Understanding Large Language Models

Carsten Eickhoff, Michael Franke and Polina Tsvilodub

Session 10: Mechanistic Interpretability

@GPT: 机械解释性（Mechanistic Interpretability）是指理解模型内部机制，解释模型是如何处理和生成信息的。这包括了解模型各层次的表示和信息流动。

Main learning goals

1. Recap
   - what-if exploration, SHAP, probing classifiers
2. Early Decoding
   - logit lens, examples
3. Residual Stream
   - rediscovering the Transformer, residual streams, head types
4. Activation Patching
   - setup, examples, metrics, direction
5. Circuit Analysis
   - more heads, causal scrubbing

Slides adapted from: Polina Tsvilodub, Ali Bahreinian, DeepLearning.ai

Recap

Counterfactual Analyses

Q: “how asymmetric are the white bricks on either side of the building”
A: *very*

Q: “how soon are the bricks fading on either side of the building”
A: *very*

Q: “how fast are the bricks speaking on either side of the building”
A: *very*

Paper: Did the model understand the question? ACL 2018

What-if Explorations

@GPT：研究者通过有意改变输入（例如更改某些词或句子的结构），观察模型输出的变化，从而分析模型的行为和特性

Probe the model on various What-If scenarios.

- Examples:
  - What if “he” was replaced with “she”
  - What if we add a punctuation at the end of the sentence
- **Intuitive:** What you see is what you get
- **Highly expressive:** Most explainability techniques are a summarization of what-if behavior

Applications:

- Model understanding / debugging
- Algorithmic Recourse
- Prompt design

SHAP

A company with two employees **Alice** and **Bob**

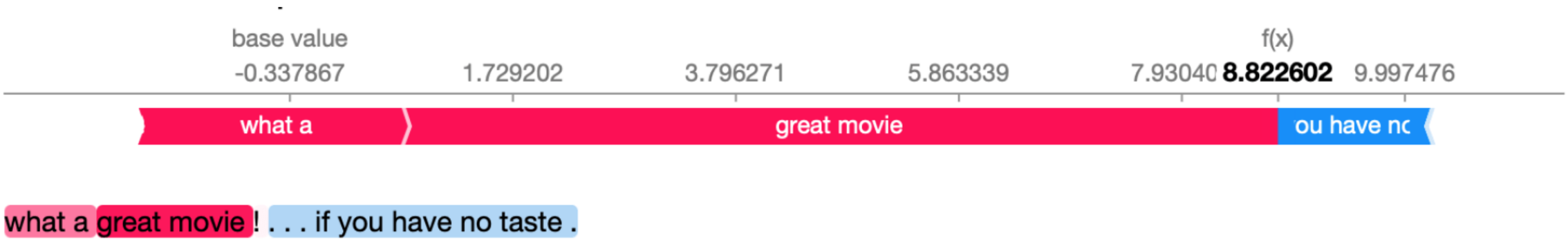
- No employees, no profit [v({}) = 0]
- Alice alone makes 20 units of profit [v({Alice}) = 20]
- Bob alone makes 10 units of profit [v({Bob}) = 10]
- Alice and Bob make 50 units of profit [v({Alice, Bob}) = 50]

What should the bonuses be?

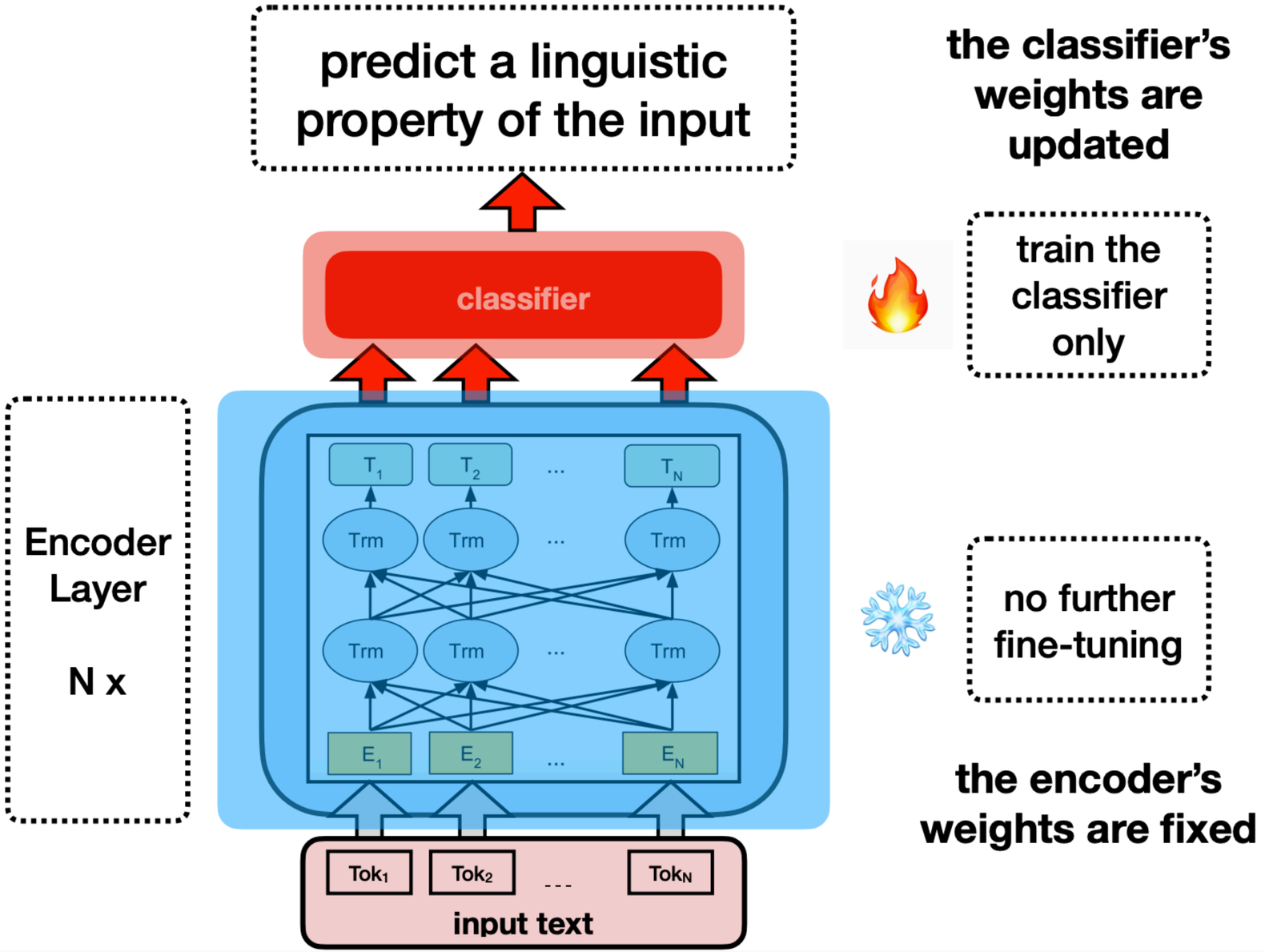
Permutation	**Marginal for Alice**	**Marginal for Bob**
Alice, Bob	20	30
Bob, Alice	40	10
Shapley Value	**30**	**20**

SHAP for NLP

- Tokens as players
- **positive** vs. **negative** affect



Probing


predict a linguistic property of the input
classifier
Encoder Layer
N x
T1
T2
...
TN
Trm
Trm
...
Trm
Trm
Trm
...
Trm
E1
E2
...
EN
Tok1
Tok2

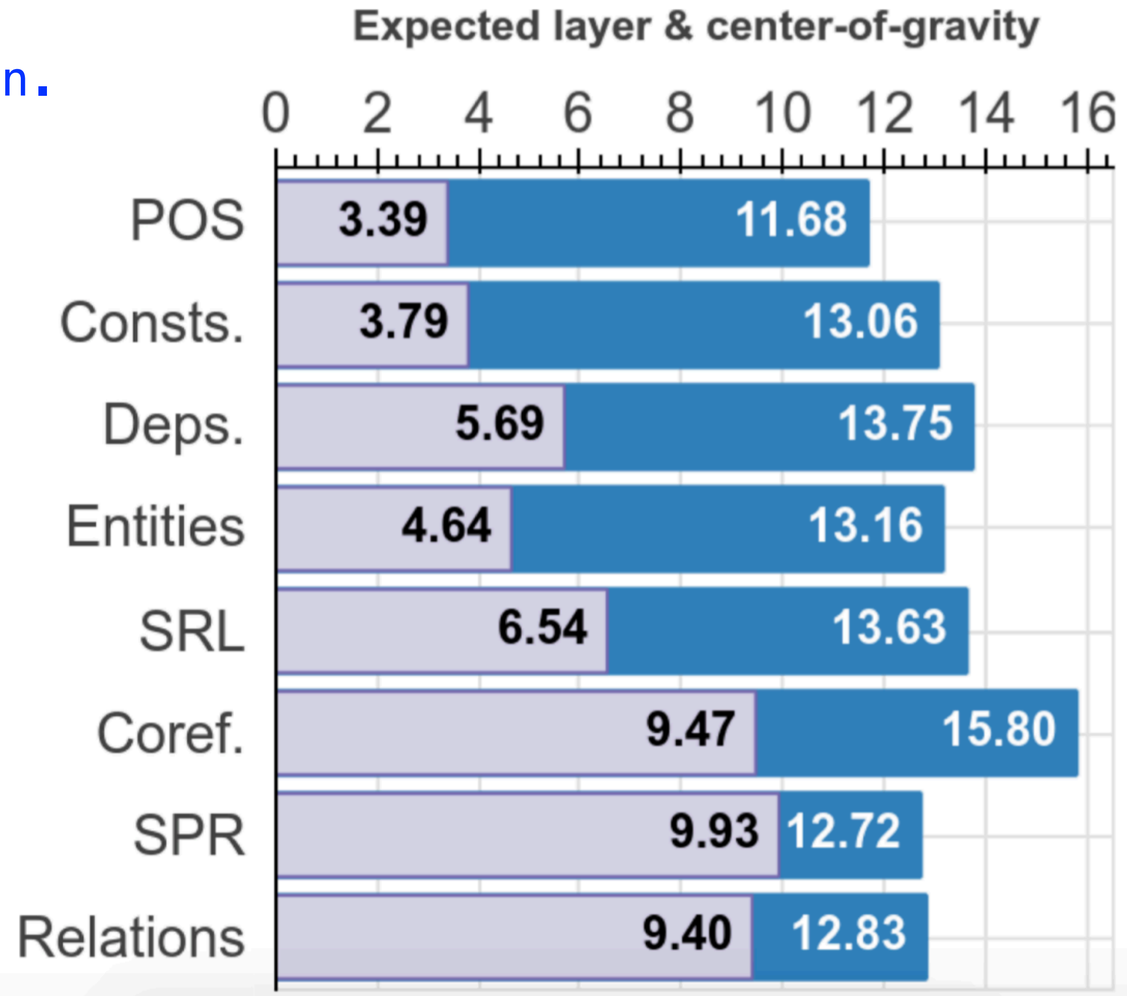
TokN
input text
the classifier's weights are updated
train the classifier only
no further fine-tuning
the encoder's weights are fixed

Probing

the shallow stuff happens at the early layers,
the more involved at later.
historically aggregated information.

the expected layer at which the probing model correctly labels an example

a higher center-of-gravity means that the information needed for that task is captured by higher layers



Where We Stand…

- Jittering the inputs (cleverly) and observing outputs
- Freezing parts of the model to understand their information content
- Problematic interpretation of negatives
  - Absence of signal
  - Low probe performance

Early Decoding

早期解码

Logit Lens

输入投影：将输入词汇投影到高维嵌入空间。
多层处理：嵌入经过多层 Transformer 层的处理。
提前解码：在早期或中间层，将高维嵌入投影回词汇空间。
观察输出：分析这些层次的输出，了解每层的中间表示。

- From a distance, the Transformer:
  - Projects input tokens into high-dimensional embedding space
  - Modifies that space many times
  - Projects the final high-dimensional representation back to vocabulary space
- The shape of the embedding space remains constant across layers
- In the end, we have some dictionary W, that projects back to vocab space
- Let's project to vocab space early
- What do early layers "want to say"?
- Reminiscent of probing classifiers

@GPT:
Logit Lens 是一种探索和可视化语言模型内部工作机制的技术。它的核心思想是通过【早期解码（early decoding）——通过模型早期或中间层将高维嵌入投影回词汇空间，提前解码输出】，从而来观察模型在不同层次的表示。

这页幻灯片讲述了 Logit Lens 技术及其在 Transformer模型中的应用。
它介绍了如何通过 early decoding 提前将模型的中间层表示投影到词汇空间，从而观察和理解模型在不同层次的内部表示。
这种方法类似于探测分类器，通过分析模型中间层的输出，帮助研究者更好地理解模型的工作机制和每一层“想要说什么”。

nostralgebraist. Interpreting GPT - The Logit Lens. 2020

Logit Lens

model's top token and its logit

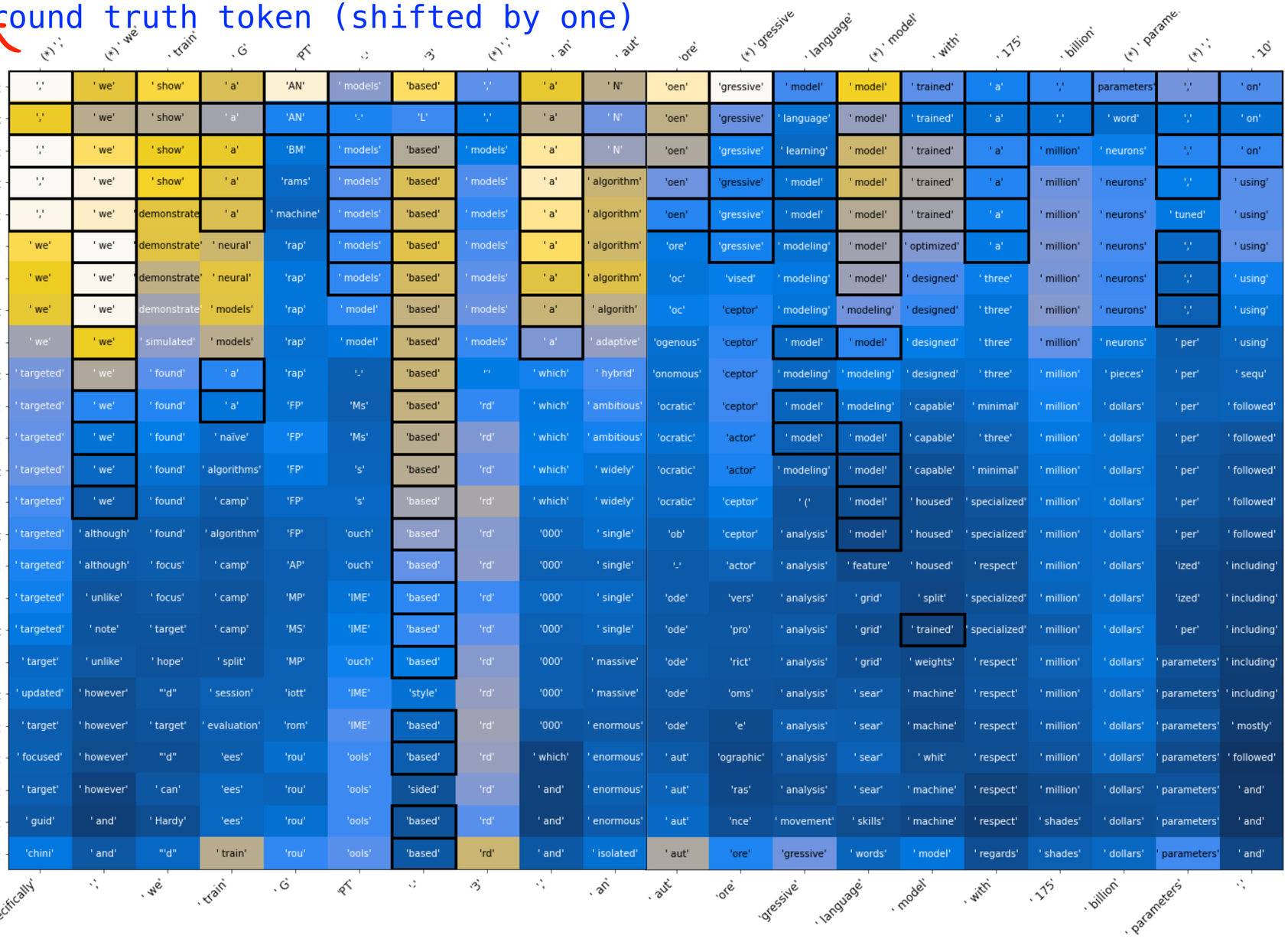
colour: the whiter, the more certain

ground truth token (shifted by one)

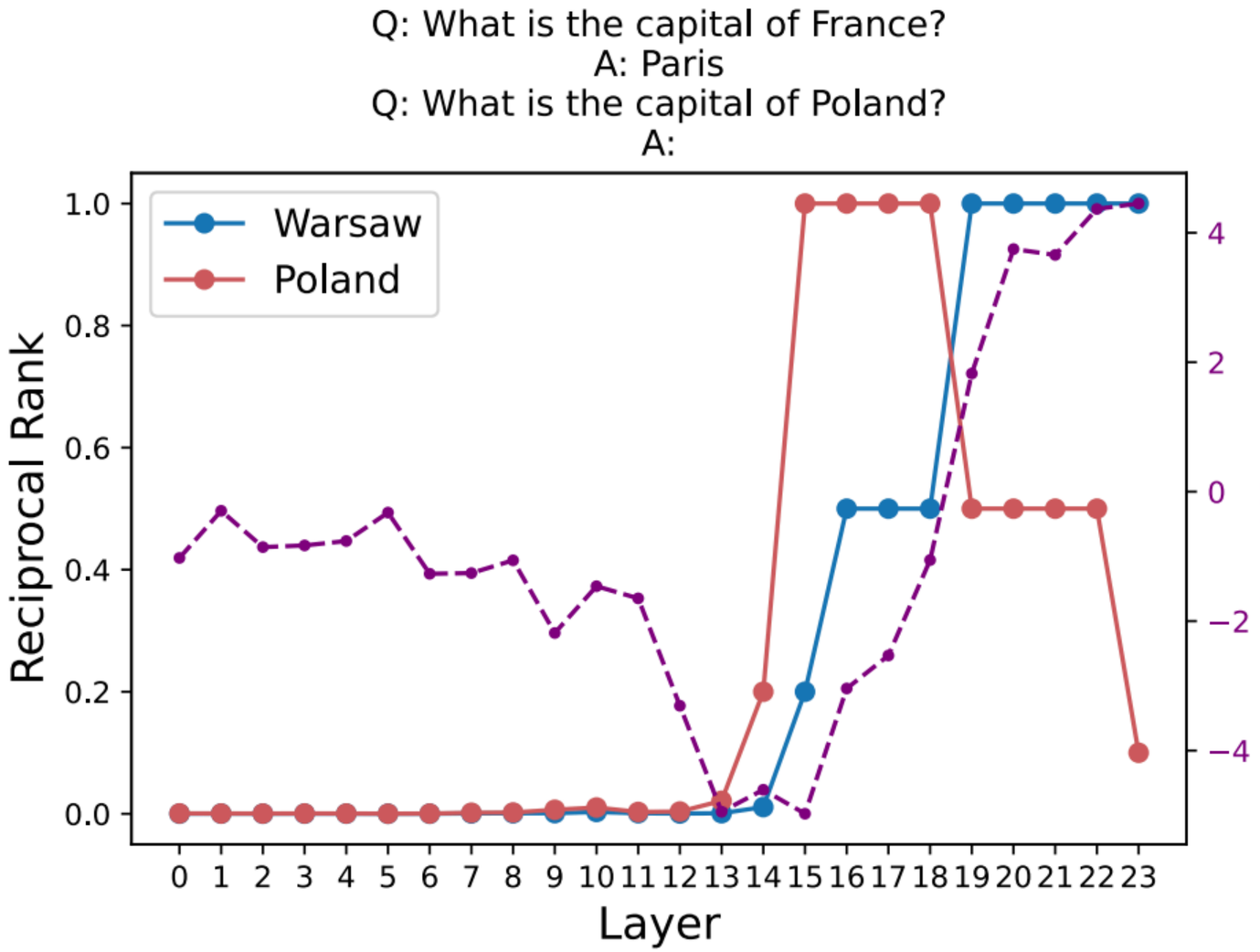
in the later layers, we are more certain

layers of model

input sent



Understanding Question Processing in Models



Layer	Top Token	
0	(	
1	A	
2	A	
3	A	
4	A	
5	A	
6	No	
7	C	
8	A	
9	A	
10	A	
11	A	
12	Unknown	
13	C	
14	St	
15	Poland	A
16	Poland	
17	Poland	
18	Poland	
19	Warsaw	B
20	Warsaw	C
21	Warsaw	
22	Warsaw	
23	Warsaw	

early layers generate rubbish, coz at this stage it's trying to understand the user prompts, haven't started generation

in the middle, getting in to topicality, tho wrong, but proper words

Merullo et al. Language Models Implement Simple Word2Vec-style Vector Arithmetic. 2024

@GPT: 残差流。

Residual Stream 是指在 Transformer 模型中，各层的输出与输入通过残差连接（residual connections）相加，从而在每层都保留原始输入信息的一部分。

这种结构使得信息可以更容易地在深层网络中流动，缓解梯度消失问题，并有助于模型更好地捕捉和利用特征。

Residual Stream

The Transformer Architecture (again)

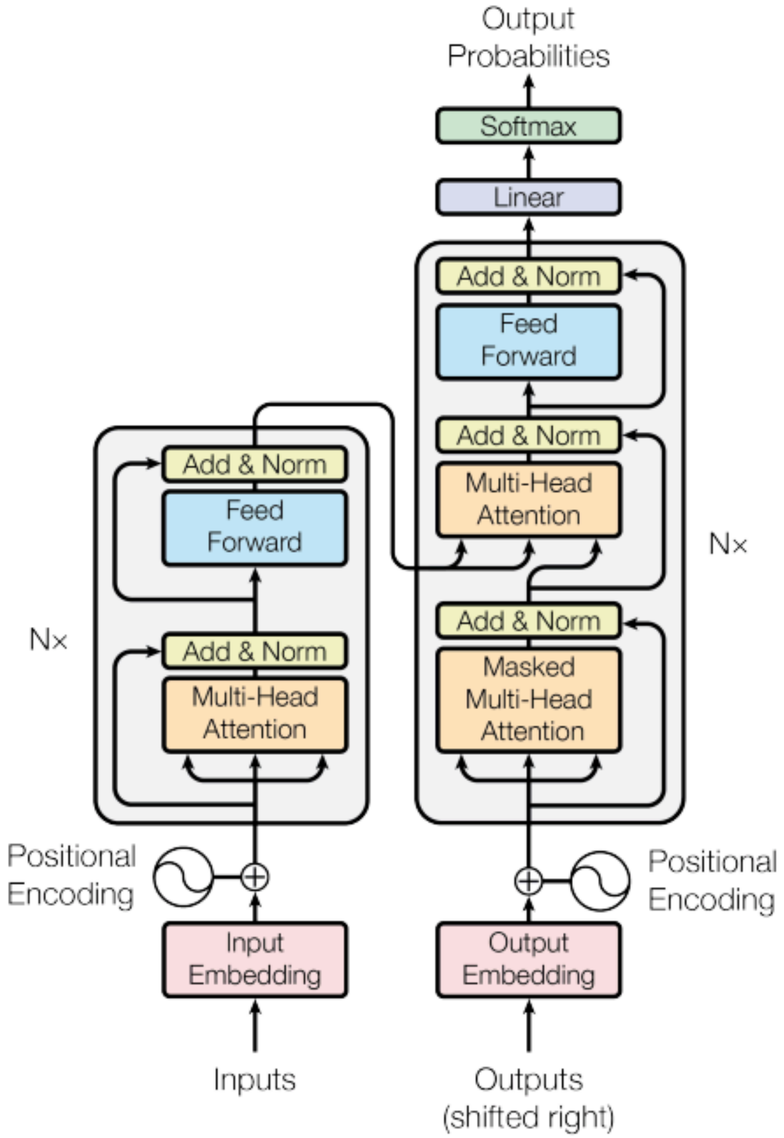
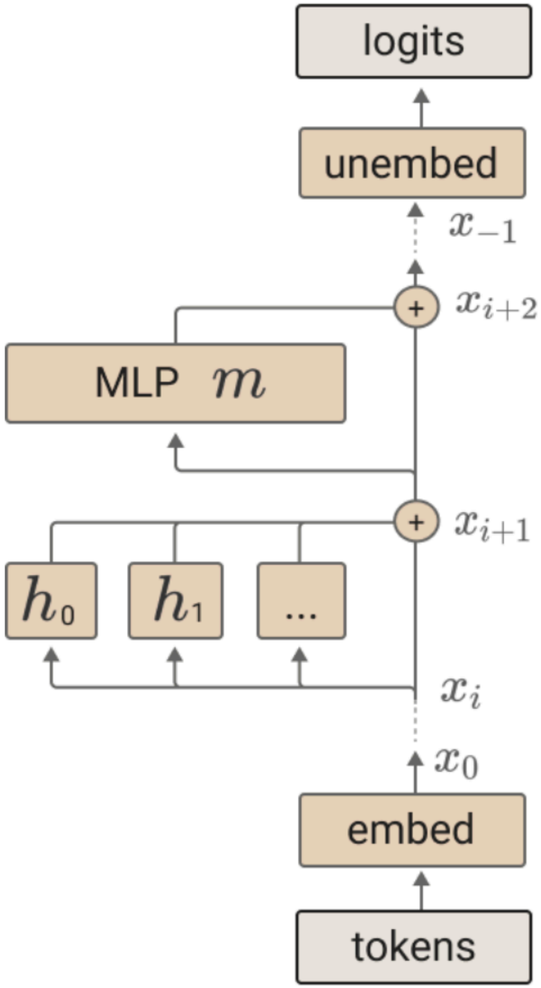


Figure 1: The Transformer - model architecture.

Vaswani et al. Attention is all you need. 2017

The Transformer Architecture (yet again?)



The final logits are produced by applying the unembedding.

$$T(t) = W_U x_{-1}$$

An MLP layer, m, is run and added to the residual stream.

$$x_{i+2} = x_{i+1} + m(x_{i+1})$$

Each attention head, h, is run and added to the residual stream.

$$x_{i+1} = x_i + \sum_{h \in H_i} h(x_i)$$

One residual block

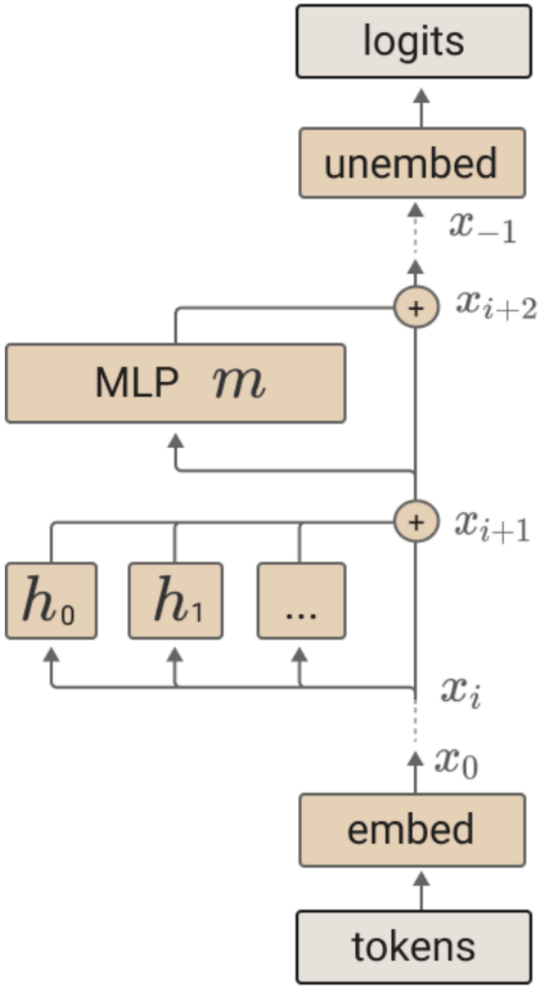
Token embedding.

$$x_0 = W_E t$$

nice paper, recommended

Elhage et al. A Mathematical Framework for Transformer Circuits. 2021

The Transformer Architecture (yet again?)



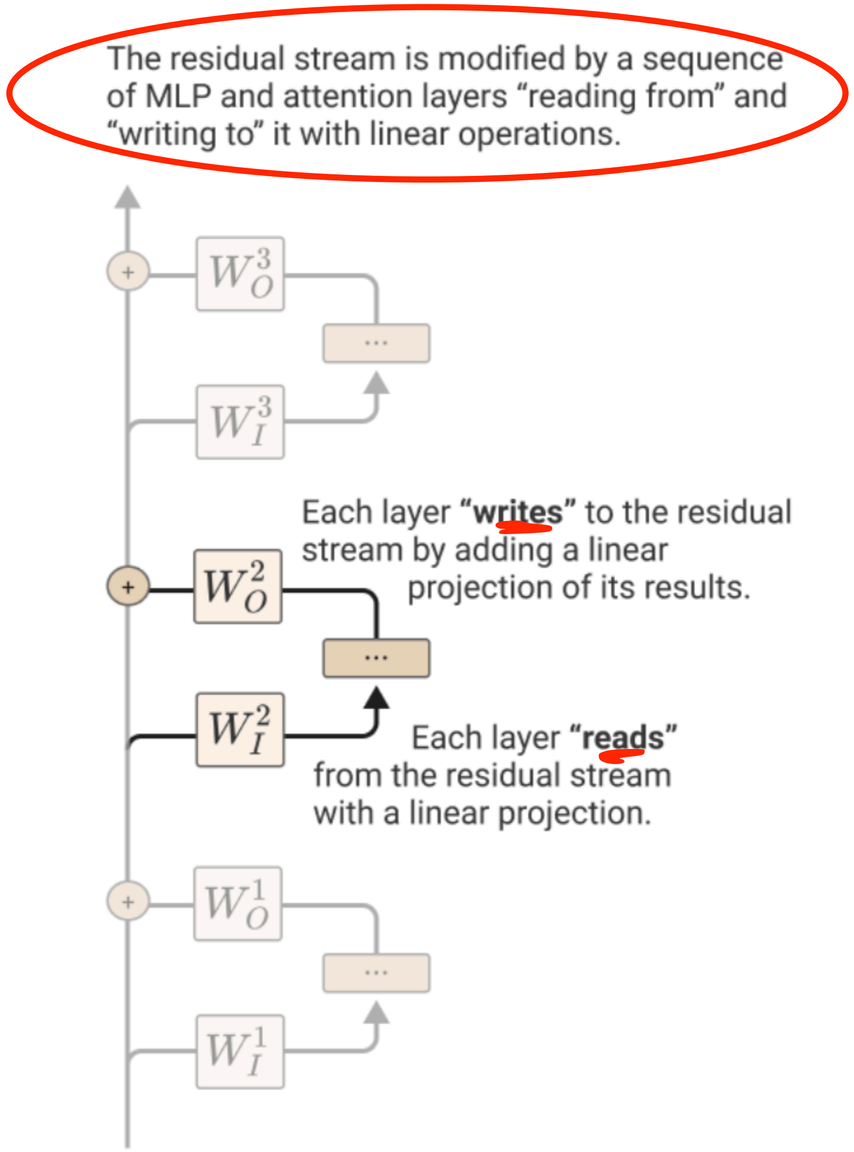
we are going to

it's a

that has

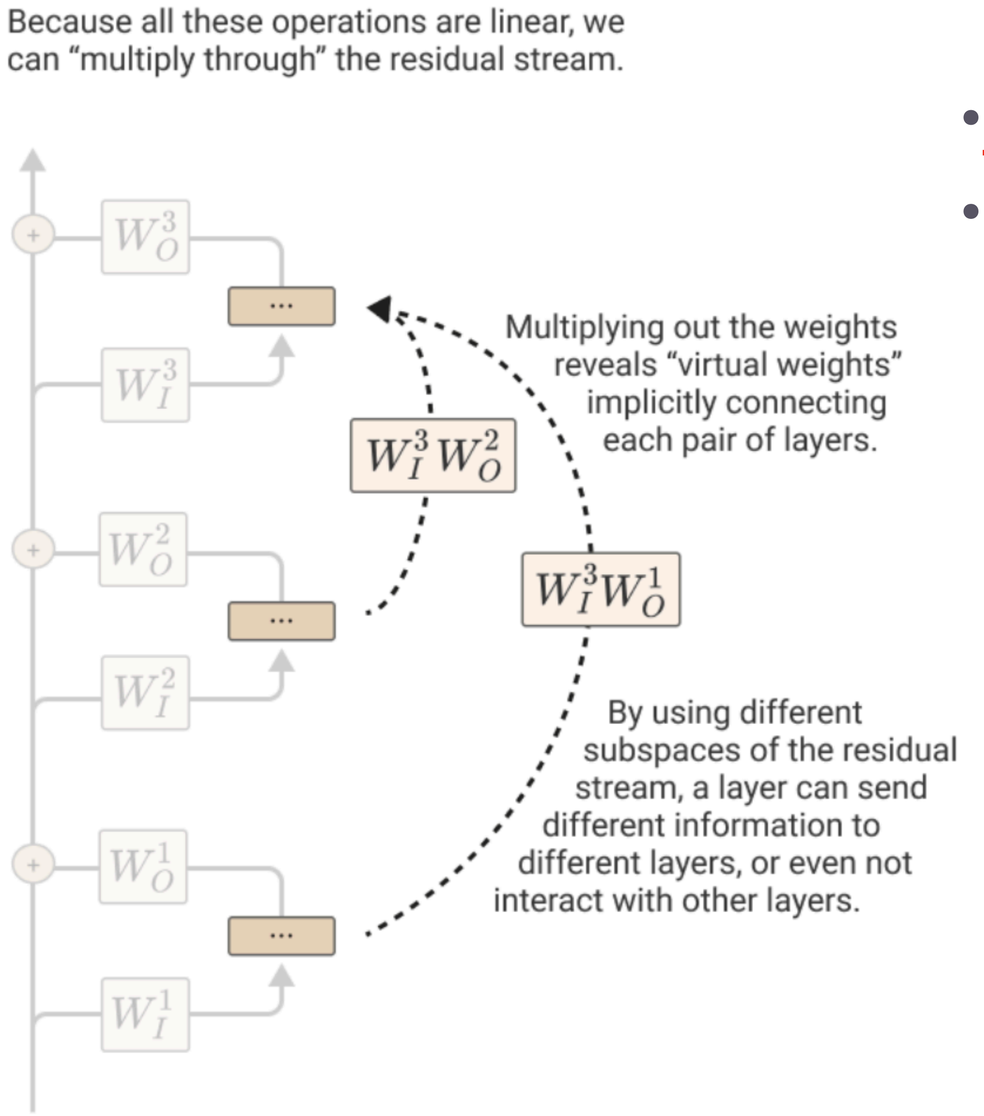
- Working with simplified "Toy Transformers"
- Decoder-only models (à la GPT)
- No MLP layers
- No biases
- No layer normalization

The Transformer Architecture Rewritten



- Exploiting the linear structure of the Transformer
- Communication via the Residual Stream
  - Read from the stream
  - Add a linear transformation to the stream
- Challenge:
  - Notation for efficiency (code) vs.
  - Notation for human interpretation (here)

Virtual Communication Channels



- Implicit virtual weights “connecting” layers
- How much does an input layer read from a previous one?

通过子空间实现info跨层传递

Bandwidth and Subspaces

- The residual stream is high-dimensional
- 100s to 10,000s of dimensions
- Each head operates on small subspaces
- The computational bandwidth is much higher than the communication bandwidth
- “Bottleneck Activations”
- Some heads operate as “memory managers”

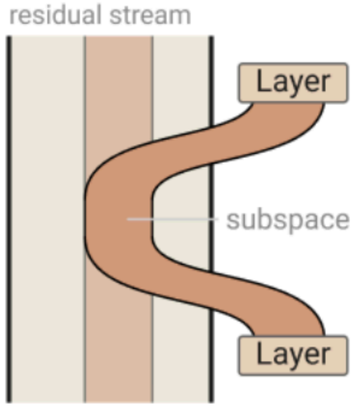
Residual stream is high-dimensional with 100s to 10,000s of dimensions,

but each attention head only operates on small subspaces of this high-dimen space, on particular features or patterns.

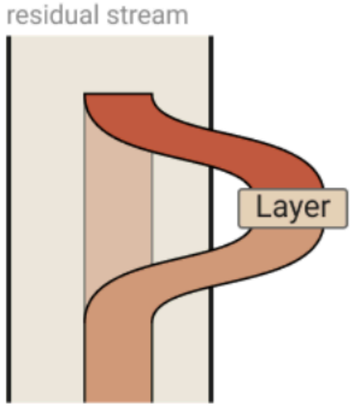
计算带宽：模型在处理和计算时能够处理的信息量（高，因为操作是在高维空间中进行的）。
通信带宽：模型层之间传递信息的能力（相对低，因为每层的信息传递是通过较小的子空间进行的）。
瓶颈激活（Bottleneck Activations）：在信息传递过程中，会存在一些限制点或瓶颈，这些点限制了信息的流动效率。

有些注意力头起到“内存管理”的作用，负责存储和检索重要信息，帮助模型在处理过程中有效地利用和传递信息。

The residual stream is high dimensional, and can be divided into different subspaces.



Layers can interact by writing to and reading from the same or overlapping subspaces. If they write to and read from disjoint subspaces, they won’t interact. Typically the spaces only partially overlap.

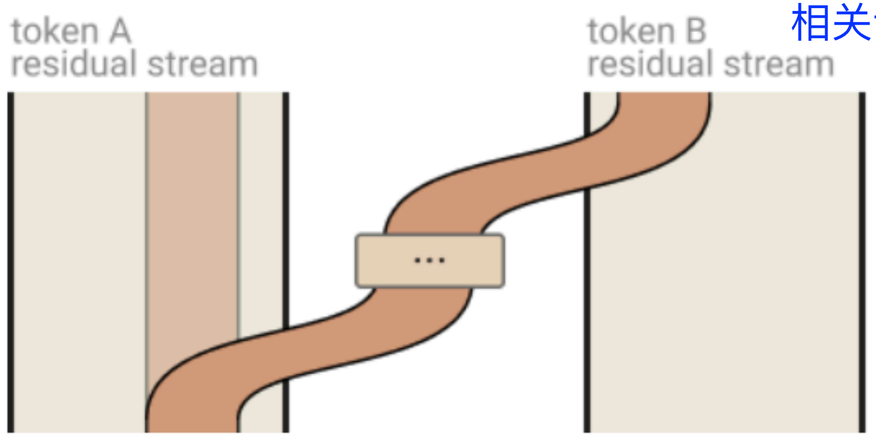


Layers can delete information from the residual stream by reading in a subspace and then writing the negative verison.

Information Management

- The read and write operations of parallel heads happen independently
- Often these updates cross the token boundary
- Linear subspaces relating to other tokens emerge in "contextual word embeddings"

att_head可以独立进行读写操作，不受其它head影响
att_head通过跨越词边界的读写更新操作来建立词与词之间的联系
相关词的子空间会在cwe中自然出现，embedding包含上下文其他词的关系信息



Attention heads copy information from the residual stream of one token to the residual stream of another. They typically write to a different subspace than they read from.

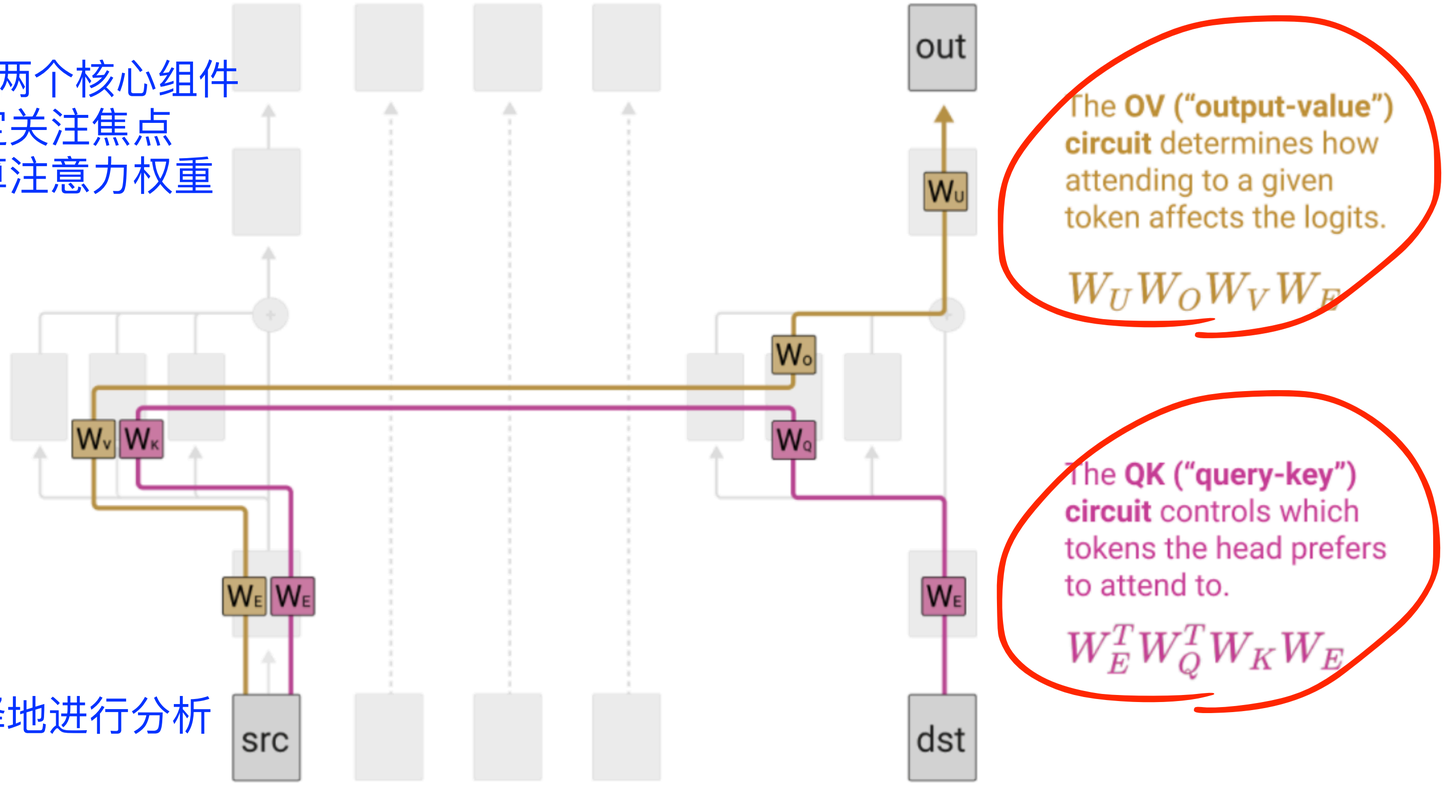
一般从A token读，写入B token，
即read & write subspace一般不同

Dissecting Attention Heads

解剖

- Two key operations in attention heads:
  - Query-key (QK) circuit
  - Output-value (OV) circuit
- Separating these operations allows understanding of mechanisms
- But these matrices are still huge
- Let's cherry-pick

head的两个核心组件
QK 确定关注焦点
OV 计算注意力权重



分离后matrices QKVO仍然过大过复杂，因此有选择地进行分析

The Power of Copying

- Two key operations in attention heads:
  - Query-key (QK) circuit
  - Output-value (OV) circuit
- Separating these operations allows understanding of mechanisms
- But these matrices are still huge
- Let's cherry-pick
- Most Transformer operations are achieved via copying

Source Token	Destination Token	Out Token	Example Skip Tri-grams
" perfect"	" are", " looks", " is", " provides"	" perfect", " super", " absolute", " pure"	" perfect... are perfect", " perfect... looks super"
" large"	" contains", " using", " specify", " contain"	" large", " small", " very", " huge"	" large... using large", " large... contains small"
" two"	" One", "\n ", " has", "\r\n ", "One"	" two", " three", " four", " five", " one"	" two... One two", " two... has three"
"lambda"	" $\\", "}{\\", "+\\", "(\\", " ${\\"	"lambda", "sorted", " lambda", "operator"	"lambda... $\\lambda", "lambda... +\\lambda"
"nbsp"	"&", " \"&", "}&", ">&", "=&"	"nbsp", "01", "gt", "00012", "nbs", "quot"	"nbsp... ", "nbsp... > "
"Great"	"The", " The", " the", " contains", " /"	" Great", " great", " poor", " Every"	"Great... The Great", "Great... the great"

模型在处理信息时，常常是通过复制输入数据或中间表示来生成输出。

Repairing the Tokenizer

- Tokenizers split up some terms into multiple tokens
- Some English terms are multi white-space-separated to begin with
- Our models have to deal with it
- We can observe the consequences in the realized QK/OV matrices

见下页

一些term需要被拆分成多个tokens，且一些english term本身就包含多个空格分割，会在tokenise时产生额外tokens

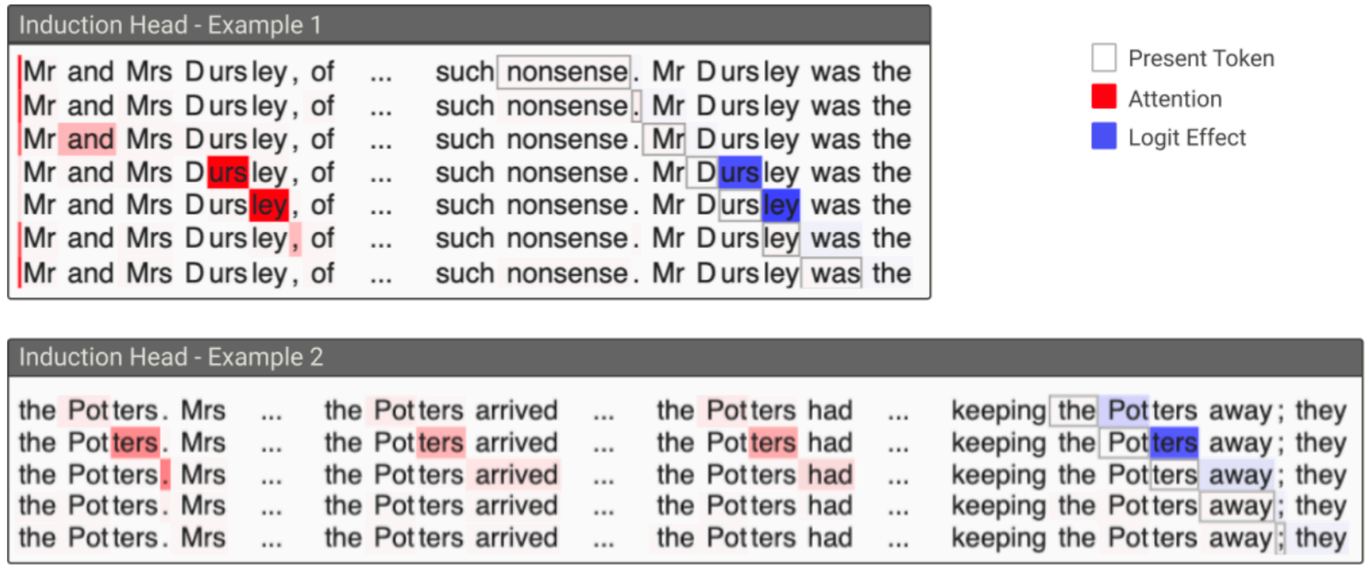
More examples of large entries QK/OV circuit

Source Token	Destination Token	Out Token	Example Skip Tri-grams
“indy”	“ C”, “C”, “ V”, “V”, “ R”, “ c”	“indy”, “obby”, “INDY”, “loyd”	“indy... Cindy”, “indy... CINDY”
“ Pike”	“ P”, “P”, “V”, “Sp”, “ V”, “R”	“ike”, “ikes”, “ishing”, “owler”	“ Pike... Pike”, “ Pike... Spikes”
“ Ralph”	“R”, “ R”, “ P”, “P”, “V”, “ r”	“alph”, “ALPH”, “obby”, “erald”	“ Ralph... Ralph”, “ Ralph... RALPH”
“ Lloyd”	“L”, “ L”, “ P”, “P”, “R”, “ C”	“loyd”, “alph”, “\n ”, “acman”, ... “atherine”	“ Lloyd... Lloyd”, “ Lloyd... Catherine”
“ Pixmap”	“ P”, “ Q”, “P”, “ p”, “ U”	“ixmap”, “Canvas”, “Embed”, “grade”	“ Pixmap... Pixmap”, “ Pixmap... QCanvas”

Head Types: Induction Heads

诱导头

- The QK/OV circuits let us observe the emergence of specialized attention head types
- **Induction Heads** try to complete patterns previously encountered in the context:
  - If the current token has been seen before, attend to the immediately next token
  - Otherwise, dump attention onto the first token in the sequence.



- 对上下文中已见过的token，induction head会直接关注immediately next token
- 对unseen token，会集中注意力到序列第一个token

在 Transformer 模型中，Residual Stream 是一种关键结构，通过在每一层之间使用残差连接，保留了输入信息的一部分，从而帮助信息在深层网络中有效流动。每个 Transformer 层的残差流是高维的，通常有数百到数万个维度。然而，每个注意力头只在这些高维空间的较小子空间中操作，导致计算带宽远高于通信带宽。也就是说，模型的计算能力在处理这些高维信息时非常强大，但在不同层之间的信息传递则受限于较小的子空间，这导致了所谓的“瓶颈激活”(Bottleneck Activations)。

在这种背景下，Logit Lens 技术允许我们通过提前解码，探索和理解 Transformer 各层的内部表示。这种方法通过在模型的早期或中间层将高维嵌入投影回词汇空间，帮助我们观察这些层的输出，并理解模型在不同层次上的信息处理和表征。

当我们进一步分析模型的内部机制时，发现注意力头有两个关键操作：Query-Key（QK）电路和Output-Value（OV）电路。分离这些操作有助于我们理解模型如何处理信息，但由于涉及的矩阵仍然很大，分析时需要选择性地关注关键部分。在此基础上，模型中的许多操作实际上是通过复制（power of copy）来实现的，这意味着 Transformer 模型在处理信息时依赖于复制输入数据或中间表示的机制。

同时，分词器在处理文本时可能会将一些术语拆分为多个标记，这种拆分（repairing tokenizer）对模型的处理有重要影响。通过观察 QK 和 OV 矩阵，我们可以分析这些拆分如何影响模型的注意力机制和最终输出。

最后，诱导头（Induction Heads）展示了注意力头的专门化能力。诱导头专注于完成上下文中已见过的模式。如果当前标记已经出现过，诱导头会关注下一个紧接的标记；如果标记是新的，诱导头则会将注意力集中到序列中的第一个标记。这种类型的头帮助模型在处理上下文信息时更好地捕捉和预测模式。

综上所述，从 Residual Stream 的高维表示到注意力头的具体功能，这些内容共同揭示了 Transformer 模型在信息处理和表示方面的复杂机制，帮助我们更深入地理解模型的工作原理和表现。

Activation Patching

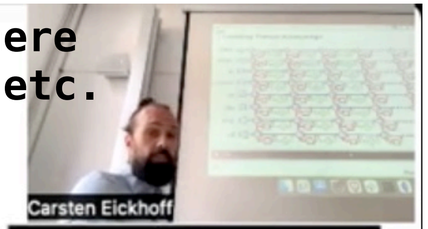
central to mech_interpret

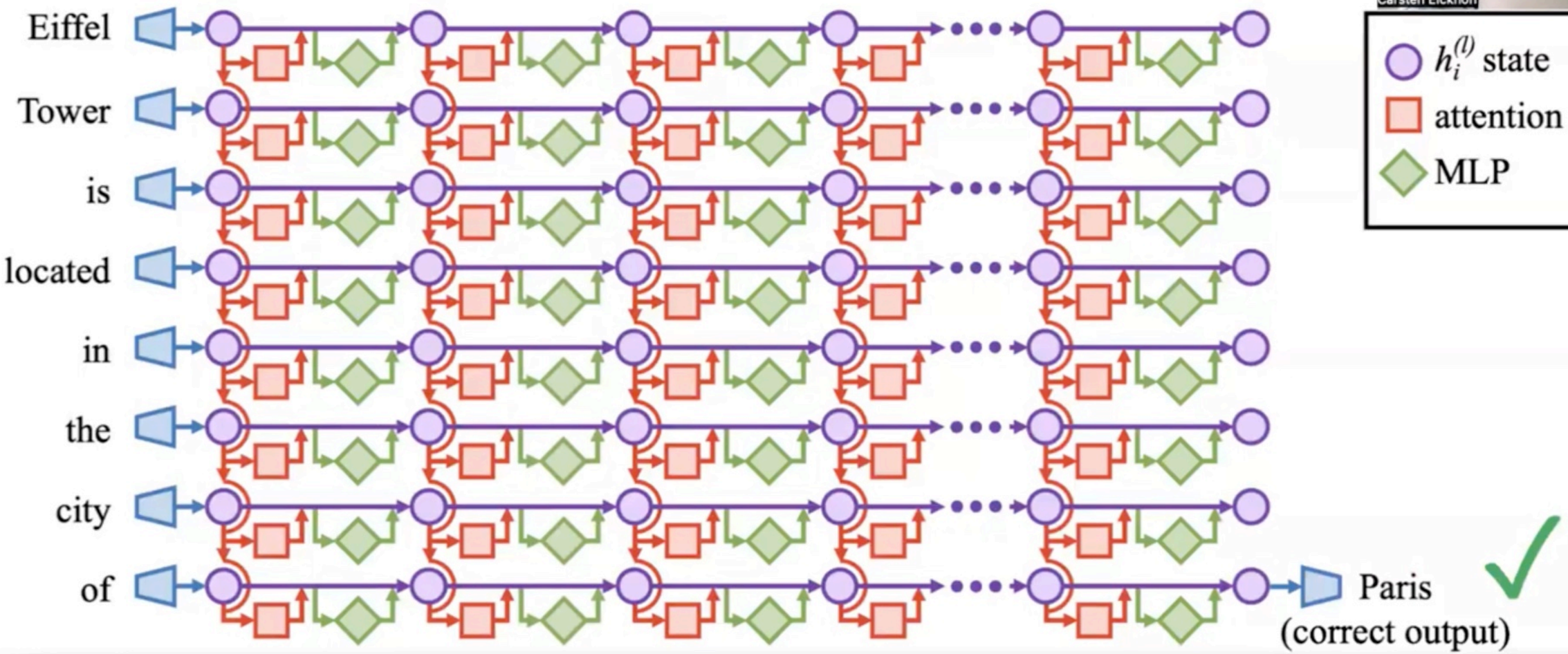
激活 修补

activation: the output value of neurons in each layer after receiving input and passing through the activation function (ReLU, sigmoid, tanh, softmax, etc)

Locating Factual Knowledge

Which part of the model knows that there is such a thing that is Eiffel tower etc. Where is the knowledge?





00:15 -00:22

Meng et al. Locating and Editing Factual Associations in GPT. 2022

Activation Patching

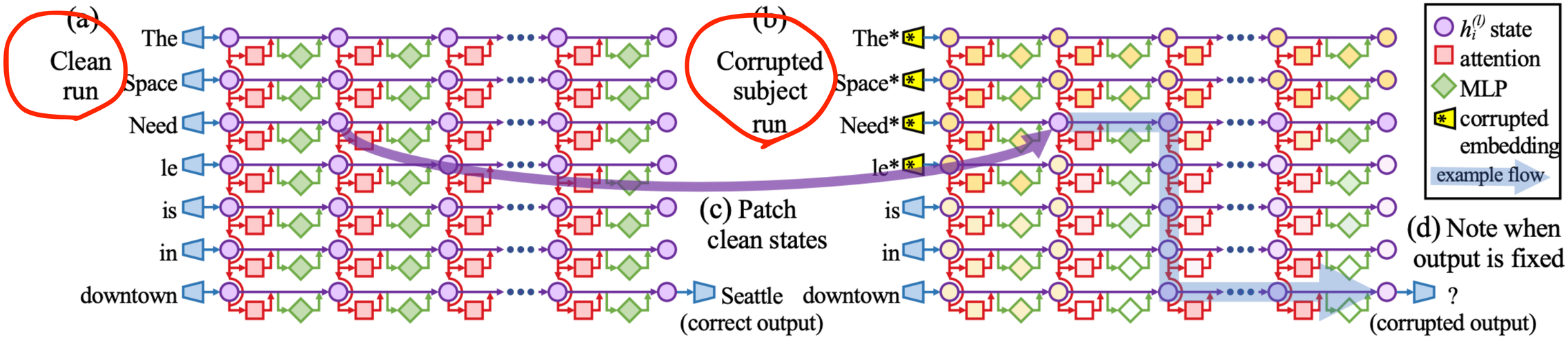
太空针塔
罗马斗兽场

The **Space Needle** is in downtown -> **Seattle**.

The **Colosseum** is in downtown -> **Rome**.

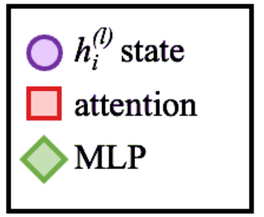
原始句子是“The Space Needle is in downtown”，模型应补充出“Seattle”。

通过修改模型的激活值并观察其对输出的影响，可以探究模型内部哪些部分负责存储和检索关于“Space Needle”的信息。



takes parts of either clean/corrupted one and plug into the other, observe the differences after.

Patch Location

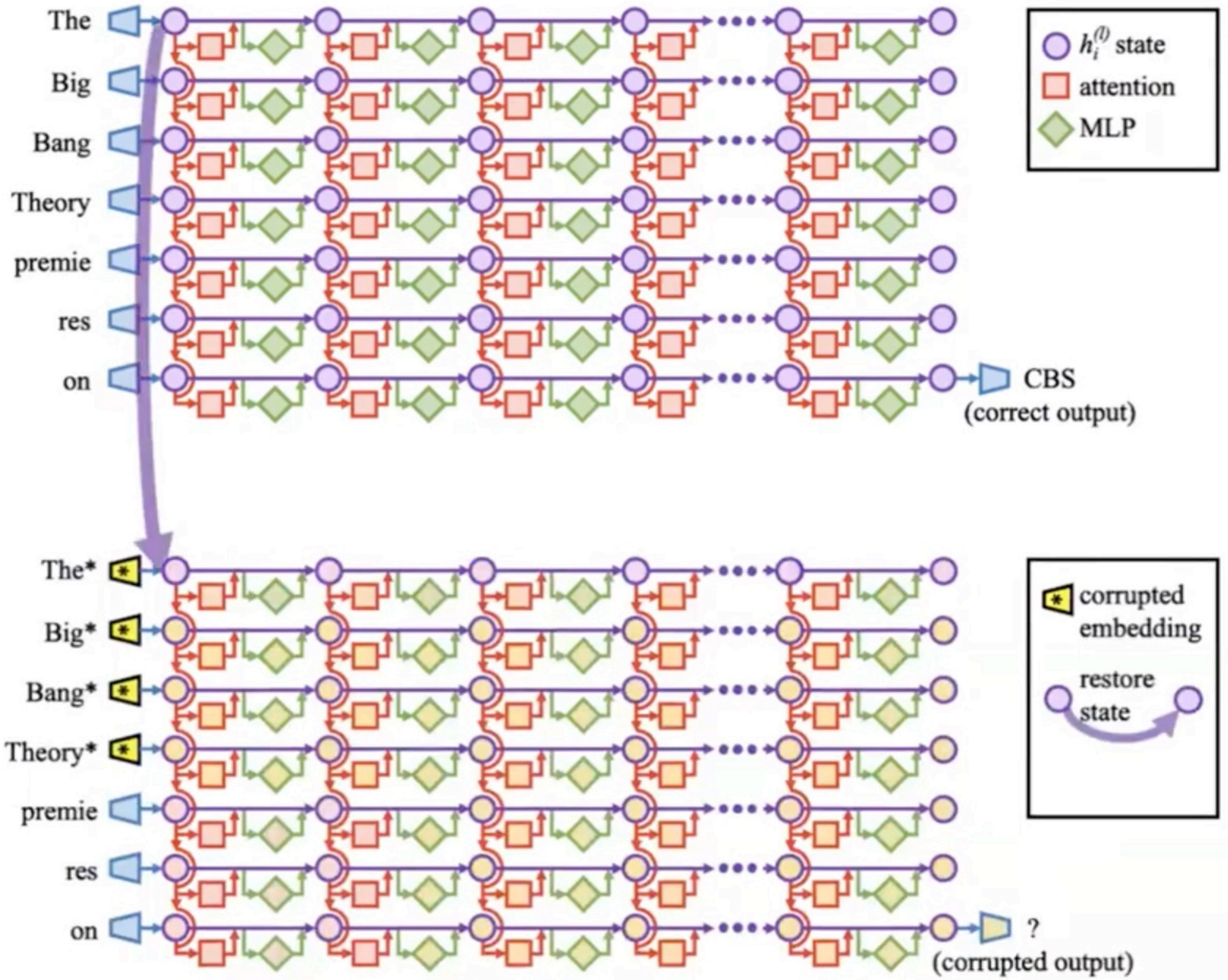


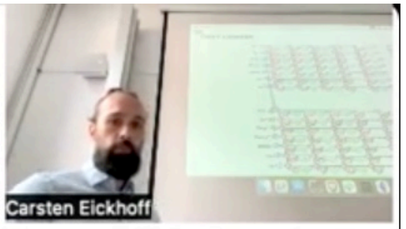
the tricky is to find the right location.

!!! To locate knowledge, test every location and find states with a causal effect.

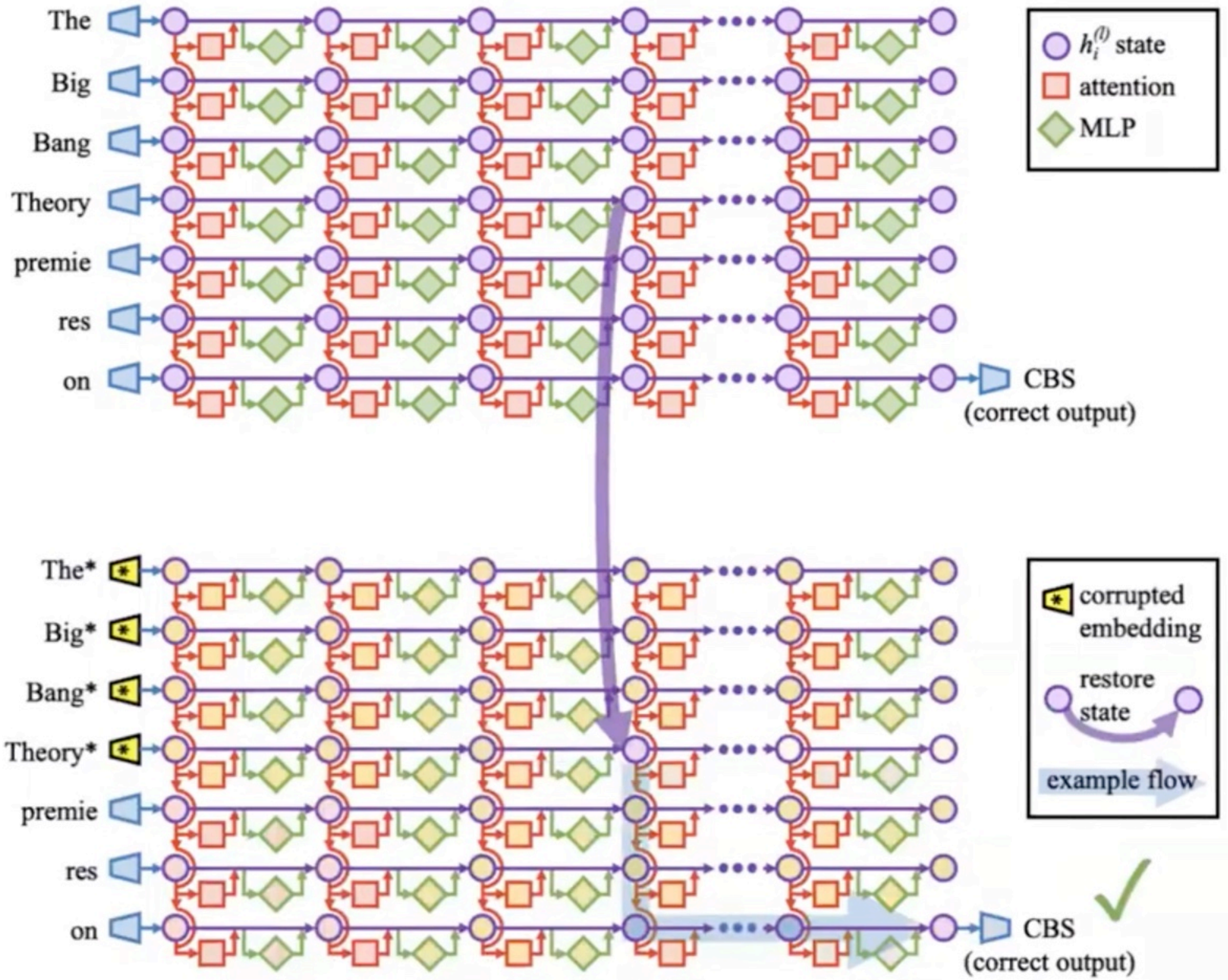
From first location to first location, and so on through all components, everytime to see if there are effect.

Patch Location

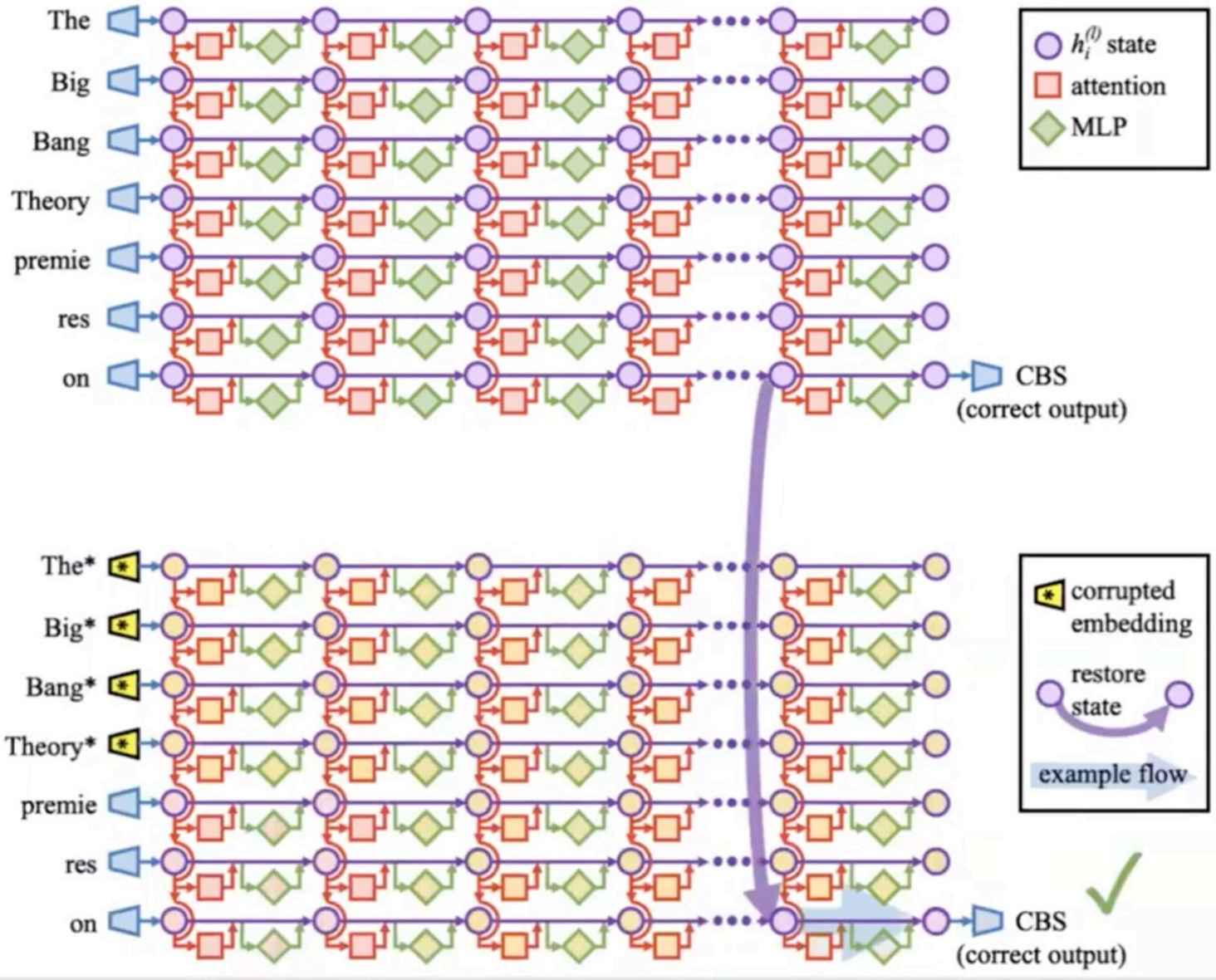




Patch Location



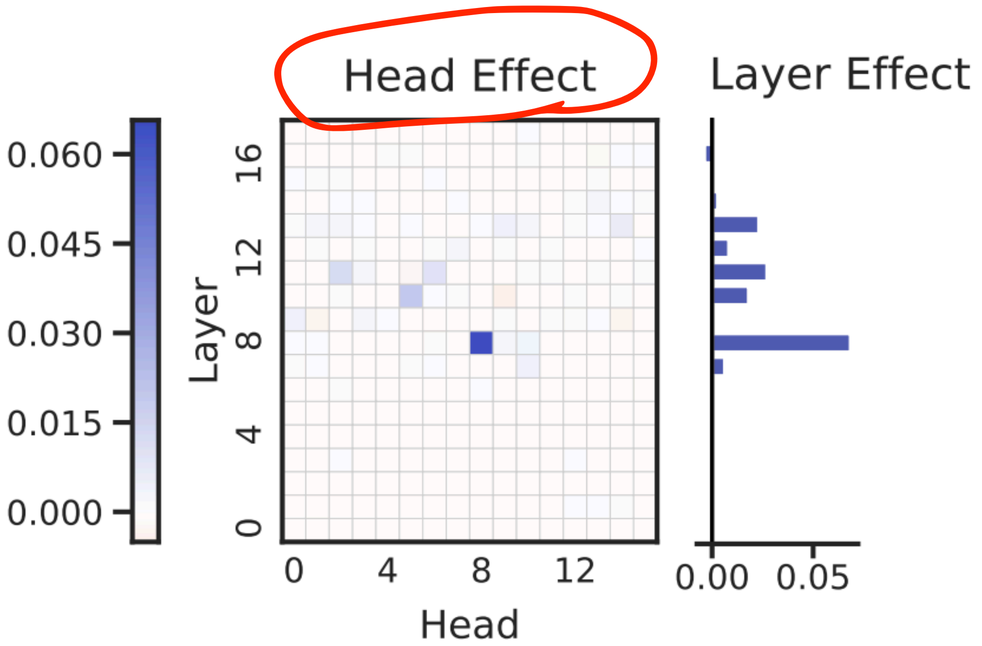
Patch Location



Patch Effects 修补效果

- Track patch effect across the layers and components of the model
- Locate components with strong effect when patched
- E.g., gender bias

The **nurse** entered the room -> **She** …
The **doctor** entered the room-> **He** …



通过修补这些句子的激活值并观察模型的输出变化，
可以分析模型在处理这些句子时是否存在性别偏见
（如默认将护士设定为女性，将医生设定为男性）

Big et al. Causal Mediation Analysis for Interpreting Neural NLP. 2020

Finding an Expressive Metric

how to express the changes,
how to quantify 修补效果

- **Logit Difference** between correct/incorrect answers
- **Raw Class Logits** for the correct answer
- **Log Probability** of correct answer
- **Raw Probability** of correct answer
- **Binary performance metrics** such as accuracy or rank of correct answer

Logit 值（有时也称为 *log-odds*）是指模型在分类任务中生成的原始输出分数，这些分数反映了模型对每个类别的置信程度。*Logit* 值是从模型的输出层得到的，还没有经过 *softmax* 或 *sigmoid* 等激活函数转换成概率。

Desired property:	Continuous (not discrete)	Track model confidence linearly	Measure John vs Mary independent of P(name)	False-positive when "breaking the model"
Logit Difference:	Yes	Yes	Yes	Yes
Logit:	Yes	Yes (typically)	Yes	Maybe
Logprob:	Yes	Not close to p=1	No	No
Probability:	Yes	No	No	No
Binary metrics:	No	No	No	No

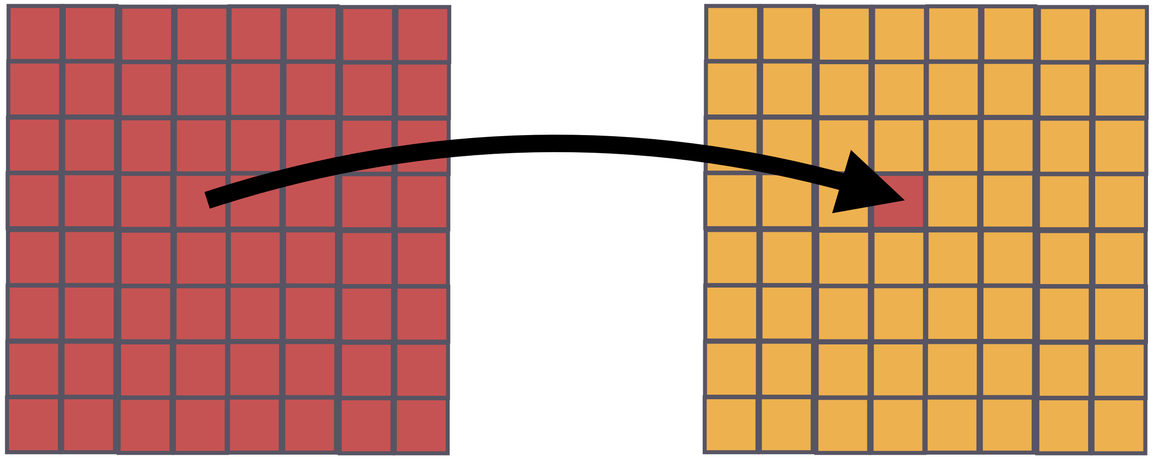
Heimersheim and Nanda. How to use and Interpret Activation Patching. 2024

Patch Direction

两种修补方向：从“损坏”到“清洁”的修补和从“清洁”到“损坏”的修补

- **Corrupted to Clean** patching shows whether shows whether activations are necessary to maintain model behavior (noising).

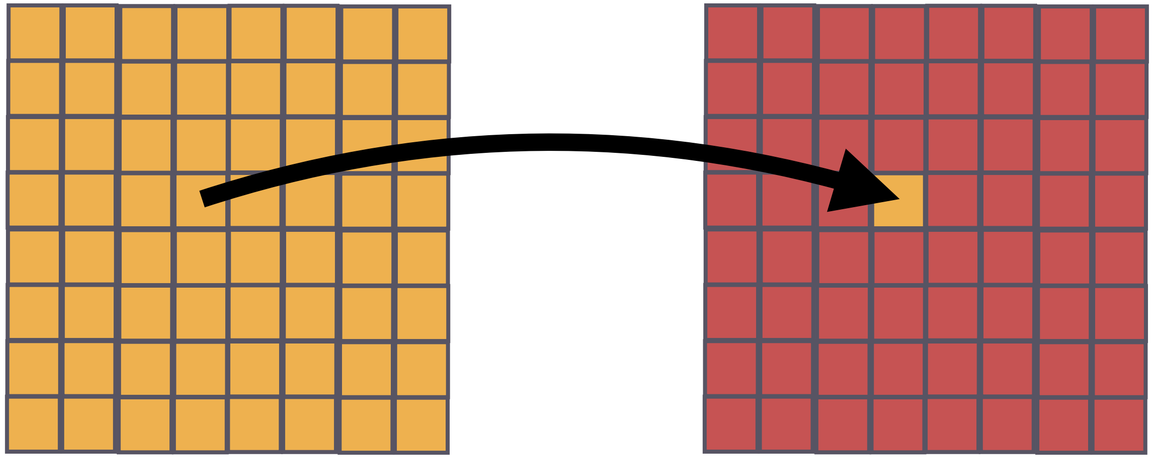
检查模型的激活值是否必要以维持模型行为，方法类似于引入噪声



The **nurse** entered the room -> **She** …
The **doctor** entered the room-> **He** …

- **Clean to Corrupted** patching shows whether activations are sufficient to restore model behavior (denoising).

检查模型的激活值是否足以恢复模型行为。类似去噪。



Heimersheim and Nanda. How to use and Interpret Activation Patching. 2024

在 Activation Patching 这一节中，我们探讨了通过修改神经网络模型的激活值来理解和分析模型内部行为的技术。

首先，通过具体示例如 “The Space Needle is in downtown -> Seattle”和 “The Colosseum is in downtown -> Rome”，我们了解了如何通过修改激活值来定位模型中的事实知识。这些示例展示了激活修补技术如何帮助我们理解模型如何存储和检索特定的事实信息。

接下来，我们探讨了修补效果的跟踪，特别是如何通过观察模型输出的变化来定位在修补过程中产生显著影响的模型组件。这一部分还通过性别偏见的示例展示了激活修补技术在检测模型偏见中的应用，如默认将护士设定为女性、医生设定为男性的倾向。

进一步，我们讨论了选择适当的表达性指标以量化修补效果的重要性。指标如正确/错误答案的 logit 差异、正确答案的原始类 logits、对数概率、原始概率以及二元性能指标（如准确率），帮助研究者全面评估模型在处理各种任务时的表现。

最后，我们介绍了修补方向的概念，通过从“损坏”到“清洁”的修补（Corrupted to Clean Patching）和从“清洁”到“损坏”的修补（Clean to Corrupted Patching），研究者可以分别检查激活值在维持和恢复模型行为中的必要性和充分性。这些方法使我们能够更深入地理解模型的依赖关系和信息处理机制，为改进和优化模型提供了重要的工具和方法。

总体而言，这一节通过详细的技术探讨和实际应用示例，展示了 Activation Patching 技术在理解、分析和改进神经网络模型中的重要作用。

Circuit Analysis

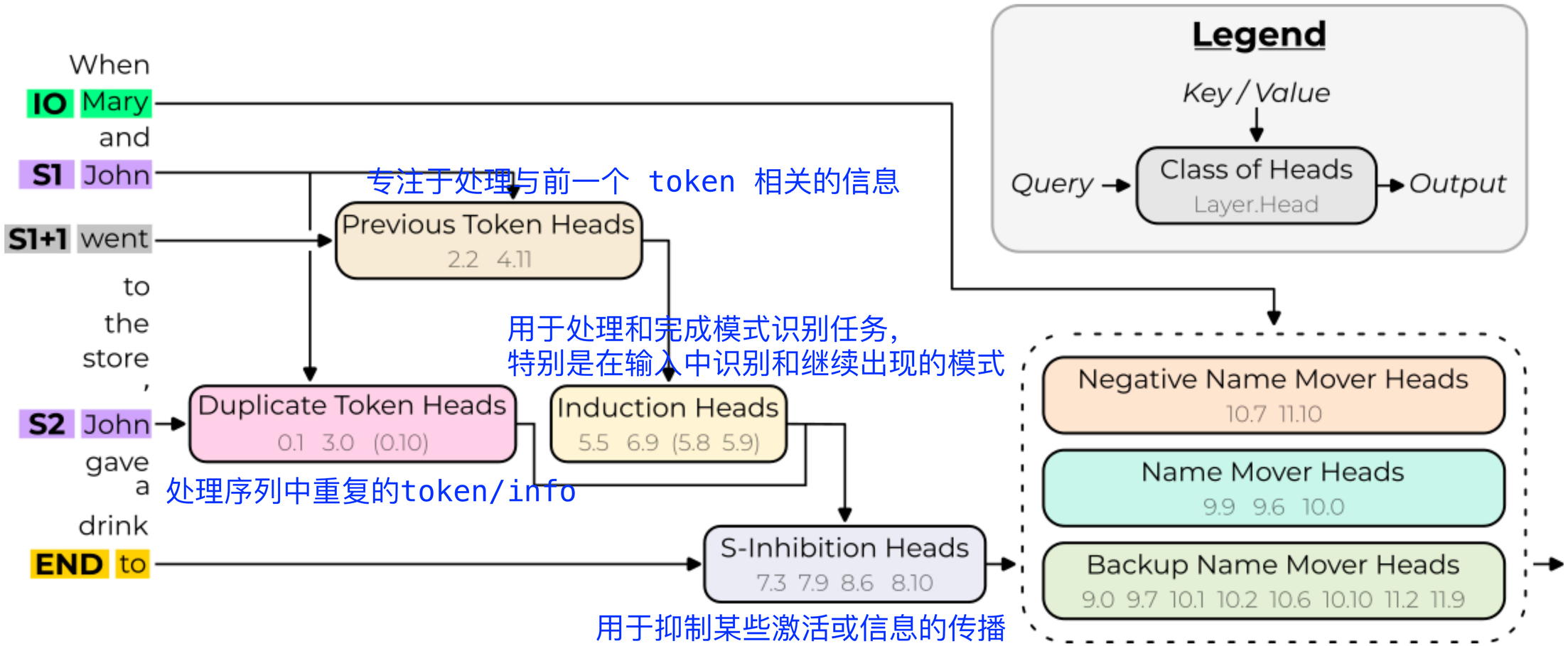
电路分析。

@GPT: 在神经网络研究中用于理解和解释模型内部工作机制的一种方法。

这种方法通过分析网络中的特定子结构（称为“电路”），揭示模型如何处理输入、存储信息和生成输出。

电路分析的目标是将复杂的神经网络行为分解成可解释的组件，类似于在电子电路中分析各个元件的功能和作用。

More Heads…

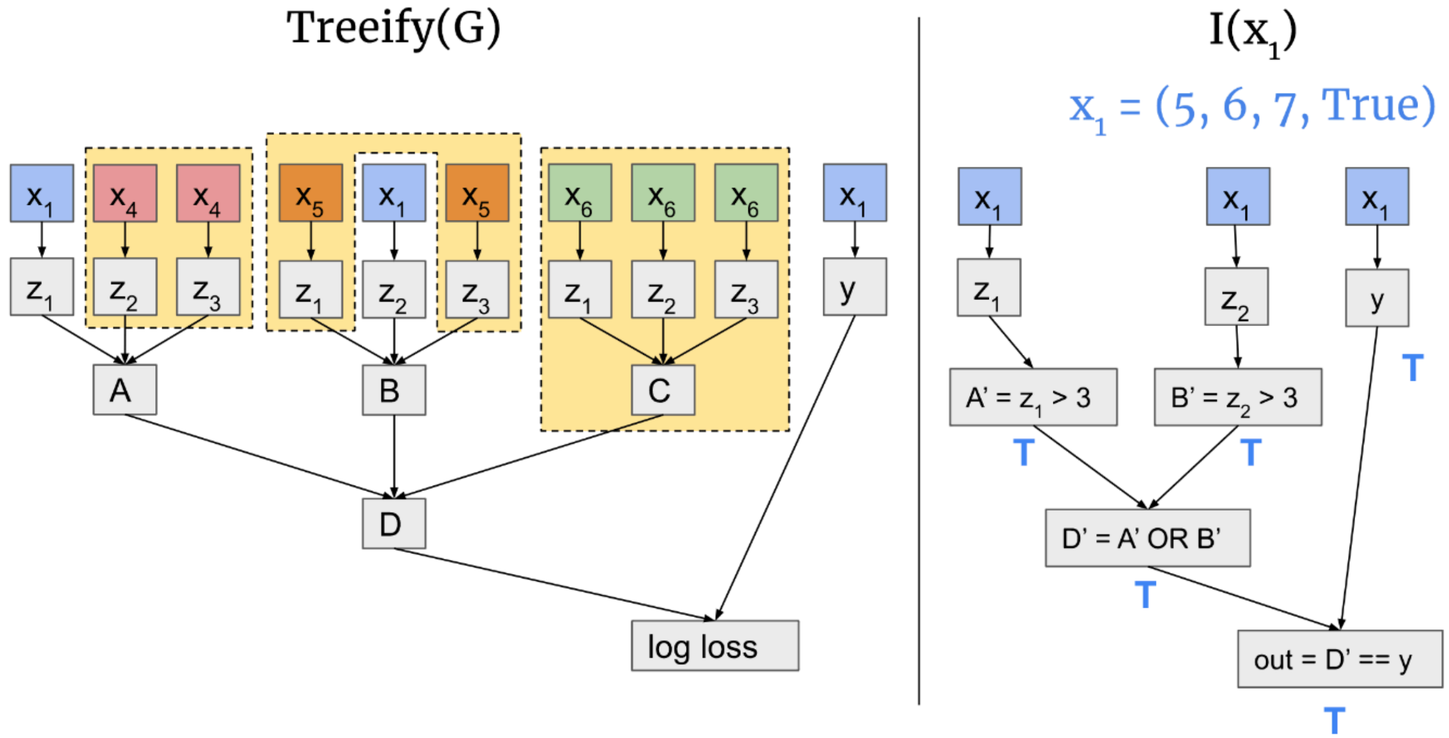


Wang et al. Interpretability in the Wild. 2023

Causal Scrubbing

- Establish hypothesis
  - e.g., via activation patching
- Systematically test hypothesis by recursively find all invariant model components

通过递归地查找模型中所有不变的组件来测试假设



Chan et al. Causal Scrubbing: a method for rigorously testing interpretability hypotheses, 2022

What just happened!?

- Early Decoding
- Residual Stream
- Patching
- Circuit Analysis

What is to come…

- Tutorial **July 5th** (health permitting)
- Next lecture **July 9th**

首先，我们探讨了 Early Decoding，这一技术允许我们在 Transformer 网络的早期层中将高维的嵌入空间投影回词汇空间。通过这种方式，我们能够观察模型在不同层次的早期信息处理和决策情况，类似于“窥视”(probing)模型在处理输入时的初步输出。这种方法帮助我们理解早期层的激活如何影响模型的最终输出，提供了对模型内在信息处理的初步视角。

接着，我们深入了解了 Residual Stream，即在 Transformer 网络中，每一层的输出都会通过残差连接直接传递到下一层。这种结构保证了信息在网络中的持续流动，使得模型能够在多个层次上进行复杂的变换和表达。通过分析残差流，我们可以理解模型如何在不同层次处理和整合信息，以及如何通过不同的头部（如 s-inhibition heads、induction heads 等）来执行具体的任务和功能。理解残差流为我们揭示了模型内部信息流动和处理机制的关键细节。

在此基础上，我们讨论了 Activation Patching，一种技术，通过修改模型的激活值来观察模型行为的变化。这种方法包括两种修补方向：从损坏到清洁的修补，测试激活值的必要性；以及从清洁到损坏的修补，测试激活值的充分性。通过这些操作，我们可以系统性地验证模型中各个组件的作用，并且揭示模型在不同情况下的行为和依赖关系。这为模型的调试和改进提供了强有力的工具。

最后，我们探讨了 Circuit Analysis，一种通过分析模型内部特定的子结构或电路来理解模型工作机制的方法。这种分析方法着重于分离和研究特定的注意力头部（如 Query-Key 电路和 Output-Value 电路），帮助我们理解每个电路的功能和作用。电路分析使我们能够更精确地识别模型在处理信息时的关键组件和机制，揭示模型如何实现复杂的行为和决策。

因此，我们能够从宏观和微观两个层面深入理解神经网络模型的工作原理，从模型的信息处理和流动到特定机制和组件的分析。这种综合的分析方法不仅揭示了模型的内部机制，还为模型的优化和改进提供了坚实的基础。