Course evaluation

- https://evaluation.shanghaitech.edu.cn/tqnmaep/login

Advanced topics of AI

Large Language Models

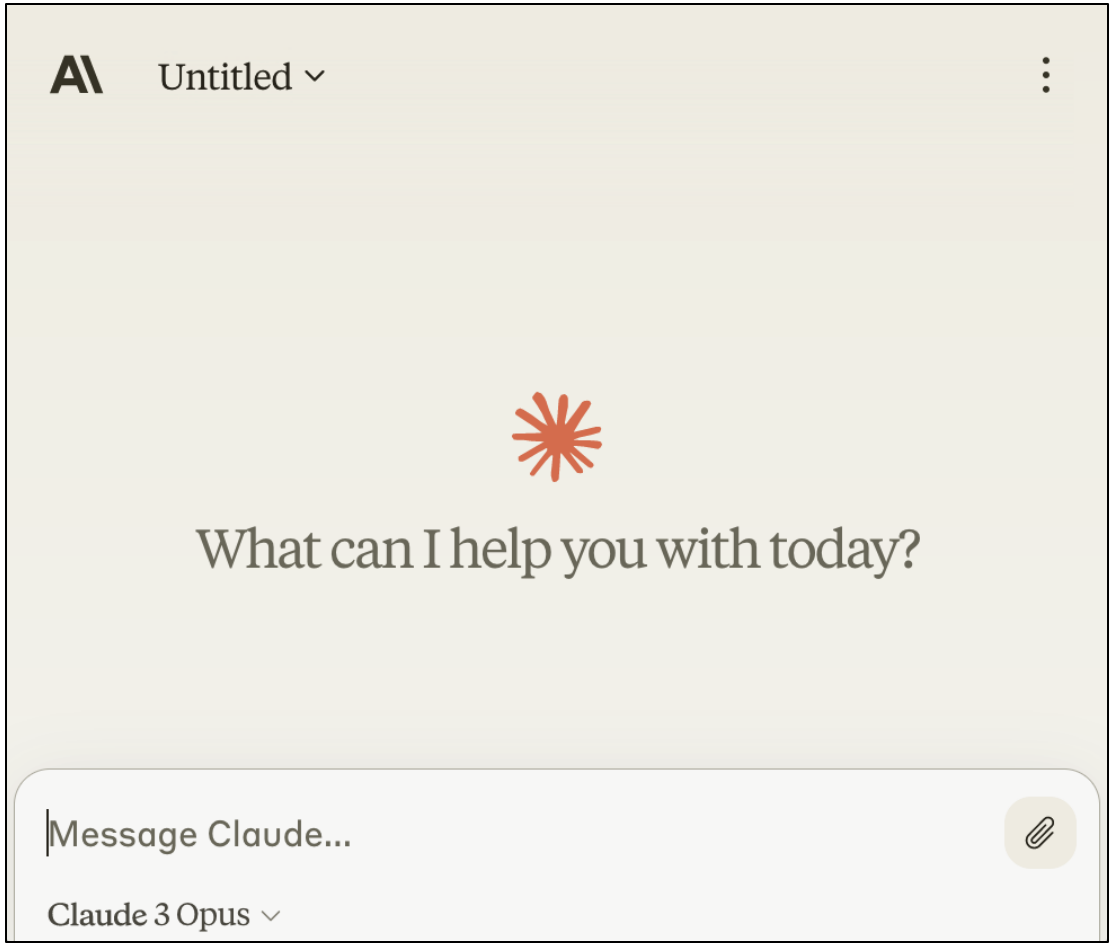
[Adapted from slides by Dan Klein and Pieter Abbeel at UC Berkeley]

Advanced topics of AI

- Large language models
- Advanced deep learning
- Advanced reinforcement learning
- Responsible AI

Today’s AI

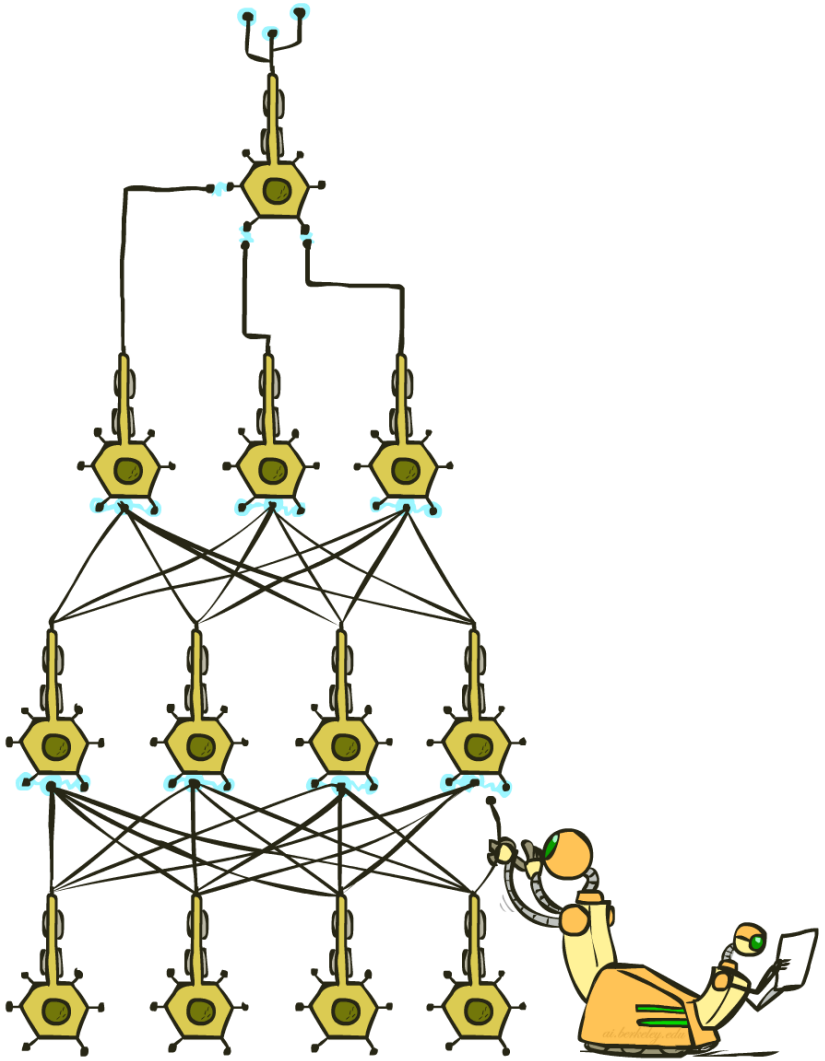

ChatGPT 4
How can I help you today?
Message ChatGPT...
ChatGPT can make mistakes. Consider checking important information.


Untitled
What can I help you with today?
Message Claude...
Claude 3 Opus

Large Language Models

- Feature engineering
  - Text tokenization
  - Word embeddings
- Deep neural networks
  - Autoregressive models
  - Self-attention mechanisms
  - Transformer architecture
- Multi-class classification
- Supervised learning
  - Self-supervised learning
  - Instruction tuning
- Reinforcement learning
  - … from human feedback (RLHF)
- Policy search
  - Policy gradient methods
- Beam search

Deep Neural Networks



- Input: some text
  - “The dog chased the”
- Output: more text
  - … “ ball”
- Implementation:
  - Linear algebra
  - How??

Text Tokenization

GPT-3.5 & GPT-4 | GPT-3 (Legacy)

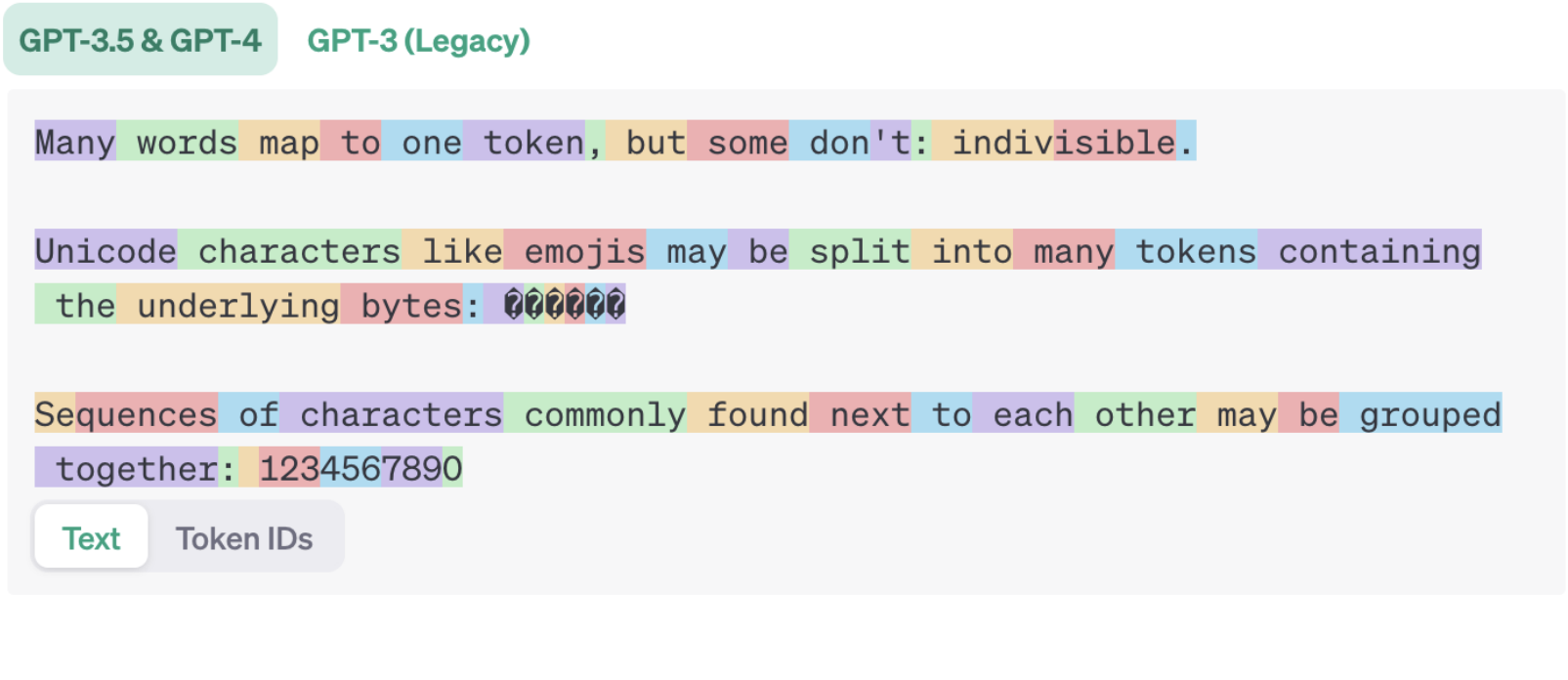
```
Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing
the underlying bytes: 🤚

Sequences of characters commonly found next to each other may be grouped
together: 1234567890
```

Clear | Show example

Tokens	Characters
57	252

https://platform.openai.com/tokenizer

Text Tokenization



https://platform.openai.com/tokenizer

Text Tokenization

GPT-3.5 & GPT-4 | GPT-3 (Legacy)

```
[8607, 4339, 2472, 311, 832, 4037, 11, 719, 1063, 1541, 956, 25, 3687,
23936, 382, 35020, 5885, 1093, 100166, 1253, 387, 6859, 1139, 1690,
11460, 8649, 279, 16940, 5943, 25, 11410, 97, 248, 9468, 237, 122, 271,
1542, 45045, 315, 5885, 17037, 1766, 1828, 311, 1855, 1023, 1253, 387,
41141, 3871, 25, 220, 4513, 10961, 16474, 15]
```

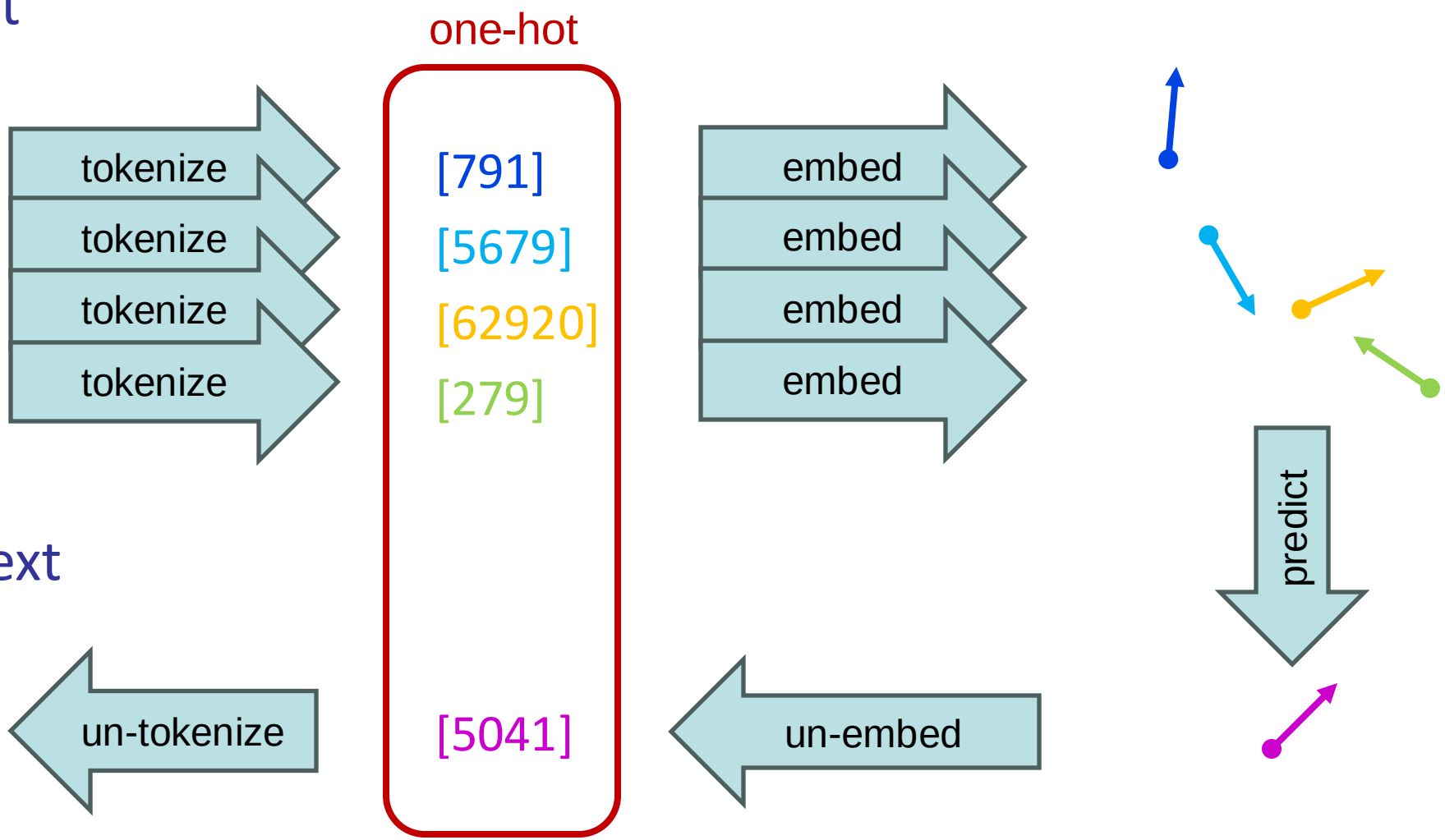
Text | Token IDs

Tokens	Characters
57	252

https://platform.openai.com/tokenizer

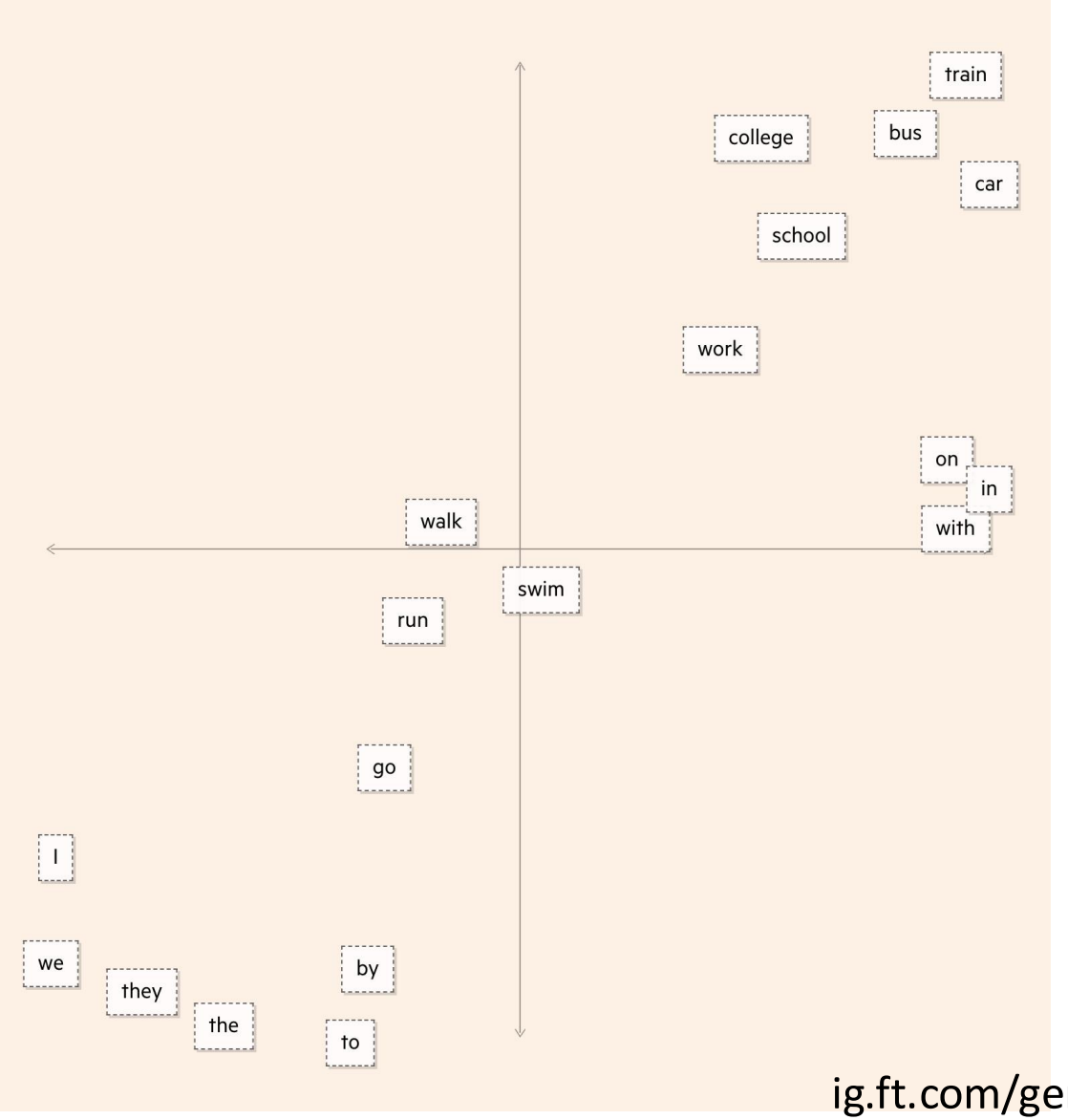
Word Embeddings

- Input: some text
  - “The”
  - “ dog”
  - “ chased”
  - “ the”
- Output: more text
  - “ ball”



What do word embeddings look like?

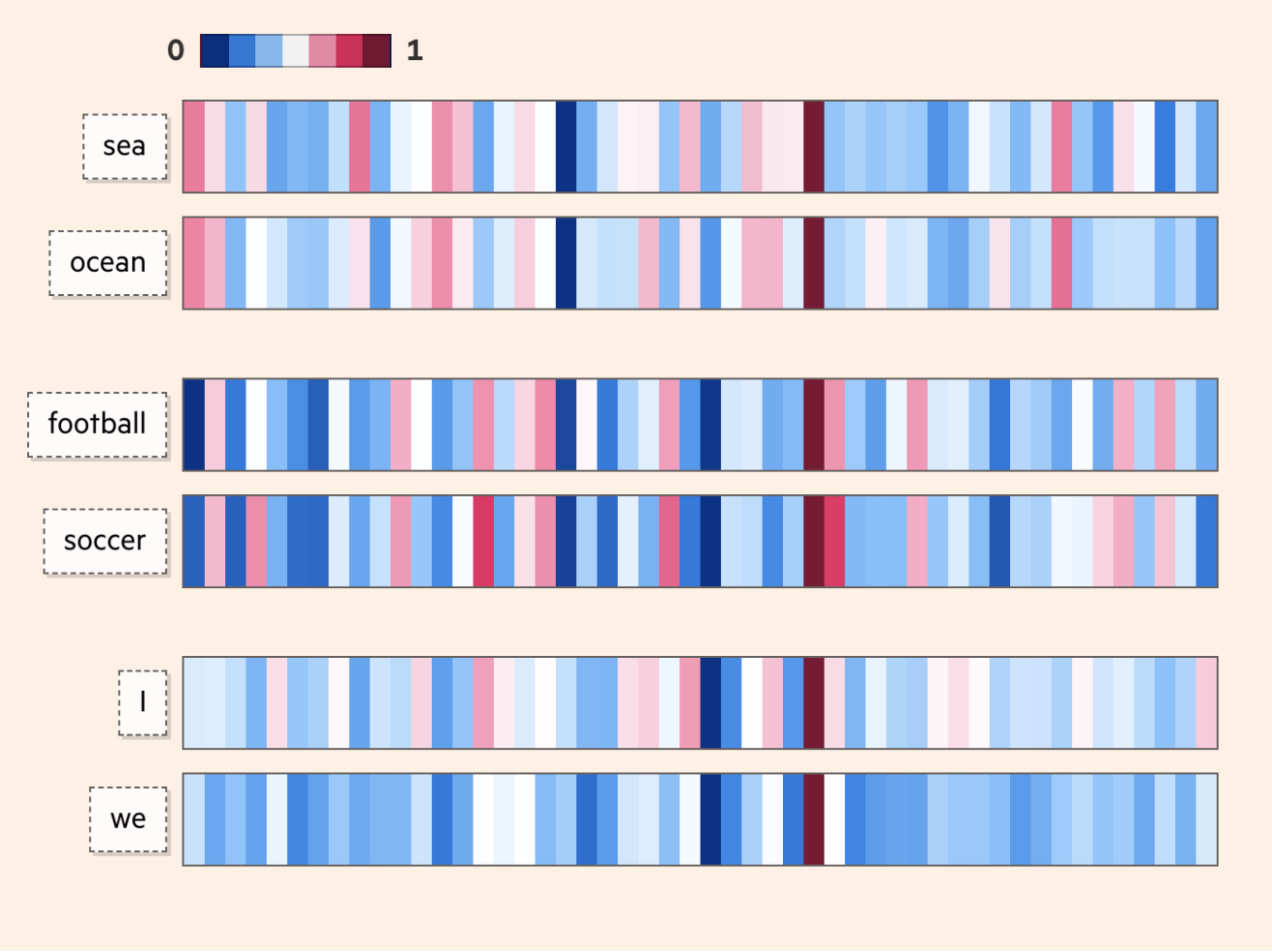
- Words cluster by similarity:



ig.ft.com/generative-ai

What do word embeddings look like?

- Features learned in language models:



ig.ft.com/generative-ai

What do word embeddings look like?

- Signs of sensible algebra in embedding space:

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

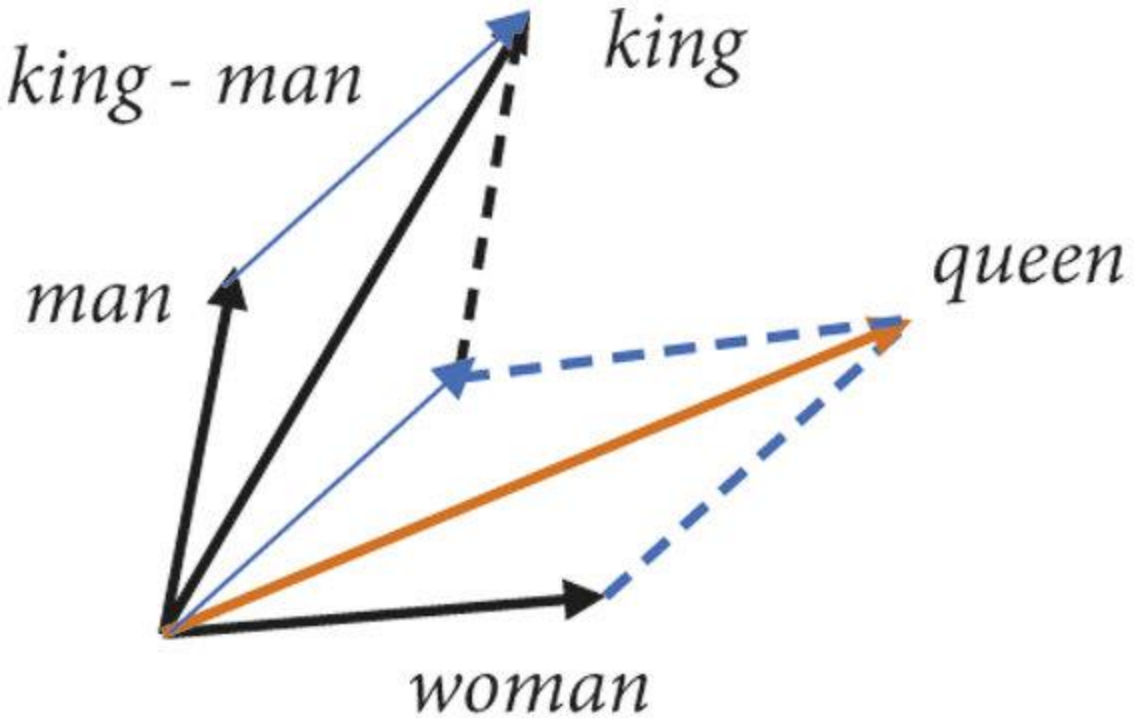
Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

[Efficient estimation of word representations in vector space, Mikolov et al, 2013]

What do word embeddings look like?

- Signs of sensible algebra in embedding space:



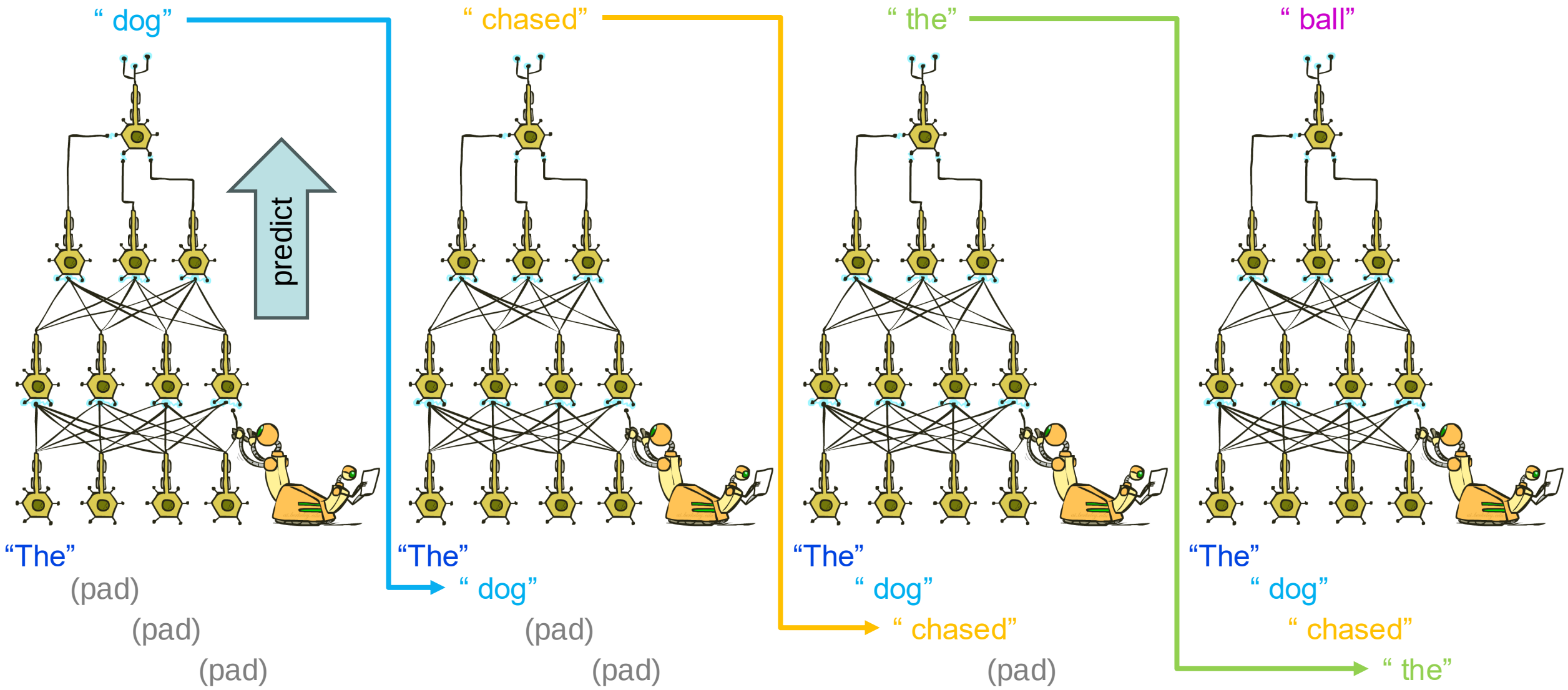
$$king - man + woman \approx queen$$

[Efficient estimation of word representations in vector space, Mikolov et al, 2013]

Large Language Models

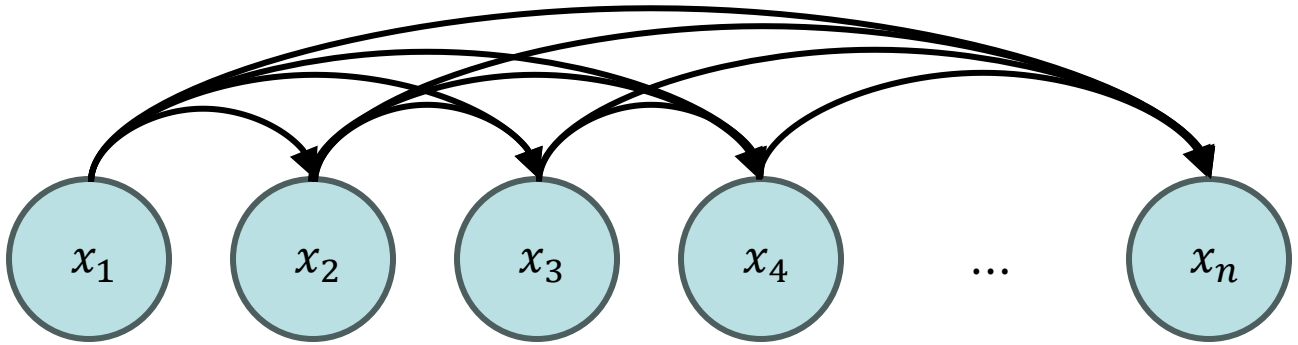
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- Deep neural networks
  - Autoregressive models
  - Self-attention mechanisms
  - Transformer architectures
- Multi-class classification
- Supervised learning
  - Self-supervised learning
  - Instruction tuning
- Reinforcement learning
  - … from human feedback (RLHF)
- Policy search
  - Policy gradient methods
- Beam search

Autoregressive Models



Autoregressive Models

- Predict output one piece at a time (e.g. word, token, pixel, etc.)
- Concatenate: input + output
- Feed result back in as new input
- Repeat



Autoregressive Models

- The formulation of autoregressive models
  - At each step t, the model captures the conditional distribution of

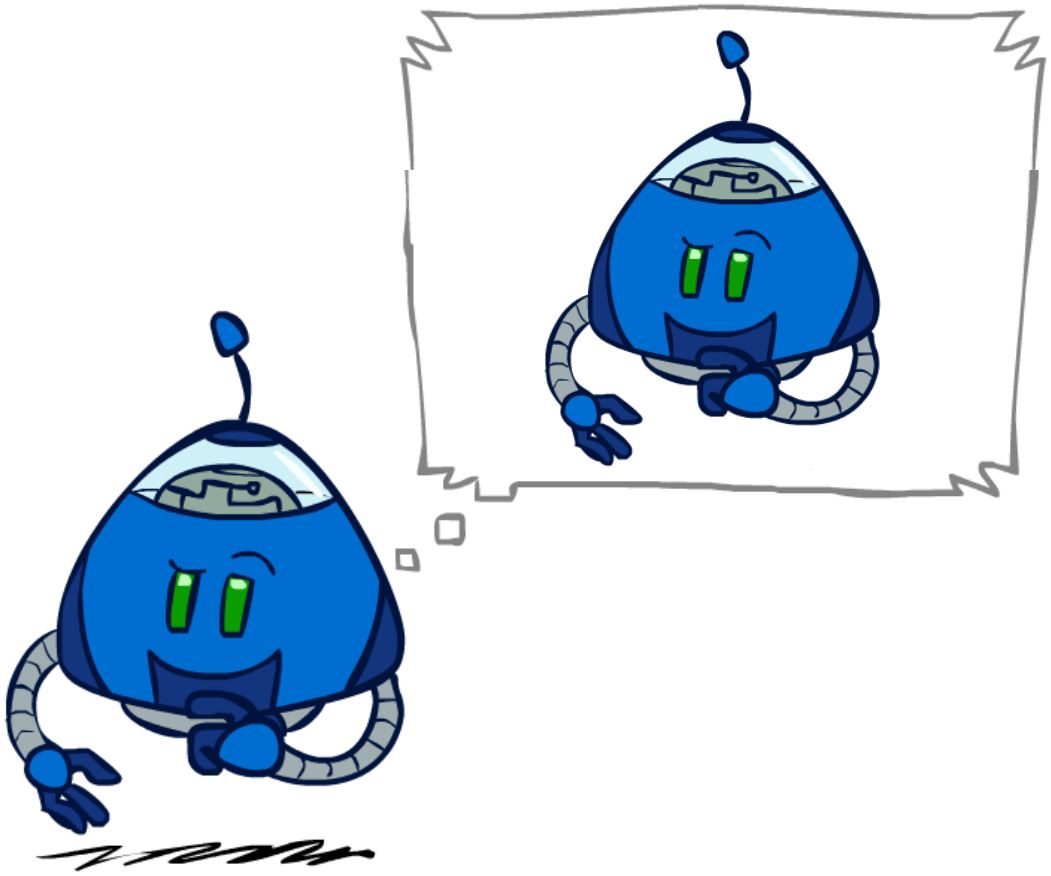
$$p(o_t|o_1, o_2, \dots, o_{t-1})$$

where $o_t, t \in [1, T]$ is the token at the i-th position.

  - The joint distribution of the whole sentence is

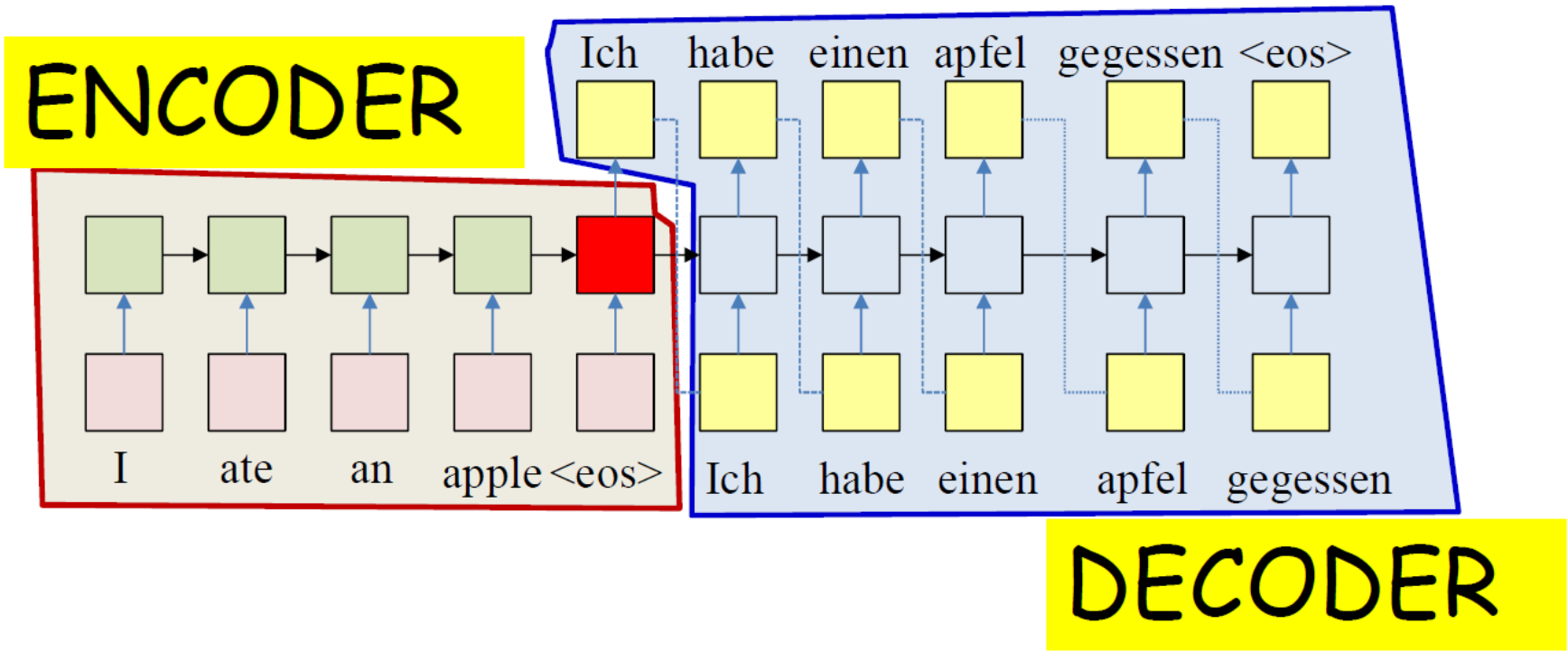
$$p(o_1, o_2, \dots, o_T) = \prod_{t=1}^{T} p(o_t|o_1, o_2, \dots, o_{t-1})$$

Self-Attention Mechanisms



The “simple” translation model

- This is also referred to as an encoder-decoder structure



The “simple” translation model

- This is also referred to as an encoder-decoder structure

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

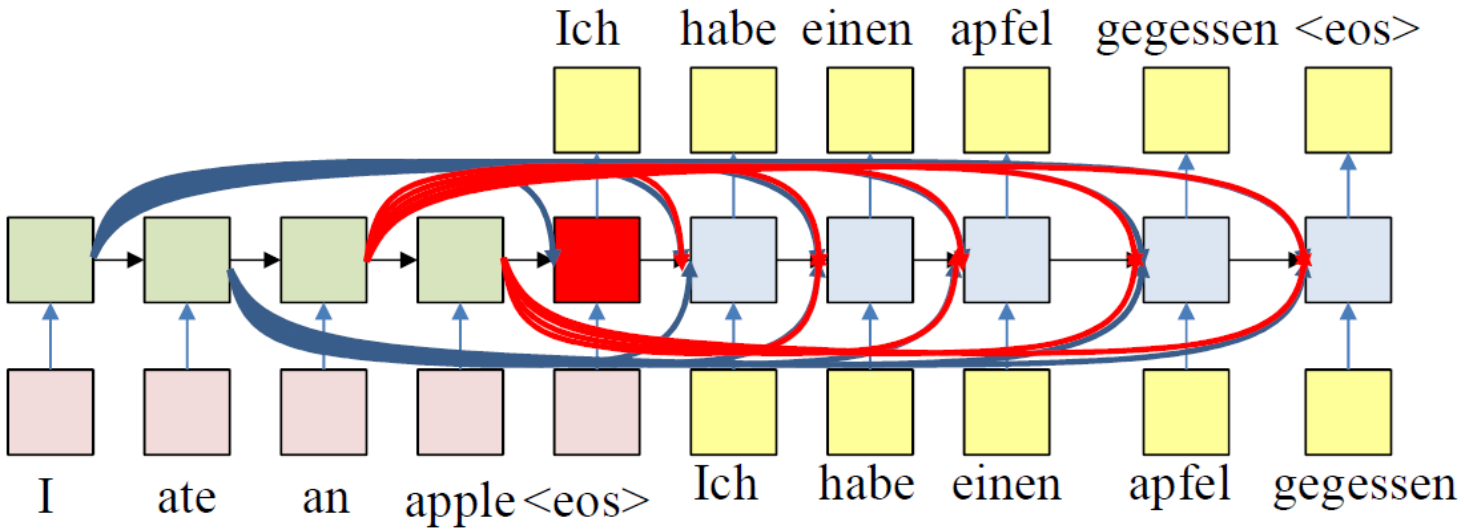
Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Sutskever, I. “Sequence to Sequence Learning with Neural Networks.” *Proceedings of the 27th International Conference on Neural Information Processing Systems. 2014.*

A problem with encoder-decoder framework

- All latent values carry information
  - Some of which may be diluted downstream
  - Different outputs are related to different inputs
  - Connecting everything to everything is infeasible



Encoder-decoder framework

- Attention mechanism

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

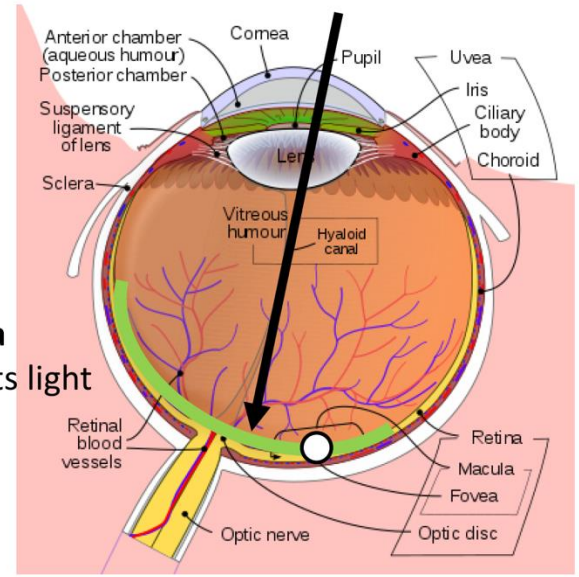
KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Bahdanau, Dzmitry. "Neural machine translation by jointly learning to align and translate." ICLR 2015.

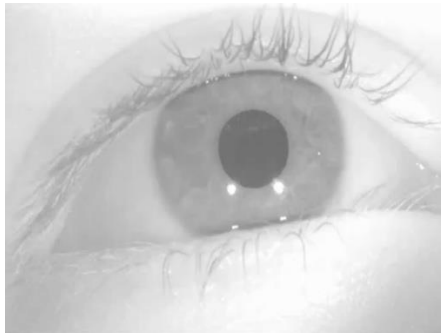
Attention Mechanism

- Human Vision: Fovea

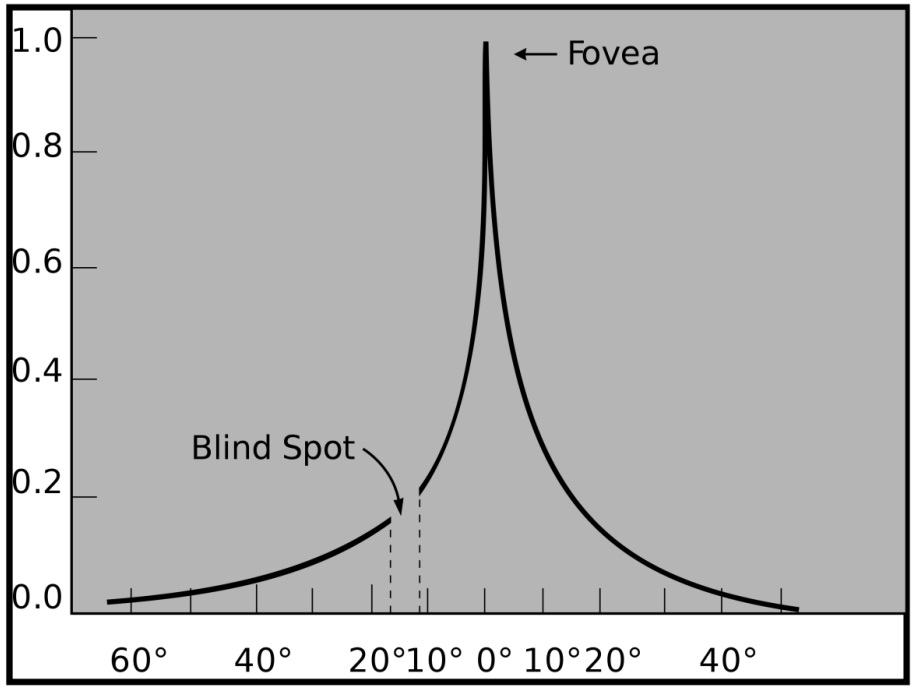
Light enters eye



Retina detects light

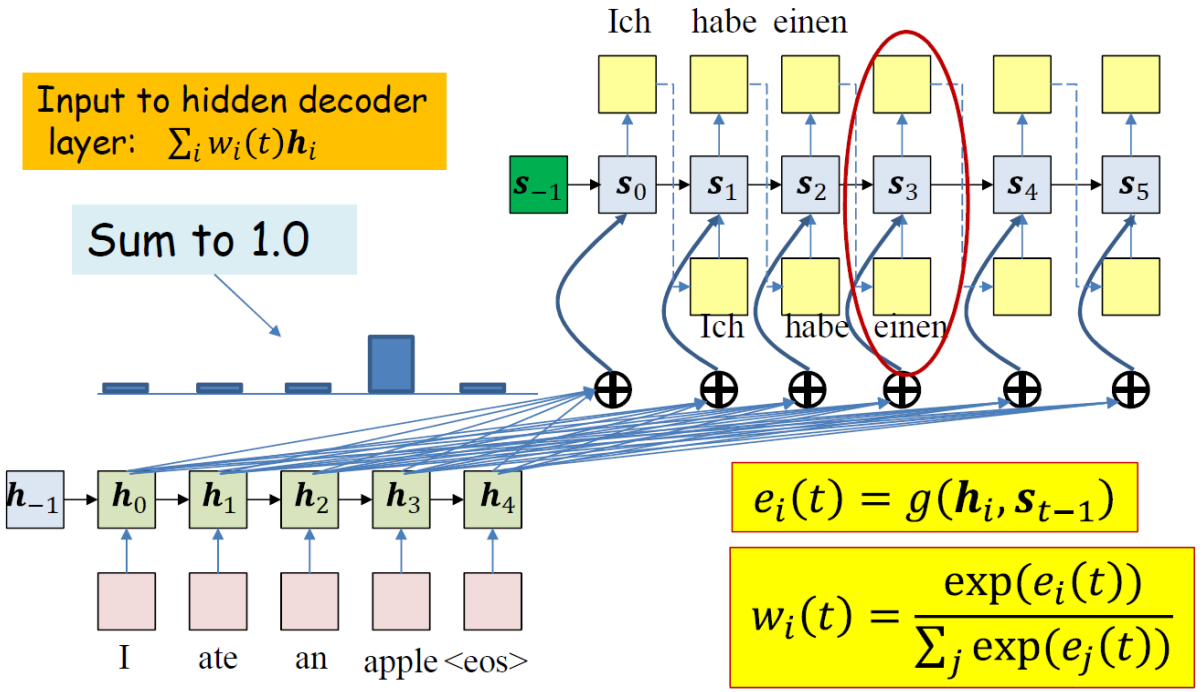


The **fovea** is a tiny region of the retina that can see with high acuity



Attention models

- The weights are a distribution over the input
  - A function g() on two hidden states followed by a softmax



Bahdanau, Dzmitry. "Neural machine translation by jointly learning to align and translate." ICLR 2015.

Self-attention in Transformer

- Attention is all you need. Vaswani et al. 2017.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

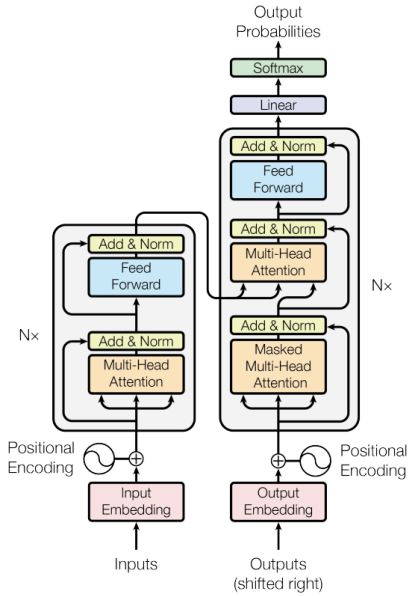


Figure 1: The Transformer - model architecture.

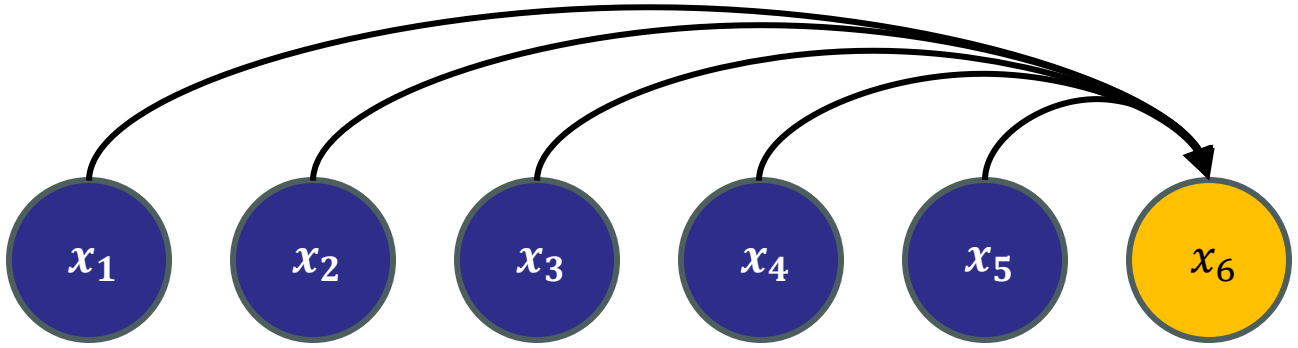
Attention is all you need

A Vaswani, N Shazeer, N Parmar... - Advances in neural ..., 2017 - proceedings.neurips.cc

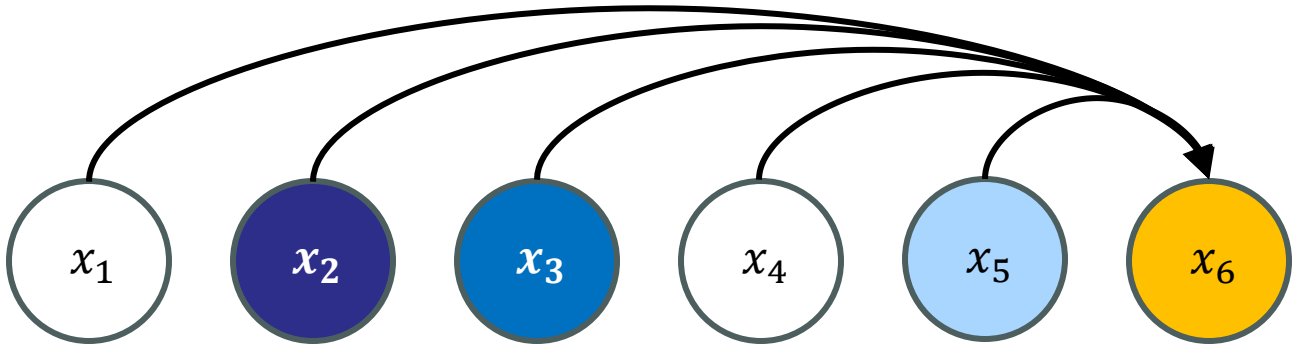
... to attend to **all** positions in the decoder up to and including that position. **We need** to prevent ... **We** implement this inside of scaled dot-product **attention** by masking out (setting to $-\infty$) ...

☆ Save 99 Cite Cited by 117858 Related articles All 87 versions »

Self-Attention Mechanisms

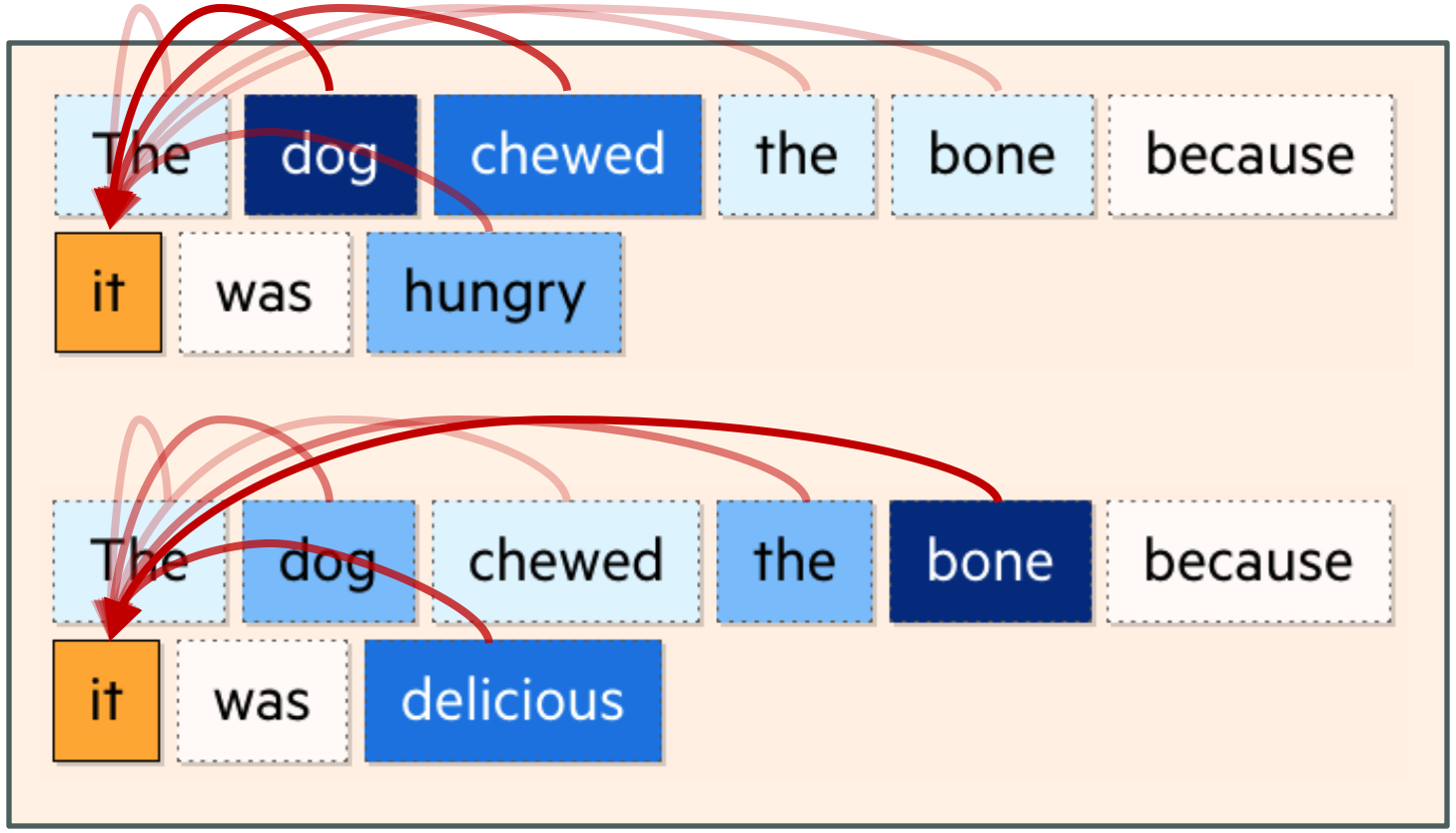


- Instead of conditioning on *all* input tokens equally…

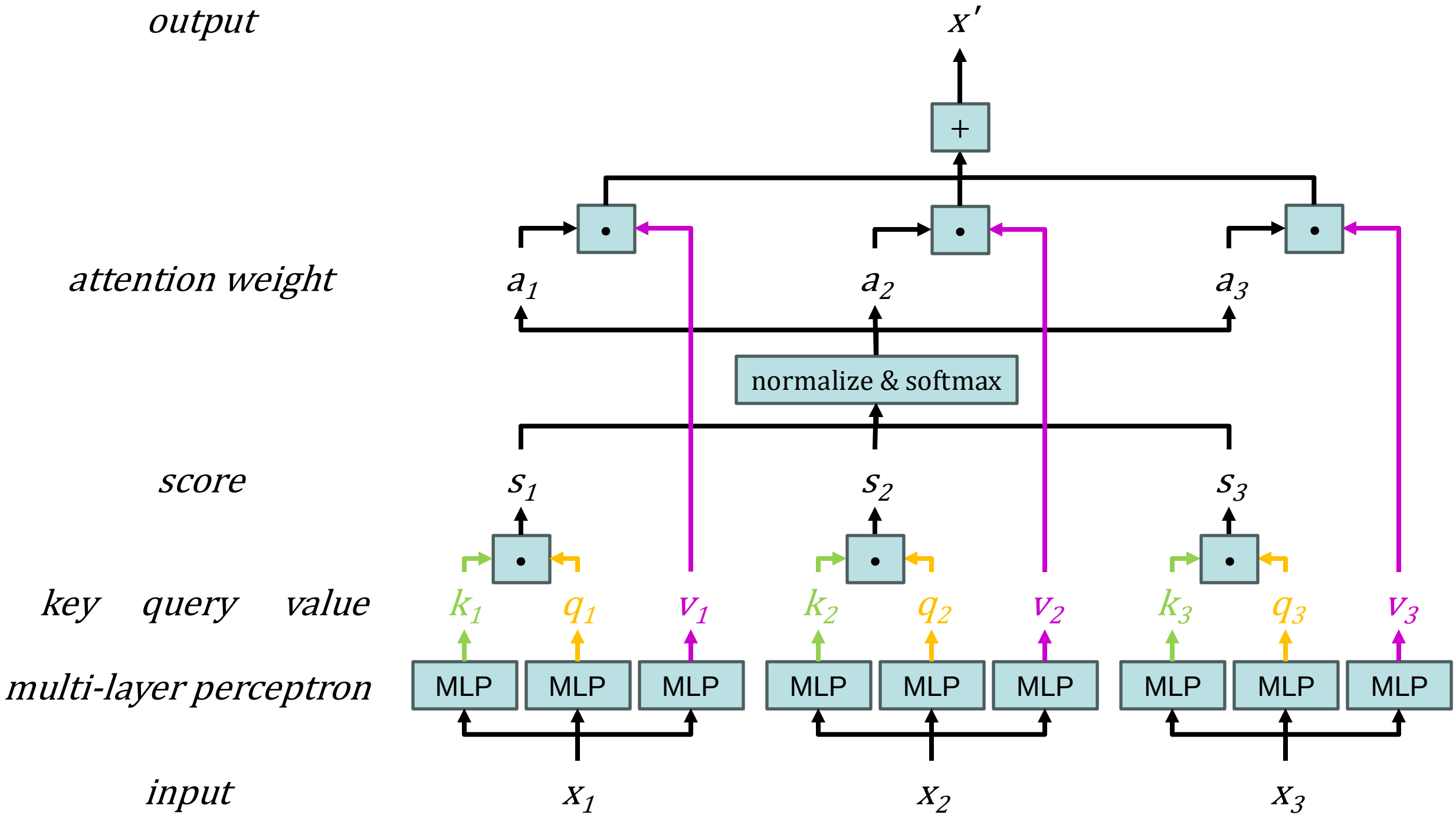


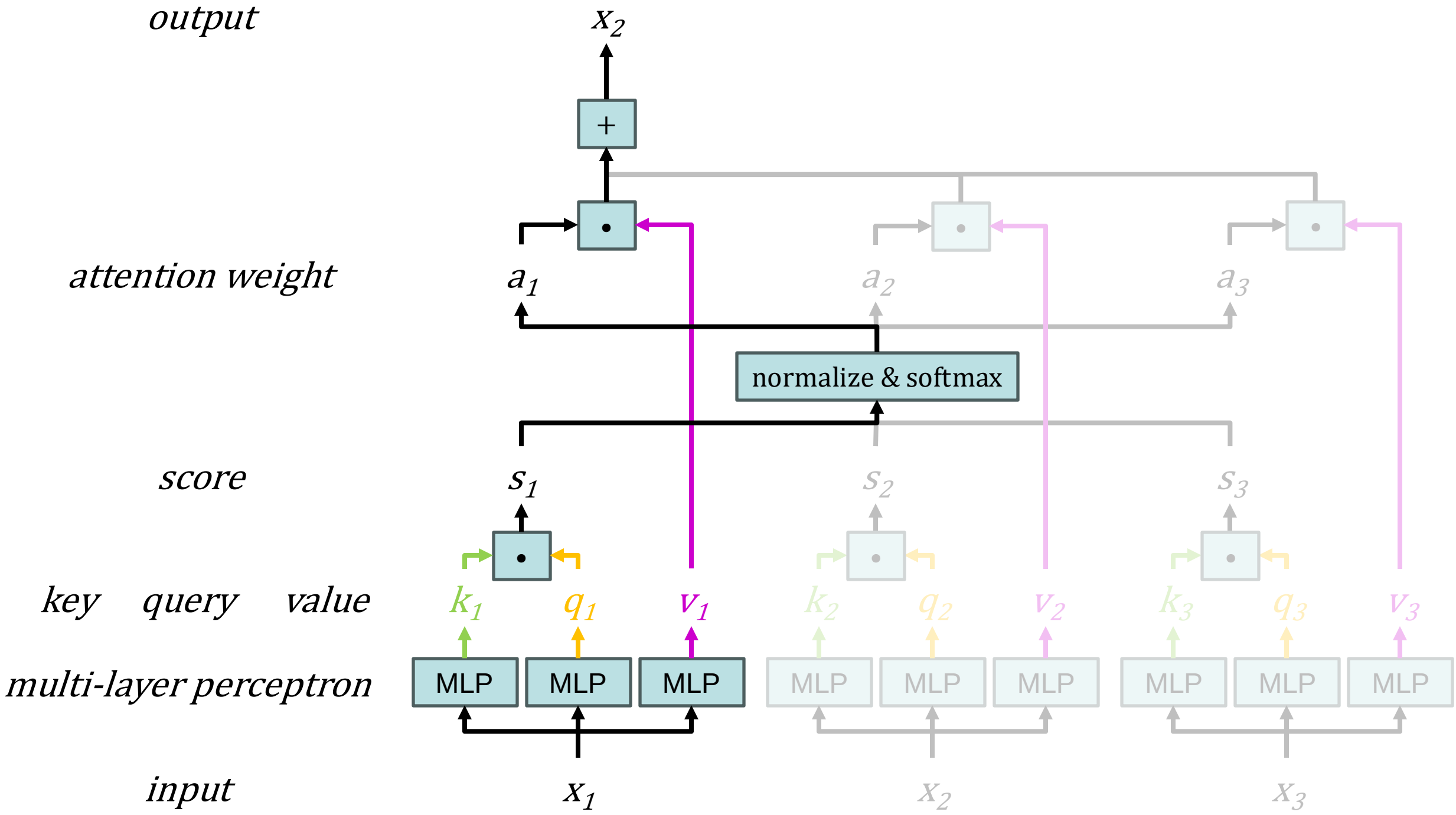
- Pay more attention to relevant tokens!

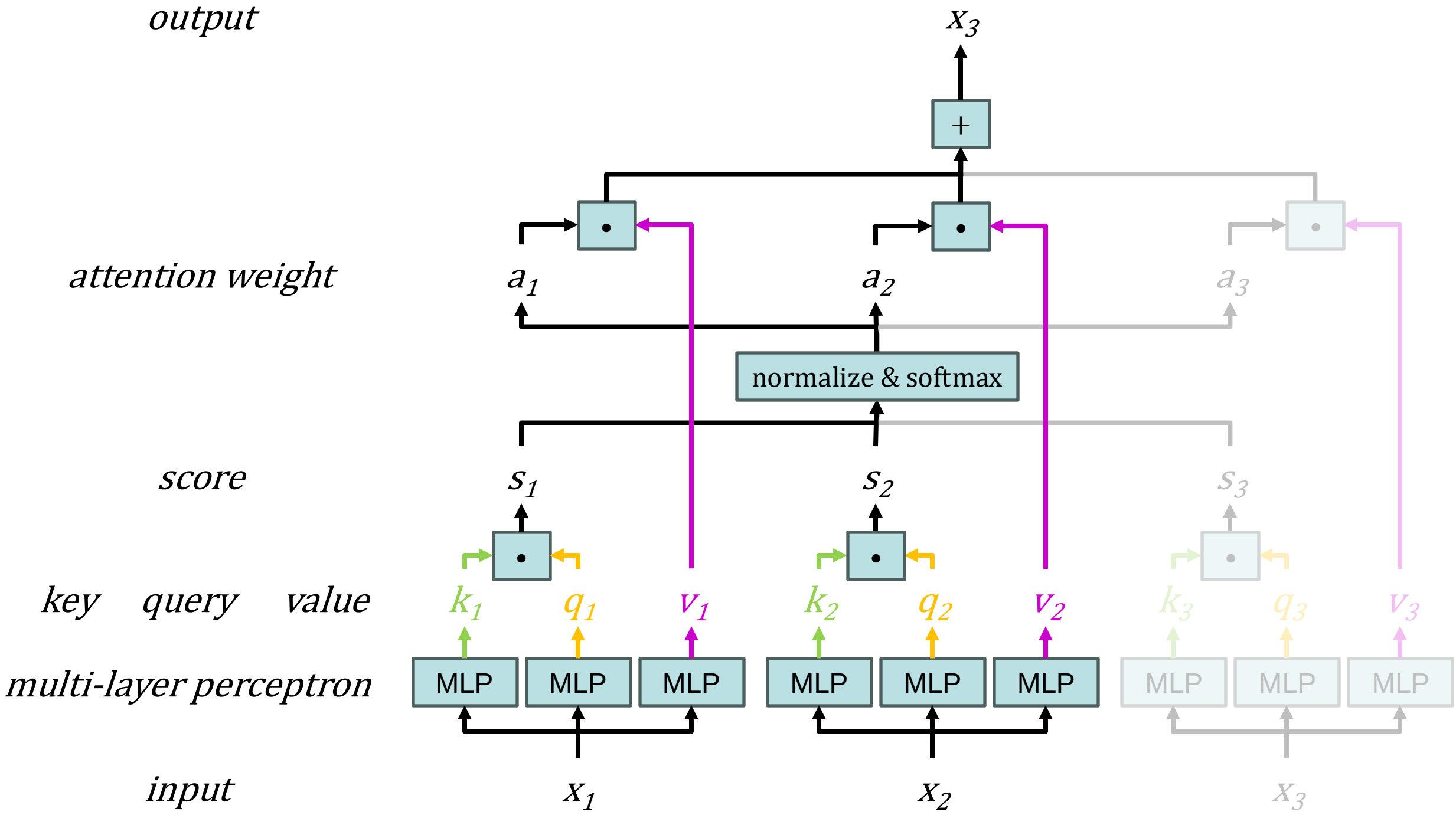
Self-Attention Mechanisms

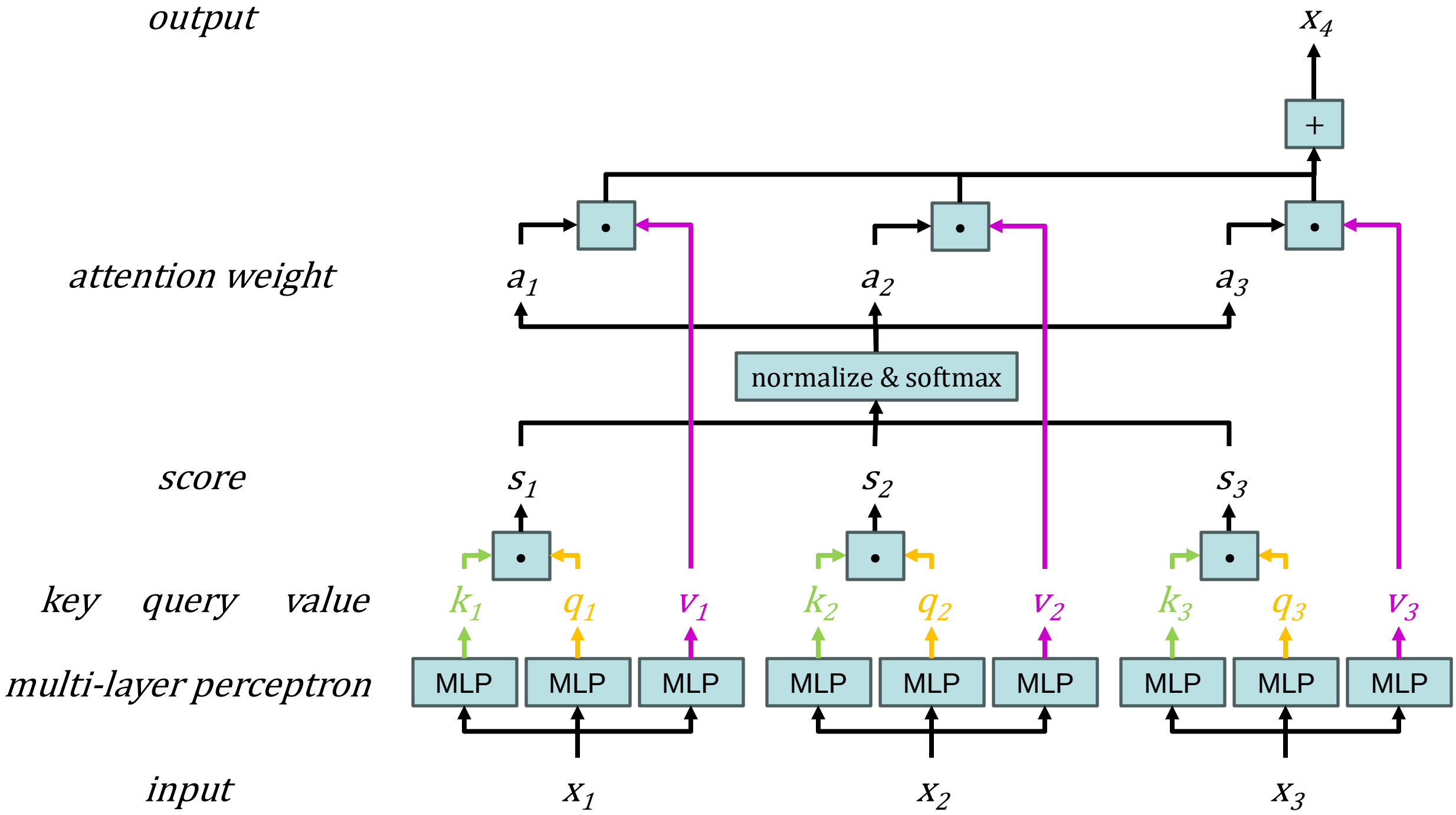


ig.ft.com/generative-ai


output
x'
+
attention weight
a_1
a_2
a_3
normalize & softmax
score
s_1
s_2
s_3
key
query
value
k_1
q_1
v_1
k_2
q_2
v_2
k_3
q_3
v_3
multi-layer perceptron
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
input
x_1
x_2
x_3

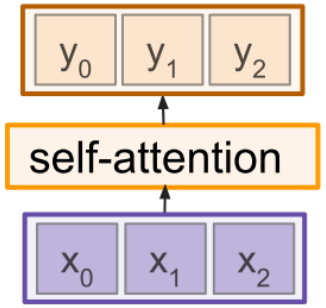

output
x_2
+
attention weight
a_1
a_2
a_3
normalize & softmax
score
s_1
s_2
s_3
key
query
value
k_1
q_1
v_1
k_2
q_2
v_2
k_3
q_3
v_3
multi-layer perceptron
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
input
x_1
x_2
x_3


output
attention weight
score
key
query
value
multi-layer perceptron
input
x_3
+
a_1
a_2
a_3
normalize & softmax
s_1
s_2
s_3
k_1
q_1
v_1
k_2
q_2
v_2
k_3
q_3
v_3
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
x_1
x_2
x_3


output
attention weight
score
key
query
value
multi-layer perceptron
input
x_4
+
a_1
a_2
a_3
normalize & softmax
s_1
s_2
s_3
k_1
q_1
v_1
k_2
q_2
v_2
k_3
q_3
v_3
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
x_1
x_2
x_3

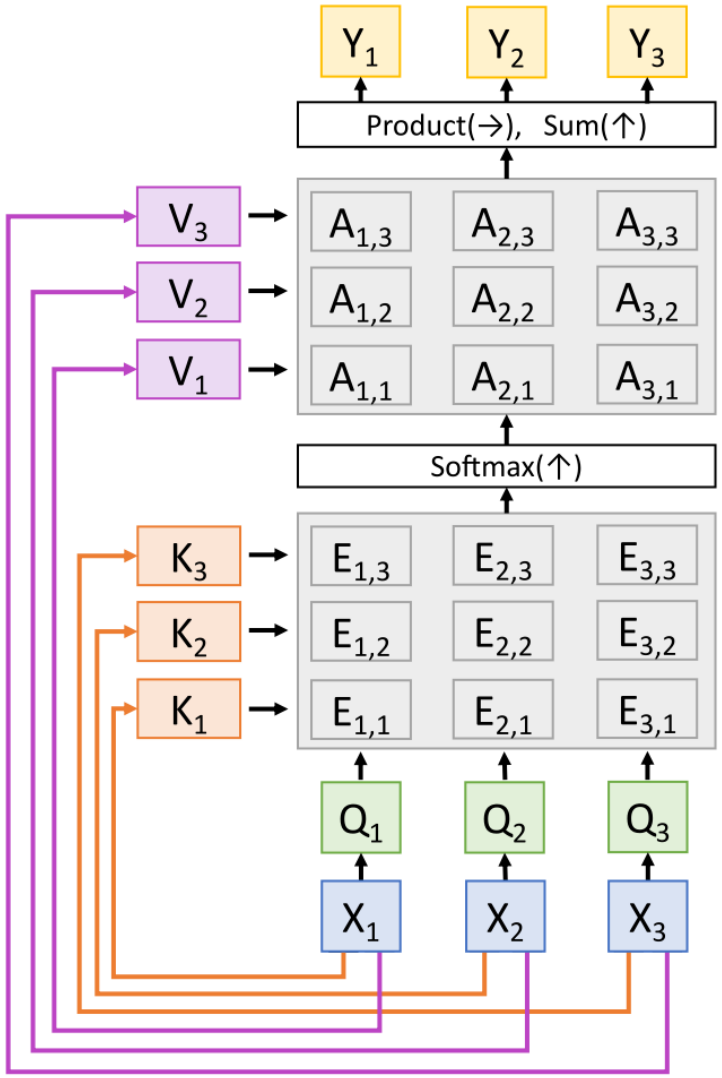
Self-attention Layer Summary

- One query per input vector



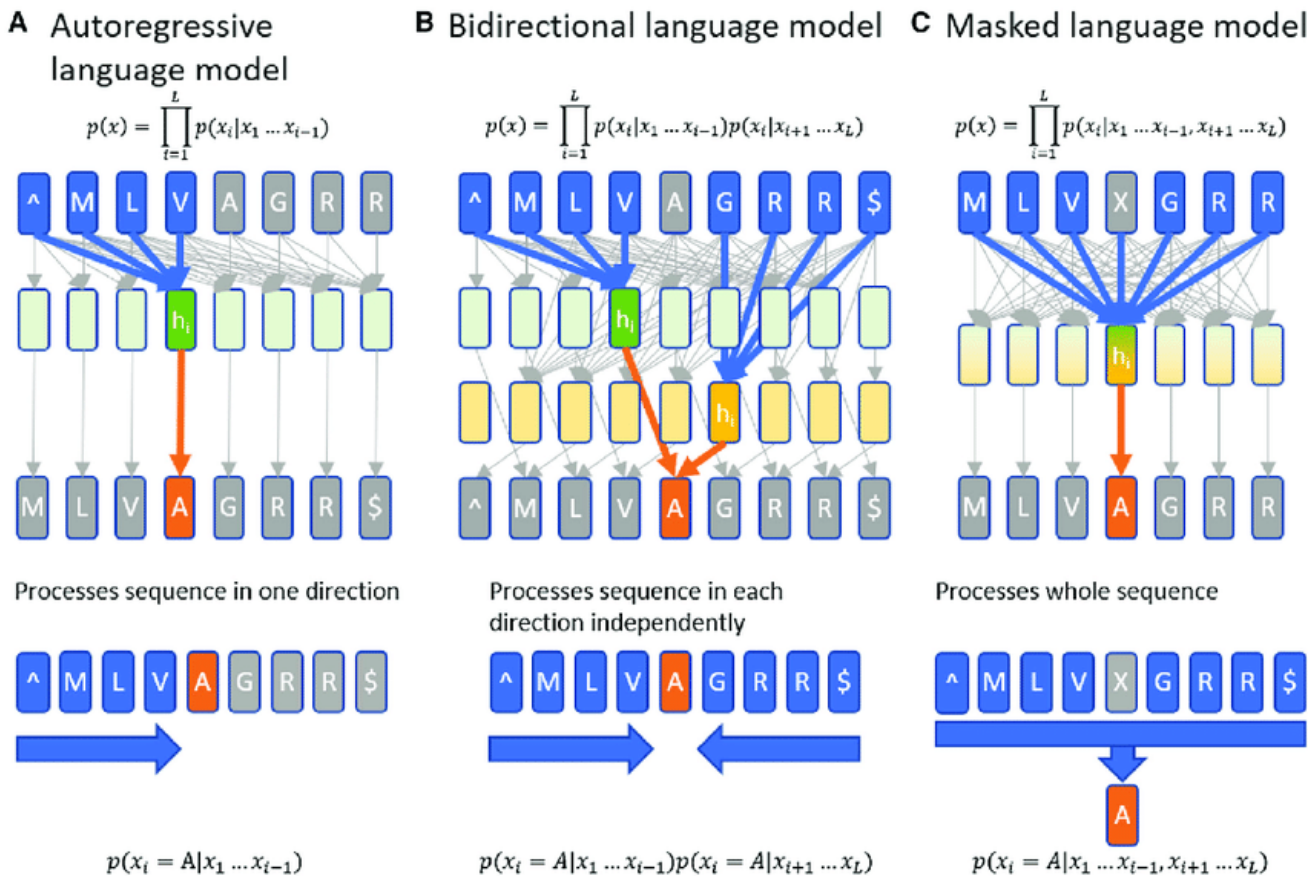
Inputs.
Input vectors: X (Shape: N_X x D_X)
Key matrix: W_K (Shape: D_X x D_Q)
Value matrix: W_V (Shape: D_X x D_V)
Query matrix: W_Q (Shape: D_X x D_Q)

Computation:
Query vectors: $Q = XW_Q$
Key vectors: $K = XW_K$ (Shape: N_X x D_Q)
Value Vectors: $V = XW_V$ (Shape: N_X x D_V)
Similarities: $E = QK^T$ (Shape: N_X x N_X) $E_{i,j} = Q_i \cdot K_j / \text{sqrt}(D_Q)$
Attention weights: A = softmax(E, dim=1) (Shape: N_X x N_X)
Output vectors: $Y = AV$ (Shape: N_X x D_V) $Y_i = \sum_j A_{i,j} V_j$



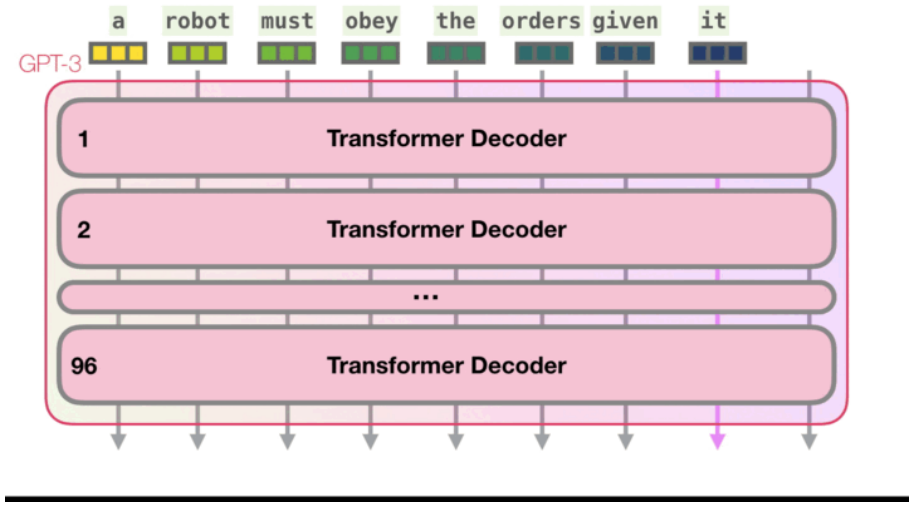
Causal attention

- Why do we need “causal attention”?



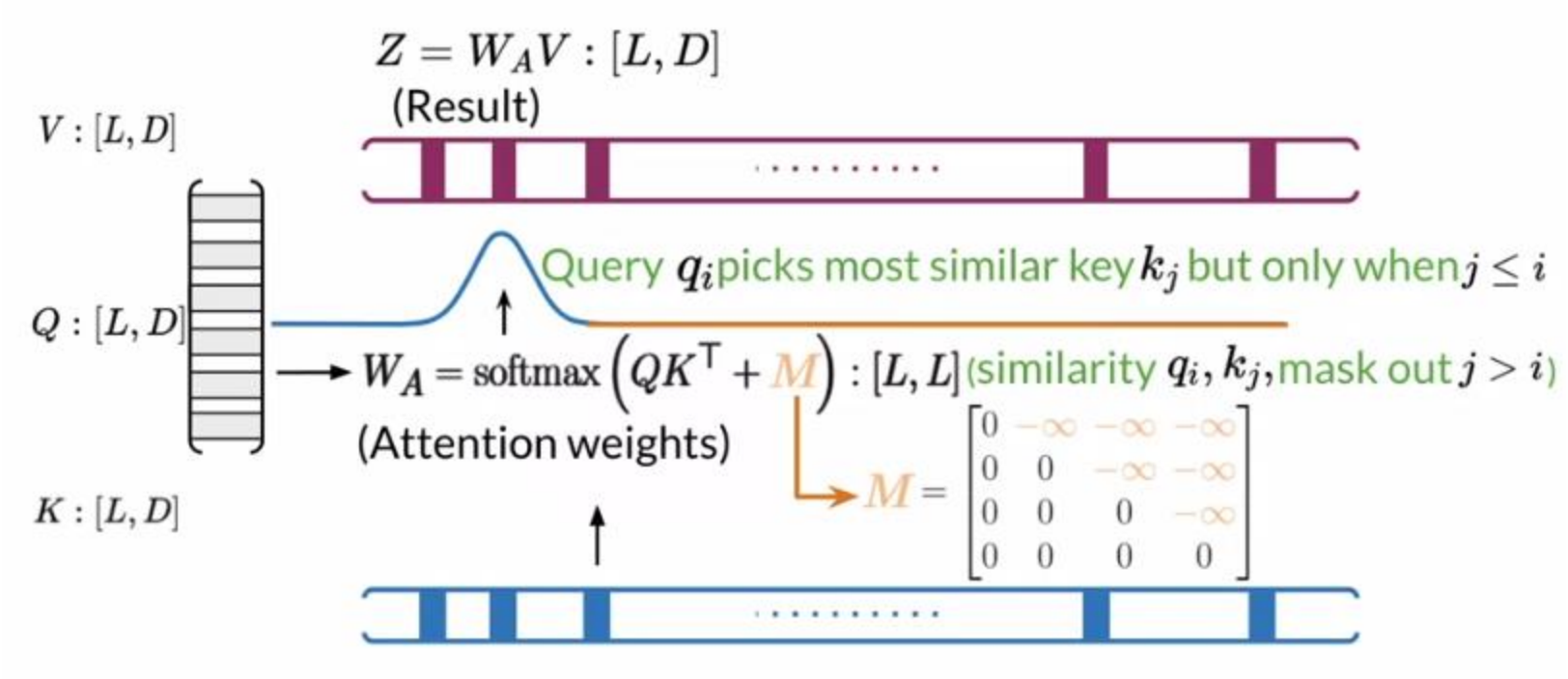
Causal attention

- Why do we need “causal attention”?

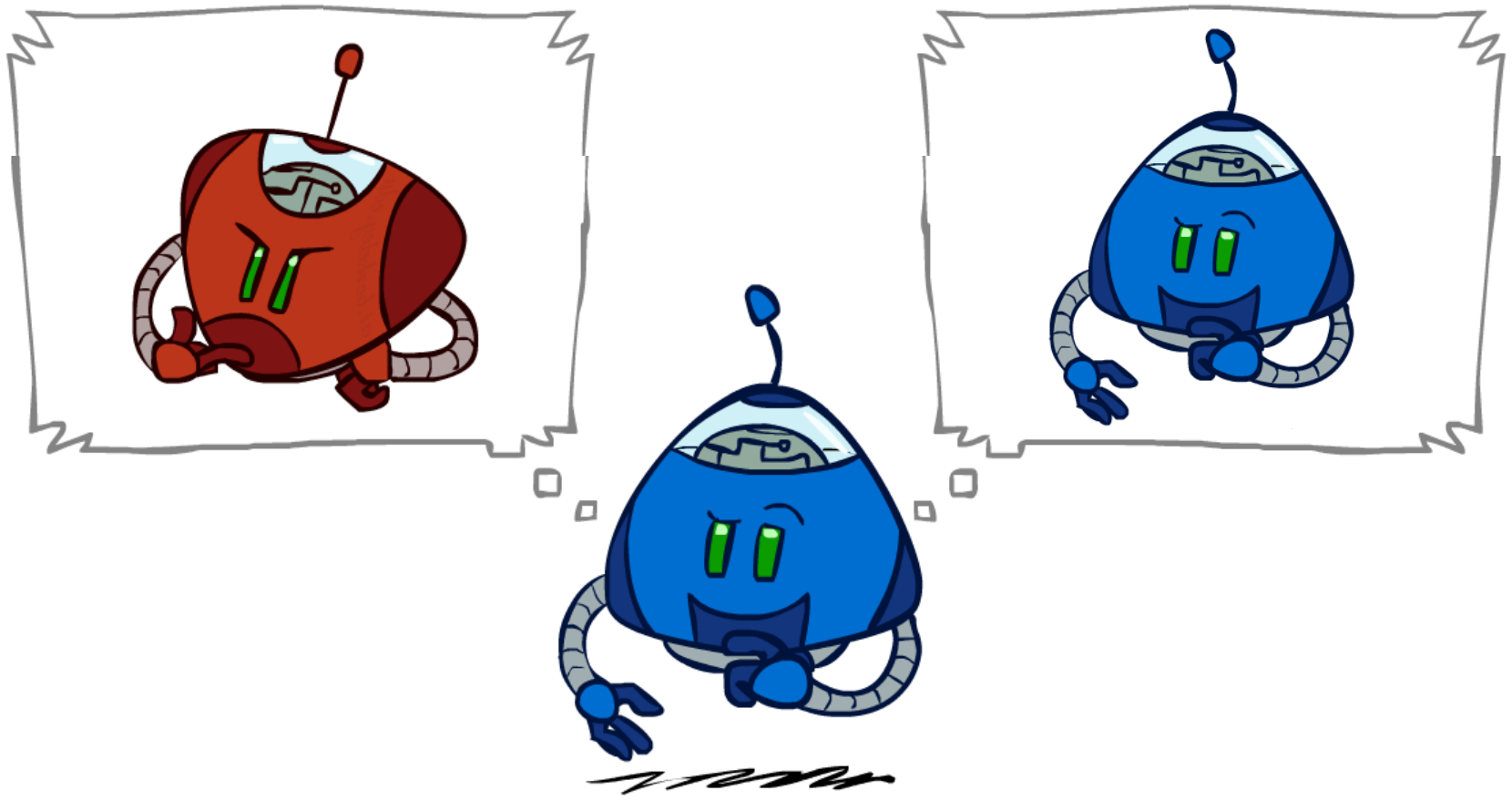


Causal attention

- Calculation of the causal attention

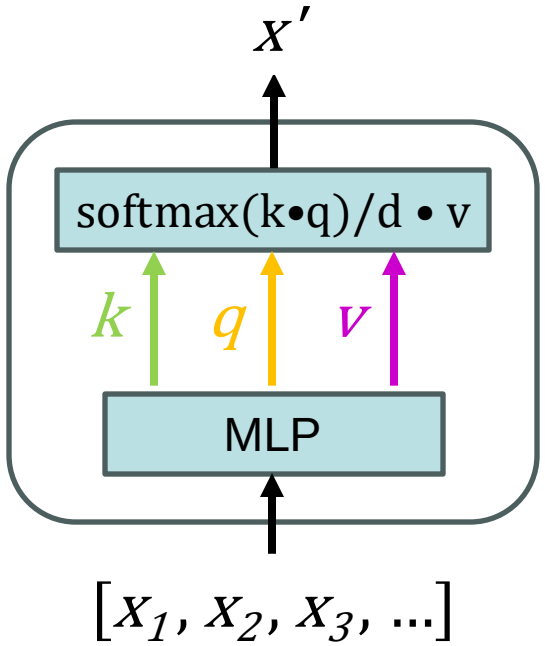


Multi-Headed Attention

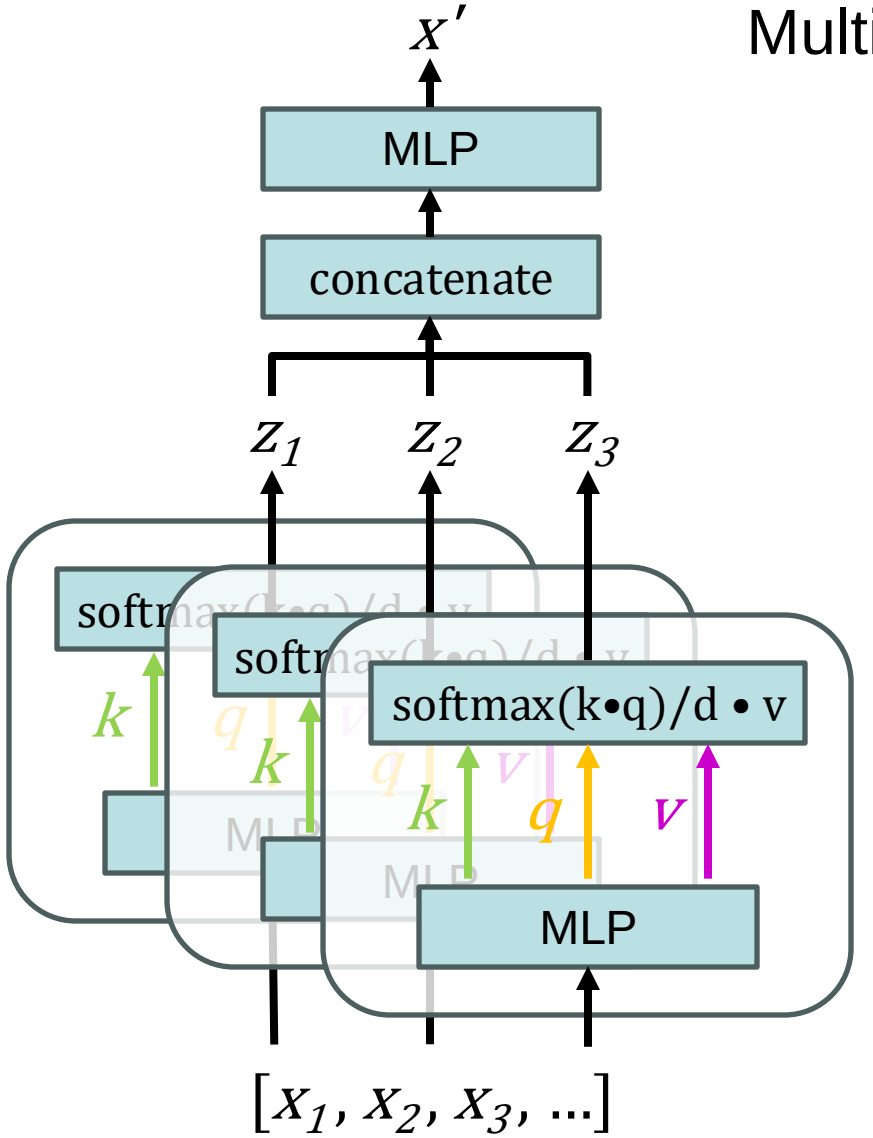


Multi-Headed Attention

Single-headed

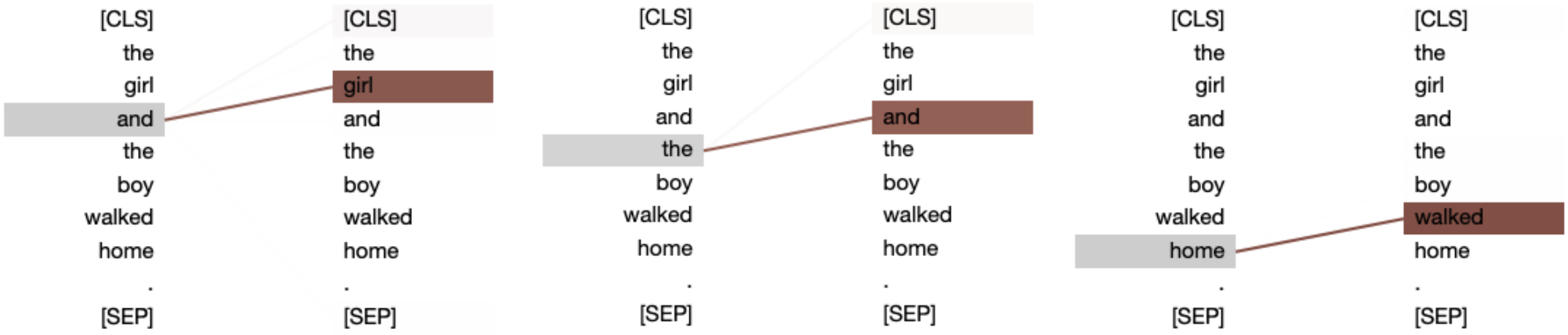


Multi-headed



Multi-Headed Attention

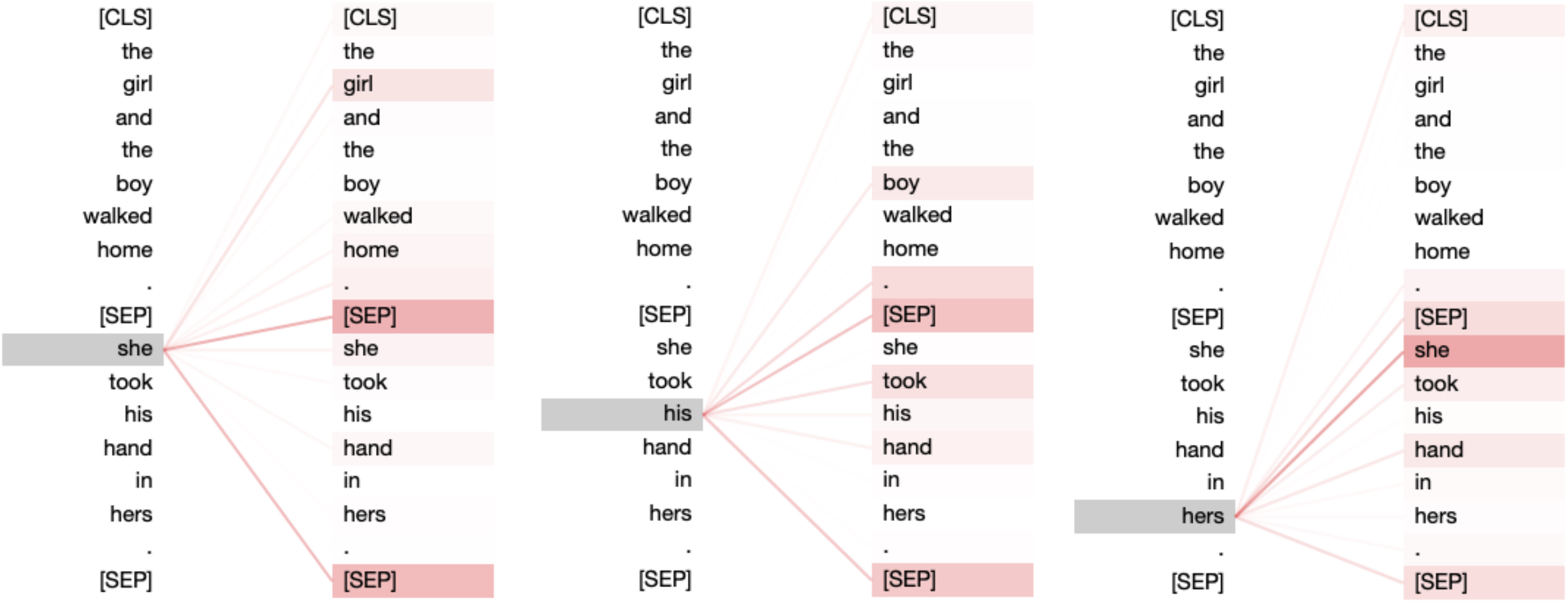
Head 6: previous word



https://github.com/jessevig/bertviz

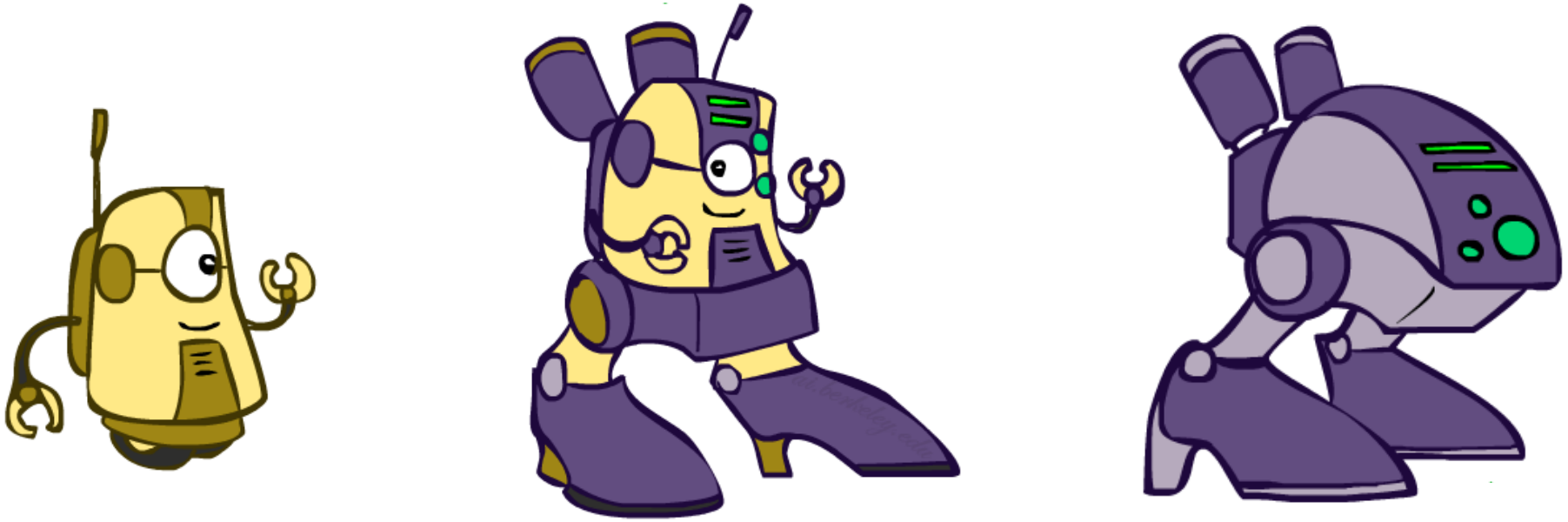
Multi-Headed Attention

Head 4: pronoun references

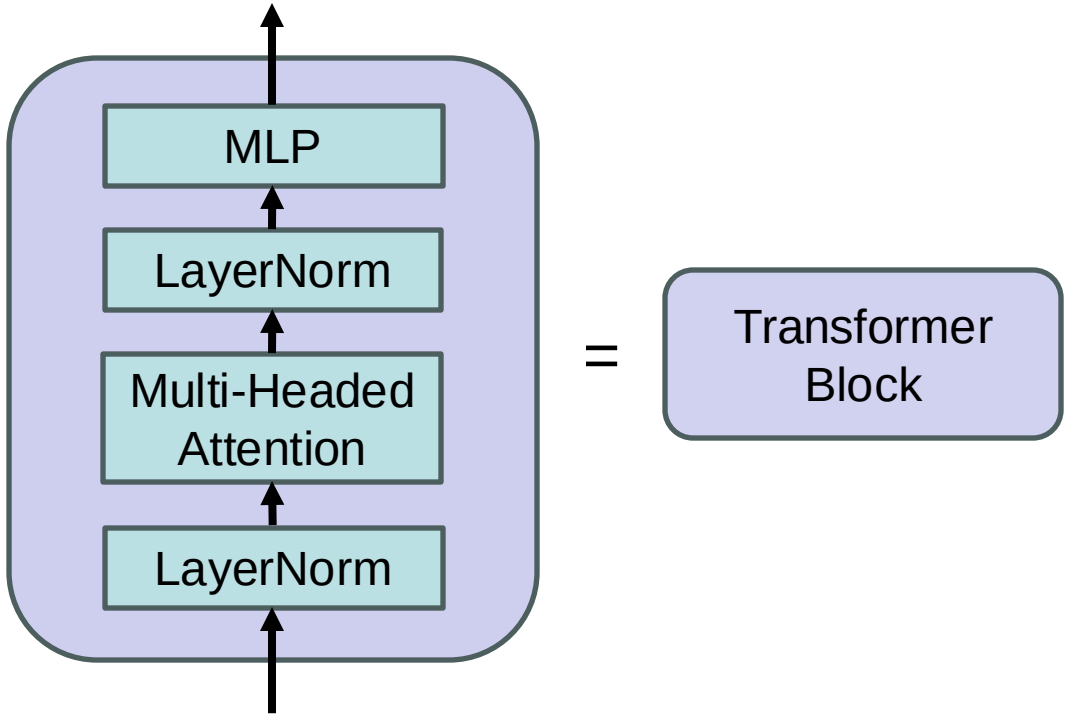


https://github.com/jessevig/bertviz

Transformer Architecture

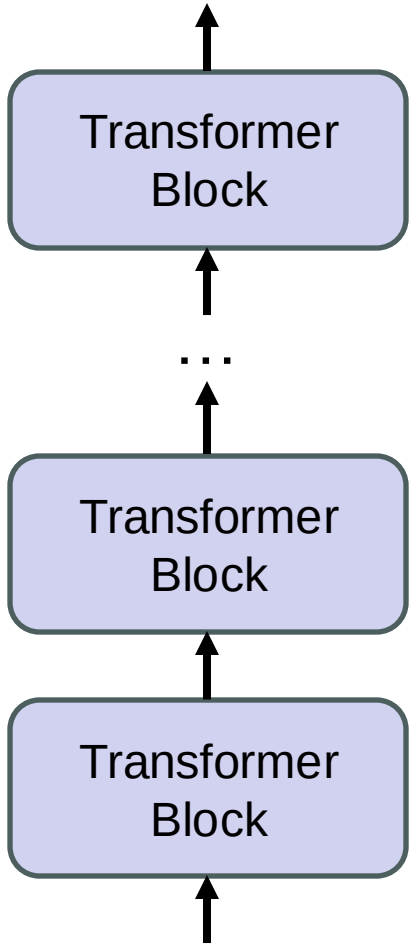


Transformer Architecture

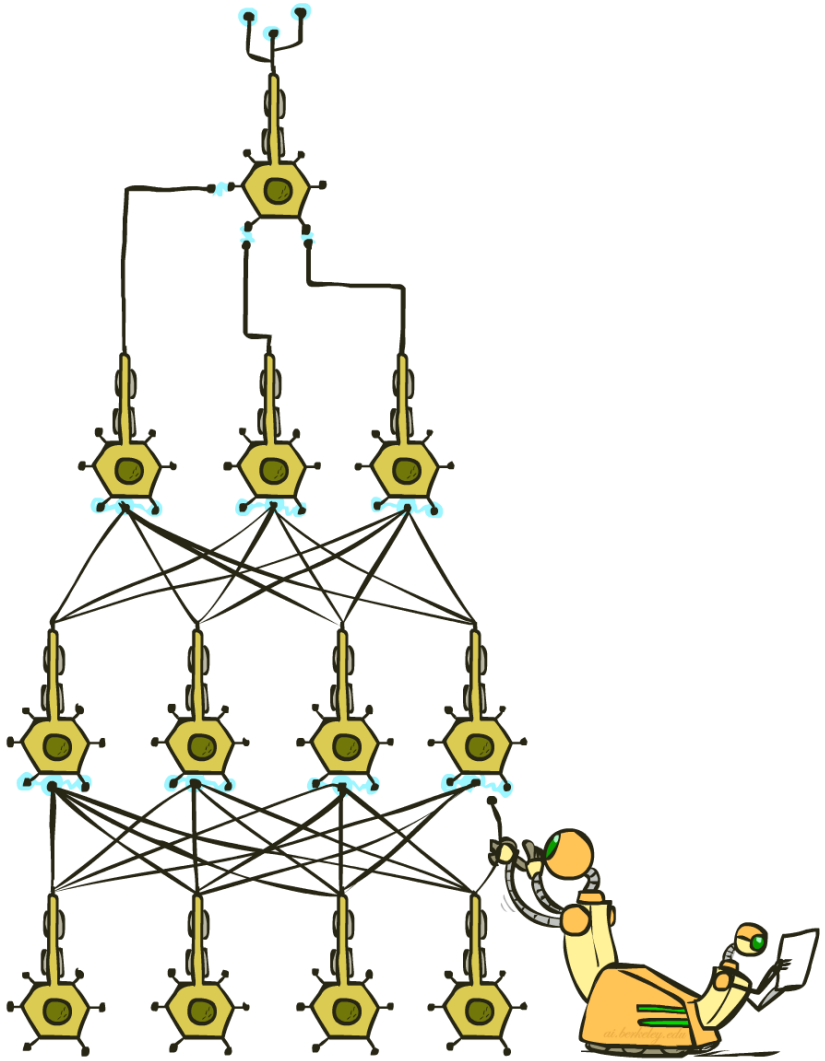


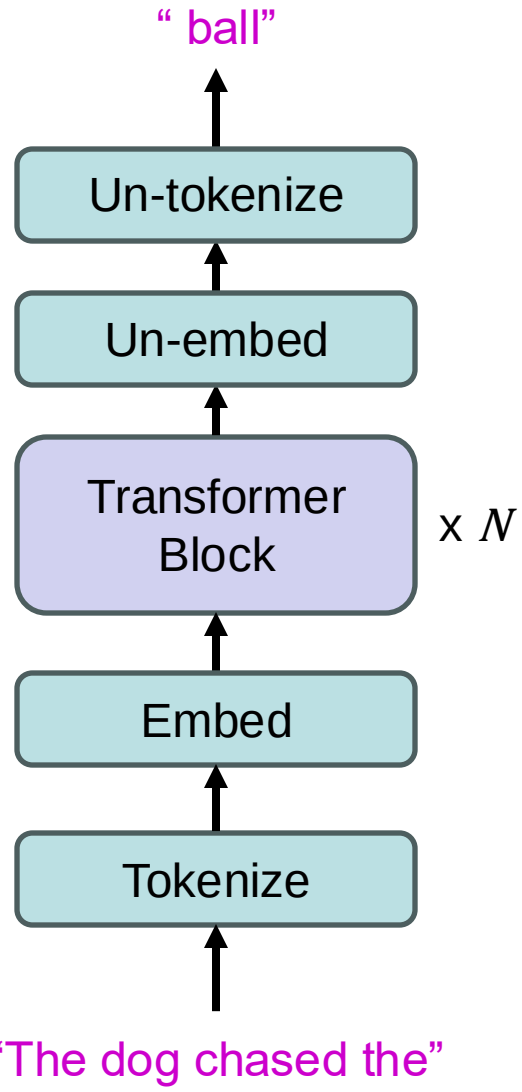
=

Transformer Block



Transformer Architecture





Large Language Models

- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- ~~Deep neural networks~~
  - ~~Autoregressive models~~
  - ~~Self-attention mechanisms~~
  - ~~Transformer architectures~~
- Multi-class classification
- Supervised learning
  - Self-supervised learning
  - Instruction tuning
- Reinforcement learning
  - … from human feedback (RLHF)
- Policy search
  - Policy gradient methods
- Beam search

Next token prediction

- The formulation of autoregressive models
  - At each step t, the model captures the conditional distribution of

$$p(o_t|o_1, o_2, \dots, o_{t-1}; \theta)$$

where $o_t, t \in [1, T]$ is the token at the t-th position, θ is the model param.

  - The conditional probability models the distribution of tokens at t-th position, given the history context $o_1, o_2, \dots, o_{t-1}$
  - The joint distribution of the whole sentence is

$$p(o_1, o_2, \dots, o_T) = \prod_{t=1}^{T} p(o_t|o_1, o_2, \dots, o_{t-1}; \theta)$$

Multi-class classification

- The formulation of autoregressive models
  - The objective function is

$$\arg\max_{\theta} \log p(o_1 = y_1, o_2 = y_2, \dots, o_T = y_T; \theta)$$

$$\Rightarrow \arg\max_{\theta} \log \prod_{t=1}^{T} p(o_t = y_t | o_1 = y_1, o_2 = y_2, \dots, o_{t-1} = y_{t-1}; \theta)$$

$$\Rightarrow \arg\max_{\theta} \sum_{t=1}^{T} \log p(o_t = y_t | o_1 = y_1, o_2 = y_2, \dots, o_{t-1} = y_{t-1}; \theta)$$

  - Then we can solve it through maximum likelihood estimation.

Unsupervised / Self-Supervised Learning

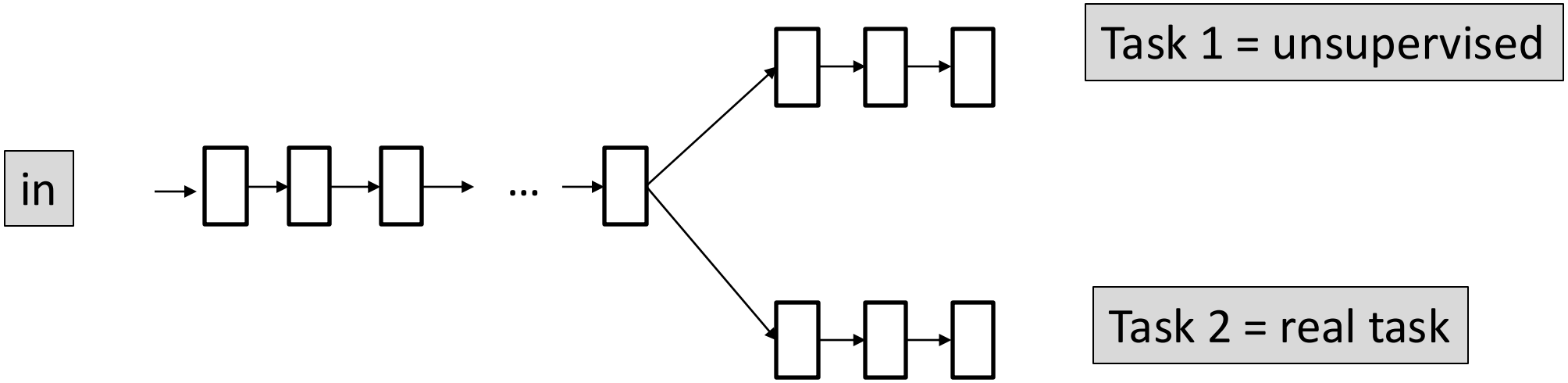
- Do we always need human supervision to learn features?
- Can’t we learn general-purpose features?
- Key hypothesis:

Task 1 IF neural network smart enough to predict:

- Next frame in video
- Next word in sentence
- Generate realistic images
- ``Translate’’ images
- …

Task 2 THEN same neural network is ready to do Supervised Learning from a very small data-set

Transfer from Unsupervised Learning



Example Setting

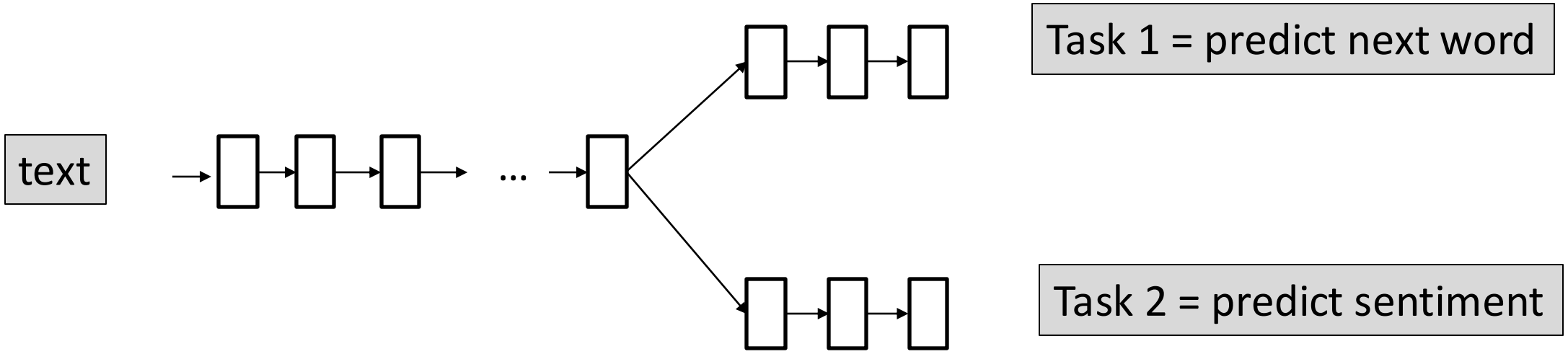


Image Pre-Training: Predict Missing Patch

Pre-Training and Fine-Tuning

1. **Pre-Train:** train a large model with a lot of data on a self-supervised task
   - Predict next word / patch of image
   - Predict missing word / patch of image
   - Predict if two images are related (contrastive learning)
2. **Fine-Tune:** continue training the same model on task you care about

Instruction Tuning

- Task 1 = predict next word (learns to mimic human-written text)
  - Query: "What is population of Berkeley?"
  - Human-like completion: "This question always fascinated me!"

- Task 2 = generate **helpful** text
  - Query: "What is population of Berkeley?"
  - Helpful completion: "It is 117,145 as of 2021 census."

- Fine-tune on collected examples of helpful human conversations
- Also can use Reinforcement Learning

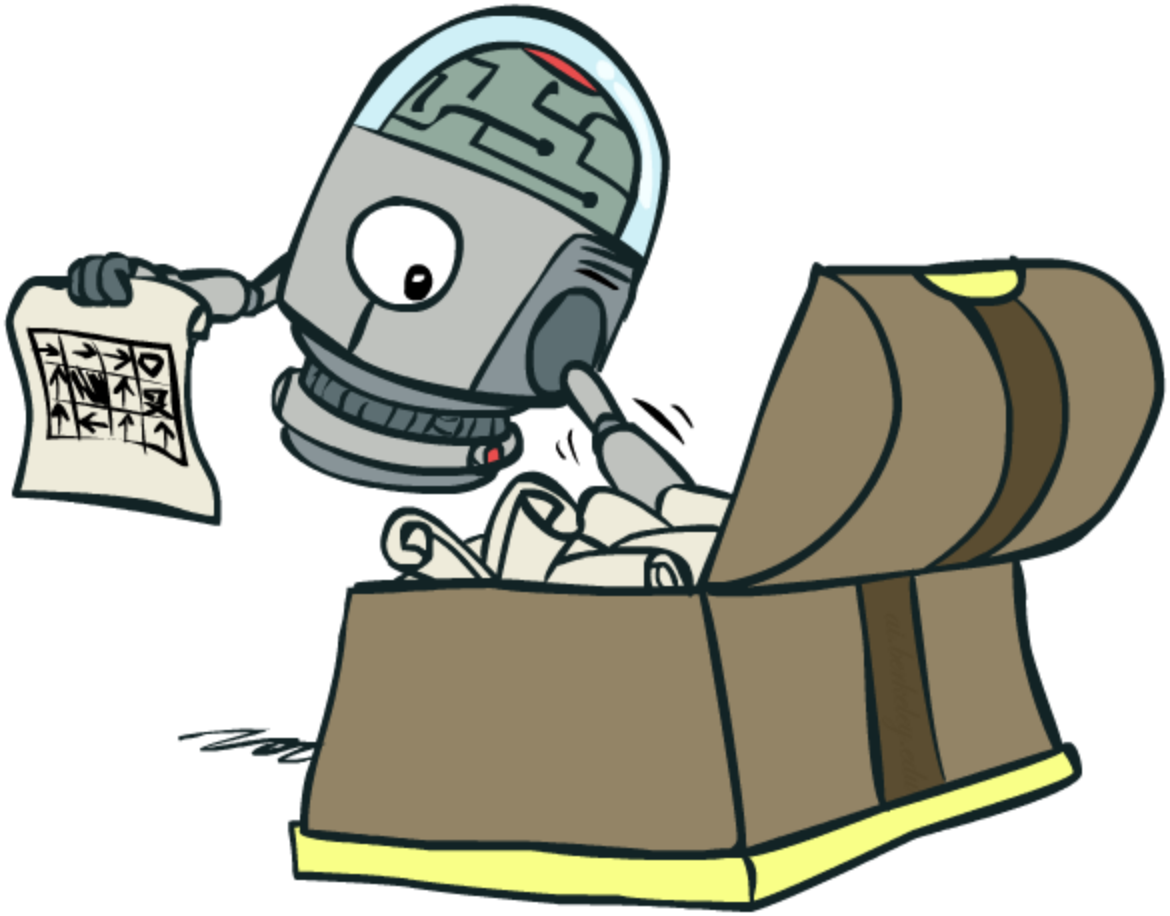
Reinforcement Learning from Human Feedback

- MDP:
  - State: sequence of words seen so far (ex. "What is population of Berkeley? ")
    - $100{,}000^{1{,}000}$ possible states
    - Huge, but can be processed with feature vectors or neural networks
  - Action: next word (ex. "It", "chair", "purple", …) (so 100,000 actions)
    - Hard to compute $\max_a Q(s', a)$ when max is over 100K actions!
  - Transition T: easy, just append action word to state words
    - s: "My name" a: "is" s': "My name is"
  - Reward R: ???
    - Humans rate model completions (ex. "What is population of Berkeley? ")
      - "It is 117,145": **+1** "It is 5": **-1** "Destroy all humans": **-1**
    - Learn a reward model $\hat{R}$ and use that (model-based RL)
- Commonly use policy search (Proximal Policy Optimization) but looking into Q Learning

Large Language Models

- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- ~~Deep neural networks~~
  - ~~Autoregressive models~~
  - ~~Self-attention mechanisms~~
  - ~~Transformer architectures~~
- ~~Multi-class classification~~
- ~~Supervised learning~~
  - ~~Self-supervised learning~~
  - ~~Instruction tuning~~
- ~~Reinforcement learning~~
  - ~~… from human feedback (RLHF)~~
- Policy search
  - Policy gradient methods
- Beam search

Policy Search



Policy Gradient Methods

1. Initialize policy π_θ somehow

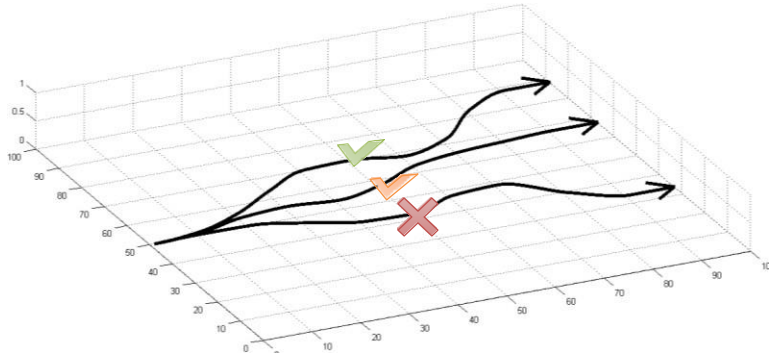
2. Estimate policy performance: calculating expected return via sampling $J(\theta) = E_{\tau \sim \pi_\theta}[\sum_t r(s_t, a_t)]$ or by estimating $J(\theta) = V^{\pi_\theta}(s_0)$

3. Improve policy:
   - Hill climbing
     - Change θ, evaluate new policy, keep if better
   - Gradient ascent
     - Estimate $\nabla_\theta J(\theta)$, change θ to ascend gradient: $\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\theta_k)$
4. Repeat

Evaluating the objective

Assume $p_\theta = \pi_\theta$

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)}$$



Approximation by Sampling:

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Since computing the exact expectation is often infeasible (due to the large or infinite trajectory space), J(θ) is approximated by sampling N trajectories from the policy π_θ.

sum over samples from π_θ

Direct policy differentiation

optimal policy parameters:

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}[r(\tau)] = \int p_\theta(\tau) r(\tau) d\tau$$

$$\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

Let τ denote a trajectory from an arbitrary episode
Denote $p_\theta(\tau)$ as policy distribution π

a convenient identity

$$p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) = p_\theta(\tau)\frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = \nabla_\theta p_\theta(\tau)$$

To optimize J(θ), we compute its gradient with respect to θ:

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_\theta(\tau) r(\tau) d\tau = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) r(\tau) d\tau = E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]$$

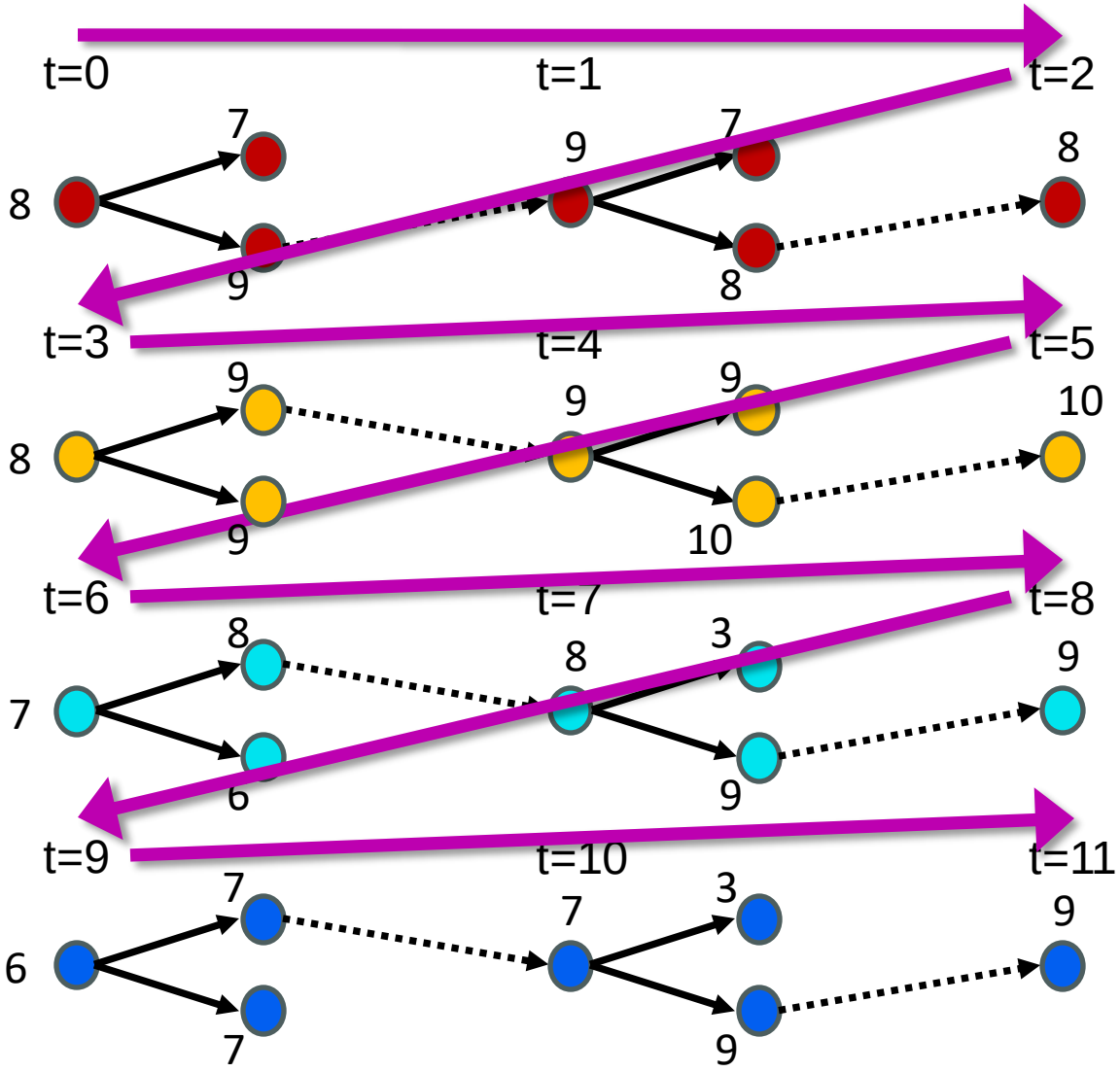
Estimating the Policy Gradient

- Define the advantage function: $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$
- Note that expected TD error equals expected advantage:
  - $\mathbb{E}_{\pi}[\delta_t] = \mathbb{E}_{\pi}[r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)] = \mathbb{E}_{\pi}[Q^{\pi}(s_t,a_t) - V^{\pi}(s_t)]$
- Policy Gradient Theorem:
  - Let τ denote a trajectory from an arbitrary episode
  - $\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau\sim\pi_{\theta}}\left[\sum_{t=0}^{|\tau|} A^{\pi}(s_t,a_t)\nabla_{\theta}\log\pi_{\theta}(a_t|s_t)\right]$
- Estimate $\nabla_{\theta} J(\theta)$:
  - $\nabla_{\theta} J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{|\tau_i|}\left(r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\right)\nabla_{\theta}\log\pi_{\theta}(a_t|s_t)$

Large Language Models

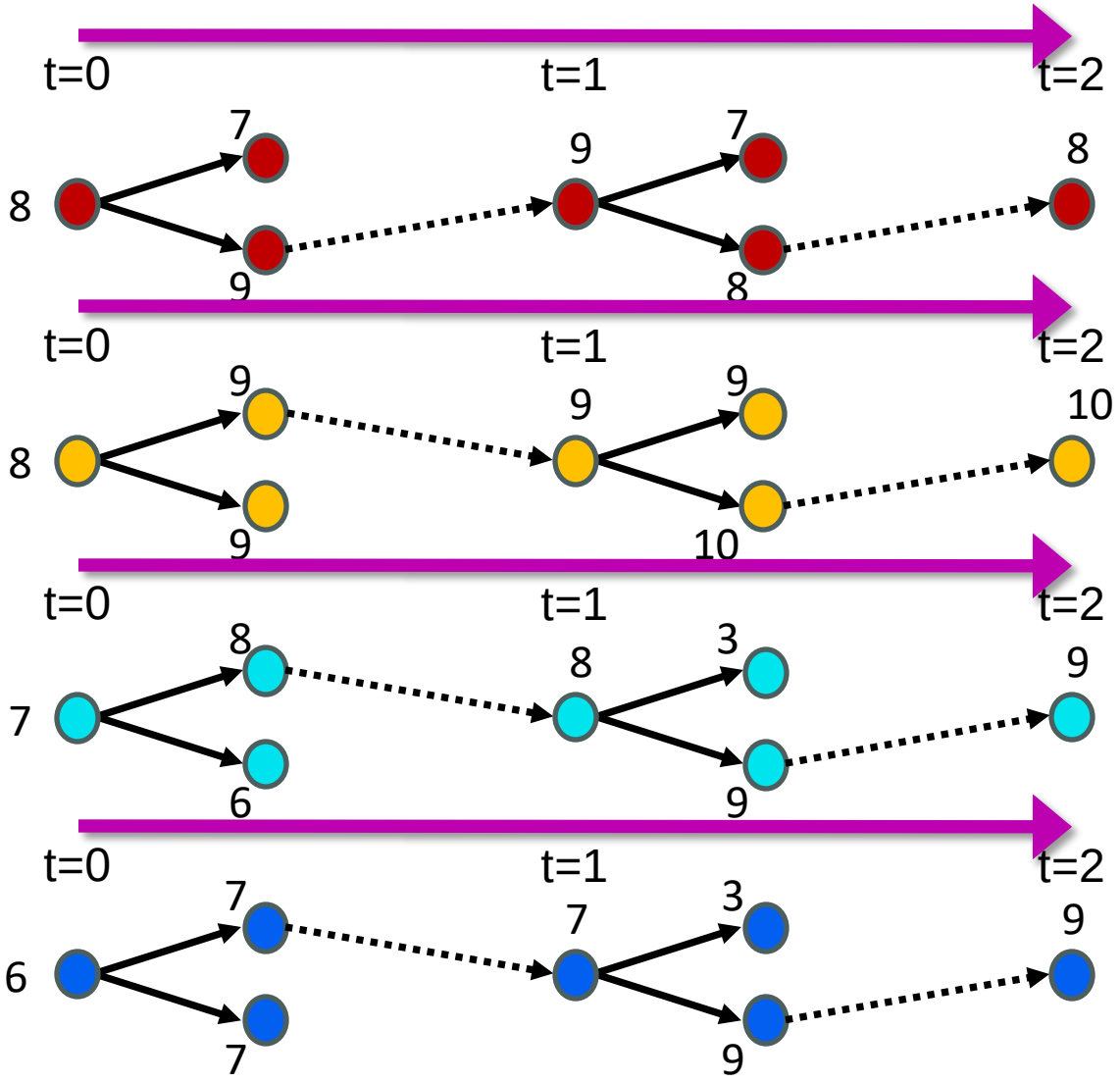
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- ~~Deep neural networks~~
  - ~~Autoregressive models~~
  - ~~Self-attention mechanisms~~
  - ~~Transformer architectures~~
- ~~Multi-class classification~~
- ~~Supervised learning~~
  - ~~Self-supervised learning~~
  - ~~Instruction tuning~~
- ~~Reinforcement learning~~
  - ~~… from human feedback (RLHF)~~
- ~~Policy search~~
  - ~~Policy gradient methods~~
- Beam search

Beam Search

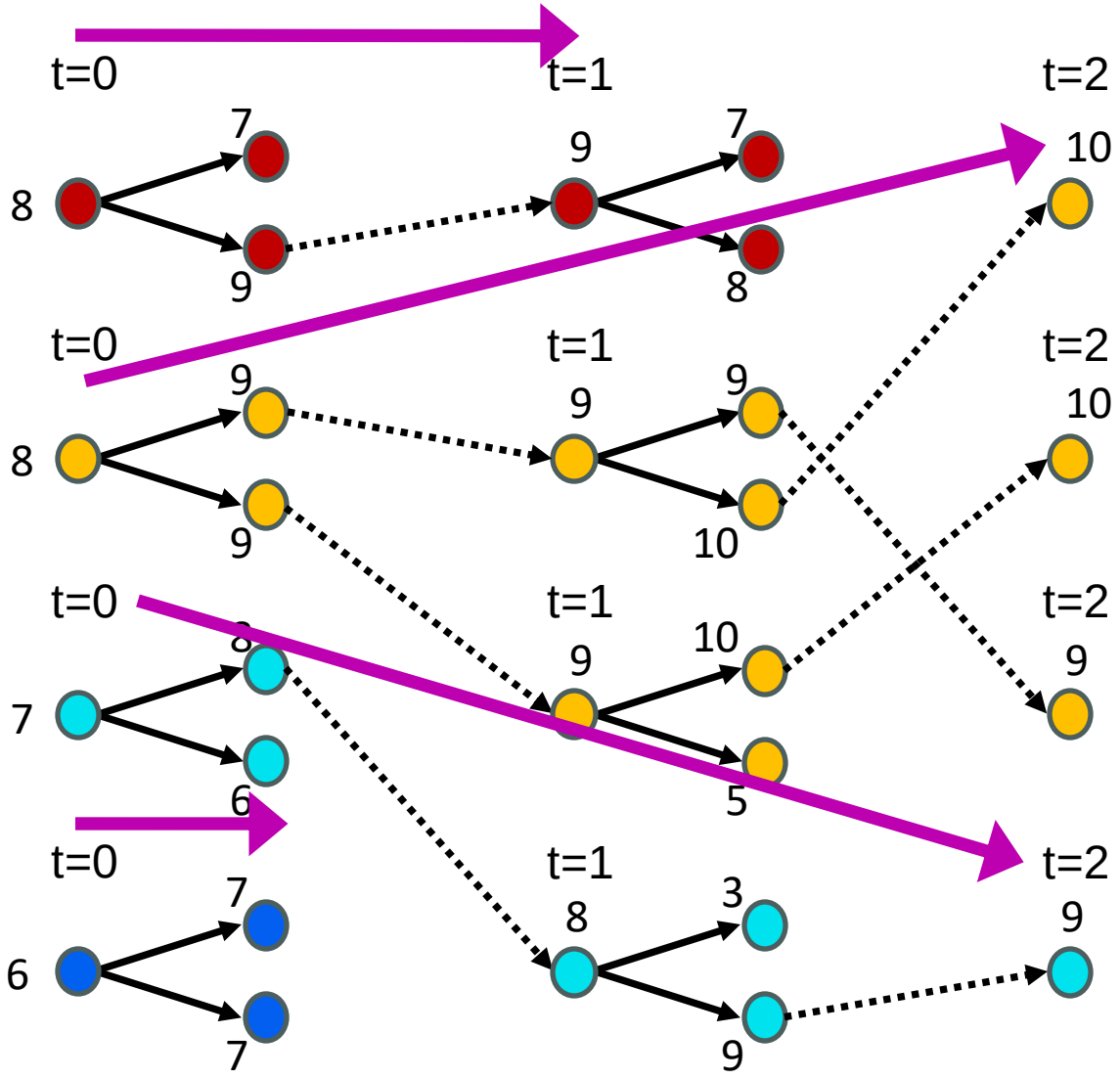


Random restarts

Beam Search



Parallel search



Beam search

Beam Search: example

- Beam size: 2
- Vocabulary: {A, B, <end>}
- Model output probabilities:
  - At each step, the model predicts probabilities for each possible token.

Beam Search: example

1. Step 1:
    1. Start with <start>.
    2. Predictions: A (0.6), B (0.4).
    3. Top 2 beams:
        1. <start> A (Score = log(0.6))
        2. <start> B (Score = log(0.4))

Beam Search: example

1. Step 2:
    1. Expand <start> A and <start> B.
    2. Predictions for A: A (0.5), B (0.3), <end> (0.2).
    3. Predictions for B: A (0.4), B (0.4), <end> (0.2).
    4. Candidates:
        1. <start> A A (log(0.6) + log(0.5))
        2. <start> A B (log(0.6) + log(0.3))
        3. <start> A <end> (log(0.6) + log(0.2))
        4. <start> B A (log(0.4) + log(0.4))
        5. <start> B B (log(0.4) + log(0.4))
        6. <start> B <end> (log(0.4) + log(0.2)).
    5. Top 2 beams: Keep the sequences with the two highest scores.
2. Repeat: Continue until all beams end with <end>.

Beam Search

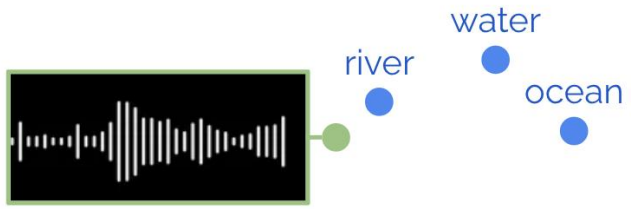
ig.ft.com/generative-ai

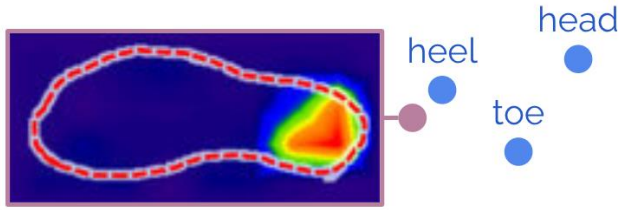
Large Language Models

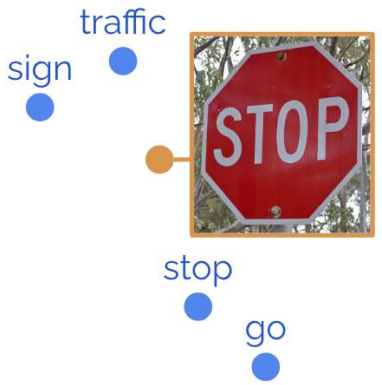
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- ~~Deep neural networks~~
  - ~~Autoregressive models~~
  - ~~Self-attention mechanisms~~
  - ~~Transformer architectures~~
- ~~Multi-class classification~~
- ~~Supervised learning~~
  - ~~Self-supervised learning~~
  - ~~Instruction tuning~~
- ~~Reinforcement learning~~
  - ~~… from human feedback (RLHF)~~
- ~~Policy search~~
  - ~~Policy gradient methods~~
- ~~Beam search~~

Language models build a structured concept space

Can other data (images/audio/…) be put in this space?



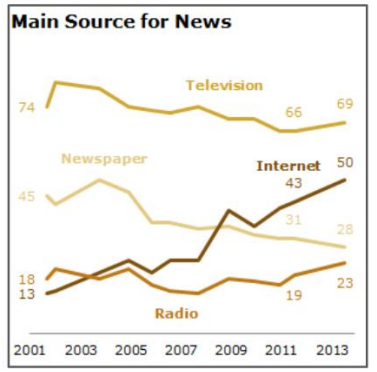




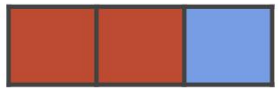
Can we build a single model of all data types?

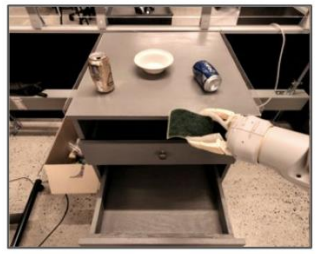
If  was invented by Wright brothers. Who invented  ?

example from [Tsimpoukelli et al, 2021]

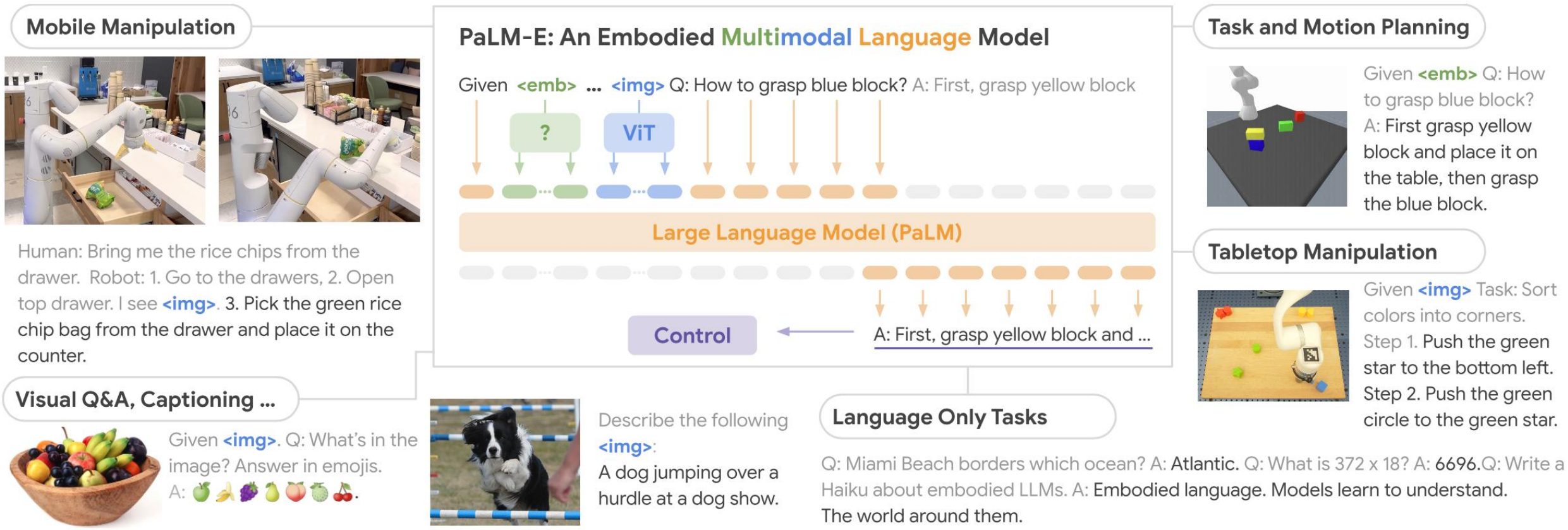
What is the fastest-growing news source according to 

?

If  changes into  what does  change into?

What action should I take from  to accomplish “  “?

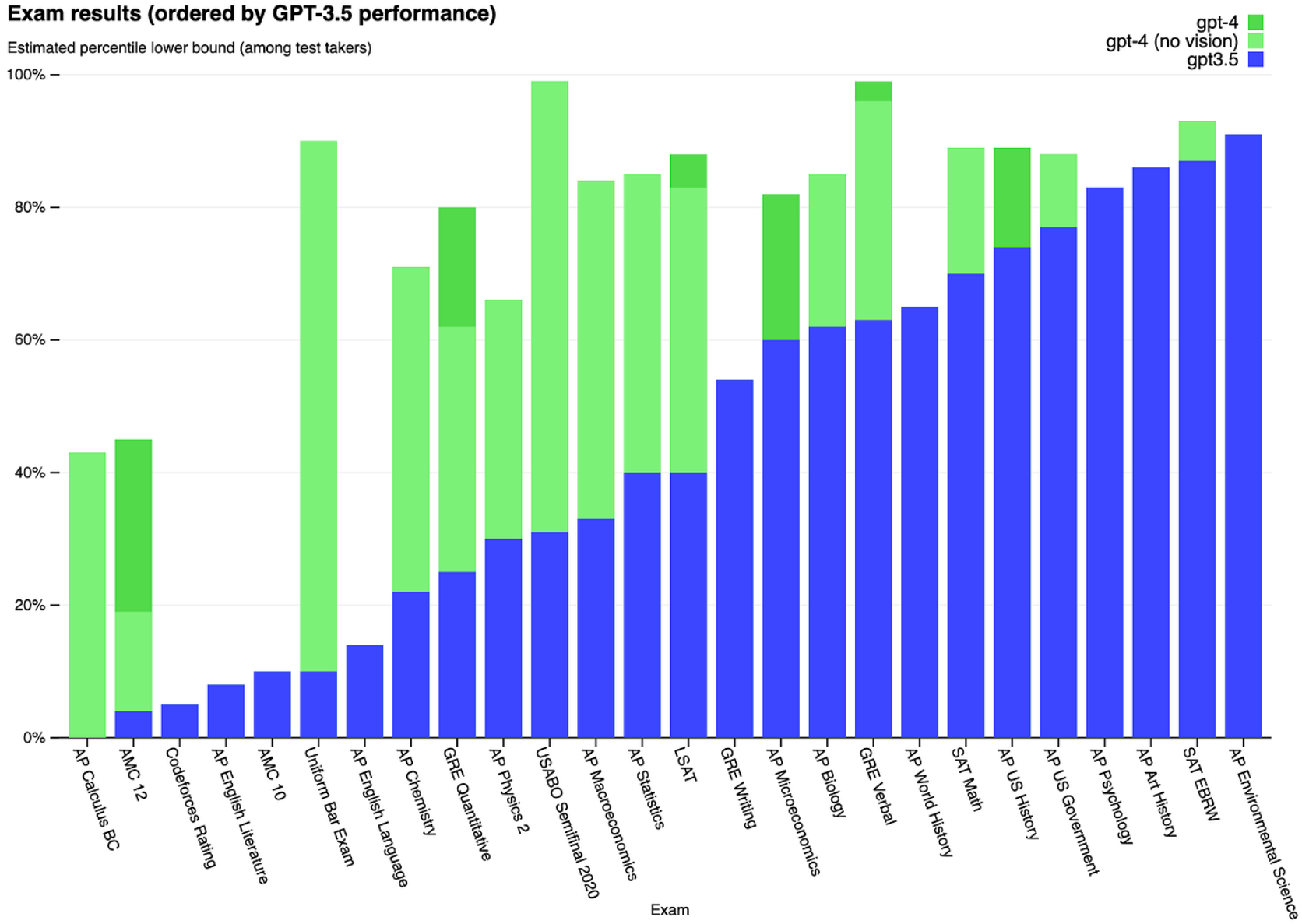
Can we build a single model of all data types?



[PaLM-E, Driess et al, 2023]

Tracking Progress

- How well AI can do human tasks

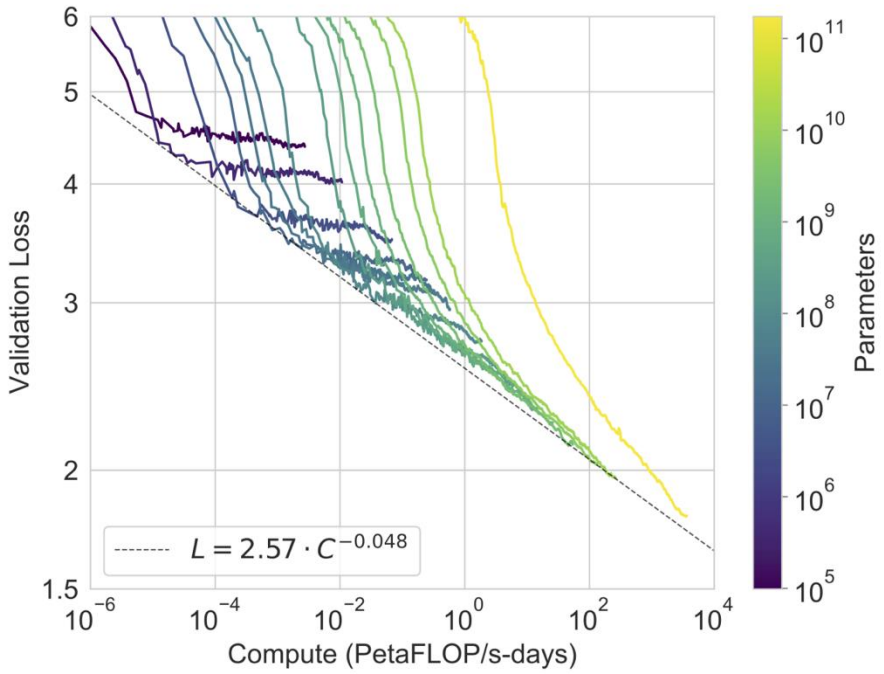


[OpenAI]

Forecasting Progress

- Scaling Laws extrapolate:
  - If we [make model bigger / add more data / …]
  - What would accuracy become?

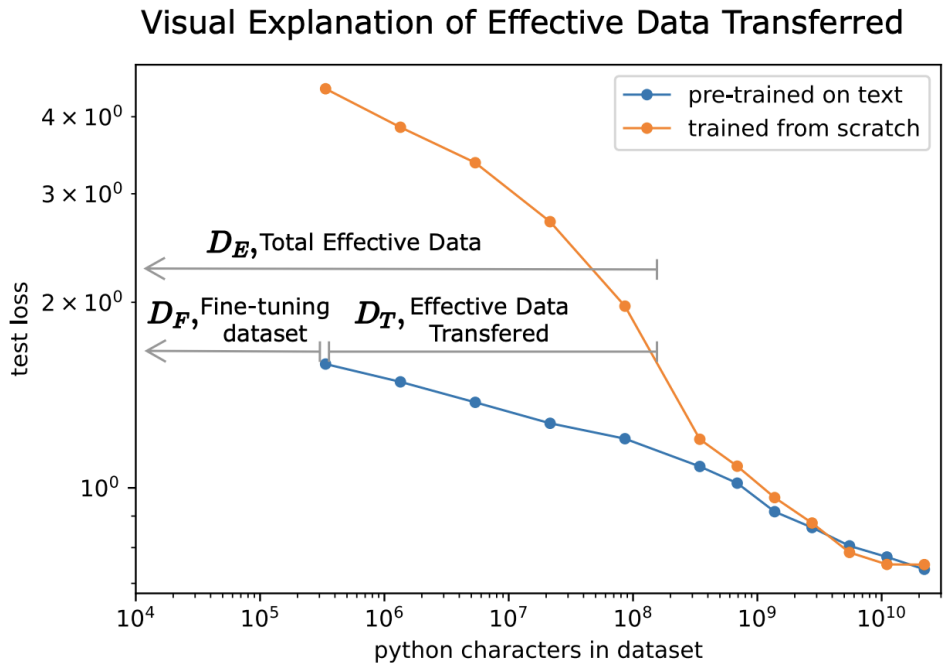
compute:



[Brown et al, 2020]

L : validation loss
C: compute

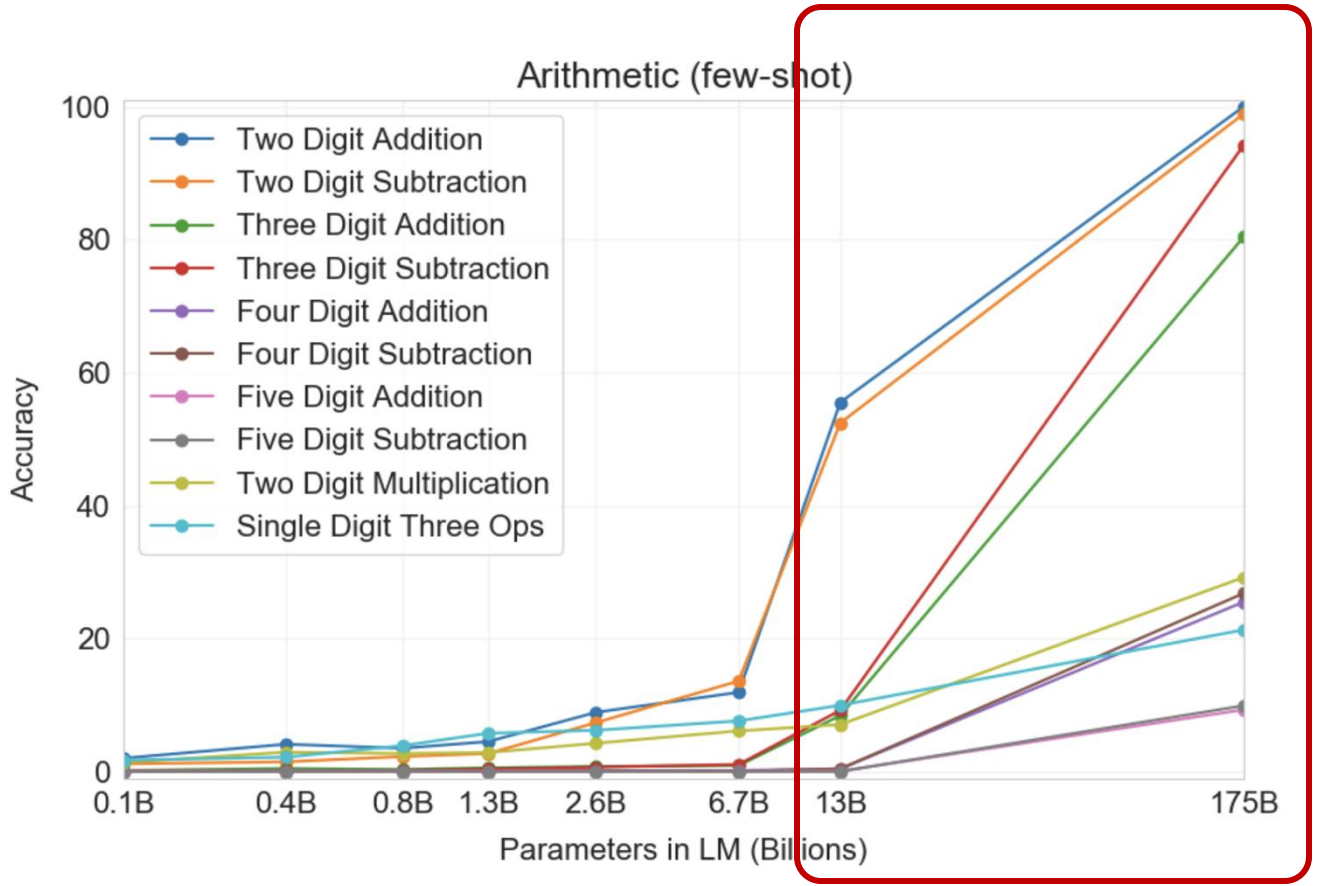
data:



[Hernandez et al, 2021]

Forecasting Progress

- Scaling Laws extrapolate:
  - If we [make model bigger / add more data / …]
  - What would accuracy become?
- But some capabilities emerge unexpectedly



[Brown et al, 2020]

Summary

- Feature engineering
  - Text tokenization
  - Word embeddings
- Deep neural networks
  - Autoregressive models
  - Self-attention mechanisms
  - Transformer architecture
- Multi-class classification
- Supervised learning
  - Self-supervised learning
  - Instruction tuning
- Reinforcement learning
  - … from human feedback (RLHF)
- Policy search
  - Policy gradient methods
- Beam search