Mechanistic Interpretability

Fundamentals
WT 2024/25

Frederick Riemenschneider

24.10.2024

Recap


attention
transformers
"the transformer"
sequence-to-sequence learning
BERT
encoder
decoder
GPT
T5
LSTM

Encoder-decoder Architecture

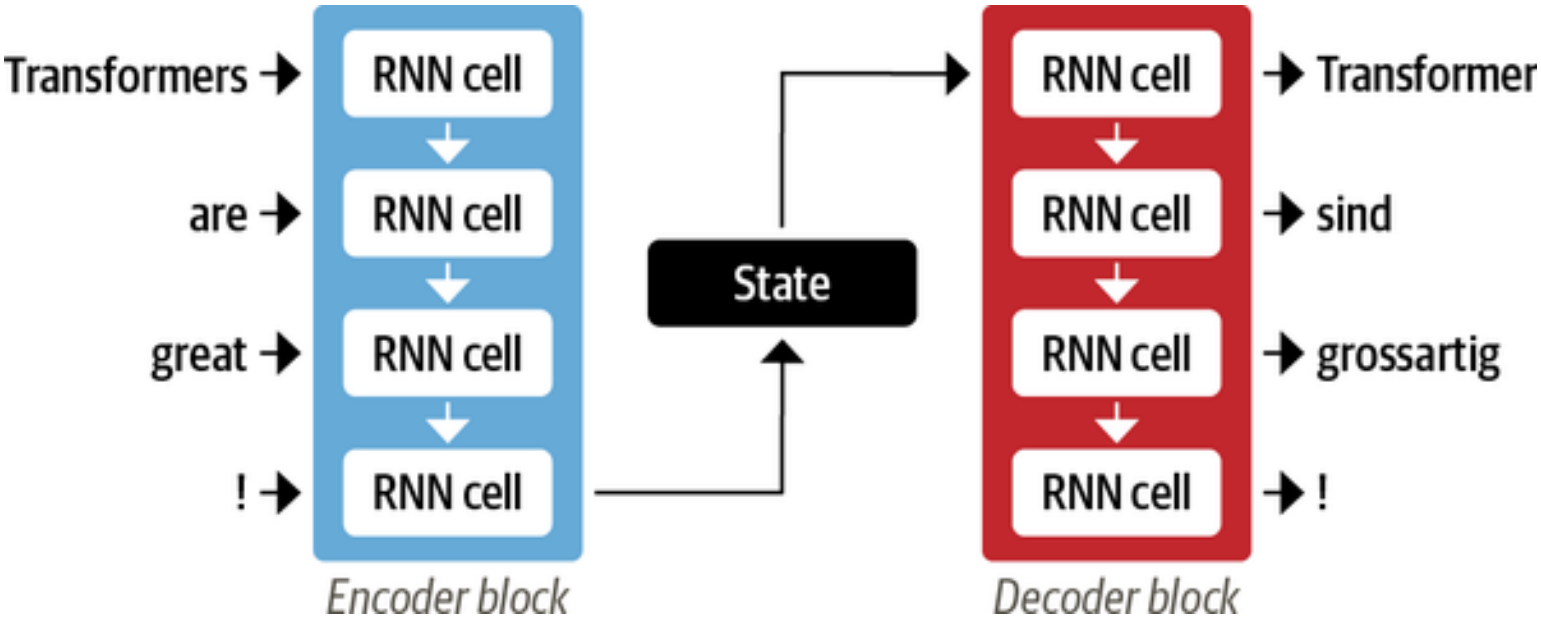


Figure 1: Tunstall et al. 2022, p. 4.

Attention

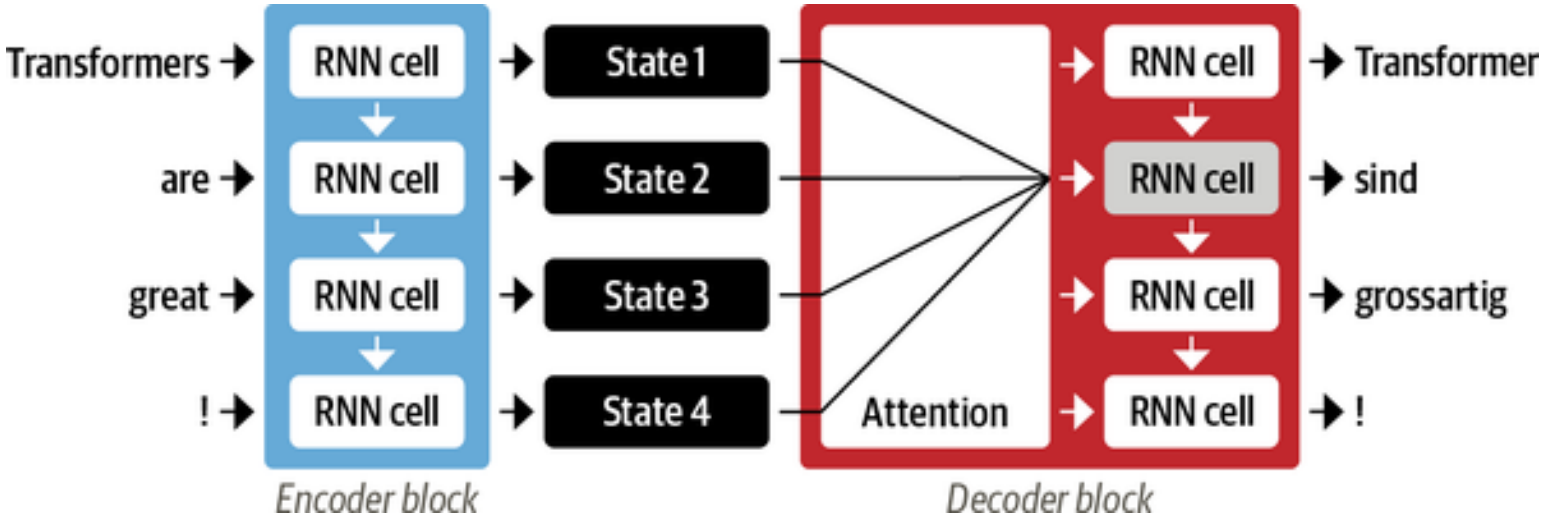


Figure 2: Tunstall et al. 2022, p. 5.

Attention

- context as a weighted average:

$$\boldsymbol{c}_i = \sum_j^{T^e} \alpha_{i,j} \boldsymbol{h}_j^e$$

- normalization via softmax:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_k^{T^e} \exp(e_{i,k})}$$

- importance of $\boldsymbol{h}_j^e$ for $\boldsymbol{h}_{i-1}^d$:

$$e_{i,j} = a(\boldsymbol{h}_{i-1}^d, \boldsymbol{h}_j^e)$$

Self-Attention

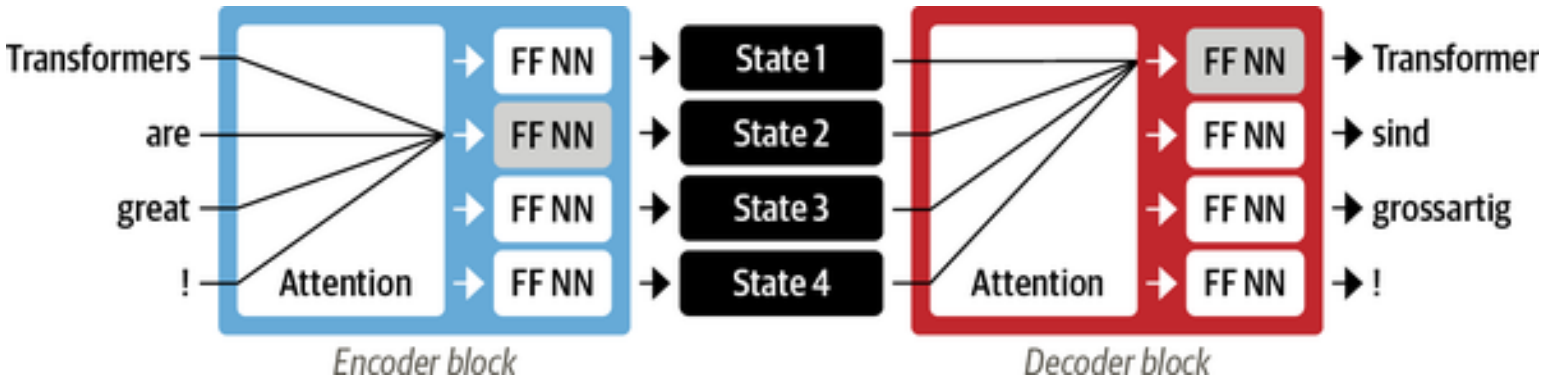


Figure 3: Tunstall et al. 2022, p. 6.

Attention Masks

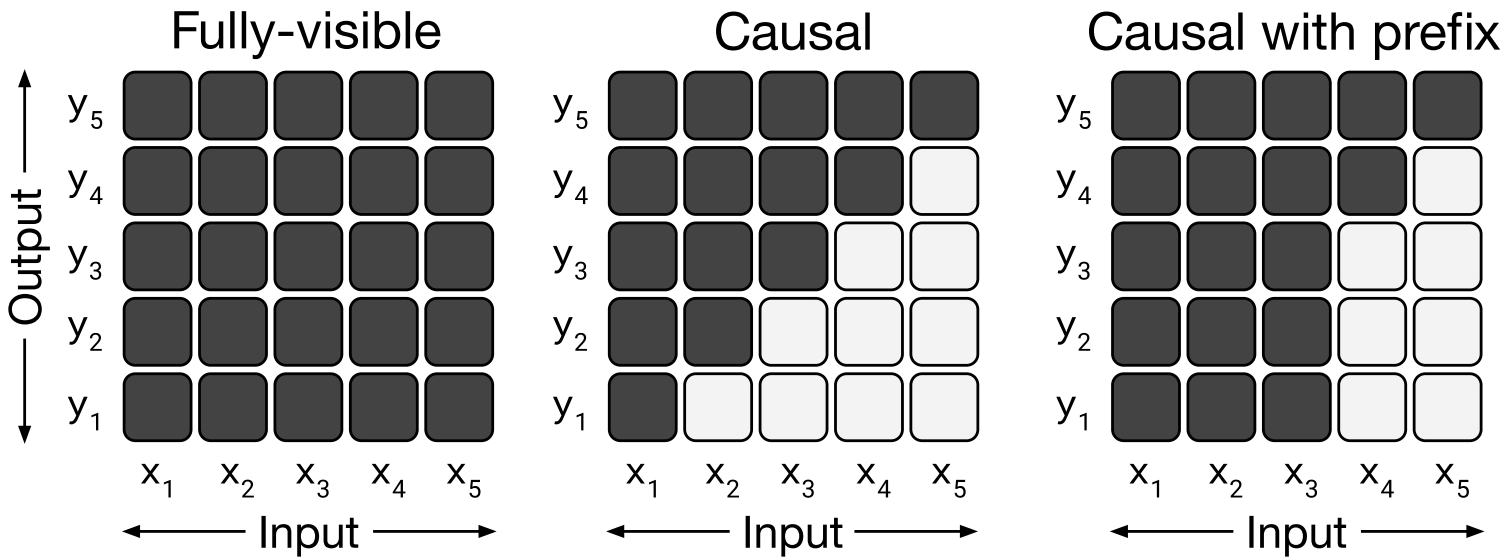


Figure 4: Raffel et al. 2020.

Which pre-training objectives do you know?

A Mathematical Framework for Transformer Circuits

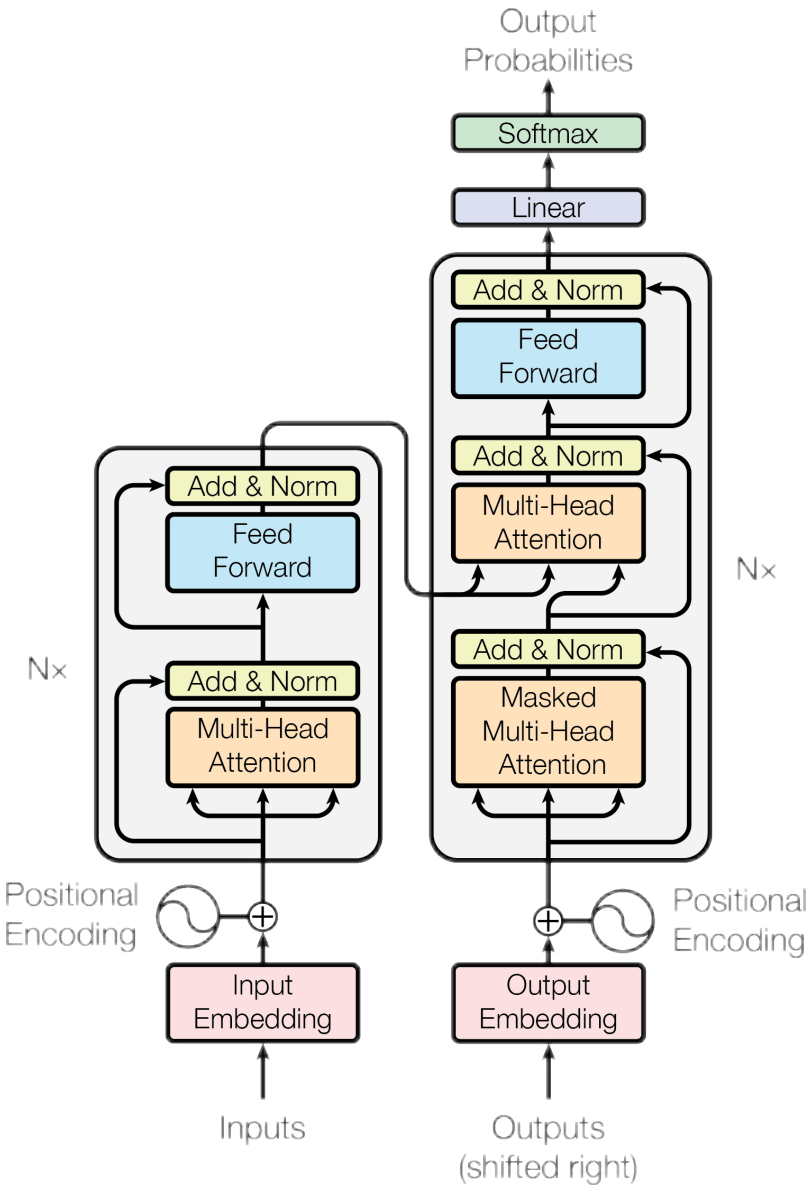
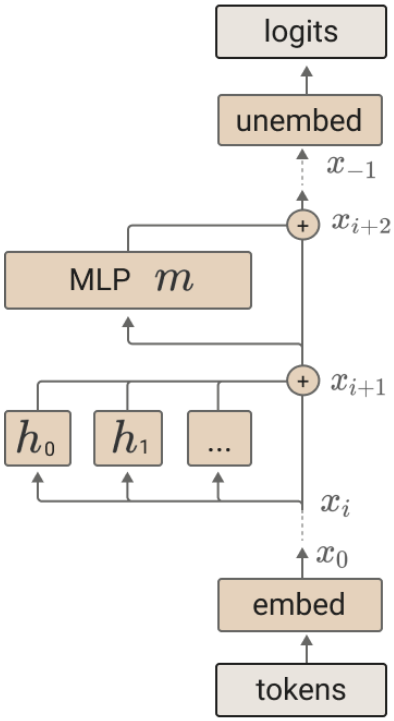


Figure 5: Vaswani et al. 2017.



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.

$$x_0 = W_E t$$

Figure 6: Elhage et al. 2021.

Residual Stream as Communication Channel

- main idea: attention heads and MLPs *add information* to the residual stream
- compositionality, responsibility splitting
- residual stream hardly interpretable, components adding to it may be interpretable
- **We (probably) don't want to interpret the residual stream!**

Attention Heads are Independent and Additive

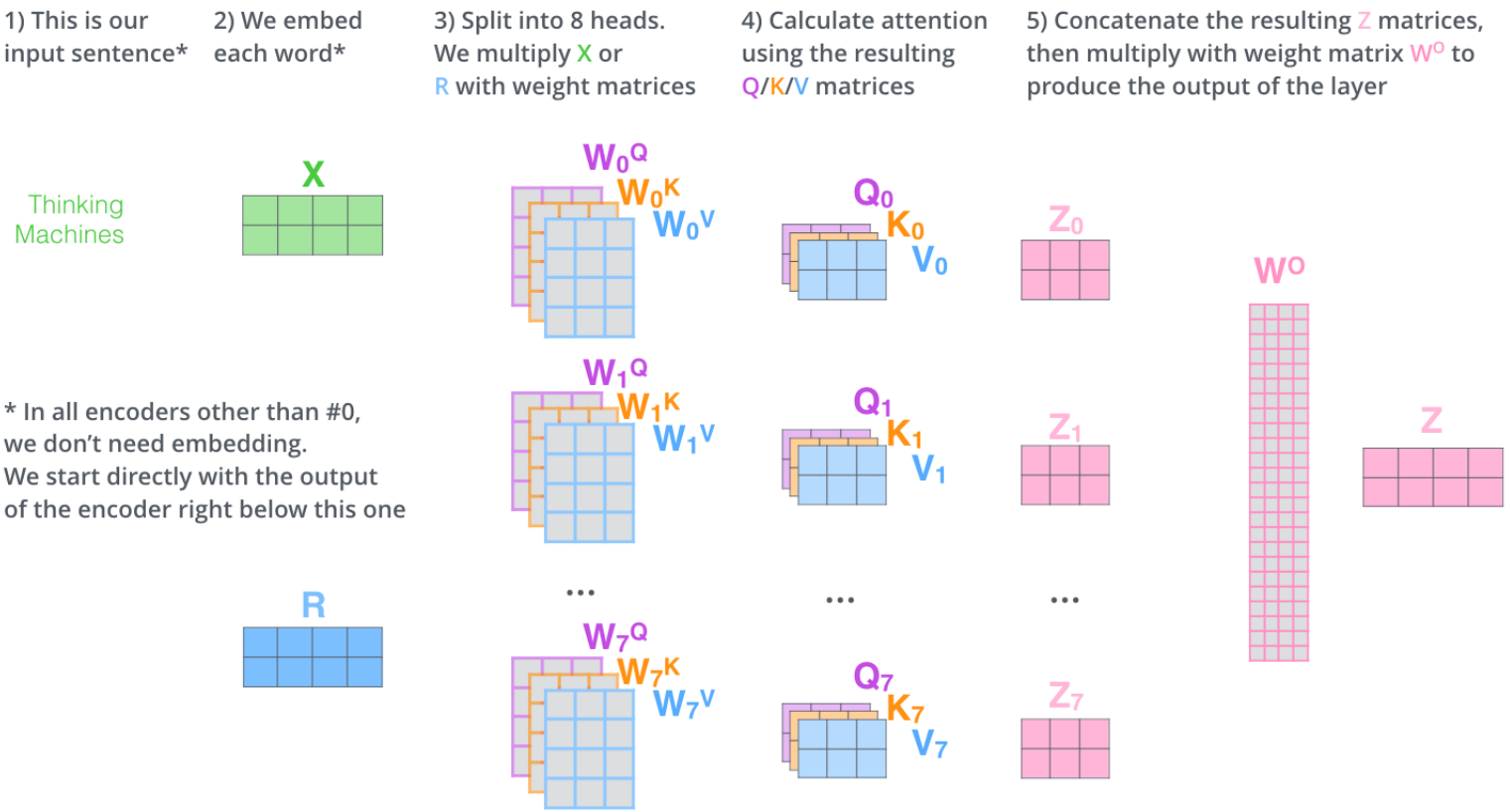


Figure 7: Alammar 2018.

Attention Heads as Information Movement

Each vector receives three representations ("roles")

$W_Q \times = $ Query: vector **from** which the attention is looking

"Hey there, do you have this information?"

$W_K \times = $ Key: vector **at** which the query looks to compute weights

"Hi, I have this information – give me a large weight!"

$W_V \times = $ Value: their weighted sum is attention output

"Here's the information I have!"

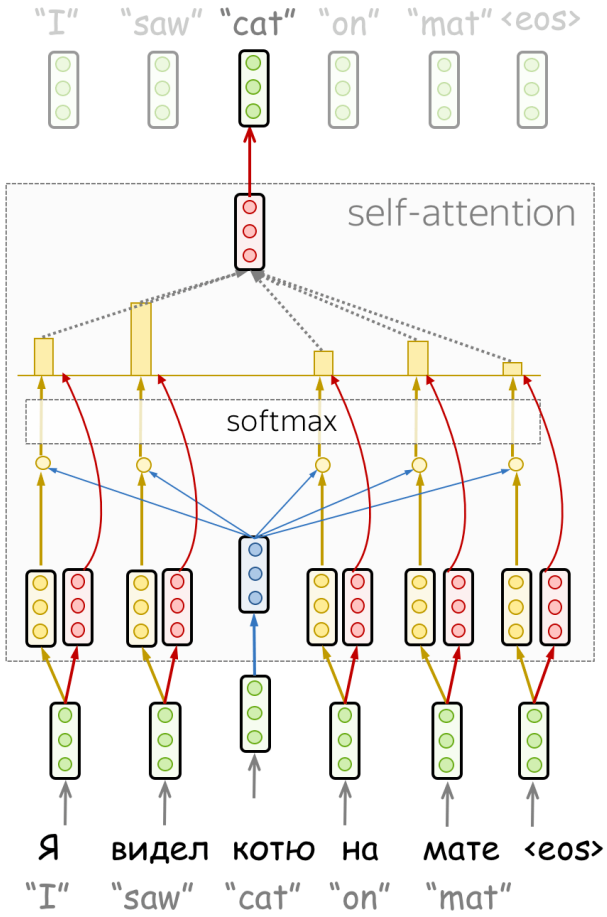
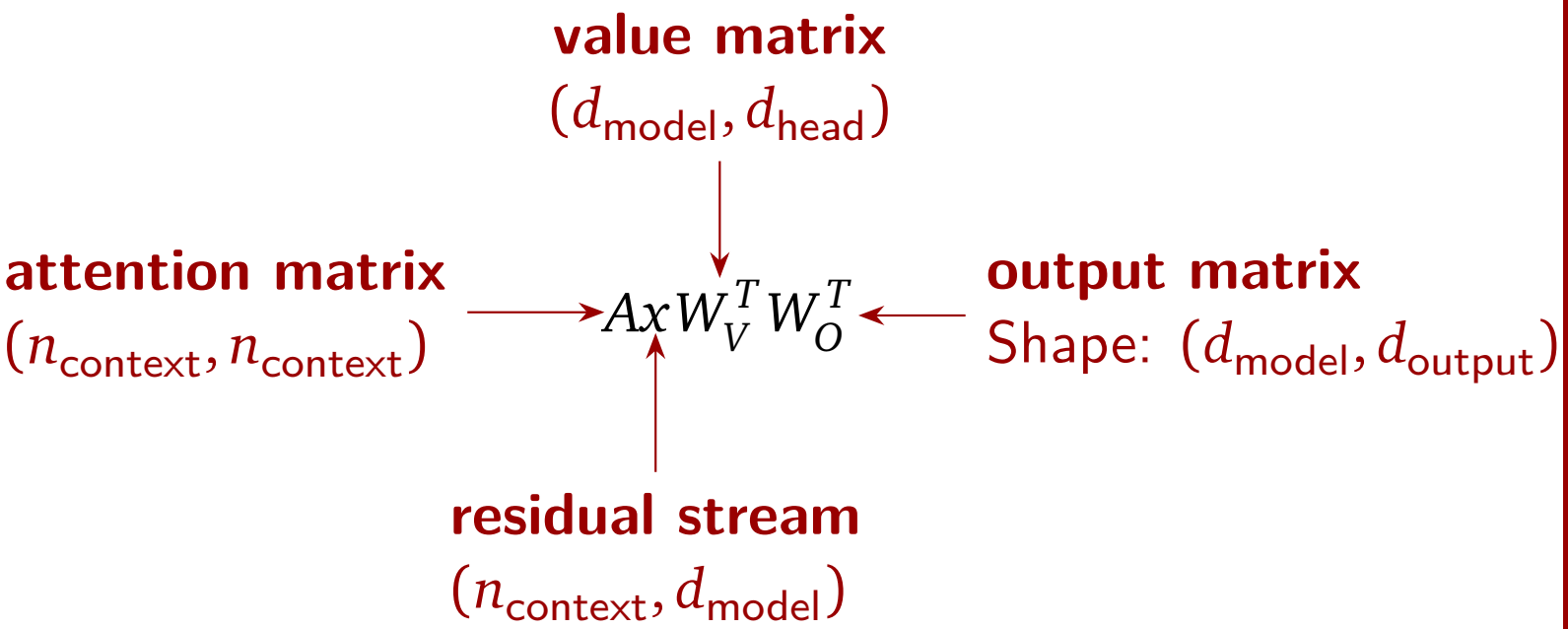
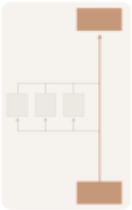


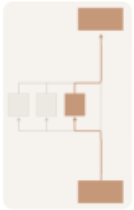
Figure 8: Voita 2023.

Attention Heads as Information Movement



$$T = \underbrace{\mathrm{Id} \otimes W_U W_E}_{\text{"Direct path" term contributes to bigram statistics.}} + \underbrace{\sum_{h \in H} A^h \otimes (W_U W_{OV}^h W_E)}_{\text{The attention head terms describe the effects of attention heads in linking input tokens to logits. } A^h \text{ describes which tokens are attended to while } W_U W_{OV}^h W_E \text{ describes how each token changes the logits if attended to.}}$$





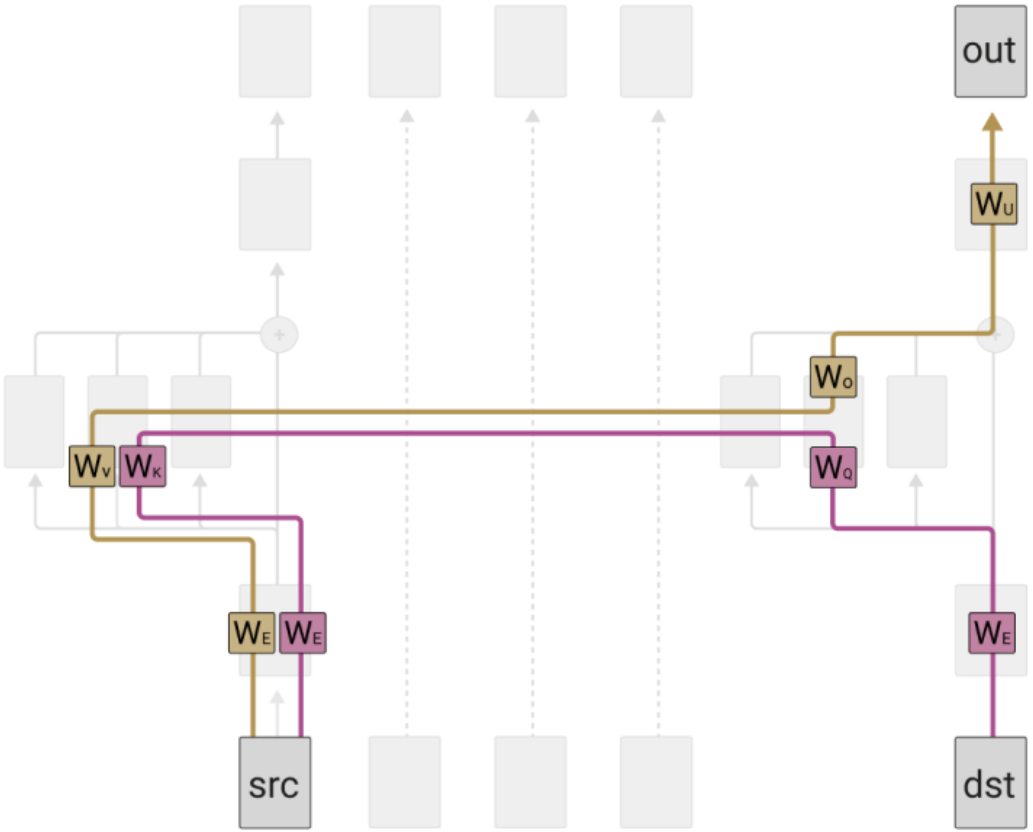
$$T(x) = W_U W_E x^T + \sum_{h \in H} A^h x (W_U W_{OV}^h W_E)^T$$

$$T = \underbrace{\mathrm{Id} \otimes W_U W_E}_{\text{"Direct path" term contributes to bigram statistics.}} + \underbrace{\sum_{h \in H} A^h \otimes (W_U W_{OV}^h W_E)}_{\text{The attention head terms describe the effects of attention heads in linking input tokens to logits. } A^h \text{ describes which tokens are attended to while } W_U W_{OV}^h W_E \text{ describes how each token changes the logits if attended to.}}$$

$$T(x) = W_U W_E x^T + \sum_{h \in H} A^h x (W_U W_{OV}^h W_E)^T$$

We can look at each attention head independently. Attention is the only communication possibility, enabling skip-trigrams.

Attention Heads as Information Movement



The **OV ("output-value") circuit** determines how attending to a given token affects the logits.

$$W_U W_O W_V W_E$$

The **QK ("query-key") circuit** controls which tokens the head prefers to attend to.

$$W_E^T W_Q^T W_K W_E$$

Skip-Trigrams

Some examples of large entries QK/OV circuit

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"

The Tokenizer

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”

Bugs

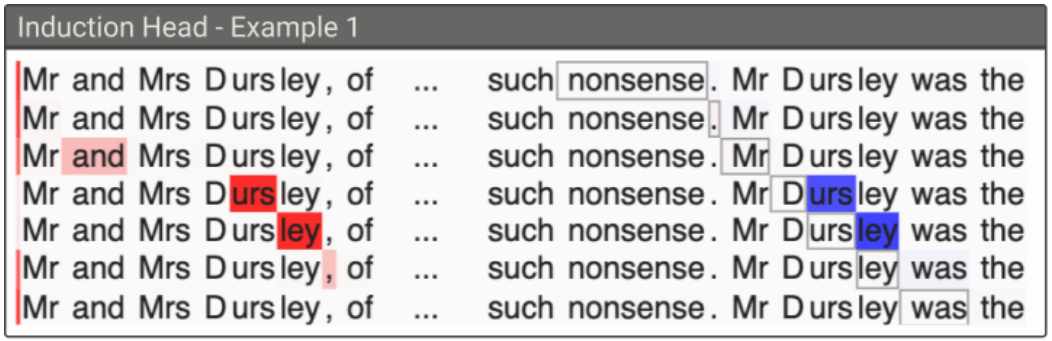
Limited Expressivity Can Create Bugs which Seem Strange from the Outside

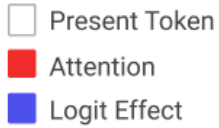
Source Token	Destination Token	Out Token	"Correct" Skip Tri-grams	"Bug" Skip Tri-grams
" Pixmap"	" P", " Q", "P", " p", " U"	"ixmap", "Canvas", "Embed", "grade"	" Pixmap... Pixmap", " Pixmap... QCanvas"	" Pixmap... PCanvas"

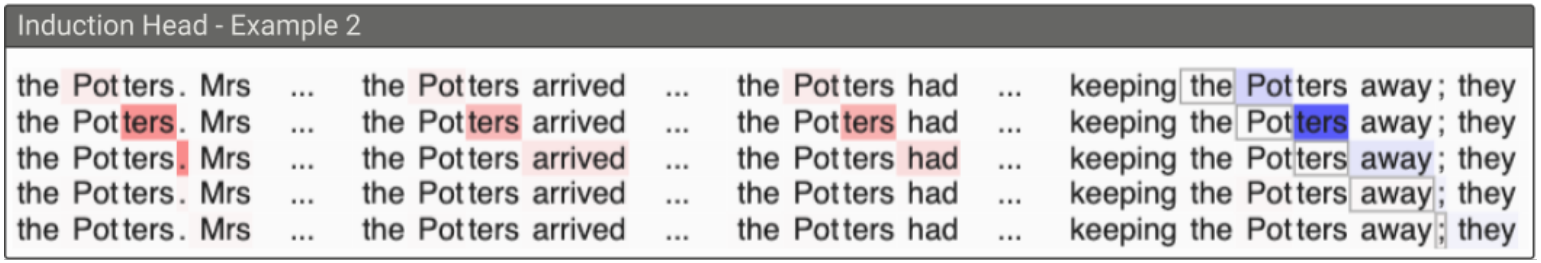
Source Token	Destination Token	Out Token	"Correct" Skip Tri-grams	"Bug" Skip Tri-grams
" Lloyd"	"L", " L", " P", "P", "R", " C"	"loyd", "alph", "\n ", "acman", ... "atherine"	" Lloyd... Lloyd", " Lloyd... Catherine"	" Lloyd... Cloyd", " Lloyd... Latherine"

Source Token	Destination Token	Out Token	"Correct" Skip Tri-grams	"Bug" Skip Tri-grams
" keep"	" in", " at", " out", " under", " off"	" bay", ... " mind", ... " wraps"	" keep... in mind", " keep... at bay", " keep... under wraps"	" keep... in bay", " keep... at wraps", " keep... under mind"

Induction Heads







Wrapping Up

- How would you conceptualize the residual stream?
- What does attention essentially do?
  - What is the meaning of query, key, and value?
- What does an MLP essentially do?
- What can a One-Layer Attention-Only Transformer do?
- What do we enable with two layers?
- What is an Induction Head?

References

References


Alammar, Jay (2018). *The Illustrated Transformer*. https://jalammar.github.io/illustrated-transformer/. [Blog post].


Elhage, Nelson et al. (2021). "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread*. https://transformer-circuits.pub/2021/framework/index.html.


Raffel, Colin et al. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140, pp. 1–67. url: http://jmlr.org/papers/v21/20-074.html.


Tunstall, Lewis, Leandro von Werra, and Thomas Wolf (2022). *Natural Language Processing with Transformers*. Sebastopol: O'Reilly.

References

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. url: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Voita, Elena (Nov. 2023). *NLP Course For You*. url: https://lena-voita.github.io/nlp_course.html.