UNIVERSITY OF MANNHEIM

Web Mining

Web Content Mining: Named Entity Recognition

Simone Paolo Ponzetto

FSS 2024

Information Extraction

- Information extraction (IE) is the automatic identification of selected types of entities, relations, or events in free text

- Traditionally, IE tasks tasks are the following:
  - Named entity recognition and classification (NERC)
  - Coreference resolution
  - Relation extraction
  - Event extraction

- The following tasks loosely belong to IE:
  - Keywords/keyphrase extraction
  - Terminology extraction
  - Collocation extraction

Outline

Supervised Named Entity Recognition

- In information extraction, a named entity is a real-world object, such as a person, location, organization, product, etc., that can be denoted with a proper name. It can be abstract or have a physical existence.
- Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

Named Entity Recognition

Eastern Ukraine is gripped by an armed separatist uprising, with pro-Russian protesters occupying government buildings in more than a dozen towns and cities, despite an ongoing ”anti-terror” operation launched by the Ukrainian military. Vyacheslav Ponomaryov is the self-proclaimed pro-Russian mayor of Sloviansk, Donetsk region, the stronghold of the separatist movement in eastern Ukraine. He was involved in the seizure of a group of military observers from the Organization for Security and Co-operation in Europe (OSCE). One of the best-known leaders of the uprising, Igor Strelkov directs armed pro-Russian activists in eastern Ukraine, especially in Sloviansk. The word is he works for the GRU (Russian military intelligence agency), and his real name is Igor Girkin. He was born in 1970 and registered in Moscow.

- **PERson, LOCation, ORGanization, TIME**
- **Q: What type of NLP task would NER be (from the machine learning perspective)?**

Rule-Based Named Entity Recognition

- Large number of extraction patterns / rules
- Each pattern detects some type of named entities

```
[capitalized-word]+[' Corp.' ] ⇒ Organization
[' Mr.' ][capitalized-word]+ ⇒ Person
[in|at|on][capitalized-word]+ ⇒ Location
```

- Unfortunately, most rules have exceptions...

"She lost hope she would ever meet *Mr. Right One*." (Person?)
"God only knows what goes on in *Putin*'s mind." (Location?)

Building a Named-Entity Tagger

- We can add additional rules to handle exceptions

- E.g., gazetteers: word lists for each of the NER categories
- Some potential gazetteer rules:

```
[cap-word-names-gazetteer]+[cap-word-surnames-gazetteer]+
```

Personal names: Aaliyah, Aaron, Abbey, ..., Zygmunt, Zyta
Surnames: Abbott, Abney, Abraham, ..., Zysett, Zyskowsky Organizations: Abbott Laboratories, Abercrombie & Fitch, Association for Computational Linguistics, . . . , WorldCom, World Help Foundation
Locations: Alabama, Arkansas, ..., Zimbabwe

- Problem: Gazetteers are always incomplete
- Generally, too many rules, difficult to maintain, etc.

Supervised Named Entity Recognition

- We need: a corpus manually annotated with named entities
- Annotations done according to annotation standard
  - The most renowned annotation standard: MUC-7 (Chinchor & Robinson, 1997)
- MUC-7 named entity types
  - Entity names (ENAMEX) – Person, Organization, Location
  - Temporal expressions (TIMEX) – Date, Time
  - Quantities (NUMEX) – Monetary value, Percentage
- Annotation of named entities is not particularly demanding
  - No need to hire experts (e.g., linguists)
  - Virtually any native speaker can annotate (after training)

Supervised Named Entity Recognition

– **NER is a prototypical sequence labelling task**
  - **But named entities are generally multi-token expressions**
  - **Q: What labels do we assign to individual tokens?**
– **We need to make a distinction between the first token of a named entity and all other tokens**
  - **Q: Why?**

Barcelona's/ORG draw/O with/O Atletico/ORG Madrid/ORG at/O Camp/LOC Nou/LOC was/O not/O expected/O, says/O British/ORG Broadcast/ORG Channel's/ORG La/ORG Liga/ORG football expert Andy/PER West/PER.

- **„*British Broadcast Channel's La Liga*" – one or two organizations?**

Supervised Named Entity Recognition

- NER is a prototypical **sequence labelling** task
  - But named entities are generally multi-token expressions
- B-I-O annotation scheme
  - B – Begins a named entity (i.e., first NE token)
  - I – Inside a named entity (i.e., second and subsequent NE tokens)
  - O – Outside of a named entity (i.e., token is not part of any NE)

Barcelona’s/B-ORG draw/O with/O Atletico/B-ORG Madrid/I-ORG at/O Camp/B-LOC Nou/I-LOC was/O not/O expected/O, says/O British/B-ORG Broadcast/I-ORG Channel’s/I-ORG La/B-ORG Liga/I-ORG football expert Andy/B-PER West/I-PER.

- „*British Broadcast Channel’s La Liga*” – **two organizations!**

Supervised Named Entity Recognition

Supervised approaches to NER:

1. **Token-level classification**
   - Naive Bayes, SVM, Logistic regression, Feed-forward NN
   - Cannot use labels from both token sides as features
2. **Sequence labelling**
   - Hidden Markov Models (HMM), Conditional Random Fields (CRF)
     - Require manual feature design
   - Recurrent (or gated convolutional) neural networks
     - Word embeddings as input, no feature design
     - State-of-the-art results

Common features (for feature-based learning algorithms):

- Linguistic features: word, lemma, POS-tag, sentence start, capitalization, ...
- Gazetteer features: is gazetteer entry, starts gazetteer entry, inside of a gazetteer entry (for all gazetteers)

Named Entity Recognition – Document Level

- **Sequence models predict BIO labels at the sentence level**
- **Thus, it's possible to have different labels for the same named entity at the document level**

Eastern Ukraine is gripped by an armed separatist uprising. Vyacheslav Ponomaryov is the self-proclaimed pro-Russian mayor of Sloviansk, Donetsk region, the stronghold of the separatist movement in eastern Ukraine. He was involved in the seizure of a group of military observers from the Organization for Security and Co-operation in Europe (OSCE). One of the best-known leaders of the uprising, Igor Strelkov directs armed pro-Russian activists in eastern Ukraine, especially in Sloviansk.

- **Enforcing document-level consistency improves NER performance**

Outline

1. **Named Entity Recognition**
2. **Evaluation**
3. **RNNs**

Named Entity Recognition Evaluation

– Comparing system predicted Named Entities (NEs) with gold-annotated Nes
  - In terms of precision, recall, and F-score

1. Lenient (aka MUC) evaluation
   - System NE and gold NE need to be of the same type and overlap in token spans in order to count as a match (i.e., true positive)
2. Strict (aka Exact) evaluation
   - System NE and gold NE need to be of the same type and exactly the same token span order to count as a match (i.e., true positive)

Gold: „The Faculty of Business Informatics and Mathematics issued a diploma…”
Sys1: „The Faculty of Business Informatics and Mathematics issued a diploma…”
Sys2: „The Faculty of Business Informatics and Mathematics issued a diploma…”

– State-of-the-art NER performance (coarse-grained entity types) is around 94% F-score for English, and significantly less for other languages

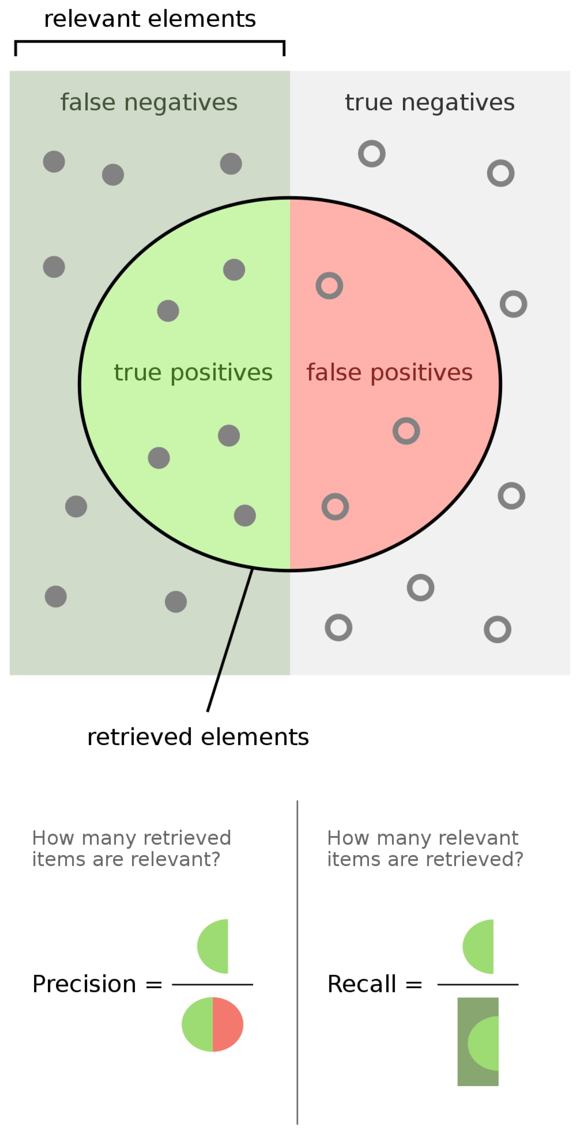
Named Entity Recognition Evaluation

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}}$$

The F1 score is the <span>harmonic mean</span> of the precision and recall.

$$F_1 = 2\,\frac{recall \cdot precision}{recall + precision}$$

$$F_1 = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$



Wikipedia, released under Creative Commons Attribution-ShareAlike License 3.0 (https://creativecommons.org/licenses/by-sa/3.0/)

I. Surface string and entity type match

Gold Standard		System Prediction	
Token	Entity Type	Token	Entity Type
in	O	in	O
New	B-LOC	New	B-LOC
York	I-LOC	York	I-LOC
.	O	.	O

II. System hypothesized an entity

Gold Standard		System Prediction	
Token	Entity Type	Token	Entity Type
an	O	an	O
Awful	O	Awful	B-ORG
Headache	O	Headache	I-ORG
in	O	in	O

III. System misses an entity

Gold Standard		System Prediction	
Token	Entity Type	Token	Entity Type
in	O	in	O
Palo	B-LOC	Palo	O
Alto	I-LOC	Alto	O
,	O	,	O

Note

- Note that considering only this 3 scenarios, and discarding every other possible scenario we have a simple classification evaluation that can be measured in terms of false negatives, true positives and false positives, and subsequently compute precision, recall and f1-score for each named-entity type.
- But of course we are discarding partial matches, or other scenarios when the NER system gets the named-entity surface string correct but the type wrong, and we might also want to evaluate these scenarios again at a full-entity level.

IV. System assigns the wrong entity type

Gold Standard		System Prediction	
Token	Entity Type	Token	Entity Type
I	O	I	O
live	O	live	O
in	O	in	O
Palo	B-LOC	Palo	B-ORG
Alto	I-LOC	Alto	I-ORG
,	O	,	O

V. System gets the boundaries of the surface string wrong

Gold Standard		System Prediction	
Token	Entity Type	Token	Entity Type
Unless	O	Unless	B-PER
Karl	B-PER	Karl	I-PER
Smith	I-PER	Smith	I-PER
resigns	O	resigns	O

VI. System gets the boundaries and entity type wrong

Gold Standard		System Prediction	
Token	Entity Type	Token	Entity Type
Unless	O	Unless	B-ORG
Karl	B-PER	Karl	I-ORG
Smith	I-PER	Smith	I-ORG
resigns	O	resigns	O

CoNLL: NER task

- The **Language-Independent Named Entity Recognition task** introduced at CoNLL-2003 measures the performance of the systems in terms of precision, recall and f1-score, where:
- *"precision is the percentage of named entities found by the learning system that are correct. Recall is the percentage of named entities present in the corpus that are found by the system.* ***A named entity is correct only if it is an exact match of the corresponding entity in the data file."***
- so basically it only considers scenarios I, II and III, the others described scenarios are not considered for evaluation.

Outline

1. **Named Entity Recognition**
2. **Evaluation**
3. **RNNs**

Recurrent Neural Networks (RNNs)

- Martin & Jurafsky (2023): A network that contains a cycle within its network connections, meaning that *the value of some unit is directly, or indirectly, dependent on its own earlier outputs as an input*

Recurrent neural networks

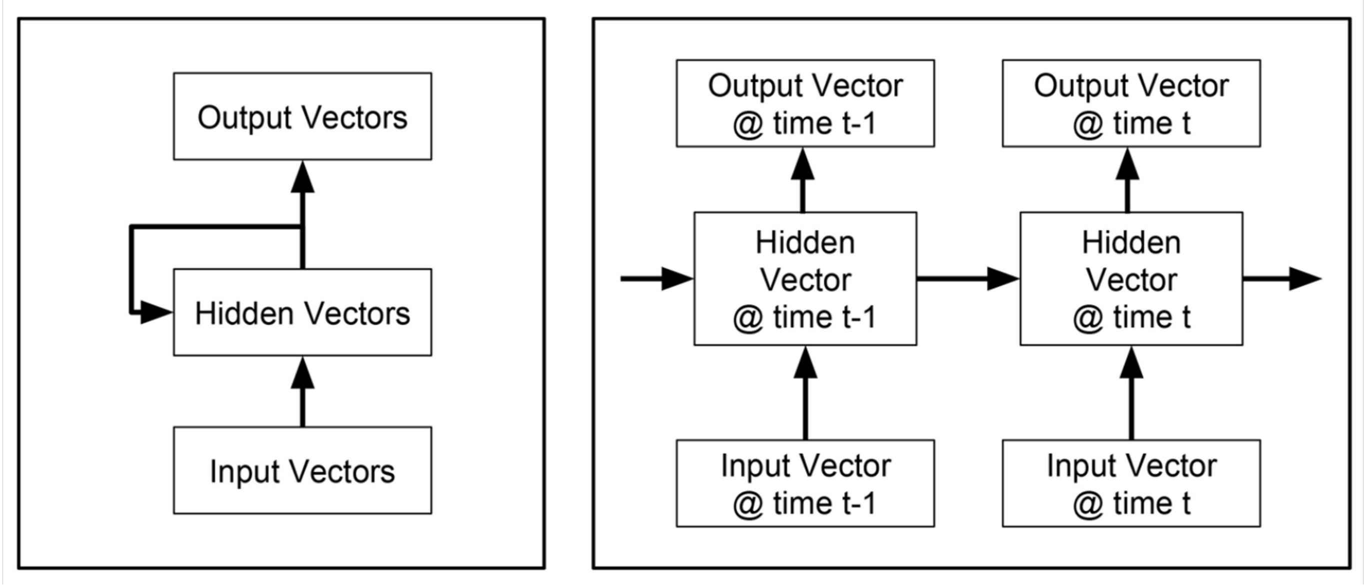
- Recurrent neural networks are neural models that explicitly take into account the sequences
  - Sequences of words in a sentence, sentences in a paragraph, etc.

General RNN model:

- Input: sequence of input vectors (e.g., word embeddings): $x_1, \ldots, x_n$
- RNN is a function that converts an arbitrary size sequence $x_1, \ldots, x_n$ into a fixed size output vector y_n
  - Analogously, the subsequence $x_1, \ldots, x_i$ will produce the output y_i
- The output vector y_{i-1} of the previous step ($i-1$) is combined with the current input x_i to produce the output y_i
- The RNN network is, at time step i, represented with its current state s_i

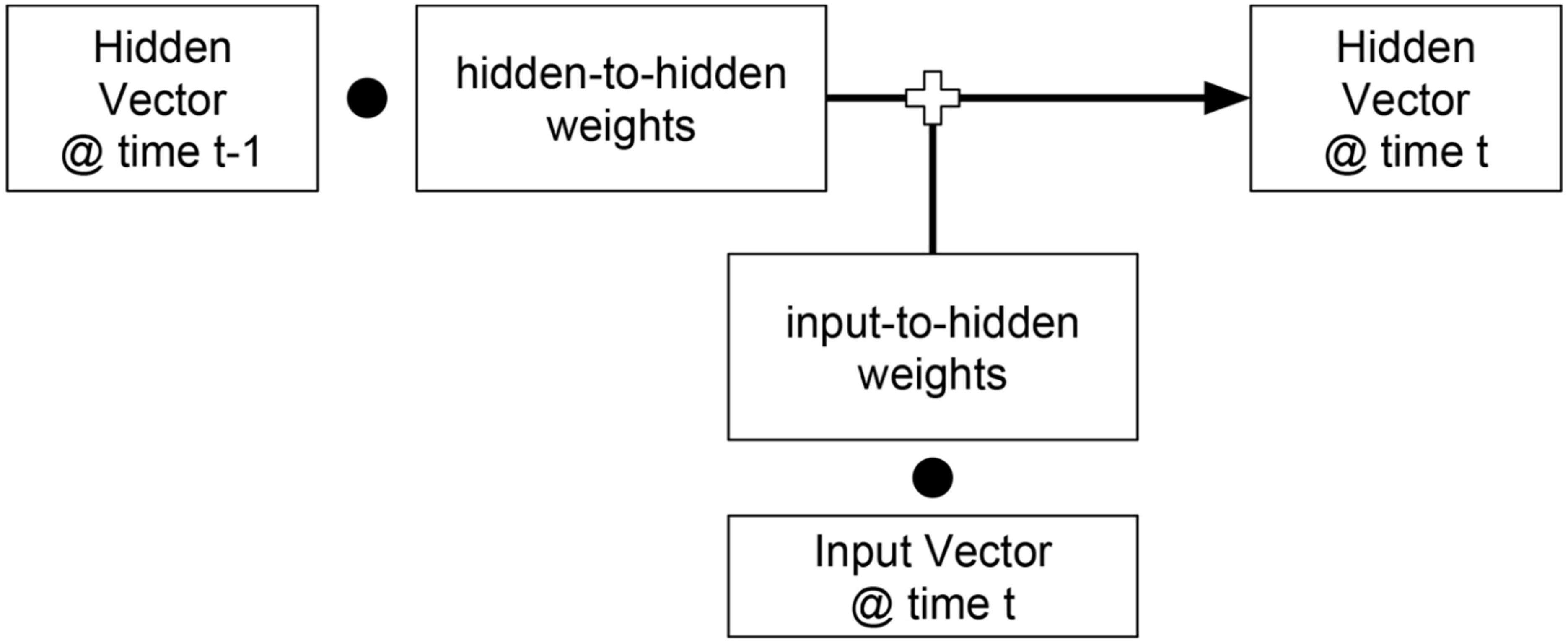
Elman (1990) Recurrent Neural Network (RNN)

- The goal is to learn a representation of a sequence by *maintaining a hidden state vector that act as form of memory (or context) to encode the sequence seen so far*
- The hidden layer includes a recurrent connection as part of its input
- The hidden state vector is computed from both a current input vector and the previous hidden state vector.



Elman (1990) Recurrent Neural Network (RNN)

- **Input vector from the current time step and the hidden state vector from the previous time step are mapped to the hidden state vector of the current time step:**

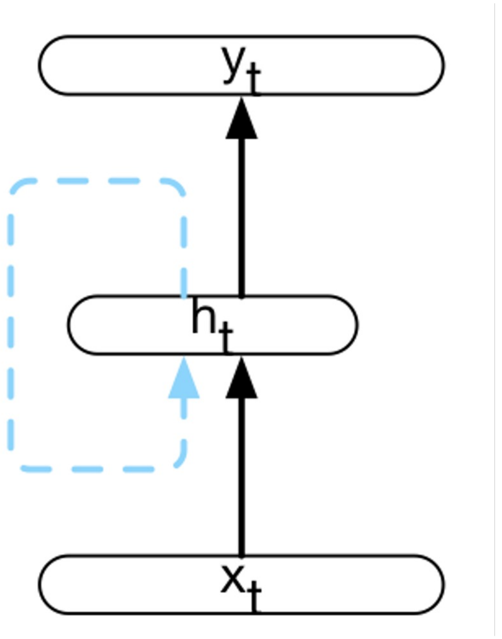


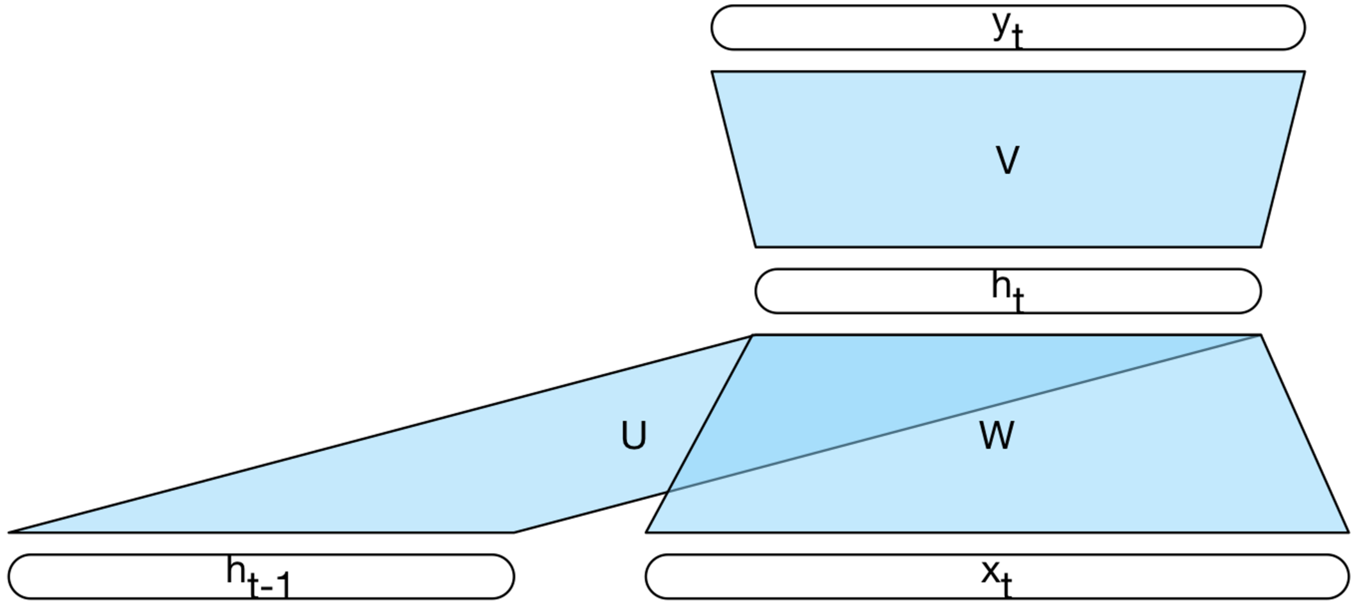
Elman (1990) Recurrent Neural Network (RNN)

- Hidden-to-hidden and input to hidden weights are shared across the different time steps
- Weights are adjusted so that the RNN is learning how to incorporate incoming information and maintain a state representation summarizing the input seen so far
- RNN does not have any way of knowing which time step it is on: RNN is “only” learning how to transition from one time step to another and maintain a state representation that will minimize its loss.

Elman (1990) or “Simple” RNN

- input vector representing the current input at time step *t* x_t
- hidden units h_t
- output y_t

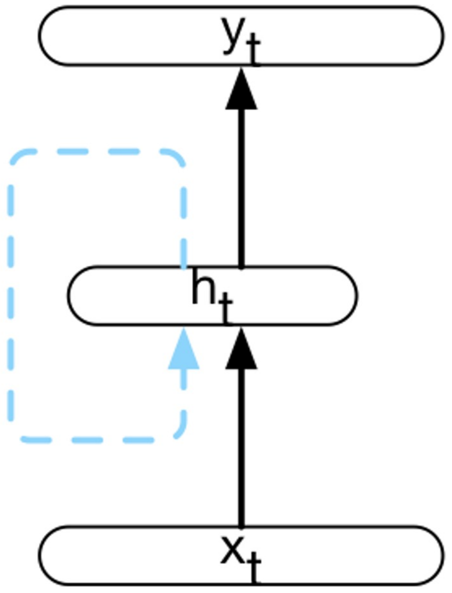


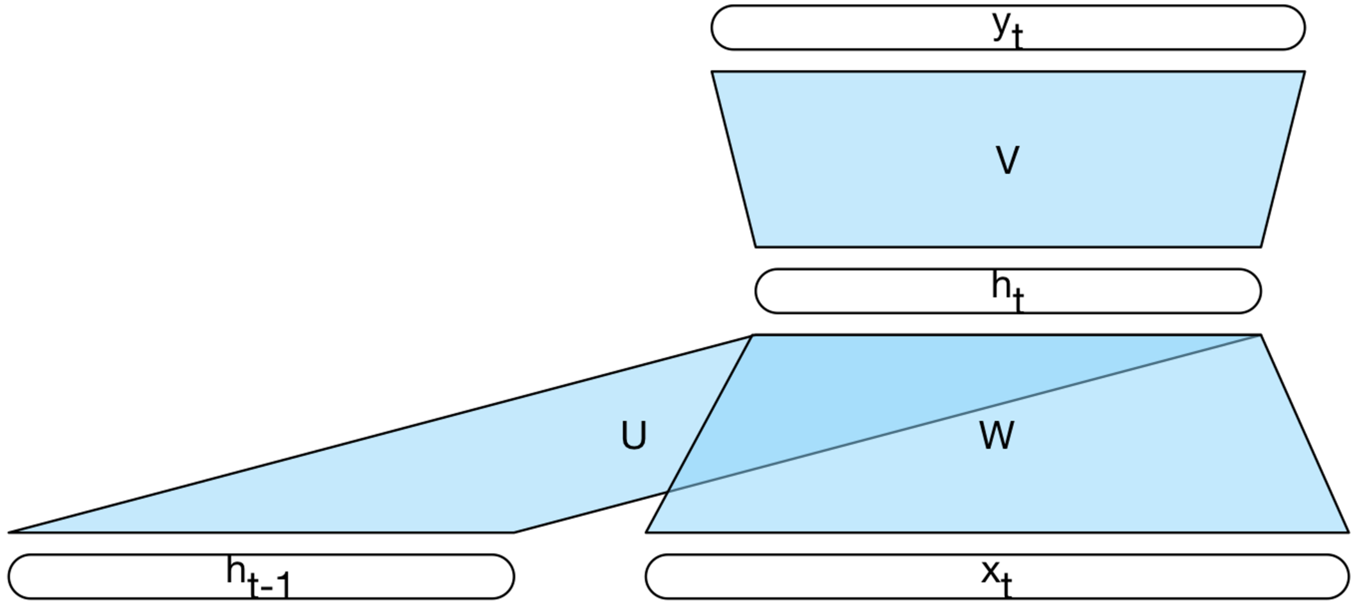


$$
\begin{aligned}
h_t &= g(Uh_{t-1} + Wx_t) \\
y_t &= f(Vh_t) \\
y_t &= softmax(Vh_t)
\end{aligned}
$$

Elman (1990) or “Simple” RNN

- **W: weights from the input layer to the hidden layer**
- **U: weights from the previous hidden layer to the current hidden layer**
- **V: weights from the hidden layer to the output layer**

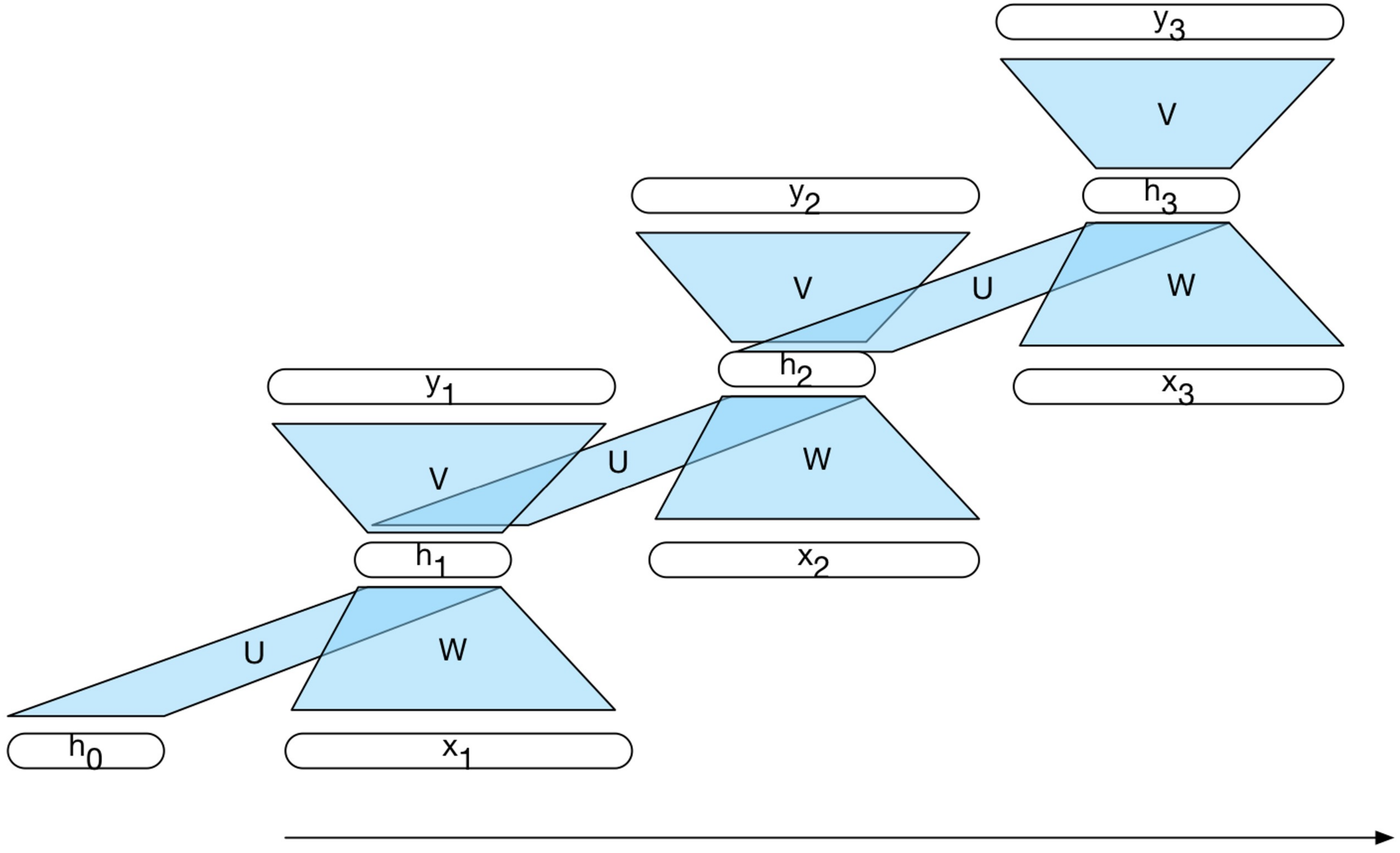




$$\begin{aligned} h_t &= g(Uh_{t-1} + Wx_t) \\ y_t &= f(Vh_t) \\ y_t &= softmax(Vh_t) \end{aligned}$$

Unrolling the simple RNN

- **Network layers are copied for each time step, while the weights U, V and W are shared in common across all time steps.**



Forward inference

- **Forward inference (mapping a sequence of inputs to a sequence of outputs) requires an inference algorithm that proceeds from the start of the sequence to the end**

```
**function** FORWARDRNN($x$, $network$) **returns** output sequence $y$

  $h_0 \leftarrow 0$
  **for** $i \leftarrow 1$ **to** LENGTH($x$) **do**
      $h_i \leftarrow g(U\ h_{i-1} + W\ x_i)$
      $y_i \leftarrow f(V\ h_i)$
  **return** $y$
```

- **The matrices U, V and W are shared across time, while new values for *h* and *y* are calculated with each time step.**

A RNN Language Model

$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

hidden states

$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$

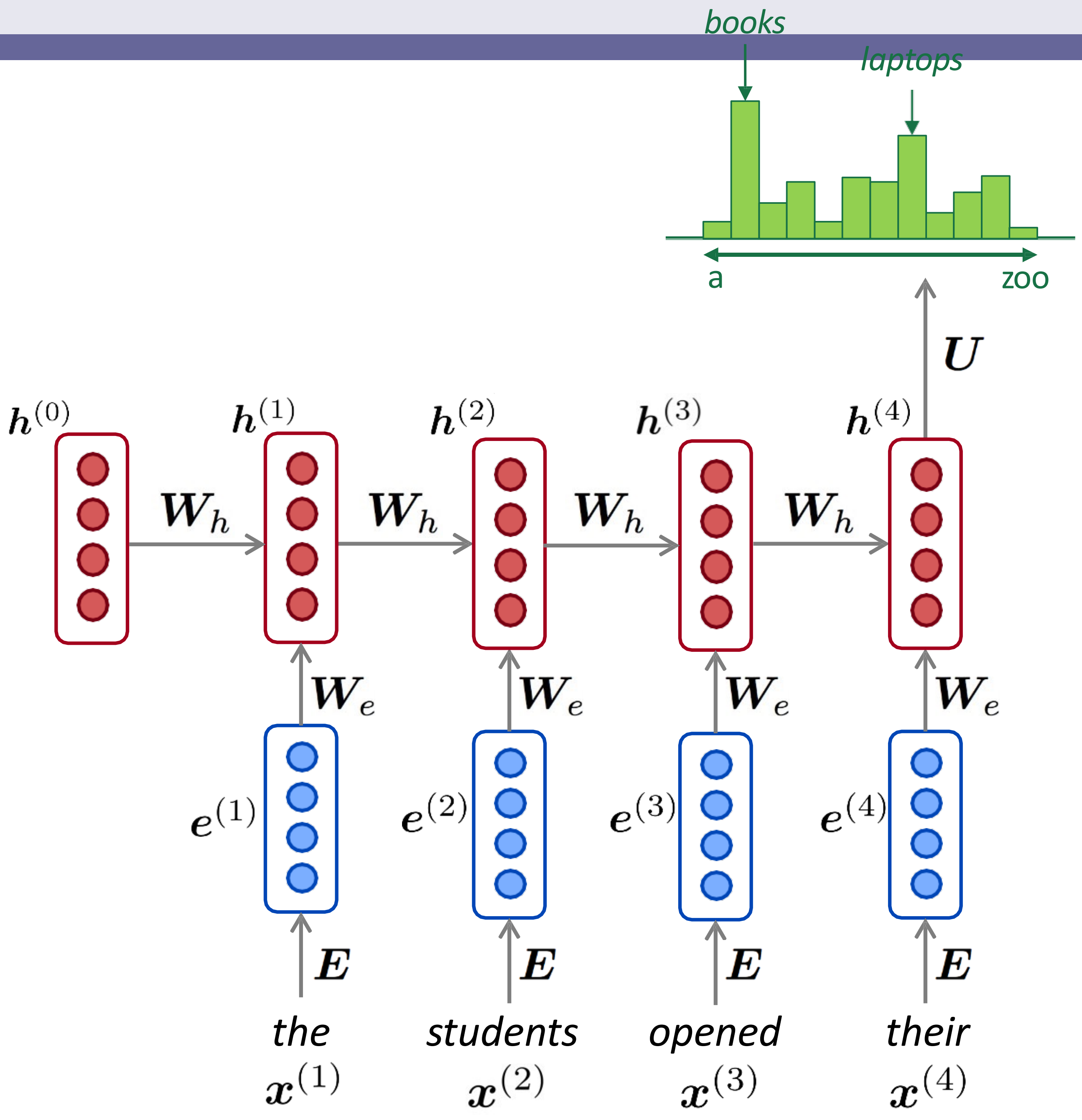
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



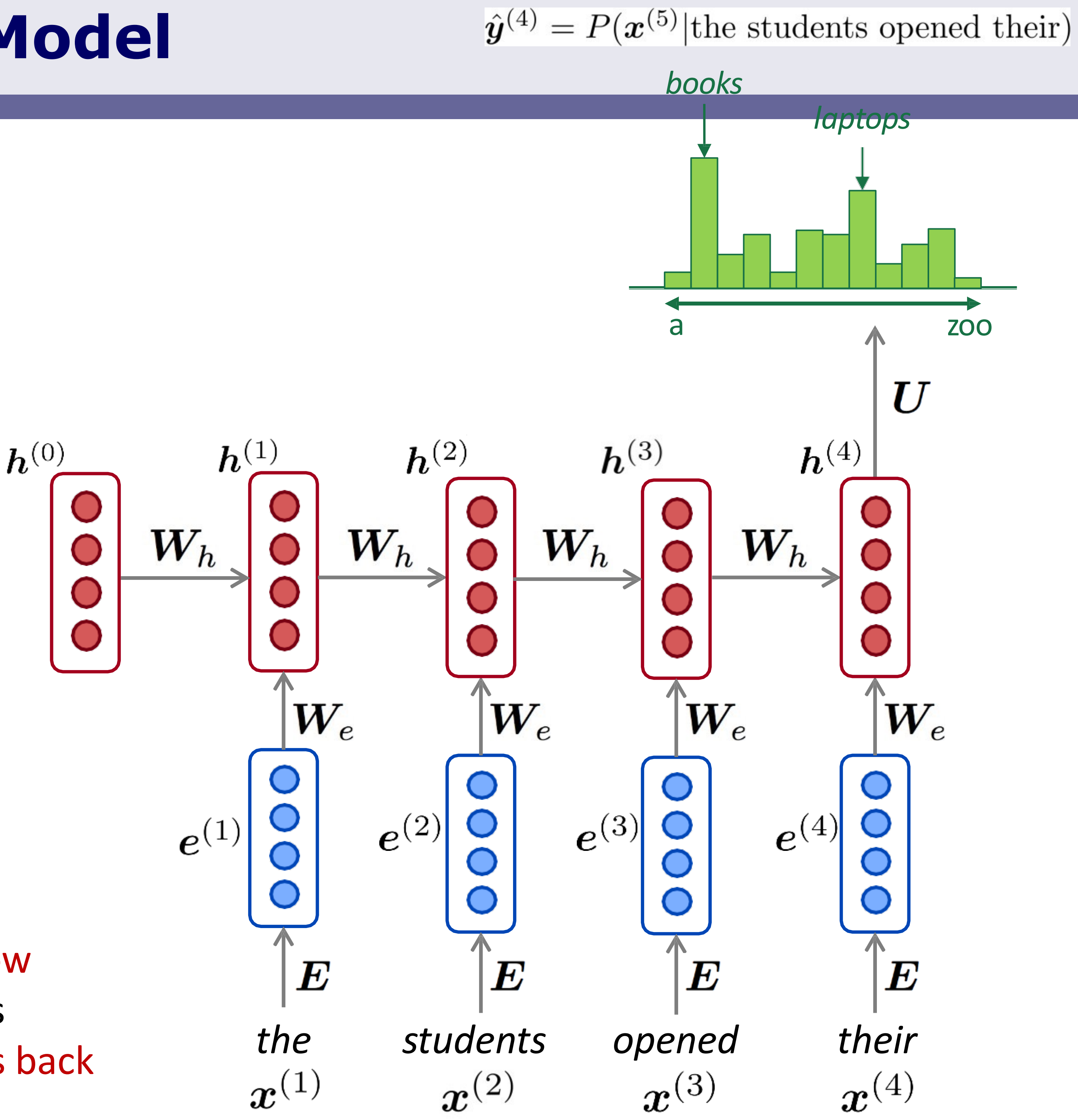
A RNN Language Model

RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back



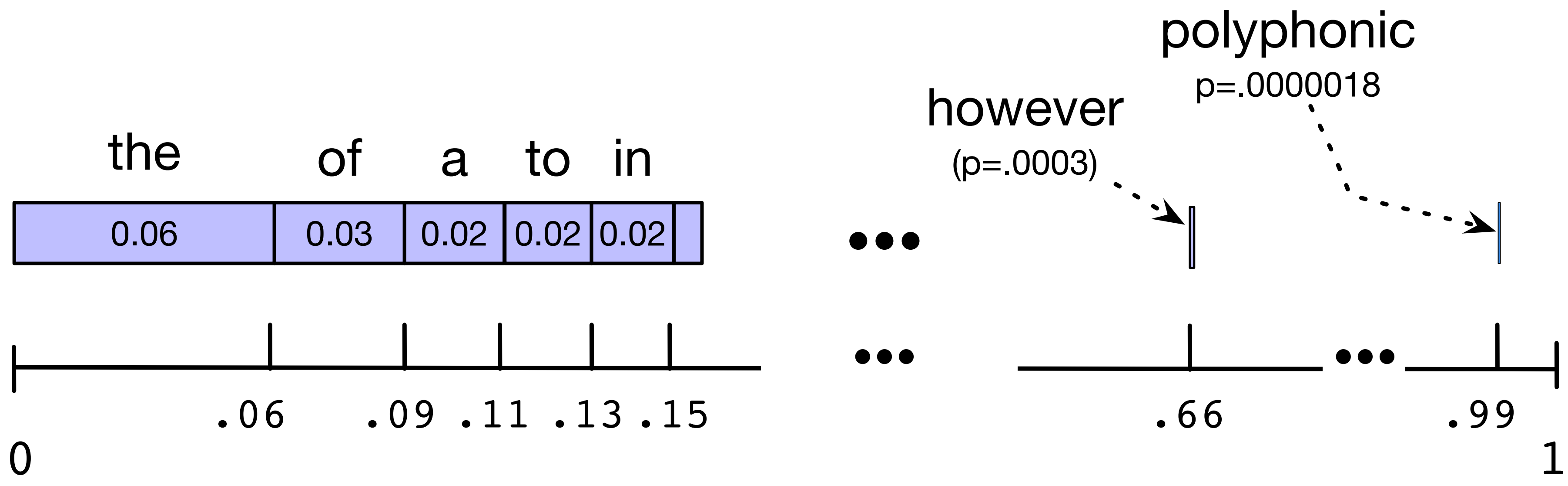
Generating with an RNN LM

Also known as autoregressive generation or causal LM generation

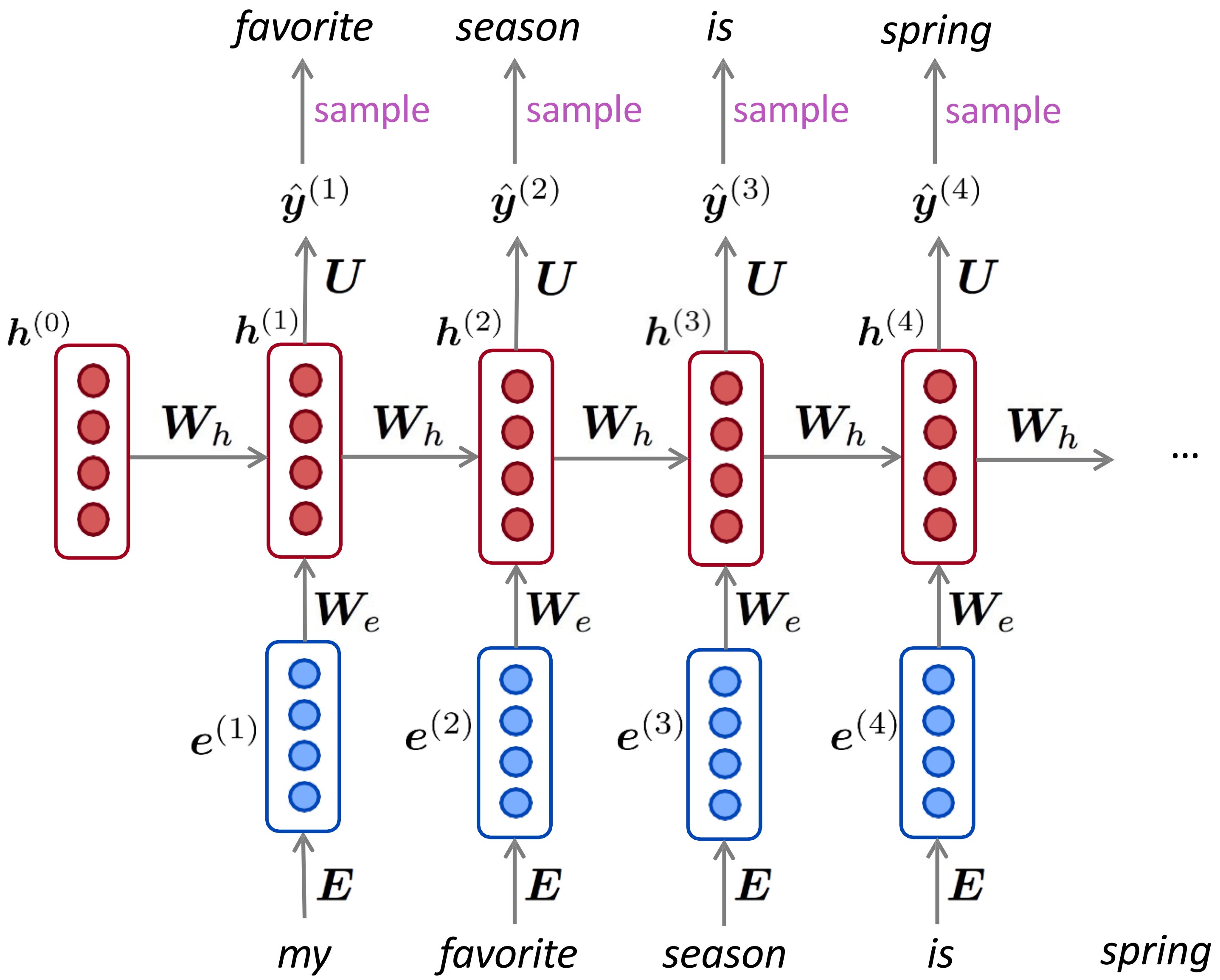
- **Sample a word in the output from the softmax distribution that results from using the beginning of sentence marker, <s>, as the first input**
- **Use the word embedding for that first word as the input to the network at the next time step, and then sample the next word in the same fashion**
- **Continue generating until the end of sentence marker, </s>, is sampled or a fixed length limit is reached**

Using a language model as generator through sampling

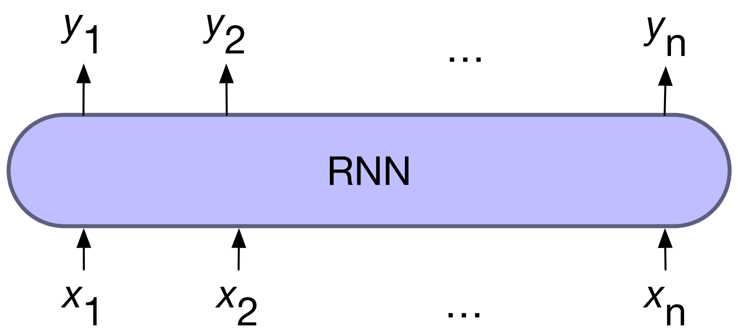
- First proposed by Shannon (1951) and Miller and Selfridge (1950)
- Simple example: unigram case



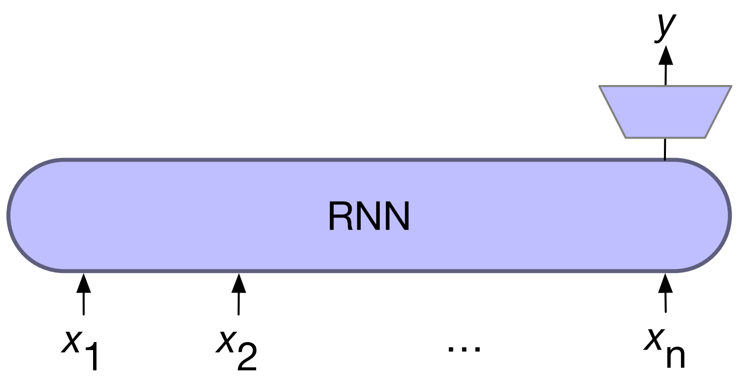
Generating with an RNN LM



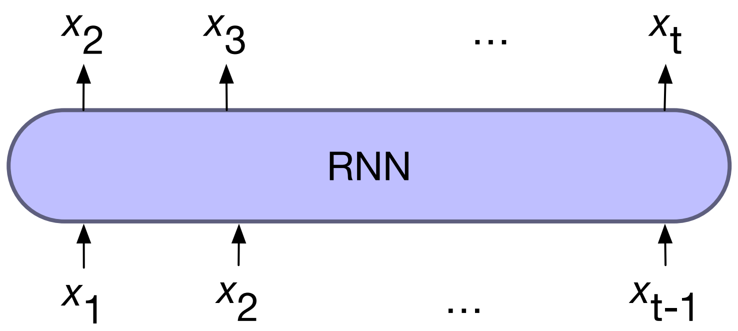
Common RNN architectures used in NLP



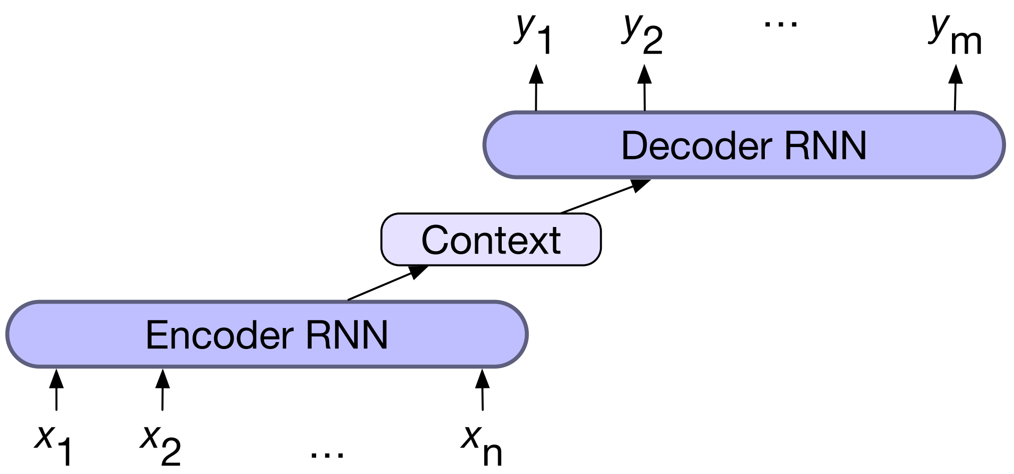
a) sequence labeling



b) sequence classification

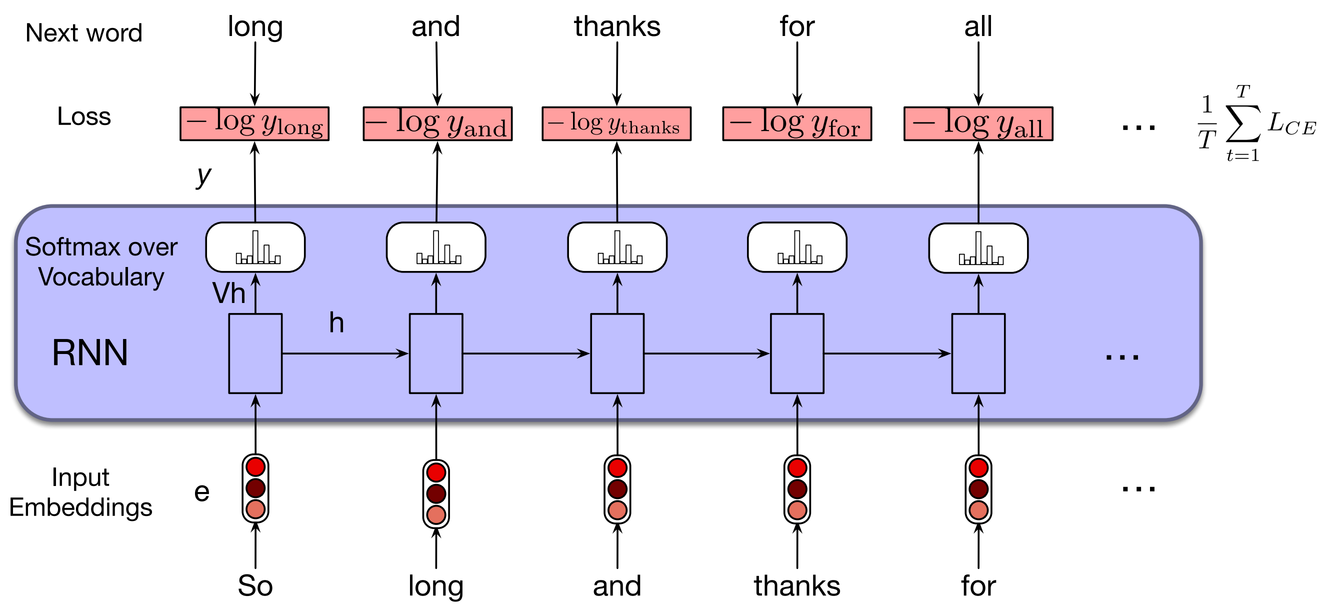


c) language modeling



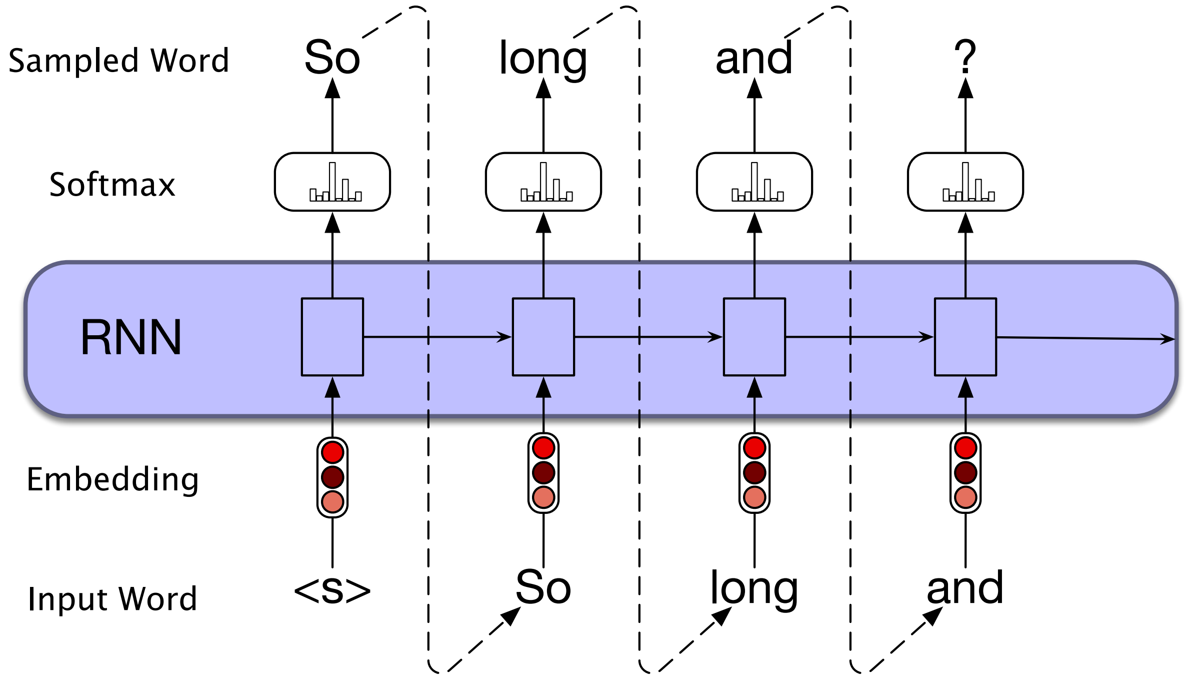
d) encoder-decoder

RNNs as language models

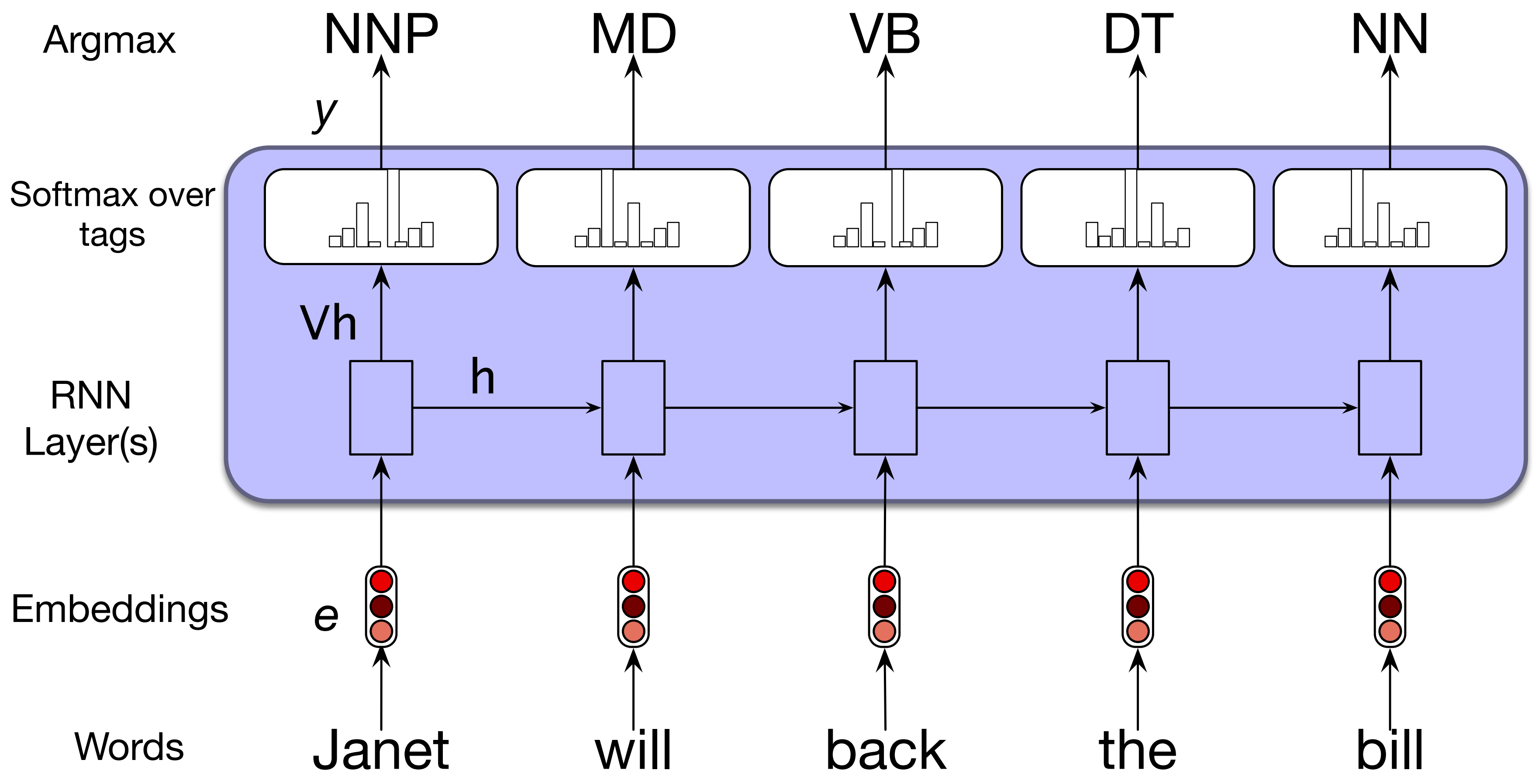


(training)

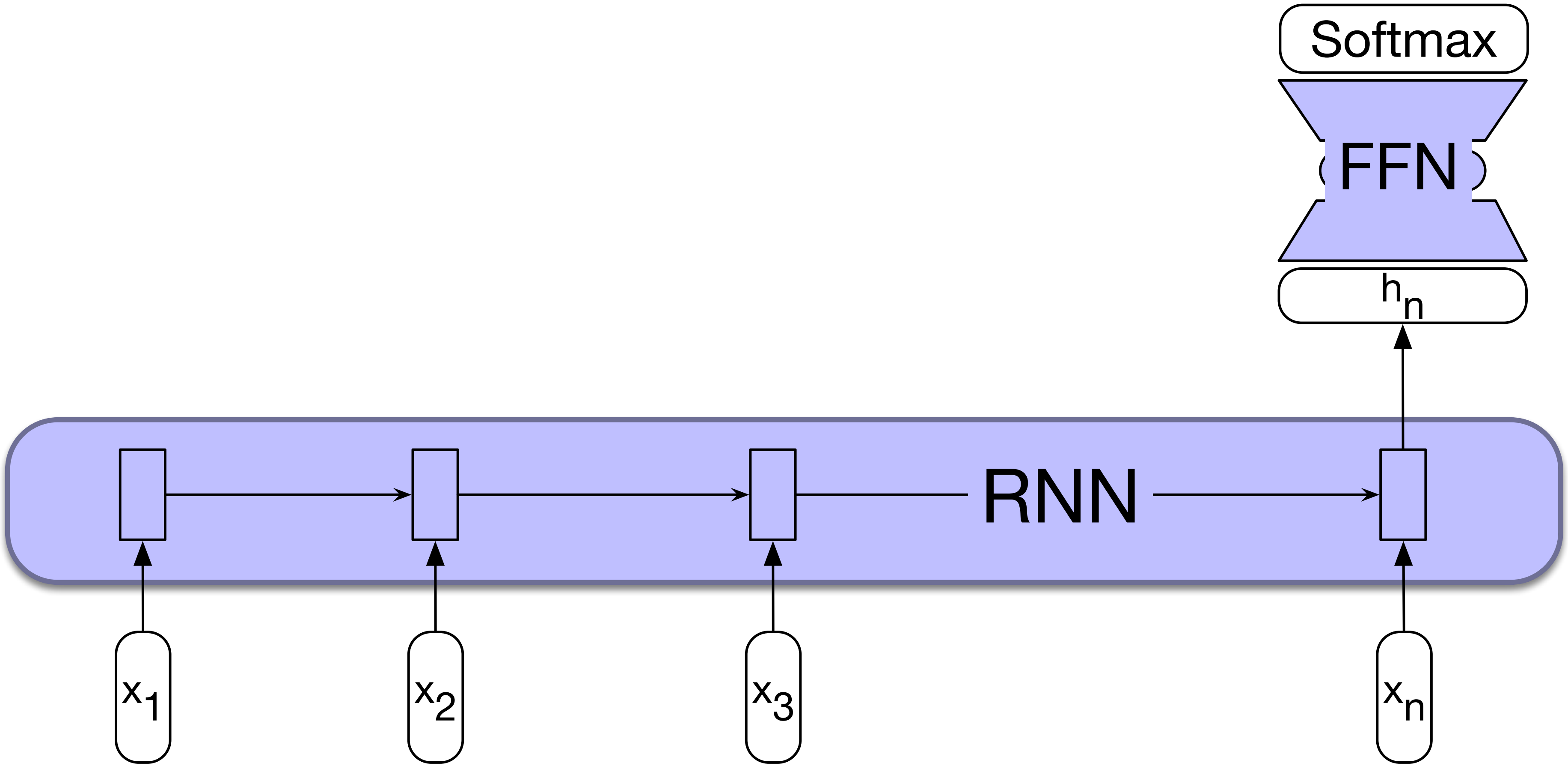
(generation)



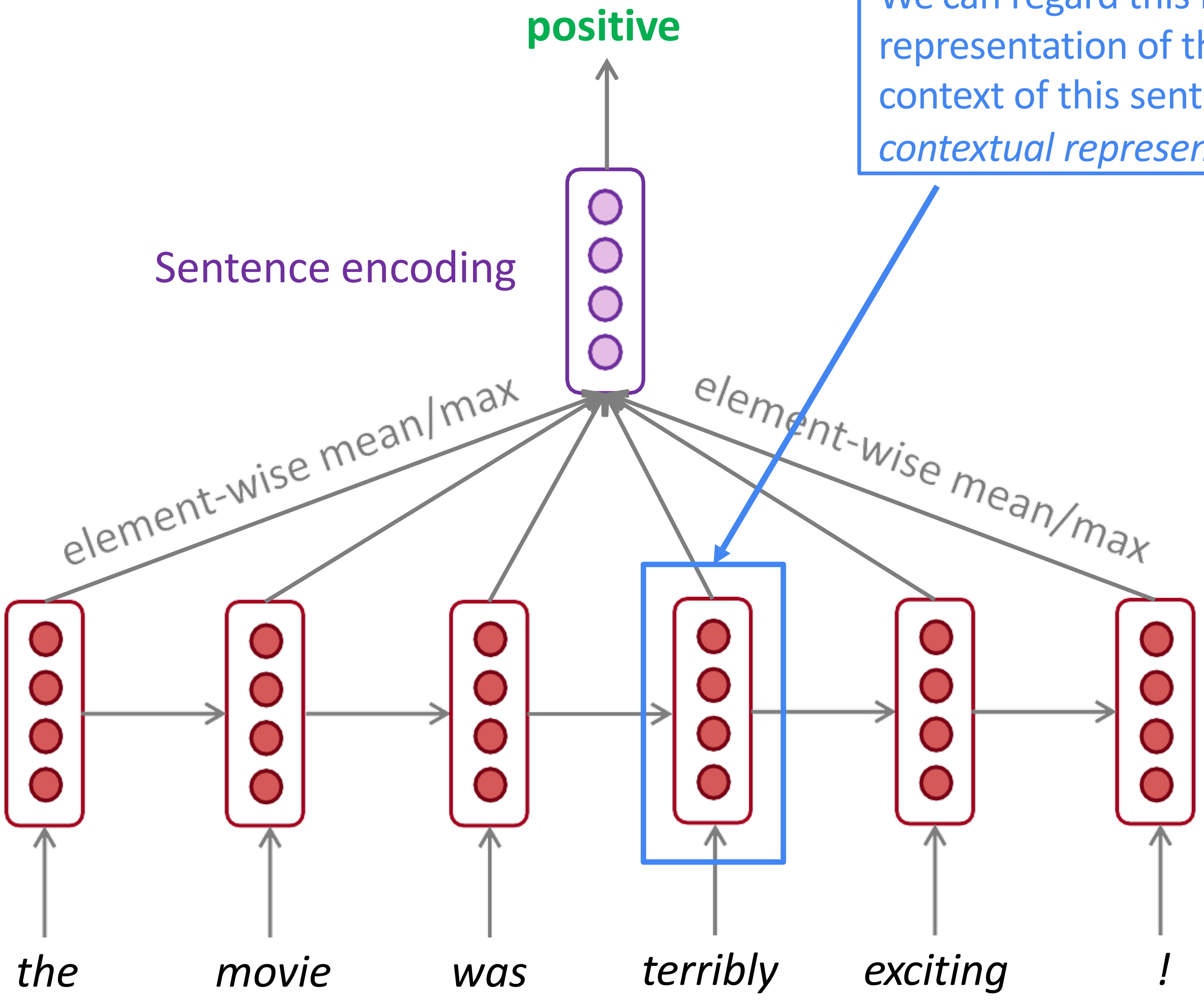
RNNs for sequence labeling



RNNs for sequence classification



Bidirectional RNNs: motivation



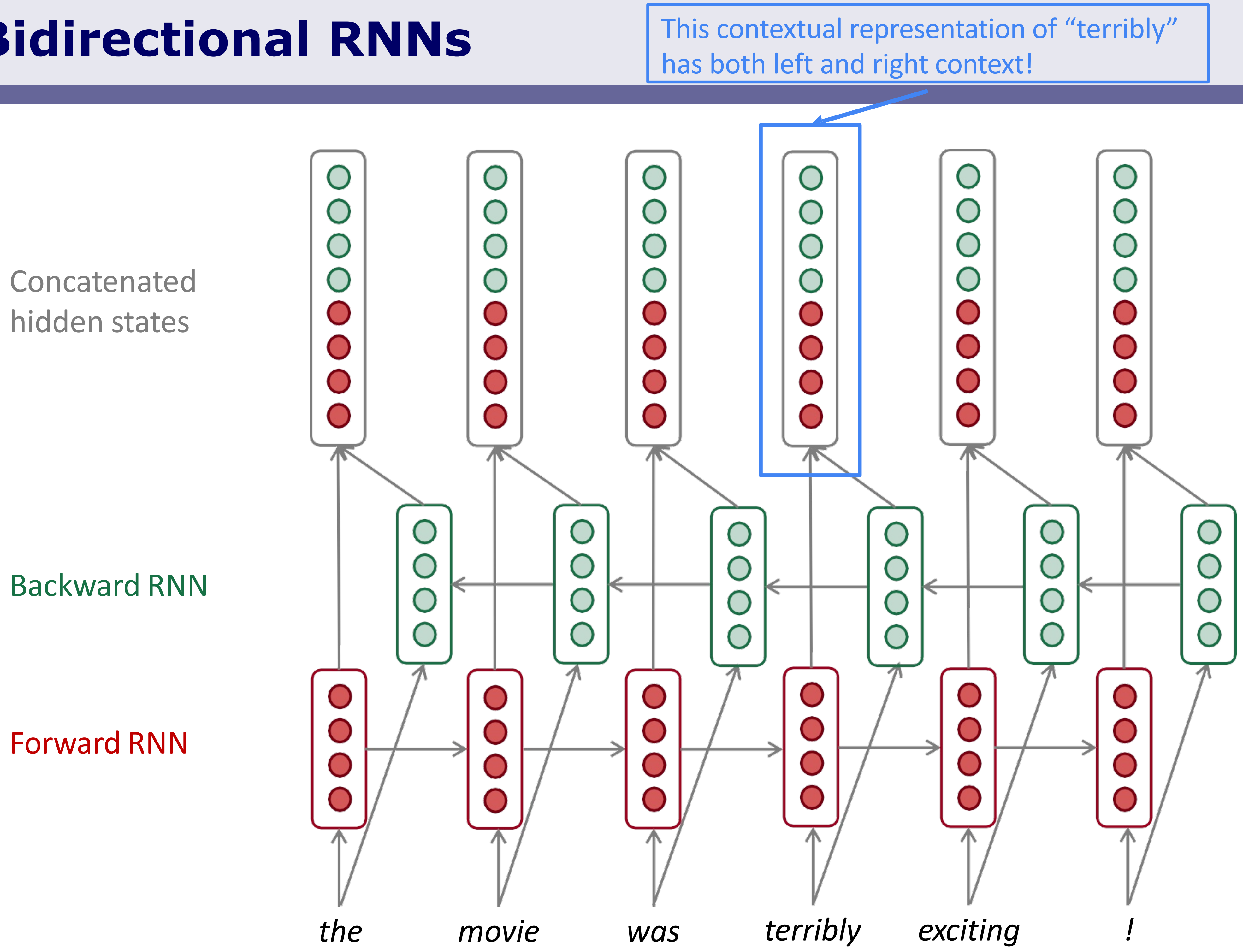
We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation.*

These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)

Bidirectional RNNs


This contextual representation of "terribly" has both left and right context!
Concatenated hidden states
Backward RNN
Forward RNN
the
movie
was
terribly
exciting
!

Bidirectional RNNs

On timestep *t*:

This is a general notation to mean "compute one forward step of the RNN" – it could be a vanilla, LSTM or GRU computation.

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \boxed{\mathrm{RNN}_{\mathrm{FW}}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

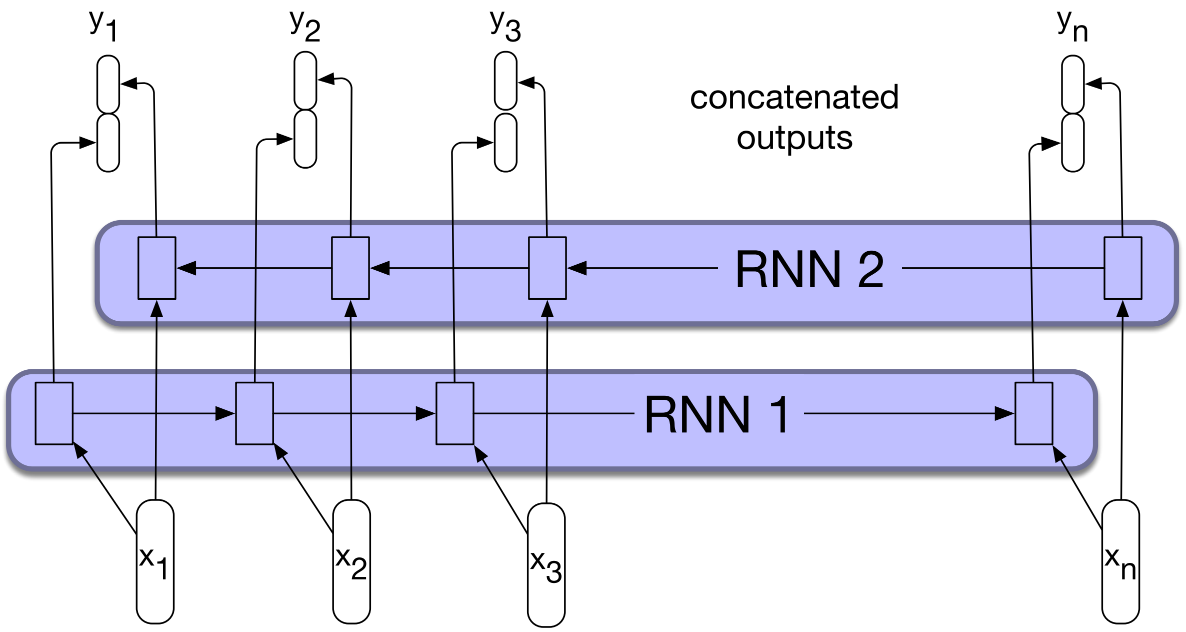
Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

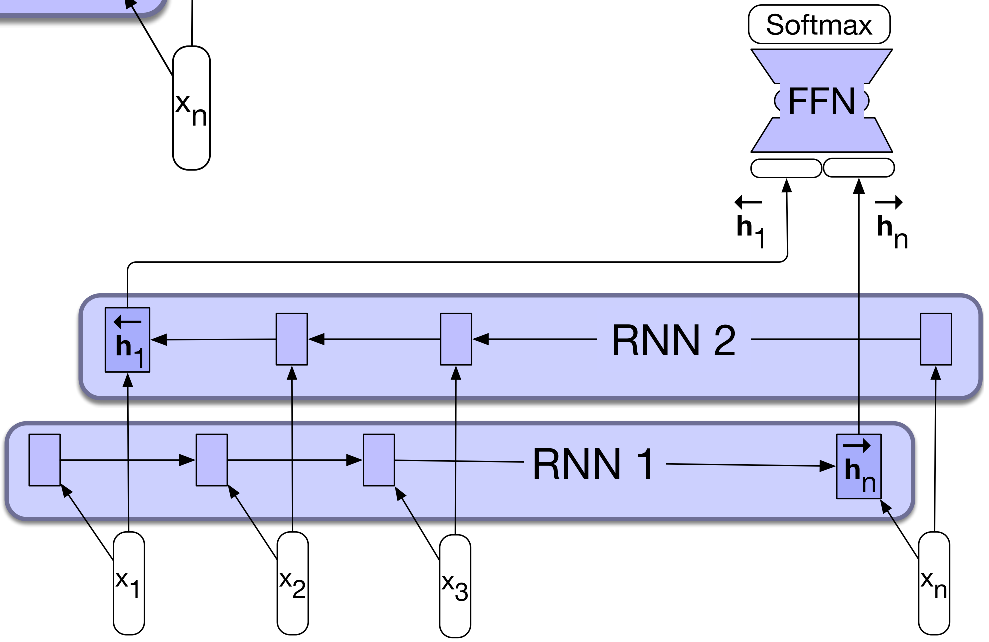
Generally, these two RNNs have separate weights

Concatenated hidden states $\boxed{\boldsymbol{h}^{(t)}} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

We regard this as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network.

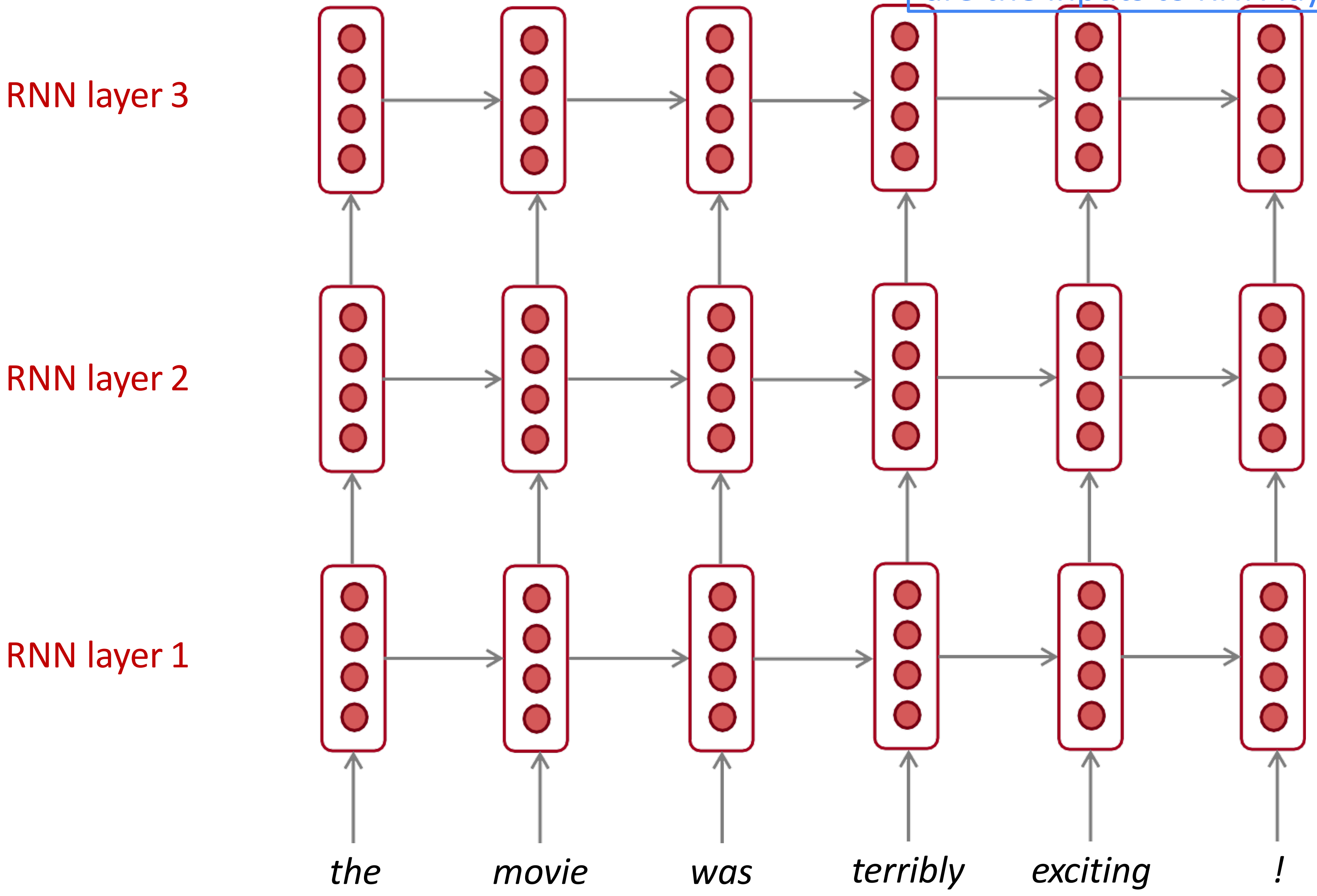
“Vanilla” Bidirectional RNNs vs. for sequence classification





Multi-layer RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i+1*

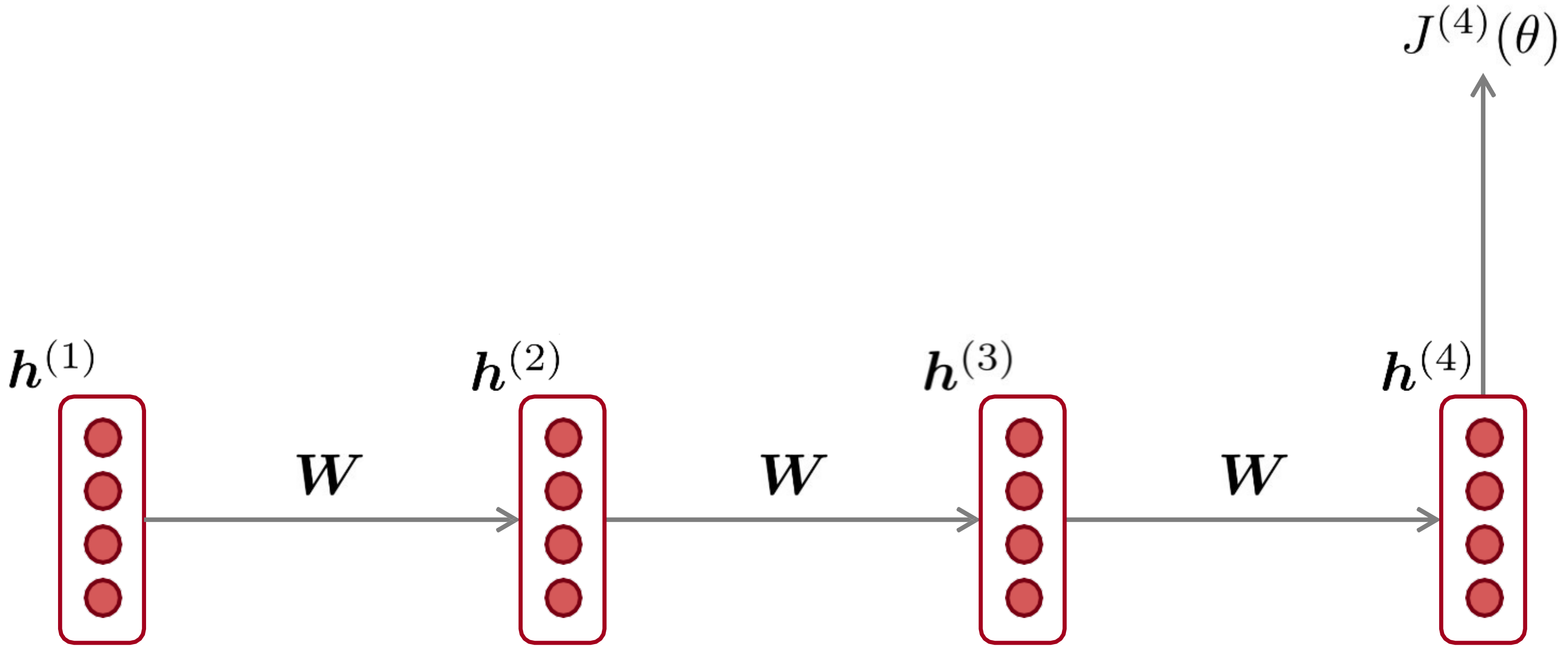


Abigail See

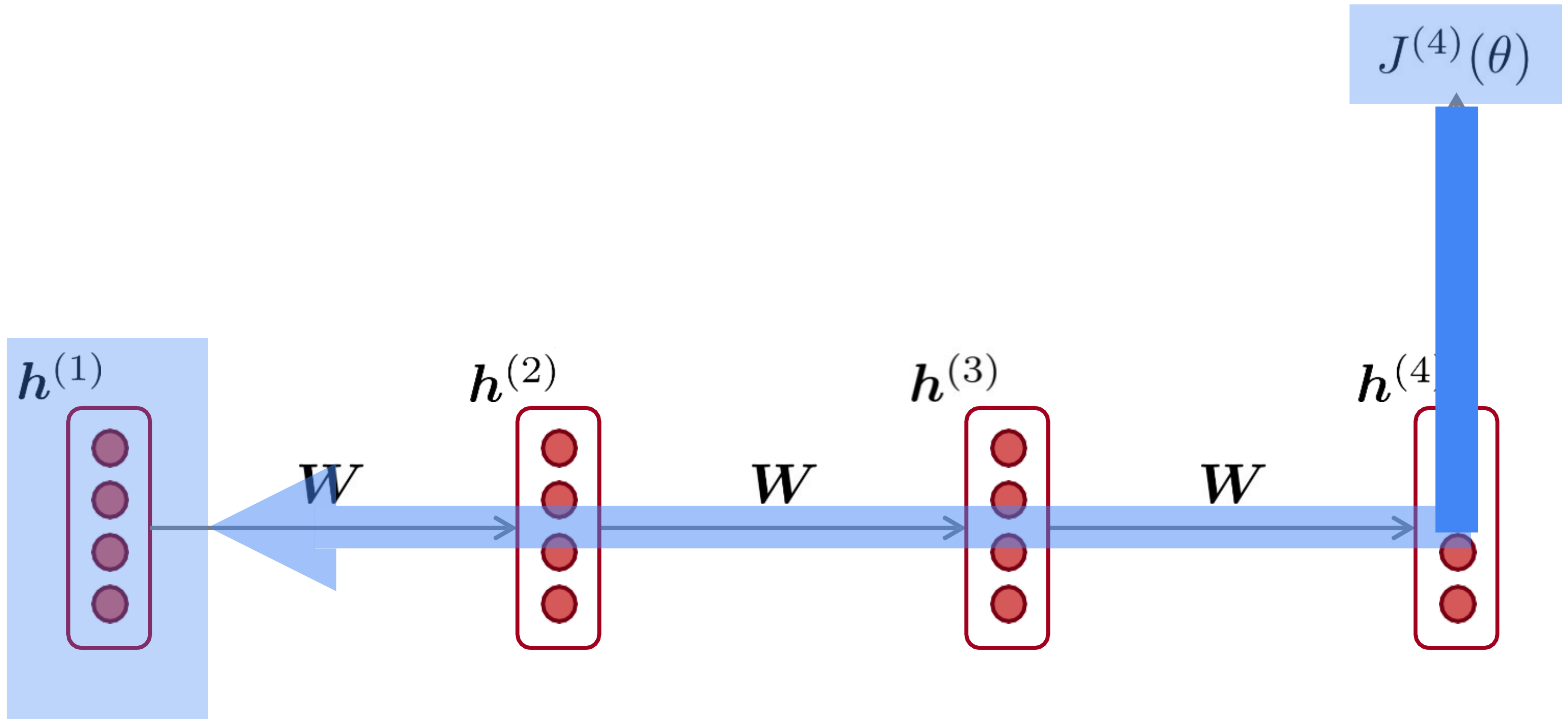
The Problem with Vanilla RNNs (or Elman/Simple RNNs)

- **The inability to retain information for long-range predictions:**
  - **at each time step we simply updated the hidden state vector regardless of whether it made sense;**
  - **RNN has no control over which values are retained and which are discarded in the hidden state;**
    - **that is entirely determined by the input;**
    - **there is no way to decide if the update is optional or not**
- **Vanishing (and) exploding gradients**

Vanishing gradient intuition

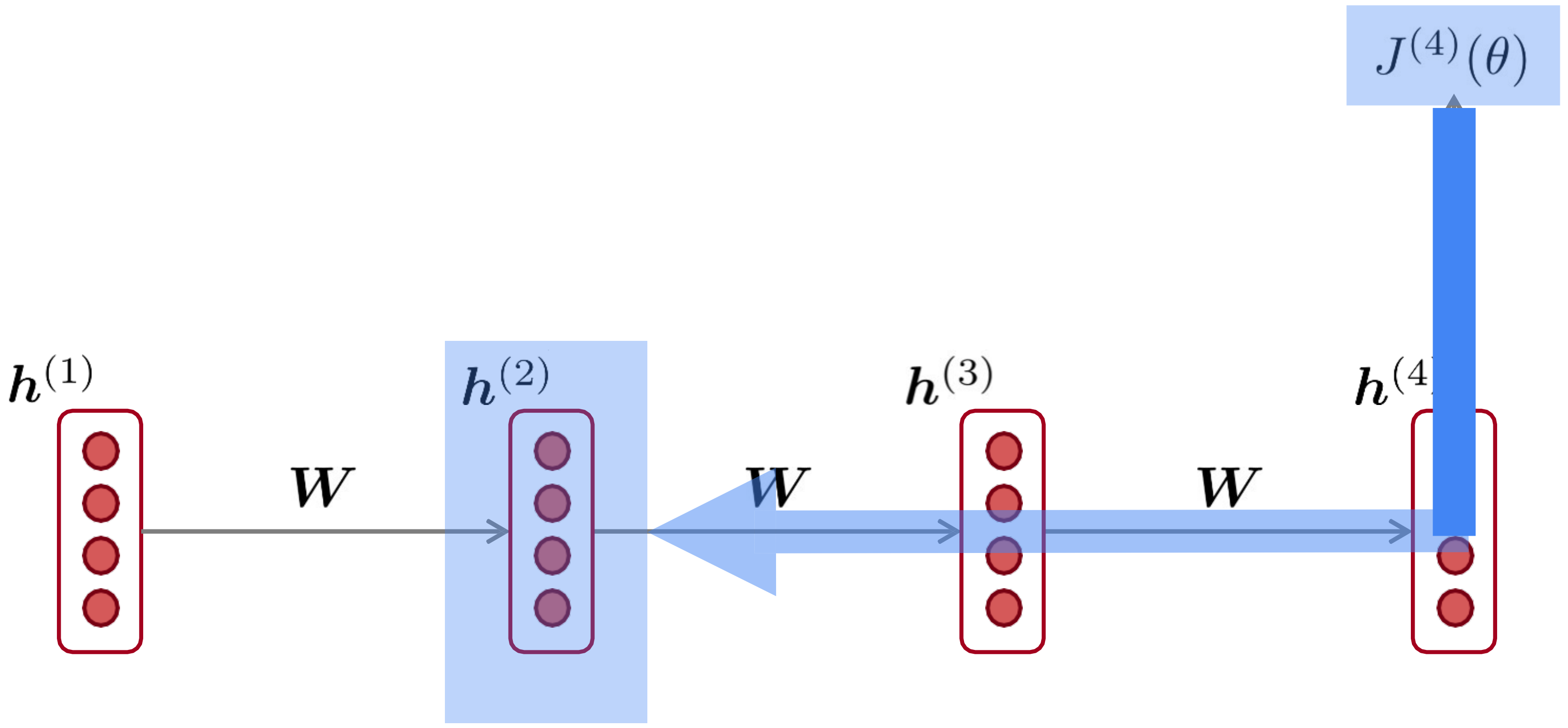


Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = ?$$

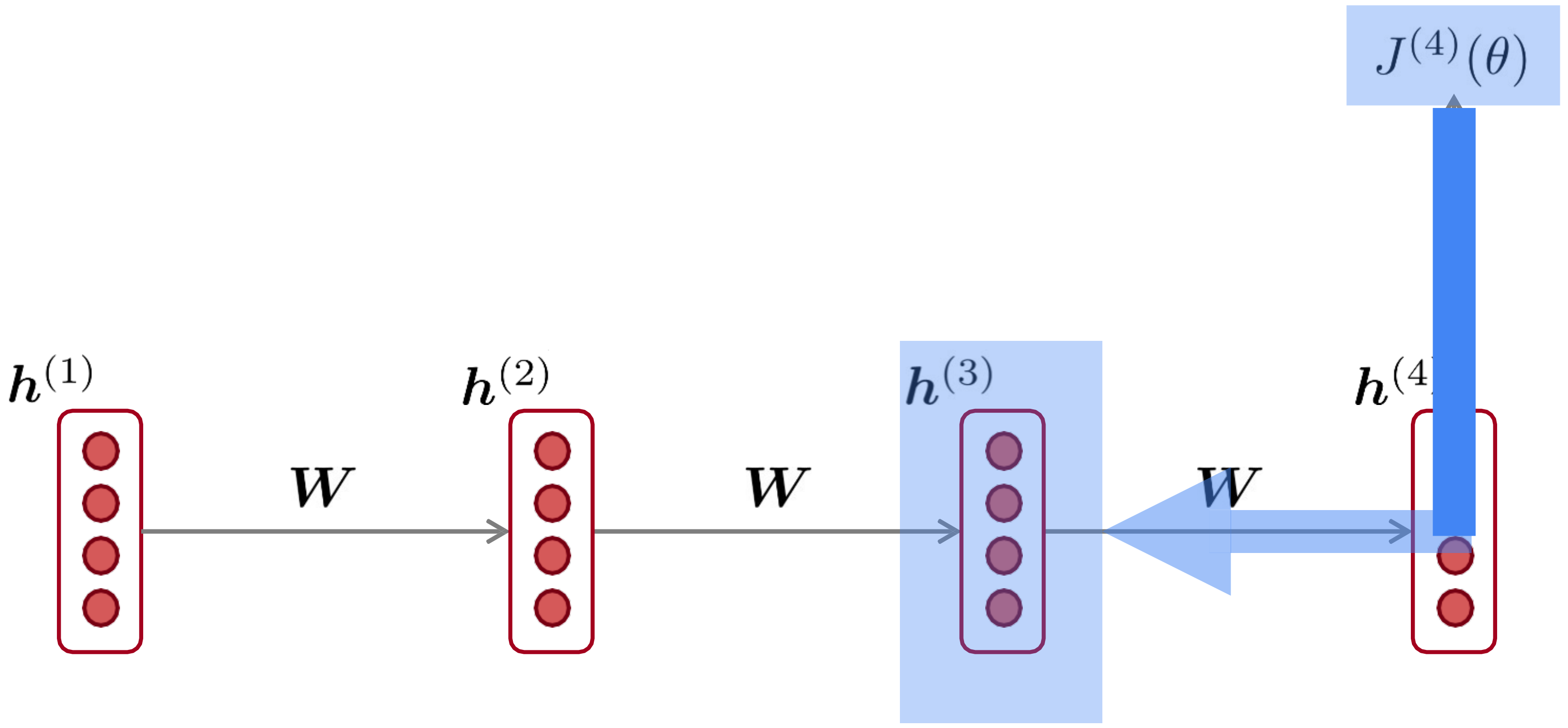
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$

chain rule!

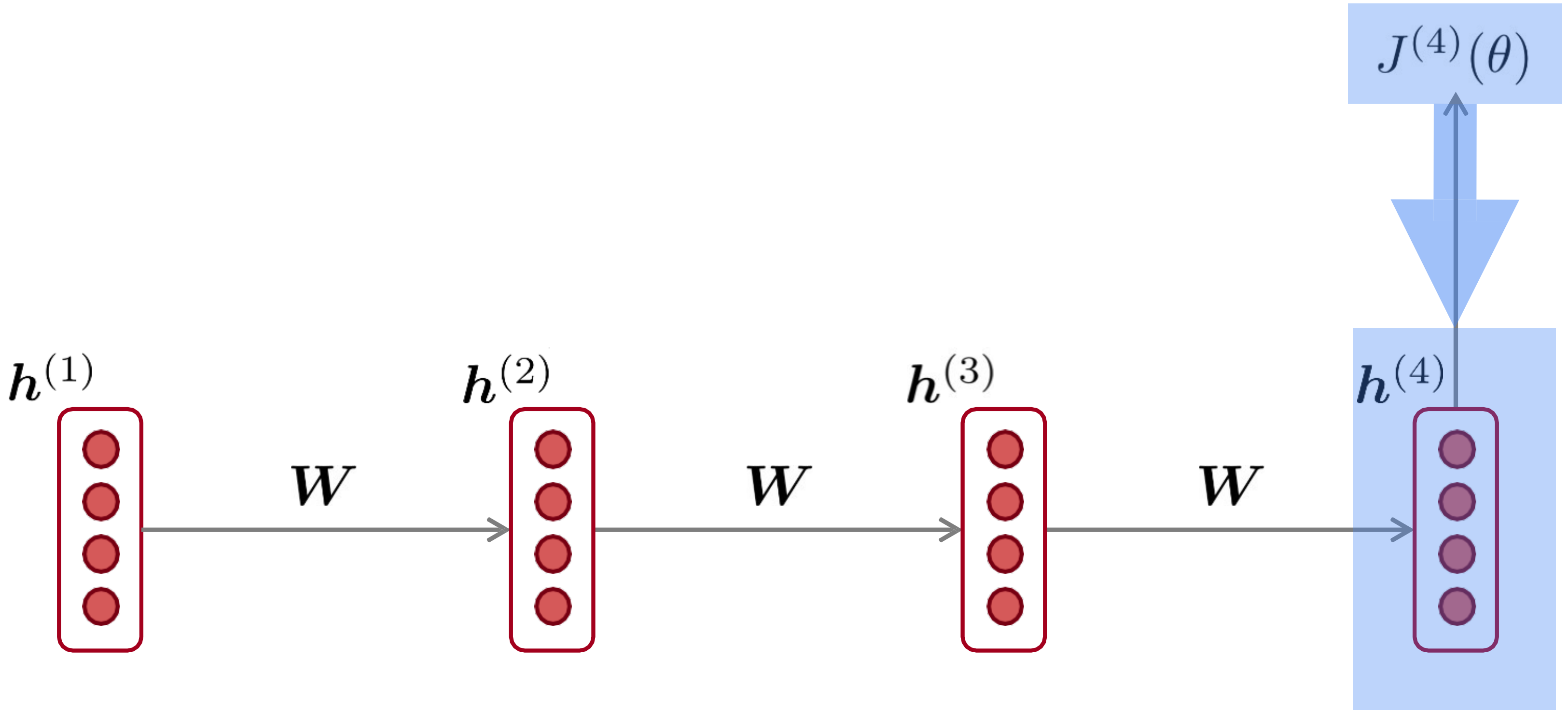
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \qquad \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(3)}}$$

chain rule!

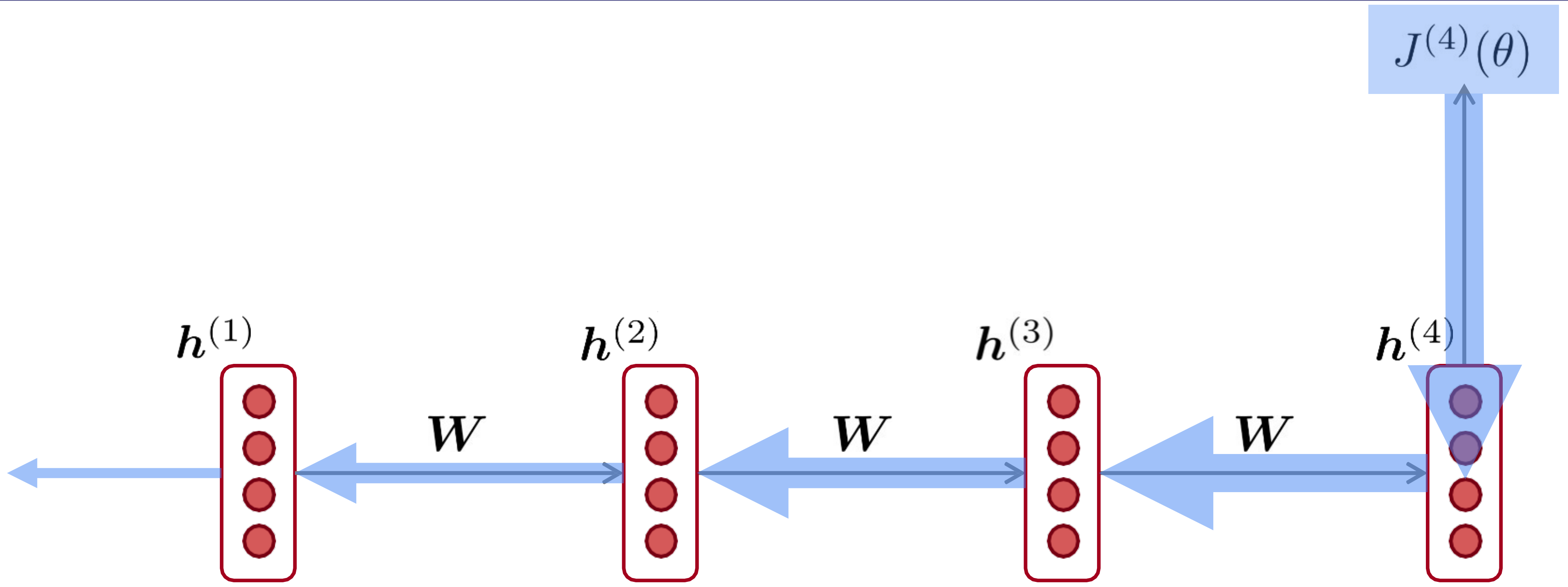
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

Vanishing gradient intuition

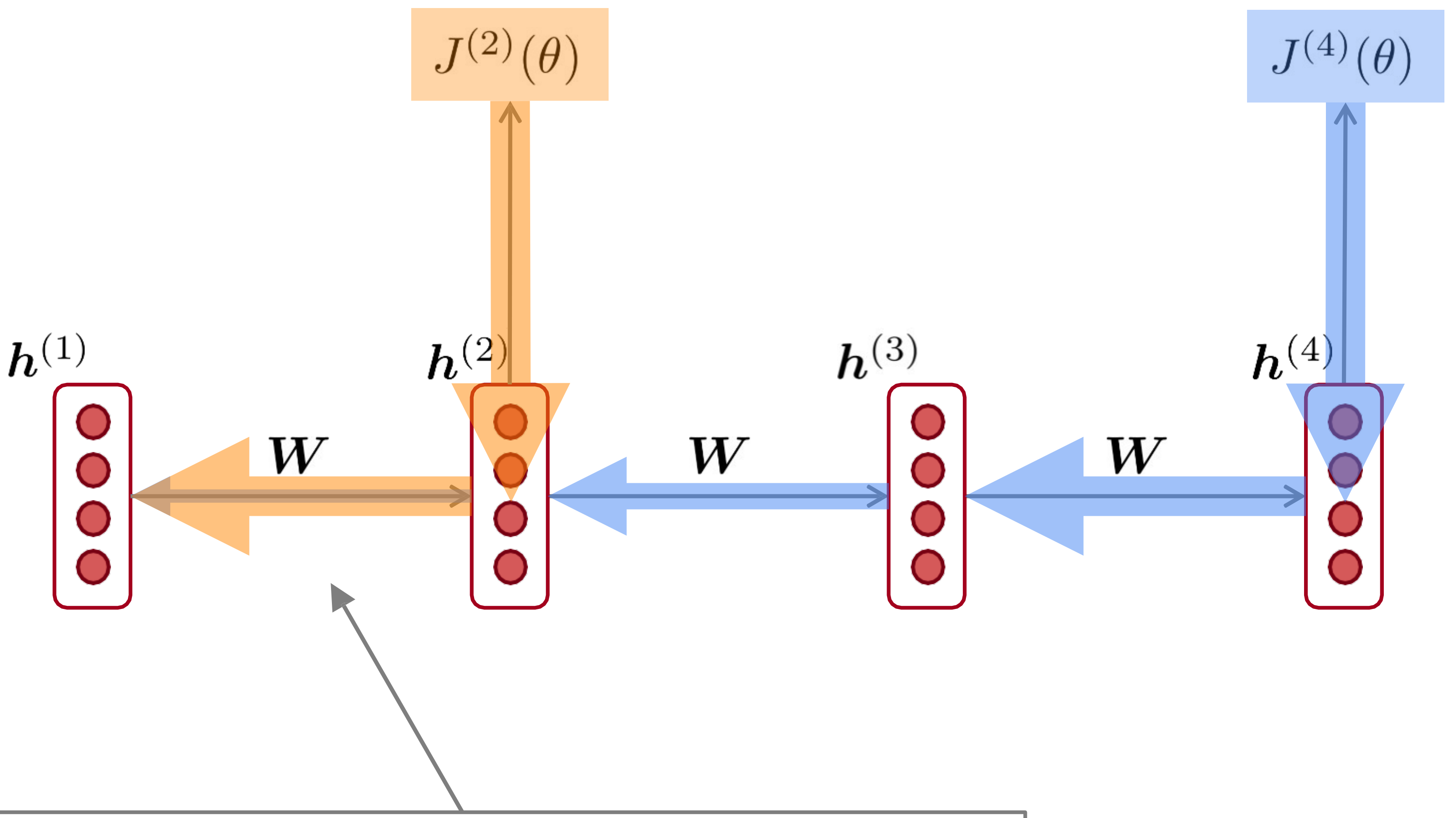


$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

What happens if these are small?

<u>Vanishing gradient problem:</u>
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Why is vanishing gradient a problem?



Gradient signal from faraway is lost because it's much smaller than gradient signal from close-by.

So model weights are only updated only with respect to near effects, not long-term effects.

The Problem with Vanilla RNNs (or Elman/Simple RNNs)

- Simple (Elman) architecture suffers from a problem known as vanishing gradients
  - Error signals from later steps in the sequence diminish quickly in the backpropagation algorithm
  - Thus, the updates for early inputs that come from errors in later steps are very small
- Solution: Gated architectures
  - Do not update the whole state at every step
  - Gate vectors define which parts of the new state are taken from the previous state and which from the current input
  - Ex.: Long short-term memory (LSTM), Gated Recurrent Unit (GRU)

Intuition behind the gating mechanism

- **Suppose that you were adding two quantities, a and b, but you wanted to control how much of b gets into the sum:**

$$a + \lambda b$$

- **λ is a value between 0 and 1.**
- **λ acts as a "switch" or a "gate" in controlling the amount of b that gets into the sum.**

A simple gate example

- **Elman RNN:** $h_t = h_{t-1} + F(h_{t-1}, x_t)$
- **A gated version of Elman RNN:**

$$h_t = h_{t-1} + \lambda(h_{t-1}, x_t) F(h_{t-1}, x_t)$$

  - **function λ controls how much of the current input gets to update the state h_{t-1};**
  - **function λ is context-dependent.**
- **Incorporate not only conditional updates, but also forgetting of the values in the previous state h_{t-1}**

$$h_t = \mu(h_{t-1}, x_t) h_{t-1} + \lambda(h_{t-1}, x_t) F(h_{t-1}, x_t)$$

LSTM intuitions

Memory cell:

- **Internal state serves as a memory**

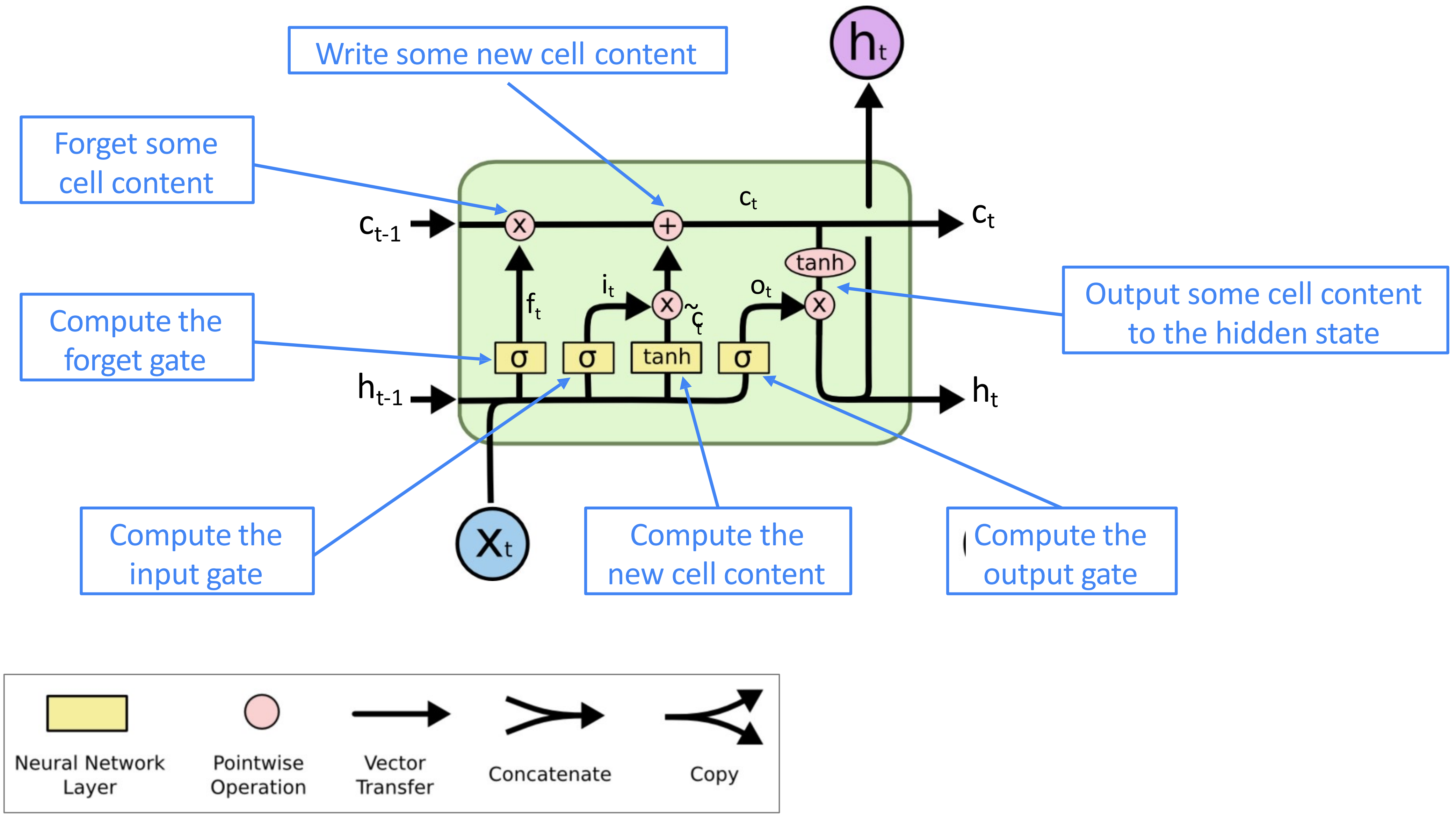
Gates:

- **Pointwise multiplication in gates regulates how much is passed through, based on inputs. This way, for instance, an LSTM can learn:**
- **when to reset its memory**
- **when to let the input in**
- **when to let the output out**

Long Short-Term Memory Networks (LSTMs)

1. **Explicitly splits the RNN state intwo two halves: $s_i = [c_i; h_i]$**
   - c_i is the „memory cell”, whereas h_i is the „working memory”
2. **Introduces differentiable gating mechanisms – smooth functions that simulate logical gates.**
   - Forget gate: decides which parts of the memory cell should be forgotten due to new input
   - Input/add gate: decides how much of the current input x_i should be written to the memory cell c_i
   - Output gate: decides which parts of the memory cell should be copied to the current hidden state / working memory
   - Gate vectors themselves are computed from the current input x_i and the previous state of the working memory h_{i-1}

Long Short-Term Memory Networks (LSTMs)



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gates: common design pattern

- All gates consist of a feed-forward layer, a sigmoid activation function, and a pointwise multiplication with the layer being gated
- Sigmoid as the activation function pushes its outputs to either 0 or 1.
- Combined with a pointwise multiplication it acts a sort of binary mask:
  - Values in the layer being gated that align with values near 1 in the mask are *passed through nearly unchanged*
  - Values corresponding to lower values are *essentially erased*

Forget gate

- It computes a weighted sum of the previous state’s hidden layer and the current input and passes that through a sigmoid
- The mask is then multiplied element-wise by the context vector to remove the information that is no longer required from the context (i.e., the memory cell)

$$\begin{aligned} \mathbf{f}_t &= \sigma(\mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{W}_f \mathbf{x}_t) \\ \mathbf{k}_t &= \mathbf{c}_{t-1} \odot \mathbf{f}_t \end{aligned}$$

Input/add gate

- Computes first the actual information we need to extract from the previous hidden state and current inputs (similar to what we did in vanilla RNNs)

$$\mathbf{g}_t = \tanh(\mathbf{U}_g\mathbf{h}_{t-1} + \mathbf{W}_g\mathbf{x}_t)$$

- Generates the mask to select the information to add to the current context

$$\begin{aligned}\mathbf{i}_t &= \sigma(\mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{W}_i\mathbf{x}_t) \\ \mathbf{j}_t &= \mathbf{g}_t \odot \mathbf{i}_t\end{aligned}$$

- Add this to the modified context vector to get the new context vector

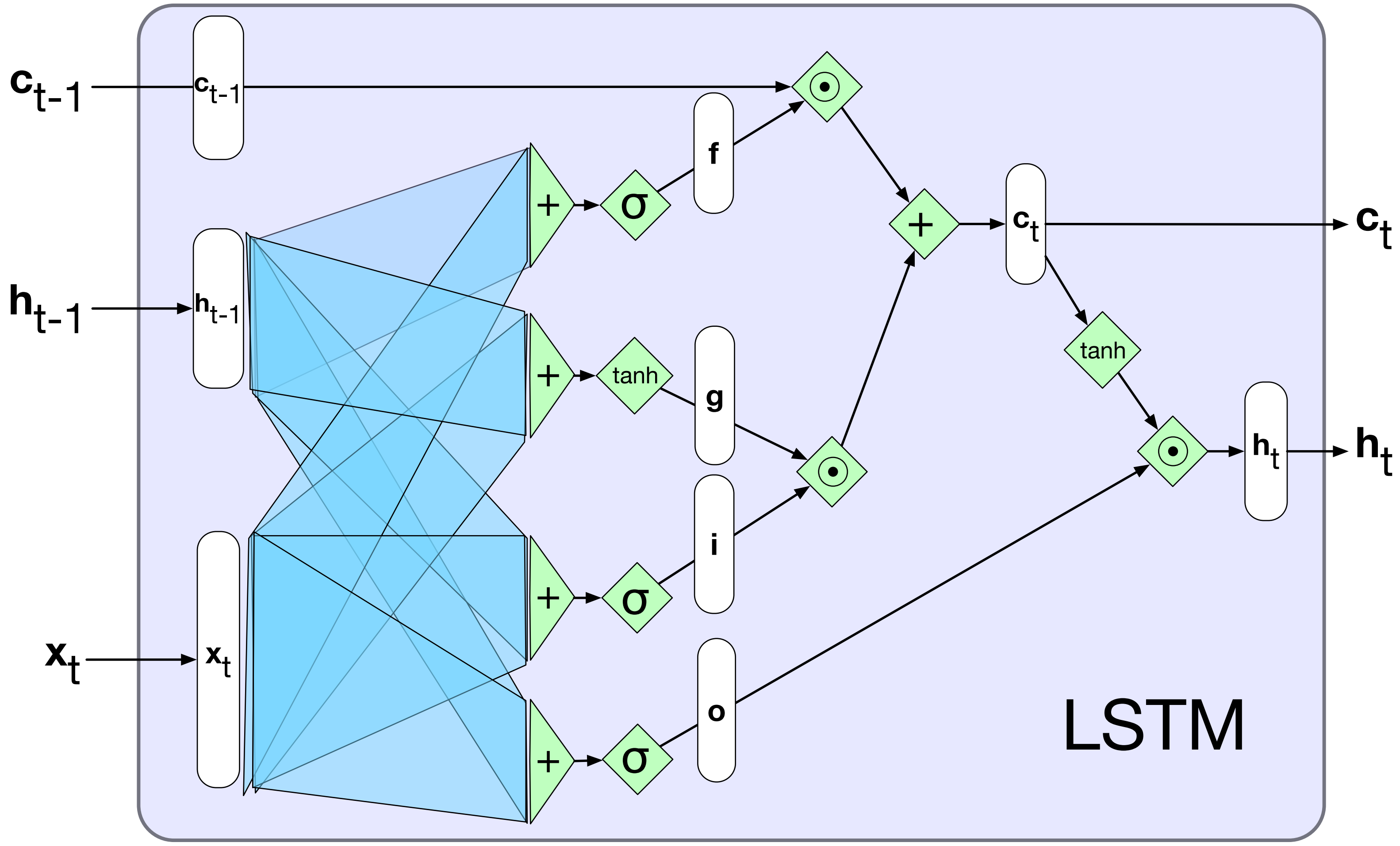
$$\mathbf{c}_t = \mathbf{j}_t + \mathbf{k}_t$$

Output gate

- Create a mask to select information from the memory cell that is required for the current hidden state

$$\begin{aligned}\mathbf{o}_t &= \sigma(\mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{W}_o \mathbf{x}_t) \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)\end{aligned}$$

Bringing it all together...



Web Content Mining

<u>**What we covered today**</u>

- **Named Entity Recognition**
- **Evaluation**
- **RNNs**