Training Large Language Models to Reason in a Continuous Latent Space

Shibo Hao, et al. COLM 2025

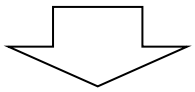
読む人: 持橋大地
統計数理研究所/国立国語研究所
最先端NLP 2025

どんな論文？

- 一言で言うと...「無言で思考するChain-of-Thought (CoT)」
- デコードした言葉のかわりに、その直前の潜在ベクトルを履歴として使う
  - 通常のCoTより高性能
  - 特定の言葉で深さ優先探索にならざるをえない通常のCoTと違い、暗黙的に幅優先探索で複数の解を探索できる
- Chain of Continuous Thought=COCONUT

問題意識

- 通常の言葉によるChain-of-Thoughtは、
  - 思考ステップの可能性を1通りに絞ってしまう
  - 言葉と実際の推論過程が違っていることがある (Wang+ 2022, Turpin+ 2024)
  - 思考と関係ない、流暢性だけのためのトークンが多数ある一方で、一部に思考が必要な複雑なトークン (例: “irreducibility”) があり、それらに同じ計算資源が割り当てられる
- 脳科学の知見によると、人間が推論を行っている時は言語野はほとんど活動していないことが多い



言語に制約されない思考を表現する必要がある

Coconutの学習データ (1)

- 学習データ: 通常のCoT用の学習データ
  - Question文とAnswer文の間に、それを繋ぐ
    中間的なCoTステップが文として与えられている
- 算数問題のGSM8kの例：

GSM8k

Question = "John cuts his grass to 2 inches. It grows .5 inches per month. When it gets to 4 inches he cuts it back down to 2 inches. It cost $100 to get his grass cut. How much does he pay per year?"
Steps = ["«4-2=2»", "«2/.5=4»", "«12/4=3»", "«100*3=300»"]
Answer = "300"

Coconutの学習データ (2)

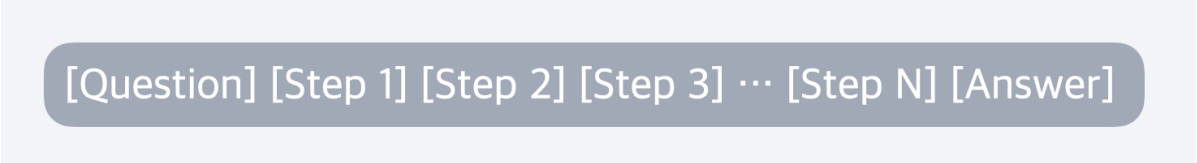
- 論理問題のProsQAの例：

ProsQA

Question = "Every shumpus is a rempus. Every shumpus is a yimpus. Every terpus is a fompus. Every terpus is a gerpus. Every gerpus is a brimpus. Alex is a rempus. Every rorpus is a scrompus. Every rorpus is a yimpus. Every terpus is a brimpus. Every brimpus is a lempus. Tom is a terpus. Every shumpus is a timpus. Every yimpus is a boompus. Davis is a shumpus. Every gerpus is a lorpus. Davis is a fompus. Every shumpus is a boompus. Every shumpus is a rorpus. Every terpus is a lorpus. Every boompus is a timpus. Every fompus is a yerpus. Tom is a dumpus. Every rempus is a rorpus. Is Tom a lempus or scrompus?"
Steps = ["Tom is a terpus.", "Every terpus is a brimpus.", "Every brimpus is a lempus."]
Answer = "Tom is a lempus."

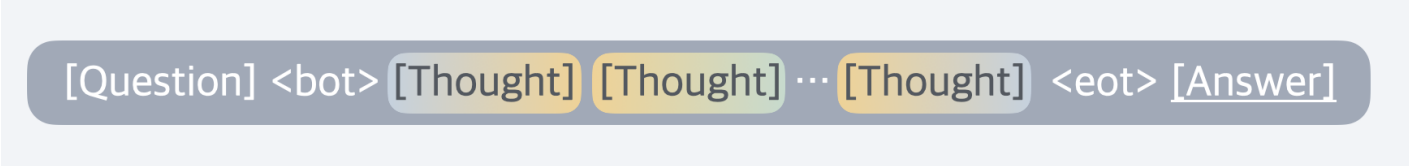
Coconutの学習 (1)

- 通常のChain-of-Thought



  - [Step *n*]はそれぞれ自然言語の文で、この埋め込みを次々と言語モデルにフィードバック

- COCONUT



  - <bot>..<eot>で囲まれたトークンは、その文を与えた直後の最終層のロジット (=“Continuous Thought”)
  - 単語埋め込みのかわりに、これを言語モデルにフィードバック

Coconutの学習 (2)

- Transformerの構造を単純化して数式で書くと、

$$\begin{cases} h_t = \mathrm{Transformer}(E_t) \\ p(w_{t+1}|w_{\leq t}) = \mathrm{Softmax}(Wh_t) \end{cases}$$

論文のものを、わかりやすく改変

- ここでトークン埋め込み関数 $e()$ を用いた履歴が

$$E_t = \left[e(w_1), e(w_2), \cdots, e(w_t)\right]$$

で、h_t は最後の隠れ層の状態

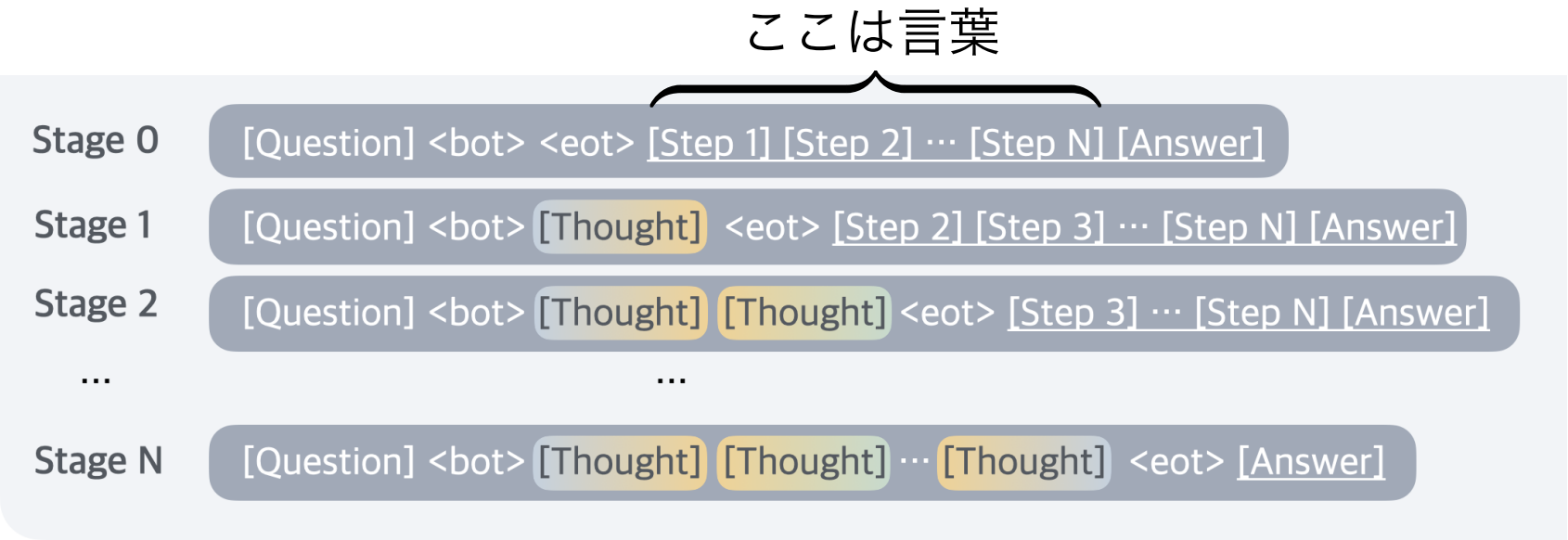
- 連続ベクトルで考えている間は、E_t が次になる

$$E_t = \left[e(w_1), \cdots, e(w_i), h_i, \cdots, h_{t-1}\right]$$

ここで $w_i = <\mathrm{bot}>$

Coconutの学習 (3)

- いきなり連続ベクトルの系列を学習しろと言っても無理なので、通常の言葉によるCoTから始めて、徐々に始めの方のトークンを連続ベクトルに置き換える



- 目的関数は連続ベクトルの分は含まず、推論結果の確率を最適化 →言語に縛られない効率的な思考を学習

関連研究

- implicit CoT (Deng+ 2024) :
  訓練中に推論ステップの文を最初から徐々に省略して、
  モデルが結果を「先読み」できるようにする → 性能向上
- “Pause”トークン (Goyal+ 2023) :
  CoTに学習可能な“<pause>” トークンを挟むことで、
  性能の向上を確認
- LLMに明示的に木構造を探索させるようにする
  (Xie+ 2023; Yao+ 2023; Hao+ 2024)
- Diffusion-of-Thoughts (Ye+ 2024) :
  言語のDiffusionモデルにCoTを導入
  →最初から連続ベクトルが対応している

実験

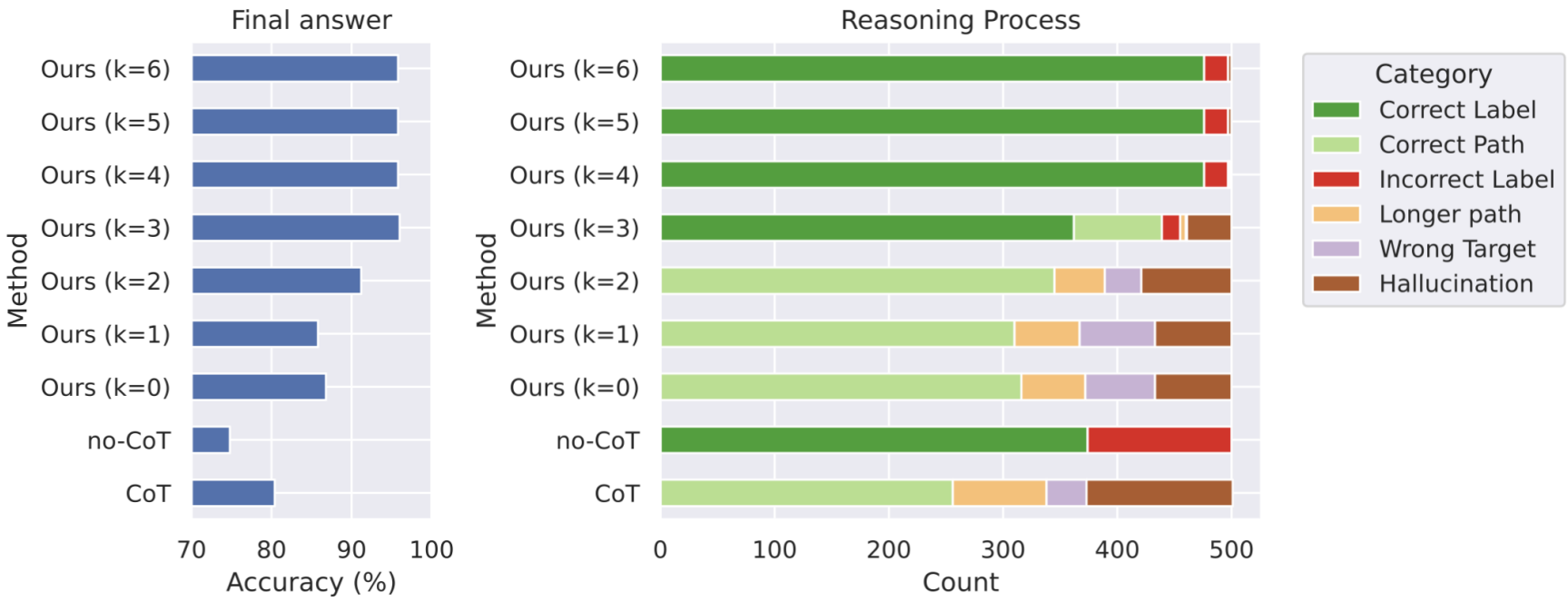
Method	GSM8k		ProntoQA		ProsQA	
	Acc. (%)	# Tokens	Acc. (%)	# Tokens	Acc. (%)	# Tokens
CoT	42.9 ±0.2	25.0	98.8 ±0.8	92.5	77.5 ±1.9	49.4
No-CoT	16.5 ±0.5	2.2	93.8 ±0.7	3.0	76.7 ±1.0	8.2
iCoT	30.0*	2.2	99.8 ±0.3	3.0	98.2 ±0.3	8.2
Pause Token	16.4 ±1.8	2.2	77.7 ±21.0	3.0	75.9 ±0.7	8.2
COCONUT (Ours)	34.1 ±1.5	8.2	99.8 ±0.2	9.0	97.0 ±0.3	14.2
- *w/o curriculum*	14.4 ±0.8	8.2	52.4 ±0.4	9.0	76.1 ±0.2	14.2
- *w/o thought*	21.6 ±0.5	2.3	99.9 ±0.1	3.0	95.5 ±1.1	8.2
- *pause as thought*	24.1 ±0.7	2.2	100.0 ±0.1	3.0	96.6 ±0.8	8.2

Table 1 Results on three datasets: GSM8l, ProntoQA and ProsQA. Higher accuracy indicates stronger reasoning ability, while generating fewer tokens indicates better efficiency. *The result is from Deng et al. (2024).

- GSM8k: 算数の問題
- ProntoQA: 論理的推論、これは推論が簡単なため、ランダムなDAGを生成して難しくしたのがProsQA
- 算数は素のCoTより若干低いが、論理的推論は高性能

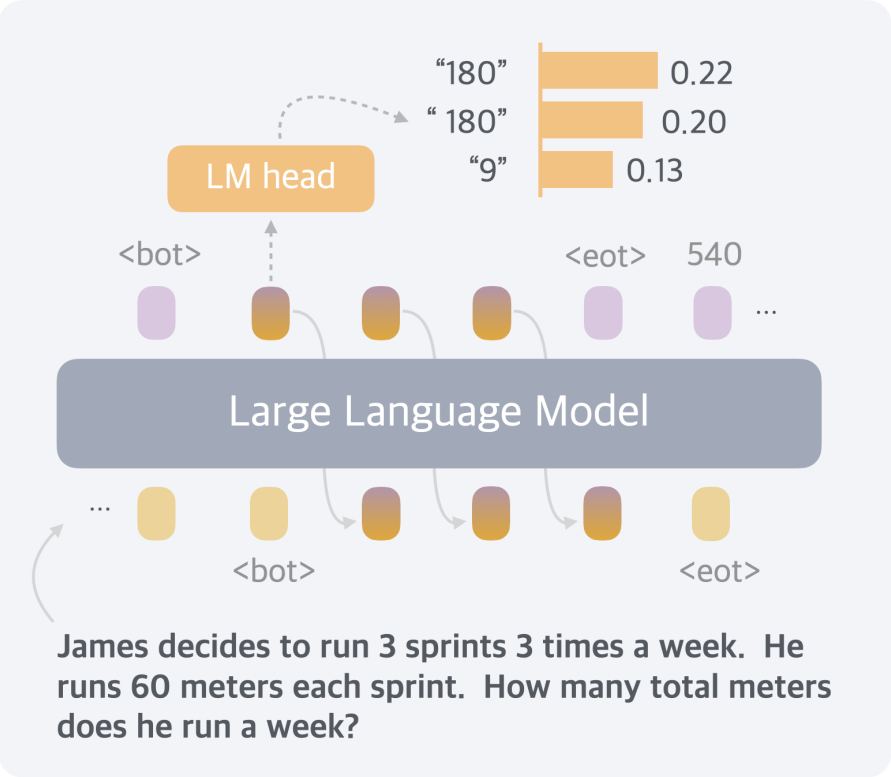
推論ステップ数と推論

ProsQAでの結果



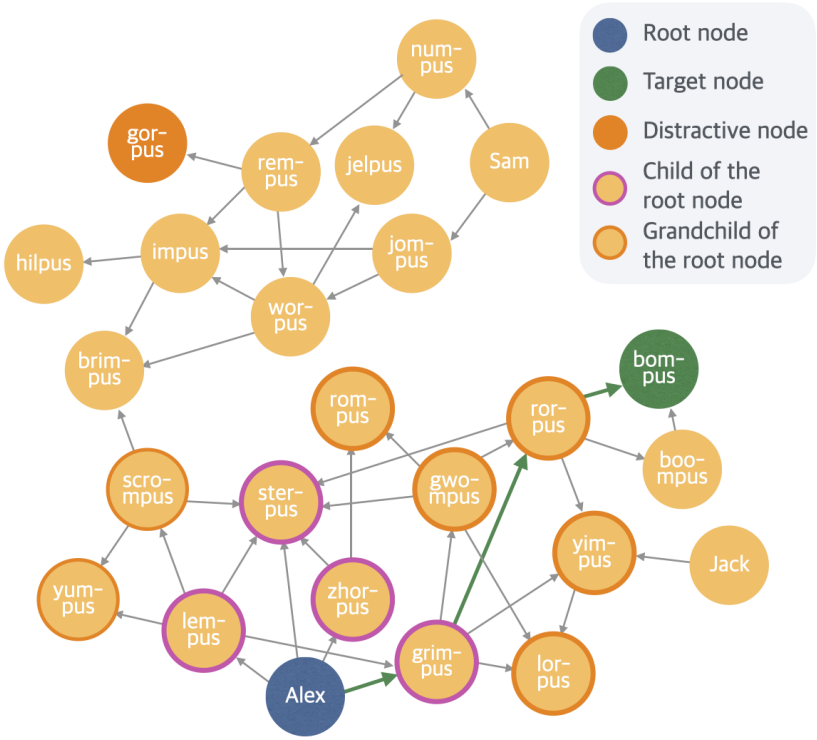
- 連続的推論のステップ数kが上がると、性能が上昇
  - *k*=4程度でほぼ飽和
- 素のCoTと比べて、正解の質も上がっている

思考ベクトルと内部状態



- 3x3x60=540 を計算させる問題の思考ベクトルをtokenにマップすると、“180” “_180” “9” のトークンの確率が高くなっていた
- 複数の思考ステップを統合して学習

ケーススタディ



Question:

Every grimpus is a yimpus. Every worpus is a jelpus. Every zhorpus is a sterpus. Alex is a grimpus ⋯ Every lumps is a yumpus.
Question: **Is Alex a gorpus or bompus?**

Ground Truth Solution

Alex is a grimpus.
Every grimpus is a rorpus.
Every rorpus is a bompus.
Alex is a bompus

CoT

Alex is a lempus.
Every lempus is a scrompus.
Every scrompus is a yumpus.
Every yumpus is a rempus.
Every rempus is a gorpus.
Alex is a gorpus ✗

(Hallucination)

Coconut (k=1)

<bot> [Thought] <eot>
Every lempus is a scrompus.
Every scrompus is a brimpus.
Alex is a brimpus ✗

(Wrong Target)

Coconut (k=2)

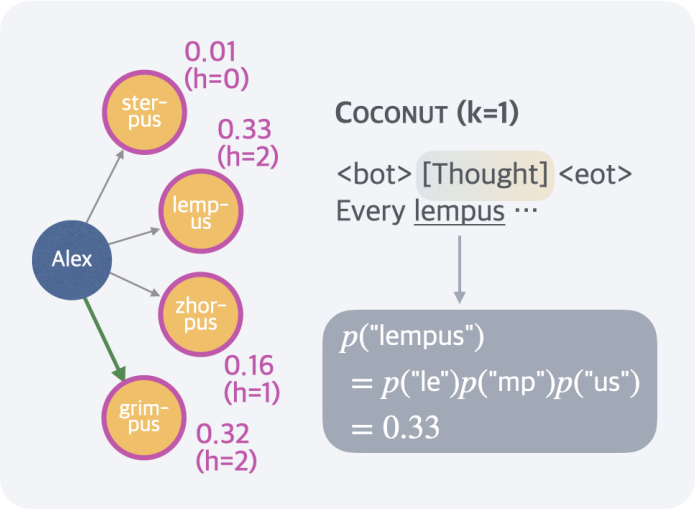
<bot> [Thought] [Thought] <eot>
Every rorpus is a bompus.
Alex is a bompus ✓

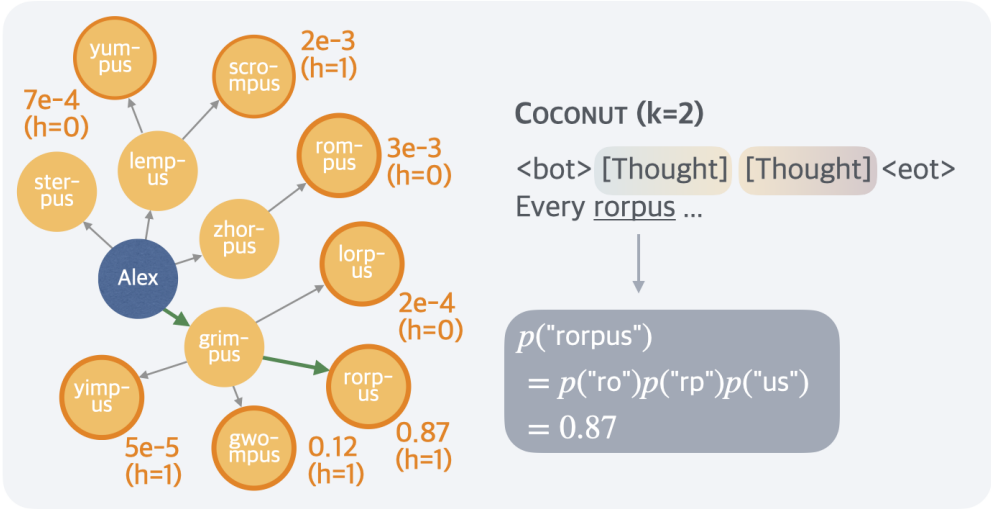
(Correct Path)

- 通常のCoTは、赤文字の誤った論理的関係をhallucinateしてしまう
- k=2のCoconutは、正しく結果を推論

ケーススタディ (2)

- この背後で何が起きているのか？





- 連続的思考の終わった<eot>の後に “Every [X] is..” を出力させてXの確率を計算すると、その時点で見込みのあるノード(lempus)に高い確率
- 2ステップ目で同様にすると、目標直前の rorpus に確率が集中している →BFS的な挙動

まとめ

まとめ

- COCONUT: 言葉を使わないChain-of-Thought
  - Chain-of-Thoughtの各文を、最後の単語の出力直前の潜在層で置き換え
- 通常のCoTよりも、場合によって高い性能
  - 思考を言葉にしてしまうと一つの可能性しか保持できず、深さ優先探索的になるが、連続ベクトルにすることで複数の可能性を幅優先探索
  - 言語に縛られないことで、効率的で短いCoTを達成