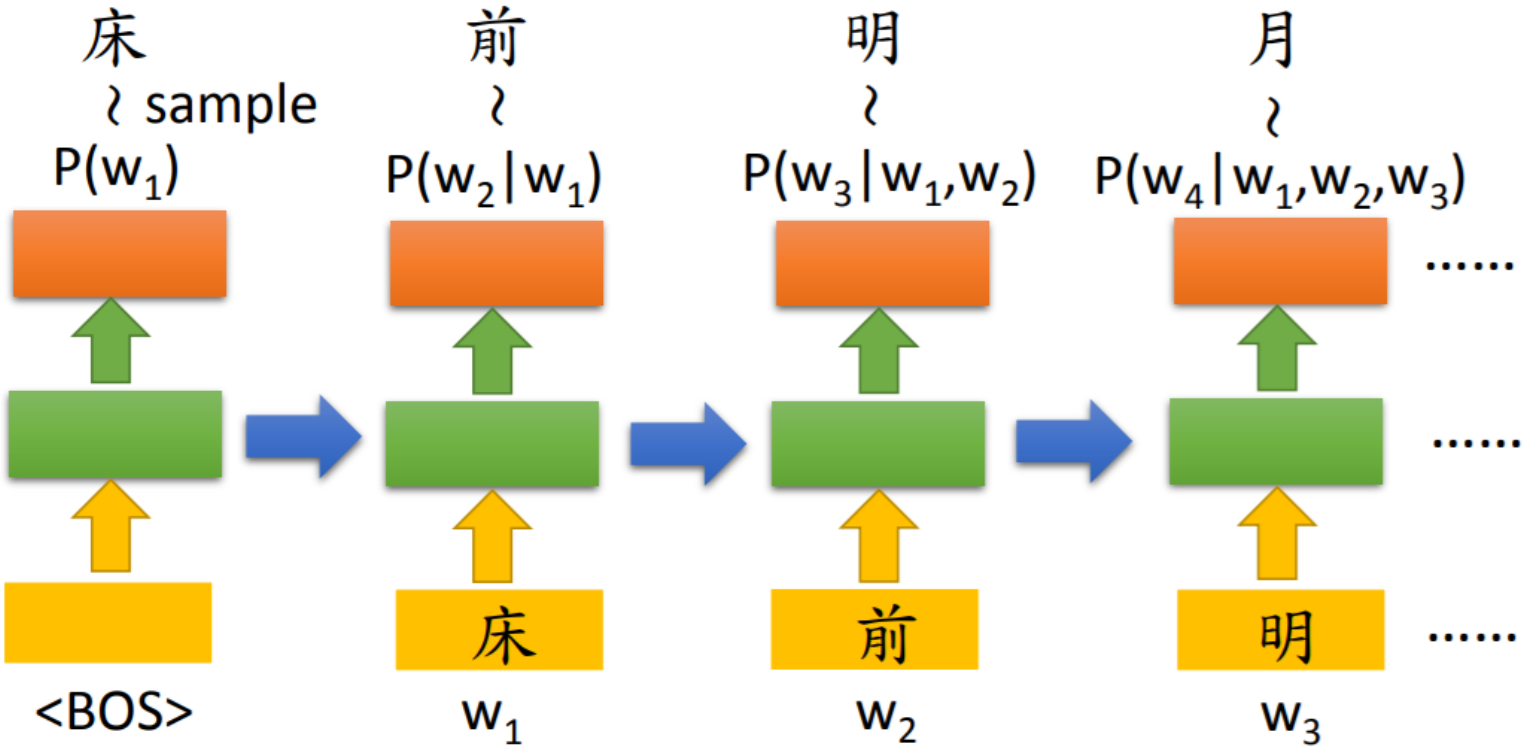
注意力机制

许瑞晗

2020年12月25日

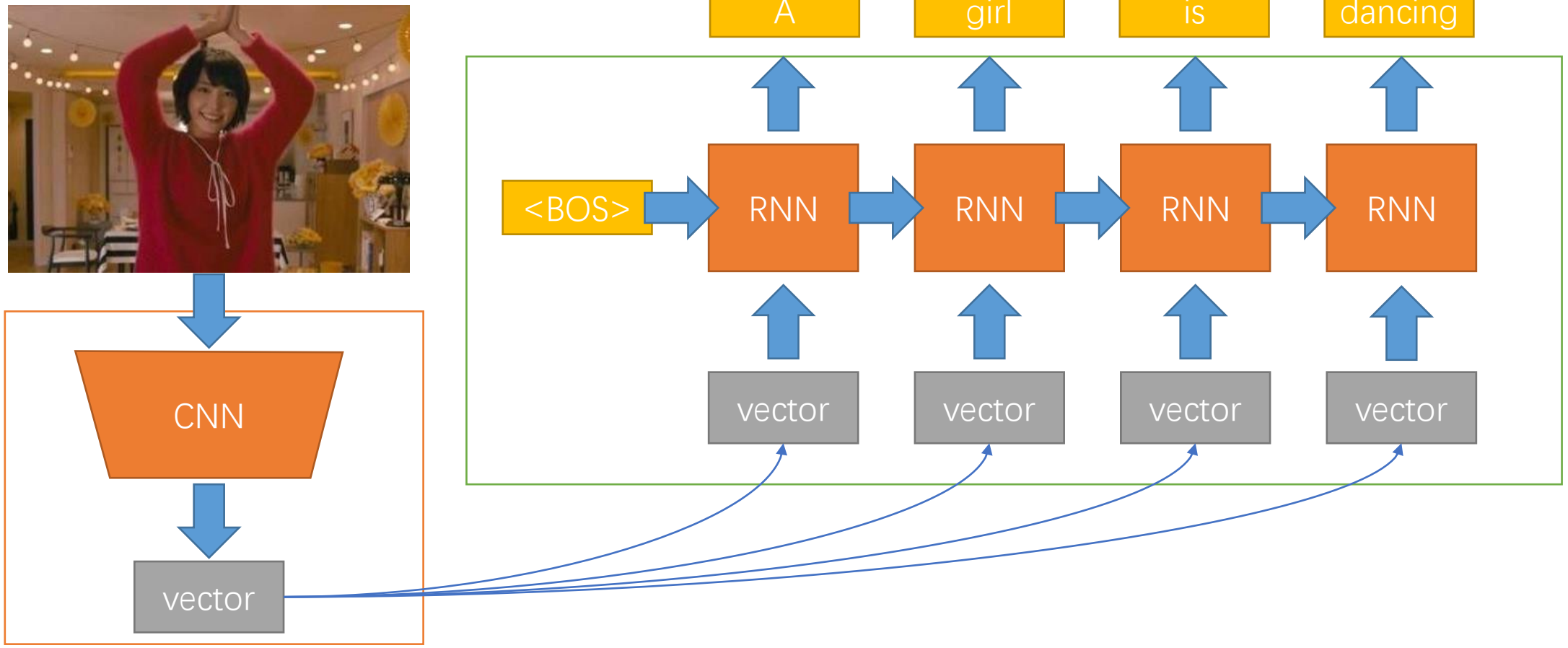
Generation

- Generating a structured object component-by-component
- 不是 generative model



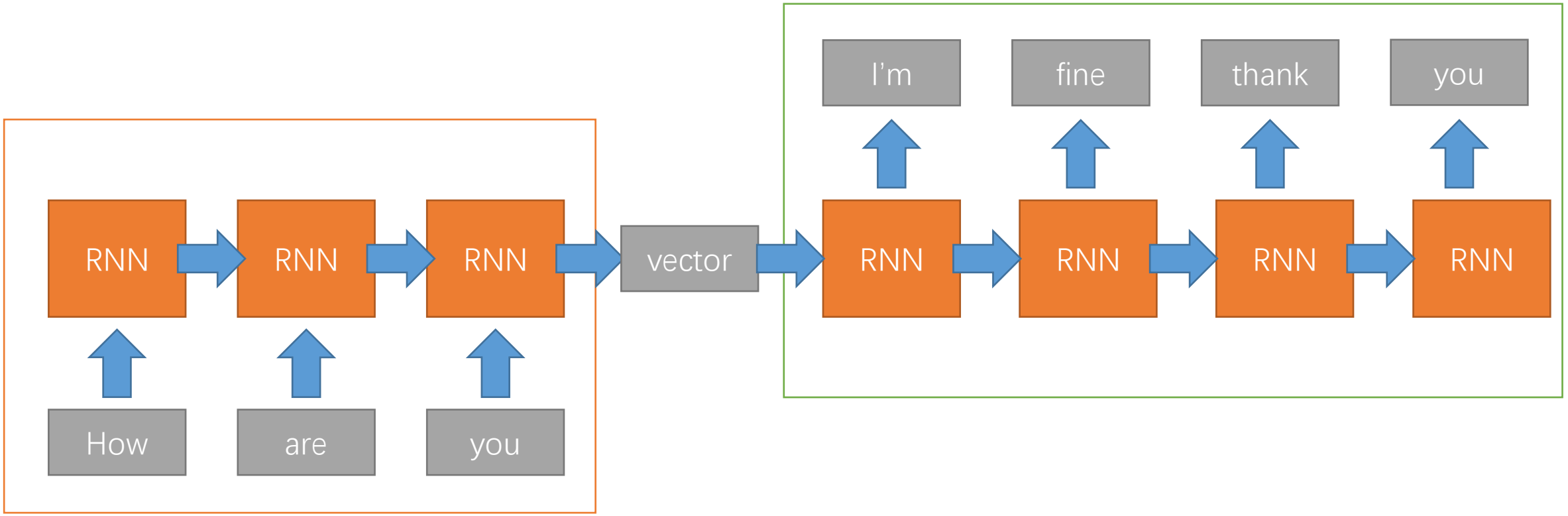
Conditional Generation

- Image Captioning



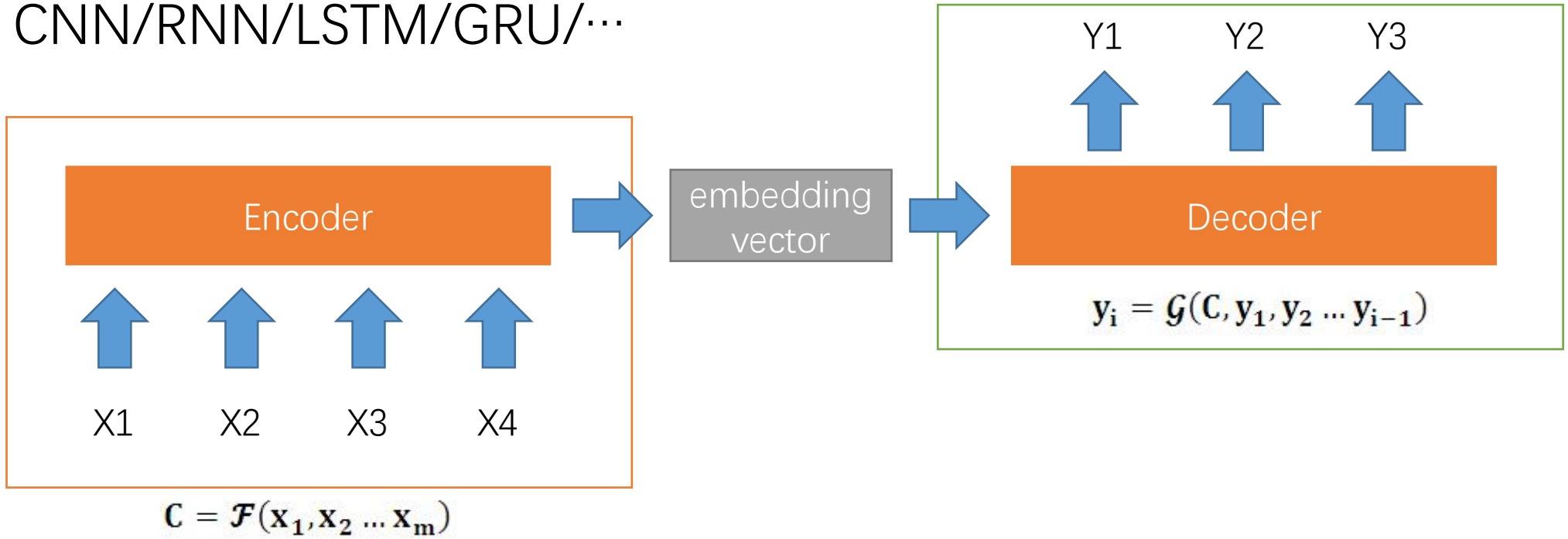
Conditional Generation

- Chat-Bot/Machine Translation



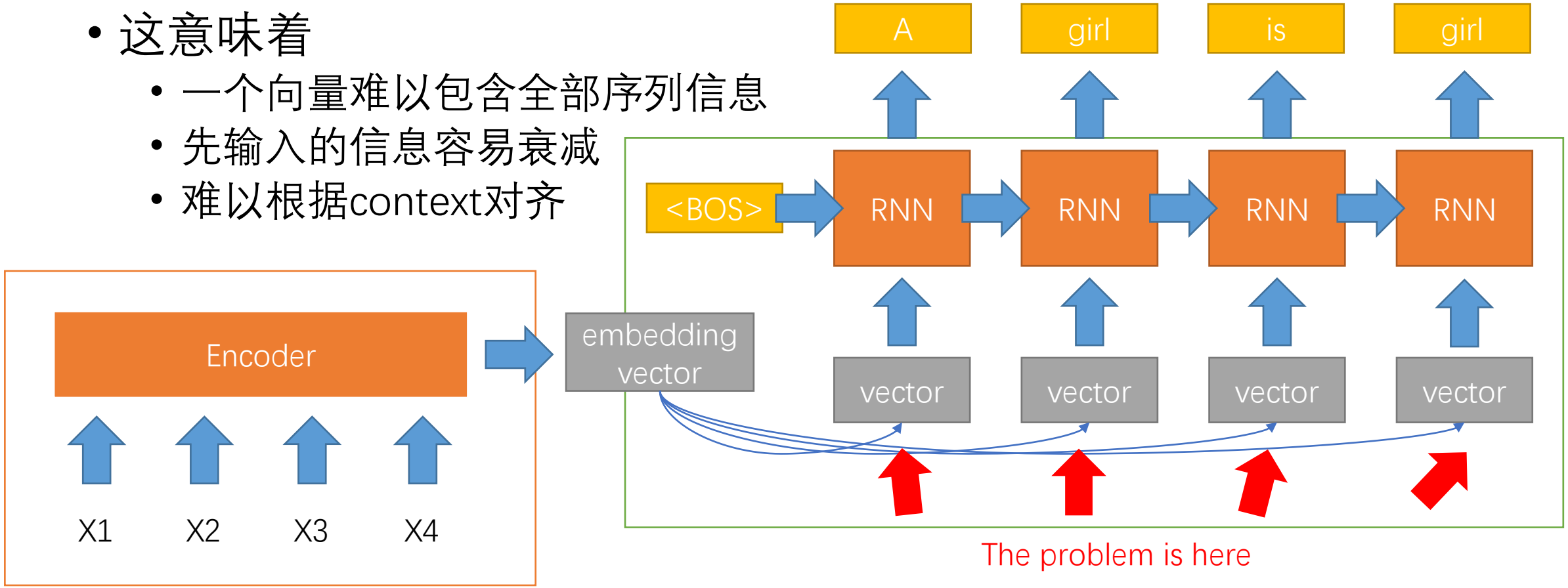
Encoder-Decoder

- 解决seq2seq问题的一种方法 $\mathbf{X} = \langle \mathbf{x_1}, \mathbf{x_2} \ldots \mathbf{x_m} \rangle$ $\mathbf{Y} = \langle \mathbf{y_1}, \mathbf{y_2} \ldots \mathbf{y_n} \rangle$
- 定长输入→ 定长输出，多余的用空字符补齐
- Encoder/Decoder can be any model, including CNN/RNN/LSTM/GRU/…



The problem of RNN in Encoder-decoder

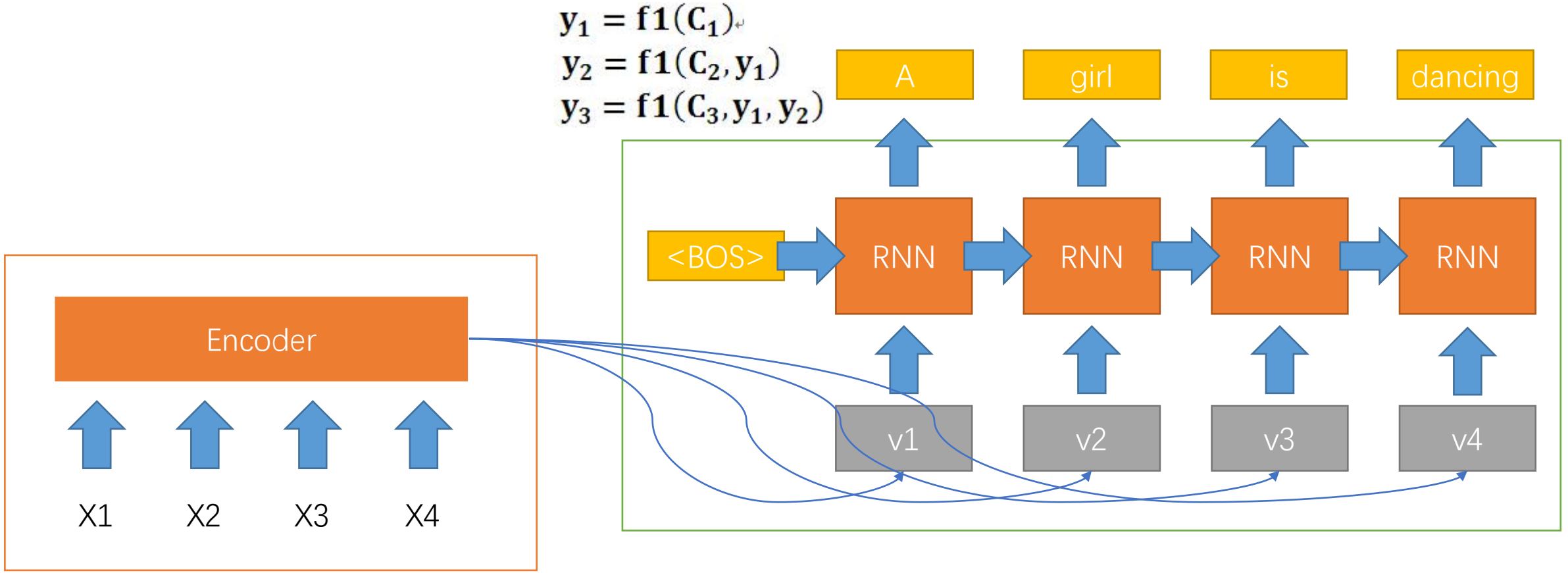
- Decoder中的输入的每个embedding vector都是相同的
- 这意味着
  - 一个向量难以包含全部序列信息
  - 先输入的信息容易衰减
  - 难以根据context对齐



Attention Model

- 增加了一个注意力范围，强调接下来输出内容应该关注哪一部分

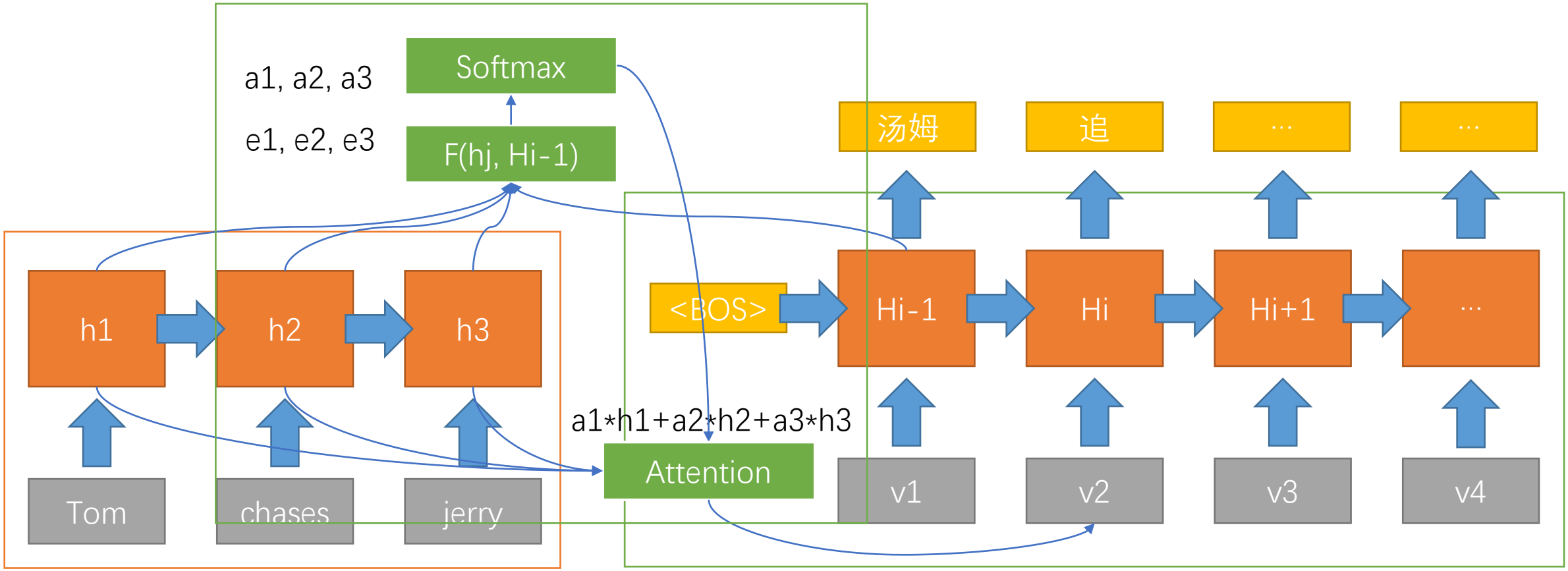
$$y_1 = f1(C_1)$$
$$y_2 = f1(C_2, y_1)$$
$$y_3 = f1(C_3, y_1, y_2)$$



How to do Attention?

Bahdanau D , Cho K , Bengio Y . Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer ence, 2014.

- 以机器翻译为例

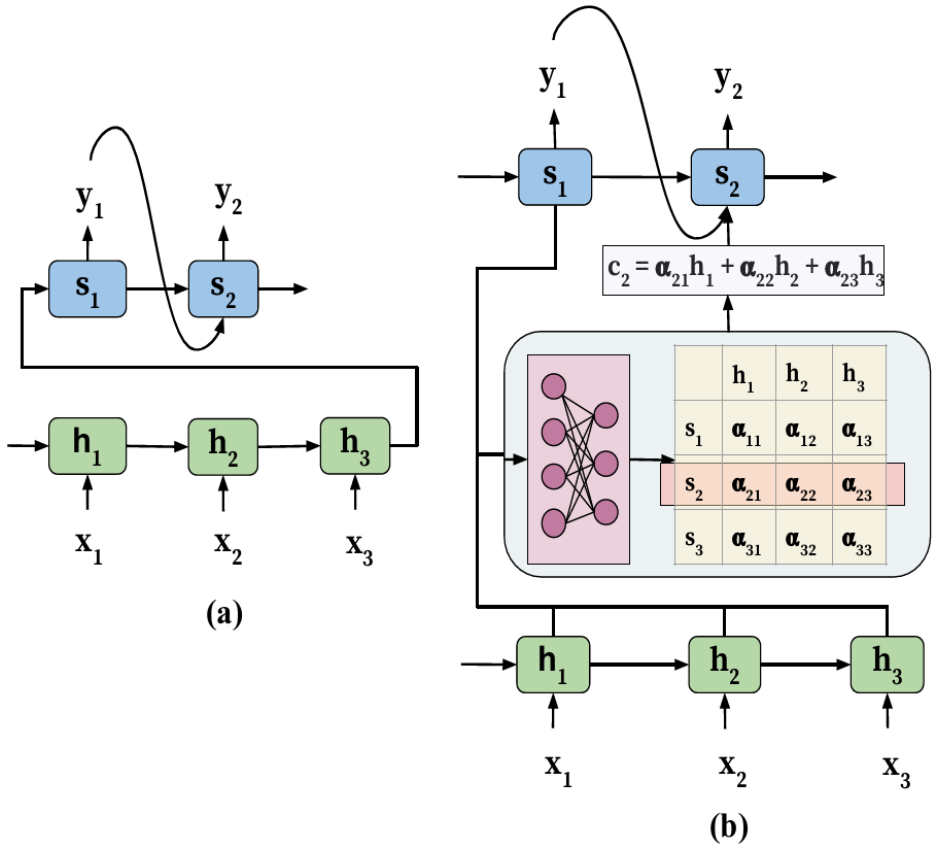


Comparison

Function	Traditional Encoder-Decoder	Encoder-Decoder with Attention
Encode	$h_i = f(x_i, h_{i-1})$	$h_i = f(x_i, h_{i-1})$
Context	$c = h_T$	$c_j = \sum_{i=1}^{T} \alpha_{ij} h_i$ $\alpha_{ij} = p(e_{ij})$ $e_{ij} = a(s_{j-1}, h_i)$
Decode	$s_j = f(s_{j-1}, y_{j-1}, c)$	$s_j = f(s_{j-1}, y_{j-1}, c_j)$
Generate	$y_j = g(y_{j-1}, s_j, c)$	$y_j = g(y_{j-1}, s_j, c_j)$

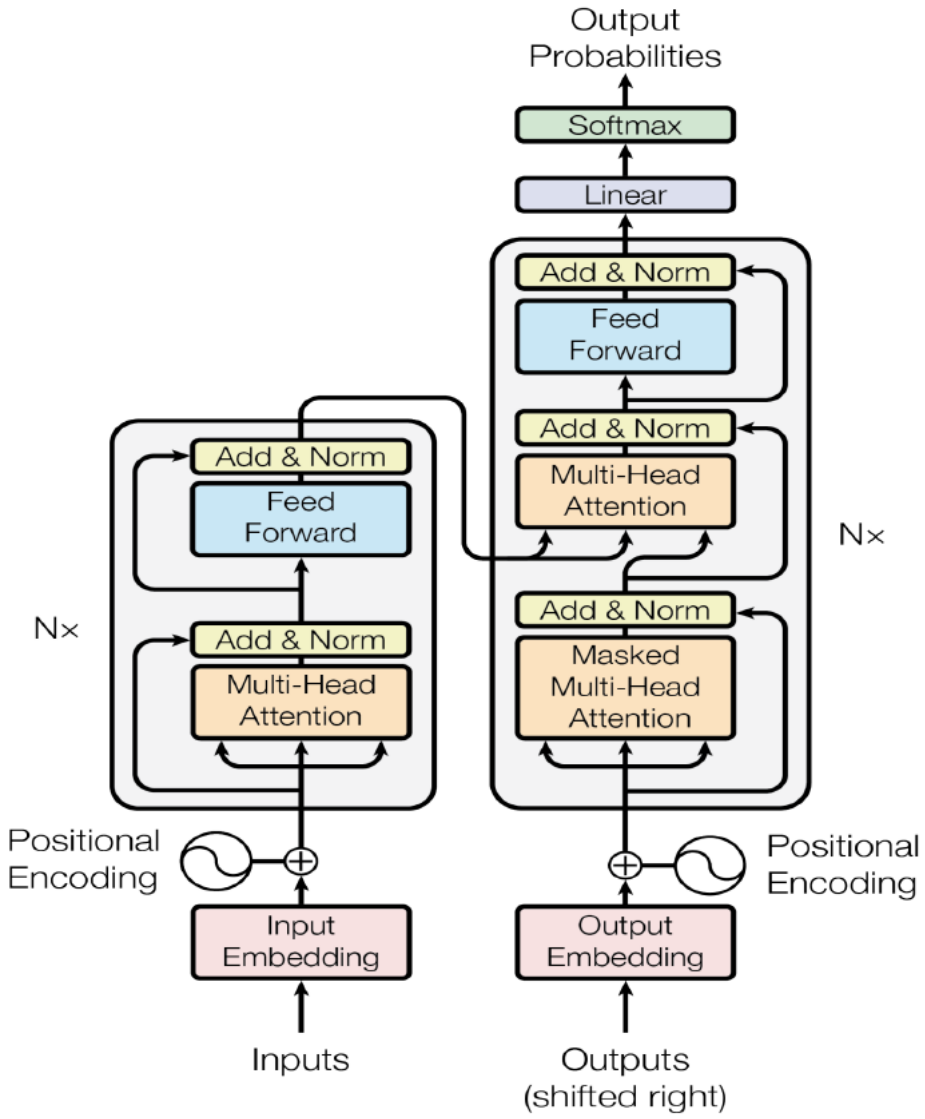
$x = (x_1, ..., x_T)$: input sequence, T: length of input sequence, h_i: hidden states of encoder, c: context vector, α_{ij}: attention weights over input, s_j: decoder hidden state, y_j: output token, f, g: non-linear functions, a: alignment function, p: distribution function

Chaudhari S , Polatkan G , Ramanath R , et al. An Attentive Survey of Attention Models[J]. 2019.



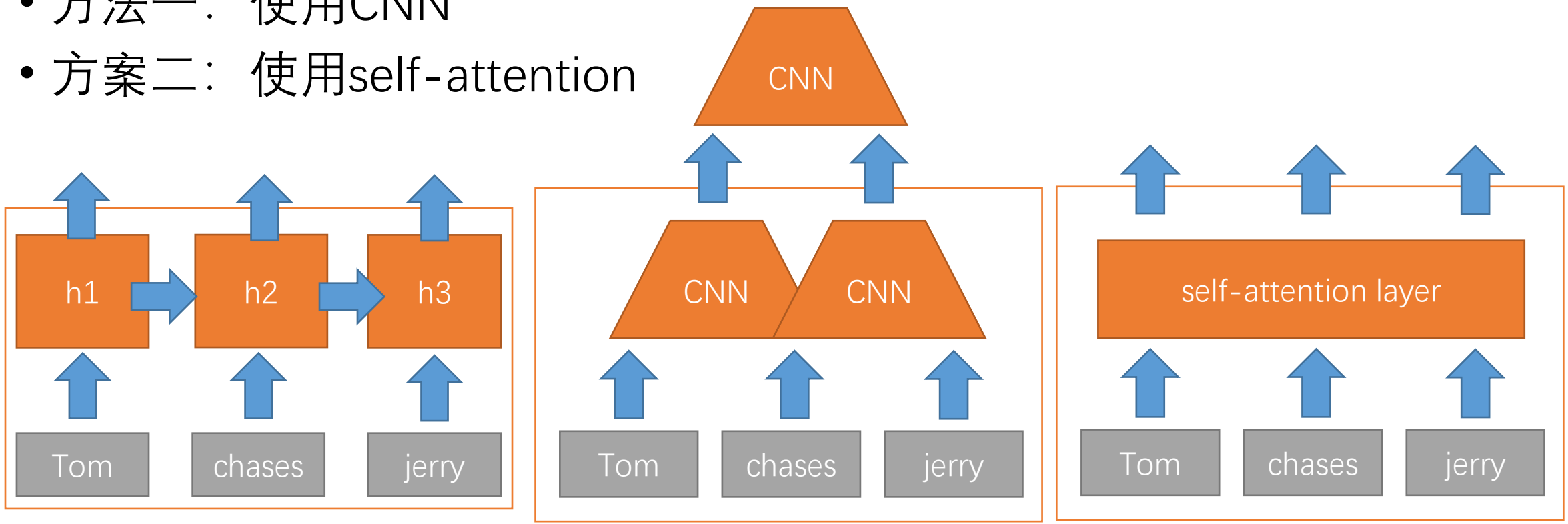
Transformer

- 只用Attention，抛弃RNN(CNN)
- 在各种任务上都取得了on par的结果



Transformer

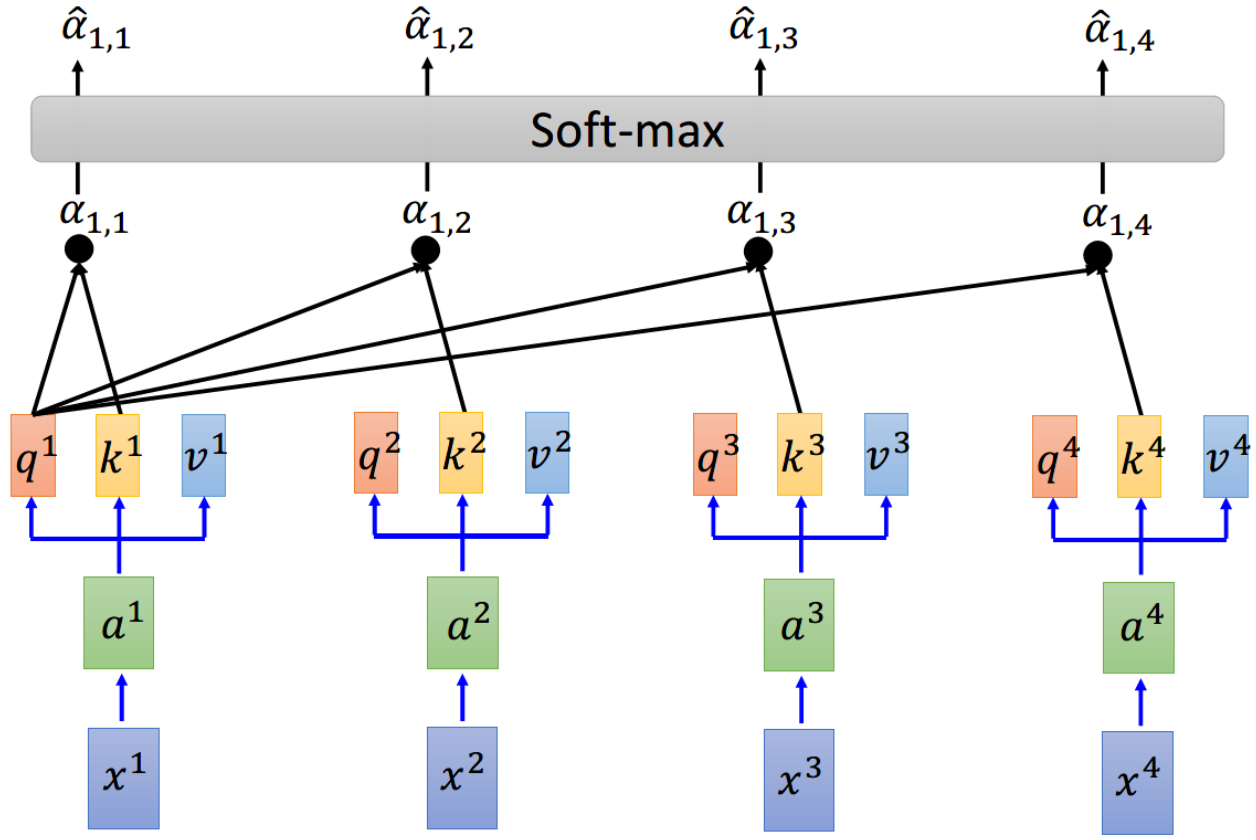
- RNN难以并行
- 方法一：使用CNN
- 方案二：使用self-attention



Self-attention

- q – query
- k – key
- v – value

- x是输入，a是x的embedding
- 对a做线性变换得到q,k,v
- 用q和k计算相似度，得到attention系数



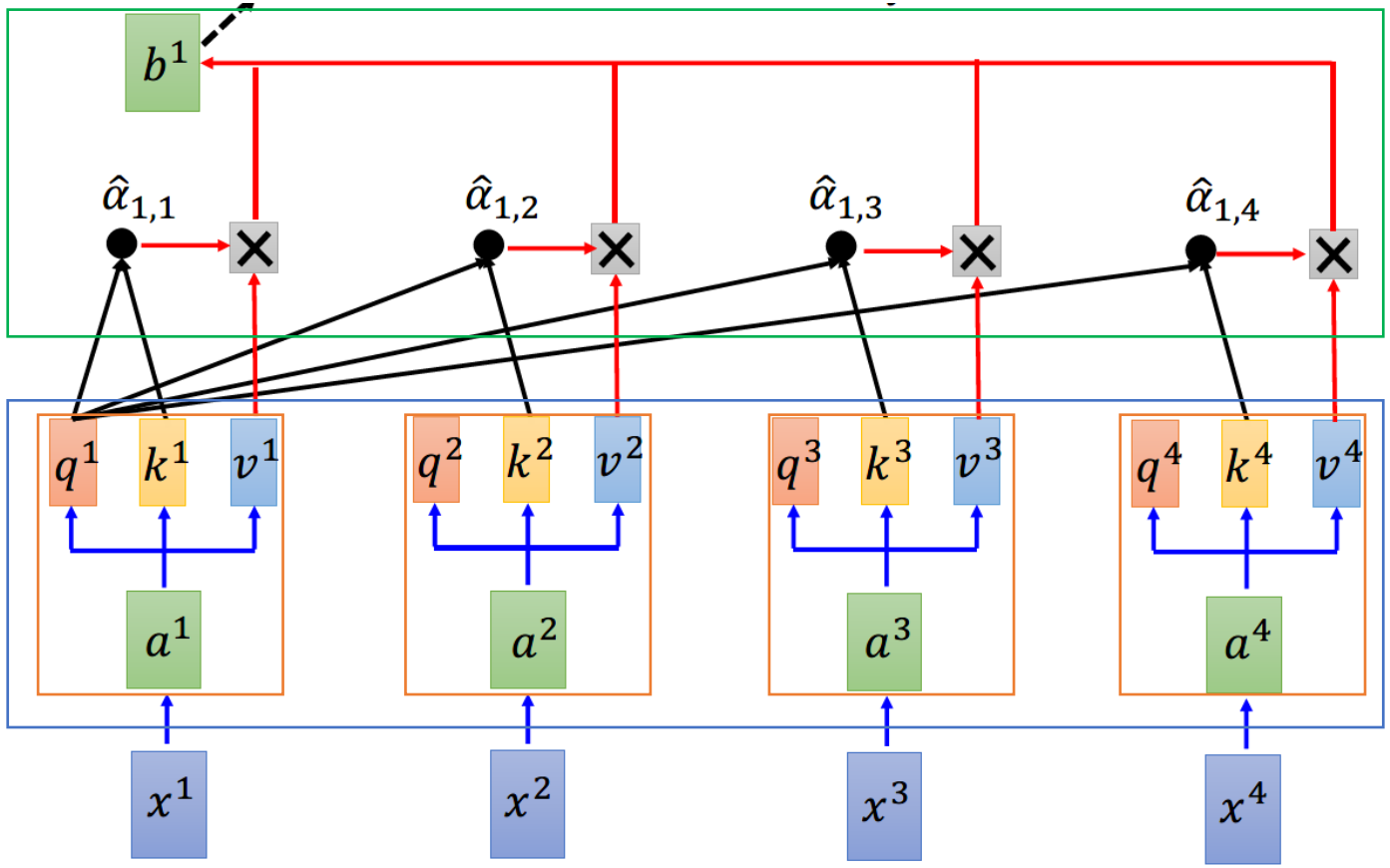
Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

Self-attention

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

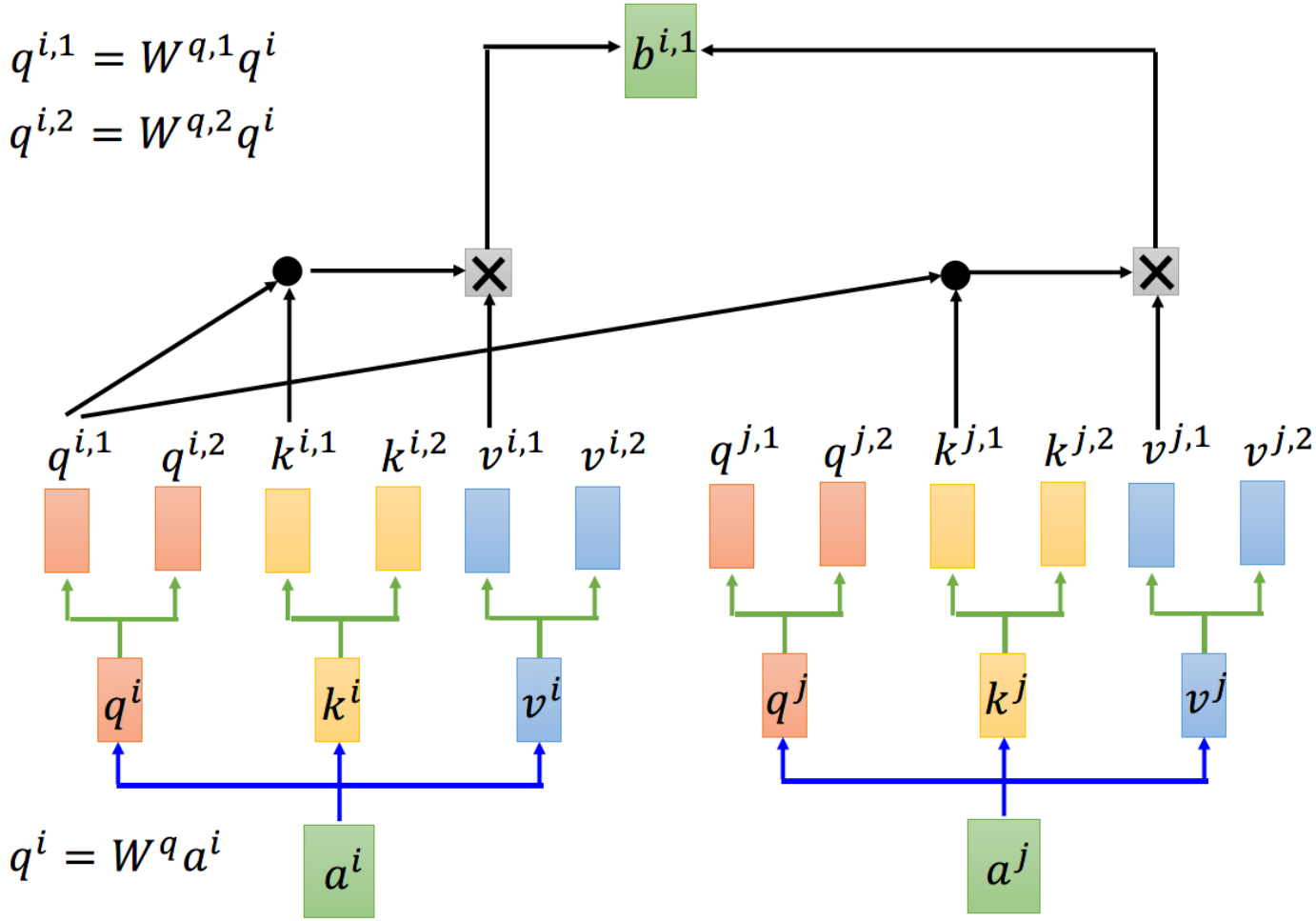
- 用q1得到的系数计算第1个位置的输出
- 其余同理

- 并行效率：
  - 每一个q, k, v都可以并行计算
  - 每一个输出b都可以并行计算
  - 全部为矩阵乘法，可用GPU



Multi-head self-attention

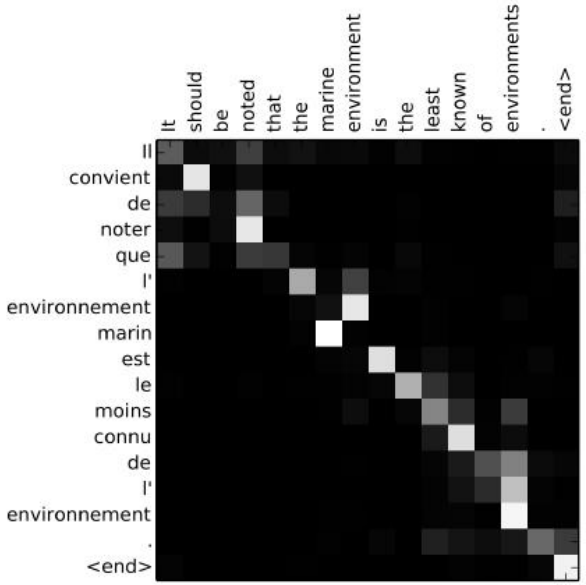
- 使用两个head，其余类似



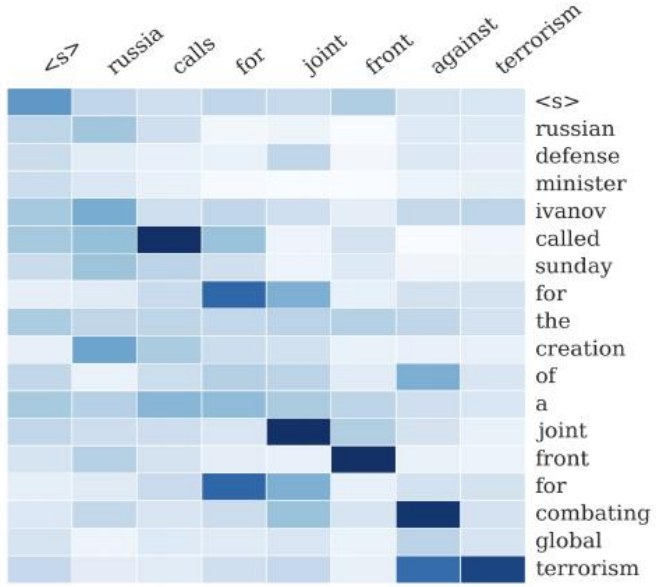
Different Attention Models

Chaudhari S , Polatkan G , Ramanath R , et al. An Attentive Survey of Attention Models[J]. 2019.

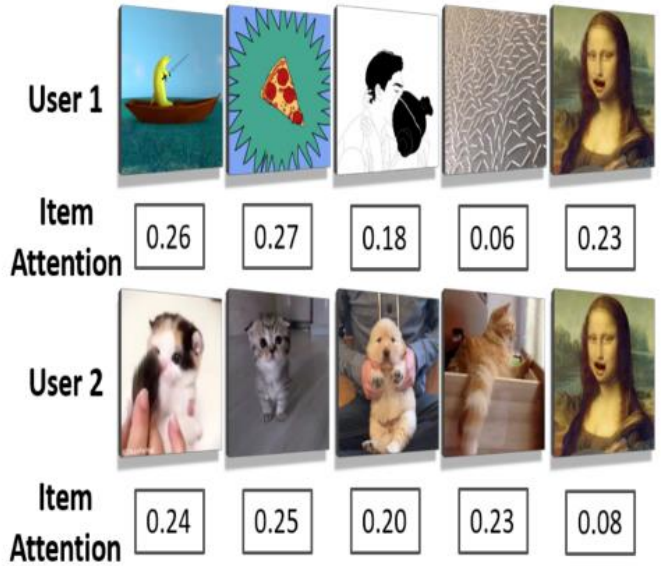
- 注意力机制增强了神经网络的可解释性



(a) Alignment of French and English sentences in MT [Bahdanau et al. 2015]



(b) Alignment of input and output sequences for summarization [Rush et al. 2015]



(c) Weights of items in user's history for recommendation [He et al. 2018]

Attention can also be used in captioning

- Xu K , Ba J , Kiros R , et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention[J]. Computer Science, 2015:2048-2057.

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1})$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{L} \exp(e_{tk})}.$$

$$\hat{\mathbf{z}}_t = \phi\left(\{\mathbf{a}_i\}, \{\alpha_i\}\right),$$

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Attention can also be used in captioning

- Xu K , Ba J , Kiros R , et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention[J]. Computer Science, 2015:2048-2057.

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Attention 代码实现

```
# lstm_output : [batch_size, n_step, n_hidden * num_directions(=2)], F matrix
def attention_net(self, lstm_output, final_state):
    # hidden : [batch_size, n_hidden * num_directions(=2), 1(=n_layer)]
    hidden = final_state.view(-1, self.n_hidden * 2, 1)
    # attn_weights : [batch_size, n_step]
    attn_weights = torch.bmm(lstm_output, hidden).squeeze(2)
    soft_attn_weights = F.softmax(attn_weights, 1)

    # [batch_size, n_hidden * num_directions(=2), n_step] * [batch_size, n_step, 1]
    # = [batch_size, n_hidden * num_directions(=2), 1]
    context = torch.bmm(lstm_output.transpose(1, 2),
                        soft_attn_weights.unsqueeze(2)).squeeze(2)
    # context : [batch_size, n_hidden * num_directions(=2)]
    return context, soft_attn_weights.data.numpy()
```