

---

# Neural Networks Language Models

Philipp Koehn

1 October 2020



# N-Gram Backoff Language Model



- Previously, we approximated

$$p(W) = p(w_1, w_2, \dots, w_n)$$

- ... by applying the chain rule

$$p(W) = \sum_i p(w_i | w_1, \dots, w_{i-1})$$

- ... and limiting the history (Markov order)

$$p(w_i | w_1, \dots, w_{i-1}) \simeq p(w_i | w_{i-4}, w_{i-3}, w_{i-2}, w_{i-1})$$

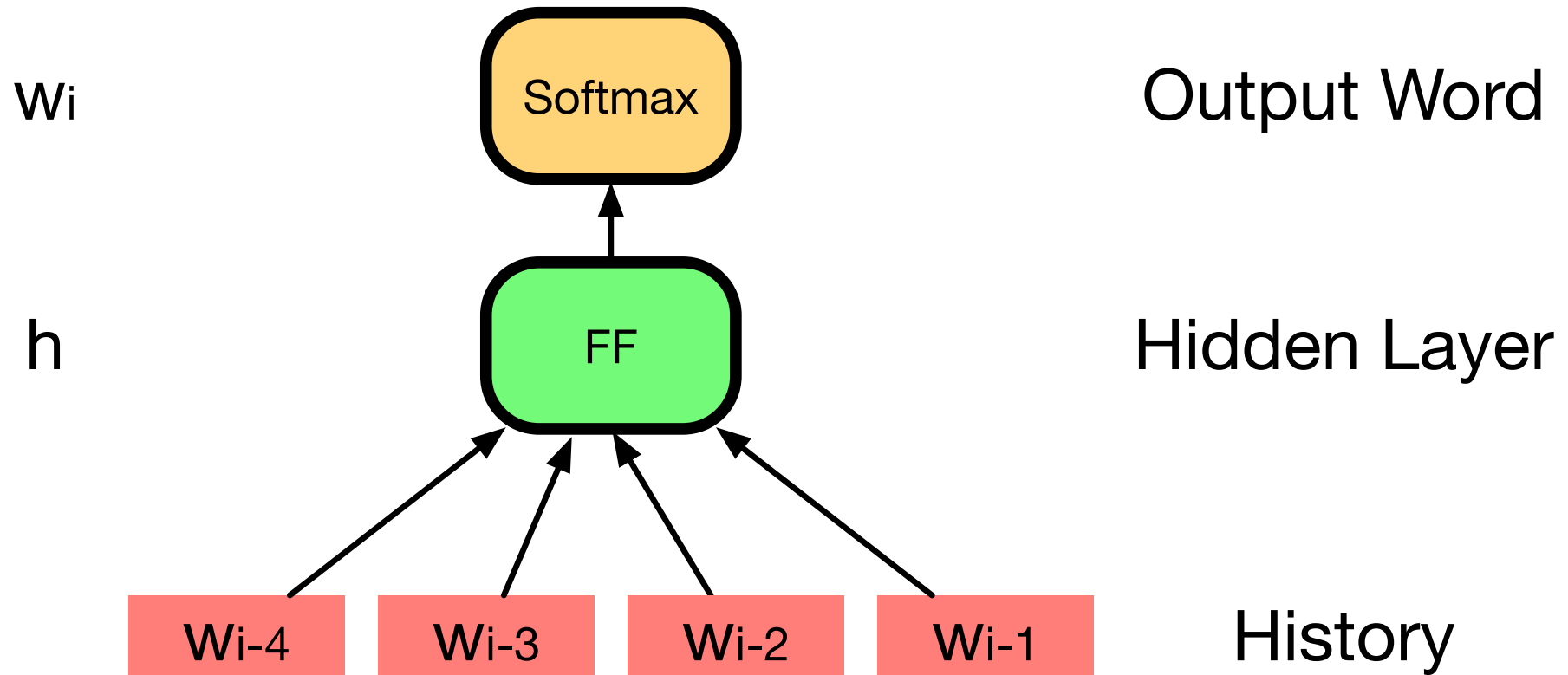
- Each  $p(w_i | w_{i-4}, w_{i-3}, w_{i-2}, w_{i-1})$  may not have enough statistics to estimate
  - we back off to  $p(w_i | w_{i-3}, w_{i-2}, w_{i-1})$ ,  $p(w_i | w_{i-2}, w_{i-1})$ , etc., all the way to  $p(w_i)$
  - exact details of backing off get complicated — “interpolated Kneser-Ney”

- A whole family of back-off schemes
- Skip-n gram models that may back off to  $p(w_i|w_{i-2})$
- Class-based models  $p(C(w_i)|C(w_{i-4}), C(w_{i-3}), C(w_{i-2}), C(w_{i-1}))$

⇒ We are wrestling here with

- using as much relevant evidence as possible
- pooling evidence between words

# First Sketch



# Representing Words



4

- Words are represented with a one-hot vector, e.g.,
  - **dog** = (0,0,0,0,1,0,0,0,0,...)
  - **cat** = (0,0,0,0,0,0,0,1,0,...)
  - **eat** = (0,1,0,0,0,0,0,0,0,...)
- That's a large vector!
- Remedies
  - limit to, say, 20,000 most frequent words, rest are OTHER
  - place words in  $\sqrt{n}$  classes, so each word is represented by
    - \* 1 class label
    - \* 1 word in class label

# Word Classes for Two-Hot Representations

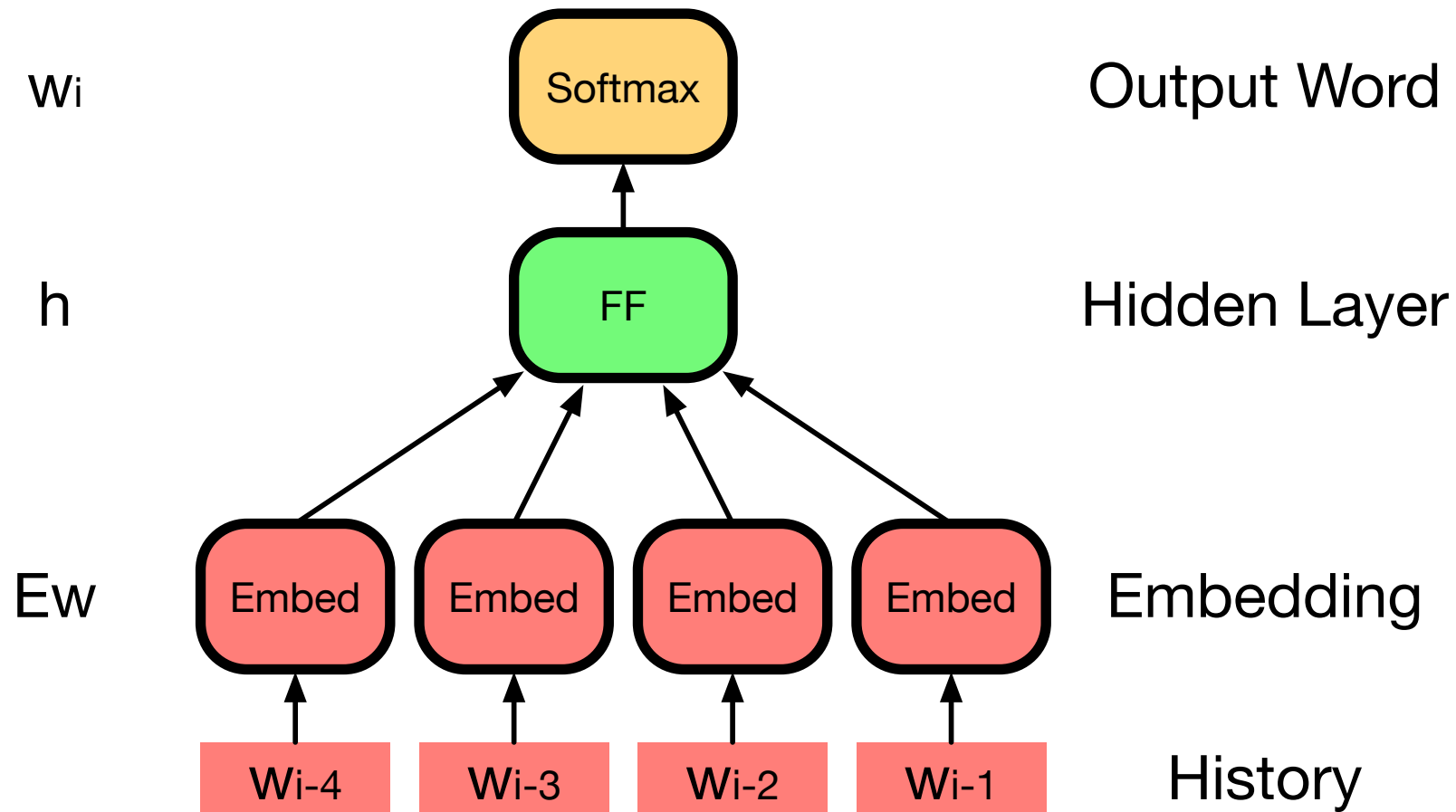


5

- WordNet classes
- Brown clusters
- Frequency binning
  - sort words by frequency
  - place them in order into classes
  - each class has same token count
  - very frequent words have their own class
  - rare words share class with many other words
- Anything goes: assign words randomly to classes

# word embeddings

# Add a Hidden Layer



- Map each word first into a lower-dimensional real-valued space
- Shared weight matrix  $E$

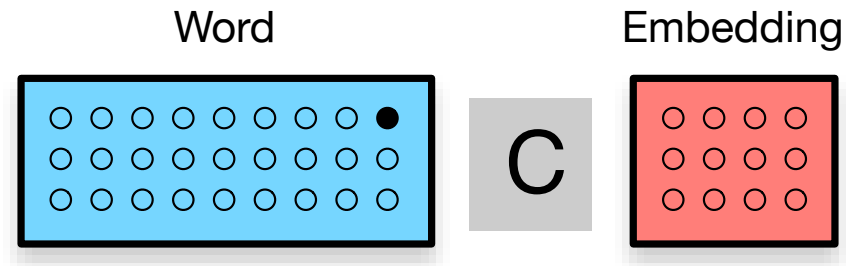


# Details (Bengio et al., 2003)



- Add direct connections from embedding layer to output layer
- Activation functions
  - input→embedding: none
  - embedding→hidden: tanh
  - hidden→output: softmax
- Training
  - loop through the entire corpus
  - update between predicted probabilities and 1-hot vector for output word

# Word Embeddings



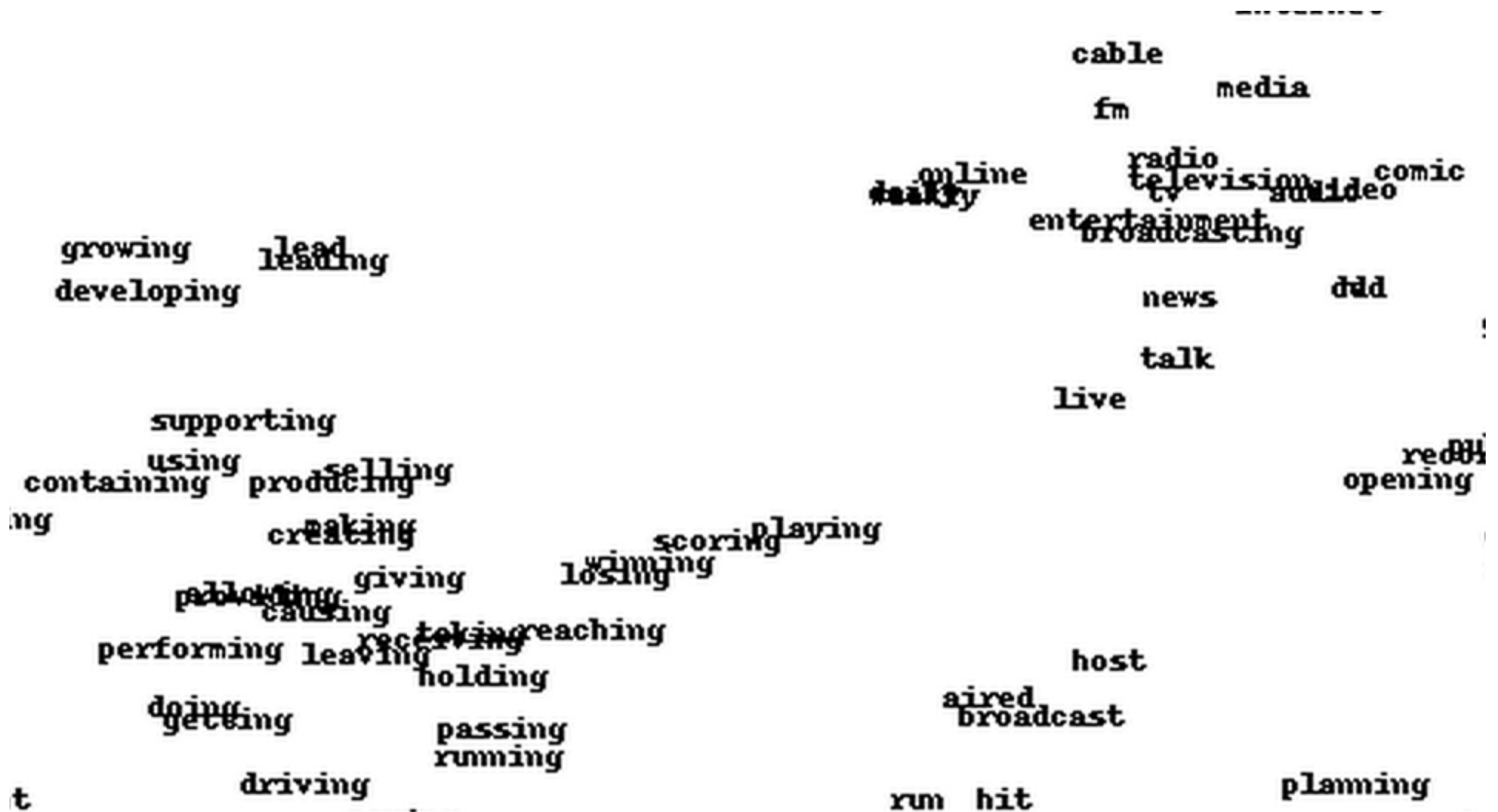
- By-product: embedding of word into continuous space
- Similar contexts  $\rightarrow$  similar embedding
- Recall: distributional semantics

10

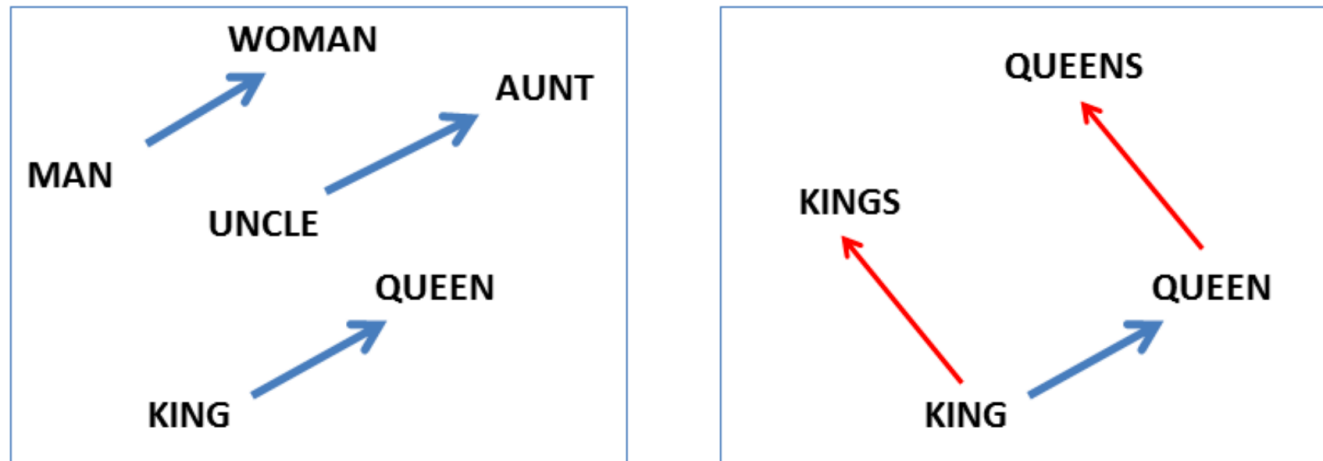


# Word Embeddings

11



# Are Word Embeddings Magic?



- Morphosyntactic regularities (Mikolov et al., 2013)
  - adjectives base form vs. comparative, e.g., **good**, **better**
  - nouns singular vs. plural, e.g., **year**, **years**
  - verbs present tense vs. past tense, e.g., **see**, **saw**
- Semantic regularities
  - **clothing** is to **shirt** as **dish** is to **bowl**
  - evaluated on human judgment data of semantic similarities

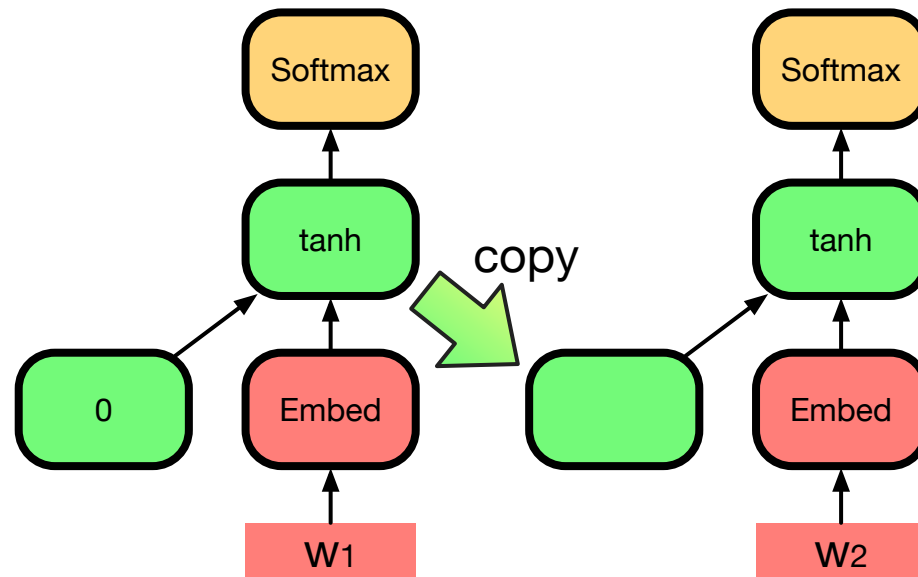
# recurrent neural networks

# Recurrent Neural Networks



- Start: predict second word from first
- Mystery layer with nodes all with value 1

# Recurrent Neural Networks



Output Word

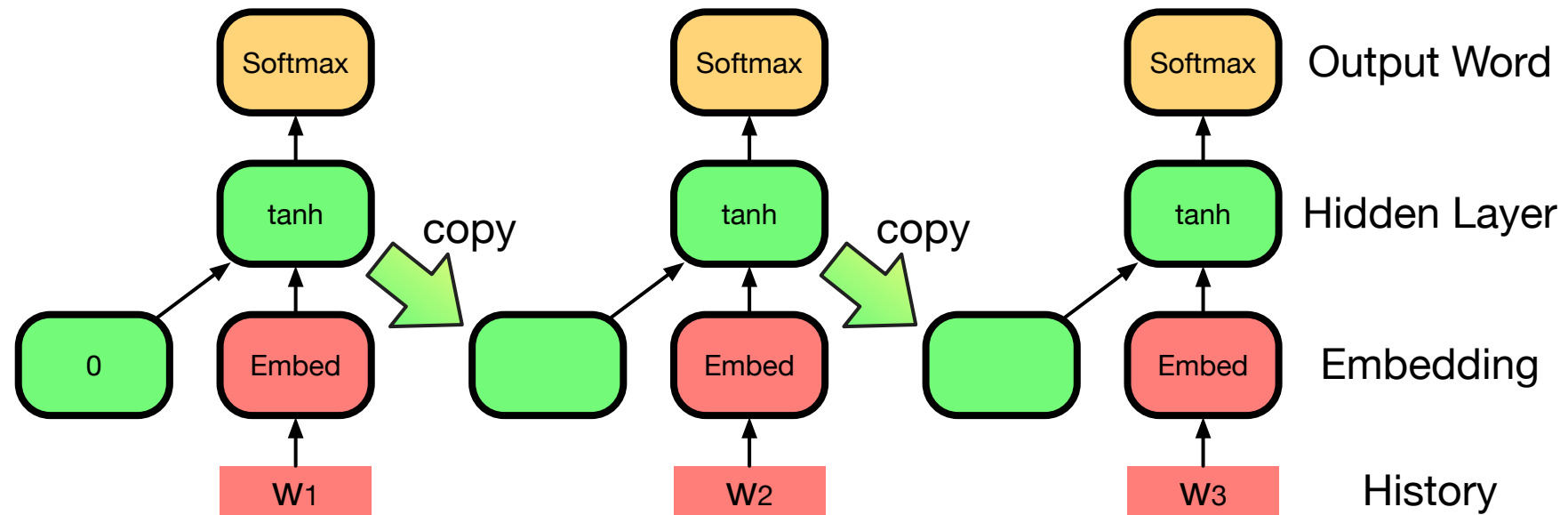
Hidden Layer

Embedding

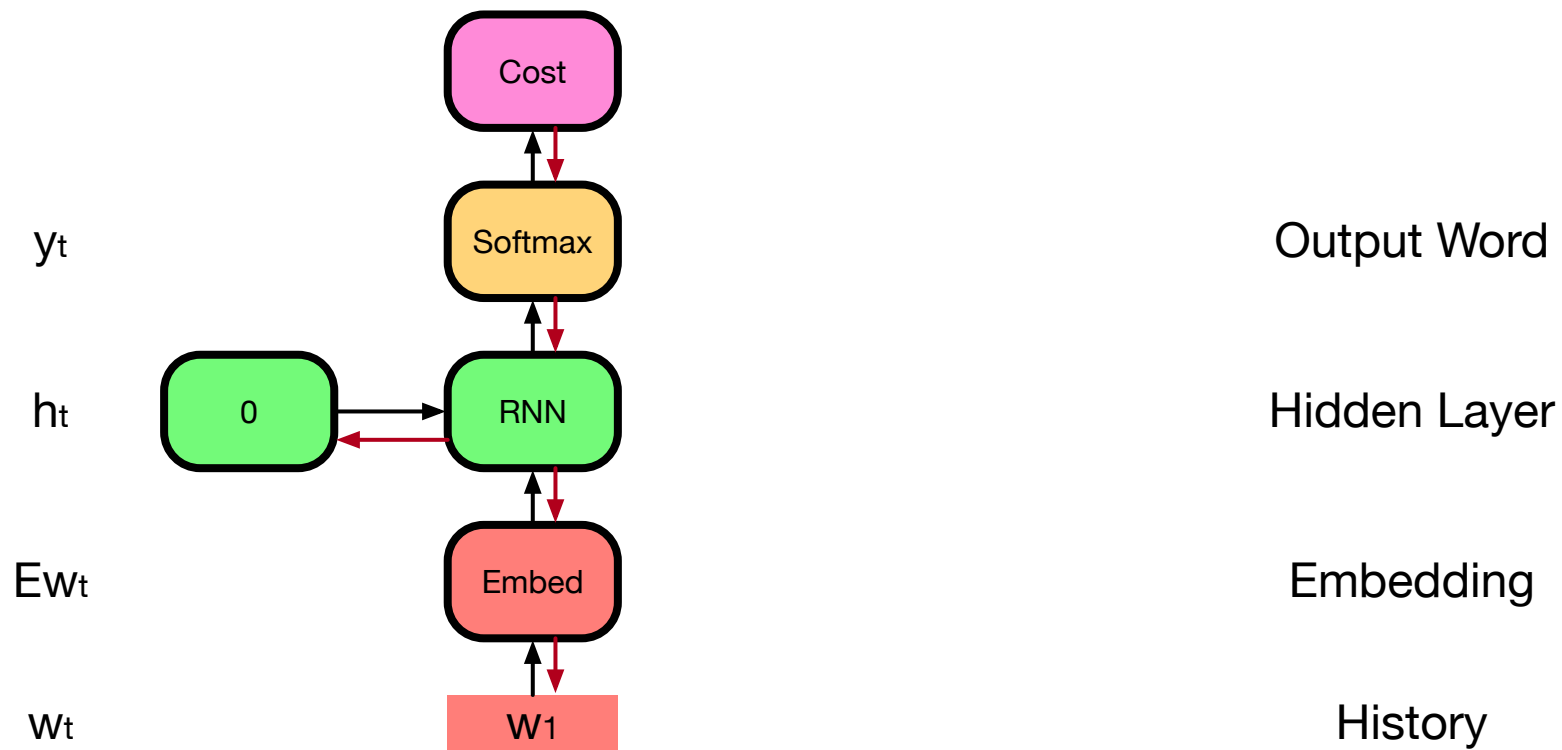
History



# Recurrent Neural Networks

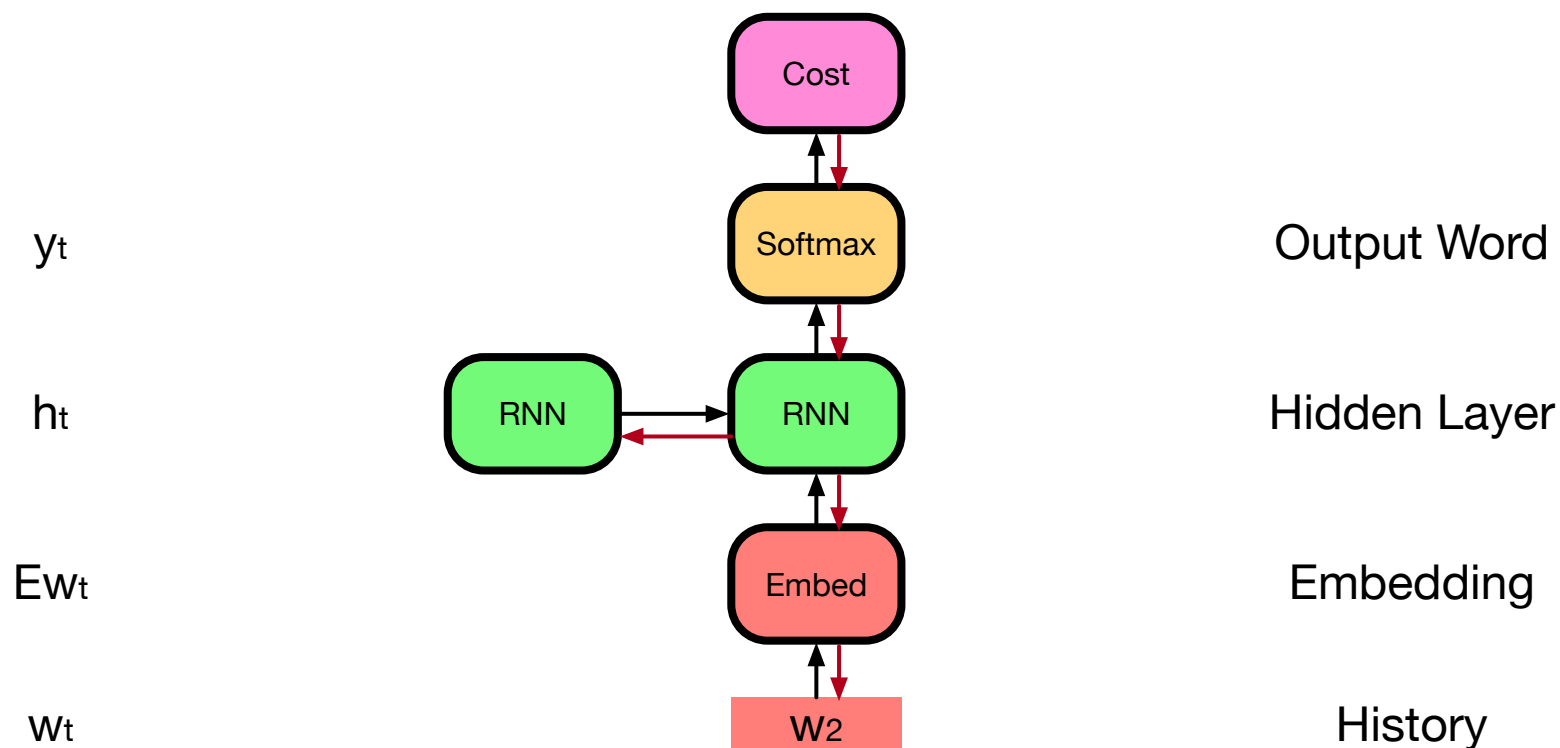



# Training



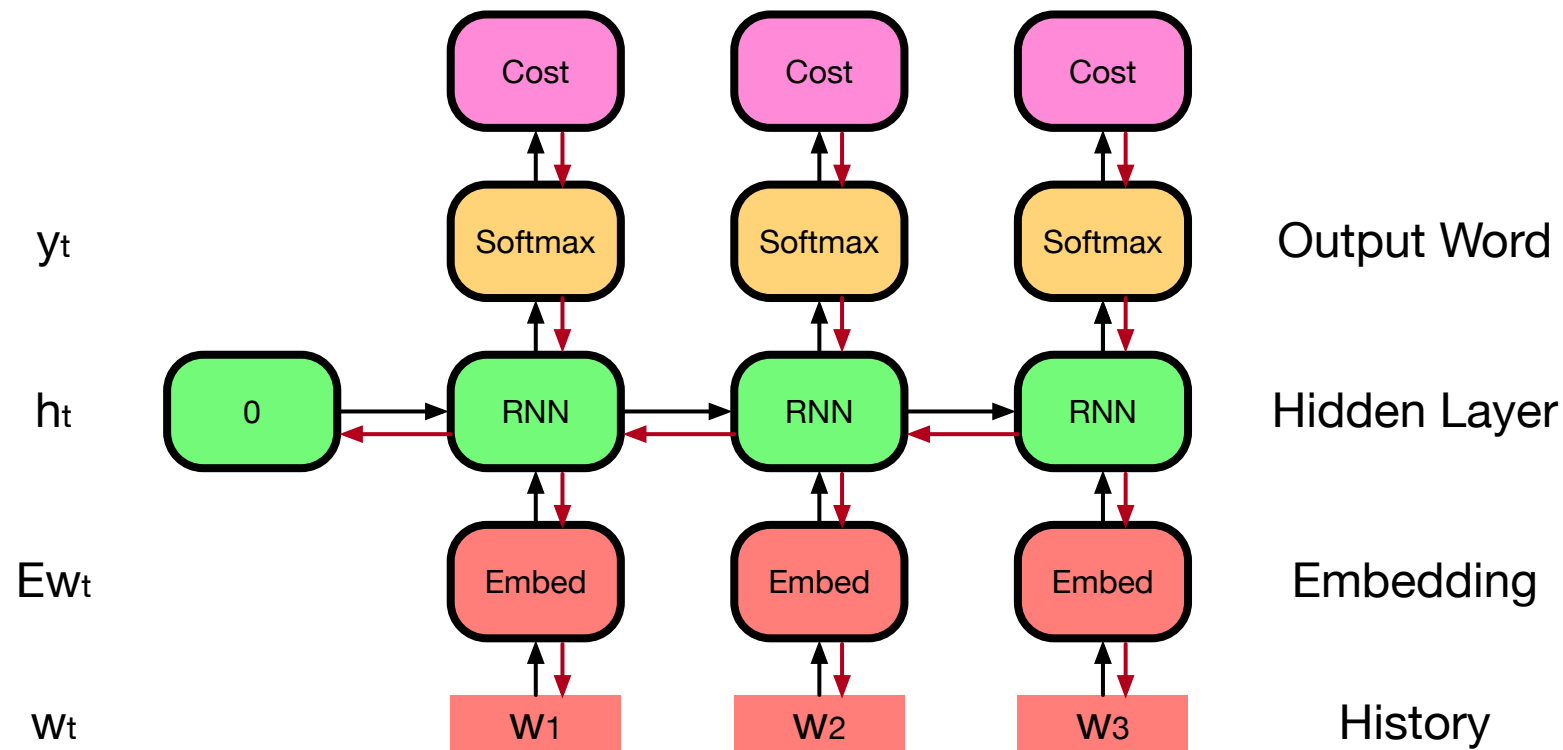
- Process first training example
- Update weights with back-propagation

# Training



- Process second training example
- Update weights with back-propagation
- And so on... 
- But: no feedback to previous history

# Back-Propagation Through Time



- After processing a few training examples, update through the unfolded recurrent neural network

# Back-Propagation Through Time

- Carry out back-propagation through time (BPTT) after each training example
  - 5 time steps seems to be sufficient
  - network learns to store information for more than 5 time steps
- Or: update in mini-batches
  - process 10-20 training examples
  - update backwards through all examples
  - removes need for multiple steps for each training example

# long short term memory

# Vanishing Gradients

- Error is propagated to previous steps
- Updates consider
  - prediction at that time step
  - impact on future time steps
- Vanishing gradient: propagated error disappears

# Recent vs. Early History

- Hidden layer plays double duty
  - memory of the network
  - continuous space representation used to predict output words

- Sometimes only recent context important

*After much economic progress over the years, the **country** → has*

- Sometimes much earlier context important

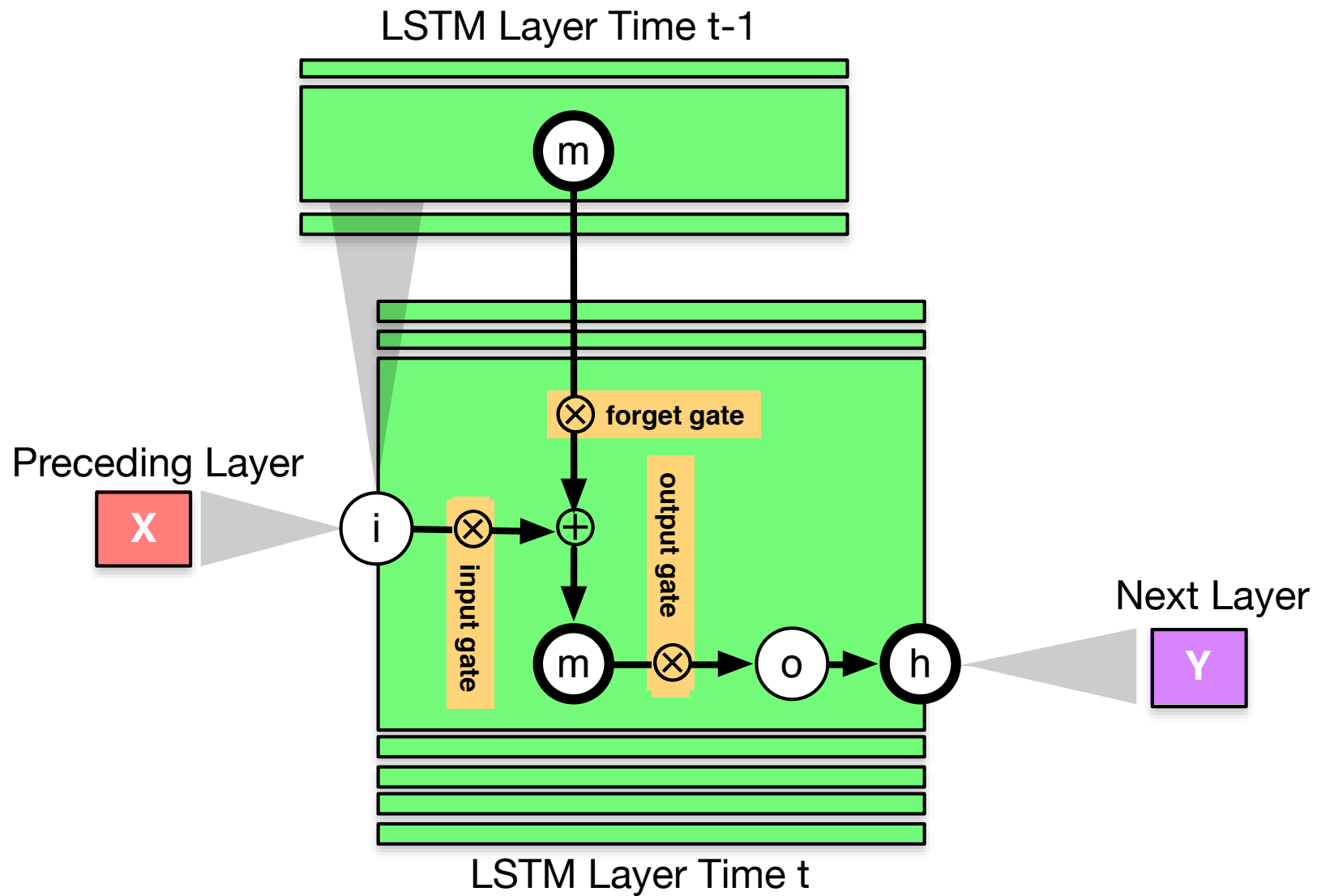
*The **country** which has made much economic progress over the years still → has*



# Long Short Term Memory (LSTM)

- Design quite elaborate, although not very complicated to use
- Basic building block: **LSTM cell**
  - similar to a node in a hidden layer
  - but: has an explicit memory state
- Output and memory state change depends on gates
  - **input gate**: how much new input changes memory state
  - **forget gate**: how much of prior memory state is retained
  - **output gate**: how strongly memory state is passed on to next layer.
- Gates can be not just be open (1) and closed (0), but slightly ajar (e.g., 0.2)

# LSTM Cell



# LSTM Cell (Math)

- Memory and output values at time step  $t$

$$\text{memory}^t = \text{gate}_{\text{input}} \times \text{input}^t + \text{gate}_{\text{forget}} \times \text{memory}^{t-1}$$

$$\text{output}^t = \text{gate}_{\text{output}} \times \text{memory}^t$$

- Hidden node value  $h^t$  passed on to next layer applies activation function  $f$

$$h^t = f(\text{output}^t)$$

- Input computed as input to recurrent neural network node

- given node values for prior layer  $\vec{x}^t = (x_1^t, \dots, x_X^t)$
- given values for hidden layer from previous time step  $\vec{h}^{t-1} = (h_1^{t-1}, \dots, h_H^{t-1})$
- input value is combination of matrix multiplication with weights  $w^x$  and  $w^h$  and activation function  $g$

$$\text{input}^t = g \left( \sum_{i=1}^X w_i^x x_i^t + \sum_{i=1}^H w_i^h h_i^{t-1} \right)$$

# Values for Gates

- Gates are very important
- How do we compute their value?  
→ with a neural network layer!
- For each gate  $a \in (\text{input, forget, output})$ 
  - weight matrix  $W^{xa}$  to consider node values in previous layer  $\vec{x}^t$
  - weight matrix  $W^{ha}$  to consider hidden layer  $\vec{h}^{t-1}$  at previous time step
  - weight matrix  $W^{ma}$  to consider memory at previous time step  $\text{memory}^{t-1}$
  - activation function  $h$

$$\text{gate}_a = h \left( \sum_{i=1}^X w_i^{xa} x_i^t + \sum_{i=1}^H w_i^{ha} h_i^{t-1} + \sum_{i=1}^H w_i^{ma} \text{memory}_i^{t-1} \right)$$

- LSTM are trained the same way as recurrent neural networks
- Back-propagation through time
- This looks all very complex, but:
  - all the operations are still based on
    - \* matrix multiplications
    - \* differentiable activation functions
- we can compute gradients for objective function with respect to all parameters
- we can compute update functions

# What is the Point?

- (a) wie wirksam die daraus resultierende strategie sein wird , hängt daher von der genauigkeit dieser annahmen **ab**  
**Gloss:** *how effective the from-that resulting strategy be will, depends therefore on the accuracy of-these measures*  
**Translation:** *how effective the resulting strategy will be, therefore, depends on the accuracy of these measures*
- (b) ... die lage versetzen werden , eine schlüsselrolle bei der eindämmung der regionalen ambitionen chinas zu **spielen**  
**Gloss:** *... the position place will, a key-role in the curbing of-the regional ambitions China's to play*  
**Translation:** *...which will put him in a position to play a key role in curbing the regional ambitions of China*
- (c) ... che fu insignito nel 1692 dall' Imperatore Leopoldo I del **titolo** di Nobile ...  
**Gloss:** *... who was awarded in 1662 by-the Emperor Leopold I of-the title of Noble*  
**Translation:** *... who was awarded the title of Noble by Emperor Leopold I in 1662*

(from Tran, Bisazza, Monz, 2016)

- Each node has memory  $\text{memory}_i$  independent from current output  $h_i$
  - Memory may be carried through unchanged ( $\text{gate}_{\text{input}}^i = 0, \text{gate}_{\text{memory}}^i = 1$ )
- ⇒ can remember important features over long time span  
(capture long distance dependencies)

# Visualizing Individual Cells

30



Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Karpathy et al. (2015): "Visualizing and Understanding Recurrent Networks"

# Visualizing Individual Cells

31



Cell that robustly activates inside if statements:

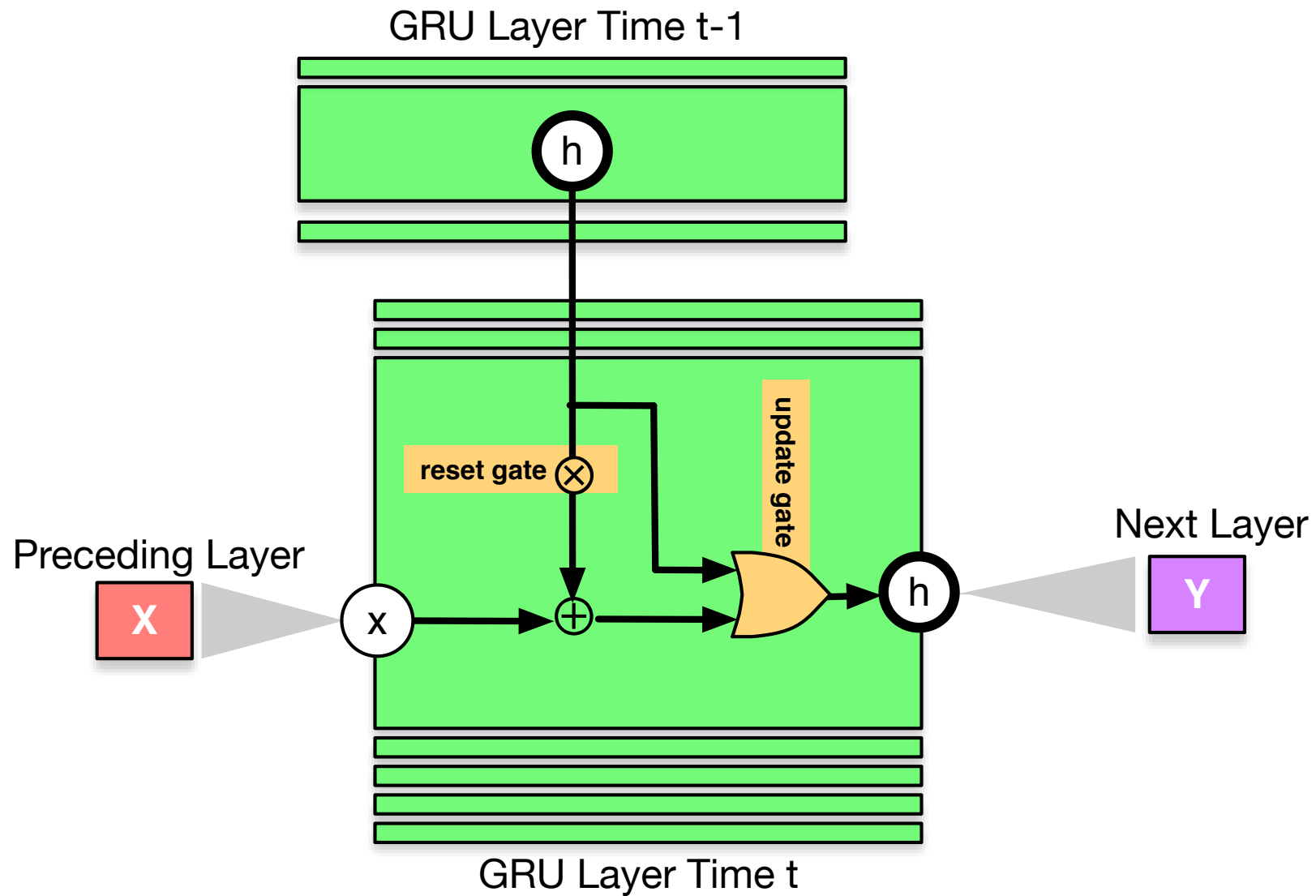
```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
                           siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
     */
}
```



# Gated Recurrent Unit (GRU)



# Gated Recurrent Unit (Math)

- Two Gates

$$\text{update}_t = g(W_{\text{update}} \text{input}_t + U_{\text{update}} \text{state}_{t-1} + \text{bias}_{\text{update}})$$

$$\text{reset}_t = g(W_{\text{reset}} \text{input}_t + U_{\text{reset}} \text{state}_{t-1} + \text{bias}_{\text{reset}})$$

- Combination of input and previous state  
(similar to traditional recurrent neural network)

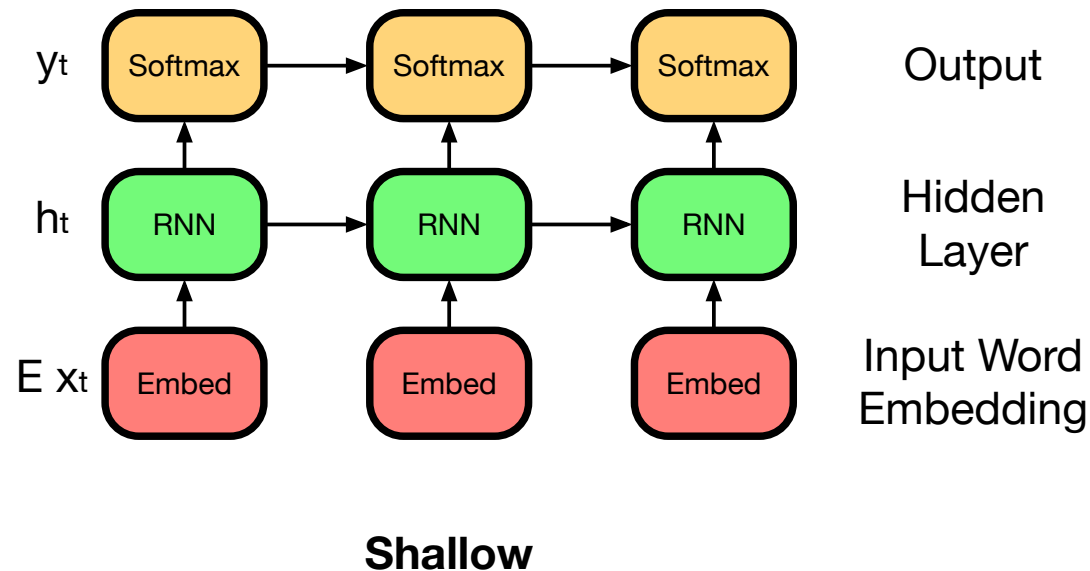
$$\text{combination}_t = f(W \text{input}_t + U(\text{reset}_t \circ \text{state}_{t-1}))$$

- Interpolation with previous state

$$\text{state}_t = (1 - \text{update}_t) \circ \text{state}_{t-1} + \text{update}_t \circ \text{combination}_t + \text{bias}$$

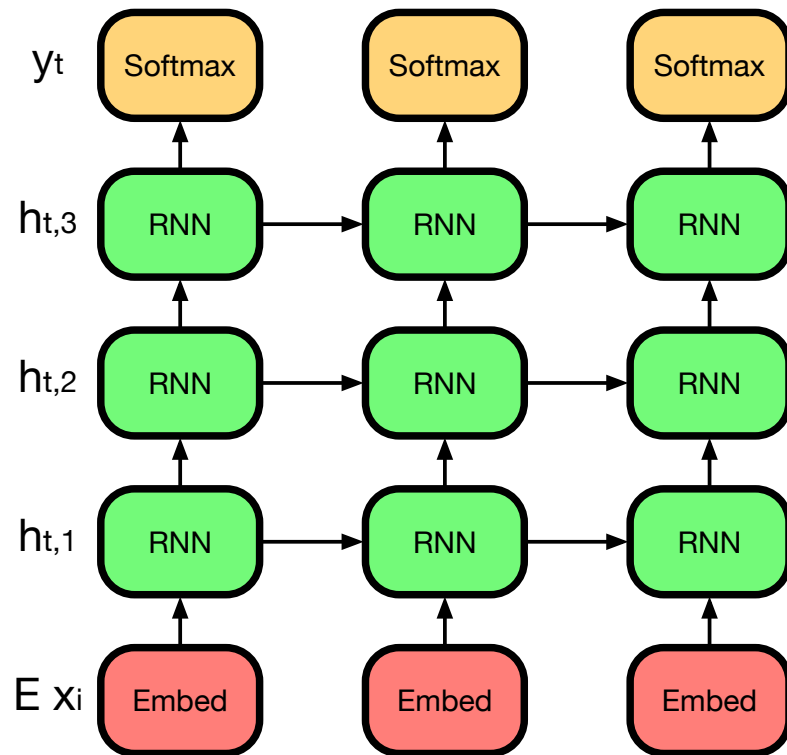
# deeper models

# Deep Learning?

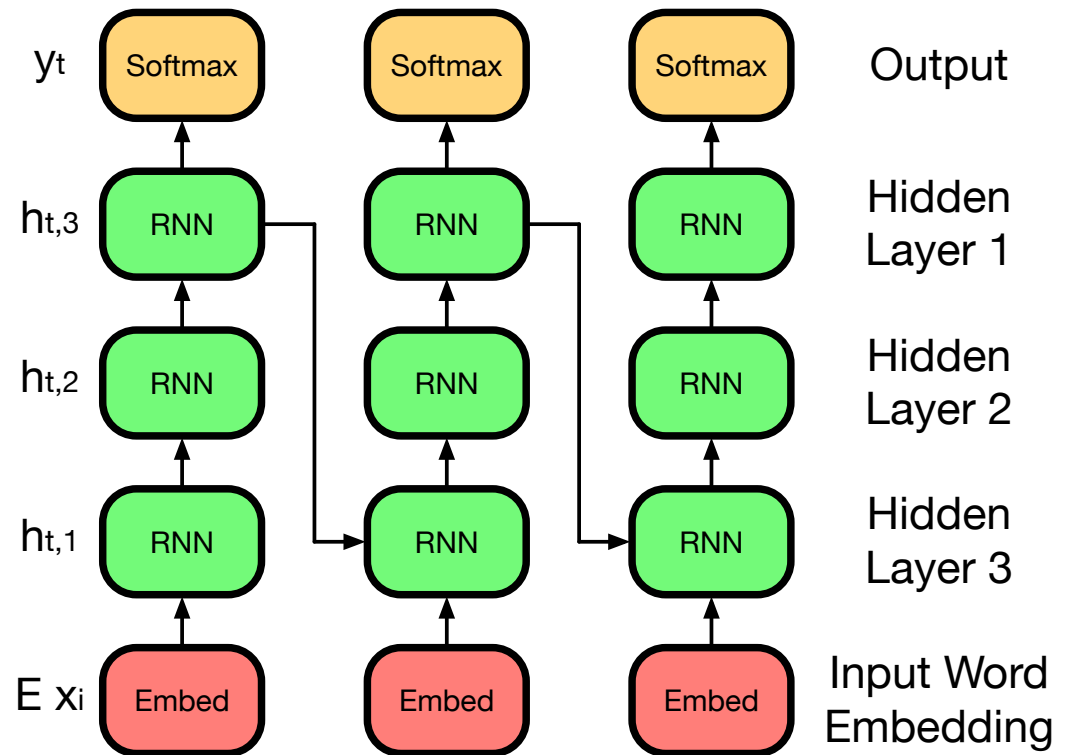


- Not much **deep** learning so far
- Between prediction from input to output: only 1 hidden layer
- How about more hidden layers?

# Deep Models



**Deep Stacked**



**Deep Transitional**

questions?