Stat 9911
Principles of AI: LLMs
Large Language Model Architectures 02

Edgar Dobriban

Department of Statistics and Data Science, the Wharton School, University of Pennsylvania

January 30, 2025

Plan

- We aim to provide insight into transformer architectures.
- Review: (Ferrando et al., 2024).

Table of Contents

How Do LLMs Encode Information?

Attention as Context-Dependent Markov Chain

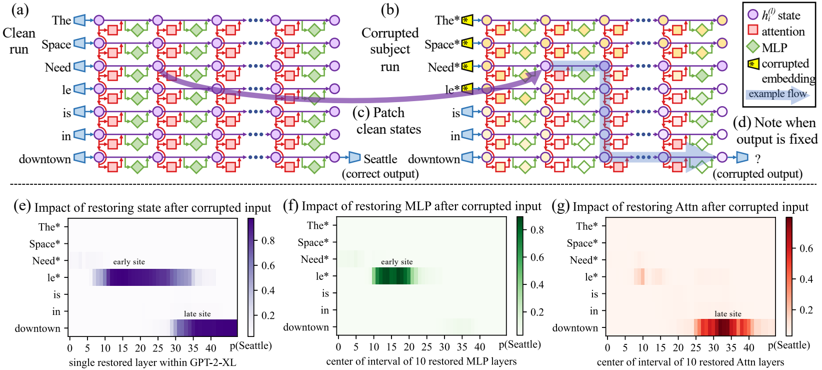
Representational and Computational Abilities of Transformers

How Do LLMs Encode Information?

Some general facts:

- Information is encoded in the weights/parameters of NNs
- Can also extract information by looking at the neurons/activations and representations $e(x)$ for specific inputs x
- Methods for identifying neuron roles: probing, causal interventions.
- In NNs, specialized neurons may represent concepts. In LLMs, a classic example is sentiment neurons (Radford et al., 2017).
- In LLMs, representations of subject tokens act as keys for retrieving facts: after an input text mentions a subject, LMs construct enriched representations of subjects that encode information about those subjects (Li et al., 2021).

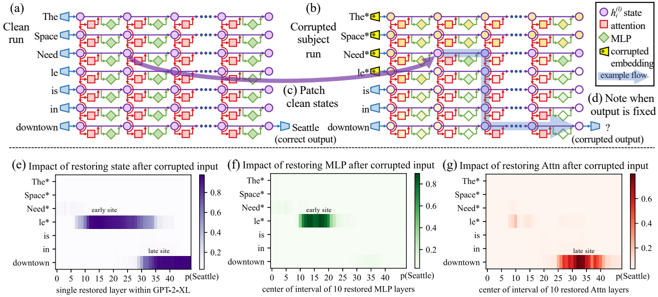
Causal Tracing in LLMs (Meng et al., 2022)



Consider the computational graph as a causal graph, as in causal mediation analysis for NLP (Vig et al., 2020):

1. Run LM on input text (e.g., "The Space Needle is in").
2. Corrupt subject, e.g., "**[i.i.d. Gaussian activations]** is in", and re-run LM.
3. For each neuron group, patch activations from clean run into corrupted run; run LM again to see if it restores original logits.

Causal Tracing in LLMs (Meng et al., 2022)



(a) Clean run
The
Space
Need
le
is
in
downtown
Seattle (correct output)
(b) Corrupted subject run
The*
Space*
Need*
le*
is
in
downtown
?
(corrupted output)
(c) Patch clean states
(d) Note when output is fixed
$h_i^{(l)}$ state
attention
MLP
corrupted embedding
example flow
(e) Impact of restoring state after corrupted input
early site
late site
single restored layer within GPT-2-XL
(f) Impact of restoring MLP after corrupted input
early site
center of interval of 10 restored MLP layers
(g) Impact of restoring Attn after corrupted input
late site
center of interval of 10 restored Attn layers
0
5
10
15
20
25
30
35
40
p(Seattle)
0.2
0.4
0.6
0.8

- "Each midlayer MLP accepts inputs that encode a subject, then produces outputs that recall memorized properties about that subject."
- "Then the summed information is copied to the last token by attention at high layers."

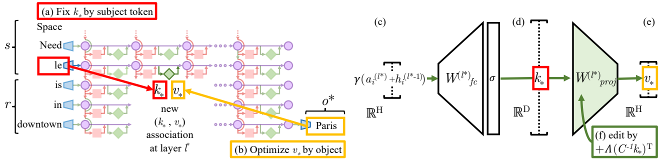
Information Editing in LLMs (Meng et al., 2022)

- Claim: The two-layer FFN acts as a linear associative memory (Kohonen, 1972; Anderson, 1972). [refines Geva et al. (2020), who view individual neurons as memories]
- Consider 2L-NN $e' = W_{\text{proj}}\sigma(W_1 e)$. Can view $k = \sigma(W_1 e)$ as a key and $v = W_{\text{proj}} k$ as a value it retrieves.
- For any given collection of key-value pairs k_i, v_i arranged in matrix form in $(\tilde{K}, \tilde{V})$, find $W = W_{\text{proj}}$ to approximately operate as a key-value store, solving:

  $$\min_W \|W\tilde{K} - \tilde{V}\|_F,$$

  obtaining $\tilde{W} = \tilde{V}\tilde{K}^+$.
- Insert a key-value pair, e.g., $k^* =$ Space Needle, $v^* =$ Paris, with minimal weight perturbation: Solve $\min_W \|W\tilde{K} - \tilde{V}\|_F$, subject to $Wk^* = v^*$.

Rank-One Model Editing (Meng et al., 2022, ROME)



- The solution is $W^* = \tilde{W} + \Lambda \left(C^{-1} k^*\right)^\top$, where

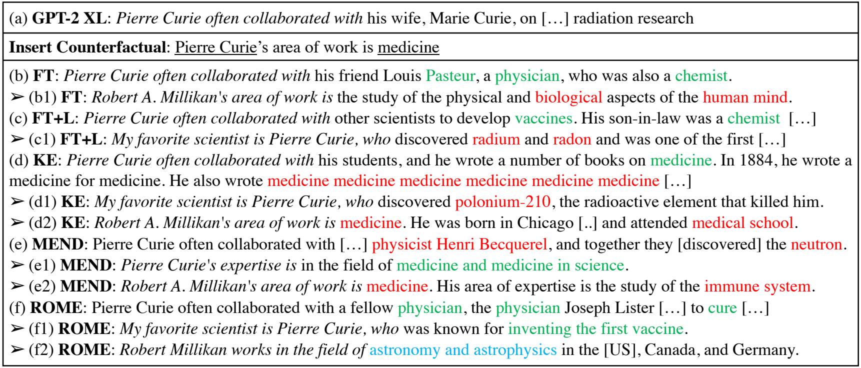
$$C = \tilde{K}\tilde{K}^\top \quad \text{and} \quad \Lambda = (v^* - \tilde{W}k^*)/k^{*,\top} C^{-1} k^*.$$

- Estimate C based on a data sample from the distribution that we expect k^* to follow.
- To insert a specific memory, use k^* of original text, e.g. "Space Needle" (with some random contexts).
- To find v^* to insert, solve:

$$\max_{v} P(o^* \mid \text{prefix} + x, v(x) \mapsto v),$$

where x are the tokens for k^*; and o^* is target output, e.g., "Paris".

Example



(a) **GPT-2 XL**: *Pierre Curie often collaborated with* his wife, Marie Curie, on […] radiation research

Insert Counterfactual: Pierre Curie's area of work is medicine

(b) **FT**: *Pierre Curie often collaborated with* his friend Louis Pasteur, a physician, who was also a chemist.

➢ (b1) **FT**: *Robert A. Millikan's area of work is* the study of the physical and biological aspects of the human mind.

(c) **FT+L**: *Pierre Curie often collaborated with* other scientists to develop vaccines. His son-in-law was a chemist […]

➢ (c1) **FT+L**: *My favorite scientist is Pierre Curie, who* discovered radium and radon and was one of the first […]

(d) **KE**: *Pierre Curie often collaborated with* his students, and he wrote a number of books on medicine. In 1884, he wrote a medicine for medicine. He also wrote medicine medicine medicine medicine medicine medicine […]

➢ (d1) **KE**: *My favorite scientist is Pierre Curie, who* discovered polonium-210, the radioactive element that killed him.

➢ (d2) **KE**: *Robert A. Millikan's area of work is* medicine. He was born in Chicago [..] and attended medical school.

(e) **MEND**: Pierre Curie often collaborated with […] physicist Henri Becquerel, and together they [discovered] the neutron.

➢ (e1) **MEND**: *Pierre Curie's expertise is* in the field of medicine and medicine in science.

➢ (e2) **MEND**: *Robert A. Millikan's area of work is* medicine. His area of expertise is the study of the immune system.

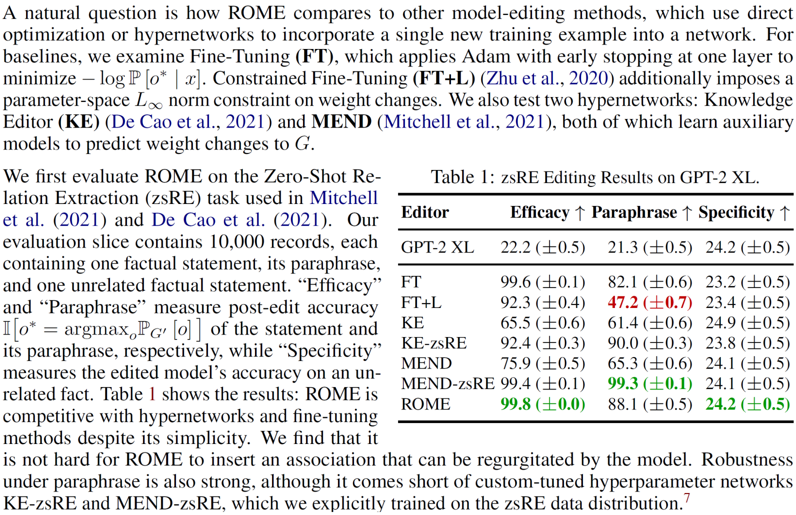
(f) **ROME**: Pierre Curie often collaborated with a fellow physician, the physician Joseph Lister […] to cure […]

➢ (f1) **ROME**: *My favorite scientist is Pierre Curie, who* was known for inventing the first vaccine.

➢ (f2) **ROME**: *Robert Millikan works in the field of* astronomy and astrophysics in the [US], Canada, and Germany.

Figure: Examples of information editing from Meng et al. (2022); comparing several baselines and their proposed ROME method. "Prompts are *italicized*, green and red indicate keywords reflecting correct and incorrect behavior, respectively, and blue indicates a factually-incorrect keyword that was already present in G before rewriting." Show paraphrases and specificity.

Performance of ROME (Meng et al., 2022)



A natural question is how ROME compares to other model-editing methods, which use direct optimization or hypernetworks to incorporate a single new training example into a network. For baselines, we examine Fine-Tuning **(FT)**, which applies Adam with early stopping at one layer to minimize $-\log \mathbb{P}\left[o^* \mid x\right]$. Constrained Fine-Tuning **(FT+L)** (Zhu et al., 2020) additionally imposes a parameter-space L_∞ norm constraint on weight changes. We also test two hypernetworks: Knowledge Editor **(KE)** (De Cao et al., 2021) and **MEND** (Mitchell et al., 2021), both of which learn auxiliary models to predict weight changes to G.

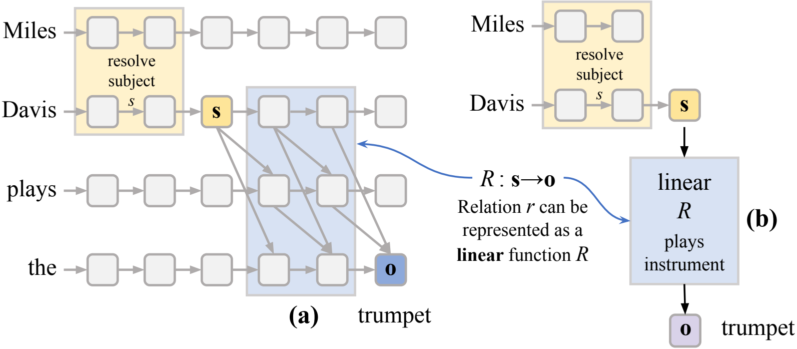
We first evaluate ROME on the Zero-Shot Relation Extraction (zsRE) task used in Mitchell et al. (2021) and De Cao et al. (2021). Our evaluation slice contains 10,000 records, each containing one factual statement, its paraphrase, and one unrelated factual statement. "Efficacy" and "Paraphrase" measure post-edit accuracy $\mathbb{I}\big[o^* = \operatorname{argmax}_o \mathbb{P}_{G'}\left[o\right]\big]$ of the statement and its paraphrase, respectively, while "Specificity" measures the edited model's accuracy on an unrelated fact. Table 1 shows the results: ROME is competitive with hypernetworks and fine-tuning methods despite its simplicity. We find that it is not hard for ROME to insert an association that can be regurgitated by the model. Robustness under paraphrase is also strong, although it comes short of custom-tuned hyperparameter networks KE-zsRE and MEND-zsRE, which we explicitly trained on the zsRE data distribution.[7]

Table 1: zsRE Editing Results on GPT-2 XL.

Editor	**Efficacy ↑**	**Paraphrase ↑**	**Specificity ↑**
GPT-2 XL	22.2 (±0.5)	21.3 (±0.5)	24.2 (±0.5)
FT	99.6 (±0.1)	82.1 (±0.6)	23.2 (±0.5)
FT+L	92.3 (±0.4)	**47.2 (±0.7)**	23.4 (±0.5)
KE	65.5 (±0.6)	61.4 (±0.6)	24.9 (±0.5)
KE-zsRE	92.4 (±0.3)	90.0 (±0.3)	23.8 (±0.5)
MEND	75.9 (±0.5)	65.3 (±0.6)	24.1 (±0.5)
MEND-zsRE	99.4 (±0.1)	**99.3 (±0.1)**	24.1 (±0.5)
ROME	**99.8 (±0.0)**	88.1 (±0.5)	**24.2 (±0.5)**

Representing Relations in LLMs

- Consider relations, e.g., (Jimi Hendrix, plays the, guitar), (Miles Davis, plays the, trumpet).
- Some of them are approximately affine maps $s \to o \approx \partial o / \partial s \cdot (s - s_0) + o(s_0)$, where s is a sufficiently contextualized representation of the input, and o are the output logits (Hernandez et al., 2024).



Relation *r* can be represented as a **linear** function *R*

Representing Relations in LLMs (ctd)

- Why??
  - Linear maps are the simplest non-trivial class, all smooth maps are locally approx. linear
  - Prior work on knowledge graph embeddings e.g., Yang et al. (2014), models relations as $\|o'Ms\|$ being small for some M. This is zero iff $o = M^{\perp}s + v$, any $v \perp M \cdot \mathrm{span}(s)$, $M^{\perp}$ an ortho. complement of M
  - Prior work has shown that "linear shortcuts" are enough for accurate decoding from early transformer layers (Din et al., 2024)

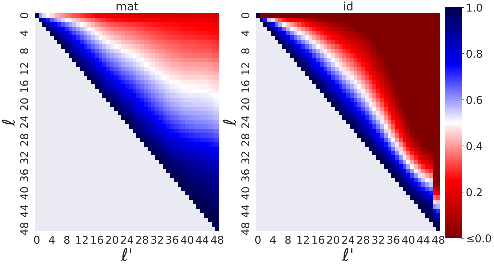


Figure: R^2 between activations transformed from layer ℓ to ℓ', in GPT-2. Left: best linear transform; Right: no transform.

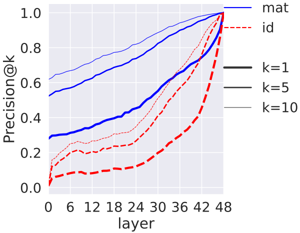


Figure: Accuracy of linear decoding from a given layer in GPT-2.

Representing Relations in LLMs (ctd)

- Challenges handled in Hernandez et al. (2024): Due to LayerNorm, scale is not propagated. More accurate to approximate $s \to o \approx \beta \partial o / \partial s \cdot (s - s_0) + o(s_0)$, for some $\beta > 0$

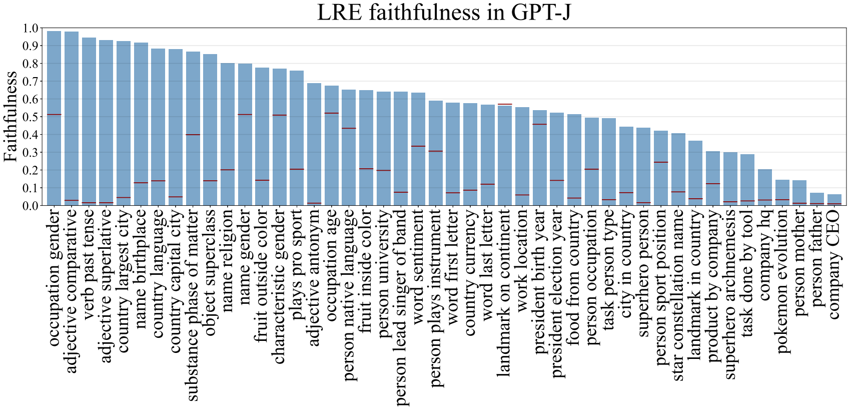


Figure: Accuracy of linear relational decoding across various categories.

Mechanistic Interpretability Perspective

- Mechanistic interpretability is an area that aims to understand the working mechanisms of NNs and LLMs; see work by Anthropic
- For factual recall, Nanda et al. (2023) follow a similar path to what we have seen, with a few key differences
  - Their terminology is based on circuits: sparse sub-networks obtained by pruning most weights in a NN

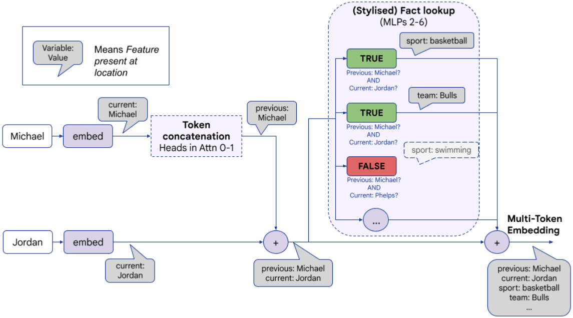


Figure: Source

  - In experiments, prune to the circuit and observe behavior

Table of Contents

How Do LLMs Encode Information?

Attention as Context-Dependent Markov Chain

Representational and Computational Abilities of Transformers

Attention as Context-Dependent Markov Chain (Ildiz et al., 2024)

- View autoregressive generation as a Markov Chain.
- Predictions depend on token frequencies in the context.
- Frequent tokens can dominate the distribution.

Attention as a Context-Dependent Markov Chain (Ildiz et al., 2024)

- T tokens, $|V|$ vocab size.
- Prompt:

$$x = (x_1, x_2, \ldots, x_T), \quad x_i \in [|V|]$$

- Matrix encoding:

$$E = (e_{x_1}, e_{x_2}, \ldots, e_{x_T})^\top, \quad E \in \mathbb{R}^{T \times |V|}$$

where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$ is the i-th unit vector in $\mathbb{R}^{|V|}$.

Calculating the Next Token Probability

Next token probability:

$$
\begin{aligned}
f_W(E) &= E^\top S\big(EWe_{x_T}\big) = E^\top S(EW_{:,x_T}) = E^\top S(Ev) \\
&= E^\top \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{x_T} \end{bmatrix} = \begin{bmatrix} \mathrm{Num}(1) \cdot s_1 \\ \mathrm{Num}(2) \cdot s_2 \\ \vdots \\ \mathrm{Num}(|V|) \cdot s_{|V|} \end{bmatrix}
\end{aligned}
$$

where $v := W_{:,x_T}$ and

1. $$s_i = \frac{\exp(v_i)}{\sum_{j \in x} \exp(v_j)} = \frac{\exp(v_i)}{\sum_{k \in [|V|]} \mathrm{Num}(k) \cdot \exp(v_k)}.$$
2. $\mathrm{Num}(k)$ is the number of times k occurs in x

Interpretation and Implications

$$f_W(E) = \begin{bmatrix} \mathrm{Num}(1) \cdot s_1 \\ \vdots \\ \mathrm{Num}(|V|) \cdot s_{|V|} \end{bmatrix}$$

- W (i.e., v and s) corresponds to the architecture.
- For a fixed W, f_W's predictions can be viewed as s weighted by the frequencies of each token in the context.
- Autoregressive generation can be viewed as a context-dependent Markov Chain: v is fixed, but the frequencies $\mathrm{Num}(k)$ make it depend on the context
- Implication: Frequent tokens (even if occurring by chance) may dominate the distribution; relevant for understanding repetitive sampling.

Table of Contents

How Do LLMs Encode Information?

Attention as Context-Dependent Markov Chain

Representational and Computational Abilities of Transformers

Representational and Computational Abilities of Transformers

- Discussed in Sanford et al. (2024b,a), etc.
- Informally, view transformers as a "general-purpose differentiable computer" (Karpathy):
  - expressive, general message-passing-like architecture can express many algorithms
  - optimizable: residual connections, layer normalizations, and softmax attention remove flat tails
  - highly parallel, wide/shallow compute graph allows efficient use of GPUs

The RASP Language (Weiss et al., 2021)

- RASP (Restricted Access Sequence Processing) is a programming language that describes operations similar to those performed by transformers.
- Operates on strings/sequences/lists (`"hi"`) with indices (`[0,1]`). Sequence operator (s-op): function with input string, output equal-length string. (analogous to FF layer)
- Built-ins:
  - identity, aka `tokens`: `tokens("hi")="hi"`
  - `indices("hi")=[0,1]`
  - `length("hi")=[2,2]`
- Elementwise combination of s-ops:
  - `(indices+1)("hi")=[1,2]`.
  - `((indices+1)==length)("hi")=[F,T]`.
- Ternary operator:
  - `(tokens if (indices%2==0) else "-")("hello")="h-l-o"`.

Select Function in RASP

Select and Aggregate operations combine information from different sequence positions. (analogous to attention layer)

Selection operation: `select`

- Takes two s-ops `k` and `q` (representing *keys* and *queries*), and a comparison operator $\circ$.
- Returns a *selector* sel$(\cdot, \cdot, \circ)$ composed with `k` and `q`.
- Computes a selection matrix for each key-query pair (k, q) based on the comparison.

Example:

- `a=select(indices,indices,<)` is a selector.
- `a("hey")=[[F,F,F],[T,F,F],[T,T,F]]`. Or,

$$\begin{bmatrix} F & F & F \\ T & F & F \\ T & T & F \end{bmatrix}$$

Aggregate Function in RASP

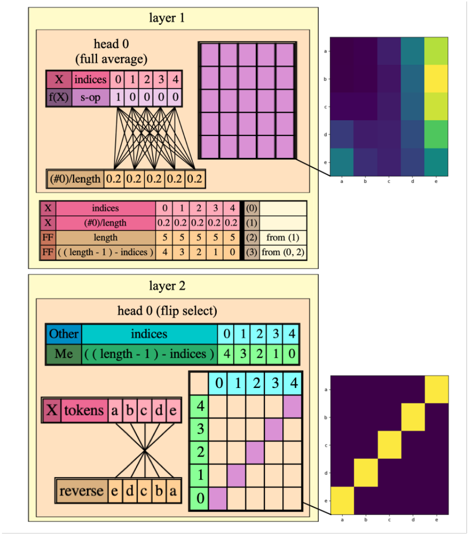
Aggregation operation: `aggregate`

- Takes one selector and one s-op.
- For each row of the selector matrix, averages the corresponding values in the s-op.
- **Example:**
  - $$agg(\begin{bmatrix} T & F & F \\ T & T & F \\ T & T & T \end{bmatrix}, [10, 20, 30]) = [10, 15, 20]$$
  - `aggregate(a,indices+1)("hey")=[0,1,1.5]`.
- For positions with no selected values, a default output value is returned (e.g., 0)

Example: Reversing a String

RASP code and compilation to a transformer, along with learned attention heatmaps.

```
opp_index = length - indices - 1;
flip = select(indices, opp_index, ==);
reverse = aggregate(flip, tokens);
```



X	indices	0	1	2	3	4	(0)	
X	(#0)/length	0.2	0.2	0.2	0.2	0.2	(1)	
FF	length	5	5	5	5	5	(2)	from (1)
FF	((length - 1) - indices)	4	3	2	1	0	(3)	from (0, 2)

Example: Double histogram

Double histogram: for each input token, how many unique input tokens have the same frequency?

```
same_tok = select(tokens, tokens, ==);
hist = selector_width(
            same_tok,
            assume_bos = True);
first = not has_prev(tokens);
same_count = select(hist, hist, ==);
same_count_reprs = same_count and
    select(first, True, ==);
hist2 = selector_width(
            same_count_reprs,
            assume_bos = True);
```

`hist2("§aaabbccdef")=[§,1,1,1,2,2,2,2,3,3,3]`

(internal to has_prev)

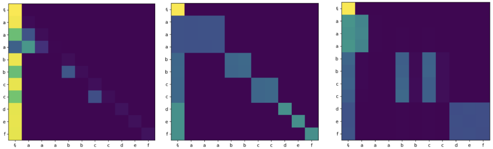
			0	1	2	3	4	5	6	7	8	9	10
			§	a	a	a	b	b	c	c	d	e	f
0	§	0	■										
1	a	0	■										
2	a	0	■	■									
3	a	0	■	■	■								
4	b	0	■										
5	b	0	■				■						
6	c	0	■										
7	c	0	■						■				
8	d	0	■										
9	e	0	■										
10	f	0	■										

same_tok

		0	1	2	3	4	5	6	7	8	9	10
		§	a	a	a	b	b	c	c	d	e	f
§	0	■										
a	0	■	■	■	■							
a	0	■	■	■	■							
a	0	■	■	■	■							
b	0	■				■	■					
b	0	■				■	■					
c	0	■						■	■			
c	0	■						■	■			
d	0	■								■		
e	0	■									■	
f	0	■										■

same_count_reprs

			0	1	2	3	4	5	6	7	8	9	10
			0.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0	1.0	1.0	1.0
			F	F	T	T	F	T	F	T	F	F	F
0.0	F	0	■										
3.0	F	0	■	■									
3.0	F	0	■	■									
3.0	F	0	■	■									
2.0	F	0	■				■		■				
2.0	F	0	■				■		■				
2.0	F	0	■				■		■				
2.0	F	0	■				■		■				
1.0	F	0	■								■	■	■
1.0	F	0	■								■	■	■
1.0	F	0	■								■	■	■



- `selector_width` : input a selector; output s-op that computes, for each output position, the number of input values which the selector chooses.
- E.g., `same_token=select(tokens,tokens,==)` `selector_width(same_token)("hello")=[1,1,2,2,1].`
- `assume_bos` : is there a special beginning of string token?

Example algorithms in RASP

- Using RASP operations, one can implement a variety of algorithms, such as:
  - Sorting: A) Lexicographic; B) Unique input tokens in order of decreasing frequency
  - Recognize formal languages, e.g., sequences of correctly balanced parentheses
- For more info, see Weiss et al. (2021), github repo, e.g., cheat sheet

References

J. A. Anderson. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220, 1972.

A. Y. Din, T. Karidi, L. Choshen, and M. Geva. Jump to conclusions: Short-cutting transformers with linear transformations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9615–9625, 2024.

J. Ferrando, G. Sarti, A. Bisazza, and M. R. Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

E. Hernandez, A. S. Sharma, T. Haklay, K. Meng, M. Wattenberg, J. Andreas, Y. Belinkov, and D. Bau. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=w7LU2s14kE.

M. E. Ildiz, Y. Huang, Y. Li, A. S. Rawat, and S. Oymak. From self-attention to markov models: Unveiling the dynamics of generative transformers. *arXiv preprint arXiv:2402.13512*, 2024.

T. Kohonen. Correlation matrix memories. *IEEE transactions on computers*, 100(4): 353–359, 1972.

References

B. Z. Li, M. Nye, and J. Andreas. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, 2021.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35: 17359–17372, 2022.

N. Nanda, S. Rajamanoharan, J. Kramár, and R. Shah. Fact-finding: Attempting to reverse-engineer factual recall, 2023. URL https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall.

A. Radford, R. Jozefowicz, and I. Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

C. Sanford, D. Hsu, and M. Telgarsky. Transformers, parallel computation, and logarithmic depth. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=QCZabhKQhB.

C. Sanford, D. J. Hsu, and M. Telgarsky. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36, 2024b.

J. Vig, S. Gehrmann, Y. Belinkov, S. Qian, D. Nevo, Y. Singer, and S. Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

References

G. Weiss, Y. Goldberg, and E. Yahav. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR, 2021.

B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.