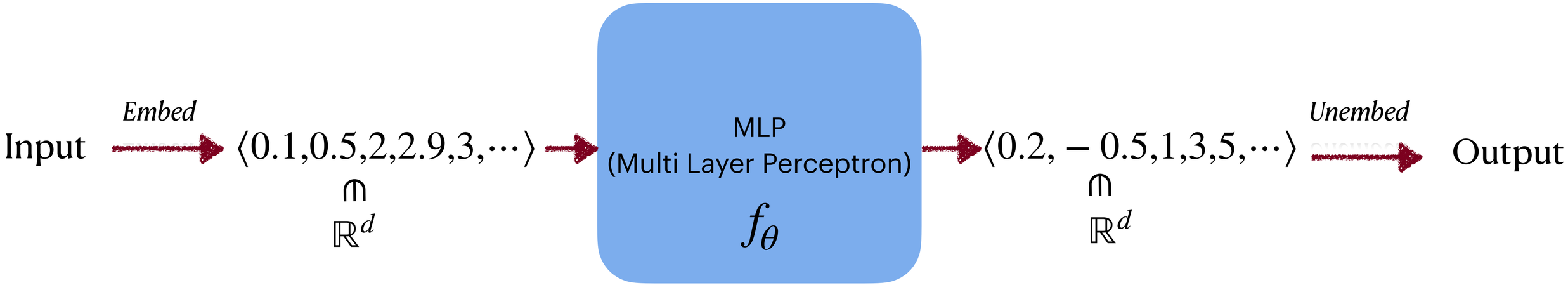
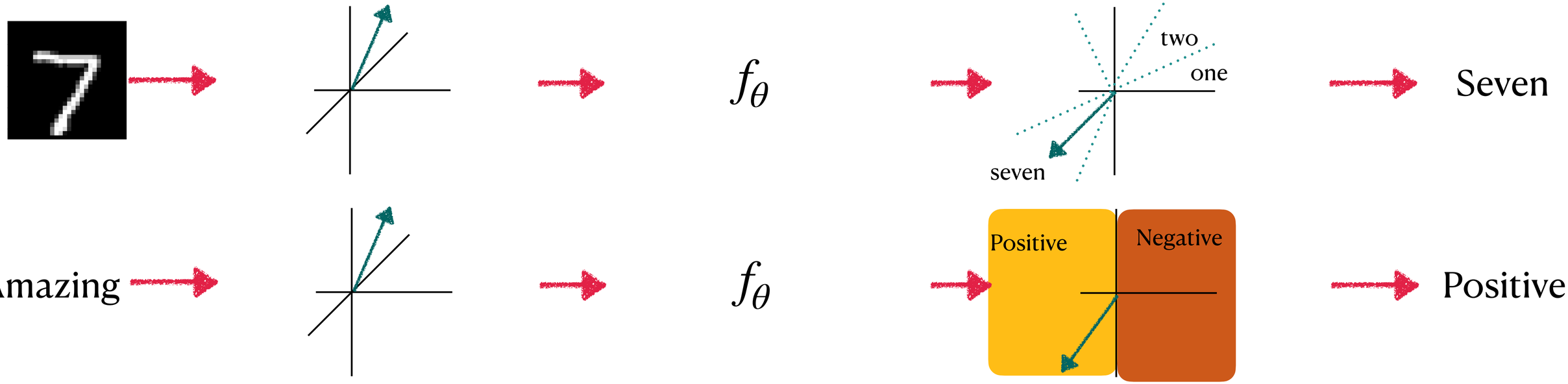
Analysis of reasoning models

JMM 2026, Washington DC

Abhinav Chand 01/06/2026

MLP


Input
Embed
⟨0.1,0.5,2,2.9,3,⋯⟩
⋔
$\mathbb{R}^d$
MLP
(Multi Layer Perceptron)
f_θ
⟨0.2, − 0.5,1,3,5,⋯⟩
⋔
$\mathbb{R}^d$
Unembed
Output


f_θ
two
one
seven
Seven
Amazing
f_θ
Positive
Negative
Positive

Transformer

Sequence → *Embed* $\begin{pmatrix} 0.1, 0.5, 2, 2.9, 3, \cdots \\ \cdots\cdots \\ 0, 0.1, 1, 1.9, 0, \cdots \end{pmatrix}$ → Transformer f_θ → $\begin{pmatrix} 0, 0.2, 2.1, 1.9, -0.1, \cdots \\ \cdots\cdots \\ 0, -0.1, 1.1, 1.9, 0, \cdots \end{pmatrix}$ *Unembed* → Output

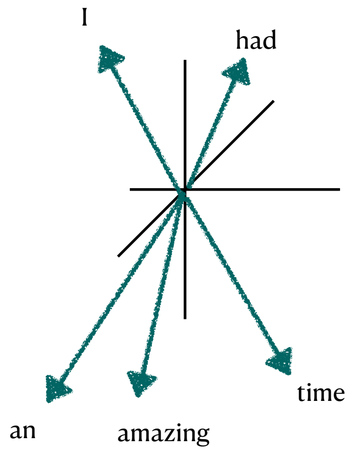
$\Uparrow$

$\mathbb{R}^{n \times d}$

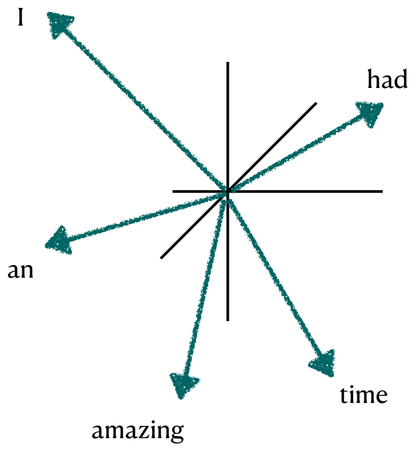
$\Uparrow$

$\mathbb{R}^{n \times d}$

I had an amazing time. → *Embed* →



→ f_θ →

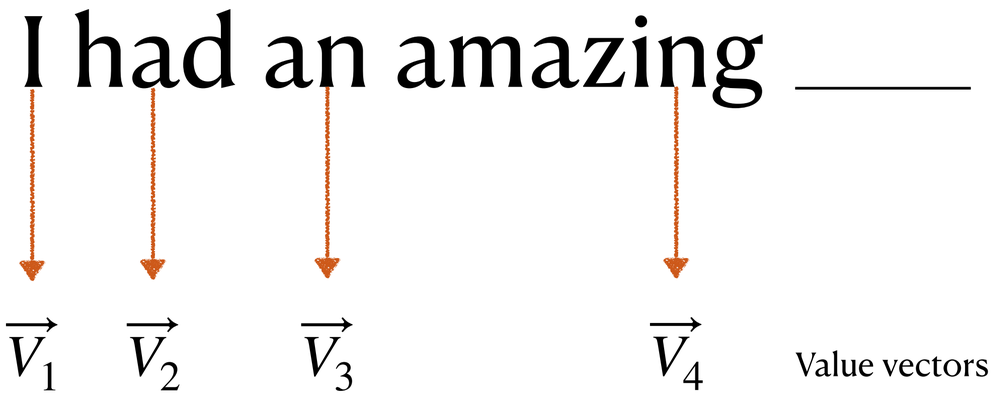


→ Positive

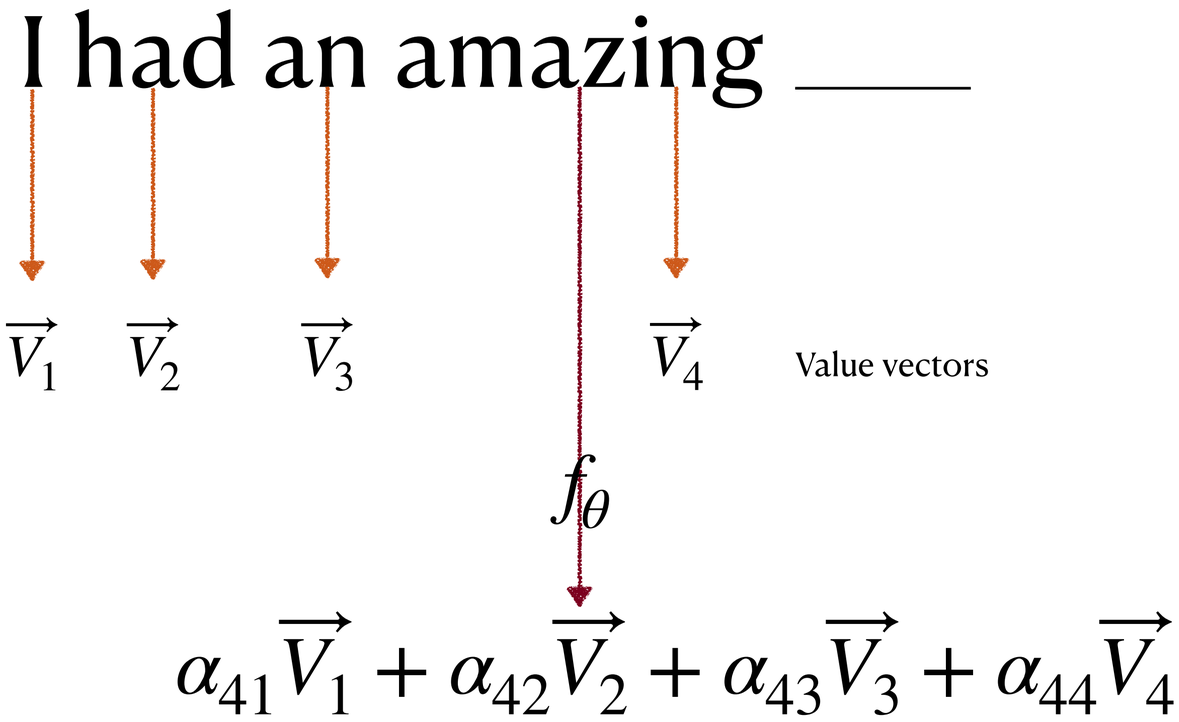
Attention is all we need

I had an amazing ____

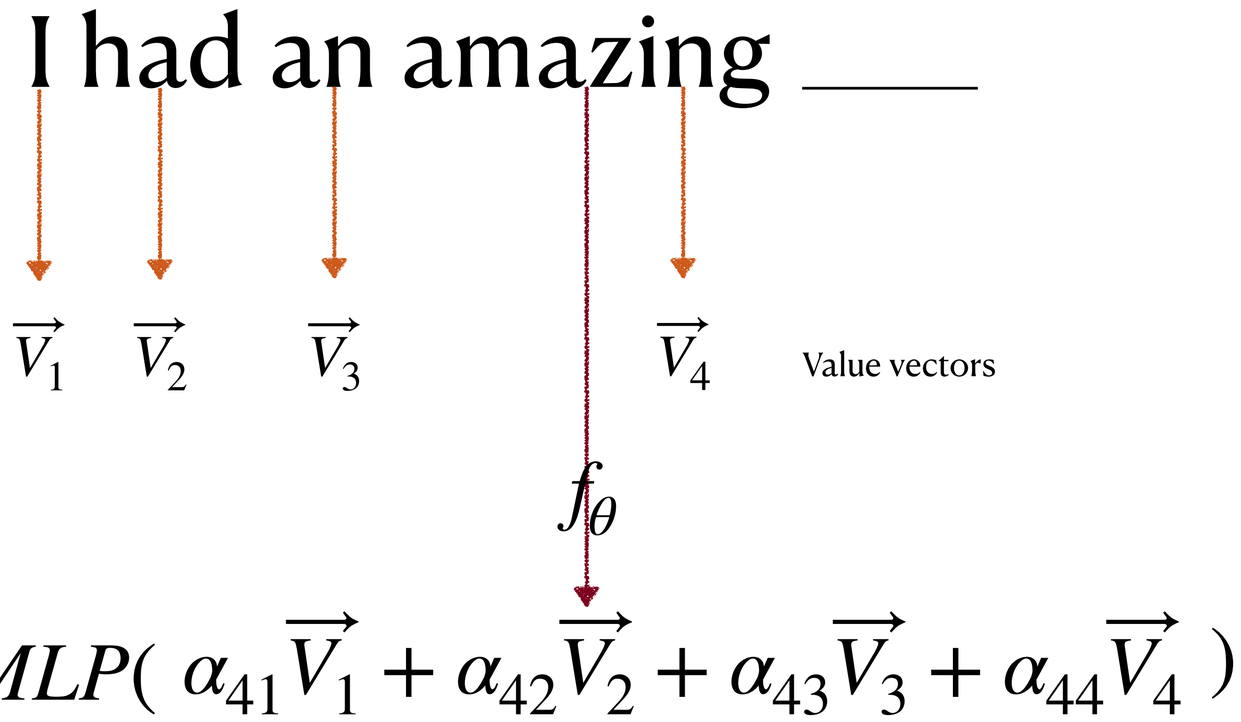
Attention is all we need



Attention is all we need

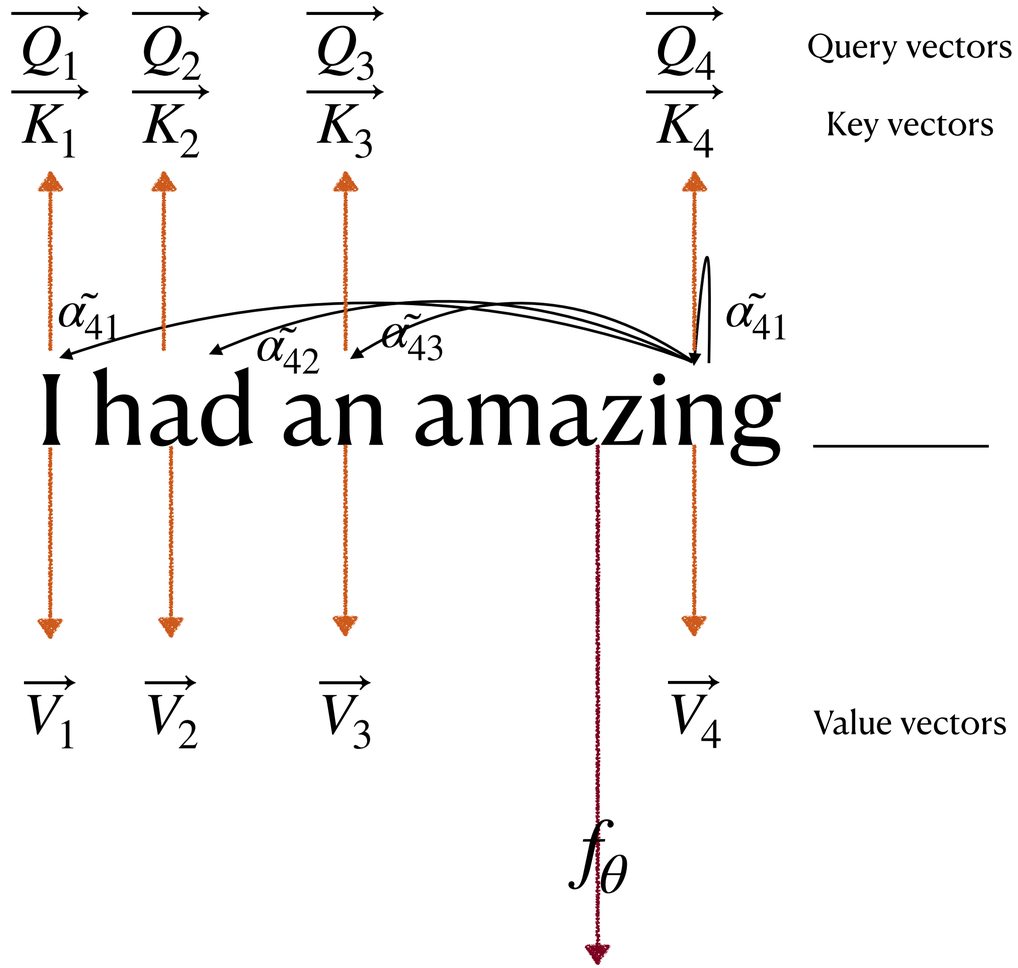


Attention is all we need



Attention is all we need

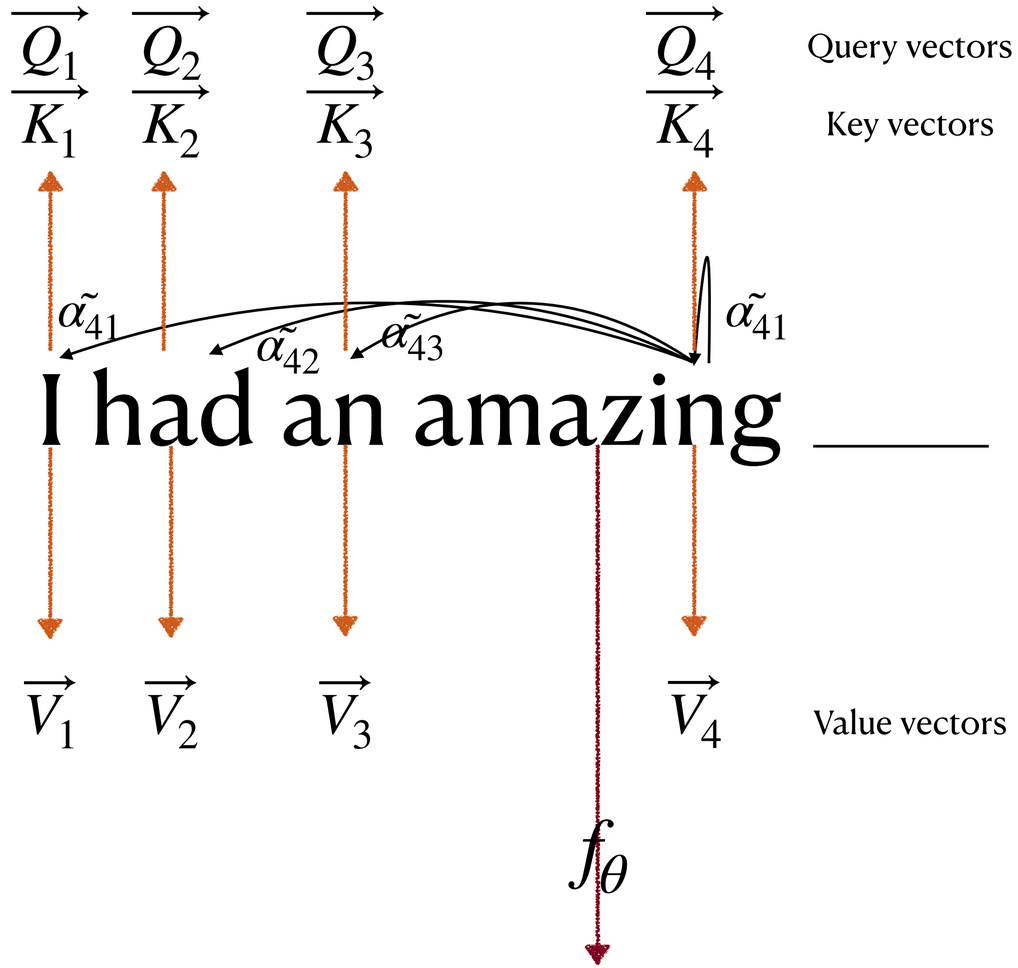
$$\tilde{\alpha}_{ij} = Q_i K_j^T$$



$$MLP(\ \alpha_{41}\overrightarrow{V_1} + \alpha_{42}\overrightarrow{V_2} + \alpha_{43}\overrightarrow{V_3} + \alpha_{44}\overrightarrow{V_4}\ )$$

Attention is all we need

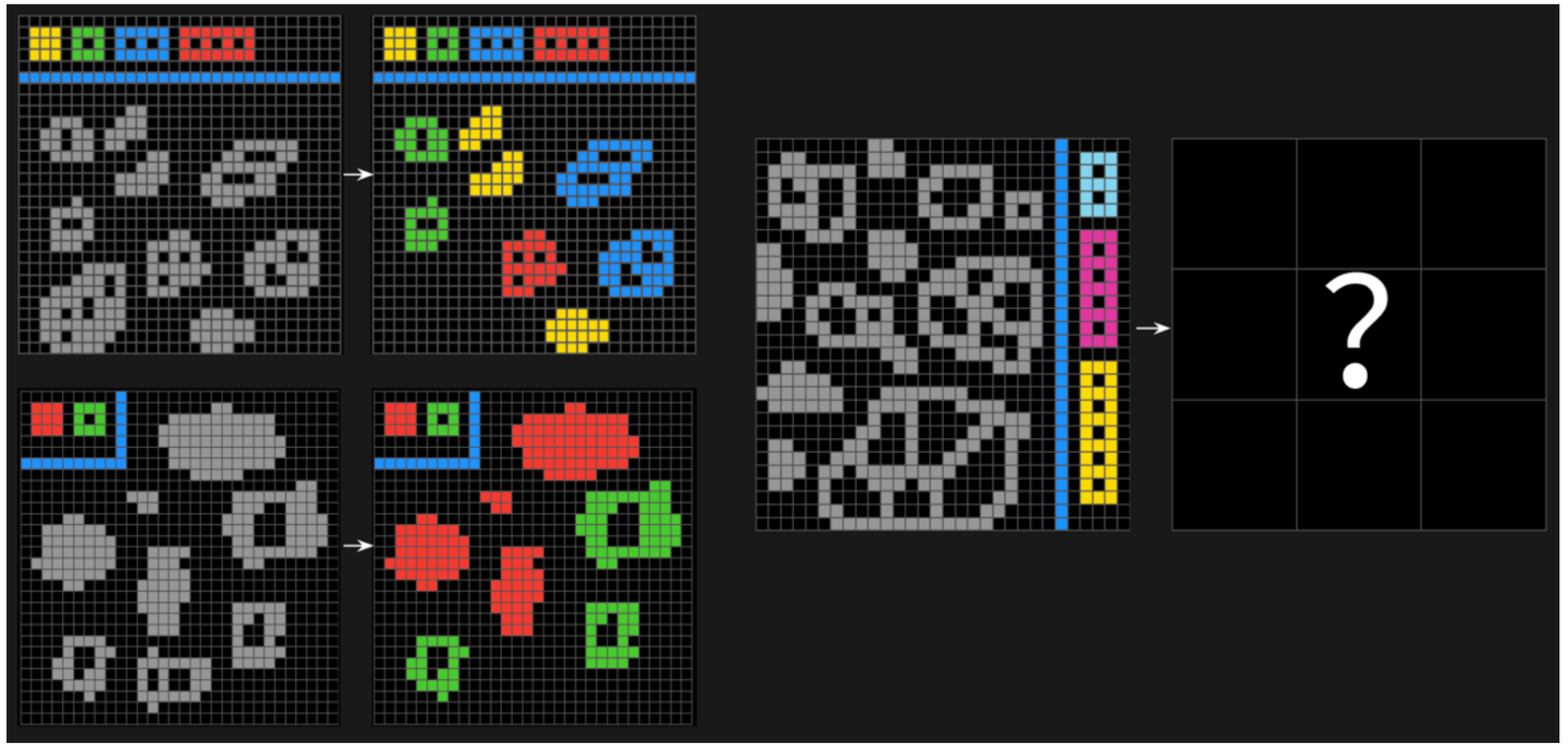
$$\tilde{\alpha}_{ij} = Q_i K_j^T$$



$$\langle \alpha_{41}, \alpha_{42}, \alpha_{43}, \alpha_{44} \rangle$$
$$= \text{softmax}(\langle \tilde{\alpha}_{41}, \tilde{\alpha}_{42}, \tilde{\alpha}_{43}, \tilde{\alpha}_{44} \rangle)$$
$$= \left\langle \cdots, \frac{e^{\tilde{\alpha}_{4i}}}{\sum_j e^{\tilde{\alpha}_{4j}}}, \cdots \right\rangle$$

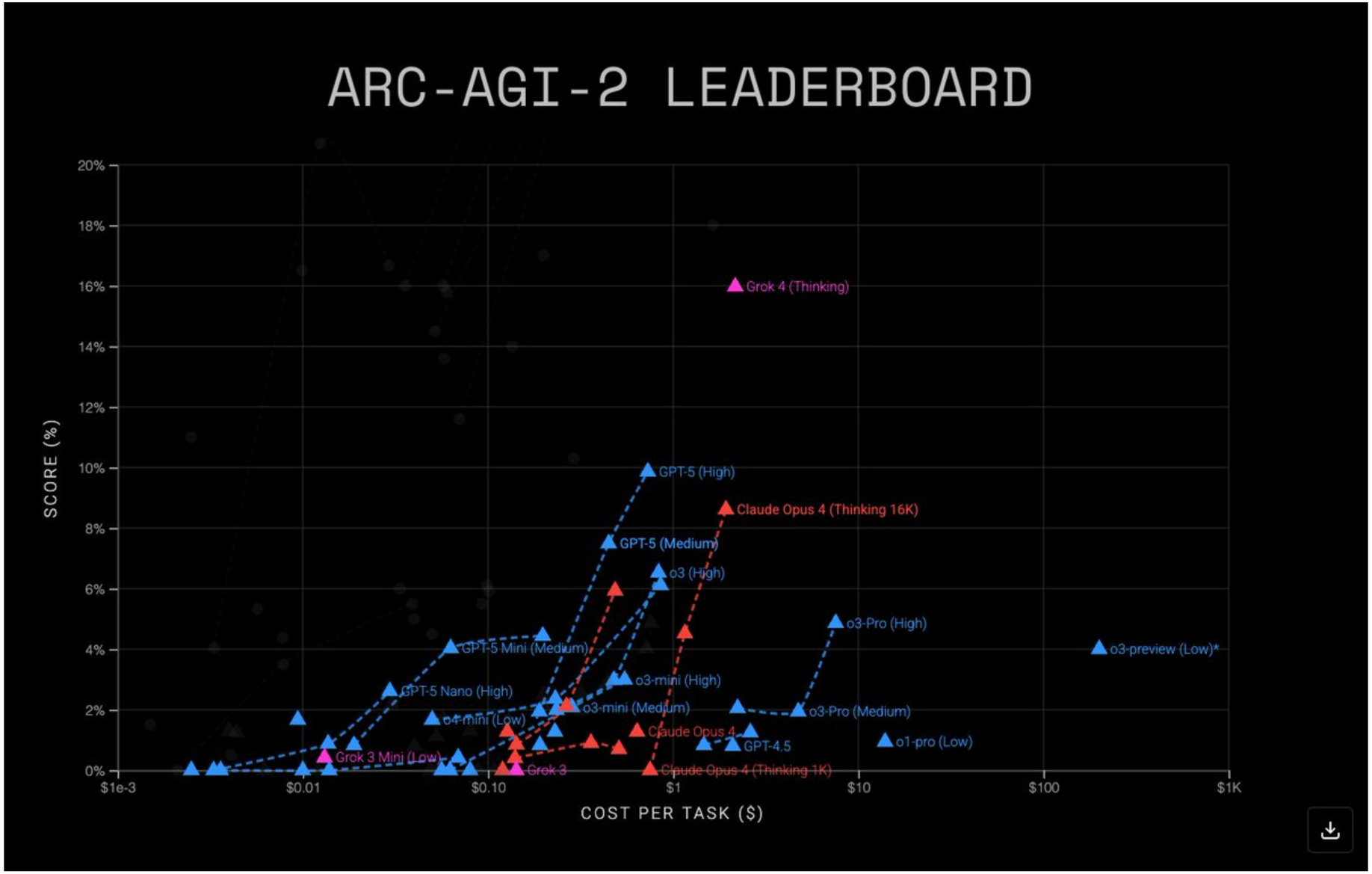
$$MLP(\ \alpha_{41}\overrightarrow{V_1} + \alpha_{42}\overrightarrow{V_2} + \alpha_{43}\overrightarrow{V_3} + \alpha_{44}\overrightarrow{V_4}\ )$$

ARC Prize

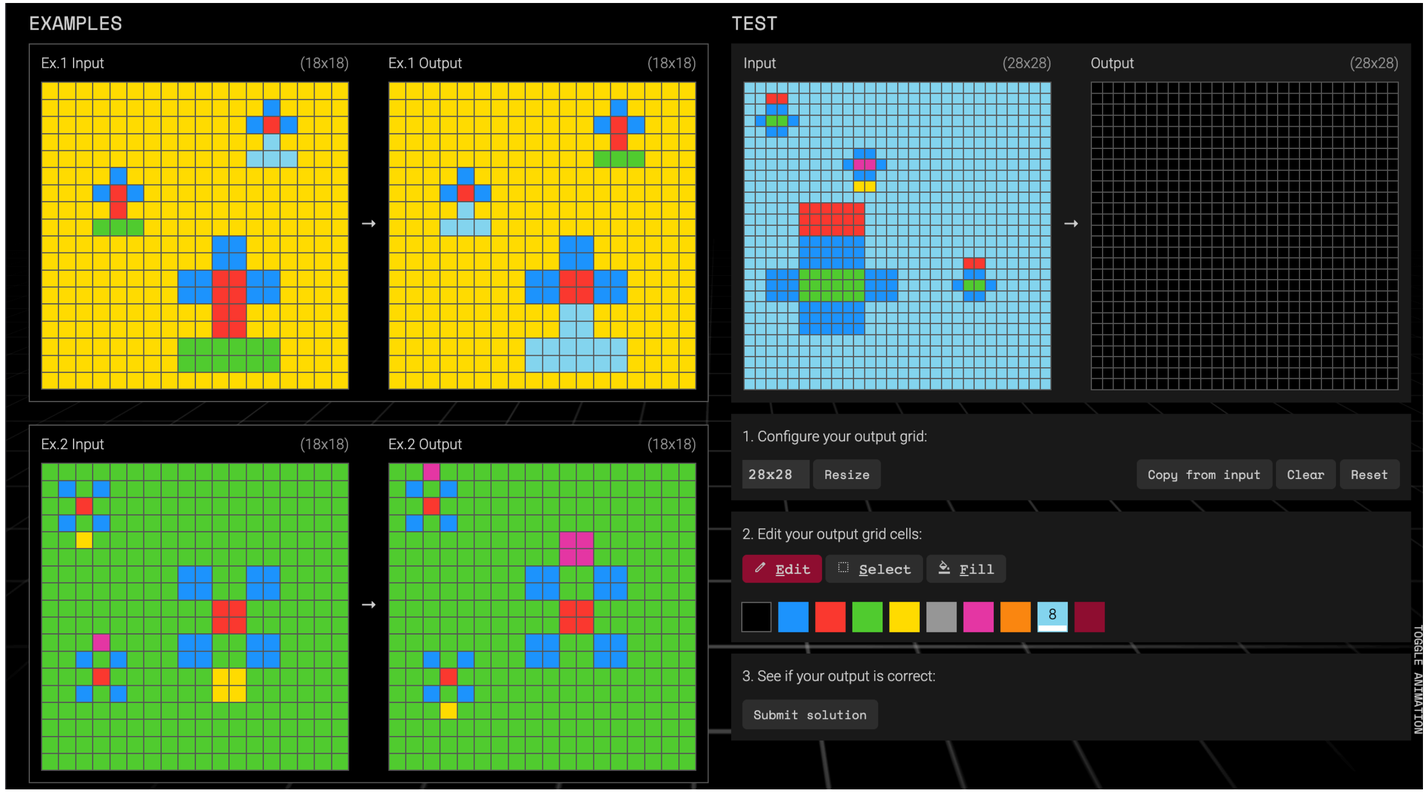


- Abstraction and Reasoning Corpus challenge
- ARC-AGI I (2019), ARC-AGI 2 (2025)

August leaderboard



Novel problem for deep learning



- Each puzzle is unique. Test puzzles not similar to training puzzles.
- Few shot learning

Hierarchical Reasoning Model

EyeingAI
@EyeingAI · **Follow**

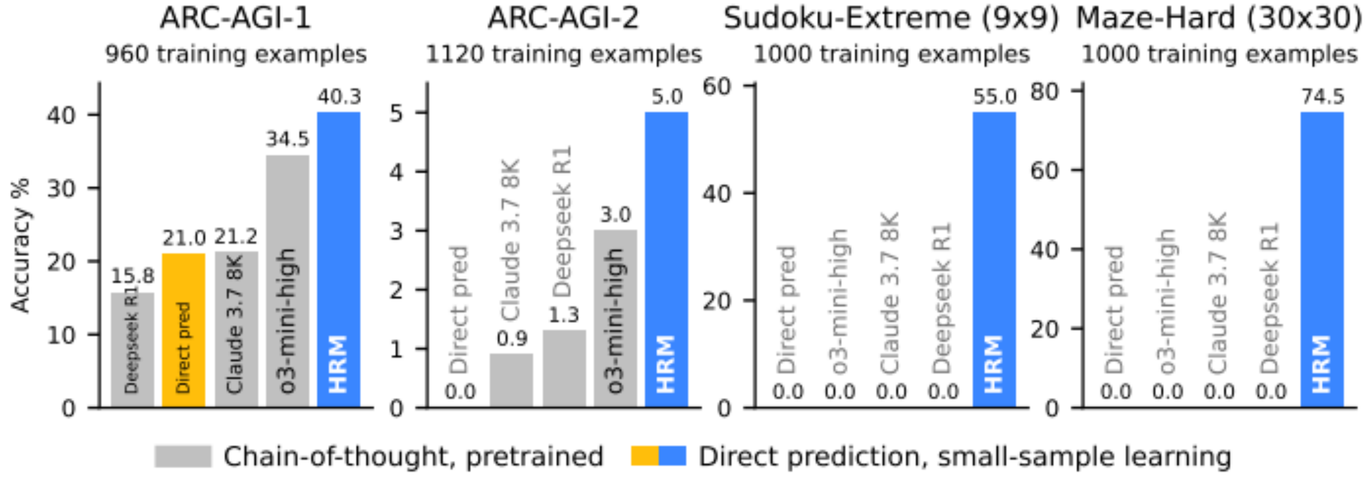
WTF... First they said no to Elon Musk.

Then they built an AI that just shook the entire LLM industry.

– 27M parameters
– Trained on only 1000 examples
– Inspired by the human brain
– Beat Claude, o3-mini & Deepseek

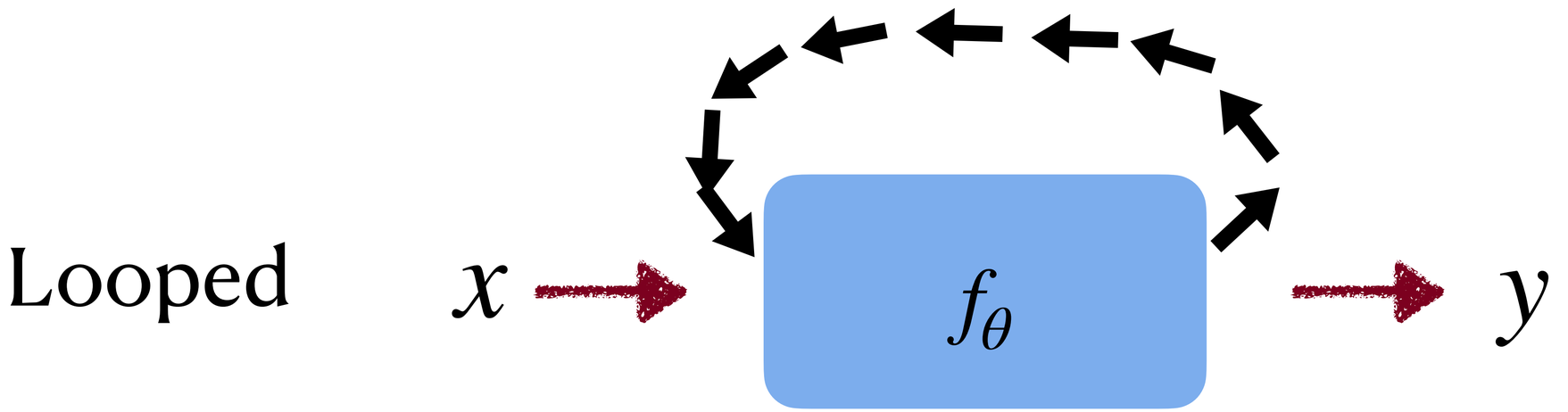
You can't afford to ignore this: 🧵

8:47 AM · Aug 1, 2025

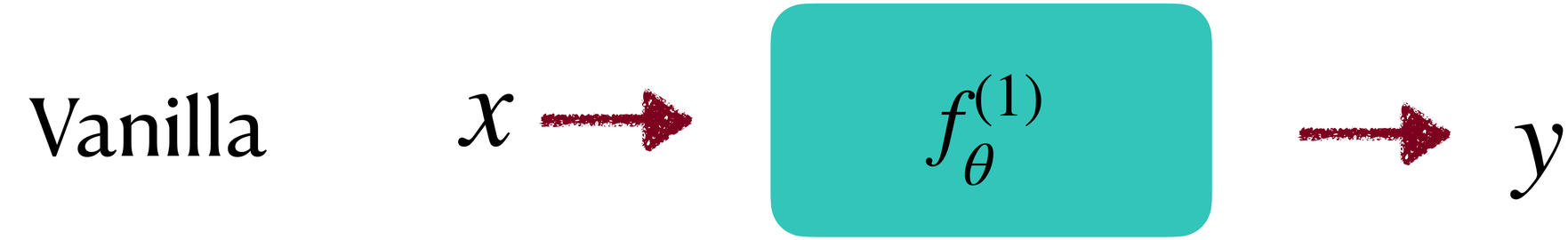


Looped transformers

Why looped transformers?



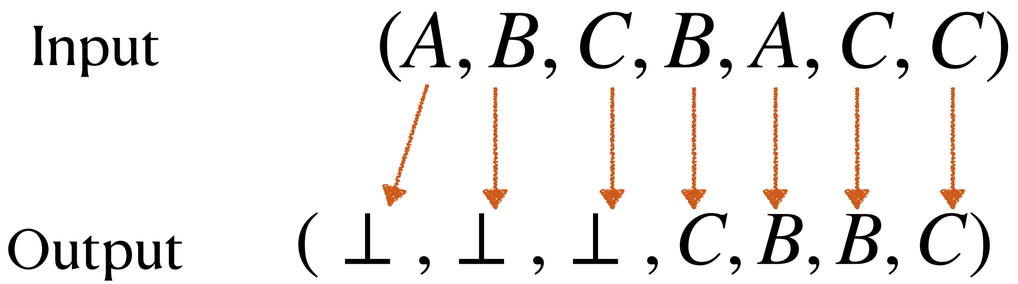
vs.



Induction head task

$$\Sigma := \{A, B, C\}$$

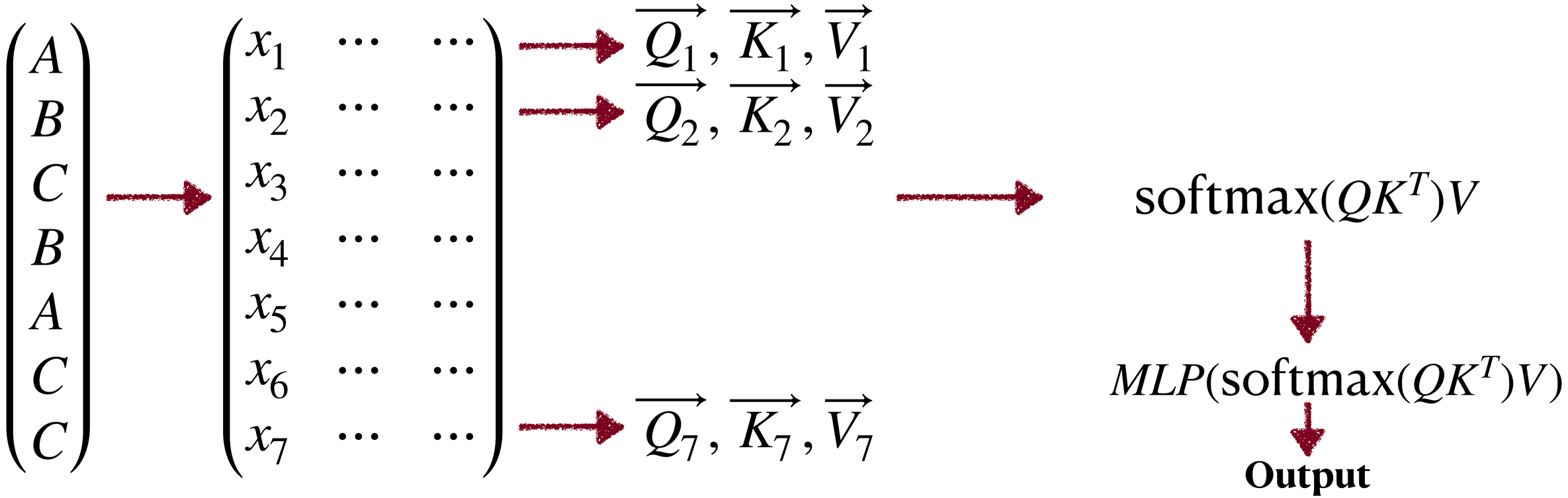
Take a sequence of alphabets:



- Look at previous occurrence.
- Output the token that follows the previous token.
- Output $\perp$ if there is no previous occurrence.

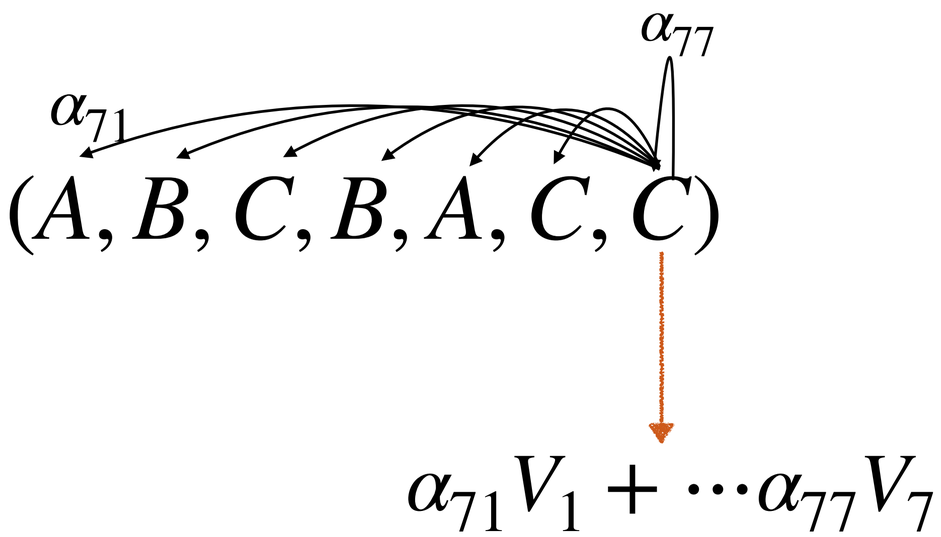
Transformer architecture

$$(A, B, C, B, A, C, C)$$

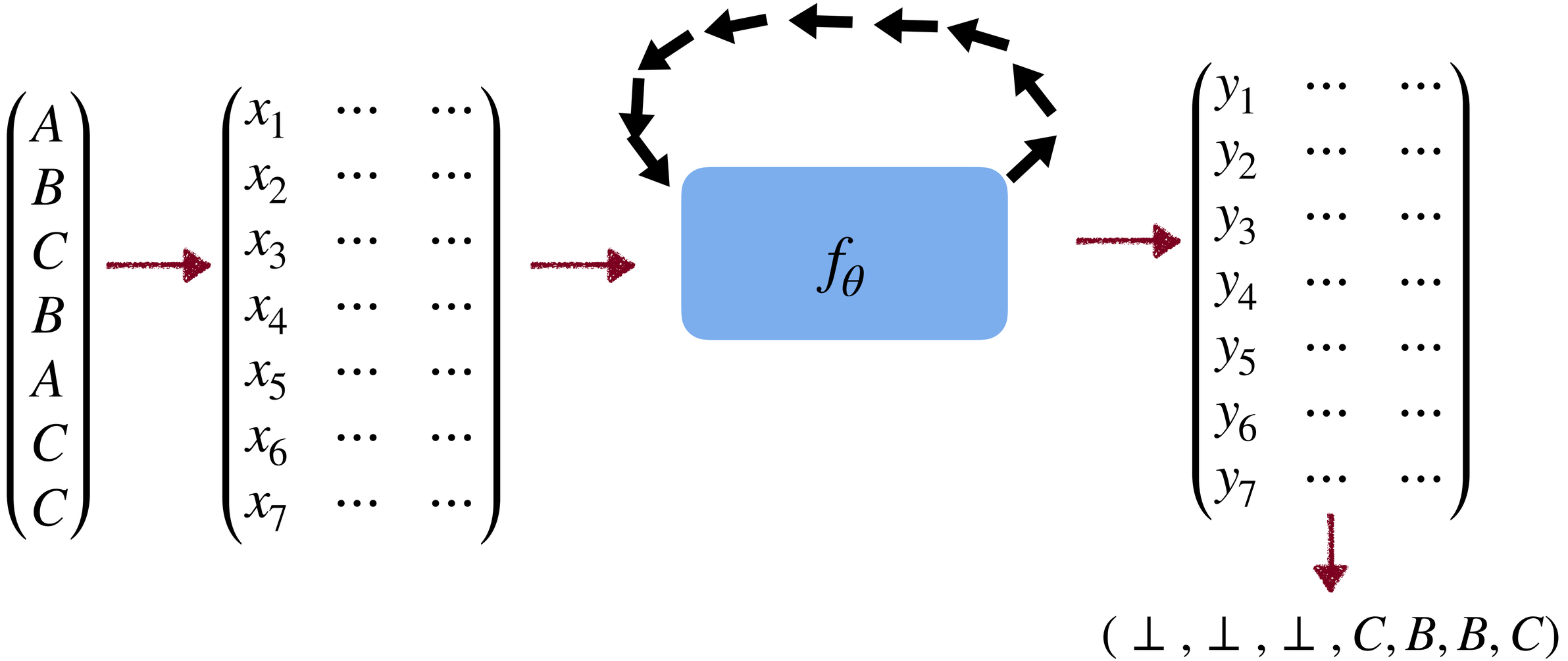


Attention is all we need

$$\alpha_{ij} = Q_i K_j^T$$



Loop transformer architecture



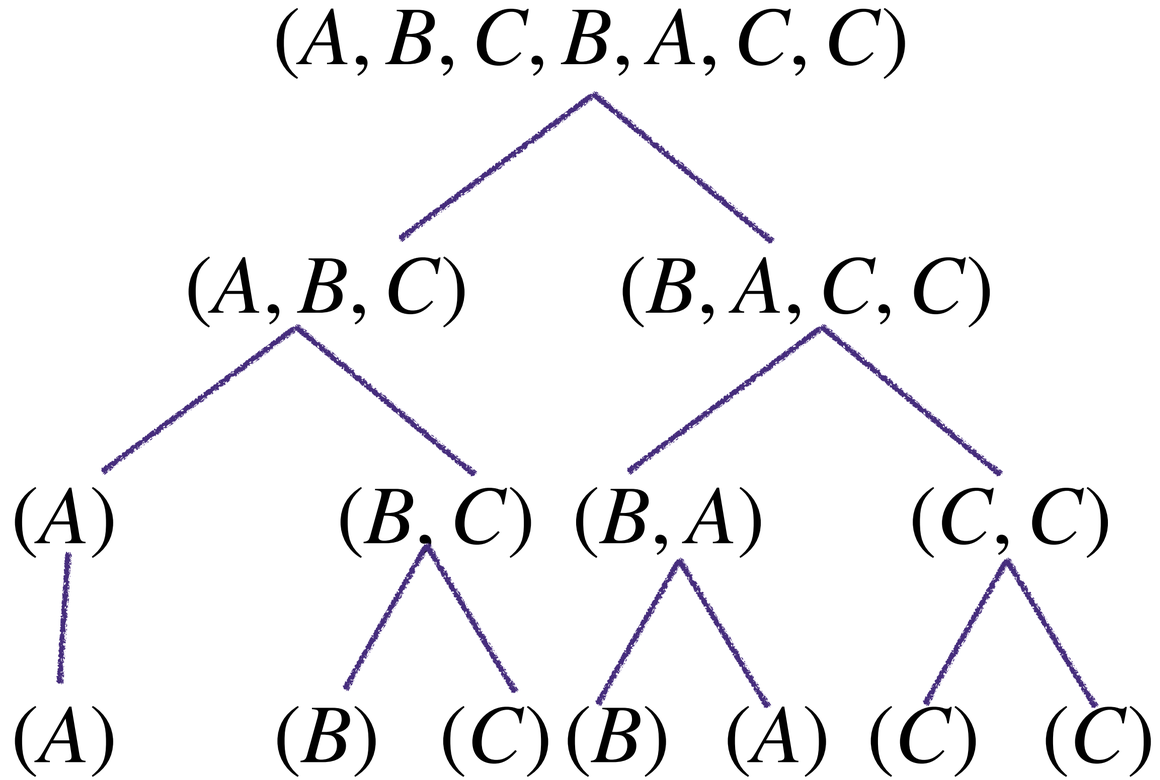
Theory first (mistake?)

GNN solution

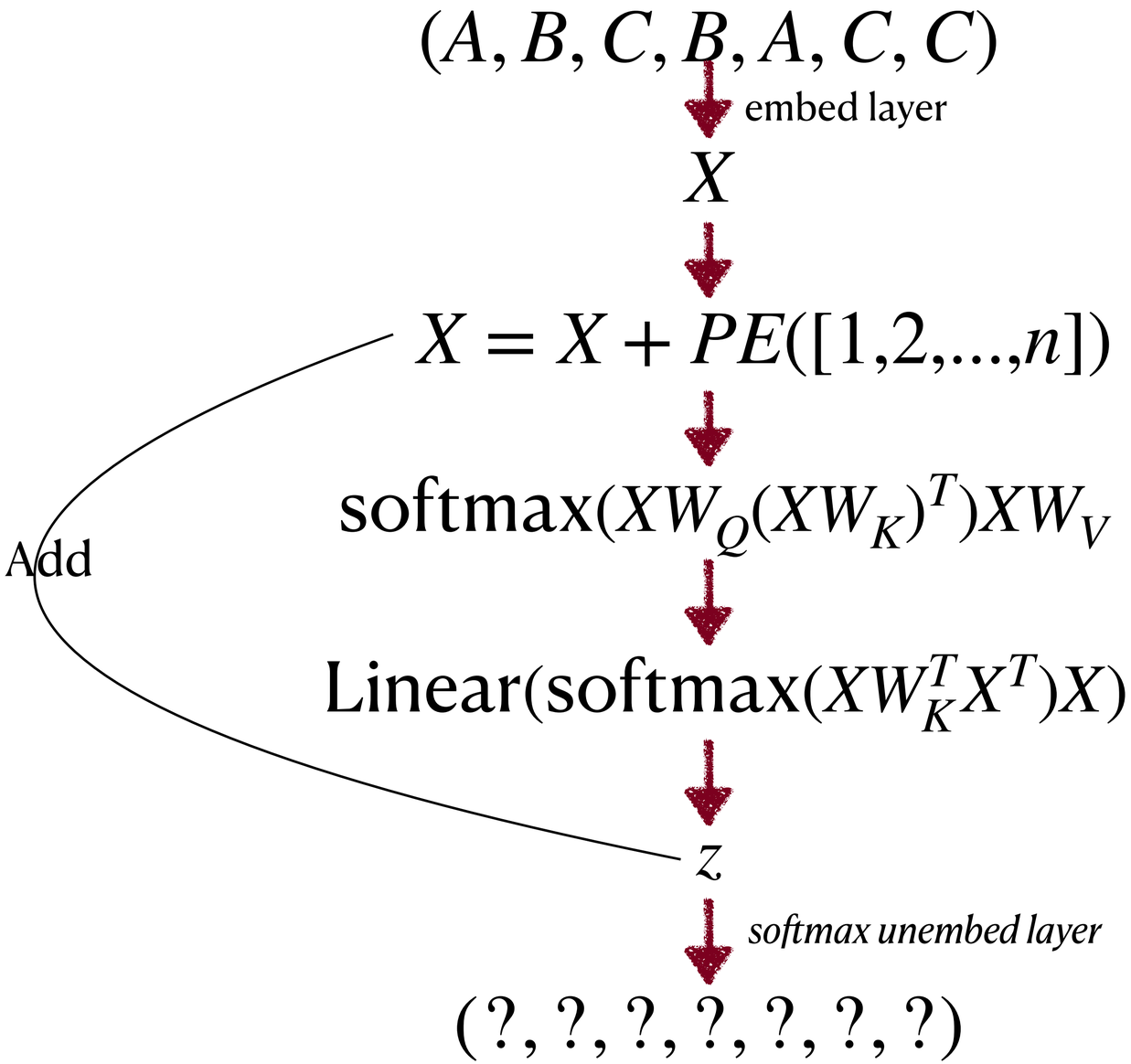
(A, B, C, B, A, C, C)

- Shift until you find a match.
- Path graph neural network

Tree solution



Experiment



$$PE : \mathbb{N} \to R^d$$

Identity → $W_Q : R^d \to R^d$

Trainable → $W_K : R^d \to R^d$

Identity → $W_V : R^d \to R^d$

Recursive equation

$$X, \quad z_0 = \vec{0}$$

Transformer module:

f_θ

$$z_1 = f_\theta(X + z_0)$$

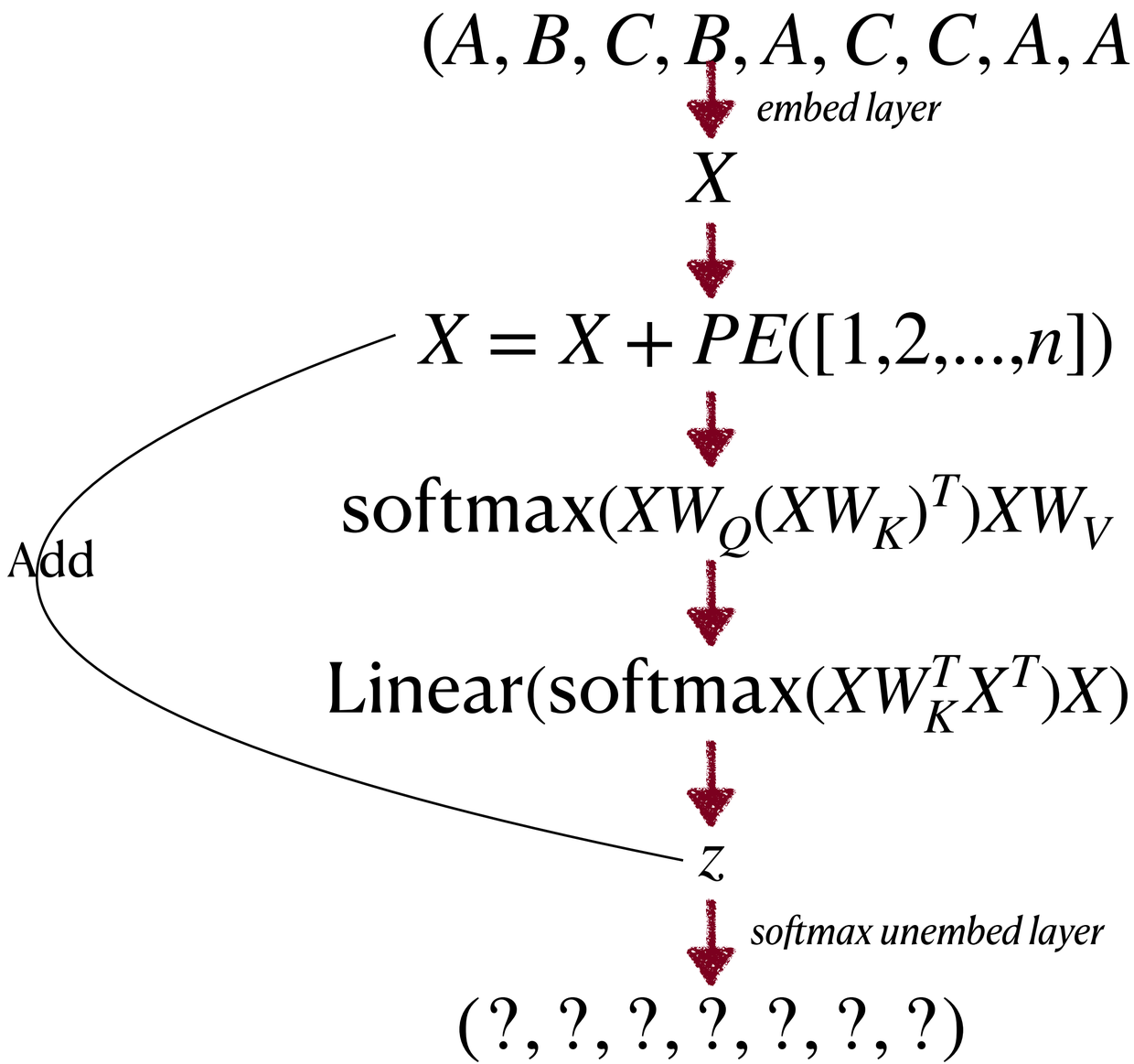
$$z_2 = f_\theta(X + z_1)$$

$$\dots \qquad \dots$$

$$z_k = f_\theta(X + z_{k-1})$$

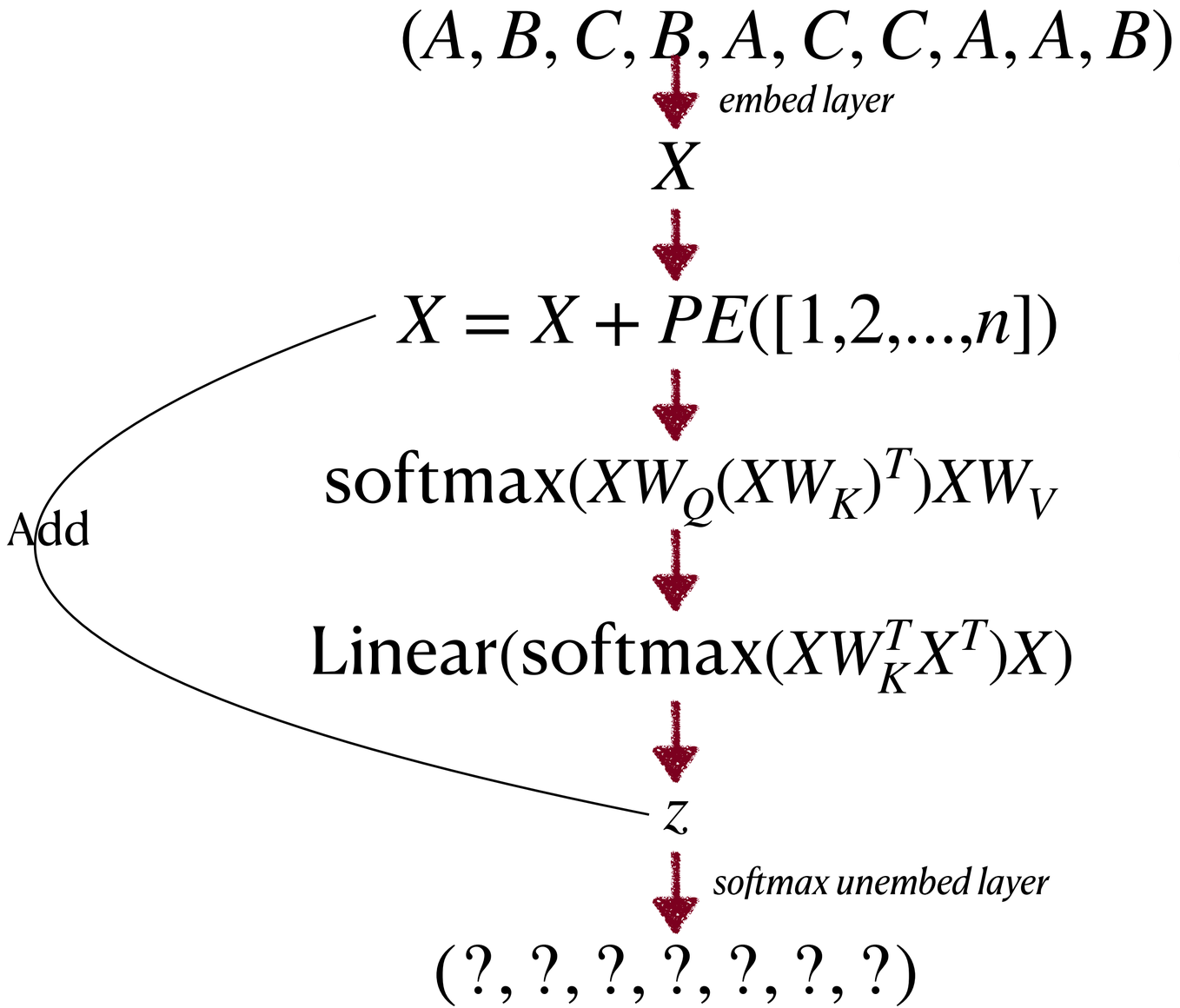
$$Y = \text{Decoder}(z_k)$$

Dataset

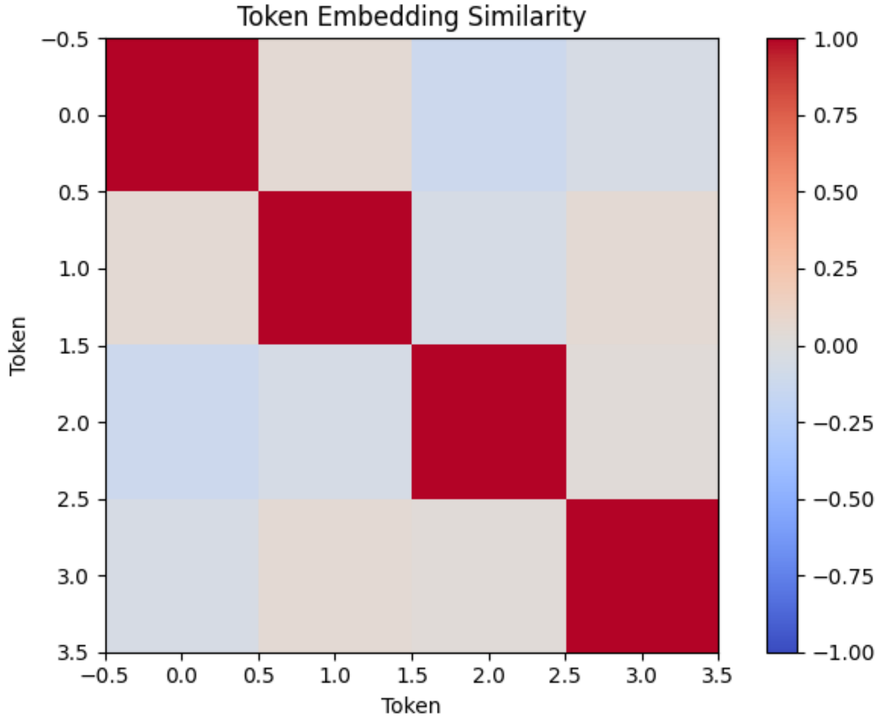


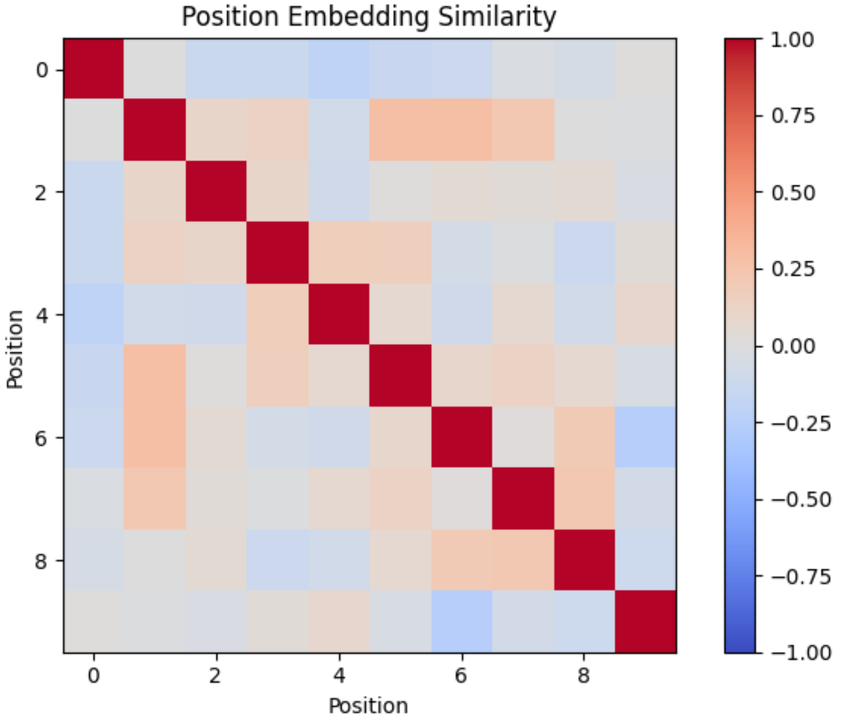
- Sequence length = 10
- Alphabet size = 3
- Generated 1000 random sequences and corresponding induction head target
- 80% - 20% train-test split

Hyperparameters



- Embedding dimension = 128
- Learning rate = 0.001
- Number of loops = 10
- Each embedding vector= random gaussian $\sim N(0, I_d)$

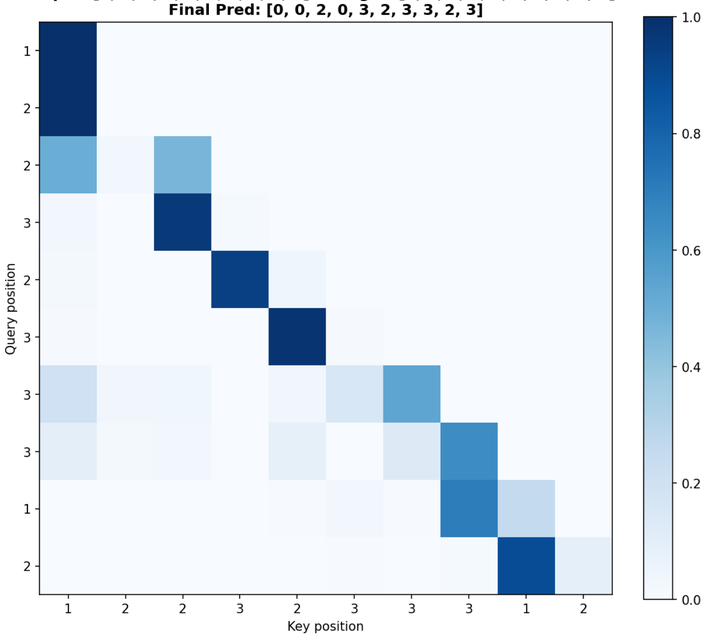

Token Embedding Similarity
Token
Token
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Position Embedding Similarity
Position
Position
0
2
4
6
8
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Epoch	Loss	Train Acc	Test Acc
10	0.6265	8.71%	9.60%
20	0.3384	27.90%	24.75%
30	0.1749	53.28%	48.48%
40	0.0889	79.55%	65.66%
50	0.0577	87.25%	73.23%
60	0.0429	86.74%	73.74%
70	0.0443	92.80%	80.81%
80	0.0343	85.48%	74.75%
90	0.0278	91.67%	76.77%
100	0.0235	95.33%	85.86%
110	0.0169	92.80%	82.83%
120	0.0101	95.71%	88.38%
130	0.0116	97.22%	93.43%
140	0.0091	95.71%	92.93%
150	0.0094	96.34%	90.40%
160	0.0064	98.86%	97.47%
170	0.0089	96.34%	95.96%
180	0.0046	99.49%	95.45%
190	0.0006	99.12%	95.96%
200	0.0005	99.87%	98.48%

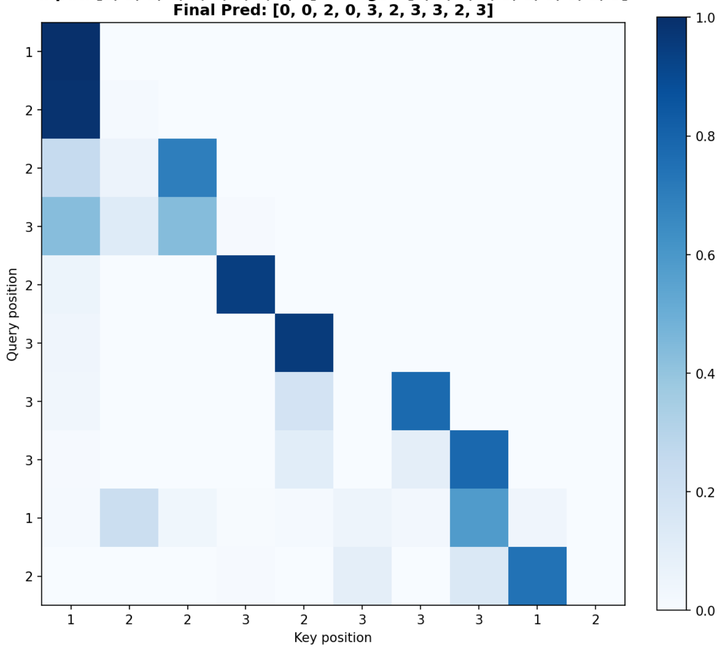
Iteration 1
z: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



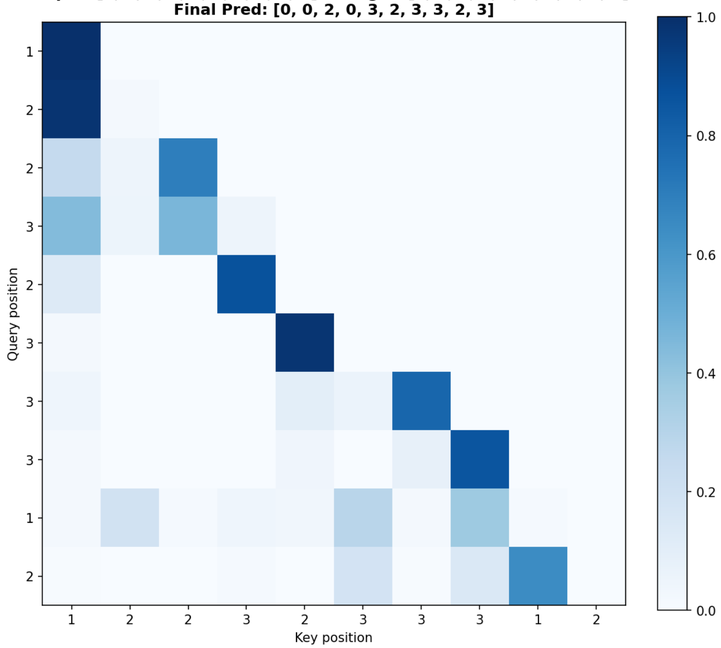
Iteration 2
z: [0, 0, 0, 2, 3, 2, 0, 3, 3, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



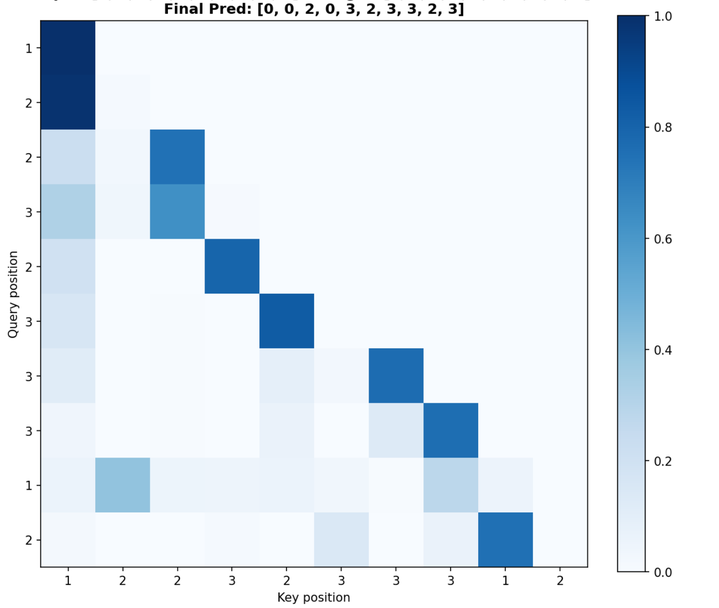
Iteration 3
z: [0, 0, 0, 0, 3, 2, 3, 3, 0, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



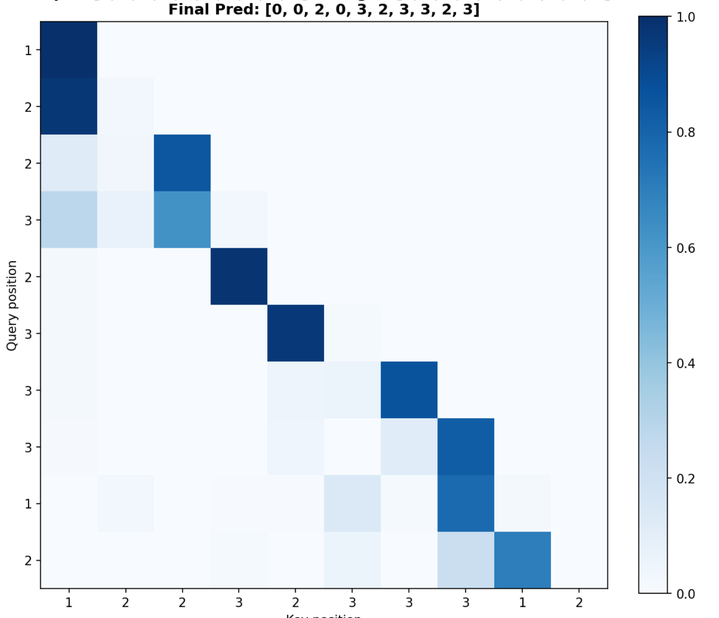
Iteration 4
z: [0, 0, 2, 0, 3, 2, 3, 3, 3, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



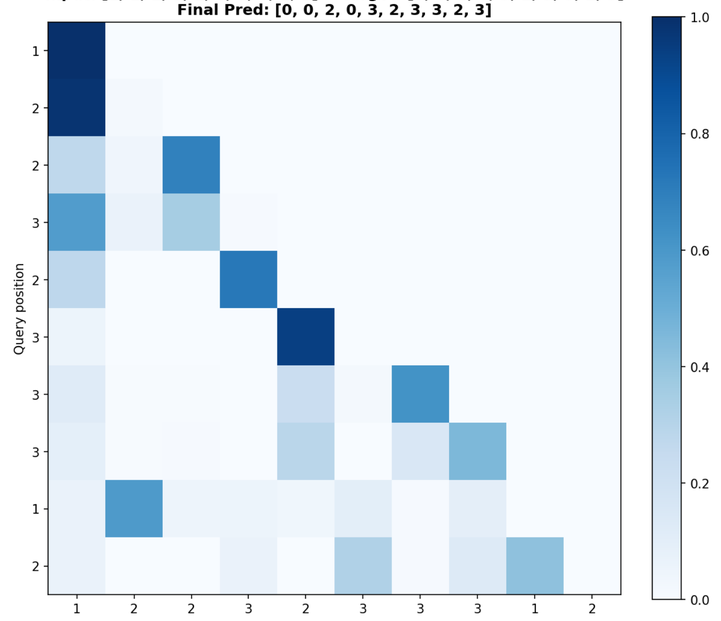
Iteration 5
z: [0, 0, 2, 2, 3, 2, 3, 3, 2, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



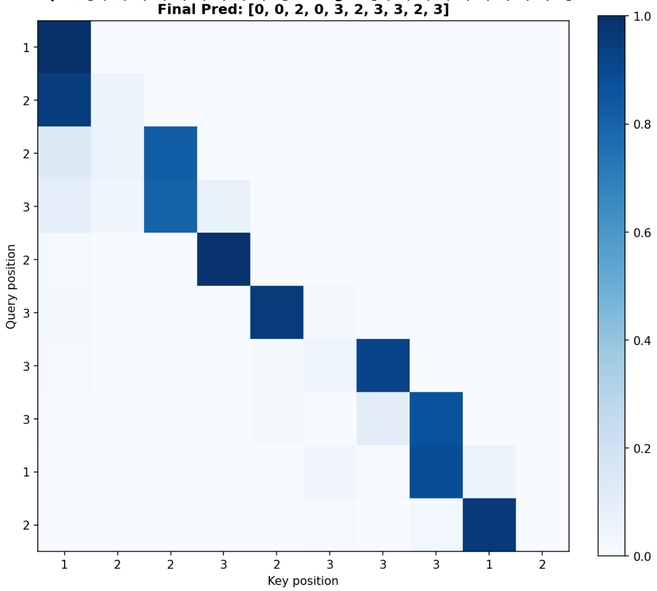
Iteration 6
z: [0, 0, 2, 0, 3, 2, 3, 3, 3, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



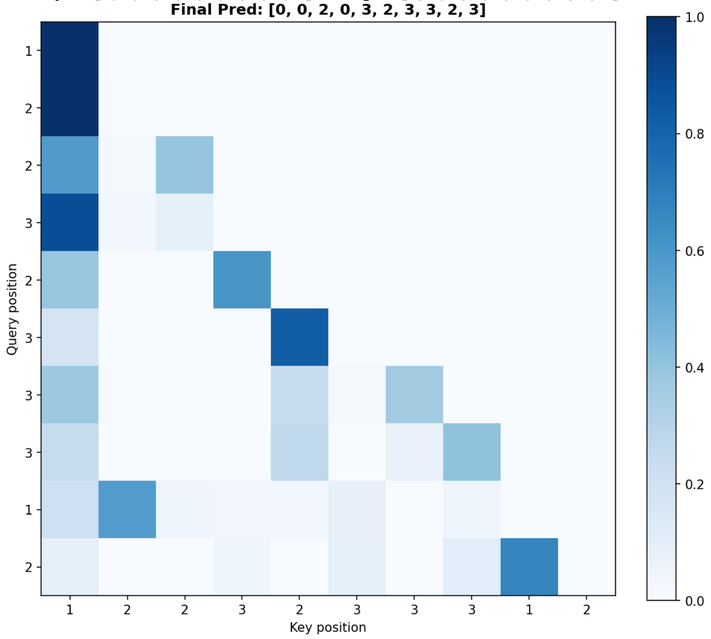
Iteration 7
z: [0, 0, 2, 0, 3, 2, 3, 3, 2, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



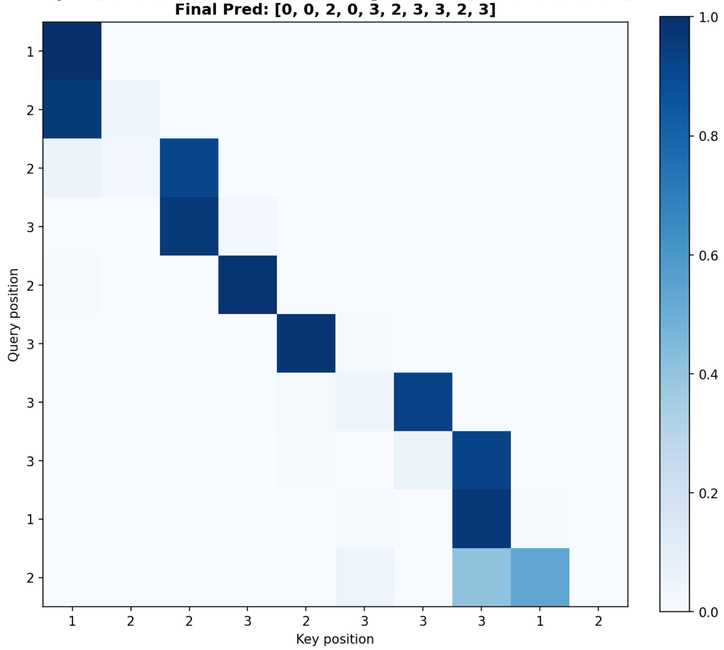
Iteration 8
z: [0, 0, 2, 2, 3, 2, 3, 3, 3, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



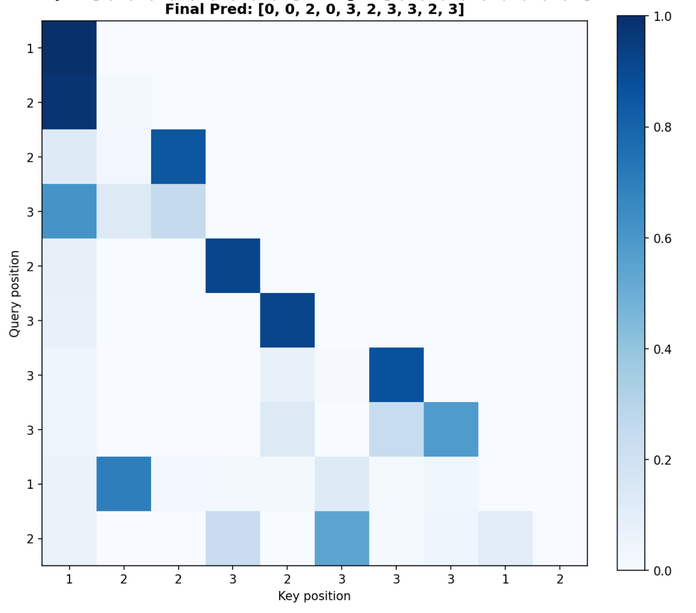
Iteration 9
z: [0, 0, 0, 0, 0, 2, 0, 0, 2, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]



Iteration 10
z: [0, 0, 2, 2, 3, 2, 3, 3, 3, 1]
x+z: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2]

Input: [1, 2, 2, 3, 2, 3, 3, 3, 1, 2] → Target: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]
Final Pred: [0, 0, 2, 0, 3, 2, 3, 3, 2, 3]

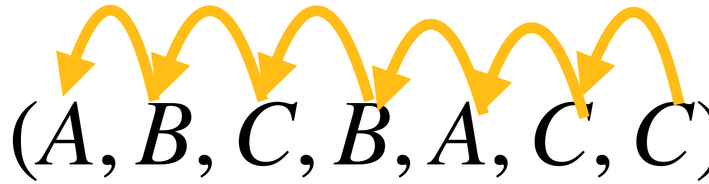


Copy mechanism heuristics

(A, B, C, B, A, C, C)

Copy mechanism heuristics

Copy mechanism heuristics



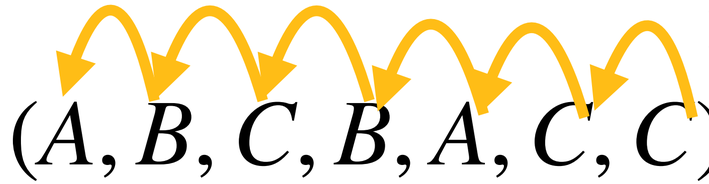
Look backwards

(A, B, C, B, A, C, C)

Copy the previous token

Previous token $\begin{pmatrix} \perp, A, B, C, B, A, C \\ A, B, C, B, A, C, C \end{pmatrix}$

Copy mechanism heuristics



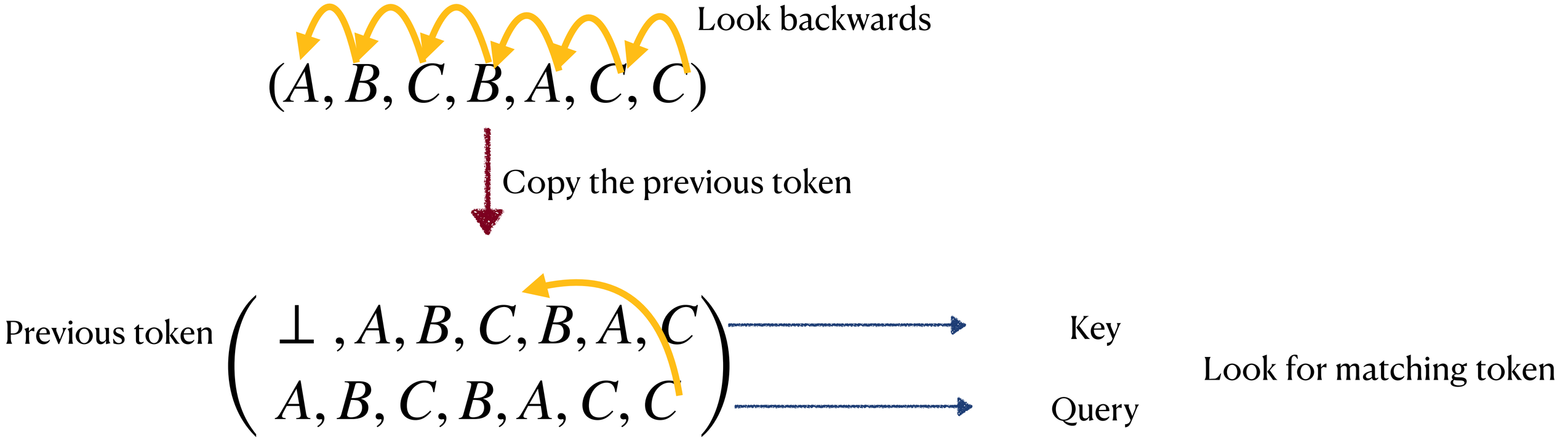
Look backwards

(A, B, C, B, A, C, C)

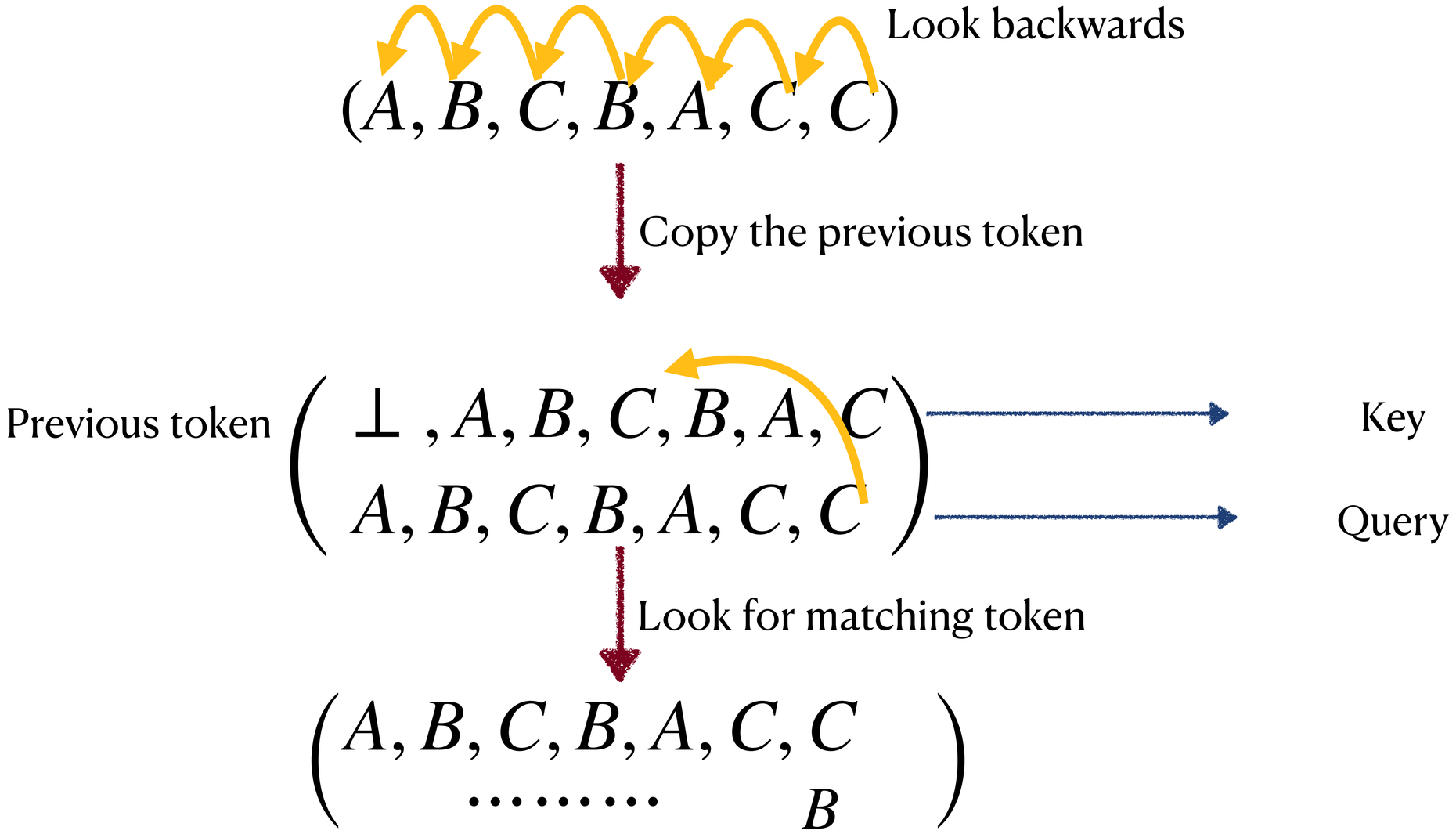
Copy the previous token

Previous token $\begin{pmatrix} \perp, A, B, C, B, A, C \\ A, B, C, B, A, C, C \end{pmatrix}$

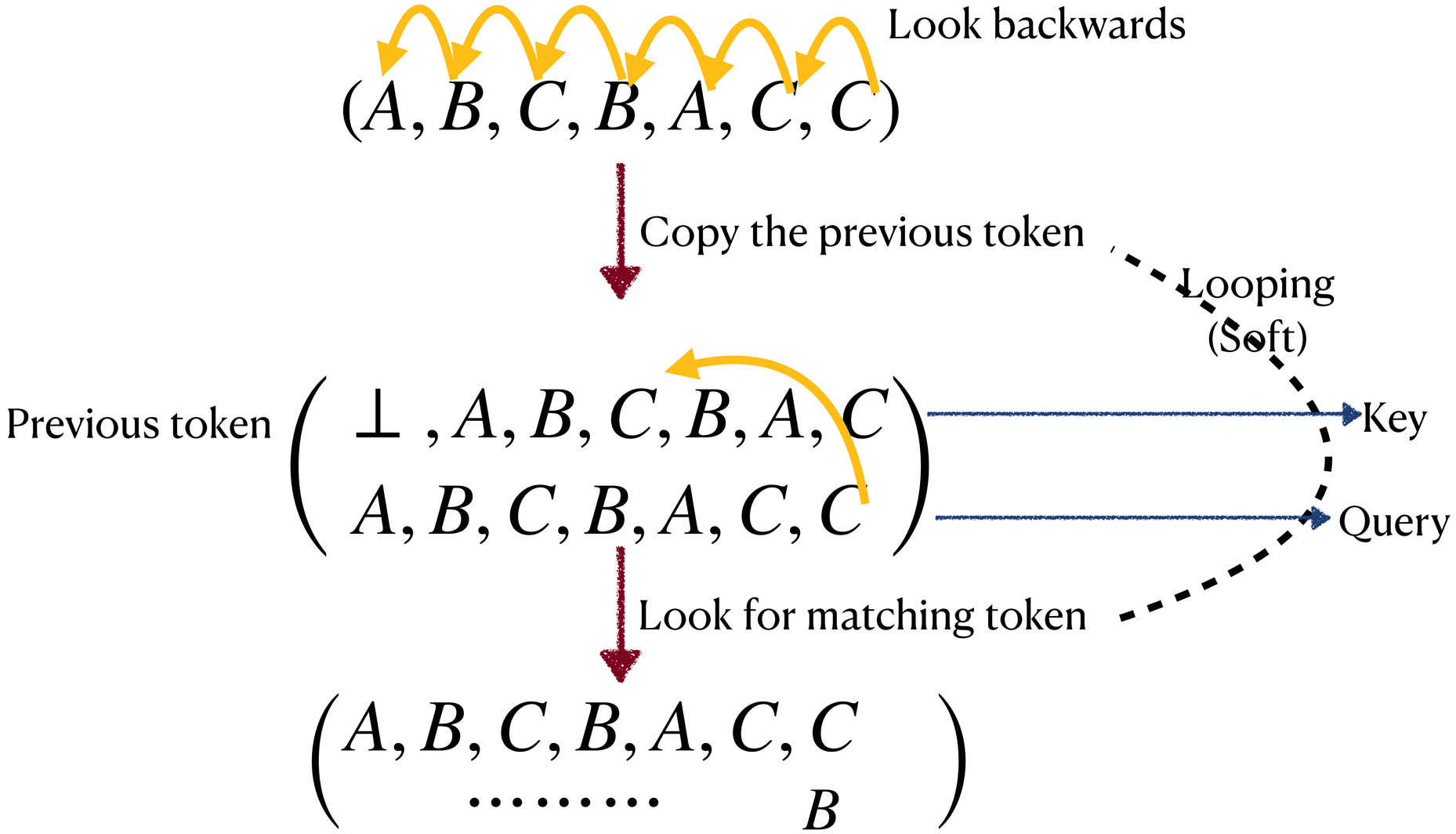
Copy mechanism heuristics



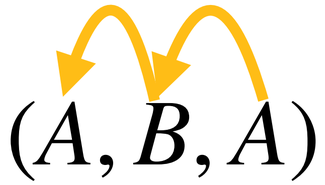
Copy mechanism heuristics



Copy mechanism heuristics



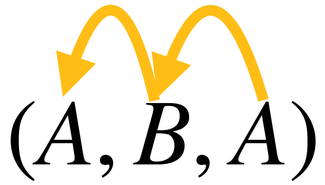
Heuristic vectorial explanation
For preceding position



$$\begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix} \end{matrix}, \begin{matrix} B \\ \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \end{matrix}, \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix} \end{matrix}$$

Previous token $\begin{pmatrix} A, B, A \\ \dots \quad B \end{pmatrix}$

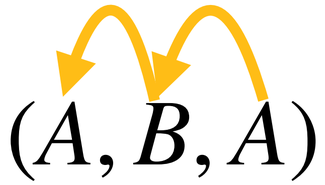
Heuristic vectorial explanation
For preceding position



$$\begin{array}{lccc} & A & B & A \\ \text{token information} & 1 & 0 & 0 \\ & 0 & 1 & 0 \\ \text{position information} & \cos(0) & \cos(\frac{\pi}{4}) & \cos(\frac{\pi}{2}) \\ & \sin(0) & \sin(\frac{\pi}{4}) & \sin(\frac{\pi}{2}) \end{array}$$

Previous token $\begin{pmatrix} A, B, A \\ \dots \quad B \end{pmatrix}$

Heuristic vectorial explanation
For preceding position

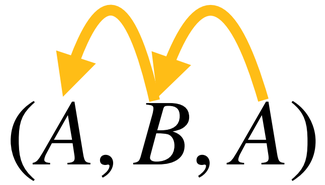


$$\overrightarrow{V} = \text{token information} \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix} \end{matrix}, \begin{matrix} B \\ \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \end{matrix}, \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix} \end{matrix}$$

$$\overrightarrow{Q} = \overrightarrow{K} = \text{position information}$$

Previous token $\begin{pmatrix} A, B, A \\ \dots \quad B \end{pmatrix}$

Heuristic vectorial explanation
For preceding position

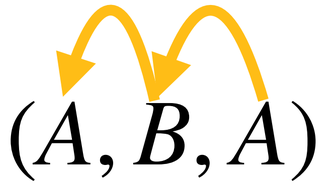


$$\overrightarrow{V} = \text{token information} \quad \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix} \end{matrix}, \begin{matrix} B \\ \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \end{matrix}, \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix} \end{matrix}$$

$$\overrightarrow{Q} = \overrightarrow{K} = \text{position information}$$

Previous token $\begin{pmatrix} A, B, A \\ \dots \quad B \end{pmatrix}$

Heuristic vectorial explanation
For preceding position



$$\overrightarrow{V} = \text{token information} \quad \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix} \end{matrix}, \begin{matrix} B \\ \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \end{matrix}, \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix} \end{matrix}$$

$$\overrightarrow{Q} = \overrightarrow{K} = \text{position information}$$

Previous token $\begin{pmatrix} A, B, A \\ \dots \quad B \end{pmatrix}$

$$\begin{bmatrix} 0 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix}$$

Heuristic vectorial explanation for matching tokens

$$\begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix} \end{matrix}, \begin{matrix} B \\ \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \end{matrix}, \begin{matrix} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix} \end{matrix}$$

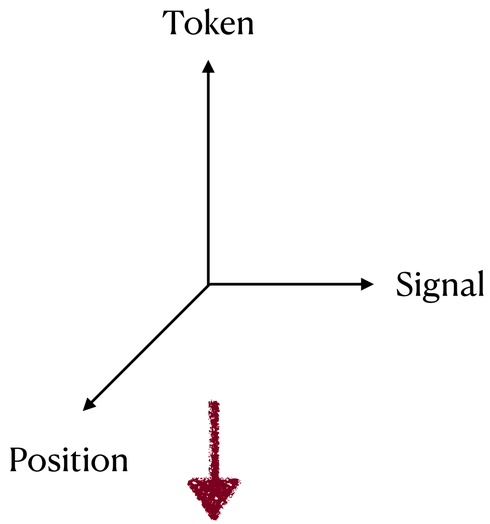
Heuristic vectorial explanation for matching tokens

$$\begin{array}{l} \\ \text{token information} \\ \text{position information} \end{array} \begin{array}{c} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix} \end{array}, \begin{array}{c} B \\ \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix} \end{array}, \begin{array}{c} A \\ \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix} \end{array}$$

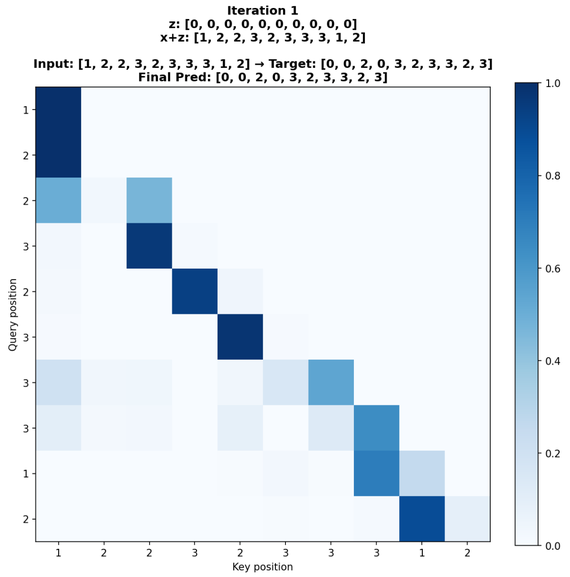
Heuristic vectorial explanation for matching tokens

$$\overrightarrow{Q} = \overrightarrow{K} = \begin{matrix} \text{token information} \\ \\ \text{position information} \end{matrix} \begin{bmatrix} 1 \\ 0 \\ \cos(0) \\ \sin(0) \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \cos(\frac{\pi}{4}) \\ \sin(\frac{\pi}{4}) \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ \cos(\frac{\pi}{2}) \\ \sin(\frac{\pi}{2}) \end{bmatrix}$$

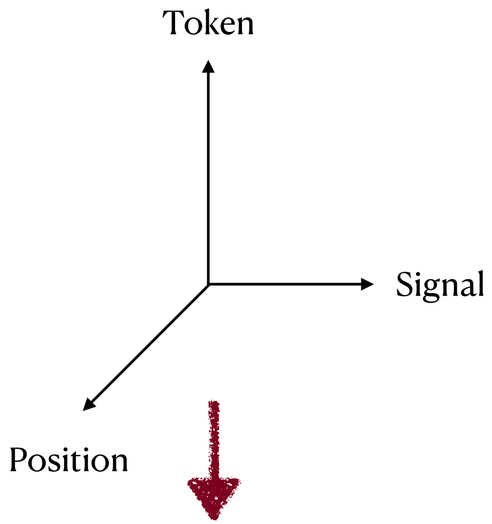
Transformation of the latent z

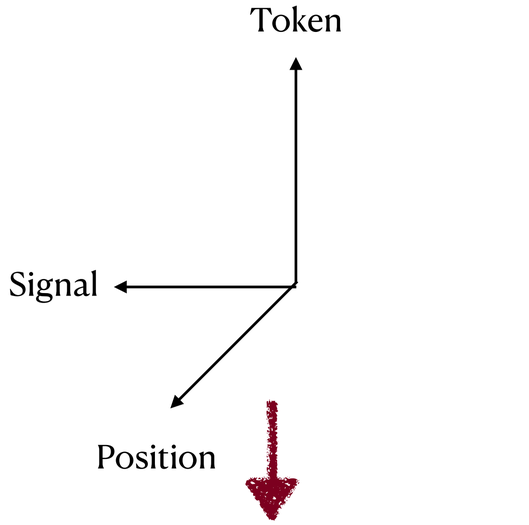


Look at previous token and copy it

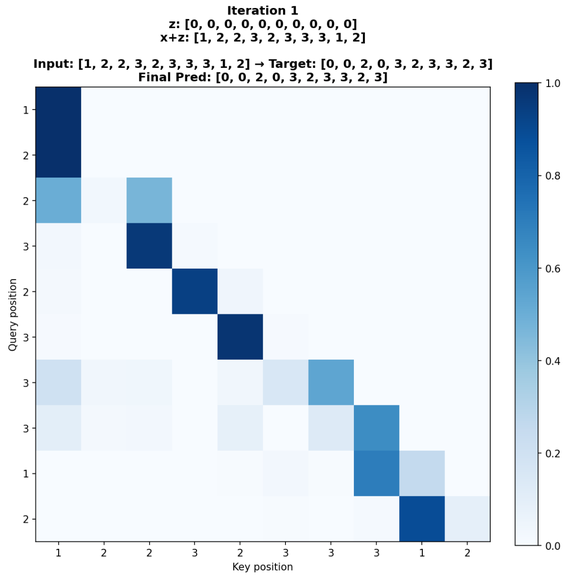


Transformation of the latent z

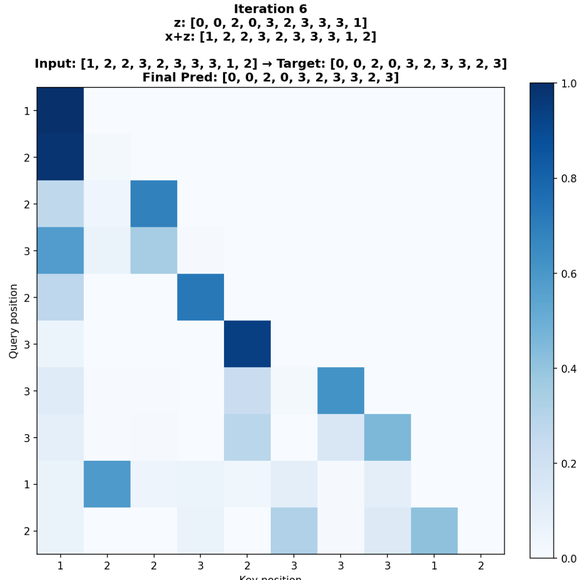




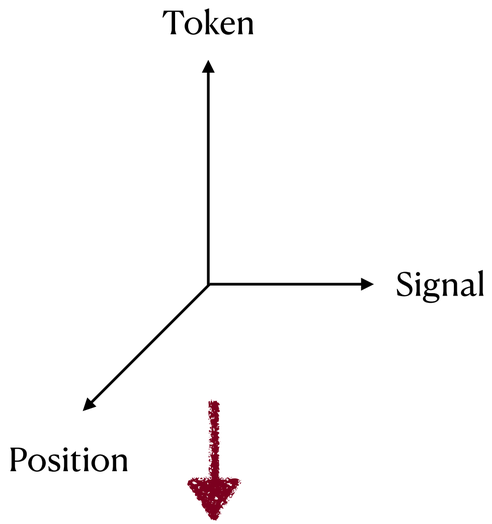
Look at previous token and copy it



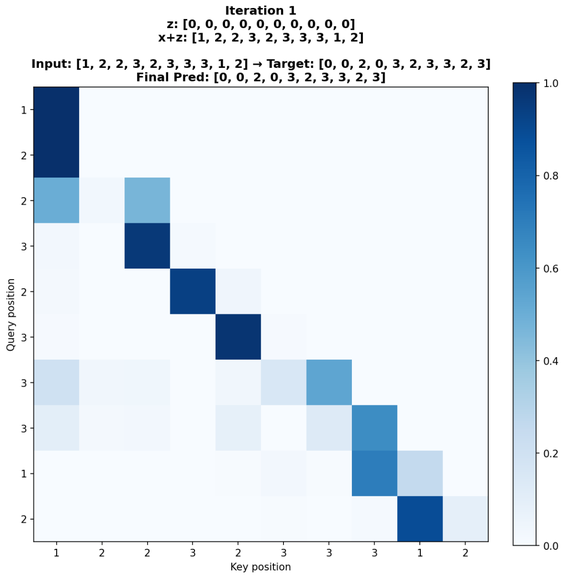
Look at previous occurrence (induction)

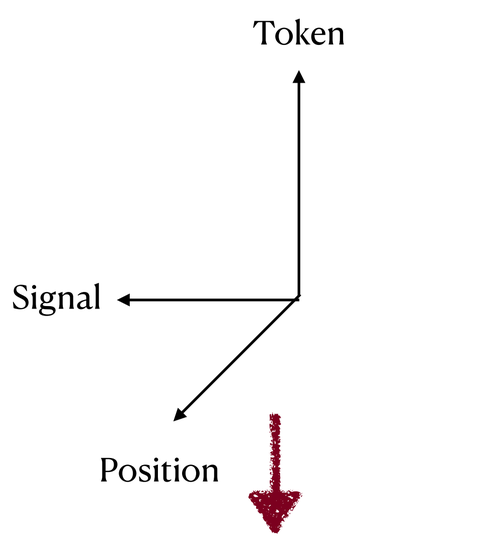


Transformation of the latent z

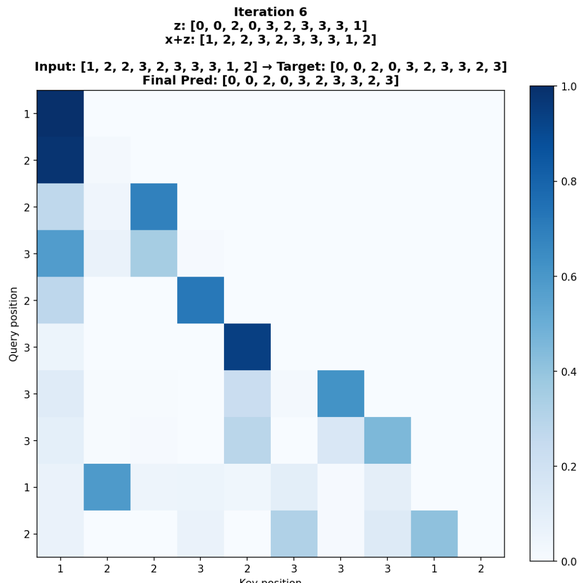


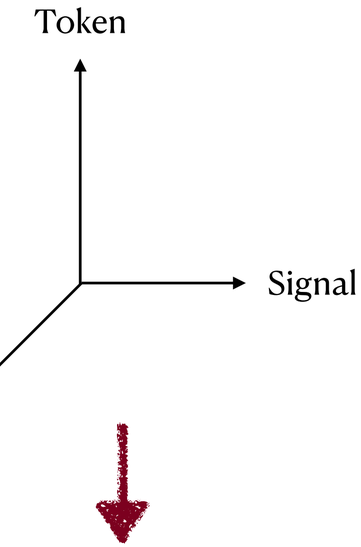
Look at previous token and copy it



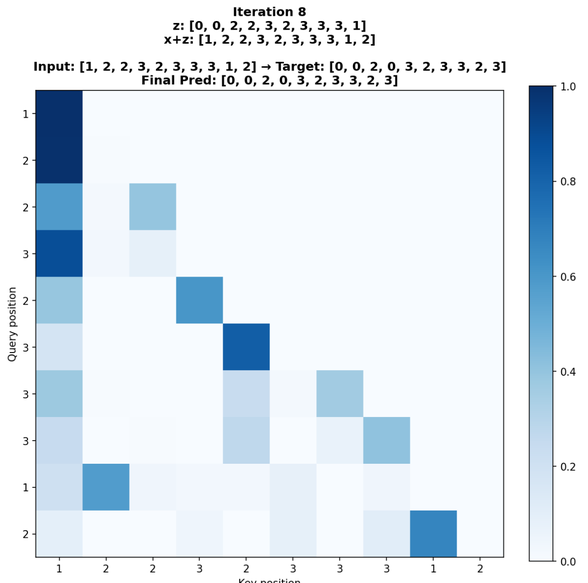


Look at previous occurrence (induction)

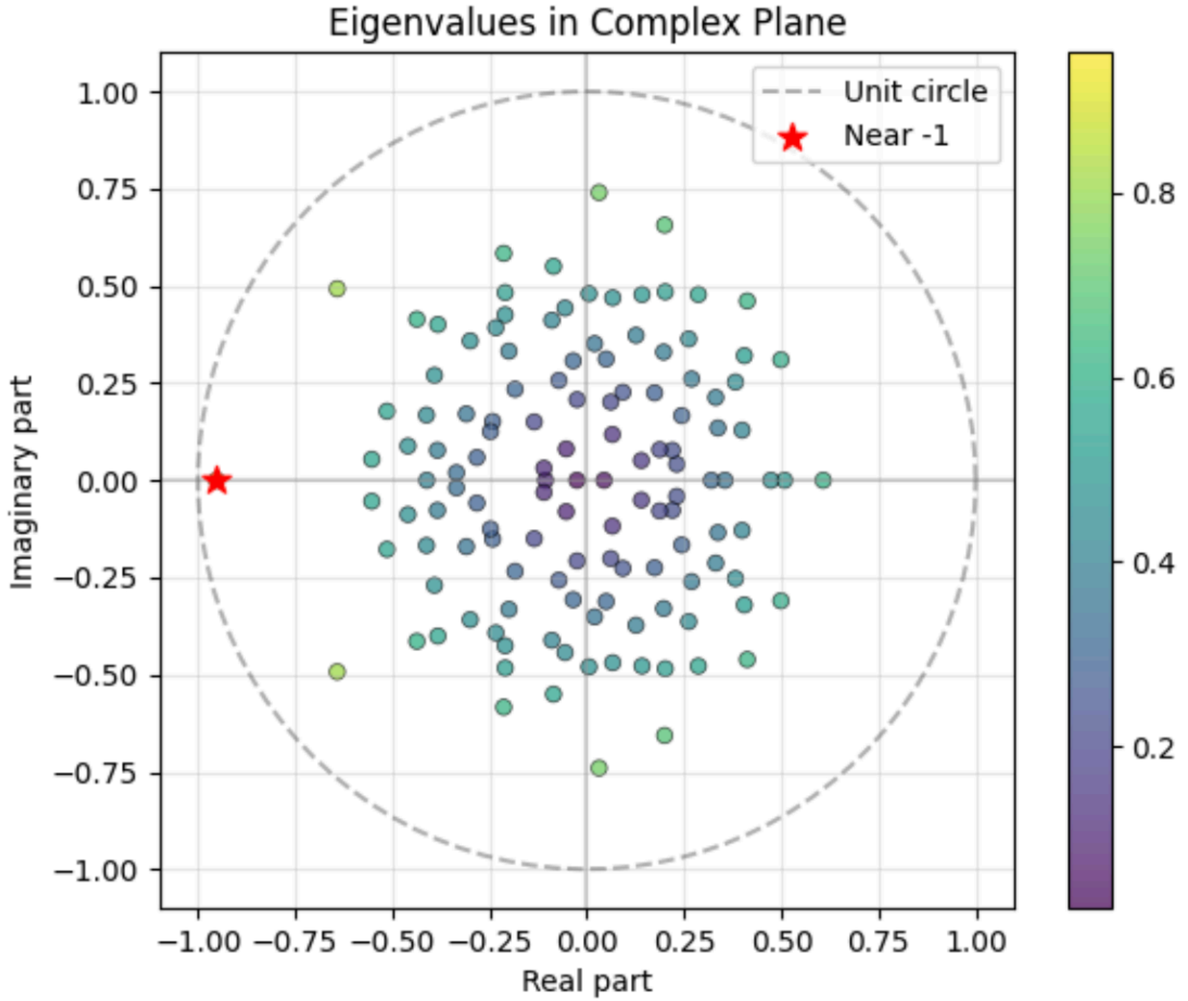




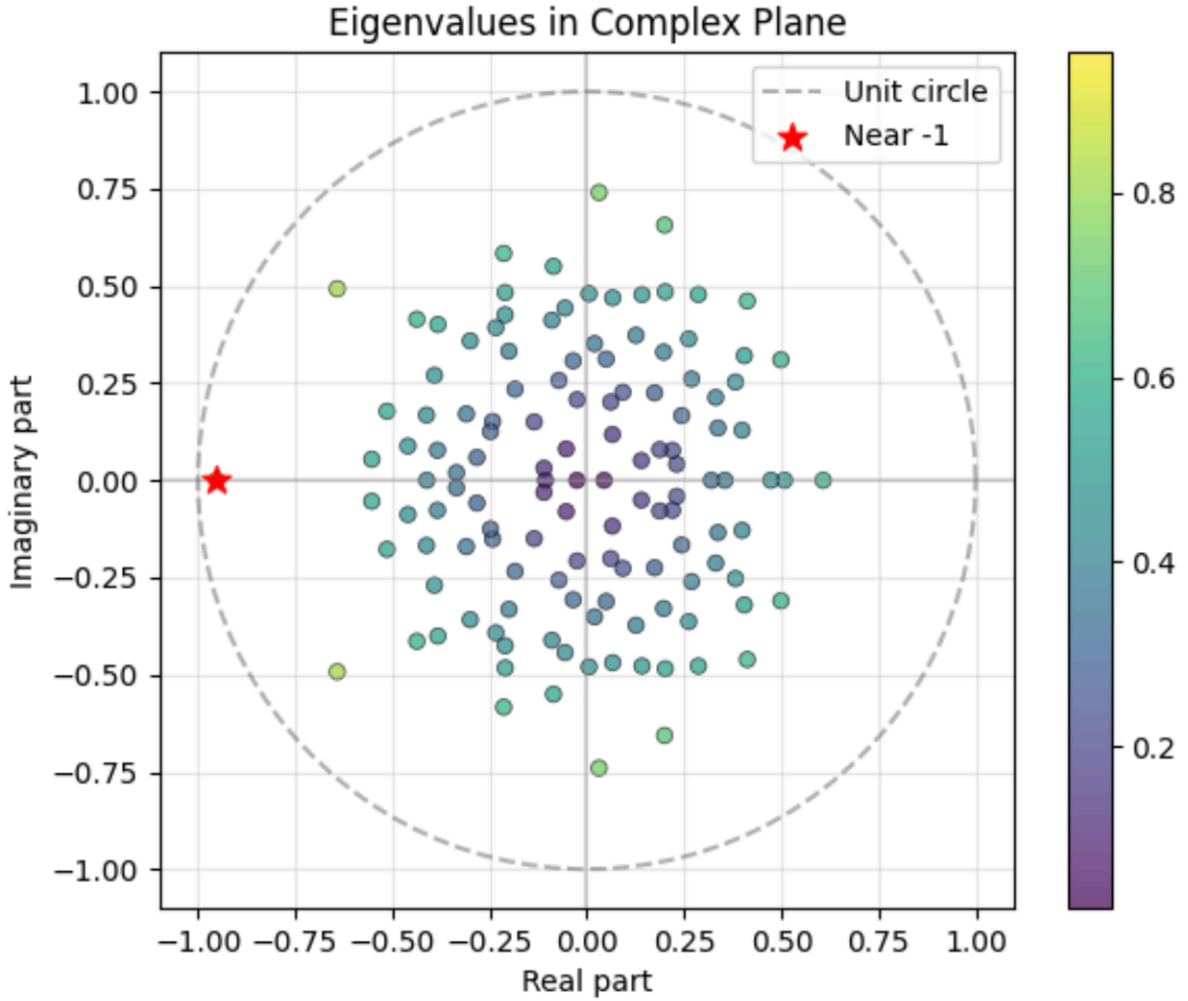
Look at previous token and copy it

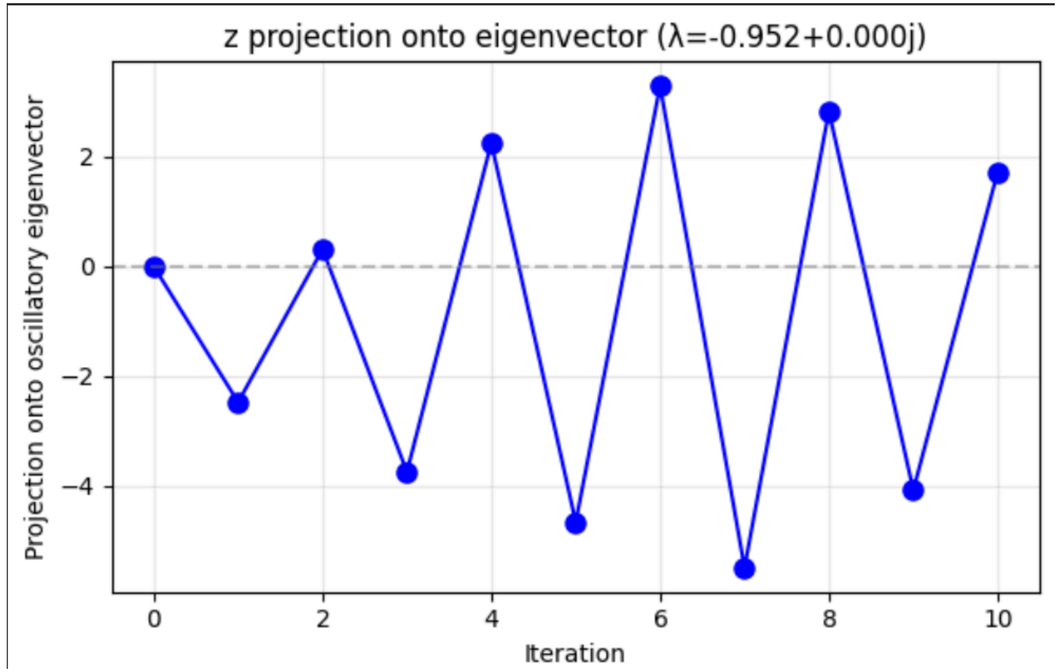


Feed forward layer



Feed forward layer





Conclusion

- Loop transformers come up with a clever way to solve the induction head task.

Ongoing work

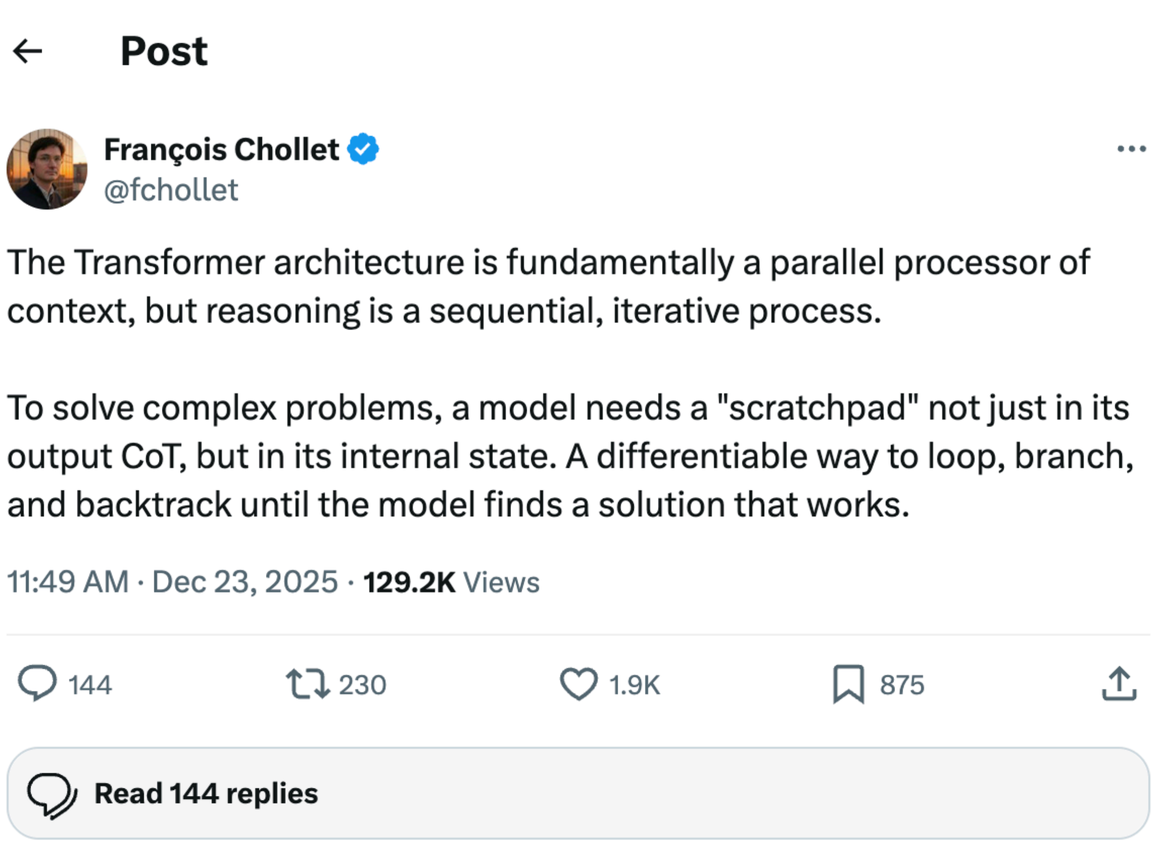
Conjecture: Looped transformers generalize better and are more sample efficient than vanilla transformers.

Idea: Look at loss landscapes of

$\sigma(W_2\sigma(W_1X + b_1) + b_2)$ vs. $\sigma(W\sigma(WX + b) + b)$

Vanilla Looped

- Need to make the role of the latent vector more rigorous.
- Formalize some ARC-AGI tasks and demonstrate how loop transformers solve them.


Post
François Chollet
@fchollet
The Transformer architecture is fundamentally a parallel processor of context, but reasoning is a sequential, iterative process.
To solve complex problems, a model needs a "scratchpad" not just in its output CoT, but in its internal state. A differentiable way to loop, branch, and backtrack until the model finds a solution that works.
11:49 AM · Dec 23, 2025 · 129.2K Views
144
230
1.9K
875
Read 144 replies

Thank you