Neural Machine Translation

Shusen Wang

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/

Datasets

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

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 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 Karagana, 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 start sign
- '\n' => 2 stop sign
- 'a' => 3
- 'b' => 4
- 'c' => 5
- 'd' => 6
- …
- 'z' => 28
- '_' => 29

2. One-Hot Encoding

```
"I_am_okay."
```

Eng_Tokenizer

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

Encoding using Eng_Dictionary

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

2. One-Hot Encoding

"Es geht mir gut”

Deu_Tokenizer

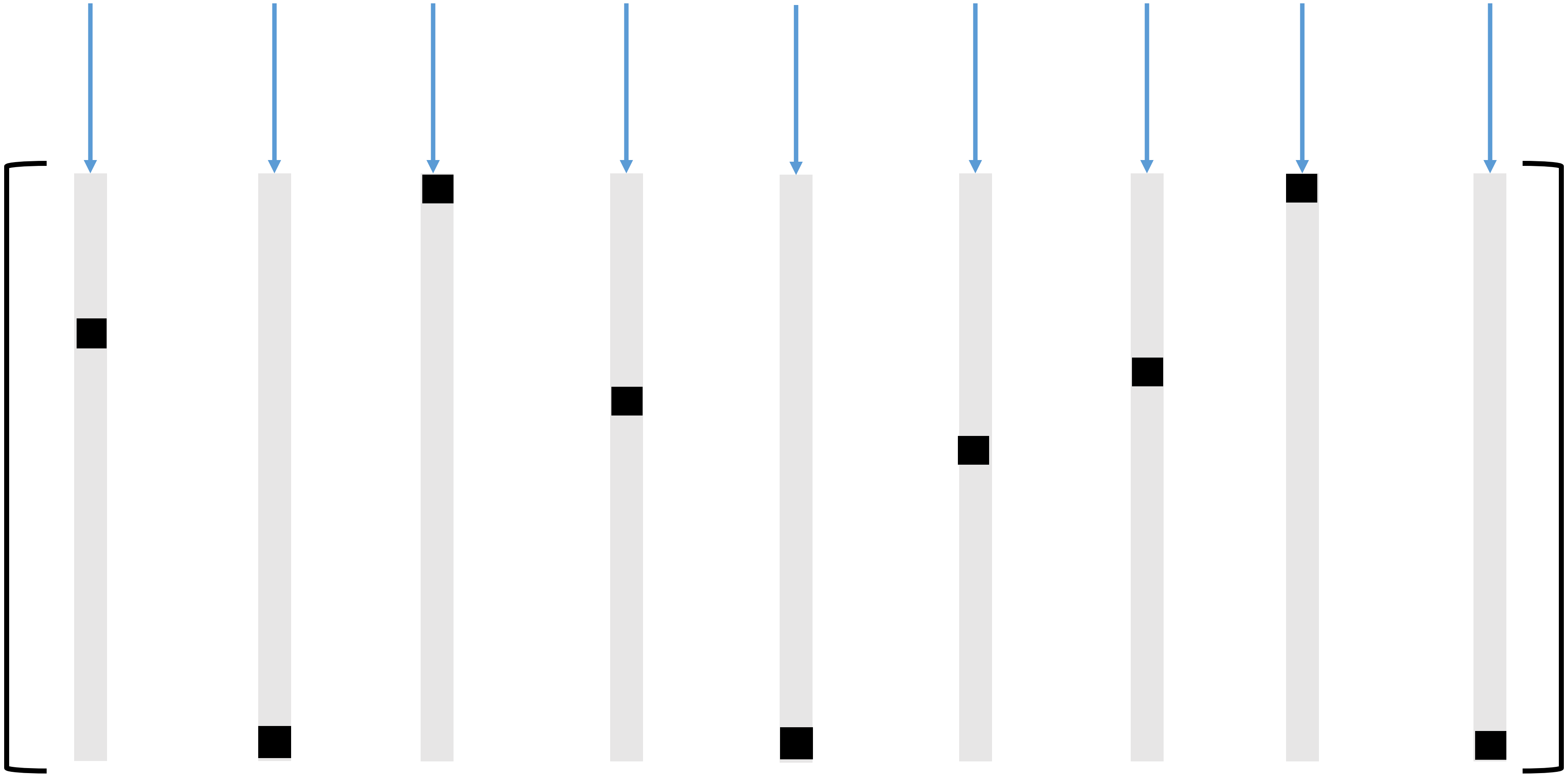
['e', 's', '_', 'g', 'e', ..., 'g', 'u', 't']

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]
```



Why not using embedding?

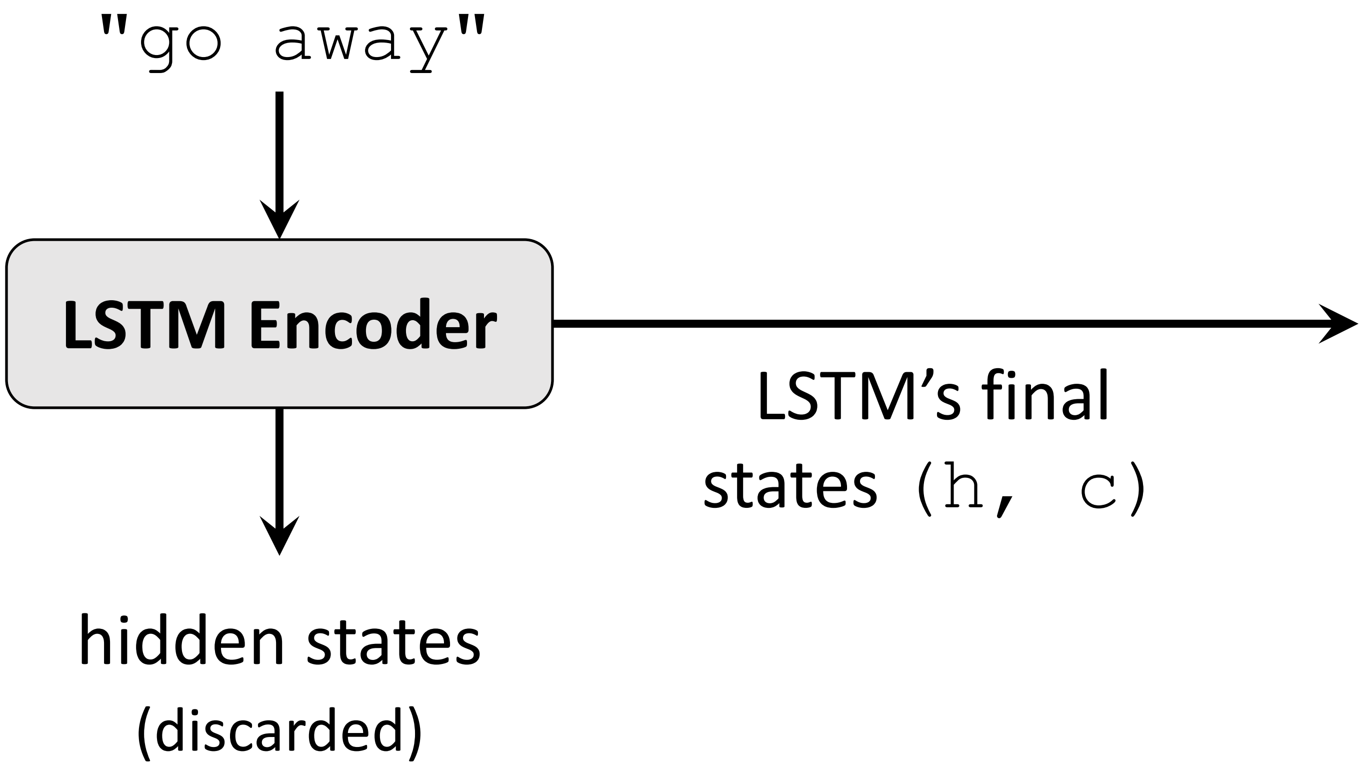
- There are around 100 frequent characters. (Vocabulary $\approx$ 100.)
- One-hot converts a character to 100-dim vector.

- There are around $10K$ frequent English words. (Vocabulary $\approx 10K$.)
- One-hot converts a word to $10K$-dim vector. (Too big.)

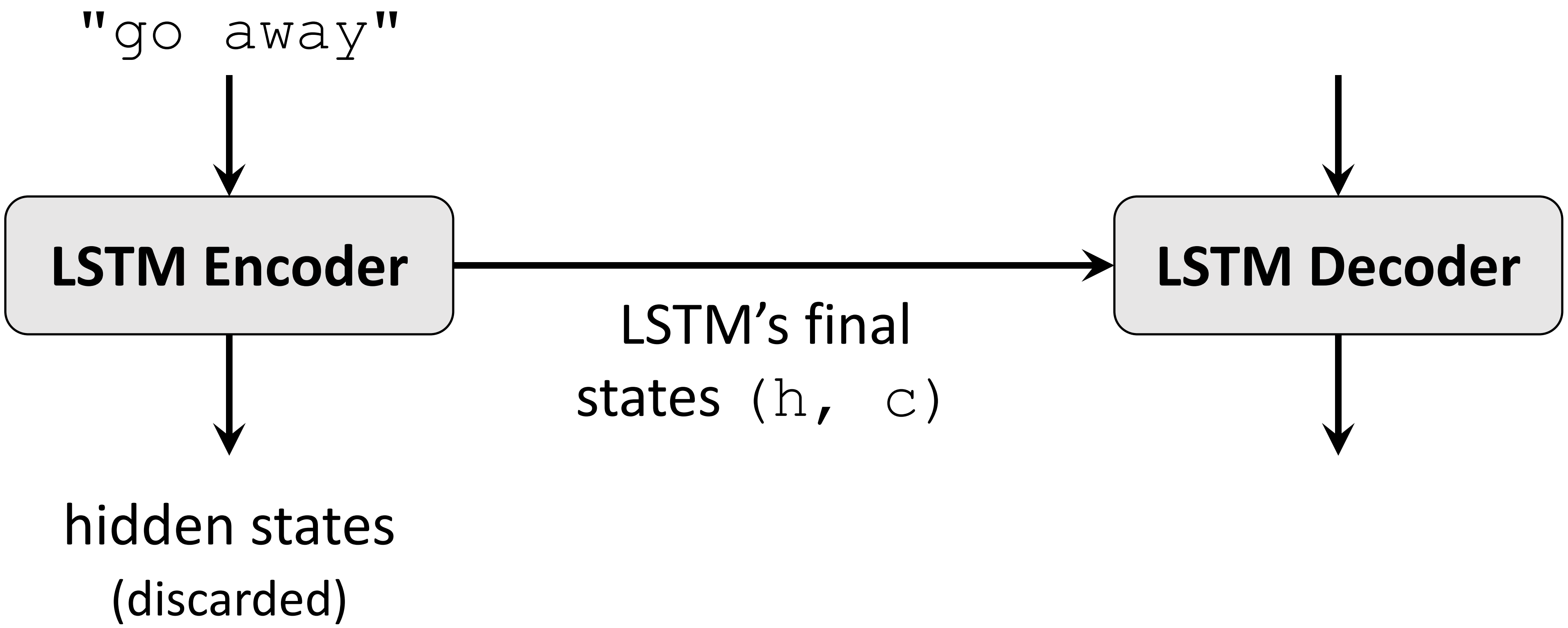
- Conclusion:
  - For char-level tokenization, do not use embedding layer.
  - For word-level tokenization, use an embedding layer.

Training the Seq2Seq Model

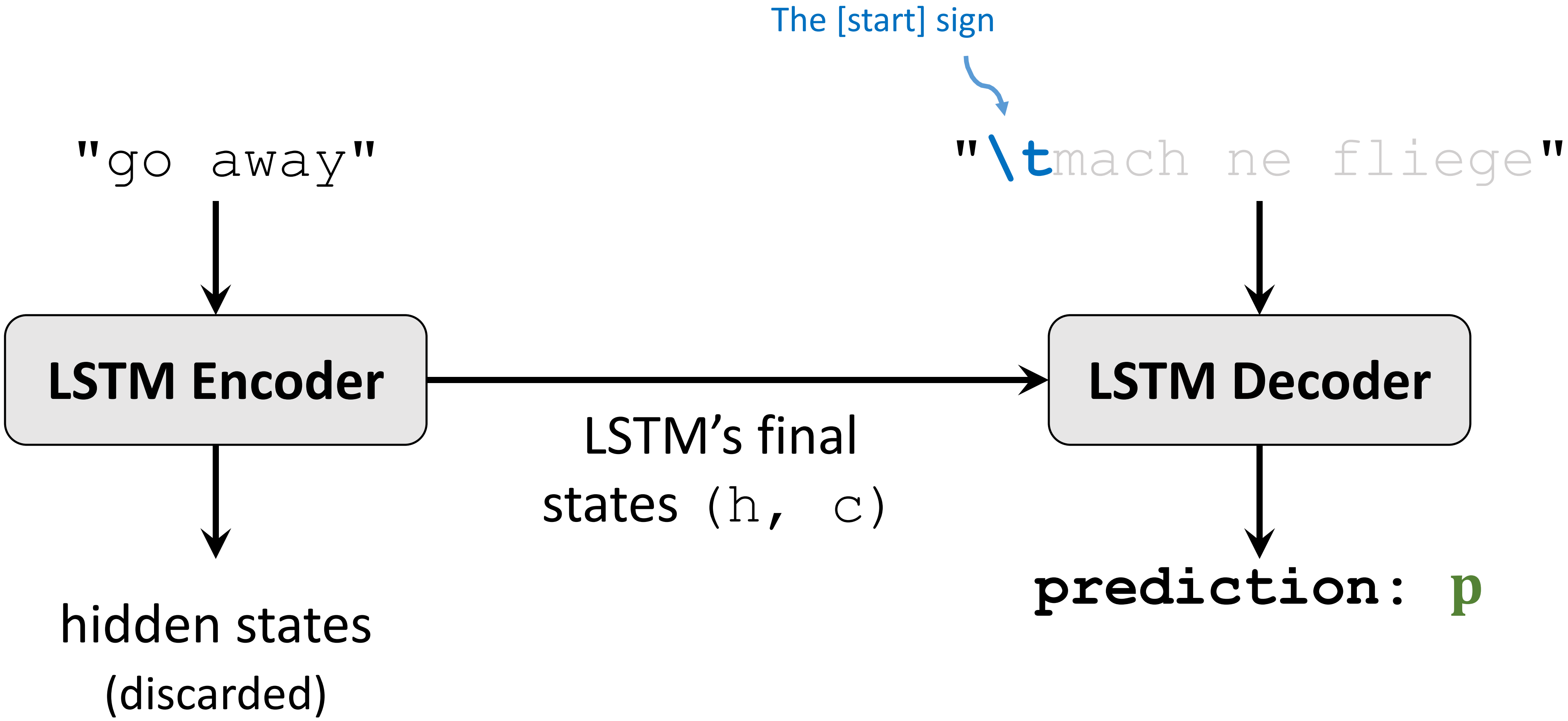
Training of the Seq2Seq Model



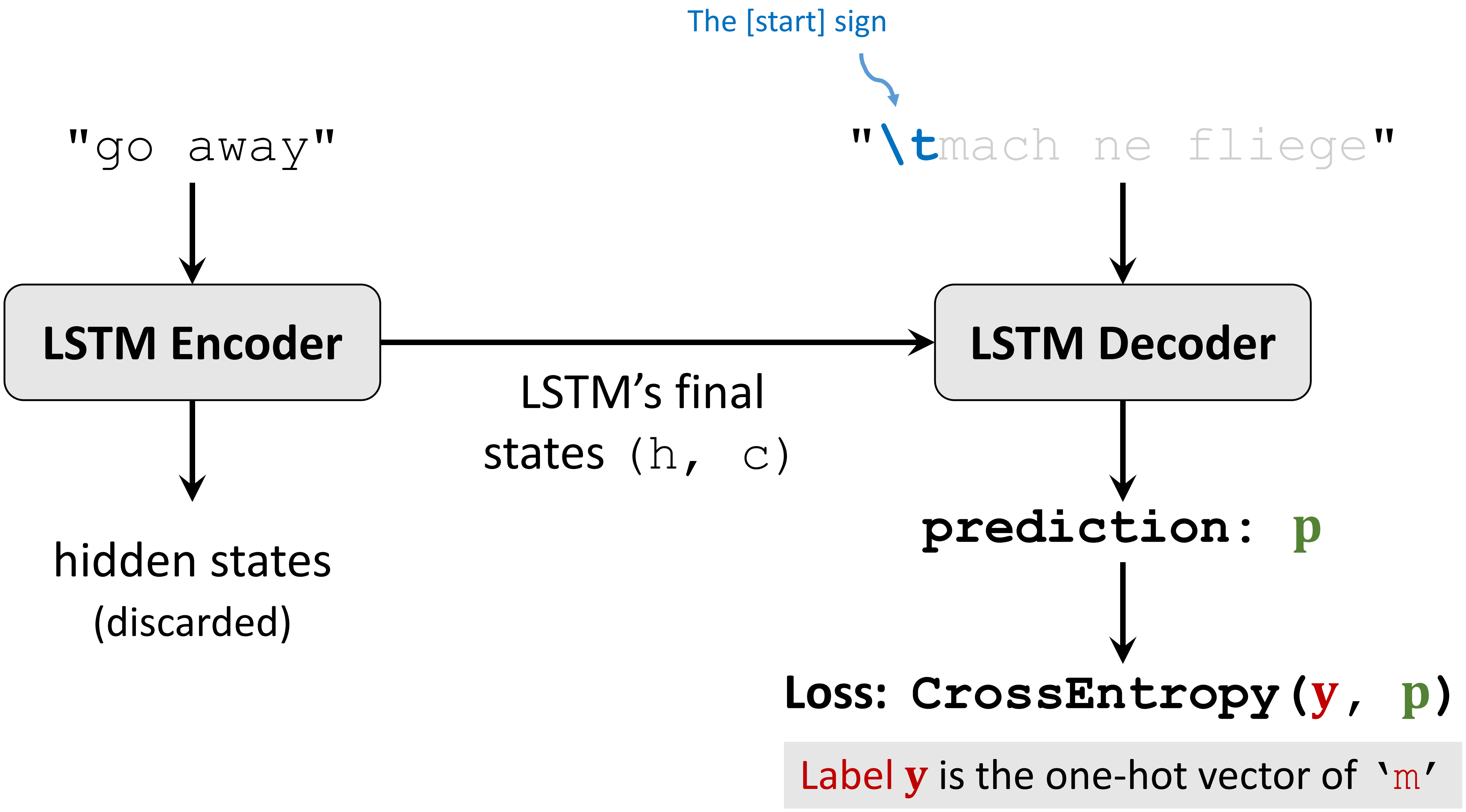
Training of the Seq2Seq Model



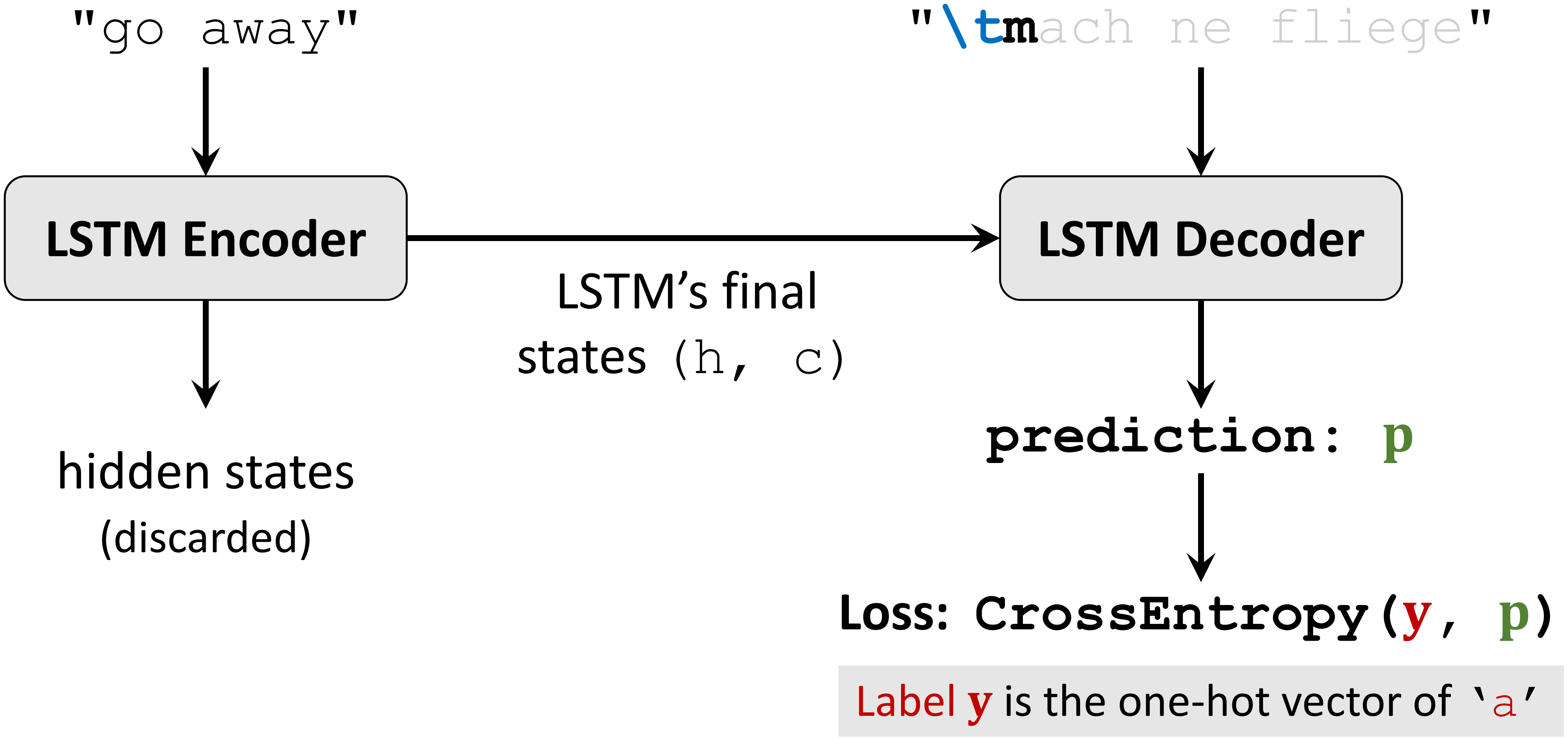
Training of the Seq2Seq Model



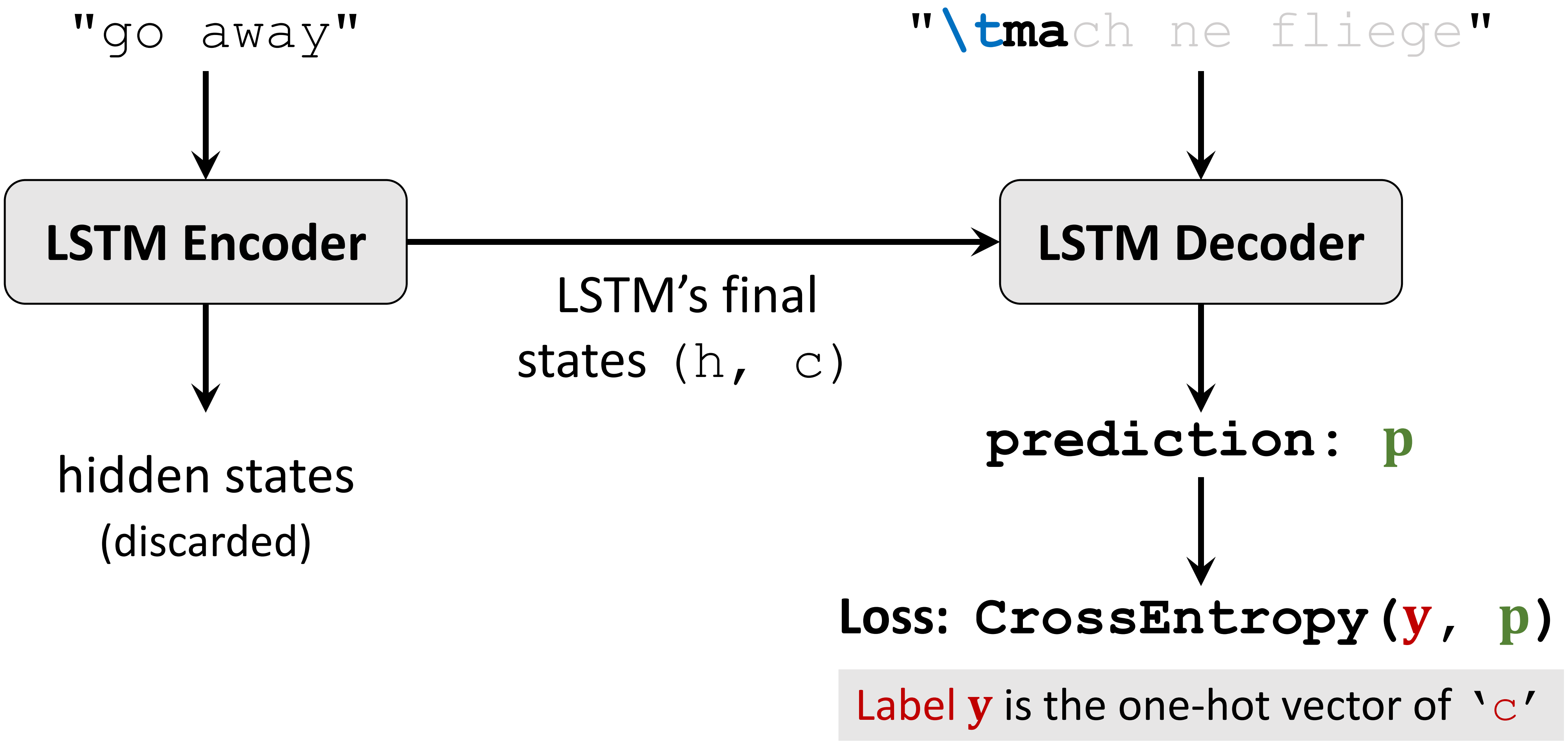
Training of the Seq2Seq Model



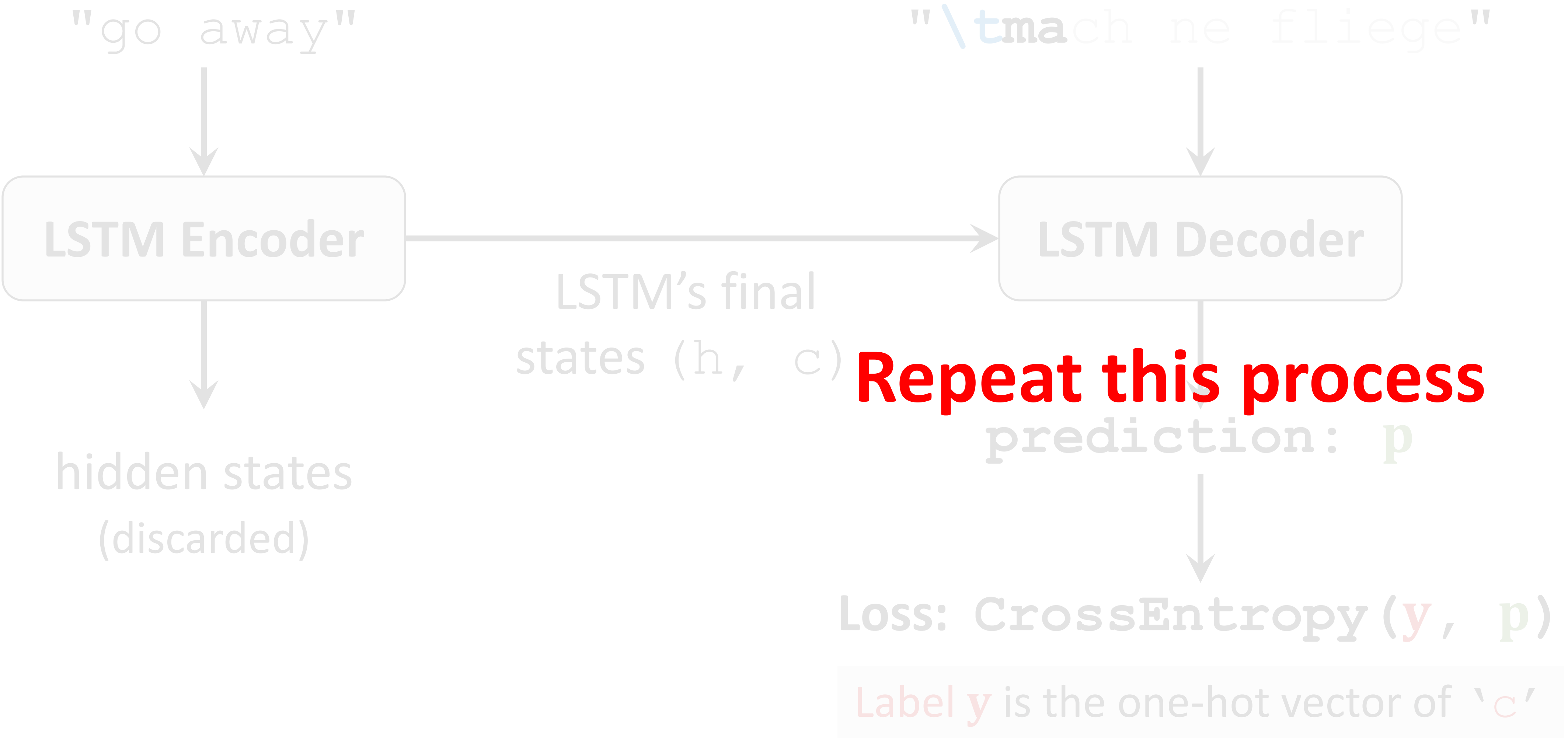
Training of the Seq2Seq Model



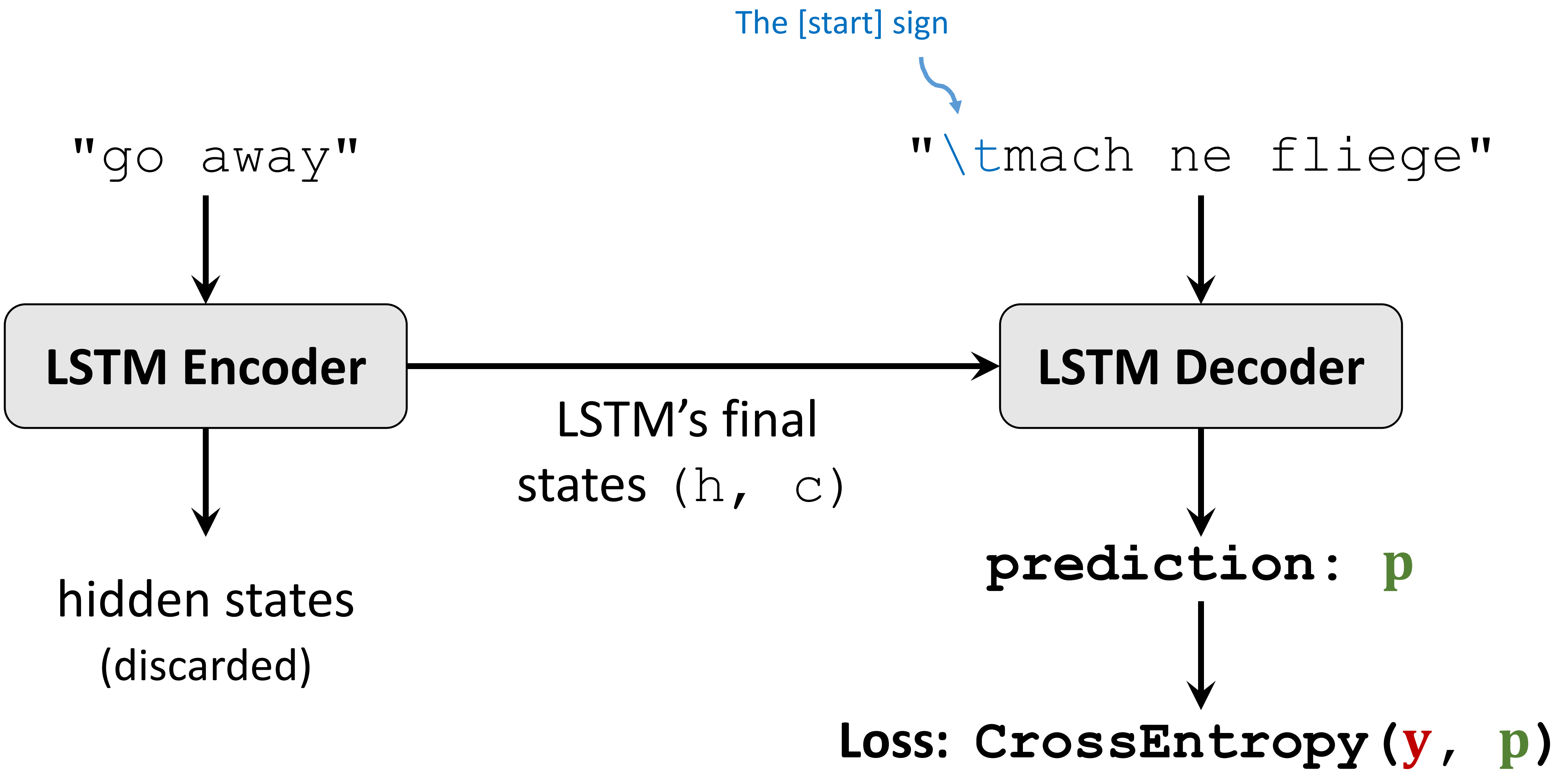
Training of the Seq2Seq Model



Training of the Seq2Seq Model

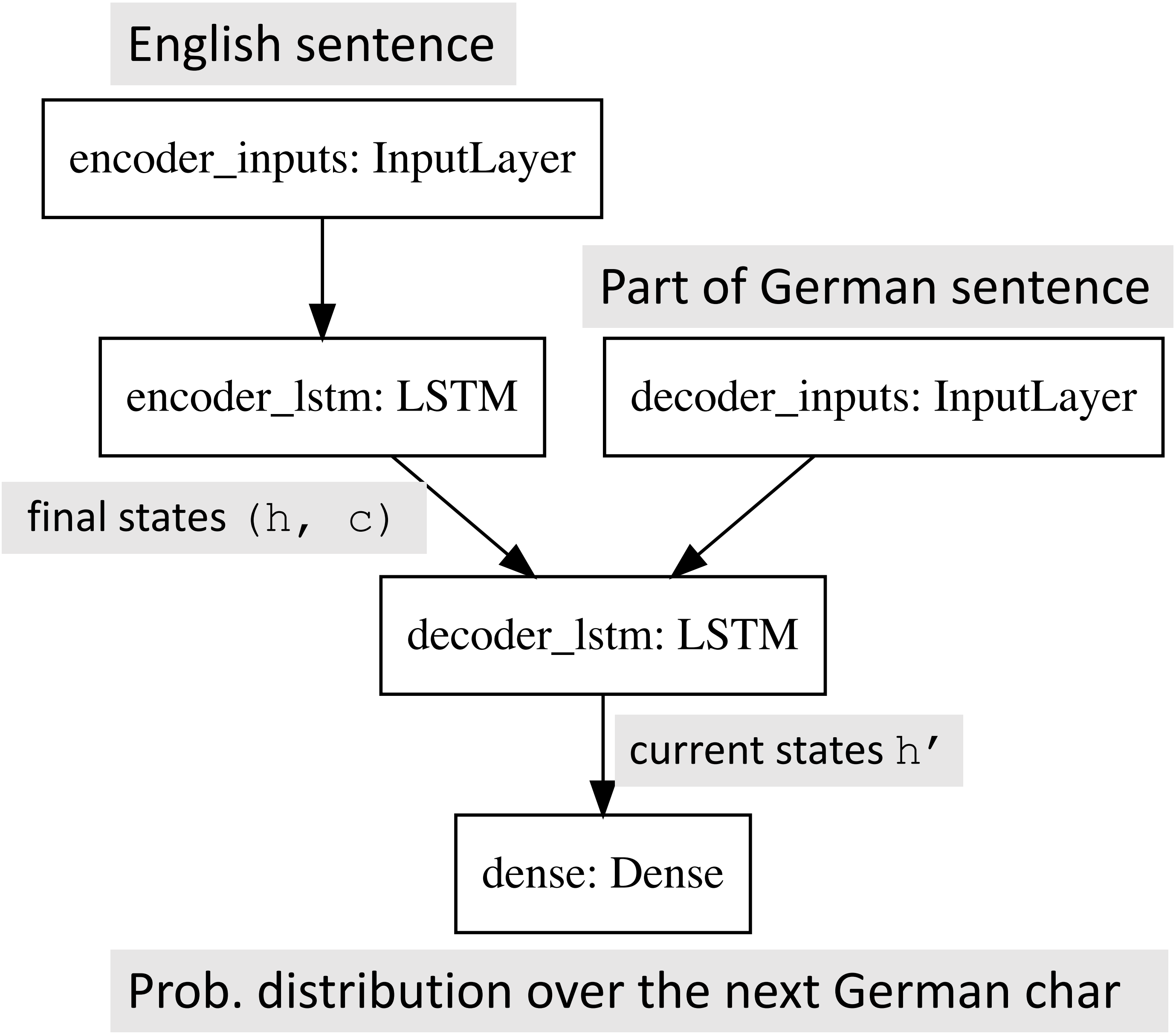


Training of the Seq2Seq Model



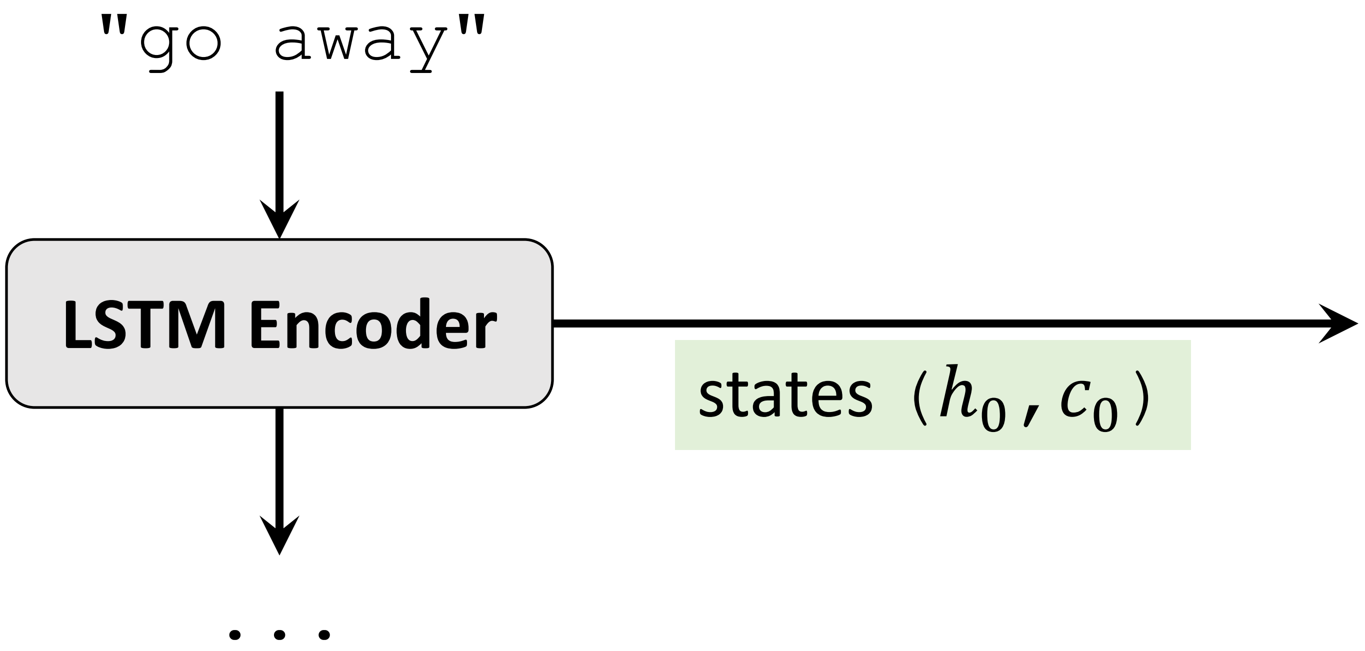
Label **y** is the one-hot vector of the [stop] sign.

Seq2Seq Model in Keras

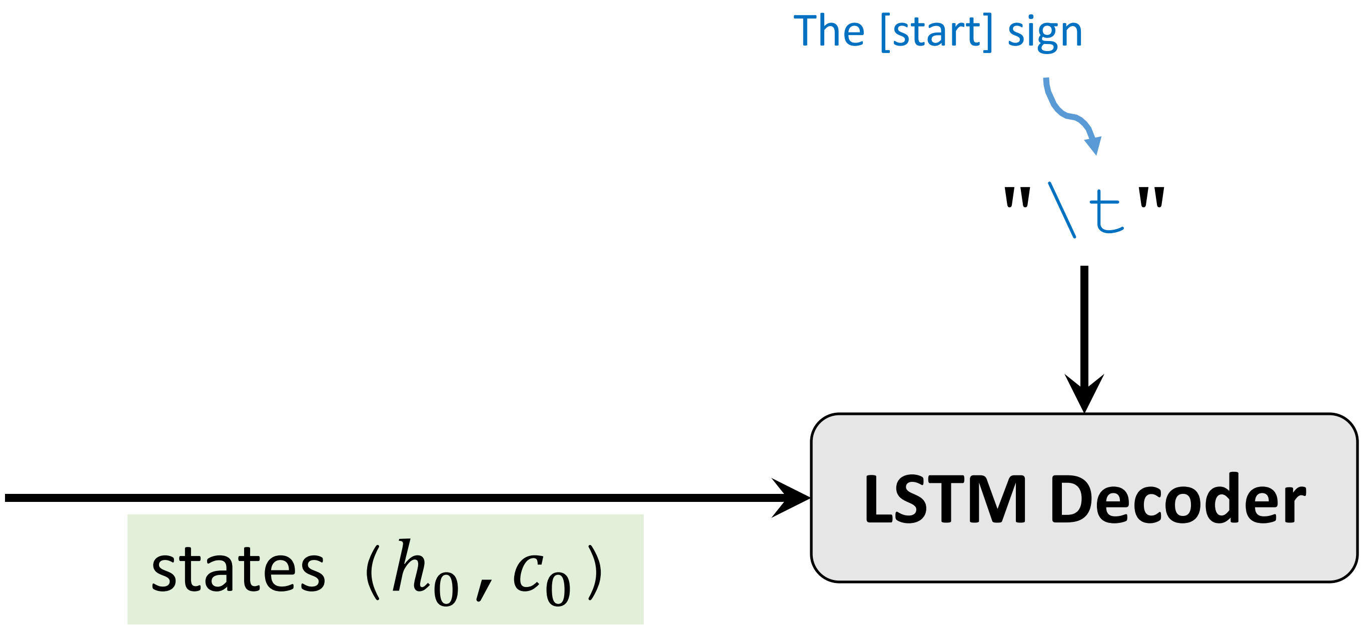


Inference Using the Seq2Seq Model

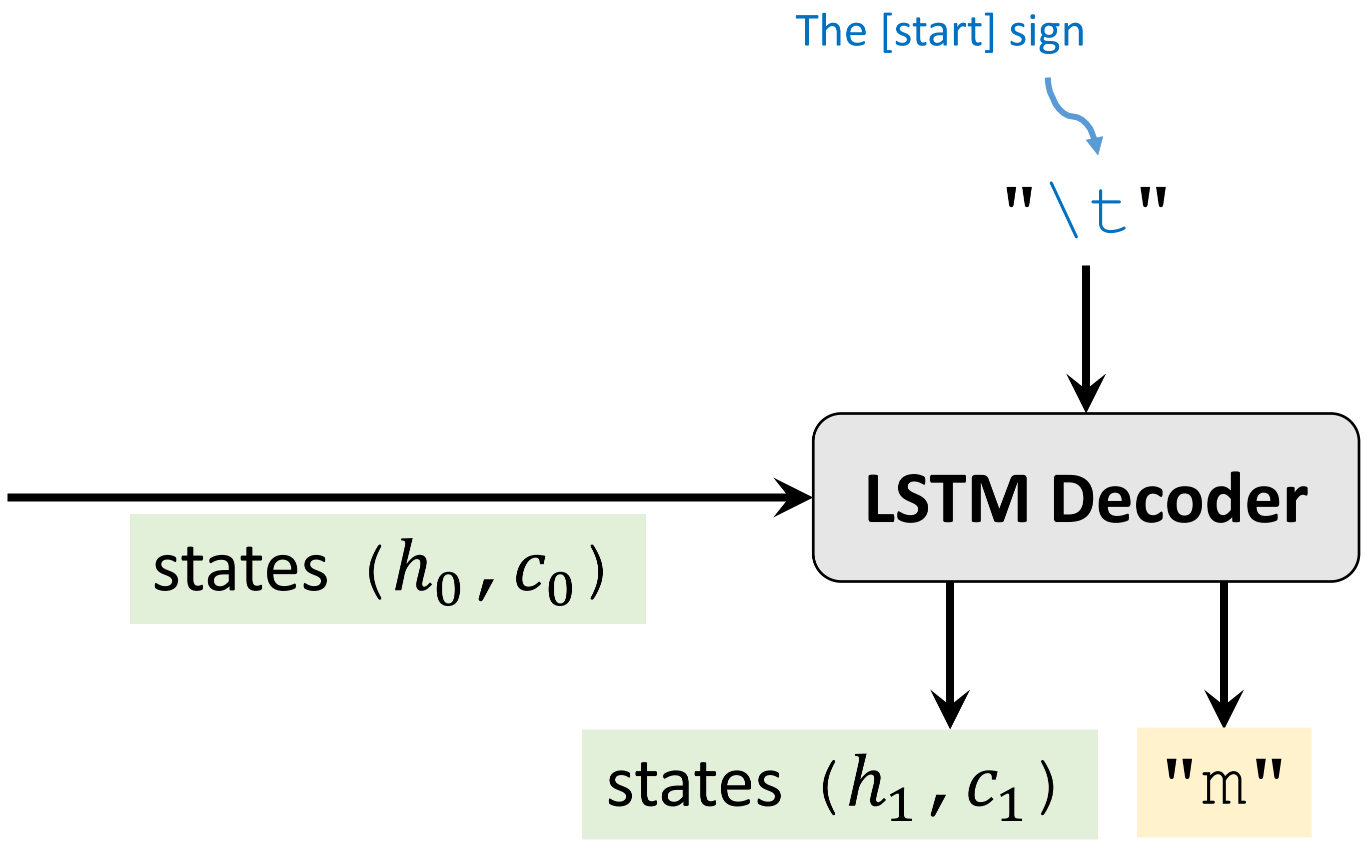
Inference



Inference

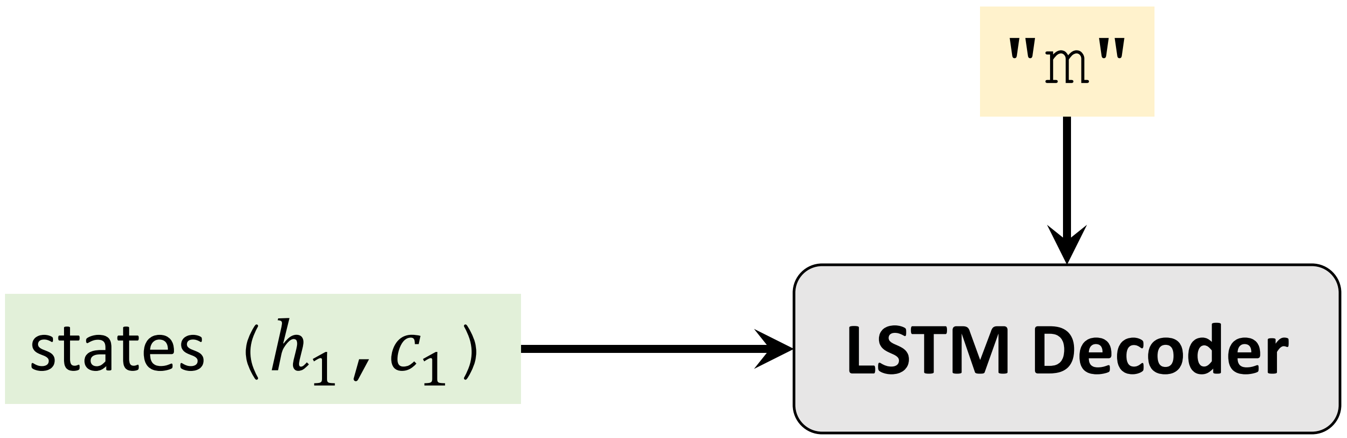


Inference



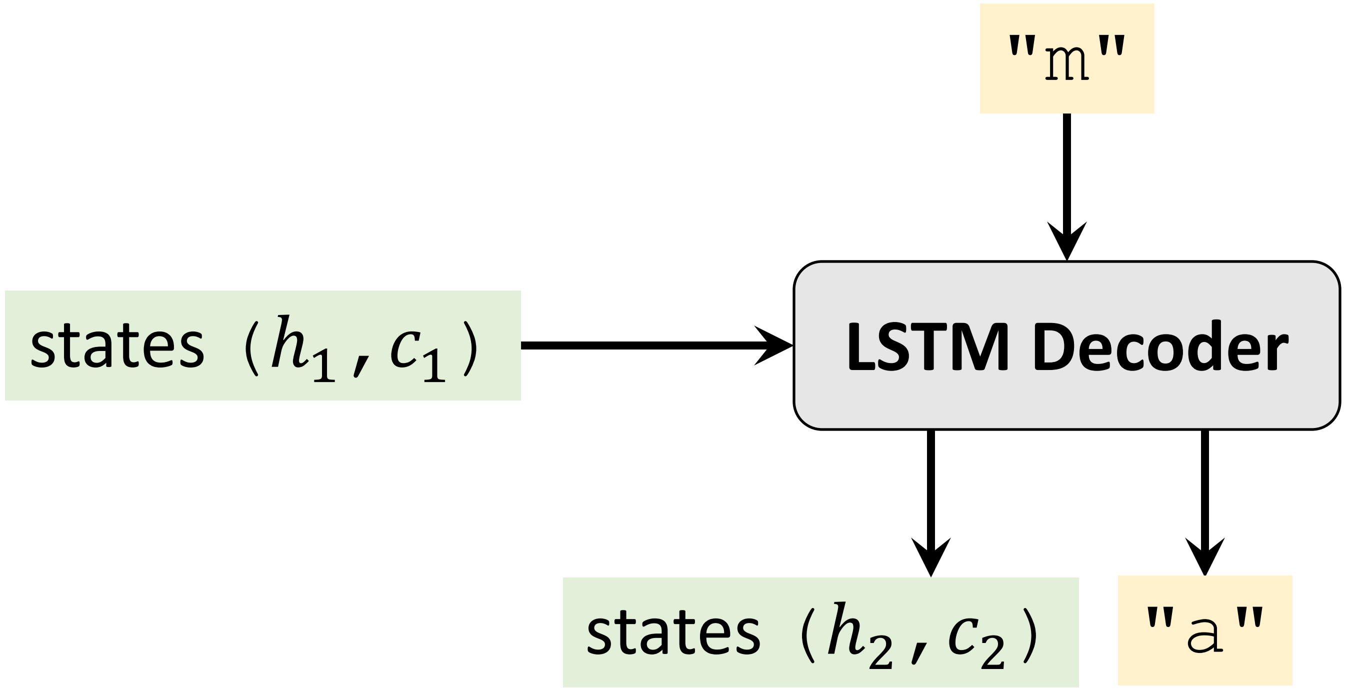
Record: "m"

Inference



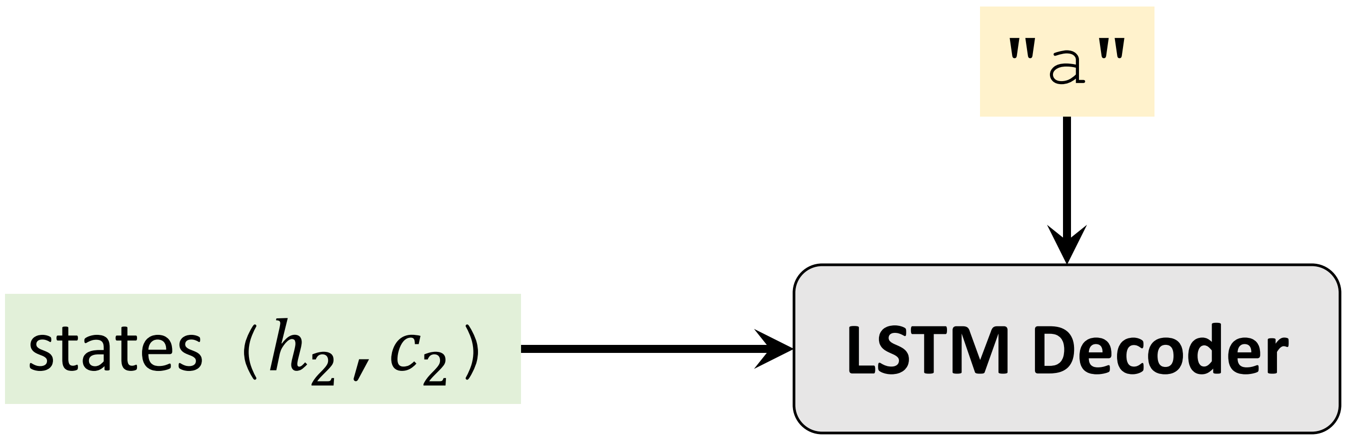
Record: "m"

Inference



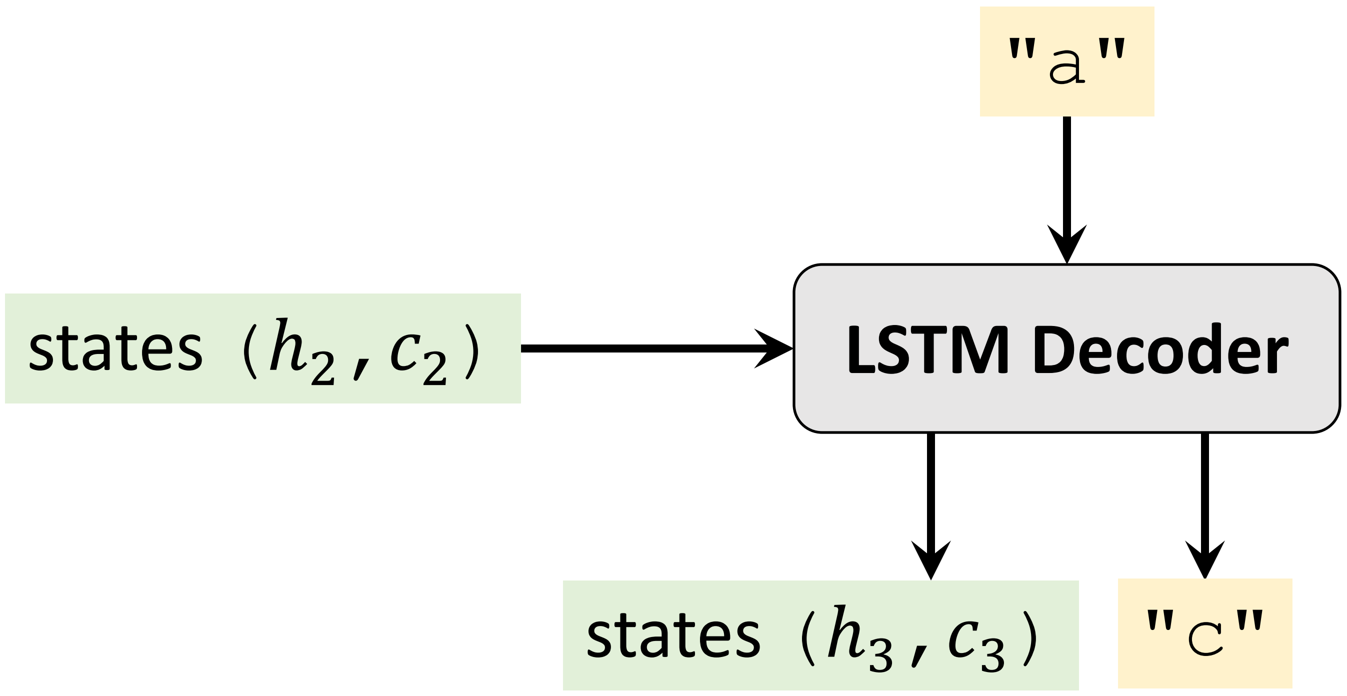
Record: "ma"

Inference



Record: "ma"

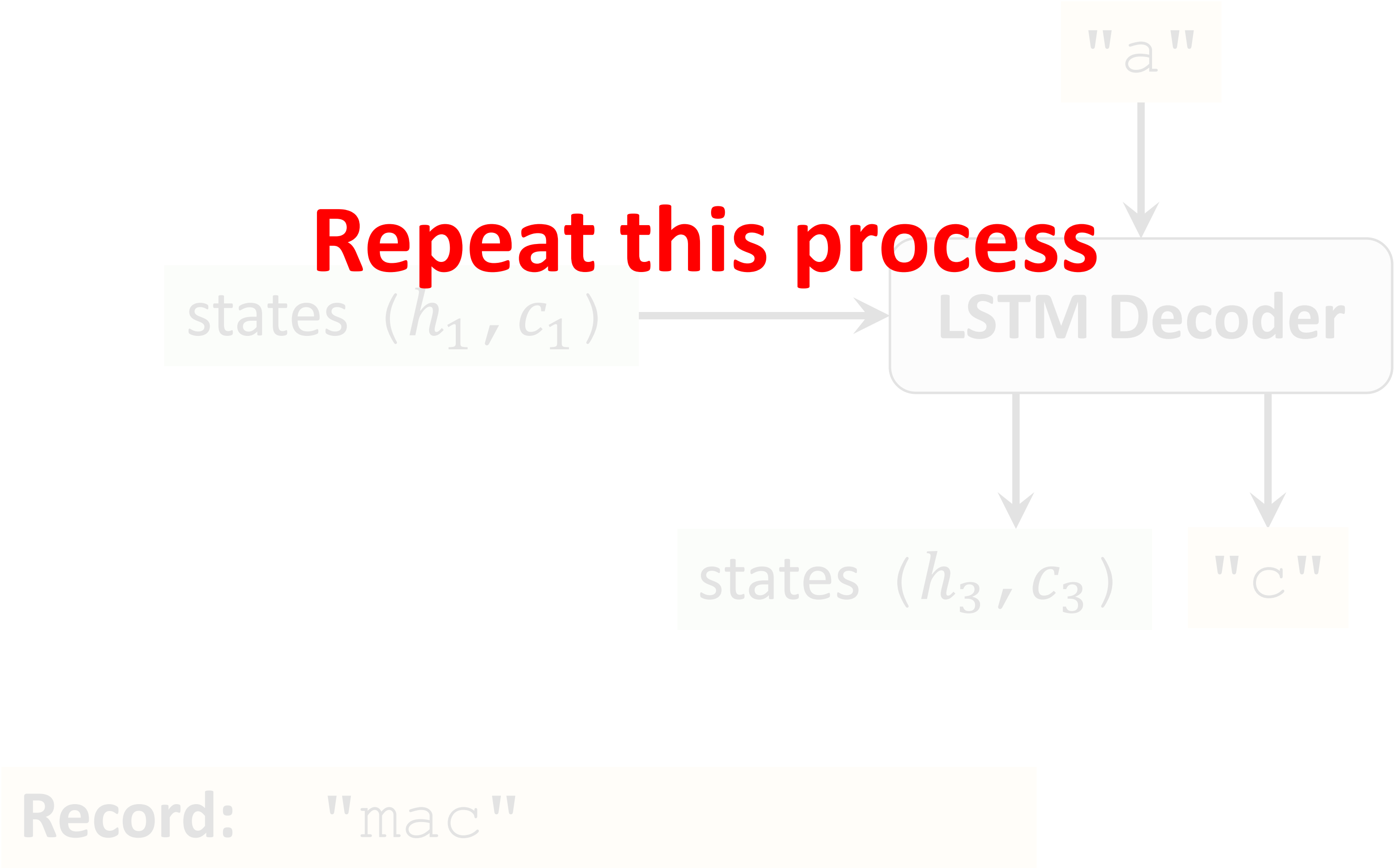
Inference



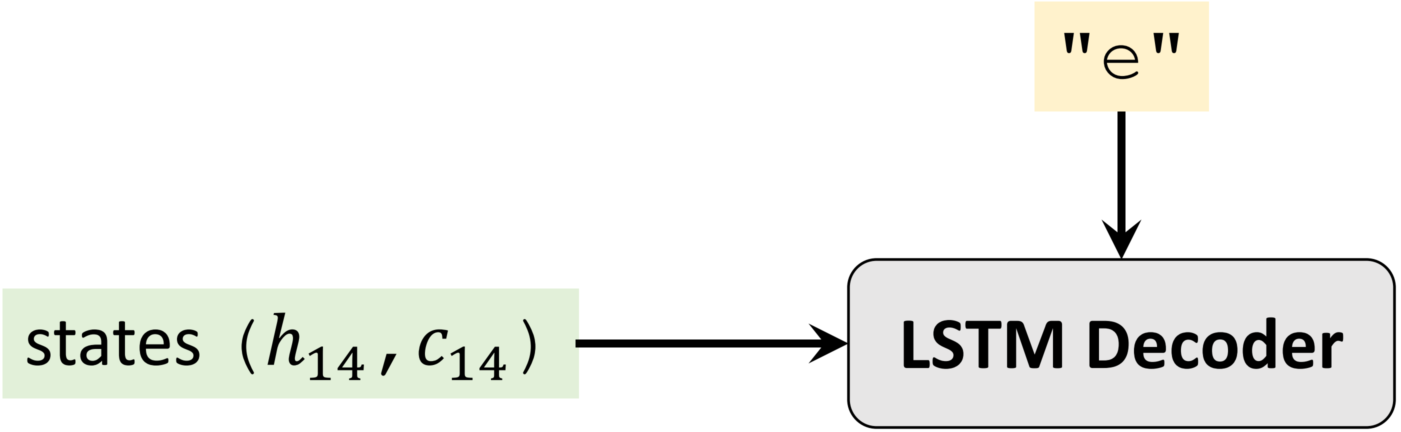
Record: "mac"

Inference

Repeat this process

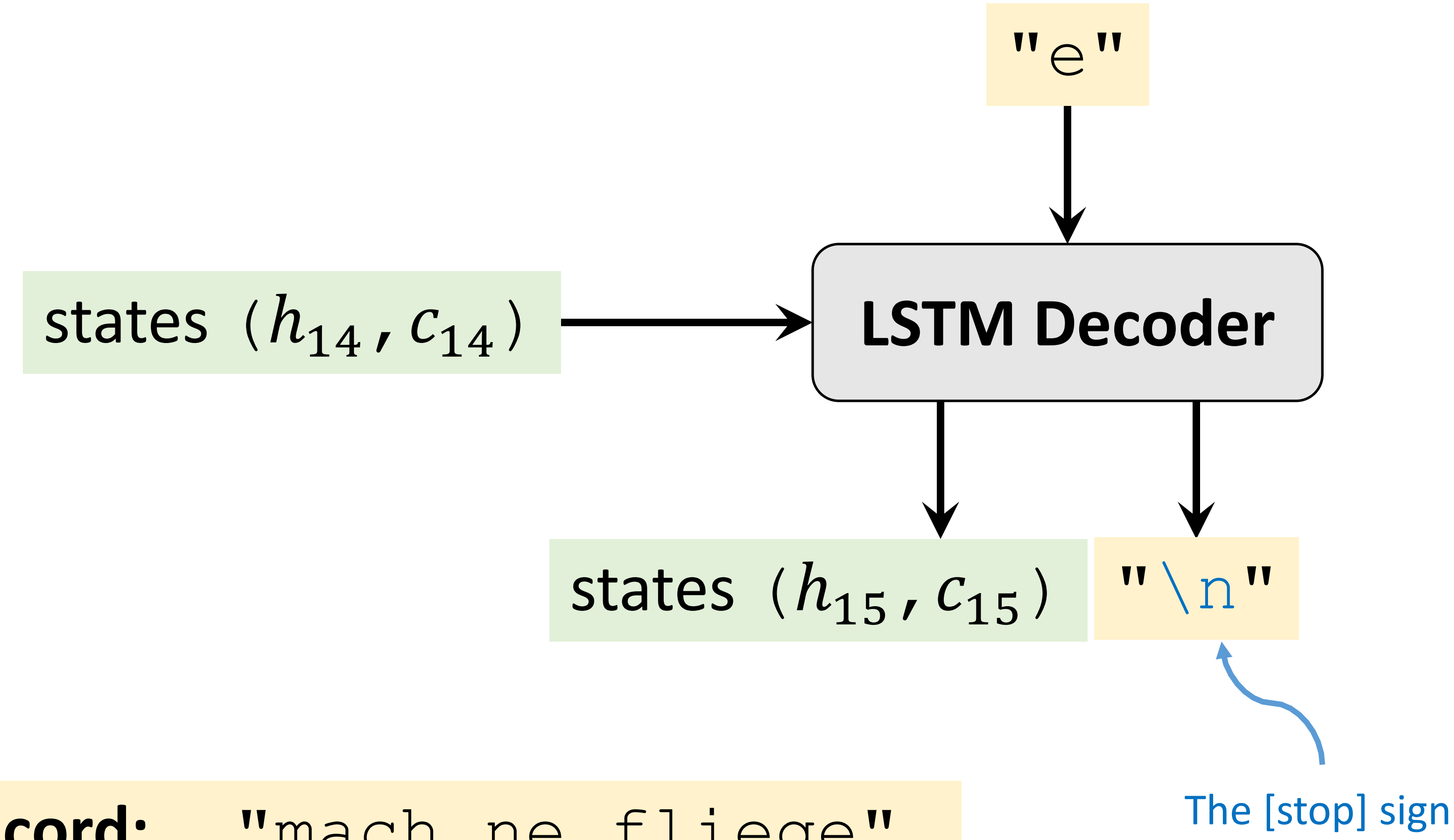


Inference



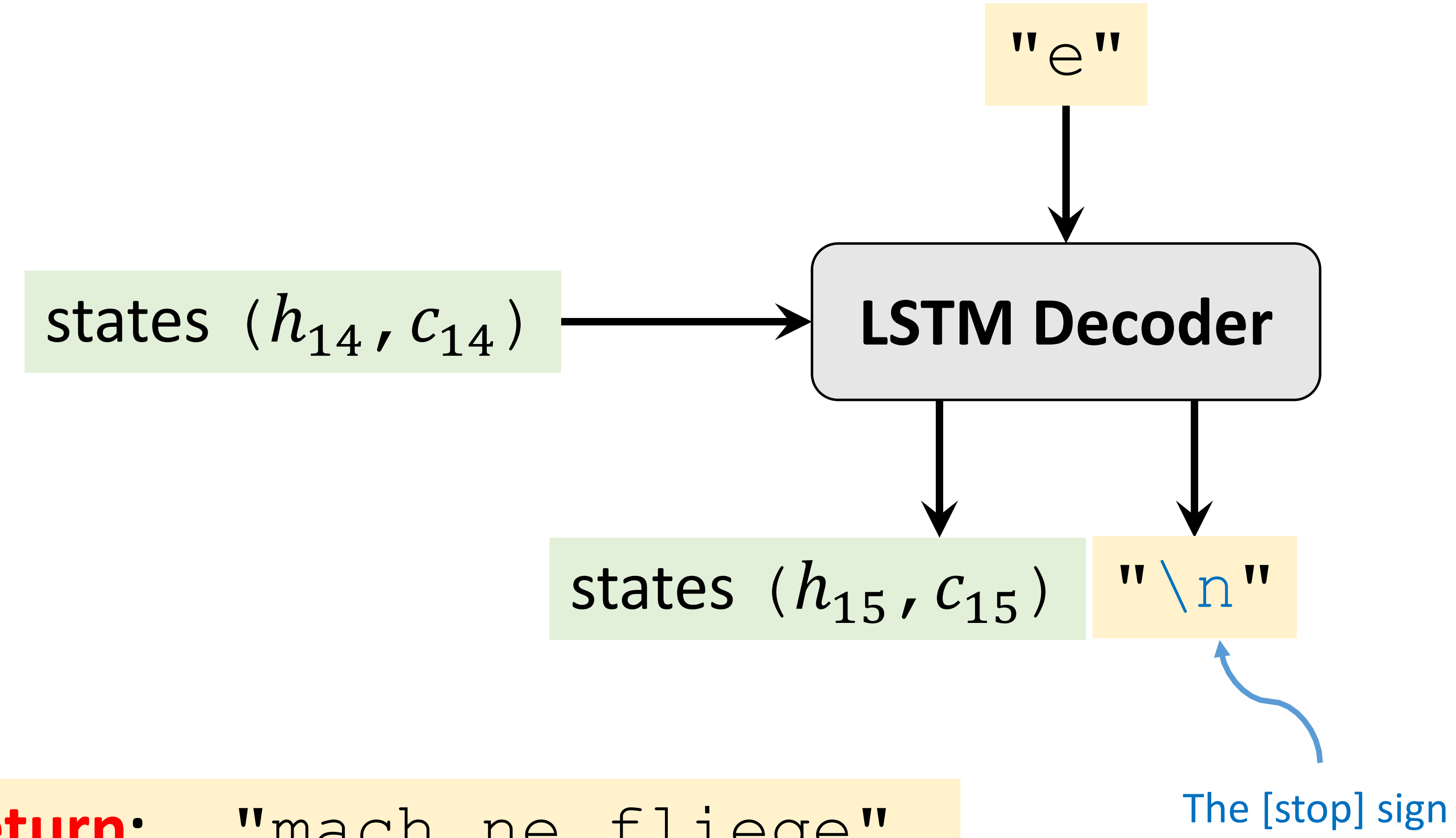
Record: `"mach ne fliege"`

Inference



Record: "mach ne fliege"

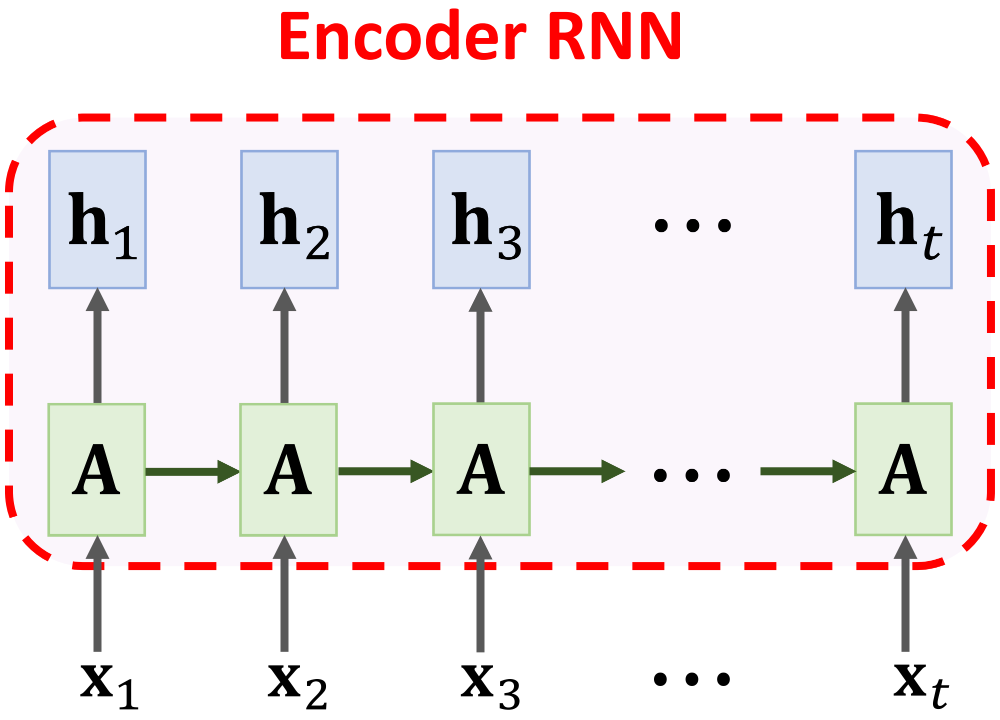
Inference



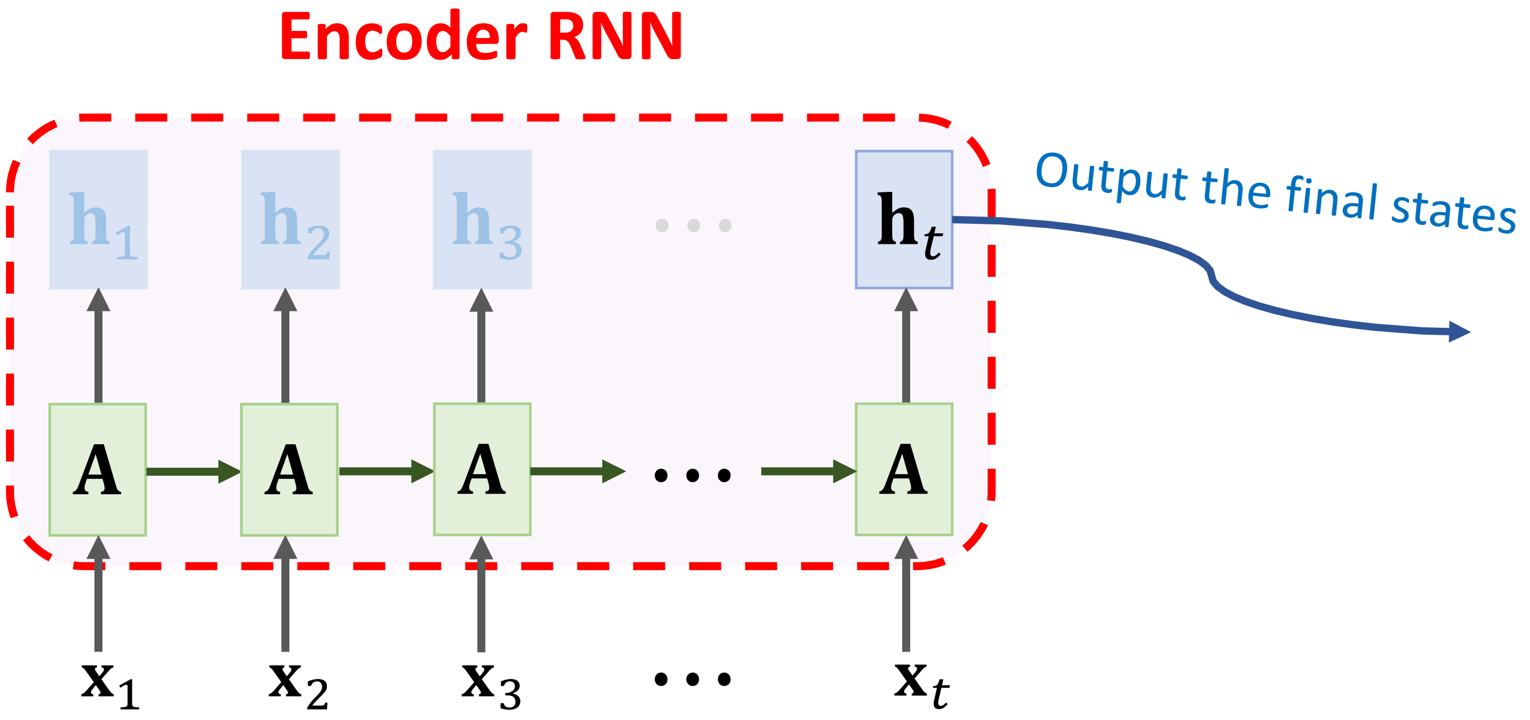
Return: `"mach ne fliege"`

Summary

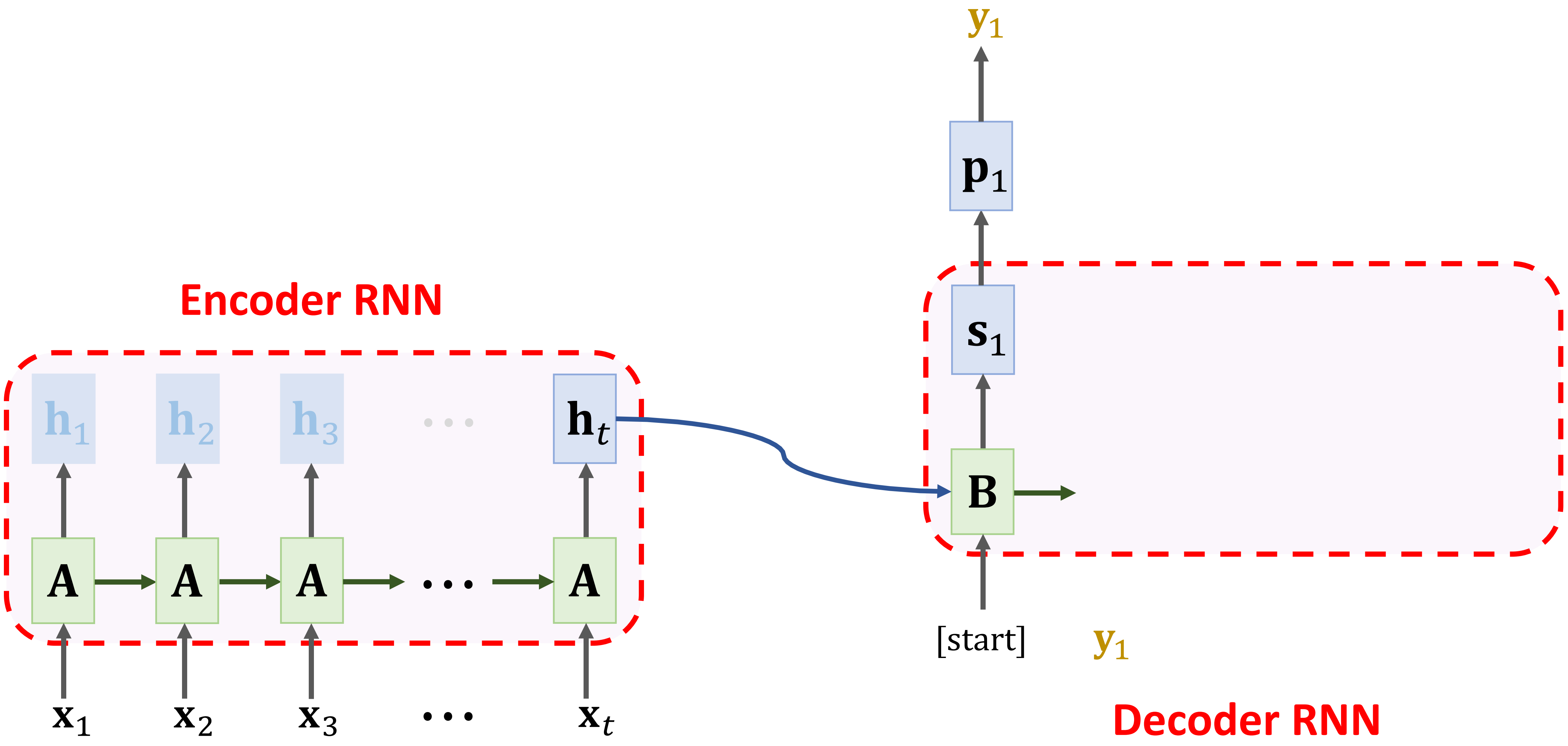
Seq2Seq Model



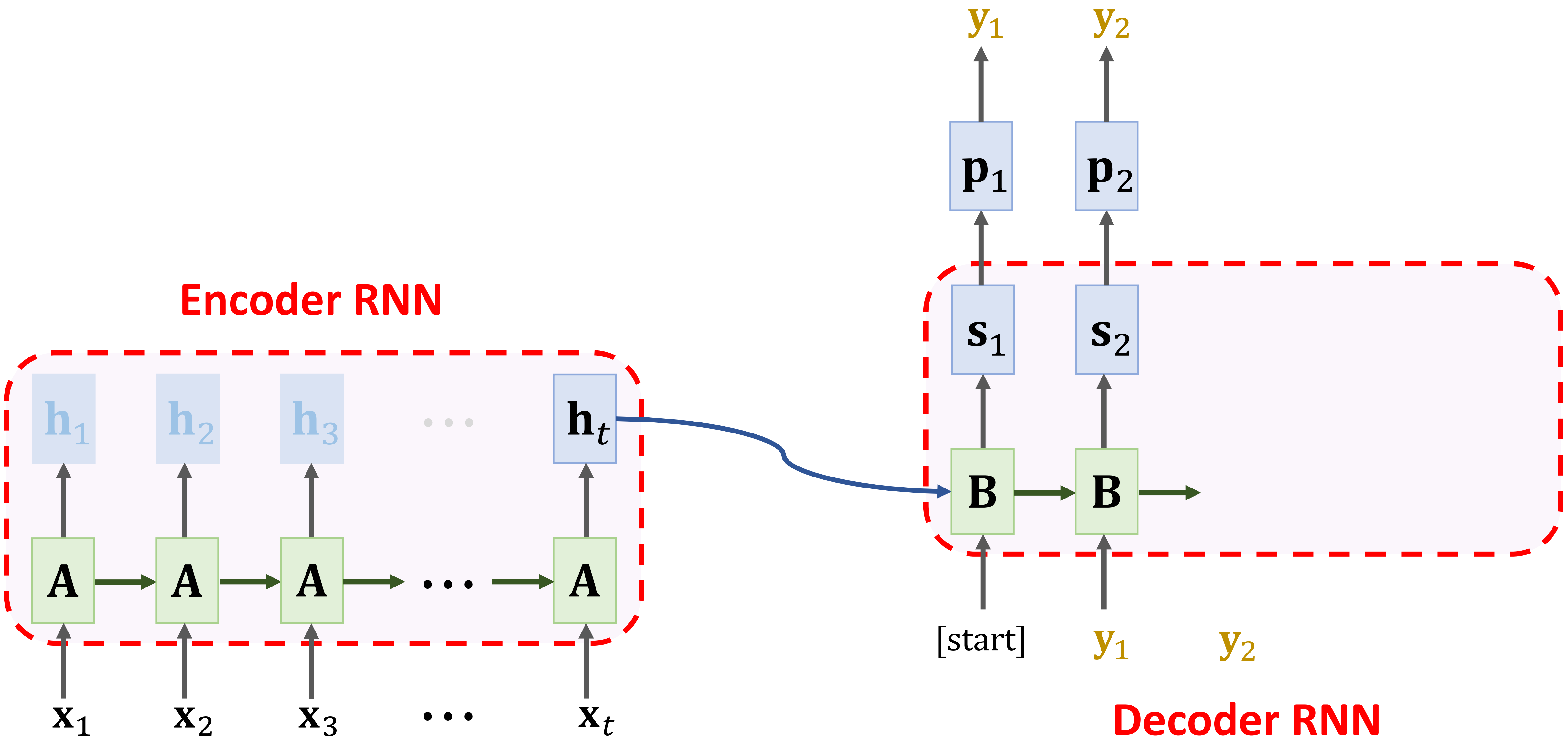
Seq2Seq Model



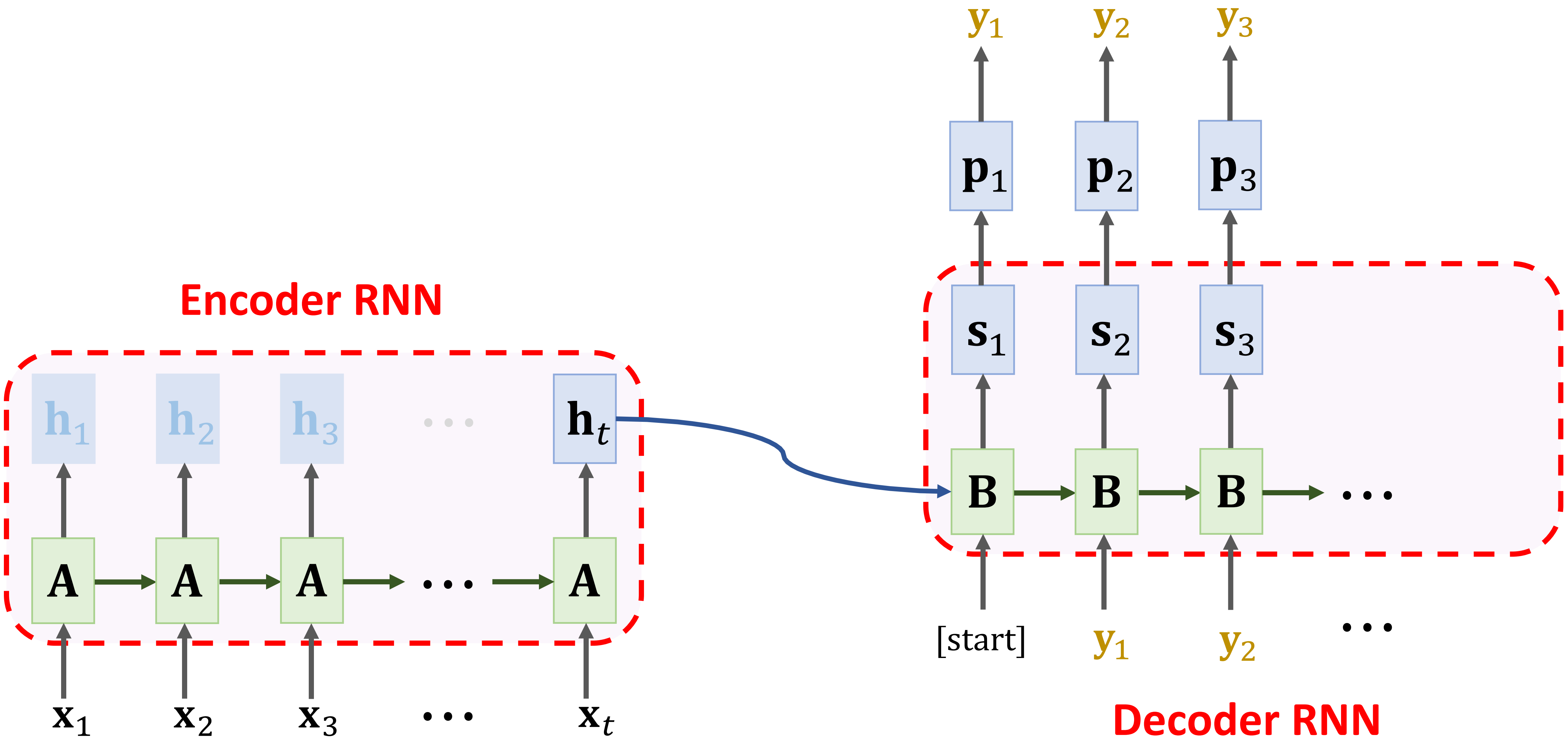
Seq2Seq Model



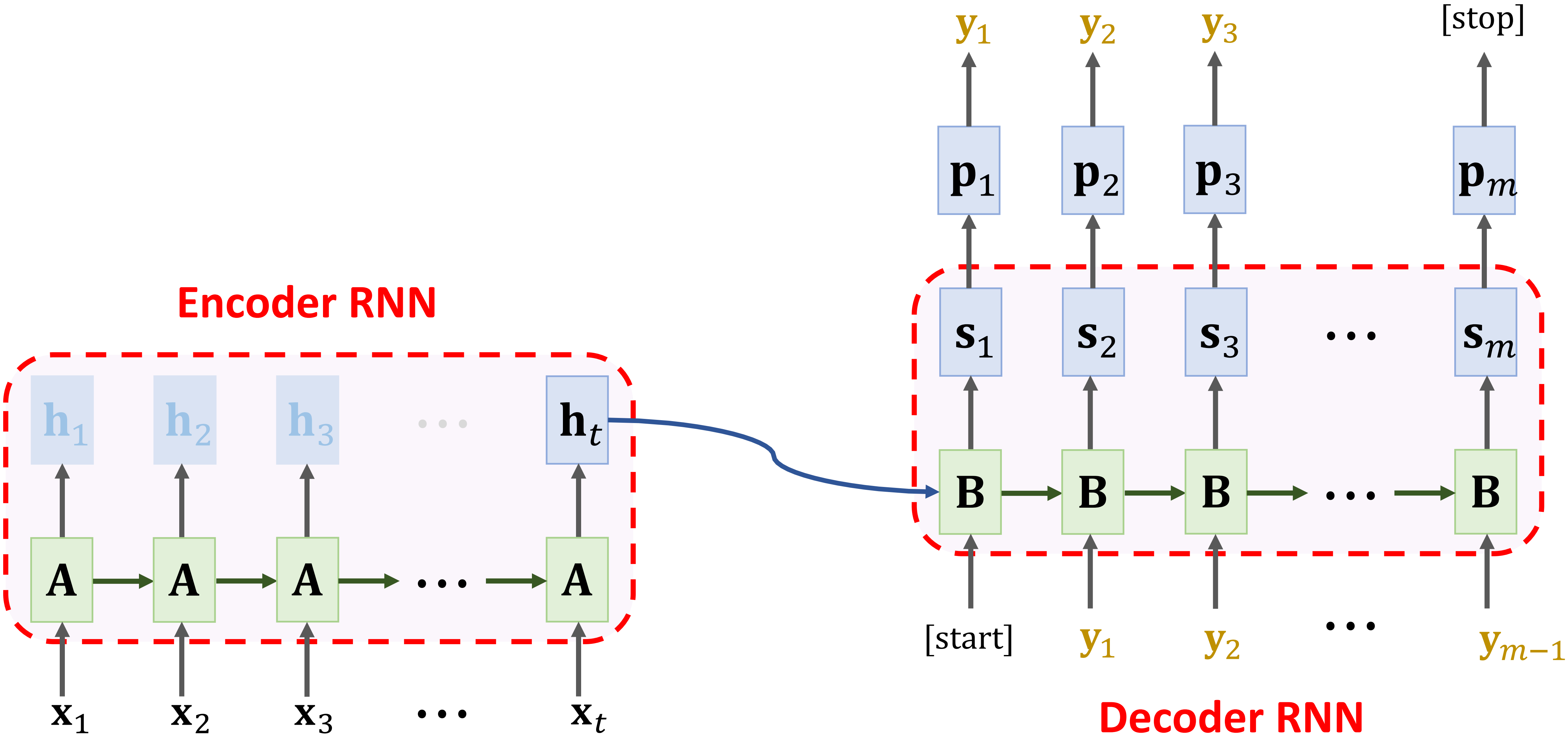
Seq2Seq Model



Seq2Seq Model



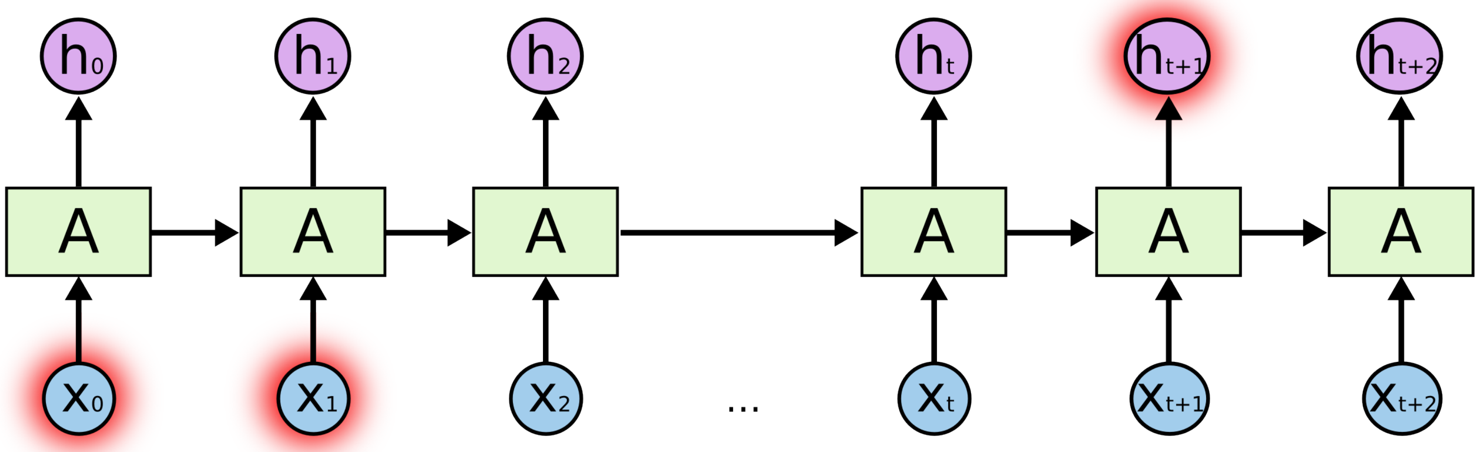
Seq2Seq Model



How to Improve?

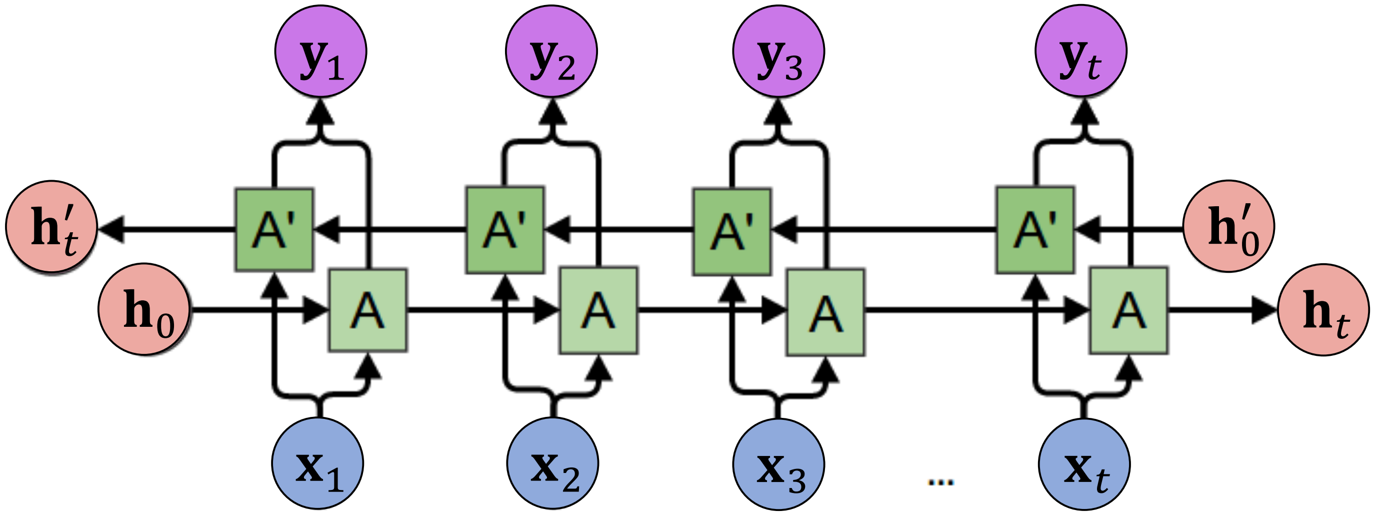
1. Bi-LSTM instead of LSTM (Encoder only!)

- The final states ($\mathbf{h}_t$ and $\mathbf{c}_t$) of the Encoder have all the information of the English sentence.
- Really?
  - If the sentence is long, the final states have forgotten the first tokens.



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

- The final states ($\mathbf{h}_t$ and $\mathbf{c}_t$) of the Encoder have all the information of the English sentence.
- Really?
  - If the sentence is long, the final states have forgotten the first tokens.
- Bi-LSTM (left-to-right and right-to-left) remembers the first tokens.



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

- The final states ($\mathbf{h}_t$ and $\mathbf{c}_t$) of the Encoder have all the information of the English sentence.
- Really?
  - If the sentence is long, the final states have forgotten the first tokens.
- Bi-LSTM (left-to-right and right-to-left) remembers the first tokens.
- Use Bi-LSTM in the encoder; use unidirectional LSTM in the decoder.

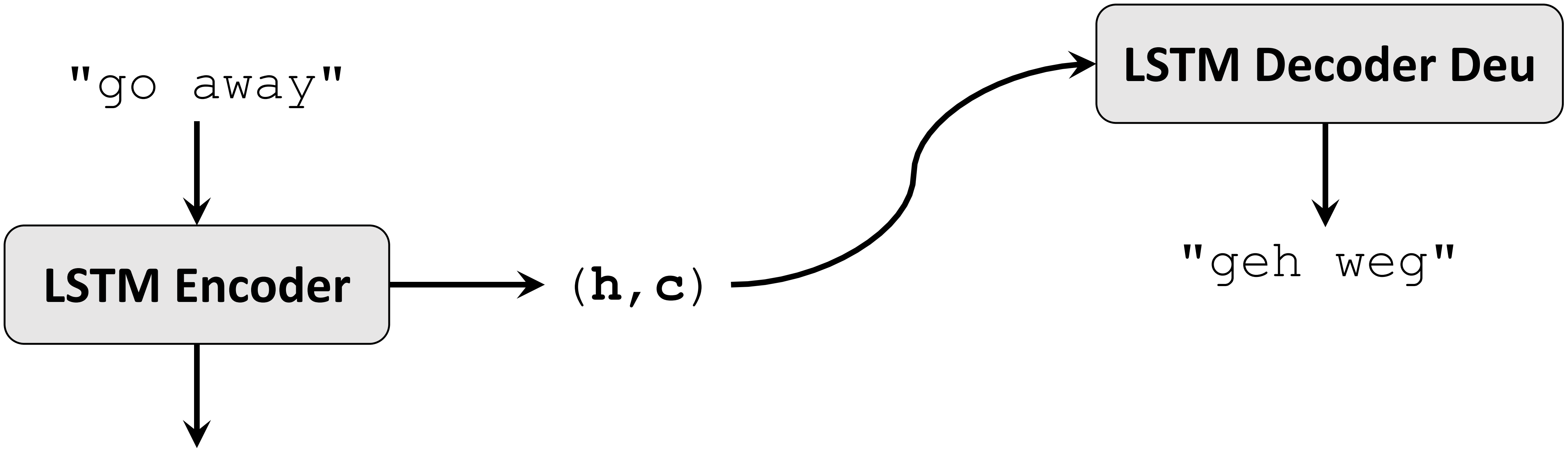
2. Word-Level Tokenization

- 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.

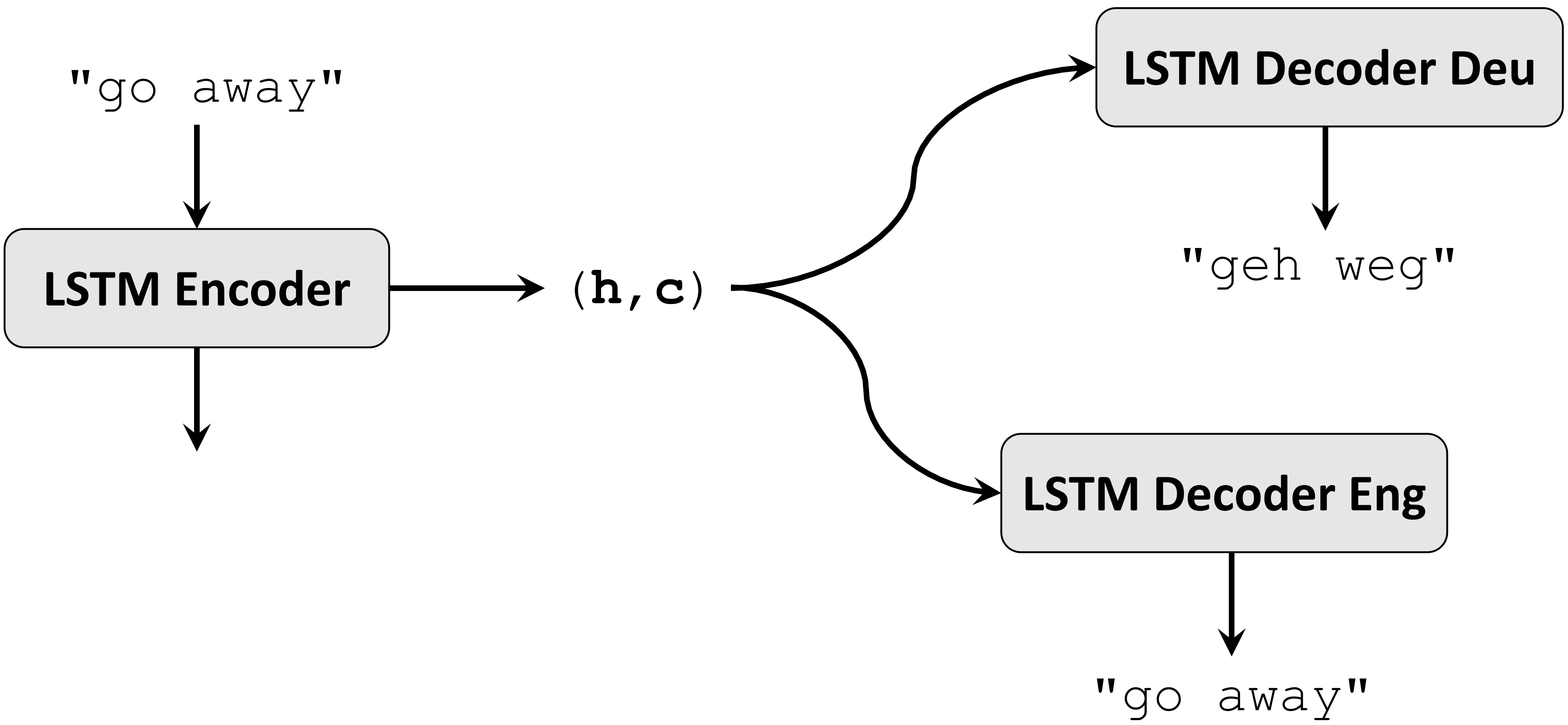
2. Word-Level Tokenization

- 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 ~10^2 ➔ one-hot suffices.
  - # of (frequently used) words is ~10^4 ➔ must use embedding.
  - Embedding Layer has many parameters ➔ overfitting!

3. Multi-Task Learning



3. Multi-Task Learning



3. Multi-Task Learning

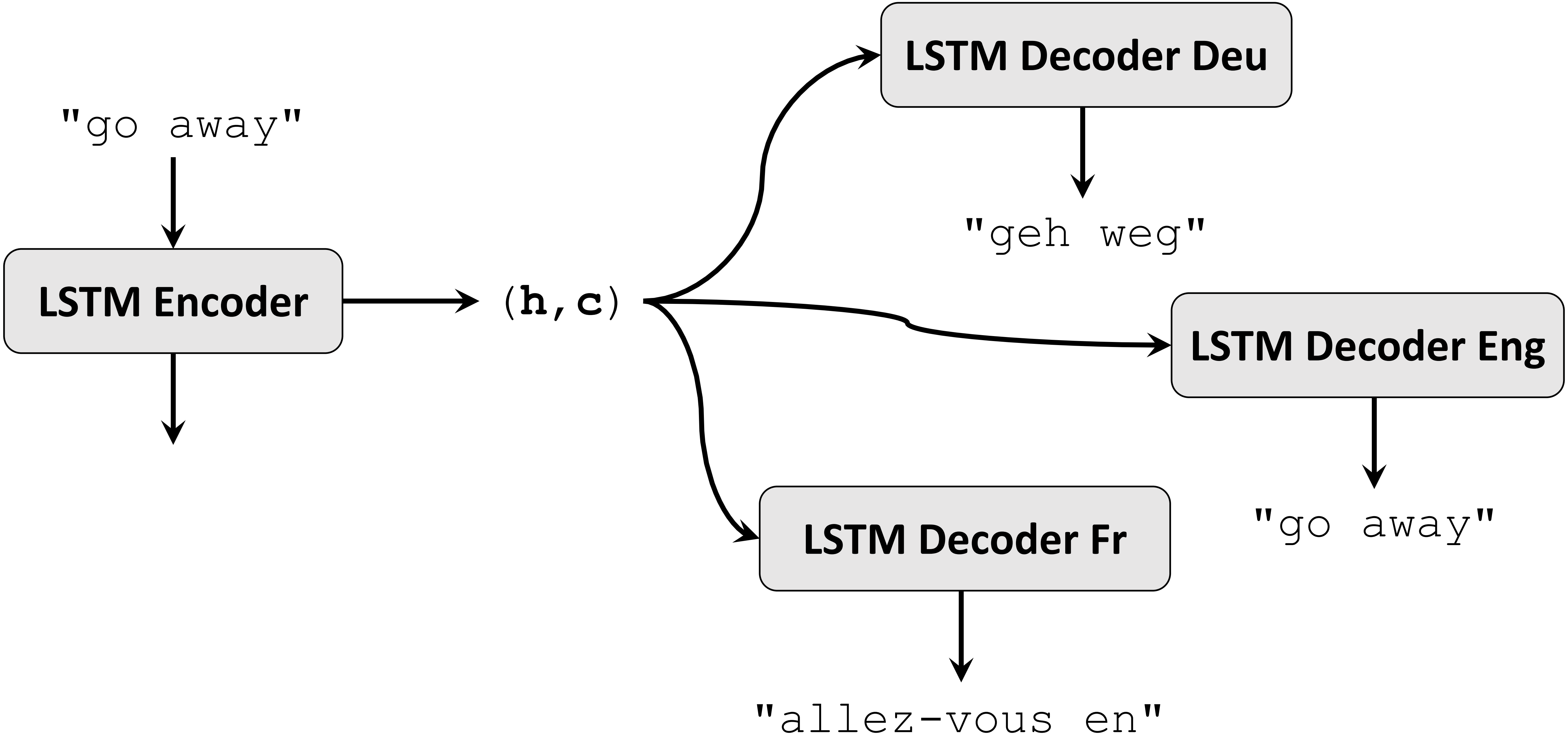
- Even if you want to translate English to German, you can use all the datasets:

- 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.)

Homework 5

- Build a seq2seq model for machine translation.
  - Anything languages except for [English ==> German].
  - Follow my IPython Notebook.
- Make as least one improvement over my naïve model.
  - E.g., Bi-LSTM, attention, etc.
- Evaluate your model using BLEU score. (Optional.)
  - BLEU (BiLingual Evaluation Understudy).
  - Reference:
    - Wikipedia: https://en.wikipedia.org/wiki/BLEU
    - Blog: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

Homework 5

- You can get up to 2 bonus scores:
  - 1pt: attention
  - 1pt: BLEU score

- You will get a bonus score only if you do it right.
  - E.g., BLEU score should be around 0.1~0.5; over-high or over-low means something is wrong.
  - If your result looks obviously wrong, don't submit; it would be a waste of everyone's time.

Thank you!