Reasoning in Latent Space

Recall the behavior of LLM's that use reasoning/"Chain of Thought"

Given prompt

$$\x_{(1:\bar{T})}$$

rather than *immediately* producing response

$$\y_{(1:\bar{T})}$$

giving computation trace

$$\x, \y$$

- (dropping the sequence subscript that indexes tokens)

an LLM is trained to think "Step by Step"

- creating a sequence of steps
- the Chain of Thought ("reasoning" trace) $\rat$
- enumerating sequential steps of a process that produces the response
$$\rat = [\rat_{(1)}, \ldots, \rat_{(\text{num\_thoughts})}]$$
- where each $\rat_{\tp}$ is a thought represented as multi-token sequence

resulting in trace

\x, \rat, \v

All the tokens in sequences $\backslash \mathrm{x}$, $\backslash \mathrm{rat}$, $\backslash \mathrm{y}$ come from the vocabulary $\backslash \mathrm{Vocab}$.

Thus, all the token sequences are in "Natural Language".

But the reasoning trace $\backslash \mathrm{rat}$ is often hidden from the user.

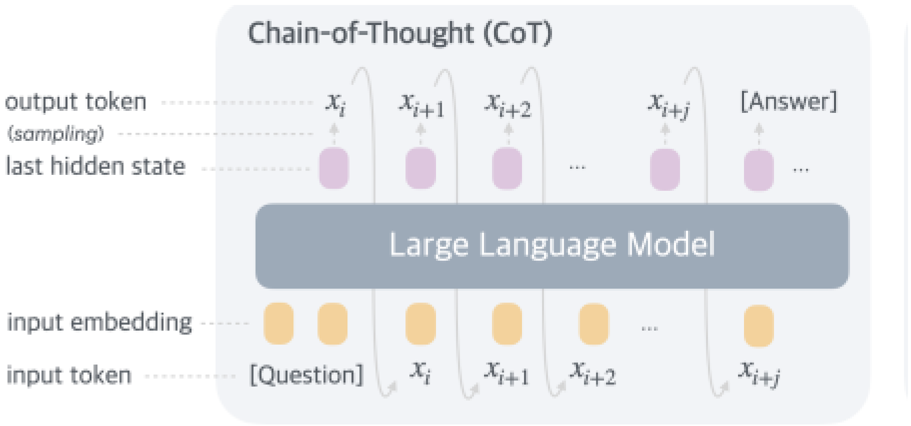
- so $\backslash \mathrm{rat}$ is not visible to the user

Does the LLM have to "think" in Natural Language ?

Here is an illustration of Chain of Thought reasoning using

- reasoning tokens
- limited to only Natural Language tokens

Chain of Thought



Attribution: https://arxiv.org/pdf/2412.06769#page=2

(Note: paper uses \x as a denotation of the output sequence vs. our standard of \y)

Note

We are not quite precise in referring to the elements of the sequence as "tokens"

Technically:

- the elements of the depicted reasoning trace are the dense *embedding vectors* associated with the tokens
    - token: syntax; embedding: vector
    - one-to-one correspondence
- key point: elements of the reasoning are selected from a *finite vocabulary*
    - tokens/embeddings

Suppose we expanded the vocabulary of the reasoning trace

- to include embeddings
- that are **not restricted** to discrete categorical values from a finite Vocabulary
- but which are continuous vectors

Being continuous rather than discrete

- enables a richer space of "thoughts"
- which can't be mapped back to a token from a fixed vocabulary

This is called *Chain of Continuous Thought (Coconut)*.

What do these non-verbal continuous "tokens" represent ?

Some theories

- pictures/diagrams that "visualize" the answer
- super-position of two (or more) continuations
    - e.g., narrow down multiple choice from 4 to 2 possibilities -- keep those 2 possibilities alive

Chain of Thought

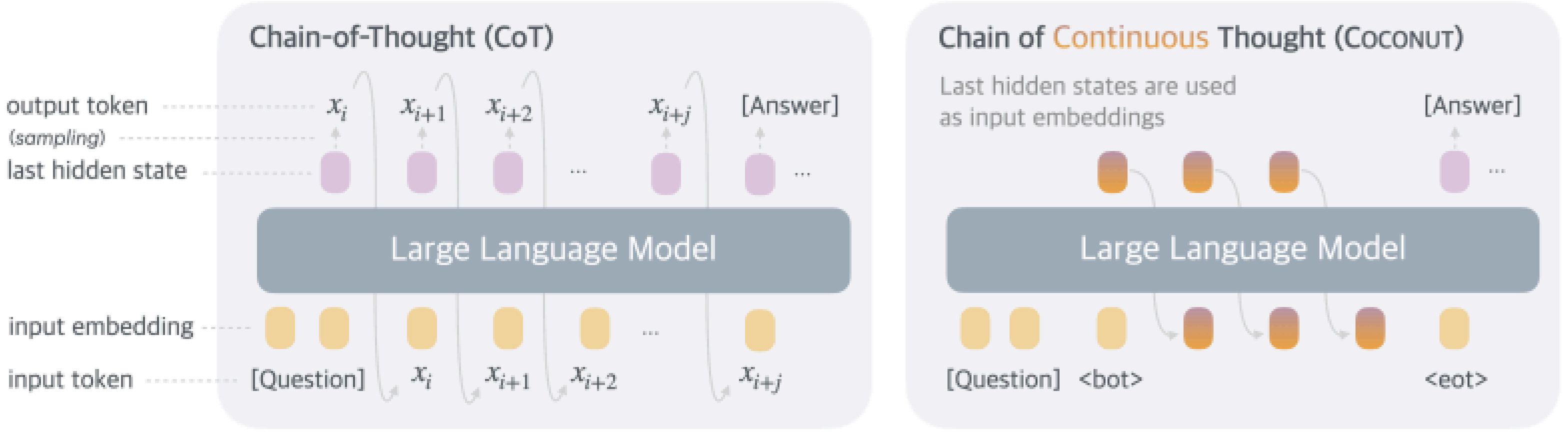


Figure 1 A comparison of Chain of Continuous Thought (COCONUT) with Chain-of-Thought (CoT). In CoT, the model generates the reasoning process as a word token sequence (e.g., $[x_i, x_{i+1}, ..., x_{i+j}]$ in the figure). COCONUT regards the last hidden state as a representation of the reasoning state (termed "continuous thought"), and directly uses it as the next input embedding. This allows the LLM to reason in an unrestricted latent space instead of a language space.

Attribution: https://arxiv.org/pdf/2412.06769#page=2

Continuous thoughts

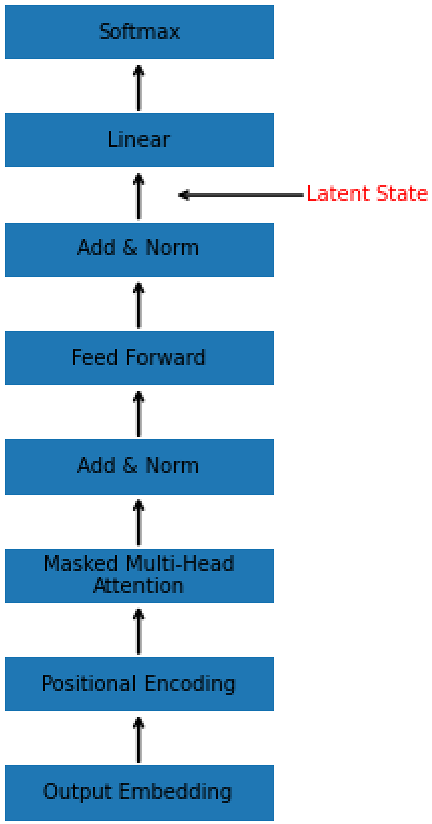
Where do the tokens that form the continuous thoughts come from ?

It is a simple adaptation of the Transformer architecture.

- see diagram below

In [5]:
```
fig_transf
```

Out[5]:



The Transformer operates in Auto-regressive mode:

- outputs one token at a time
- at step it's output $\backslash\hat{\mathbf{y}}_{\backslash\mathrm{tp}}$
- is appended to the previous output sequence $\backslash\hat{\mathbf{y}}_{(1:^{-}-1)}$
- which becomes the input for step +1

The tokens (vectors of length $|\V|$, where $\V$ is the Vocabulary of tokens) are

- *embedded* into vectors of length $d = d_{\text{model}}$
- and processed by the "body" of the Transformer
    - everything before the Linear layer

The output before the linear layer (referred to as the "Latent state")

- is a vector of length d
- which is *un-embedded* by the Linear layer into a vector of length $|\V|$
    - a probability vector over the finite, discrete set of language tokens $\V$
    - so a single token in $\V$ can be output

We represent this schematic description of the Transformer in the left plot "Language Mode".

See diagram below.

The Embed block

- transforms discrete tokens into vectors whose length d is the path width of the Transformer

The Un-embed block

- transforms the latent state vector of length d back into a Language token
    - which must then be embedded before it can be used in the next time step

The Continuous Thought tokens

- are the final vectors of the Body
    - length d
    - *Latent state*

They are *not* un-embedded.

They can be fed *directly* into the Body at the next time step

- by-passing the Un-embed of step and the subsequent Embed of step $(+1)$.

See the schematic diagram for "Latent Mode"

What does a Continuous token represent ?

Since a continuous token is not output to the user, it's vector representation does not have to be interpretable.

But the authors conducted an experiment

- decode the continuous token using the Un-embed
- resulting in a probability vector over the Language tokens in the NLP Vocabulary

Here is the input to the LLM

```
James decides to run 3 sprints 3 times a week.
He runs 60 meters each sprint.
How many total meters does he run a week ?
```

There are several paths that lead to the correct answer ($3 * 3 * 60 = 540$)

- First compute $(3 * 3)$, then multiple the result by 60
  - leading to outputting 9 as the first token of the reasoning trace
- First compute $(60 * 3)$
  - leading to outputting 180 as the first token of the reasoning trace

When the continuous thought vector is un-embedded

- it shows a probability distribution over the likely next tokens: 9 and 180

The authors interpret the thought vector as a "super-position" of possible next tokens.

Decoding a Continuous Thought

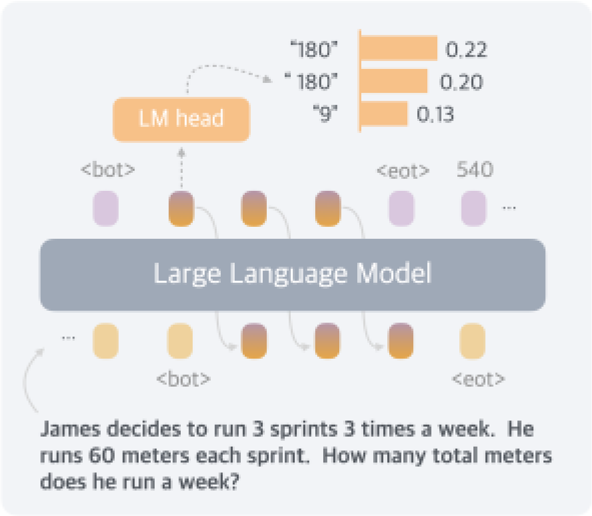


Figure 4 A case study where we decode the continuous thought into language tokens.

Attribution: https://arxiv.org/pdf/2412.06769#page=7

Absent "latent mode"

- the LLM would have to commit to **one** of the choices
    - e.g., "180" has the highest probability
    - in extending the reasoning trace
- by not committing immediately:
    - the LLM can "simultaneously" explore **both** possibilities
    - essentially allowing multiple traces (one beginning with "180", another beginning with "9")

The authors interpret this as the LLM

- performing a search across possible completions
- preserving possibilities
- until a later point in time

Training the LLM to reason in mixed language/latent mode

An LLM can be trained in CoT reasoning by using examples of reasoning traces

$$\backslash \mathrm{x}, \backslash \mathrm{rat}, \backslash \mathrm{y}$$

But this works only is the tokens of every thought in $\backslash \mathrm{rat}$ are *language* tokens.

How do we train when $\backslash \mathrm{rat}$ is a mixture of language and continuous tokens ?

- the continuous tokens are *defined* by training; not pre-specified

Clarification

Trace \rat consists of

- a number of *steps*
- each step consists of c "thoughts" (tokens: either language or continuous)

The authors use a clever technique:

Multi-stage training *curriculum*

- different training set at each stage
- at stage k
    - replace the first k *steps* ($k * c$ Natural Language tokens)
    - by continuous steps ($k * c$ continuous thoughts)
- stage 0 is traditional CoT training examples

At stage 0, the LLM learns how to reason.

- calculate a loss per language token in $\backslash\mathrm{rat}$
- guiding the learning-to-reason process

At subsequent stages, it learns to derive continuous tokens incrementally

- increasing k, the number of continuous thoughts

By proceeding incrementally

- the model does not "un-learn" traditional CoT in language tokens

The loss for continuous tokens

- is **masked** (not calculated)
    - no actual target with which to compare the prediction
- the loss of the language tokens following the continuous tokens
    - is reduced
    - when the preceding continuous tokens contribute to the prediction of the language tokens

The underlined elements in the diagram are the ones for which a loss is calculated.

Note

- each thought is a sequence of up to c tokens

Continuous tokens bracketed by `<bot>, <eot>`

Decoding a Continuous Thought

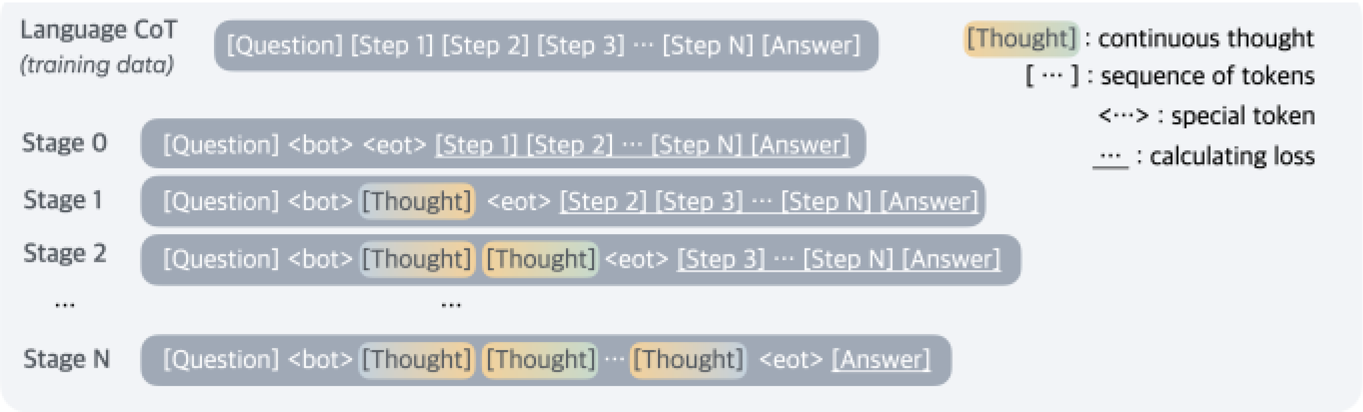


Figure 2 Training procedure of Chain of Continuous Thought (Coconut). Given training data with language reasoning steps, at each training stage we integrate c additional continuous thoughts ($c = 1$ in this example), and remove one language reasoning step. The cross-entropy loss is then used on the remaining tokens after continuous thoughts.

Attribution: https://arxiv.org/pdf/2412.06769#page=4

Why is Latent Space potentially powerful

The reasoning paradigm of "think before you speak" is very powerful

- CoT leads to better outcomes
- by (potentially) hiding the thinking steps from the user
    - enables model to
        - reflect and revise
        - avoiding prematurely committing to a final answer

Allowing the thinking to occur in latent space

- superposition of multiple answers: no premature commitment to final answer
- non-verbal thoughts
    - images (e.g., a map enables one to think in relative position rather than precise directions)

Recall our original model for dealing with sequences

- Recurrent models with latent state
    - control of Finite State Machine
    - long term memory (LSTM)

Do the (hidden) "thoughts" of CoT have the same effect ?

- Hidden thought: a reminder of what to do next/later (control)

In [8]:
```
print("Done")
```

```
Done
```