RADONC AI CURRICULUM

Introduction to Deep Learning

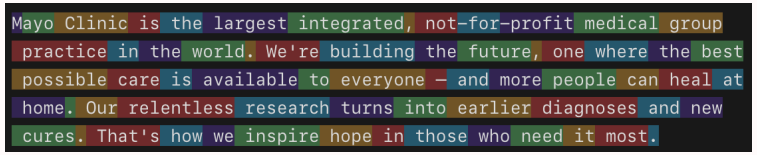
LECTURE 7: Deep Learning for Language

ANDREW Y. K. FOONG, PH.D.

May 29th, 2026

MAYO CLINIC

Radiation Oncology
AI & Data Analytics
AIDA



Today's lecture

1. Representing language
   - *From words to numbers*
2. Tokenization
   - *Representing language efficiently*
3. Byte pair encoding
   - *The standard approach to tokenization*
4. Generating text
   - *Next-token prediction*
5. LLM training data
   - *Scraping the Internet*
6. Learning from the Internet
   - *Predictive distributions*
7. Q&A

Large language models from 40,000 feet

- ChatGPT, Claude, Gemini, Copilot, DeepSeek…
- Still deep neural networks!
- Plan for LLMs:
    1. Represent words as (many) numbers.
    2. Generate sentences by predicting next word.
    3. Train on data from Internet.
    4. Specialized neural network architecture for text.
    5. Steer network to be helpful and accurate.

Large language models from 40,000 feet

- ChatGPT, Claude, Gemini, Copilot, DeepSeek…
- Still deep neural networks!
- Plan for LLMs:
  1. Represent words as (many) numbers.
  2. Generate sentences by predicting next word.
  3. Train on data from Internet.
  4. Specialized neural network architecture for text.
  5. Steer network to be helpful and accurate.

Representing language

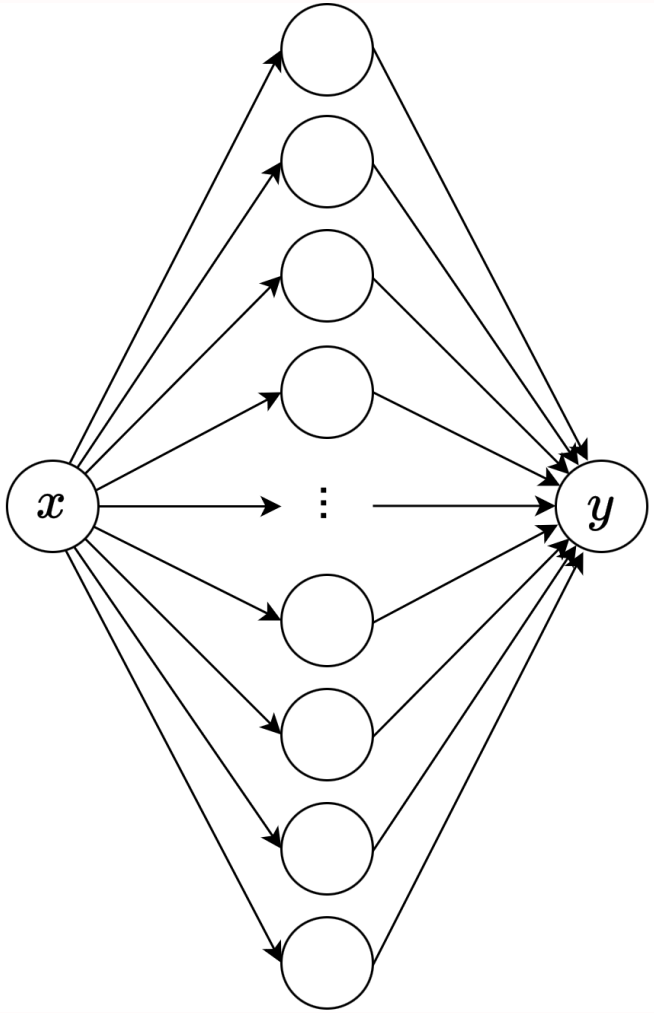
From words to numbers

From words to numbers

- Neural networks process numbers:

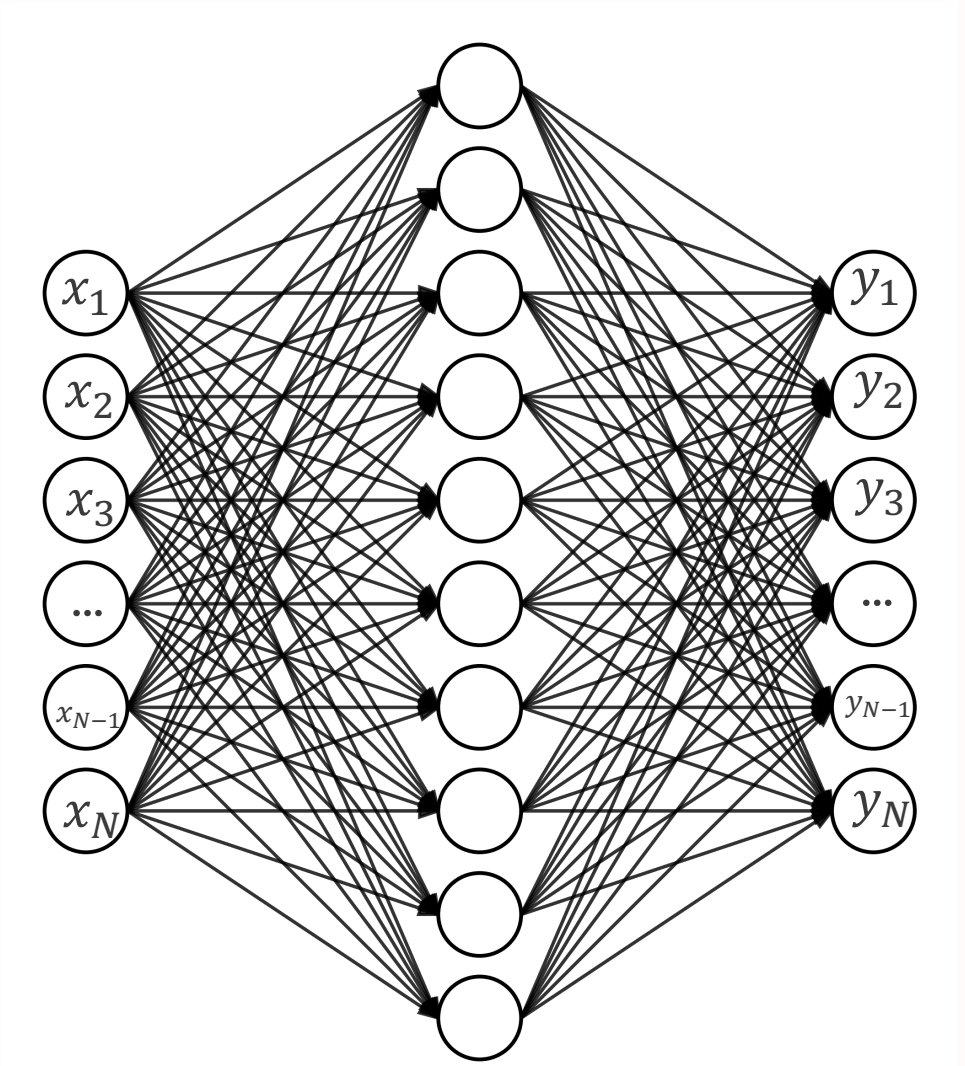
$$f(x) = y$$

- x and y are numbers.



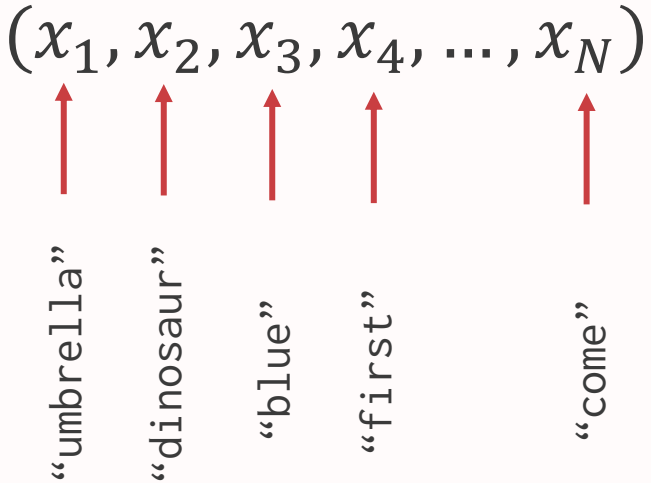
From words to numbers

- Neural networks process numbers:
$$f(x) = y$$
- x and y are numbers.
- Multiple numbers possible too—vectors:
$$f(x_1, \ldots, x_N) = y_1, \ldots, y_N$$
- How to input and output words?



From words to numbers

- Assume dictionary has N different words:
  - Represent each word as N numbers.
  - Each "slot" assigned to each word.



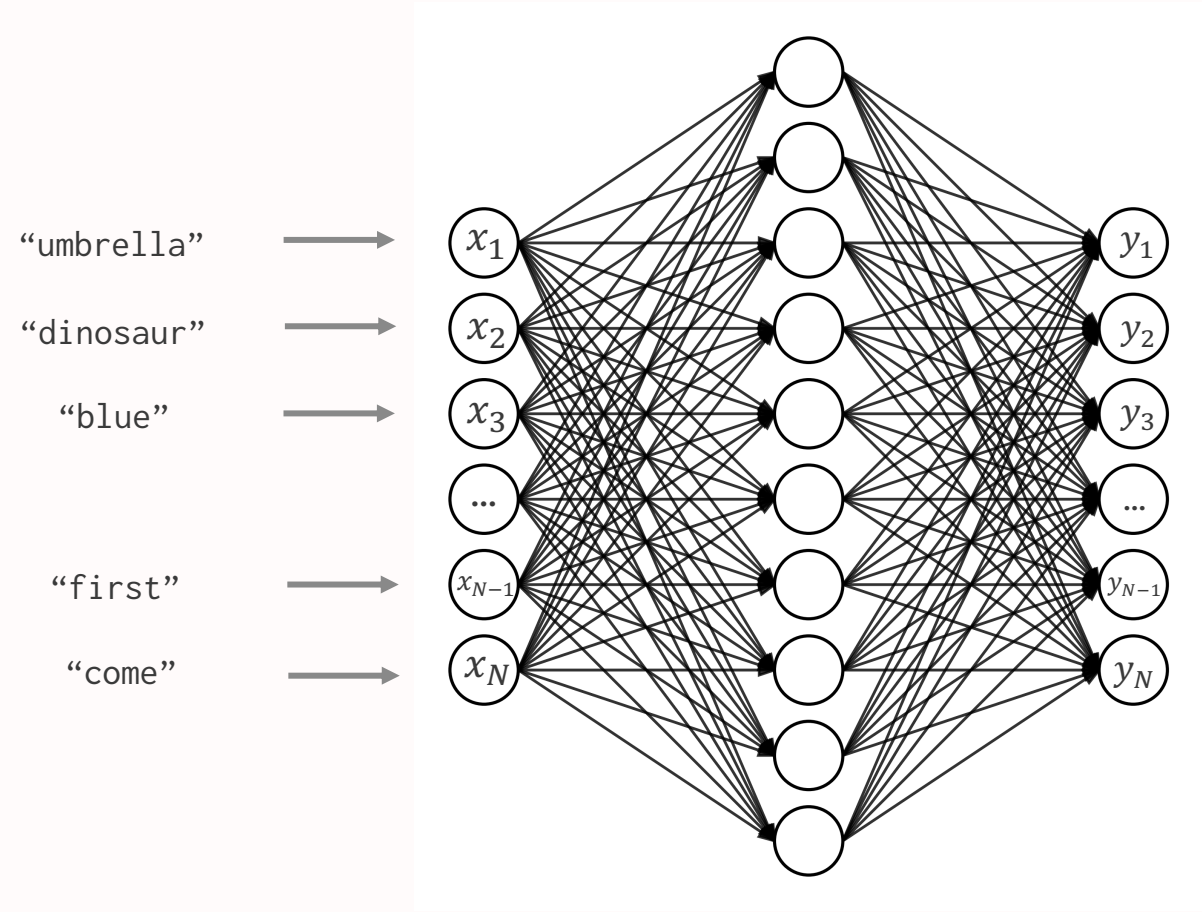
  - Each word represented by 1 in corresponding slot, 0 elsewhere:

$$\text{"blue"} \rightarrow (0, 0, 1, 0, \ldots, 0)$$

  - Called one-hot encoding.
  - Each neural network input is called a token.

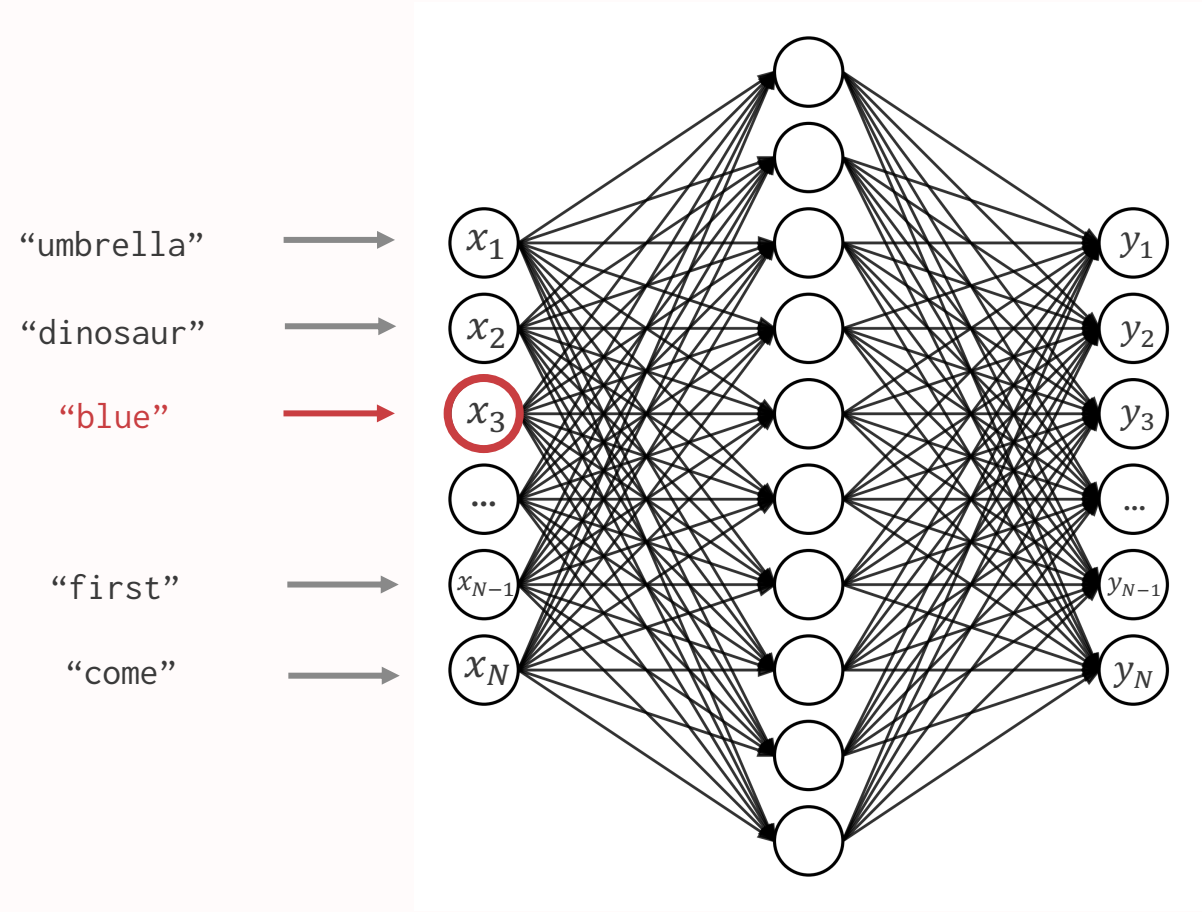
From words to numbers

EXAMPLE: saying the word “blue” to a neural network



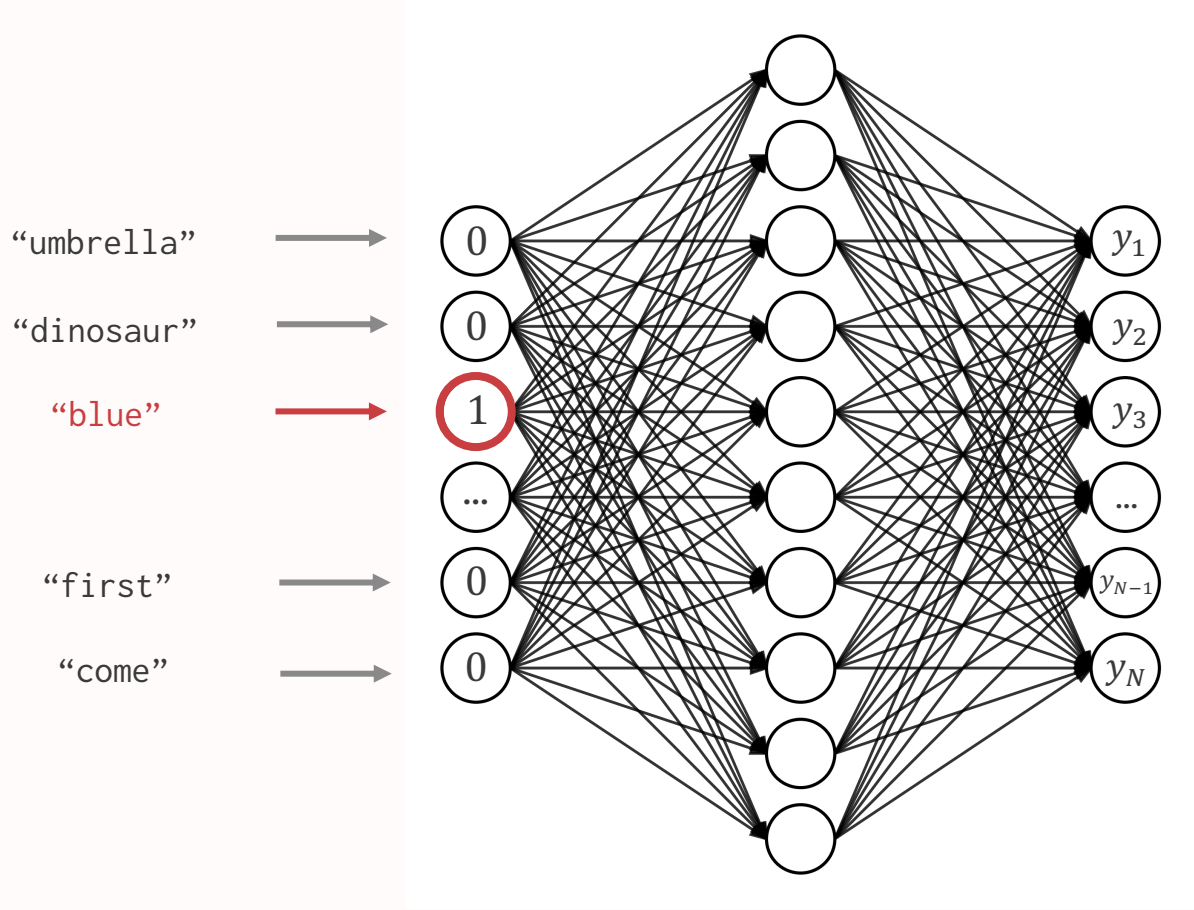
From words to numbers

EXAMPLE: saying the word “blue” to a neural network



From words to numbers

EXAMPLE: saying the word “blue” to a neural network

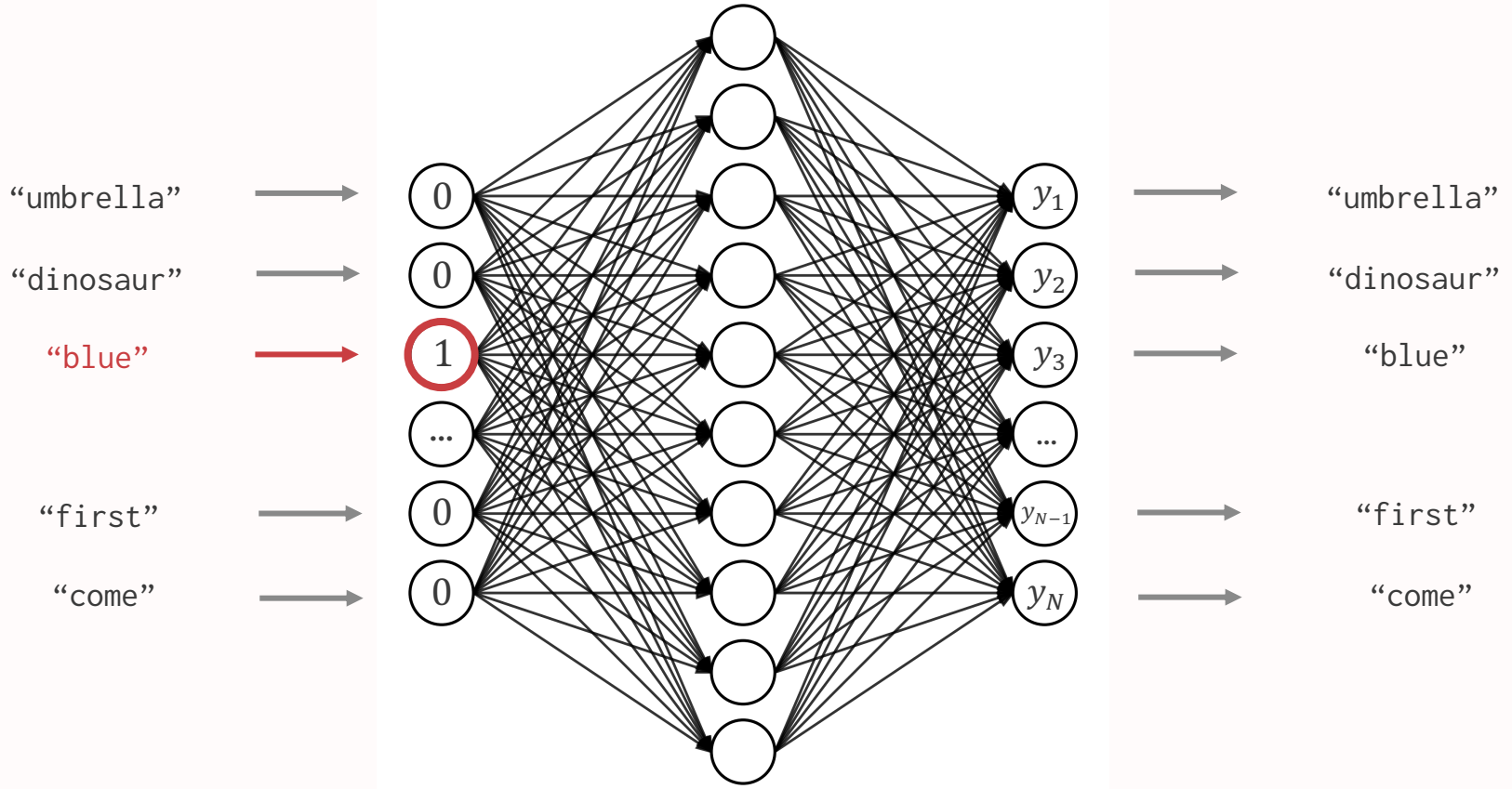


From words to numbers

- We now know how to input words.
  - “Speaking to the neural network”
- How can we output words?
  - “Neural network speaks back to us”

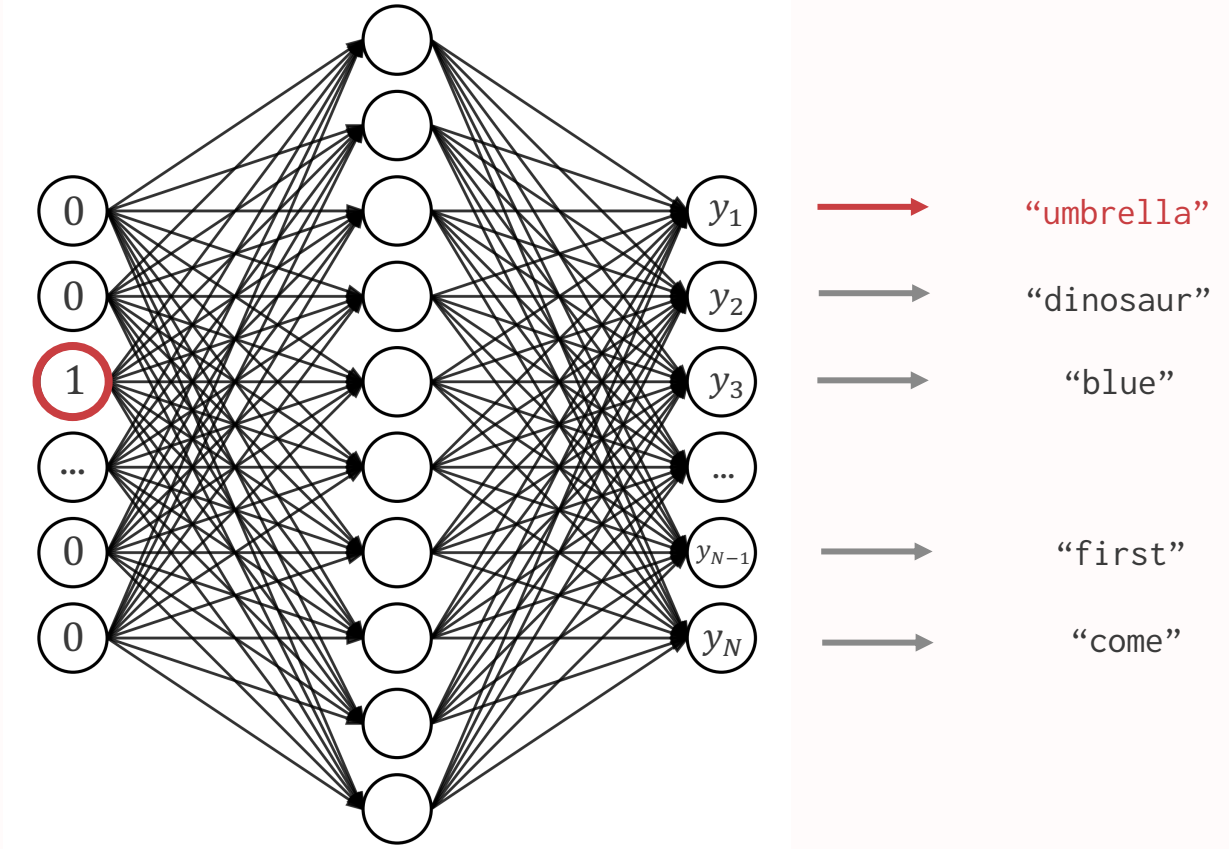
From numbers to words

EXAMPLE: a neural network replying with the word “umbrella”



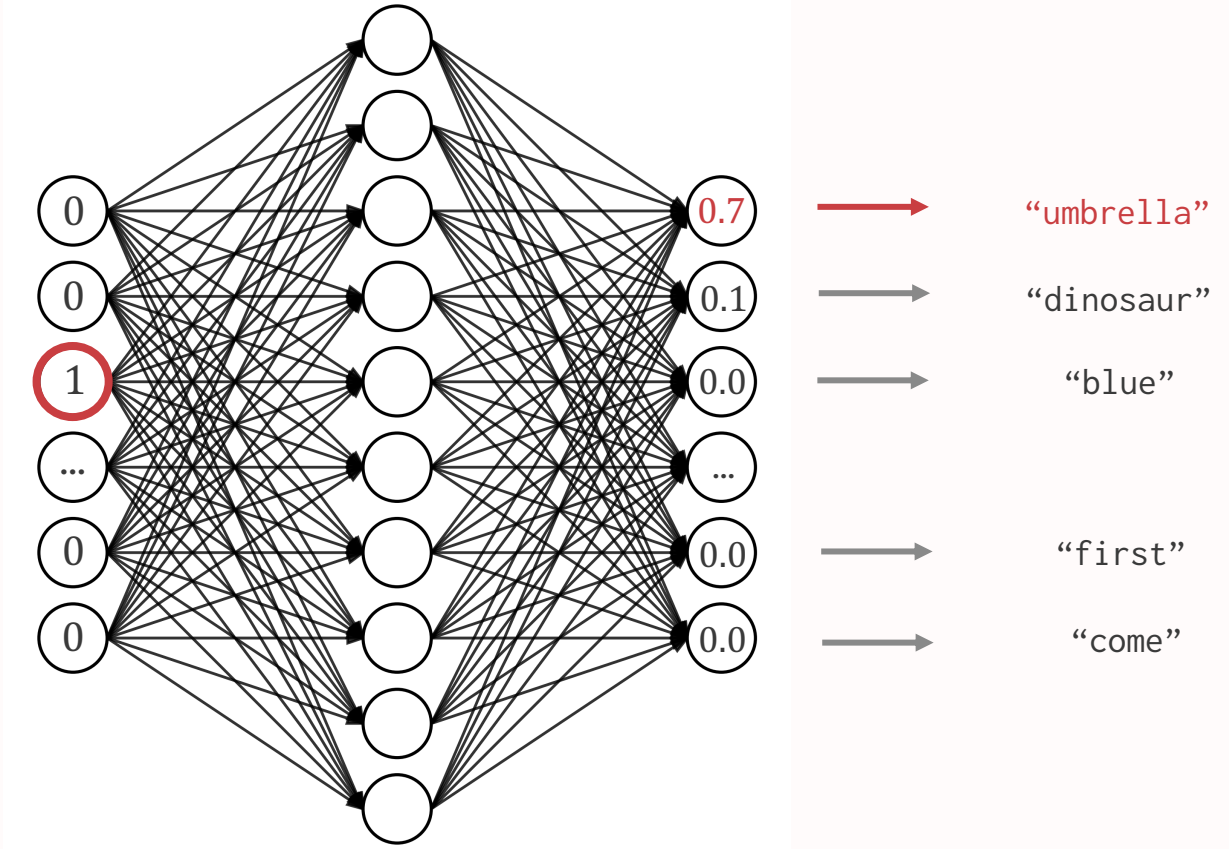
From numbers to words

EXAMPLE: a neural network replying with the word “umbrella”



From numbers to words

EXAMPLE: a neural network replying with the word “umbrella”



Output values interpreted as probabilities.

Tokenization

Representing language efficiently

Tokenization

Problems with this strategy:

1. Some "words" will *not* be in this dictionary.
   - Proper nouns
   - Strings of information, e.g., URLs: "`https://www.science.org/doi/10.1126/science.adv9817`"
   - Phone numbers, addresses, etc.
   - Words from foreign languages
2. How do we handle punctuation?
   - Need to encode "?", ".", "!", ";" etc.
3. Would need separate inputs for each and every *form* of a word, e.g.:
   - "walk"
   - "walks"
   - "walked"
   - "walking"

Tokenization

- Ideally, we want a system that:
  - Can handle *any* string of characters
  - Has similar representations for similar words
  - Minimizes the number of inputs needed to represent a word
  - Minimizes the total number of possible inputs

Tokenization

- Ideally, we want a system that:
  - Can handle *any* string of characters
  - Has similar representations for similar words
  - Minimizes the number of inputs needed to represent a word
  - Minimizes the total number of possible inputs
- IDEA 1: Each word in the dictionary is given its own input.

Tokenization

- Ideally, we want a system that:
  - Can handle *any* string of characters
  - Has similar representations for similar words
  - Minimizes the number of inputs needed to represent a word
  - Minimizes the total number of possible inputs
- IDEA 1: Each word in the dictionary is given its own input.

Tokenization

- Ideally, we want a system that:
  - Can handle *any* string of characters
  - Has similar representations for similar words
  - Minimizes the number of inputs needed to represent a word
  - Minimizes the total number of possible inputs
- IDEA 2: Each key on the keyboard is given its own input.

Tokenization

- Ideally, we want a system that:
  - Can handle *any* string of characters
  - Has similar representations for similar words
  - Minimizes the number of inputs needed to represent a word
  - Minimizes the total number of possible inputs
- IDEA 2: Each key on the keyboard is given its own input.

Tokenization

- Ideally, we want a system that:
  - Can handle *any* string of characters
  - Has similar representations for similar words
  - Minimizes the number of inputs needed to represent a word
  - Minimizes the total number of possible inputs
- IDEA 3: Byte pair encoding.
  - Encode *parts* of words.
  - Common pairs of words get *merged* into a single *token*.

Byte pair encoding

The standard approach to tokenization

Byte pair encoding

- ITERATION 1:
  - Count frequency of all characters in the corpus.

Text

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

Frequency

_	e	s	a	t	o	h	l	u	b	d	w	c	f	i	m	n	p	r
33	28	15	12	11	8	6	6	4	3	3	3	2	1	1	1	1	1	1

Prince, Simon J. D. Understanding Deep Learning. Cambridge, MA: The MIT Press, 2023. http://udlbook.com

Byte pair encoding

- ITERATION 2:
  - *Merge* most common pair of tokens into a *new* token.
  - "s" followed by "e" → "se"
  - Count frequency of all *tokens* in the corpus.

Text

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

Frequency

_	e	se	a	t	o	h	l	u	b	d	w	c	s	f	i	m	n	p	r
33	15	13	12	11	8	6	6	4	3	3	3	2	2	1	1	1	1	1	1

Prince, Simon J. D. Understanding Deep Learning. Cambridge, MA: The MIT Press, 2023. http://udlbook.com

Byte pair encoding

- ITERATION 3:
  - *Merge* most common pair of tokens into a *new* token.
  - "e" followed by "_" → "e_"
  - Count frequency of all *tokens* in the corpus.

Text

a_sailor_went_to_sea_sea_sea_
to_see_what_he_could_see_see_see_
but_all_that_he_could_see_see_see_
was_the_bottom_of_the_deep_blue_sea_sea_sea_

Frequency

_	se	a	e_	t	o	h	l	u	b	d	e	w	c	s	f	i	m	n	p	r
21	13	12	12	11	8	6	6	4	3	3	3	3	2	2	1	1	1	1	1	1

Prince, Simon J. D. Understanding Deep Learning. Cambridge, MA: The MIT Press, 2023. http://udlbook.com

Byte pair encoding

- ITERATION N:
  - “see_”, “sea_”, “could_”, “sailor_” are each represented by a *single* token.
  - All original single characters are still valid tokens.
  - Common phrases get mapped to single tokens.
  - Arbitrary phrases can still be represented.
  - In practice, byte pair encoding is performed over a *large* body of text.

Frequency

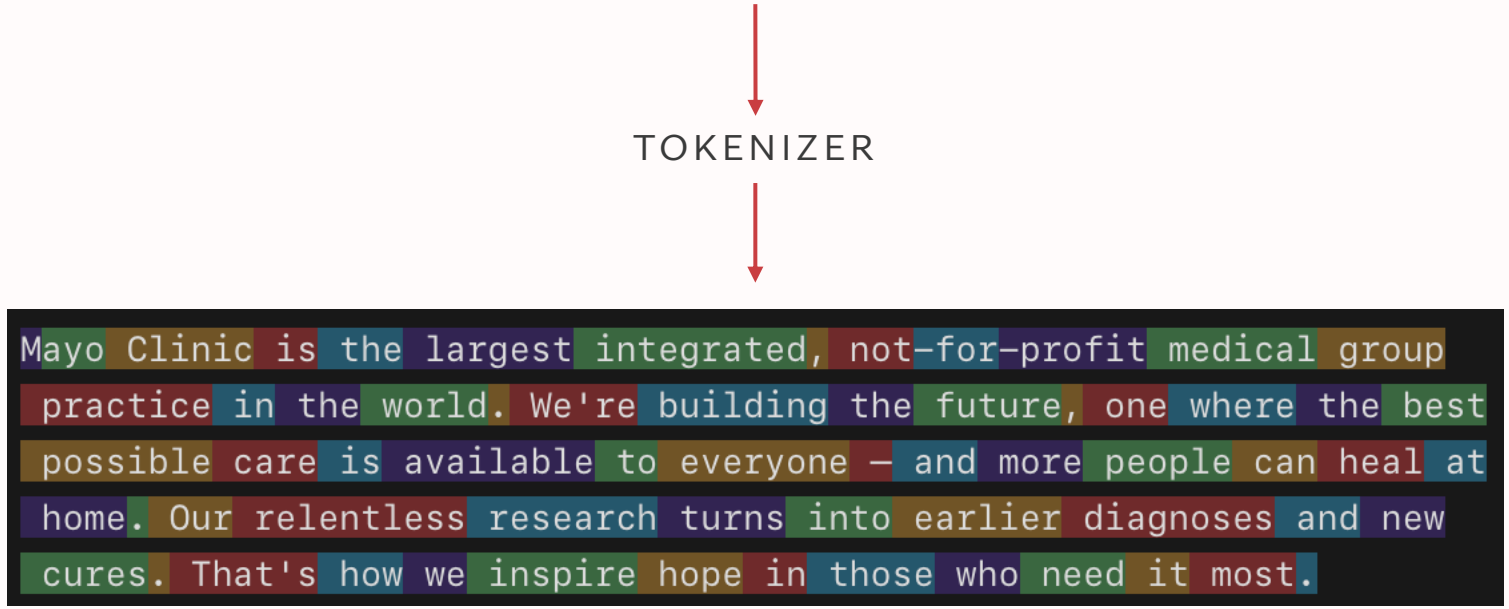
see_	sea_	e	b	l	w	a	could_	hat_	he_	o	t	t_	the_	to_	u	a_	d	f	m	n	p	s	sailor_	to
7	6	4	3	3	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1

Prince, Simon J. D. Understanding Deep Learning. Cambridge, MA: The MIT Press, 2023. http://udlbook.com

Byte pair encoding

“Mayo Clinic is the largest integrated, not-for-profit medical group practice in the world. We're building the future, one where the best possible care is available to everyone – and more people can heal at home. Our relentless research turns into earlier diagnoses and new cures. That's how we inspire hope in those who need it most.”

333 characters



65 tokens

https://platform.openai.com/tokenizer

Byte pair encoding

“Mayo Clinic is the largest integrated, not-for-profit medical group practice in the world. We're building the future, one where the best possible care is available to everyone – and more people can heal at home. Our relentless research turns into earlier diagnoses and new cures. That's how we inspire hope in those who need it most.”

333 characters

65 tokens

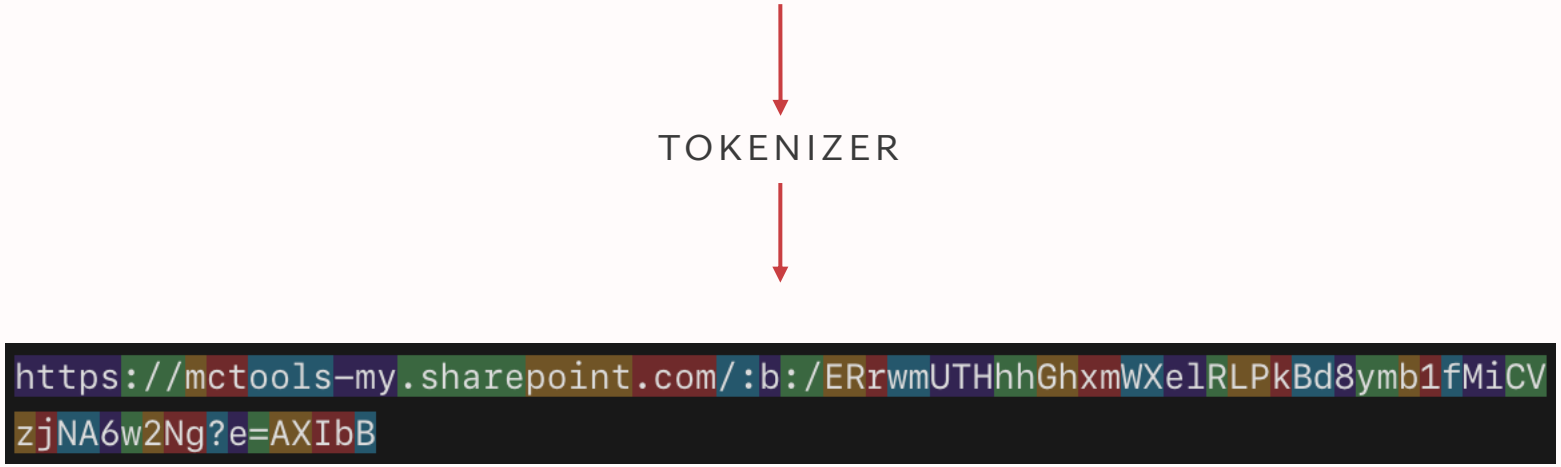
≈ 5 characters/token

https://platform.openai.com/tokenizer

Byte pair encoding

“https://mctools-my.sharepoint.com/:b:/ERrwmUTHhhGhxmWXelRLPkBd8ymb1fMiCVzjNA6w2Ng?e=AXIbB”

89 characters



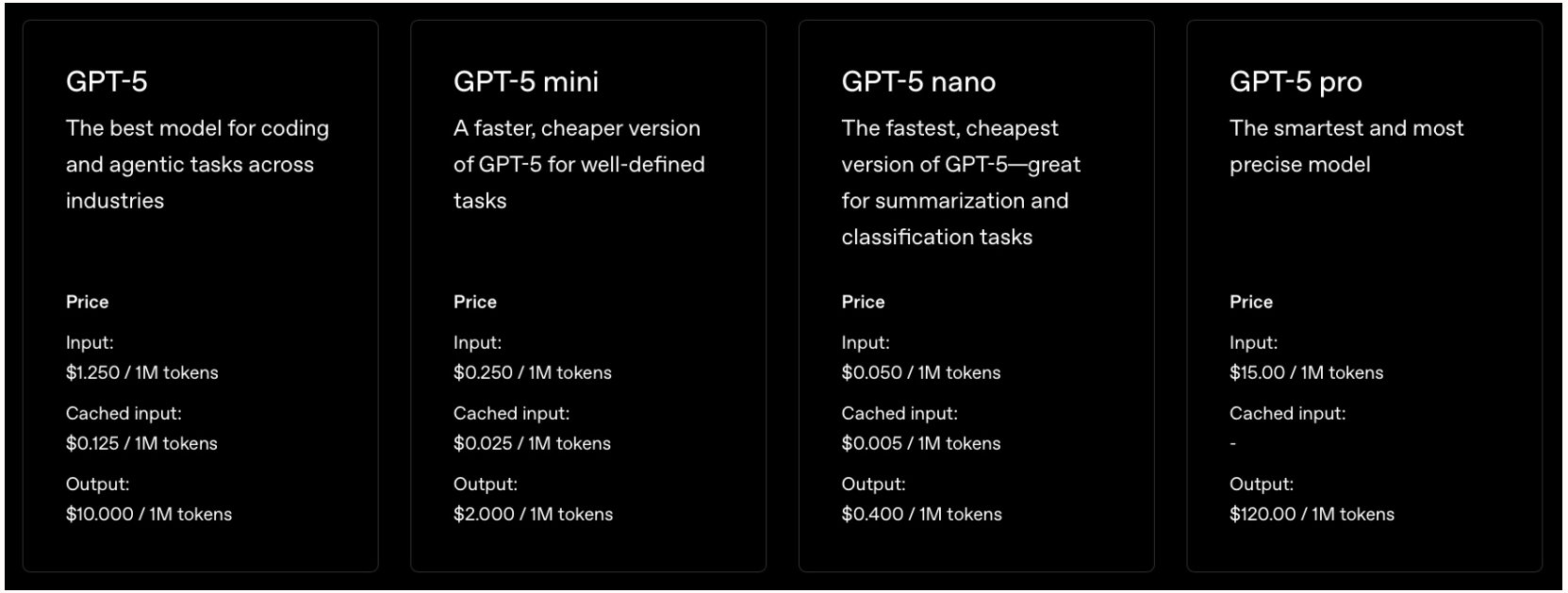
45 tokens

≈ 2 characters/token

https://platform.openai.com/tokenizer

Tokenization and cost

- The fundamental processing unit is the *token*.
- Each token represents running the neural network once.
- Hence *large language model use is billed in tokens.*



GPT-5	GPT-5 mini	GPT-5 nano	GPT-5 pro
The best model for coding and agentic tasks across industries	A faster, cheaper version of GPT-5 for well-defined tasks	The fastest, cheapest version of GPT-5—great for summarization and classification tasks	The smartest and most precise model
Price	**Price**	**Price**	**Price**
Input: $1.250 / 1M tokens	Input: $0.250 / 1M tokens	Input: $0.050 / 1M tokens	Input: $15.00 / 1M tokens
Cached input: $0.125 / 1M tokens	Cached input: $0.025 / 1M tokens	Cached input: $0.005 / 1M tokens	Cached input: -
Output: $10.000 / 1M tokens	Output: $2.000 / 1M tokens	Output: $0.400 / 1M tokens	Output: $120.00 / 1M tokens

https://openai.com/api/pricing/

Large language models from 40,000 feet

- ChatGPT, Claude, Gemini, Copilot, DeepSeek…
- Still deep neural networks!
- Plan for LLMs:
    1. Represent words as (many) numbers.
    2. Generate sentences by predicting next word.
    3. Train on data from Internet.
    4. Specialized neural network architecture for text.
    5. Steer network to be helpful and accurate.

Large language models from 40,000 feet

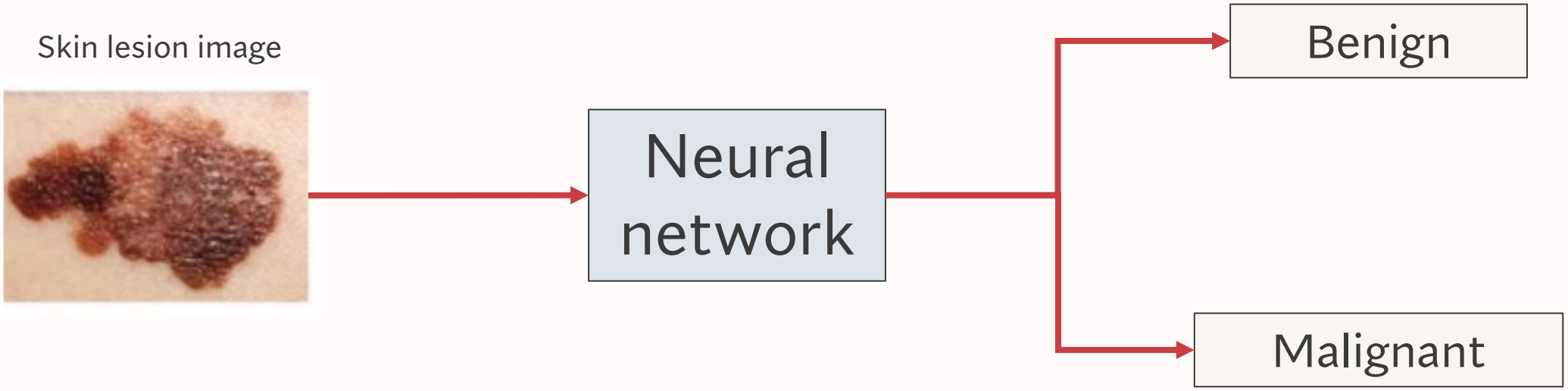
- ChatGPT, Claude, Gemini, Copilot, DeepSeek…
- Still deep neural networks!
- Plan for LLMs:
    1. Represent words as (many) numbers.
    2. Generate sentences by predicting next word.
    3. Train on data from Internet.
    4. Specialized neural network architecture for text.
    5. Steer network to be helpful and accurate.

Generating text

Next-token prediction

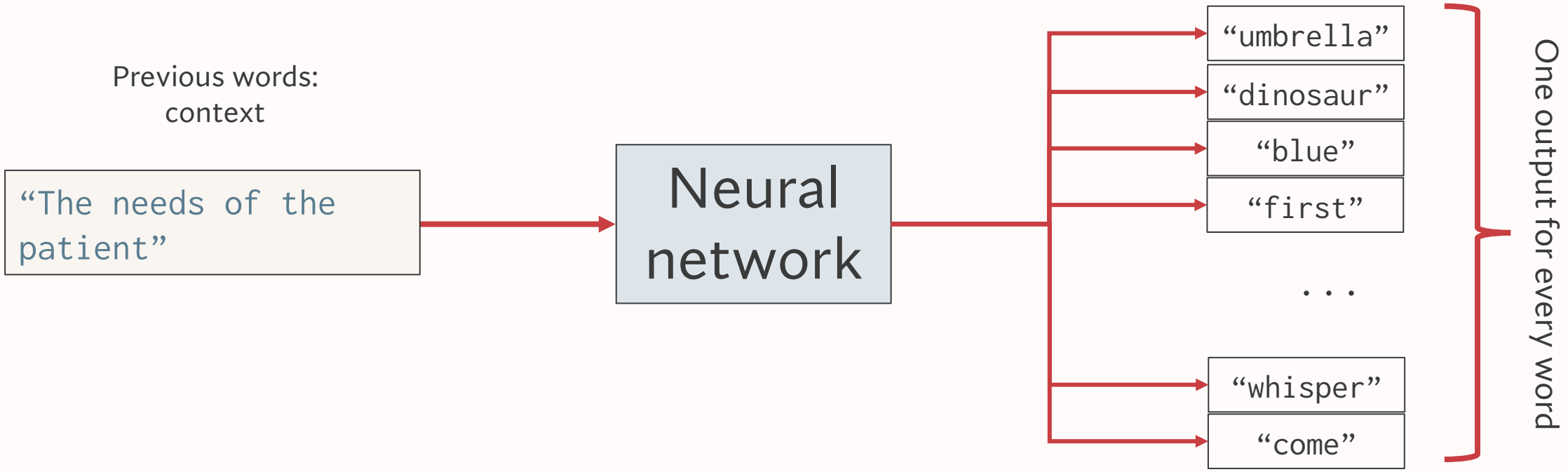
Generation by prediction

- How to generate entire sentences with a neural network?
- Idea: sentence generation is just *repeated* next-word prediction.
- LAST TIME: predict malignancy given image.



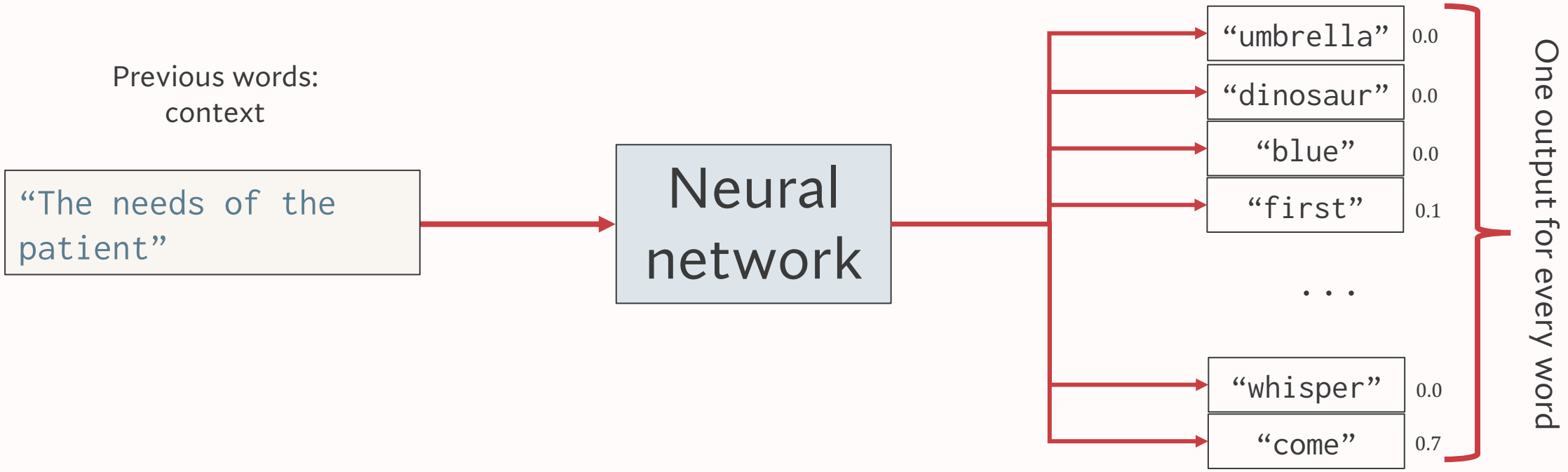
- NOW: predict *next word* given *previous words*.

Generation by prediction



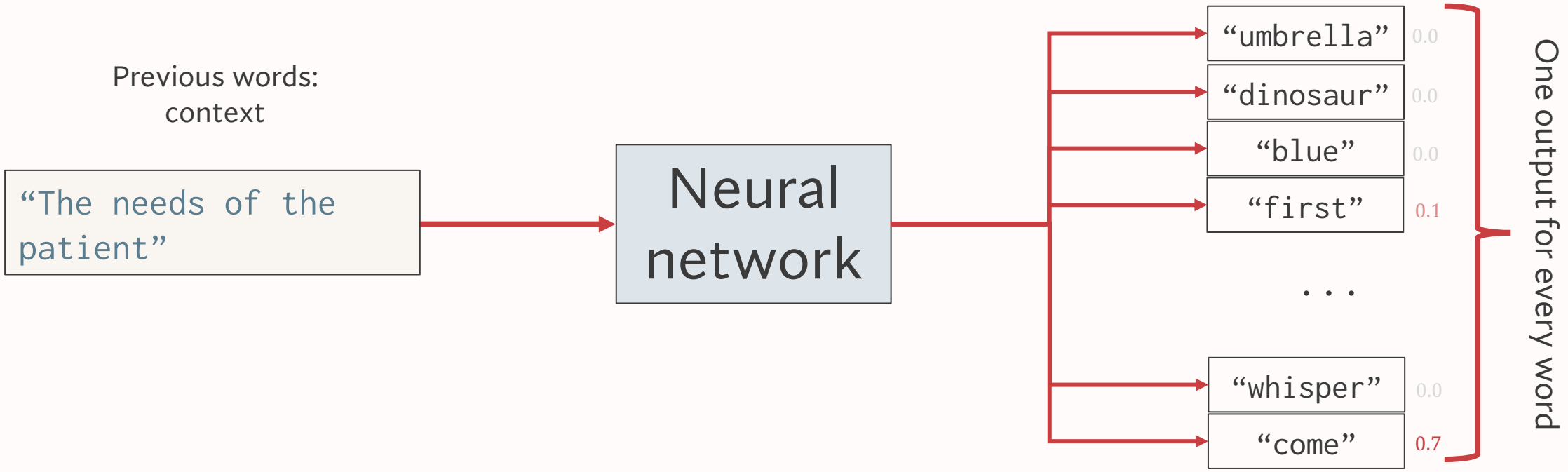
1. Predict most probable next word.

Generation by prediction



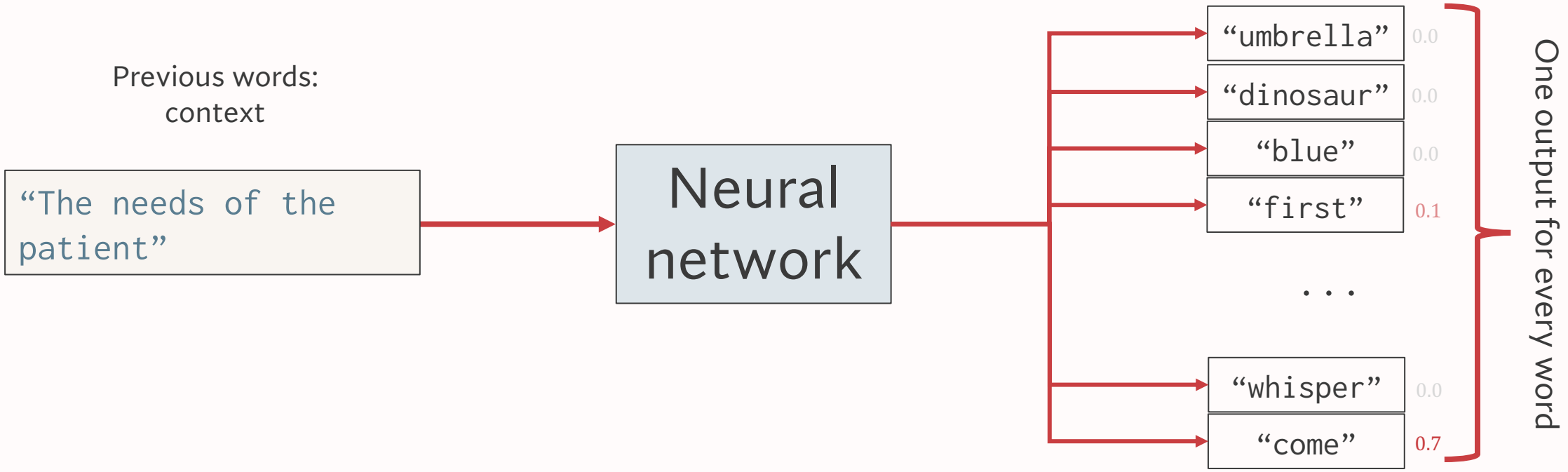
1. Predict most probable next word.

Generation by prediction



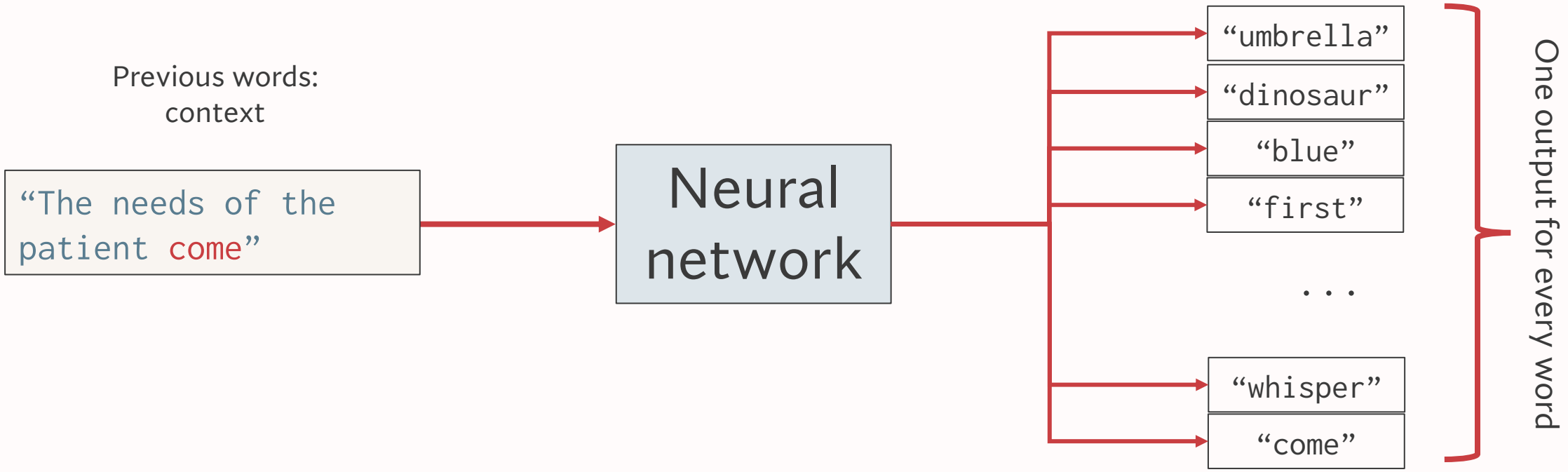
1. Predict most probable next word.

Generation by prediction



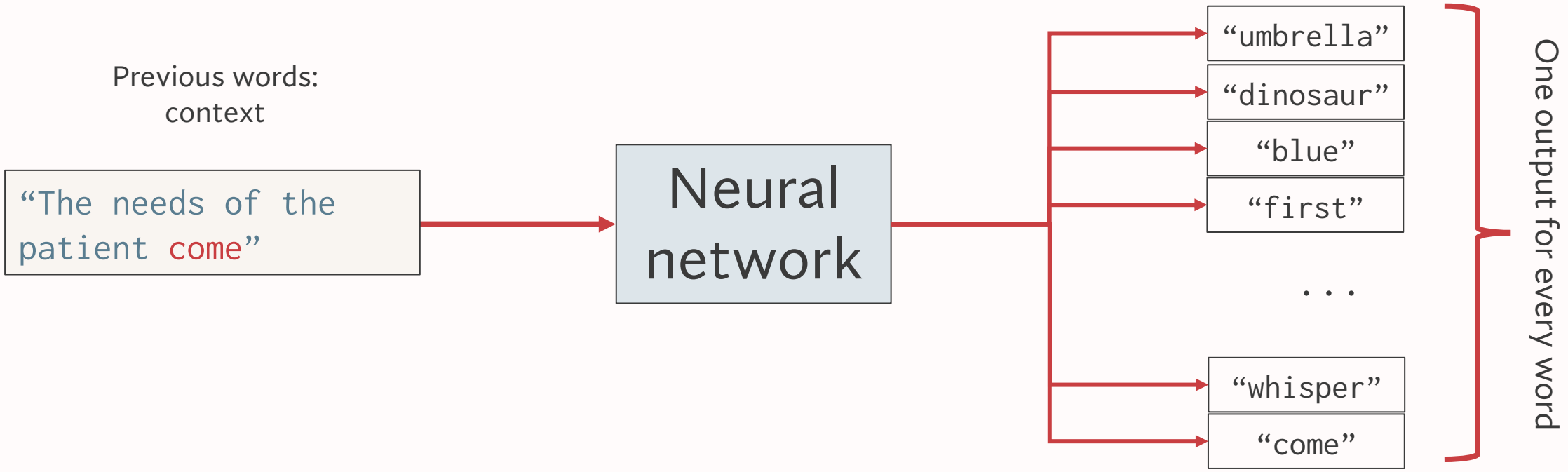
1. Predict most probable next word.
2. Add it onto the context.

Generation by prediction



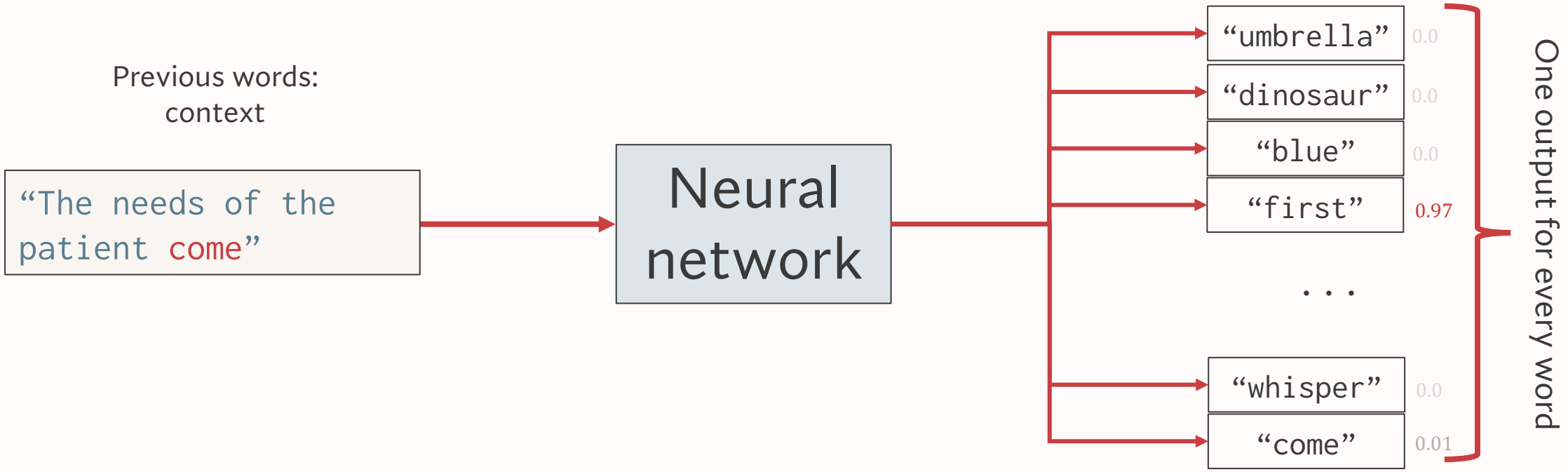
1. Predict most probable next word.
2. Add it onto the context.

Generation by prediction



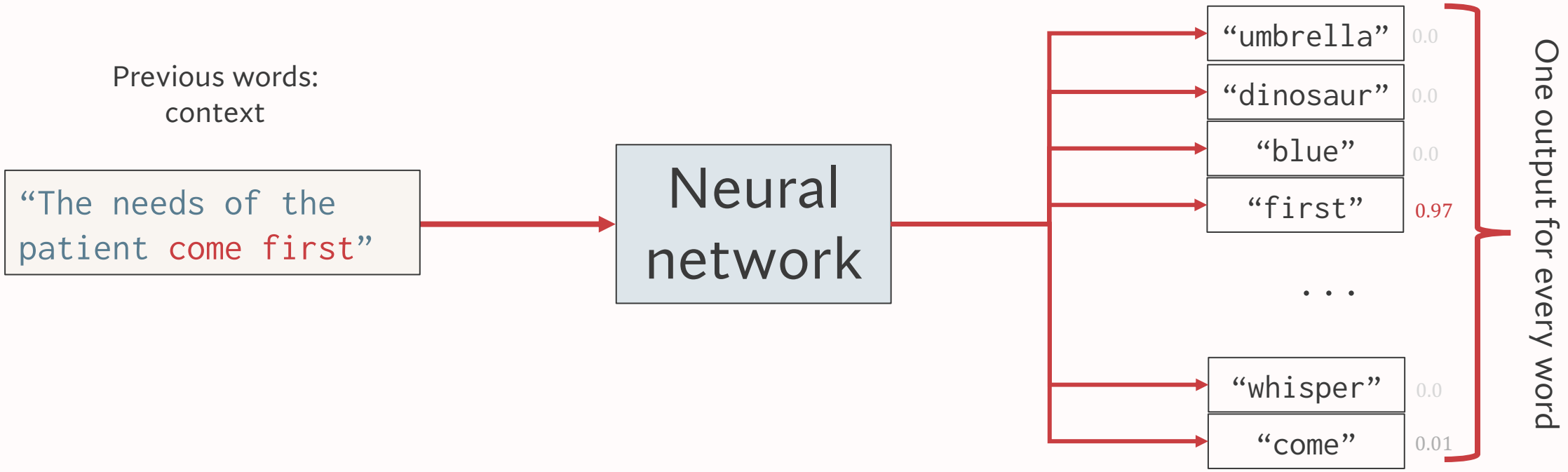
1. Predict most probable next word.
2. Add it onto the context.
3. Go back to step 1.

Generation by prediction



1. Predict most probable next word.
2. Add it onto the context.
3. Go back to step 1.

Generation by prediction



1. Predict most probable next word.
2. Add it onto the context.
3. Go back to step 1.

Generation by prediction

Repeat until entire document/conversation is generated

1. Predict
2. Add it onto the context.
3. Go back to step 1.

Large language models from 40,000 feet

Plan for LLMs:

1. Represent words as (many) numbers.
2. Generate sentences by predicting next word.
3. Train on data from Internet.
4. Specialized neural network architecture for text.
5. Steer network to be helpful and accurate.

Large language models from 40,000 feet

Plan for LLMs:

1. Represent words as (many) numbers.
2. Generate sentences by predicting next word.
3. Train on data from Internet.
4. Specialized neural network architecture for text.
5. Steer network to be helpful and accurate.

LLM Training data

Scraping the Internet

Language model training data

- Training requires dataset of many datapoints.
  - Datapoint = an example of a correct (INPUT, OUTPUT) pair.
  - Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), ..., etc.
- *Last time*:
  - Dataset = ( , MALIGNANT), ( , BENIGN), ..., etc.
- *Now*: labels are next word in sentence.
  - Dataset = (`"Mary had a little"`, `"lamb"`),
    (`"To be or not to"`, `"be"`),
    (`"Bold Forward"`, `"Unbound"`), ..., etc.
- Where to get such a dataset?—The Internet!

Language model training data

- Computers automatically download text in billions of webpages.
- Common Crawl releases monthly web “snapshots”.
- Includes:
  - Wikipedia
  - Forum discussions
  - Online courseware
  - Academic papers
  - Books
  - Open-source code
- All human knowledge!*

Mini-internet training data

- Imagine entire internet had only *two* websites: a MINI-INTERNET.
- Each website has just *one* sentence:
  - `www.mayo.com` says: "`Mayo Clinic saves lives`"
  - `www.ai.com` says: "`Mayo Clinic uses AI`"
- We scrape this "internet" and save all text as training data.
- *What would happen if we trained an LLM on this data?*

Mini-internet training data

- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), …, etc.
- *Six* training datapoints obtained from the two sentences!
  - “`Mayo Clinic saves lives`”
  - “`Mayo Clinic uses AI`”

Mini-internet training data

- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), …, etc.
- *Six* training datapoints obtained from the two sentences!
  - "`Mayo Clinic saves lives`"
  - "`Mayo Clinic uses AI`"
- Becomes:
  1. ("`Mayo`", "`Clinic`")

Mini-internet training data

- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), …, etc.
- *Six* training datapoints obtained from the two sentences!
  - "Mayo Clinic saves lives"
  - "Mayo Clinic uses AI"
- Becomes:
  1. ("Mayo", "Clinic")
  2. ("Mayo Clinic", "saves")

Mini-internet training data

- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), …, etc.
- *Six* training datapoints obtained from the two sentences!
  - "Mayo Clinic saves lives"
  - "Mayo Clinic uses AI"
- Becomes:
  1. ("Mayo", "Clinic")
  2. ("Mayo Clinic", "saves")
  3. ("Mayo Clinic saves", "lives")

Mini-internet training data

- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), …, etc.
- *Six* training datapoints obtained from the two sentences!
  - "Mayo Clinic saves lives"
  - "Mayo Clinic uses AI"
- Becomes:
  1. ("Mayo", "Clinic")
  2. ("Mayo Clinic", "saves")
  3. ("Mayo Clinic saves", "lives")
  4. ("Mayo", "Clinic")

Mini-internet training data

- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), …, etc.
- *Six* training datapoints obtained from the two sentences!
  - "Mayo Clinic saves lives"
  - "Mayo Clinic uses AI"
- Becomes:
  1. ("Mayo", "Clinic")
  2. ("Mayo Clinic", "saves")
  3. ("Mayo Clinic saves", "lives")
  4. ("Mayo", "Clinic")
  5. ("Mayo Clinic", "uses")

Mini-internet training data

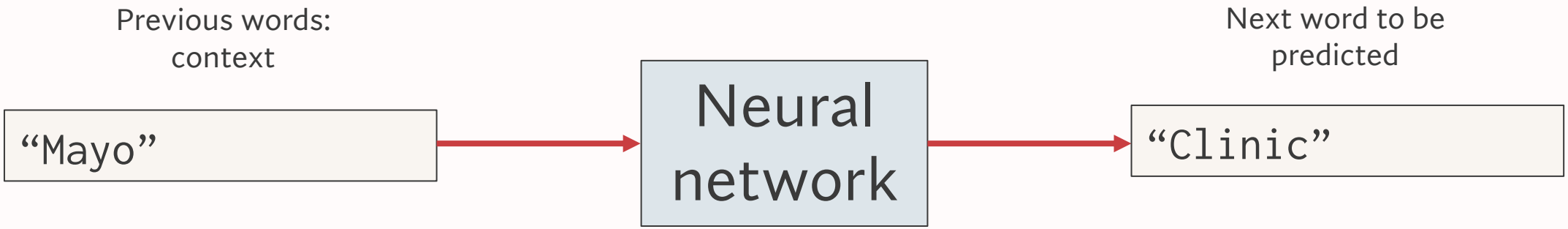
- Dataset = (INPUT 1, OUTPUT 1), (INPUT 2, OUTPUT 2), ..., etc.
- *Six* training datapoints obtained from the two sentences!
  - "Mayo Clinic saves lives"
  - "Mayo Clinic uses AI"
- Becomes:
  1. ("Mayo", "Clinic")
  2. ("Mayo Clinic", "saves")
  3. ("Mayo Clinic saves", "lives")
  4. ("Mayo", "Clinic")
  5. ("Mayo Clinic", "uses")
  6. ("Mayo Clinic uses", "AI")

Learning from the Internet

Predictive distributions

Stochastic gradient descent

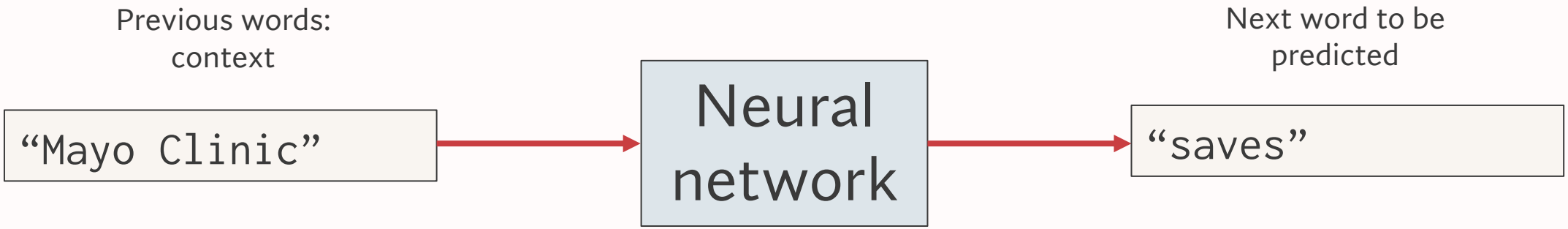
- Stochastic gradient descent iterates from one pair to the next:



- Makes small update to the parameters.
- Probability of the correct next word, given this input, goes up slightly.

Stochastic gradient descent

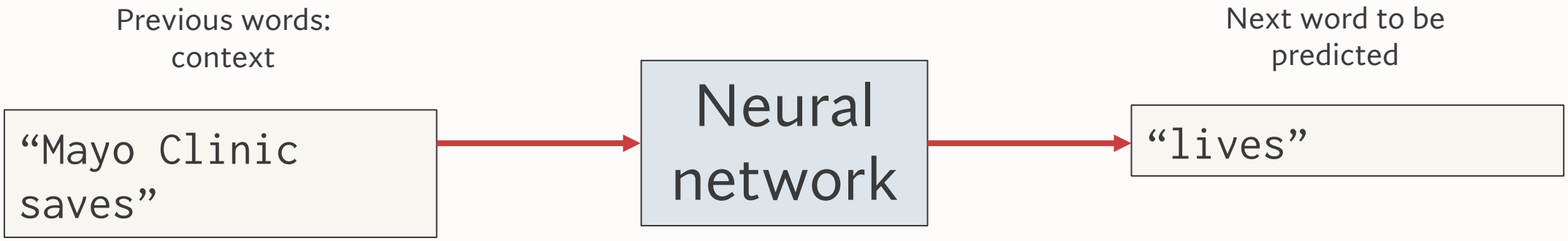
- Stochastic gradient descent iterates from one pair to the next:



- Makes small update to the parameters.
- Probability of the correct next word, given this input, goes up slightly.

Stochastic gradient descent

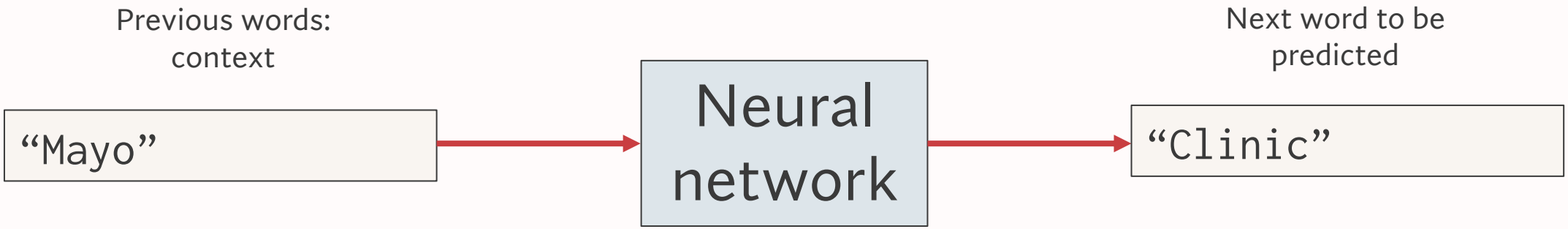
- Stochastic gradient descent iterates from one pair to the next:



- Makes small update to the parameters.
- Probability of the correct next word, given this input, goes up slightly.

Stochastic gradient descent

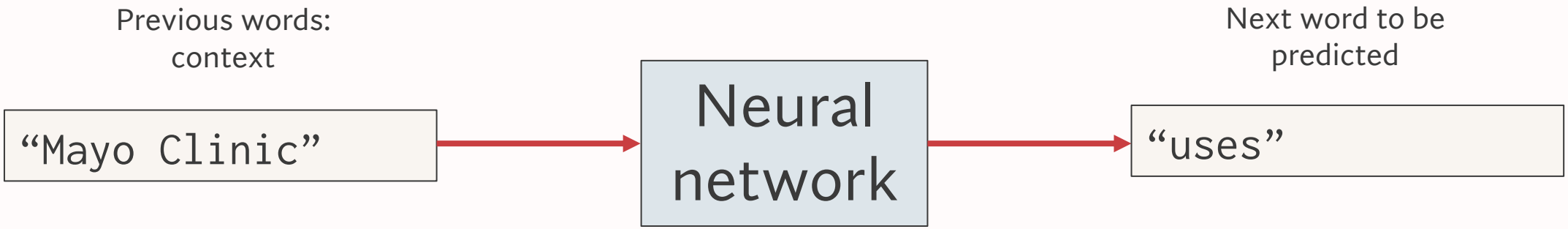
- Stochastic gradient descent iterates from one pair to the next:



- Makes small update to the parameters.
- Probability of the correct next word, given this input, goes up slightly.

Stochastic gradient descent

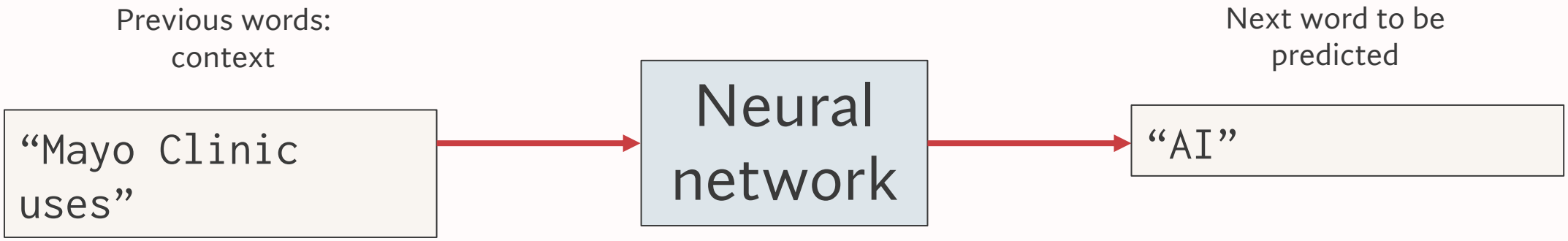
- Stochastic gradient descent iterates from one pair to the next:



- Makes small update to the parameters.
- Probability of the correct next word, given this input, goes up slightly.

Stochastic gradient descent

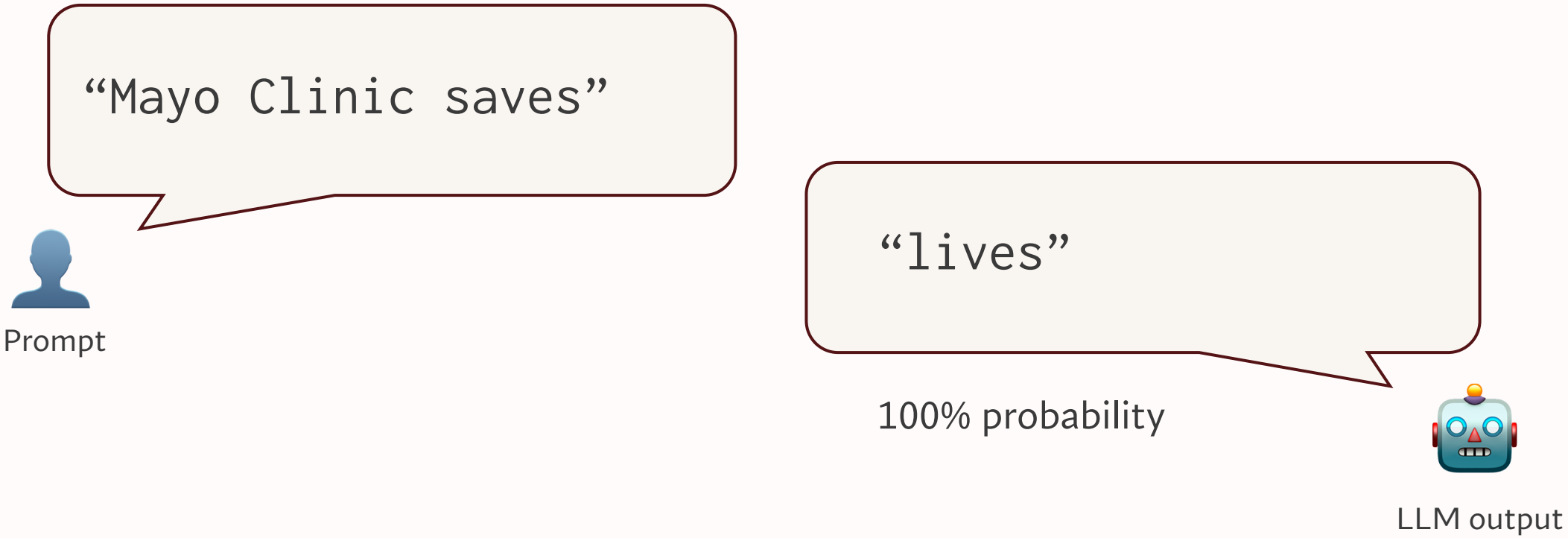
- Stochastic gradient descent iterates from one pair to the next:



- Makes small update to the parameters.
- Probability of the correct next word, given this input, goes up slightly.

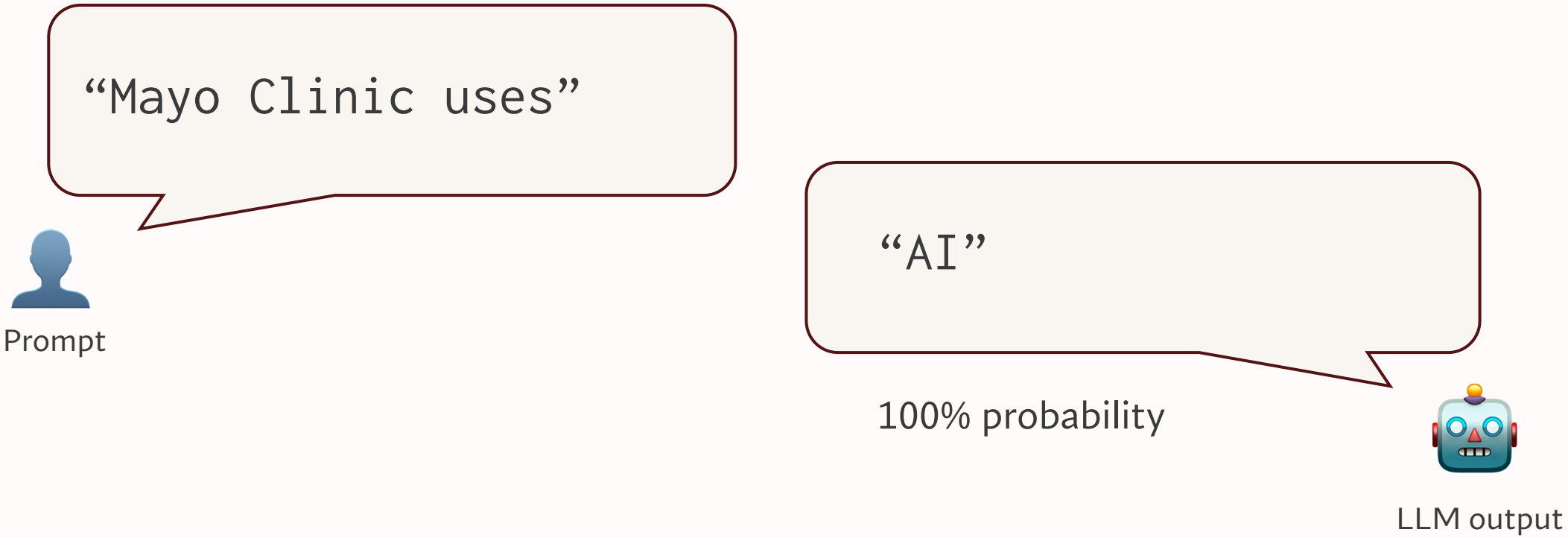
What will the neural network learn?

- If gradient descent succeeds, will learn to match distribution of training dataset.
- Like a parrot!



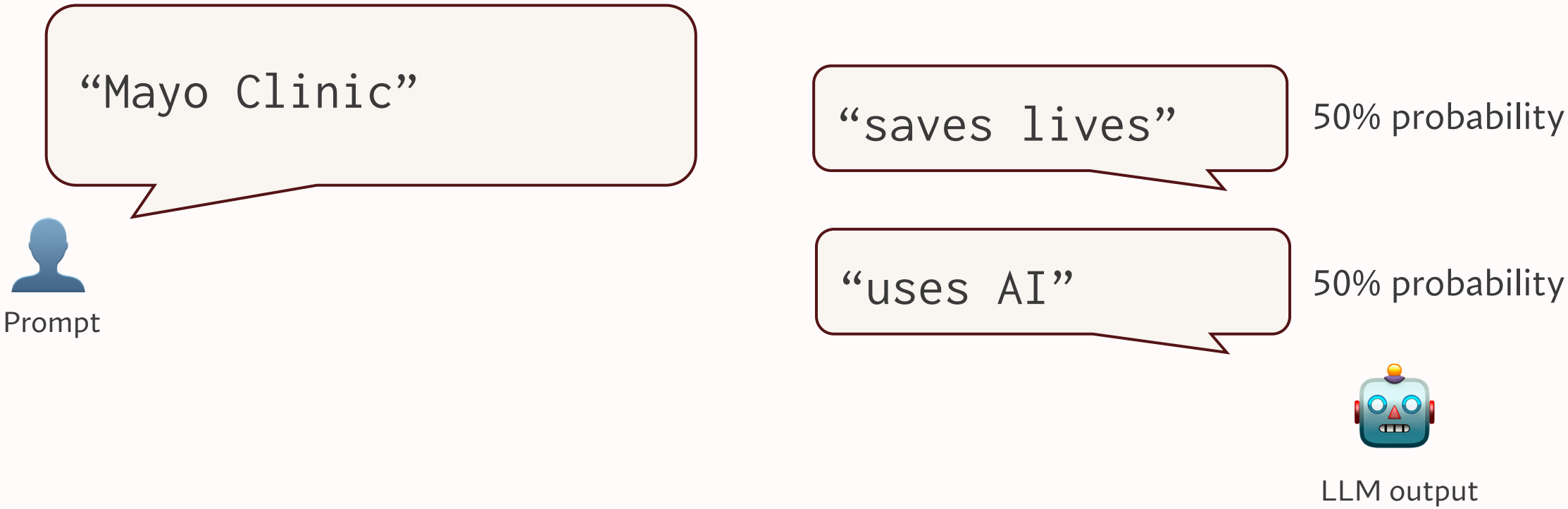
What will the neural network learn?

- If gradient descent succeeds, will learn to match distribution of training dataset.
- Like a parrot!



What will the neural network learn?

- What if the training data has multiple occurrences of the same phrase?
- Minimize loss → assign equal probability to each outcome.



Question & Answer