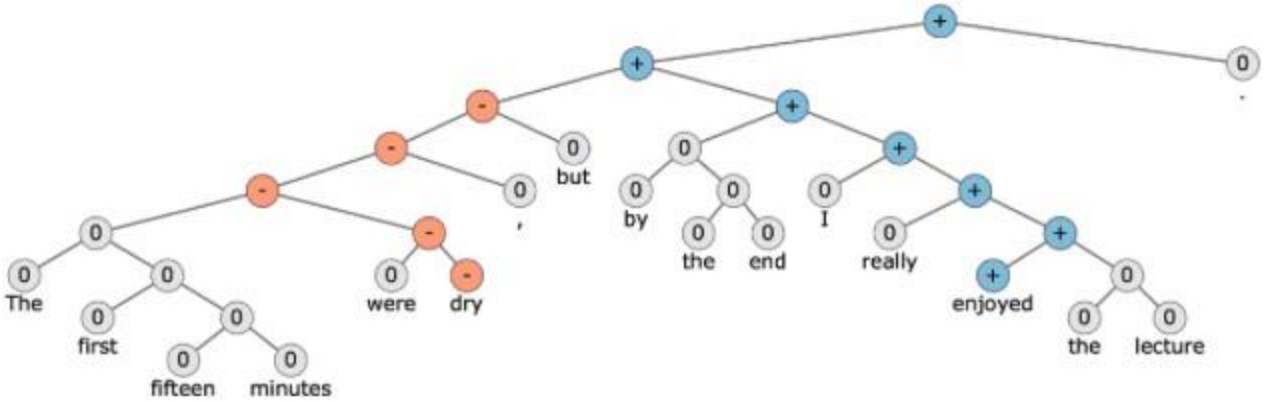
Deep Learning: Natural Language Processing with Deep Learning

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

Natural Language Processing

[Google Translate System - 2016]



[Socher 2015]

Natural Language Processing

- Sentence/Document level Classification (topic, sentiment)
- Topic modeling (LDA, ...)
- Translation
- Chatbots /dialogue systems /assistants (Alexa, ...)
- Summarization

Useful open source projects

Outline

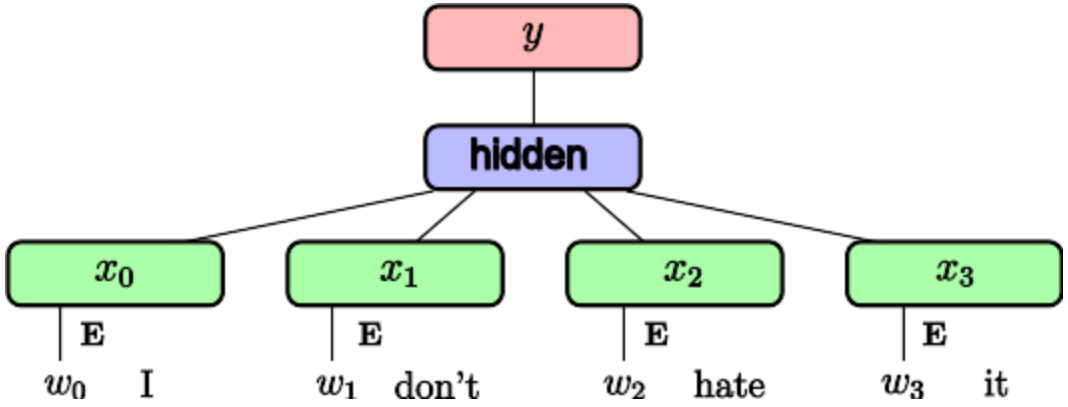
- Classification and word representation
- Word2Vec
- Language Modelling
- Transformers

Word Representation and Word2Vec

Word representation

- Words are indexed and represented as 1-hot vectors
- Large Vocabulary of possible words $|V|$
- Use of Embeddings as inputs in all Deep NLP tasks
- Word embeddings usually have dimensions 50, 100, 200, 300

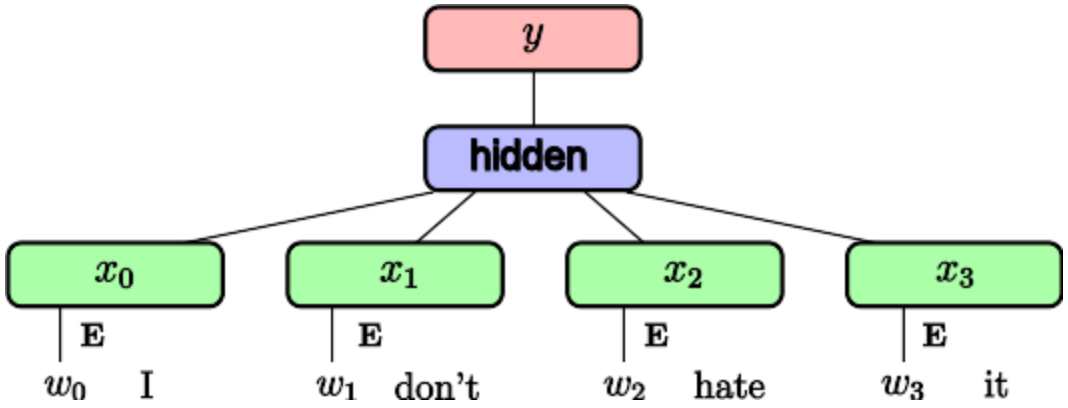
Supervised Text Classification



- $\mathbf{E}$ embedding (linear projection) $\rightarrow$ `|V| x H`
- Embeddings are averaged $\rightarrow$ hidden activation size: `H`
- Dense output connection $\mathbf{W}, \mathbf{b} \rightarrow$ `H x K`
- Softmax and cross-entropy loss

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

Supervised Text Classification



- Very efficient (speed and accuracy) on large datasets
- State-of-the-art (or close to) on several classification, when adding bigrams/trigrams
- Little gains from depth

Joulin, Armand, et al. "Bag of tricks for efficient text classification." FAIR 2016

Transfer Learning for Text

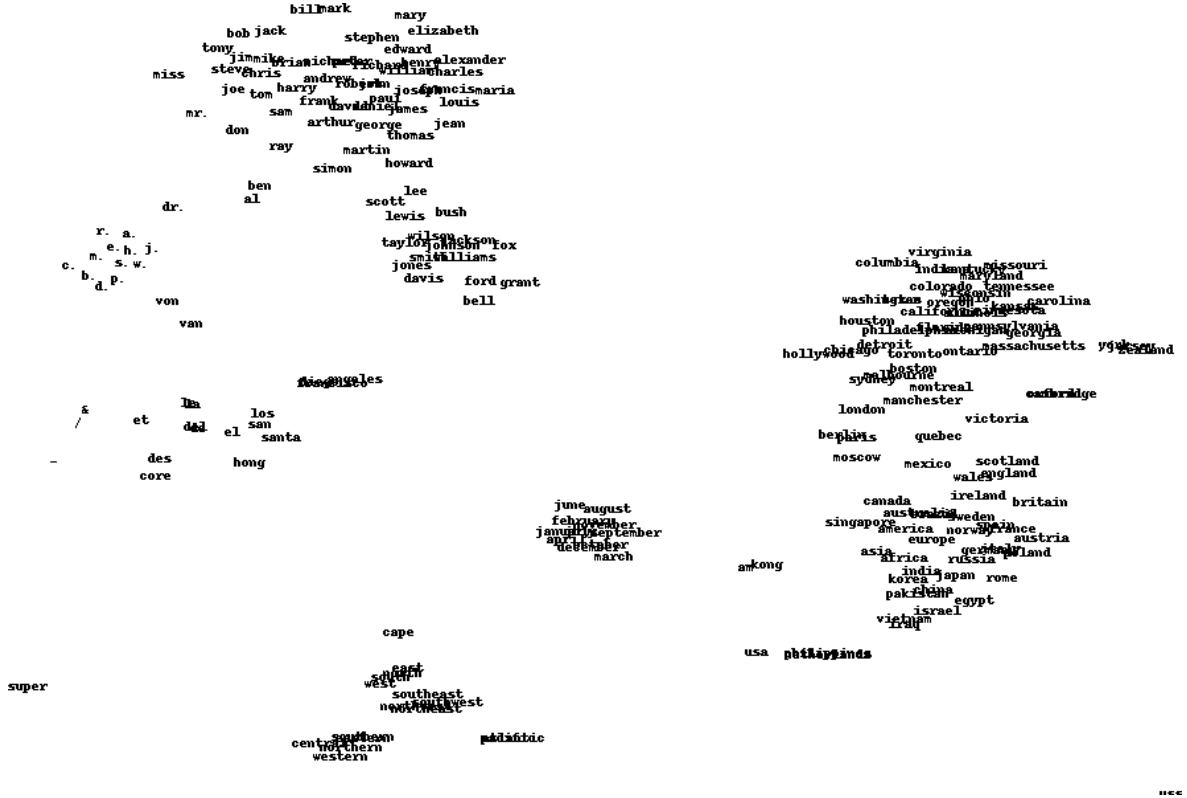
Similar to image: can we have word representations that are generic enough to transfer from one task to another?

Unsupervised /self-supervised learning of word representations

Unlabelled text data is almost infinite:

- Wikipedia dumps
- Project Gutenberg
- Social Networks
- Common Crawl

Word Vectors



excerpt from work by J. Turian on a model trained by R. Collobert et al. 2008

Word2Vec

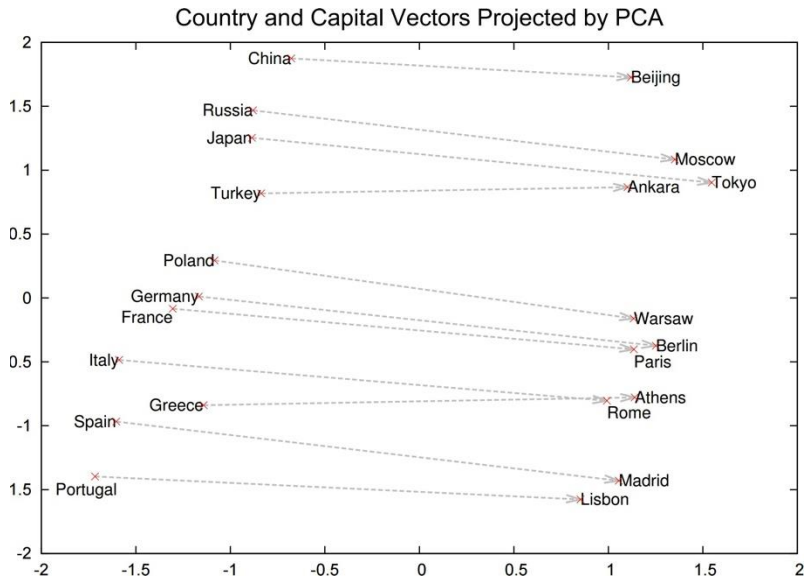
FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Compositionality

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Colobert et al. 2011, Mikolov, et al. 2013

Word Analogies



- Linear relations in Word2Vec embeddings
- Many come from text structure (e.g. Wikipedia)

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Self-supervised training

Distributional Hypothesis (Harris, 1954):
"words are characterized by the company that they keep"

Main idea: learning word embeddings by predicting word contexts

Given a word e.g. "carrot" and any other word $w \in V$ predict probability $P(w|\text{carrot})$ that w occurs in the context of "carrot".

- Unsupervised /self-supervised: no need for class labels.
- (Self-)supervision comes from context.
- Requires a lot of text data to cover rare words correctly.

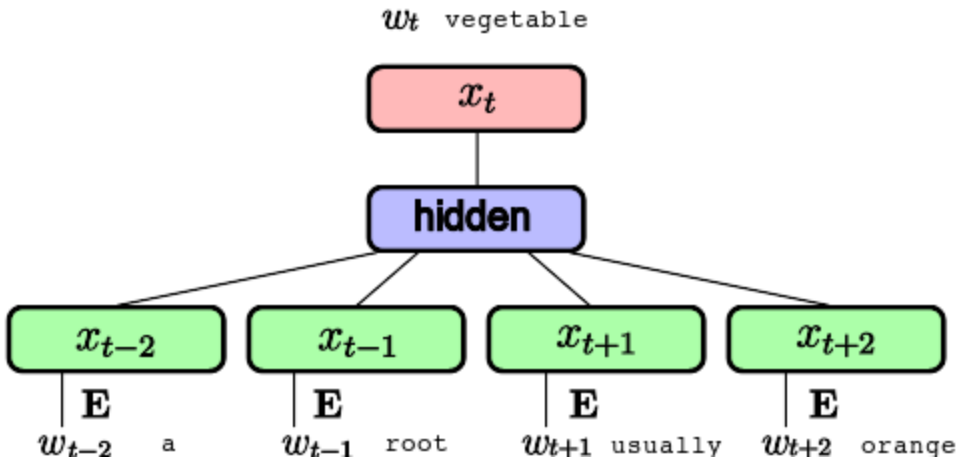
Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: CBoW

CBoW: representing the context as Continuous Bag-of-Words

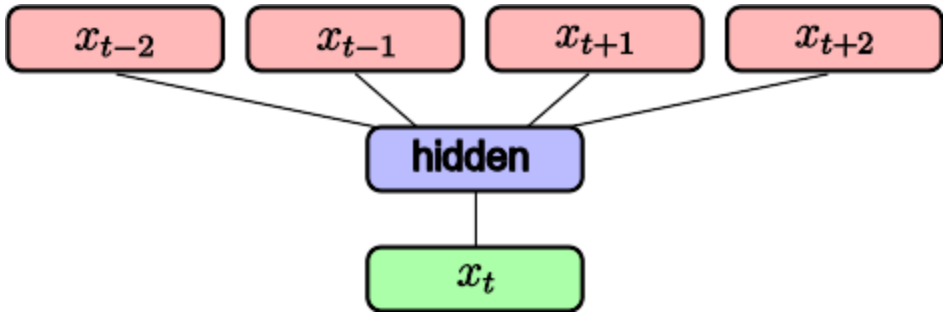
Self-supervision from large unlabeled corpus of text: *slide* over an anchor word and its context:

```
the carrot is a root vegetable, usually orang
```



Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." NIPS 2013

Word2Vec: Skip Gram



- Given the central word, predict occurrence of other words in its context.
- Widely used in practice

Word2Vec: Negative Sampling

- Task is simplified further: binary classification of word pairs
- For the sentence "The quick brown fox jumps over the lazy dog":
- "quick" and "fox" are positive examples (if context window is 2)
- "quick" and "apple" are negative examples
- By sampling negative examples, we don't just bring similar words' embeddings closer, but also push away dissimilar words' embeddings.

Transformer-based methods

- Attention mechanism: more recent and more powerful than Word2Vec
- BERT (Bidirectional Encoder Representations from Transformers) allows for contextual embeddings (different embeddings for the same word in different contexts)
- For example, "bank" in "river bank" and "bank account" will have different embeddings
- This means converting a word to a vector is no longer a simple lookup in a table, but a function of the entire sentence

Transformer-based methods

- Sub-word tokenization: BERT uses a sub-word tokenization, which allows it to handle out-of-vocabulary words better than Word2Vec
- For example, "unbelievable" can be split into "un" and "believable"
- This means that the model can guess the meaning of words it has never seen before, based on the meanings of their parts
- OpenAI tokenization example: https://platform.openai.com/tokenizer

Take Away on Embeddings

For text applications, inputs of Neural Networks are Embeddings

- If little training data and a wide vocabulary not well covered by training data, use pre-trained self-supervised embeddings (word2vec, or with more time and resources, BERT, GPT, etc.)
- If large training data with labels, directly learn task-specific embedding for more precise representation.
- word2vec uses Bag-of-Words (BoW): they ignore the order in word sequences
- Depth & non-linear activations on hidden layers are not that useful for BoW text classification.

Foundations of LLMs

Fundamental goal of language modelling: next word prediction

$$P(cat \mid the\ dog\ and\ the)$$

To generate, pick the word with highest likelihood

Early models could handle one, two words of context

Locally coherent, but longer texts quickly lose meaning

More context requires more complexity!

How do LLMs work?

We'll begin with a *user prompt*: the message that the user sends to the system.

```
User: What's the capital of France?
```

How do LLMs work?

We'll begin with a *user prompt*: the message that the user sends to the system.
Our system doesn't just receive this prompt. It also receives a *system prompt*, which primes the model to behave how we'd like.

```
System: You are a helpful assistant that provides clear, concise, and accurate answers. When answering, you always give context and explain your reasoning where appropriate.
User: What's the capital of France?
```

How do LLMs work?

We'll begin with a *user prompt*: the message that the user sends to the system.
Our system doesn't just receive this prompt. It also receives a *system prompt*, which primes the model to behave how we'd like.
Tools like ChatGPT may also supply "memories" about the user, or user-defined instructions.

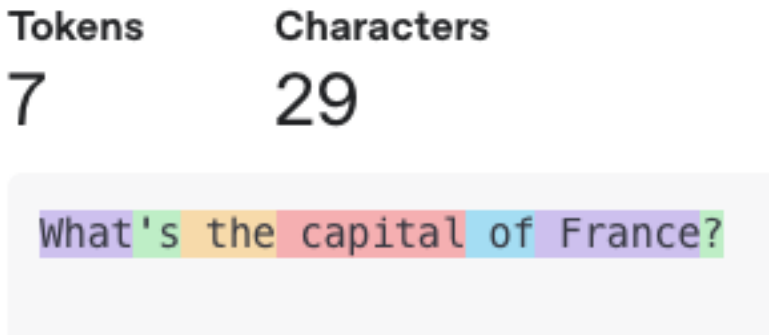
```
System: You are a helpful assistant that provides clear, concise, and accurate answers. When answering, you always give context and explain your reasoning where appropriate.
Memories:
    2024-04-08 User asked for recommendations on modern philosophy. Recommended "The History of Philosophy" by A.C. Grayling
    2024-03-15 User reported trouble installing Python libraries on a Mac. Explained how to use Pip and Homebrew to install Python packages.
User Profile:
    Name: Alex
    Profession: Senior Research Associate
    Interaction Style: professional and concise
User: What's the capital of France?
```

ChatGPT System Prompt

1. Time since user arrived on the page is 7.0 seconds.
2. User's average message length is 7201.4.
3. User is currently not using dark mode.
4. User is currently using ChatGPT in a web browser on a desktop computer.
5. User's account is 190 weeks old.
6. User's current device page dimensions are 1314x2544.
7. User's average conversation depth is 5.2.
8. User's device pixel ratio is 1.0.
9. User is currently using the following user agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:137.0) Gecko/20100101 Firefox/137.0.
10. User's current device screen dimensions are 1440x2560.
11. User is currently in Canada. This may be inaccurate if, for example, the user is using a VPN.
12. User's local hour is currently 13.

Tokenization

LLMs only know a fixed number of words
Many words are made of smaller parts
Tokenization involves breaking up words into parts if necessary

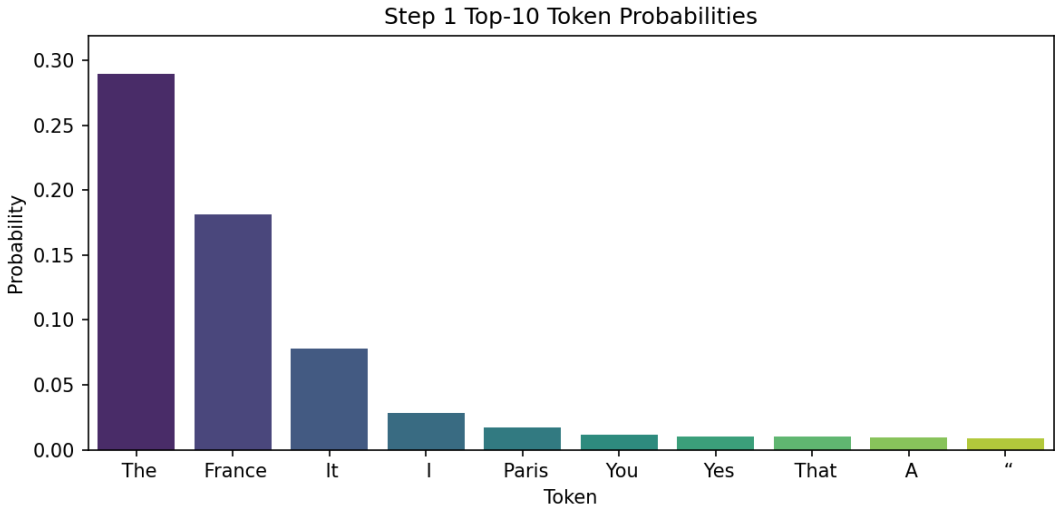


Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input

The token to show to the user is semi-randomly selected

```
User: What's the capital of France?
```



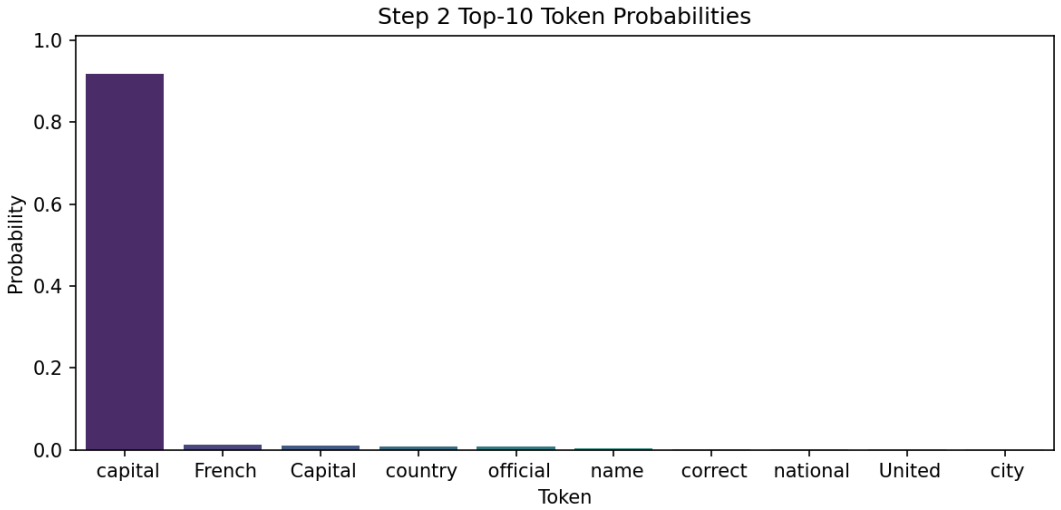

```
System: The
```

Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input

The token to show to the user is semi-randomly selected

```
User: What's the capital of France?
```

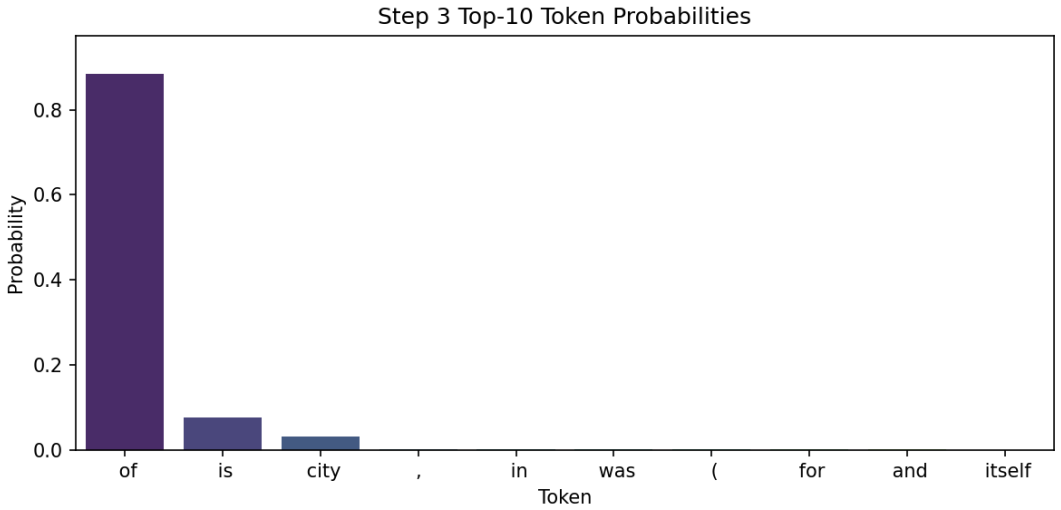



```
System: The capital
```

Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
The token to show to the user is semi-randomly selected

```
User: What's the capital of France?
```



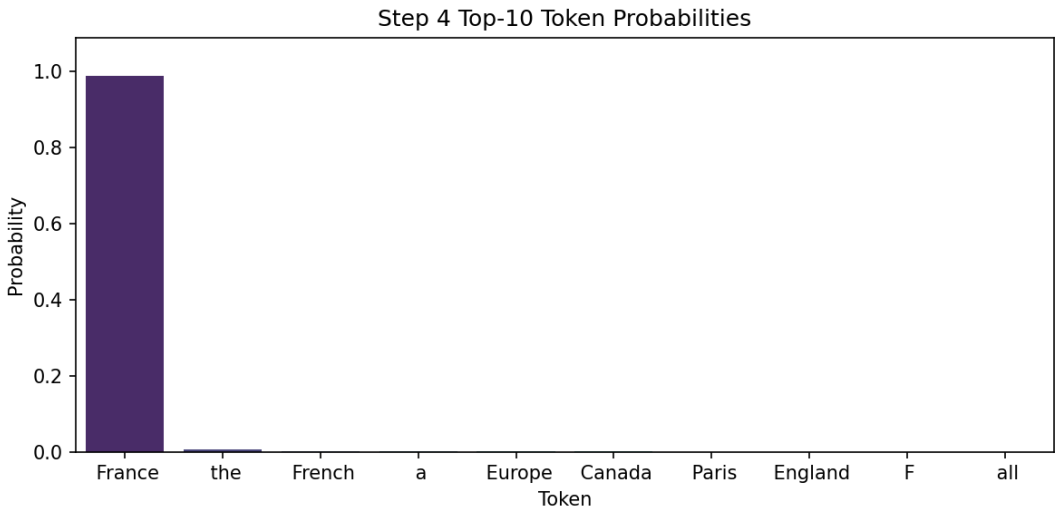

```
System: The capital of
```

Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input

The token to show to the user is semi-randomly selected

```
User: What's the capital of France?
```



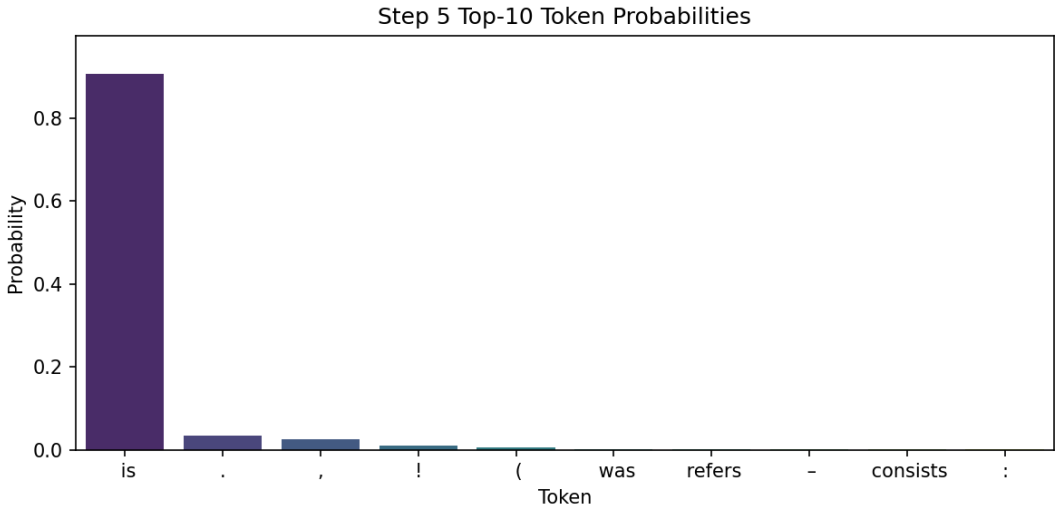

```
System: The capital of France
```

Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input

The token to show to the user is semi-randomly selected

`User: What's the capital of France?`



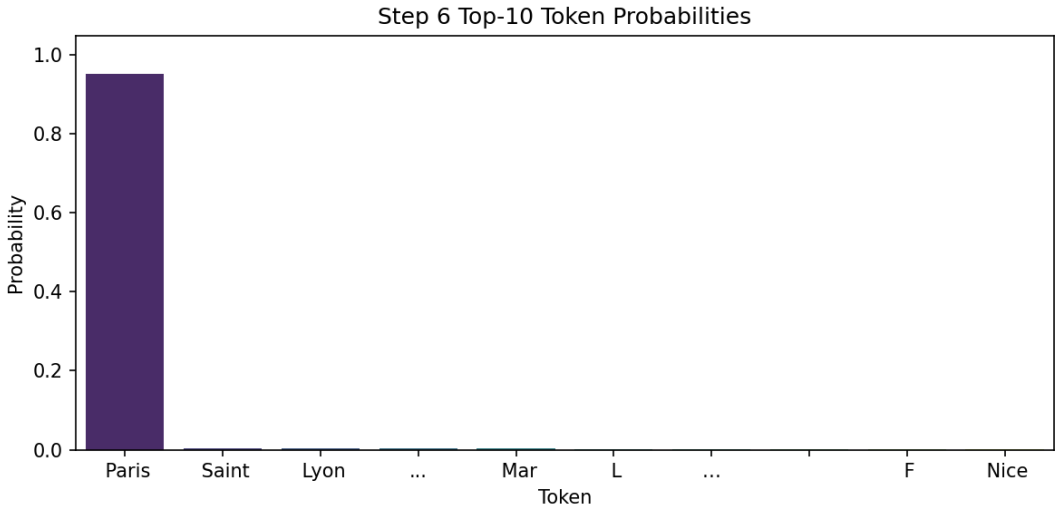
`System: The capital of France is`

Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input

The token to show to the user is semi-randomly selected

```
User: What's the capital of France?
```

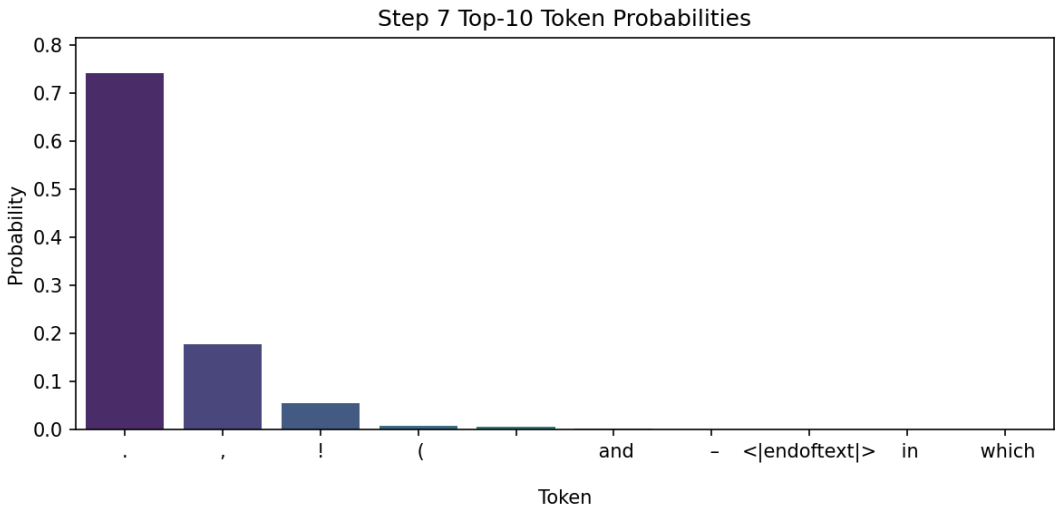



```
System: The capital of France is Paris
```

Text Generation

LLMs generate text by estimating the probability of *each token in the vocabulary* coming next after all the input
The token to show to the user is semi-randomly selected

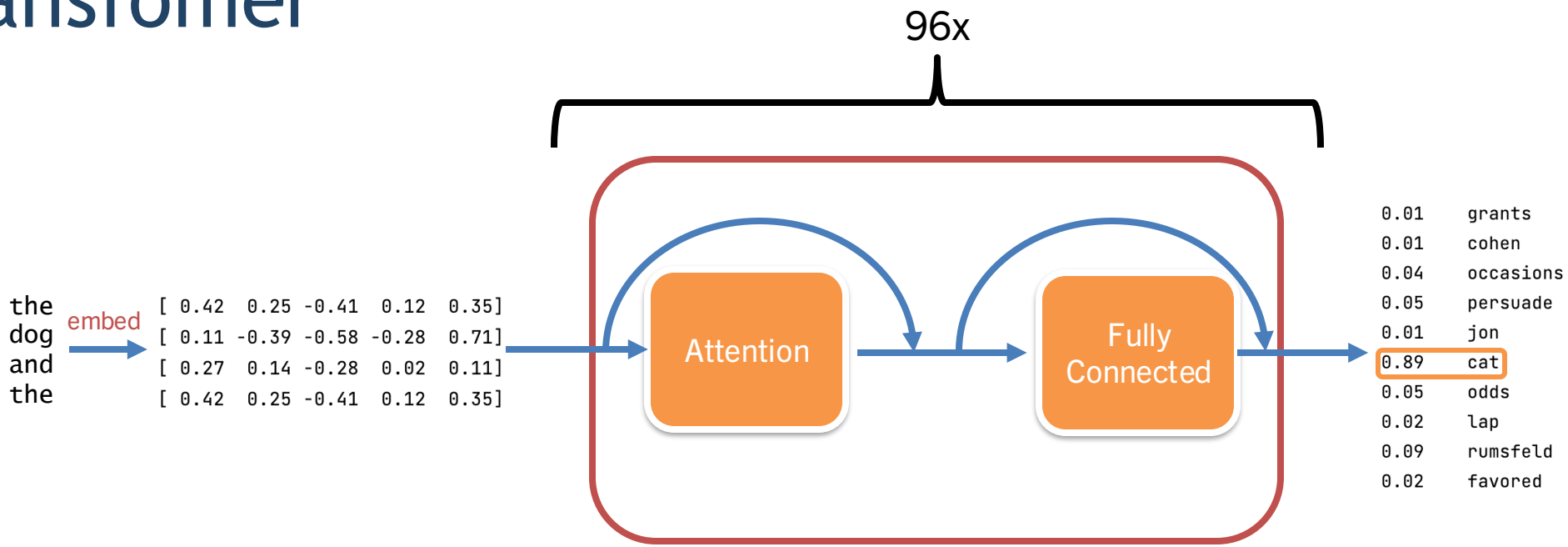
```
User: What's the capital of France?
```



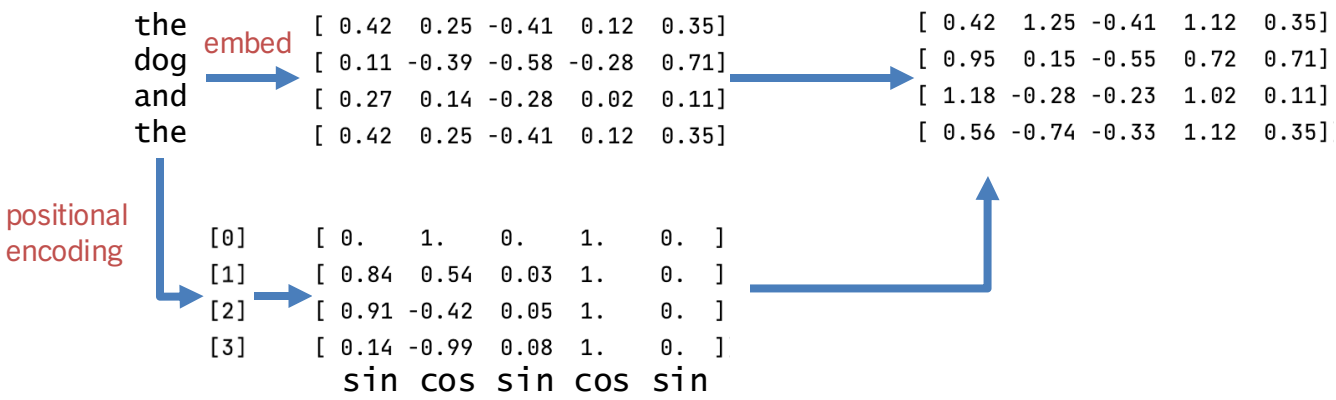

```
System: The capital of France is Paris.
```

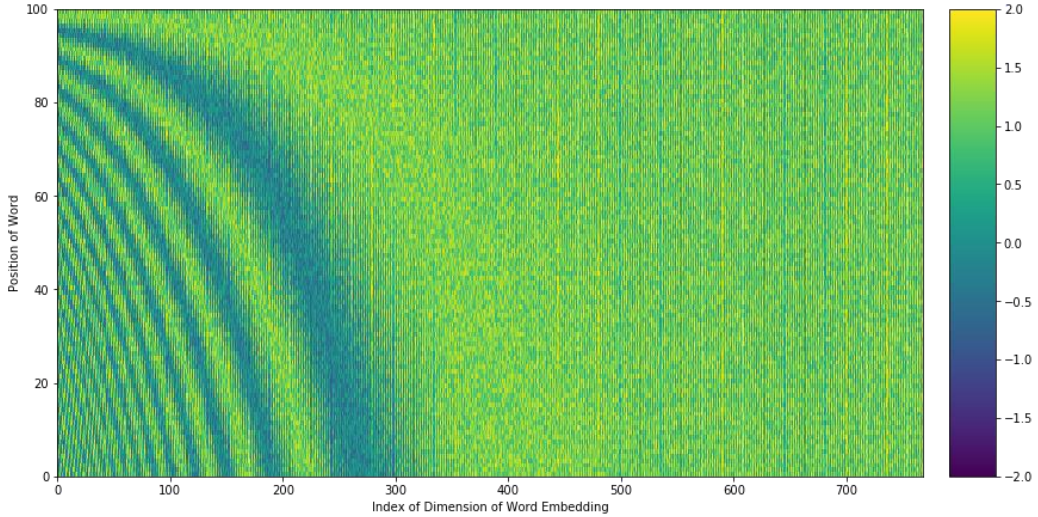
Building GPT

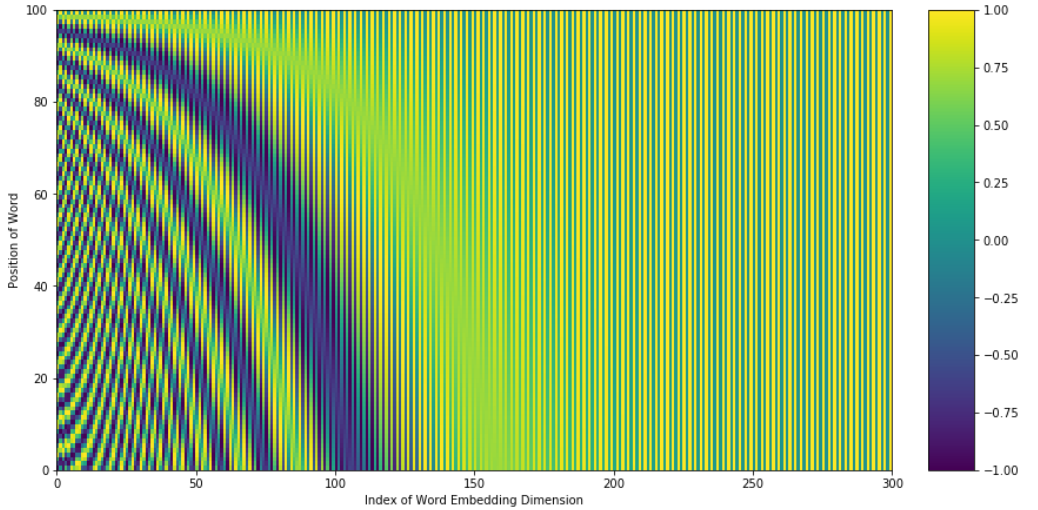
Building GPT: The Transfomer



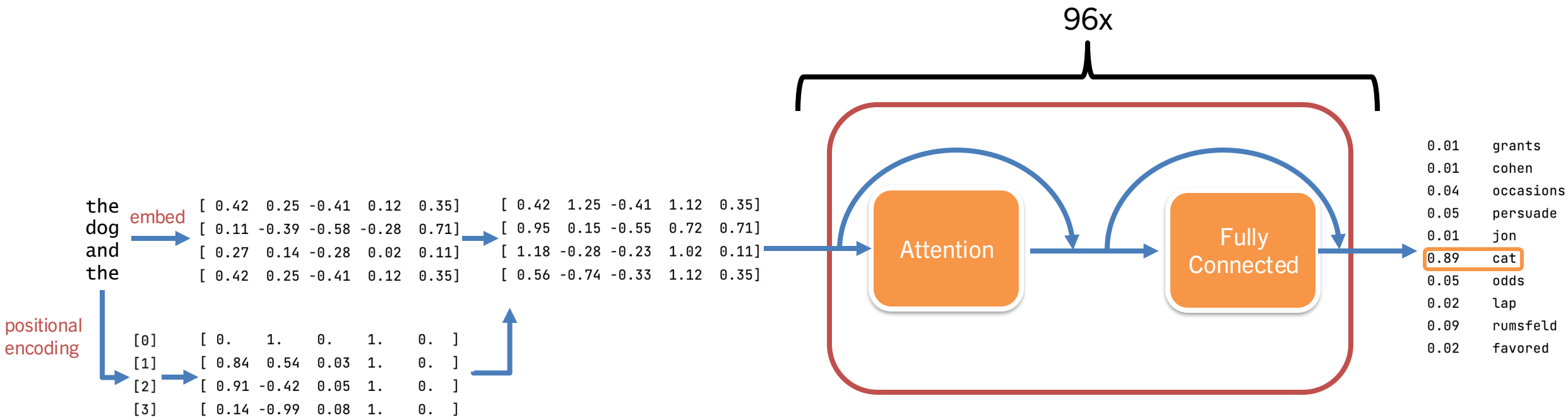
Building GPT: Positional Embedding



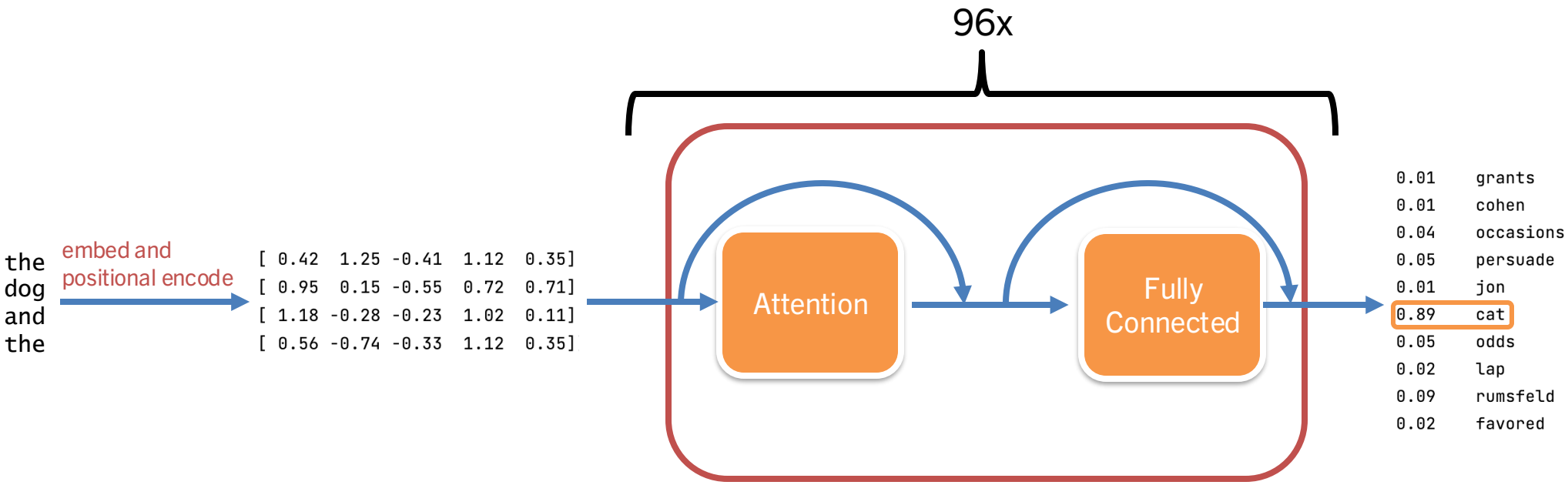




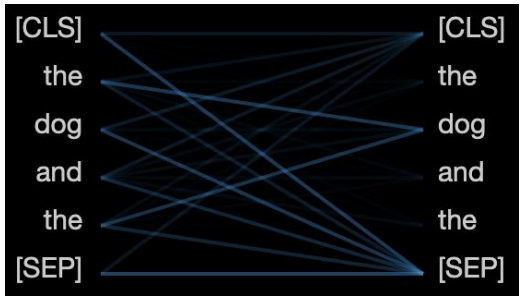
Building GPT



Building GPT



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

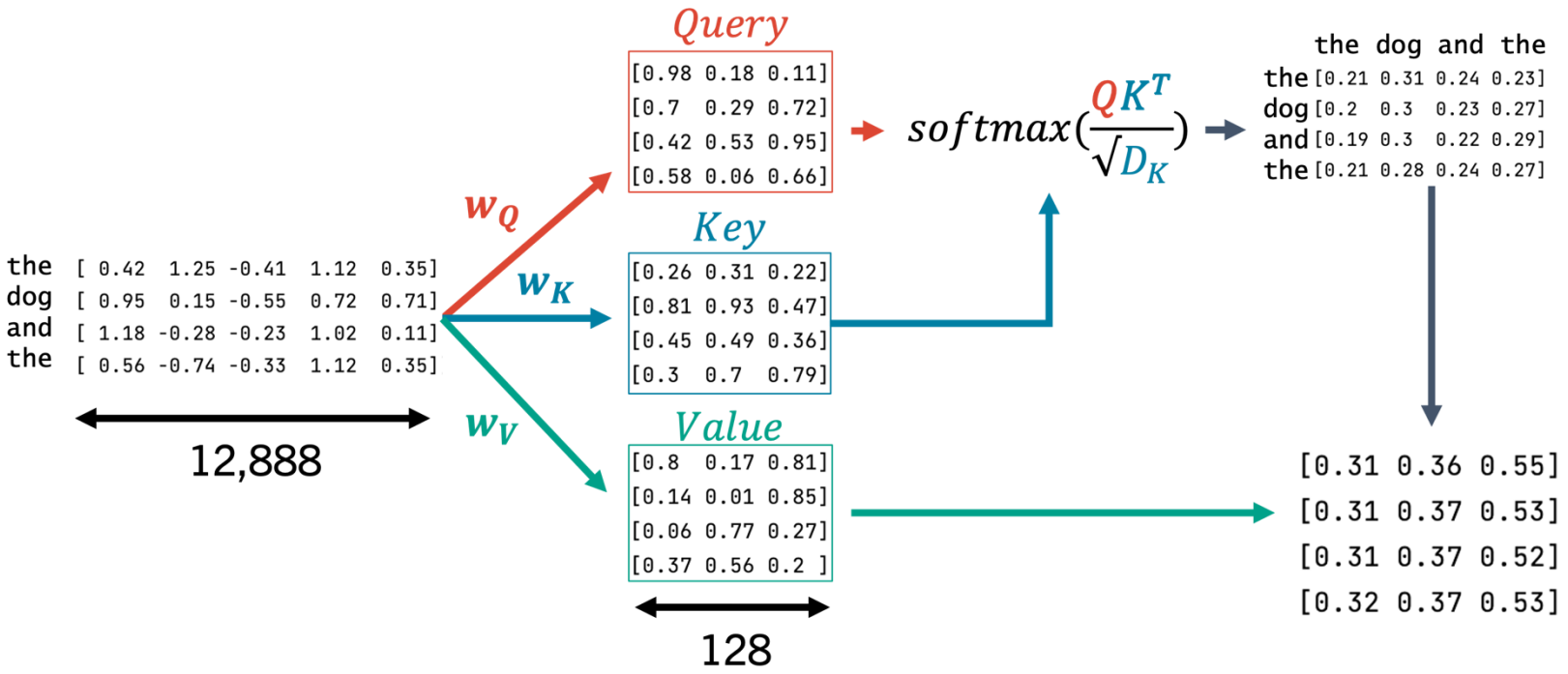
[0.8 0.17 0.81]
[0.14 0.01 0.85]
[0.06 0.77 0.27]
[0.37 0.56 0.2]

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

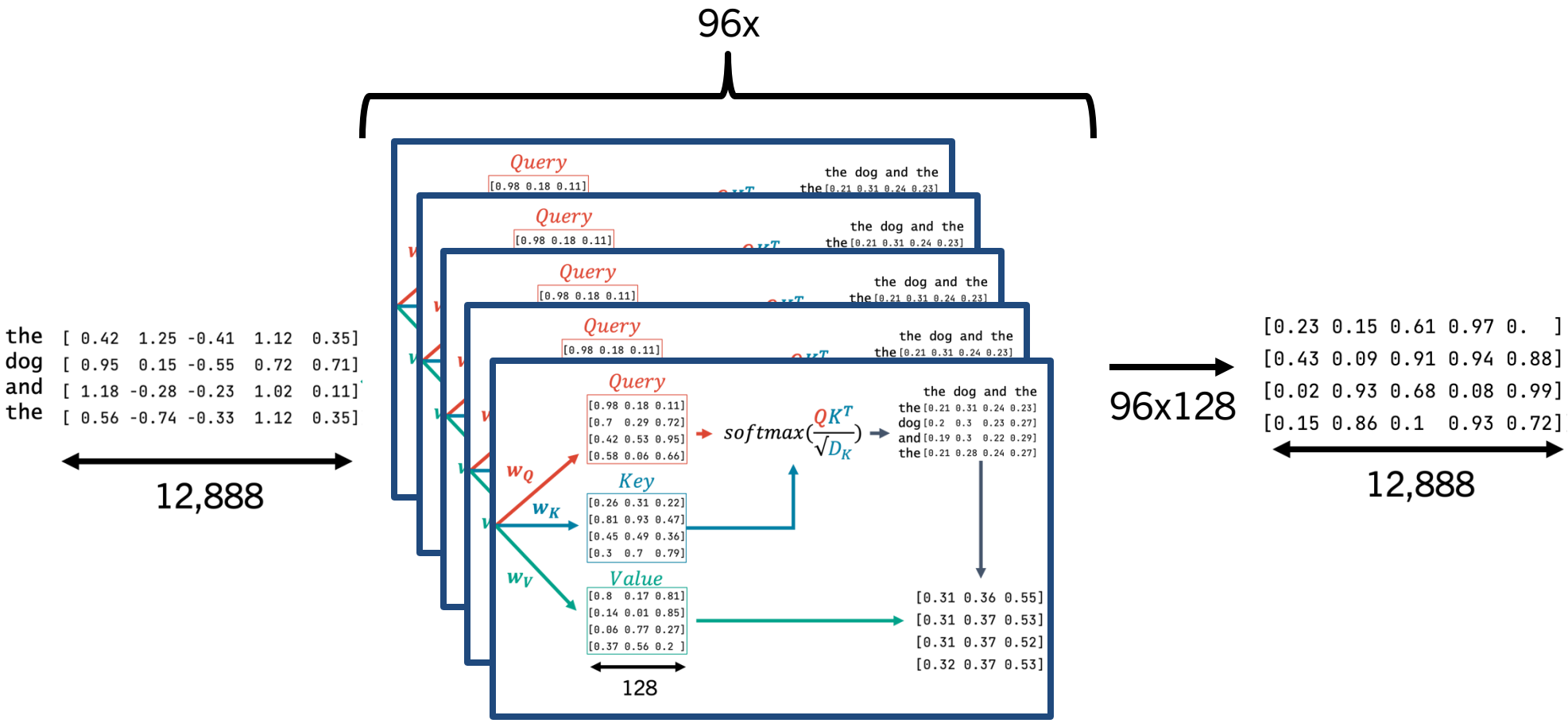
	the	dog	and	the
the	[0.21	0.31	0.24	0.23]
dog	[0.2	0.3	0.23	0.27]
and	[0.19	0.3	0.22	0.29]
the	[0.21	0.28	0.24	0.27]

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

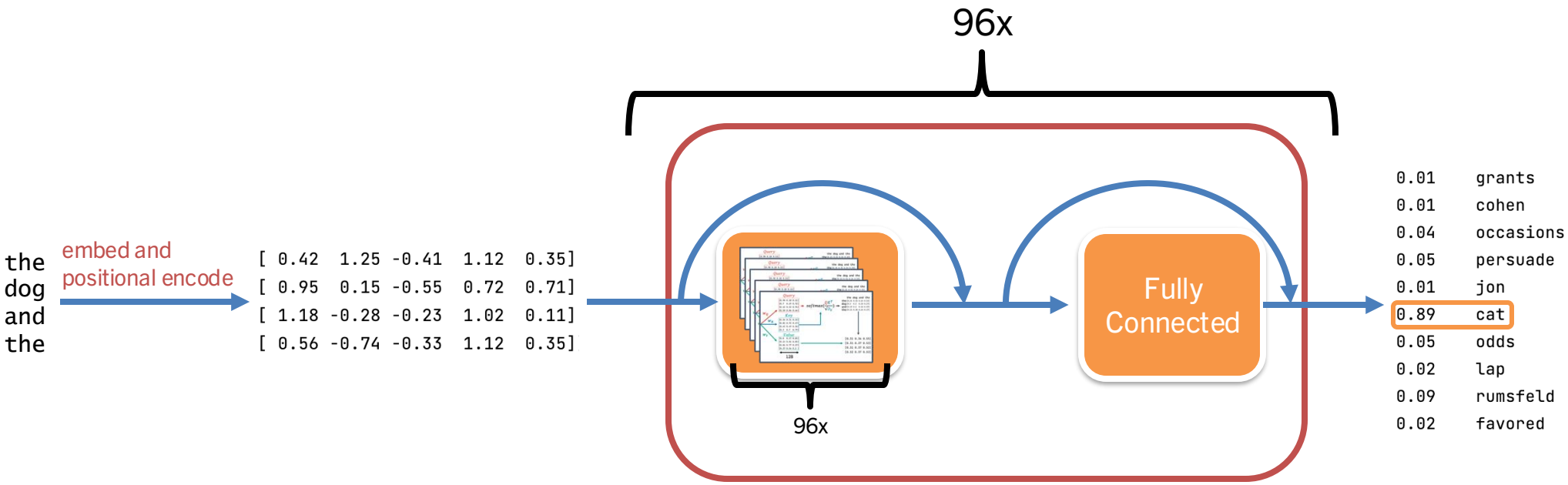
Building GPT: Attention



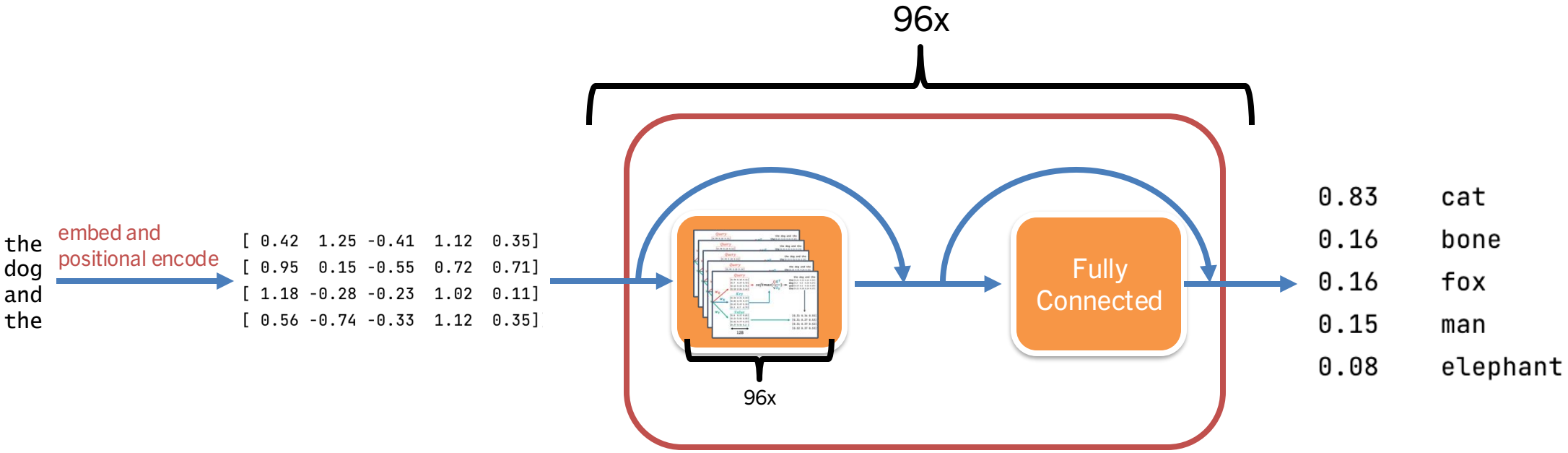
Building GPT: Attention



Building GPT



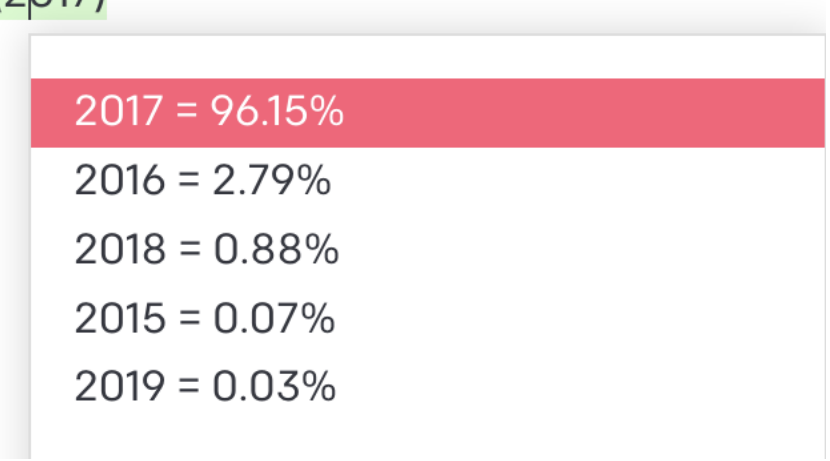
Building GPT: Top-P



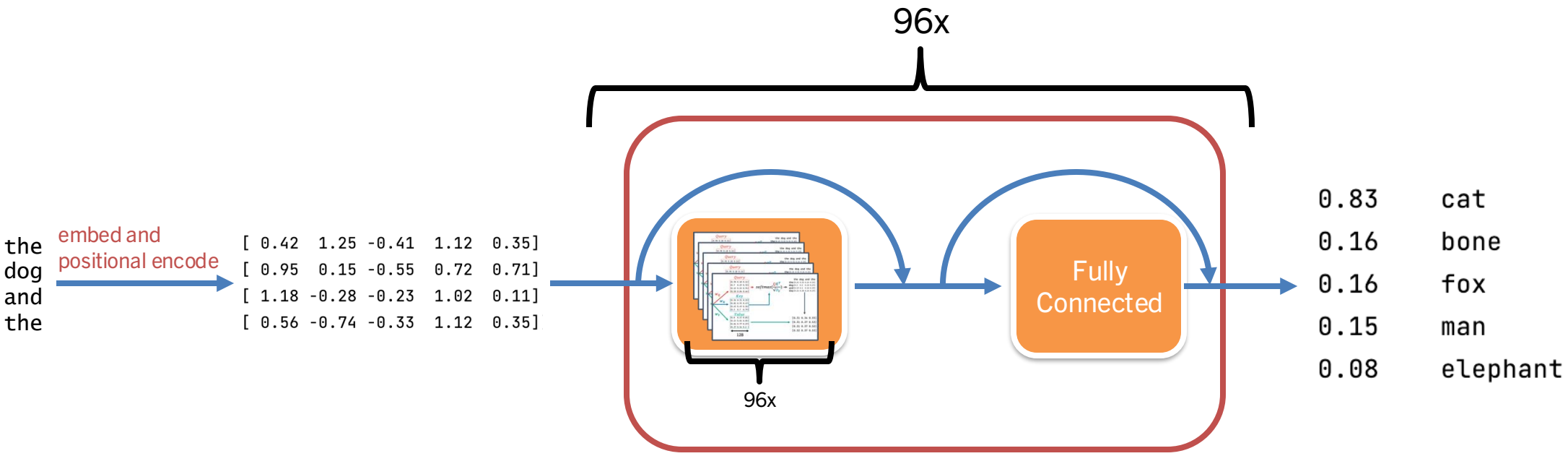
Building GPT: Top-P

Top 10 documentaries about artificial intelligence:

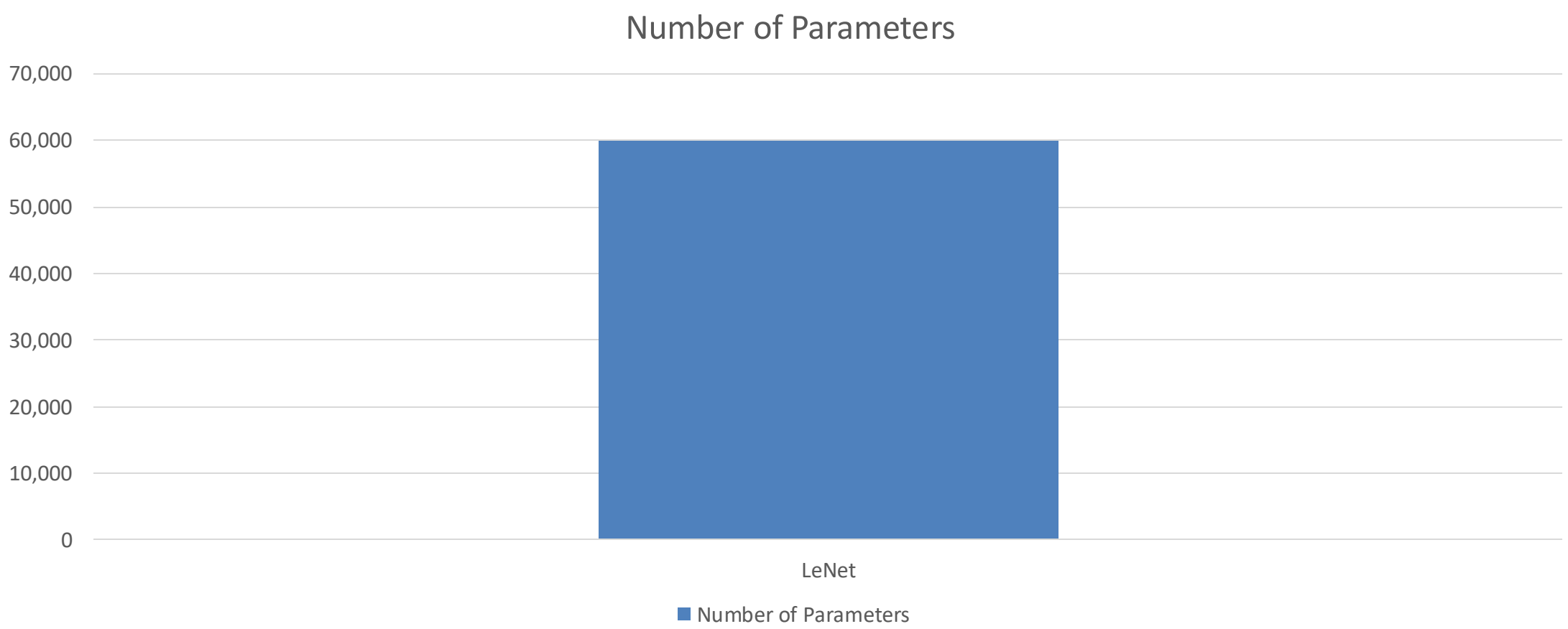
1. AlphaGo (2017)



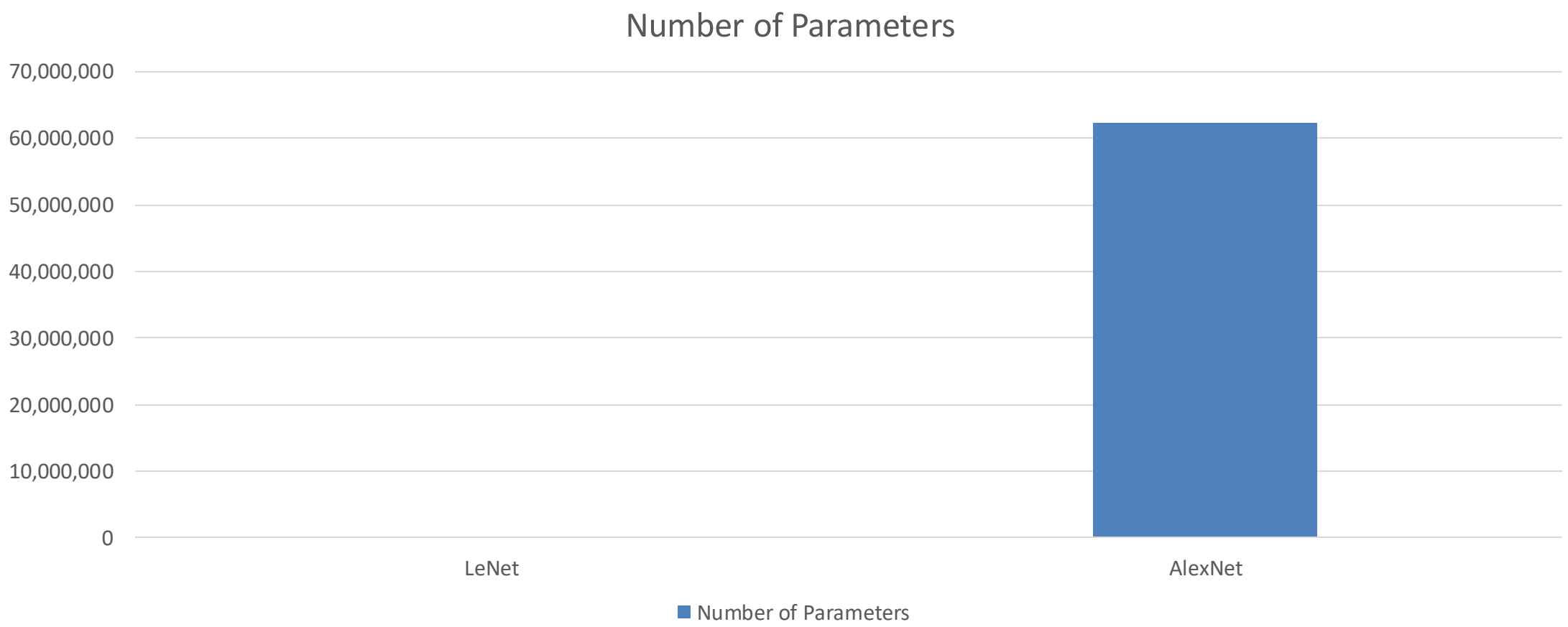
Building GPT



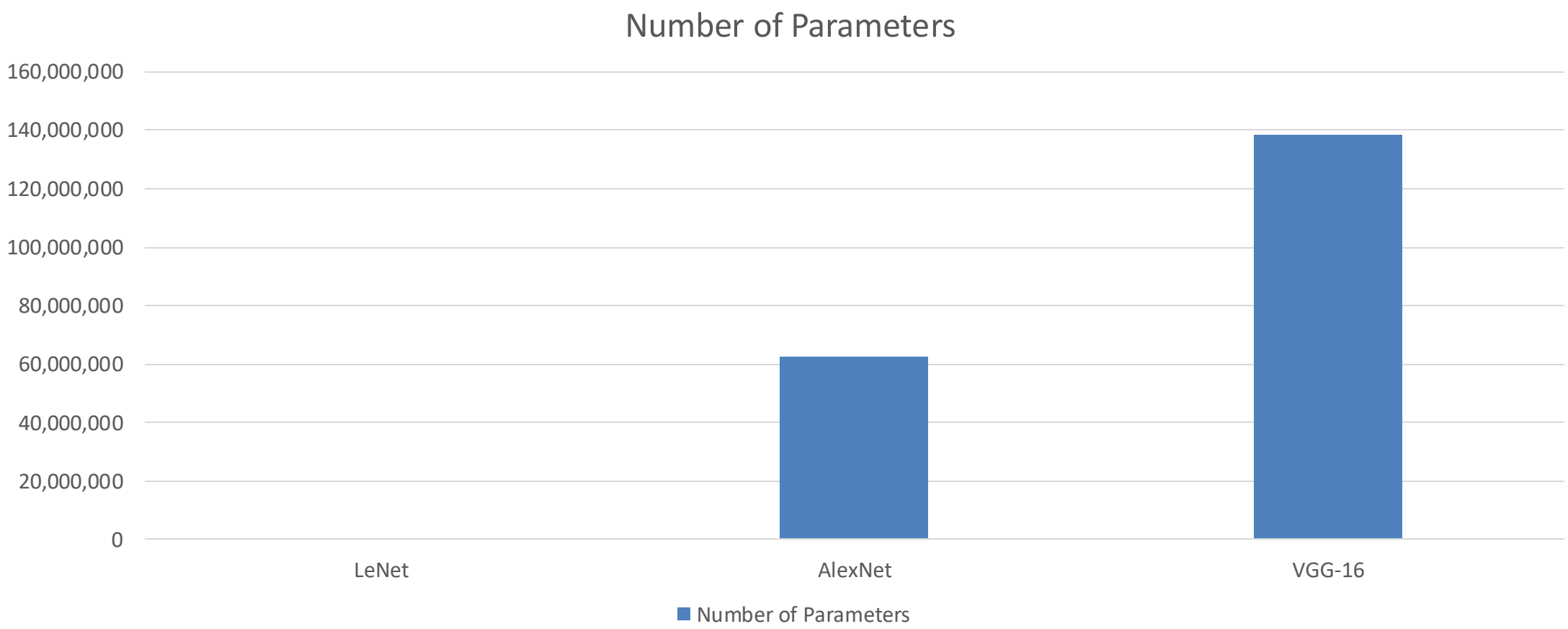
Scale of GPT


Number of Parameters
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
LeNet
Number of Parameters

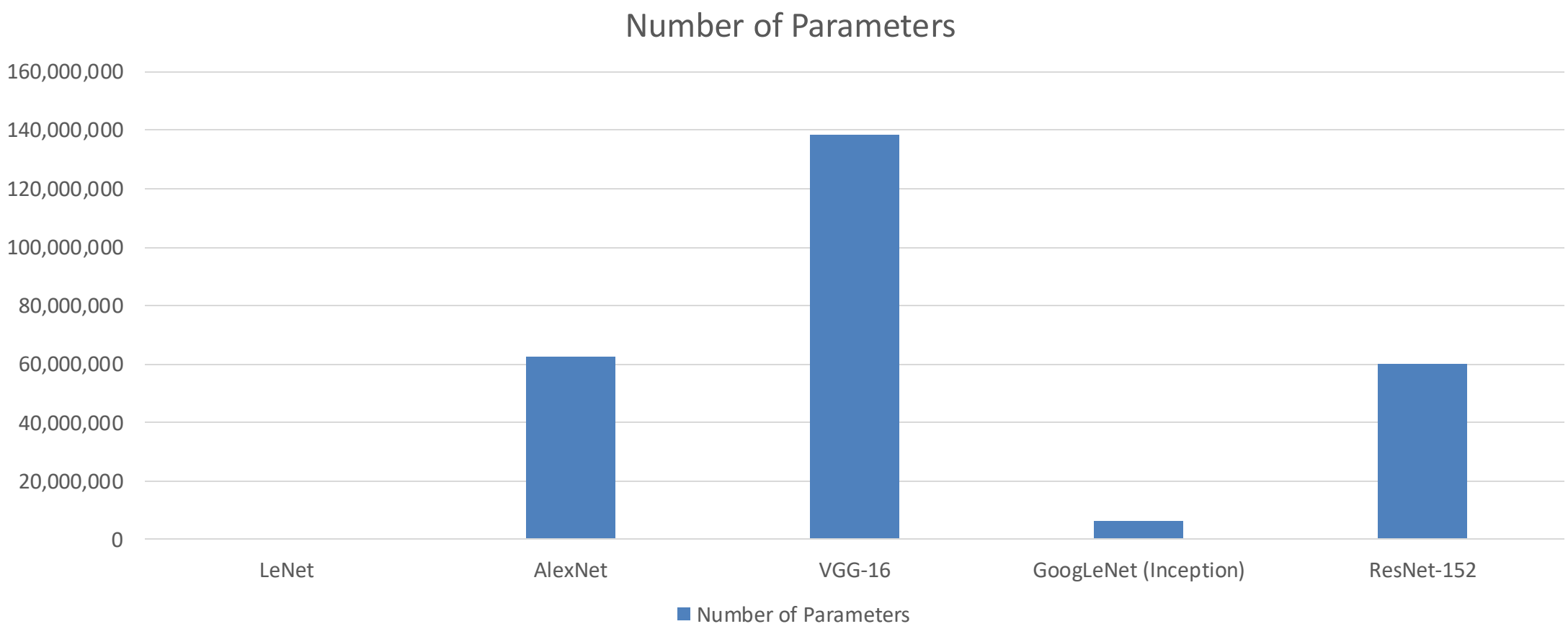
Scale of GPT


Number of Parameters
70,000,000
60,000,000
50,000,000
40,000,000
30,000,000
20,000,000
10,000,000
0
LeNet
AlexNet
Number of Parameters

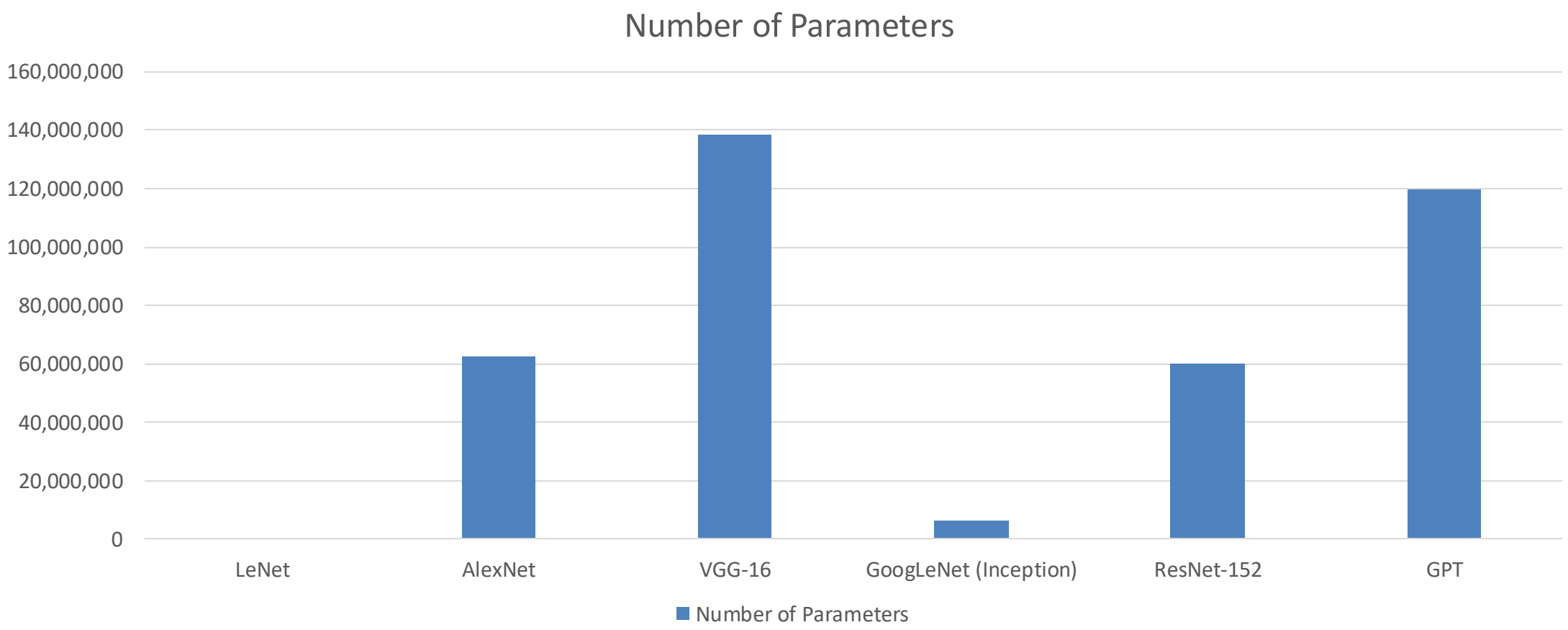
Scale of GPT


Number of Parameters
160,000,000
140,000,000
120,000,000
100,000,000
80,000,000
60,000,000
40,000,000
20,000,000
0
LeNet
AlexNet
VGG-16
Number of Parameters

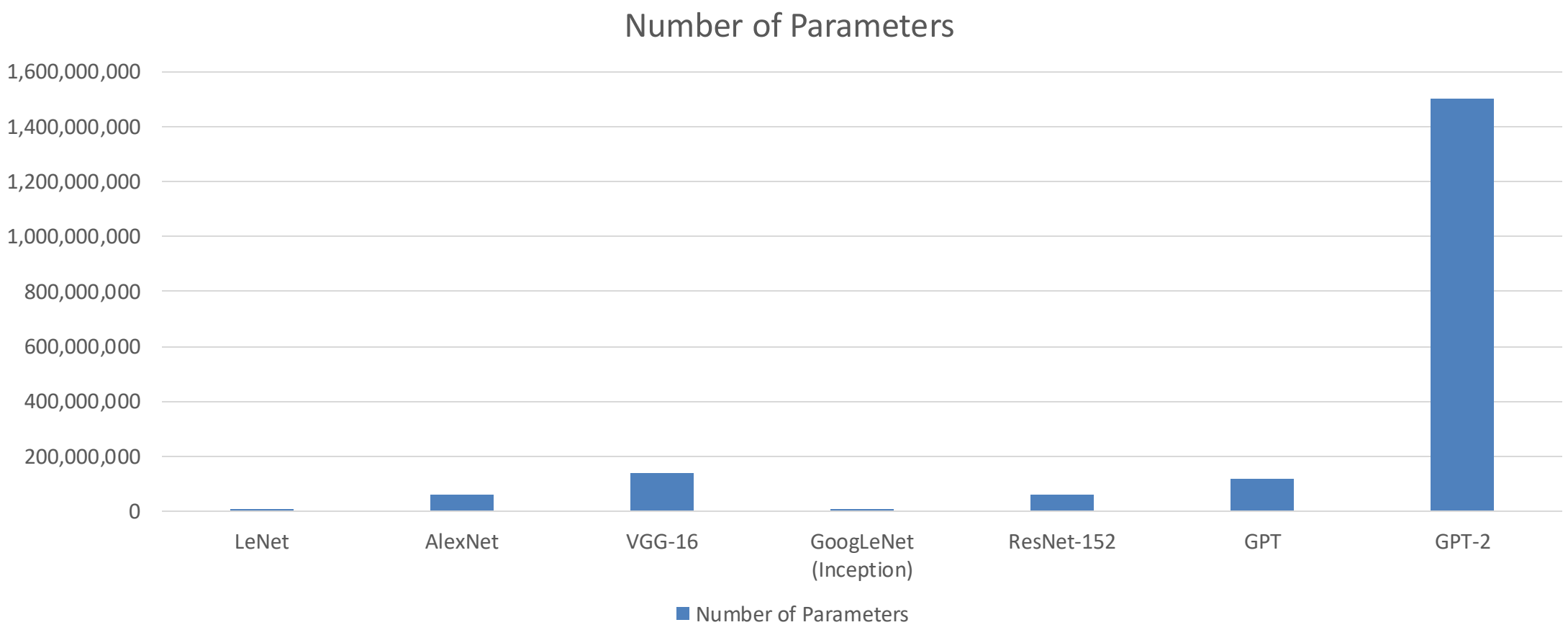
Scale of GPT


Number of Parameters
160,000,000
140,000,000
120,000,000
100,000,000
80,000,000
60,000,000
40,000,000
20,000,000
0
LeNet
AlexNet
VGG-16
GoogLeNet (Inception)
ResNet-152
Number of Parameters

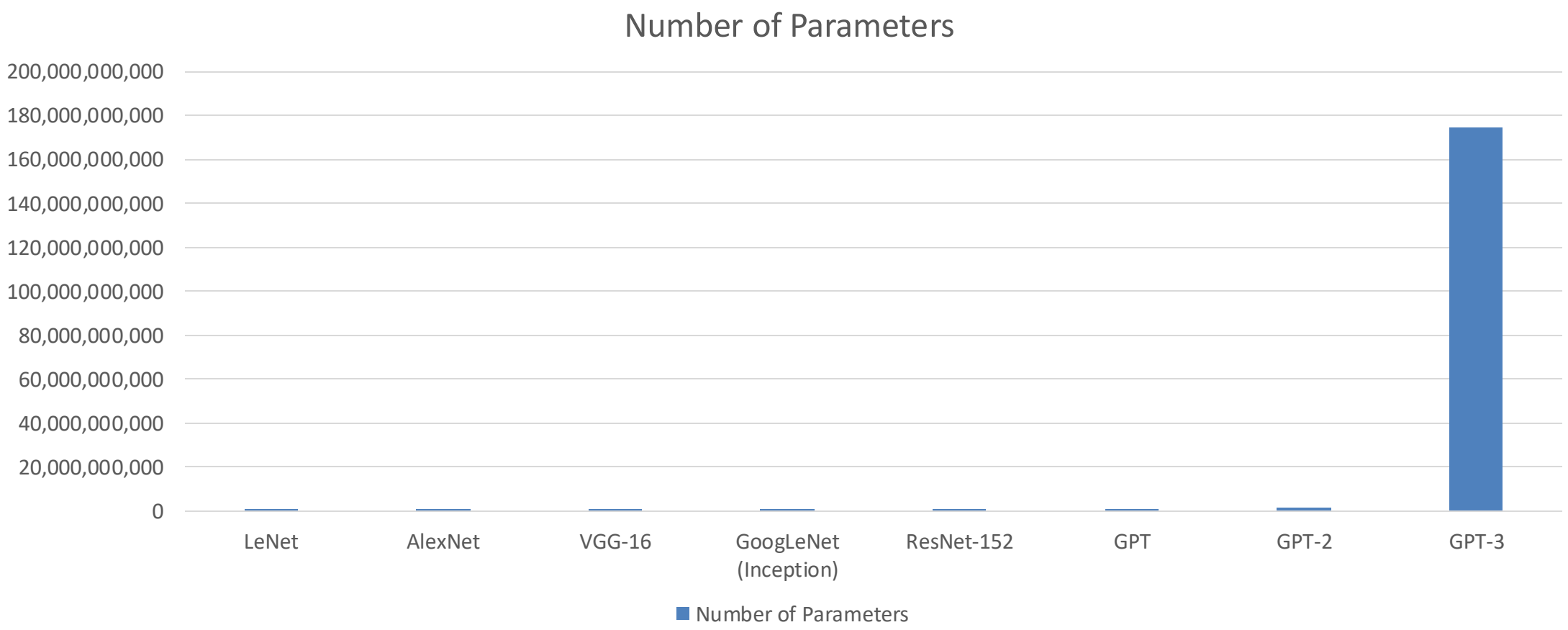
Scale of GPT


Number of Parameters
160,000,000
140,000,000
120,000,000
100,000,000
80,000,000
60,000,000
40,000,000
20,000,000
0
LeNet
AlexNet
VGG-16
GoogLeNet (Inception)
ResNet-152
GPT
Number of Parameters

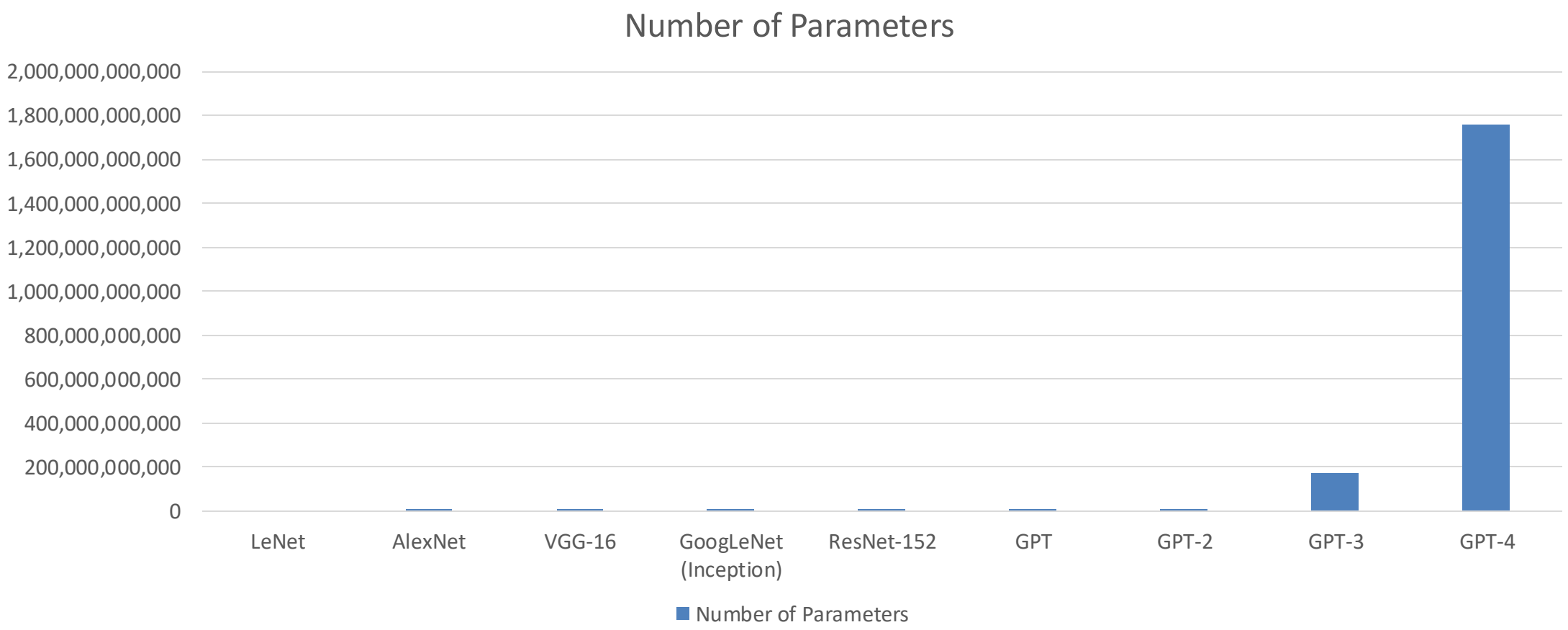
Scale of GPT


Number of Parameters
1,600,000,000
1,400,000,000
1,200,000,000
1,000,000,000
800,000,000
600,000,000
400,000,000
200,000,000
0
LeNet
AlexNet
VGG-16
GoogLeNet (Inception)
ResNet-152
GPT
GPT-2
Number of Parameters

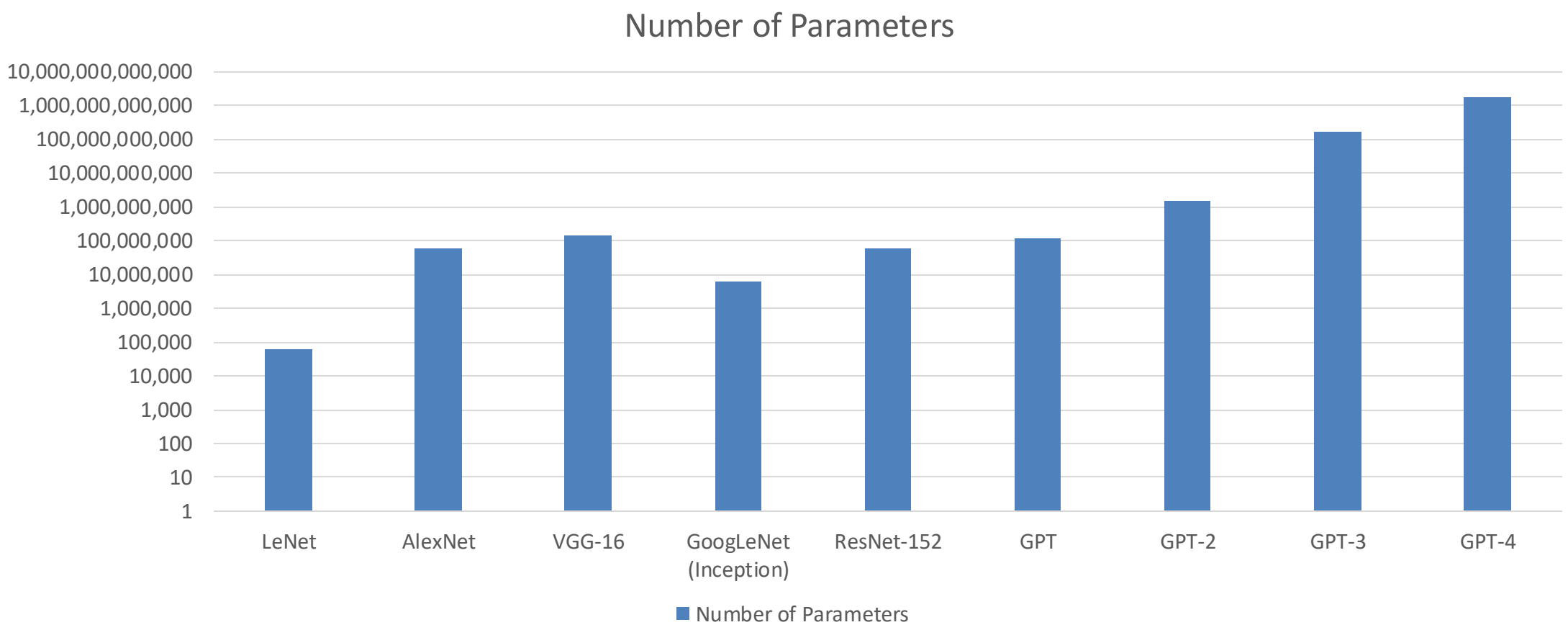
Scale of GPT


Number of Parameters
200,000,000,000
180,000,000,000
160,000,000,000
140,000,000,000
120,000,000,000
100,000,000,000
80,000,000,000
60,000,000,000
40,000,000,000
20,000,000,000
0
LeNet
AlexNet
VGG-16
GoogLeNet (Inception)
ResNet-152
GPT
GPT-2
GPT-3
Number of Parameters

Scale of GPT


Number of Parameters
2,000,000,000,000
1,800,000,000,000
1,600,000,000,000
1,400,000,000,000
1,200,000,000,000
1,000,000,000,000
800,000,000,000
600,000,000,000
400,000,000,000
200,000,000,000
0
LeNet
AlexNet
VGG-16
GoogLeNet (Inception)
ResNet-152
GPT
GPT-2
GPT-3
GPT-4
Number of Parameters

Scale of GPT


Number of Parameters
10,000,000,000,000
1,000,000,000,000
100,000,000,000
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
100
10
1
LeNet
AlexNet
VGG-16
GoogLeNet (Inception)
ResNet-152
GPT
GPT-2
GPT-3
GPT-4
Number of Parameters

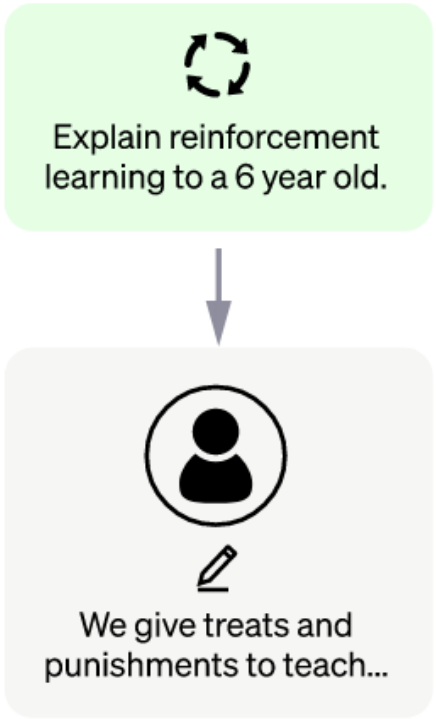
GPT's Training Data

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

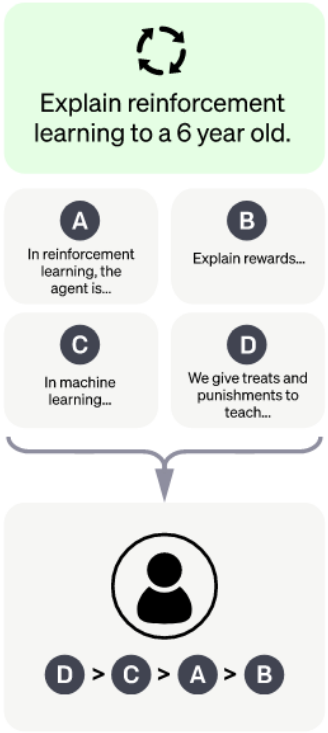
The training innovation of ChatGPT

Human annotators write answers to questions



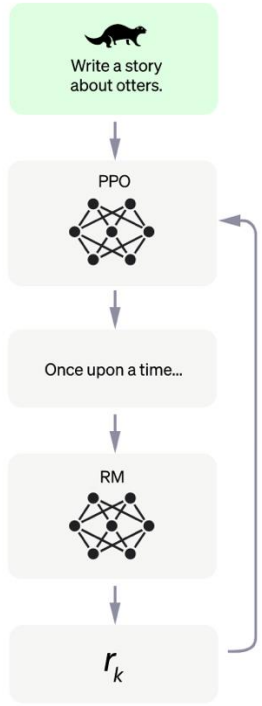
The generalist GPT model is taught from these Q&A pairs

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



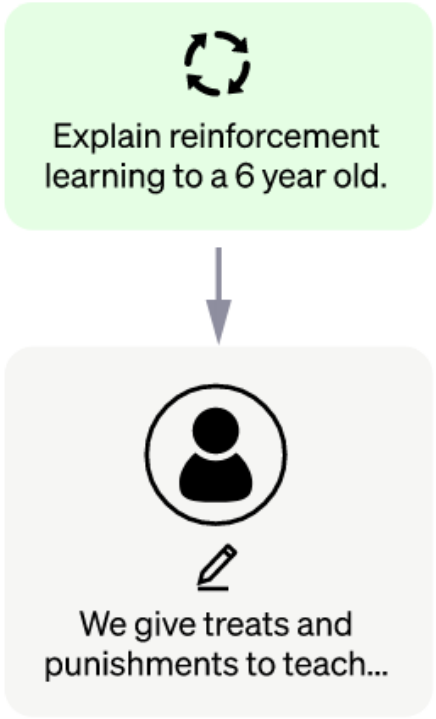
The reward model rates each answer, allowing GPT to keep learning

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

The training innovation of ChatGPT

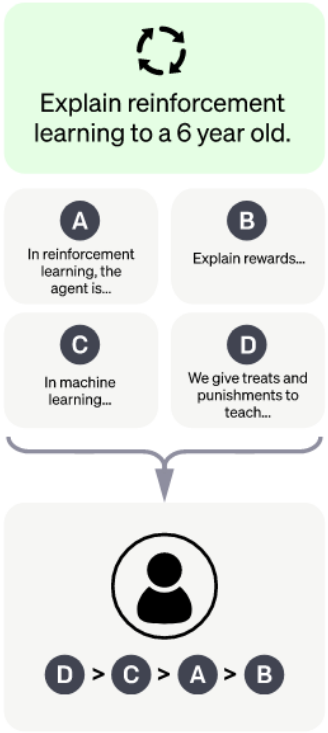
No more humans involved!

Human annotators write answers to questions



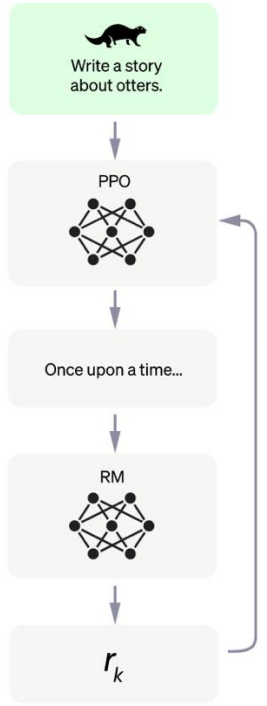
The generalist GPT model is taught from these Q&A pairs

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/