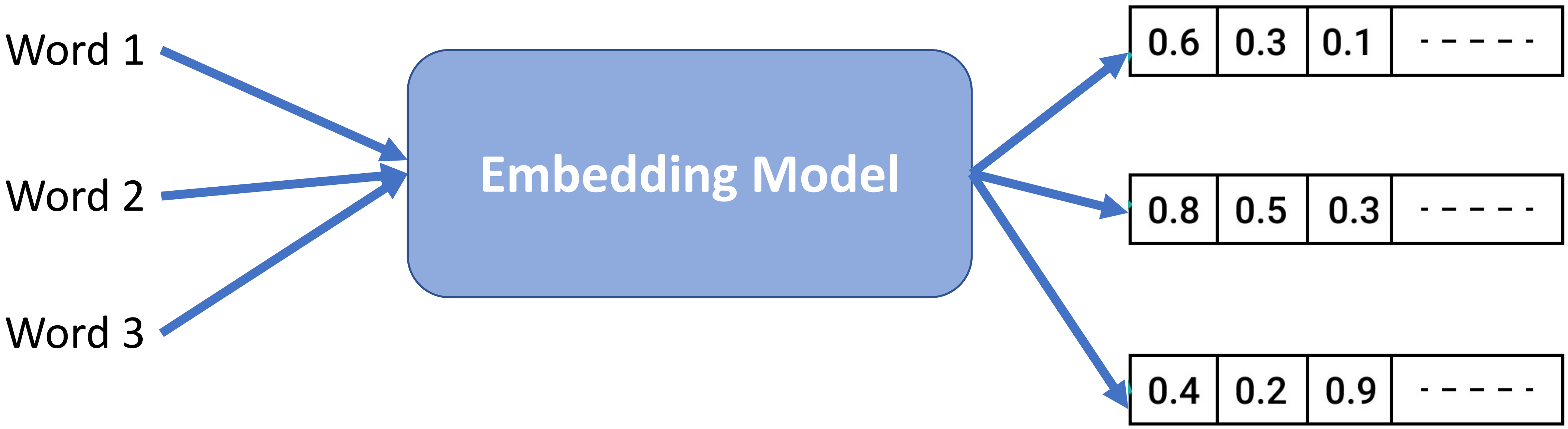
Topic 4

Word2vec

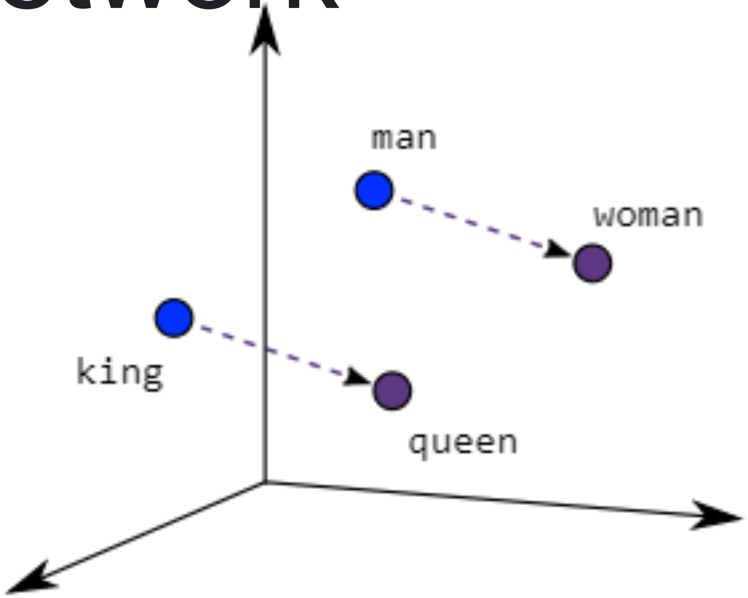
1. Use word2vec to find synonym
2. Replace the words with the most similar synonym

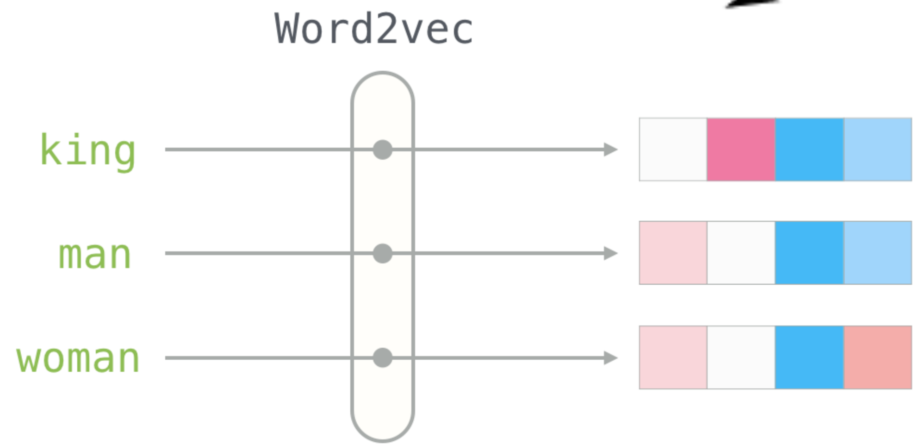
Our Goal



Word2Vec can be thought of as a shallow, two-layer neural network

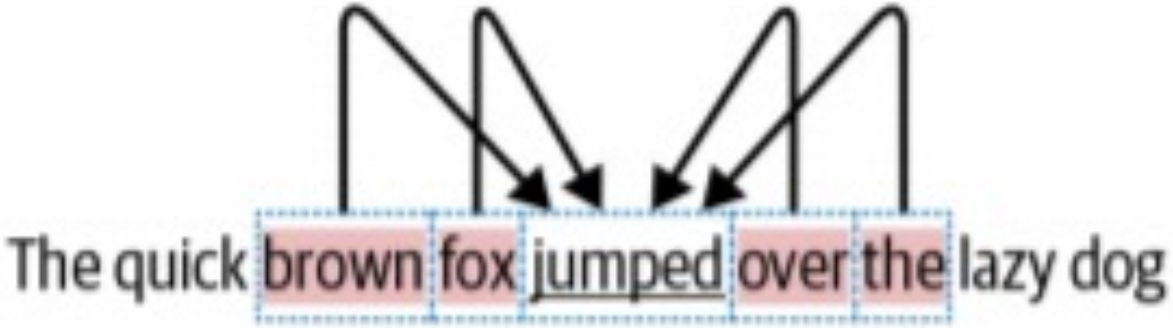
- Similar words will be closer together in vector space
- Unrelated words will be farther apart
- Mathematical relationships





“You shall know a word by the company it keeps” J.R. Firth

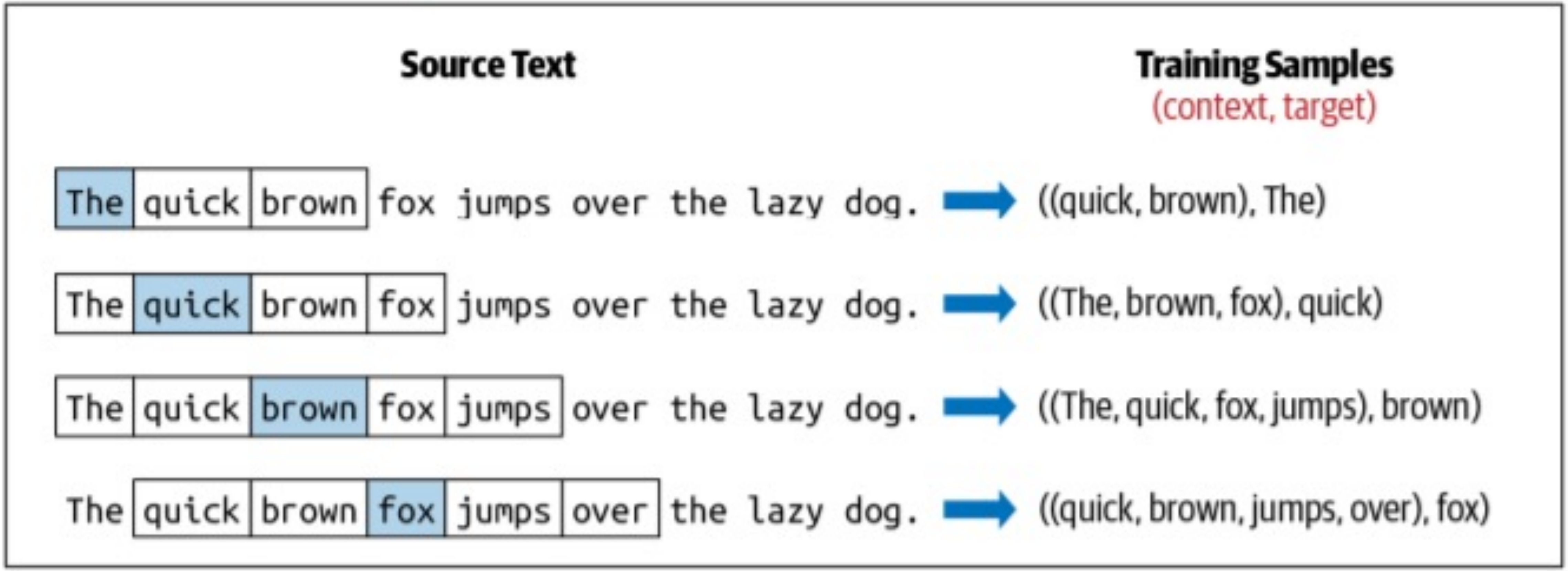
1. Continuous Bag of Words (CBOW)



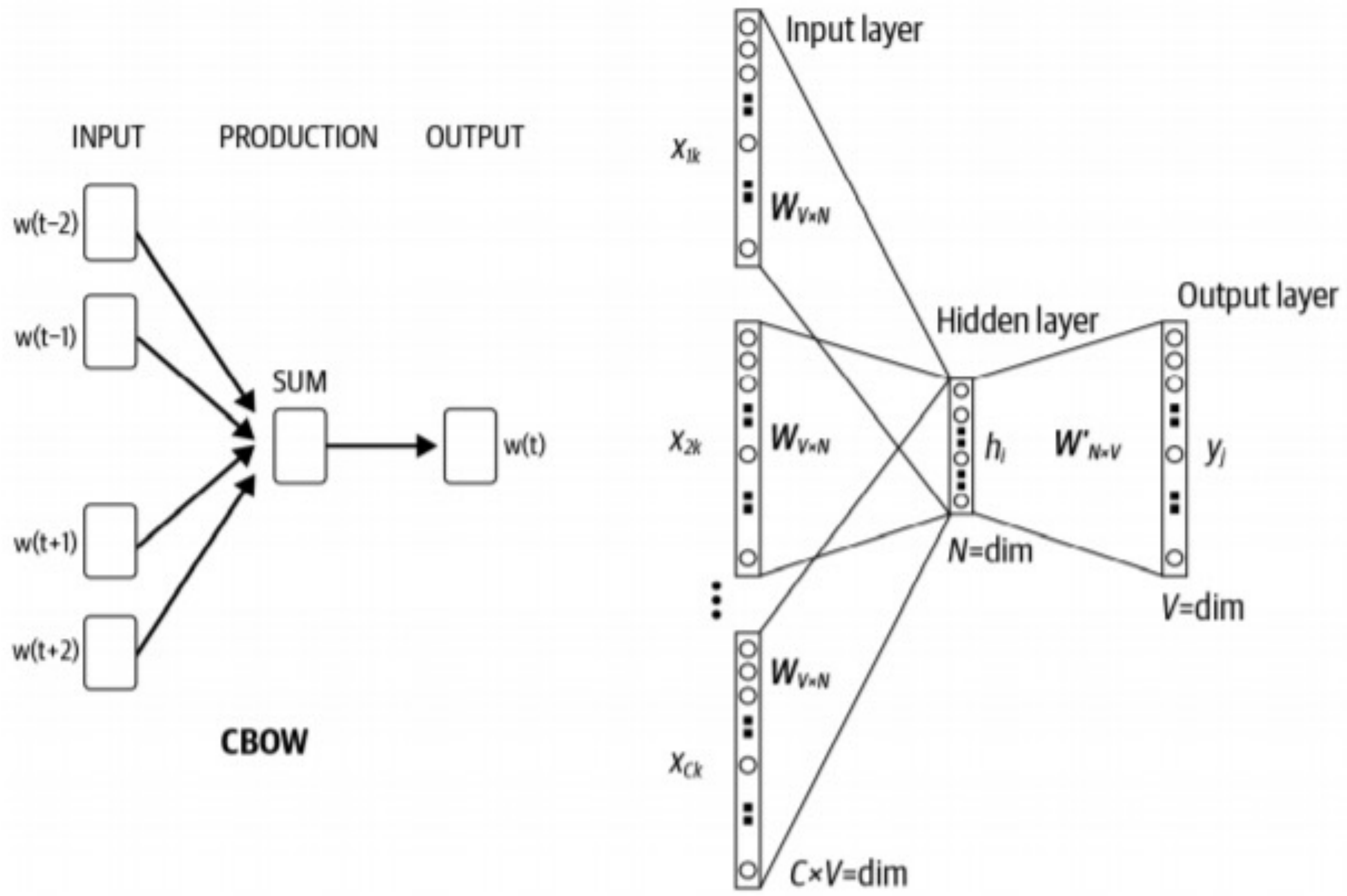
vs

2. Skip-gram

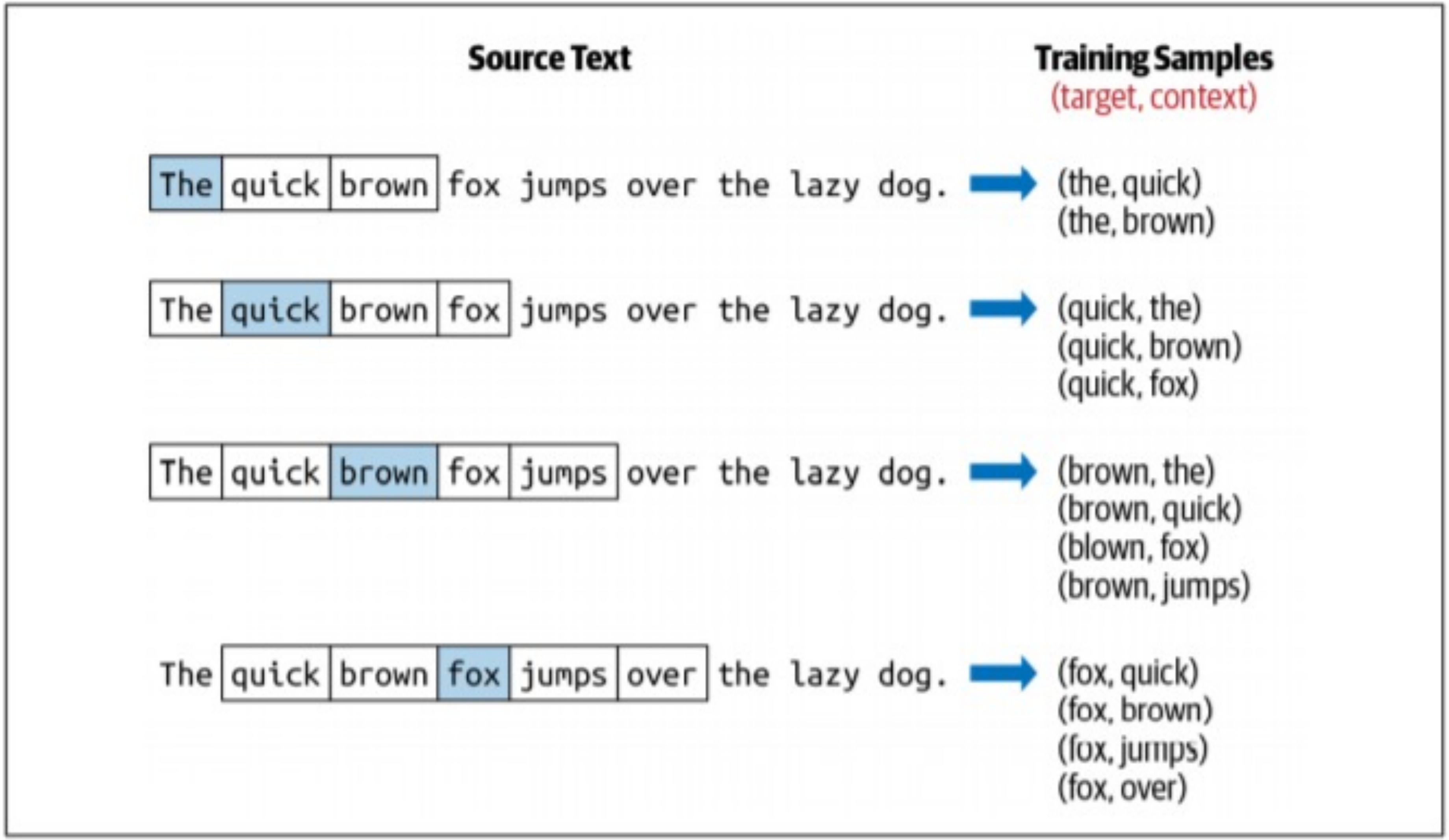
Preparing a training dataset for CBOW



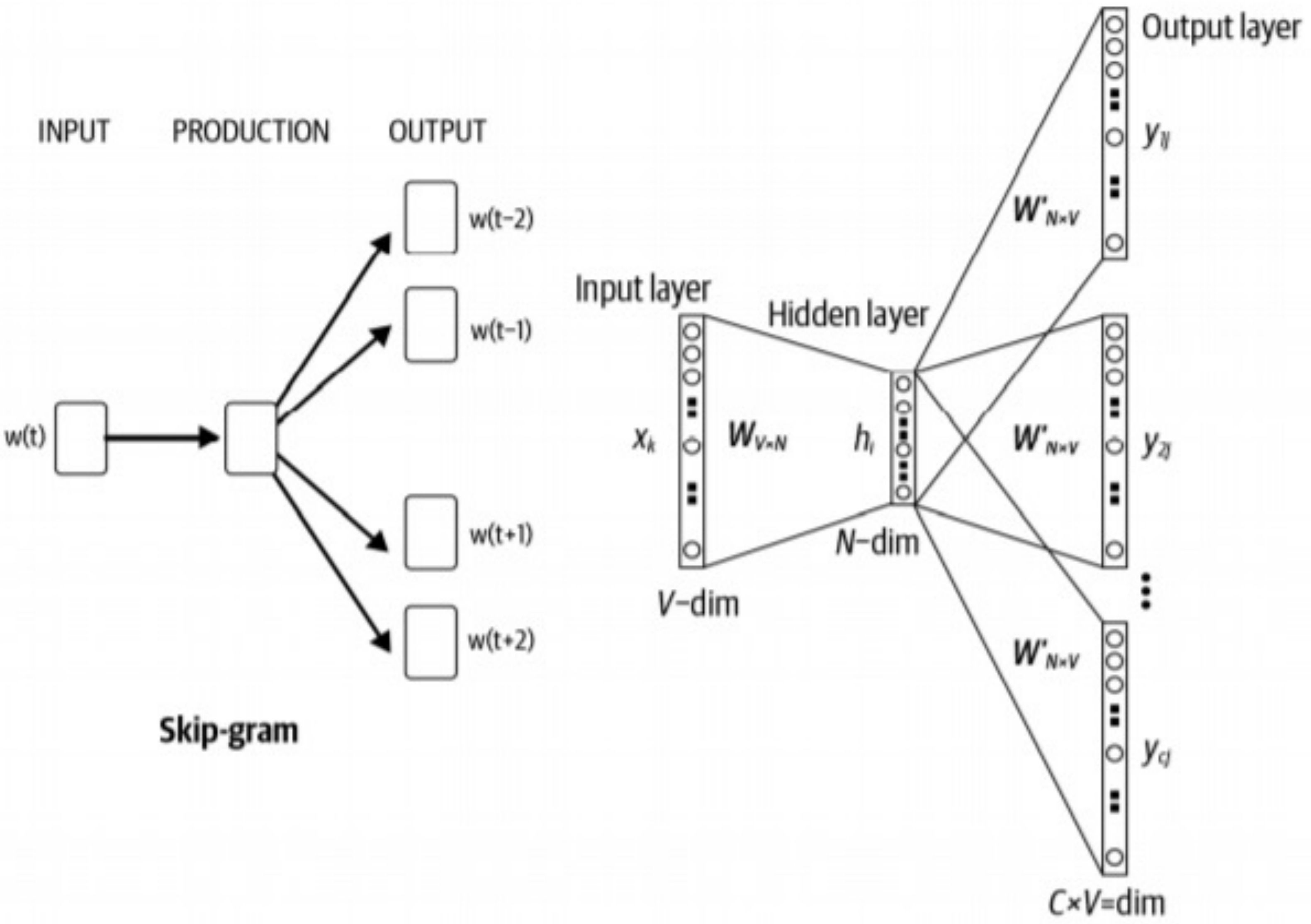
CBOW Model


INPUT
PRODUCTION
OUTPUT
w(t−2)
w(t−1)
w(t+1)
w(t+2)
SUM
w(t)
CBOW
Input layer
Hidden layer
Output layer
x_{1k}
x_{2k}
x_{Ck}
$W_{V\times N}$
$W_{V\times N}$
$W_{V\times N}$
h_i
$W'_{N\times V}$
y_j
N=dim
V=dim
$C\times V$=dim

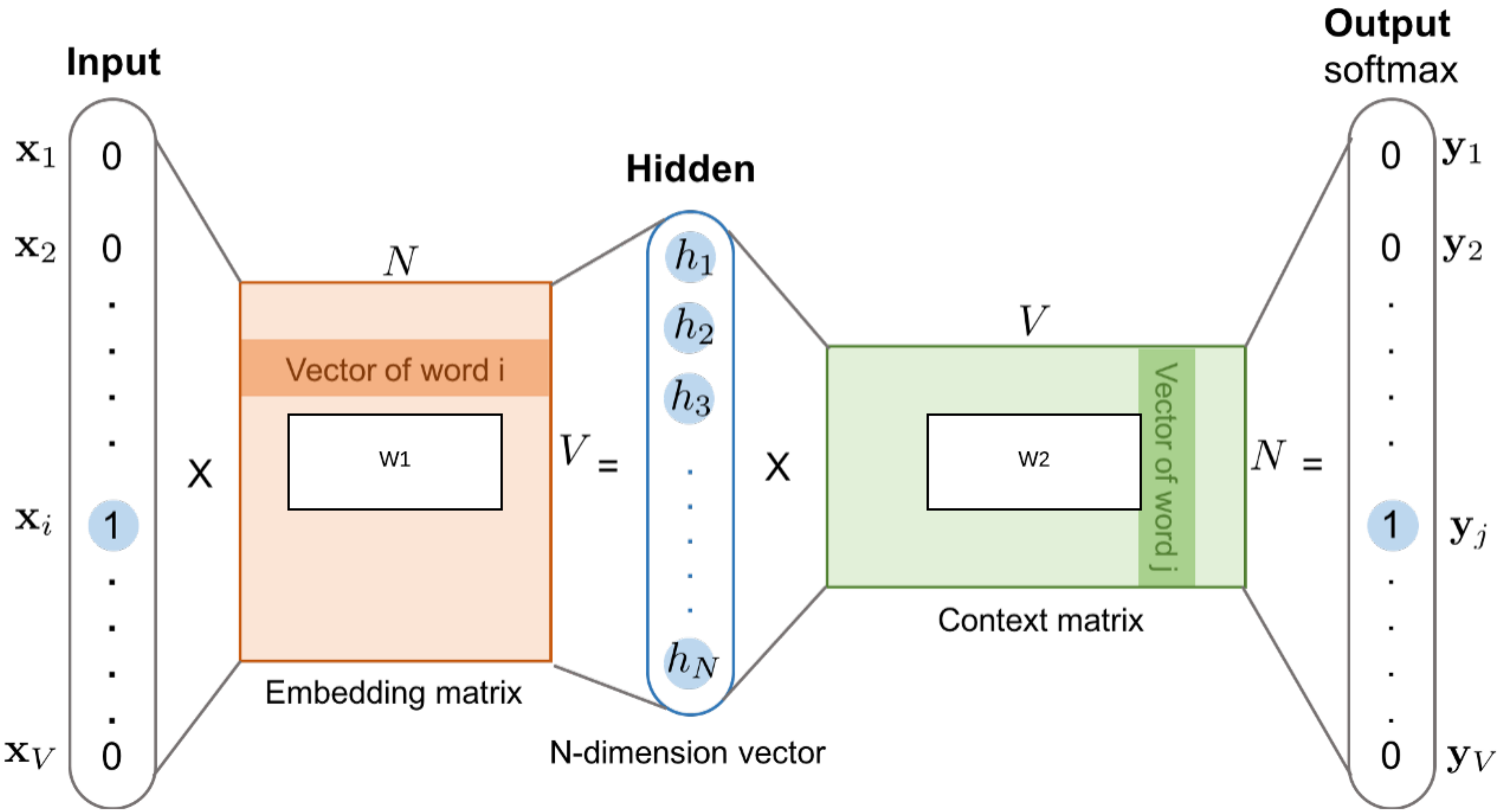
Preparing training data for Skip-gram



Continuous Skip-gram Model



Overview of the Word2Vec architecture



What does our input look like?

- A one hot encoding representing our input word

Input

quick

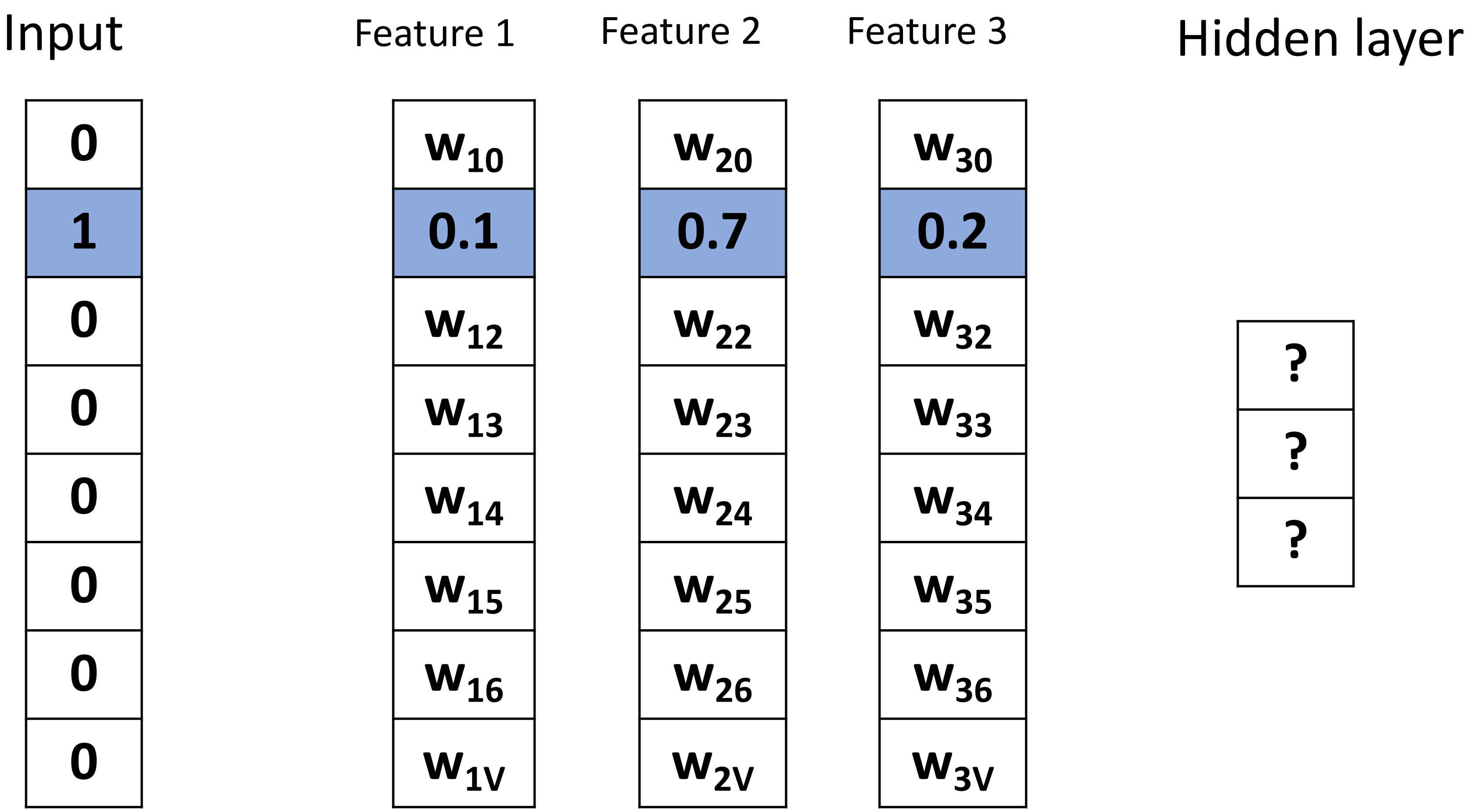
the	**0**
quick	**1**
brown	**0**
fox	**0**
jumped	**0**
over	**0**
lazy	**0**
dog	**0**

Training samples

(quick, the)

(quick, brown)
(quick, the)
(quick, brown)
(quick, fox)
.
.
.

Word embedding lookup



Input
(V-dim)

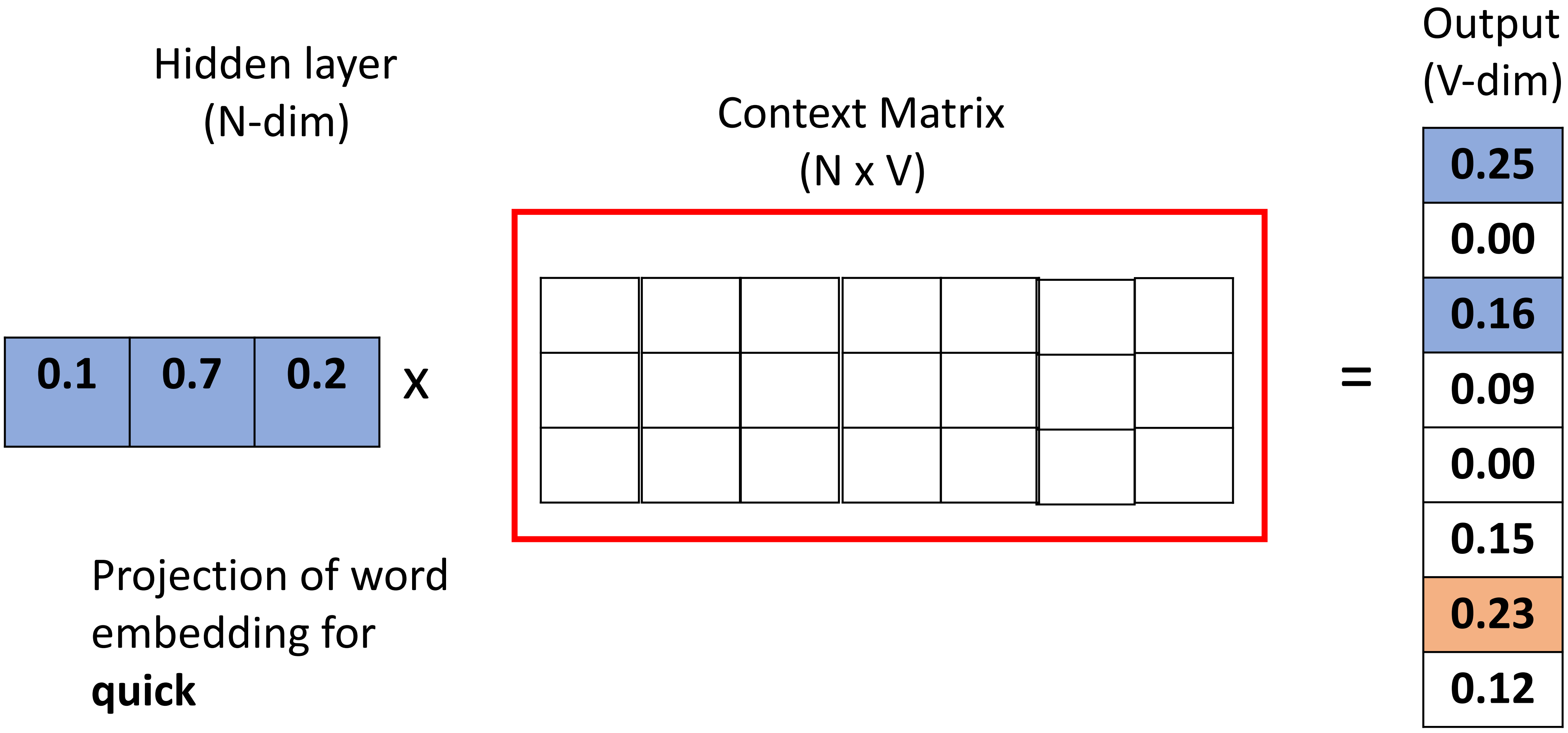
Hidden layer
(N-dim)

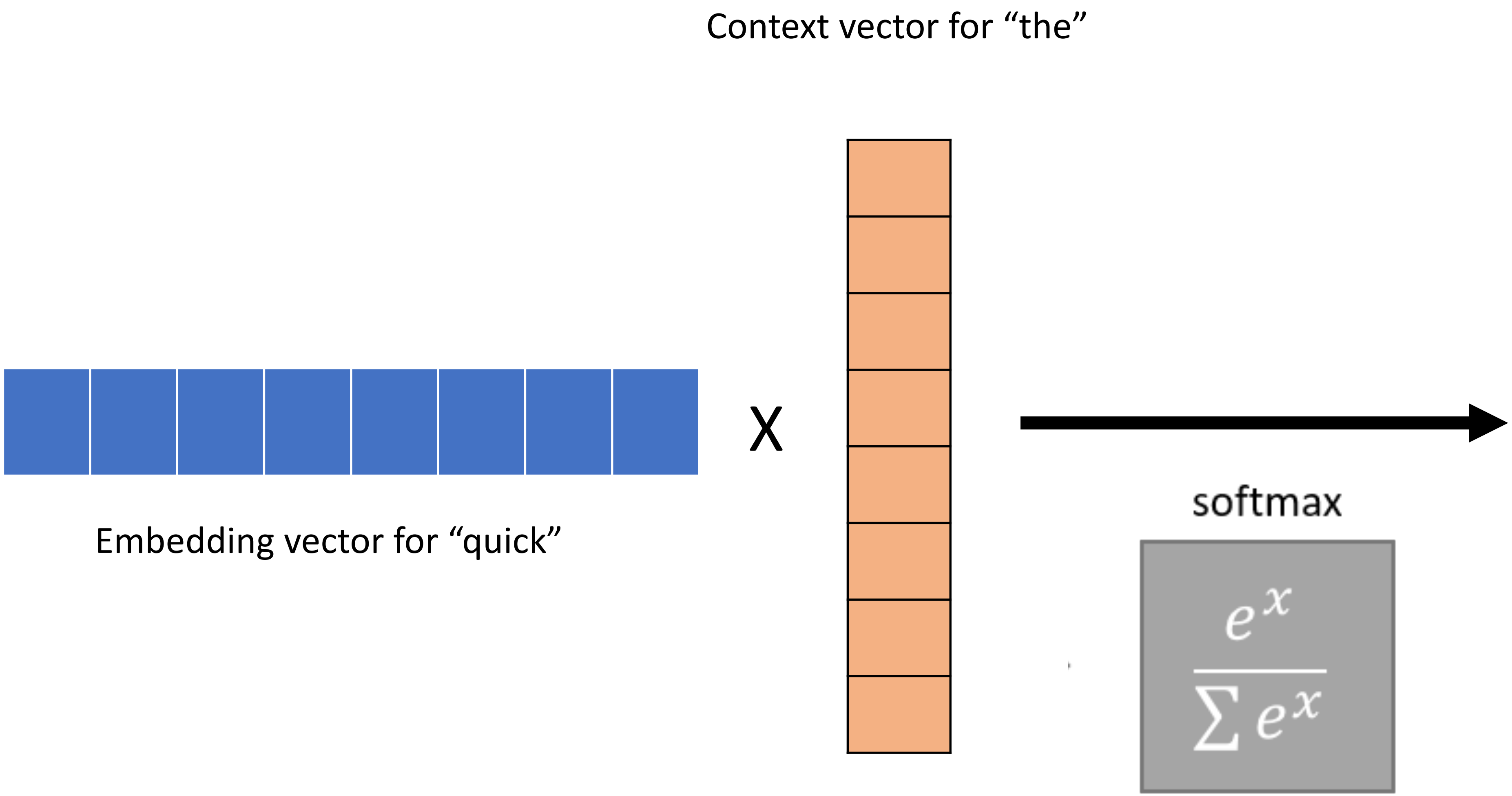
$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} w_{10} & w_{20} & w_{30} \\ 0.1 & 0.7 & 0.2 \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \\ w_{14} & w_{24} & w_{34} \\ w_{15} & w_{25} & w_{35} \\ w_{16} & w_{26} & w_{36} \\ w_{1V} & w_{2V} & w_{3V} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.2 \end{bmatrix}$$

Just a projection of the weights for word x_1!

(V x vector size)

Embedding matrix
(V x N)


Hidden layer
(N-dim)
0.1
0.7
0.2
x
Context Matrix
(N x V)
=
Output
(V-dim)
0.25
0.00
0.16
0.09
0.00
0.15
0.23
0.12
Projection of word embedding for **quick**



Probability that if a word is selected beside “quick”, it will be equal to “the”

Now we need a loss function

Output (V-dim)	What did we expect as output?	Error (predicted – true)
0.25	1	-0.75
0.00	0	0
0.16	0	0.16
0.09	0	0.09
0.00	0	0
0.15	0	0.15
0.23	0	0.23
0.12	0	0.12
	the	

Now we need a loss function

Output (V-dim)	What did we expect as output?	Error (predicted – true)
0.25	**0**	**0.25**
0.00	**0**	**0**
0.16	**1**	**-0.74**
0.09	**0**	**0.09**
0.00	**0**	**0**
0.15	**0**	**0.15**
0.23	**0**	**0.23**
0.12	**0**	**0.12**

brown

At the end of the day, the goal is to learn the **embedding matrix**

- Refine the values in this matrix with your training data
- We actually have no use for our output layer
  - throw it away because the task we have trained the model to do is not useful for us

Relevant Hyperparameters for making our training dataset

- We do have some decisions to make if we want to use word2vec or train one ourselves

- **window**: How many context words are we going to consider when building our training dataset?
- **size:** The number of "features" we want to represent each word in our corpus (N)
- **min_count**: We will disregard the vocabulary that has counts less than this number

We can simplify Word2Vec even further

- Performing the weight and bias term updates is computationally expensive if you are going to update everything in every round of the training process
- Solution: Negative sampling

Skipgram

shalt	not	make	a	machine

input	output
make	shalt
make	not
make	a
make	machine

Negative Sampling

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0

More Visual Explanation of Word2Vec

- Reference: https://jalammar.github.io/illustrated-word2vec/
- Gensim documentation for Word2Vec: https://radimrehurek.com/gensim/models/word2vec.html

Is Word2Vec our only option?

- Doc2Vec
- FastText
- GloVe

GloVe

- Different mechanism and equations to create the embedding matrix
- GloVe based on matrix factorization of the global co-occurrence matrix
- Both Word2Vec and GloVe give similar results

Contextual Embeddings

- One key limitation of tradition embedding representations such as Word2Vec is the problem of **word sense disambiguation**
  - Every possible meaning of a word is encoded into the same embedding.
- Eg. Word 'play' in two different sentences have quite different meaning:
  - I went to a play at the theatre.
  - John wants to play with his friends.

Notebook Exercises

- Access the Jupyter notebook on Word2vec

Next time:

- What can we do with these representations when we have them?
- Some of the methods of visualization we can use to explore different text representations