# 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.

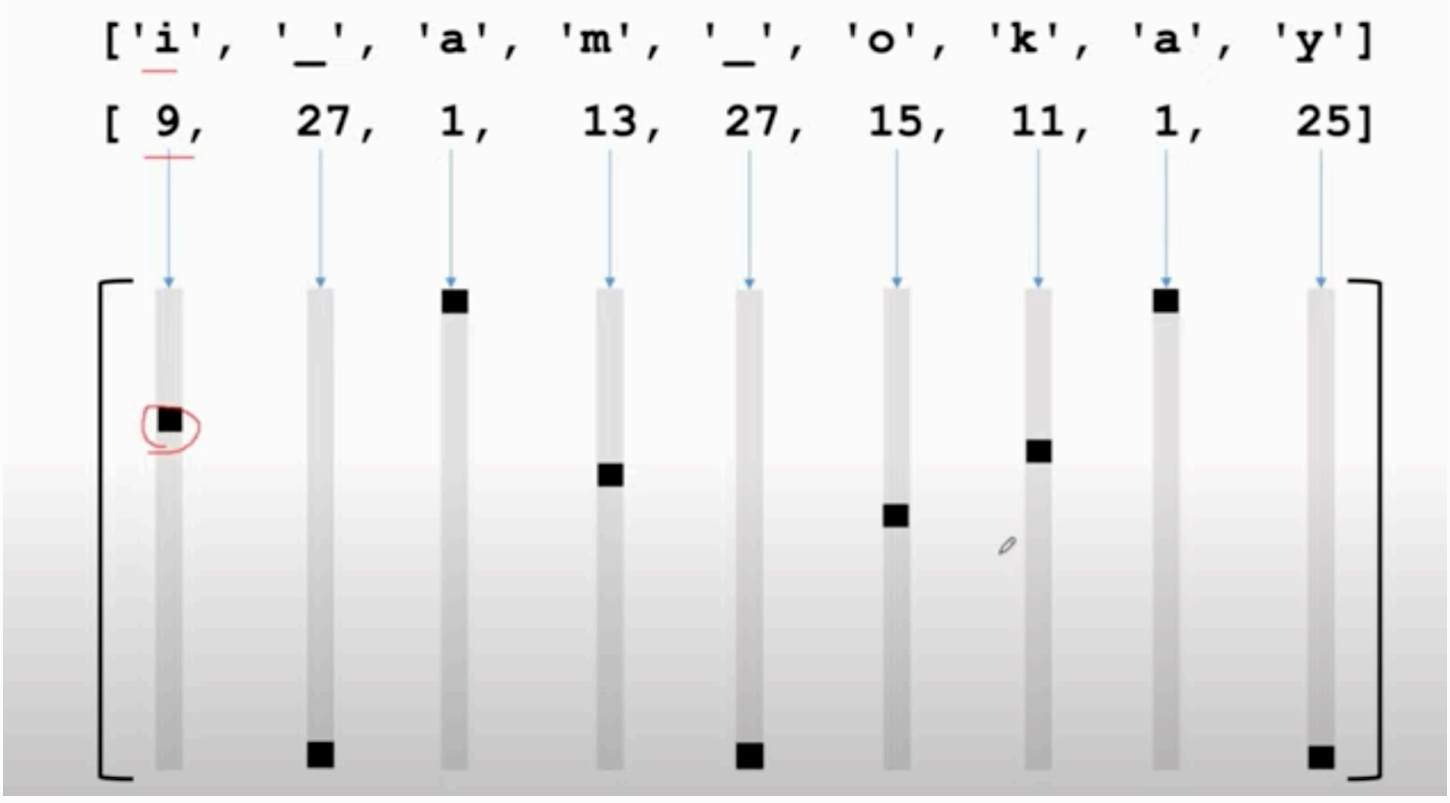
- English:

```
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.
```

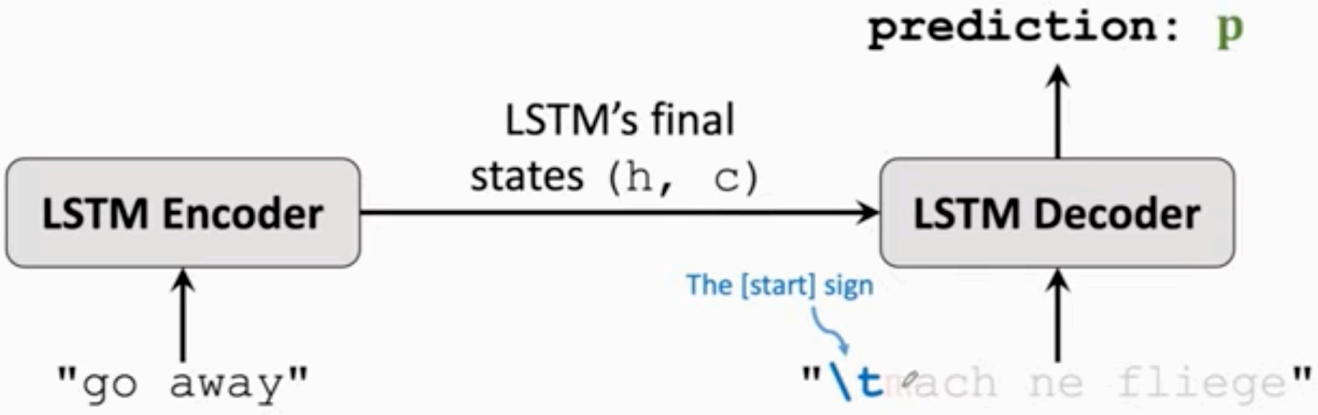
- Deutsche:

```
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.
```

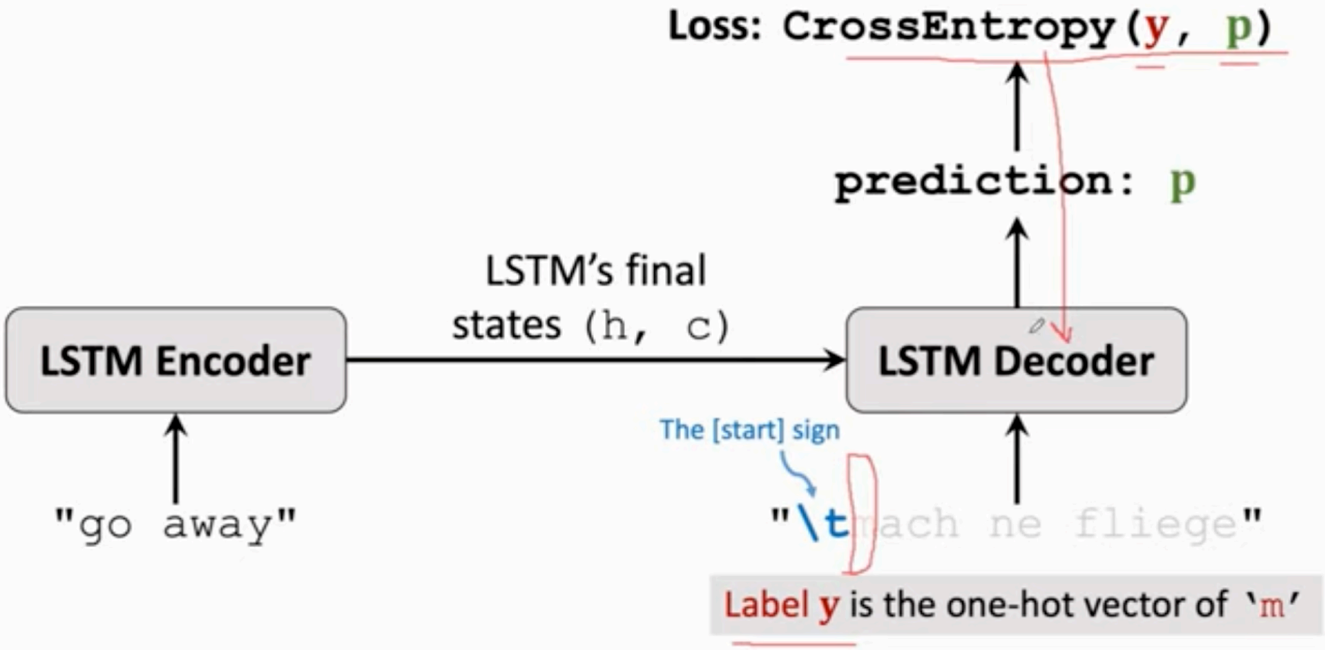
# 2. One-Hot Encoding



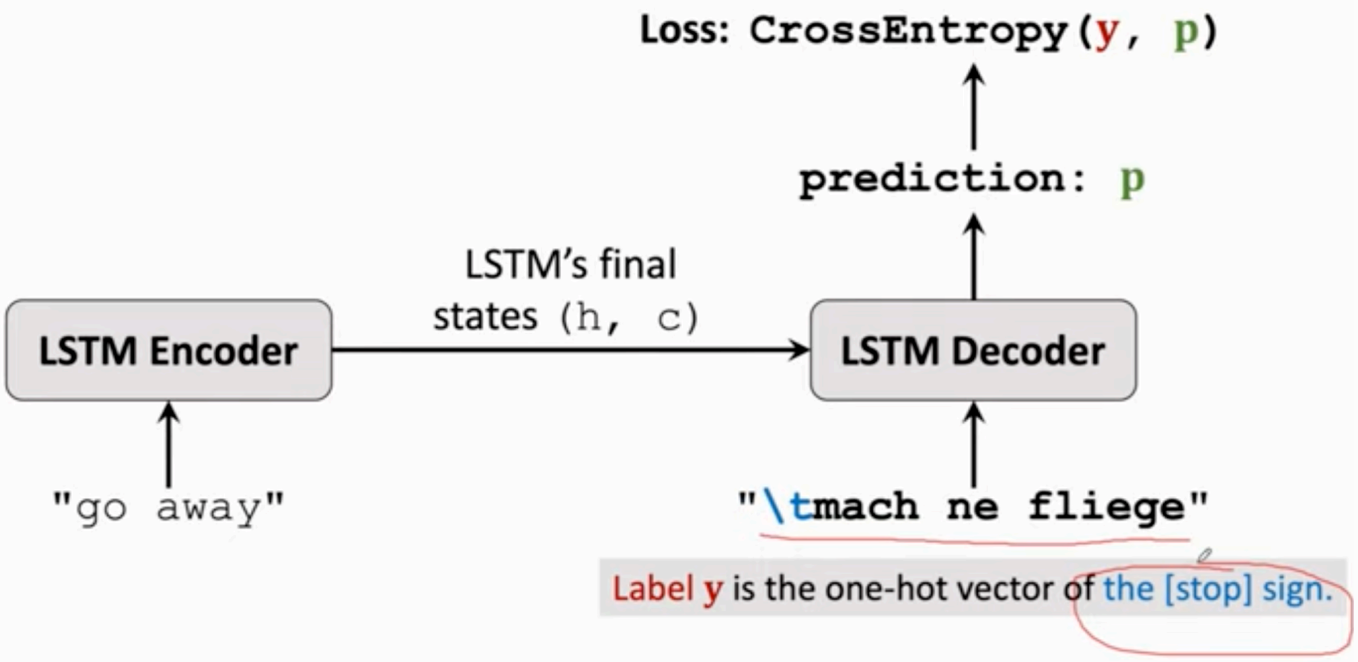
# Training Seq2Seq Model



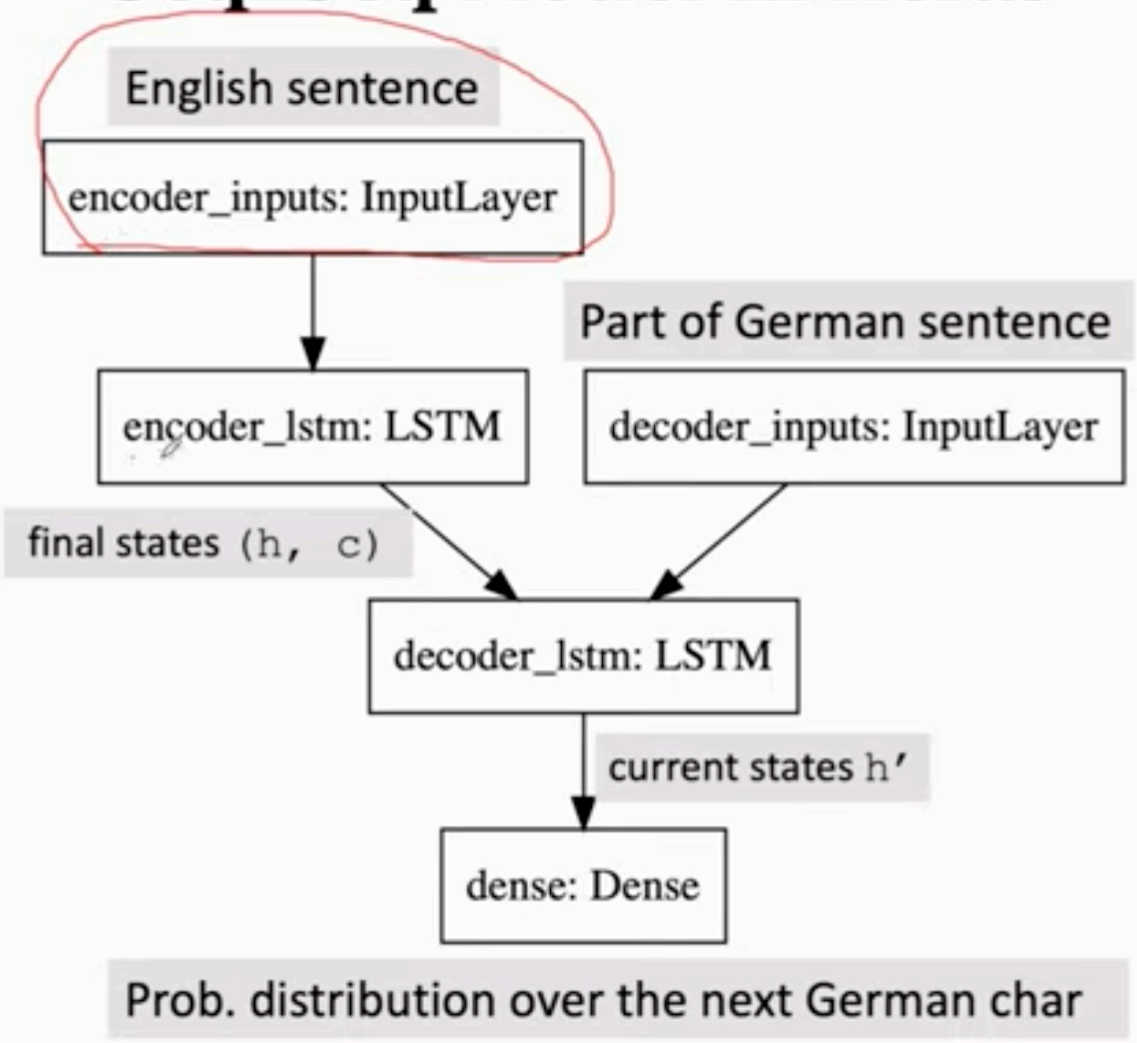

# Training Seq2Seq Model



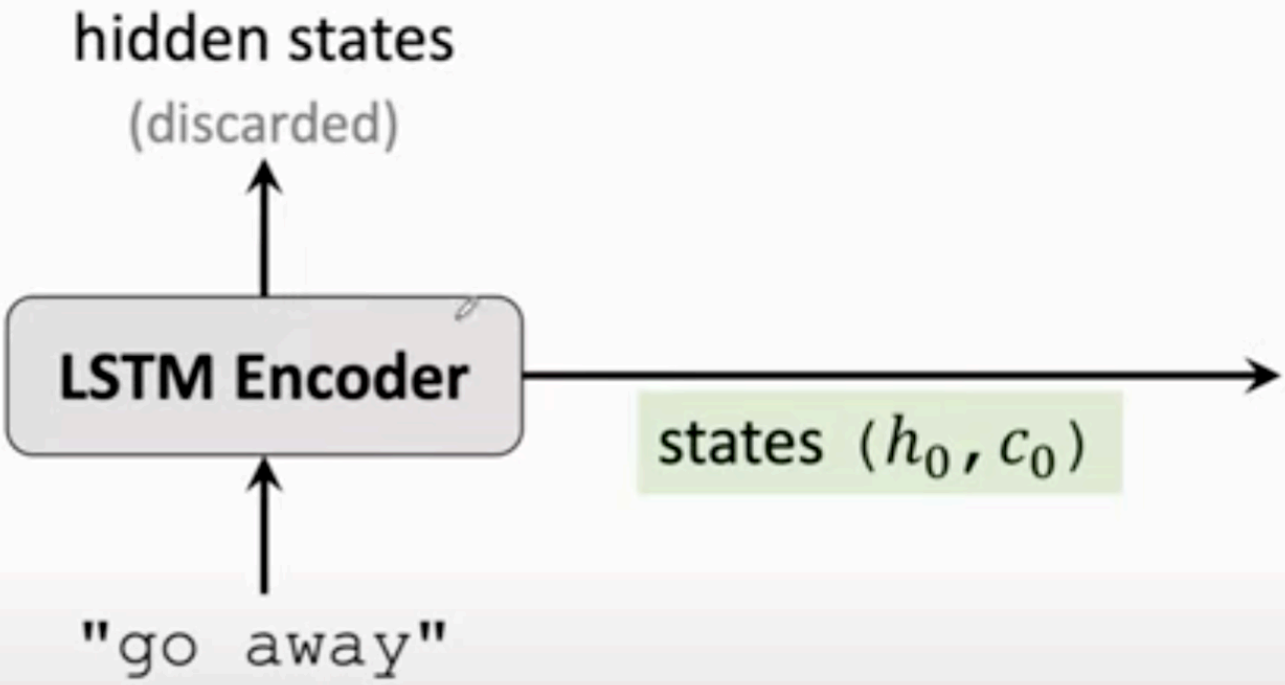
# Training Seq2Seq Model



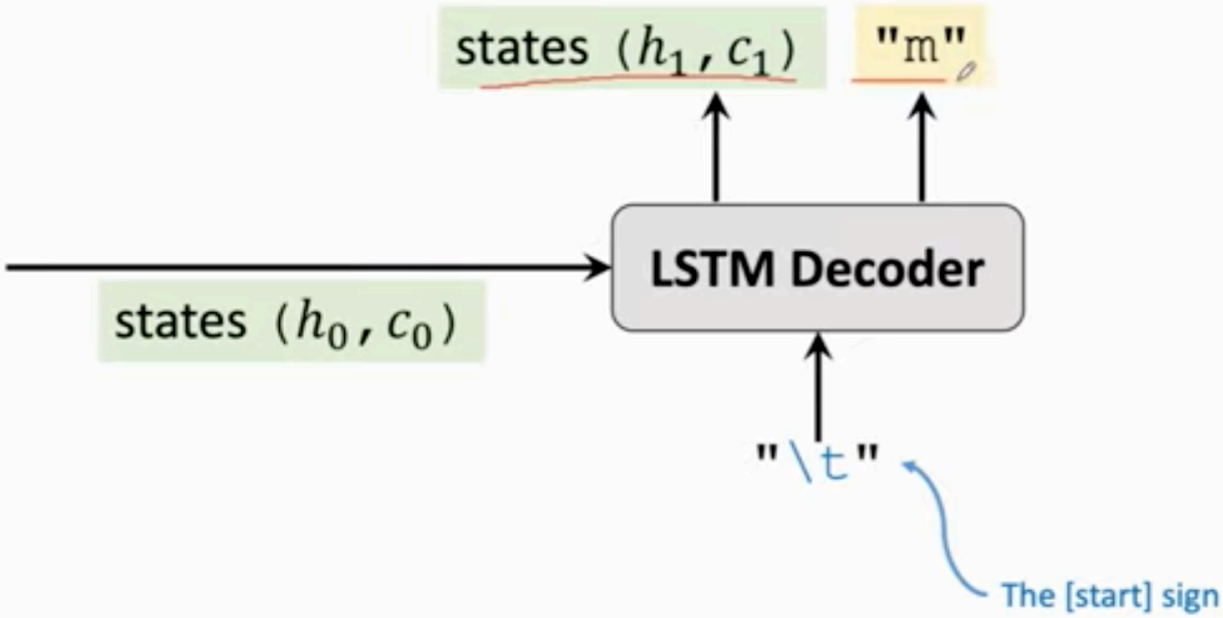
# Seq2Seq Model in Keras



# Inference

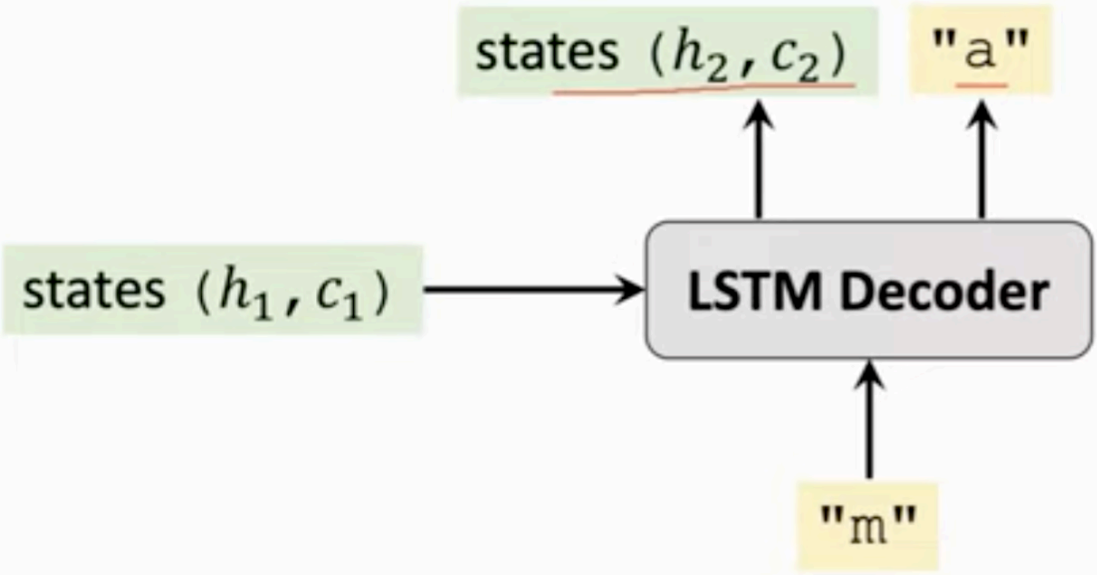


# Inference



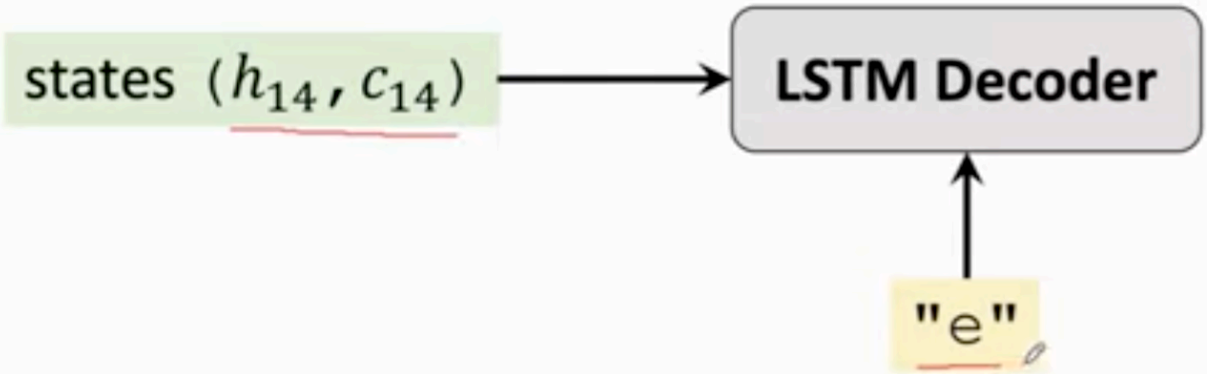

**Record:** "m"

# Inference

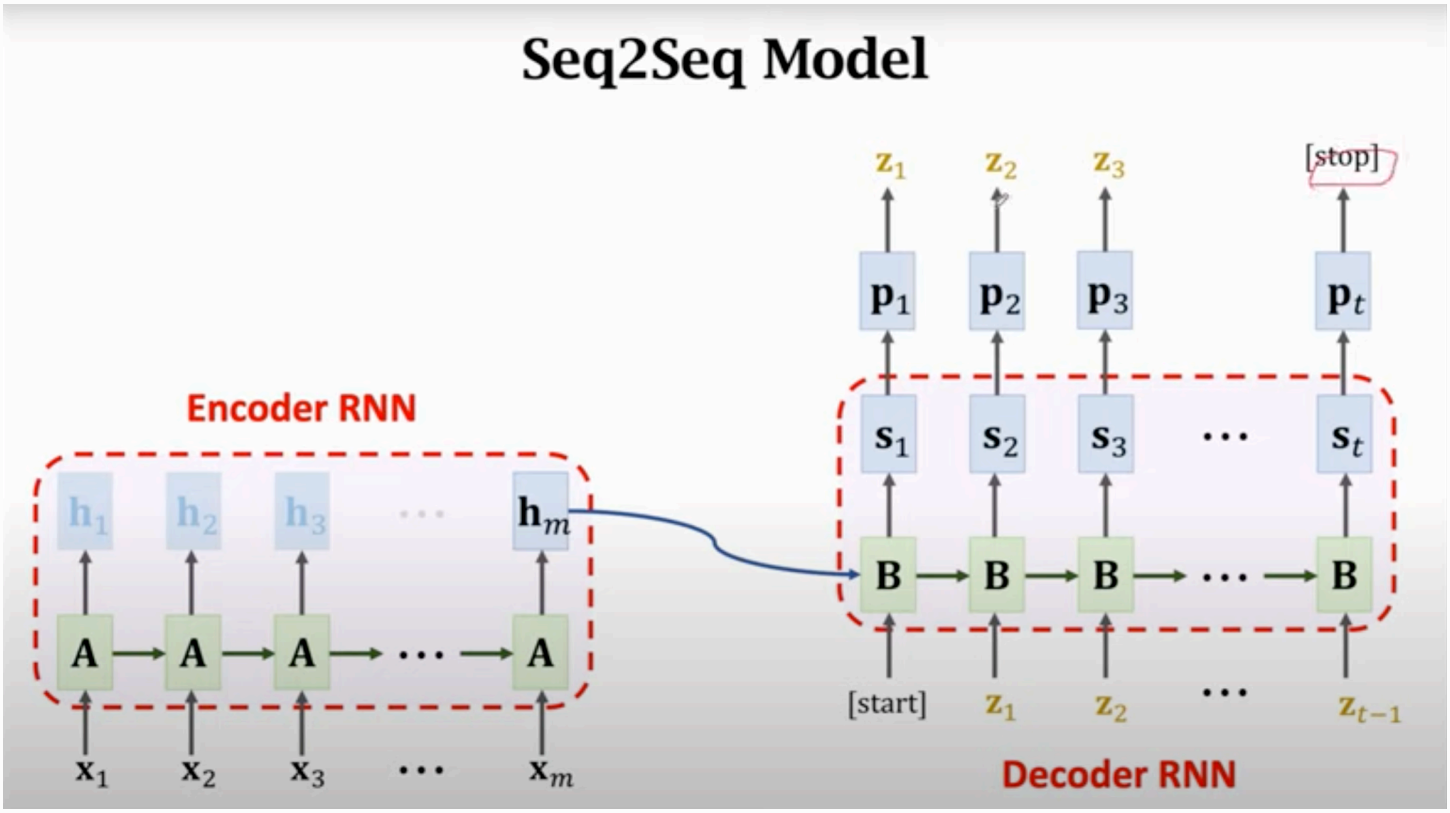



**Record:** "ma"

# Inference

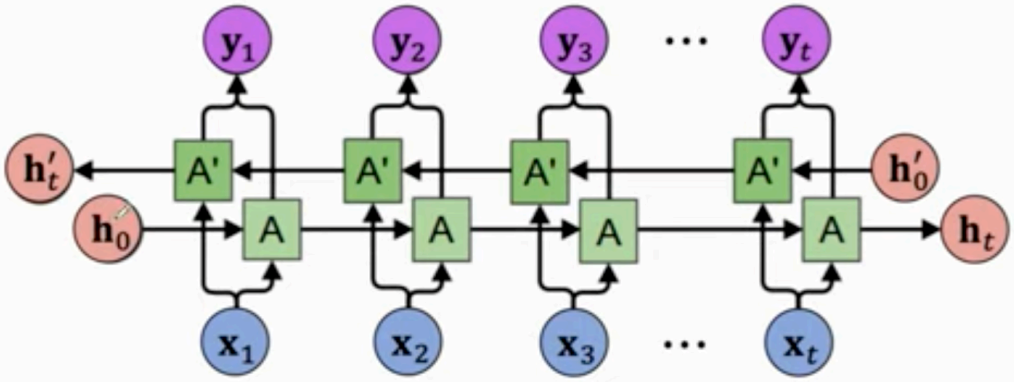



**Record:** "mach ne fliege"




# 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.

- 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