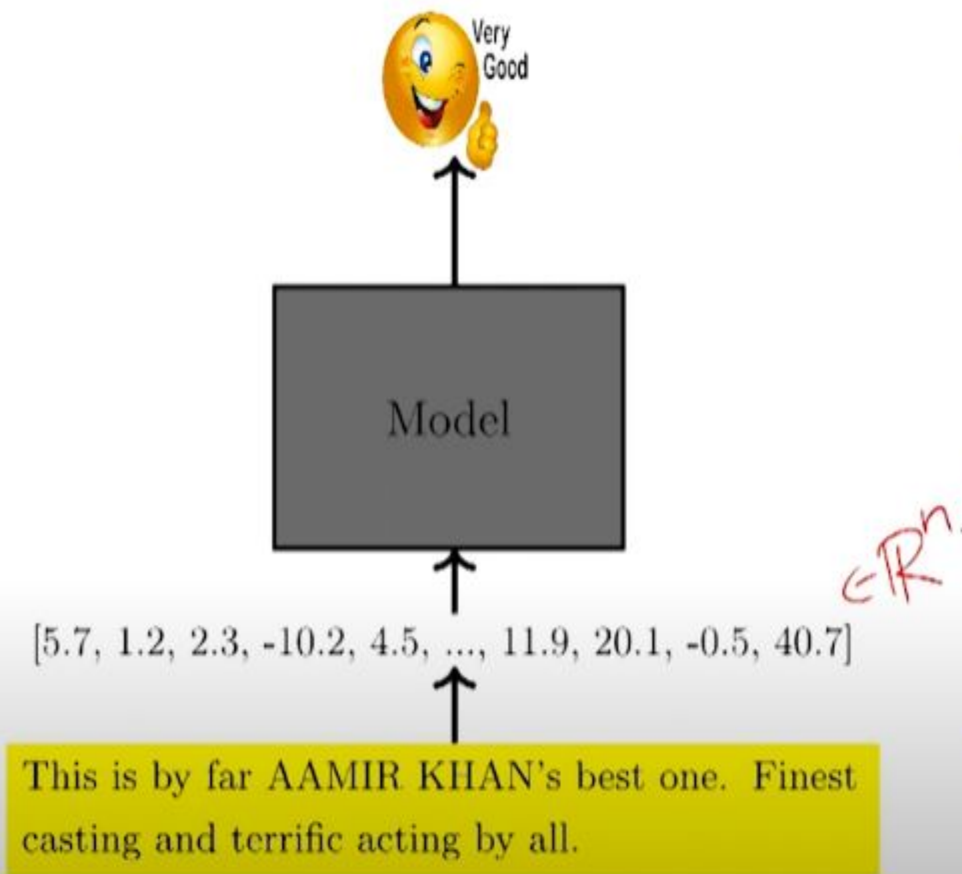
Word to Vector Representation



- Let us start with a very simple motivation for why we are interested in vectorial representations of words
- Suppose we are given an input stream of words (sentence, document, etc.) and we are interested in learning some function of it (say, $\hat{y} = sentiments(words)$)
- Say, we employ a machine learning algorithm (some mathematical model) for learning such a function ($\hat{y} = f(\mathbf{x})$)
- We first need a way of converting the input stream (or each word in the stream) to a vector $\mathbf{x}$ (a mathematical quantity)

Corpus:

- Human machine interface for computer applications
- User opinion of computer system response time
- User interface management system
- System engineering for improved response time

V = [human,machine, interface, for, computer, applications, user, opinion, of, system, response, time, interface, management, engineering, improved]

- Given a corpus, consider the set V of all unique words across all input streams (*i.e.*, all sentences or documents)
- V is called the **vocabulary** of the corpus (*i.e.*, all sentences or documents)
- We need a representation for every word in V
- One very simple way of doing this is to use one-hot vectors of size $|V|$

cat:	0	0	0	0	0	1	0
dog:	0	1	0	0	0	0	0
truck:	0	0	0	1	0	0	0

$$euclid\_dist(\mathbf{cat}, \mathbf{dog}) = \sqrt{2}$$
$$euclid\_dist(\mathbf{dog}, \mathbf{truck}) = \sqrt{2}$$
$$cosine\_sim(\mathbf{cat}, \mathbf{dog}) = 0$$
$$cosine\_sim(\mathbf{dog}, \mathbf{truck}) = 0$$

Problems:

- V tends to be very large (for example, 50K for PTB, 13M for Google 1T corpus)
- These representations do not capture any notion of similarity
- Ideally, we would want the representations of cat and dog (both domestic animals) to be closer to each other than the representations of cat and truck
- However, with 1-hot representations, the Euclidean distance between **any two words** in the vocabulary in $\sqrt{2}$

  And the cosine similarity between **any two words** in the vocabulary is 0

A bank is a **financial** institution that accepts **deposits** from the public and creates **credit**.

The idea is to use the accompanying words (financial, deposits, credit) to represent bank

- *You shall know a word by the company it keeps - Firth, J. R. 1957:11*
- Distributional similarity based representations
- This leads us to the idea of co-occurrence matrix

Corpus:

- Human machine interface for computer applications
- User opinion of computer system response time
- User interface management system
- System engineering for improved response time

- A co-occurrence matrix is a **terms** $\times$ **terms** matrix which captures the number of times a term appears in the context of another term

Corpus:

- Human machine interface for computer applications
- User opinion of computer system response time
- User interface management system
- System engineering for improved response time

	human	machine	system	for	....	user
human	0	1	0	1	....	0
machine	1	0	0	1	....	0
system	0	0	0	1	....	2
for	1	1	1	0	....	0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
user	0	0	2	0	....	0

Co-occurence Matrix

- A co-occurrence matrix is a **terms** × **terms** matrix which captures the number of times a term appears in the context of another term
- The context is defined as a window of k words around the terms
- Let us build a co-occurrence matrix for this toy corpus with $k = 2$
- This is also known as a **word** × **context** matrix
- You could choose the set of **words** and **contexts** to be same or different
- Each row (column) of the co-occurrence matrix gives a vectorial representation of the corresponding word (context)

	human	machine	system	for	...	user
human	0	1	0	x	...	0
machine	1	0	0	x	...	0
system	0	0	0	x	...	2
for	x	x	x	x	...	x
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
user	0	0	2	x	...	0

Some (fixable) problems

- Stop words (a, the, for, etc.) are very frequent → these counts will be very high
- Solution 1: Ignore very frequent words
- Solution 2: Use a threshold t (say, t = 100)

$$X_{ij} = min(count(w_i, c_j), t),$$

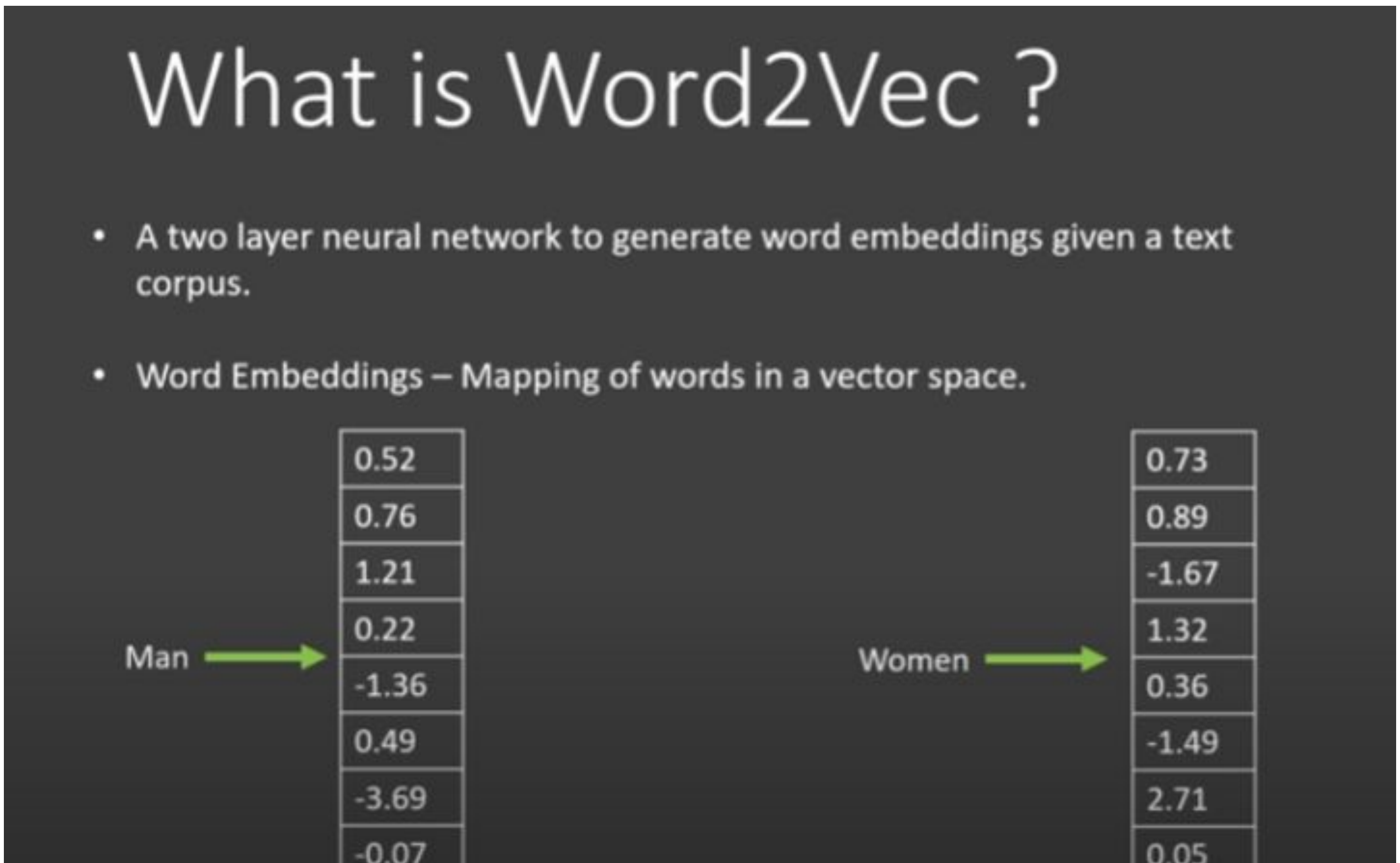
where w is word and c is context.

Some (severe) problems

- Very high dimensional ($|V|$)
- Very sparse
- Grows with the size of the vocabulary
- **Solution:** Use dimensionality reduction (SVD)

Continuous Bag of Words (CBOW)

- The methods that we have seen so far are called **count based models** because they use the co-occurrence counts of words
- We will now see methods which directly **learn** word representations (these are called **(direct) prediction based models**)



Why Word2vec?

- Preserves relationship between words.
- Deals with addition of new words in the vocabulary.
- Better results in lots of deep learning applications.

Working of word2Vec

- The word2vec objective function causes the words that occur in similar contexts to have similar embeddings.

Example: The **kid** said he would grow up to be superman.

The **child** said he would grow up to be superman.

CBOW

- Predict the target word from the context.

Skip Gram

- Predict the context words from target.

1. Take a fake problem
2. Solve it using neural network
3. You get word embeddings as a side effect

fake problem: fill in a missing word in a sentence

There lived a king called Ashoka in India. After Kalinga battle, he converted to Buddhism. This mighty king ordered his ministers to put together a peaceful treaty with their neighboring kingdoms. The emperor ordered his ministers to also build stupa, a monument with Buddha's teachings.

Fake problem

______ ordered his ministers

________ ordered his ministers

There lived a king called Ashoka in India. After Kalinga battle, he converted to Buddhism. This mighty king ordered his ministers to put together a peaceful treaty with their neighboring kingdoms. The emperor ordered his ministers to also build stupa, a monument with Buddha's teachings.

Fake problem

king ordered his ministers

emperor ordered his ministers

Side effect

king $\begin{bmatrix} 0.7 \\ 0.4 \\ 1.2 \\ 3.8 \end{bmatrix}$ emperor $\begin{bmatrix} 0.7 \\ 0.5 \\ 1.2 \\ 3.8 \end{bmatrix}$

king ~ emperor

eating ____ is very healthy

table, angry, truck, apple, pizza, walnut

NASA launched ____ last month

table, angry, truck, rocket, apple, pizza

There lived a king called Ashoka in India. After Kalinga battle, he converted to Buddhism. This mighty king ordered his ministers to put together a peaceful treaty with their neighboring kingdoms. The emperor ordered his ministers to also build stupa, a monument with Buddha's teachings.

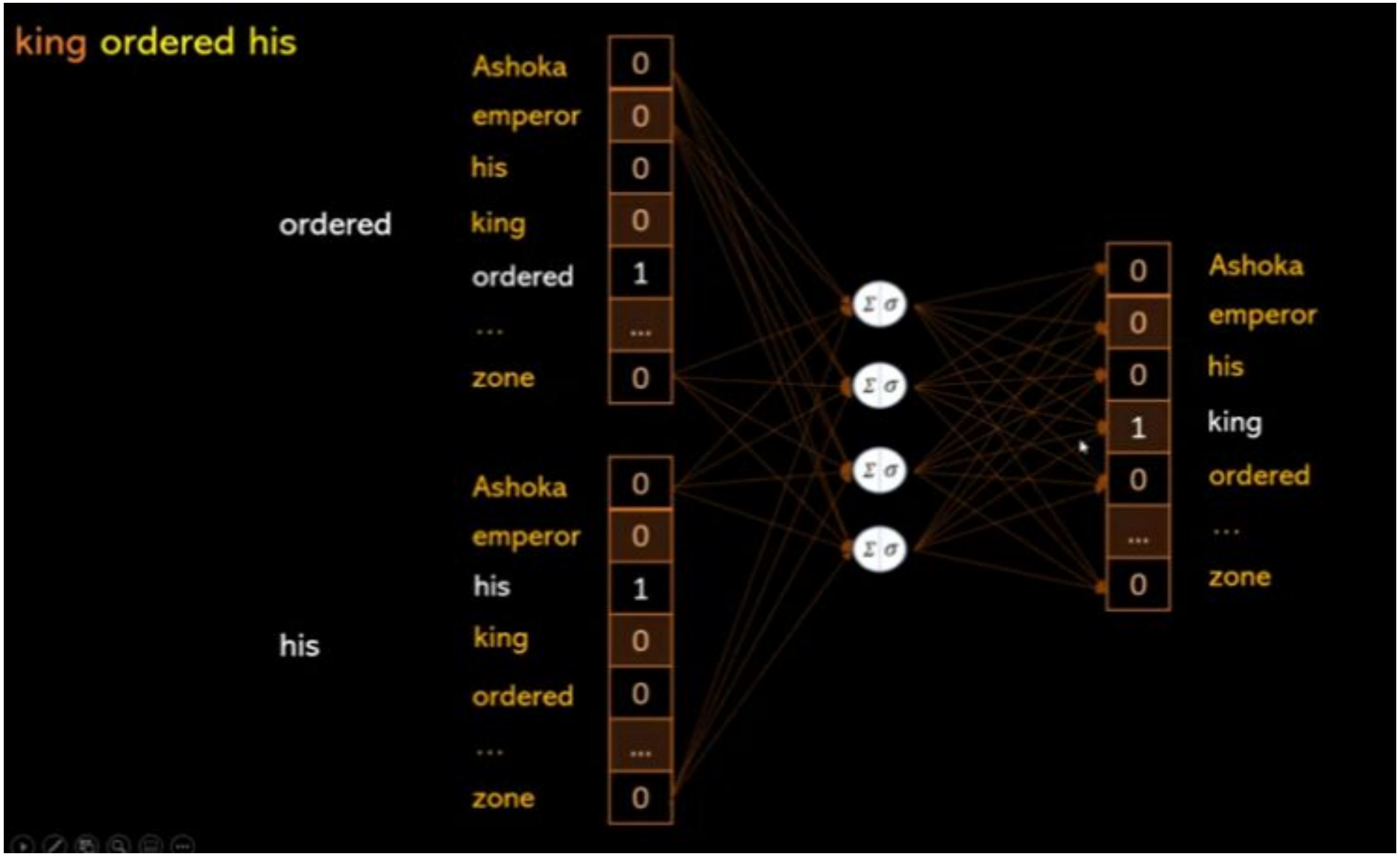
Training samples

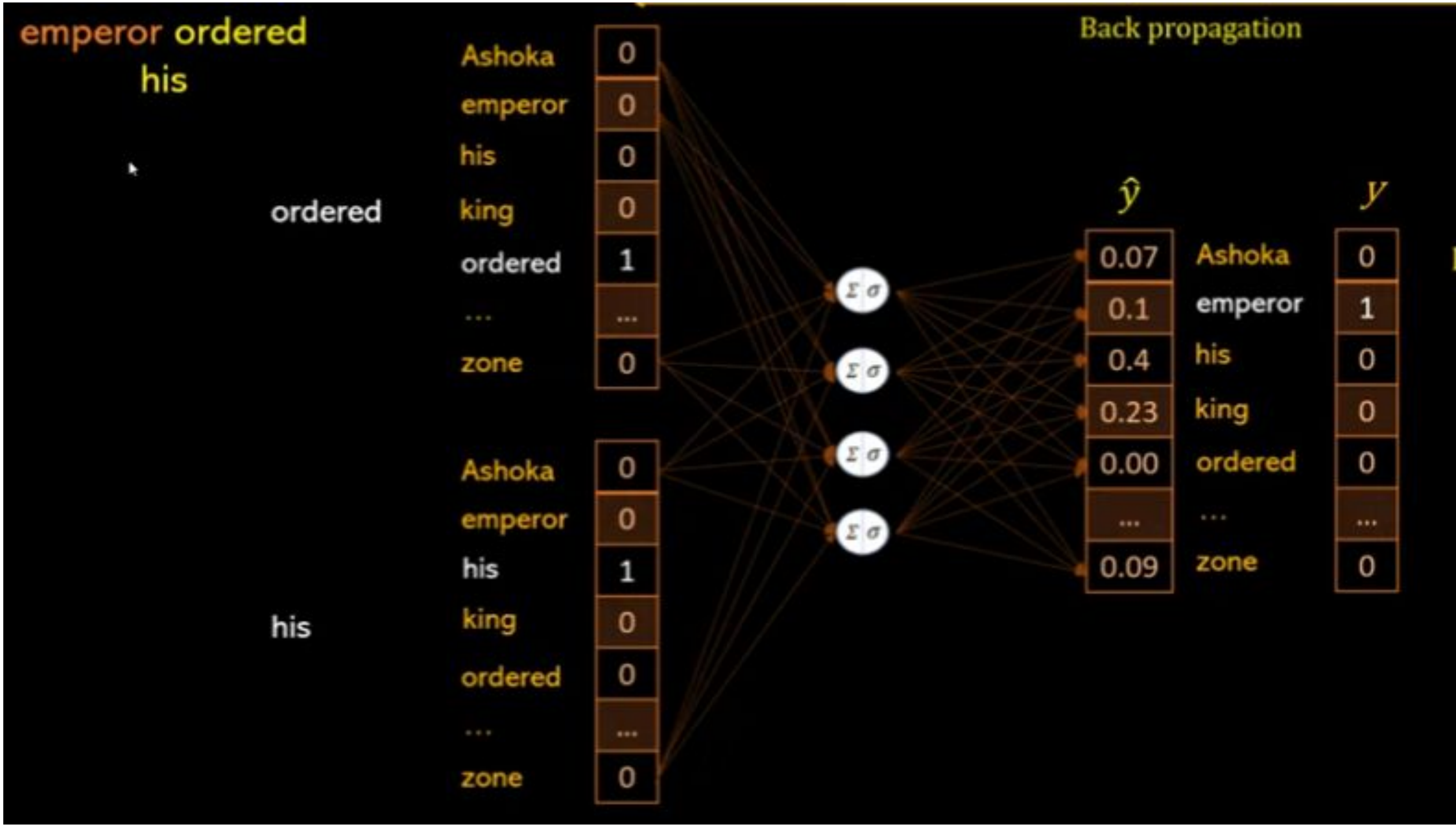
lived, a → There

a, king → lived

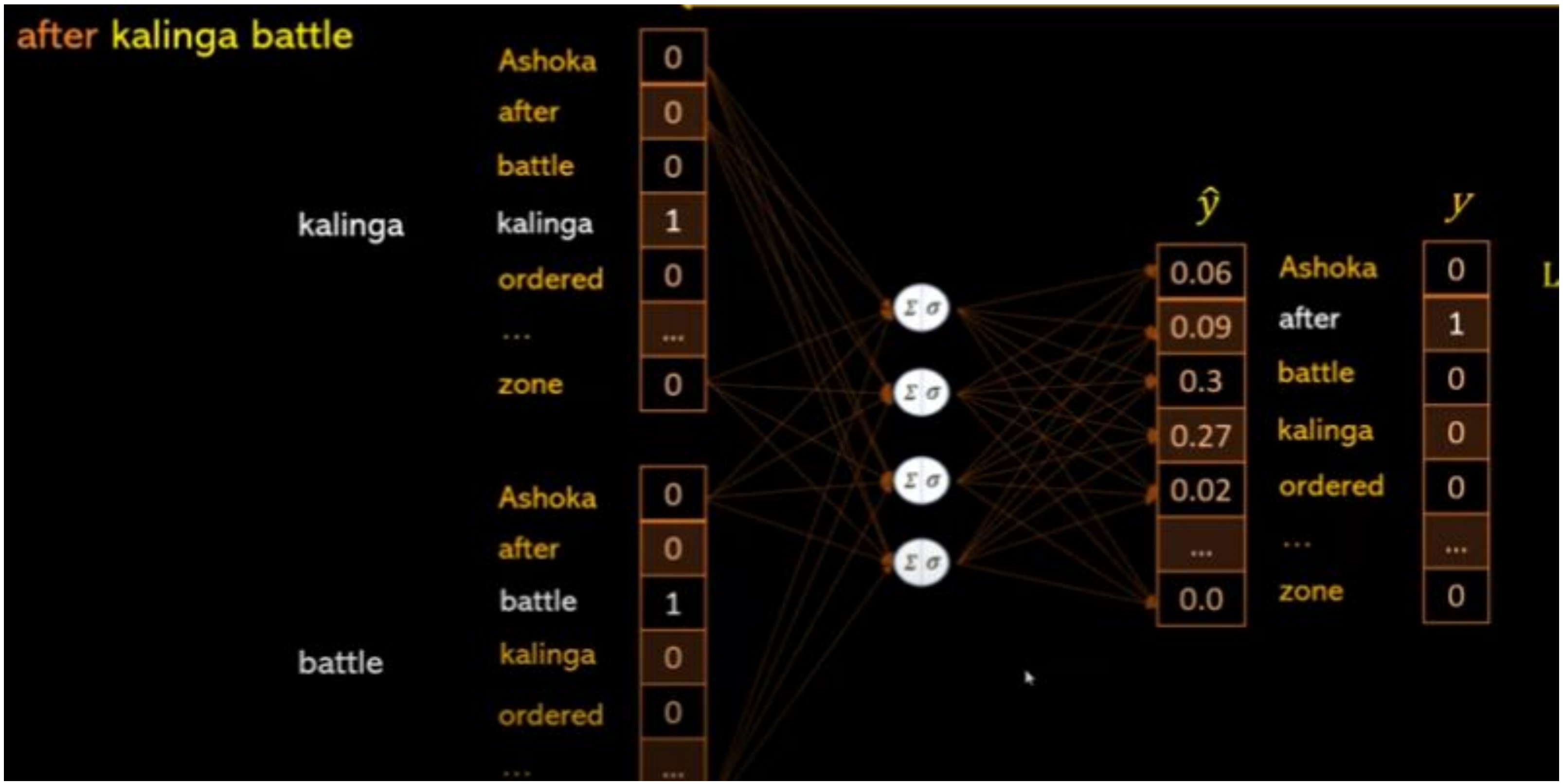
ordered, his → king

ordered, his → emperor


king ordered his
ordered
Ashoka
emperor
his
king
ordered
...
zone
0
0
0
0
1
...
0
his
Ashoka
emperor
his
king
ordered
...
zone
0
0
1
0
0
...
0
Σ σ
Σ σ
Σ σ
Σ σ
0
0
0
1
0
...
0
Ashoka
emperor
his
king
ordered
...
zone

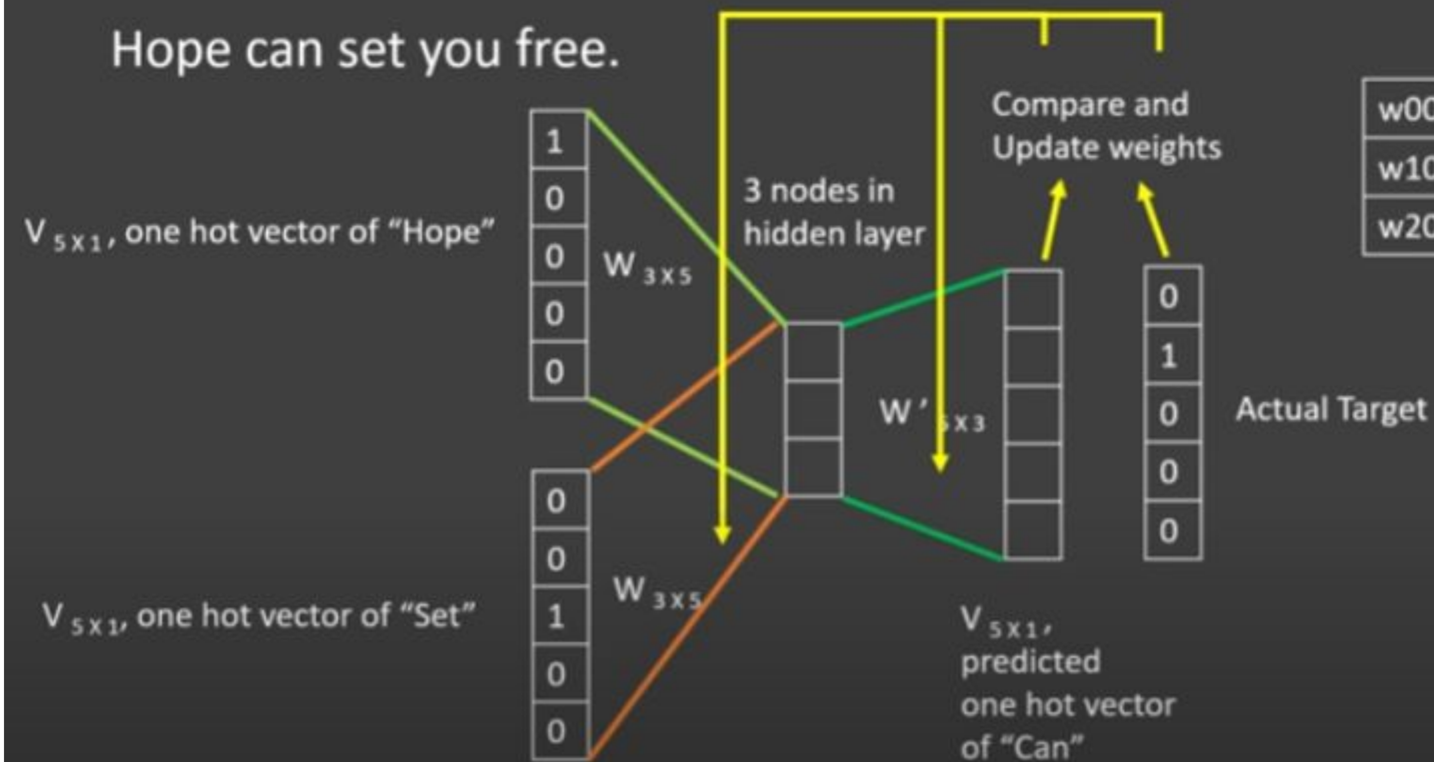


emperor ordered
his
Back propagation
ordered
his
Ashoka 0
emperor 0
his 0
king 0
ordered 1
... ...
zone 0
Ashoka 0
emperor 0
his 1
king 0
ordered 0
... ...
zone 0
$\hat{y}$
y
0.07 Ashoka 0
0.1 emperor 1
0.4 his 0
0.23 king 0
0.00 ordered 0
... ...
0.09 zone 0


after kalinga battle
kalinga
battle
Ashoka
after
battle
kalinga
ordered
...
zone
0
0
0
1
0
...
0
Ashoka
after
battle
kalinga
ordered
...
0
0
1
0
0
...
Σ σ
Σ σ
Σ σ
Σ σ
ŷ
0.06
0.09
0.3
0.27
0.02
...
0.0
Ashoka
after
battle
kalinga
ordered
...
zone
y
0
1
0
0
0
...
0

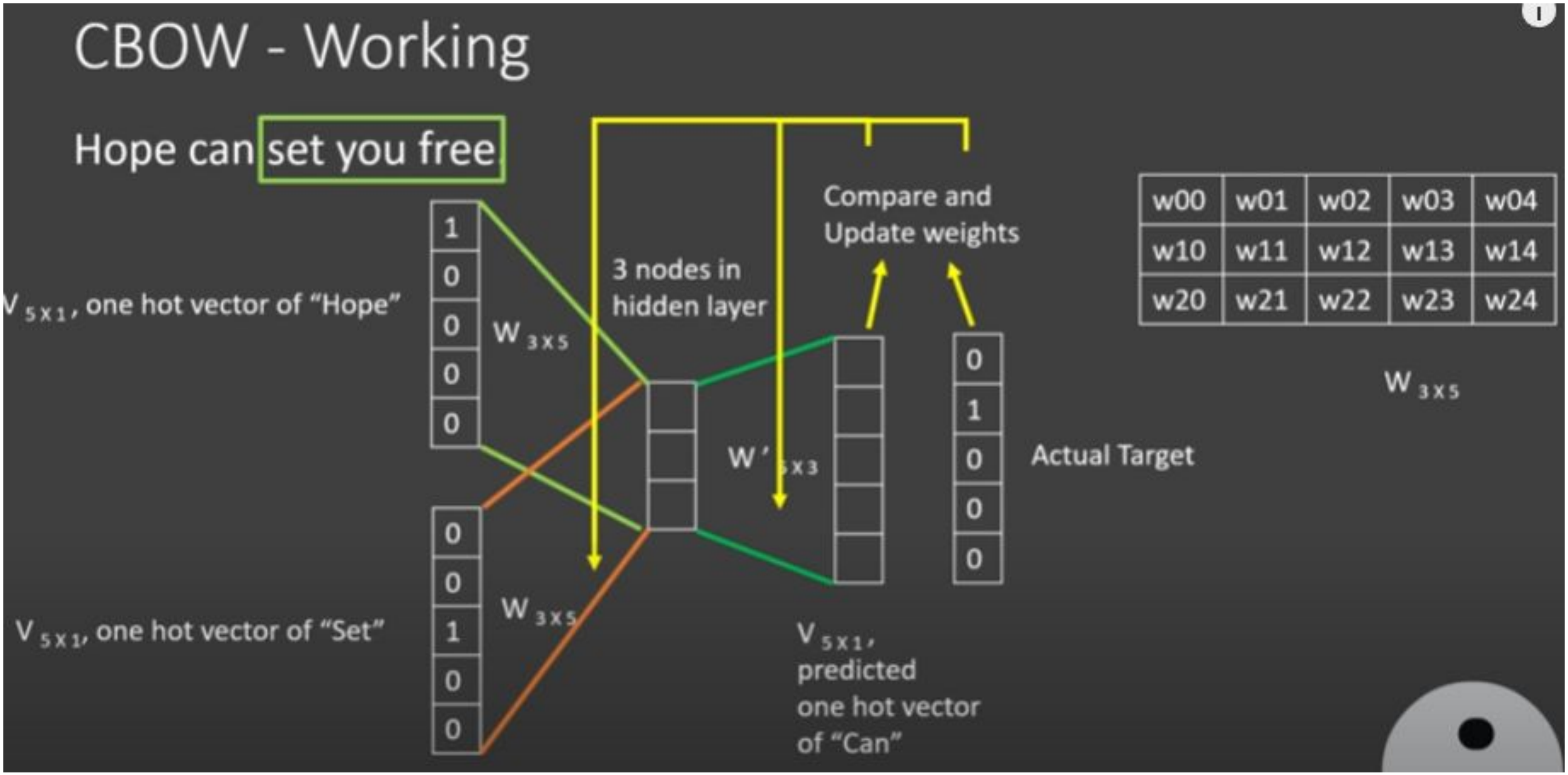
CBOW - Working

Hope can set you free.



w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

$W_{3 \times 5}$


CBOW - Working
Hope can set you free
V 5x1, one hot vector of "Hope"
1
0
0
0
0
W 3x5
V 5x1, one hot vector of "Set"
0
0
1
0
0
W 3x5
3 nodes in hidden layer
W' 5x3
Compare and Update weights
V 5x1, predicted one hot vector of "Can"
0
1
0
0
0
Actual Target
w00 w01 w02 w03 w04
w10 w11 w12 w13 w14
w20 w21 w22 w23 w24
W 3x5

Getting word embeddings

One Hot vector of words $V_{5 \times 1}$

Hope	can	set	you	free
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Weights after training

$W_{3 \times 5}$

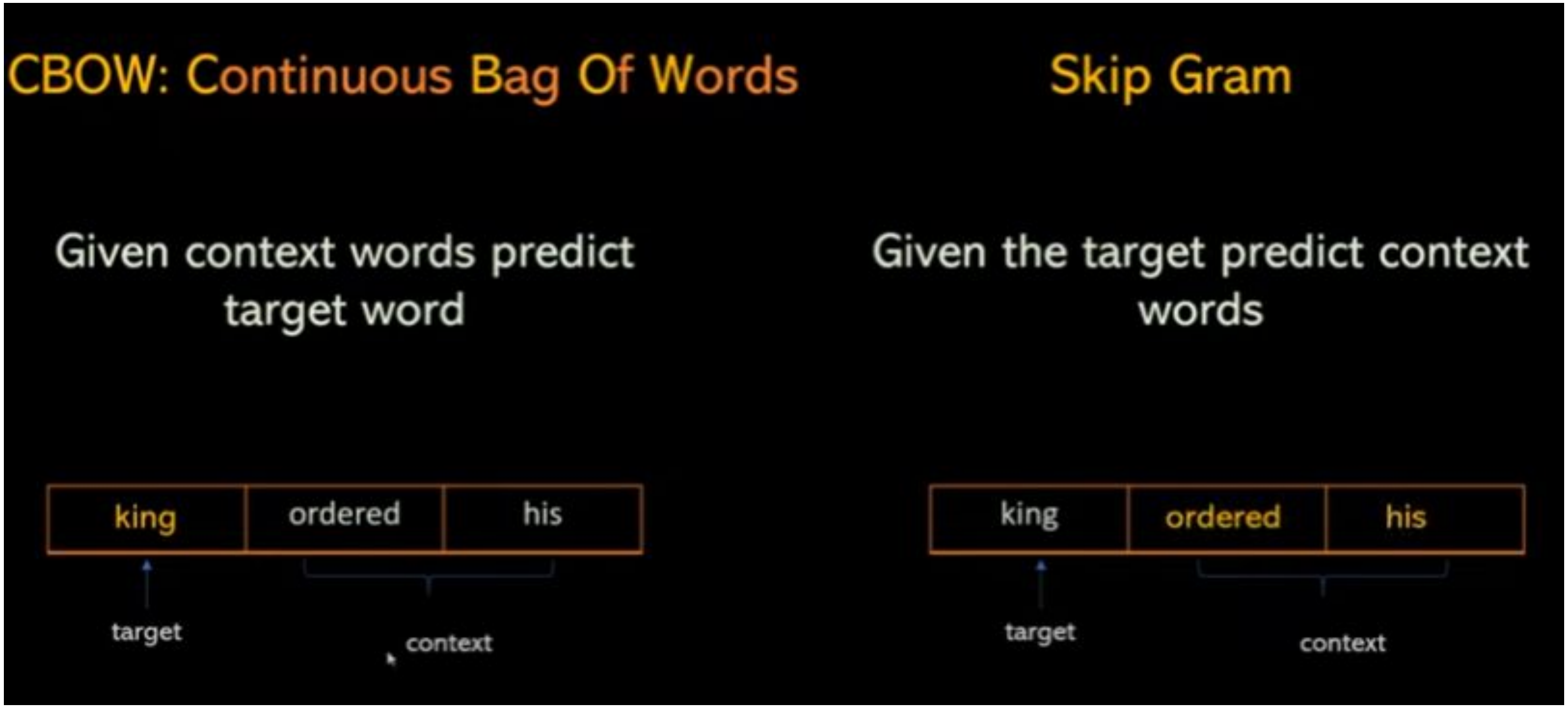
w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

Word Vector for hope $= W_{3 \times 5} \times V_{5 \times 1}$

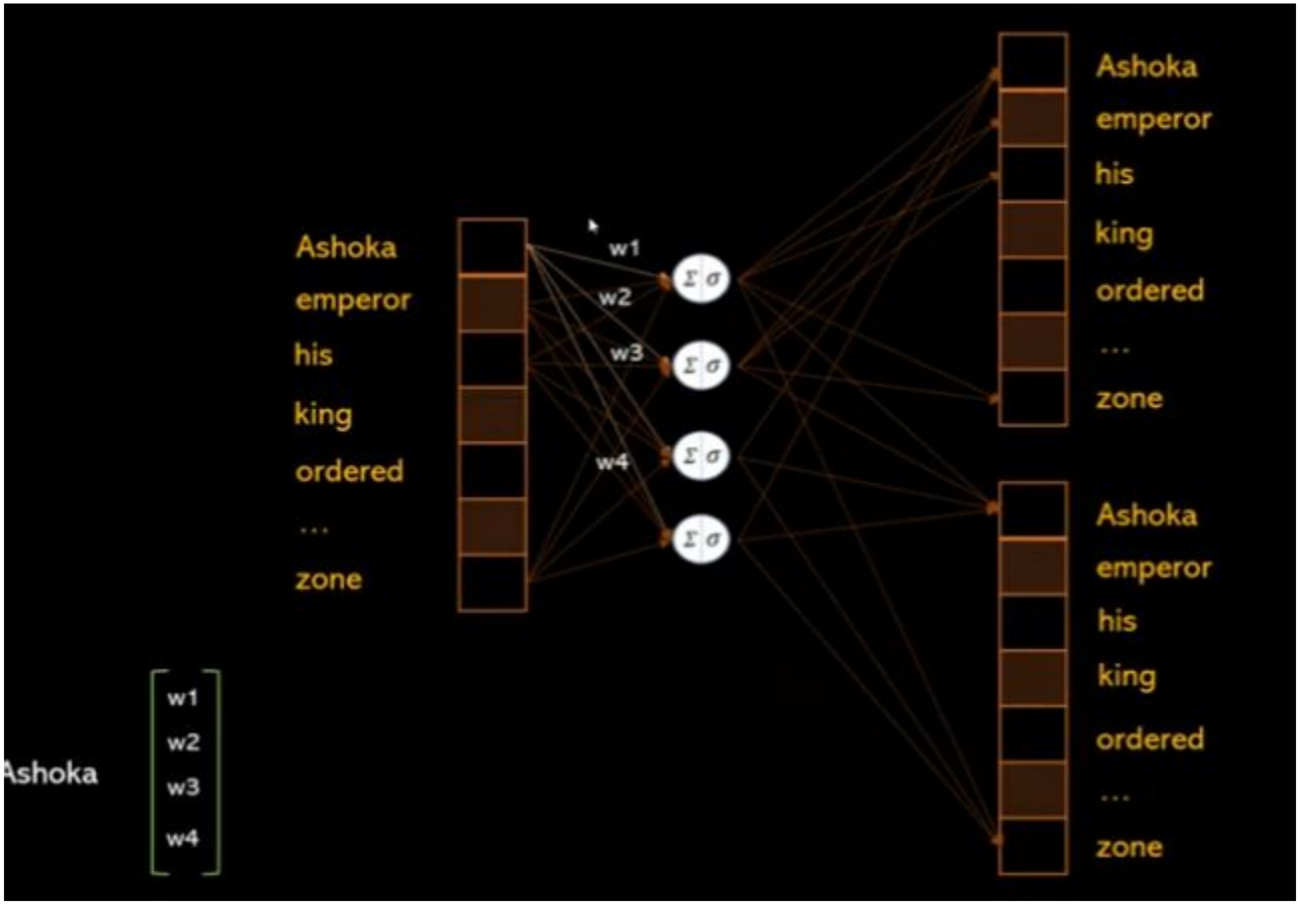
$$\begin{bmatrix} w00 & w01 & w02 & w03 & w04 \\ w10 & w11 & w12 & w13 & w14 \\ w20 & w21 & w22 & w23 & w24 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \underset{V_{3 \times 1}}{\begin{bmatrix} w00 \\ w10 \\ w20 \end{bmatrix}}$$

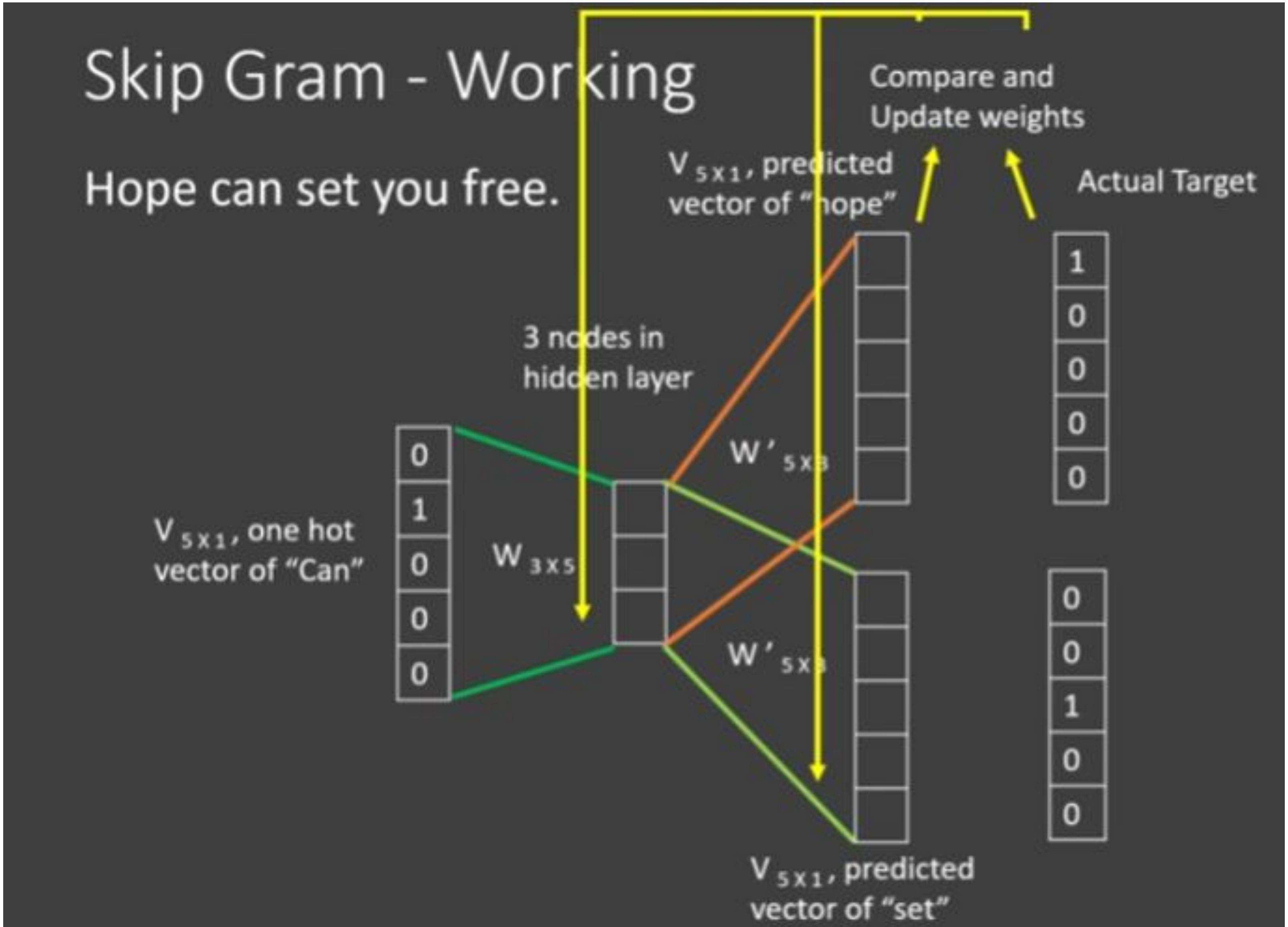
Improving the accuracy

- Choice of Model architecture (CBOW / Skipgra
  - Large Corpus, higher dimensions, slower– Skipgram
  - Small Corpus, Faster - CBOW
- Increasing the training dataset.
- Increasing the vector dimensions
- Increasing the windows size.


CBOW: Continuous Bag Of Words
Skip Gram
Given context words predict target word
Given the target predict context words
king
ordered
his
target
context
king
ordered
his
target
context

Skip Gram Model





Skip Gram - Working
Hope can set you free.
Compare and Update weights
V 5x1, predicted vector of "hope"
Actual Target
3 nodes in hidden layer
W' 5x3
V 5x1, one hot vector of "Can"
W 3x5
W' 5x3
V 5x1, predicted vector of "set"

Getting word embeddings

Weights after training

$W_{3 \times 5}$

w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

One Hot vector of words $V_{5 \times 1}$

Hope	can	set	you	free
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Word Vector for hope $= W_{3 \times 5} \times V_{5 \times 1}$

$$\begin{bmatrix} w00 & w01 & w02 & w03 & w04 \\ w10 & w11 & w12 & w13 & w14 \\ w20 & w21 & w22 & w23 & w24 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \underset{V_{3 \times 1}}{\begin{bmatrix} w00 \\ w10 \\ w20 \end{bmatrix}}$$