Introduction to

Artificial Intelligence

with Python

Language

Natural Language Processing

Natural Language Processing

- automatic summarization
- information extraction
- machine translation
- question answering
- text classification
- ...

Syntax

"Just before nine o'clock Sherlock Holmes stepped briskly into the room."

"Just before Sherlock Holmes nine o'clock stepped briskly the room."

"I saw the man on the mountain with a telescope."

Semantics

"Just before nine o'clock Sherlock Holmes stepped briskly into the room."

"A few minutes before nine, Sherlock Holmes walked quickly into the room."

"Colorless green ideas sleep furiously."

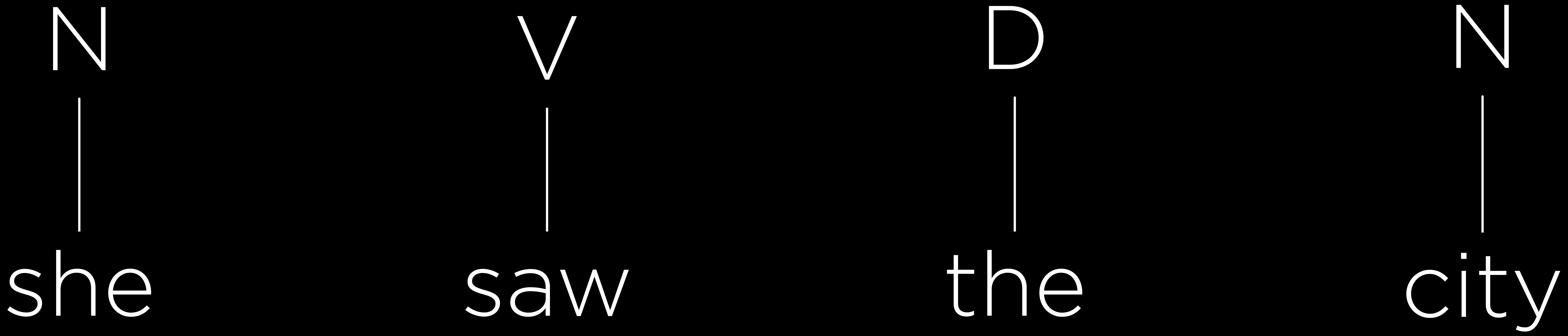
Natural Language Processing

formal grammar

a system of rules for generating sentences in a language

Context-Free Grammar

she saw the city



N V D N

she saw the city

N → she | city | car | Harry | ...

D → the | a | an | ...

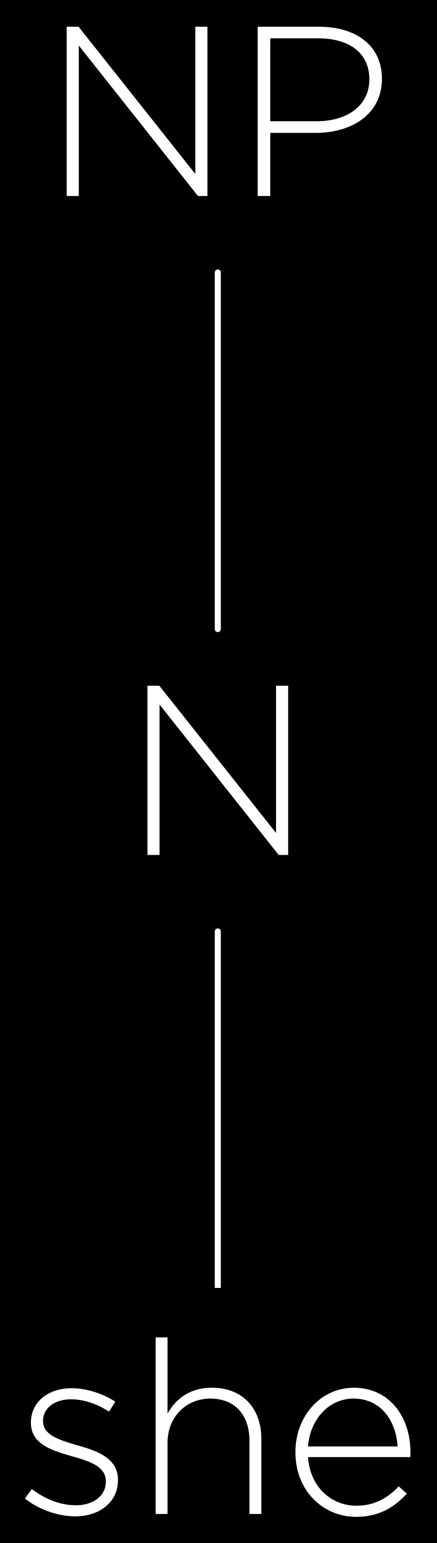
V → saw | ate | walked | ...

P → to | on | over | ...

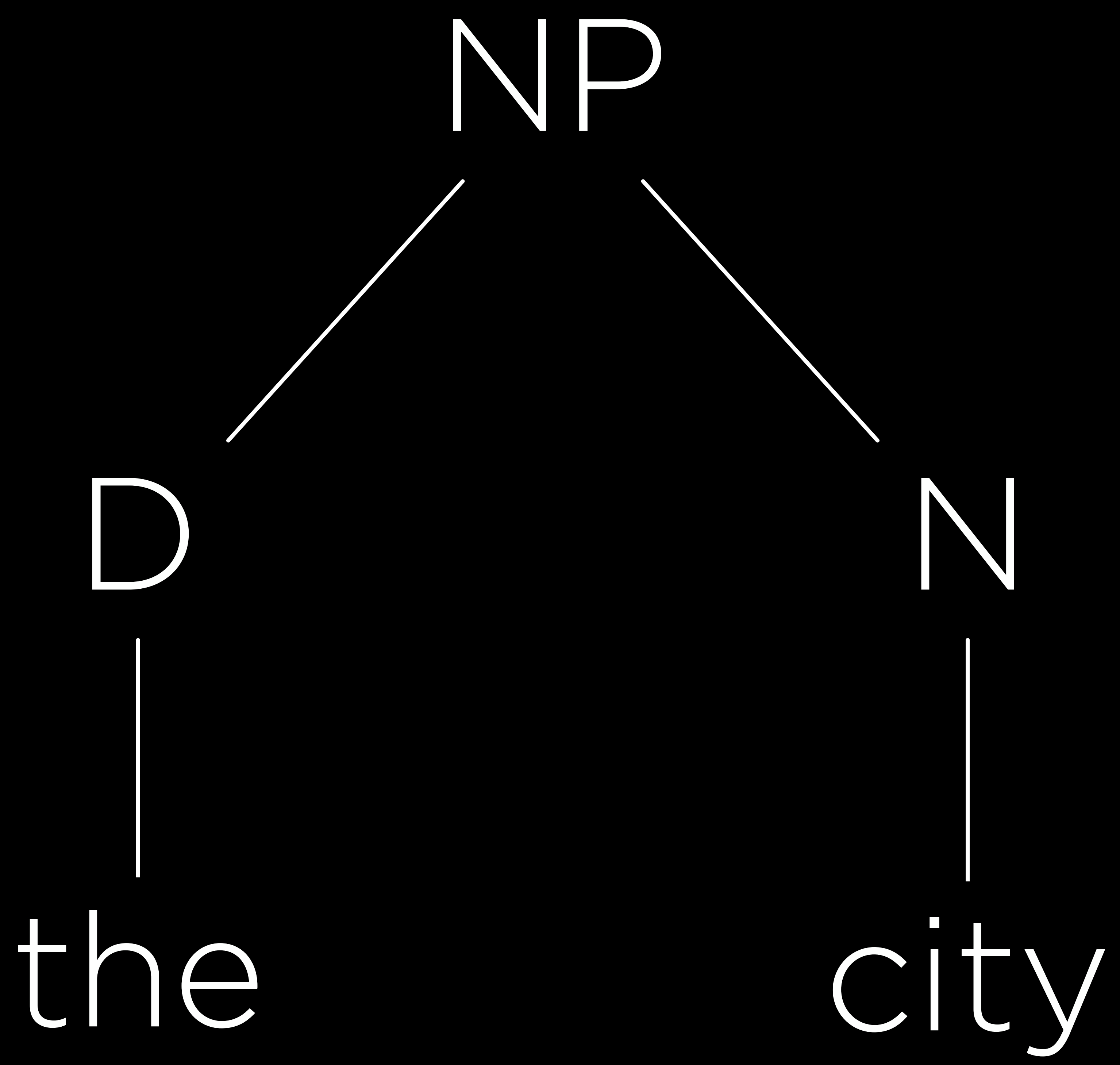
ADJ → blue | busy | old | ...

$$NP \rightarrow N \mid D\ N$$

NP → N | D N

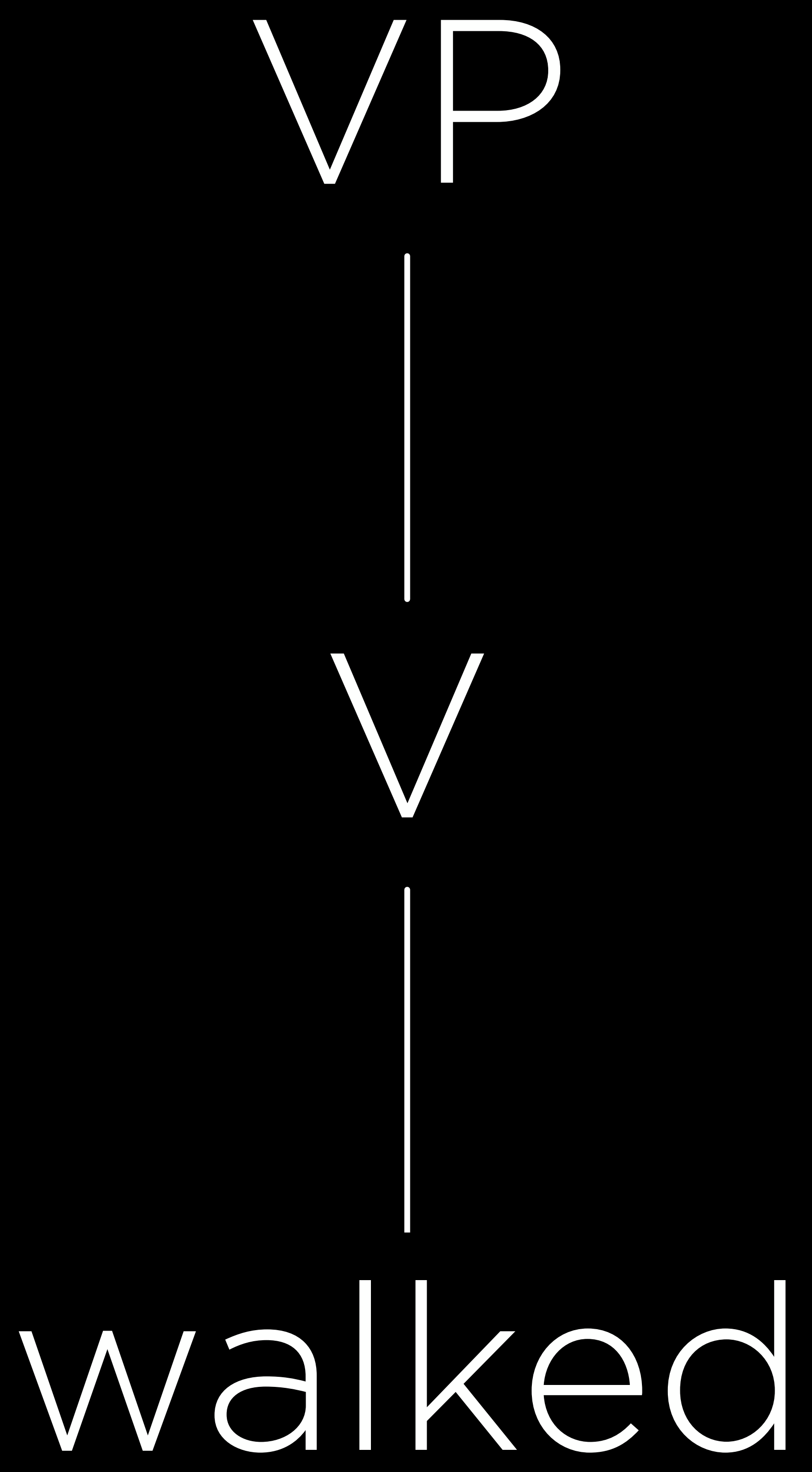


NP → N | D N

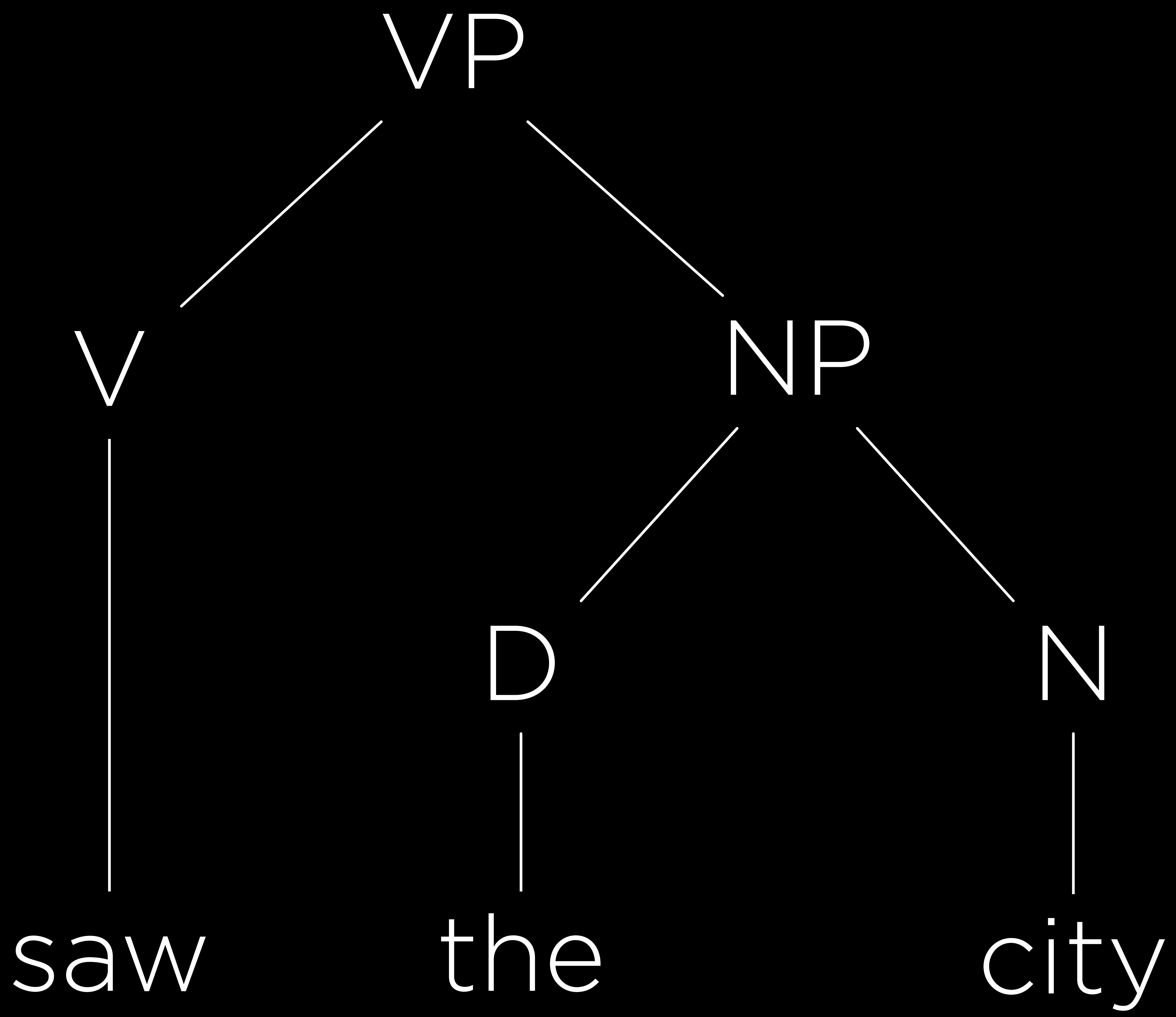


$$\mathrm{VP} \rightarrow \mathrm{V} \;\Big|\; \mathrm{V}\ \mathrm{NP}$$

VP → V | V NP

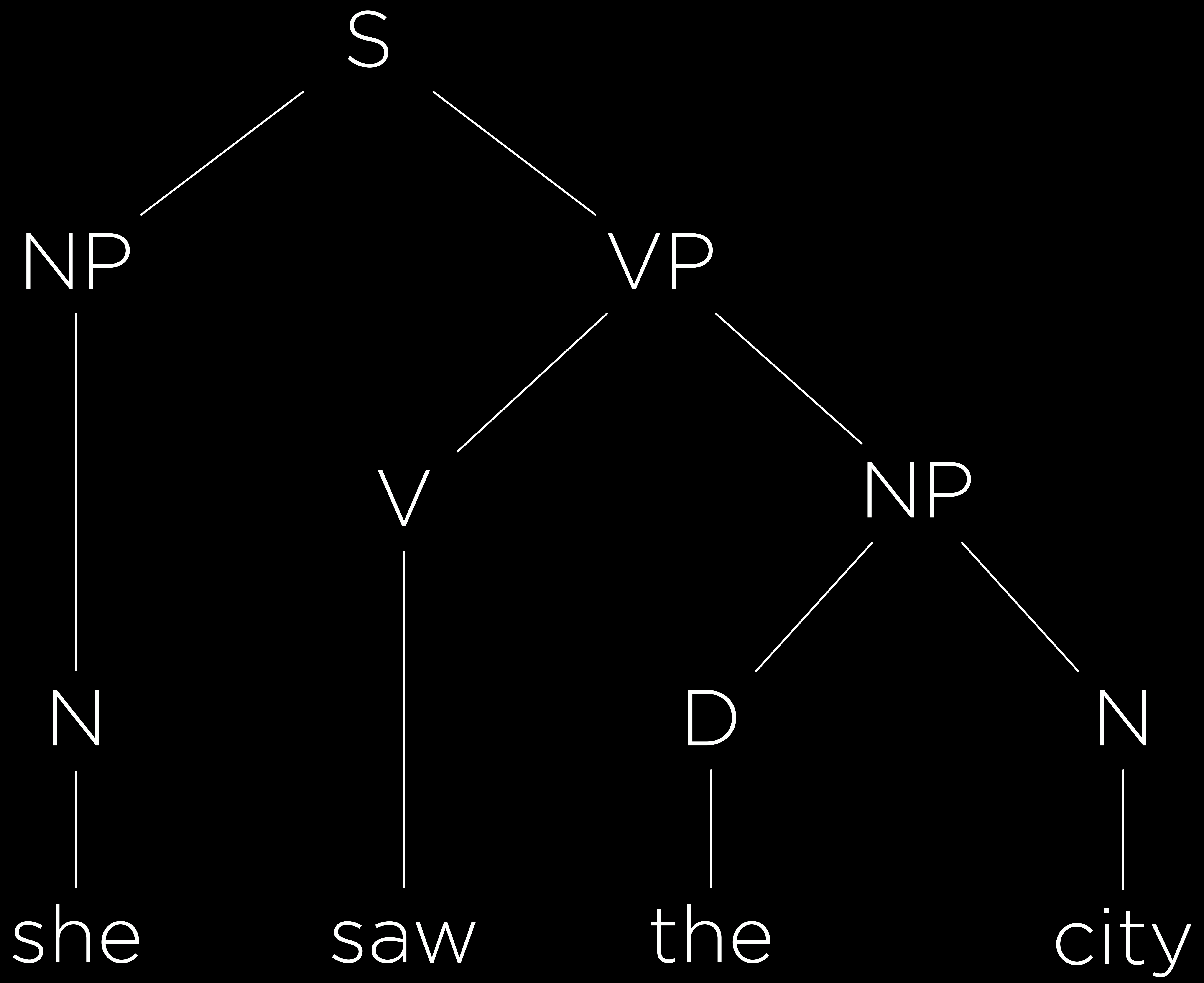


VP → V | V NP



S → NP VP

S → NP VP



nltk

n-gram

a contiguous sequence of *n* items
from a sample of text

"How often have I said to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"**How often have** I said to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How **often have I** said to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How often **have I said** to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How often have **I said to** you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

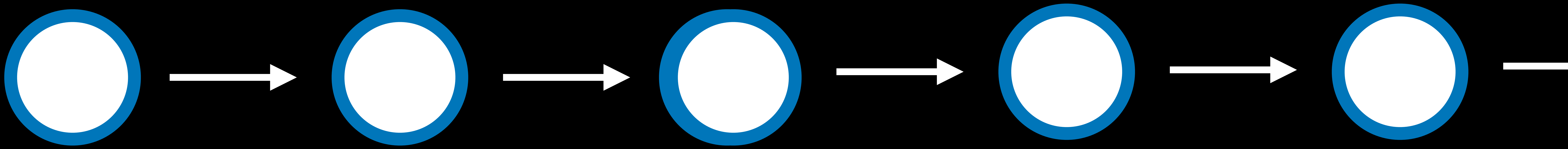
"How often have I **said to you** that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How often have I said **to you that** when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

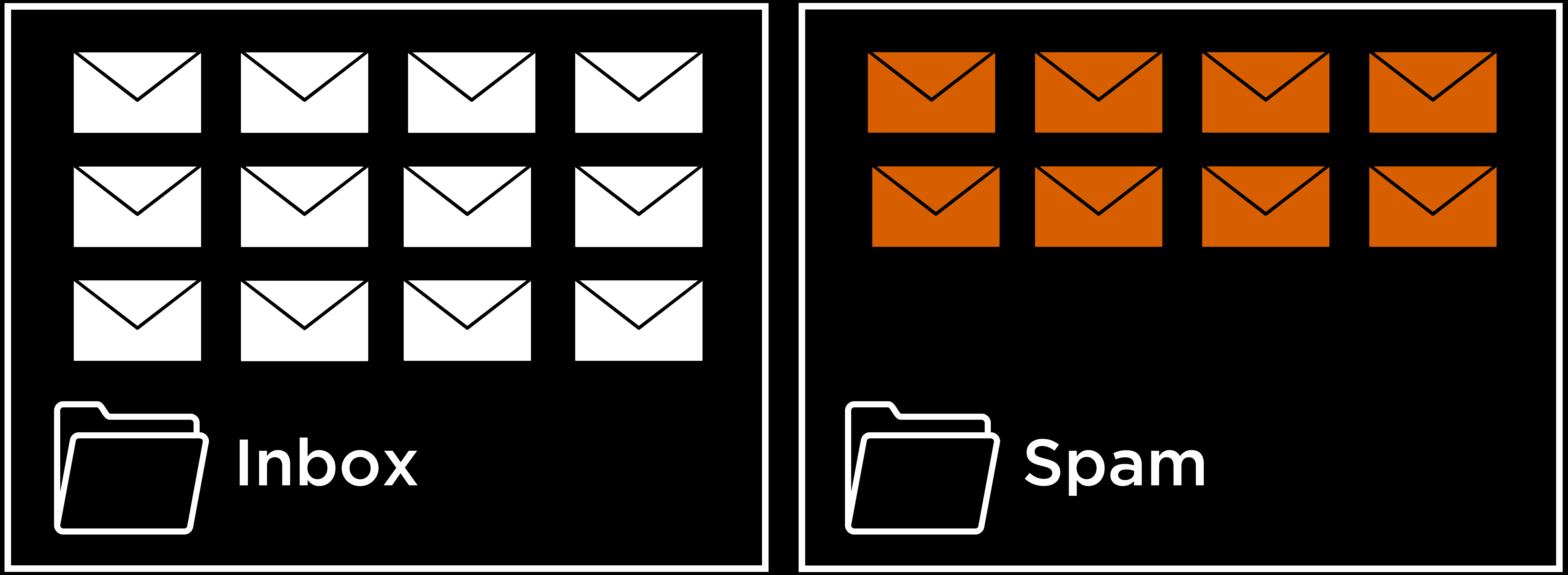
tokenization

the task of splitting a sequence of characters into pieces (tokens)

Markov Chains



Text Categorization



Inbox

Spam

😀 🙁

"My grandson loved it! So much fun!"

"Product broke after a few days."

"One of the best games I've played in a long time."

"Kind of cheap and flimsy, not worth it."

😀 "My grandson loved it! So much fun!"

🙁 "Product broke after a few days."

😀 "One of the best games I've played in a long time."

🙁 "Kind of cheap and flimsy, not worth it."

😀 "My grandson **loved** it! So much **fun**!"

🙁 "Product **broke** after a few days."

😀 "One of the **best** games I've played in a long time."

🙁 "Kind of **cheap** and **flimsy**, not worth it."

bag-of-words model

model that represents text as an unordered collection of words

Naive Bayes

Bayes' Rule

$$P(b \quad a) = \frac{P(a \quad b)\, P(b)}{P(a)}$$

$$P(\text{Positive})$$

$$P(\text{Negative})$$

P(😀)

P(🙁)

"My grandson loved it!"

P(😀)

P(😀 | "my grandson loved it")

$$P(\text{😀} \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

$$P(\text{😀} \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

$$P(\text{😀} \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

equal to

$$\frac{P(\text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"} \mid \text{😀})\,P(\text{😀})}{P(\text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})}$$

$$P(\text{😀} \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

proportional to

$$P(\text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"} \mid \text{😀})\, P(\text{😀})$$

P(😀 | "my", "grandson", "loved", "it")

proportional to

P(😀, "my", "grandson", "loved", "it")

$$P(\text{😀} \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

naively proportional to

$$P(\text{😀})P(\text{"my"} \mid \text{😀})P(\text{"grandson"} \mid \text{😀})$$
$$P(\text{"loved"} \mid \text{😀})\ P(\text{"it"} \mid \text{😀})$$

$$P(\text{😀}) = \frac{\text{number of positive samples}}{\text{number of total samples}}$$

$$P(\text{"loved"} \mid \text{😀}) = \frac{\text{number of positive samples with "loved"}}{\text{number of positive samples}}$$

$$P(\text{😀})P(\text{"my"} \mid \text{😀})P(\text{"grandson"} \mid \text{😀})$$

$$P(\text{"loved"} \mid \text{😀})\ P(\text{"it"} \mid \text{😀})$$

😀	🙁
0.49	0.51

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{😀})P(\text{"my"} \mid \text{😀})P(\text{"grandson"} \mid \text{😀})$$

$$P(\text{"loved"} \mid \text{😀})\ P(\text{"it"} \mid \text{😀})$$

😀	🙁
0.49	0.51

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

P(😀)*P*("my" | 😀)*P*("grandson" | 😀)
P("loved" | 😀) *P*("it" | 😀)

😀	🙁
0.49	0.51

😀 `0.00014112`

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{😀})P(\text{"my"} \mid \text{😀})P(\text{"grandson"} \mid \text{😀})$$
$$P(\text{"loved"} \mid \text{😀})\ P(\text{"it"} \mid \text{😀})$$

😀	🙁
0.49	0.51

😀 0.00014112

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

P(🙁)*P*("my" | 🙁)*P*("grandson" | 🙁)
P("loved" | 🙁) *P*("it" | 🙁)

😀	🙁
0.49	0.51

😀 0.00014112

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

P(🙁)P("my" | 🙁)P("grandson" | 🙁)

P("loved" | 🙁) P("it" | 🙁)

😀	🙁
0.49	0.51

😀 0.00014112

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{🙁})P(\text{"my"} \mid \text{🙁})P(\text{"grandson"} \mid \text{🙁})$$

$$P(\text{"loved"} \mid \text{🙁})\ P(\text{"it"} \mid \text{🙁})$$

😀	🙁
0.49	0.51

😀 `0.00014112`

🙁 `0.00006528`

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{🙁})P(\text{"my"} \mid \text{🙁})P(\text{"grandson"} \mid \text{🙁})$$
$$P(\text{"loved"} \mid \text{🙁})\ P(\text{"it"} \mid \text{🙁})$$

😀	🙁
0.49	0.51

😀 0.00014112

🙁 0.00006528

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{🙁})P(\text{"my"} \mid \text{🙁})P(\text{"grandson"} \mid \text{🙁})$$

$$P(\text{"loved"} \mid \text{🙁})\ P(\text{"it"} \mid \text{🙁})$$

😀	🙁
0.49	0.51

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

😀 0.6837

🙁 0.3163

$$P(\text{🙁})P(\text{"my"} \mid \text{🙁})P(\text{"grandson"} \mid \text{🙁})$$

$$P(\text{"loved"} \mid \text{🙁})\ P(\text{"it"} \mid \text{🙁})$$

😀	🙁
0.49	0.51

	😀	🙁
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$P($🙁$)P($"my" | 🙁$)P($"grandson" | 🙁$)$
$P($"loved" | 🙁$)$ $P($"it" | 🙁$)$

😀	🙁
0.49	0.51

	😀	🙁
my	0.30	0.20
grandson	0.00	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{🙁})P(\text{"my"} \mid \text{🙁})P(\text{"grandson"} \mid \text{🙁})$$

$$P(\text{"loved"} \mid \text{🙁})\ P(\text{"it"} \mid \text{🙁})$$

😀	🙁
0.49	0.51

😀 0.00000000

🙁 0.00006528

	😀	🙁
my	0.30	0.20
grandson	0.00	0.02
loved	0.32	0.08
it	0.30	0.40

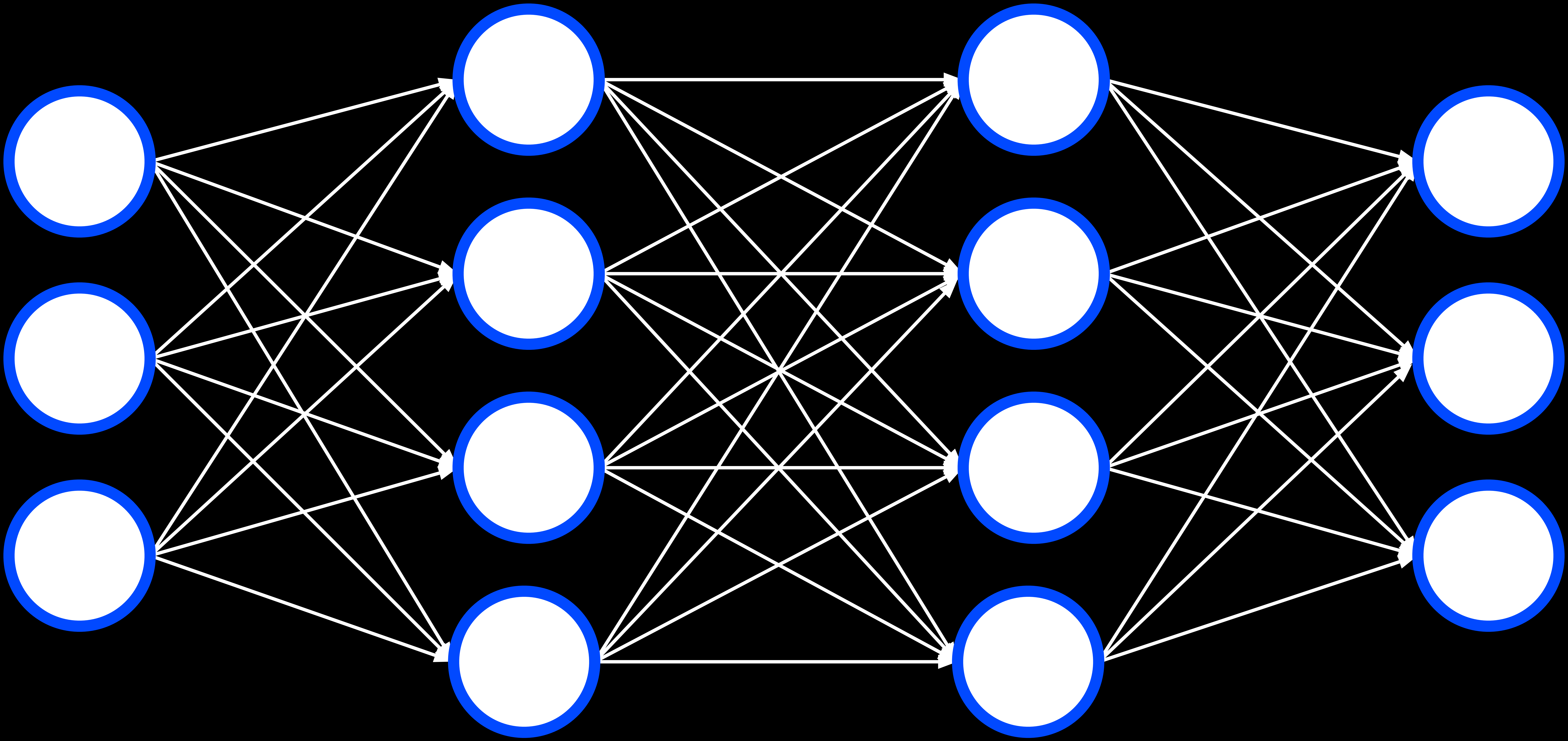
additive smoothing

adding a value α to each value in our distribution to smooth the data

Laplace smoothing

adding 1 to each value in our distribution: pretending we've seen each value one more time than we actually have

Word Representation



"He wrote a book."

```
he    [1, 0, 0, 0]
wrote [0, 1, 0, 0]
a     [0, 0, 1, 0]
book  [0, 0, 0, 1]
```

one-hot representation

representation of meaning as a vector with a single 1, and with other values as 0

"He wrote a book."

```
he    [1, 0, 0, 0]
wrote [0, 1, 0, 0]
a     [0, 0, 1, 0]
book  [0, 0, 0, 1]
```

"He wrote a book."

```
he    [1, 0, 0, 0, 0, 0, 0, ...]
wrote [0, 1, 0, 0, 0, 0, 0, ...]
a     [0, 0, 1, 0, 0, 0, 0, ...]
book  [0, 0, 0, 1, 0, 0, 0, ...]
```

"He wrote a book."

"He authored a novel."

```
wrote     [0, 1, 0, 0, 0, 0, 0, ...]
authored  [0, 0, 0, 0, 1, 0, 0, ...]

book      [0, 0, 0, 1, 0, 0, 0, ...]
novel     [0, 0, 0, 0, 0, 0, 1, ...]
```

distributed representation

representation of meaning distributed across multiple values

"He wrote a book."

he	[-0.34, -0.08, 0.02, -0.18, 0.22, ...]
wrote	[-0.27, 0.40, 0.00, -0.65, -0.15, ...]
a	[-0.12, -0.25, 0.29, -0.09, 0.40, ...]
book	[-0.23, -0.16, -0.05, -0.57, 0.05, ...]

"You shall know a word by the company it keeps."

J. R. Firth, 1957

for		he	ate

for	**breakfast**	he	ate

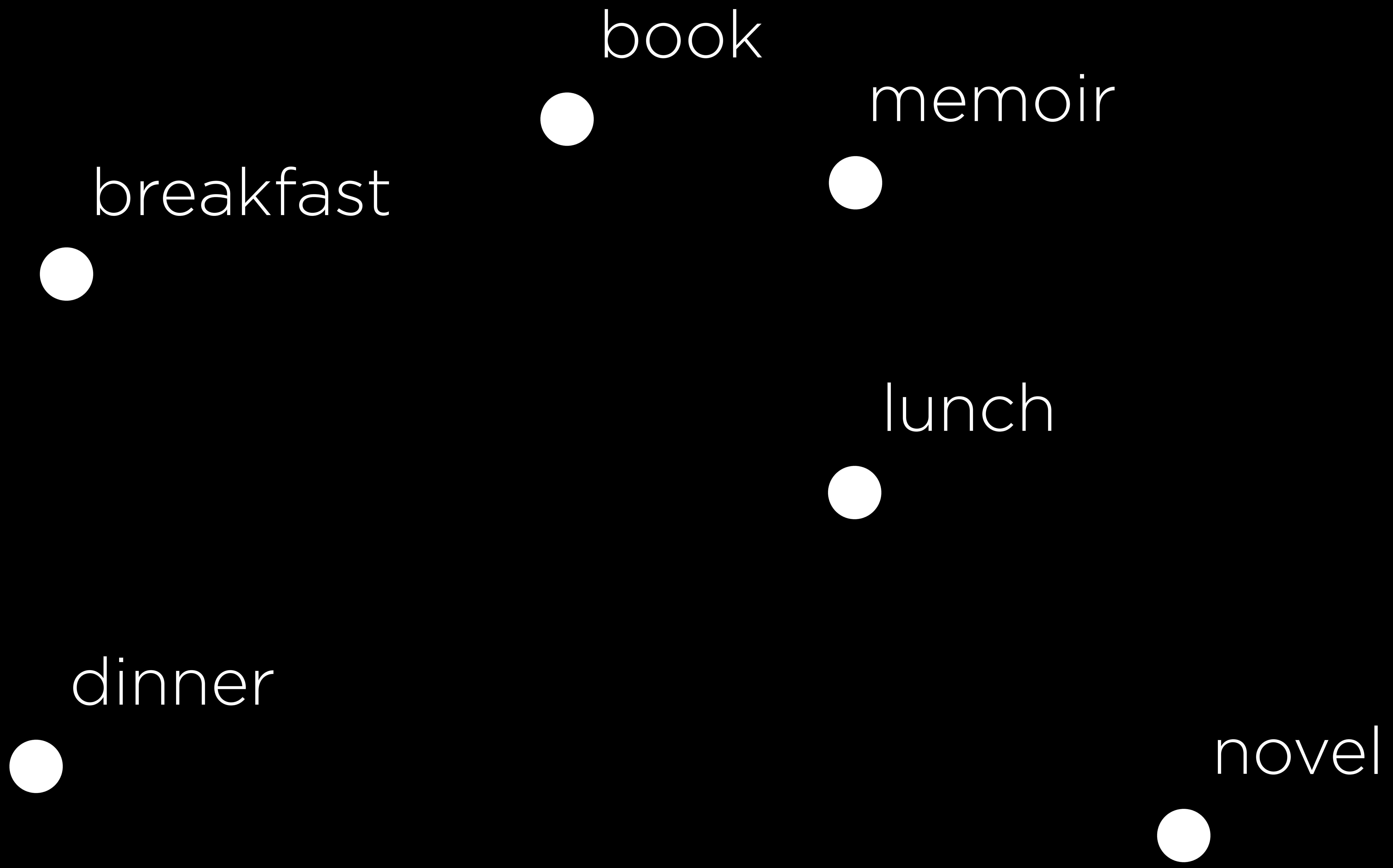
for	**lunch**	he	ate

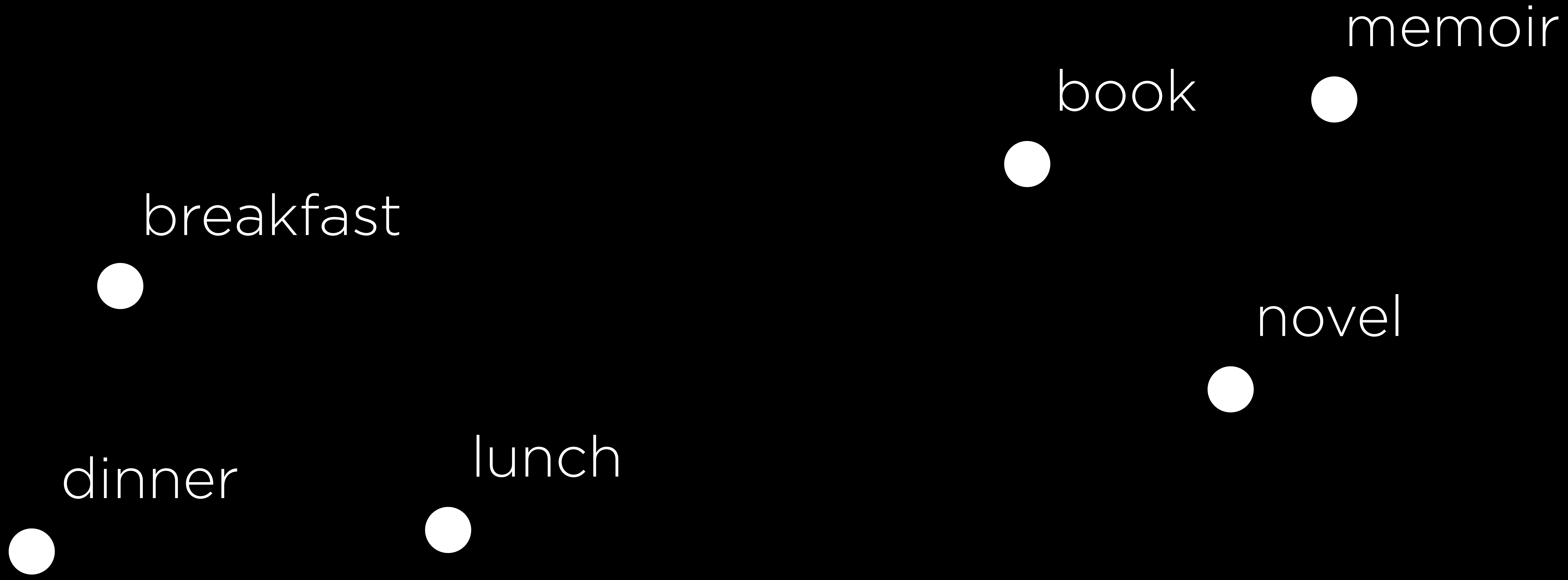
for	**dinner**	he	ate

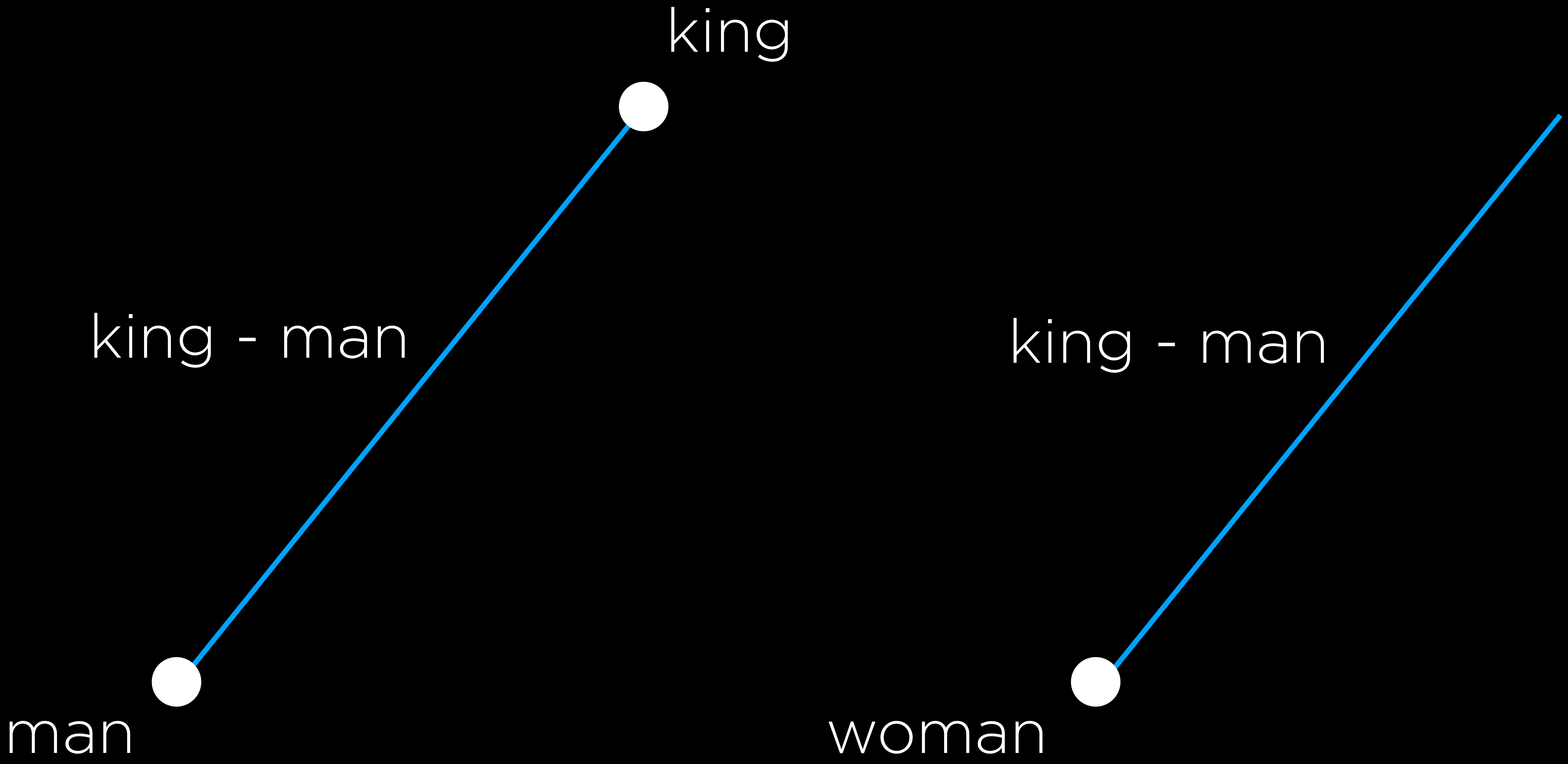
for		he	ate

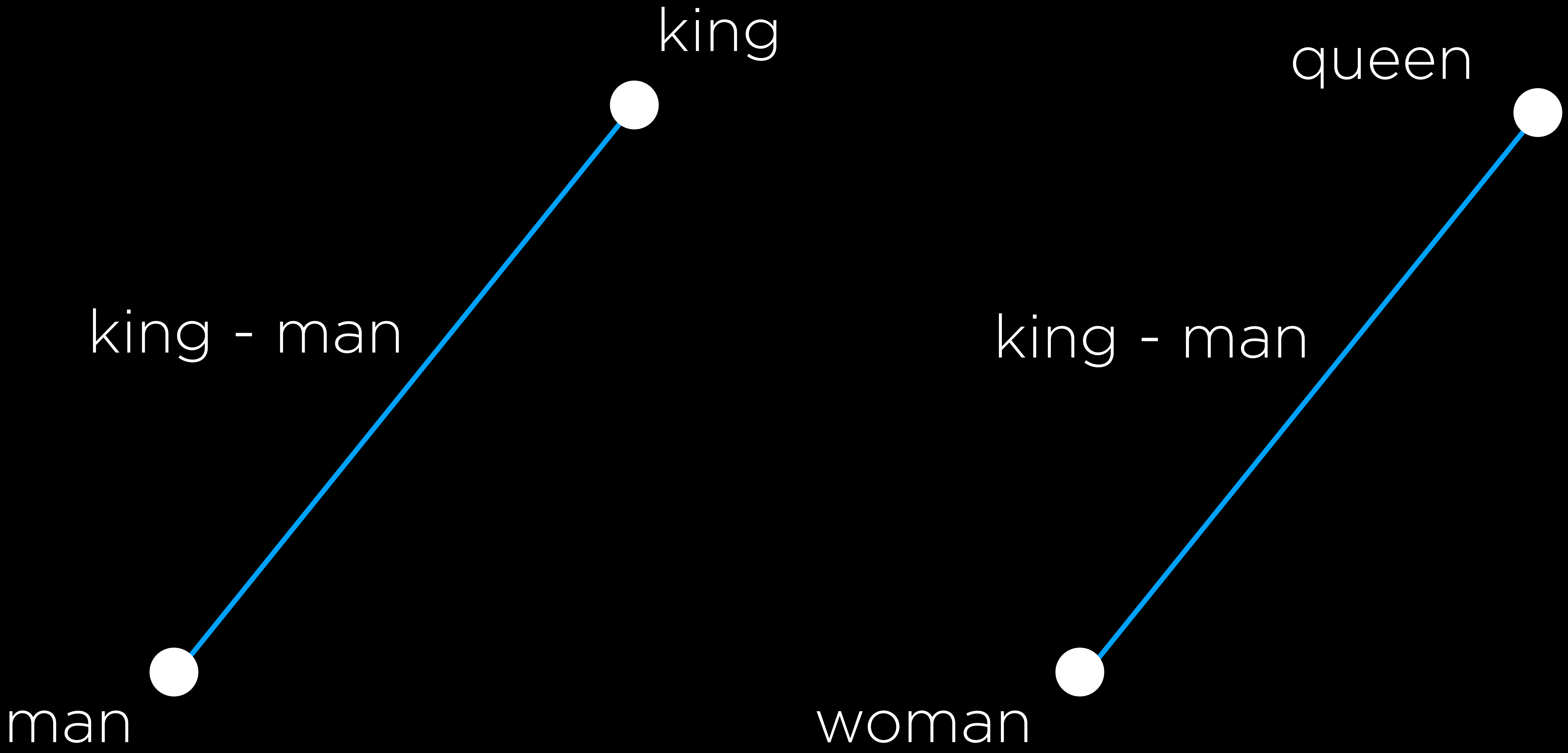
word2vec

model for generating word vectors

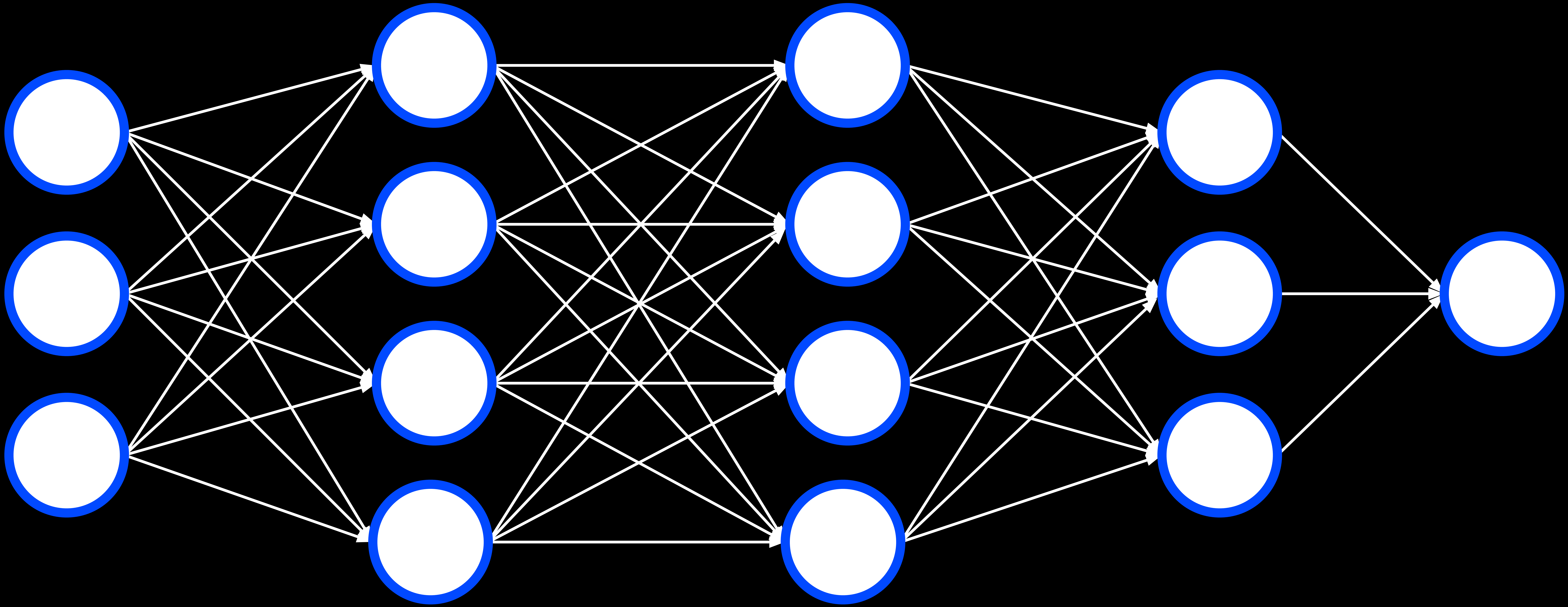

book
memoir
breakfast
lunch
dinner
novel

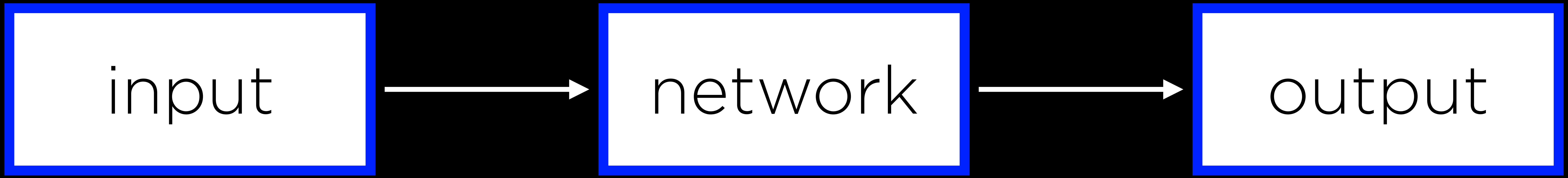

memoir
book
breakfast
novel
dinner
lunch

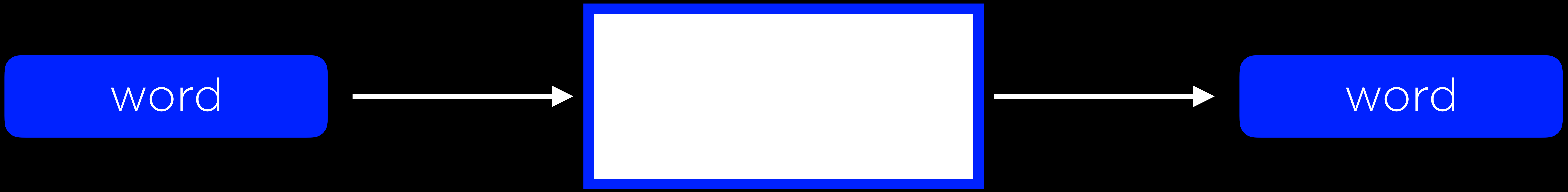

king
king - man
man
king - man
woman


king
queen
king - man
king - man
man
woman

Neural Networks

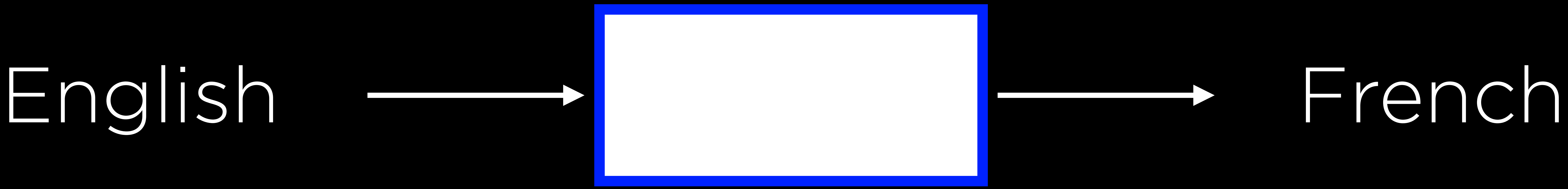



input
network
output



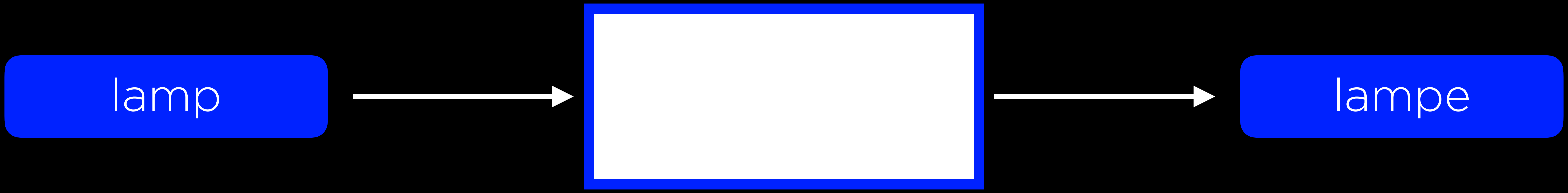
word

word



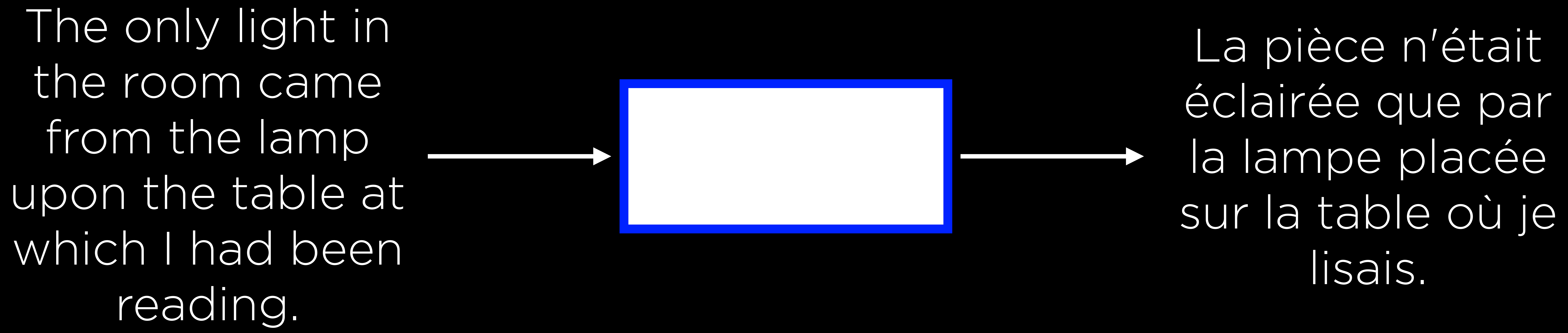
English

French



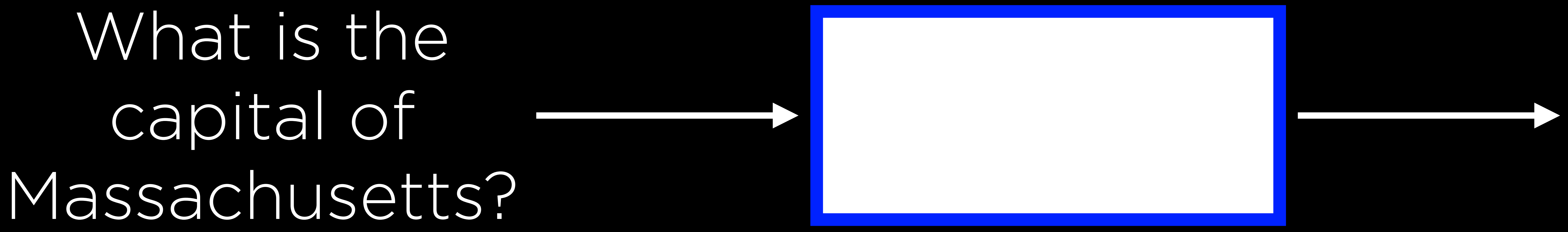
lamp

lampe



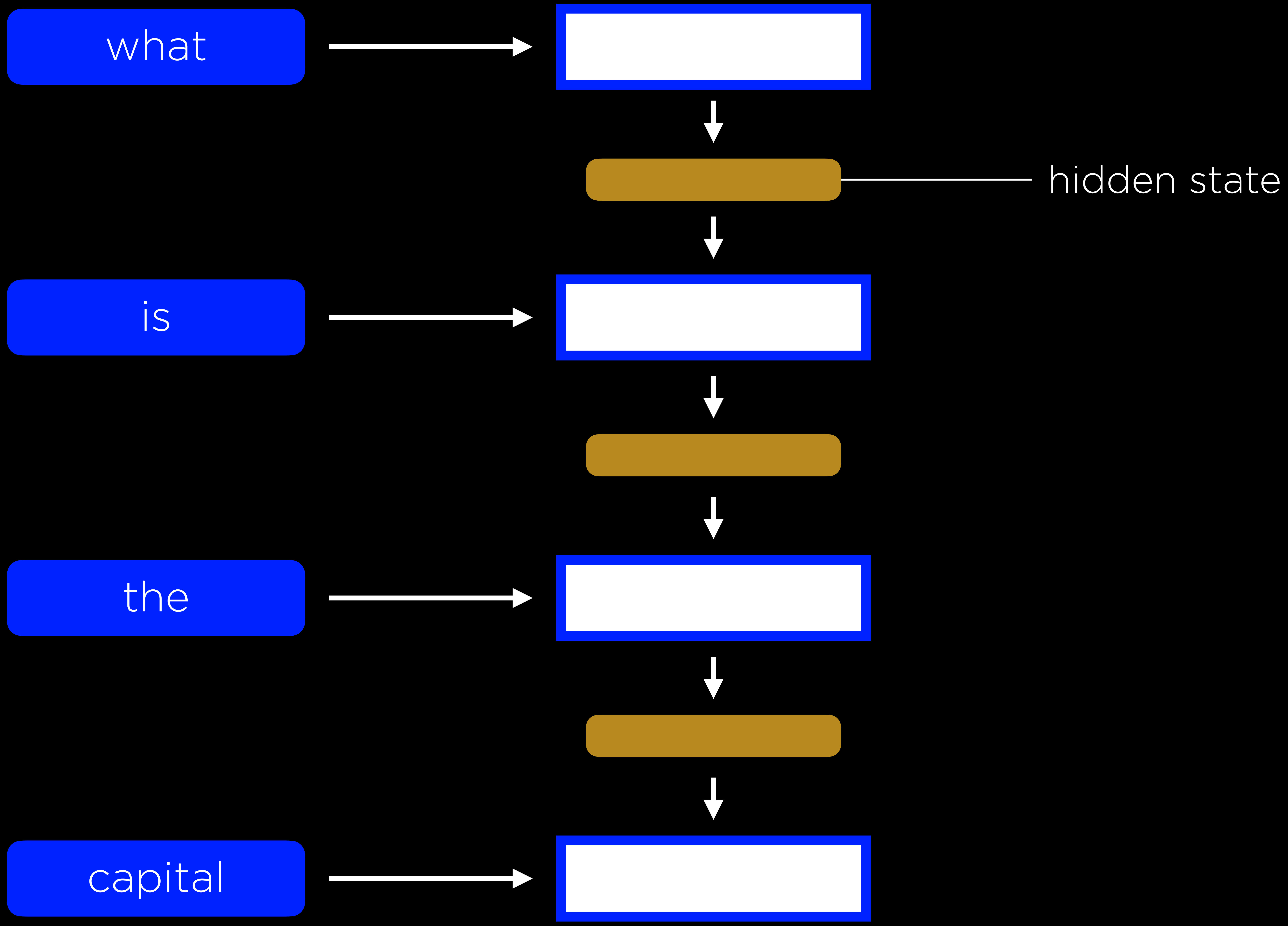
The only light in the room came from the lamp upon the table at which I had been reading.

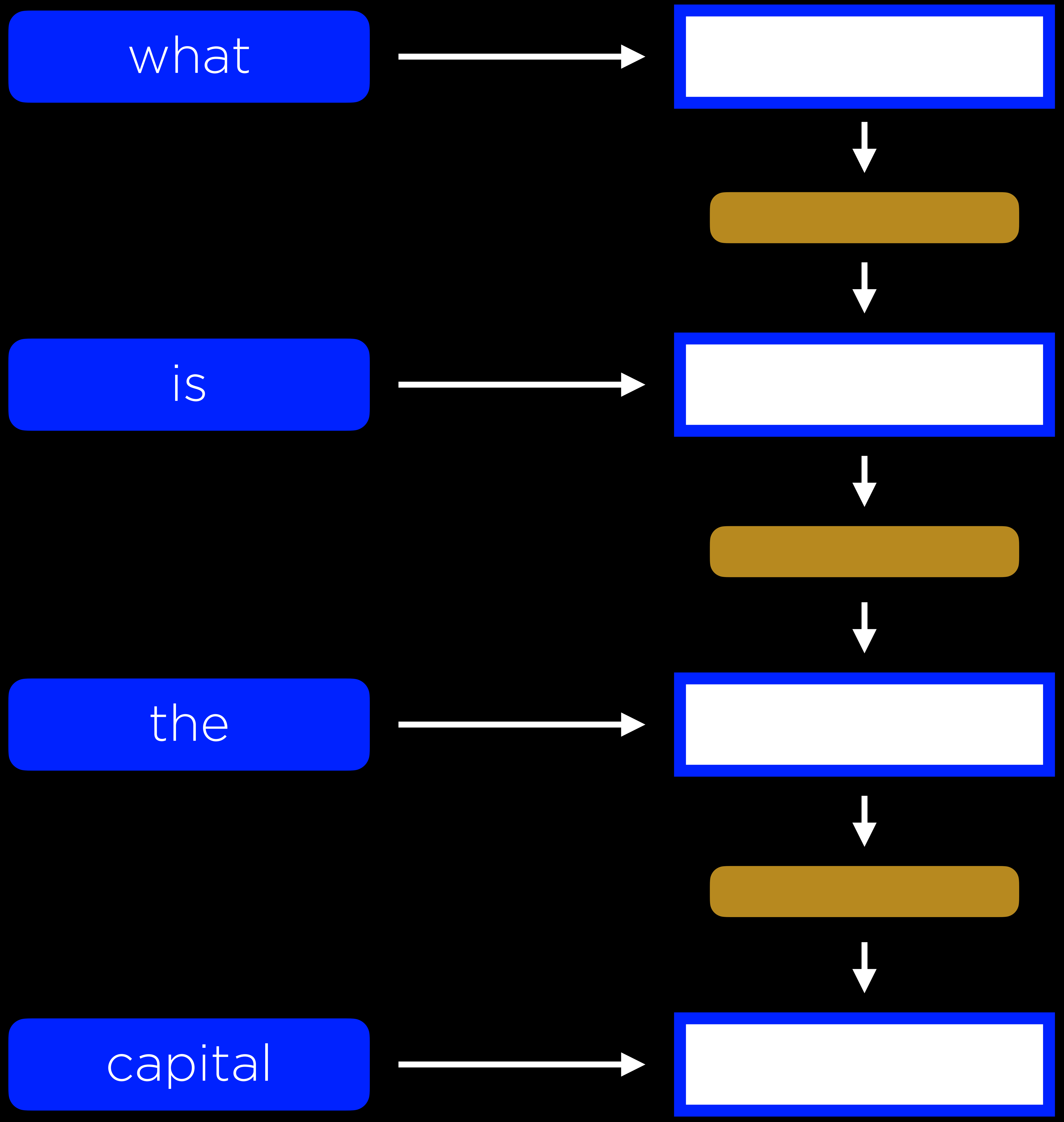
La pièce n'était éclairée que par la lampe placée sur la table où je lisais.

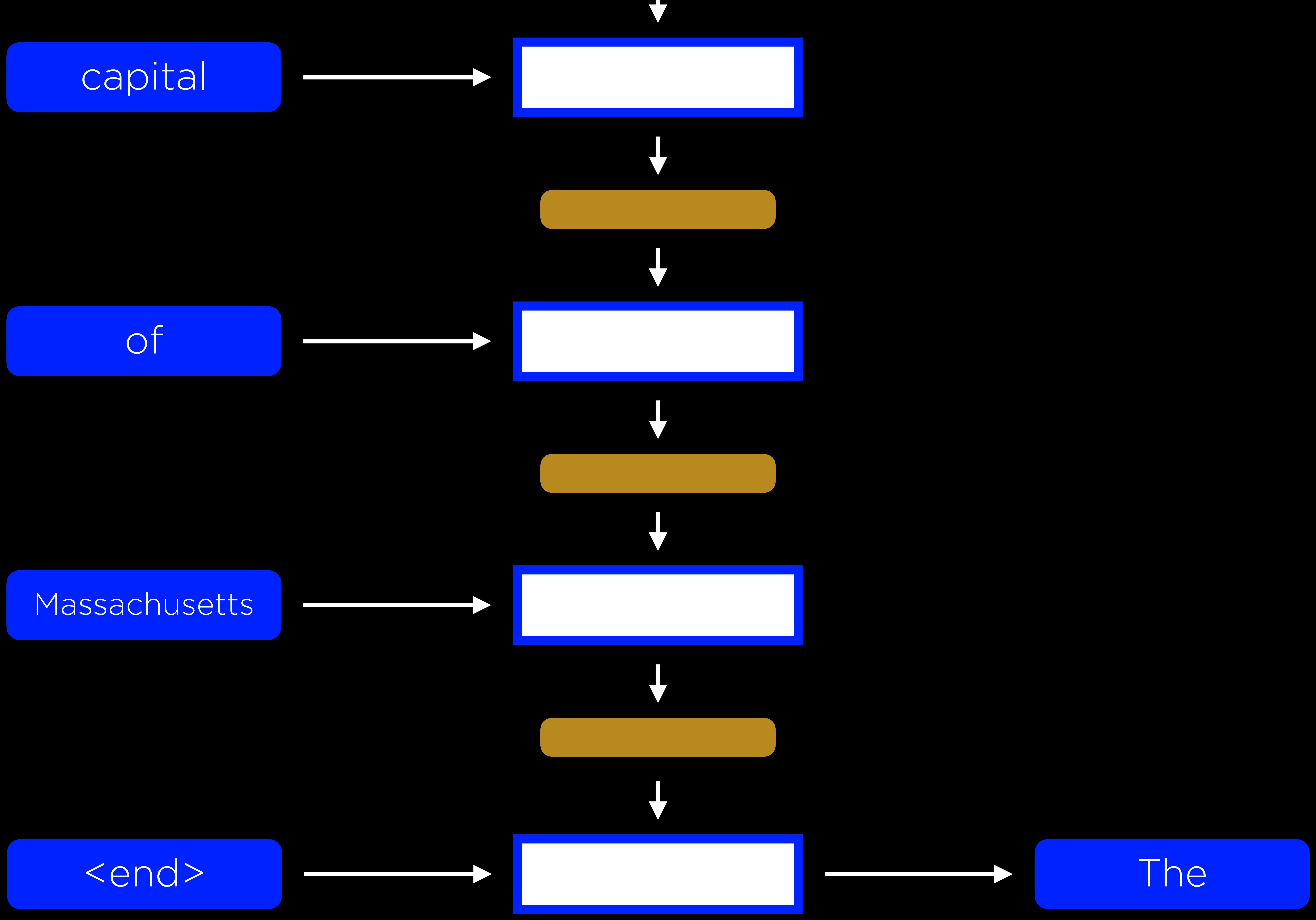


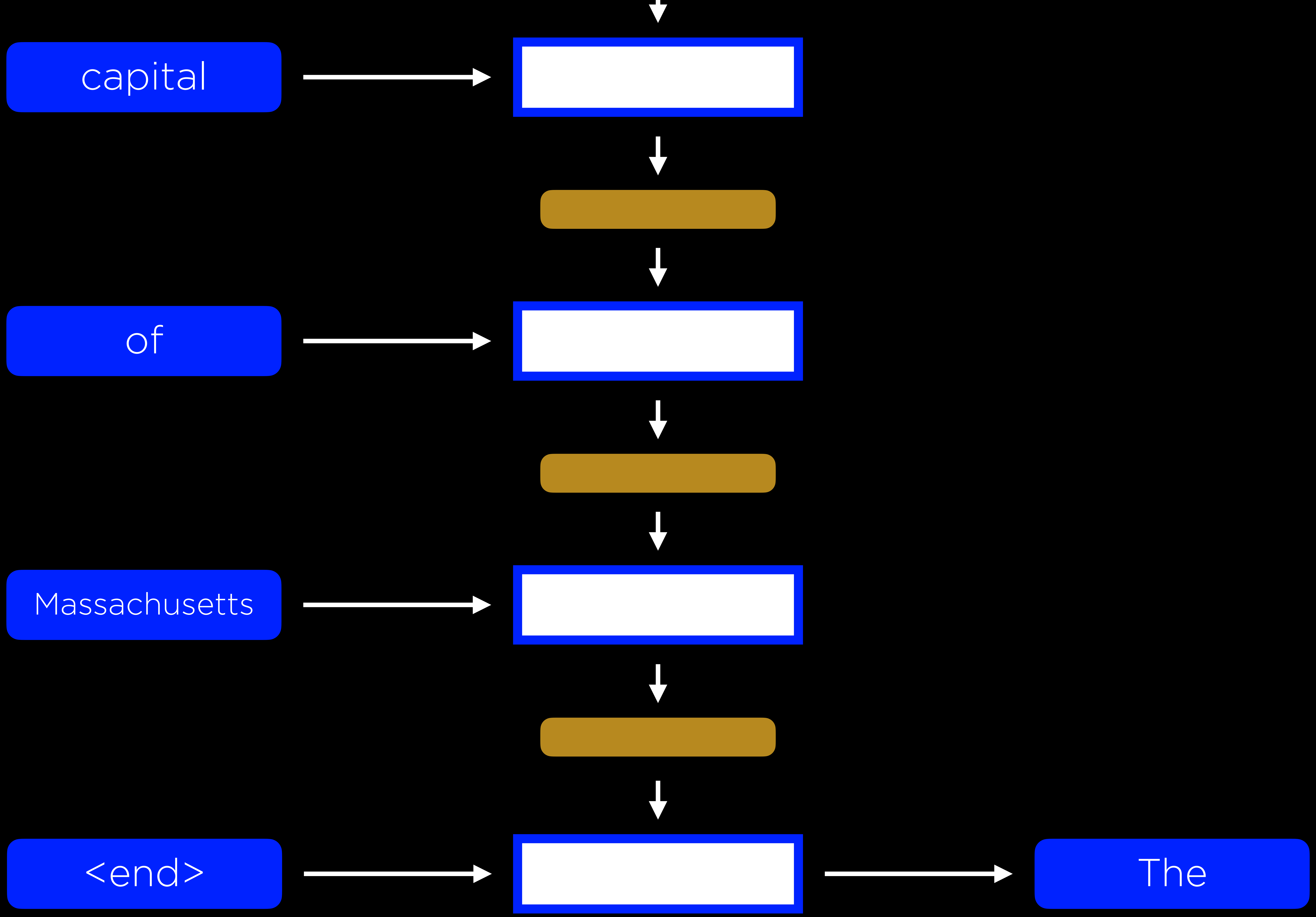
What is the capital of Massachusetts?

The capital is Boston.


what
hidden state
is
the
capital


what
is
the
capital


capital
of
Massachusetts
<end>
The



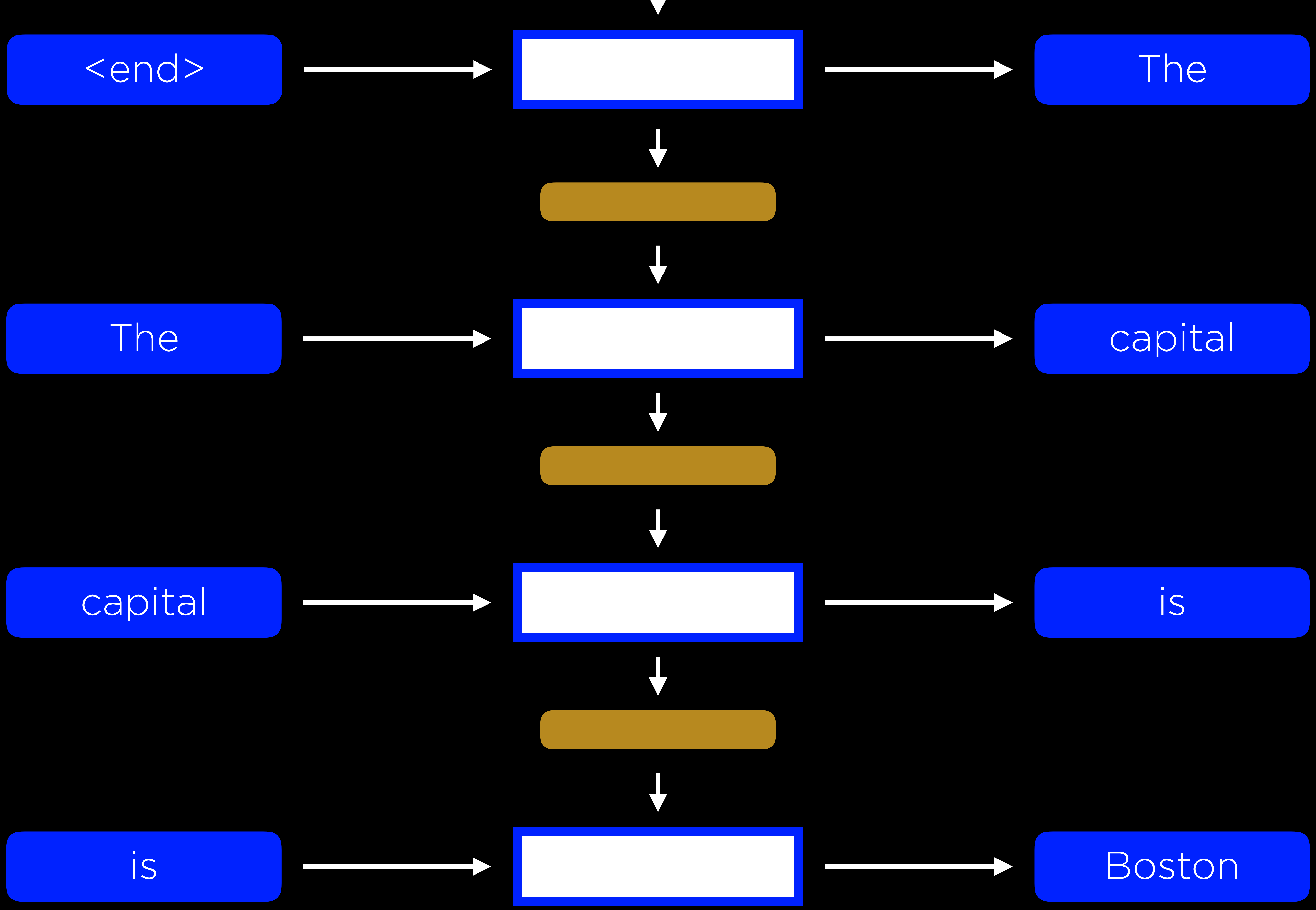
capital

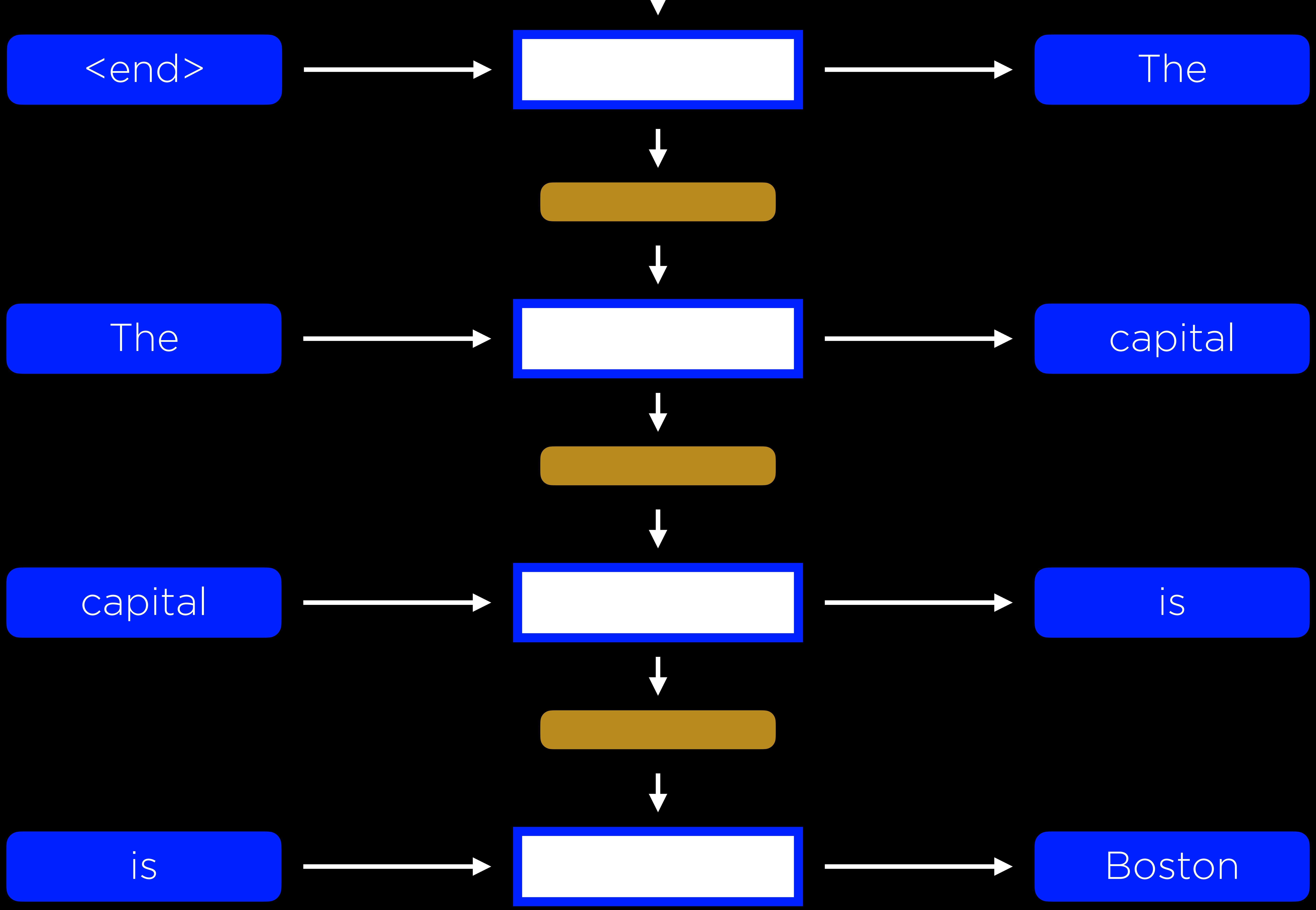
of

Massachusetts

<end>

The


<end>
The
The
capital
capital
is
is
Boston



<end>

The

The

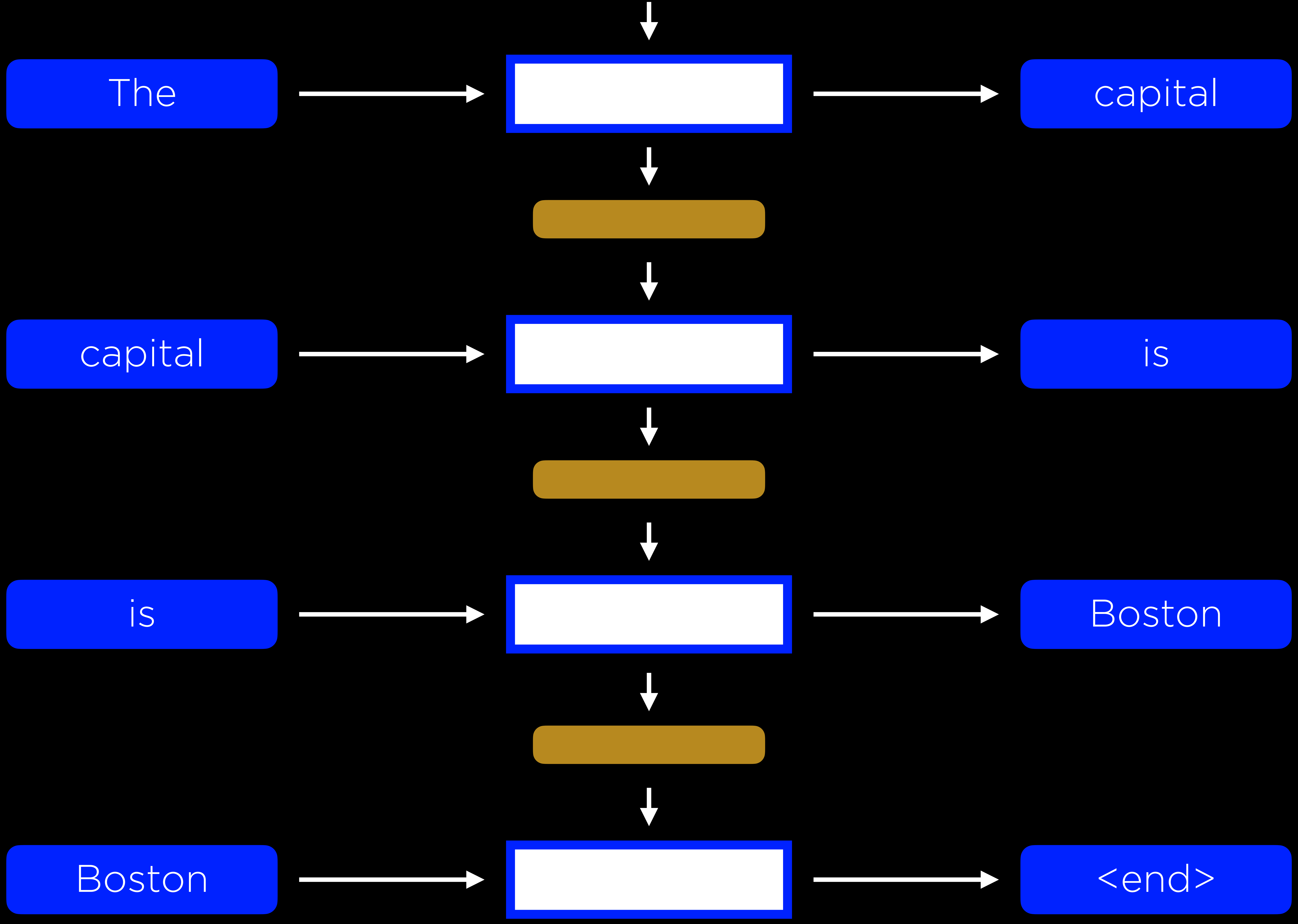
capital

capital

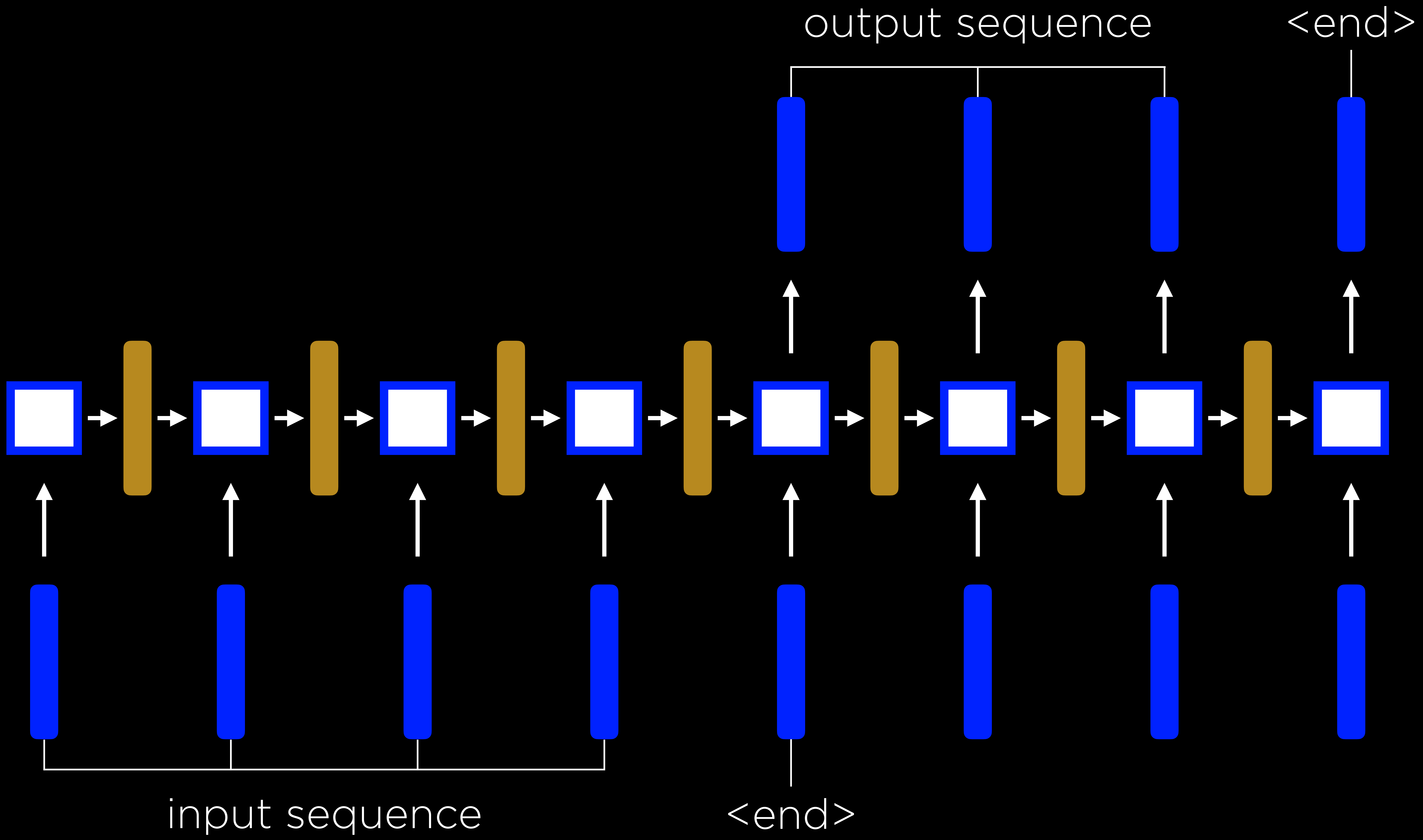
is

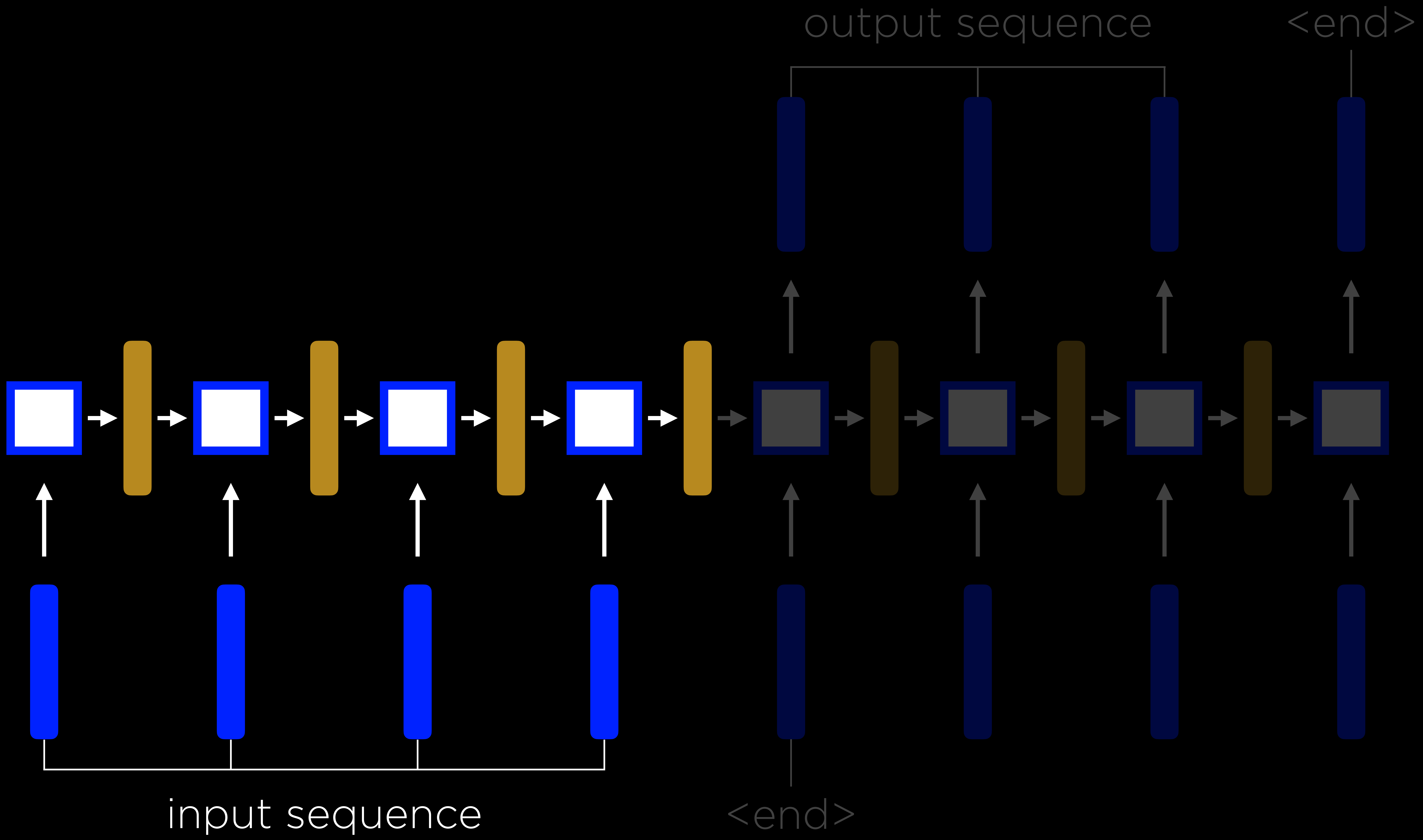
is

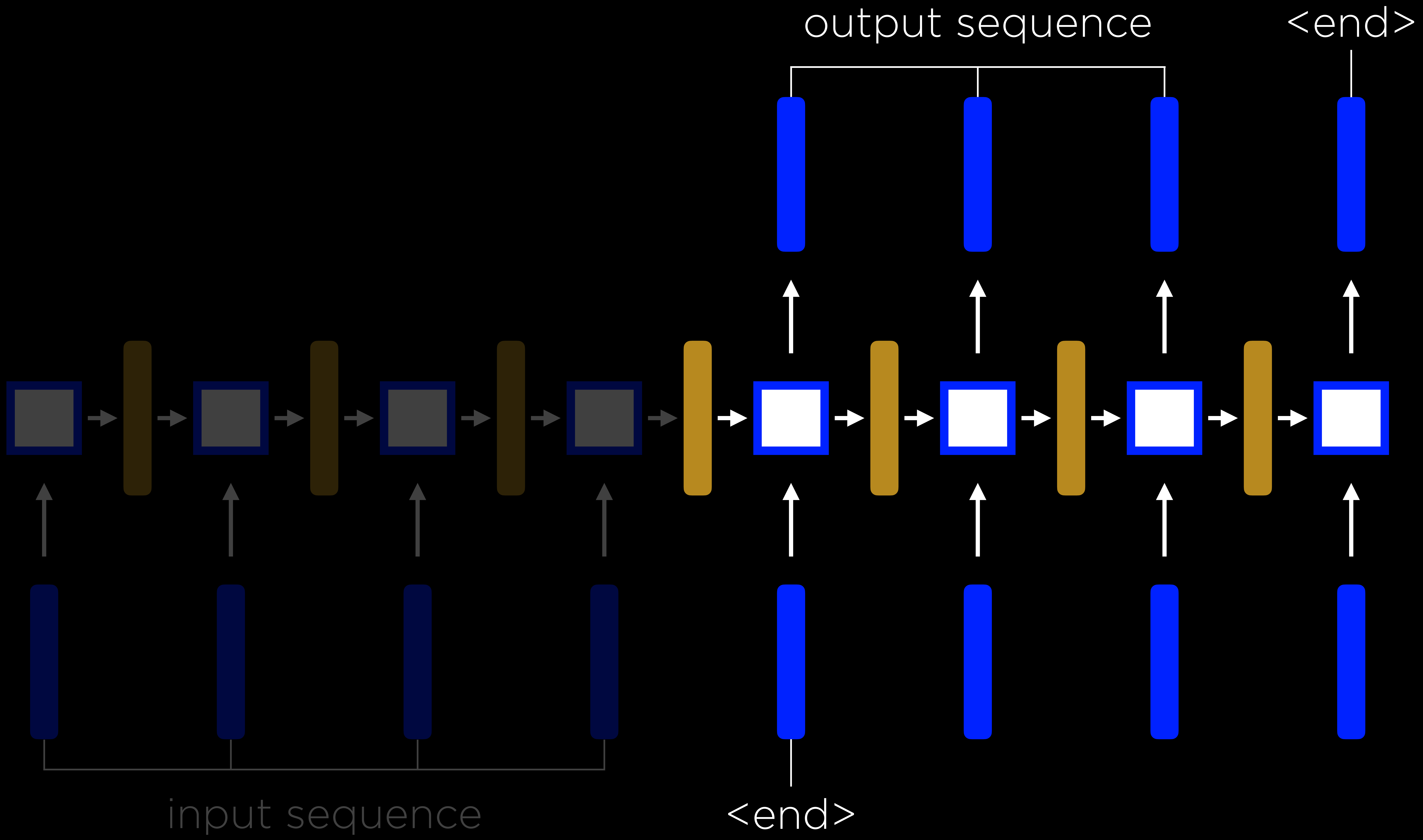
Boston

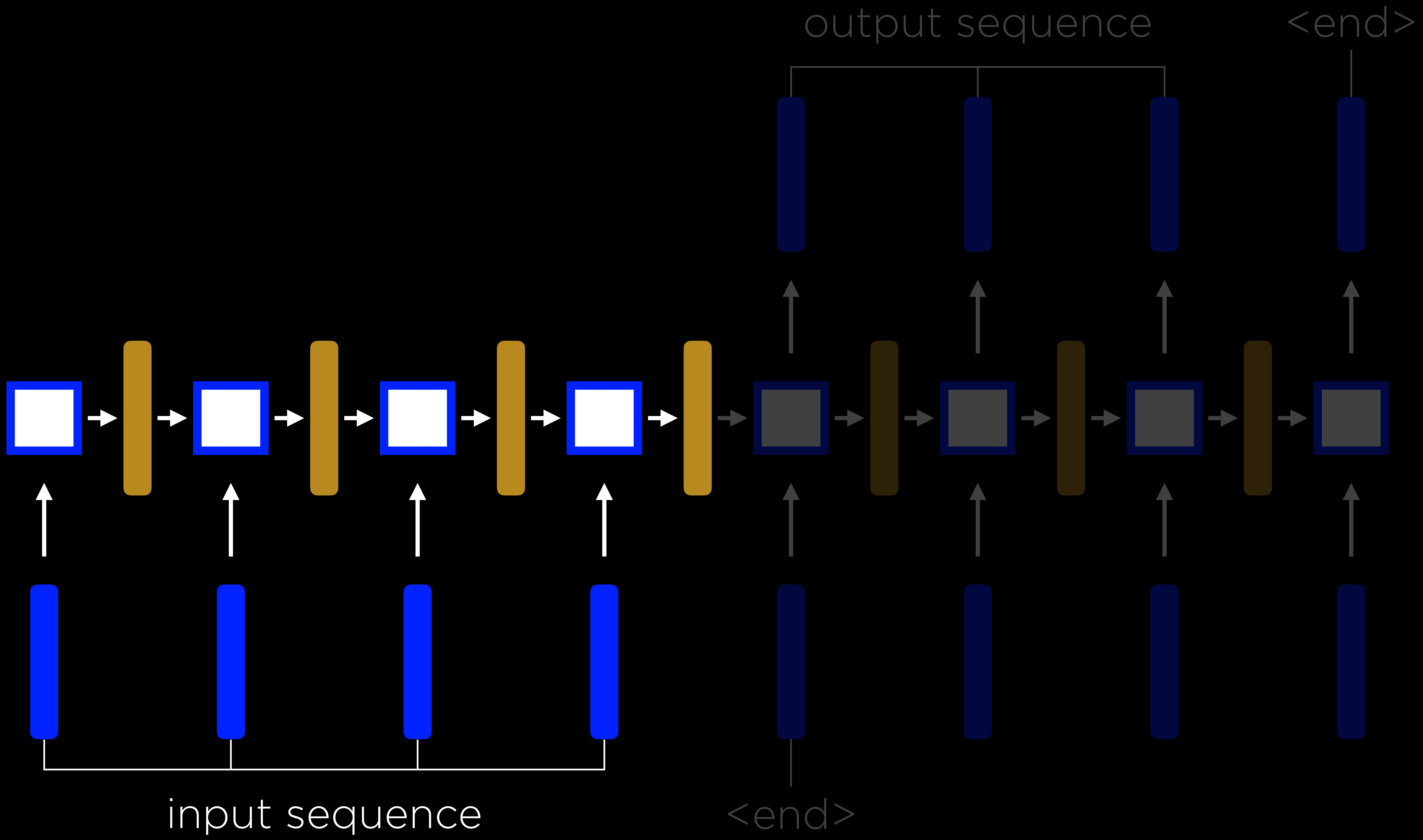


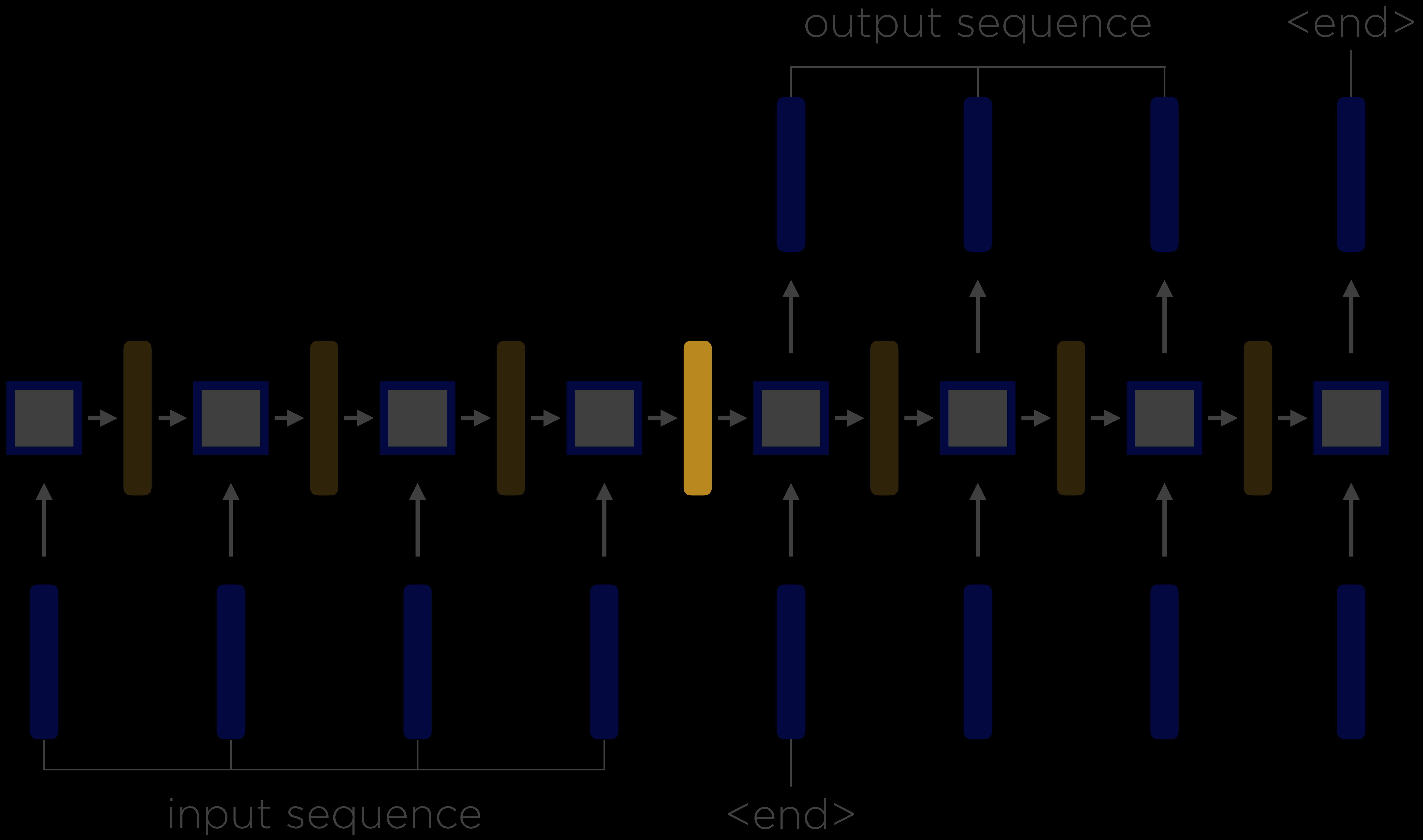
The
capital
capital
is
is
Boston
Boston
<end>

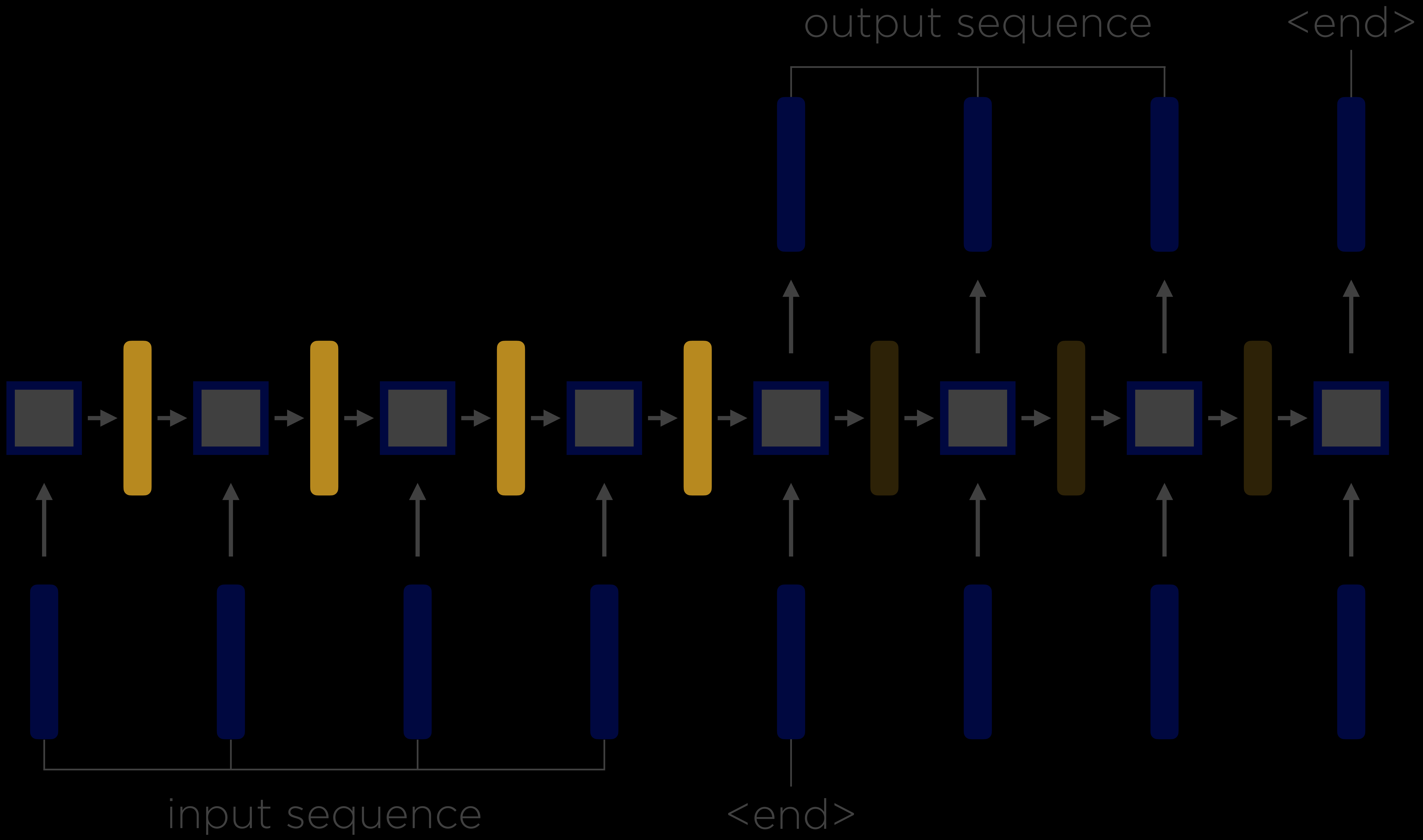

output sequence
<end>
input sequence
<end>


output sequence
<end>
input sequence
<end>

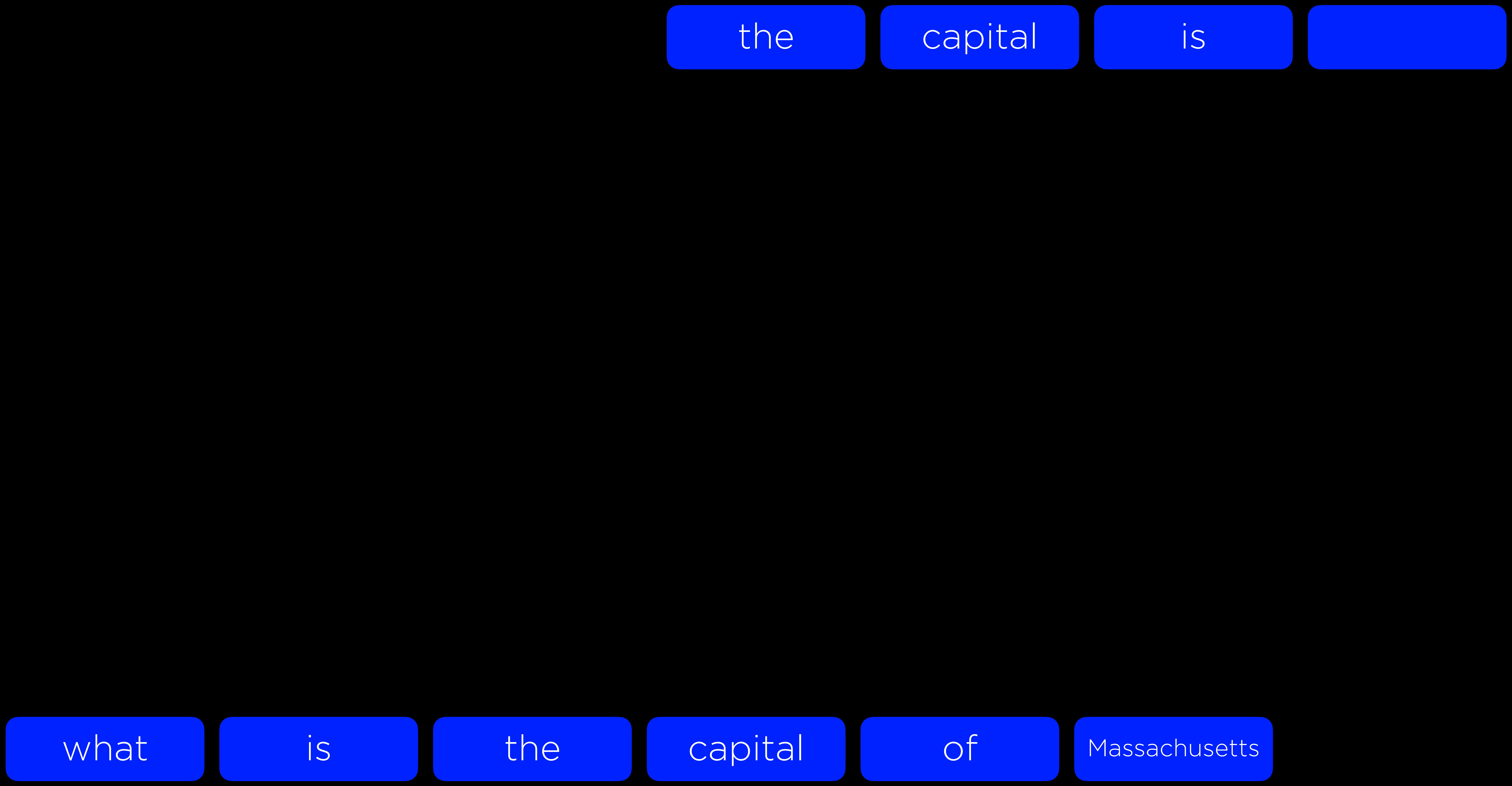

output sequence
<end>
<end>
input sequence


output sequence
<end>
input sequence
<end>


output sequence
<end>
input sequence
<end>

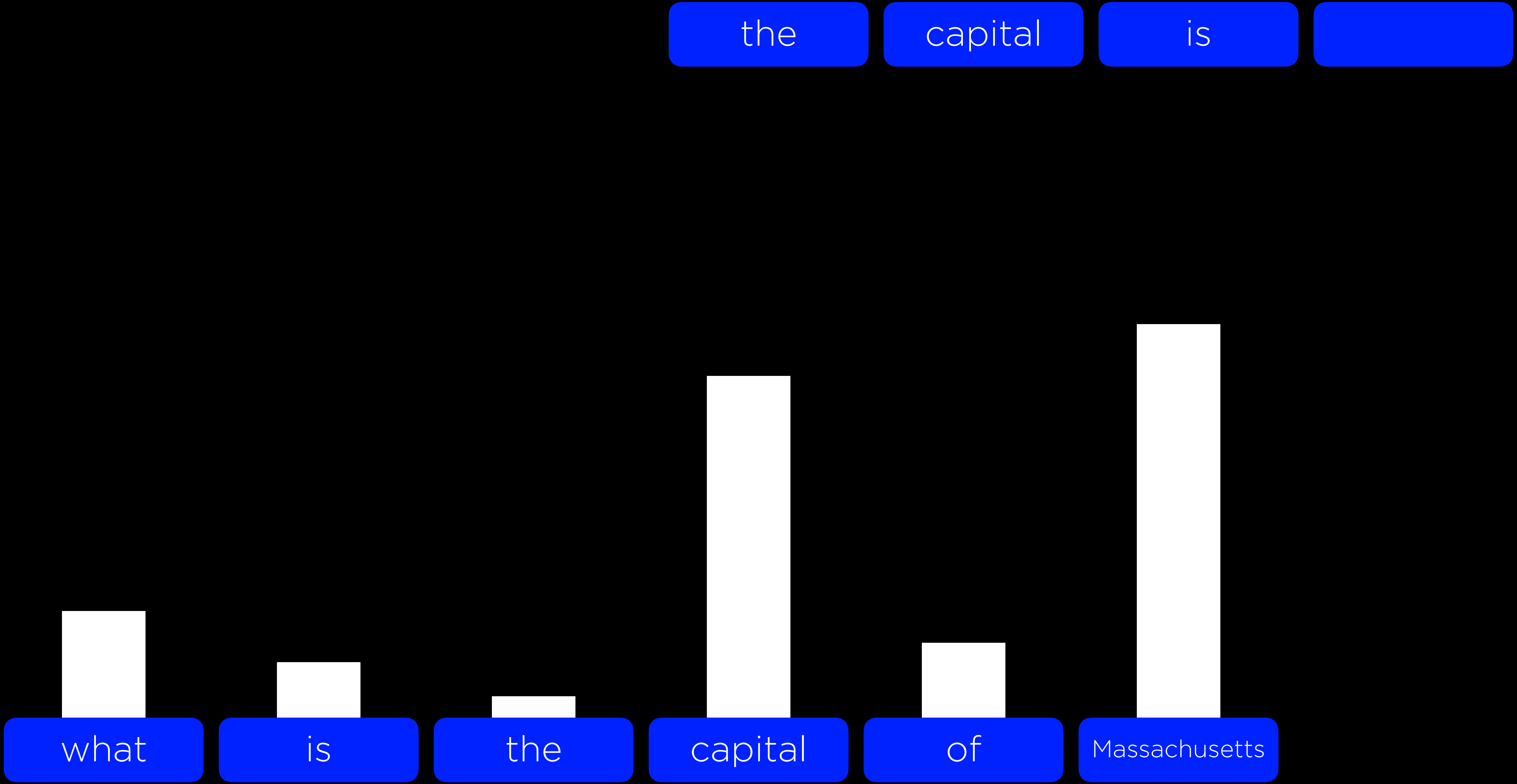

output sequence
<end>
input sequence
<end>

Attention



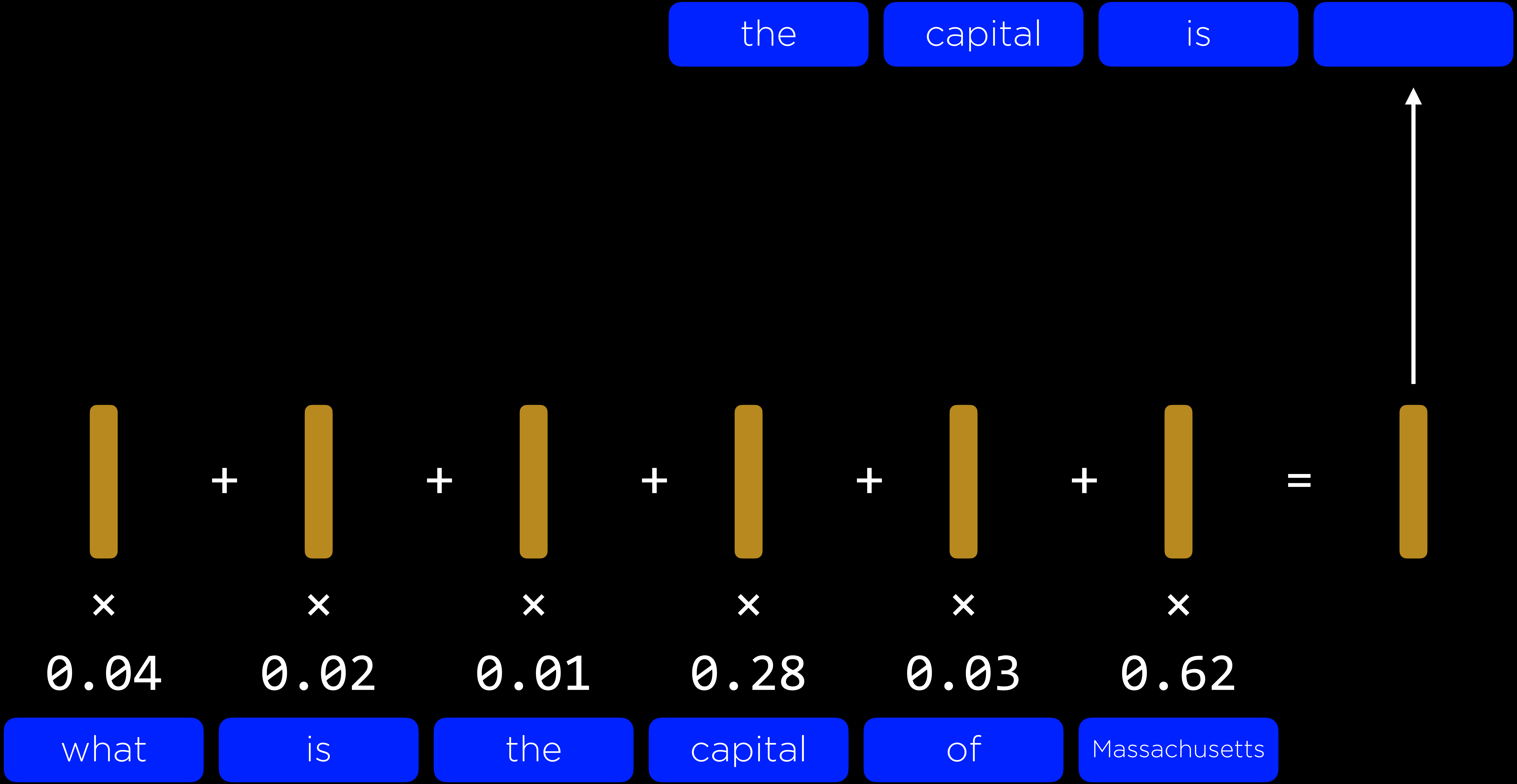
the capital is

what is the capital of Massachusetts



the capital is

what is the capital of Massachusetts



the capital is

× 0.04 what

+ × 0.02 is

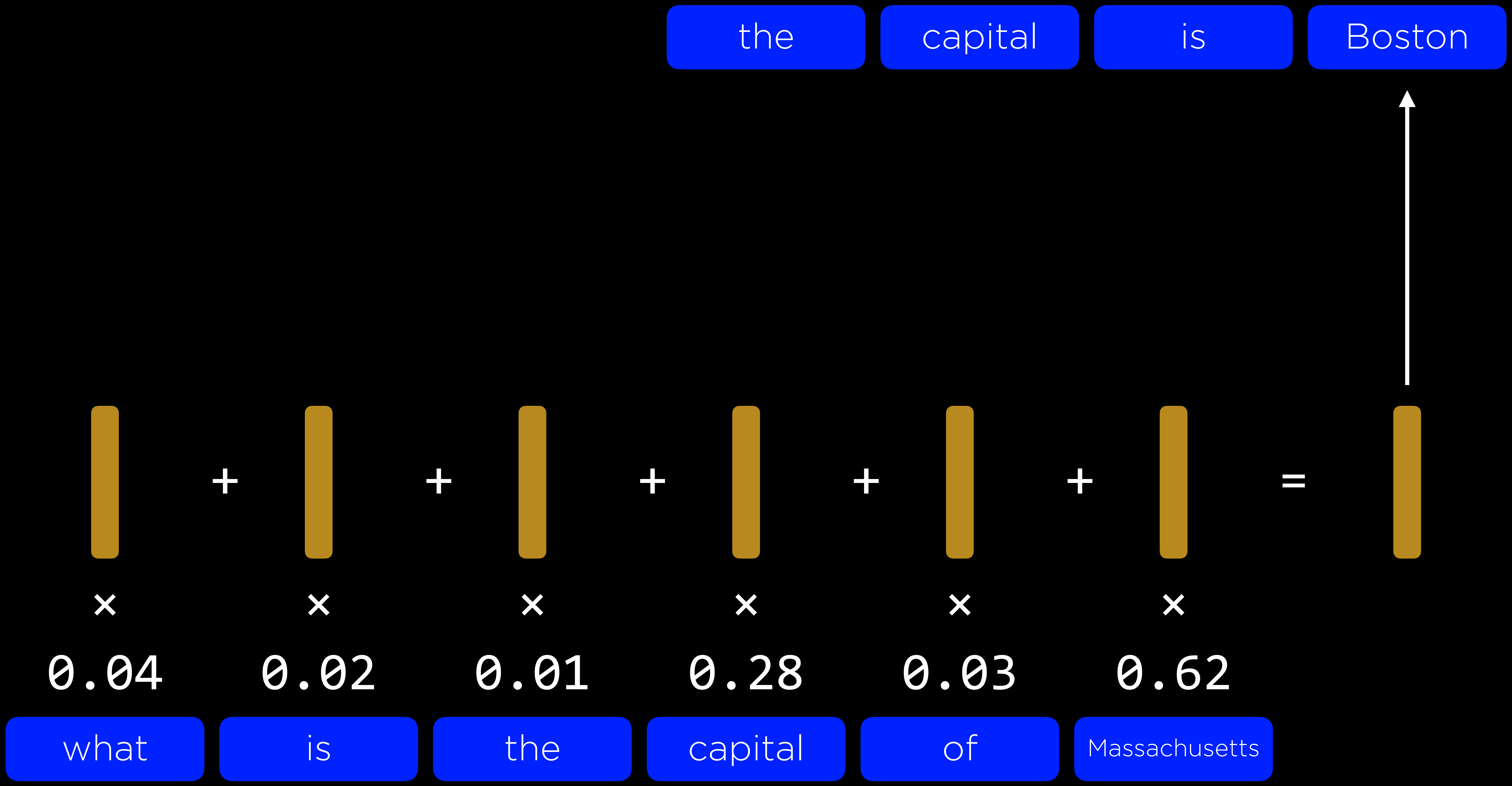
+ × 0.01 the

+ × 0.28 capital

+ × 0.03 of

+ × 0.62 Massachusetts

=



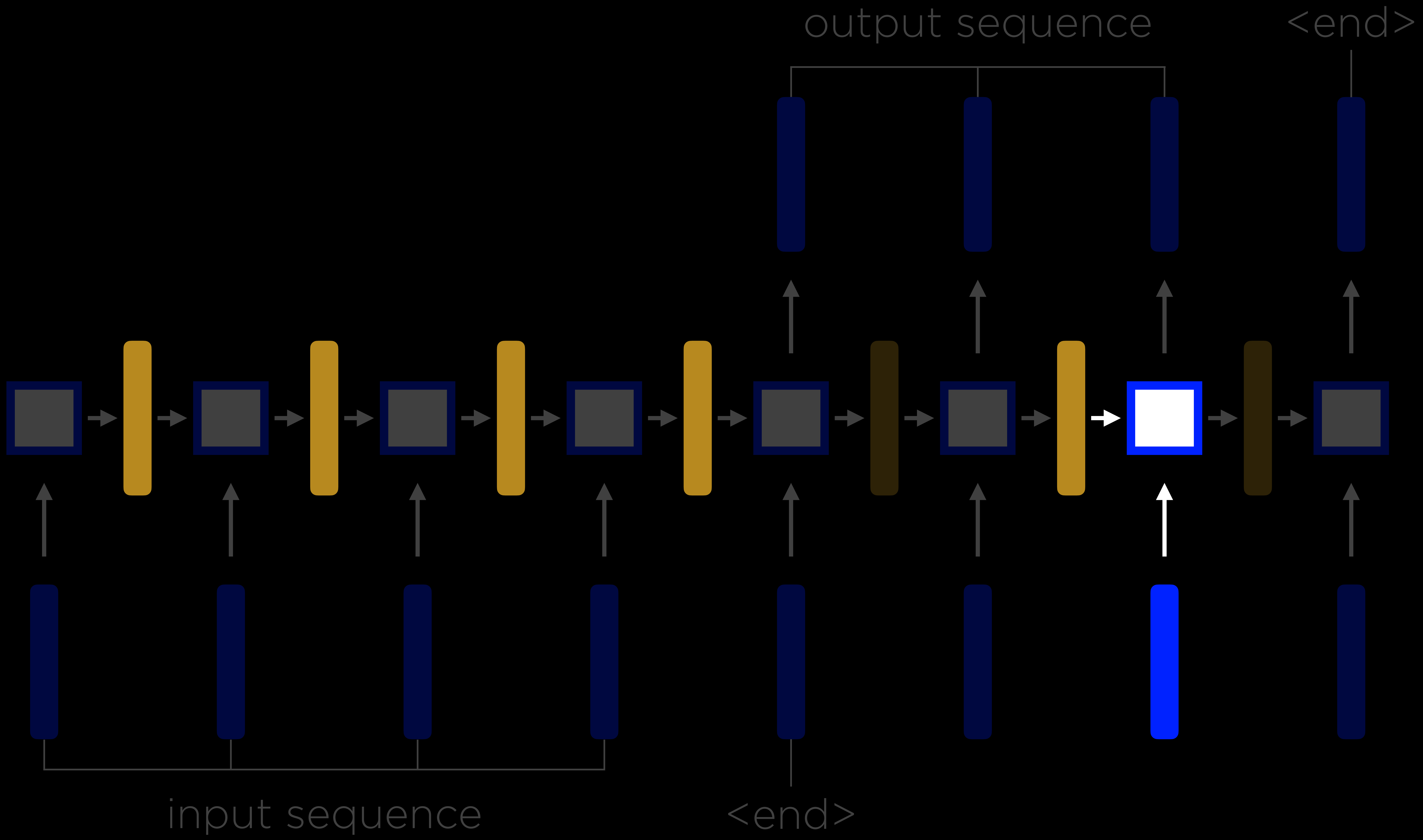
the capital is Boston

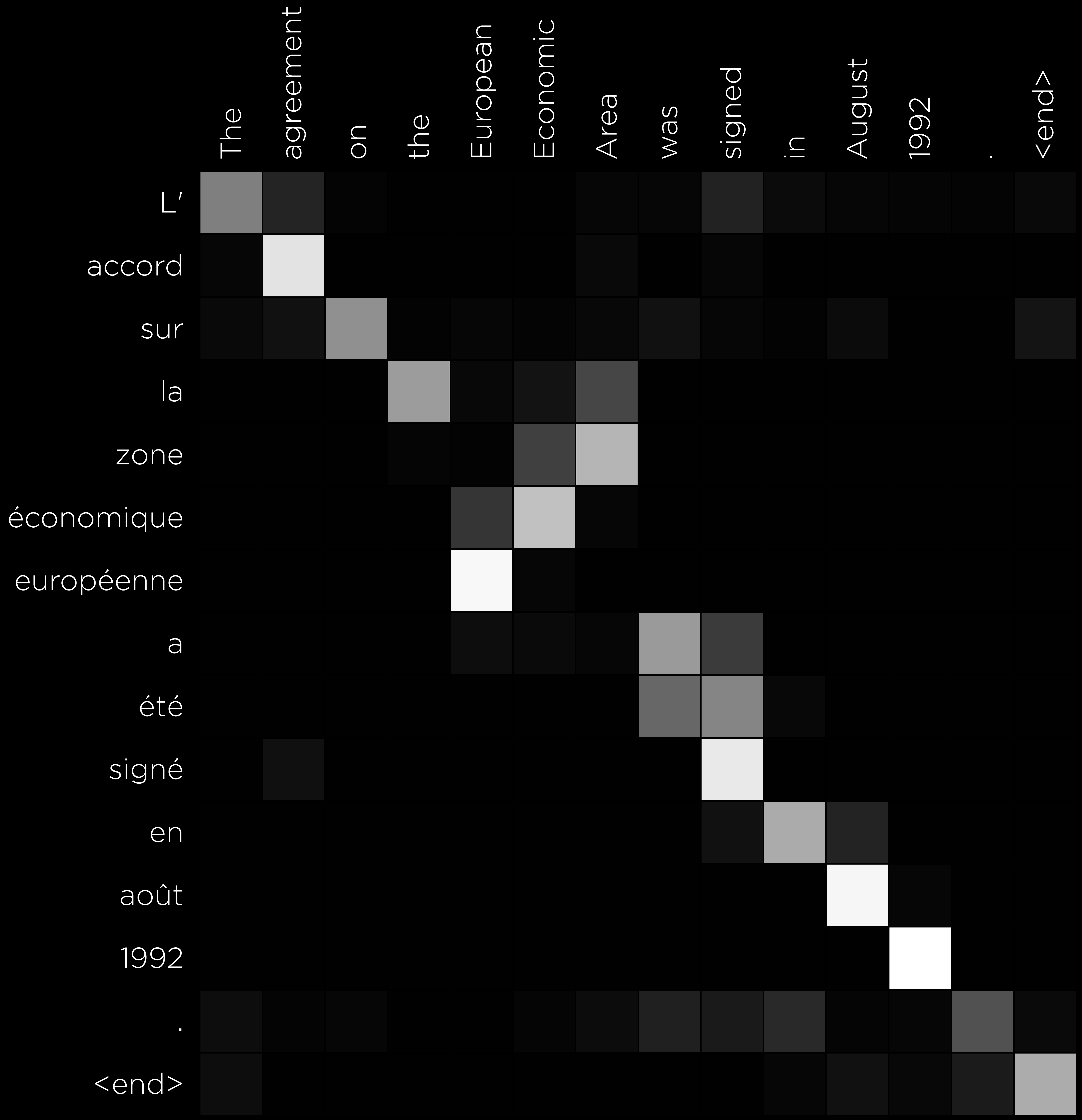
+ + + + + =

× × × × × ×

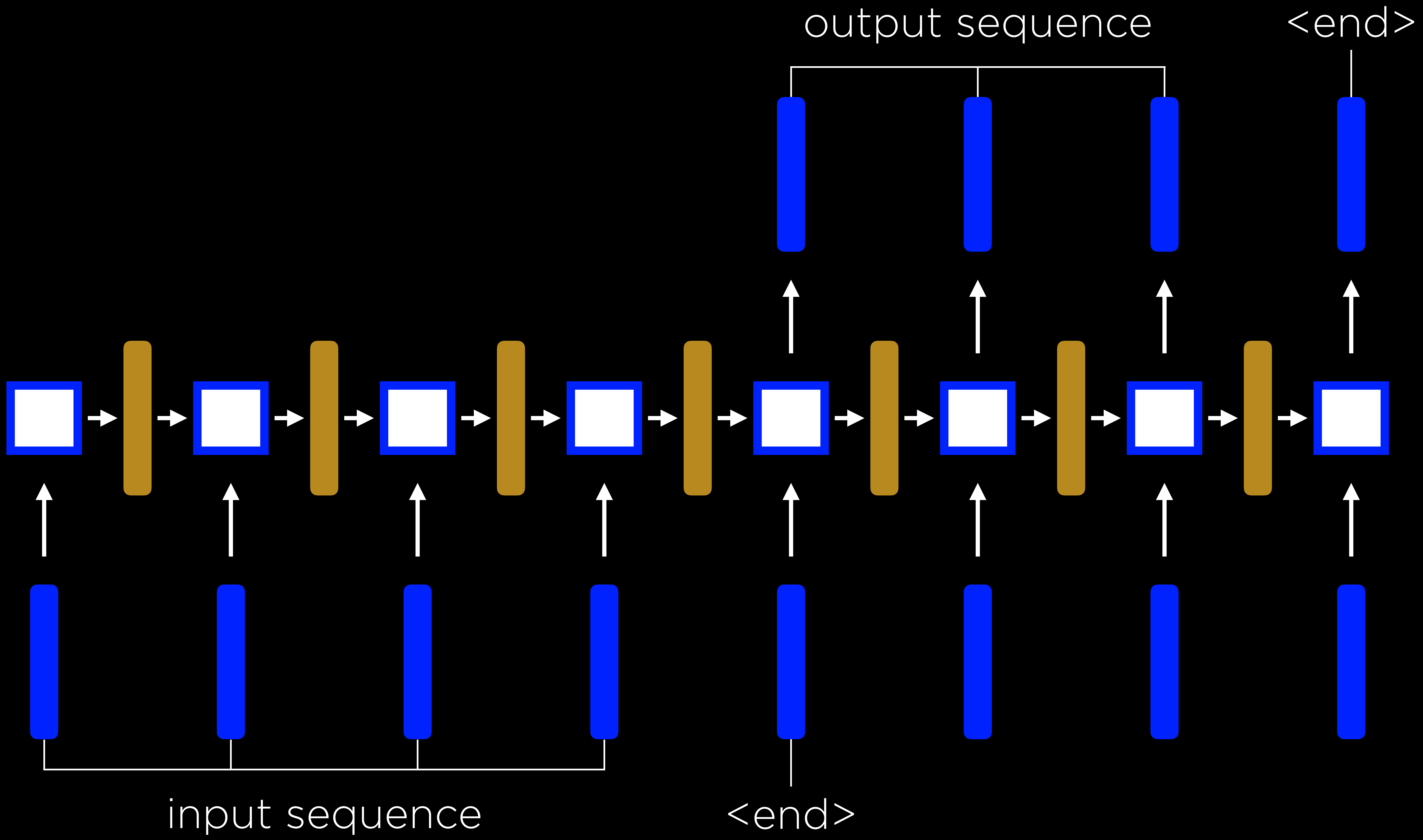
0.04 0.02 0.01 0.28 0.03 0.62

what is the capital of Massachusetts

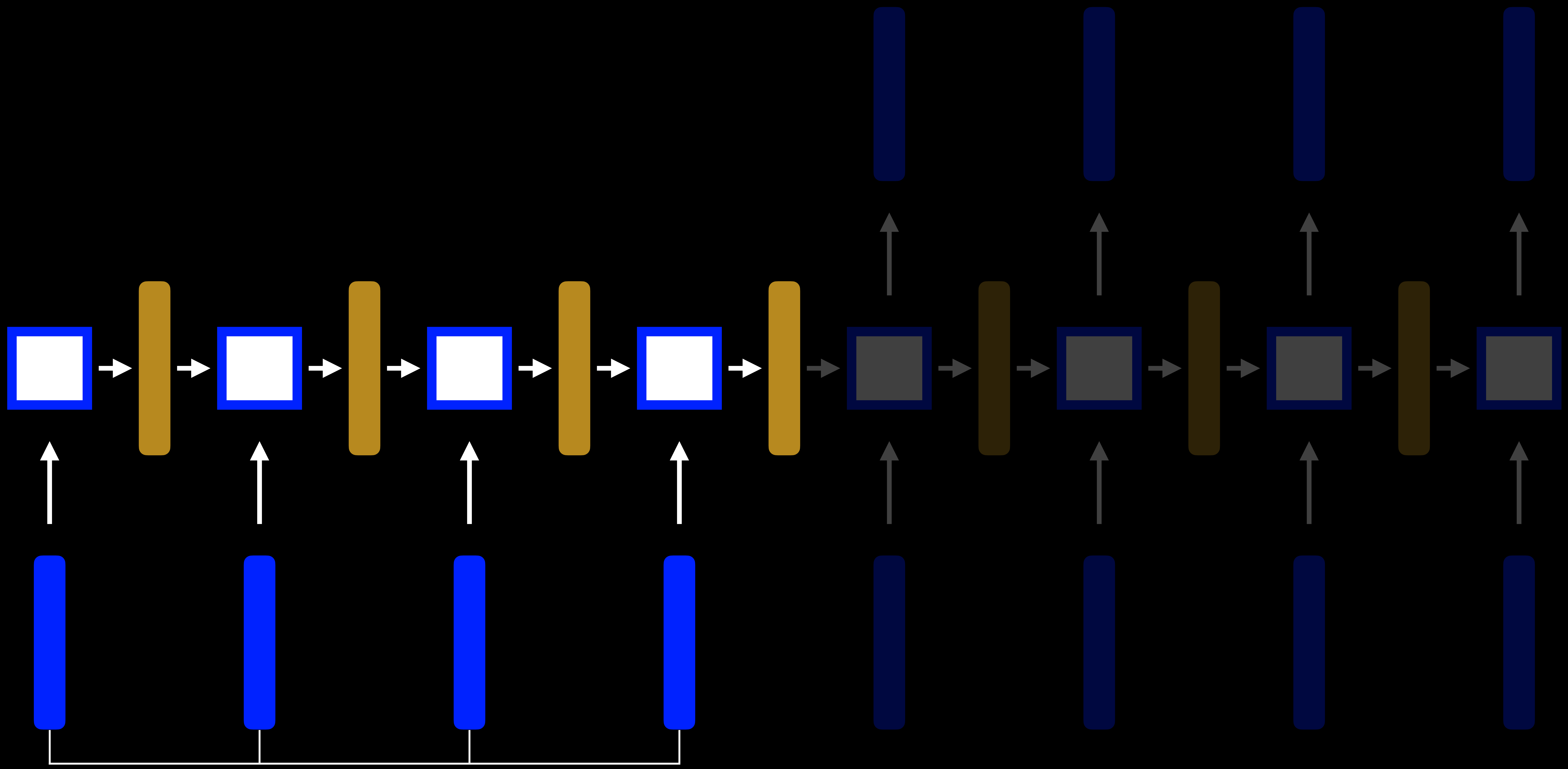

output sequence
<end>
input sequence
<end>



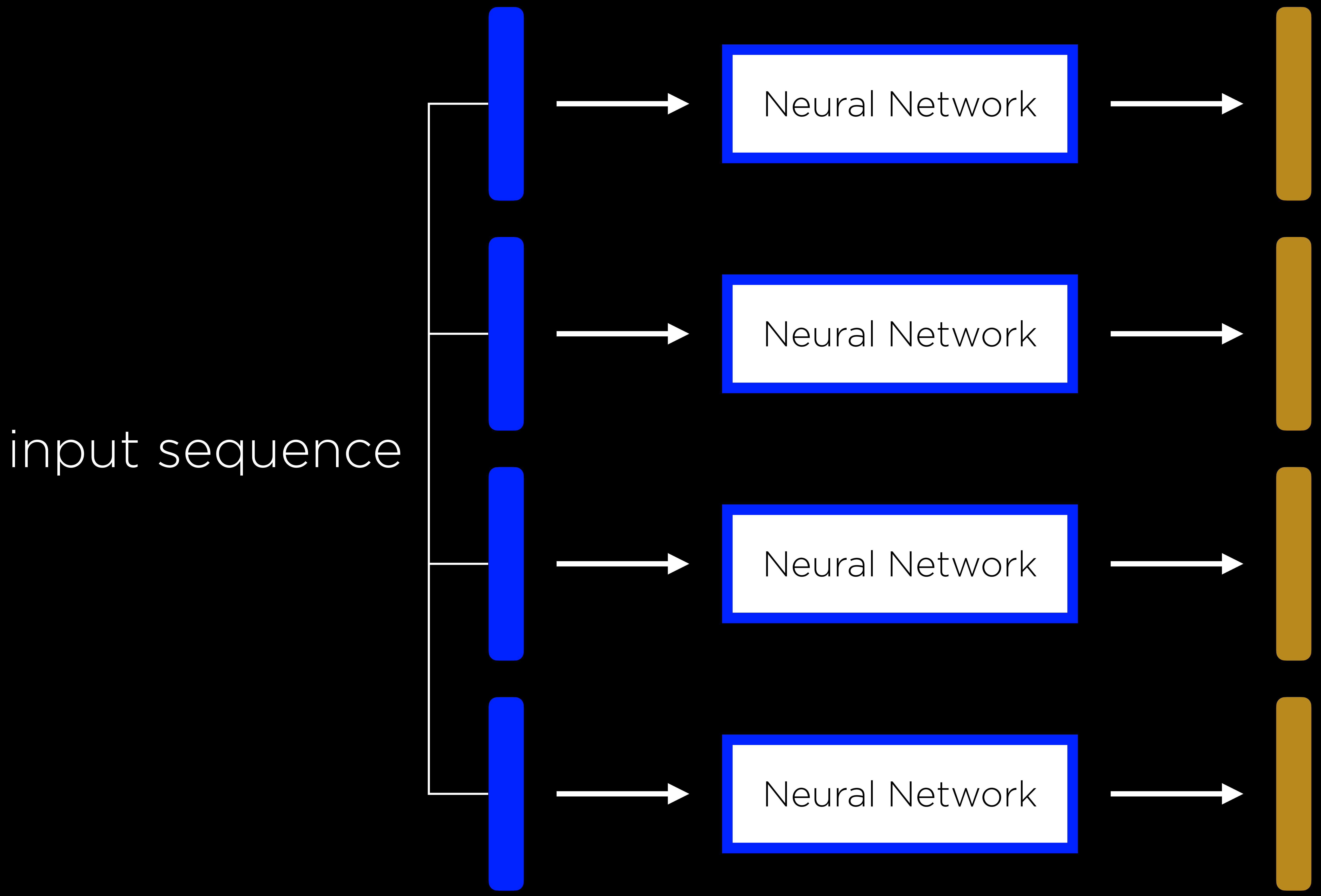
Adapted from Bahdanau et al. 2015.
Neural machine translation by jointly
learning to align and translate

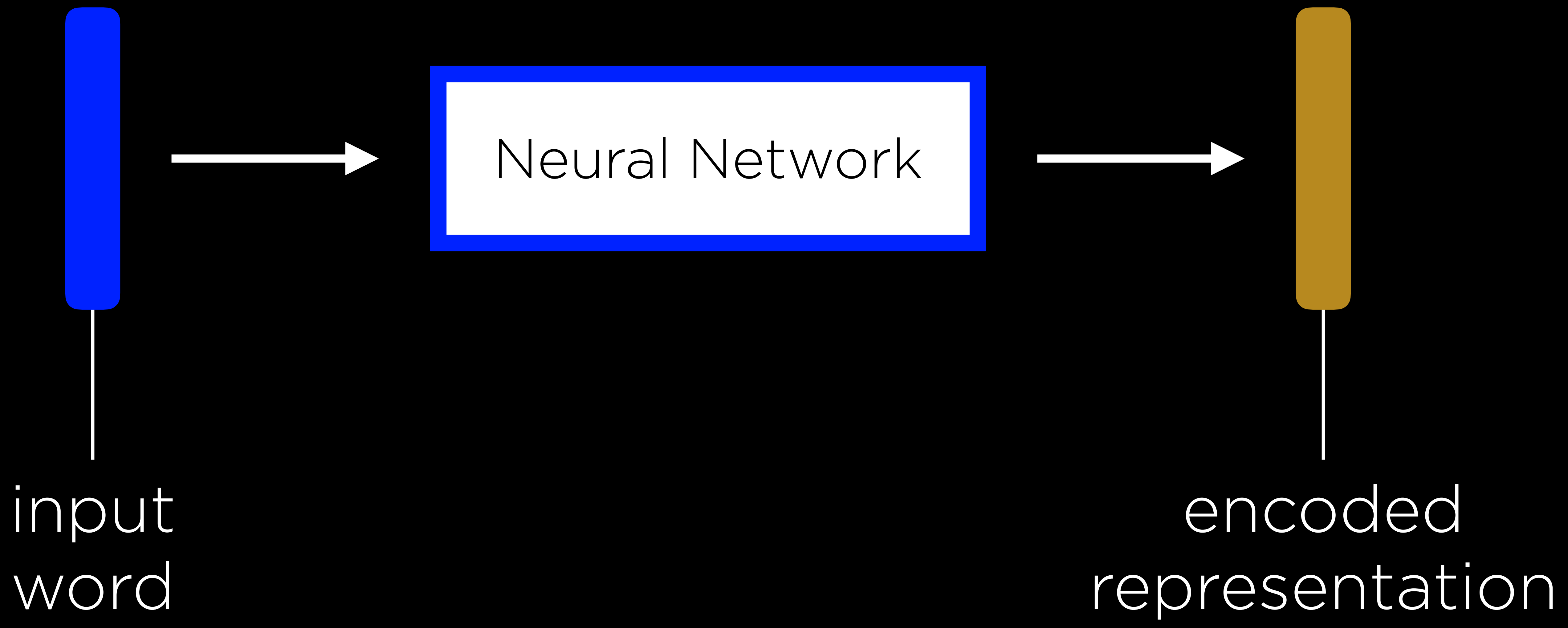

output sequence
<end>
input sequence
<end>

Transformers



input sequence

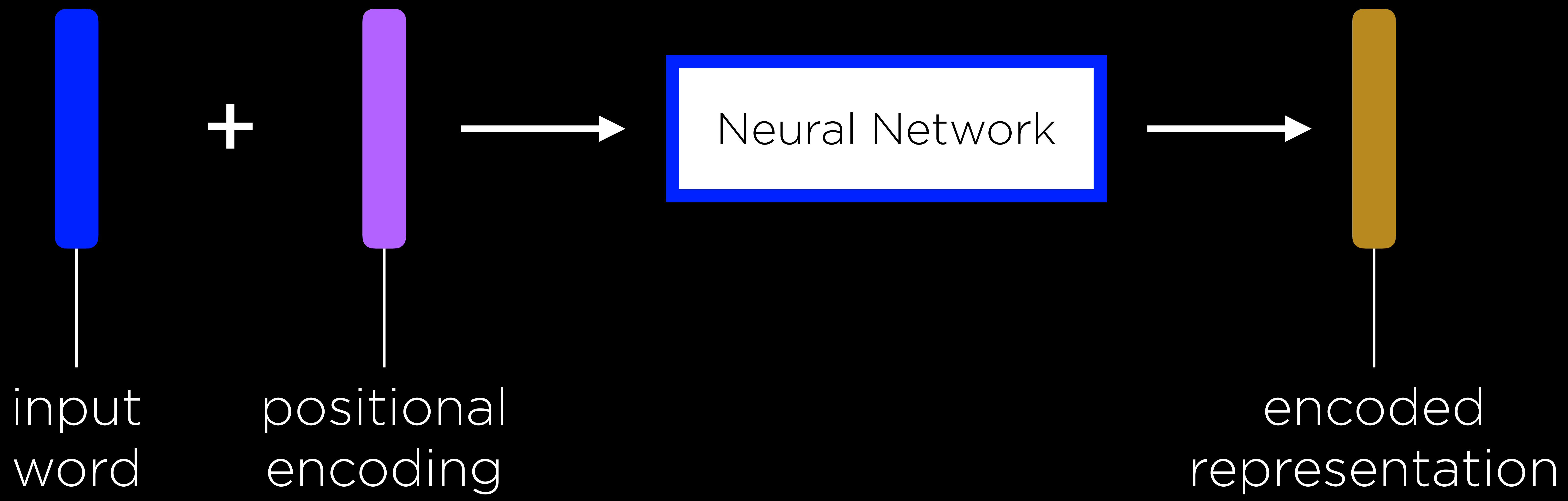

input sequence
Neural Network
Neural Network
Neural Network
Neural Network



Neural Network

input
word

encoded
representation



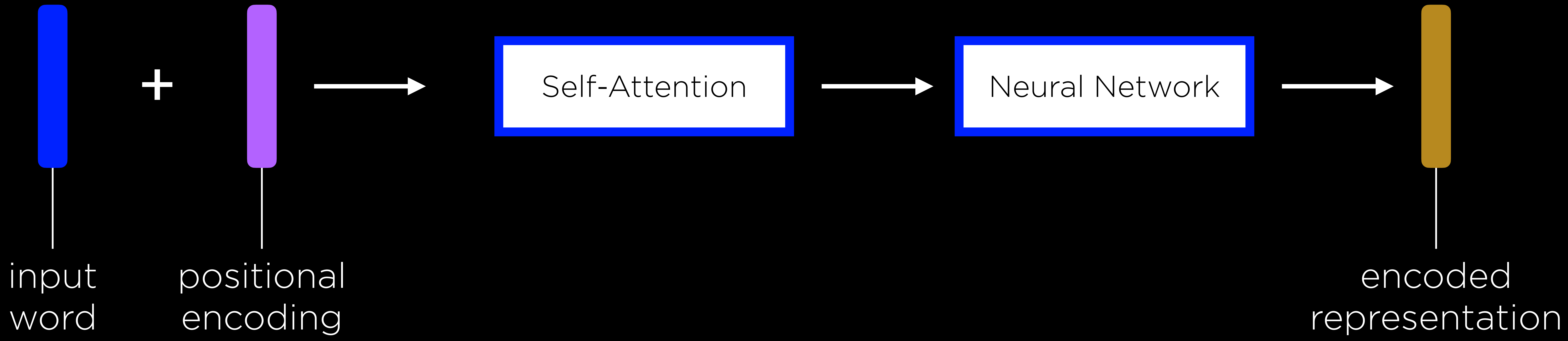
input word

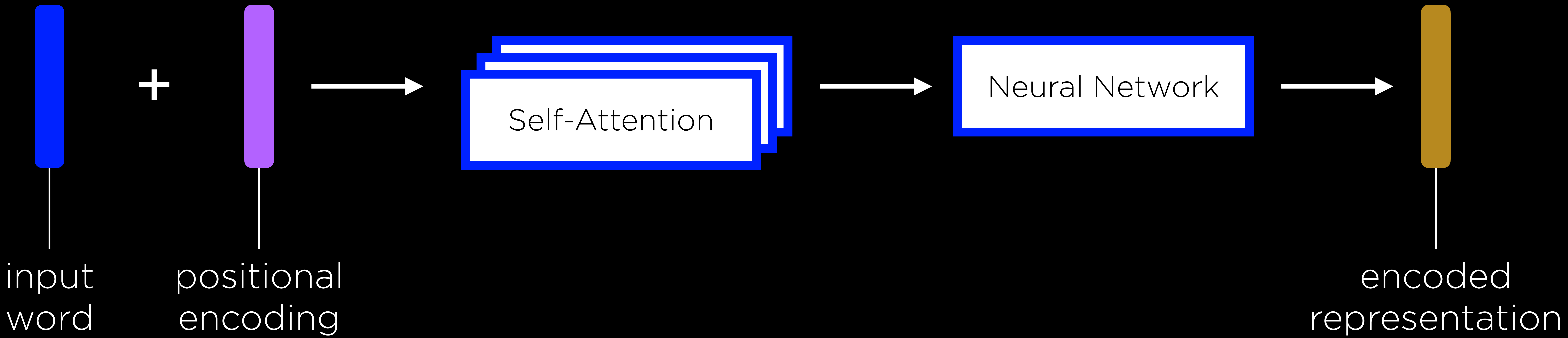
+

positional encoding

Neural Network

encoded representation


input word
+
positional encoding
Self-Attention
Neural Network
encoded representation



input word

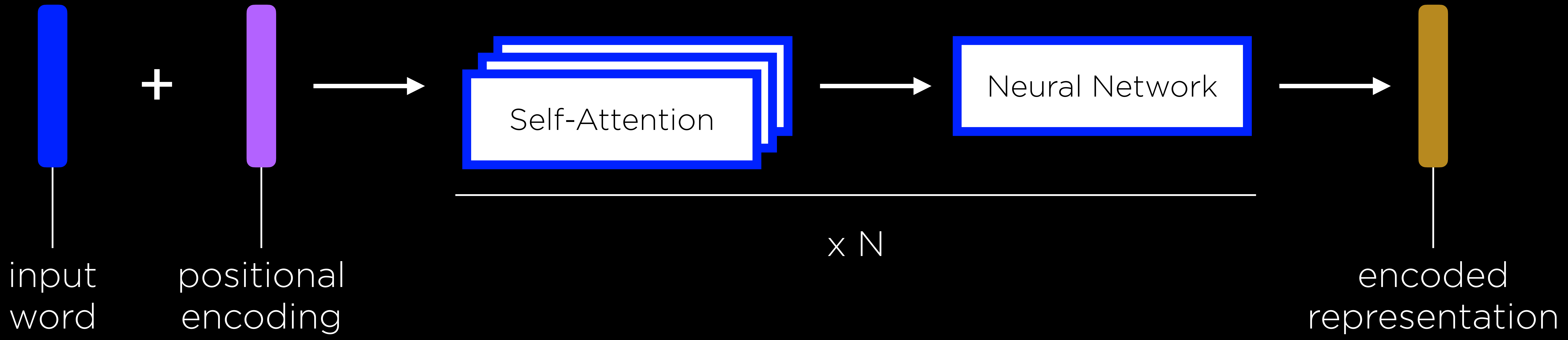
+

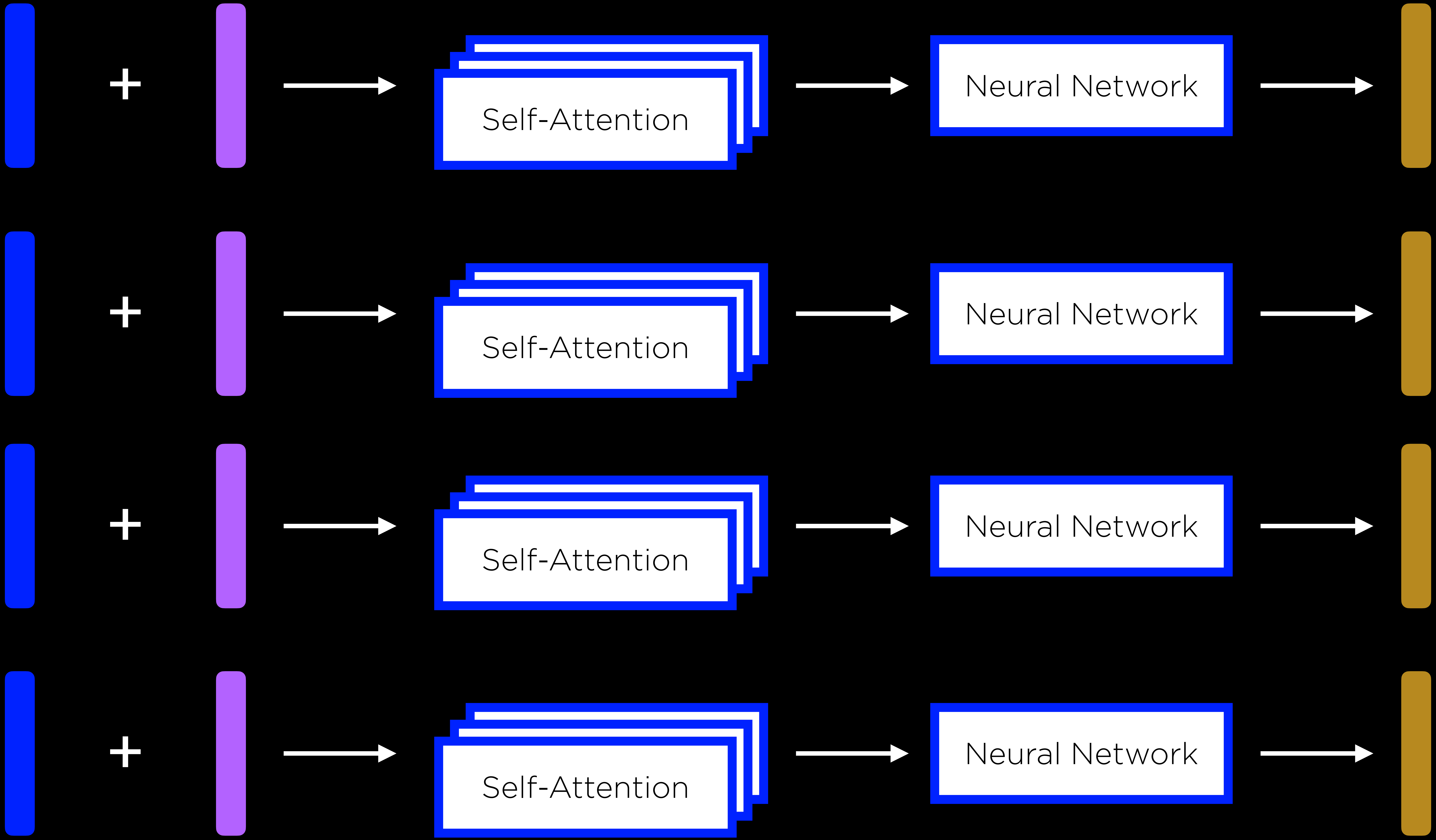
positional encoding

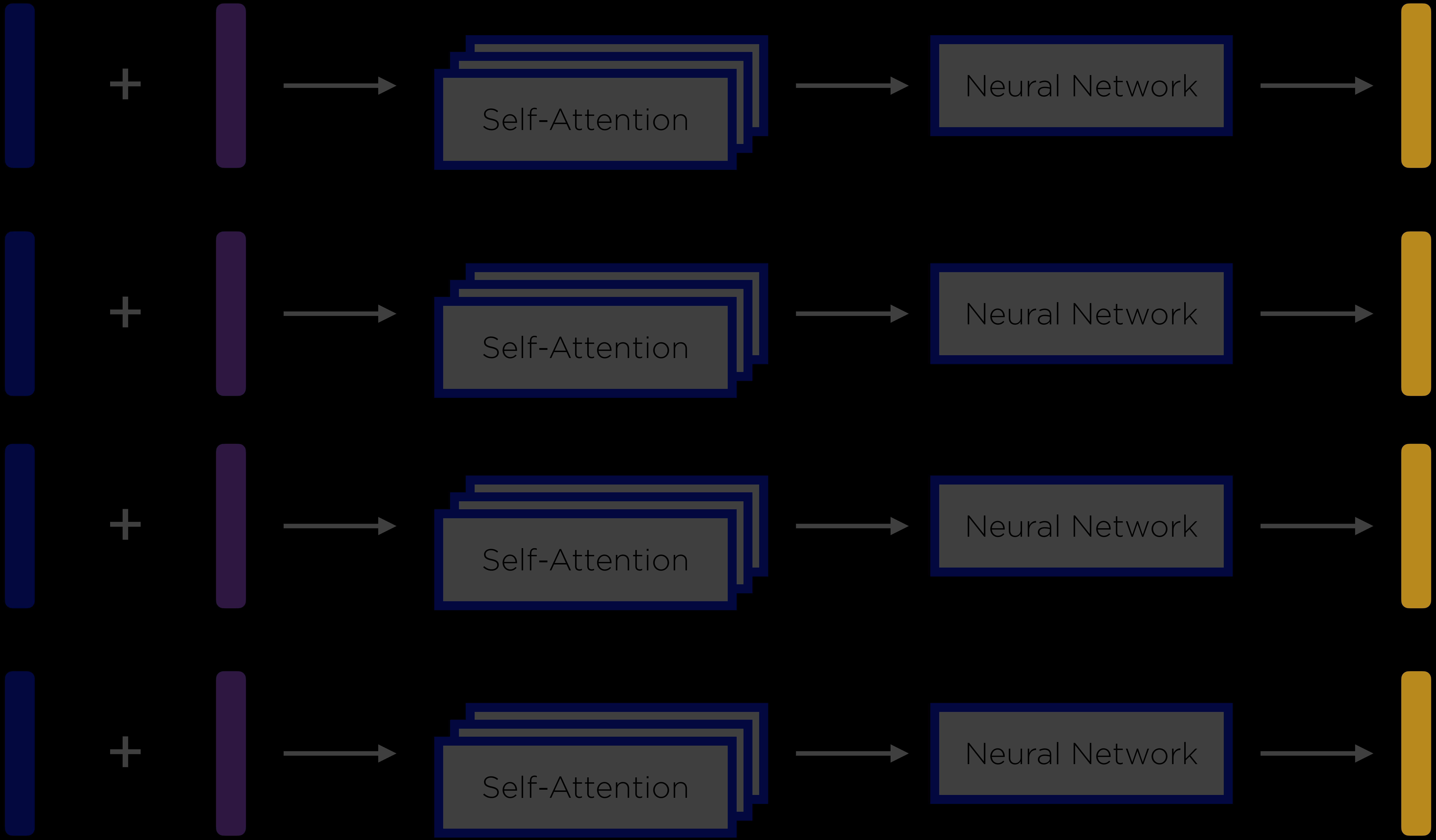
Self-Attention

Neural Network

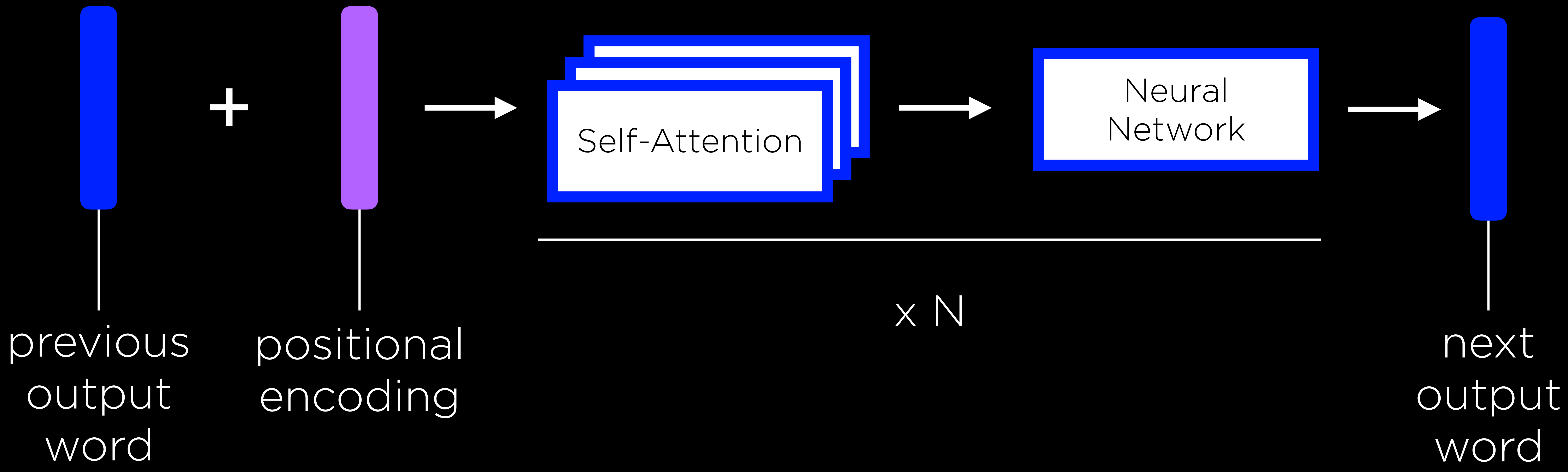
encoded representation

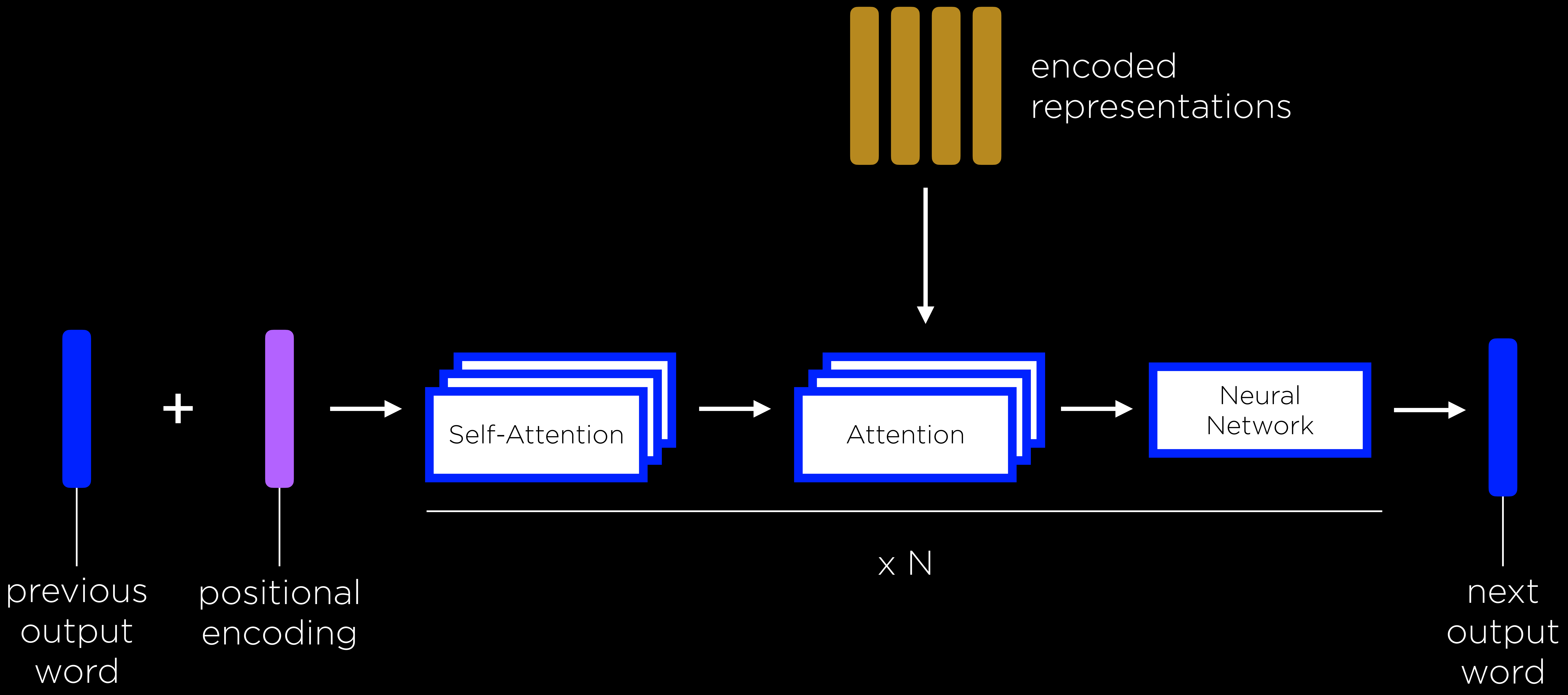

+
input word
positional encoding
Self-Attention
Neural Network
x N
encoded representation

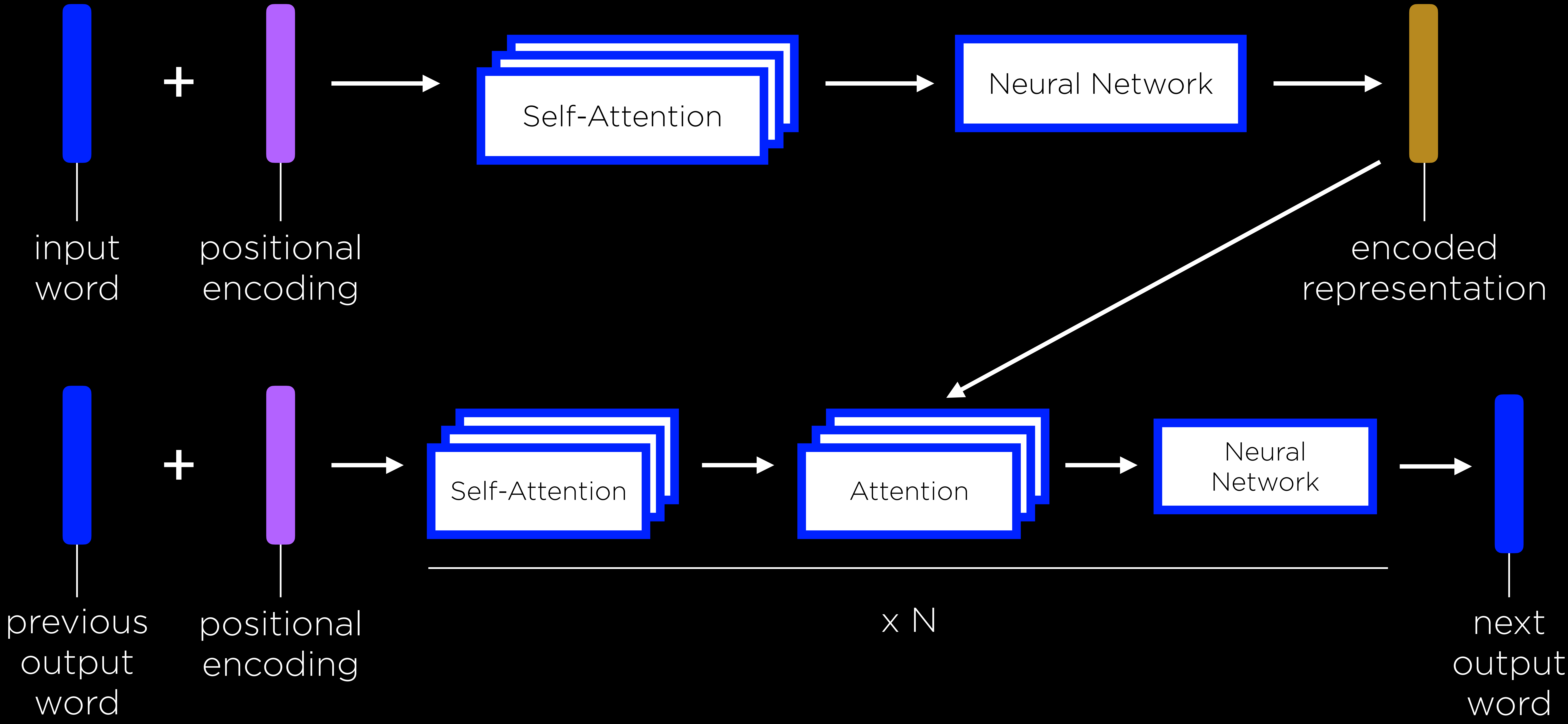

+
Self-Attention
Neural Network
+
Self-Attention
Neural Network
+
Self-Attention
Neural Network
+
Self-Attention
Neural Network



encoded representations


previous output word
+
positional encoding
Self-Attention
Neural Network
x N
next output word


encoded
representations
previous
output
word
+
positional
encoding
Self-Attention
Attention
Neural
Network
x N
next
output
word

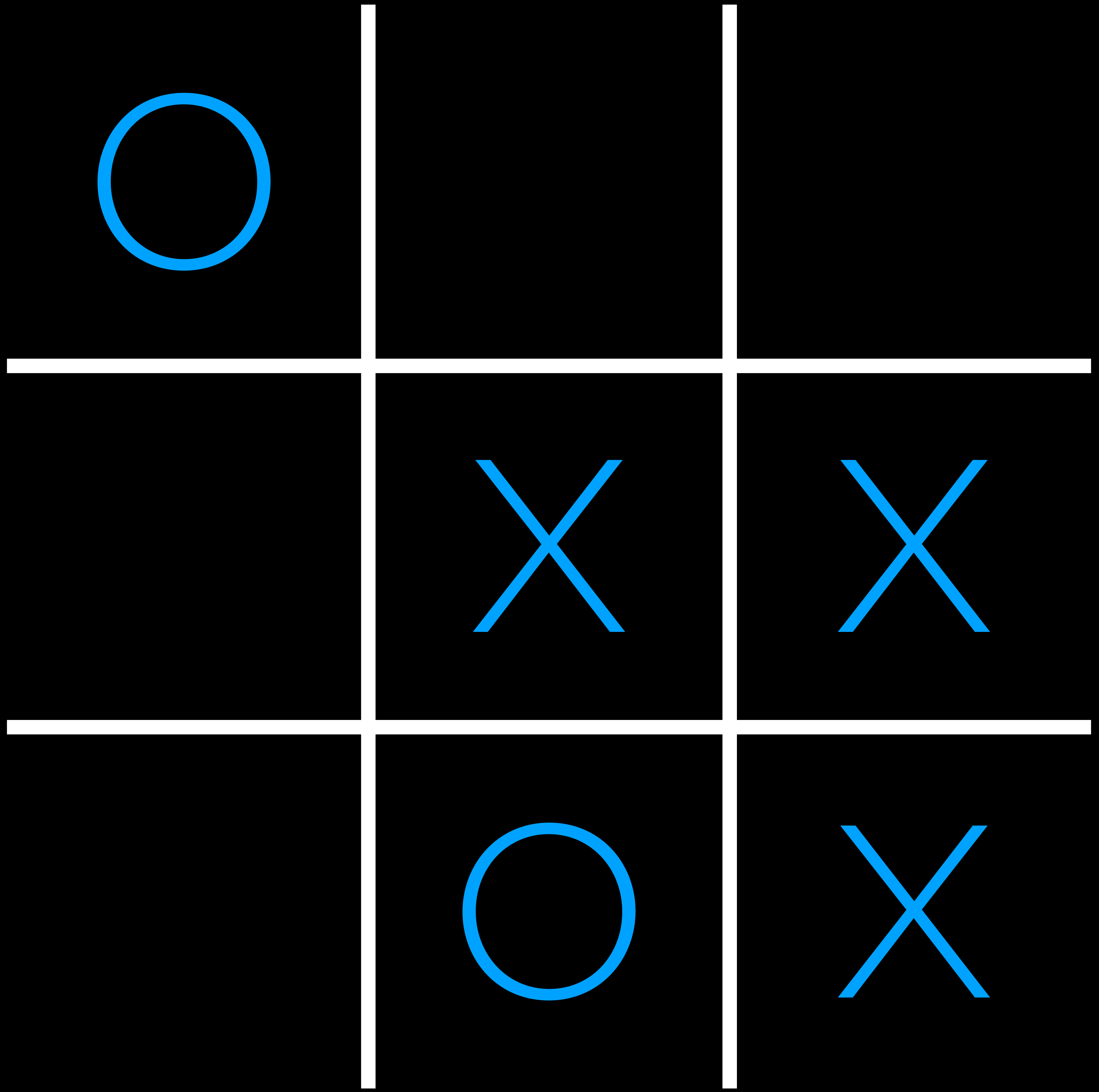


input word
+
positional encoding
Self-Attention
Neural Network
encoded representation
previous output word
+
positional encoding
Self-Attention
Attention
Neural Network
next output word
x N

Language

Artificial Intelligence

Search

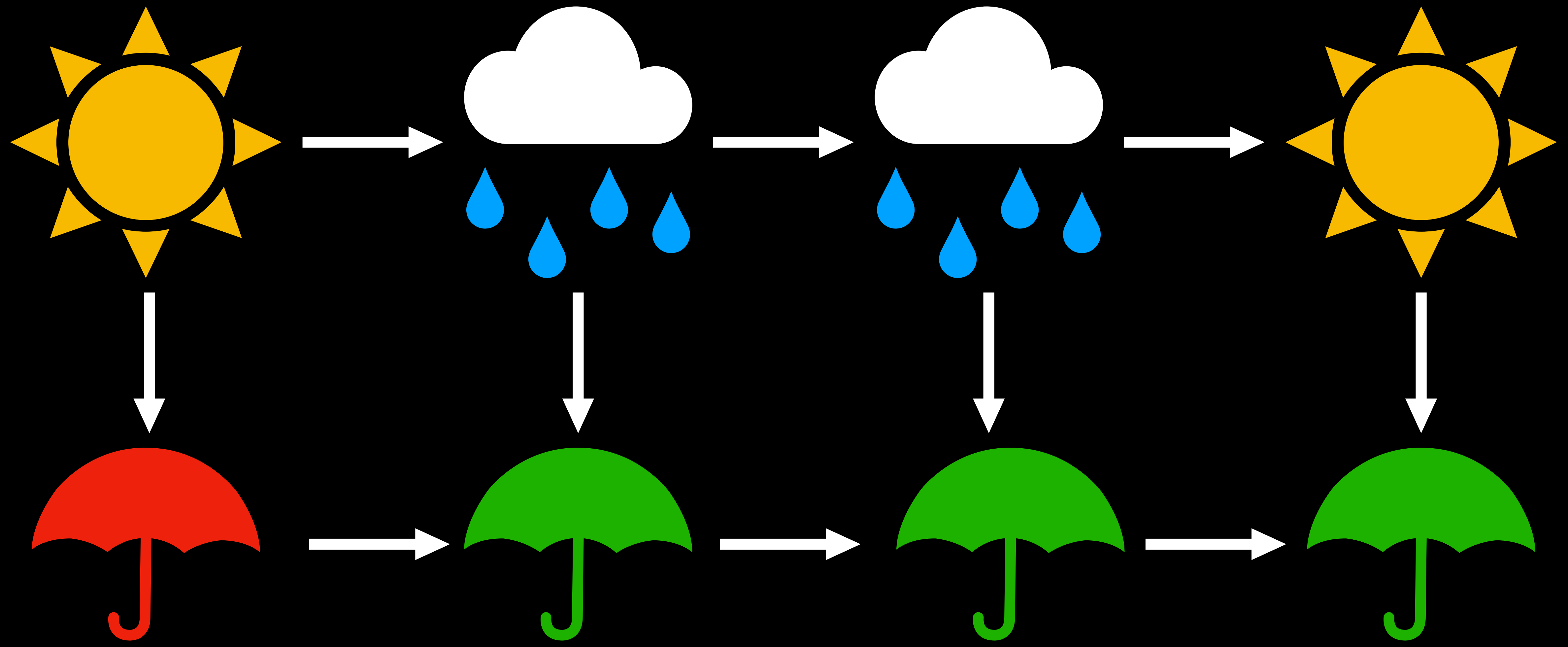


O		
	X	X
	O	X

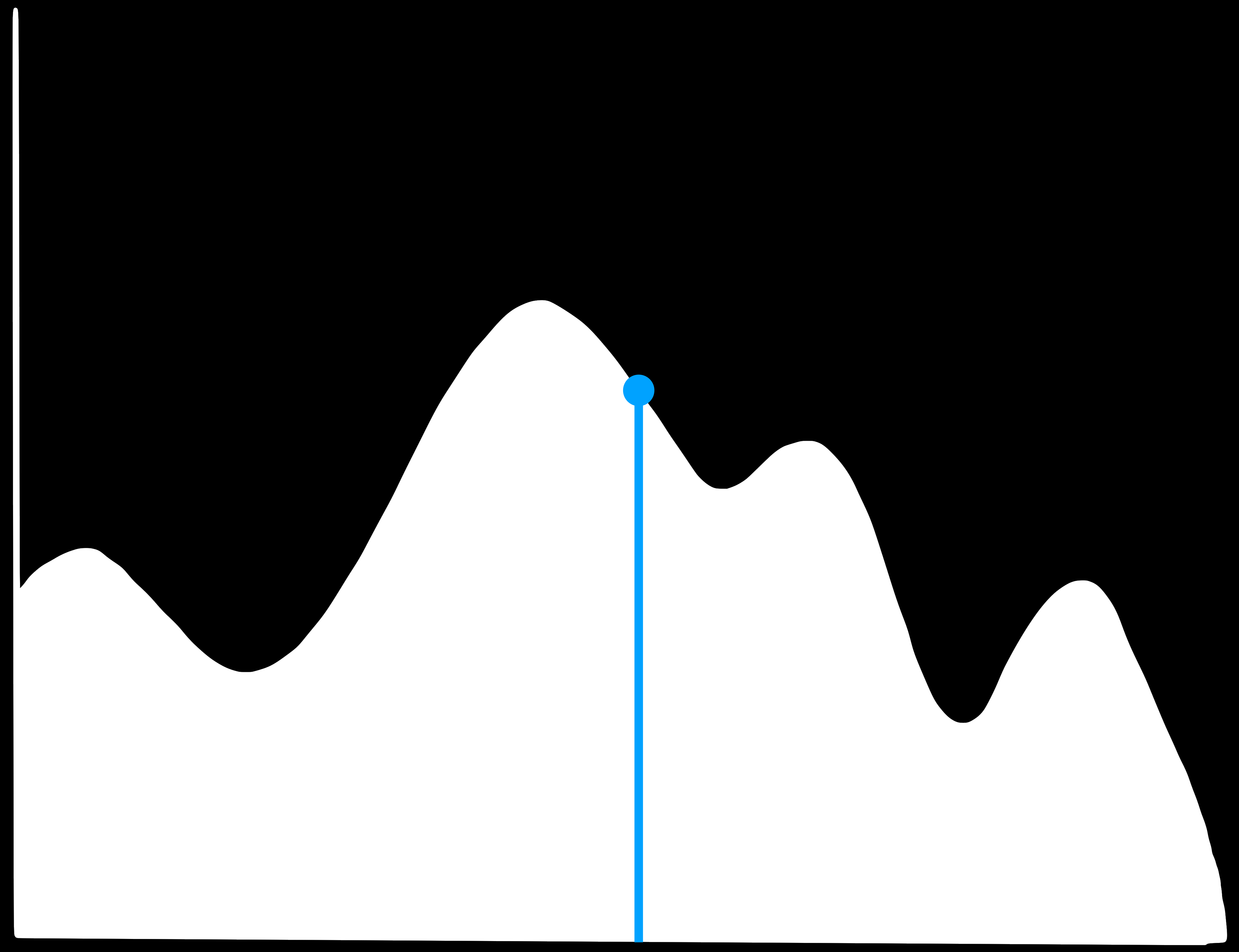
Knowledge

$$\frac{\begin{array}{c} P \to Q \\ P \end{array}}{Q}$$

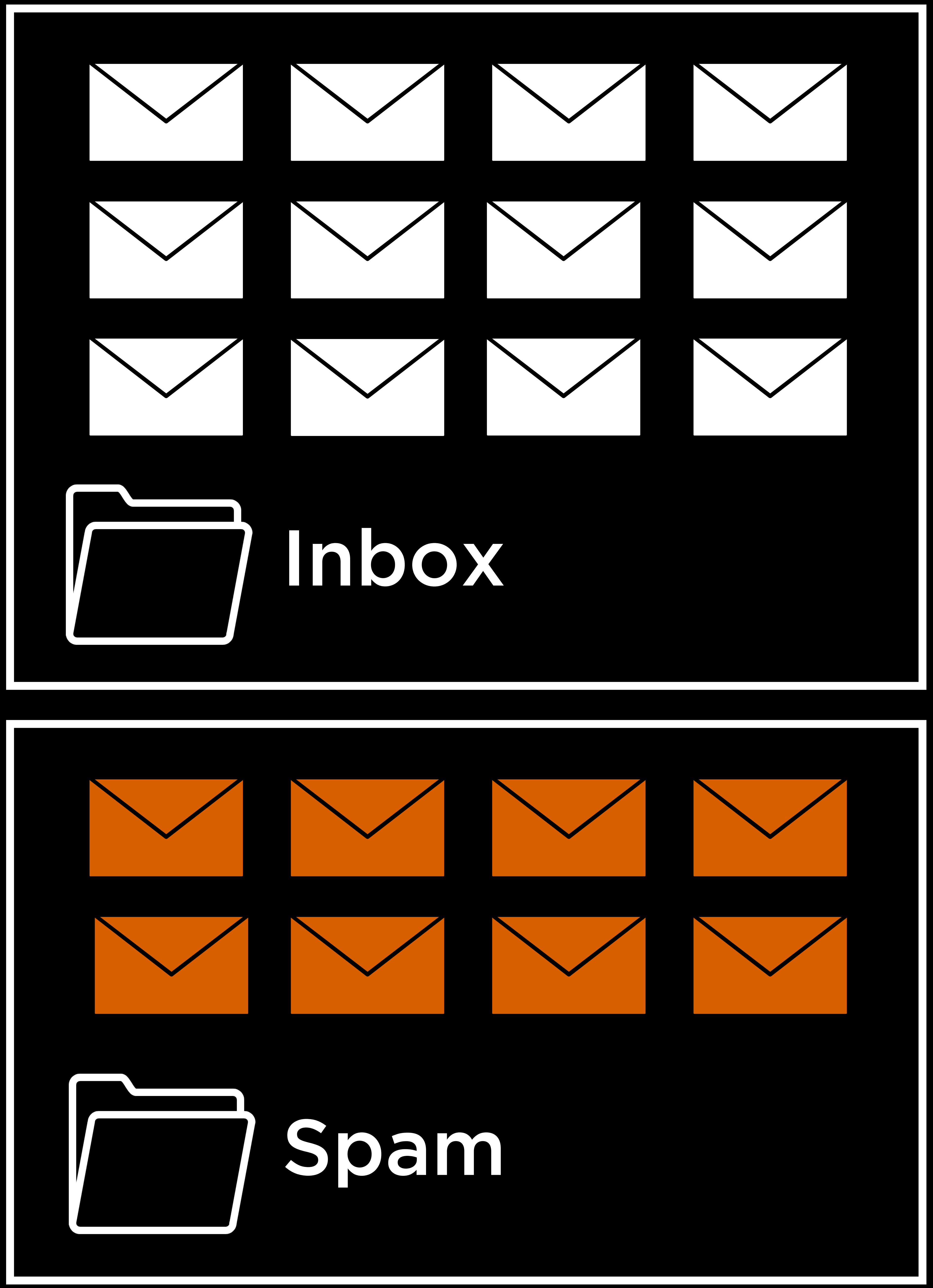
Uncertainty



Optimization



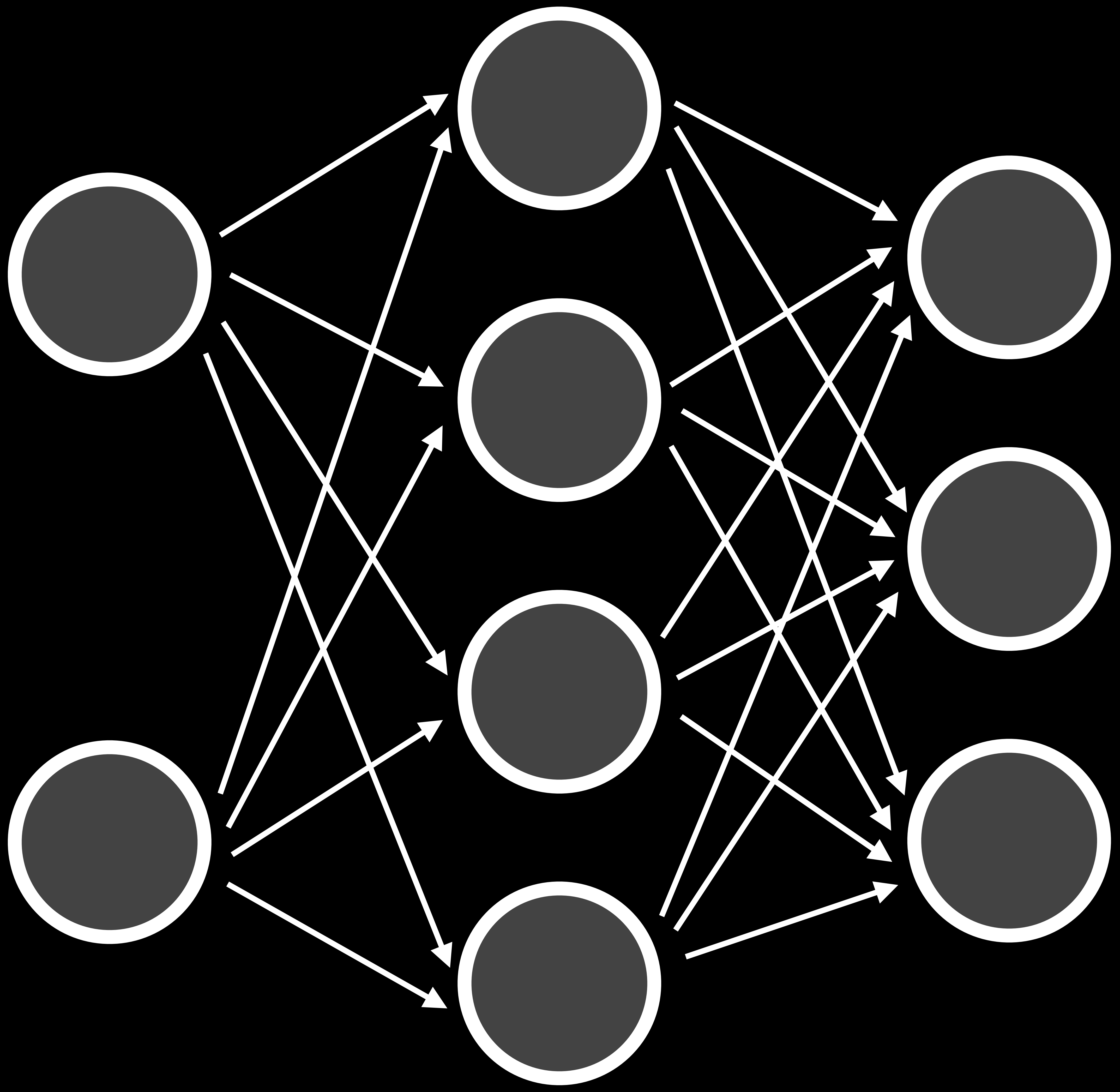
Learning



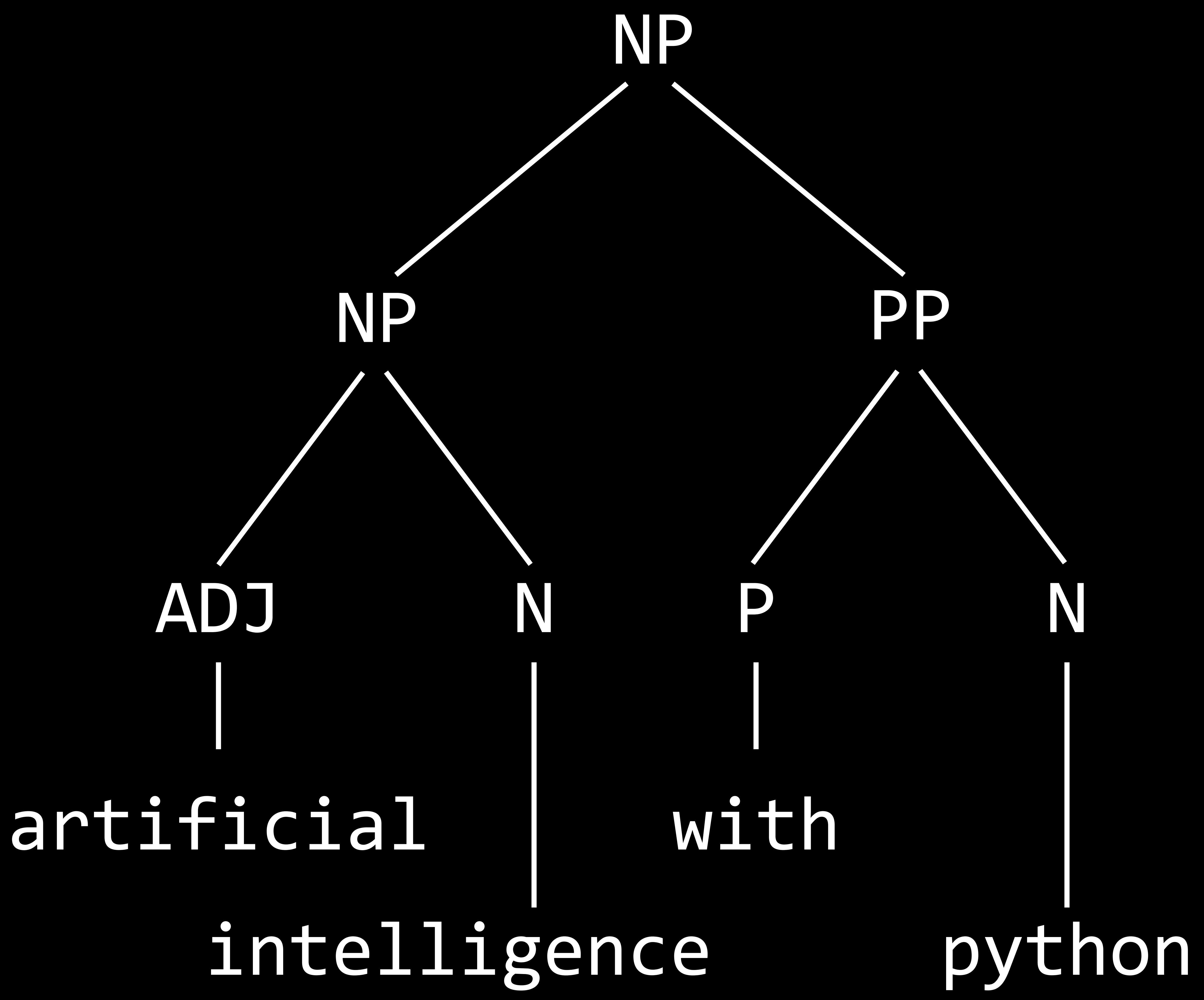
Inbox

Spam

Neural Networks



Language



Introduction to

Artificial Intelligence

with Python