Representational strengths and limitations of transformers

Daniel Hsu
Columbia University

Joint work with Clayton Sanford (Columbia) and Matus Telgarsky (NYU)

Yale S&DS Seminar
November 27, 2023

Motivation

Transformer [Vaswani et al, 2017]: self-attention nets used in large language models

Motivation

Transformer [Vaswani et al, 2017]: self-attention nets used in large language models

- Alternative to "classical" neural network architectures, e.g.,
  - fully-connected neural networks (FNNs)
  - convolutional neural networks (CNNs)
  - recurrent neural networks (RNNs)

Motivation

Transformer [Vaswani et al, 2017]: self-attention nets used in large language models

- Alternative to "classical" neural network architectures, e.g.,
  - fully-connected neural networks (FNNs)
  - convolutional neural networks (CNNs)
  - recurrent neural networks (RNNs)
- Amazing theoretical capabilities
  - Turing-completeness [Pérez, Barceló, Marinkovic, 2021; Wei, Chen, Ma, 2021; ...]
  - Recognize formal languages [Bhattamishra, Ahuja, Goyal, 2020; Hahn, 2020; Yao, Peng, Papadimitriou, Narasimhan, 2021; Hao, Angluin, Frank, 2022; Liu, Ash, Goel, Krishnamurthy, Zhang, 2022; Angluin, Chiang, Yang, 2023; ...]
  - Solve inference/learning problems ("in-context learning") [Garg, Tsipras, Liang, Valiant, 2022; Akyürek, Schuurmans, Andreas, Ma, Zhou, 2022; Zhang, Frei, Bartlett, 2023; Abernethy, Agarwal, Marinov, Warmuth, 2023; Bai, Chen, Wang, Xiong, Mei, 2023; ...]
  - ...

What is special about transformers?

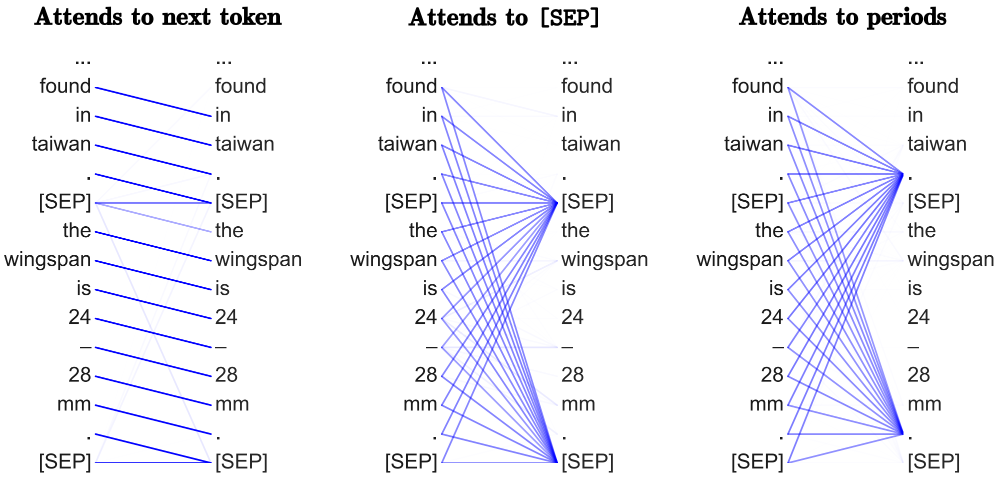
Transformers now underlie (many) learning-based used in NLP (and beyond)

What is special about transformers?

Transformers now underlie (many) learning-based used in NLP (and beyond)

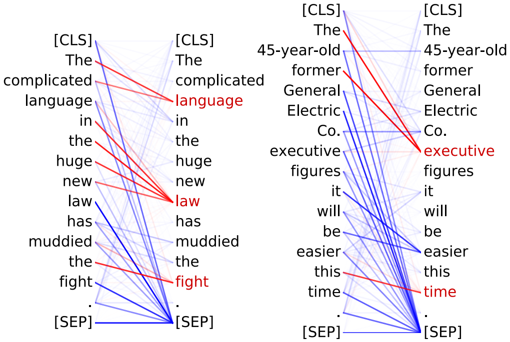
- Succinct parameterization of sequence-to-sequence functions (?)
- Ability to capture “long-range interactions” (?)

“What does BERT look at?” [Clark, Khandelwal, Levy, Manning, 2019]

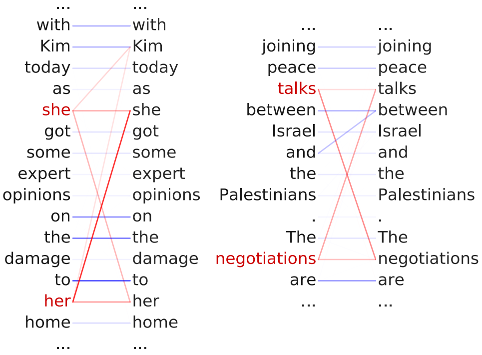


“What does BERT look at?” [Clark, Khandelwal, Levy, Manning, 2019]

- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation



- **Coreferent** mentions attend to their antecedents
- 65.1% accuracy at linking the head of a coreferent mention to the head of an antecedent



Our research questions

For which problems are transformers especially well/ill-suited?

Our research questions

For which problems are transformers especially well/ill-suited?

- How do transformers compare to classical architectures?

Our research questions

For which problems are transformers especially well/ill-suited?

- How do transformers compare to classical architectures?
- How does complexity scale with input size N?

Our research questions

For which problems are transformers especially well/ill-suited?

- How do transformers compare to classical architectures?
- How does complexity scale with input size N?
- This work:

$$\underbrace{N^{o(1)}}_{\text{“easy”}} \quad \text{vs.} \quad \underbrace{N^{\Omega(1)}}_{\text{“hard”}}$$

Our research questions

For which problems are transformers especially well/ill-suited?

- How do transformers compare to classical architectures?
- How does complexity scale with input size N?
- This work:

$$\underbrace{N^{o(1)}}_{\text{“easy”}} \quad \text{vs.} \quad \underbrace{N^{\Omega(1)}}_{\text{“hard”}}$$

What we do: Formalize advantages of transformers over classical architectures (as well as limitations of transformers) in terms of “communication” bottlenecks

Caveat

Results are only about **representational** strengths/limitations of transformers

(No direct analysis of learning/generalization)

Outline of talk

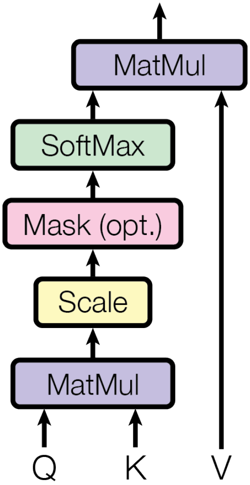
1. Transformers 101 + our results

2. Sparse Averaging

3. Element matching problems

1. Transformers 101 + our results

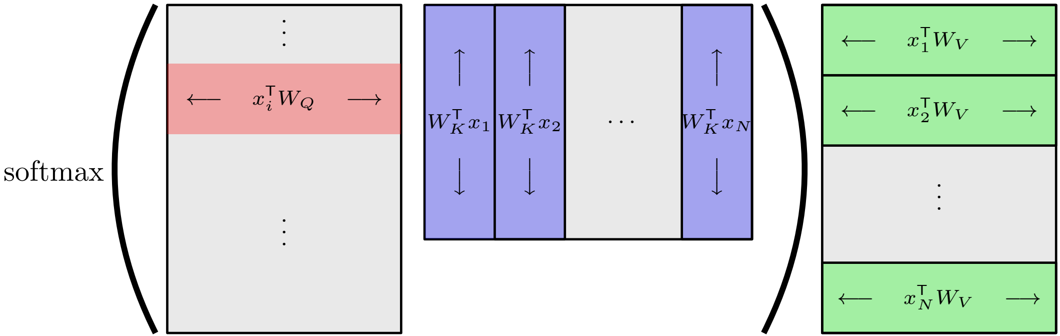
What is a self-attention unit?



[Vaswani et al, 2017]

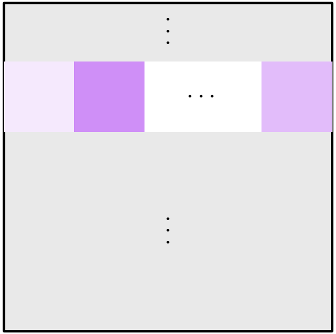
What is a self-attention unit?

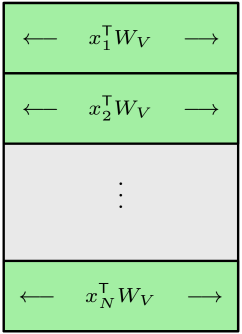
Self-attention unit: mapping of N-tuples from $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ to N-tuples from $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$ of a particular parametric form



What is a self-attention unit?

Self-attention unit: mapping of N-tuples from $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ to N-tuples from $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$ of a particular parametric form





What is a self-attention unit?

Self-attention unit: mapping of N-tuples from $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ to N-tuples from $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$ of a particular parametric form

$$\text{att}(X) = \text{softmax}\big((XW_Q)(XW_K)^\mathsf{T}\big)\, XW_V$$

- Parameters: $W_Q, W_K \in \mathbb{R}^{d_{\text{in}} \times m}, W_V \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ (query, key, & value params.)
- $m =$ (internal) embedding dimension
- softmax is applied row-wise:

$$\text{softmax}(M)_{i,j} = \frac{\exp(M_{i,j})}{\sum_{k=1}^{N} \exp(M_{i,k})}$$

What is a self-attention unit?

Self-attention unit: mapping of N-tuples from $\mathcal{X} = \mathbb{R}^{d_{\text{in}}}$ to N-tuples from $\mathcal{Y} = \mathbb{R}^{d_{\text{out}}}$ of a particular parametric form

$$\text{att}(X) = \text{softmax}\big((XW_Q)(XW_K)^{\mathsf{T}}\big)\, XW_V$$

- Parameters: $W_Q, W_K \in \mathbb{R}^{d_{\text{in}} \times m}, W_V \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ (query, key, & value params.)
- $m =$ (internal) embedding dimension
- softmax is applied row-wise:

$$\text{softmax}(M)_{i,j} = \frac{\exp(M_{i,j})}{\sum_{k=1}^{N} \exp(M_{i,k})}$$

- Mapping is permutation-equivariant
- Each row of $\text{att}(X)$ is in convex hull of $\{\text{rows of } XW_V\}$

What is a transformer?

- H-headed self-attention layer: sum of H self-attention units

$$X \mapsto \sum_{h=1}^{H} \mathrm{att}_{W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}}(X)$$

What is a transformer?

- *H-headed self-attention layer*: sum of H self-attention units

$$X \mapsto \sum_{h=1}^{H} \mathrm{att}_{W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}}(X)$$

- *Transformer*: composition of multi-headed self-attention layers

What is a transformer?

- H-headed self-attention layer: sum of H self-attention units

$$X \mapsto \sum_{h=1}^{H} \mathrm{att}_{W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}}(X)$$

- Transformer: composition of multi-headed self-attention layers
- Each self-attention unit is also allowed to process each element of input tuple using a feedforward neural network

$$\phi \colon \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}^{d'_{\mathrm{in}}}$$

(same ϕ is applied to each element of input tuple) with $d'_{\mathrm{in}} = O(m)$
- Can also process output tuple with some $\phi \colon \mathbb{R}^{d'_{\mathrm{out}}} \to \mathbb{R}^{d_{\mathrm{out}}}$, $d'_{\mathrm{out}} = O(m)$

What is a transformer?

- H-headed self-attention layer: sum of H self-attention units

$$X \mapsto \sum_{h=1}^{H} \mathrm{att}_{W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}}(X)$$

- Transformer: composition of multi-headed self-attention layers
- Each self-attention unit is also allowed to process each element of input tuple using a feedforward neural network

$$\phi \colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d'_{\text{in}}}$$

  (same ϕ is applied to each element of input tuple) with $d'_{\text{in}} = O(m)$
- Can also process output tuple with some $\phi \colon \mathbb{R}^{d'_{\text{out}}} \to \mathbb{R}^{d_{\text{out}}}$, $d'_{\text{out}} = O(m)$
- ϕ's are akin to "activation functions" in classical architectures

Our research questions

For which problems are transformers especially well/ill-suited?

- How do transformers compare to classical architectures?
- How does complexity scale with input size N?
- This work:

$$\underbrace{N^{o(1)}}_{\text{“easy”}} \quad \text{vs.} \quad \underbrace{N^{\Omega(1)}}_{\text{“hard”}}$$

Our research questions

For which problems are transformers especially well/ill-suited?

- How do transformers compare to classical architectures?
- How does complexity scale with input size N?
- This work:

$$\underbrace{N^{o(1)}}_{\text{“easy”}} \quad \text{vs.} \quad \underbrace{N^{\Omega(1)}}_{\text{“hard”}}$$

  - Allow element-wise maps ϕ to be arbitrary functions
  - How must “size” parameters L, H, m grow with N?

Our results

- On a sparse decoding problem: “Sparse Averaging”

- On element matching problems: “Pair/Triple Matching”

Our results

- On a sparse decoding problem: "Sparse Averaging"
  - Self-att. unit with $m = O(d_{\text{in}} + q \log N)$ suffices for sparsity level q[1]
  - Every FNN requires width $\Omega(N)$ even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$
  - Every RNN requires hidden state of $\Omega(N)$ bits even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$

- On element matching problems: "Pair/Triple Matching"

[1] Also (almost) matching lower bound

Our results

- On a sparse decoding problem: "Sparse Averaging"
  - Self-att. unit with $m = O(d_{\text{in}} + q \log N)$ suffices for sparsity level q[1]
  - Every FNN requires width $\Omega(N)$ even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$
  - Every RNN requires hidden state of $\Omega(N)$ bits even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$

- On element matching problems: "Pair/Triple Matching"[2]
  - (Standard) self-att. unit can solve Pair Matching with $m = O(d_{\text{in}})$
  - "Third-order" self-att. unit can solve Triple Matching with $m = O(d_{\text{in}})$

[1] Also (almost) matching lower bound

[2] Lower bounds assume W's and ϕ's use $\operatorname{poly} \log N$ bit precision numbers

Our results

- On a sparse decoding problem: "Sparse Averaging"
  - Self-att. unit with $m = O(d_{\text{in}} + q \log N)$ suffices for sparsity level q[1]
  - Every FNN requires width $\Omega(N)$ even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$
  - Every RNN requires hidden state of $\Omega(N)$ bits even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$

- On element matching problems: "Pair/Triple Matching"[2]
  - (Standard) self-att. unit can solve Pair Matching with $m = O(d_{\text{in}})$
  - "Third-order" self-att. unit can solve Triple Matching with $m = O(d_{\text{in}})$
  - Multi-headed self-att. layer requires $Hm = \tilde{\Omega}(N)$ to solve Triple Matching[3]

[1] Also (almost) matching lower bound
[2] Lower bounds assume W's and ϕ's use $\mathrm{poly}\log N$ bit precision numbers
[3] Also (almost) matching upper bound

Our results

- On a sparse decoding problem: "Sparse Averaging"
  - Self-att. unit with $m = O(d_{\text{in}} + q \log N)$ suffices for sparsity level q[1]
  - Every FNN requires width $\Omega(N)$ even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$
  - Every RNN requires hidden state of $\Omega(N)$ bits even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$

- On element matching problems: "Pair/Triple Matching"[2]
  - (Standard) self-att. unit can solve Pair Matching with $m = O(d_{\text{in}})$
  - "Third-order" self-att. unit can solve Triple Matching with $m = O(d_{\text{in}})$
  - Multi-headed self-att. layer requires $Hm = \tilde{\Omega}(N)$ to solve Triple Matching[3]
  - **Conjecture:** Transformer requires $LHm = \tilde{\Omega}(N^{\Omega(1)})$ to solve Triple Matching

[1] Also (almost) matching lower bound

[2] Lower bounds assume W's and ϕ's use $\text{poly} \log N$ bit precision numbers

[3] Also (almost) matching upper bound

Our results

- On a sparse decoding problem: "Sparse Averaging"
  - Self-att. unit with $m = O(d_{\text{in}} + q \log N)$ suffices for sparsity level q[1]
  - Every FNN requires width $\Omega(N)$ even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$
  - Every RNN requires hidden state of $\Omega(N)$ bits even if $q = 1$, $d_{\text{in}} = \tilde{O}(1)$

- On element matching problems: "Pair/Triple Matching"[2]
  - (Standard) self-att. unit can solve Pair Matching with $m = O(d_{\text{in}})$
  - "Third-order" self-att. unit can solve Triple Matching with $m = O(d_{\text{in}})$
  - Multi-headed self-att. layer requires $Hm = \tilde{\Omega}(N)$ to solve Triple Matching[3]
  - **Conjecture:** Transformer requires $LHm = \tilde{\Omega}(N^{\Omega(1)})$ to solve Triple Matching

[1] Also (almost) matching lower bound
[2] Lower bounds assume W's and ϕ's use $\operatorname{poly}\log N$ bit precision numbers
[3] Also (almost) matching upper bound

2. Sparse Averaging

q-Sparse Averaging (qSA)

Input: $(x_1, x_2, \ldots, x_N)$ where

$$x_i = (\mathrm{enc}(i), \mathrm{enc}(S_i), v_i) \in \mathbb{R}^{d_{\mathrm{in}}}, \qquad d_{\mathrm{in}} = O(d + (q+1)\log N),$$

and

$$\begin{aligned} 1, 2, \ldots, N \quad & \text{are the "keys"} \\ S_1, S_2, \ldots, S_N \in \binom{[N]}{q} \quad & \text{are the "queries"} \\ v_1, v_2, \ldots, v_N \in \mathbb{R}^d \quad & \text{are the "values" (with } \|v_i\| \leq 1) \end{aligned}$$

q-Sparse Averaging (qSA)

Input: $(x_1, x_2, \ldots, x_N)$ where

$$x_i = (\mathrm{enc}(i), \mathrm{enc}(S_i), v_i) \in \mathbb{R}^{d_{\mathrm{in}}}, \qquad d_{\mathrm{in}} = O(d + (q+1)\log N),$$

and

$$\begin{aligned} 1, 2, \ldots, N \quad & \text{are the "keys"} \\ S_1, S_2, \ldots, S_N \in \binom{[N]}{q} \quad & \text{are the "queries"} \\ v_1, v_2, \ldots, v_N \in \mathbb{R}^d \quad & \text{are the "values" (with } \|v_i\| \le 1) \end{aligned}$$

Output: N vectors in $\mathbb{R}^{d_{\mathrm{out}}}$ with $d_{\mathrm{out}} = d$, where ith output vector is

$$\approx \frac{1}{q} \sum_{j \in S_i} v_j$$

What we show ($q\mathrm{SA}$)

- Self-att. unit with $m = O(d_{\mathrm{in}} + q \log N)$ suffices for sparsity level q (+ almost matching lower bound)
- Every FNN requires width $\Omega(N)$ even if $q = 1$, $d_{\mathrm{in}} = \tilde{O}(1)$
- Every RNN requires hidden state of $\Omega(N)$ bits even if $q = 1$, $d_{\mathrm{in}} = \tilde{O}(1)$

Self-attention solution (overview)

Design $\phi\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d'_{\text{in}}}$, $W_Q, W_K \in \mathbb{R}^{d'_{\text{in}} \times m}$, $W_V \in \mathbb{R}^{m \times d_{\text{out}}}$ such that

$$\operatorname{softmax}((\phi(X)W_Q)(\phi(X)W_K)^\mathsf{T})_{i,j} \approx \begin{cases} 1/q & \text{if } S_i \ni j \\ 0 & \text{if } S_i \not\ni j \end{cases}$$

and

$$\phi(X)W_Q = \begin{bmatrix} \longleftarrow & w_{S_1}^\mathsf{T} & \longrightarrow \\ & \vdots & \\ \longleftarrow & w_{S_N}^\mathsf{T} & \longrightarrow \end{bmatrix}, \ (\phi(X)W_K)^\mathsf{T} = \begin{bmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_N \\ \downarrow & & \downarrow \end{bmatrix}, \ \phi(X)W_V = \begin{bmatrix} \longleftarrow & v_1^\mathsf{T} & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_N^\mathsf{T} & \longrightarrow \end{bmatrix}$$

Self-attention solution (overview)

Design $\phi\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d'_{\text{in}}}$, $W_Q, W_K \in \mathbb{R}^{d'_{\text{in}} \times m}$, $W_V \in \mathbb{R}^{m \times d_{\text{out}}}$ such that

$$\operatorname{softmax}((\phi(X)W_Q)(\phi(X)W_K)^\top)_{i,j} \approx \begin{cases} 1/q & \text{if } S_i \ni j \\ 0 & \text{if } S_i \not\ni j \end{cases}$$

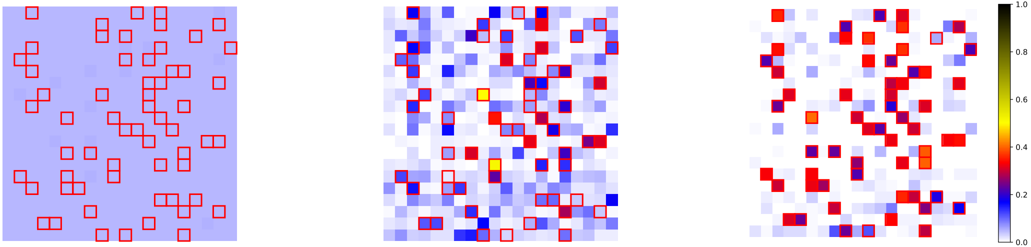
and

$$\phi(X)W_Q = \begin{bmatrix} \longleftarrow & w_{S_1}^\top & \longrightarrow \\ & \vdots & \\ \longleftarrow & w_{S_N}^\top & \longrightarrow \end{bmatrix}, \quad (\phi(X)W_K)^\top = \begin{bmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_N \\ \downarrow & & \downarrow \end{bmatrix}, \quad \phi(X)W_V = \begin{bmatrix} \longleftarrow & v_1^\top & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_N^\top & \longrightarrow \end{bmatrix}$$

ϕ will do most of the work; $\quad W_Q, W_K, W_V$ extract relevant parts of each $\phi(x_i)$

Empirical solution

"Attention matrices" $\mathrm{softmax}((\phi(X)W_Q)(\phi(X)W_K)^\intercal) \in \mathbb{R}^{20\times 20}$ for same fixed X, after training transformer for T epochs to solve $q\mathrm{SA}$ with $q = 3$



$T = 0$ $\quad$ $T = 1000$ $\quad$ $T = 40000$

Construction using q-neighborly 0/1 polytopes

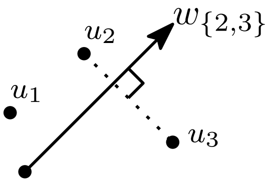
[Candès & Tao, 2005] There exist $u_1, u_2, \ldots, u_N \in \{\pm\frac{1}{\sqrt{k}}\}^k$ with $k = O(q \log N)$, such that, for every $S \in \binom{[N]}{q}$, there exists $w_S \in \mathbb{R}^k$ satisfying

$$\begin{aligned} \|w_S\| &\leq 2\sqrt{q} & \\ \langle w_S, u_j\rangle &= 1 & \text{for all } j \in S \\ |\langle w_S, u_j\rangle| &\leq 1/2 & \text{for all } j \notin S \end{aligned}$$

Construction using q-neighborly 0/1 polytopes

[Candès & Tao, 2005] There exist $u_1, u_2, \ldots, u_N \in \{\pm\frac{1}{\sqrt{k}}\}^k$ with $k = O(q \log N)$, such that, for every $S \in \binom{[N]}{q}$, there exists $w_S \in \mathbb{R}^k$ satisfying

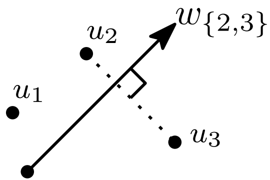
$$\|w_S\| \leq 2\sqrt{q}$$
$$\langle w_S, u_j \rangle = 1 \qquad \text{for all } j \in S$$
$$|\langle w_S, u_j \rangle| \leq 1/2 \qquad \text{for all } j \notin S$$



Construction using q-neighborly 0/1 polytopes

[Candès & Tao, 2005] There exist $u_1, u_2, \ldots, u_N \in \{\pm\frac{1}{\sqrt{k}}\}^k$ with $k = O(q \log N)$, such that, for every $S \in \binom{[N]}{q}$, there exists $w_S \in \mathbb{R}^k$ satisfying

$$\begin{aligned} \|w_S\| &\le 2\sqrt{q} \\ \langle w_S, u_j \rangle &= 1 \qquad \text{for all } j \in S \\ |\langle w_S, u_j \rangle| &\le 1/2 \qquad \text{for all } j \notin S \end{aligned}$$



Our $\phi\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d'_{\text{in}}}$ with $d'_{\text{in}} = O(d + q \log N)$ is

$$\phi(\text{enc}(i),\, \text{enc}(S_i),\, v_i) = (u_i,\, \alpha\, w_{S_i},\, v_i)$$

for suitably large $\alpha > 0$

(Simplified) lower bound for RNNs

Every RNN that computes $q\mathrm{SA}_N$ ($q = d = 1$) requires $\Omega(N)$ bit hidden state

(Simplified) lower bound for RNNs

Every RNN that computes $q\mathrm{SA}_N$ ($q = d = 1$) requires $\Omega(N)$ bit hidden state

- Reduction from **INDEX** problem ($n = N - 1$)
  Input: Alice has $a \in \{0,1\}^n$, Bob has $b \in [n]$
  Goal: After Alice sends Bob a message, Bob outputs a_b
  Pigeonhole principle lower bound: Alice must send at least n bits

(Simplified) lower bound for RNNs

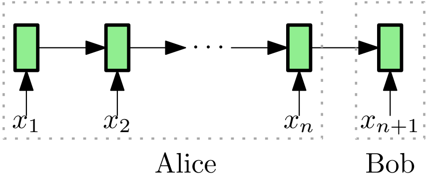
Every RNN that computes $q\mathrm{SA}_N$ ($q = d = 1$) requires $\Omega(N)$ bit hidden state

- Reduction from **INDEX** problem ($n = N - 1$)
  Input: Alice has $a \in \{0, 1\}^n$, Bob has $b \in [n]$
  Goal: After Alice sends Bob a message, Bob outputs a_b
  Pigeonhole principle lower bound: Alice must send at least n bits
- Consider any RNN that processes $(x_1, x_2, \ldots, x_{n+1})$ one element at a time and produces correct $(n + 1)$th output

(Simplified) lower bound for RNNs

Every RNN that computes $q\mathrm{SA}_N$ $(q = d = 1)$ requires $\Omega(N)$ bit hidden state

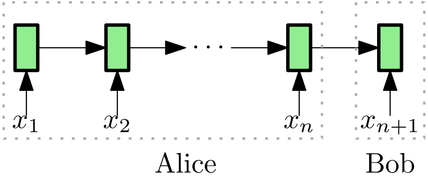
- Reduction from **INDEX** problem $(n = N - 1)$
  Input: Alice has $a \in \{0, 1\}^n$, Bob has $b \in [n]$
  Goal: After Alice sends Bob a message, Bob outputs a_b
  Pigeonhole principle lower bound: Alice must send at least n bits
- Consider any RNN that processes $(x_1, x_2, \ldots, x_{n+1})$ one element at a time and produces correct $(n + 1)$th output



(Simplified) lower bound for RNNs

Every RNN that computes $q\mathrm{SA}_N$ ($q = d = 1$) requires $\Omega(N)$ bit hidden state

- Reduction from **INDEX** problem ($n = N - 1$)
  Input: Alice has $a \in \{0,1\}^n$, Bob has $b \in [n]$
  Goal: After Alice sends Bob a message, Bob outputs a_b
  Pigeonhole principle lower bound: Alice must send at least n bits
- Consider any RNN that processes $(x_1, x_2, \ldots, x_{n+1})$ one element at a time and produces correct $(n+1)$th output



$$x_i = (\mathrm{enc}(i), \mathrm{enc}(\emptyset), a_i) \quad \text{for all } i \in [N]$$

(Alice sends nth hidden state to Bob)

$$x_{n+1} = (\mathrm{enc}(n+1), \mathrm{enc}(\{b\}), 0)$$

3. Element matching problems

Pair and Triple Matching

Input: $(x_0, x_1, x_2, \ldots, x_N)$ where

$$\text{dummy element:} \quad x_0 = \mathrm{enc}(\bot), \qquad \text{(for technical reasons)}$$
$$\text{for all } i \in [N]\text{:} \quad x_i = \mathrm{enc}(z_i), \qquad z_i \in \{1, 2, \ldots, M\}$$

and $N \ll M = \mathrm{poly}(N)$

Pair and Triple Matching

Input: $(x_0, x_1, x_2, \ldots, x_N)$ where

$$\begin{aligned} \text{dummy element:} \quad & x_0 = \mathrm{enc}(\perp), & \text{(for technical reasons)} \\ \text{for all } i \in [N]\text{:} \quad & x_i = \mathrm{enc}(z_i), & z_i \in \{1, 2, \ldots, M\} \end{aligned}$$

and $N \ll M = \mathrm{poly}(N)$

(Pair Matching) Output: ith output is

$$\mathbb{1}\{\exists j \in [N] \quad \text{s.t.} \quad z_i + z_j = 0 \pmod M\}$$

Pair and Triple Matching

Input: $(x_0, x_1, x_2, \ldots, x_N)$ where

$$\text{dummy element: } \quad x_0 = \mathrm{enc}(\bot), \qquad \text{(for technical reasons)}$$
$$\text{for all } i \in [N]\text{: } \quad x_i = \mathrm{enc}(z_i), \qquad z_i \in \{1, 2, \ldots, M\}$$

and $N \ll M = \mathrm{poly}(N)$

(Pair Matching) Output: ith output is

$$\mathbb{1}\{\exists j \in [N] \quad \text{s.t.} \quad z_i + z_j = 0 \pmod{M}\}$$

(Triple Matching) Output: ith output is

$$\mathbb{1}\{\exists j, k \in [N] \quad \text{s.t.} \quad z_i + z_j + z_k = 0 \pmod{M}\}$$

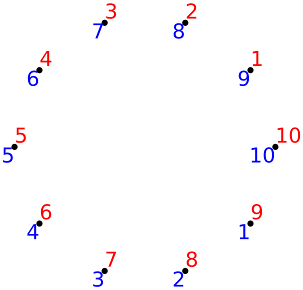
What we show (element matching problems)

- (Standard) self-att. unit can solve Pair Matching with $m = O(d_{\text{in}})$
- "Third-order" self-att. unit can solve Triple Matching with $m = O(d_{\text{in}})$
- Multi-headed self-att. layer requires $Hm = \tilde{\Omega}(N)$ to solve Triple Matching (+ almost matching upper bound)

Self-attention solution (Pair Matching)

Main idea: Choose $\phi\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^m$, W_Q, W_K s.t.

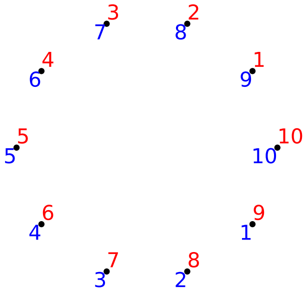
$$\left\langle W_Q^\intercal \phi(\mathrm{enc}(z)), W_K^\intercal \phi(\mathrm{enc}(z')) \right\rangle = \alpha\ \cos\left(\frac{2\pi(z+z')}{M}\right)$$



Self-attention solution (Pair Matching)

Main idea: Choose $\phi\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^m$, W_Q, W_K s.t.

$$\langle W_Q^\top \phi(\text{enc}(z)), W_K^\top \phi(\text{enc}(z')) \rangle = \alpha \, \cos\left(\frac{2\pi(z+z')}{M}\right)$$



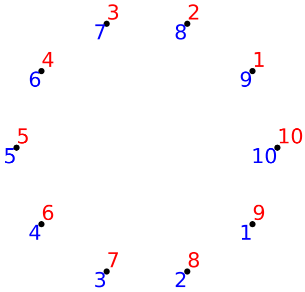
Large enough $\alpha = O(M^2 \log N)$ ensures we can distinguish between

- z_i's with at least one match
- z_i's with no matches

Self-attention solution (Pair Matching)

Main idea: Choose $\phi\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^m$, W_Q, W_K s.t.

$$\left\langle W_Q^\intercal \phi(\mathrm{enc}(z)), W_K^\intercal \phi(\mathrm{enc}(z')) \right\rangle = \alpha \cos\left(\frac{2\pi(z+z')}{M}\right)$$



Large enough $\alpha = O(M^2 \log N)$ ensures we can distinguish between

- z_i's with at least one match
- z_i's with no matches

(Dummy element supplies $W_V^\intercal \phi(\mathrm{enc}(\bot)) \neq W_V^\intercal \phi(\mathrm{enc}(z_i))$)

Lower bound for self-attention layers (Triple Matching)

Every H-headed self-attention layer with embedding dimension m that solves Triple Matching requires $H \times m = \tilde{\Omega}(N)$

Lower bound for self-attention layers (Triple Matching)

Every H-headed self-attention layer with embedding dimension m that solves Triple Matching requires $H \times m = \tilde{\Omega}(N)$

- Reduction from **DISJOINTNESS** problem
  Input: Alice has $a \in \{0,1\}^n$, Bob has $b \in \{0,1\}^n$
  Goal: After some communication, Bob determines if $\forall i \in [n]$, $a_i \wedge b_i = 0$
  Lower bound (Yao, 1979): Alice and Bob must exchange at least n bits

Lower bound for self-attention layers (Triple Matching)

Every H-headed self-attention layer with embedding dimension m that solves Triple Matching requires $H \times m = \tilde{\Omega}(N)$

- Reduction from **DISJOINTNESS** problem
  Input: Alice has $a \in \{0,1\}^n$, Bob has $b \in \{0,1\}^n$
  Goal: After some communication, Bob determines if $\forall i \in [n]$, $a_i \wedge b_i = 0$
  Lower bound (Yao, 1979): Alice and Bob must exchange at least n bits
- Create input x for Triple Matching from a and b (with $N = 2n+1$):

$$x_i = \begin{cases} \mathrm{enc}(1) & \text{if } a_i = 0 \\ \mathrm{enc}(i+1) & \text{if } a_i = 1 \end{cases}$$

$$x_{n+i} = \begin{cases} \mathrm{enc}(1) & \text{if } b_i = 0 \\ \mathrm{enc}(M-(i+1)) & \text{if } b_i = 1 \end{cases}$$

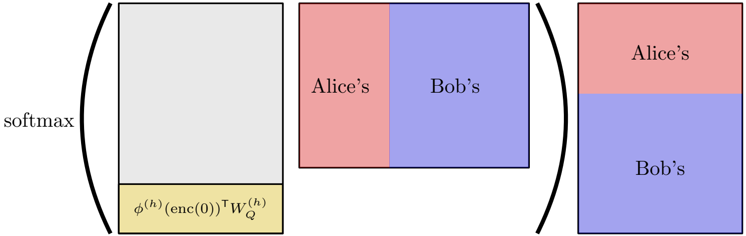
$$x_{2n+1} = \mathrm{enc}(0)$$

triple match with Nth query iff $\mathrm{DISJ}(a,b) = 0$

H-headed self-attention layer with embedding dimension m for Triple Matching provides a communication protocol using $H \times m \times \operatorname{poly}\log(N)$ bits

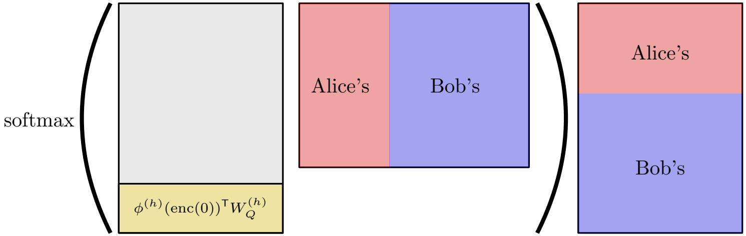
H-headed self-attention layer with embedding dimension m for Triple Matching provides a communication protocol using $H \times m \times \text{poly}\log(N)$ bits

Main idea: Alice & Bob can jointly compute $(2n+1)$th output



H-headed self-attention layer with embedding dimension m for Triple Matching provides a communication protocol using $H \times m \times \operatorname{poly}\log(N)$ bits

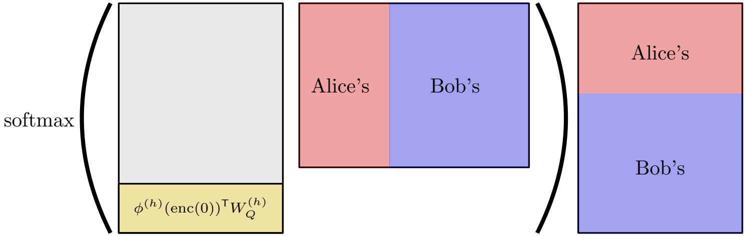
Main idea: Alice & Bob can jointly compute $(2n+1)$th output



1. Alice sends parts of softmax normalization terms to Bob

H-headed self-attention layer with embedding dimension m for Triple Matching provides a communication protocol using $H \times m \times \text{poly}\log(N)$ bits

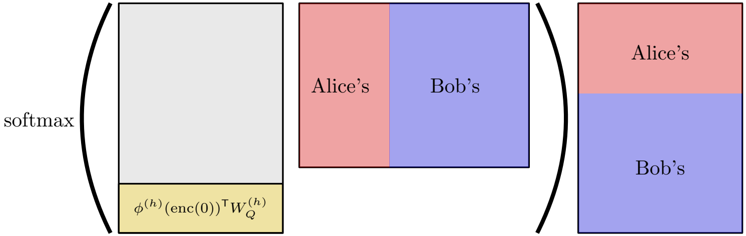
Main idea: Alice & Bob can jointly compute $(2n+1)$th output



1. Alice sends parts of softmax normalization terms to Bob
2. Bob completes softmax normalization terms; sends back to Alice

H-headed self-attention layer with embedding dimension m for Triple Matching provides a communication protocol using $H \times m \times \text{poly}\log(N)$ bits

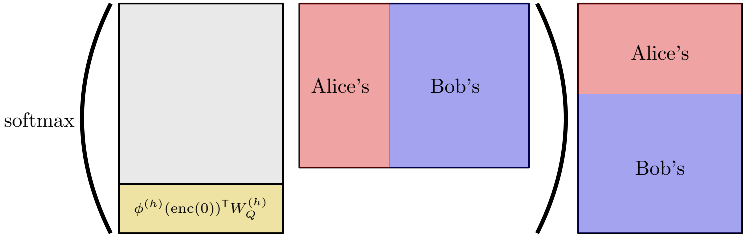
Main idea: Alice & Bob can jointly compute $(2n+1)$th output



1. Alice sends parts of softmax normalization terms to Bob
2. Bob completes softmax normalization terms; sends back to Alice
3. Alice sends partial weighted averages of “values” to Bob

H-headed self-attention layer with embedding dimension m for Triple Matching provides a communication protocol using $H \times m \times \operatorname{poly}\log(N)$ bits

Main idea: Alice & Bob can jointly compute $(2n+1)$th output



1. Alice sends parts of softmax normalization terms to Bob
2. Bob completes softmax normalization terms; sends back to Alice
3. Alice sends partial weighted averages of "values" to Bob
4. Bob completes computation of weighted averages of "values"

Lower bounds for L-layer transformers with $L > 1$

We conjecture (and wanted to prove):

Every transformer requires $L \times H \times m = N^{\Omega(1)}$ *to solve Triple Matching*

Lower bounds for L-layer transformers with $L > 1$

We conjecture (and wanted to prove):

Every transformer requires $L \times H \times m = N^{\Omega(1)}$ to solve Triple Matching

Instead we proved

Every "graph transformer" requires $LHm = \tilde{\Omega}(N)$ to solve Directed 3-Cycle

(But efficiently solved by a "higher-order graph self-attention unit")

Lower bounds for L-layer transformers with $L > 1$

We conjecture (and wanted to prove):

Every transformer requires $L \times H \times m = N^{\Omega(1)}$ to solve Triple Matching

Instead we proved

Every "graph transformer" requires $LHm = \tilde{\Omega}(N)$ to solve Directed 3-Cycle

(But efficiently solved by a "higher-order graph self-attention unit")

¯_(ツ)_/¯

Easier variants of Triple Matching

Easier variants of Triple Matching

- Variant 1 (family of hard instances for $L = 1$ lower bound):
  $x_N = 0$ and only care about Nth output

Easier variants of Triple Matching

- Variant 1 (family of hard instances for $L = 1$ lower bound):
  $x_N = 0$ and only care about Nth output

  There is a $L = 2$ transformer solution with $H = 1$ and $m = O(d_{\text{in}})$

  (This separates efficient $L = 1$ transformers from efficient $L = 2$ transformers)

Easier variants of Triple Matching

- Variant 1 (family of hard instances for $L = 1$ lower bound):
  $x_N = 0$ and only care about Nth output

  There is a $L = 2$ transformer solution with $H = 1$ and $m = O(d_{\text{in}})$

  (This separates efficient $L = 1$ transformers from efficient $L = 2$ transformers)

- Variant 2 ("Local" Triple Matching): ith output is

$$\mathbb{1}\{\exists j, k \in [i - T, i + T] \quad \text{s.t.} \quad z_i + z_j + z_k = 0 \pmod{M}\}$$

Easier variants of Triple Matching

- Variant 1 (family of hard instances for $L = 1$ lower bound): $x_N = 0$ and only care about Nth output

  There is a $L = 2$ transformer solution with $H = 1$ and $m = O(d_{\text{in}})$

  (This separates efficient $L = 1$ transformers from efficient $L = 2$ transformers)

- Variant 2 ("Local" Triple Matching): ith output is

$$\mathbb{1}\{\exists j, k \in [i - T, i + T] \quad \text{s.t.} \quad z_i + z_j + z_k = 0 \pmod{M}\}$$

  Can reduce to qSA problem with $q = O(T)$ and $d_{\text{in}} = \tilde{O}(T)$

Conclusion and future work

- Transformers offer some natural advantages over FNNs, RNNs

Conclusion and future work

- Transformers offer some natural advantages over FNNs, RNNs
- Pair Matching is a canonical easy problem for Transformer

Conclusion and future work

- Transformers offer some natural advantages over FNNs, RNNs
- Pair Matching is a canonical easy problem for Transformer
- Some reasons to think Triple Matching is canonical hard problem

Conclusion and future work

- Transformers offer some natural advantages over FNNs, RNNs
- Pair Matching is a canonical easy problem for Transformer
- Some reasons to think Triple Matching is canonical hard problem
- Also to-do:
  - Learning/generalization theory? [Edelman, Goel, Kakade, Zhang, 2022]
  - Pair-wise nature of "real problems" solved by Transformers?

Conclusion and future work

- Transformers offer some natural advantages over FNNs, RNNs
- Pair Matching is a canonical easy problem for Transformer
- Some reasons to think Triple Matching is canonical hard problem
- Also to-do:
  - Learning/generalization theory? [Edelman, Goel, Kakade, Zhang, 2022]
  - Pair-wise nature of "real problems" solved by Transformers?

Thank you!

```
%llion@: FAIR's paper seems to concentrate solely on the convolutional aspect
%of their model and have the attention as an after thought almost, this gives
%us a good opportunity to differentiate ourselves from their paper.

%We are simpler in a number of ways and should have the simplicity as a big selling point:
%\begin{itemize}
%\item No convolutions
%\item No need for such careful initializations and
%normalization.
%\item Simpler non-lineararities, they use the gated linear
%units.
%\item Less layers?
%\end{itemize}
%One thing we do more is that we have self attention.
%Another selling point is the increased interpretability as
%shown with the visualizations. Which comes from the
%simplicity and use of only attentions.
```

Third-order self-attention unit

Third-order self-attention unit:

$$f(X) = \mathrm{softmax}\Big((XW_Q)\big((XW_K^{(1)}) \star (XW_K^{(2)})\big)^{\mathsf{T}}\Big)\,\big((XW_V^{(1)}) \star (XW_V^{(2)})\big)$$

where $\star$ is column-wise Kornecker product (a.k.a. Khatri-Rao product)

Graph self-attention unit

- Input: $X \in \mathbb{R}^{N \times N}$, adjacency matrix of digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = [N]$
- Graph self-attention unit:

$$\mathrm{softmax}\Big(\kappa\big(X, (XW_Q)(XW_K)^\intercal\big)\Big)\, XW_V$$

where $\kappa \colon \mathbb{R} \to \mathbb{R}$ is an arbitrary function applied element-wise

Directed 3-Cycle and Undirected 5-Cycle

(Directed 3-Cycle) Output: ith output is

$$\mathbb{1}\{\exists j, k \in [N] \quad \text{s.t.} \quad (i, j), (j, k), (k, i) \in \mathcal{E}\}$$

(Undirected 5-Cycle) Output: ith output is

$$\mathbb{1}\{\exists j_1, j_2, j_3, j_4 \in [N] \quad \text{s.t.} \quad \{i, j_1\}, \{j_1, j_2\}, \{j_2, j_3\}, \{j_3, j_4\}, \{j_4, i\} \in \mathcal{E}\}$$

Model sizes

Model	Input size (N)	# Layers (L)	# Heads/layer (H)	Emb. dim. (m)	# nodes/ϕ
BERT	512	24	16	1024	4K
GPT-2	1K	12	12	768	?
GPT-3	2K	96	96	128	12K
GPT-4	32K	120 ?	?	?	?