Self-Attention, MLP, and Interpretability

Haw-Shiuan Chang

Logistics

- TA Nguyen Tran permanently moved his TA office hour
  - from Fri 3pm-4pm
  - to Fri 4pm-5pm (1 hour later)

Deadlines

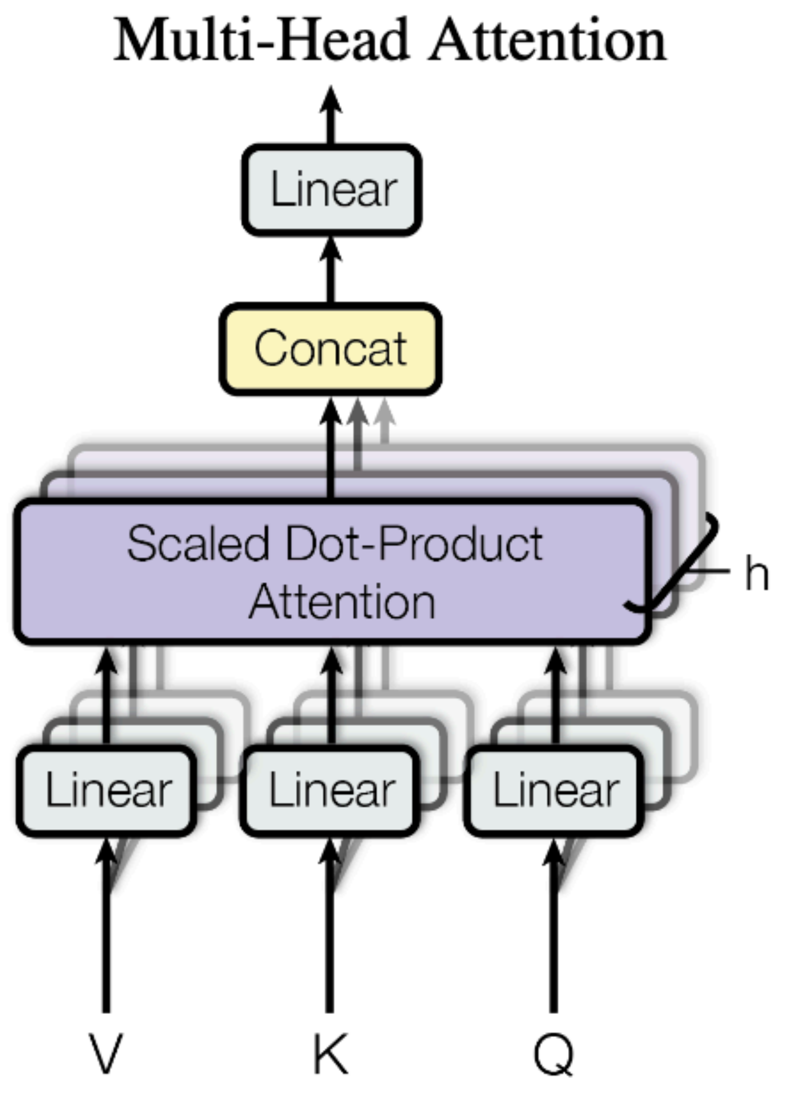
- **https://people.cs.umass.edu/~hschang/cs685/schedule.html**

- **3/3:** Quiz 2 due
  - I might extend the deadline on Piazza (if I cannot finish teaching cross-attention this Wednesday)
- **3/7**: Project proposals due
  - If you cannot reach all of your group members this week, please report the situation on Piazza privately
  - If you have some project ideas, you can go to TA office hours to seek some feedback.
- **3/14**: HW 1 due
  - It's about BERT, but you should start the annotation early

Please stop me if you don't understand!

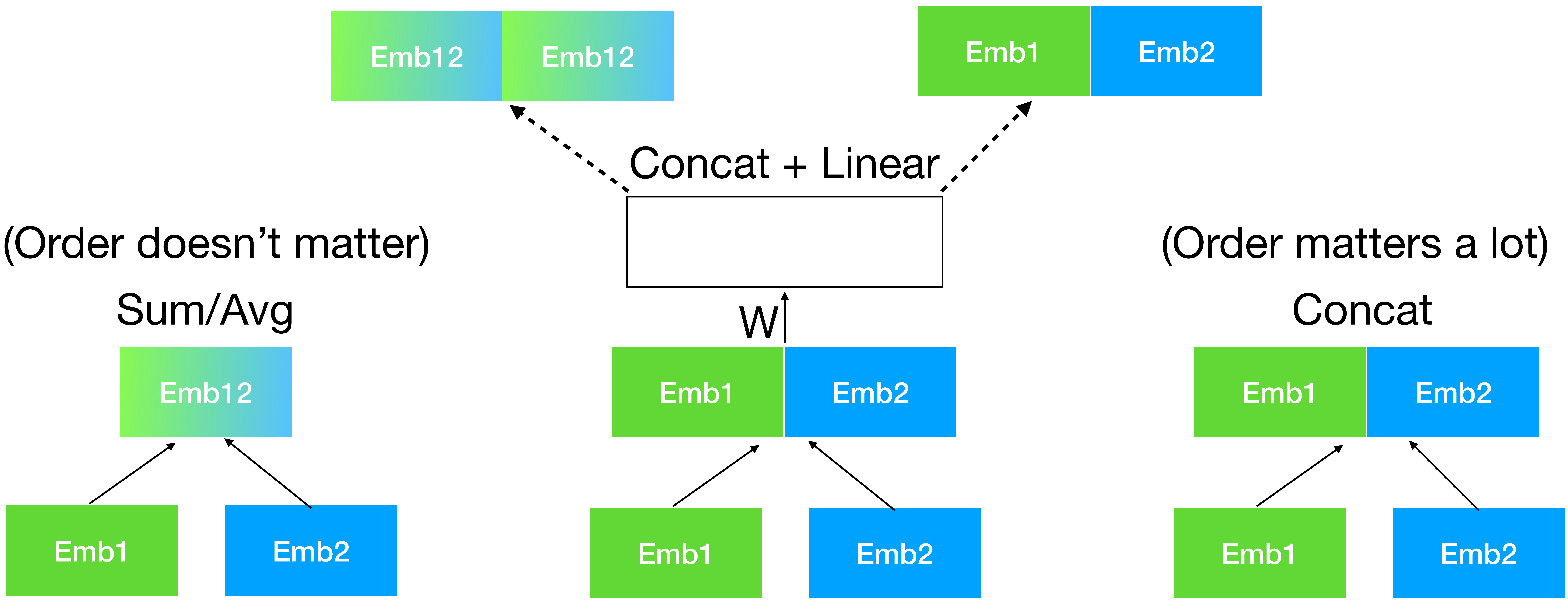
This Lecture will be more advanced, I will explain more slowly

Last Year Note Review



$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

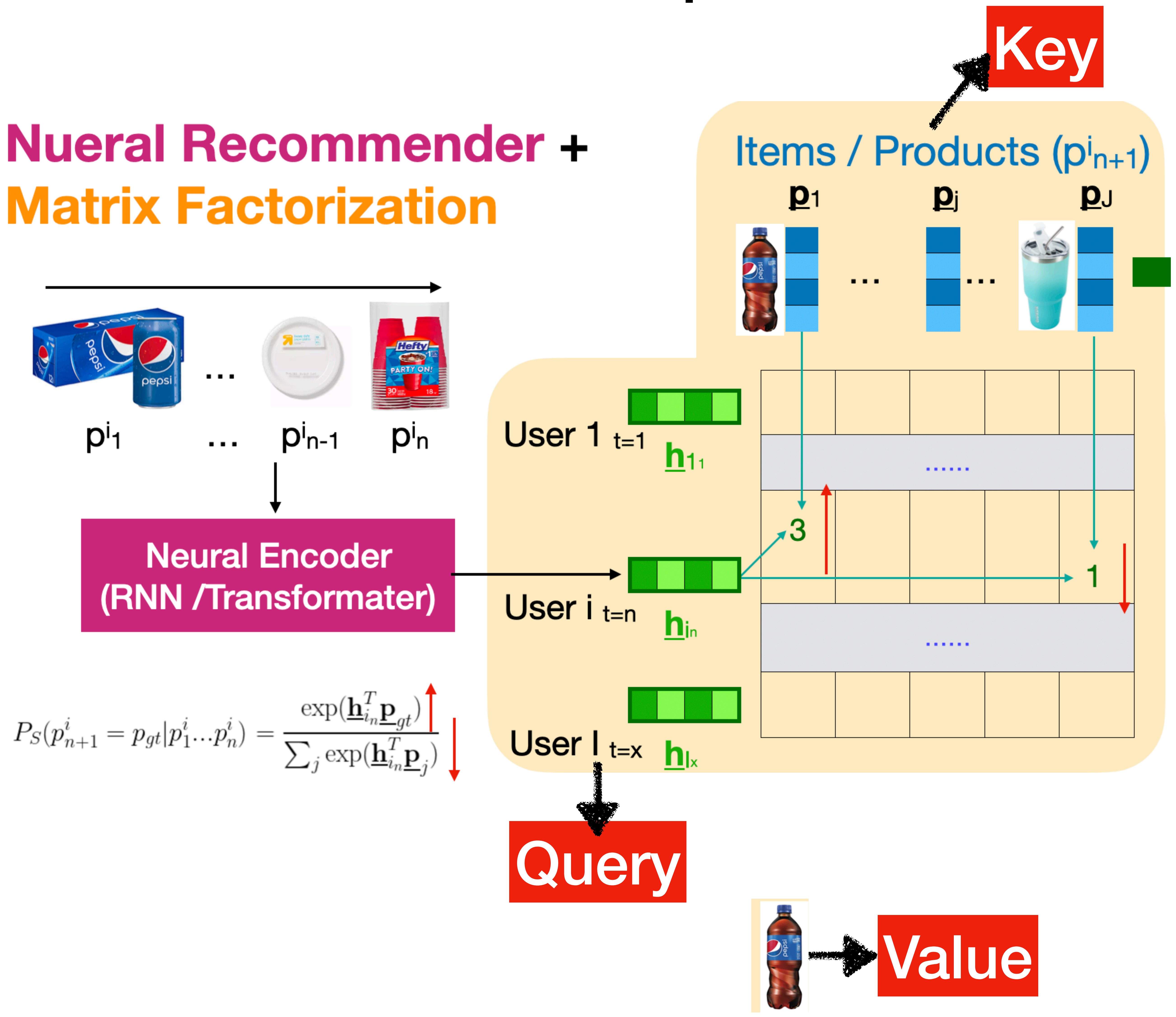
Merging Two Embeddings



Other less common merging includes max, min, product, minus, …

A Metaphor

Nueral Recommender + Matrix Factorization



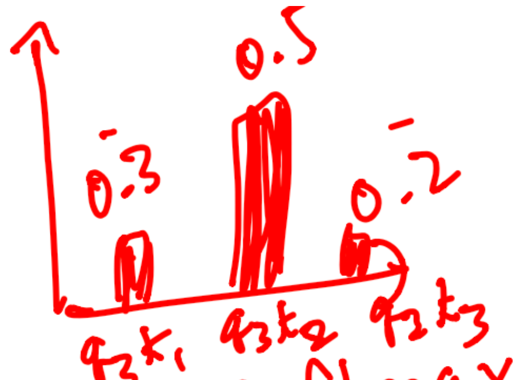
Why do queries and keys need to be different?

1. Matrix factorization's need
2. The diagonal value cannot be low
    1. $q_1^T q_1 > q_1^T q_i$

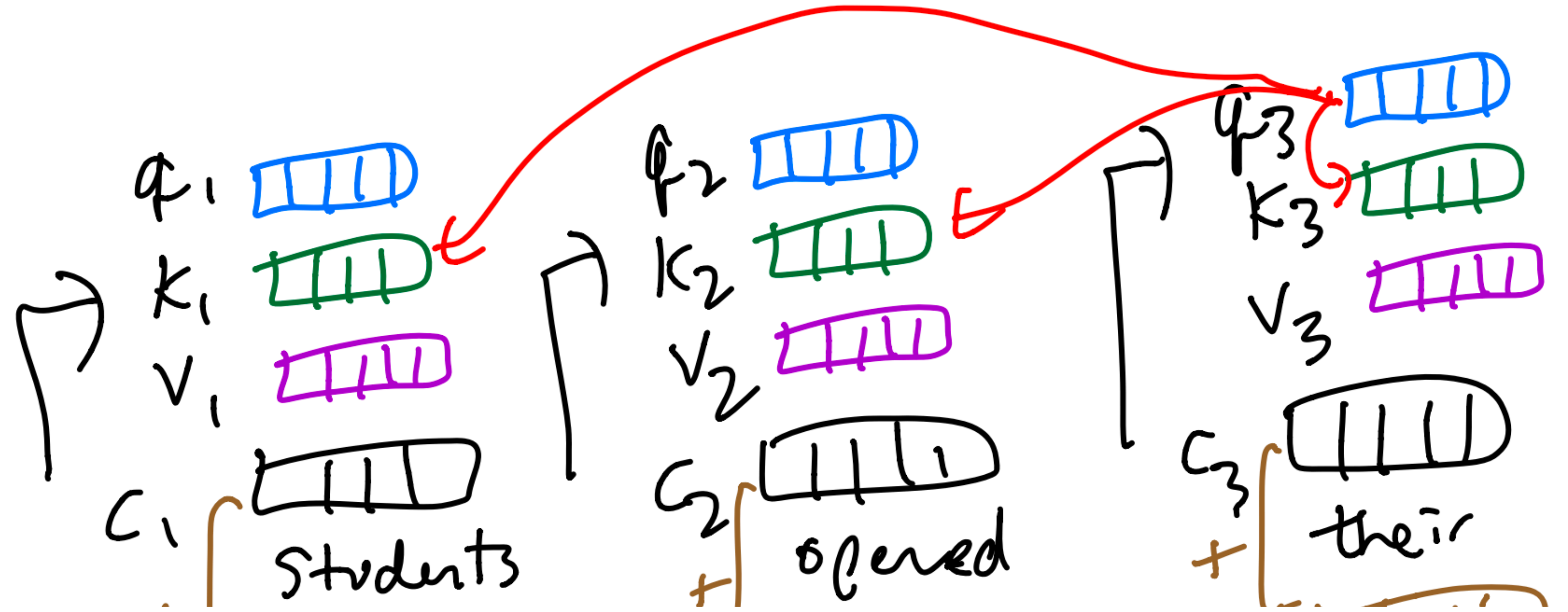
$h_3 = 0.3v_1 + 0.5v_2 + 0.2v_3$

→ softmax

predict "books"



attn scores: softmax($\langle q_3 \cdot k_1, q_3 \cdot k_2, q_3 \cdot k_3 \rangle$)



Note: This is a cross-domain association (not common knowledge among NLP researchers)

Self-Attention Illustrative Example

Why do we need multiple heads?

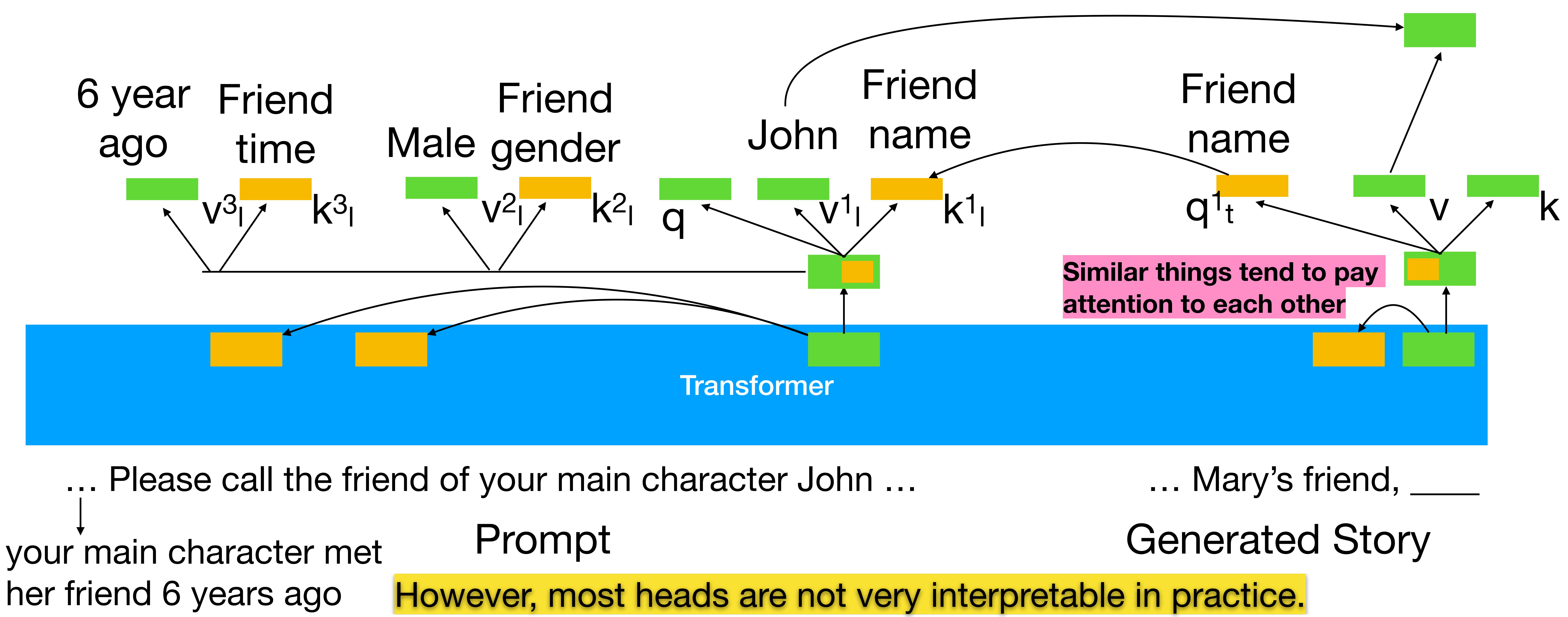
Retrieving only partial information

How do they learn these?

Gradient descent

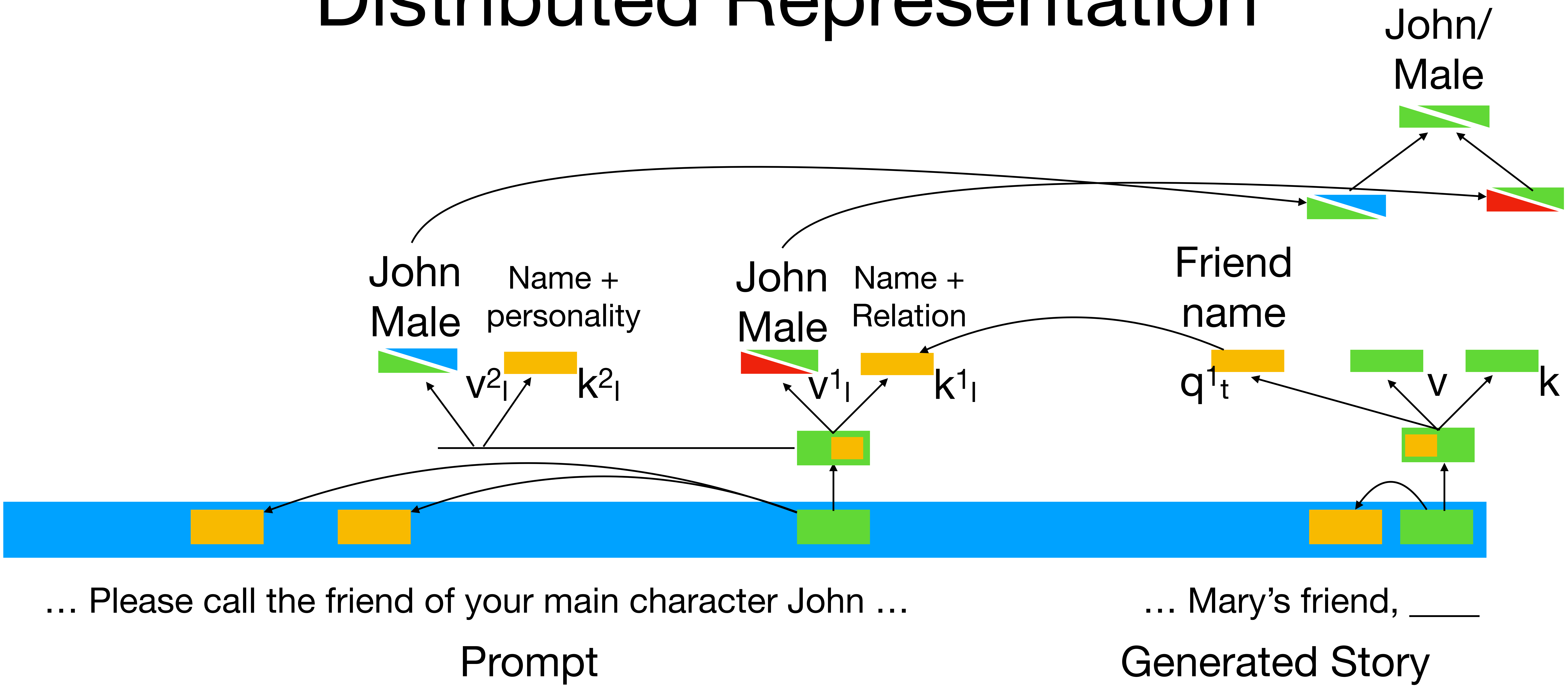
Why are keys and values different?

Action != “property”

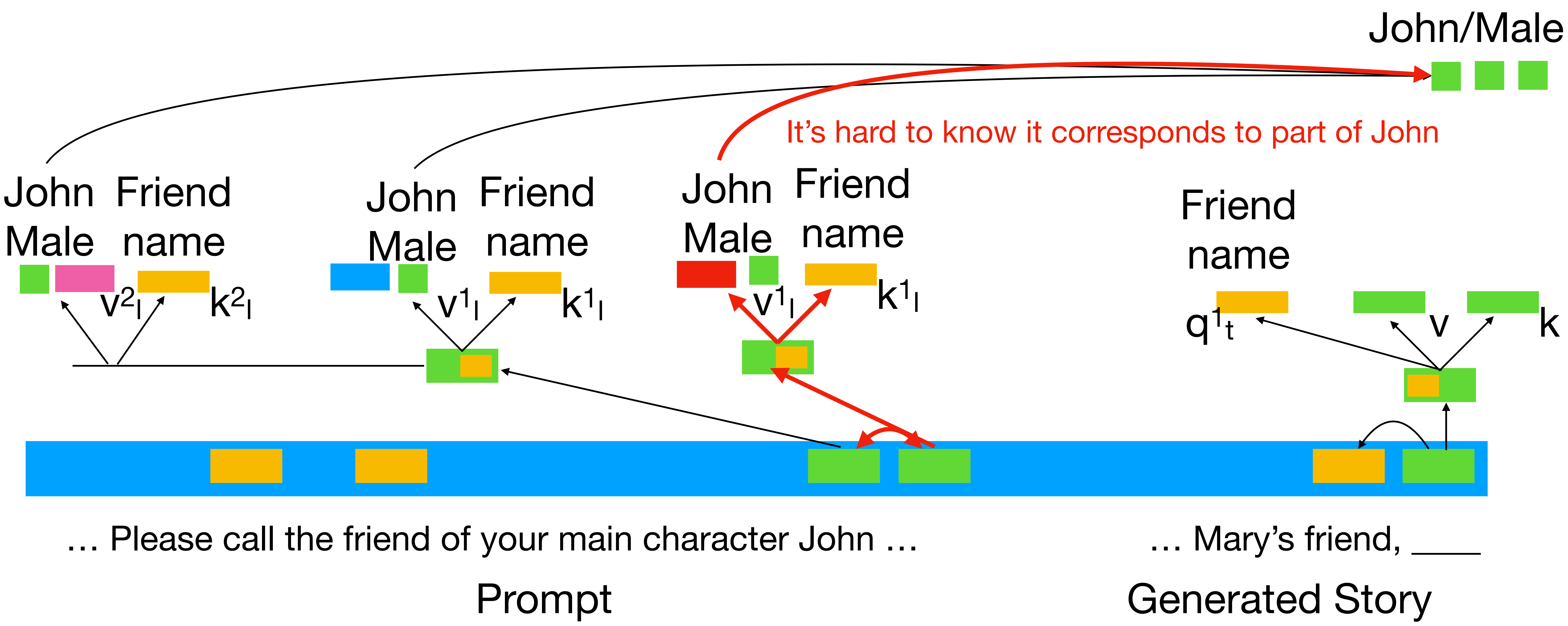


However, most heads are not very interpretable in practice.

Distributed Representation

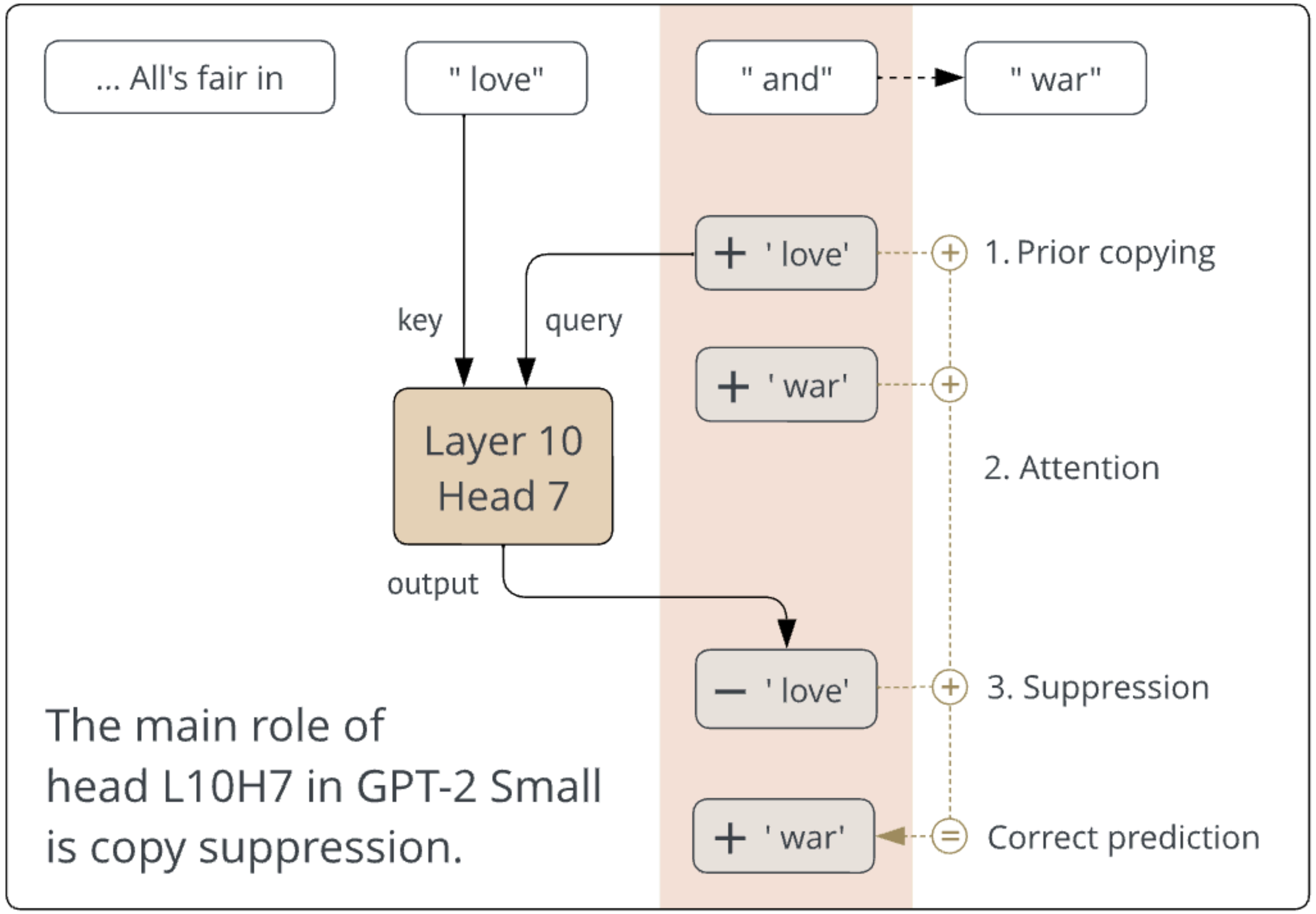


Distributed Representation



https://transformer-circuits.pub/2021/framework/index.html

Interpretable Head Exists but Rare



COPY SUPPRESSION: COMPREHENSIVELY UNDERSTANDING AN ATTENTION HEAD
https://arxiv.org/pdf/2310.04625

A RNN Language Model

Testing time: Autoregressive LM

Inference limitation for self attention

hidden states

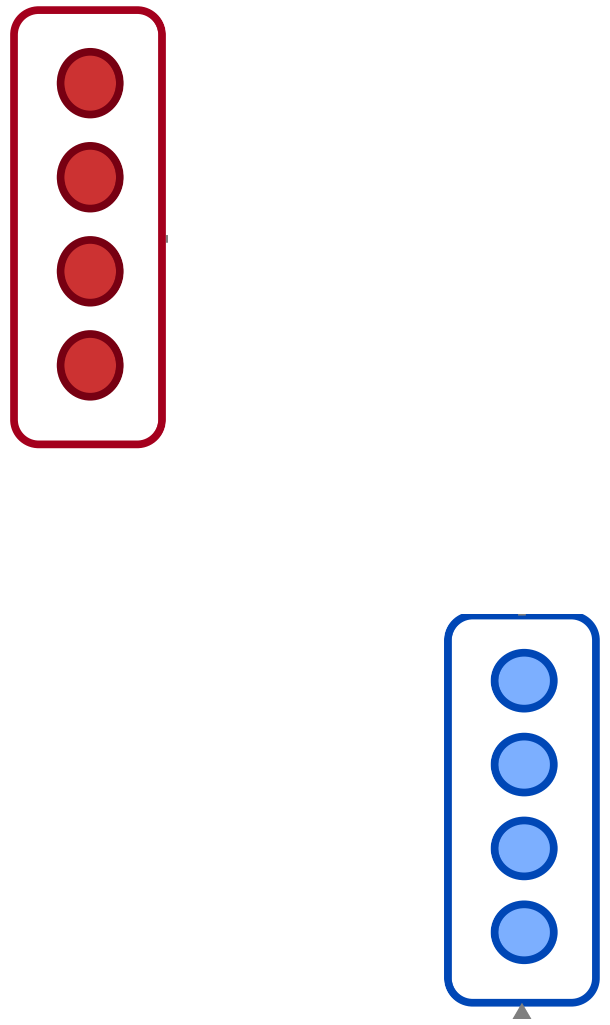
$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

$h^{(0)}$ is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$

$\boldsymbol{h}^{(0)}$



the

c_1

A RNN Language Model

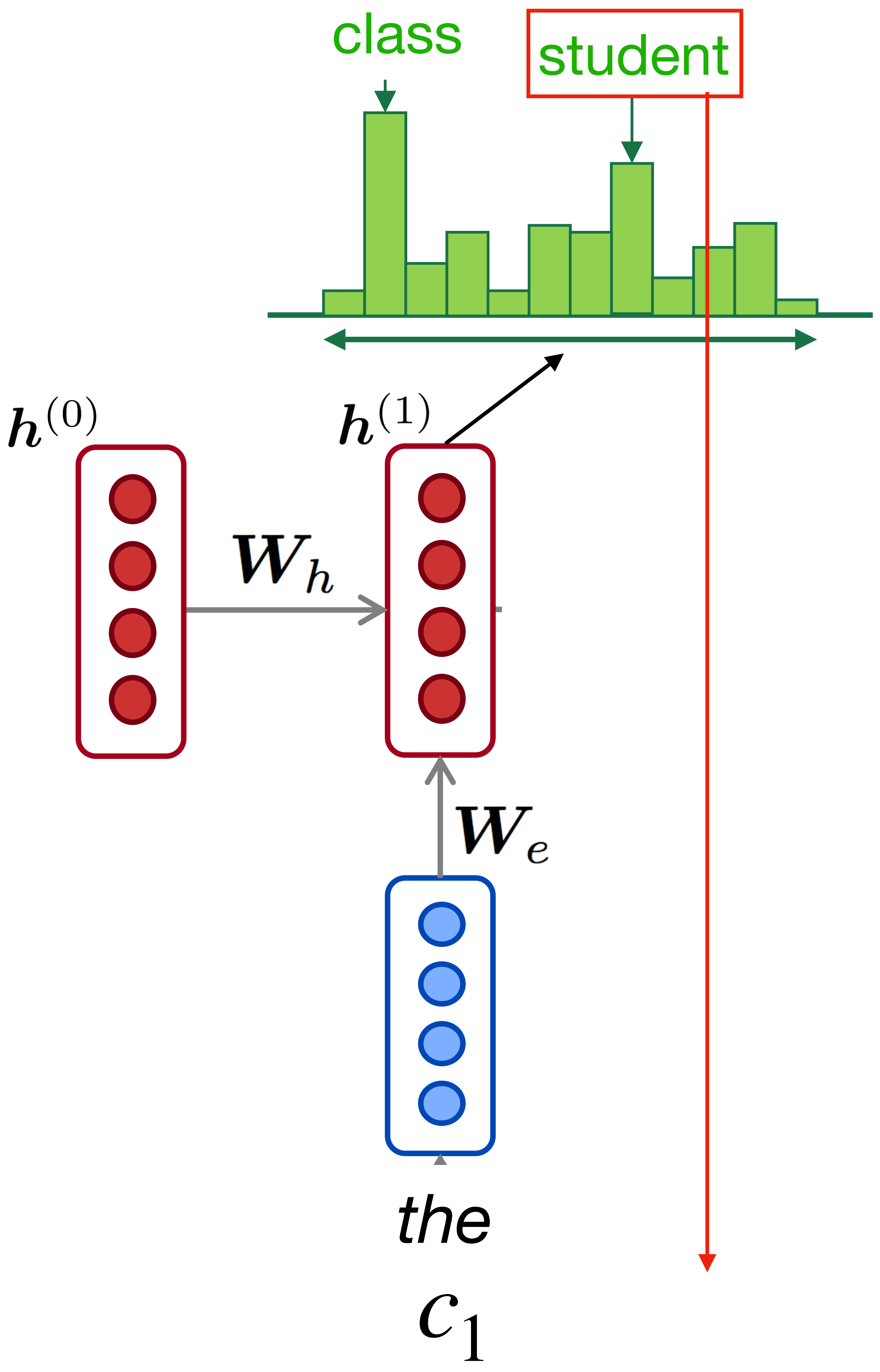
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

$h^{(0)}$ is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



A RNN Language Model

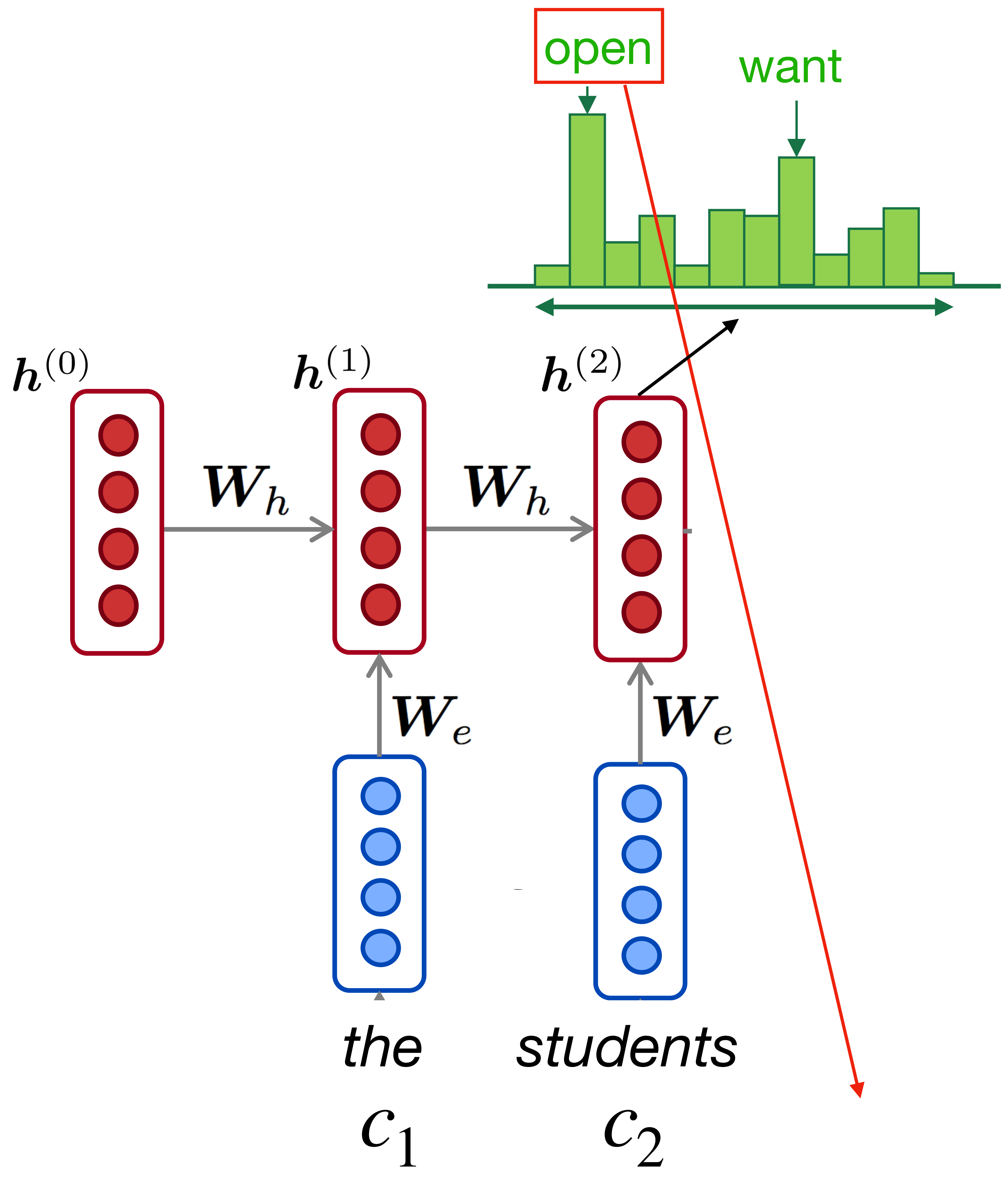
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

$h^{(0)}$ is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



A RNN Language Model

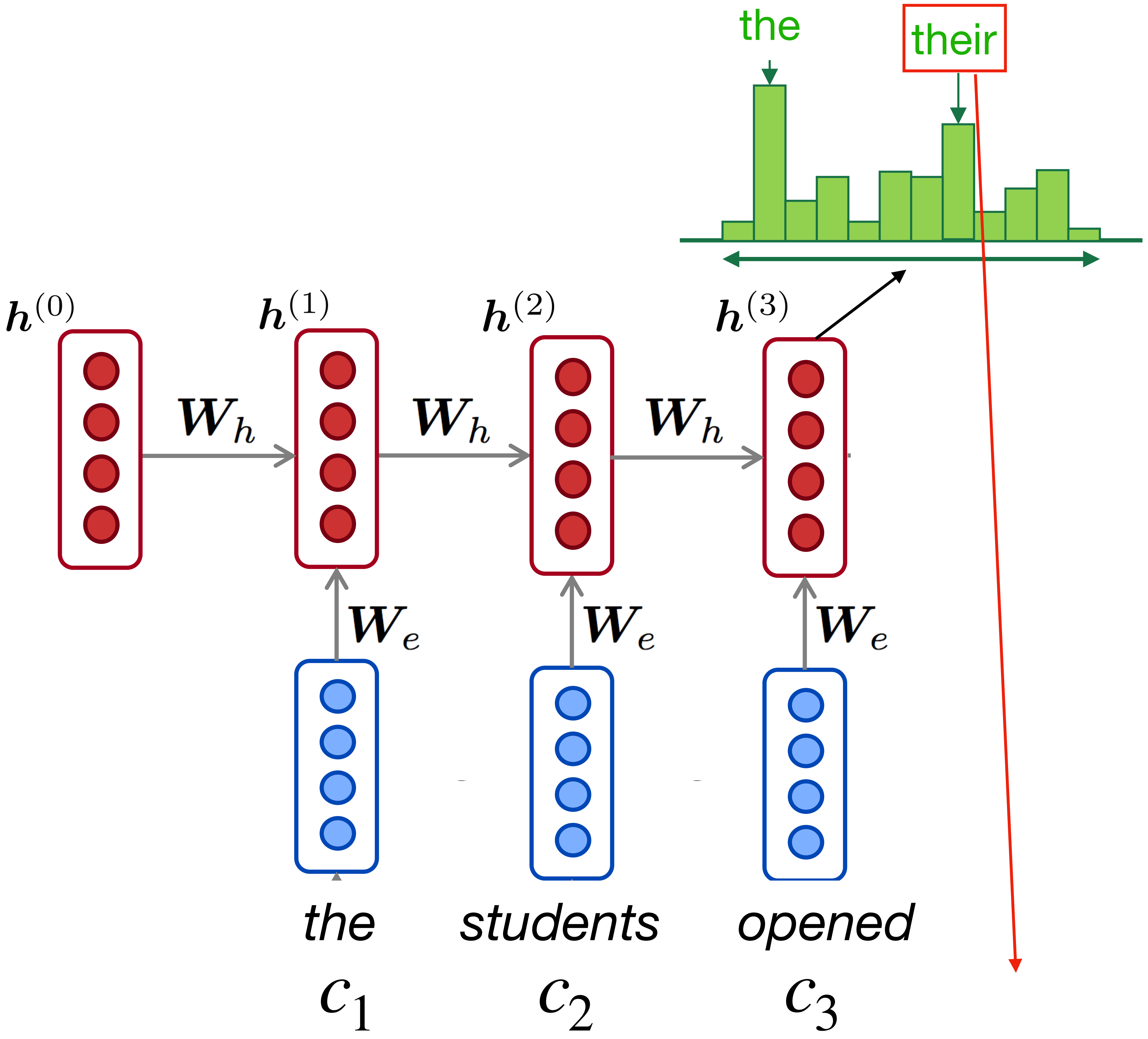
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

$h^{(0)}$ is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



A RNN Language Model

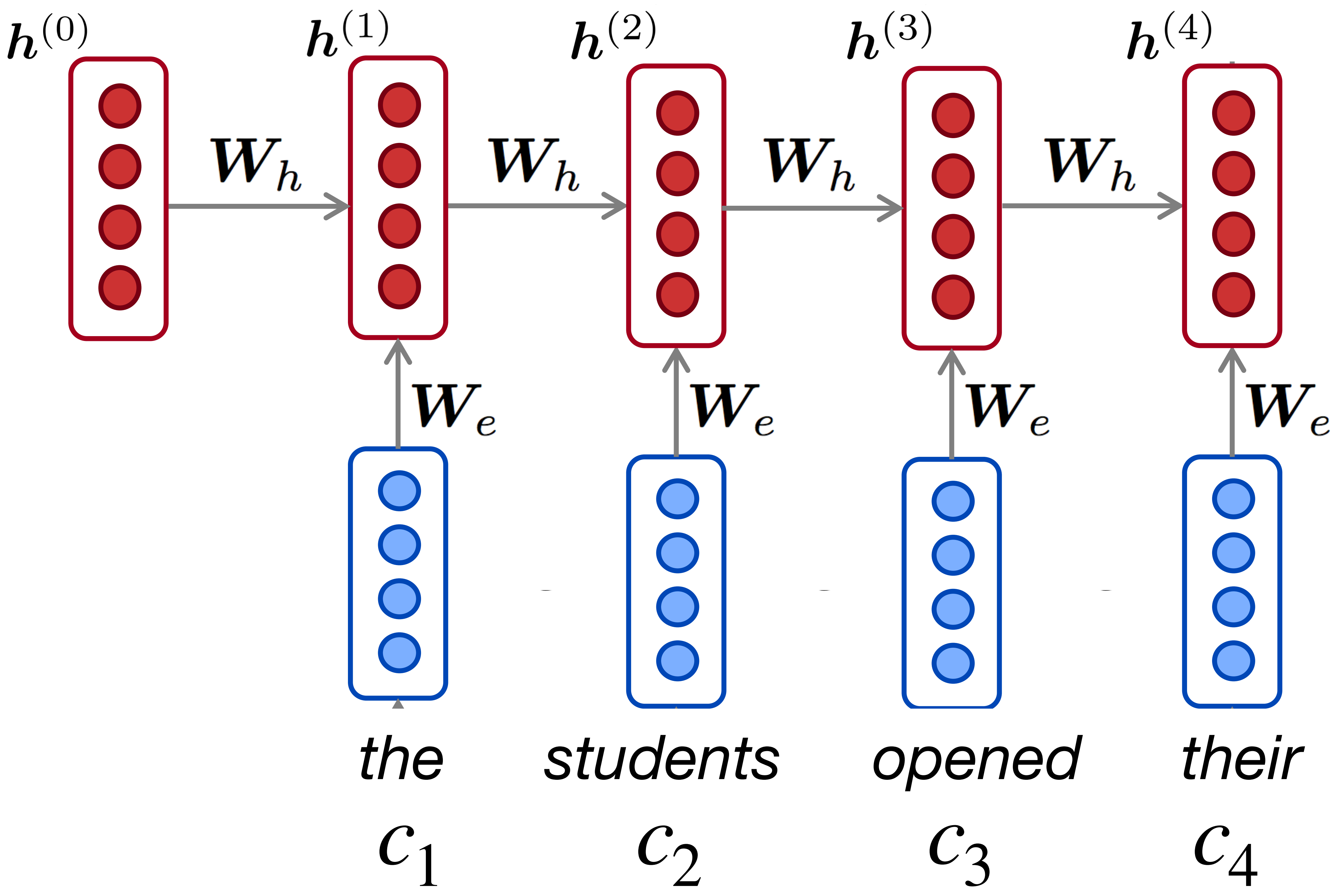
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

$h^{(0)}$ is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



A RNN Language Model

output distribution

$$\hat{y} = \text{softmax}(W_2 h^{(t)})$$

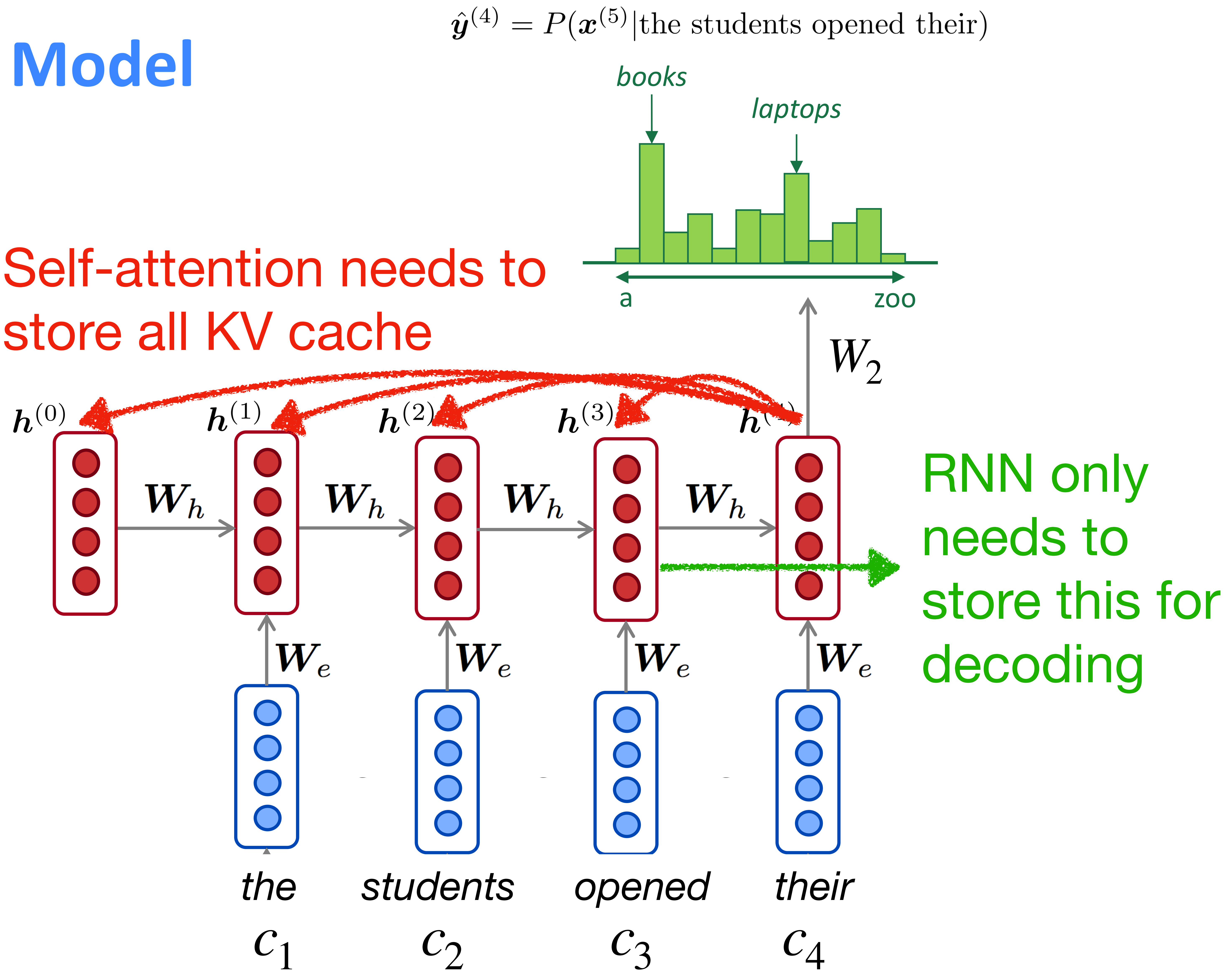
hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c_t)$$

h(0) is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



Multi-Head Latent Attention (MLA)

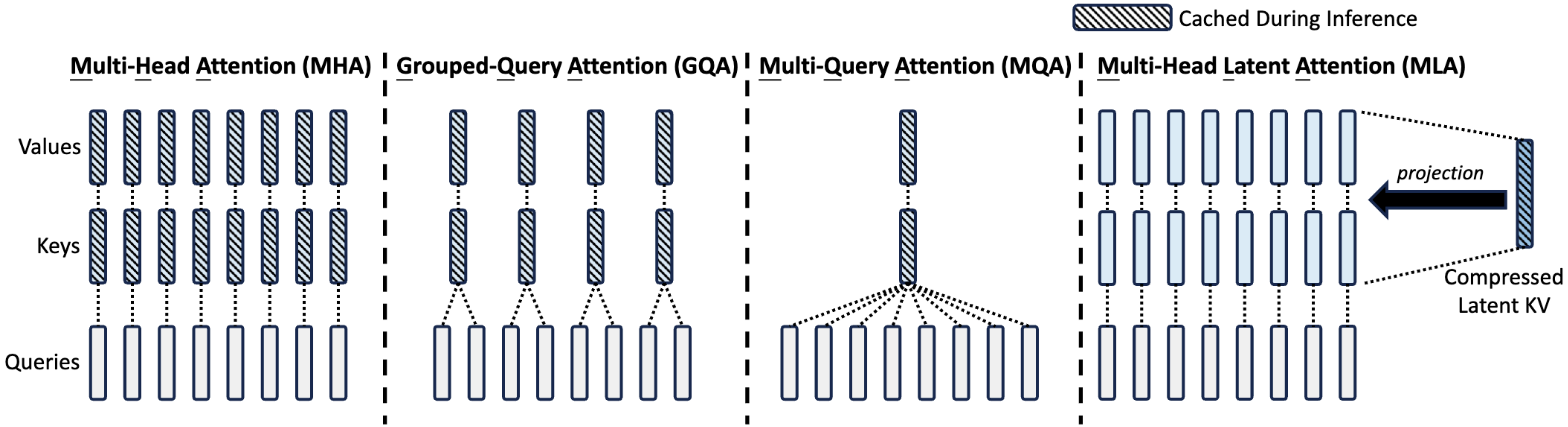
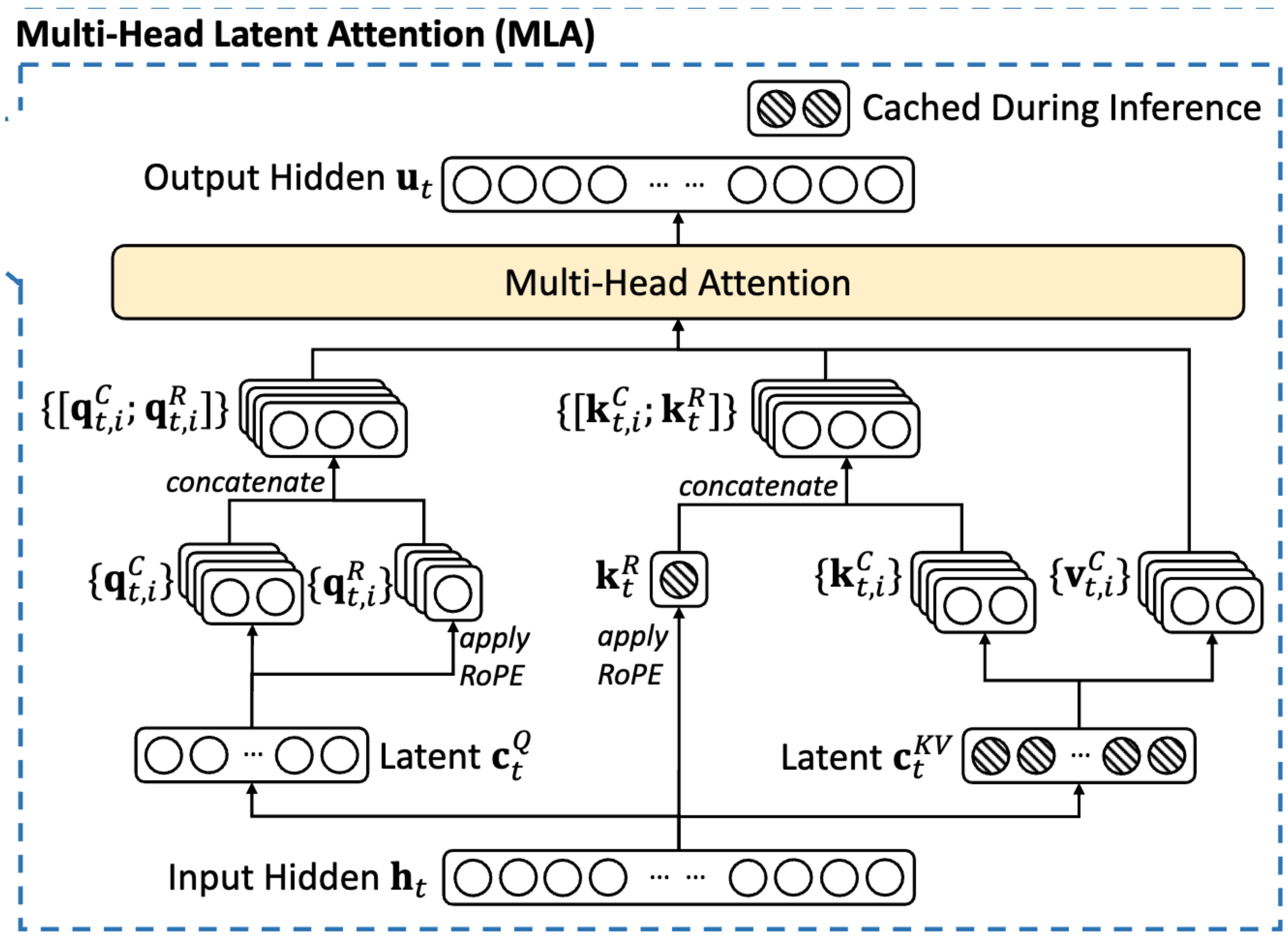


Figure 3 | Simplified illustration of Multi-Head Attention (MHA), Grouped-Query Attention (GQA), Multi-Query Attention (MQA), and Multi-head Latent Attention (MLA). Through jointly compressing the keys and values into a latent vector, MLA significantly reduces the KV cache during inference.

Deepseek V2 (https://arxiv.org/pdf/2405.04434)

Multi-Head Latent Attention (MLA)



Why not store h?

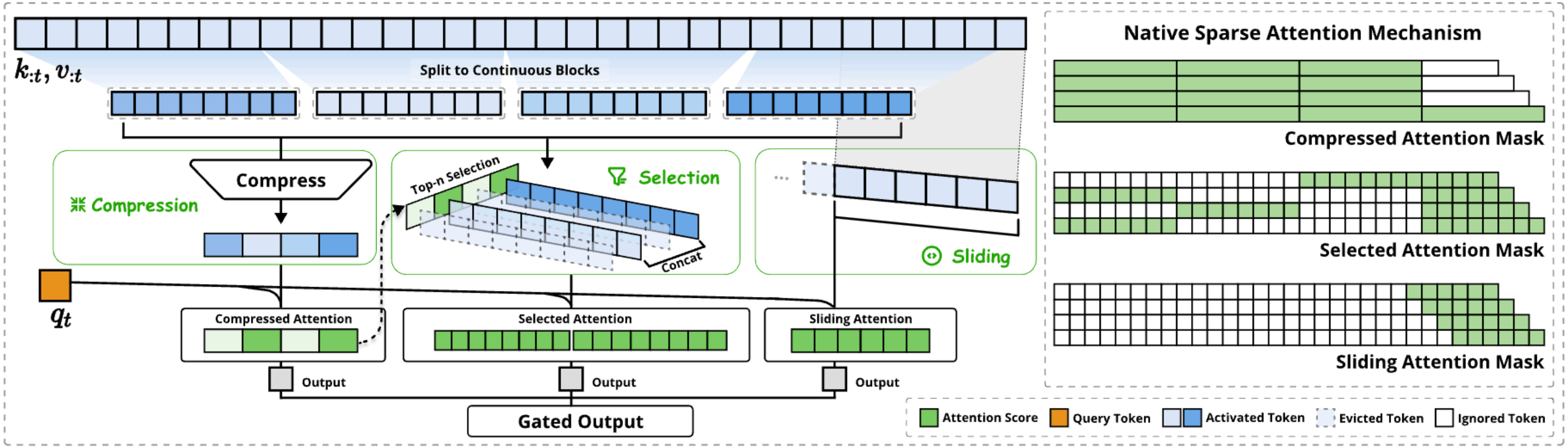
$$q^T k = (W_q h)^T (W_k h) = h^T W_q^T W_k h$$

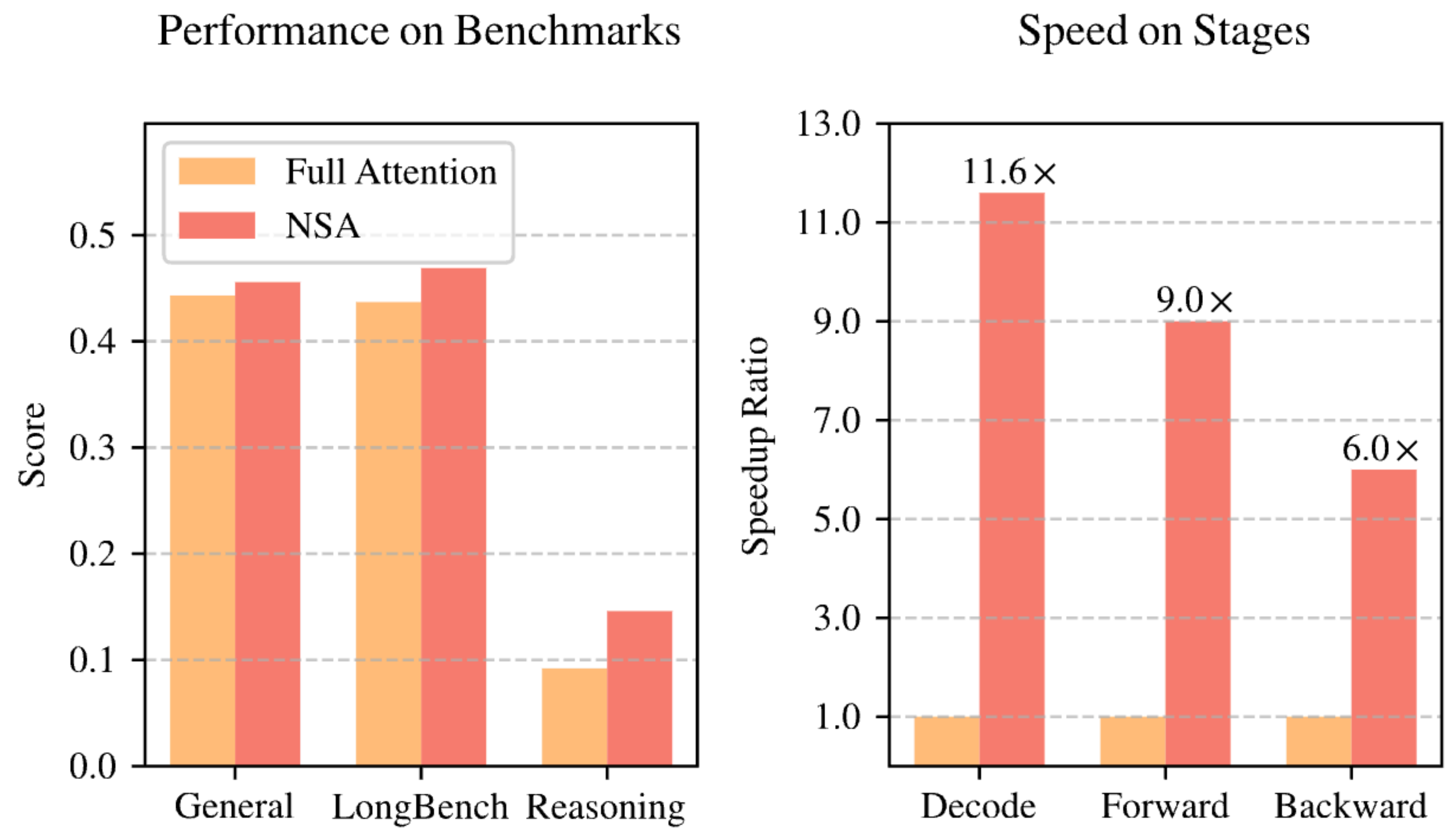
Attention Mechanism	KV Cache per Token (# Element)	Capability
Multi-Head Attention (MHA)	$2n_h d_h l$	Strong
Grouped-Query Attention (GQA)	$2n_g d_h l$	Moderate
Multi-Query Attention (MQA)	$2d_h l$	Weak
MLA (Ours)	$(d_c + d_h^R)l \approx \frac{9}{2} d_h l$	Stronger

Table 1 | Comparison of the KV cache per token among different attention mechanisms. n_h denotes the number of attention heads, d_h denotes the dimension per attention head, l denotes the number of layers, n_g denotes the number of groups in GQA, and d_c and d_h^R denote the KV compression dimension and the per-head dimension of the decoupled queries and key in MLA, respectively. The amount of KV cache is measured by the number of elements, regardless of the storage precision. For DeepSeek-V2, d_c is set to $4d_h$ and d_h^R is set to $\frac{d_h}{2}$. So, its KV cache is equal to GQA with only 2.25 groups, but its performance is stronger than MHA.

https://github.com/Zefan-Cai/Awesome-LLM-KV-Cache

Hierarchical Attention





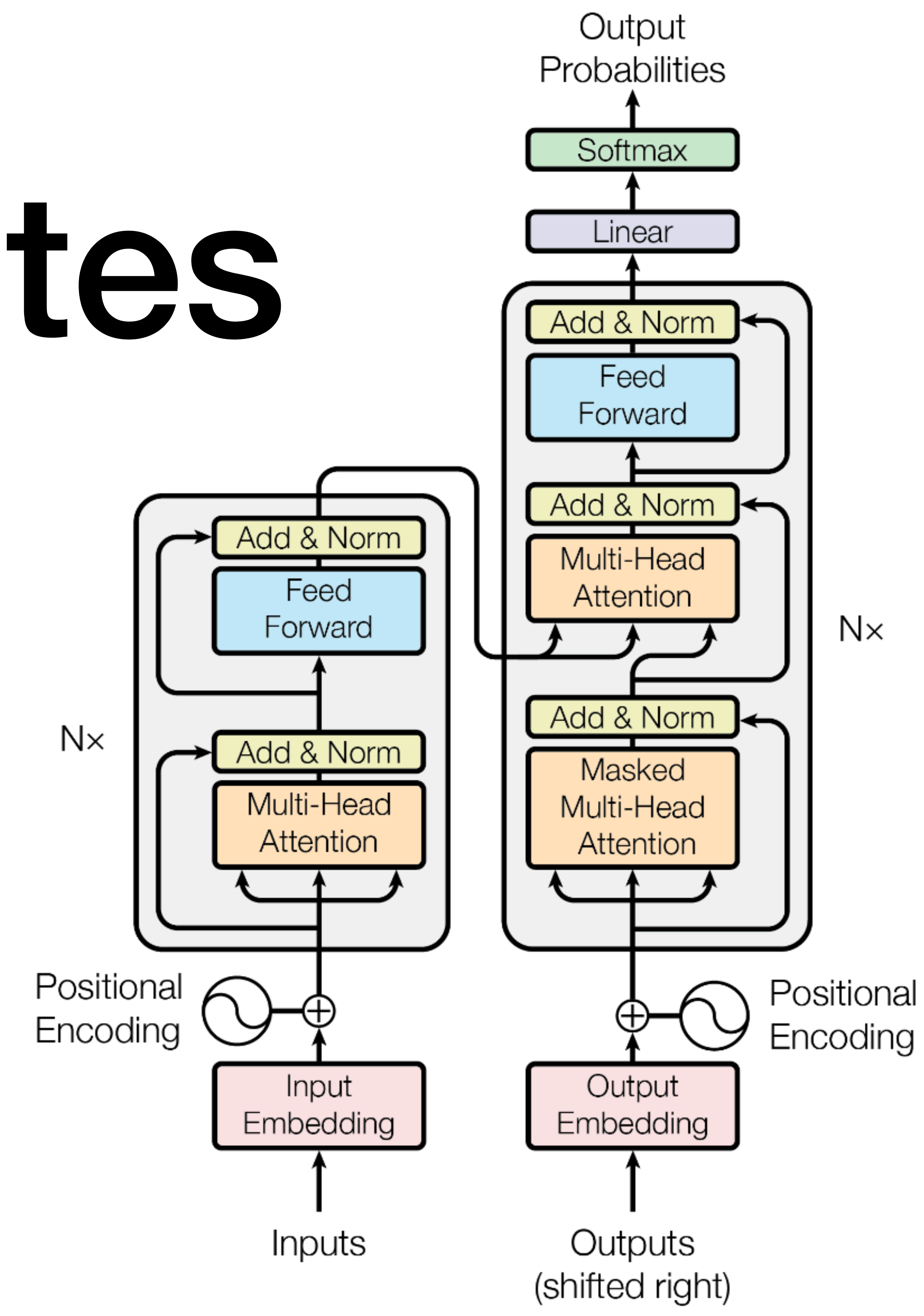
Similar to RAG

Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention (https://arxiv.org/pdf/2502.11089)

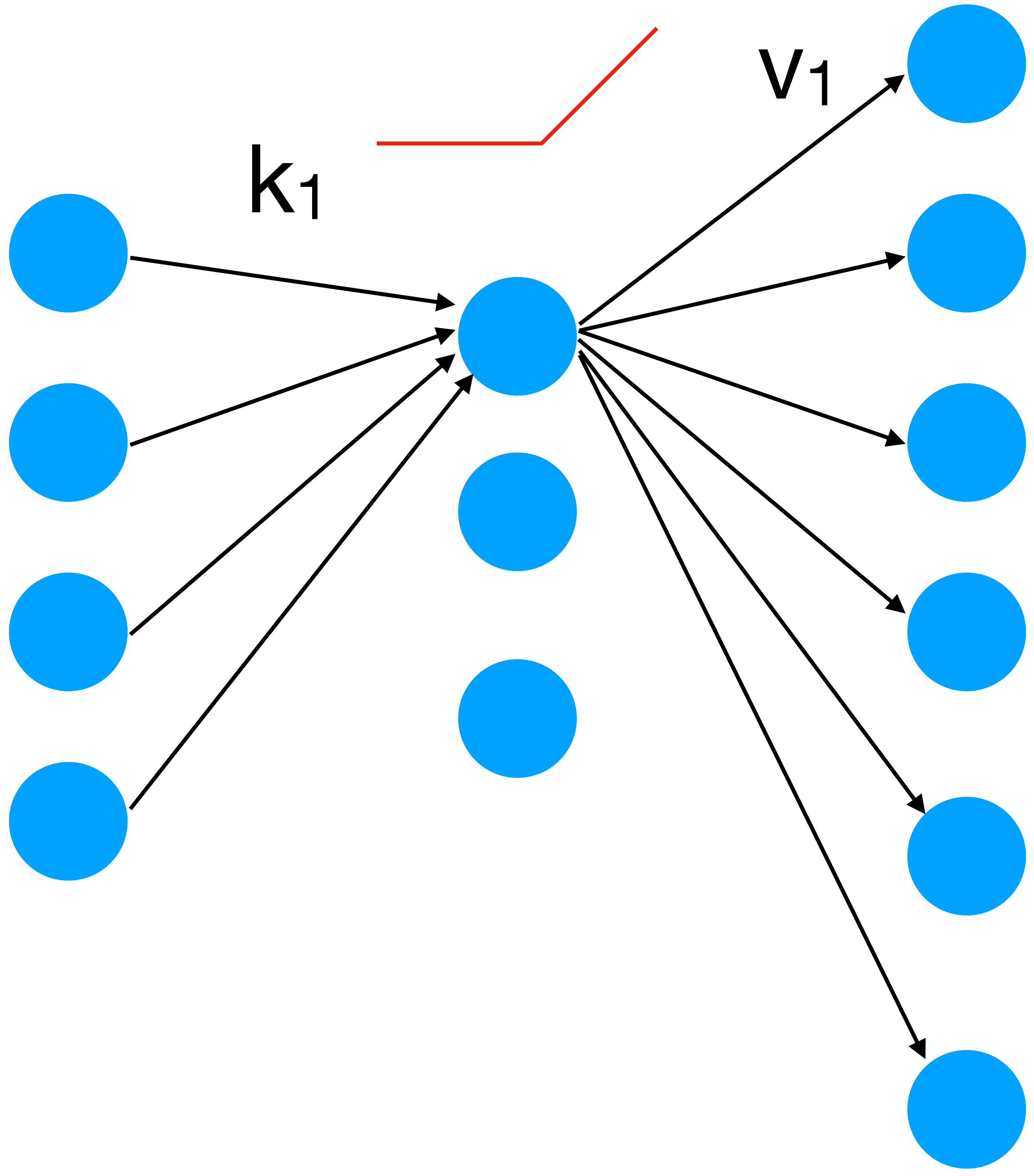
(From deepseek AI)

Last Year Notes

$$Layernorm(x) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$



MLP / Feed forward NN



Transformer Feed-Forward Layers Are Key-Value Memories (https://arxiv.org/pdf/2012.14913)

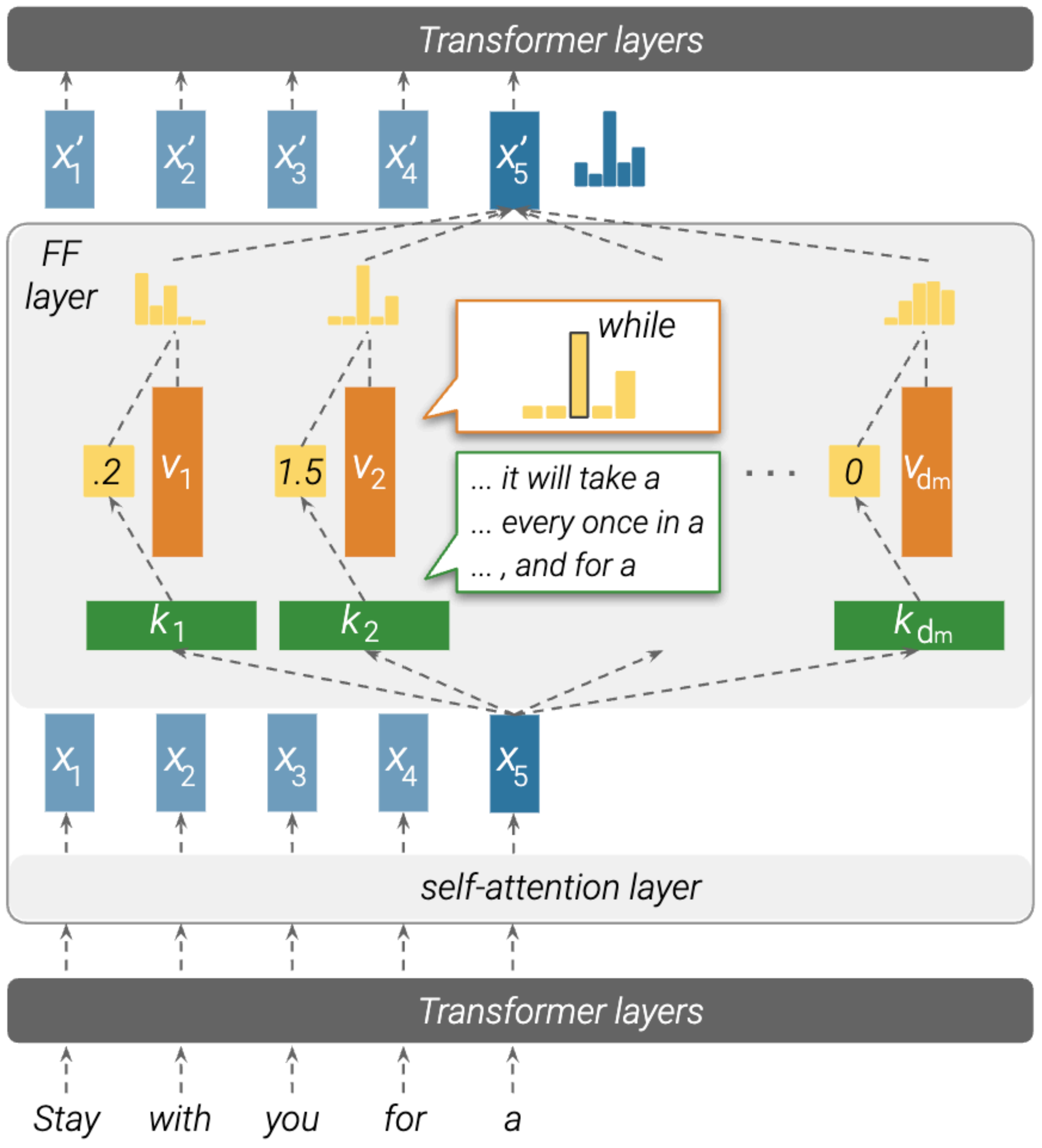


Figure 1: An illustration of how a feed-forward layer emulates a key-value memory. Input vectors (here, $\mathbf{x}_5$) are multiplied by *keys* to produce *memory coefficients* (e.g., the memory coefficient for $\mathbf{v}_1$ is 0.2), which then weigh distributions over the output vocabulary, stored in the *values*. The feed-forward layer's output is thus the weighted sum of its values.

Non-negativity -> Meaningful Dimensions

V

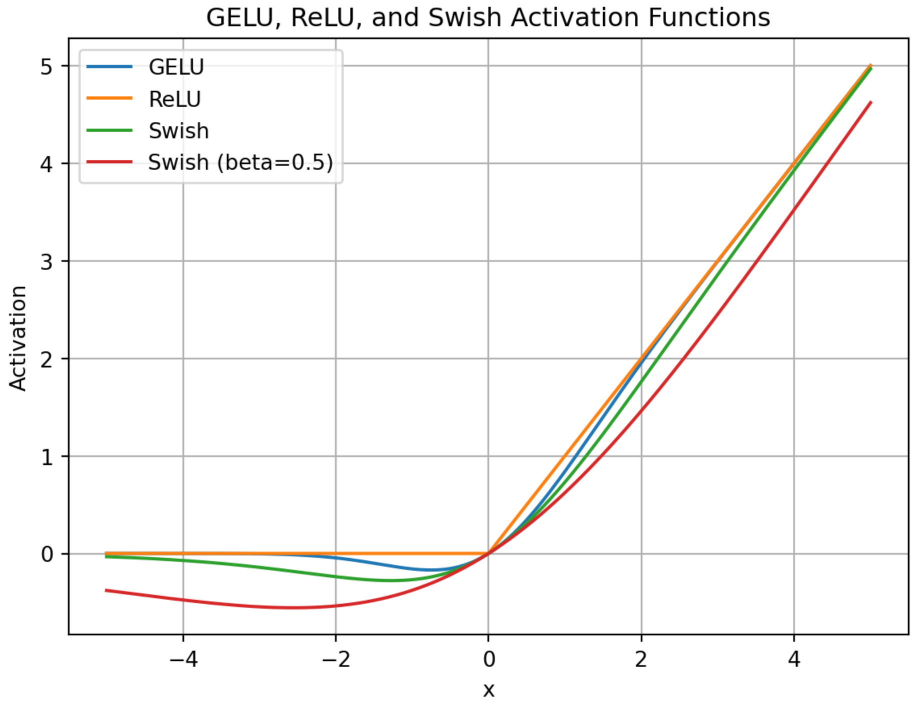
1	-1	0	0	0	0
0	0	2	-2	0	0
0	0	0	0	3	-3

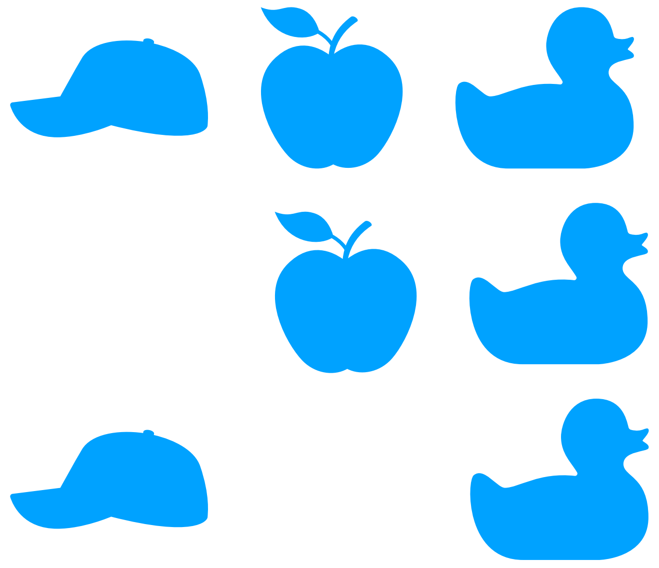
Hidden state

0	0	1
0	1	0
1	0	0
0	1	1
1	0	1
1	1	0
1	1	1

0	0	0	0	3	-3
0	0	2	-2	0	0
1	-1	0	0	0	0
0	0	2	-2	3	-3
1	-1	0	0	3	-3
1	-1	2	-2	0	0
1	-1	2	-2	3	-3

Distributed Representation





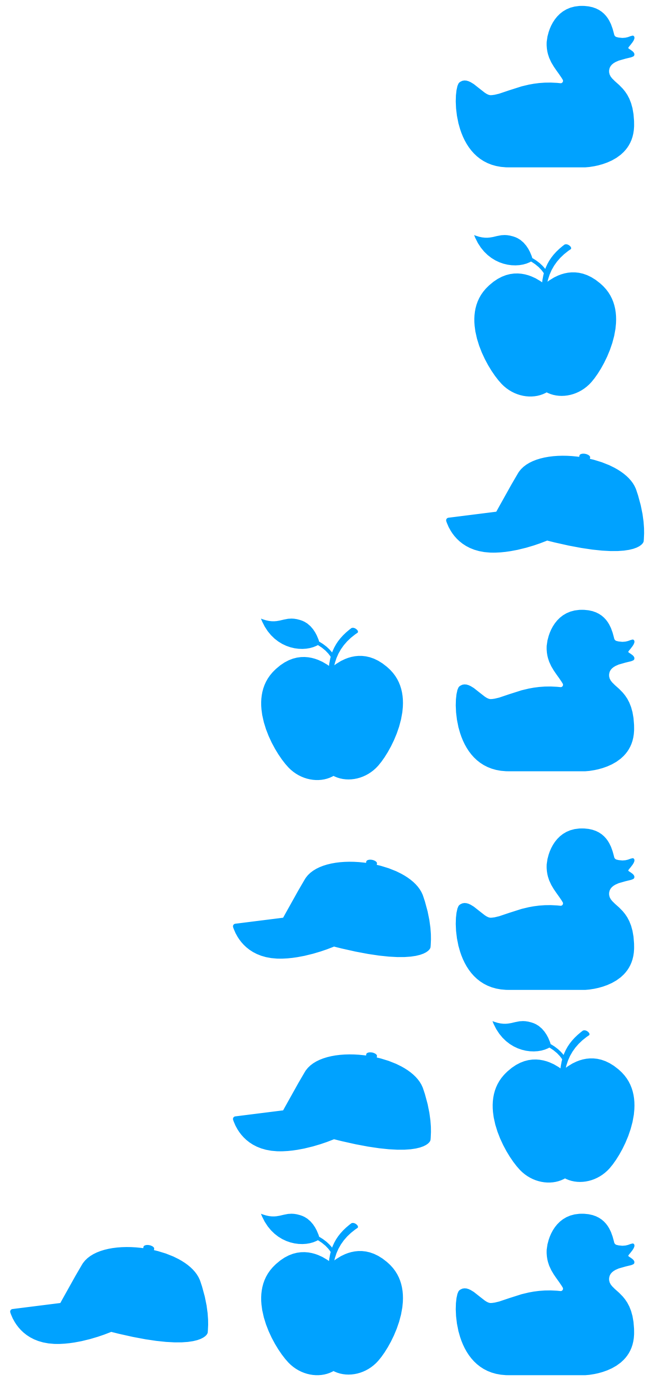
1	-1	-2	-2	3	-3
0	0	2	-2	3	-3
1	-1	0	0	3	-3

A good example showing that the mechanism interpretability highly depends on the small architecture change

Linear probes extract attributes from hidden states

-1	1	1
1	0	-1
1	-1	0
0	1	0
0	0	1
2	-1	-1
1	0	0

0	0	0	0	3	-3
0	0	2	-2	0	0
1	-1	0	0	0	0
0	0	2	-2	3	-3
1	-1	0	0	3	-3
1	-1	2	-2	0	0
1	-1	2	-2	3	-3

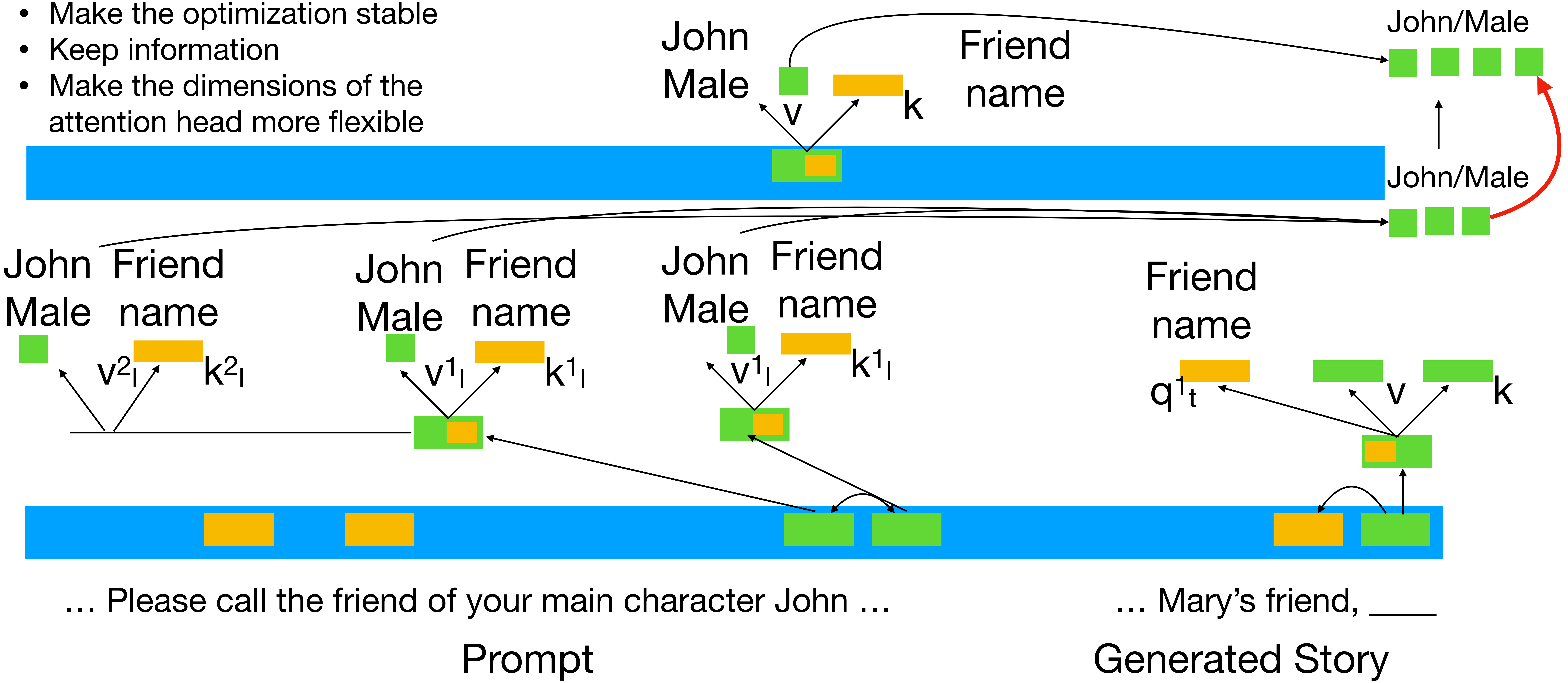


Transformer LM

Haw-Shiuan Chang

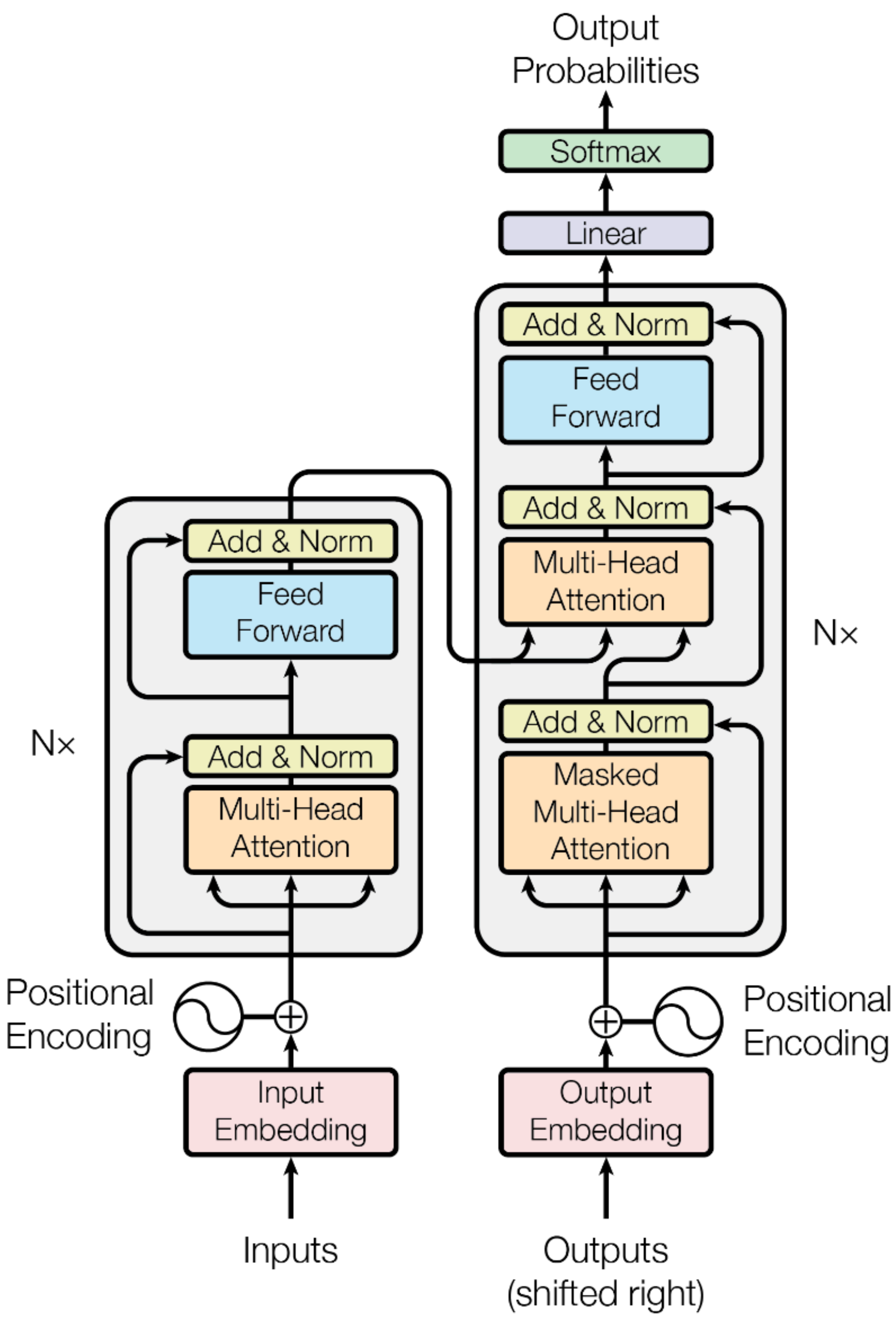
Residual Link

- Make the optimization stable
- Keep information
- Make the dimensions of the attention head more flexible

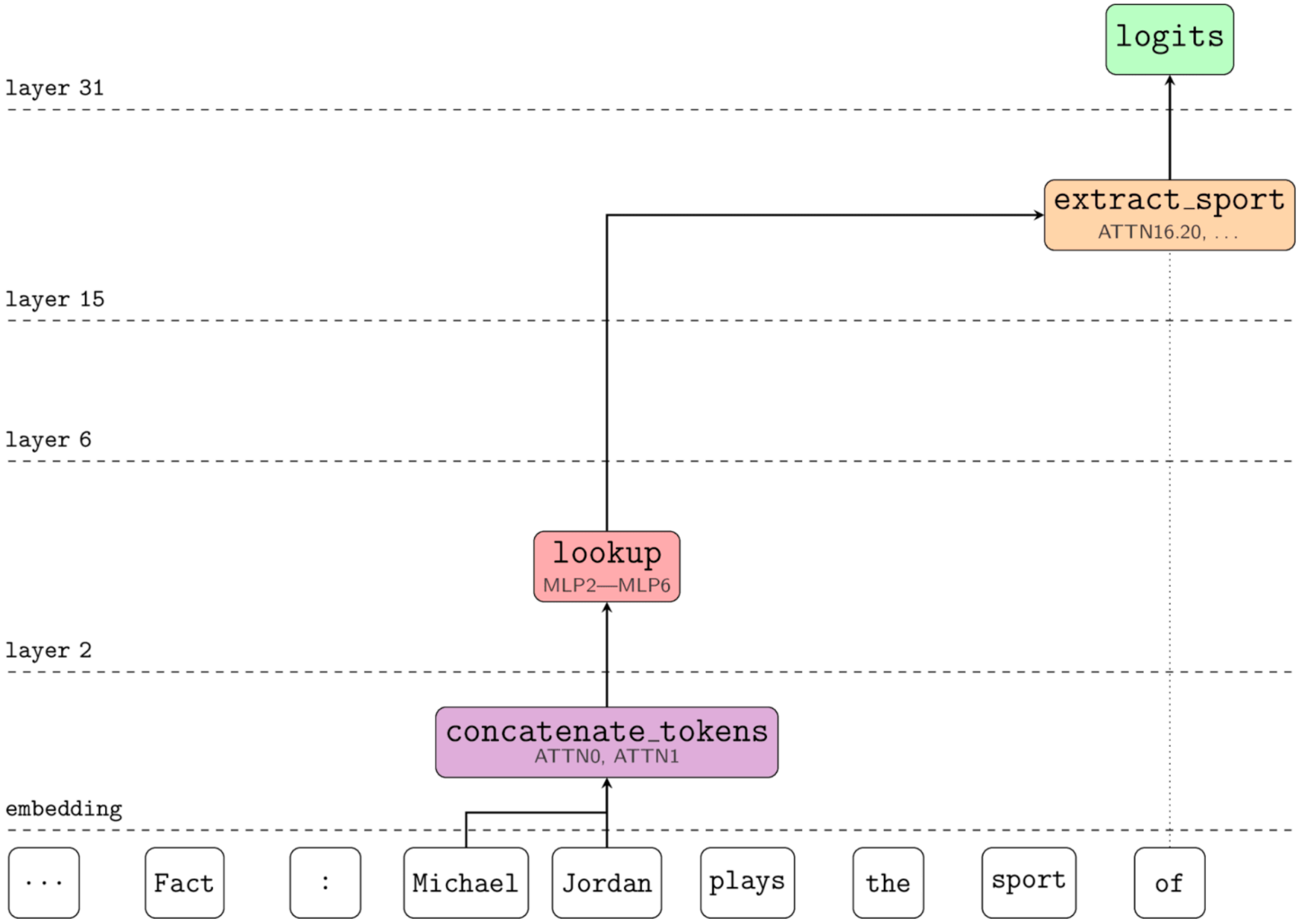


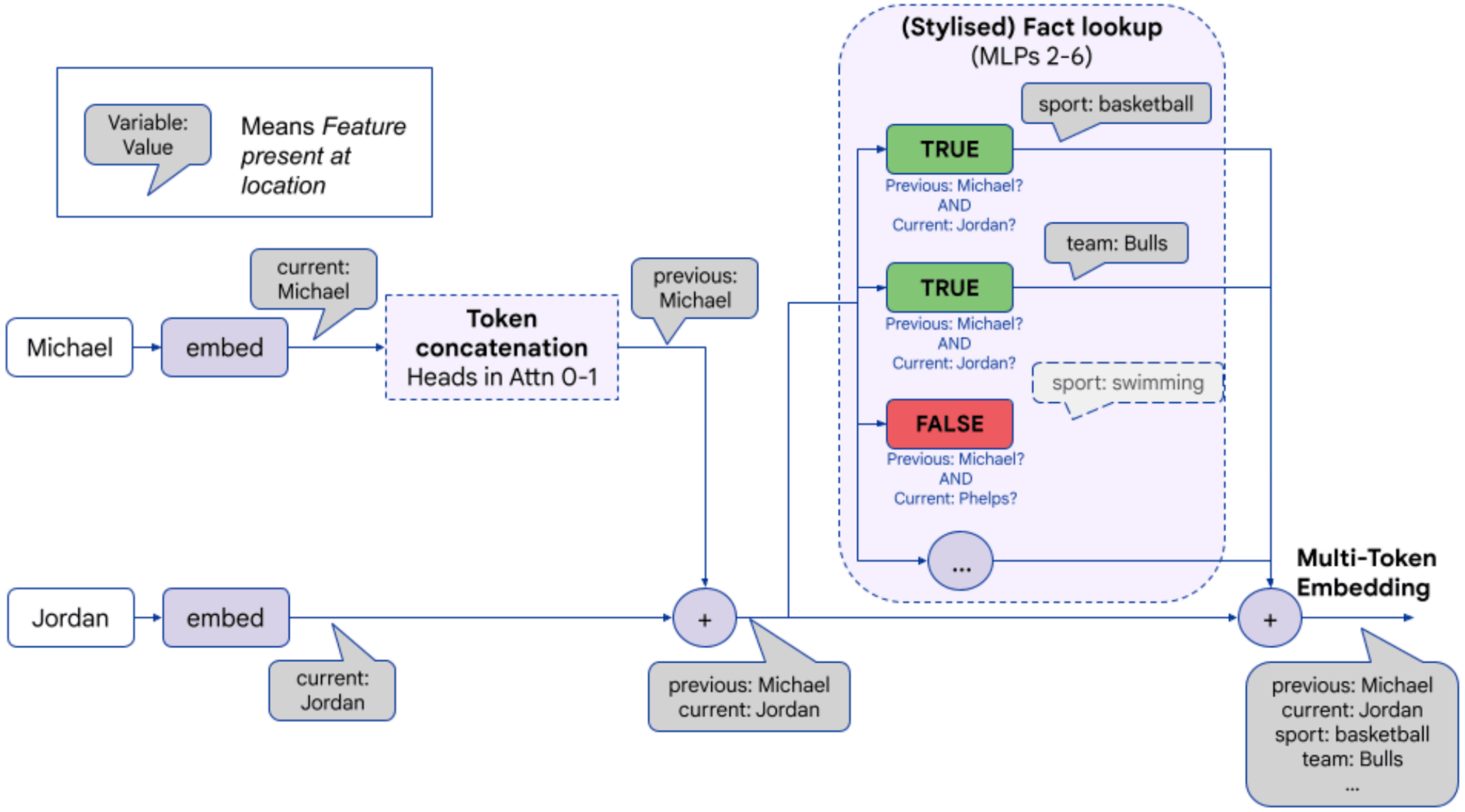
Transformer Architecture

- Self-attention needs ~ 1/3 parameters (GPT-3)
  - Most context processing happens here
  - You can replace this with RNN
- MLP needs ~ 2/3 parameters
  - Most memorization happens here
- Optionally sharing word embeddings
- The architectures are designed for GPU



Example for Attention & MLP

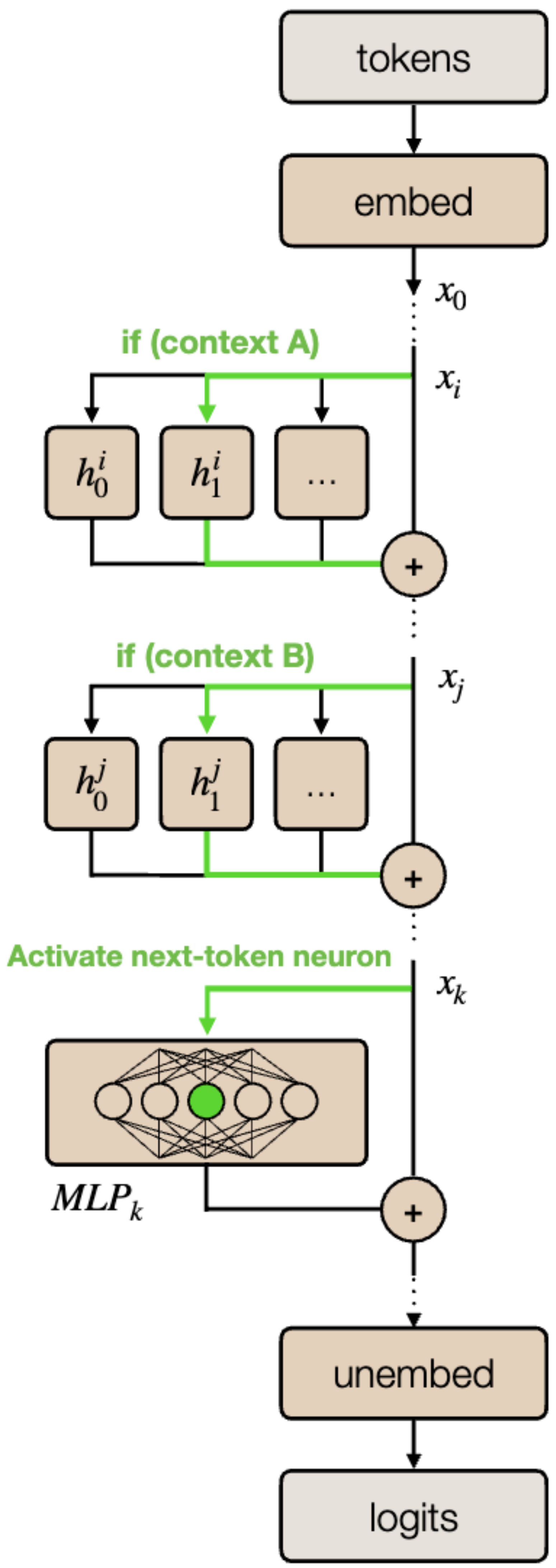




Top deep-learning scientists such as Ilya could probably see these after reading the Transformer paper

https://www.lesswrong.com/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall

Another Perspective

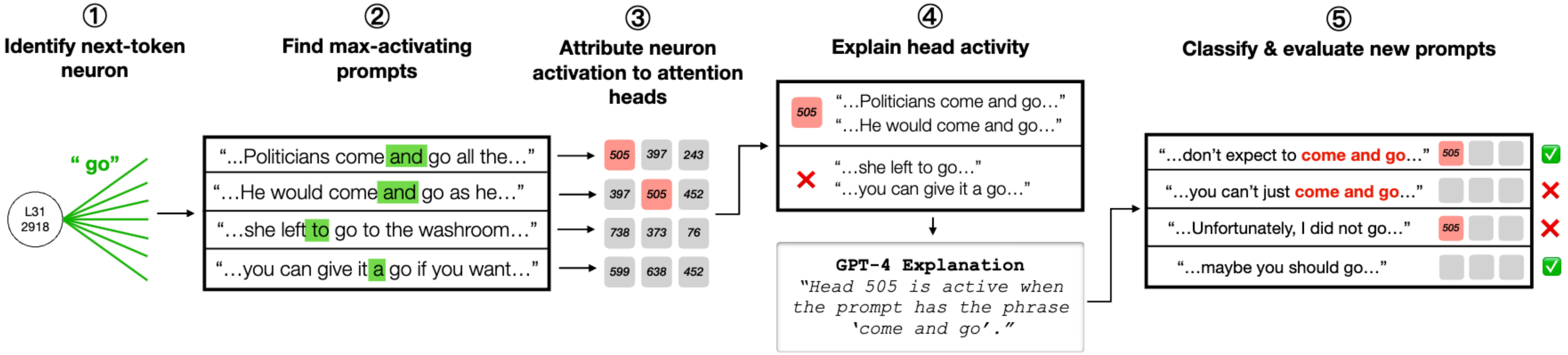


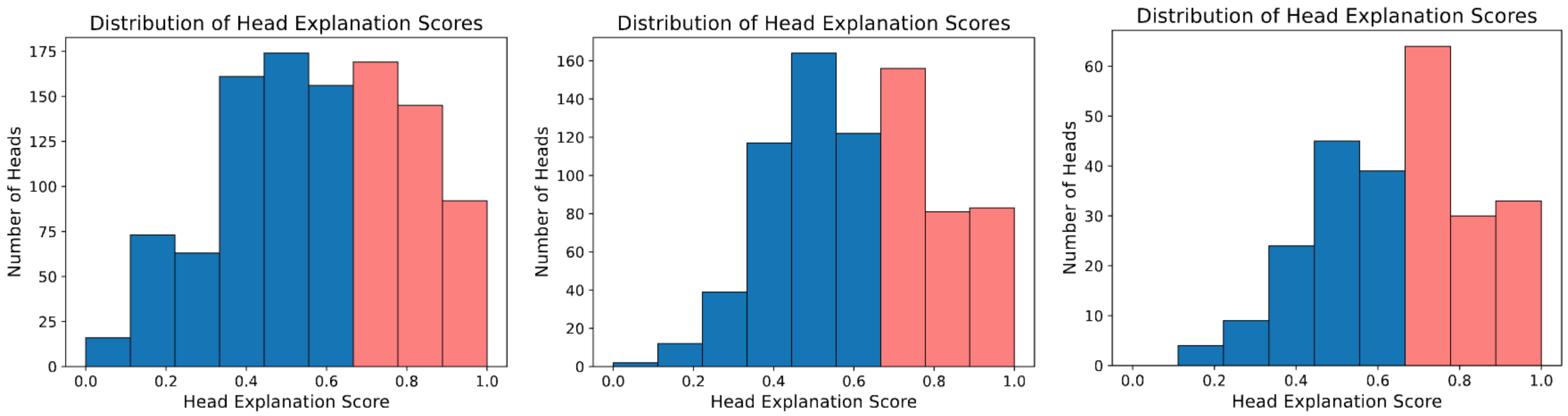
A word can be used differently in multiple contexts.

Different attention heads can pick up on different contexts, and have an output that activates a MLP neuron in later layers.

When the neuron activates, it outputs the embedding of that token into the residual stream, increasing the probability of that token.

The neuron facilitates the model's ability to predict that token over different context or phrases.





(a) GPT-2 Large
Skewness $= -0.132$

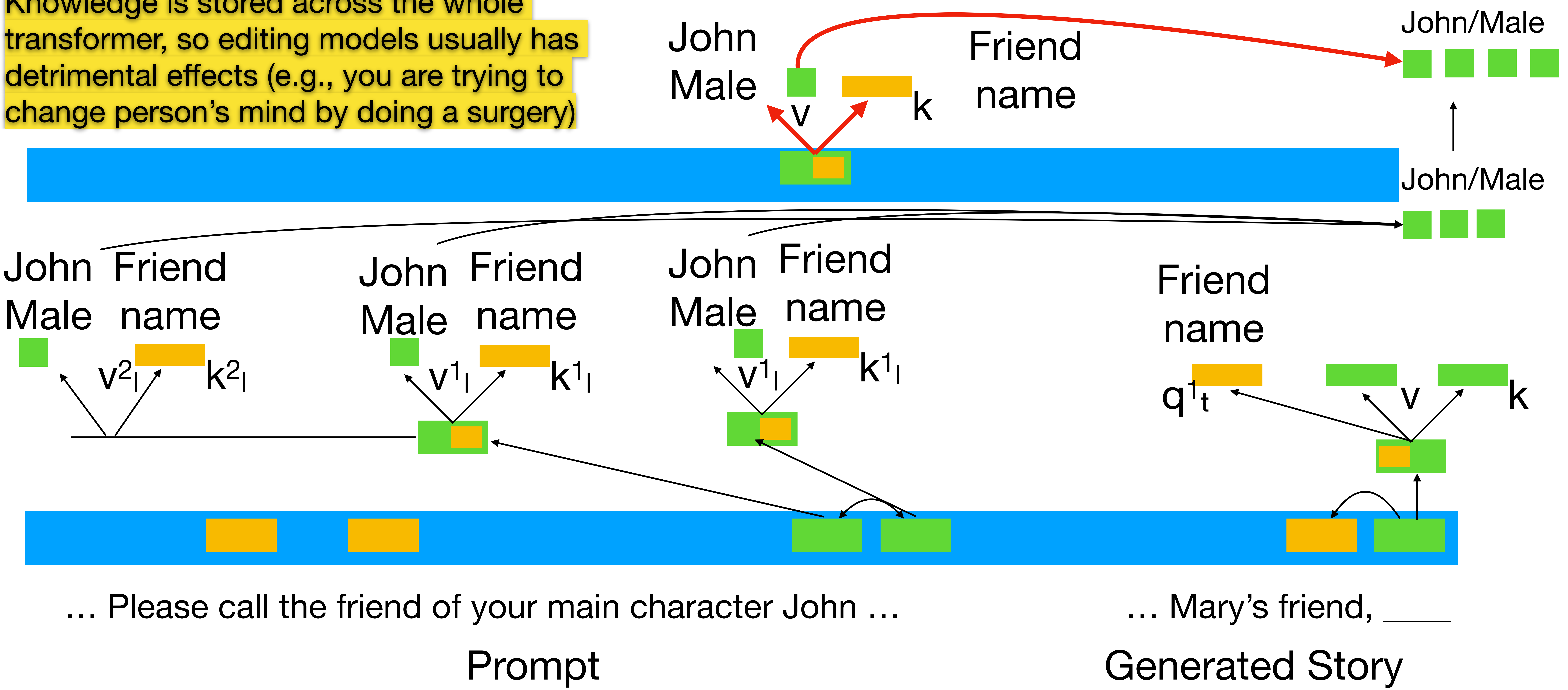
(b) GPT-2 Medium
Skewness $= +0.113$

(c) GPT-2 Small
Skewness $= -0.132$

Interpreting Context Look-ups in Transformers: Investigating Attention-MLP Interactions (https://arxiv.org/pdf/2402.15055v1)

Distributed Representation

Knowledge is stored across the whole transformer, so editing models usually has detrimental effects (e.g., you are trying to change person's mind by doing a surgery)



Demystifying Verbatim Memorization in Large Language Models (https://arxiv.org/abs/2407.17817)

Can Attentions Tell you Token “Importance”?

- Multi-head multi-layer self-attention
  - Unlikely
    - But could be used to compress the KV cache
- Multi-head self-attention
  - Probably not
- Single self-attention
  - Task Dependent

Attention is not important

Attention is important

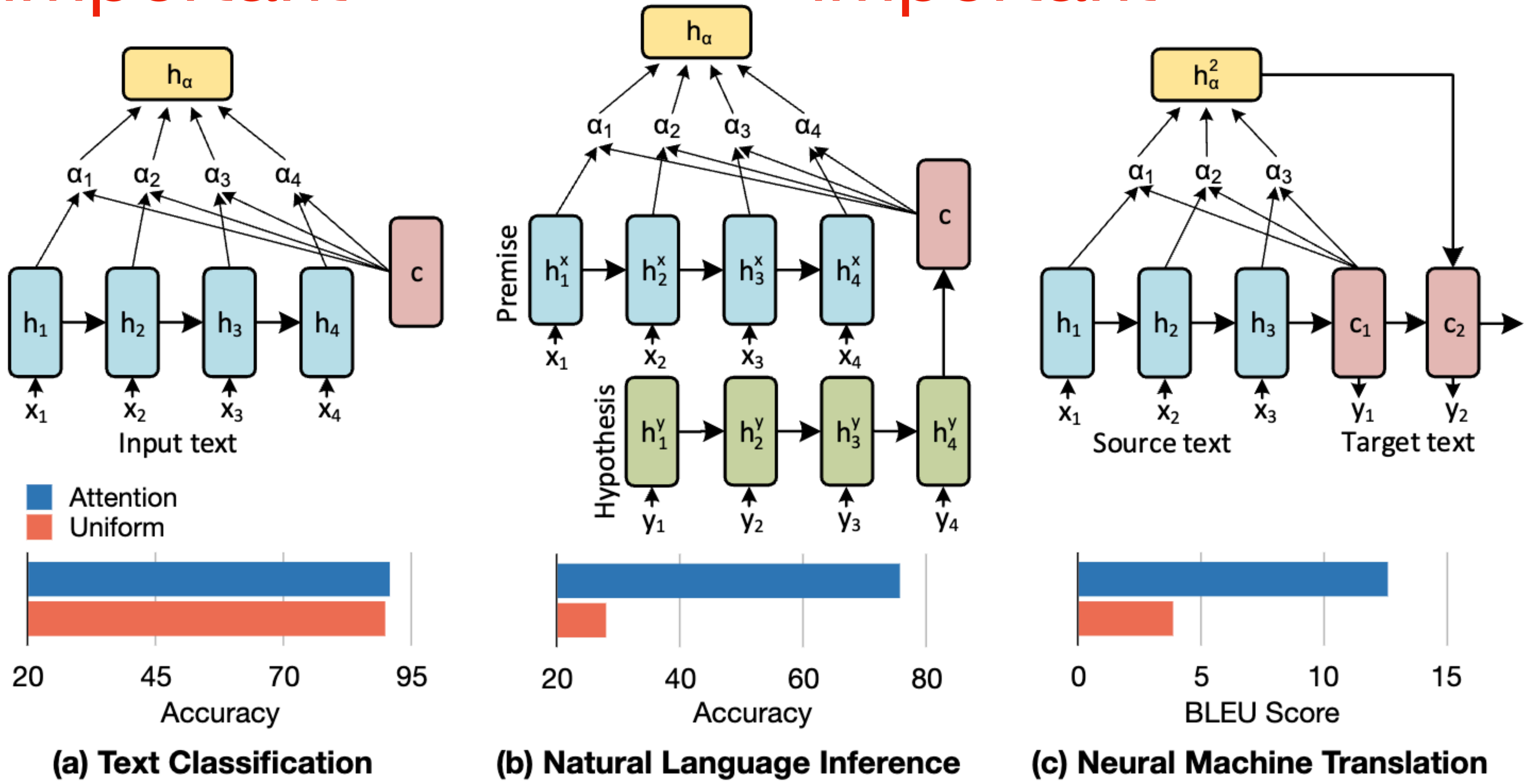


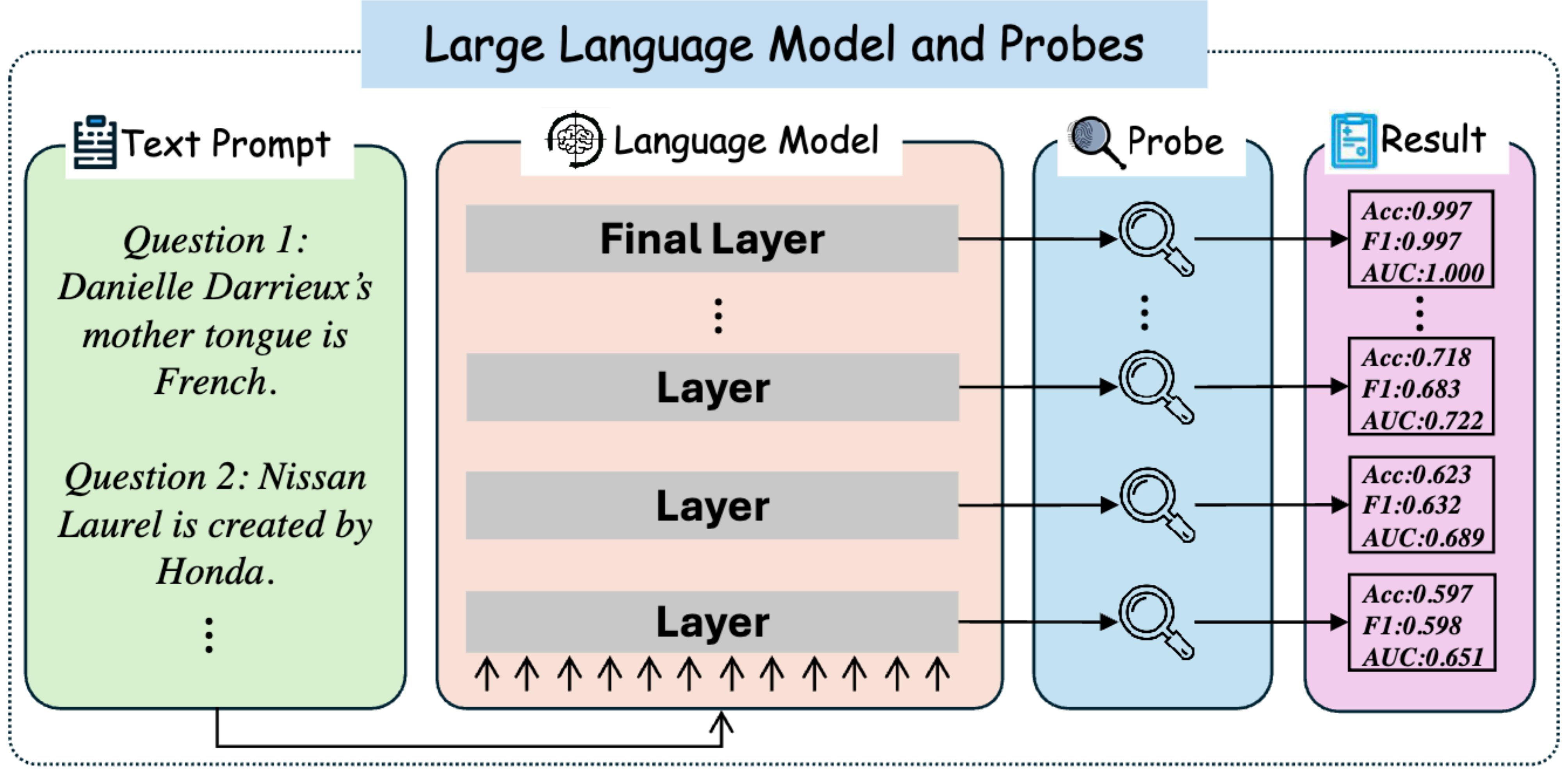
Figure 1: Comparison of performance with and without neural attention on text classification (IMDB), Natural Language Inference tasks (SNLI) and Neural Machine Translation (News Commentary). Here, α and c denote attention weights and context vector respectively. The results show that attention does not substantially effect performance on text classification. However, the same does not hold for other tasks.

Attention Score is not All You Need for Token Importance Indicator in KV Cache Reduction: Value Also Matters (https://aclanthology.org/2024.emnlp-main.1178.pdf)

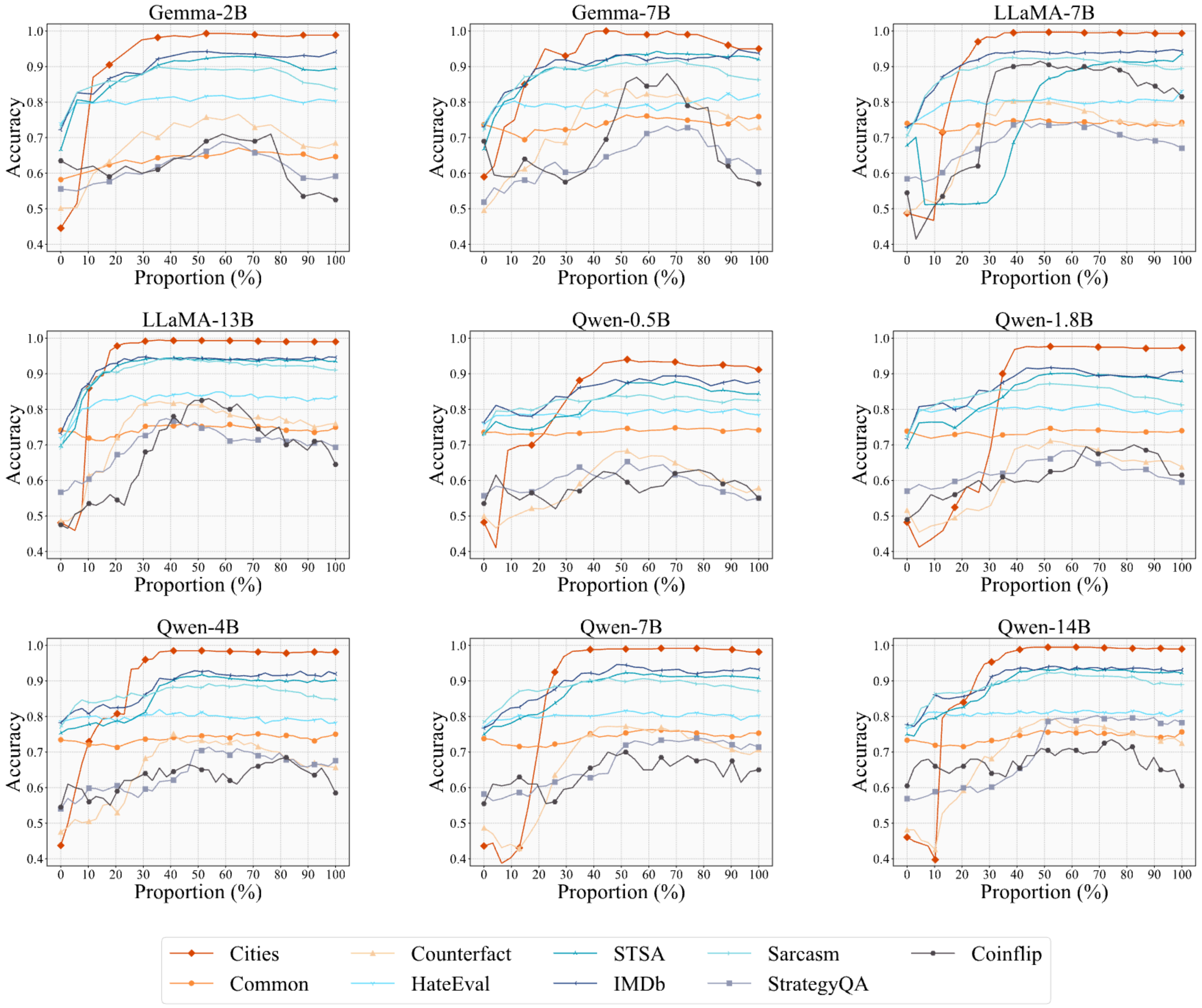
ATTENTION INTERPRETABILITY ACROSS NLP TASKS (https://arxiv.org/pdf/1909.11218)

Where are the Facts Stored?

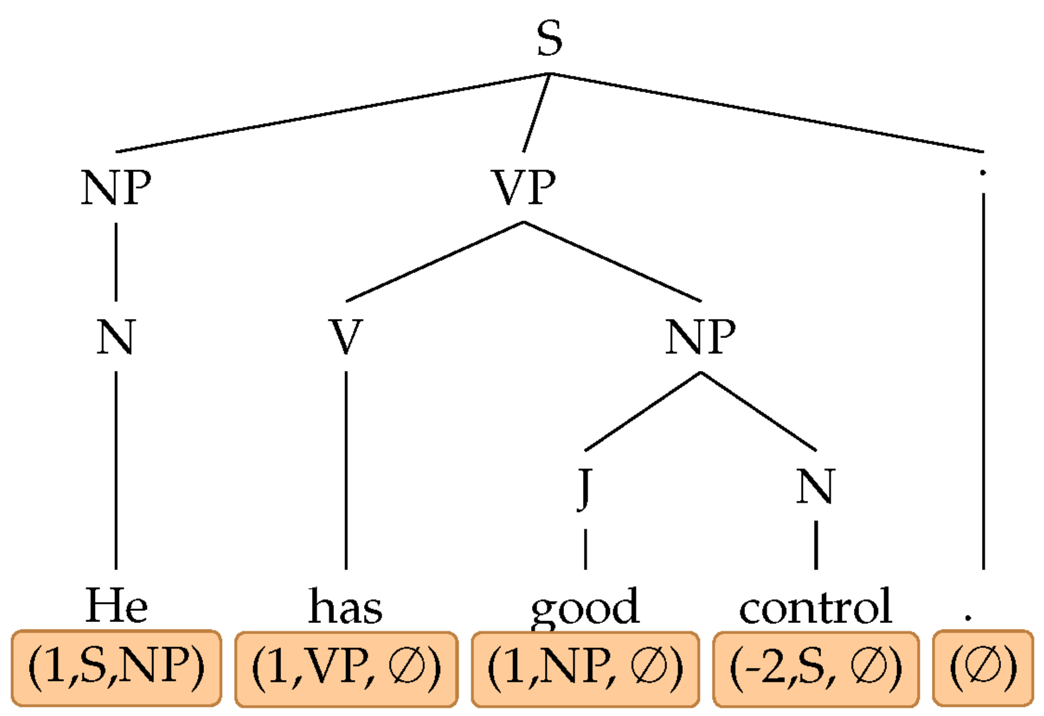
- Facts “tend to” be stored in earlier layers



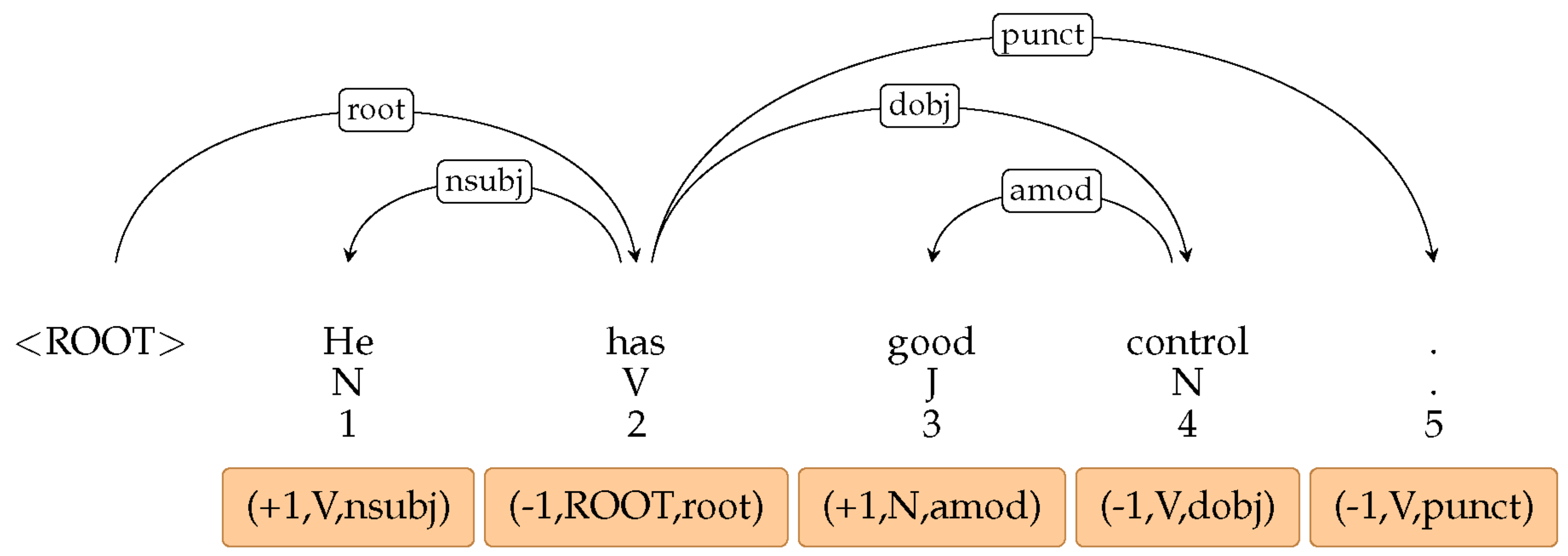
Exploring Concept Depth: How Large Language Models Acquire Knowledge and Concepts at Different Layers? (https://arxiv.org/pdf/2404.07066)



Classic NLP Tasks



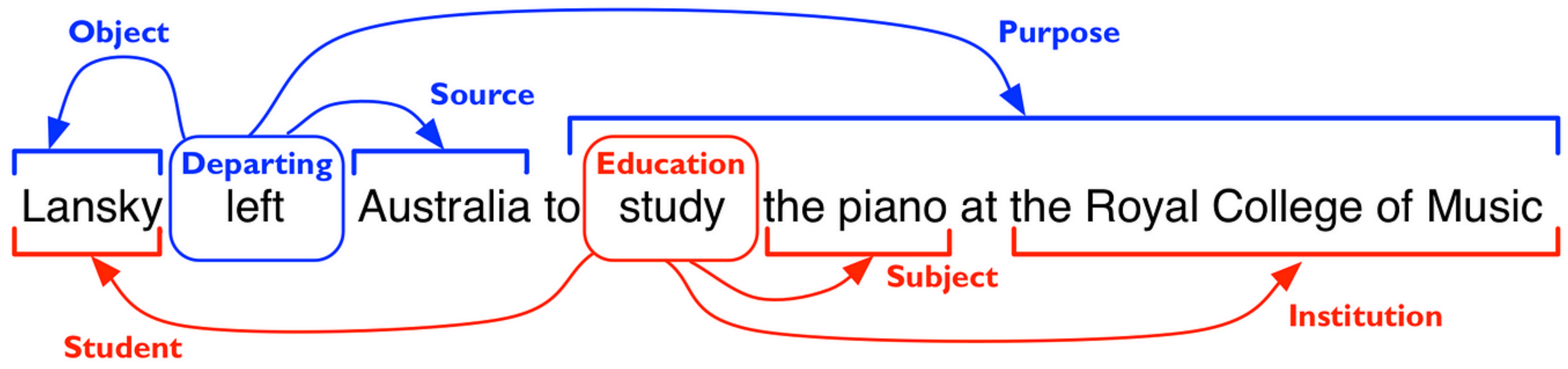
(**a**) A constituency tree



(**b**) A dependency tree

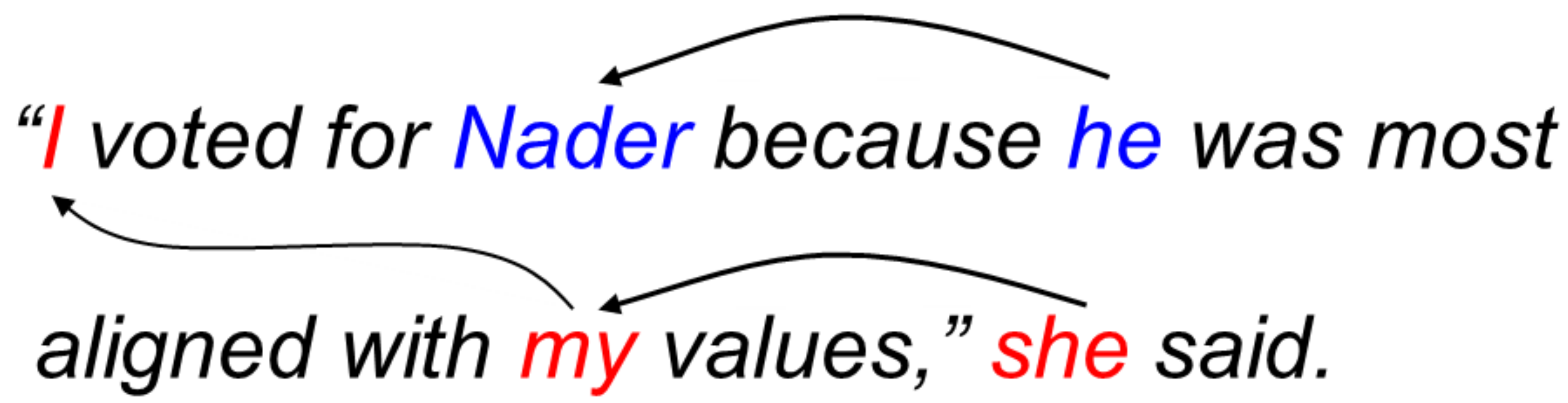
https://www.mdpi.com/2504-3900/21/1/49

SRL



https://medium.com/thedeephub/deciphering-sentences-a-glimpse-into-semantic-role-labeling-with-deep-learning-6b7809bfdcbf

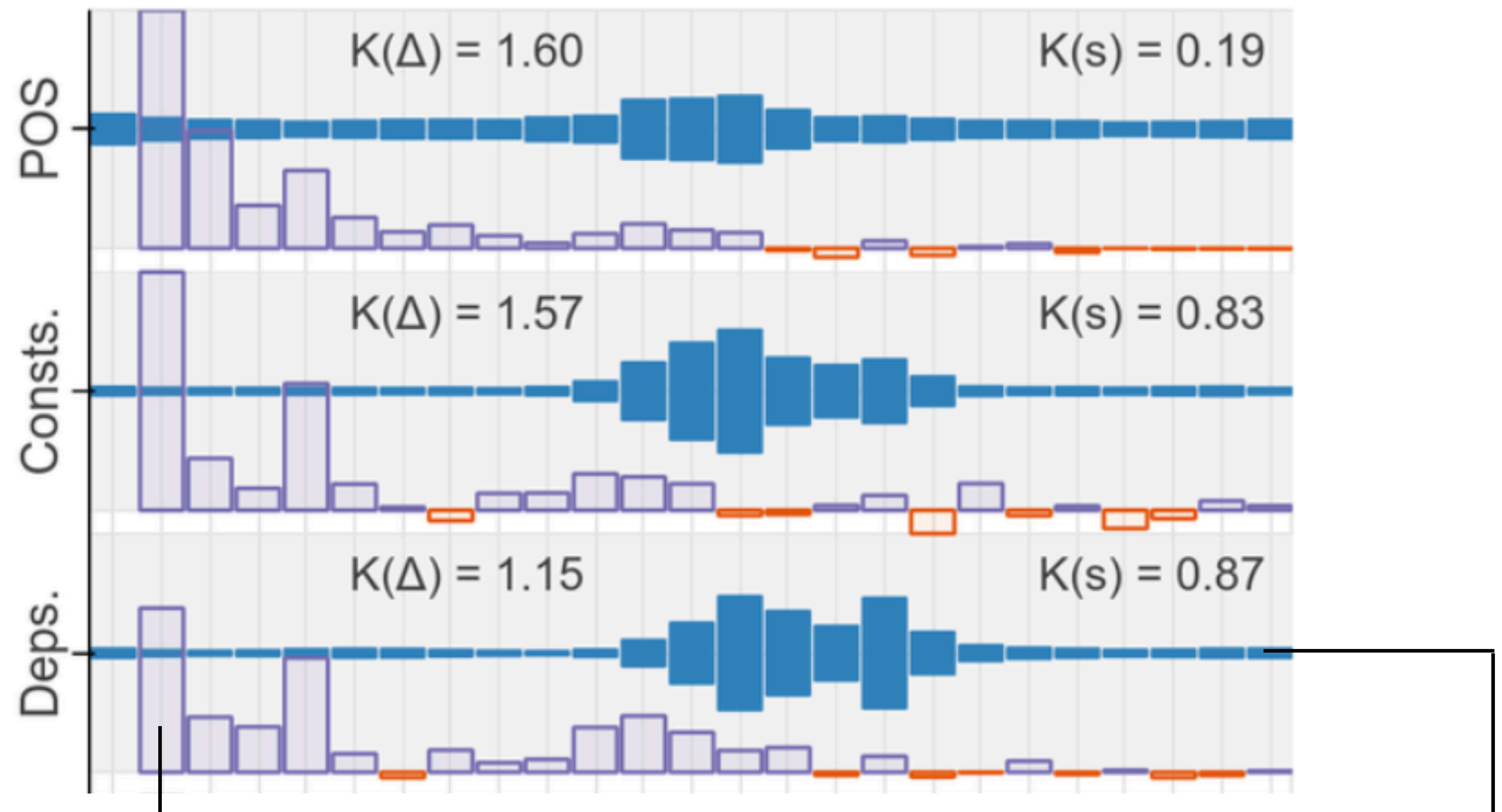
Coref



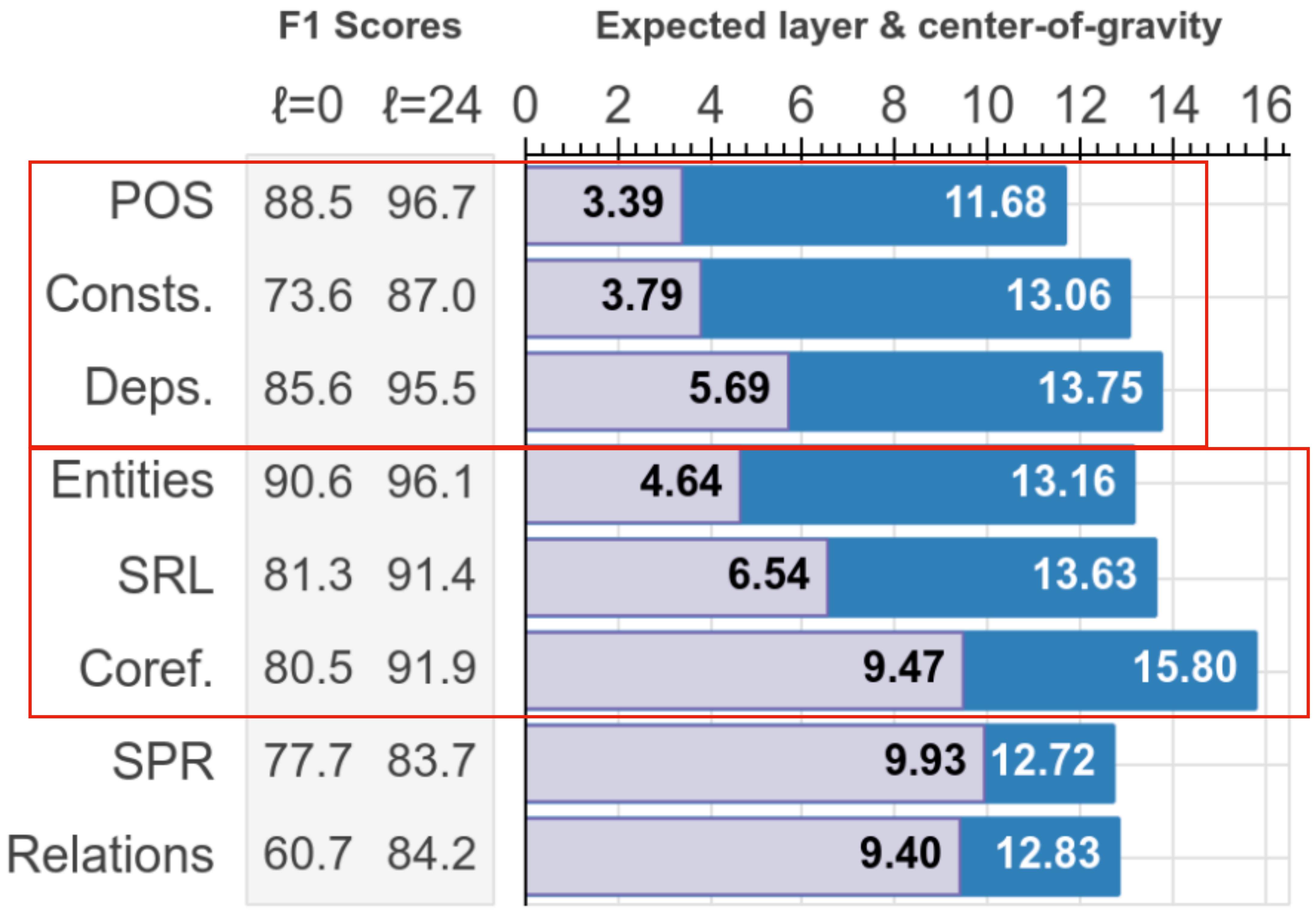
https://nlp.stanford.edu/projects/coref.shtml

What does each Layer Do?

- Higher layers “tend to” handle more semantic information



$$\Delta_{\tau}^{(\ell)} = \mathrm{Score}(P_{\tau}^{(\ell)}) - \mathrm{Score}(P_{\tau}^{(\ell-1)}) \qquad \mathbf{h}_{i,\tau} = \gamma_{\tau} \sum_{\ell=0}^{L} s_{\tau}^{(\ell)} \mathbf{h}_{i}^{(\ell)}$$



BERT Rediscovers the Classical NLP Pipeline
(https://arxiv.org/abs/1905.05950)

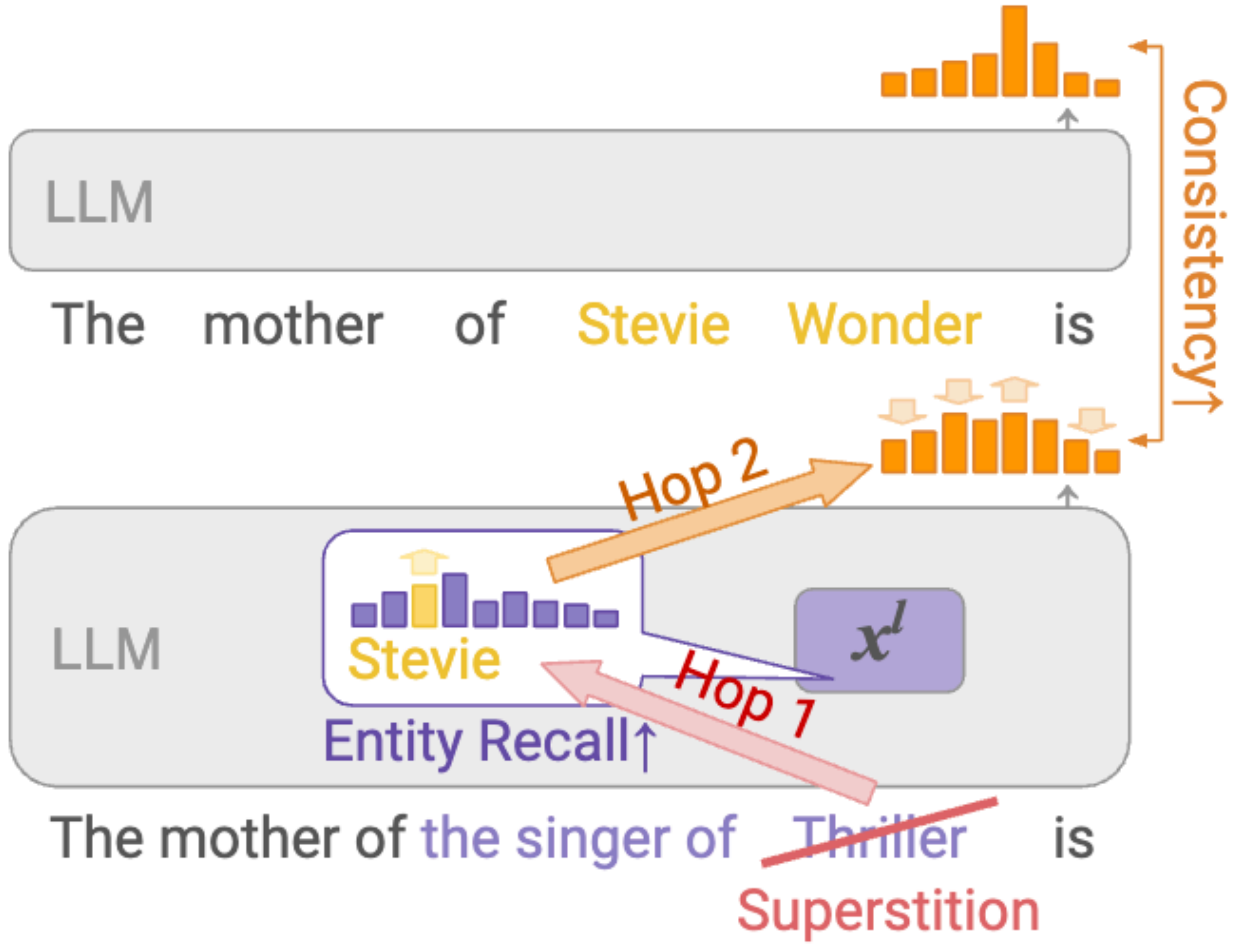


Figure 1: We investigate the latent multi-hop reasoning of LLMs. For the first hop, we change the input prompt to refer to the bridge entity (Stevie Wonder) and check how often it increases the model's internal recall of the bridge entity. For the second hop, we check if increasing this recall causes the model output to be more consistent with respect to what it knows about the bridge entity's attribute (mother of Stevie Wonder).

Do Large Language Models Latently Perform Multi-Hop Reasoning? (https://arxiv.org/pdf/2402.16837)

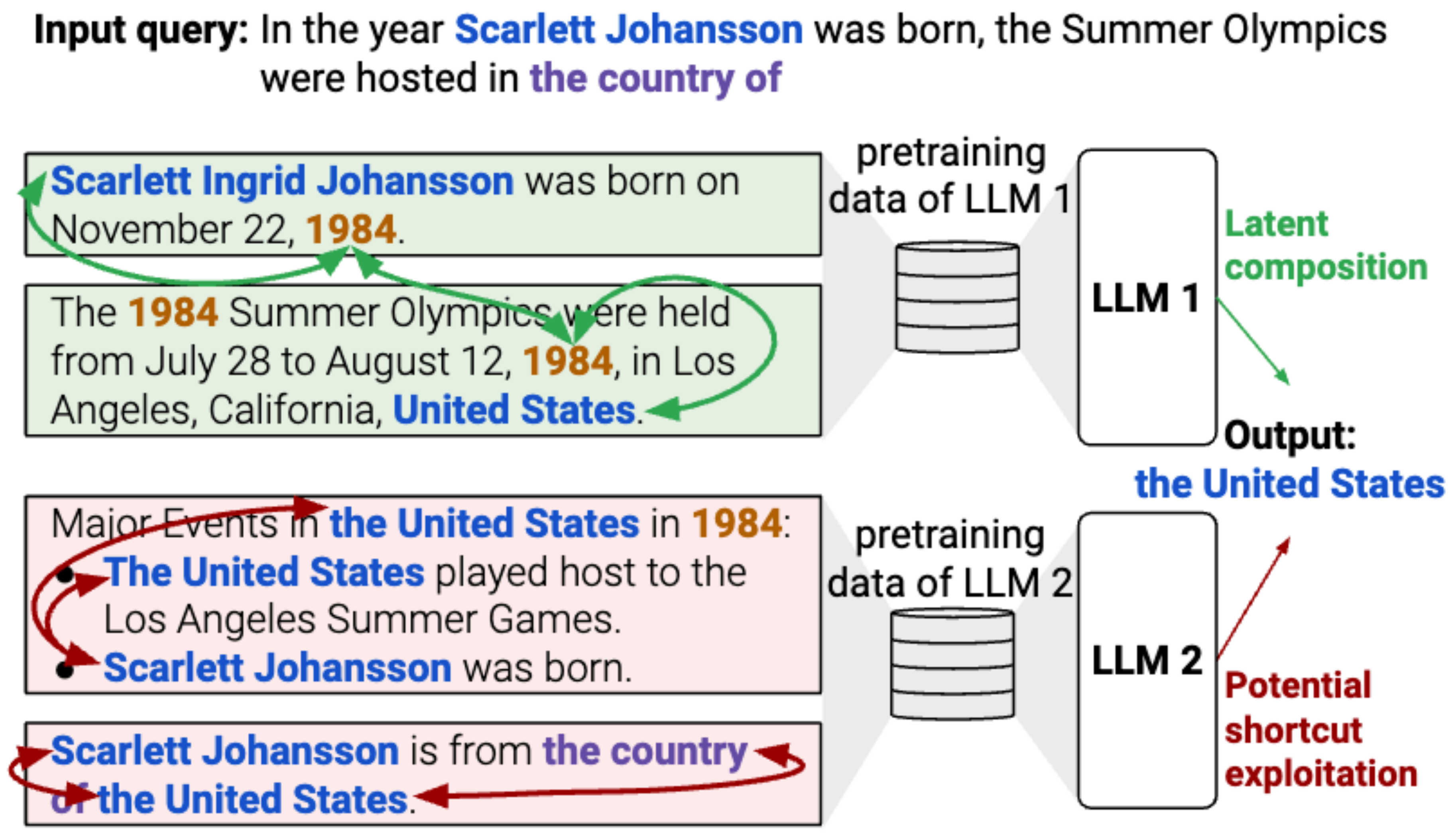
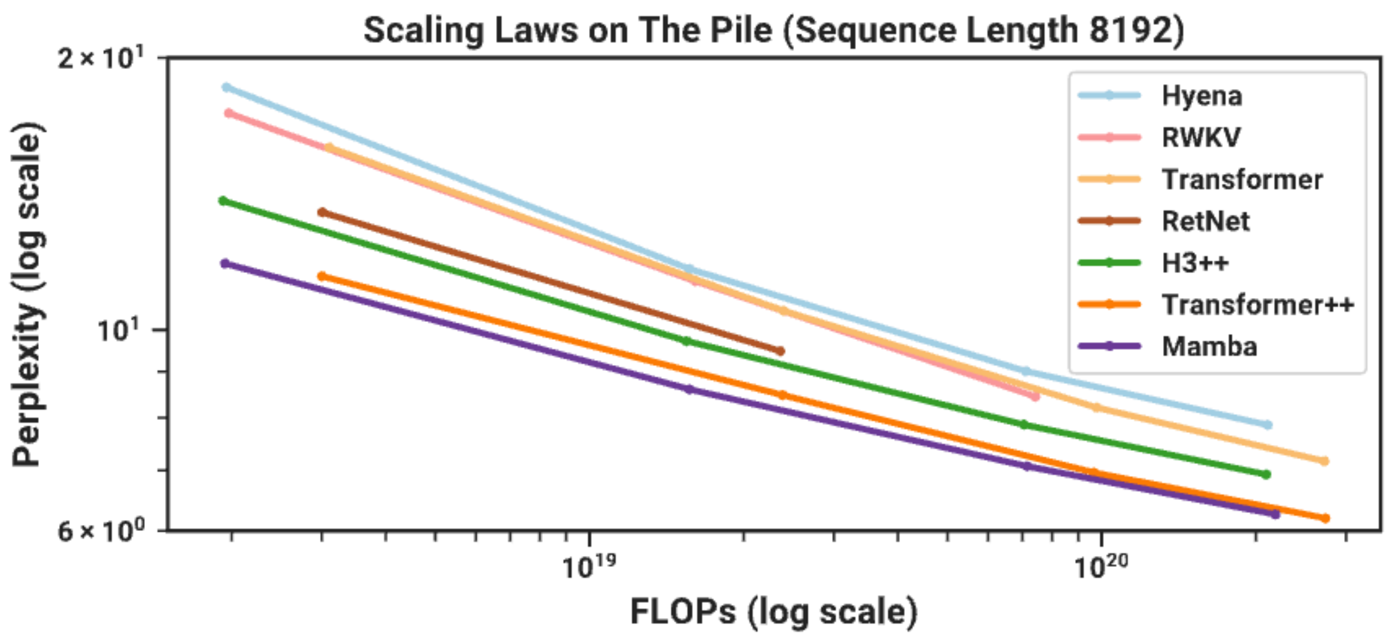
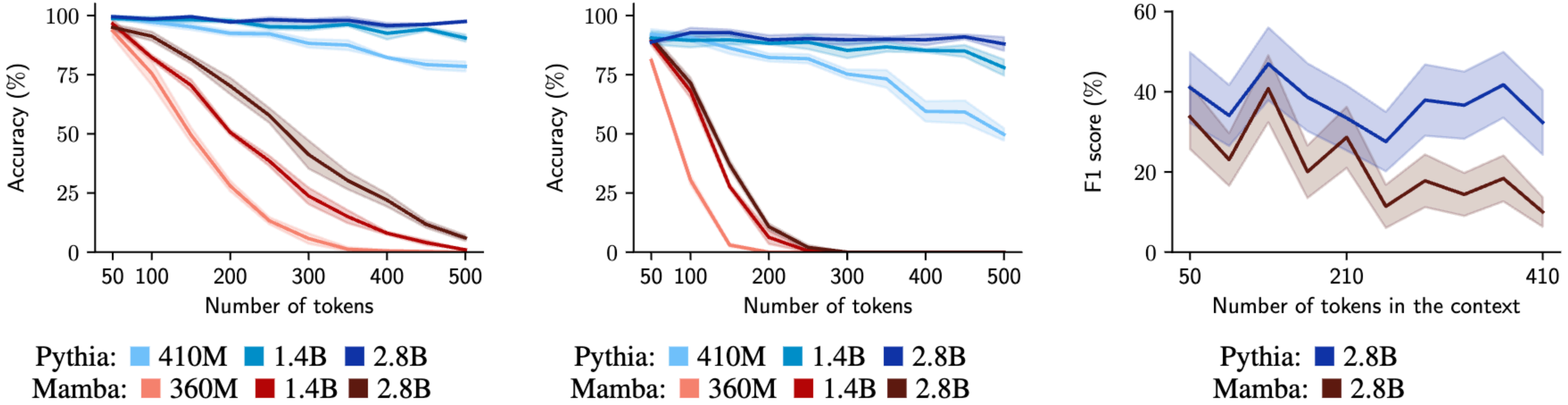


Figure 1: Evaluation of latent multi-hop reasoning should exclude cases where LLMs can bypass the process of latently composing the single-hop facts by exploiting shortcuts. LLMs can develop shortcuts when they frequently encounter the head entity (*"Scarlett Johansson"*) or the relation pattern in the query (*"the country of"*) with the answer entity (*"United States"*). We propose desiderata for shortcut-free evaluation of latent multi-hop reasoning ability to minimize the chance of shortcuts.

Do Large Language Models Perform Latent Multi-Hop Reasoning without Exploiting Shortcuts? (https://arxiv.org/pdf/2411.16679)

Why is Attention Effective?



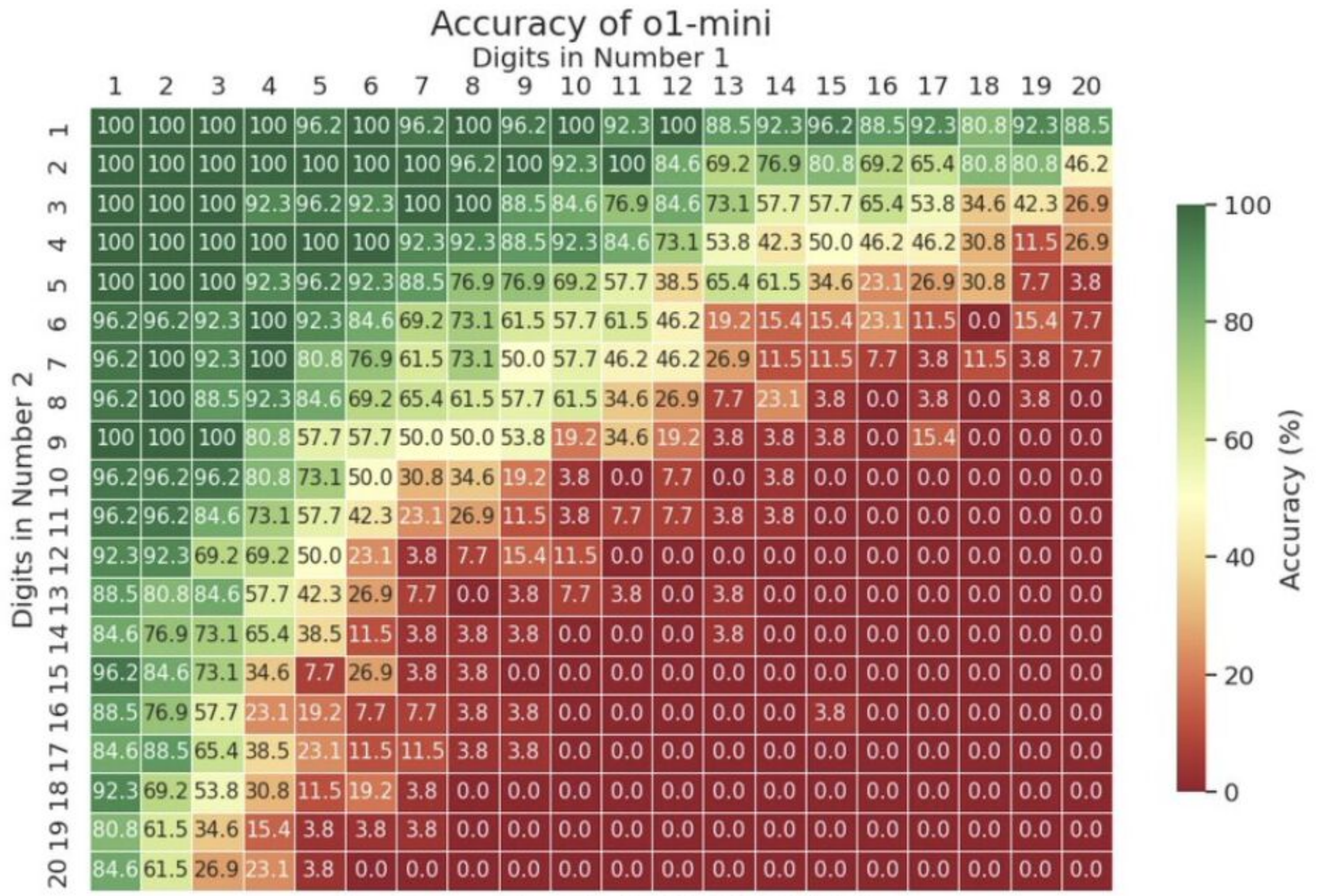


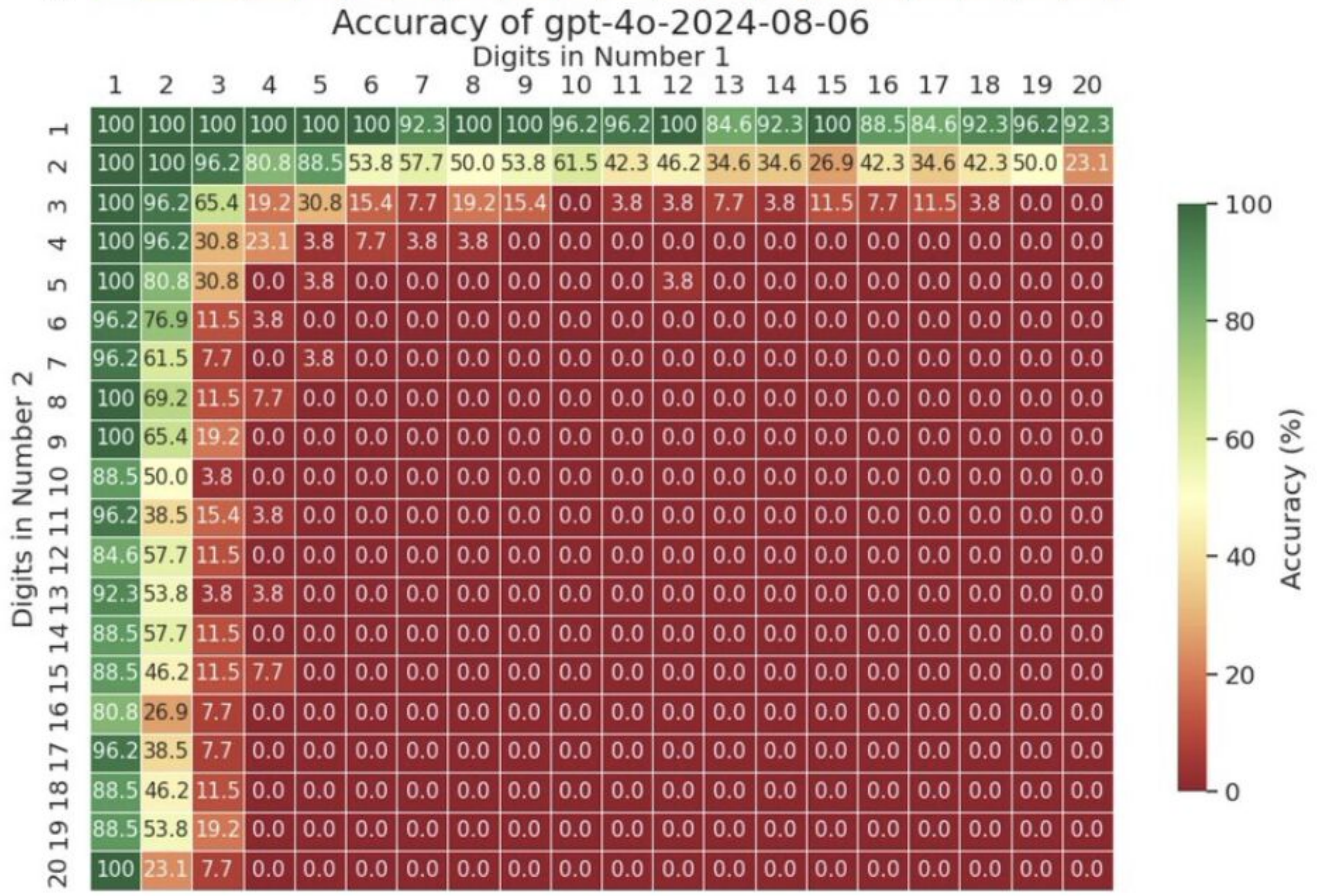
(a) Copy: natural language strings (b) Copy: shuffled strings (c) Question answering (SQUAD)

Figure 7. **(a) Copy: natural language strings.** We compare pretrained models on their ability to copy natural language strings sampled from C4 of varying lengths and report string-level accuracy. The transformer models substantially outperform the GSSMs. **(b) Copy: shuffled strings.** To test whether it mattered that the strings were in natural language, we randomly shuffle the word order of the strings from the previous experiment. We find that this degrades performance, especially for the Mamba models. **(c) Question answering (SQUAD).** We compare Pythia and Mamba on a standard question answering dataset where we bin the dataset based on the length of the context paragraph. We find that Mamba performance decays more quickly with the length of the context.

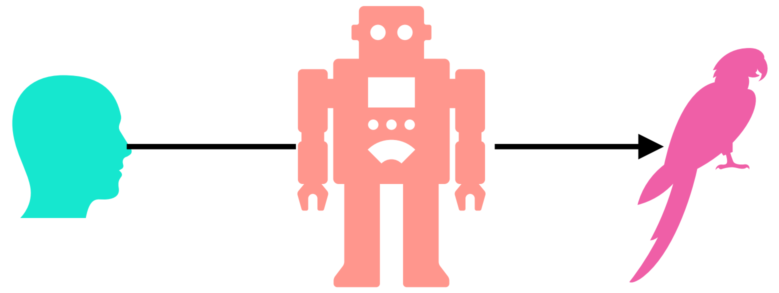
Repeat After Me: Transformers are Better than State Space Models at Copying (https://arxiv.org/abs/2402.01032)

Arithmetic Computation Limitation





- Transformer architecture design
  - MLP retrieves the relevant memory from training
  - Self-attention retrieves and merges the memories MLP retrieves
- Transformers are not designed to handle logic



https://x.com/yuntiandeng/status/1889704768135905332

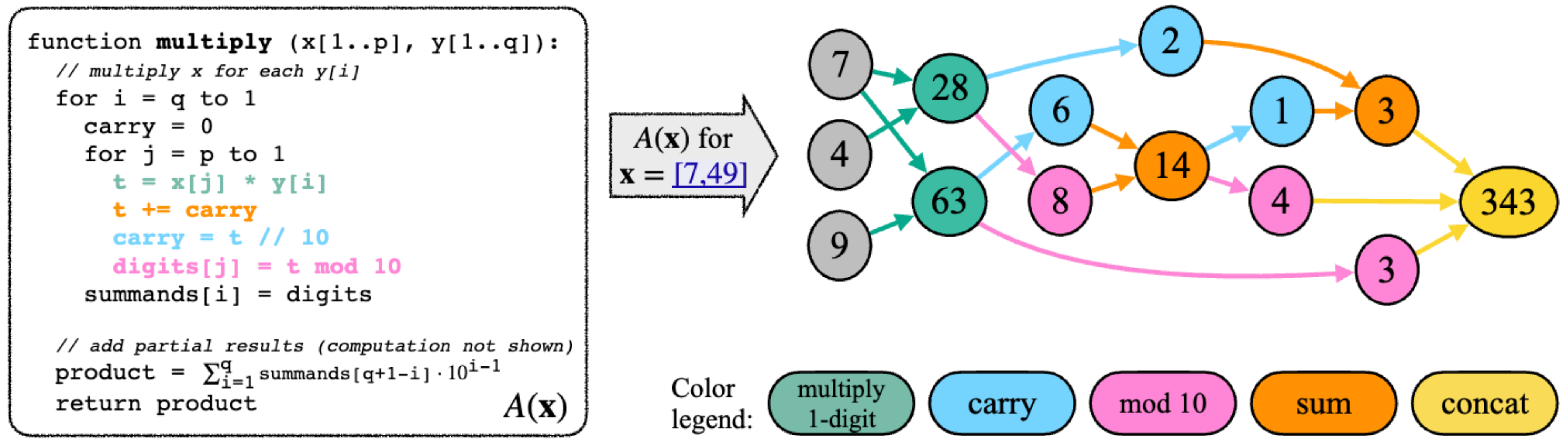


Figure 1: Transformation of an algorithm A to its computational graph $G_{A(\mathbf{x})}$. The depicted example is of long-form multiplication algorithm A, for inputs $\mathbf{x} = [7, 49]$ (i.e. computing 7×49).

Faith and Fate: Limits of Transformers on Compositionality (https://proceedings.neurips.cc/paper_files/paper/2023/file/deb3c28192f979302c157cb653c15e90-Paper-Conference.pdf)

Arithmetic Computation Limitation

5 + 3 = 8

353 + 237 = 590

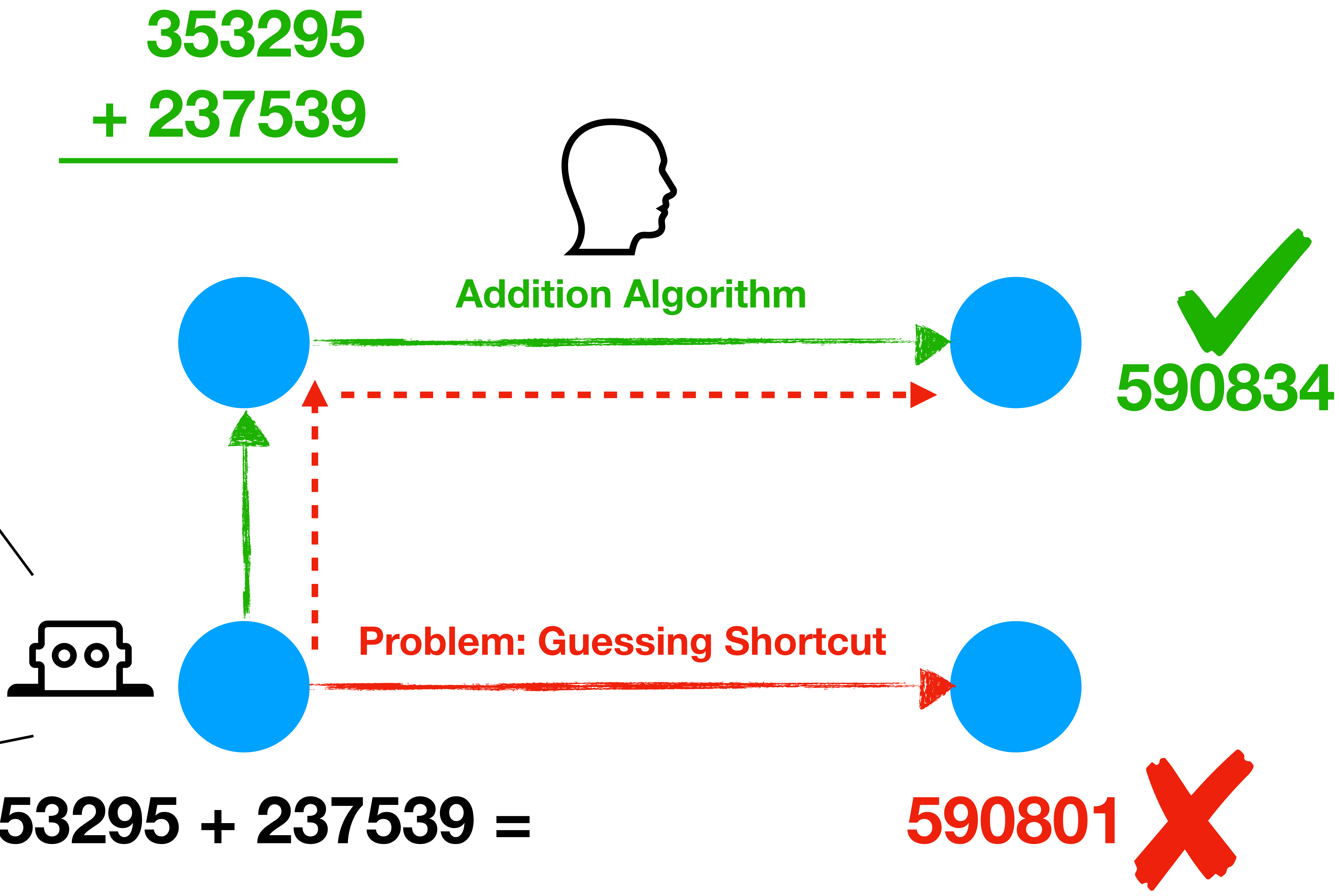
32 + 75 = 107

353219 + 237582 = 590801

353295 + 227439 = 580734

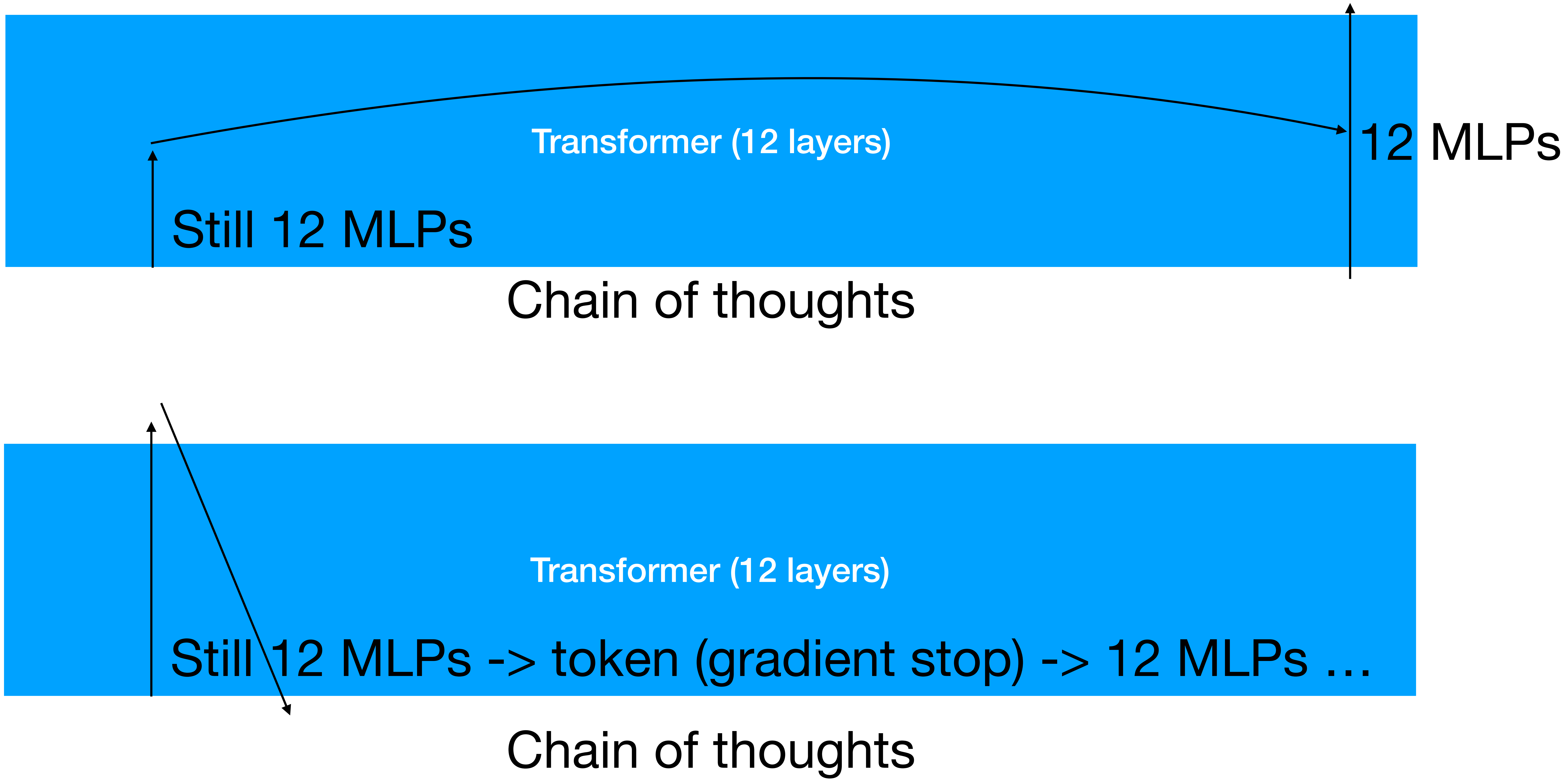
35329 + 274 = 35603

5329 + 2743 = 8072



Math Exercise

Limited Reasoning Chain Length



Midterm Example Questions

Q1: RNN represents one sequence using one embedding, but Transformer also represents one sequence using one embedding. Why does Transformer mitigate the embedding bottleneck problem?

Q2: Which model is more expensive to train? RNN or Self-attention?

Self-Attention vs RNN vs CNN

- CNN
  - Pros
    - Super Fast
  - Cons
    - Bad long distance

- RNN
  - Pros
    - Long reasoning chain length
    - Low inference memory
    - Fast Inference
    - Good at positional Information
  - Cons
    - Copy difficulty
    - Simple operation or slow parallel training

- Self-attention
  - Pros
    - Long Distance Copy
    - Parallel Training
    - Good at handling a set
  - Cons
    - Limited reasoning chain length
    - High inference memory
    - Relatively Slow Inference
      - $O(n^2)$

Sequence Processing

Feedforward NN

RNN

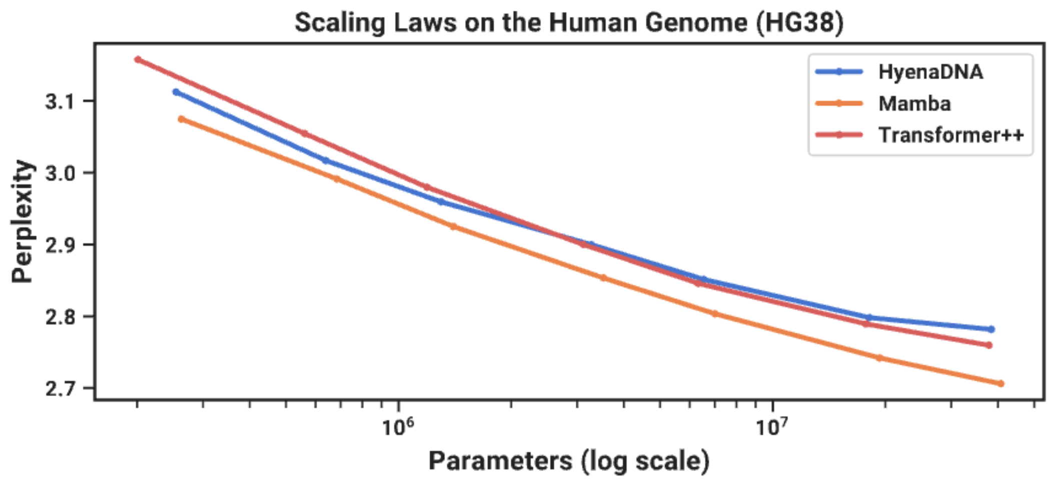
CNN

Transformer

Computation Cost — More →

Long Dependency — More →

Position Information — Less →



Mamba: Linear-Time Sequence Modeling with Selective State Spaces (https://arxiv.org/pdf/2312.00752)