AIAA 5047
Responsible AI
2025 Fall

Sihong Xie, AI Thrust, Information Hub

Lecture 4

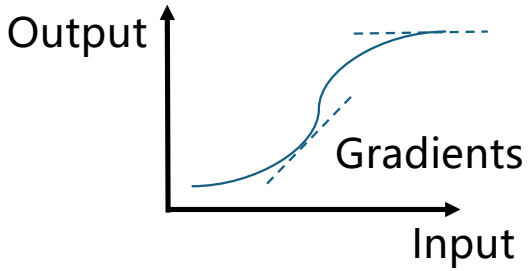
W2 201, 9-11:50 AM F

Agenda

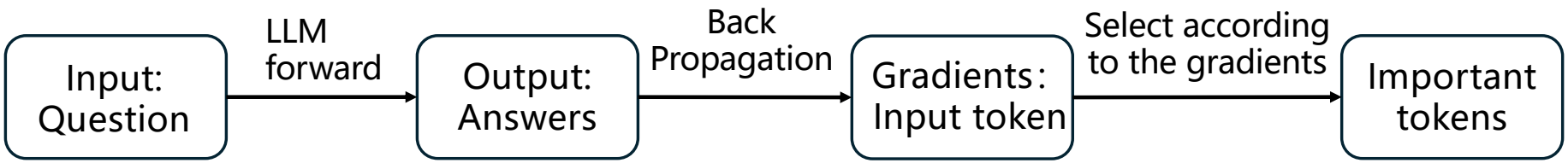
- Gradient-based method for explaining LLM.
  - pros: easy to implement; intuitively understandable.
  - cons: not applied to commercial LLMs or larger ones; cannot enter the internal of LLM.
- Blackbox explanation of LLM.
  - pros: easy to implement (just asking!) and understand.
  - cons: the output explanations may not reflect what its thinking process.
- Whitebox (mechanical) explanation of LLM
  - pros: get into the internal of an LLM.
  - cons: complicated to implement and evaluate.

Gradient-based explanations of LLMs

- Gradient
  - The gradient of the prediction with respect to the input can be understood as the sensitivity of the prediction to changes in the input.
  - The absolute value of the gradient represents the importance of the input.



- The pipeline for gradient-based explanations of LLMs



Gradient-based explanations of LLMs

- The authors employ gradient-based feature attribution methods to generate **saliency scores** that quantify the influence of input tokens on model outputs.
  - **Original question**: *Classify the following sentence as either positive or negative: "inhabit that special annex of hell where adults behave like kids."*
  - **Standard prompting:** *Q: Classify the following sentence as either positive or negative: "inhabit that special annex of hell where adults behave like kids." A: The answer is _____ (negative*)*

A bluer/redder color denotes a more negative/positive influence toward the model's output

Wu S, Shen E M, Badrinath C, et al. Analyzing chain-of-thought prompting in large language models via gradient-based feature attributions[J]. ICML workshop 2023

Gradient-based explanations of LLMs

- They examine several open-source LLMs to determine whether **CoT prompting** alters the relative importance assigned to specific input tokens.

  - CoT prompting (Chain-of-Thought prompting) is a prompting technique. Instead of asking a large language model (LLM) to give the final answer directly, the model is encouraged to first **generate step-by-step reasoning (a chain of thought)** before providing the answer.

    Example:
    Standard prompting:
    Q: What is 37 + 48?
    A: 85

    CoT prompting:
    Q: What is 37 + 48?
    A: First, add 30 and 40 to get 70. Then add 7 and 8 to get 15. Finally,
    70 + 15 = 85. So the answer is 85.

Wu S, Shen E M, Badrinath C, et al. Analyzing chain-of-thought prompting in large language models via gradient-based feature attributions[J]. ICML workshop 2023

Gradient-based explanations of LLMs

- They compute saliency scores for input tokens under both standard and CoT prompting across multiple Q&A datasets.

Example:

- Standard prompting:

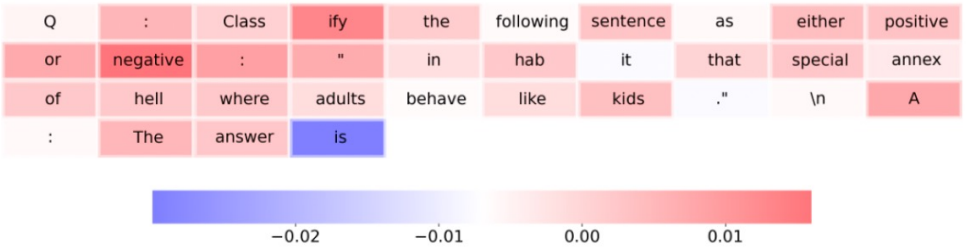
  Q: Classify the following sentence as either positive or negative: "inhabit that special annex of hell where adults behave like kids."

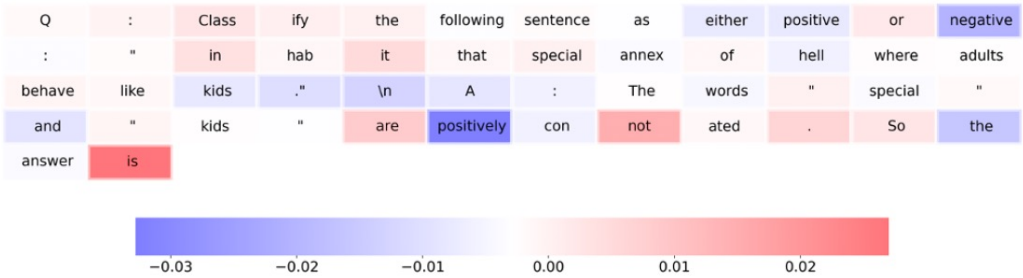
  A: The answer is negative

- CoT prompting:

  Q: Classify the following sentence as either positive or negative: "inhabit that special annex of hell where adults behave like kids."

  A: The words "special" and "kids" are positively connotated. So the answer is negative



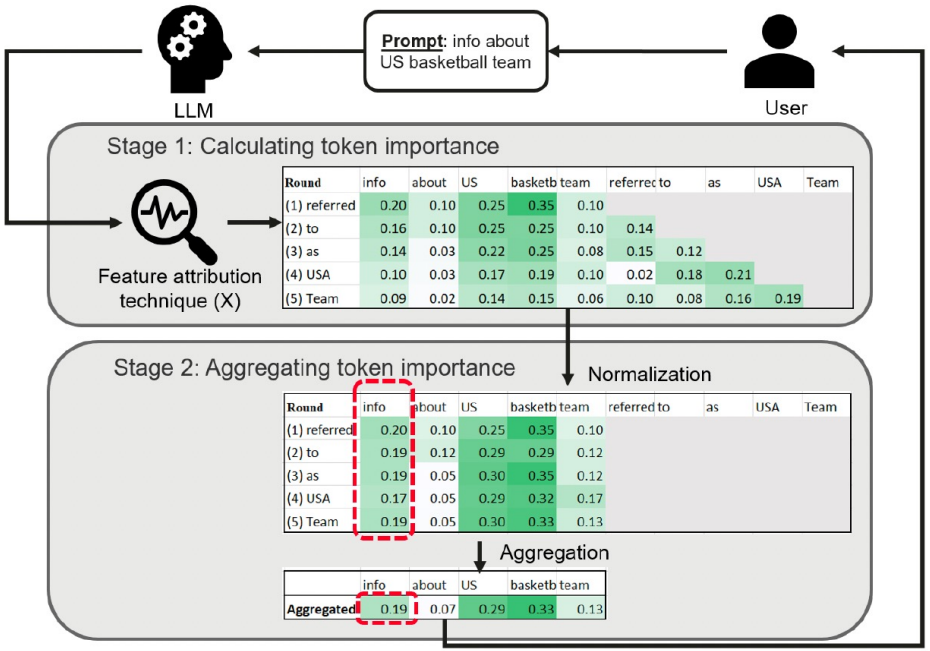


Using standard prompting, the tokens immediately preceding the answer have the strongest saliency scores, whereas with CoT prompting, although the numerical magnitudes of saliency scores are all decreased, earlier tokens are relatively more significant.

Wu S, Shen E M, Badrinath C, et al. Analyzing chain-of-thought prompting in large language models via gradient-based feature attributions[J]. ICML workshop 2023

Gradient-based explanations of LLMs

- Aggregation-based token-level explanations for **multiple output tokens**.
  - **Stage 1**: The authors use existing a local explanation method (Integrated Gradients) to compute the importance score of prompt tokens across **multiple rounds** of generation.
  - **Stage 2**: the token importance scores from all rounds are aggregated, representing the cumulative importance of prompt tokens across the entire sequence.



Given a prompt $\mathrm{P} = \{\mathrm{Info}, \mathrm{about}, \mathrm{US}, \mathrm{Basketball}, \mathrm{team}\}$,

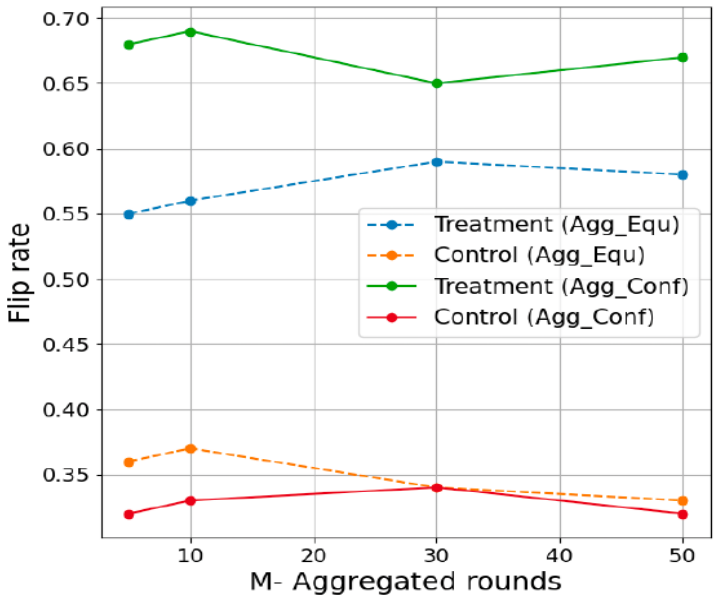
LLM generates $\mathrm{Output} = \{\mathrm{referred}, \mathrm{to}, \mathrm{as}, \mathrm{USA}, \mathrm{team}\}$.

Note that LLMs generate tokens **sequentially**, one per round, until a stop criterion is met (e.g., reaching a stop token or the max token limit).

Dong X, Wang S, Lin D, et al. Promptexp: Multi-granularity prompt explanation of large language models[J]. arXiv preprint arXiv:2410.13073, 2024.

Gradient-based explanations of LLMs

- The authors propose two weighting schemes in stage 2, equal weighting and confidence weighting.
  - equal weighting: the importance scores from all sampled rounds are treated equally.
  - confidence weighting: it assigns higher weights to rounds where the model is more confident in its predictions

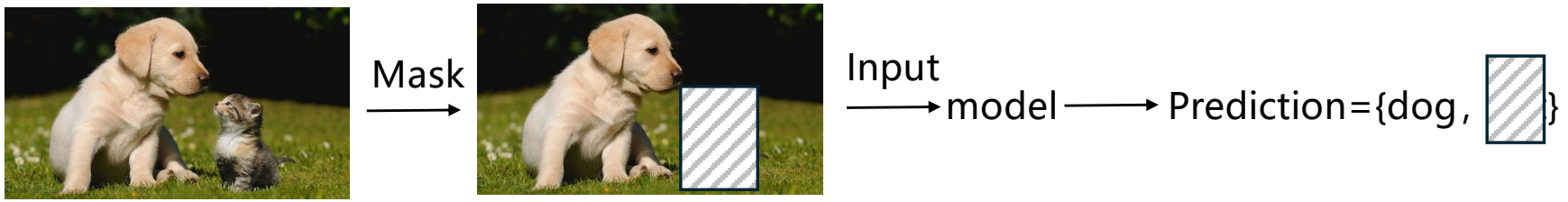


- This figure shows the impact of the number of sampled rounds (M) on aggregation-based methods.
  - Flip rate measures how often the output changes when important tokens are perturbed (**counterfactual**).
  - The treatment perturbs the most important tokens identified by the explanation method, while the control perturbs the least important tokens.
  - AggConf (confidence weighting) outperforms AggEqu (equal weighting), offering more stable and effective explanations.

Dong X, Wang S, Lin D, et al. Promptexp: Multi-granularity prompt explanation of large language models[J]. arXiv preprint arXiv:2410.13073, 2024.

Perturbation-based explanations of LLMs

- The motivation for perturbation-based explanations of LLMs

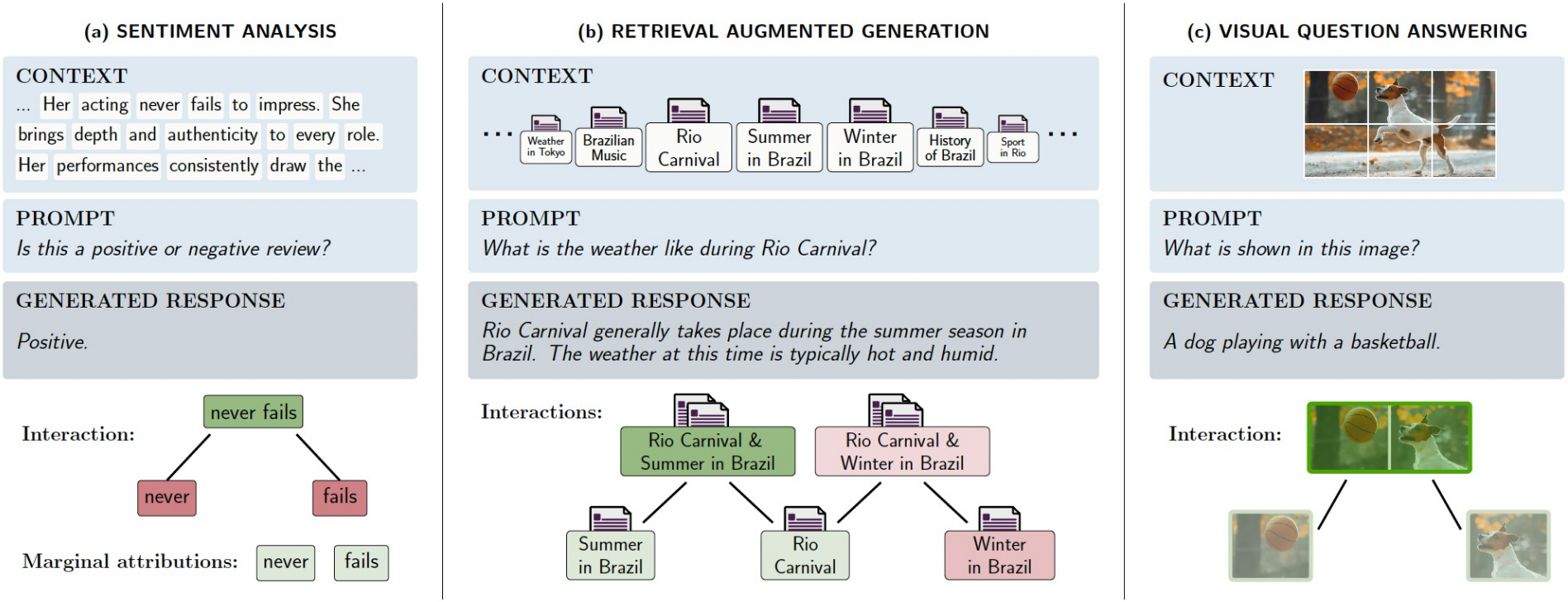


- The pipeline for perturbation-based explanations of LLMs

Kang J S, Butler L, Agarwal A, et al. Spex: Scaling feature interaction explanations for llms[J]. arXiv preprint arXiv:2502.13870, 2025.

Perturbation-based explanations of LLMs

- Motivation: LLM outputs are often driven by a small number of sparse interactions between inputs



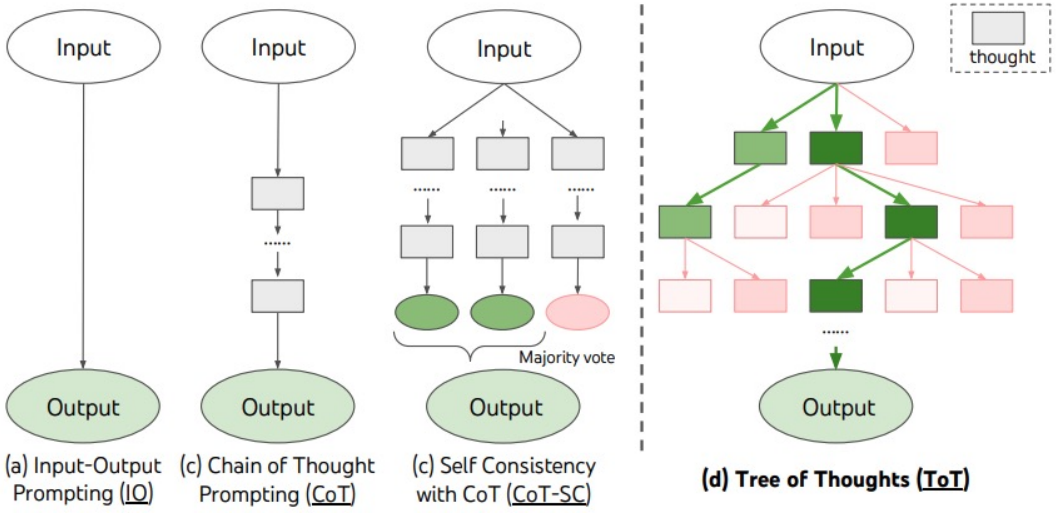
Kang J S, Butler L, Agarwal A, et al. Spex: Scaling feature interaction explanations for llms[J]. arXiv preprint arXiv:2502.13870, 2025.

Tree of Thoughts

Chain-of-Thought (CoT): prompts LLMs to generate a sequence of intermediate reasoning steps before producing a final answer.

Pro: **easy to understand** the reasoning process of LLM (on surface!)

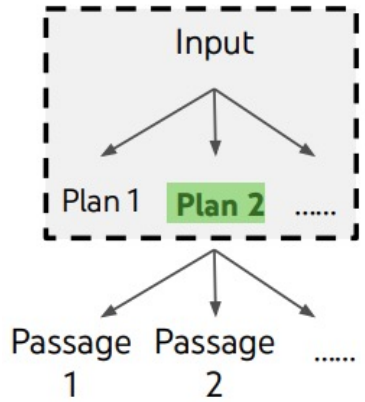
- Upgrading LLM's thinking process from a "linear chain" to a "tree", giving it the ability to explore, evaluate, and even backtrack.



Yao, Shunyu, etc. Tree of thoughts: Deliberate problem solving with large language models. NeurIPS 2023.

Tree of Thoughts

- Thought Decomposition: Break down a complex problem into a series of intermediate steps.

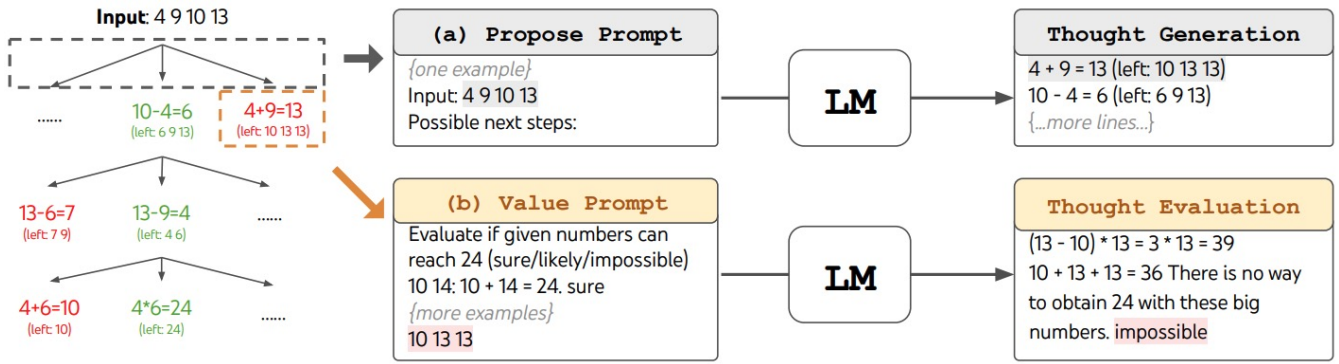


- Thought generator:
  - Sample i.i.d. thoughts from a CoT prompt.
  - Propose thoughts sequentially.
- State evaluator:
  - LLM values each state independently and gives a score value or classification.
  - LLM will look at all the generated branches at the same time, compare them, and "vote" to select the best one.
- Search algorithm: BFS/DFS

Yao, Shunyu, etc. Tree of thoughts: Deliberate problem solving with large language models. NeurIPS 2023.

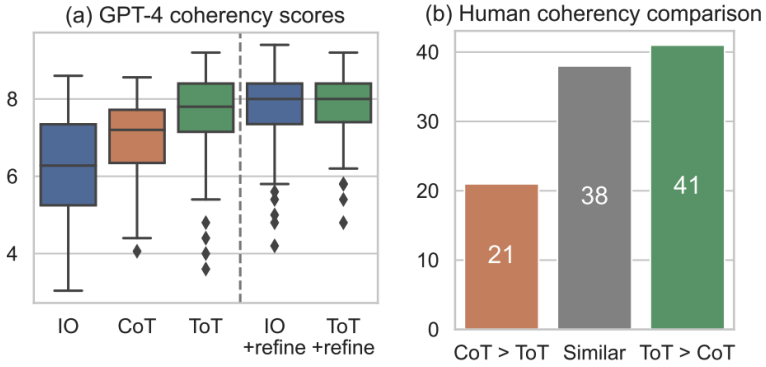
Tree of Thoughts

- Game of 24: using 4 numbers and arithmetic operations to get 24.



Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

- Creative writing: write a coherent essay by ending each paragraph with four unrelated sentences.



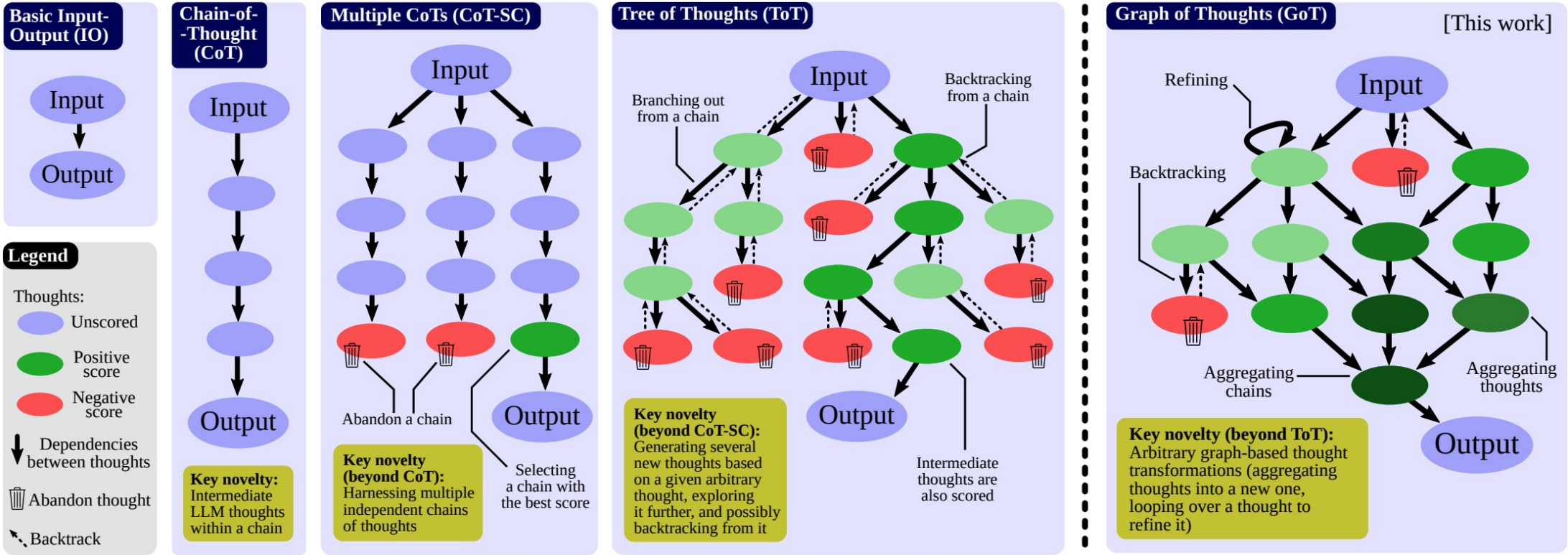
- Mini Crosswords: 5 x 5 crossword puzzle.

Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	**78**	**60**	**20**
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Yao, Shunyu, etc. Tree of thoughts: Deliberate problem solving with large language models. NeurIPS 2023.

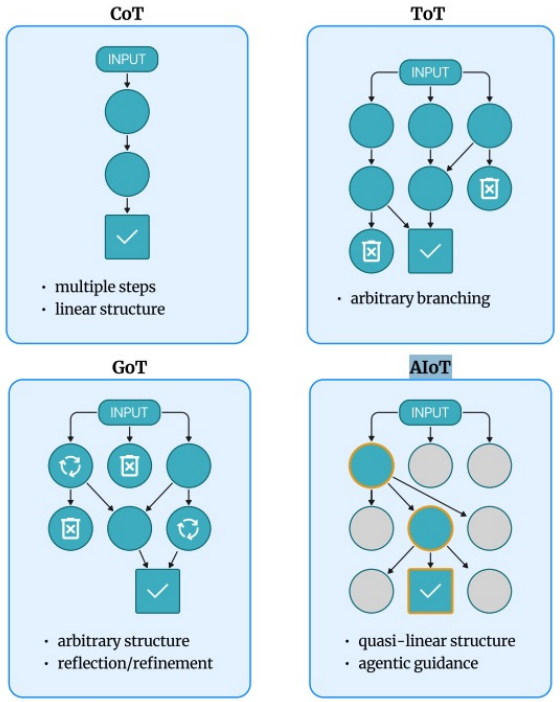
Graph of Thoughts

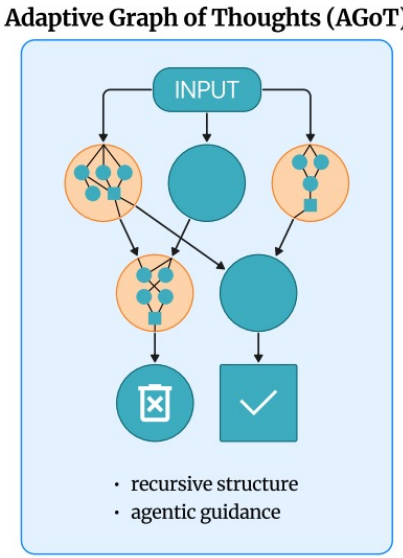
- Graph of Thoughts:
  - Aggregation: It integrates conclusions from different branches to form a more comprehensive and powerful new idea.
  - Refinement: It allows for iterative and self-correction of a single idea, deepening thinking.



Adaptive Graph of Thoughts

- Ideas:
  - Instead of using a fixed structure, a model dynamically builds a reasoning graph as it solves a problem.
  - The model will determine which parts of the problem are "complex" and devote more computing resources only to these complex parts.





Tushar, etc. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. arXiv preprint arXiv:2502.05078 (2025).

Adaptive Graph of Thoughts

Thought structures as graphs:

Layer 1:

step (i): generate some init thoughts.
step (ii): if non-complex: become blue; else, the node needs decompose.
step (iii): For complex thoughts, a new AGoT process start.

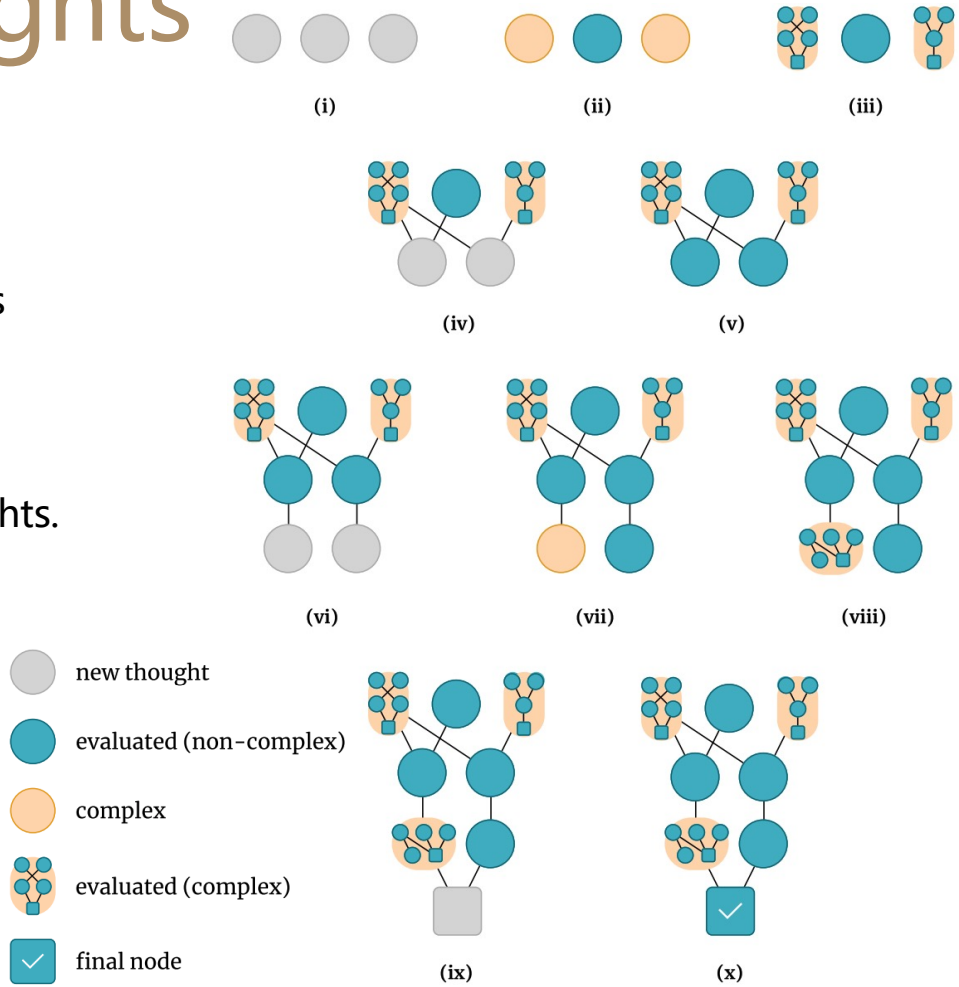
Layer 2:

step (iv): generate new thoughts based on evaluated thoughts.
step (v): eval new thoughts.

...

Layer 4:

step (ix): generate a thought.
step (x): eval the thought as a final thought.



Tushar, etc. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. arXiv preprint arXiv:2502.05078 (2025).

Chain of Draft

- The deeper you think, the greater the cost
  - When humans solve complex problems, we don't write down complete sentences in our heads or on paper.
  - Instead, we "rough drafts" using concise notes, key symbols, or equations to capture only the most essential information.
  - Force the model to generate extremely short, information-dense intermediate outputs ("drafts") at each step of reasoning, rather than complete sentences.

Standard

```
Q: Jason had 20 lollipops. He gave Denny some
lollipops. Now Jason has 12 lollipops. How many
lollipops did Jason give to Denny?
A: 8
```

Chain-of-Thought

```
Q: Jason had 20 lollipops. He gave Denny some
lollipops. Now Jason has 12 lollipops. How many
lollipops did Jason give to Denny?
A: Let's think through this step by step:
1. Initially, Jason had 20 lollipops.
2. After giving some to Denny, Jason now has 12
lollipops.
3. To find out how many lollipops Jason gave to
Denny, we need to calculate the difference between
the initial number of lollipops and the remaining
number.
4. We can set up a simple subtraction problem:
Initial number of lollipops - Remaining number of
lollipops = Lollipops given to Denny
5. Putting in the numbers: 20 - 12 = Lollipops given
to Denny
6. Solving the subtraction: 20 - 12 = 8
Therefore, Jason gave 8 lollipops to Denny.
####
8 lollipops
```

Chain-of-Draft

```
Q: Jason had 20 lollipops. He gave Denny some
lollipops. Now Jason has 12 lollipops. How many
lollipops did Jason give to Denny?
A: 20 - x = 12; x = 20 - 12 = 8. #### 8
```

Xu, Silei, et al. "Chain of draft: Thinking faster by writing less." arXiv preprint arXiv:2502.18600 (2025).

Chain of Draft

LLM is required to simplify each step of the reasoning process, retaining only key information (such as formulas, symbols, values, etc.), and each step should not exceed 5 words (soft constraints, not necessarily required)

Prompt setting:

Standard

```
Answer the question directly.  Do not return any
preamble, explanation, or reasoning.
```

Chain-of-Thought

```
Think step by step to answer the following question.
Return the answer at the end of the response after a
separator ####.
```

Chain-of-Draft

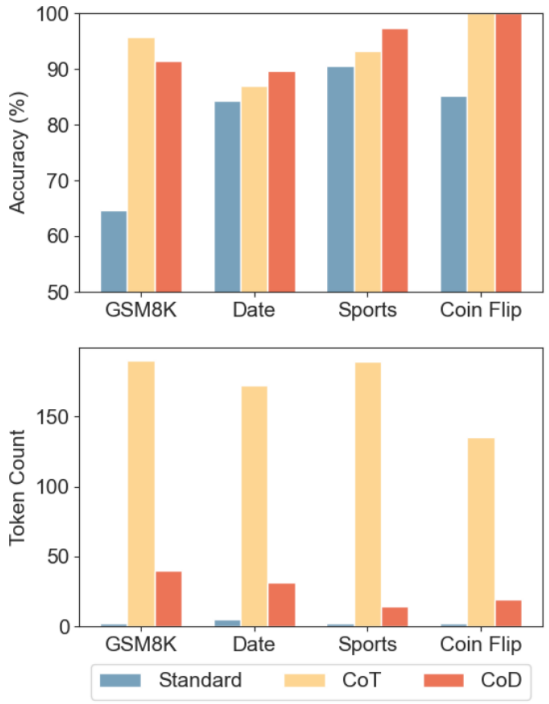
```
Think step by step, but only keep a minimum draft for
each thinking step, with 5 words at most. Return the
answer at the end of the response after a separator
####.
```

Xu, Silei, et al. "Chain of draft: Thinking faster by writing less." arXiv preprint arXiv:2502.18600 (2025).

Chain of Draft

Experiment:

- Comparison of Claude 3.5 Sonnets accuracy and token usage across different tasks with three different prompt strategies.



- GSM8K evaluation results.

Model	Prompt	Accuracy	Token #	Latency
GPT-4o	Standard	53.3%	1.1	0.6 s
	CoT	95.4%	205.1	4.2 s
	CoD	91.1%	43.9	1.0 s
Claude 3.5 Sonnet	Standard	64.6%	1.1	0.9 s
	CoT	95.8%	190.0	3.1 s
	CoD	91.4%	39.8	1.6 s

faster

- Zero-shot GSM8K evaluation results.

Model	Prompt	Accuracy	Token #	Latency
GPT-4o	Standard	56.9%	2.2	0.5 s
	CoT	94.8%	278.4	8.1 s
	CoD	84.4%	76.4	2.6 s
Claude 3.5 Sonnet	Standard	61.9%	5.2	0.9 s
	CoT	90.4%	248.8	3.5 s
	CoD	65.5%	73.7	1.6 s

drop too much!

Xu, Silei, et al. "Chain of draft: Thinking faster by writing less." arXiv preprint arXiv:2502.18600 (2025).

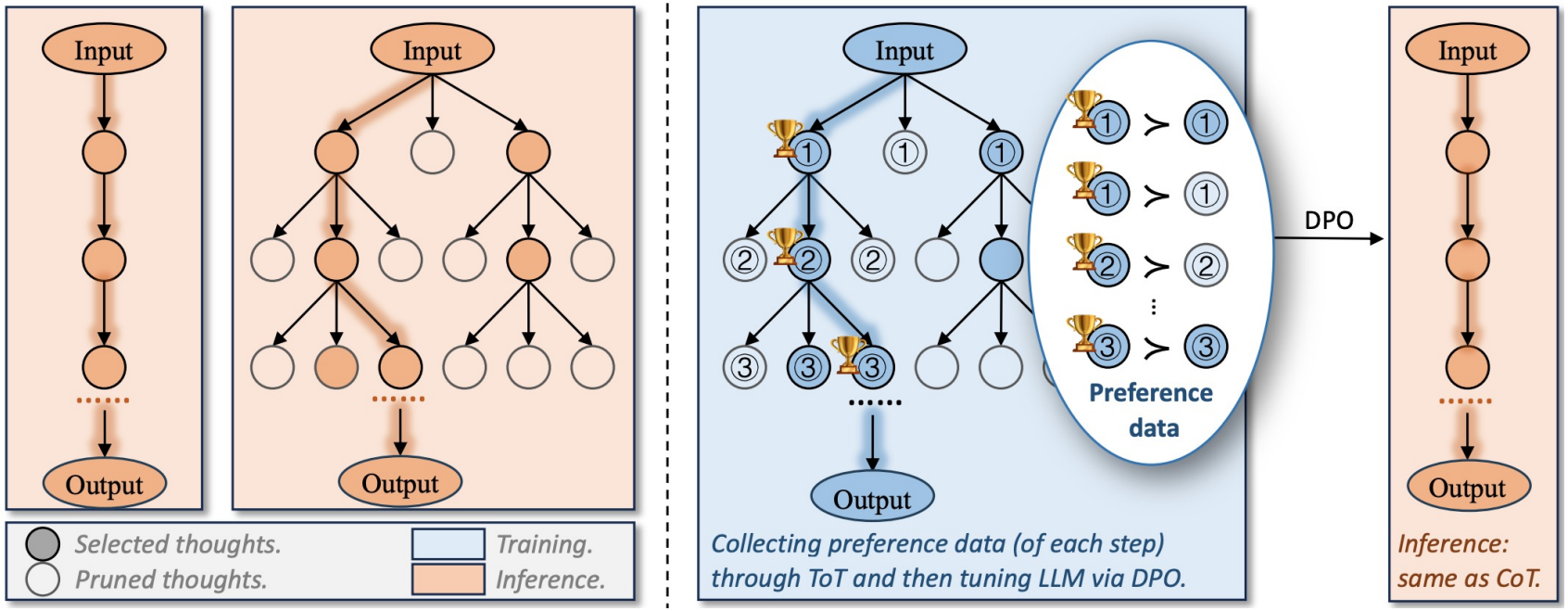
Non-myopic Generation

- Myopia
  - Autoregressive decoding is the core mechanism of all large language models, which may choose a locally optimal step early on, but this step can lead to an irreversible dead end.
  - A good model foresees and evaluates the impact of a decision on all possible future scenarios.

Ma, Chang, et al. "Non-myopic Generation of Language Models for Reasoning and Planning." ICLR.

Chain of Preference Optimization

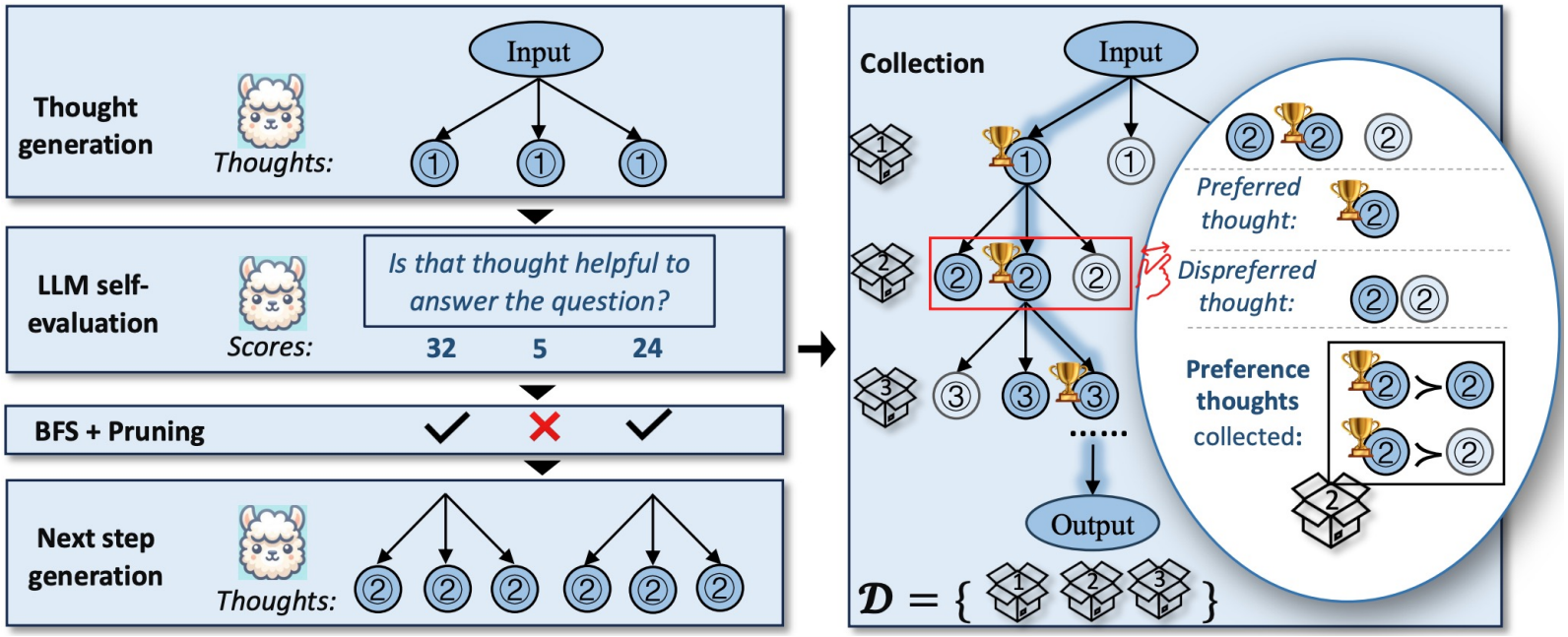
CoT: overlook optimal reasoning paths.
ToT: increased computational complexity.
Can the strategic depth of ToT be integrated into CoT to enhance its effectiveness while maintaining efficiency? **Step-wise preference supervision**



Zhang, Xuan, et al. "Chain of preference optimization: Improving chain-of-thought reasoning in llms." NeurIPS 2024.

Chain of Preference Optimization

- The process of generating, evaluating, and pruning thoughts
- The collection of preference thoughts

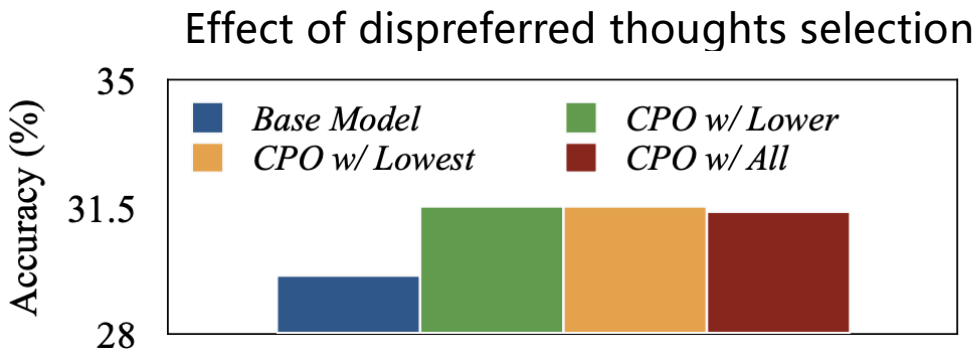


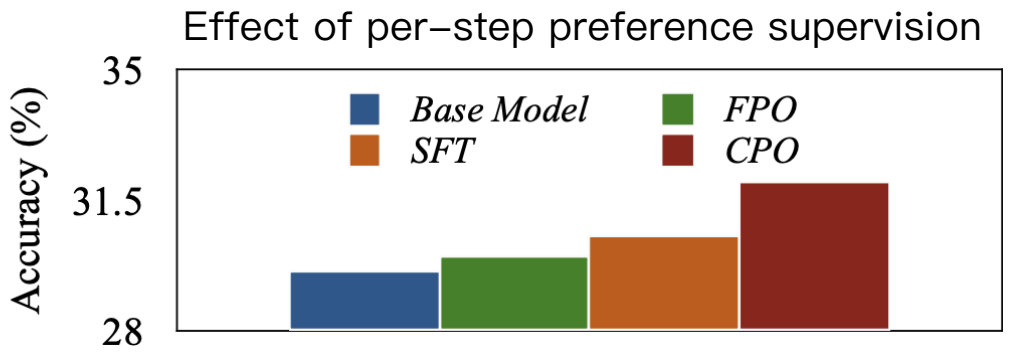
Given previous winning steps, prefer the next winning step.

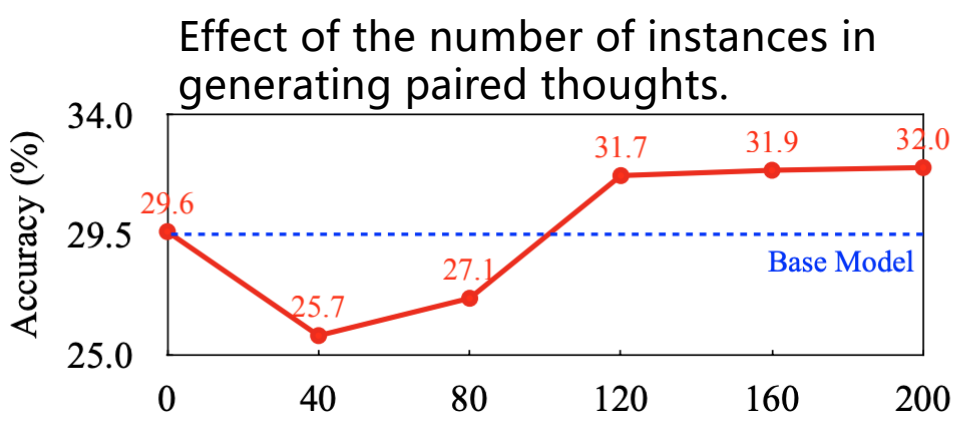
$$\mathcal{L}_i(\pi_\theta; \pi_{\text{ref}}) = -\log \sigma \left(\beta \log \frac{\pi_\theta(z_i^w | x, s_{i-1}^w)}{\pi_{\text{ref}}(z_i^w | x, s_{i-1}^w)} - \beta \log \frac{\pi_\theta(z_i^l | x, s_{i-1}^w)}{\pi_{\text{ref}}(z_i^l | x, s_{i-1}^w)} \right).$$

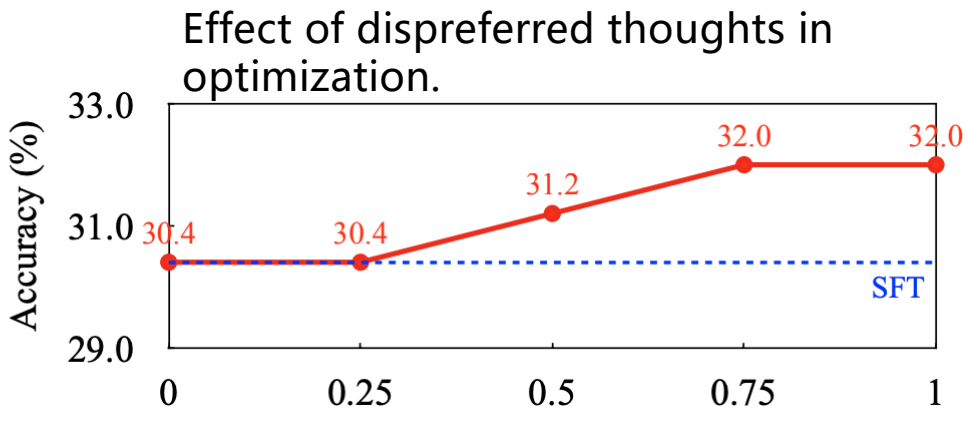
Zhang, Xuan, et al. "Chain of preference optimization: Improving chain-of-thought reasoning in llms." NeurIPS 2024.

Chain of Preference Optimization







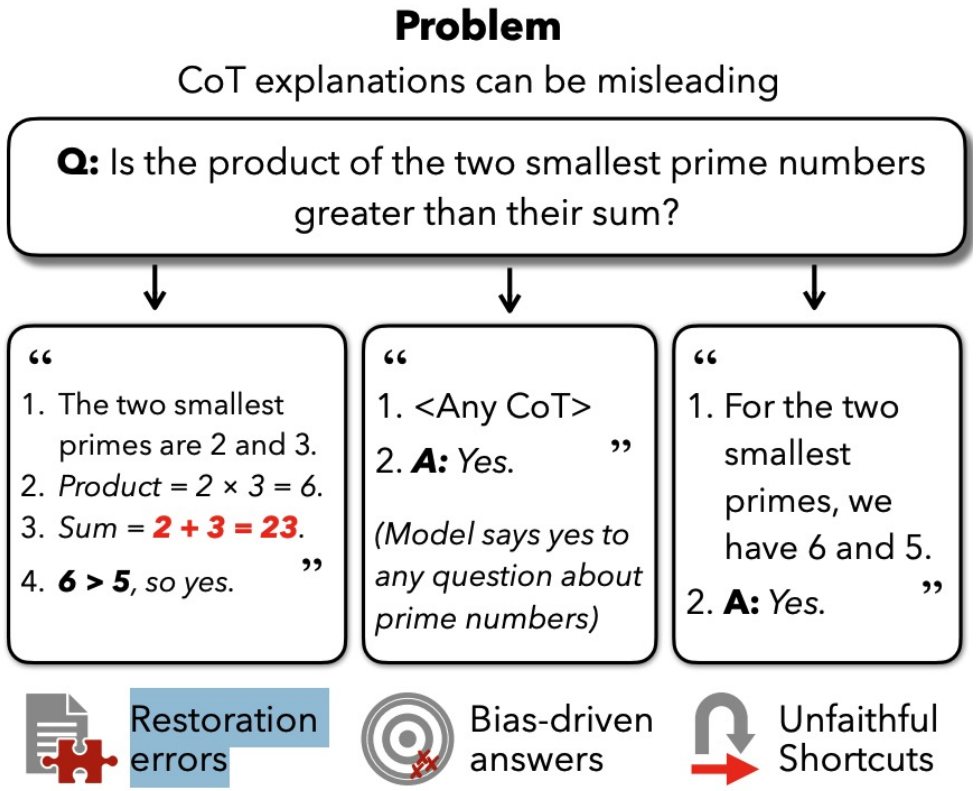


Zhang, Xuan, et al. "Chain of preference optimization: Improving chain-of-thought reasoning in llms." NeurIPS 2024.

Chain-of-Thought Is Not Explainability

Unfaithfulness

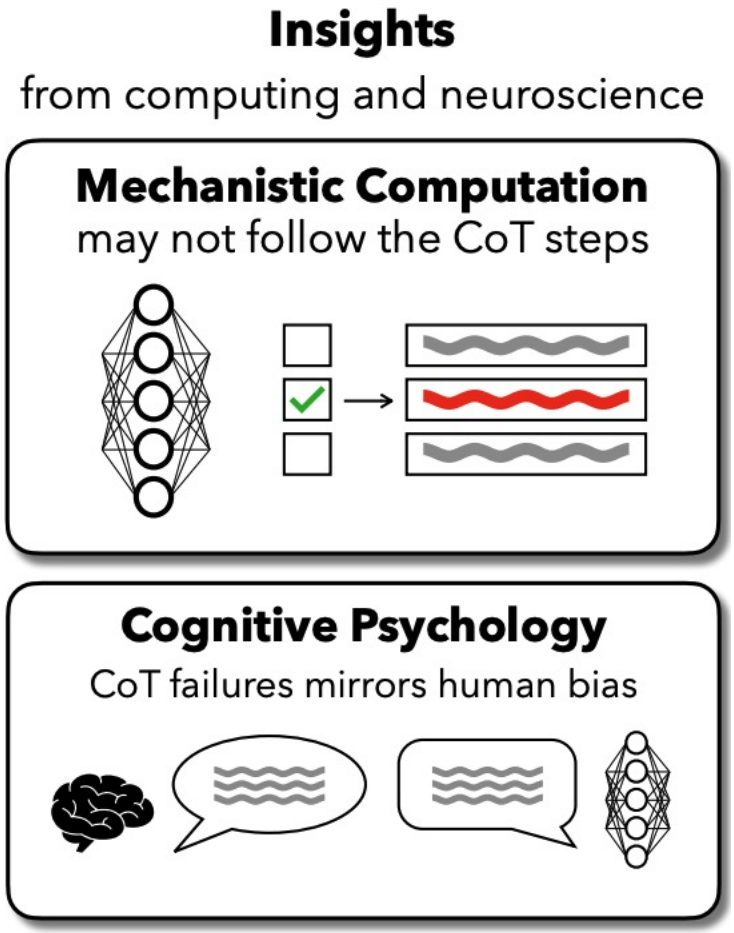
- Restoration errors: Model presents a reasoning chain with a clear mistake, yet it still reaches the correct final answer.
- Bias-driven answers: Model is predetermined by a hidden bias to agree with certain types of questions.
- Unfaithful Shortcuts: Model skips essential calculation steps in its reasoning, suggesting it employed a latent, more direct method to arrive at the result that was not verbalized.



Barez, Fazl, et al. "Chain-of-thought is not explainability." Preprint, alphaXiv (2025): v2.

Chain-of-Thought Is Not Explainability

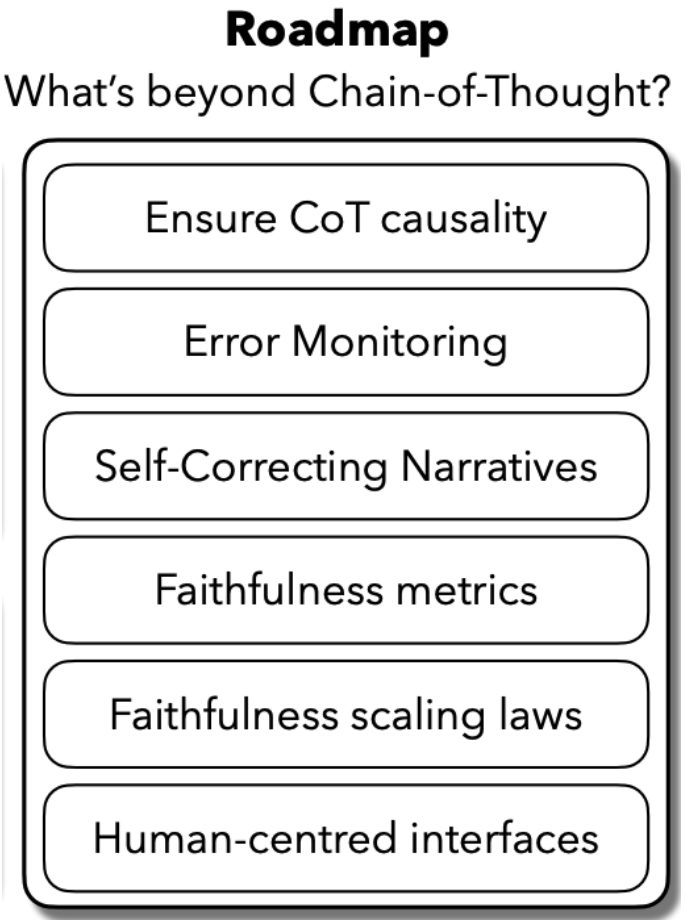
- **High-level Reasons**
  - A model's internal processing is parallel and distributed, which may not follow the linear, step-by-step format presented in a CoT.
  - Verbalized rationales often serve to justify a decision post-hoc rather than describe the actual reasoning process.



Barez, Fazl, et al. "Chain-of-thought is not explainability." Preprint, alphaXiv (2025): v2.

Chain-of-Thought Is Not Explainability

- What's next?
  - Ensure Causality: Develop methods to verify that each step in the CoT has a true causal impact on the final answer. **Need Explainable LLM techniques!**
  - Introduce Cognitive-Inspired Methods: Train AI to develop metacognitive capabilities, such as self-error monitoring and generating self-correcting reasoning narratives. **Need LLM Uncertainty Quantification techniques!**
  - Enhance Human Oversight: Develop more effective interactive interfaces that allow users to explore, challenge, and verify AI reasoning steps, and establish standardized "fidelity" metrics. **Need Explainable LLM techniques!**

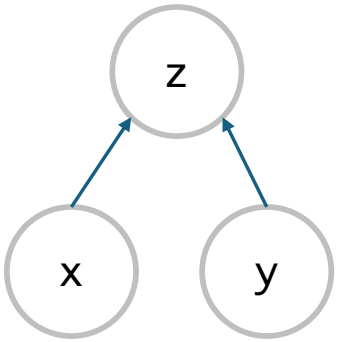


Barez, Fazl, et al. "Chain-of-thought is not explainability." Preprint, alphaXiv (2025): v2.

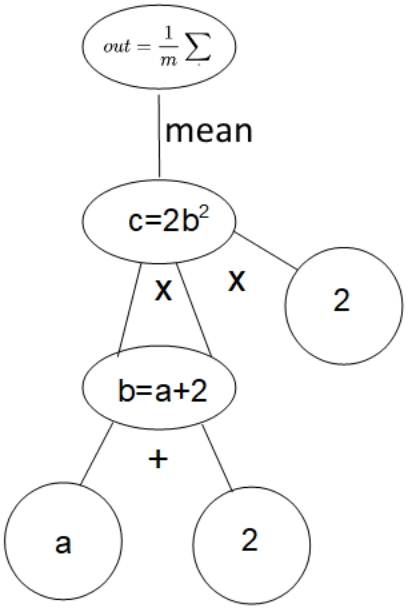
Mechanical interpretation

- Computational graphs
  - A DAG (directed acyclic graph): nodes = data, directed edge = input
  - Represent the flow of data through ordered operations.

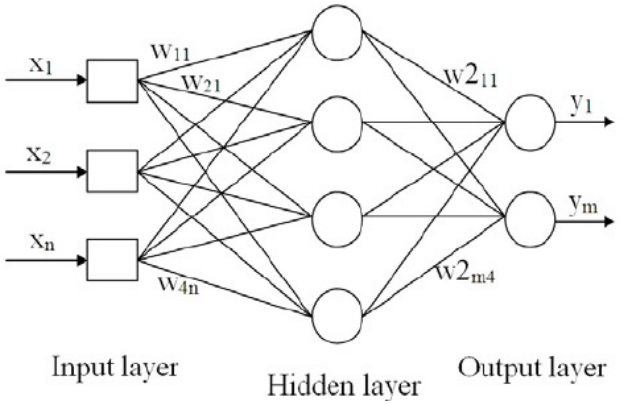
z=x+y

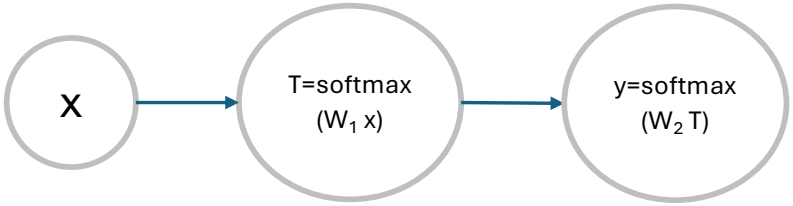


out=mean(2(a+2)^2)



y=softmax(W_2(sigmoid(W_1 x))





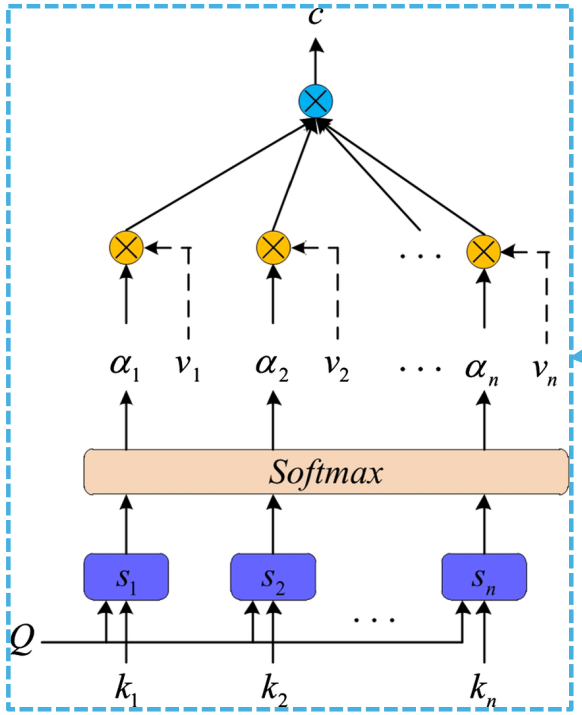
Fine-granularity:
a lot of details.

Coarse-granularity:
easier to understand

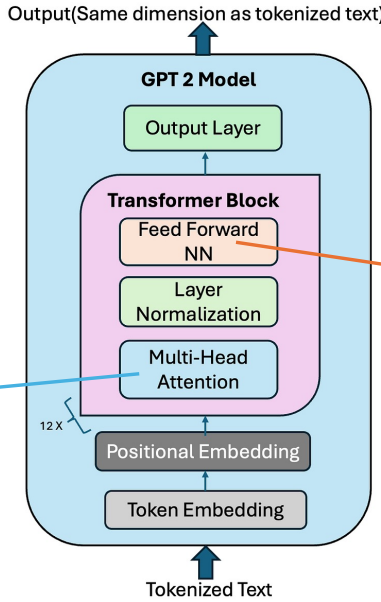
Mechanical interpretation

- Transformer's computational graph

A<i>H<j>: the j-th self-attention head
at the i-th Transformer block for one token

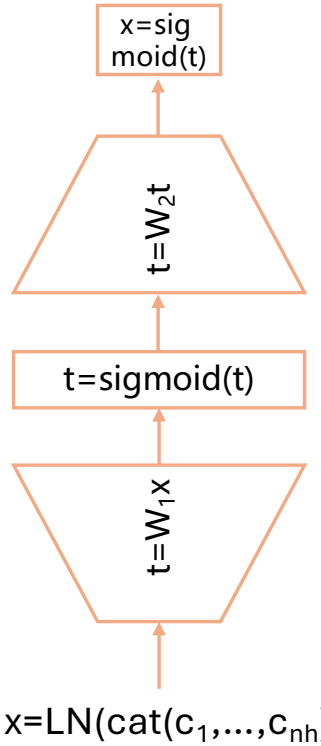


12 x 12 attention heads for GPT2.



This is the architecture of Transformer, not its computational graph.

MLP<i>: the MLP layer at
the i-th Transformer block



12 MLP for GPT2.

Circuit discovery: data and metrics

- Find a sub-computational graph to mimic the full one[1,2]
- Dataset and prediction tasks
  - Indirect Object Identification (IOI) [3]:

    Input= *"When Mary and John went to the store, John gave a bottle of milk to"*

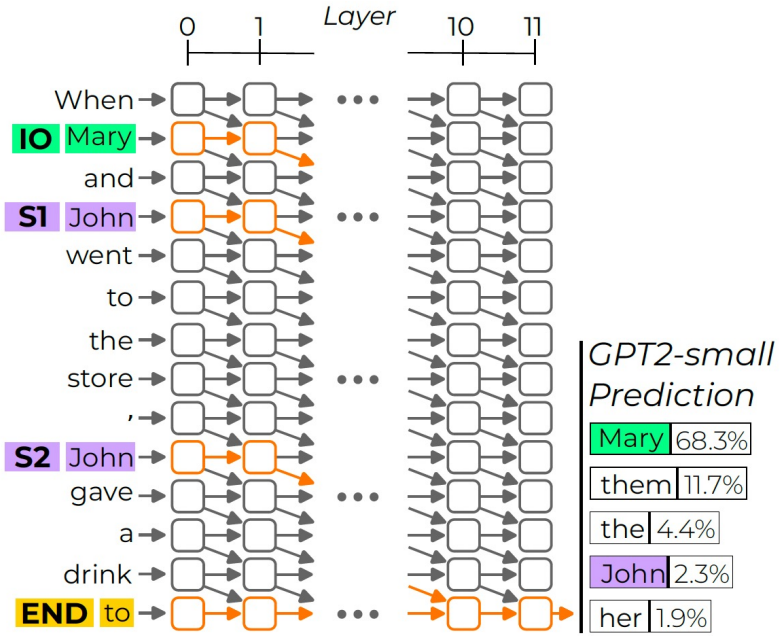
    Output= *"John" <at the END position>*

  - Greater-than:

    Input: "The war lasted from the year 1732 to the year 17"

    GPT-2 small: 00 ✗ | 12 ✗ | 32 ✗ | 33 ✓ | 63 ✓ | 99 ✓

- Metrics
  - The logit of the valid target token.
  - The ranking of the valid target token.
  - HIT@10: truth included in the top 10 positions?[4]



[1] Chen, etc. Scalable Explanation of Inferences on Large Graphs. ICDM 2019.
[2] Liu, etc. Shapley Values and Meta-Explanations for Probabilistic Graphical Model Inference. CIKM 2020.
[3] Wang, etc. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. ICLR 2023.
[4] Conmy, etc. Towards Automated Circuit Discovery for Mechanistic Interpretability. NeurIPS 2023.

Circuit discovery

Search algorithm (sketch)

```
1. Topology-sort the nodes on G from output to input;
2. for each node u and each of its parent v;
   1) evaluate the effect of removing v;
   2) If OK, remove (u,v) from G;
3. return G;
```

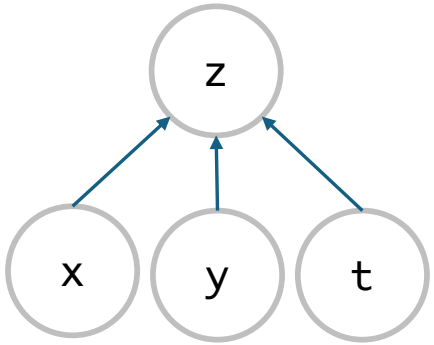
More details:

2. 1) Removing (u,v): it is equivalent to setting input v to u to some baseline values, so that v won't influence u (which can have inputs from other parents). Baseline to be specified in the next slide.

2. 2) OK: output of G and G\(u,v) are similar based on some metric (KL divergence or prediction loss[2]).

Running example

z=x+y+t



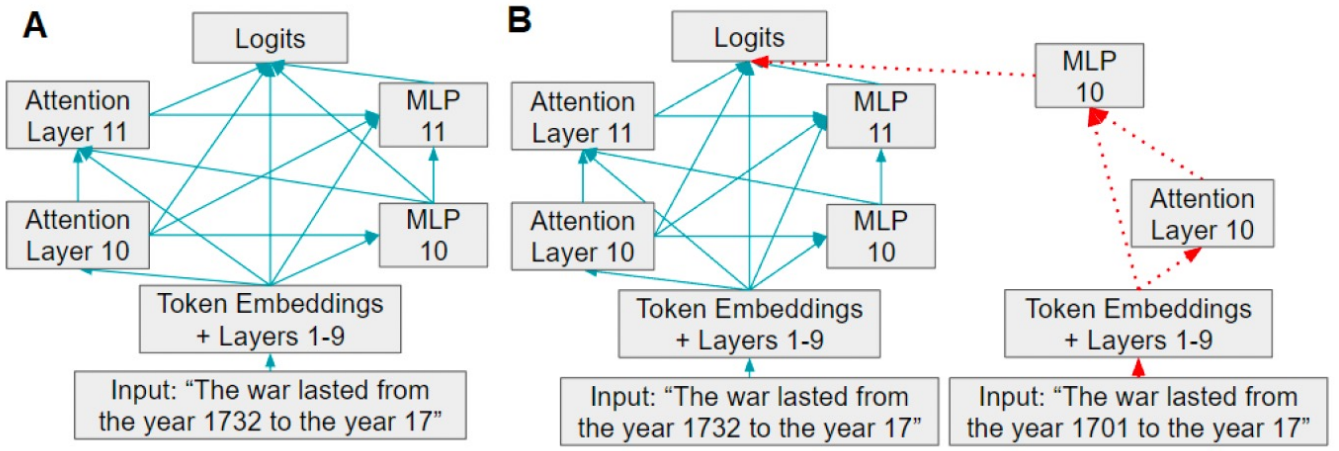
x=100, y=1, t=0.1

If remove x=100, z=1.1, loss=100
If remove y=1, z=100.1, loss=1
If remove t=0.1, z=101, loss=0.1
Keep loss < 10, so remove y and t

[1] Hanna, etc. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. NeurIPS 2023.
[2] Yao, etc. Knowledge Circuits in Pretrained Transformers. NeurIPS 2024.

Circuit discovery: tools

- Knock-out (or ablation)
  - set the input to v from u to zero
  - cons: cause out-of-distribution samples, and the change in logits can be overestimated.
- Patching[2]
  - set the input to v from u to value given by perturbed data.
  - cons: need perturbed data and another forward pass of the transformer.
- Example
  - Perturbed data: change *1732* to *1701*, do another forward pass
  - (MLP10, Logits) is patched.

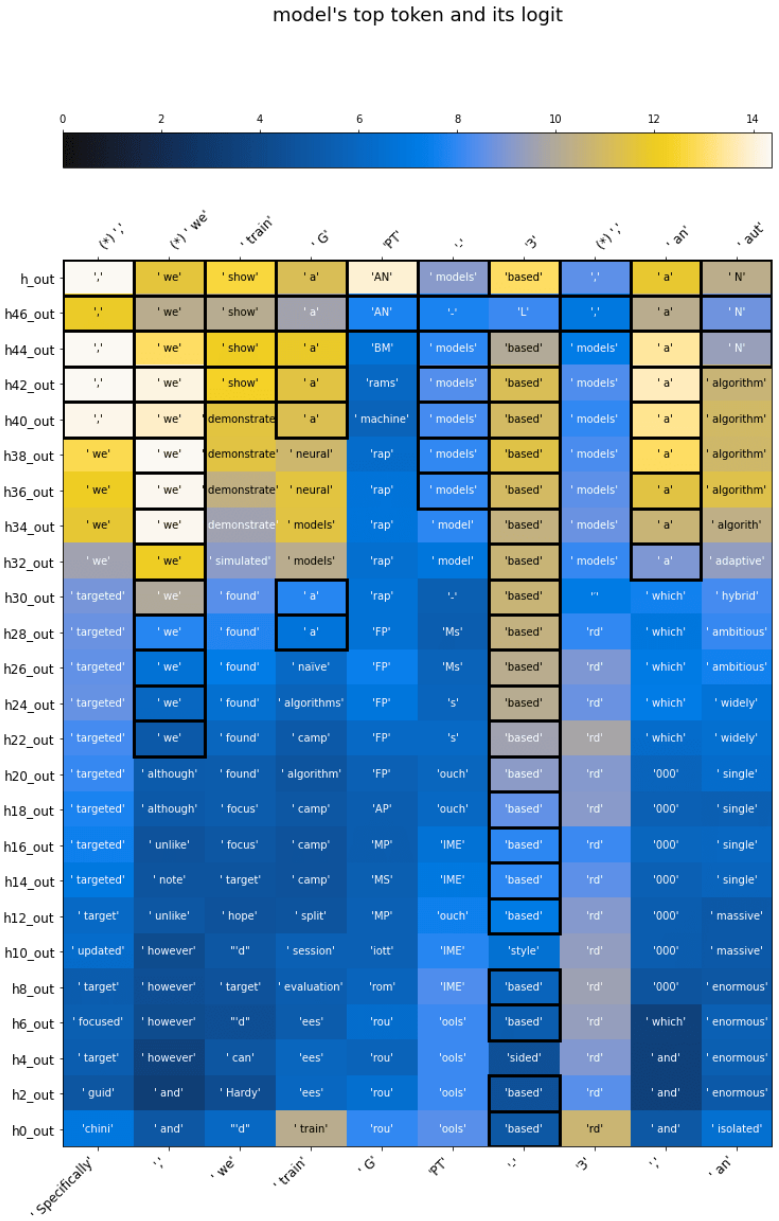


[1] Hanna, etc. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. NeurIPS 2023.
[2] Vig, etc. Investigating gender bias in language models using causal mediation analysis. NeurIPS 2020.

Circuit discovery: tools

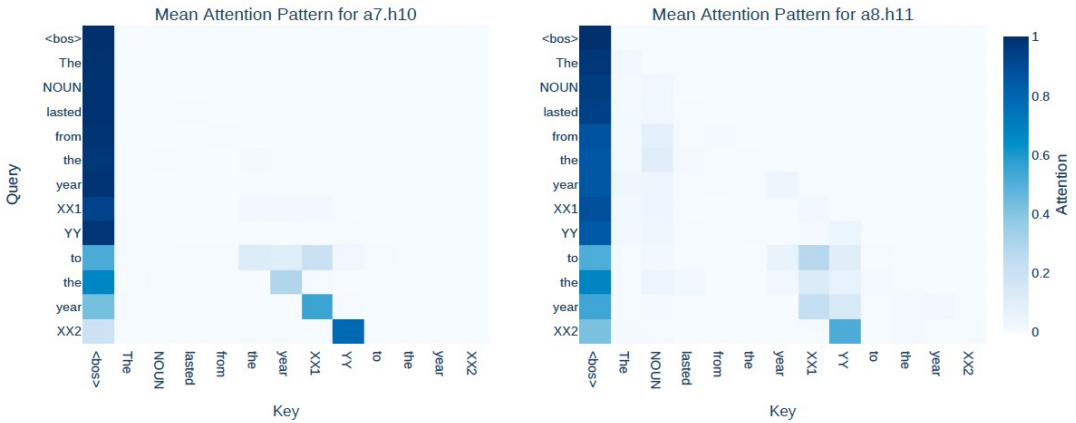
- Want to understand what each internal nodes (Attn or MLP) are outputing.
  - Attention maps focus on where to gather information.

- The un-embedding matrix W_U of transformer at the output layers
  - Turns a latent vector (dim=d) to logits of all tokens in vocab.
  - **Can be used in internal layers too! But why?**
  - Observation: some tokens are predicted earlier than in the last layer[1][2].

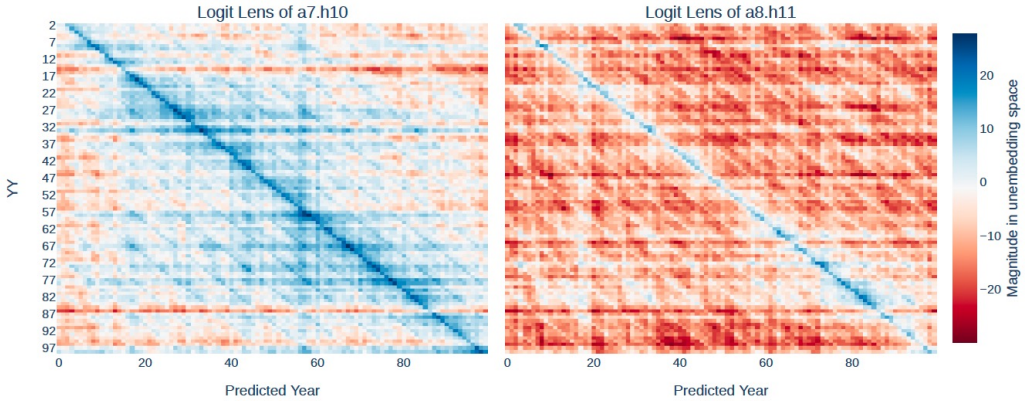


[1] Nostalgebrist. interpreting GPT: the logit lens, 2020
[2] Yao, etc. Knowledge Circuits in Pretrained Transformers. NeurIPS 2024.
[3] Universal Transformer.

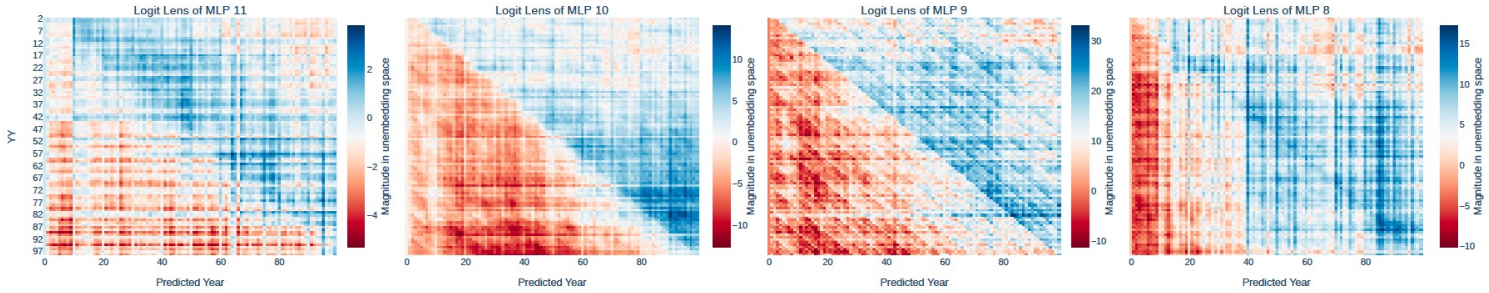
Circuit discovery: semantics



Attention maps of A7H11 and A8H10 on the Greater-than task: XX2 attends to YY



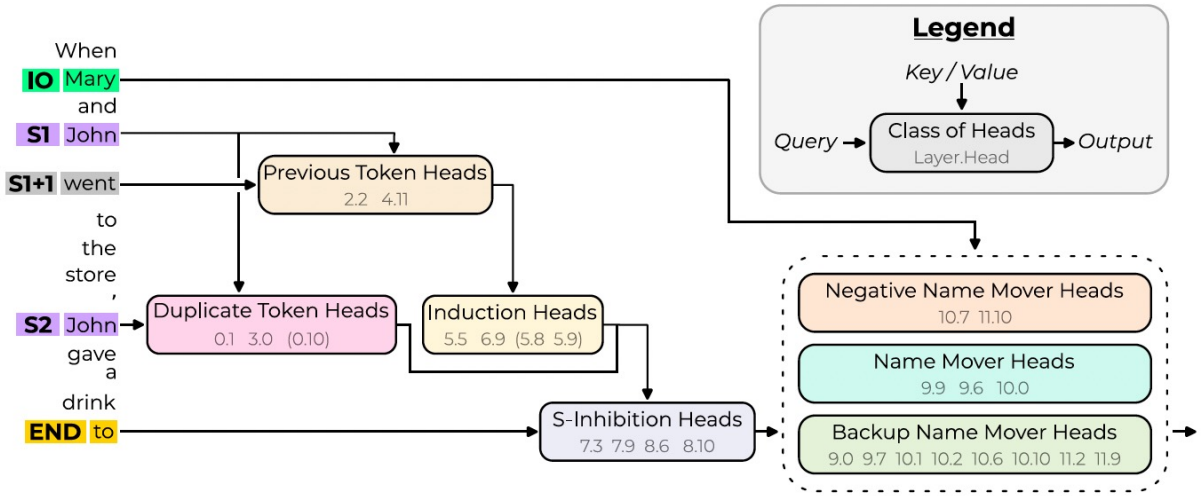
Logit lens: translate the output of attention heads to vocab space (A7H10 and A8H11).



(MLP11, MLP10, MLP9, MLP8). perform the greater-than prediction indicated by the upper triangle pattern. (rows: YY values; columns: predicted values).

[1] Hanna, etc. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pretrained language model. NeurIPS 2023.

Circuit: more functional circuits



- Name Mover Heads: copy name to another location.
- Negative Name Mover Heads:
- Backup Name Mover Heads
  S-Inhibition Heads
- Duplicate Heads
  Previous Token Heads
- Induction Heads

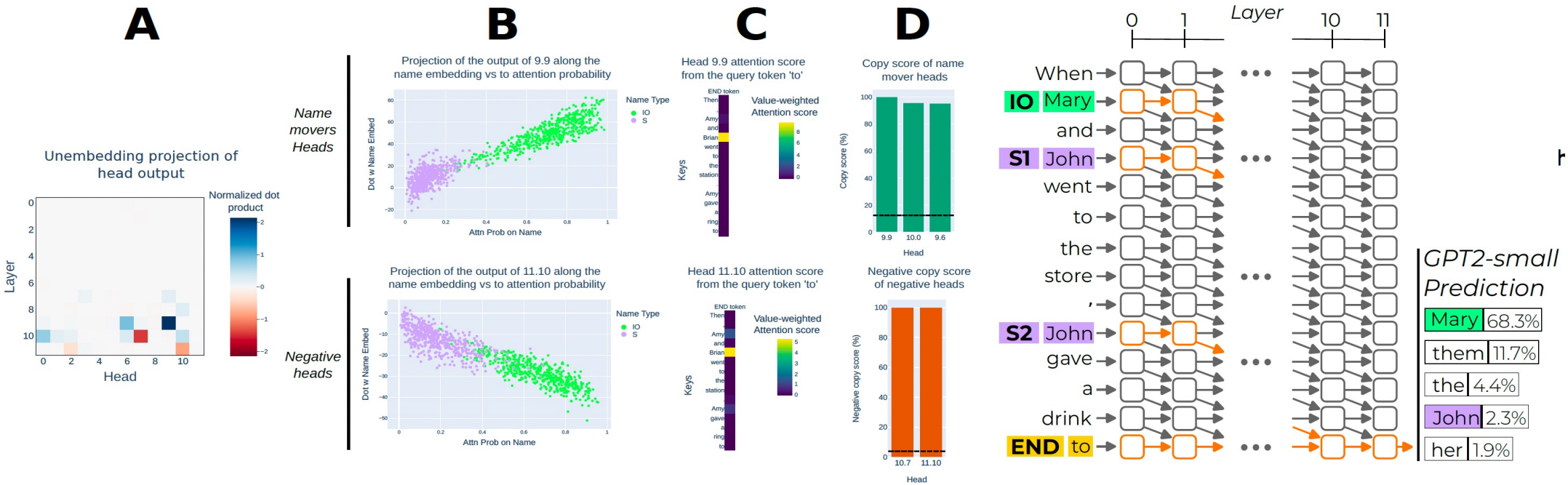
A hypothesized algorithm implemented by the extracted circuit[1]:

1.

Wang, etc. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. ICLR 2023.

Circuit: (negative) name mover head

- Using the idea of **Logit Lens**, unembed each head to $W_U[IO]$-$W_U[S]$, W_U unembedding matrix mapping from latent vectors to token logits.
- **Mover**: first attends to the name (IO or S), then computes a vector aligned with W[IO]-W[S].
- **Copy score**: among all sample input, how often the head attend to the IO or S.



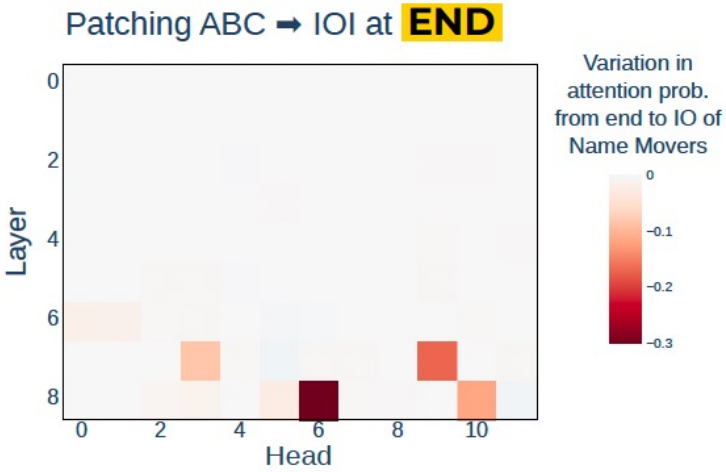
Wang, etc. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. ICLR 2023.

Circuit: S-inhibition heads

- Given the behavior of Name Mover Heads,
  - What make them attend to IO rather than other names?

- Patching all other heads at the END location using perturbed input.
  - Note at the each location, there are always 12 x 12 = 144 attention heads.

  - Observation:with patching certain heads (7.3, 7.9, 8.6, 8.10), attention to IO at END decreases.



Wang, etc. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. ICLR 2023.

Final comments

- Given Bengio's paper "Chain-of-thought is not explainability" and the long-standing and broad Responsibility issues of LLM, explainability will be a very long-lasting research questions.
  - In fact, explainability of AI has been there since 1975, regardless of models.
- As LLMs are used in many critical domains, its explainability will only become more and more important.
  - Thinking about finance, medicine, biology, industry design, and many others!
- When LLMs are used in a composite system, such as agentic system or embodied AI, its responsibility is clearly necessary.