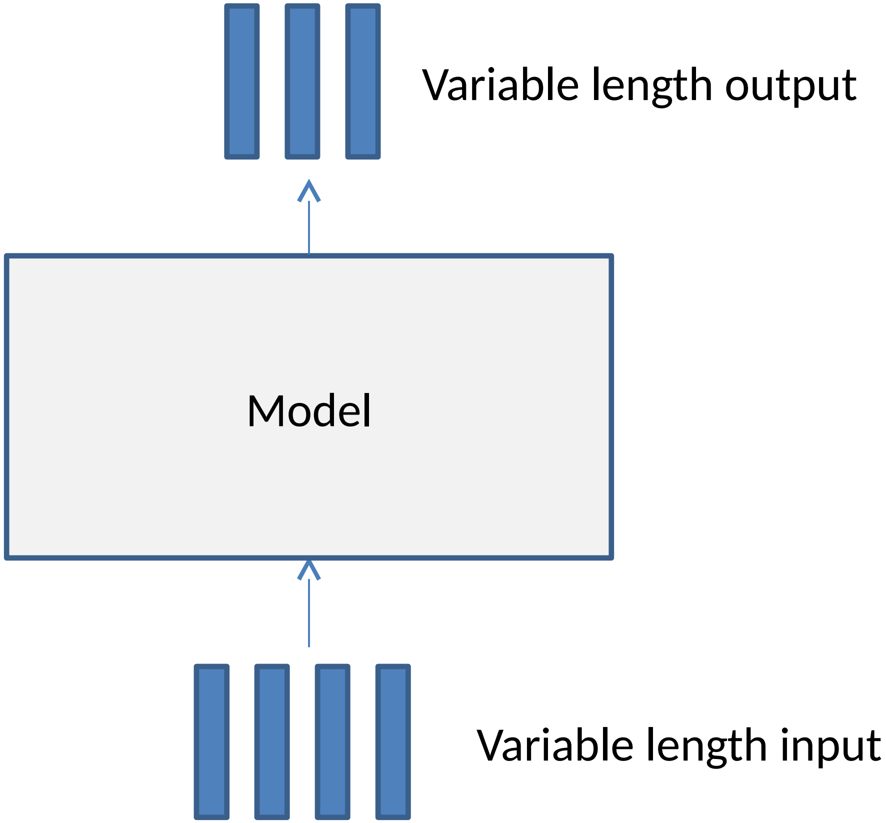
Sequnce(s)-to-Sequence Transformations in Text Processing

Narada Warakagoda

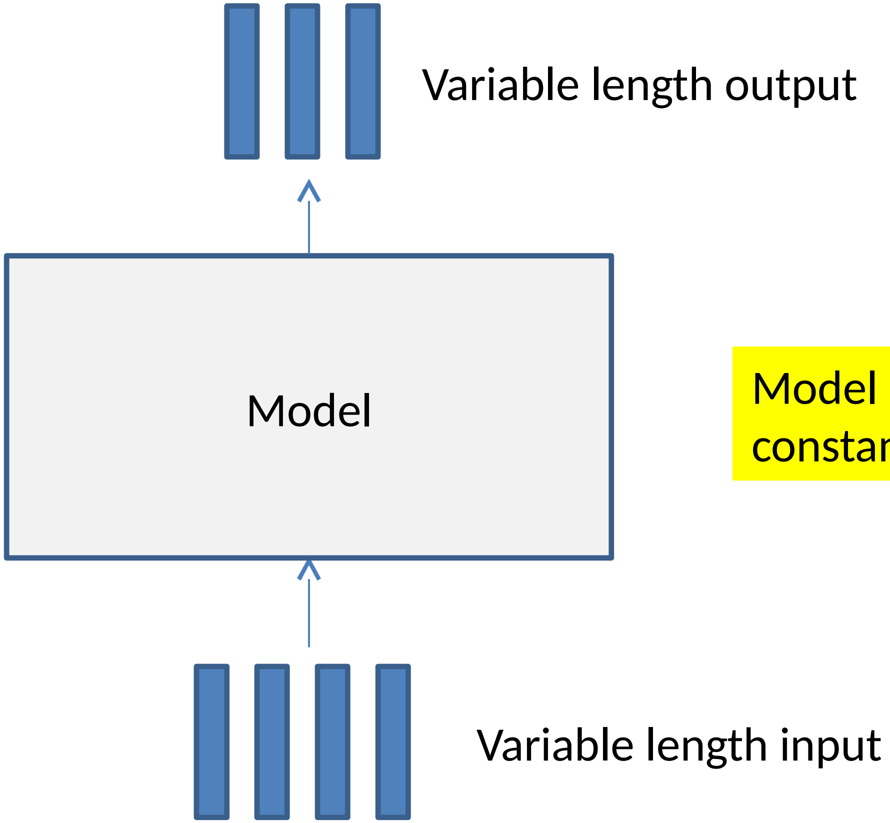
Seq2seq Transformation



Example Applications

- Summarization (extractive/abstractive)
- Machine translation
- Dialog systems /chatbots
- Text generation
- Question answering

Seq2seq Transformation



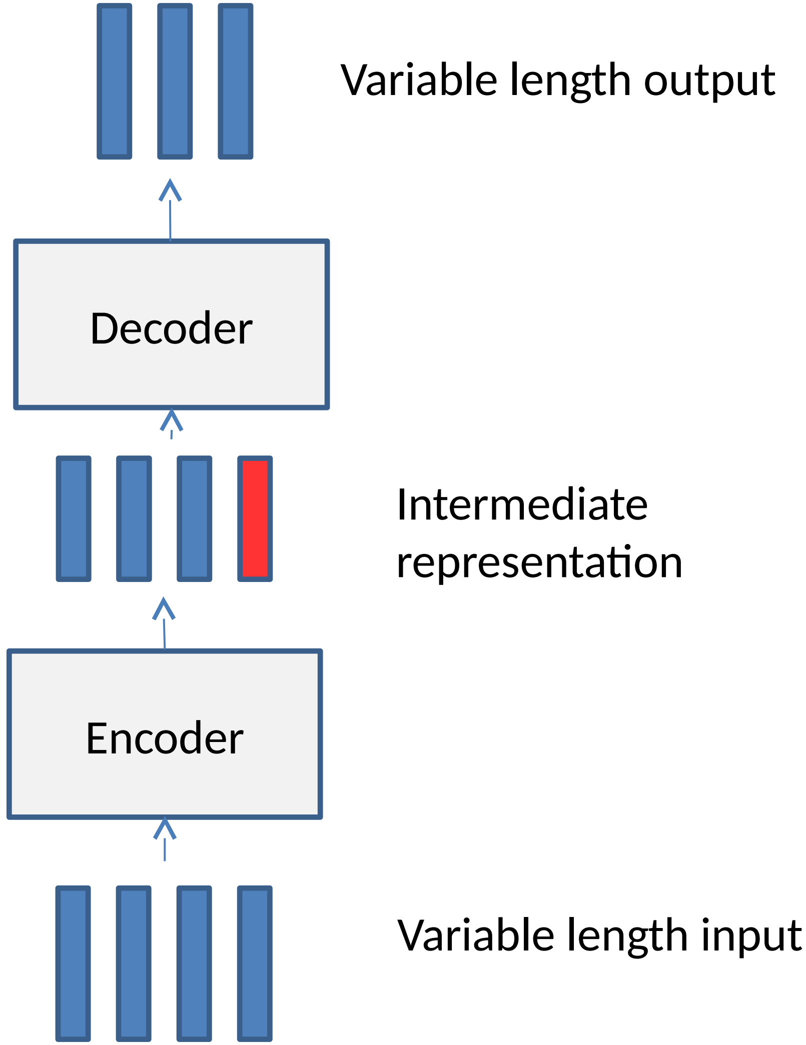
Model size should be constant.

Solution: Apply a constant sized neural net module repeatedly on the data

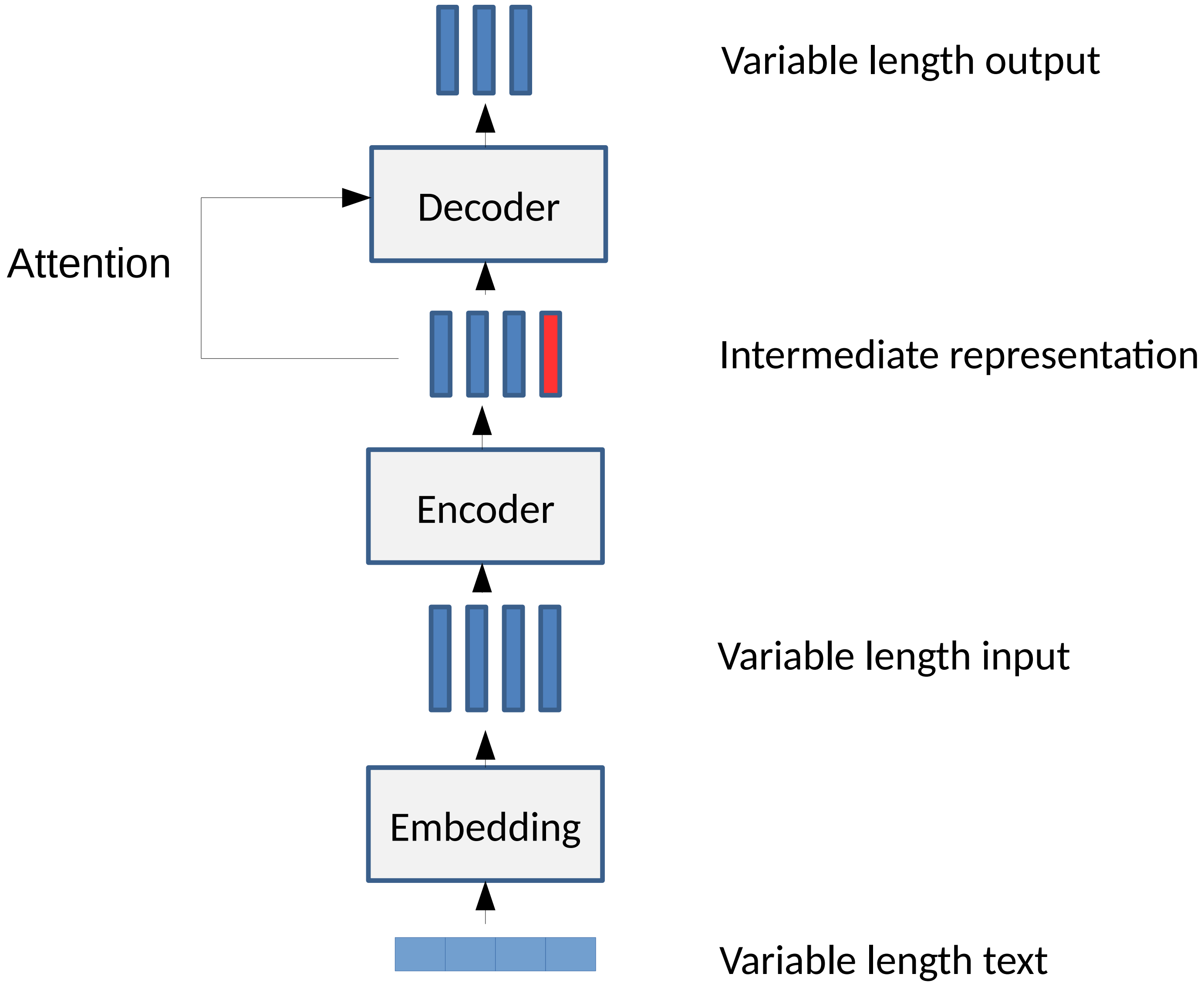
Possible Approaches

- Recurrent networks
  - Apply the NN module in a serial fashion
- Convolutions networks
  - Apply the NN modules in a hierarchical fashion
- Self-attention
  - Apply the NN module in a parallel fashion

Processing Pipeline



Processing Pipeline



Architecture Variants

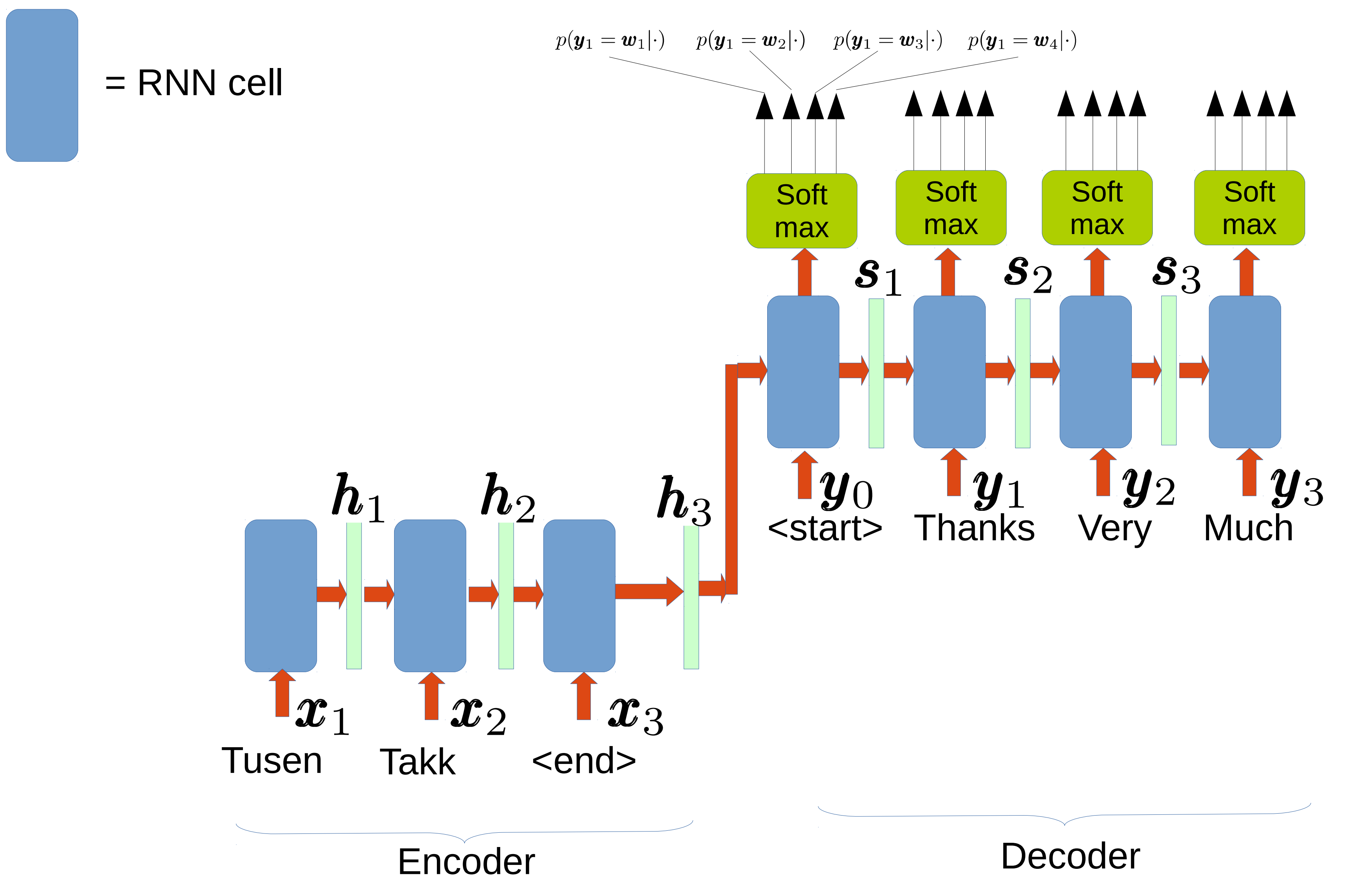
Encoder	Decoder	Attention
Recurrent net	Recurrent net	No
Recurrent net	Recurrent net	Yes
Convolutional net	Convolutional net	No
Convolutional net	Recurrent net	Yes
Convolutional net	Convolutional net	Yes
Fully connected net with self-attention	Fully connected net with self-attention	Yes

Possible Approaches

- Recurrent networks
  - Apply the NN module in a serial fashion
- Convolutions networks
  - Apply the NN modules in a hierarchical fashion
- Self-attention
  - Apply the NN module in a parallel fashion

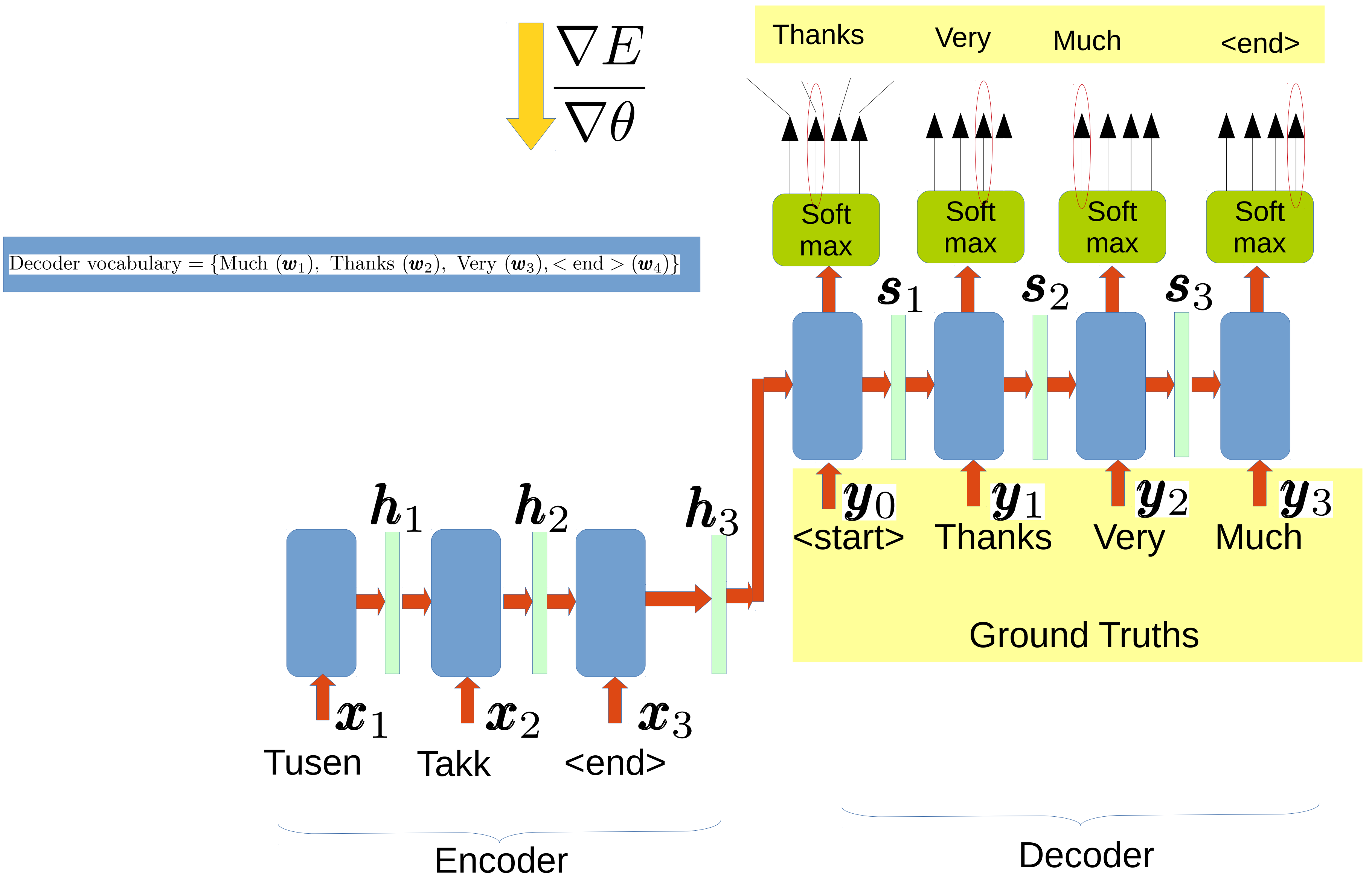
RNN-decoder with RNN-encoder

$$\text{Decoder vocabulary} = \{\text{Much } (\boldsymbol{w}_1),\ \text{Thanks } (\boldsymbol{w}_2),\ \text{Very } (\boldsymbol{w}_3), < \text{end} > (\boldsymbol{w}_4)\}$$



RNN-dec with RNN-enc, Training

$$E = \log L = \log \left[p(\boldsymbol{y}_1 = w_2 | \boldsymbol{X}) \cdot p(\boldsymbol{y}_2 = w_3 | w_2, \boldsymbol{X}) \cdot p(\boldsymbol{y}_3 = w_1 | w_2, w_3, \boldsymbol{X}) \cdot p(\boldsymbol{y}_4 = w_4 | w_2, w_3, w_1, \boldsymbol{X}) \right]$$

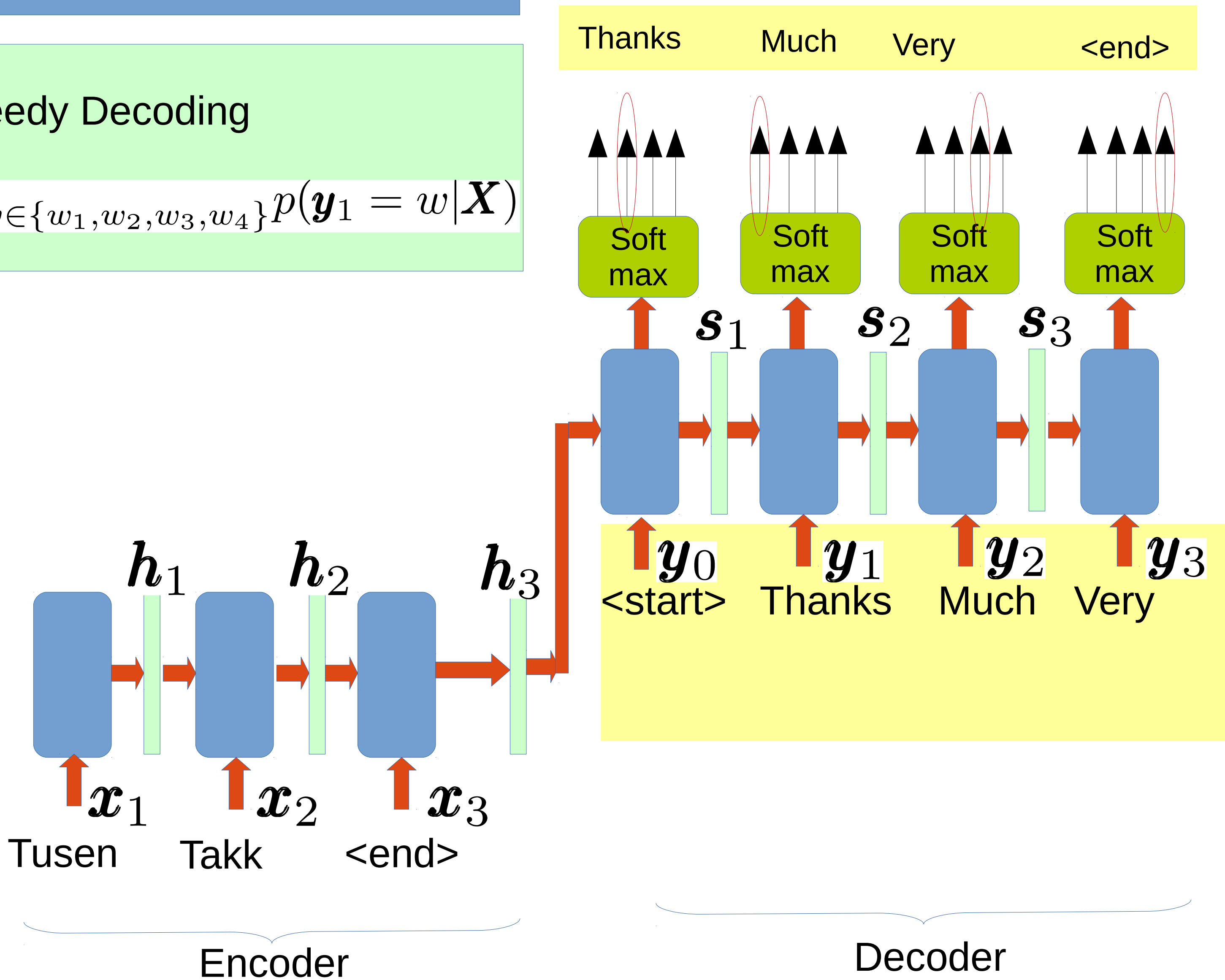


RNN-dec with RNN-enc, Decoding

Decoder vocabulary = {Much ($\boldsymbol{w}_1$), Thanks ($\boldsymbol{w}_2$), Very ($\boldsymbol{w}_3$), < end > ($\boldsymbol{w}_4$)}

Greedy Decoding

$$\boldsymbol{y}_1 = \text{argmax}_{w \in \{w_1, w_2, w_3, w_4\}} p(\boldsymbol{y}_1 = w | \boldsymbol{X})$$



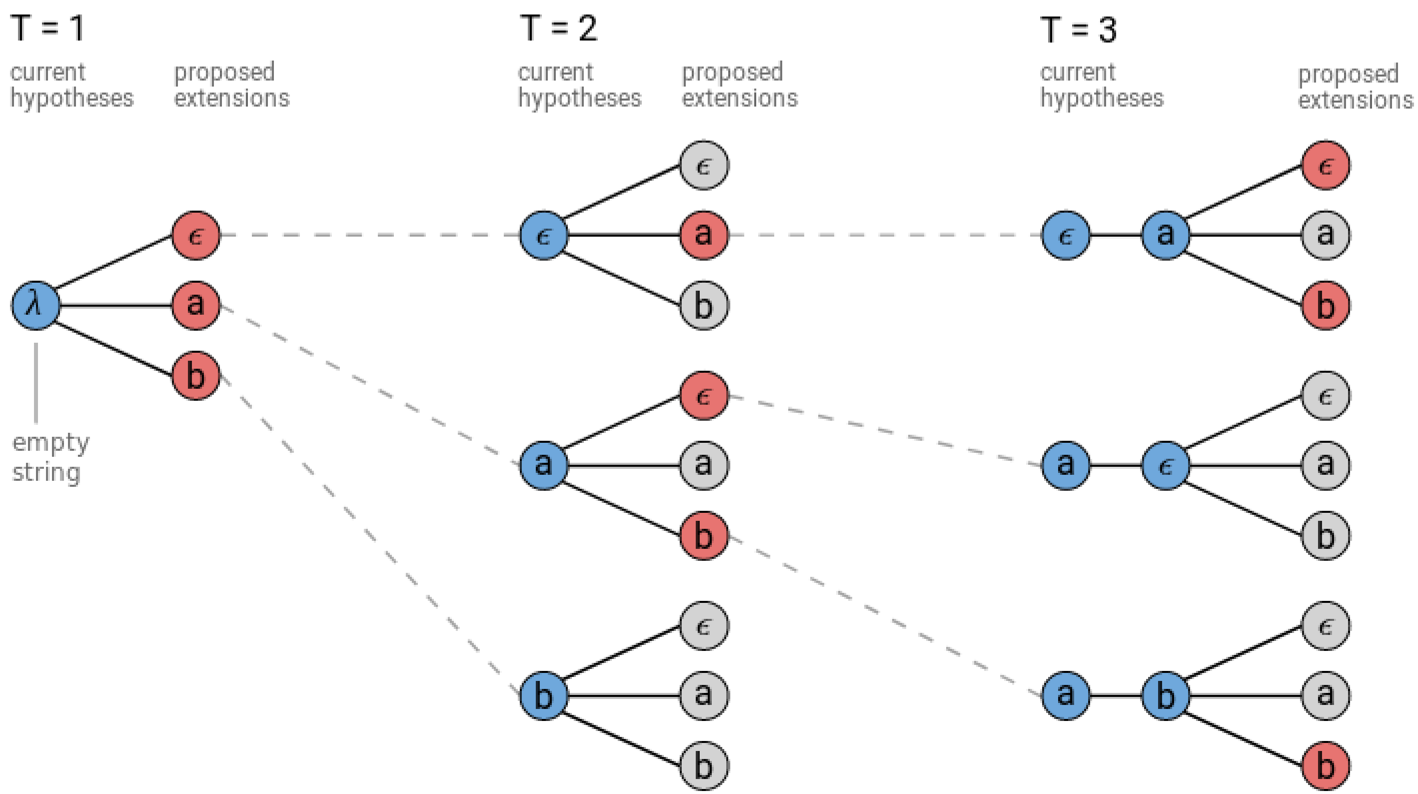
Decoding Approaches

- Optimal decoding

  Find $\boldsymbol{w} = \{w_1, w_2, w_3, w_4\}$ such that $p(w_1, w_2, w_3, w_4|\boldsymbol{X})$ is maximum

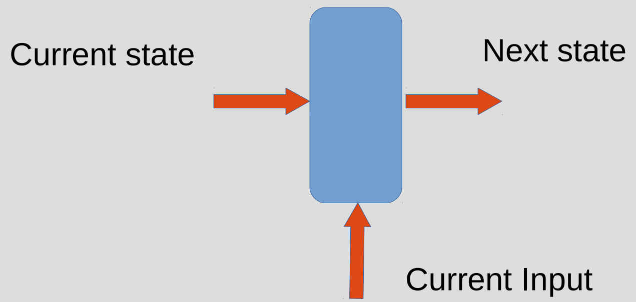
- Greedy decoding
  - Easy
  - Not optimal
- Beam search
  - Closer to optimal decoder
  - Choose top N candidates instead of the best one at each step.

Beam Search Decoding

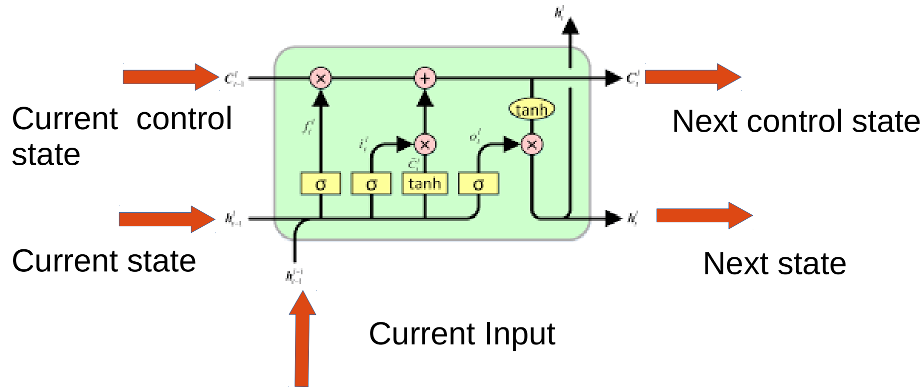


Beam Width = 3

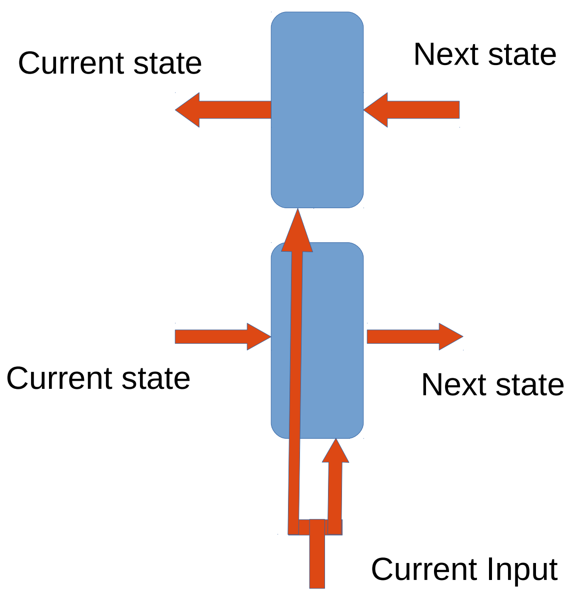
Straight-forward Extensions



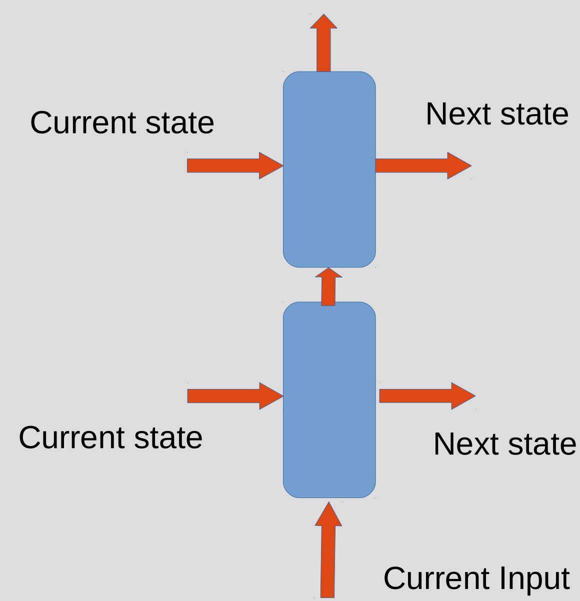
RNN Cell



LSTM Cell



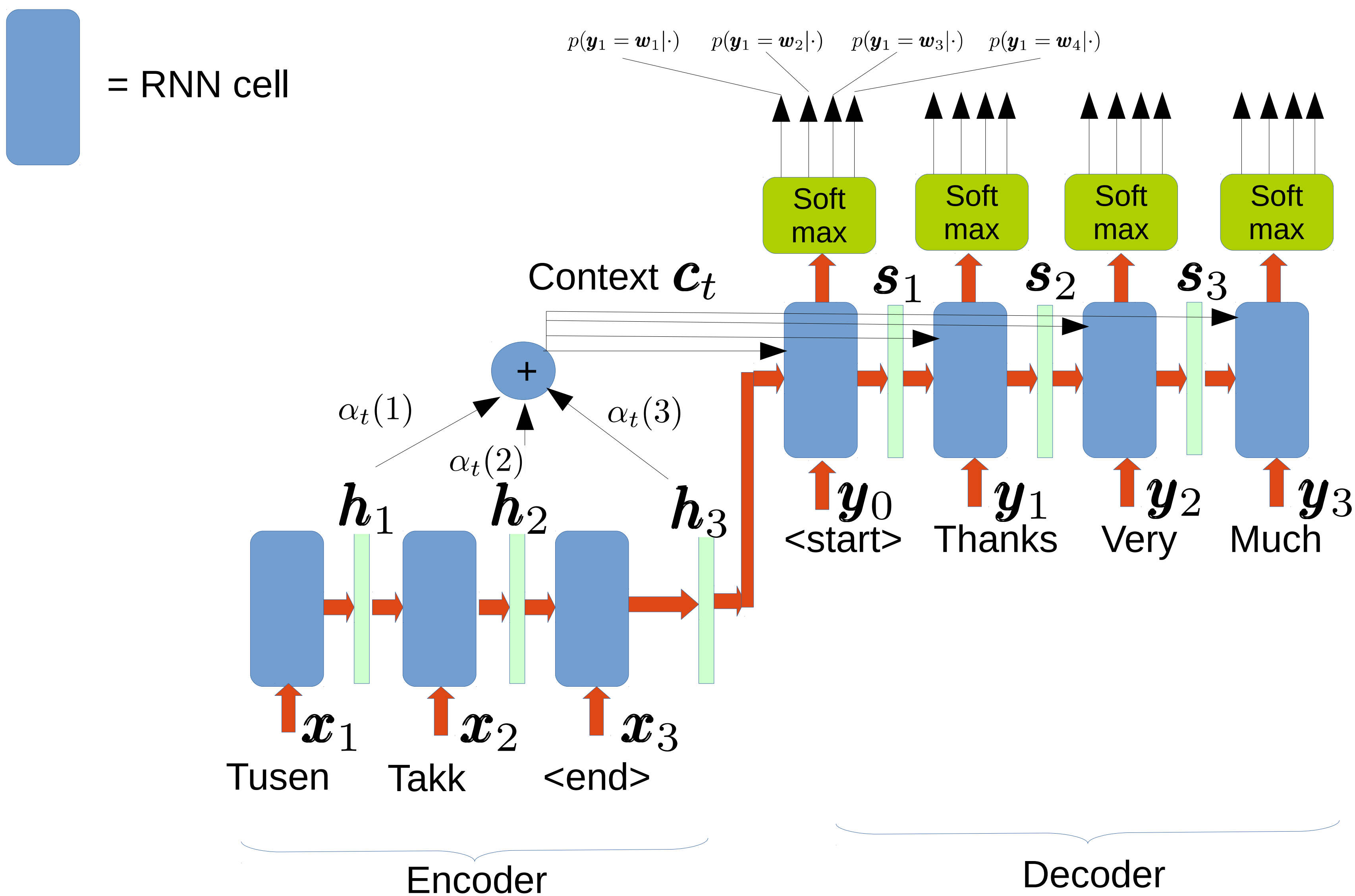
Bidirectional Cell



Stacked Cell

RNN-decoder with RNN-encoder with Attention

$$\text{Decoder vocabulary} = \{\text{Much } (\boldsymbol{w}_1),\ \text{Thanks } (\boldsymbol{w}_2),\ \text{Very } (\boldsymbol{w}_3), <\text{end}> (\boldsymbol{w}_4)\}$$



Attention

- Context is given by $\boldsymbol{c}_t = \sum_{j=1}^{T_x} \alpha_t(j) \boldsymbol{h}_j$

- Attention weights $\alpha_t(j)$ are dynamic
- Generally defined by $\alpha_t(j) = \frac{\exp(e_t(j))}{\sum_{k=1}^{T_x} \exp(e_t(k))}$ with $e_t(j) = f(\boldsymbol{s}_{t-1}, \boldsymbol{h}_j)$

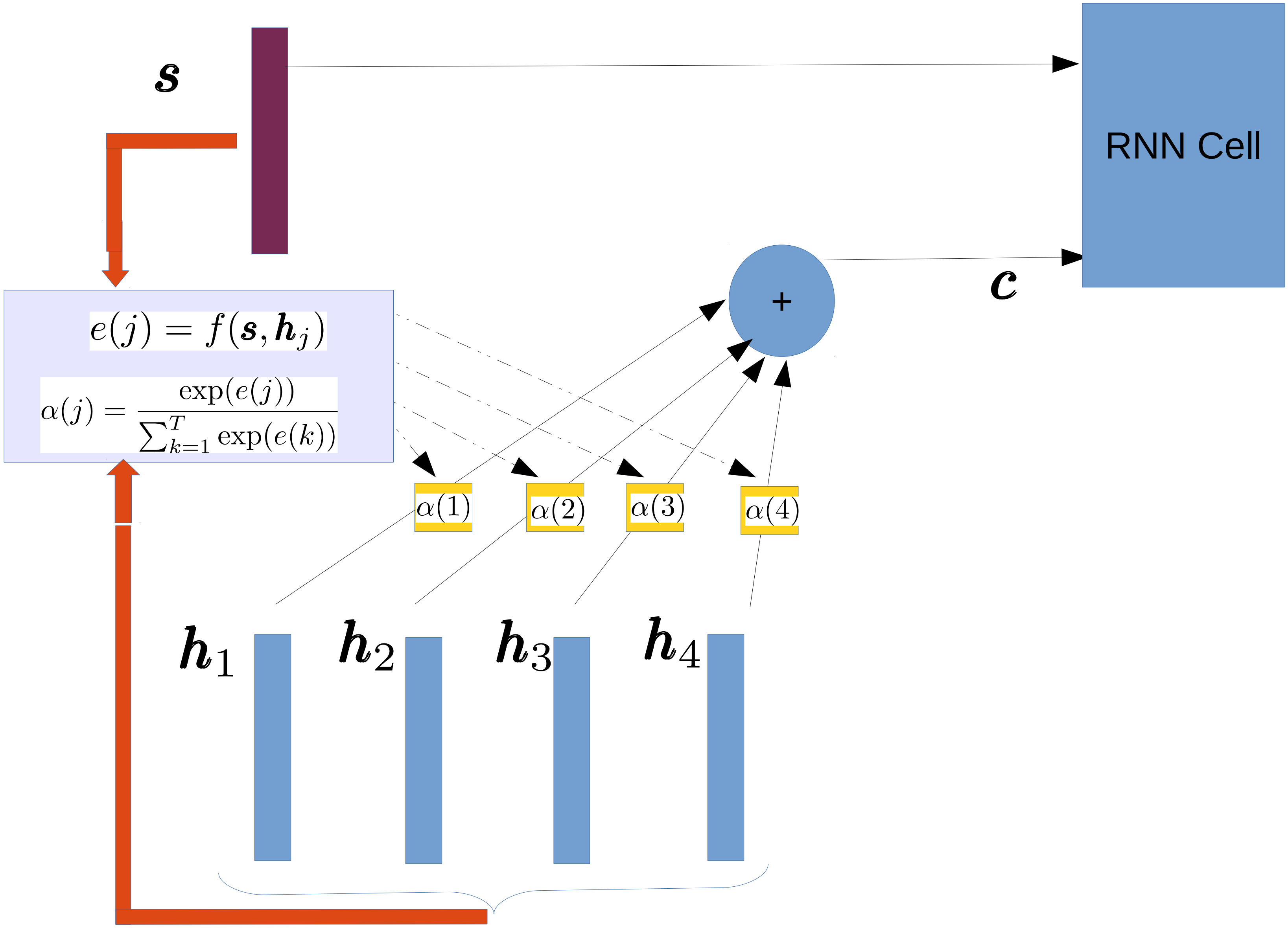
  where function f can be defined in several ways.

  - Dot product $e_t(j) = \boldsymbol{s}_{t-1}^T \cdot \boldsymbol{h}_j$

  - Weighted dot product $e_t(j) = \boldsymbol{s}_{t-1}^T \cdot \boldsymbol{W} \cdot \boldsymbol{h}_j$

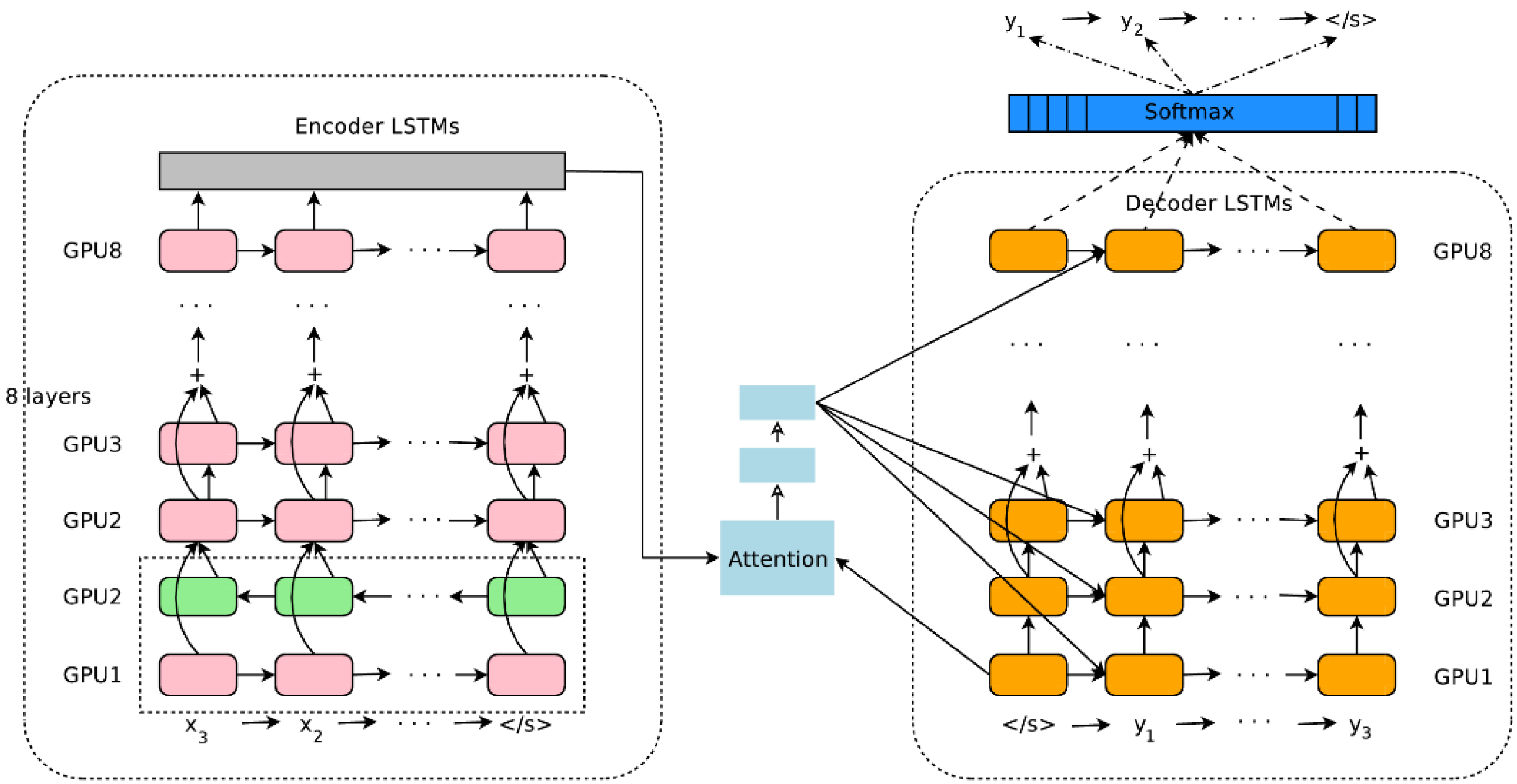
  - Use another MLP (eg: 2 layer) $e_t(j) = \boldsymbol{v}^T \cdot \tanh(\boldsymbol{W} \cdot [\boldsymbol{h}_j; \boldsymbol{s}_{t-1}])$

Attention



$\boldsymbol{s}$
RNN Cell
+
$\boldsymbol{c}$
$e(j) = f(\boldsymbol{s}, \boldsymbol{h}_j)$
$\alpha(j) = \frac{\exp(e(j))}{\sum_{k=1}^{T} \exp(e(k))}$
$\alpha(1)$
$\alpha(2)$
$\alpha(3)$
$\alpha(4)$
$\boldsymbol{h}_1$
$\boldsymbol{h}_2$
$\boldsymbol{h}_3$
$\boldsymbol{h}_4$

Example: Google Neural Machine Translation



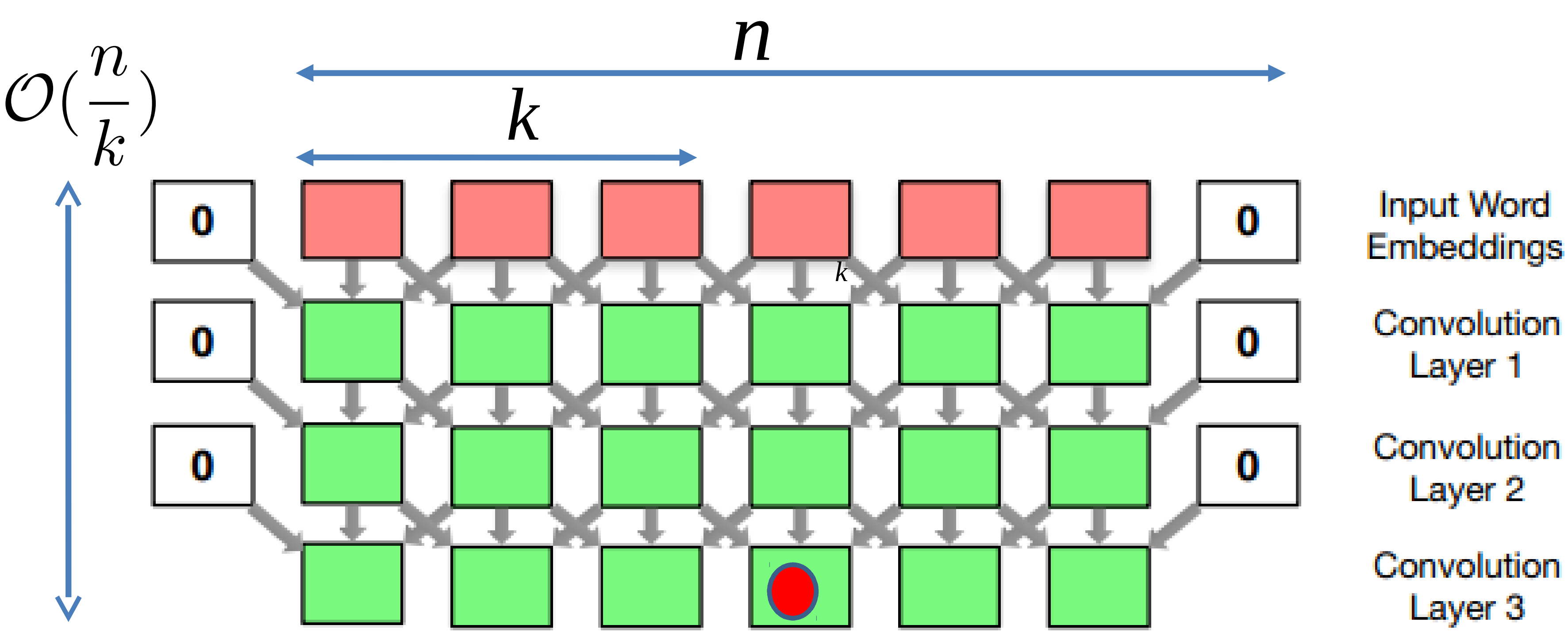
Possible Approaches

- Recurrent networks
  - Apply the NN module in a serial fashion
- Convolutions networks
  - Apply the NN modules in a hierarchical fashion
- Self-attention
  - Apply the NN module in a parallel fashion

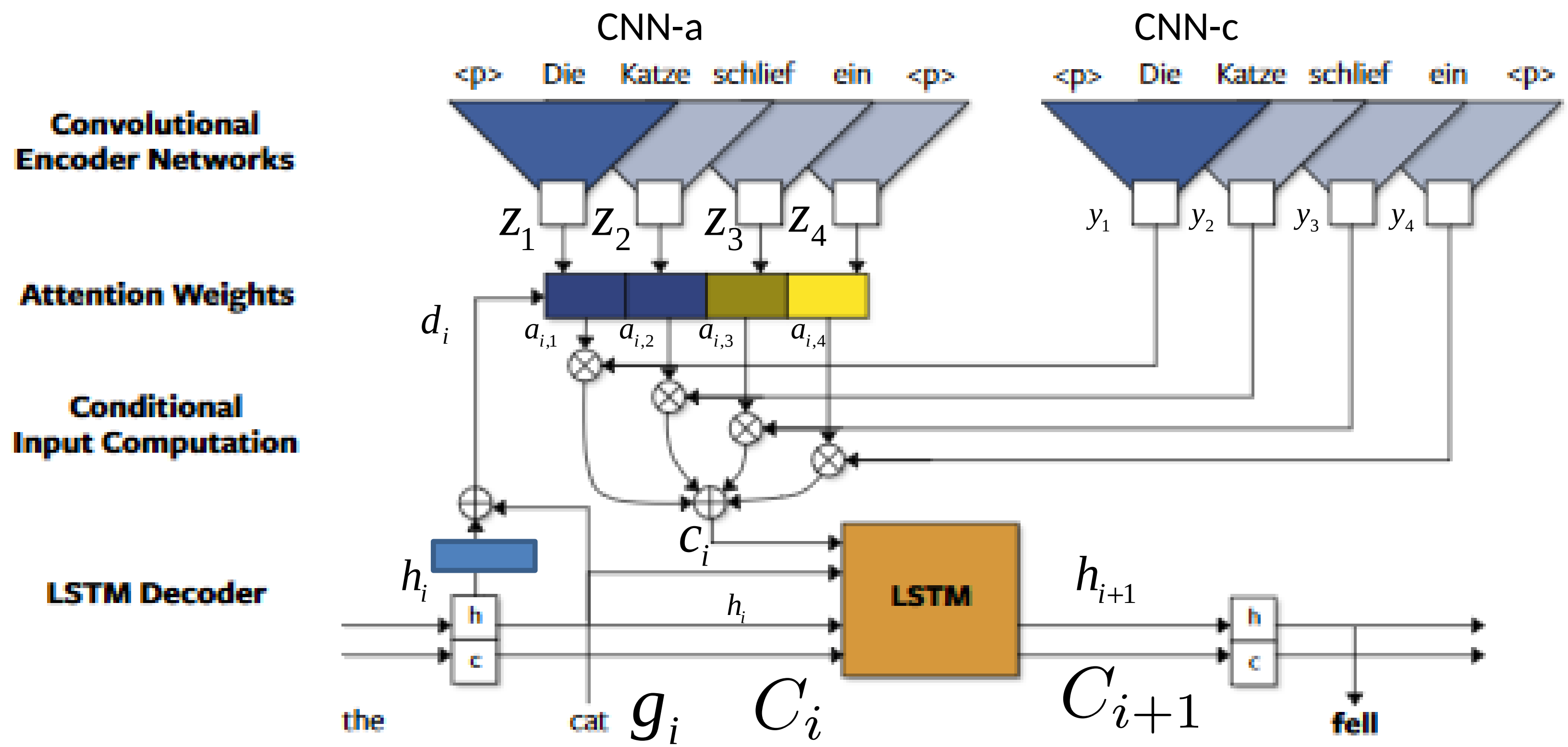
Why Convolution

- Recurrent networks are serial
  - Unable to be parallelized
  - “Distance” between feature vector and different inputs are not constant
- Convolutions networks
  - Can be parallelized (faster)
  - “Distance” between feature vector and different inputs are constant

Long range dependency capture with conv nets



Conv net, Recurrent net with Attention



$$d_i = W_d h_i + g_i$$

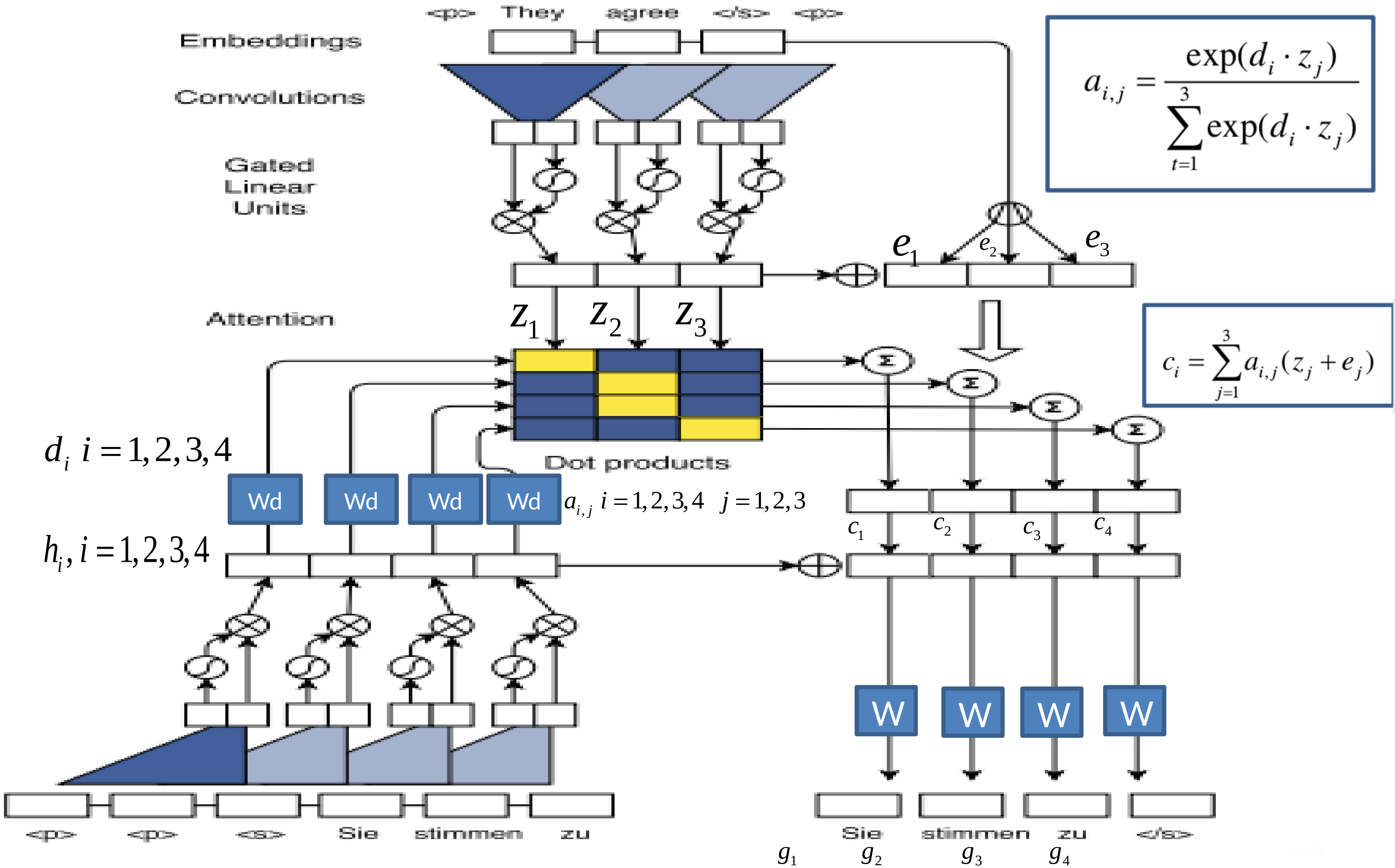
$$a_{i,j} = \frac{\exp(d_i \cdot z_j)}{\sum_{t=1}^{4} \exp(d_i \cdot z_t)}$$

$$c_i = \sum_{j=1}^{4} a_{i,j} y_j$$

$$h_{i+1}, C_{i+1} = \mathrm{LSTM}(c_i, h_i, g_i, C_i)$$

Gehring et.al. *A Convolutional Encoder Model for Neural Machine Translation (2016)*

Two conv nets with attention



$$a_{i,j} = \frac{\exp(d_i \cdot z_j)}{\sum_{t=1}^{3} \exp(d_i \cdot z_j)}$$

$$c_i = \sum_{j=1}^{3} a_{i,j}(z_j + e_j)$$

e_1 e_2 e_3

z_1 z_2 z_3

$d_i \; i = 1,2,3,4$

$h_i, i = 1,2,3,4$

$a_{i,j} \; i = 1,2,3,4 \quad j = 1,2,3$

c_1 c_2 c_3 c_4

g_1 g_2 g_3 g_4

$$P(g_i \mid g_{i-1}, g_{i-2}, \ldots) = \text{softmax}(W(c_{i-1} + h_{i-1}))$$

Gehring et.al, *Convolutional Sequence to Sequence Learning*, 2017

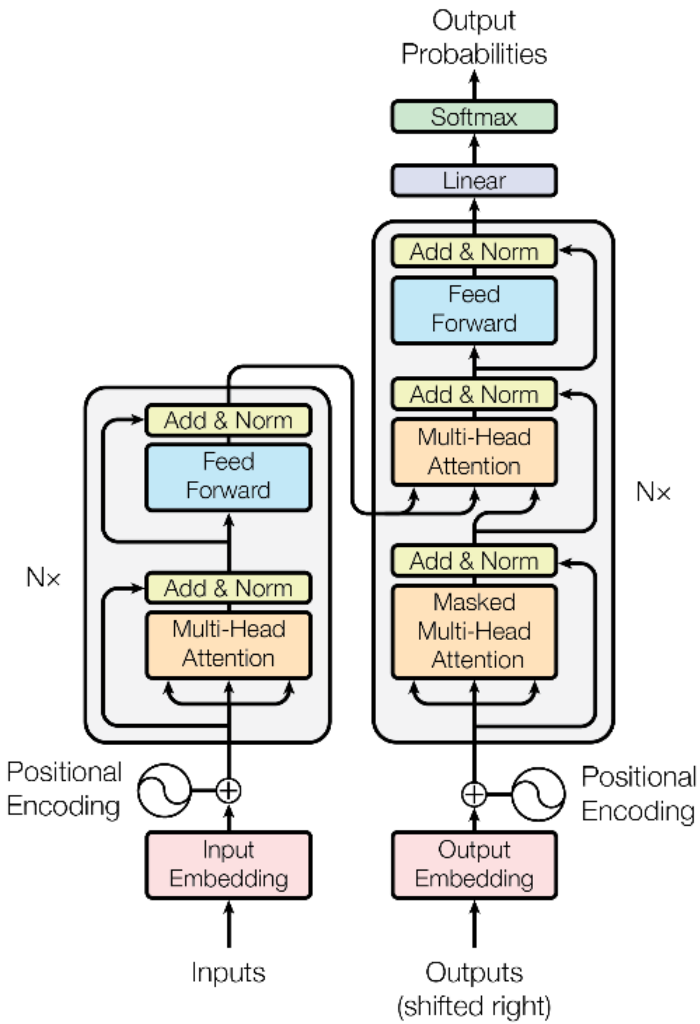
Possible Approaches

- Recurrent networks
  - Apply the NN module in a serial fashion
- Convolutions networks
  - Apply the NN modules in a hierarchical fashion
- Self-attention
  - Apply the NN module in a parallel fashion

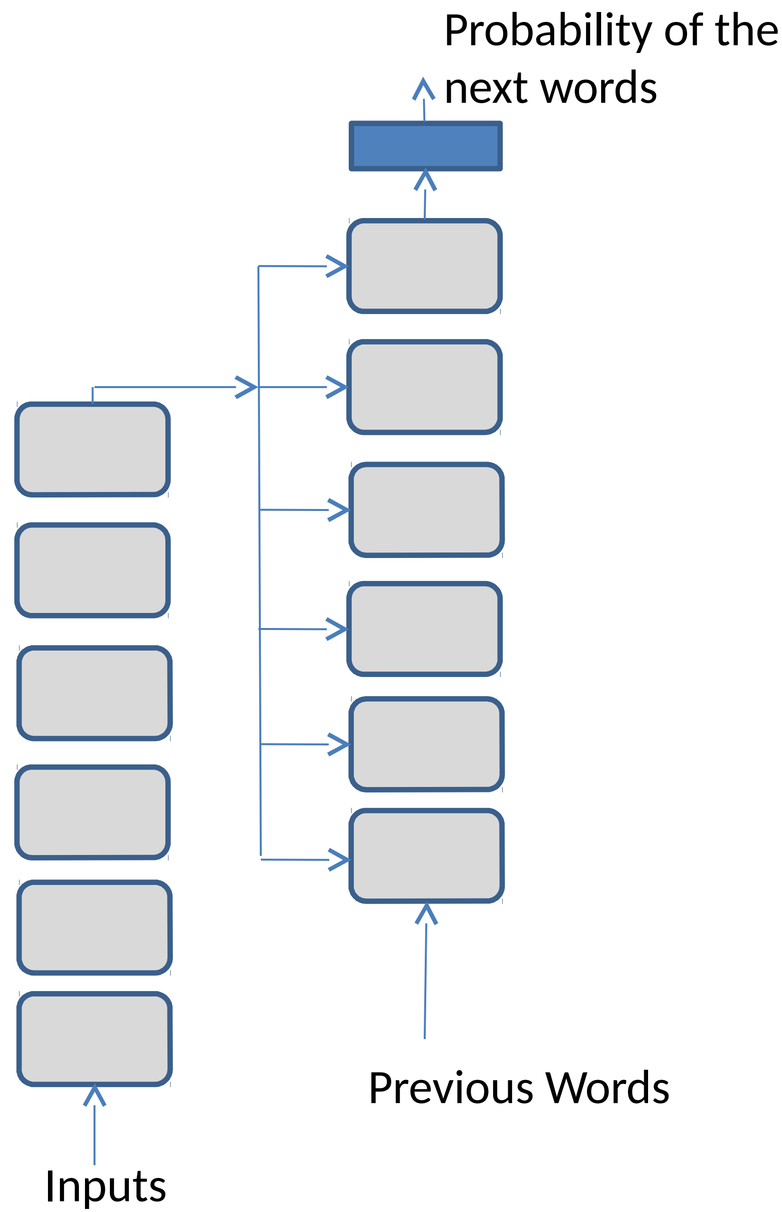
Why Self-attention

- Recurrent networks are serial
  - Unable to be parallelized
  - “Distance” between feature vector and different inputs are not constant
- Self-attention networks
  - Can be parallelized (faster)
  - “Distance” between feature vector and different inputs does not depend on the input length

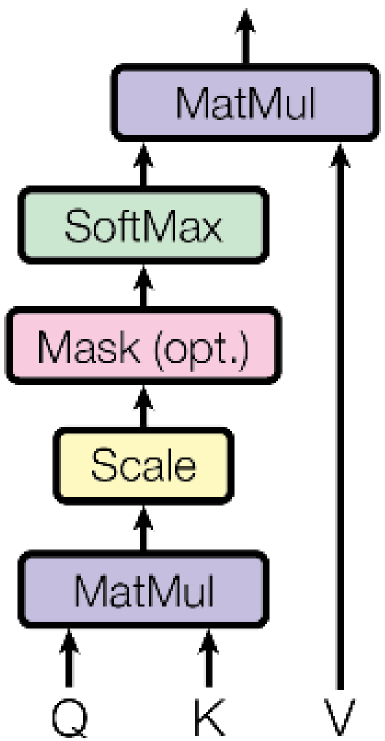
FCN with self-attention



Vasvani et.al, *Attention is all you need*, 2017



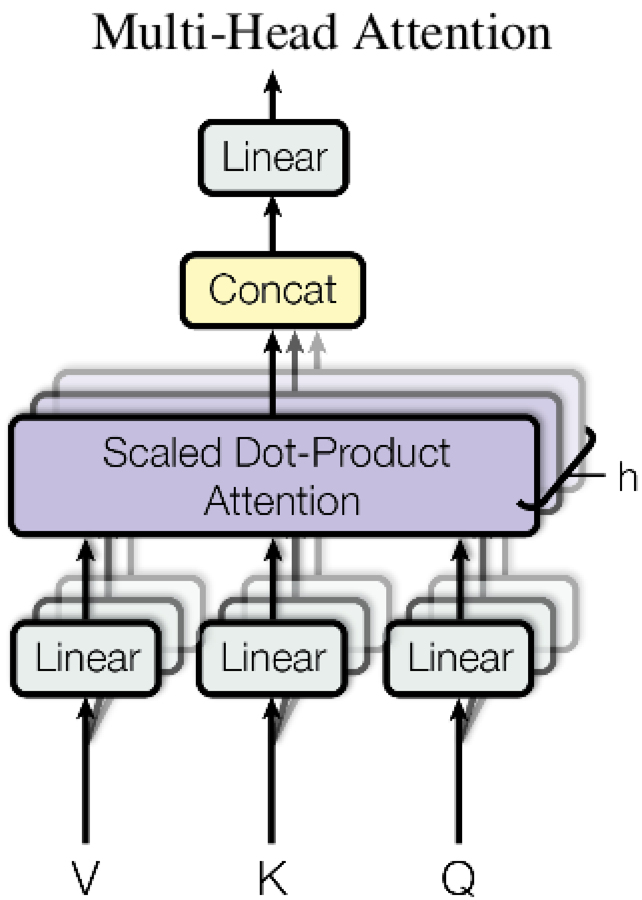
Scaled dot product attention



Query Keys Values

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-Head Attention



Encoder Self-attention

Encoder Self-Attention

$\boldsymbol{q}_1 \quad \boldsymbol{q}_2 \quad \boldsymbol{q}_3$

$\boldsymbol{k}_1 \quad \boldsymbol{k}_2 \quad \boldsymbol{k}_3$

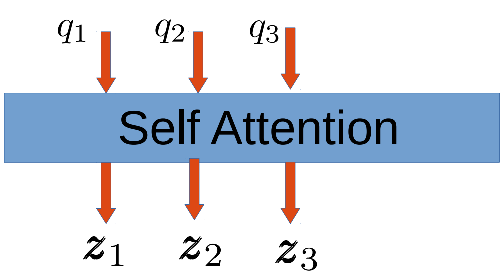
$\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \boldsymbol{v}_3$

$$\alpha_{1,1} = \frac{\exp(\boldsymbol{q}_1 \boldsymbol{k}_1^T)}{\sum_j \exp(\boldsymbol{q}_1 \boldsymbol{k}_j^T)} \qquad \alpha_{1,2} = \frac{\exp(\boldsymbol{q}_1 \boldsymbol{k}_2^T)}{\sum_j \exp(\boldsymbol{q}_1 \boldsymbol{k}_j^T)} \qquad \alpha_{1,3} = \frac{\exp(\boldsymbol{q}_1 \boldsymbol{k}_3^T)}{\sum_j \exp(\boldsymbol{q}_1 \boldsymbol{k}_j^T)}$$

$$\boldsymbol{z}_1 = \alpha_{1,1}\boldsymbol{v}_1 + \alpha_{1,2}\boldsymbol{v}_2 + \alpha_{1,3}\boldsymbol{v}_3$$

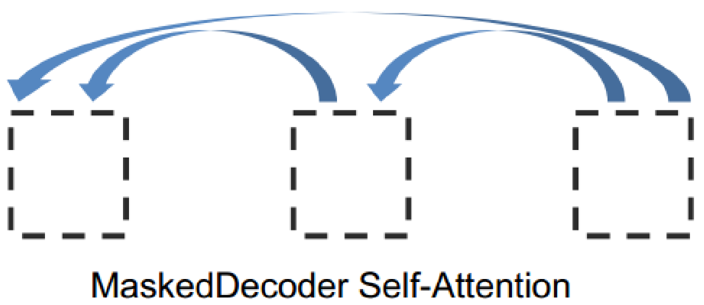
$$\boldsymbol{z}_2 = \alpha_{2,1}\boldsymbol{v}_1 + \alpha_{2,2}\boldsymbol{v}_2 + \alpha_{2,3}\boldsymbol{v}_3$$

$$\boldsymbol{z}_3 = \alpha_{3,1}\boldsymbol{v}_1 + \alpha_{3,2}\boldsymbol{v}_2 + \alpha_{3,3}\boldsymbol{v}_3$$



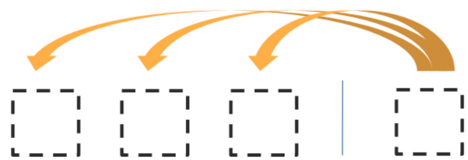
Decoder Self-attention

- Almost same as encoder self attention
- But only leftward positions are considered.



Encoder-decoder attention

Encoder states   Decoder state

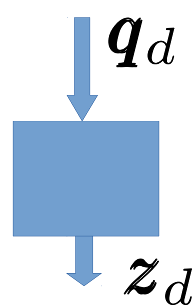


Encoder-Decoder Attention

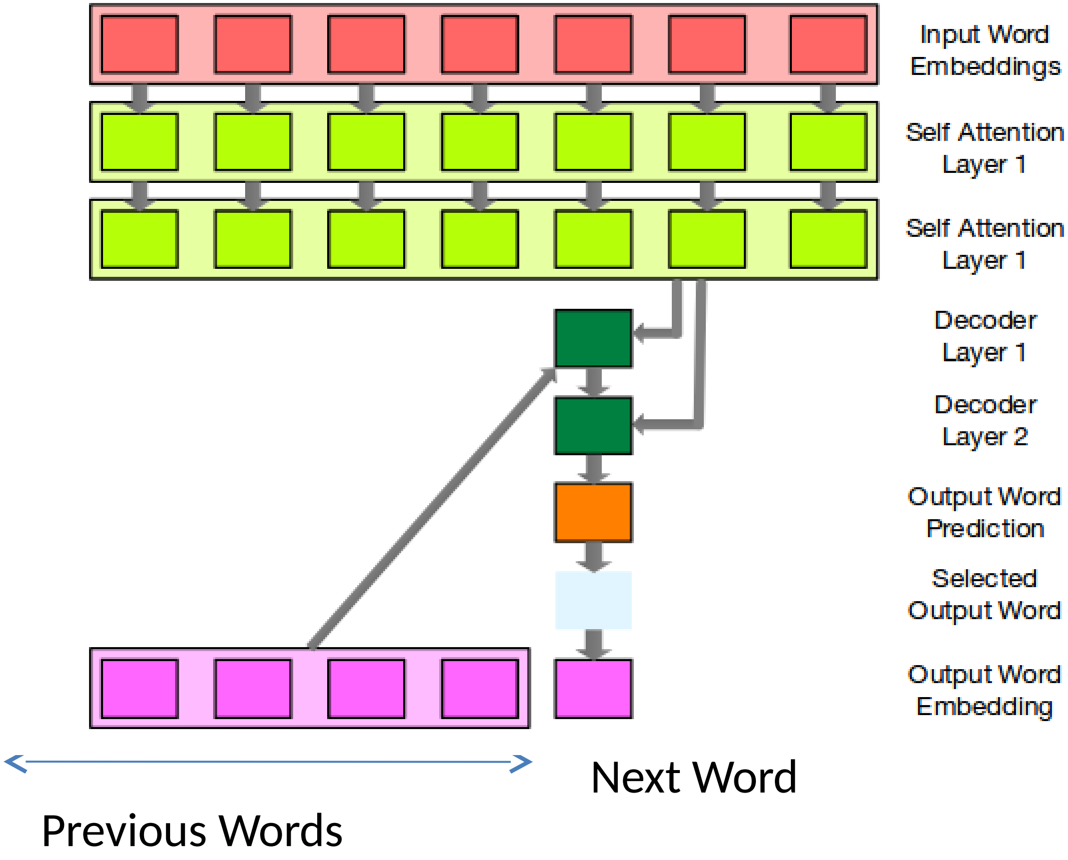
$$
\begin{array}{cccc}
\boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \boldsymbol{q}_d \\
\boldsymbol{k}_1 & \boldsymbol{k}_2 & \boldsymbol{k}_3 & \boldsymbol{k}_d \\
\boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 & \boldsymbol{v}_d
\end{array}
$$

$$\alpha_{1,1} = \frac{\exp(\boldsymbol{q}_d \boldsymbol{k}_1^T)}{\sum_j \exp(\boldsymbol{q}_d \boldsymbol{k}_j^T)} \qquad \alpha_{1,2} = \frac{\exp(\boldsymbol{q}_d \boldsymbol{k}_2^T)}{\sum_j \exp(\boldsymbol{q}_d \boldsymbol{k}_j^T)} \qquad \alpha_{1,3} = \frac{\exp(\boldsymbol{q}_d \boldsymbol{k}_3^T)}{\sum_j \exp(\boldsymbol{q}_d \boldsymbol{k}_j^T)}$$

$$\boldsymbol{z}_d = \alpha_{1,1}\boldsymbol{v}_1 + \alpha_{1,2}\boldsymbol{v}_2 + \alpha_{1,3}\boldsymbol{v}_3$$



Overall Operation



Neural machine translation, philipp Koehn

Reinforcement Learning

- Machine Translation/Summarization
- Dialog Systems
-
-

Reinforcement Learning

- Machine Translation/Summarization
- Dialog Systems
-
-

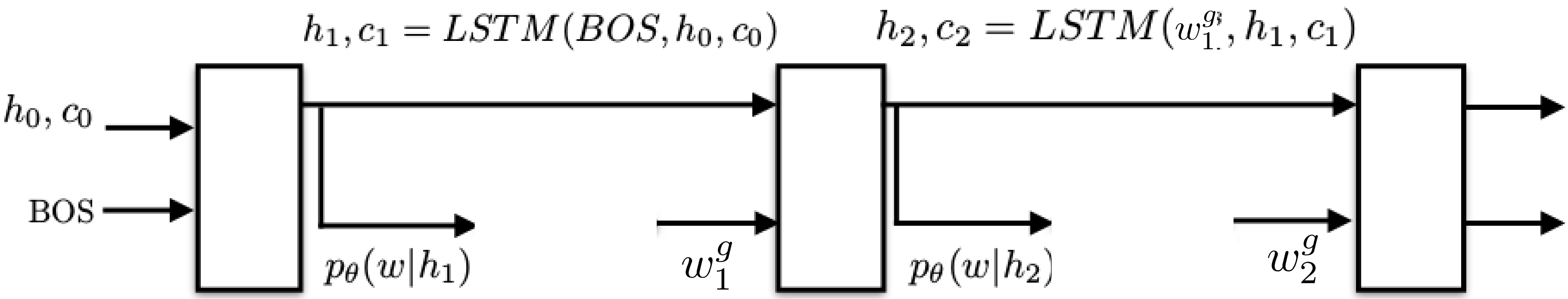
Why Reinforcement Learning

- Exposure bias
  - In training ground truths are used. In testing, generated word in the previous step is used to generate the next word.
  - Use generated words in training needs sampling : Non differentiable
- Maximum Likelihood criterion is not directly relevant to evaluation metrics
  - BLEU (Machine translation)
  - ROUGE (Summarization)
  - Use BLEU/ROUGE in training: Non differentiable

Sequence Generation as Reinforcement Learning

- **Agent:** The Recurrent Net
- **State:** Hidden layers, Attention weights etc.
- **Action:** Next Word
- **Policy:** Generate the next word (*action*) given the current hidden layers and attention weights (*state*)
- **Reward:** Score computed using the evaluation metric (eg: BLEU)

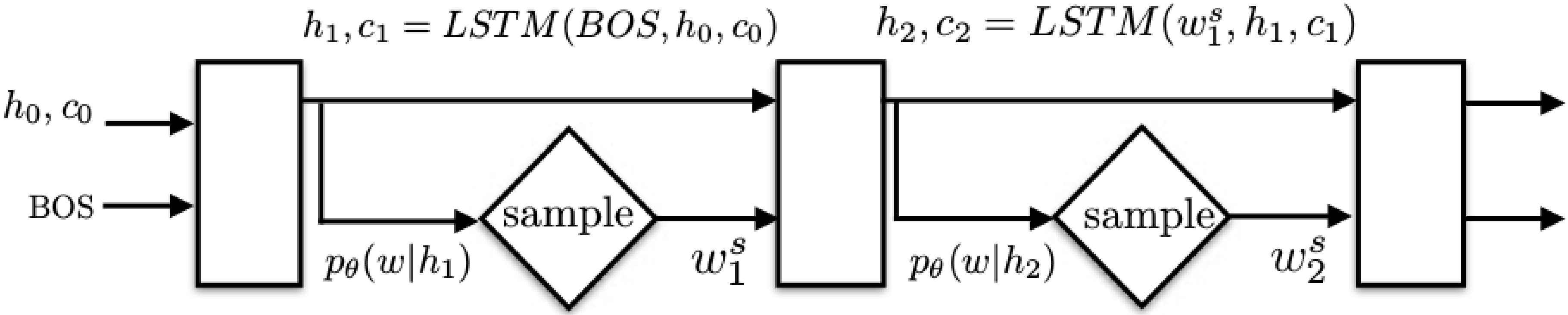
Maximum Likelihood Training (Revisit)



$$\text{Log Likelihood } = \sum_{t=1}^{T} \log p_\theta(w_t^g | h_t)$$

Minimize the negative log likelihood

Reinforcement Learning Formulation



$$\text{Reward} \;= r(w^s) = r(w_1^s, w_2^s, \cdots, w_T^s)$$

Minimize the expected negative reward, $L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}\left[r(w^s)\right]$
using REINFORCE algorithm

Reinforcement Learning Details

- Expected reward $L(\theta) = -\sum_{w} p_\theta(w) r(w)$
- We need the gradient $\nabla_\theta L(\theta) = -\sum_{w} r(w) \nabla_\theta p_\theta(w)$
- Need to write this as an expectation, so that we can evaluate it using samples. Use the log derivative trick:

$$\nabla_\theta L(\theta) = -\sum_{w} r(w) p_\theta(w) \nabla_\theta \log p_\theta(w)$$

- This is an expectation $\nabla_\theta L(\theta) = -\mathbb{E}_{w^s \sim p_\theta} \left[r(w^s) \nabla_\theta \log p_\theta(w^s) \right]$
- Approximate this with sample mean

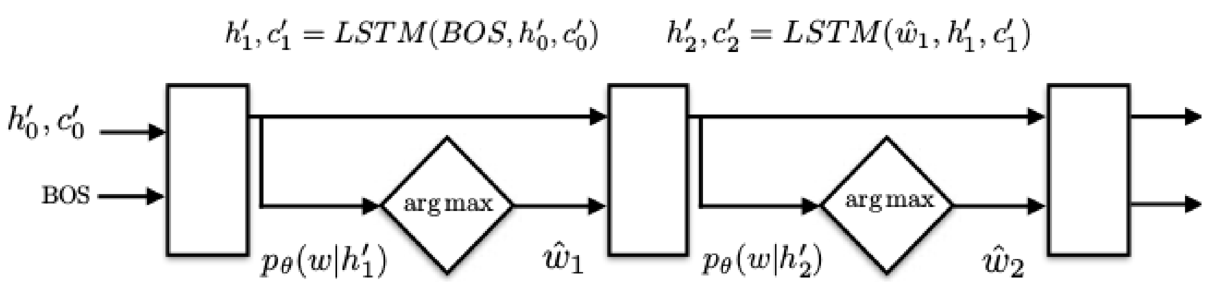
$$\nabla_\theta L(\theta) \approx -\frac{1}{N} \sum_{s=1}^{N} r(w^s) \nabla_\theta \log p_\theta(w^s)$$

- In practice we use only one sample

$$\nabla_\theta L(\theta) \approx -r(w^s) \nabla_\theta \log p_\theta(w^s)$$

Reinforcement Learning Details

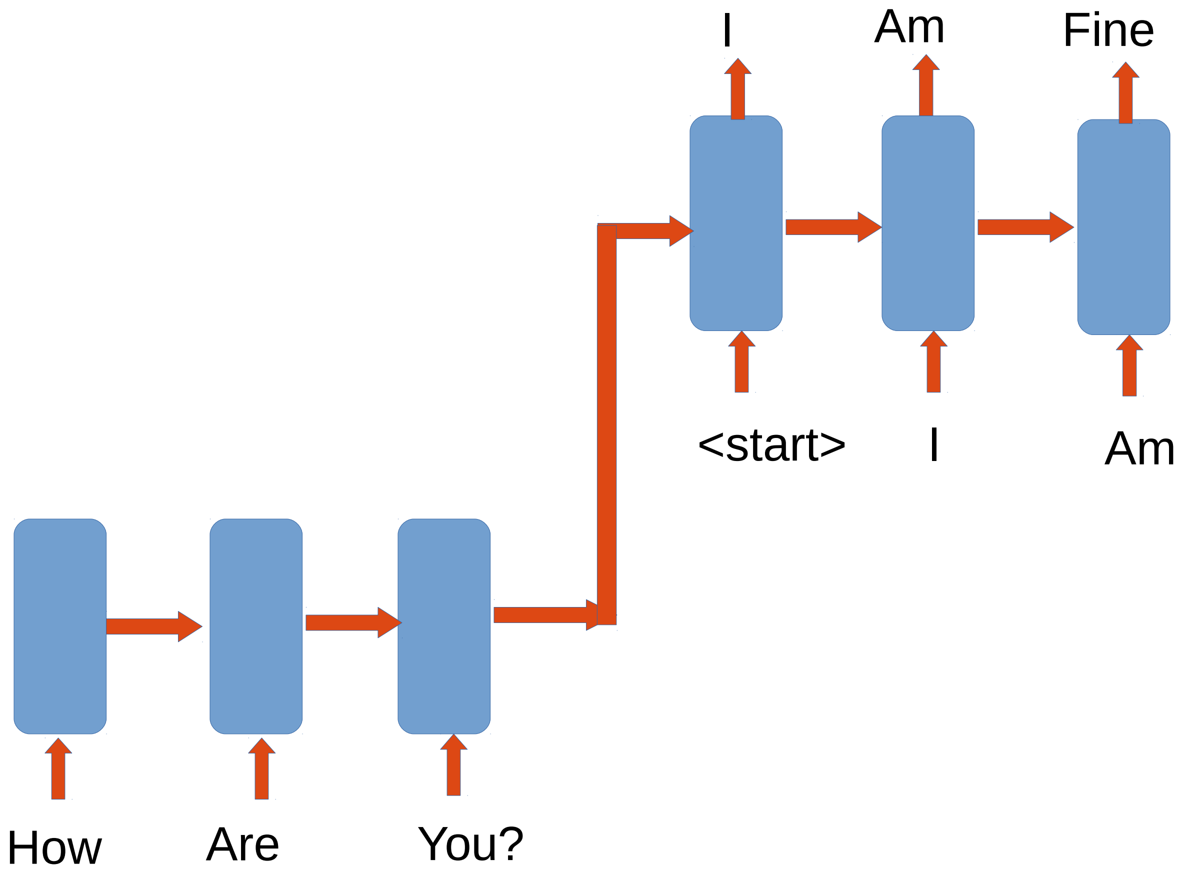
- Gradient $$\nabla_\theta L(\theta) \approx -r(w^s)\nabla_\theta \log p_\theta(w^s)$$
- This estimation has high variance. Use a baseline to combat this problem.
$$\nabla_\theta L(\theta) \approx -(r(w^s) - b)\nabla_\theta \log p_\theta(w^s)$$
- Baseline can be anything independent of w^s
- It can for example be estimated as the reward for word sequence generated using argmax at each cell.
$$b = r(\hat{w}_1, \hat{w}_2, \hat{w}_3, \cdots, \hat{w}_T)$$



Reinforcement Learning

- Machine Translation/Summarization
- Dialog Systems
-
-

Maximum Likelihood Dialog Systems



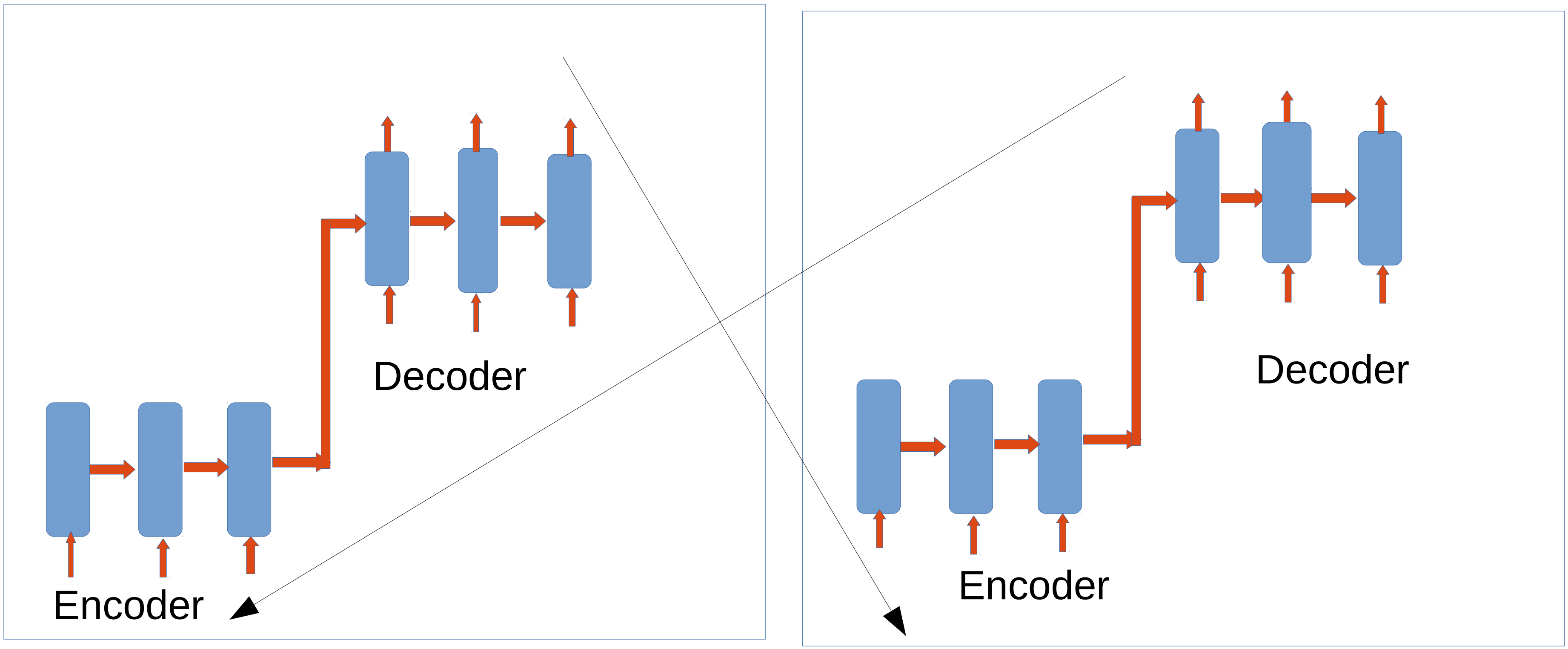
Why Reinforcement Learning

- Maximum Likelihood criterion is not directly relevant to successful dialogs
  - Dull responses (“I don’t know”)
  - Repetitive responses
- Need to integrate developer defined rewards relevant to longer term goals of the dialog

Dialog Generation as Reinforcement Learning

- **Agent:** The Recurrent Net
- **State:** Previous dialog turns
- **Action:** Next dialog utterance
- **Policy:** Generate the next dialog utterance (*action*) given the previous dialog turns (*state*)
- **Reward:** Score computed based on relevant factors such as ease of answering, information flow, semantic coherence etc.

Training Setup



Agent 1

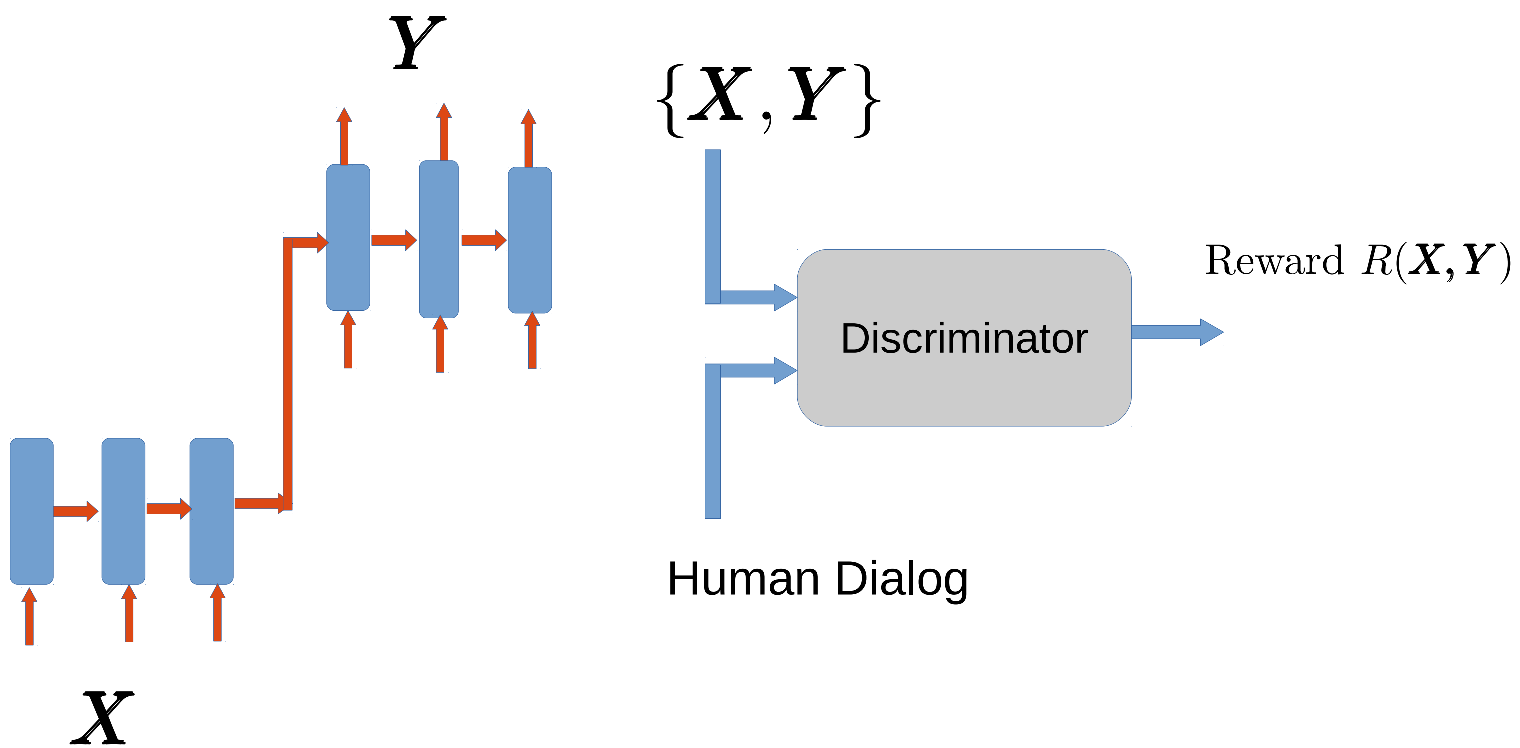
Agent 2

Training Procedure

- From the viewpoint of a given agent, the procedure is similar to that of sequence generation
  - REINFORCE algorithm
- Appropriate rewards must be calculated based on current and previous dialog turns.
- Can be initialized with maximum likelihood trained models.

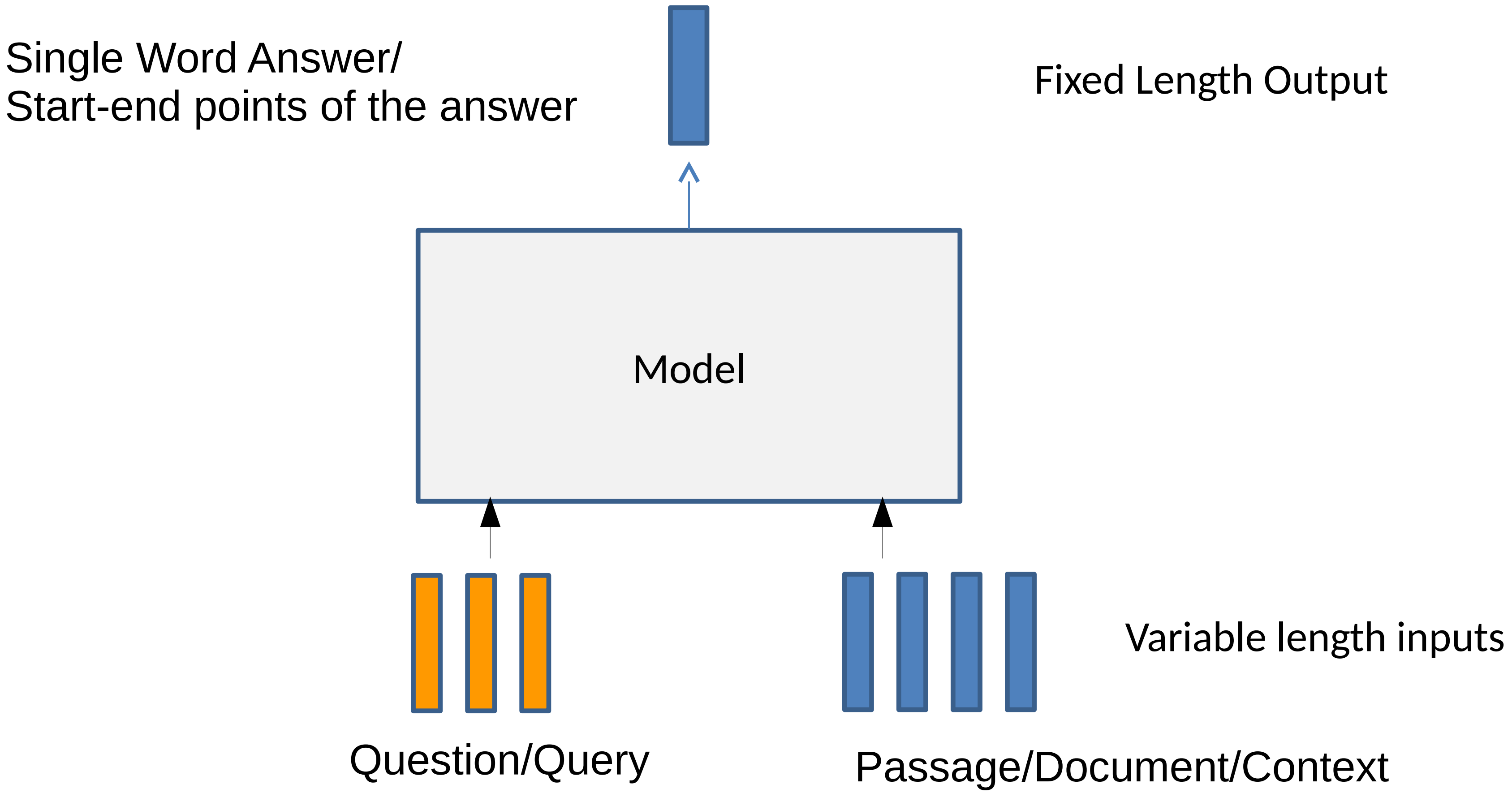
Adversarial Learning

- Use a discriminator as in GANs to calculate the reward
- Same training procedure based on REINFORCE for generator



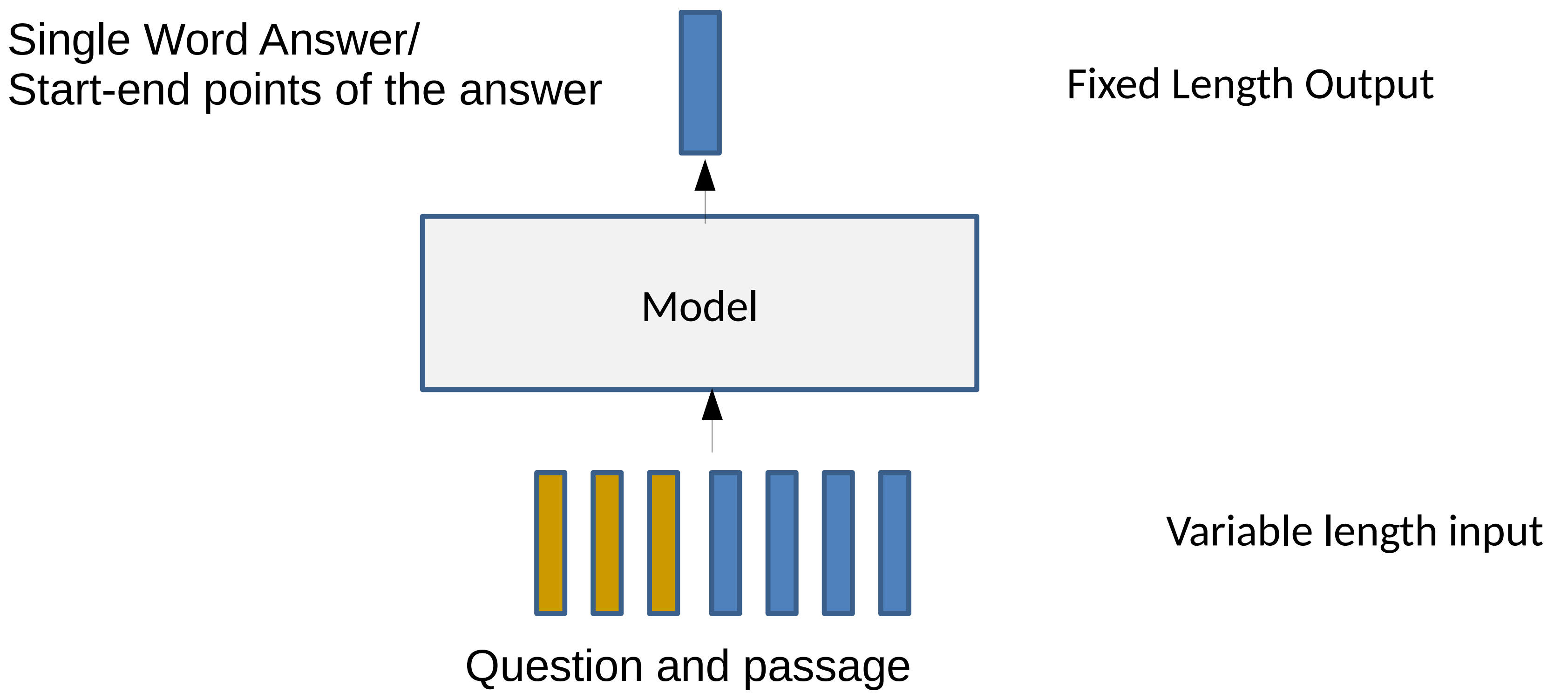
Question Answering

- Slightly different from sequence-to-sequence model.



QA- Naive Approach

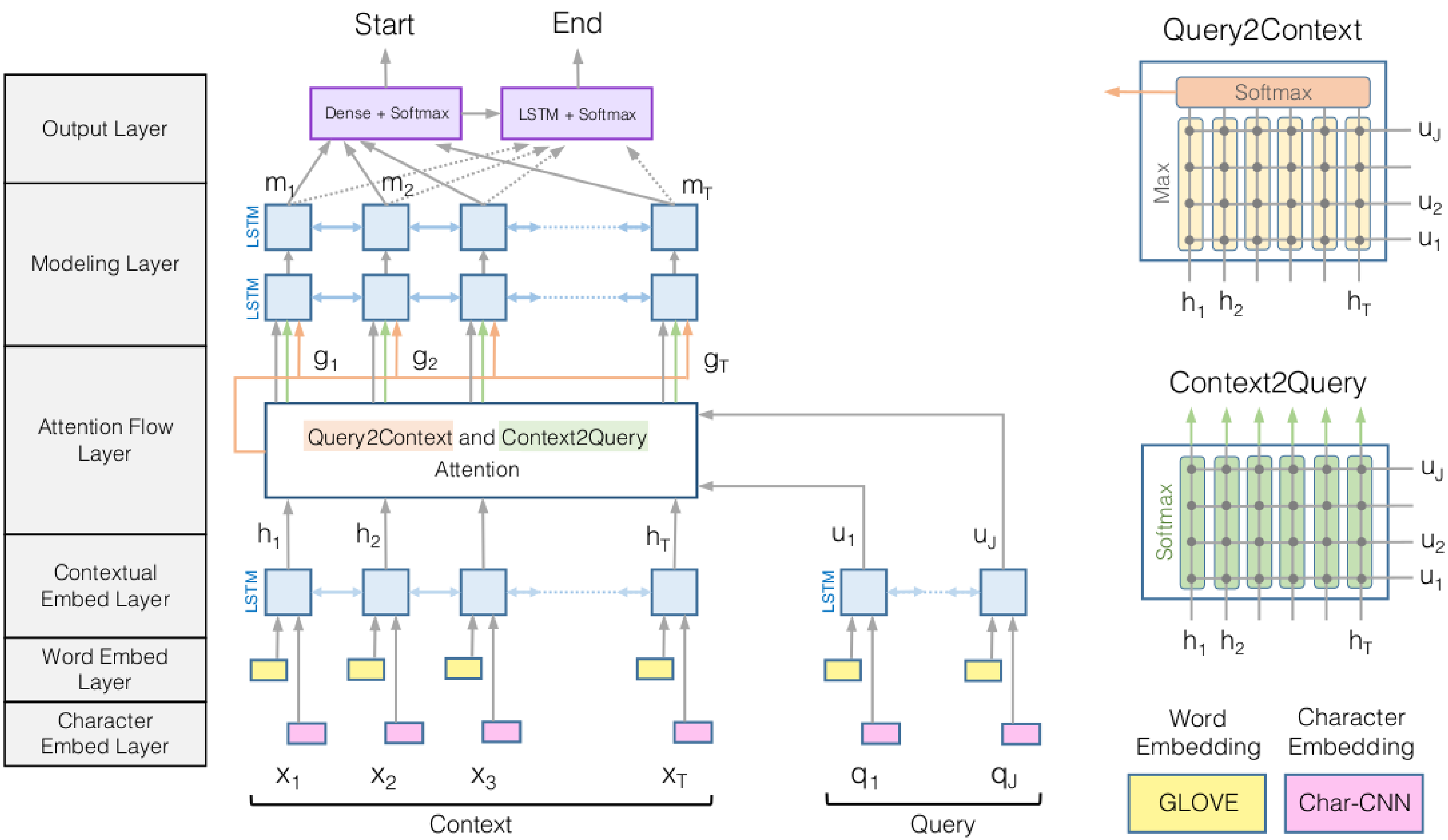
- Combine question and passage and use an RNN to classify it.
- Will not work because relationship between the passage and question is not adequately captured.



QA- More Successful Approach

- Use attention between the question and passage
  - Bi-directional attention, co-attention
- Temporal relationship modeling
- Classification or predict start and end-point of the answer within passage.

QA Example with Bi-directional Attention



Bi-directional attention flow for machine comprehension Seo M. et.al