Transformer LM

Haw-Shiuan Chang

Deadlines

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

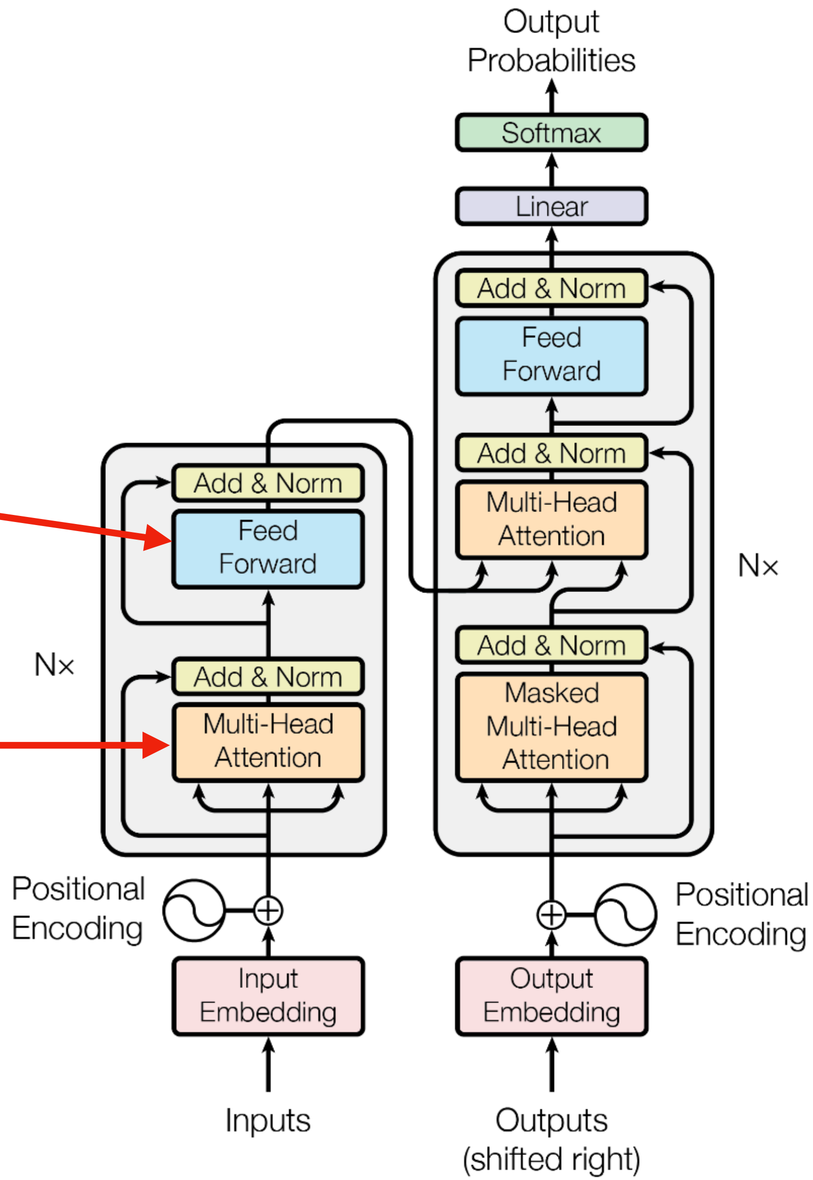
- **3/3: Quiz 2 due**
  - If I cannot finish teaching cross-attention today, I will extend the deadline. If you don't see the announcement on Piazza, the deadline is 3/3.
- **3/7**: Project proposals due
  - If you have some project ideas, you can go to TA office hours to seek some feedback.
  - If your group members don't contribute to the project, please let us know
- **3/14**: HW 1 due

Thanks for the Feedback

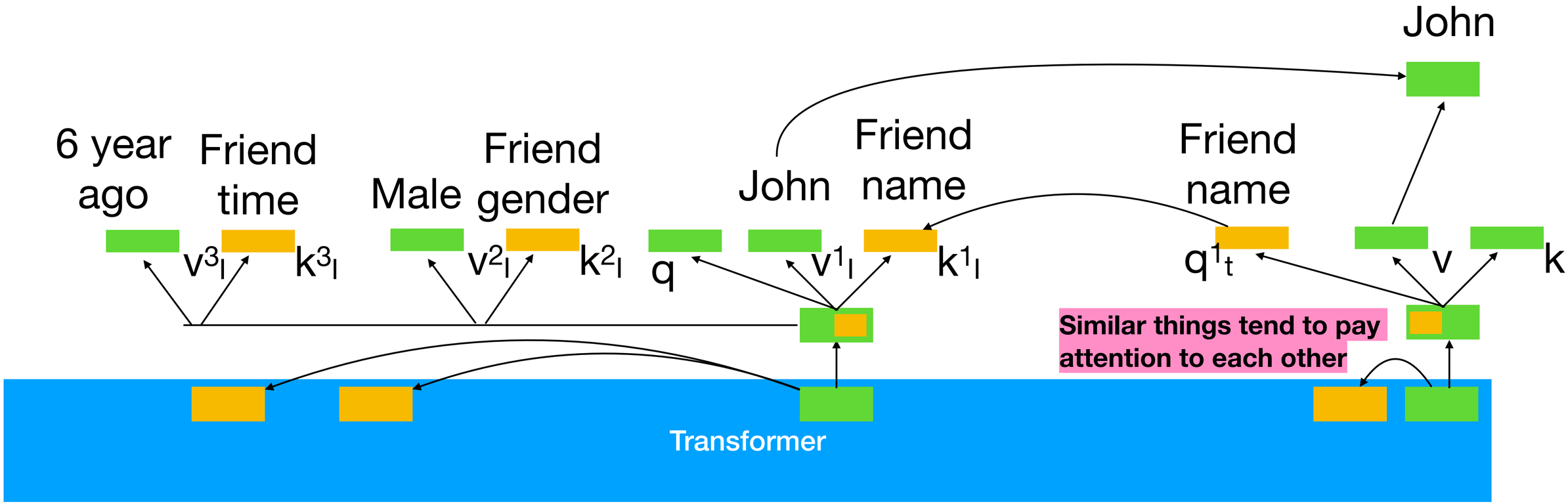
- My last lecture is too difficult for a beginner (~60% too hard, too fast, and too unclear.)
  - You can focus on the conclusions or intuitive examples
    - More likely to appear in the midterm
  - Try to understand those interpretation methods and what really happened in practice
    - Especially if you want to do related research or know where the conclusions come from or when these conclusions hold
- In the future, I will try to
  - explain things more slowly
  - go through easier materials first and leave difficult ones to the end
- Let’s review the conclusions together

Transformer Architecture

- MLP needs ~ 2/3 parameters (GPT-3)
  - Most memorization happens here
- Self-attention needs ~ 1/3 parameters (GPT-3)
  - Most context processing happens here



Self-Attention Illustrative Example



… Please call the friend of your main character John …

your main character met her friend 6 years ago

Prompt

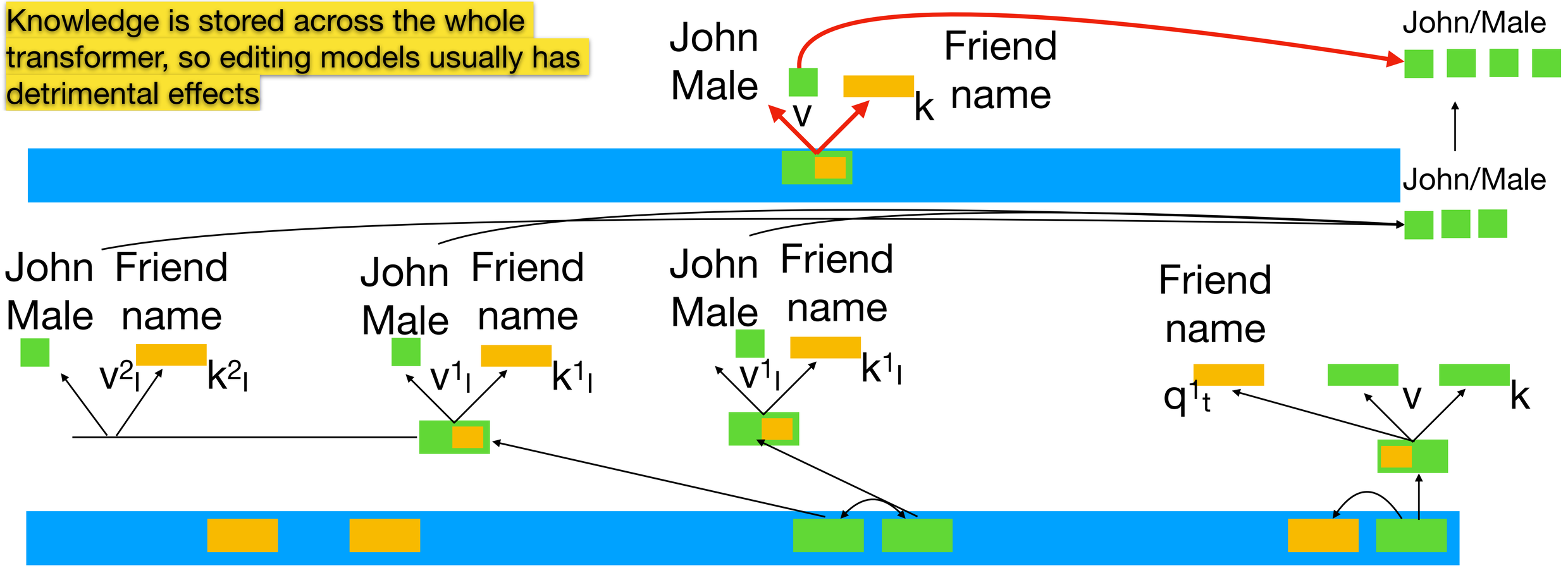
… Mary's friend, ____

Generated Story

However, most heads are not very interpretable in practice.

Distributed Representation

Knowledge is stored across the whole transformer, so editing models usually has detrimental effects



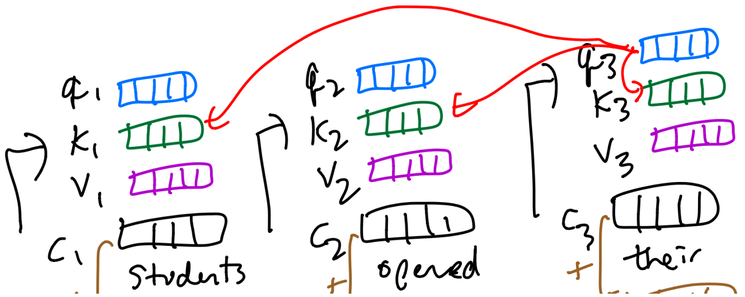
… Please call the friend of your main character John …

Prompt

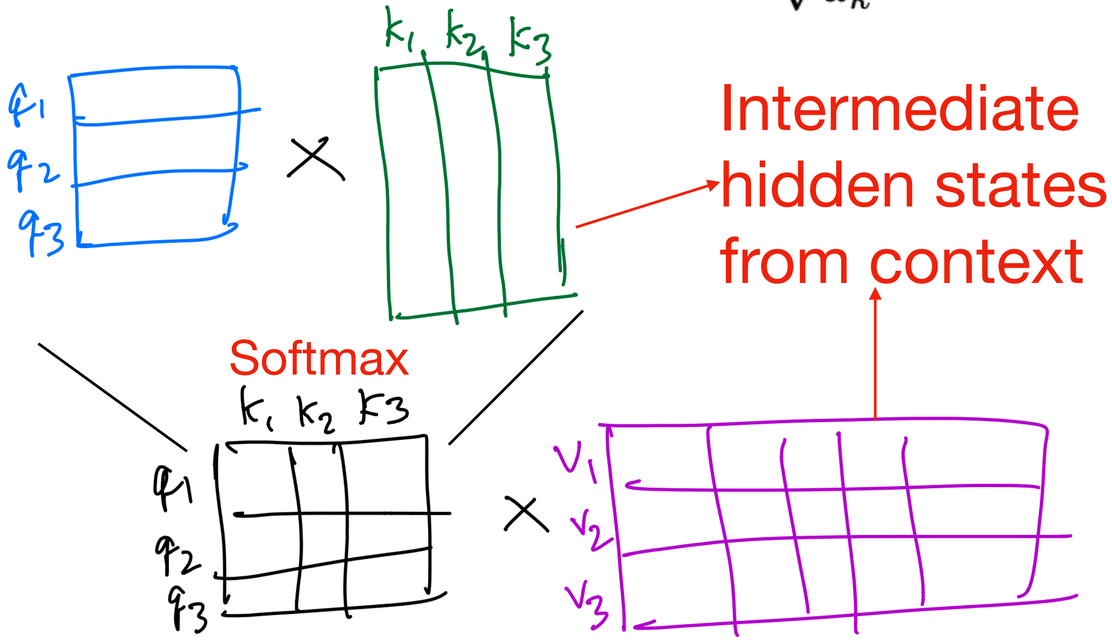
… Mary's friend, ____

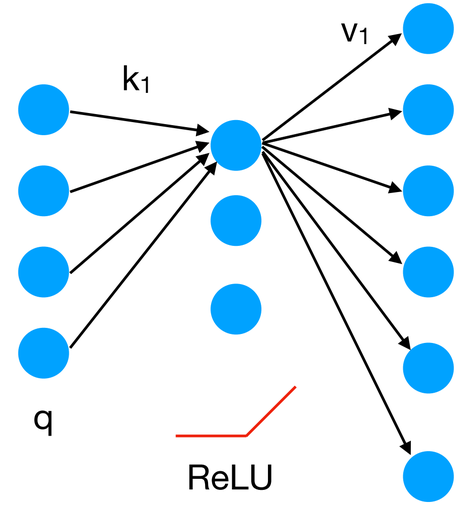
Generated Story

Self-attention vs MLP

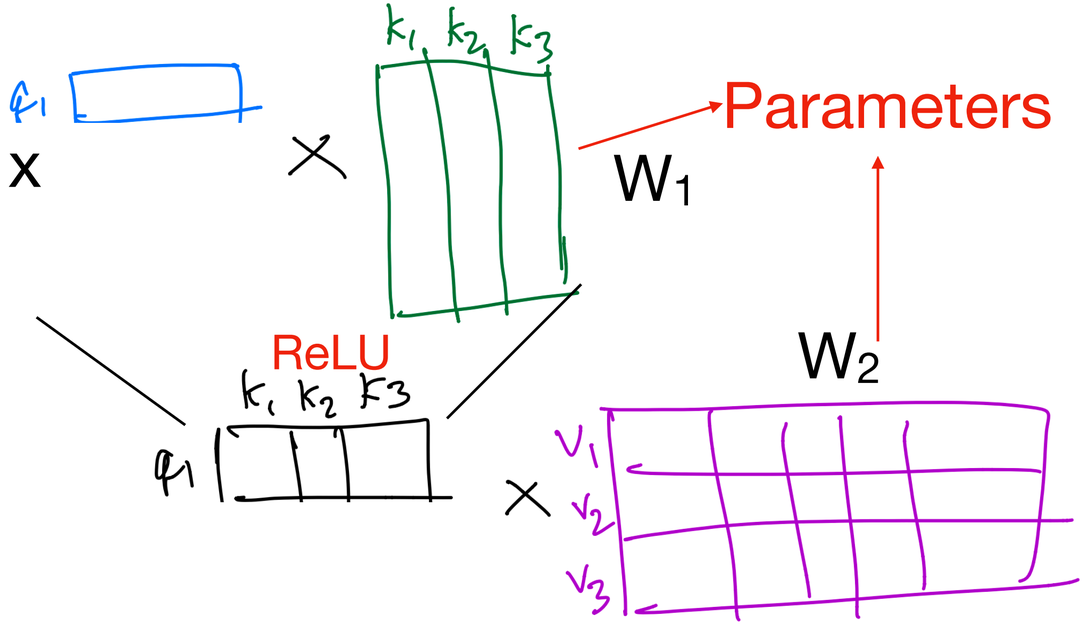


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

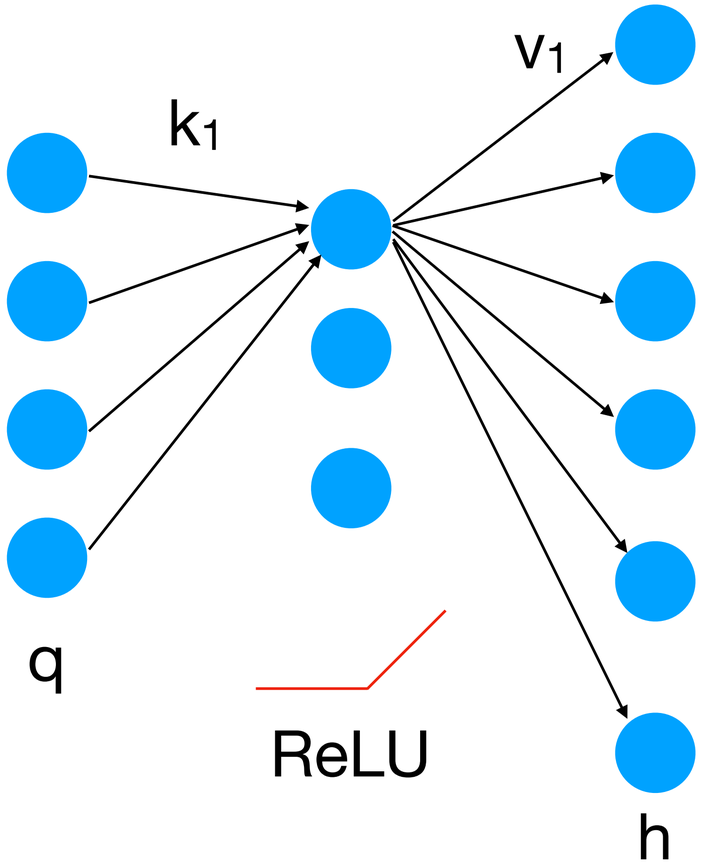




$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



MLP / Feed Forward NN



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

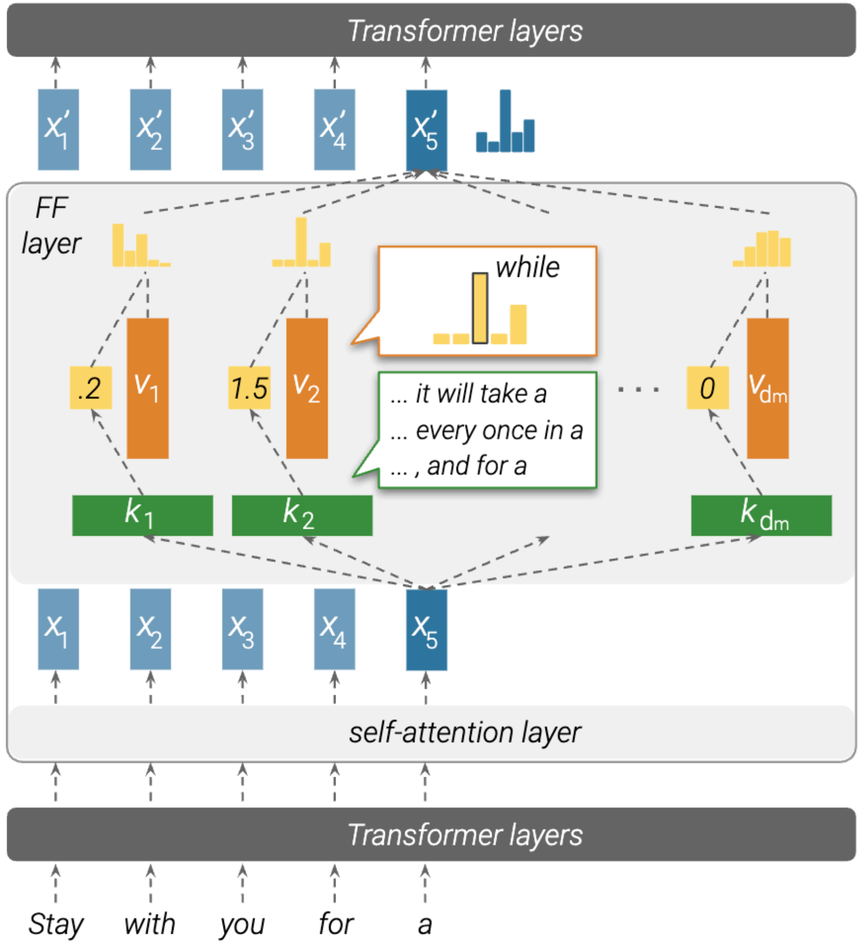
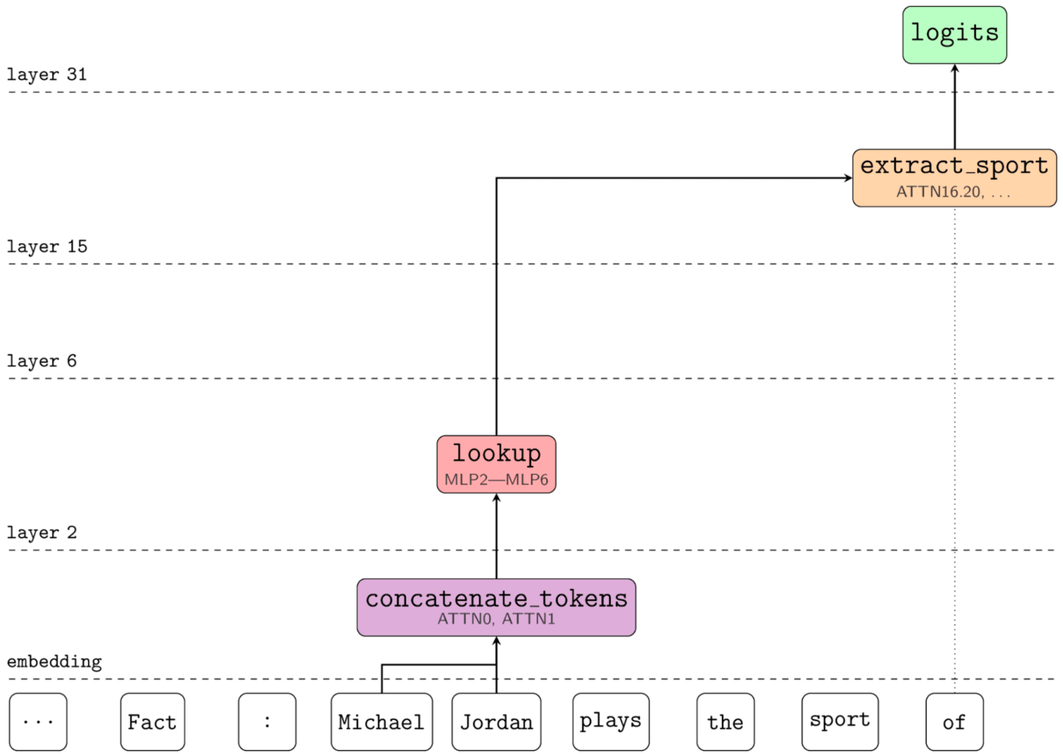
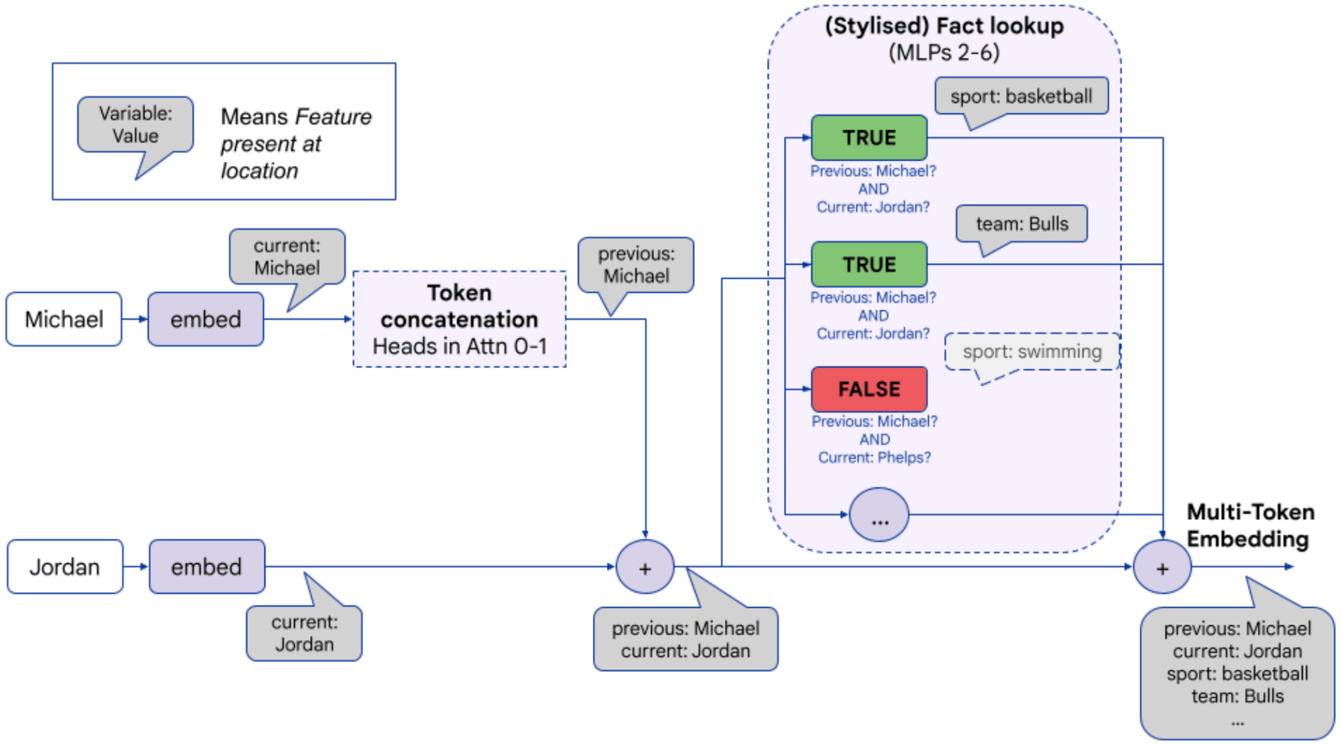


Figure 1: An illustration of how a feed-forward layer emulates a key-value memory. Input vectors (here, $\mathbf{x}_5$) are multiplied by *keys* to produce *memory coefficients* (e.g., the memory coefficient for $\mathbf{v}_1$ is 0.2), which then weigh distributions over the output vocabulary, stored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

Example for Attention & MLP





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

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

A RNN Language Model

output distribution

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

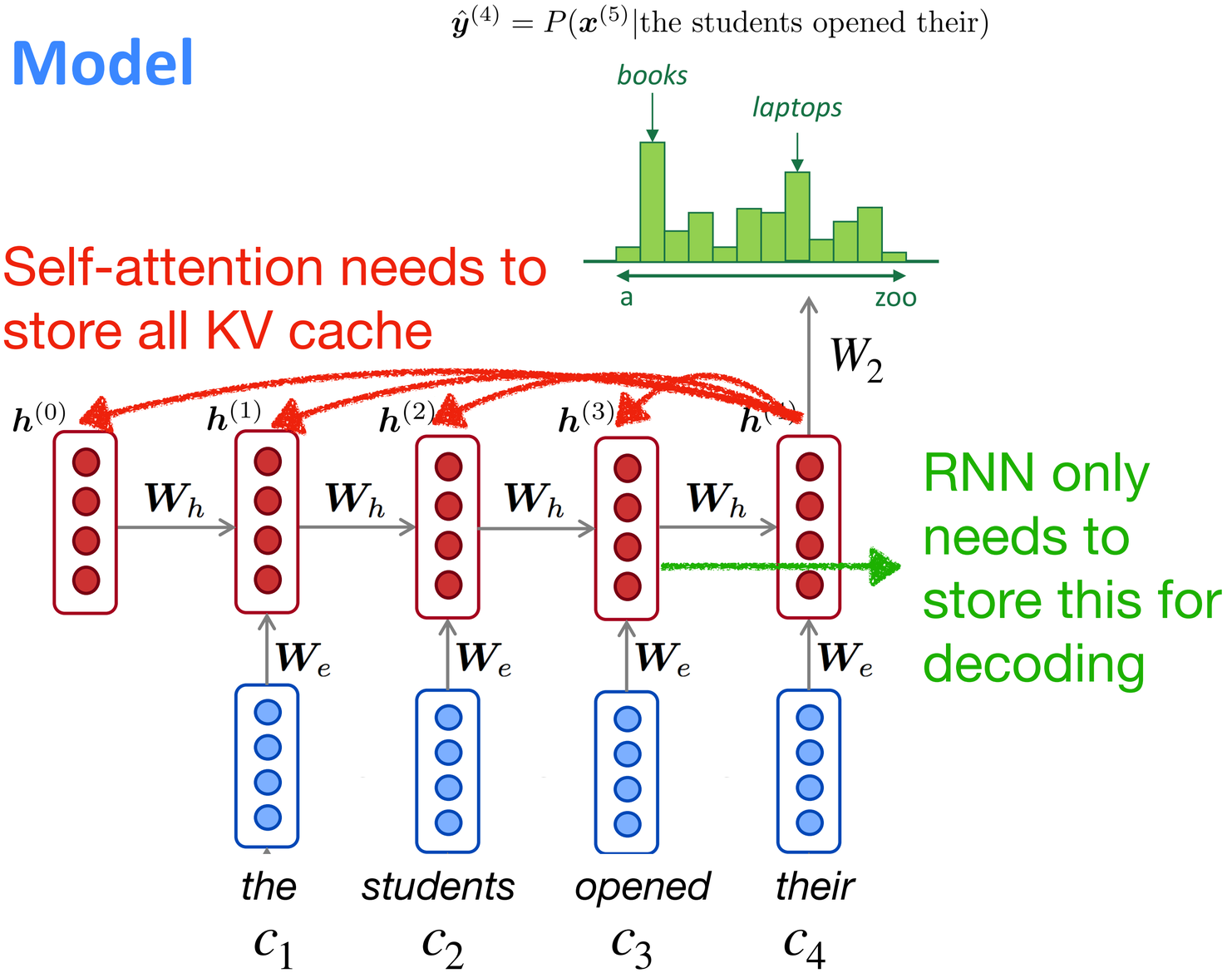
hidden states

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

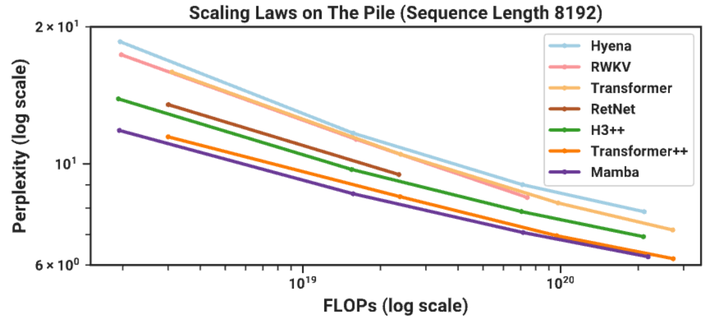
h(0) is initial hidden state!

word embeddings

$$c_1, c_2, c_3, c_4$$



Why is Attention Effective?



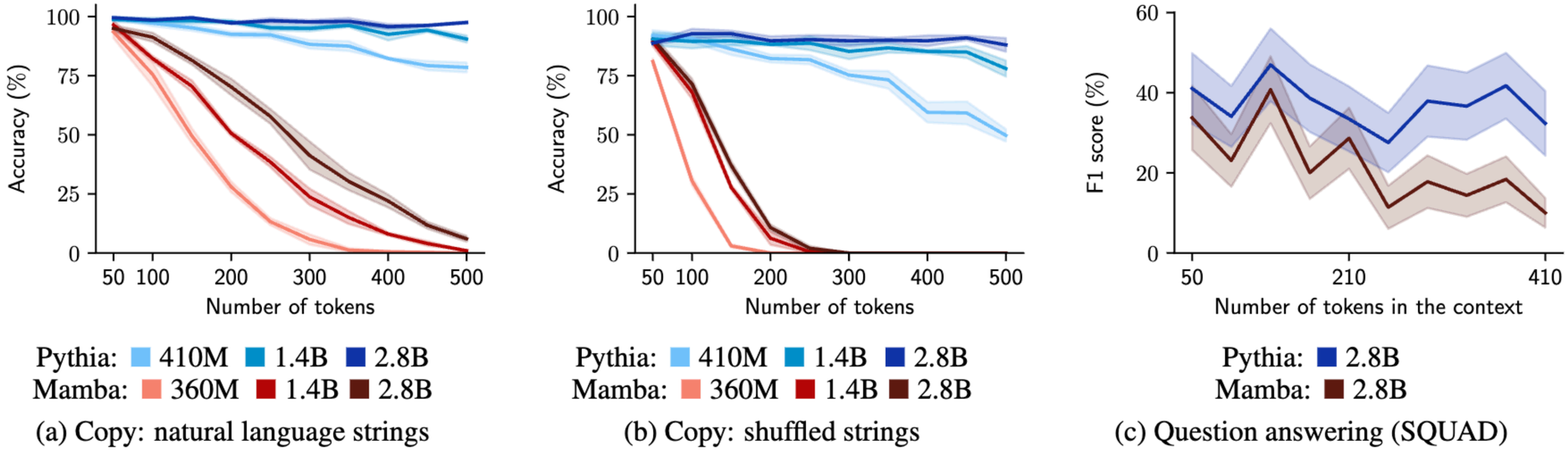


Figure 7. **(a) Copy: natural language strings.** We compare pretrained models on their ability to copy natural language strings sampled from C4 of varying lengths and report string-level accuracy. The transformer models substantially outperform the GSSMs. **(b) Copy: shuffled strings.** To test whether 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)

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?

A1: Because RNN can only take the values from the previous hidden state but the Transformer can take the values from all previous hidden states

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

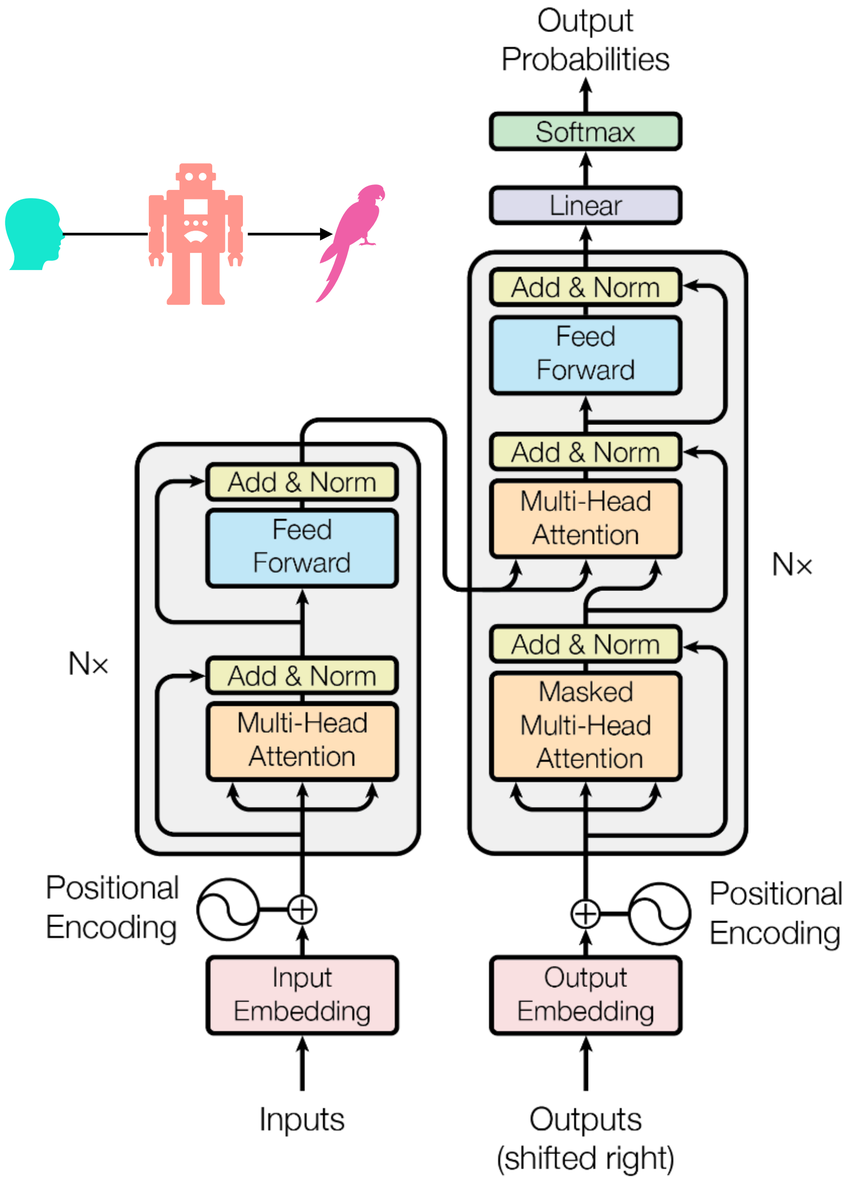
A2: Standard self-attention needs more computation, but they can be parallelized

Transformer Architecture

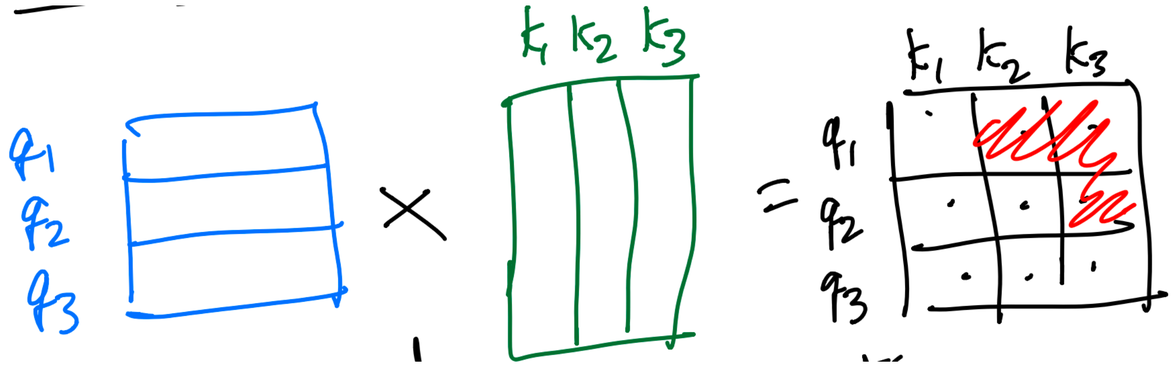
- The major components in the Transformer are all some forms of matrix factorization
  - All generalizations are based on similarity.
  - Similar to the recommendation engine.
  - Not designed for inference based on the rules.

- LLM -> AGI?
  - Matrix factorizations are enough for intelligence?
  - Humans are also a kind of parrot?

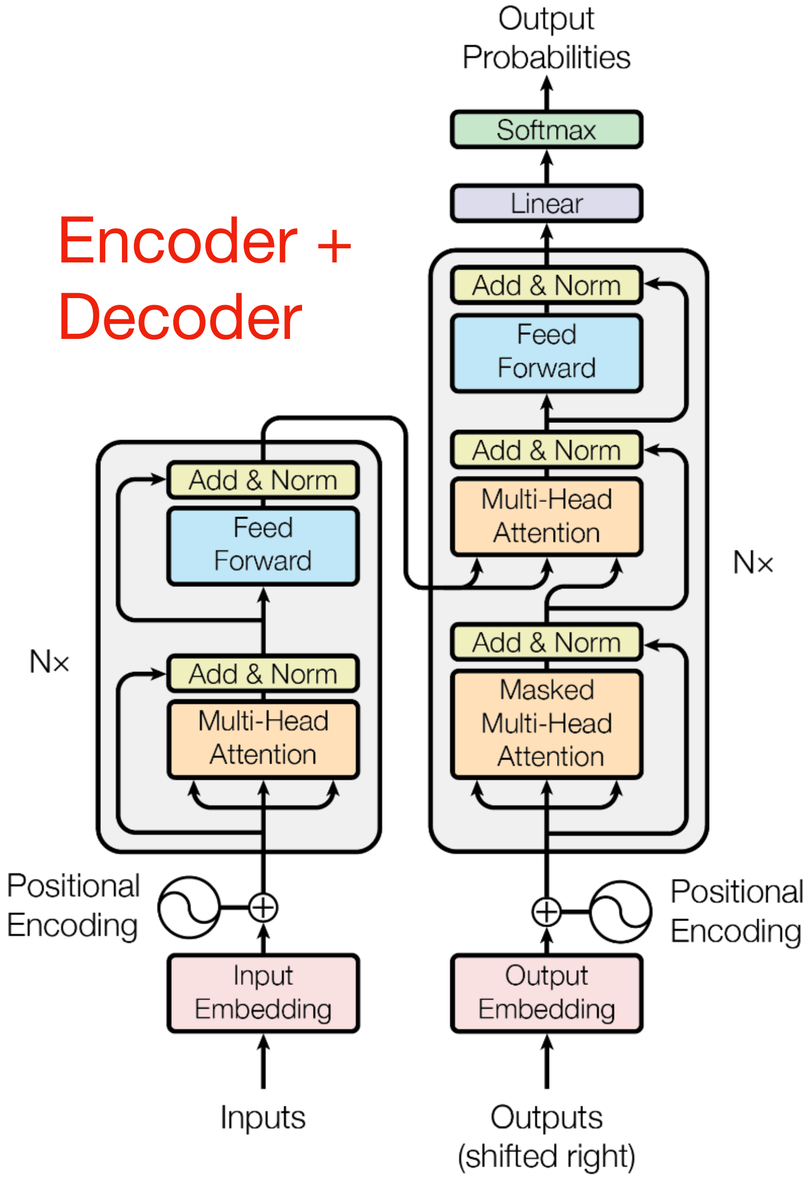
By the way, Ilya started to study compression recently



Last Year Notes



Encoder + Decoder



Architecture Comparison

Encoder

Decoder

Encoder + Decoder

XX BERT

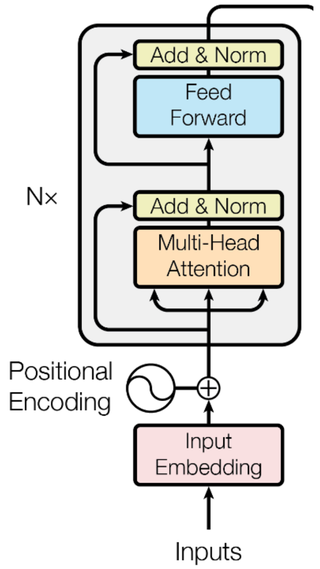
GPT

T5

ChatGPT
Claude
Gamini
Llama
Deepseek

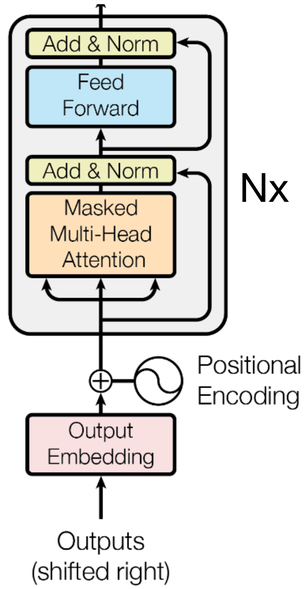
Encoder

Loss?



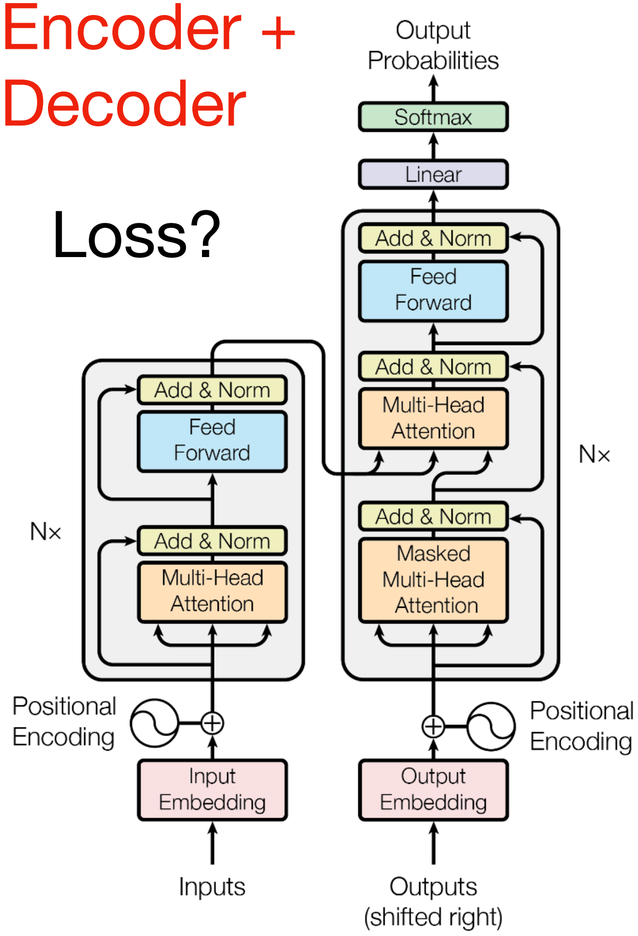
Decoder

Loss: Predicting next token



Encoder + Decoder

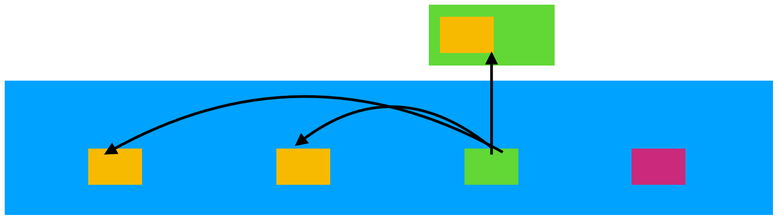
Loss?



Pros and Cons of Encoder

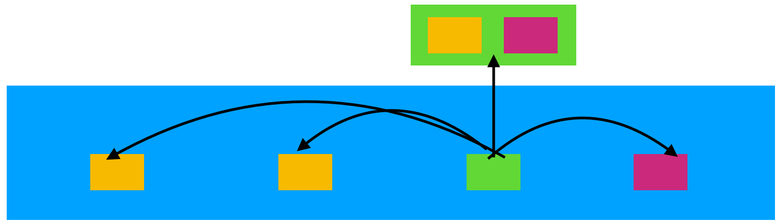
Pros: Get Embeddings

GPT



Mary's friend, John helps

BERT



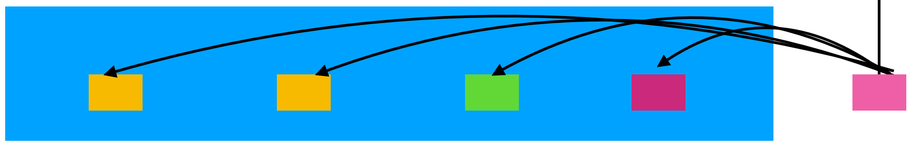
Mary's friend, John helps

Cons: Generation

GPT

Old hidden states

New hidden states

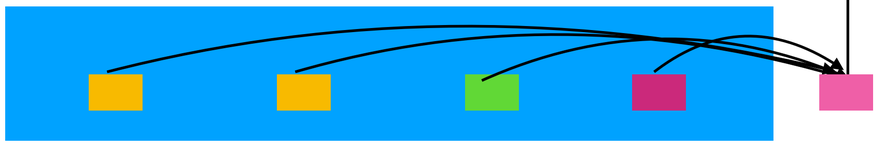


Mary's friend, John helps me

BERT

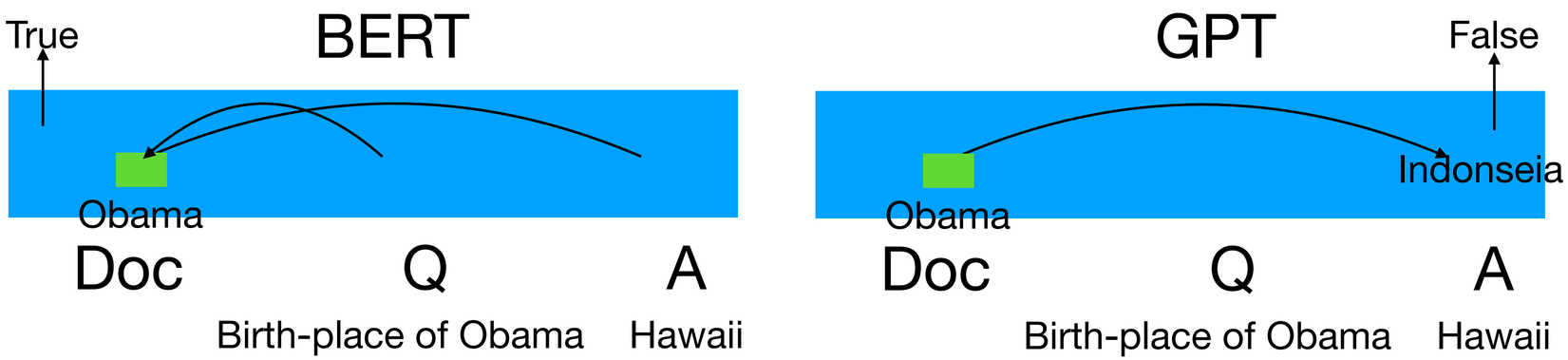
New hidden states

New hidden states



Mary's friend, John helps

BERT vs GPT



Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

- But why nowadays, everyone use GPT?
  - The gap decreases as the model size increases
    - You can store all possible future possibilities
  - NLG > NLU
    - The performance of free-form instruction becomes more important
  - Data > Structure
    - Training faster -> Training more data
    - Compress/memorize all data on the Internet

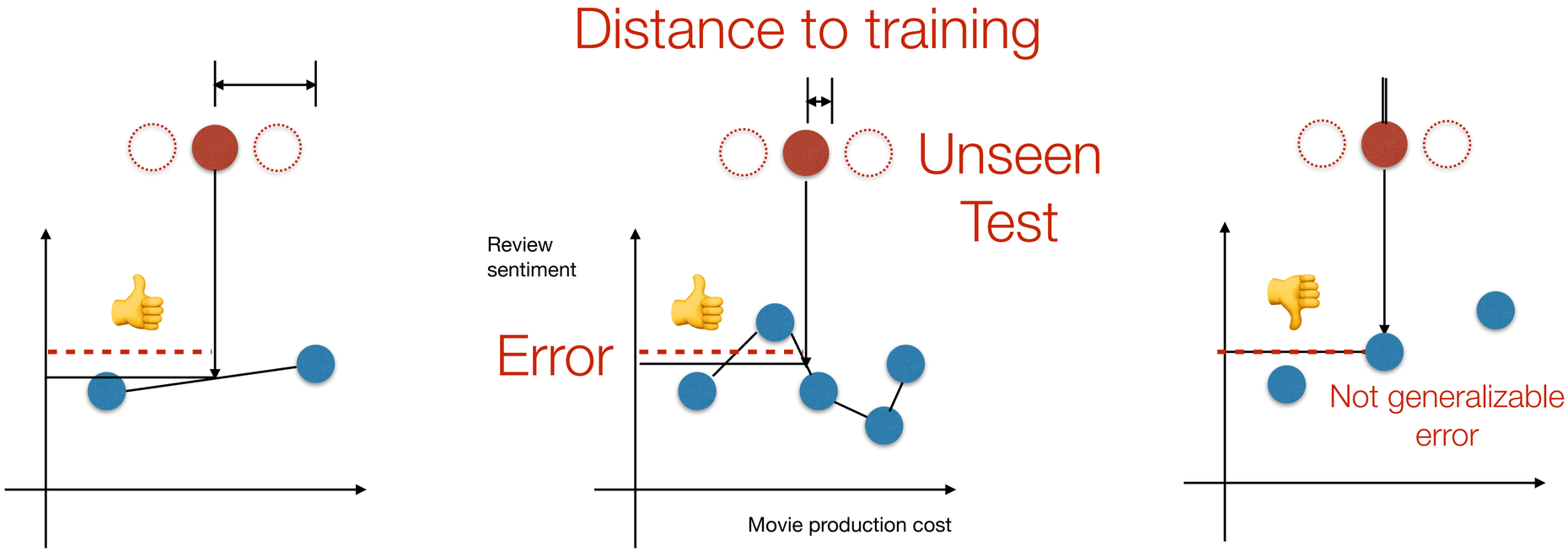
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/pdf/1810.04805)

Data is the King for Maximizing Task Performances

SFT

Pretraining
(Large LM)
SFT

Training on
Testing data



Bitter Lesson (http://www.incompleteideas.net/IncIdeas/BitterLesson.html)

One of the most important concepts in this course, but More Data -> AGI? I personally disagree.

Possible Reasons that OpenAI led

- Google
  - A goal of scientists: conduct novel research
    - Architecture > Data
  - A goal of managers: increase the profits
    - Specialized NLU models were more practical
  - Encoder + Decoder > Decoder

- OpenAI
  - The goal is AGI
    - We can use NLG to do NLU, but the reverse is not true
    - Scaling is more important
  - Decoder > Encoder + Decoder

Encoder Architectures are still Useful

- Efficient applications
  - T5
    - Fine-tuning a small model for seq2seq is sometimes better than LLMs
  - XX BERT
    - Bert -> Roberta-> Electra/Deberta-v3 -> Modernbert
    - Reward model
    - Retriever
    - Recommendation
    - Small verifier
      - Will talk more about this later

Midterm Example Question

- Assuming you want to automatically evaluate the stories based on the special taste of a very successful editor. You collect a set of acceptance and rejection judgments of 20k from the editor. Assuming that you have no resources to fine-tune the LLMs. Which options are the best in this case?

- (A) Fine-tuning BERT because bidirectional attention is powerful
- (B) Fine-tuning GPT-2 because the training could be better parallelized
- (C) Fine-tuning T5 because it has bidirectional attention and could be trained parallelly
- (D) LLM few-shot because LLM is powerful

Language Modeling / Pretraining

- Task:
  - Predict the next token
- Loss:
  - Maximal Likelihood / Cross-entropy
- Model:
  - Tables -> Neural Network -> Transformer
    - Self-attention and Interpretation
    - **Future**: tokenization and positional embedding
- Optimization:
  - Counting -> Gradient Descent