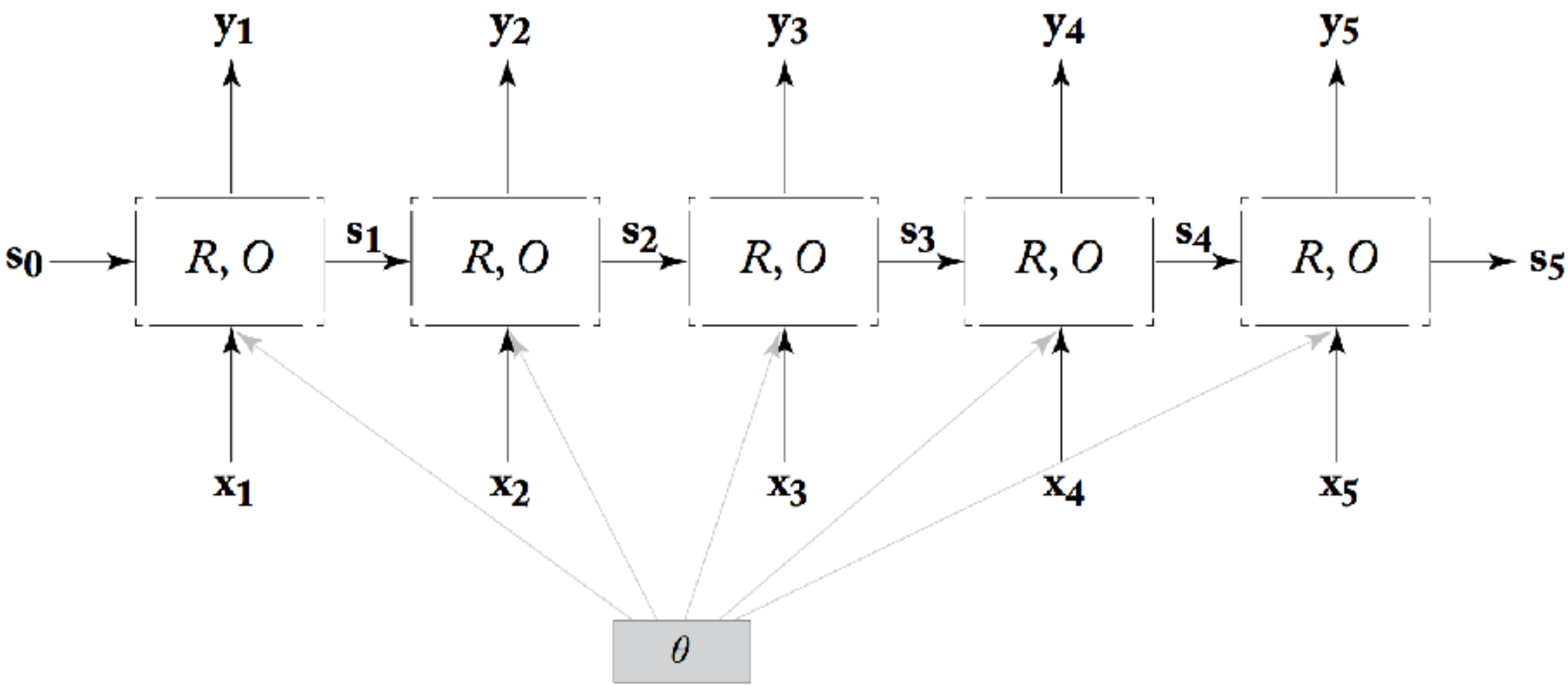
Natural Language Processing

Mehmet Can Yavuz, PhD

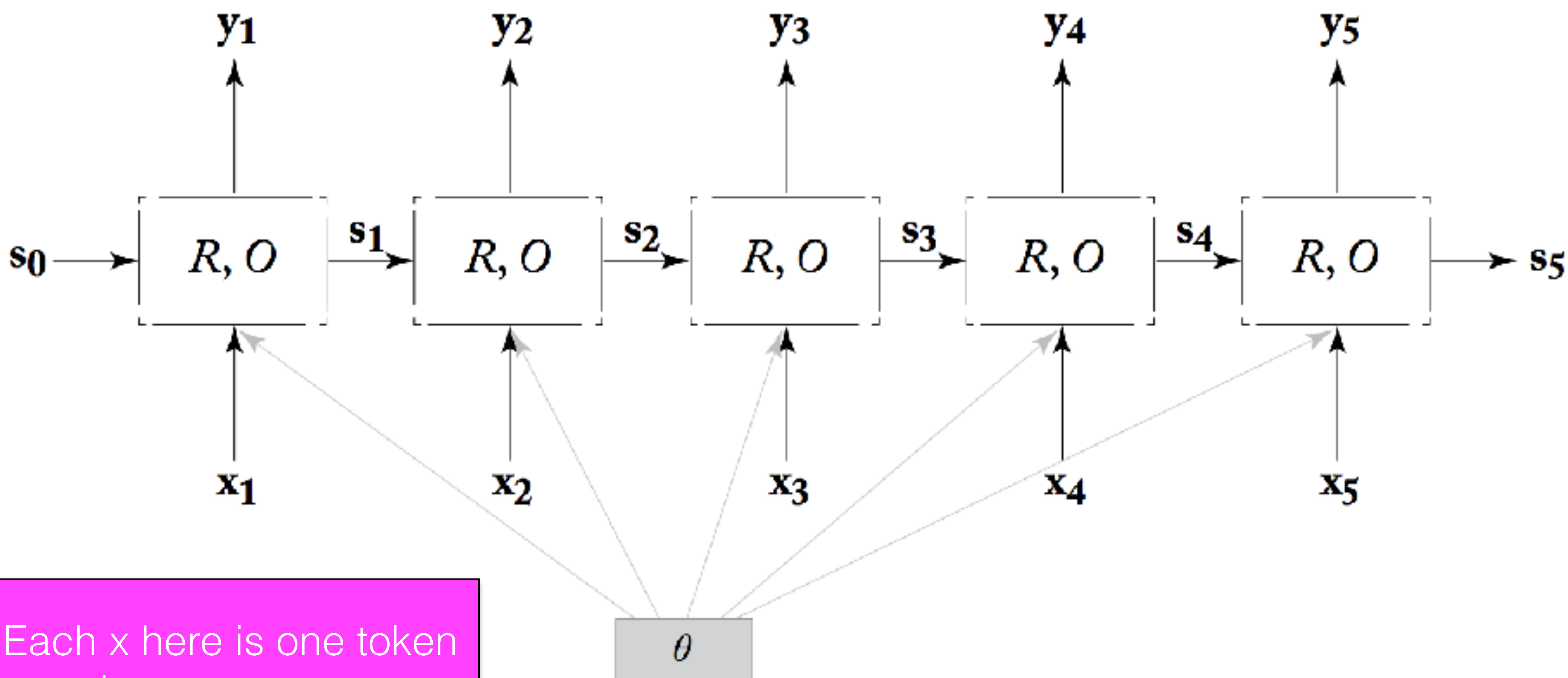
Adapted from Info 256 - David Bamman, UC Berkeley

Recurrent neural network



Goldberg 2017

Each y is the output of the RNN at that time step; sometimes we use this information (POS tagging, LM); sometimes we only use the output for the final state (s_5)

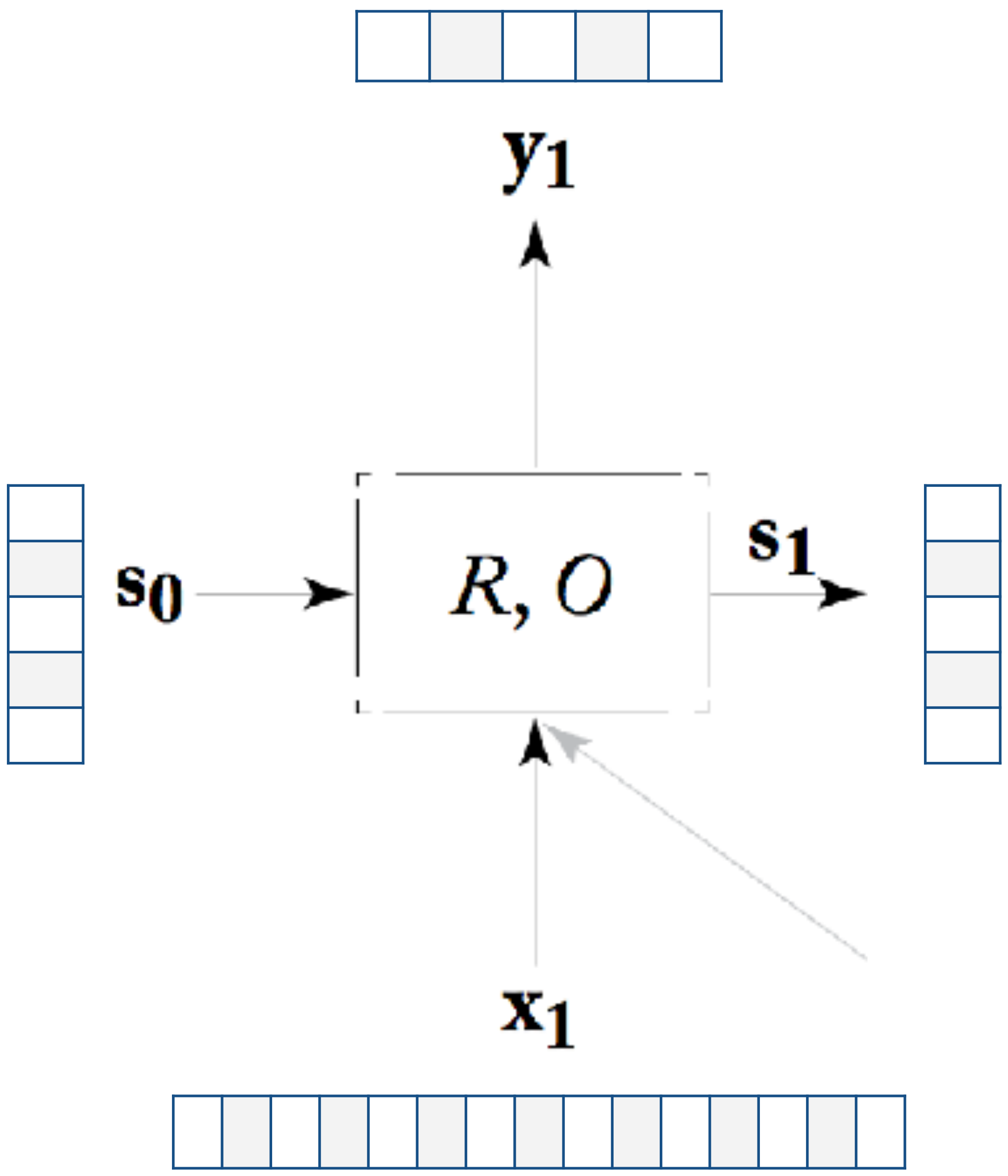


Each x here is one token in a sequence

Goldberg 2017

Recurrent neural network

- Each time step has two inputs:
  - x_i (the observation at time step i); one-hot vector, feature vector or word embedding.
  - s_{i-1} (the output of the previous state); base case: s_0 = 0 vector



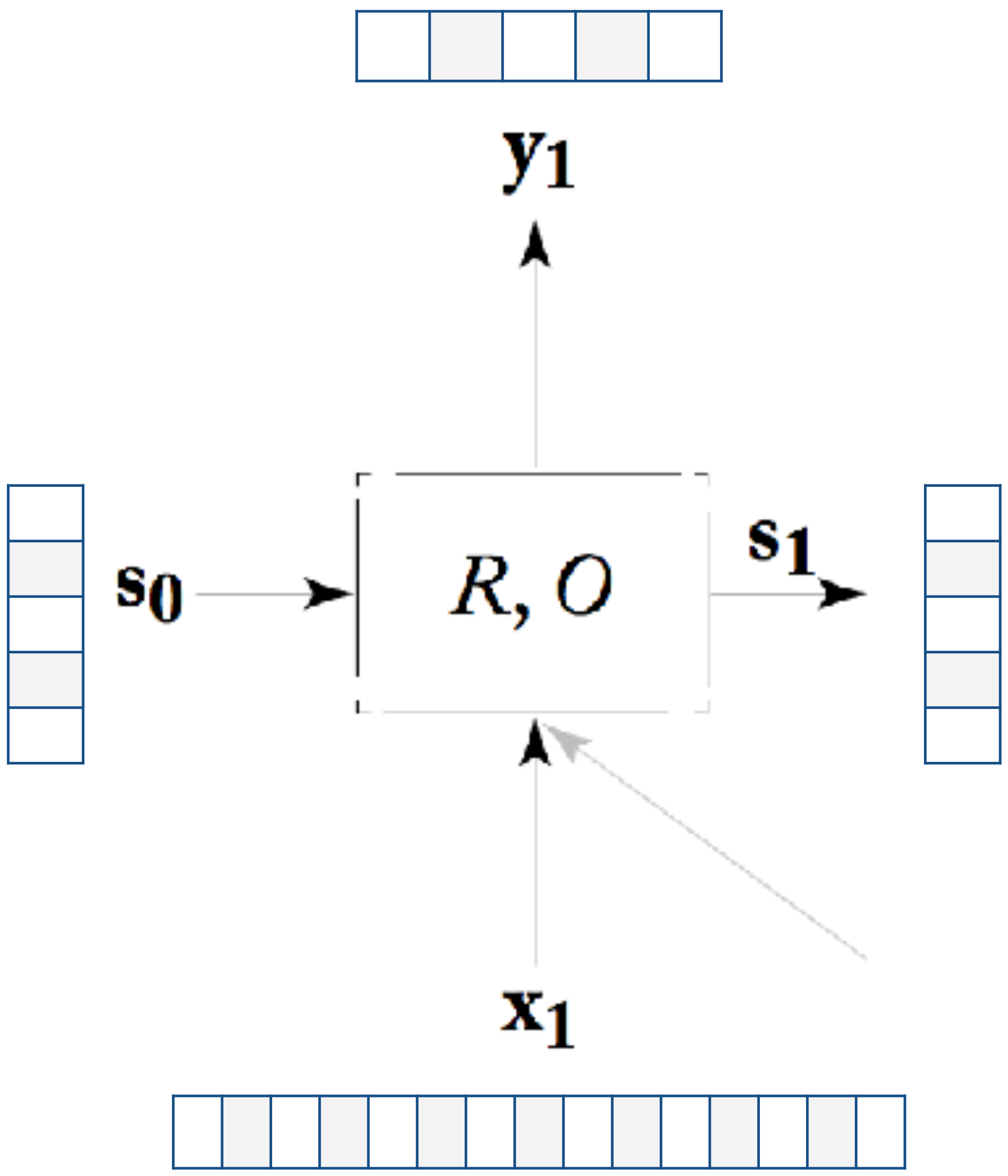
Recurrent neural network

$$s_i = R(x_i, s_{i-1})$$

R computes the output state as a function of the current input and previous state

$$y_i = O(s_i)$$

O computes the output as a function of the current output state



“Simple” RNN

g = tanh or relu

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$

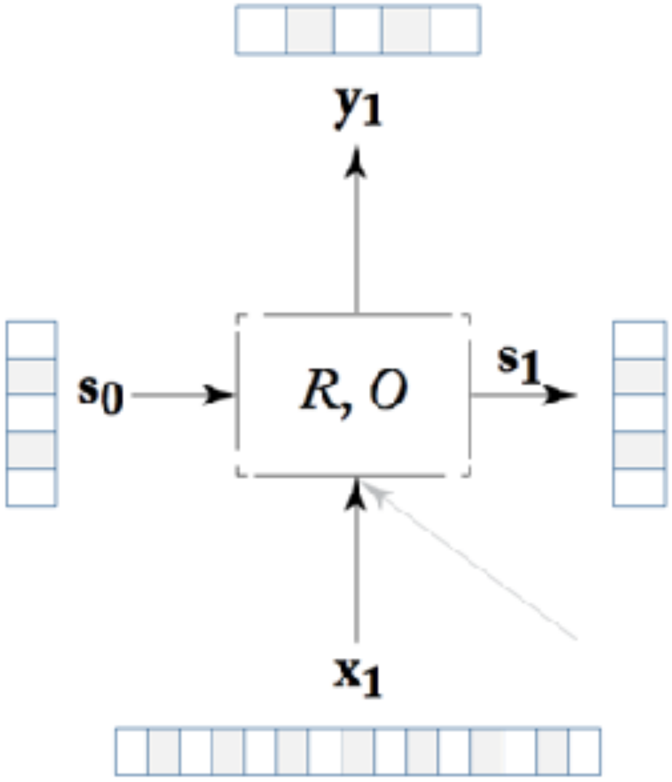
Different weight vectors W transform the previous state and current input before combining

$$W^s \in \mathbb{R}^{H \times H}$$

$$W^x \in \mathbb{R}^{D \times H}$$

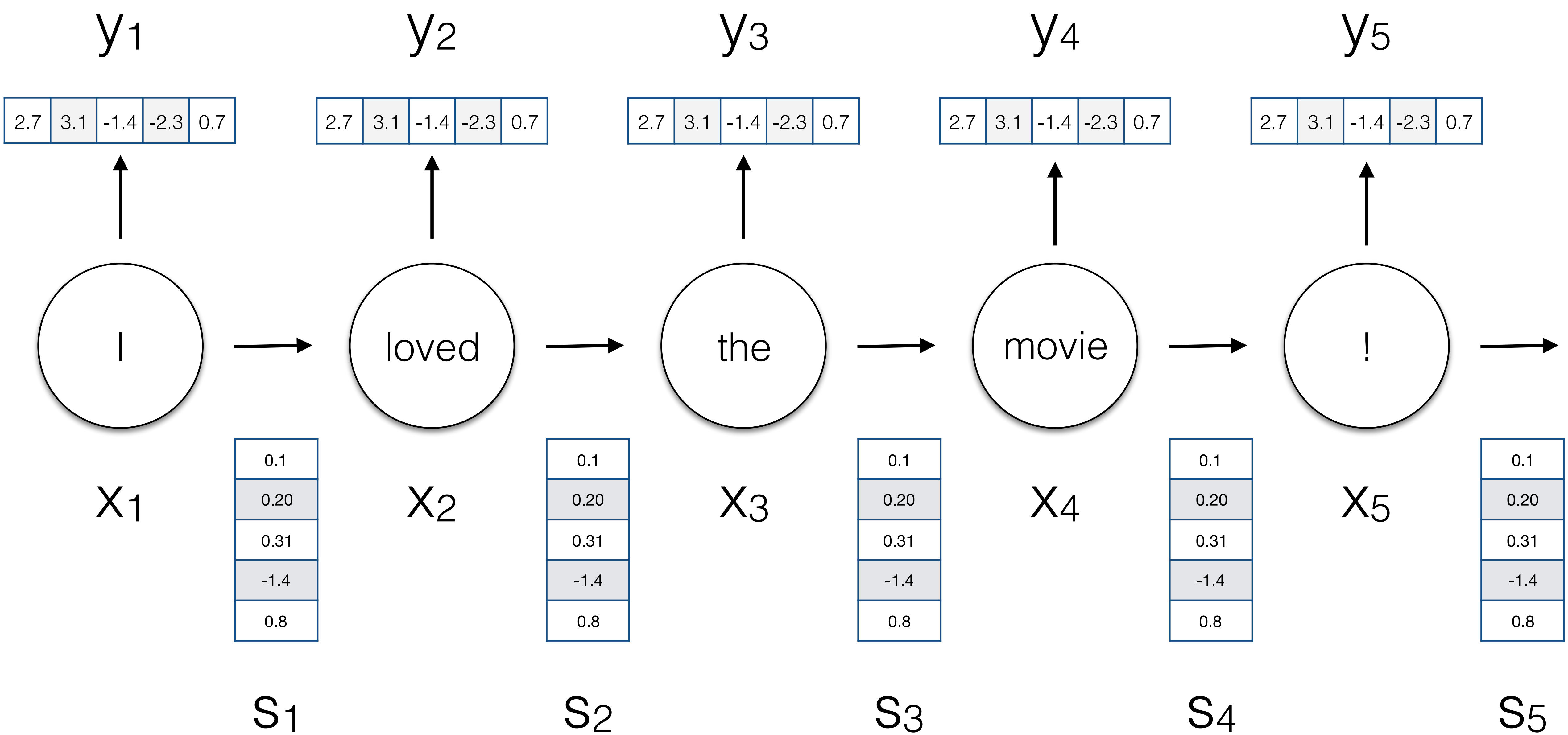
$$b \in \mathbb{R}^H$$

$$y_i = O(s_i) = s_i$$

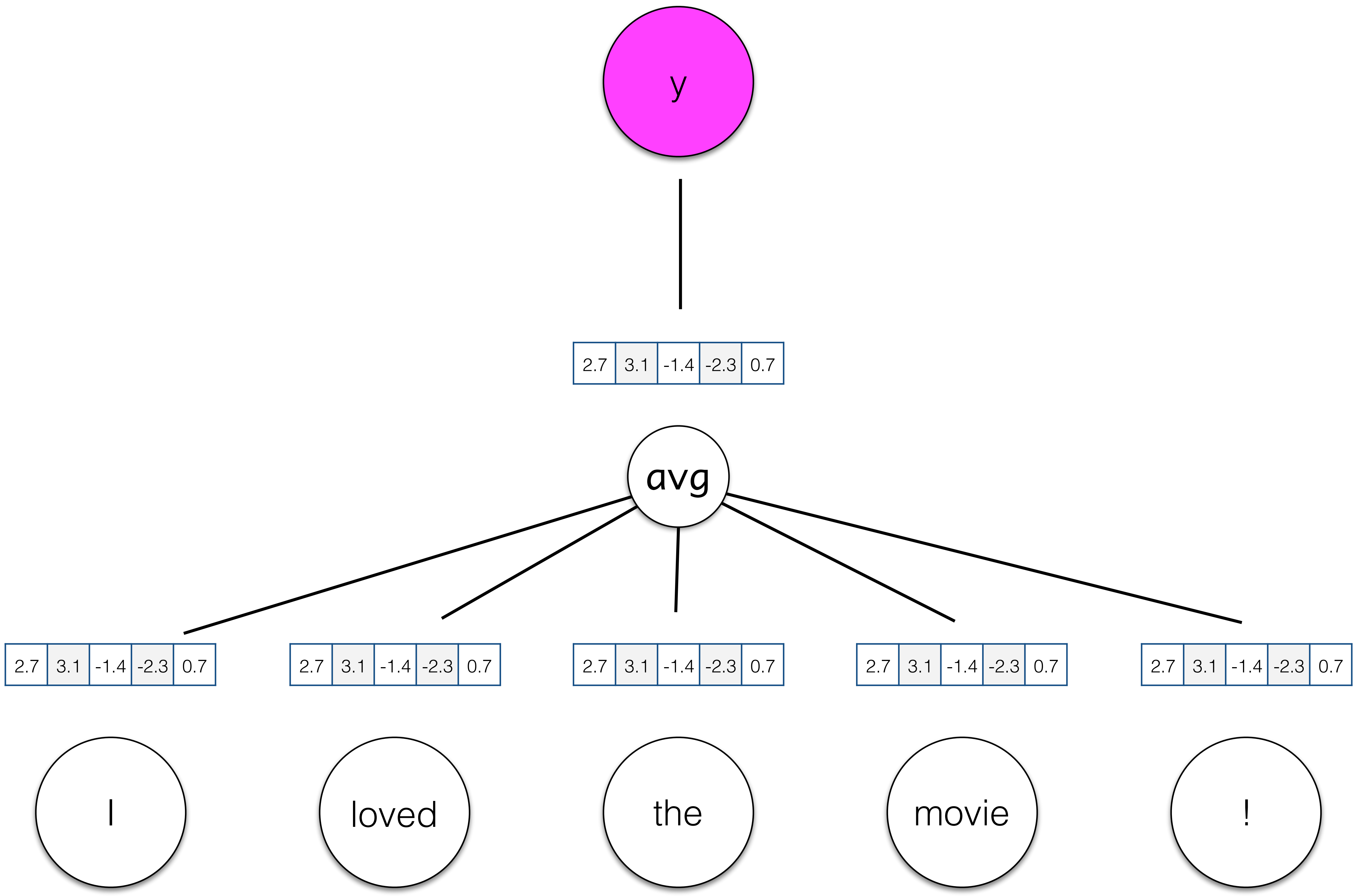


Elman 1990, Mikolov 2012

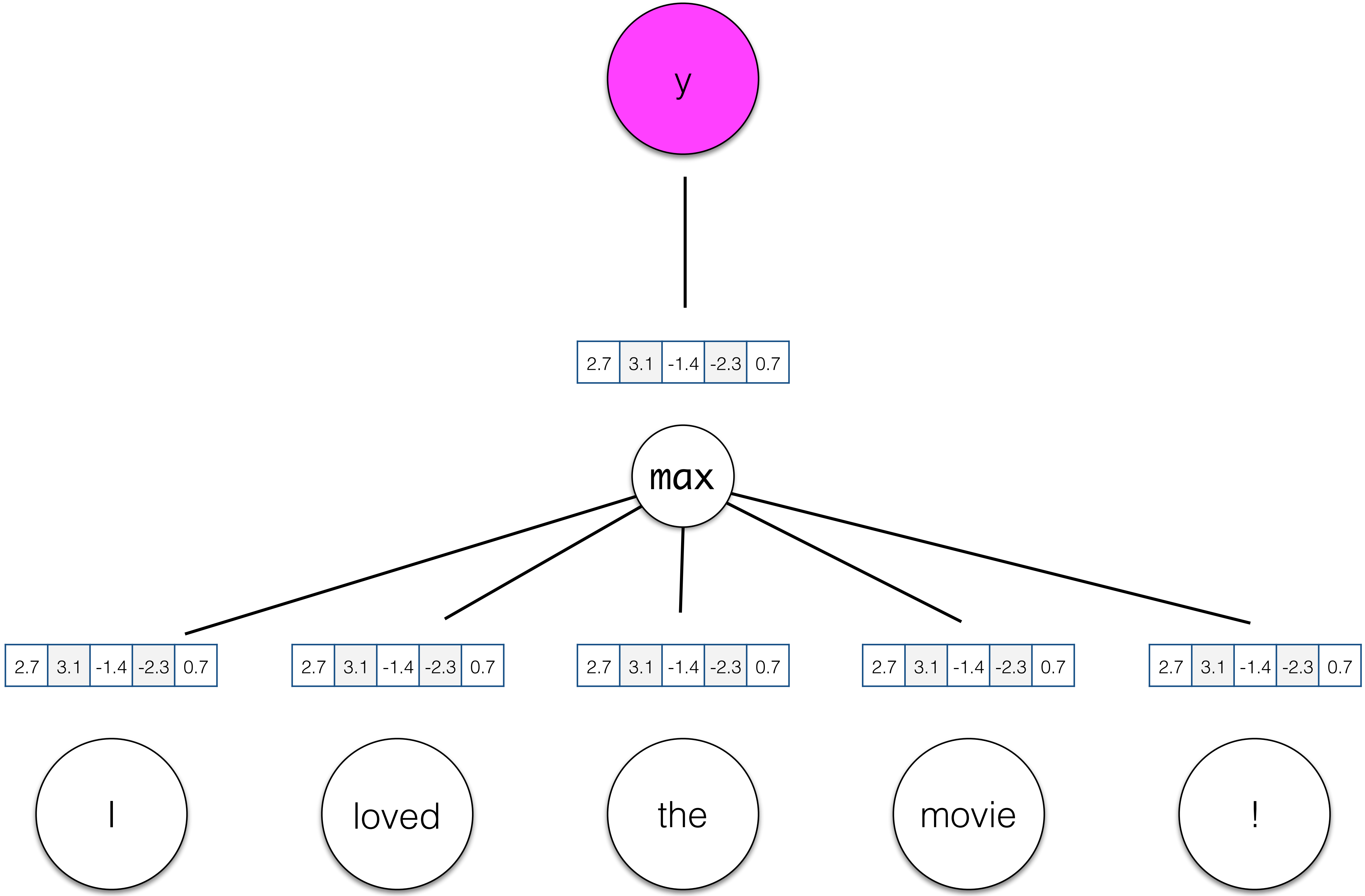
How do we use RNNs for document classification?



y1
y2
y3
y4
y5
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!
x1
x2
x3
x4
x5
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
s1
s2
s3
s4
s5

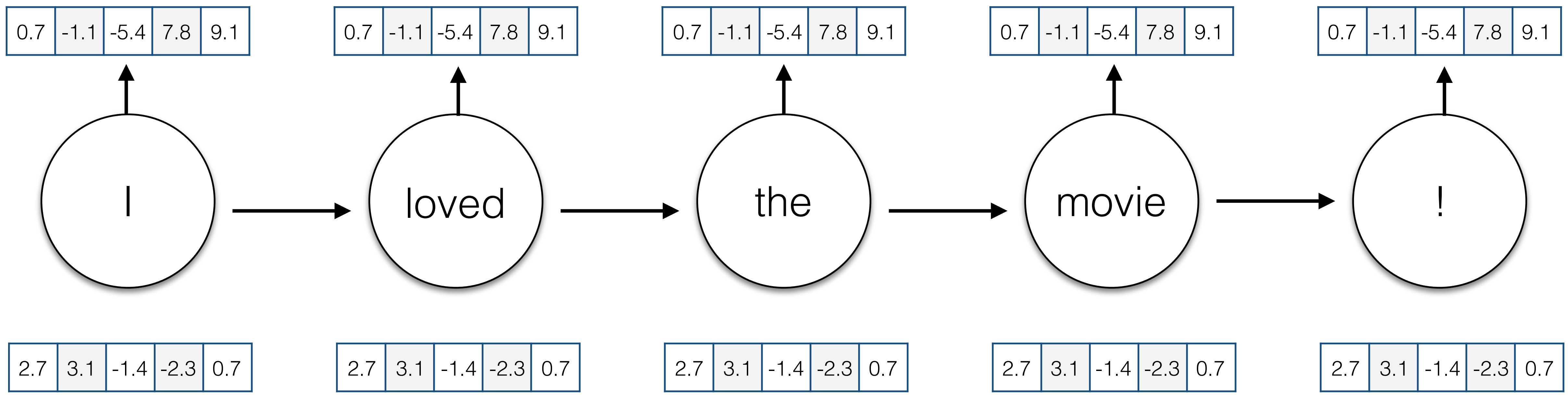


Iyyer et al. (2015), "Deep Unordered Composition Rivals Syntactic Methods for Text Classification" (ACL)

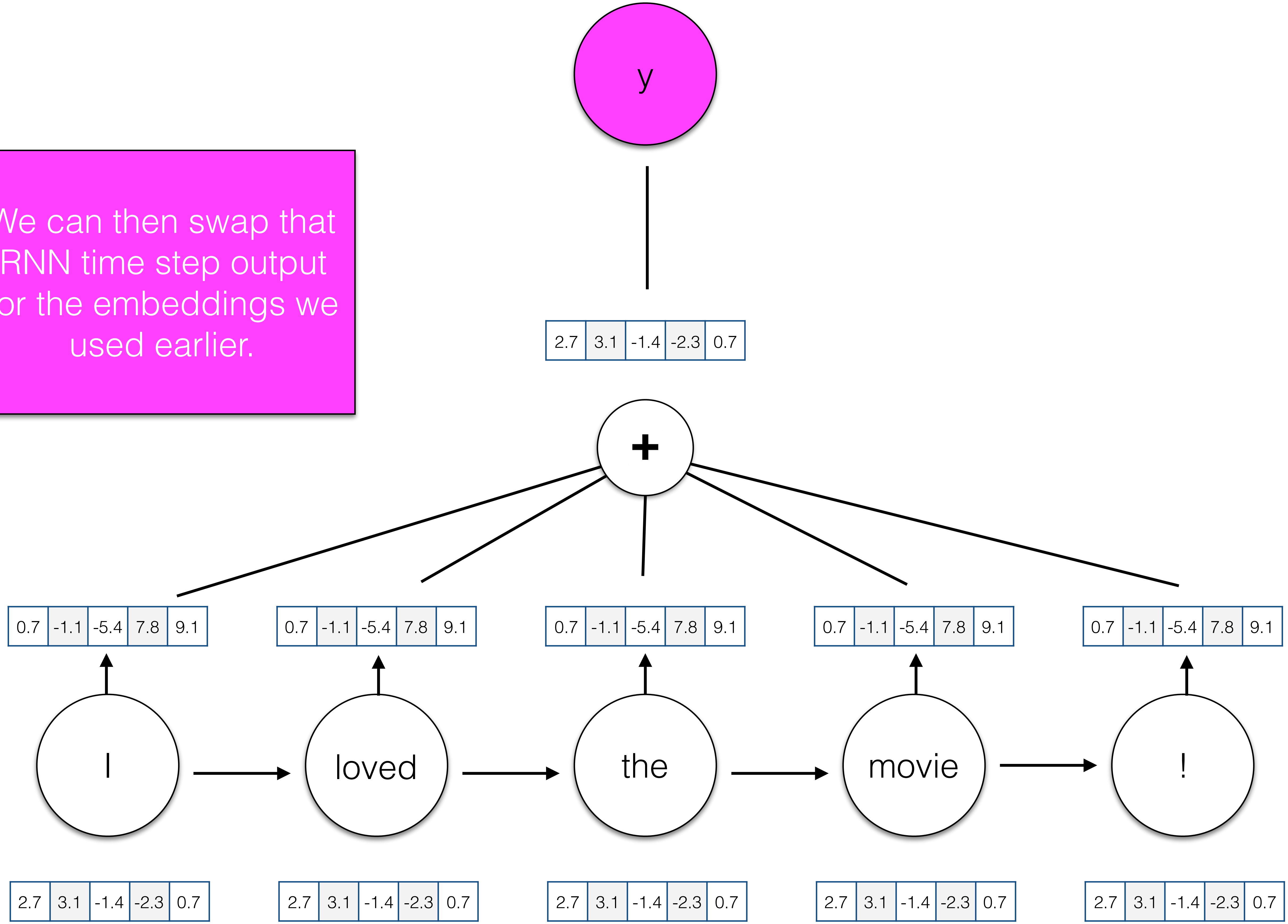

y
2.7 3.1 -1.4 -2.3 0.7
max
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!

RNN

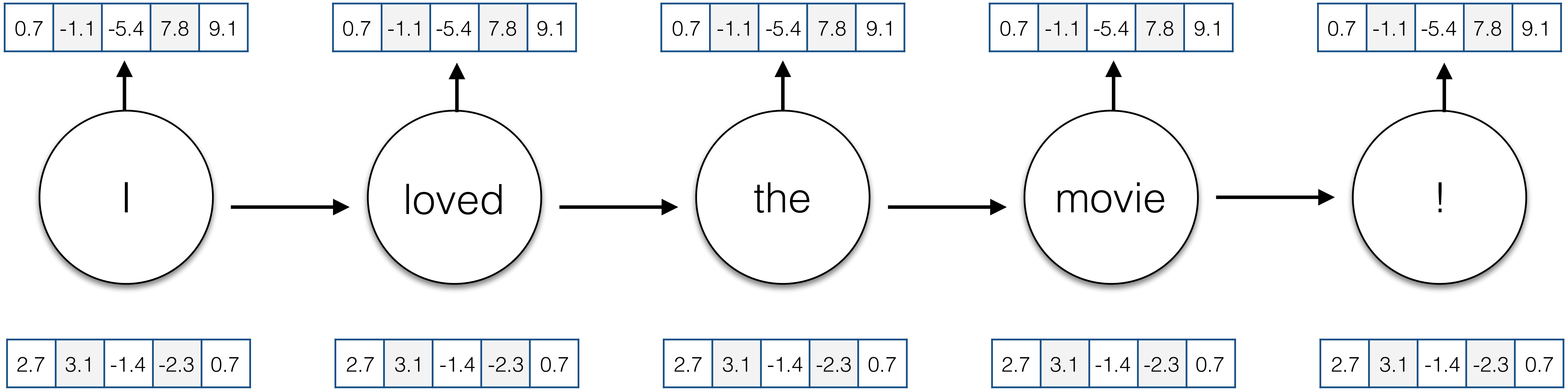
- With an RNN, we can generate a representation of the sequence as seen through time t.
- This encodes a representation of meaning specific to the local context a word is used in.



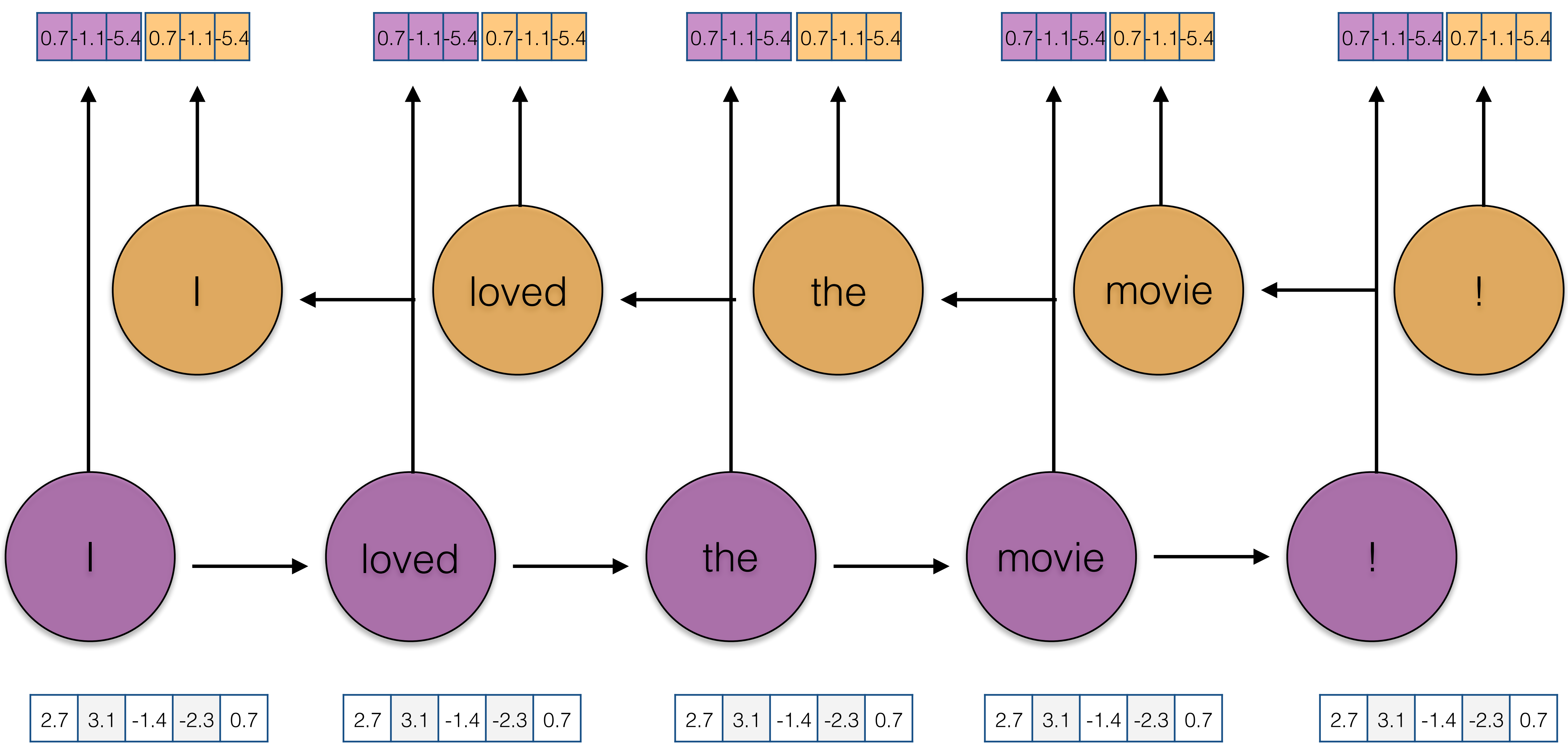
We can then swap that RNN time step output for the embeddings we used earlier.



What about the future context?



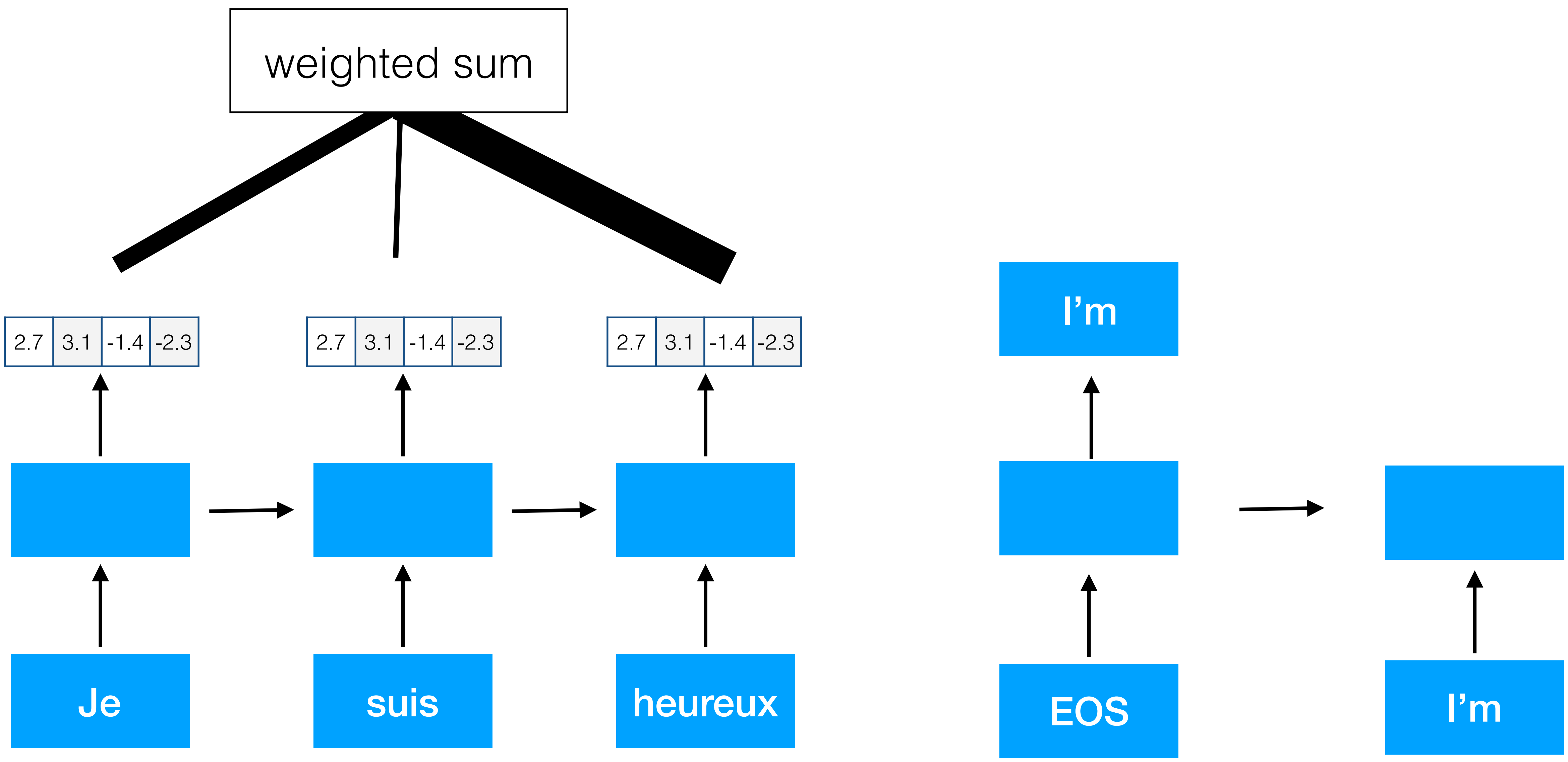
Bidirectional RNN



Attention

- Let’s incorporate structure (and parameters) into a network that captures which elements in the input we should be attending to (and which we can ignore).

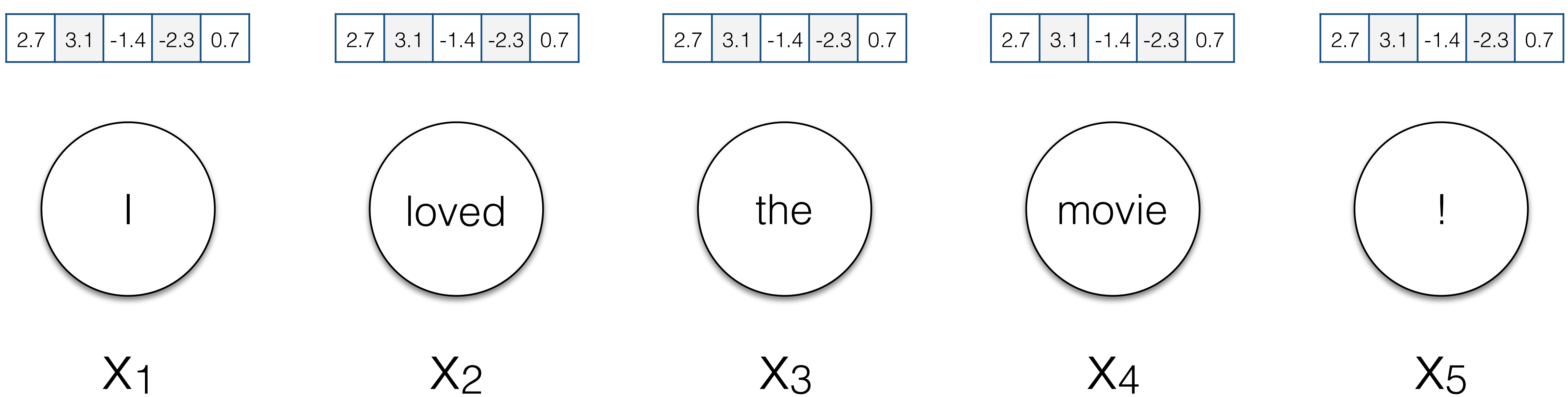
Machine translation



$v \in \mathscr{R}^H$

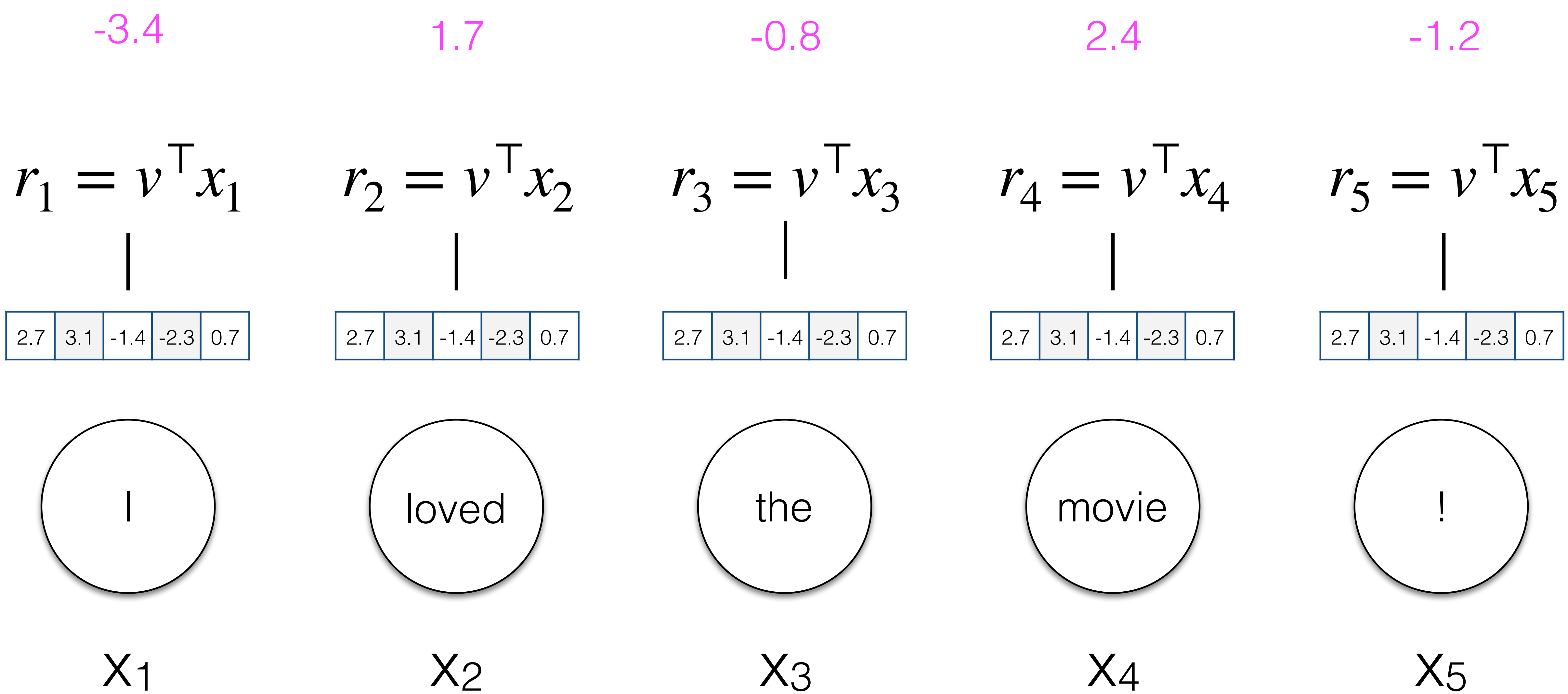
2.7	3.1	-1.4	-2.3	0.7

Define v to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector



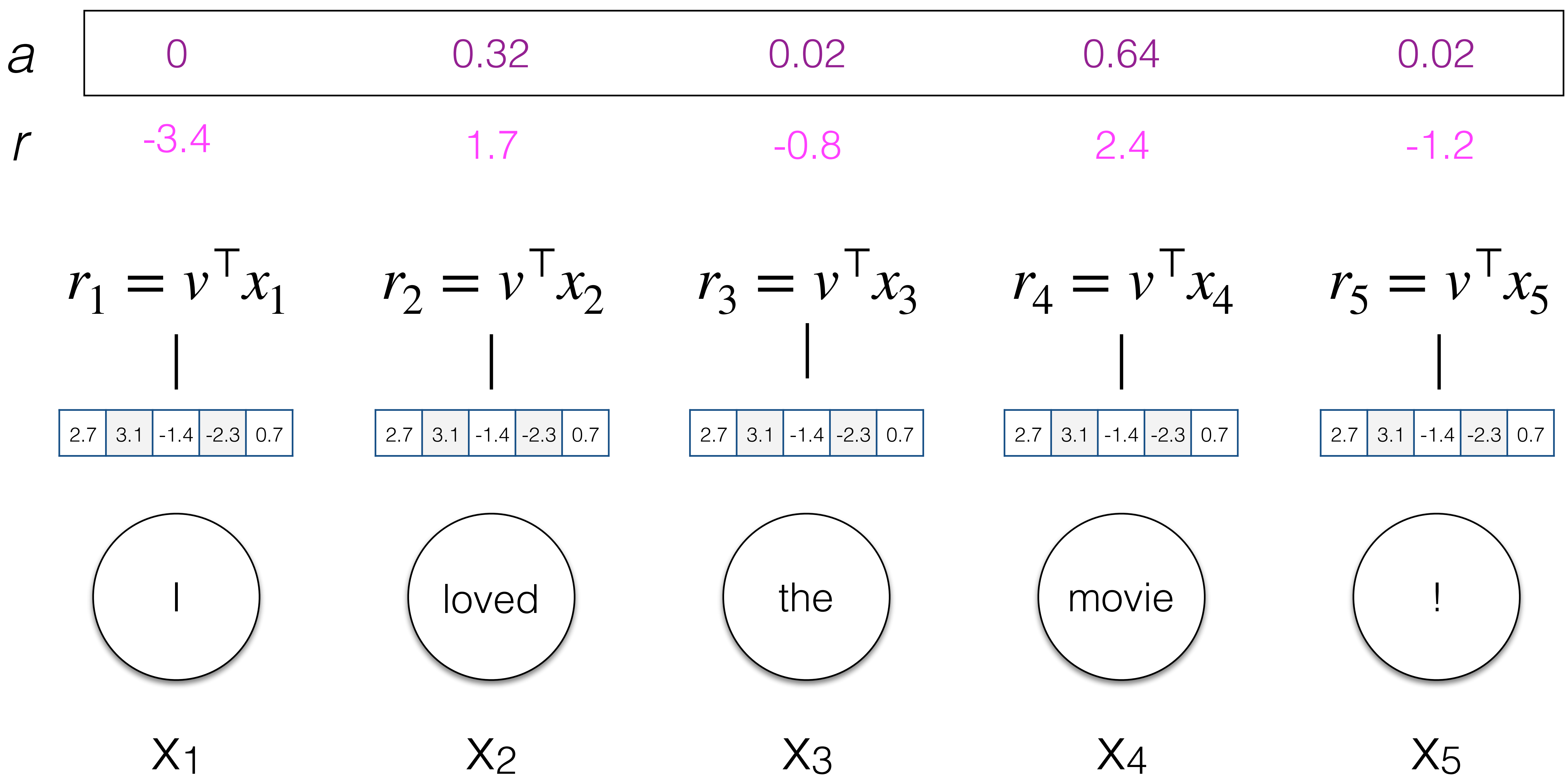
$v \in \mathcal{R}^H$

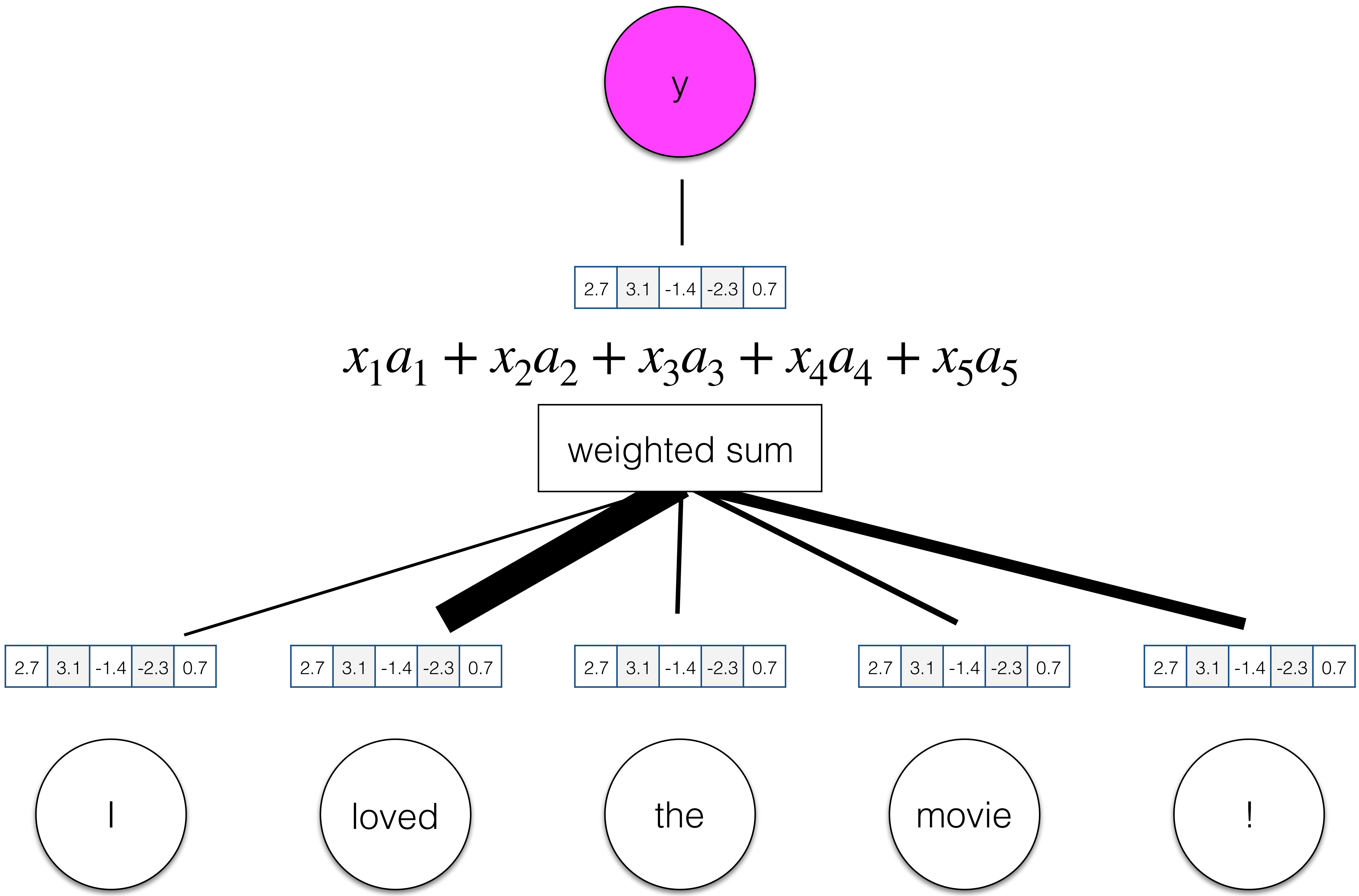
2.7	3.1	-1.4	-2.3	0.7



Convert r into a vector of normalized weights that sum to 1.

$$a = \mathrm{softmax}(r)$$




y
2.7 3.1 -1.4 -2.3 0.7
$x_1a_1 + x_2a_2 + x_3a_3 + x_4a_4 + x_5a_5$
weighted sum
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!

Attention

- For a document with n words and LSTM output e (= n x d dimensions)
- Dot product between each e_k and attention vector v to yield one r_k for k = [1, … n] (r = n dimensions)
- a=softmax(r) (= n dimensions)
- Multiply a * e to generate document representation (= d dimensions)

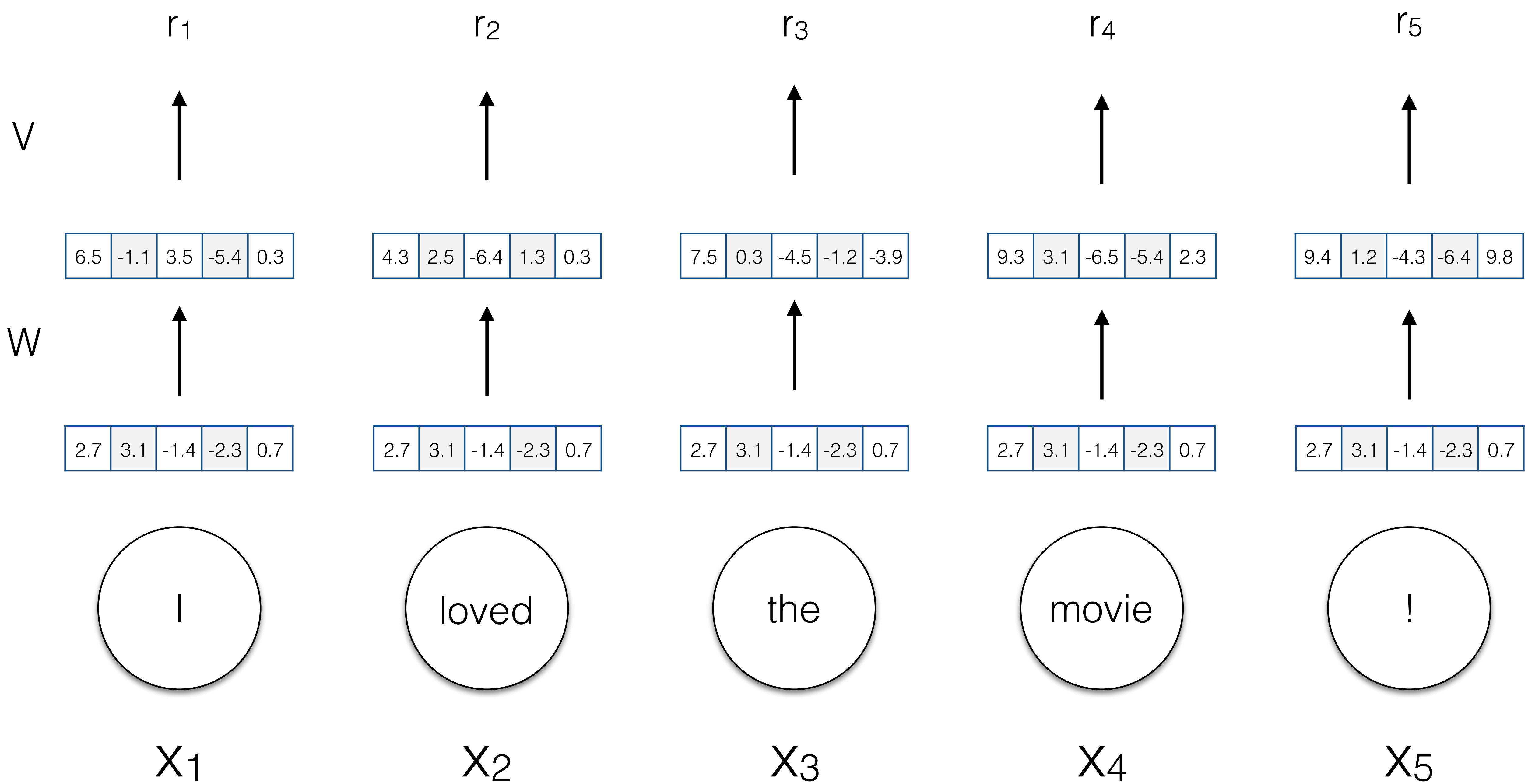
Attention

- Lots of variations on attention:
  - Linear transformation of x into before dotting with v

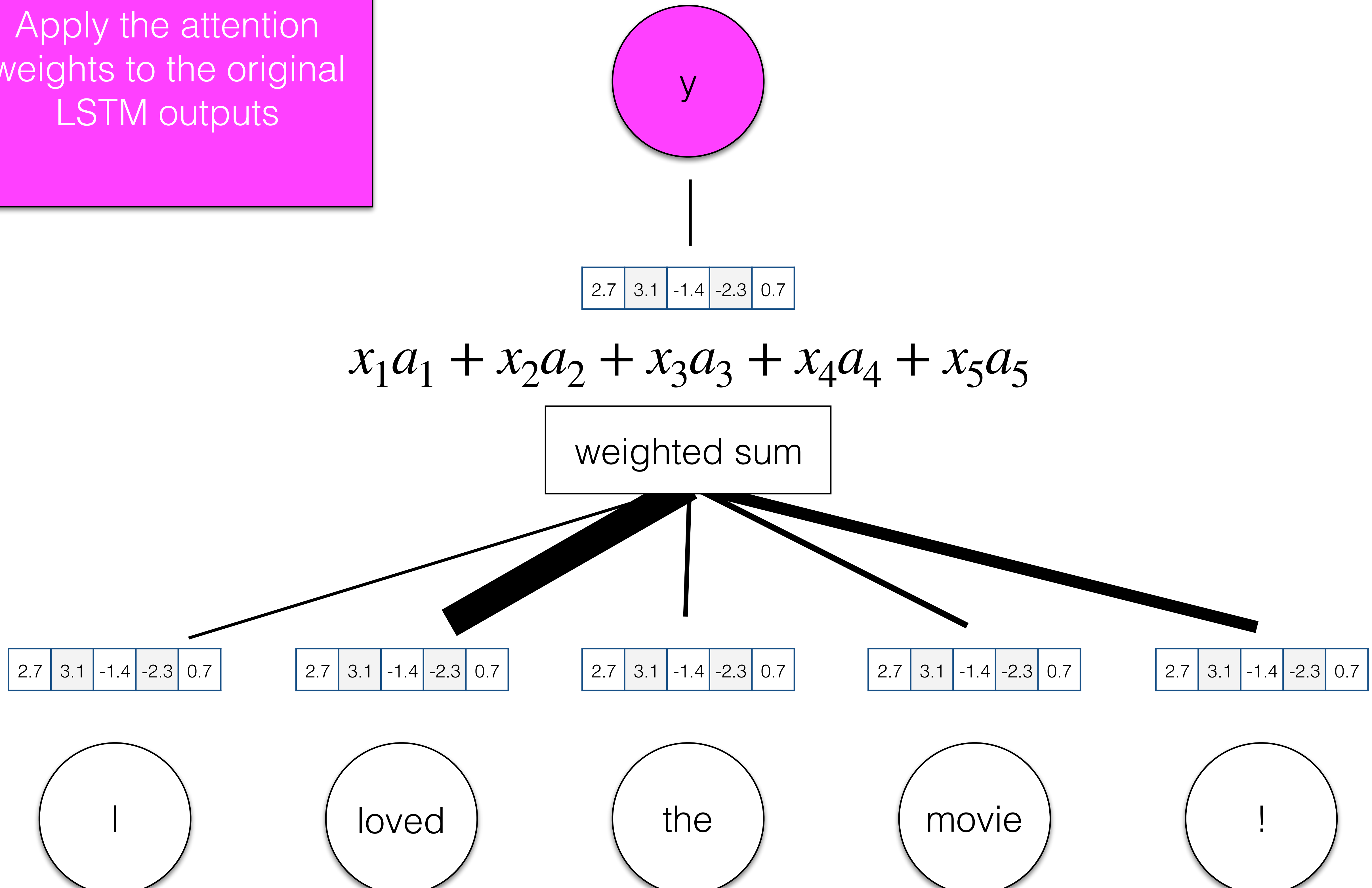
Pass the LSTM output through a mini neural network to generate r

0.1	0.5	0.1	0.2	0.1

$$a = \text{softmax}(r)$$



Apply the attention weights to the original LSTM outputs



Attention

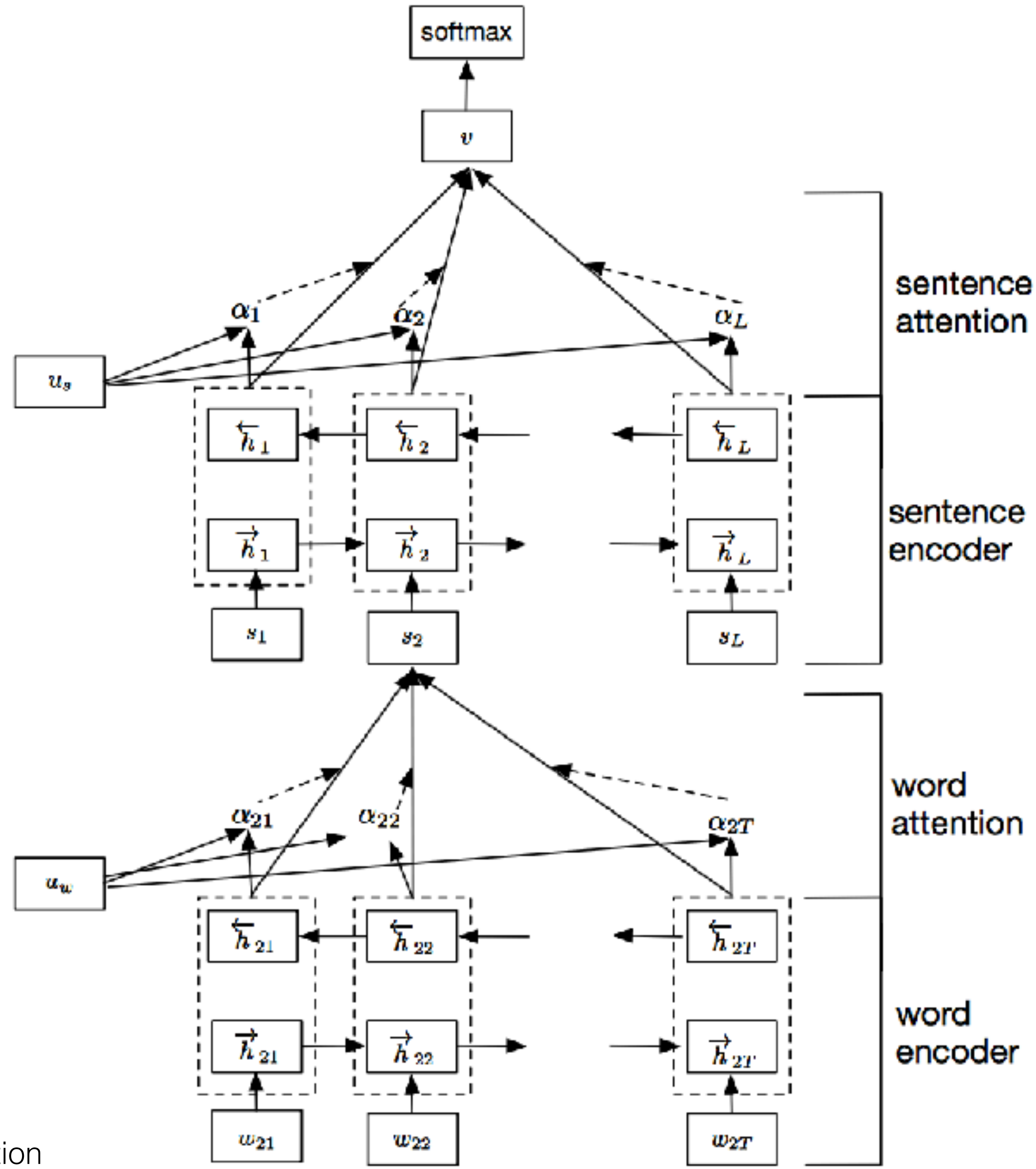
- Lots of variations on attention:
  - Linear transformation of x into before dotting with v
  - Non-linearities after each operation.
  - "Multi-head attention": multiple v vectors to capture different phenomena that can be attended to in the input.
  - Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).

attention over sentences

bidirectional GRU over sentence representations

attention over words

bidirectional GRU over word representations



Yang et al. (2016), “Hierarchical Attention Networks for Document Classification”

Attention

- Attention gives us a normalized weight for every token in a sequence that tells us how important that word was for the prediction

- This can be useful for visualization