

Transformers and Turing Machines

CSE 60963: Theory of Neural Networks

David Chiang (Univ. of Notre Dame, USA)

Jon Rawski (MIT/San Jose State Univ., USA)

Lena Strobl (Umeå University, Sweden)

Andy Yang (Univ. of Notre Dame, USA)

2024-11-04

Main Result (too informal)

Theorem (Pérez et al., 2019, Pérez et al., 2021)

Transformers are Turing-complete!!!1

Main Result (too informal)

▲ tambourine_man on June 15, 2023 | parent | next [-]

Transformers are Turing complete, right?

▲ nyrikki on June 15, 2023 | root | parent | next [-]

The paper from yesterday:

<https://news.ycombinator.c...>

Showed that attention with positional encodings and arbitrary precision rational activation functions is Turing complete.

Using a finite precision, nonrational activation function and/or without positional encodings is not Turing complete.

▲ canjobear on June 15, 2023 | root | parent | prev | next [-]

No, they are actually very limited formally. For example you can't model a language of nested brackets to arbitrary depth (as you can with an RNN). That makes it all the more interesting that they are so successful.

<https://news.ycombinator.com/item?id=36311871>

Main Result (informal)

Theorem (Pérez et al., 2019, Pérez et al., 2021)

An average-hard attention transformer decoder using position embeddings and intermediate steps can simulate a Turing machine.

Today's Goals

- **Define** a transformer *decoder* as well as *intermediate steps* and how they relate to *chain of thought*
- **Explain** in what sense a transformer is and isn't Turing-complete
- **Understand** how a transformer, which has no memory, can simulate a Turing machine's tape

Background

Main Result (informal)

Theorem (Pérez et al., 2019, Pérez et al., 2021)

An *average-hard attention* transformer decoder using position embeddings and intermediate steps can simulate a Turing machine.

Average-Hard Attention

- Like unique-hard attention, attention goes only to highest-scoring positions
- Unlike unique-hard attention, attention is divided equally among highest-scoring positions

| scores s | 0 | 1 | 1 |
|----------------|------|------|------|
| softmax | 0.16 | 0.42 | 0.42 |
| leftmost-hard | 0 | 1 | 0 |
| rightmost-hard | 0 | 0 | 1 |
| average-hard | 0 | 0.5 | 0.5 |

Average-Hard Attention

Exercises:

| | | | | | |
|------------|---|---|---|---|---|
| scores s | 2 | 2 | 0 | 2 | 1 |
|------------|---|---|---|---|---|

average-hard

| | | | | | |
|------------|---|---|---|---|---|
| scores s | 0 | 0 | 0 | 0 | 0 |
|------------|---|---|---|---|---|

average-hard

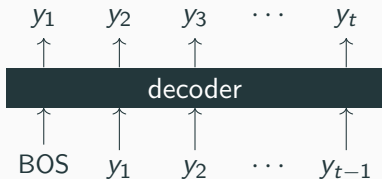
Main Result (informal)

Theorem (Pérez et al., 2019, Pérez et al., 2021)

*An average-hard attention **transformer decoder** using position embeddings and intermediate steps can simulate a Turing machine.*

Transformer Decoders

- At each time step, the decoder outputs a word
 - In practice: a probability distribution over words
 - Often in theory: the exact embedding of a word
- The output word becomes the next input word (autoregression)



Future Masking

- At each time step i , attention only attends to positions $j \leq i$
- Optional in encoders, mandatory in decoders

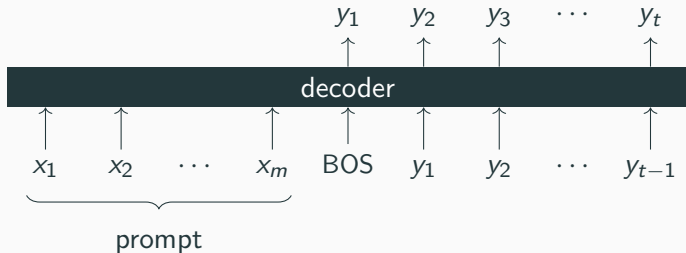
Exercise: Without autoregression, which vectors does vector x depend on?

| | | | | | | |
|---|---|---|---|---|---|---|
| i | i | i | j | k | k | k |
| g | g | g | x | h | h | h |
| d | d | d | e | f | f | f |
| a | a | a | b | c | c | c |

(How about with autoregression?)

Prompting

- BOS can be preceded by input symbols, known as a *prompt*



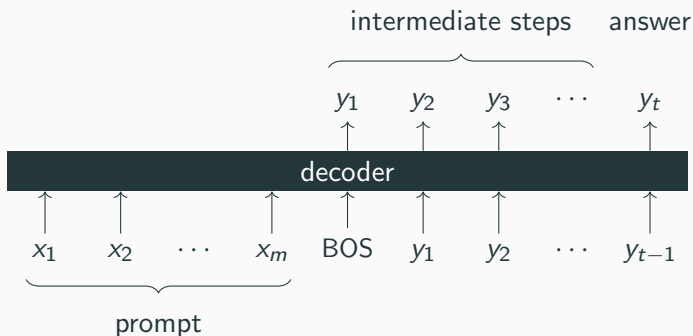
Main Result (informal)

Theorem (Pérez et al., 2019, Pérez et al., 2021)

*An average-hard attention transformer decoder using position embeddings and **intermediate steps** can simulate a Turing machine.*

Intermediate Steps

- Allow *intermediate* output symbols between BOS and the final output (here, just a single symbol)



Theory Predicts Practice

Standard Prompting

Model Input

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

A: The answer is 11.

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

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

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

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

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

Model Output

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

[Wei et al., 2022]

Main Result (informal)

Theorem (Pérez et al., 2019, Pérez et al., 2021)

An average-hard attention transformer decoder using position embeddings and intermediate steps can simulate a
Turing machine.

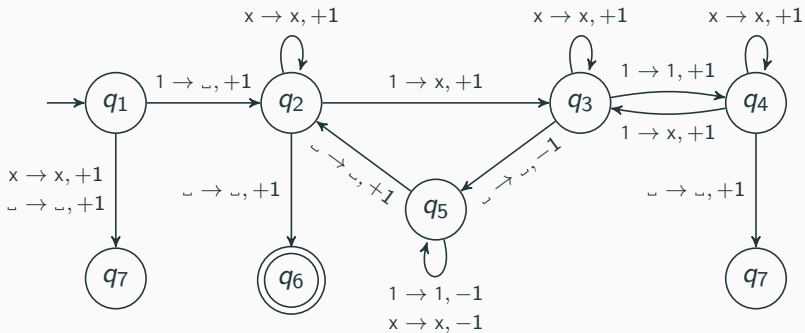
Turing Machines



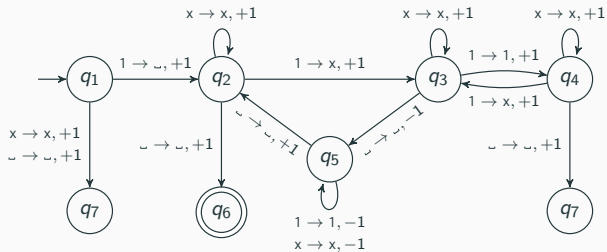
- Semi-infinite tape
- Cells are numbered starting from 0
- At each step, head moves either left (-1) or right ($+1$)

Turing Machines

Turing machine for $\{1^{2^m} \mid m \geq 0\}$



Turing Machines



q_1 $\hat{1}1 \sqcup \dots$

q_2 $\sqcup \hat{1} \sqcup \dots$

q_3 $\sqcup x \hat{\sqcup} \dots$

q_5 $\sqcup x \hat{\sqcup} \dots$

q_5 $\hat{\sqcup} x \sqcup \dots$

q_2 $\sqcup x \hat{\sqcup} \dots$

q_6 accept

Question: What are the inputs and outputs of a Turing machine's transition function?

$$\delta(q, a) = (q', b, m)$$

Simulating Turing Machines

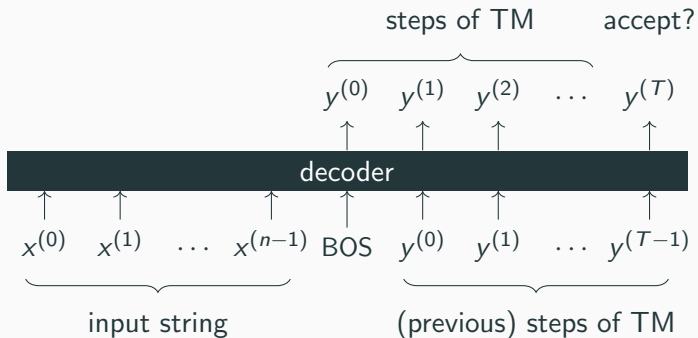
Theorem (Pérez et al., 2019, Pérez et al., 2021)

For any Turing machine M with input alphabet Σ , there is an average-hard attention transformer decoder f with position embedding $i \mapsto (1/(i+1), 1, i, i^2)$ that is equivalent to M in the following sense. For any string $w \in \Sigma^$:*

- *If M halts and accepts on input w , then there is a T such that f , on prompt w , outputs ACC after T intermediate steps.*
- *If M halts and rejects on input w , then there is a T such that f , on prompt w , outputs REJ after T intermediate steps.*
- *If M does not halt on input w , then there does not exist a T such that f , on prompt w , outputs either ACC or REJ.*

Key Idea 1

Use intermediate steps to simulate steps of Turing machine



Key Idea 2

How can a transformer, which has no memory, simulate a Turing machine's head?

Key Idea 2

How can a transformer, which has no memory, simulate a Turing machine's head?

Answer: Sum all the previous moves

| | | | | | |
|----------------------------|----|----|----|----|----|
| previous move $m^{(i-1)}$ | +1 | +1 | -1 | -1 | +1 |
| current position $h^{(i)}$ | 1 | 2 | 1 | 0 | 1 |

Key Idea 3

How can a transformer, which has no memory, simulate a Turing machine's tape?

Key Idea 3

How can a transformer, which has no memory, simulate a Turing machine's tape?

Answer: Look at the most recent time the head was at current position $h^{(i)}$

- If none, the current symbol is the $h^{(i)}$ -th input symbol
- If time j , the current symbol is what was written at time j

Input: 11 \sqcup \cdots

| | | | | | |
|-------------------------------|----------|----------|----------|----------|----------|
| previous position $h^{(i-1)}$ | 0 | 1 | 2 | 1 | 0 |
| previous write $b^{(i-1)}$ | \sqcup | x | \sqcup | x | \sqcup |
| current position $h^{(i)}$ | 1 | 2 | 1 | 0 | 1 |
| current symbol $a^{(i)}$ | 1 | \sqcup | x | \sqcup | x |

Position Embeddings

Position Embeddings

Adding a few simple elements to the position embeddings enables a very useful set of tricks [Barceló et al., 2024]:

$$\text{PE}(i) = \begin{bmatrix} 1/(i+1) \\ 1 \\ i \\ i^2 \end{bmatrix} .$$

Forward Lookup

Assume that we have computed, for $i \in [n]$,

$$f(i) \in [n]$$

$$g(i) \in \mathbb{R}$$

Forward lookup: Compute $g(f(i))$ for $i \in [n]$

| | | | | | | |
|-----------|--|---|---|---|---|----|
| i | | 0 | 1 | 2 | 3 | 4 |
| $f(i)$ | | 0 | 1 | 1 | 2 | 3 |
| $g(i)$ | | 0 | 1 | 4 | 9 | 16 |
| $g(f(i))$ | | 0 | 1 | 1 | 4 | 9 |

Forward Lookup

Assume that we have computed, for $i \in [n]$,

$$f(i) \in [n]$$

$$g(i) \in \mathbb{R}$$

Forward lookup: Compute $g(f(i))$ for $i \in [n]$

$$\text{query}_i = \begin{bmatrix} f(i) \\ 1 \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2j \\ -j^2 \end{bmatrix} \quad \text{value}_j = [g(j)]$$

Exercise: Compute $\text{score}_{ij} = \text{query}_i \cdot \text{key}_j$ and maximize with respect to j .

Forward Lookup

Assume that we have computed, for $i \in [n]$,

$$f(i) \in [n]$$

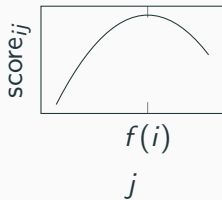
$$g(i) \in \mathbb{R}$$

Forward lookup: Compute $g(f(i))$ for $i \in [n]$

$$\text{query}_i = \begin{bmatrix} f(i) \\ 1 \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2j \\ -j^2 \end{bmatrix} \quad \text{value}_j = [g(j)]$$

$$\begin{aligned} \text{score}_{ij} &= \text{query}_i \cdot \text{key}_j \\ &= -j^2 + 2f(i)j \end{aligned}$$

$$\underset{j}{\operatorname{argmax}} \text{score}_{ij} = f(i).$$



Backward Lookup

Assume that we have computed, for $i \in [n]$,

$$f(i) \in \mathbb{R}$$

$$g(i) \in \mathbb{R}$$

Backward lookup: compute $g^{-1}(f(i))$ for $i \in [n]$. That is, find j such that $g(j) = f(i)$.

| | | | | | | |
|----------------|--|---|---|---|---|----|
| i | | 0 | 1 | 2 | 3 | 4 |
| <hr/> | | | | | | |
| $f(i)$ | | 0 | 1 | 1 | 4 | 9 |
| $g(i)$ | | 0 | 1 | 4 | 9 | 16 |
| <hr/> | | | | | | |
| $g^{-1}(f(i))$ | | 0 | 1 | 1 | 2 | 3 |

Backward Lookup

Assume that we have computed, for $i \in [n]$,

$$f(i) \in \mathbb{R}$$

$$g(i) \in \mathbb{R}$$

Backward lookup: compute $g^{-1}(f(i))$ for $i \in [n]$. That is, find j such that $g(j) = f(i)$.

$$\text{query}_i = \begin{bmatrix} f(i) \\ 1 \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2g(j) \\ -g(j)^2 \end{bmatrix} \quad \text{value}_j = [j]$$

$$\begin{aligned} \text{score}_{ij} &= \text{query}_i \cdot \text{key}_j \\ &= -g(j)^2 + 2f(i)g(j) \end{aligned}$$

$$\operatorname{argmax}_j \text{score}_{ij} = g^{-1}(f(i)).$$

Backward Lookup

Assume that we have computed, for $i \in [n]$,

$$f(i) \in \mathbb{R}$$

$$g(i) \in \mathbb{R}$$

Backward lookup: compute $g^{-1}(f(i))$ for $i \in [n]$. That is, find j such that $g(j) = f(i)$.

$$\text{query}_i = \begin{bmatrix} f(i) \\ 1 \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2g(j) \\ -g(j)^2 \end{bmatrix} \quad \text{value}_j = [j]$$

forward lookup

$$\begin{aligned} \text{score}_{ij} &= \text{query}_i \cdot \text{key}_j \\ &= -g(j)^2 + 2f(i)g(j) \end{aligned}$$

$$\operatorname{argmax}_j \text{score}_{ij} = g^{-1}(f(i)).$$

Multiplication and Division

Assume that we have computed, for $i \in [n]$,

$$f(i) \in \mathbb{R}$$

$$g(i) \in \mathbb{R}$$

and we want to compute $f(i)/g(i) \in [n]$.

| | | | | | | |
|-----------|--|---|---------------|---------------|---------------|---------------|
| i | | 0 | 1 | 2 | 3 | 4 |
| $f(i)$ | | 1 | $\frac{1}{2}$ | $\frac{1}{3}$ | $\frac{3}{4}$ | $\frac{2}{5}$ |
| $g(i)$ | | 1 | $\frac{1}{2}$ | $\frac{1}{3}$ | $\frac{1}{4}$ | $\frac{1}{5}$ |
| $g(f(i))$ | | 1 | 1 | 1 | 3 | 2 |

Multiplication and Division

Assume that we have computed, for $i \in [n]$,

$$f(i) \in \mathbb{R}$$

$$g(i) \in \mathbb{R}$$

and we want to compute $f(i)/g(i) \in [n]$.

$$\text{query}_i = \begin{bmatrix} f(i) \\ g(i) \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2j \\ -j^2 \end{bmatrix} \quad \text{value}_j = [j]$$

$$\begin{aligned} \text{score}_{ij} &= \text{query}_i \cdot \text{key}_j \\ &= -g(i)j^2 + 2f(i)j \end{aligned}$$

$$\underset{j}{\operatorname{argmax}} \text{score}_{ij} = f(i)/g(i).$$

Exercise: How would you compute $f(i)/g(i)$?

Simulating Turing Machines: In More Detail

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|--------------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1} _ \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1} _ \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1} _ \dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1} _ \dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_ \hat{1} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_ _ \hat{x} _ \dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_ _ \hat{x} _ \dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_ _ \hat{x} _ \dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_ _ \hat{x} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

wait

Transformer reads in input string, simulated TM does nothing

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ | |
|-----|--------------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|------|
| 0 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 | |
| 1 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 | |
| 2 | $\hat{1}1_\dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ | init |
| 3 | $\hat{1}1_\dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ | |
| 4 | $_ \hat{1} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ | |
| 5 | $_ x _ \hat{_} \dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ | |
| 6 | $_ x _ \hat{_} \dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ | |
| 7 | $_ x _ \hat{_} \dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ | |
| 8 | $_ x _ \hat{_} \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC | |

Transformer outputs fake action, simulated TM enters start state

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|--------------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1}1_\dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1}1_\dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_ \hat{1} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_ x _ \hat{_} \dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_ x _ \hat{_} \dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_ x _ \hat{_} \dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_ x _ \hat{_} \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

run

Transformer outputs the actions of the simulated TM one by one

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|--------------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1} _ \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1} _ \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1} _ \dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1} _ \dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_ \hat{1} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_ _ \hat{x} _ \dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_ _ \hat{x} _ \dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_ _ \hat{x} _ \dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_ _ \hat{x} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

input

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|-----------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1}1 \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1}1 \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1}1 \dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1}1 \dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_ \hat{1} \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_ x \hat{_} \dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_ x \hat{_} \dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_ x \hat{_} \dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_ x \hat{_} \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

step 1

Unpack previous action into state (q), write (b), and move (m)

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|-------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1}1_\dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1}1_\dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_1_\dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_x_\dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_x_\dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_x_\dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_x_\dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

step 2

Compute head position (h) by summing previous moves

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|-------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1}1_\dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1}1_\dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1}1_\dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_1_\dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_x_\dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_x_\dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_x_\dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_x_\dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

step 3

Compute current tape symbol (a)

Detailed Overview

| i | tape | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ | $h^{(i)}$ | $a^{(i)}$ | $y^{(i)}$ |
|-----|--------------------------|-----------------|-----------|-------------|-------------|-----------|-----------|-----------------|
| 0 | $\hat{1} _ \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 1 | $\hat{1} _ \dots$ | 1 | \perp | \perp | 0 | 0 | 1 | 1 |
| 2 | $\hat{1} _ \dots$ | BOS | \perp | \perp | 0 | 0 | 1 | $(q_1, 1, 0)$ |
| 3 | $\hat{1} _ \dots$ | $(q_1, 1, 0)$ | q_1 | 1 | 0 | 0 | 1 | $(q_2, _, +1)$ |
| 4 | $_ \hat{1} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | 1 | $(q_3, x, +1)$ |
| 5 | $_ _ \hat{x} _ \dots$ | $(q_3, x, +1)$ | q_3 | x | +1 | 2 | $_$ | $(q_5, _, -1)$ |
| 6 | $_ _ \hat{x} _ \dots$ | $(q_5, _, -1)$ | q_5 | $_$ | -1 | 1 | x | $(q_5, x, -1)$ |
| 7 | $_ _ \hat{x} _ \dots$ | $(q_5, x, -1)$ | q_5 | x | -1 | 0 | $_$ | $(q_2, _, +1)$ |
| 8 | $_ _ \hat{x} _ \dots$ | $(q_2, _, +1)$ | q_2 | $_$ | +1 | 1 | x | ACC |

step 4

Output next action

Step 1: Unpack Input Symbol

...into current state, previously written symbol, and previous move.

| phase | $x^{(i)}$ | $q^{(i)}$ | $b^{(i-1)}$ | $m^{(i-1)}$ |
|-------|--------------|-----------|-------------|-------------|
| wait | $\in \Sigma$ | \perp | \perp | 0 |
| init | BOS | \perp | \perp | 0 |
| run | (r, b, m) | r | b | m |

Use a FFN to compute the function

$$x^{(i)} \mapsto \begin{bmatrix} q^{(i)} \\ b^{(i-1)} \\ m^{(i-1)} \end{bmatrix}$$

Step 2a: Compute Head Position

The current head position is just the sum of all previous moves:

$$h^{(i)} = \sum_{j=0}^i m^{(j-1)}$$

Using uniform self-attention, we can compute

$$h^{(i)} / (i + 1) = \frac{1}{i + 1} \sum_{j=0}^i m^{(j-1)}$$

Question: How do we get rid of the $\cdot / (i + 1)$?

Step 3: Compute Tape Symbol

Find most recent $j < i$ such that $h^{(j)} = h^{(i)}$ and return $b^{(j)}$;
if none, return $x^{h^{(i)}}$.

Step 3: Compute Tape Symbol

Find most recent $j < i$ such that $h^{(j)} = h^{(i)}$ and return $b^{(j)}$;
if none, return $x^{h^{(i)}}$.

We can't do this because we only have non-strict masking ($j \leq i$).

Step 3: Compute Tape Symbol

Find most recent $j < i$ such that $h^{(j)} = h^{(i)}$ and return $b^{(j)}$;
if none, return $x^{h^{(i)}}$.

Find most recent $j \leq i$ such that $h^{(j-1)} = h^{(i)}$ and return $b^{(j-1)}$;
if none, return $x^{h^{(i)}}$.

This is a reverse table lookup of $h^{(i)}$ in $j \mapsto h^{(j-1)}$. But finding the *most recent* will require some extra care.

Step 3b: Compute Tape Symbol

Find most recent $j \leq i$ such that $h^{(j-1)} = h^{(i)}$ and return $b^{(j-1)}$; if none, return $x^{h^{(i)}}$.

Reverse table lookup with tie-breaking:

$$\text{query}_i = \begin{bmatrix} h^{(i)} \\ 1 \\ \frac{1}{2(i+1)} \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2h^{(j-1)} \\ -(h^{(j-1)})^2 \\ j \end{bmatrix} \quad \text{value}_j = \begin{bmatrix} h^{(j-1)} \\ b^{(j-1)} \end{bmatrix}$$

Attention scores are:

$$\text{score}_{i,j} = \underbrace{-(h^{(j-1)})^2 + 2h^{(i)}h^{(j-1)}}_{\text{find } h^{(j-1)} = h^{(i)}} + \underbrace{\frac{j}{2(i+1)}}_{\text{find rightmost}}$$

Step 3b: Compute Tape Symbol

Find most recent $j \leq i$ such that $h^{(j-1)} = h^{(i)}$ and return $b^{(j-1)}$; if none, return $x^{h^{(i)}}$.

Reverse table lookup with tie-breaking: $= h^{(j)} - m^{(j-1)}$

$$\text{query}_i = \begin{bmatrix} h^{(i)} \\ 1 \\ \frac{1}{2(i+1)} \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2h^{(j-1)} \\ -(h^{(j-1)})^2 \\ j \end{bmatrix} \quad \text{value}_j = \begin{bmatrix} h^{(j-1)} \\ b^{(j-1)} \end{bmatrix}$$

Attention scores are:

$$\text{score}_{i,j} = \underbrace{-(h^{(j-1)})^2 + 2h^{(i)}h^{(j-1)}}_{\text{find } h^{(j-1)} = h^{(i)}} + \underbrace{\frac{j}{2(i+1)}}_{\text{find rightmost}}$$

Step 3b: Compute Tape Symbol

Find most recent $j \leq i$ such that $h^{(j-1)} = h^{(i)}$ and return $b^{(j-1)}$; if none, return $x^{h^{(i)}}$.

Reverse table lookup with tie-breaking: $= h^{(j)} - m^{(j-1)}$

query_i = $\begin{bmatrix} h^{(i)} \\ 1 \\ \frac{1}{2(i+1)} \end{bmatrix}$ key_j = $\begin{bmatrix} 2h^{(j-1)} \\ -(h^{(j-1)})^2 \\ j \end{bmatrix}$ value_j = $\begin{bmatrix} h^{(j-1)} \\ b^{(j-1)} \end{bmatrix}$

forward lookup

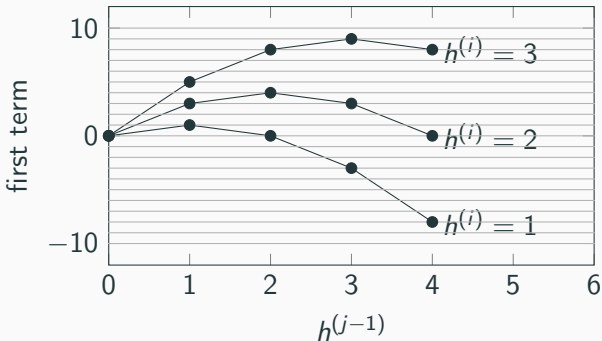
Attention scores are:

$$\text{score}_{i,j} = \underbrace{-(h^{(j-1)})^2 + 2h^{(i)}h^{(j-1)}}_{\text{find } h^{(j-1)} = h^{(i)}} + \underbrace{\frac{j}{2(i+1)}}_{\text{find rightmost}}$$

Step 3b: Compute Tape Symbol

$$\text{score}_{i,j} = \underbrace{-(h^{(j-1)})^2 + 2h^{(i)}h^{(j-1)}}_{\text{find } h^{(j-1)} = h^{(i)}} + \underbrace{\frac{j}{2(i+1)}}_{\text{find rightmost}}$$

In first term, difference between best and second-best score is 1:



So we make the second term $\frac{j}{2(i+1)} \leq \frac{1}{2} < 1$.

Step 3b: Compute Tape Symbol

Find most recent $j \leq i$ such that $h^{(j-1)} = h^{(i)}$ and return $b^{(j-1)}$; if none, return $x^{h^{(j-1)}}$.

$$\text{query}_i = \begin{bmatrix} h^{(i)} \\ 1 \\ \frac{1}{2(i+1)} \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2h^{(j-1)} \\ -(h^{(j-1)})^2 \\ j \end{bmatrix} \quad \text{value}_j = \begin{bmatrix} h^{(j-1)} \\ b^{(j-1)} \end{bmatrix}$$

Attention scores are:

$$\text{score}_{i,j} = \underbrace{-(h^{(j-1)})^2 + 2h^{(i)}h^{(j-1)}}_{\text{find } h^{(j-1)} = h^{(i)}} + \underbrace{\frac{j}{2(i+1)}}_{\text{find rightmost}}$$

FFN:

$$\begin{aligned} \text{if } h^{(j-1)} = h^{(i)} \text{ and } b^{(j-1)} \neq \perp &\rightsquigarrow a^{(i)} := b^{(j-1)} \\ \text{else} &\rightsquigarrow a^{(i)} := x^{(h^{(i)})} \end{aligned}$$

Step 3b: Compute Tape Symbol

Find most recent $j \leq i$ such that $h^{(j-1)} = h^{(i)}$ and return $b^{(j-1)}$; if none, return $x^{h^{(j-1)}}$.

$$\text{query}_i = \begin{bmatrix} h^{(i)} \\ 1 \\ \frac{1}{2(i+1)} \end{bmatrix} \quad \text{key}_j = \begin{bmatrix} 2h^{(j-1)} \\ -(h^{(j-1)})^2 \\ j \end{bmatrix} \quad \text{value}_j = \begin{bmatrix} h^{(j-1)} \\ b^{(j-1)} \end{bmatrix}$$

Attention scores are:

$$\text{score}_{i,j} = \underbrace{-(h^{(j-1)})^2 + 2h^{(i)}h^{(j-1)}}_{\text{find } h^{(j-1)} = h^{(i)}} + \underbrace{\frac{j}{2(i+1)}}_{\text{find rightmost}}$$

FFN:

$$\begin{aligned} \text{if } h^{(j-1)} = h^{(i)} \text{ and } b^{(j-1)} \neq \perp &\rightsquigarrow a^{(i)} := b^{(j-1)} \\ \text{else} &\rightsquigarrow a^{(i)} := x^{h^{(i)}} \end{aligned}$$

forward lookup

Step 4: Compute Transition

Given $x^{(i)}$ = current input symbol and $a^{(i)}$ = current tape symbol:

if $x^{(i)} \in \Sigma \rightsquigarrow$ output $x^{(i)}$

if $x^{(i)} = \text{BOS} \rightsquigarrow$ let $(r, b, m) = (q_{\text{start}}, a^{(i)}, 0)$

else \rightsquigarrow let $(r, b, m) = \delta(q^{(i)}, a^{(i)})$

if $r = q_{\text{accept}} \rightsquigarrow$ output ACC

if $r = q_{\text{reject}} \rightsquigarrow$ output REJ

else \rightsquigarrow output (r, b, m)

Recap

- Transformer decoders generate strings; they may be allowed to generate *intermediate steps* (a.k.a. chain of thought).
- A transformer decoder can simulate a Turing machine by generating the steps of the Turing machine as intermediate steps. But not if the answer is required immediately.
- Even though it has no memory, a transformer can use attention to reconstruct what it needs to know about the Turing machine configuration.

- Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir Podolskii. Logical languages accepted by transformer encoders with hard attention. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=gbrHZq07mq>.
- Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22:75:1–75:35, 2021. URL <http://jmlr.org/papers/v22/20-302.html>.
- Jorge Pérez, Javier Marinković, and Pablo Barceló. On the Turing completeness of modern neural network architectures. In *Proceedings of the Seventh International Conference on Learning Representations (ICLR)*, 2019. URL <https://openreview.net/forum?id=HyGBdo0qFm>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pages 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.