AI Training Workshop for Chemical Engineers

Alex Olson, CARTE

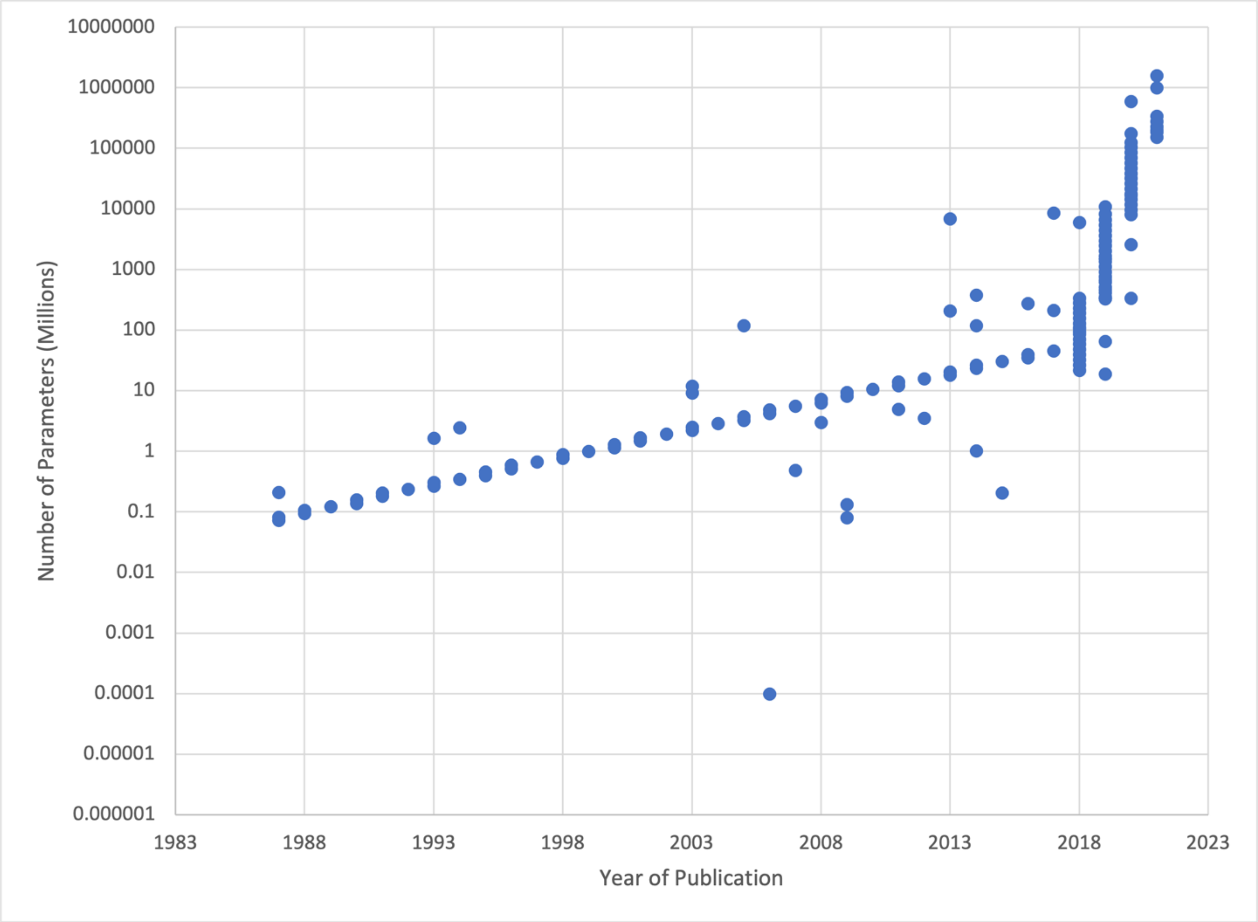
Language Models

- Estimates the likelihood of a sequence of words occurring
- To generate text, select the word most likely to appear next
- How do we estimate likelihood? By looking at lots of text
- Simple approach: look up the number of times a sequence occurs
- More sophisticated: Neural Networks

$$P(The, dog, and, the, cat) > P(The, dog, and, the, ostrich)$$

<u>Large</u> Language Models

- Latest models are capable of learning from much more data
- Both thanks to technological improvements, and a willingness to spend more money



UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
Centre for Analytics and Artificial Intelligence Engineering

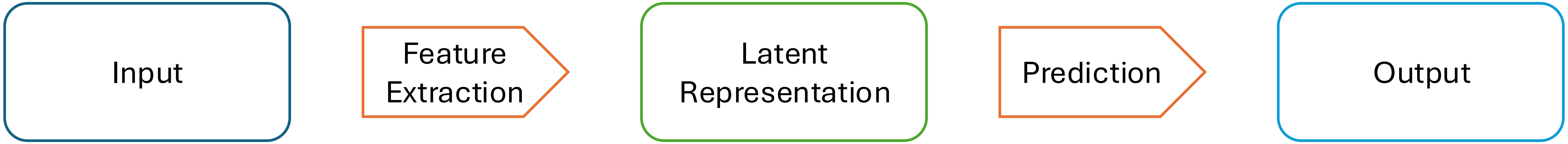
Challenges in NLP

- **Ambiguity:** Words can mean different things depending on context
- **Nuances:** Languages are full of idioms, slang, cultural references, sarcasm…
- **Syntax vs Semantics:** A grammatically correct sentence might not make sense, or a grammatically incorrect one might be easy to understand

- I saw a man on the hill with the telescope
- That's a cool cat
- Colourless green ideas sleep furiously
- Me went store

Deep Learning at a high level

- Modern deep learning models can be thought of as a two-part process:



- When we are *using* an AI model, this typically looks like one single application
- But we can build these two parts separately!

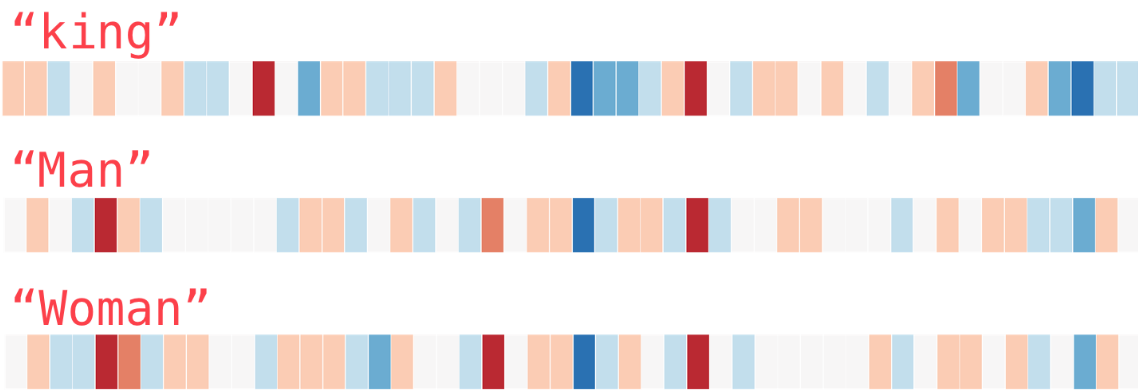
Deep Learning at a high level

- We will focus on turning *symbols* into *meaningful representations* first

- In order to do reasoning with the word "dog", we need to have an underlying concept of what that word means

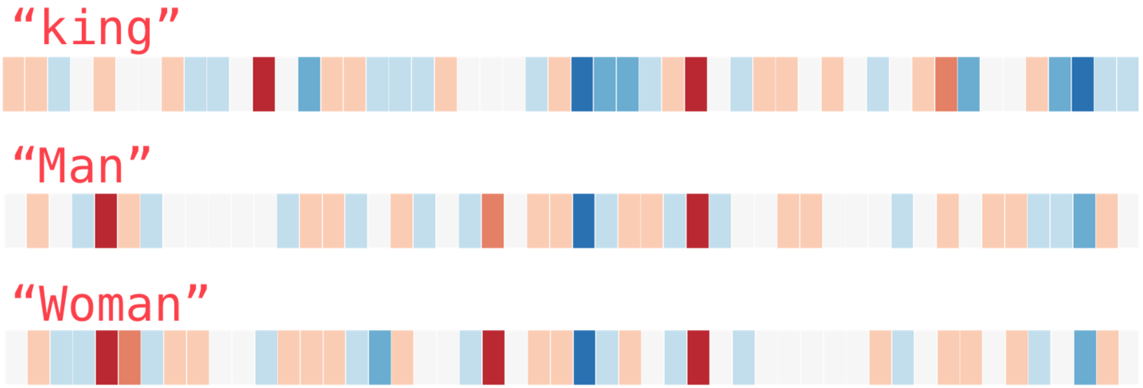
How does an LM “understand” word meaning?

- In order to predict the likelihood of a word, we must have some sense of its meaning
- Some words have similar meanings, and can easily fit in the same place
- Once we have a useful mapping from symbols to representations, it can be reused again and again
- Word2Vec: 300 features
- GPT-3: 12,888

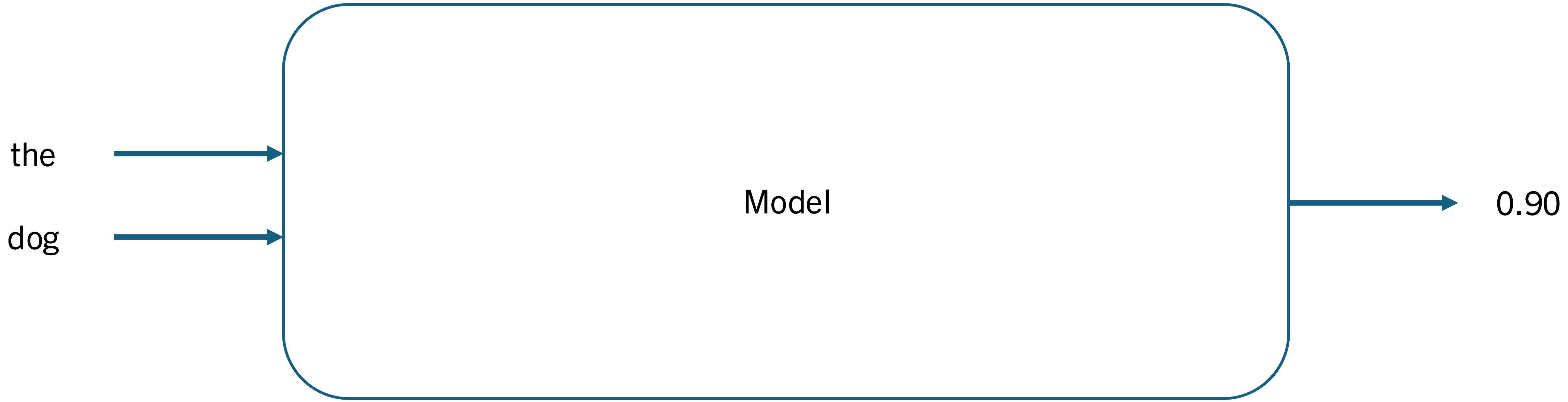


How does an LM “understand” word meaning?

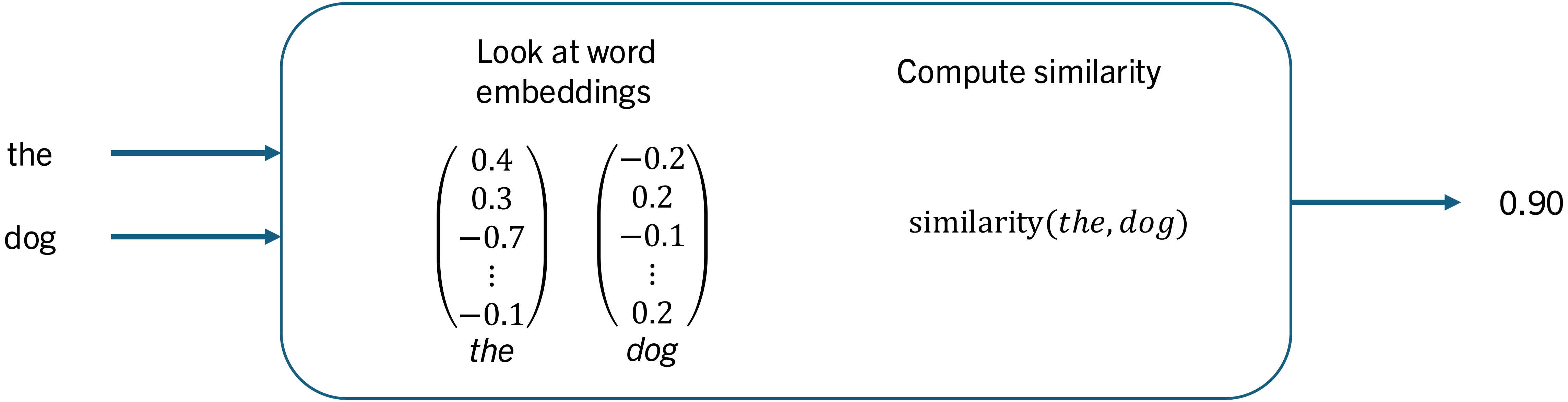
- Key concept of word embeddings: similar words should have similar vectors
- How do we accomplish this?



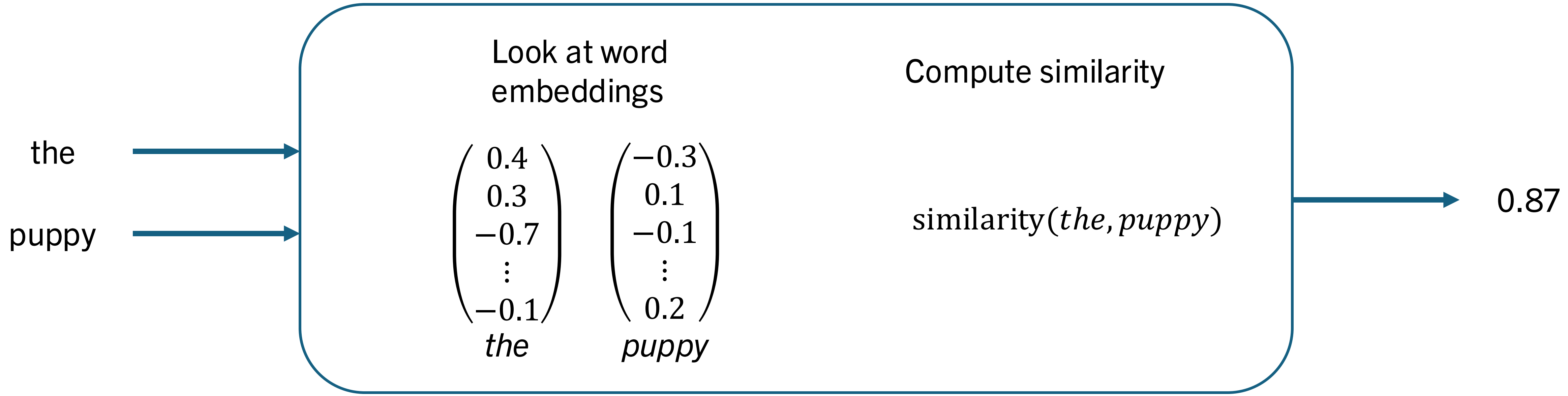
Building Word Embeddings



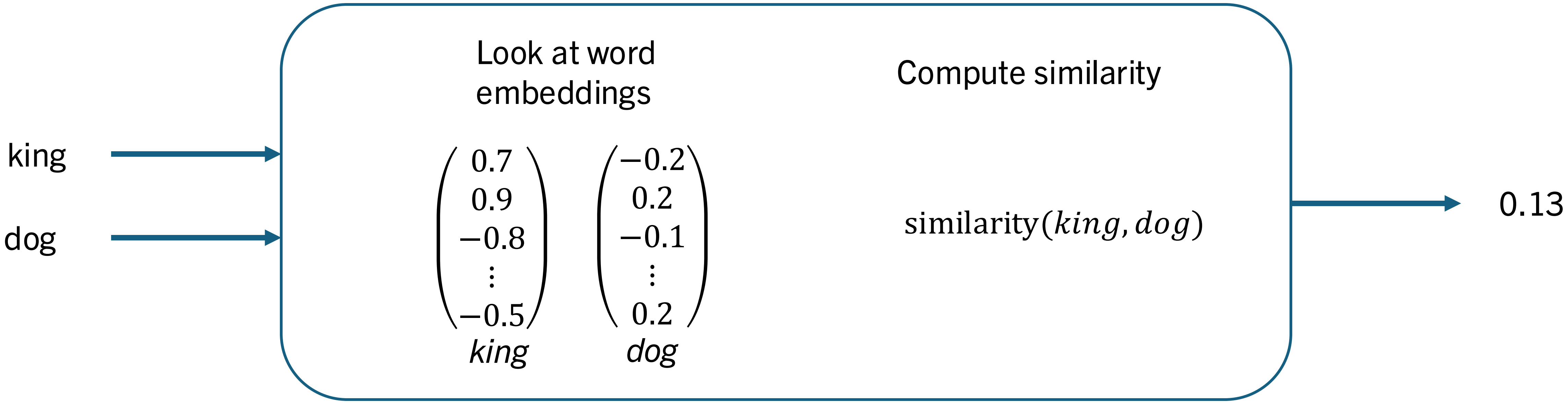
Building Word Embeddings



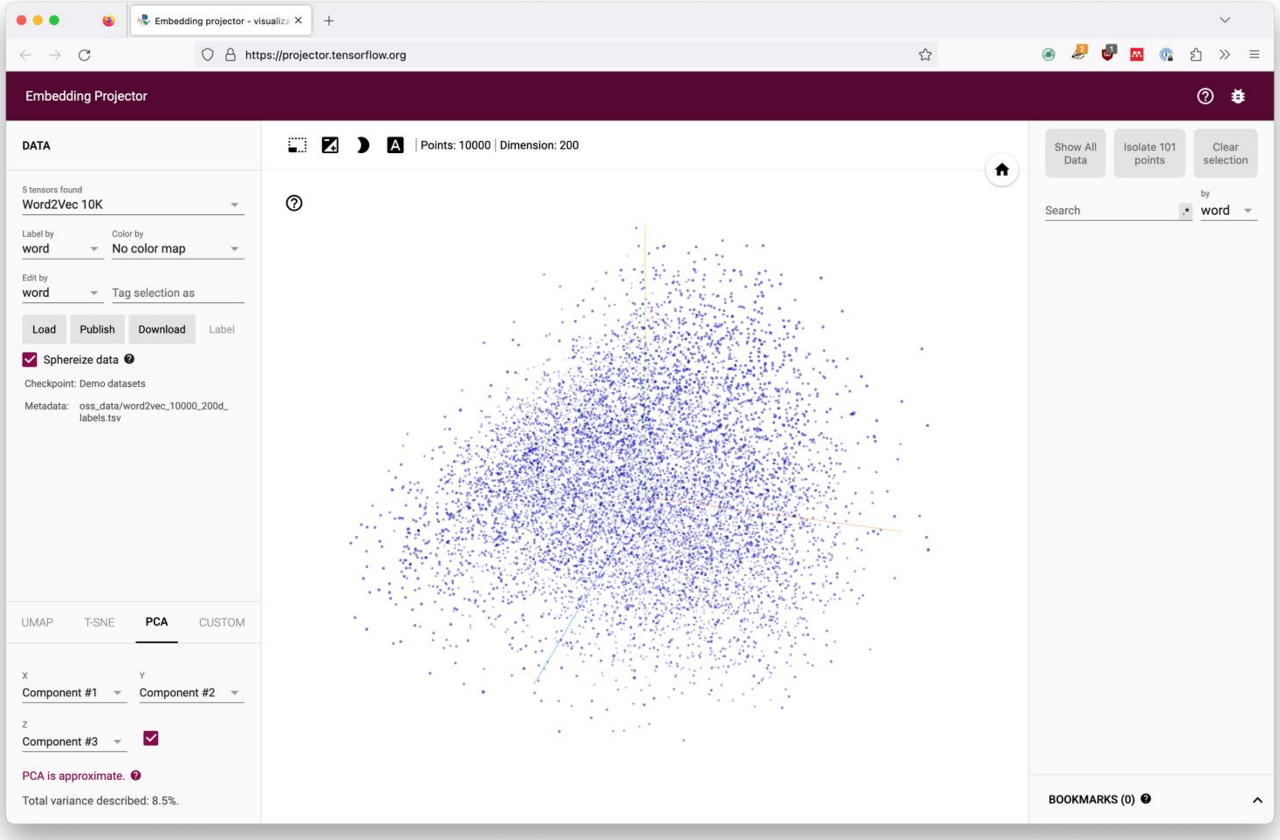
Building Word Embeddings



Building Word Embeddings

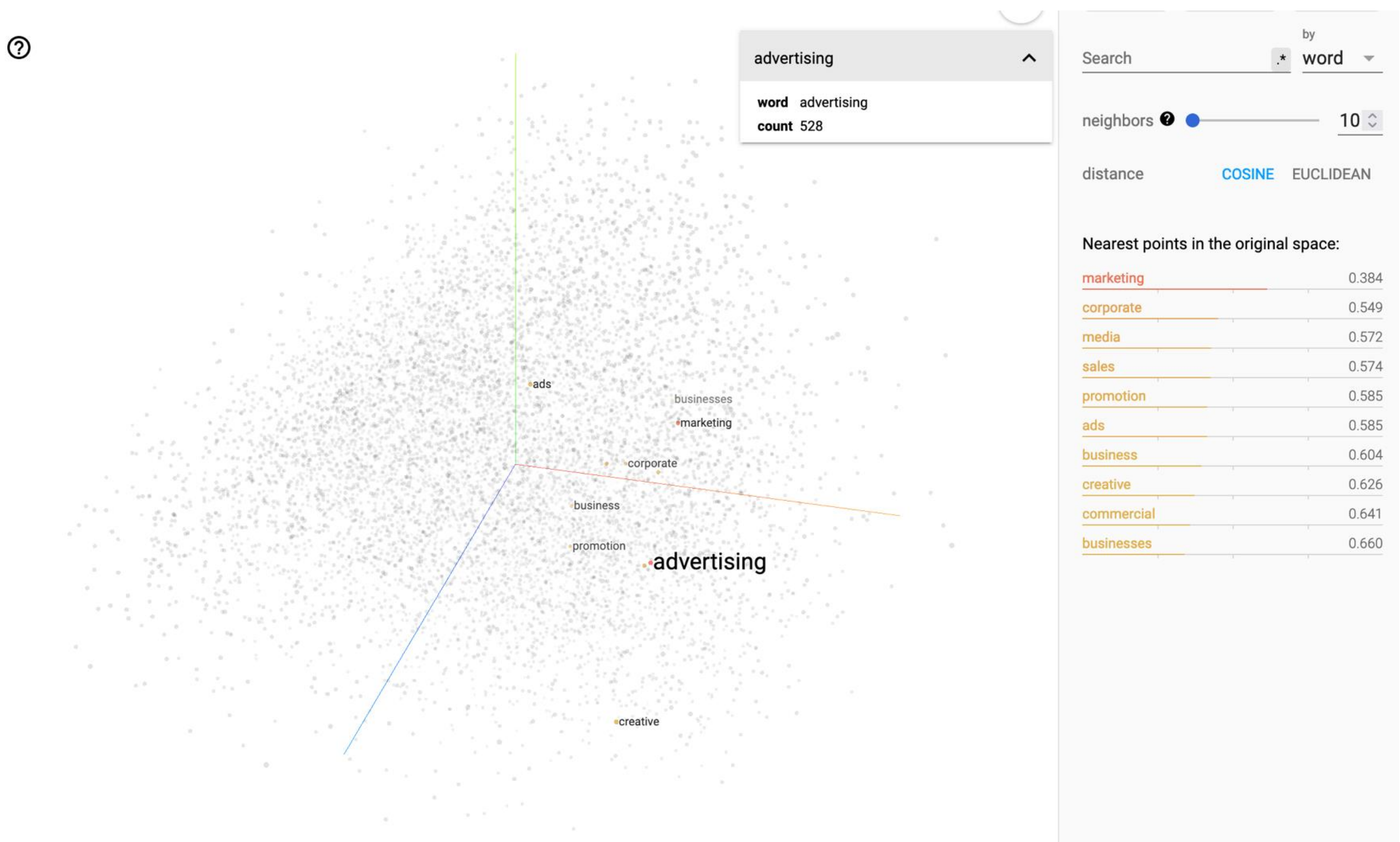


Results



Embedding projector - visualiz
https://projector.tensorflow.org
Embedding Projector
DATA
5 tensors found
Word2Vec 10K
Label by
word
Color by
No color map
Edit by
word
Tag selection as
Load
Publish
Download
Label
Sphereize data
Checkpoint: Demo datasets
Metadata: oss_data/word2vec_10000_200d_labels.tsv
UMAP
T-SNE
PCA
CUSTOM
X
Component #1
Y
Component #2
Z
Component #3
PCA is approximate.
Total variance described: 8.5%.
Points: 10000 | Dimension: 200
Show All Data
Isolate 101 points
Clear selection
Search
by
word
BOOKMARKS (0)

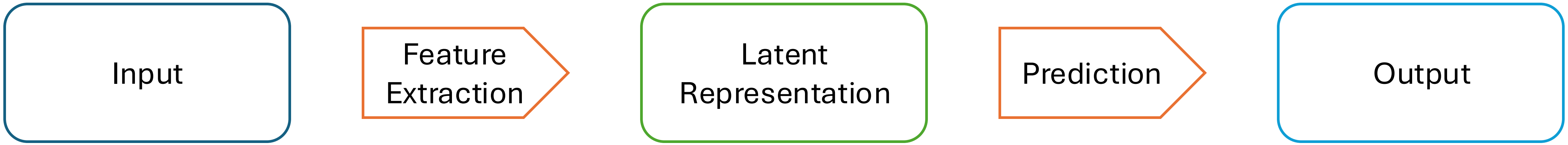
Results



advertising
word advertising
count 528
ads
businesses
marketing
corporate
business
promotion
advertising
creative
Search
by
word
neighbors
10
distance
COSINE
EUCLIDEAN
Nearest points in the original space:
marketing 0.384
corporate 0.549
media 0.572
sales 0.574
promotion 0.585
ads 0.585
business 0.604
creative 0.626
commercial 0.641
businesses 0.660

Deep Learning at a high level

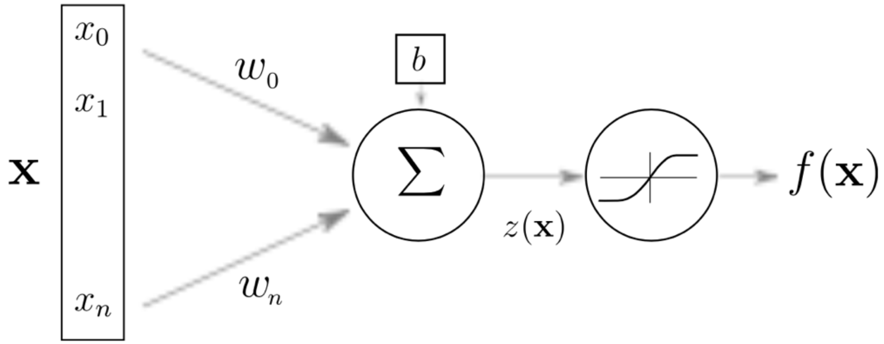
- Now that we have a method to *extract features* from our symbols, we can use those representations to predict



- In many cases, feature extraction is the hard part, and prediction is comparatively easy (e.g. many vision problems)
- This is not really true for language, however

Building GPT

Artificial Neuron



$$z(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

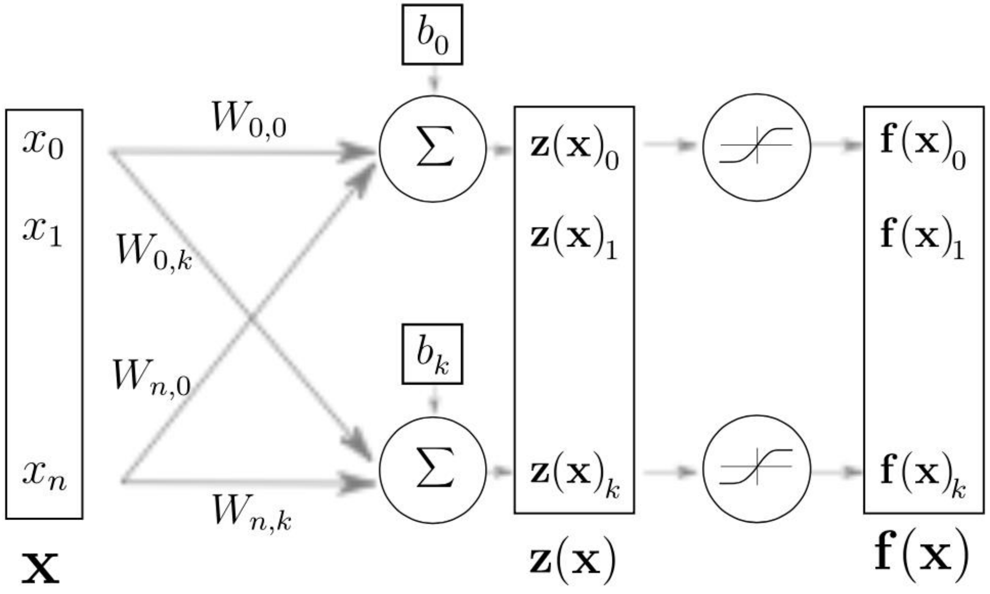
$$f(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b)$$

- $\mathbf{x}, f(\mathbf{x})$ input and output
- $z(\mathbf{x})$ pre-activation
- $\mathbf{w}, b$ weights and bias
- g activation function

Concrete Example

- Say we have two input dimensions x_1 and x_2 and one output dimension $f(x)$ (sometimes, $\hat{y}$ - the predicted value of y - is used instead of $f(x)$)
- Our weights and biases could be $\mathbf{W} = [3, -2]$ and $b = 1$
- Our non-linearity could be ReLU: $g(z) = \max(0, z)$
- Now $z(x) = 3x_1 - 2x_2 + 1$ and $f(x) = \max(0, 3x_1 - 2x_2 + 1)$
- *Every neuron* in a neural network is a function like this!

Layer of Neurons (Vectorization)

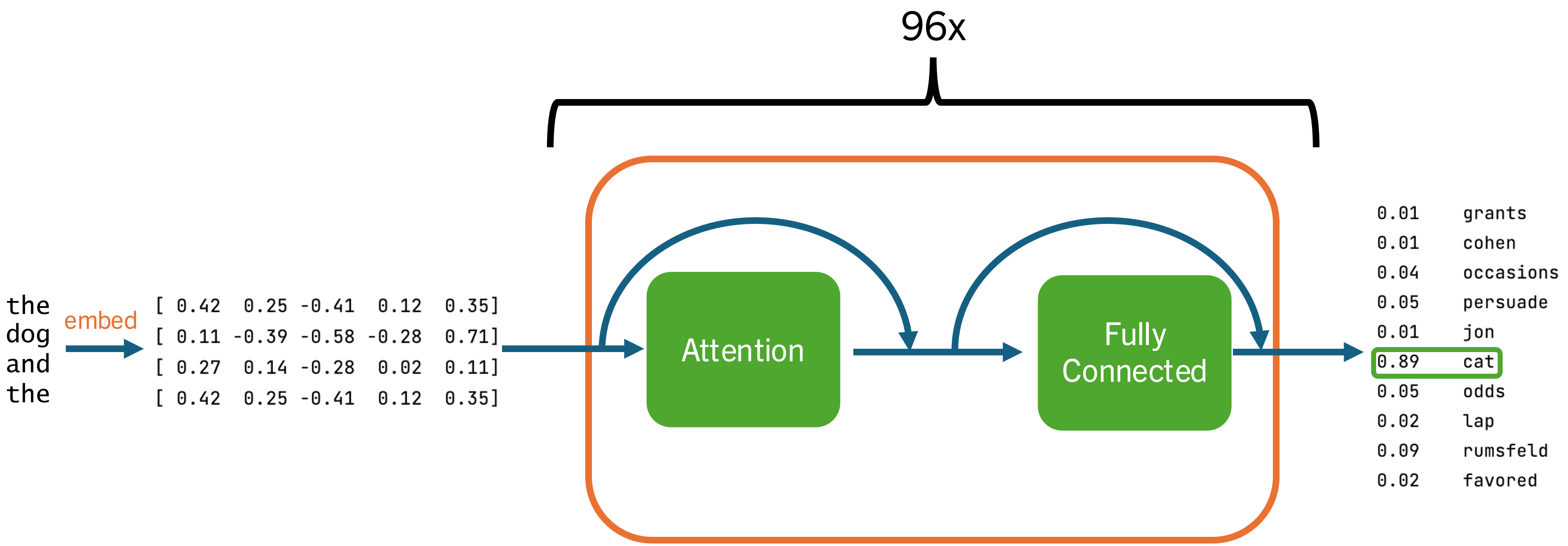


$$\mathbf{f}(\mathbf{x}) = g(\mathbf{z}(\mathbf{x})) = g(\mathbf{W}\mathbf{x} + \mathbf{b})$$

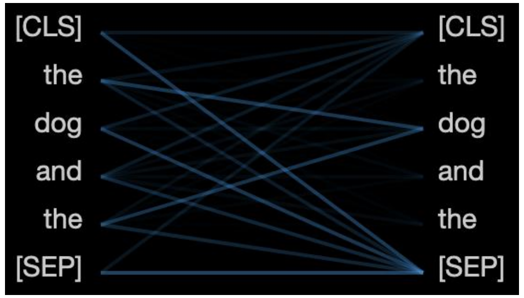
- $\mathbf{W}, \mathbf{b}$ now matrix and vector

Building GPT

Building GPT: The Transfomer



Building GPT: Attention



the	0.42	1.25	-0.41	1.12	0.35
dog	0.95	0.15	-0.55	0.72	0.71
and	1.18	-0.28	-0.23	1.02	0.11
the	0.56	-0.74	-0.33	1.12	0.35

w_Q → *Query*

```
[0.98 0.18 0.11]
[0.7  0.29 0.72]
[0.42 0.53 0.95]
[0.58 0.06 0.66]
```

w_K → *Key*

```
[0.26 0.31 0.22]
[0.81 0.93 0.47]
[0.45 0.49 0.36]
[0.3  0.7  0.79]
```

w_V → *Value*

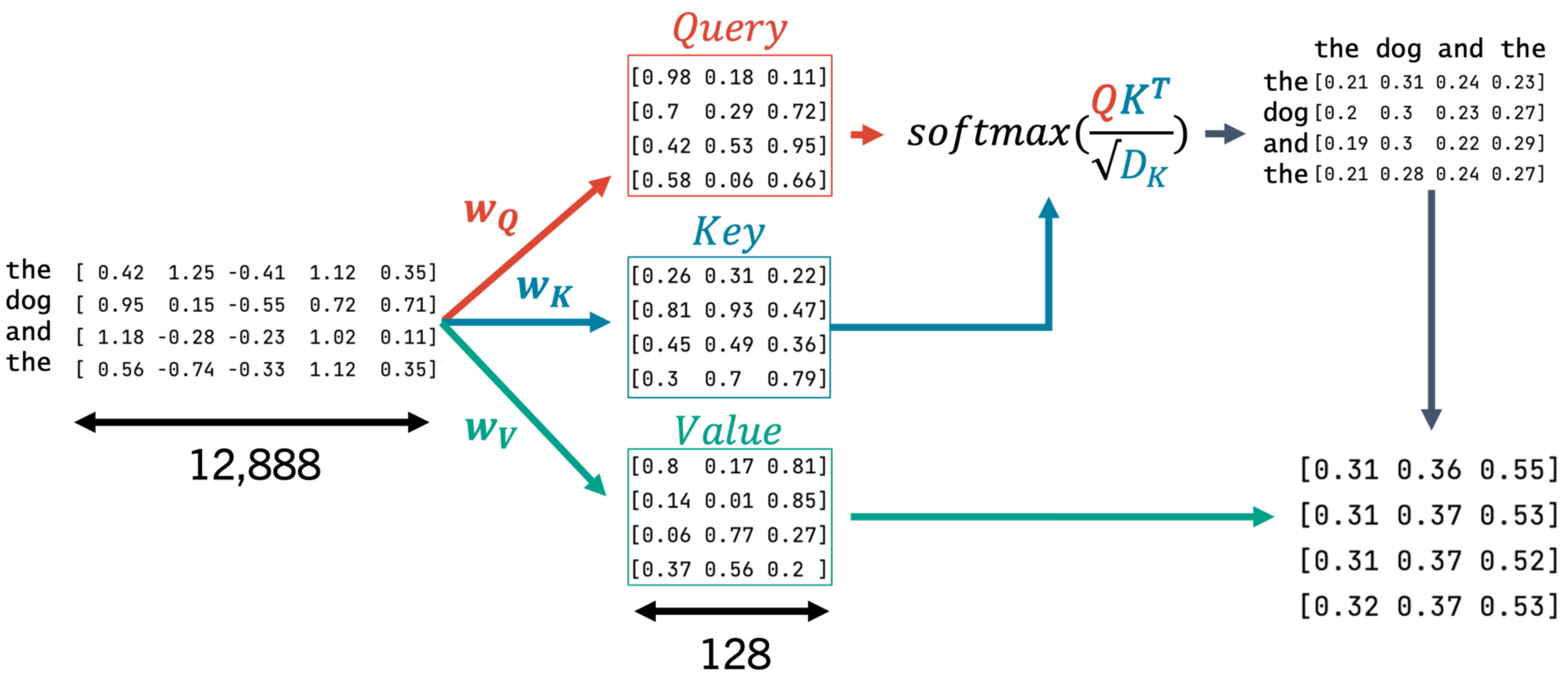
```
[0.8  0.17 0.81]
[0.14 0.01 0.85]
[0.06 0.77 0.27]
[0.37 0.56 0.2 ]
```

$$softmax\left(\frac{QK^T}{\sqrt{D_K}}\right)$$

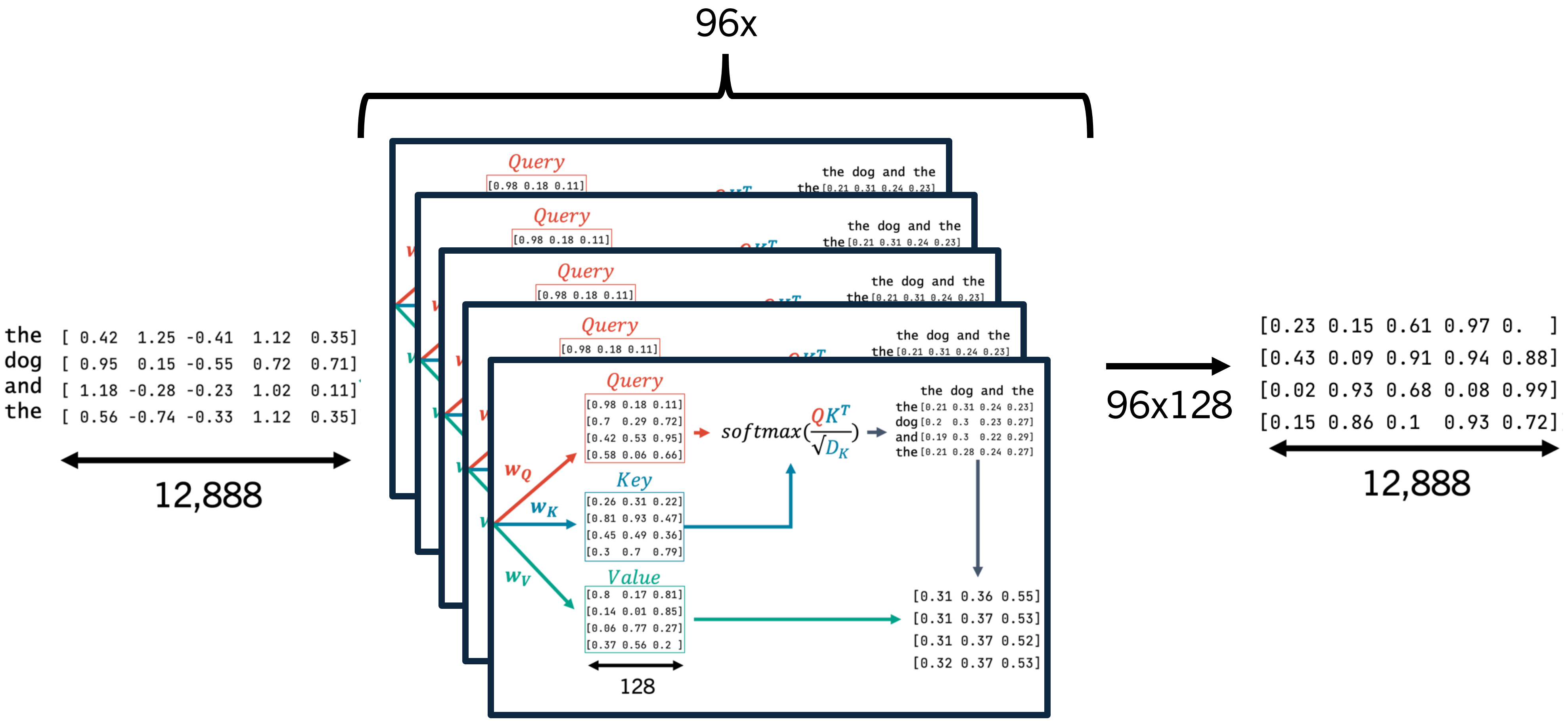
	the	dog	and	the
the	0.21	0.31	0.24	0.23
dog	0.2	0.3	0.23	0.27
and	0.19	0.3	0.22	0.29
the	0.21	0.28	0.24	0.27

```
[0.31 0.36 0.55]
[0.31 0.37 0.53]
[0.31 0.37 0.52]
[0.32 0.37 0.53]
```

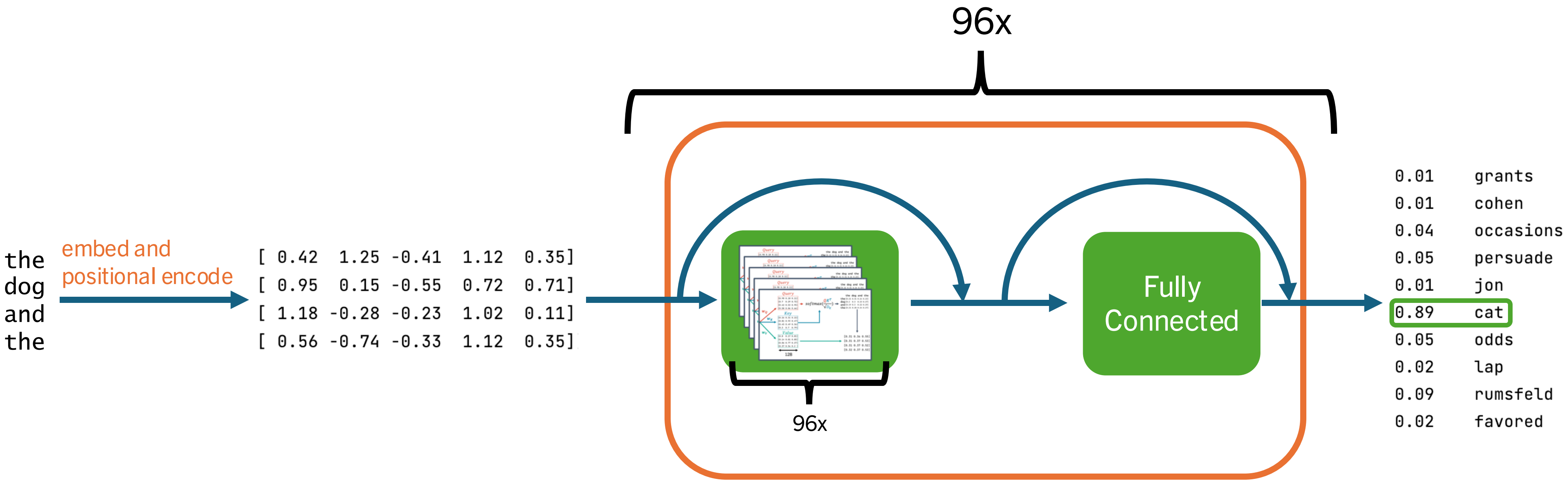
Building GPT: Attention



Building GPT: Attention



Building GPT



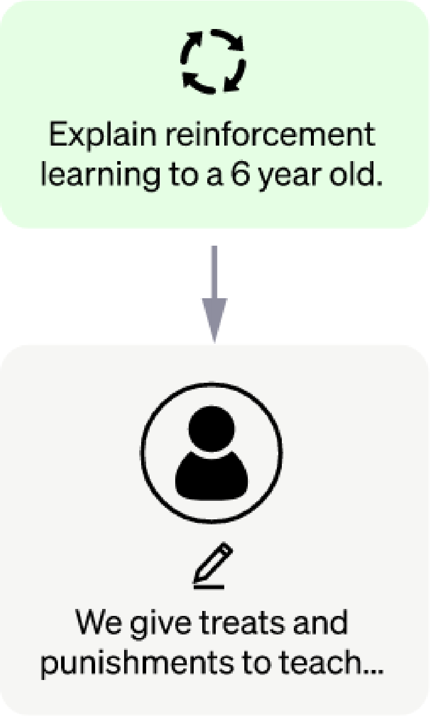
GPT's Training Data

- 1 token ≈ ¾ word
- Some datasets are sampled more times than others
- Common Crawl: billions of webpages collected over 7 years
- Webtext2: Dataset of webpages that have been shared on Reddit
- Books1: Free ebooks (?)
- Books2: Secret!
- English Wikipedia

Dataset	Quantity (tokens)	Weight in training mix

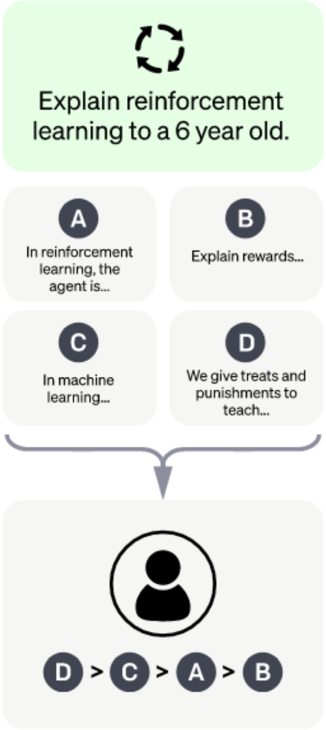
The training innovation of ChatGPT

Human annotators write answers to questions



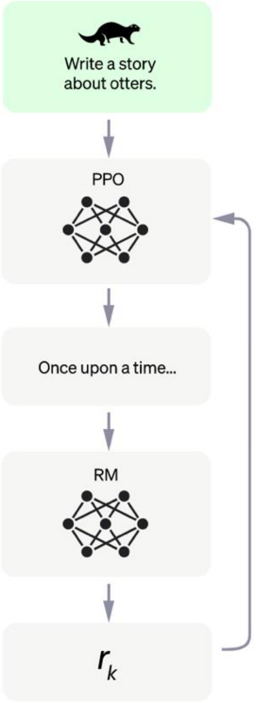
The generalist GPT model is taught from these Q&A pairs

Human annotators write more answers, and someone else ranks them



A separate model learns to rate the quality of an answer

GPT writes answers to sampled questions

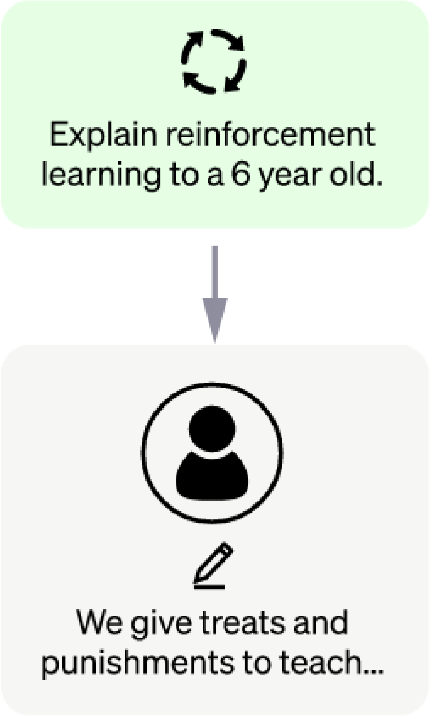


The reward model rates each answer, allowing GPT to keep learning

Figures: https://openai.com/blog/chatgpt/

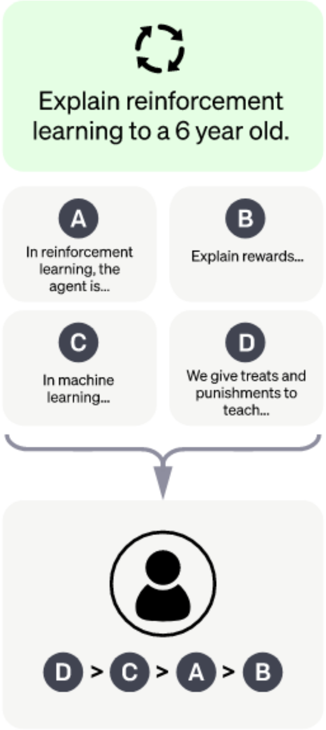
The training innovation of ChatGPT

Human annotators write answers to questions



The generalist GPT model is taught from these Q&A pairs

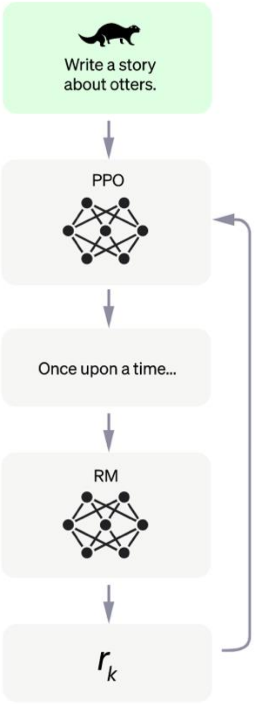
Human annotators write <u>more</u> answers, and someone else ranks them



A <u>separate</u> model learns to rate the quality of an answer

No more humans involved!

GPT writes answers to sampled questions

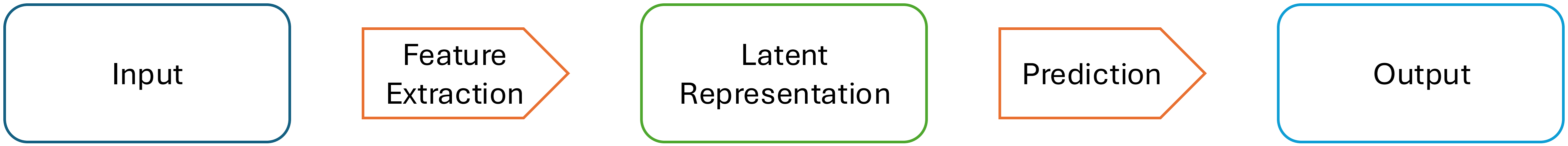


The reward model rates each answer, allowing GPT to keep learning

Figures: https://openai.com/blog/chatgpt/

Fine-Tuning a model

- Because of our modular approach to prediction, we can swap out a prediction task while using the same feature extraction:



- This is called *fine-tuning*, and has become a major factor in applications of deep learning (since training an LLM from scratch is expensive and time consuming)