

Lecture 9

Tian Han

Outline

- Long Short Term Memory (LSTM)
- Several tricks
- RNNs: generation, translation

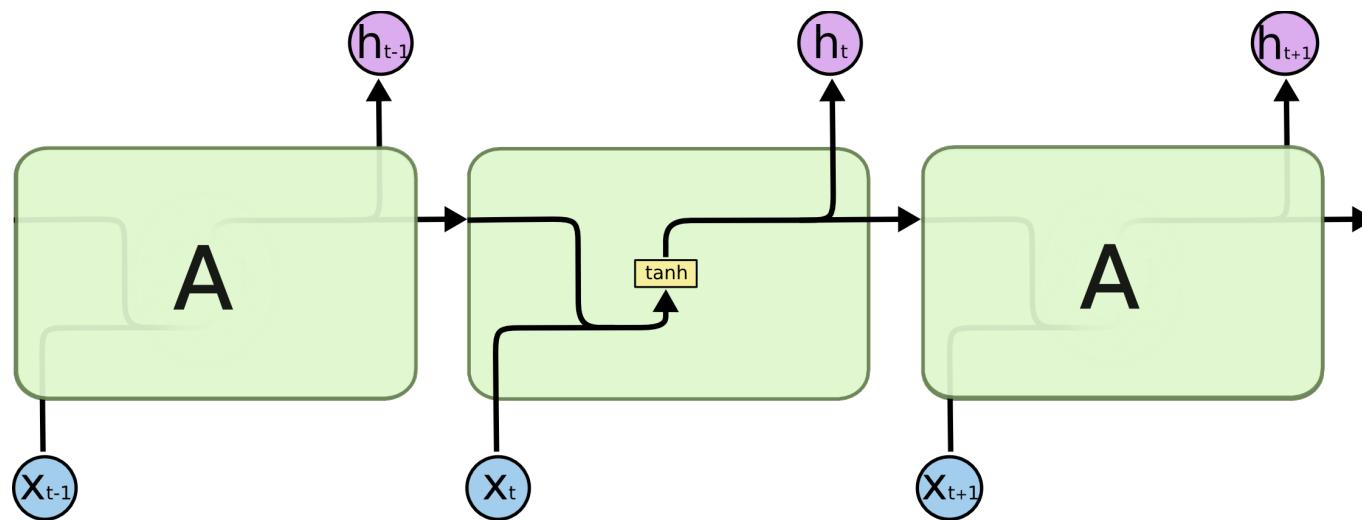
Long Short Term Memory (LSTM)

LSTM Model

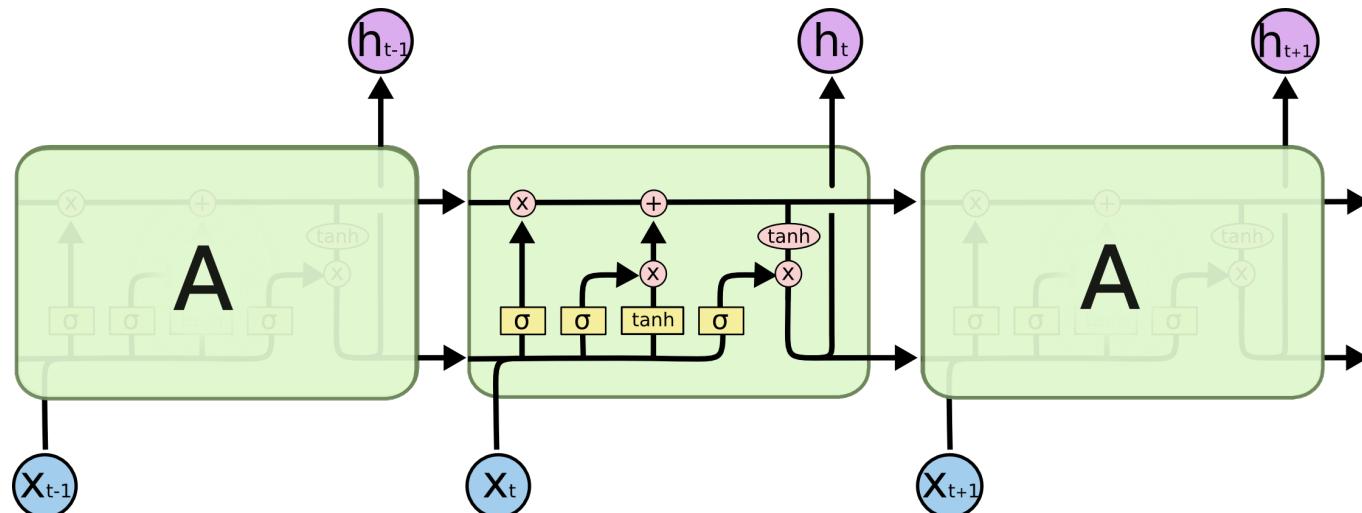
Reference

- Hochreiter and Schmidhuber. [Long short-term memory](#). *Neural computation*, 1997.

LSTM Networks



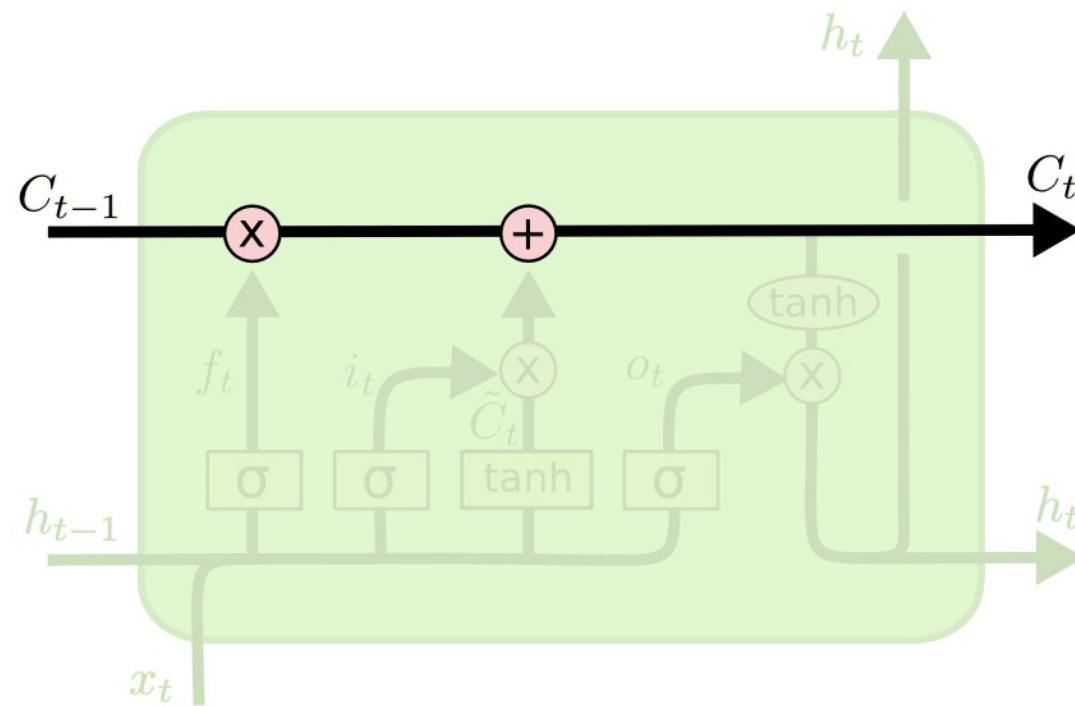
Simple RNN



LSTM

LSTM: Conveyor Belt

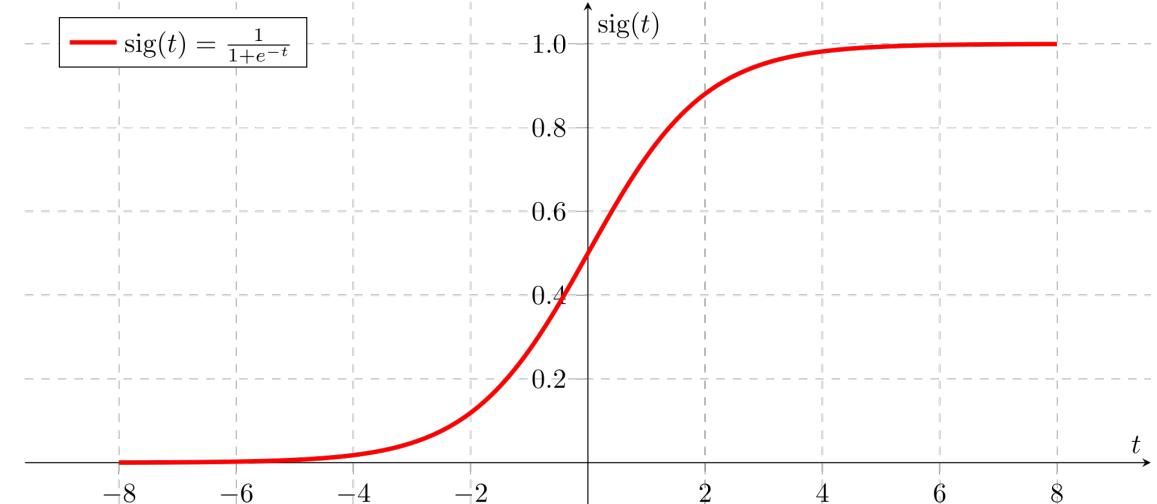
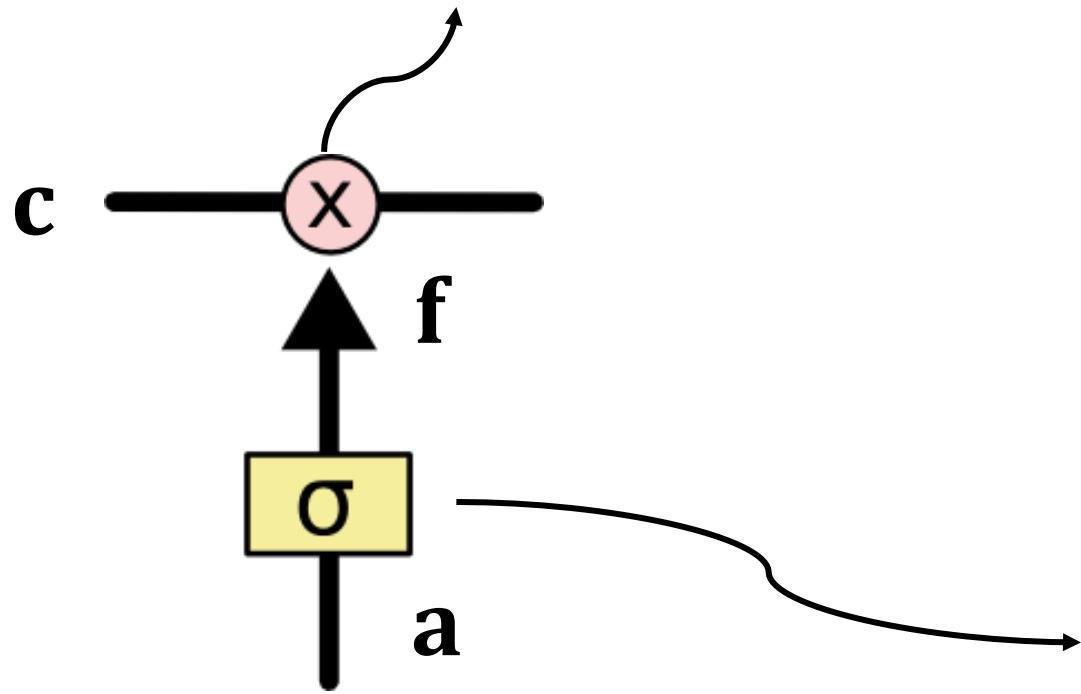
- Conveyor belt: allow past information directly flows to the future.



The Figure is from Christopher Olah's blog: Understanding LSTM Networks.

LSTM: Forget Gate

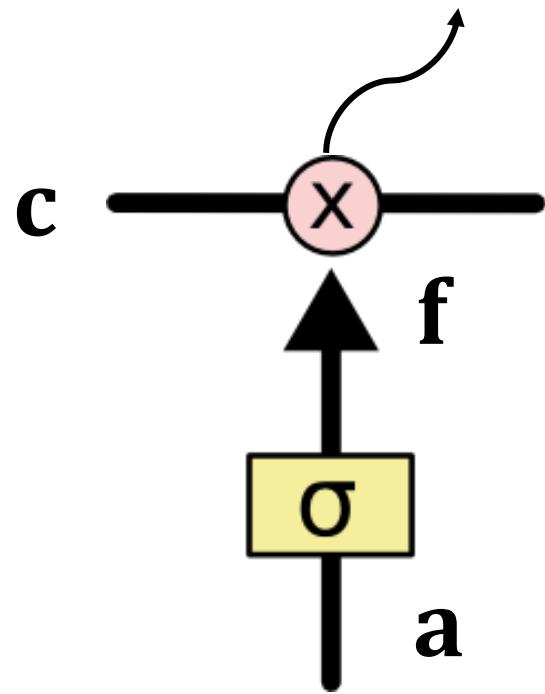
Elementwise multiplication of 2 vectors.



Sigmoid function: between 0 and 1.

LSTM: Forget Gate

Elementwise multiplication of 2 vectors.

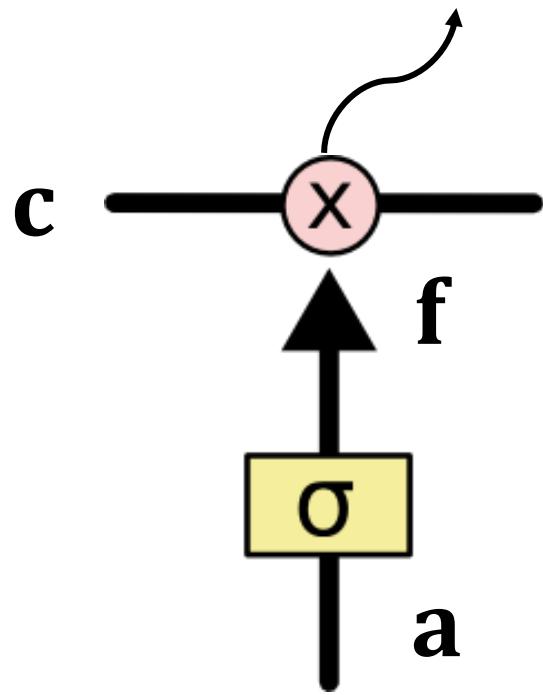


$$\sigma \begin{pmatrix} 1 \\ 3 \\ 0 \\ -2 \end{pmatrix} = \begin{bmatrix} 0.73 \\ 0.95 \\ 0.5 \\ 0.12 \end{bmatrix}$$

The diagram shows the sigmoid function σ applied elementwise to a vector a (represented by the first four entries of the matrix) to produce a vector f (represented by the second matrix). Arrows indicate the mapping from a to the sigmoid function and from the sigmoid function to f .

LSTM: Forget Gate

Elementwise multiplication of 2 vectors.



$$\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$$

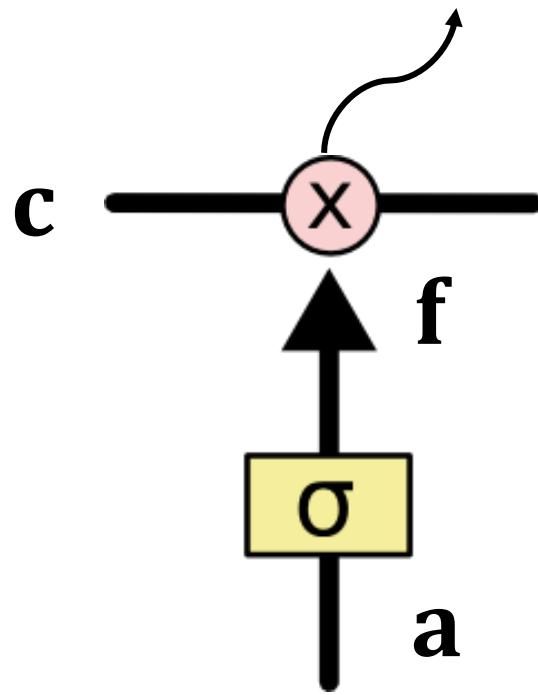
Diagram illustrating the elementwise multiplication of two vectors:

- Input vector c : $\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix}$
- Input vector f : $\begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix}$
- Output vector: $\begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$, labeled as "output"

LSTM: Forget Gate

- Forget gate (f): a vector (the same shape as c and h).
 - A value of **zero** means “let **nothing** through”.
 - A value of **one** means “let **everything** through!”

Elementwise multiplication of 2 vectors.



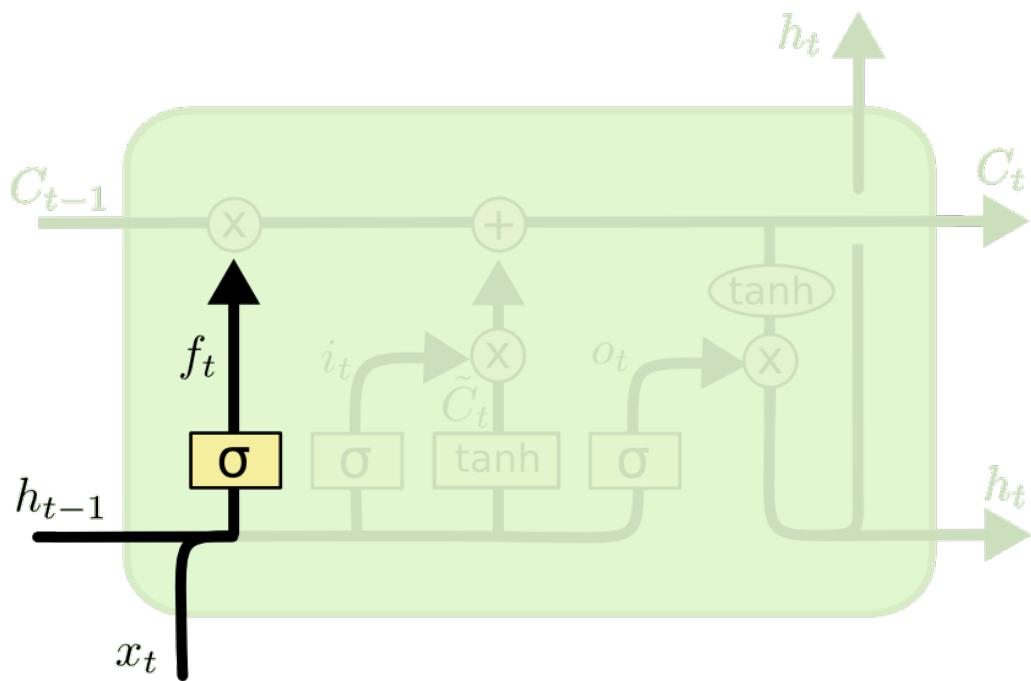
$$\begin{bmatrix} 0.9 \\ 0.2 \\ -0.5 \\ -0.1 \end{bmatrix} \circ \begin{bmatrix} 0.5 \\ 0 \\ 1 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0 \\ -0.5 \\ -0.08 \end{bmatrix}$$

Diagram illustrating the elementwise multiplication of two vectors:

- c**: Input vector
- f**: Forget gate vector
- output**: Result of elementwise multiplication

LSTM: Forget Gate

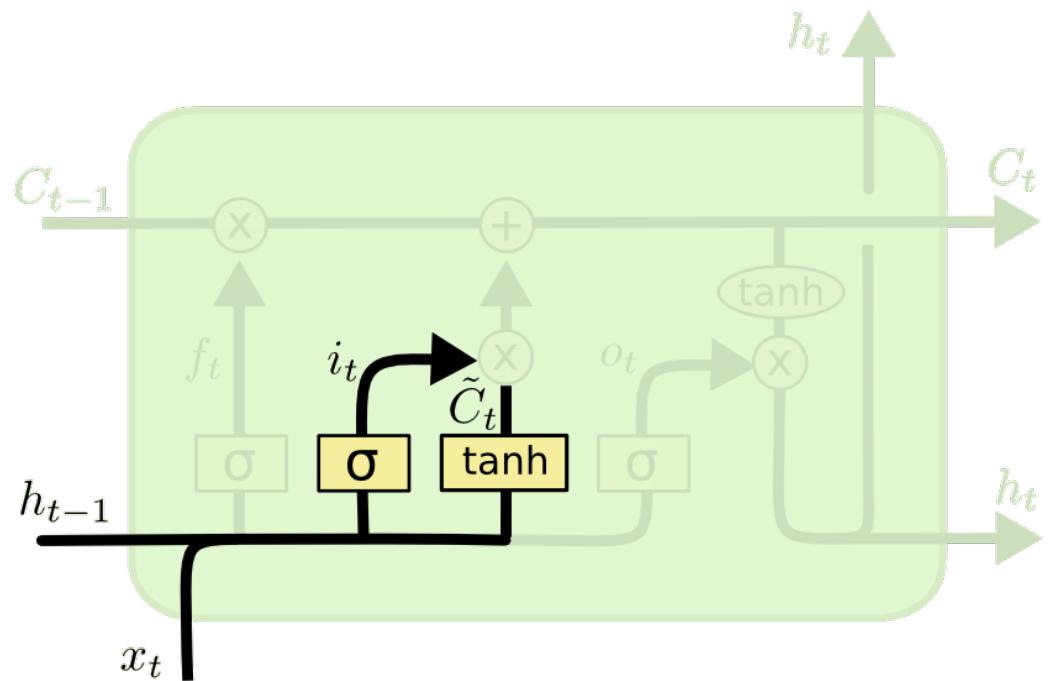
- Forget gate (f): a vector (the same shape as \mathbf{c} and \mathbf{h}).
 - A value of zero means “let nothing through”.
 - A value of one means “let everything through!”



$$f_t = \sigma \begin{bmatrix} \text{purple grid} & \text{blue grid} \end{bmatrix} \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$$

LSTM: Input Gate

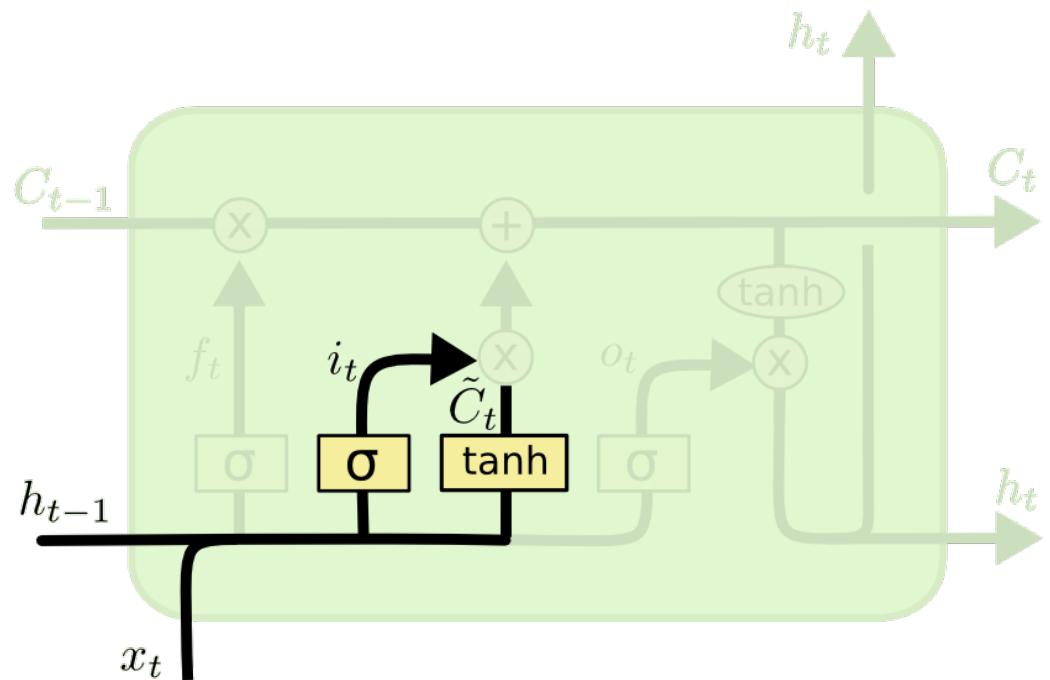
- Input gate (i_t): decides which values of the conveyor belt we'll update.



$$i_t = \sigma \left[\begin{array}{c|c} \text{purple} & \text{blue} \\ \hline \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \end{array} \right] \cdot \left[\begin{array}{c} h_{t-1} \\ x_t \end{array} \right]$$

LSTM: New Value

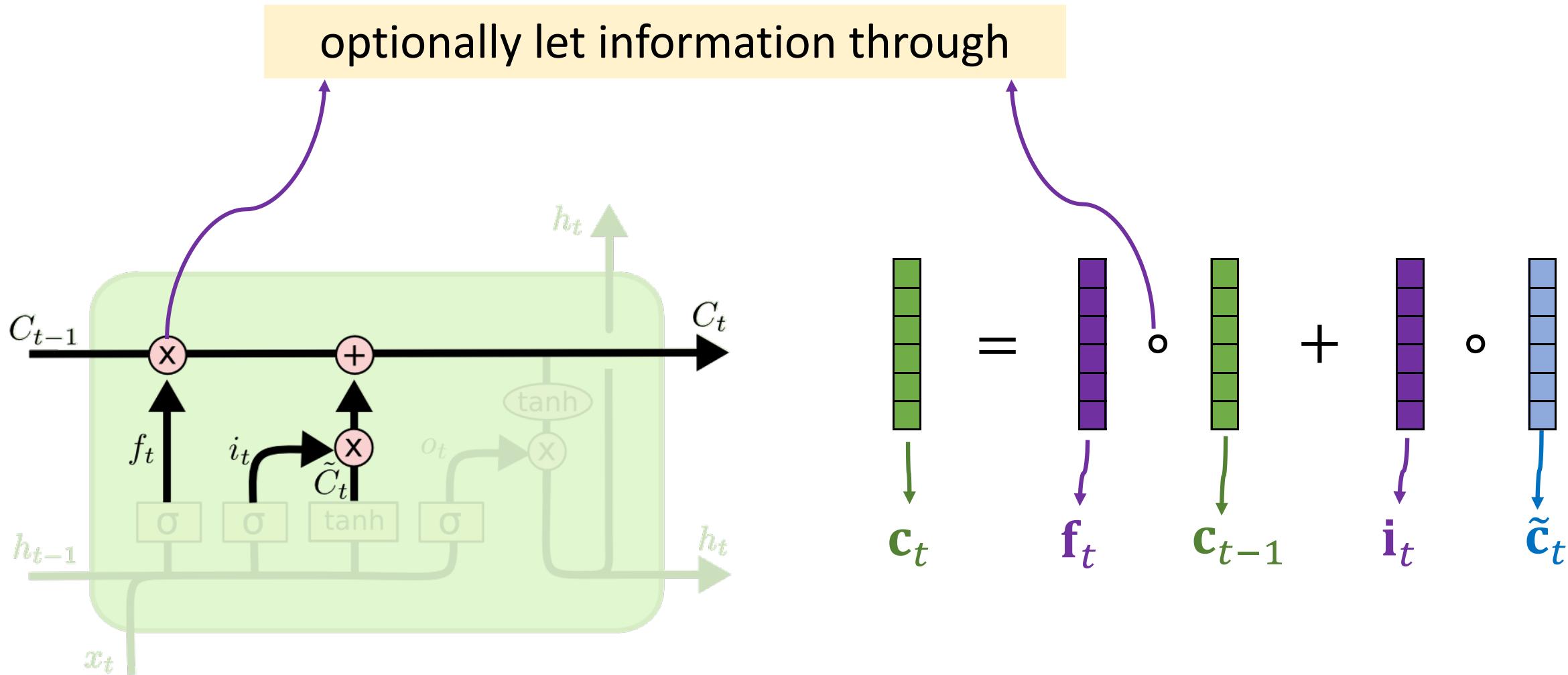
- New value (\tilde{c}_t): to be added to the conveyor belt.



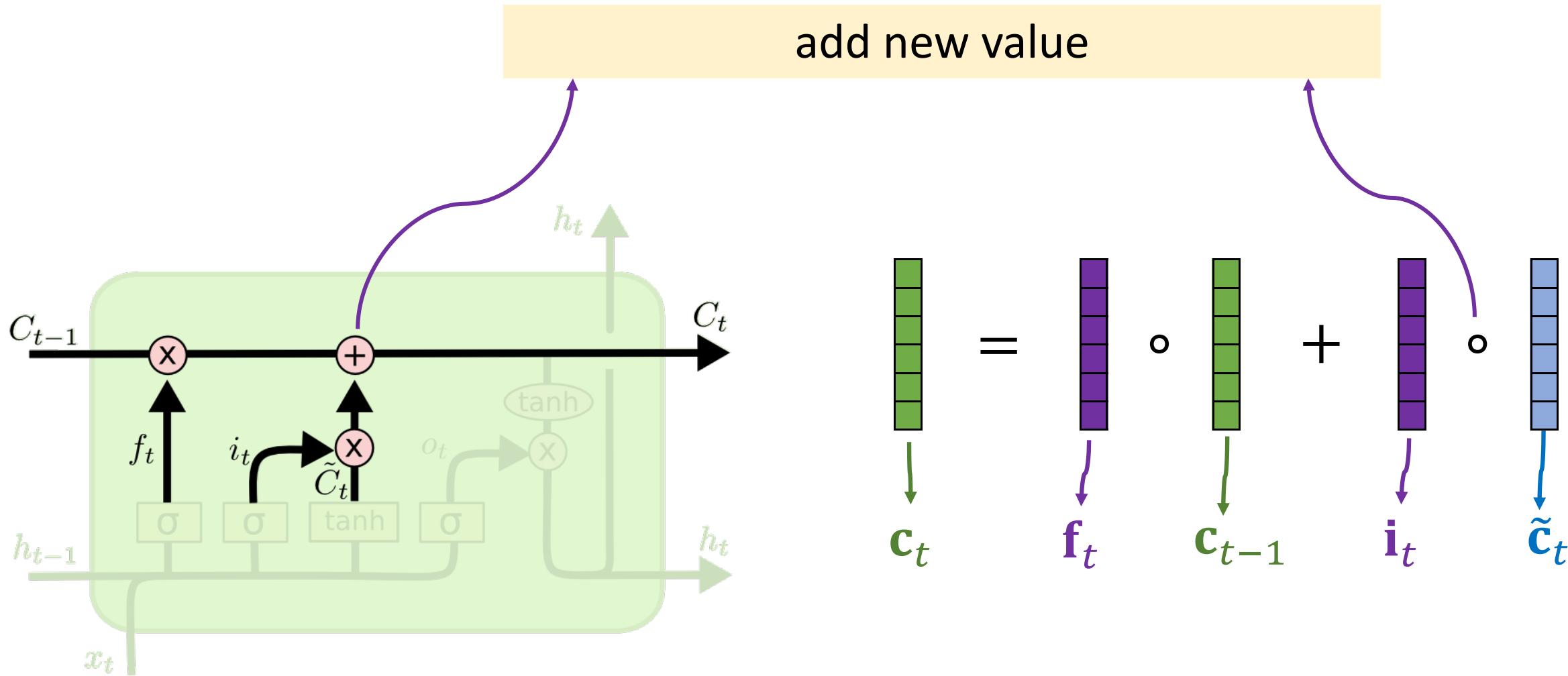
$$\tilde{c}_t = \tanh \left[\begin{array}{c|c} \text{purple} & \text{blue} \\ \hline \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \\ \text{purple} & \text{blue} \end{array} \right] \cdot \mathbf{W}_c$$

The diagram shows a matrix multiplication operation where a vertical vector of purple and blue squares is multiplied by a weight matrix \mathbf{W}_c . The result is a horizontal vector consisting of a purple segment followed by a blue segment. This vector is labeled \tilde{c}_t . To the right of the multiplication, there is a dot symbol, followed by a vertical vector of purple and blue squares labeled \mathbf{h}_{t-1} , and another dot symbol followed by a vertical vector of blue squares labeled \mathbf{x}_t .

LSTM: Update the Conveyor Belt

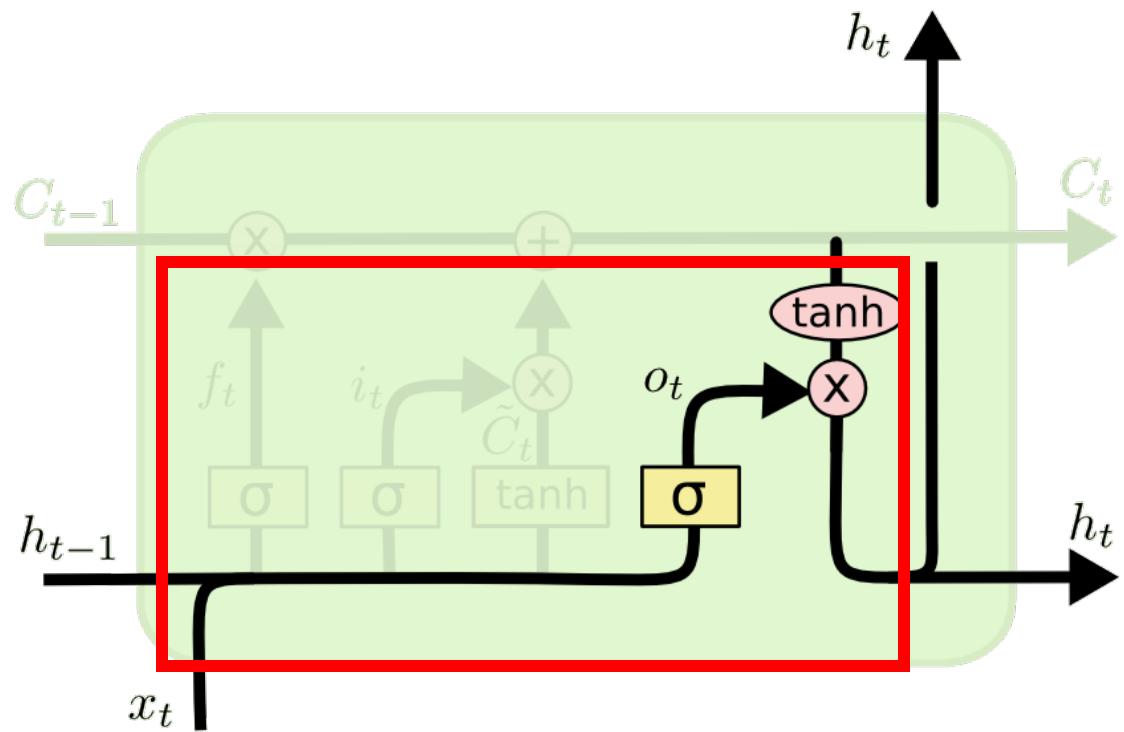


LSTM: Update the Conveyor Belt



LSTM: Output Gate

- Output gate (\mathbf{o}_t): decide what flows from the conveyor belt C_t to the state h_t .

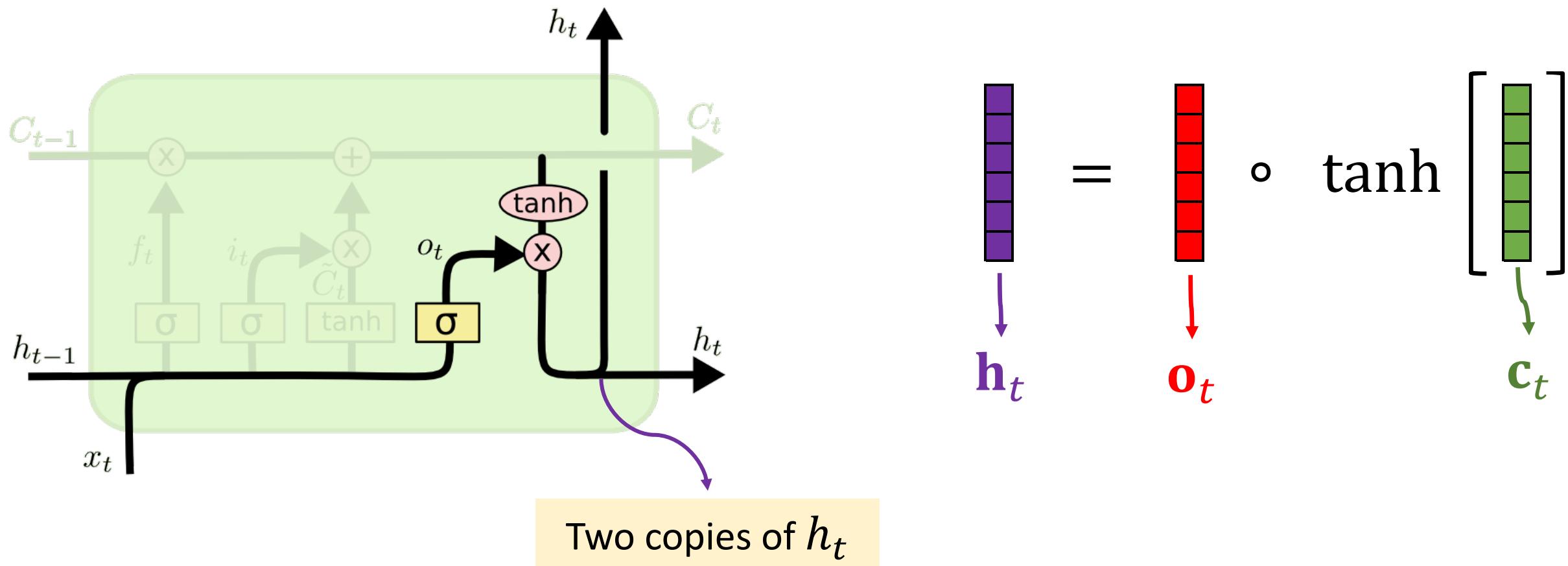


$$\mathbf{o}_t = \sigma \begin{bmatrix} \text{red vertical vector} \\ \text{purple grid} \\ \text{blue grid} \end{bmatrix} \cdot \mathbf{W}_o$$

The diagram shows the mathematical representation of the output gate. The output \mathbf{o}_t is calculated as the sigmoid function (σ) applied to a weighted sum of three components: a red vertical vector, a purple grid, and a blue grid. These weights are represented by the matrix \mathbf{W}_o .

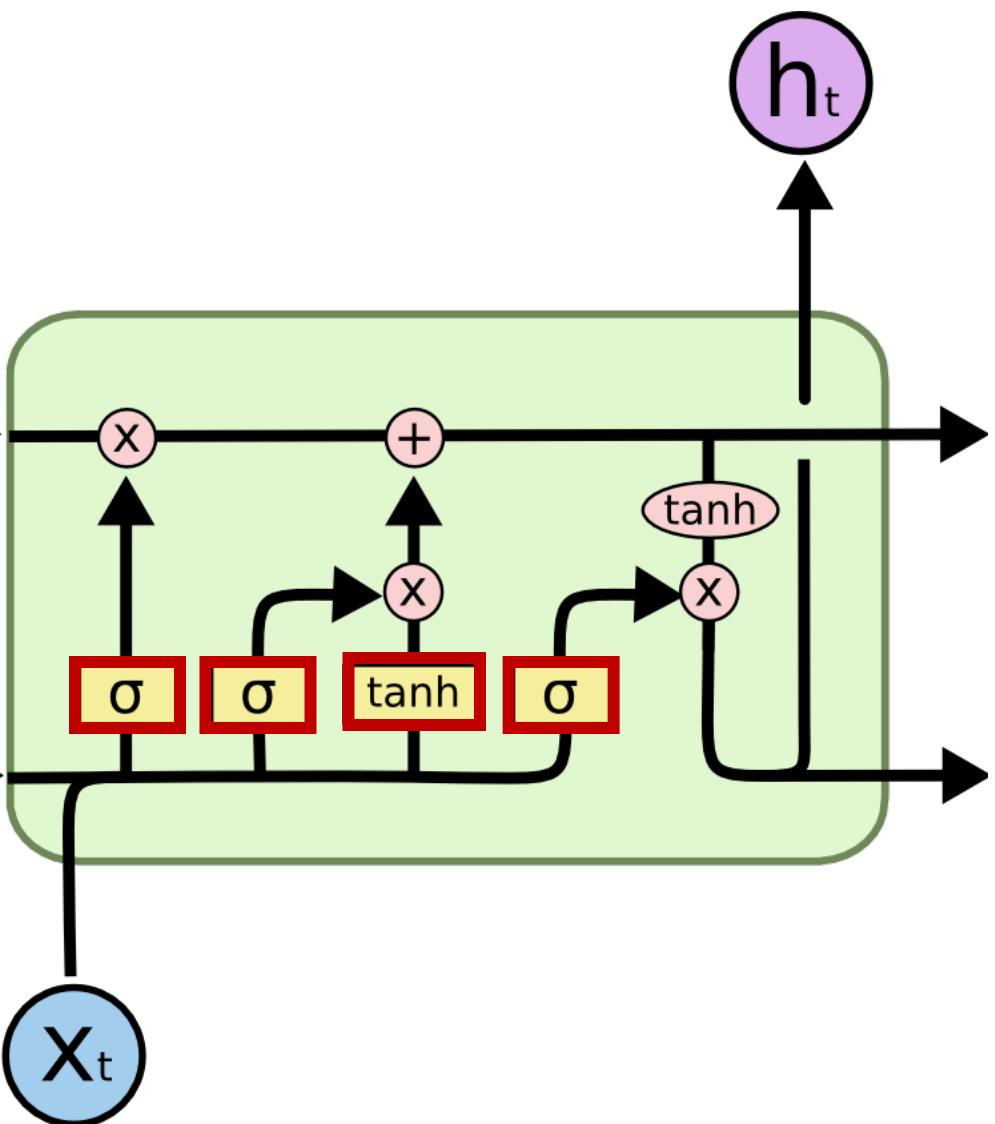
LSTM: Update State

- State (\mathbf{h}_t): the output of LSTM.



The left figure is from Christopher Olah's blog: Understanding LSTM Networks.

LSTM: Number of Parameters

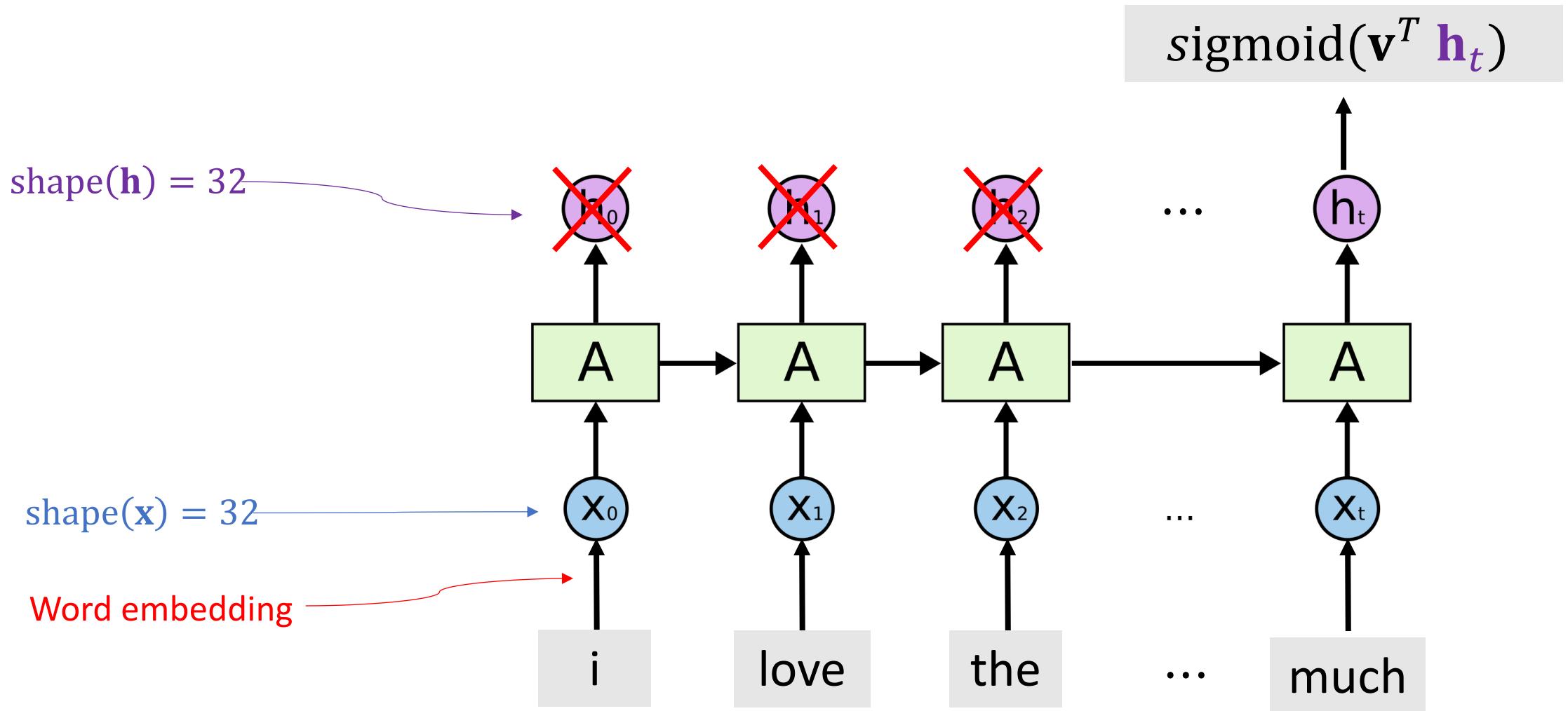


#parameters: **4 times as many as SimpleRNN**

- **4** parameter matrices, each of which has
 - #rows: shape (h)
 - #cols: shape (h) +shape (x)
- #parameter (do not count intercept):
$$4 \times \text{shape}(\mathbf{h}) \times [\text{shape}(\mathbf{h}) + \text{shape}(\mathbf{x})]$$

LSTM Using Keras

LSTM for IMDB Review



LSTM for IMDB Review

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Flatten

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(LSTM(state_dim, return_sequences=False))
model.add(Dense(1, activation='sigmoid'))

Replace "SimpleRNN" by "LSTM".
model.summary()
```

LSTM for IMDB Review

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, 500, 32)	320000
lstm_1 (LSTM)	(None, 32)	8320
dense_1 (Dense)	(None, 1)	33
=====		
Total params:	328,353	
Trainable params:	328,353	
Non-trainable params:	0	

LSTM for IMDB Review

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
lstm_1 (LSTM)	(None, 32)	8320
dense_1 (Dense)	(None, 1)	33
Total params: 328,353	#parameters in LSTM:	
Trainable params: 328,353	• $8320 = 2080 \times 4$	
Non-trainable params: 0	• $2080 = 32 \times (32 + 32) + 32$	

shape(h) = 32

shape(x)

LSTM for IMDB Review

- Training Accuracy: 91.8%
- Validation Accuracy: 88.7%
- Test Accuracy: 88.6%

Substantial improvement over SimpleRNN (whose test accuracy is 84%).

LSTM Dropout

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Flatten

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(LSTM(state_dim, return_sequences=False, dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Summary

- LSTM uses a “conveyor belt” to get longer memory than SimpleRNN.

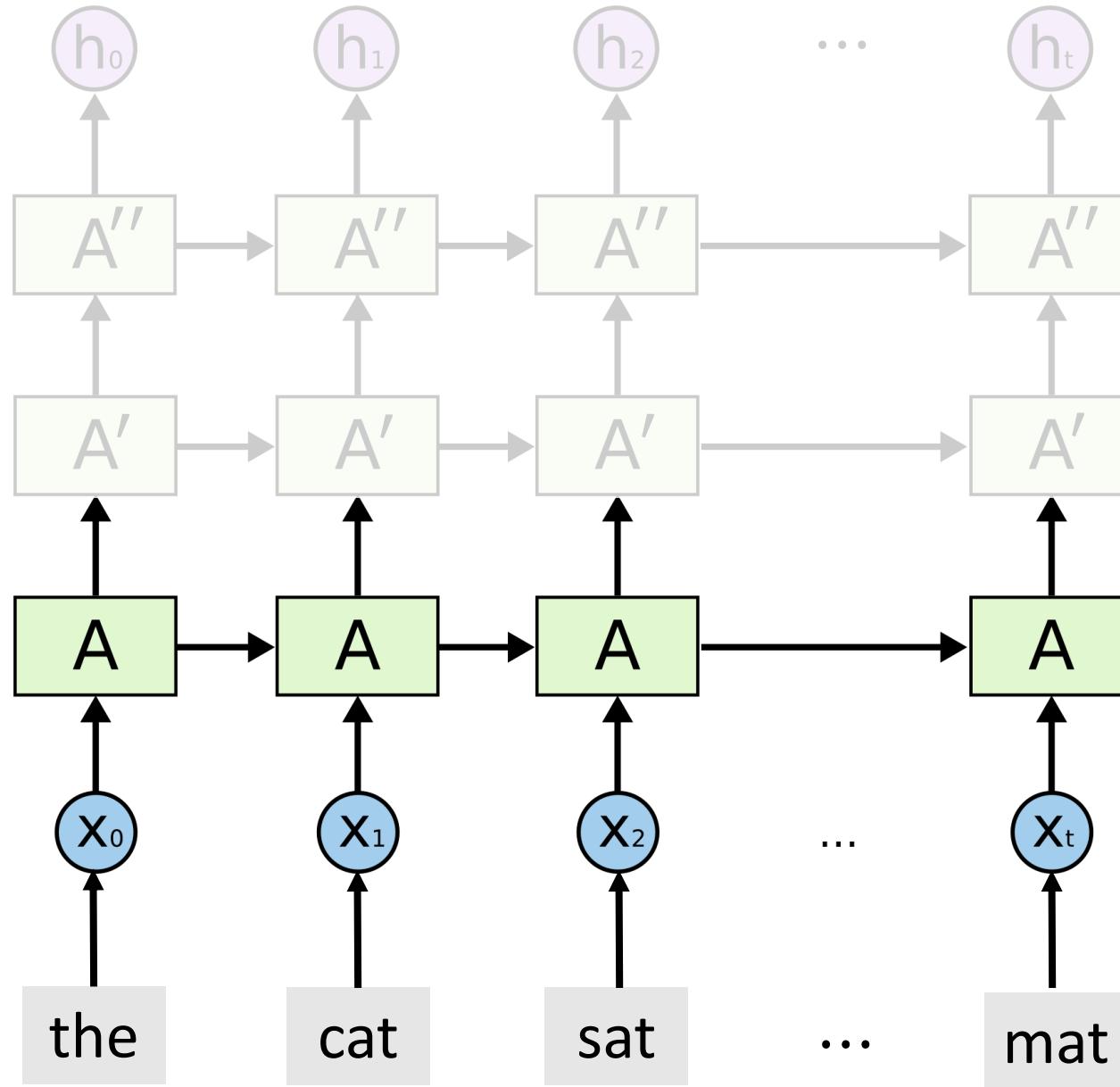
Summary

- LSTM uses a “conveyor belt” to get longer memory than SimpleRNN.
- Each of the following blocks has a parameter matrix:
 - Forget gate.
 - Input gate.
 - New values.
 - Output gate.
- Number of parameters:
$$4 \times \text{shape}(\mathbf{h}) \times [\text{shape}(\mathbf{h}) + \text{shape}(\mathbf{x})].$$

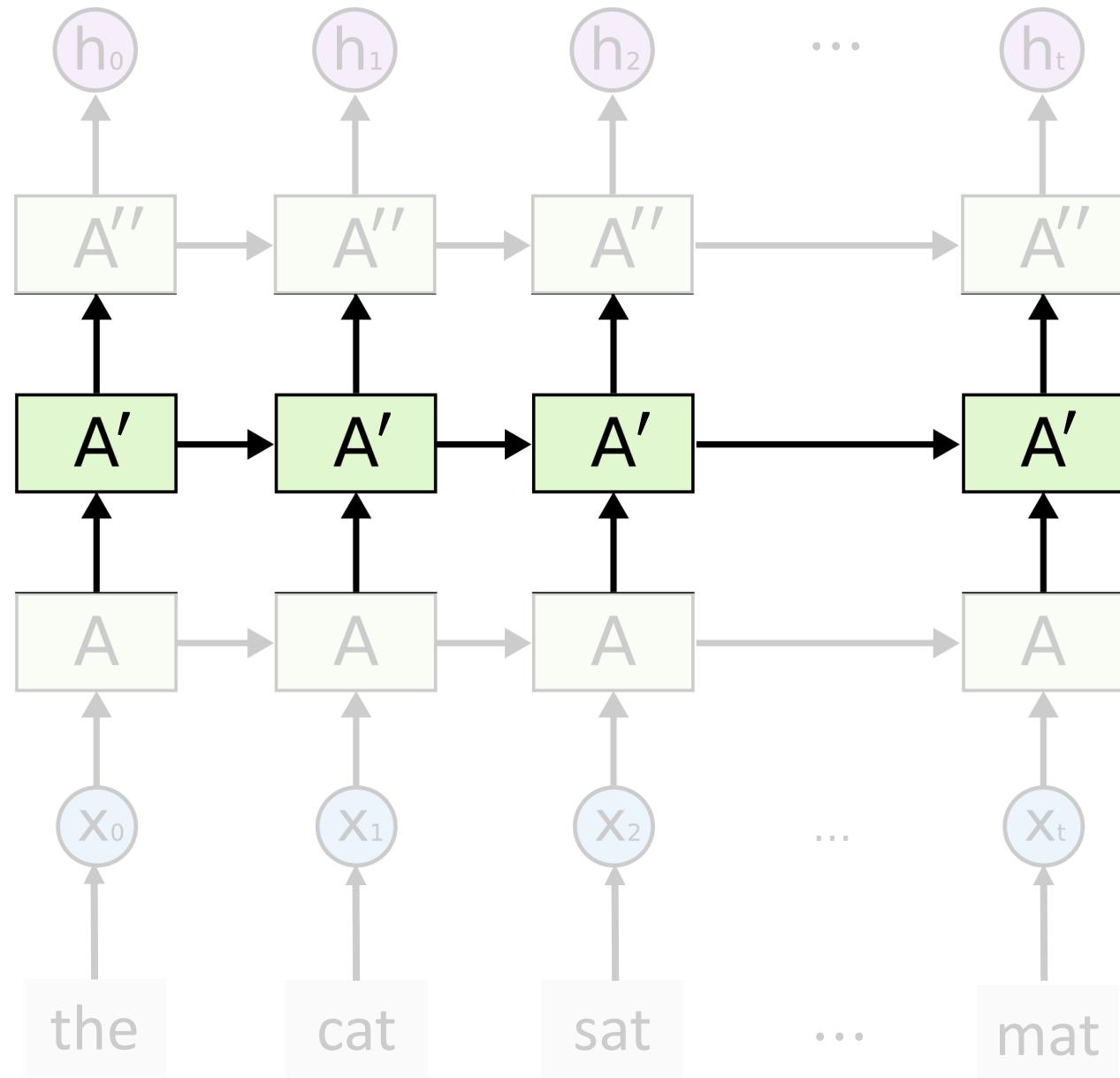
Making RNNs More Effective

Stacked RNN

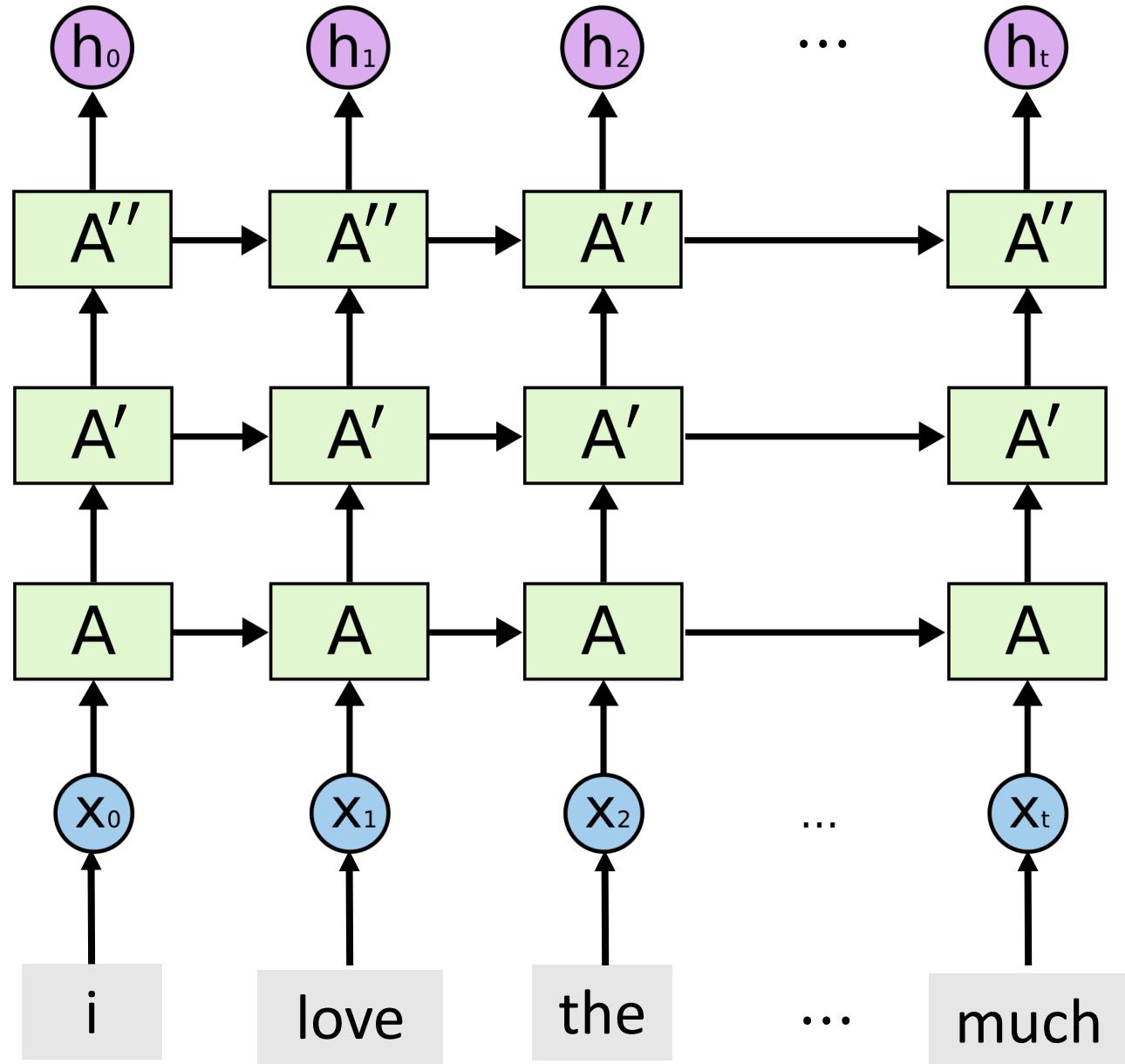
Stacked RNN



Stacked RNN



Stacked RNN



Stacked LSTM

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(LSTM(state_dim, return_sequences=True, dropout=0.2))
model.add(LSTM(state_dim, return_sequences=True, dropout=0.2))
model.add(LSTM(state_dim, return_sequences=False, dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

Stacked LSTM

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, 500, 32)	320000
lstm_1 (LSTM)	(None, 500, 32)	8320
lstm_2 (LSTM)	(None, 500, 32)	8320
lstm_3 (LSTM)	(None, 32)	8320
dense_1 (Dense)	(None, 1)	33
=====		

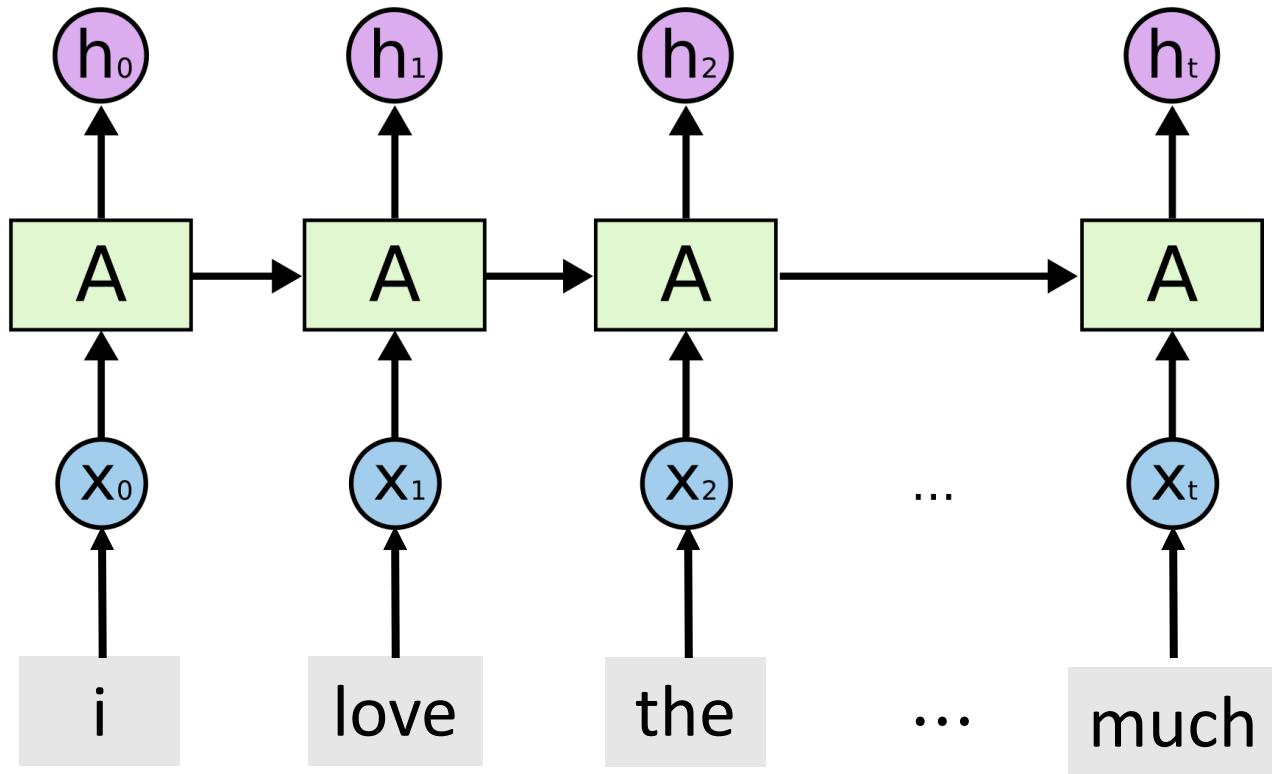
Total params: 344,993

Trainable params: 344,993

Non-trainable params: 0

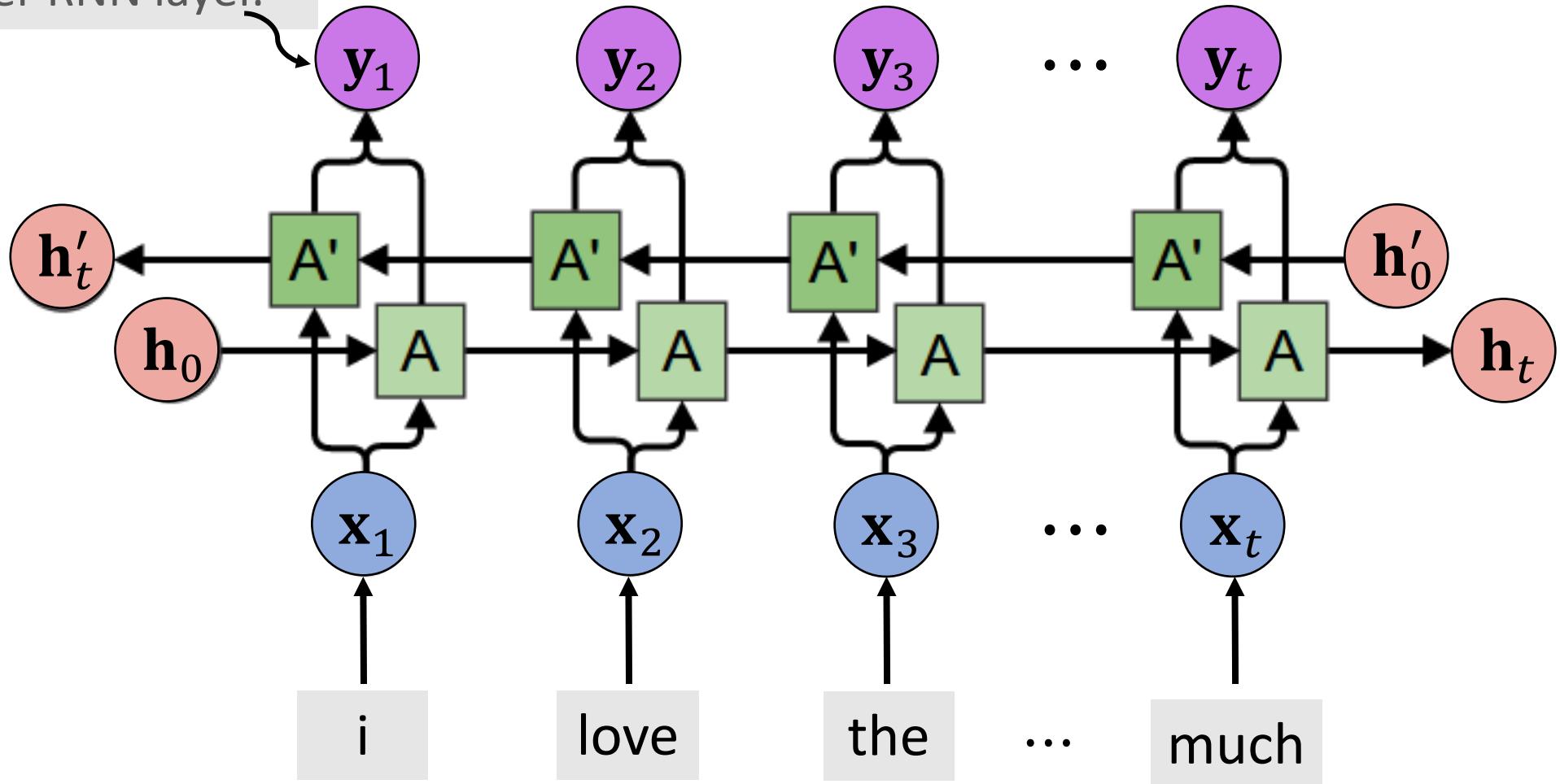
Bidirectional RNN

Standard RNN



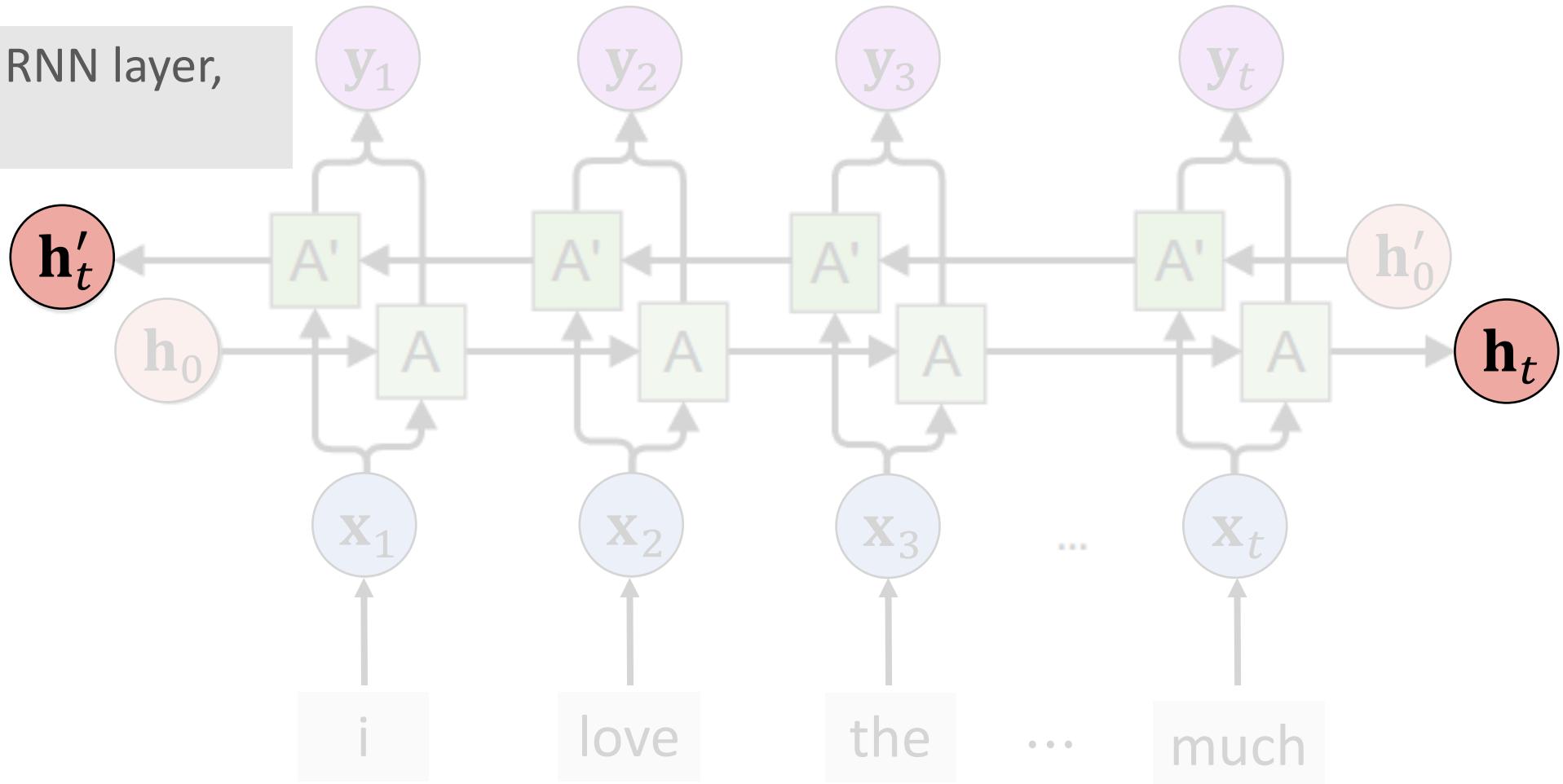
Bidirectional RNN

- Stack of 2 states.
- Passed to the upper RNN layer.



Bidirectional RNN

If there is no upper RNN layer,
then return $[h_t, h'_t]$



Bi-LSTM

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Bidirectional

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(Bidirectional(LSTM(state_dim, return_sequences=False, dropout=0.2)))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Bi-LSTM

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
bidirectional_1 (Bidirection)	(None, 64)	16640
dense_1 (Dense)	(None, 1)	65

Total params: 336,705

Trainable params: 336,705

Non-trainable params: 0

Pretrain

Why pretraining?

Observation: The **embedding layer** contributes most of the parameters!

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
bidirectional_1 (Bidirection)	(None, 64)	16640
dense_1 (Dense)	(None, 1)	65
<hr/>		

Total params: 336,705

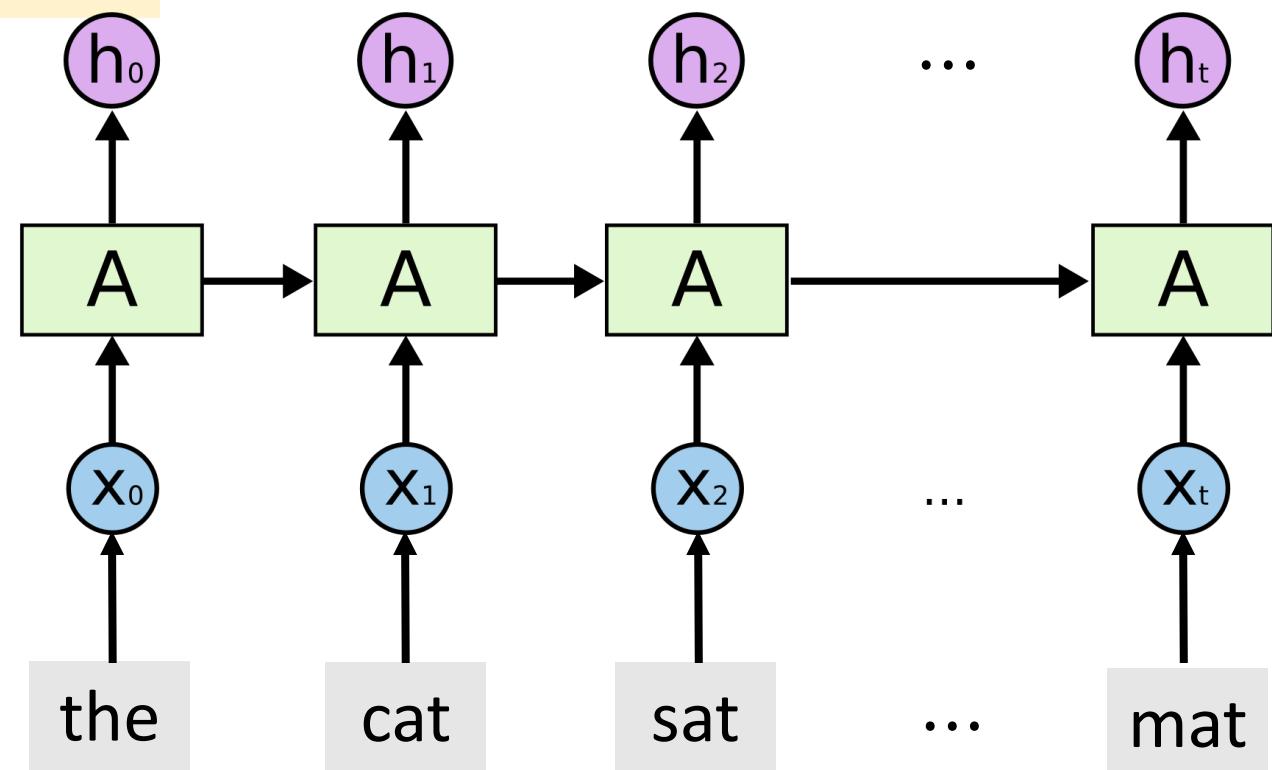
Trainable params: 336,705

Non-trainable params: 0

Pretrain the Embedding Layer

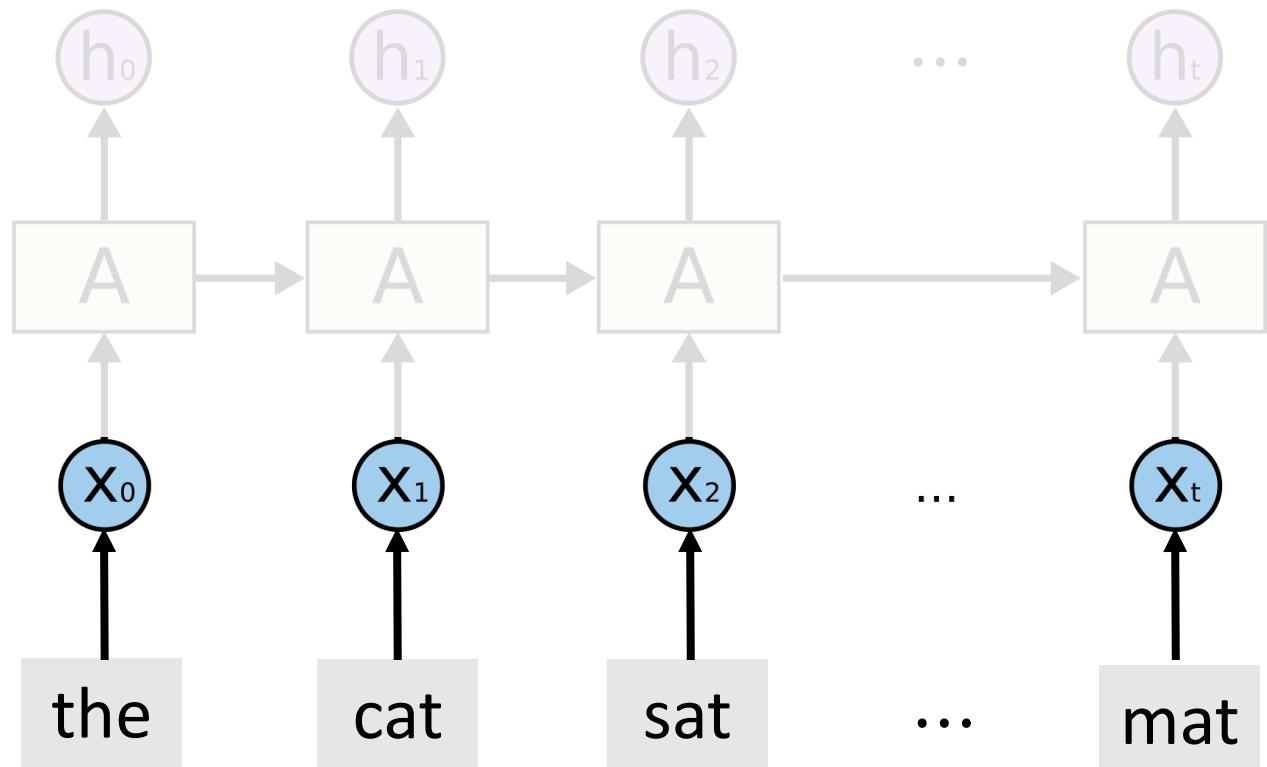
Step 1: Train a model on large dataset.

- Perhaps different problem.
- Perhaps different model.



Pretrain the Embedding Layer

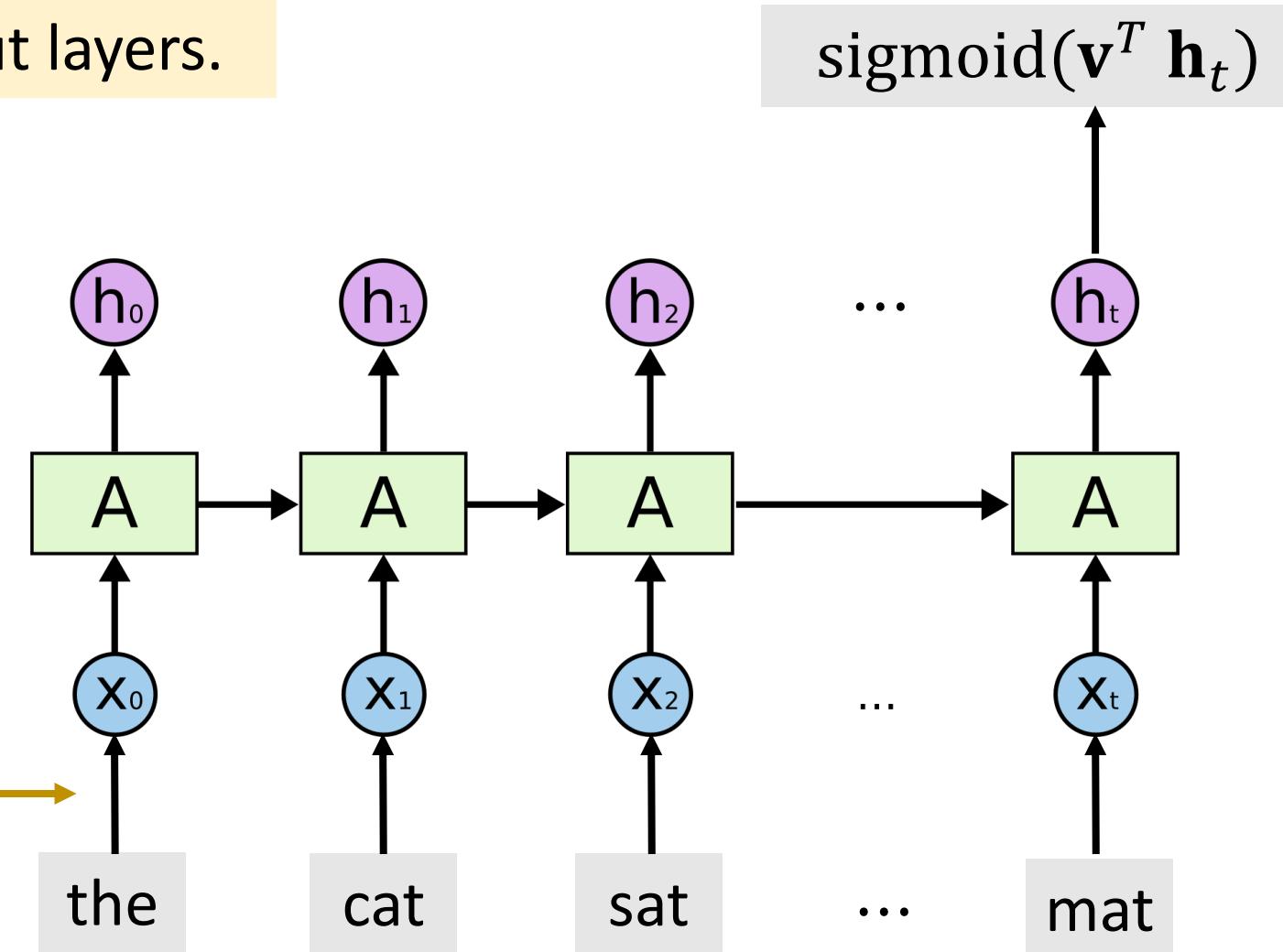
Step 2: Keep only the embedding layer.



Pretrain the Embedding Layer

Step 3: Train new LSTM and output layers.

Set the embedding layer **non-trainable**.



Summary

Summary

- SimpleRNN and LSTM are two kinds of RNNs; always use **LSTM** instead of **SimpleRNN**.
- Use **Bi-RNN** instead of RNN whenever possible.
- **Stacked RNN** may be better than a single RNN layer (if n is big).
- **Pretrain** the embedding layer (if n is small).

Text Generation

Main Idea

RNN for Text Prediction

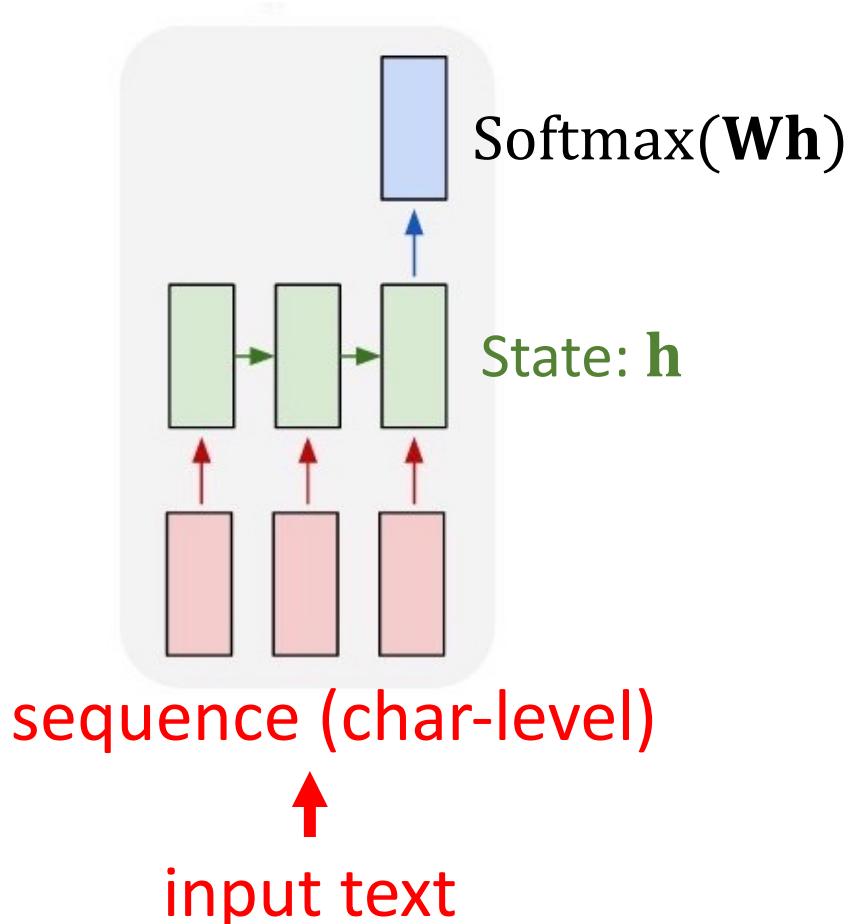
Example

- Input text: “the cat sat on the ma”
- Question: what is the next char?

RNN for Text Prediction

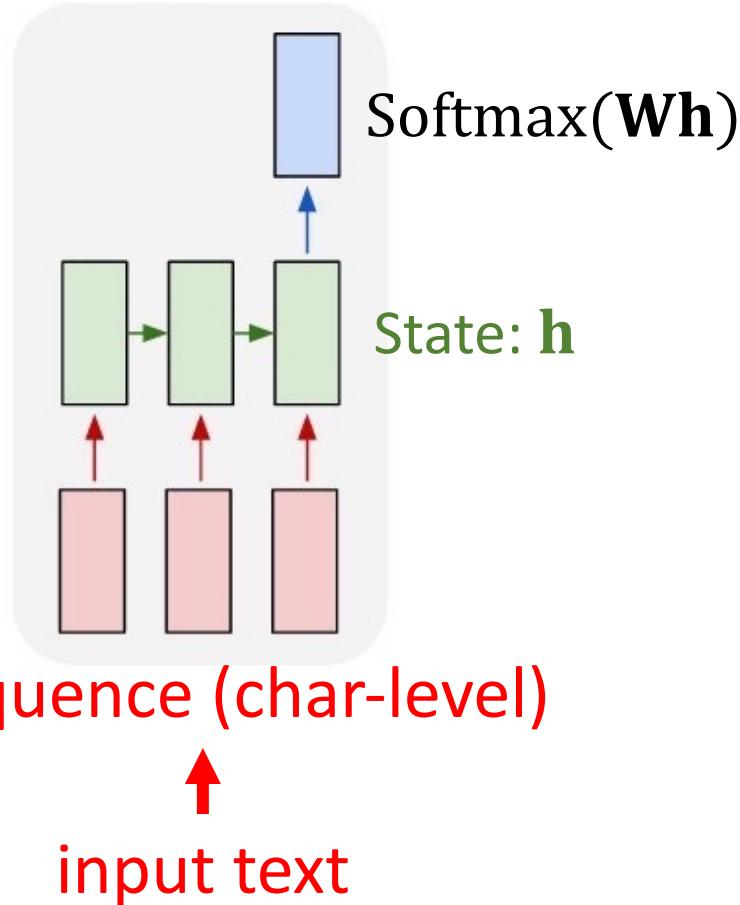
Example

- Input text: “the cat sat on the ma”
- Question: what is the next char?



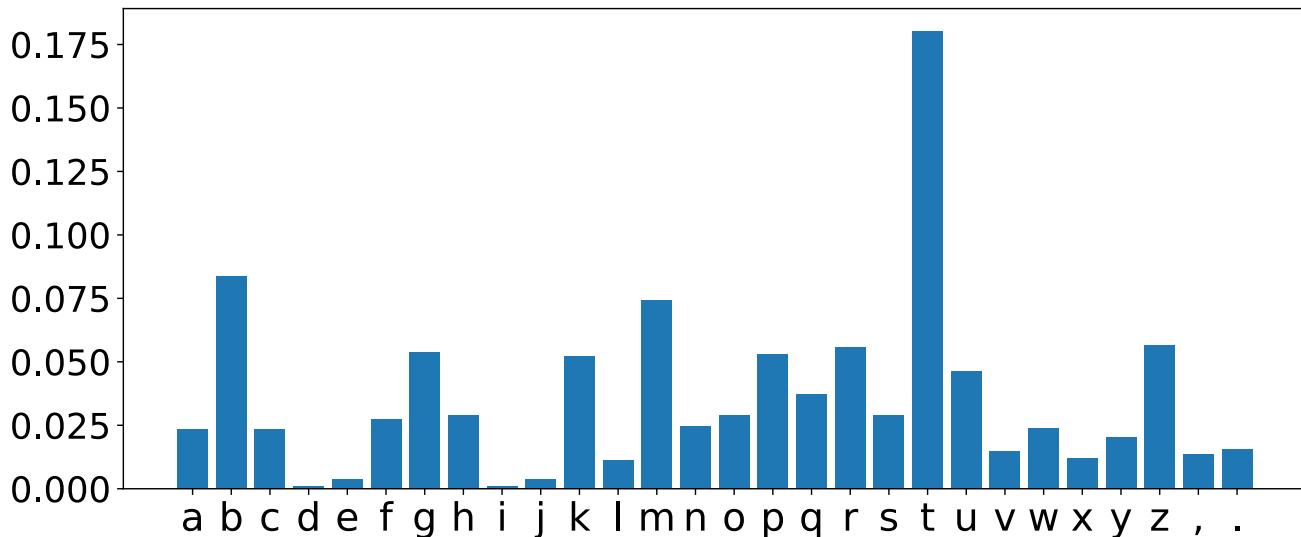
RNN for Text Prediction

predict the next char



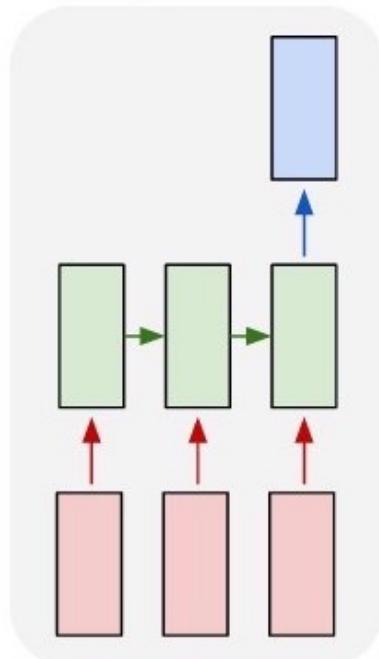
Example

- Input text: “the cat sat on the ma”
- Question: what is the next char?
- RNN outputs a distribution over the chars.



RNN for Text Prediction

predict the next char



sequence (char-level)



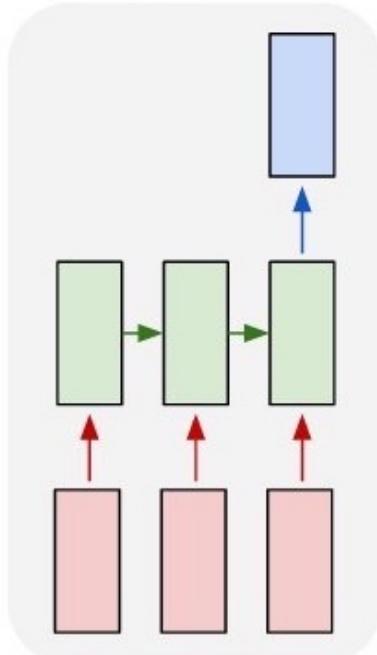
input text

Example

- Input text: “the cat sat on the ma”
- Question: what is the next char?
 - RNN outputs a distribution over the chars.
 - Sample a char from it; we may get ‘t’.
 - Take “the cat sat on the mat” as input.
 - Maybe the next char is period ‘.’.

Train an RNN for Text Prediction

predict the next char



sequence (char-level)



input text

How do we train such an RNN?

- Cut text to segments (with overlap).
 - E.g., `seg_len=40` and `stride=3`.

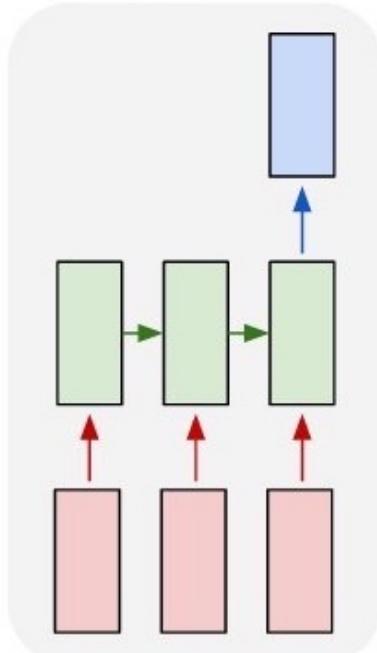
Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...

...

Train an RNN for Text Prediction

predict the next char



sequence (char-level)



input text

How do we train such an RNN?

- Cut text to segments (with overlap).
 - E.g., `seg_len=40` and `stride=3`.

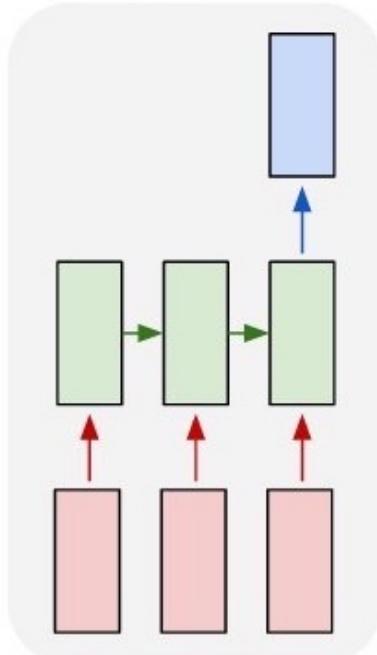
Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...

...

Train an RNN for Text Prediction

predict the next char



sequence (char-level)



input text

How do we train such an RNN?

- Cut text to segments (with overlap).
 - E.g., `seg_len=40` and `stride=3`.

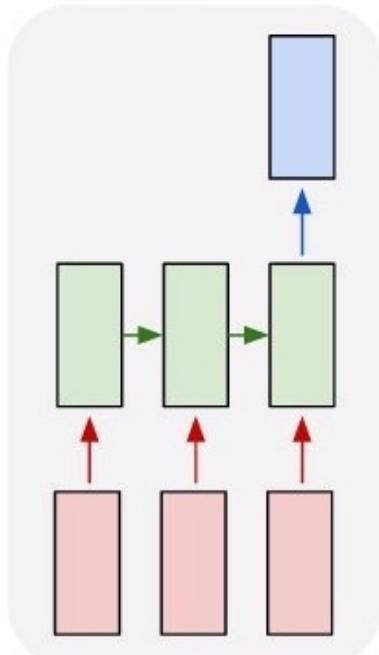
Machine **learning** is a subset of **artificial intelligence** in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...

...

Train an RNN for Text Prediction

predict the next char



sequence (char-level)



input text

How do we train such an RNN?

- Cut text to segments (with overlap).
 - E.g., `seg_len=40` and `stride=3`.

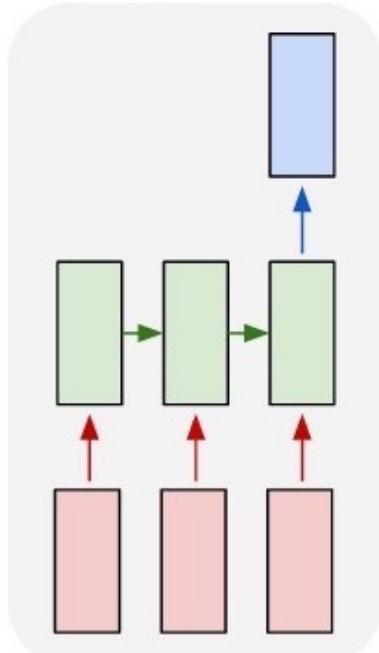
Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...

...

Train an RNN for Text Prediction

predict the next char



sequence (char-level)



input text

How do we train such an RNN?

- Cut text to segments (with overlap).
- A **segment** is used as input text.
- Its **next char** is used as label.
- Training data: (**segment**, **next_char**) pairs

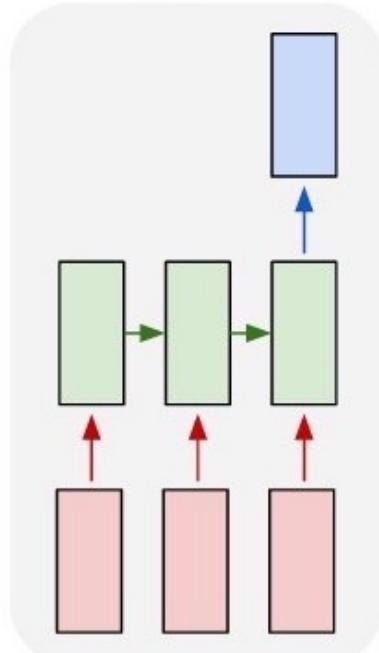
Machine **learning is a subset of artificial intelligence** in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. The name machine learning was coined in 1959 by Arthur Samuel.

...

...

Train an RNN for Text Prediction

predict the next char

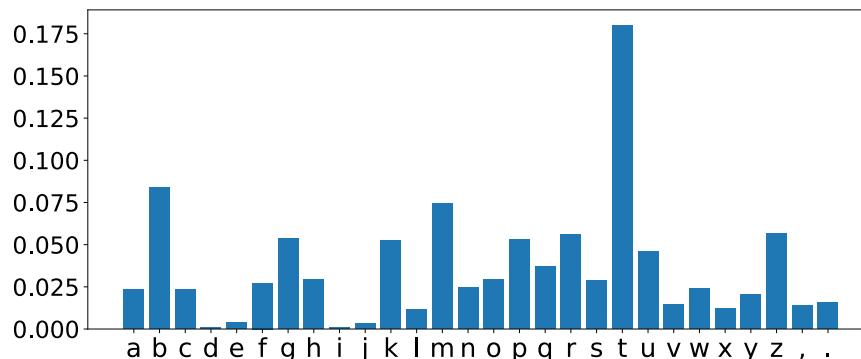


sequence (char-level)

input text

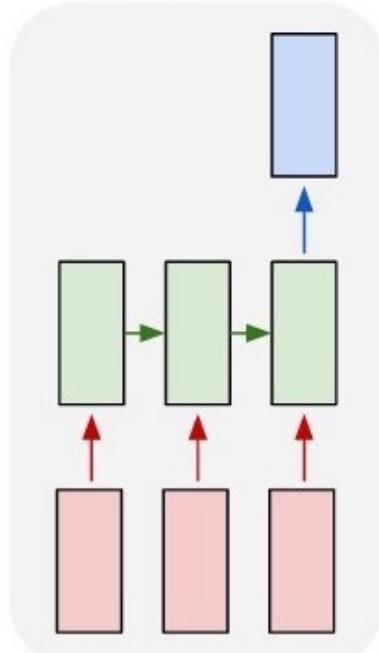
How do we train such an RNN?

- Cut text to segments (with overlap).
- A **segment** is used as input text.
- Its next char is used as label.
- Training data: (**segment**, **next_char**) pairs
- It is a multi-class classification problem.
 - #class = #unique chars.



Train an RNN for Text Prediction

predict the next char



sequence (char-level)



input text

How do we train such an RNN?

- Cut text to segments (with overlap).
- A **segment** is used as input text.
- Its next char is used as label.
- Training data: (**segment**, **next_char**) pairs
- It is a multi-class classification problem.
 - #class = #unique chars.

If the RNN is trained on Shakespeare's books, then the generated text is Shakespeare's style.

Fun with Text Generation

Generate baby names (trained on 8000 baby names).

Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacrie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa Wallon Martine Merus Jelen Candica Wallin Tel Rachene Tarine Ozila Ketia Shanne Arnande Karella Roselina Alessia Chasty Deland Berther Geamar Jackein Mellisand Sagdy Nenc Lessie Rasemy Guen Gavi Milea Anneda Margoris Janin Rodelin Zeanna Elyne Janah Ferzina Susta Pey Castina

Fun with Text Generation

Generate C code (trained on Linux source code).

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```

Fun with Text Generation

Generate academic articles (LaTeX source files).

For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m,\bullet} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ???. Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{\mathcal{M}}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)^{opp}_{fppf}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ???. It may replace S by $X_{spaces, \acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ???. Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x,\dots,0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq p$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.

Lemma 0.3. In Situation ???. Hence we may assume $q' = 0$.

Proof. We will use the property we see that p is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

Training a Text Generator

1. Prepare Training Data

```
print('Text length:', len(text))
```

Text length: 600893

```
text[0:1000]
```

'preface\n\n\nsupposing that truth is a woman--what then? is there not ground\nfor suspecting that all philosophers, in so far as they have been\nDogmatists, have failed to understand women--that the terrible\nseriousness and clumsy importunity with which they have usually paid\ntheir addresses to truth, have been unskilled and unseemly methods for\nwinning a woman? certainly she has never allowed herself to be won; and\nat present every kind of dogma stands with sad and discouraged mien--if,\nindeed, it stands at all! for there are scoffers who maintain that it\nhas fallen, that all dogma lies on the ground--nay more, that it is at\nits last gasp. but to speak seriously, there are good grounds for hoping\nthat all dogmatizing in philosophy, whatever solemn, whatever conclusive\nand decided airs it has assumed, may have been only a noble puerilism\nand tyronism; and probably the time is at hand when it will be once\nand again un

1. Prepare Training Data

segment:

preface\n\n\nsupposing that truth is a woman--what then? is there
not ground\nfor suspecting that all philosophers, in so far as they
have been\ndogmatists, have failed to understand women--that the te
rrible\nseriousness and clumsy importunity with which they have usu
ally paid\nt heir addresses to truth, have been unskilled and unseem

segments[0]: preface\n\n\nsupposing that truth **next_chars[0]:** i

1. Prepare Training Data

segment:

'preface\n\n\nsupposing that truth is a woman--what then? is there
not ground\nfor suspecting that all philosophers, in so far as they
have been\ndogmatists, have failed to understand women--that the te
rrible\nseriousness and clumsy importunity with which they have usu
ally paid\ntheir addresses to truth, have been unskilled and unseem

segments[0]: preface\n\n\nsupposing that truth

segments[1]: face\n\n\nsupposing that truth is

next_chars[0]: i

next_chars[1]: a

1. Prepare Training Data

segment:

'preface\n\n\nsupposing that truth is a woman--what then? is there
not ground\nfor suspecting that all philosophers, in so far as they
have been\ndogmatists, have failed to understand women--that the te
rrible\nseriousness and clumsy importunity with which they have usu
ally paid\ntheir addresses to truth, have been unskilled and unseem

segments[0]: preface\n\n\nsupposing that truth

next_chars[0]: i

segments[1]: face\n\n\nsupposing that truth is

next_chars[1]: a

segments[2]: e\n\n\nsupposing that truth is a w

next_chars[2]: o

1. Prepare Training Data

segment:

'preface\n\n\n\nsupposing that truth is a woman--what then? is there
not ground\nnfor suspecting that all philosophers, in so far as they
have been\nndogmatists, have failed to understand women--that the te
rrible\nnseriousness and clumsy importunity with which they have usu
ally paid\nntheir addresses to truth, have been unskilled and unseem

segments[0]: preface\n\n\n\nsupposing that truth

segments[1]: face\n\n\n\nsupposing that truth is

segments[2]: e\n\n\n\nsupposing that truth is a w

segments[3]: \n\nsupposing that truth is a woma

⋮

next_chars[0]: i

next_chars[1]: a

next_chars[2]: o

next_chars[3]: n

⋮

2. Character to Vector

Dictionary

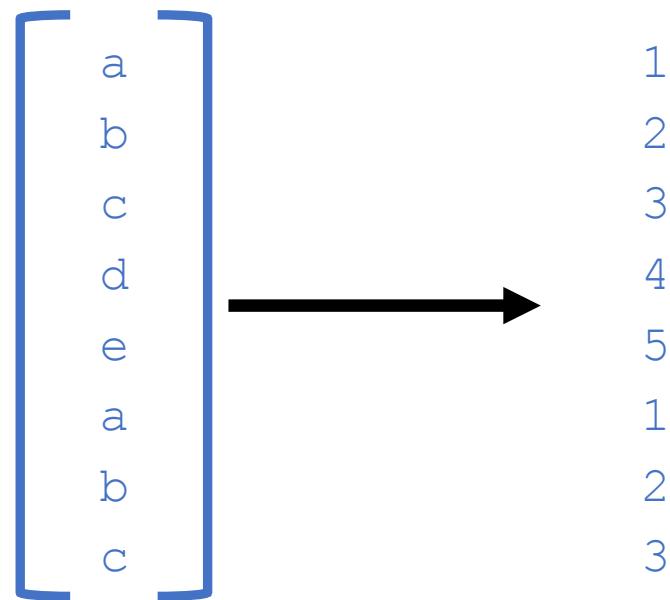
Token	Index
a	1
b	2
c	3
d	4
e	5

2. Character to Vector

Dictionary

Token	Index
a	1
b	2
c	3
d	4
e	5

One-hot encoding (token to vector)

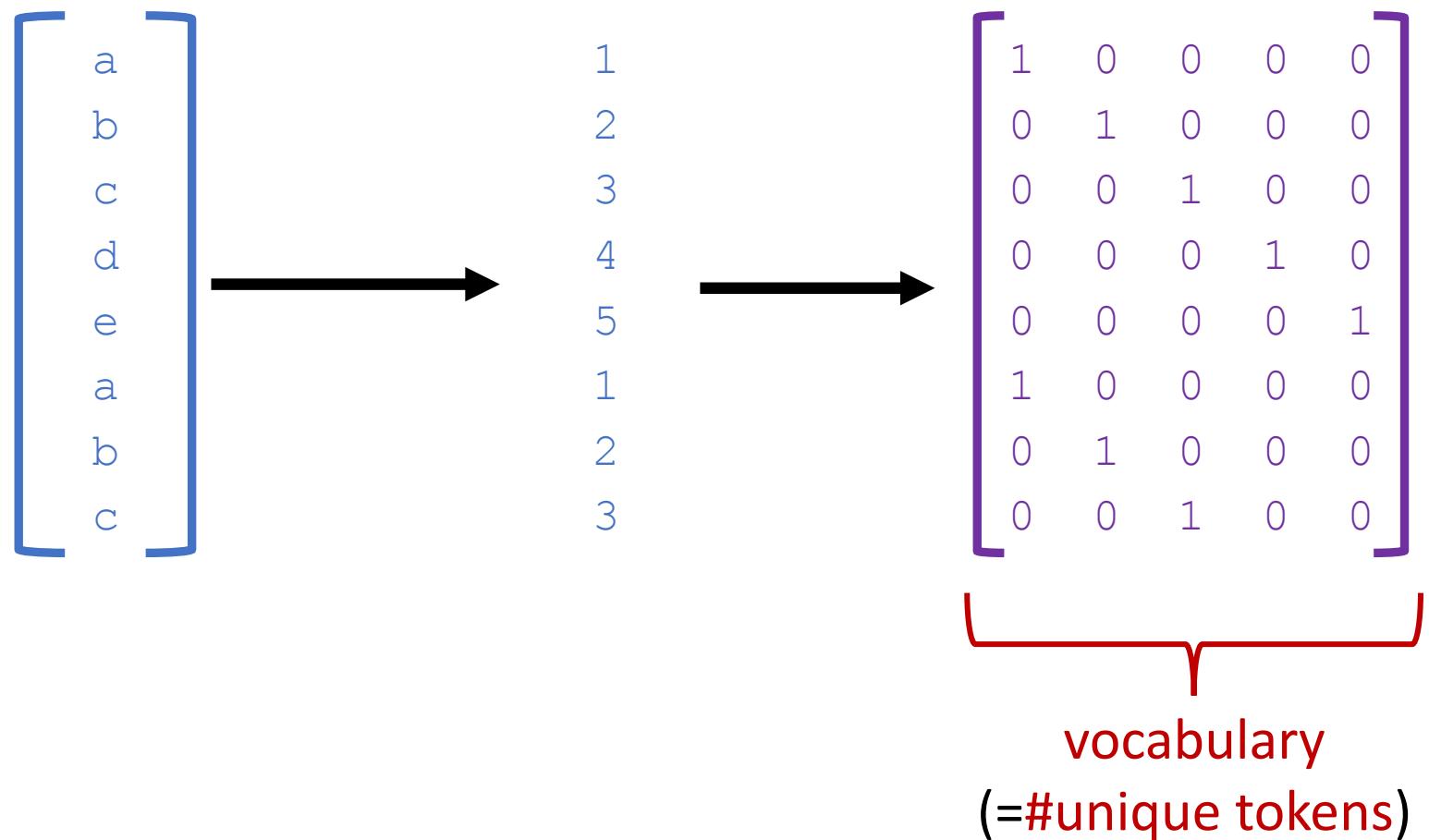


2. Character to Vector

Dictionary

Token	Index
a	1
b	2
c	3
d	4
e	5

One-hot encoding (token to vector)



2. Character to Vector

- Vocabulary is $v = 57$ (including letter, blank, punctuation, etc.)
- Segment length is $l = 60$. (A segment has 60 chars.)

`segments[i]: \nsupposing that truth is a woma`



`X: 60×57 matrix`

`next_chars[i]: n`



`y: 57×1 vector`

2. Character to Vector

- Vocabulary is $v = 57$ (including letter, blank, punctuation, etc.)
- Segment length is $l = 60$. (A segment has 60 chars.)
- Number of segments is $n = 200,278$. (Number of training samples.)

`segments[i]: \nsupposing that truth is a woma`



`X: 60×57 matrix`

`next_chars[i]: n`



`y: 57×1 vector`

There are $n = 200,278$ such pairs.

3. Build a Neural Network

```
from keras import layers

model = keras.models.Sequential()          l = 60      v = 57
model.add(layers.LSTM(128, input_shape=(seg_len, vocabulary)))
model.add(layers.Dense(vocabulary, activation='softmax'))
model.summary()                          v = 57
```

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 128)	95232
dense_1 (Dense)	(None, 57)	7353
<hr/>		
Total params: 102,585		
Trainable params: 102,585		
Non-trainable params: 0		

4. Train the Neural Network

```
optimizer = keras.optimizers.RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

4. Train the Neural Network

```
optimizer = keras.optimizers.RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

```
model.fit(x, y, batch_size=128, epochs=1)
```

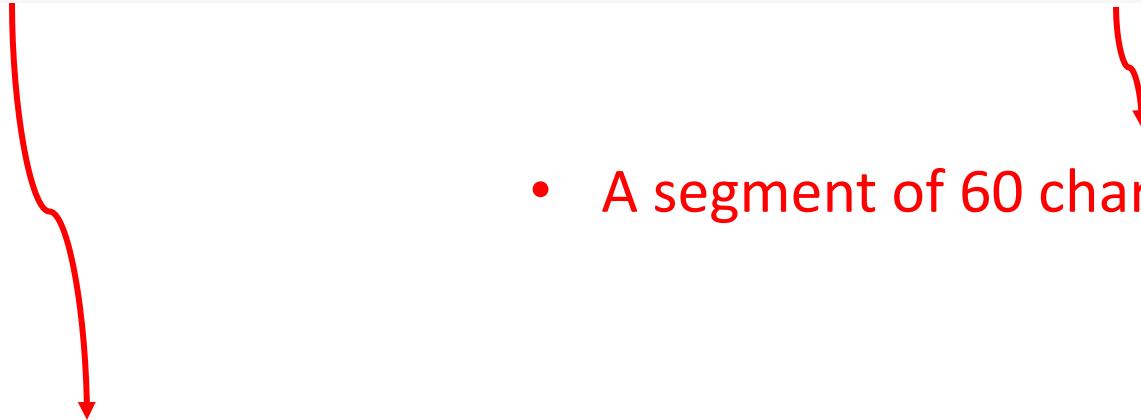
```
Epoch 1/1
200278/200278 [=====] - 117s 586us/step -
loss: 1.6746
```

- $x[i, :, :, :]$ is a segment of 60 chars represented by a 60×57 matrix.
- $y[i, :]$ is the next char represented by a 57-dim vector.

Text Generation

Predict the Next Char

```
pred = model.predict(x_input, verbose=0)[0]
```



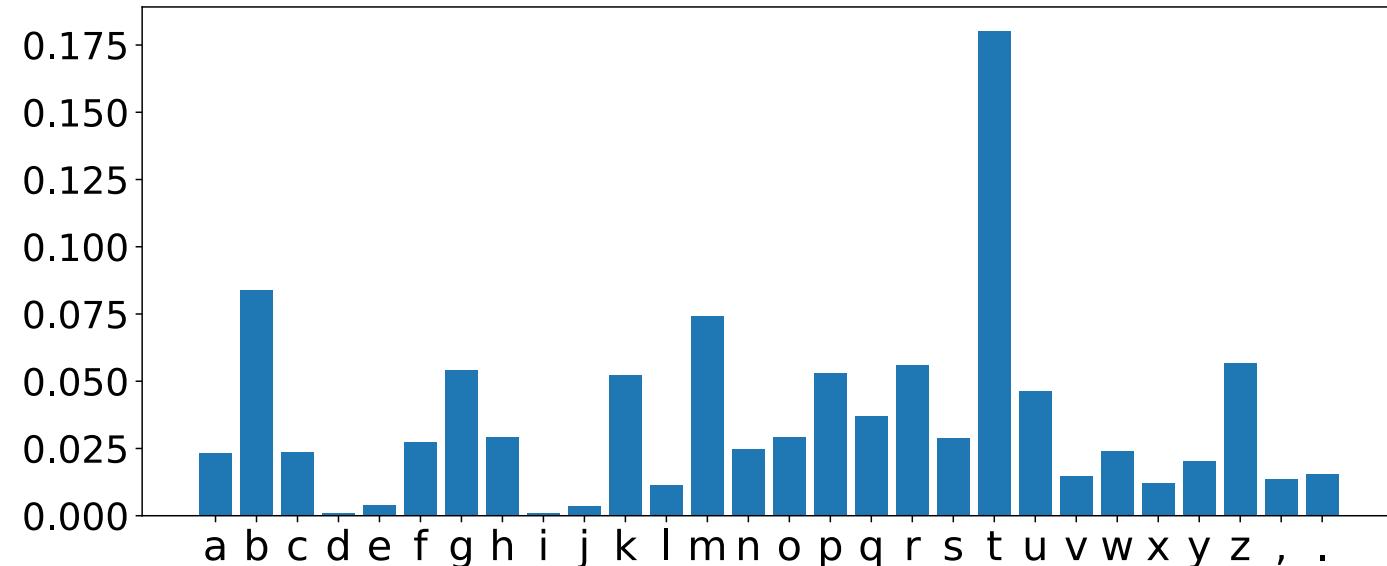
- A segment of 60 chars (represented by a 60×57 matrix).

The probability distribution over the 57 unique chars.

Predict the Next Char

```
pred = model.predict(x_input, verbose=0)[0]
```

Question: Given the **distribution**, what will be the next char?



Predict the Next Char

```
pred = model.predict(x_input, verbose=0)[0]
```

Question: Given the **distribution**, what will be the next char?

- Option 1: greedy selection.
 - `next_index = np.argmax(pred)`
 - It is deterministic.
 - Empirically not good.

Predict the Next Char

```
pred = model.predict(x_input, verbose=0)[0]
```

Question: Given the **distribution**, what will be the next char?

- Option 1: greedy selection.
- Option 2: sampling from the multinomial distribution.
 - `next_onehot = np.random.multinomial(1, pred, 1)`
 - `next_index = np.argmax(next_onehot)`
 - Maybe too random.

Predict the Next Char

```
pred = model.predict(x_input, verbose=0)[0]
```

Question: Given the **distribution**, what will be the next char?

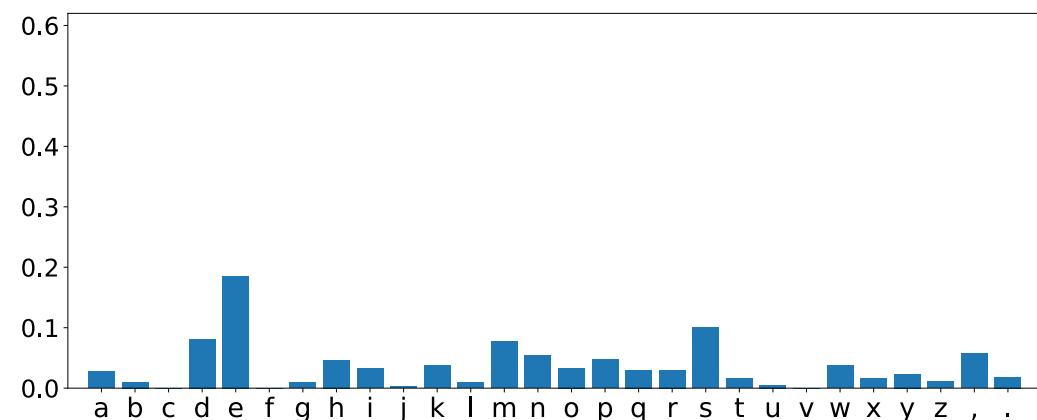
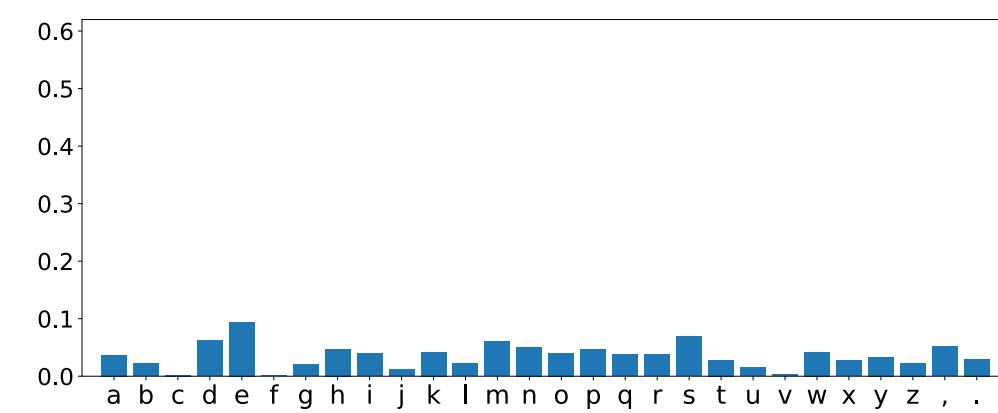
- Option 1: greedy selection.
- Option 2: sampling from the multinomial distribution.
- Option 3: adjusting the multinomial distribution.
 - `pred = pred ** (1 / temperature)`
 - `pred = pred / np.sum(pred)`
 - Sample according to `pred`.
 - Between greedy and multinomial (controlled `temperature`).

Predict the Next Char

Option 3: adjusting the multinomial distribution.

- `pred = pred ** (1 / temperature)`
- `pred = pred / np.sum(pred)`
- Between greedy and multinomial (controlled `temperature`).

`temperature=0.5`

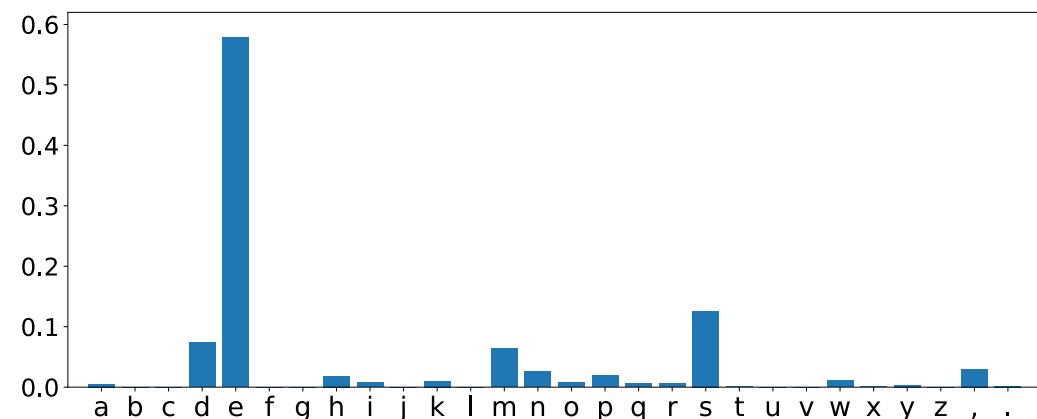
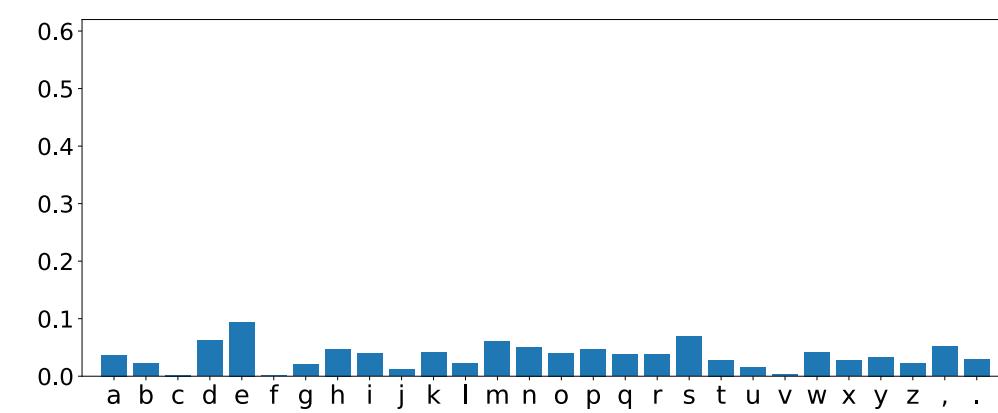


Predict the Next Char

Option 3: adjusting the multinomial distribution.

- `pred = pred ** (1 / temperature)`
- `pred = pred / np.sum(pred)`
- Between greedy and multinomial (controlled `temperature`).

`temperature=0.2`



Text Generation: An Example

Example: seg_len=18

initial_input (seed): “the cat sat on the”

next_char: '_' (blank)

Text Generation: An Example

Example: seg_len=18

initial_input (seed): “the cat sat on the”

next_char: ' '_ (blank)

input: “he cat sat on the _”

next_char: 'm'

Text Generation: An Example

Example: seg_len=18

initial_input (seed): “the cat sat on the”

next_char: '_' (blank)

input: “he cat sat on the ”

next_char: 'm'

input: “e cat sat on the m”

next_char: 'a'

Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 1 epoch

"immediately disclose what it really is--namely,
a will to the the believest constive the art of
the persision the says of a gan himself need not
religion a consting be naturing and suld the
pretendents as the constion. the are the good
teach that the most and find of endent and the
self it of instinct a mean constition of can the
constive in this sour from the bad and all the
philosophy to condividuation men and the goence
the condines the all as most strat"

Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 1 epoch (another sample)

"immediately disclose what it really is--namely,
a will to thes and in the world the mist a dest
reality and lading the been in the most as and
fanition as the reaption of the stenles a prease
and be and disting and regard the goven the most
and distentive to the from stinct forms and
instinct the believes the need itself the feess
and virtue this says of the gone consting to man
instinctian the become and discative of the
present the free the consine of pas"

Generate Text Using the Trained LSTM

Initial input (seed)

"immediately disclose what it really is--namely,
a will to the"

Generated text after 5 epochs

"immediately disclose what it really is--namely,
a will to the oll and power and that the
complises the fundamental and emotions of the
developation of the propest and sympathy of the
far the long standard, and the most present--
under the personally man hard and every stands
have been hat the soul and conscience, the
intellecation of germans of should for the
religious profoundly in this included and
naturally as we progres and "will of the most
same same and"

Generate Text Using the Trained LSTM

Initial input (seed)

“immediately disclose what it really is--namely,
a will to the”

Generated text after 20 epochs

“immediately disclose what it really is--namely,
a will to the self-religious superitious self-
partial religion himself of the superstition of
the contrast in the account in the person of the
assertion, at the valuations and consequences and
things has no longer and how only a man of the
soul and manifest of the disciplines and an whom
all to the constancy of its own power, in the
constraining in the truth of the strength of life
of the see to remain in the”

Summary

Train a Neural Network

1. Partition text to (segment, next_char) pairs.
2. One-hot encode the characters.
 - Character $\rightarrow v \times 1$ vector.
 - Segment $\rightarrow l \times v$ matrix.
3. Build and train a neural network.
 - $l \times v$ matrix ==> LSTM ==> Dense ==> $v \times 1$ vector.

Text Generation

1. Propose a seed segment.
2. Repeat the followings:
 - a) Feed the segment (with one-hot) to the neural network.
 - b) The neural network outputs probabilities.
 - c) $\text{next_char} \leftarrow$ Sample from the probabilities.
 - d) Append next_char to the segment.

RNNs for Machine Translation

Sequence-to-Sequence Model (Seq2Seq)

English

German

“do you agree” => **[Seq2Seq]** => “bist du einverstanden”

“go to sleep” => **[Seq2Seq]** => “gehen Sie schlafen”

“We will fight” => **[Seq2Seq]** => “Wir werden kämpfen”

Machine Translation Data

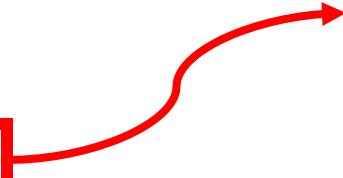
Datasets

- Tab-delimited Bilingual Sentence Pairs: <http://www.manythings.org/anki/>

C ⓘ www.manythings.org/anki/

Things.org Reading ▾ Sentences ▾ More ▾

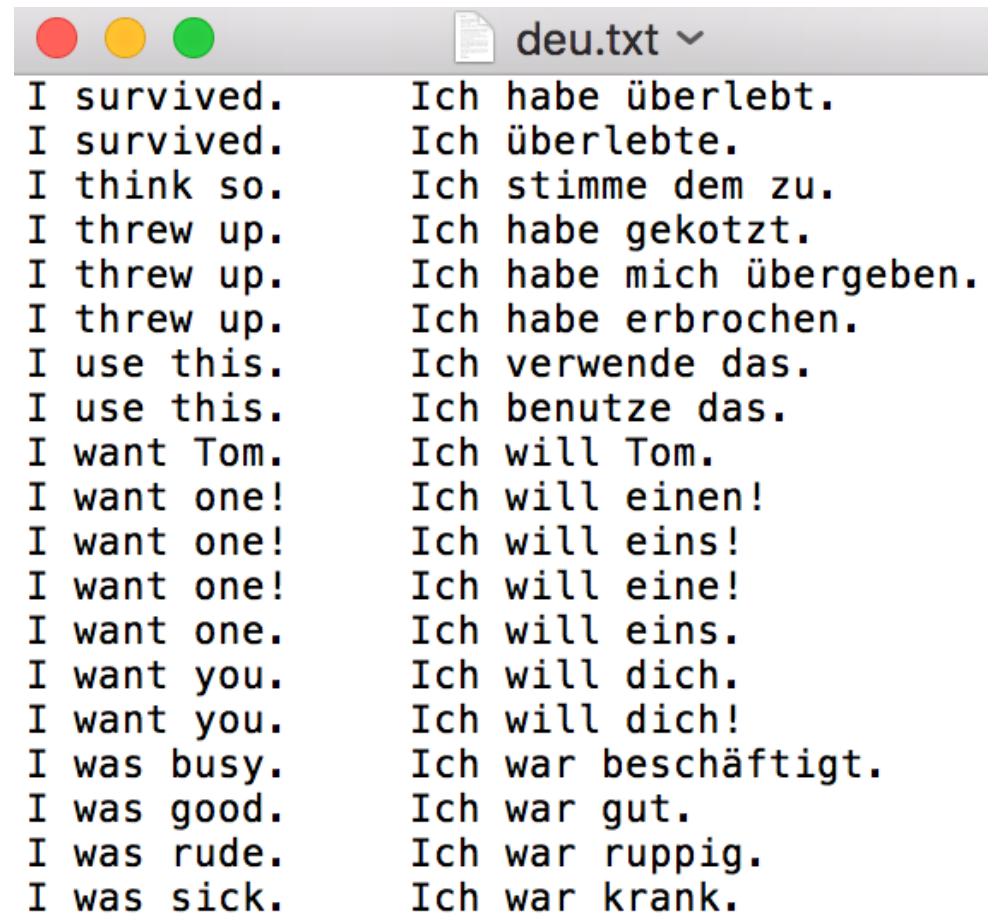
- 🇩🇰 Danish - English [dan-eng.zip](#) (14639)
- 🇳🇱 Dutch - English [nld-eng.zip](#) (28224)
- 🇪🇪 Estonian - English [est-eng.zip](#) (1550)
- 🇫🇮 Finnish - English [fin-eng.zip](#) (49208)
- 🇫🇷 French - English [fra-eng.zip](#) (155507)
- 🇬🇧 Galician - English [glg-eng.zip](#) (126)
- 🇬🇪 Georgian - English [kat-eng.zip](#) (125)
- 🇩🇪 German - English [deu-eng.zip](#) (170297) (highlighted)
- 🇬🇷 Greek - English [ell-eng.zip](#) (14485)
- 🇮🇱 Hebrew - English [heb-eng.zip](#) (126137)
- 🇮🇳 Hindi - English [hin-eng.zip](#) (2867)
- 🇭🇺 Hungarian - English [hun-eng.zip](#) (54197)
- 🇮🇸 Icelandic - English [isl-eng.zip](#) (6664)
- 🇮🇩 Indonesian - English [ind-eng.zip](#) (6702)
- 🇮🇹 Italian - English [ita-eng.zip](#) (297241)



		deu.txt ▾
I survived.		Ich habe überlebt.
I survived.		Ich überlebte.
I think so.		Ich stimme dem zu.
I threw up.		Ich habe gekotzt.
I threw up.		Ich habe mich übergeben.
I use this.		Ich habe erbrochen.
I use this.		Ich verwende das.
I want Tom.		Ich benutze das.
I want one!		Ich will Tom.
I want one!		Ich will einen!
I want one!		Ich will eins!
I want one!		Ich will eine!
I want one.		Ich will eins.
I want you.		Ich will dich.
I want you.		Ich will dich!
I was busy.		Ich war beschäftigt.
I was good.		Ich war gut.
I was rude.		Ich war ruppig.
I was sick.		Ich war krank.

Datasets

- **Preprocessing:** to lower case, remove punctuation, etc.



A screenshot of a text editor window titled "deu.txt". The window contains a list of sentence pairs, where each English sentence is followed by a German translation. The English sentences are on the left, and the German translations are on the right. The text is in a monospaced font.

I survived.	Ich habe überlebt.
I survived.	Ich überlebte.
I think so.	Ich stimme dem zu.
I threw up.	Ich habe gekotzt.
I threw up.	Ich habe mich übergeben.
I threw up.	Ich habe erbrochen.
I use this.	Ich verwende das.
I use this.	Ich benutze das.
I want Tom.	Ich will Tom.
I want one!	Ich will einen!
I want one!	Ich will eins!
I want one!	Ich will eine!
I want one.	Ich will eins.
I want you.	Ich will dich.
I want you.	Ich will dich!
I was busy.	Ich war beschäftigt.
I was good.	Ich war gut.
I was rude.	Ich war ruppig.
I was sick.	Ich war krank.

1. Tokenization & Build Dictionary

- input_texts => [**Eng_Tokenizer**] => input_tokens
- target_texts => [**Deu_Tokenizer**] => target_tokens



- Use 2 different tokenizers for the 2 languages.
- Then build 2 different dictionaries.

1. Tokenization & Build Dictionary

- input_texts => [**Eng_Tokenizer**] => input_tokens
- target_texts => [**Deu_Tokenizer**] => target_tokens

Tokenization in the **char-level**.

Tokenization in the **word-level**.

1. Tokenization & Build Dictionary

- input_texts => [**Eng_Tokenizer**] => input_tokens
- target_texts => [**Deu_Tokenizer**] => target_tokens

Tokenization in the **char-level**.

Eng_Tokenizer

- "I_am_okay." => ['i', '_', 'a', 'm', ..., 'a', 'y']

Deu_Tokenizer

- "Es geht mir gut" => ['e', 's', '_', ..., 'u', 't']

1. Tokenization & Build Dictionary

Question: Why 2 different tokenizers and dictionaries?

Answer: In the **char-level**, languages have different **alphabets/chars**.

- English: A a, B b, C c ..., Z z. (26 letters ×2).
- German: 26 letters, 3 umlauts (Ä,Ö,Ü), and one ligature (ß).
- Greek: Α α, Β β, Γ γ, Δ δ, ..., Ω ω. (24 letters ×2).
- Chinese: 金 木 水 火 土 ... 赵 钱 孙 李 (a few thousands characters).
- Japanese: あ い う え お ... (46 Hiragana, 46 Katakana, hundreds 漢字).

1. Tokenization & Build Dictionary

Question: Why 2 different tokenizers and dictionaries?

Answer: In the **word-level**, languages have different **vocabulary**.

- English:

Machine learning is a generic term for the artificial generation of knowledge from experience: An artificial system learns from examples and can generalize these after completion of the learning phase.

- Deutsche:

Maschinelles Lernen ist ein Oberbegriff für die künstliche Generierung von Wissen aus Erfahrung: Ein künstliches System lernt aus Beispielen und kann diese nach Beendigung der Lernphase verallgemeinern.

1. Tokenization & Build Dictionary

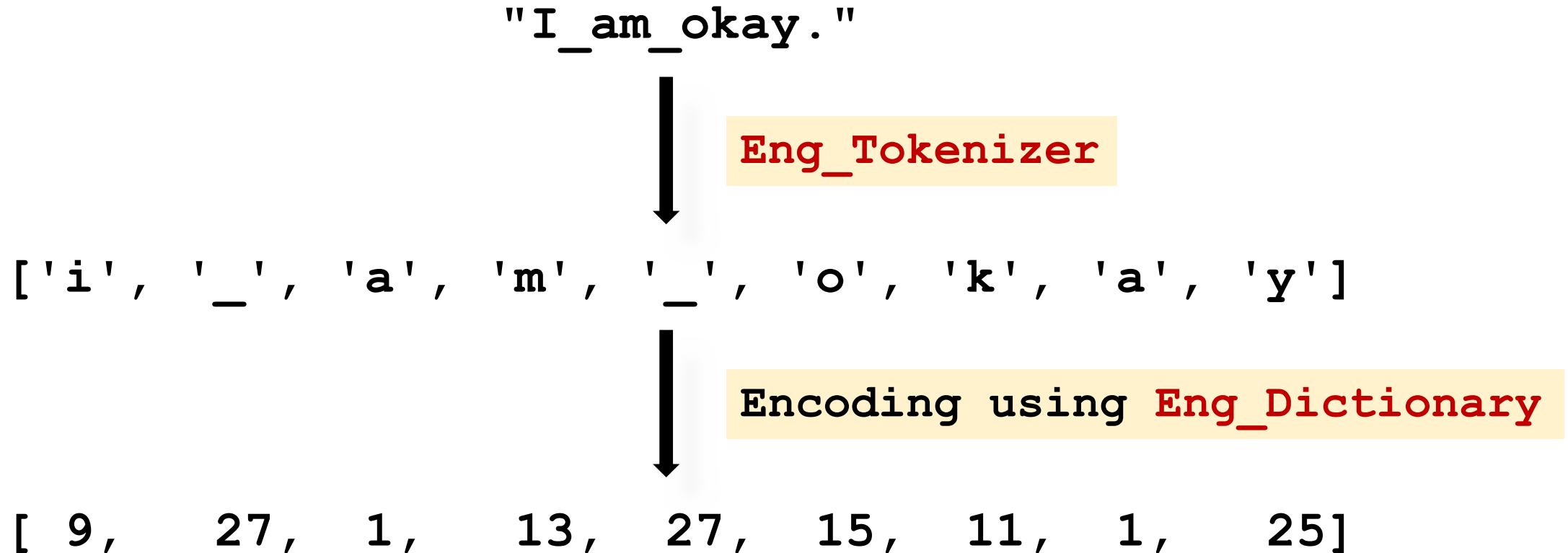
Eng_Dictionary

- 'a' => 1
- 'b' => 2
- 'c' => 3
- 'd' => 4
- ...
- 'z' => 26
- '_' => 27

Deu_Dictionary

- '\t' => 1 
- '\n' => 2 
- 'a' => 3
- 'b' => 4
- 'c' => 5
- 'd' => 6
- ...
- 'z' => 28
- '_' => 29

2. One-Hot Encoding



2. One-Hot Encoding

"Es geht mir gut"



Deu_Tokenizer

['e' , 's' , '_' , 'g' , 'e' , ... , 'g' , 'u' , 't']



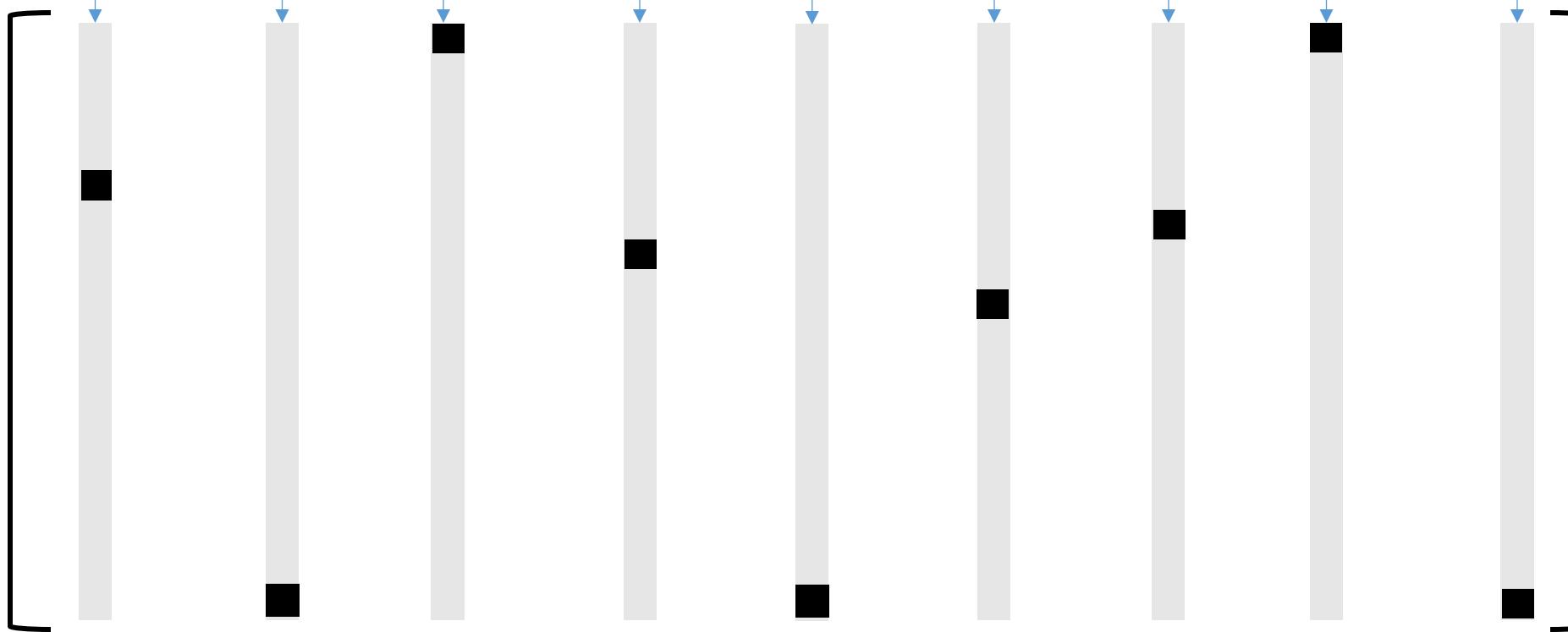
Encoding using Deu_Dictionary

[7 , 21 , 29 , 9 , 7 , ... , 9 , 23 , 22]

2. One-Hot Encoding

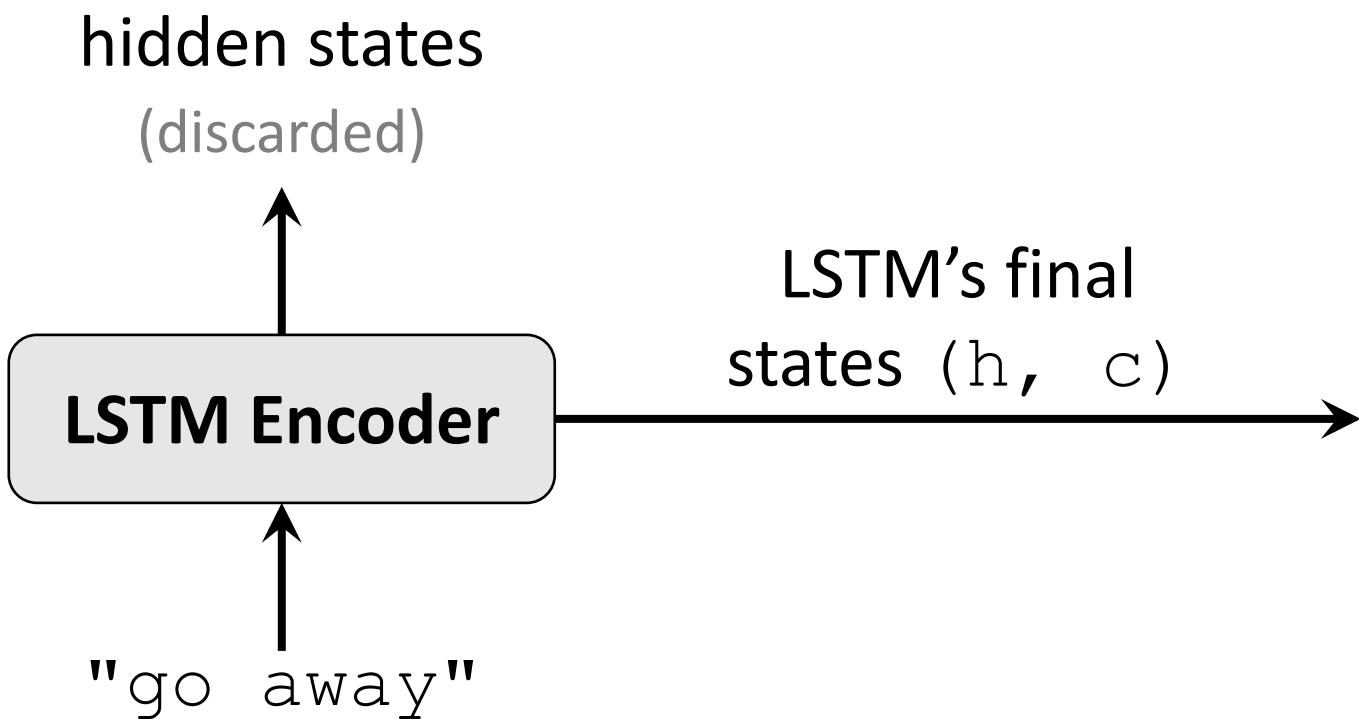
```
['i', '_', 'a', 'm', '_', 'o', 'k', 'a', 'y']
```

```
[ 9, 27, 1, 13, 27, 15, 11, 1, 25]
```

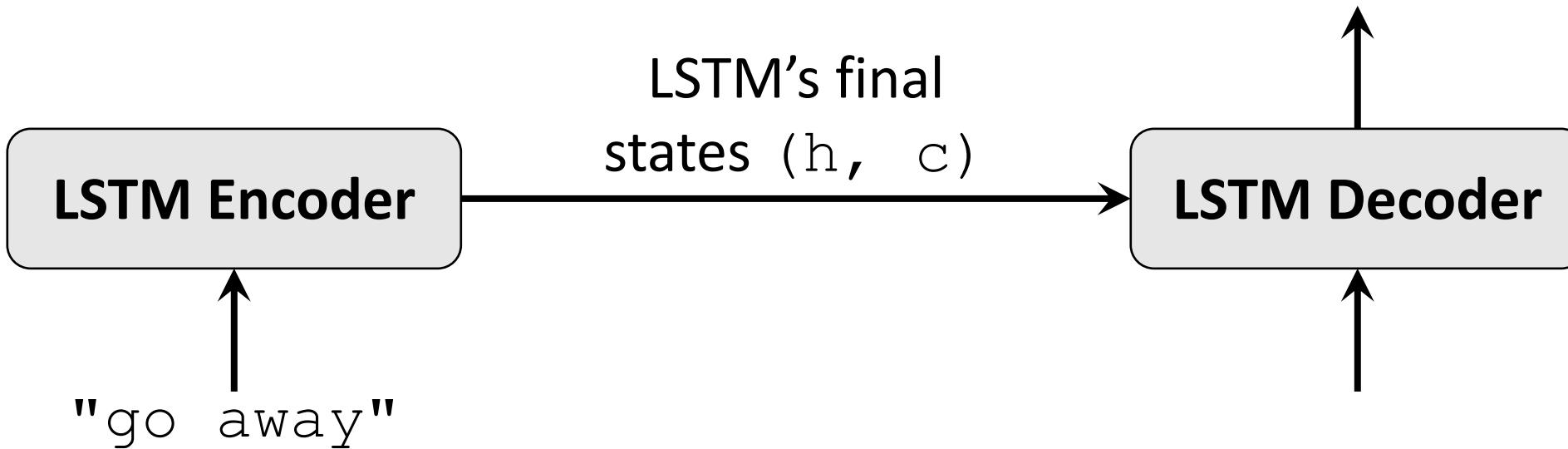


Training Seq2Seq Model

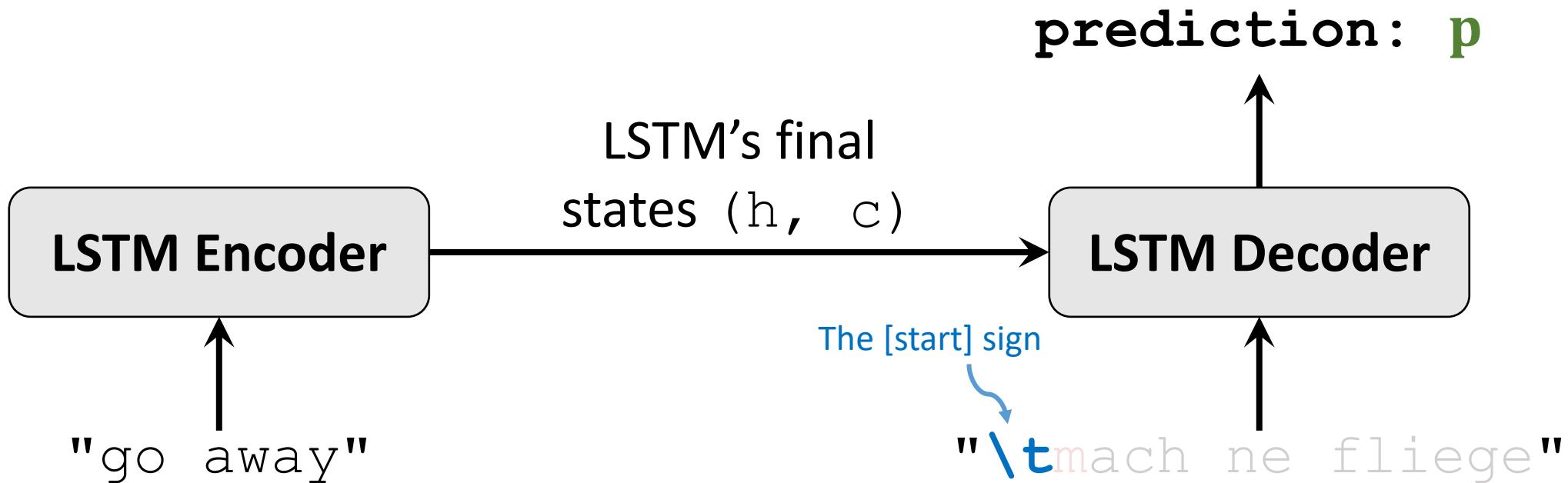
Seq2Seq Model



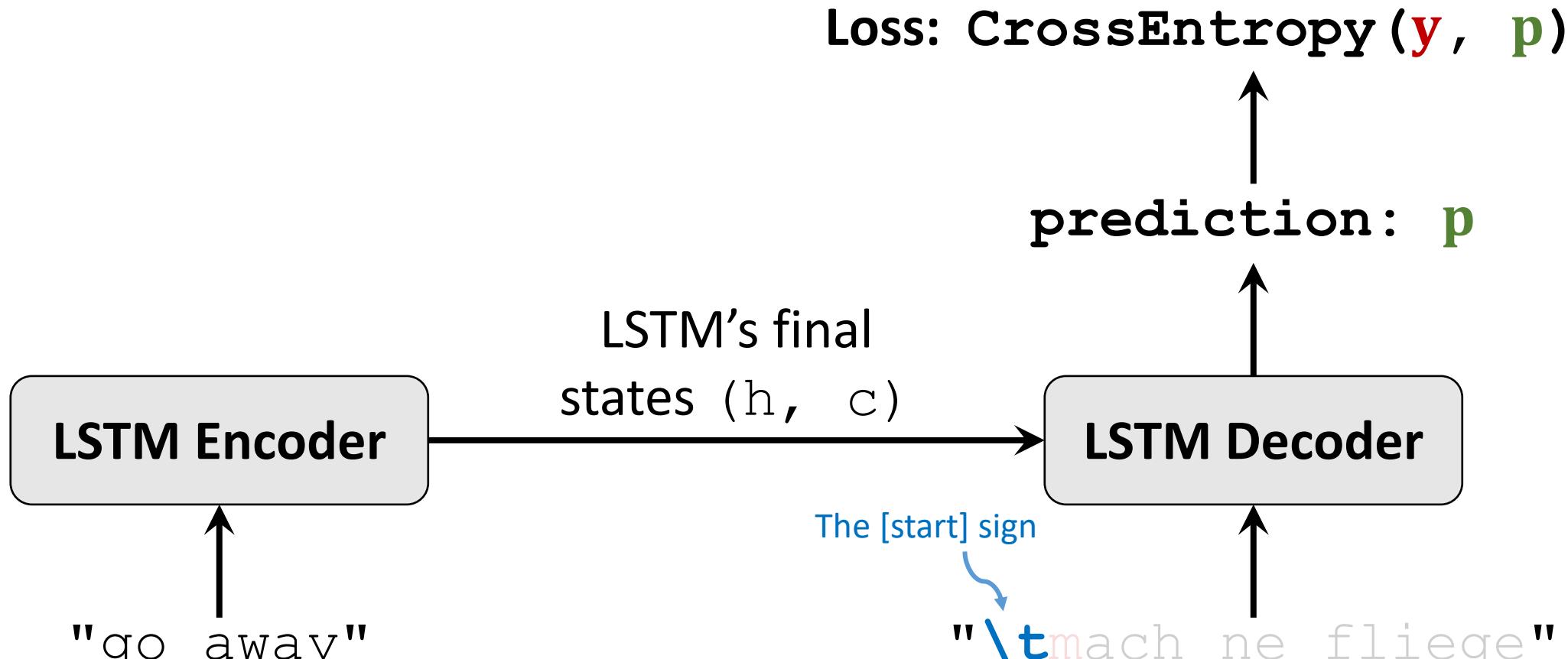
Seq2Seq Model



Training Seq2Seq Model

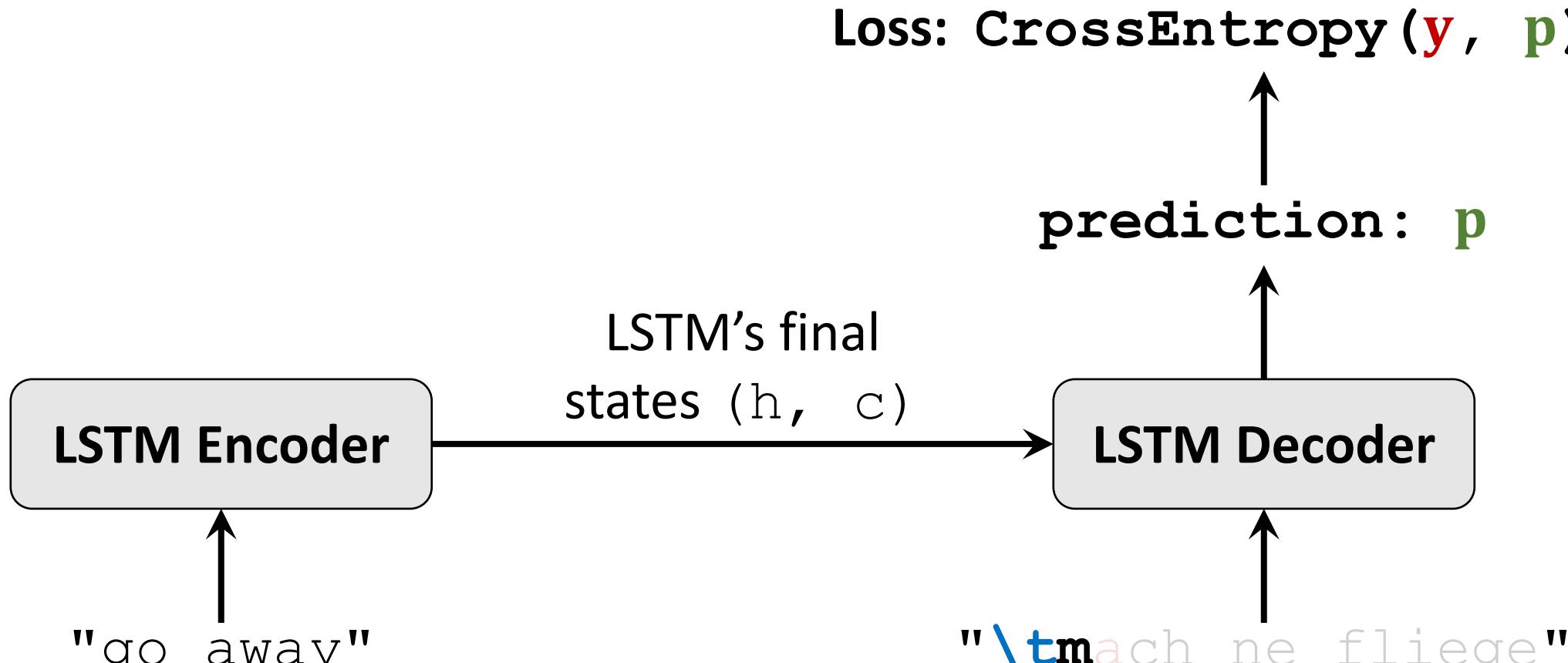


Training Seq2Seq Model



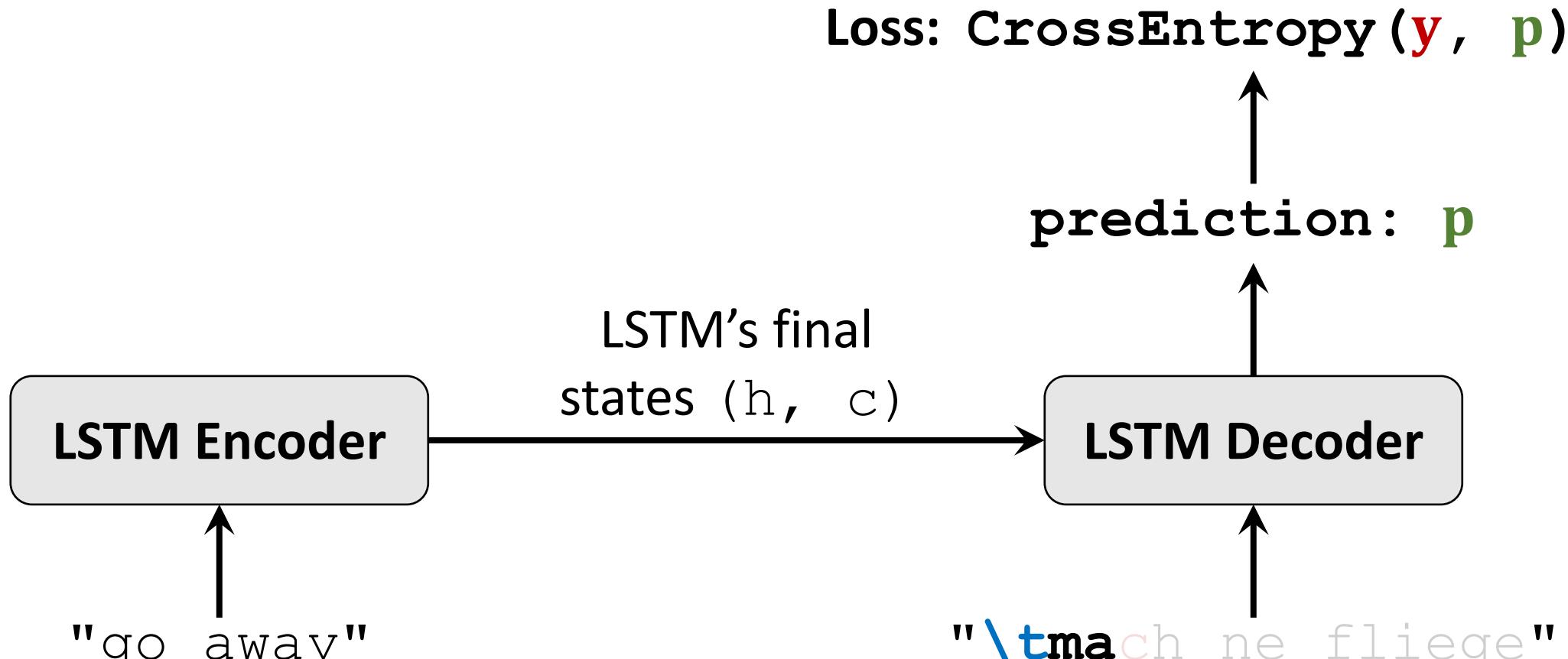
Label y is the one-hot vector of 'm'

Training Seq2Seq Model



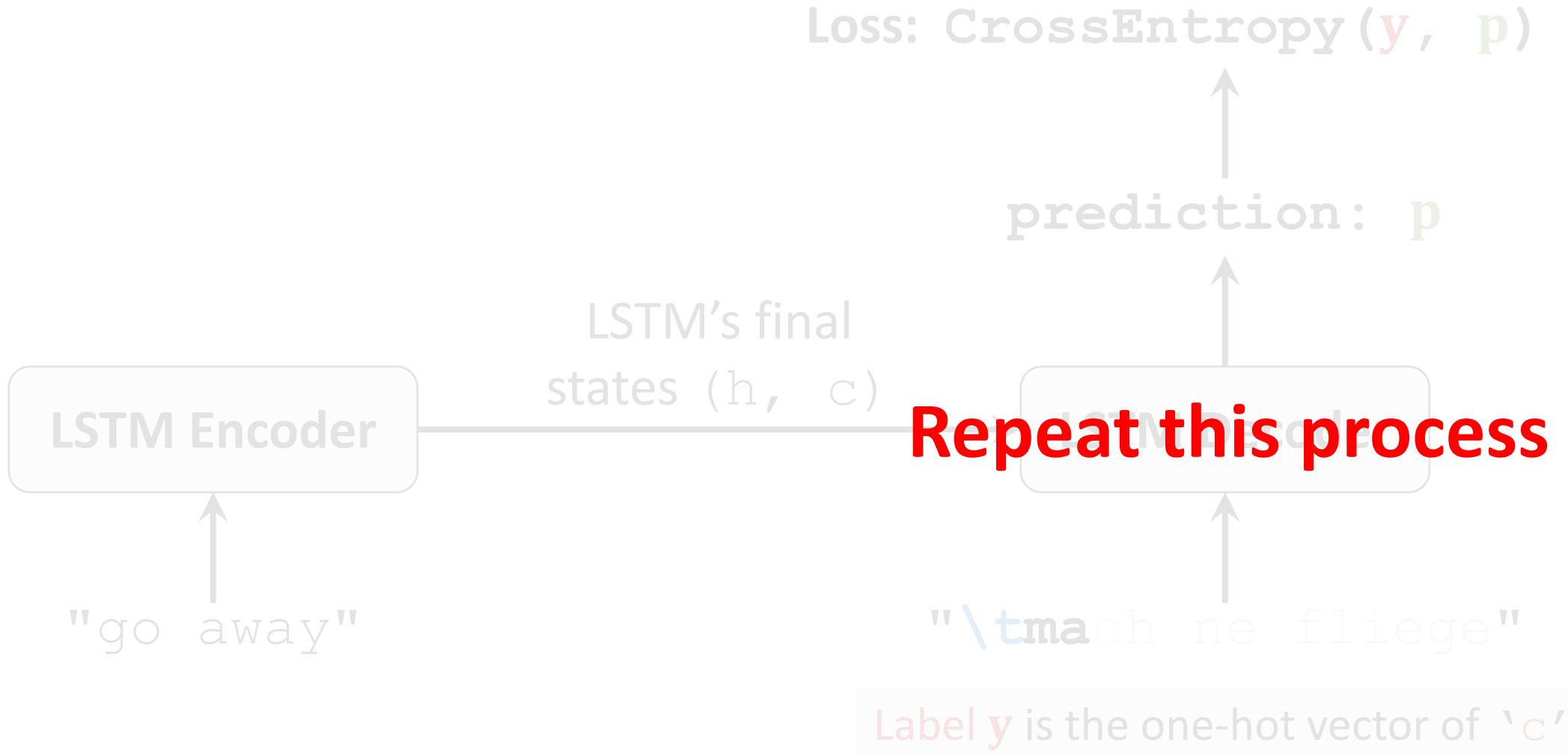
Label y is the one-hot vector of 'a'

Training Seq2Seq Model

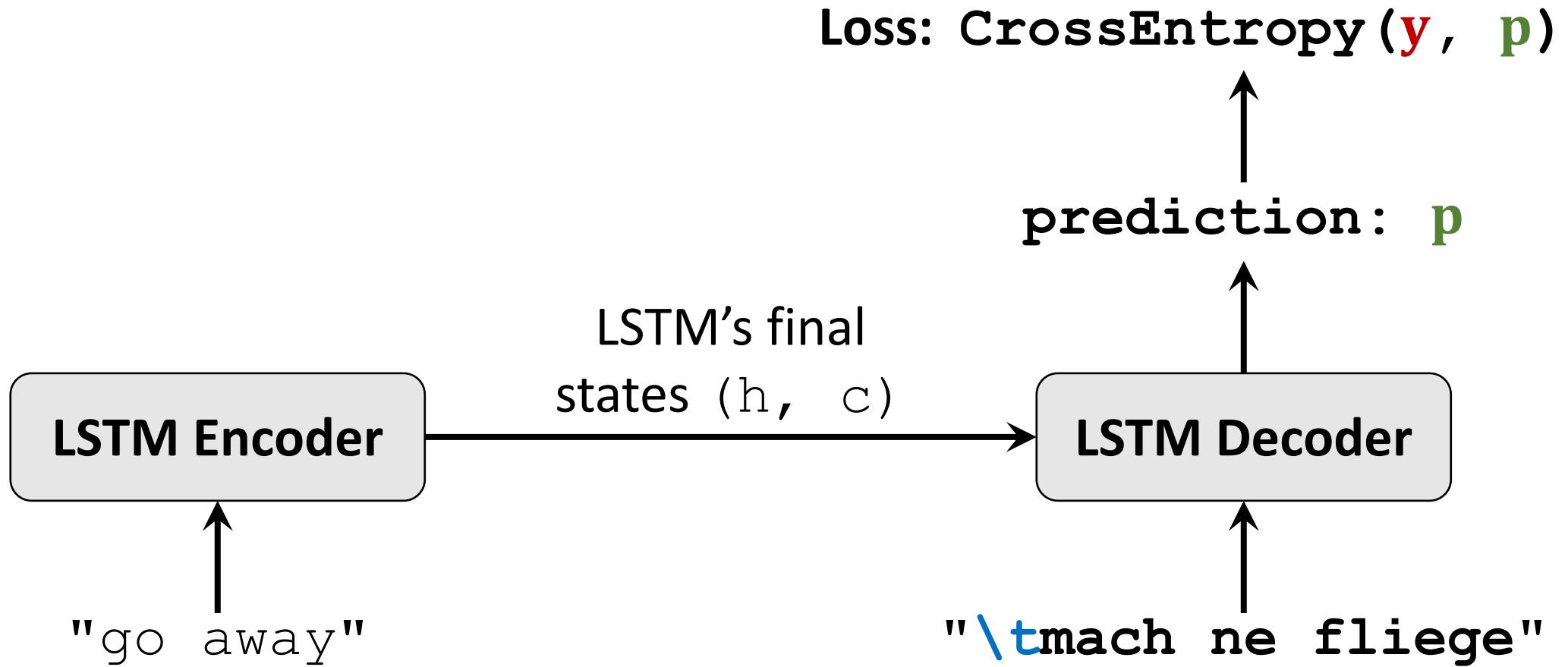


Label y is the one-hot vector of 'c'

Training Seq2Seq Model

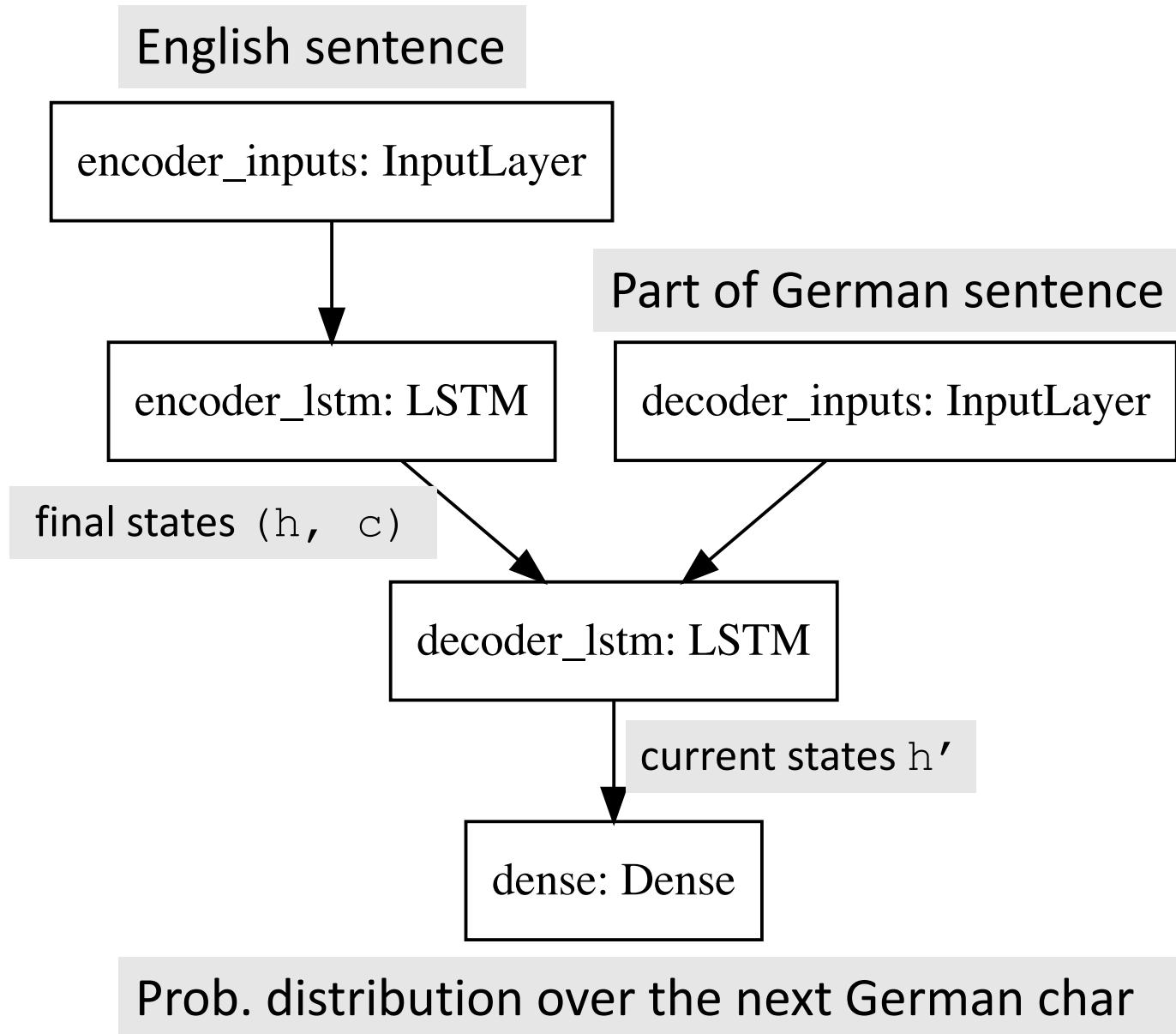


Training Seq2Seq Model



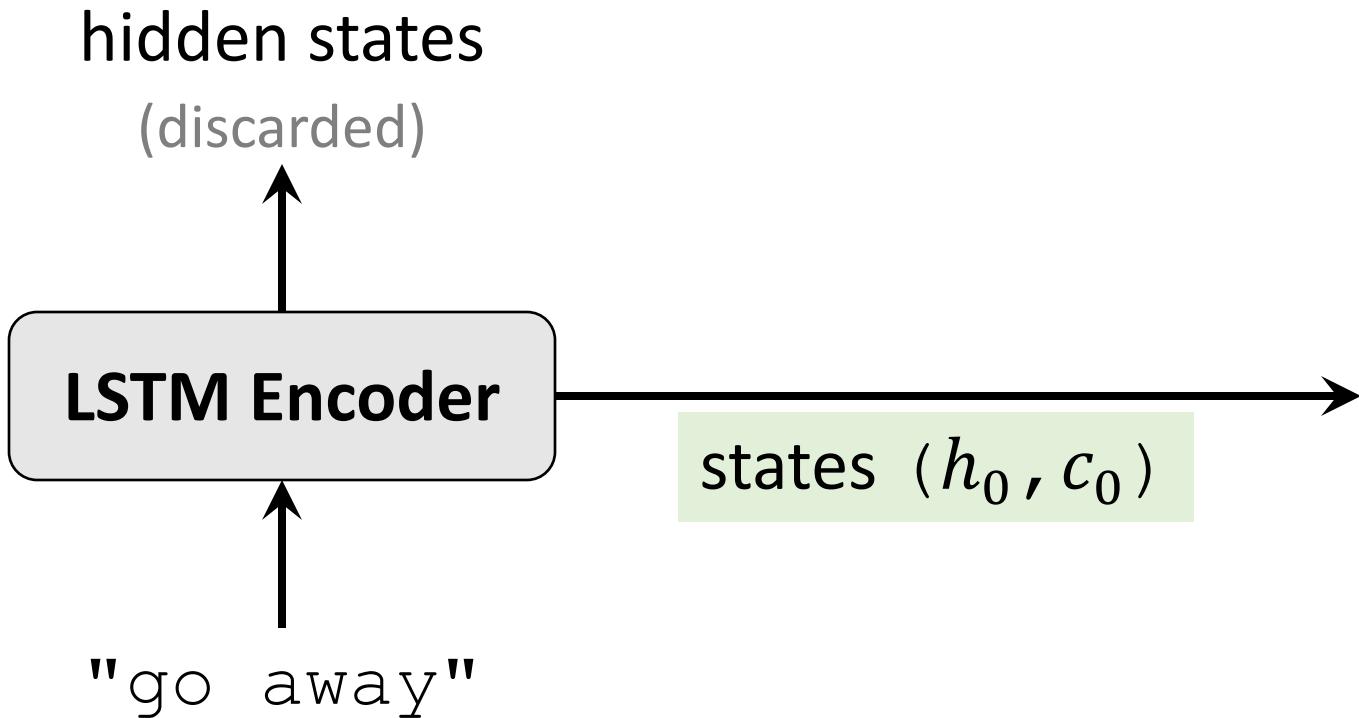
Label y is the one-hot vector of the [stop] sign.

Seq2Seq Model in Keras

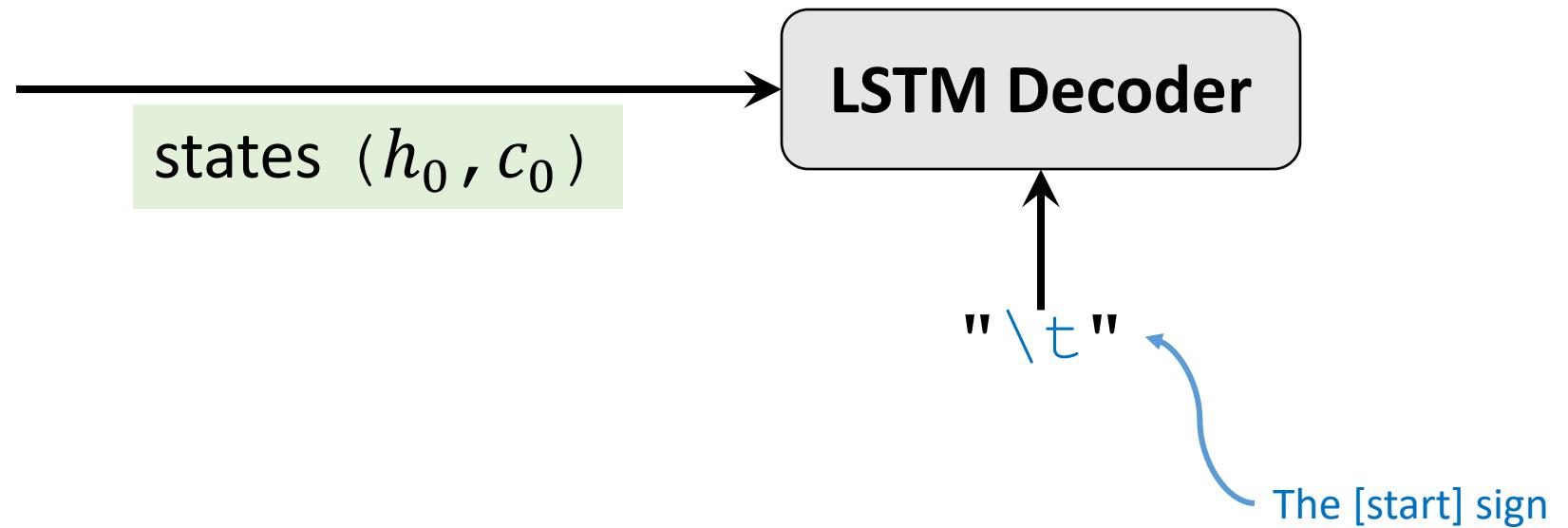


Inference Using the Seq2Seq Model

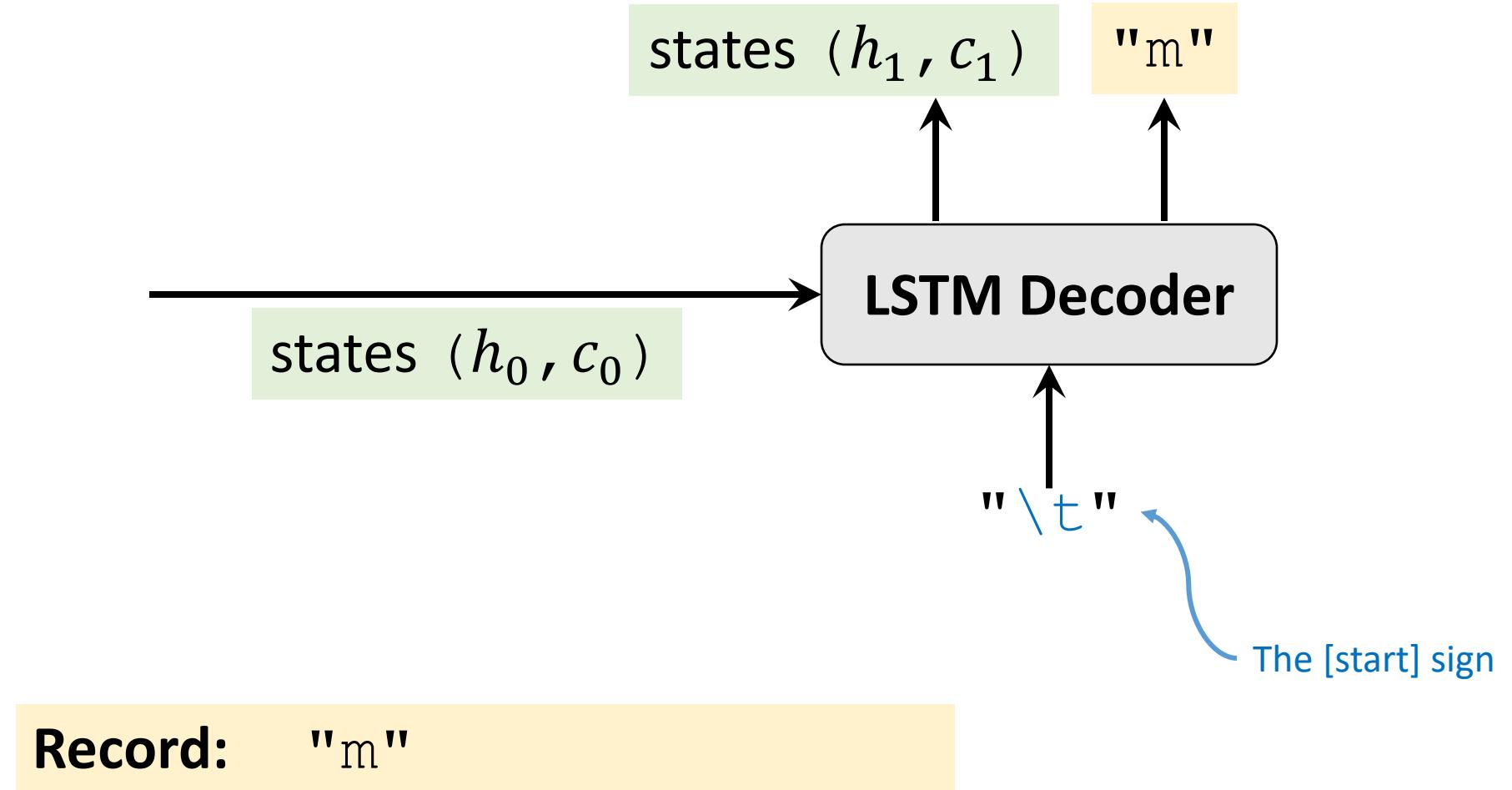
Inference



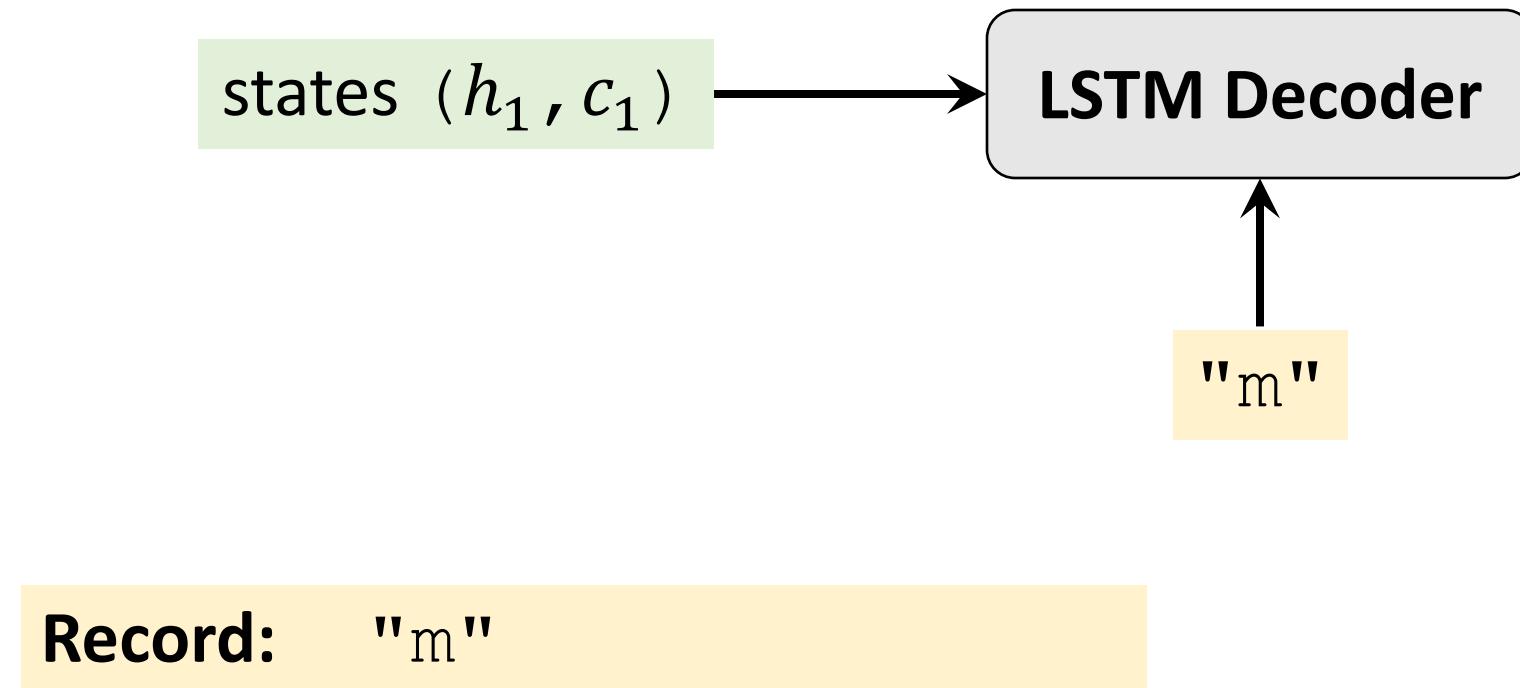
Inference



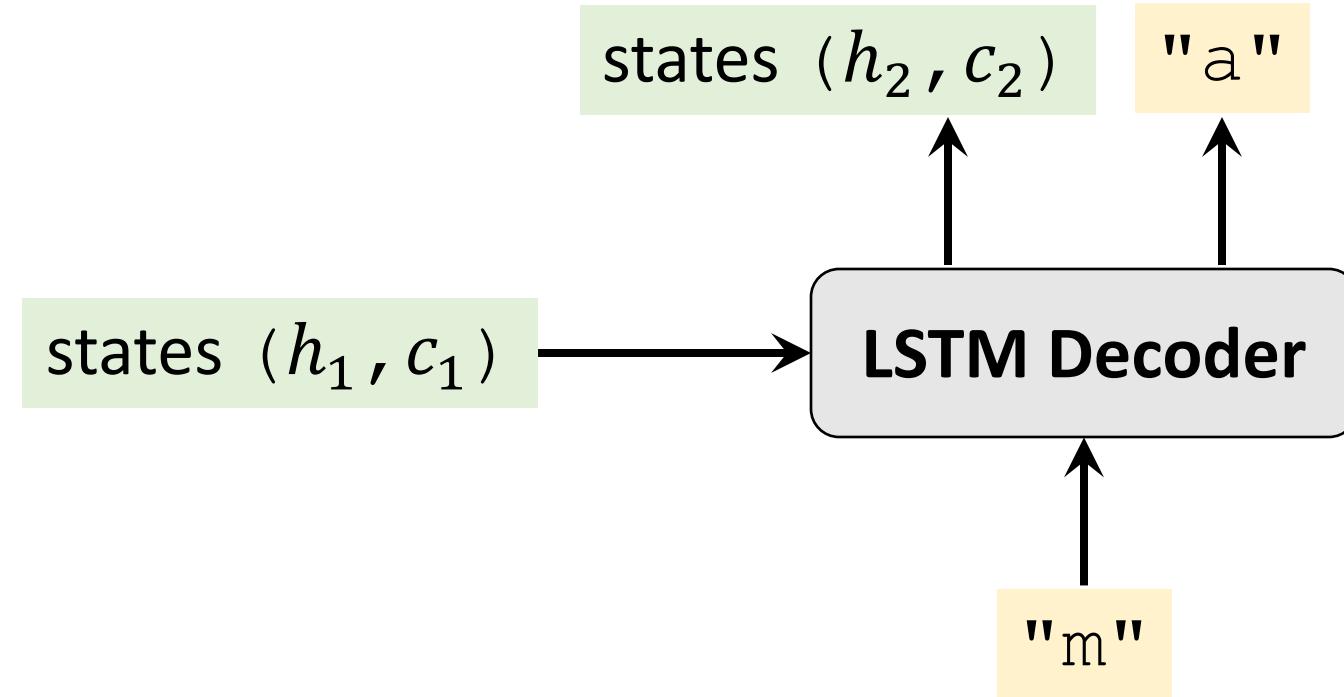
Inference



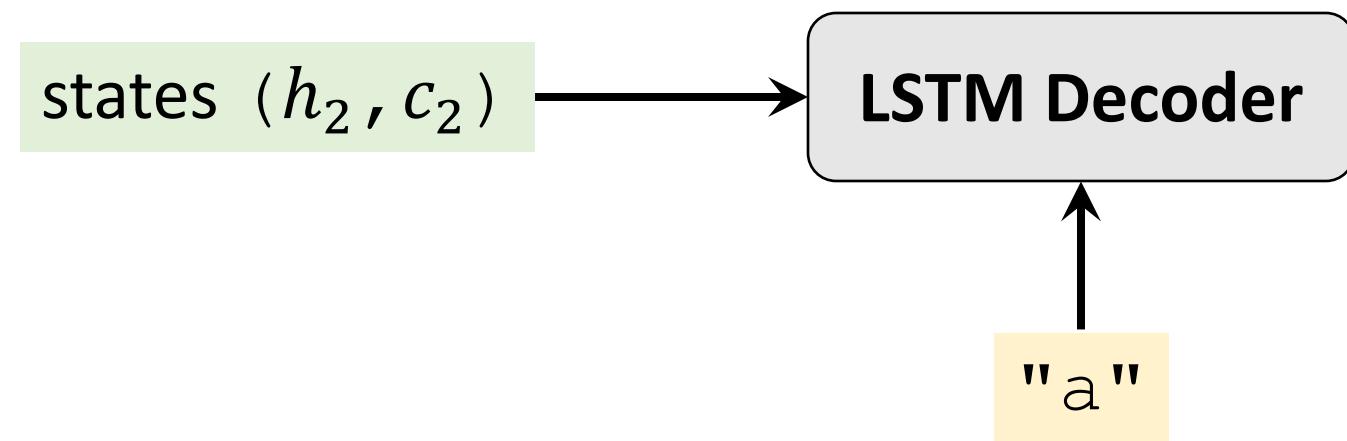
Inference



Inference

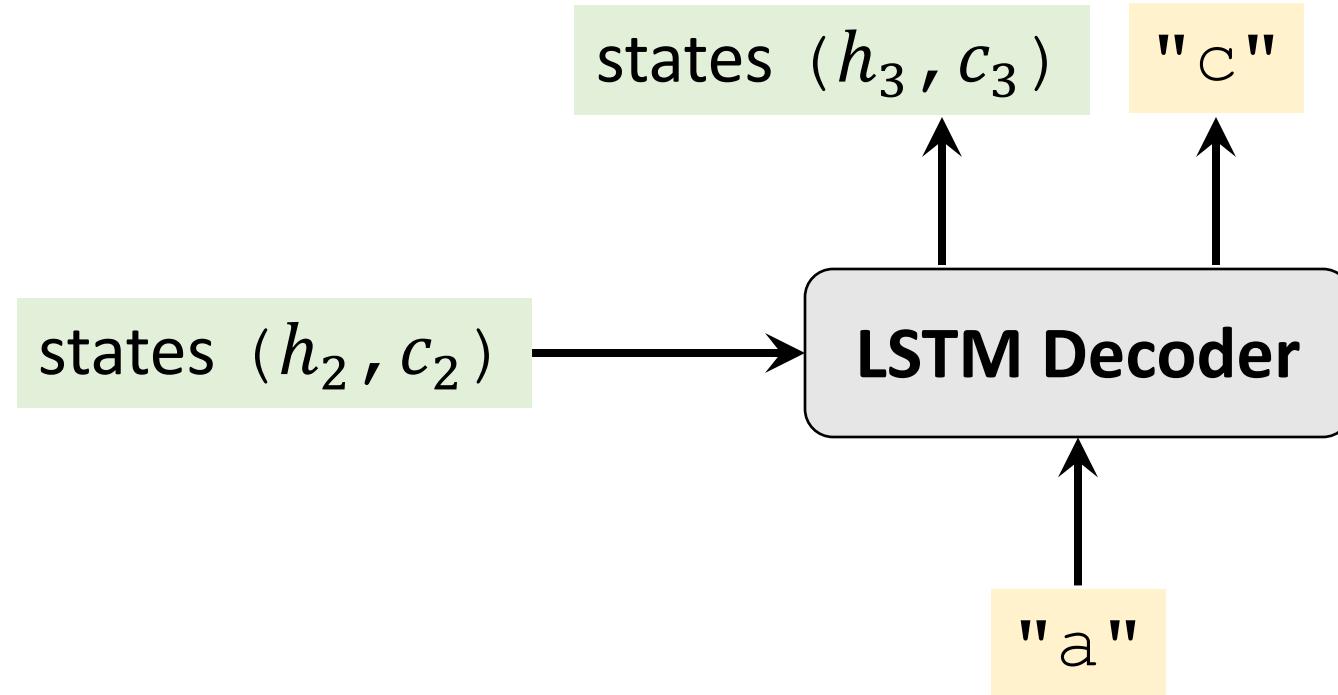


Inference



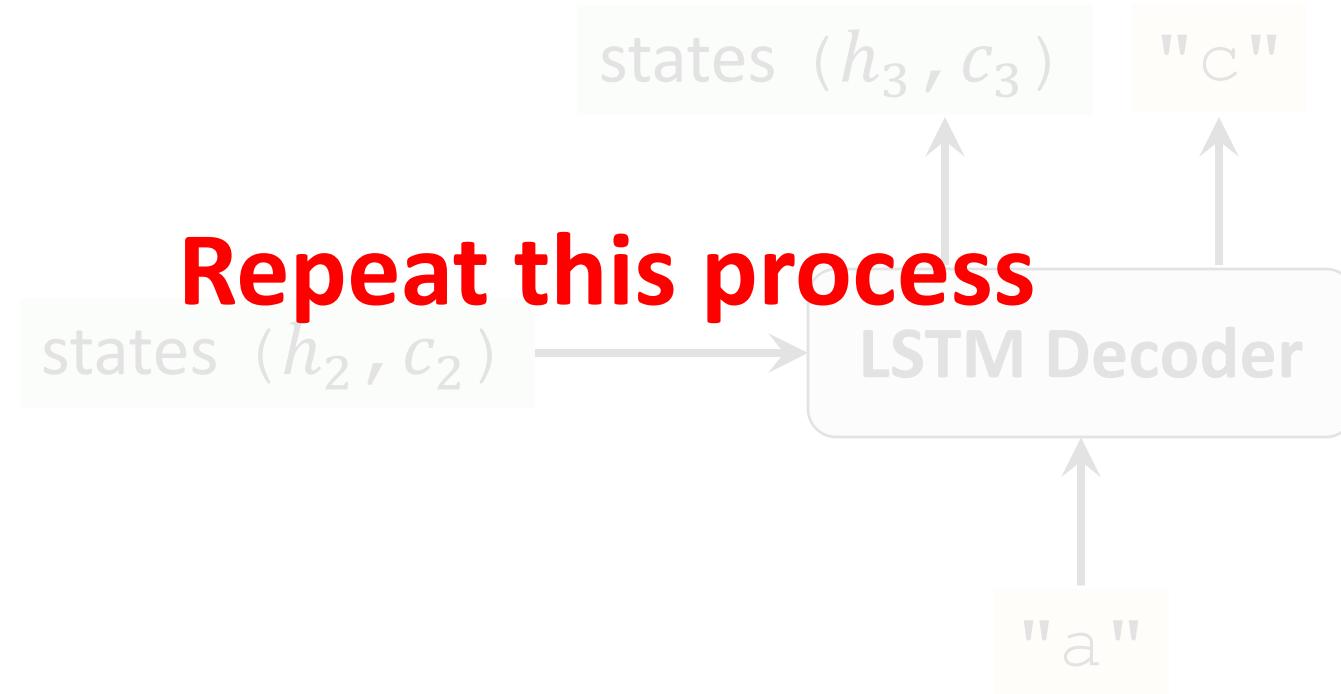
Record: "ma"

Inference



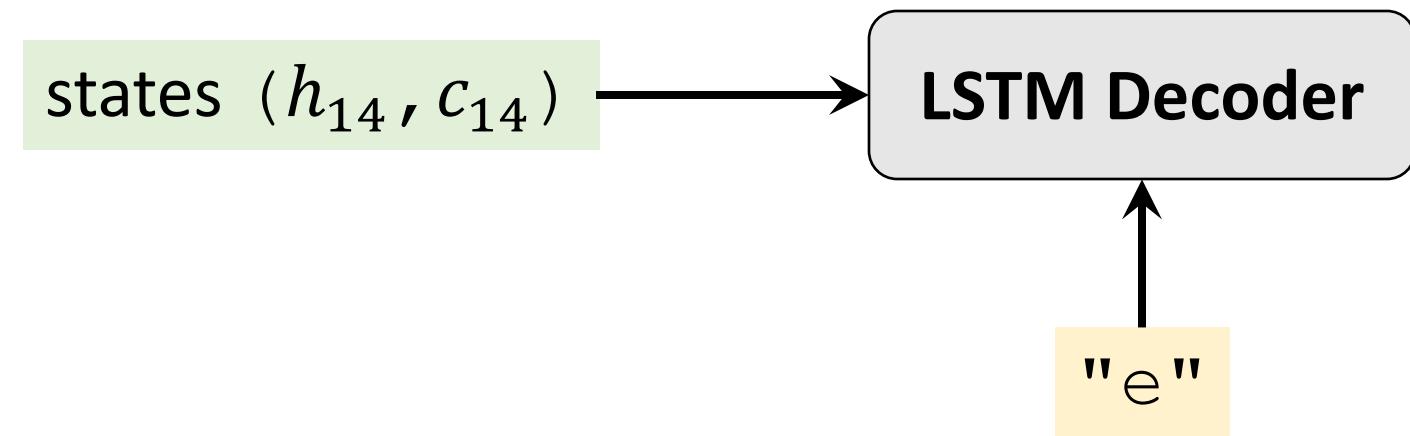
Record: "mac"

Inference



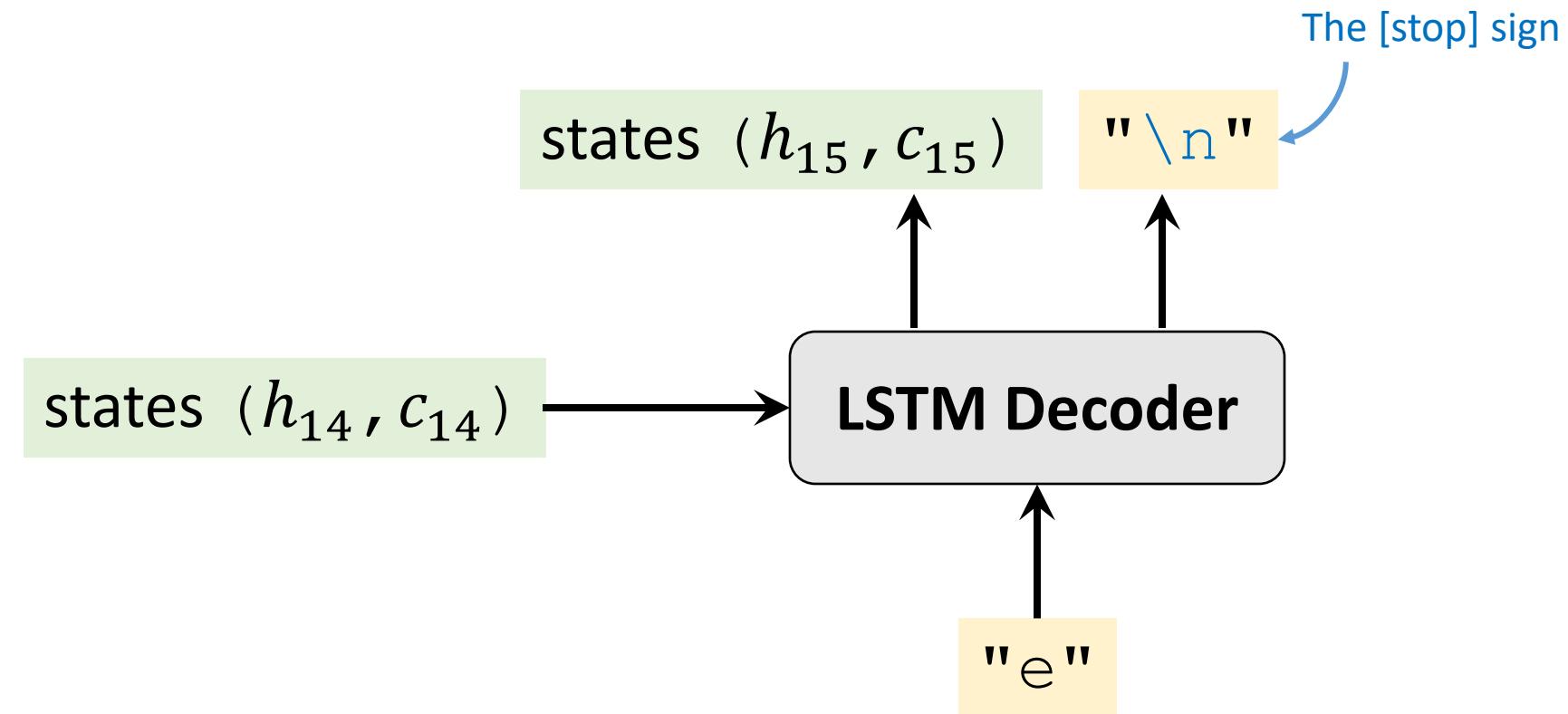
Record: "mac"

Inference



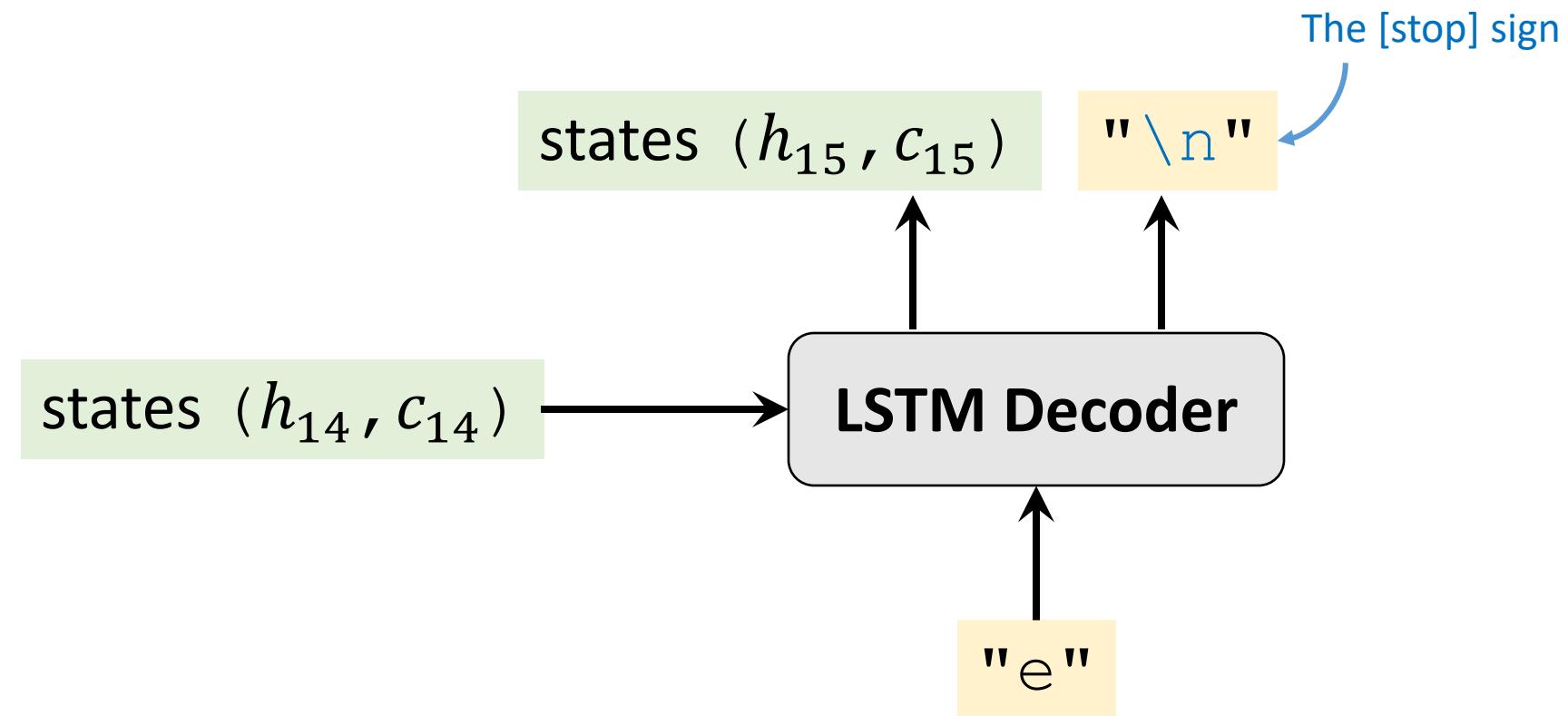
Record: "mach ne fliege"

Inference



Record: "mach ne fliege"

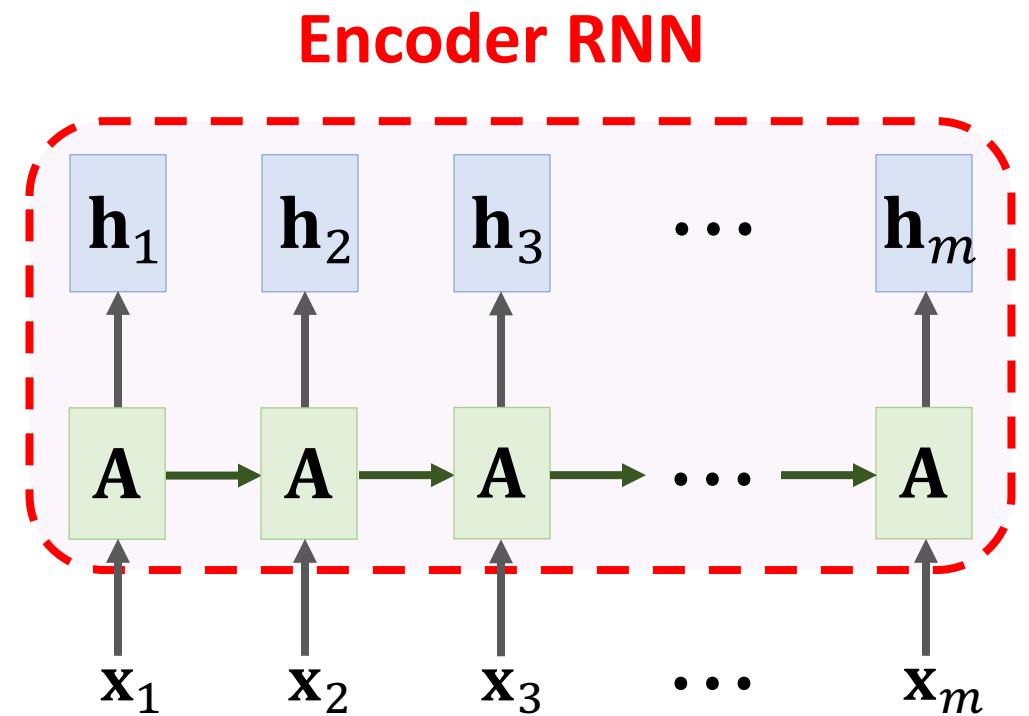
Inference



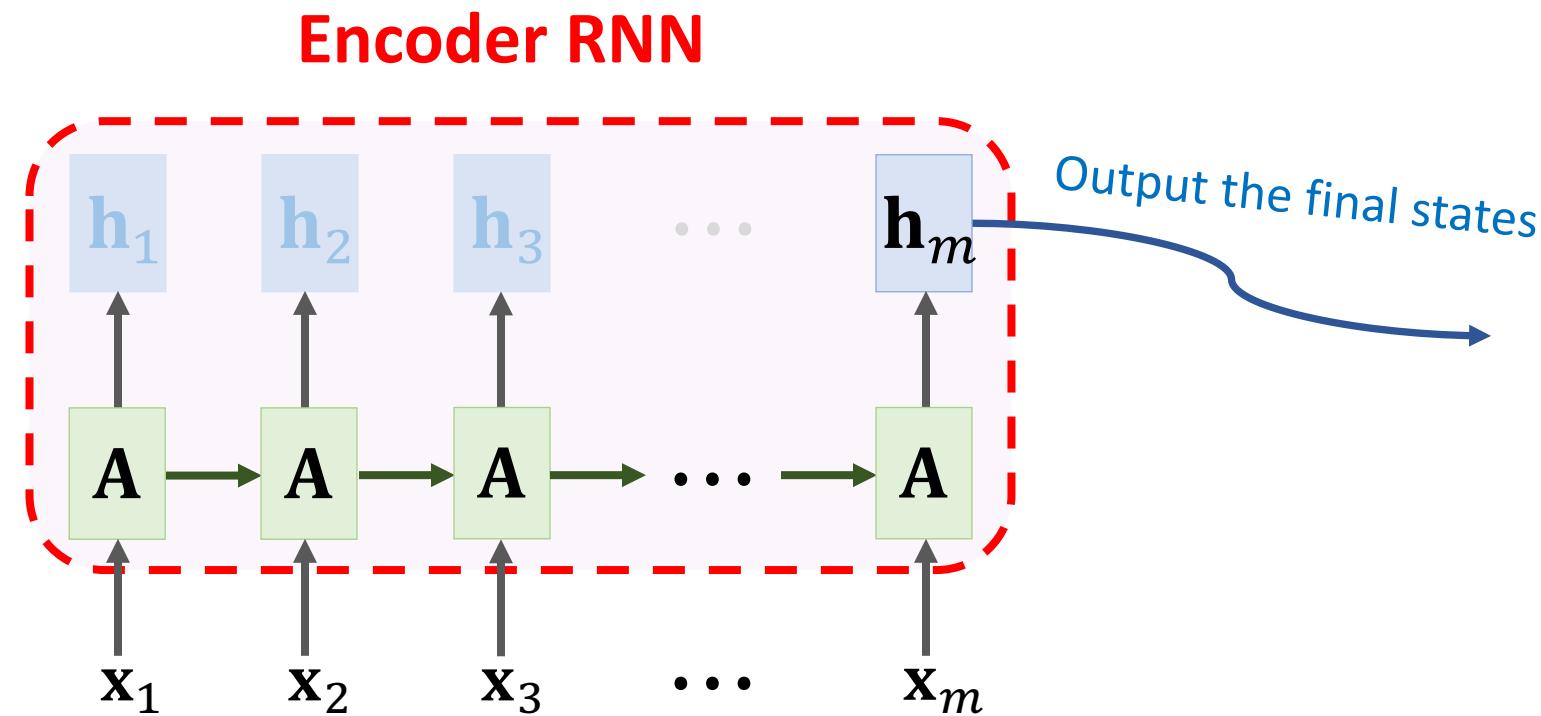
Return: "mach ne fliege"

Summary

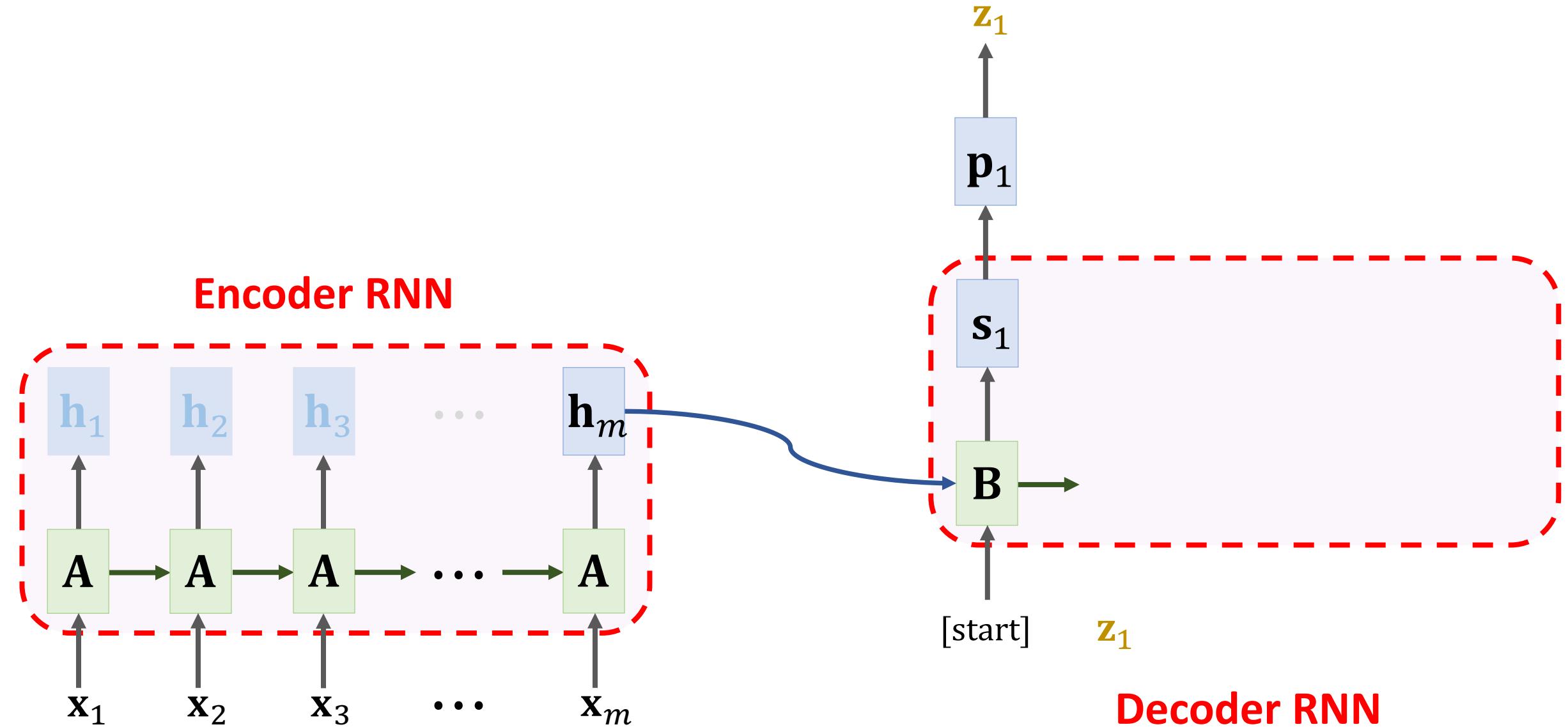
Seq2Seq Model



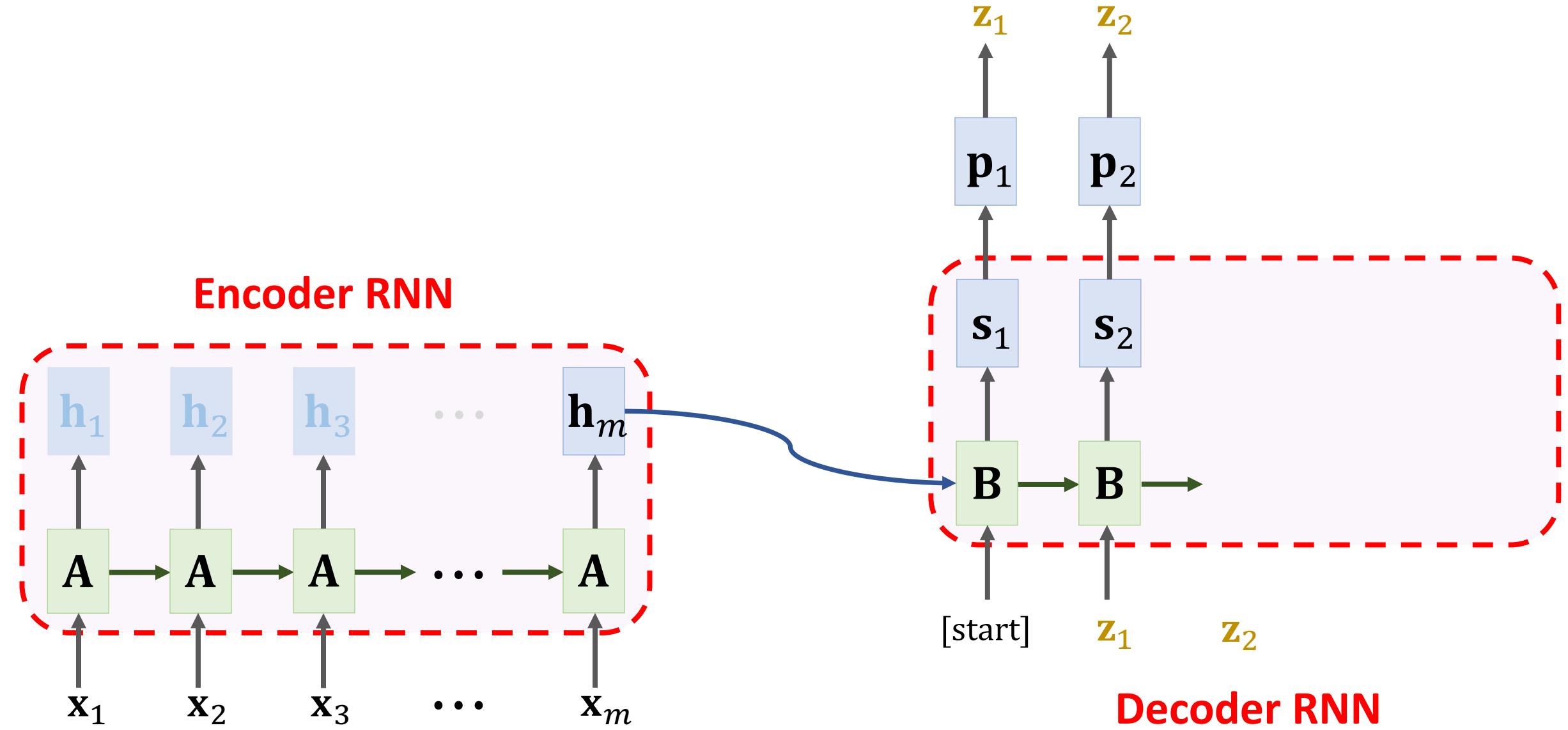
Seq2Seq Model



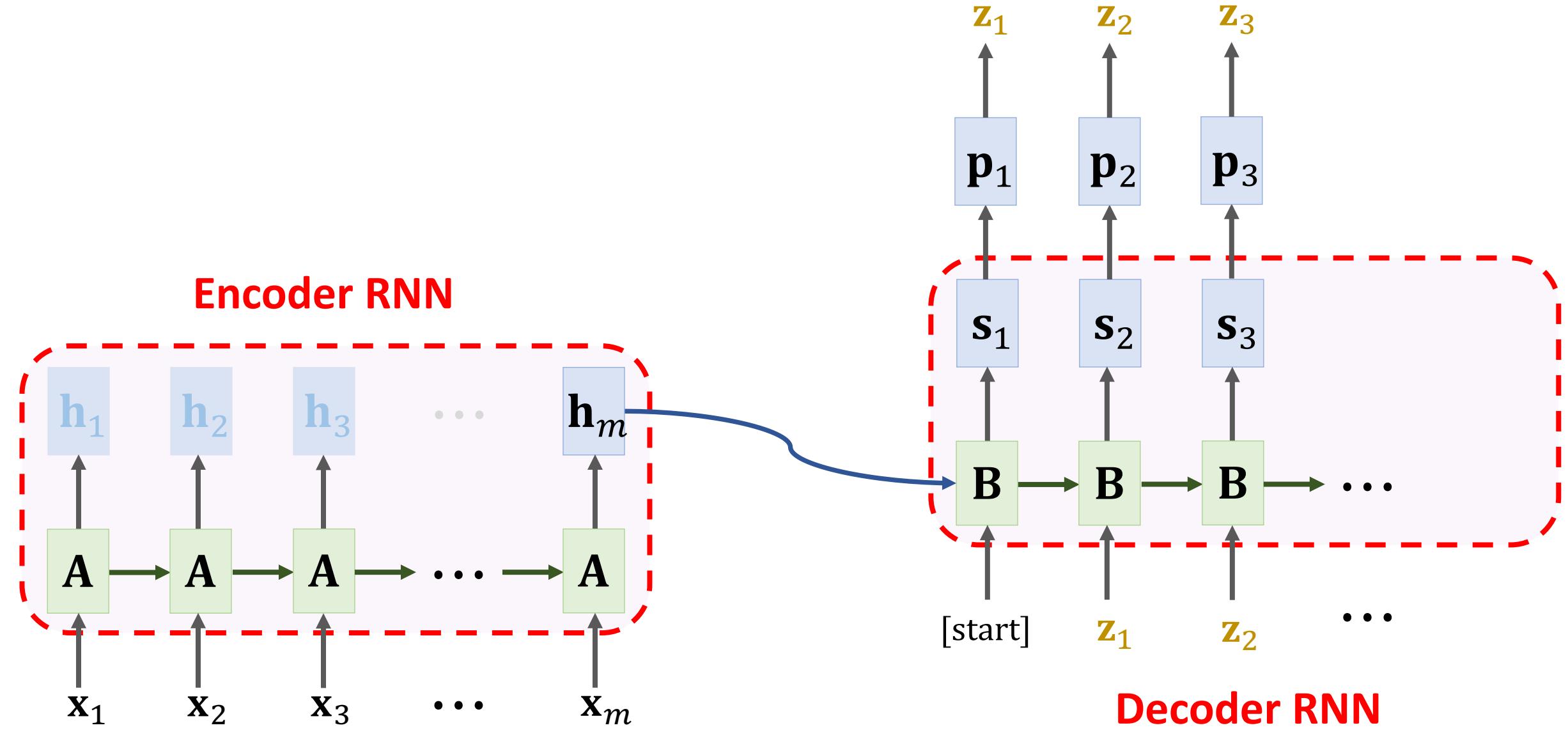
Seq2Seq Model



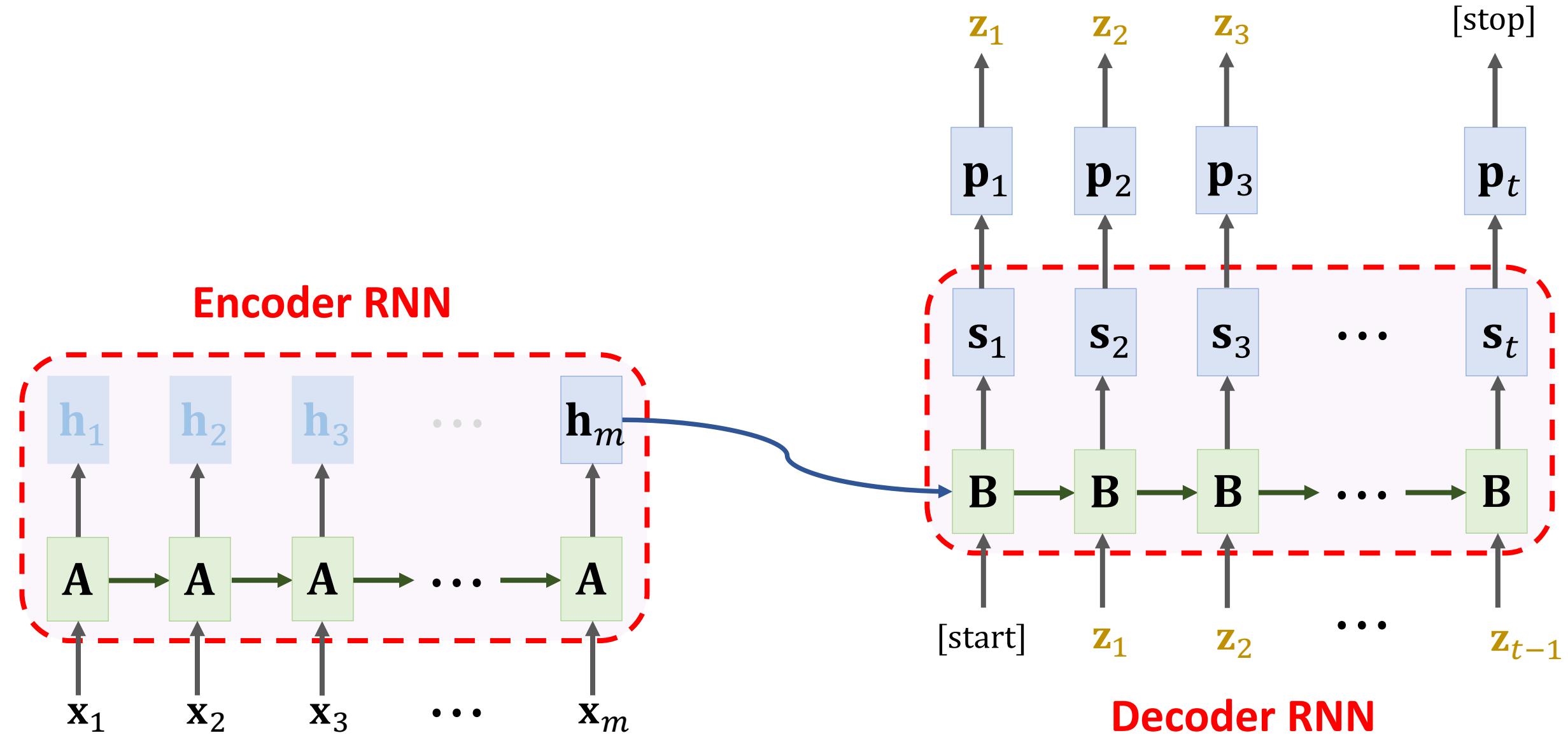
Seq2Seq Model



Seq2Seq Model



Seq2Seq Model



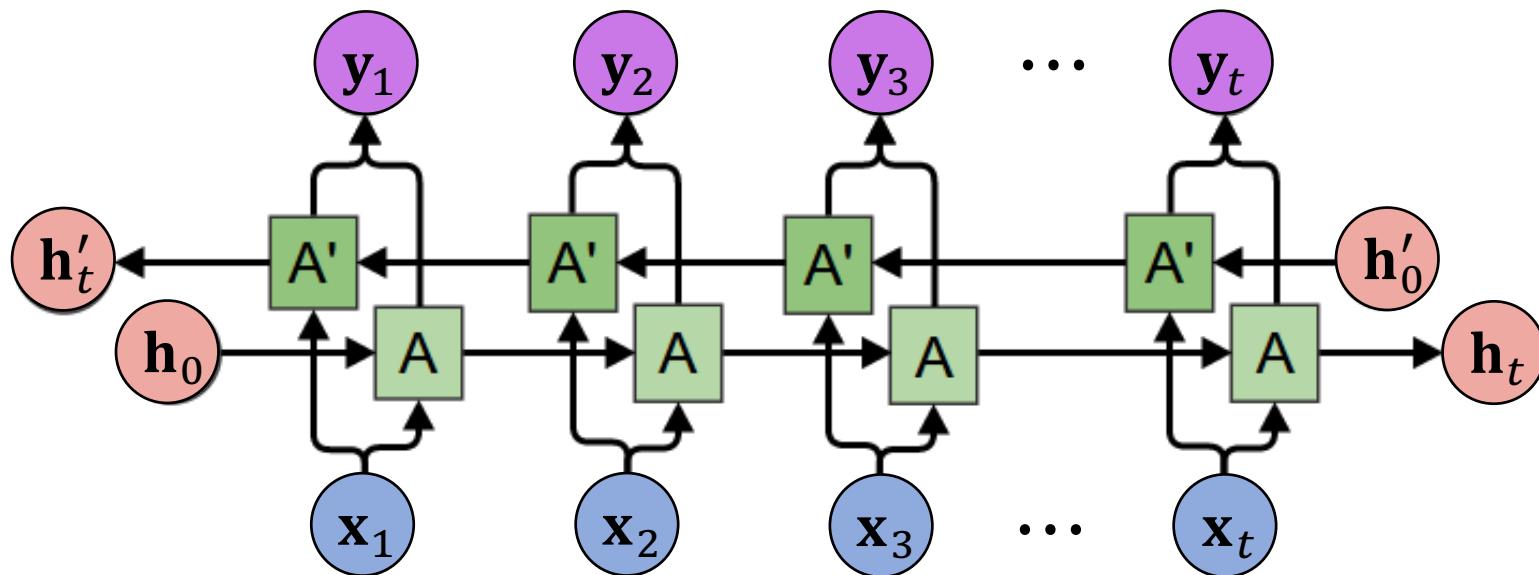
How to Improve?

1. Bi-LSTM instead of LSTM (Encoder only!)

- Encoder's final states (\mathbf{h}_t and \mathbf{c}_t) have all the information of the English sentence.
- If the sentence is long, the final states have forgotten early inputs.

1. Bi-LSTM instead of LSTM (Encoder only!)

- Encoder's final states (\mathbf{h}_t and \mathbf{c}_t) have all the information of the English sentence.
- If the sentence is long, the final states have forgotten early inputs.
- Bi-LSTM (left-to-right and right-to-left) has longer memory.



1. Bi-LSTM instead of LSTM (Encoder only!)

- Encoder's final states (\mathbf{h}_t and \mathbf{c}_t) have all the information of the English sentence.
- If the sentence is long, the final states have forgotten early inputs.
- Bi-LSTM (left-to-right and right-to-left) has longer memory.
- Use Bi-LSTM in the encoder; use unidirectional LSTM in the decoder.

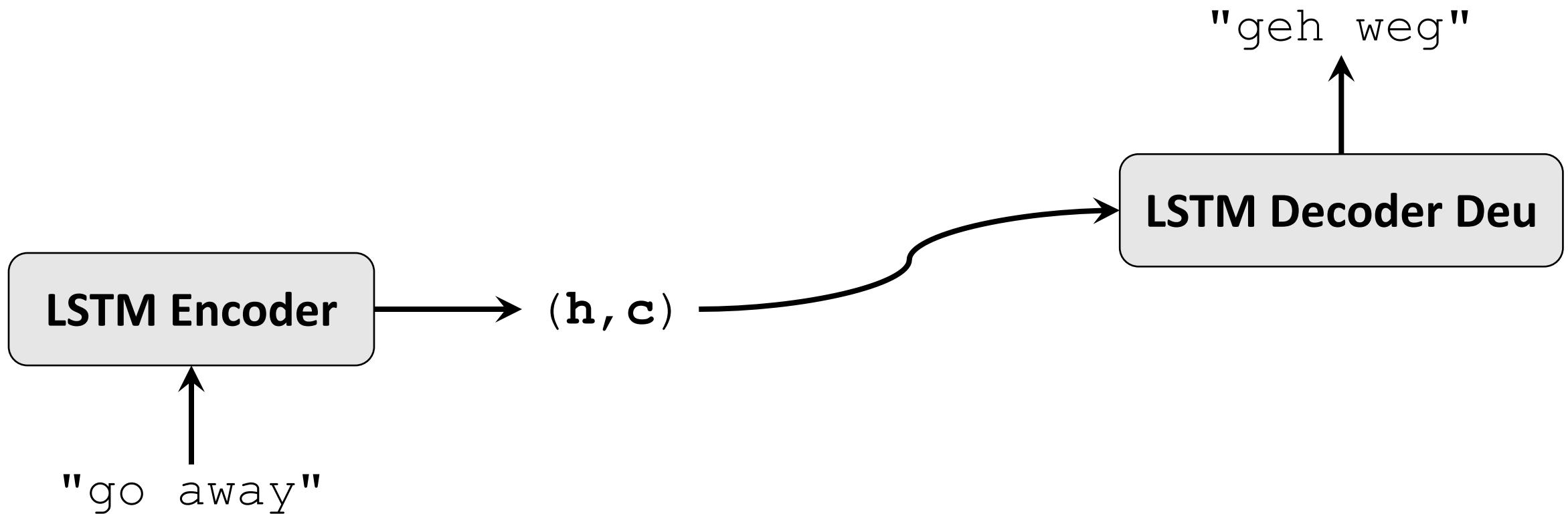
2. Word-Level Tokenization

- Word-level tokenization instead of char-level.
 - The average length of English words is **4.5** letters.
 - The sequences will be **4.5x** shorter.
 - Shorter sequence → less likely to forget.

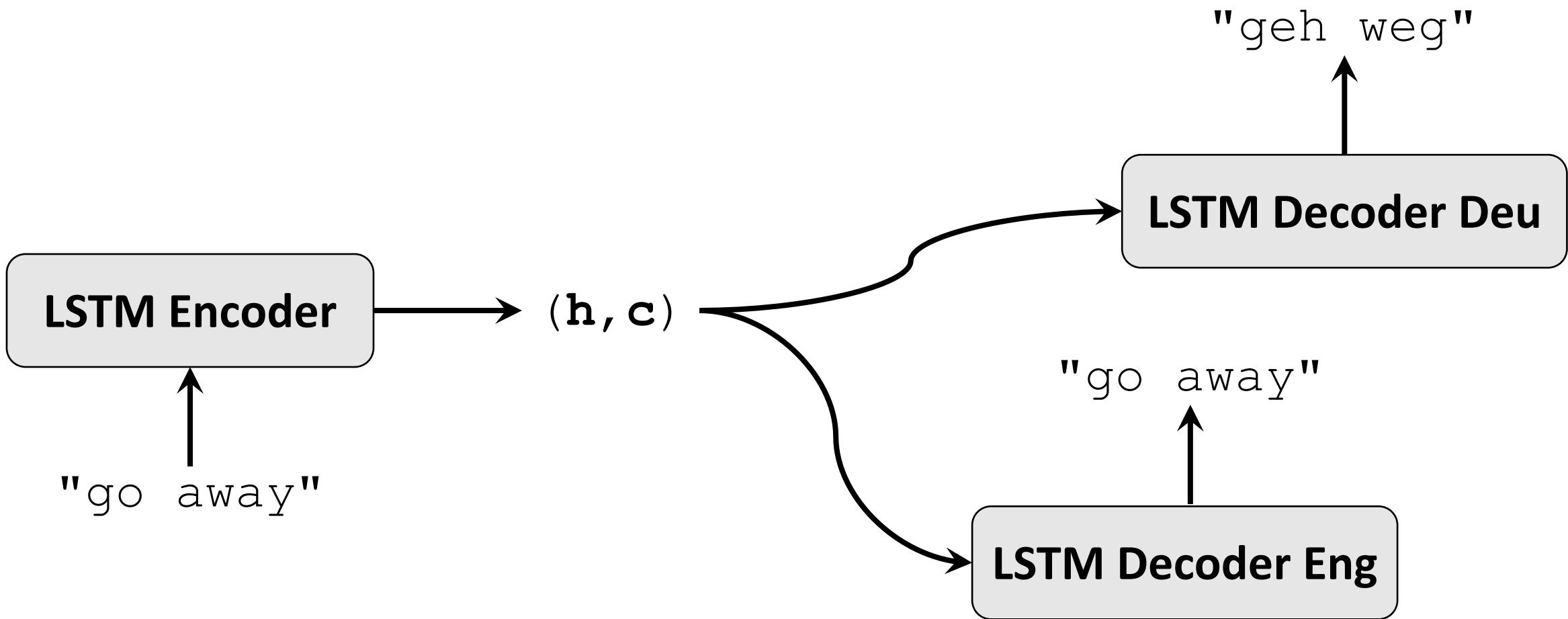
2. Word-Level Tokenization

- Word-level tokenization instead of char-level.
 - The average length of English words is **4.5** letters.
 - The sequences will be **4.5x** shorter.
 - Shorter sequence → less likely to forget.
- But you will need a large dataset!
 - # of (frequently used) chars is $\sim 10^2$ → one-hot suffices.
 - # of (frequently used) words is $\sim 10^4$ → must use embedding.
 - Embedding Layer has many parameters → overfitting!

3. Multi-Task Learning



3. Multi-Task Learning

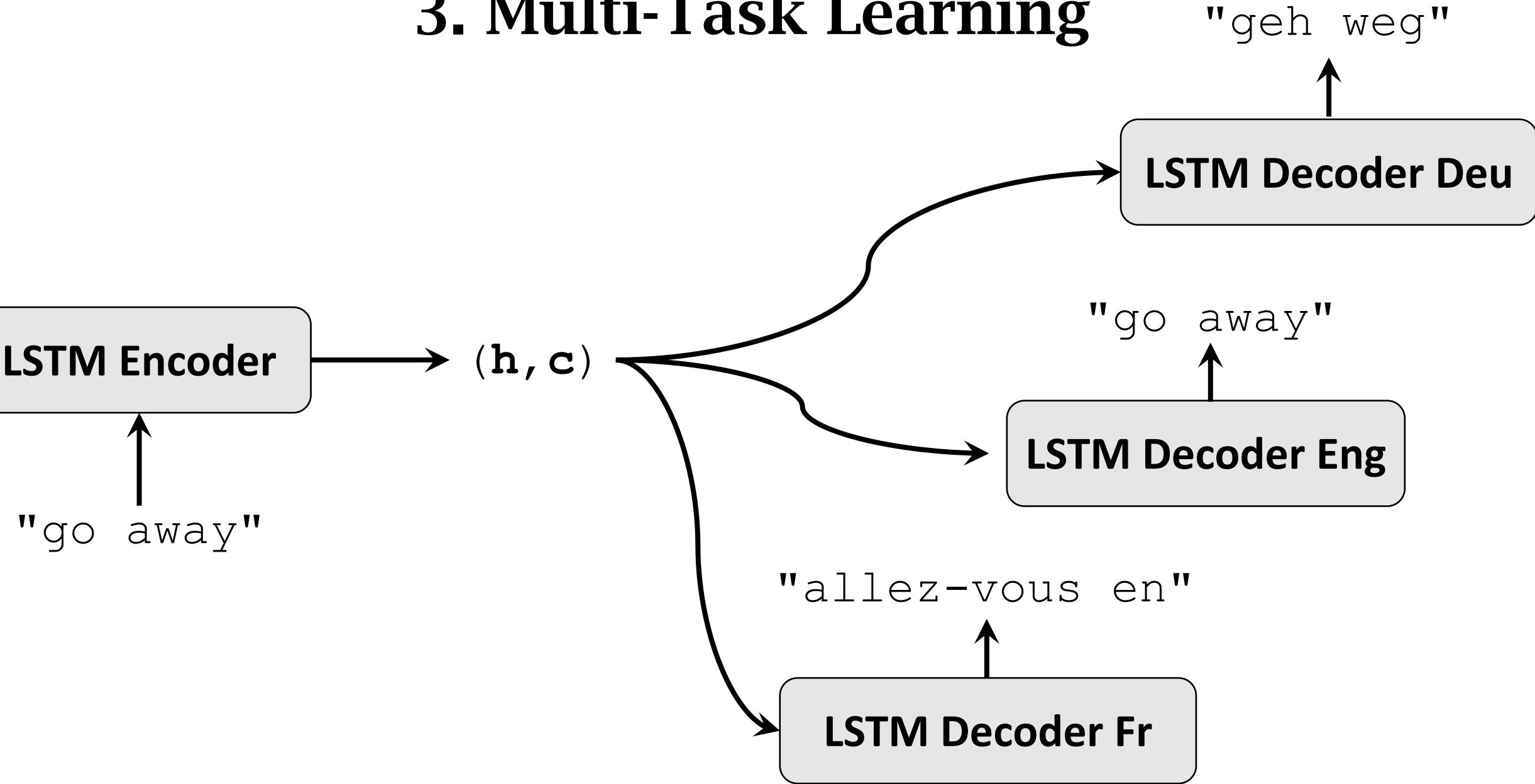


3. Multi-Task Learning

- Even if you want to **translate English to German**, you can use all the datasets:

-  Afrikaans - English [afr-eng.zip](#) (725)
-  Aklanon - English [akl-eng.zip](#) (22)
-  Albanian - English [sqi-eng.zip](#) (412)
-  Algerian Arabic - English [arq-eng.zip](#) (156)
-  Arabic - English [ara-eng.zip](#) (11009)
-  Arabic (Gulf) - English [afb-eng.zip](#) (28)
-  Assamese - English [asm-eng.zip](#) (23)
-  Asturian - English [ast-eng.zip](#) (23)
-  Azerbaijani - English [aze-eng.zip](#) (2131)
-  Basque - English [eus-eng.zip](#) (667)
-  Belarusian - English [bel-eng.zip](#) (2698)
-  Bengali - English [ben-eng.zip](#) (4399)
-  Berber - English [ber-eng.zip](#) (54988)
-  Bulgarian - English [bul-eng.zip](#) (14968)
-  Spanish - English [spa-eng.zip](#) (120799)
-  Swedish - English [swe-eng.zip](#) (17409)
-  Tagalog - English [tgl-eng.zip](#) (3144)
-  Tamil - English [tam-eng.zip](#) (197)
-  Tatar - English [tat-eng.zip](#) (529)
-  Telugu - English [tel-eng.zip](#) (138)
-  Thai - English [tha-eng.zip](#) (110)
-  Turkish - English [tur-eng.zip](#) (497249)
-  Ukrainian - English [ukr-eng.zip](#) (113396)
-  Urdu - English [urd-eng.zip](#) (1180)
-  Uyghur - English [uig-eng.zip](#) (285)
-  Vietnamese - English [vie-eng.zip](#) (3413)
-  Waray - English [war-eng.zip](#) (1181)
-  Zaza - English [zza-eng.zip](#) (345)

3. Multi-Task Learning



How to Improve?

1. Bi-LSTM instead of LSTM. (Encoder only!)
2. Tokenization in the word-level (instead of char-level.)
3. Multi-task learning.
4. **Attention! (Next lecture.)**

Thank you!