Web search engine / ...

I saw a cat|

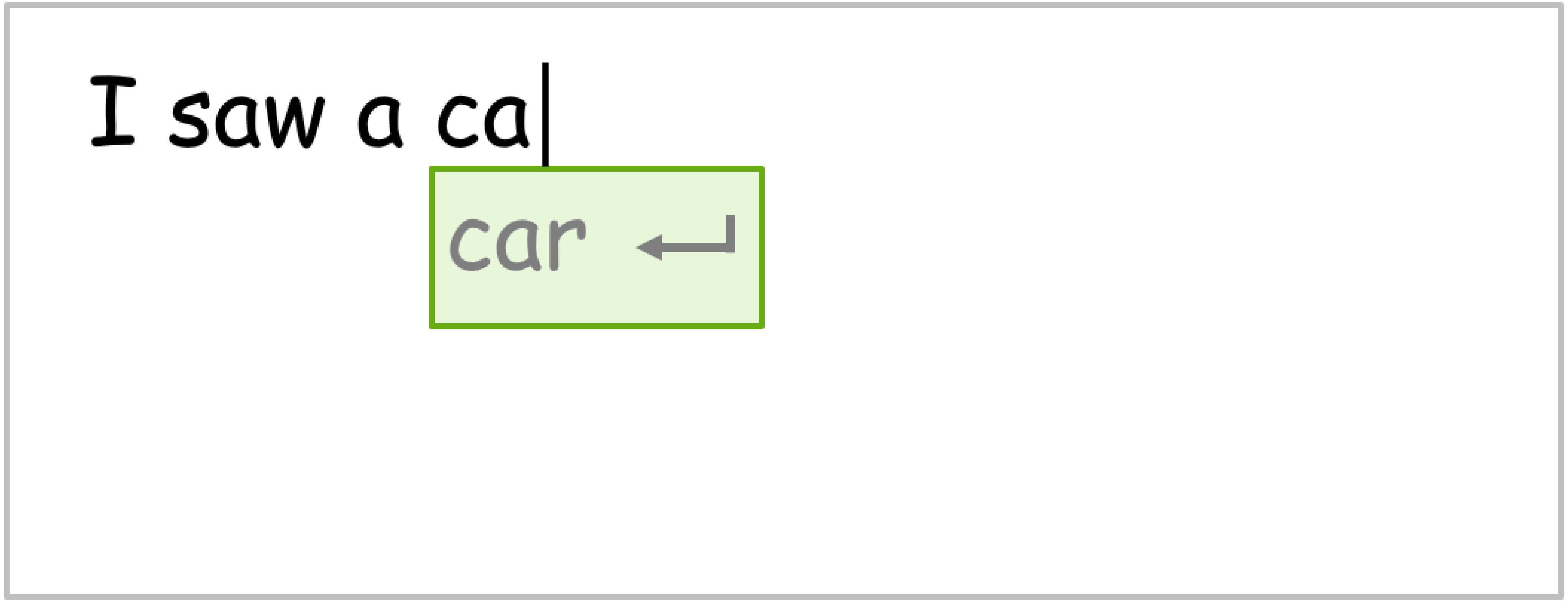
I saw a cat on the chair

I saw a cat running after a dog

I saw a cat in my dream

I saw a cat book

Translation service / mail agent / ...

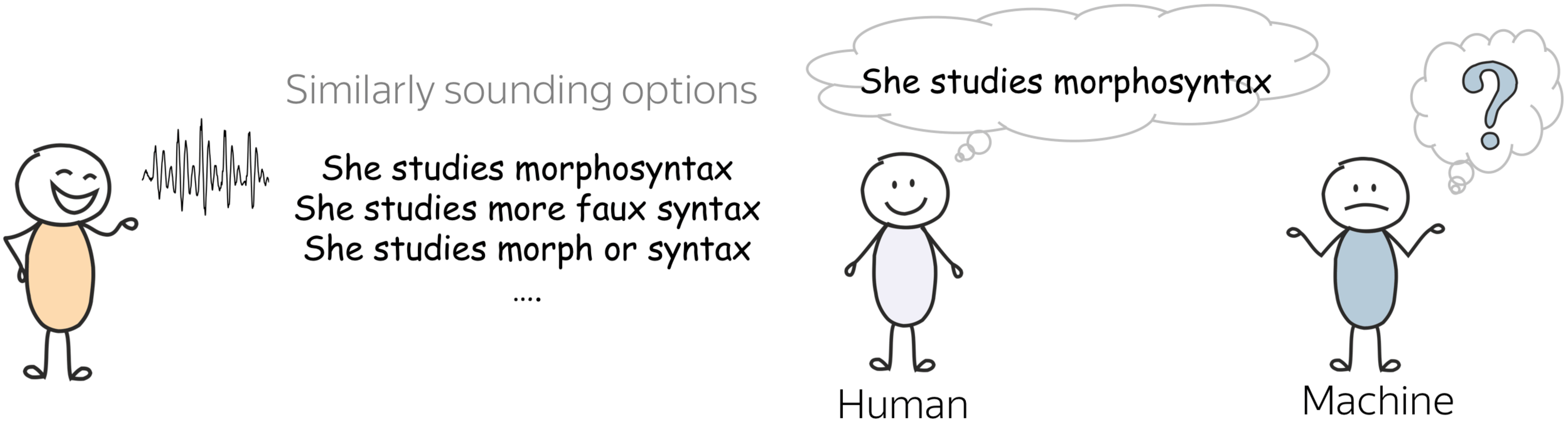


Translation service / mail agent / ...

I saw a catt

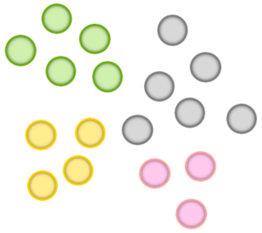
Probably you meant I saw a cat

Keyboard / mail agent / ...



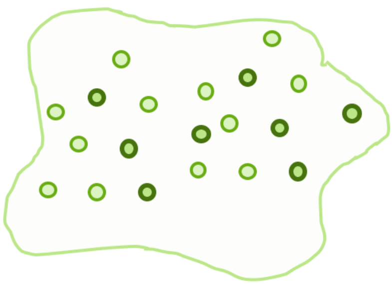
The morphosyntax example is from the slides by Alex Lascarides and Sharon Goldwater, Foundations of Natural Language Processing course at the University of Edinburgh.

What is the probability to pick a green ball?



$$\frac{5}{5+6+4+3} = \frac{5}{18}$$

Can we do the same for sentences?



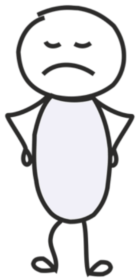
Text corpus

$$P(\text{the mut is tinming the tebn}) = \frac{0}{|\text{corpus}|} = 0$$

$$P(\text{mut the tinming tebn is the}) = \frac{0}{|\text{corpus}|} = 0$$

With this approach, sentences that never occurred in the corpus will receive zero probability

But the first sentence is “more likely” than the second! This method is not good!



P(I) =

$\underbrace{\text{P}(\text{I})}$

Probability of I

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t}).$$

Need to define:

- how to compute $P(y_t|y_1, y_2, \dots, y_{t-1})$

N-gram models

Neural models

I ____

N-gram Language Models

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t}).$$

$$P(y_t|y_1, \dots, y_{t-1}) = \frac{N(y_1, \dots, y_{t-1}, y_t)}{N(y_1, \dots, y_{t-1})},$$

N-gram Language Models

Formally, n-gram models assume that

$$P(y_t|y_1, \ldots, y_{t-1}) = P(y_t|y_{t-n+1}, \ldots, y_{t-1}).$$

For example,

- n=3 (trigram model): $P(y_t|y_1, \ldots, y_{t-1}) = P(y_t|y_{t-2}, y_{t-1})$,
- n=2 (bigram model): $P(y_t|y_1, \ldots, y_{t-1}) = P(y_t|y_{t-1})$,
- n=1 (unigram model): $P(y_t|y_1, \ldots, y_{t-1}) = P(y_t)$.

N-gram Language Models

Before

P(I saw a cat on a mat) =

P(I)
· P(saw | I)
· P(a | I saw)
· P(cat | I saw a)
· P(on | I saw a cat)
· P(a | I saw a cat on)
· P(mat | I saw a cat on a)

After (3-gram)

P(I saw a cat on a mat) =

P(I) → P(I)
· P(saw | I) → · P(saw | I)
· P(a | I saw) → · P(a | I saw)
· P(cat | ~~I~~ saw a) → · P(cat | saw a)
· P(on | ~~I saw~~ a cat) → · P(on | a cat)
· P(a | ~~I saw a~~ cat on) → · P(a | cat on)
· P(mat | ~~I saw a cat~~ on a) → · P(mat | on a)

ignore use

N-gram Language Models

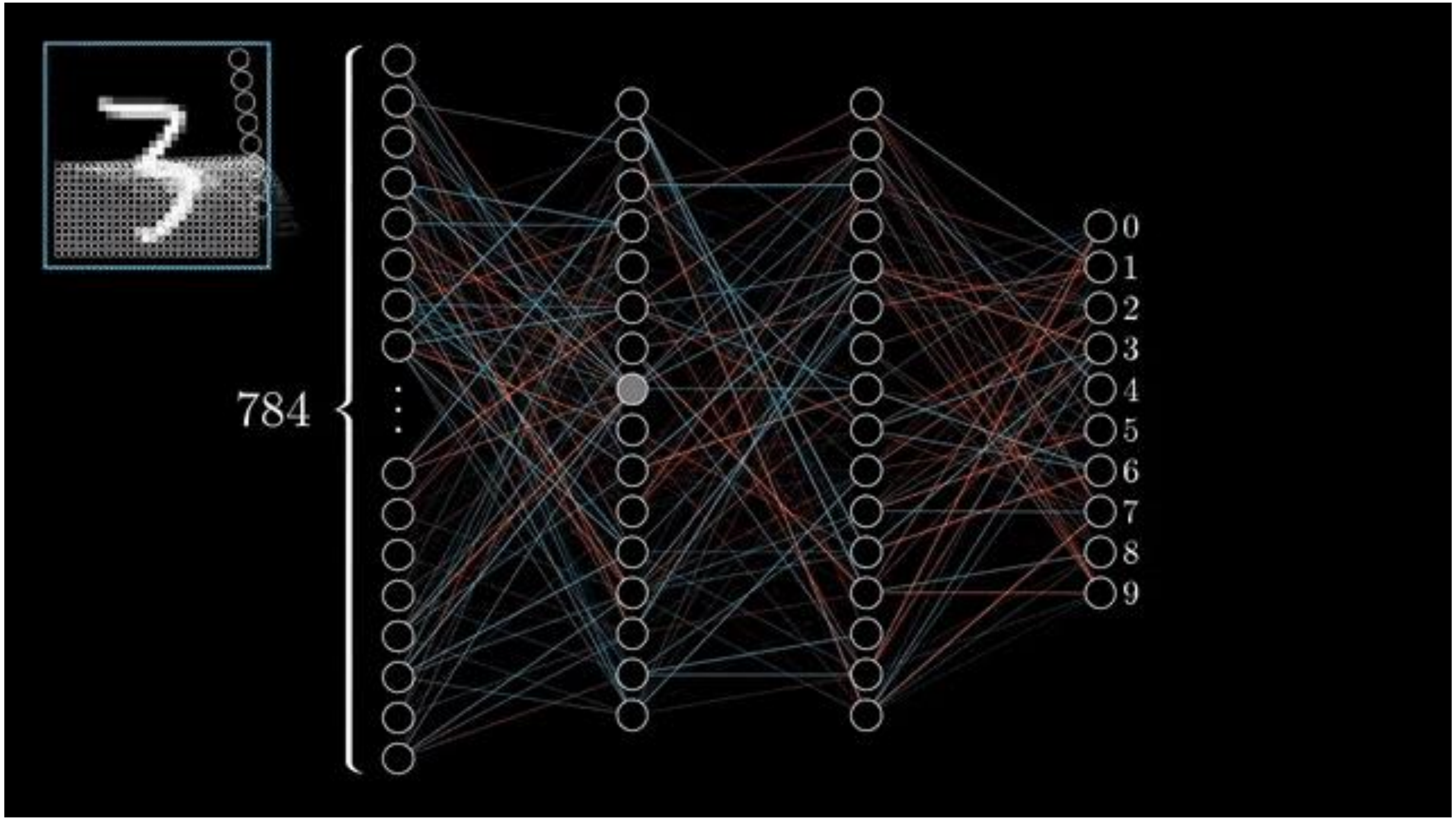
I ____

N-gram Language Models

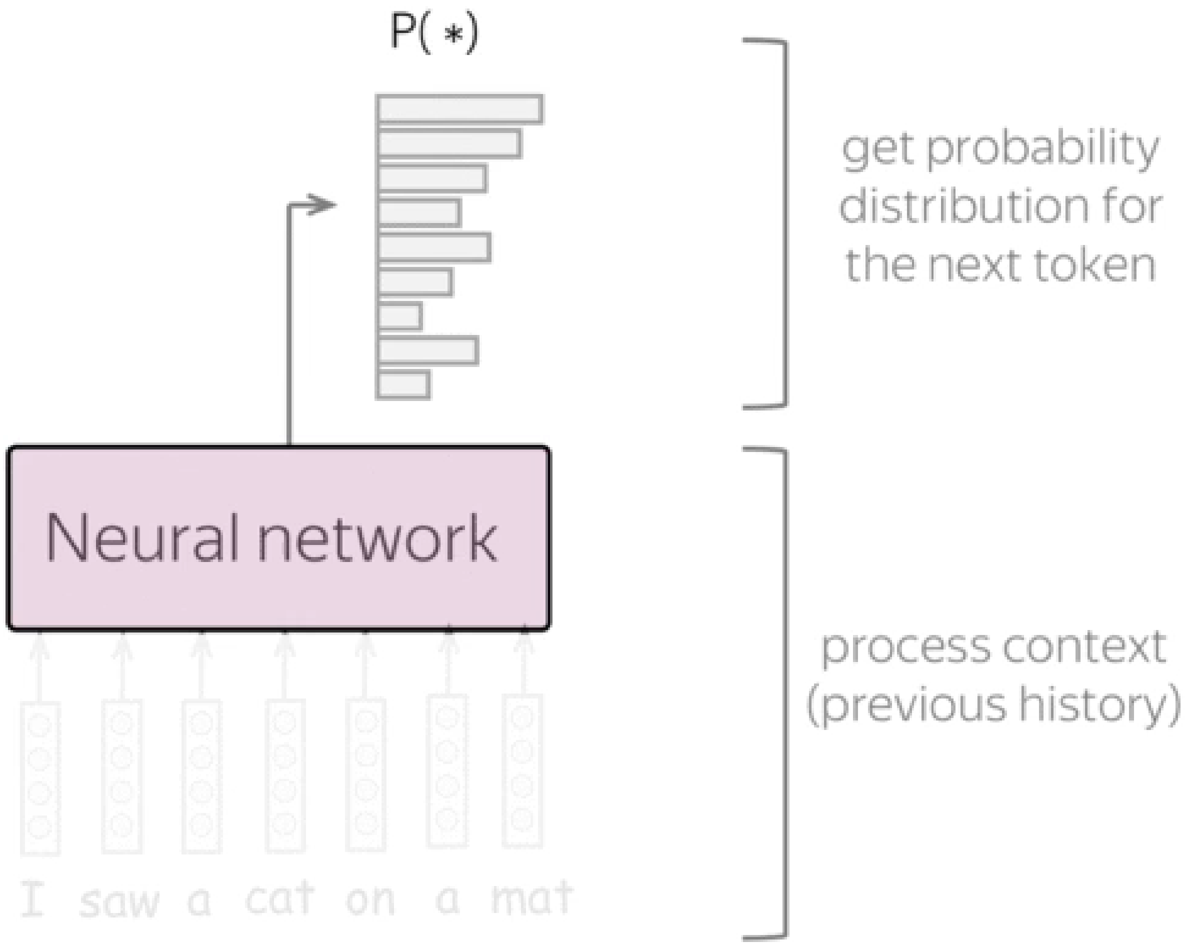
```
so even when i talk a bit short , there was no easy
thing to do different buffer flushing strategies in the
future , due to huge list of number - one just has
started production of frits in the process and has free
wi - fi ” operation .... _eos_
```

N-gram Language Models

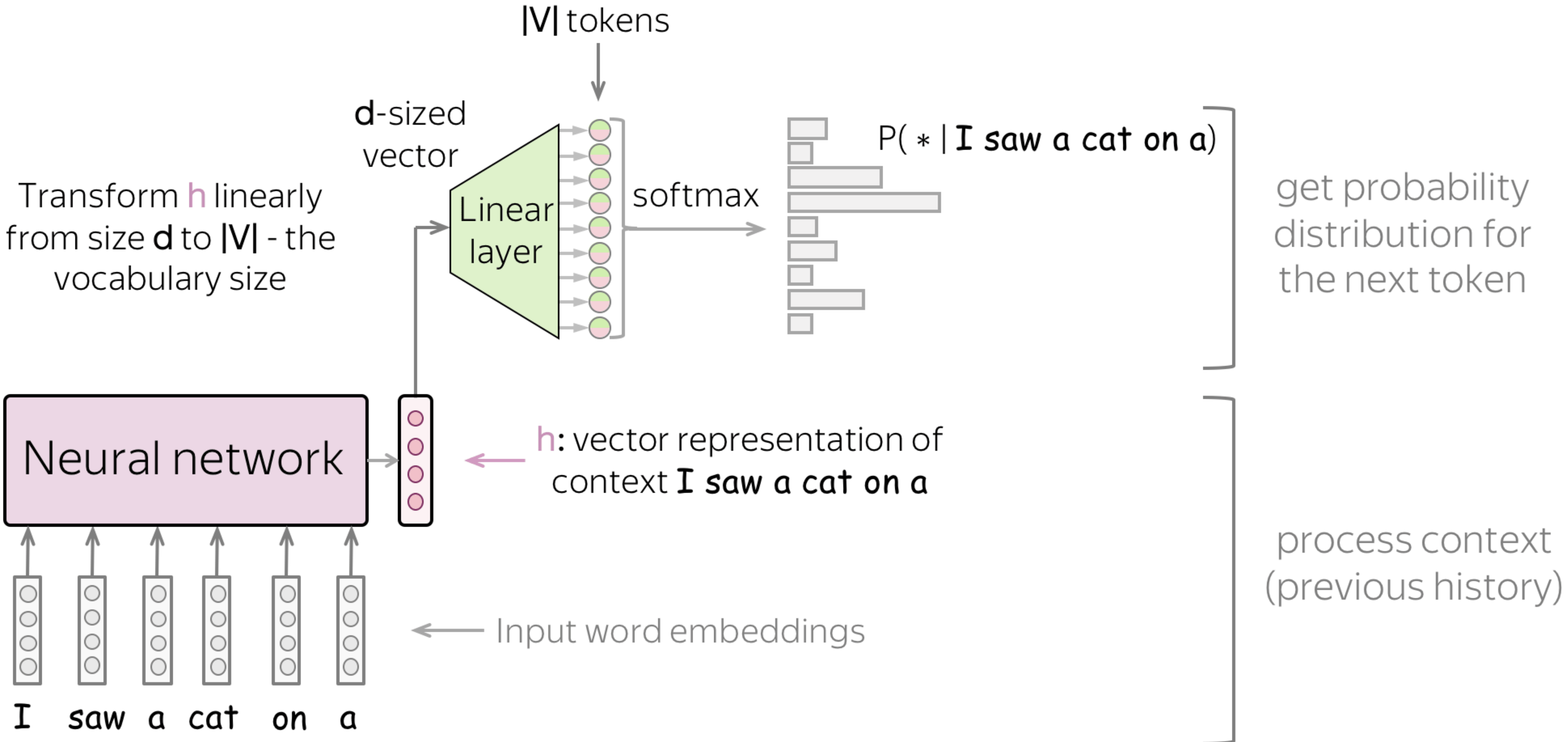
```
so even if the us , and the united states , the hotel is
located in the list of songs , you can add them in our
collection by this form . _eos_
```

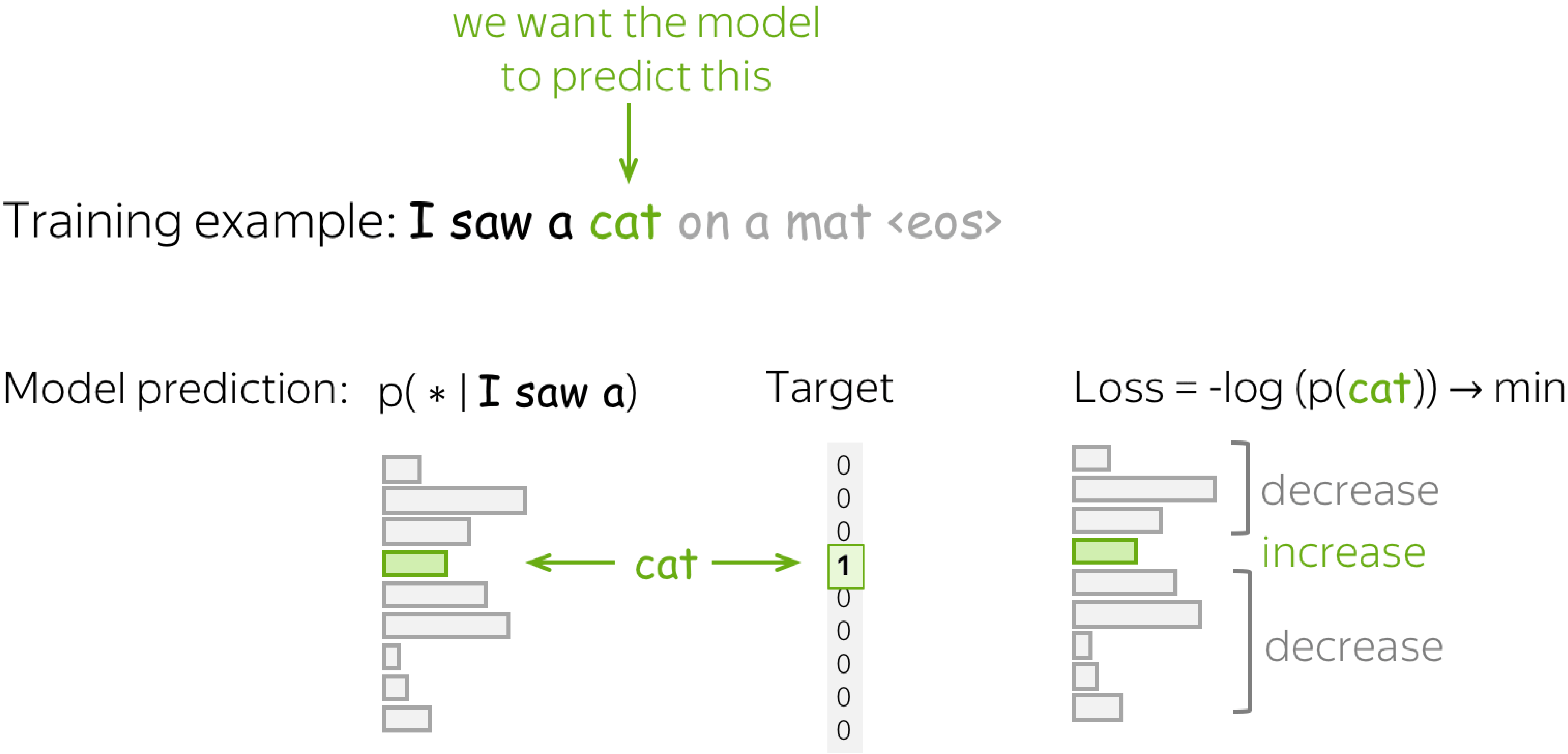

784
0
1
2
3
4
5
6
7
8
9

Neural Language Models



Neural Language Models




we want the model
to predict this
Training example: I saw a cat on a mat <eos>
Model prediction: p(* | I saw a)
Target
Loss = -log (p(cat)) → min
cat
0
0
0
1
0
0
0
0
0
decrease
increase
decrease

Initial
RNN state

Start: do not have
input, want to predict
the first token

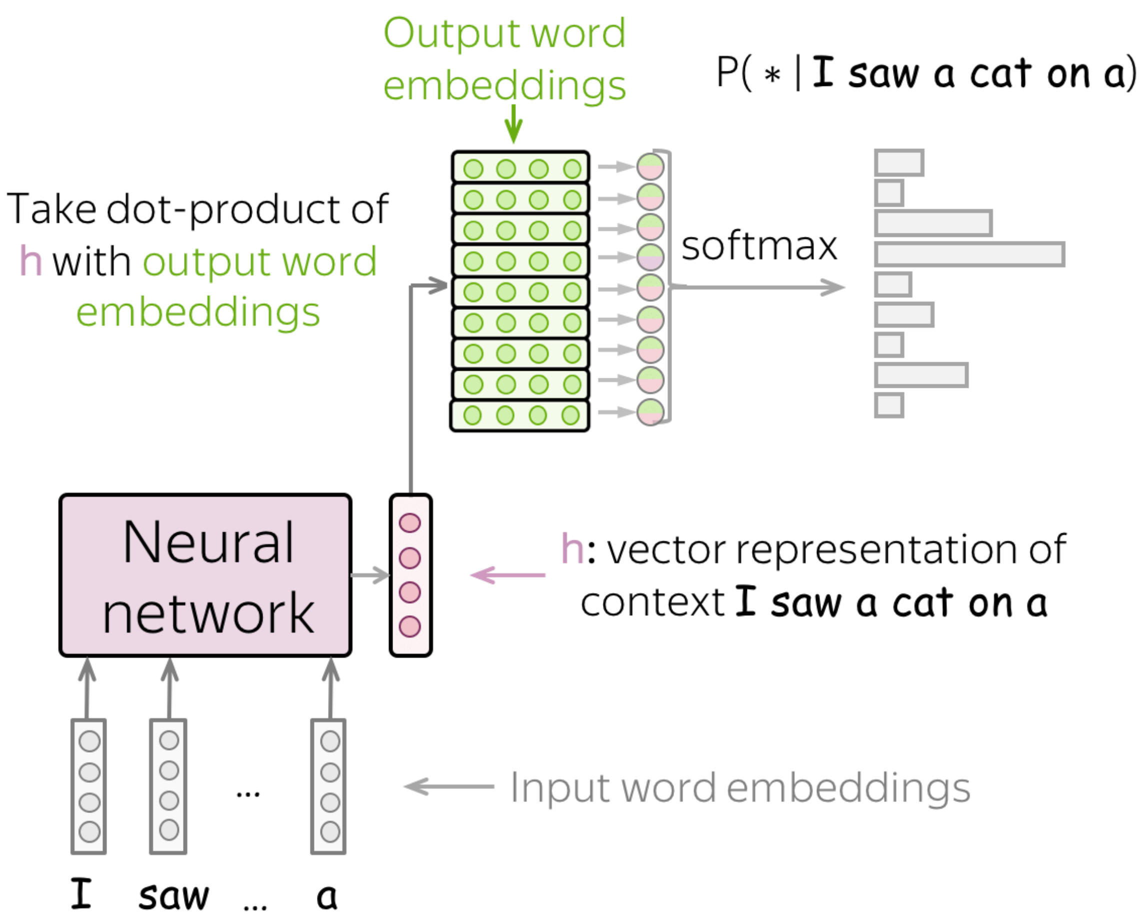
we want the model
to predict this

↓

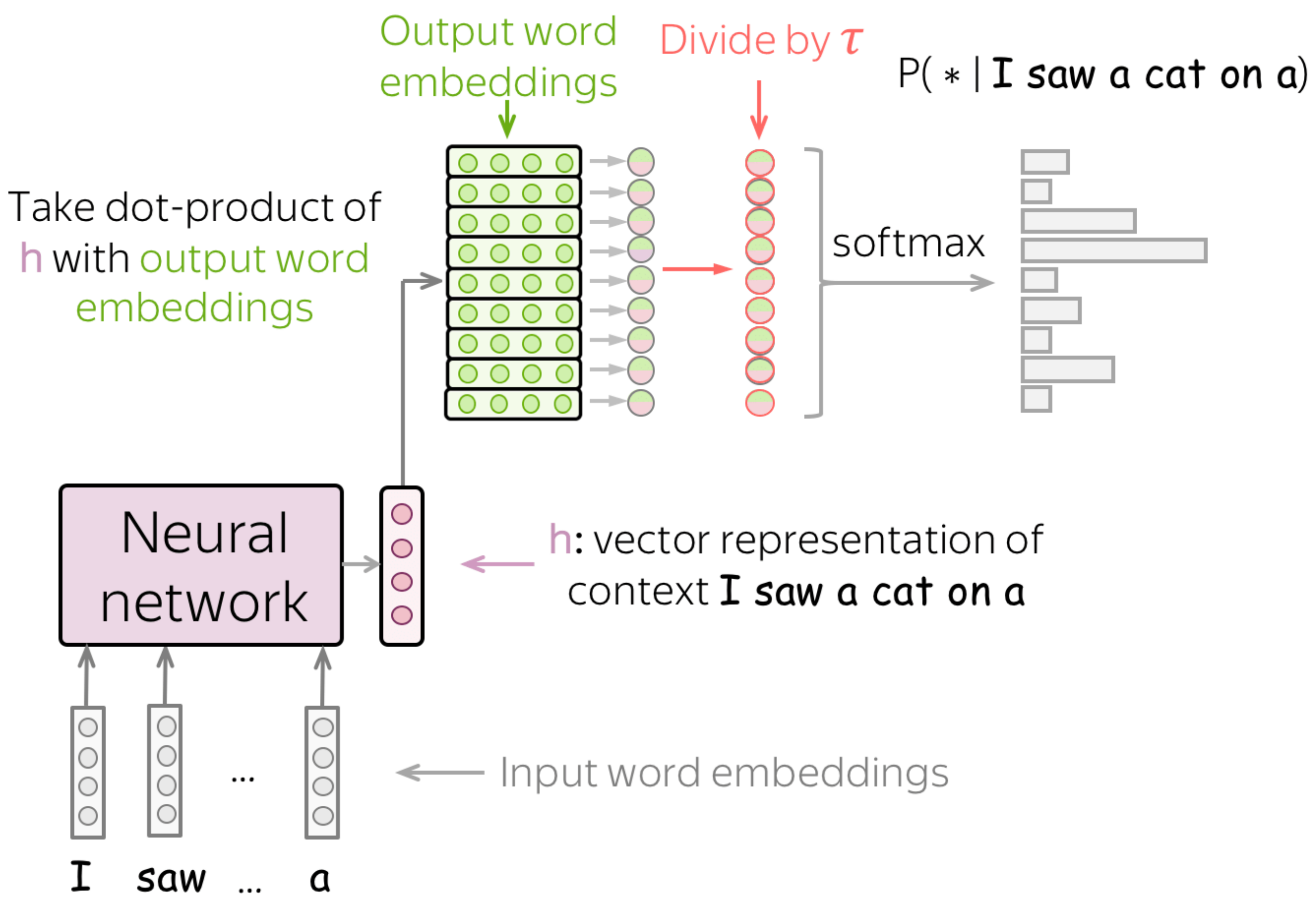
Training example: I saw a cat on a mat <eos>

Sampling with temperature

Before

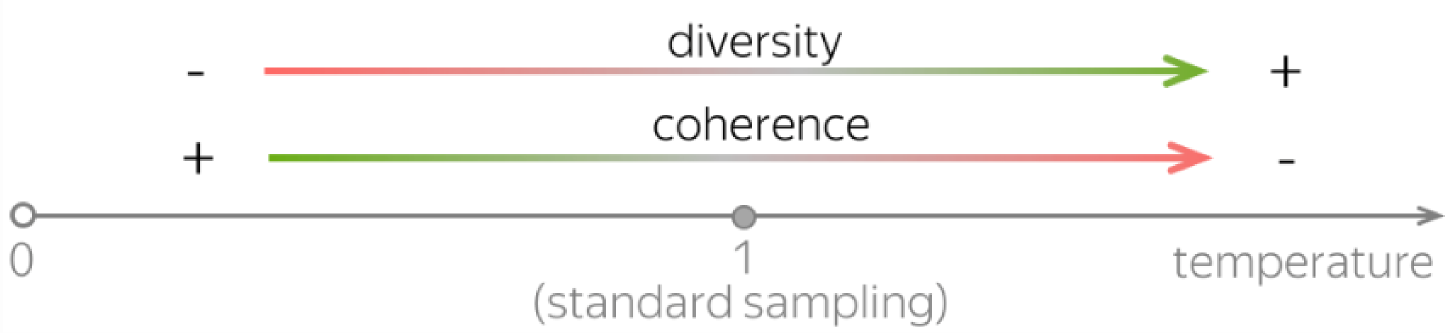


After



```
paradise sits farms started paint hollow almost
unprecedented decisions, care using withdrawal from
rebel cis ( , saying graphics mongolia official line,
greeted agenda victor is exploring anger :) draw testify
liberalization decay productive 2 went exchanges of
marketing drawing enabling challenging systematic crisis
influencing the executive arrangement performs designs
```

```
the first time the two - year - old - old girl with a
new version of the new version of the new version of the
new version of the new version of the new version of the
new version of the new version of the new version of the
```



Input Prompt: `Recite the first law of robotics`

Output:

Untrained
GPT-3

Unsupervised Pre-training

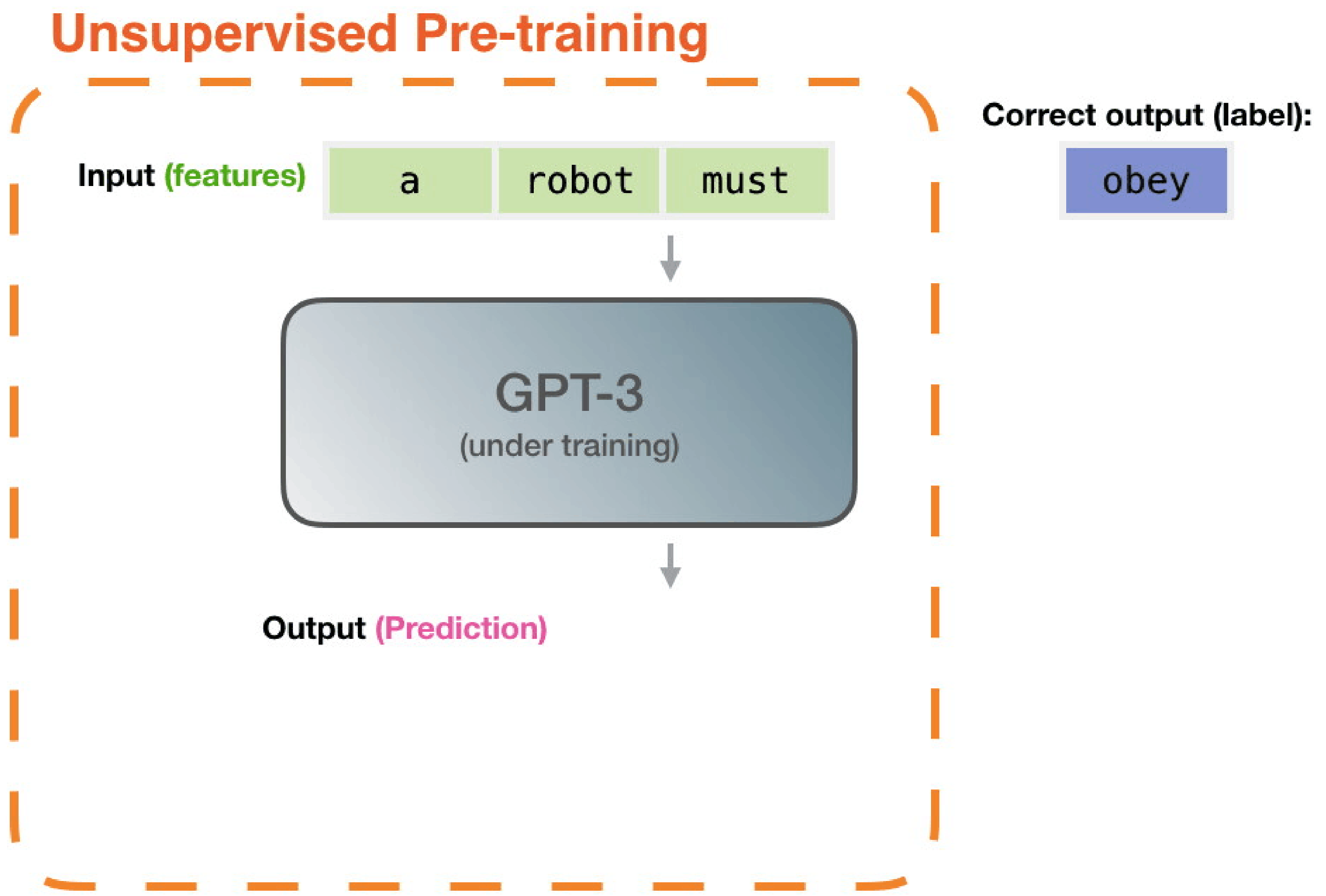
Expensive training on massive datasets

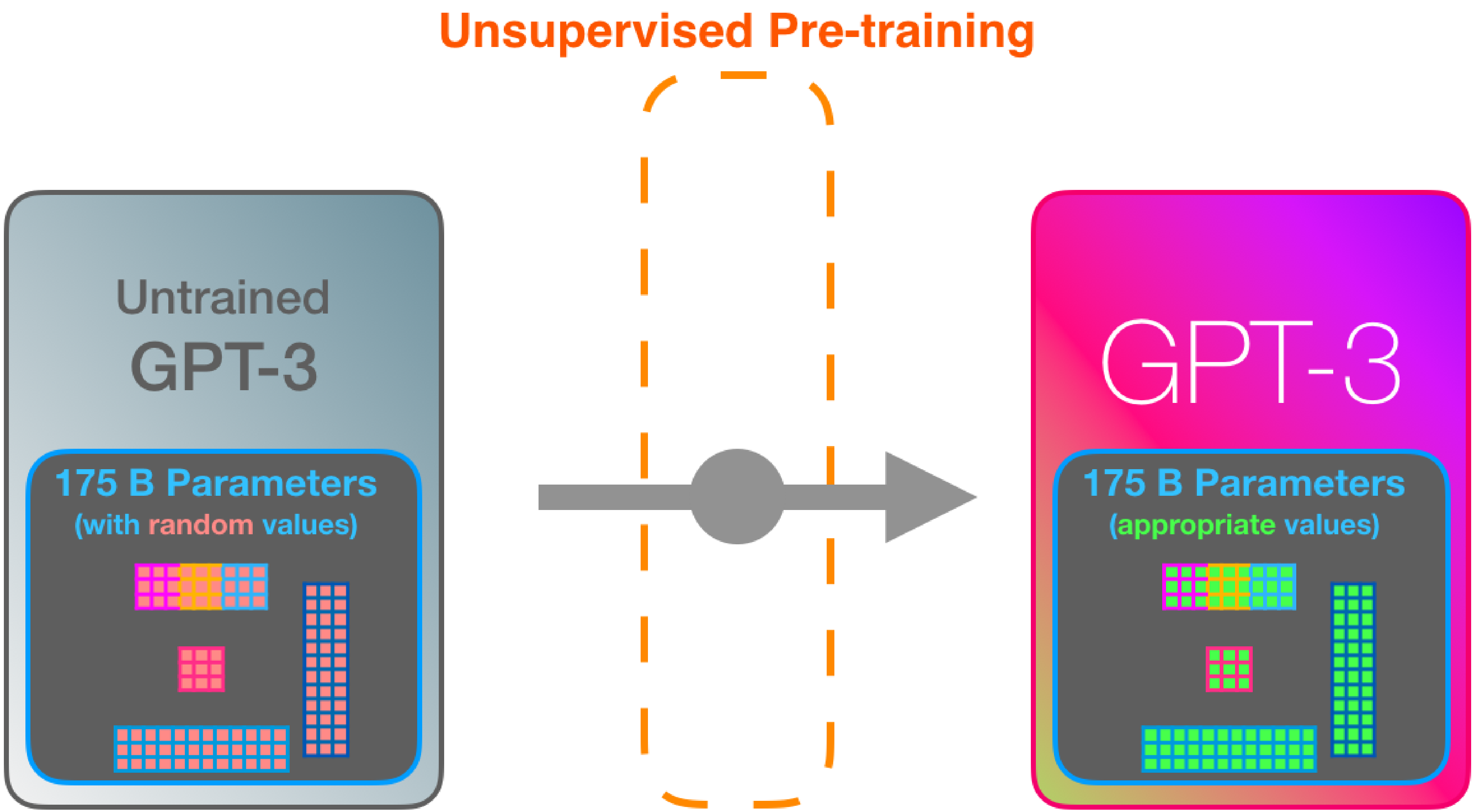
Dataset: 300 billion tokens of text

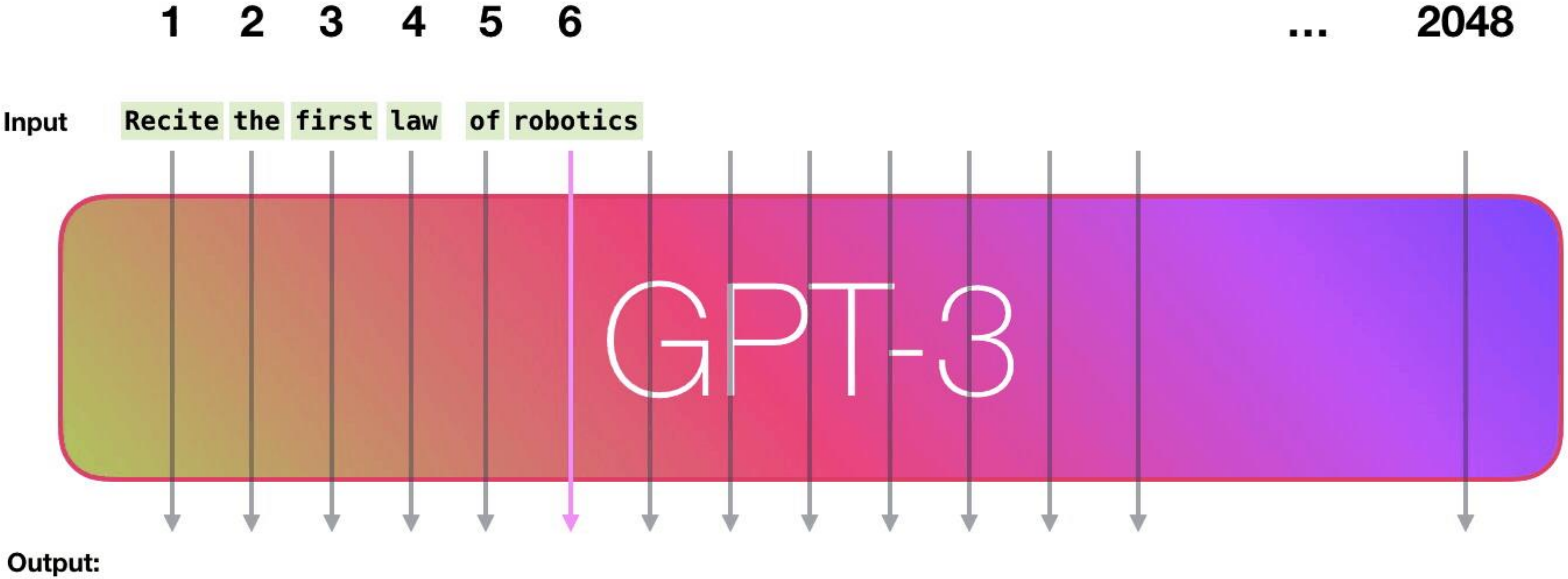
Objective: Predict the next word

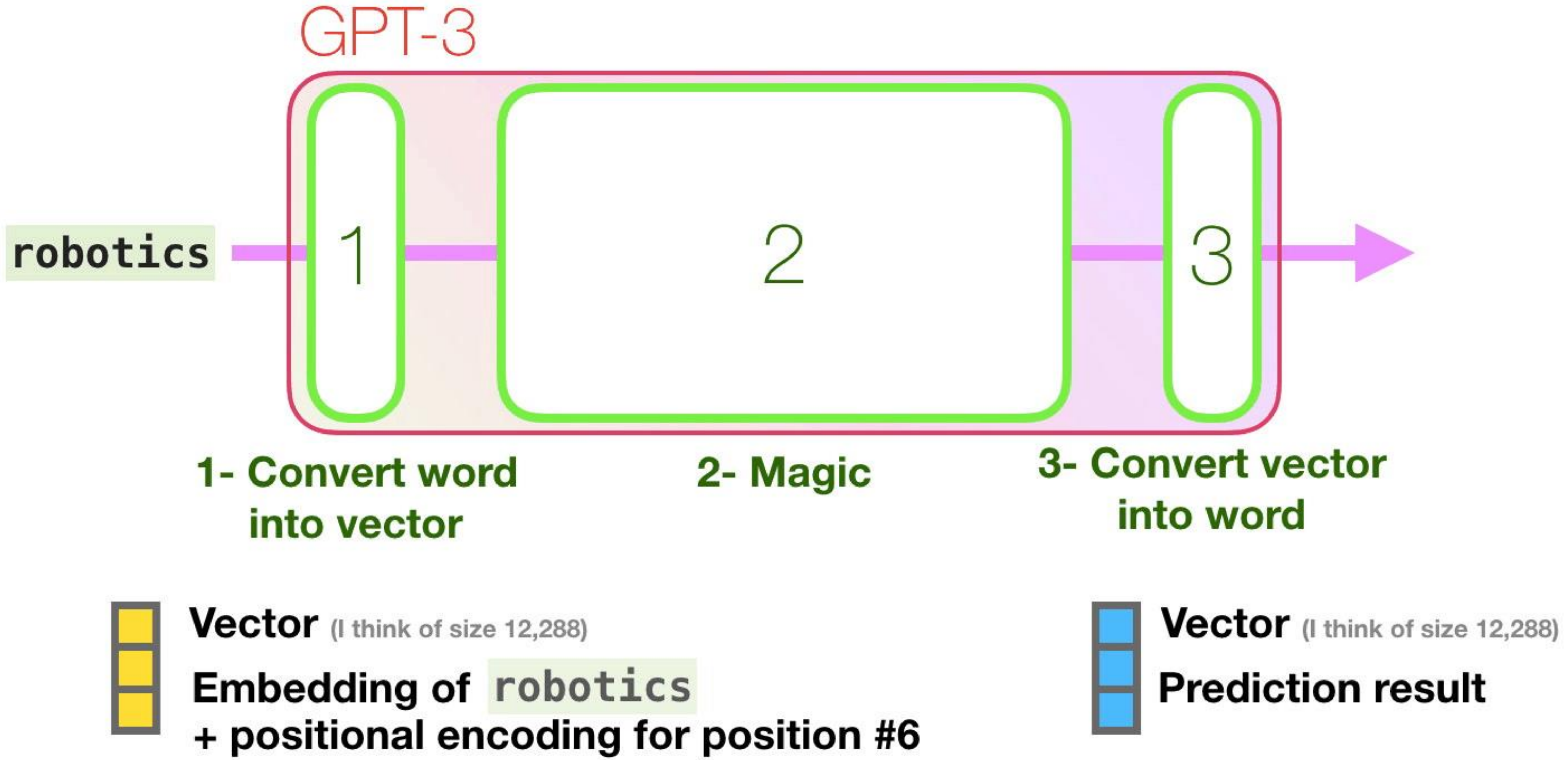
Example:

a	robot	must	?

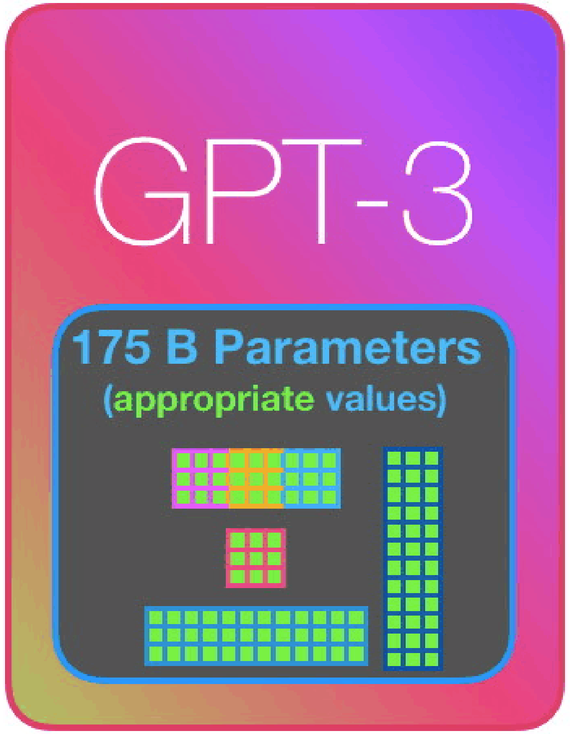

Unsupervised Pre-training
Correct output (label):
obey
Input (features)
a
robot
must
GPT-3
(under training)
Output (Prediction)


Unsupervised Pre-training
Untrained
GPT-3
175 B Parameters
(with random values)
GPT-3
175 B Parameters
(appropriate values)


1
2
3
4
5
6
...
2048
Input
Recite
the
first
law
of
robotics
GPT-3
Output:


GPT-3
robotics
1
2
3
1- Convert word into vector
2- Magic
3- Convert vector into word
Vector (I think of size 12,288)
Embedding of robotics
+ positional encoding for position #6
Vector (I think of size 12,288)
Prediction result

Pre-training

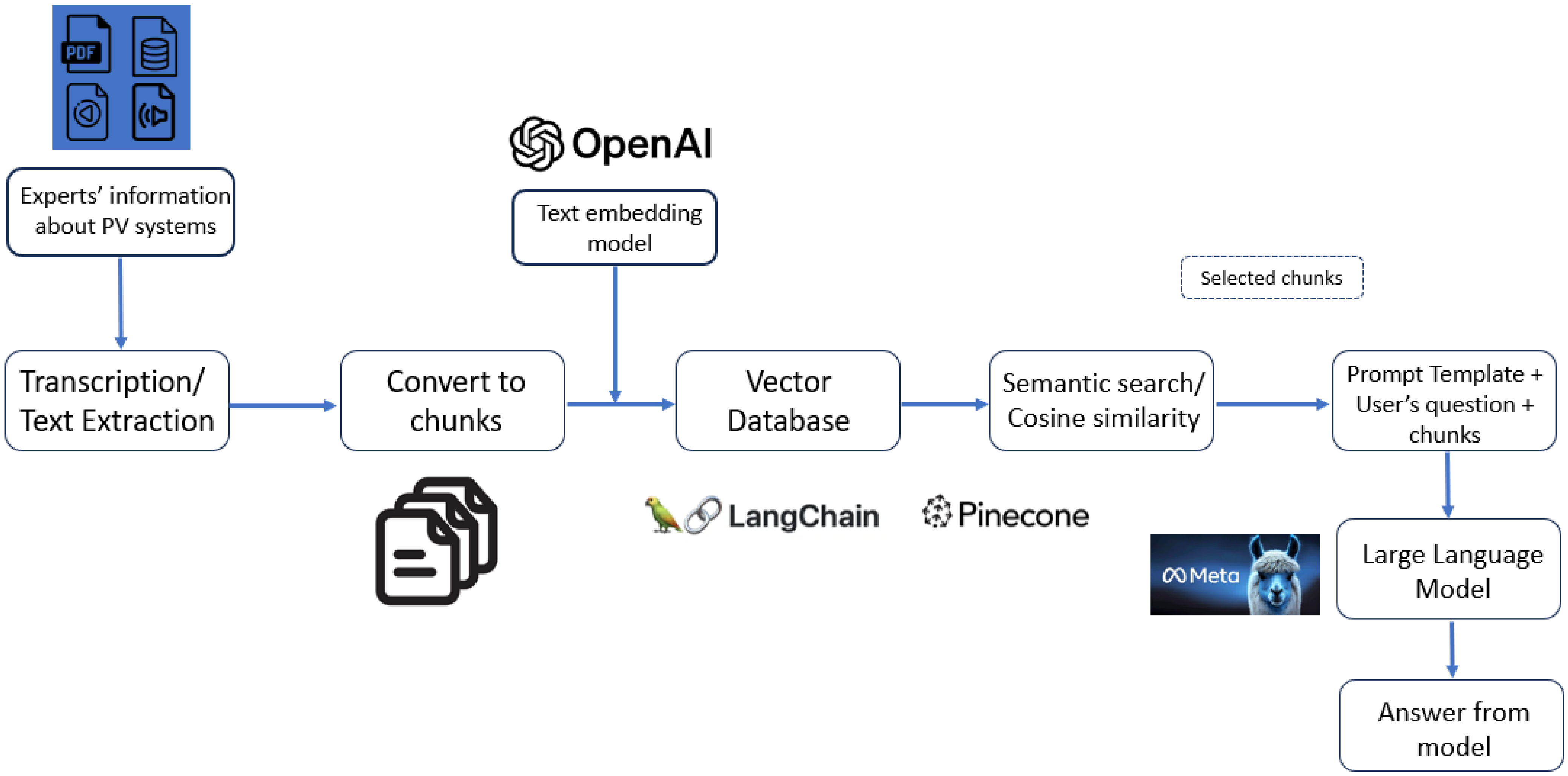


Fine-tuning

Additional training to become better at a certain task

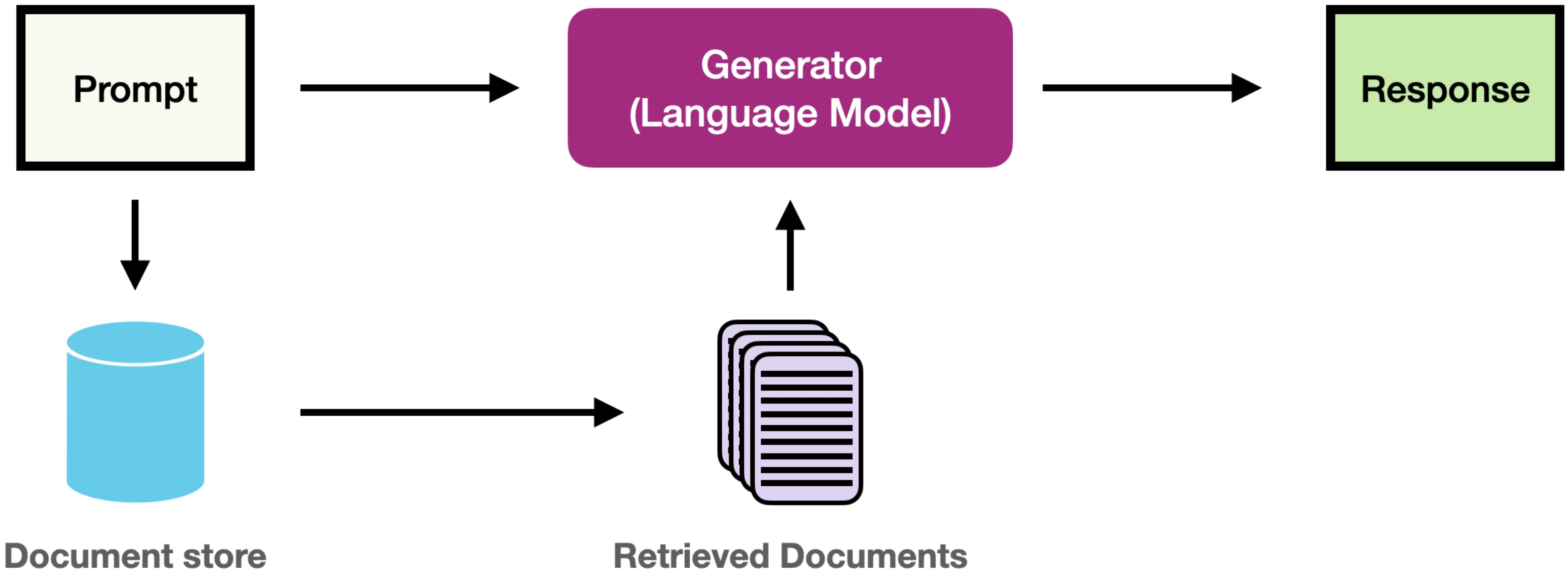
Example: English to French Translation


User Input
Can you recommend a delicious recipe for dinner?
LLM Response
Yes, here is a delicious recipe for lunch. So how about fried chicken with mashed potatoes? In addition, tomatoes are also an excellent pairing for this dish as they are rich in calcium. Enjoy this steak!
Hallucination Explanation
Input-Conflicting Hallucination: the user wants a recipe for dinner while LLM provide one for lunch.
Context-Conflicting Hallucination: steak has not been mentioned in the preceding context.
Fact-Conflicting Hallucination: tomatoes are not rich in calcium in fact.



PDF
OpenAI
Experts' information about PV systems
Text embedding model
Selected chunks
Transcription/ Text Extraction
Convert to chunks
Vector Database
Semantic search/ Cosine similarity
Prompt Template + User's question + chunks
LangChain
Pinecone
Meta
Large Language Model
Answer from model

Retrieval Augmented Generation



One-hot Encoding

		Rome	Paris						word V
Rome	=	[1,	0,	0,	0,	0,	0,	…,	0]
Paris	=	[0,	1,	0,	0,	0,	0,	…,	0]
Italy	=	[0,	0,	1,	0,	0,	0,	…,	0]
France	=	[0,	0,	0,	1,	0,	0,	…,	0]

Word Embeddings

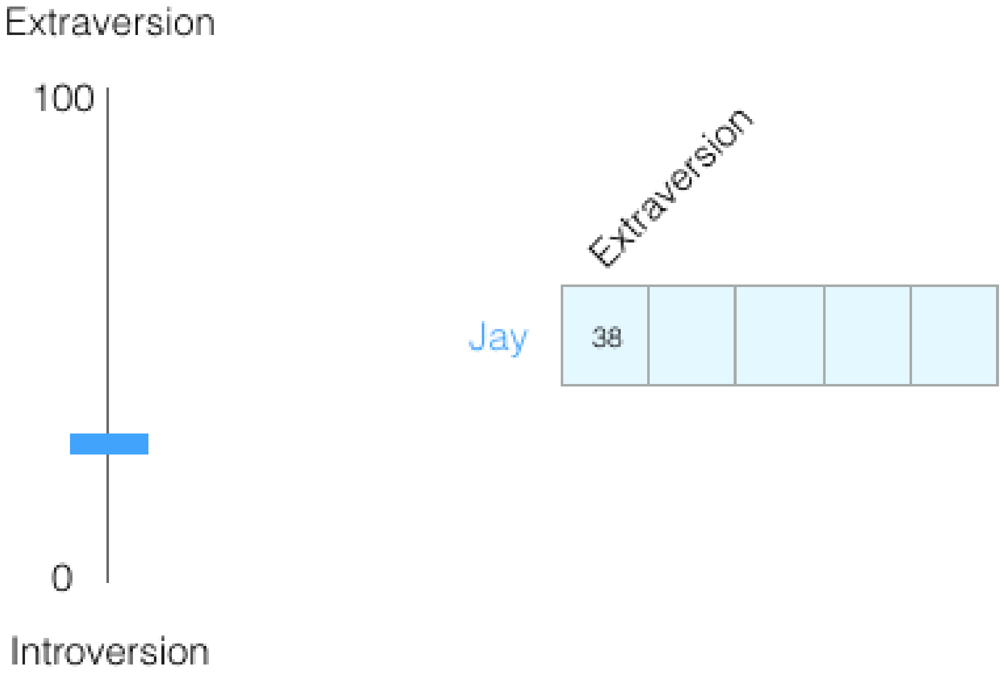
Rome = [0.91, 0.83, 0.17, …, 0.41]

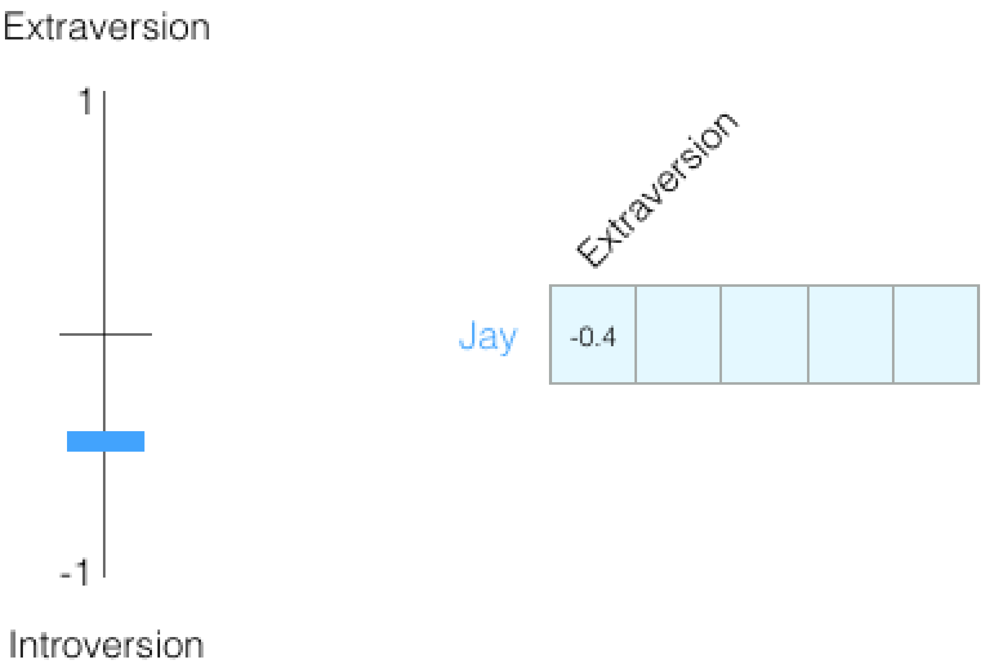
Paris = [0.92, 0.82, 0.17, …, 0.98]

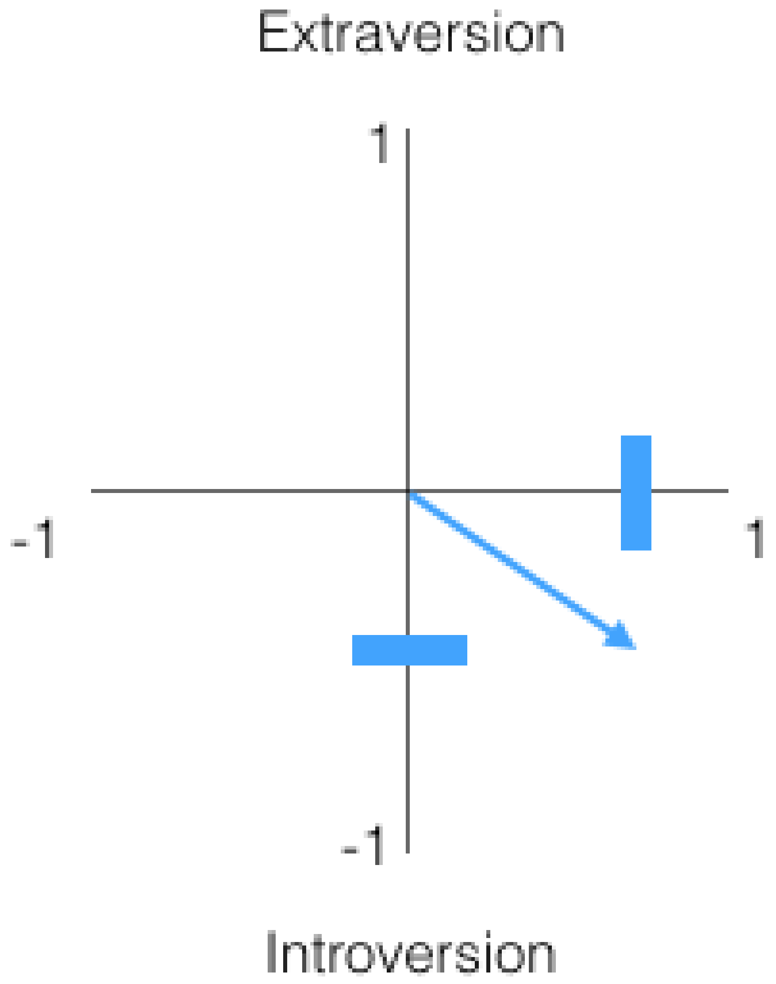
Italy = [0.32, 0.77, 0.67, …, 0.42]

France = [0.33, 0.78, 0.66, …, 0.97]

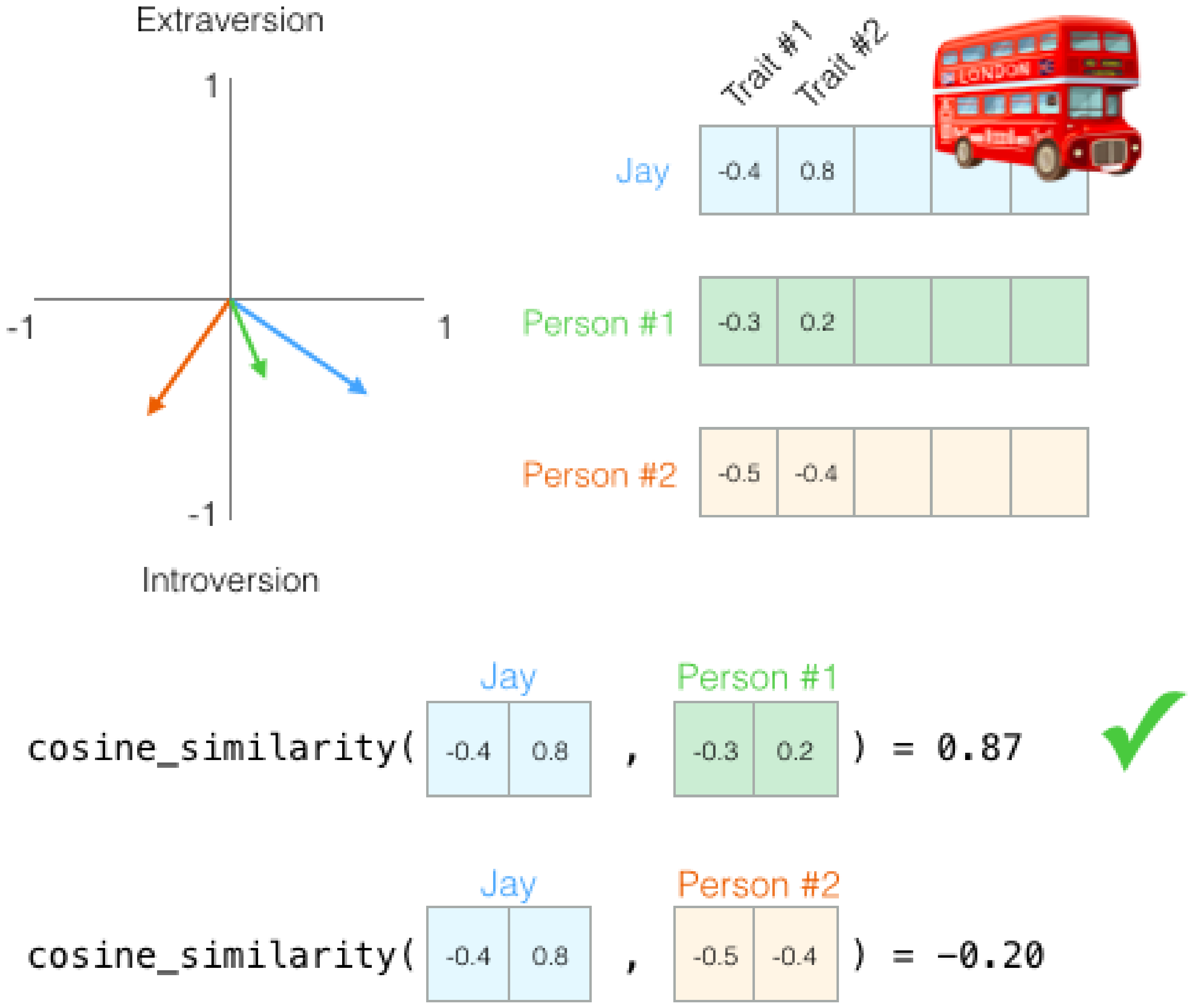
Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100

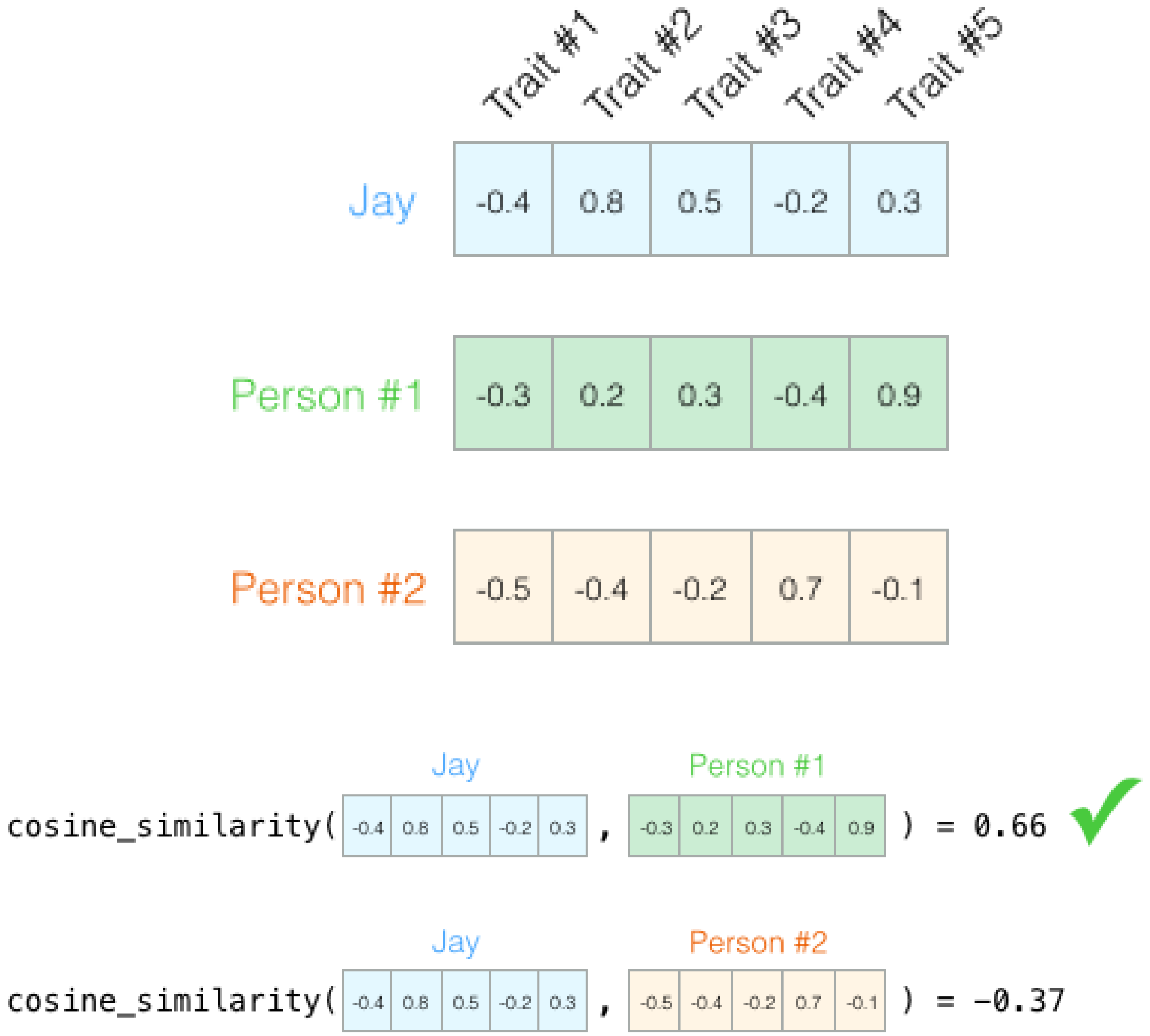

Extraversion
100
0
Introversion
Extraversion
Jay
38


Extraversion
1
-1
Introversion
Extraversion
Jay
-0.4



	Trait #1	Trait #2			
Jay	-0.4	0.8			


Extraversion
1
-1
1
-1
Introversion
Trait #1
Trait #2
Jay
-0.4
0.8
Person #1
-0.3
0.2
Person #2
-0.5
-0.4
cosine_similarity(Jay -0.4 0.8 , Person #1 -0.3 0.2) = 0.87
cosine_similarity(Jay -0.4 0.8 , Person #2 -0.5 -0.4) = -0.20


Trait #1
Trait #2
Trait #3
Trait #4
Trait #5
Jay
-0.4 0.8 0.5 -0.2 0.3
Person #1
-0.3 0.2 0.3 -0.4 0.9
Person #2
-0.5 -0.4 -0.2 0.7 -0.1
Jay
Person #1
cosine_similarity(-0.4 0.8 0.5 -0.2 0.3 , -0.3 0.2 0.3 -0.4 0.9) = 0.66
Jay
Person #2
cosine_similarity(-0.4 0.8 0.5 -0.2 0.3 , -0.5 -0.4 -0.2 0.7 -0.1) = -0.37

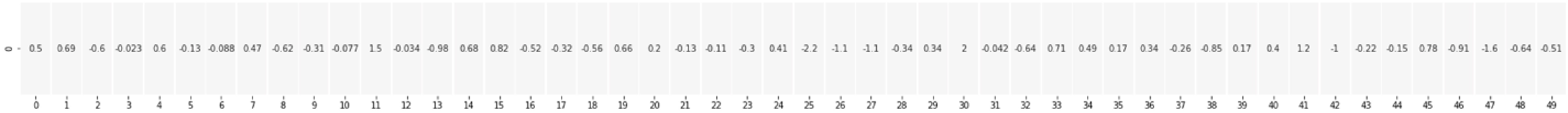
1- We can represent things (and people) as vectors of numbers
(Which is great for machines!)

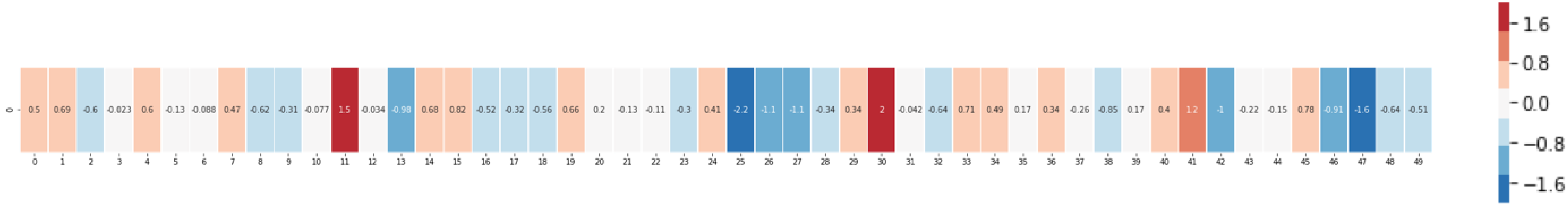
Jay	-0.4	0.8	0.5	-0.2	0.3

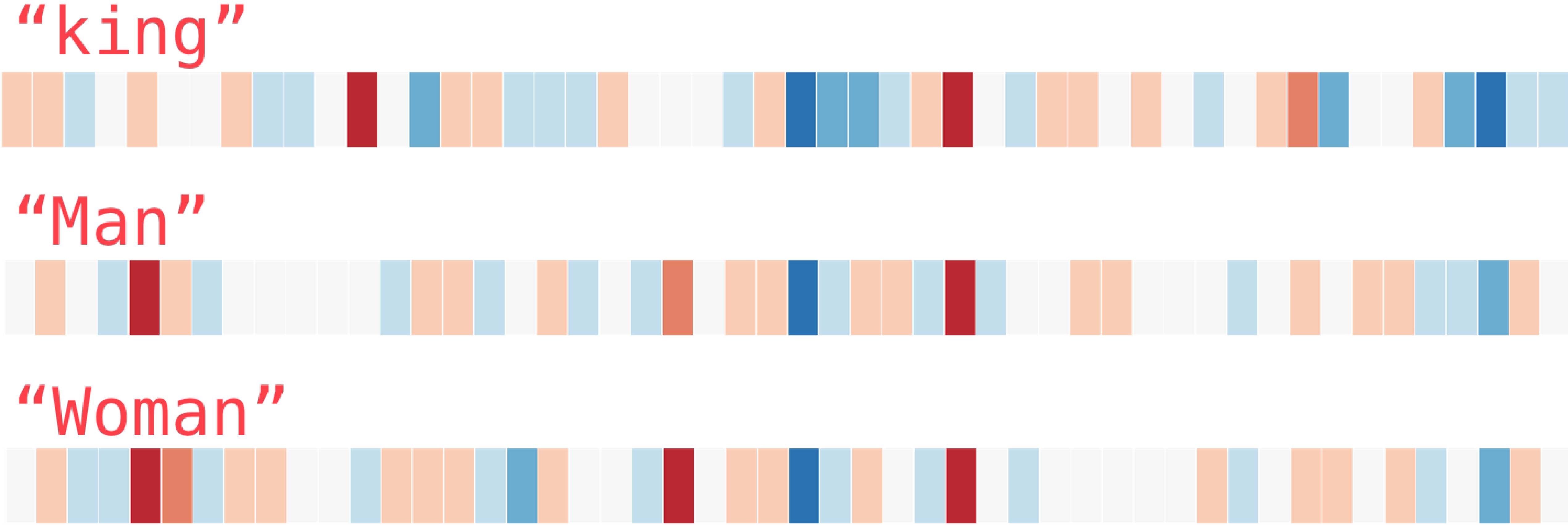
2- We can easily calculate how similar vectors are to each other

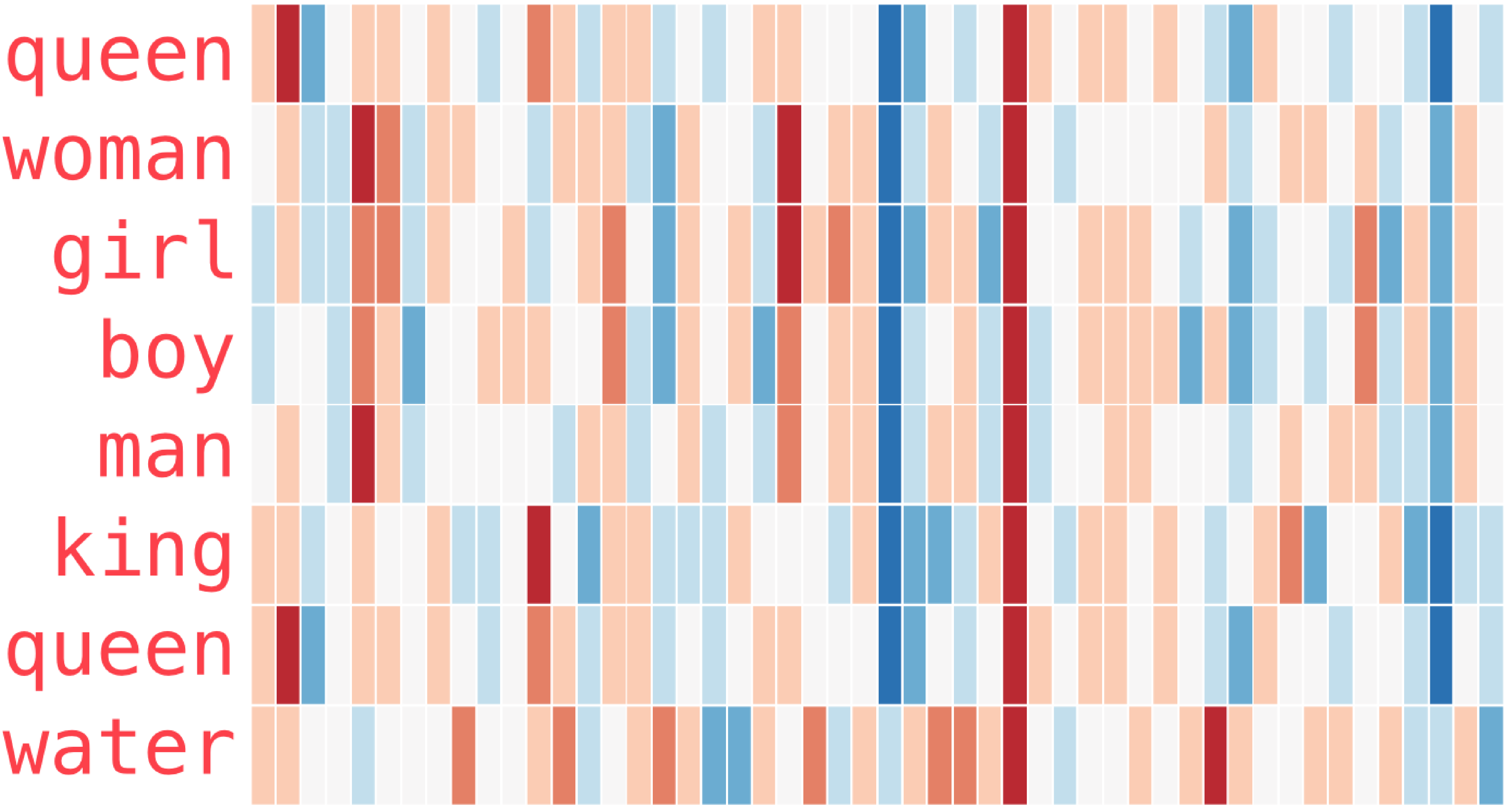
The people most similar to Jay are:

	cosine_similarity ▼
Person #1	0.86
Person #2	0.5
Person #3	-0.20

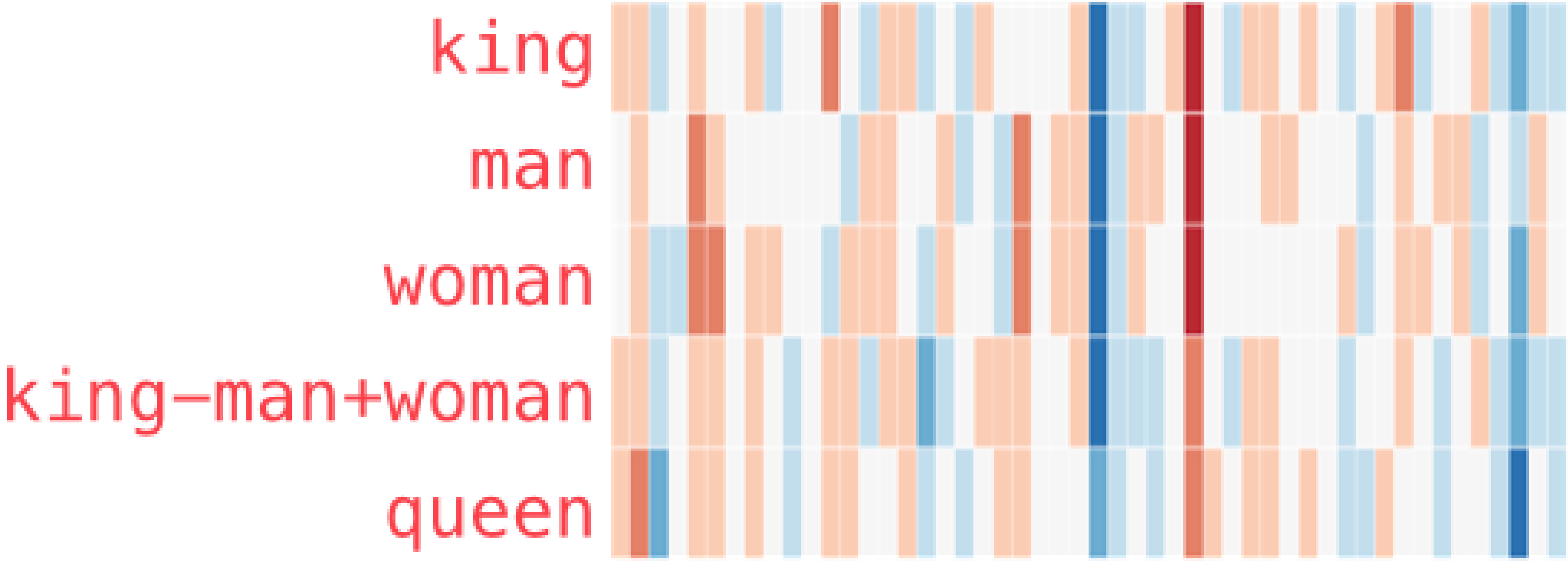

0
0.5 0.69 -0.6 -0.023 0.6 -0.13 -0.088 0.47 -0.62 -0.31 -0.077 1.5 -0.034 -0.98 0.68 0.82 -0.52 -0.32 -0.56 0.66 0.2 -0.13 -0.11 -0.3 0.41 -2.2 -1.1 -1.1 -0.34 0.34 2 -0.042 -0.64 0.71 0.49 0.17 0.34 -0.26 -0.85 0.17 0.4 1.2 -1 -0.22 -0.15 0.78 -0.91 -1.6 -0.64 -0.51
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

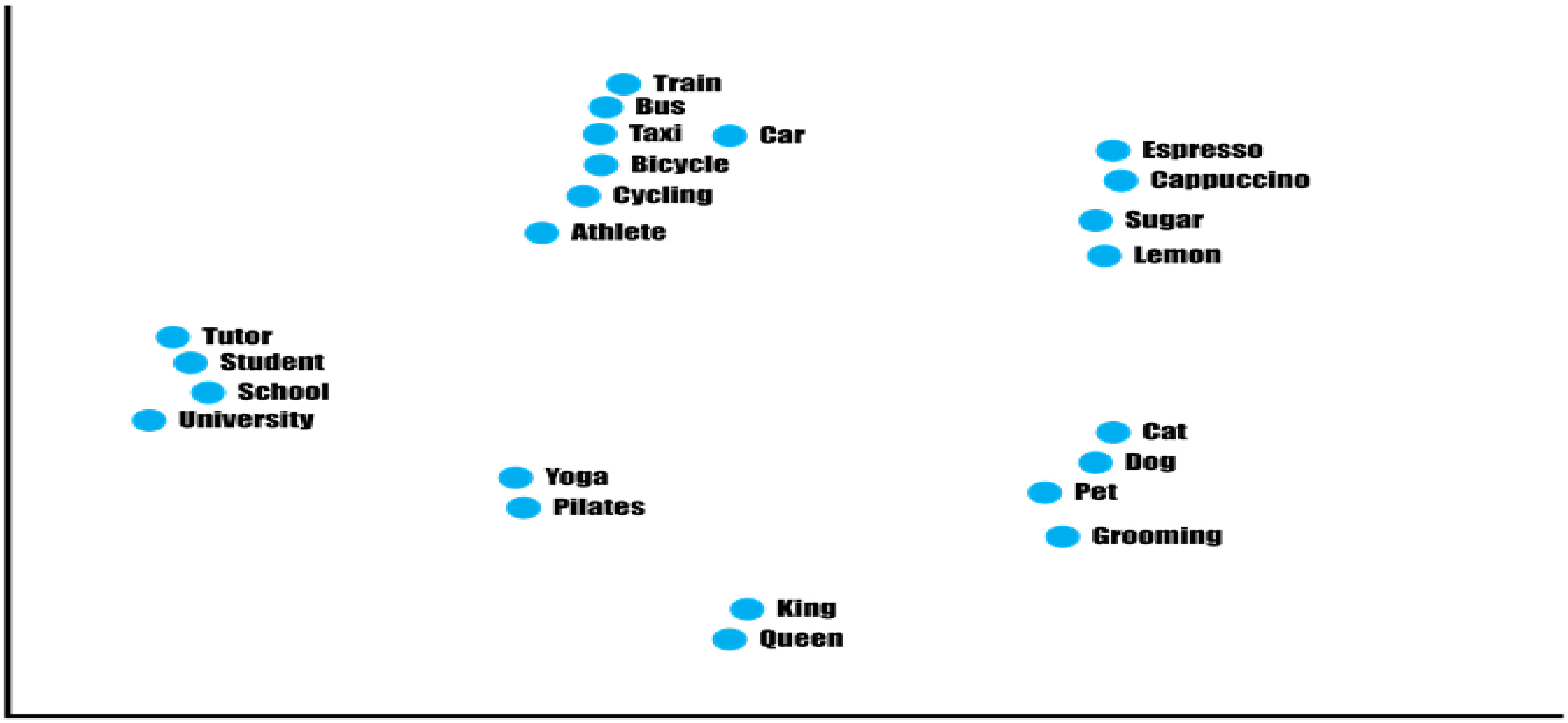

0
0.5 0.69 -0.6 -0.023 0.6 -0.13 -0.088 0.47 -0.62 -0.31 -0.077 1.5 -0.034 -0.98 0.68 0.82 -0.52 -0.32 -0.56 0.66 0.2 -0.13 -0.11 -0.3 0.41 -2.2 -1.1 -1.1 -0.34 0.34 2 -0.042 -0.64 0.71 0.49 0.17 0.34 -0.26 -0.85 0.17 0.4 1.2 -1 -0.22 -0.15 0.78 -0.91 -1.6 -0.64 -0.51
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
1.6
0.8
0.0
−0.8
−1.6

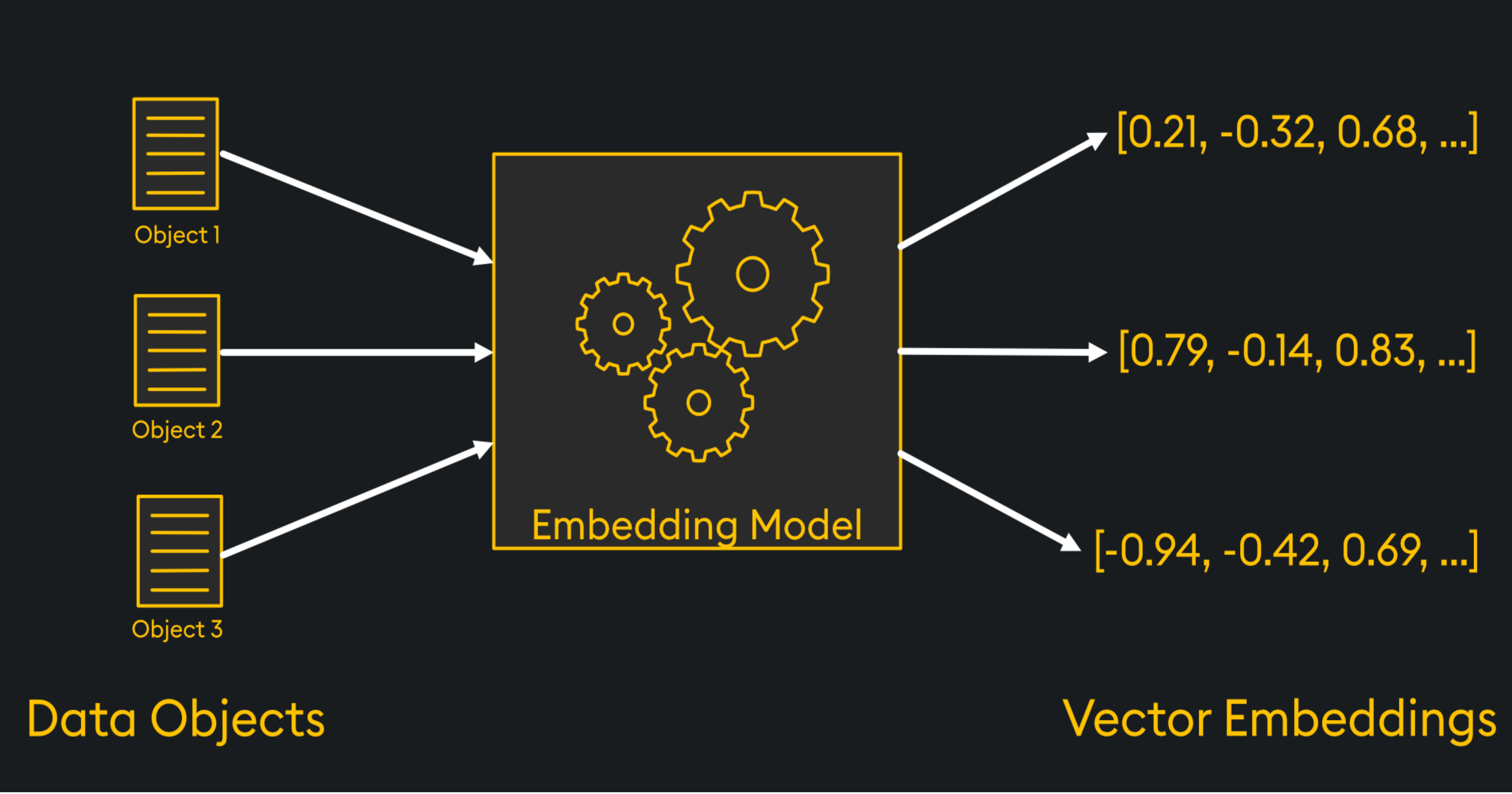

"king"
"Man"
"Woman"

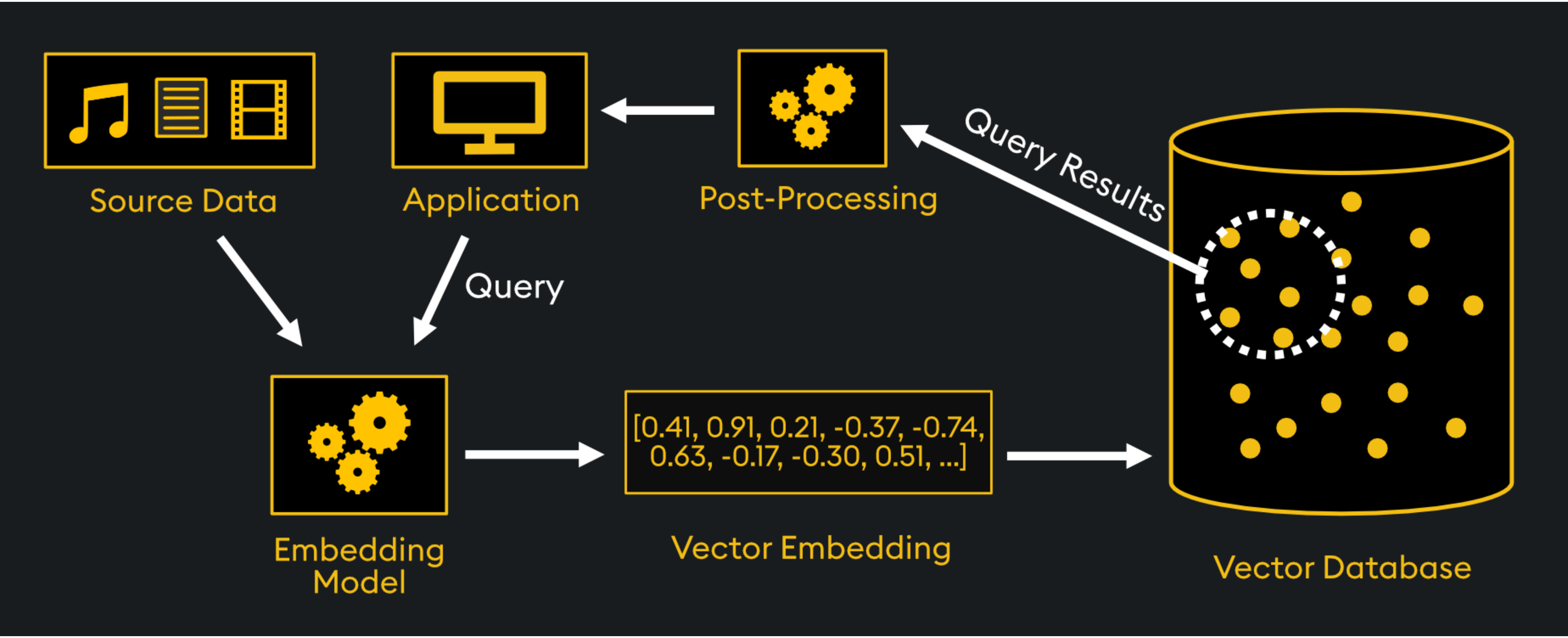

queen
woman
girl
boy
man
king
queen
water

king - man + woman ~= queen




Train
Bus
Taxi
Car
Bicycle
Cycling
Athlete
Espresso
Cappuccino
Sugar
Lemon
Tutor
Student
School
University
Yoga
Pilates
Cat
Dog
Pet
Grooming
King
Queen


Object 1
Object 2
Object 3
Embedding Model
[0.21, -0.32, 0.68, ...]
[0.79, -0.14, 0.83, ...]
[-0.94, -0.42, 0.69, ...]
Data Objects
Vector Embeddings


Source Data
Application
Post-Processing
Query Results
Query
Embedding
Model
[0.41, 0.91, 0.21, -0.37, -0.74,
0.63, -0.17, -0.30, 0.51, ...]
Vector Embedding
Vector Database