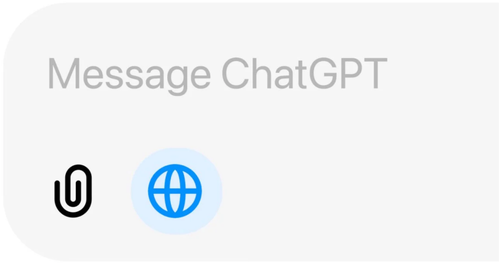
A Brief Introduction to Large Language Models

Zahra Dehghanighobadi

LLMs are all the craze



ChatGPT

[OpenAI]

Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, Yi Zhang

[arxiv]

EMERGING TECHNOLOGIES

How generative AI could add trillions to the global economy

[World Economic Forum]

Apple Intelligence

[Apple]

Modern LLMs are great

- Fast becoming everyday assistants to help with a wide range of tasks

Legal QA

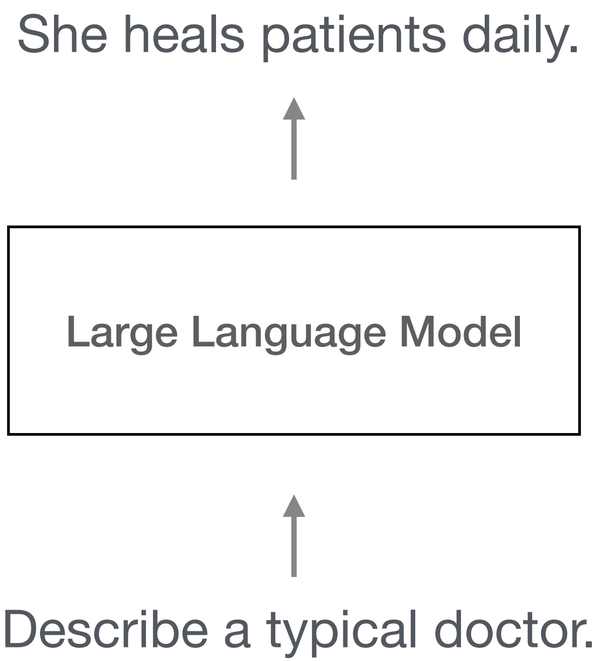
Theorem Proving

Holiday Planning

- According to some assessments, pass the Turing Test [Biever 2023]

Anatomy of LLMs

(Rough) Anatomy of a generation



*Fictitious Example

(Rough) Anatomy of a generation

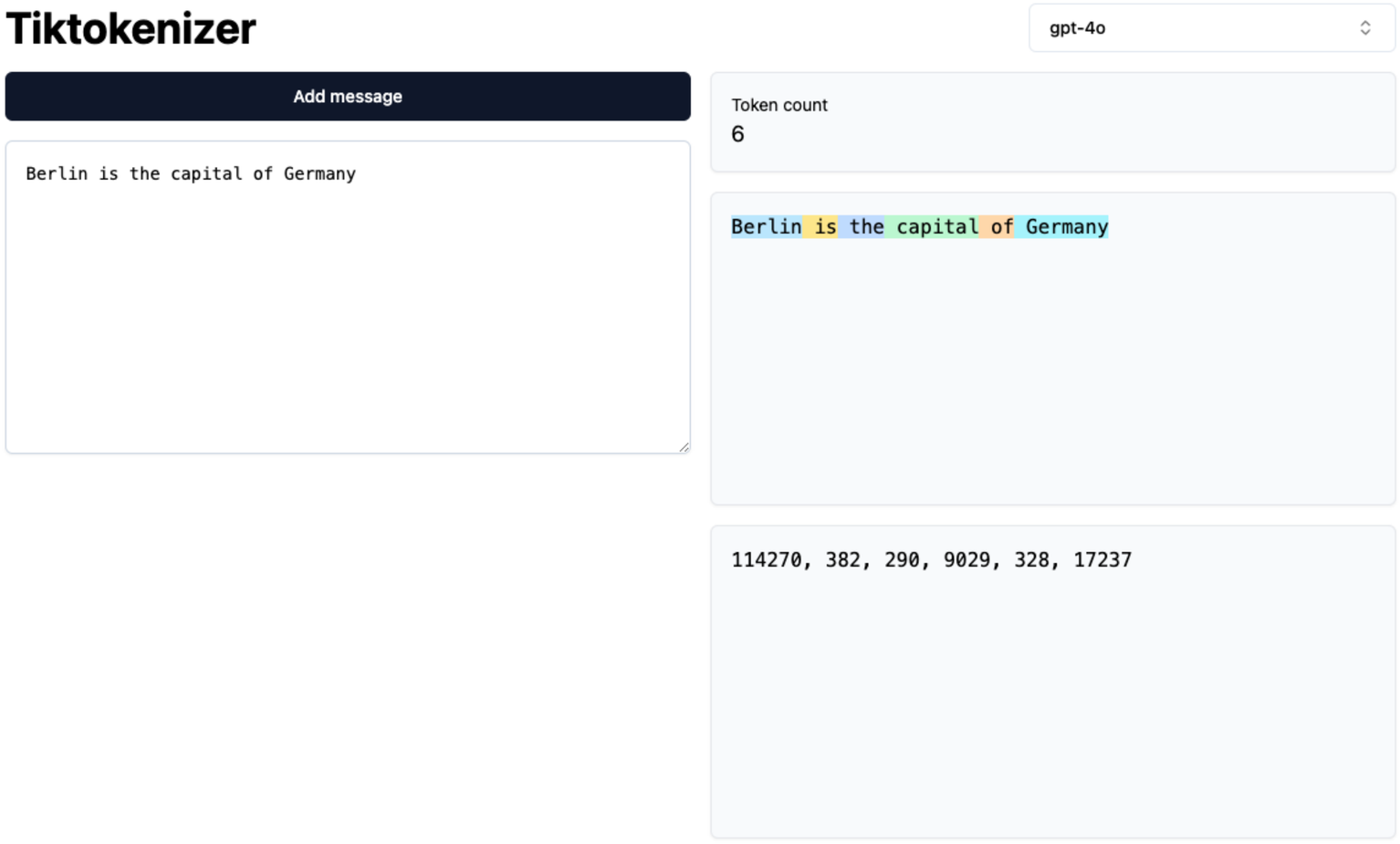
Describe a typical doctor.

Step 1: Tokenization

Tokenization

- Split the input text into individual **tokens**
- A token is usually smaller than a word, e.g., *hopeful* → *hope* + *ful*
- English vocabulary is very large and we will end up with huge input embedding lookup tables
- But words share subparts, e.g., consider the 7 words with 2 variations (27 words in total)
  - color, hope, help, harm, lust, mean, power
  - colorful, hopeful, helpful, harmful, lustful, meaningful, powerful
  - coloring, hoping, helping, harming, lusting, meaning, powering
- With ful and ing as subwords, we can represent all words as 7+2 = 9 tokens instead of 27 words
- Learn more about tokenization in this HuggingFace tutorial

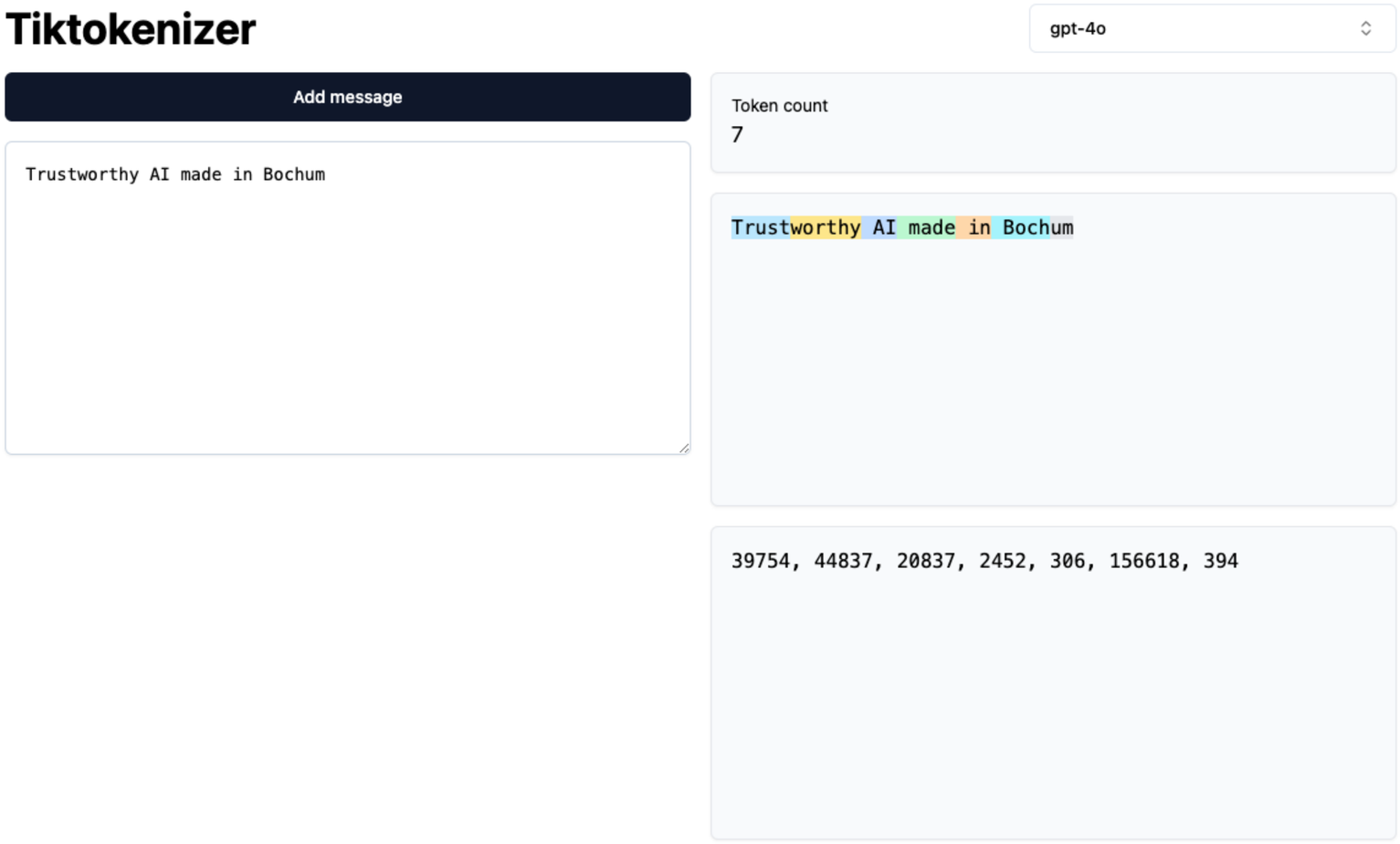
Tokenization



[Tiktokenizer]

Each word corresponds to a token

Tokenization



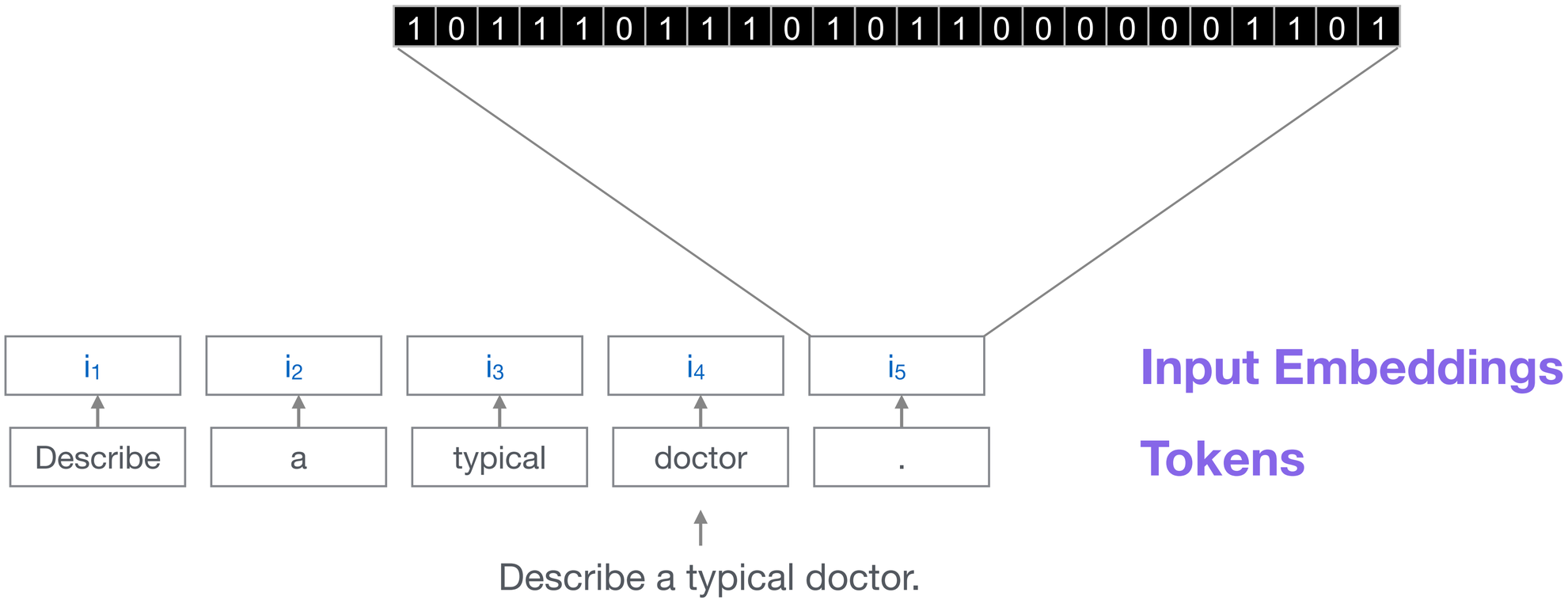
[Tiktokenizer]

Some words are split into multiple tokens

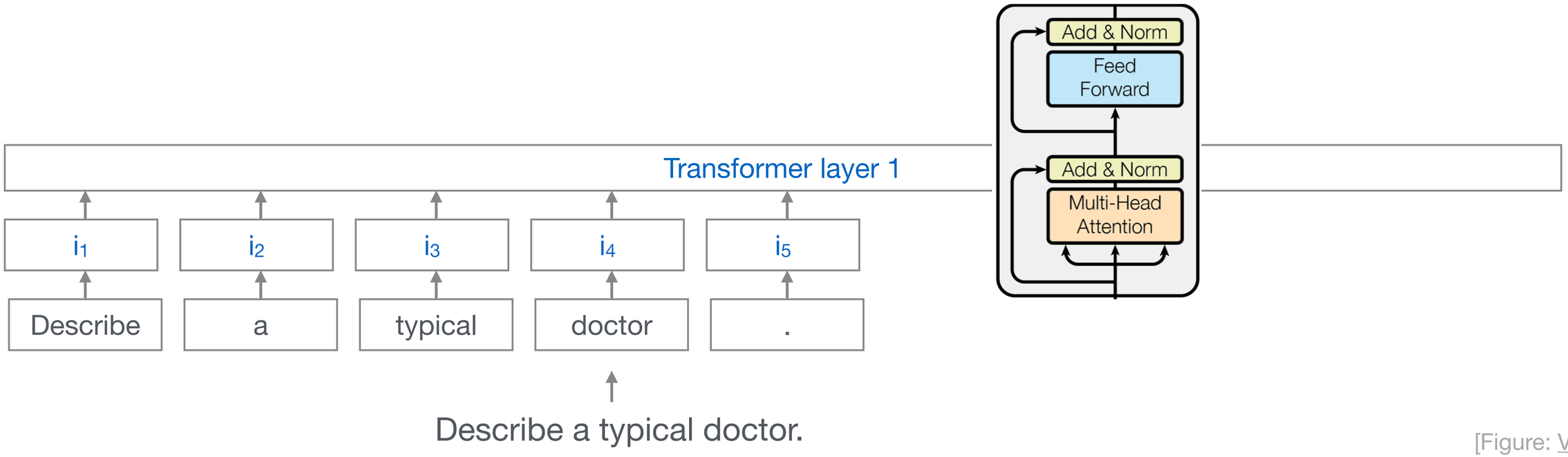
Step 1: Tokenization

Step 2: Conversion to input embeddings

- More than a simple look up table
- There is also *Positional Encoding* but we can safely ignore it in this course
- To learn more, see this blog by Lilian Weng

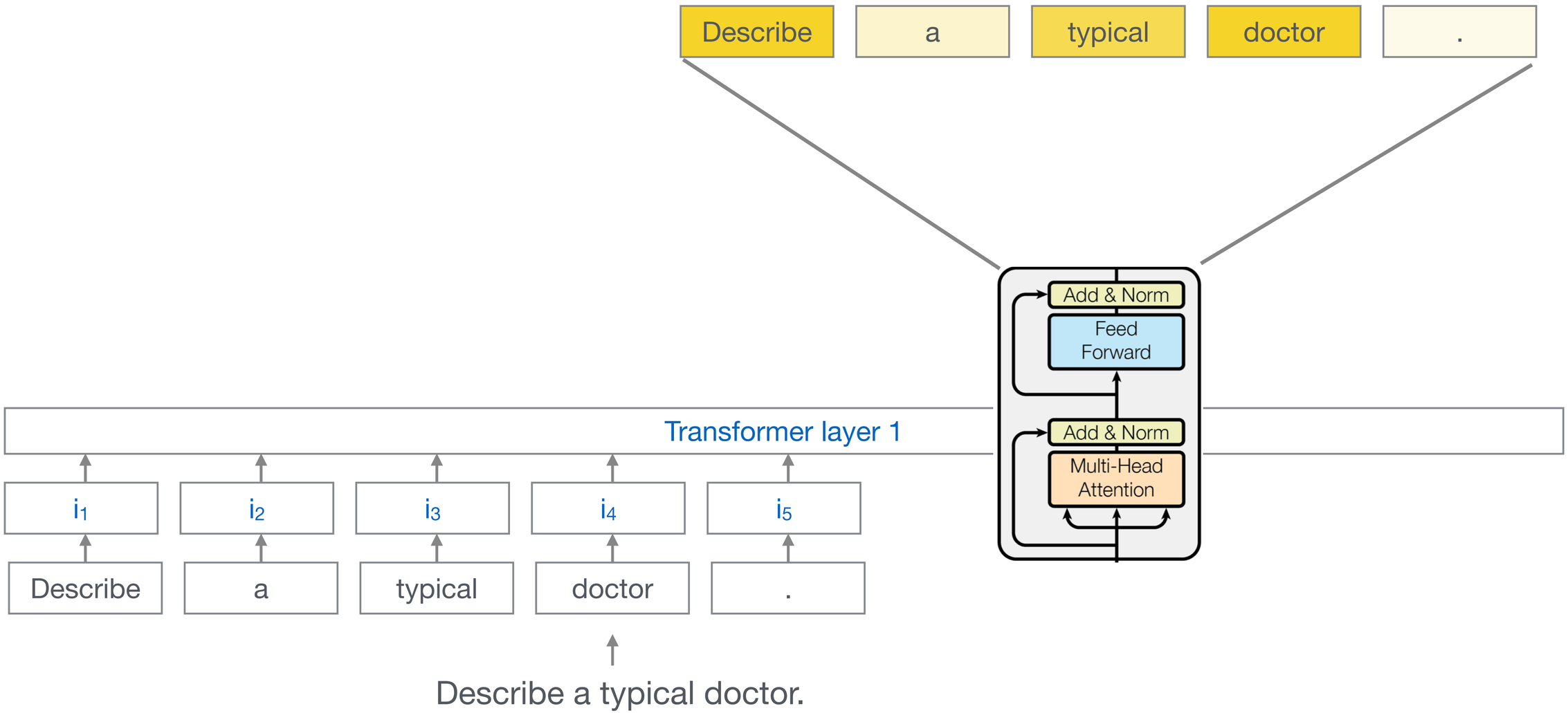


Step 3: Self-attention

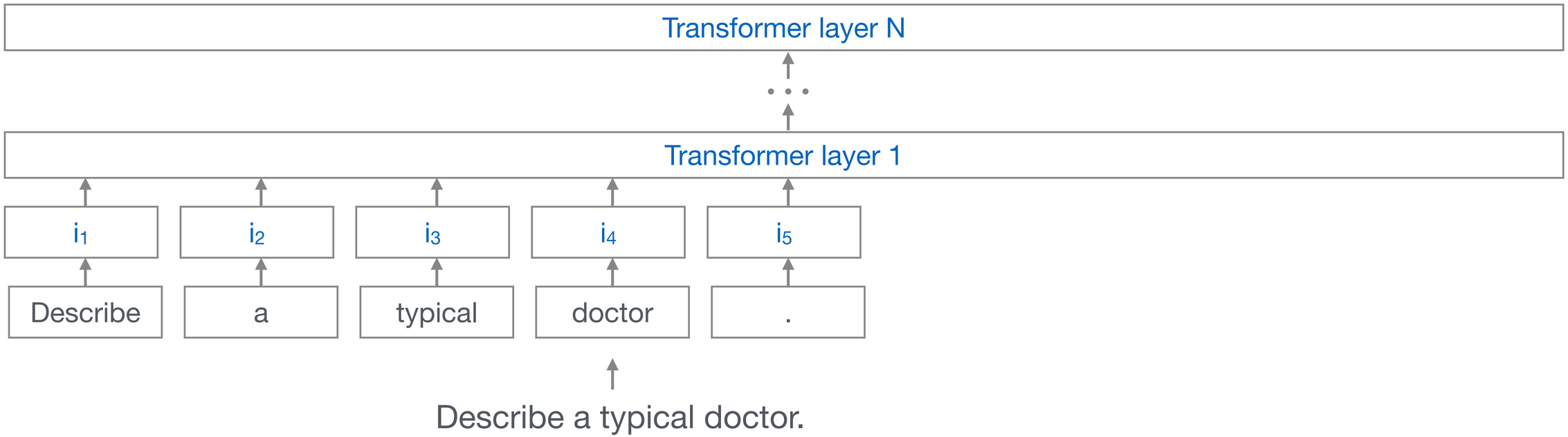


[Figure: Vaswani et al]

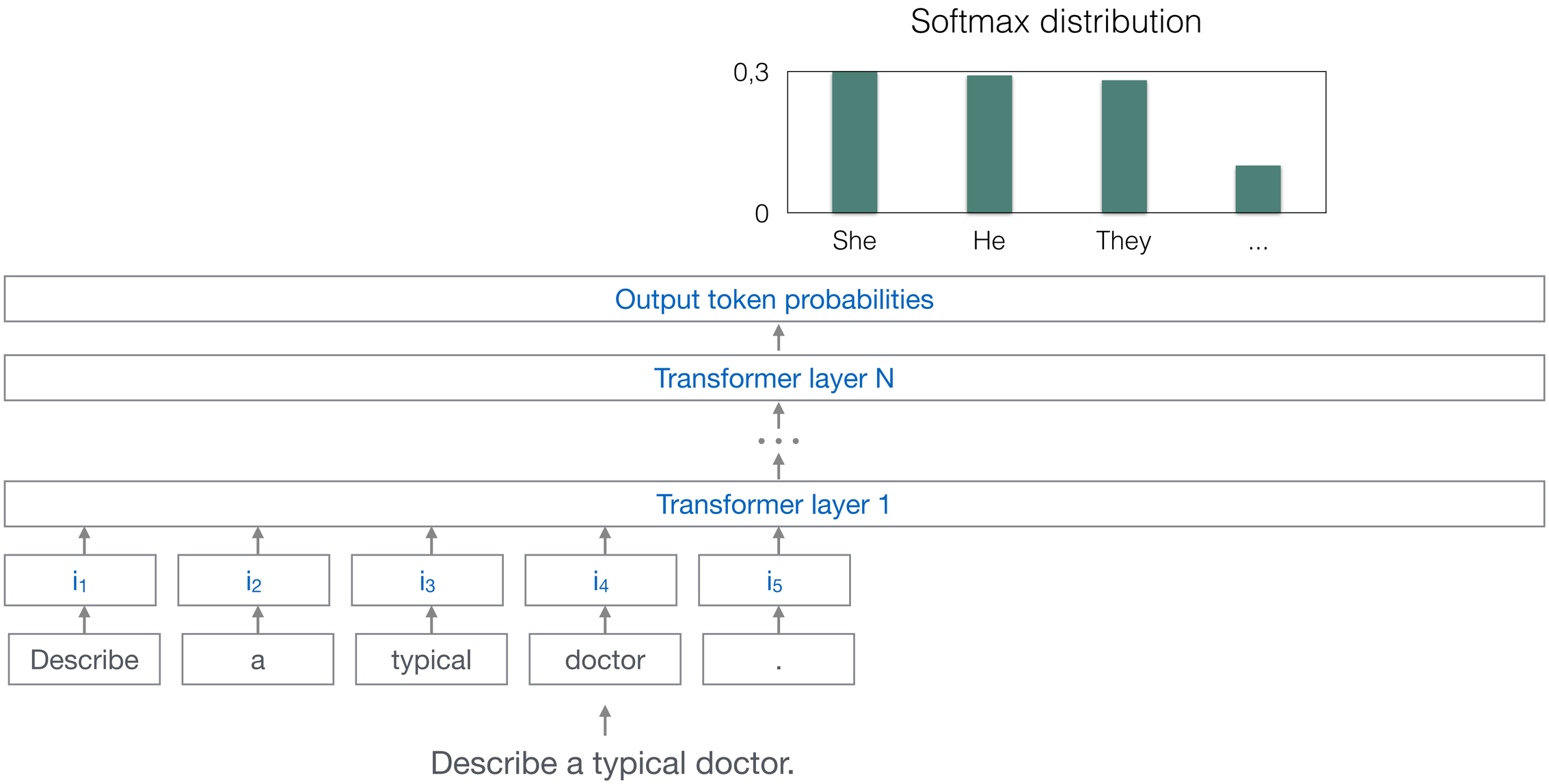
Step 3: Self-attention



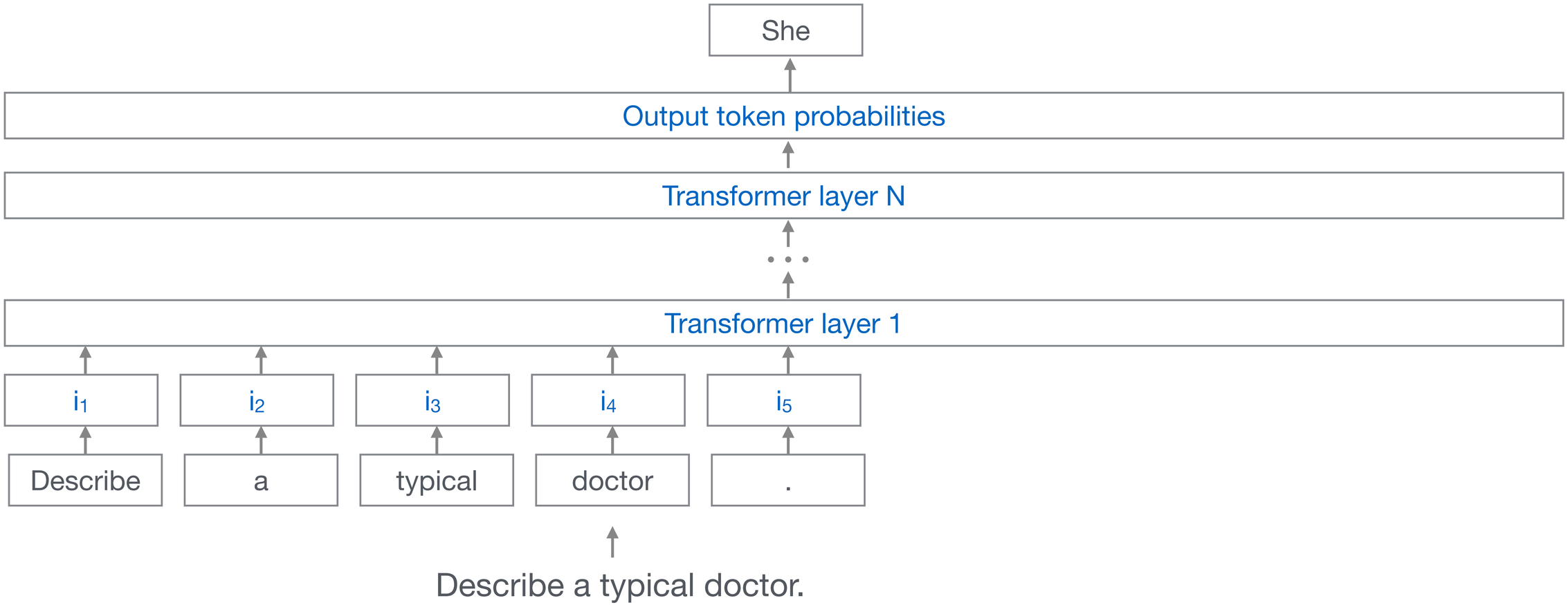
Step 3: Self-attention



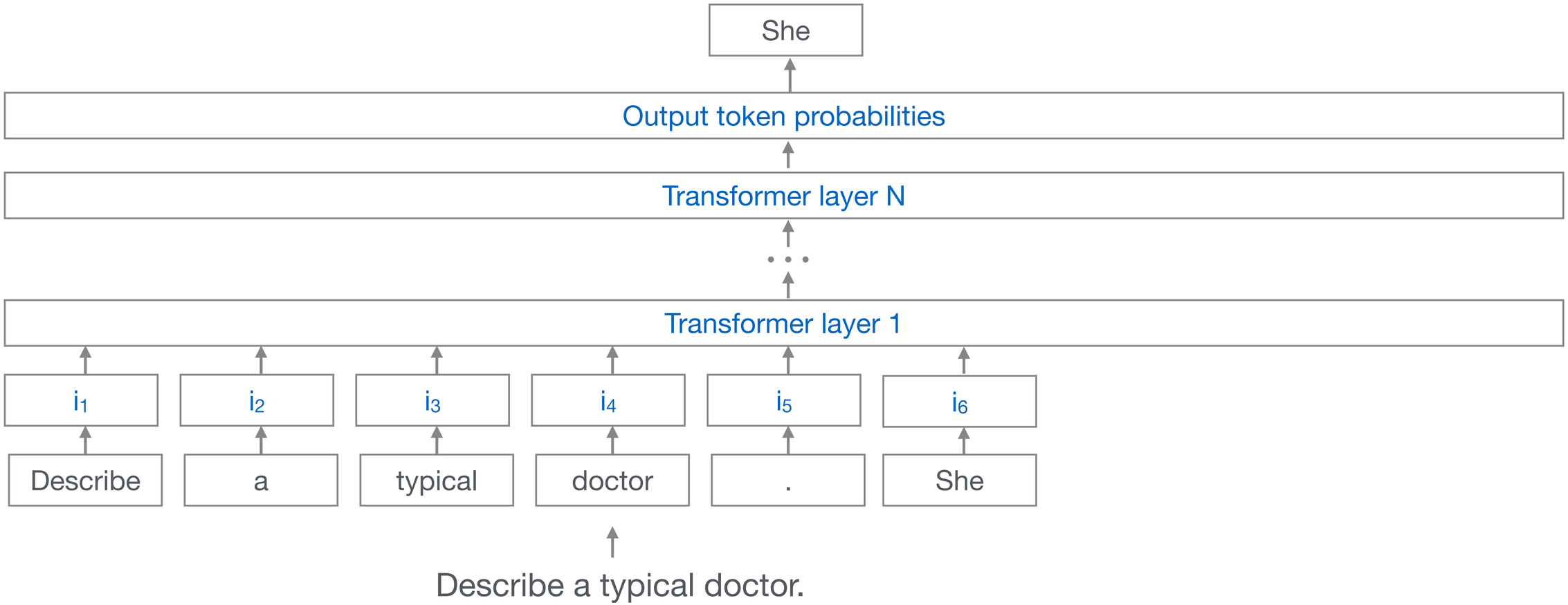
Step 4: Probability of the next token



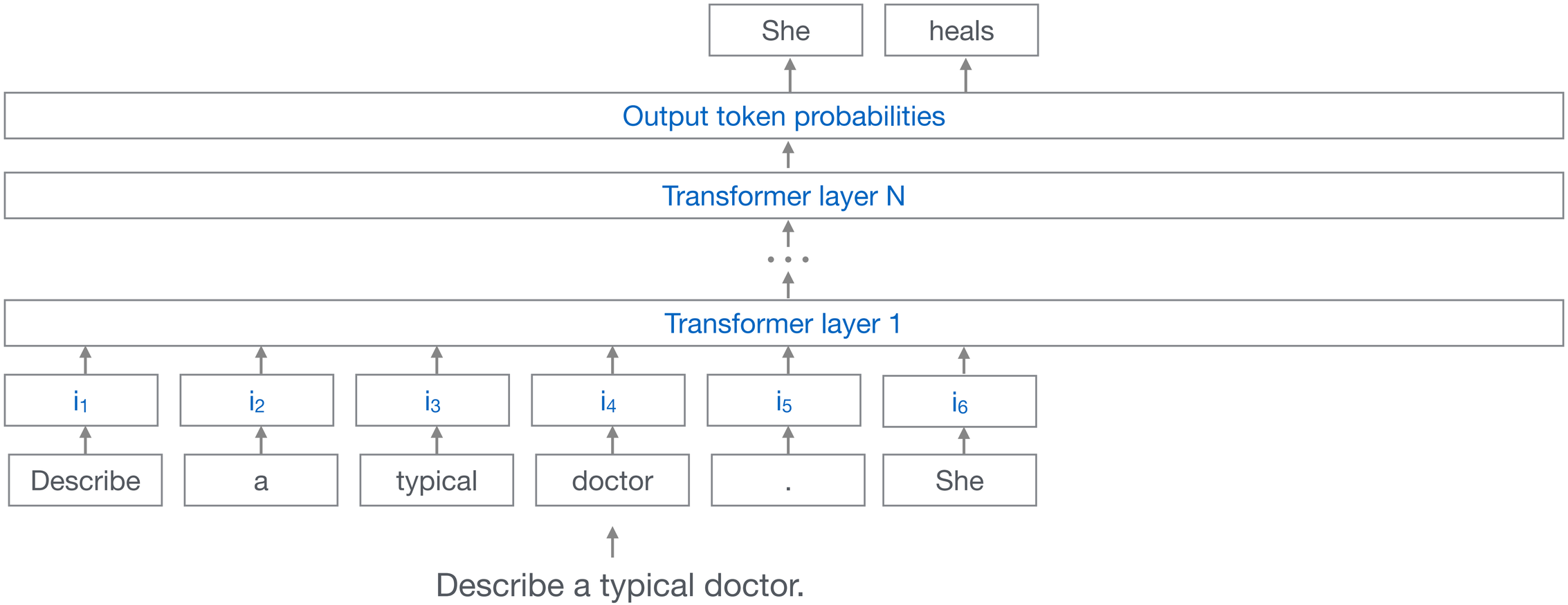
Step 5: Generate the token



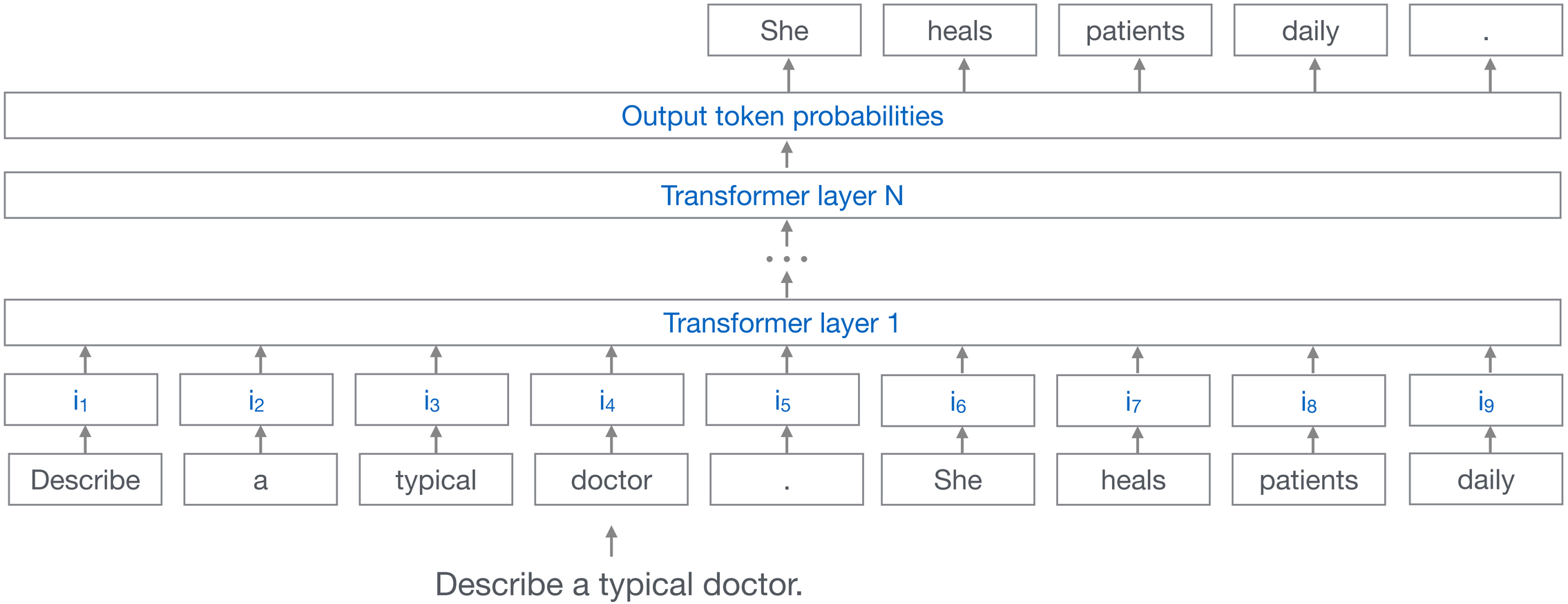
Step 6: Add the token to the input



Continue ...


She
heals
Output token probabilities
Transformer layer N
Transformer layer 1
i1
i2
i3
i4
i5
i6
Describe
a
typical
doctor
.
She
Describe a typical doctor.

Until some stopping condition is met

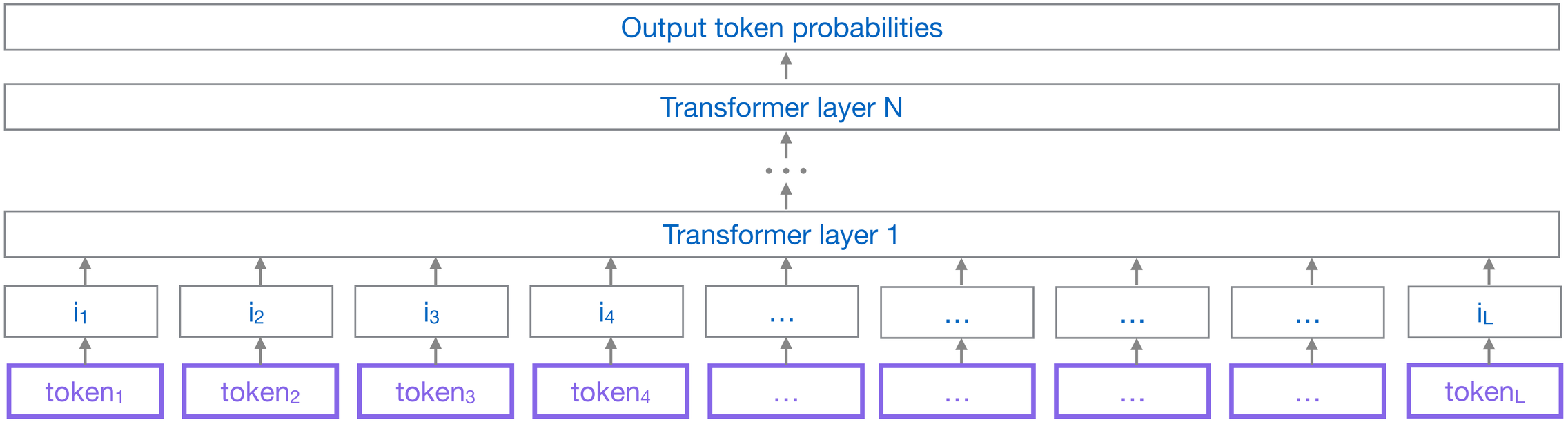


Stopping conditions

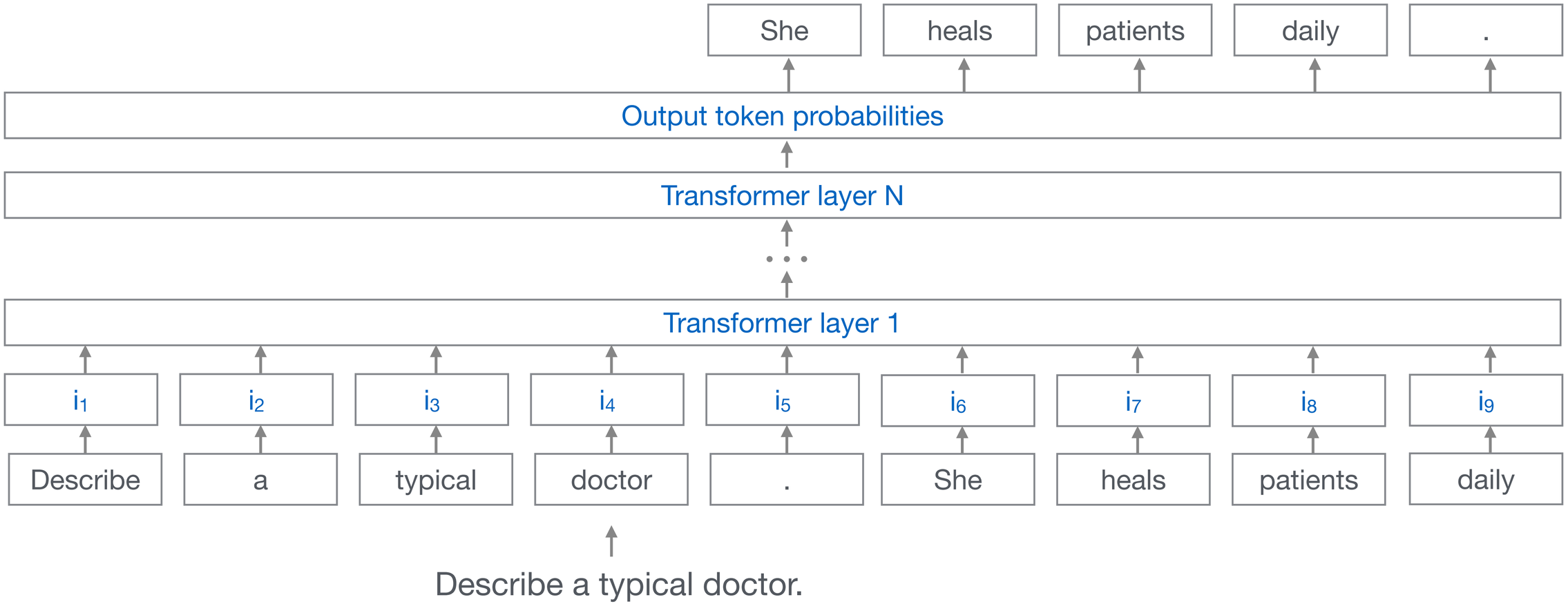
- Generate only 10 new tokens
- Stop when the model generates a specific token, e.g., fullstop “.”

Transformers have a maximum sequence length

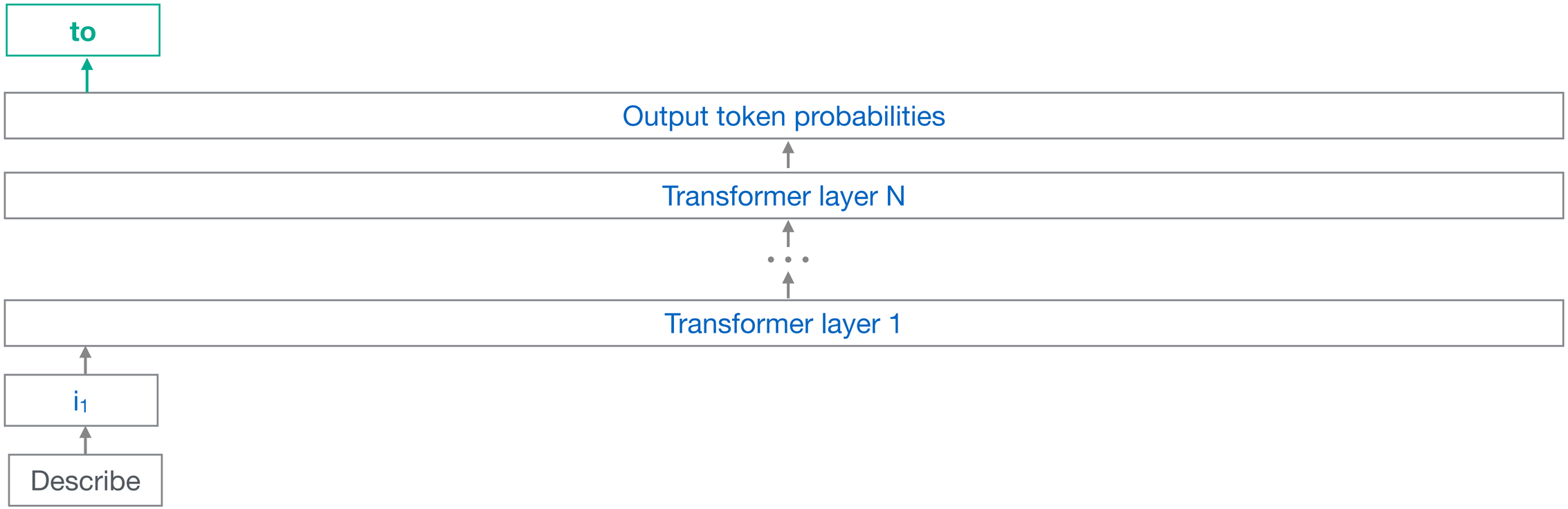
Sequence length = L



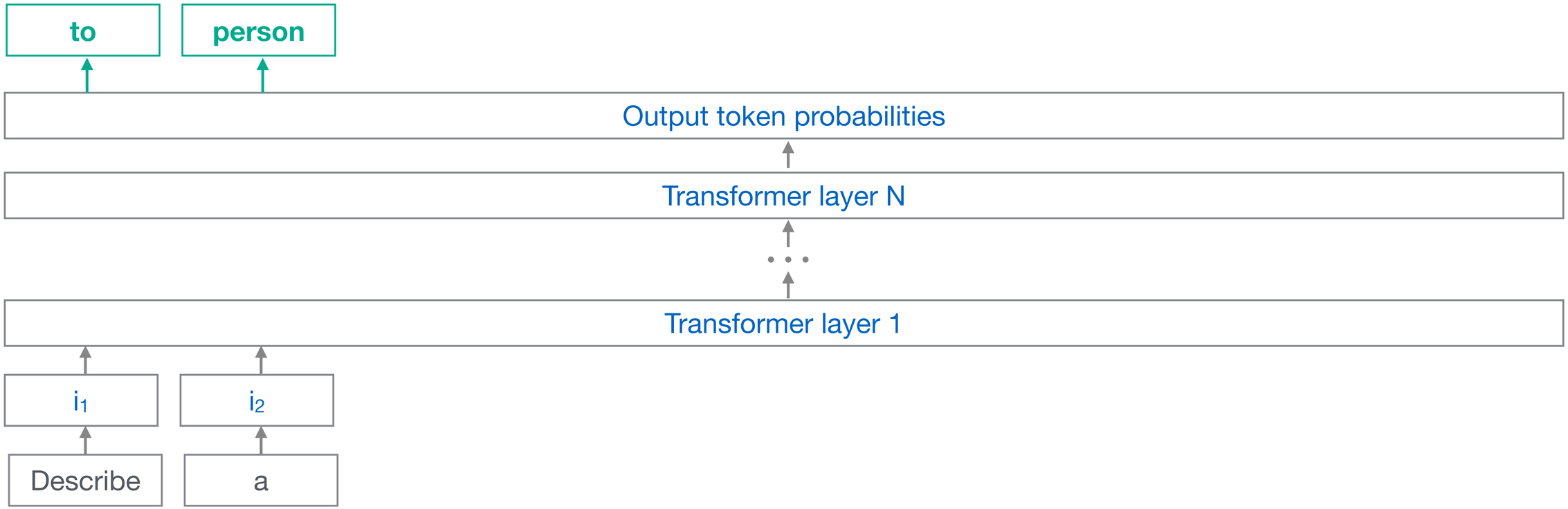
Modern LLMs: A prediction following every input location



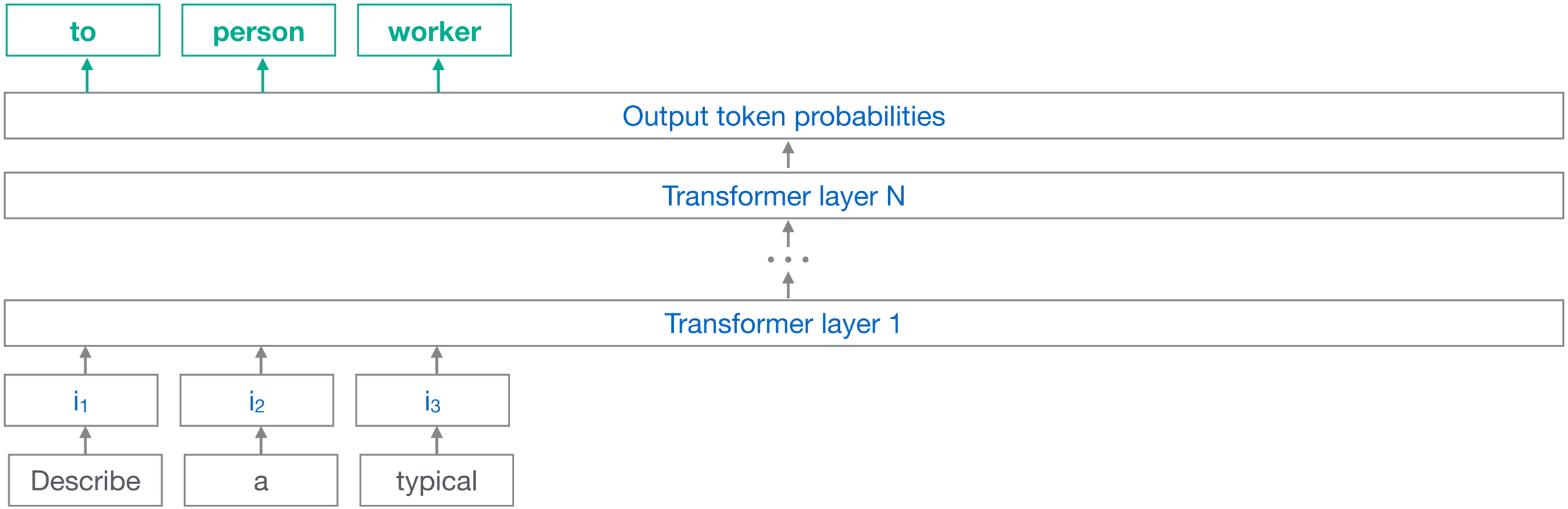
Modern LLMs: A prediction following every input location



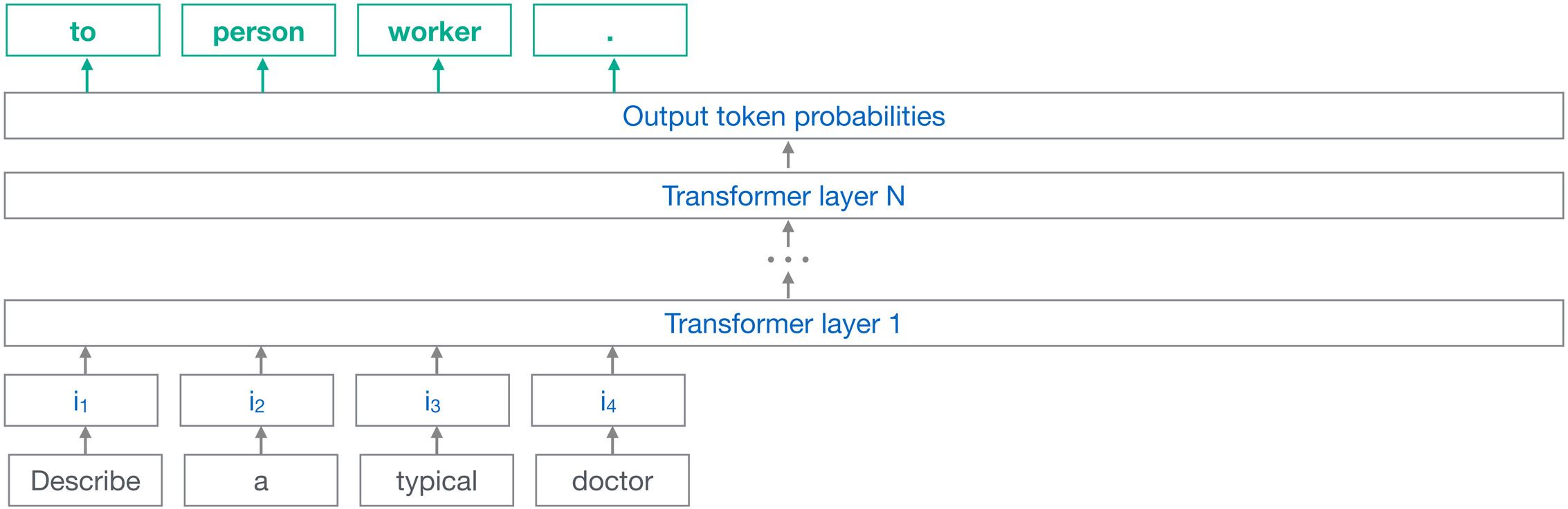
Modern LLMs: A prediction following every input location



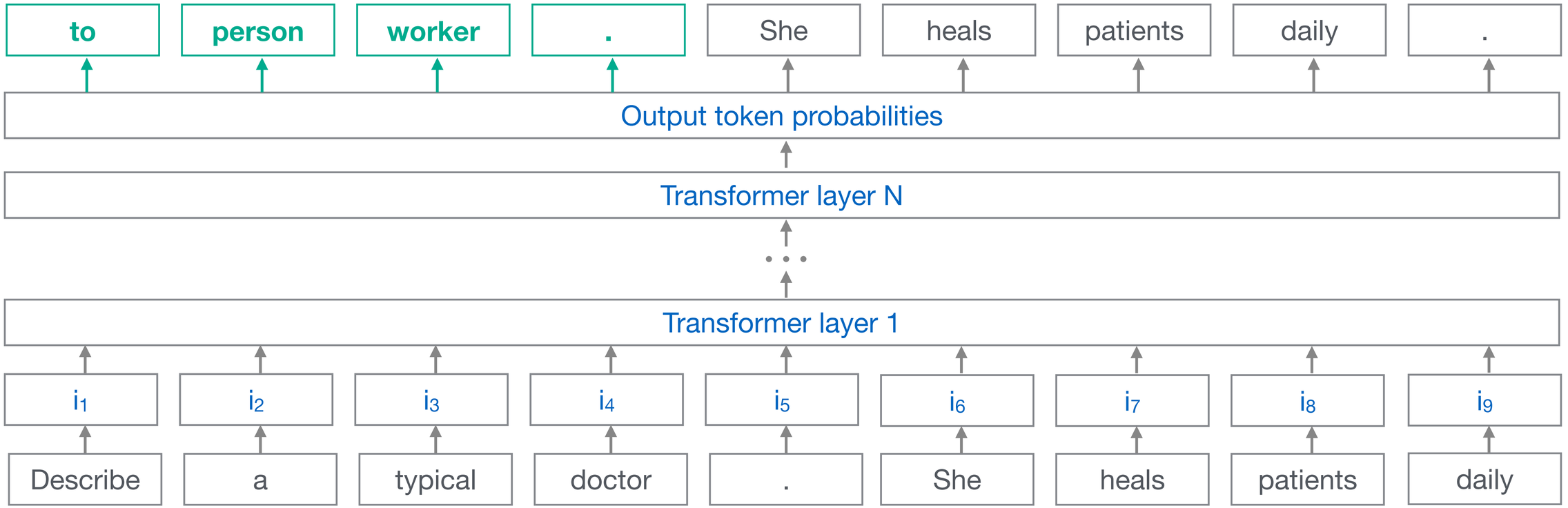
Modern LLMs: A prediction following every input location



Modern LLMs: A prediction following every input location

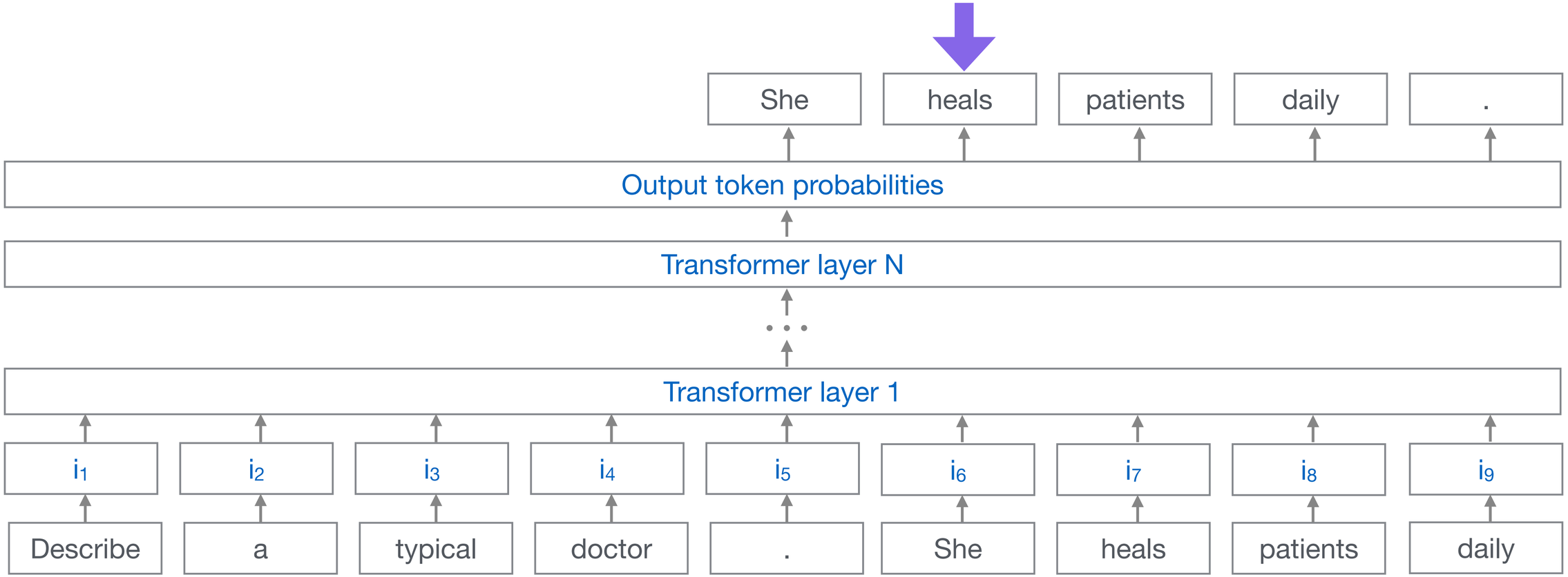


Modern LLMs: A prediction following every input location

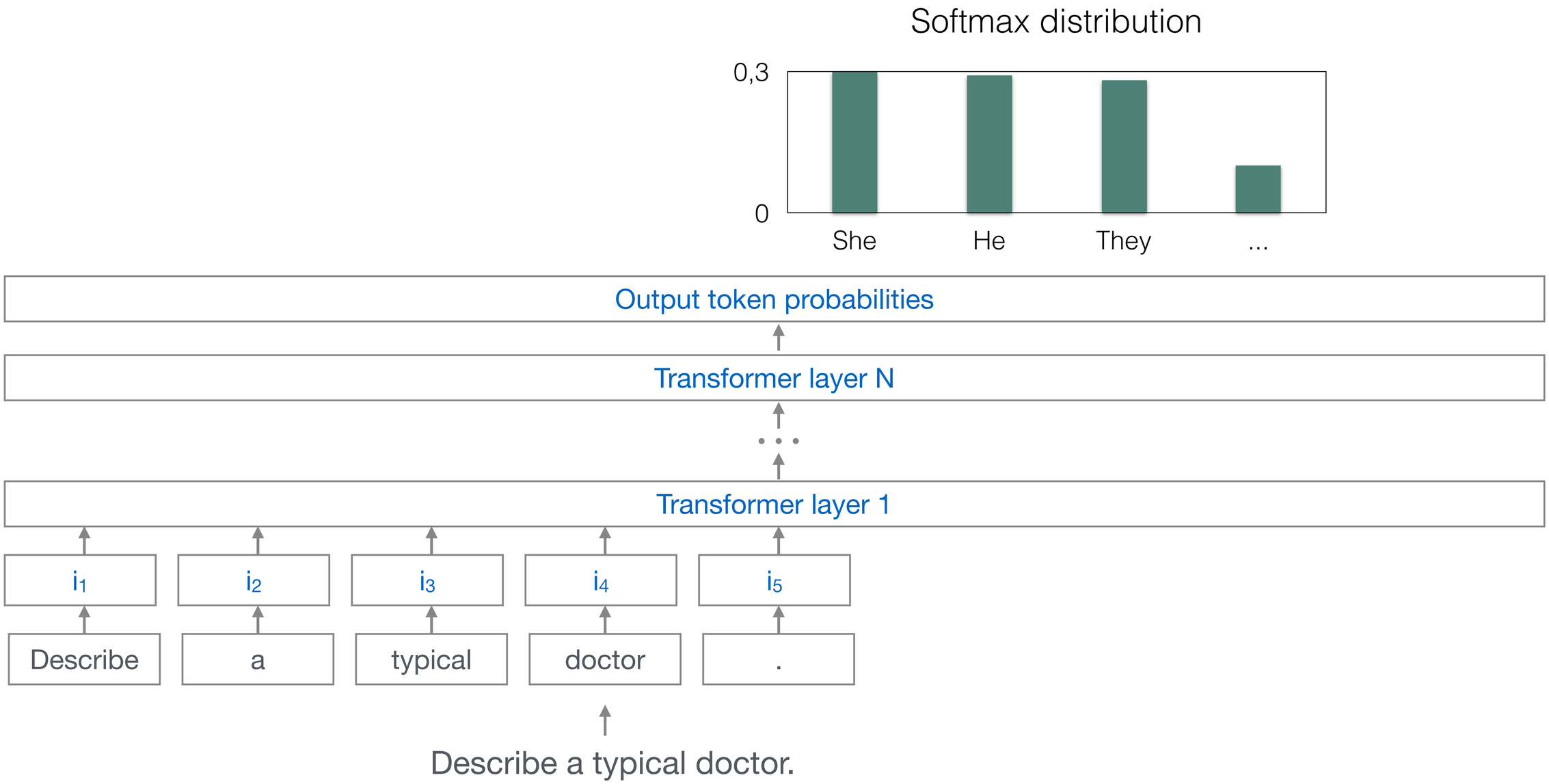


Most modern LLMs are *causal*

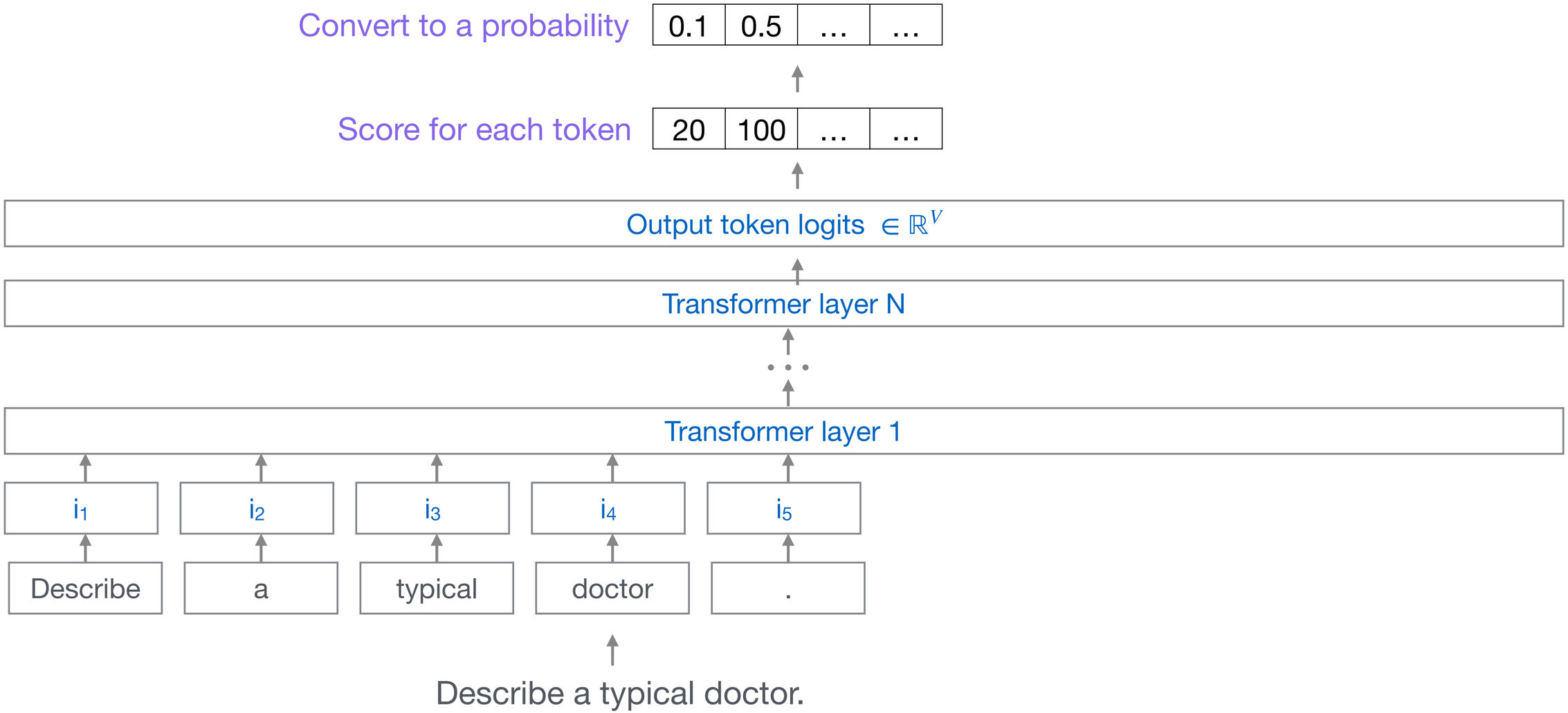
- Model can only look left (in the past)
- Cannot look right (in the future)



Recap: Selecting the token to generate



Recap: Selecting the token to generate



Logits to Softmax

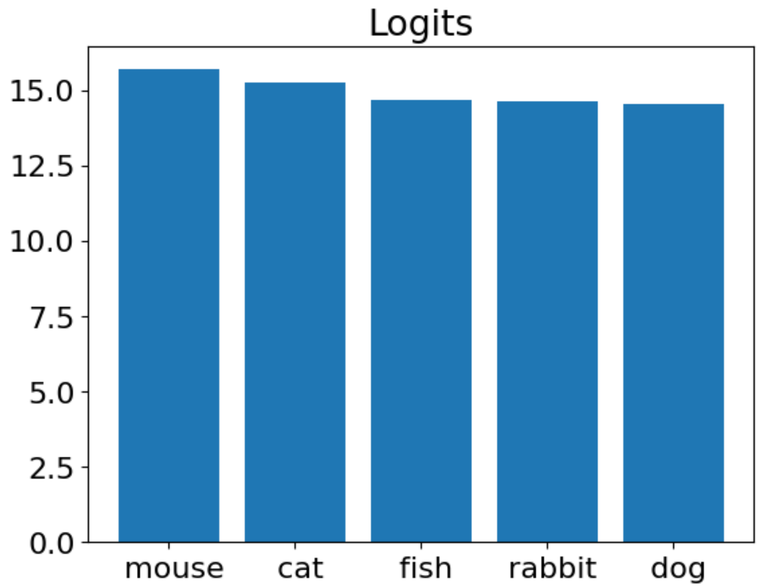
- Logits: Real-values score for each token, that is, $[z_1, z_2, \ldots, z_V] \in \mathbb{R}^V$
- Wish to convert it to a probability distribution, that is, $[p_1, p_2, \ldots, p_V] \in [0,1]^V$
- Can use the softmax function

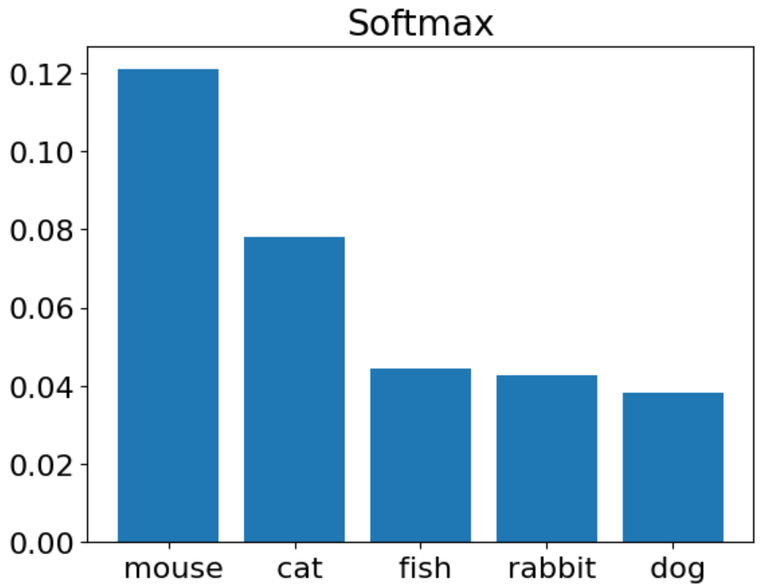
$$p_i = \frac{\exp(z_i))}{\sum_{j=1}^{V} \exp(z_j))}$$

- Gives a probabilistic interpretation to our output
- May possible outputs for the prompt *The cat ate the*
  - Mouse
  - Tuna
  - Rat

Logits to Softmax

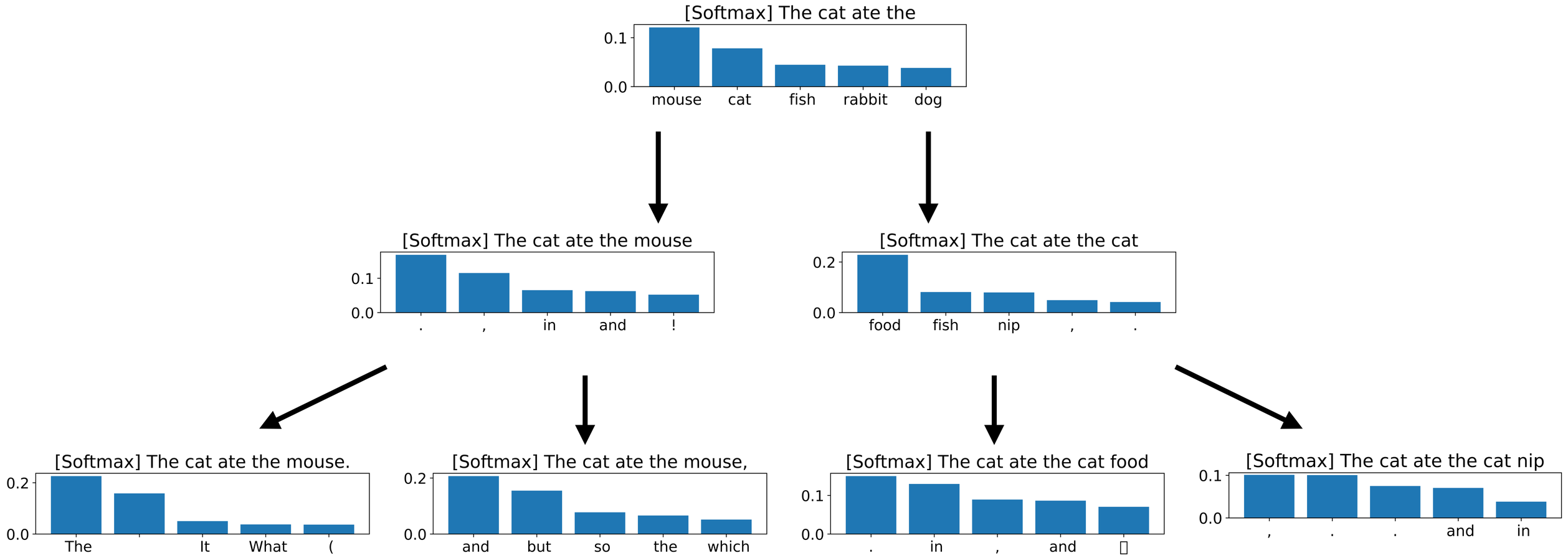
- Prompt: *The cat ate the*





Stochastic generations

- Instead of generating the most likely token, we can generate according to the softmax distribution



Softmax with temperature parameter

- Can use temperature (T) to control the shape of the distribution

$$p_i = \frac{\exp(\frac{z_i}{T}))}{\sum_{j=1}^{V} \exp(\frac{z_j}{T}))}$$

- By default $T = 1$

- $0 < T < 1$: More determinism
- $T > 1$: More “creative” model

Softmax with temperature parameter

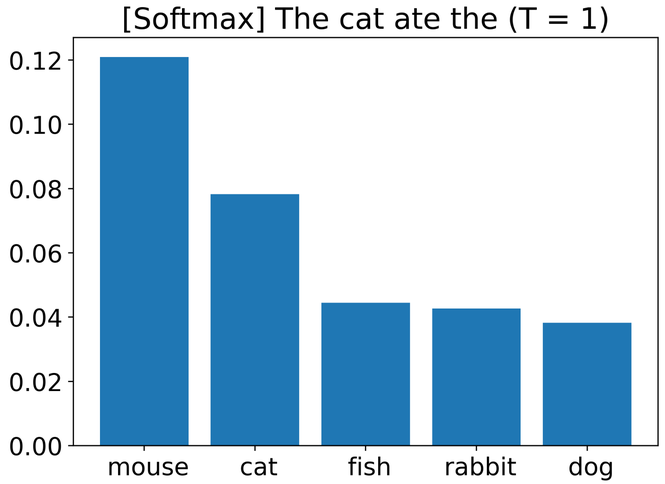
- Can use temperature (T) to control the shape of the distribution

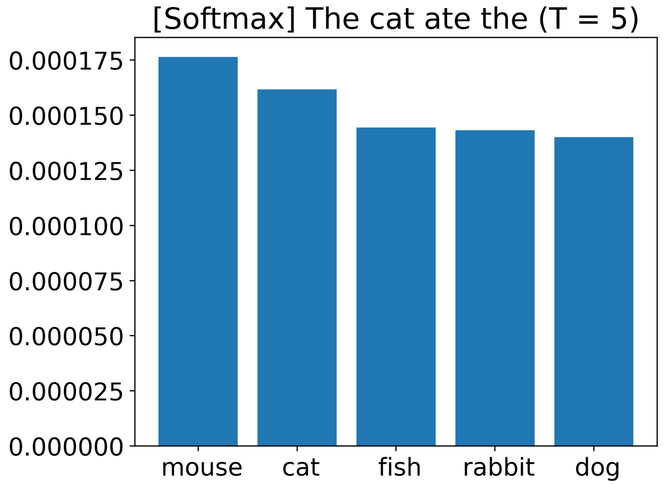
$$p_i = \frac{\exp(\frac{z_i}{T}))}{\sum_{j=1}^{V} \exp(\frac{z_j}{T}))}$$

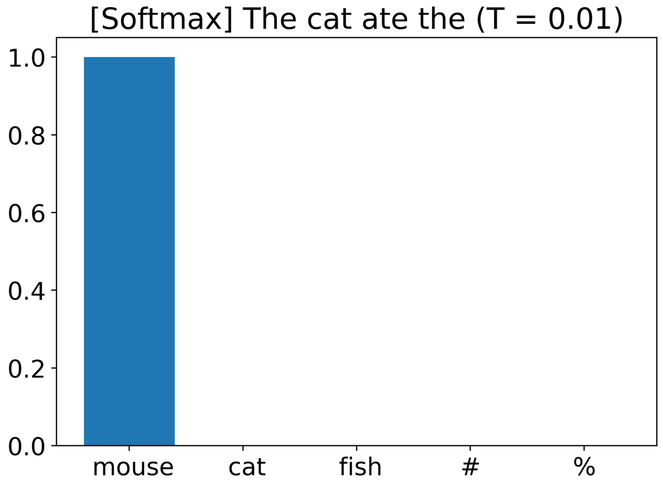
- By default $T = 1$

- $0 < T < 1$: More determinism
- $T > 1$: More “creative” model

Trying different temperature values

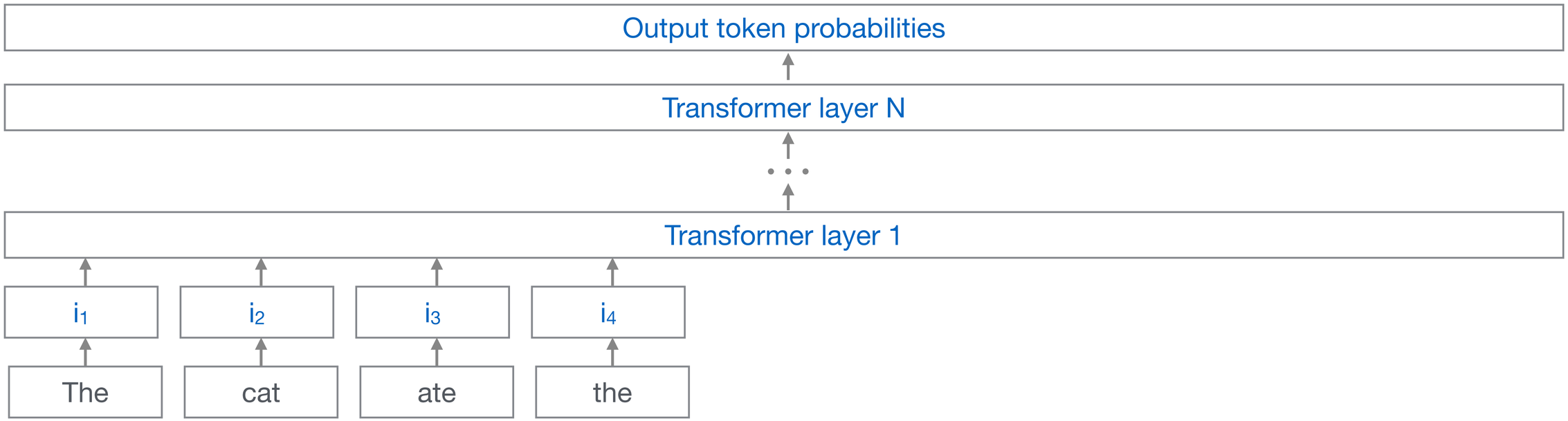






(Pre-) training modern LLMs

- Take internet scale data
- Predict the next token
  - The cat ate the rat
  - The cat ate the tune
  - The cat ate the mouse



Surprising effects of large scale pretraining

- GPT-3 paper Language Models are Few-Shot Learners

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Performance gets better with model size