CS 188: Artificial Intelligence

Large Language Models and Transformers

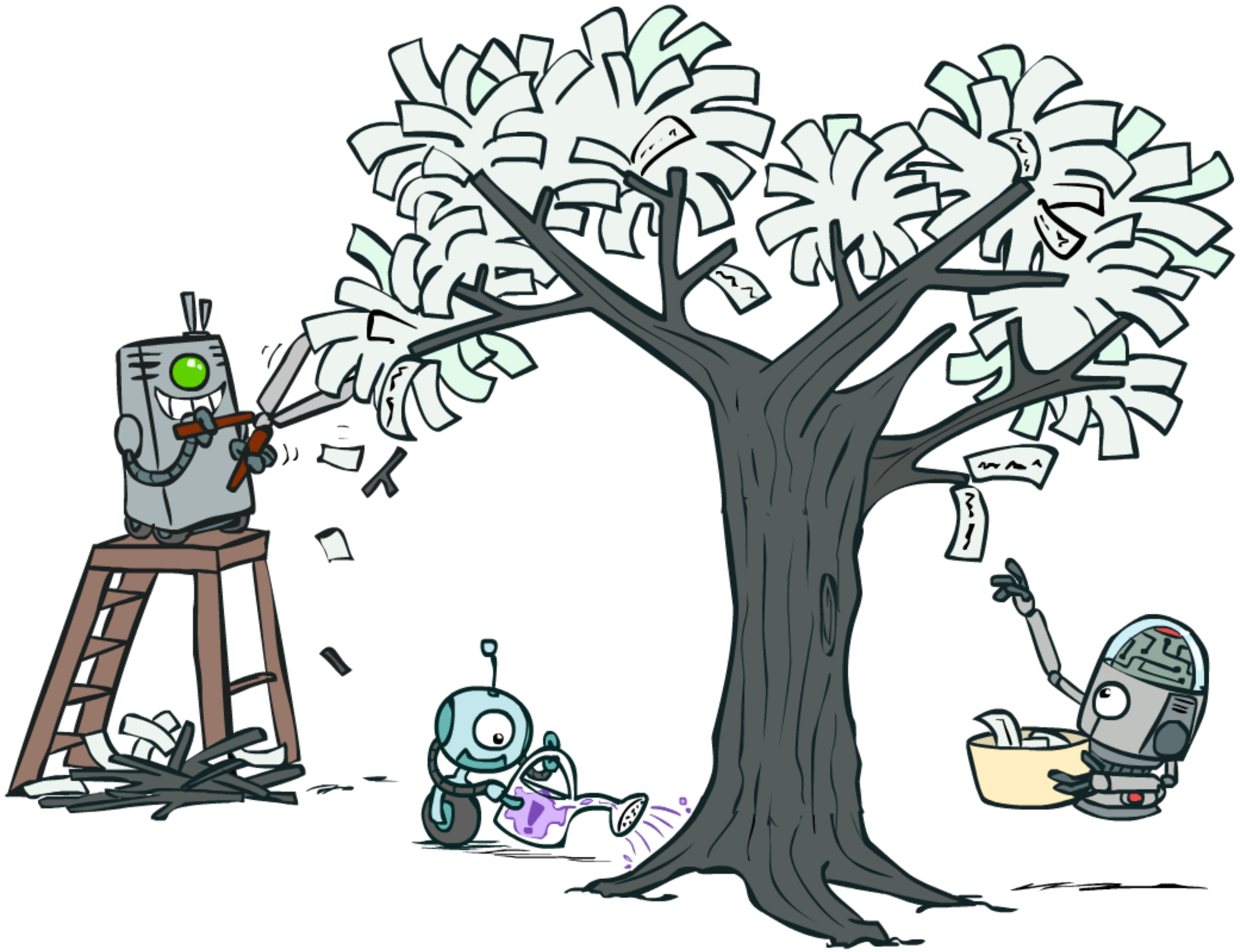
Instructors: John Canny and Oliver Grillmeyer --- University of California, Berkeley

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Announcements

- Project 5 (last project)
  - Due Friday 4/25 at 11:59pm
- HW10 (last homework)
  - Due Wednesday 4/23 at 11:59pm
- Final Exam (last exam)
  - Thursday 5/15 from 3:00-6:00pm
  - See Exam Logistics on CS 188 website

Decision Trees: wrap-up

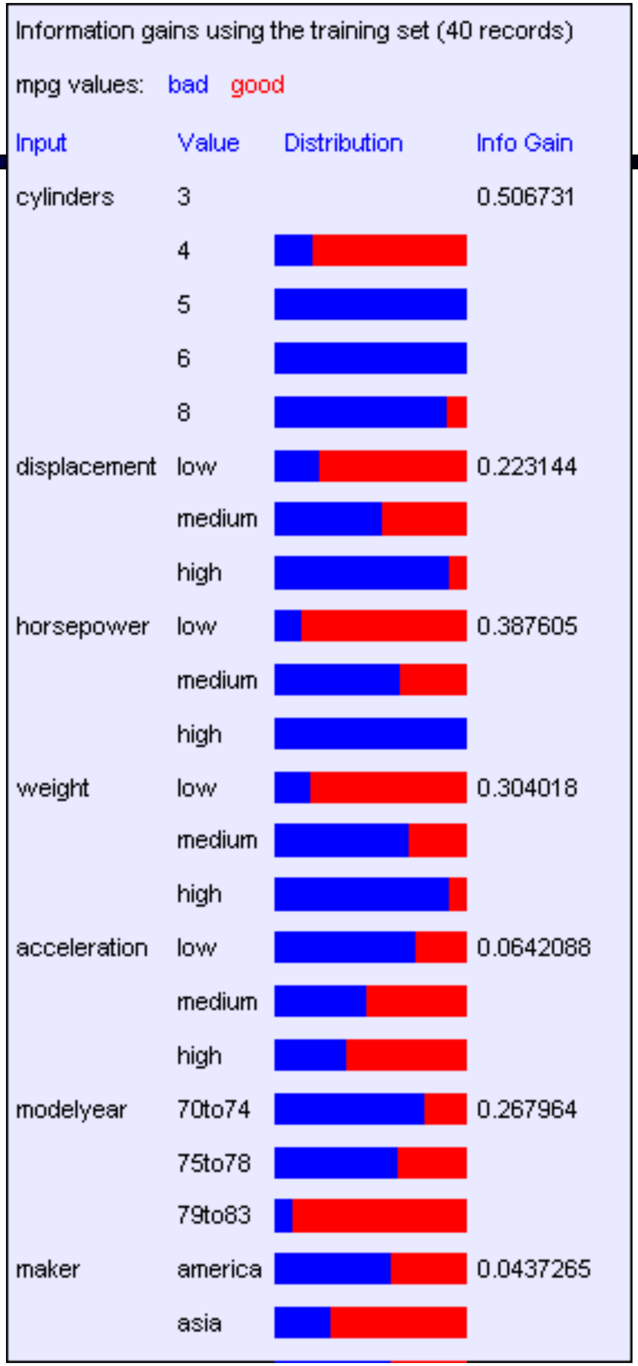


Example: Miles Per Gallon

40 Examples

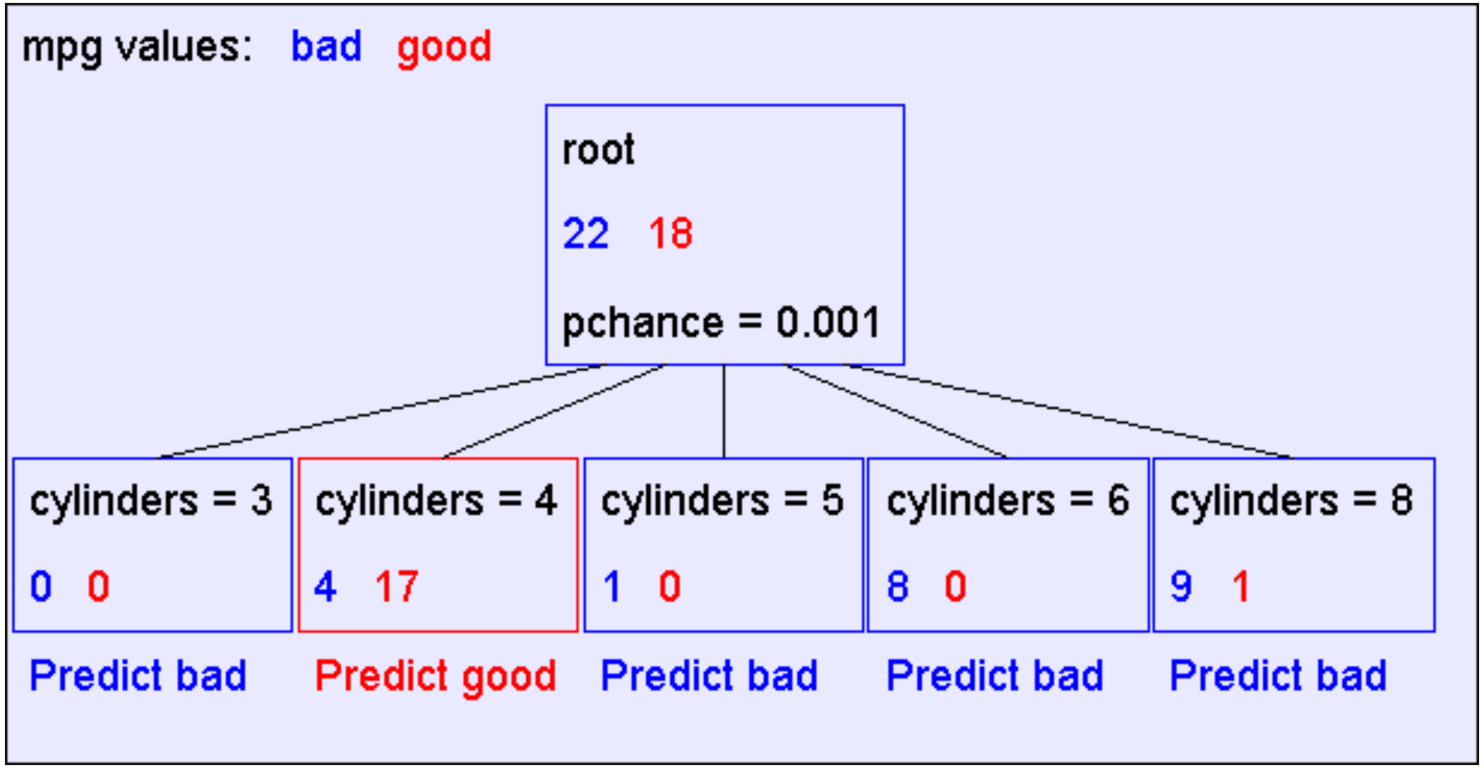
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

Find the First Split

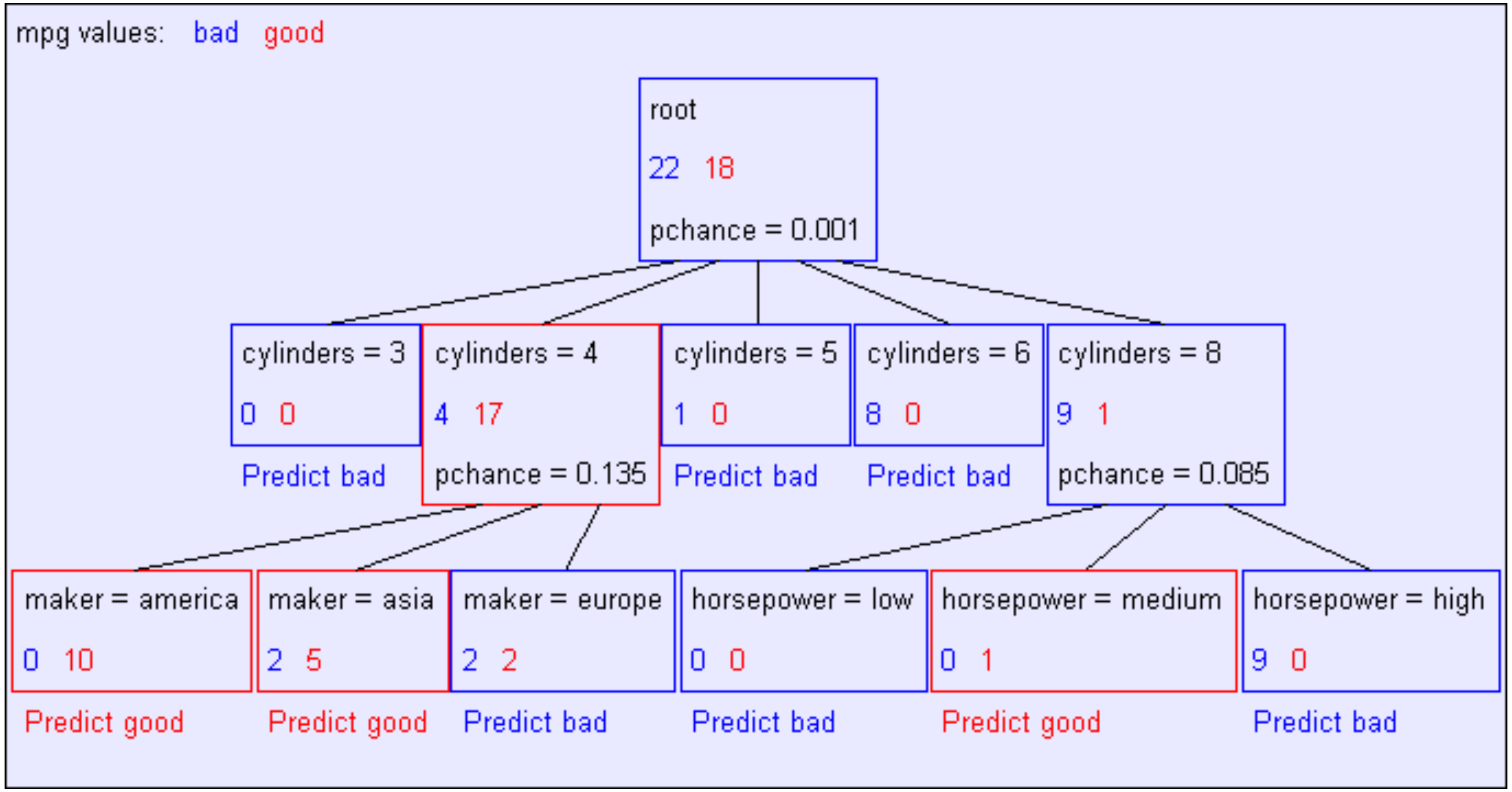


- Look at information gain for each attribute
- Note that each attribute is correlated with the target!
- What do we split on?

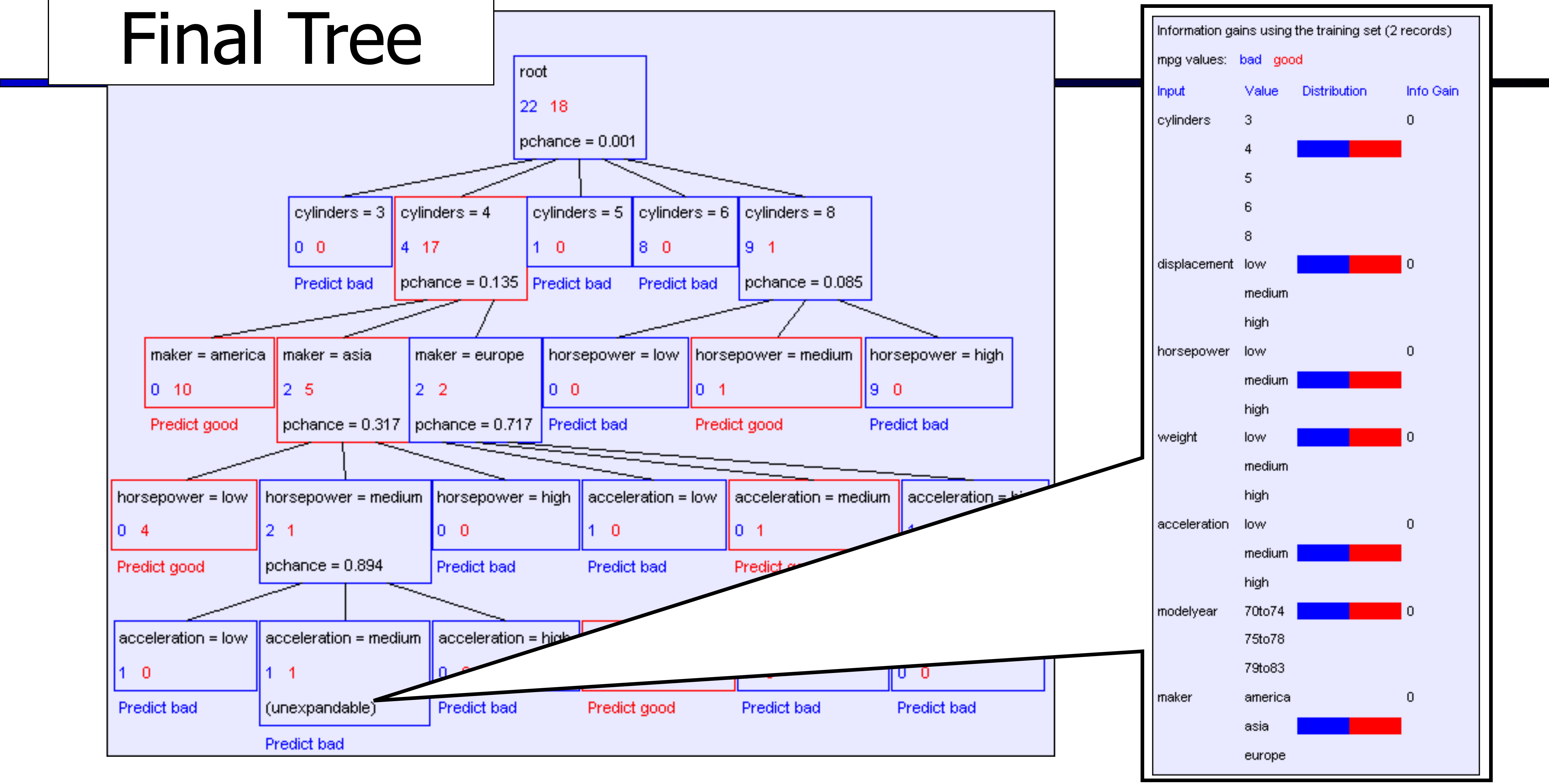
Result: Decision Stump



Second Level

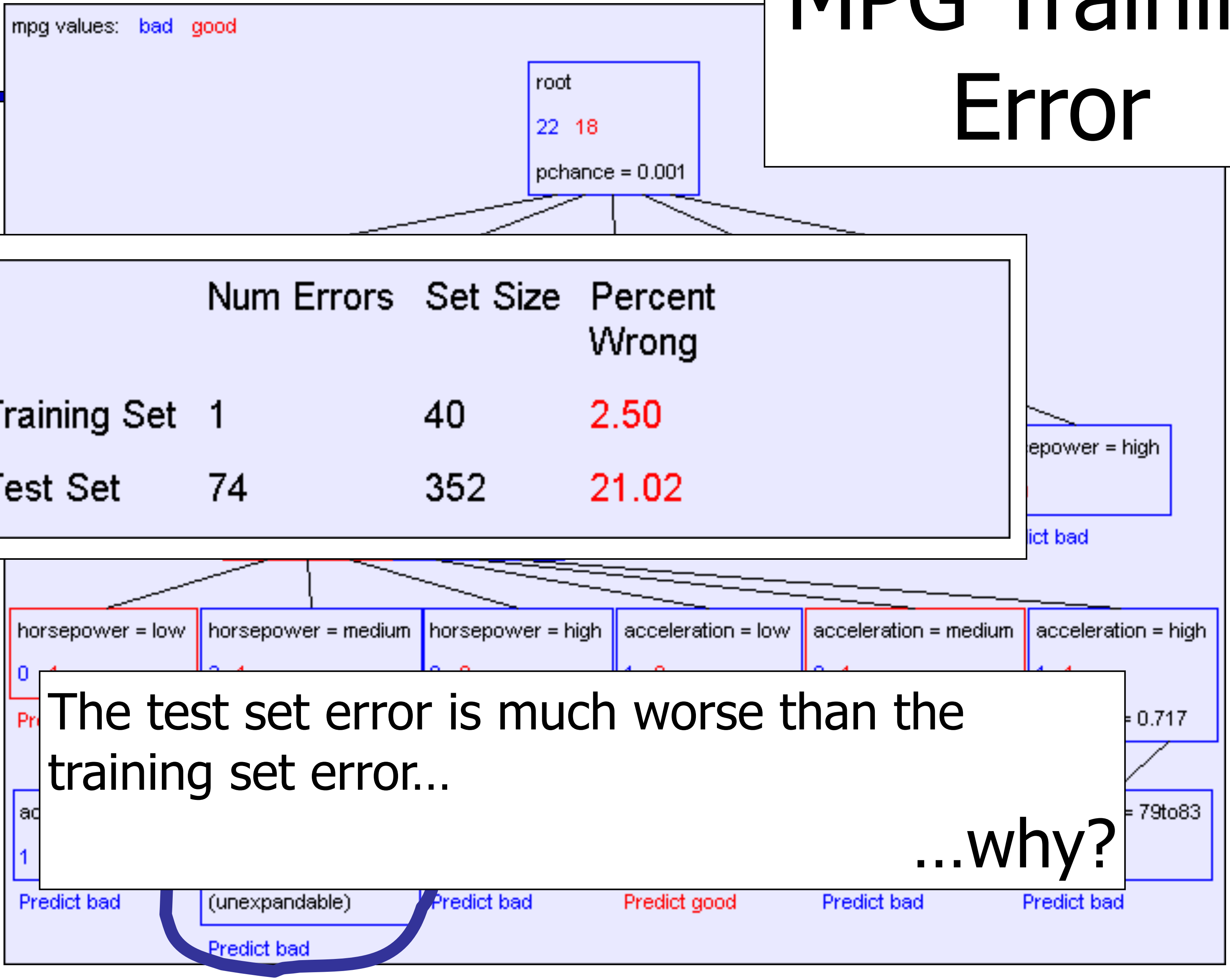


Final Tree



root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower = low
0 0
Predict bad
horsepower = medium
0 1
Predict good
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
horsepower = high
0 0
Predict bad
acceleration = low
1 0
Predict bad
acceleration = medium
0 1
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
Predict bad
Predict good
Predict bad
0 0
Predict bad
Information gains using the training set (2 records)
mpg values: bad good
Input
Value
Distribution
Info Gain
cylinders
3
0
4
5
6
8
displacement
low
0
medium
high
horsepower
low
0
medium
high
weight
low
0
medium
high
acceleration
low
0
medium
high
modelyear
70to74
0
75to78
79to83
maker
america
0
asia
europe

MPG Training Error



	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02

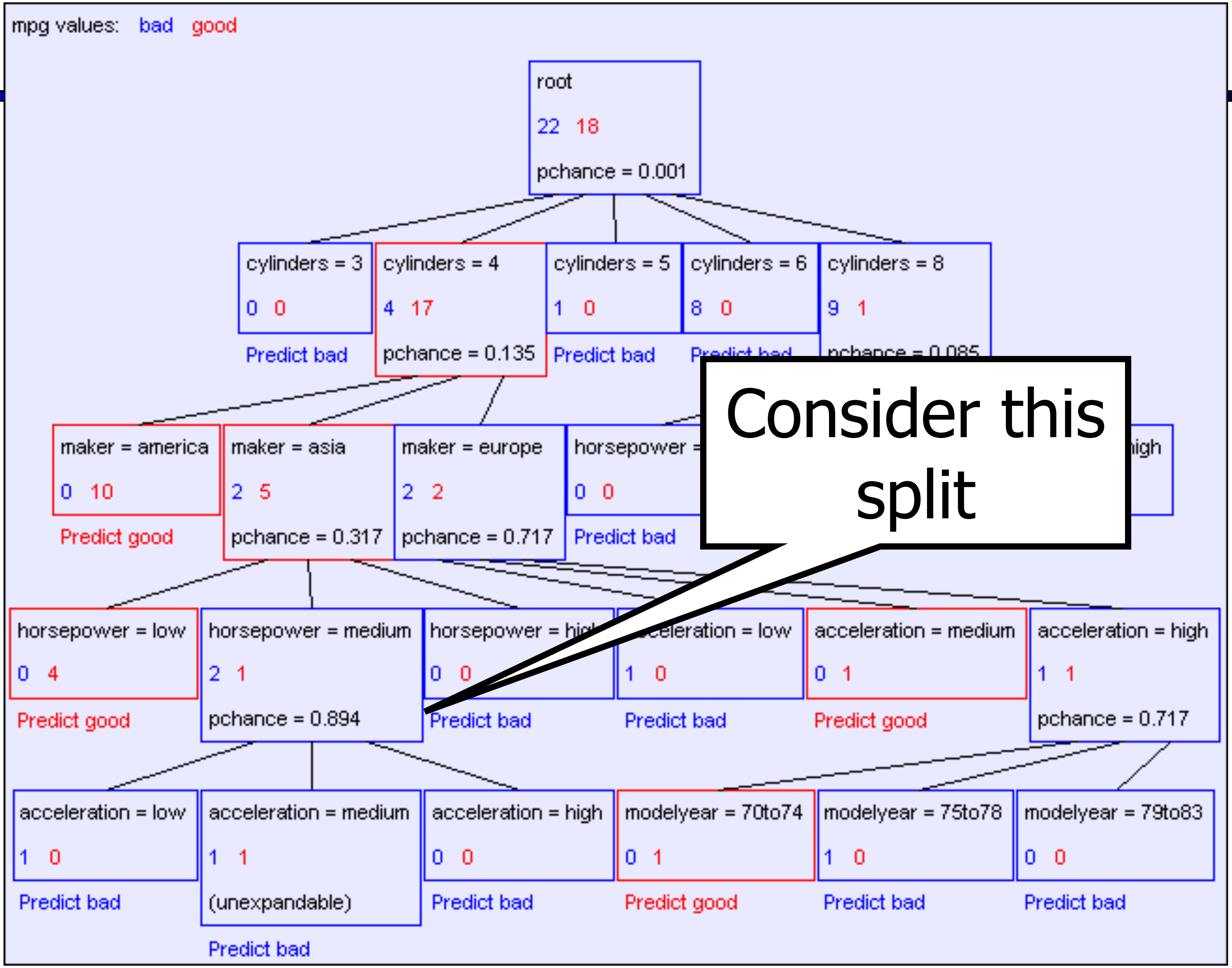
The test set error is much worse than the training set error…

…why?

Reminder: Overfitting

- Overfitting:
  - When you stop modeling the patterns in the training data (which generalize)
  - And start modeling the noise (which doesn’t)

- We had this before:
  - Naïve Bayes: needed to smooth
  - Perceptron: early stopping


mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
cylinders = 8
9 1
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
0 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
0 0
Predict bad
1 0
Predict bad
acceleration = medium
0 1
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad
Consider this split

Keeping it General

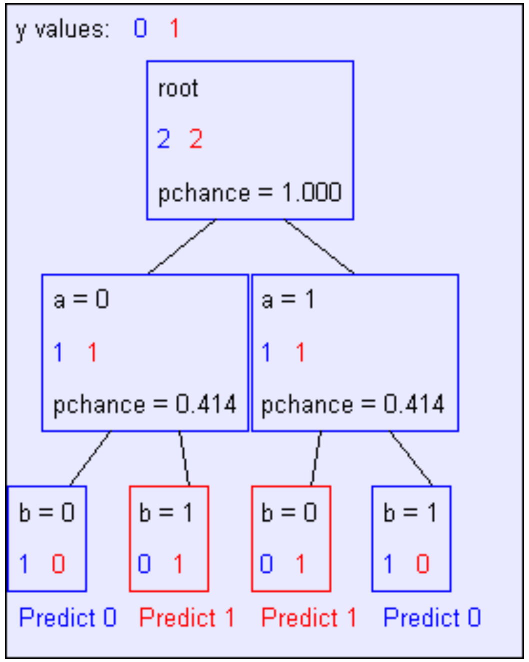
- Pruning:
  - Build the full decision tree
  - Begin at the bottom of the tree
  - Delete splits in which

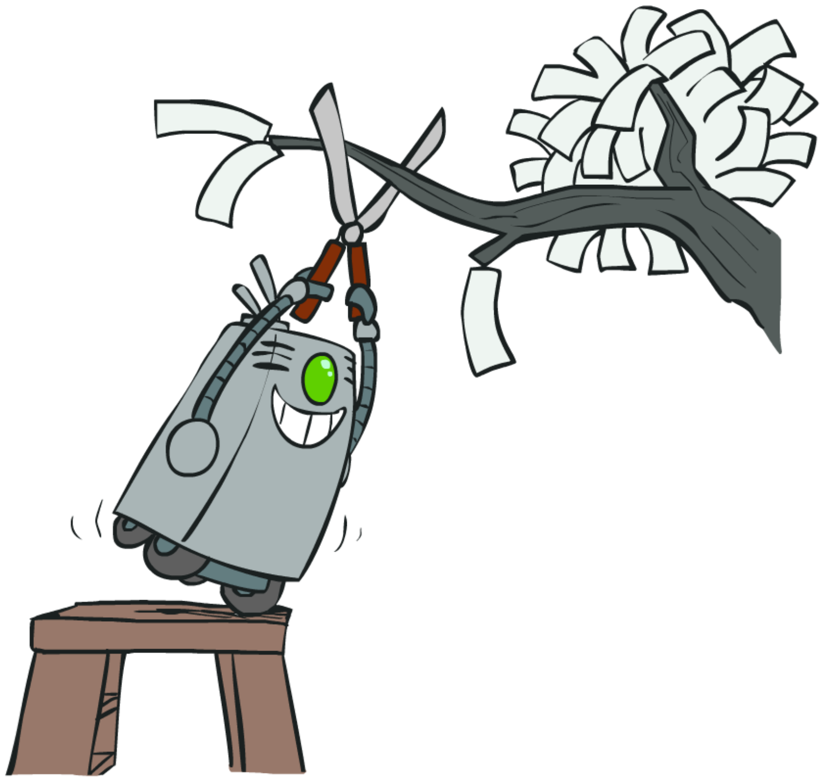
    $p_{CHANCE} > MaxP_{CHANCE}$

  - Continue working upward until there are no more prunable nodes
  - Note: some chance nodes may not get pruned because they were “redeemed” later

y = a XOR b

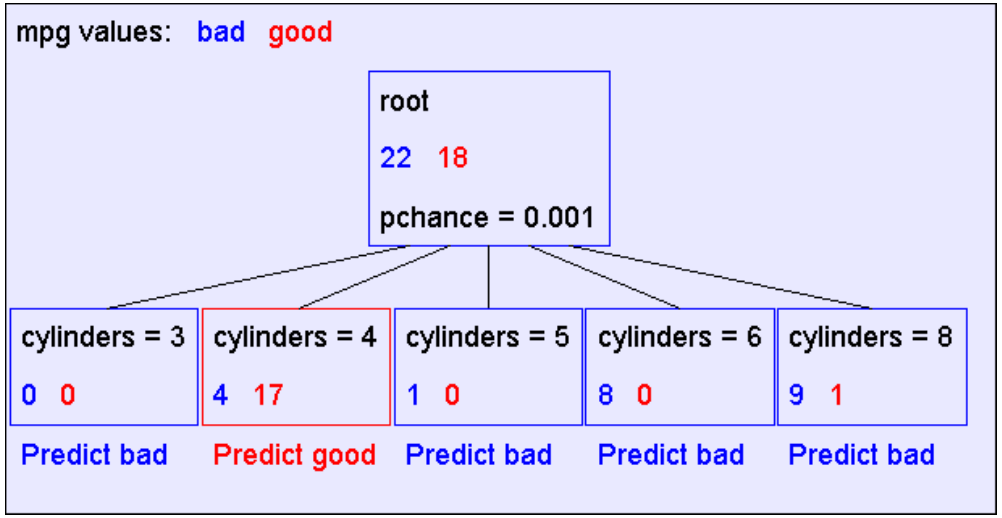
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0





Pruning example

- With $MaxP_{CHANCE} = 0.1$:

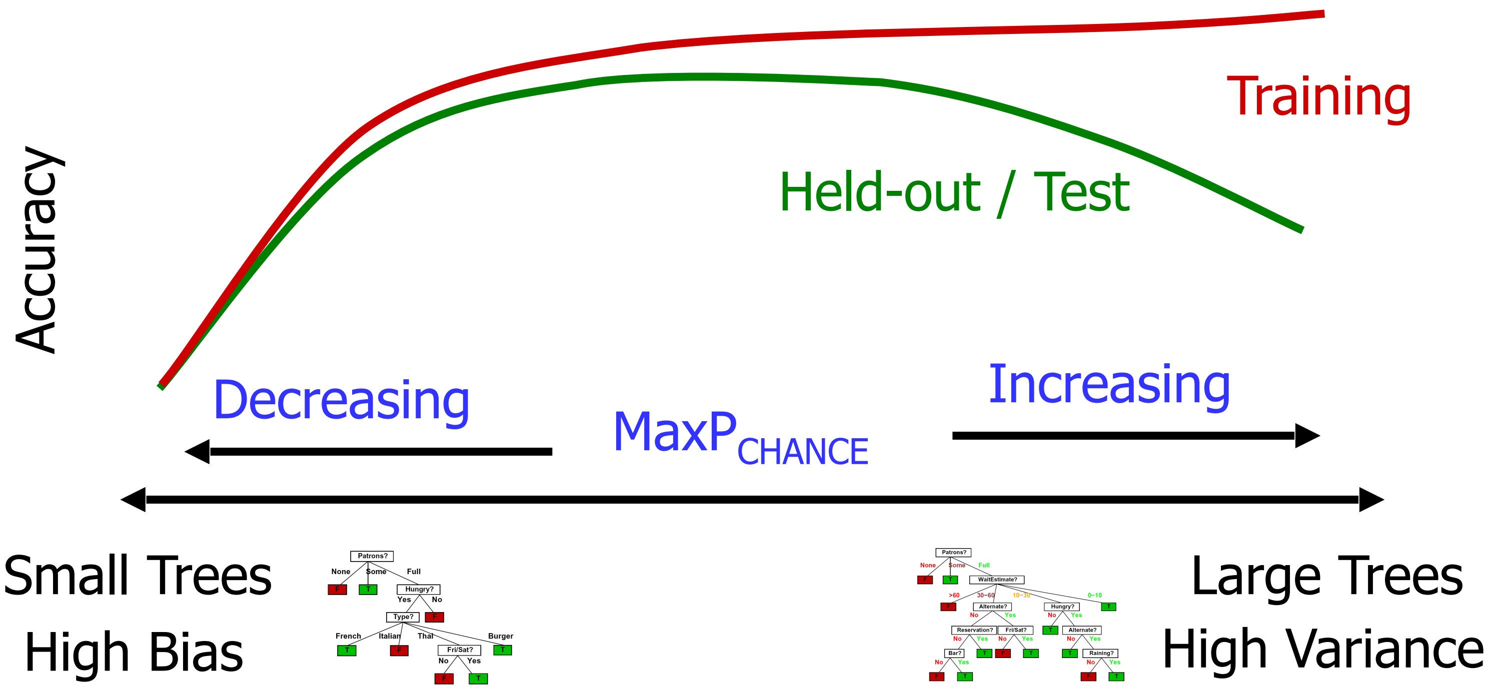


Note the improved test set accuracy compared with the unpruned tree

	Num Errors	Set Size	Percent Wrong
Training Set	5	40	12.50
Test Set	56	352	15.91

Regularization

- $\text{MaxP}_{\text{CHANCE}}$ is a regularization parameter
- Generally, set it using held-out data (as usual)

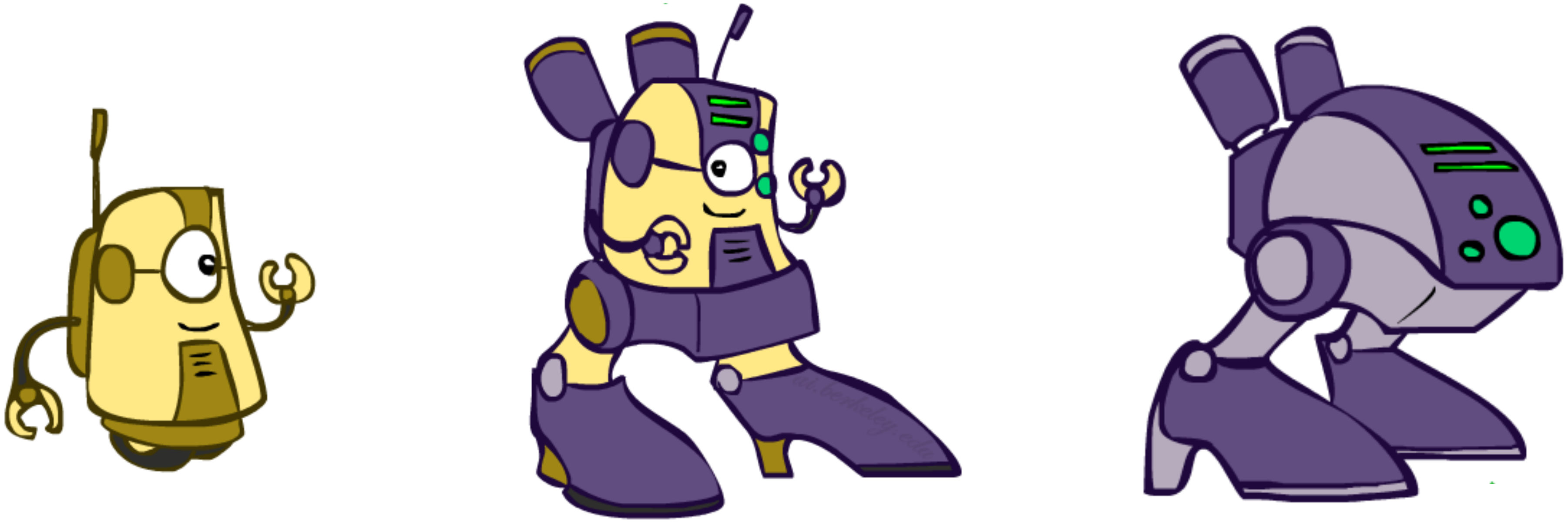


Two Ways of Controlling Overfitting

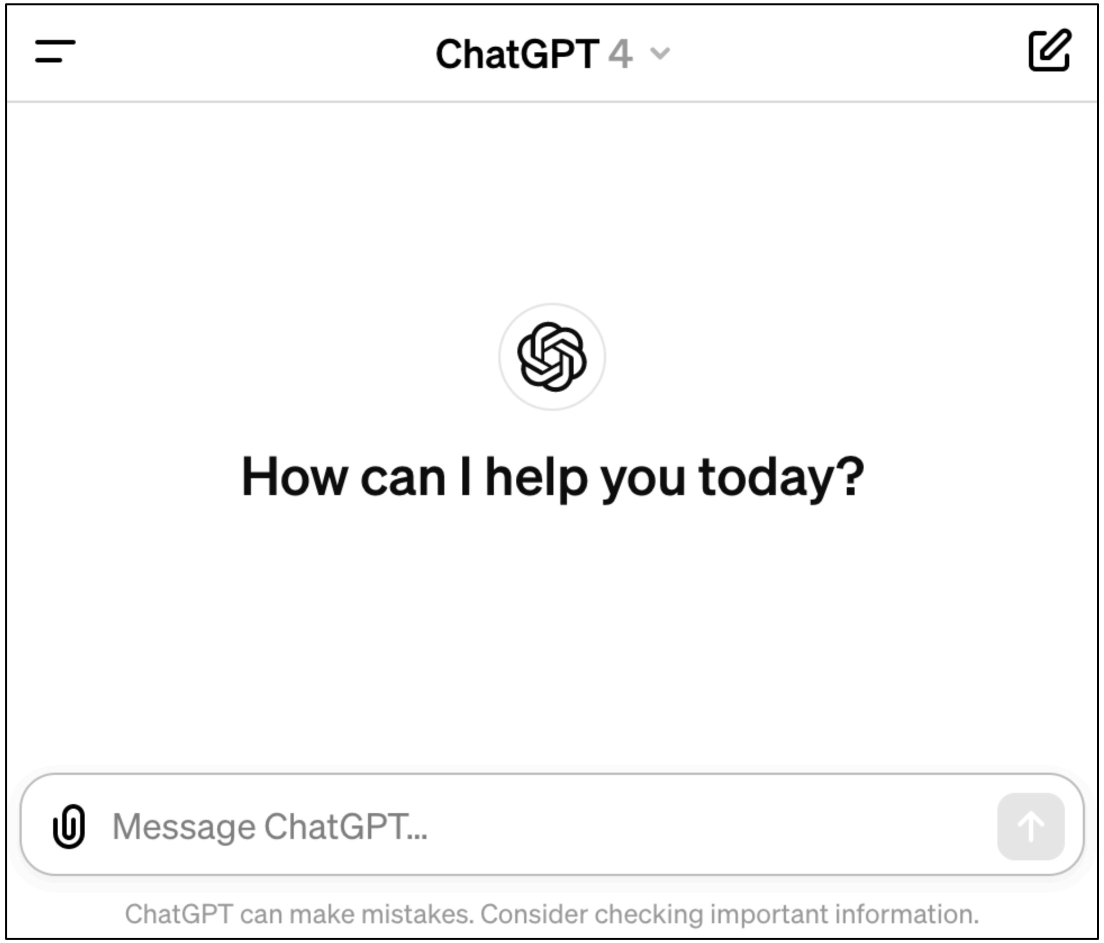
- Limit the hypothesis space
  - E.g. limit the max depth of trees
  - Easier to analyze

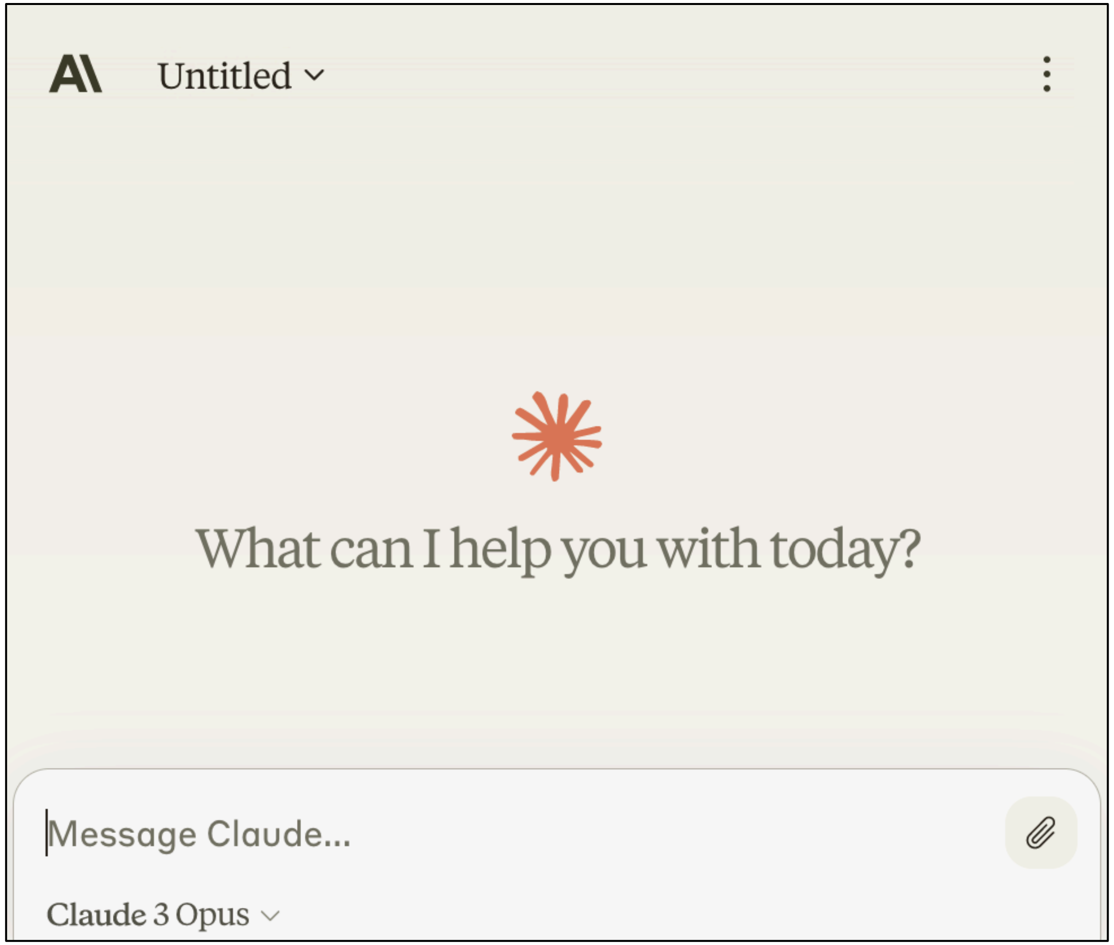
- Regularize the hypothesis selection
  - E.g. chance cutoff
  - Skip most of the hypotheses unless data is clear
  - Usually done in practice

Large Language Model Transformers



Today’s AI

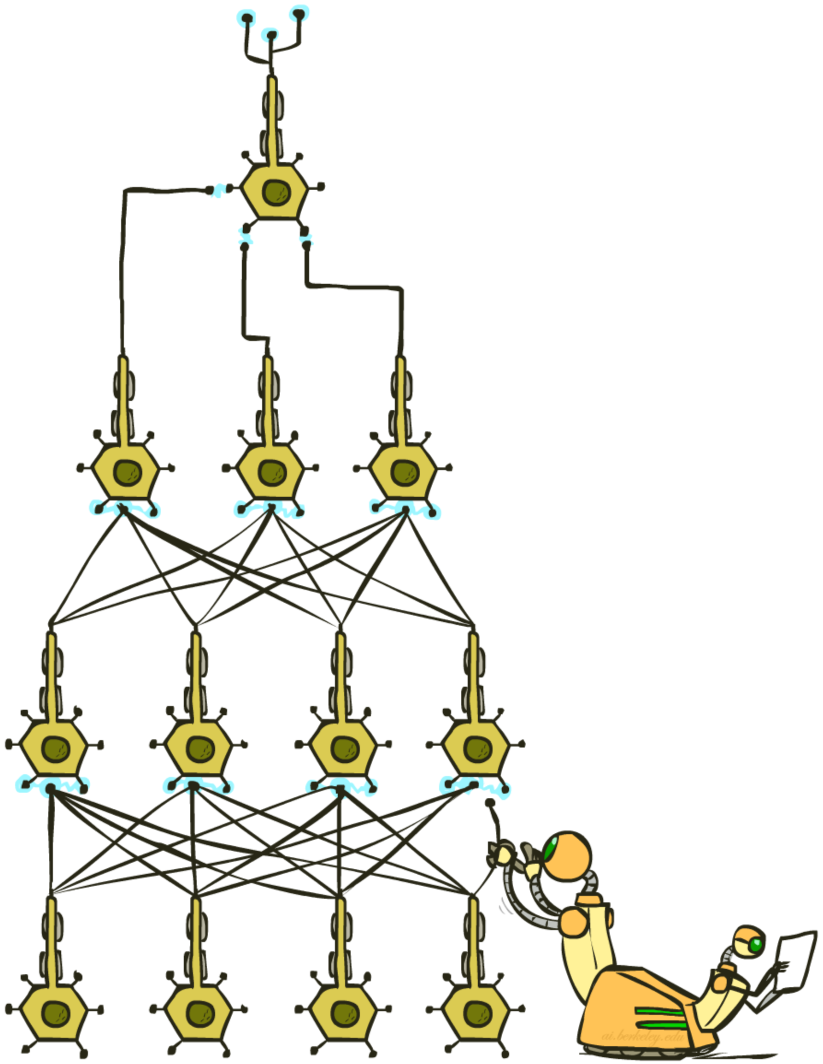

ChatGPT 4
How can I help you today?
Message ChatGPT...
ChatGPT can make mistakes. Consider checking important information.


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

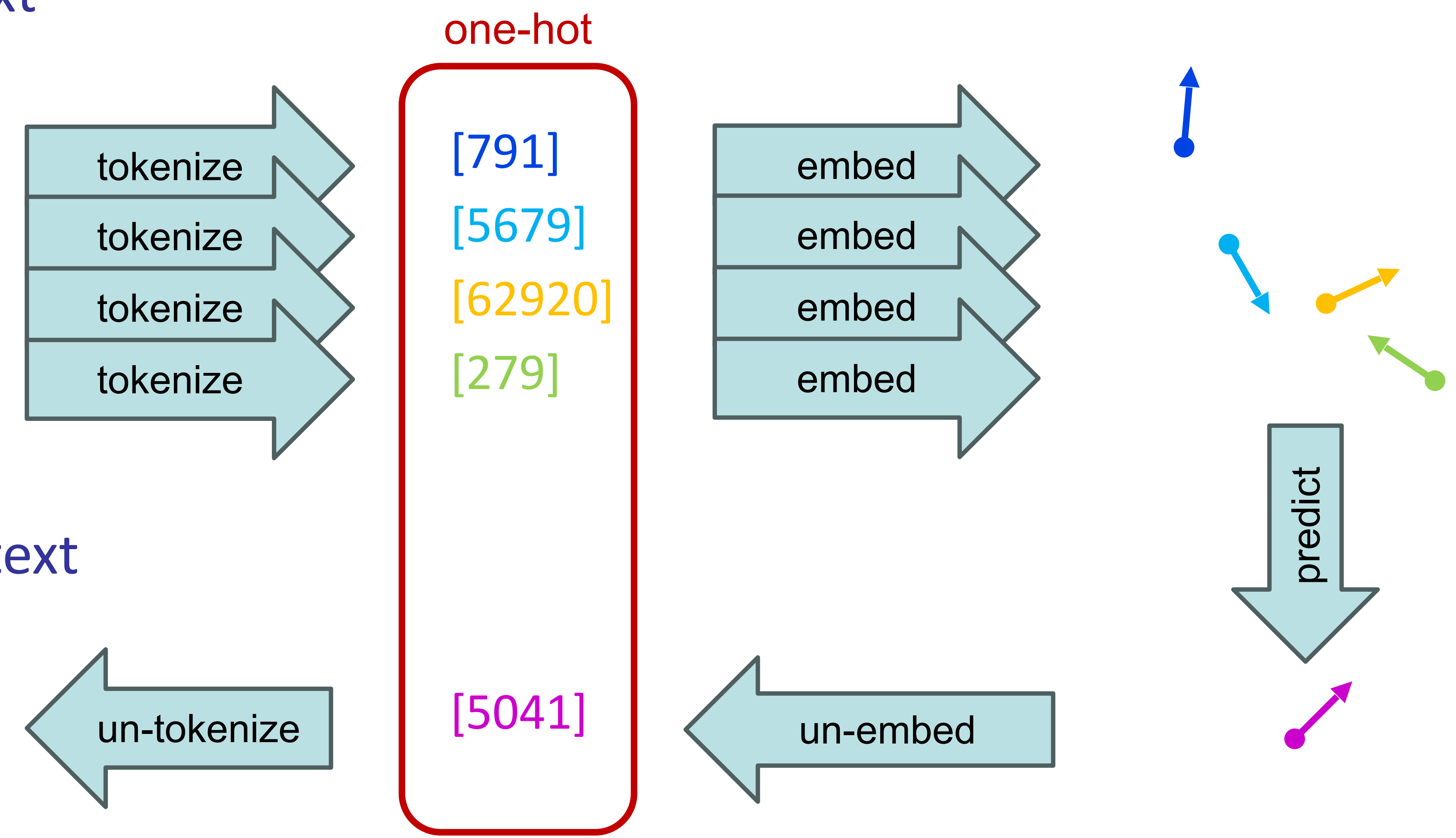
Tokens 57

Characters 252

https://platform.openai.com/tokenizer

Word Embeddings

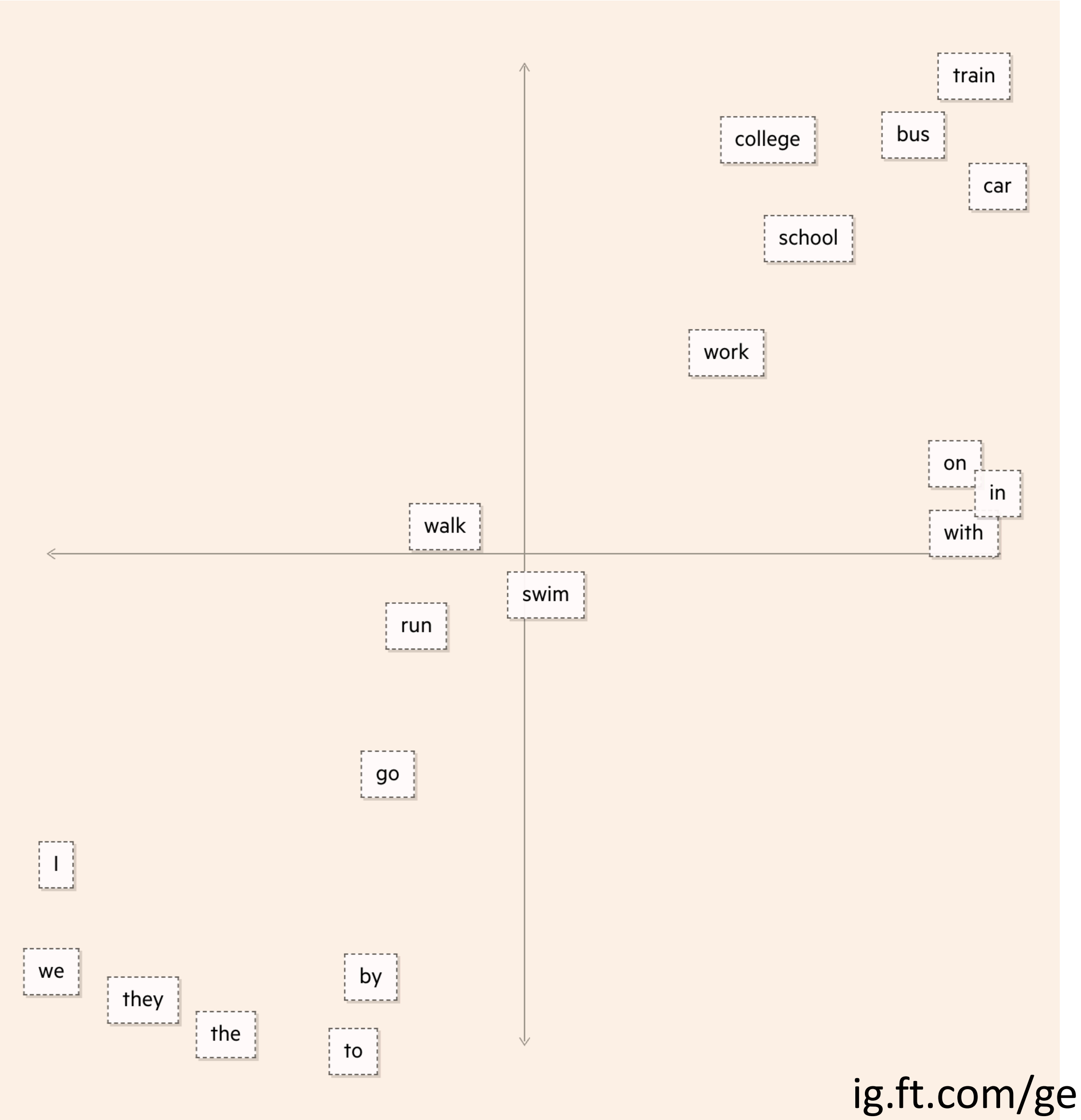
- Input: some text
  - “The”
  - “ dog”
  - “ chased”
  - “ the”



- Output: more text
  - “ ball”

What do word embeddings look like?

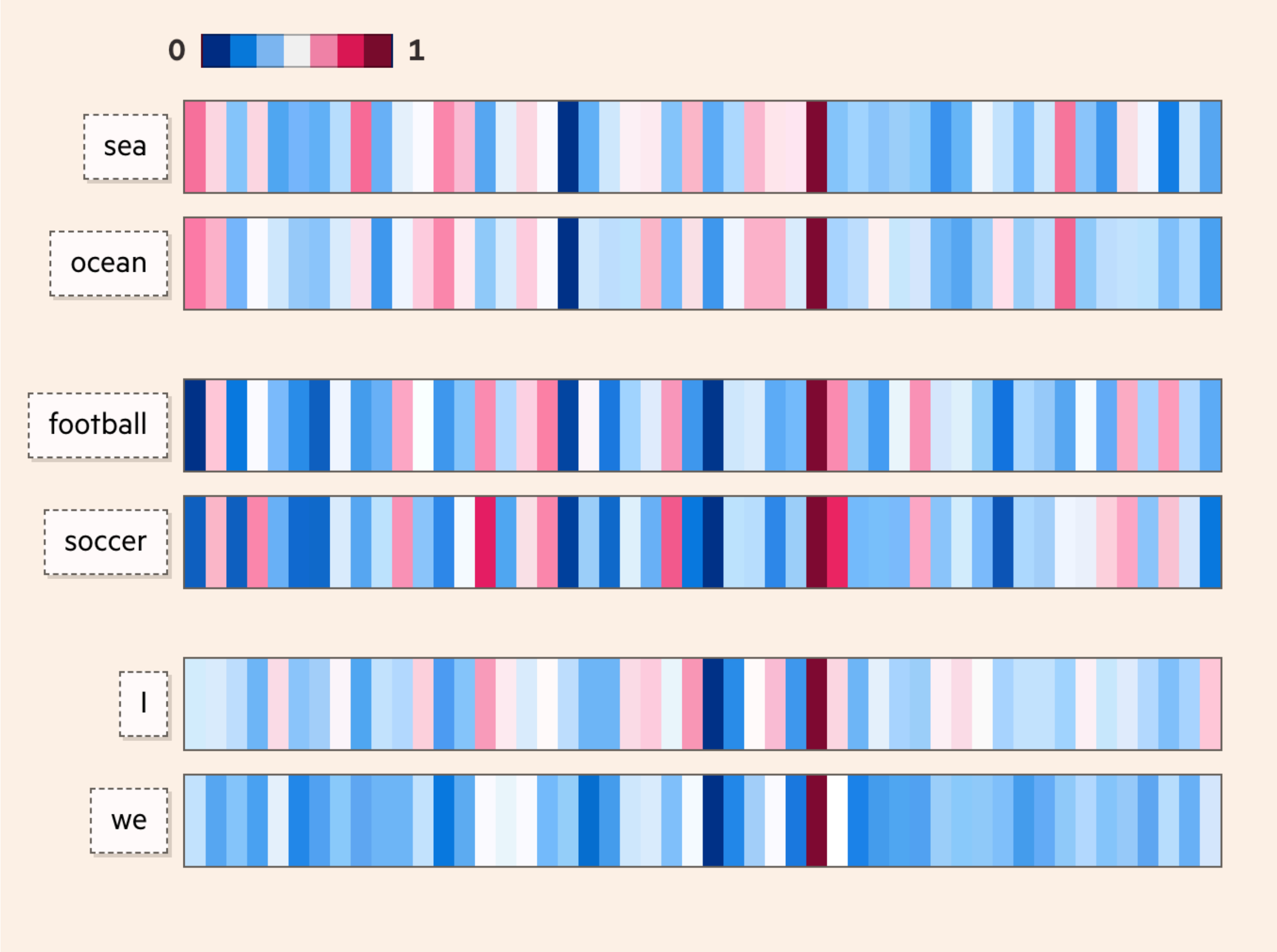
- Words cluster by similarity:



ig.ft.com/generative-ai

What do word embeddings look like?

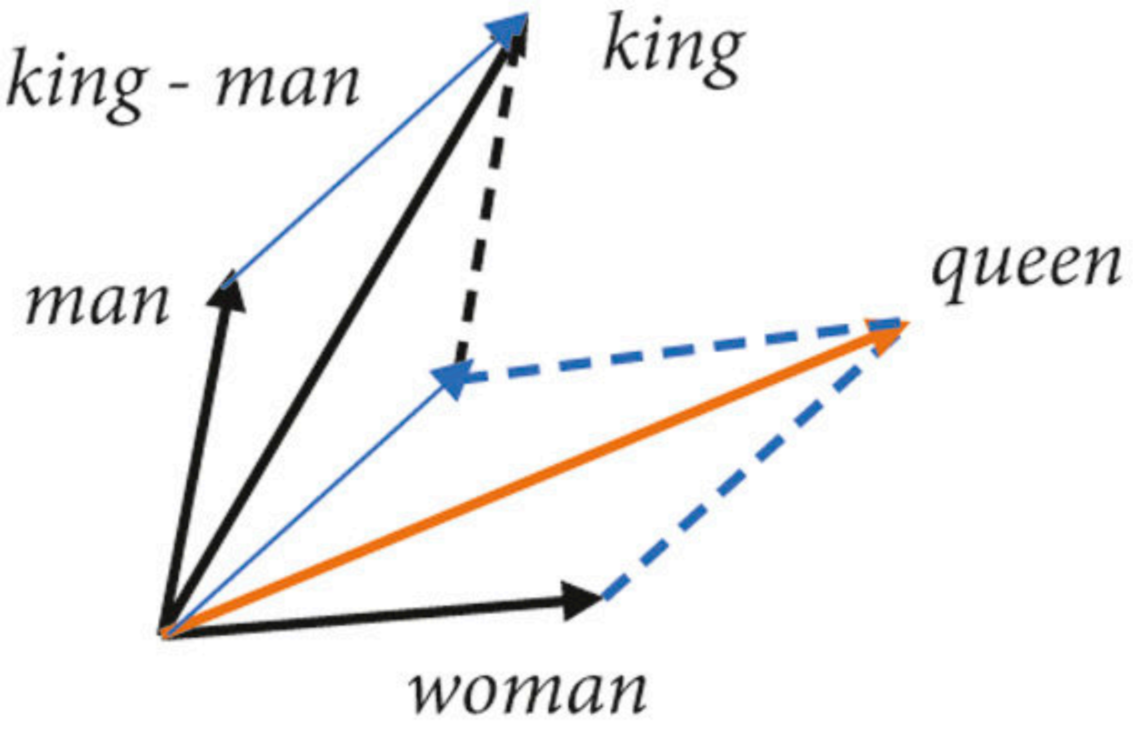
- Features learned in language models:



ig.ft.com/generative-ai

What do word embeddings look like?

- Signs of sensible algebra in embedding space:



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

[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

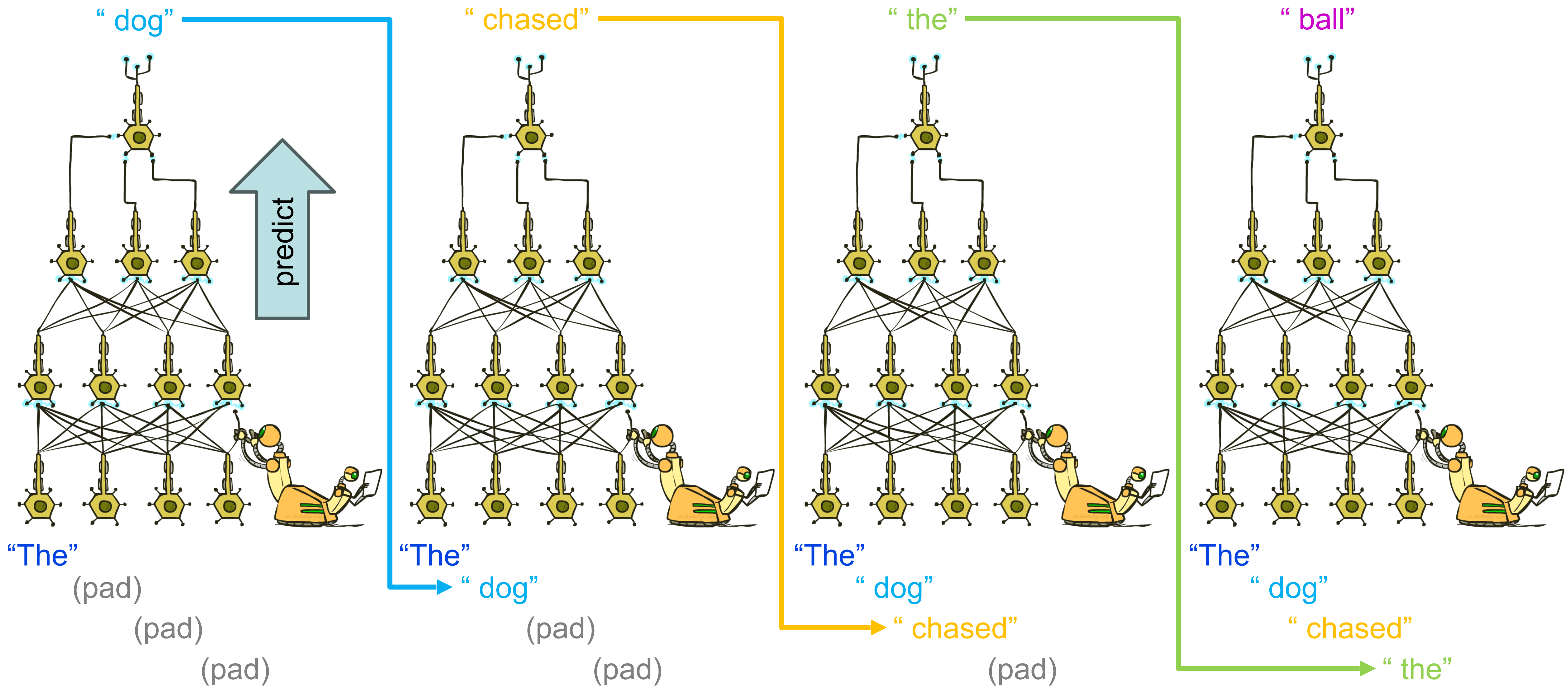
This is how it works

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

Large Language Models

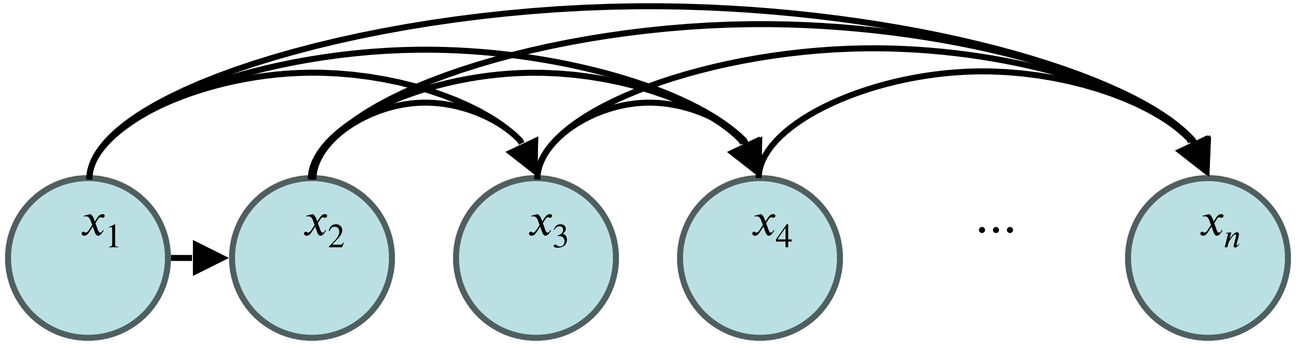
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- Deep neural networks
  - Autoregressive models
  - Self-attention mechanisms
  - Transformer architectures
- Multi-class classification
- Supervised learning
  - Self-supervised learning
  - Instruction tuning
- Reinforcement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

Autoregressive Models

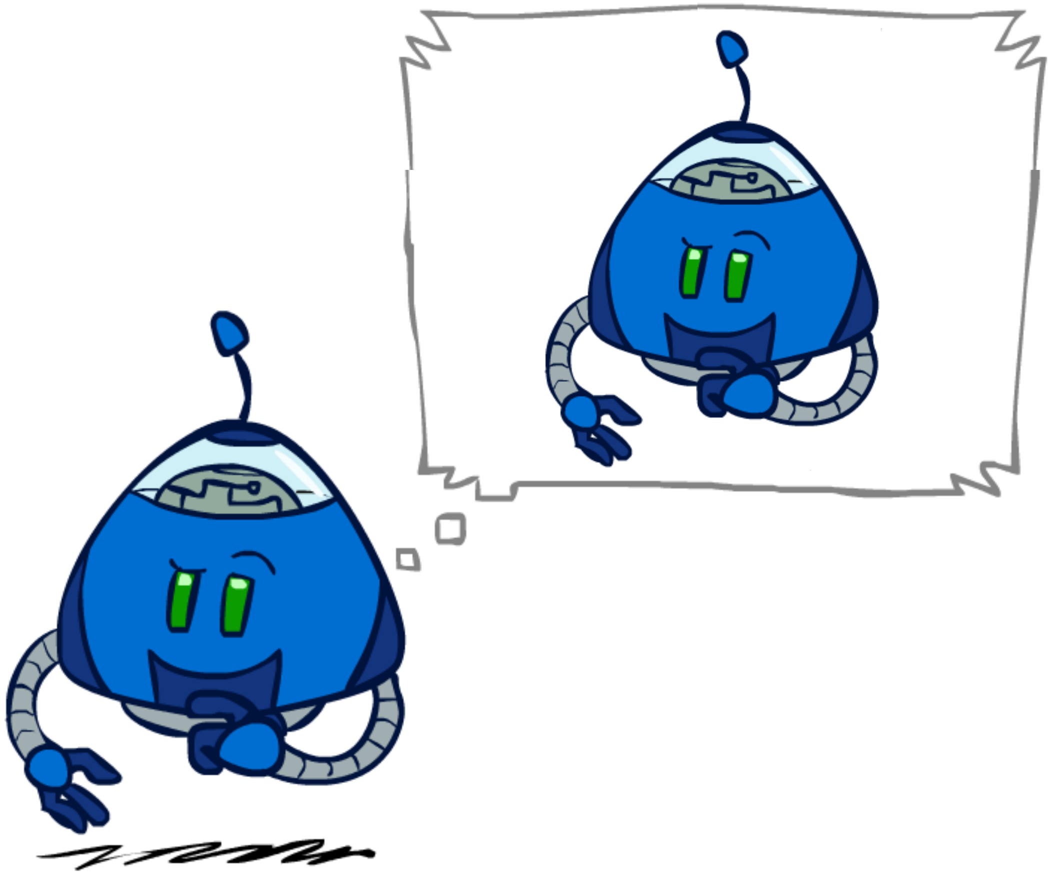


Autoregressive Models

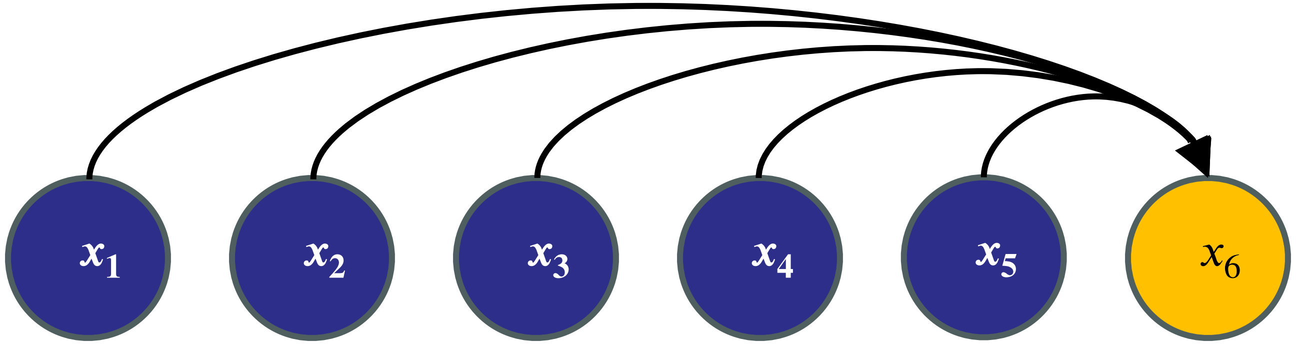
- Predict output one piece at a time (e.g. word, token, pixel, etc.)
- Concatenate: input + output
- Feed result back in as new input
- Repeat



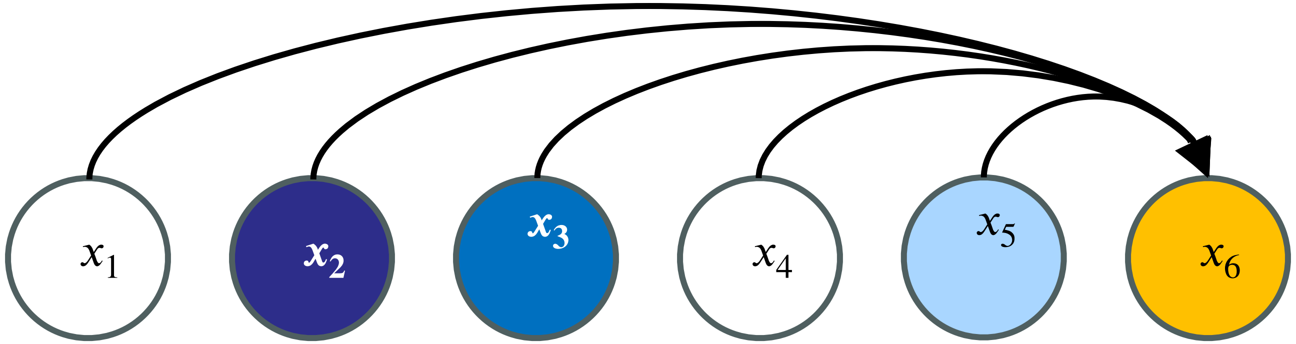
Self-Attention Mechanisms



Self-Attention Mechanisms

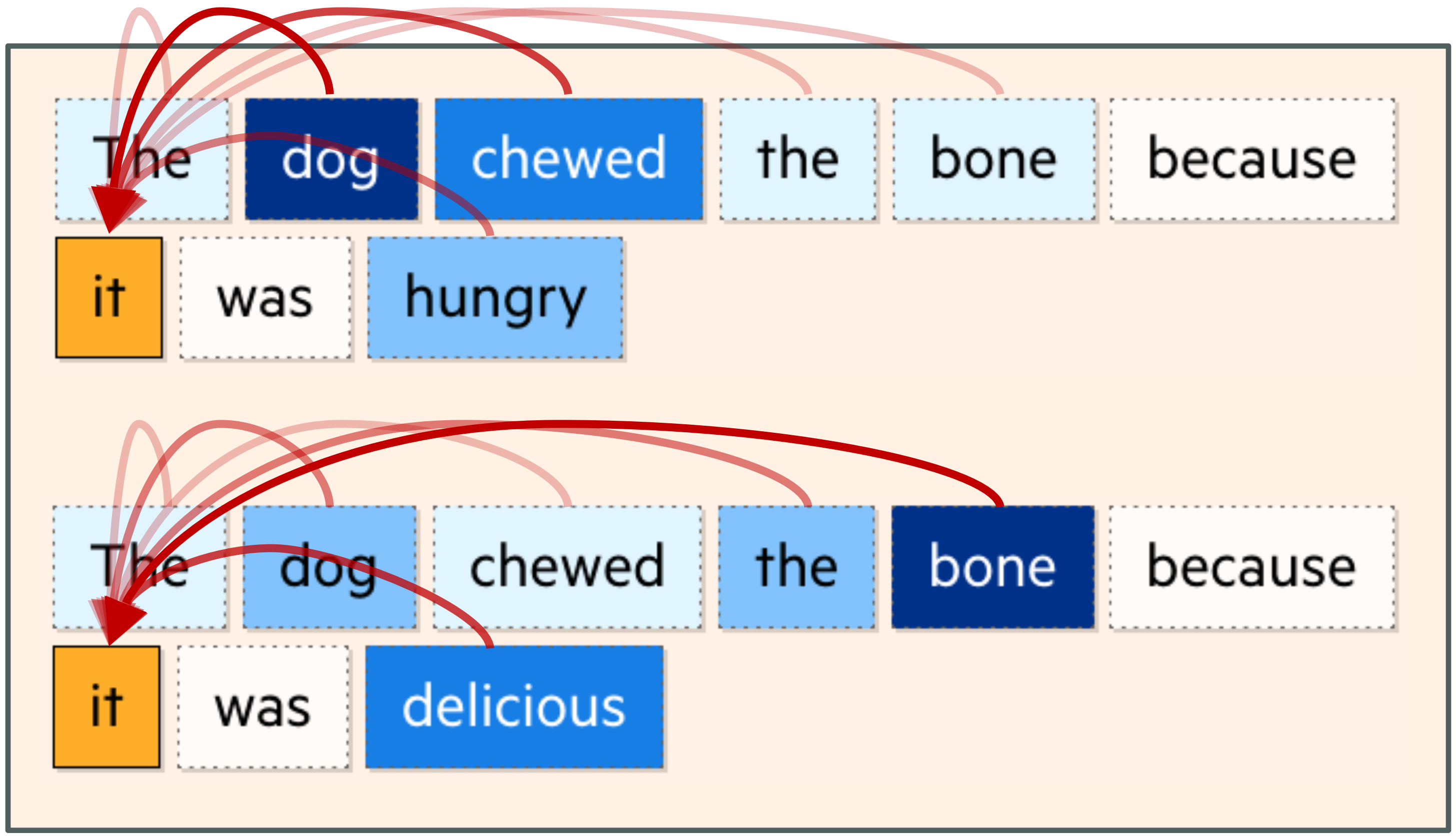


- Instead of conditioning on *all* input tokens equally…

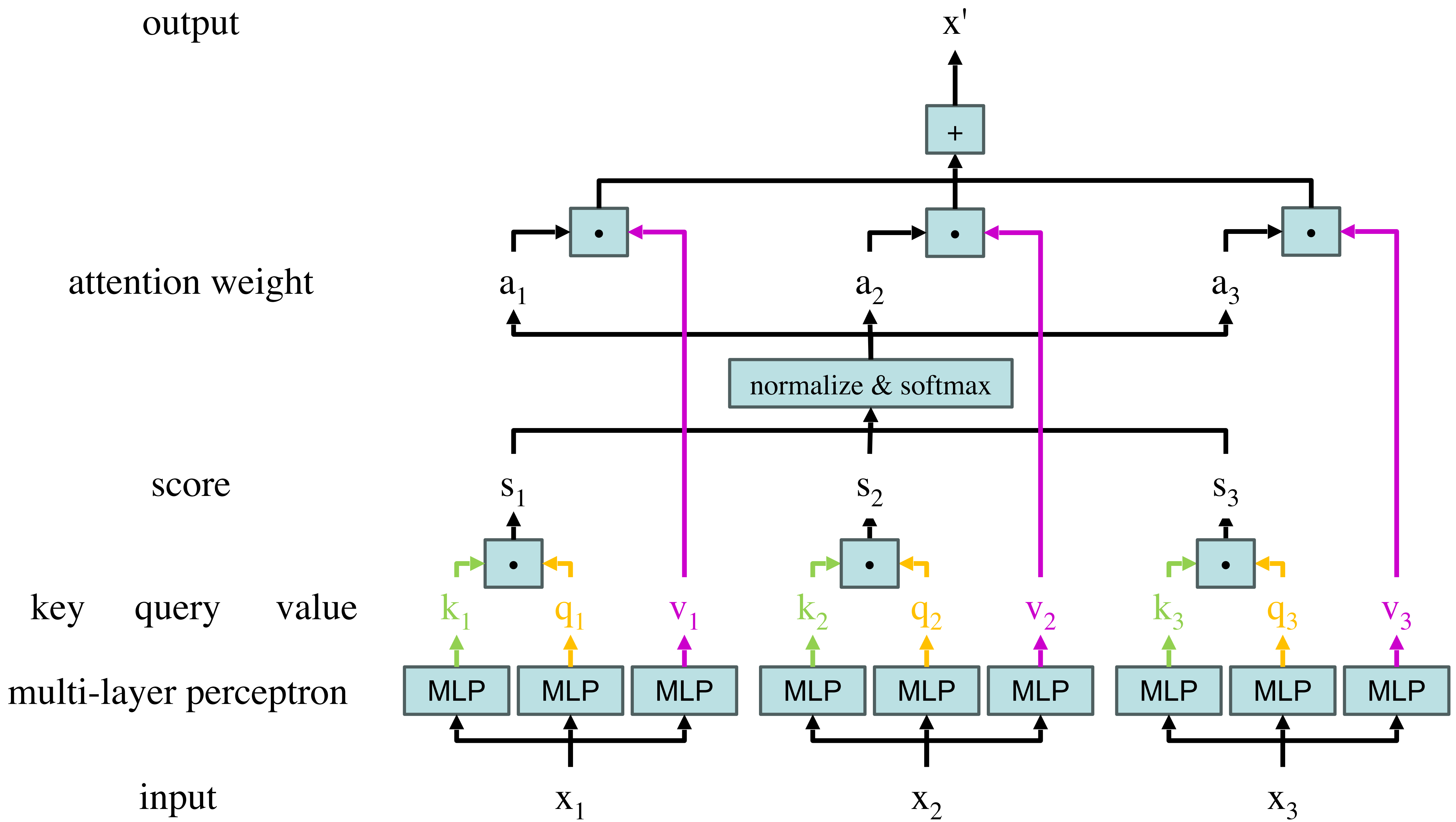


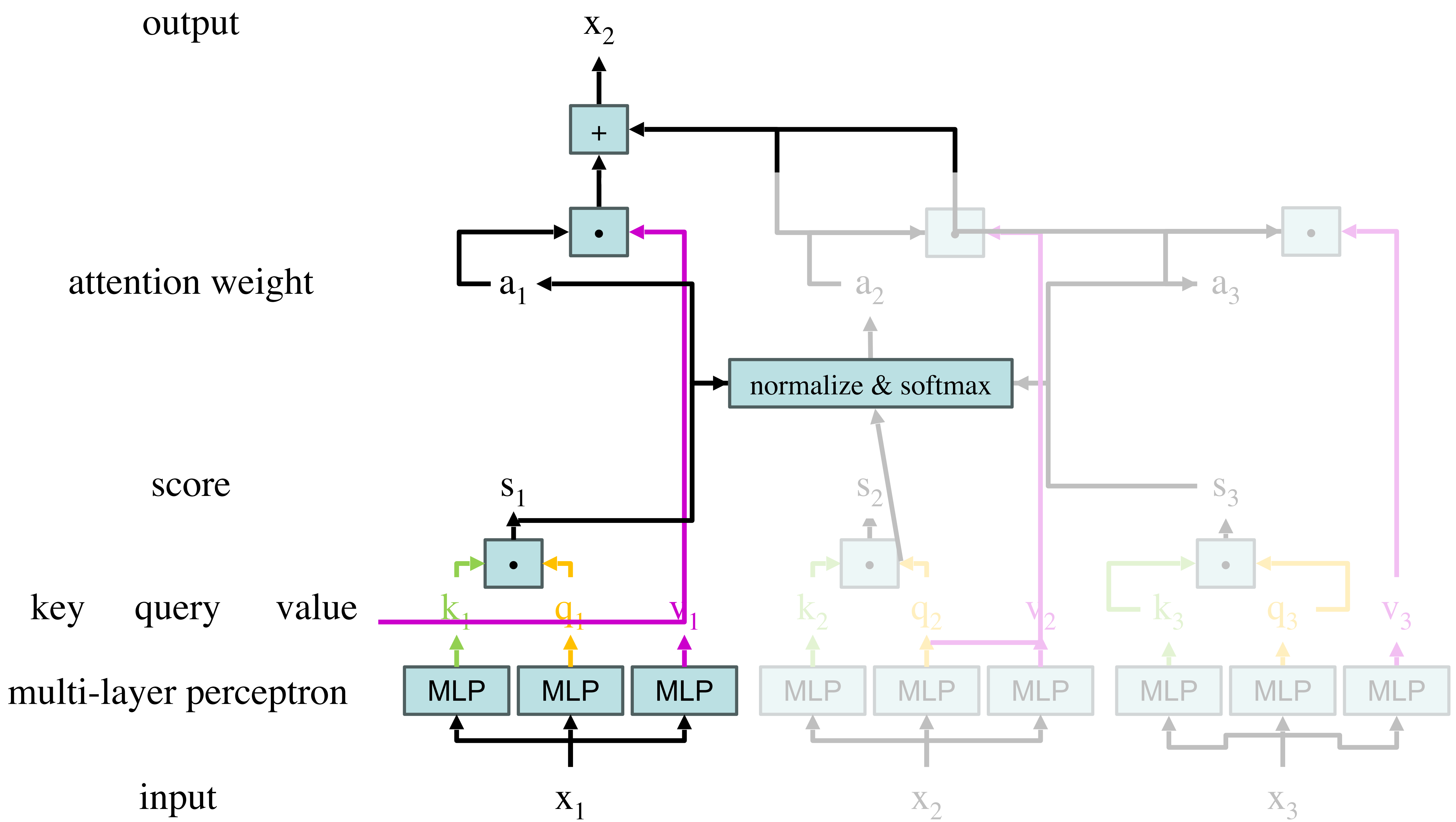
- Pay more attention to relevant tokens!

Self-Attention Mechanisms

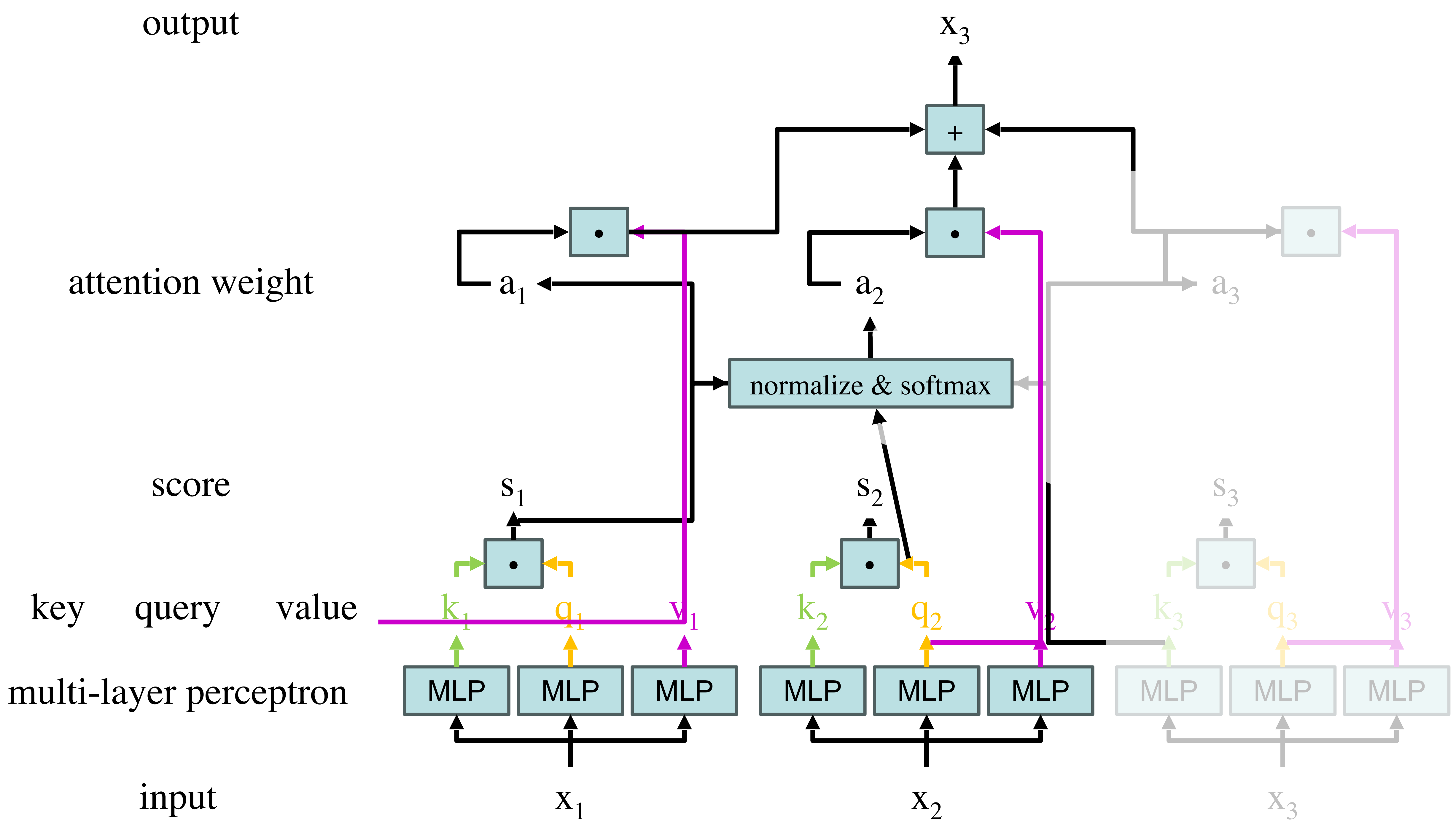


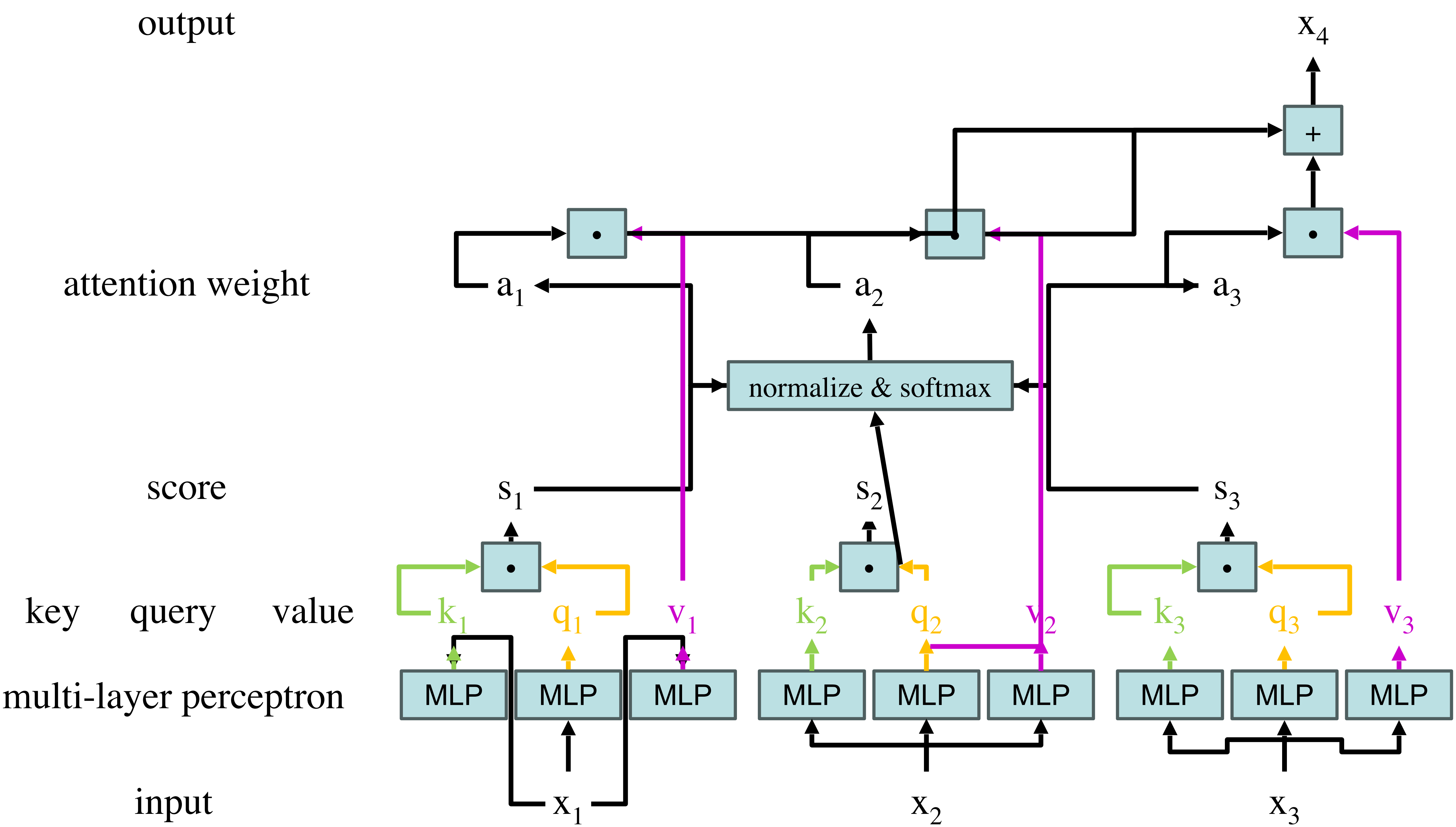
ig.ft.com/generative-ai


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

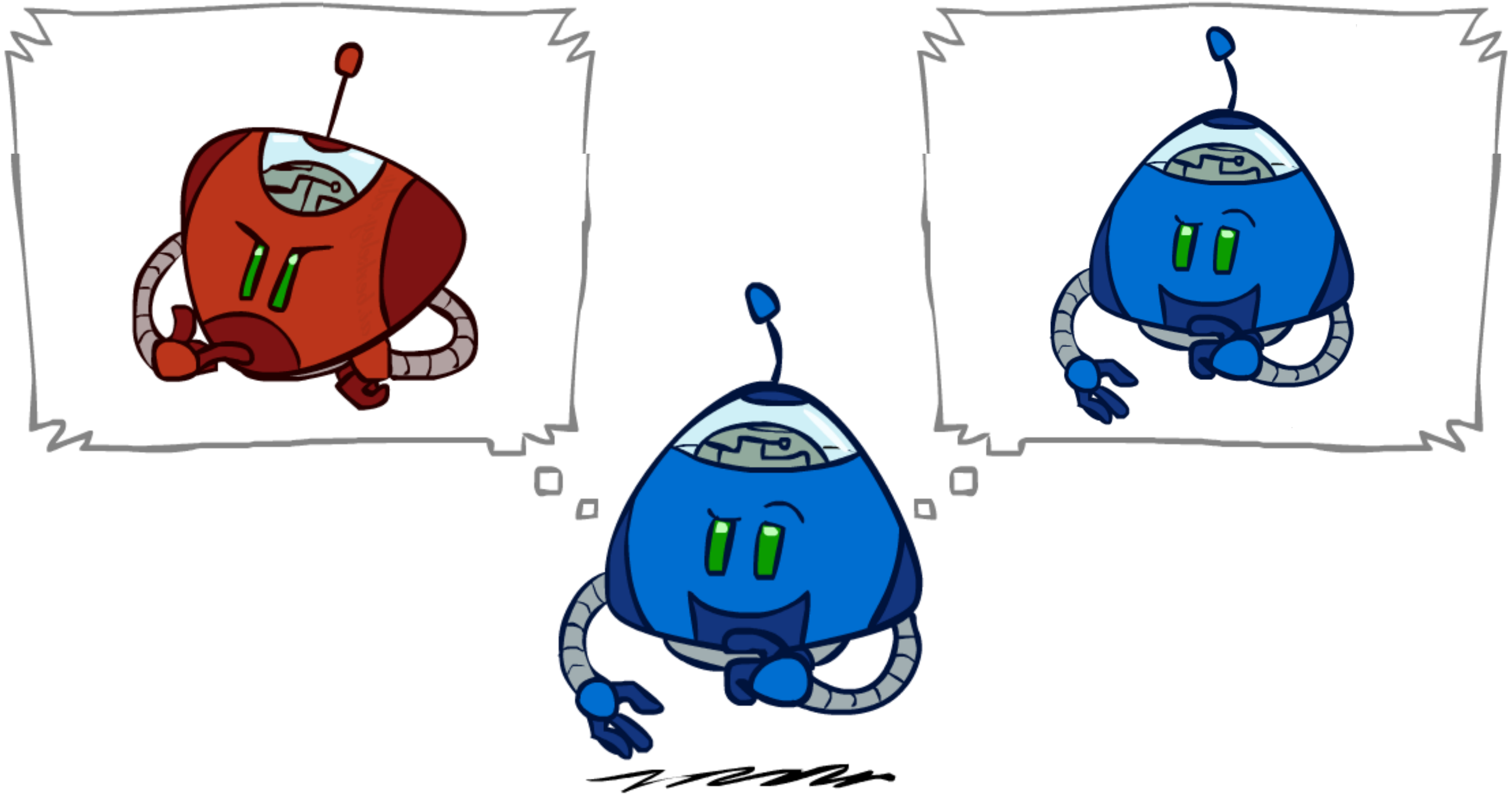


output
x2
+
attention weight
a1
a2
a3
normalize & softmax
score
s1
s2
s3
key
query
value
k1
q1
v1
k2
q2
v2
k3
q3
v3
multi-layer perceptron
MLP
input
x1
x2
x3


output
x_3
+
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

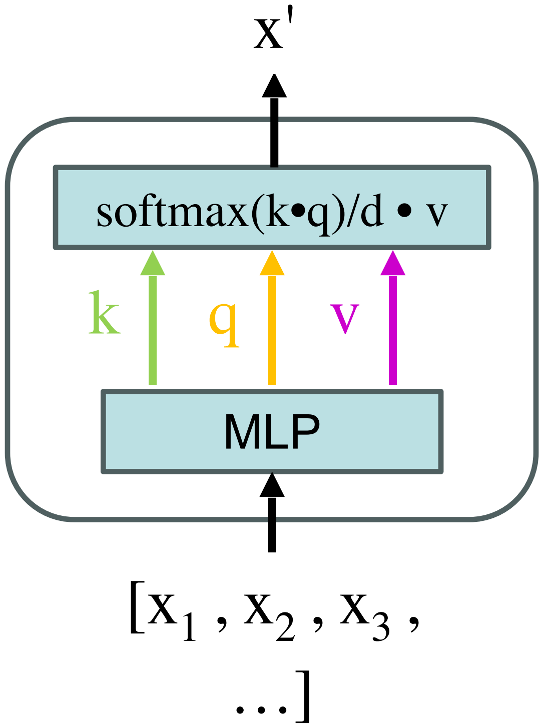

output
attention weight
score
key
query
value
multi-layer perceptron
input
x4
+
a1
a2
a3
normalize & softmax
s1
s2
s3
k1
q1
v1
k2
q2
v2
k3
q3
v3
MLP
x1
x2
x3

Multi-Headed Attention

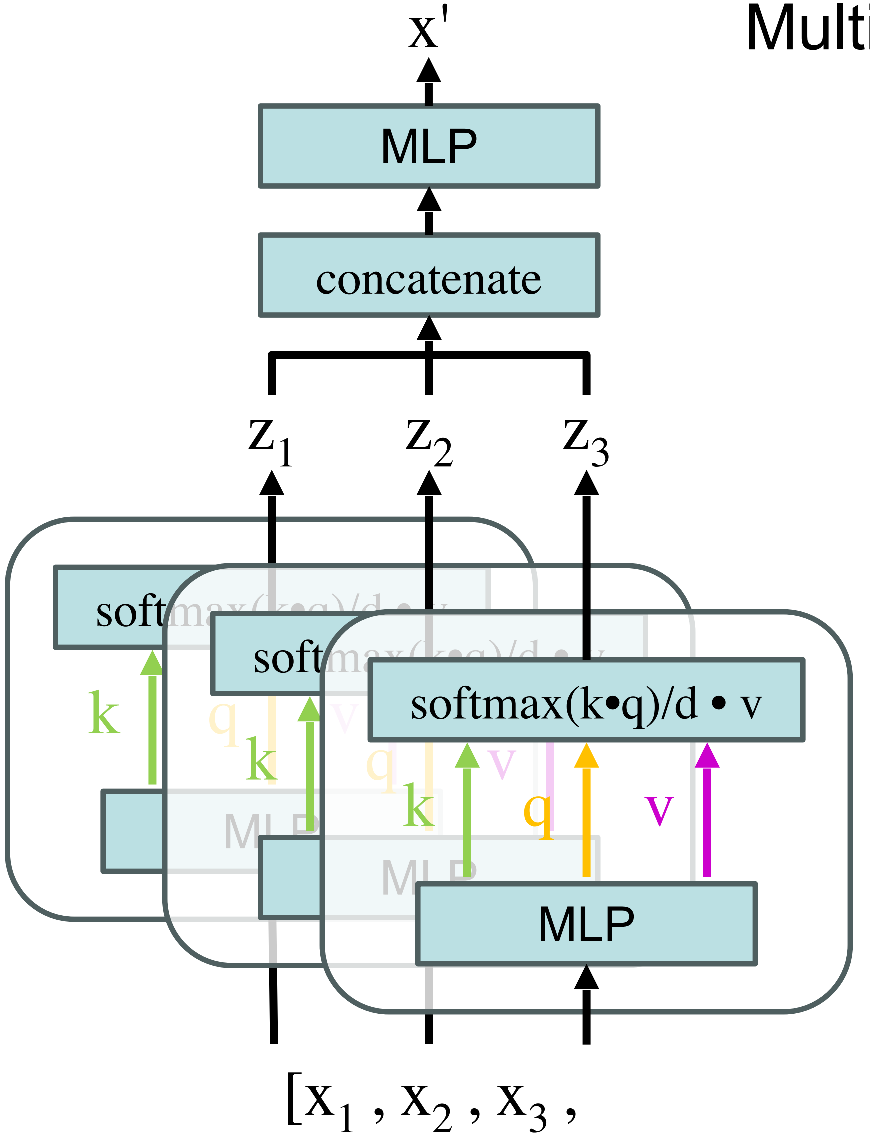


Multi-Headed Attention

Single-headed

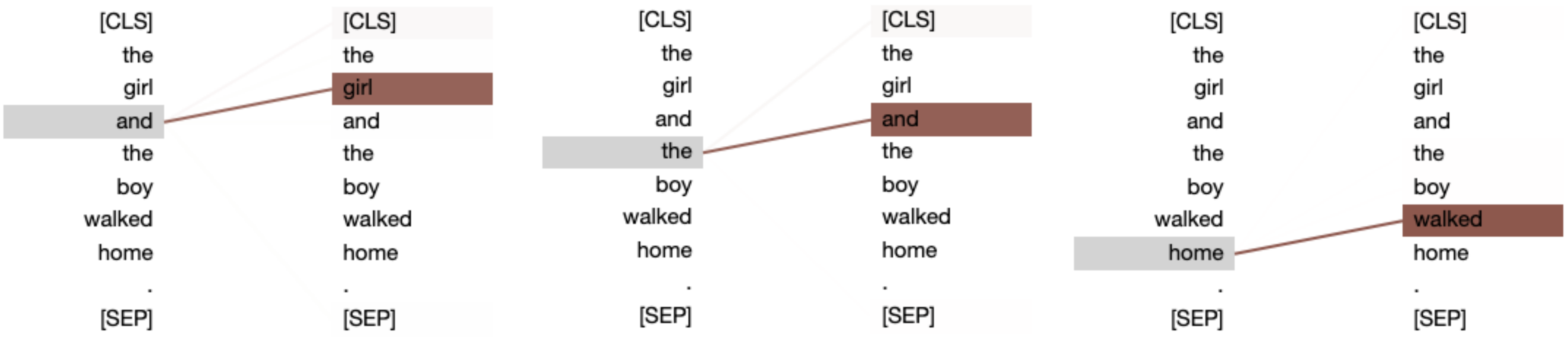


Multi-headed



Multi-Headed Attention

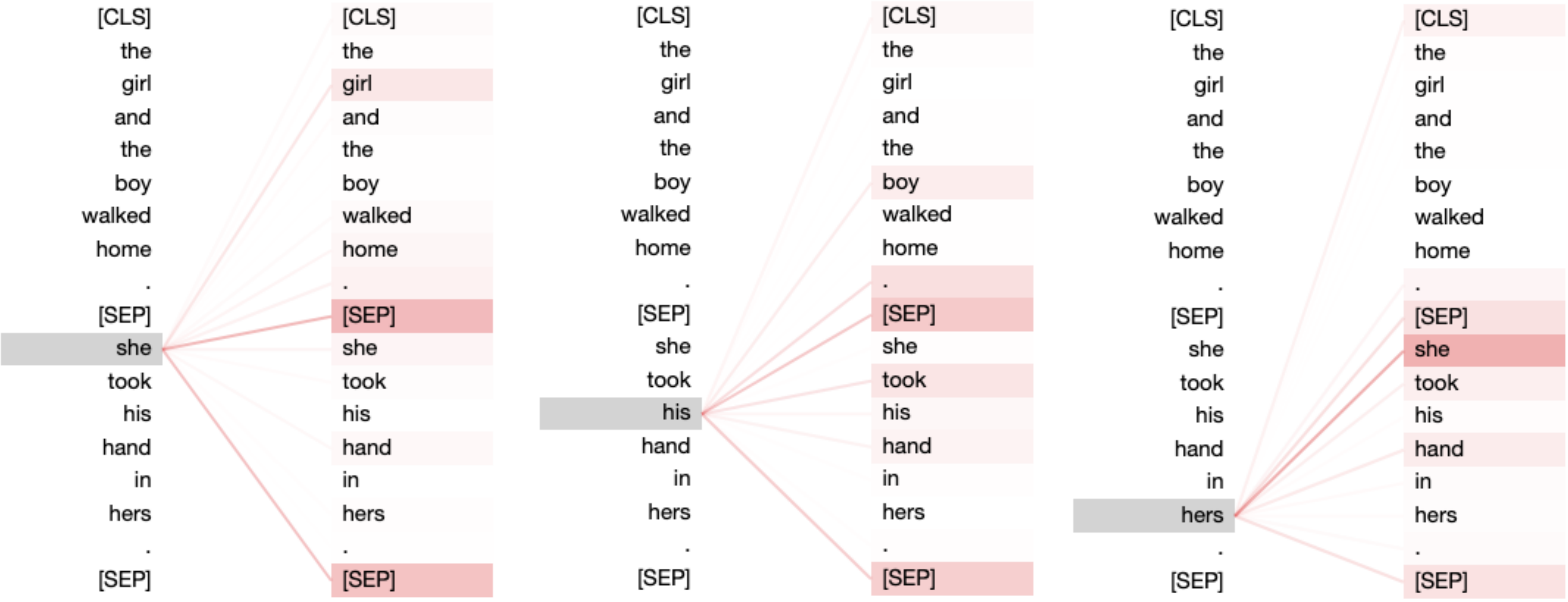
Head 6: previous word



https://github.com/jessevig/bertviz

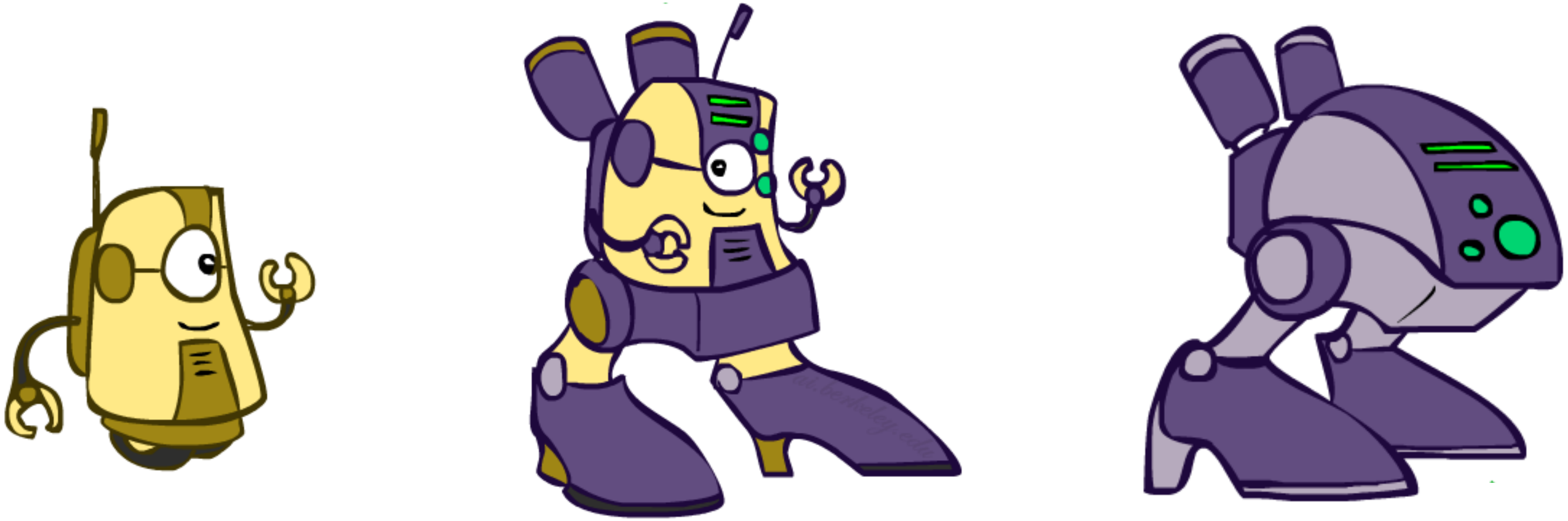
Multi-Headed Attention

Head 4: pronoun references

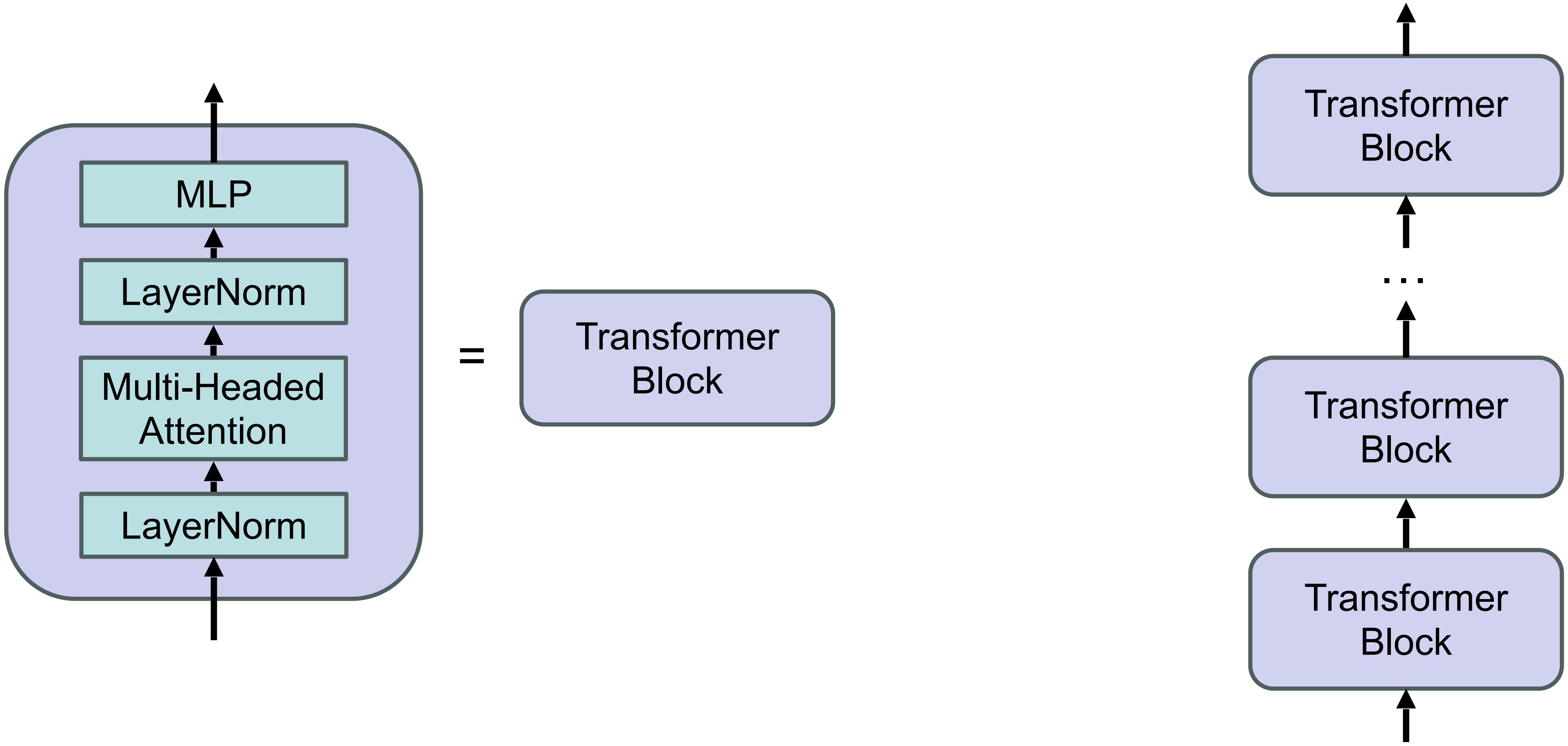


https://github.com/jessevig/bertviz

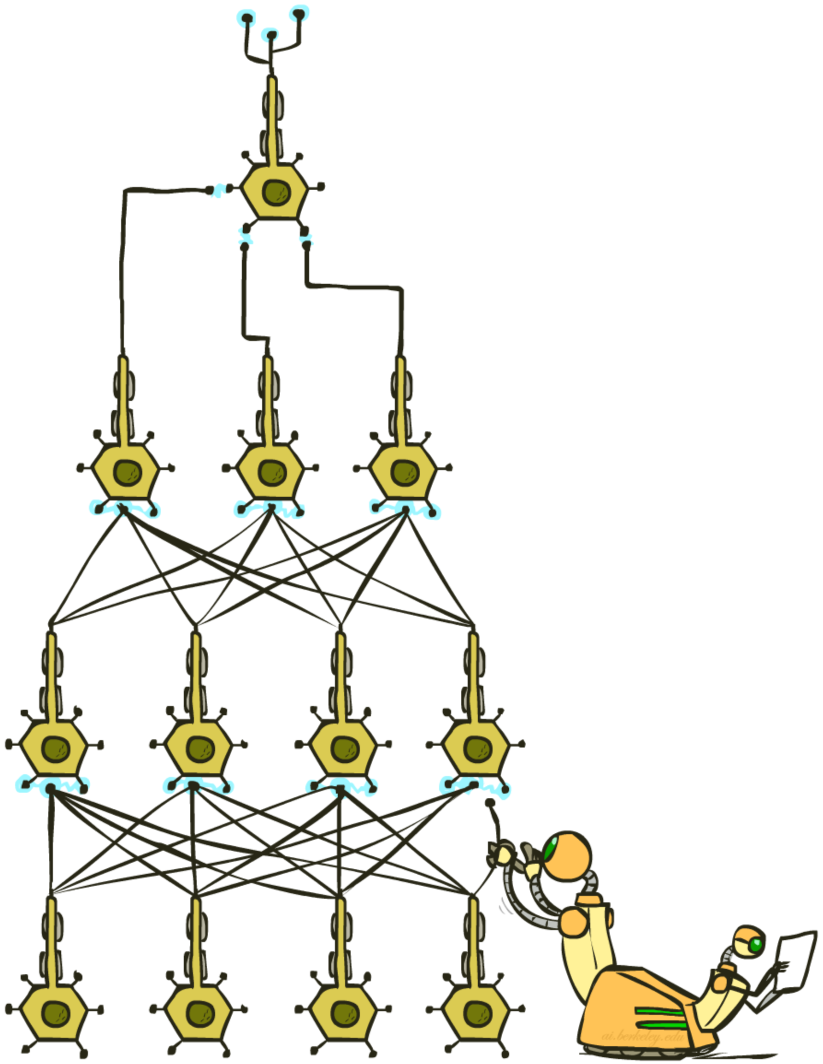
Transformer Architecture

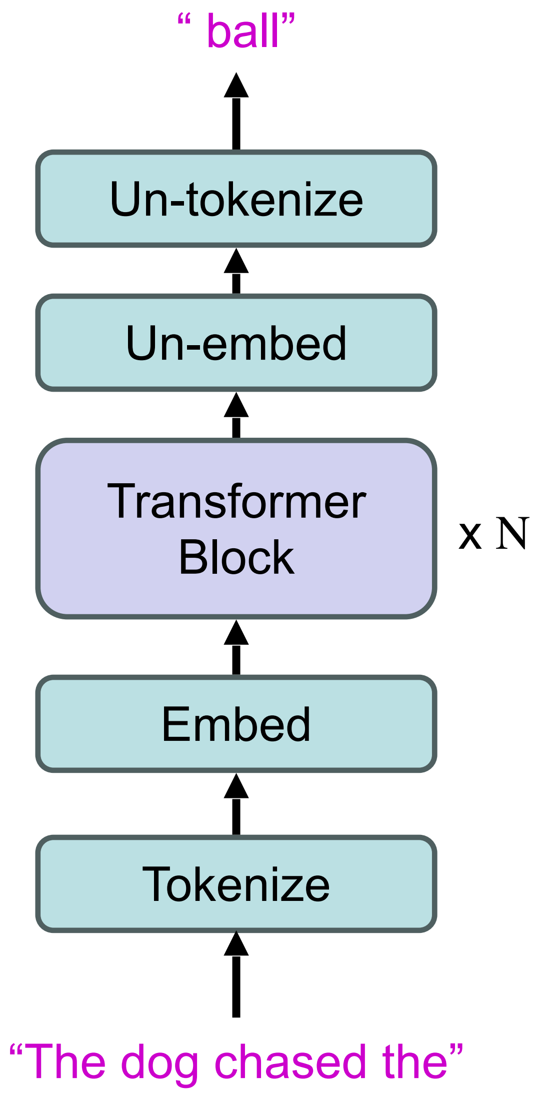


Transformer Architecture



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

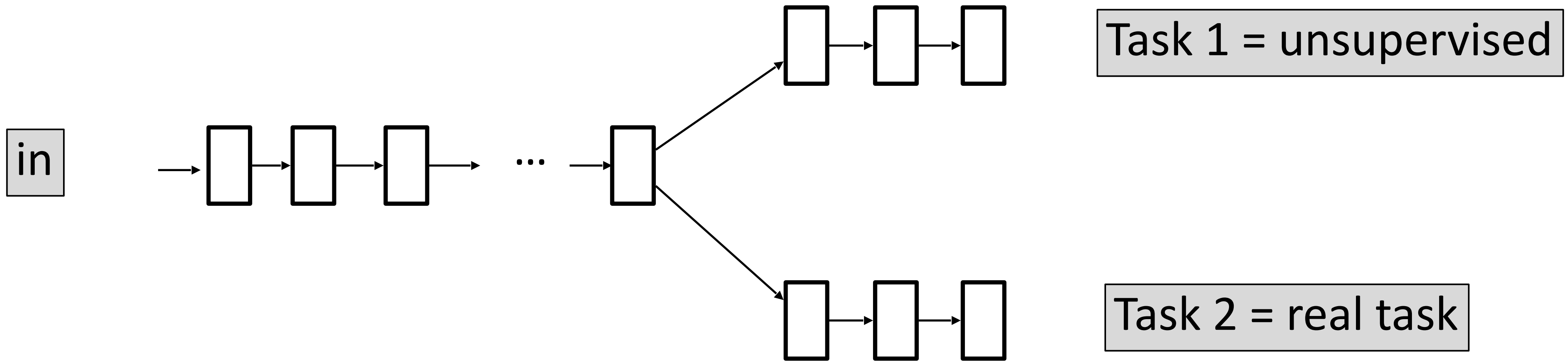
- Do we always need human supervision to learn features?
- Can't we learn general-purpose features?
- Key hypothesis:
  - IF neural network smart enough to predict:

Task 1

    - Next frame in video
    - Next word in sentence
    - Generate realistic images
    - ``Translate'' images
    - …
  - THEN same neural network is ready to do Supervised Learning from a very small data-set

Task 2

Transfer from Unsupervised Learning



Example Setting

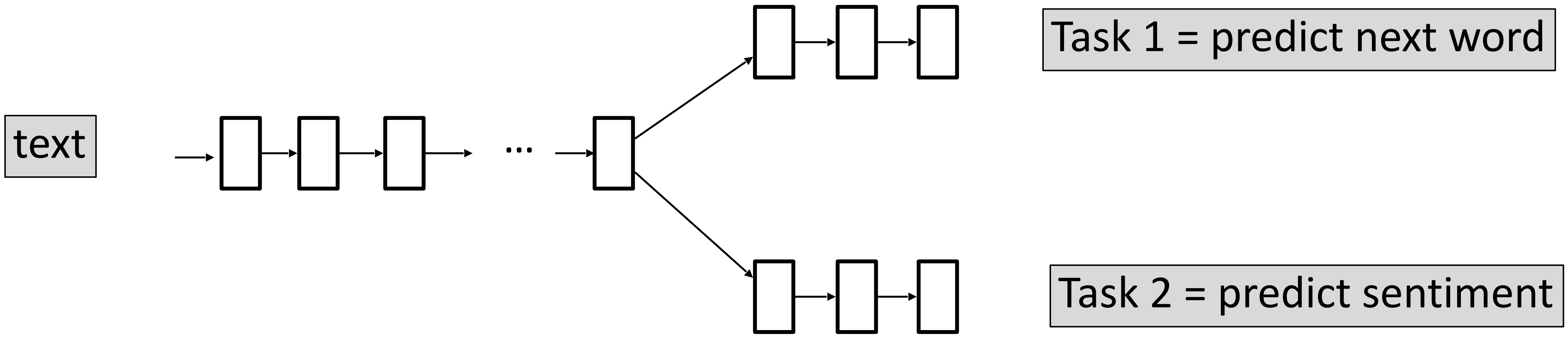


Image Pre-Training: Predict Missing Patch

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

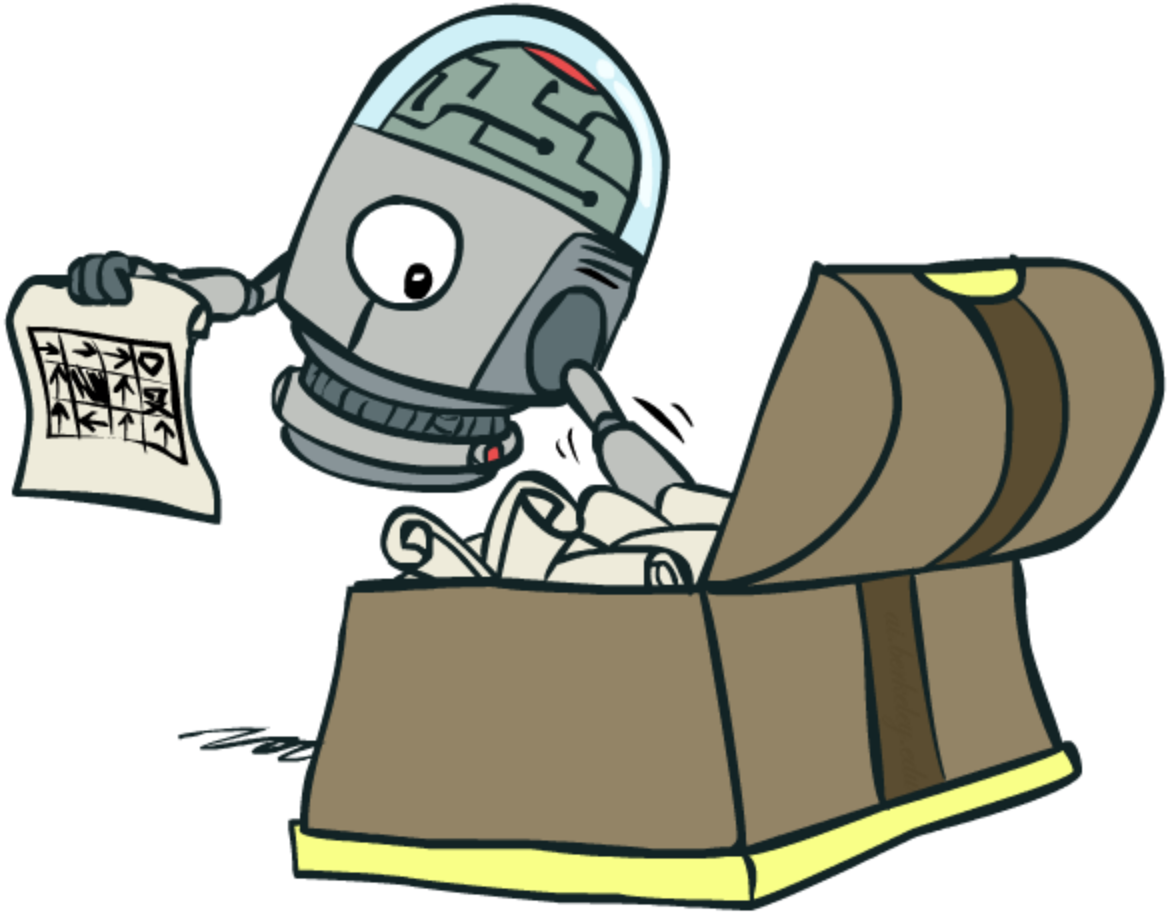
Reinforcement Learning from Human Feedback

- 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

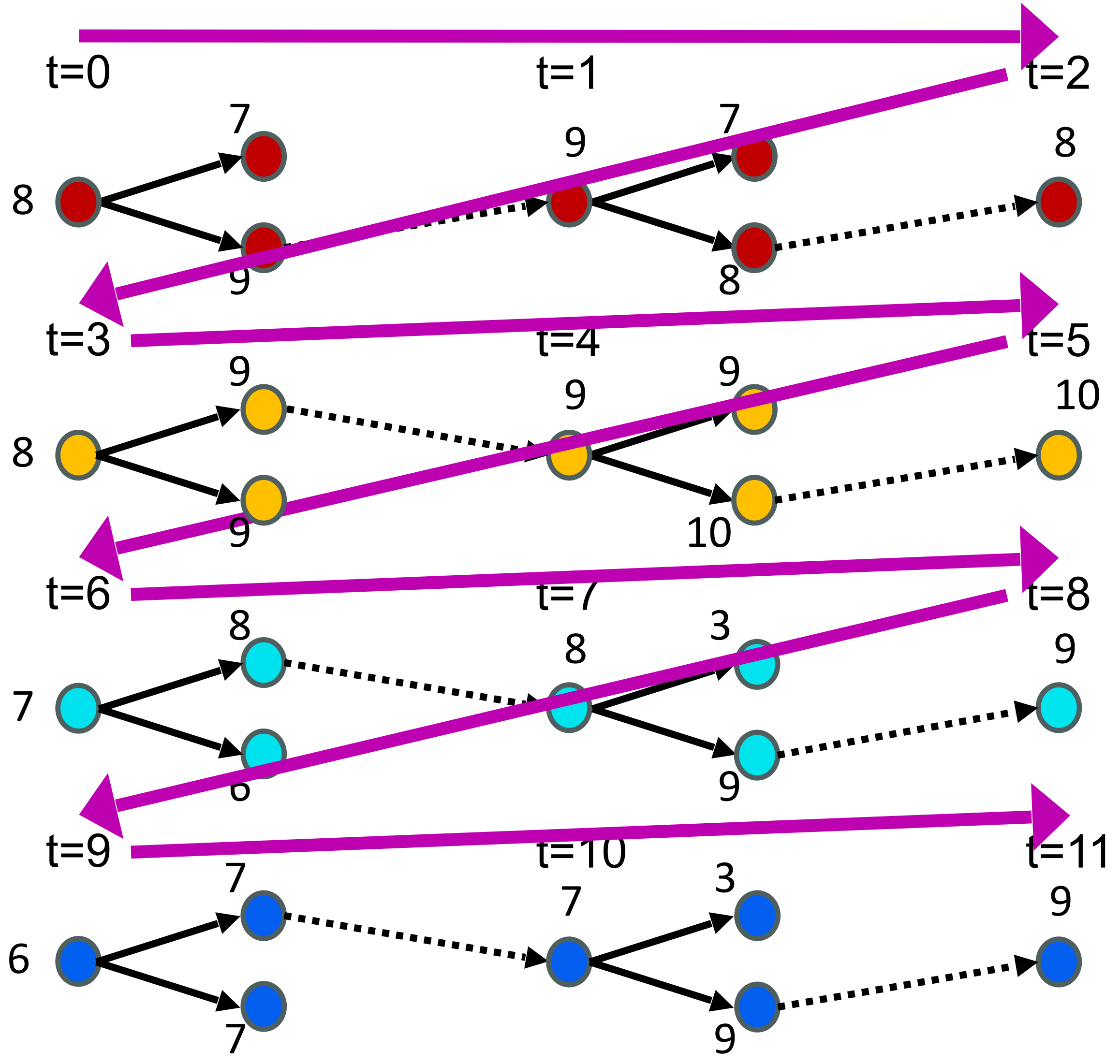
Estimating the Policy Gradient*

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

Large Language Models

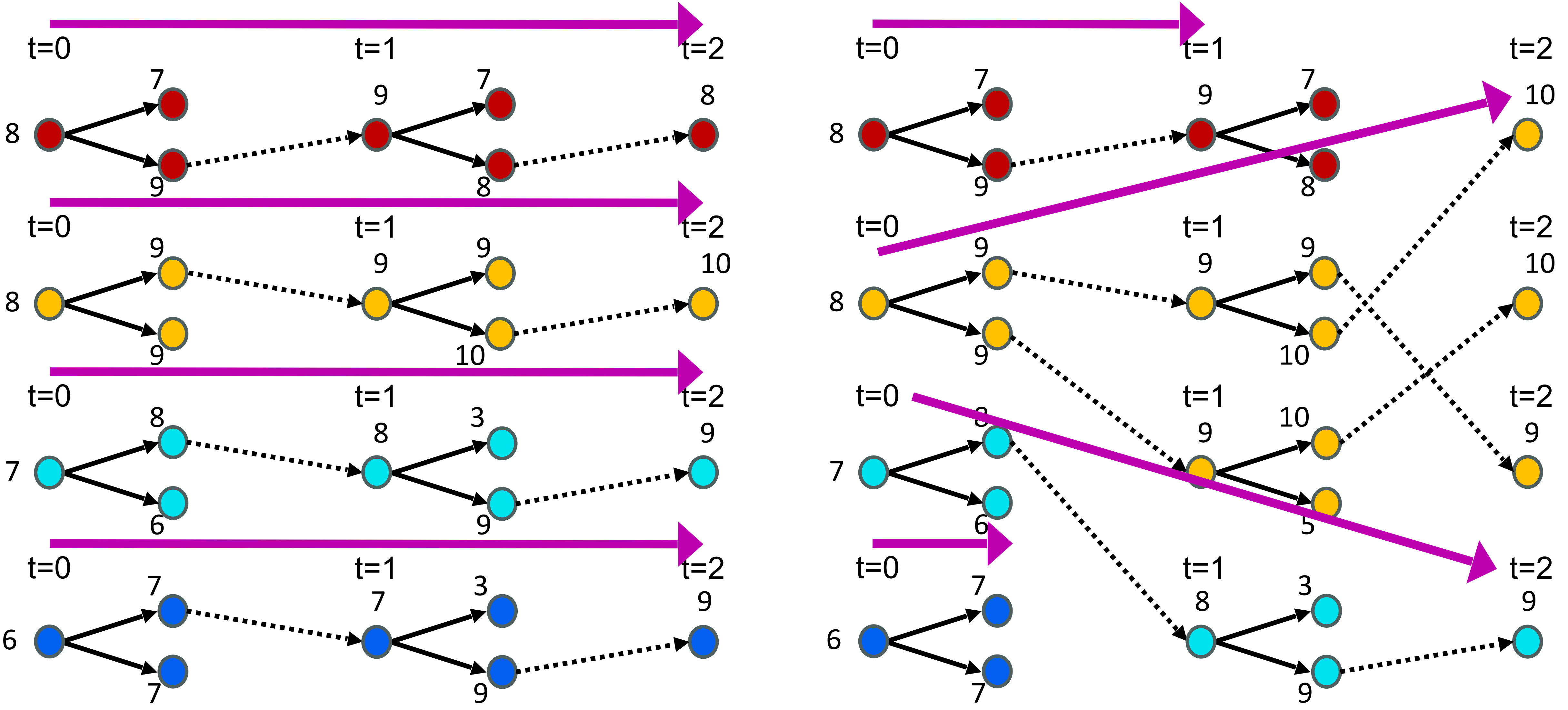
- ~~Feature engineering~~
  - ~~Text tokenization~~
  - ~~Word embeddings~~
- ~~Deep neural networks~~
  - ~~Autoregressive models~~
  - ~~Self-attention mechanisms~~
  - ~~Transformer architectures~~
- ~~Multi-class classification~~
- ~~Supervised learning~~
  - ~~Self-supervised learning~~
  - ~~Instruction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

Beam Search



Random restarts

Beam Search



Parallel search

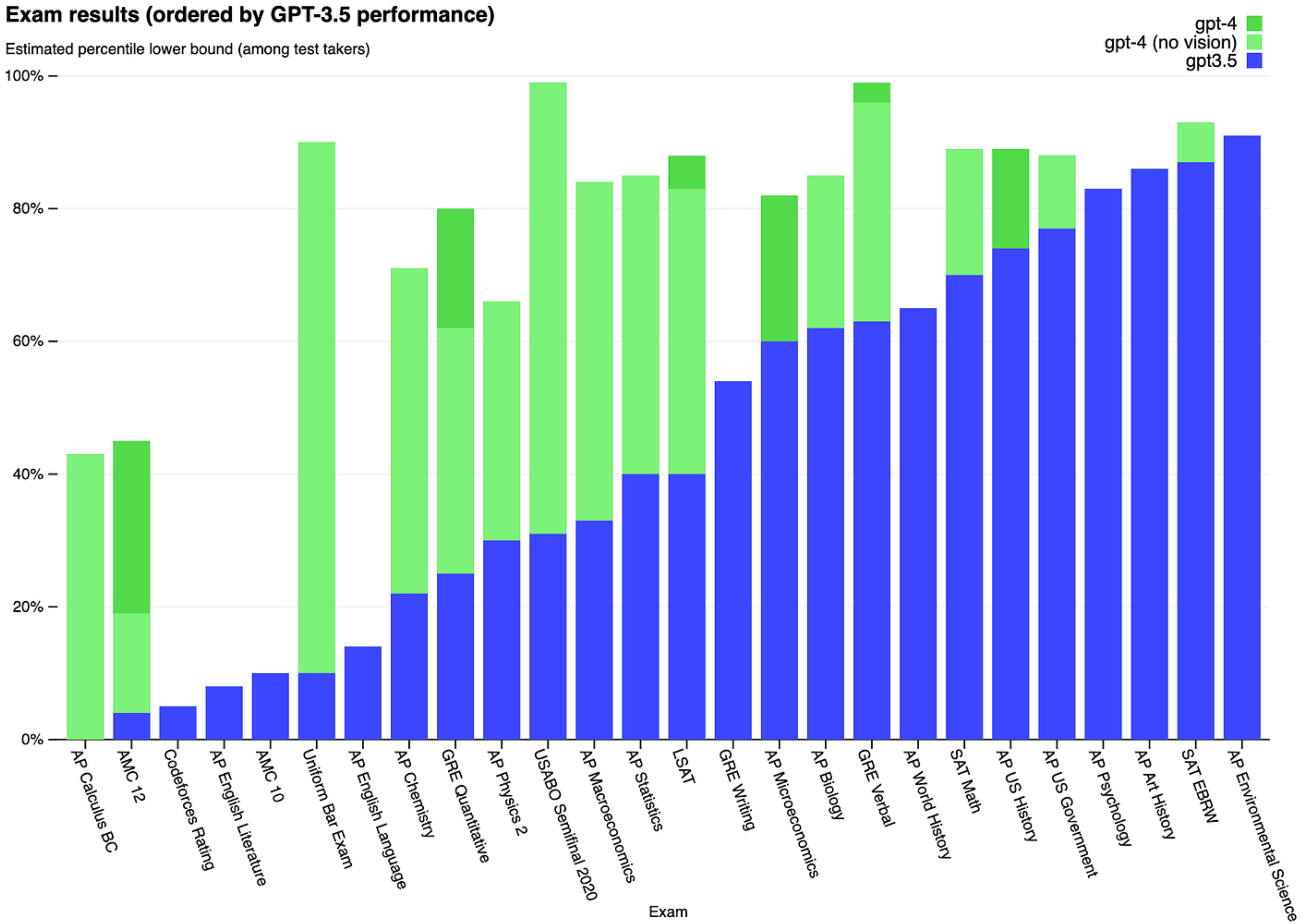
Beam search

Beam Search

ig.ft.com/generative-ai

Tracking Progress

- How well AI can do human tasks

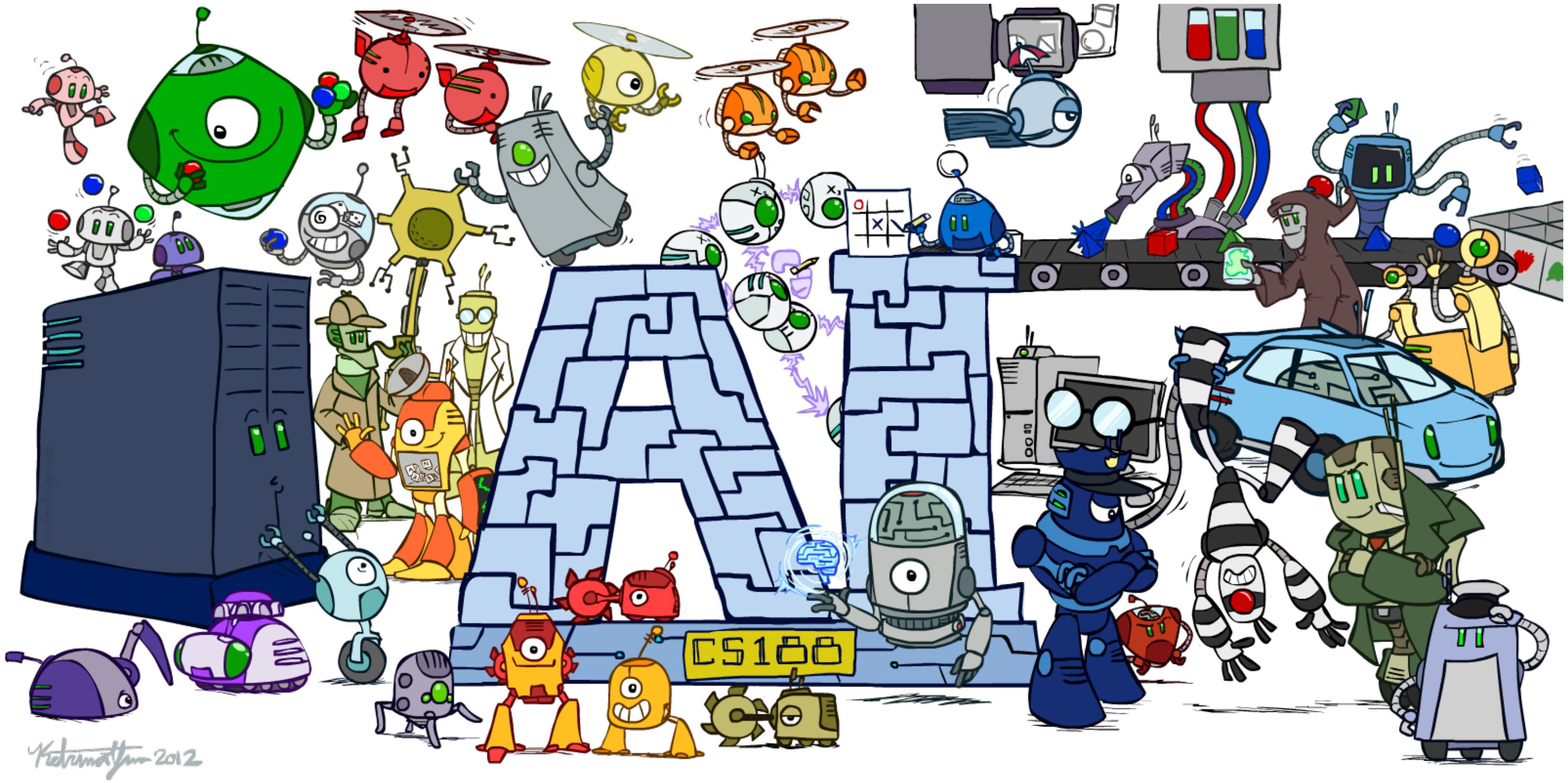


[OpenAI]

Where to go next?

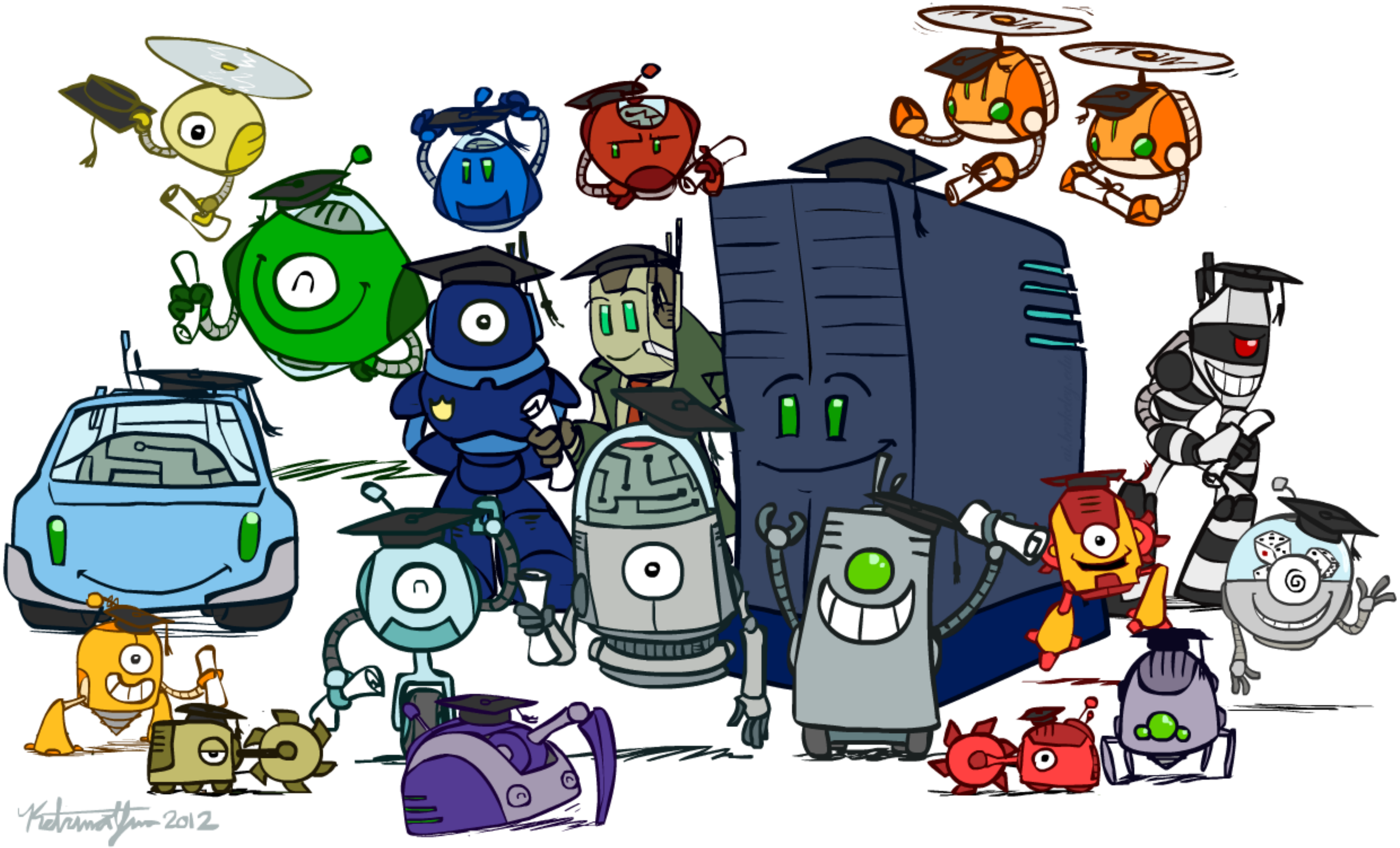
- Congratulations, you've seen the basics of modern AI
  - … and done some amazing work putting it to use!
- How to continue:
  - Machine learning: cs189, cs182, stat154, ind. eng. 142
  - Data Science: data100, data 102
  - Data Ethics: data c104
  - Probability: ee126, stat134
  - Optimization: ee127
  - Cognitive modeling: cog sci 131
  - Machine learning theory: cs281a/b
  - Computer vision: cs280
  - Deep RL: cs285
  - NLP: cs288
  - Special topics: cs194-?
  - … and more; ask if you're interested

Special Thanks



Ketrina Yim

CS188 Artist


Ketrinayim 2012