法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

神经序列模型 IV

主讲人： 史兴

07/14/2017

提纲

- Seq2Seq Model
- Beam Search
- Attention

Seq2Seq Model

- 核心：
  - $p(E|F) = p(e_1^N \middle| f_1^M)$
    $$= \prod_{i=1}^{N} p(e_i|e_1^{i-1}, f_1^M)$$
    - F: input sequence/source sequence
    - E: output sequence/target sequence

Seq2Seq Model

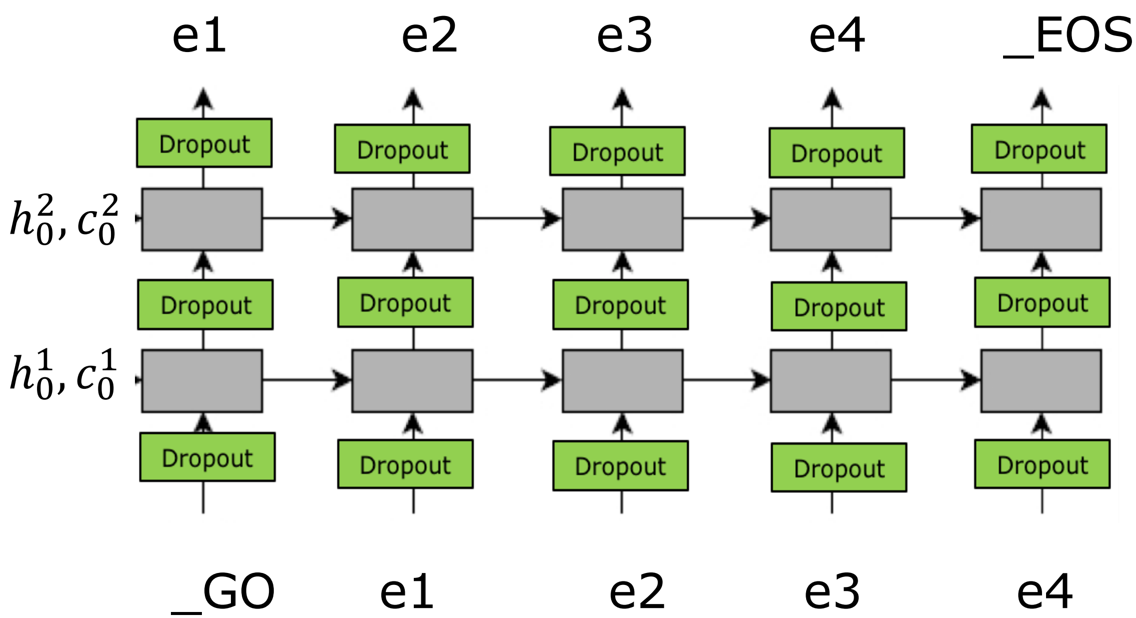
- 自然语言处理很多都可以转化成seq2seq
  - 核心：$p(e_1^N|f_1^M) = \prod_{i=1}^N p(e_i|e_1^{i-1}, f_1^M)$

问题	E	F
语言模型	句子	null
机器翻译	中文	外文
句子分类	类别	句子
诗歌生成	诗句	主题词
对话机器人	回复	话语
问答系统	答案	问题

Seq2Seq Model

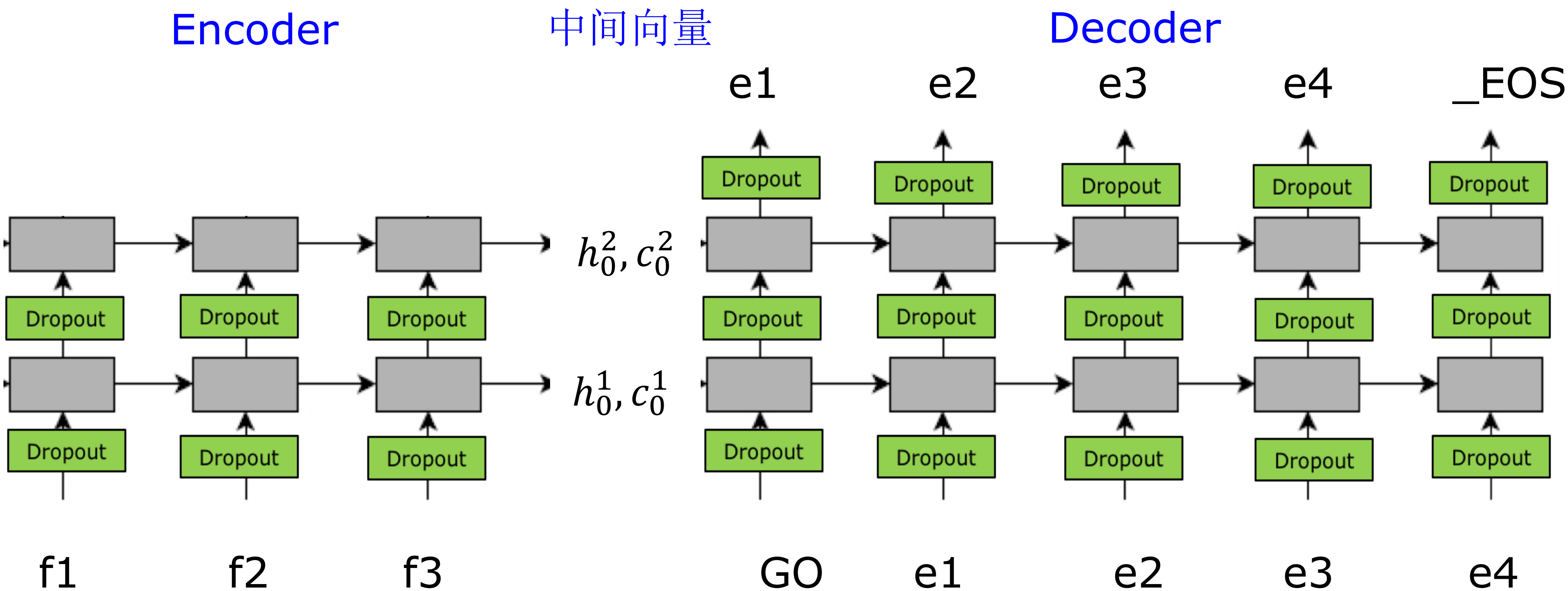
- seq2seq的结构
  - 核心：$p(e_1^N|f_1^M) = \prod_{i=1}^{N} p(e_i|e_1^{i-1}, f_1^M)$

全零向量



Seq2Seq Model

- seq2seq的结构
  - 核心： $p(e_1^N|f_1^M) = \prod_{i=1}^{N} p(e_i|e_1^{i-1}, f_1^M)$
  - h_0, c_0 由另外一个LSTM来计算

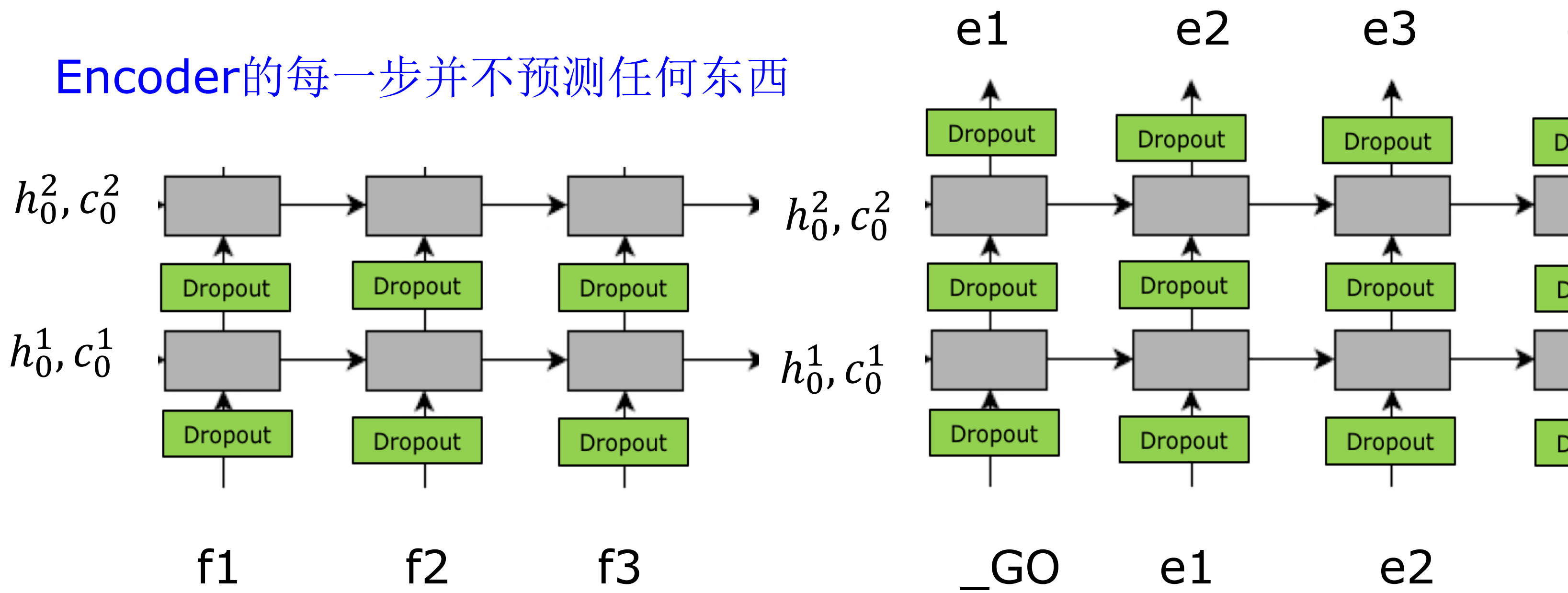


Seq2Seq Model

- 自然语言处理很多都可以转化成seq2seq
  - 核心： $p(e_1^N | f_1^M) = \prod_{i=1}^{N} p(e_i | e_1^{i-1}, f_1^M)$
  - Encoder和Decoder是完全不同的参数

Encoder的每一步并不预测任何东西

全零向量



Seq2Seq Model

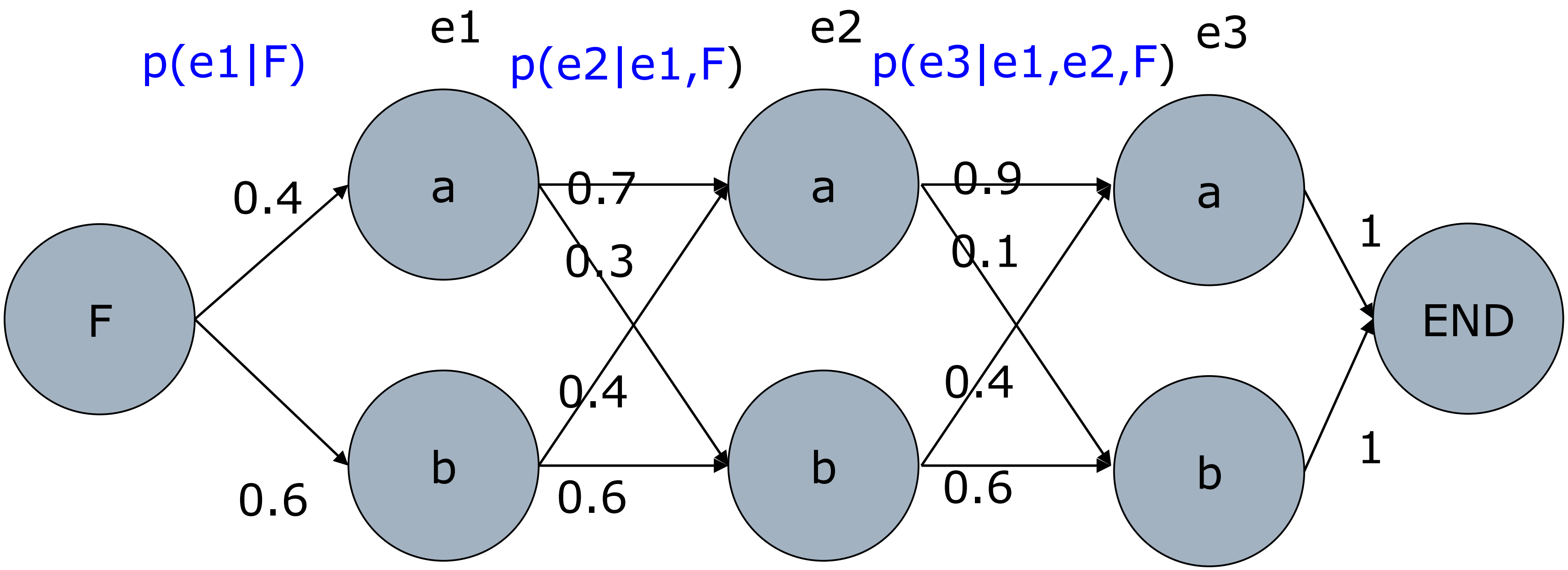
- Seq2Seq的参数集合
  - 超参数
    - 层数=n, hidden size=d
    - vocab for F = V_F, vocab for E = V_E
  - Encoder:
    - Input: input embedding for f : $|V_F| * d$
    - LSTM: 第一层，第二层: $\mathrm{n}(8d^2+4d)$
  - Decoder:
    - Input: input embedding for e: $|V_E| * d$
    - LSTM: 第一层，第二层: $\mathrm{n}(8d^2+4d)$
    - Output:
      - output embedding for e: $|V_E| * d$
      - output bias for e: $|V_E|$

Beam Search

- Language Model：
  - 输入： 句子
  - 输出： P(句子)
- Seq2Seq Model:
  - 输入： source sequence
  - 输出： target sequence (结构化预测)
  - $e_1^N = argmax_{e_1^N} [\log p(e_1^N | f_1^M)]$
    - 搜索空间: $|V_E|^N$

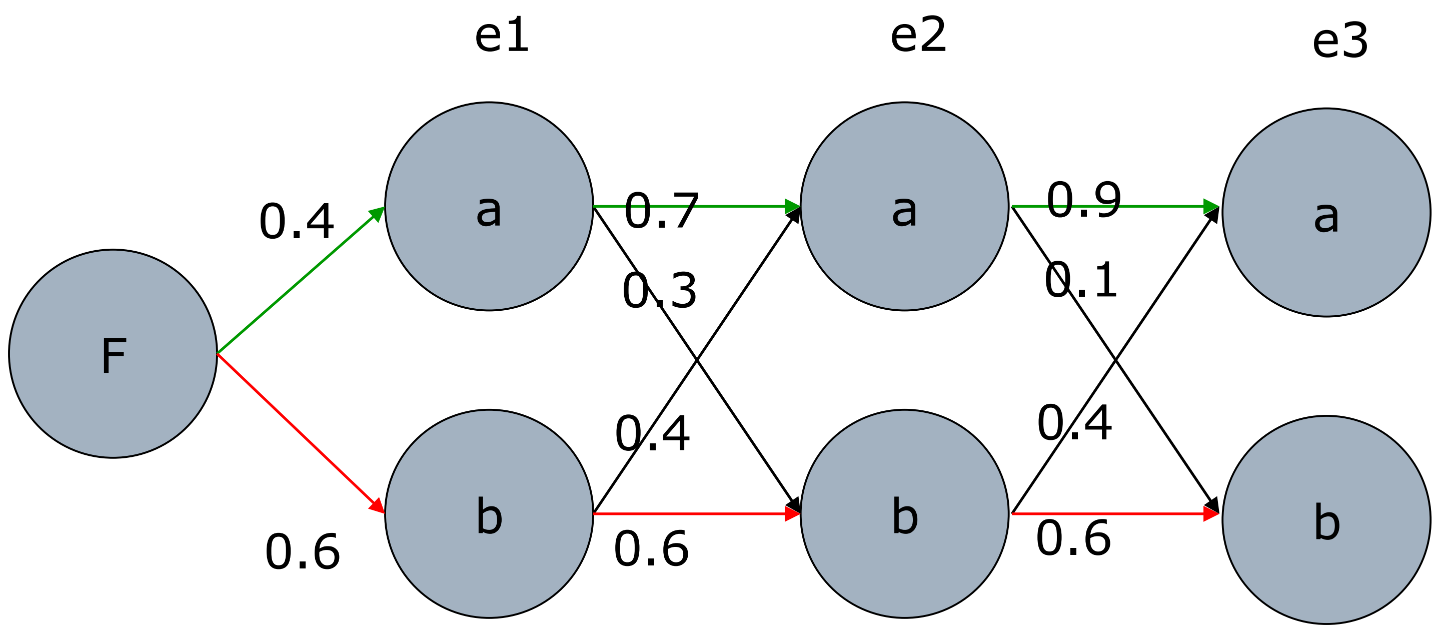
Beam Search

- $V_E = \{a, b\}$
- $argmax_{e_1^3} p(e_1, e_2, e_3|F)$?



Beam Search

- $argmax_{e_1^3} p(e_1, e_2, e_3|F)$?
- 贪心算法：每步都选最大的概率
- p(b,b,b) = 0.6 * 0.6 * 0.6 = 0.216



Beam Search

Viterbi Algorithm

$s(v,n)$ 以v结尾的最大概率的sequence的概率

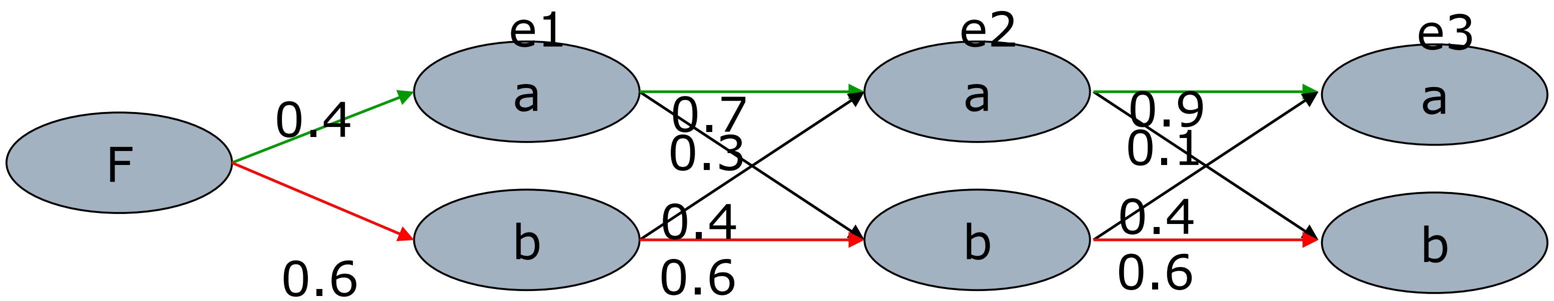
$t(v_1,v_2,n)$ 第n-1步从v_1到第n步的v_2的概率

动态规划递推公式

$$max_{e_1^3} p(e_1,e_2,e_3|F) = \max\{\mathrm{s(v,3)|v} = 1,\ldots,\mathrm{V}\}$$

$$s(v=j,n) = \max\{s(v=i,n-1) * t(i,j,n)|\ i=1,\ldots,V\}$$

$s(v,n)$	n = 1	n = 2	n = 3
v = 1			
v = 2			

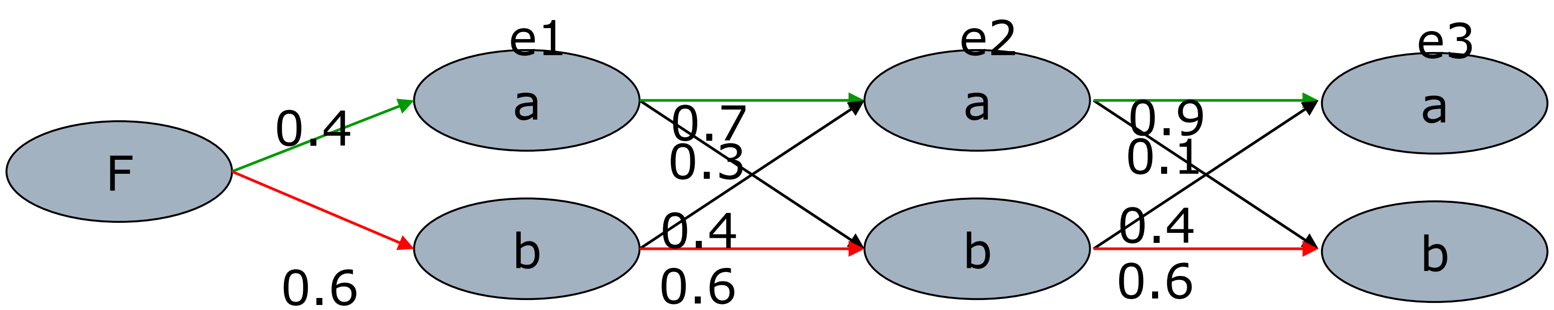


Beam Search

$s(v,n)$ 以v结尾的最大概率的sequence的概率
$t(v_1,v_2,n)$ 第n-1步从v_1到第n步的v_2的概率

动态规划递推公式
$max_{e_1^3} p(e_1,e_2,e_3|F) = \max\{s(v,3)|v = 1,\dots,V\}$
$s(v=j,n) = \max\{s(v=i,n-1) * t(i,j,n)|\ i = 1,\dots,V\}$

$s(v,n)$	n = 1	n = 2	n = 3
v = 1	0.4(F)		
v = 2	0.6(F)		

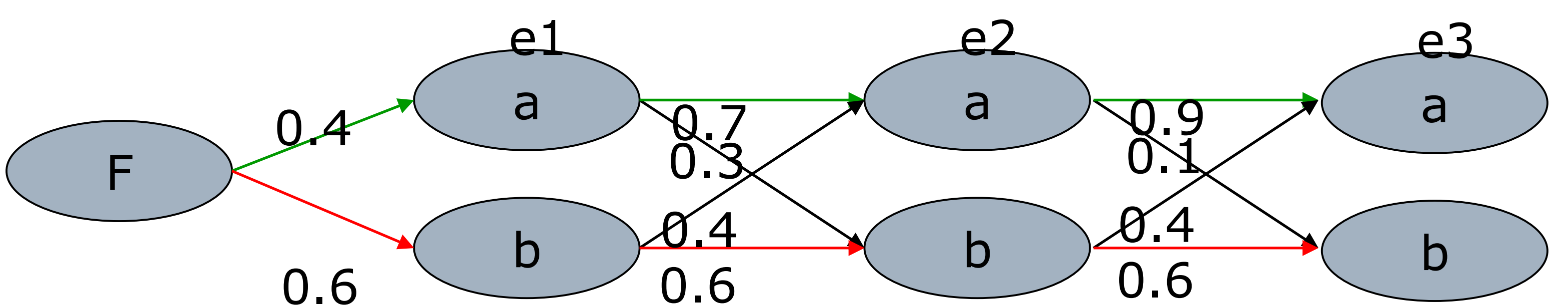


Beam Search

$s(v,n)$ 以v结尾的最大概率的sequence的概率
$t(v_1,v_2,n)$ 第n-1步从v_1到第n步的v_2的概率

动态规划递推公式
$max_{e_1^3} p(e_1,e_2,e_3|F) = \max\{s(v,3)|v = 1,\ldots,V\}$
$s(v=j,n) = \max\{s(v=i,n-1) * t(i,j,n)|\, i = 1,\ldots,V\}$

$s(v,n)$	n = 1	n = 2	n = 3
v = 1	0.4(F)	0.28(1)	
v = 2	0.6(F)	0.36(2)	



Beam Search

$s(v,n)$ 以v结尾的最大概率的sequence的概率

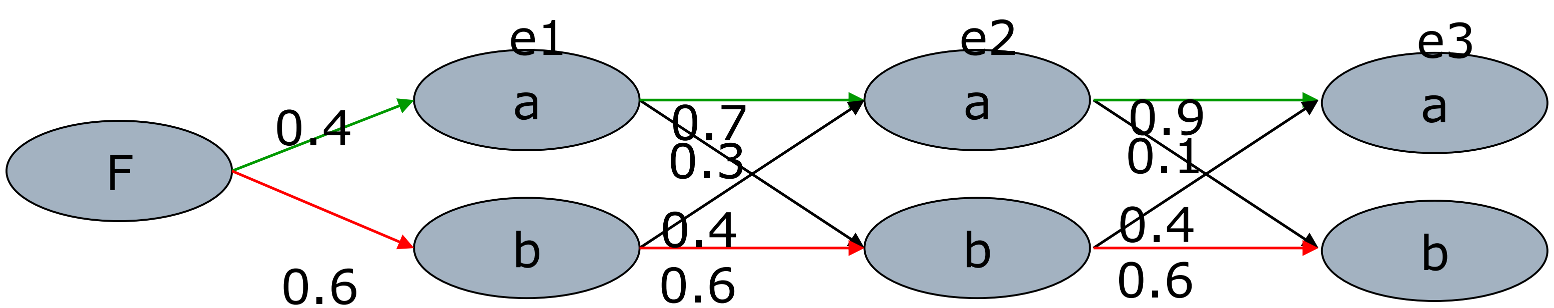
$t(v_1,v_2,n)$ 第n-1步从v_1到第n步的v_2的概率

动态规划递推公式

$max_{e_1^3} p(e_1,e_2,e_3|F) = \max\{\mathrm{s(v,3)|v = 1,\dots,V}\}$

$s(v=j,n) = \max\{s(v=i,n-1) * t(i,j,n)|\, i = 1,\dots,V\}$

$s(v,n)$	n = 1	n = 2	n = 3
v = 1	0.4(F)	0.28(1)	0.252(1)
v = 2	0.6(F)	0.36(2)	0.216(2)



Beam Search

$s(v,n)$ 以v结尾的最大概率的sequence的概率
$t(v_1,v_2,n)$ 第n-1步从v_1到第n步的v_2的概率

动态规划递推公式
$max_{e_1^3} p(e_1,e_2,e_3|F) = \max\{s(v,3)|v = 1,\ldots,V\}$
$s(v=j,n) = \max\{s(v=i,n-1) * t(i,j,n)|\, i = 1,\ldots,V\}$

$s(v,n)$	n = 1	n = 2	n = 3
v = 1	0.4(F)	0.28(1)	0.252(1)
v = 2	0.6(F)	0.36(2)	0.216(2)

计算复杂度 ： $O(V^2N)$ 空间复杂度： $O(VN)$

加入我们要生成所有一句话的时候， V > 40000

Beam Search

$s(v,n)$ 以v结尾的最大概率的sequence的概率
$t(v_1,v_2,n)$ 第n-1步从v_1到第n步的v_2的概率

动态规划递推公式
$max_{e_1^3} p(e_1,e_2,e_3|F) = \max\{s(v,3)|v = 1,\dots,V\}$
$s(v = j,n) = \max\{s(v = i,n-1) * t(i,j,n)|\ i = 1,\dots,V\}$

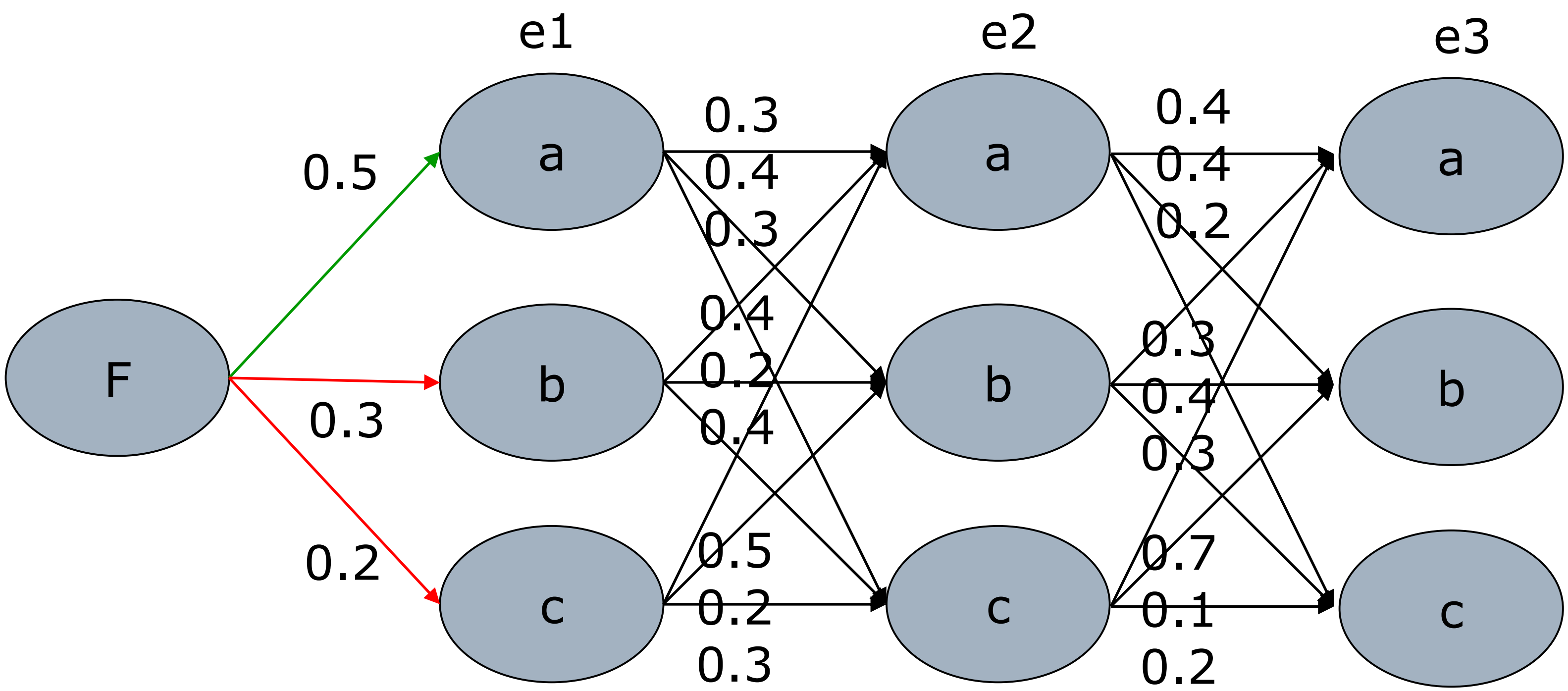
$s(v,n)$	n = 1	n = 2	n = 3
v = 1	0.4(F)	0.28(1)	0.252(1)
v = 2	0.6(F)	0.36(2)	0.216(2)

计算复杂度 ： $O(V^2N)$ 空间复杂度： $O(VN)$

加入我们要生成所有一句话的时候， V > 40000

Beam Search

- Beam Search



Beam Search

- 大家自己做一遍Viterbi Algorithm

$s(v,n)$	n = 1	n = 2	n = 3
v = 1			
v = 2			
v = 3			

Beam Search

- Beam Search
  - Beam Size B << V
    - $h(b,n)$ 在(b,n) 位置的已经生成的sequence
    - $s(b,n) = p(h(b,n))$
  - 递推算法：
    - 已知 s(b,n)和h(b,n)， 求：s(b,n+1), h(b,n+1)
    - temp = [(h(b,n)+v, p(h(b,n)+v) | b = 1,…,B, v=1,…,V]
    - temp = sort(temp, key = lambda x: [1])
    - b(i,n+1), s(i,n+1) = temp[i], i=1,…,B

Beam Search

- Beam Search
  - beam size = 2
    - 已知 s(b,n)和h(b,n)， 求： s(b,n+1), h(b,n+1)
    - temp = [(h(b,n)+v, p(h(b,n)+v) | b = 1,…,B, v=1,…,V]
    - temp = sort(temp, key = lambda x: [1])
    - b(i,n+1), s(i,n+1) = temp[i], i=1,…,B

h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5		
b = 2	b,0.3		

temp = [(a,0.5),(b,0.3),(c,0.2)]

Beam Search

- Beam Search
  - beam size = 2
    - 已知 s(b,n)和h(b,n)， 求：s(b,n+1), h(b,n+1)
    - temp = [(h(b,n)+v, p(h(b,n)+v) | b = 1,…,B, v=1,…,V]
    - temp = sort(temp, key = lambda x: [1])
    - b(i,n+1), s(i,n+1) = temp[i], i=1,…,B

h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5	ab, 0.20	
b = 2	b,0.3	ac, 0.15	

temp = [(aa,0.5*0.3),(ab,0.5*0.4),(ac,0.5*0.3),
(ba,0.3*0.4),(bb,0.3*0.2),(bc,0.3*0.4)]

Beam Search

- Beam Search
  - beam size = 2
    - 已知 s(b,n)和h(b,n)， 求：s(b,n+1), h(b,n+1)
    - temp = [(h(b,n)+v, p(h(b,n)+v) | b = 1,…,B, v=1,…,V]
    - temp = sort(temp, key = lambda x: [1])
    - b(i,n+1), s(i,n+1) = temp[i], i=1,…,B

h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5	ab, 0.20	aca, 0.105
b = 2	b,0.3	ac, 0.15	abb,0.08

temp = [(aba,0.2*0.3),(abb,0.2*0.4),(abc,0.2*0.3),
(aca,0.15*0.7),(acb,0.15*0.1),(acc,0.15*0.2)]

Beam Search

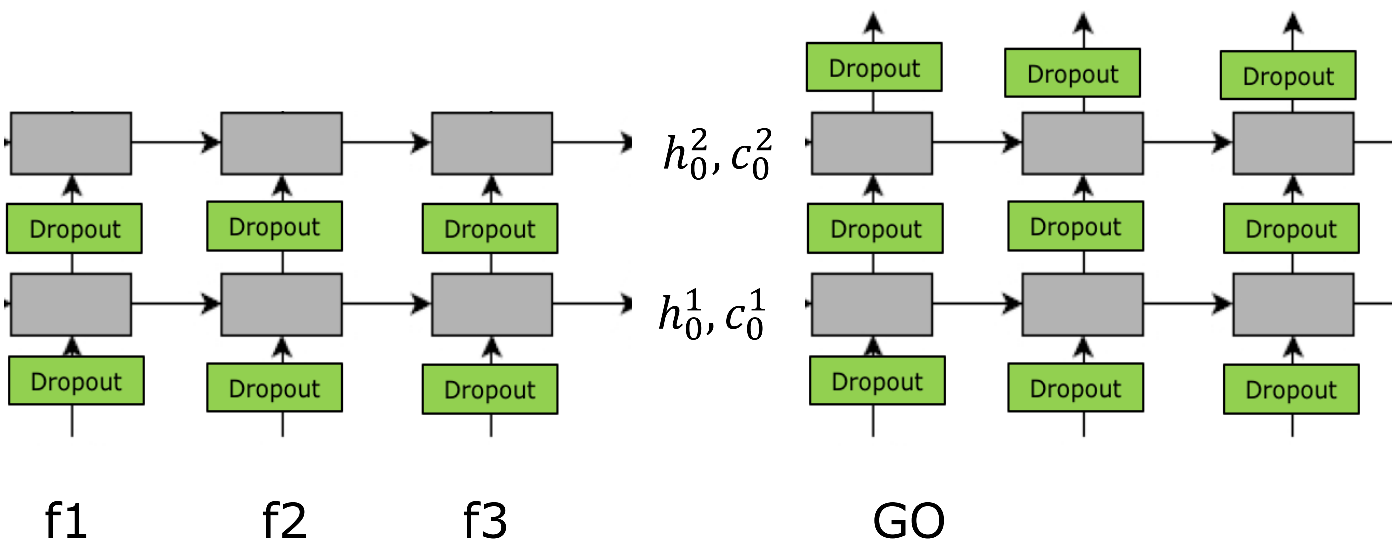
- Beam Search
  - 计算复杂度： $O(\mathrm{B}log(V)VN)$
  - 空间复杂度： $O(\mathrm{B}N)$

h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5	ab, 0.20	aca, 0.105
b = 2	b,0.3	ac, 0.15	abb,0.08

Beam Search

h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5	ab, 0.20	aca, 0.105
b = 2	b,0.3	ac, 0.15	abb,0.08

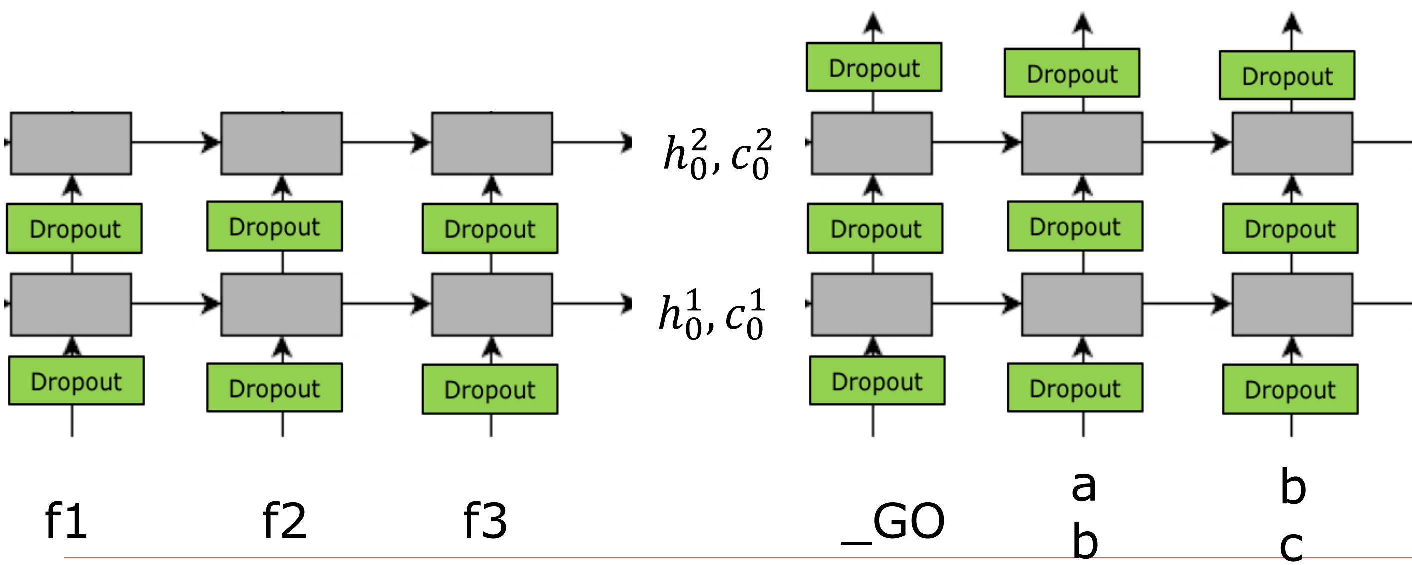
a:0.5 b:0.3 c:0.2



Beam Search

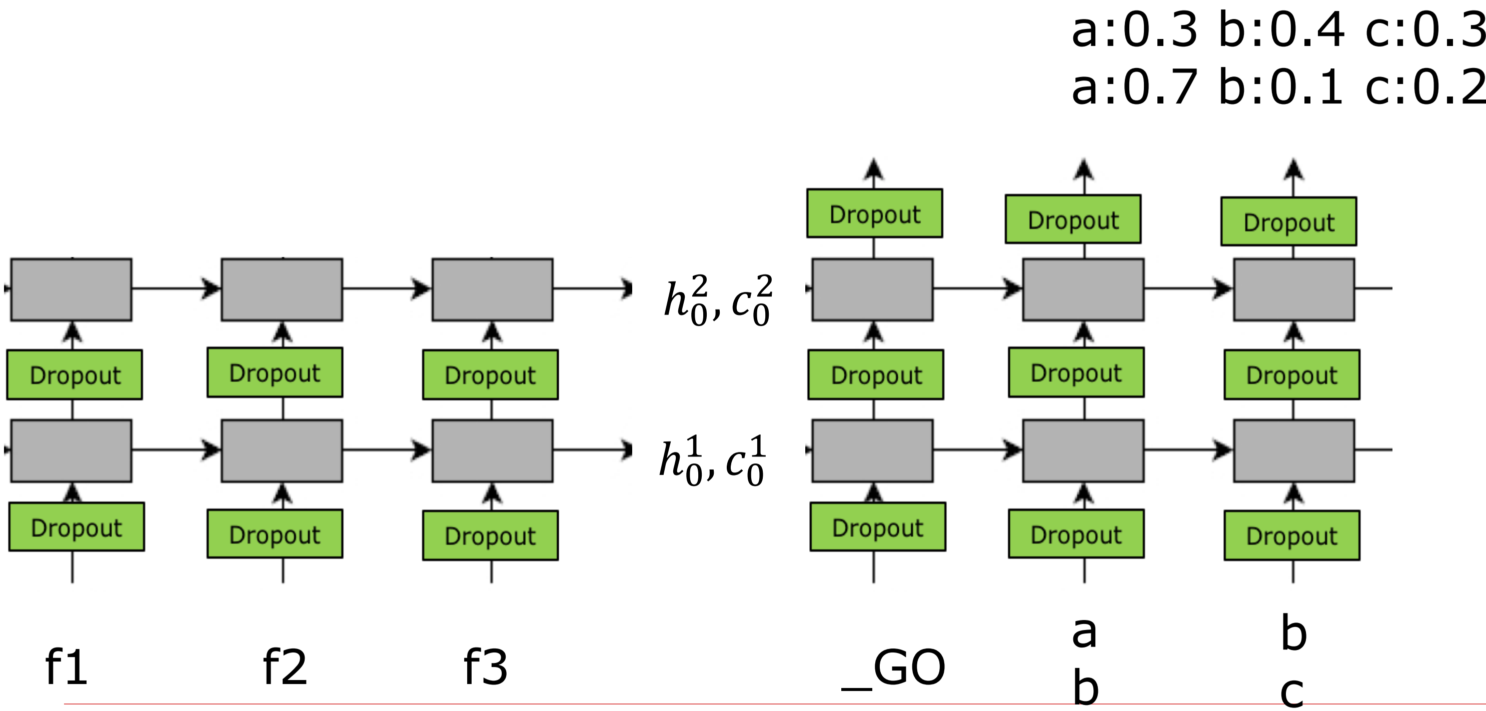
h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5	ab, 0.20	aca, 0.105
b = 2	b,0.3	ac, 0.15	abb,0.08

a:0.3 b:0.4 c:0.3
a:0.4 b:0.2 c:0.4



Beam Search

h(b,n), $s(b,n)$	n = 1	n = 2	n = 3
b = 1	a,0.5	ab, 0.20	aca, 0.105
b = 2	b,0.3	ac, 0.15	abb,0.08

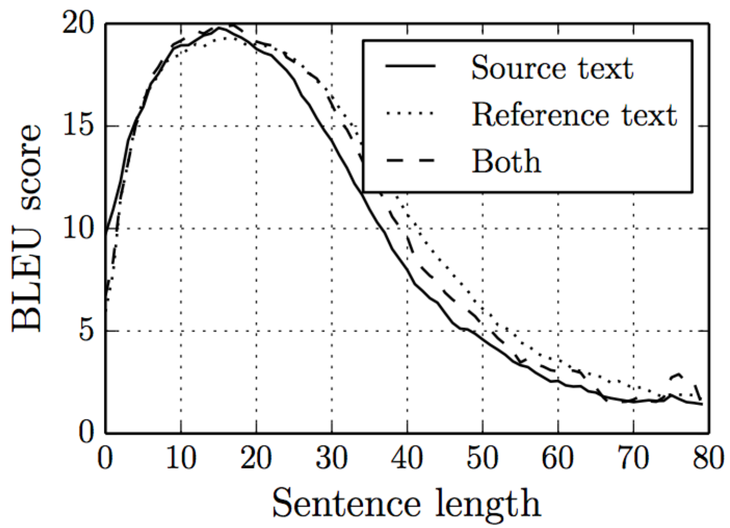


Beam Search

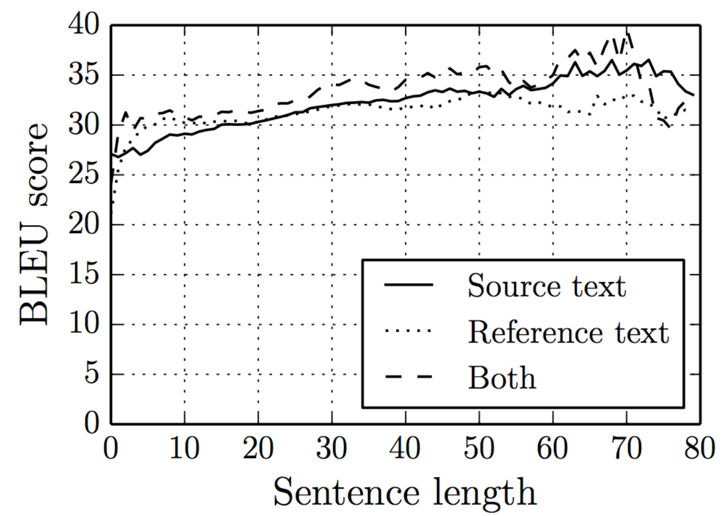
- 求最大概率路径
  - Viterbi Algorithm 全局最优解
    - 计算复杂度$O(V^2N)$
    - LSTM计算复杂度 $O(VN)$
    - 空间复杂度 $O(VN)$
  - Beam Search 近似最优解
    - 计算复杂度 $O(\mathrm{B}log(V)VN)$
    - LSTM计算复杂度 $O(BN)$
    - 空间复杂度$O(BN)$
  - 贪心算法 (Beam Size = 1)

Attention

- N-gram 有限的历史 (n)
- LSTM 无限的历史？
- 以机器翻译为例：



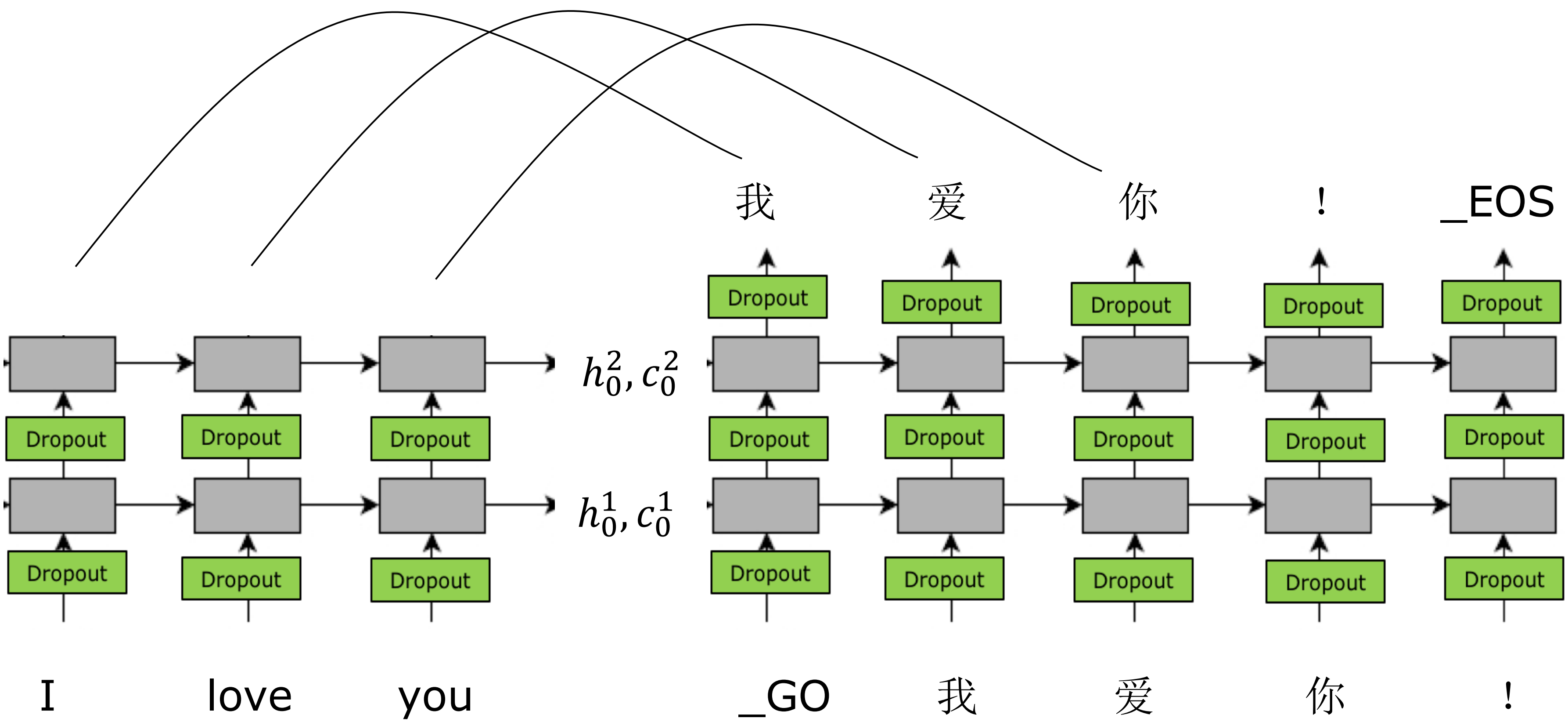
Encoder-Decoder



传统机器翻译

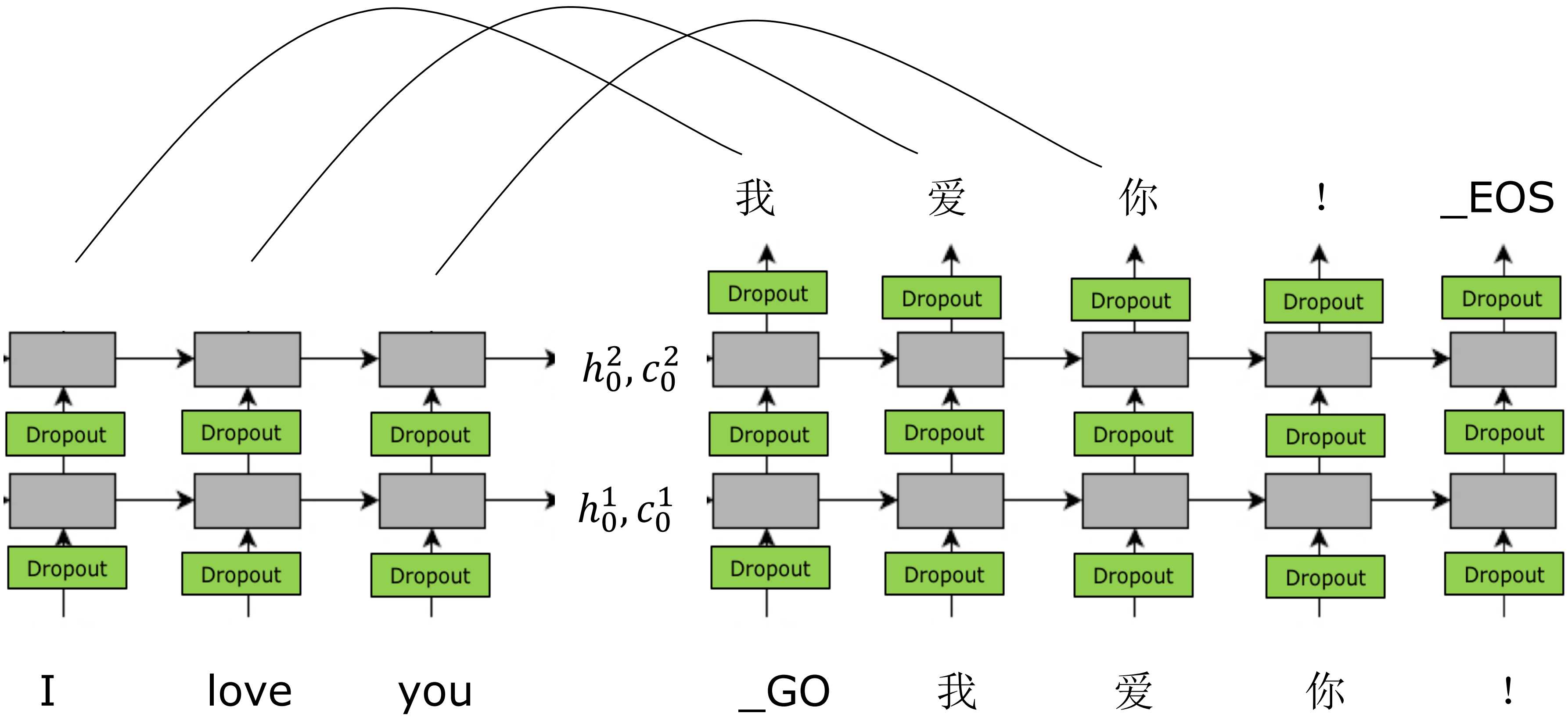
Attention

- 越长距离的关系，LSTM的能力在下降



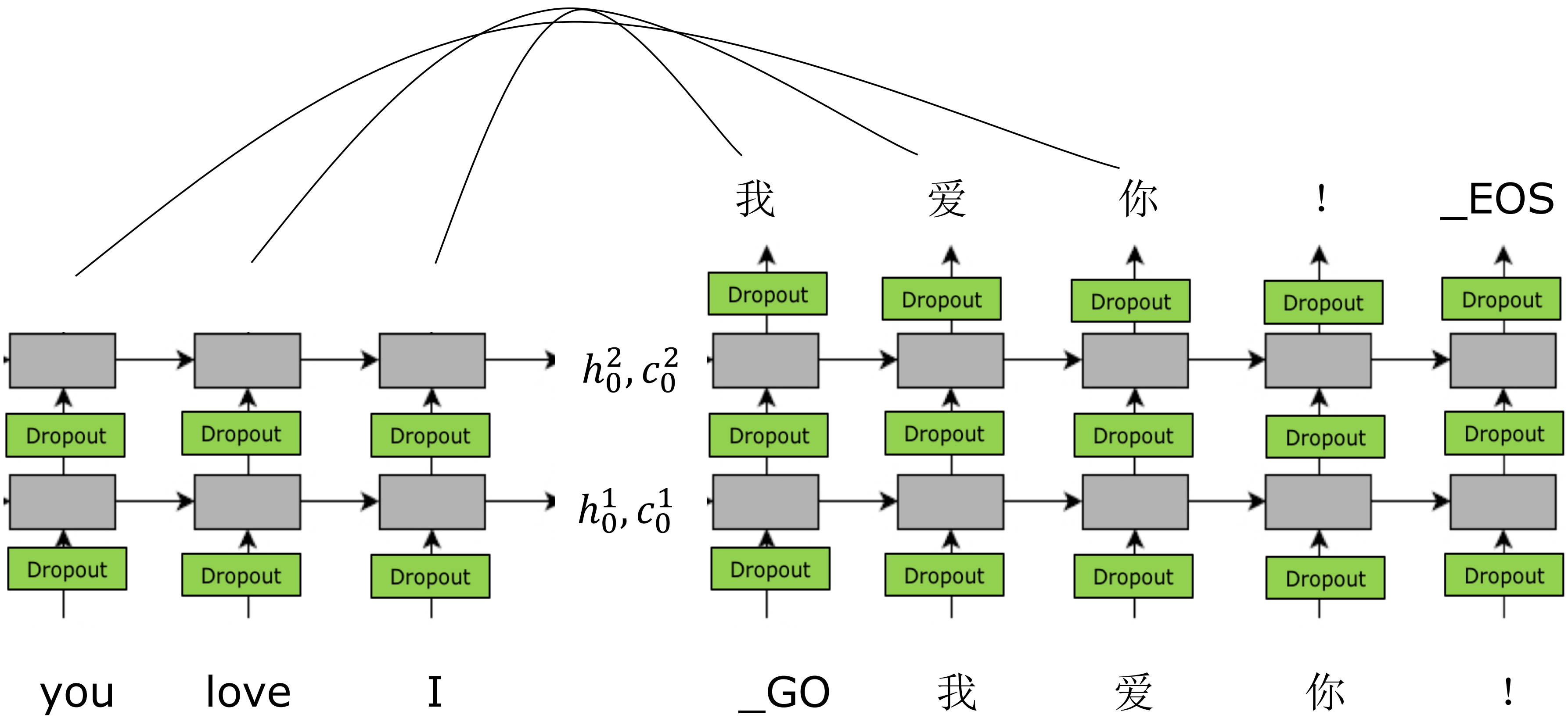
Attention

- 越长距离的关系，LSTM的能力在下降
  - Forward/Backward propagation所要经历的步数



Attention

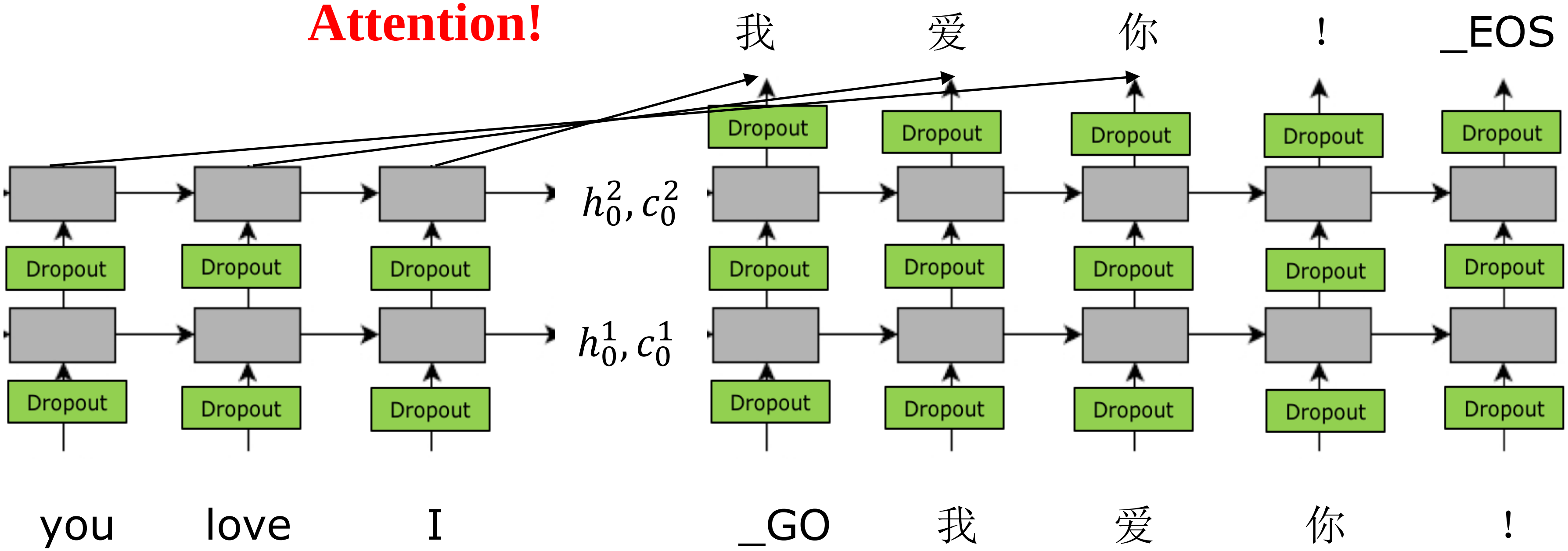
- 越长距离的关系，LSTM的能力在下降
  - 将source sentence倒序输入



Attention

- 越长距离的关系，LSTM的能力在下降
  - 如何减少B/F propagation的步数？
    - 增加 Skip Connection
    - 如何根据输入和输出动态的选择connection？

Attention!



Attention

- Attention

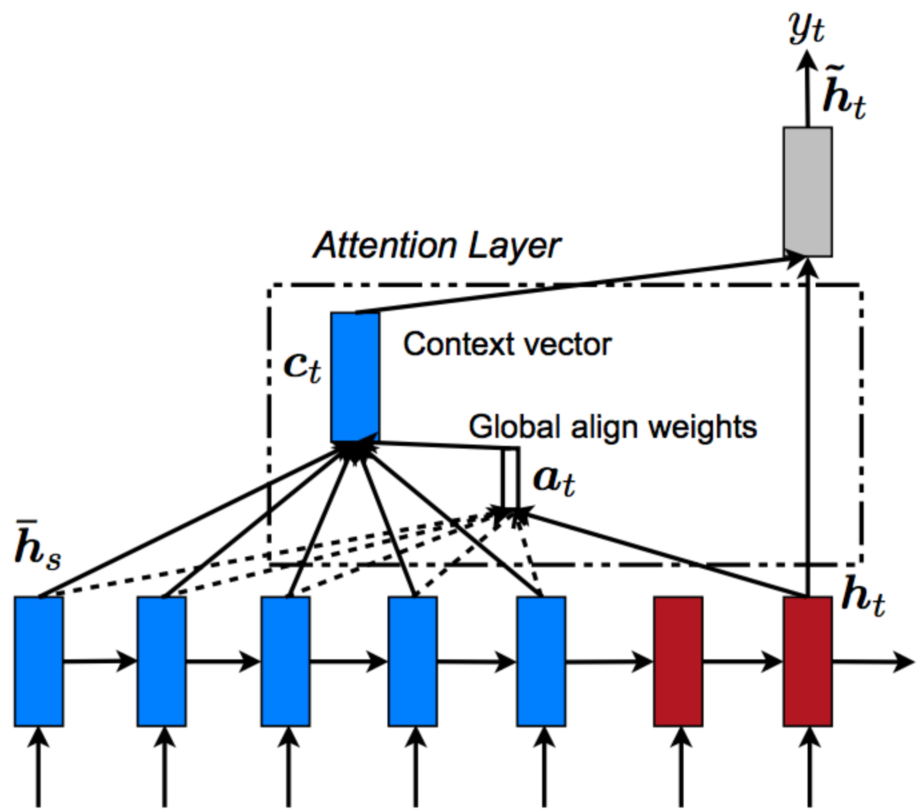


Figure from https://arxiv.org/pdf/1508.04025.pdf

Attention

Attention

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

$$\boldsymbol{a}_t = \mathrm{softmax}(\boldsymbol{W_a} \boldsymbol{h}_t) \qquad \textit{location}$$

$$\boldsymbol{a}_t(s) = \mathrm{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

本质是Softmax
所以，$\sum_s a_t(s) = 1$

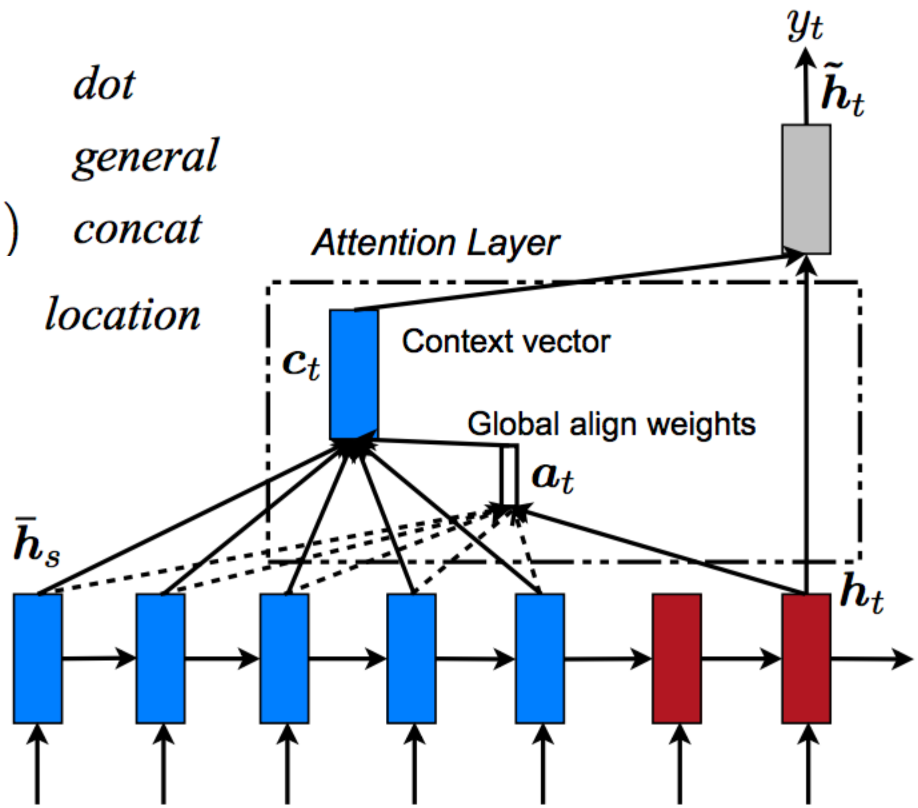


Figure from https://arxiv.org/pdf/1508.04025.pdf

Attention

- Attention

$$c_t = \sum_{s'} a_t(s')\,\overline{h_{s'}}$$

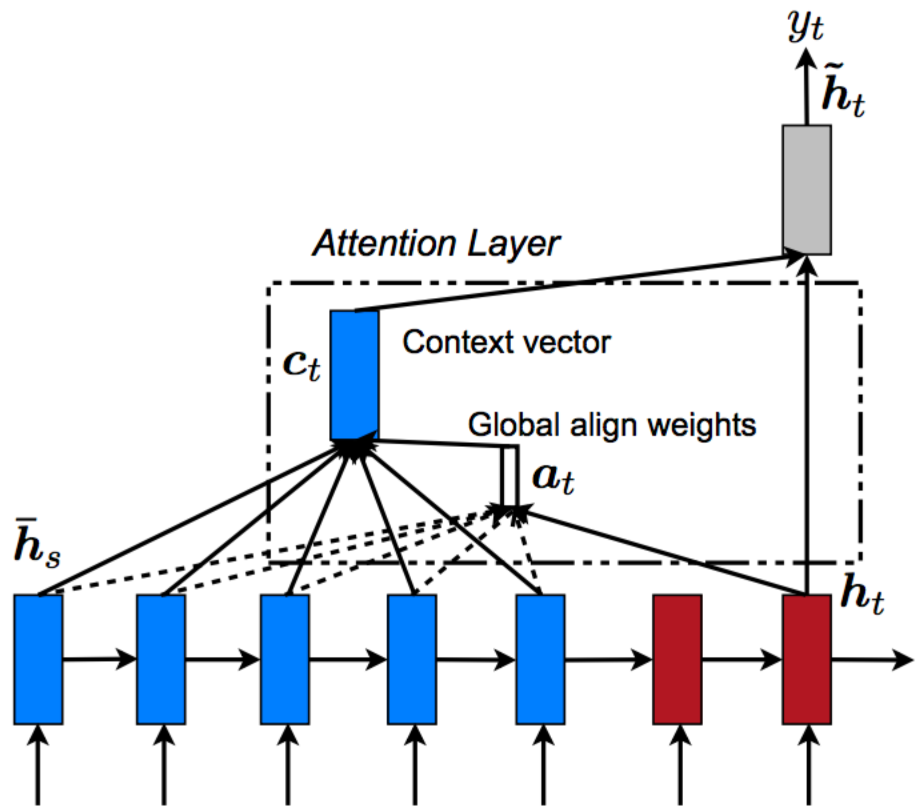


Figure from https://arxiv.org/pdf/1508.04025.pdf

Attention

Attention

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

现在的forward/backward 的path是怎样的？

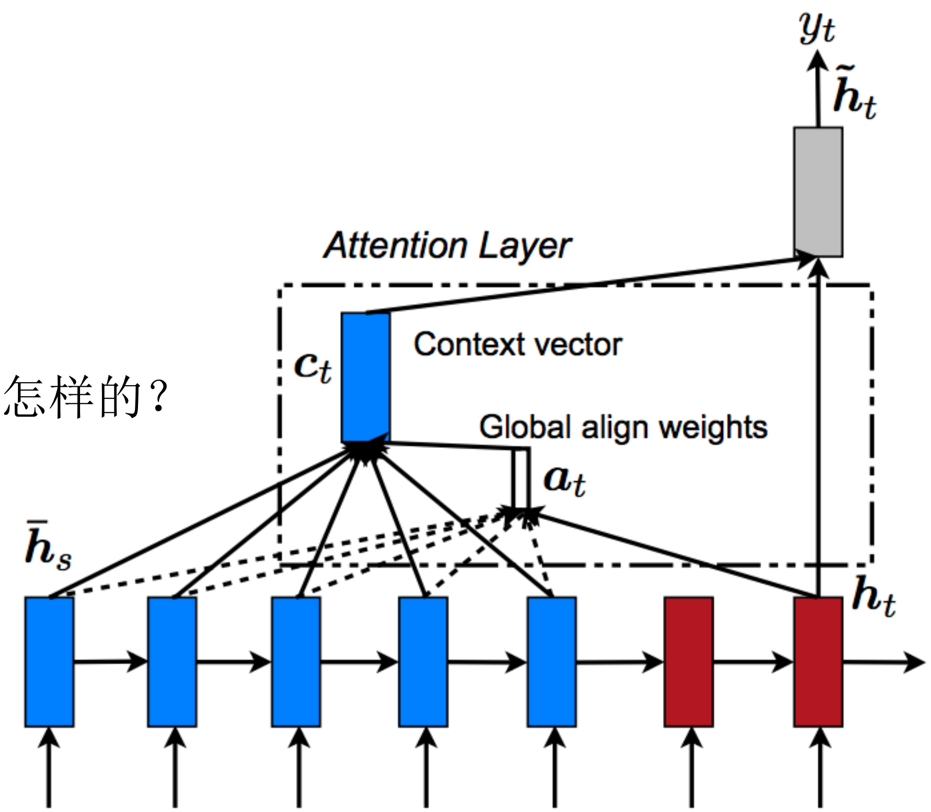


Figure from https://arxiv.org/pdf/1508.04025.pdf

Attention

- Attention
  - feed-input：下一个单词知道上一个单词的attention

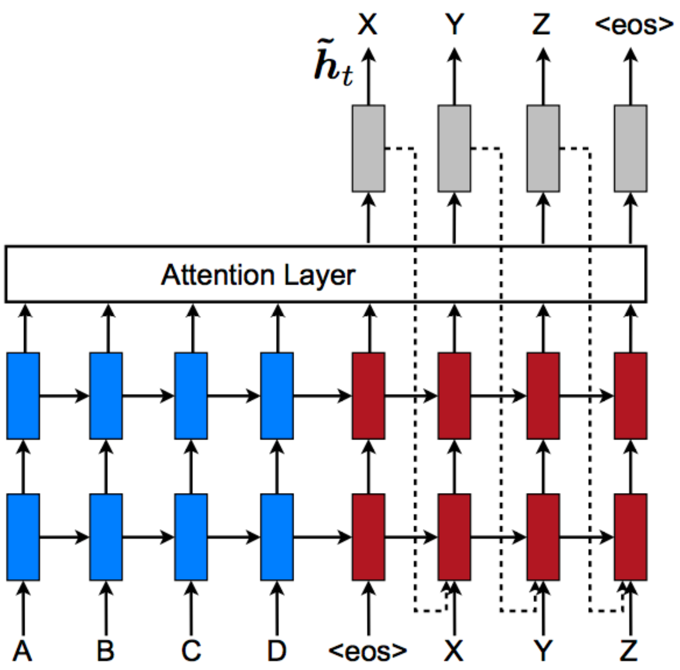


Figure from https://arxiv.org/pdf/1508.04025.pdf

Attention

- Seq2Seq的技巧总结：
  - dropout + reverse + attention + feed-input

System	Ppl	BLEU
Winning WMT'14 system – *phrase-based* + *large LM* (Buck et al., 2014)		20.7
Base	10.6	11.3
Base + reverse	9.9	12.6 (+*1.3*)
Base + reverse + dropout	8.1	14.0 (+*1.4*)
Base + reverse + dropout + global attention (*location*)	7.3	16.8 (+*2.8*)
Base + reverse + dropout + global attention (*location*) + feed input	6.4	18.1 (+*1.3*)

Figure from https://arxiv.org/pdf/1508.04025.pdf

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop