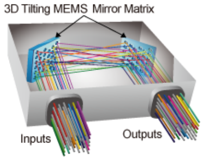
A theorist takes a look at LLMs

Dean Foster, Amazon

November 5, 2024



What I won't tell you about today

- What I've been doing for the past 2 years:
  - Engineering: building LLMs
  - Alignment: defending LLMs ([1])
  - Optimization: picking the batch size ([2])
  - Applications: tutoring children ([3], [4])
  - Theory: ?

  Today, I'll tell you mostly about my theory work

What can theory or even science tell us about LLMs?

Theory? What is it good for?

What can theory or even science tell us about LLMs?

Theory? What is it good for?

to quote Edwin Star (or mis-attribute to James Brown):

What can theory or even science tell us about LLMs?

Theory? What is it good for?

to quote Edwin Star (or mis-attribute to James Brown):

Absolutely Nothing!

What can theory add?

Examples of cool theory:

1. Chain of though can learn a Turing machine (Malach 2023)
2. what LLMs can learn ("leap complexity" 2023)
3. saddle point escape
4. Many papers on two layer network theory as tensors
5. Many theory paper on the first step of SGD / Adam
6. μP (asymptotics: "Tensors" 2020-23)
7. Matyroshka (principal of marginality: Kakade 2023)

What can theory add?

Examples of cool theory:

1. Chain of though can learn a Turing machine (Malach 2023)
2. what LLMs can learn ("leap complexity" 2023)
3. saddle point escape
4. Many papers on two layer network theory as tensors
5. Many theory paper on the first step of SGD / Adam
6. μP (asymptotics: "Tensors" 2020-23)
7. Matyroshka (principal of marginality: Kakade 2023)

But only 6 and 7 offer practical advice.

This talk: useful TCS for LLMs

I'll present 3 short ideas with implications for real NNs. The theory will be stolen from:

1. Computation complexity
2. cryptography
3. statistics

Idea #1:

Computational complexity

Some current NN computability theorems

Theorem (Daniel Hsu 2023)

An transformer LLM can answer the "two sum" problem, but to answer a "three sum" requires it to be extremely wide. (arxiv)

Some current NN computability theorems

Theorem (Merrill and Sabharwal 2023)

An LLM can only answer questions in TC(0) if asked directly for the answer. (arxiv)

Some current NN computability theorems

Theorem (Malach 2023)

A linear LLM can be trained to mimic a Turing machine using chain-of-thought.

Some current NN computability theorems

Theorem (Giannou, Rajput, Sohn, Lee, Lee, and Papailiopoulos 2023)

Looped Transformers are general computers.

Chain of thought

$$\text{Is } \sqrt{2\pi} \stackrel{?}{>} e?$$

Chain of thought

$$\text{Is } \sqrt{2\pi} \stackrel{?}{>} e?$$

- Asking directly forces the LLM to guess.

Chain of thought

$$\text{Is } \sqrt{2\pi} \stackrel{?}{>} e?$$

- Asking, “Think step-by-step and work out...”
- Higher accuracy

Chain of thought

$$\text{Is } \sqrt{2\pi} \overset{?}{>} e?$$

- Asking, “Take a deep breath and work out...” (Sept 2023)
- Even higher accuracy

Contrasting native LLMs vs Chain of Thought

Theorem (Merrill and Sabharwal 2023)

(rephrased) An LLM can not answer questions in PSPACE.

Contrasting native LLMs vs Chain of Thought

Theorem (Merrill and Sabharwal 2023)

(rephrased) An LLM can not answer questions in PSPACE.

Theorem (F. and Madeka 2023)

Using chain of thought reasoning, an LLM can solve any problem in PSPACE.

Contrasting native LLMs vs Chain of Thought

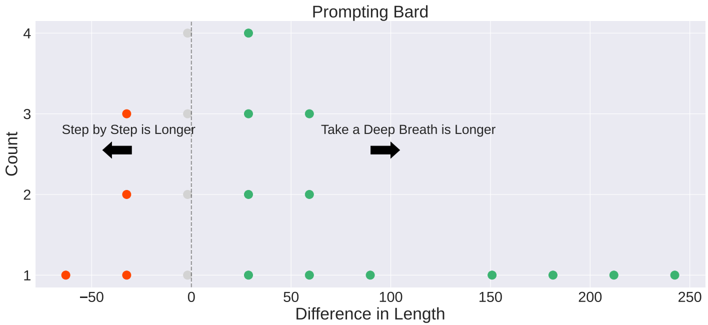
Theorem (Merrill and Sabharwal 2023)

(rephrased) An LLM can not answer questions in PSPACE.

Theorem (F. and Madeka 2023)

Using chain of thought reasoning, an LLM can solve any problem in PSPACE.

A long roll out is where the new power comes from.



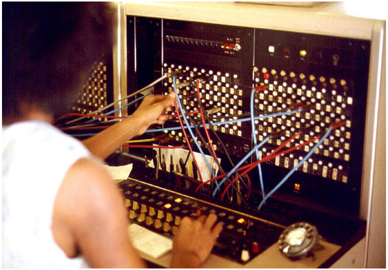
Prompting Bard
Step by Step is Longer
Take a Deep Breath is Longer
Count
Difference in Length
4
3
2
1
−50
0
50
100
150
200
250

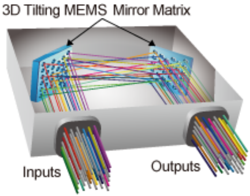
Implication #1:

Use tiers of NN

Hardware:

- Compute: 1000s of GPUs instances
- Communication: TBytes/s





[MEMS]

Tiered model

- Bottom tier:
  - training: usual transformer model
  - Generates “roll outs”

Tiered model

- Bottom tier:
  - training: usual transformer model
  - Generates “roll outs”
- Middle tiers:
  - training: Using history and roll-out, predict next word
  - generates new roll outs

Tiered model

- Bottom tier:
  - training: usual transformer model
  - Generates “roll outs”
- Middle tiers:
  - training: Using history and roll-out, predict next word
  - generates new roll outs
- Top tier:
  - Training: Reads all roll outs and history then predictions the next word
  - inference: Generate all roll outs and then generate next word

Alternative implementation

To reify our theorem, you need to build an interpreter into the LLM.

Alternative Implementation

To reify our theorem, you need to build an interpreter into the LLM.

- This means training it on doing step-by-step thinking
- If you are going to externalize some thinking, it should be trained on that also
- For example, using *Lean* as a proof assistant:

```
...text {Lean code}{Lean output} text...
```

- This should be part of the training data

Idea #2:

One way functions

One way functions

A one way function is one where $f(x)$ is easy to compute, but $f^{-1}(y)$ is hard to compute.
Examples:

- Cryptography
- pseudo random number generators
- block chain

Causal mask

We process words L2R in a transformer based LLM.

- Not as obvious as in an auto-regressive LLM (e.g. LSTM)
- Still, all values are “time stamped”
  - Every node in a transformer has a time stamp
  - It only depends on tokens that came before that time stamp
  - represented by the causal triangle matrix

Simple L2R doesn't mean simple R2L

Example (L2R is easy, R2L is hard)

There is a function of T tokens that can be computed L2R in one pass each step of degree 2 such that when it is computed R2L in one pass it requires degree 2^T.

Simple L2R doesn't mean simple R2L

Example (L2R is easy, R2L is hard)

There is a function of T tokens that can be computed L2R in one pass each step of degree 2 such that when it is computed R2L in one pass it requires degree 2^T.

How to beat the counter-example:

- copy all data to the last token T
- Now mimic the L2R

Simple L2R doesn't mean simple R2L

Example (L2R is easy, R2L is hard)

There is a function of T tokens that can be computed L2R in one pass each step of degree 2 such that when it is computed R2L in one pass it requires degree 2^T.

Theorem

Any function of T tokens which can be simply computed L2R in L layers and embedding dimension d can be simply computed R2L in $L+1$ layers and an embedding dimension of Td.

Simple L2R doesn't mean simple R2L

Example (L2R is easy, R2L is hard)

There is a function of T tokens that can be computed L2R in one pass each step of degree 2 such that when it is computed R2L in one pass it requires degree 2^T.

Theorem

Any function of T tokens which can be simply computed L2R in L layers and embedding dimension d can be simply computed R2L in $L + 1$ layers and an embedding dimension of Td.

Requires a huge embedding space

Implication #2:

a causal mask enlarges embeddings

Removing some of the causal mask

Divide our micro-batch into two pieces:

- non-causal, non-predicted *recent* history (R)
- *causal* sequence of predicted tokens (C)
- Details:
  - the full micro-batch is:

$$[R_1, R_2, \ldots, R_r, C_1, C_2, \ldots C_T]$$

  - R_i^{L+1} can depend on R_j^L for all pairs i and j.
  - C_i^{L+1} can only depend on C_j^L for $j \leq i$, and on any R_k^L

Empirical results (on tiny models)

Comparing a full causal model to a model divided into $[R, C]$, running with the same compute and same number of parameters:

- causal: entropy = 3.4 nats/token
- $[R, C]$: entropy = 3.3 nats/token

Implications for state space models

- Some researchers like state space models
- Token $\rightarrow$ hidden state $\rightarrow$ next token

Implications for state space models

- Some researchers like state space models
- Token $\to$ hidden state $\to$ next token
- (Token $\times$ hidden state) $\to$ hidden state $\to$ next

Implications for state space models

- Some researchers like state space models
- Token $\rightarrow$ hidden state $\rightarrow$ next token
- (Token $\times$ hidden state) $\rightarrow$ hidden state $\rightarrow$ next
- Use a conditional random field to model the context
  - Looks like a state space model
  - But information flows in both direction

Idea #3:

Statistical vs computational batches

LLM find patterns

$$\begin{array}{rcl}
\overline{L}(\text{random guessing}) & = & 15 = \log_2(60,000) \\
\overline{L}(\text{unigrams word frequency}) & = & 11.7 = \log_2(3300) \\
\overline{L}(\text{bigrams (aka Markov)}) & = & 8.8 = \log_2(500) \\
\overline{L}(\text{gzip (LZ compression)}) & = & 8.2 = \log_2(300) \\
\overline{L}(\text{small LLM}) & = & 7.5 = \log_2(200) \\
\overline{L}(\text{Humans})) & \approx & 4 \\
\overline{L}(\text{LLM}) & = & 3.6 = \log_2(12)
\end{array}$$

(All in bits per token. I did the small LLM. Shannon, Cover/King did the human subjects estimation.)

Statistics independence

Palm masked out the first 10% of their tokens in every batch.

- Worked with a batch of 2000 tokens
- $Y_1, \dots, Y_{t-1}$ used to predict Y_t
- But only for $t = 201, 202, \dots, 2000$
- First 200 tokens not predicted in this batch

Statistics independence

Our recent / causal model:

- has 512 tokens in R
- has 512 tokens in C
- So only half of the microbatch is predicted

Implication #3:

Statistical length $\neq$ window length

Use overlapping batches

- L = batch size
- s = "stride" (the number of predictions made)

Traditional batches:

$$\begin{aligned}
\text{batch } 1 &= [1, L] \\
\text{batch } 2 &= [L+1, L+L] \\
\text{batch } 3 &= [2L+1, 2L+L] \\
\text{batch } 4 &= [3L+1, 3L+L] \\
\vdots &\ \vdots\ \vdots \\
\text{batch i} &= [iL+1, iL+L]
\end{aligned}$$

Use overlapping batches

- L = batch size
- s = "stride" (the number of predictions made)

Statistical batches:

$$
\begin{aligned}
\text{batch 1} &= [1, L] \\
\text{batch 2} &= [s+1, s+L] \\
\text{batch 3} &= [2s+1, 2s+L] \\
\text{batch 4} &= [3s+1, 3s+L] \\
\vdots \quad &\vdots \quad \vdots \\
\text{batch i} &= [is+1, is+L]
\end{aligned}
$$

Our $[R, L]$ network

which beat the standard transformer,

consumed half as much data.

Summary

We presented:

1. complexity of chain of thought
2. one way functions
3. degrees of freedom

Which implied we should:

- use tiered NNs (roll-outs)
- limit causal masks ($[R, C]$ network)
- distinguish statistical stride from window length

Summary

We presented:

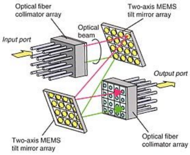
1. complexity of chain of thought
2. one way functions
3. degrees of freedom

Which implied we should:

- use tiered NNs (roll-outs)
- limit causal masks ([R, C] network)
- distinguish statistical stride from window length

THANKS!

THANKS!


Optical fiber
collimator array
Two-axis MEMS
tilt mirror array
Optical
beam
Input port
Output port
Two-axis MEMS
tilt mirror array
Optical fiber
collimator array