Advanced topics of AI

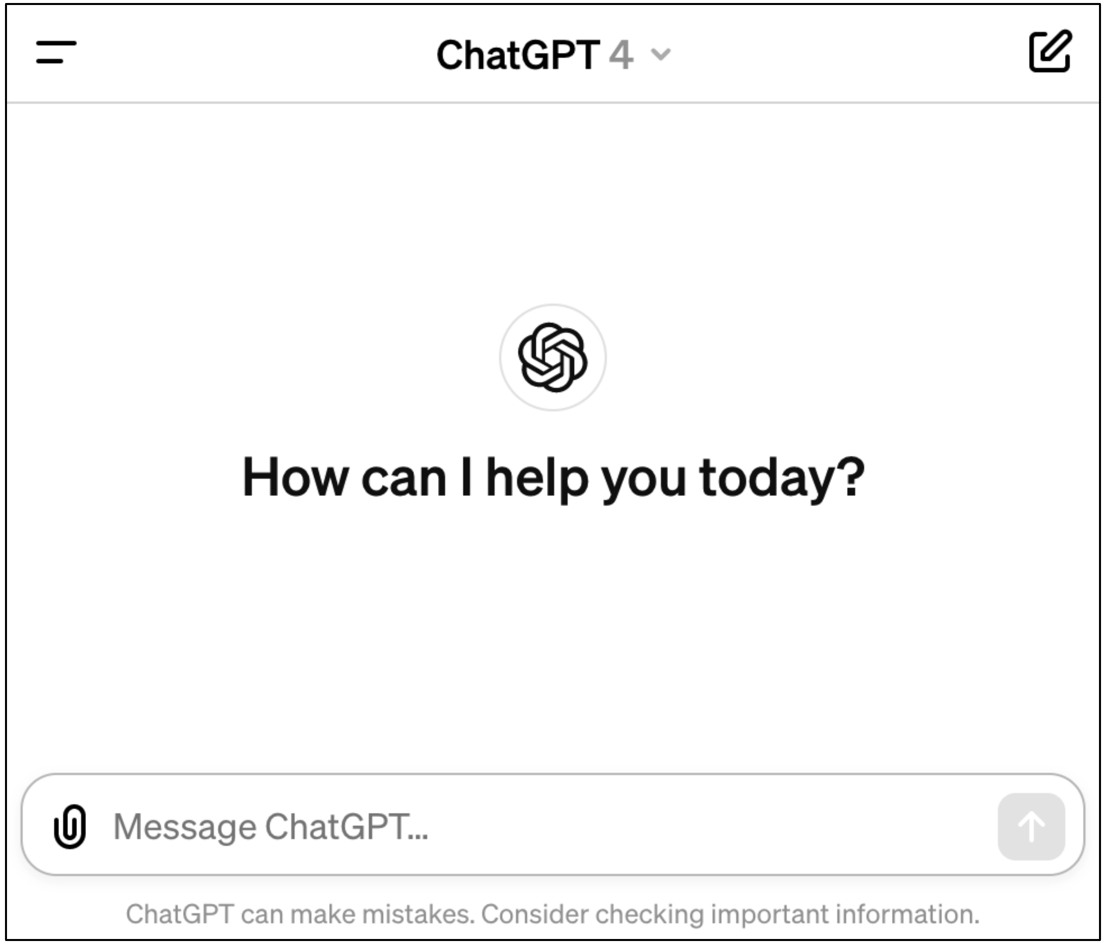
Large Language Models

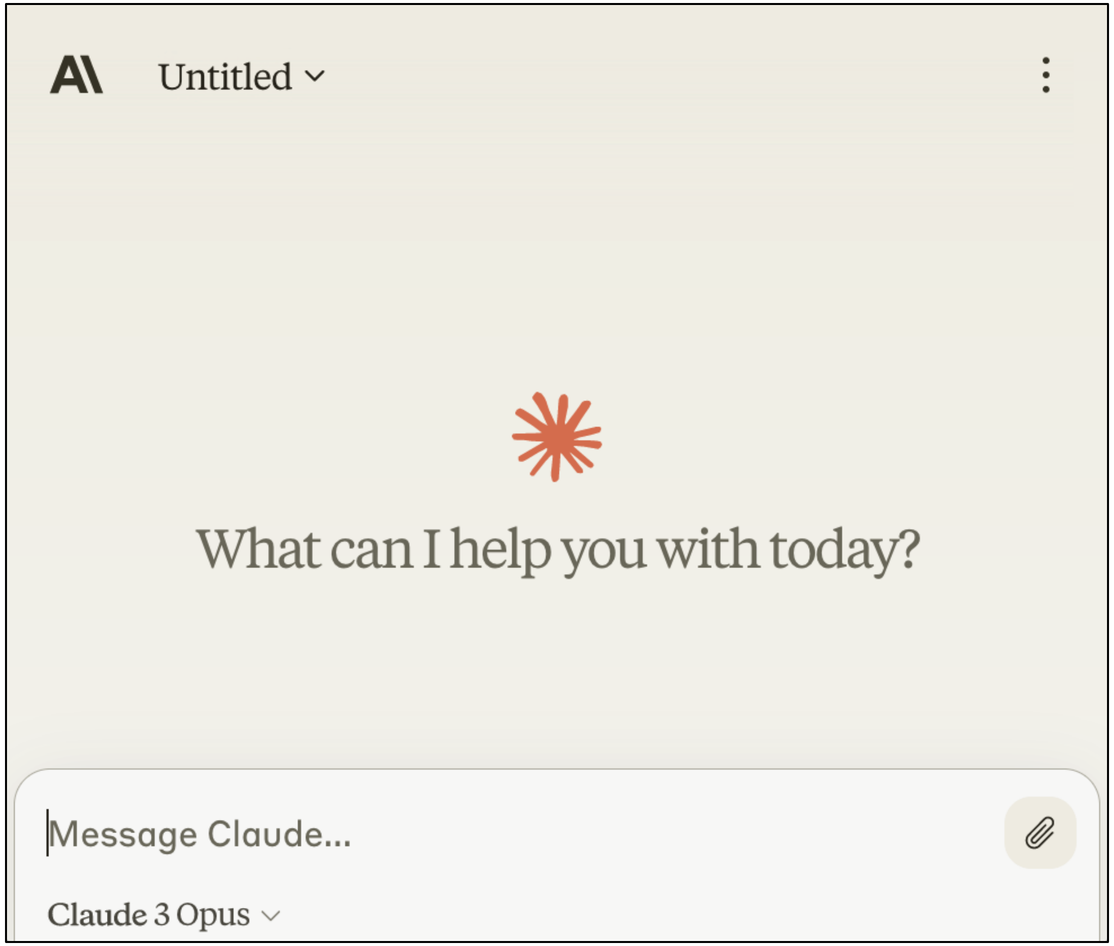
[Adapted from slides by Dan Klein and Pieter Abbeel at UC Berkeley]

Advanced topics of AI

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

Today's AI

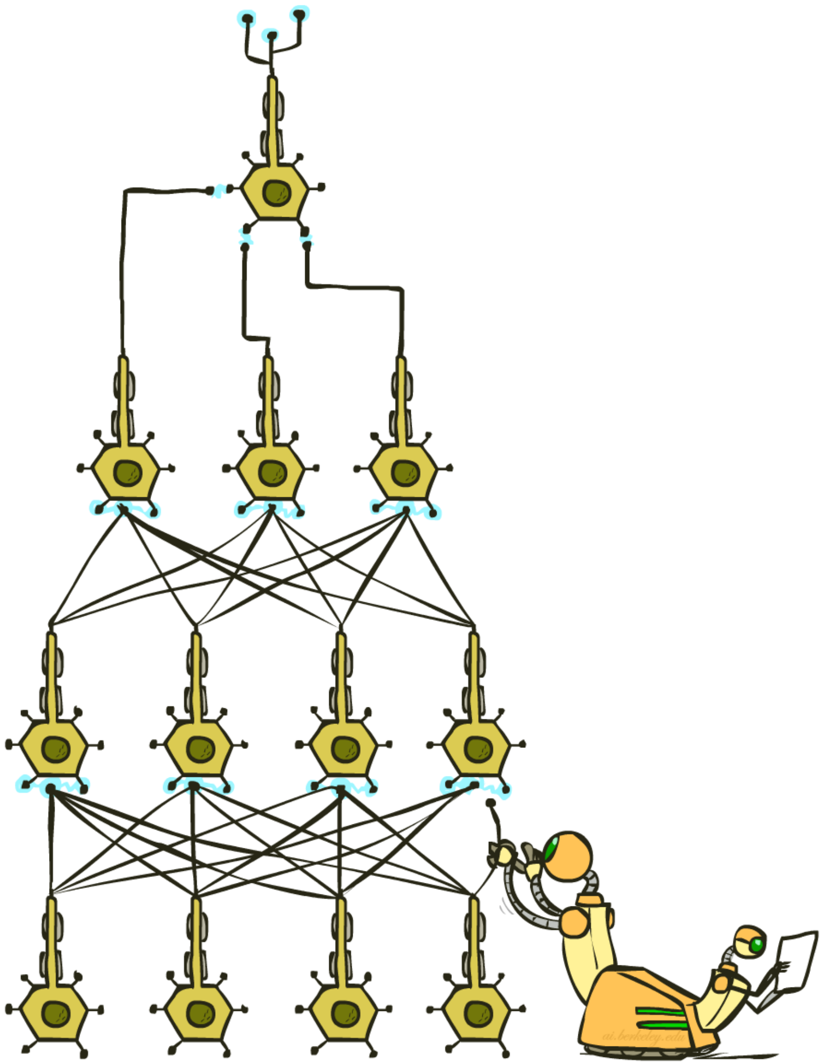

ChatGPT 4
How can I help you today?
Message ChatGPT...
ChatGPT can make mistakes. Consider checking important information.


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 57

Characters 252

https://platform.openai.com/tokenizer

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

Text Token IDs

Tokens 57

Characters 252

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]
```

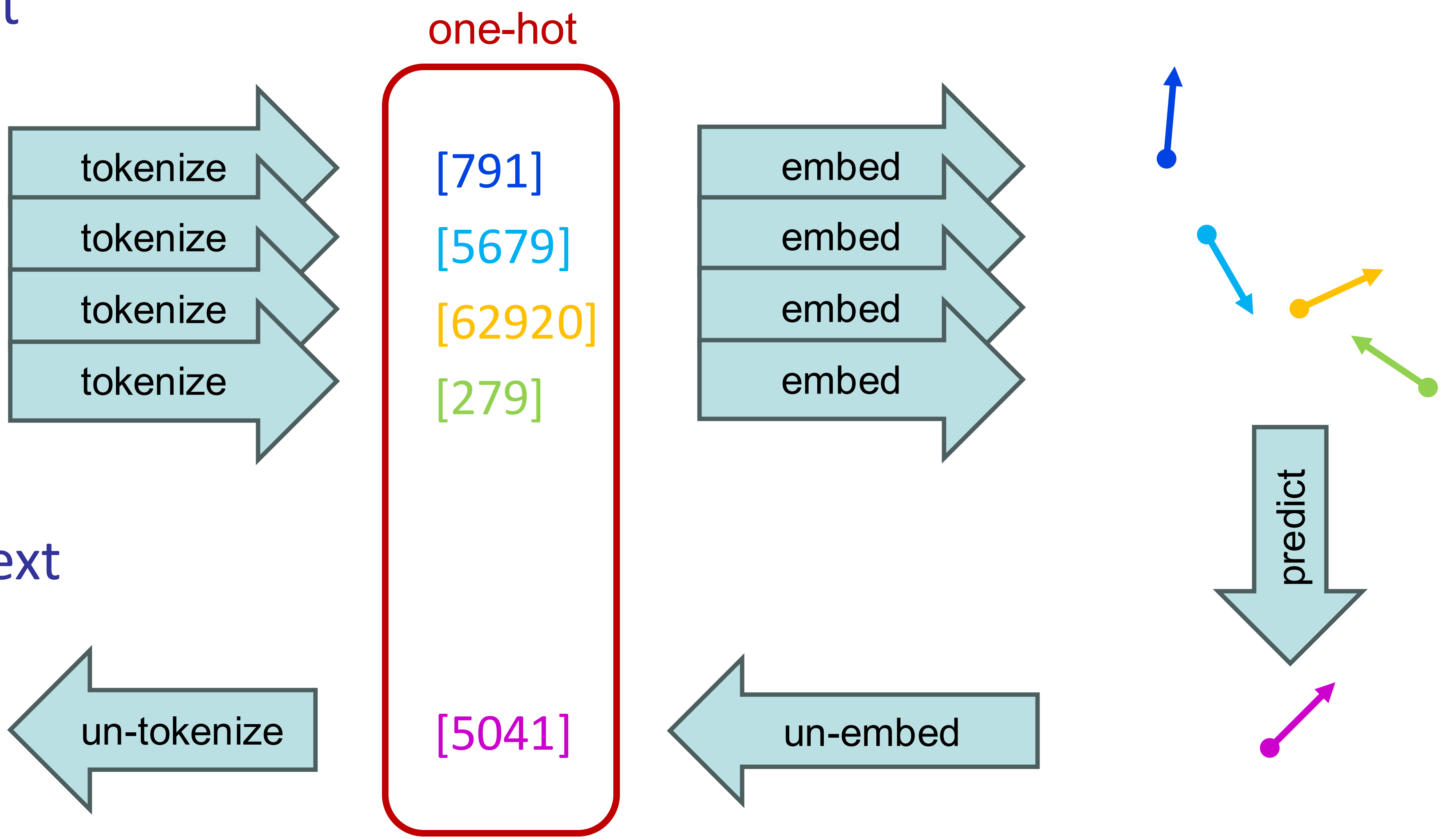
Text Token IDs

Tokens	Characters
57	252

https://platform.openai.com/tokenizer

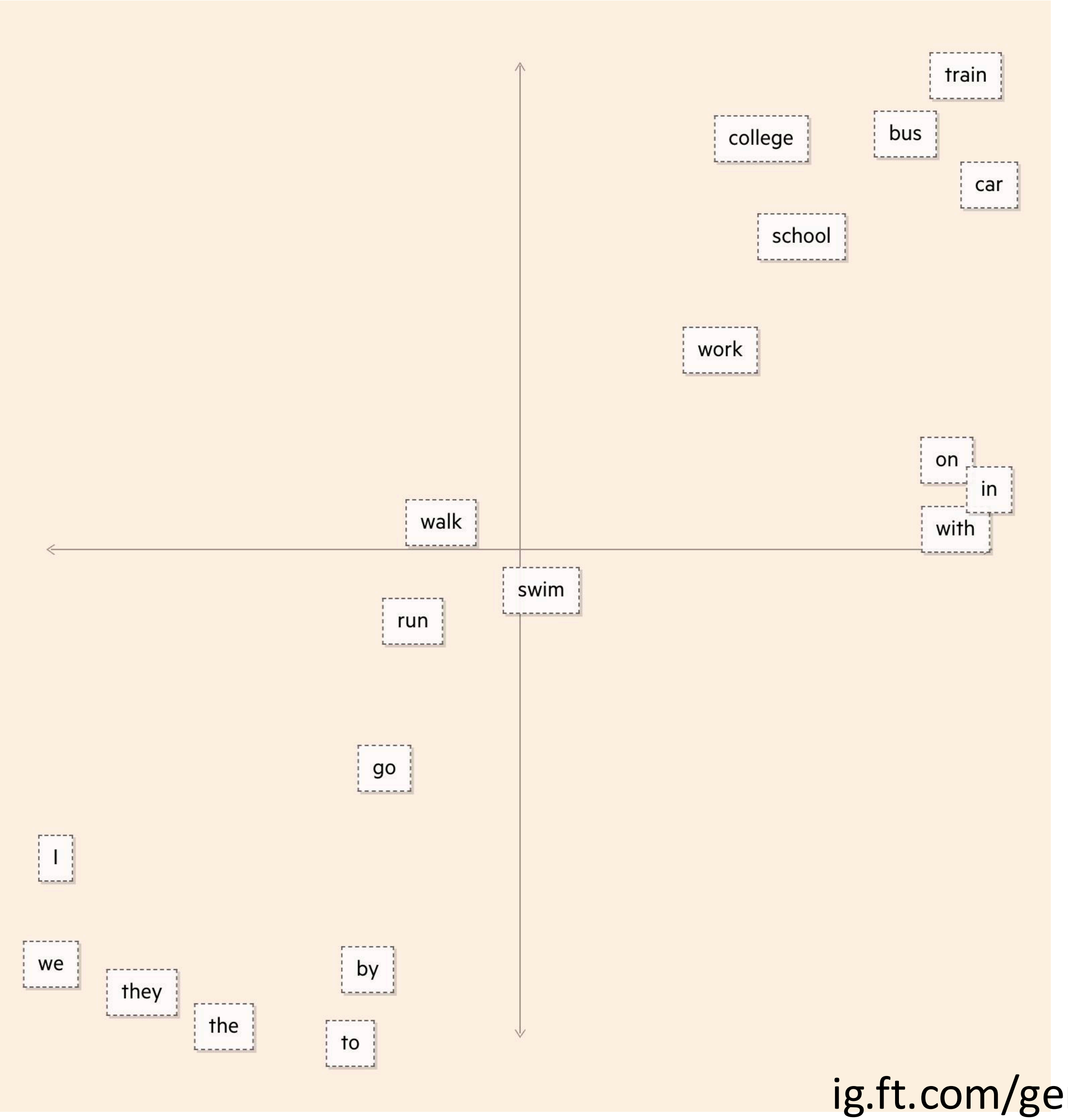
Word Embeddings

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



What do word embeddings look like?

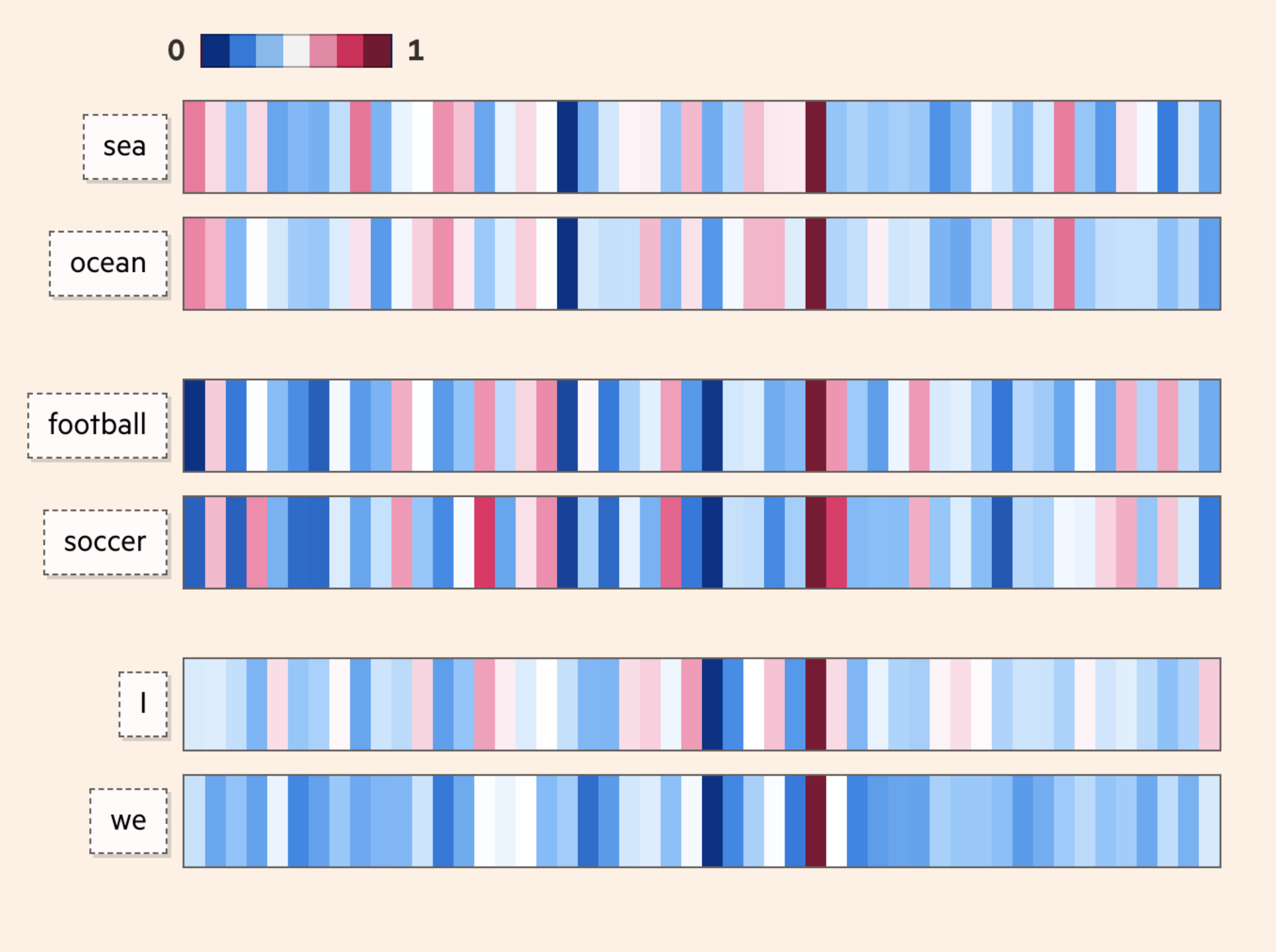
- Words cluster by similarity:



ig.ft.com/generative-ai

What do word embeddings look like?

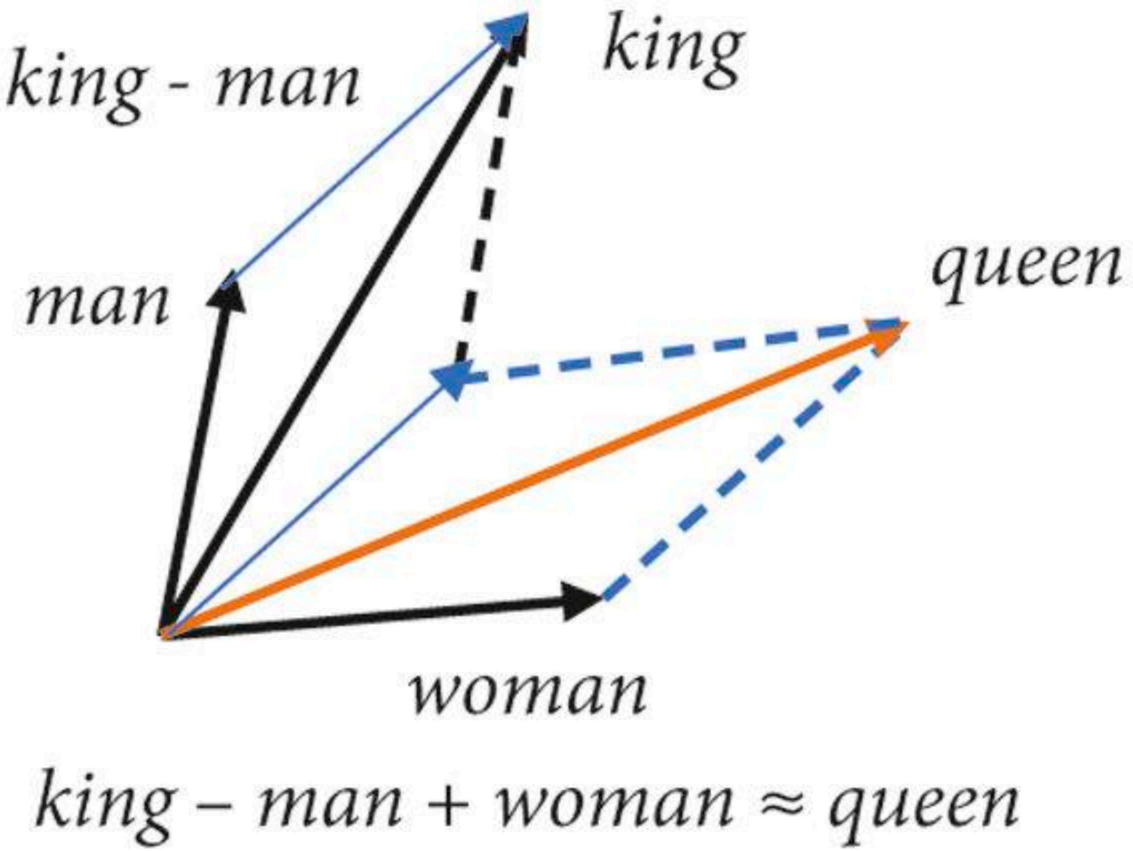
- Features learned in language models:



ig.ft.com/generative-ai

What do word embeddings look like?

- Signs of sensible algebra in embedding space:



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

Aside: interactive explainer of modern language models

ig.ft.com/generative-ai

Artificial Intelligence

Generative AI exists because of the transformer

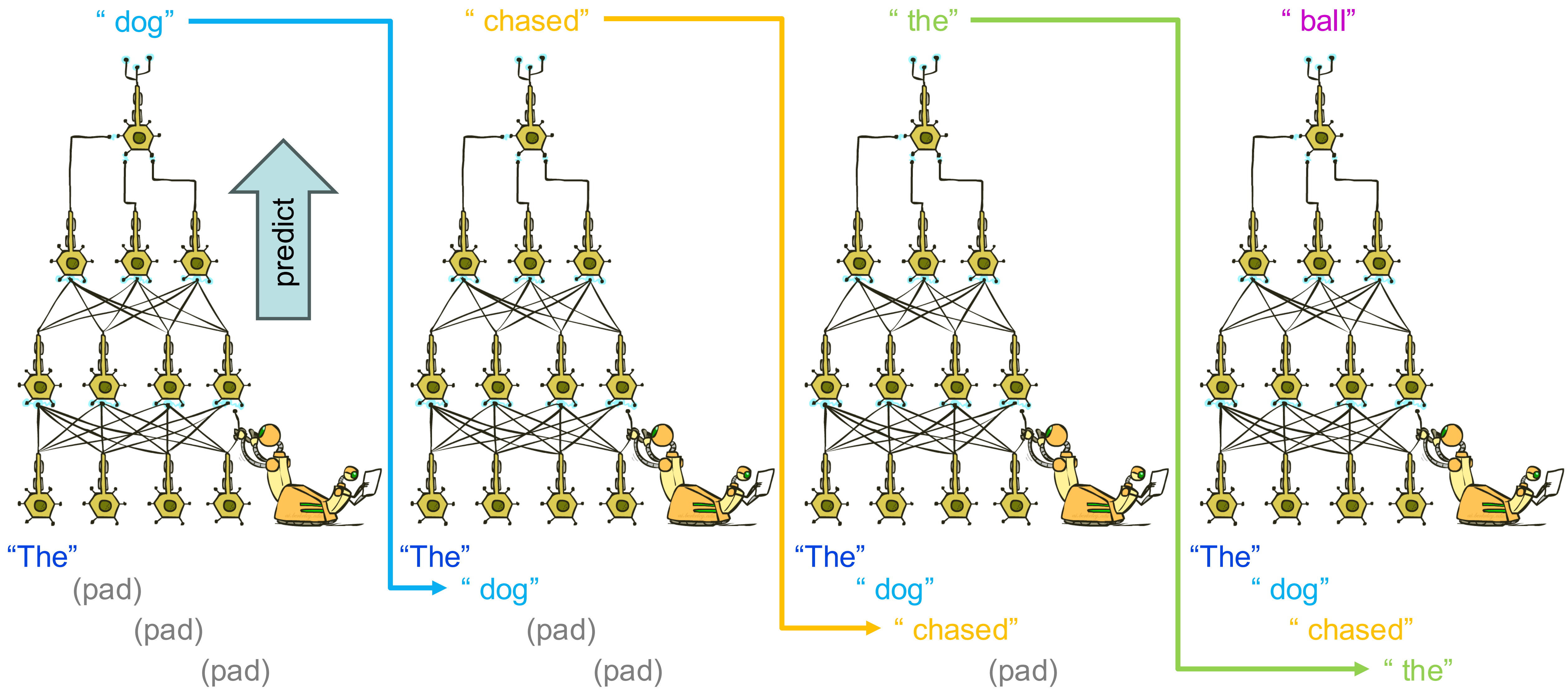
This is how it works

By **Visual Storytelling Team** and **Madhumita Murgia** in London SEPTEMBER 11 2023

Large Language Models

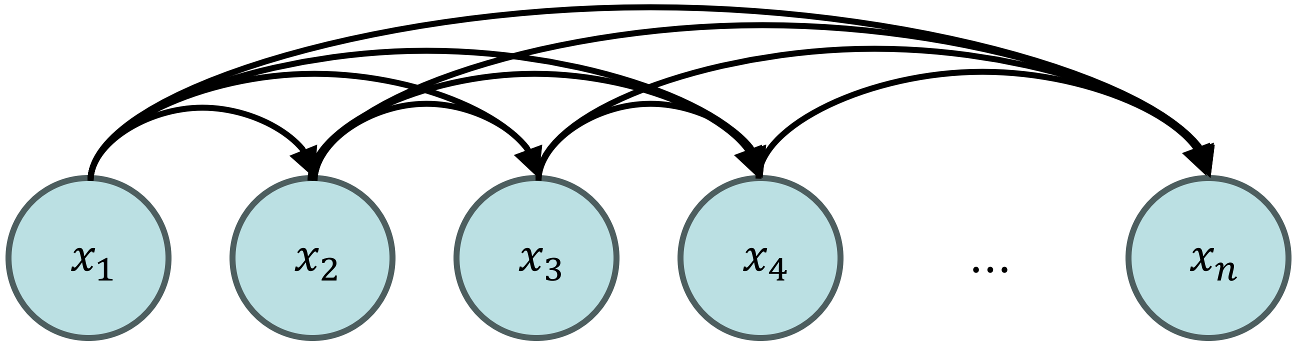
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- Deep neural networks
  - Autoregressive models
  - Self-attention mechanisms
  - Transformer architectures
- Multi-class classification
- Supervised learning
  - Self-supervised learning
  - Instruction tuning
- Reinforcement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

Autoregressive Models

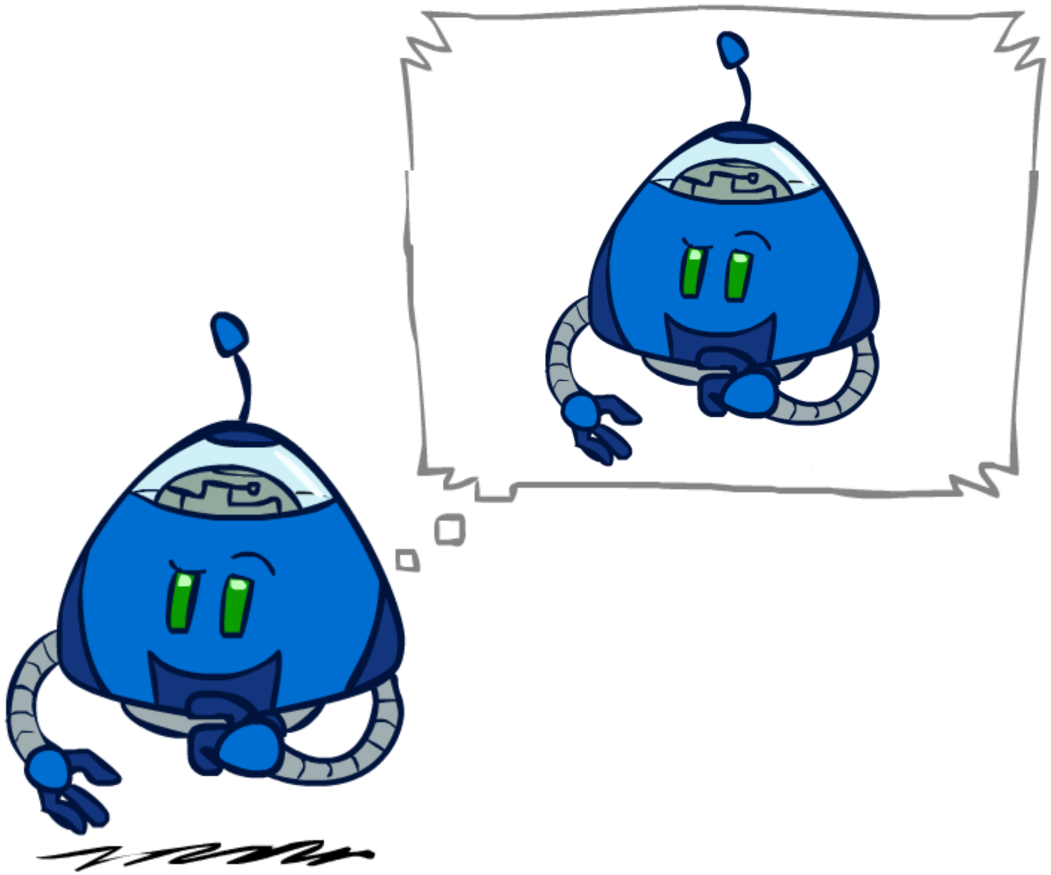


Autoregressive Models

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



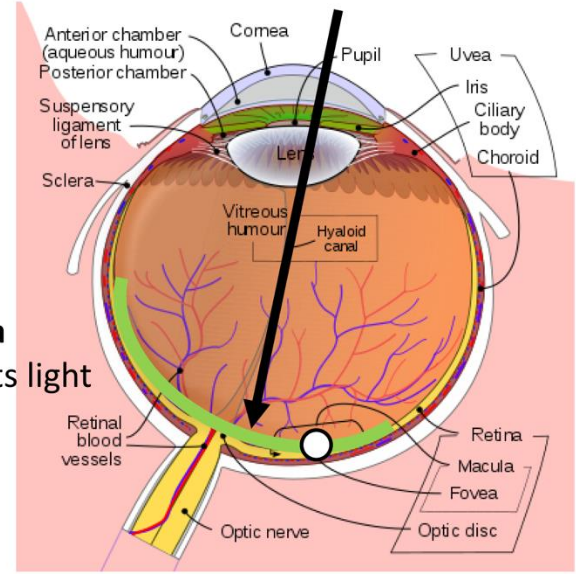
Self-Attention Mechanisms



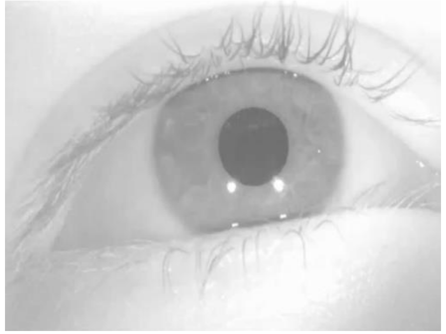
Attention Mechanism

- Human Vision: Fovea

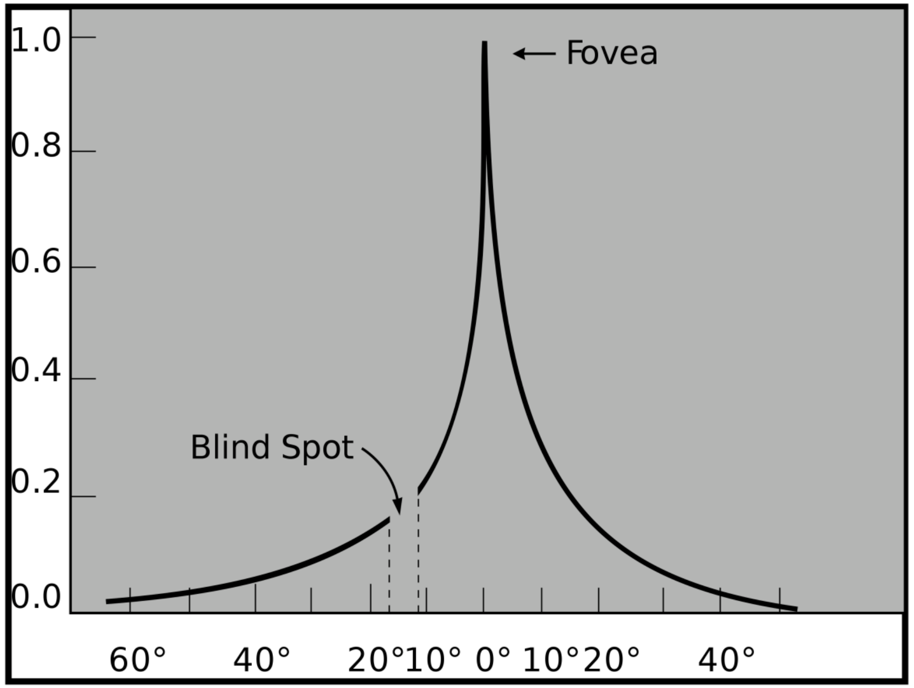
Light enters eye



Retina
detects light

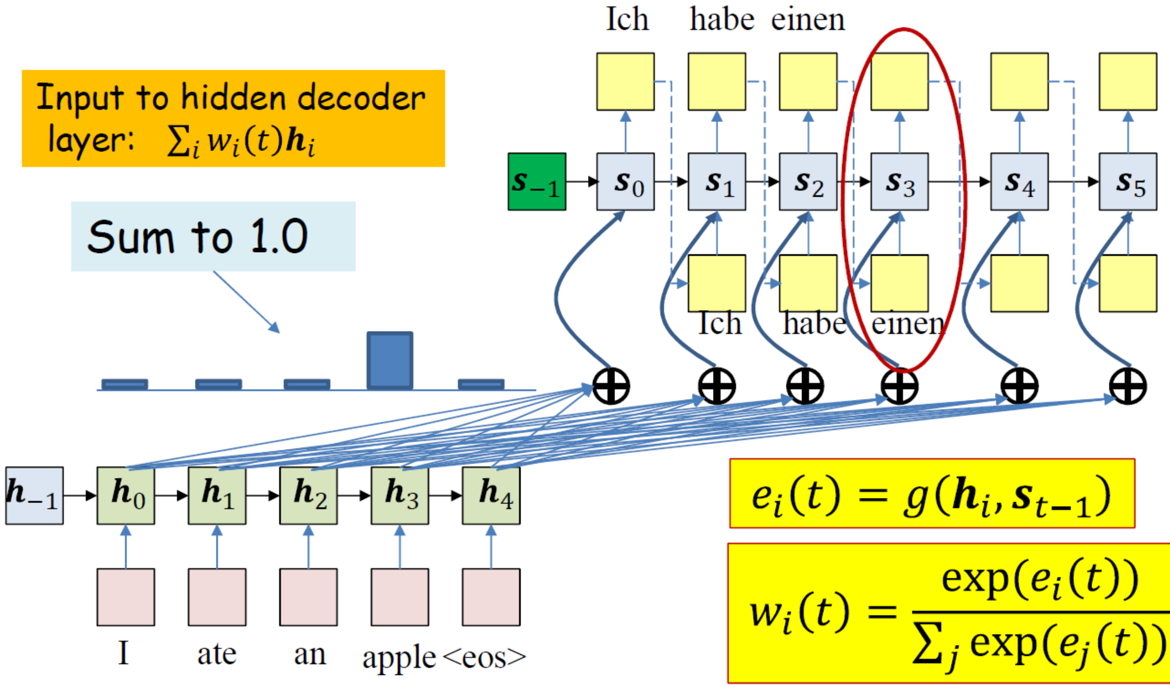


The **fovea** is a tiny region of the retina that can see with high acuity



Attention models

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



$$e_i(t) = g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1})$$

$$w_i(t) = \frac{\exp(e_i(t))}{\sum_j \exp(e_j(t))}$$

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

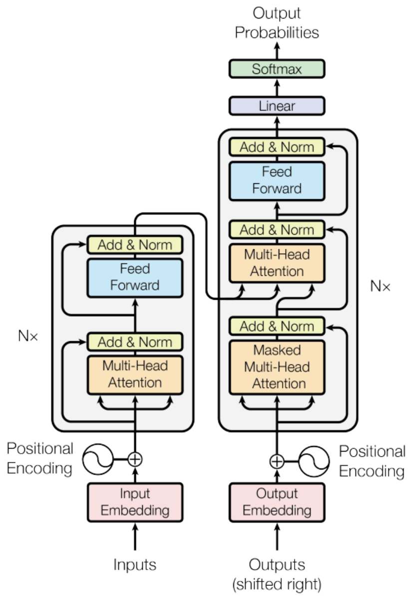


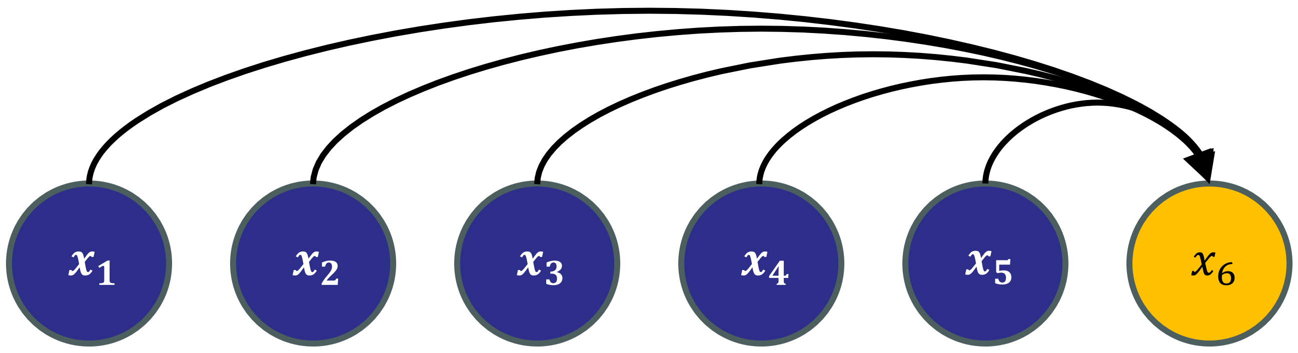
Figure 1: The Transformer - model architecture.

Attention is **all you need**

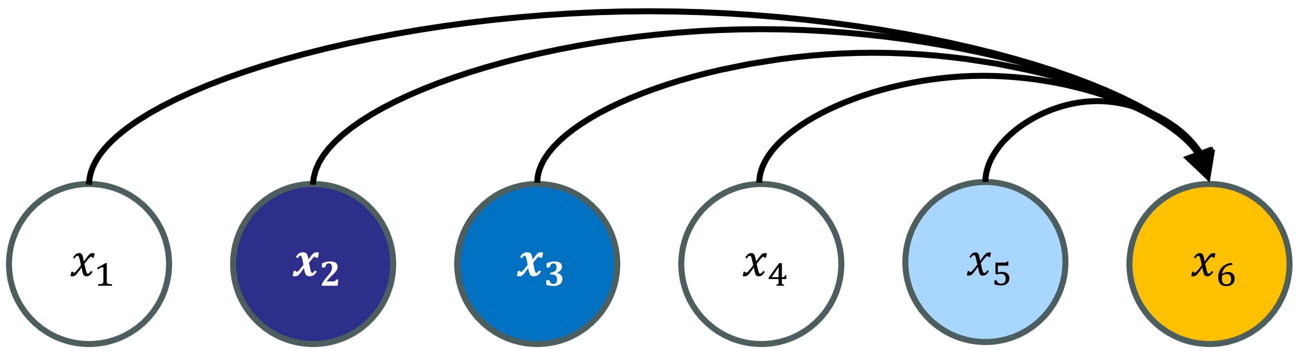
A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc

… 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 −∞) …

Self-Attention Mechanisms

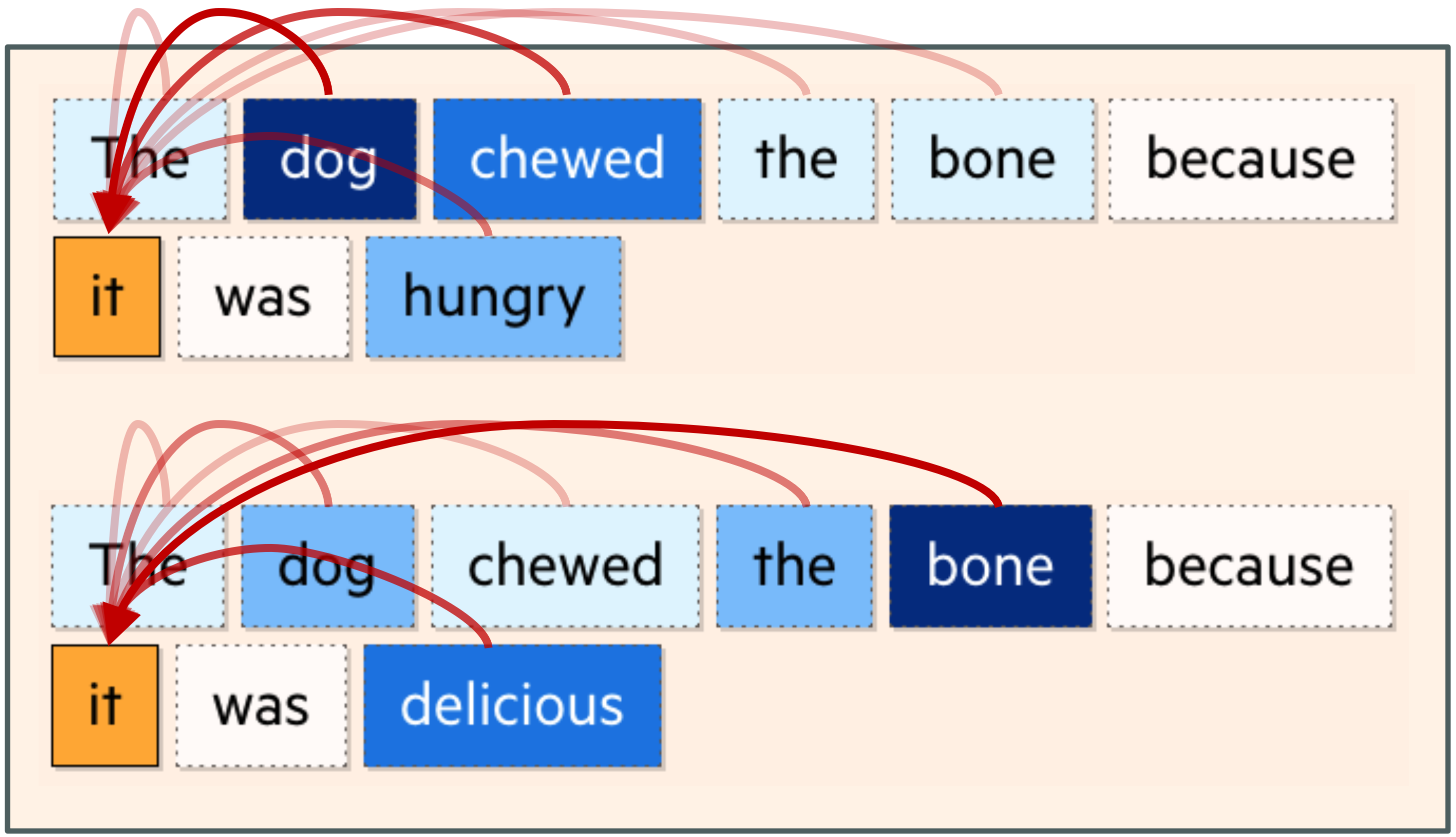


- Instead of conditioning on *all* input tokens equally…

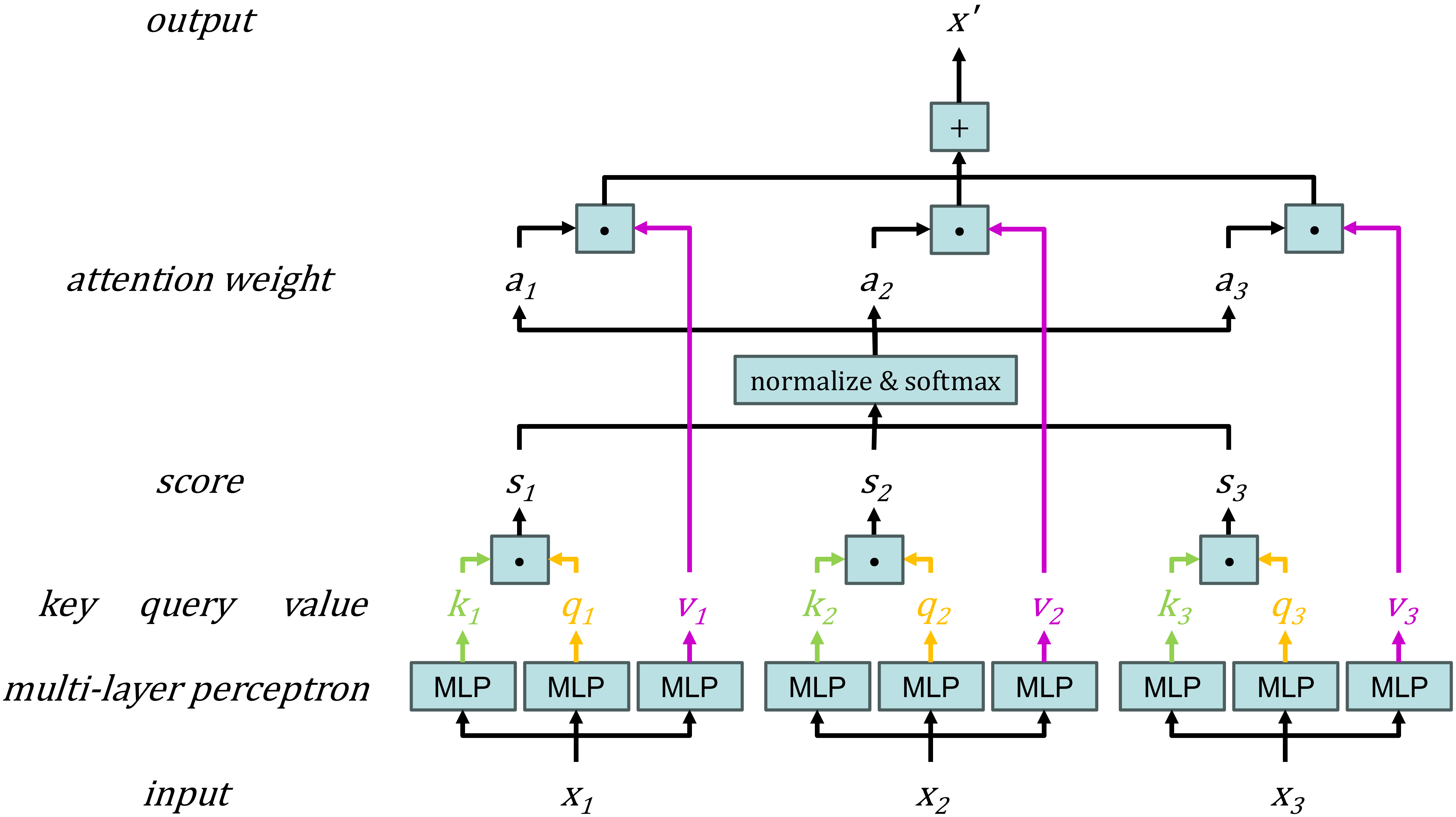


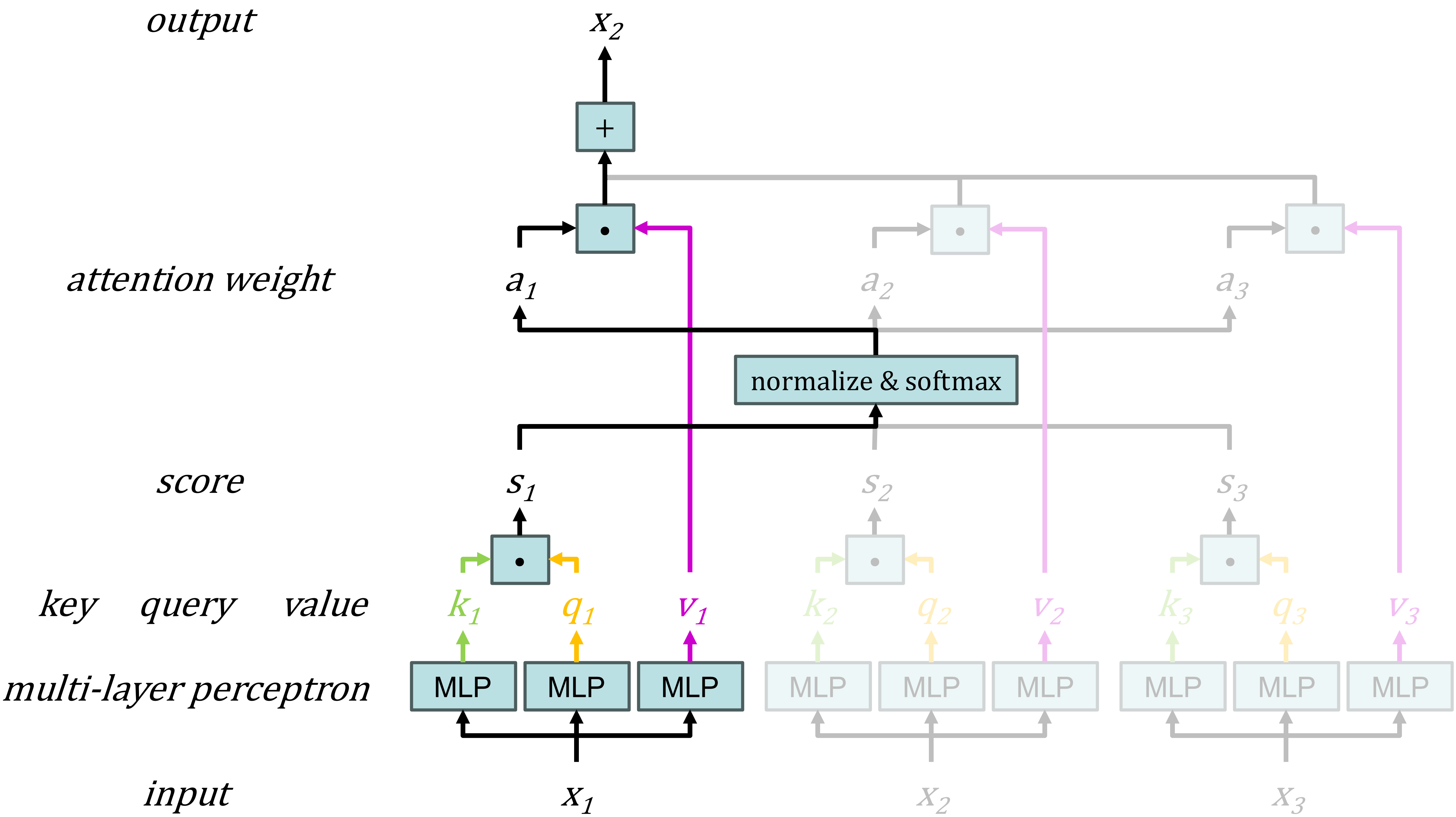
- Pay more attention to relevant tokens!

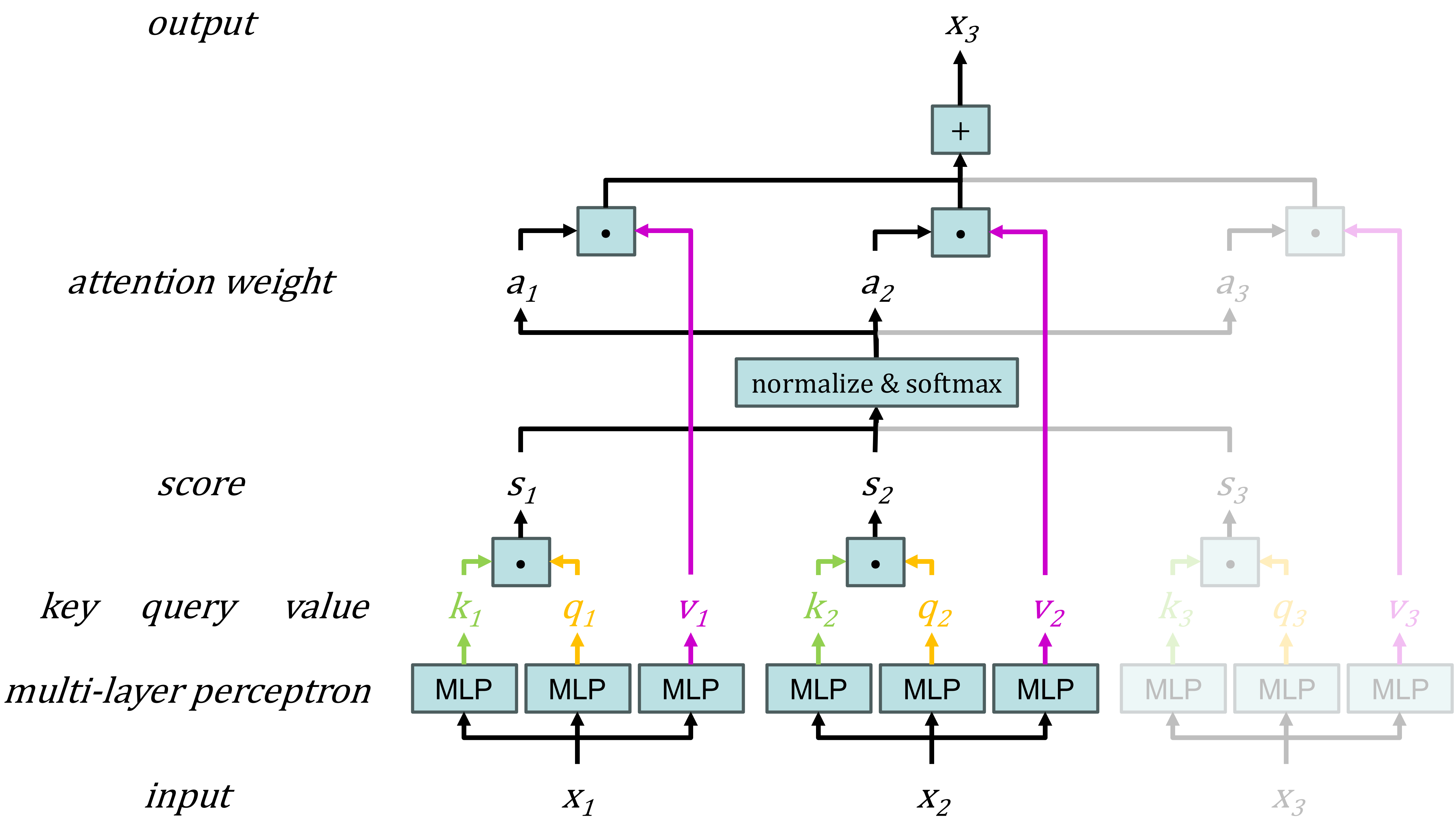
Self-Attention Mechanisms



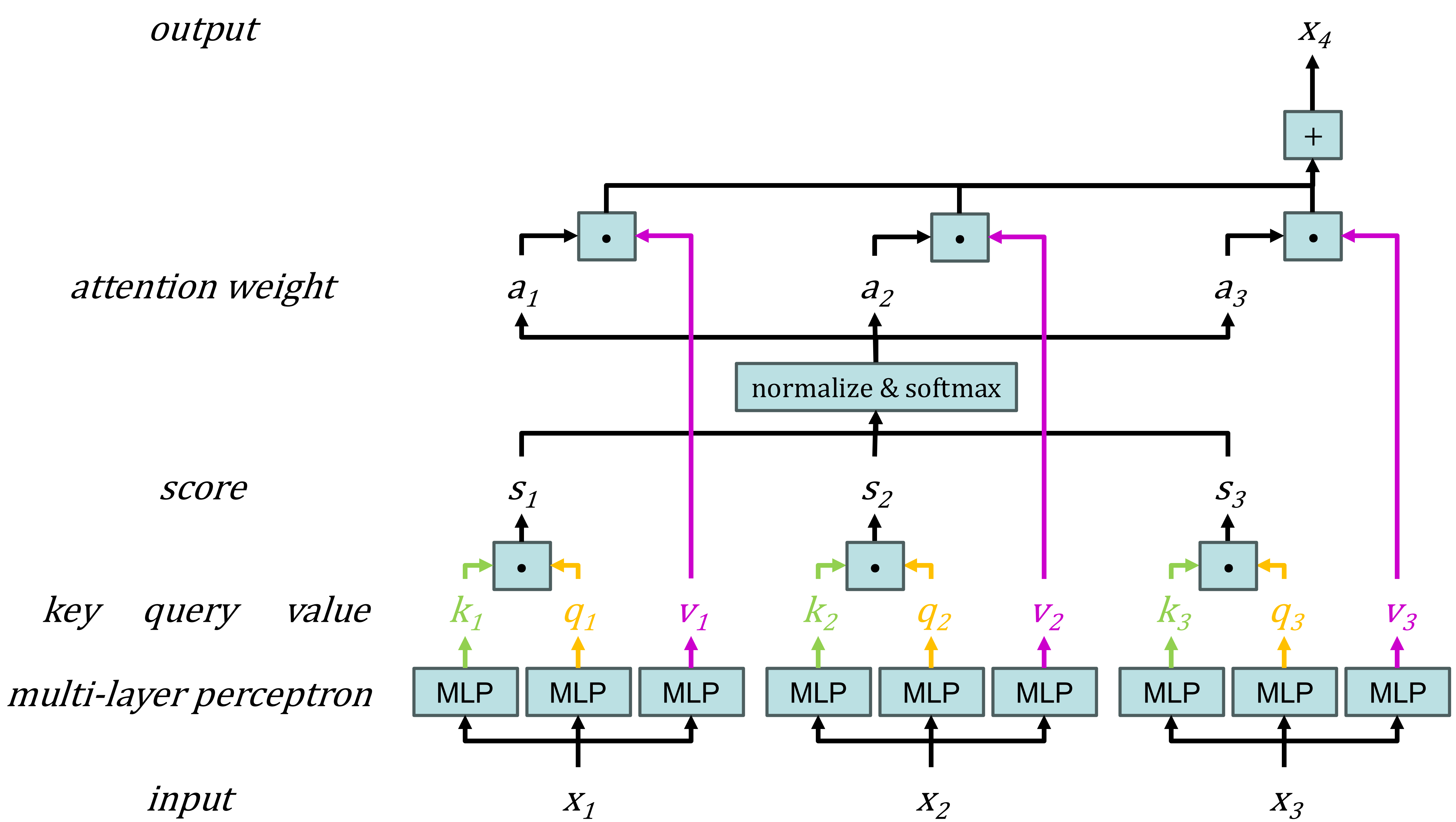
ig.ft.com/generative-ai


output
x'
+
attention weight
a1
a2
a3
normalize & softmax
score
s1
s2
s3
key
query
value
k1
q1
v1
k2
q2
v2
k3
q3
v3
multi-layer perceptron
MLP
input
x1
x2
x3

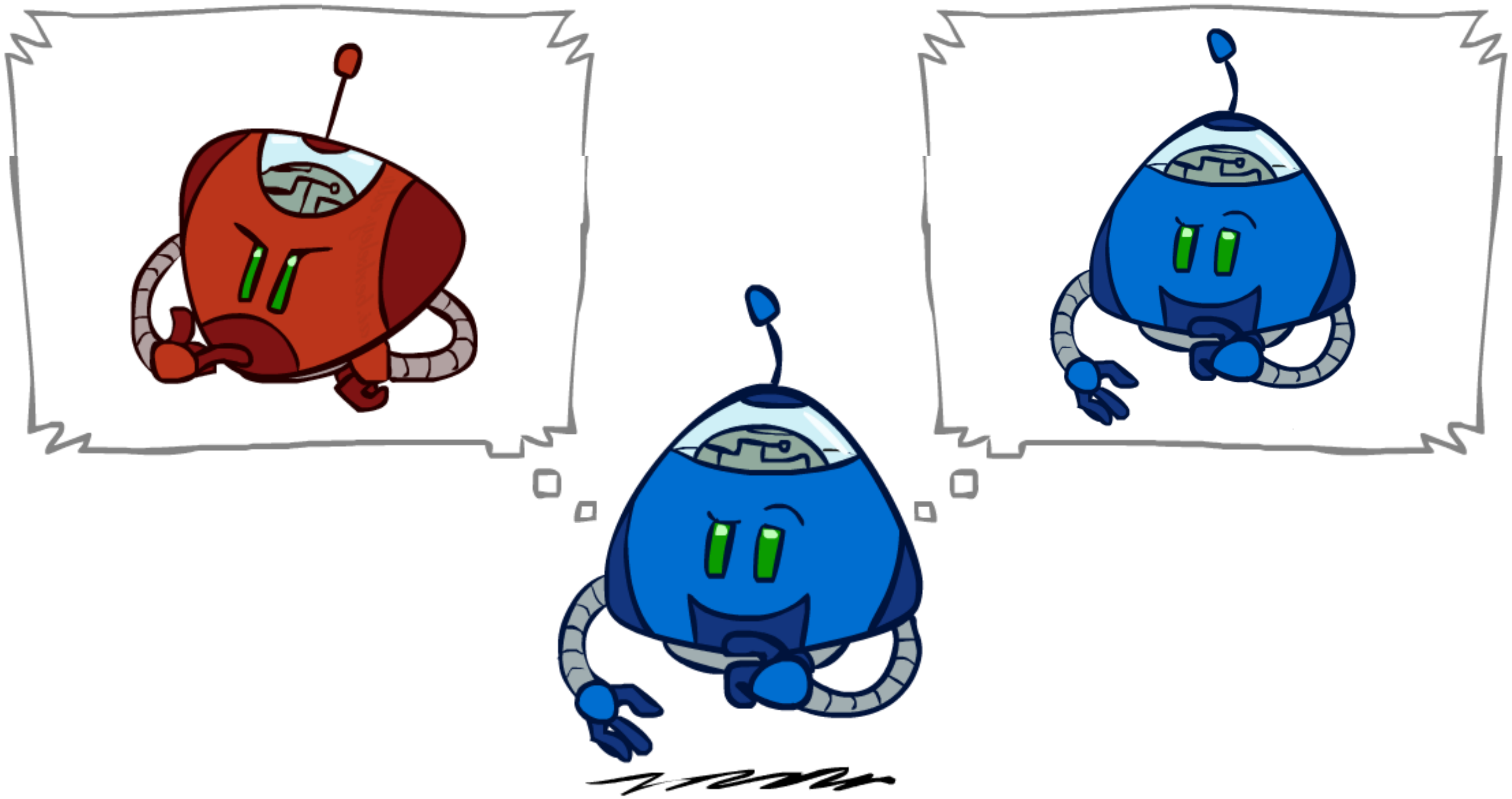

output
x_2
+
attention weight
a_1
a_2
a_3
normalize & softmax
score
s_1
s_2
s_3
key
query
value
k_1
q_1
v_1
k_2
q_2
v_2
k_3
q_3
v_3
multi-layer perceptron
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
input
x_1
x_2
x_3



output
x3
+
attention weight
a1
a2
a3
normalize & softmax
score
s1
s2
s3
key
query
value
k1
q1
v1
k2
q2
v2
k3
q3
v3
multi-layer perceptron
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
input
x1
x2
x3

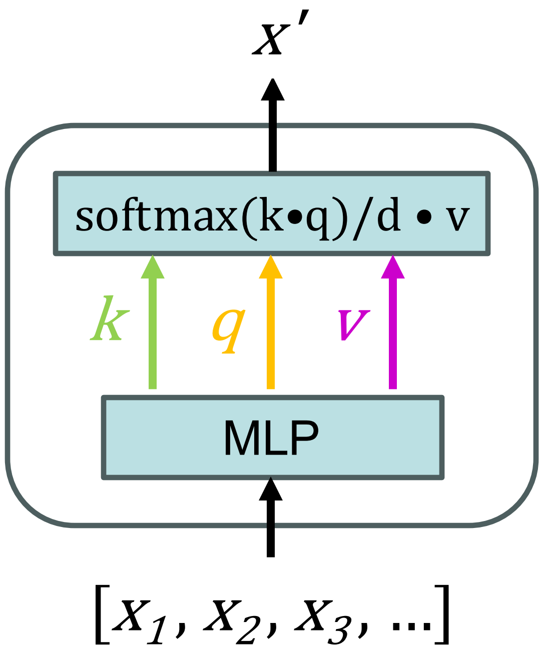

output
x_4
+
attention weight
a_1
a_2
a_3
normalize & softmax
score
s_1
s_2
s_3
key
query
value
k_1
q_1
v_1
k_2
q_2
v_2
k_3
q_3
v_3
multi-layer perceptron
MLP
input
x_1
x_2
x_3

Multi-Headed Attention

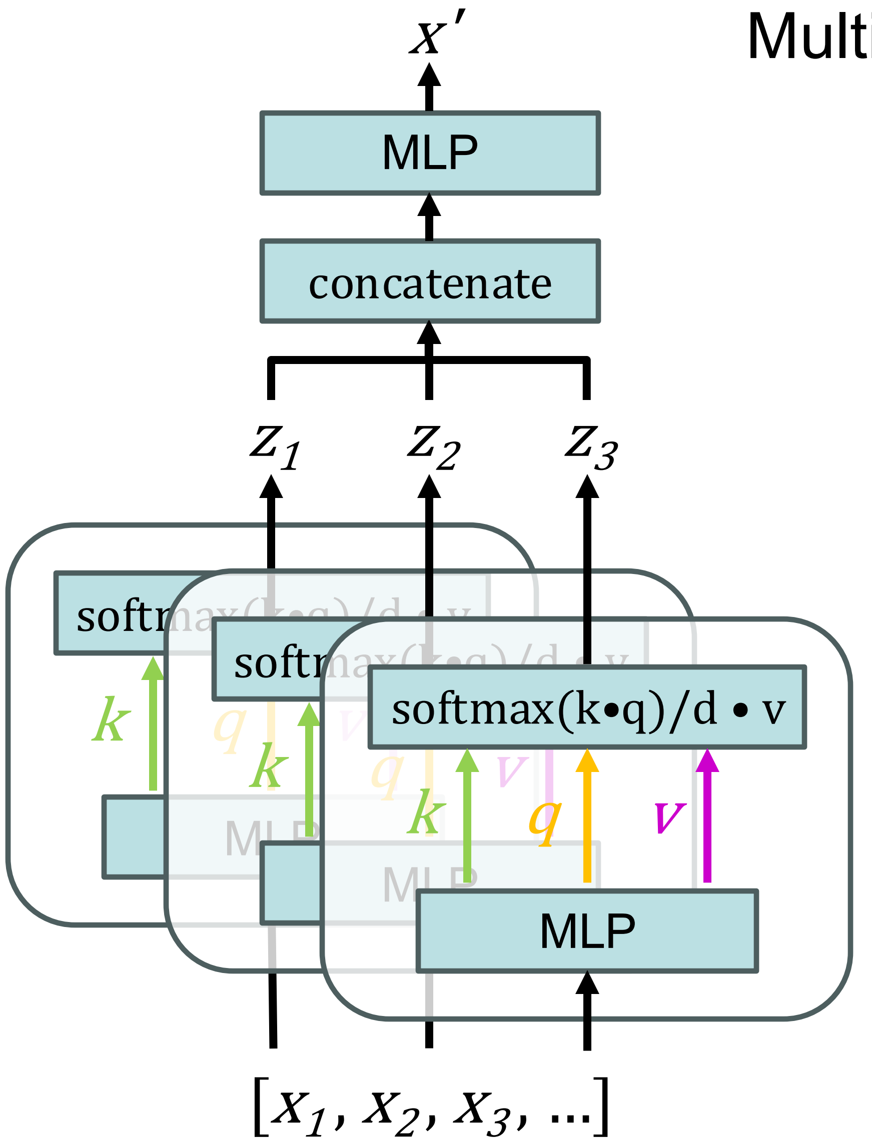


Multi-Headed Attention

Single-headed

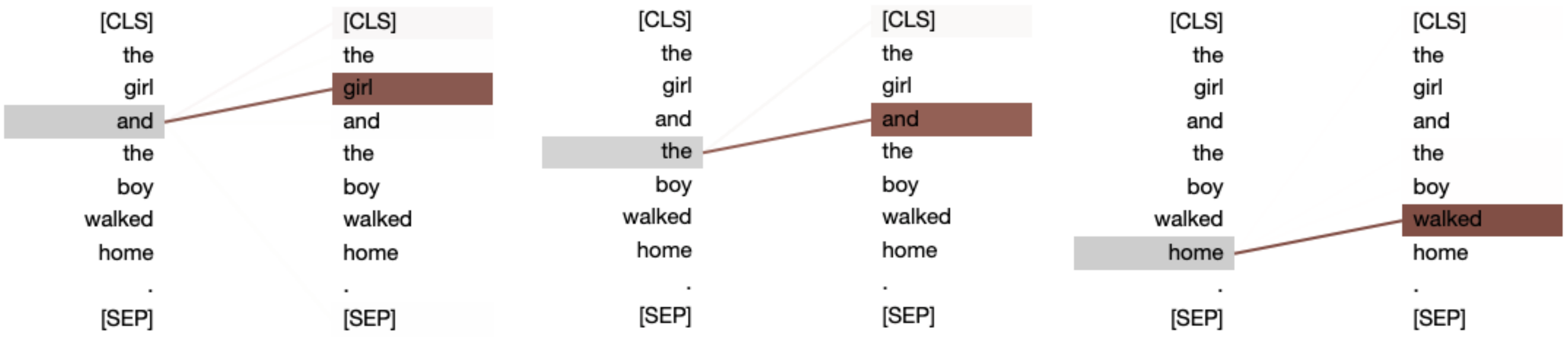


Multi-headed



Multi-Headed Attention

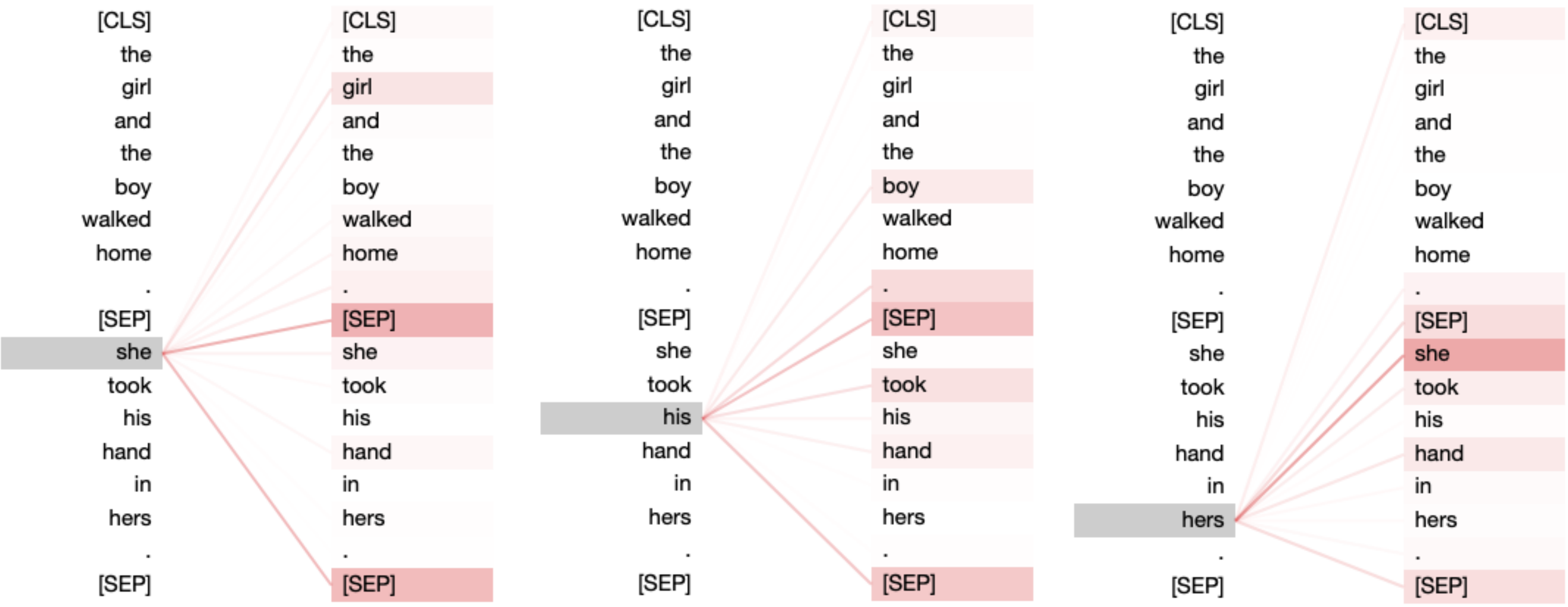
Head 6: previous word



https://github.com/jessevig/bertviz

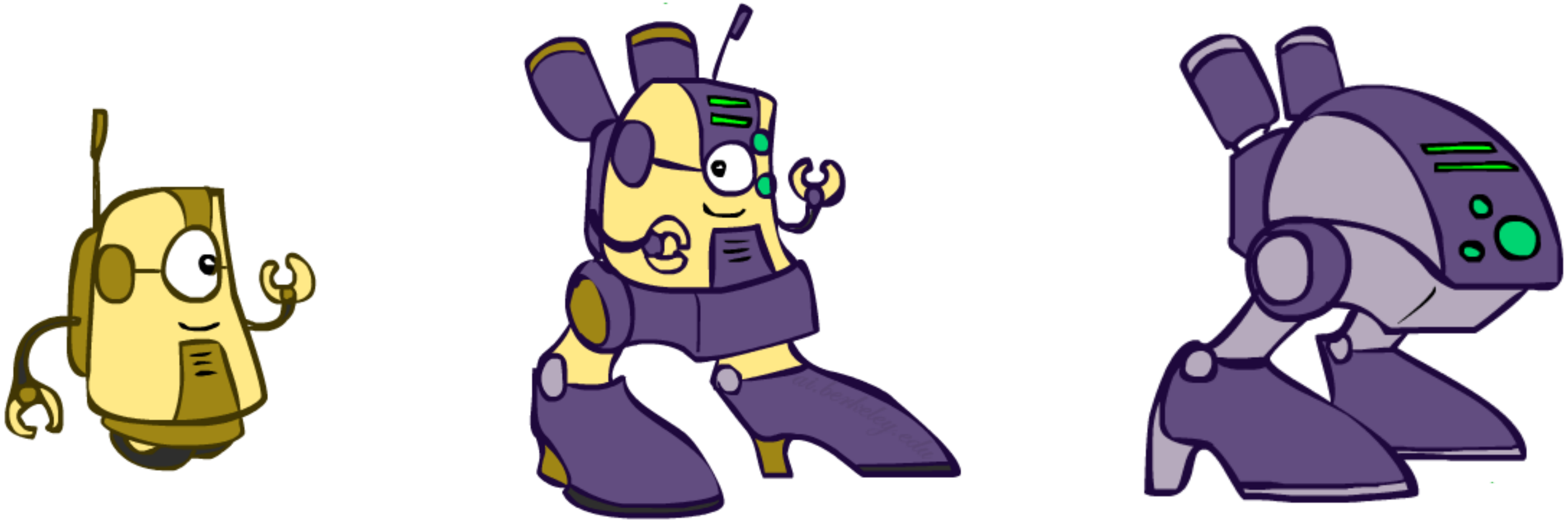
Multi-Headed Attention

Head 4: pronoun references

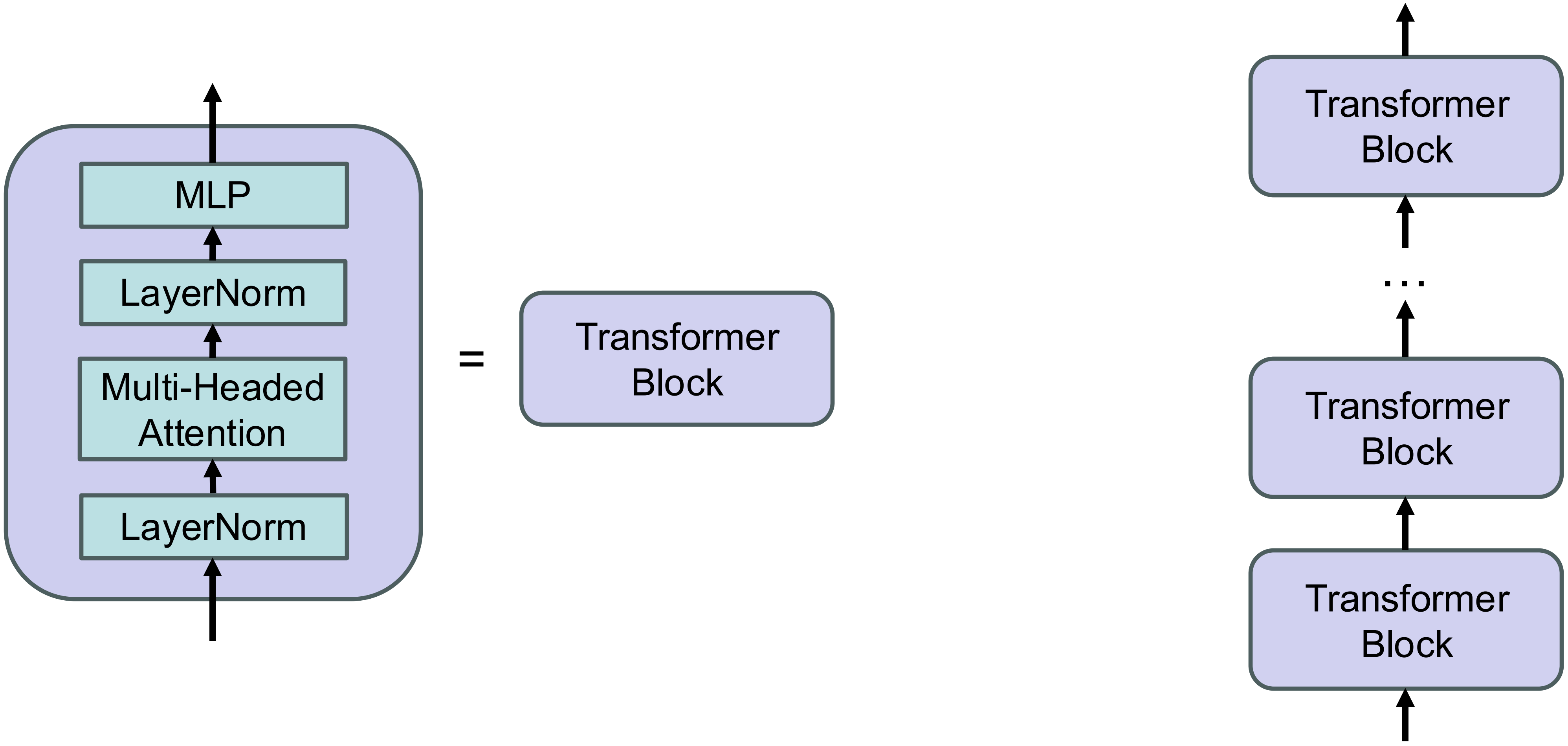


https://github.com/jessevig/bertviz

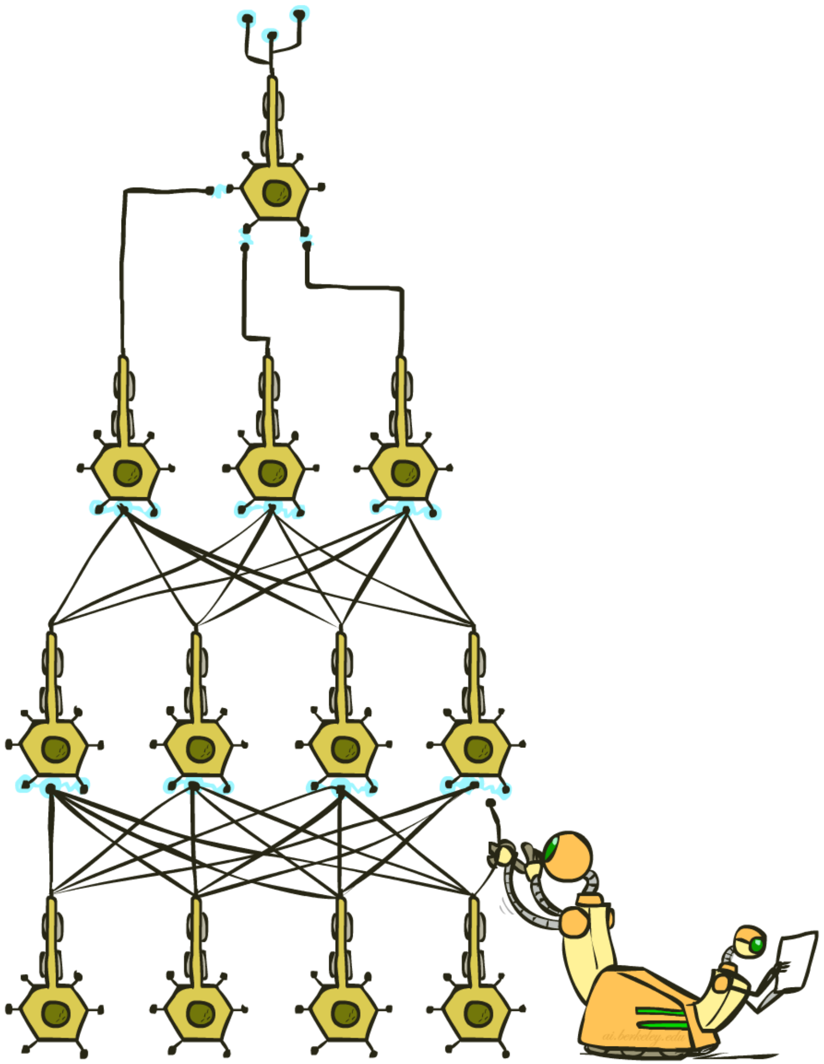
Transformer Architecture

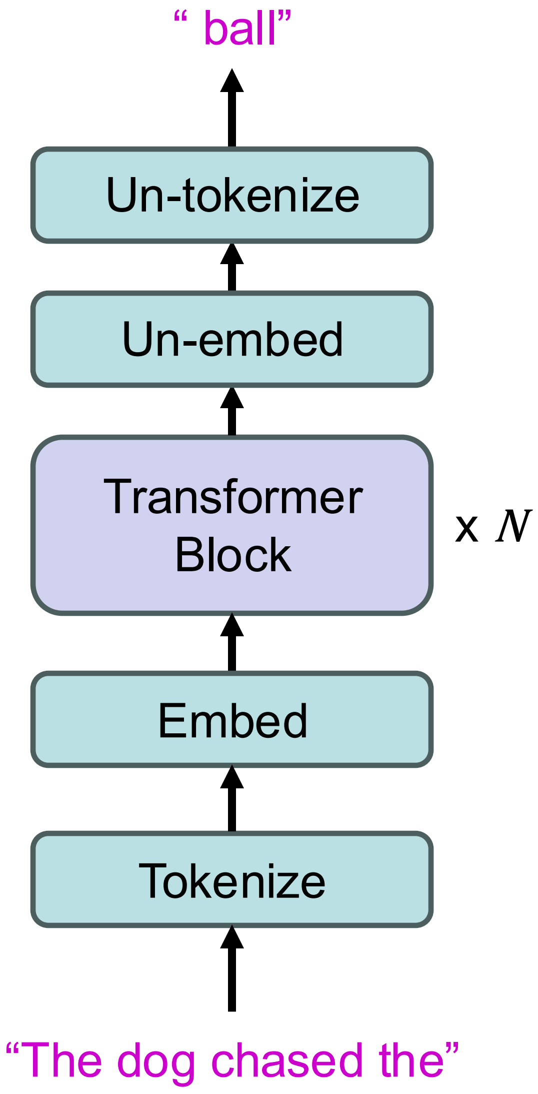


Transformer Architecture



Transformer 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

Unsupervised / Self-Supervised Learning

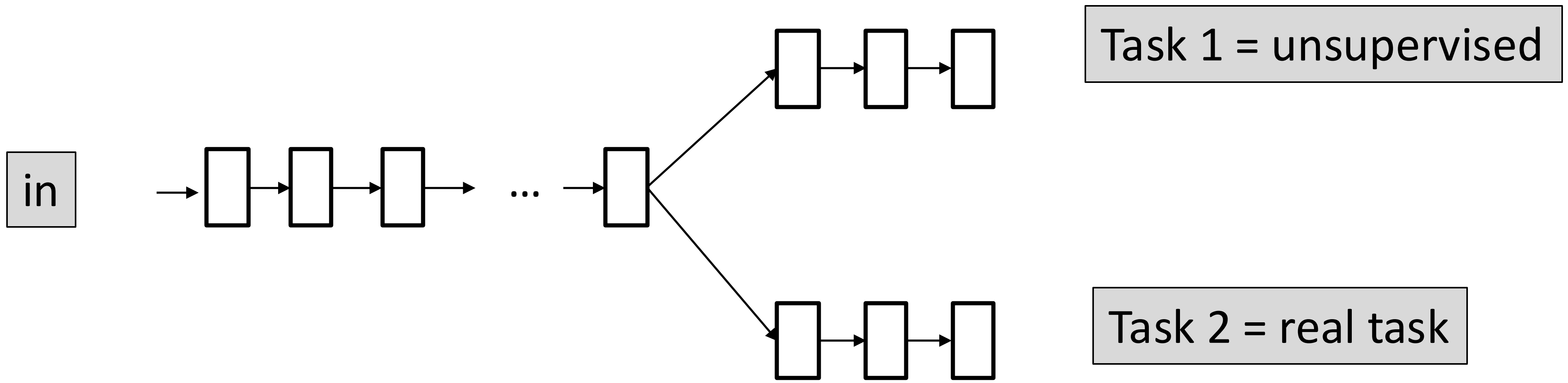
- Do we always need human supervision to learn features?
- Can’t we learn general-purpose features?
- Key hypothesis:

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

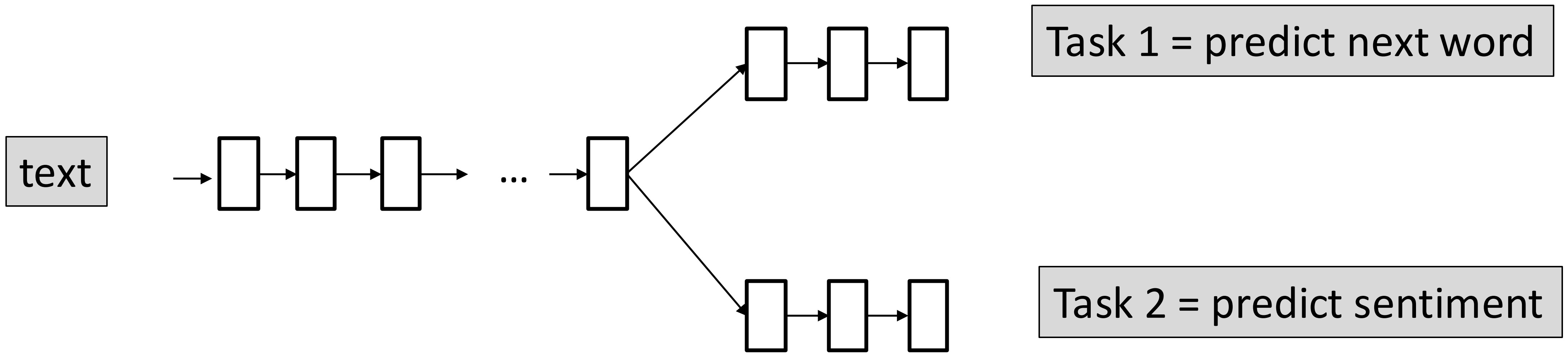


Image Pre-Training: Predict Missing Patch

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

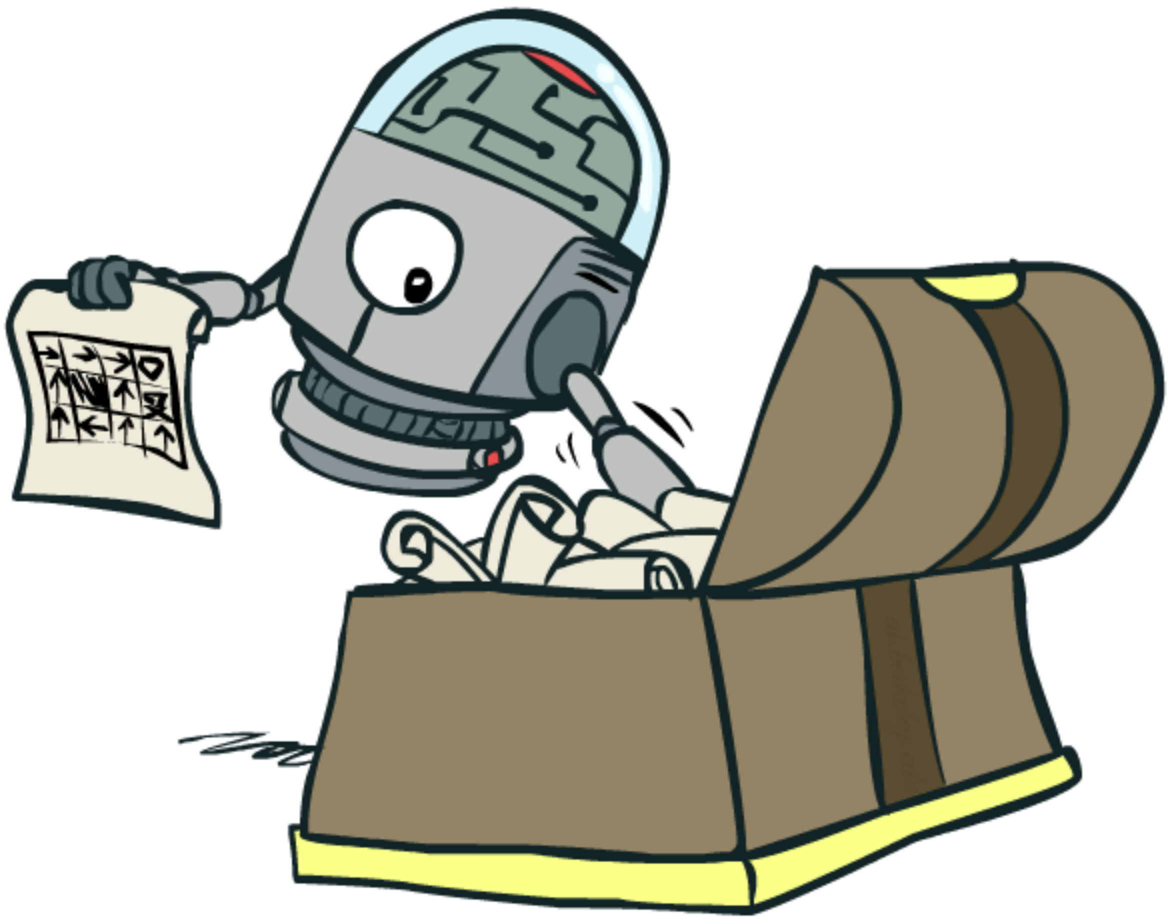
Reinforcement Learning from Human Feedback

- 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

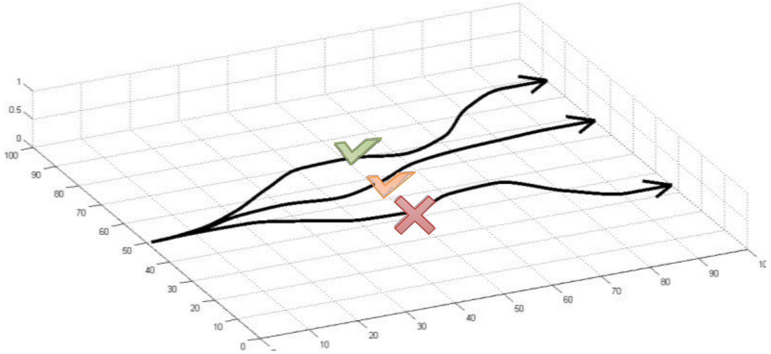
2. Estimate policy performance: $J(\theta) = V^{\pi_\theta}(s_0)$

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

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



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 πθ.

sum over samples from π_θ

Direct policy differentiation

optimal policy parameters:

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

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

a convenient identity

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

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t)}] = \int p_\theta(\tau) r(\tau) d\tau$$

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

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

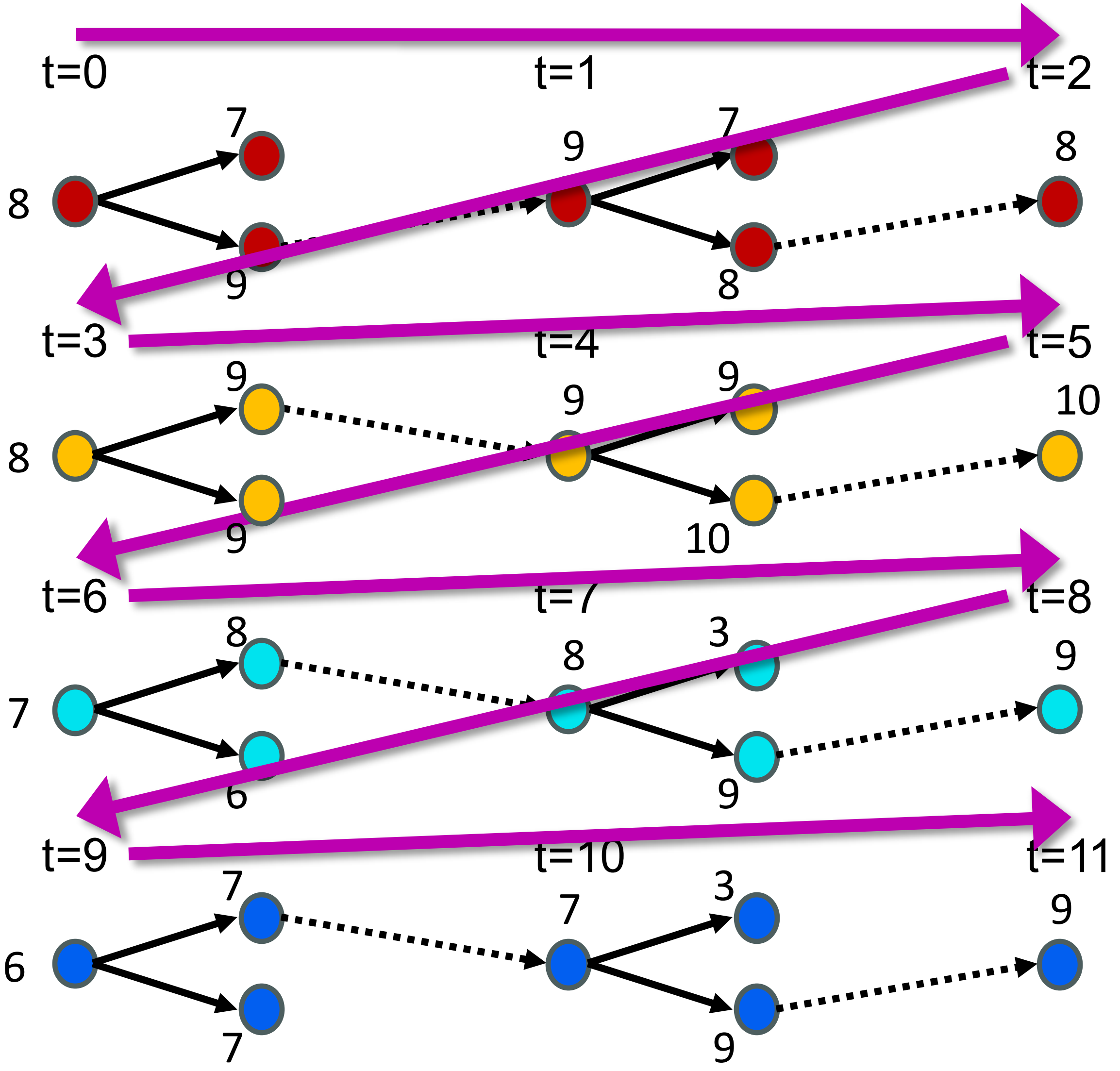
Estimating the Policy Gradient

- 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

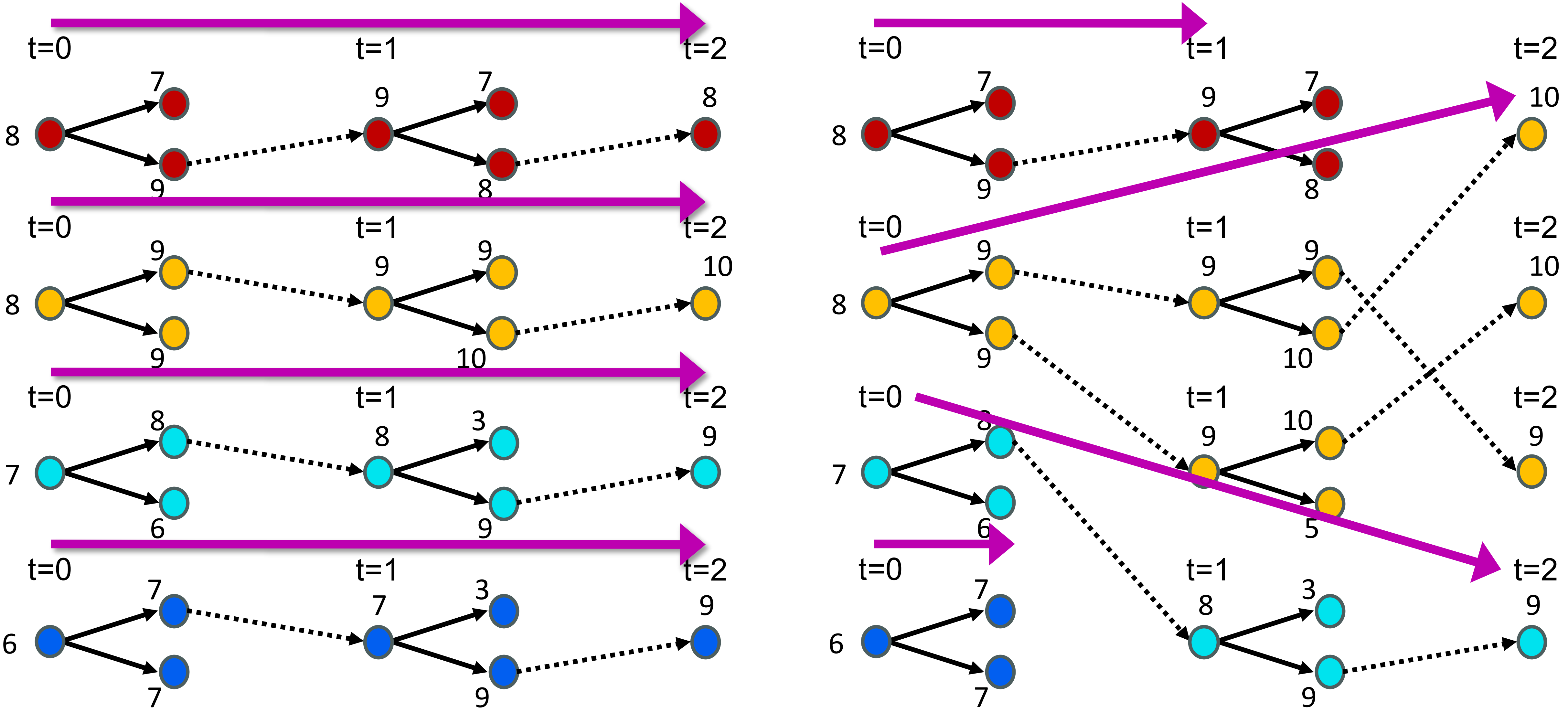
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- ~~Deep neural networks~~
  - ~~Autoregressive models~~
  - ~~Self-attention mechanisms~~
  - ~~Transformer architectures~~
- ~~Multi-class classification~~
- ~~Supervised learning~~
  - ~~Self-supervised learning~~
  - ~~Instruction 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

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~~

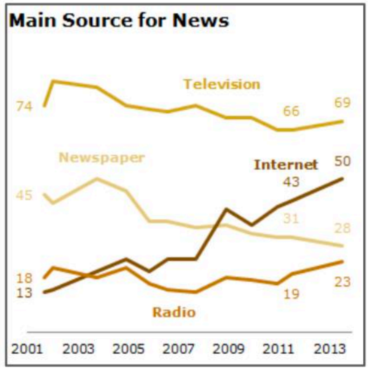
Language models build a structured concept space

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

Can we build a single model of all data types?

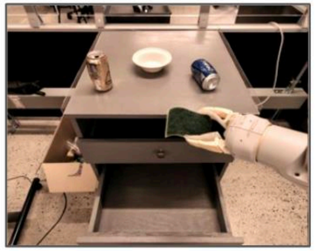
If  was invented by Wright brothers. Who invented  ?

example from [Tsimpoukelli et al, 2021]

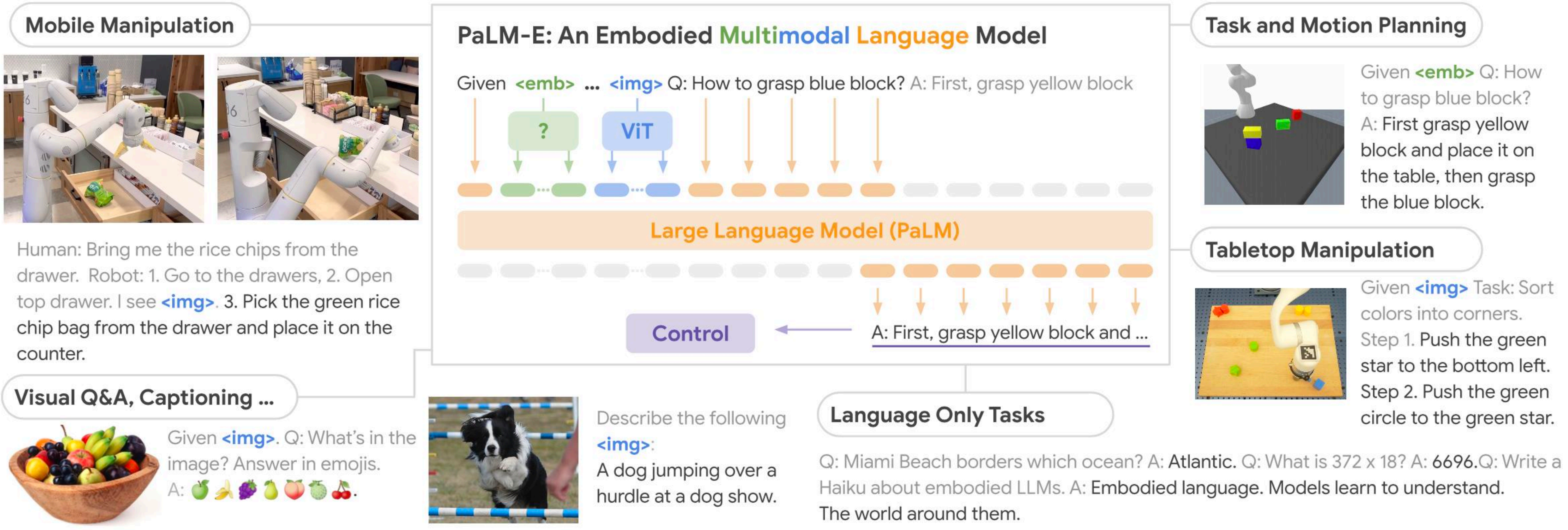
What is the fastest-growing news source according to 

?

If  changes into  what does  change into?

What action should I take from  to accomplish “  ”?

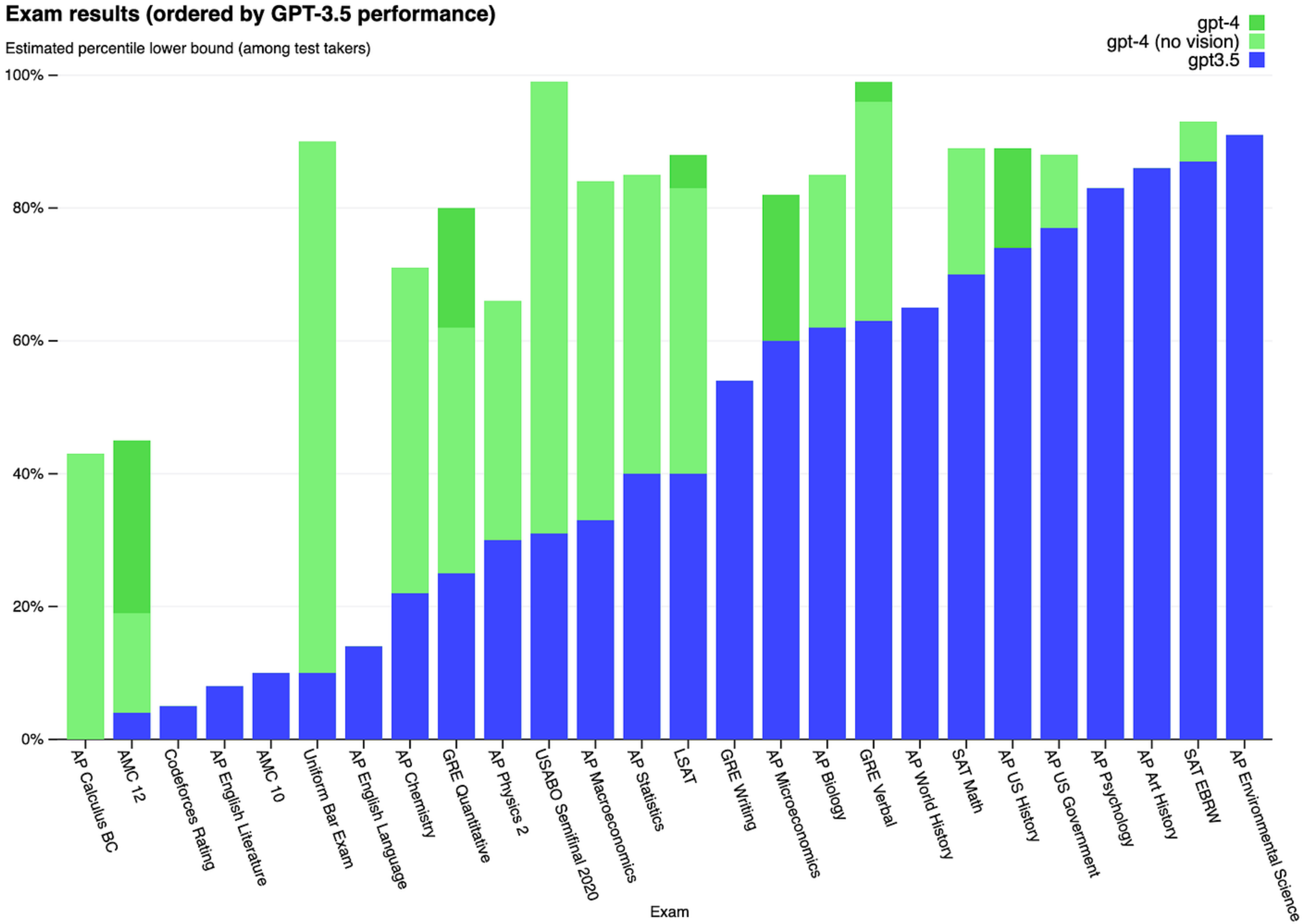
Can we build a single model of all data types?



[PaLM-E, Driess et al, 2023]

Tracking Progress

- How well AI can do human tasks

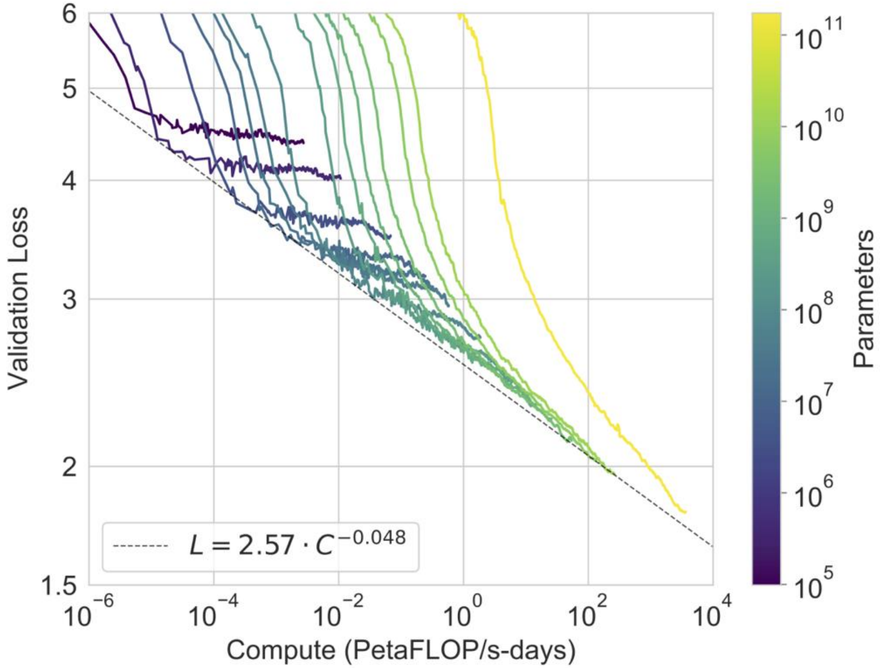


[OpenAI]

Forecasting Progress

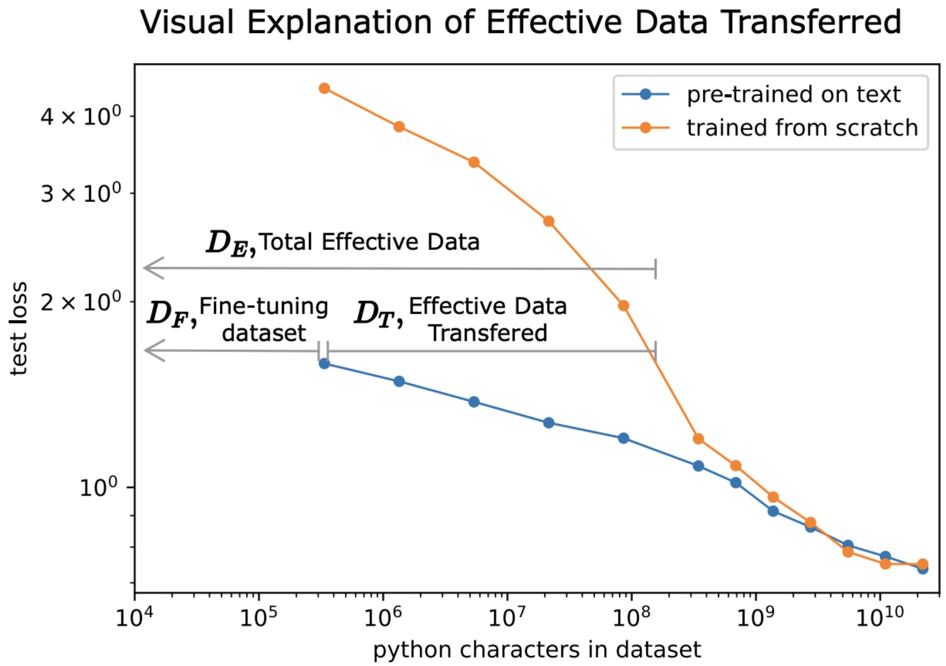
- Scaling Laws extrapolate:
  - If we [make model bigger / add more data / …]
  - What would accuracy become?

compute:



[Brown et al, 2020]

data:

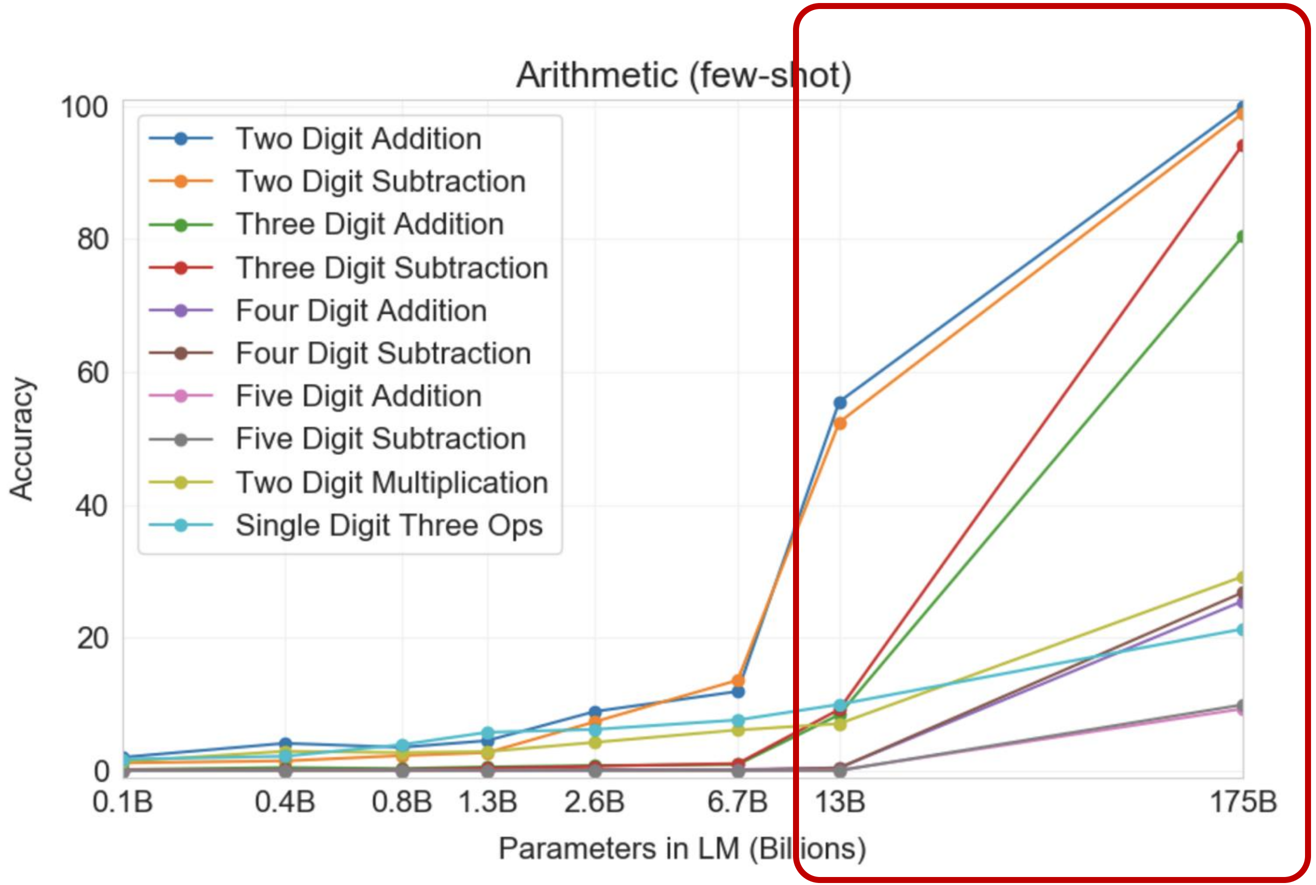


[Hernandez et al, 2021]

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]