Python for LLM Research

The NNSight Python Library for Mechanistic Interpretability

- bolu ben-adeola

Intro To Mechanistic Interpretability

1 Neural networks solve an increasing number of important tasks really well.

Intro To Mechanistic Interpretability

2 It would be at least ***interesting***, and probably ***important*** to understand how.

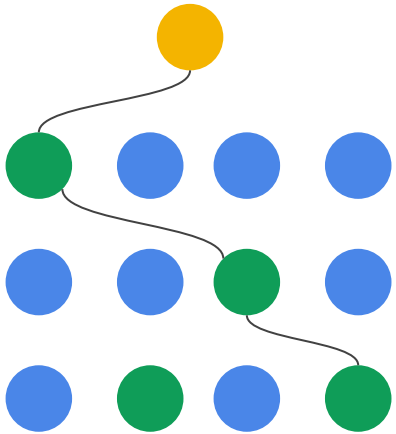
Intro To Mechanistic Interpretability

3 Mechanistic Interpretability (Mech Interp) tackles this problem.

Mech Interp

Mechanistic Model:

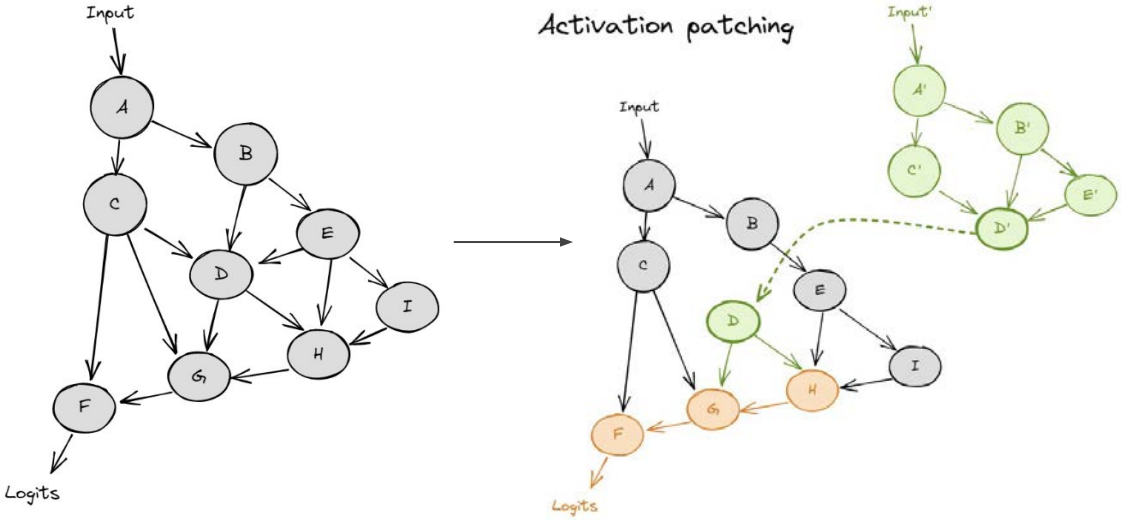
A granular causal model of how components in a network contribute to observed phenomena.



Mech Interp Toolkit: Causal Interventions

Measure the impact of perturbing some node in the computational graph on the final output.

This could be a full replacement, ablative, additive etc



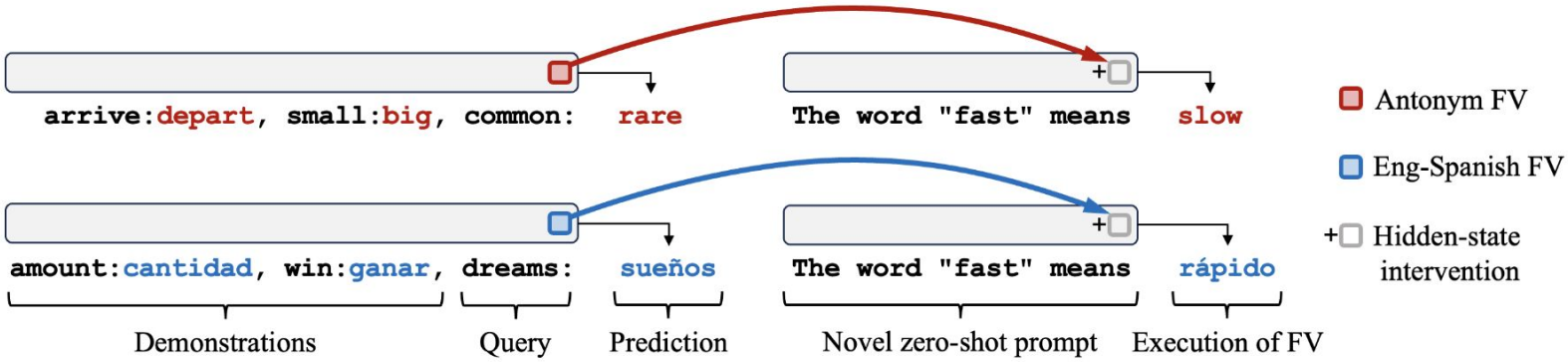
Example Mech Interp Research

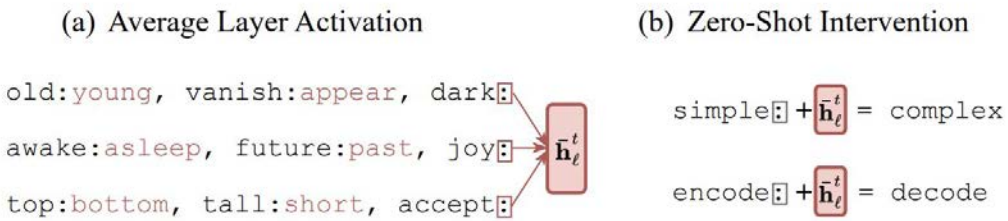
FUNCTION VECTORS IN LARGE LANGUAGE MODELS

Eric Todd,* Millicent L. Li, Arnab Sen Sharma, Aaron Mueller,
Byron C. Wallace, and David Bau
Khoury College of Computer Sciences, Northeastern University

arXiv:2310.15213v1

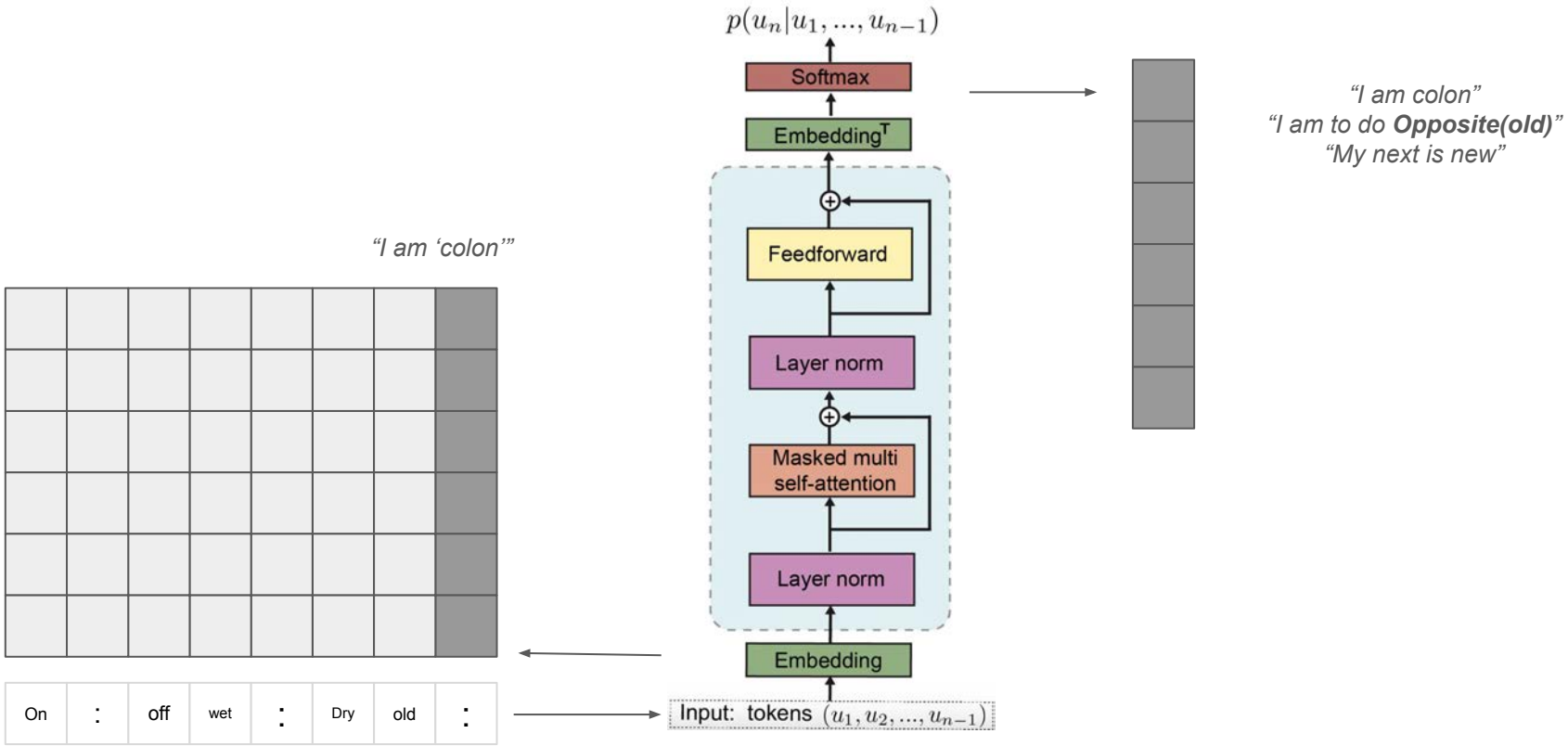
Activation Vectors



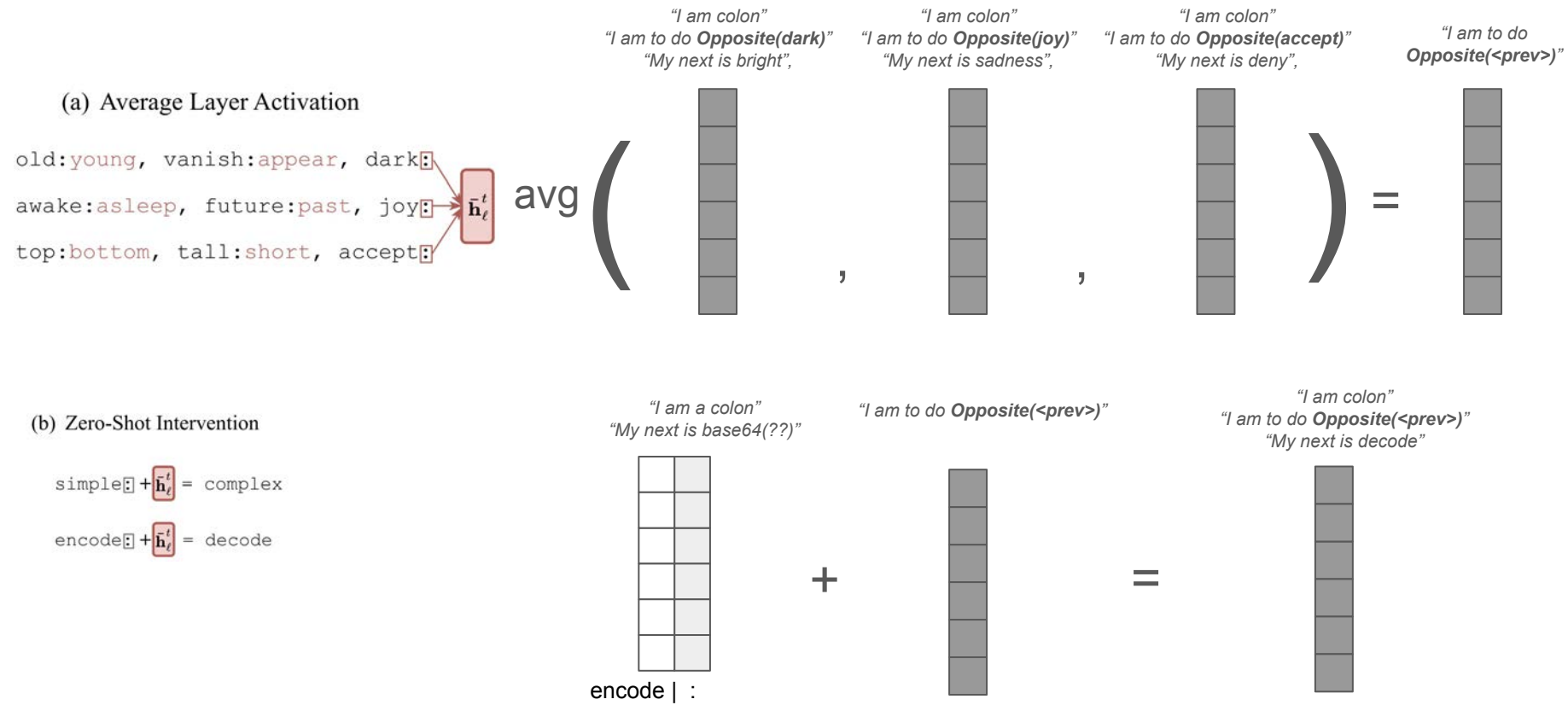


arXiv:2310.15213v1

Information Flow



Averaged-out Information Vectors



Interpretability Libraries and Packages

NNsight

Interpretable Neural Networks

NNsight (/ɛn.saɪt/) is a package for interpreting and manipulating the internals of large models

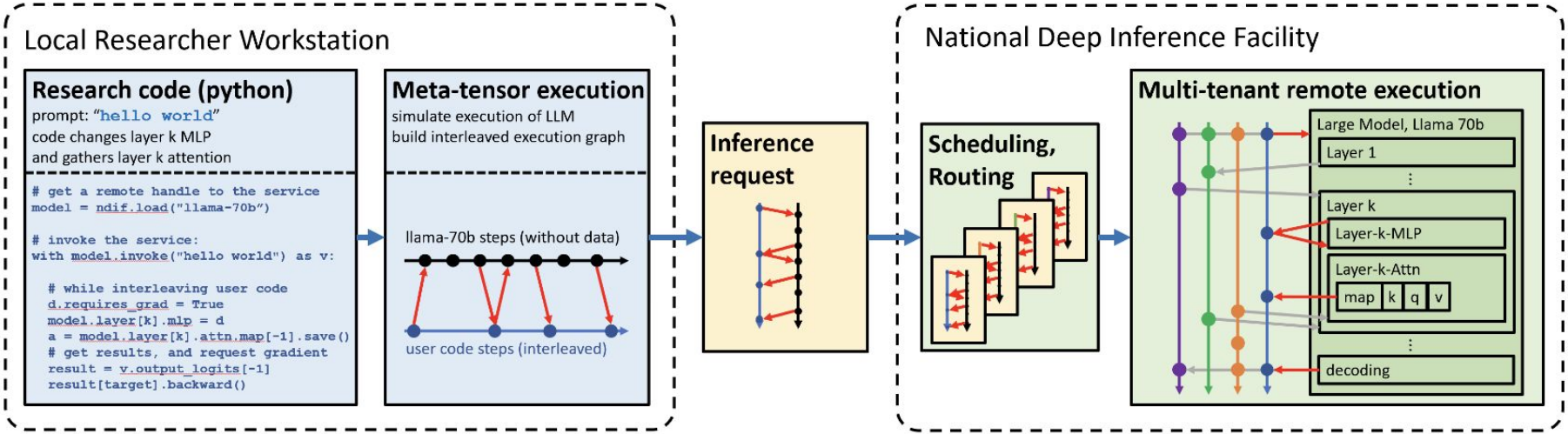
Get Started | Tutorials | Docs

TransformerLens

(Formerly known as EasyTransformer) pypi v1.14.0

A Library for Mechanistic Interpretability of Generative Language Models

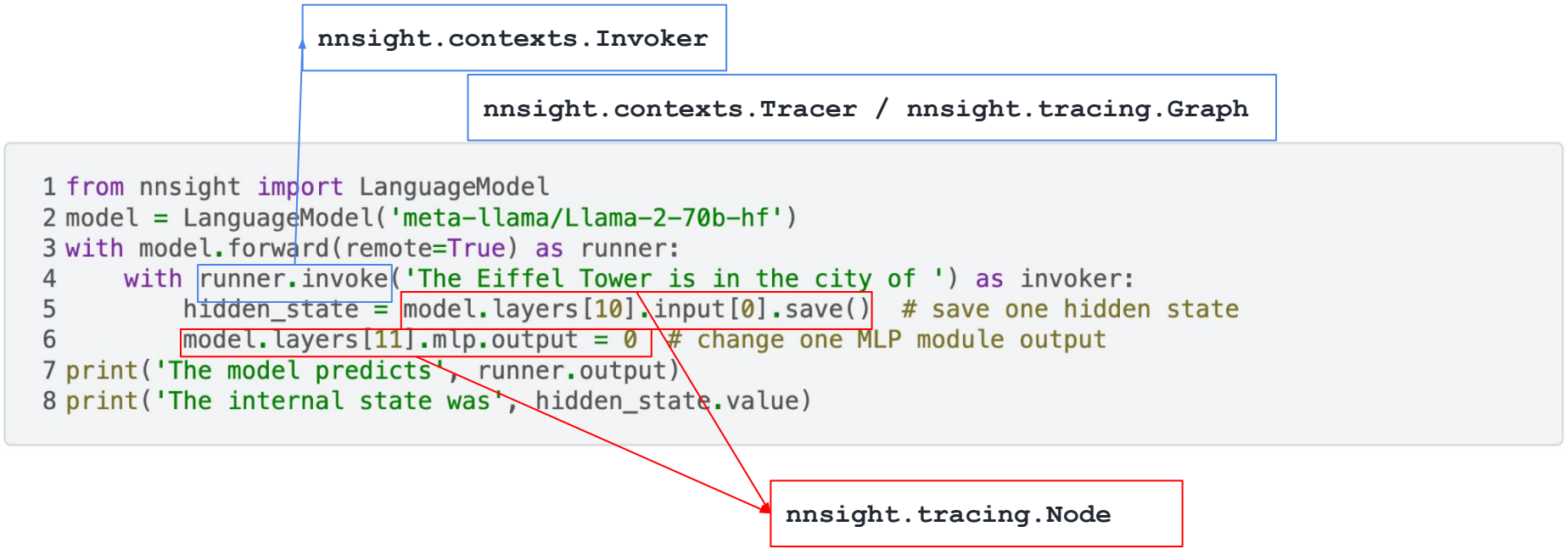
NNsight Architecture



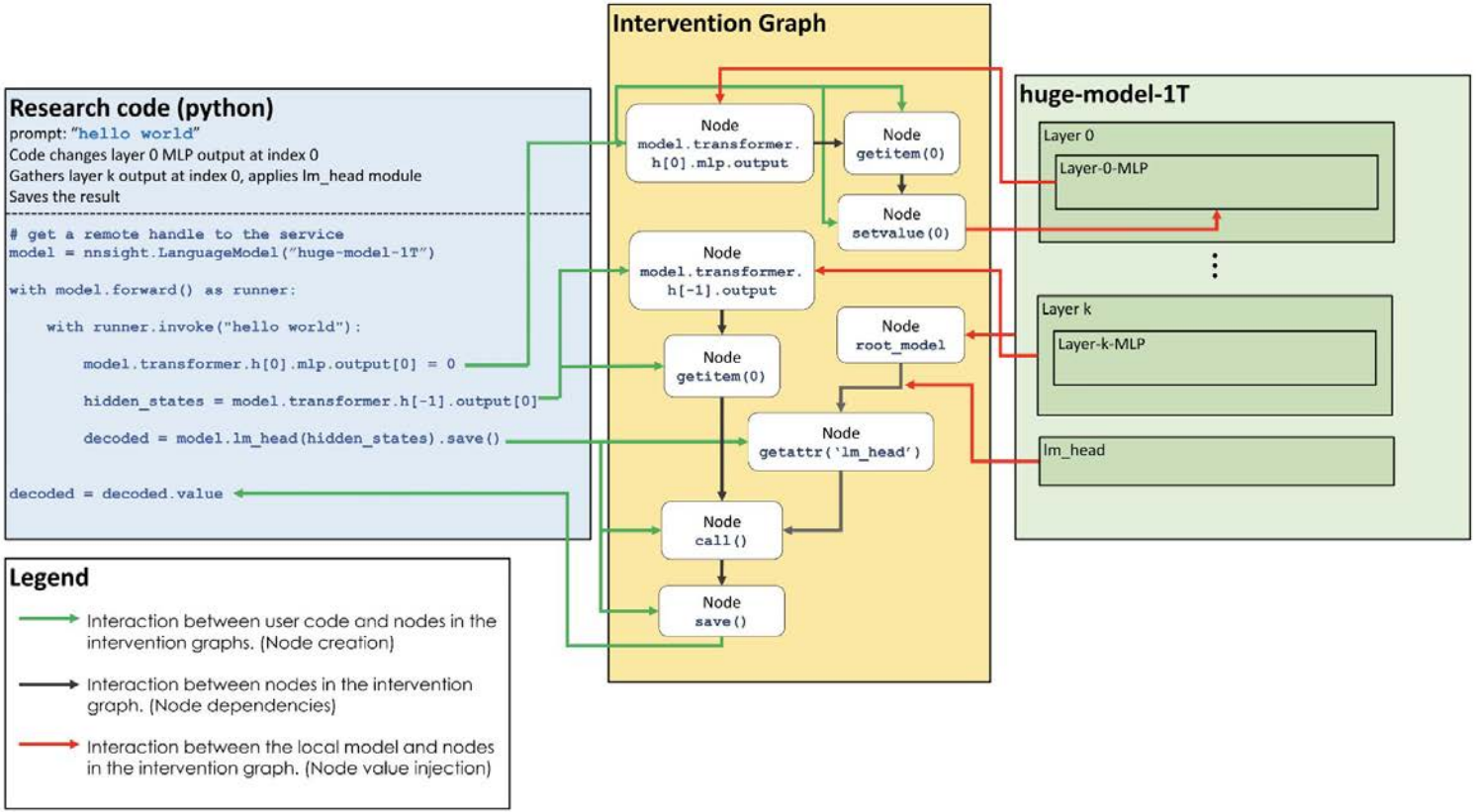
Example Intervention

```python
from nnsight import LanguageModel
model = LanguageModel('meta-llama/Llama-2-70b-hf')
with model.forward(remote=True) as runner:
    with runner.invoke('The Eiffel Tower is in the city of ') as invoker:
        hidden_state = model.layers[10].input[0].save()  # save one hidden state
        model.layers[11].mlp.output = 0  # change one MLP module output
print('The model predicts', runner.output)
print('The internal state was', hidden_state.value)
```

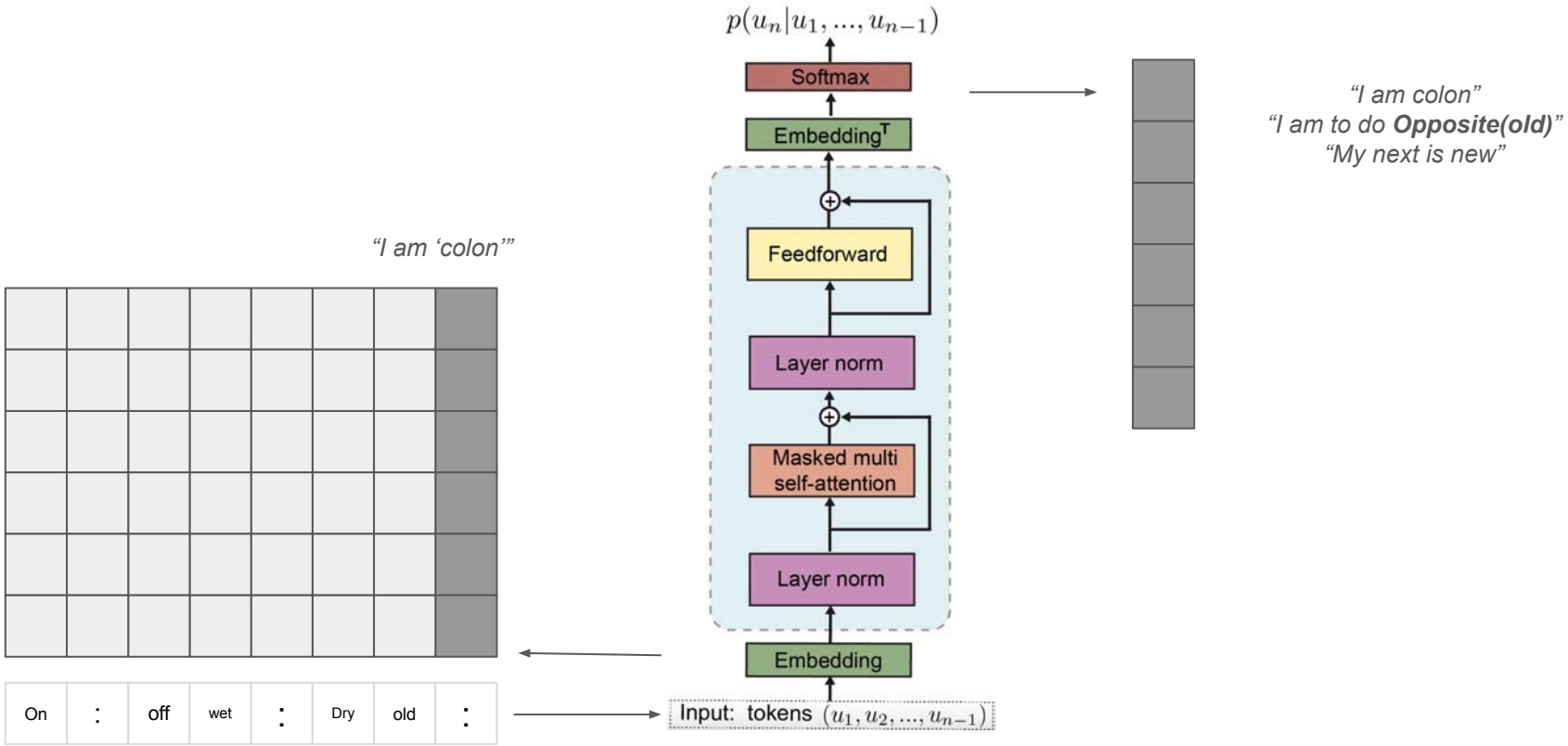
Anatomy of an Intervention



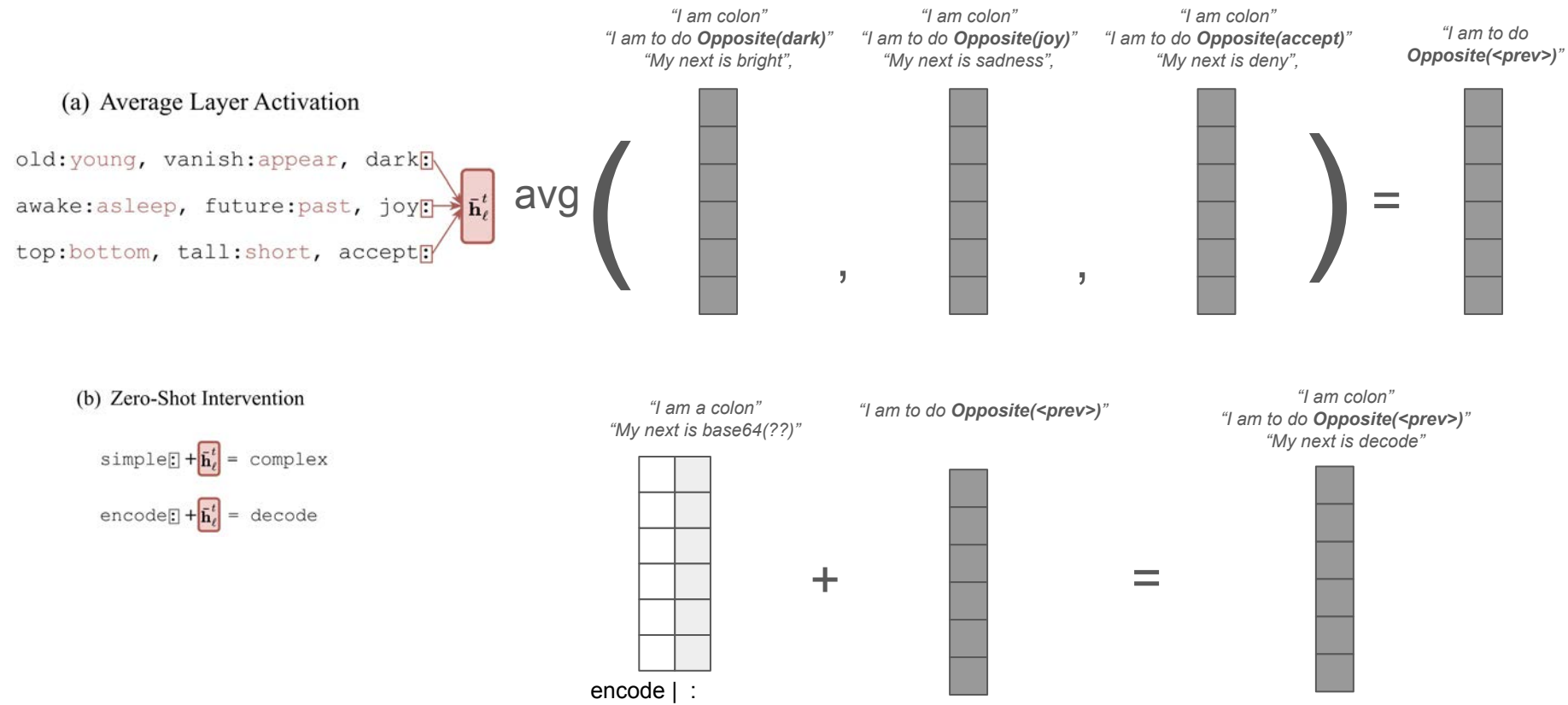
Model Internal I/O are nodes on the Intervention Graph



Information Flow



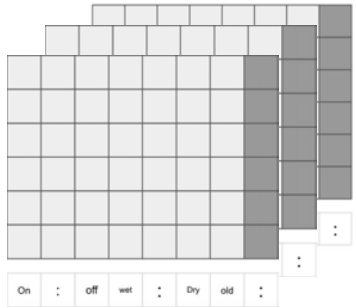
Averaged-out Information Vectors

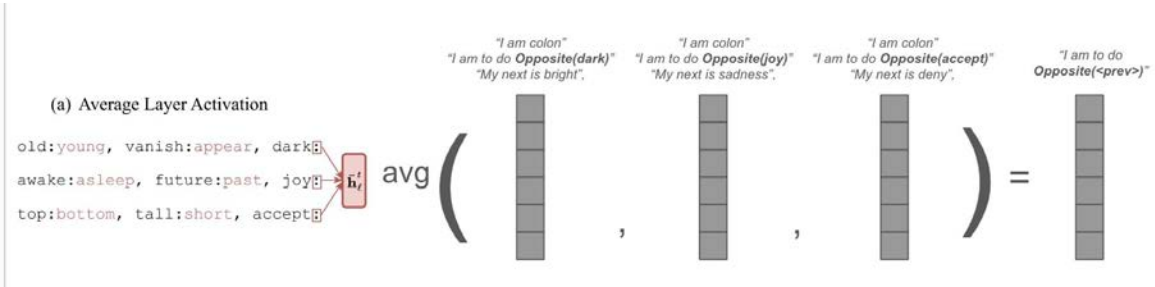


Getting the average activation Vector

```python
sequence_position = -1
layer = 8
with model.forward(remote=True) as runner:
  with runner.invoke(opposites_dataset) as invoker:
      # shape: batch, sequence_length, d_model
      hidden_states = model.transformer.h[layer].output[0]
      # shape: batch, d_model
      hidden_states_at_last_pos = [:, sequence_position]
      # shape: d_model
      average_hidden_state_at_last = hidden_states_at_last_pos.mean(dim=0).save()

h = average_hidden_state_at_last.value
```





Code snippet: Arena.education

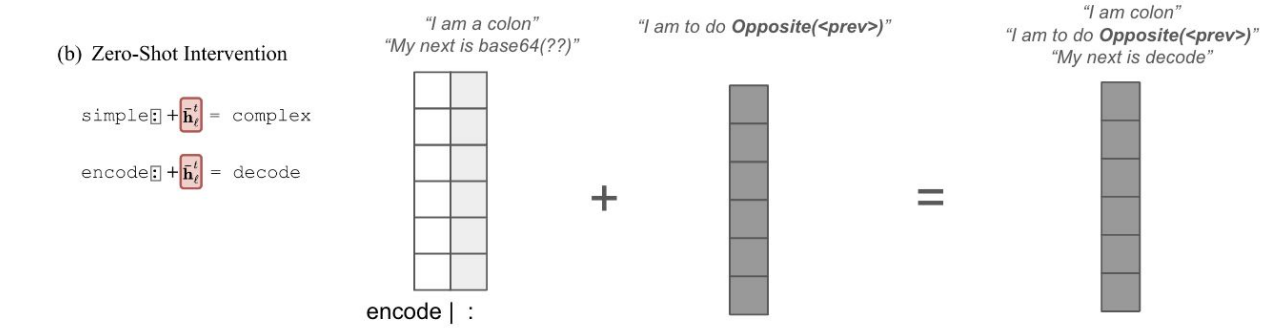
Adding the average to One-shot Activation Vectors

```python
with model.forward(remote=REMOTE) as runner:

    # First, run a forward pass where we don't intervene, just save token id completions
    with runner.invoke(zero_shot_dataset.prompts) as invoker:
      token_completions_zero_shot = model.lm_head.output[:, -1].save()

    # Next, run a forward pass on the zero-shot prompts where we do intervene
    with runner.invoke(zero_shot_dataset.prompts) as invoker:
      # Add the h-vector to the residual stream, at the last sequence position
      hidden_states = model.transformer.h[layer].output[0]
      hidden_states[:, sequence_position] += h
      # Also save completions
      token_completions_intervention = model.lm_head.output[:, -1].save()

compare(token_completions_zero_shot, token_completions_intervention)
```



Code snippet: Arena.education

Results With Average Activation Vector

Prompt	Model's completion (no intervention)	Model's completion (intervention)	Correct completion
minimum ->	' minimum'	' **maximum**'	' maximum'
arrogant ->	' arrogant'	' arrogant'	' humble'
inside ->	' inside'	' **outside**'	' outside'
reject ->	' reject'	' reject'	' accept'
invisible ->	' invisible'	' invisible'	' visible'
victory ->	' victory'	' victory'	' defeat'
up ->	' up'	' **down**'	' down'
open ->	' open'	' **closed**'	' closed'
under ->	' under'	' under'	' over'
inside ->	' inside'	' **outside**'	' outside'
solid ->	' solid'	' solid'	' liquid'
optimist ->	'\n'	' optim'	' pessimist'
noisy ->	' noisy'	' noisy'	' quiet'
guilty ->	' guilty'	' guilty'	' innocent'
answer ->	' yes'	' answer'	' question'
on ->	' I'	' on'	' off'
junior ->	' senior'	' **senior**'	' senior'
loose ->	' loose'	' loose'	' tight'
introduce ->	' introduce'	' introduce'	' remove'
innocent ->	' innocent'	' **guilty**'	' guilty'

Resources

Getting started with the NNSight Library: https://nnsight.net/

Learning Mech Interp: https: //www.arena.education/