CS5242 : Neural Networks and Deep Learning

Lecture 9: Recurrent Neural Networks
Inference and Learning

Semester 1 2021/22

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- Word prediction
- Text generation
- Sentiment analysis

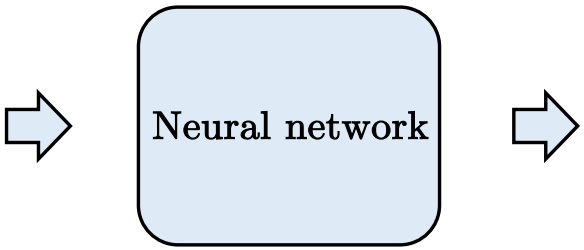
Outline

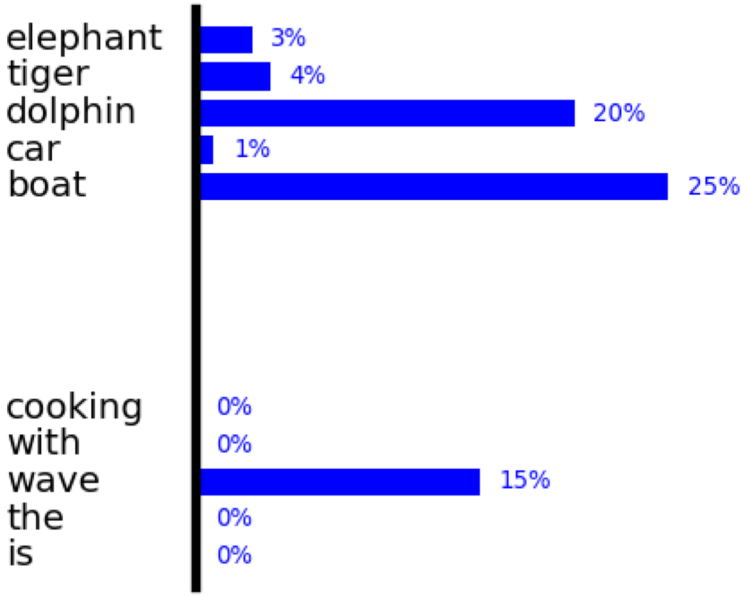
- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- Word prediction
- Text generation
- Sentiment analysis

Natural Language Processing

- NLP task: Predicting the next word.
  - This is the most fundamental problem in NLP.

"Yesterday I went to the beach and I saw a ..."





Input:

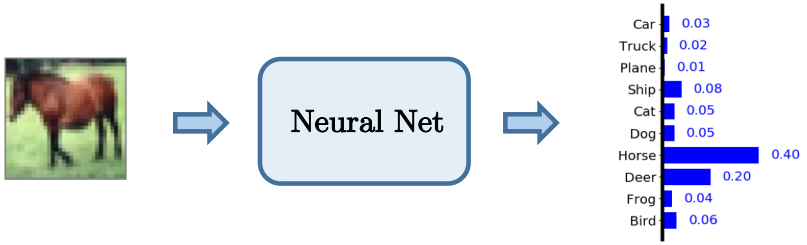
Sequence of words

Output:

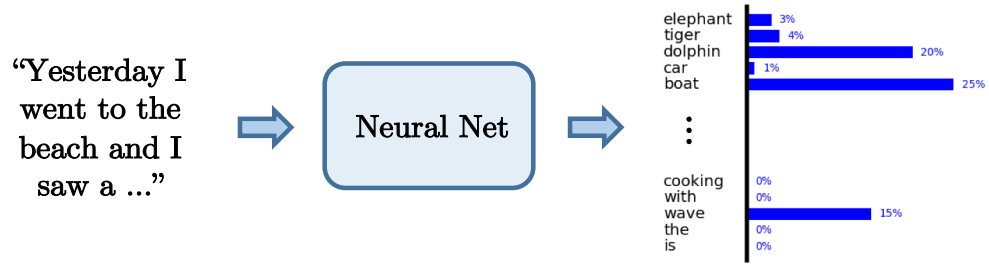
Probability distribution over the dictionary/vocabulary

NLP vs Computer Vision

- Computer vision and NLP have some important differences:
  - Image classification :
    - 10 classes
    - All images have the same size: 3 x 32 x 32

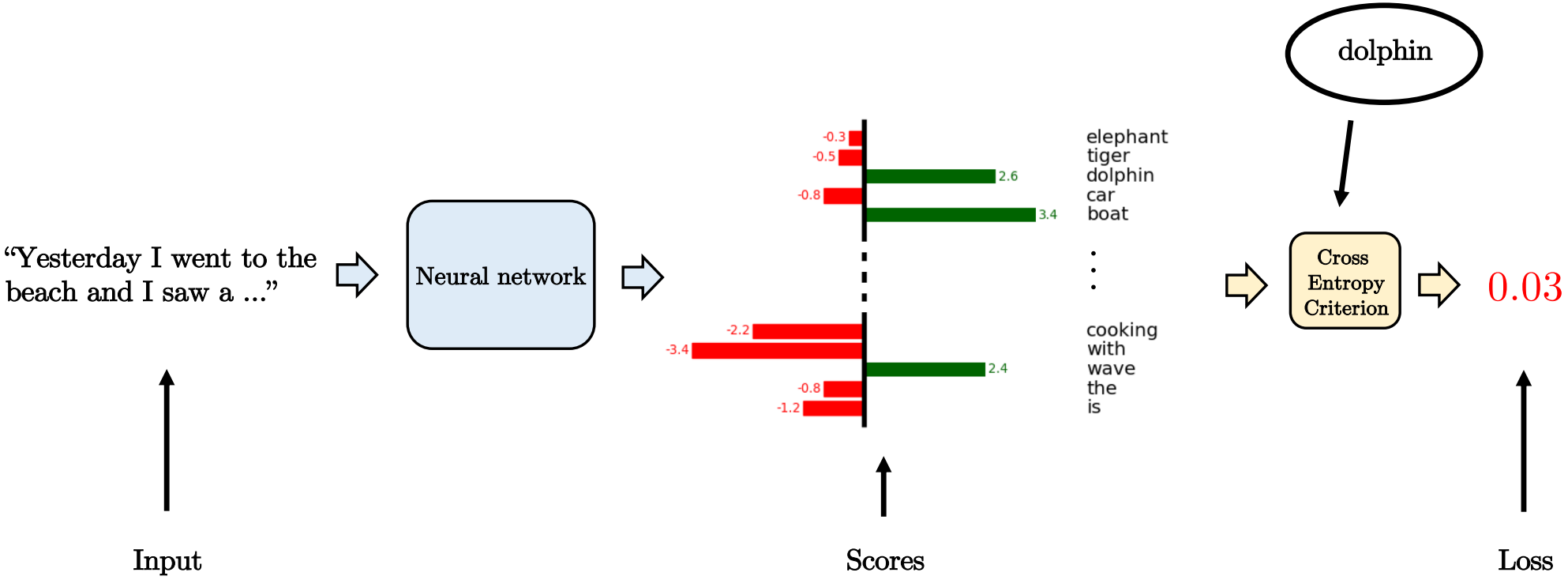


  - Language modeling :
    - 10,000 classes
    - Sentences have different sizes: 1 to 100 words
      - Average size is 15 words.



NLP vs Computer Vision

- But training NLP networks have also important similarities with computer vision:



Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- Word prediction
- Text generation
- Sentiment analysis

PTB dataset

- A benchmark dataset for NLP is Penn Tree Bank corpus (PTB):
  - Collection of articles from the Wall Street Journal :
    - Total length: 1 million words
    - Vocabulary size: 10,000
  - Notes:
    - Only the most frequent 10,000 words of the English vocabulary are kept.
    - Rare words are replaced by the word "<unk>".
    - Numbers are replaced by word "N".

PTB dataset

- Dataset: Text of length 1 million words

once again the specialists were not able to handle the imbalances on the floor of the new york stock exchange said christopher <unk> senior vice president at <unk> securities corp <unk> james <unk> chairman of specialists henderson brothers inc. it is easy to say the specialist is n't doing his job when the dollar is in a <unk> even central banks ca n't stop it speculators are calling for a degree of liquidity that is not there in the market many money managers and some traders had already left their offices early friday afternoon on a warm autumn day because the stock market was so quiet then in a <unk> plunge the dow jones industrials in barely an hour surrendered about a third of their gains this year <unk> up a N point or N N loss on the day in <unk> trading volume <unk> trading accelerated to N million shares a record for the big board at the end of the day N million shares were traded the dow jones industrials closed at N...

- Vocabulary: 10,000 most frequent words occurring in the corpus.

Word 1: "the"
Word 2: "and"
Word 3: "a"
Word 4: "that"
⋮
Word 101: "different"
Word 102: "day"
⋮
Word 1001: "chairman"
Word 1002: "industrial"
Word 1003: "conglomerate"
⋮
Word 9,998: "high-technology"
Word 9,999: "supercomputer"
Word 10,000: "<unk>"

Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- Word prediction
- Text generation
- Sentiment analysis

One-hot encoding

- Categorical data are usually represented by one-hot encoding vector :
  - A one-hot encoding vector is a Kronecker delta function.
  - Here, a vector of length 10,000 is associated to each word of the vocabulary :

$$\text{word 1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \text{word 2} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \text{word 3} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \ldots \quad \text{word 10,000} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

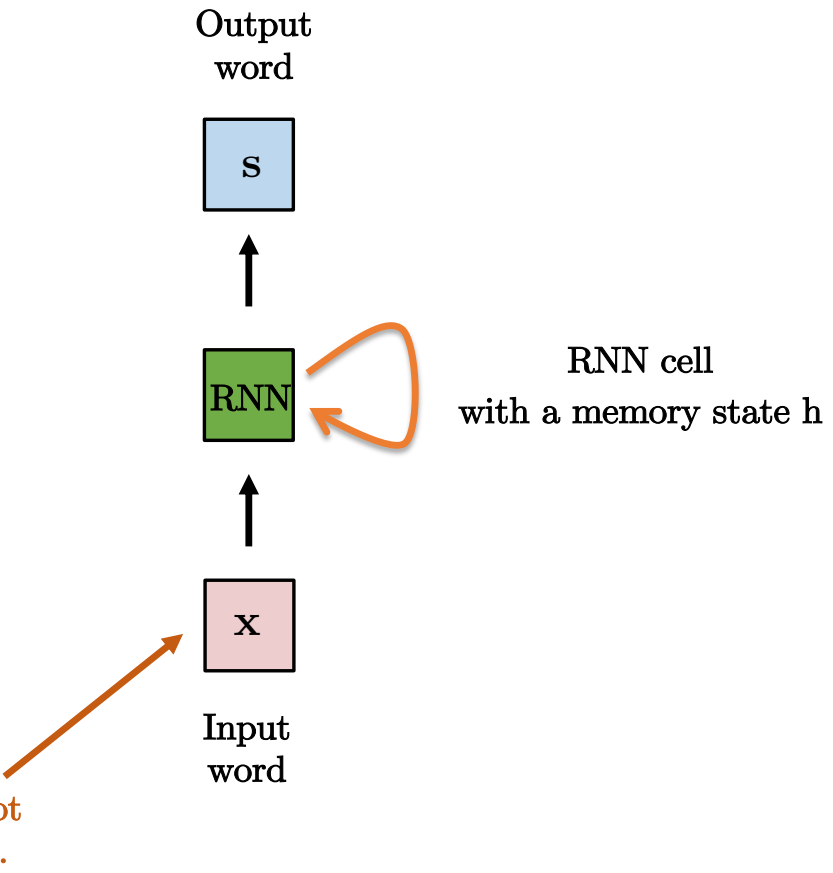
Word 1: "the" Word 2: "and" Word 3: "a" Word 10,000: "<unk>"

Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- **Vanilla recurrent neural networks**
- Training VRNNs
- Word prediction
- Text generation
- Sentiment analysis

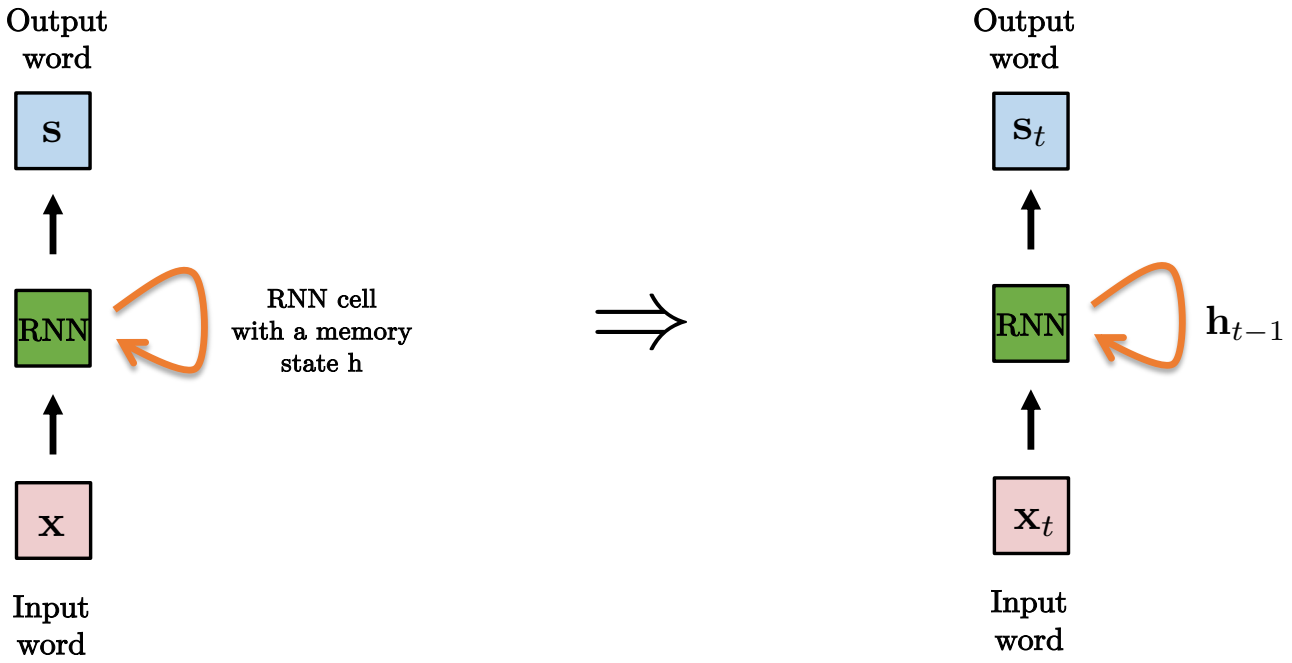
Vanilla RNNs

- RNNs are recurrent machine learning techniques :
  - Memory state : RNNs have a memory h that summarizes the past to predict the future. This memory/history vector will be learned (by backpropagation).
  - Prediction : RNNs use the input word x and the memory state h to predict the next word s.



Vanilla RNNs

- Introducing a time step t :



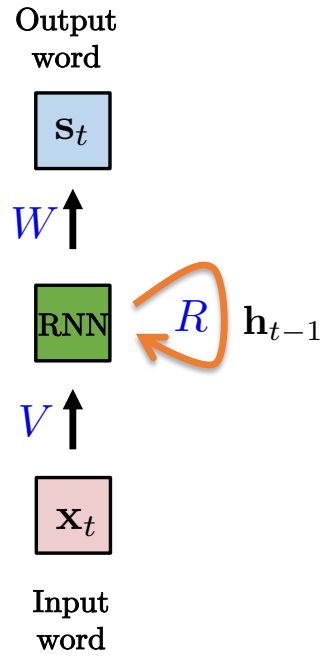
Vanilla RNNs

- The memory state h of VRNNs is updated with the recurrence formula :

$$\mathbf{h}_t = \sigma\left(R\,\mathbf{h}_{t-1} + V\,\mathbf{x}_t \right)$$

Time step t

Matrices R and V are trained to properly update the history vector h.



- The output prediction vector is :

$$\mathbf{s}_t = W\,\mathbf{h}_t$$

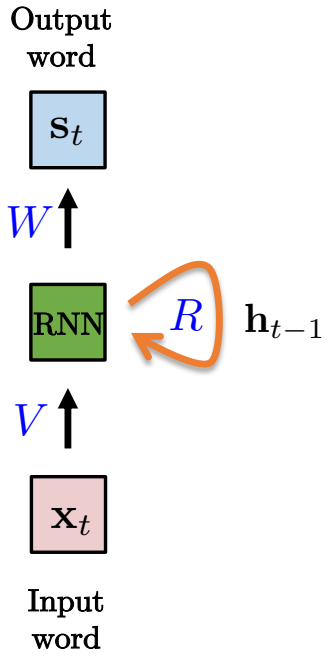
The matrix W is trained to use the history h and predict the next word.

Vanilla RNNs

- RNN properties:
  - Recurrence vector h and output vector s are independent of time step *t*.
    - The same formulas are used at every time step :

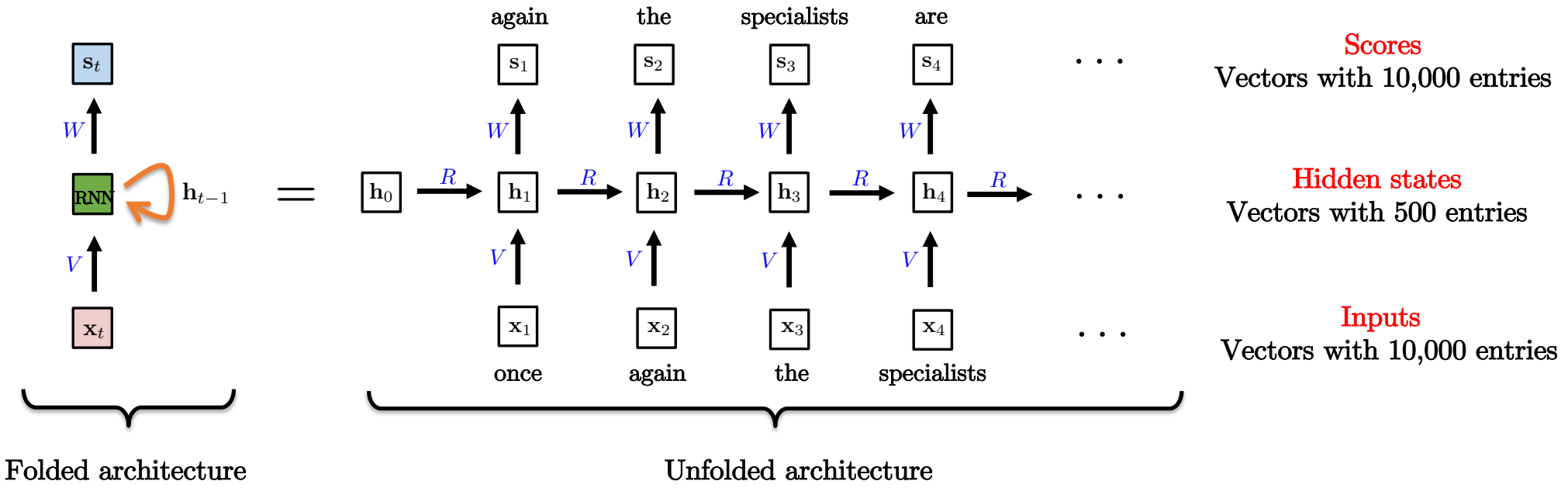
$$\mathbf{h}_t = \sigma(R\,\mathbf{h}_{t-1} + V\mathbf{x}_t)$$
$$\mathbf{h} \leftarrow \sigma(R\,\mathbf{h} + V\mathbf{x})$$

  - V, W, R matrices are the internal parameters of the RNN.
  - Updating V, W, R changes the dynamic behavior and prediction of the RNN.
  - Weights V, W, R are learned by backpropagation.



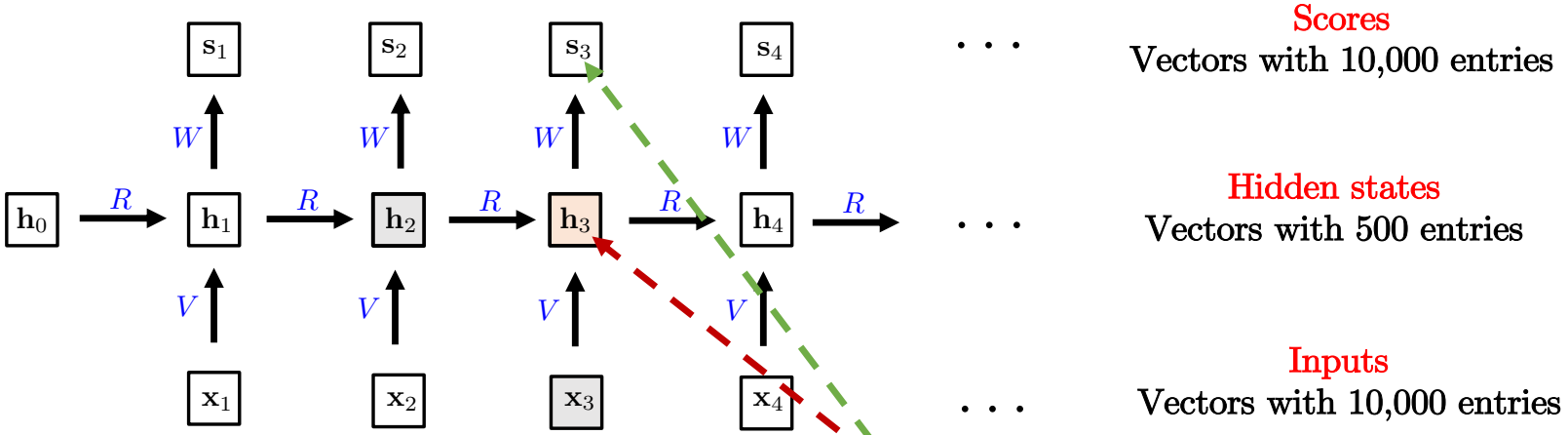
Unfolding RNNs

- Unrolled RNNs are easier to analyze :



Unfolding RNNs

- Let us compute the memory vector $\mathbf{h}_3$:



$$\mathbf{h}_t = \sigma\left(R\,\mathbf{h}_{t-1} + V\mathbf{x}_t \right)$$
$$\mathbf{s}_t = W\,\mathbf{h}_t$$

General formulas

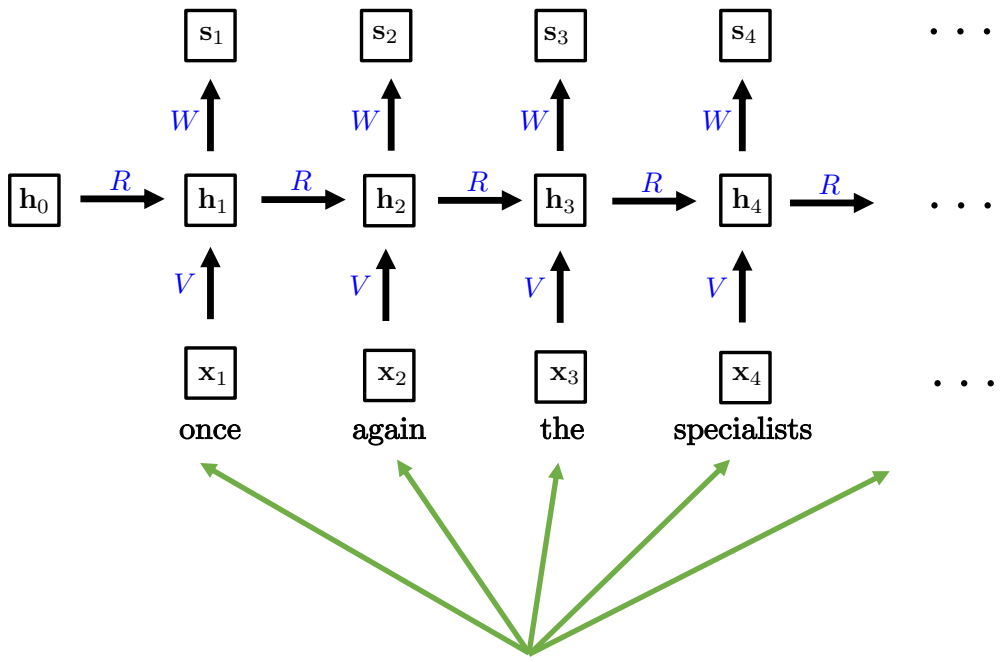
$\Rightarrow$

$$\mathbf{h}_3 = \sigma\left(R\,\mathbf{h}_2 + V\mathbf{x}_3 \right)$$
$$\mathbf{s}_3 = W\,\mathbf{h}_3$$

Formulas at time t=3

RNNs are (very) deep networks

- Number of layers = number of words in the document.
  - PTB has 1 millions words ⇒ 1 million layers !



Scores
Vectors with 10,000 entries

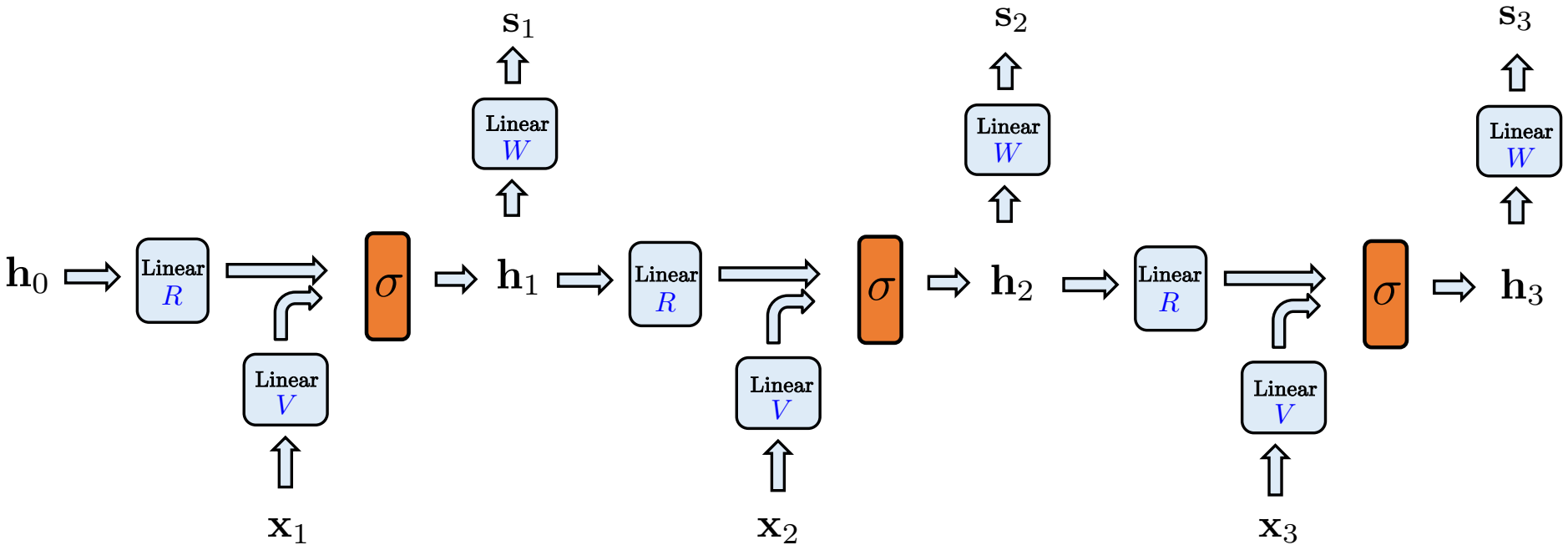
Hidden states
Vectors with 500 entries

Inputs
Vectors with 10,000 entries

1 million words contained in PTB = 1 million layers

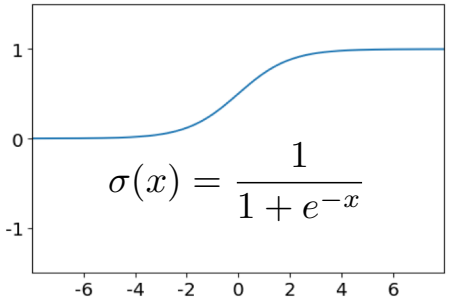
Weight sharing

- RNNs form a succession of layers (compositional function) that share the same parameters :
  - PTB has 1 millions words, which implies 1 million layers.
    - But there are only three weight matrices: R, V, W !

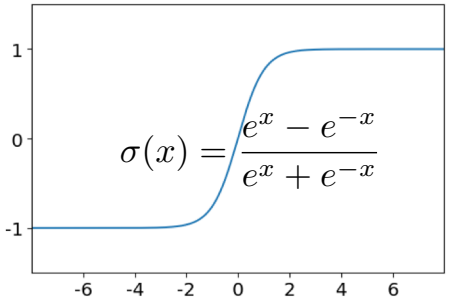


RNN non-linearities

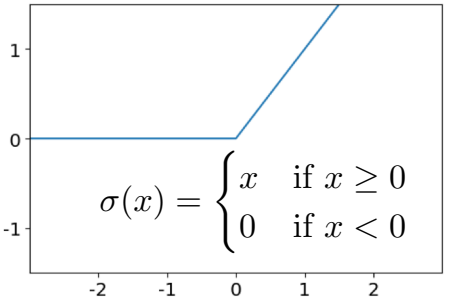
- RNNs use sigmoid and tanh for non-linear activations :
  - Sigmoid is a soft gate function : {0,1} → [0,1].
  - Tanh encodes information flow (positive and negative).
  - RNNs do not use ReLU as ConvNets, although ReLU does not suffer from the vanishing gradient problem (for the positive value).
    - Still, ReLU does not help here.
  - RNNs like LSTM or GRU are less prone to the vanishing gradient problem than VRNNs.
    - Still, LSTM or GRU cannot learn very long dependencies (no more than 50 steps).



Sigmoid



Tanh



ReLU

Example #1

- Tiny data set :

  the car is red

- Length :

  4 words

- Suppose we have a vocabulary of 6 words :

  Word 1: "the"
  Word 2: "is"
  Word 3: "bike"
  Word 4: "car"
  Word 5: "blue"
  Word 6: "red"

Example #1

- One-hot-encoding of the vocabulary :

Word 1: "the"
Word 2: "is"
Word 3: "bike"
Word 4: "car"
Word 5: "blue"
Word 6: "red"

Vocabulary

$$\text{the} = \begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{is} = \begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix} \quad \text{bike} = \begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix} \quad \text{car} = \begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix} \quad \text{blue} = \begin{bmatrix}0\\0\\0\\0\\1\\0\end{bmatrix} \quad \text{red} = \begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}$$

One-hot encoding

- As a consequence, the text becomes a sequence of vectors :

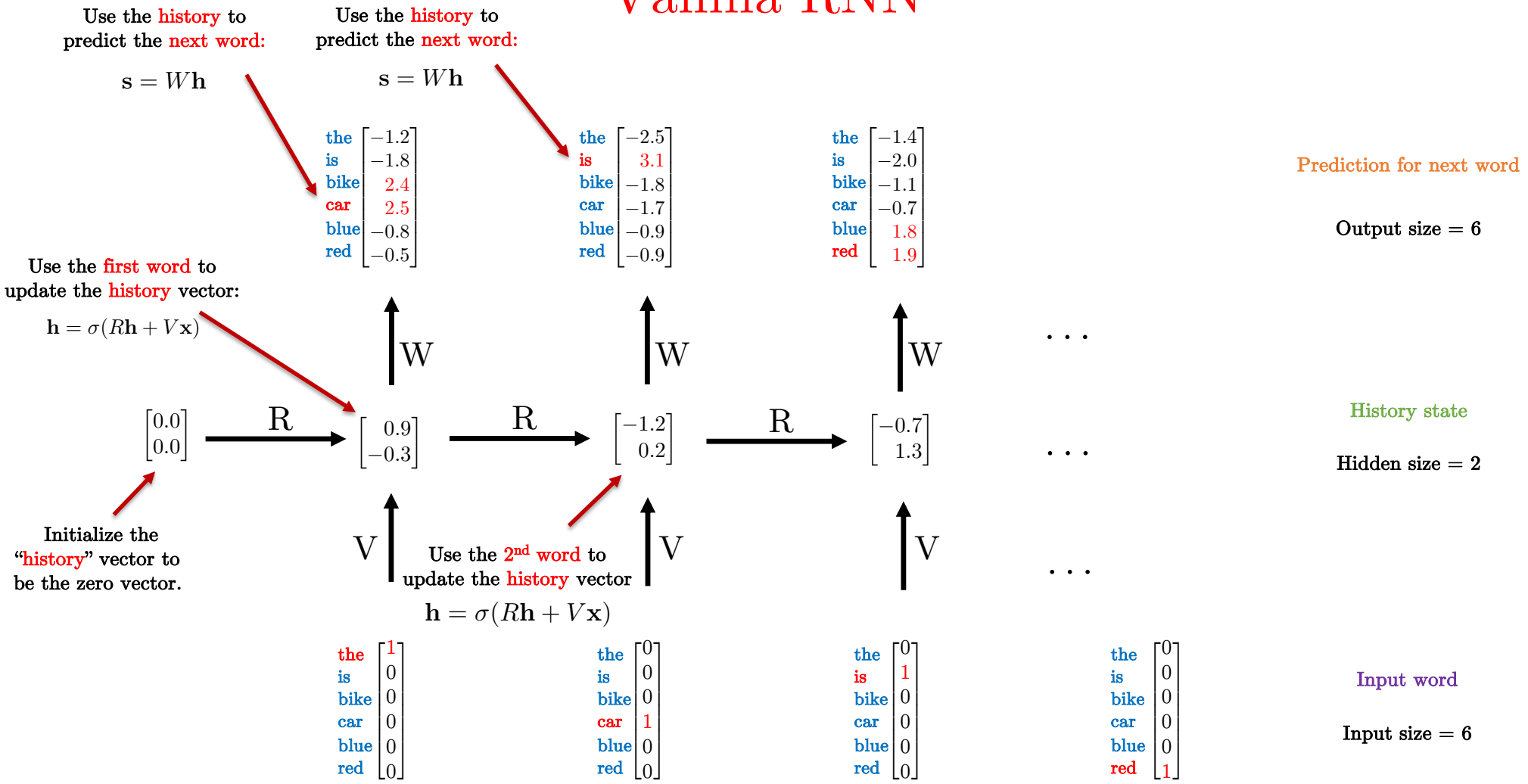
"the car is red"

Sequence of 4 words

$$\begin{matrix}\text{the}\\ \text{is}\\ \text{bike}\\ \text{car}\\ \text{blue}\\ \text{red}\end{matrix}\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix} \quad \begin{matrix}\text{the}\\ \text{is}\\ \text{bike}\\ \text{car}\\ \text{blue}\\ \text{red}\end{matrix}\begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix} \quad \begin{matrix}\text{the}\\ \text{is}\\ \text{bike}\\ \text{car}\\ \text{blue}\\ \text{red}\end{matrix}\begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix} \quad \begin{matrix}\text{the}\\ \text{is}\\ \text{bike}\\ \text{car}\\ \text{blue}\\ \text{red}\end{matrix}\begin{bmatrix}0\\0\\0\\0\\0\\1\end{bmatrix}$$

Sequence of 4 one-hot vectors

Vanilla RNN


Use the history to predict the next word:
$\mathbf{s} = W\mathbf{h}$
Use the history to predict the next word:
$\mathbf{s} = W\mathbf{h}$
the −1.2 −2.5 −1.4
is −1.8 3.1 −2.0
bike 2.4 −1.8 −1.1
car 2.5 −1.7 −0.7
blue −0.8 −0.9 1.8
red −0.5 −0.9 1.9
Use the first word to update the history vector:
$\mathbf{h} = \sigma(R\mathbf{h} + V\mathbf{x})$
W
W
W
R
R
R
0.0 0.9 −1.2 −0.7
0.0 −0.3 0.2 1.3
Initialize the "history" vector to be the zero vector.
V
V
V
Use the 2nd word to update the history vector
$\mathbf{h} = \sigma(R\mathbf{h} + V\mathbf{x})$
the 1 0 0 0
is 0 0 1 0
bike 0 0 0 0
car 0 1 0 0
blue 0 0 0 0
red 0 0 0 1
Prediction for next word
Output size = 6
History state
Hidden size = 2
Input word
Input size = 6

Example #2

- Dataset: Text of length 1 million words

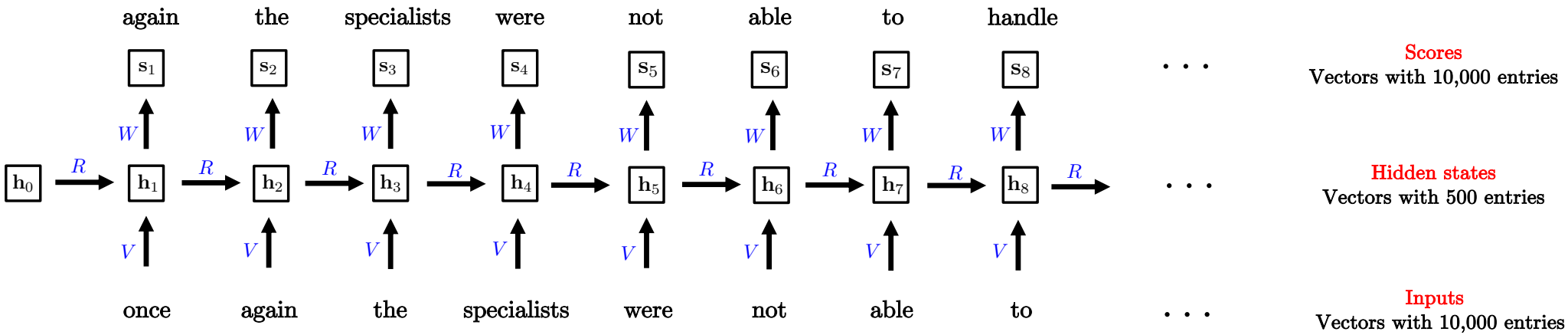
once again the specialists were not able to handle the imbalances on the floor of the new york stock exchange said christopher <unk> senior vice president at <unk> securities corp <unk> james <unk> chairman of specialists henderson brothers inc. it is easy to say the specialist is n't doing his job when the dollar is in a <unk> even central banks ca n't stop it speculators are calling for a degree of liquidity that is not there in the market many money managers and some traders had already left their offices early friday afternoon on a warm autumn day because the stock market was so quiet then in a <unk> plunge the dow jones industrials in barely an hour surrendered about a third of their gains this year <unk> up a N point or N N loss on the day in <unk> trading volume <unk> trading accelerated to N million shares a record for the big board at the end of the day N million shares were traded the dow jones industrials closed at N...

- Vocabulary: 10,000 most frequent words occurring in the corpus.

Word 1: "the"
Word 2: "and"
Word 3: "a"
Word 4: "that"
⋮
Word 101: "different"
Word 102: "day"
⋮
Word 1001: "chairman"
Word 1002: "industrial"
Word 1003: "conglomerate"
⋮
Word 9,998: "high-technology"
Word 9,999: "supercomputer"
Word 10,000: "<unk>"

Example 2

- As we go through the text, the history vector memorizes/extracts all the relevant information:
  - The matrix V and R learn how to properly update the history.
  - The matrix W learns how to use the history to predict the next word.



Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- **Training VRNNs**
- Word prediction
- Text generation
- Sentiment analysis

Backpropagation

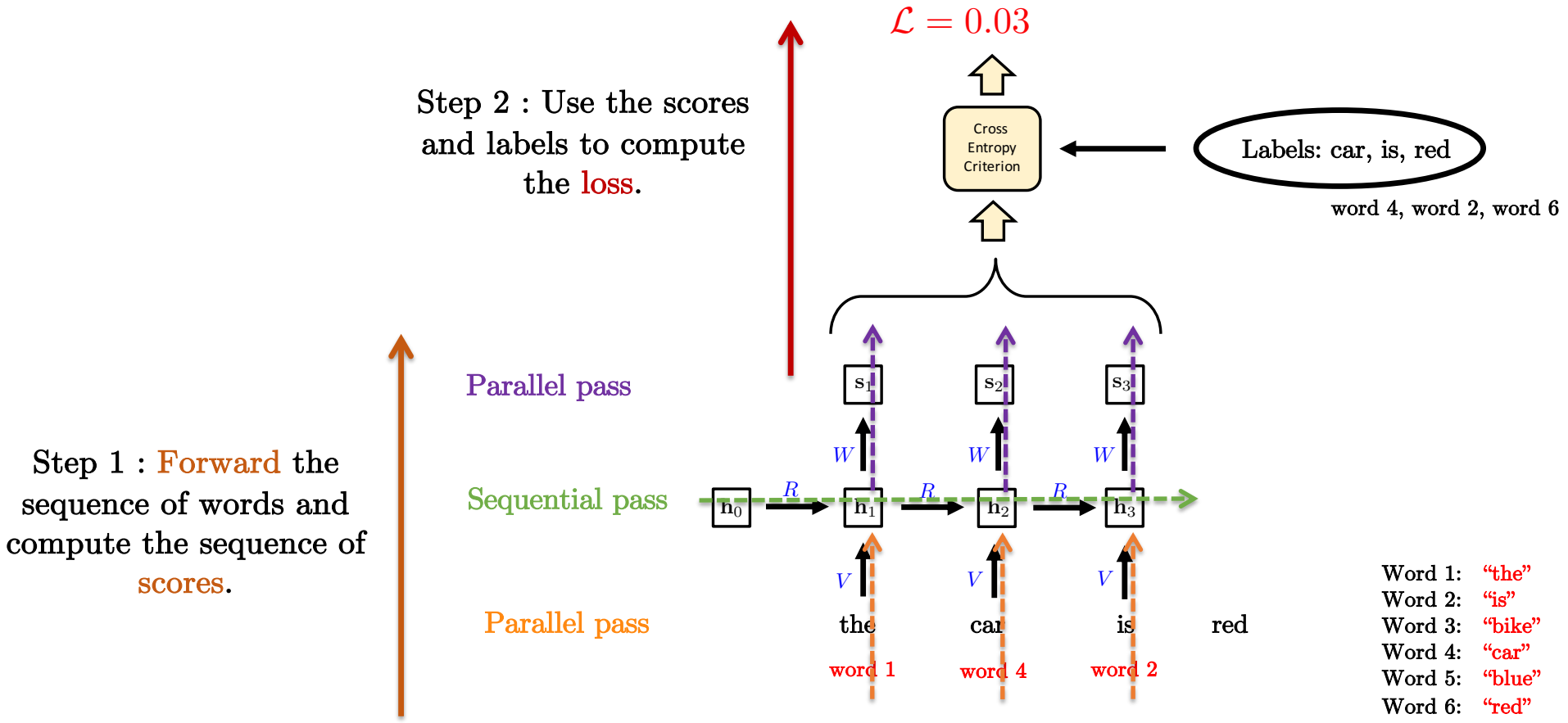
- Backpropagation
  - Step 1: Forward pass
  - Step 2: Cross-entropy loss
  - Step 3: Backward pass

- Let see this on the tiny dataset:

  the car is red

Word 1: "the"
Word 2: "is"
Word 3: "bike"
Word 4: "car"
Word 5: "blue"
Word 6: "red"

Backpropagation



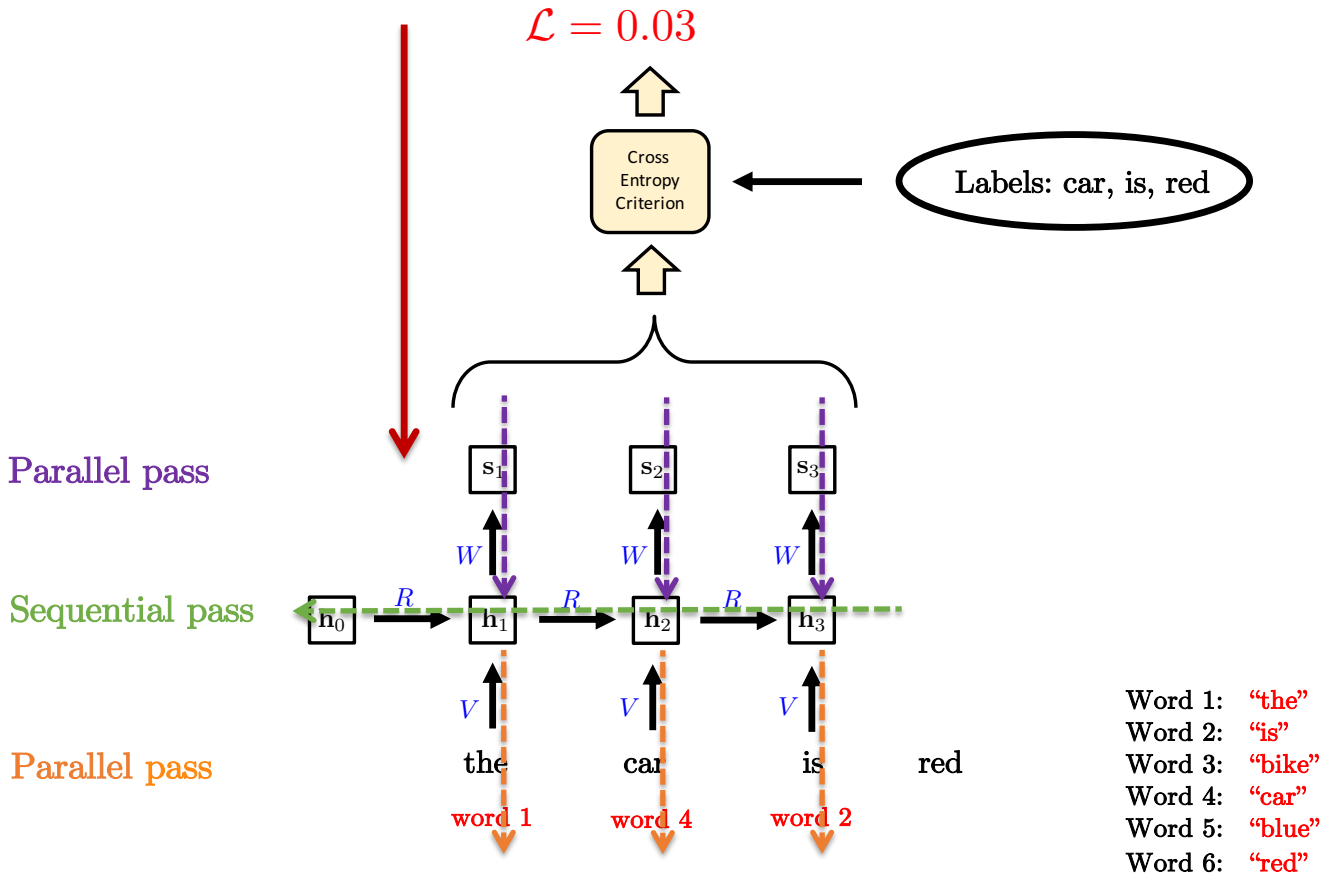
Backpropagation

- Step 3 :
  - Do the backward pass to compute the gradients.
  - Do one step of SGD to update the weights:

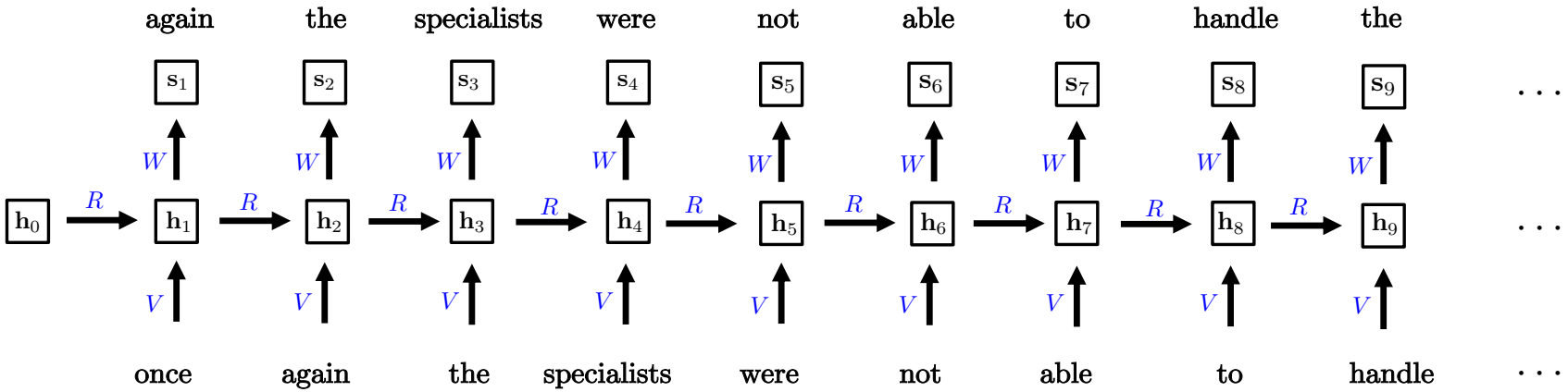
$$R = R - \text{lr}\frac{\partial \mathcal{L}}{\partial R}$$

$$V = V - \text{lr}\frac{\partial \mathcal{L}}{\partial V}$$

$$W = W - \text{lr}\frac{\partial \mathcal{L}}{\partial W}$$

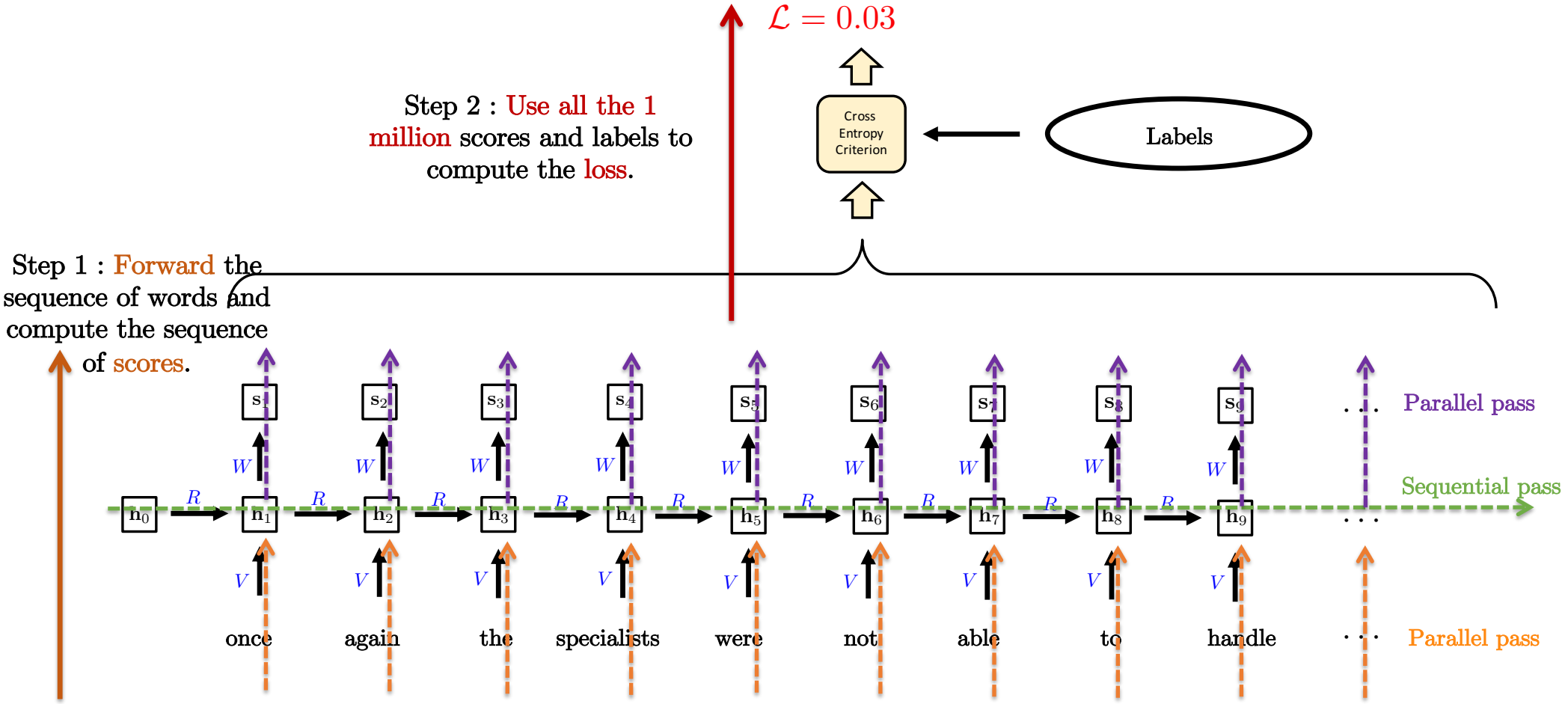


Backpropagation on large datasets



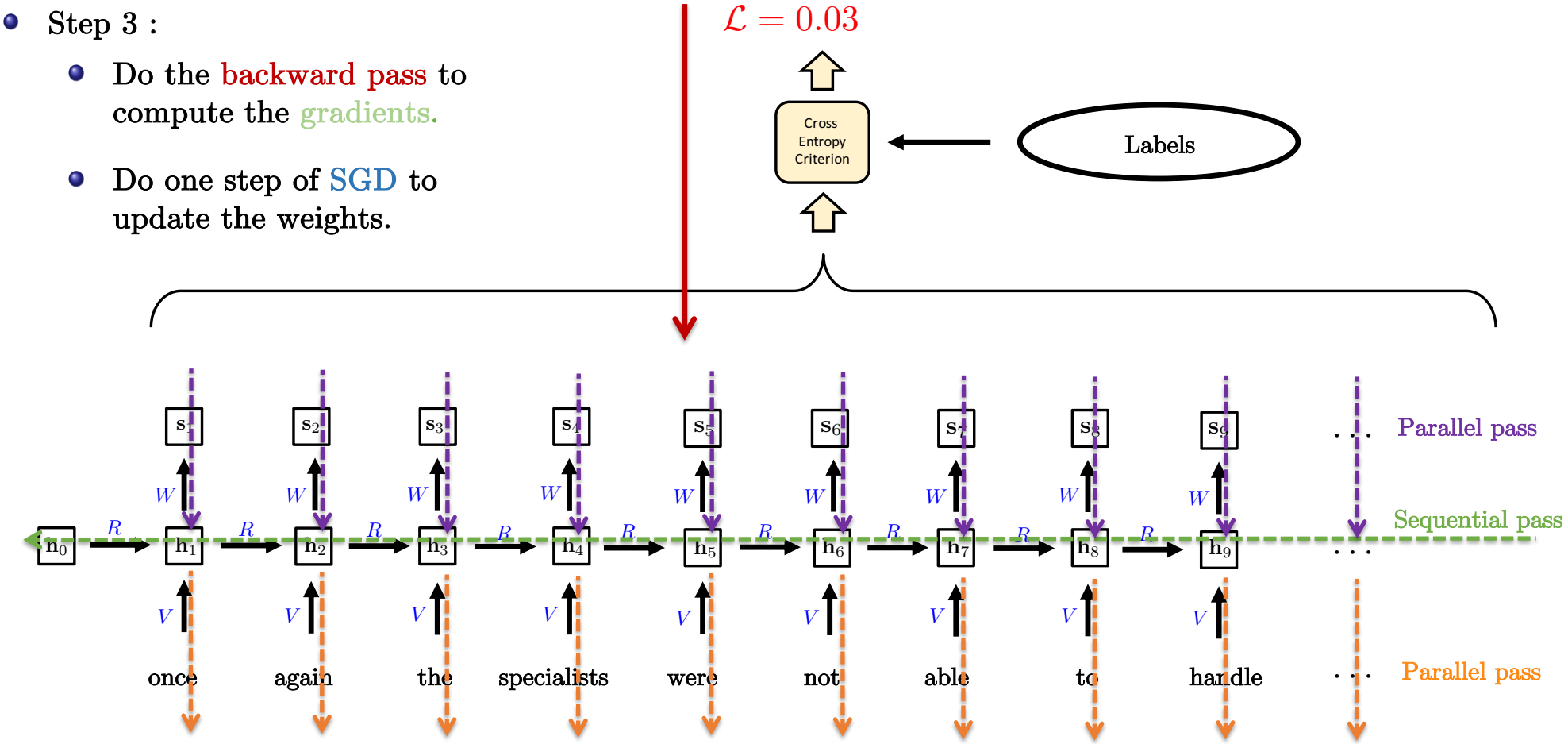
- Backpropagation on PTB :
  - The input sequence has 1 million of words.
  - We are dealing with a 1-million-layer network ⇒ We need to backpropagate the error through 1 million layers !
  - It is not speed and memory tractable to backpropagate through that much layers.

Backpropagation on large datasets



Backpropagation on large datasets

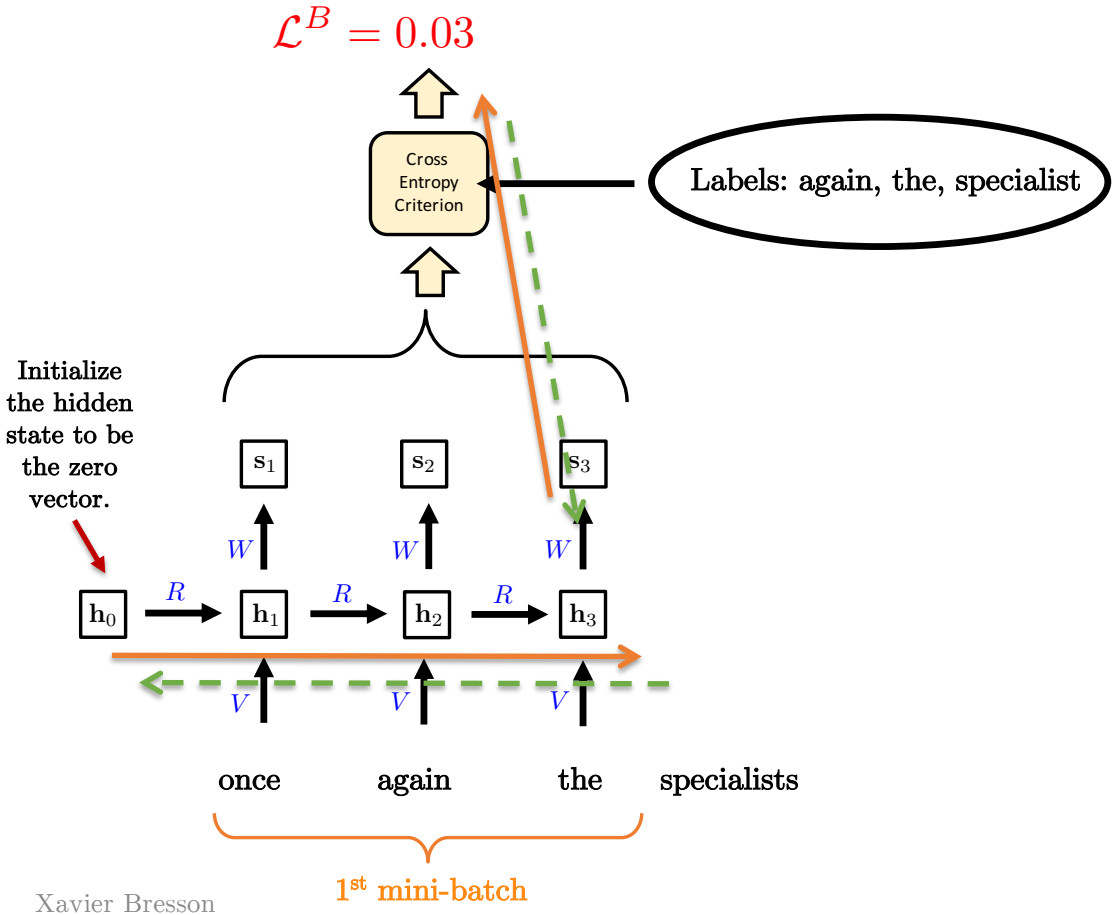
- Step 3 :
  - Do the backward pass to compute the gradients.
  - Do one step of SGD to update the weights.



Backpropagation on large datasets

- We cannot backpropagate a large number of steps.
  - So we want to proceed by mini-batch (speed and memory efficiency).
  - For ConvNets and images:
    - The mini-batches were chosen at random.
  - For RNNs and any type of sequences:
    - The mini-batches cannot be randomly chosen because of the causal nature of NLP signals.
    - The mini-batches are composed of a number B of sequential words.

Backpropagation with mini-batch



- Step 1 : Forward the first batch of words and compute the first batch of scores.
- Step 2 : Use the scores and labels to compute the average loss over the mini-batch B.
- Step 3 : Do the backward pass to compute the gradients and one one step of SGD to update the weights:

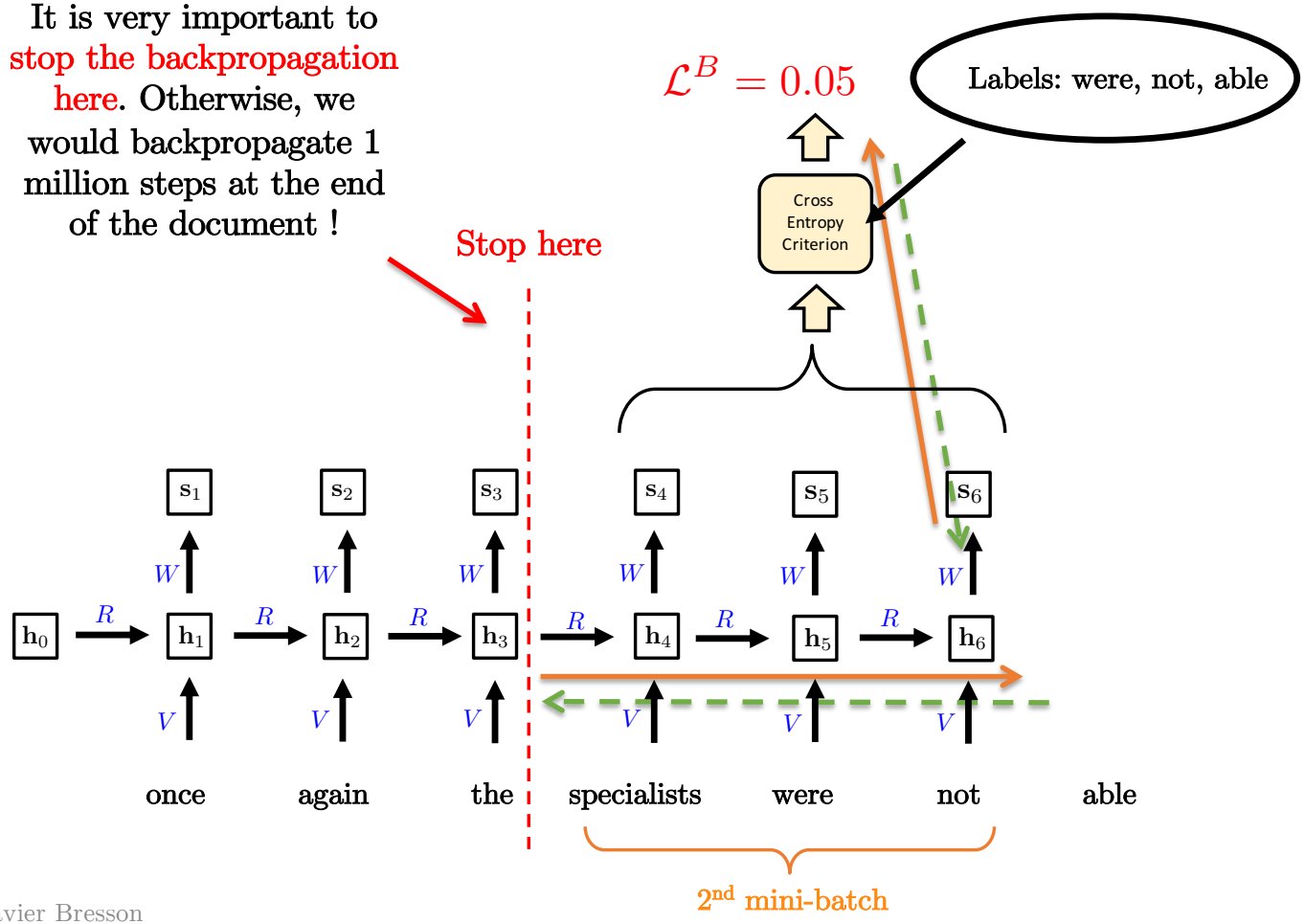
$$R = R - \text{lr}\frac{\partial \mathcal{L}^B}{\partial R}$$

$$V = V - \text{lr}\frac{\partial \mathcal{L}^B}{\partial V}$$

$$W = W - \text{lr}\frac{\partial \mathcal{L}^B}{\partial W}$$

Backpropagation with mini-batch

It is very important to stop the backpropagation here. Otherwise, we would backpropagate 1 million steps at the end of the document !



- Step 1 : Forward the second batch of words and compute the second batch of scores.
- Step 2 : Use the scores and labels to compute the average loss over the mini-batch B.
- Step 3 : Do the backward pass to compute the gradients and one one step of SGD to update the weights:

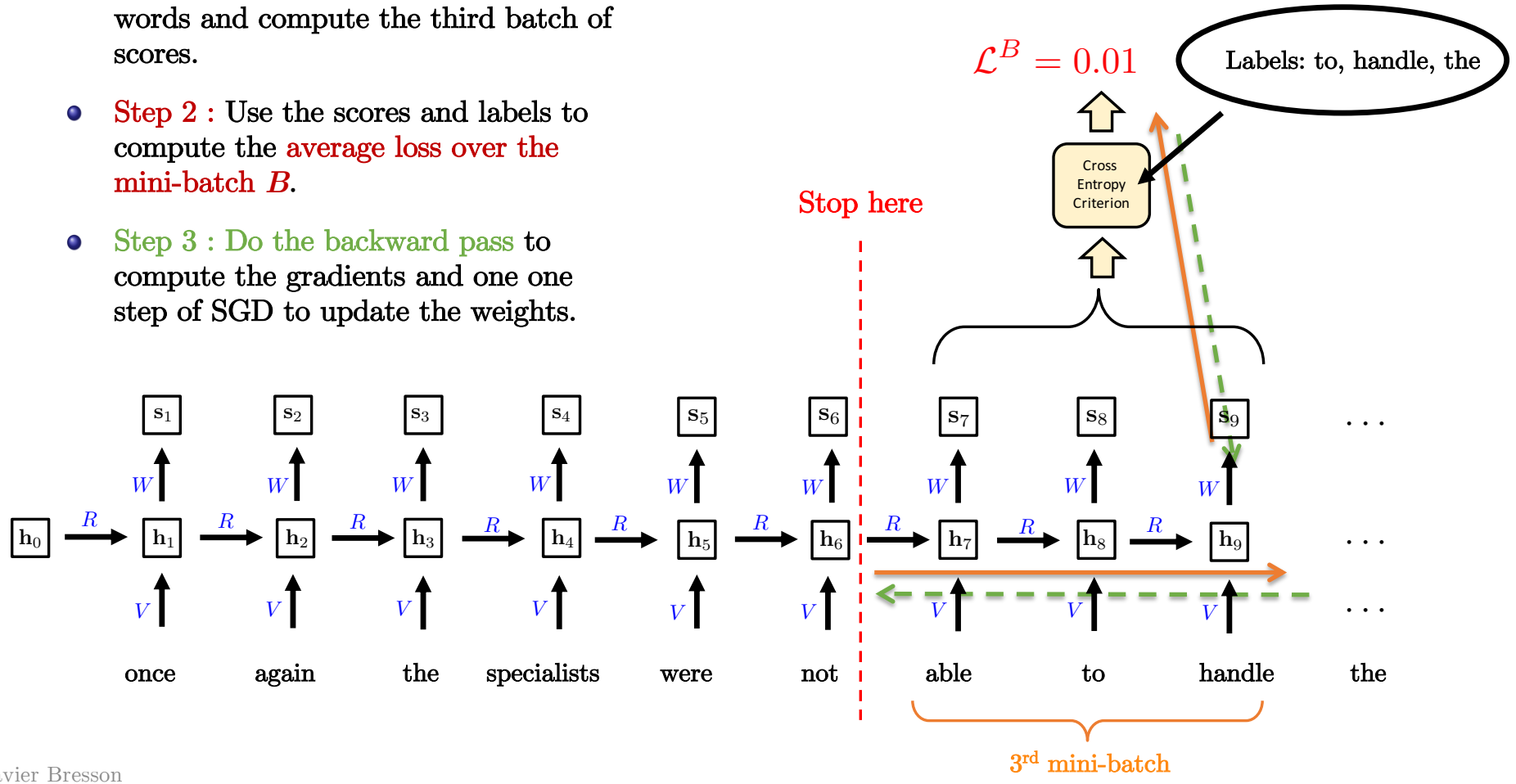
$$R = R - \text{lr}\frac{\partial \mathcal{L}^B}{\partial R}$$

$$V = V - \text{lr}\frac{\partial \mathcal{L}^B}{\partial V}$$

$$W = W - \text{lr}\frac{\partial \mathcal{L}^B}{\partial W}$$

Backpropagation with mini-batch

- Step 1 : Forward the third batch of words and compute the third batch of scores.
- Step 2 : Use the scores and labels to compute the average loss over the mini-batch B.
- Step 3 : Do the backward pass to compute the gradients and one one step of SGD to update the weights.



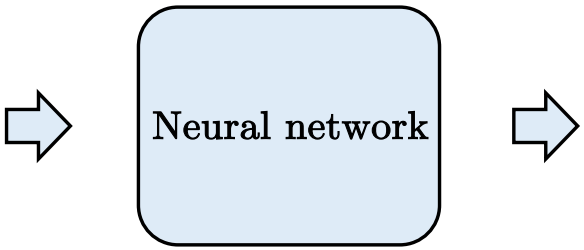
Outline

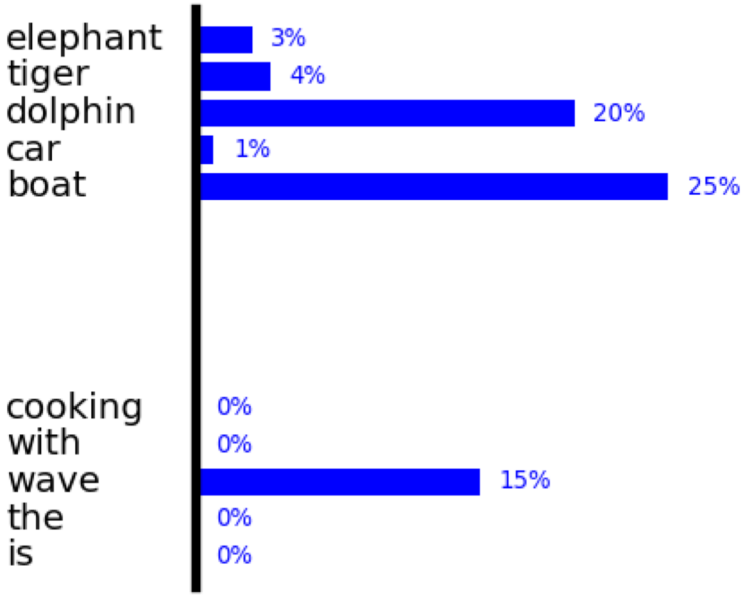
- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- **Word prediction**
- Text generation
- Sentiment analysis

Natural Language Processing

- NLP task: Predicting the next word.
  - This is the most fundamental problem in NLP.

"Yesterday I went to the beach and I saw a ..."





Input:

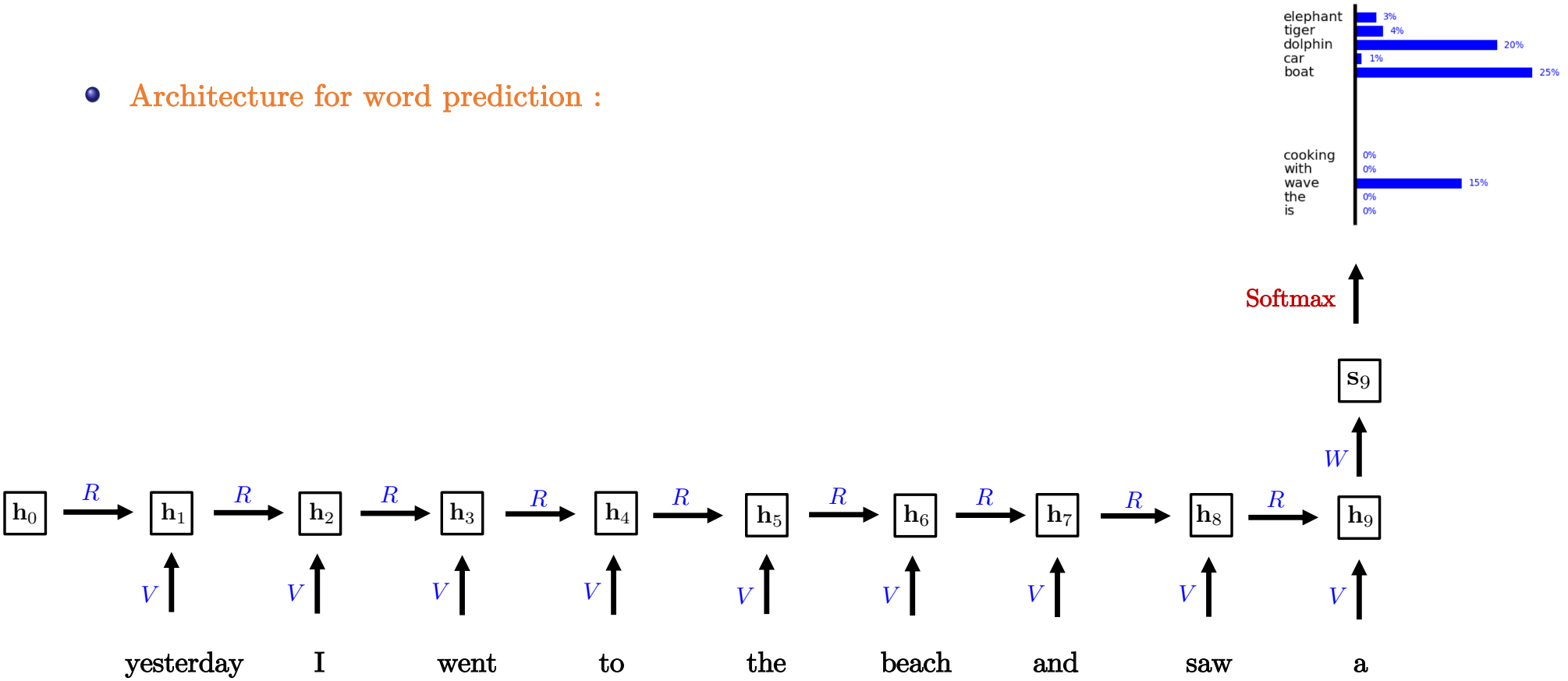
Sequence of words

Output:

Probability distribution over the dictionary/vocabulary

Language modeling

- Architecture for word prediction :

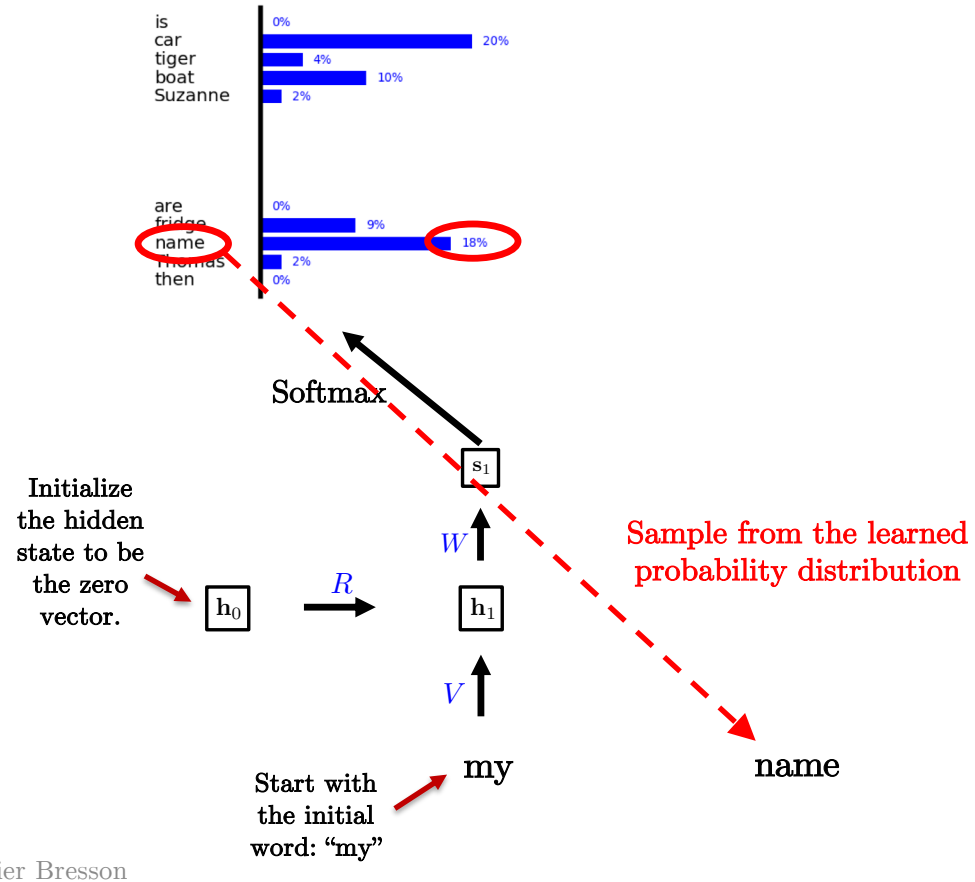


Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- Word prediction
- **Text generation**
- Sentiment analysis

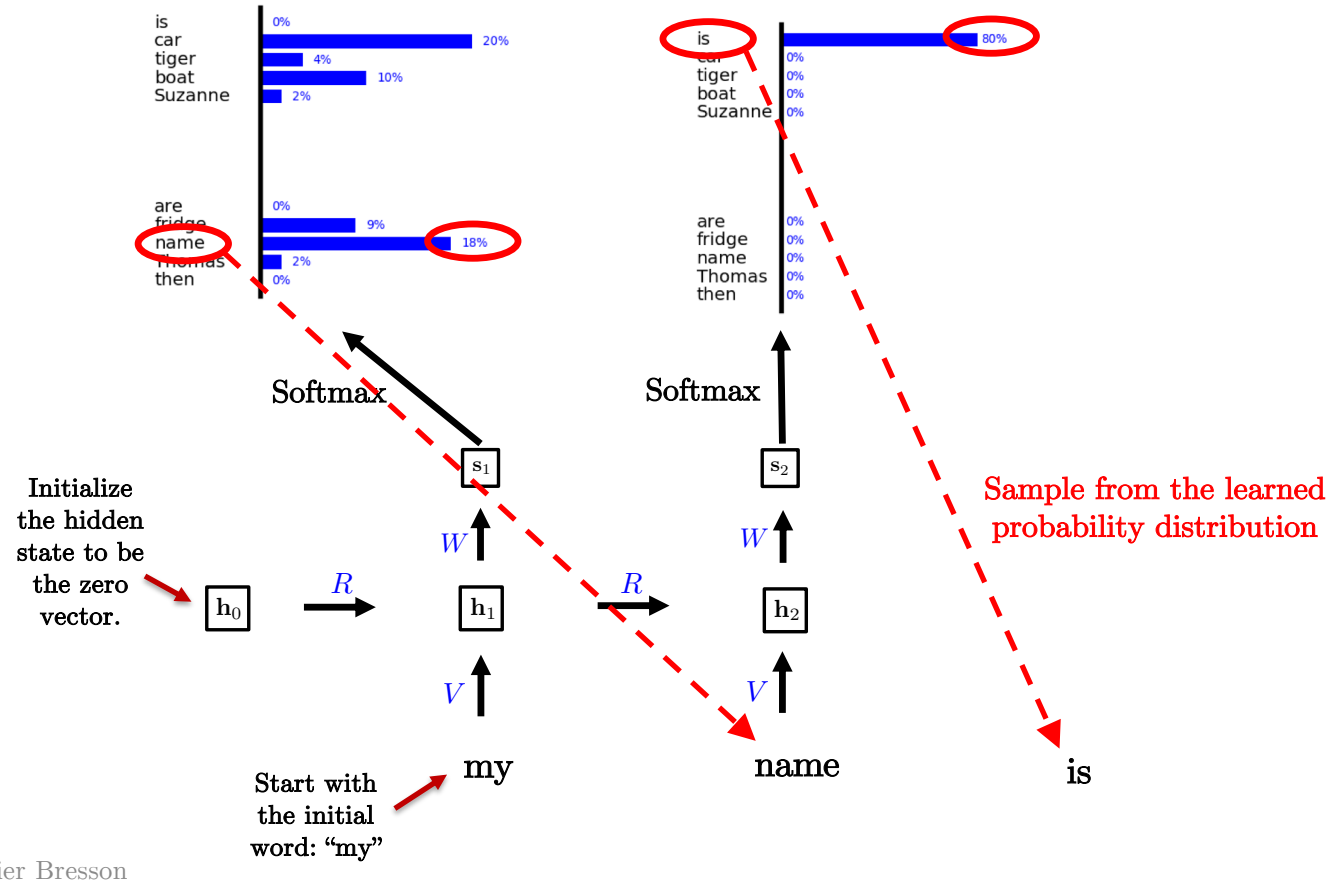
Text generation

- Text generation (with the same trained matrices R, V and W) :



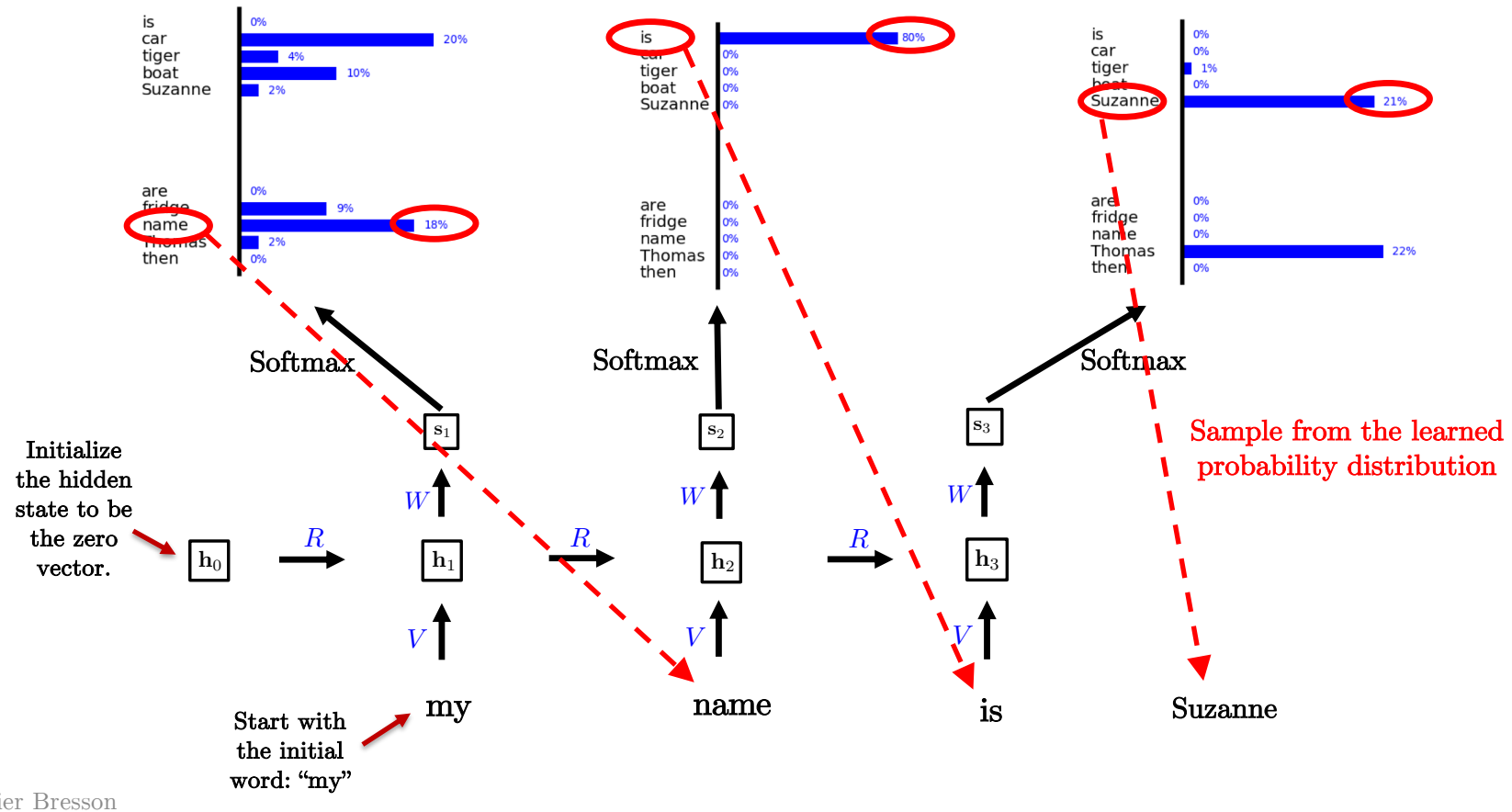
Text generation

- Text generation (with the same trained matrices R, V and W) :



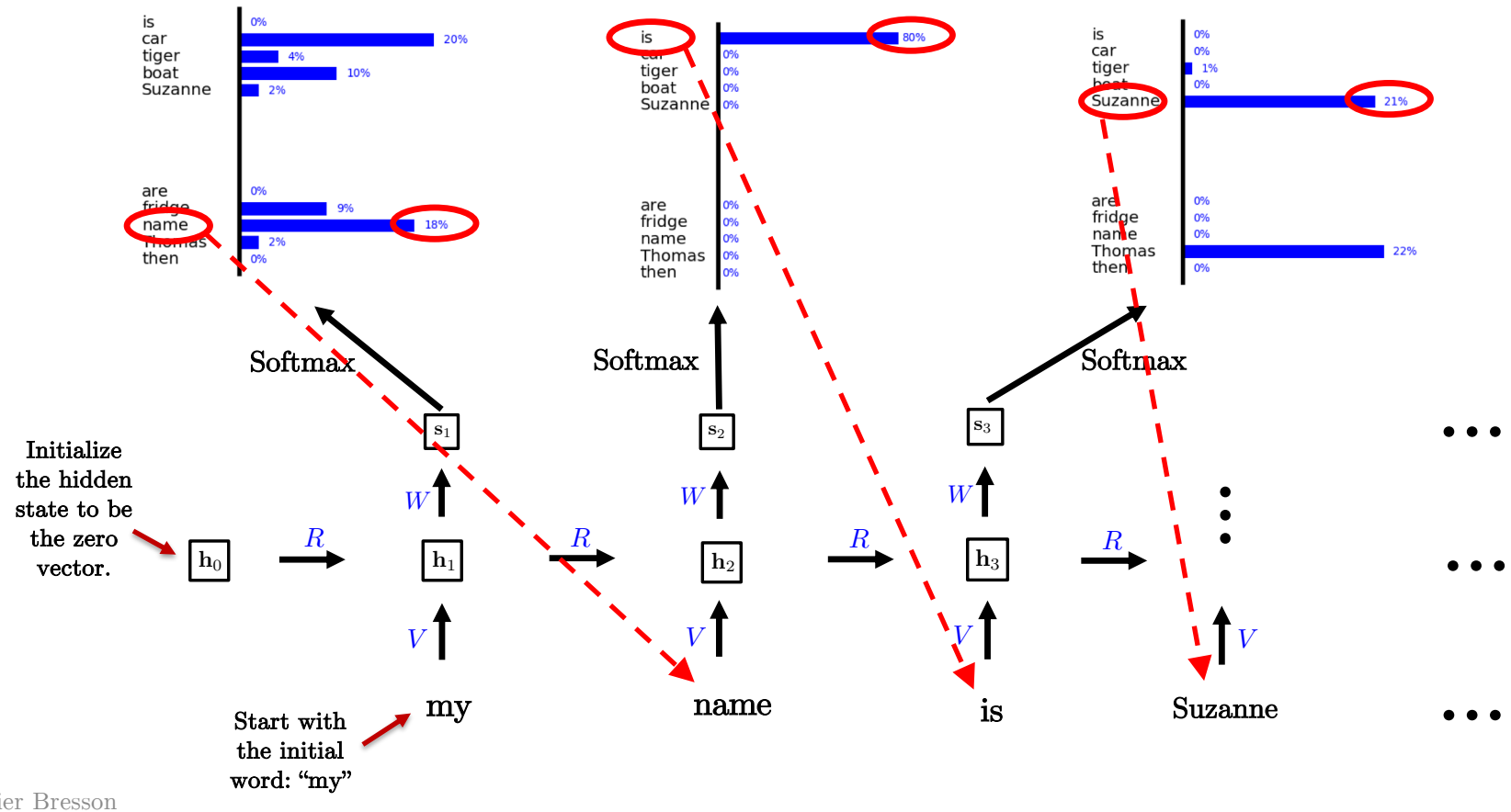
Text generation

- Text generation (with the same trained matrices R, V and W) :



Text generation

- Text generation (with the same trained matrices R, V and W) :



Text generation

- Shakespeare-like generation (character-level):

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Math generation

- Math-like generation using textbooks on algebraic geometric :

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

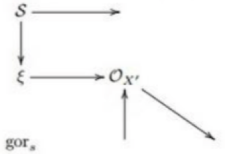
be a morphism of algebraic spaces over S and Y.

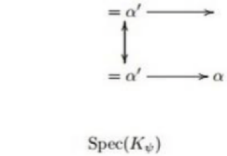
Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram





$\mathrm{Spec}(K_\psi)$ Mor_{Sets} $\mathrm{d}(\mathcal{O}_{X_{fk}}, \mathcal{G})$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type $f_\bullet$. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{s}})$$

is an isomorphism of covering of $\mathcal{O}_{X_t}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Code generation

- Code-like generation using Linux code :

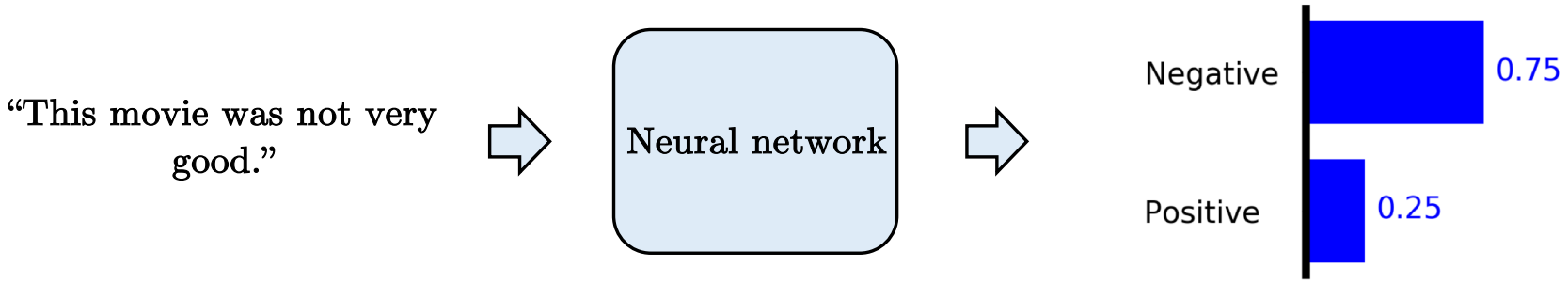
```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)

#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```

Outline

- Introduction to natural language processing
- PTB dataset
- One-hot encoding
- Vanilla recurrent neural networks
- Training VRNNs
- Word prediction
- Text generation
- **Sentiment analysis**

Sentiment analysis

- Task : Classifying a sentence/document as positive or negative.
  - Applications : Customer service review (restaurant, hotel), marketing, etc



Input:

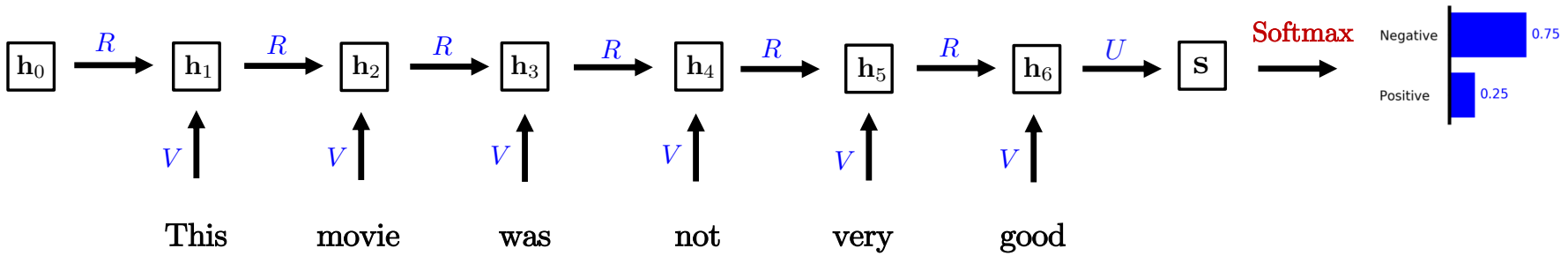
Sequence of words

Output:

Probability distribution over two classes

Sentiment analysis

- Architecture for sentiment analysis :



Questions?