt

Learns a set of “trigger” tokens using gradient-based search.

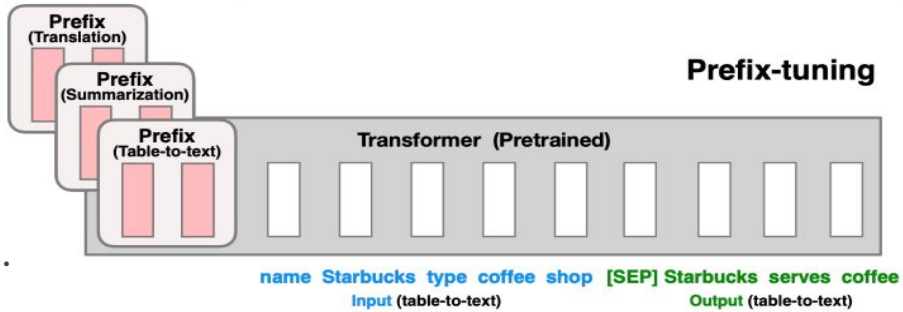


- For Masked LMs.
- Learned triggers appear to generalize to other models! (BERT, RoBERTa, GPT-2)

AutoPrompt - Shin ’20
Triggers - Wallace ‘19

Learned Prompt Tokens - Prefix-Tuning

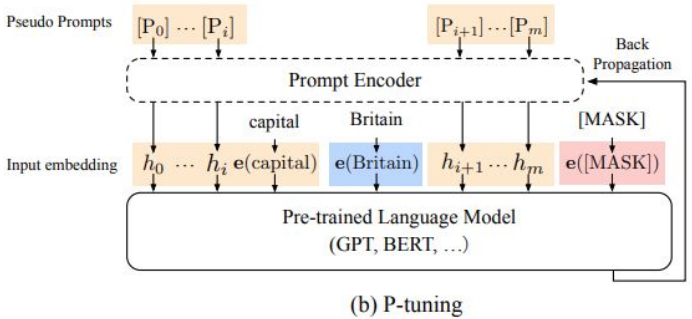
- Add prefix embeddings [PREFIX;x;y]
- Additional embeddings in all the transformer layers.
- It's parameterized as
  - A smaller matrix - prefix length x c
  - Feedforward MLP - c x emb_dim
- [x;INFIX;y] didn't work as well.
- Initialization - real words better than random.



Prefix-Tuning Li & Liang '21

Learned Prompt Tokens - P-tuning

- Additional embeddings only in the input encoder.
- Follows this template T = {[$P_{0:i}$], x, [$P_{i+1:m}$], y}
- With encoder it gets mapped to
  - {h_0, ..., h_i, e(x), h_{i+1}, ..., h_m, e(y)}
- Optimize and learn the h_i directly
- It's parameterized as
  - Hidden layers of an LSTM followed by MLP
  - $$h_i = \text{MLP}([\overrightarrow{h_i} : \overleftarrow{h_i}]) = \text{MLP}([\text{LSTM}(h_{0:i}) : \text{LSTM}(h_{i:m})])$$
- Initialization - with embeddings of real words

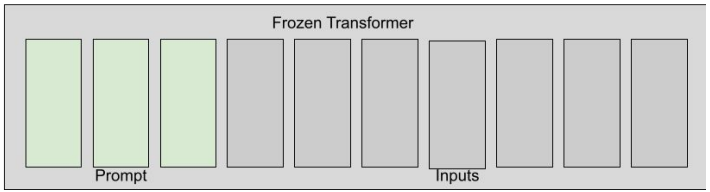


(b) P-tuning

P-tuning Liu '21

Learned Prompt Tokens - Prompt-tuning

- Similar to prefix-tuning [$P_{0:k}$;x;y]
- Only in the input embedding layer.
- P-> k x emb_dim, x-> n x emb_dim
- Initialization: the following were better than random
  - Can use embedding values of class label strings
  - Sample from top (frequent) 5K words in vocab



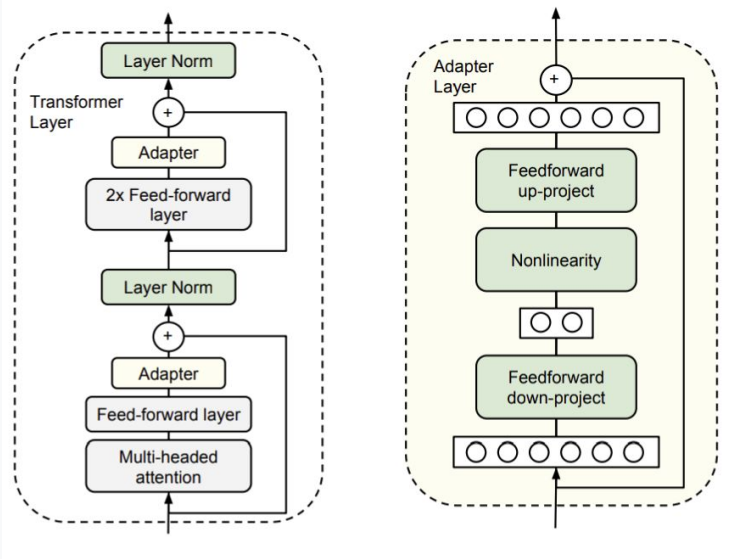
Prompt-tuning Lester ‘21

Learned Prompt Tokens - Implementation notes

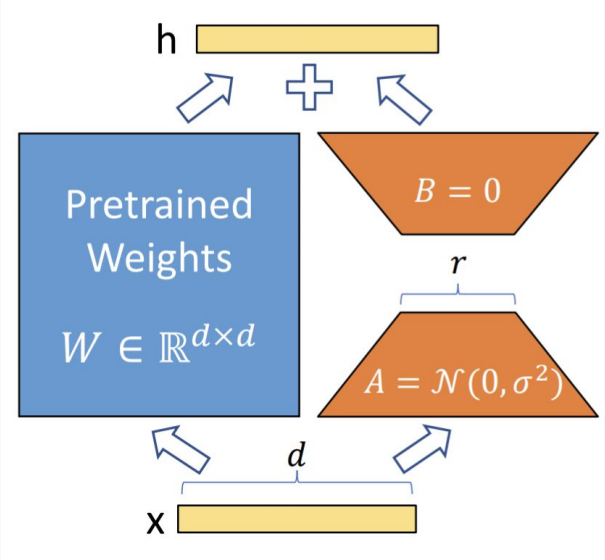
- Number of tokens.
  - Prefix-tuning: 200 for summarization, 10 for Table-2-text
  - Prompt-tuning: 5 to 100
- Great to try for low data setting.
- Initialization.
  - Embedding initialized with frequent words >> random init.
- *Promising for personalization - only store small set of weights*
- *You still only serve the main frozen model!*

Related: Adapters and Low-rank Adaptation (fine-tuning)

Keep model frozen and adapt some layers when tuning (supervised fine-tuning).



PEFT Houslby' 19



LoRA

Can keep frozen model and share just weight for adapter layers.

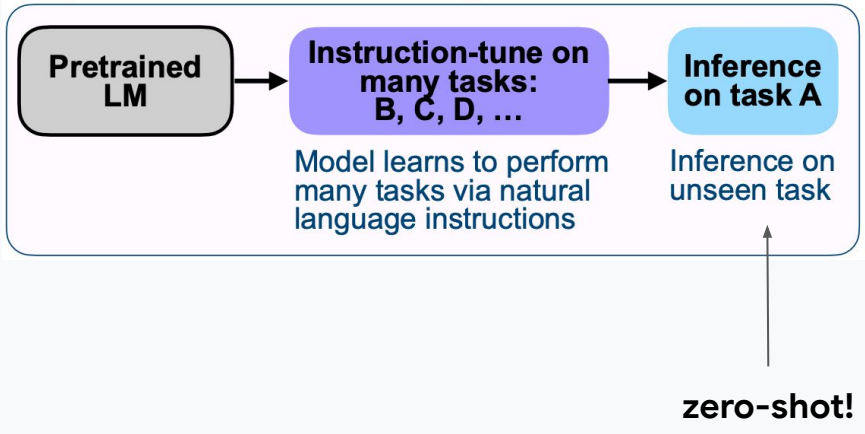
Instruction Tuning

fine-tuning a language model on a collection of tasks described via instructions in different ways —improves the zero-shot performance of language models on unseen tasks.

FLAN, T0

Instruction Tuning

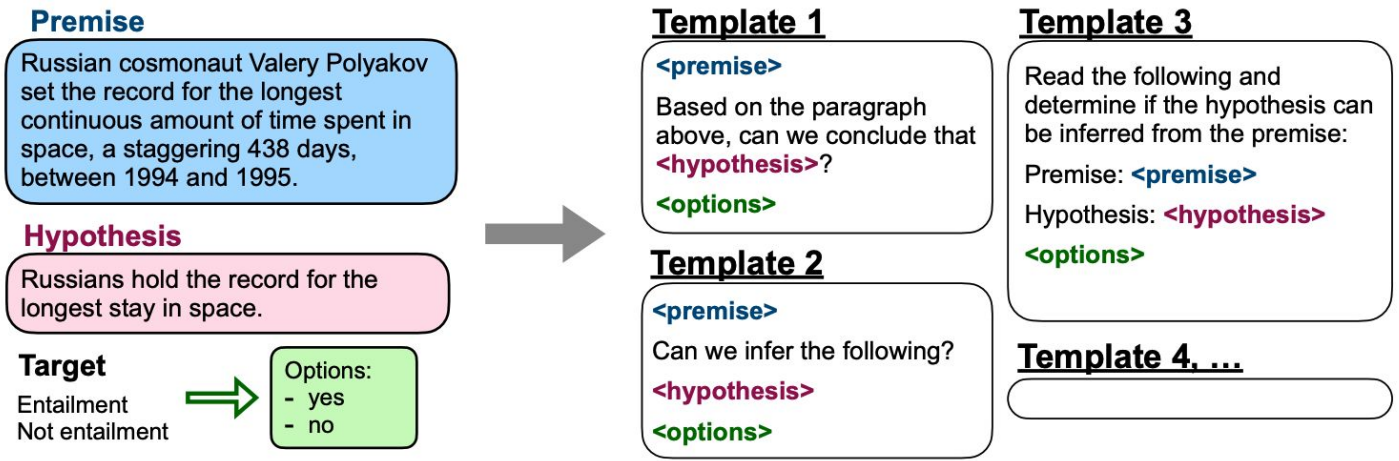
1. Cluster different NLP tasks into categories.
    a. Classification, Entity Recognition, ...
2. Generate templates for describing the instruction for each task cluster/category.
3. Apply generated instruction templates for tasks in the cluster; for all clusters.
4. Fine-tune model on generated (instruction, input, ground truth output) data



Supervised fine-tuning to teach the model to perform many NLP tasks.

image: FLAN

Instruction Tuning: Templates



Generate many (~10) natural instruction templates for each task.

image: FLAN

Some (2-3) templates reverse the task e.g. movie review rating task:

Forward template: Given {{review}}. The rating for this review is {{rating}}

Reverse template: Given {{rating}}. A plausible review is {{review}}

Instruction Tuning: Apply templates to examples

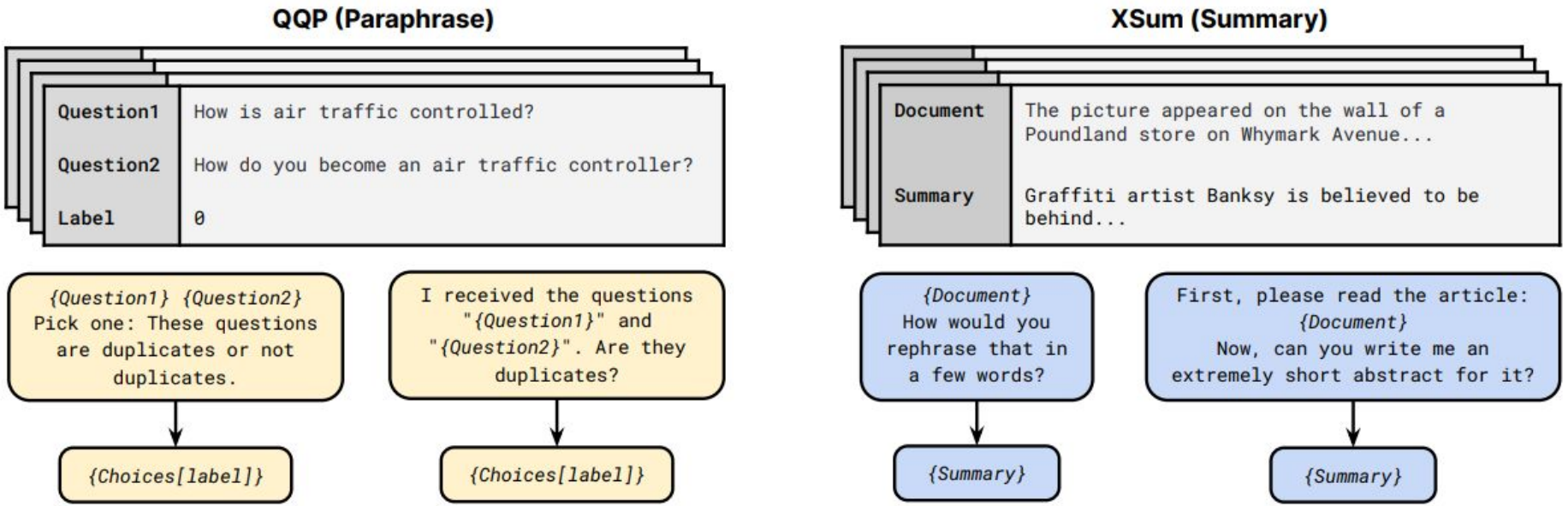
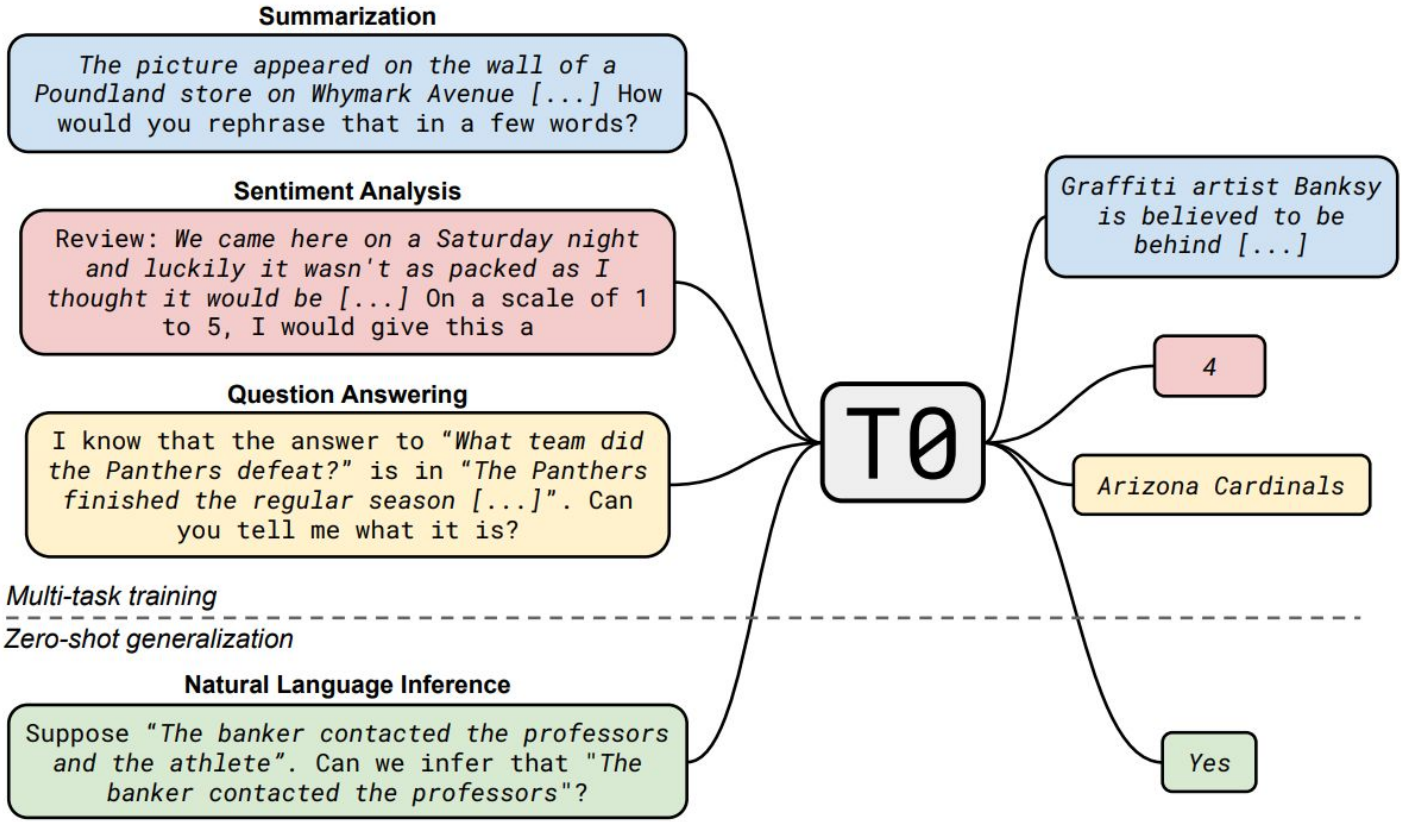


image: [T0]

Instruction Tuning: Multi-task fine-tuning



(notes) Instruction Tuning: FLAN vs T0

- Significantly improves performance making smaller models comparable to larger models having 6x-16x more params.
- Concurrent, mostly similar conclusions
- Works well on enc-decoder (3B, 11B) and larger decoder-only (>8B) models
- For generation (sentence completion tasks) instructions don't matter, perhaps better to even remove.
- T0's templates are more diverse qualitatively - adding more prompts improves performance (whereas FLAN sees some saturation).
- Prompt, template, datasets: promptsource, self-instruct, (internal flan-tuning)

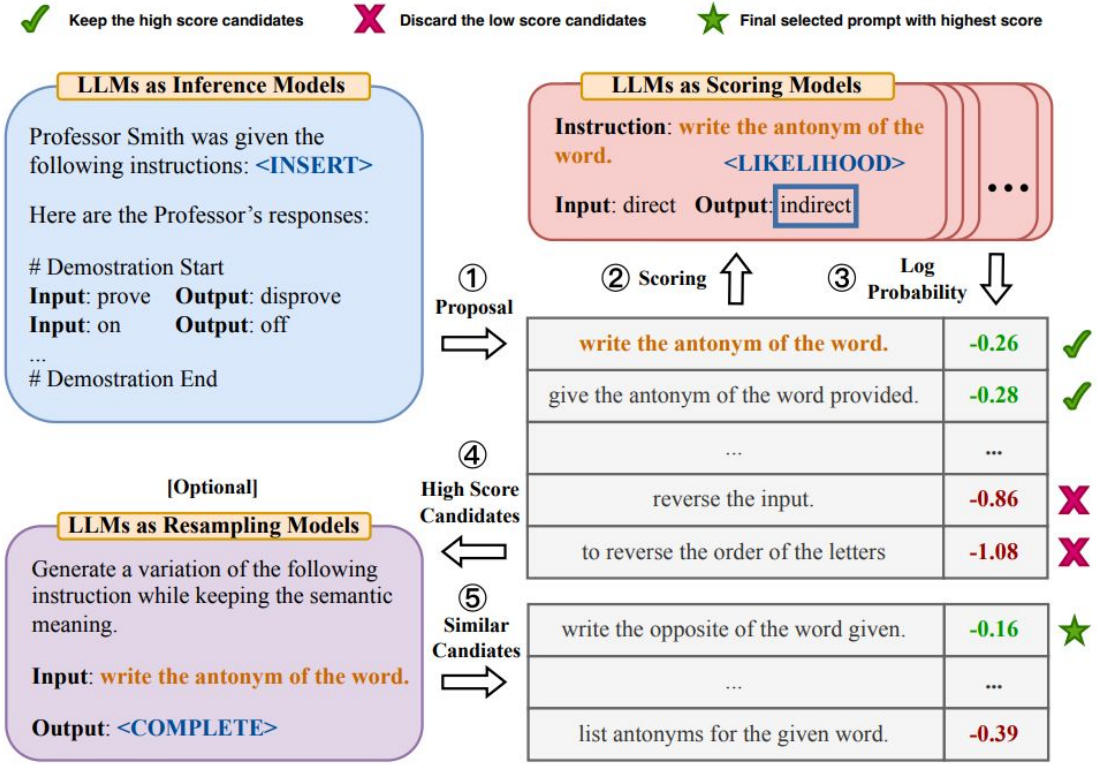
INSTRUCTOR Instruction tuning also works for models used to generate embeddings!

Instruction Tuning + few-shot multi-task

ICIL
Few-shots of different tasks even after instruction -tuning helps improve results.

```
Definition: Determine the speaker of the dialogue, "agent" or "customer".
Input: I have successfully booked your ticket.
Output: agent

Definition: Classify a given post into 1) 'Similar'  and 2) 'Dissimilar'.
Input: Sentence1: Should I ask to be [..]
Output: Dissimilar

Definition: Detect which category […] , "Quantity", and "Location".
Input: What is the oldest building in the U.S?
Output: Location

Definition: Answer a given question containing a blank (_).
Input: Jon ate the oatmeal [..] _ was spoiled.
Output:
```

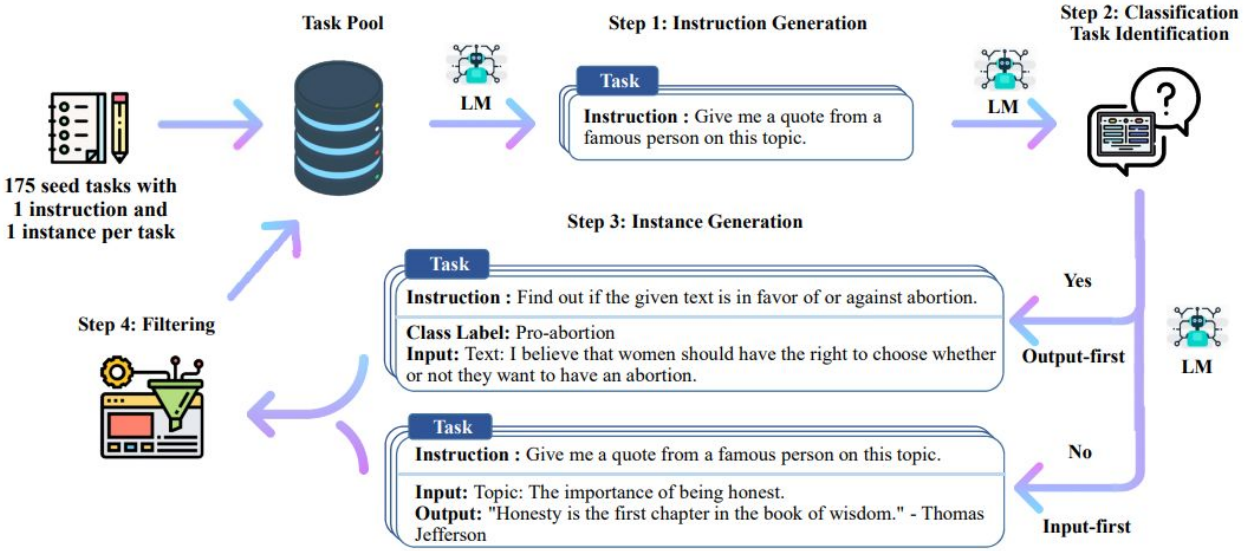
LLMs to generate, score and rephrase instructions



[APE (Zhou '23)]

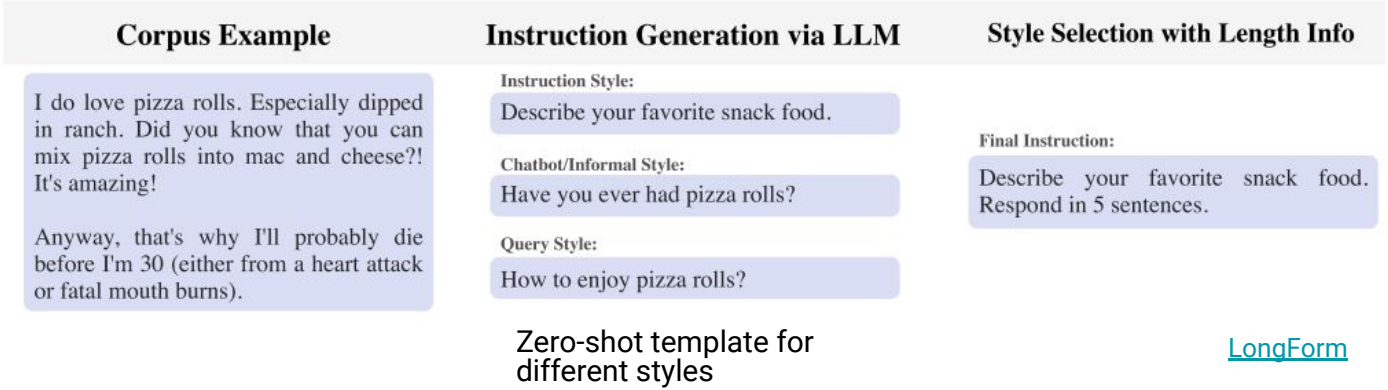
Choose the instructions that score higher on the desired target

Bootstrapping an LLM to generate and annotate data



Self-instruct

LLMs to generate instructions in different styles



Zero-shot template for different styles

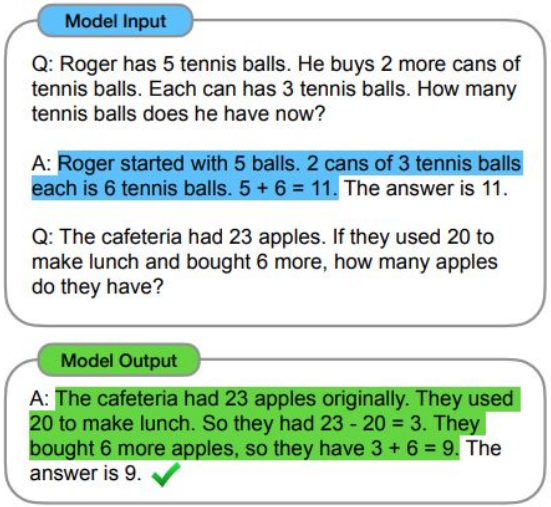
LongForm

Results show good performance for generation tasks requiring longer responses.

Chain-of-Thought

Idea: Make the model reason and output intermediate steps

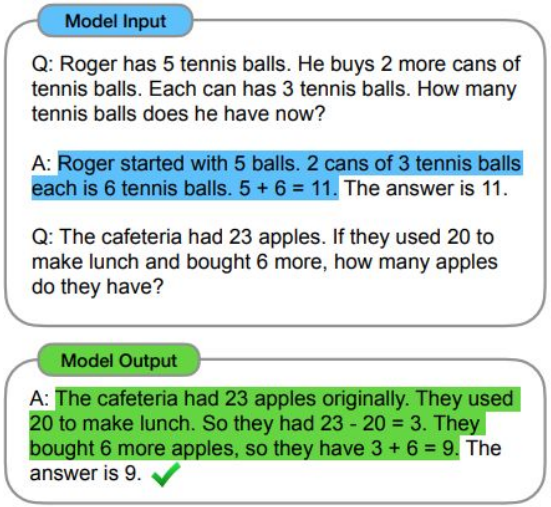
Chain-of-Thought (CoT)



CoT Wei '22

Chain-of-Thought (CoT)

few-shot



CoT Wei '22

zero-shot

```
Three cats and two birds have how many legs?
Answer: Let's think step-by-step.
```

Kojima '22

Chain-of-Thought (CoT): Phrasing matters

No.	Category	Zero-shot CoT Trigger Prompt	Accuracy
1	APE	Let's work this out in a step by step way to be sure we have the right answer.	**82.0**
2	Human-Designed	Let's think step by step. (*1)	78.7
3		First, (*2)	77.3
4		Let's think about this logically.	74.5
5		Let's solve this problem by splitting it into steps. (*3)	72.2
6		Let's be realistic and think step by step.	70.8
7		Let's think like a detective step by step.	70.3
8		Let's think	57.5
9		Before we dive into the answer,	55.7
10		The answer is after the proof.	45.7
-		(Zero-shot)	17.7

Zhou '23

- Changing 'Q' to 'Question',
- separating CoT steps with '\n' (as opposed to `` `step i` ``)

Fu '22

CoT - bootstrapping strategies

- CoT can hurt performance marginally on some NLP benchmark/tasks. [Ye & Durrett '22]
  - E.g. NLI, when explanation is non-factual
- Augment with model generated rationales [Wang et. al. '22]
  - Have the model generate explanations along with the answer. [Weigreff '22]
  - Ensemble based on the answers and pick based on majority.
- STaR - Self Taught Reasoner [Zelikman'22]
  - Few-shot with CoT → bootstrap to generate rationales on larger dataset.
  - Then fine-tune on examples with rationales that yielded correct answers. Repeat
- Complexity-based consistency - choose rationales with more steps when ensembling
  - Works for math and complex reasoning tasks.
  - May not work on simple QA tasks [Shum '23]

CoT - bootstrapping strategies

- Auto-CoT - Generate few-shot CoT examples.
  - Cluster questions in embedding space. Select question from each cluster.
  - Generate zero-shot CoT using heuristics (*Let's think step by step*)
    - Errors for rationales also tend to cluster, hence selecting examples from each cluster of questions prevents errors.
- Augment, Prune, Select [Shum '23]
  - Augment each input, output with model generated rationale.
  - Prune to only keep those examples where the final answer is correct.
    - Gives K high quality exemplars
  - Select: use variation reduced policy gradient estimator
    - Learn prob dist. over exemplars on a train set (similar task)
    - Prob. dist. over examples -> policy, val set accuracy -> reward

Retrieve from LM's memory

Liu '22

When given a question

1. First generate knowledge from model's memory
2. Then produce answer

Task	NumerSense	QASC
Prompt	Generate some numerical facts about objects. Examples:	Generate some knowledge about the input. Examples:
	Input: **penguins have <mask> wings.** Knowledge: *Birds have two wings. Penguin is a kind of bird.*	Input: **What type of water formation is formed by clouds?** Knowledge: *Clouds are made of water vapor.*
	...	...
	Input: **a typical human being has <mask> limbs.** Knowledge: *Human has two arms and two legs.*	Input: **The process by which genes are passed is** Knowledge: *Genes are passed from parent to offspring.*
	Input: **{question}** Knowledge:	Input: **{question}** Knowledge:

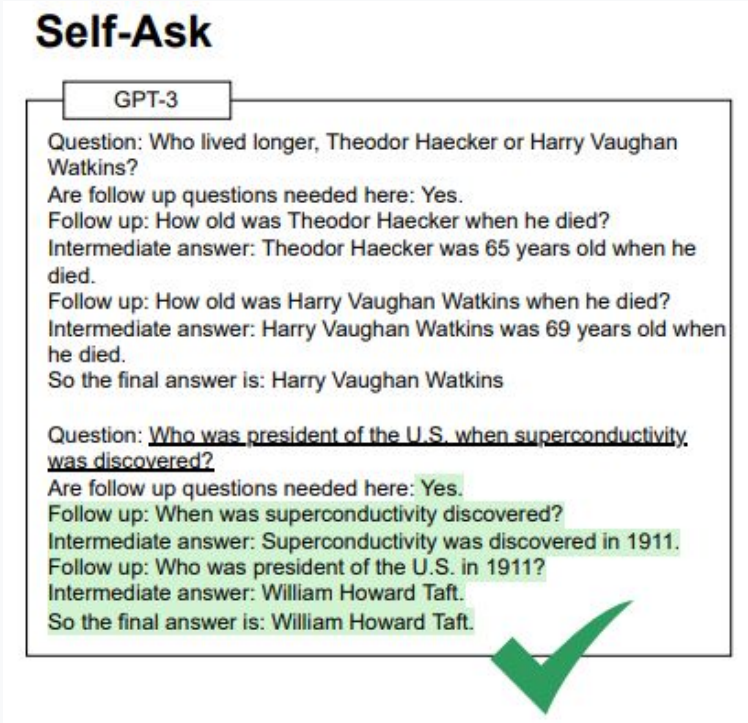
Indicates model can follow some templates.

CoT templates enable composability

Self-Ask [Press '22]

- CoT helps multi-hop reasoning.
- Add structure and ask **follow-up** questions
- **Intermediate answers** answered by model

Having structure and template in CoT examples is more powerful in improving performance on compositional tasks.



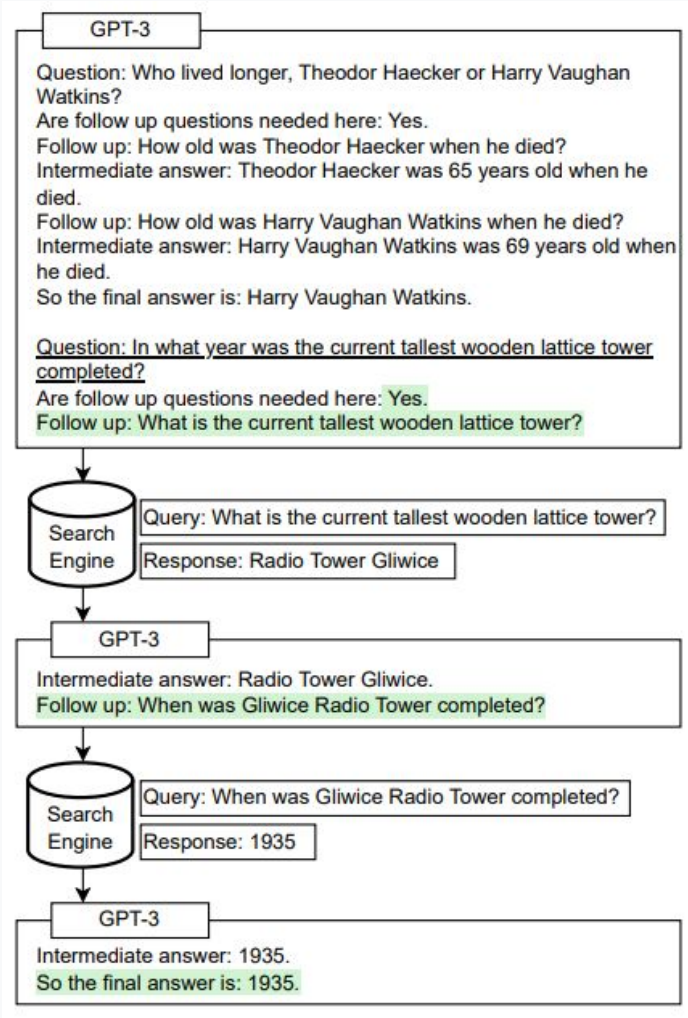
CoT templates enable composability

Self-Ask [Press '22]

- CoT helps multi-hop reasoning.
- Add structure and ask **follow-up** questions
- **Intermediate answers** answered by model

Having structure and template in CoT examples is more powerful in improving performance on compositional tasks.

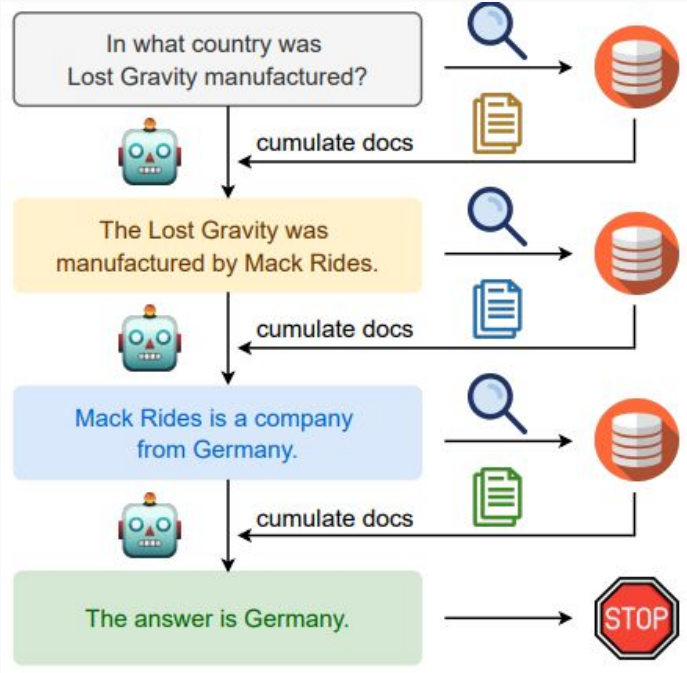
- Structure allows to add search engine query



CoT templates enable composability w. tools

IRCoT Trivedi '22

Interleave Retrieval with Chain of Thought



ReAct Yao'22

Reason + Act

(1) Hotspot QA

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

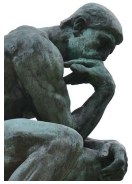
(1d) ReAct (Reason + Act)

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]

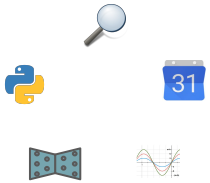


Self-critique

Iterative interaction to improve the model. Sort of bootstrapping.

Retrieval Augmented

Augment with external memory

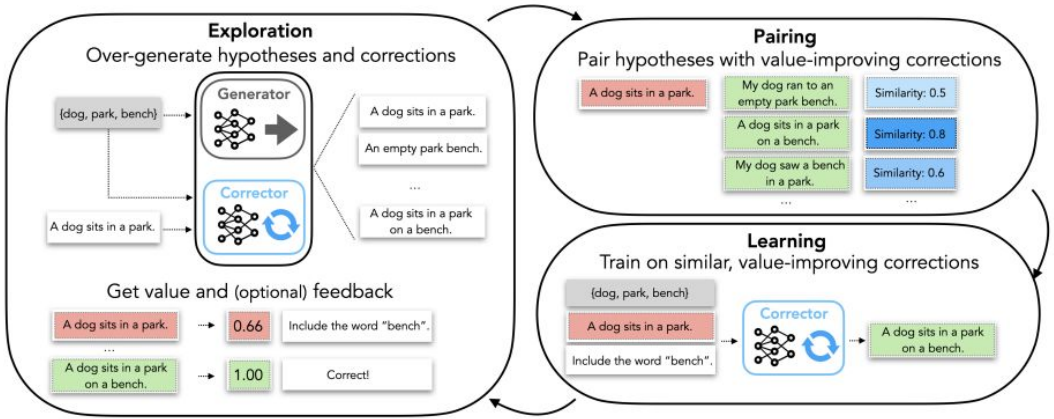


Tool Augmented

Augment with external tools, APIs, **

Self-Critique: Bootstrap LLM to correct it's outputs

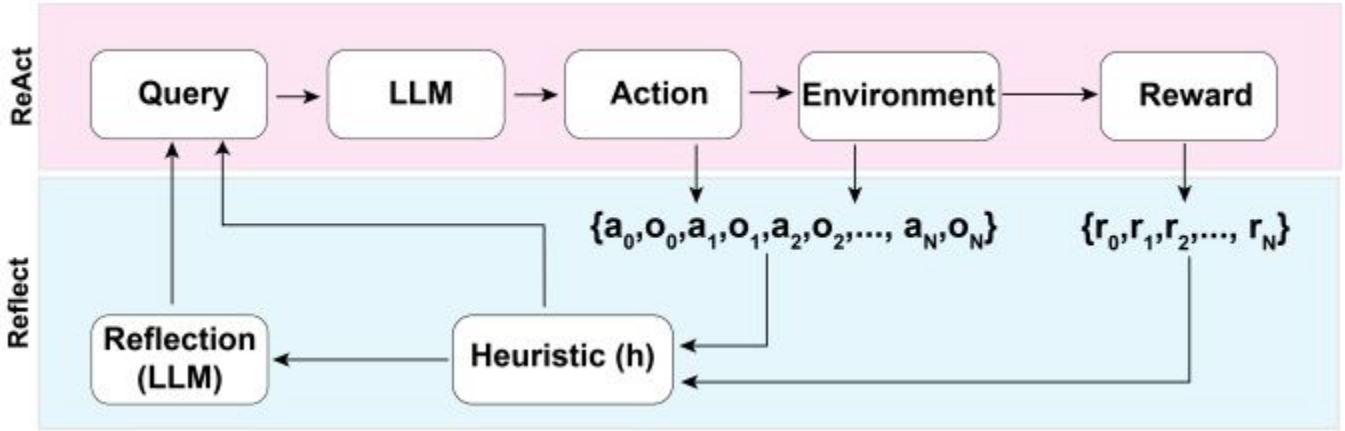
Self-correct



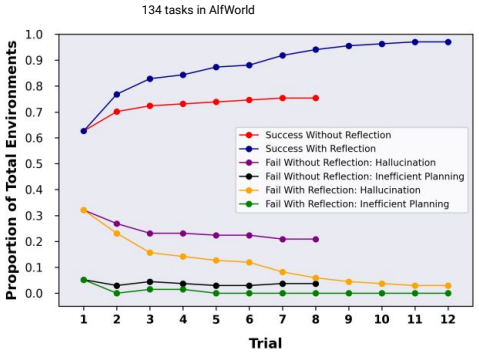
Generator and Corrector are different models.

value function is a classifier

Self-critique: LLM to correct itself iteratively



Reflexion



Build on ReAct framework. No policy is learned.

Heuristic (h) →decides to reflect if {a,o} has been repetitive or t (time steps) is large.

Reflection: LLM w. 2-shot prompt specific to dataset/task. Something like:

- I was stuck in loop where < obs. was repeated>. It did not help. Instead <what the agent should have done>
- In this environment <the task was X>. I did <something in a different order>. Next time I should do <...>

Reward - binary. Is the action valid (in case of RL environment) or final answer correct.
Action space - few-shot prompt demonstration of permissible actions.

Self-Debug

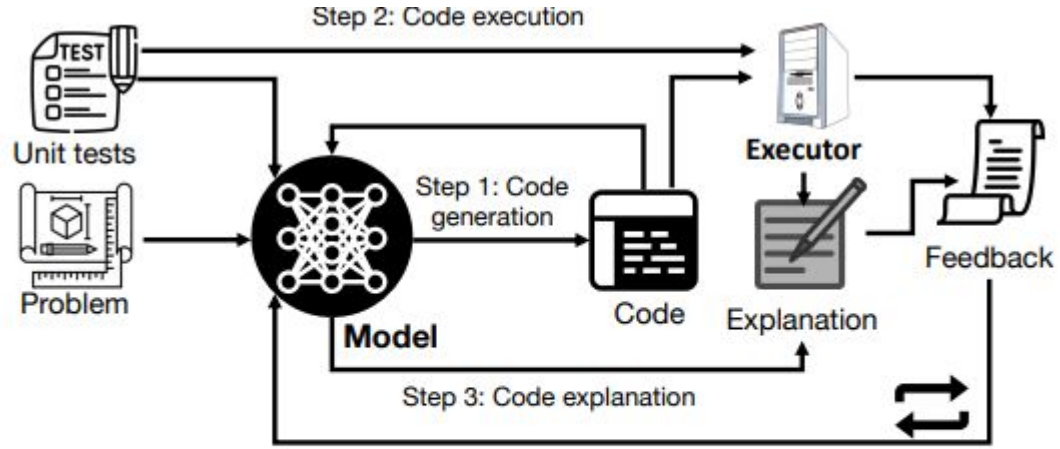


Figure 1: SELF-DEBUGGING for iterative debugging using a large language model. At each debugging step, the model first generates new code, then the code is executed and the model explains the code. The code explanation along with the execution results constitute the feedback message, which is then sent back to the model to perform more debugging steps. When unit tests are not available, the feedback can be purely based on code explanation.

self-debug

[Original Python] =

```
def count_trailing_zeroes_factorial_number(n):
    cnt = 0
    i = 5
    while ((n / i) >= 1):
        cnt = (cnt + (n / i))
        i *= 5
    return cnt
```

[Simple Feedback] =

The above Python translation does not do the same thing as the C++ code. Correct the Python translation.

[UT Feedback] =

The Python translation does not do the same thing as the C++ code. These are the results of failed unit tests that test whether the Python translation's outputs match the C++ program's outputs:

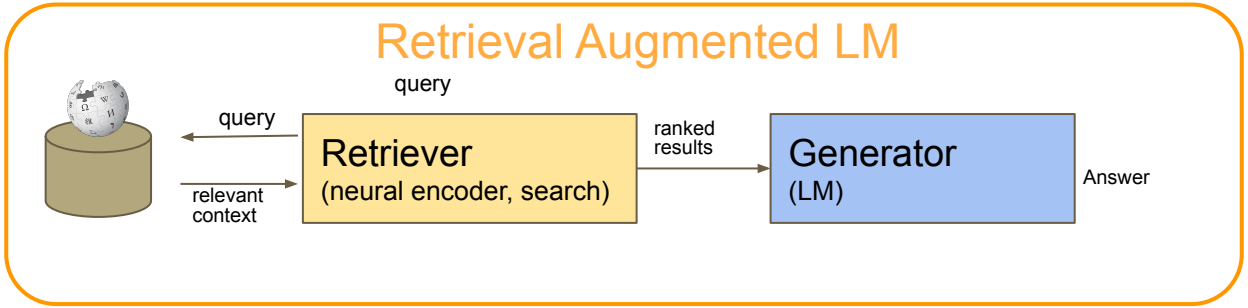
Failed: `assert count_trailing_zeroes_factorial_number(9) == 1`

Actual Result: `1.8`

Correct the translation.

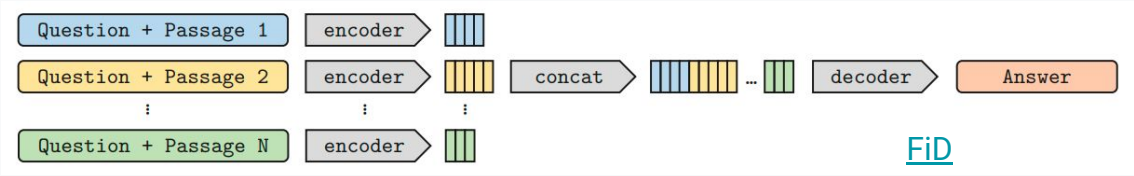
[Revised Python #n] =

```
def count_trailing_zeroes_factorial_number(n):
    cnt = 0
    i = 5
    while ((n / i) >= 1):
        cnt = (cnt + (n // i))
        i *= 5
    return cnt
```



Variations based on how you train the retriever and how you integrate the retrieved information into the answer.

Retrieval augmented LM



- User submits a query
- Retriever - essentially “search” but in LM papers it’s also differentiable
  - Consists of data - passages that are indexed e.g. via an encoder using a neural model
  - The query is encoded
  - Use similarity to retrieve relevant documents (ScAM, ScaNN, Faiss)
  - Rank top-n by relevance
- Generator - I think of “web answers”
  - Combines query and top-n responses
  - Composes answer

Retrieve from external knowledge base - variants

Train the retriever and generator jointly end-to-end

End-to-end

REALM
EMDR

Fine-tune the retrieval model

Fine-tuning

RAG
Izacard ‘20
ATLAS - pre-training+few-shot

Keep retrieval model frozen, tune layer integrating generator

Frozen (+ layer tuning)

RETRO
kNN-LM

Keep retriever and generator frozen. Engineer prompt.

Search+Prompt

LaMDA, BlenderBot, He’ 23

Augmented models survey Mialon’23

Retrieve from external knowledge base

Augmented models survey Mialon'23

Lazaridou '22 - placing the retrieved knowledge / evidence before asking the question is better

How to combine:
(a - answer, q - question, p_i - passage)

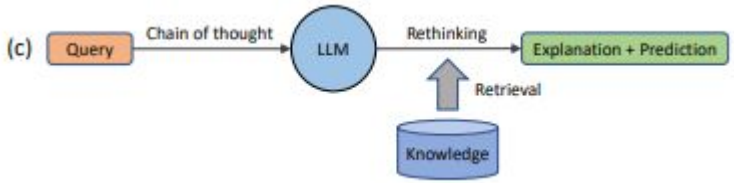
- Retrieval Augmented Generation - RAG '20
- Noisy-channel inference - Lewis '19
- Product of experts (PoE)

$$\text{maximizes } p(a \mid q) = \sum_{i=1}^{n} p_{tfidf}(p_i \mid q).p(a_i \mid q, p_i)$$

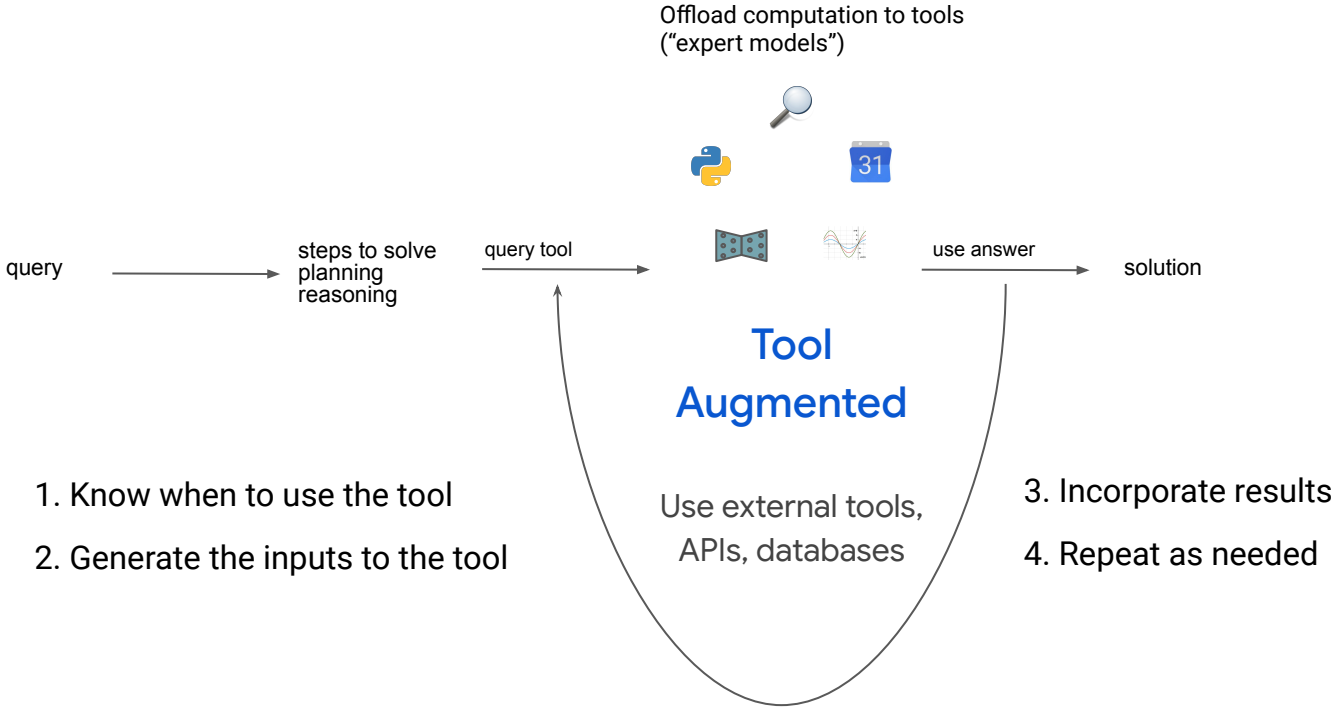
$$\text{maximizes } p(a_i, q \mid p_i) = \frac{p(q \mid a_i, p_i) \cdot p(a_i \mid p_i)}{p(q \mid p_i)}$$

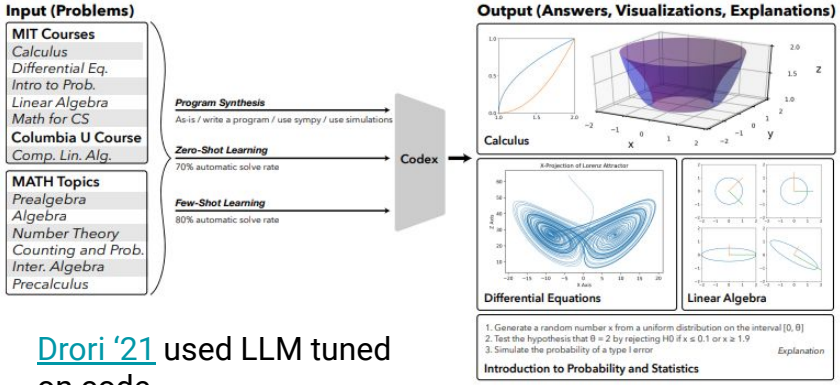
Get probs by using LM with few-shot prompt and scoring.

He '22 Retrieve from KB and add to prompt

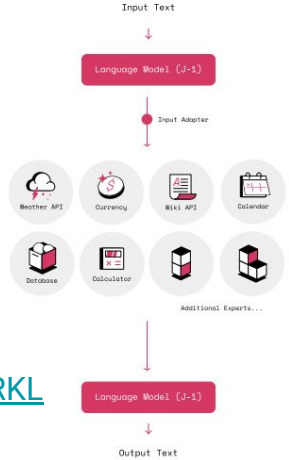


Tool Augmented Models





[Drori ‘21](#) used LLM tuned on code.



[MRKL](#)

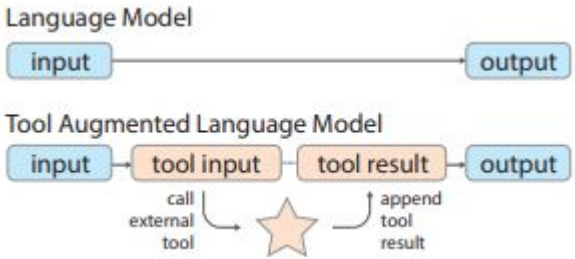
LLM extracts information and generates inputs/arguments to the tools or APIs

Tools
- Calculators
- Search Engines
- Calendar
- Weather API
- Database

[PAL](#) and [PoT](#) add Python interpreters building on Chain of Thought

```
interest_rate = Symbol('x')
sum_in_two_years_with_simple_interest= 20000 +
interest_rate * 20000 * 3
sum_in_two_years_with_compound_interest = 20000 * (1 +
interest_rate)**3
# Since compound interest is 1000 more than simple interest.
ans = solve(sum_after_in_yeras_with_compound_interest -
sum_after_two_years_in_compound_interest - 1000,
interest_rate)                                          PoT
```

[TALM](#)



Tools incorporated with few-shot, CoT examples, and templates like ReAct work with no additional tuning on LLMs!

TALM

An abstract task:
task input text |***tool-call*** tool input text |***result*** tool output text |***output*** task output text

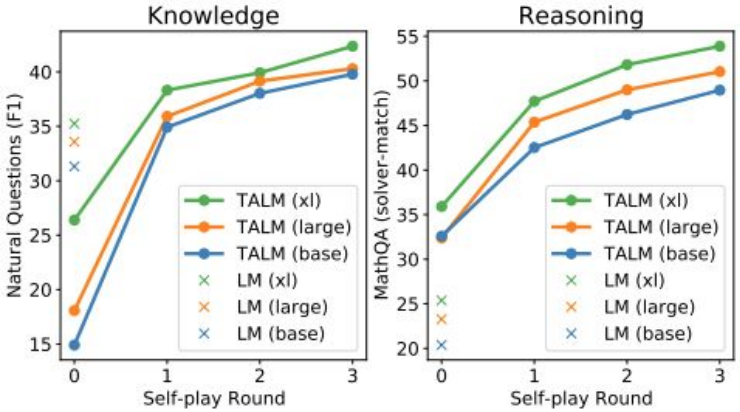
A weather task:
how hot will it get in NYC today? |***weather*** lookup region=NYC |***result*** precipitation chance: 10, high temp: 20c, low-temp: 12c |***output*** today's high will be 20C

Question: If Lily's test scores are 85 , 88 and 95 out of 100 in 3 different subjects , what will be her average score?
Formula: Divide(Add(85, Add(88, 95)), 3)
Answer: 89.33

|question If Lily's test scores are 85 , 88 and 95 out of 100 in 3 different subjects , what will be her average score? |formula Divide(Add(85, Add(88, 95)), 3) |result 89.3333333333 |output 89.33

Iterative Self-Play to finetune model for diverse invocations of API

```
# task set
# tool-use set

# self-play rounds
# finetune LM

# iterate task set

# sample tool query
# call tool API
# get task output
# filter wrong output

# update tool-use set
```



TALM

An abstract task:
task input text |*tool-call* tool input text |***result*** tool output text |*output* task output text

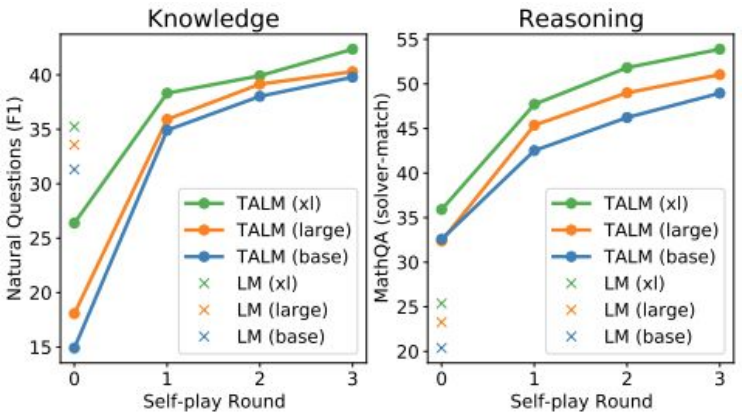
A weather task:
how hot will it get in NYC today? |*weather* lookup region=NYC |***result*** precipitation chance: 10, high temp: 20c, low-temp: 12c |*output* today's high will be 20C

Question: If Lily's test scores are 85 , 88 and 95 out of 100 in 3 different subjects , what will be her average score?
Formula: Divide(Add(85, Add(88, 95)), 3)
Answer: 89.33

|question If Lily's test scores are 85 , 88 and 95 out of 100 in 3 different subjects , what will be her average score? |formula Divide(Add(85, Add(88, 95)), 3) |result 89.3333333333 |output 89.33

Iterative Self-Play to finetune model for diverse invocations of API

```
# task set
# tool-use set

# self-play rounds
# finetune LM

# iterate task set

# sample tool query
# call tool API
# get task output
# filter wrong output

# update tool-use set
```



Toolformer

Generate annotations using few-shot on self-supervision data. Filter and then do **supervised tuning**.

Few-shot bootstrap on unlabeled data.

Sample (t=i) when to call.

Which API and arguments to pass.

Self-supervised loss on whether doing the API call and the result helped. Based on log prob of actual ground truth tokens.

Incorporate results.

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

Input: Joe Biden was born in Scranton, Pennsylvania.

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

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

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

Input: x

Output:

Use the process to annotate pre-training data and then **finetune**.

API
Question Answering
Wikipedia Search
Calculator
Calendar
Machine Translation

Cascades / Chaining

Interact with LLMs iteratively with prompt engineering, and retrieval and tools.

🦜🔗 LangChain

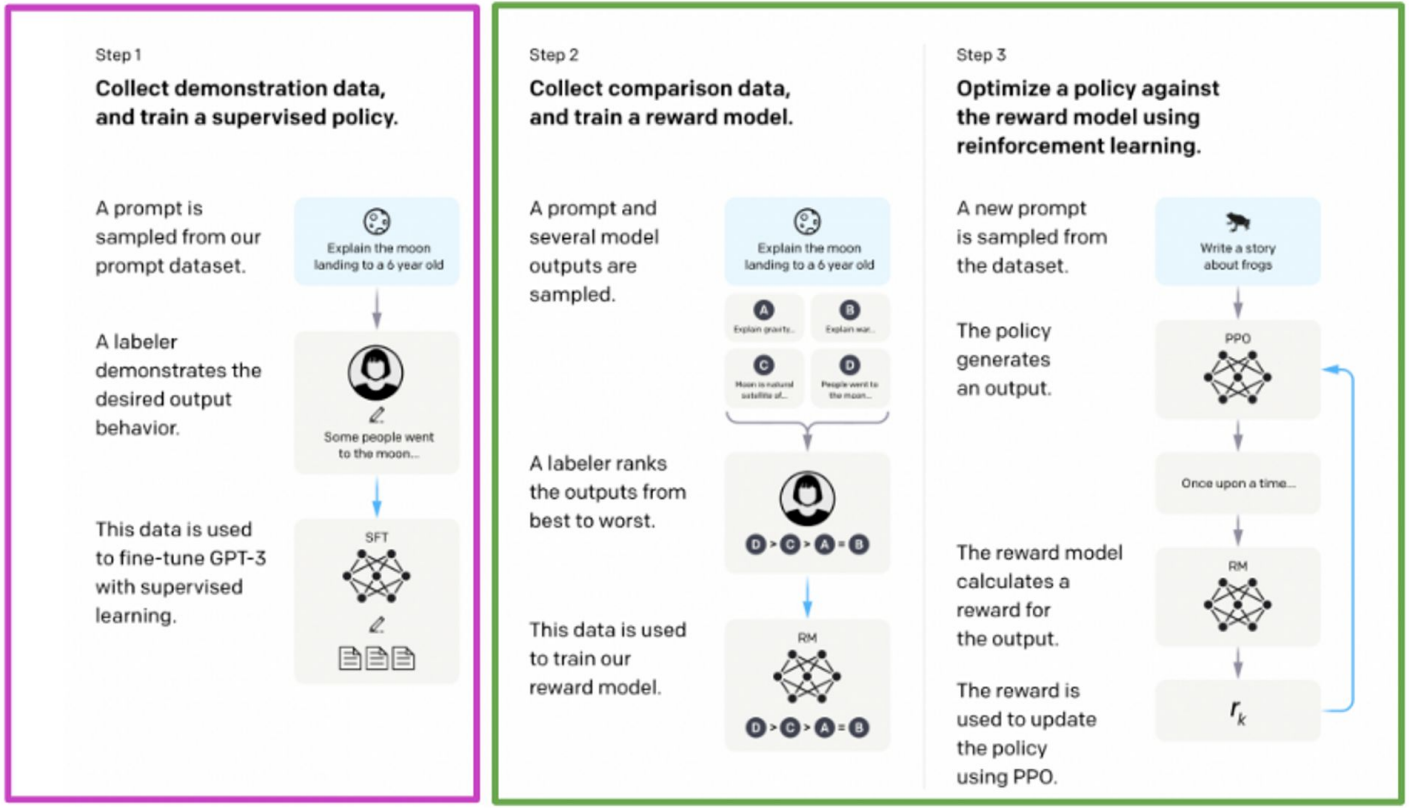
Human Feedback

Align with human preferences to reduce hallucinations, improve response under uncertainities, and calibrate model

See John Schulman's talk

Supervised Fine-tuning
(instruction following and chatty-ness)

Reinforcement learning with human feedback (RLHF)
(aligning to target values and safety)



InstructGPT

Rollout:

Reward model (r_θ) → trained separately on sets of K responses & query (x,y_w), (x,y_l).

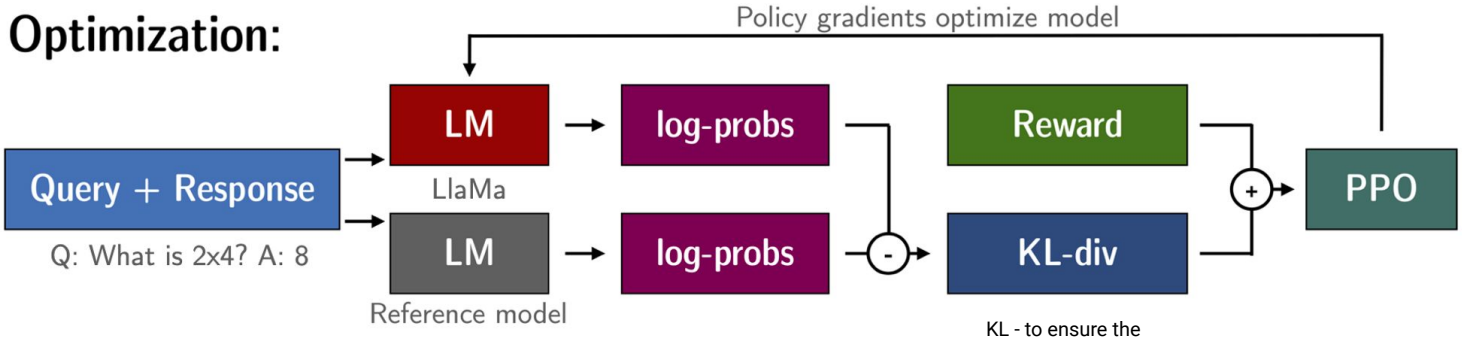
StackLLama K=2, InstructGPT K=4to9

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

Evaluation:

GPT4 - zero shot classifiers as rule based reward models (RBRM)

Optimization:



Same original model, but this is not updated.

KL - to ensure the response doesn't deviate too much (so that RL doesn't find some way to exploit the reward model)

StackLlama

Model calibration and RLHF

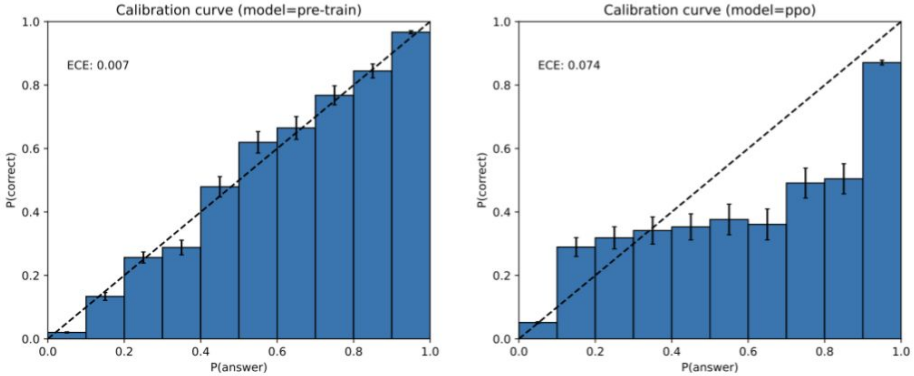


Figure 8. Left: Calibration plot of the pre-trained GPT-4 model on a subset of the MMLU dataset. On the x-axis are bins according to the model's confidence (logprob) in each of the A/B/C/D choices for each question; on the y-axis is the accuracy within each bin. The dotted diagonal line represents perfect calibration. Right: Calibration plot of the post-trained GPT-4 model on the same subset of MMLU. The post-training hurts calibration significantly.

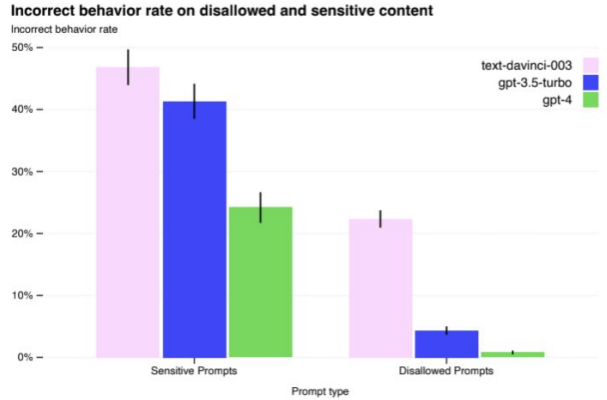


Figure 9. Rate of incorrect behavior on sensitive and disallowed prompts. Lower values are better. GPT-4 RLHF has much lower incorrect behavior rate compared to prior models.

GPT4

Issues

- Models hallucinate and can make up information. Unreliable.
    - It has the shape of the answer, but can be incorrect.
- Don't express uncertainty well e.g. They don't yet ask for clarification.
- Programming via prompts poses a significant security risk
    - Jailbreaks
    - Also apps that are chained could perhaps be rigged.
        - Ex. injecting stuff into python interpreter
- Cost - LLM agents (e.g AutoGPT) can spawn several threads compounding costs.
- More ...

Directions

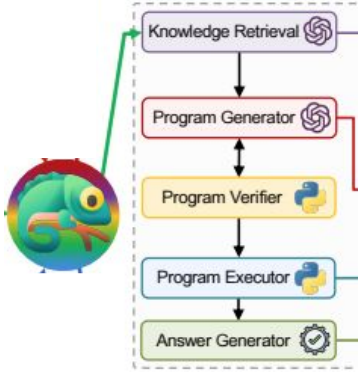
- Grounding
- Reduce hallucinations
- Concept of time

  Cost - architecture and optimization to reduce training and serving costs
- Smaller performant models - personalization? On-device?
- Training full model vs adapt small set of weights
  - LoRA (low rank Adaptation)
  - Adapters
- Inference latency
  - Model distillation, Quantization (to8bit)
  - LoRA has better inference than adapters

Directions

Chameleon, HuggingGPT

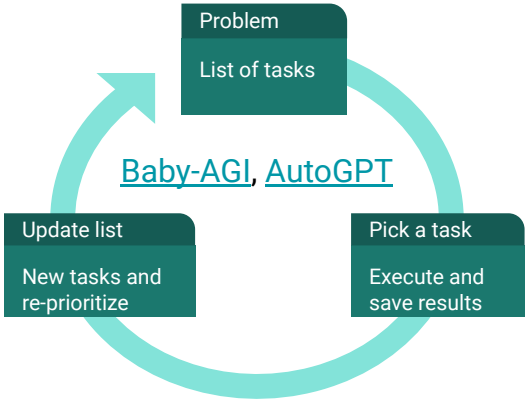
Solve multi-step problems using different tools.



ChemCrow

Accelerating science workflows.

Baby-AGI, AutoGPT

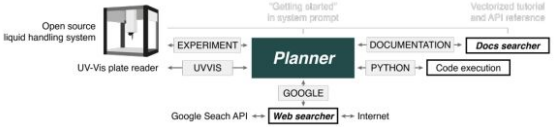


Given a goal in natural language, can attempt to achieve it by breaking it into sub-tasks and using the internet and other tools in an automatic loop

Social Simulacra

Observe, plan, reflect.

Boiko '23



Google

May 26 Update: Watch Karpathy's State of GPT talk at Microsoft Build

* Karpathy's State of GPT talk

Remarks

- Need a common benchmark for evaluation.
- Choice of model can depend on

  Shape of the problem.
  - One-shot generation / phrase completion e.g. code completion
  - Interactive e.g. dialog and instruction tuned
- Amount of data.
  - 1-8 fewshot
  - 10-100s (or 1000) Prompt tune
  - O(1000) Fine tune
- Most problems are not solved with a one-shot query.
  - Problems steering towards O(n) queries to the LLM.

Notes

- This deck
  - text only models
  - If I have missed references or not listed your work please let me know
- Costs
  - #tokens in the input query or context
  - #tokens in the response
  - Time for learning prompts / tuning

References

- Lillian Weng blog
- Lucas’guide
- OpenAI Cookbook
- Langchain
- Prompt Engineering Guide
- Learnprompting.org
- PromptPerfect
- Semantic Kernel