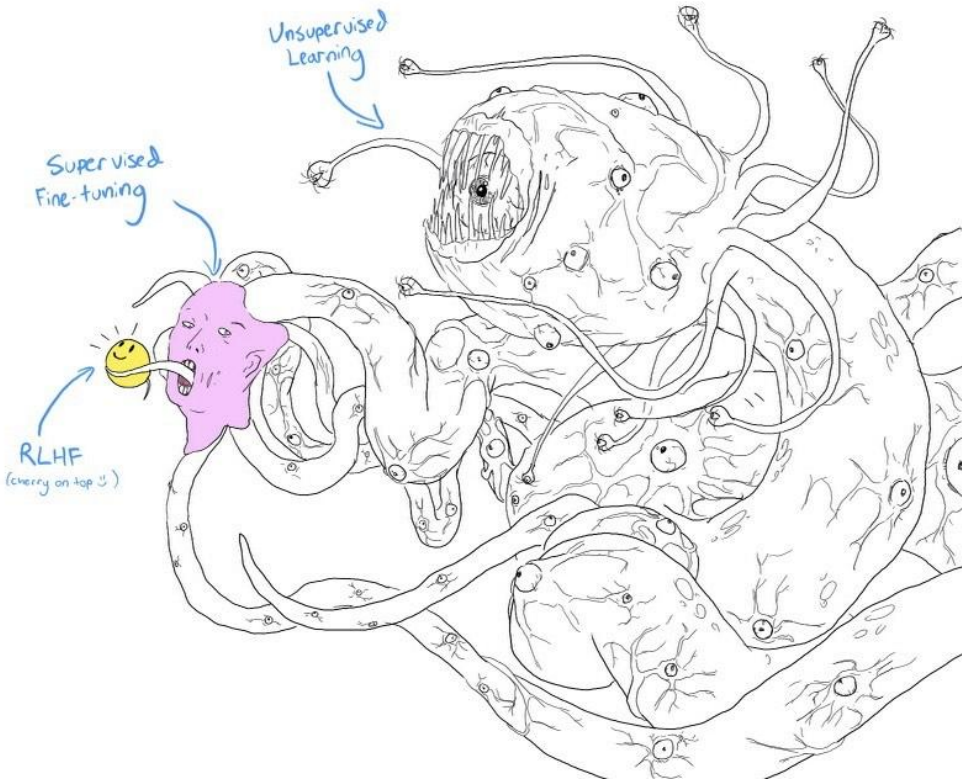
A Unifying Perspective on Language Model Interpretability

Enyan Zhang, Tom McCoy



Shoggoth with Smiley Face. Courtesy of twitter.com/anthrupad

Language Models

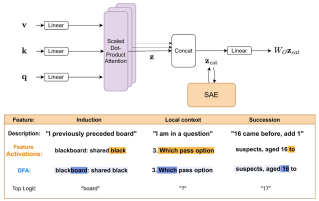
- are better understood! There's a growing body of research that seeks to interpret the inner workings of NNs.

1. Linear Probing — training a *linear classifier* from activations to features
2. Activating/Path Patching — *replacing* the output of model components
3. Steering (of activations/circuits) — *adding/subtracting* from activations
4. SAEs — *decomposing* activations into sparse, linear combinations

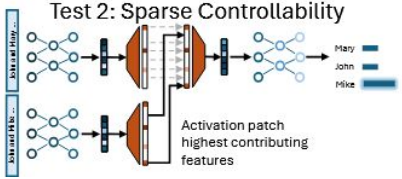
... and many more

What are we interpreting, exactly?

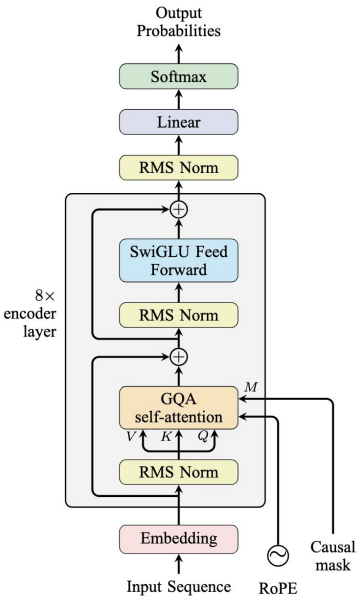
A wealth of different methods...

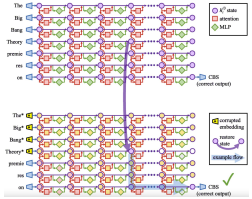


SAEs

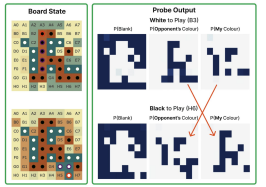


Steering





Patching



Probing

applied to the same model

What is nature of LMs,
such that all these methods work?

Roadmap

- Background — Interpreting NNs
  - Interpretability now has a wealth of techniques, is there a unifying picture?
- ***Tensor Product Representations***
- TPRs as a unifying architecture
  - TPRs → Additive Analogies
  - TPRs → Linear Probes
  - TPRs → SAEs
  - TPRs → Activation Patching

Primer: Tensor Product Representations

TPRs encode `(filler, role)` pairs into vectors

Domain: SVO Sentences

- fillers := {noun 1, noun 2, … verb 1, …}, roles := {subject, verb, object}*
- “The doctor will see the patient”
  ↔ {(doctor, subject), (see, verb), (patient, object)}

*A domain can have multiple filler/role schemes that are equally expressive

Primer: Tensor Product Representations

TPRs encode `(filler, role)` pairs into vectors

Domain: Natural Numbers

- fillers := {0, 1, … 9}, roles := {10^0, 10^1, 10^2, …}
- 376
  ↔ {(3, 10^2), (7, 10^1), (6, 10^0)}

Primer: Tensor Product Representations

1. Input: `filler x role` pairs



Figure: Tom McCoy, TPR Demo (https://rtmccoy.com/tpdn/tpr_demo.html)

Primer: Tensor Product Representations

$f_i \quad r_i$

1. Input: `filler x role` pairs
2. Embedding lookup

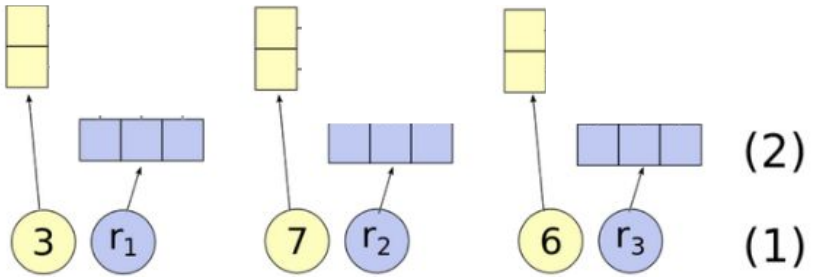


Figure: Tom McCoy, TPR Demo (https://rtmccoy.com/tpdn/tpr_demo.html)

Primer: Tensor Product Representations

$$f_i \otimes r_i$$

1. Input: `filler x role` pairs
2. Embedding lookup
3. Tensor product binding

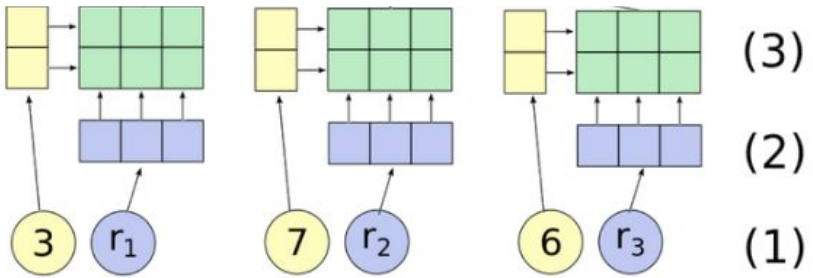


Figure: Tom McCoy, TPR Demo (https://rtmccoy.com/tpdn/tpr_demo.html)

Primer: Tensor Product Representations

$$\sum_{i=1}^{n} f_i \otimes r_i$$

1. Input: `filler x role` pairs
2. Embedding lookup
3. Tensor product binding
4. Sum

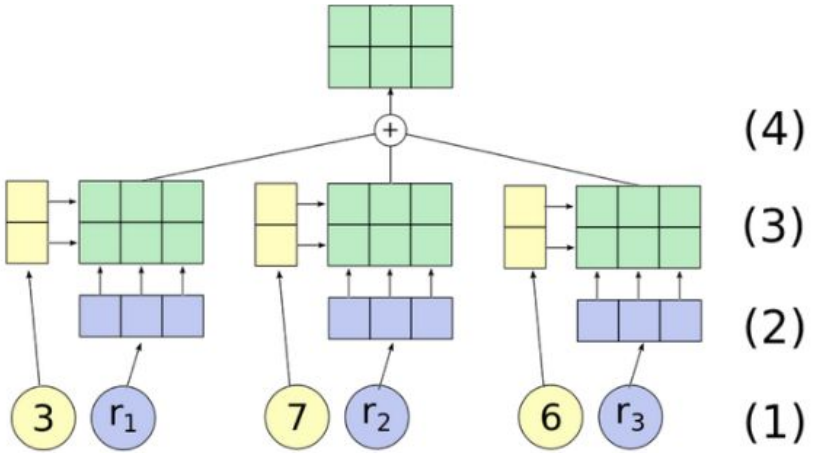


Figure: Tom McCoy, TPR Demo (https://rtmccoy.com/tpdn/tpr_demo.html)

Primer: Tensor Product Representations

$$\text{vec}\left(\sum_{i=1}^{n} f_i \otimes r_i\right)$$

1. Input: `filler x role` pairs
2. Embedding lookup
3. Tensor product binding
4. Sum
5. Flatten

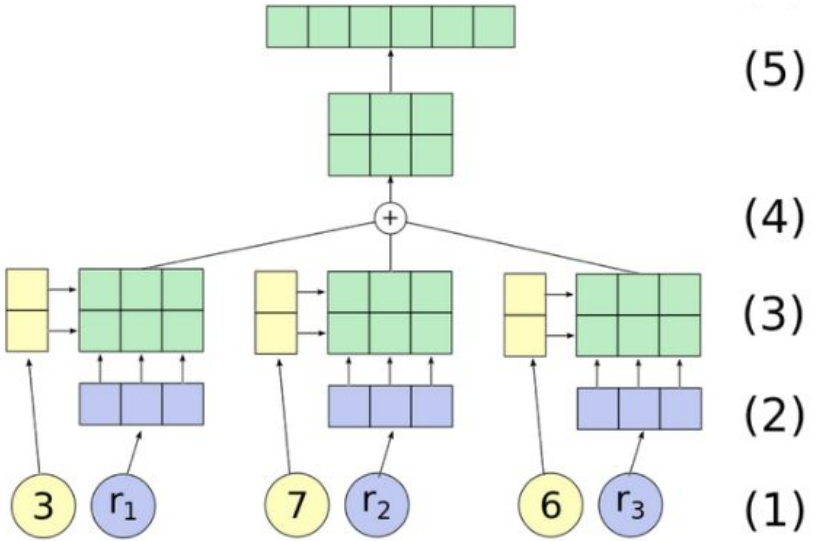


Figure: Tom McCoy, TPR Demo (https://rtmccoy.com/tpdn/tpr_demo.html)

Primer: Tensor Product Representations

$$h = W \cdot \mathrm{vec}\left(\sum_{i=1}^{n} f_i \otimes r_i\right) + b$$

1. Input: `filler x role` pairs
2. Embedding lookup
3. Tensor product binding
4. Sum
5. Flatten
6. Linear map to output

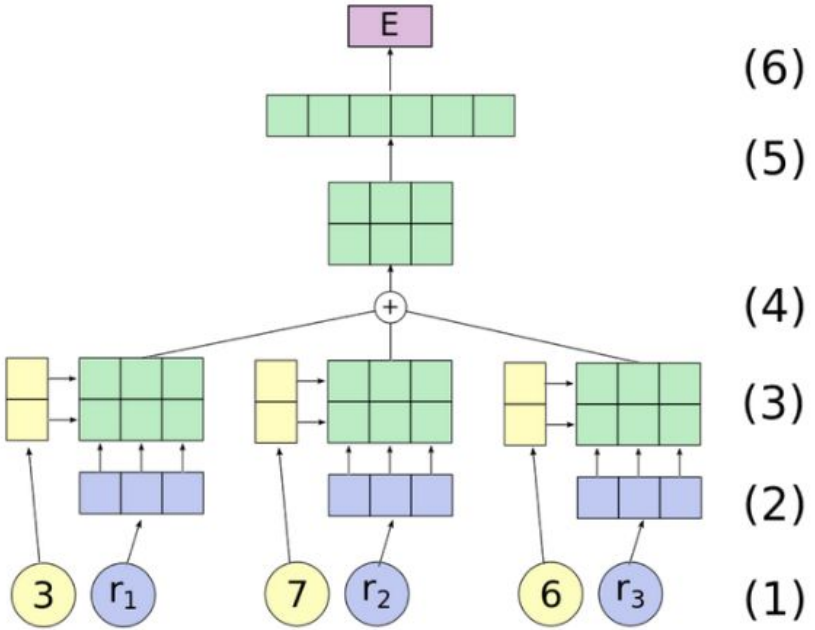


Figure: Tom McCoy, TPR Demo (https://rtmccoy.com/tpdn/tpr_demo.html)

Primer: Tensor Product Representations

Why TPRs?

- Encodes *symbolic structure* (as filler-role pairs) into vector representations losslessly
  - **Filler-role pairs** are expressive
  - **Vector representations** are neural network-friendly
- **Lossless?**
  - Yes!
  - We can find unbinding vectors that *recover* the filler bound to a role from a TPR. via the *unbinding operator u*,

$$r_i^\top u_j = \delta_{ij}$$

Primer: Tensor Product Encoders

Tensor Product Encoders can be trained to approximate Neural Networks

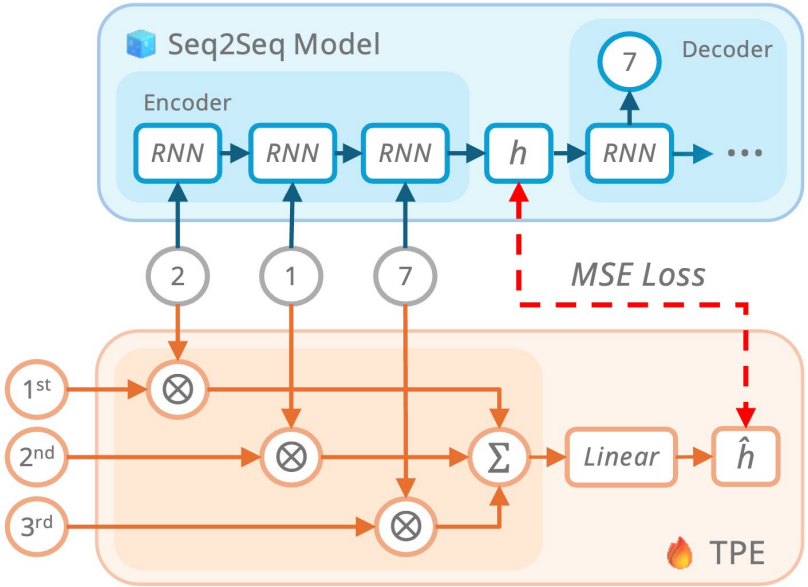
```
Loss := mse_loss(tpe(input_tpe), nn(input_nn))
```

*works for any *fixed size* representation.

Empirically, this often works!

Primer: Tensor Product Encoders

Example: TPR approximation of Seq2seq model trained on sequences

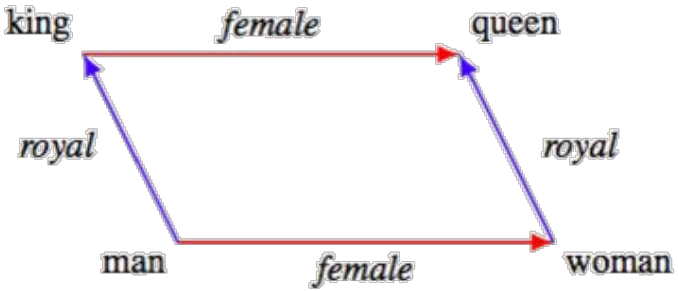


Roadmap

- Background — Interpreting NNs
  - Interpretability now has a wealth of techniques, is there a unifying picture?
- Tensor Product Representations
  - Tensor product representations are a (near) lossless encoding of filler-role representations in vector space
- ***TPRs as a unifying architecture***
  - TPRs → Additive Analogies
  - TPRs → Linear Probes
  - TPRs → SAEs
  - TPRs → Activation Patching

Experiment 1: Additive Analogies

Mikolov et. al (2013)



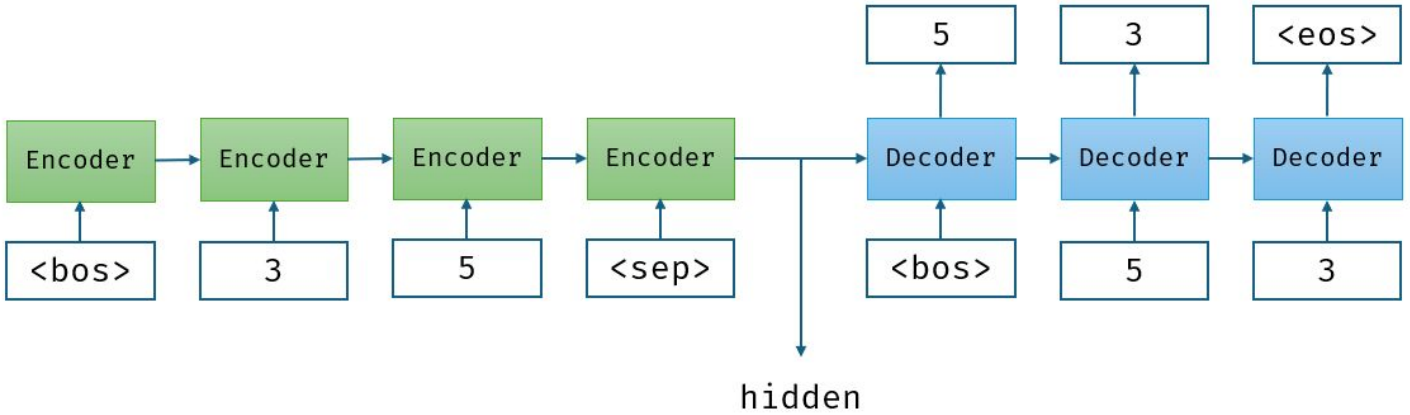
(Queen - King) ≈ (Woman - Man)

Queen - (Woman - Man) ≈ King*

*with important caveats, & distinctions from our setup

Experiment 1.1: Additive Analogies

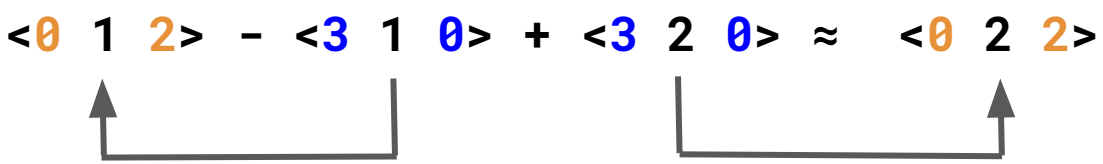
- Copy: [1, 2, 3, 5, 2, 7] → [1, 2, 3, 5, 2, 7]
- Reverse: [1, 2, 3, 5, 2, 7] → [7, 2, 5, 3, 2, 1]



Train recurrent models on sequences of vocab 20, length 2-6

Experiment 1.1: Additive Analogies

Construct analogy quartets:



Evaluate: if nearest neighbour of resultant embedding is target

Result: additive analogies hold with perfect accuracy!

Experiment 1.1: Additive Analogies

Can TPR explain this?

$$\langle 0\ 1\ 2\rangle - \langle 3\ 1\ 0\rangle + \langle 3\ 2\ 0\rangle \approx \langle 0\ 2\ 2\rangle$$

Experiment 1.1: Additive Analogies

Can TPR explain this?

<0 1 2> - <3 1 0> + <3 2 0> ≈ <0 2 2>

<0 1 2> = <0, 1st> + <1, 2nd> + <2, 3rd>
<3 1 0> = <3, 1st> + <1, 2nd> + <0, 3rd>
<3 2 0> = <3, 1st> + <2, 2nd> + <0, 3rd>

<0 1 2> - <3 1 0> + <3 2 0>
= <0 1 2> - (<3 1 0> - <3 2 0>)

Experiment 1.1: Additive Analogies

Can TPR explain this?

<0 1 2> - <3 1 0> + <3 2 0> ≈ <0 2 2>

```
<0 1 2> = <0, 1st> + <1, 2nd> + <2, 3rd>
<3 1 0> = <3, 1st> + <1, 2nd> + <0, 3rd>
<3 2 0> = <3, 1st> + <2, 2nd> + <0, 3rd>

<0 1 2> - <3 1 0> + <3 2 0>
    = <0 1 2> - (  <3, 1st> + <1, 2nd> + <0, 3rd>
                 - ( <3, 1st> + <2, 2nd> + <0, 3rd>))
```

Experiment 1.1: Additive Analogies

Can TPR explain this?

<0 1 2> - <3 1 0> + <3 2 0> ≈ <0 2 2>

<0 1 2> = <0, 1st> + <1, 2nd> + <2, 3rd>
<3 1 0> = <3, 1st> + <1, 2nd> + <0, 3rd>
<3 2 0> = <3, 1st> + <2, 2nd> + <0, 3rd>

<0 1 2> - <3 1 0> + <3 2 0>
= <0 1 2> - (~~<3, 1st>~~ + <1, 2nd> + ~~<0, 3rd>~~
- (~~<3, 1st>~~ + <2, 2nd> + ~~<0, 3rd>~~))

Experiment 1.1: Additive Analogies

Can TPR explain this?

<0 1 2> - <3 1 0> + <3 2 0> ≈ <0 2 2>

```
<0 1 2> = <0, 1st> + <1, 2nd> + <2, 3rd>
<3 1 0> = <3, 1st> + <1, 2nd> + <0, 3rd>
<3 2 0> = <3, 1st> + <2, 2nd> + <0, 3rd>

<0 1 2> - <3 1 0> + <3 2 0>
    = <0 1 2> - (<1, 2nd> - <2, 2nd>)
    = <0, 1st> + <1, 2nd> + <2, 3rd> - (<1, 2nd> - <2, 2nd>)
```

Experiment 1.1: Additive Analogies

Can TPR explain this?

<0 1 2> - <3 1 0> + <3 2 0> ≈ <0 2 2>

<0 1 2> = <0, 1st> + <1, 2nd> + <2, 3rd>
<3 1 0> = <3, 1st> + <1, 2nd> + <0, 3rd>
<3 2 0> = <3, 1st> + <2, 2nd> + <0, 3rd>

<0 1 2> - <3 1 0> + <3 2 0>
= <0 1 2> - (<1, 2nd> - <2, 2nd>)
= <0, 1st> + ~~<1, 2nd>~~ + <2, 3rd> - (~~<1, 2nd>~~ - <2, 2nd>)

Experiment 1.1: Additive Analogies

Can TPR explain this?

<0 1 2> - <3 1 0> + <3 2 0> ≈ <0 2 2>

```
<0 1 2> = <0, 1st> + <1, 2nd> + <2, 3rd>
<3 1 0> = <3, 1st> + <1, 2nd> + <0, 3rd>
<3 2 0> = <3, 1st> + <2, 2nd> + <0, 3rd>

<0 1 2> - <3 1 0> + <3 2 0>
    = <0 1 2> - (<1, 2nd> - <2, 2nd>)
    = <0, 1st> + <2, 3rd> + <2, 2nd>
    = <0 2 2>
```

Experiment 1.1: Additive Analogies

Generalize:

Analogy works because it *substitutes filler–role pairs*

Experiment 1.1: Additive Analogies

Can TPR explain this?

Yes, TPR $\vdash$ Additive Analogies*

*mathematically

Experiment 1.1: Additive Analogies

How can we test this?

If TPR holds, we can also do additive analogies easier!

Instead of...

$$\langle 0\ 1\ 2\rangle - \langle 3\ 1\ 0\rangle + \langle 3\ 2\ 0\rangle \approx \langle 0\ 2\ 2\rangle$$

Embeddings of NN

We can use...

$$\langle 0\ 1\ 2\rangle - \langle 1,\ 2nd\rangle + \langle 2,\ 2nd\rangle \approx \langle 0\ 2\ 2\rangle$$

Bindings of TPE

Experiment 1.1: Additive Analogies

... and they work equally well!

Task	Architecture	Vector Analogy	TPE Analogy
Copy	RNN	100%	100%
	GRU	100%	100%
	LSTM	100%	100%
Reverse	RNN	95.4%	98.4%
	GRU	100%	100%
	LSTM	100%	100%

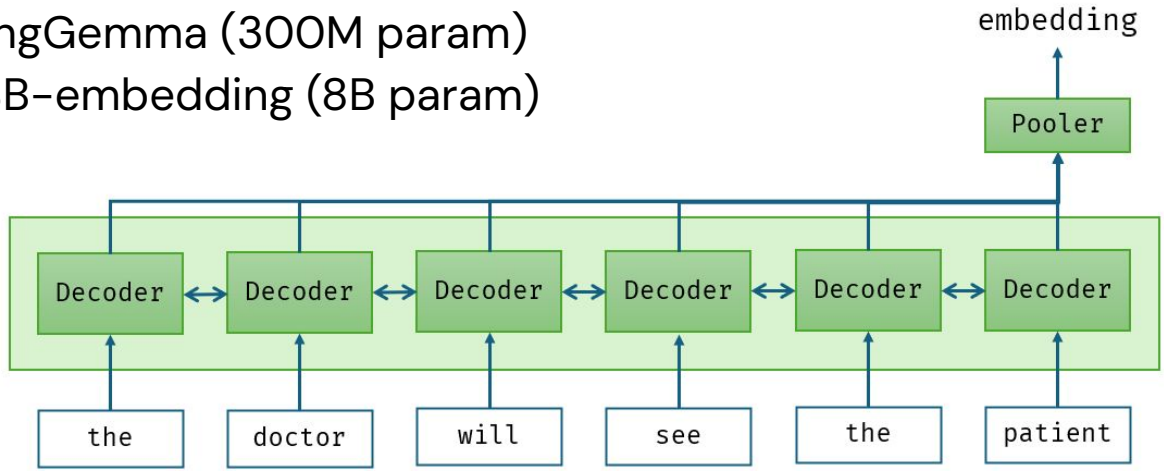
But why should we trust your toy task?

Experiment 1.2: Additive Analogies

Data: SVO sentences. 77 Occupations, “the <subj> will see the <obj>”

Model: SOTA embedding transformers:

- ModernBERT (100M param)
- EmbeddingGemma (300M param)
- Qwen3-8B-embedding (8B param)



Experiment 1.2: Additive Analogies

Expect:

```
<the doctor will see the patient>
    - <the linguist will see the patient  >
    + <the linguist will see the astronaut>
    ≈ <the doctor   will see the astronaut>
```

	ModernBERT	EmbedGemma	Qwen3
NN Analogy	92.3%	98.2%	90.8%

Experiment 1.2: Additive Analogies

Instead of...

<doctor see patient> - <linguist see patient> + <linguist see astronaut>
≈ <doctor see astronaut>

We can ...

<doctor see patient> - <patient, obj> + <astronaut, obj>
≈ <doctor see astronaut>

	ModernBERT	EmbedGemma	Qwen3-8B
NN Analogy	92.3%	98.2%	90.8%
TPE Analogy	96.5%	99.4%	94.8%

Experiment 1.2: Additive Analogies

TPR $\vdash$ Additive Analogies*

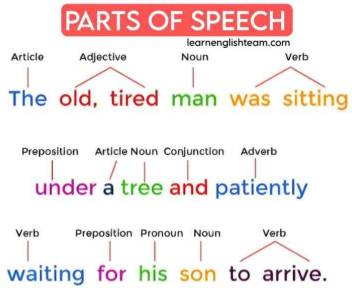
*empirically

Roadmap

- Background — Interpreting NNs
  - Interpretability now has a wealth of techniques, is there a unifying picture?
- Tensor Product Representations
  - Tensor product representations are a (near) lossless encoding of filler-role representations in vector space
- ***TPRs as a unifying architecture***
  - TPRs → Additive Analogies
  - TPRs → Linear Probes
  - TPRs → SAEs
  - TPRs → Activation Patching

Experiment 2: Linear Probing

From the era of BERTology: BERT knows*....



Part of Speech

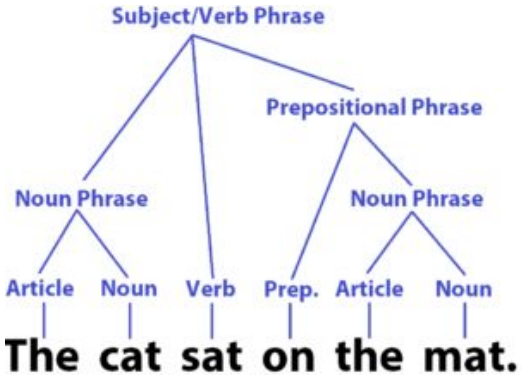
Tenney et al. 2019



*with many caveats, read the fine print (original papers)!

Experiment 2: Linear Probing

From the era of BERTology: BERT knows*....



Syntactic Structure

Hewitt et al. 2019

*with many caveats, read the fine print (original papers)!



Experiment 2: Linear Probing

From the era of BERTology: BERT knows*....

Among many others...



*with many caveats, read the fine print (original papers)!

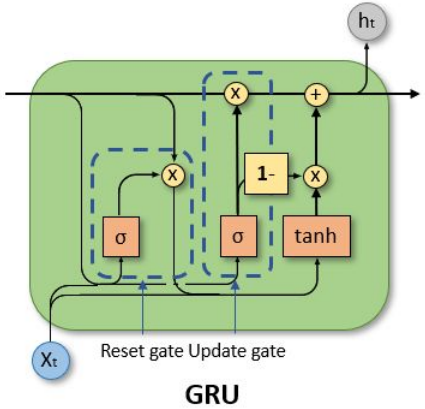
Linear Probing

Probes takes in hidden states (*h*) and linearly maps to features of interest

$$\text{logits} = Wh + b$$

They're often highly successful

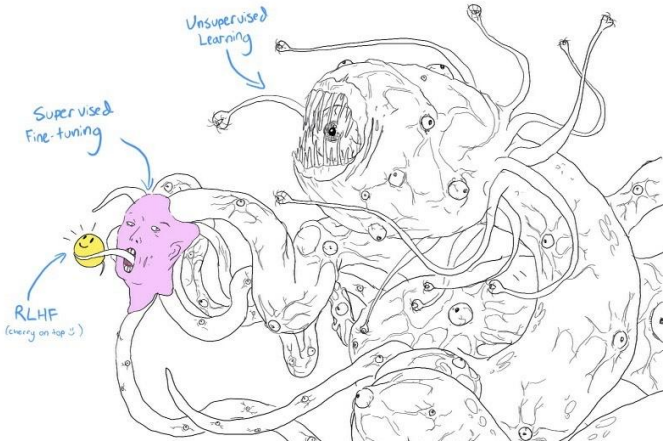
on toy models..



or BERT…



or LLMs…



Experiment 2: Linear Probing

Can TPR

$$h = W \cdot \mathrm{vec}\left(\sum_{i=1}^{n} f_i \otimes r_i \in \mathbb{R}^{d_f \times d_r}\right) + b$$

explain this?

$$\mathrm{logits} = Wh + b$$

Experiment 2: Linear Probing

$$\text{logits} = Wh + b$$

Linear probes are functions from *vector space* to *semantics*

Can we construct, with TPRs, a function that does the same?

Experiment 2: Linear Probing

Claim: Yes! Assume hidden state h is a TPR, i.e.

$$h = W \cdot \mathrm{vec}\left(\sum_{i=1}^{n} f_i \otimes r_i \in \mathbb{R}^{d_f \times d_r}\right) + b$$

there is

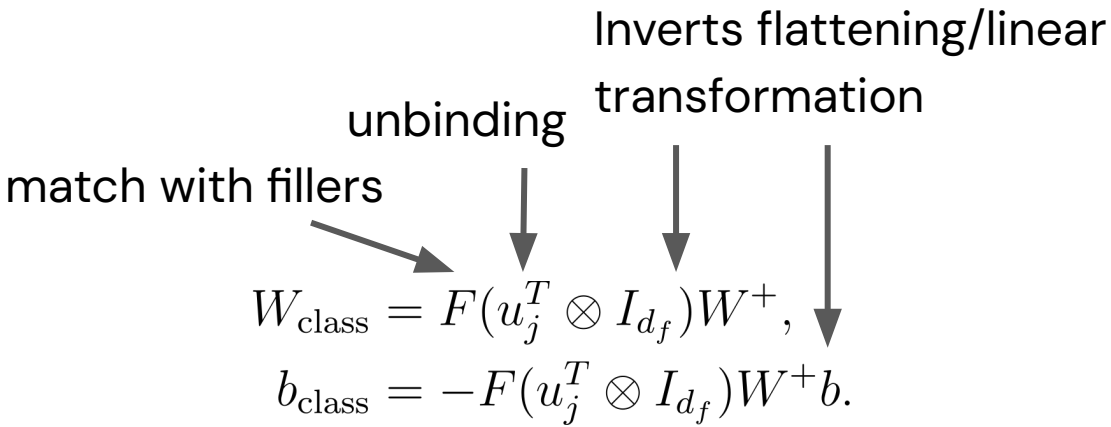
F := filler embeddings transposed
u :=unbinding vectors,
W+ := pseudoinverse of *W*

$$W_{\text{class}} = F(u_j^T \otimes I_{d_f})W^+,$$
$$b_{\text{class}} = -F(u_j^T \otimes I_{d_f})W^+ b.$$

that give logits over the vocabulary of fillers.

Experiment 2: Linear Probing

How?



Since all are linear/affine operations, they combine into a single operation.

Experiment 2: Linear Probing

Can TPR explain this?

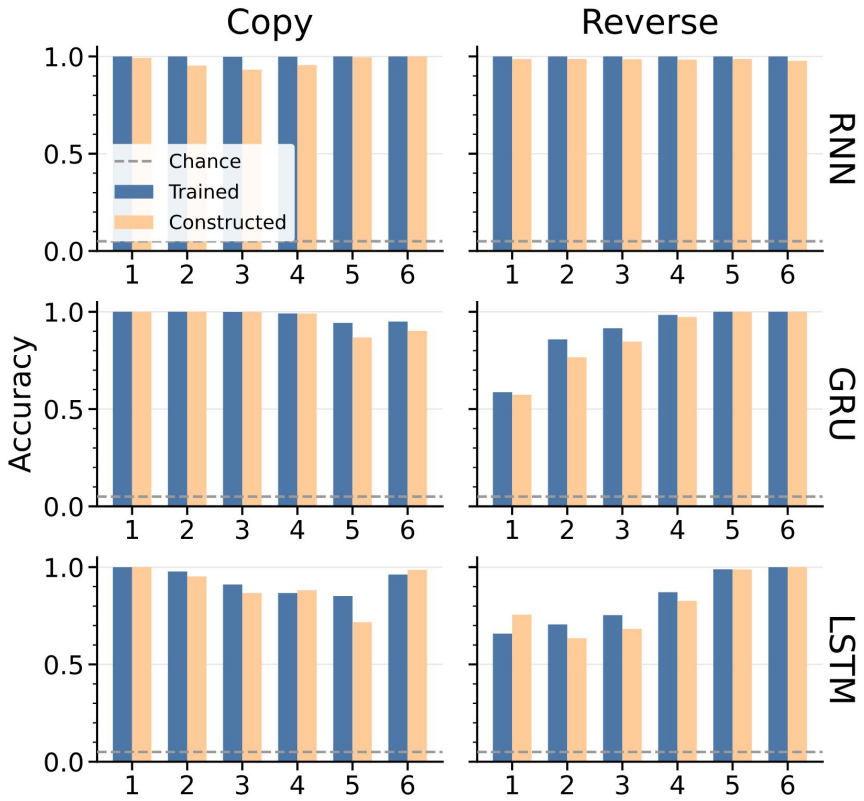
Yes, TPR $\vdash$ Linear Probing*

*mathematically

Experiment 2: Linear Probing

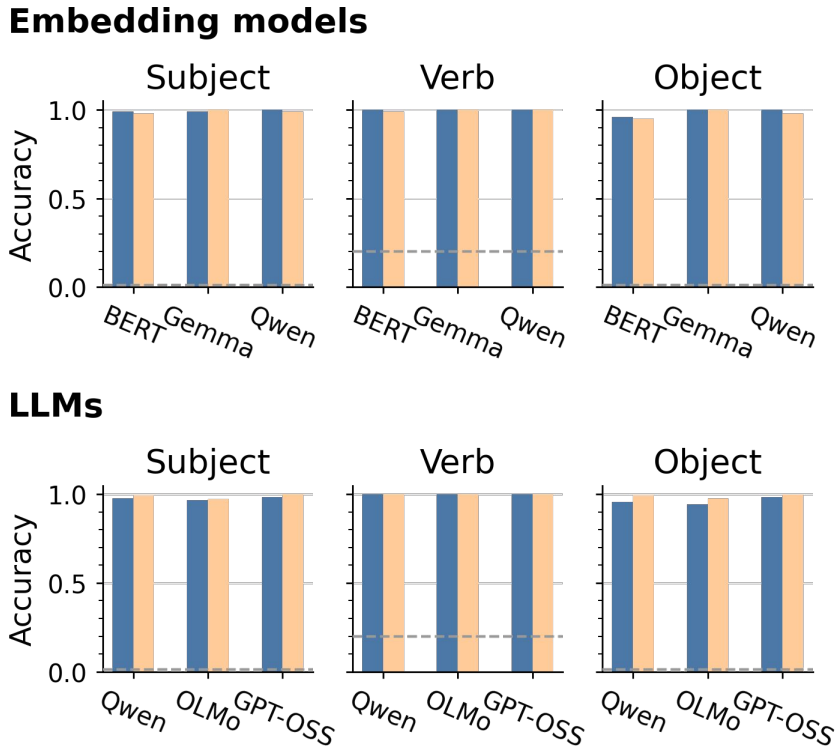
Probes can be constructed from TPEs **without further training.**

Synthetic Sequences:



Experiment 2: Linear Probing

Language Models, SVO sentences:



Experiment 2: Linear Probing

TPR $\vdash$ Linear Probing*

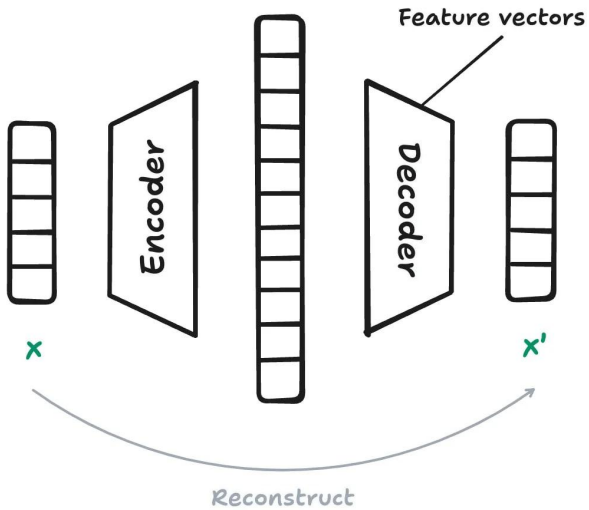
*empirically

Roadmap

- Background — Interpreting NNs
  - Interpretability now has a wealth of techniques, is there a unifying picture?
- Tensor Product Representations
  - Tensor product representations are a (near) lossless encoding of filler-role representations in vector space
- ***TPRs as a unifying architecture***
  - TPRs → Additive Analogies
  - TPRs → Linear Probes
  - TPRs → SAEs
  - TPRs → Activation Patching

Experiment 3: Sparse Autoencoders

SAEs *decompose* an activation into a large set of *sparsely activated features*



Experiment 3: Sparse Autoencoders

Feature #34M/31164353 **Golden Gate Bridge** feature example

The feature activates strongly on English descriptions and associated concepts

in the Presidio at the end (that's the↵ huge park right next to the Golden Gate bridge), perfect. But not all people

repainted, roughly, every dozen years." "while across the country in san fran cisco, the golden gate bridge was

it is a suspension bridge and has similar coloring, it is often↵> compared to the Golden Gate Bridge in San Francisco, US

They also activate in multiple other languages on the same concepts

ゴールデン・ゲート・ブリッジ、金門橋は、ア メリカ西海岸のサンフランシスコ湾と太平洋が 接続するゴールデンゲート海

골든게이트 교 또는 금문교 는 미국 캘리포니아주 골든게이트 해협에 위치한 현수교이다. 골든게이 트 교는 캘리포니아주 샌프란시

мост золотые ворота — висячий мост через пролив золотые ворота. он со единяет город сан-франс

And on relevant images as well



Scaling Monosemanticity, https://transformer-circuits.pub/2024/scaling-monosemanticity/

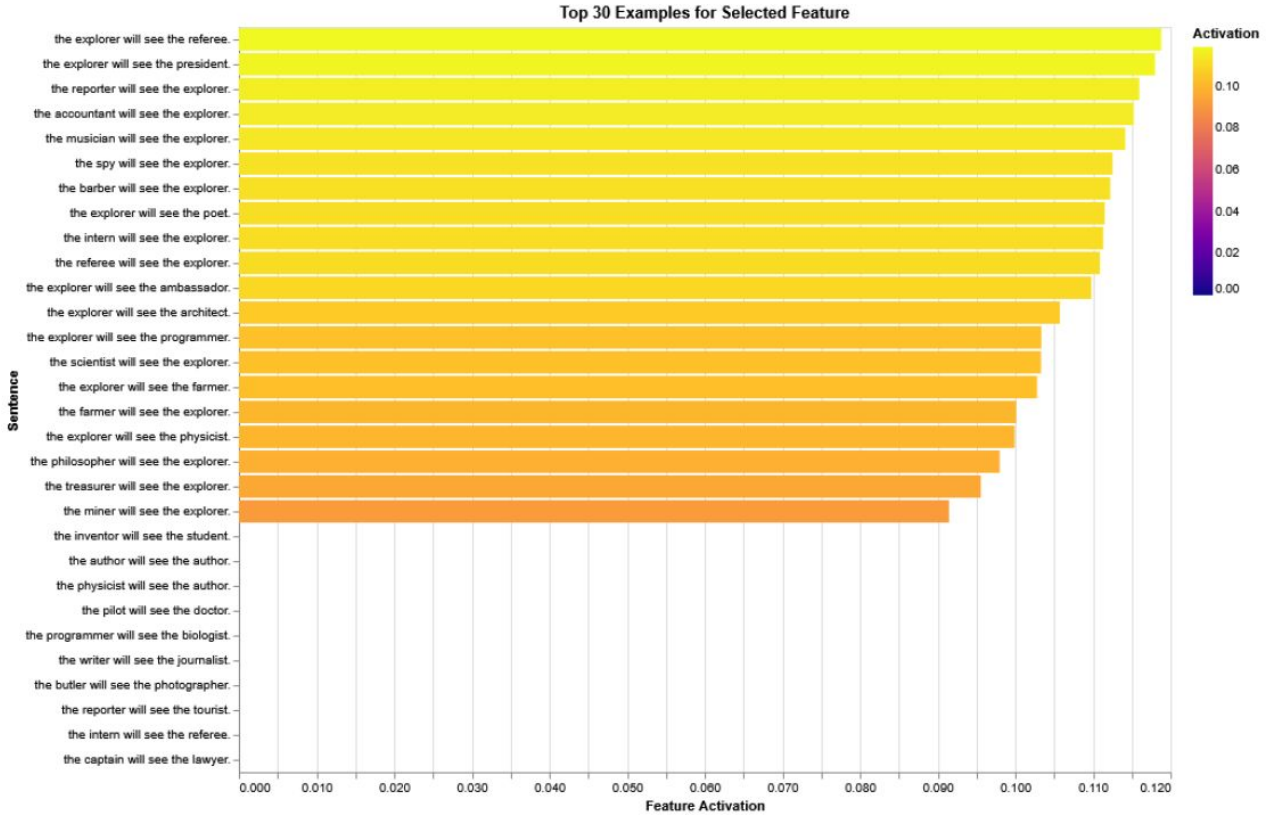
Experiment 3: Sparse Autoencoders

Train...

- Top-K Sparse Autoencoder
- On embeddings of SVO sentences by ModernBERT

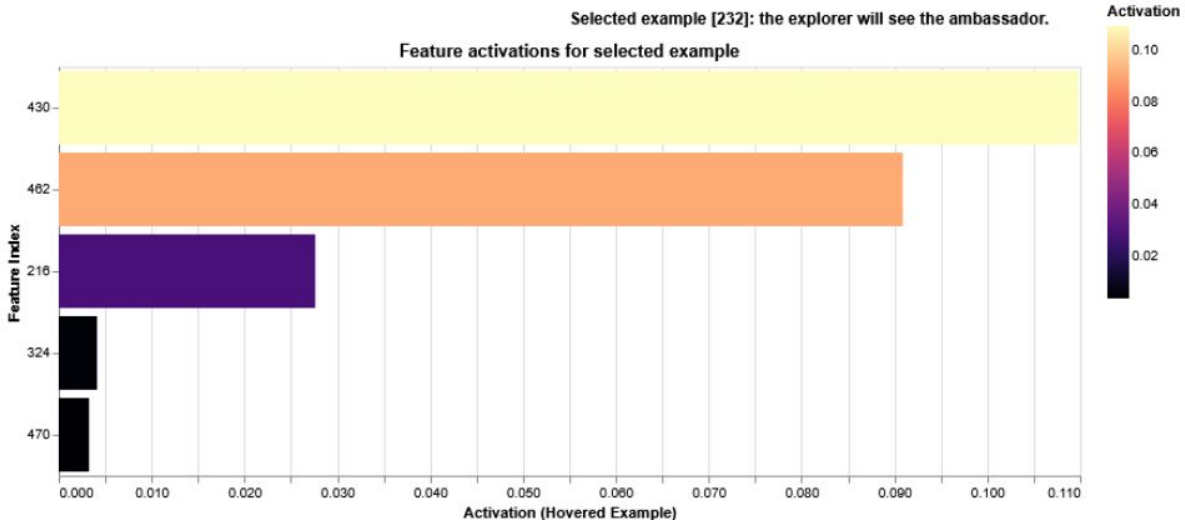
Experiment 3: Sparse Autoencoders

Feature #430:
explorer



Experiment 3: Sparse Autoencoders

Sentence #232: explorer and ambassador



Feature Index 430
Activation 0.1098
Top example 1 the explorer will see the referee.
Top example 2 the explorer will see the president.
Top example 3 the reporter will see the explorer.
Top example 4 the accountant will see the explorer.
Top example 5 the musician will see the explorer.

Feature Index 462
Activation 0.0909
Top example 1 the ambassador will see the actor.
Top example 2 the ambassador will see the detective.
Top example 3 the ambassador will see the teacher.
Top example 4 the ambassador will see the coach.
Top example 5 the ambassador will see the artist.

Experiment 3: Sparse Autoencoders

- SAE reconstructs its input activations
  - We measure with *Reconstruction R-square*
- Features that correspond meaningfully with concepts
  - We measure with *Average Feature Quality*

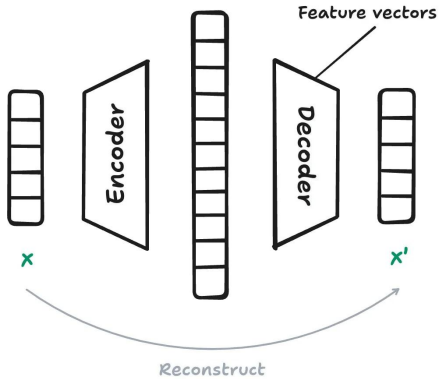
Model	SAE Type	R^2	Quality (%)
ModernBERT	Top-k	0.9803	72.52
EmbedGemma	Top-k	0.9889	92.63
Qwen3-8B	Top-k	0.9785	80.84

Experiment 3: Sparse Autoencoders

- Can TPR

$$h = W \cdot \mathrm{vec}\left(\sum_{i=1}^{n} f_i \otimes r_i \in \mathbb{R}^{d_f \times d_r}\right) + b$$

- explain this?



Experiment 3: Sparse Autoencoders

Yes! Assume hidden state is TPR, i.e.

$$h = W \cdot \mathrm{vec}\left(\sum_{i=1}^{n} f_i \otimes r_i \in \mathbb{R}^{d_f \times d_r}\right) + b$$

There are SAE feature vectors *d* with bias *c*

$$\textbf{Filler} \times \textbf{Role:} \quad d = (W^{+})^{\top}(u_k \otimes f_\star),\ c = -d^{\top} b.$$

The intuition & construction here is similar to the linear probing case.

Experiment 3: Sparse Autoencoders

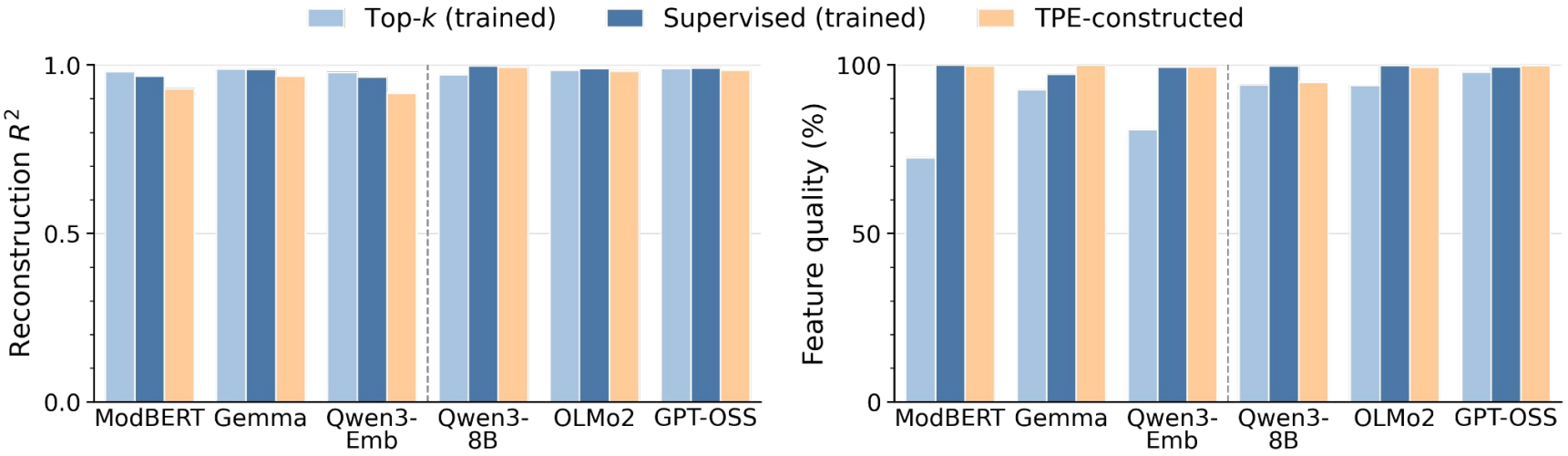
TPR $\vdash$ Sparse Autoencoders*

*mathematically

Experiment 3: Sparse Autoencoders

The derivation allows us to *construct* SAEs.

Constructed SAEs are comparable to trained ones:



Experiment 3: Sparse Autoencoders

TPR ⊢ Sparse Autoencoders*

*empirically

Roadmap

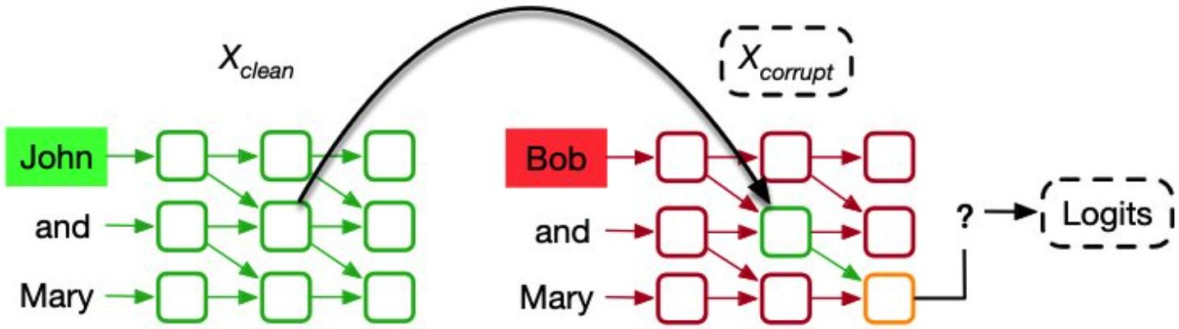
- Background — Interpreting NNs
  - Interpretability now has a wealth of techniques, is there a unifying picture?
- Tensor Product Representations
  - Tensor product representations are a (near) lossless encoding of filler-role representations in vector space
- ***TPRs as a unifying architecture***
  - TPRs → Additive Analogies
  - TPRs → Linear Probes
  - TPRs → SAEs
  - TPRs → Activation Patching

Experiment 4: Activation Patching

Study how activations affect behavior:

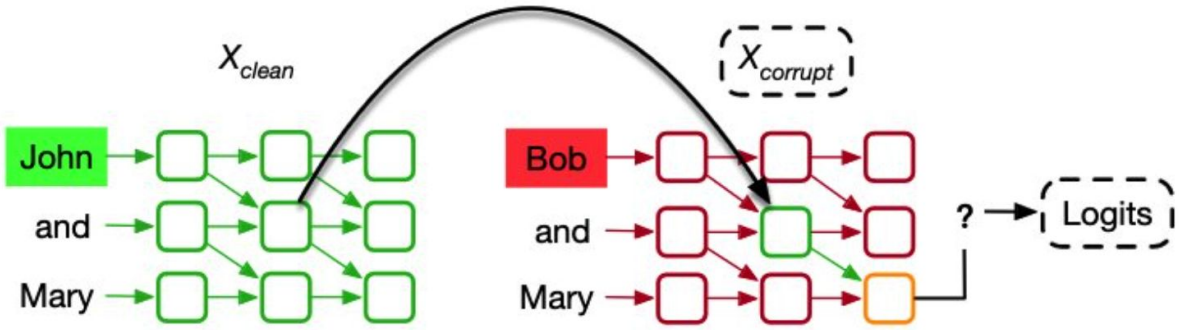
Source: `The linguist will see the student. The student will be seen by the ____`

Dest: `The actor will see the student. The student will be seen by the ____`

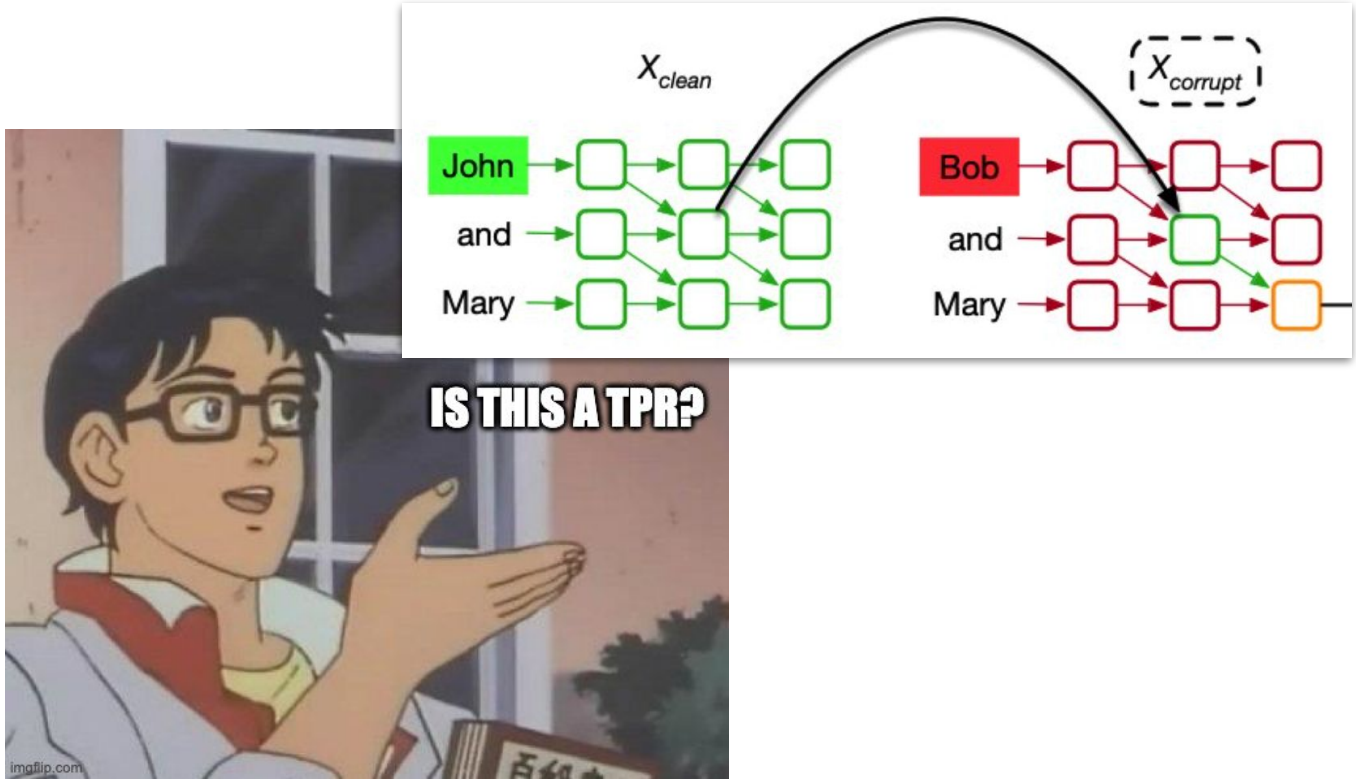


Experiment 4: Activation Patching

Observation: for this to work, representations must be **structured.**



Experiment 4: Activation Patching



Experiment 4: Activation Patching

If so:

<...linguist see student...> = <linguist, subject> + <see, verb> + <student, object>

<...actor see student...> = <actor, subject> + <see, verb> + <student, object>

Observation:

- Also predicts that difference between the patched activations is a *filler-role binding difference for the subject role*.

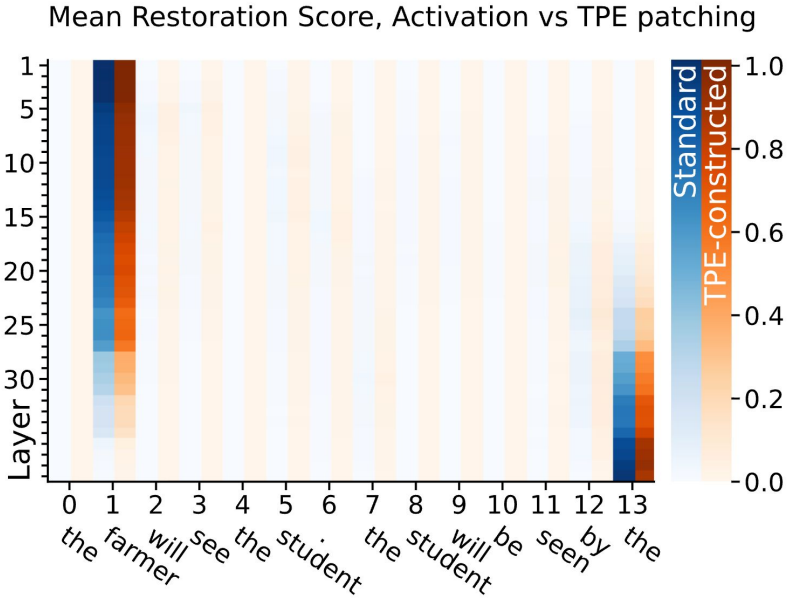
<...actor...> = <...linguist...> − <linguist, subject> + <actor, subject>

Experiment 4: Activation Patching

1. Perform activation patching
2. Train TPE to approximate LM activations at each layer
3. Perform TPE-based patching
    a. Instead of replacing activations, subtract by difference:
    <linguist, subject> = <actor, subject>

Experiment 4: Activation Patching

Achieve identical effects!



Experiment 4: Activation Patching

TPR ⊢ Activation Patching*

*mathematically, as well as empirically

Roadmap

- Background — Interpreting NNs
  - Interpretability now has a wealth of techniques, is there a unifying picture?
- Tensor Product Representations
  - Tensor product representations are a (near) lossless encoding of filler-role representations in vector space
- TPRs as a unifying architecture
  - Tensor product encoders mathematically and empirically explain the success of several approaches to model interpretation

Discussion

Big picture: rethinking mech interp as

Structure / Content

Discussion

Towards more generalized interpretation architectures:

- TPR → Interpretation techniques
- Structural hypothesis → downstream predictions as a productive interpretation pipeline that derives multiple uses from one training

Discussion

Structural hypothesis comparison:

TPR

/

Linear Representation Hypothesis

/* Overflow Slides */

Unbinding a Tensor Product Representation

We can find unbinding vectors, such that

- When multiplied with the correct role vector, it gives 1
- When multiplied with any other role vector, it gives 0

i.e.

$$r_i^\top u_j = \delta_{ij}$$

This exactly holds under a full-rank hypothesis ($d_{\text{role}} \geq |\{\text{roles}\}|$), and approximately holds with benign scaling behavior
(for allowable error, $|\{\text{roles}\}| \propto \varepsilon e^{d_{\text{role}}}$)

Unbinding a Tensor Product Representation

We can extract the filler vector bound to any role!

Given TPR $\sum_{i=1}^{n} f_i \otimes r_i$, unbinding identity $r_i^\top u_j = \delta_{ij}$

Notice that...

$$\sum_{i=1}^{n} f_i \otimes r_i u_j$$

$$= \sum_{i=1}^{n} f_i r_i^\top u_j \quad \text{(Tensor product)}$$

$$= \sum_{i=1}^{n} f_i \delta_{ij} \quad \text{(Unbinding identity)}$$

$$= f_j \quad \text{(Property of Kronecker Delta)}$$

Experiment 2: Linear Probing

Let's show the claim*!

$$W_{\text{class}}h + b_{\text{class}} = \underbrace{F\,(u_j^\top \otimes I_{d_f})\,W^+}_{W_{\text{class}}} \underbrace{\Big(W\,\text{vec}\Big(\sum_i f_i \otimes r_i\Big) + b\Big)}_{h} + \underbrace{\big(-F\,(u_j^\top \otimes I_{d_f})\,W^+ b\big)}_{b_{\text{class}}}$$

$$= F\,(u_j^\top \otimes I_{d_f})\,W^+\Big(W\,\text{vec}\Big(\sum_i f_i \otimes r_i\Big) + b\Big) \;-\; F\,(u_j^\top \otimes I_{d_f})\,W^+ b$$

$$= F\,(u_j^\top \otimes I_{d_f})\;\underbrace{W^+W}_{W^+W \approx I}\;\text{vec}\Big(\sum_i f_i \otimes r_i\Big) \qquad \text{(bias terms cancel)}$$

*I'm equally happy if you trust me on faith

Experiment 2: Linear Probing

contd...

$$F\,(u_j^\top \otimes I_{d_f})\,\mathrm{vec}\Big(\sum_i f_i \otimes r_i\Big)$$

$$= F\left(\sum_i f_i r_i^\top\right) u_j \quad \left(\text{Kronecker/vec identity } (x^\top \otimes I_m)\,\mathrm{vec}(M) = Mx;\ \text{substitute } M = \sum_i f_i r_i^\top,\ x = u_j\right)$$

$$= F \sum_i f_i \big(r_i^\top u_j\big) \qquad (\text{TPR unbinding } r_i^\top u_j = \delta_{ij})$$

$$= F\, f_j.$$

We get dot product between all fillers and recovered filler, which are logits!