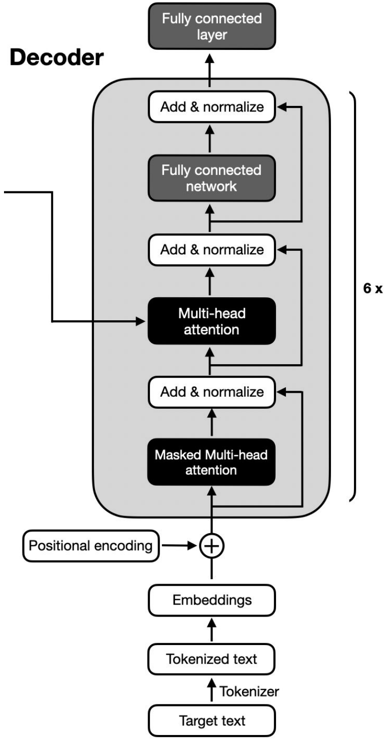
LLM interpretability via sparse autoencoders

INFO 4940: Lecture 10

Why does interpretability matter?

- Can you trust your model?
- Can you trust your data?
- Is the model biased or otherwise harmful?
- Are the features important?
- Are the features useful?
- Training and evaluation costs are high
- Deployment stakes can be high



Approaches to LLM interpretability

Weights are hopeless, but *neurons* might be useful.

There are a lot of neurons, but far fewer than there are total params. Each neuron has many weights and neurons are organized into layers.

Give input, see which neurons activate at a given layer. Hope that there is one ("monosemantic") neuron that fires. Maybe look for "circuits" of neurons that fire across layers.

Works, sometimes, but polysemanticity and superposition …

Superposition

Superposition

Ideally, want an interpretable neuron to activate iff a single "feature" is present in the input

Aside: what's a feature? Coherent concept *in the data*, nothing to do with the model

Some neurons activate in the presence of multiple, *disparate* features (e.g., cats, hex numbers, gerunds). And some features activate many neurons, but only weakly. But not always …

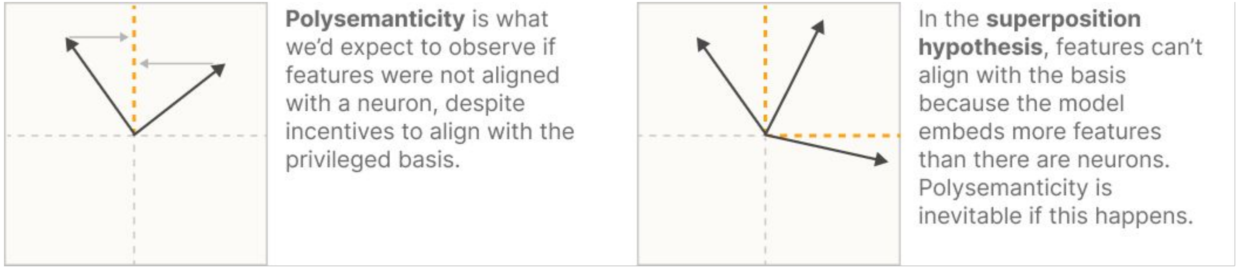
Elhage et al., "Toy Models of Superposition" (2022)

Superposition

Superposition means that features are represented via combinations of neuron activations. The activating neurons *do not* necessarily have anything to do with the feature in question.

Superposition is *required* to represent more features than there are neurons in the model (e.g., all people, ideas, etc.)

Some features are more important than others



Example of superposition

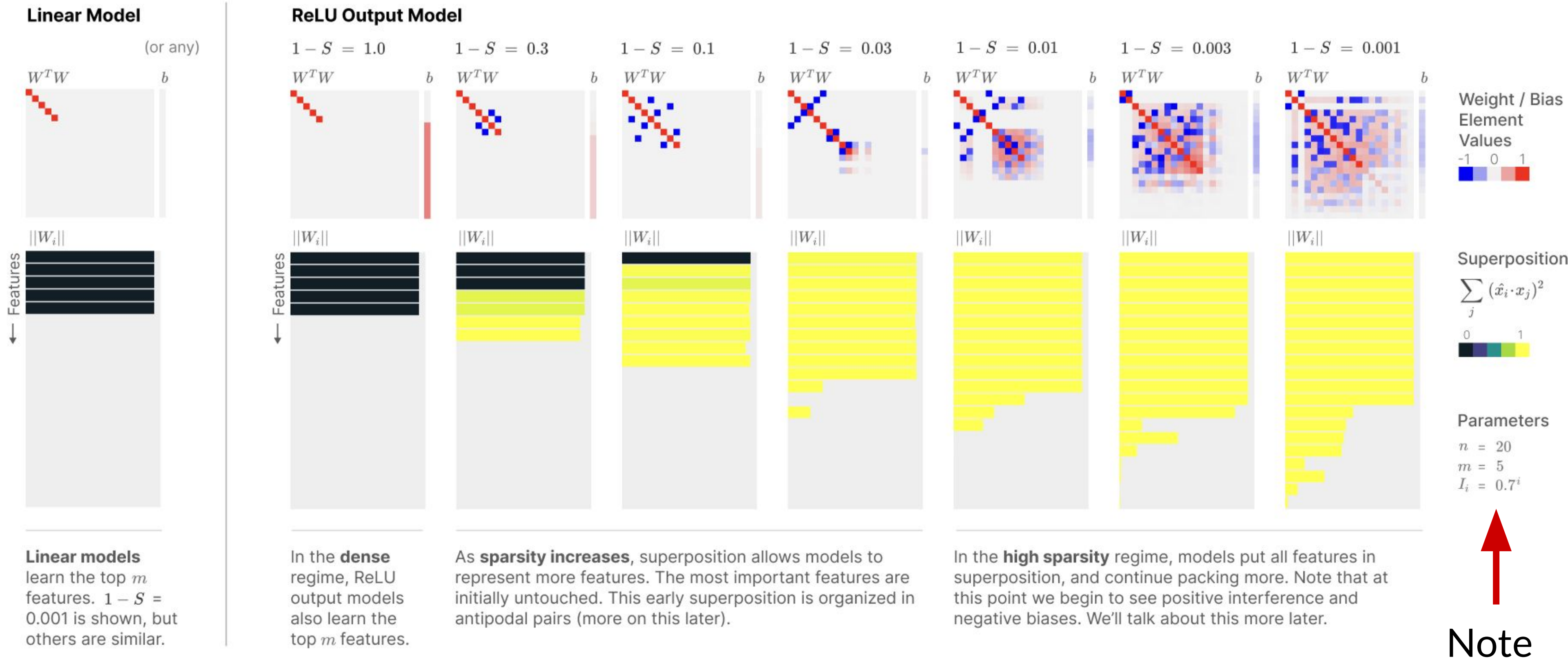
In a small language model ([“Toward monosemanticity”]), there are features for Arabic script and for Hebrew-language text. But no single neuron activates strongly on all Arabic or Hebrew text input. In fact, the top-activating examples for the neurons that *do* fire on such input contain no examples of Arabic or Hebrew inputs.

Clearly, these features are represented only via superposition. They exist across low levels of activation for many neurons.

Superposition

Superposition only works well if features are *sparse*, i.e., if most inputs have few features, and if the network has a nonlinear activation function.

If these conditions are not met, superposition causes destructive interference, because you can't tell whether the multiple activations represent a single, superposed feature or a mix of multiple features.



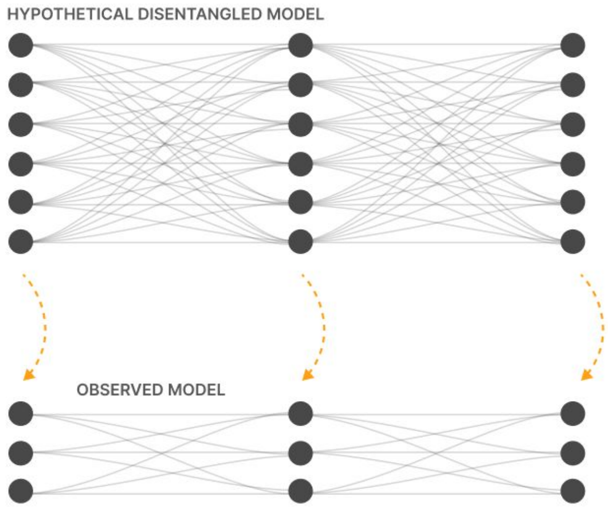
Linear Model
ReLU Output Model
(or any)
$1-S = 1.0$
$1-S = 0.3$
$1-S = 0.1$
$1-S = 0.03$
$1-S = 0.01$
$1-S = 0.003$
$1-S = 0.001$
W^TW
b
$||W_i||$
Features
Weight / Bias Element Values
-1 0 1
Superposition
$\sum_j (\hat{x}_i \cdot x_j)^2$
0 1
Parameters
$n = 20$
$m = 5$
$I_i = 0.7^i$
Linear models learn the top m features. $1-S$ = 0.001 is shown, but others are similar.
In the **dense** regime, ReLU output models also learn the top m features.
As **sparsity increases**, superposition allows models to represent more features. The most important features are initially untouched. This early superposition is organized in antipodal pairs (more on this later).
In the **high sparsity** regime, models put all features in superposition, and continue packing more. Note that at this point we begin to see positive interference and negative biases. We'll talk about this more later.

Note this

Hypothetical networks

Sparse networks with superposition can allow a network to function as a simulation of a larger network that is fully *disentangled*

LLMs may be such networks *and* we can use the same hypothesis to *probe* LLMs



Under the superposition hypothesis, the neural networks we observe are **simulations of larger networks** where every neuron is a disentangled feature.

These idealized neurons are **projected** on to the actual network as “almost orthogonal” vectors over the neurons.

The network we observe is a **low-dimensional projection** of the larger network. From the perspective of individual neurons, this presents as polysemanticity.

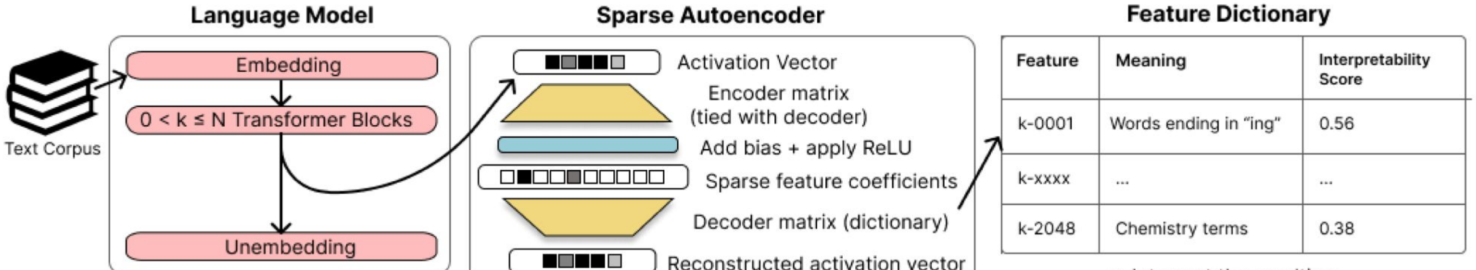
Sparse autoencoders

SAE architecture

Begin with activations from a layer of neurons for many different inputs

Seek to learn a *sparse* representation in *higher*-dimensional space that reconstructs the inputs with minimal loss

Need to penalize non-sparsity, or we just get pseudo-identity



a. Sample activations from a language model

b. Learn a feature dictionary using an autoencoder that learns to represent activation vectors as a sparse linear combination of feature vectors.

c. Interpret the resulting dictionary features

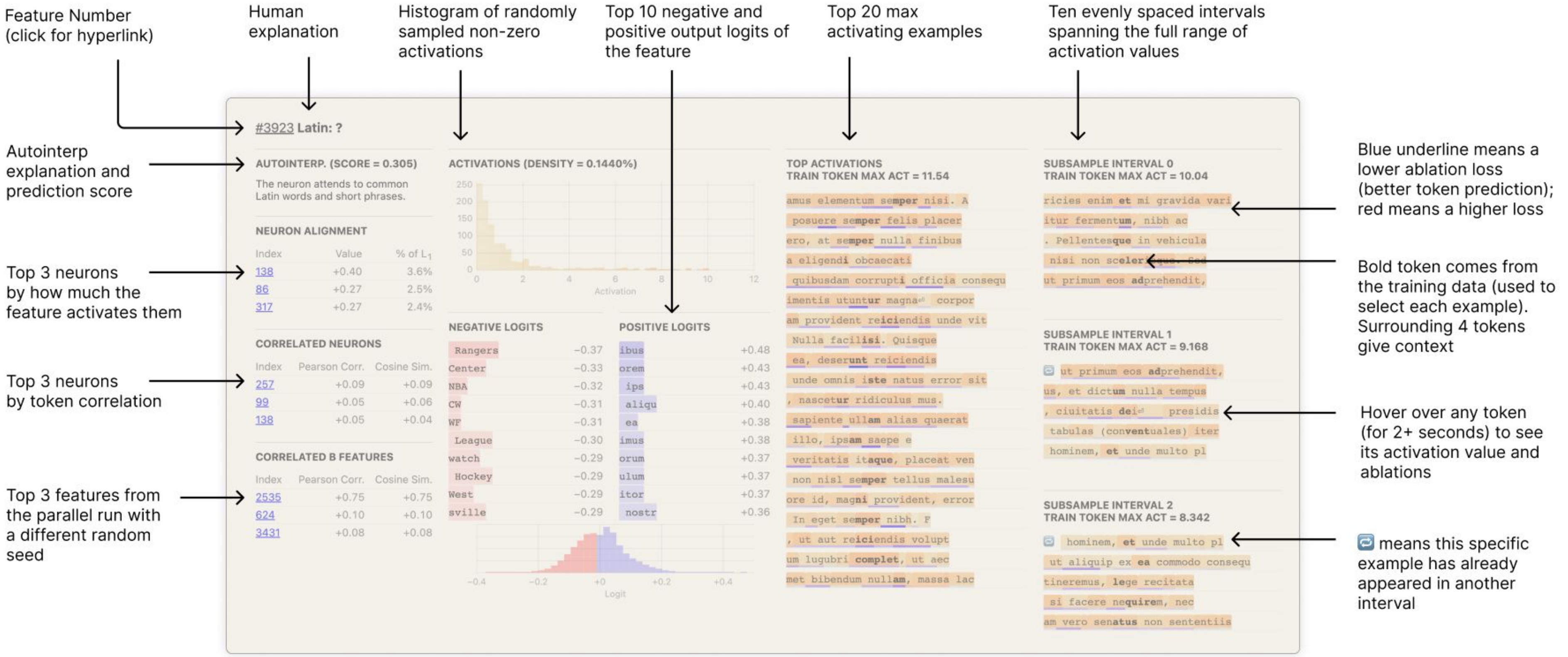
SAE interpretation

Hidden layer in the SAE is the hypothesized large feature set. In *this space*, 1 neuron = 1 feature.

For a given input, the feature set is manageable for interp, because most features/activations are zero.

Can find inputs that maximally activate each neuron/feature in the SAE.

Can change values of SAE neurons and observe effect on output logits (i.e., next token predictions).



https://transformer-circuits.pub/2023/monosemantic-features/vis/a1.html

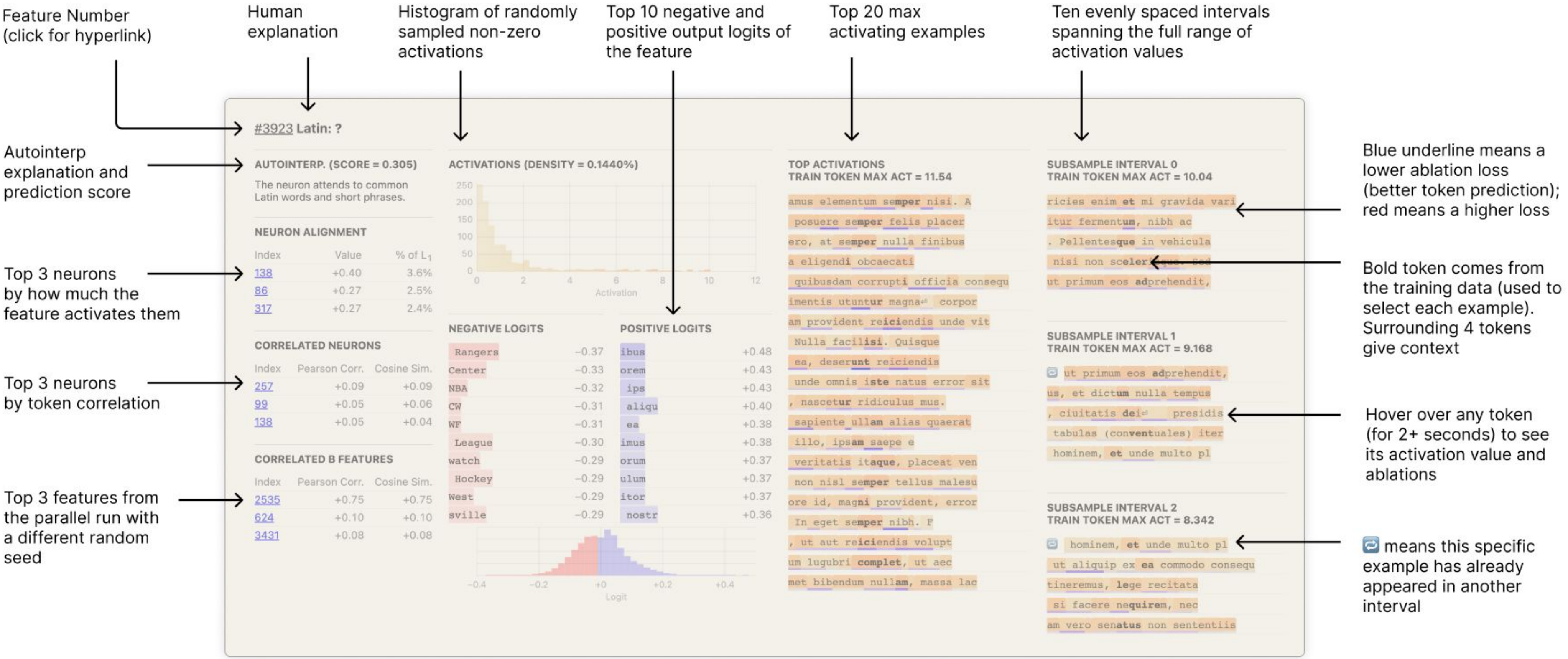
Evaluating SAE features

Are the features that we find meaningful and monosemantic?

Are similar features found across multiple models and SAE settings?

Do features split as the size of the hidden layer in the SAE increases and coalesce as it decreases?

Can SAE features be intervened on in predictable ways?



https://transformer-circuits.pub/2023/monosemantic-features/vis/a1.html

Steering

Using SAE features to change output

- An interactive demo: Gemma Scope
- A code tutorial from SAE Lens (scroll down to “feature steering” and “feature ablations”)