Large Language Models

Computational & Representational Capacity

A presentation by -

Soham Mallick & Manit Paul

DEPT. OF STATISTICS & DATA SCIENCE

Presentation for STAT 9910

27th February, 2025

Outline

1. **Introduction** - What is "**representation ability**"?

Outline

1. **Introduction** - What is "**representation ability**"?
2. **Representing Languages**
   Probability Distributions
   Svete and Cotterell (2024), Nowak et al. (2024)
   Formal Languages
   Strobl et al. (2024)

Outline

1. **Introduction** - What is "**representation ability**"?
2. **Representing Languages**

   Probability Distributions
   Svete and Cotterell (2024), Nowak et al. (2024)
   Formal Languages
   Strobl et al. (2024)
3. **Representing Functions**

Peng et al. (2024), Bhattamishra et al. (2024)

Introduction: What is Representation Ability?

What is Representation Ability?

Define the "*tasks*" that a transformer can "*do*".

What is Representation Ability?

Define the "*tasks*" that a transformer can "*do*".

Q. What languages can it "*represent*"?

Q. What other structures (distributions) in languages can it "*learn*"?

Q. What functions can it "*approximate*"?

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$$\underbrace{\begin{pmatrix} \text{“I have a cat”} \\ \text{“He likes computers”} \\ . \\ . \end{pmatrix}}_{X_0 \in 2^L} \xrightarrow{T} \underbrace{\begin{pmatrix} p(\text{“I want to go to the zoo”}) \\ p(\text{“Let us buy a headphone”}) \\ . \\ . \end{pmatrix}}_{T(X_0)}$$

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Note: T is also a function of the architecture.

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Note: T is also a function of the architecture.

$$\boxed{T(X_0) : \sigma(\Sigma^*) \longrightarrow [0, 1]}$$

Abstraction of a Language Model

Transformer & Language Model

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Note: T is also a function of the architecture.

Most papers: $T : \Sigma^* \longrightarrow [0,1]$. Is $\sum_{y \in \Sigma^*} T(y) = 1$?

Next-Token Prediction

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Next-Token Prediction

$$\mathcal{L}_T : \Sigma^* \longrightarrow \mathcal{P}(\Sigma),\ \mathcal{L}_T(y) = T(\cdot \mid y)$$

Next-Token Prediction

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Next-Token Prediction

$$\mathcal{L}_T : \Sigma^* \longrightarrow \mathcal{P}(\Sigma),\ \mathcal{L}_T(y) = T(\cdot \mid y)$$

$$\underbrace{\text{“I have a cat. It drinks”}}_{y} \xrightarrow{L_T} (p(\text{“food”}), p(\text{“water”}), p(\text{“milk”}), \ldots)$$

Next-Token Prediction

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Next-Token Prediction

$$\mathcal{L}_T : \Sigma^* \longrightarrow \mathcal{P}(\Sigma),\ \mathcal{L}_T(y) = T(\cdot \mid y)$$

$$\underbrace{\text{“I have a cat. It drinks”}}_{y} \xrightarrow{L_T} (p(\text{“food”}), p(\text{“water”}), p(\text{“milk”}), \ldots)$$

$$p(\text{“food”}) = \frac{T(\text{“I have a cat. It drinks milk”})}{T(\text{“I have a cat. It drinks”})}$$

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Distributions

Q. Which probability distributions can be produced?

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Distributions

Q. Which probability distributions can be produced? What subset of $\mathcal{P}(\Sigma^*)$ can be realized?

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Distributions

Q. Which probability distributions can be produced? What subset of $\mathcal{P}(\Sigma^*)$ can be realized?

A. At least n-gram models (lower bound).

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Formal Language Recognizers

Q. Which formal languages can be identified?

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Formal Language Recognizers

Q. Which formal languages can be identified?

$$L = \{y \in \Sigma^* \mid \sigma(T(y)) = 1, \text{ for some classifier } \sigma\}$$

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Formal Language Recognizers

Q. Which formal languages can be identified?

$$L = \{y \in \Sigma^* \mid \sigma(T(y)) = 1, \text{ for some classifier } \sigma\}$$

A. A lot of results (upper and lower bounds).

Representational Capacity

Transformer

Token Space: Σ, Space of all strings: Σ^*, Language: $L \subseteq \Sigma^*$.

$$T : 2^L \longrightarrow \mathcal{P}(\Sigma^*)$$

$T(X_0)$ (called a **Language Model**) is a prob. distribution on Σ^*.

Representing LMs as Function Approximators

Result (Upper Bound): Not very good at function composition.

More results in the second half.

Representing LMs as Probability Distributions

n-gram LMs

General Decomposition

$$p(y) = p(EOS \mid y) \prod_{t=1}^{|y|} p\left(y_t \mid \mathbf{y_{<t}}\right)$$

n-gram LMs

General Decomposition

$$p(y) = p(EOS \mid y) \prod_{t=1}^{|y|} p\left(y_t \mid \mathbf{y}_{<\mathbf{t}}\right)$$

Definition: n-gram LMs

If $p\left(y_t \mid \mathbf{y}_{<\mathbf{t}}\right) = p\left(y_t \mid y_{t-1}, y_{t-2}, \ldots y_{t-n+1}\right)$, we call the LM to be n-gram and $\mathbf{y}_{t-n+1}^{t-1} = \left(y_{t-1}, y_{t-2}, \ldots y_{t-n+1}\right)$ is called the *history* of y_t.

n-gram LMs: Results

Weak Equivalence

Two LMs p and q over Σ^* are weakly equivalent if $p(\mathbf{y}) = q(\mathbf{y}) \; \forall \; \mathbf{y} \in \Sigma^*$.

n-gram LMs: Results

Weak Equivalence

Two LMs p and q over Σ^* are weakly equivalent if $p(\mathbf{y}) = q(\mathbf{y}) \ \forall \ \mathbf{y} \in \Sigma^*$.

Main Results

Theorem 3.1 - 3.3 in Svete and Cotterell (2024)

For every n-gram LM, there exists a weakly equivalent ($\{$hard, sparse$\}$ attention) transformer LM.

n-gram LMs: Results

Weak Equivalence

Two LMs p and q over Σ^* are weakly equivalent if $p(\mathbf{y}) = q(\mathbf{y}) \ \forall \ \mathbf{y} \in \Sigma^*$.

Main Results

Theorem 3.1 - 3.3 in Svete and Cotterell (2024)

For every n-gram LM, there exists a weakly equivalent ($\{$hard, sparse$\}$ attention) transformer LM with

1. $n-1$ heads and 1 layer.
2. 1 head and $n-1$ layers.
3. 1 head and 1 layer.

n-gram LMs: Results

Weak Equivalence

Two LMs p and q over Σ^* are weakly equivalent if $p(\mathbf{y}) = q(\mathbf{y}) \; \forall \; \mathbf{y} \in \Sigma^*$.

Main Results

Theorem 3.1 - 3.3 in Svete and Cotterell (2024)

For every n-gram LM, there exists a weakly equivalent ($\{$hard, sparse$\}$ attention) transformer LM with

1. $n - 1$ heads and 1 layer.
2. 1 head and $n - 1$ layers.
3. 1 head and 1 layer.

No uniqueness!

Construction with $n - 1$ heads and 1 layer

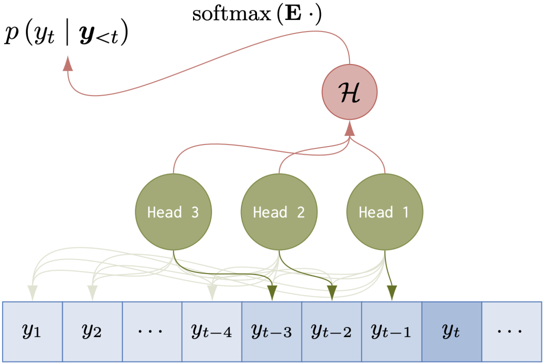


Figure: Figure showing where each head puts its attention

Construction with $n-1$ heads and 1 layer

Choice of positional encoding:

$$r(y_t, t) = \begin{pmatrix} [[y_t]] \\ \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \\ \sqrt{\frac{1}{t+1}} \\ \sqrt{1-\frac{1}{t+1}} \\ \cdot \\ \cdot \\ \cdot \\ \sqrt{\frac{1}{t+n-1}} \\ \sqrt{1-\frac{1}{t+n-1}} \end{pmatrix} \left\{ \begin{array}{l} Q_h : r(y_t, t) \to \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix} \\ K_h : r(y_t, t) \to \begin{pmatrix} \sqrt{\frac{1}{t+h}} \\ \sqrt{1-\frac{1}{t+h}} \end{pmatrix} \end{array} \right.$$

Construction with $n-1$ heads and 1 layer

Choice of score function:

$$
\begin{aligned}
& f(q_t, k_j) \\
& = \langle Q_h(r(y_t, t)), K_h(r(y_j, j)) \rangle \\
& = \langle \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \rangle \quad \begin{cases} Q : r(y_t, t) \to \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix} \\ K_h : r(y_t, t) \to \begin{pmatrix} \sqrt{\frac{1}{t+h}} \\ \sqrt{1-\frac{1}{t+h}} \end{pmatrix} \end{cases}
\end{aligned}
$$

Construction with $n-1$ heads and 1 layer

Choice of score function:

$$
\begin{aligned}
& f(q_t, k_j) \\
& = \langle Q_h(r(y_t, t)), K_h(r(y_j, j)) \rangle \\
& = \left\langle \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \right\rangle \quad \left\{ \begin{array}{l} Q : r(y_t, t) \to \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix} \\ K_h : r(y_t, t) \to \begin{pmatrix} \sqrt{\frac{1}{t+h}} \\ \sqrt{1-\frac{1}{t+h}} \end{pmatrix} \end{array} \right.
\end{aligned}
$$

Maximized when $t = j + h$, or $j = t - h$.

Construction with $n-1$ heads and 1 layer

Choice of score function:

$$
\begin{aligned}
&f(q_t, k_j)\\
&= \langle Q_h(r(y_t,t)), K_h(r(y_j,j))\rangle\\
&= \left\langle \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix}, \begin{pmatrix} \sqrt{\frac{1}{j+h}} \\ \sqrt{1-\frac{1}{j+h}} \end{pmatrix} \right\rangle
\end{aligned}
\quad
\begin{cases}
Q : r(y_t,t) \to \begin{pmatrix} \sqrt{\frac{1}{t}} \\ \sqrt{1-\frac{1}{t}} \end{pmatrix} \\[2ex]
K_h : r(y_t,t) \to \begin{pmatrix} \sqrt{\frac{1}{t+h}} \\ \sqrt{1-\frac{1}{t+h}} \end{pmatrix}
\end{cases}
$$

Maximized when $t = j+h$, or $j = t-h$. Now apply **hardmax**.

Follow-up Work

1. LMs with **Chain of Thought** (CoT) are able to express *regular* language and have been shown to be Turing complete (with arbitrary precision) (Nowak et al. (2024)).

Follow-up Work

1. LMs with **Chain of Thought** (CoT) are able to express *regular* language and have been shown to be Turing complete (with arbitrary precision) (Nowak et al. (2024)).

Observation: For Turing completeness, you need extra symbols in your alphabet (Merrill and Sabharwal (2024)).

Follow-up Work

1. LMs with **Chain of Thought** (CoT) are able to express *regular* language and have been shown to be Turing complete (with arbitrary precision) (Nowak et al. (2024)).

Observation: For Turing completeness, you need extra symbols in your alphabet (Merrill and Sabharwal (2024)).

2. Experiments to study learnability of n-gram models w.r.t n, $|\Sigma|$ and $\text{rank}(\mathbf{E})$ under a metric (like KL divergence) (Svete et al. (2024)).

Summary

1. *n*-gram LMs can be represented by Transformer LMs with hard, sparse or soft attention.
2. Space complexity has been addressed in Theorem 5.1 of Svete and Cotterell (2024).
3. Lower bound is quite loose.
4. Follow-up work has empirical results and integrates CoT.

Representing LMs as Language Recognizers

Formal Languages

Expressivity of transformers through circuit complexity & logic.

Can provide both lower and upper bounds.

Formal Languages

Expressivity of transformers through circuit complexity & logic.

Can provide both lower and upper bounds.

Eg: Transfomers LM with left-hard attention cannot recognize **PARITY**.

Define classes of languages which can be recognized.

Circuit Complexity

Circuit

A circuit is a directed acyclic graph with n input vertices and zero or more gate vertices (**NOT**, **AND**, **OR**).

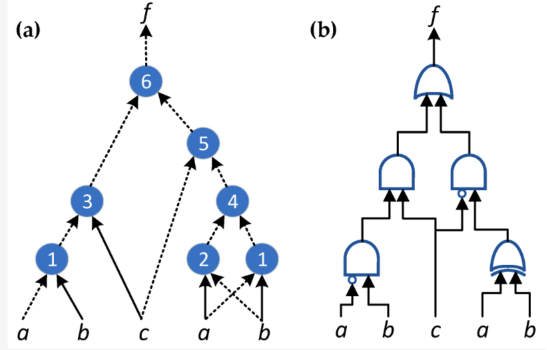


Figure: A Circuit

Circuit Complexity

Circuit

A circuit is a directed acyclic graph with n input vertices and zero or more gate vertices (**NOT**, **AND**, **OR**).

Circuit complexity classes classify circuit families and the languages they recognize based on

1. uniformity
2. depth
3. size
4. fan-in bound
5. allowed gates

Circuit Complexity

Circuit complexity classes classify circuit families and the languages they recognize based on

1. uniformity
2. depth
3. size
4. fan-in bound
5. allowed gates

E.g.: AC_0 contains those languages that can be recognized by families of circuits with unbounded fan-in, constant depth, and polynomial size.

Circuit Complexity

Circuit complexity classes classify circuit families and the languages they recognize based on

1. uniformity
2. depth
3. size
4. fan-in bound
5. allowed gates

E.g.: AC_0 contains those languages that can be recognized by families of circuits with unbounded fan-in, constant depth, and polynomial size.

$$AC_0 \subseteq ACC_0 \subseteq TC_0 \subseteq NC^1$$

Circuit Complexity

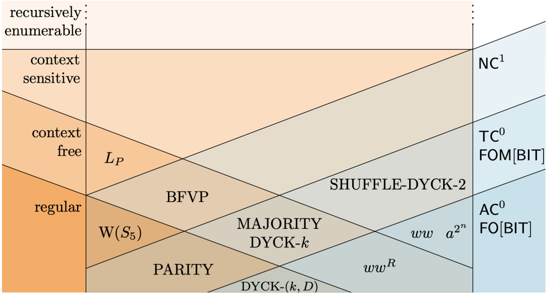


Figure: Relationship of some language classes w.r.t the Chomsky hierarchy

Main results: (Strobl et al. (2024)) Transformers with *left-most* or *right-most* hard attention can only recognise languages in AC_0. Those with *average-hard* or *soft-max* attention seem to lie between ACC_0 and TC_0.

Lower bound	Source	PE	Attention	Notes
$\ni$ MAJORITY	Pérez et al. 2019	none	average-hard	
$\ni$ SHUFFLE-DYCK-k	Bhattamishra et al. 2020a	none	softmax, future mask	
$\supseteq$ SSCMs	Bhattamishra et al. 2020a	none	softmax, future mask	
$\ni$ DYCK-k	Yao et al. 2021	$i/n, i/n^3, n$	softmax & leftmost-hard	
$\supseteq$ P	Pérez et al. 2021	$i, 1/i, 1/i^2$	average-hard	poly(n) steps
$\ni$ PARITY	Chiang and Cholak 2022	$i/n, (-1)^i$	softmax	
$\supseteq$ FOC[MOD; +]	Chiang et al. 2023	sinusoidal	softmax	
$\supseteq$ FO[Mon]	Barceló et al. 2024	arbitrary	leftmost-hard	
$\supseteq$ LTL+C[Mon]	Barceló et al. 2024	arbitrary	average-hard	
Upper bound	**Source**	**Precision**	**Attention**	**Notes**
$\not\ni$ PARITY, DYCK-1	Hahn 2020	$\mathbb{R}$	leftmost-hard	
$\not\ni$ PARITY, DYCK-2	Hahn 2020	$\mathbb{R}$	softmax, future mask	$\varepsilon_N > 0$, vanishing KL
$\subseteq$ AC^0	Hao et al. 2022	$\mathbb{Q}$	leftmost-hard	
$\subseteq$ TC^0	Merrill et al. 2022	$\mathbb{F}$	average-hard	
$\subseteq$ FOC[MOD; +]	Chiang et al. 2023	$O(1)$	softmax	
$\subseteq$ L-uniform TC^0	Merrill & Sabharwal 2023a	$O(\log n)$	softmax	
$\subseteq$ FOM[BIT]	Merrill & Sabharwal 2023b	$O(\log n)$	softmax	
$\subseteq$ L-uniform TC^0	Strobl 2023	$\mathbb{F}$	average-hard	
Equivalent	**Source**	**PE**	**Attention**	**Notes**
= RE	Pérez et al. 2021	$i, 1/i, 1/i^2$	average-hard	unbounded steps
= FO	Angluin et al. 2023	none	rightmost-hard, strict future mask	
= FO[MOD]	Angluin et al. 2023	sinusoidal	rightmost-hard, strict future mask	
= FO[Mon]	Angluin et al. 2023	arbitrary	rightmost-hard, strict future mask	
= P	Merrill & Sabharwal 2024	none	average-hard, future mask	poly(n) steps

Figure: Surveyed claims and their assumptions in Strobl et al. (2024)

Representing LMs as Function Approximators

Function Composition

Prompt: Fayes is to the west of Xaive, Jill is to the north of Ken, Fayes is to the south of Ken, where is Ken with respect to Xaive?

GPT 3.5: East

GPT 4: Northeast

Bard: Not enough information

Correct answer: Northwest

Prompt: If Amy is to the southwest of Ben, Cindy is to the northeast of Amy and directly north of Ben, is Amy further from Ben or Cindy?

GPT 3.5: Ben

GPT 4: Ben

Bard: Ben

Correct answer: Cindy

Figure: Spatial composition produces incorrect answers

Function Composition

Main Results (Informal)

Theorems 1 & 2 in Peng et al. (2024):

A *single* Transformer attention layer cannot compute the answer to a function composition query correctly with significant probability of success, as long as the *size of the domain of the function is large*.

Function Composition

Main Results (Informal)

Theorems 1 & 2 in Peng et al. (2024):

A *single* Transformer attention layer cannot compute the answer to a function composition query correctly with significant probability of success, as long as the *size of the domain of the function is large*.

Even when equipped with **CoT**, transformers need far more tokens in the generated CoT prompt to solve the composition problem.

Function Composition

Main Results

Theorem 1 in Peng et al. (2024):

Consider the functions $f : A \to B,\ g : B \to C, h : C \to D$ and the *function composition* problem $h \circ g \circ f$ with domain size $|A| = |B| = |C| = n$, and an H-headed transformer layer L with embedding dimension d and computation precision p.

If $R = n \log n - H(d+1)p > 0$, then the probability, over all possible functions and queries, that L answers the query incorrectly is at least $R/(3n \log n)$.

Function Composition

Main Results

Theorem 1 in Peng et al. (2024):

Consider the functions $f : A \to B$, $g : B \to C$, $h : C \to D$ and the *function composition* problem $h \circ g \circ f$ with domain size $|A| = |B| = |C| = n$, and an H-headed transformer layer L with embedding dimension d and computation precision p.

If $R = n \log n - H(d+1)p > 0$, then the probability, over all possible functions and queries, that L answers the query incorrectly is at least $R/(3n \log n)$.

Proof relies on **complexity theory** and this bound is **tight**.

Function Composition

Main Results

Theorem 2 in Peng et al. (2024):

Consider the functions $f : A \to B$, $g : B \to C, h : C \to D$ and the *function composition* problem $h \circ g \circ f$ with domain size $|A| = |B| = |C| = n$, and an H-headed transformer layer L with embedding dimension d and computation precision p.

A transformer layer requires $\Omega\left(\sqrt{\frac{n}{Hdp}}\right)$ CoT steps for correctly answering iterated function composition prompts.

Function Composition

Main Results

Theorem 2 in Peng et al. (2024):

Consider the functions $f : A \to B,\ g : B \to C, h : C \to D$ and the *function composition* problem $h \circ g \circ f$ with domain size $|A| = |B| = |C| = n$, and an H-headed transformer layer L with embedding dimension d and computation precision p.

A transformer layer requires $\Omega\left(\sqrt{\frac{n}{Hdp}}\right)$ CoT steps for correctly answering iterated function composition prompts.

Proof relies on **complexity theory**.

Summary

1. Producing *probability distributions*.
2. Recognizing *formal languages*.
3. *Function approximation*. Manit will speak more on this topic!

Strengths of Transformers as Function Approximators

Tasks

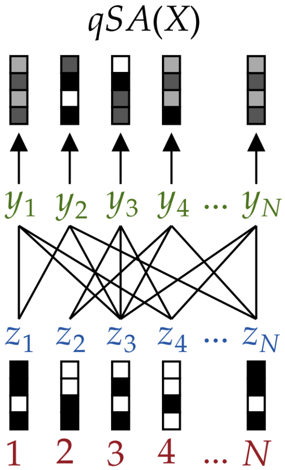
Sanford et al. (2023) introduces some tasks to study the representation power of the transformer in comparison to other neural network architectures.

(i) $q-$sparse averaging: qSA.
(ii) Similar pair detection: Match2.
(iii) Similar triple detection: Match3.

Sanford et al. (2023) further focuses on the intrinsic complexity parameters viz. embedding dimension, width, and depth required for approximately solving these tasks.

$q-$Sparse Averaging

- **Input**: $X = (x_1, \cdots, x_N) \in \mathbf{R}^{N \times d}$ where $d = d' + q + 1$ and,
  1. $x_i = (z_i; y_i; i)$,
  2. $z_i \in \mathbb{B}^{d'} := \{x \in \mathcal{R}^d : ||x||_2 \leq 1\}$,
  3. $y_i \in \binom{[N]}{q}$.
- **Output**: $\text{qSA}(X)_i = \frac{1}{q} \sum_{j \in y_i} z_j$.

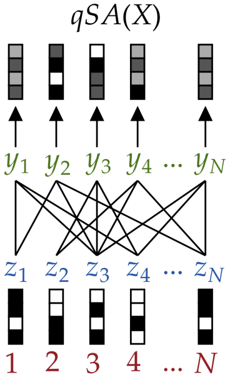


Main results

Definition: $f : \mathbb{R}^{N\times d} \to \mathbb{R}^{N\times d'}$ $\epsilon-$approximates $q-$SA if for all X,

$$\max_{i\in[N]} ||f(X)_i - \mathrm{qSA}(X)_i||_2 \le \epsilon.$$

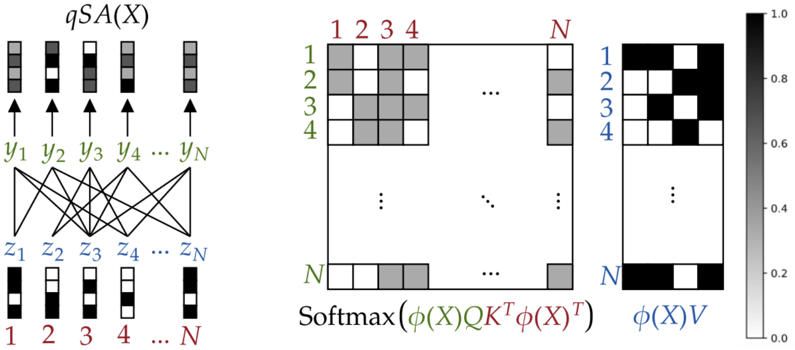
- Any fully-connected neural network (of depth at-least 2) that $1/(2q)-$approximates $q-$SA (for $q < N/2$) satisfies Width $\ge Nd'/2$.
- Any recurrent neural network with $m-$bit memory (dimension of the hidden state h_i) that $\epsilon-$approximates $q-$SA (for $q = 1, d' = 1$) must satisfy $m \ge (N-1)/2$.
- There exists a self-attention unit approximating $q-$SA iff embedding dimension $m \gtrsim q$.



Positive result

- **Theorem**: For all q, N, any $m \geq \Omega(d' + q \log N)$, any $\epsilon \in (0, 1)$, and precision $p = \Omega(\log((q/\epsilon) \log N))$, there exists a self-attention unit $f \in \mathcal{A}_{m,m,d',p}\Phi_{d,m,p}$ that $\epsilon-$approximates $q-$SA.
- **Remark**: The extra $\log N$ factor can be eliminated by taking infinite bit precision.
- Transformers are much more efficient at approximating $q-$ Sparse averaging than RNNs and FNNs for $d' \ll q \ll N$.

Proof outline



For all q, N, $m \gtrsim q \log N$ there exists a self-attention unit with $\log N$-bit precision that ϵ–approximates q–SA.

Proof outline

- Choose the MLP ϕ and value matrix V such that $V^T\phi(x_i) = z_i$.
- Choose the key and query matrix K, Q such that,

$$\mathrm{Softmax}(\phi(X)QK^T\phi(X)^T)_{i,j} \approx \begin{cases} \frac{1}{q} & \text{if } j \in y_i, \\ 0 & \text{otherwise.} \end{cases}$$

- This implies,

$$[\mathrm{Softmax}(\phi(X)QK^T\phi(X)^T)\phi(X)V]_i \approx \frac{1}{q}\sum_{j\in y_i} z_j = q\text{-SA}(X)_i \quad \text{for } i \in [N].$$

How to choose Q, K?

- **Definition**: A matrix $U \in \mathbb{R}^{m \times N}$ satisfies the $(q, \delta)-$ restricted isometry and orthogonality property if,

  $$||Uv||_2^2 \in [(1-\delta)||v||_2^2, (1+\delta)||v||_2^2] \quad \text{and} \quad |\langle Uv, Uv' \rangle| \leq \delta ||v||_2 ||v'||_2,$$

  for all vectors $v, v' \in \mathbb{R}^N$ with $|\text{supp}(v)| \leq q, |\text{supp}(v')| \leq 2q$, and $\text{supp}(v) \cap \text{supp}(v') = \phi$.

- Candes and Tao [2005] There is an absolute constant $C > 0$ such that the following holds. Fix $\delta \in (0, 1/2)$ and $q \in \mathbb{N}$. Let U denote a random $m \times N$ matrix of independent Rademacher random variables scaled by $1/\sqrt{m}$. If $m \geq C(q \log N)/\delta^2$, then with positive probability, U satisfies the (q, δ)-restricted isometry and orthogonality property.

- Candes and Tao [2005] Fix $\delta \in (0, 1/2)$ and $q \in \mathbb{N}$. Let the matrix $U = [u_1, \cdots, u_N] \in \mathbb{R}^{m \times N}$ satisfy the $(q, \delta)-$restricted isometry and orthogonality property. For every vector $v \in \{0, 1\}^N$ with $|\text{supp}(v)| \leq q$, there exists $w \in \mathbb{R}^m$ satisfying,

  $$||w||_2 \leq \sqrt{q}/(1-2\delta); \ \langle u_i, w \rangle = 1 \text{ if } v_i = 1; \ |\langle u_i, w \rangle| \leq \delta/(1-2\delta) \text{ if } v_i = 0.$$

How to choose Q, K?

- Select K such that $\phi(X)K = (u_1, \cdots, u_N) \in \mathbb{R}^{N \times m'}$, $u_i \in \{\pm 1/\sqrt{m'}\}^{m'}$ and $K^T \phi(X)^T$ satisfies the $(q, 1/4)-$restricted isometry and orthogonality property (existence guaranteed for $m' \gtrsim q \log N$ by Candes and Tao [2005]).
- By Candes and Tao [2005] for each y_i there exists $w_{y_i} \in \mathbb{R}^m$ with $||w_{y_i}||_2 \leq 2\sqrt{q}$ satisfying,

$$\langle u_j, w_{y_i} \rangle = 1 \text{ if } j \in y_i,$$
$$\langle u_j, w_{y_i} \rangle \leq \frac{1}{2} \text{ if } j \notin y_i.$$

- Given the bounded precision of the model for all $i \in [N]$ there exists $\tilde{w}_{y_i}$ such that $||\tilde{w}_{y_i} - w_{y_i}|| \leq \epsilon/(4\alpha)$ for $\alpha = \lceil 2 \log(4N/\epsilon) \rceil$.

How to choose Q, K?

- Choose ϕ, Q, K such that for $i \in [N]$,

$$\phi(x_i) = (z_i; \alpha \tilde{w}_{y_i}; u_i),$$
$$K^T \phi(x_i) = u_i,$$
$$Q^T \phi(x_i) = \alpha \tilde{w}_{y_i}.$$

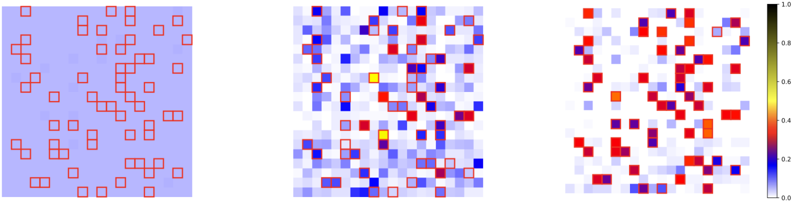
- It can be shown that,

$$\frac{1}{q}\left(1 - \frac{\epsilon}{2}\right) \leq \mathrm{Softmax}(\phi(X) Q K^T \phi(X)^T)_{i,j} \leq \frac{1}{q}\left(1 + \frac{\epsilon}{2}\right) \text{ for } j \in y_i,$$
$$\mathrm{Softmax}(\phi(X) Q K^T \phi(X)^T)_{i,j} \leq \frac{\epsilon}{2N} \text{ for } j \notin y_i.$$

- This implies for $f(X) = \mathrm{Softmax}(\phi(X) Q K^T \phi(X)^T) \phi(X) V$,

$$||f(X)_i - q\text{-SA}(X)_i||_2 \leq q \frac{\epsilon}{2q} + (N - q) \frac{\epsilon}{2N} \leq \epsilon \text{ for all } i \in [N].$$

Proof outline



(a) $T = 0$. (b) $T = 1000$. (c) $T = 40000$.

Attention matrix Softmax$(\phi(X)QK^T\phi(X)^T) \in \mathbb{R}^{20\times 20}$ for a fixed example after T epochs of training a self-attention unit to solve $q-$SA for q $= 3$.

Negative result

- **Theorem**: (Impossibility result) For any sufficiently large q, any $N \geq 2q + 1$, and any $d' \geq 1$, there exists a universal constant c such that if $m \leq cq/p$, then there exists no self-attention unit f that $1/(2q)-$approximates $q-$SA.

- **Implications**: There exists self-attention unit approximating $q-$SA iff embedding dimension $m \gtrsim q$.

- The proof is done using a standard communication complexity argument.

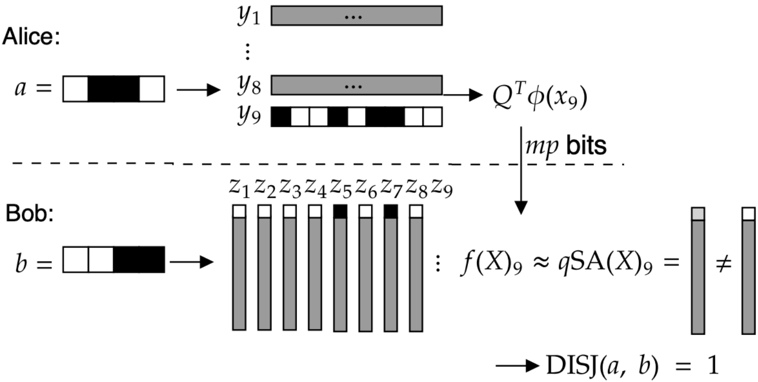
- **Key fact**: Yao [1979] Suppose Alice and Bob are given inputs $a, b \in \{0,1\}^n$ respectively with the goal of jointly computing $\mathrm{DISJ}(a,b) = \max_i a_i b_i$ by alternately sending a single bit message to the other party over a sequence of communication rounds. Any deterministic protocol for computing $\mathrm{DISJ}(a,b)$ requires at least n rounds of communication.

Proof outline

Theorem: Any self-attention unit f that approximates $q-$SA with $p-$bit precision requires embedding dimension $m \geq \Omega(q/p)$.

- Given f the goal is to design a $O(mp)-$bit communication protocol to solve $\mathrm{DISJ}(a, b)$ with $n = q$ and $N \geq 2q + 1$.
- Alice encodes a in y_{2q+1}: If $a_i = 1$ then $2i - 1 \in y_{2q+1}$, otherwise $2i \in y_{2q+1}$.
- Bob encodes b in z_j's: $z_j = e_1$ if j is odd and $b_{(j+1)/2} = 1$, $z_j = -e_1$ otherwise.
- It can be checked that $\mathrm{DISJ}(a, b) = 0$ iff $q-\mathrm{SA}(X)_{2q+1} = -e_1$.
- **Communication protocol**: (i) Using f Alice computes $Q(x_{2q+1}) \in \mathbb{R}^m$ using y_{2q+1} and sends $Q(x_{2q+1})$ to Bob using $O(mp)$ bits.
(ii) Bob computes $f(X)_{2q+1}$ and returns 0 iff $f(X)_{2q+1} = -e_1$.

Proof outline



Match2 and Match3

- **Input**: $X = (x_1, \cdots, x_N) \in [M]^N$ where $M = \text{poly}(N)$.
- **Output**:

$$\text{Match2}(X)_i = \mathbf{1}\{\exists j : x_i + x_j = 0 (\text{mod } M)\},$$
$$\text{Match3}(X)_i = \mathbf{1}\{\exists j_1, j_2 : x_i + x_{j_1} + x_{j_2} = 0 (\text{mod } M)\}.$$

- In other words, the output is an $N-$ dimensional vector in $\{0, 1\}^N$ whose i-th element is 1 iff X has a pair (for Match2) or triple (for Match3) containing x_i.

Positive result for Match2

$$\mathrm{Match2}(X)_i = \mathbf{1}\{\exists j : x_i + x_j = 0 (\mathrm{mod}\ M)\} \quad \text{for all } i \in [N].$$

- **Theorem**: For any $N, M = N^{O(1)}, p = O(\log M)$, there exists a transformer architecture with a single self attention unit $f \in \Phi_{m,1,p}\mathcal{A}_{m,m,m,p}\Phi_{1,m,p}$ with embedding dimension $m = 3$ such that for all $X \in [M]^N$,
$$f(X) = \mathrm{Match2}(X).$$

Proof outline

- A single blank token $x' = 0$ is appended at the end of the sequence X. We consider the modified input $X' = (x_1, \cdots, x_N, x')$.
- Choose the MLP ϕ and value matrix V such that $V^T\phi(x_i) = 1$ for $i \in [N]$ and $V^T\phi(x') = 0$.
- Choose the key and query matrix K, Q such that,

$$\mathrm{Softmax}(\phi(X)QK^T\phi(X)^T)_{i,j} \approx \begin{cases} \frac{1}{\beta_i+1} & \text{if } x_i + x_j = 0 (\mathrm{mod}\ M), \\ \frac{1}{\beta_i+1} & \text{for } j = N+1, \\ 0 & \text{otherwise.} \end{cases}$$

where $\beta_i = |\{j \in [N] : x_i + x_j = 0 (\mathrm{mod}\ M)\}|$.

- This implies,

$$[\mathrm{Softmax}(\phi(X)QK^T\phi(X)^T)\phi(X)V]_i \approx \begin{cases} \frac{\beta_i}{\beta_i+1} & \text{if } x_i \text{ is paired,} \\ 0 & \text{otherwise.} \end{cases}$$

How to choose Q, K?

- Define input MLP $\phi : \mathbb{R} \to \mathbb{R}^3$, and query, key matrices $Q, K \in \mathbb{R}^{3\times 3}$ such that for all $i \in [n]$,

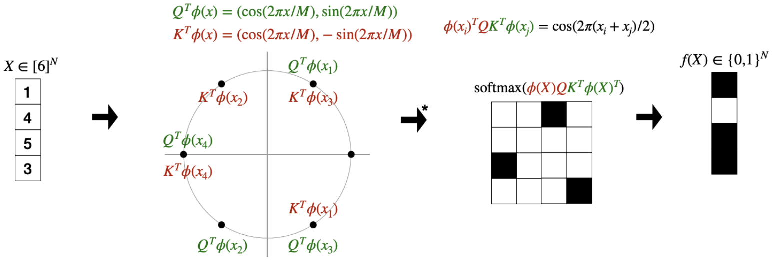
$$Q^T\phi(x_i) = c(\cos(2\pi x_i/M), \sin(2\pi x_i/M), 1);\ \ Q^T\phi(x') = \vec{0};$$
$$K^T\phi(x_i) = (\cos(2\pi x_i/M), -\sin(2\pi x_i/M), 0);\ \ K^T\phi(x') = e_3.$$

- This implies for $i, j \in [N]$,

$$\langle Q^T\phi(x_i), K^T\phi(x_j)\rangle = c\cos(2\pi(x_i + x_j)/M),$$
$$\langle Q^T\phi(x_i), K^T\phi(x')\rangle = c.$$

- $\langle Q^T\phi(x_i), K^T\phi(x_j)\rangle = c$ if $x_i + x_j = 0 (\mathrm{mod}\ M)$ and $\langle Q^T\phi(x_i), K^T\phi(x_j)\rangle \leq c(1 - (1/M^2))$ otherwise.

How to choose Q, K?



There exists a transformer architecture with a single self attention unit f (with embedding dimension $m = 3$) such that for all $X \in [M]^N$, $f(X) = \text{Match2}(X)$.

How to choose Q, K?

- If we take $c = M^2 \log(6N)$ we have $\mathrm{Softmax}(\phi(X)QK^T\phi(X)^T)_{i,j} \in$

$$\begin{cases} \left[0, \frac{1}{6N}\right] & \text{if } x_i + x_j \neq 0 (\mathrm{mod}\ N); i,j \in [N], \\ \left[\frac{1}{\beta_i+1} \pm \frac{1}{6N}\right] & \text{if } x_i + x_j = 0 (\mathrm{mod}\ N); i,j \in [N], \\ \left[\frac{1}{\beta_i+1} \pm \frac{1}{6N}\right] & \text{if } i \in [N], j = N+1. \end{cases}$$

- This implies for $i \in [N]$ $[\mathrm{Softmax}(\phi(X)QK^T\phi(X)^T)\phi(X)V]_i$

$$\begin{cases} \leq \frac{1}{6} & \text{if } \nexists j : x_i + x_j = 0 (\mathrm{mod}\ N), \\ \geq \left(\frac{\beta_i}{\beta_i+1} - \frac{1}{6}\right) & \text{if } \exists j : x_i + x_j = 0 (\mathrm{mod}\ N). \end{cases}$$

- Define an output MLP ψ such that $\psi(z) = 0$ if $z \leq 1/6$ and $\psi(z) = 1$ if $z \geq 1/3$.

Negative result for Match3

$$\text{Match3}(X)_i = \mathbf{1}\{\exists j_1, j_2 : x_i + x_{j_1} + x_{j_2} = 0 \pmod M\} \quad \text{for all } i \in [N].$$

- **Theorem**: (Impossibility result) There is universal constant $c > 0$ such that for sufficiently large N, and any $M \geq N + 1$, if $m \leq cN/(pH \log(\log N))$, then there is no transformer architecture f with a single multi-headed self-attention layer satisfying $f(X) = \text{Match3}(X)$ for all $X \in [M]^N$.
- The proof is done using a communication complexity argument.
- **Conjecture**: Every multi-layer transformer that computes Match3 must have width, depth, embedding dimension, or bit complexity at least $N^{\Omega(1)}$.

Positive results for other variants of Match3

For all $i \in [N]$ and $K << N$,

$$\text{Match3Bigram}(X)_i = \mathbf{1}\{\exists j : x_i + x_j + x_{j+1} = 0 (\text{mod } M)\}.$$

$$\text{Match3Local}(X)_i = \mathbf{1}\{\exists j_1, j_2 : x_i + x_{j_1} + x_{j_2} = 0 (\text{mod } M), |i - j_1|, |i - j_2| \leq K\}.$$

- There exists a transformer architecture f with depth $D = 2$, embedding dimension $m = 3$ such that for all X, $f(X) = \text{Match3Bigram}(X)$.
- There exists a transformer architecture f with depth $D = 1$, embedding dimension $m = O(K \log N)$ such that for all X, $f(X) = \text{Match3Local}(X)$.

Major takeaways

- We discussed two “natural” tasks that exhibit key separations between transformers and other models.
- **Sparse averaging** is efficient for transformers, but inefficient for RNNs, FNNs.
- **Pair finding** is easy for transformers, **triple finding** is not.

Bibliography I

A. Svete and R. Cotterell, “Transformers can represent n-gram language models,” *arXiv preprint arXiv:2404.14994*, 2024.

F. Nowak, A. Svete, A. Butoi, and R. Cotterell, “On the representational capacity of neural language models with chain-of-thought reasoning,” *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 12 510–12 548, 2024.

L. Strobl, W. Merrill, G. Weiss, D. Chiang, and D. Angluin, “What formal languages can transformers express? a survey,” *arXiv preprint arXiv:2311.00208*, 2024.

B. Peng, S. Narayanan, and C. Papadimitriou, “On limitations of the transformer architecture,” *COLM 2024*, 2024.

S. Bhattamishra, M. Hahn, P. Blunsom, and V. Kanade, “Separations in the representational capabilities of transformers and recurrent architectures,” *arXiv preprint arXiv:2406.09347*, 2024.

Bibliography II

W. Merrill and A. Sabharwal, "The expressive power of transformers with chain of thought," in *Proceedings of the Twelfth International Conference on Learning Representations*, 2024.

A. Svete, N. Borenstein, M. Zhou, I. Augenstein, and R. Cotterell, "Can transformers learn *n*-gram language models?" *arXiv preprint arXiv:2410.03001*, 2024, accessed: 2025-02-26. [Online]. Available: https://arxiv.org/abs/2410.03001

C. Sanford, D. Hsu, and M. Telgarsky, "Representational strengths and limitations of transformers," *arXiv preprint arXiv:2306.02896*, 2023. [Online]. Available: https://arxiv.org/abs/2306.02896

Thank You!