NAML
NAVAL APPLICATIONS OF MACHINE LEARNING

Interpretability of Large Language Models

Calvin Raab, Data Scientist, Booz Allen Hamilton

Abstract

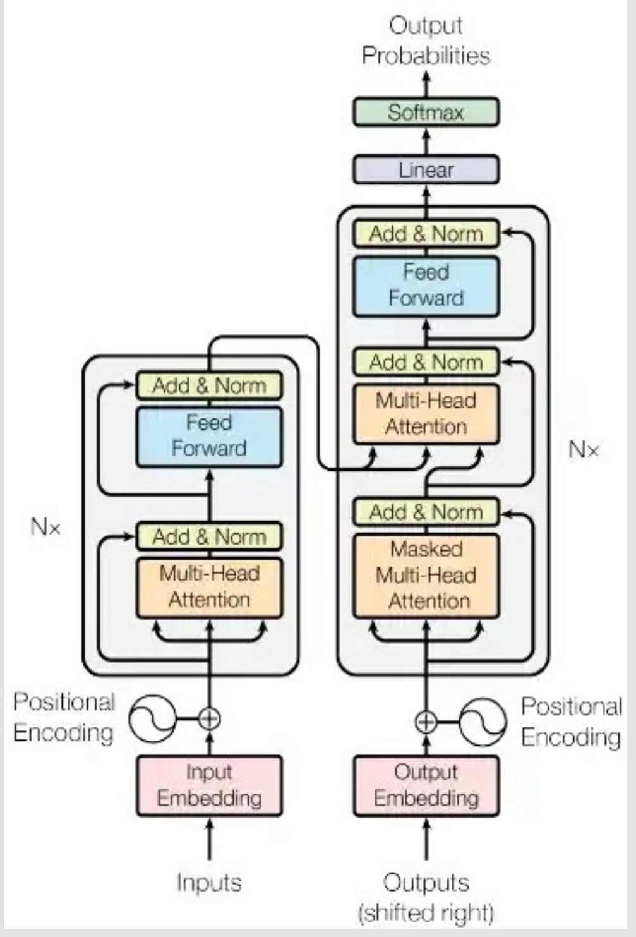
Understanding how Large Language Models (LLMs) store and retrieve knowledge is key to improving their transparency and reliability. This work explores four interpretability techniques: Attention Visualization, Probing Classifiers, Activation Patching, and Causal Tracing. These methods reveal how LLMs track dependencies, encode linguistic and factual knowledge, and store information across layers. The findings show that knowledge is distributed but can be localized, though key questions remain about how it is structured and disentangled. This research provides insights into model explainability and highlights areas for future study.

What is a Large Language Model?

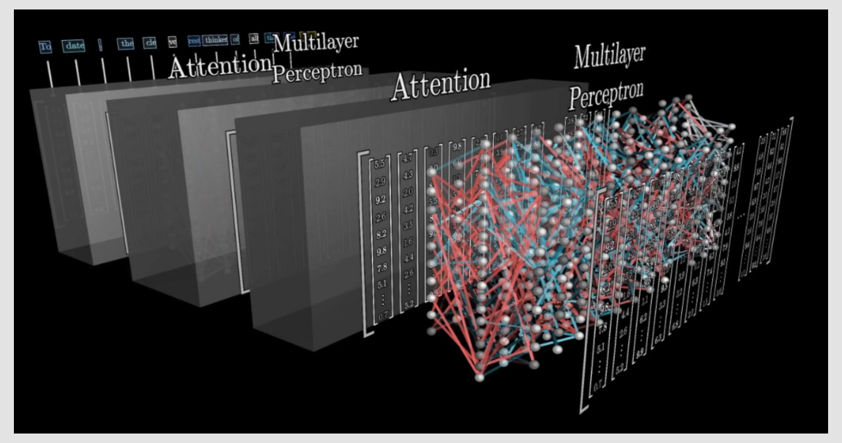
A Large Language Model (LLM) is a deep learning model trained on vast amounts of text and, in some cases, multimodal data (e.g., transcripts, code, and images). Using neural networks, particularly the transformer architecture, LLMs predict the next word in a sequence, enabling tasks like text completion, translation, and question answering [8].

What is a Transformer?

A Transformer is a deep learning architecture introduced in "Attention Is All You Need" [1] that uses self-attention and feed-forward networks to process input sequences in parallel, making it highly efficient for NLP tasks. It consists of an encoder-decoder structure, where the encoder maps input tokens into contextualized embeddings, and the decoder generates outputs autoregressively. This architecture enables models to capture long-range dependencies in text, making it the foundation of modern LLMs.



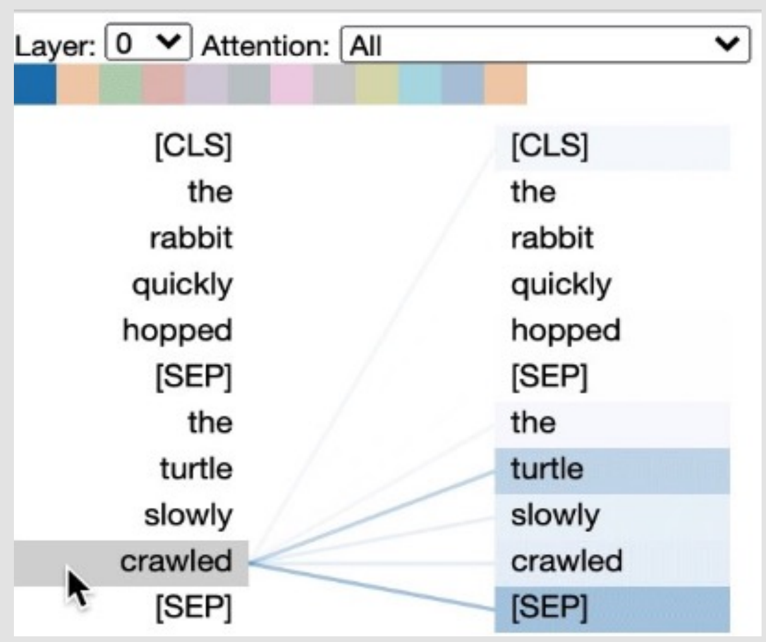
[1 - Attention is All You Need]



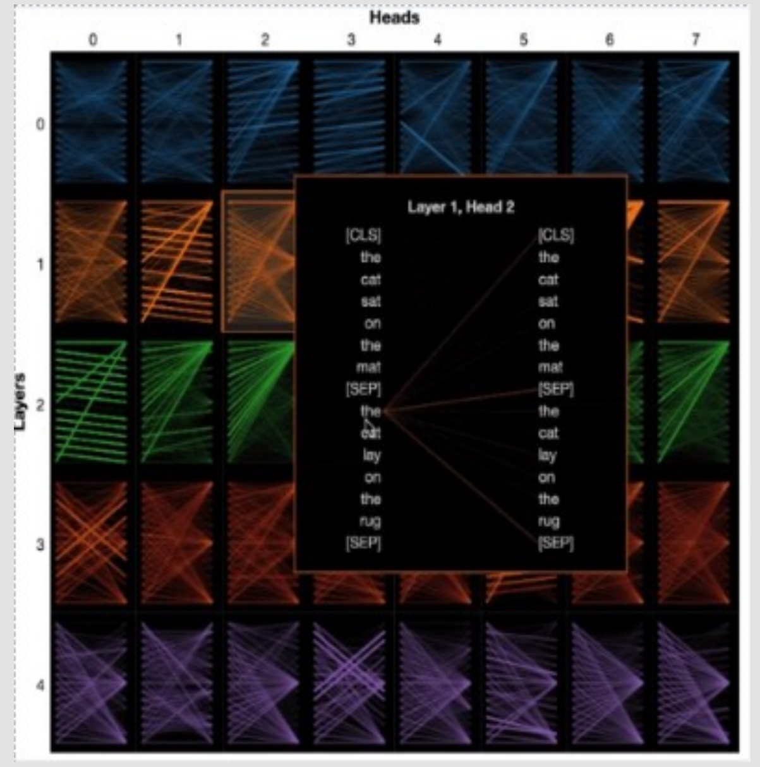
[7 - Grant Sanderson]

Attention Visualization

Attention visualization helps reveal how a language model distributes focus across tokens when processing text. Self-Attention is where the "understanding" of the sentence is created [3]. By examining attention weights at different layers and heads, we can see which words influence each other most strongly. These insights provide a window into model reasoning, showing how LLMs track dependencies, handle ambiguity, and capture context in a sentence.


Layer: 0 Attention: All

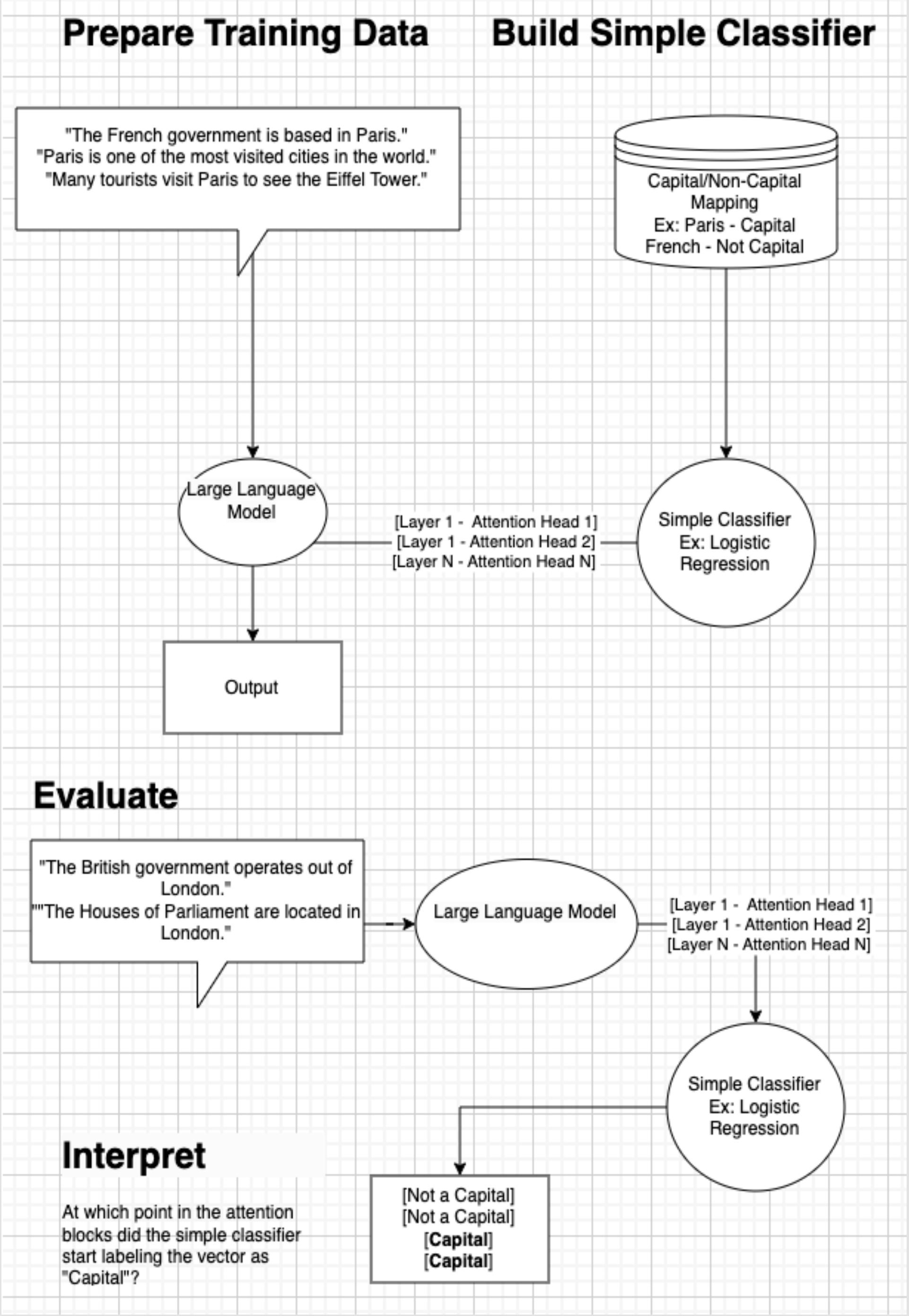
[2 - BertViz]



[2 - BertViz]

Probing Classifiers

Probing classifiers are used to identify where in a model information is stored. A simple classifier (below, a logistic regression) is trained to predict some linguistic property from a model's representation [4]. Below is a diagram representing how one would use a probing classifier to learn when in a model the knowledge of a city being a capital is learned.



Prepare Training Data

"The French government is based in Paris."
"Paris is one of the most visited cities in the world."
"Many tourists visit Paris to see the Eiffel Tower."

Large Language Model

Output

Build Simple Classifier

Capital/Non-Capital Mapping
Ex: Paris - Capital
French - Not Capital

[Layer 1 - Attention Head 1]
[Layer 1 - Attention Head 2]
[Layer N - Attention Head N]

Simple Classifier
Ex: Logistic Regression

Evaluate

"The British government operates out of London."
""The Houses of Parliament are located in London."

Large Language Model

[Layer 1 - Attention Head 1]
[Layer 1 - Attention Head 2]
[Layer N - Attention Head N]

Simple Classifier
Ex: Logistic Regression

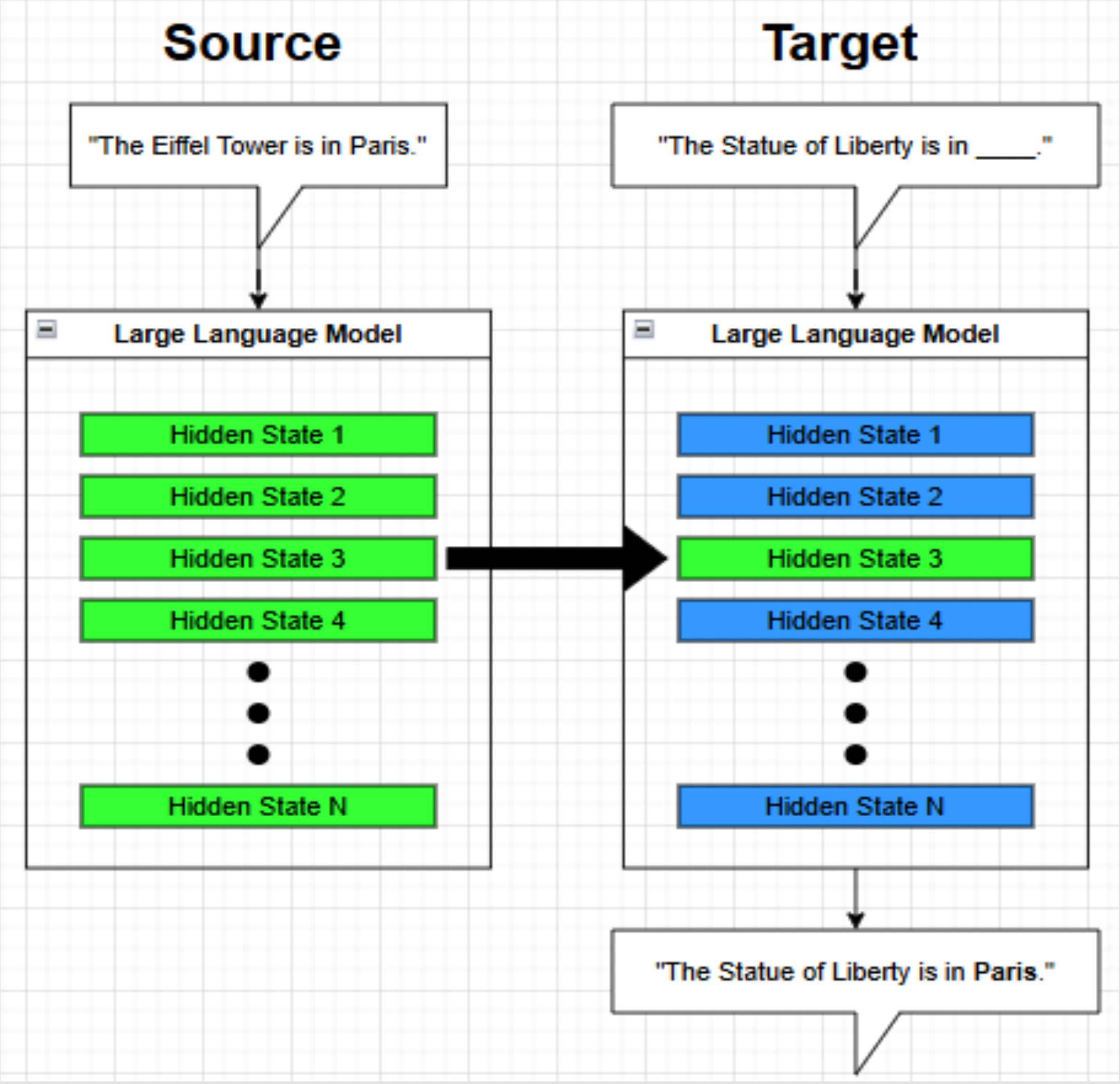
[Not a Capital]
[Not a Capital]
[Capital]
[Capital]

Interpret

At which point in the attention blocks did the simple classifier start labeling the vector as "Capital"?

Activation Patching

Activation patching is a technique for seeing which model activations are most important in determining model output [5]. By extracting hidden states from one input ("The Eiffel Tower is in Paris.") and injecting them into another input ("The Statue of Liberty is in ____"), one can see how the model's output changes. If the model changes its prediction, this indicates that the patched layer is responsible for encoding factual recall.



Source

"The Eiffel Tower is in Paris."

Large Language Model
Hidden State 1
Hidden State 2
Hidden State 3
Hidden State 4
Hidden State N

Target

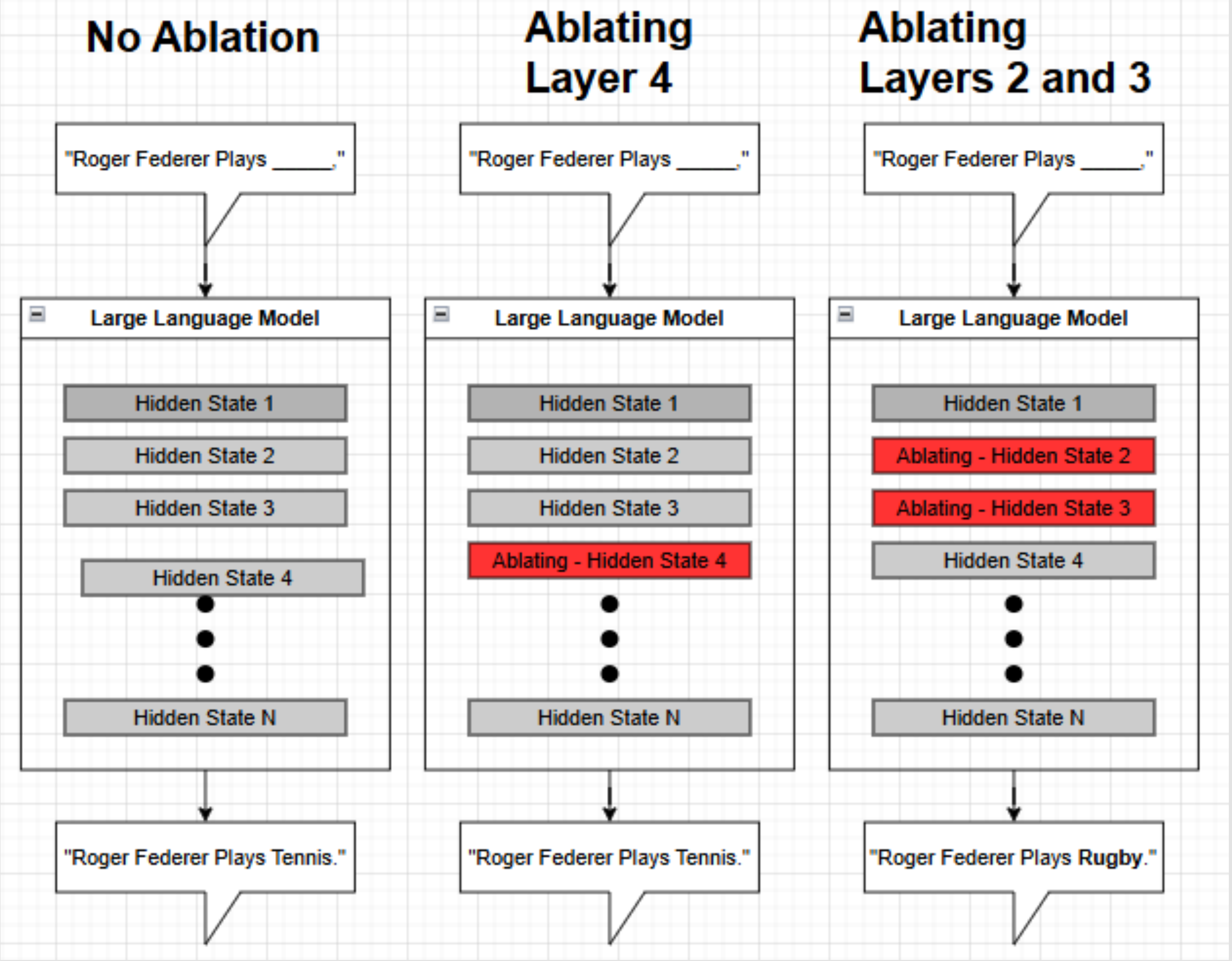
"The Statue of Liberty is in ____."

Large Language Model
Hidden State 1
Hidden State 2
Hidden State 3
Hidden State 4
Hidden State N

"The Statue of Liberty is in **Paris**."

Causal Tracing

Causal tracing is an interpretability method used to identify where factual knowledge is stored in a language model. At a high level this is done by developing a "causal intervention" for identifying critical activations [6]. This technique helps reveal whether knowledge is localized in a single layer or distributed across multiple layers, improving our understanding of how LLMs process and retain information.



No Ablation

"Roger Federer Plays _____."

Large Language Model
Hidden State 1
Hidden State 2
Hidden State 3
Hidden State 4
Hidden State N

"Roger Federer Plays Tennis."

Ablating Layer 4

"Roger Federer Plays _____."

Large Language Model
Hidden State 1
Hidden State 2
Hidden State 3
Ablating - Hidden State 4
Hidden State N

"Roger Federer Plays Tennis."

Ablating Layers 2 and 3

"Roger Federer Plays _____."

Large Language Model
Hidden State 1
Ablating - Hidden State 2
Ablating - Hidden State 3
Hidden State 4
Hidden State N

"Roger Federer Plays **Rugby**."

What We Now Understand?

1. LLMs track dependencies through attention mechanisms
2. LLMs encode linguistic and factual properties at different layers
3. LLMs store factual knowledge in hidden states, which can be manipulated
4. Factual knowledge recall is distributed but can be pinpointed

What is Still Not Understood?

1. Why LLMs sometimes attend to irrelevant tokens.
2. The precise mechanism behind factual storage and retrieval
3. How deeply factual knowledge is entangled with other representations
4. How robust these interpretability methods are across different architectures

Further Research

- Improve scalability: Create computationally feasible techniques for large-scale models.
- Uncover knowledge structures: Investigate how LLMs store and retrieve knowledge.
- Address entanglement: Study how different knowledge types overlap in representations.

Citations

1. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762
2. J. Vig, "BertViz: Visualizing Attention in Transformer Models," GitHub, 2019. [Online]. Available: https://github.com/jessevig/bertviz
3. J. Alammar, "The Illustrated Transformer," *Jay Alammar's Blog*, 2018. [Online]. Available: https://jalammar.github.io/illustrated-transformer/
4. Y. Belinkov, "Probing Classifiers: Promises, Shortcomings, and Advances," *arXiv preprint arXiv:2102.12452*, 2021. [Online]. Available: https://arxiv.org/abs/2102.12452
5. N. Nanda, "Attribution Patching," *Neel Nanda's Blog*, 2023. [Online]. Available: https://www.neelnanda.io/mechanistic-interpretability/attribution-patching
6. K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and Editing Factual Associations in GPT," *arXiv preprint arXiv:2202.05262*, 2022. [Online]. Available: https://arxiv.org/abs/2202.05262
7. Grant Sanderson. "Transformers (how LLMs work) explained visually | DL5." *YouTube*, 04/01/2024, https://www.youtube.com/watch?v=wjZofJX0v4M.
8. W. X. Zhao *et al.*, "A survey of large language models," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2303.18223