Deep learning

lecture 5, fall 22

NLP basics,
Recurrent neural networks

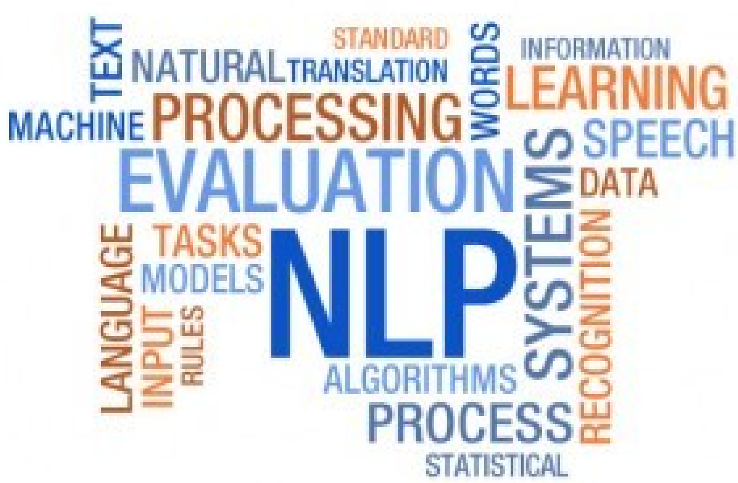
British Hedgehog Preservation Society

Skoltech

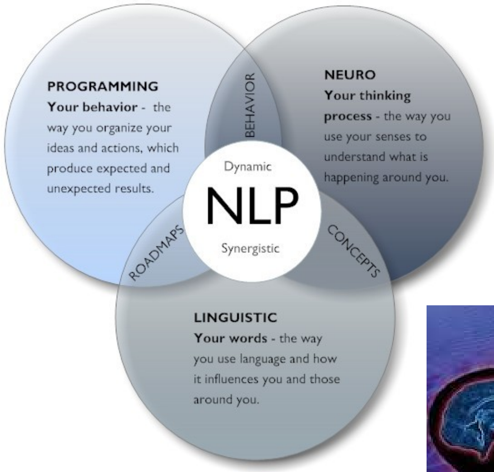
Skolkovo Institute of Science and Technology

LAMBDA

What is NLP?



NLP
Light side of the force



NLP
Dark side of the force

Text 101

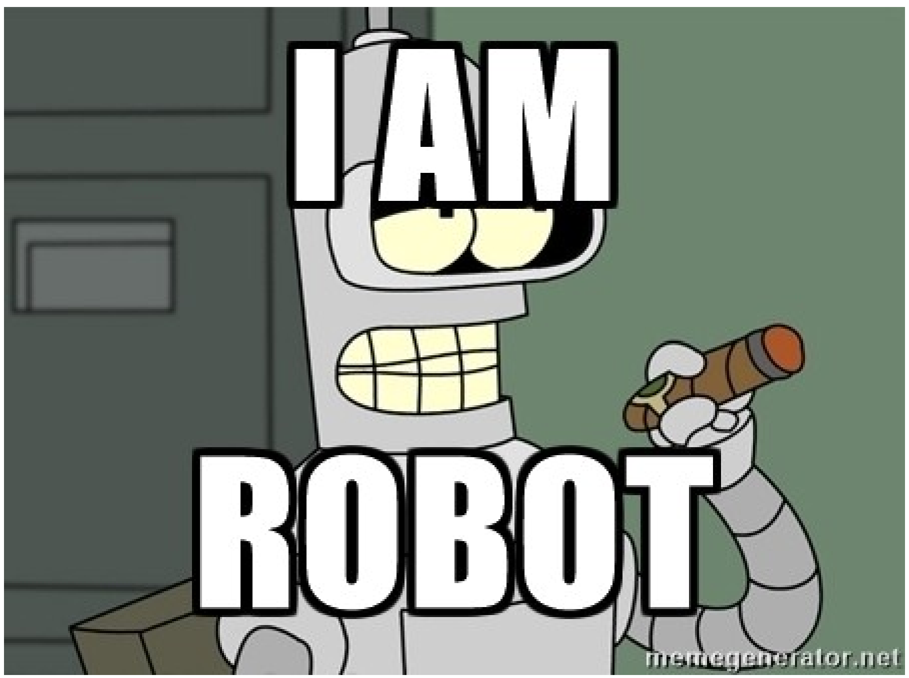
text

/tɛkst/

noun

1. a book or other written or printed work, regarded in terms of its content rather than its physical form.
   "a text which explores pain and grief"
   synonyms: written work, book, work, printed work, narrative
   "a text which explores pain and grief"
2. the main body of a book or other piece of writing, as distinct from other material such as notes, appendices, and illustrations.
   "the pictures are clear and relate well to the text"
   synonyms: words, wording; More

Text 101: nlp perspective

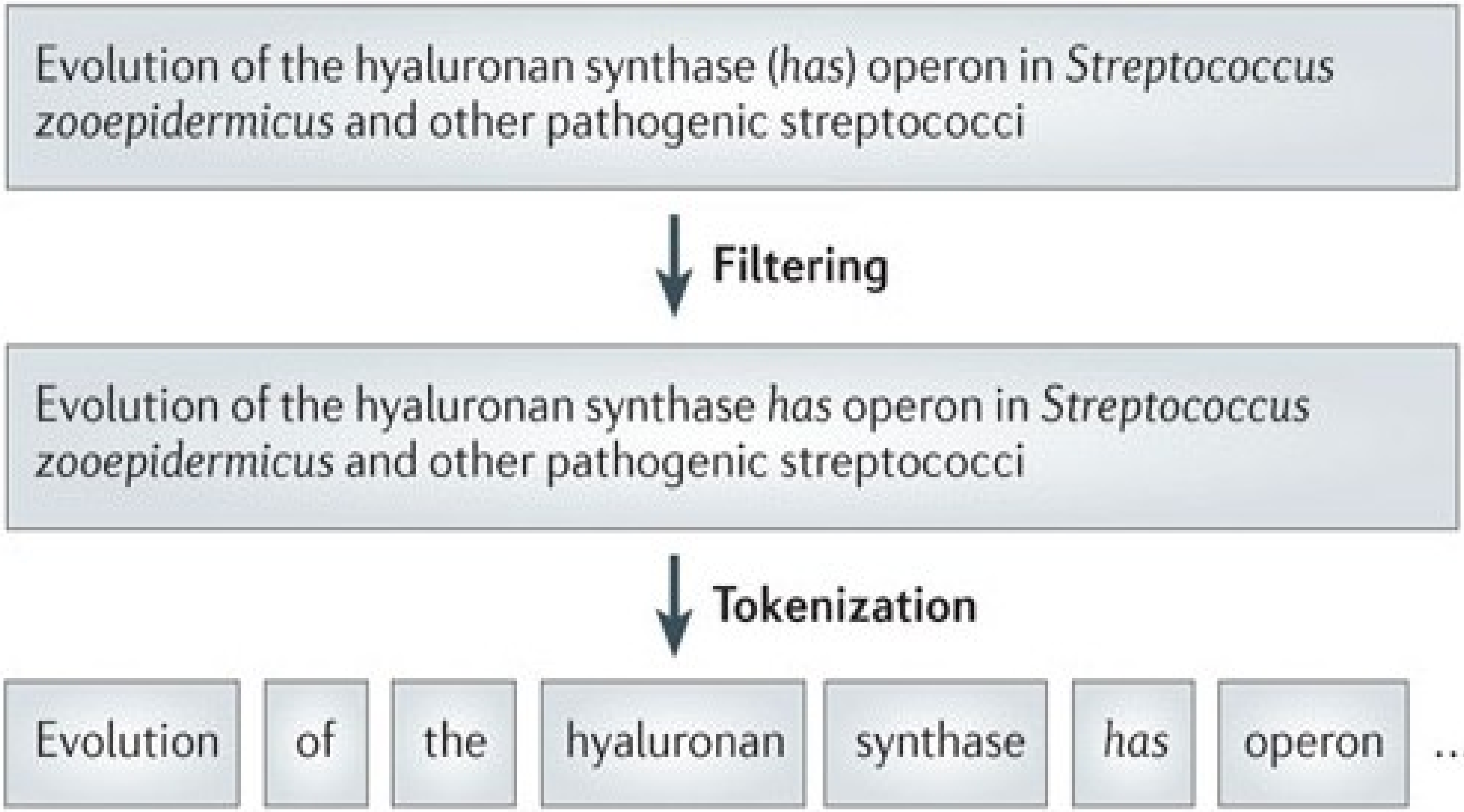


<u>Text:</u>
A sequence of tokens(words).

<u>Token</u>/word:
A sequence of characters.

<u>Character:</u>
An atomic element of text.
¯_(ツ)_/¯

Text 101: tokens



Why bother with texts?

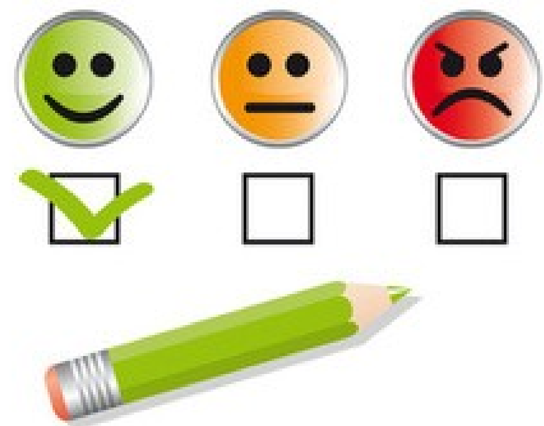
Classification/regression

- Sentiment analysis
- Comment moderation
- Predicting job salary
- Predicting tweet popularity

More complex

- Machine translation
- Information retrieval (web search)
- Conversation systems (chat bots)
- News summarization

Text classification/regression



Applications:

- Adult content filter (safe search)
- Detect age/gender/interests by search querries
- Convert movie review into “stars”
- Survey public opinion for the new iphone Vs old one (SNA)

...

Text classification/regression

text $\longrightarrow$ features $\longrightarrow$ ML model $\longrightarrow$ $P(y|x)$

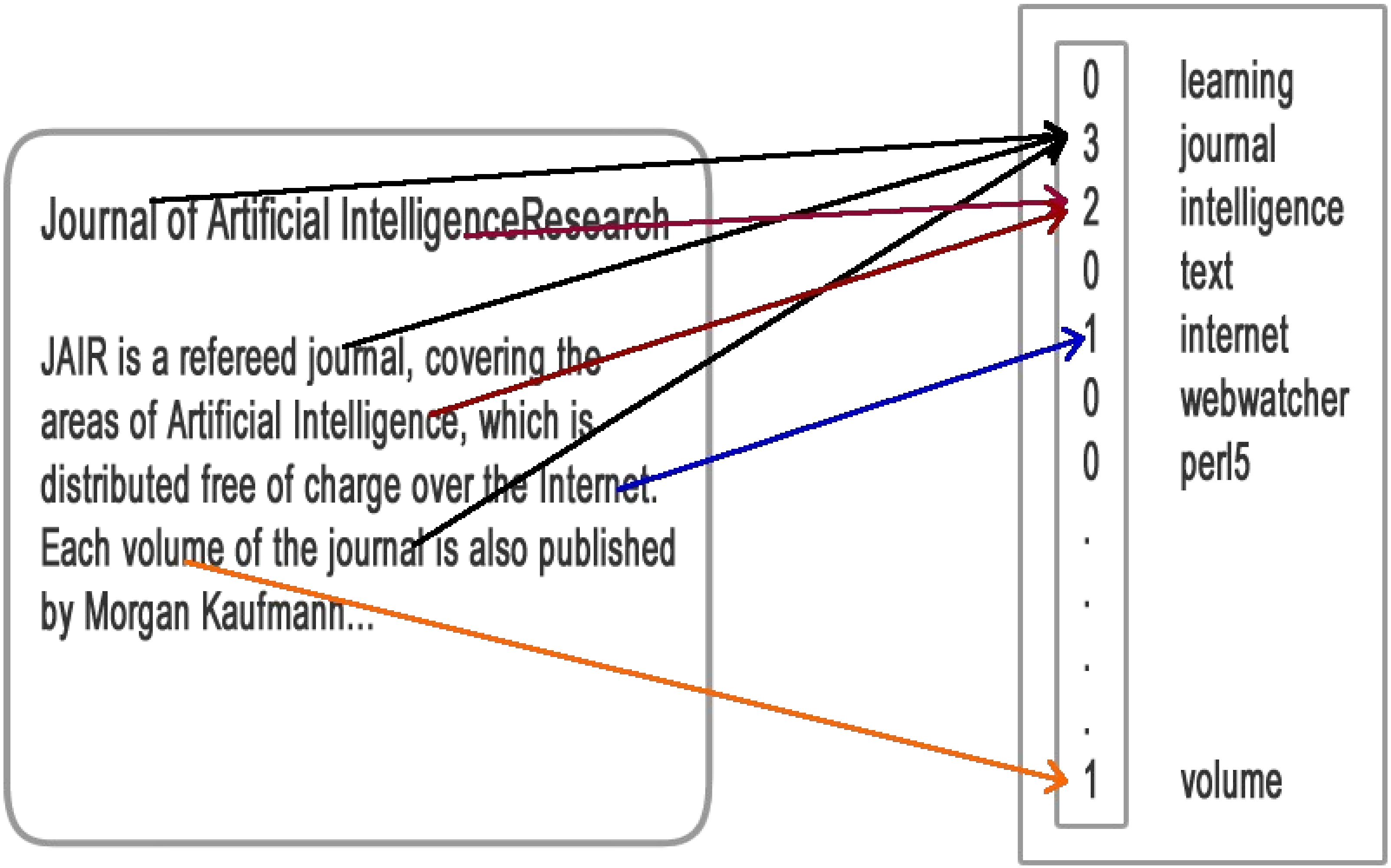
Text classification/regression

text → features → ML model → P(y|x)

↑

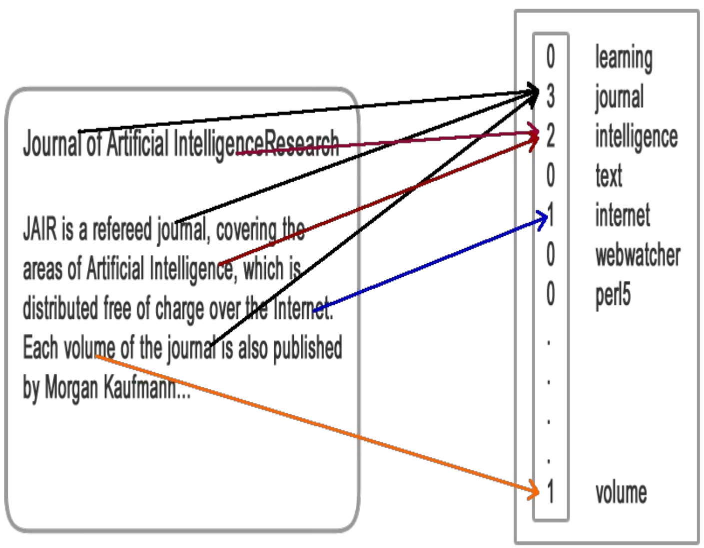
Can we represent text as a
constant-size feature vector?

Bag of words

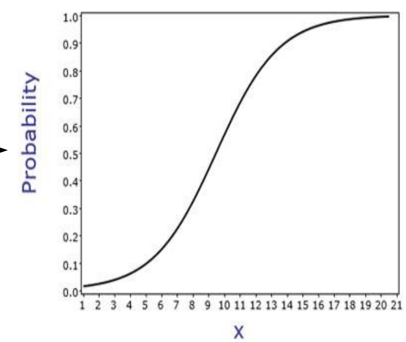


Journal of Artificial IntelligenceResearch
JAIR is a refereed journal, covering the areas of Artificial Intelligence, which is distributed free of charge over the Internet. Each volume of the journal is also published by Morgan Kaufmann...
0 learning
3 journal
2 intelligence
0 text
1 internet
0 webwatcher
0 perl5
.
.
.
.
1 volume

Bag of Words + Linear Model

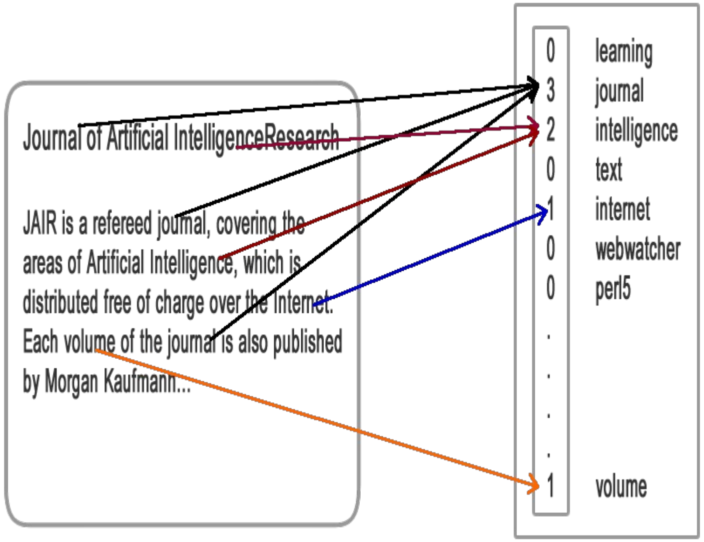


Wx +b

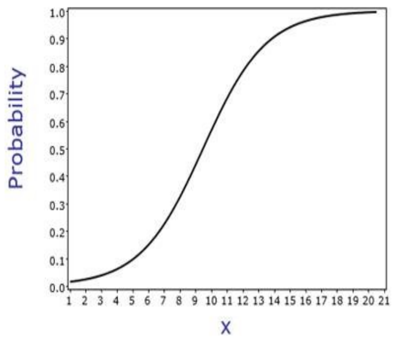


P(y)

Bag of Words + Linear Model



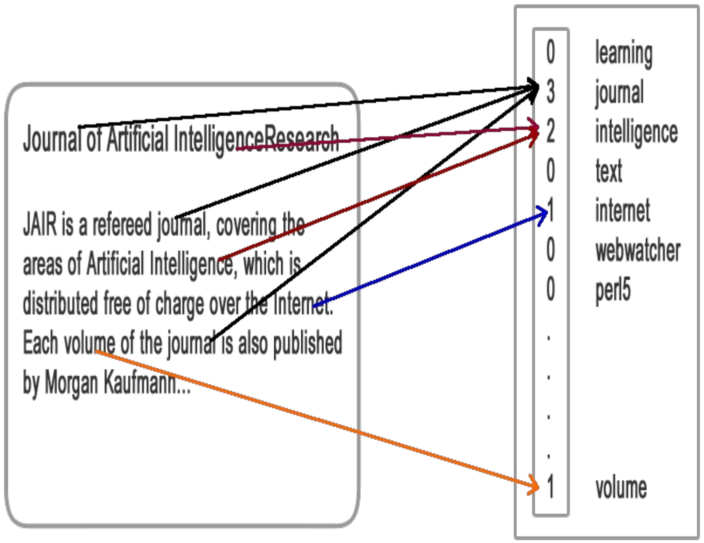
Wx +b



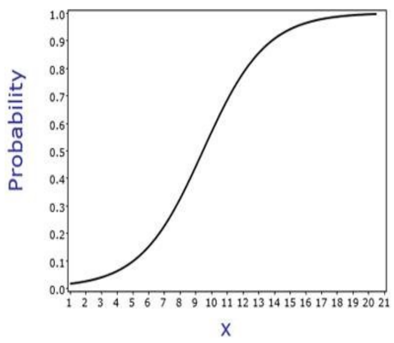
P(y)

Guess: How many features (approx) will such model have?

Bag of Words + Linear Model



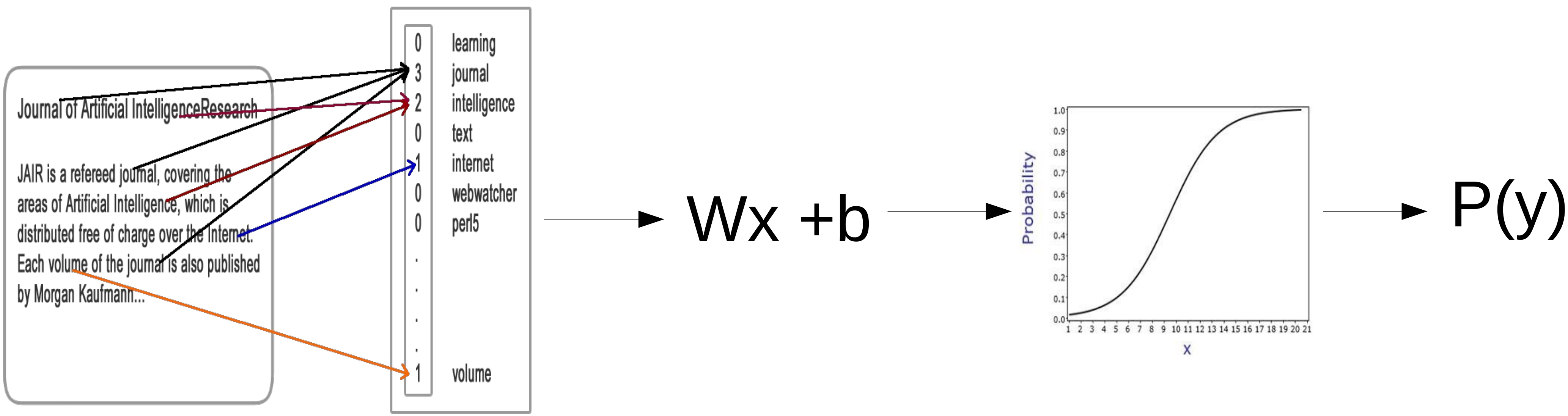
→ Wx +b →



→ P(y)

one feature for every token in the vocabulary, total ~ 10^5

Bag of Words + Linear Model



one feature for every token in the vocabulary, total ~ 10^5

Too many words

- Too many features, easy overfitting
- No information about word order
- Each word is exactly √2 away from all others

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
nice	beautiful	ugly	refrigerator

If model hasn’t seen the word ‘beautiful’, that word will be equally close to ‘nice’ and to ‘refrigerator’.

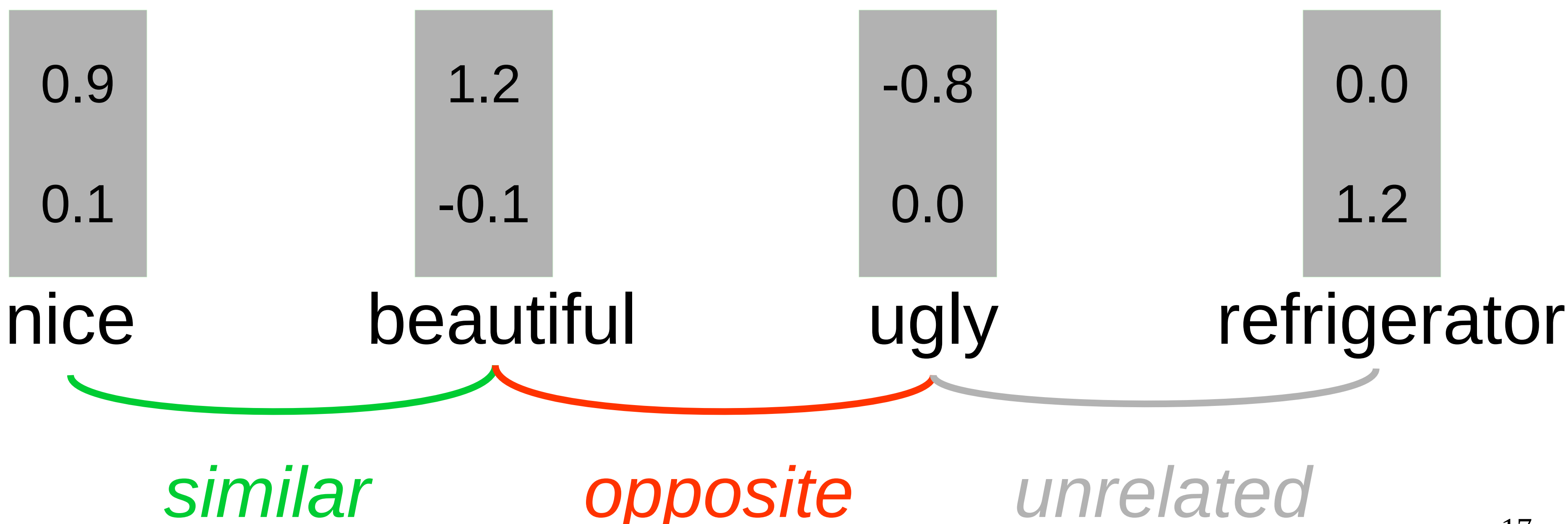
Word embeddings: core idea

Learn word features such that similar words have similar features

0.9	1.2	-0.8	0.0
0.1	-0.1	0.0	1.2
nice	beautiful	ugly	refrigerator

Word embeddings: core idea

Learn word features such that similar words have similar features



Distributional hypothesis

You shall know a word by the company it keeps
Firth (1957)

Words that occur in similar contexts
should have similar features

Nameless engineer (year unknown)

Cooccurences

He also found five fish swimming in murky water in an old **bathtub**.
We do abhor dust and dirt, and stains on the **bathtub**, and any kind of filth.
Above At the far end of the garden room a **bathtub** has been planted with herbs for the winter.
They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathtub** gin.
Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathtub**.
In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathtub** and a big box of concentrated jelly.
'I'll tell him,' said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathtub**.

the	12
a	9
of	7
and	6
in	5
...	...
like	2
water	2
boat	2
from	2
stain	1
toy	1
god-father	1
Cisco	1
...	...

(Voita et al, ysda nlp)

Cooccurence matrix

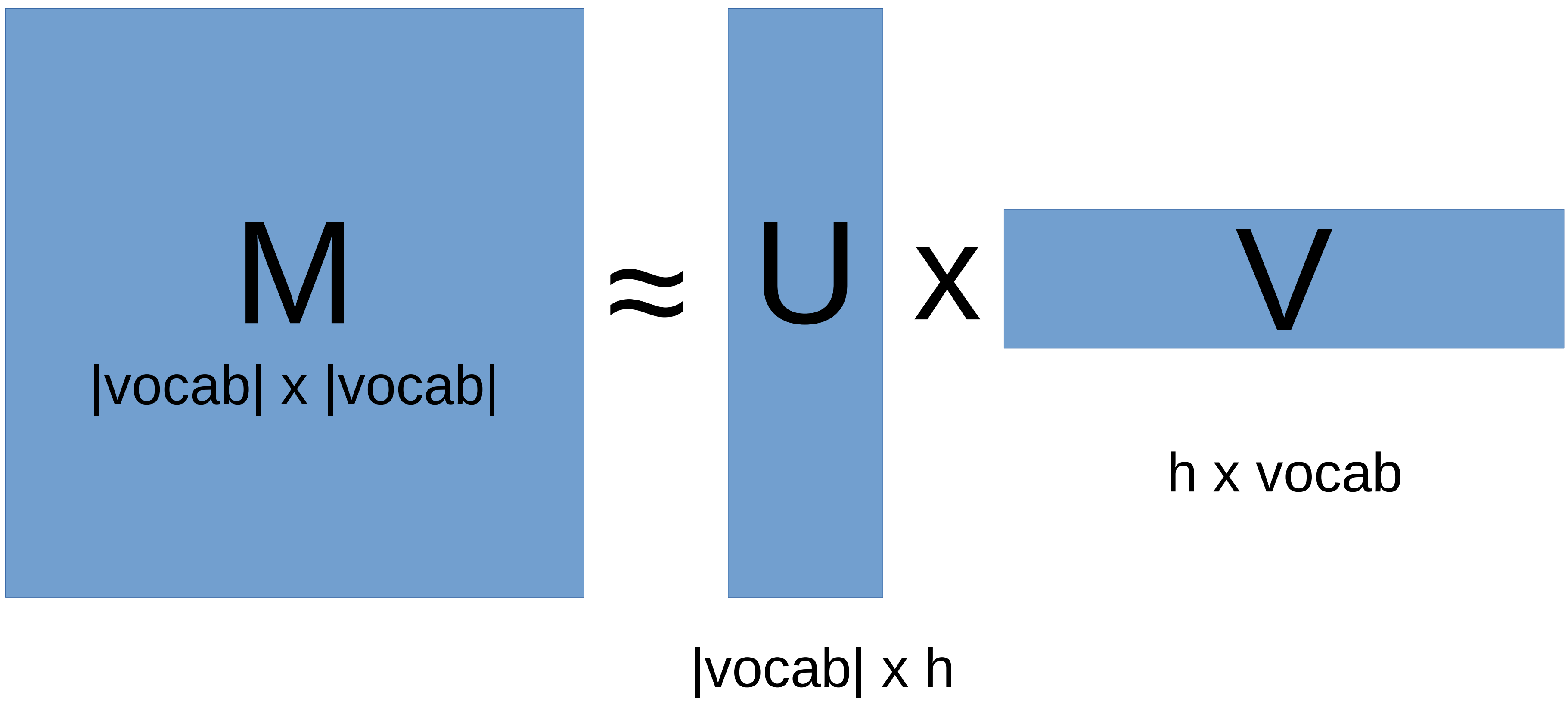
M

|vocab| x |vocab|

Co-occurence matrix
i-th row contains co-occurences of i-th word, divided by their sum

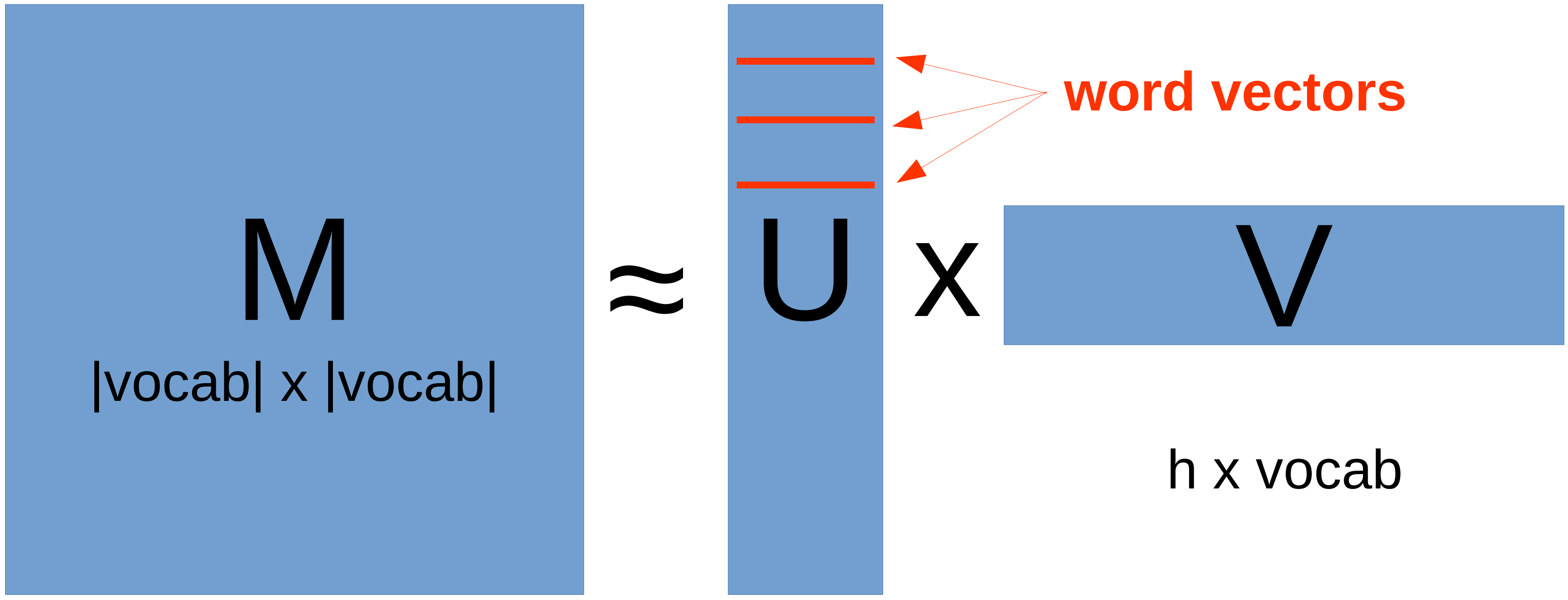
all rows sum to 1
huge, but sparse

Cooccurence matrix



h ~ 100
h << |vocab|

Cooccurence matrix



|vocab| x h

h ~ 100
h << |vocab|

Cooccurence matrix

How to obtain U and V?

$$M \approx U \times V$$

M: |vocab| x |vocab|

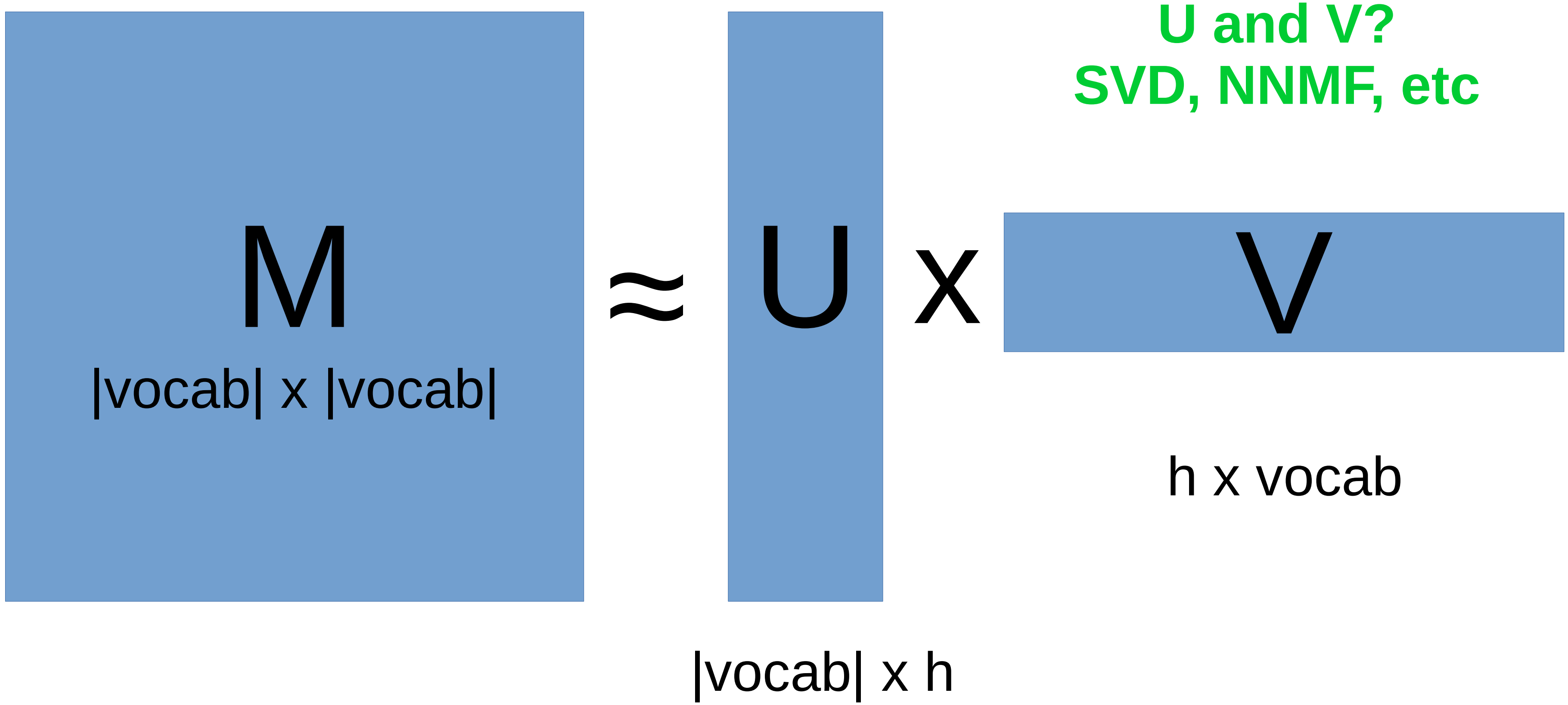
U: |vocab| x h

V: h x vocab

h ~ 100
h << |vocab|

Cooccurence matrix

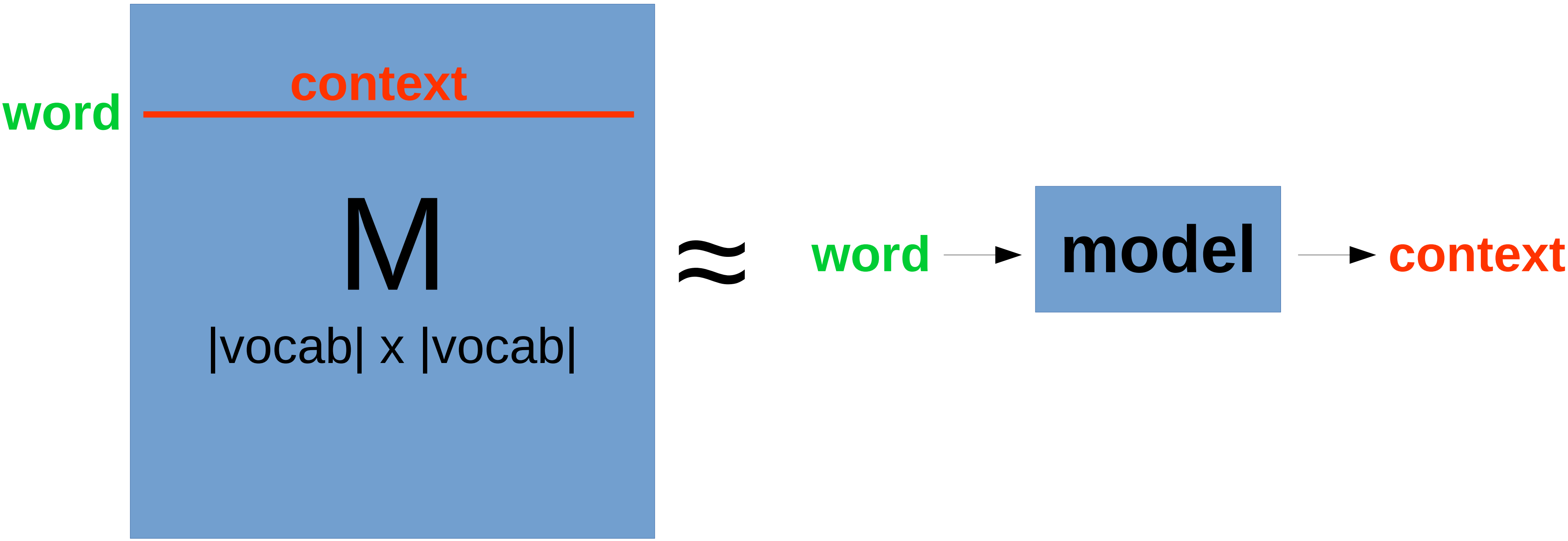
How to obtain U and V?
SVD, NNMF, etc



h ~ 100
h << |vocab|

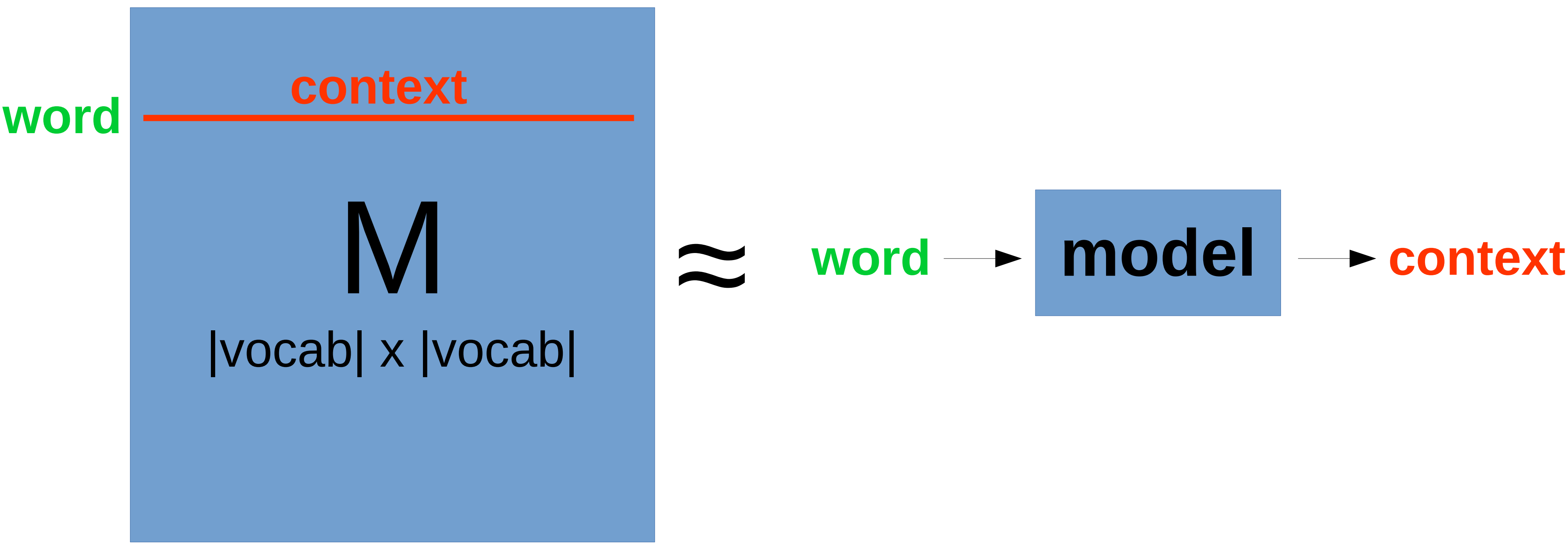
Word2vec

$$P(w_j \, is \, near | w_i) \approx e^{\langle U_i \cdot V_j \rangle} / \sum_k e^{\langle U_i \cdot V_k \rangle}$$



Word2vec

$$P(w_j\, is\, near | w_i) \approx e^{\langle U_i \cdot V_j \rangle} / \sum_k e^{\langle U_i \cdot V_k \rangle}$$



Q: how do we train it?

Word2vec

$$P\left(w_j is\, near | w_i\right) \approx e^{\langle U_i \cdot V_j \rangle} / \sum_k e^{\langle U_i \cdot V_k \rangle}$$

Crossentropy loss:

$$L = -\frac{1}{N} \sum_{w_i \in vocab} \sum_{w_j \in vocab} M_{i,j} \cdot \log P\left(w_j is\, near | w_i\right)$$

Word2vec

$$P\left(w_j is\, near \mid w_i\right) \approx e^{\langle U_i \cdot V_j \rangle} / \sum_k e^{\langle U_i \cdot V_k \rangle}$$

Crossentropy loss:

$$L = -\frac{1}{N} \sum_{w_i \in vocab} \sum_{w_j \in vocab} M_{i,j} \cdot \log P\left(w_j is\, near \mid w_i\right)$$

Re-write as sum over sentences

$$M_{i,j} = \sum_{w_i \in Text} \sum_{w_j \in context(w_j)} +1$$

$$L = -\frac{1}{N} \sum_{w_i \in Text} \sum_{w_j \in context(w_i)} \log P\left(w_j is\, near \mid w_i\right)$$

No need to compute M - train word2vec on sentences!

Embedding: word2vec

"Peace is a lie , there is only passion ."

lie

1-hot word
0
0
0
…
0
0
1
0
0
…
0
0

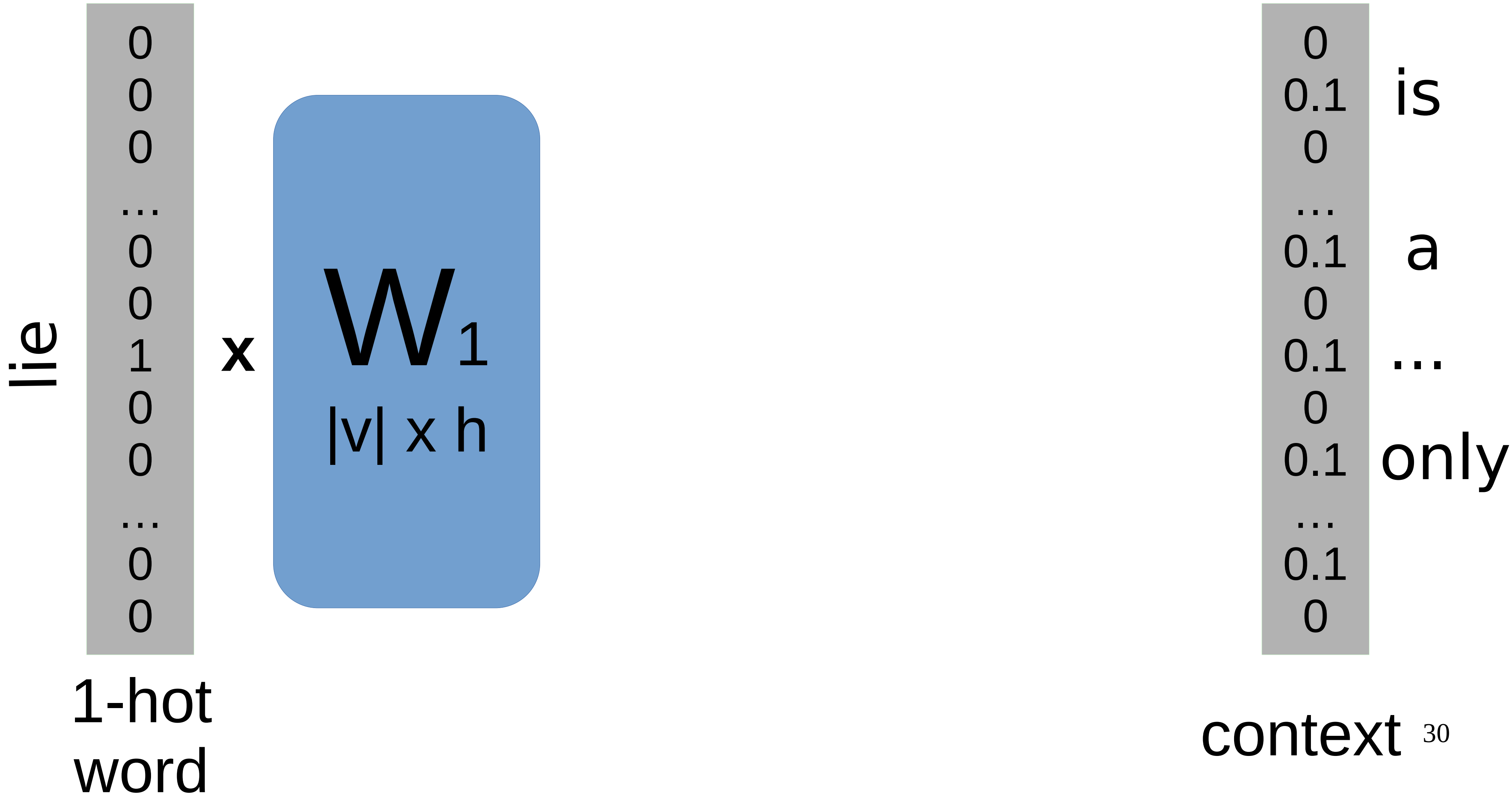
context	
0	
0.1	is
0	
…	
0.1	a
0	
0.1	…
0	
0.1	only
…	
0.1	
0	

1-hot word

context

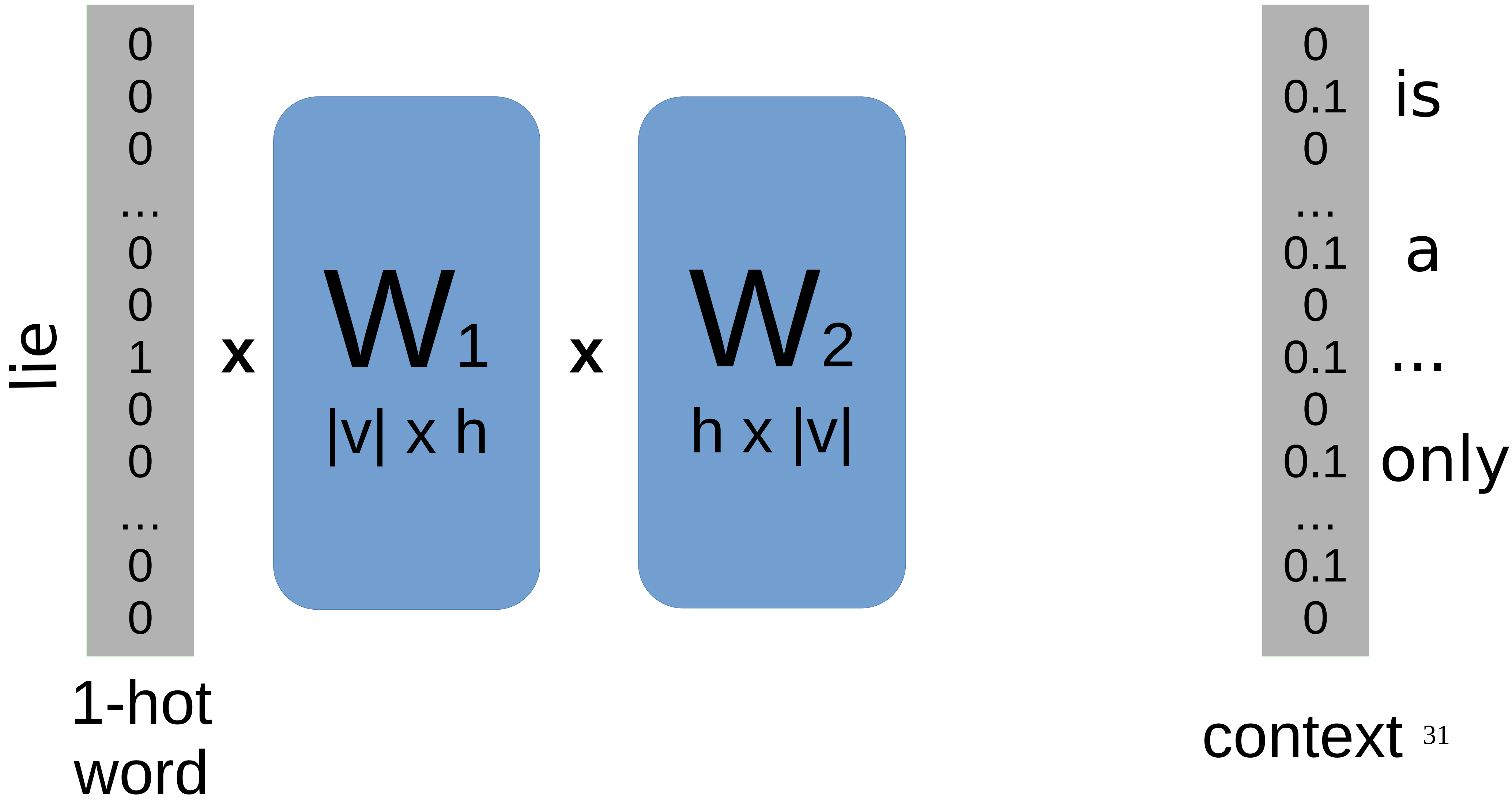
Embedding: word2vec

“Peace is a lie , there is only passion .”



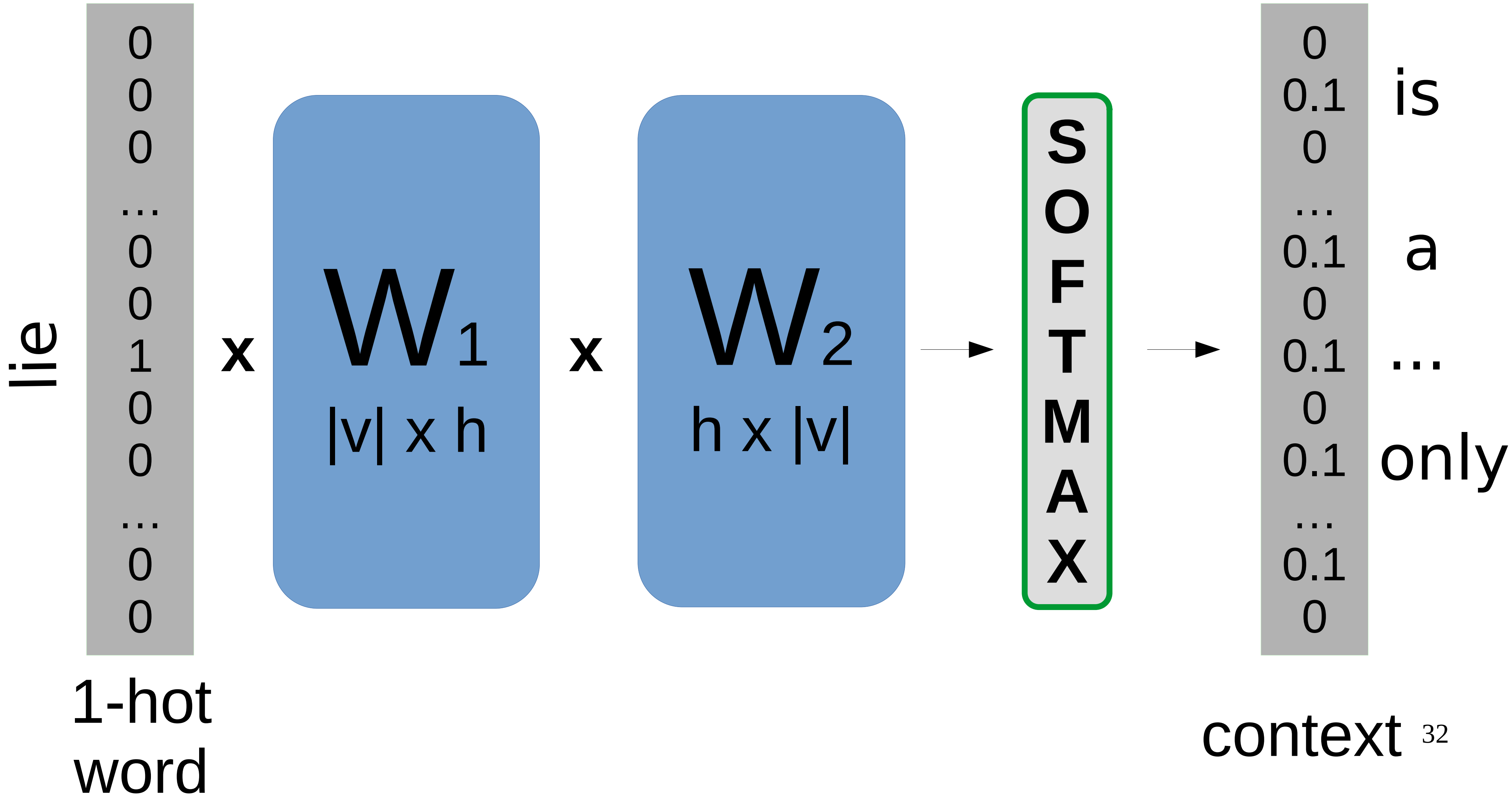
Embedding: word2vec

“Peace is a lie , there is only passion .”



Embedding: word2vec

“Peace is a lie , there is only passion .”



Embedding: word2vec

Side effect: synonyms

"nice" ~ "beautiful"

"hard" ~ "difficult"

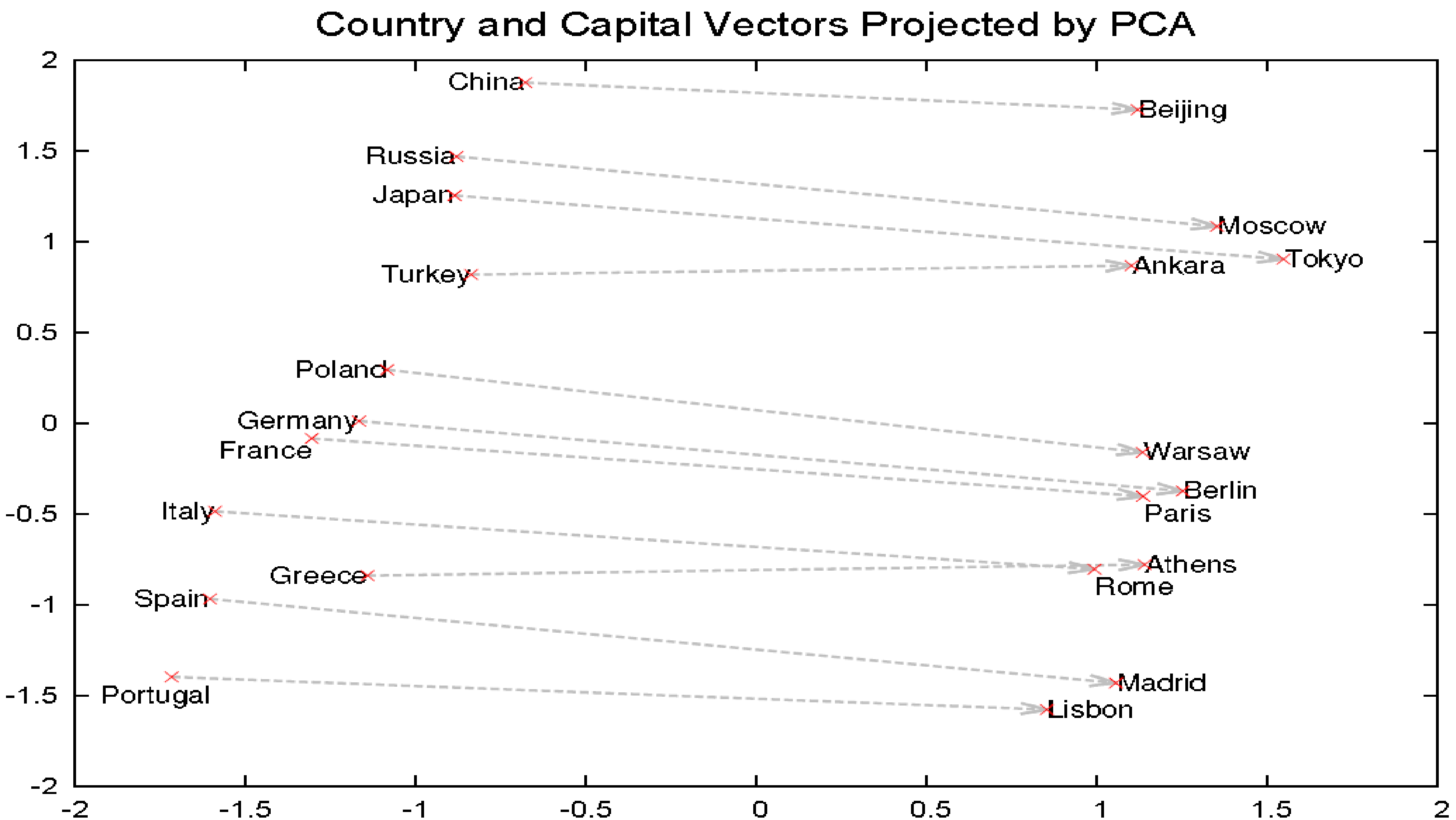
Side effect: word algebra

"king" - "man" + "woman" ~ "queen"

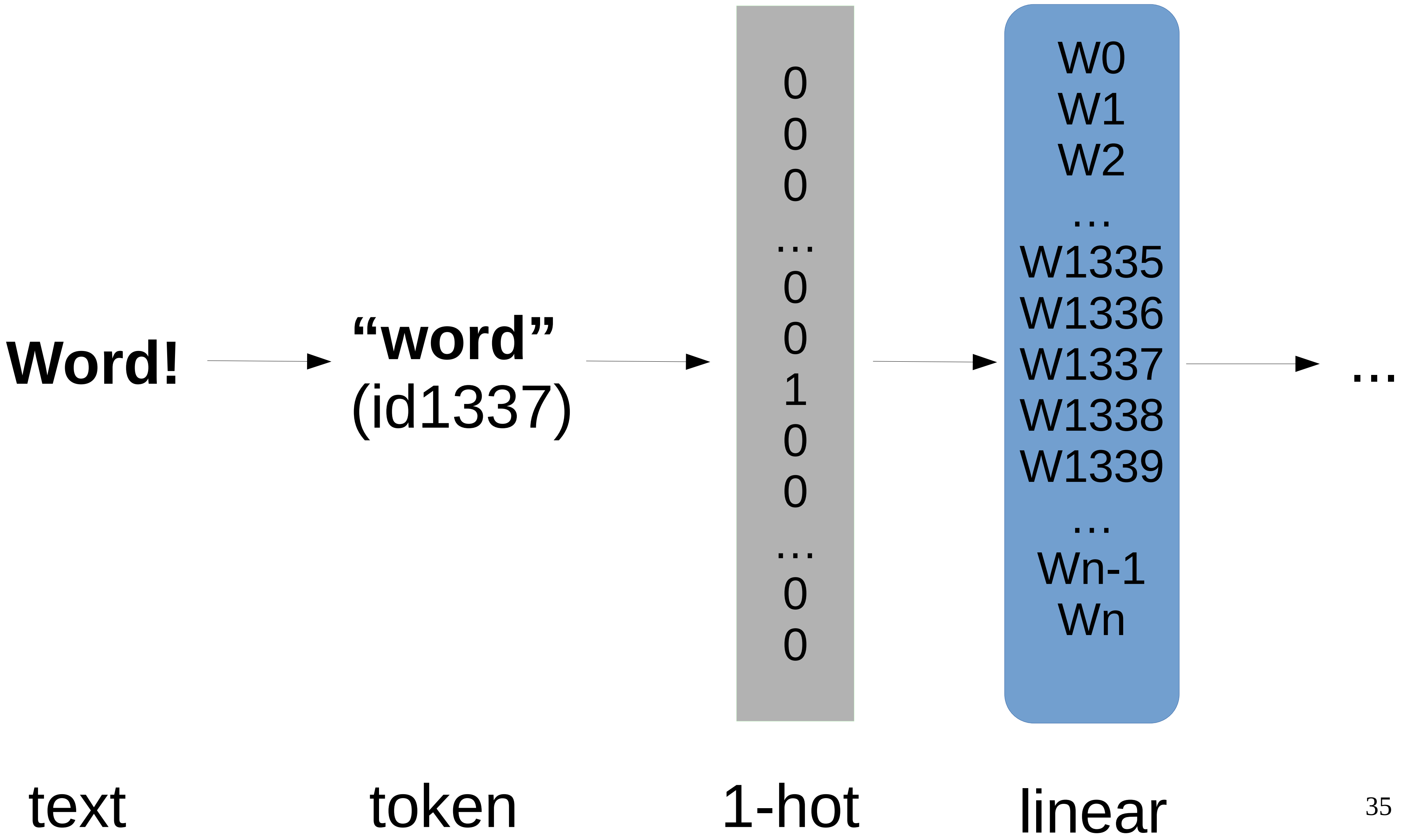
"moscow" - "russia" + "france" ~ "paris"

Embedding: word2vec

Side effect: word algebra



Sparse vector products



Sparse vector products

“word”
(id1337)

→

1-hot
0
0
0
…
0
0
1
0
0
…
0
0

dot

linear
W0
W1
W2
…
W1335
W1336
W1337
W1338
W1339
…
Wn-1
Wn

→

$$\sum_i w_i x_i = ?$$

token 1-hot linear

Sparse vector products

“word”
(id1337)

0
0
0
…
0
0
1
0
0
…
0
0

dot

W0
W1
W2
…
W1335
W1336
W1337
W1338
W1339
…
Wn-1
Wn

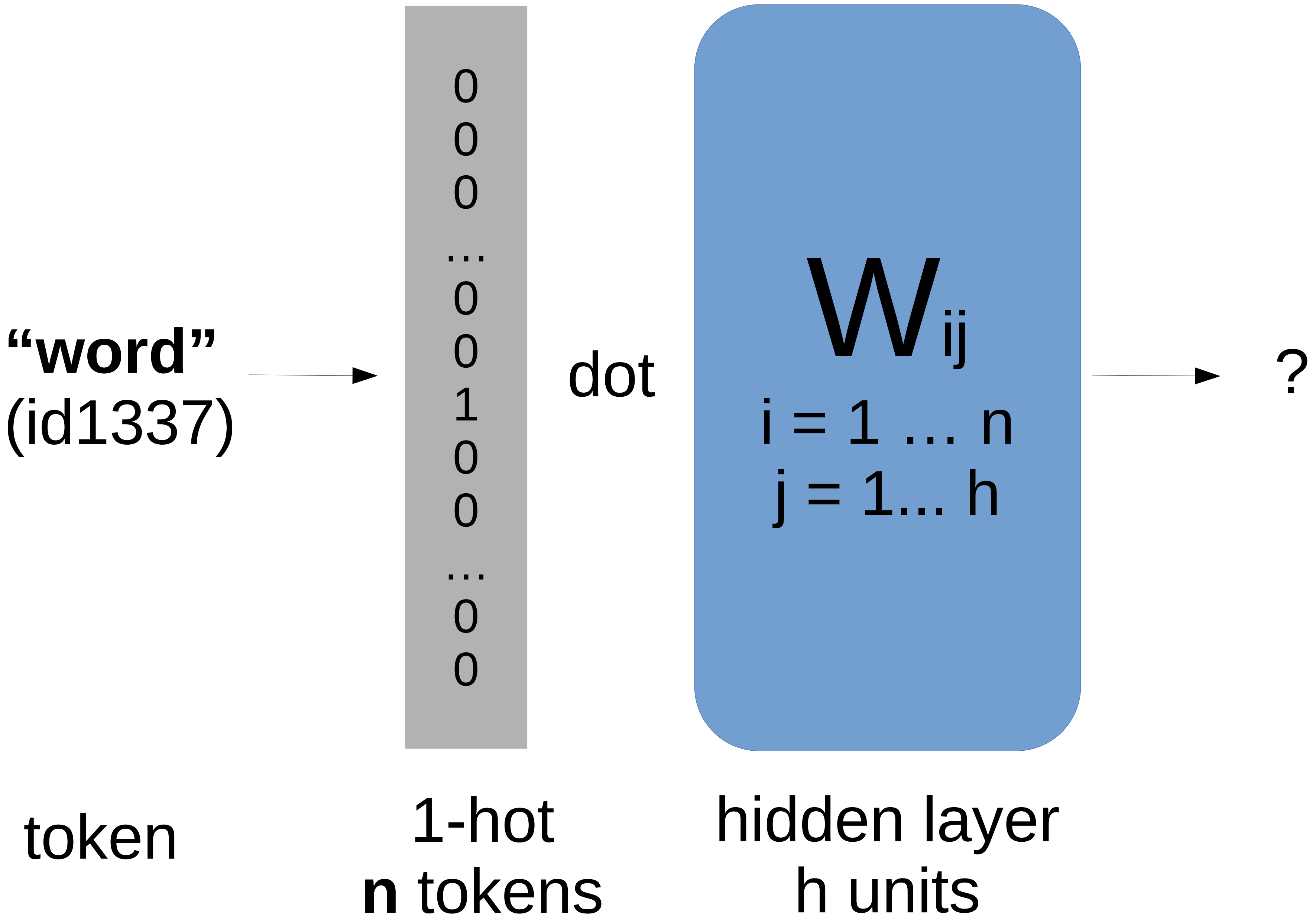
$$\sum_i w_i x_i = w_{1337}$$

token

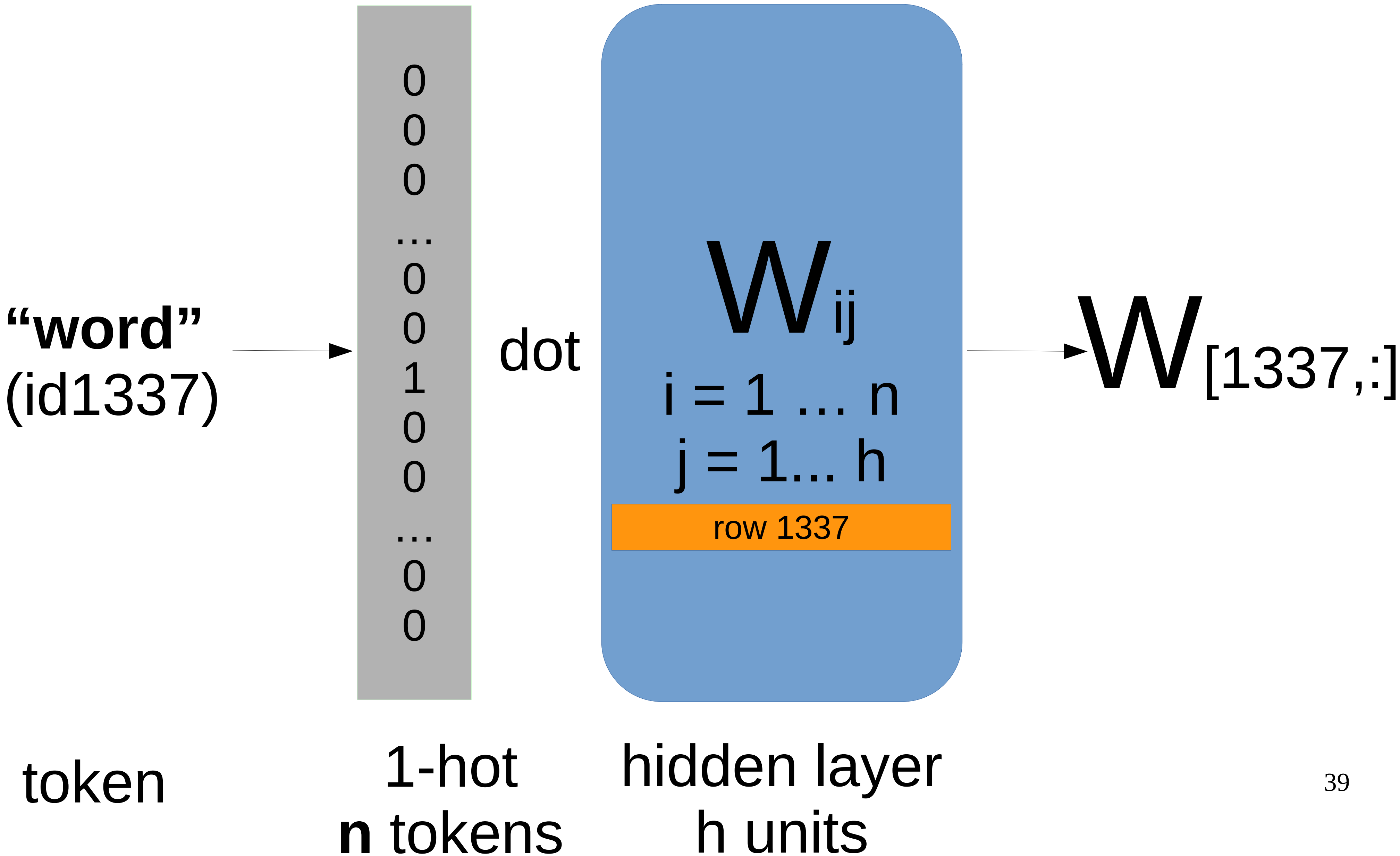
1-hot
n tokens

linear

Sparse vector products



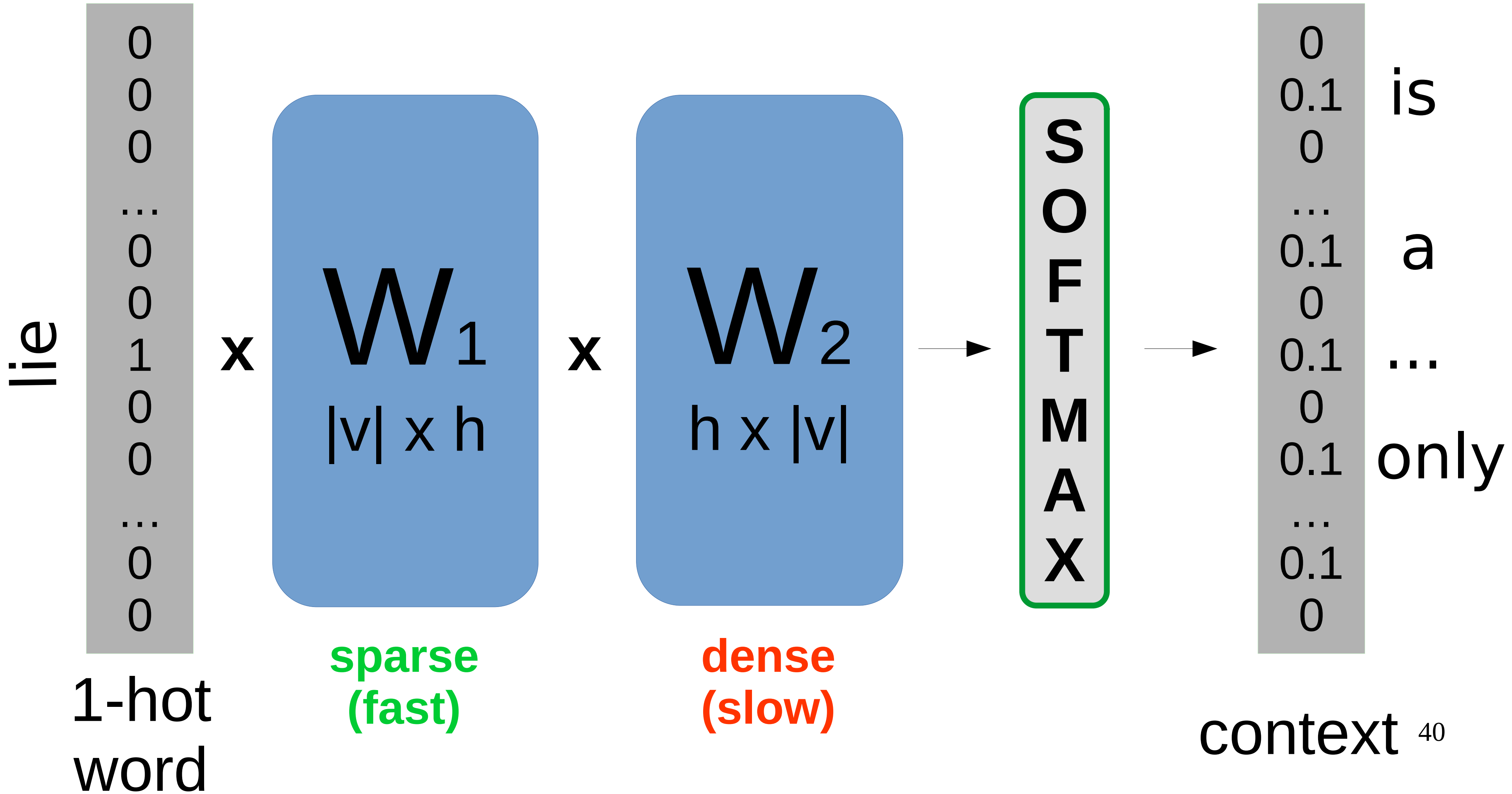
Embedding



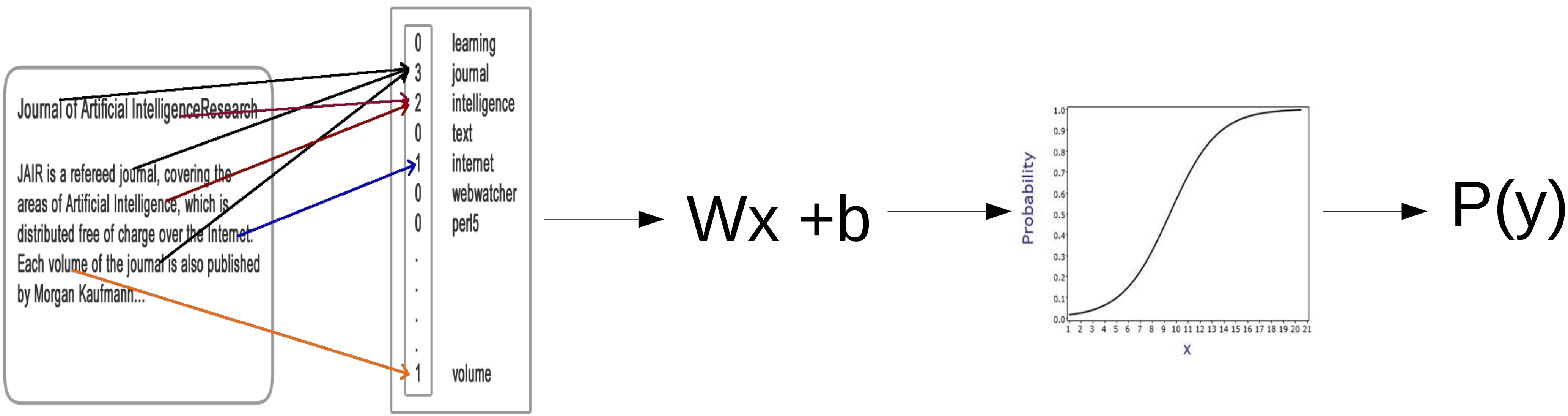
"word"
(id1337)
0
0
0
...
0
0
1
0
0
...
0
0
dot
W_{ij}
i = 1 … n
j = 1... h
row 1337
$W_{[1337,:]}$
token
1-hot
n tokens
hidden layer
h units

Embedding: word2vec

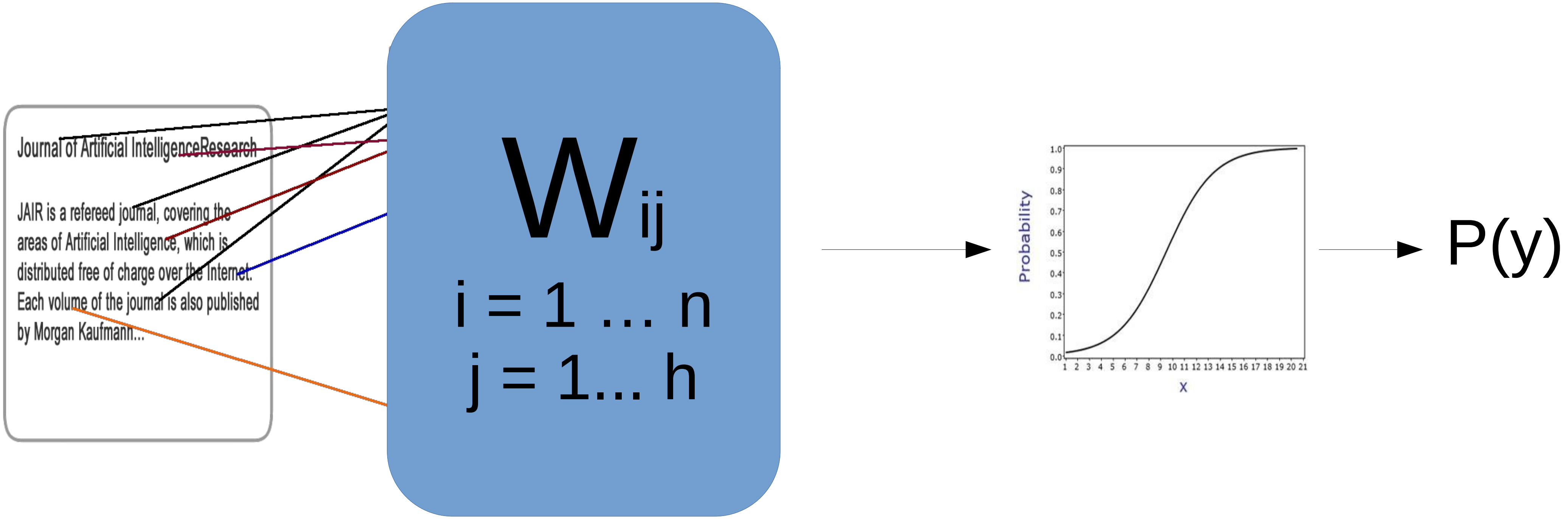
“Peace is a lie , there is only passion .”



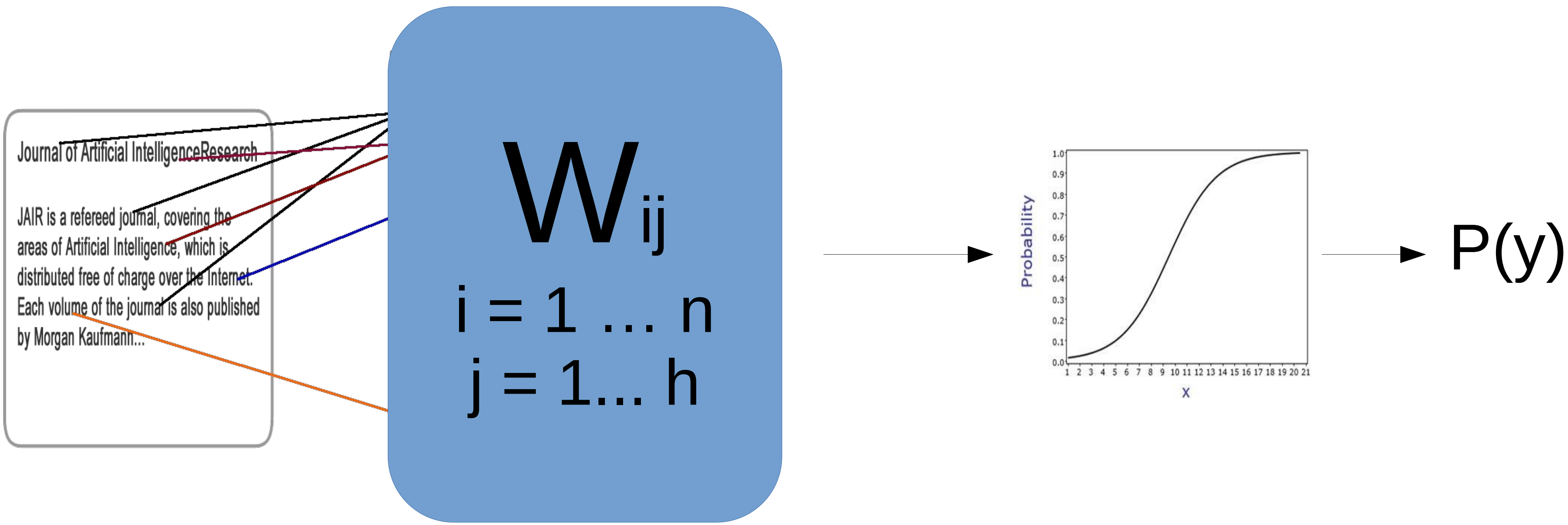
Text Classification (again)



Text Classification (again)

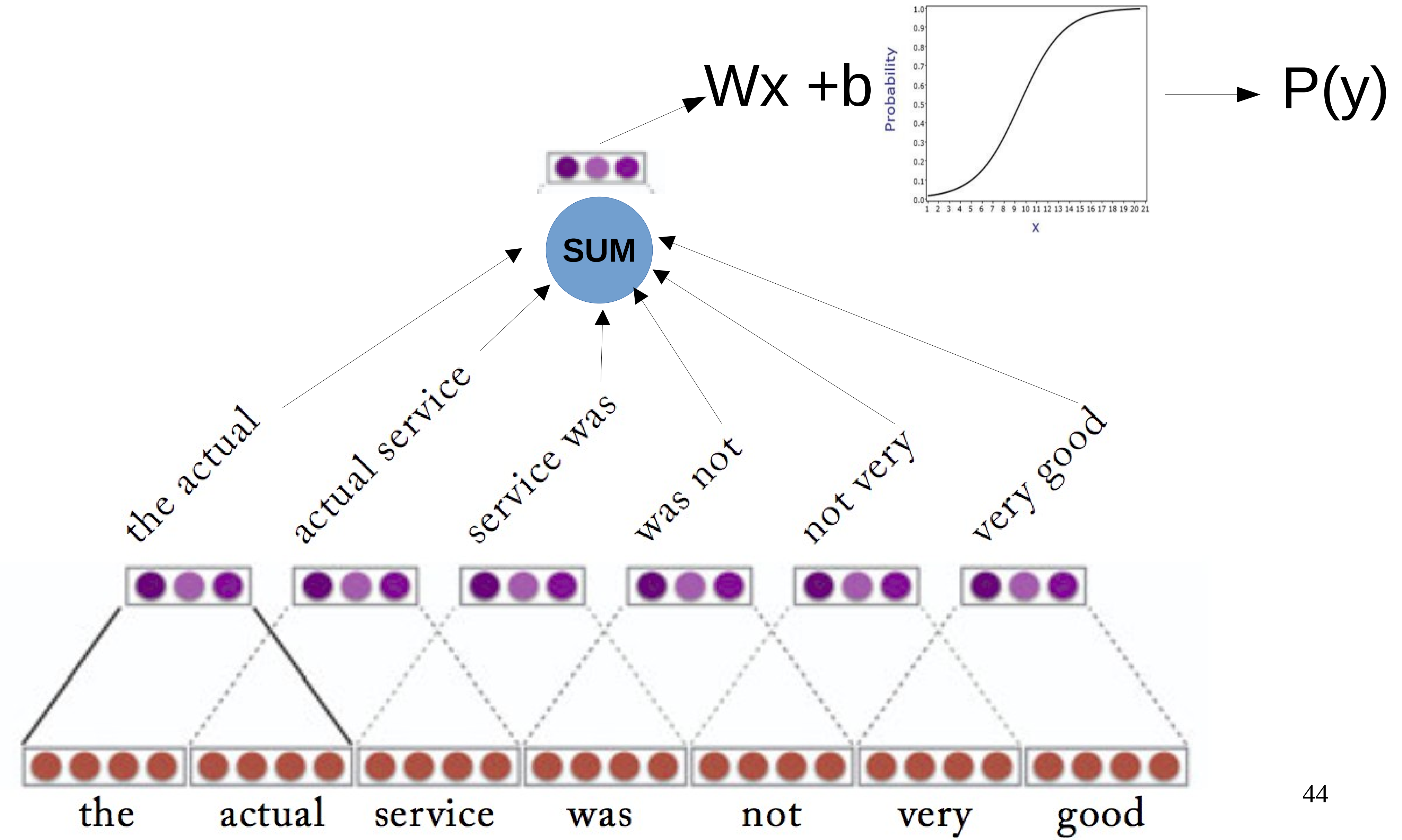


Text Classification (again)



Model does not know about word order. How to fix that?

Text Convolution



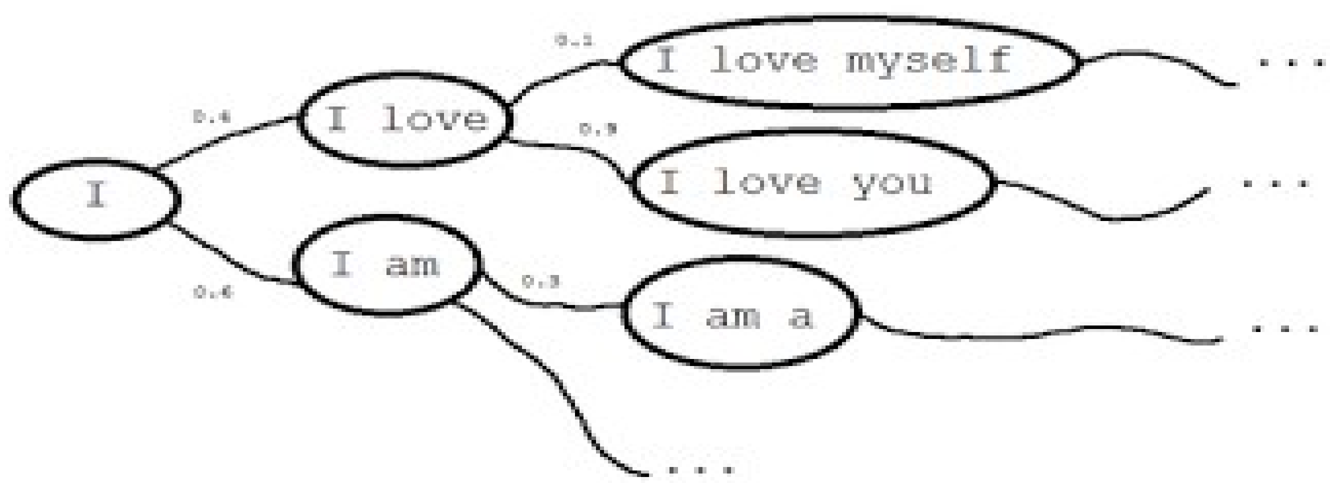
[Insert 5 minute break here]

Language model

Objective:

- Learn P(text)

$$P(text)=P(w_{0,}w_{1,}\ldots,w_n)=P(w_0)\cdot P(w_1|w_0)\cdot P(w_2|w_1w_0)\cdot\ldots\cdot P(w_n|\ldots)$$



Language model

Why learning it?

- Detect languages as P(text|language)
- Sentiment analysis P(text|happy)
- Any text analysis you can imagine
- Generate texts!
  - Cool article http://bit.ly/1K610Ie
  - Generating clickbait: http://bit.ly/21cZM70

Language model

- Actual distribution

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1 | w_0) \cdot P(w_2 | w_1 w_0) \cdot \ldots \cdot P(w_n | \ldots)$$

- Bag of words assumption (independent words)

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1) \cdot P(w_2) \cdot \ldots \cdot P(w_n)$$

- **Anything better?**

Language model

- Actual distribution

$$P(text)=P(w_{0,}w_{1,}\dots,w_n)=P(w_0)\cdot P(w_1|w_0)\cdot P(w_2|w_1 w_0)\cdot\dots\cdot P(w_n|\dots)$$

- Bag of words assumption (independent words)

$$P(text)=P(w_{0,}w_{1,}\dots,w_n)=P(w_0)\cdot P(w_1)\cdot P(w_2)\cdot\dots\cdot P(w_n)$$

- Markov assumption

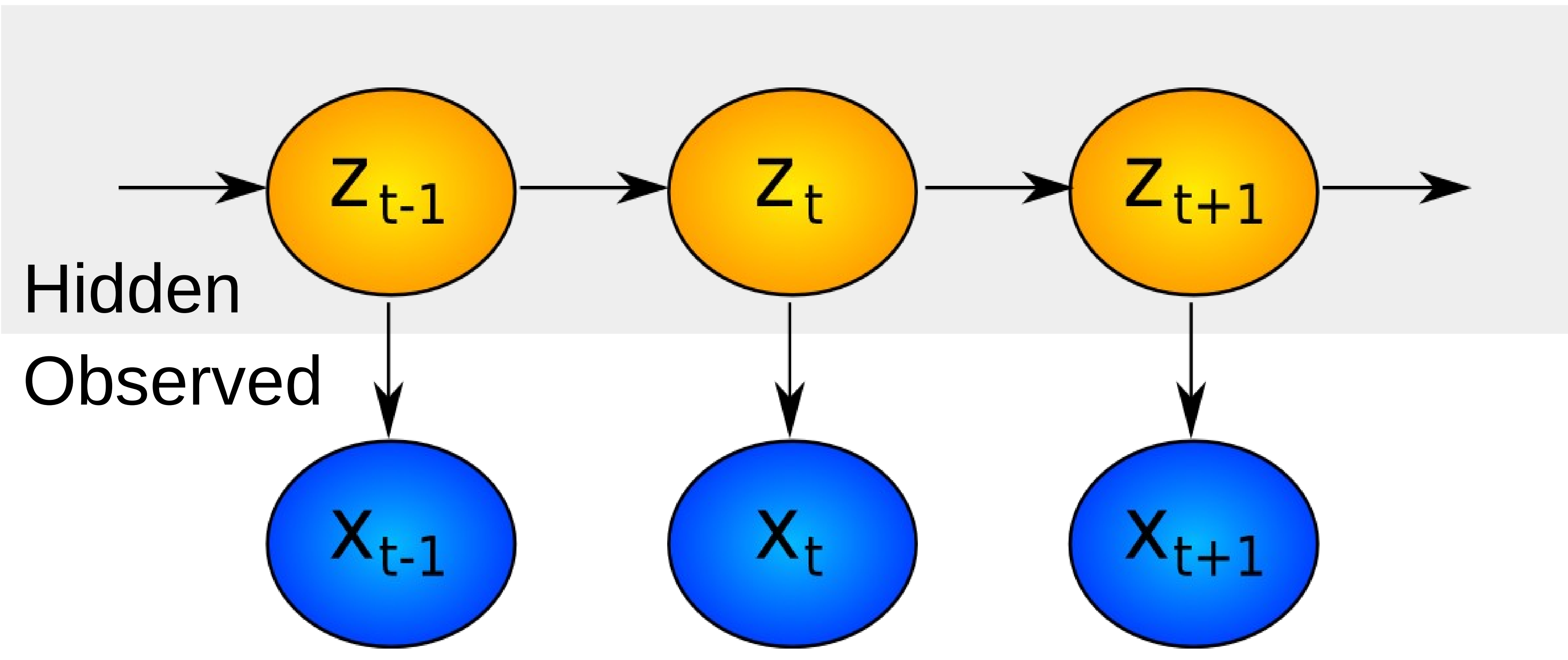
$$P(text)=P(w_{0,}w_{1,}\dots,w_n)=P(w_0)\cdot P(w_1|w\,0)\cdot P(w_2|w_1)\cdot\dots\cdot P(w_n|w_{n-1})$$

  - also 3-gram, 5-gram, 100-gram

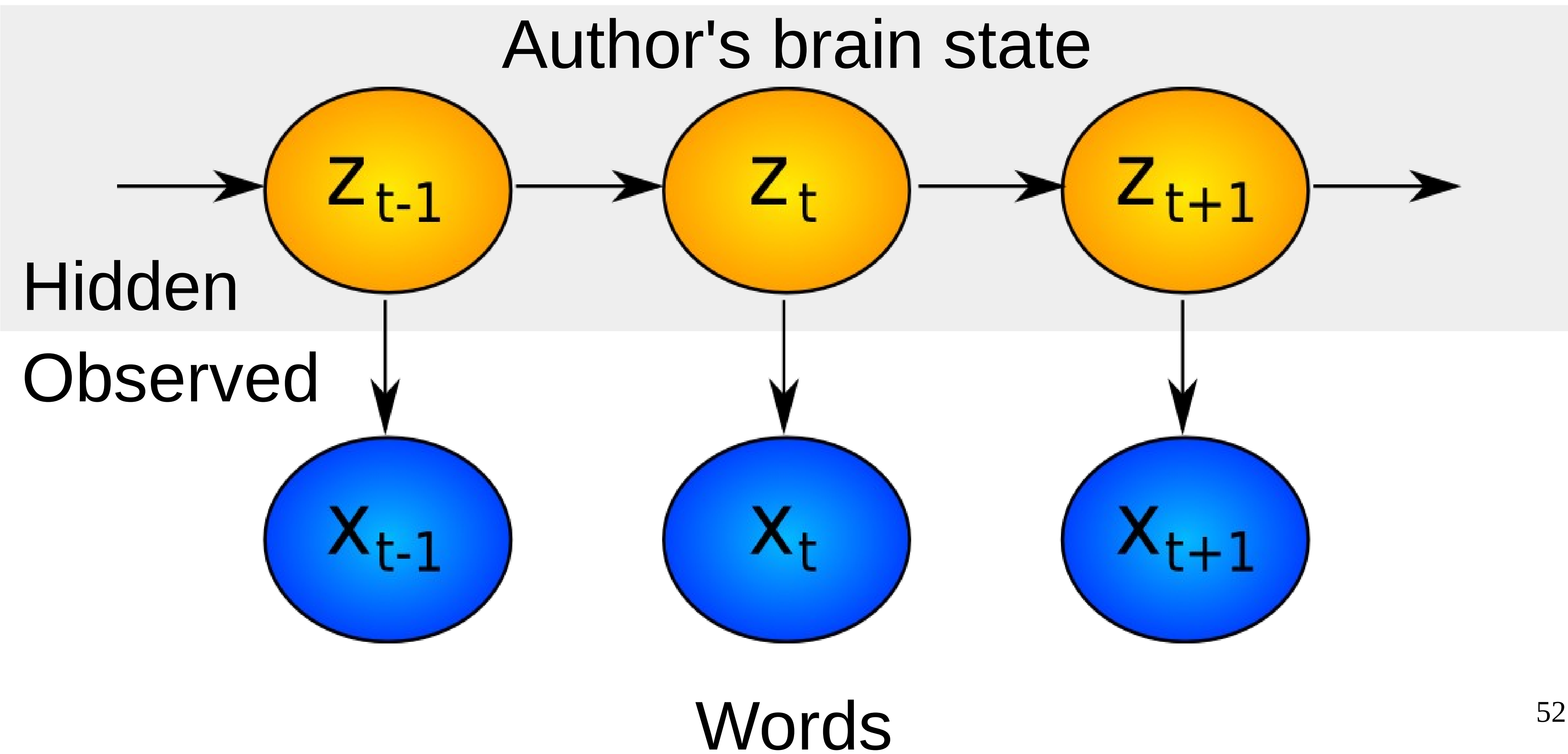
Can we learn* arbitrarily long dependencies?

* without infinitely many parameters

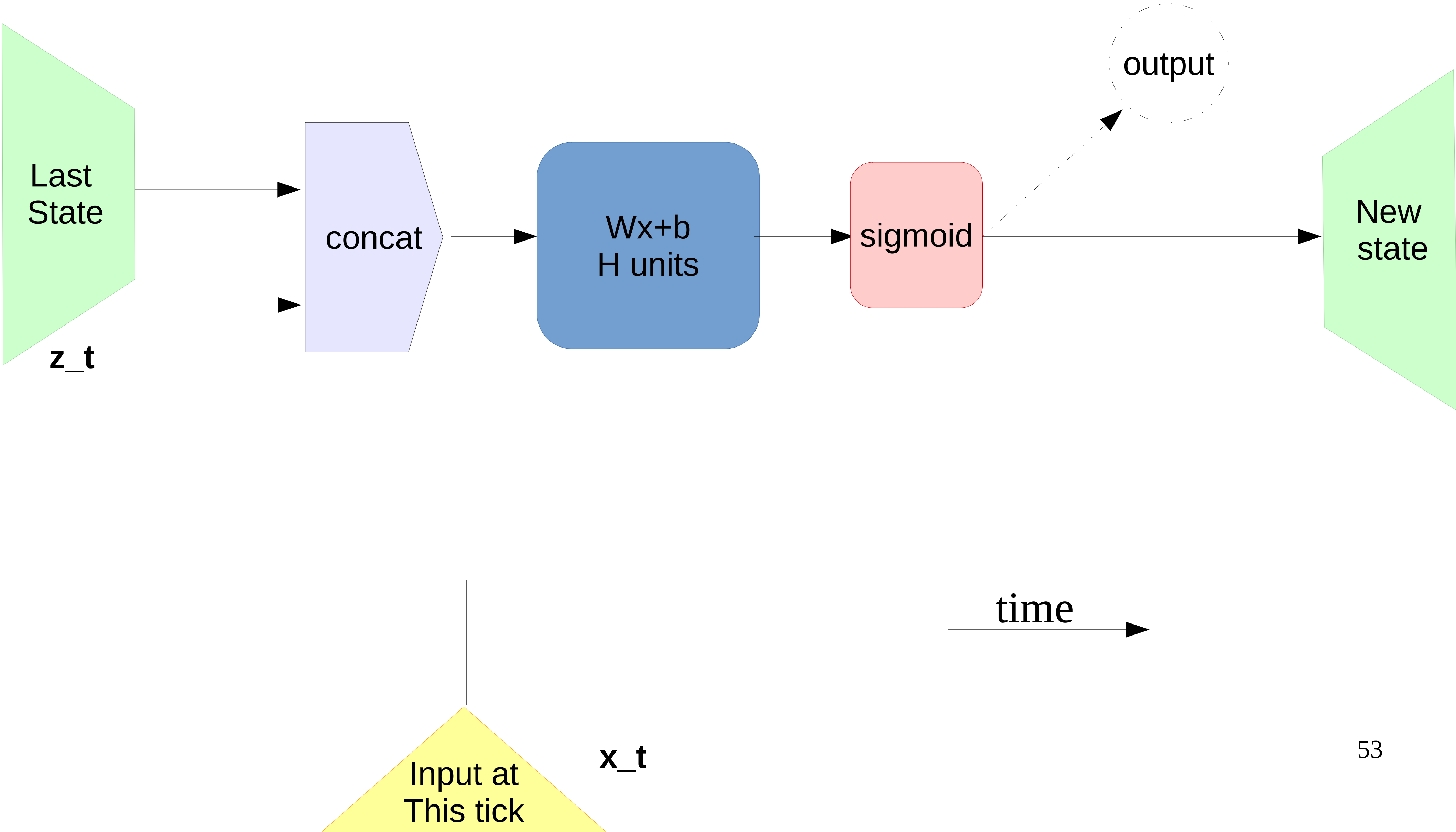
Hidden Markov Models: what's hidden



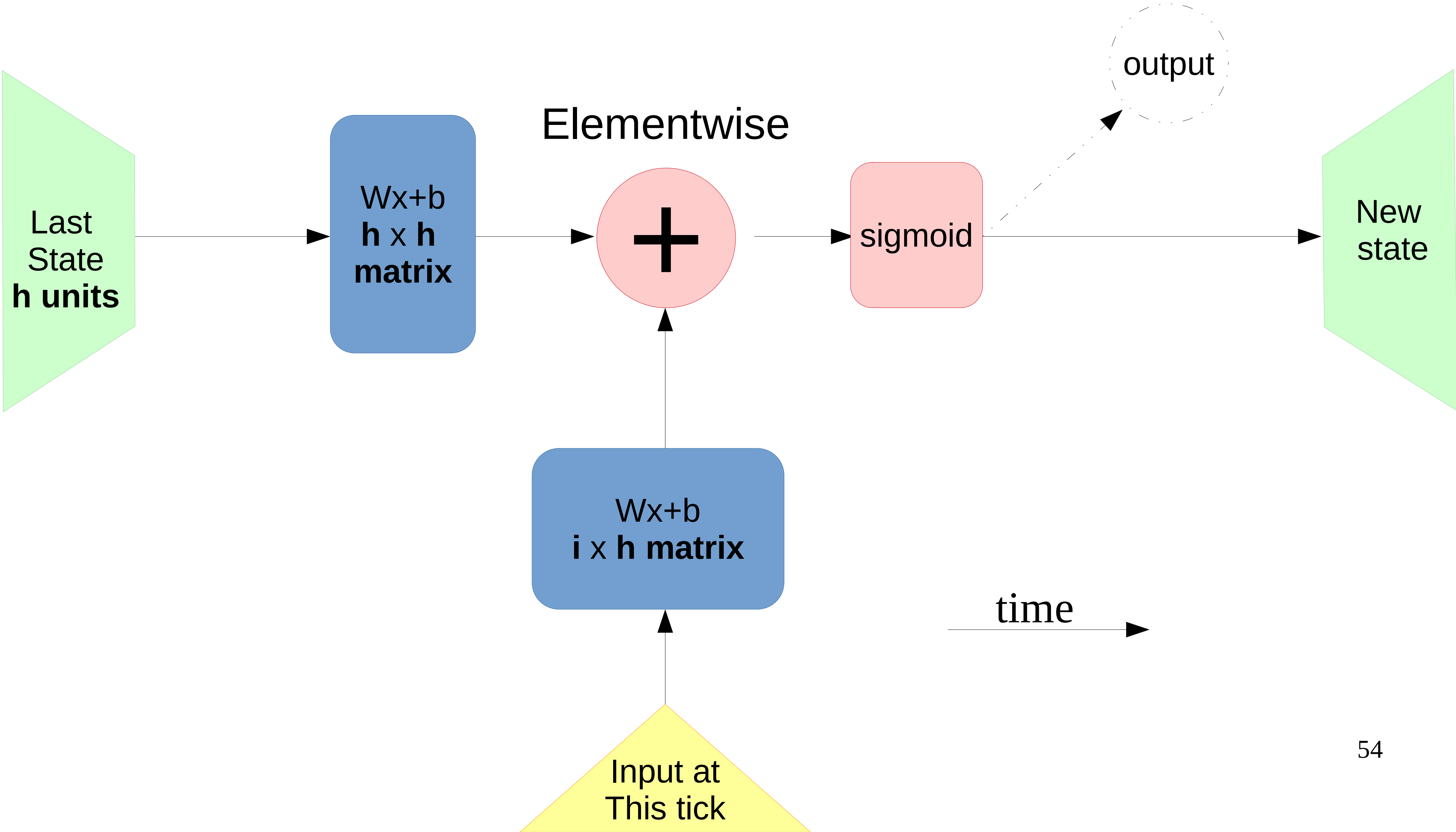
Hidden Markov Models: what is hidden



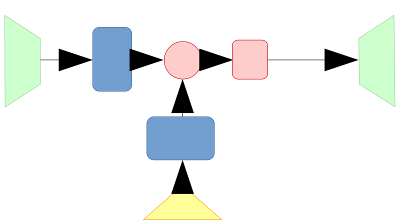
Recurrent neural network: one step



Recurrent neural network: one step

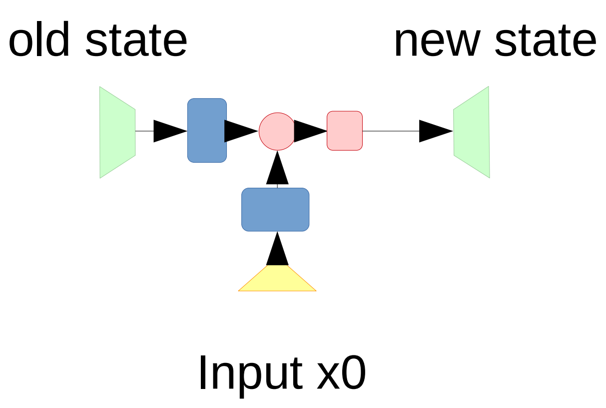


Recurrent neural network

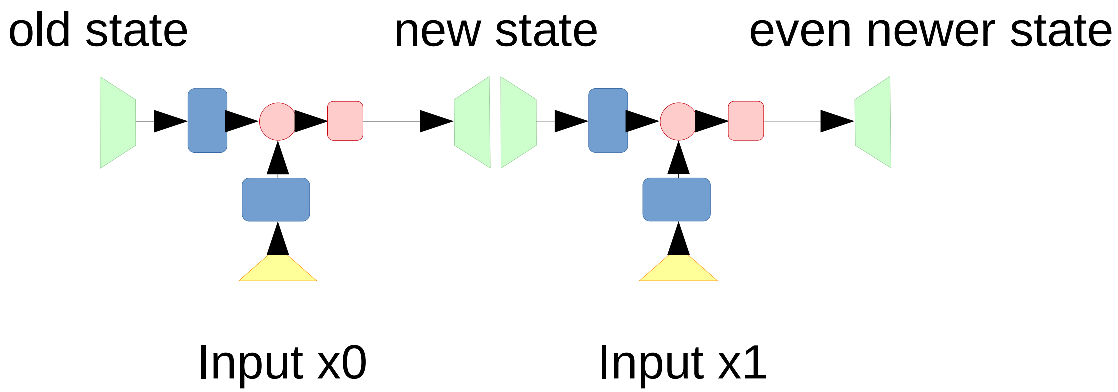


Zoom-out
of previous slide

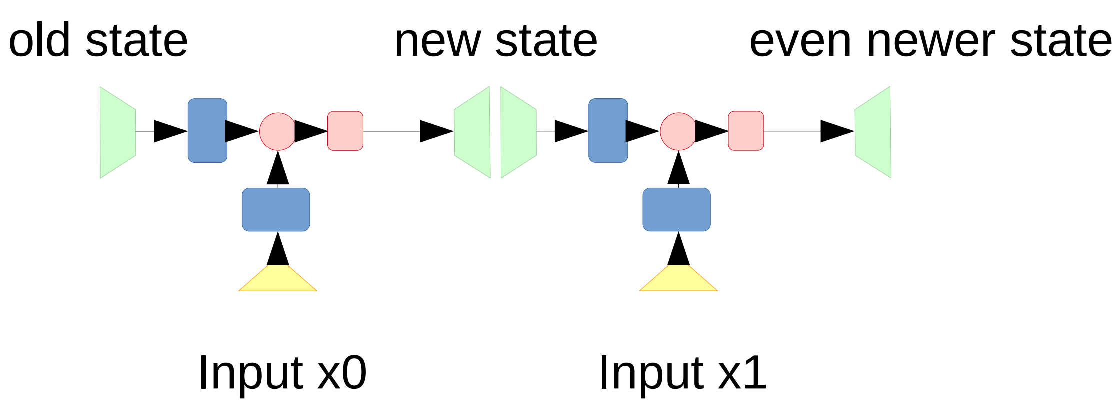
Recurrent neural network



Recurrent neural network

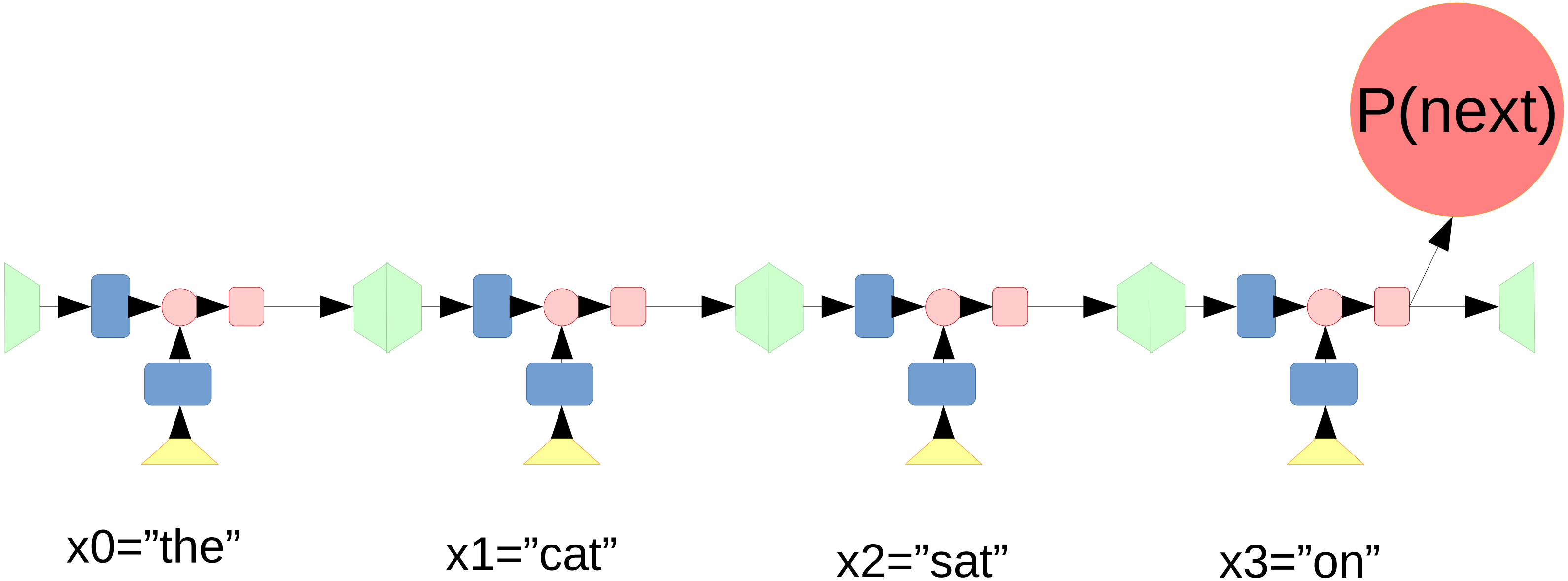


Recurrent neural network

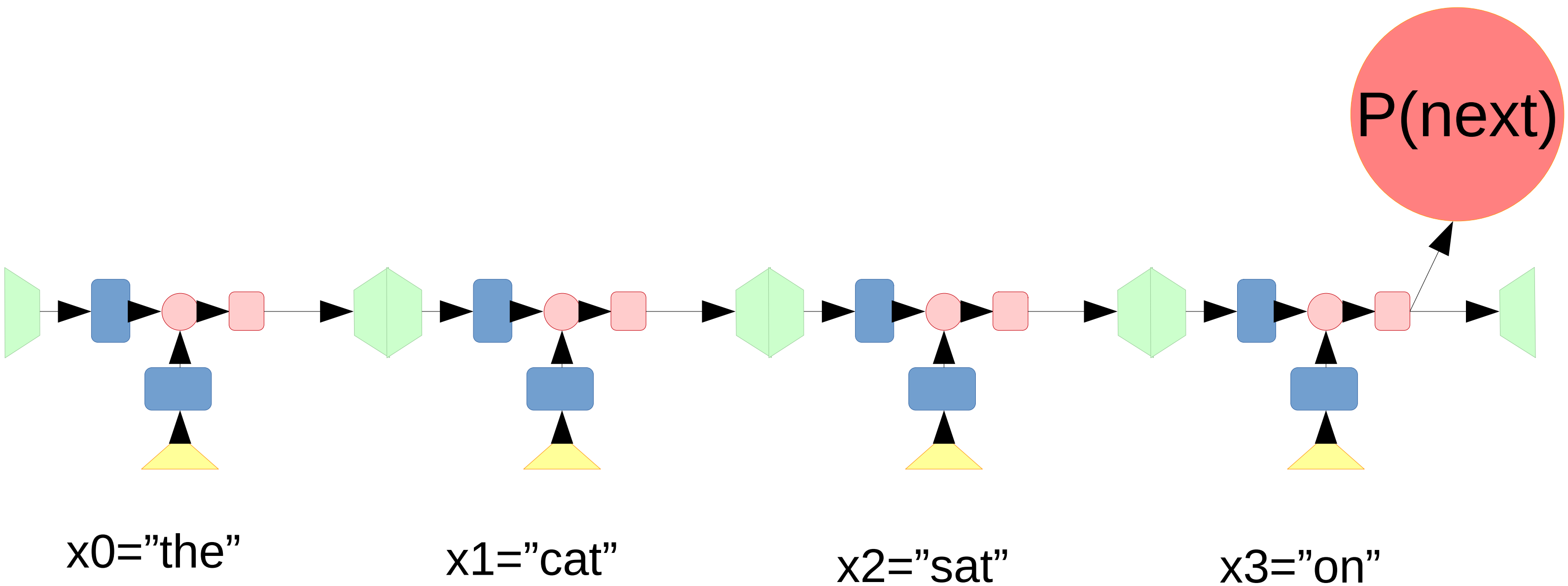


We use **same weight matrices** for all steps

Recurrent neural network

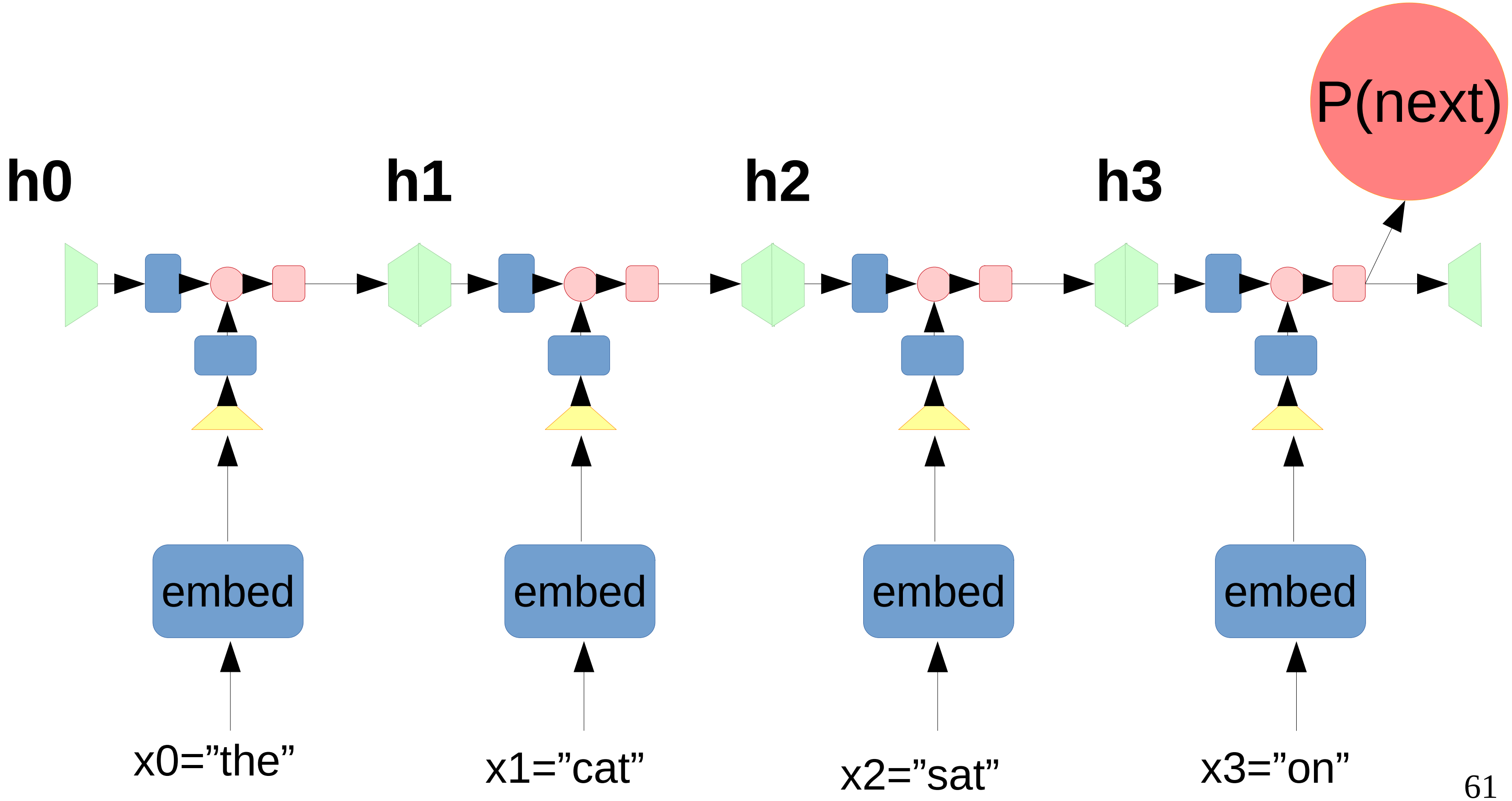


Recurrent neural network



How can we represent words?

Recurrent neural network

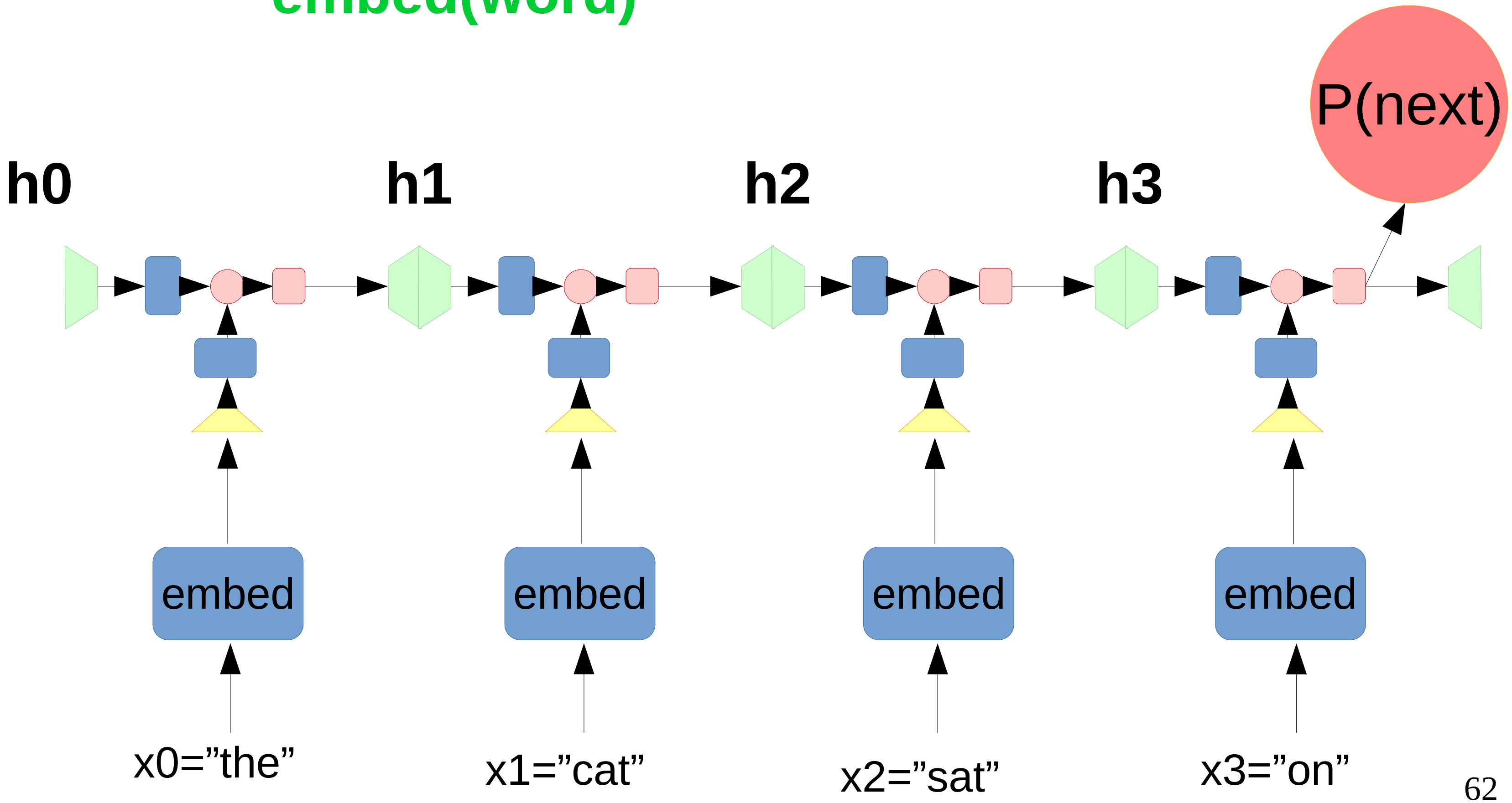


Recurrent neural network

$$h_0 = \bar{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

embed(word)

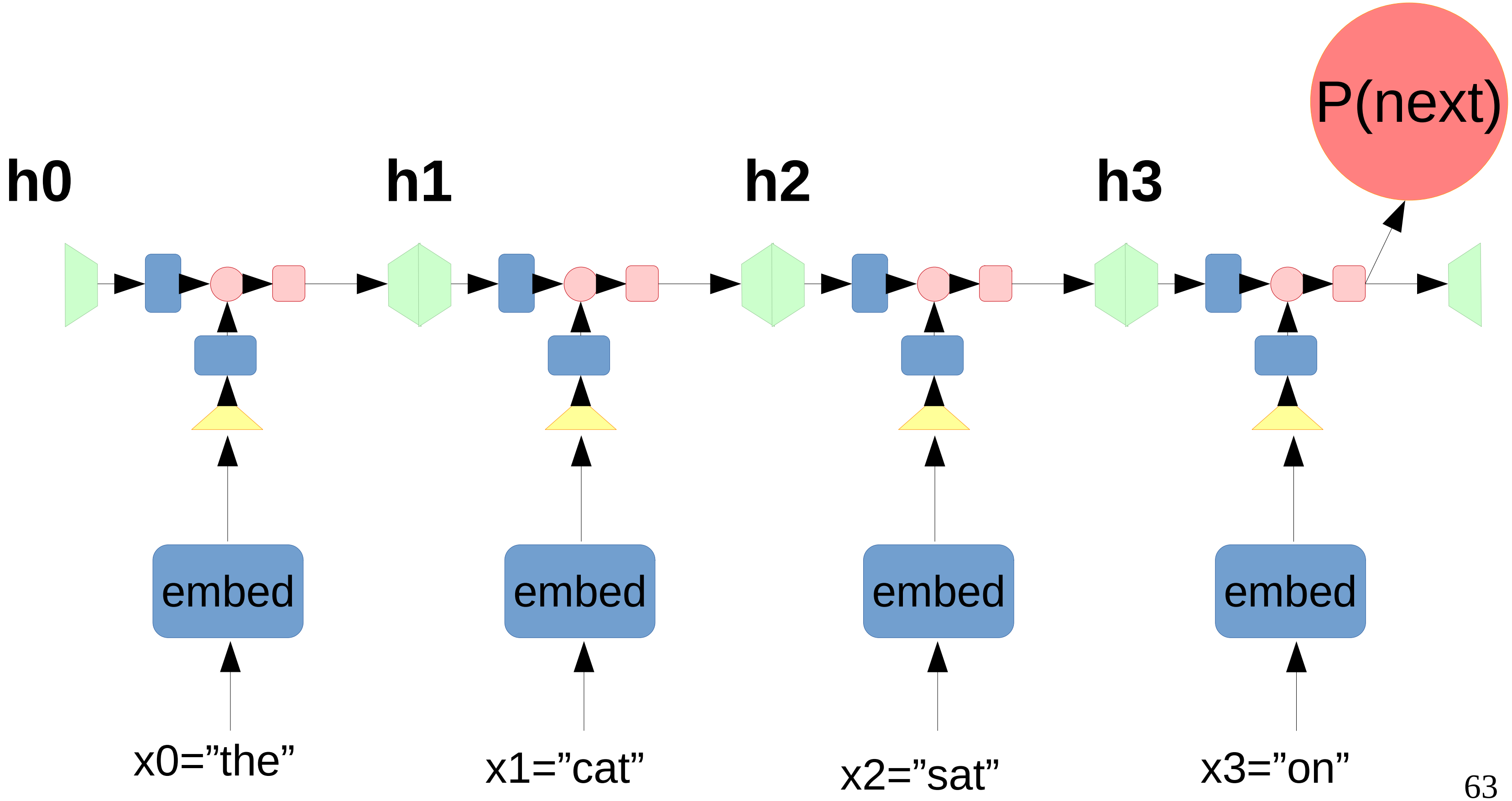


Recurrent neural network

$$h_0 = \bar{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

$$h_2 = ?$$



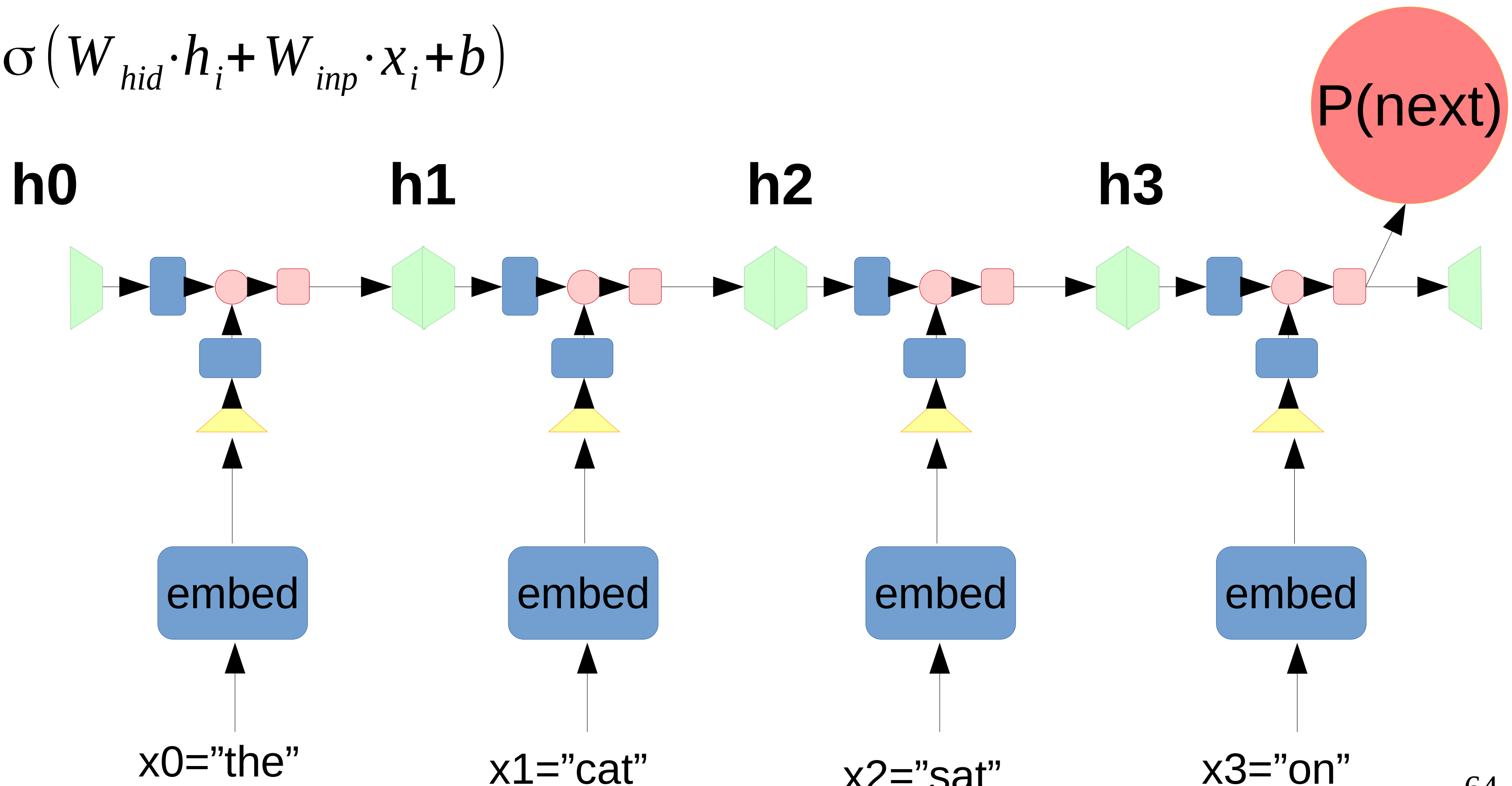
Recurrent neural network

$$h_0 = \bar{0}$$

$$h_1 = \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b)$$

$$h_2 = \sigma(W_{hid} \cdot h_1 + W_{inp} \cdot x_1 + b) = \sigma(W_{hid} \cdot \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b) + W_{inp} \cdot x_1 + b)$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$



Recurrent neural network

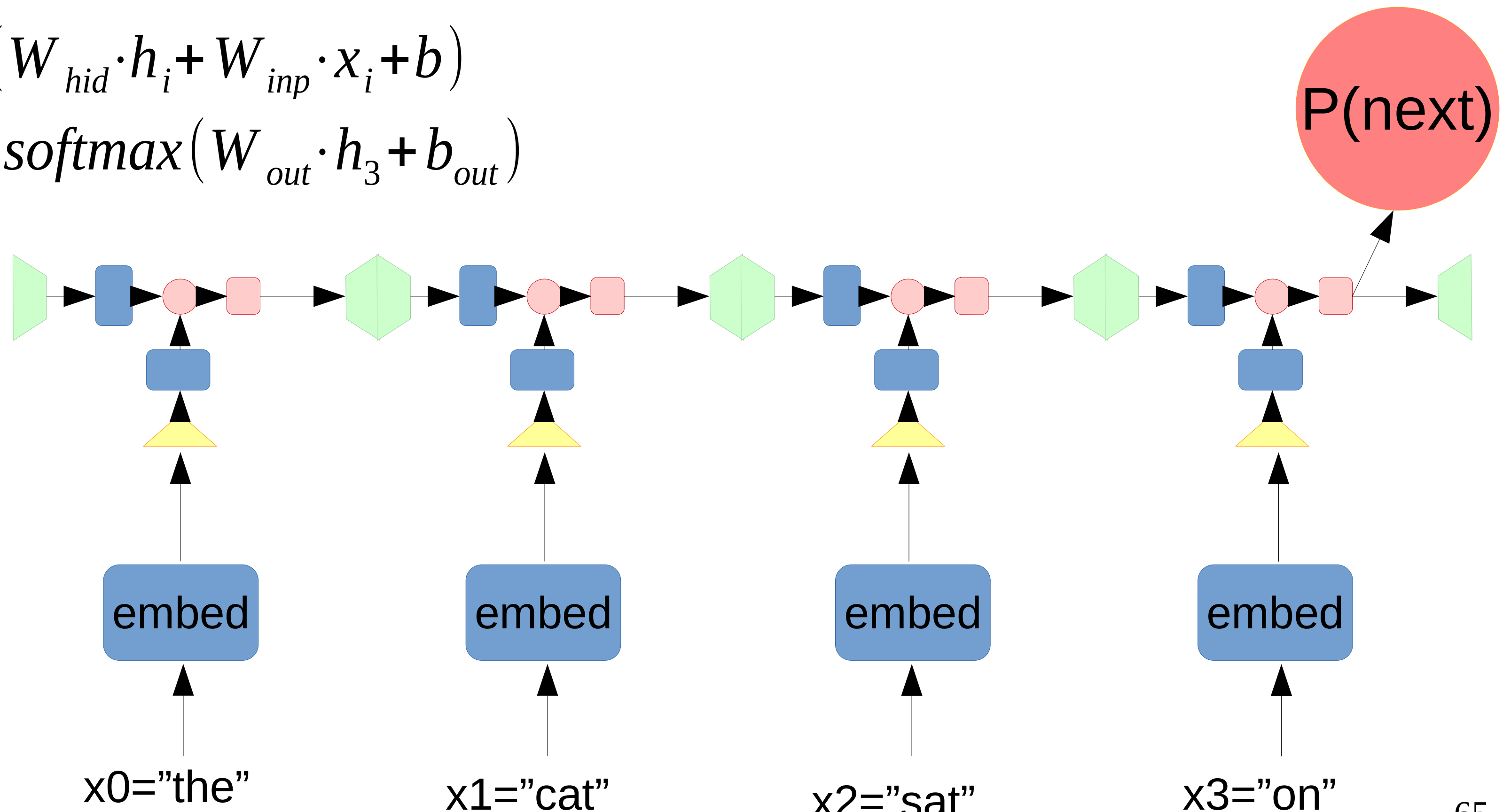
$$h_0 = \bar{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

$$h_2 = \sigma\left(W_{hid} \cdot h_1 + W_{inp} \cdot x_1 + b\right) = \sigma\left(W_{hid} \cdot \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right) + W_{inp} \cdot x_1 + b\right)$$

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

$$P\left(x_4\right) = softmax\left(W_{out} \cdot h_3 + b_{out}\right)$$

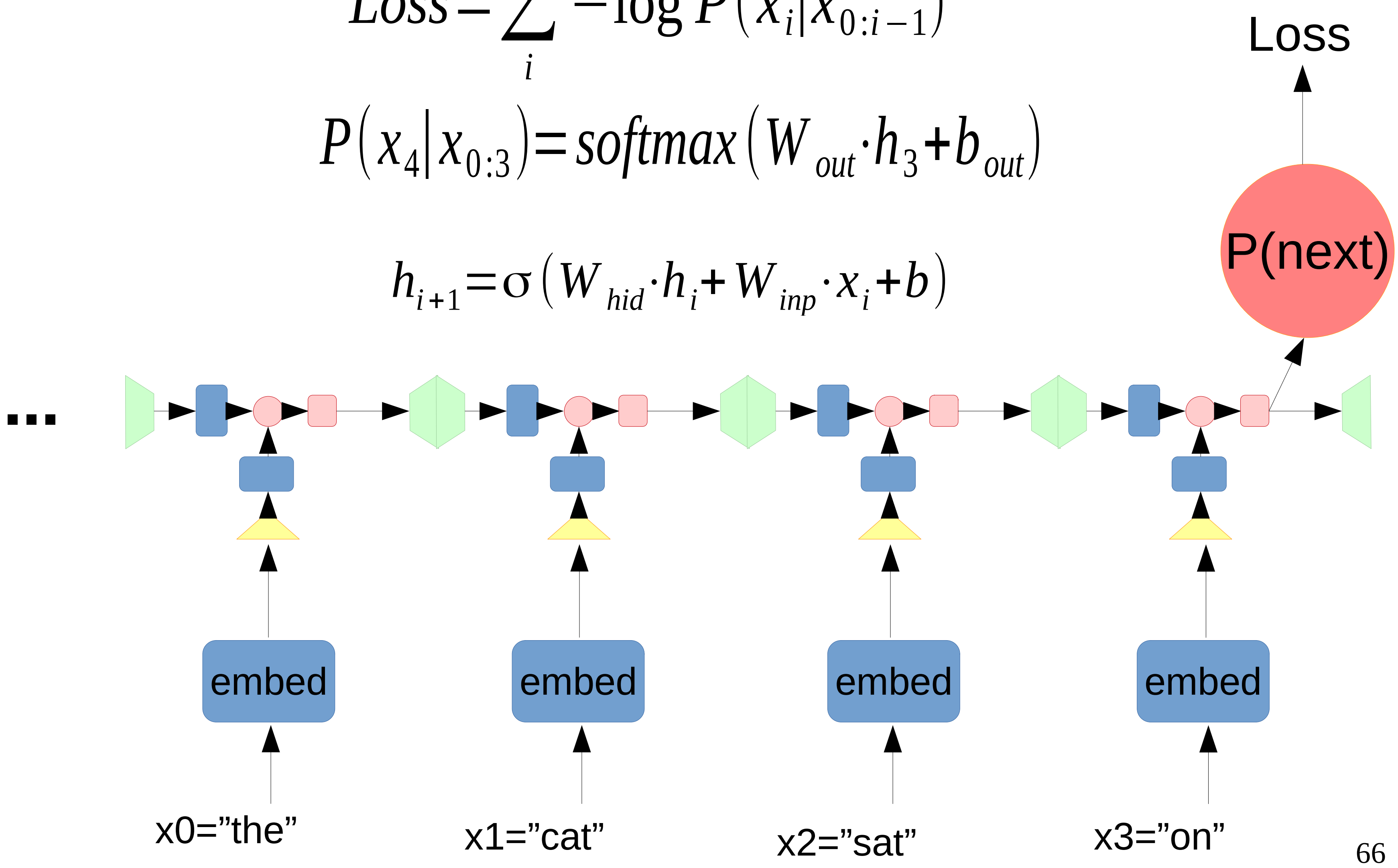


Recurrent neural network

$$Loss = \sum_i -\log P(x_i | x_{0:i-1})$$

$$P(x_4 | x_{0:3}) = softmax(W_{out} \cdot h_3 + b_{out})$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$



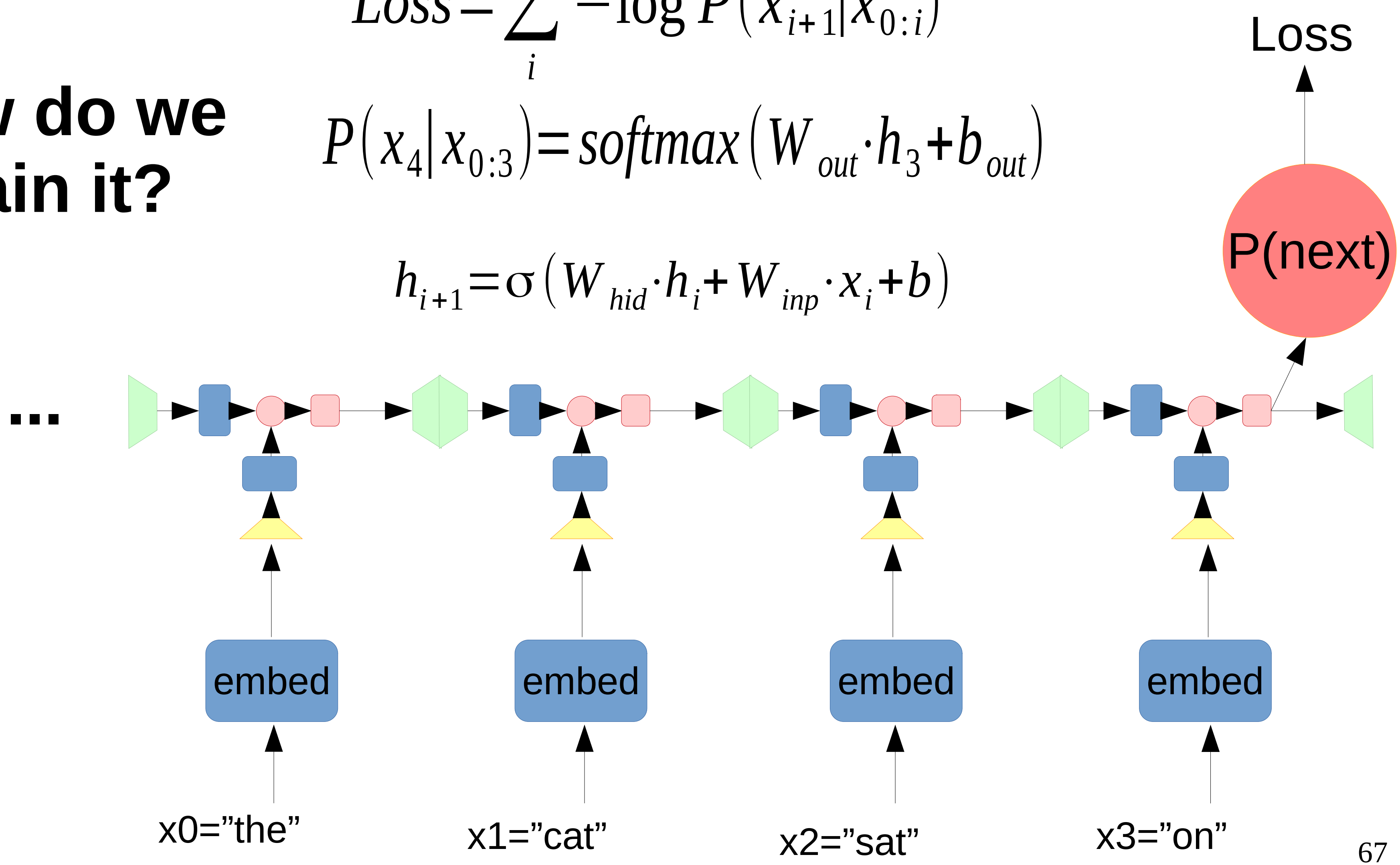
Recurrent neural network

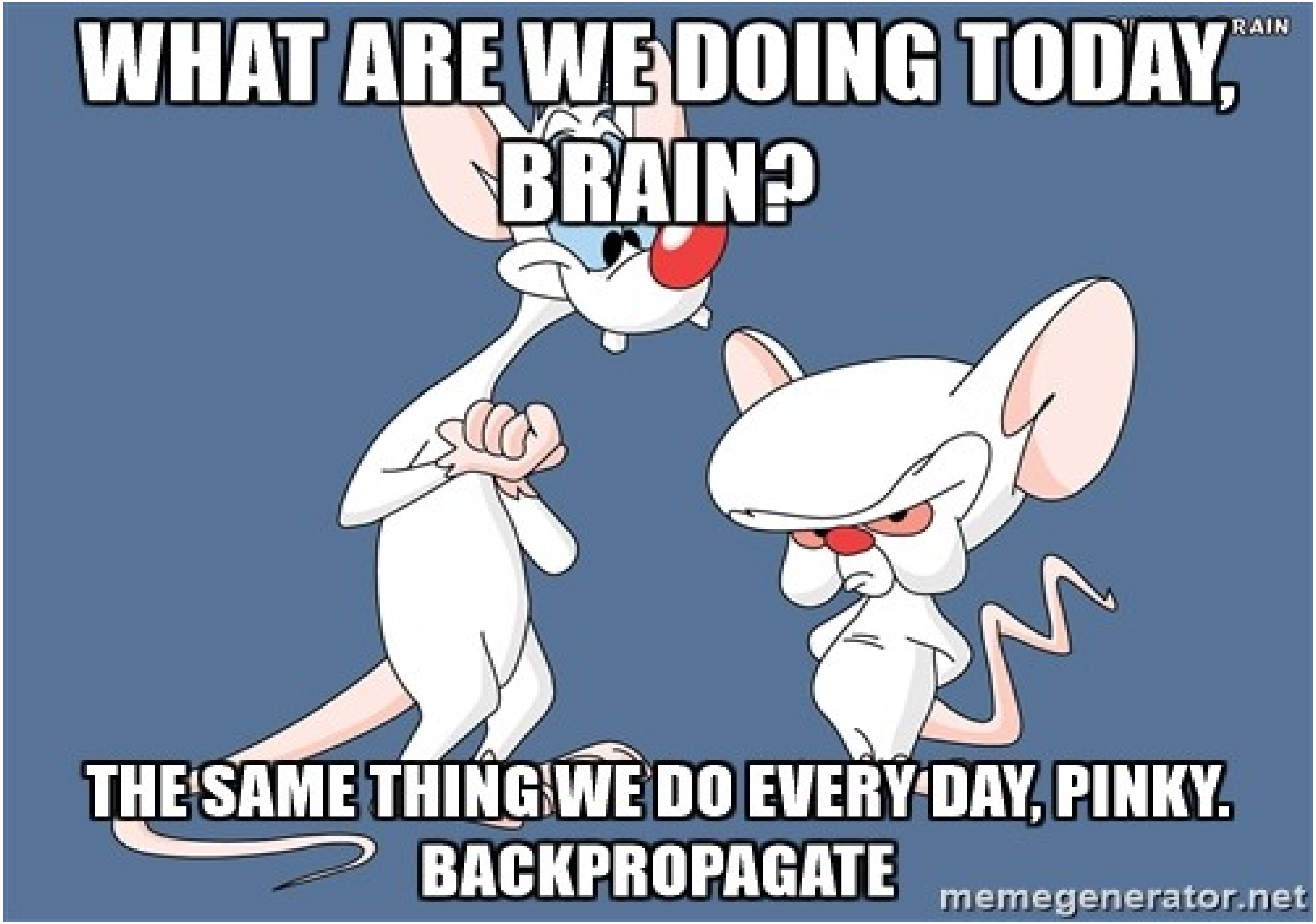
How do we train it?

$$Loss = \sum_i -\log P(x_{i+1} | x_{0:i})$$

$$P(x_4 | x_{0:3}) = softmax(W_{out} \cdot h_3 + b_{out})$$

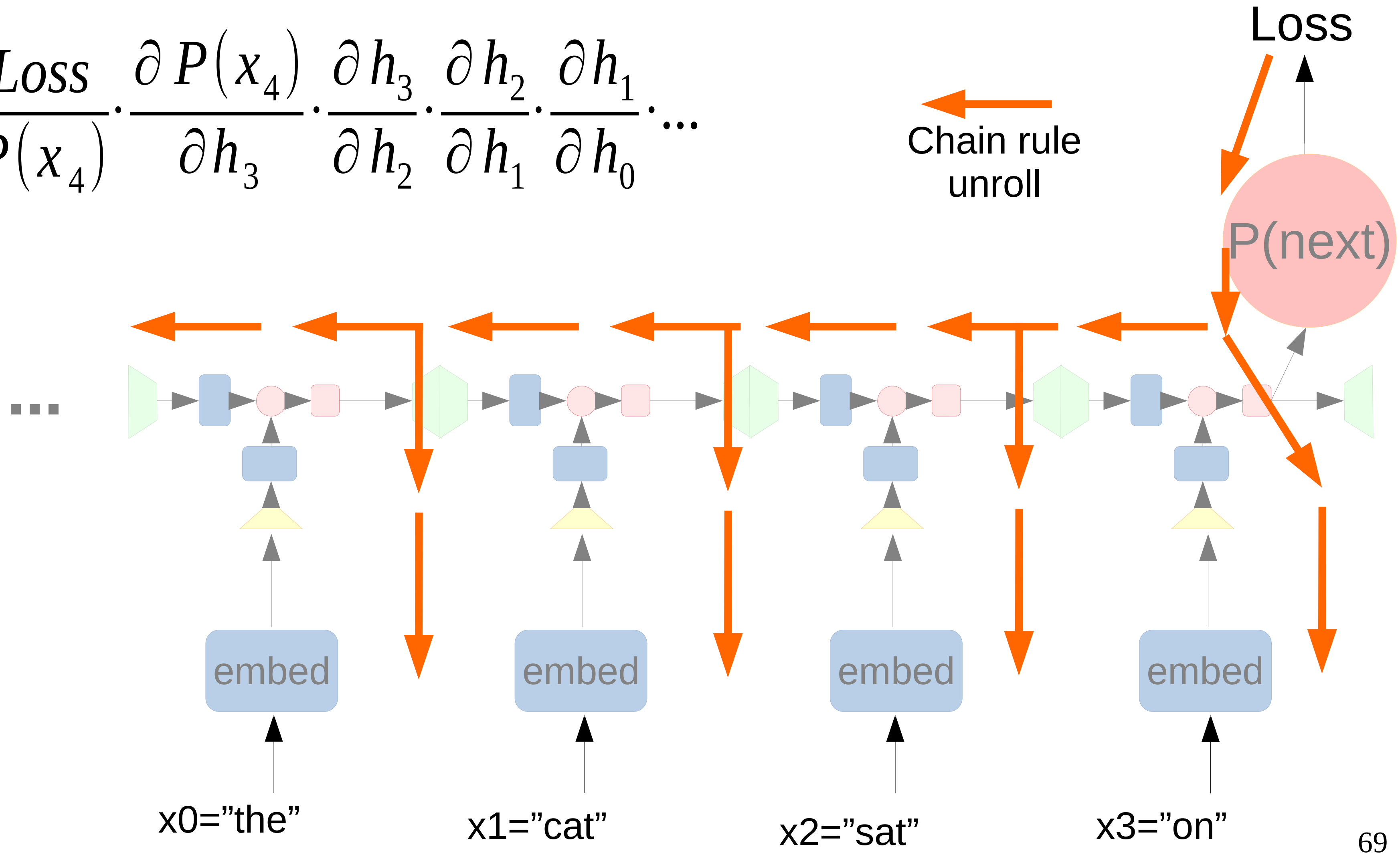
$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$




WHAT ARE WE DOING TODAY, BRAIN?
THE SAME THING WE DO EVERY DAY, PINKY. BACKPROPAGATE
memegenerator.net

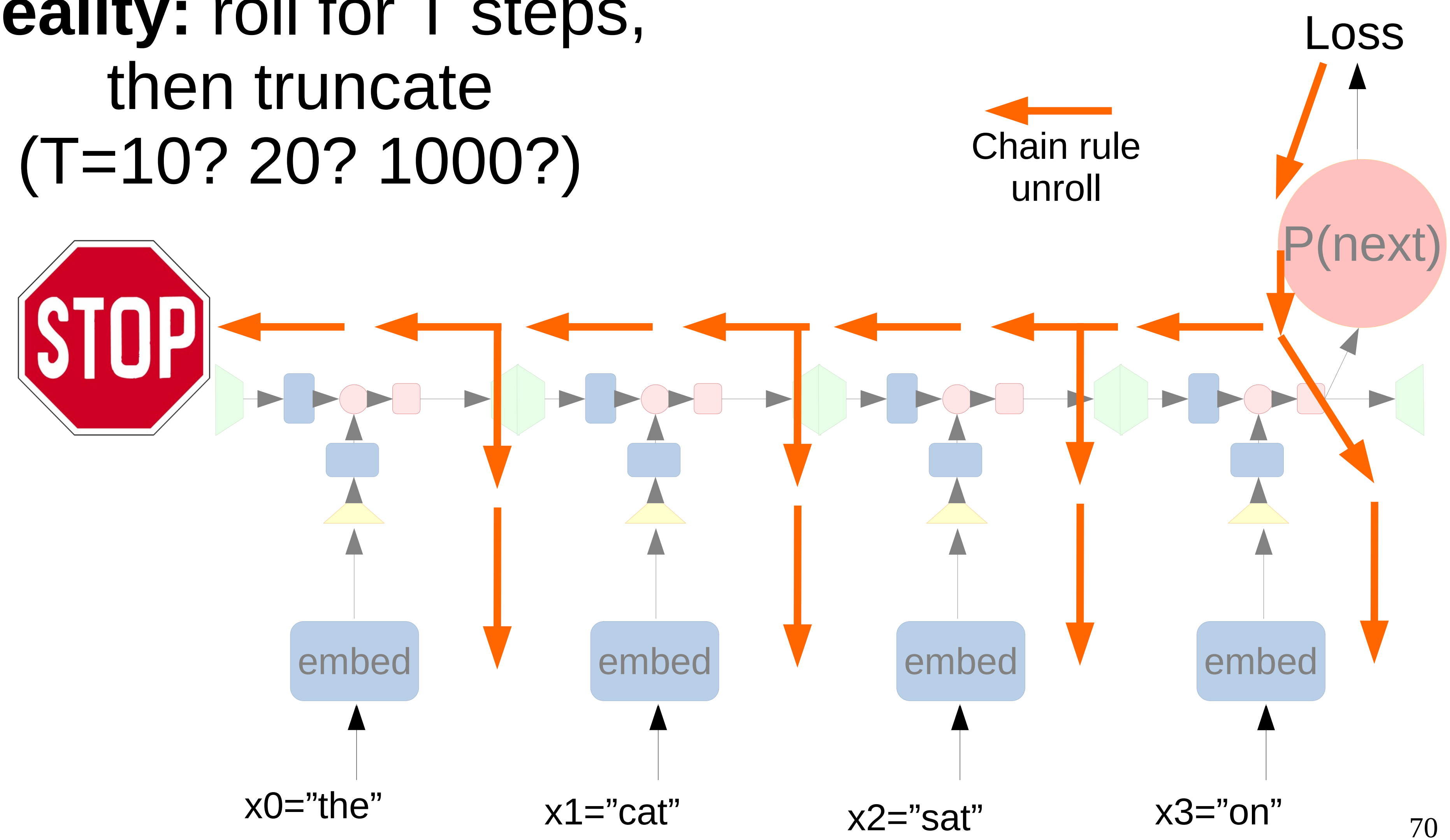
Backpropagation through time

$$\frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial h_0} \cdot \ldots$$



Truncated BPTT

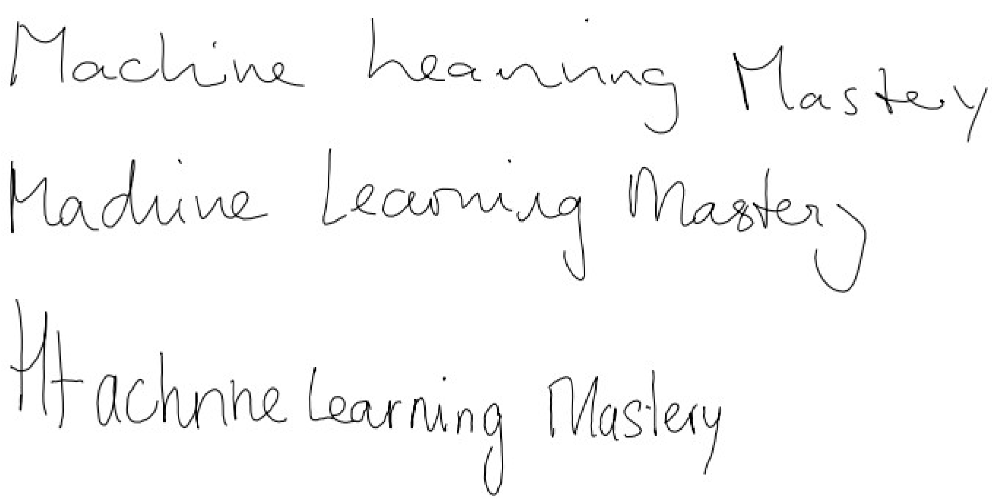
Reality: roll for T steps,
then truncate
(T=10? 20? 1000?)



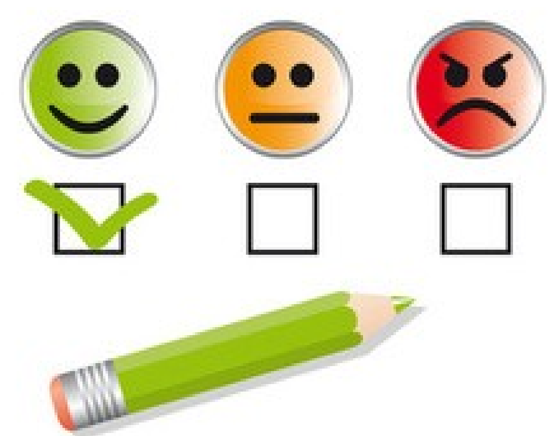
End of part 1

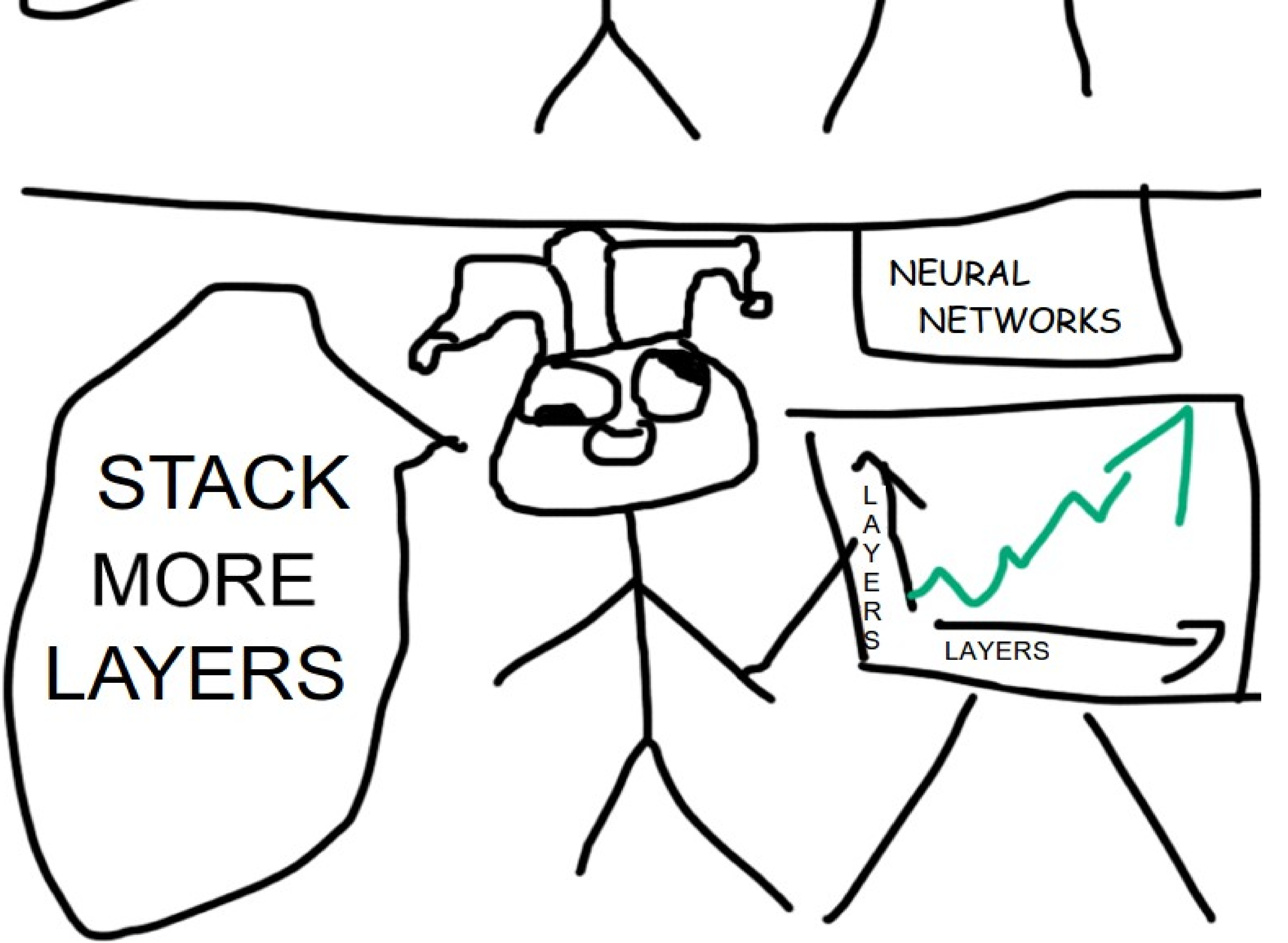
Questions for coffee break:

A) how would you apply RNN to generate random handwriting?

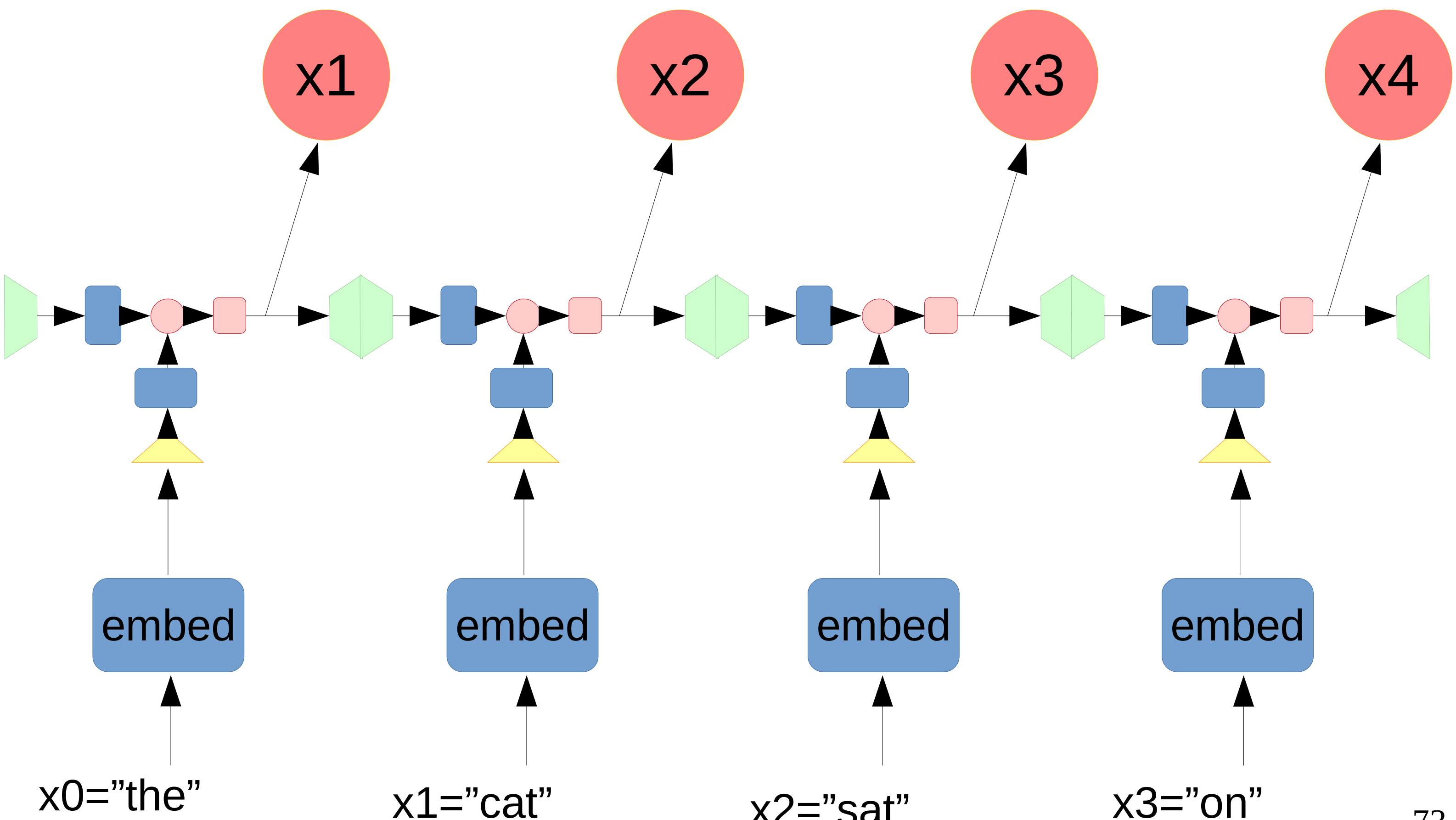


B) how would you apply RNN for sentiment classification?



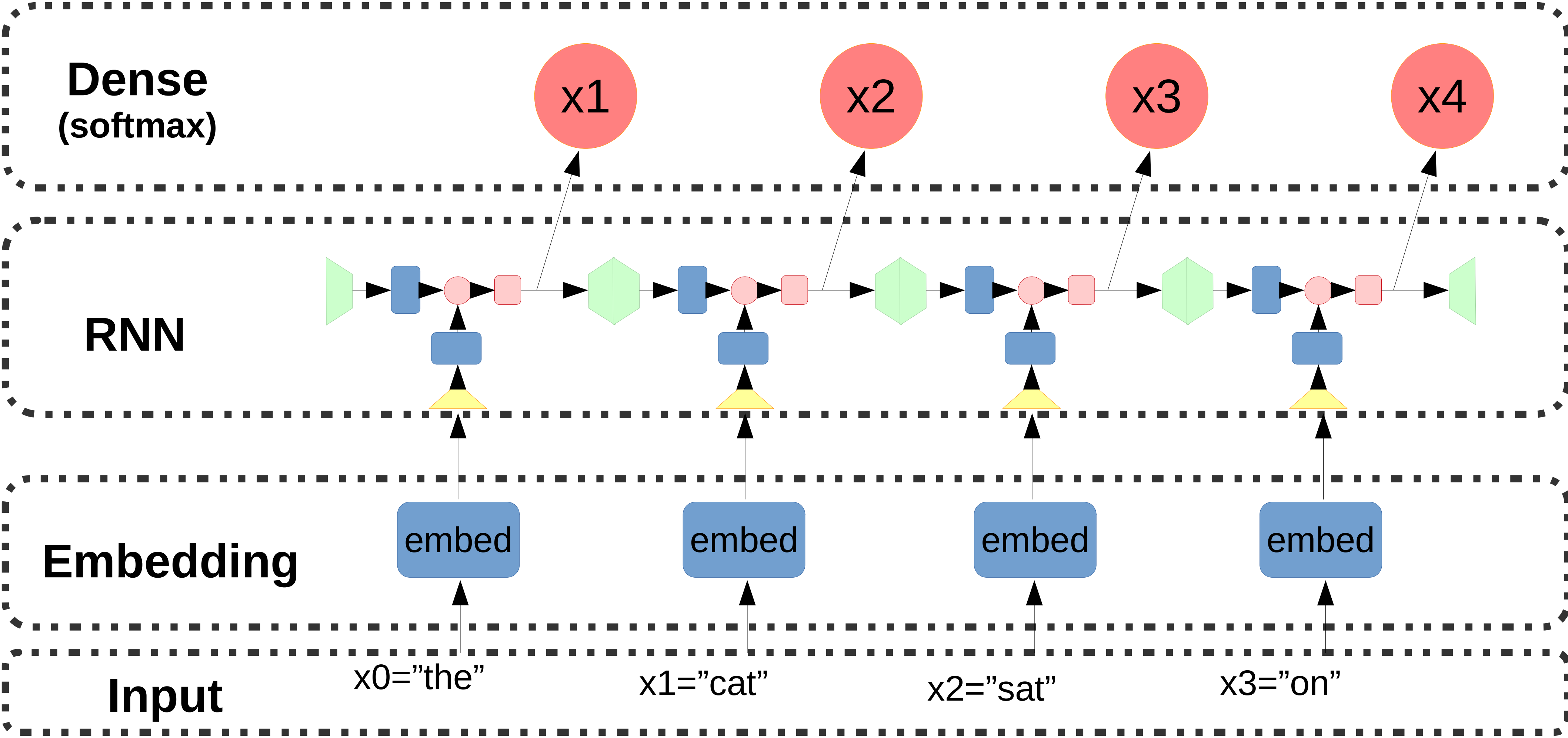

STACK MORE LAYERS
NEURAL NETWORKS
LAYERS
LAYERS

What is layer, again?

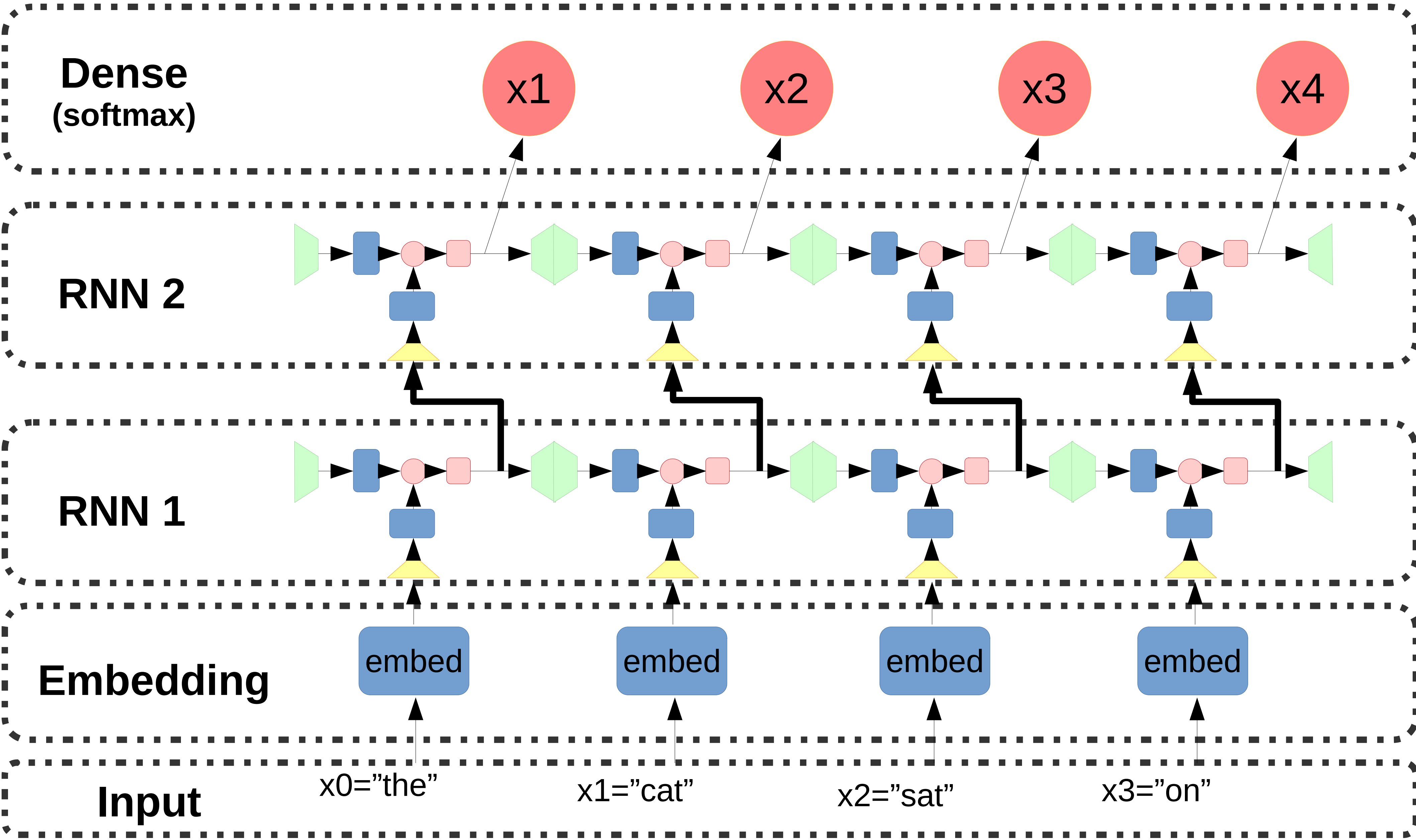


Layers

Where to stick more layers?

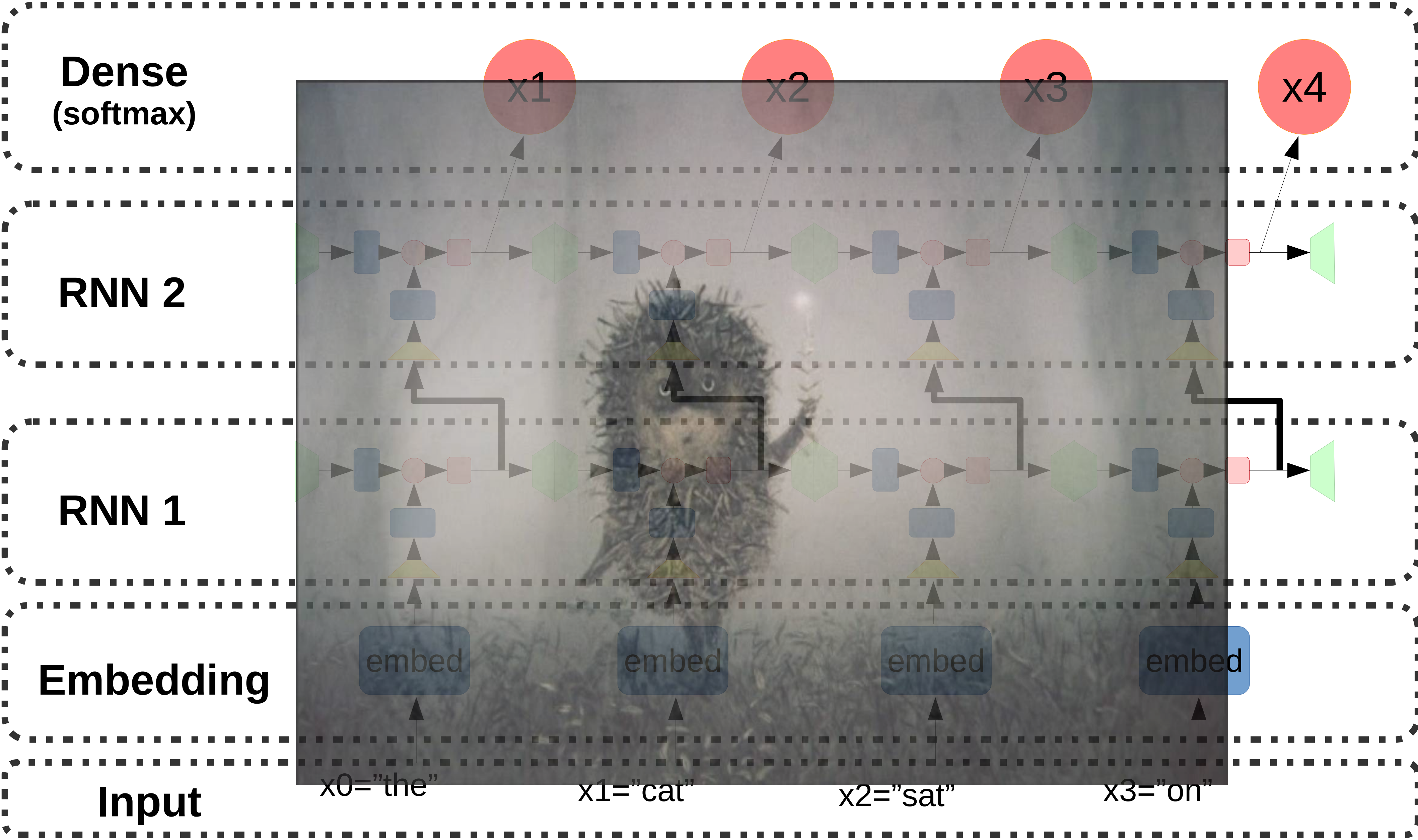


More layers

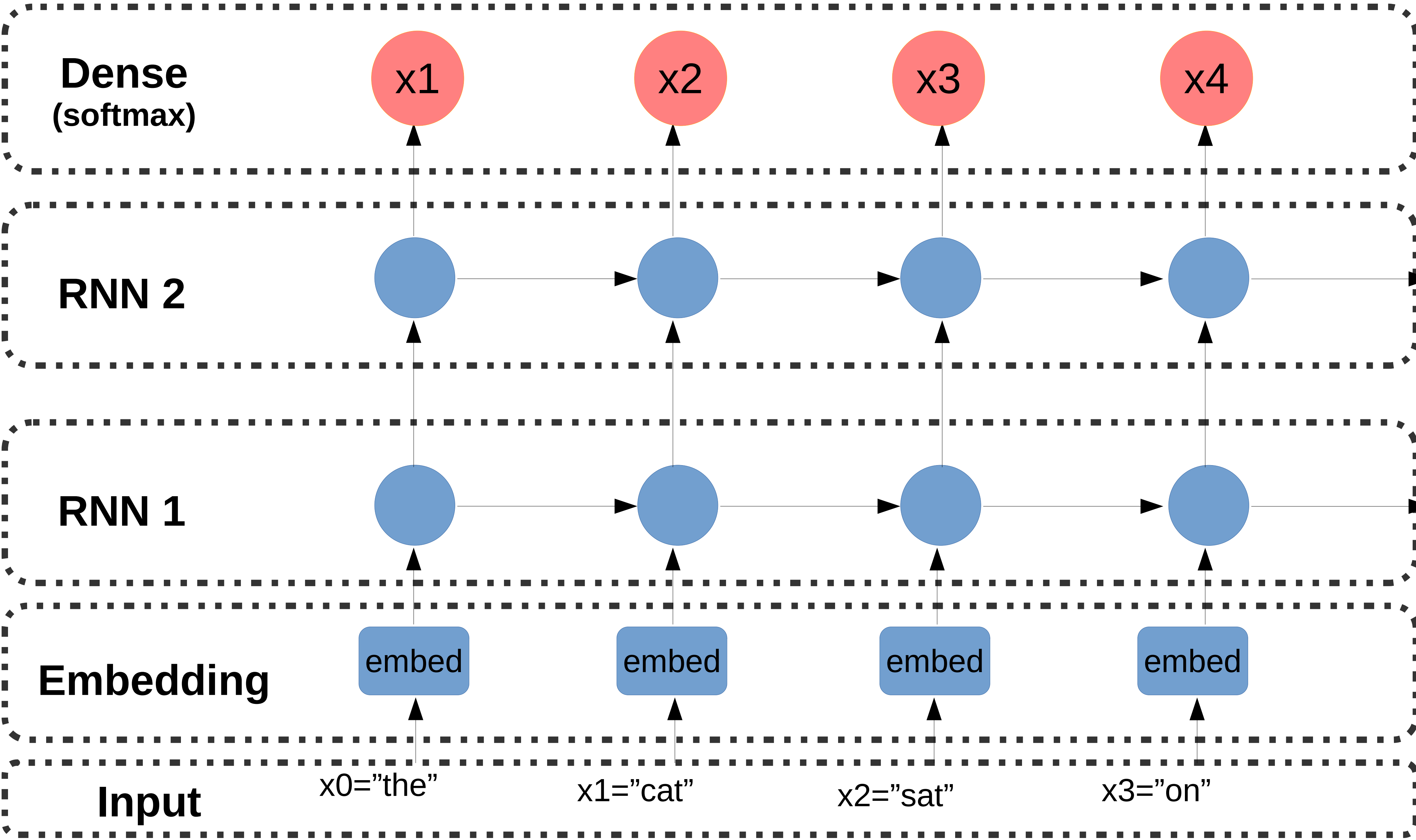


Dense
(softmax)
x1
x2
x3
x4
RNN 2
RNN 1
Embedding
embed
embed
embed
embed
Input
x0="the"
x1="cat"
x2="sat"
x3="on"

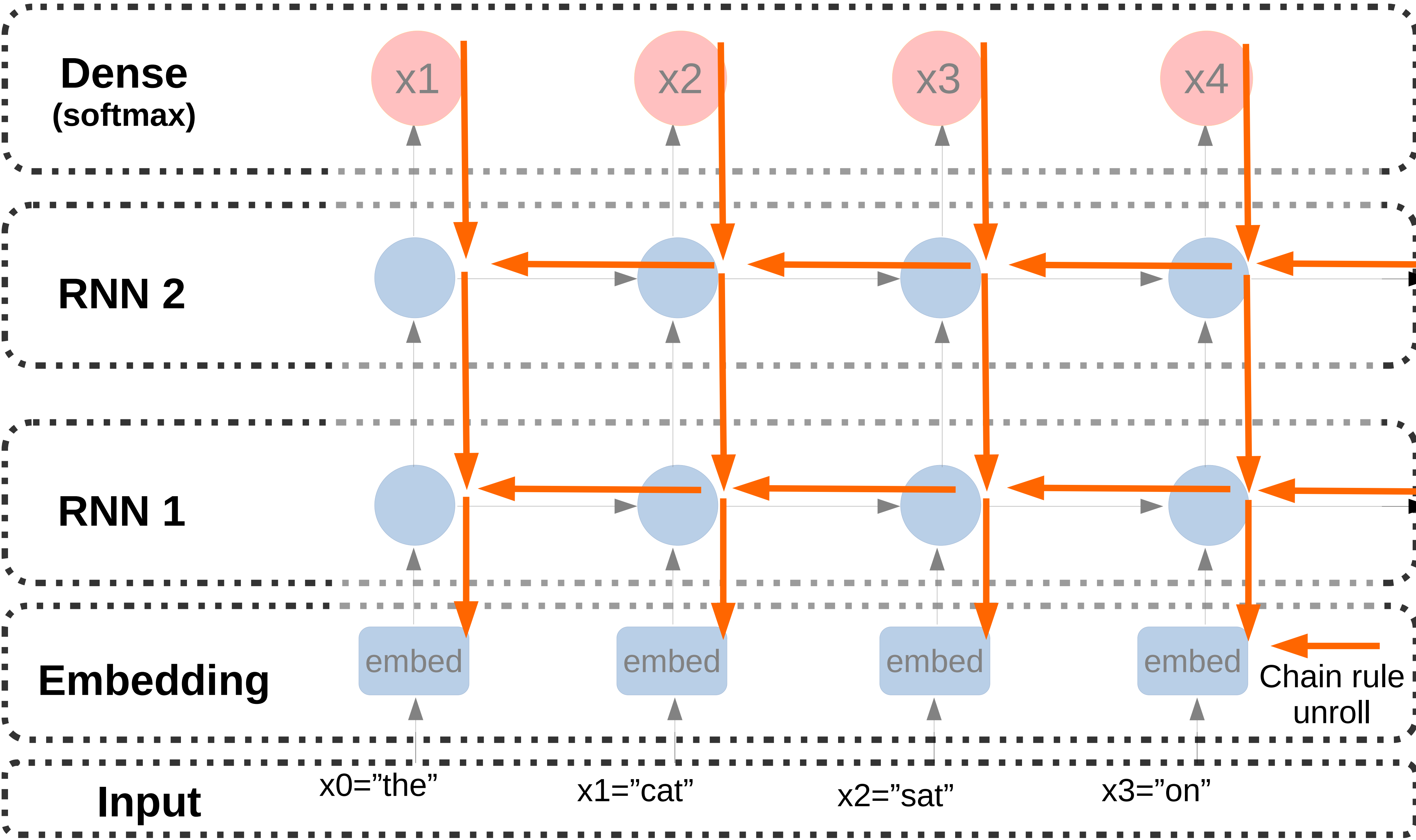
Too f**king complicated



2-layer RNN


Dense
(softmax)
x1
x2
x3
x4
RNN 2
RNN 1
Embedding
embed
embed
embed
embed
Input
x0=”the”
x1=”cat”
x2=”sat”
x3=”on”

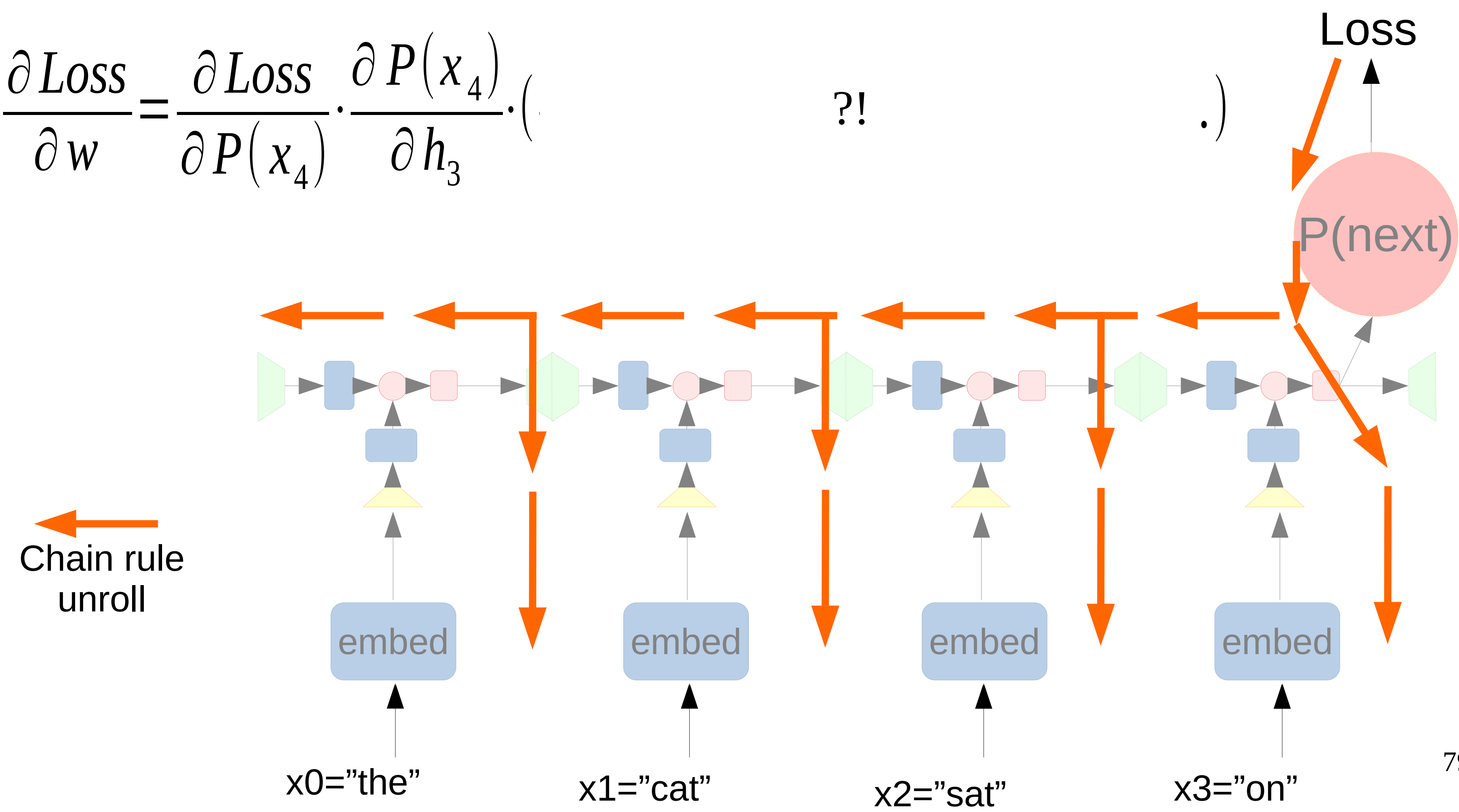
BPTT again


Dense
(softmax)
x1
x2
x3
x4
RNN 2
RNN 1
Embedding
embed
embed
embed
embed
Chain rule
unroll
Input
x0=”the”
x1=”cat”
x2=”sat”
x3=”on”

BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

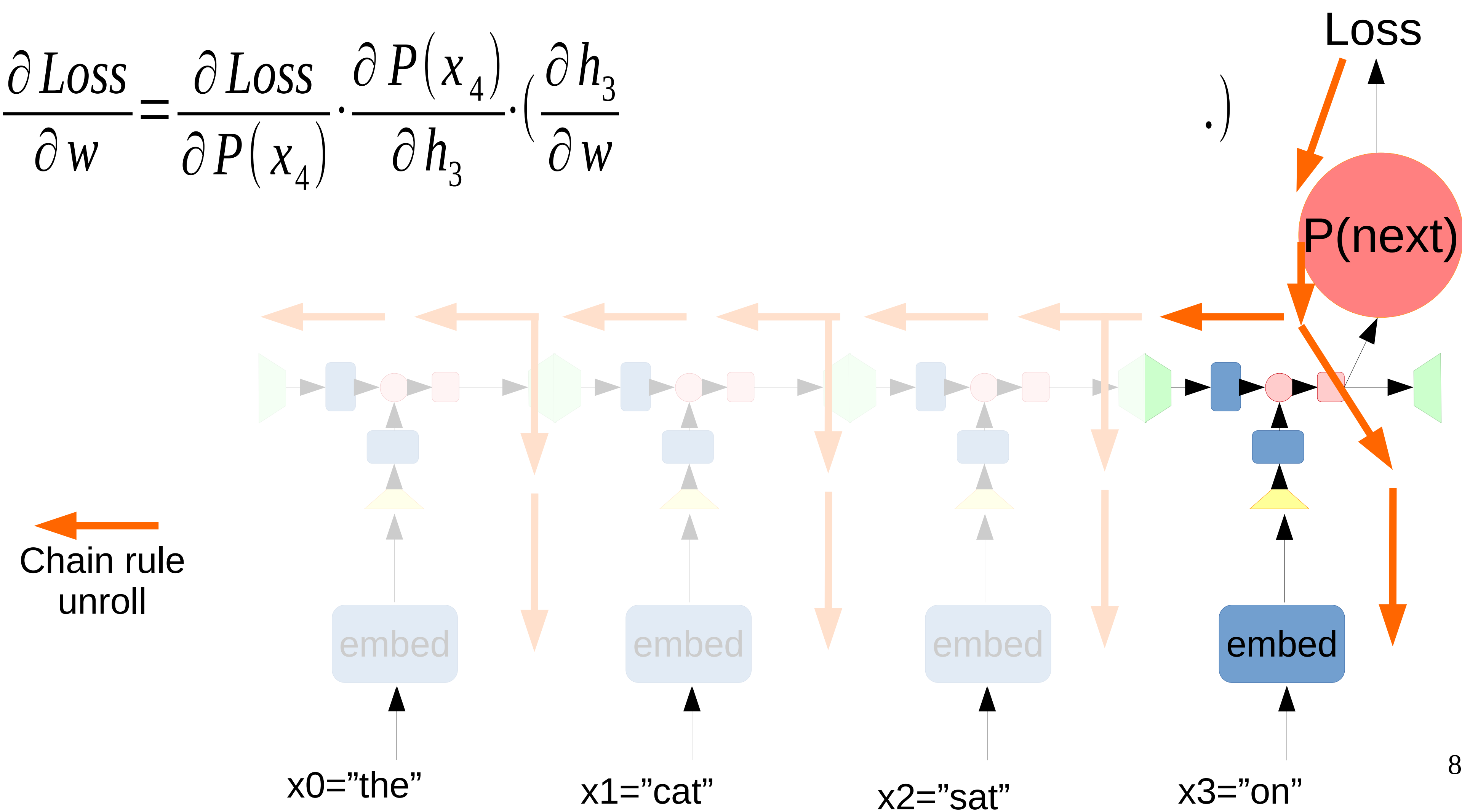
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\qquad ?! \qquad . \right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} \qquad \cdot\right)$$

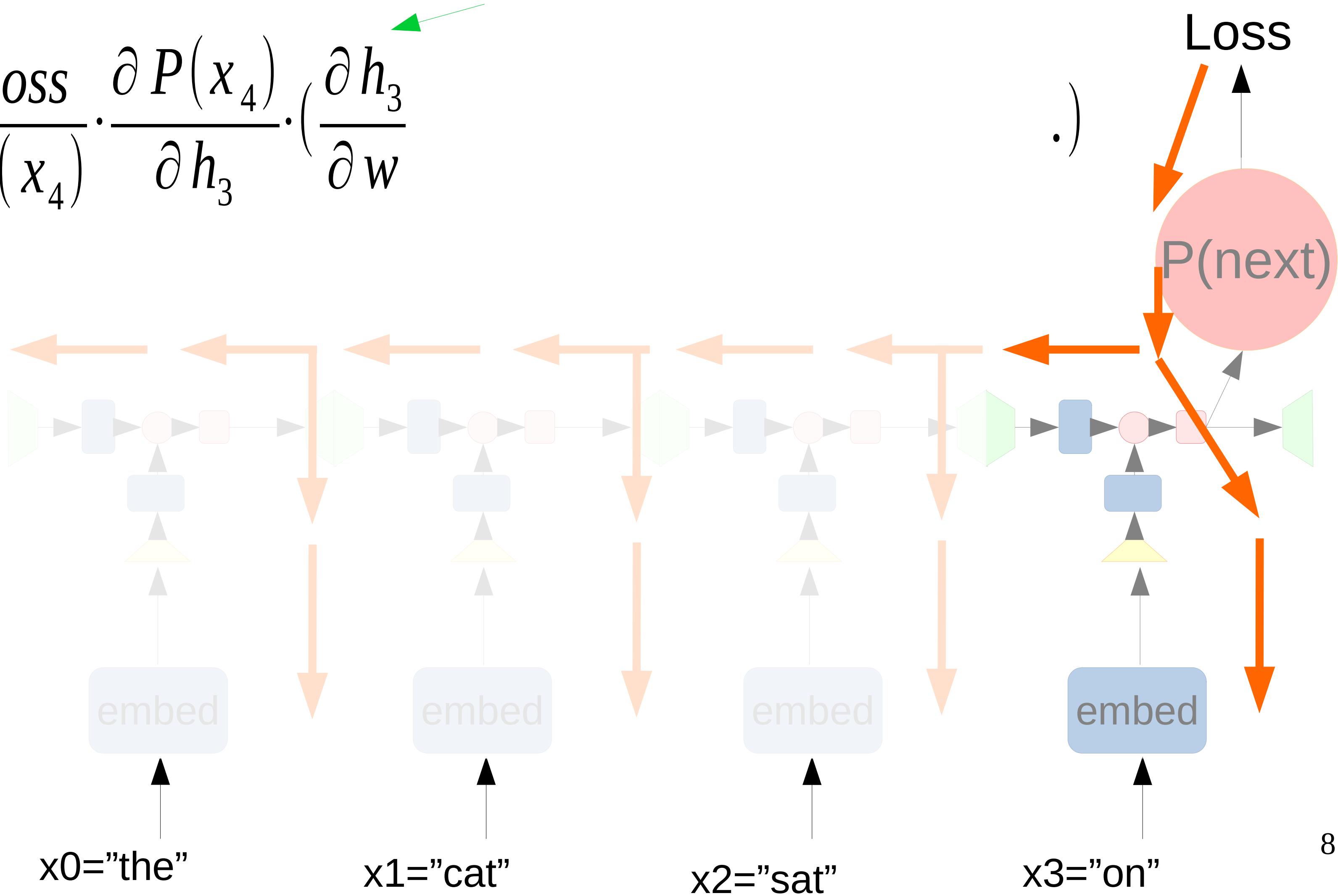


BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

consider h2 constant

$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} \qquad\qquad . \right)$$

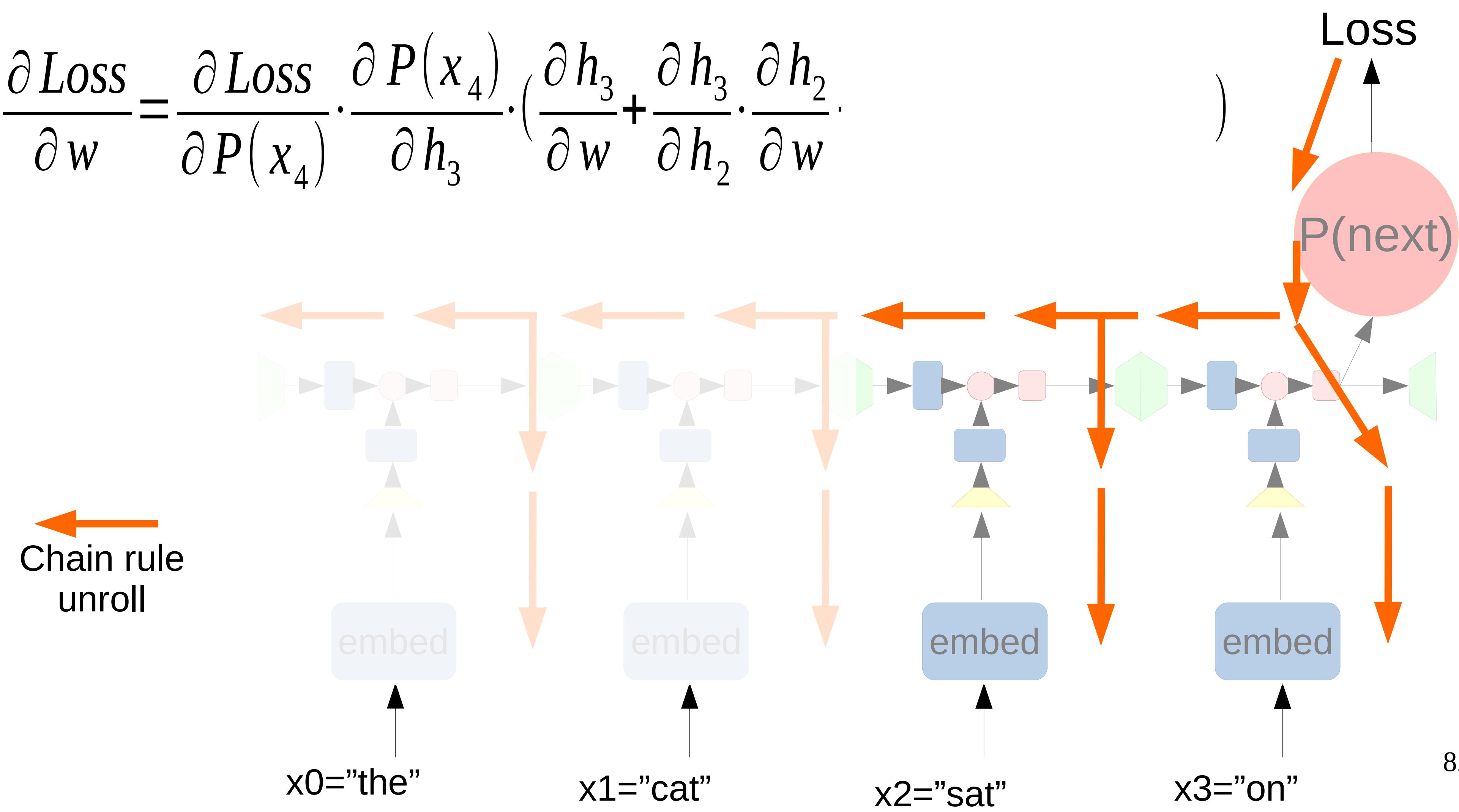


Chain rule unroll

BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

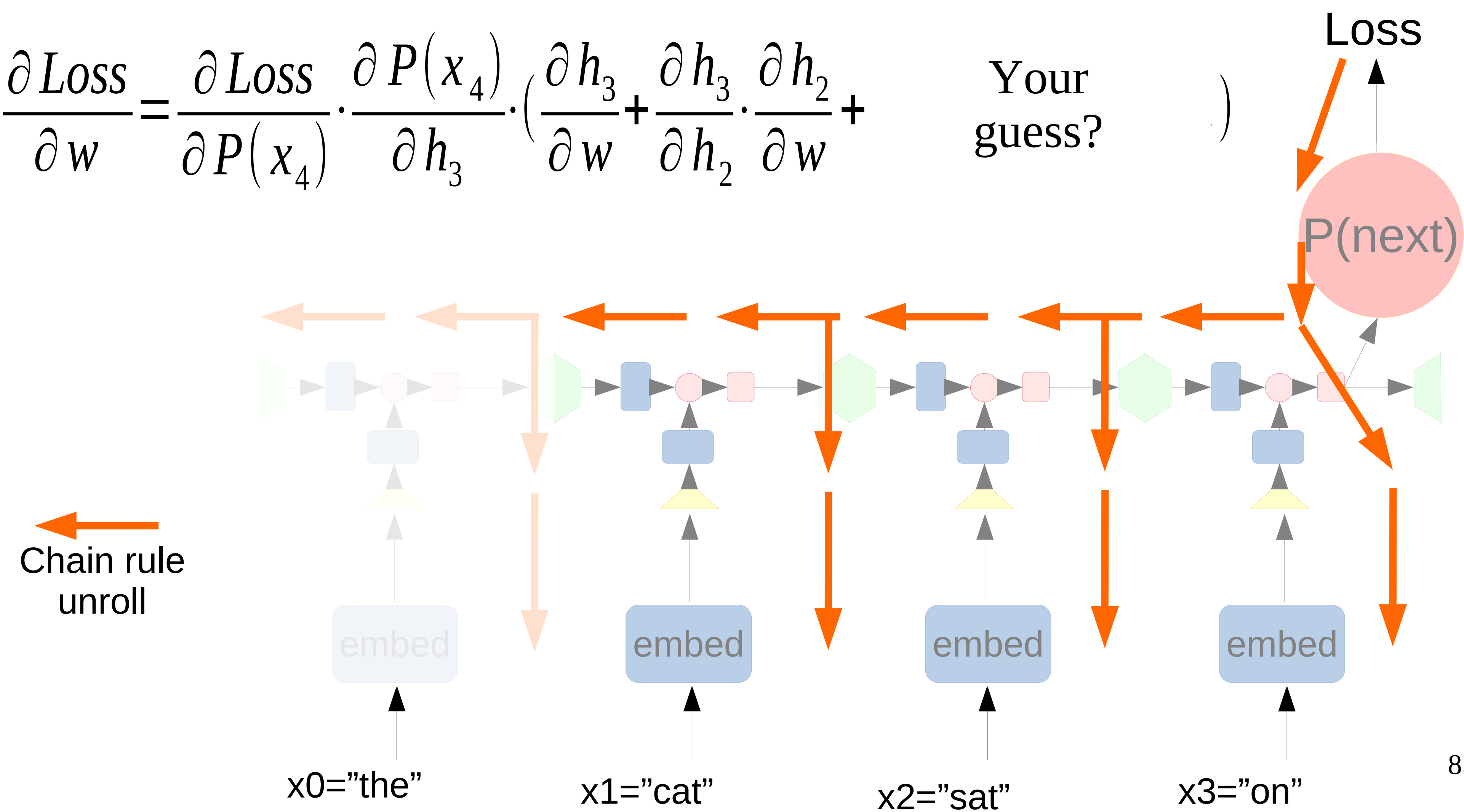
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} \cdot \qquad \right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

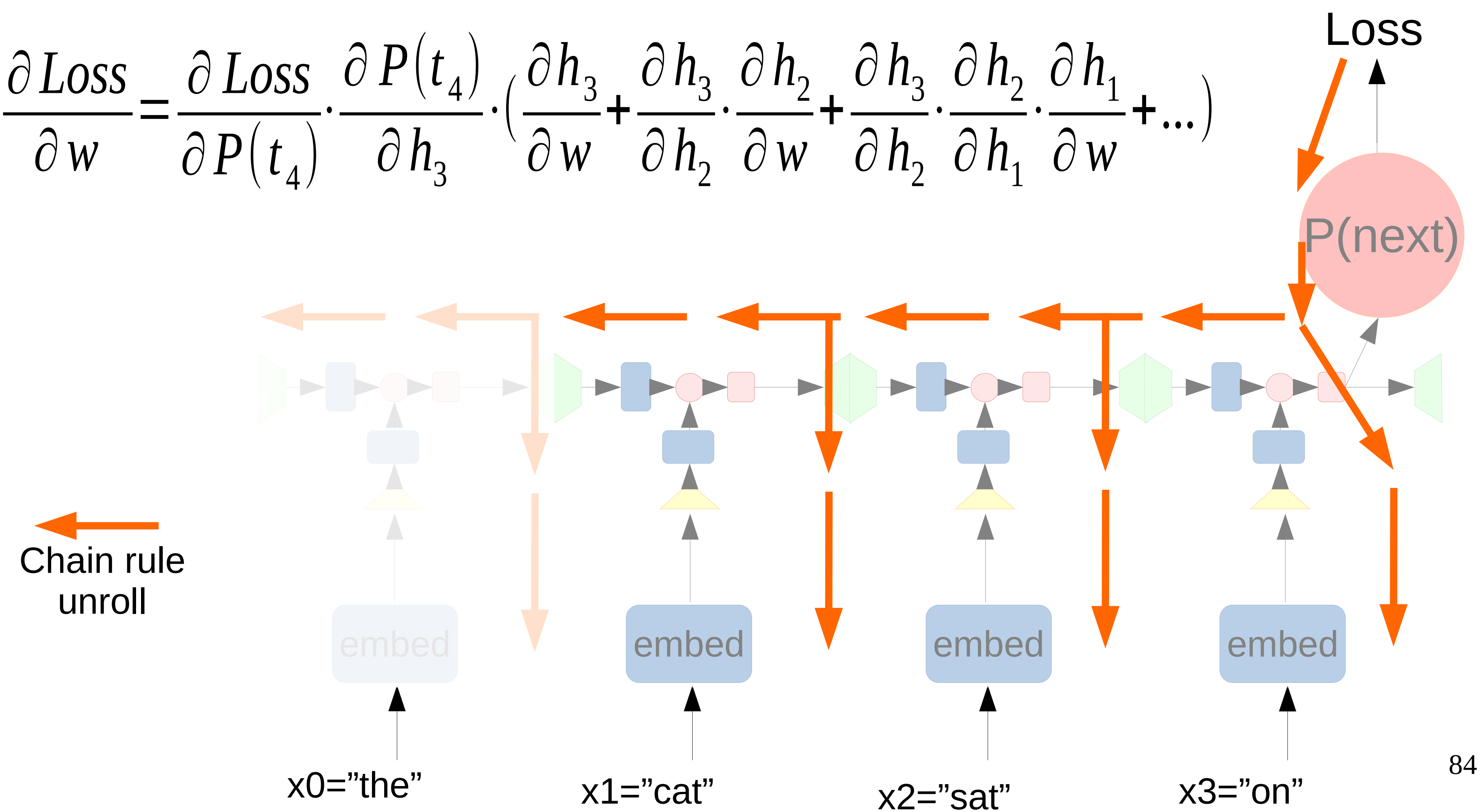
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \quad \text{Your guess?} \quad \right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(t_4)} \cdot \frac{\partial P(t_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w} + \ldots\right)$$



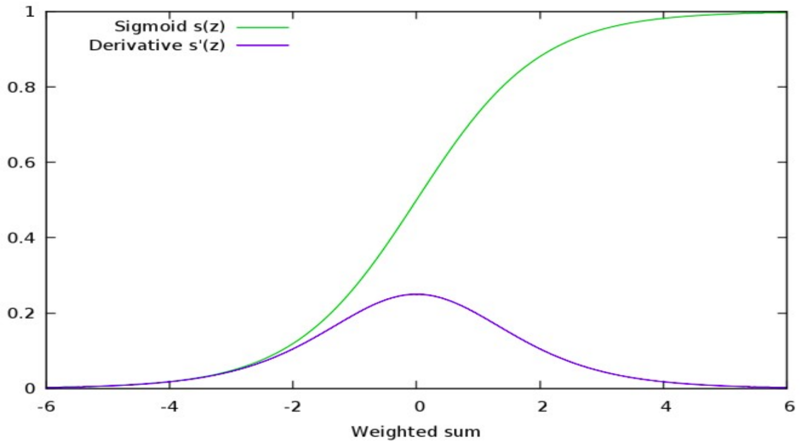
Gradient explosion and vanishing

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

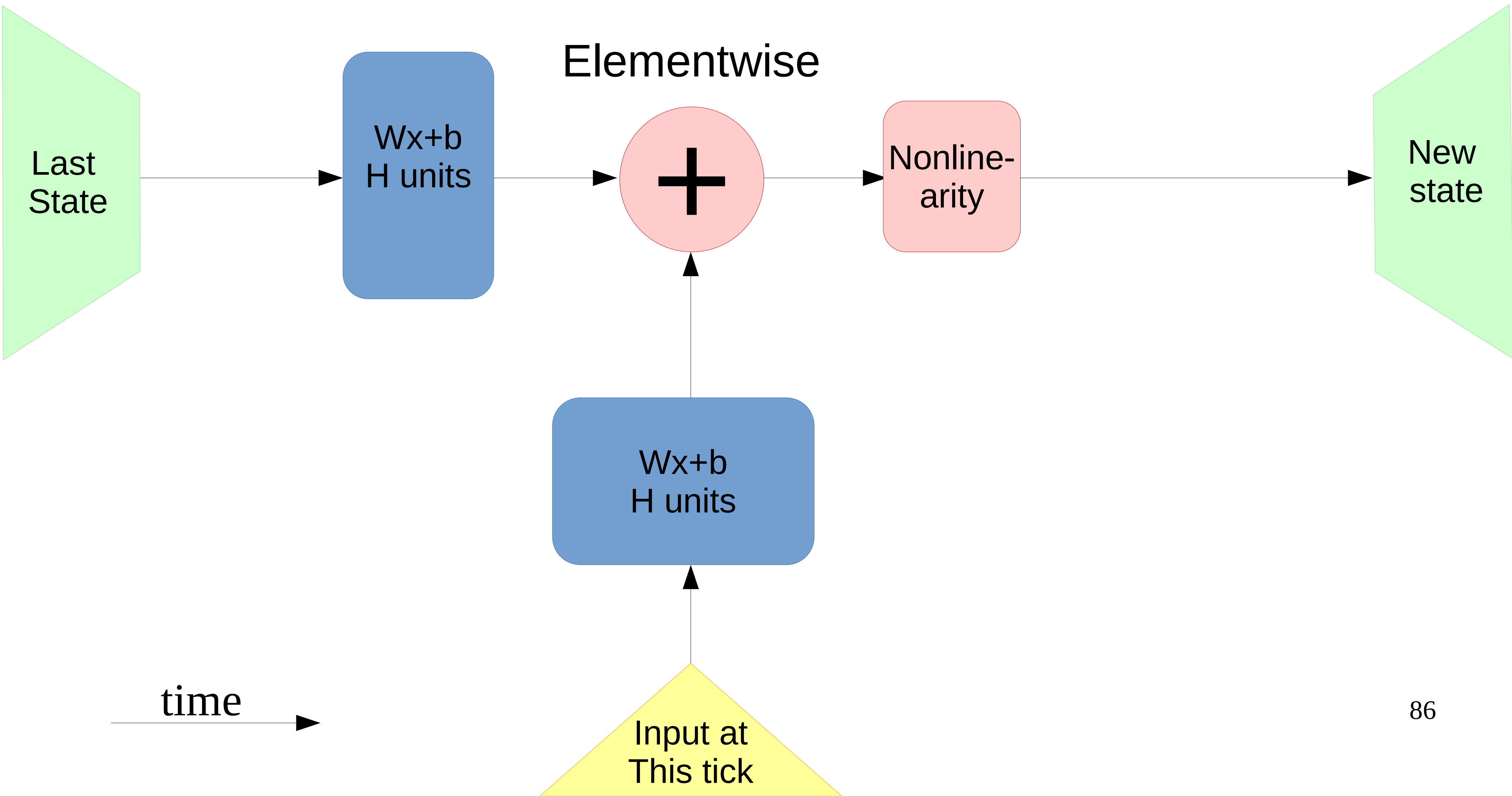
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P\left(x_4\right)} \cdot \frac{\partial P\left(x_4\right)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w} + \ldots\right)$$

- Many sigmoids near 0 or 1
  - Gradients → 0
  - Not training for long-term dependencies

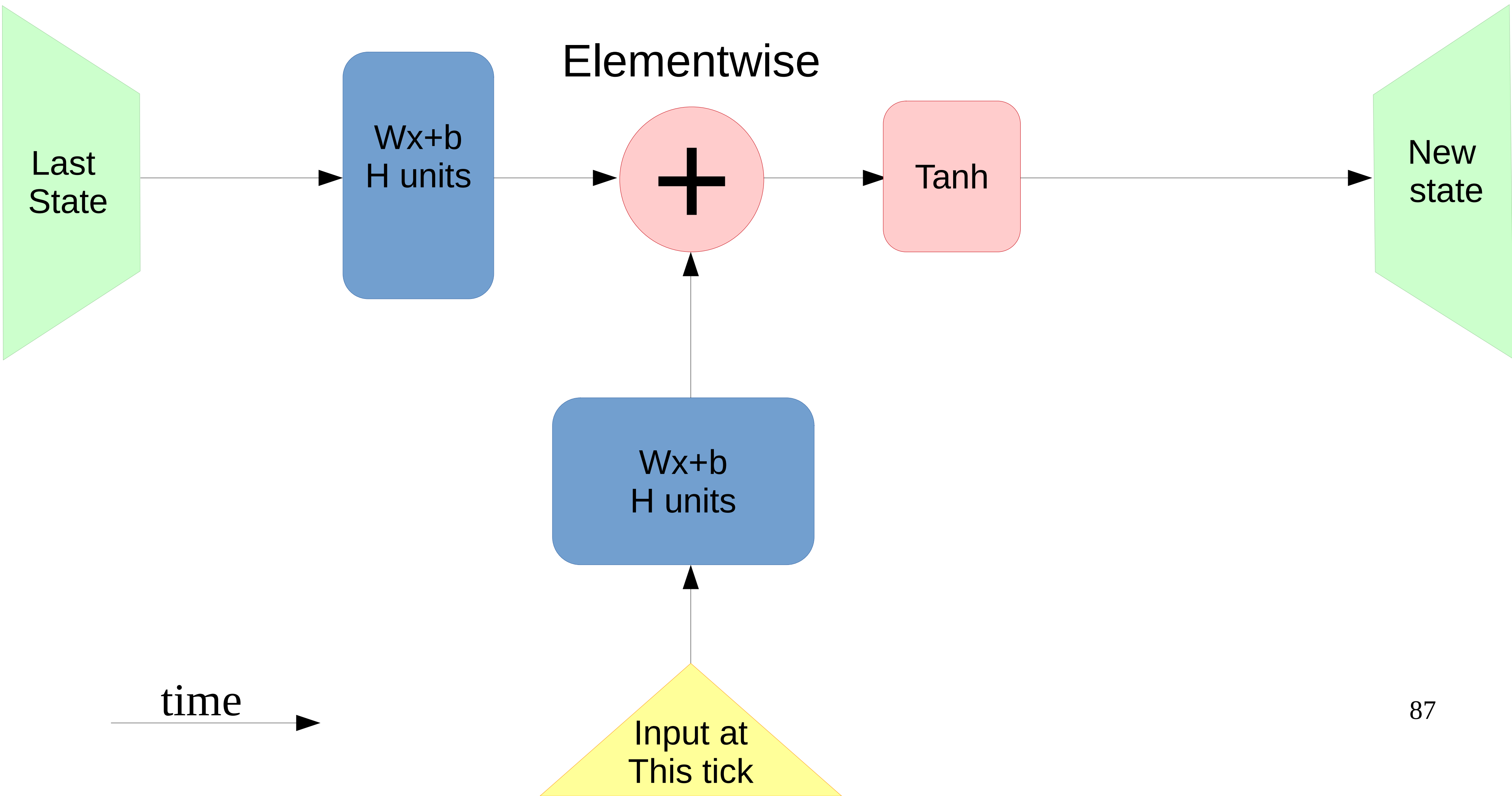
- Many nonzero values
  - Derivative stacks to >1
  - Gradients → inf
  - Weights → shit



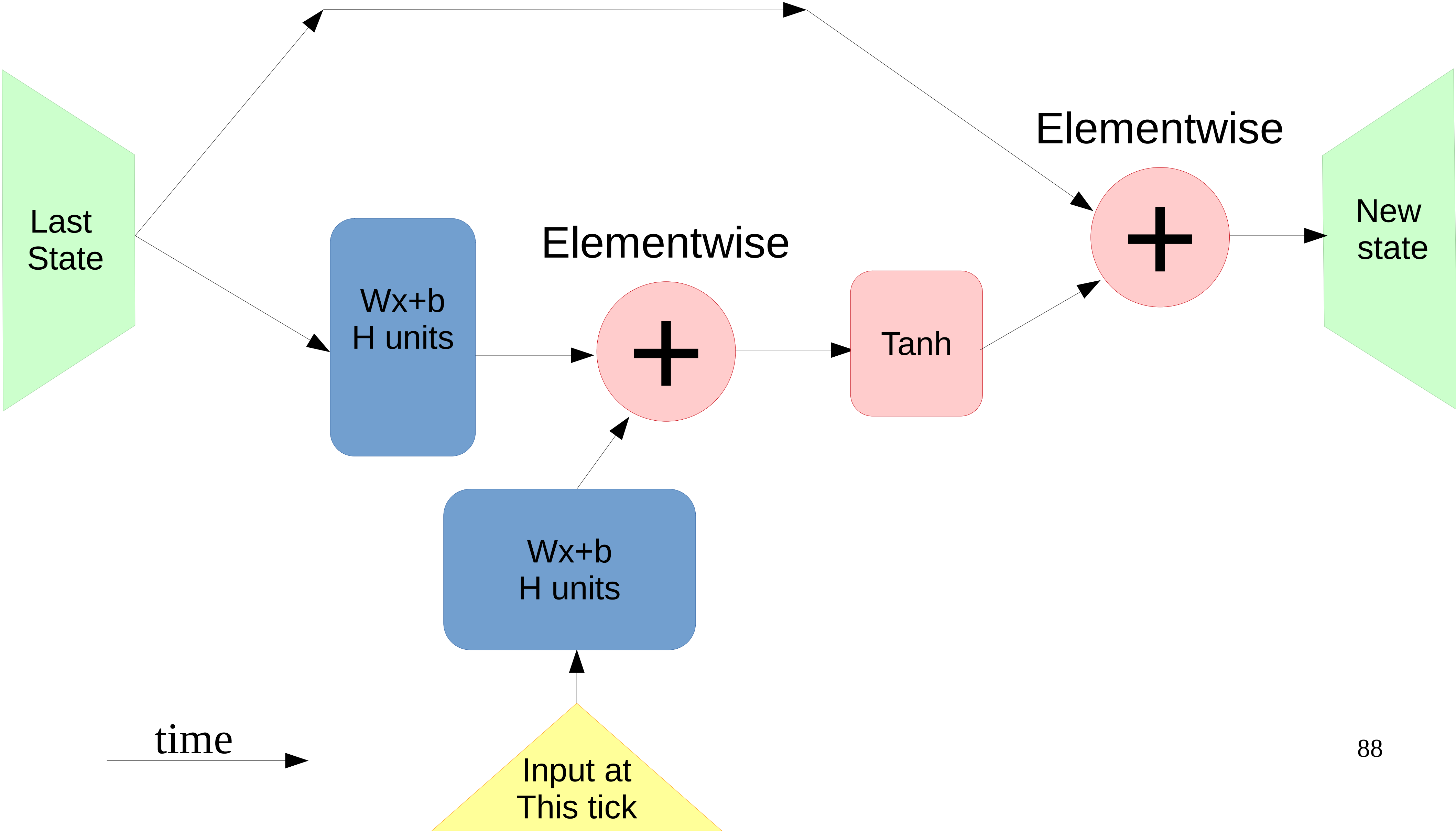
RNN step


Last State
Wx+b H units
Elementwise
+
Nonline- arity
New state
Wx+b H units
Input at This tick
time

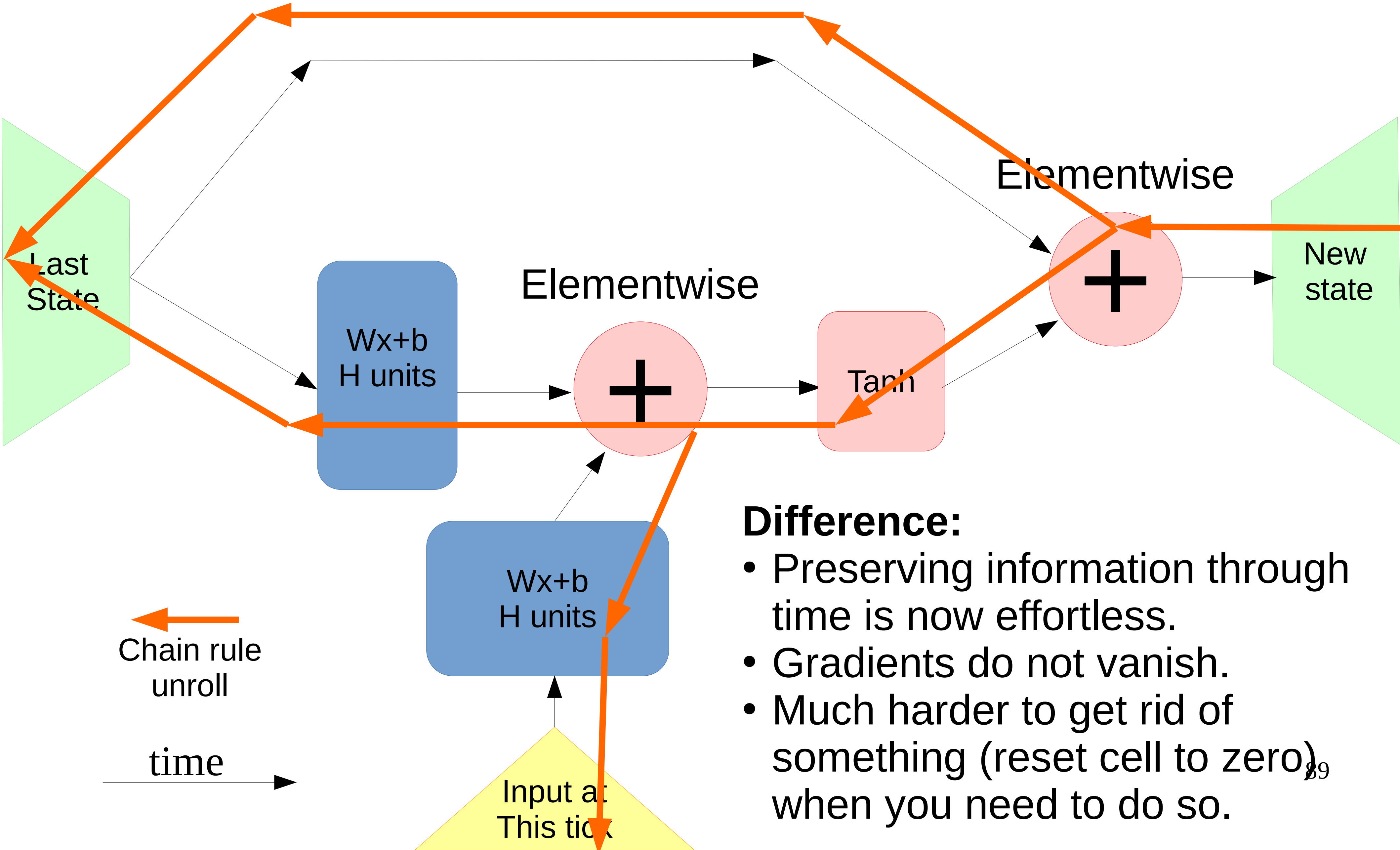
RNN step


Last State
Wx+b H units
Elementwise
+
Tanh
New state
Wx+b H units
Input at This tick
time

Residual RNN step



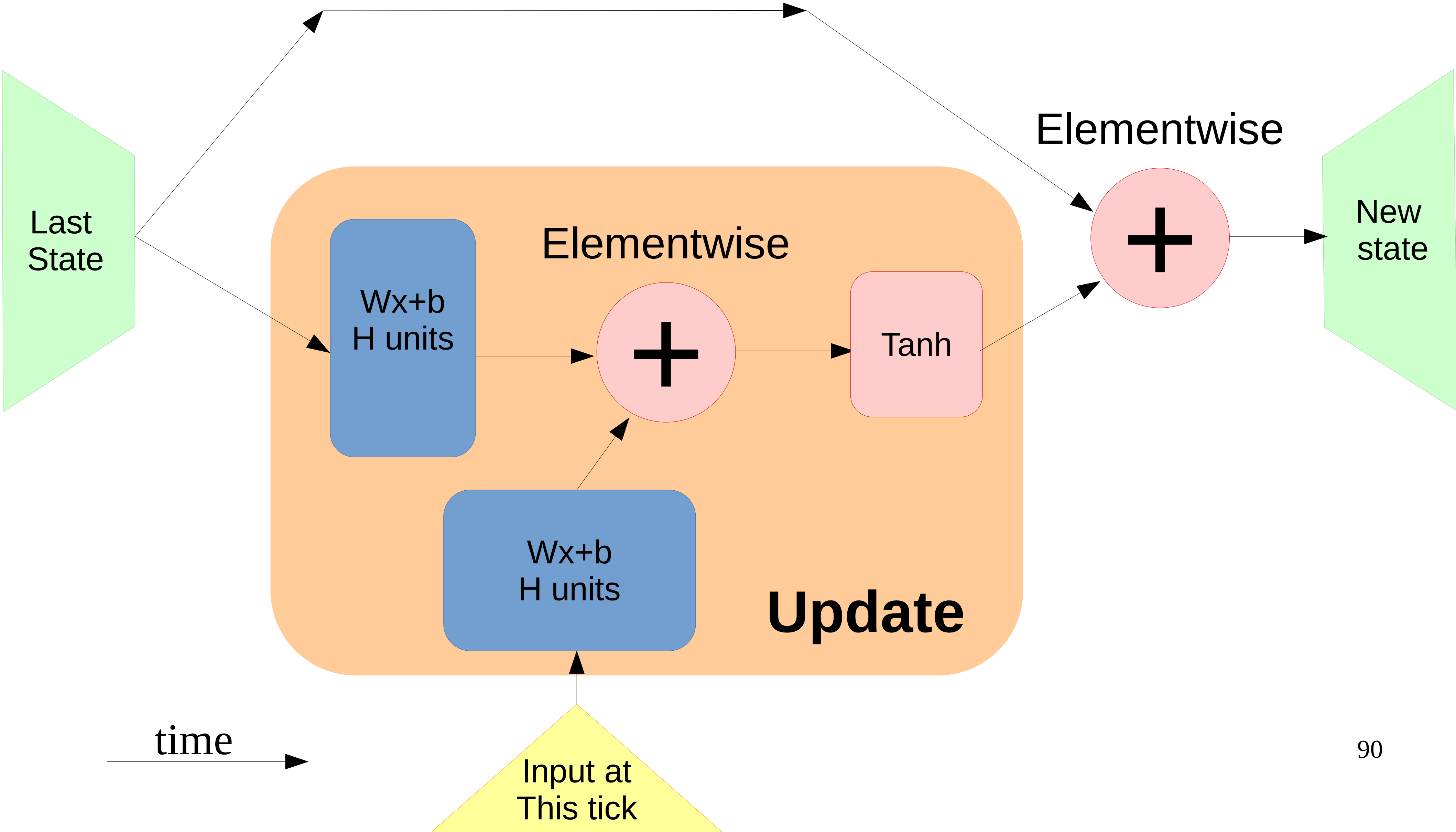
Residual RNN step



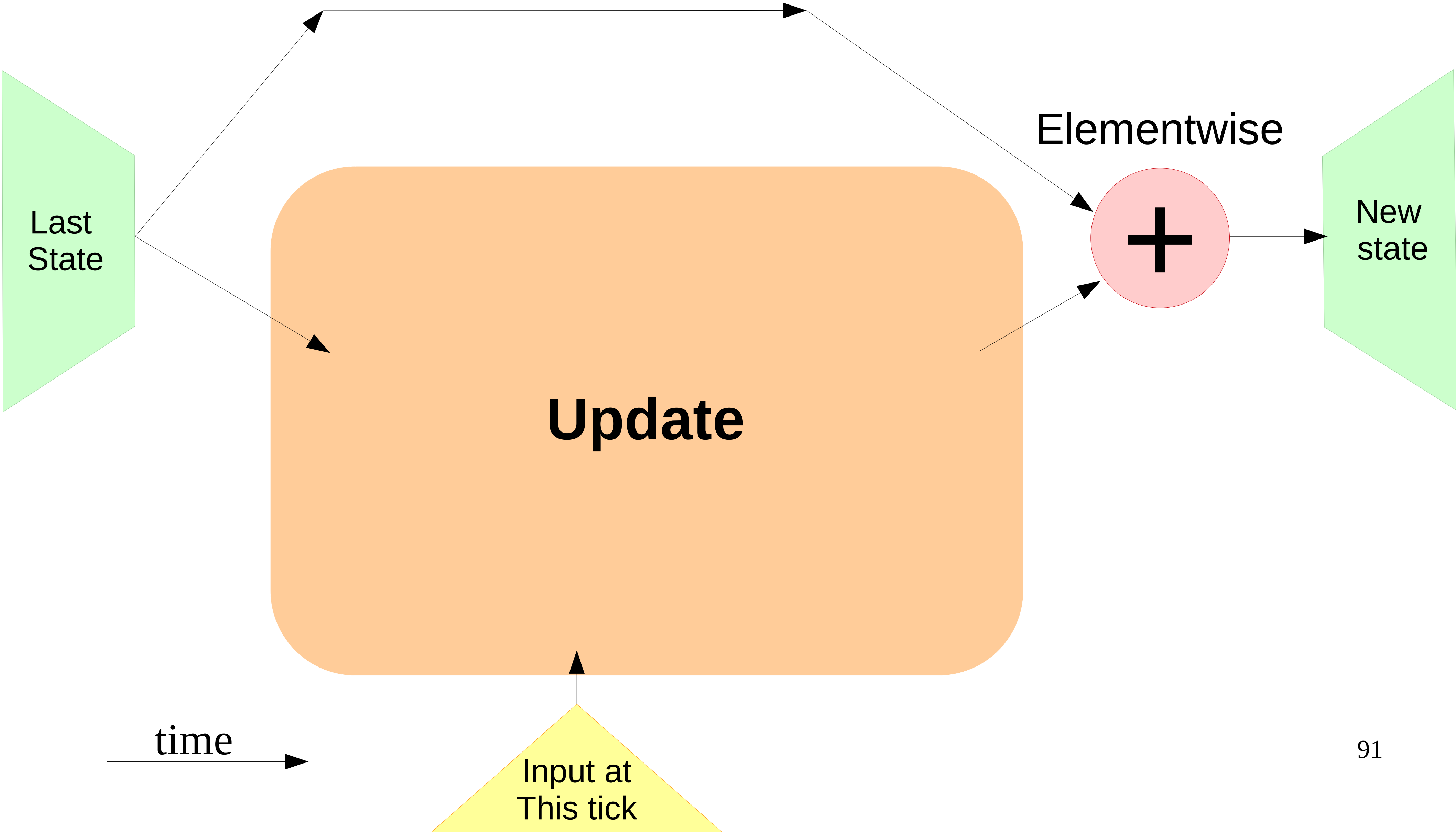
Difference:

- Preserving information through time is now effortless.
- Gradients do not vanish.
- Much harder to get rid of something (reset cell to zero) when you need to do so.

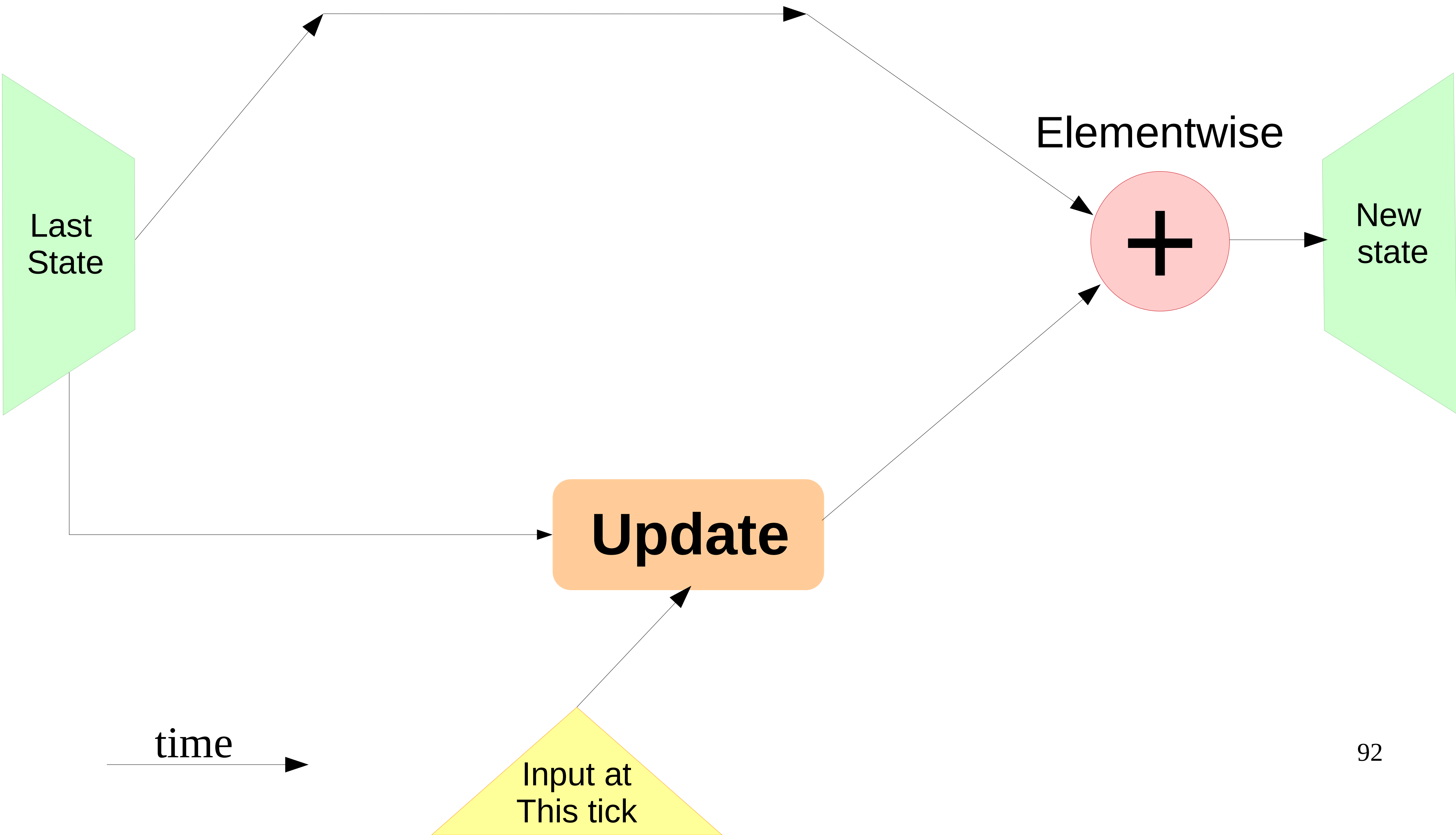
Residual RNN step



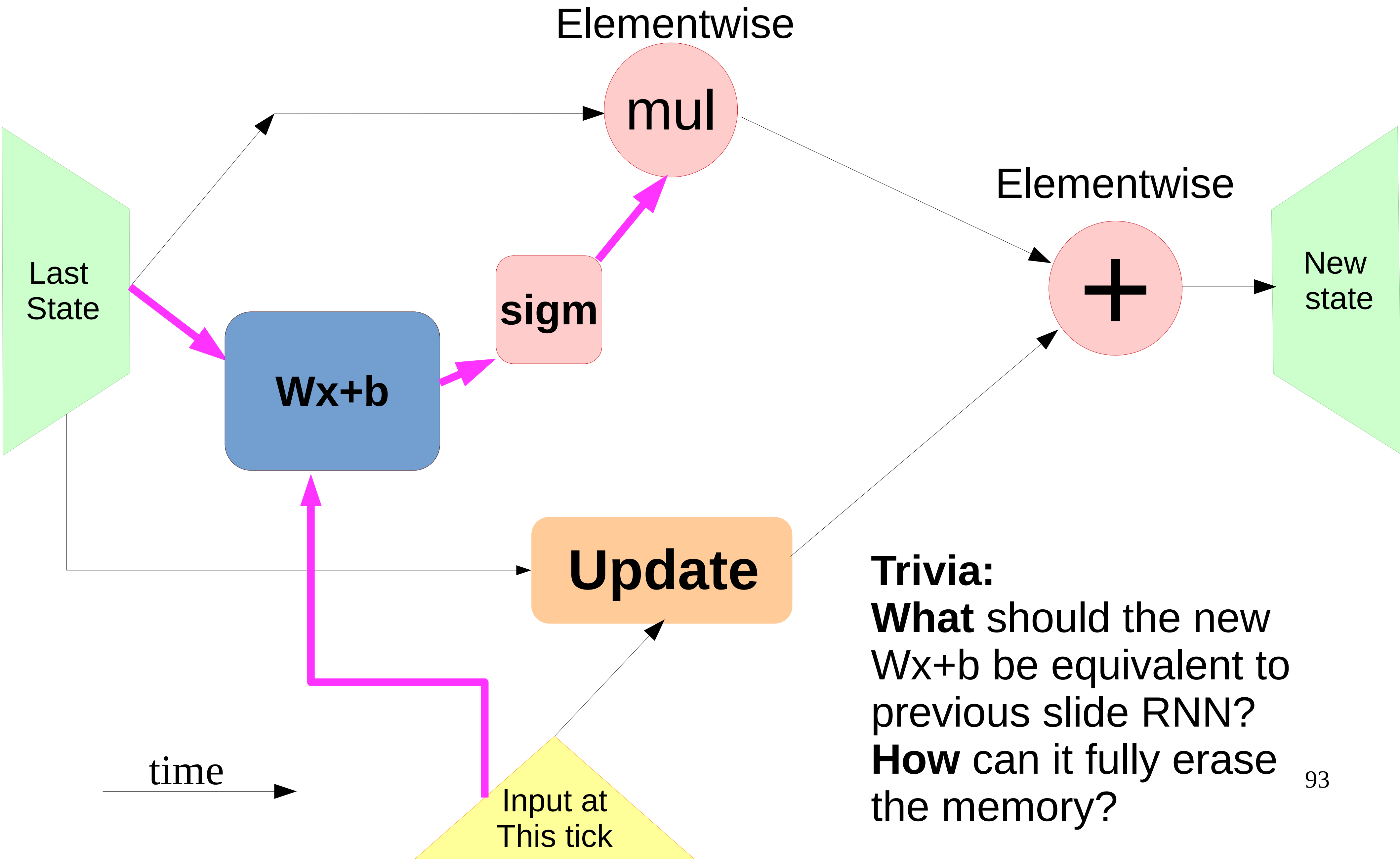
Residual RNN step



Residual RNN step

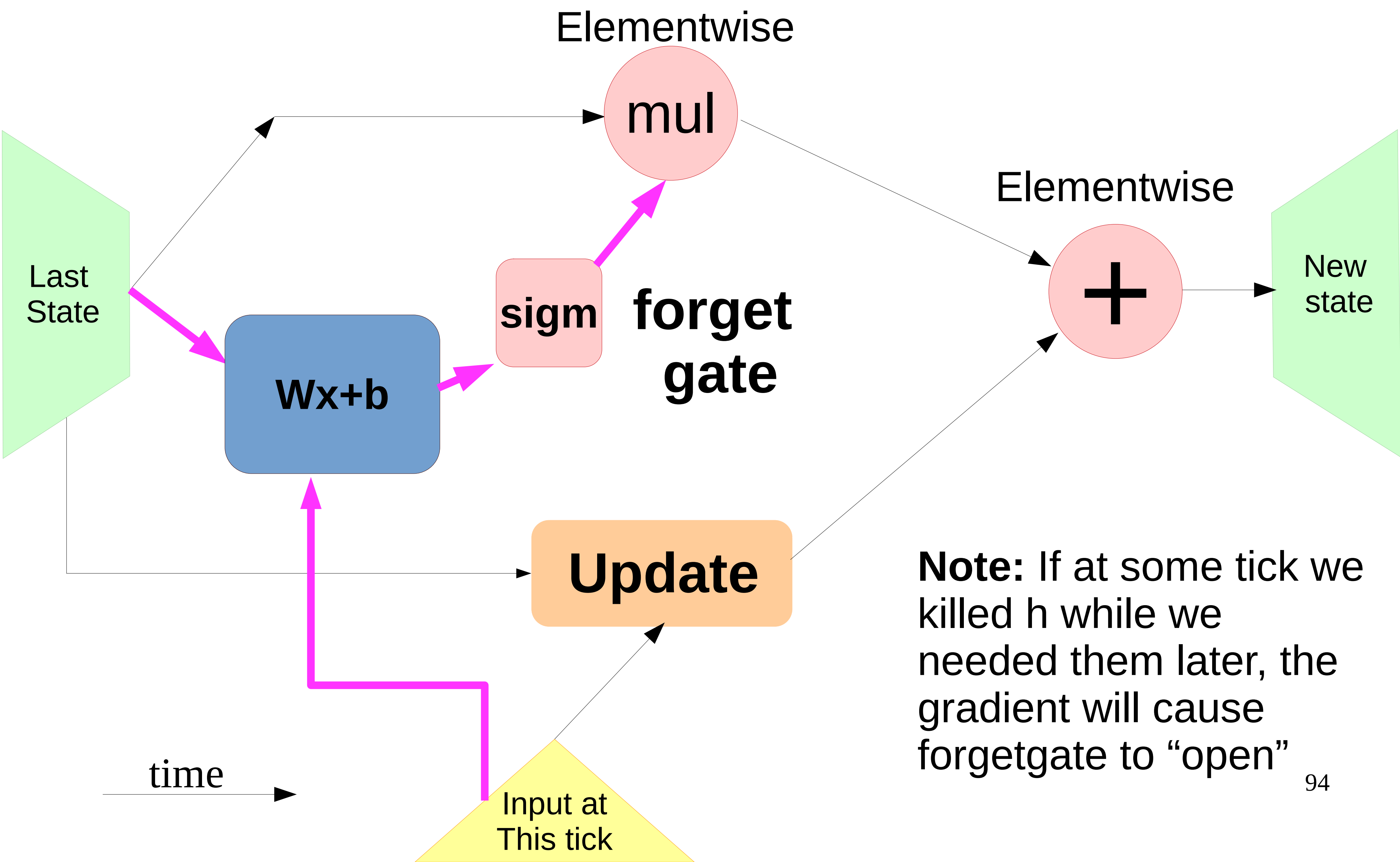


Residual RNN step



Trivia:
What should the new Wx+b be equivalent to previous slide RNN?
How can it fully erase the memory?

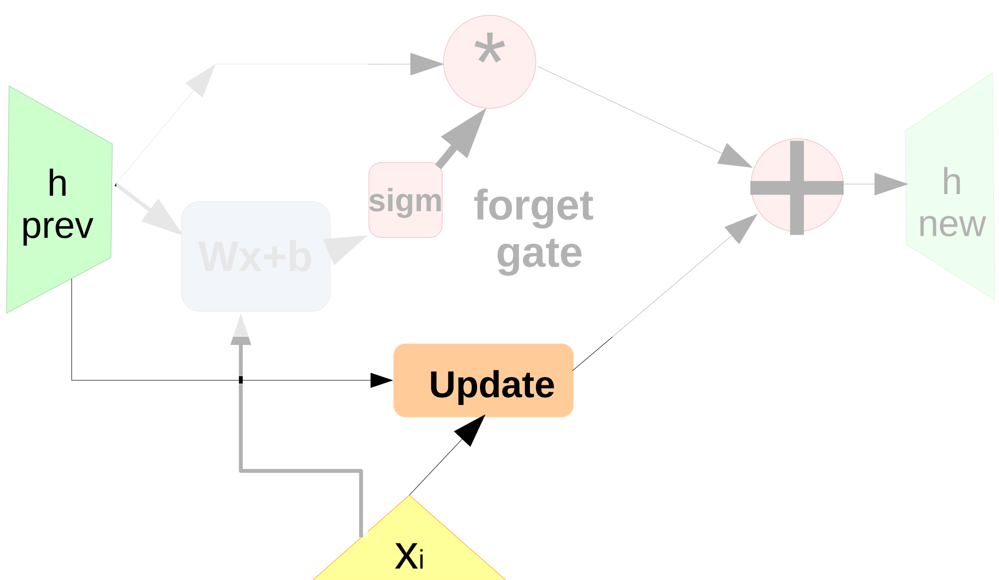
Residual RNN step



Note: If at some tick we killed h while we needed them later, the gradient will cause forgetgate to "open"

What we drew

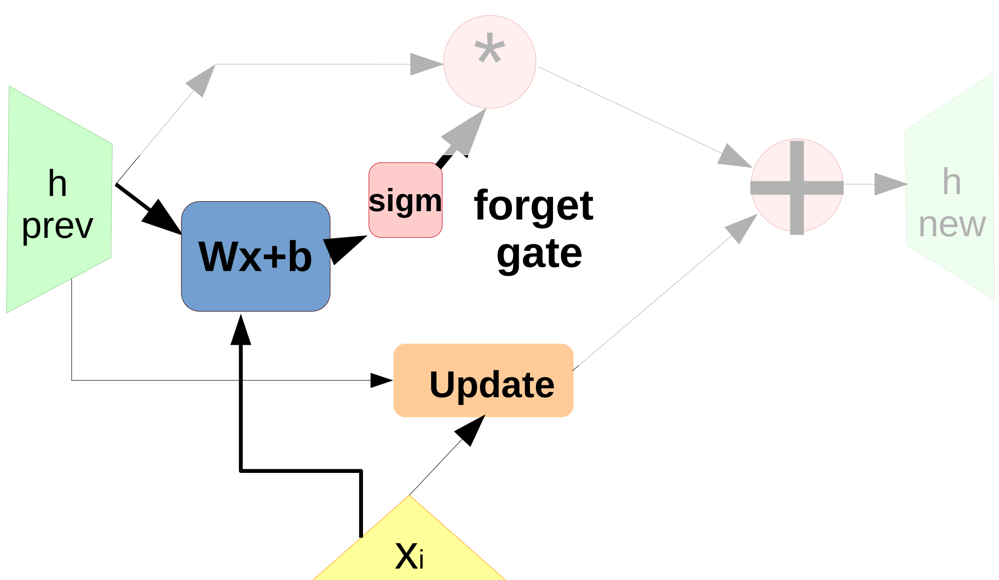
$$update\left(x_i, h_{i-1}\right) = \tanh\left(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update}\right)$$



What we drew

$$update(x_i, h_{i-1}) = \tanh\left(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update}\right)$$

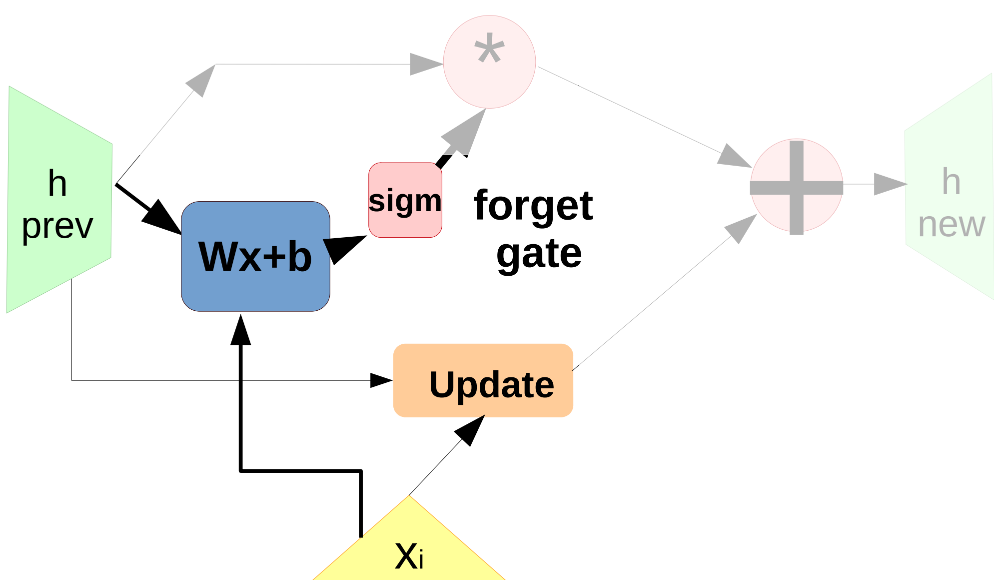
$$forget(x_i, h_{i-1}) = \sigma\left(W_{hid}^{forget} \cdot h_{i-1} + W_{inp}^{forget} \cdot x_i + b^{forget}\right)$$



What we drew

$$update(x_i, h_{i-1}) = \tanh\left(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update}\right)$$

$$forget(x_i, h_{i-1}) = \sigma\left(W_{hid}^{forget} \cdot h_{i-1} + W_{inp}^{forget} \cdot x_i + b^{forget}\right)$$



How to compute h_new?

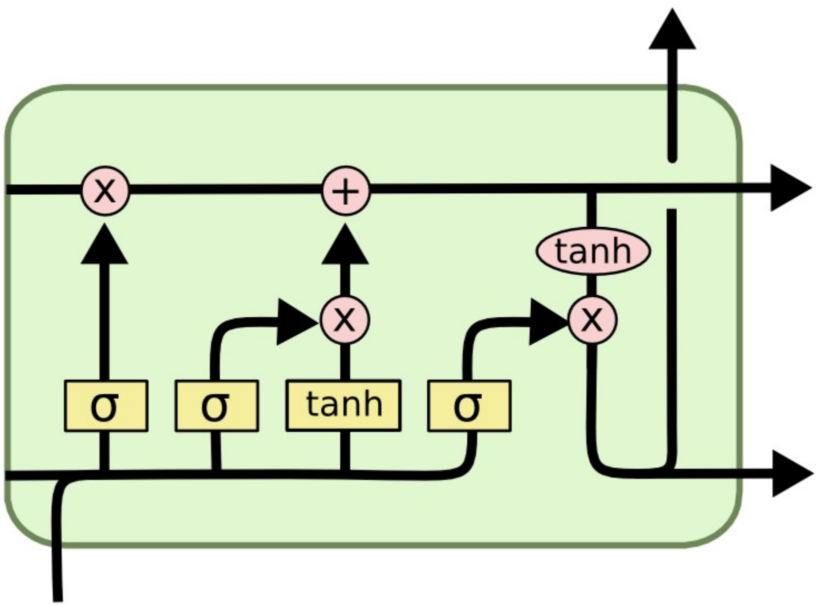
What we drew

$$update(x_i, h_{i-1}) = \tanh(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update})$$

$$forget(x_i, h_{i-1}) = \sigma(W_{hid}^{forget} \cdot h_{i-1} + W_{inp}^{forget} \cdot x_i + b^{forget})$$

$$h_i(x_i, h_{i-1}) = forget(x_i, h_{i-1}) \cdot h_{i-1} + update(x_i, h_{i-1})$$

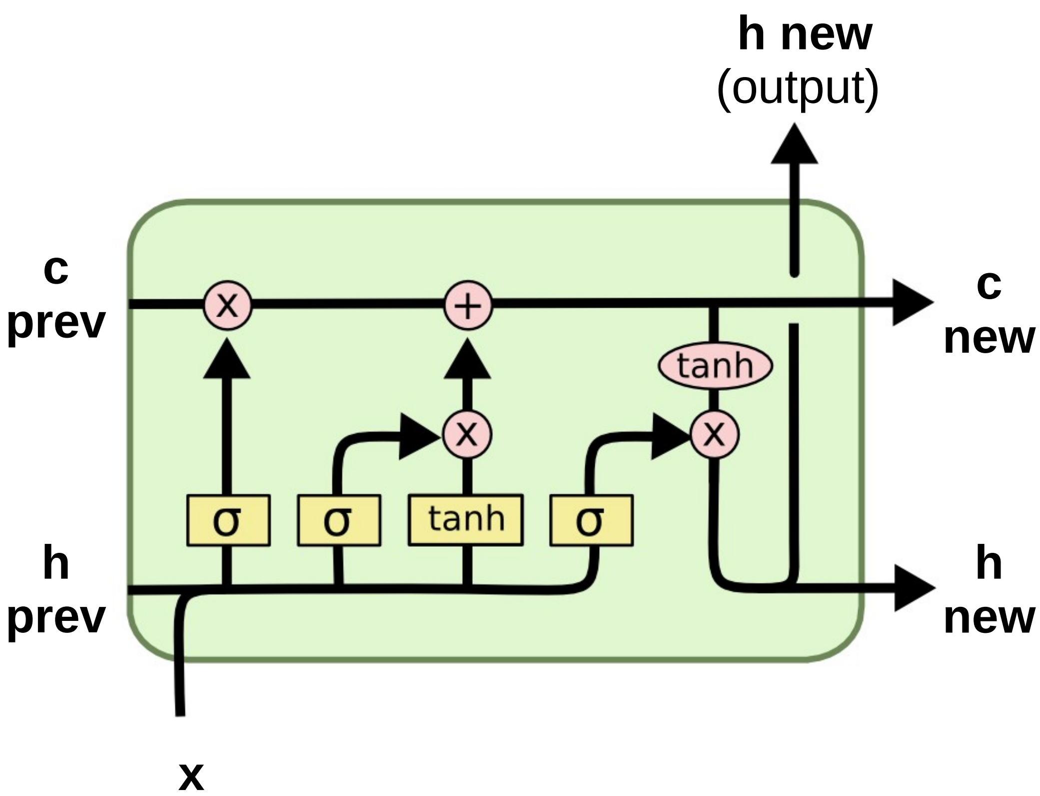
LSTM



2 hidden states:
- Cell (“private” state)
- Output (“public” state)

4 blocks:
- Update
- Forget gate
- Input gate
- Output gate

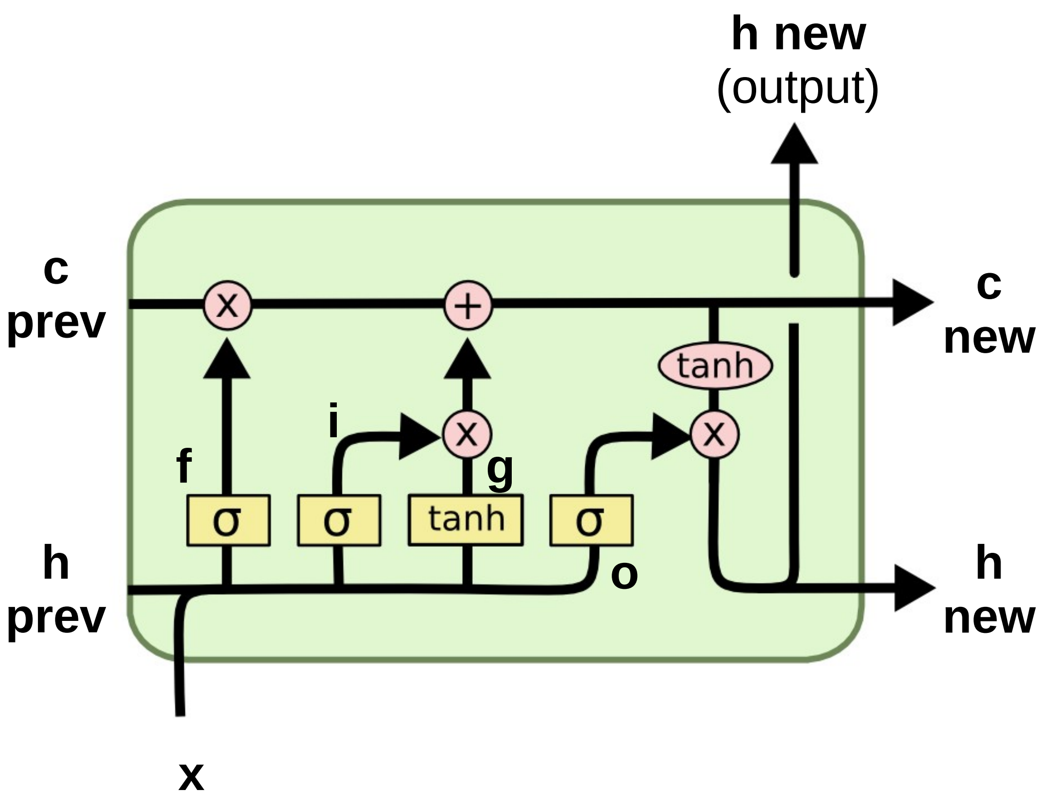
LSTM



$$i_t = Sigm(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$
$$f_t = Sigm(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)$$
$$o_t = Sigm(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$
$$g_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$
$$h_t = o_t \otimes Tanh(c_t)$$

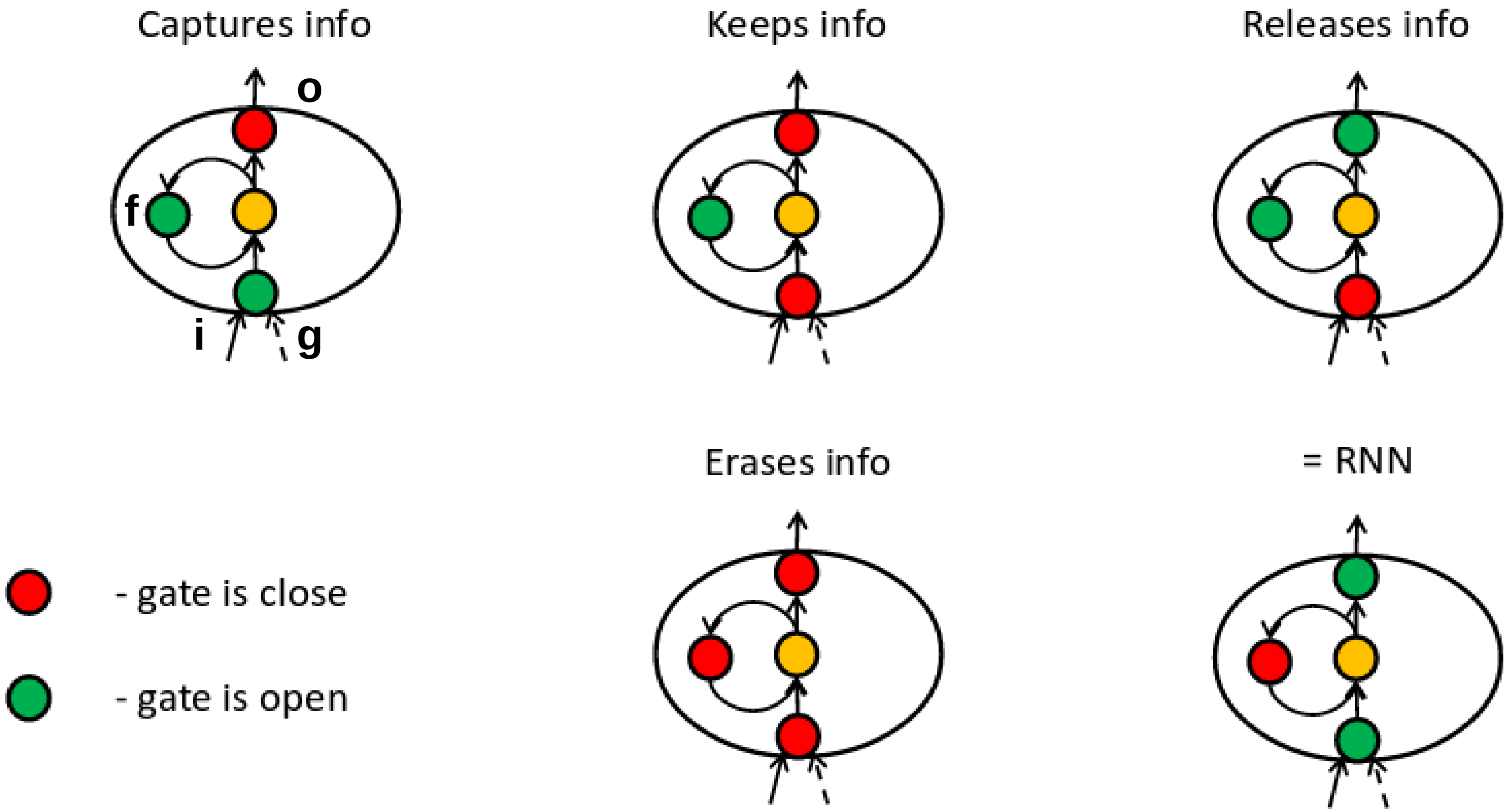
Where are the gates?

LSTM



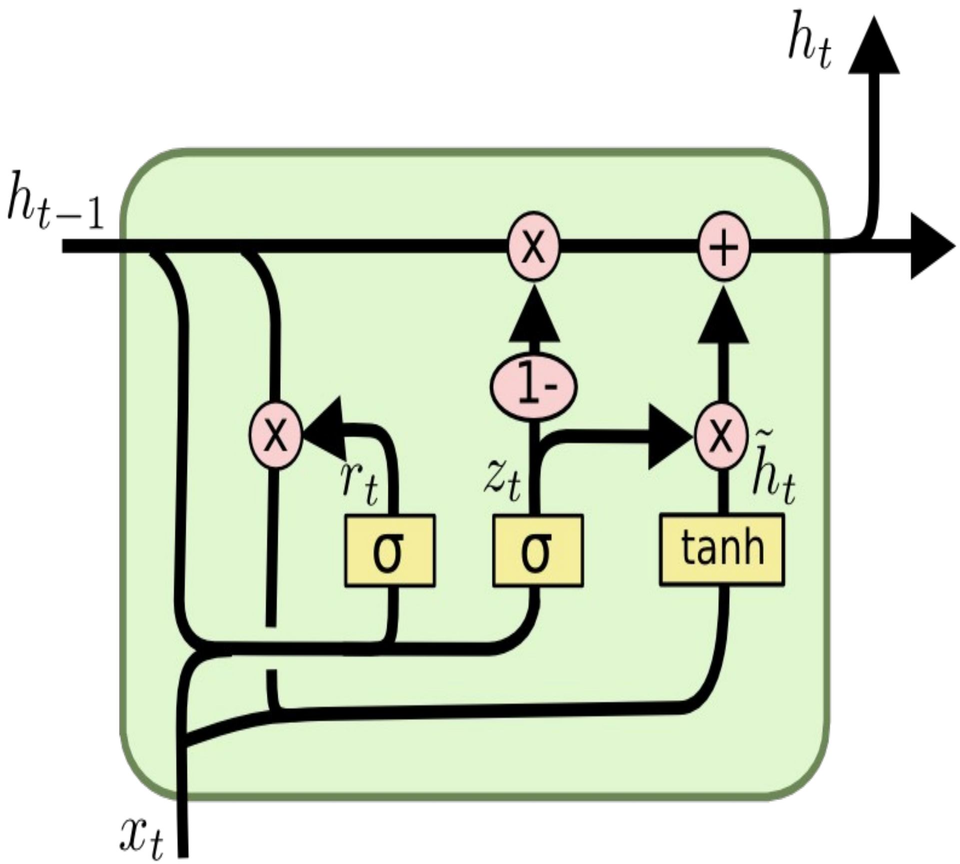
$$i_t = Sigm(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$
$$f_t = Sigm(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)$$
$$o_t = Sigm(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$
$$g_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$
$$h_t = o_t \otimes Tanh(c_t)$$

LSTM: not a monster



[Pictures: E Lobacheva, D Vetrov]

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

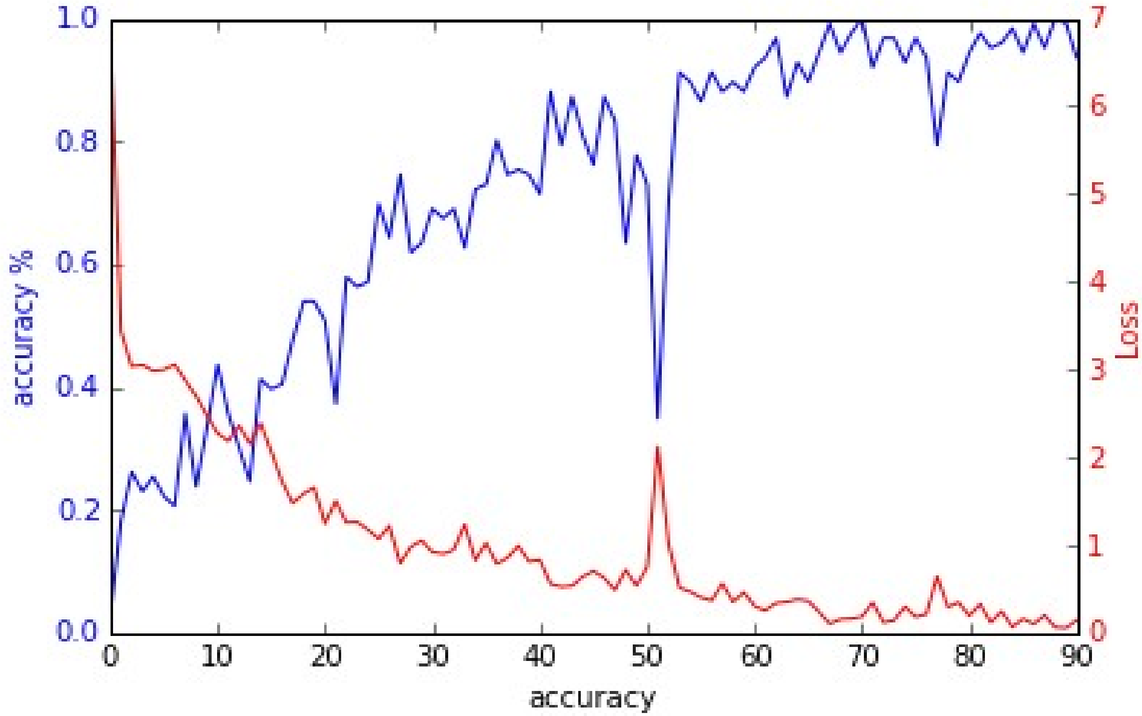
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Okay, the gradients no longer vanish

except they still do, if only slower

But how do we deal with exploding grads?



Ideas?

Gradient clipping

At each time tick,

- check if grad abs value is more than … 5?
- If so, clip it
  - large positive is now 5,
  - large negative is now -5

- How large is too large?
  - Reduce clipping threshold until explosions disappear

Gradient clipping

Where do I clip?

- Clip each element of $\delta L / \delta w$
- Clip each element of $\delta h_{i+1} / \delta h_i$
- Clip whole $\delta L / \delta w$ by norm
  - If $\left\| \frac{\delta L}{\delta w} \right\| > 5$, scale $\frac{\delta L}{\delta w} / \left\| \frac{\delta L}{\delta w} \right\| \cdot 5$

Generating stuff

Easy:

- Names, small phrases
- Arxiv article titles
- Orthographically correct delirium

Medium:

- Music (notes)
- Organic molecules (SMILES)

Hard:

- C/C++ source code
- Articles (LaTeX full text)
- Your course projects >.<
- Seq2Seq

Nuff

Coding time!