CMU 11-785 Introduction to Deep Learning, Fall 2020

Lecture 18

Neural Machine Translation & Attention

TAVE Research DL001

Changdae Oh

2021.05.02

Topics

- Seq2Seq for Machine Translation
- Decoding & Training process
- Attention model

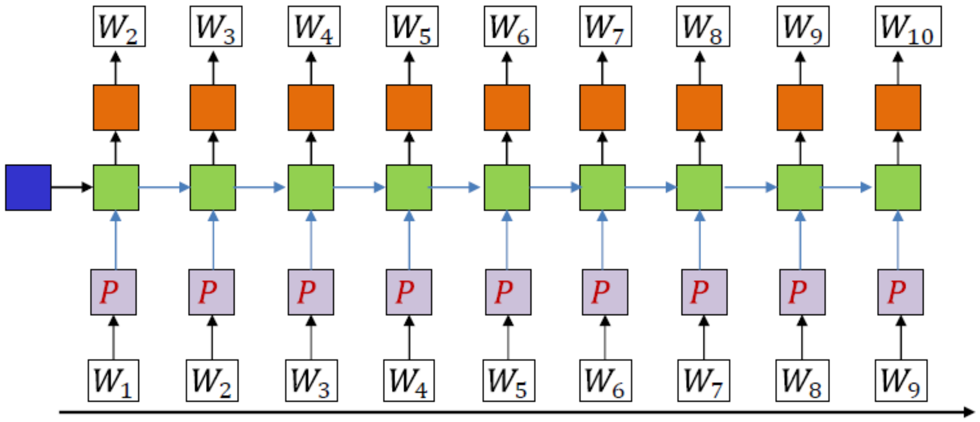
Topics

- ❖ Seq2Seq for Machine Translation
- ❖ Decoding & Training process
- ❖ Attention model

Seq2Seq for Machine Translation

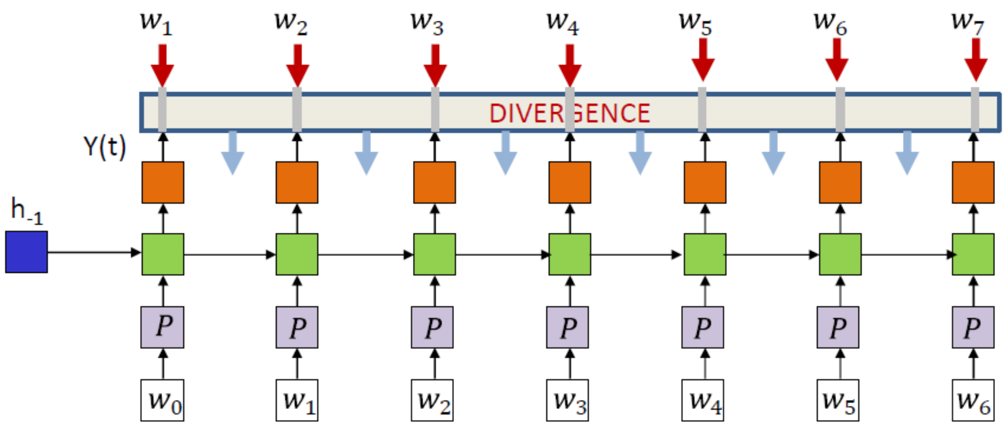
Recap

Language Modeling



- Learn a model that can predict the next symbol given a sequence of symbols
- After observing inputs w_0 , ... ,w_k (one-hot vectors) it predicts w_{k+1} (probability distribution)

Training LM



$$Y(t,i) = P(V_i | w_0 \dots w_{t-1})$$

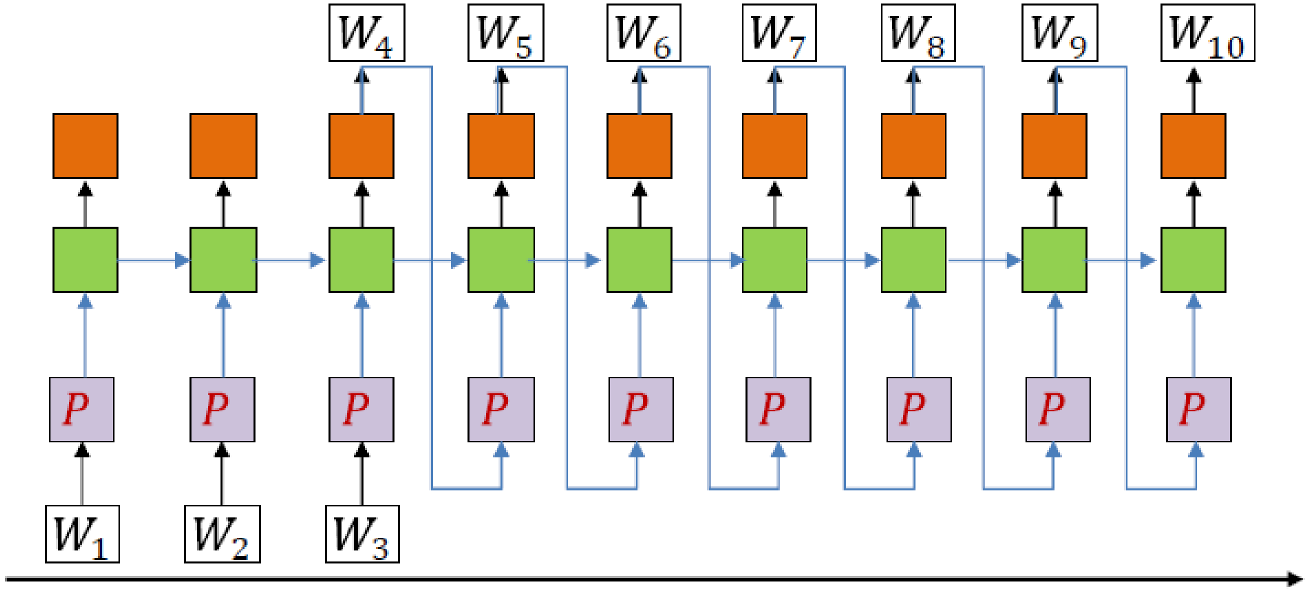
$$Div(w(1 \dots T), \mathbf{Y}(0 \dots T-1)) = \sum_t KL(w(t+1), \mathbf{Y}(t)) = -\sum_t \log Y(t, w_{t+1})$$

Probability assigned to the correct next word

Seq2Seq for Machine Translation

Recap

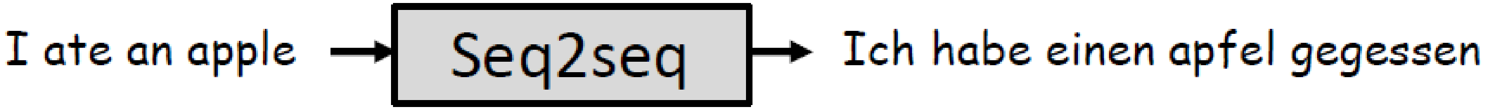
Generating Language



- Feed the drawn word as the next word in the series
- Continue until the model draws an <eos>

Seq2Seq for Machine Translation

object

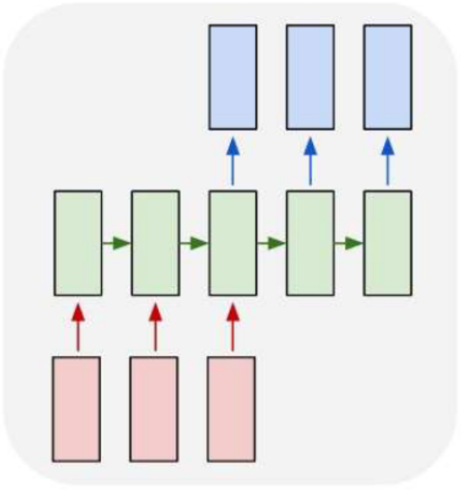


problem

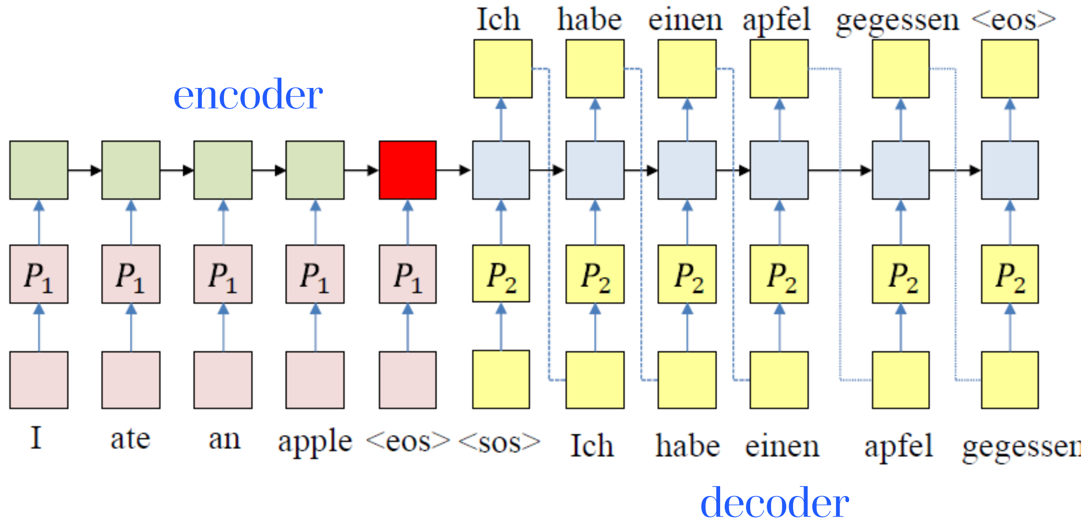
- No expected synchrony between input and output
- So, we can't solve the problem well by using only one RNN

model

many to many



Delayed self-referencing sequence to sequence

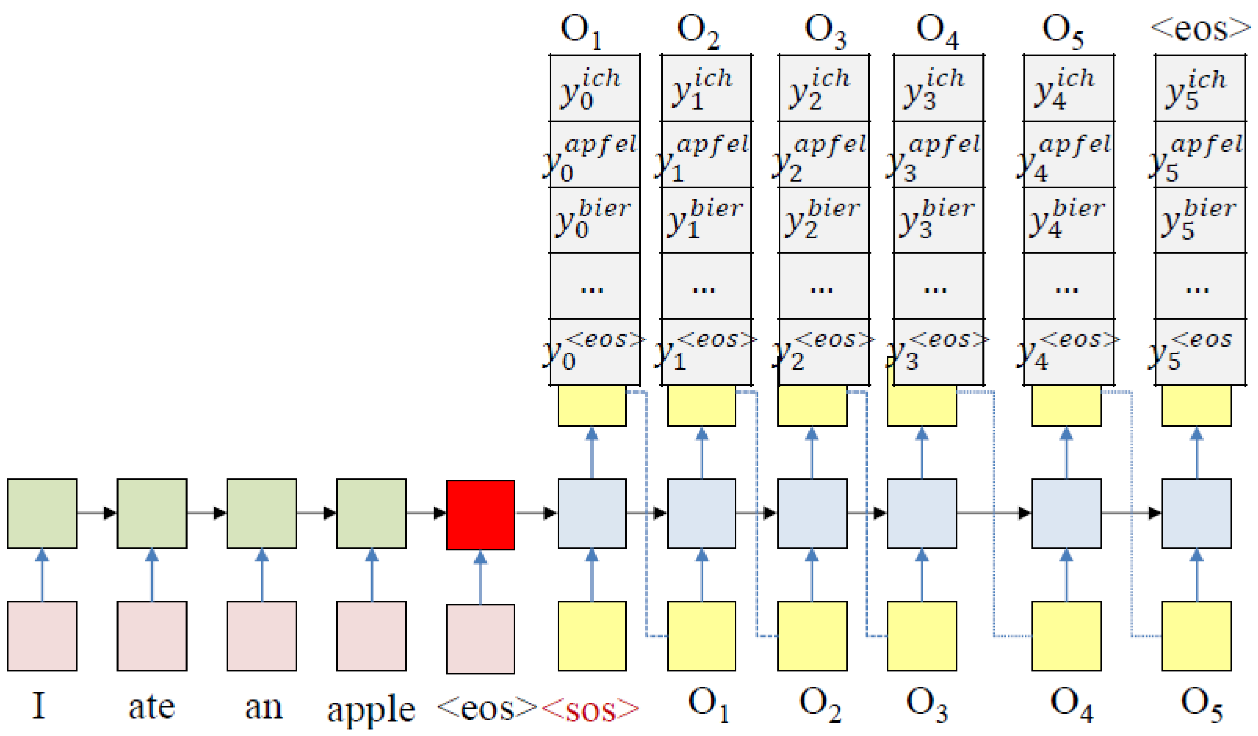


Topics

- ❖ Seq2Seq for Machine Translation
- ❖ Decoding & Training process
- ❖ Attention model

Decoding & Training process

Generating outputs



- Goal : produce the most likely output

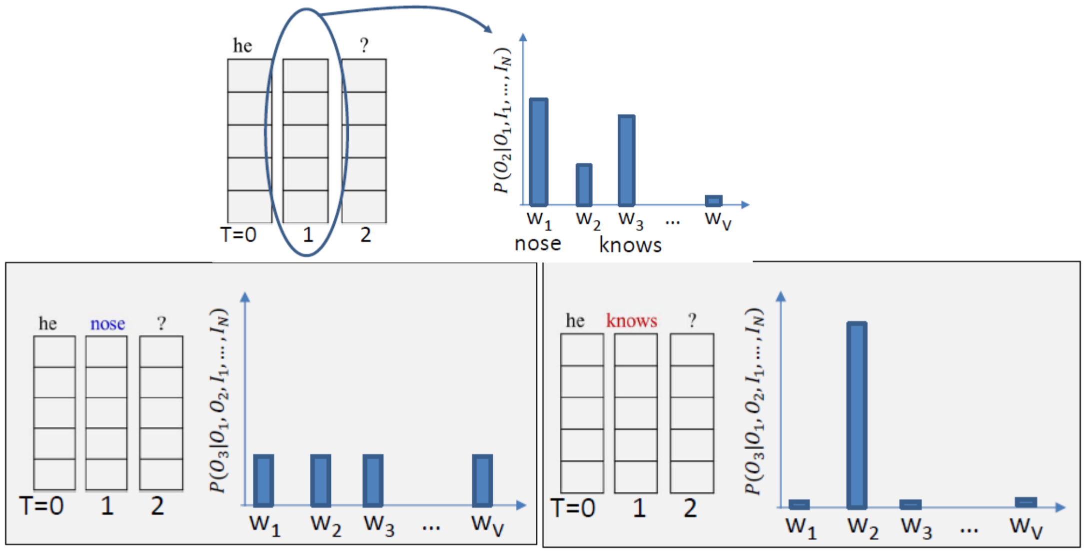
$$\underset{O_1,\ldots,O_L}{\operatorname{argmax}} P(O_1, \ldots, O_L | W_1^{in}, \ldots, W_N^{in})$$

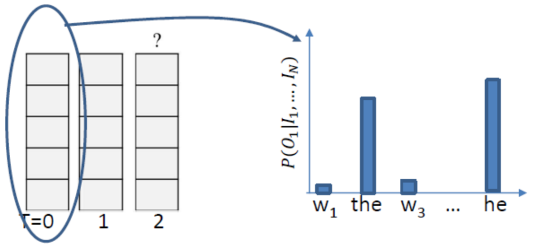
$$= \underset{O_1,\ldots,O_L}{\operatorname{argmax}} y_1^{O_1} y_2^{O_2} \ldots y_L^{O_L}$$

- Greedy drawing
- Random sampling
- Beam search

Decoding & Training process

Problems of greedy drawing





- Impossible to know a priori which word leads to the more promising future

Drawing by random sampling

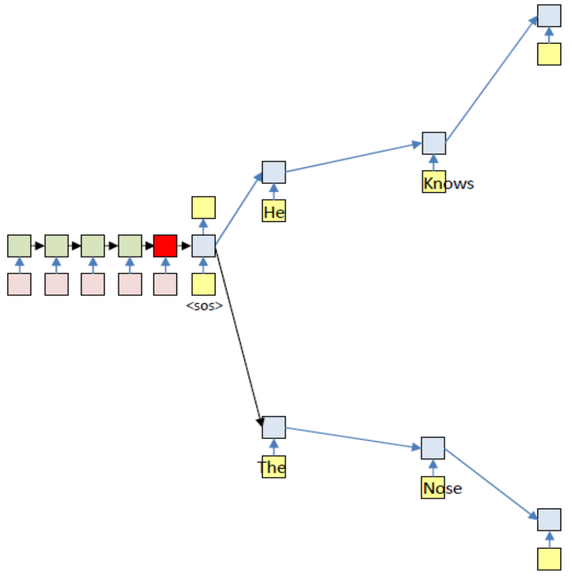
- Randomly draw a word at each time according to the output probability distribution
- Sometimes give more likely output than greedy
- But, not guaranteed to give the most likely output

- Making a poor choice at any time commits us to poor future
- But we cannot know at that time the choice was poor

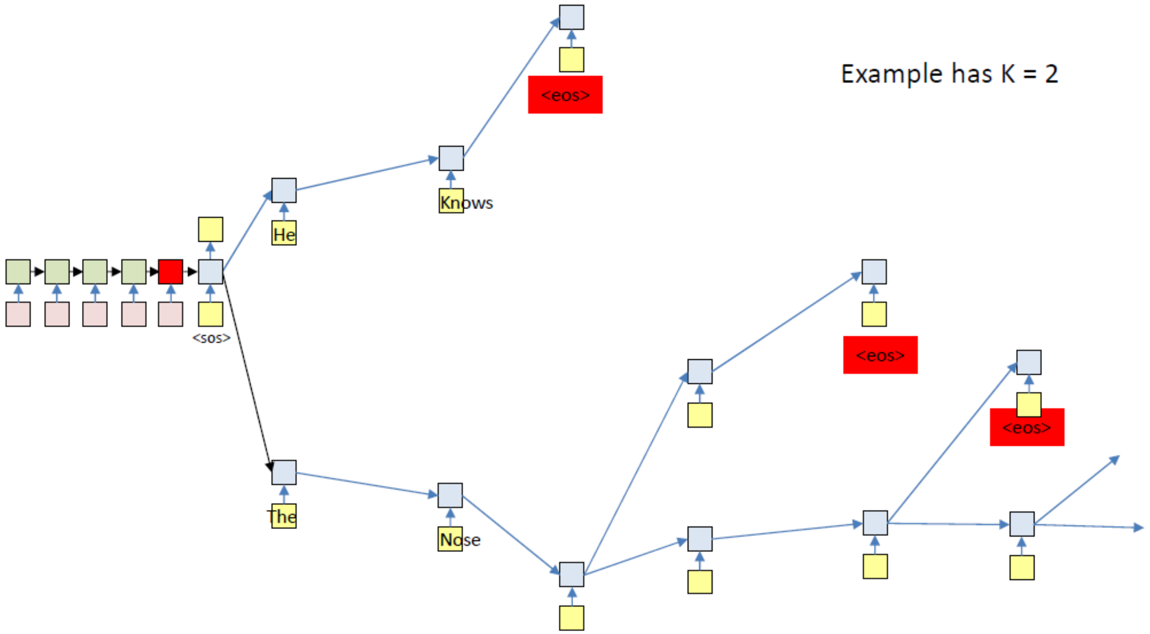
Solution : don't choose

Decoding & Training process

Multiple choices & pruning : **beam search**



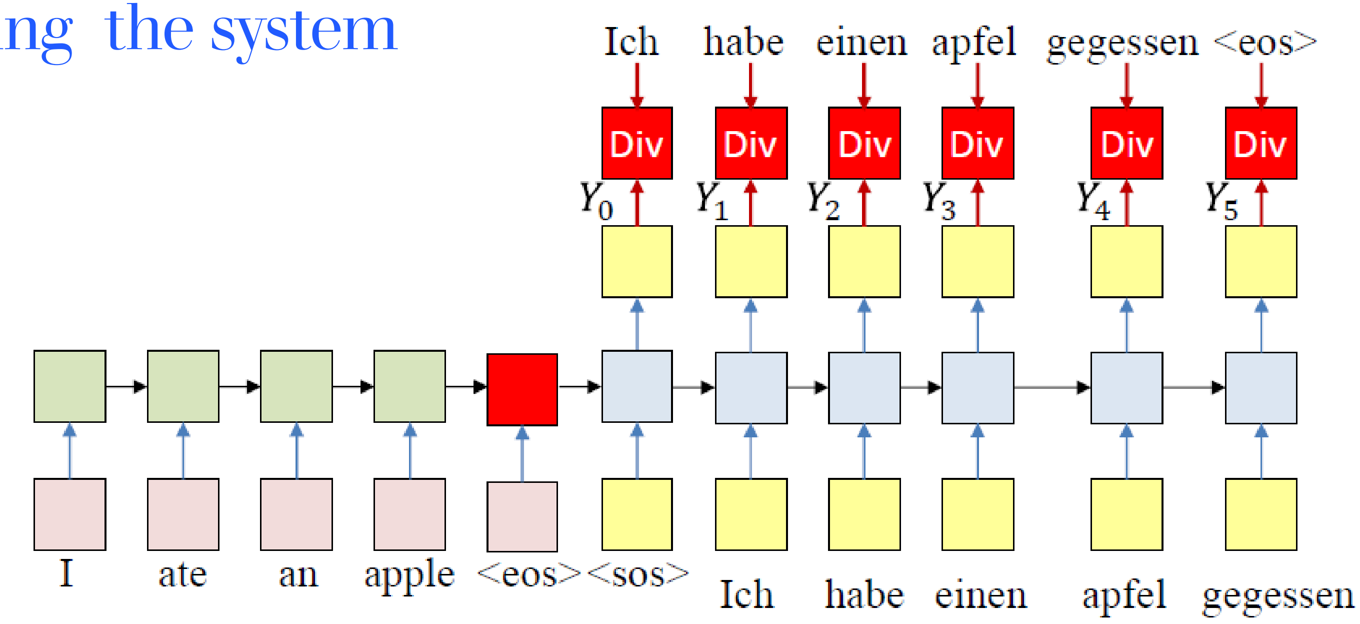
$$Top_K \prod_{t=1}^{n} P(O_t|O_1,\dots,O_{t-1},I_1,\dots,I_N)$$



- At each time, retain only the top K scoring forks
- Terminate when the current most likely path overall ends in <eos>
  - select the most likely sequence ending in <eos> across all terminating seqeences

Decoding & Training process

Training the system



✓ Can reversing the input seq

✓ Can randomly choose some output words for backprop gradients

forward

- Input source seq to encoder & target seq (ground truth) to decoder (Use teacher forcing) => easier training, calculate divergence
- Compute the divergence between the output distribution and target word sequence

backward

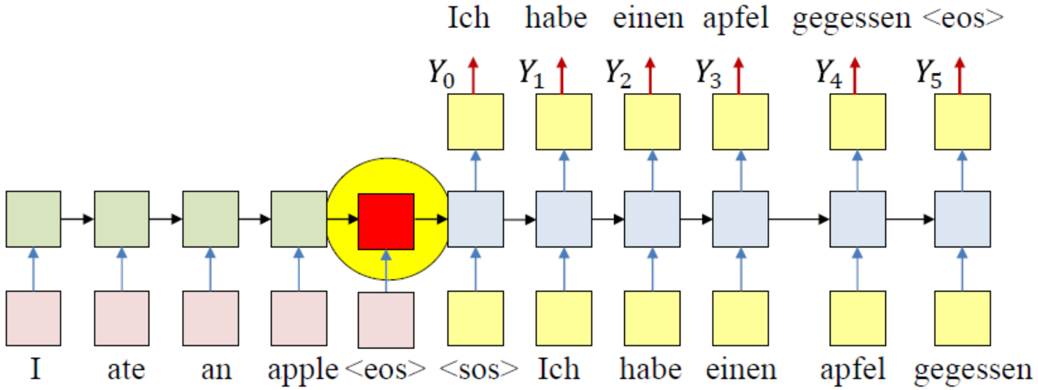
- Backprop DIV through whole end2end network

Topics

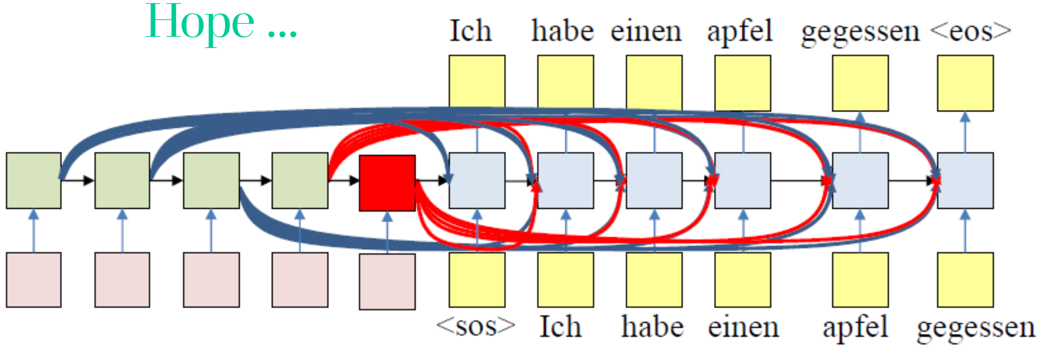
- ❖ Seq2Seq for Machine Translation
- ❖ Decoding & Training process
- ❖ Attention model

Attention model

Problem with naïve enc-dec net



Hope ...

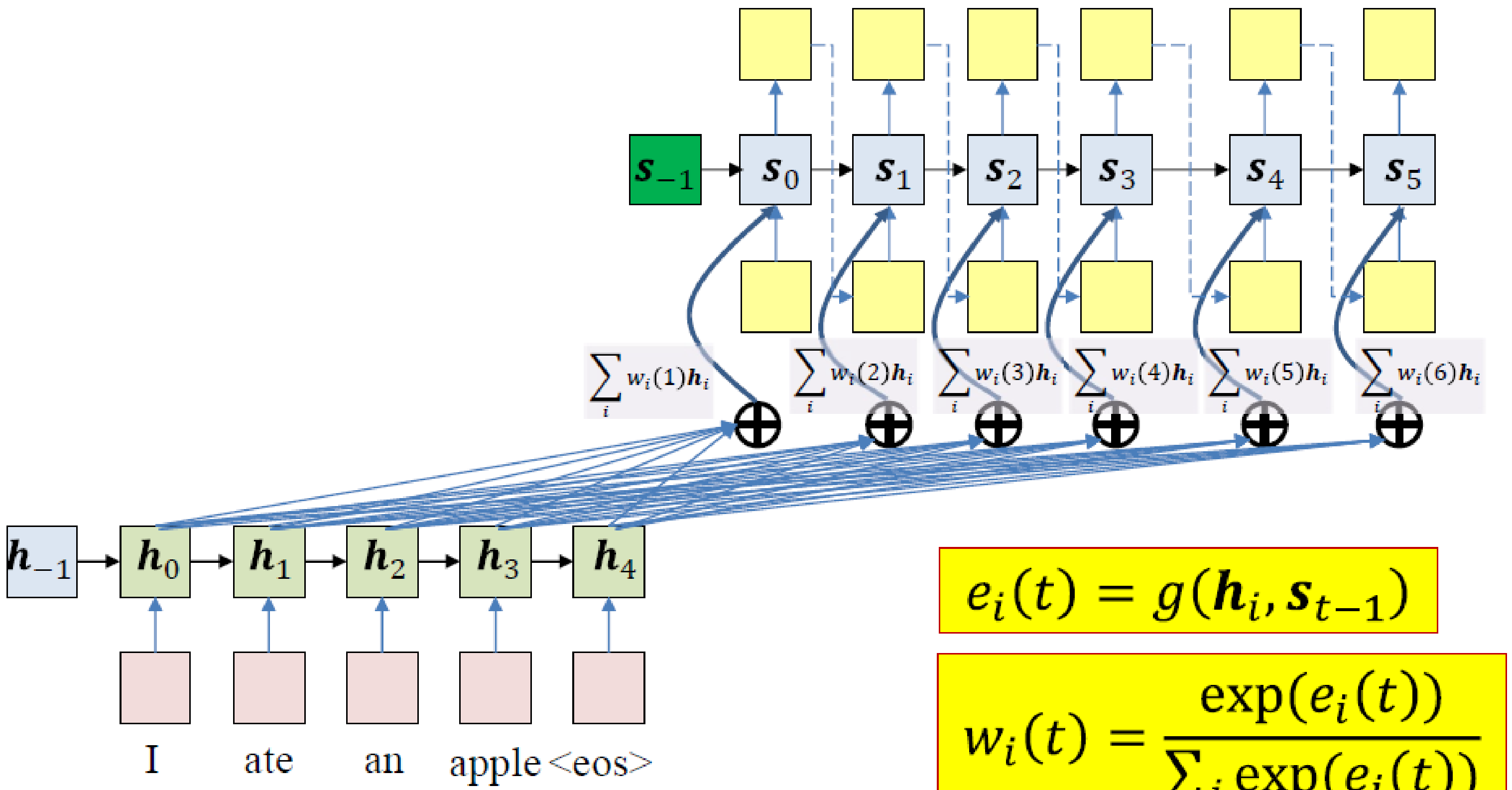


- Information bottleneck
  (all the information about the input seq is embedded into a single vector)
- In reality : all hidden values carry information
  -> some of which may be diluted downstream

Feasible solution :

Attention mechanism

Attention model



$$\boxed{e_i(t) = g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1})}$$

$$\boxed{w_i(t) = \frac{\exp(e_i(t))}{\sum_j \exp(e_j(t))}}$$

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = \boldsymbol{h}_i^T \boldsymbol{s}_{t-1}$$

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = \boldsymbol{h}_i^T \boldsymbol{W}_g \boldsymbol{s}_{t-1}$$

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = \boldsymbol{v}_g^T \boldsymbol{tanh}\left(\boldsymbol{W}_g \begin{bmatrix} \boldsymbol{h}_i \\ \boldsymbol{s}_{t-1} \end{bmatrix}\right)$$

$$g(\boldsymbol{h}_i, \boldsymbol{s}_{t-1}) = MLP([\boldsymbol{h}_i, \boldsymbol{s}_{t-1}])$$

- e : attention scores, w : attention weights
- Weights vary by output time
  (time-varying weight that specifies relationship of output time to input time)
- The weights are a distribution over the input

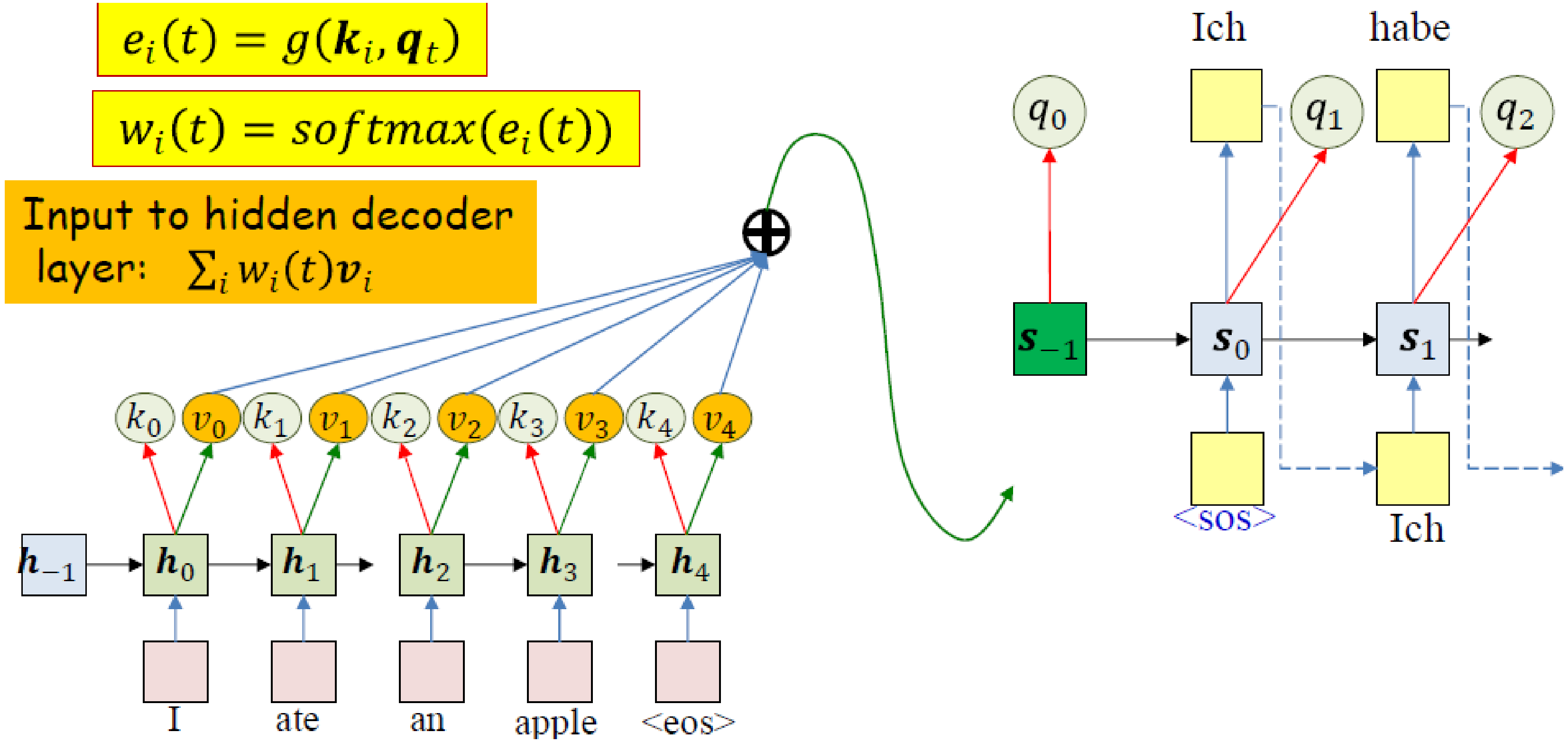
Attention model

<Query - Key - Value> : Generalize the Attention

$e_i(t) = g(\boldsymbol{k}_i, \boldsymbol{q}_t)$

$w_i(t) = softmax(e_i(t))$

Input to hidden decoder layer: $\sum_i w_i(t)\boldsymbol{v}_i$

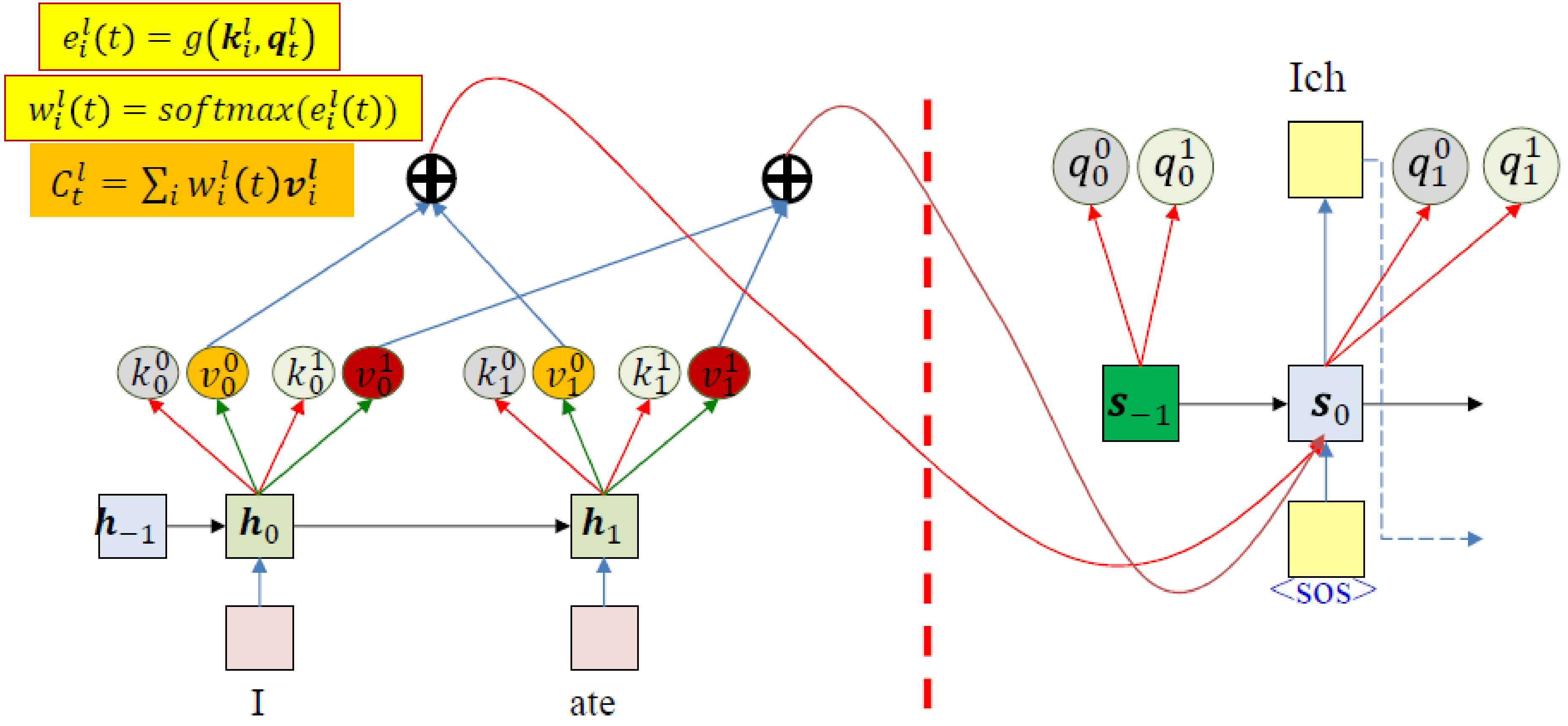


- The weight is a function of key and query
- The actual context is a weighted sum of value

Special case: $k_i = v_i = h_i$
$q_t = s_{t-1}$

Attention model

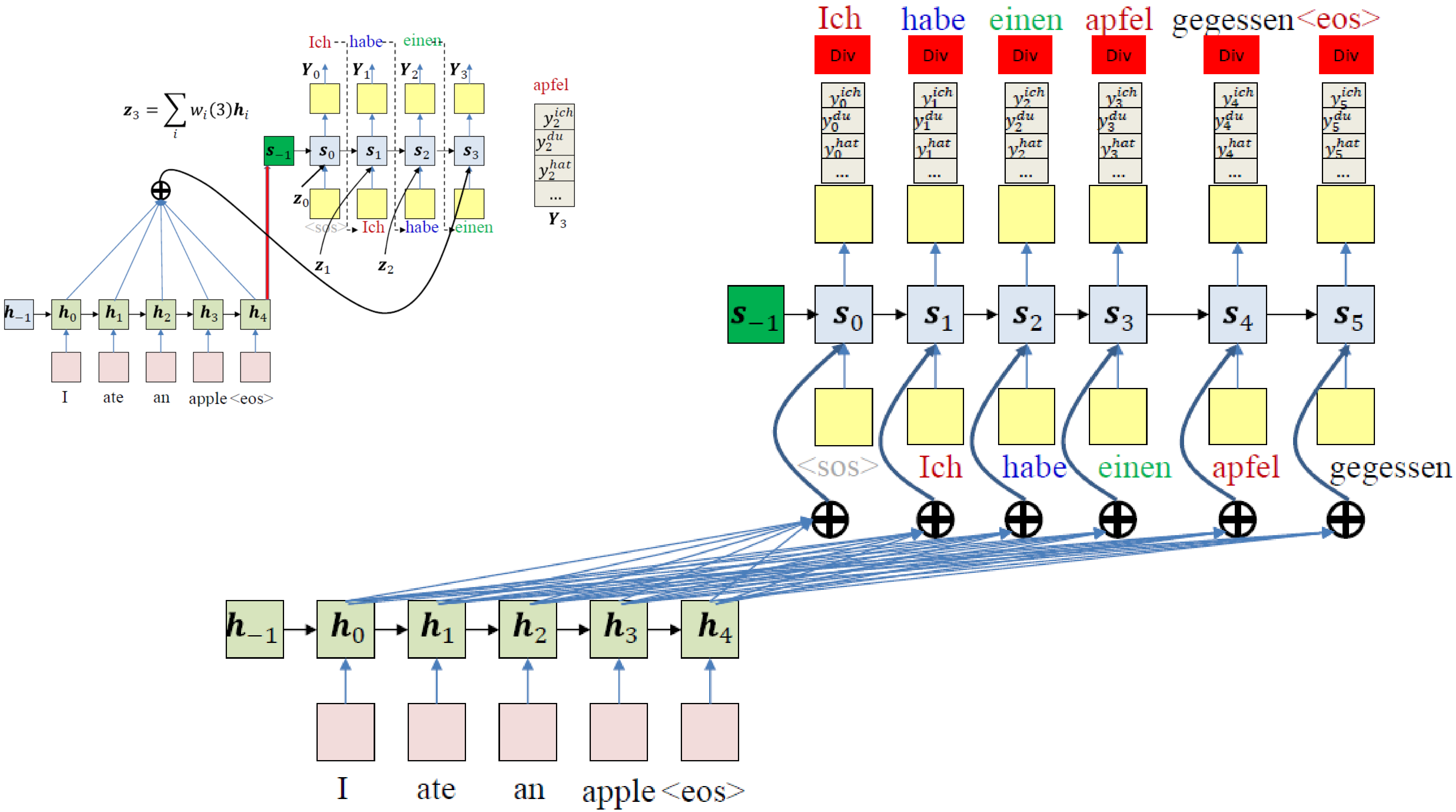
Multi-head attention



- Can have multiple Q/K/V sets (each attention head uses one of these sets)
- Each attender focuses on a different aspect of the input

Attention model

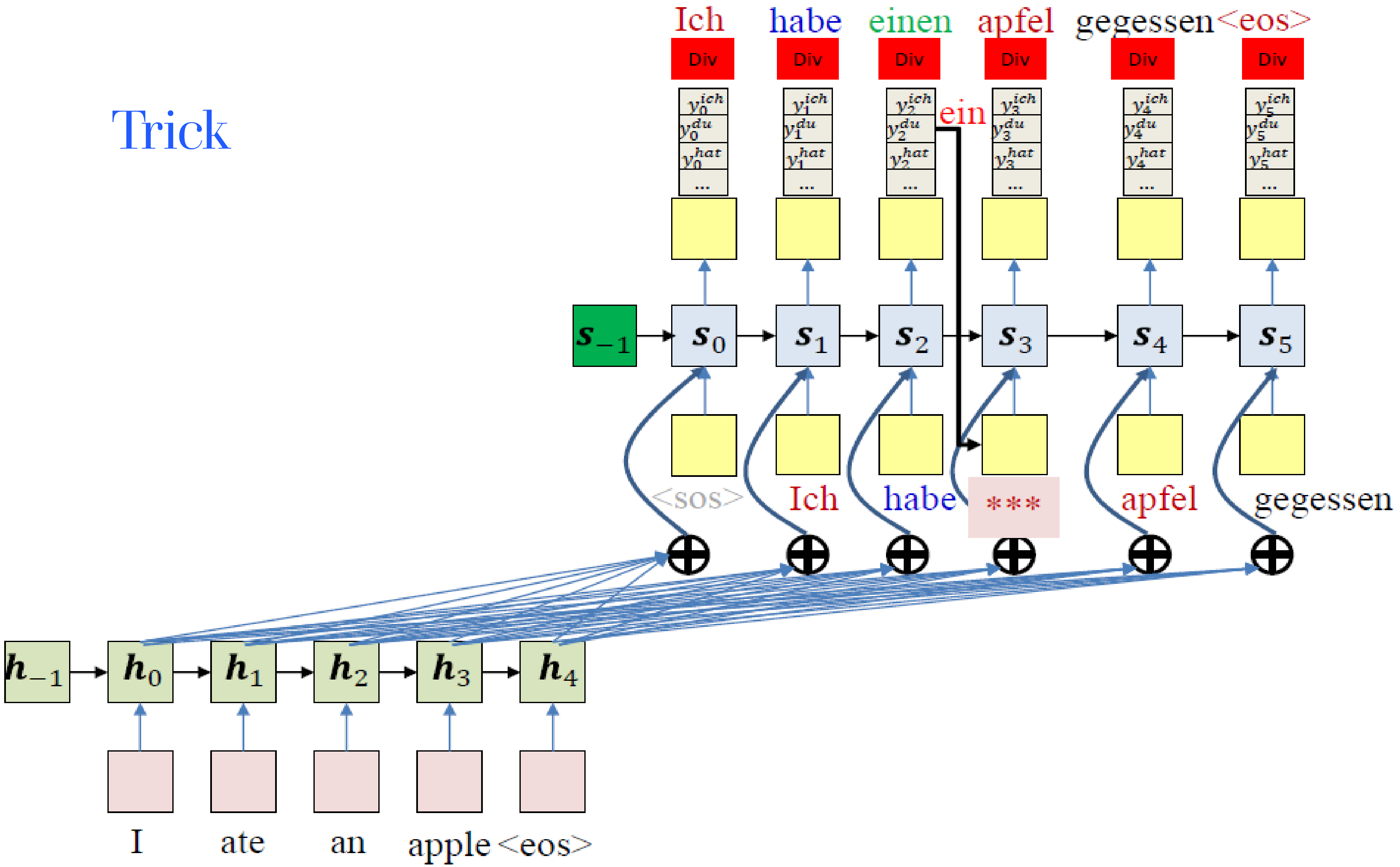
Inference & train



- If attention function is parametric ,
  back propagation also updates parameters of the attention function

Attention model

Trick



- Pass drawn output instead of ground truth, as input