Transformers Represent Belief State Geometry in their Residual Stream

by **Adam Shai** 16th Apr 2024

PIBBSS | Interpretability (ML & AI) | Transformers | AI | Curated | 2024 Top Fifty: 55%

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Produced while being an affiliate at PIBBSS[1]. The work was done initially with funding from a Lightspeed Grant, and then continued while at PIBBSS. Work done in collaboration with @Paul Riechers, @Lucas Teixeira, @Alexander Gietelink Oldenziel, and Sarah Marzen. Paul was a MATS scholar during some portion of this work. Thanks to Paul, Lucas, Alexander, Sarah, and @Guillaume Corlouer for suggestions on this writeup.

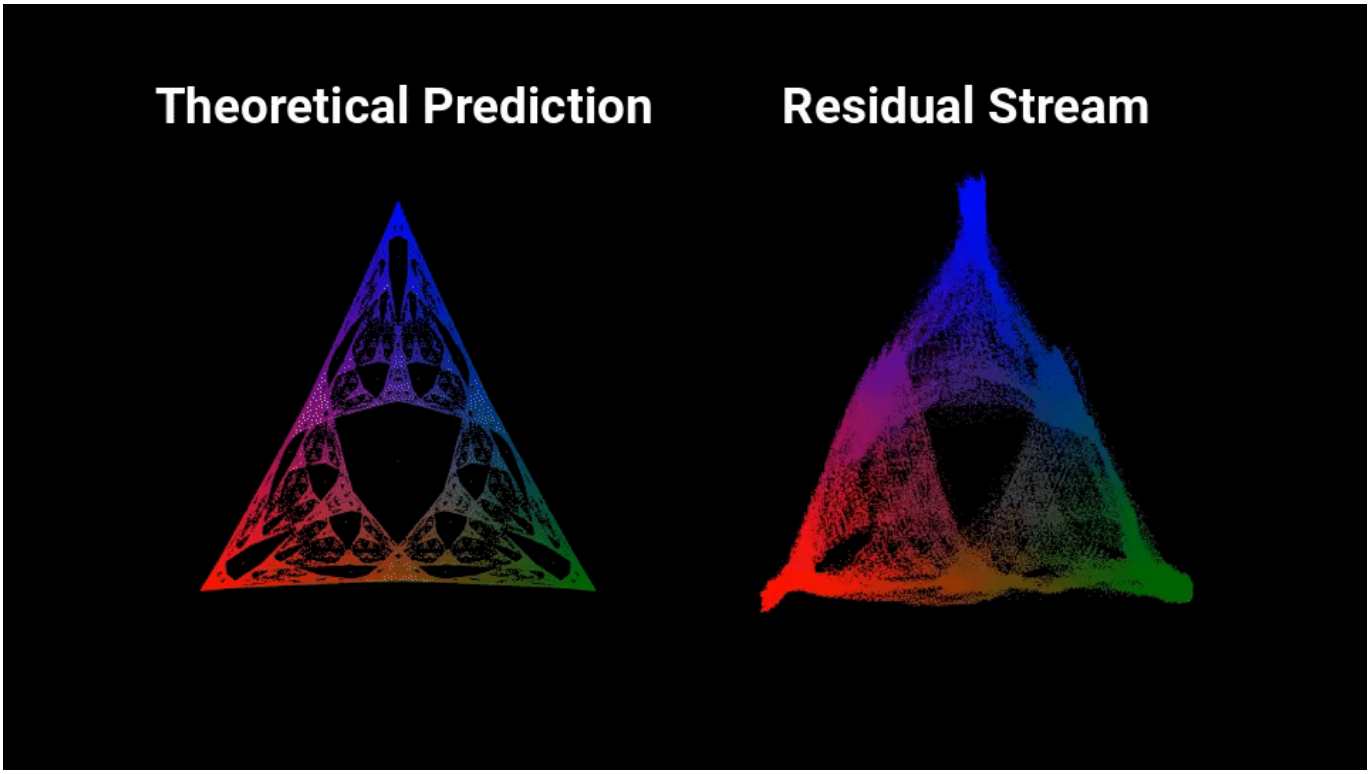
Update May 24, 2024: See our manuscript based on this work

Introduction

What computational structure are we building into LLMs when we train them on next-token prediction? In this post we present evidence that this structure is given by the **meta-dynamics of belief updating over hidden states of the data-generating process**. We'll explain exactly what this means in the post. We are excited by these results because

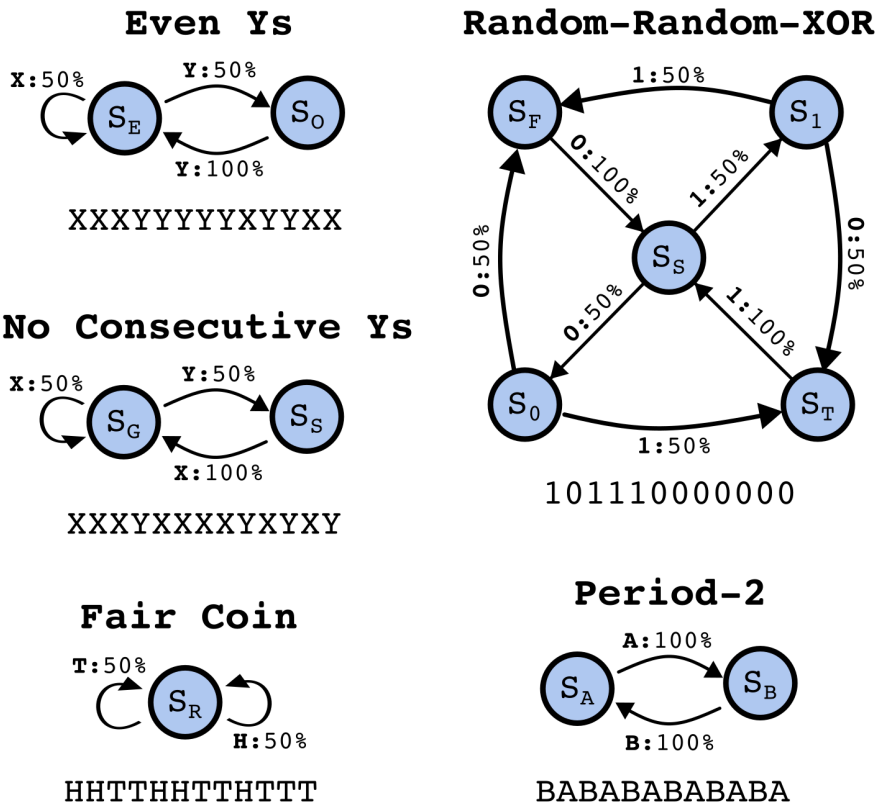
- We have a formalism that **relates training data to internal structures in LLMs**.
- Conceptually, our results mean that **LLMs synchronize to their internal world model** as they move through the context window.
- The computation associated with synchronization can be formalized with a framework called **Computational Mechanics**. In the parlance of Computational Mechanics, we say that LLMs represent the Mixed-State Presentation of the data generating process.
- The structure of synchronization is, in general, richer than the world model itself. In this sense, **LLMs learn more than a world model**.

- We have increased hope that **Computational Mechanics can be leveraged for interpretability and AI Safety more generally**.
- There's just something inherently cool about making a non-trivial prediction - in this case that the transformer will represent a specific fractal structure - and then verifying that the prediction is true. Concretely, **we are able to use Computational Mechanics to make an *a priori* and specific theoretical prediction about the geometry of residual stream activations (below on the left), and then show that this prediction holds true empirically (below on the right).**

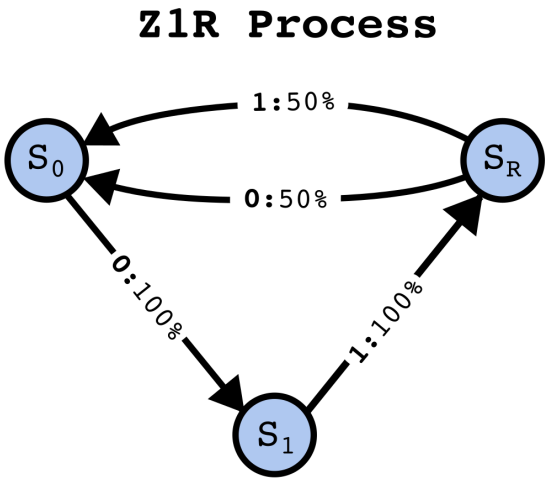


Theoretical Framework

In this post we will operationalize training data as being generated by a Hidden Markov Model (HMM)[2]. An HMM has a set of hidden states and transitions between them. The transitions are labeled with a probability and a token that it emits. Here are some example HMMs and data they generate.



Consider the relation a transformer has to an HMM that produced the data it was trained on. This is general - any dataset consisting of sequences of tokens can be represented as having been generated from an HMM. Through the discussion of the theoretical framework, let's assume a simple HMM with the following structure, which we will call the *Z1R process*[3] (for "zero one random").



The Z1R process has 3 hidden states, S_0, S_1, and S_R. Arrows of the form $S_x \xrightarrow{\mathbf{a}\,:\,p\%} S_y$ denote $P(S_y, \mathbf{a}|S_x) = p\%$, that the probability of moving to state S_y and emitting the token $\mathbf{a}$, given that the process is in state S_x, is $p\%$. In this way, taking transitions between the states stochastically generates binary strings of the form `...01R01R...` where `R` is a random 50/50 sample from {`0`, `1`}.

The HMM structure *is not* directly given by the data it produces. Think of the difference between the list of strings this HMM emits (along with their probabilities) and the hidden structure itself[4]. Since the transformer only has access to the strings of emissions from this HMM, and *not* any information about the hidden states directly, if the transformer learns anything to do with the hidden structure, then it has to do the work of *inferring* it from the training data.

What we will show is that when they predict the next token well, **transformers are doing even more computational work than inferring the hidden data generating process!**

Do Transformers Learn a Model of the World?

One natural intuition would be that the transformer must represent the hidden structure of the data-generating process (ie the "world"[2]). In this case, this would mean the three hidden states and the transition probabilities between them.

This intuition often comes up (and is argued about) in discussions about what LLM's "really understand." For instance, Ilya Sutskever has said:

> Because if you think about it, what does it mean to predict the next token well enough? It's actually a much deeper question than it seems. **Predicting the next token well means that you understand the underlying reality that led to the creation of that token.** It's not statistics. Like it is statistics but what is statistics? In order to understand those statistics to compress them, you need to understand what is it about the world that creates this set of statistics.

This type of intuition is natural, but it is not formal. Computational Mechanics is a formalism that was developed in order to study the limits of prediction in chaotic and other hard-to-predict systems, and has since expanded to a deep and rigorous theory of computational structure for any process. One of its many contributions is in providing a rigorous answer to what structures are necessary to perform optimal prediction.

Interestingly, Computational Mechanics shows that *prediction is substantially more complicated than generation*. What this means is that we should expect a transformer trained to predict the next token well should have *more structure* than the data generating process!

The Structure of Belief State Updating

But what is that structure exactly?

Imagine you know, exactly, the structure of the HMM that produces `...01R...` data. You go to sleep, you wake up, and you see that the HMM has emitted a `1`. What state is the HMM in now? It is possible to generate a `1` both from taking the deterministic transition $S_1 \xrightarrow{\mathbf{1}:100\%} S_R$ or from taking the stochastic transition $S_R \xrightarrow{\mathbf{1}:50\%} S_0$. Since the deterministic transition is twice as likely as the 50% one, the best you can do is to have some belief distribution over the current states of the HMM, in the case $P([S_0, S_1, S_R]) = [\frac{1}{3}, 0, \frac{2}{3}]$[5].

		`1`	`1`	`0`	`1...`
P(S_0)	$\frac{1}{3}$	$\frac{1}{3}$	1	0	0...
P(S_1)	$\frac{1}{3}$	0	0	1	0...
P(S_R)	$\frac{1}{3}$	$\frac{2}{3}$	0	0	1...

If now you see another `1` emitted, so that in total you've seen `11`, you can now use your previous belief about the HMM state (read: prior), and your knowledge of the HMM structure alongside the emission you just saw (read: likelihood), in order to generate a new belief state (read: posterior). An exercise for the reader: What is the equation for updating your belief state given a previous belief state, an observed token, and the transition matrix of the ground-truth HMM?[6] In this case, there is only one way for the HMM to generate `11`, $S_1 \xrightarrow{\mathbf{1}:100\%} S_R \xrightarrow{\mathbf{1}:50\%} S_0$, so you know for certain that the HMM is now in state S_0. From now on, whenever you see a new symbol, you will know exactly what state the HMM is in, and we say that you have *synchronized* to the HMM.

In general, as you observe increasing amounts of data generated from the HMM, you can continually update your belief about the HMM state. Even in this simple example there is non-trivial structure in these belief updates. For instance, it is not always the case that seeing 2 emissions is enough to synchronize to the HMM. If instead of `11...` you saw `10...` you still wouldn't be synchronized, since there are two different paths through the HMM that generate `10`.

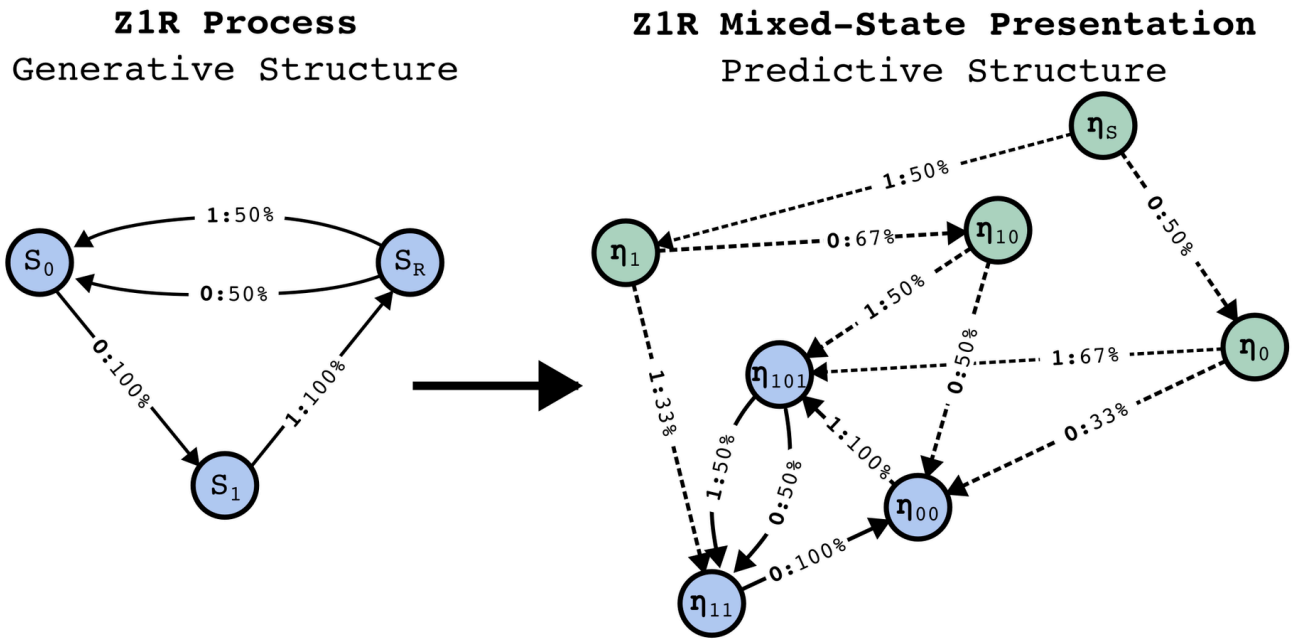
The structure of belief-state updating is given by the *Mixed-State Presentation.*

The Mixed-State Presentation

Notice that just as the data-generating structure is an HMM - at a given moment ***the process*** is in a hidden state, then, given an emission, ***the process*** move to another hidden state - so to is your belief updating! ***You*** are in some belief state, then given an emission that you observe, ***you*** move to some other belief state.

	Data Generating Process	**Belief State Process**
States belong to	The data generating mechanism	The observer of the outputs of the data generating process
States are	Sets of sequences that constrain the future in particular ways	The observer's beliefs over the states of the data generating process
Sequences of hidden states emit	Valid sequences of data	Valid sequences of data
Interpretation of emissions	The observables/tokens the data generating process emits	What the observer sees from the data generating process

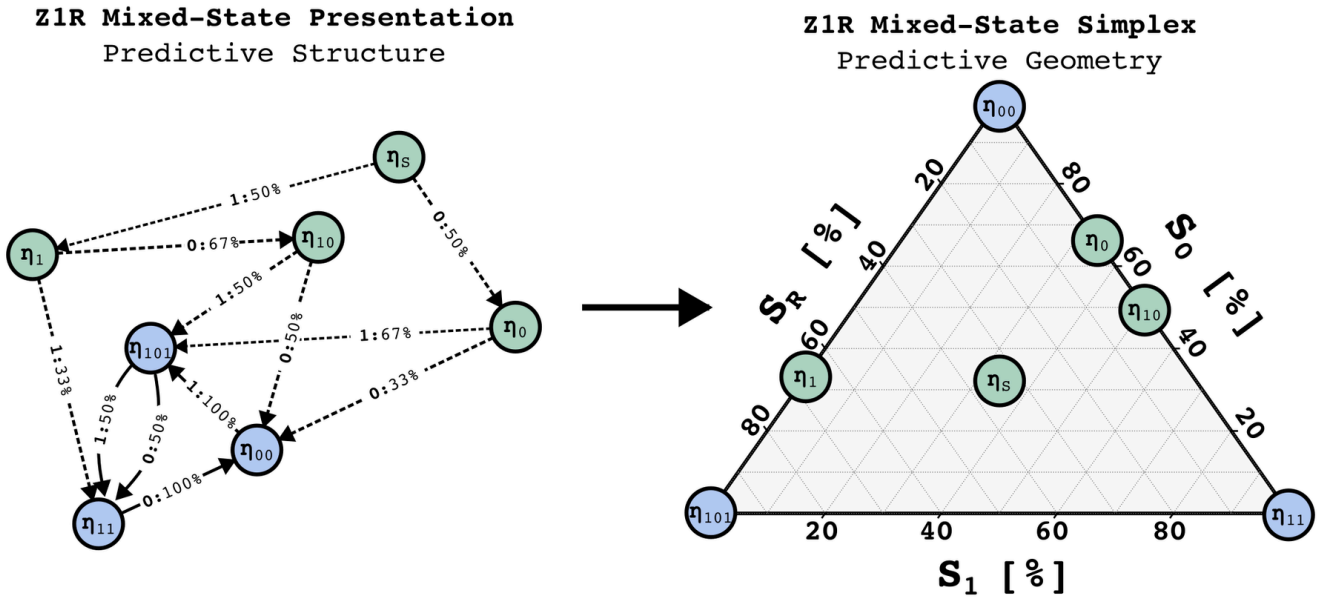
The meta-dynamics of belief state updating are formally another HMM, where the hidden states are your *belief states*. This meta-structure is called the *Mixed-State Presentation (MSP)* in Computational Mechanics.



Note that the MSP has transitory states (in green above) that lead to a recurrent set of belief states that are isomorphic to the data-generating process - this always happens, though there might be infinite transitory states. *Synchronization* is the process of moving through the transitory states towards convergence to the data-generating process.

A lesson from Computational Mechanics is that in order to perform optimal prediction of the next token based on observing a finite-length history of tokens, one must implement the Mixed-State Presentation (MSP). That is to say, to predict the next token well one should know what state the data-generating process is in as best as possible, and to know what state the data-generating process is in, implement the MSP.

The MSP has a geometry associated with it, given by plotting the belief-state values on a simplex. In general, if our data generating process has N states, then probability distributions over those states will have $N-1$ degrees of freedom, and since all probabilities must be between 0 and 1, all possible belief distributions lie on an $N-1$ simplex. In the case of Z1R, that means a 2-simplex (i.e. a triangle). We can plot each of our possible belief states in this 2-simplex, as shown on the right below.



What we show in this post is that when we train a transformer to do next token prediction on data generated from the 3-state HMM, we can find a linear representation of the MSP geometry in the residual stream. This is surprising! Note that the points on the simplex, the belief states, *are not the next token probabilities.* In fact, multiple points here have literally the same next token predictions. In particular, in this case, η_{10}, η_S, and η_{101}, all have the same optimal next token predictions.

Another way to think about this claim is that **transformers keep track of distinctions in anticipated distribution over the entire future, beyond distinctions in next token predictions, even though the transformer is only trained explicitly on next token prediction!** That means the transformer is keeping track of extra information than what is necessary just for the local next token prediction.

Another way to think about our claim is that **transformers perform two types of inference**: one to infer the structure of the data-generating process, and another meta-inference to update it's internal beliefs over which state the data-generating process is in, given some history of finite data (ie the context window). This second type of inference can be thought of as the algorithmic or computational structure of synchronizing to the hidden structure of the data-generating process.

A final theoretical note about Computational Mechanics and the theory presented here: because Computational Mechanics is not contingent on the specifics of transformer architectures and is a well-developed first-principles framework, we can apply this framework to any optimal predictor, not just transformers[7].

Experiment and Results

Experimental Design

To repeat the question we are trying to answer:

> What computational structure are we building into LLMs when we train them on next-token prediction?

To test our theoretical predictions, we designed an experiment with the following steps:

1. Generate training data from a known HMM structure, specifically the 3-state HMM described in the "Data-Generating Process and MSP" section below.
2. Train a transformer on this data to perform next-token prediction. In the experiments shown here we use a 4-layer transformer with 64 dimensional residual stream, and 4 attention heads per layer.
3. Analyze the final layer of the transformer's residual stream to look for a linear subspace with a geometry matching the predicted fractal structure of the Mixed-State Presentation (MSP).

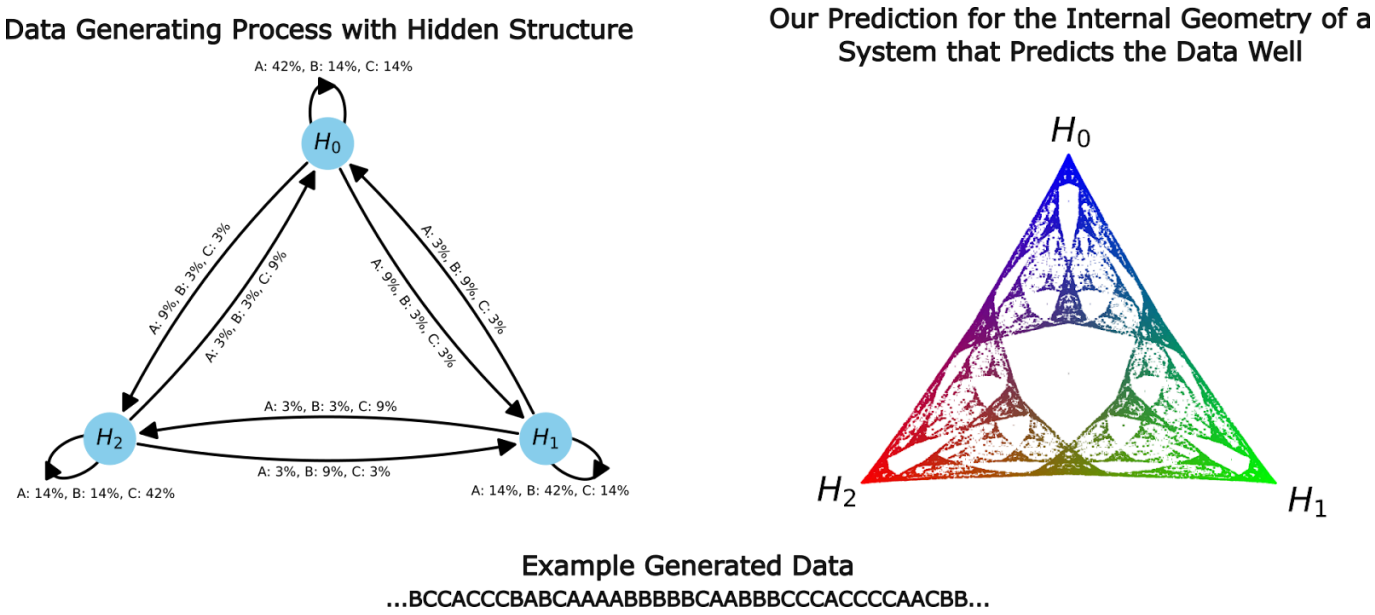
By controlling the structure of the training data using an HMM, we can make concrete, falsifiable predictions about the computational structure the transformer should implement during inference. Computational Mechanics, as presented in the "Theoretical Framework" section above, provides the framework for making these predictions based on the HMM's structure.

The specific HMM we chose has an MSP with an infinite fractal geometry, serving as a highly non-trivial prediction about what we should find in the transformer's residual stream activations if our theory is correct.

The Data-Generating Process and MSP

For this experiment we trained a transformer on data generated by a simple HMM, called the Mess3 Process, that has just 3 hidden states[8]. Moving between the 3 hidden states according to the emission probabilities on the edges generates strings over a 3-token

vocabulary: {A, B, C}. The HMM for this data-generating process is given on the left of the figure below.



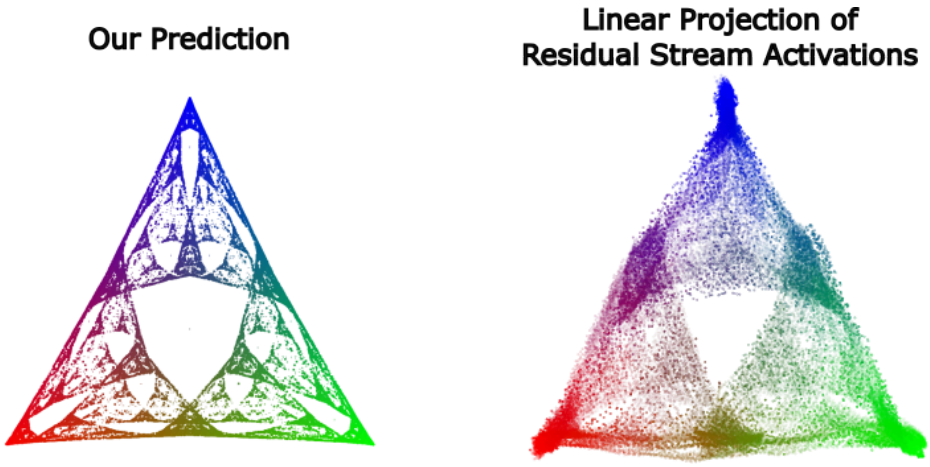
(Left) The data-generating process has 3 hidden states and outputs data made of a token-vocabulary of {A, B, C}. **(Bottom)** Paths through this structure generate training data, by sampling the token distributions of edges leaving a hidden state. We use this data to then train a transformer. **(Right)** The Mess3 MSP of internal states of a system that predicts future tokens of the data generating process based on observing previous tokens. Points in this space correspond to probability distributions over the hidden states of the data generating process, and thus lie in a 2D plane, since probability distributions over 3 objects are 2D. **Importantly, this structure is not the structure of the next-token predictions! It is instead the meta-structure of an observer's belief updates over the hidden states of the generating structure!** The middle point of the triangle corresponds to maximum uncertainty over the 3 hidden states, while corners correspond to total certainty in one of the hidden states. Colors are assigned by treating the 3D probability distributions as RGB values.

Our approach allows us to make rigorous and testable predictions about the internal structures of transformers. In the case of this HMM, the theory (outlined above) says that transformers trained on this data should instantiate the computational structure associated with the fractal geometry shown on the right of the figure above. Every colored point in the simplex on the above right panel is a distinct belief state.

We chose the Mess3 HMM because it's MSP has an infinite fractal structure, and thus acts as a highly-nontrivial prediction about what geometry we should find in the residual stream.

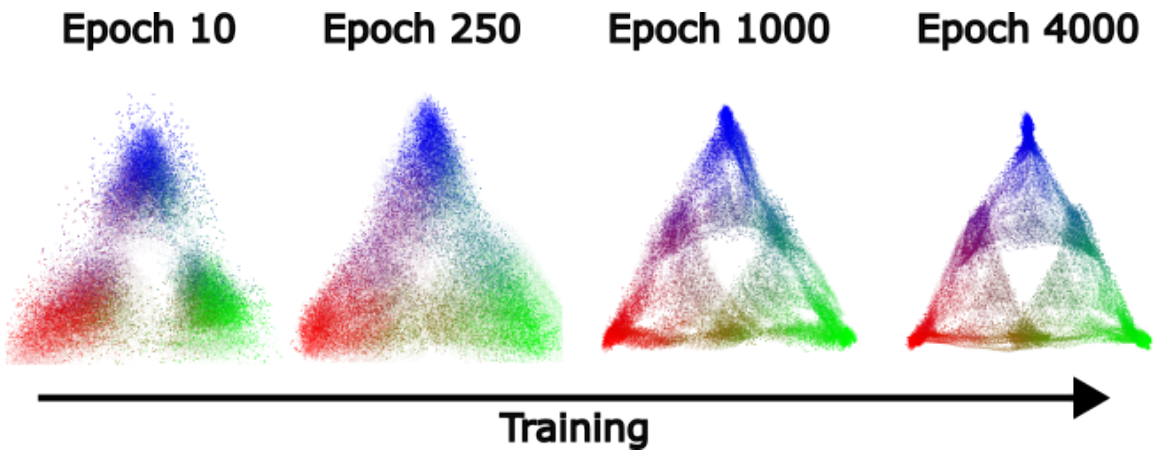
The Results!

We train a transformer on data generated by the Mess3 HMM. We look in the final layer of the residual stream and find a linear 2D subspace where activations have a structure remarkably similar to that of our predicted fractal. We do this by performing standard linear regression from the residual stream activations (64 dimensional vectors) to the belief distributions (3 dimensional vectors) which associated with them in the MSP.



(Left) The prediction we make for the internal geometry of the trained transformer, as shown in Figure 1. **(Right)** The experimental results. We find a 2D linear projection of the final residual stream activations of our trained transformer whose geometry recapitulates our theoretical prediction! Colors are assigned according to the ground truth belief distributions (as shown on the left).

We can also look at how this structure emerges over training, which shows (1) that the structure we find is not trivial[9] since it doesn't exist in detail early in training, and (2) the step-wise refinement of the transformers activations to the fractal structure we predict.



Over training we see the restructuring of the transformers internal activations in the residual stream refine to the fractal geometry predicted by our framework.

A movie of this process is shown below. Because we used Stochastic Gradient Descent for training, the 2D projection of the activations wiggles, even after training has converged. In

this wiggling you can see that fractal structures remain intact.

Limitations and Next Steps

Limitations

- Presented here was one simple data structure given by an HMM with 3 states, with a vocab size of 3. In practice, the LLMs that currently exist are much larger, have vocab sizes of >50,000, and natural language has infinite Markov order. Though we've tested this theory on other HMMs and everything continues to work, all tests so far have been on similarly small examples. How this relates to larger, more complicated, and more realistic settings is unknown (but we have ideas!).
- Though we haven't focused on it in this post, the MSP is an input-driven dynamical system. For each possible input in the system, we have dynamics that determine where in the belief simplex one should move to given the current belief. We have not explicitly tested that the LLM instantiates these dynamics, and instead have only tested that the belief states and their geometry is represented in the transformer.
- Computational Mechanics is primarily a story about *optimal* prediction. LLMs in practice won't be literally optimal. A number of papers exist studying near-

optimality, non-optimality, and rate-distortion phenomenon from the point of view of Computational Mechanics, but applying that to LLMs has not been done.

- We have focused on ergodic and stationary processes in the work presented here. Computational Mechanics can relax those assumptions, but again, we have not applied those (very interesting!) extensions of Computational Mechanics to LLMs. Non-ergodicity, in particular, is likely at the heart of in-context learning.
- In the experiment presented in this post we focussed on the final layer of the residual stream, right before the unembedding. In other experiments we've run (not presented here), the MSP is not well-represented in the final layer but is instead spread out amongst earlier layers. We think this occurs because in general there are groups of belief states that are degenerate in the sense that they have the same next-token distribution. In that case, the formalism presented in this post says that even though the distinction between those states must be represented in the transformers internal, the transformer is able to lose those distinctions for the purpose of predicting the next token (in the local sense), which occurs most directly right before the unembedding.

Next Steps

- We are hopeful that the framing presented in this post provides a formal handle on data structure, internal network structure, and network behavior.
- There are many open questions about how this work relates to other technical AI Safety work. I'll list a few ideas very quickly, and expand on these and more in future posts. In particular:
  - What is the relationship between *features* and *circuits*, as studied in Mechanistic Interpretability, and the Mixed-State Geometry?
  - Is there a story about superposition and compression of the MSP in cases where the residual stream is too small to "fit" the MSP.
  - Can we relate the development of MSP geometric structure over training to phenomenon in SLT? see Towards Developmental Interpretability°
  - Can we use our formalism to operationalize particular capabilities (in-context learning, out-of-distribution generalization, situational awareness, sleeper agents, etc.) and study them in toy models from our framework?
  - Can we use our formalism to understand task structure and how distinct tasks relate to each other? see A starting point for making sense of task structure (in

machine learning)°

- As mentioned in the Limitations section, how MSP structures in transformers divide across the layers of the transformer, and how the functional form of the attention mechanism relates to that is an obvious next step.
- We will be releasing a python library soon to be able to perform these types of experiments. Here is the github repo.
- Computational Mechanics is a well-developed framework, and this post has only focused on one small section of it. In the future we hope to bring other aspects of it to bear on neural networks and safety issues, and also to expand Computational Mechanics and combine it with other methods and frameworks.
- If you're interested in learning more about Computational Mechanics, we recommend starting with these papers: Shalizi and Crutchfield (2000), Riechers and Crutchfield (2018a), and Riechers and Crutchfield (2018b)
- We (Paul and Adam) have received funding to start a **new AI Safety research org, called Simplex!** Presented here was one small facet of the type of work we hope to do, and very much only the beginning. Stay tuned for posts that outline our broader vision in the future.
- In about a month we will be teaming up with Apart to run a Hackathon! We will post about that soon as well, alongside an open problems post, and some more resources to run experiments.
- There's a lot of work to do going forward! This research plan has many parts that span the highly mathematical/theoretical to experimental. If you are interested in being a part of this please have a low threshold for reaching out!

---

1. ^ PIBBSS is hiring°! I wholeheartedly recommend them as an organization.

2. ^ One way to conceptualize this is to think of "the world" as having some hidden structure (initially unknown to you), that emits observables. Our task is then to take sequences of observables and infer the hidden structure of the world - maybe in the service of optimal future prediction, but also maybe just because figuring out how the world works is inherently interesting. Inside of us, we have a "world model" that serves as the internal structure that let's us "understand" the hidden structure of the world. The term world model is contentious and nothing in this post depends on that concept much. However, one motivation for this work is to formalize and make concrete statements about peoples intuitions and arguments regarding neural networks and world models - which are often handwavy and ill-defined.
3. ^ Technically speaking, the term *process* refers to a probability distribution over infinite strings of tokens, while a *presentation* refers to a particular HMM that produces strings according to the probability distribution. A process has an infinite number of presentations.
4. ^ Any HMM defines a probability distribution over infinite sequences of the emissions.
5. ^ Our initial belief distribution, in this particular case, is the uniform distribution over the 3 states of the data generating process. However this is not always the case. In general the initial belief distribution is given by the stationary distribution of the data generating HMM.
6. ^ You can find the answer in section IV of this paper by @Paul Riechers.
7. ^ There is work in Computational Mechanics that studies non-optimal or near-optimal prediction, and the tradeoffs one incurs when relaxing optimality. This is likely relevant to neural networks in practice. See Marzen and Crutchfield 2021 and Marzen and Crutchfield 2014.
8. ^ This process is called the mess3 process, and was defined in a paper by Sarah Marzen and James Crutchfield. In the work presented we use x=0.05, alpha=0.85.
9. ^ We've also run another control where we retain the ground truth fractal structure but shuffle which inputs corresponds to which points in the simplex (you can think of this as shuffling the colors in the ground truth plot). In this case when we run our regression we get that every residual stream activation is mapped to the center point of the simplex, which is the center of mass of all the points.

PIBBSS 2 | Interpretability (ML & AI) 1 | Transformers 1 | AI 1 | Curated

Mentioned in

180 Why Would Belief-States Have A Fractal Structure, And Why Would That Matter For Interpretability? An Explainer

163 Announcing ILIAD — Theoretical AI Alignment Conference

154 Language Models Model Us

34 Computational Mechanics Hackathon (June 1 & 2)

100 comments, sorted by top scoring

Some comments are truncated due to high volume. (⌘F to expand all) Change truncation settings

[-] **Rohin Shah** 2mo Ω 33 < 55 > × 12 ✓

Is it accurate to summarize the headline result as follows?

- Train a Transformer to predict next tokens on a distribution generated from an HMM.
- One optimal predictor for this data would be to maintain a belief over which of the three HMM states we are in, and perform Bayesian updating on each new token. That is, it maintains $p(\text{hidden state} = H_i)$.
- **Key result:** A linear probe on the residual stream is able to reconstruct $p(\text{hidden state} = H_i)$.

(I don't know what Computational Mechanics or MSPs are so this could be totally off.)

EDIT: Looks like yes. From this post°:

> Part of what this all illustrates is that the fractal shape is kinda… baked into any Bayesian-ish system tracking the hidden state of the Markov model. So in some sense, it's not very surprising to find it linearly embedded in activations of a residual stream; all that really means is that the probabilities for each hidden state are linearly represented in the residual stream.

✓ 1 ! 1

[-] **Adam Shai** 2mo Ω 7 < 11 > × 3 ✓

That is a fair summary.

3 **eggsyntax** 2mo As well as inferring the HMM itself from the data.

[-] **Chris_Leong** 2mo < 30 > × 14 ✓

"The structure of synchronization is, in general, richer than the world model itself. In this sense, LLMs learn more than a world model" given that I expect this is the statement that will catch a lot of people's attention.

Just in case this claim caught anyone else's attention, what they mean by this is that it contains:
• A model of the world
• A model of the agent's process for updating its belief about which state the world is in

3 **snewman** 2mo I am trying to wrap my head around the high-level implications of this statement. I can …

2 **AlbertGarde** 2mo You are drawing a distinction between agents that maintain a probability distributio…

2 **snewman** 2mo As I go about my day, I need to maintain a probability distribution over states of the …

1 **Brent** 1mo I agree with you that the LLM's job is harder, but I think that has a lot to do with the tas…

1 **snewman** 1mo I think we're saying the same thing? "The LLM being given less information [abou…

[-] **ChrisCundy** 2mo 16 3

The figures remind me of figures 3 and 4 from *Meta-learning of Sequential Strategies, Ortega et al 2019,* which also study how autoregressive models (RNNs) infer underlying structure. Could be a good reference to check out!

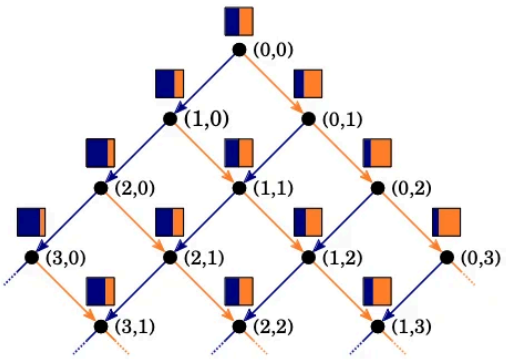


Figure 3. Minimal state machine for a predictor of coin tosses with a fixed, unknown bias. The hypothesis class can be modeled as a 2-sided coin (see Example 1). Dark and light state transitions correspond to observing the outcomes 'Head' and 'Tail' respectively, and the states are annotated with $(n_{\mathrm{H}}, n_{\mathrm{T}})$, the number of times Head and Tail have been observed. The predictions made from each state are shown in the stacked bar charts: the probability of Head is $P(x_t = \mathrm{H}|x_{<t}) = \frac{n_{\mathrm{H}}+1}{t+2}$ (which is how these predictions are implemented in a computer program). Note how different observations sequences can lead to the same state (e.g. HT and TH).

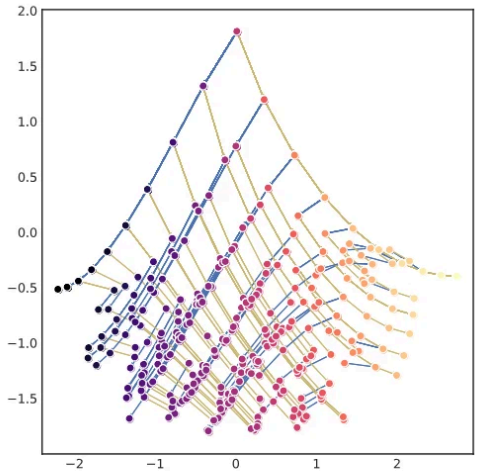


Figure 4. Meta-learned state machine for a predictor of coin tosses. The figure shows the memory dynamics of a standard memory-based predictor projected onto the first two eigenvectors. Notice the striking similarity with Figure 3. The predictor consists of 20 LSTM cells with softmax predictions, which was trained using Algorithm 1 on 1000 batches of 100 rollouts, where rollouts were of length 10. For training, we used the Adam optimization algorithm [Kingma and Ba, 2014].

2 **Adam Shai** 2mo this looks highly relevant! thanks!

2 **Ran W** 2mo This reminds me of the paper Chris linked as well. I think there's very solid evidence on th…

7 **gwern** 2mo My earlier comment on meta-learning and Bayesian RL/inference for background: https:…

[-] **johnswentworth** 2mo 16 -2

[EDIT: I no longer endorse this response, see thread.]

(This comment is mainly for people other than the authors.)

If your reaction to this post is "hot damn, look at that graph", then I think you should probably dial back your excitement somewhat. IIUC the fractal structure is largely an artifact of how the data is visualized, which means the results visually look more striking than they really are.

It is still a cool piece of work, and the visuals are beautiful. The correct amount of excitement is greater than zero.

7

[-] **Vladimir_Nesov** 2mo 11 6

To me the consequences of this response were more valuable than the-post-without-this-response, since it led to the clarification° by the post's author on a crucial point that wasn't clear in the post and reframed it

substantially. And once that clarification arrived, this thread ceased being highly upvoted, which seems the opposite of the right thing to happen.

> I no longer endorse this response

(So it's a case where value of content in hindsight disagrees with value of the consequences of its existence. Doesn't even imply there was originally an error, without the benefit of hindsight.)

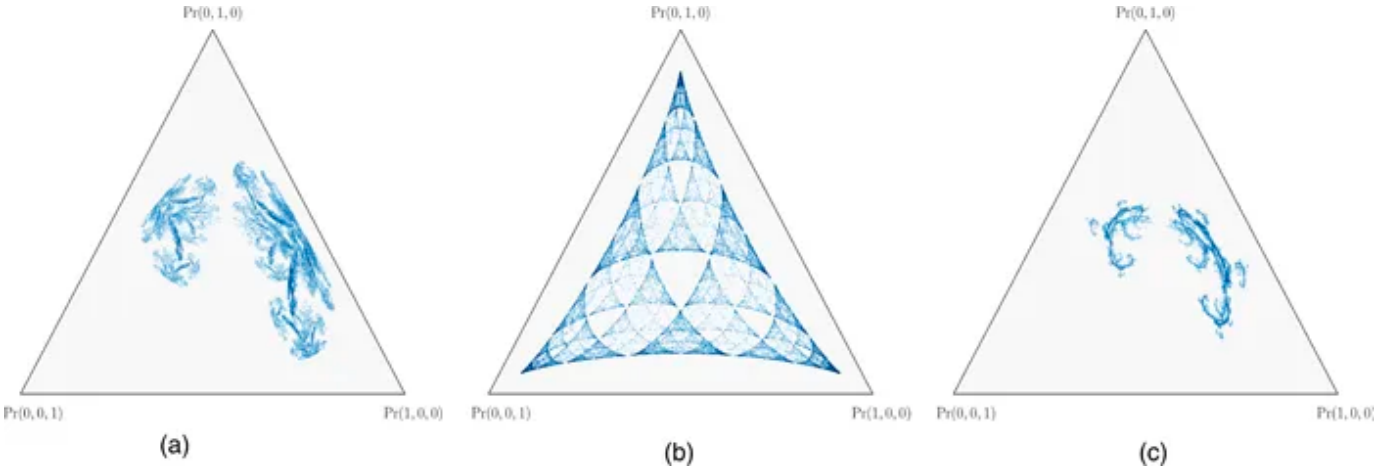
[-] **Adam Shai** 2mo 10 0

Can you elaborate on how the fractal is an artifact of how the data is visualized?

From my perspective, the fractal is there because we chose this data generating structure precisely because it has this fractal pattern as it's Mixed State Presentation (ie. we chose it because then the ground truth would be a fractal, which felt like highly nontrivial structure to us, and thus a good falsifiable test that this framework is at all relevant for transformers. Also, yes, it is pretty :)). The fractal is a natural consequence of that choice of data generating structure - it is what Computational Mechanics says is the geometric structure of synchronization for the HMM. That there is a linear 2d plane in the residual stream that when you project onto it you get that same fractal seems highly non-artifactual, and is what we were testing.

Though it should be said that an HMM with a fractal MSP is a quite generic choice. It's remarkably easy to get such fractal structures. If you randomly chose an HMM from the space of HMMs for a given number of states and vocab size, you will often get synchronizations structures with infinite transient states and fractals.

This isn't a proof of that previous claim, but here are some examples of fractal MSPs from https://arxiv.org/abs/2102.10487:



6 **johnswentworth** 2mo I don't know the details of the MSP, but my current understanding is that it's a …

[-] **Adam Shai** 2mo 25 0

Responding in reverse order:

> If there's literally a linear projection of the residual stream into two dimensions which directly produces that fractal, with no further processing/transformation in between "linear projection" and

> "fractal", then I would change my mind about the fractal structure being mostly an artifact of the visualization method.

There is literally a linear projection (~~well, we allow a constant offset actually, so affine~~) of the residual stream into two dimensions which directly produces that fractal. There's no distributions in the middle or anything. I ~~suspect the offset is not necessary but I haven't checked ::adding to to-do list::~~

edit: the offset isn't necessary. There is literally a linear projection of the residual stream into 2D which directly produces the fractal.

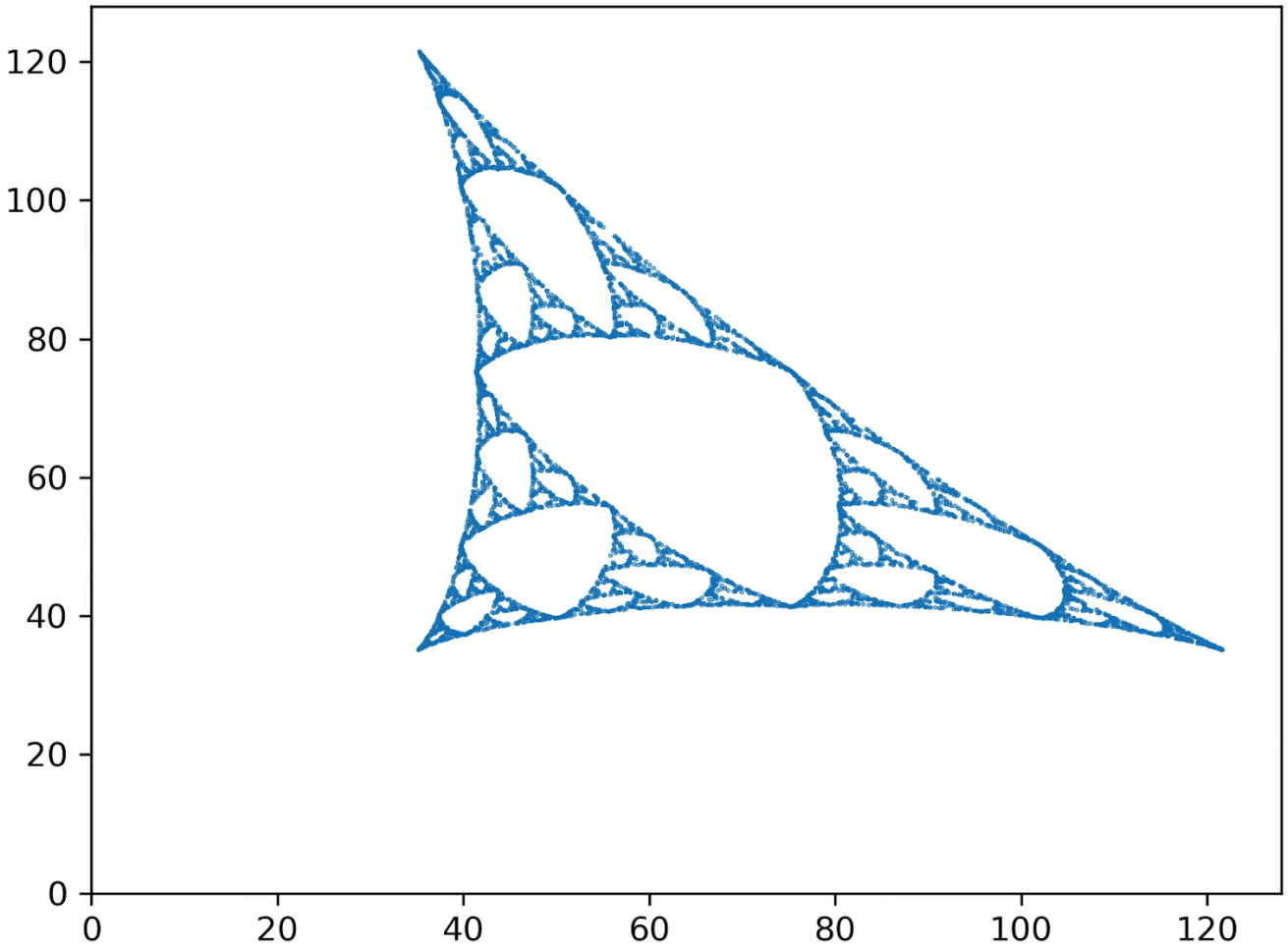
> But the "fractal-ness" is mostly an artifact of the MSP as a representation-method IIUC; the stochastic process itself is not especially "naturally fractal".
>
> (As I said I don't know the details of the MSP very well; my intuition here is instead coming from some background knowledge of where fractals which look like those often come from, specifically chaos games.)

I'm not sure I'm following, but... (read more)

1 1

[-] **johnswentworth** 2mo 15 0

We're now working through understanding all the pieces of this, and we've calculated an MSP which doesn't *quite* look like the one in the post:



(Ignore the skew, David's still fiddling with the projection into 2D. The important noticeable part is the absence of "overlap" between the three copies of the main shape, compared to the fractal from the post.)

Specifically, each point in that visual corresponds to a distribution $(P[H^t = H_0|O^{<t}], P[H^t = H_1|O^{<t}], P[H^t = H_2|O^{<t}])$ for some value of the observed symbols O. The image itself is of the points on the probability simplex. From looking at a couple of Crutchfield papers, it sounds like that's what the MSP is supposed to be.

The update equations are:

- $P[H^{t+1}|O^{\leq t}] = \sum_{H^t} P[H^{t+1}|H^t]P[H^t|O^{\leq t}]$
- $P[H^t|O^{\leq t}] = \frac{1}{Z}P[O^t|H^t]P[H^t|O^{<t}]$

with $P[H^{t+1}|H^t]$ given by the transition probabilities, $P[O^t|H^t]$ given by the observation probabilities, and Z a normalizer. We generate the image above by running initializing some random distribution $P[H^0]$, then iterating the equations and plotting each point.

Off the top of your head, any idea what might account for the mismatch (other than a bug in our code, which we're alread... (read more)

[-] **Adam Shai** 2mo 10 0

Everything looks right to me! This is the annoying problem that people forget to write the actual parameters they used in their work (sorry).

Try x=0.05, alpha=0.85. I've edited the footnote with this info as well.

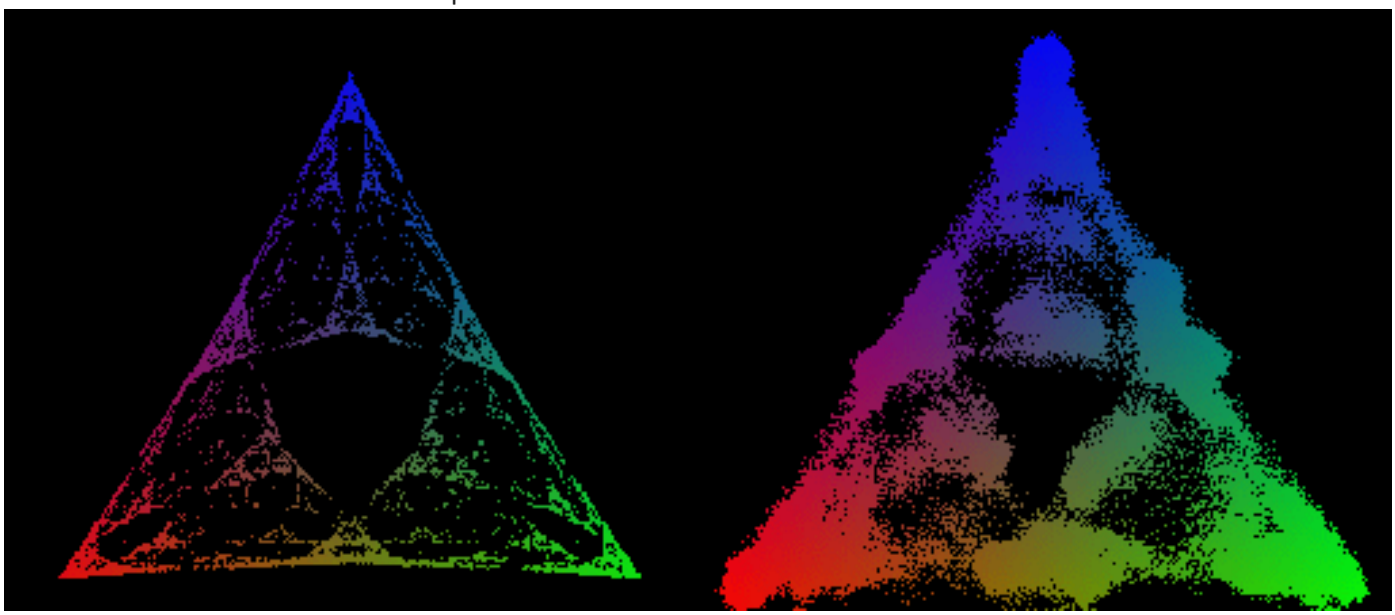
3 **johnswentworth** 2mo Yup, that was it, thankyou!

5 **Adam Shai** 2mo Oh wait one thing that looks not quite right is the initial distribution. Instead of startin…

1 **Jett** 1mo For the two sets of mess3 parameters I checked the stationary distribution was uniform.

[-] **Jett** 1mo 14 0

This is such a cool result! I tried to reproduce it in this notebook



[-] **aysja** 2mo 13 0

This is very cool! I'm excited to see where it goes :)

A couple questions (mostly me grappling with what the implications of this work might be):

- Given a dataset of sequences of tokens, how do you find the HMM that could have generated it, and can this be done automatically? Also, is the mapping from dataset to HMM unique?
- This question is possibly more confused on my end, sorry if so. I'm trying to get at something like "how interpretable will these simplexes be with much larger models?" Like, if I'm imagining that each state is a single token, and the HMM is capable of generating the totality of data the model sees, then I'm imagining something quite unwieldy, i.e., something with about the amount of complexity and interpretability as, e.g., the signaling cascade networks in a cell. Is this imagination wrong? Or is it more like, you start with this unwieldy structure (but which has some nice properties nonetheless), and then from there you try to make the initial structure more parse-able? Maybe a more straightforward way to ask: you say you're interested in formalizing things like situational awareness with these tools—how might that work?

[-] **Adam Shai** 2mo 11 0

Thanks!

- one way to construct an HMM is by finding all past histories of tokens that condition the future tokens with the same probablity distribution, and make that equivalence class a hidden state in your HMM. Then the conditional distributions determine the arrows coming out of your state and which state you go to next. This is called the "epsilon machine" in Comp Mech, and it is unique. It is one *presentation* of the data generating process, but in general there are an infinite number of HMM presntations that would generate the same data. The epsilon machine is a particular type of HMM presentation - it is the smallest one where the hidden states are the minimal sufficient statistics for predicting the future based on the past. The epsilon machine is one of the most fundamental things in Comp Mech but I didn't talk about it in this post. In the future we plan to make a more generic Comp Mech primer that will go through these and other concepts.
- The interpretability of these simplexes is an issue that's in my mind a lot these days. The short answer is I'm still wrestling with it. We have a rough experimental plan to go about studying this issue but for now, here are some related que

... (read more)

1 **lillybaeum** 2mo The following is text from Claude Opus 3. I generally find people just dumping answer…

4 **Alexander Gietelink Oldenziel** 2mo Not at all cringe! This is the age of AI. We either channel its imm…

[-] **habryka** 2mo Ω 3 9 1

Promoted to curated: Formalizing what it means for transformers to learn "the underlying world model" when engaging in next-token prediction tasks seems pretty useful, in that it's an abstraction that I see used all the time when discussing risks from models where the vast majority of the compute was spent in pre-training, where the details usually get handwaived. It seems useful to understand what exactly we mean by that in more detail.

I have not done a thorough review of this kind of work, but it seems to me that also others thought the basic ideas in the work hold up, and I thought reading this post gave me crisper abstractions to talk about this kind of stuff in the future.

1

[-] **Aprillion (Peter Hozák)** 2mo 9 4

> transformer is only trained explicitly on next token prediction!

I find myself understanding language/multimodal transformer capabilities better when I think about the whole document (up to context length) as a mini-batch for calculating the gradient in transformer (pre-)training, so I imagine it is minimizing the document-global prediction error, it wasn't trained to optimize for just a single-next token accuracy...

[-] **Garrett Baker** 2mo 14 9

There is evidence that transformers are *not* in fact even implicitly, internally, optimized for reducing global prediction error (except insofar as comp-mech says they must in order to do well on the task they are optimized for).

> Do transformers "think ahead" during inference at a given position? It is known transformers prepare information in the hidden states of the forward pass at t that is then used in future forward passes t+τ. We posit two explanations for this phenomenon: pre-caching, in which off-diagonal gradient terms present in training result in the model computing features at t irrelevant to the present inference task but useful for the future, and breadcrumbs, in which features most relevant to time step t are already the same as those that would most benefit inference at time t+τ. We test these hypotheses by training language models without propagating gradients to past timesteps, a scheme we formalize as myopic training. In a synthetic data setting, we find clear evidence for pre-caching. In the autoregressive language modeling setting, our experiments are more suggestive of the breadcrumbs hypothesis.

[-] **Erik Jenner** 2mo 12 7

I think that paper is some evidence that there's typically no huge effect from internal activations being optimized for predicting future tokens (on natural language). But I don't think it's much (if any) evidence that this doesn't happen to some small extent or that it couldn't be a huge effect on certain other natural language tasks.

(In fact, I think the myopia gap is probably the more relevant number than the local myopia bonus, in which case I'd argue the paper actually shows a pretty non-trivial effect, kind of contrary to how the authors interpret it. But I haven't read the paper super closely.)

Also, sounds like you're aware of this, but I'd want to highlight more that the paper does demonstrate internal activations being optimized for predicting future tokens on synthetic data where this is necessary. So, arguably, the main question is to what extent natural language data incentivizes this rather than being specifically about what transformers can/tend to do.

In that sense, thinking of transformer internals as "trying to" minimize the loss on an entire document might be exactly the right intuition empirically (and the question is mainly how different that is from being myopic on a given dataset). Given that the internal states are optimized for this, that would also make sense theoretically IMO.

1

7 **ryan_greenblatt** 2mo +1 to this comment, also I expect the importance of activations being optimiz…

5 **Aleksey Bykhun** 2mo I have tried to play with Claude – I would ask it to think of a number, drop the hi…

2 **Garrett Baker** 2mo Post the chat logs?

2 **Adam Shai** 2mo That's an interesting framing. From my perspective that is still just local next-token acc…

1 **Aprillion (Peter Hozák)** 2mo To me as a programmer and not a mathematitian, the distinction doesn't…

[-] **cousin_it** 2mo Ω 3 7 0

I have maybe a naive question. What information is needed to find the MSP image within the neural network? Do we have to know the HMM to begin with? Or could it be feasible someday to inspect a neural network, find something that looks like an MSP image, and infer the HMM from it?

[-] **Leon Lang** 2mo 7 3

I really enjoyed reading this post! It's quite well-written. Thanks for writing it.

The only critique is that I would have appreciated more details on how the linear regression parameters are trained and what exactly the projection is doing. John's thread is a bit clarifying on this.

One question: If you optimize the representation in the residual stream such that it corresponds to a particular chosen belief state, does the transformer than predict the next token *as if* in that belief state? I.e., does the transformer use the belief state for making predictions?

1 **Adam Shai** 2mo Thanks! I appreciate the critique. From this comment and from John's it seems correct…

1 **Leon Lang** 2mo Yes the first! Thanks for the link!

[-] **eggsyntax** 2mo 6 0

I struggled with the notation on the figures; this comment tries to clarify a few points for anyone else who may be confused by it.

- There are three main diagrams to pay attention to in order to understand what's going on here:
  - The Z1R Process (this is a straightforward Hidden Markov Model diagram, look them up if it's unclear).
  - The Z1R Mixed-State Presentation, representing the belief states of a model as it learns the underlying structure.
  - The Z1R Mixed-State Simplex. Importantly, unlike the other two this is a graph and spatial placement is meaningful.
- It's b

... (read more)

1 **Adam Shai** 2mo This all looks correct to me! Thanks for this.

[-] **ChosunOne** 2mo 6 0

I'm curious how much space is left after learning the MSP in the network. Does representing the MSP take up the full bandwidth of the model (even if it is represented inefficiently)? Could you maintain performance of the model by subtracting out the contributions of anything else that isn't part of the MSP?

1 **Adam Shai** 2mo Cool question. This is one of the things we'd like to explore more going forward. We ar…

[-] **dr_s** 2mo 5 0

This is extremely cool! Can you go into more detail about the step used to project the 64 dimensional residual stream to 3 dimensional space? Did you do a linear fit over a few test points and then used it on all the others?

[-] **Nina Rimsky** 2mo 5 0

This is really cool work!!

> In other experiments we've run (not presented here), the MSP is not well-represented in the final layer but is instead spread out amongst earlier layers. We think this occurs because in general there are groups of belief states that are degenerate in the sense that they have the same next-token distribution. In that case, the formalism presented in this post says that even though the distinction between those states must be represented in the transformers internal, the transformer is able to lose those distinctions for the purpose

... (read more)

1 **Adam Shai** 2mo Thanks! I'll have more thorough results to share about layer-wise reprsentations of the...

[-] **Sandi** 2mo 5 0

> We do this by performing standard linear regression from the residual stream activations (64 dimensional vectors) to the belief distributions (3 dimensional vectors) which associated with them in the MSP.

I don't understand how we go from this to the fractal. The linear probe gives us a single 2D point for every forward pass of the transformer, correct? How do we get the picture with many points in it? Is it by sampling from the transformer while reading the probe after every token and then putting all the points from that on one graph?

Is this result equiva... (read more)

3 **Adam Shai** 2mo I should have explained this better in my post. For every input into the transformer (of ...

2 **dr_s** 2mo Given that the model eventually outputs the next token, shouldn't the final embedding mat...

1 **Sandi** 2mo Yep, that's what I was trying to describe as well. Thanks!

[-] **Fiora from Rosebloom** 2mo 3 0

> We look in the final layer of the residual stream and find a linear 2D subspace where activations have a structure remarkably similar to that of our predicted fractal. We do this by performing standard linear regression from the residual stream activations (64 dimensional vectors) to the belief distributions (3 dimensional vectors) which associated with them in the MSP.

Naive technical question, but can I ask for a more detailed description of how you go from the activations in the residual stream to the map you have here? Or like, can someone point m... (read more)

[-] **Sodium** 1mo 2 1

I thought that the part about models needing to keep track of a more complicated mix-state presentation as opposed to just the world model is one of those technical insights that's blindingly obvious once someone points it out to you (i.e., the best type of insight :)). I love how the post starts out by describing the simple ZIR example to help us get a sense of what these mixed state presentations are like. Bravo!

1 **Adam Shai** 1mo Thanks! In my experience Computational Mechanics has many of those types of techni…

[-] **Alexander Gietelink Oldenziel** 2mo 2 0

Non exhaustive list of reasons one could be interested in computational mechanics:
https://www.lesswrong.com/posts/GG2NFdgtxxjEssyiE/dalcy-s-shortform?commentId=DdnaLZmJwusPkGn96°

[-] **Moughees Ahmed** 2mo 2 0

This might be an adjacent question but assuming this is true and comprehensively explains the belief updating process. What does it say, if anything, about whether transformers can produce new (undiscovered) knowledge/states? If they can't observe a novel state - something that doesn't exist in the data - can they never discover new knowledge on their own?

1 **Adam Shai** 2mo This is a great question, and one of the things I'm most excited about using this frame…

1 **Moughees Ahmed** 2mo Excited to see what you come up with! Plausibly, one could think that if a mo…

[-] **Review Bot** 2mo 2 0

The LessWrong Review runs every year to select the posts that have most stood the test of time. This post is not yet eligible for review, but will be at the end of 2025. The top fifty or so posts are featured prominently on the site throughout the year.

Hopefully, the review is better than karma at judging enduring value. If we have accurate prediction markets on the review results, maybe we can have better incentives on LessWrong today. Will this post make the top fifty?°

[-] **Exa Watson** 2mo 2 0

If I understand this right, you train a transformer on data generated from a hidden markov process, of the form {0,1,R} and find that there is a mechanism for tracking when R occurs in the residual stream, as well as that the transformer learns the hidden markov process. is that correct?

4 **Keenan Pepper** 2mo No, the actual hidden Markov process used to generate the awesome triangle frac…

[-] **Nisan** 2mo 2 0

If I understand correctly, the next-token prediction of Mess3 is related to the current-state prediction by a nonsingular linear transformation. So a linear probe showing "the meta-structure of an observer's belief updates over the hidden states of the generating structure" is equivalent to one showing "the structure of the next-token predictions", no?

3 **Nisan** 2mo I suppose if you had more hidden states than observables, you could distinguish hidden-stat…

[-] **p.b.** 2mo 2 0

This reminds me a lot of a toy project I have in the back of my mind but will probably never get around to:

Which is to train a transformer on the sequences generated by the logic models from the apperception engine paper (which in the paper are inferred by the apperception engine from the sequences) with the aim of predicting the logic model.

2 **Adam Shai** 2mo That sounds interesting. Do you have a link to the apperception paper?

3 **p.b.** 2mo https://www.sciencedirect.com/science/article/pii/S0004370220301855#se0050 https://w…

3 **MondSemmel** 2mo This book chapter and this paper, maybe?

1 **p.b.** 2mo Hah, I didn't see your answer but our links complement nicely. I think my first link was the…

[-] **Vladimir_Nesov** 2mo 2 0

This is interesting as commentary on superposition, where activation vectors with N dimensions can be used to represent many more concepts, since the N-dimensional space/sphere can be partitioned into many more regions than N, each with its own meaning. If similar fractal structure substantially occurs in the original activation bases (such as the Vs of attention, as in the V part of KV-cache) and not just after having been projected to dramatically fewer dimensions, this gives a story for role of nuance that improves with scale that's different from it be... (read more)

[-] **Aprillion (Peter Hozák)** 2mo 2 0

Can you help me understand a minor labeling convention that puzzles me? I can see how we can label S_R from the Z1R process as η_{11} in MSP because we observe 11 to get there, but why S_1 is labeled as η_{01} after observing either 100 or 00, please?

2 **Adam Shai** 2mo Good catch! That should be eta_00, thanks! I'll change it tomorrow.

[-] **Dalcy** 2mo 2 0

What is the shape predicted by compmech under a generation setting, and do you expect it instead of the fractal shape to show up under, say, a GAN loss? If so, and if their shapes are sufficiently distinct from the controls that

are run to make sure the fractals aren't just a visualization artifact, that would be further evidence in favor of the applicability of compmech in this setup.

2 **Adam Shai** 2mo Cool idea! I don't know enough about GANs and their loss so I don't have a prediction ...

[-] **Laurence Aitchison** 26d 1 0

One nice little prediction from this approach: you'd expect the first few tokens to have denser (as in SAE) features, as there is less context, so the "HMM" could be in a broad range of states. Whereas once you've seen more tokens, you have much more information so the state is pinned down more precisely and you'd expect to be denser.

There's also a big literature from computational neuroscience about how you represent probabilities. This is suggesting a "mean parameter code", where the LLM activations are a function of E[z| data]. But lots of other possibilities are available, e.g. see:

http://www.gatsby.ucl.ac.uk/teaching/courses/tn1-2021/slides/uncert-slides.pdf

[-] **Joel Ye** 1mo 1 0

Thanks for the post, it's neat to see the fields and terms existing for these questions.

I have two questions for hope of using this type of analysis in my work to analyze a lack of transfer between two distinct datasets A and B. (I see this is in your future work?)

1. Where does OOD data project, or data that is implausible for the model?

2. For more complex data, might we expect this MSP to most clearly show in places other than the final layer?

re: transfer, my hypothesis is that we might be able to see, having trained on A and B, that during inferenc... (read more)

[-] **Chipmonk** 1mo 1 0

this post seems like a win for PIBBSS gee

[-] **Steve Kommrusch** 1mo 1 0

This is very interesting work, showing the fractal graph is a good way to visualize the predictive model being learned. I've had many conversations with folks who struggle with the idea 'the model is just predicting the next token, how can it be doing anything interesting'?. My standard response had been that conceptually the transformer model matches up tokens at the first layer (using the key and query vectors), then matches up sentences a few layers up, and then paragraphs a few layers above that; hence the model, when presented with an input, was not j... (read more)

[-] **PoignardAzur** 2mo 1 0

> We think this occurs because in general there are groups of belief states that are degenerate in the sense that they have the same next-token distribution. In that case, the formalism presented in this post says that even though the distinction between those states must be represented in the transformers internal, the transformer is able to lose those distinctions for the purpose of predicting the next token (in the local sense), which occurs most directly right before the unembedding.

I wonder if you could force the Mixed-State Presentation to be "conse... (read more)

[-] **Daniel Munro** 2mo 1 0

Fascinating. But are these diagrams really showing HMMs? I thought each state in an HMM had a set of transition probabilities and another set of emission probabilities, which at each step are sampled independently. In these diagrams, the two processes are coupled. If "Even Ys" were a conventional HMM, S_E would sometimes emit X and transition to S_O, which would result in some even and some odd runs of Y. Are these a special variant of HMM, or some other type of state machine? And would these results apply to conventional HMMs with separate tr... (read more)

[-] **Oliver Sourbut** 2mo 1 0

Nice explanation of MSP and good visuals.

> This is surprising!

Were you in fact surprised? If so, why? (This is a straightforward consequence of the good regulator theorem[1].)

In general I'd encourage you to carefully track claims about transformers, HMM-predictors, and LLMs, and to distinguish between trained NNs and the training process. In this writeup, all of these are quite blended.

---

1. John has a good explication here° ↩

7 **kave** 2mo IIUC, the good regulator theorem doesn't say anything about how the model of the system s...

3 **Adam Shai** 2mo It's surprising for a few reasons: * The structure of the points in the simplex is NOT * T...

2 **Oliver Sourbut** 2mo I guess my question would be 'how else did you think a well-generalising sequen...

7 **Alexander Gietelink Oldenziel** 2mo I agree with you that the new/surprising thing is the linearity of ...

1 **Oliver Sourbut** 2mo Lol! I guess if there was a more precise theorem statement in the vicinity gest...

[-] **JoNeedsSleep** 2mo 1 0

Thank you for the insightful post! You mentioned that:

> Consider the relation a transformer has to an HMM that produced the data it was trained on. This is general - any dataset consisting of sequences of tokens can be represented as having been generated from an HMM.

and the linear projection consists of:

> Linear regression from the residual stream activations (64 dimensional vectors) to the belief distributions (3 dimensional vectors).

Given any natural language dataset, if we didn't have the ground truth belief distribution, is it possible to reverse engineer... (read more)

2 **Adam Shai** 2mo If I'm understanding your question correctly, then the answer is yes, though in practice…

[-] **Niclas Kupper** 2mo 1 0

Is there some theoretical result along the lines of "A sufficiently large transformer can learn any HMM"?

3 **Alexander Gietelink Oldenziel** 2mo Depending on what one means by 'learn' this is provably impossibl…

3 **Olli Järviniemi** 25d Huh, either I'm misunderstanding or this is wrong. If you have Hidden Markov M…

6 **Alexander Gietelink Oldenziel** 21d You are absolutely right and I am of course absolutely and embar…

1 **Alexander Gietelink Oldenziel** 25d Behold

1 **Niclas Kupper** 2mo Where can I read about this 2-state HMM? By learn I just mean approximate via a…

[-] **ProgramCrafter** 2mo 1 0

Speaking of next steps, I'd love to see a transformer that was trained to manipulate those states (given target state and interactor's tokens, would emit its own tokens for interleaving)! I believe this would look even cooler, and may be useful in detecting if AI starts to manipulate someone.

[-] **Keenan Pepper** 2mo 1 0

Can you share the hyperparameters used to make this figure?

1 **Keenan Pepper** 2mo Ah, never mind, I believe I found the relevant hyperparameters here: https://githu…

1 **Keenan Pepper** 2mo Actually I would still really appreciate the training hyperparameters like batch siz…

1 **tropea@gwu.edu** 2mo A simple suggestion on word usage: from "belief state" to "interpretive state…

1 **Keenan Pepper** 2mo I think you may have meant this as a top-level comment rather than a reply t…

Moderation Log