Deep Learning: Natural Language Processing with Deep Learning

```
$ echo "Data Sciences Institute"
```

Outline

- Building to ChatGPT piece-by-piece:
- Attention mechanism and the Transformer architecture
- From "attention is all you need" to large language models (LLMs)
- Turning general LLMs into useful applications: RLHF and model fine-tuning
- Frontier topics: agentic systems, multimodal models, and more

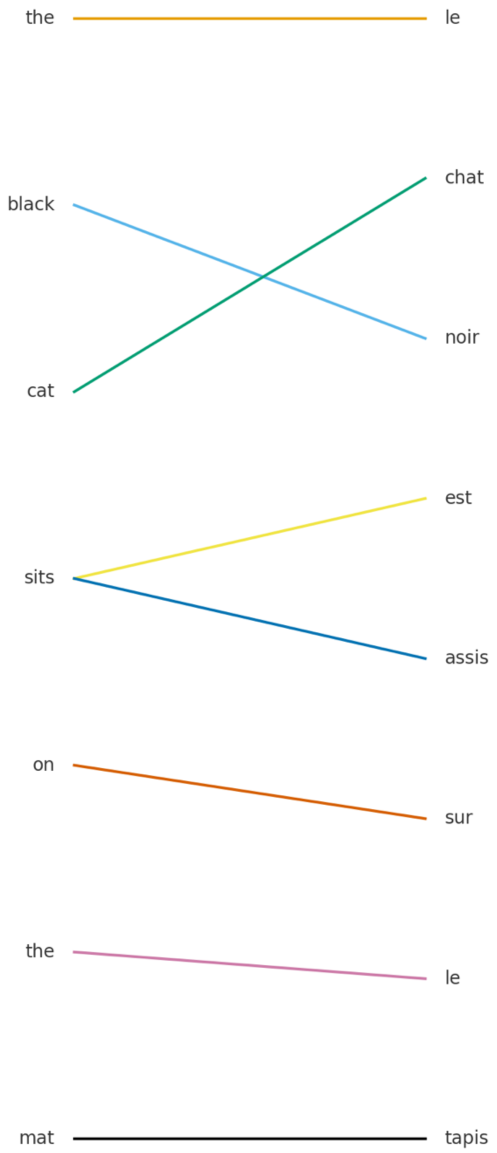
The Attention Mechanism

Sequential Models: a false start?

- Recurrent Neural Networks (RNNs) were the first widely used Deep Learning models for NLP
- RNNs can in theory capture long-term dependencies in sequences
- In practice, RNNs are difficult to train on long sequences due to vanishing/exploding gradients
- LSTMs and GRUs were introduced to mitigate these issues, but still have limitations
- RNNs are inherently sequential, making them slow to train and difficult to parallelize

Machine Translation

- Given a sentence in a source language, produce its translation in a target language
- Early approaches used RNNs with an encoder-decoder architecture
- Key idea: encode the source into an internal representation, then decode it into the target language
- However, RNNs struggled with long sentences and complex structures



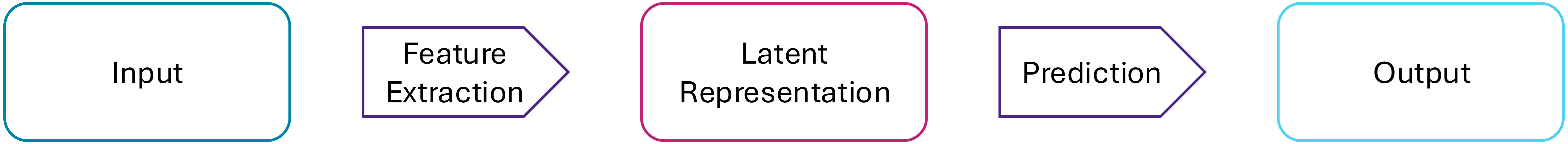
Language Models

- 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

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

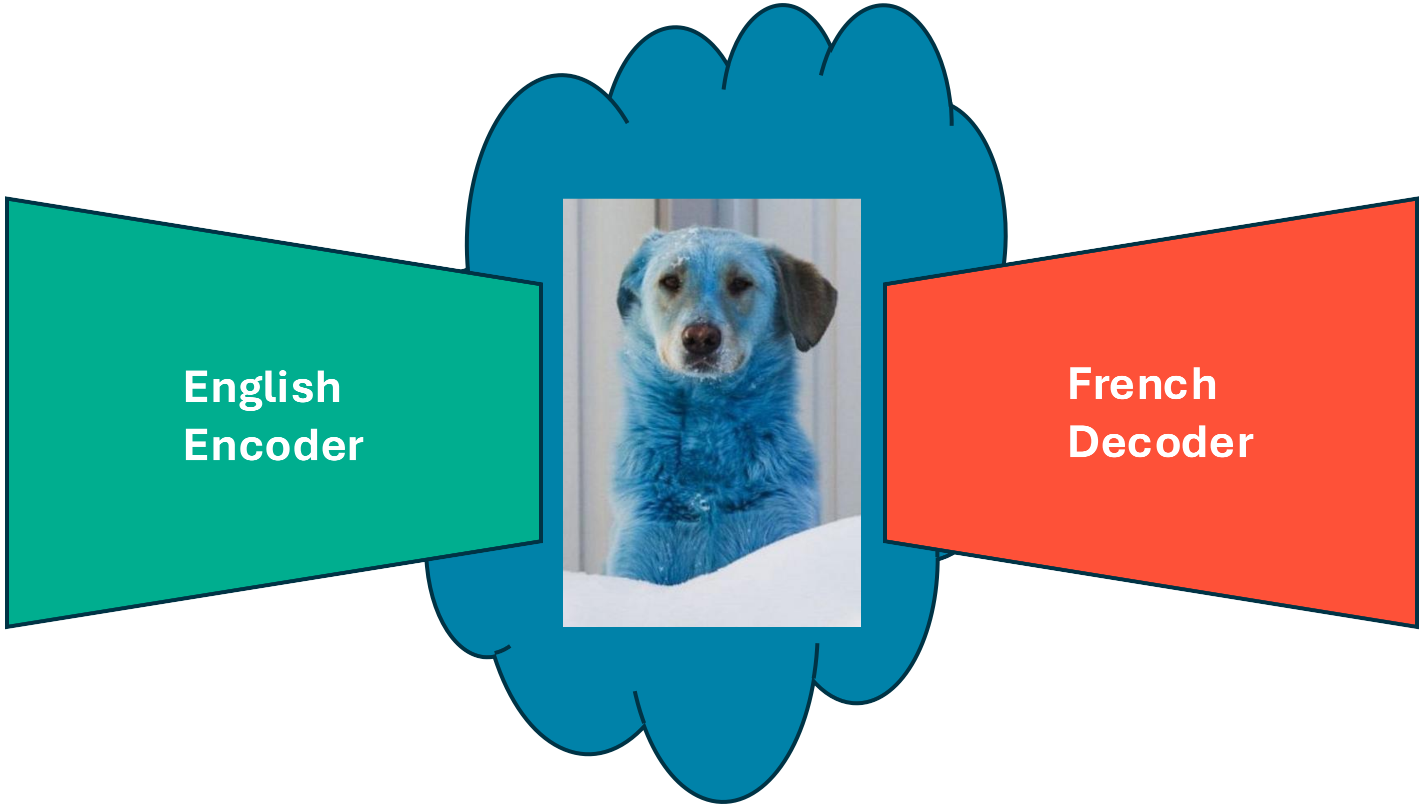
Deep Learning at a high level

- 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

Encoder-Decoder Networks



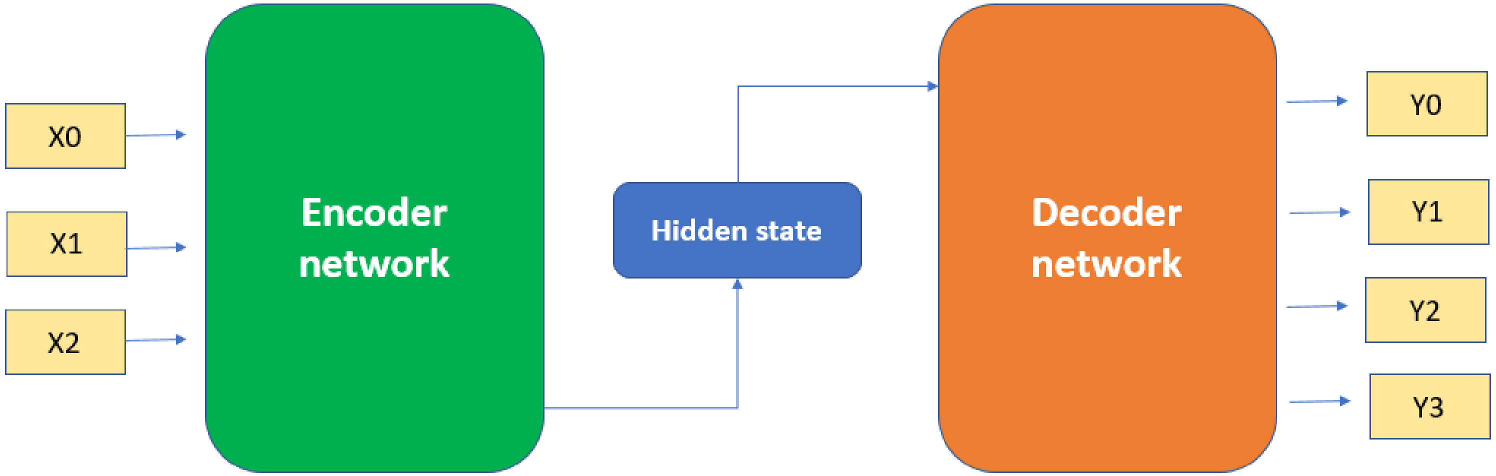
“the blue dog”

English Encoder

French Decoder

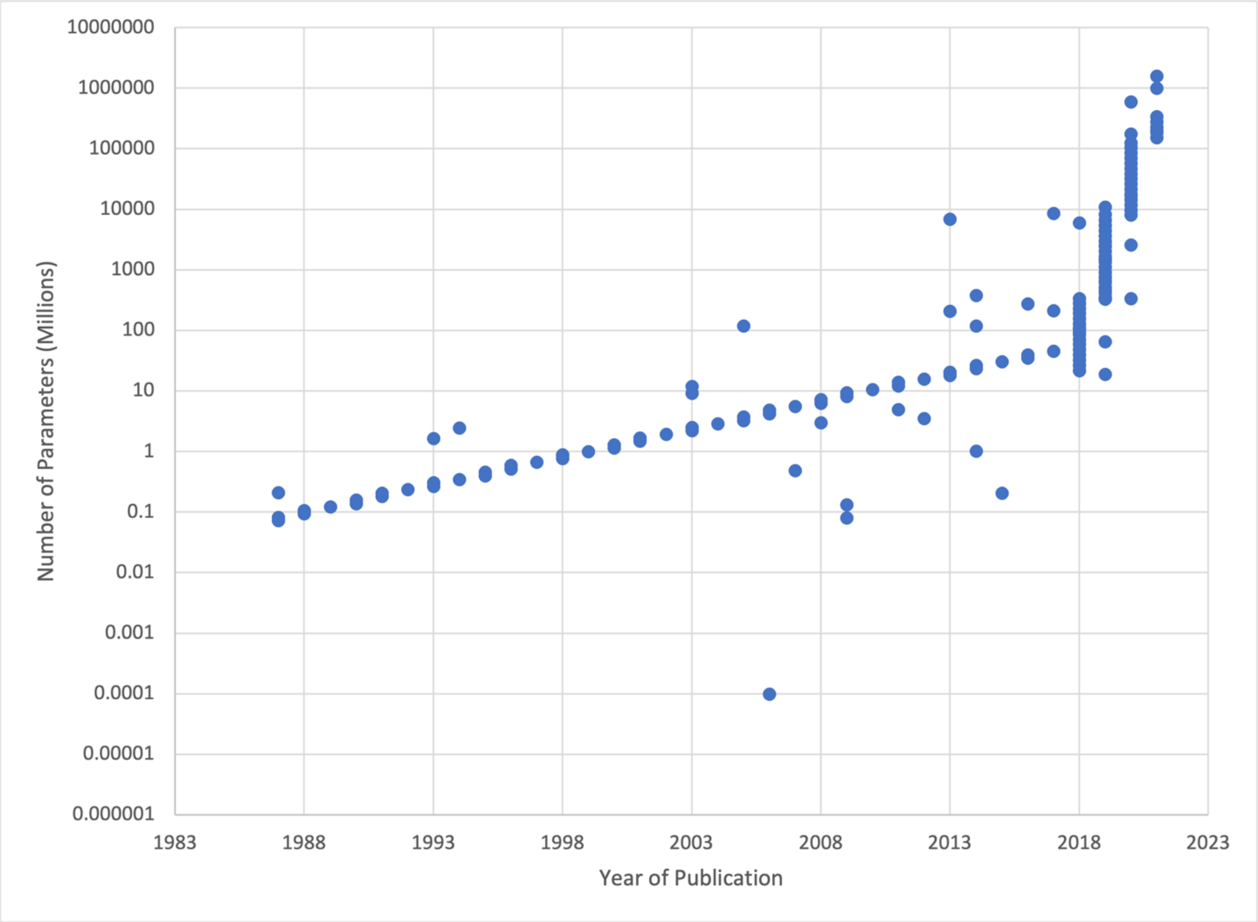
“le chien bleu”

Encoder-Decoder Networks



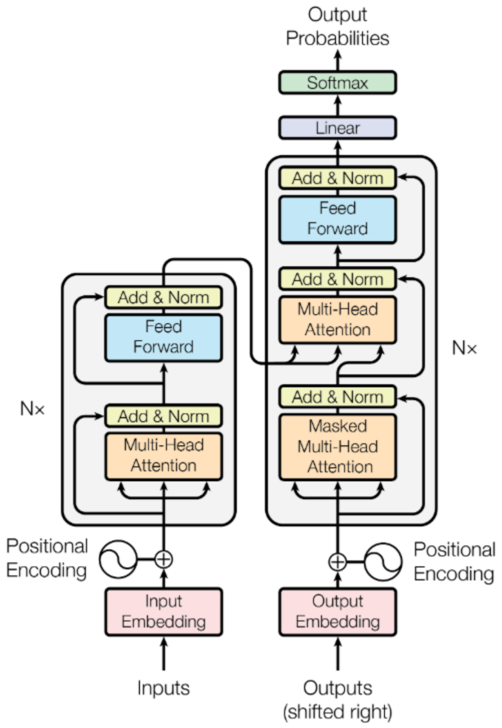
Large Language Models

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

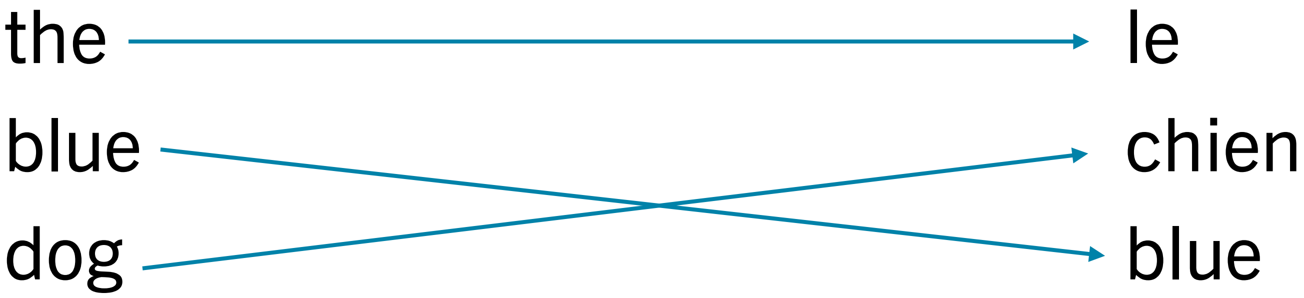


Attention is All You Need

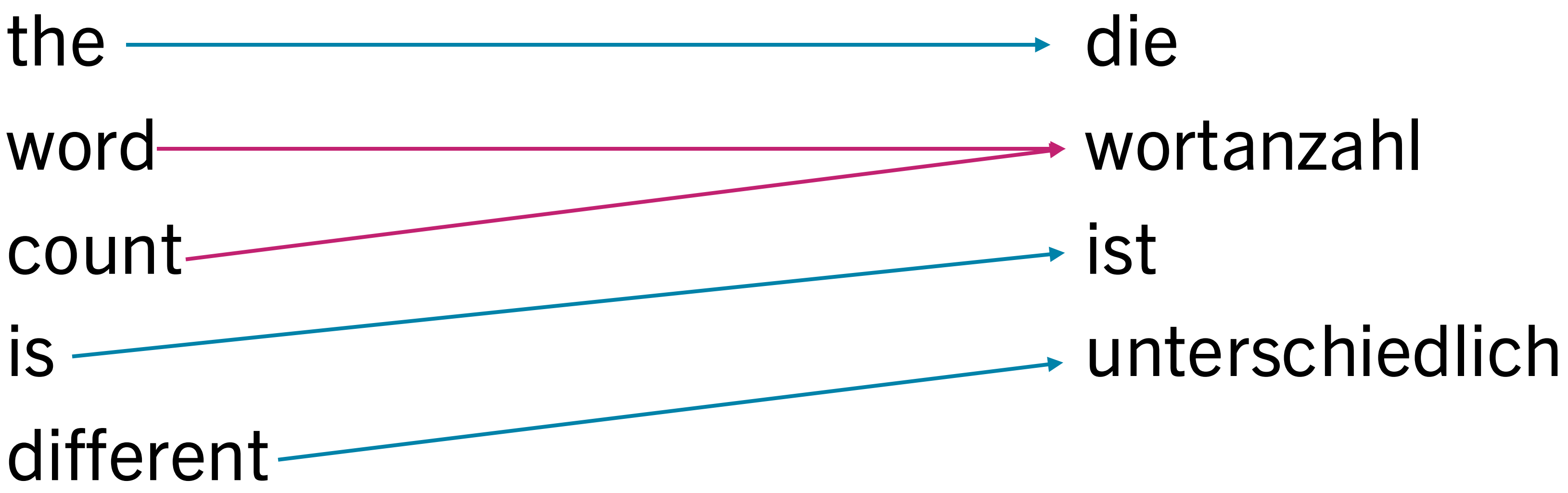
- Vaswani et al., 2017 introduced the Transformer architecture to address these issues
- Key innovation: the **attention mechanism**, which allows the model to focus on relevant parts of the input sequence
- Transformers process sequences in parallel, enabling faster training and better handling of long-range dependencies



Attention



Attention



the

word

count

is

different

die

wortanzahl

ist

unterschiedlich

Attention

- Attention mechanism predicts how much each word *depends* on the words in the input
- By understanding this relationship, prediction power for text is greatly improved

	le	chien	blue
the	1		
blue		0.2	1
dog		1	0.2

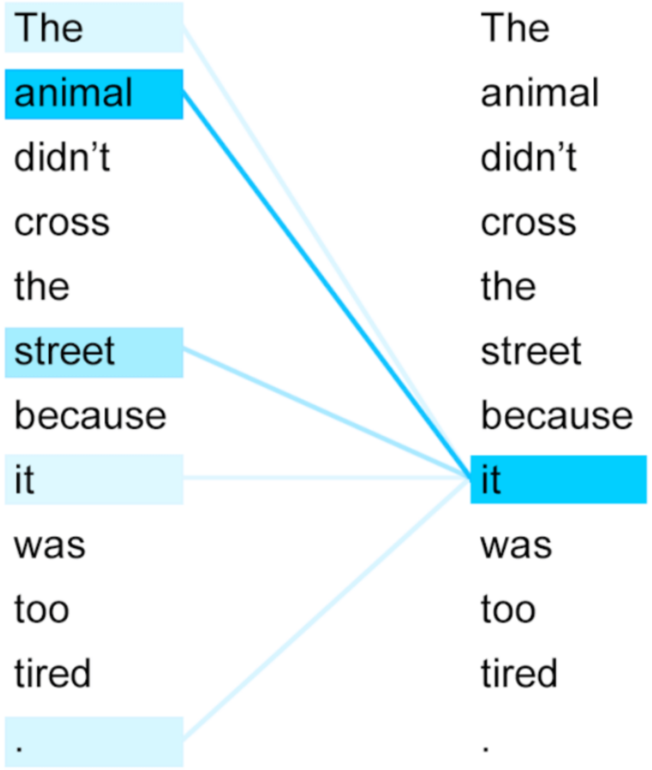
Attention

- This dependency matrix can then be multiplied against the word embeddings to create new, contextual word embeddings

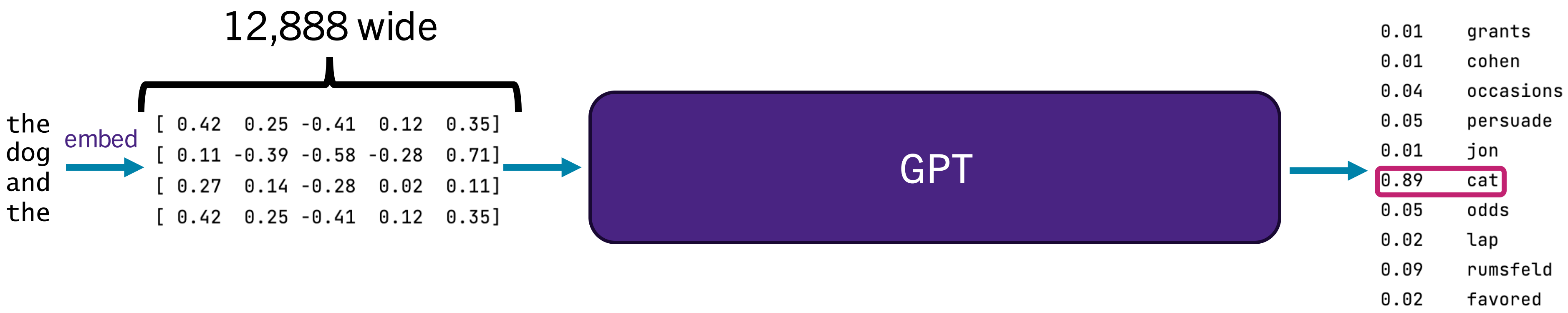
	wortanzahl	variiert
word	0.5	
count	0.5	
differs		1

Self-Attention

- In models like ChatGPT, we use *self-attention* – simply put, the relationship is now between the phrase and itself



Building GPT



12,888 wide

the
dog
and
the

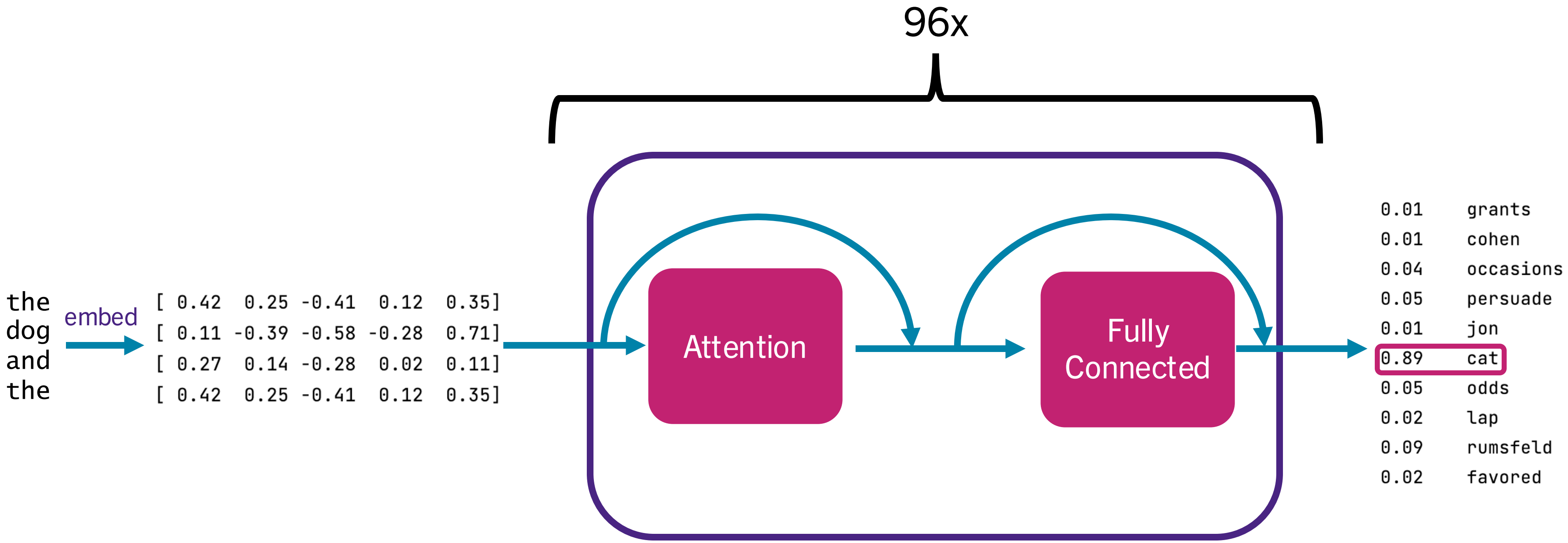
embed

```
[ 0.42  0.25 -0.41  0.12  0.35]
[ 0.11 -0.39 -0.58 -0.28  0.71]
[ 0.27  0.14 -0.28  0.02  0.11]
[ 0.42  0.25 -0.41  0.12  0.35]
```

GPT

0.01	grants
0.01	cohen
0.04	occasions
0.05	persuade
0.01	jon
0.89	cat
0.05	odds
0.02	lap
0.09	rumsfeld
0.02	favored

Building GPT: The Transfomer

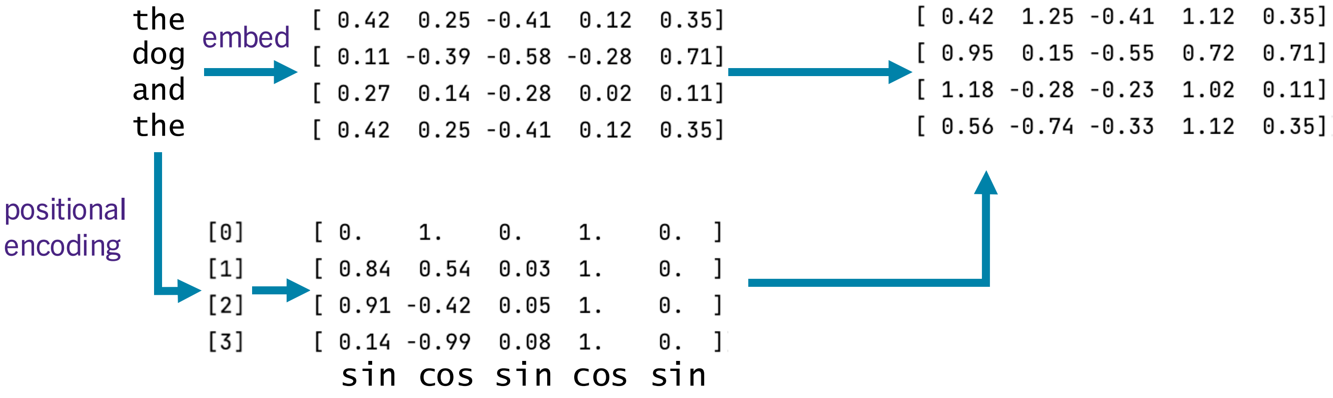


the
dog
and
the

```
[ 0.42  0.25 -0.41  0.12  0.35]
[ 0.11 -0.39 -0.58 -0.28  0.71]
[ 0.27  0.14 -0.28  0.02  0.11]
[ 0.42  0.25 -0.41  0.12  0.35]
```

0.01	grants
0.01	cohen
0.04	occasions
0.05	persuade
0.01	jon
0.89	cat
0.05	odds
0.02	lap
0.09	rumsfeld
0.02	favored

Building GPT: Positional Embedding

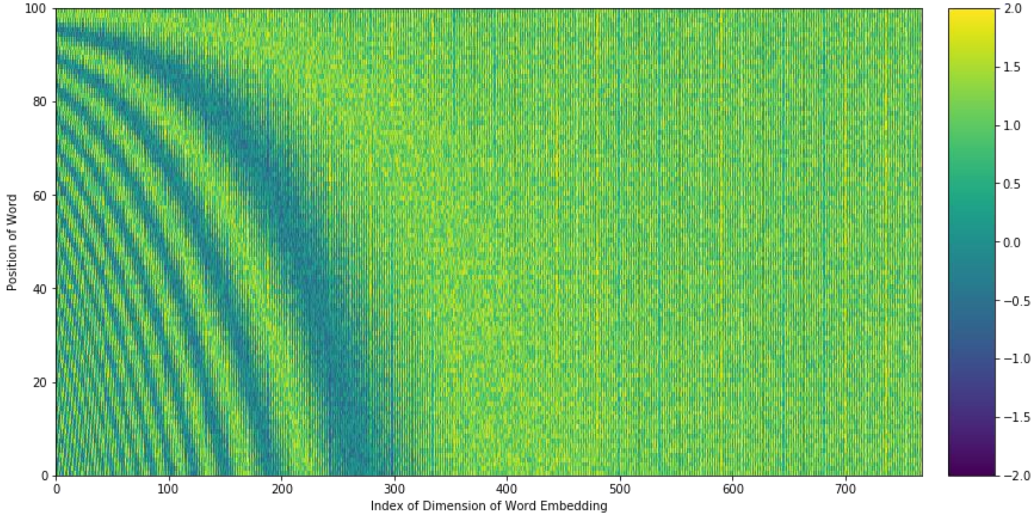


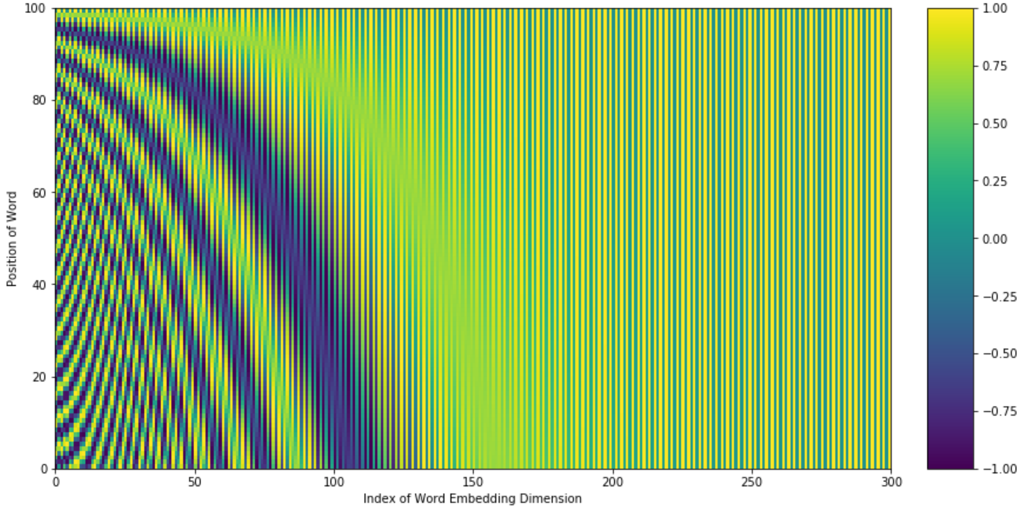
```
the       [ 0.42  0.25 -0.41  0.12  0.35]        [ 0.42  1.25 -0.41  1.12  0.35]
dog       [ 0.11 -0.39 -0.58 -0.28  0.71]        [ 0.95  0.15 -0.55  0.72  0.71]
and       [ 0.27  0.14 -0.28  0.02  0.11]        [ 1.18 -0.28 -0.23  1.02  0.11]
the       [ 0.42  0.25 -0.41  0.12  0.35]        [ 0.56 -0.74 -0.33  1.12  0.35]
```

embed

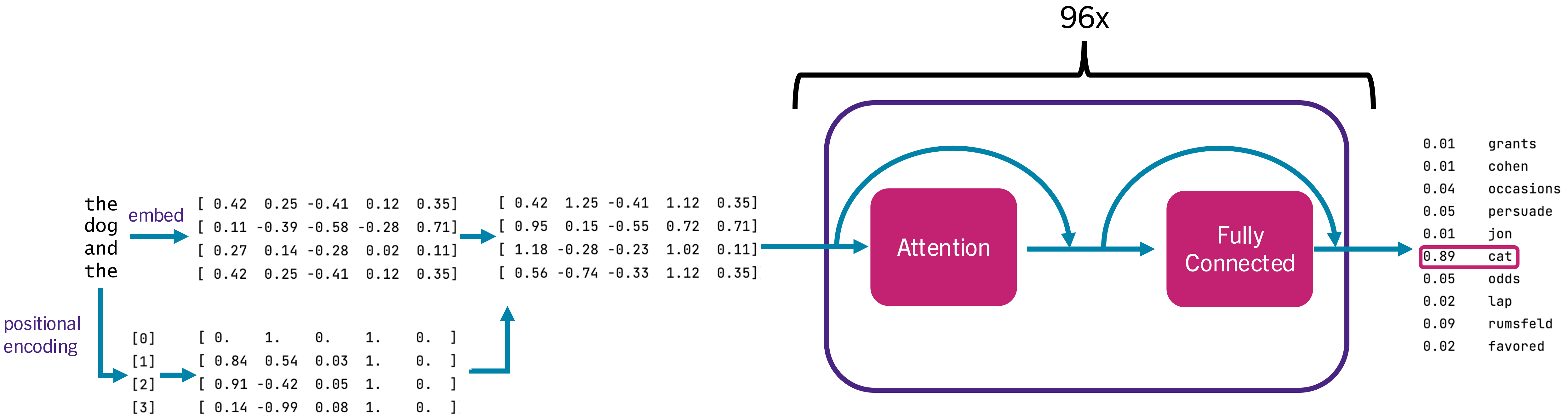
positional encoding

```
[0]   [ 0.    1.    0.    1.    0.  ]
[1]   [ 0.84  0.54  0.03  1.    0.  ]
[2]   [ 0.91 -0.42  0.05  1.    0.  ]
[3]   [ 0.14 -0.99  0.08  1.    0.  ]
        sin   cos   sin   cos   sin
```





Building GPT



```
the
dog
and
the
```

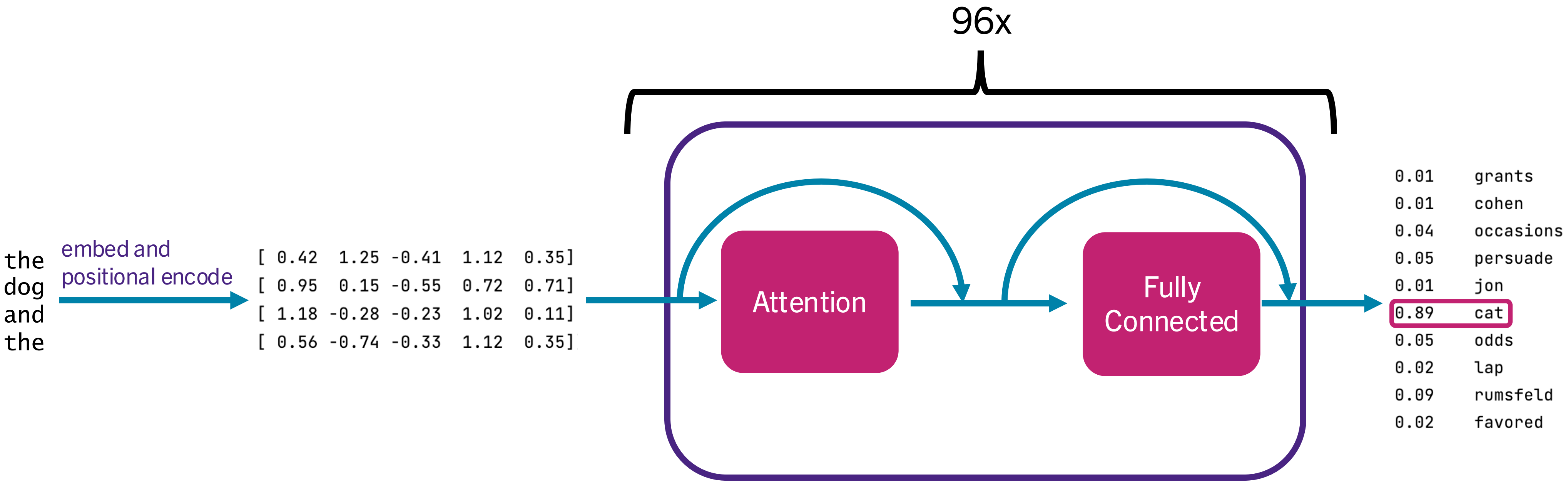
```
[ 0.42  0.25 -0.41  0.12  0.35]
[ 0.11 -0.39 -0.58 -0.28  0.71]
[ 0.27  0.14 -0.28  0.02  0.11]
[ 0.42  0.25 -0.41  0.12  0.35]
```

```
[0]     [ 0.    1.    0.    1.    0.  ]
[1]     [ 0.84  0.54  0.03  1.    0.  ]
[2]     [ 0.91 -0.42  0.05  1.    0.  ]
[3]     [ 0.14 -0.99  0.08  1.    0.  ]
```

```
[ 0.42  1.25 -0.41  1.12  0.35]
[ 0.95  0.15 -0.55  0.72  0.71]
[ 1.18 -0.28 -0.23  1.02  0.11]
[ 0.56 -0.74 -0.33  1.12  0.35]
```

0.01	grants
0.01	cohen
0.04	occasions
0.05	persuade
0.01	jon
0.89	cat
0.05	odds
0.02	lap
0.09	rumsfeld
0.02	favored

Building GPT



the
dog
and
the

embed and
positional encode

```
[ 0.42  1.25 -0.41  1.12  0.35]
[ 0.95  0.15 -0.55  0.72  0.71]
[ 1.18 -0.28 -0.23  1.02  0.11]
[ 0.56 -0.74 -0.33  1.12  0.35]
```

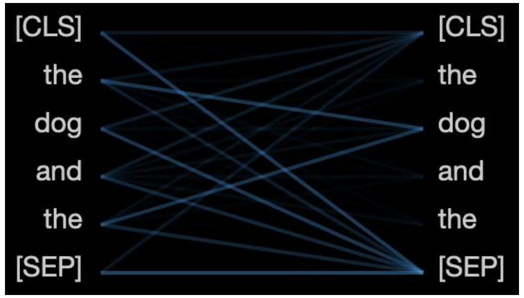
96x

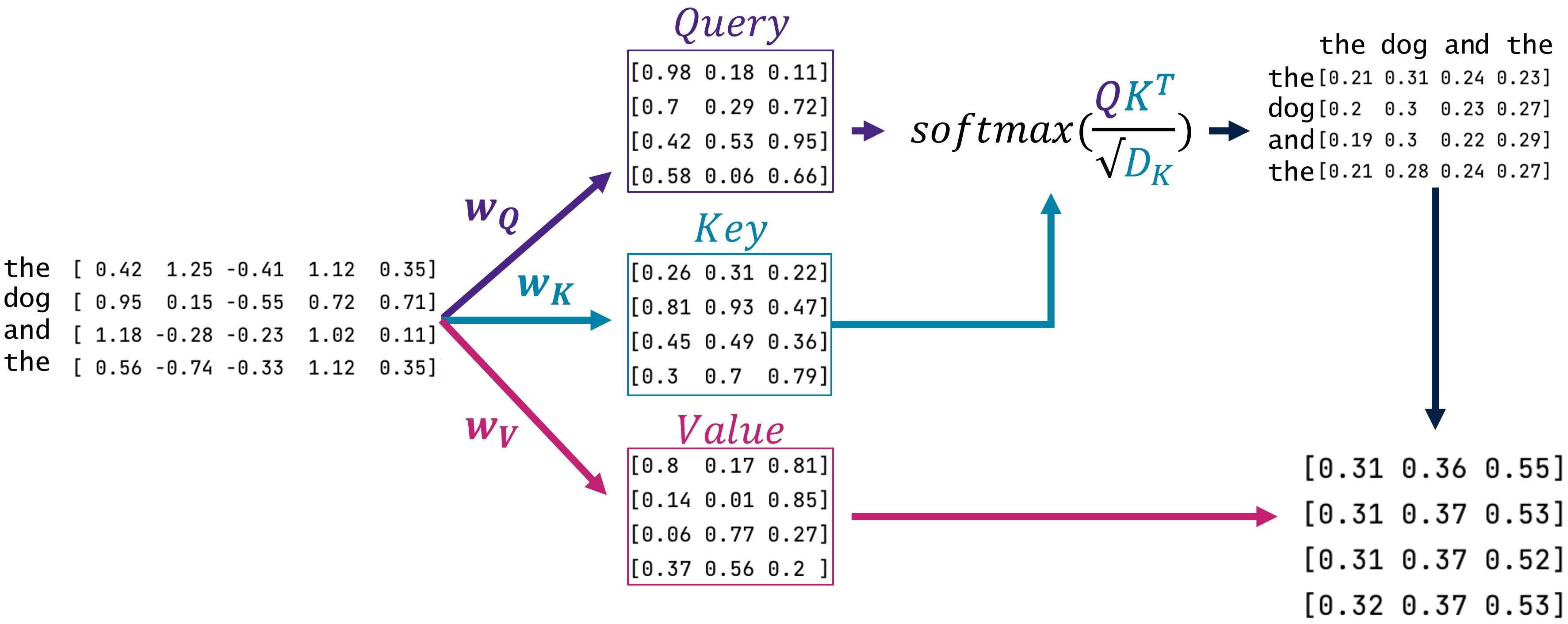
Attention

Fully Connected

0.01	grants
0.01	cohen
0.04	occasions
0.05	persuade
0.01	jon
0.89	cat
0.05	odds
0.02	lap
0.09	rumsfeld
0.02	favored

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

w_K

w_V

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]
```

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]
```

Value

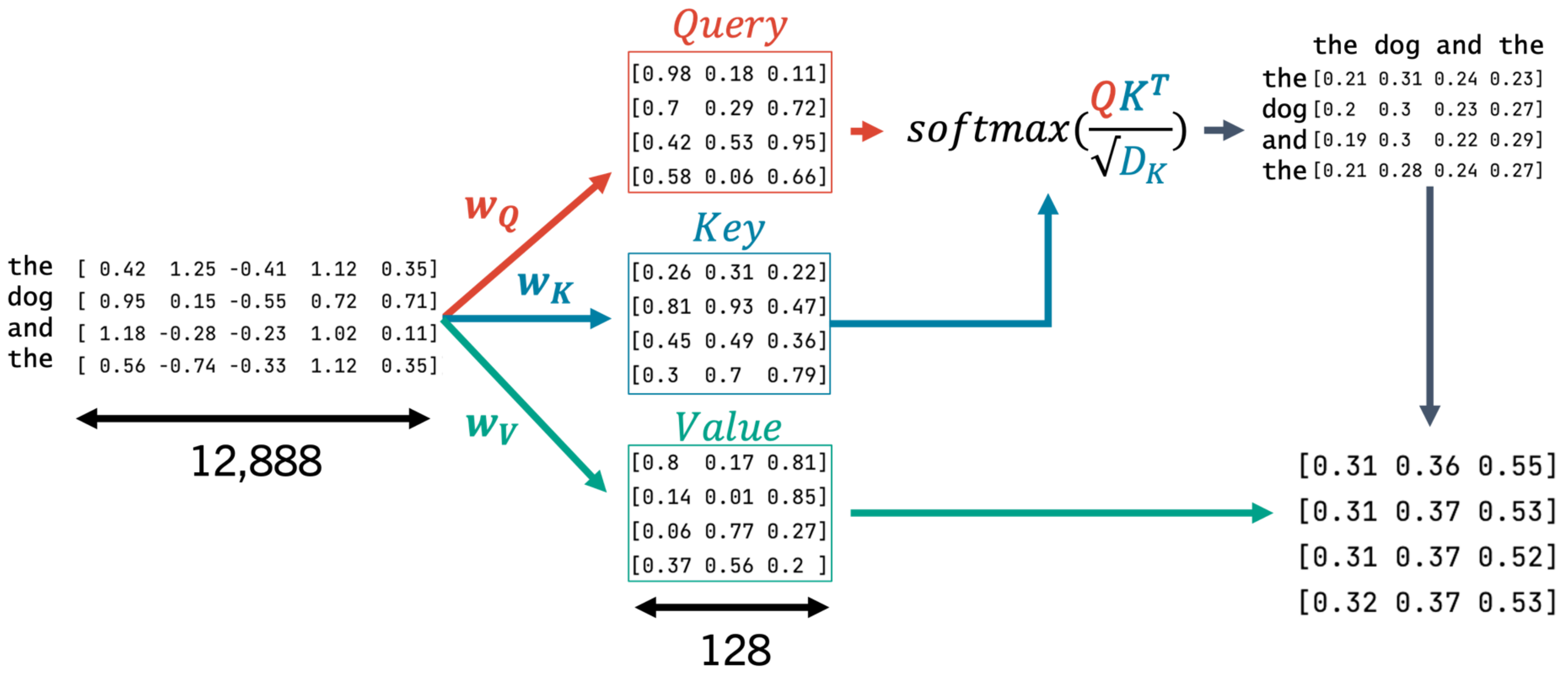
```
[0.8  0.17 0.81]
[0.14 0.01 0.85]
[0.06 0.77 0.27]
[0.37 0.56 0.2 ]
```

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

$$\begin{array}{l|llll} & \text{the} & \text{dog} & \text{and} & \text{the} \\ \text{the} & 0.21 & 0.31 & 0.24 & 0.23 \\ \text{dog} & 0.2 & 0.3 & 0.23 & 0.27 \\ \text{and} & 0.19 & 0.3 & 0.22 & 0.29 \\ \text{the} & 0.21 & 0.28 & 0.24 & 0.27 \end{array}$$

```
[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



```
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]
```

12,888

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]
```

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]
```

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 ]
```

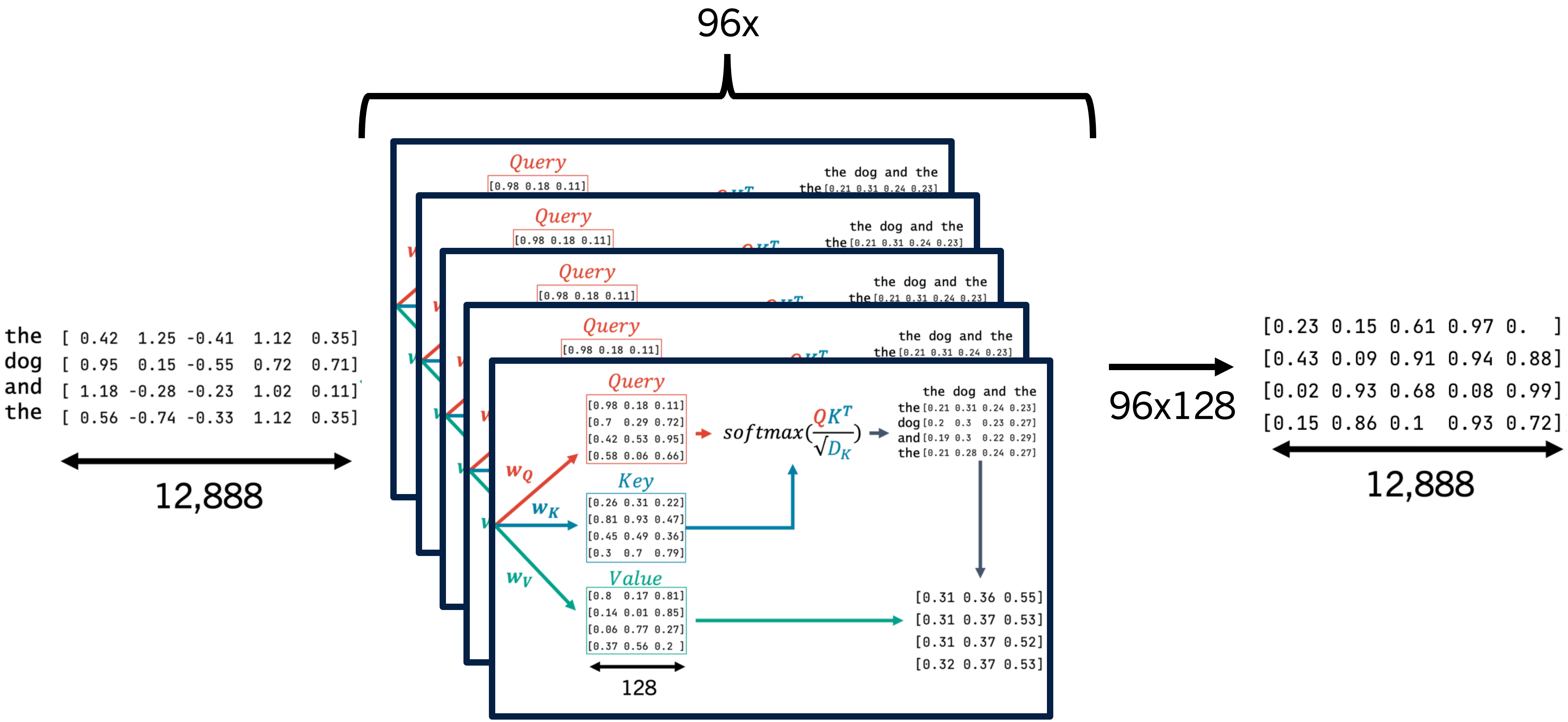
128

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

```
    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



96x

```
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]
```

12,888

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]
```

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]
```

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 ]
```

128

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

```
     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]
```

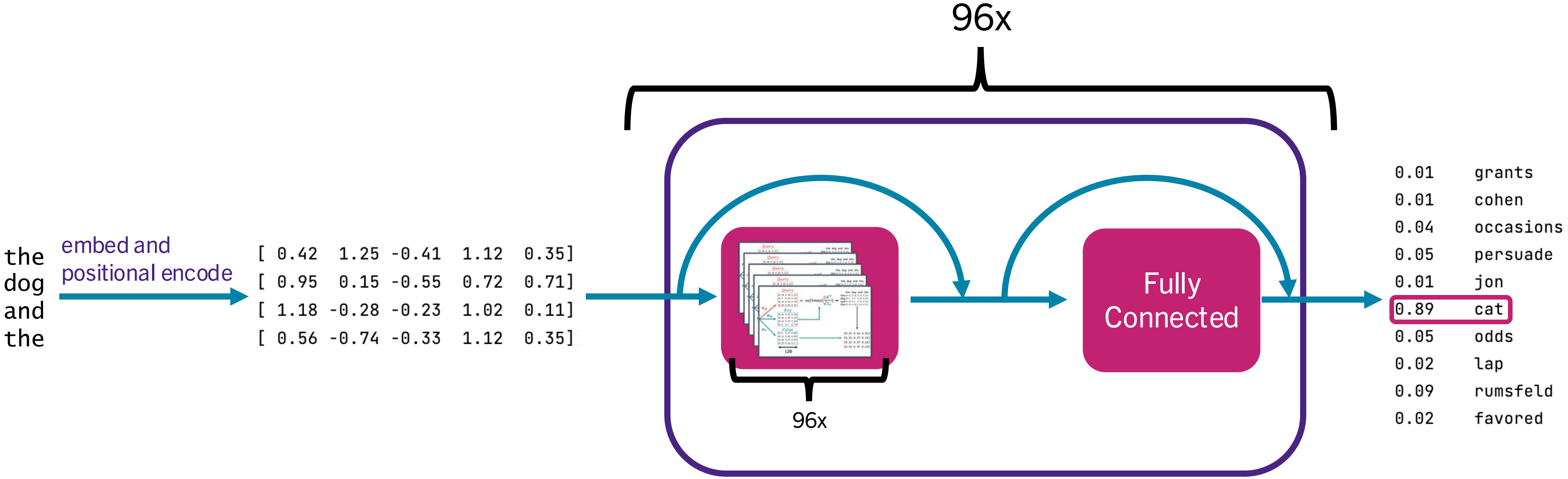
```
[0.31 0.36 0.55]
[0.31 0.37 0.53]
[0.31 0.37 0.52]
[0.32 0.37 0.53]
```

96x128

```
[0.23 0.15 0.61 0.97 0.  ]
[0.43 0.09 0.91 0.94 0.88]
[0.02 0.93 0.68 0.08 0.99]
[0.15 0.86 0.1  0.93 0.72]
```

12,888

Building GPT

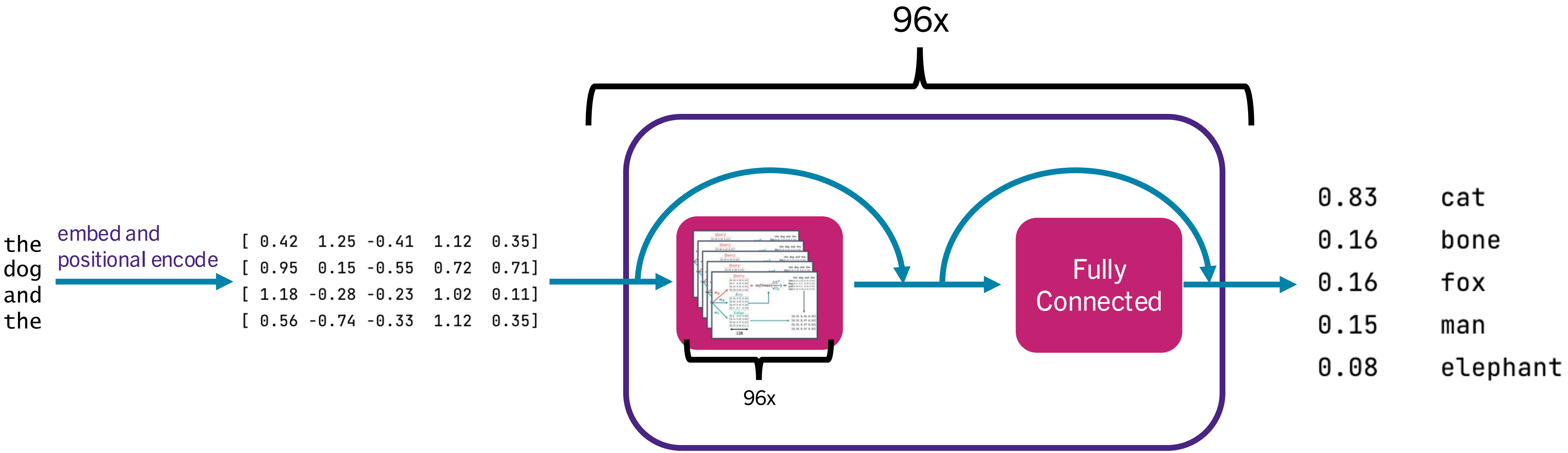


the
dog
and
the

```
[ 0.42  1.25 -0.41  1.12  0.35]
[ 0.95  0.15 -0.55  0.72  0.71]
[ 1.18 -0.28 -0.23  1.02  0.11]
[ 0.56 -0.74 -0.33  1.12  0.35]
```

0.01	grants
0.01	cohen
0.04	occasions
0.05	persuade
0.01	jon
0.89	cat
0.05	odds
0.02	lap
0.09	rumsfeld
0.02	favored

Building GPT: Top-P



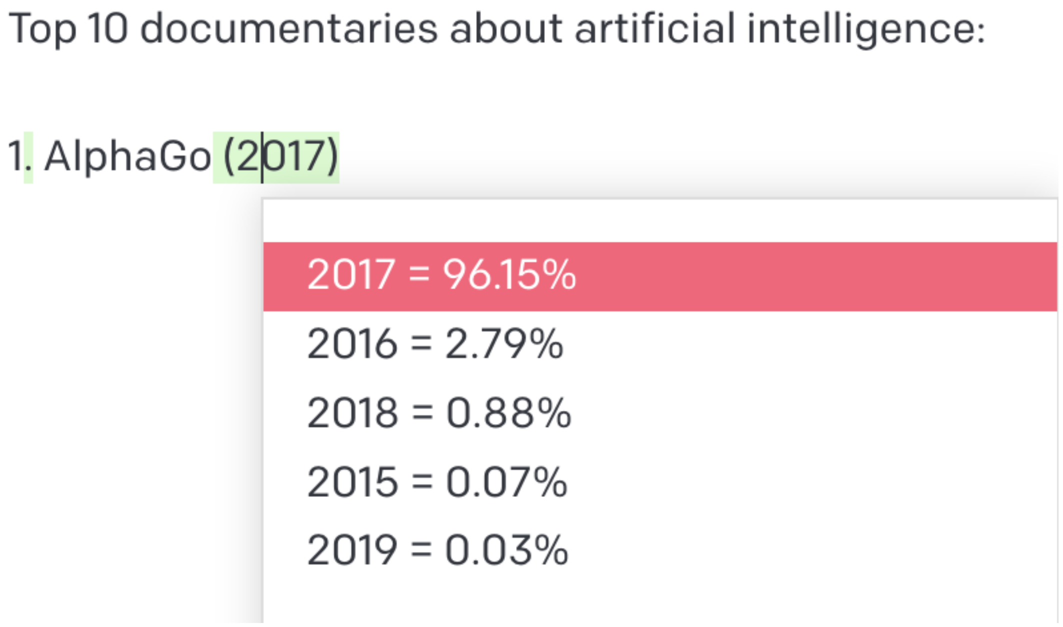
the
dog
and
the

embed and positional encode

```
[ 0.42  1.25 -0.41  1.12  0.35]
[ 0.95  0.15 -0.55  0.72  0.71]
[ 1.18 -0.28 -0.23  1.02  0.11]
[ 0.56 -0.74 -0.33  1.12  0.35]
```

0.83	cat
0.16	bone
0.16	fox
0.15	man
0.08	elephant

Building GPT: Top-P



Top 10 documentaries about artificial intelligence:

1. AlphaGo (2017)

2017 = 96.15%

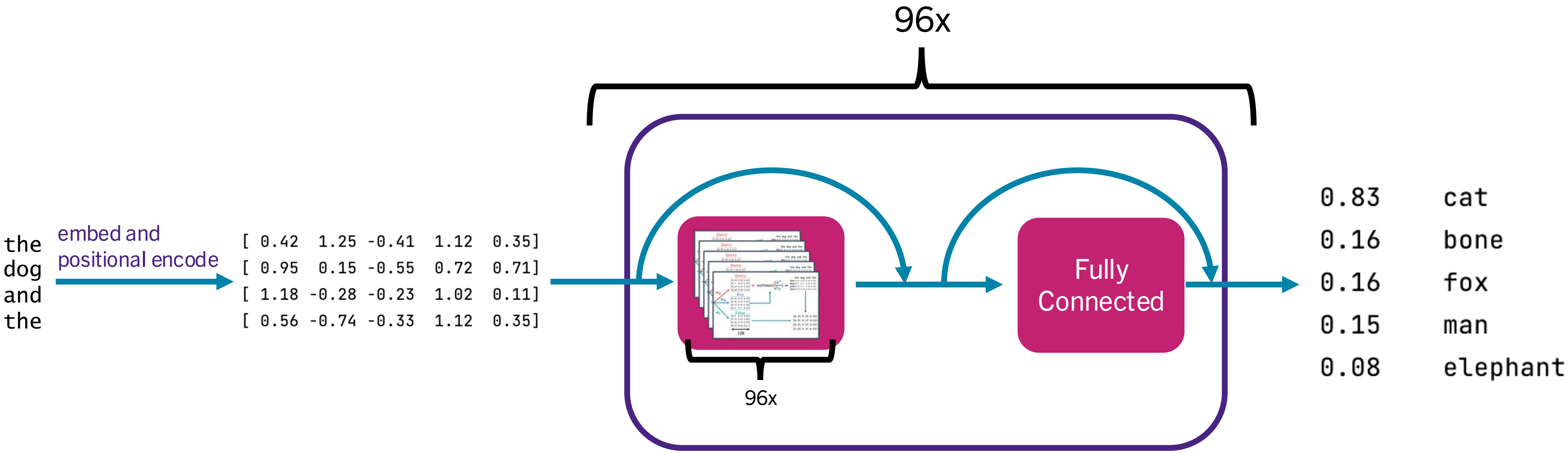
2016 = 2.79%

2018 = 0.88%

2015 = 0.07%

2019 = 0.03%

Building GPT



the
dog
and
the

embed and positional encode

```
[ 0.42  1.25 -0.41  1.12  0.35]
[ 0.95  0.15 -0.55  0.72  0.71]
[ 1.18 -0.28 -0.23  1.02  0.11]
[ 0.56 -0.74 -0.33  1.12  0.35]
```

96x

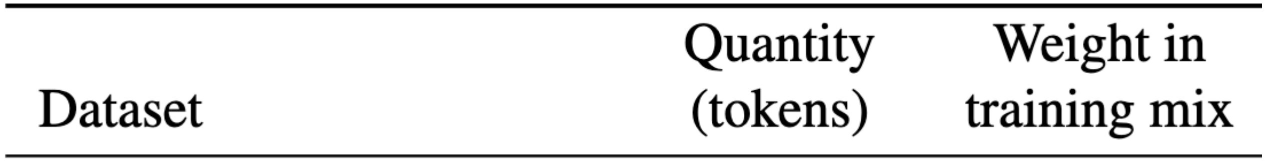
96x

Fully Connected

0.83	cat
0.16	bone
0.16	fox
0.15	man
0.08	elephant

GPT’s Training Data

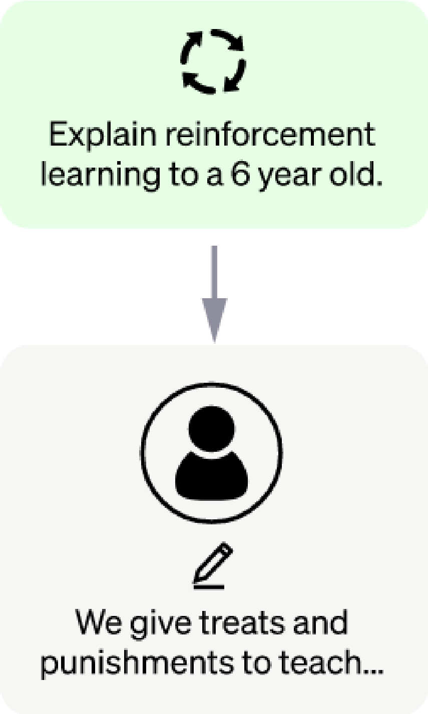
- 1 token ≈ ¾ word
- Some datasets are sampled more times than others
- Common Crawl: billions of webpages collected over 7 years
- Webtext2: Dataset of webpages that have been shared on Reddit
- Books1: Free ebooks (?)
- Books2: Secret!
- English Wikipedia



	Quantity	Weight in
Dataset	(tokens)	training mix

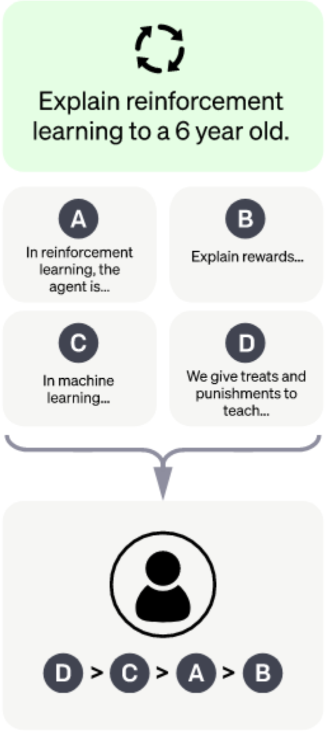
The training innovation of ChatGPT

Human annotators write answers to questions


Explain reinforcement learning to a 6 year old.
We give treats and punishments to teach...

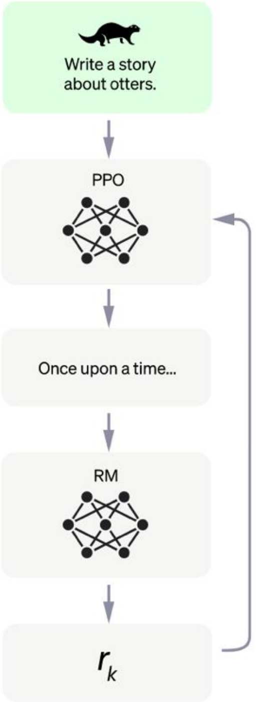
The generalist GPT model is taught from these Q&A pairs

Human annotators write more answers, and someone else ranks them


Explain reinforcement learning to a 6 year old.
A
In reinforcement learning, the agent is...
B
Explain rewards...
C
In machine learning...
D
We give treats and punishments to teach...
D > C > A > B

A separate model learns to rate the quality of an answer

GPT writes answers to sampled questions

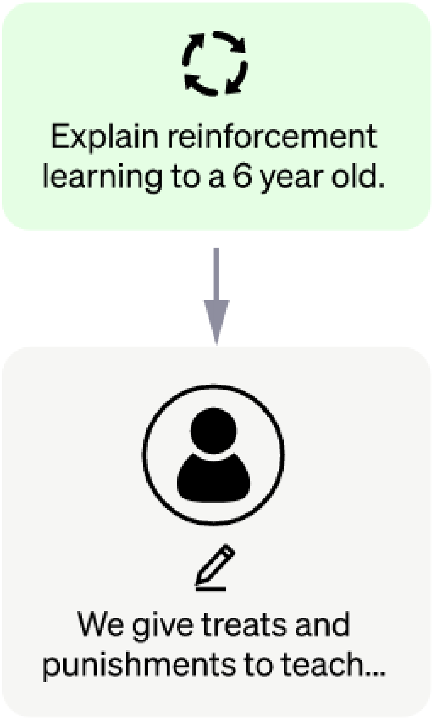


The reward model rates each answer, allowing GPT to keep learning

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

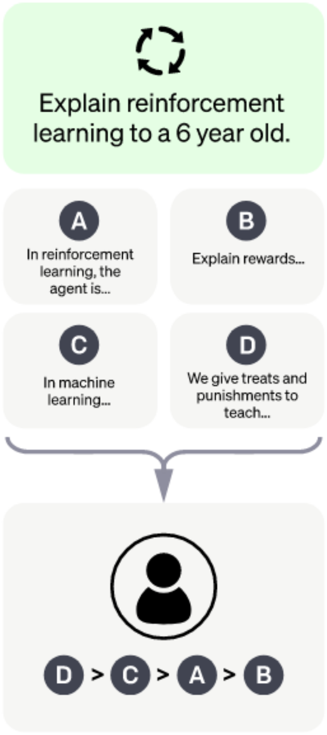
The training innovation of ChatGPT

Human annotators write answers to questions


Explain reinforcement learning to a 6 year old.
We give treats and punishments to teach...

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

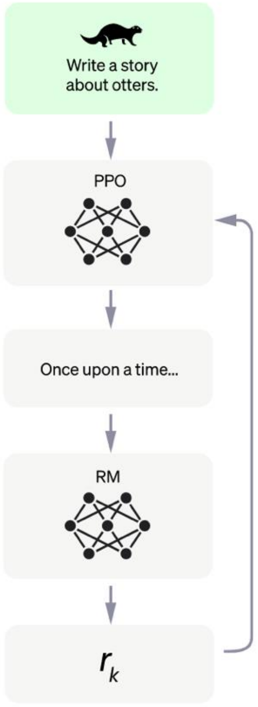
Human annotators write more answers, and someone else ranks them


Explain reinforcement learning to a 6 year old.
A
In reinforcement learning, the agent is...
B
Explain rewards...
C
In machine learning...
D
We give treats and punishments to teach...
D > C > A > B

A separate model learns to rate the quality of an answer

No more humans involved!

GPT writes answers to sampled questions

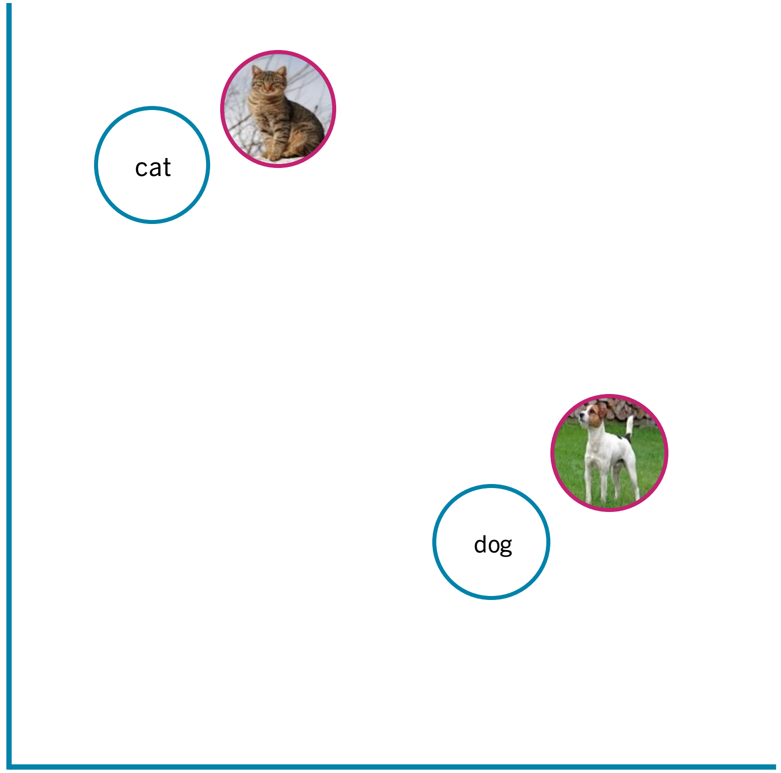


The reward model rates each answer, allowing GPT to keep learning

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

Text-to-image

- With the advent of large language models, it soon became possible to direct the generation using text description
- This is achieved by representing text and imagery in the same *embedding space* as one another



Text-to-Speech

- Text-to-Speech (TTS) has existed for decades
- Typical:
  - Break text into standard word sounds (phonemes)
  - Combine pre-created (or even pre-recorded) phonemes
  - Result: speech
  - Only the first step requires prediction – phonemes vary depending on word order, context, etc. (e.g. “I will read” vs “I have read”)

Microsoft SAM (1982)
rule-based signal generation

DECtalk (1990s)
phoneme rules + prosody control

Google WaveNet (2016)
deep-learning based

Modern TTS

- State-of-the-art text to speech works similarly to image generation
- Text is *co-embedded* with *audio tokens*
- A denoising model then converts audio noise into coherent speech using the *audio tokens* as input
- Tone, emotion and pacing are encoded implicitly in the process
  - One phoneme may have many corresponding audio tokens depending on these additional features
- In this way, the model predicts more than a flat computer voice

Google NotebookLM (2025)

Video challenges

- At their most basic, video generation can be achieved by training image generators to slightly vary content
    - Therefore producing frames
- However, this approach lacks a “plan” of what should happen in the video
- Even worse, there is no understanding of what has already occurred

StableDiffusion (2023)

How video generation works

1. Text description is converted into embeddings
2. *Spatio-temporal chunks* (e.g. four seconds of top-right corner) initialized
3. Content of each chunk is predicted based on input *(across time)*
4. Model evaluates “plan” all together – checks for coherence
5. Frames are generated from plan using diffusion

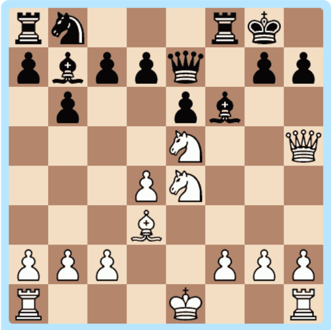
Predictor to Agent

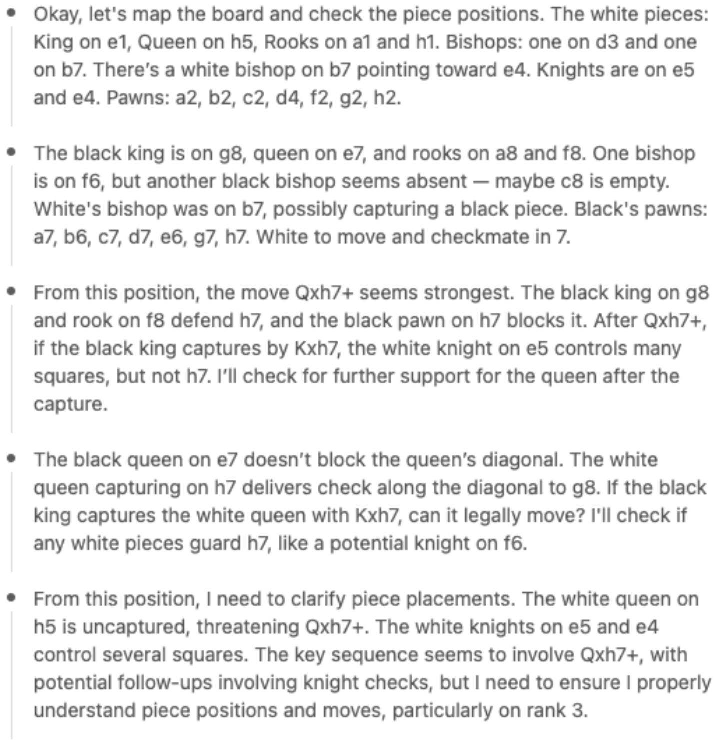
- Predictor takes input and makes prediction
  - Fixed world understanding (developed in training)
  - No ability to take actions – no influence on the world
- Agent requires intermediate steps
  - Plan: consider what course of action would produce the desired result
  - Act: select and execute an action that moves toward that result
  - Observe: receive updated information about the world
  - Reflect: determine if action had desired consequence; if result is achieved
  - *Feedback loop*

Planning

- For LLMs to plan, they must *say it out loud*
- This is where Chain-of-Thought (CoT) reasoning comes in
- CoT allows the model to plan *before* making a final response to the user
- The reasoning may be hidden from the user, but is not fundamentally different to a reply

Edward Lasker vs. sir George Alan Thomas, London, 1912. White to move and win in 7! In the real confrontation the mate comes after the 8th, spectacular, move.

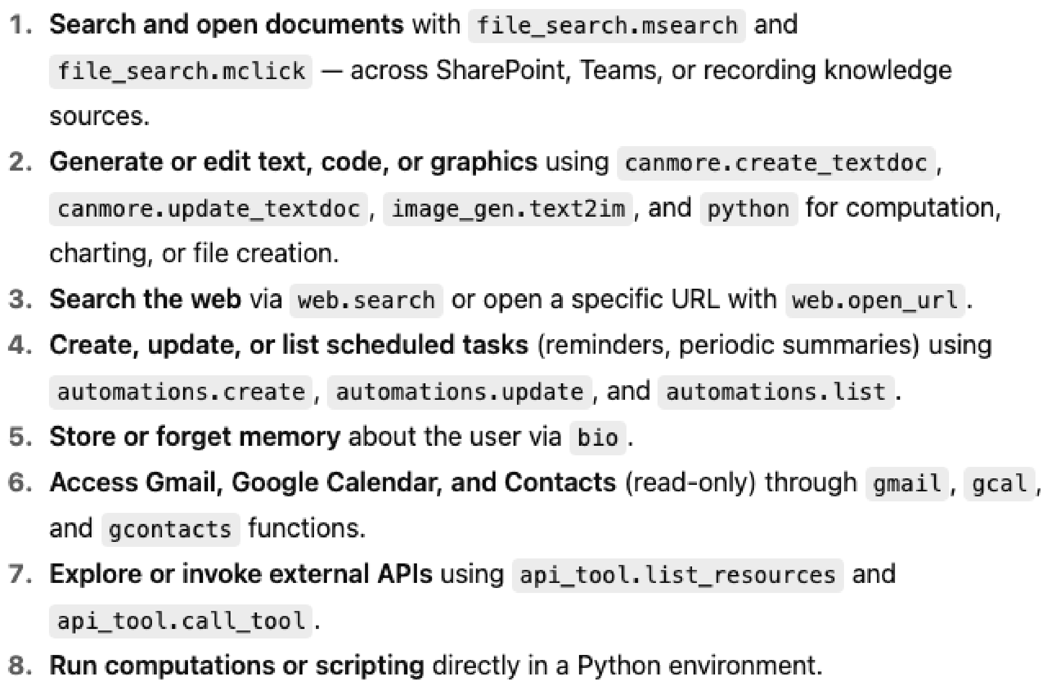




- Okay, let's map the board and check the piece positions. The white pieces: King on e1, Queen on h5, Rooks on a1 and h1. Bishops: one on d3 and one on b7. There's a white bishop on b7 pointing toward e4. Knights are on e5 and e4. Pawns: a2, b2, c2, d4, f2, g2, h2.
- The black king is on g8, queen on e7, and rooks on a8 and f8. One bishop is on f6, but another black bishop seems absent — maybe c8 is empty. White's bishop was on b7, possibly capturing a black piece. Black's pawns: a7, b6, c7, d7, e6, g7, h7. White to move and checkmate in 7.
- From this position, the move Qxh7+ seems strongest. The black king on g8 and rook on f8 defend h7, and the black pawn on h7 blocks it. After Qxh7+, if the black king captures by Kxh7, the white knight on e5 controls many squares, but not h7. I'll check for further support for the queen after the capture.
- The black queen on e7 doesn't block the queen's diagonal. The white queen capturing on h7 delivers check along the diagonal to g8. If the black king captures the white queen with Kxh7, can it legally move? I'll check if any white pieces guard h7, like a potential knight on f6.
- From this position, I need to clarify piece placements. The white queen on h5 is uncaptured, threatening Qxh7+. The white knights on e5 and e4 control several squares. The key sequence seems to involve Qxh7+, with potential follow-ups involving knight checks, but I need to ensure I properly understand piece positions and moves, particularly on rank 3.

Tool Use

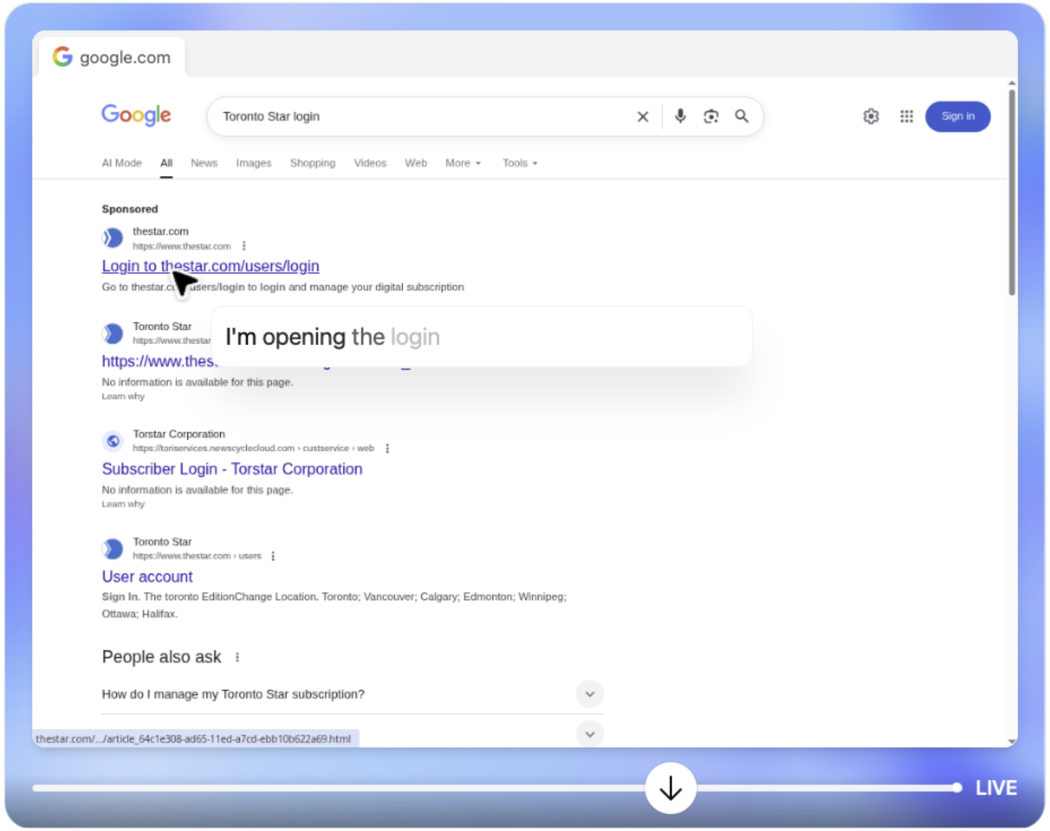
- To meaningfully act, LLM agents must be able to *use tools*
- In early development, this largely revolved around APIs — pre-existing technology for programmatic interface with systems
- OpenAI has added several in-built tools to ChatGPT



1. **Search and open documents** with `file_search.msearch` and `file_search.mclick` — across SharePoint, Teams, or recording knowledge sources.
2. **Generate or edit text, code, or graphics** using `canmore.create_textdoc`, `canmore.update_textdoc`, `image_gen.text2im`, and `python` for computation, charting, or file creation.
3. **Search the web** via `web.search` or open a specific URL with `web.open_url`.
4. **Create, update, or list scheduled tasks** (reminders, periodic summaries) using `automations.create`, `automations.update`, and `automations.list`.
5. **Store or forget memory** about the user via `bio`.
6. **Access Gmail, Google Calendar, and Contacts** (read-only) through `gmail`, `gcal`, and `gcontacts` functions.
7. **Explore or invoke external APIs** using `api_tool.list_resources` and `api_tool.call_tool`.
8. **Run computations or scripting** directly in a Python environment.

Tool Use

- ChatGPT Agent mode allows for direct interface with a virtual computer
- This greatly expands the variety of tools available for a model to use
- However, usage is slow: interacting with a system designed for people is non-trivial



Sponsored

thestar.com
https://www.thestar.com
Login to thestar.com/users/login
Go to thestar.co users/login to login and manage your digital subscription

Toronto Star
https://www.thestar
https://www.thes.
No information is available for this page.
Learn why

I'm opening the login

Torstar Corporation
https://toriservices.newscyclecloud.com › custservice › web
Subscriber Login - Torstar Corporation
No information is available for this page.
Learn why

Toronto Star
https://www.thestar.com › users
User account
Sign In. The toronto EditionChange Location. Toronto; Vancouver; Calgary; Edmonton; Winnipeg; Ottawa; Halifax.

People also ask

How do I manage my Toronto Star subscription?

thestar.com/.../article_64c1e308-ad65-11ed-a7cd-ebb10b622a69.html

Memory & Context

- LLMs have a fixed **context window**: the number of tokens that can be processed by the system
- GPT-5 has a context window of 128,000 tokens (approximately 215 pages of text)
- Chain-of-Thought reasoning and tool use can quickly use up this window
- Model may forget what the user wants it to do
- Or what it has already tried!

Next: Lab 6!