揚州大學
YANGZHOU UNIVERSITY

自 然 语 言 处 理

Natural language Processing

机器翻译

第7章

CONTENTS

01 语言差异与类型学

02 RNN编码器解码器模型

03 训练编解码器

04 注意力机制

05 波束搜索

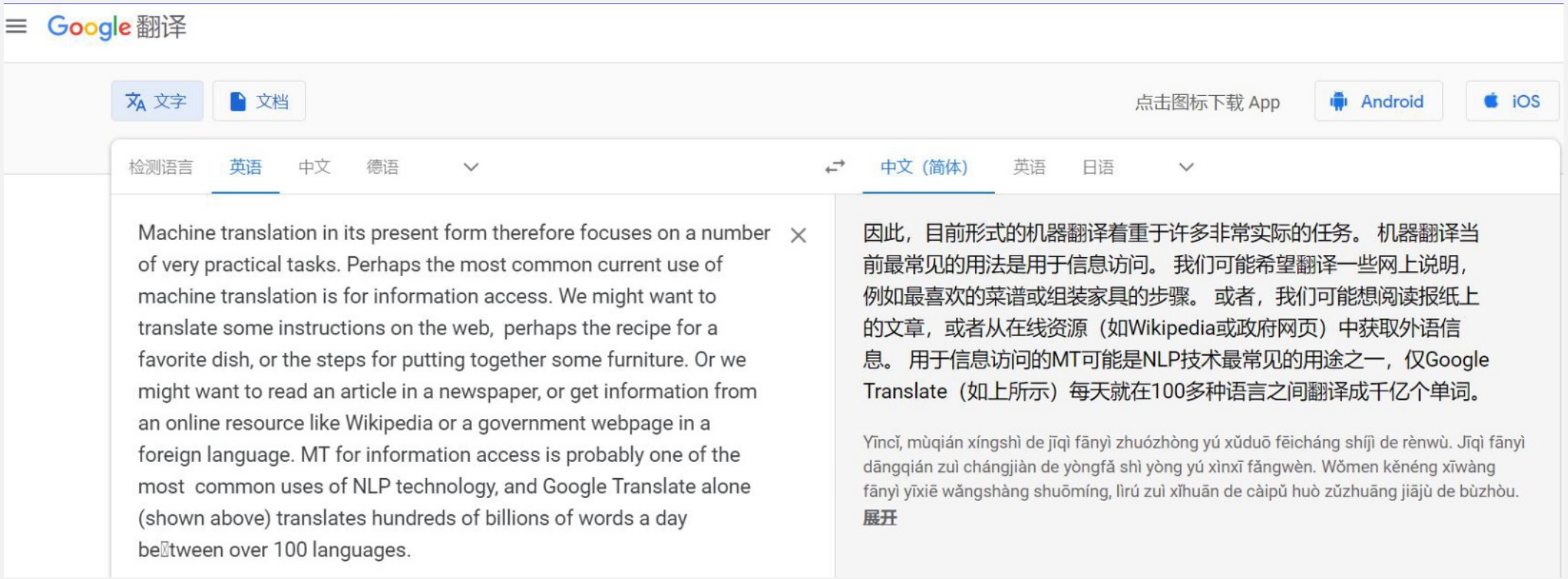
06 编码解码器与Transformer

07 构建机器翻译系统

08 机器翻译评估

什么是机器翻译

使用机器 - 将一种语言翻译为另一种语言



机器翻译用途

信息访问- information access

- 通过阅读外文信息获取知识信息，如：
- 新闻
- 论文
- 代码注释

计算机辅助翻译- computer-aided translation- CAT

- 辅助人工翻译(如我们正在使用的该课程书籍中文版)
  - 生成草稿翻译
  - 由人工翻译在后期编辑阶段进行巩固

机器翻译用途

通常用于“本地化”内容或产品来适应特定语言任务

机器翻译的最新应用为满足人们即时的交流需求

- 同声传译
- 增量式翻译
- 实时翻译

MT实现算法

标准算法

- 编码器解码器(encoder-decoder)或 序列到序列(sequence-to-sequence)

- 可用RNN或Transformer实现

不同语言之间的语法结构不同,要求机器翻译不能只按照词序进行翻译。而需要用一个**复杂函数**将输入序列映射到输出序列

English:	*He wrote a letter to a friend*		
Japanese:	*tomodachi ni tegami-o kaita*		
	friend	to letter	wrote

不同语言之间的差异

大会/General Assembly 在/on 1982年/1982 12月/December 10日/10 通过了/adopted 第37号/37th 决议/resolution， 核准了/approved 第二次/second 探索/exploration 及/and 和平peaceful 利用/using 外层空间/outer space 会议/conference 的/of 各项/various 建议/suggestions 。

On 10 December 1982 , the General Assembly adopted resolution 37 in which it endorsed the recommendations of the Second United Nations Conference on the Exploration and Peaceful Uses of Outer Space .

在这个例子里，可以找到两种语言之间的不同有哪些？

编码器-解码器

编解码器可以很好的处理机器翻译中复杂映射关系。

也可以用于处理其他涉及两个序列复杂映射关系的任务，包括：

- 摘要提取(summarization)
- 对话系统(dialogue)
- 语义解析(semantic parsing)
- 等等

Part.01

语言差异与类型学

语言之间的差异

不同语言之间的共性：

- 名词 – 表示人物、地点等
- 动词 – 表示行为、状态等
- 形容词 – 表示人或事物的性质、状态、特征等
- ……

合理的探究是什么导致语言间的翻译差异(translation divergences)

- 有助于构造良好的机器翻译模型

语言类型学(linguistic typology)

用于探究语言之间的系统性差异和相似(不同语言的特征)

可以为机器翻译的质量提升提供有效的理论支持

世界语言结构图谱(World Atlas of Language Structures)

https://wals.info/

词序类型学(word order typology)

主谓宾结构(Subject – Verb – Object)

- 英语、法语、德语、汉语等

主宾谓结构(Subject – Object – Verb)

- 印地语、日语等

谓主宾结构(Verb – Subject –Object)

- 爱尔兰语、阿拉伯语等

词序相同的语言之间有一定共性，

例如：

- VO结构下的语言通常有介词
- OV结构下的语言通常有后置词

举例：词序差异

English: *He wrote a letter to a friend*

Japanese: *tomodachi ni tegami-o kaita*
friend to letter wrote

Arabic: *katabt risāla li ṡadq*
wrote letter to friend

举例：词序差异

英语
The green witch
is
at home
this week
德语
Diese Woche
ist
die grüne Hexe
zu Hause
汉语
cheng long
dao xiang gang
qu
英语
Jackie Chan
went
to Hong Kong

词汇差异

任何翻译中，一个合适的词在不同上下文中的使用也有所不同，例如：

- 英语单词 bass 在西班牙语中可以表示lubina(鱼)或bajo(乐器)
- 英语中的brother、elder brother、younger brother
  - 在中文中分别代表兄弟、哥哥、弟弟

词汇空白，比如中文中的“孝”英文没有对应词或短语直接表达，

  - 需含义近似 filial piety, loving child, or good son/daugther

动词框架：在动词上标记运动方向、(日语、泰米尔语等)

词汇差异

外围词框架：使用外围词辅助动词表示方向。(英语、汉语等)

- crawl out, float off, jump down, run after.

指代密度指的是语言对介词的省略频率，介词越多，指代密度稠密，省略频率越低。

- 汉语指代密度稀疏，英语指代密度稠密

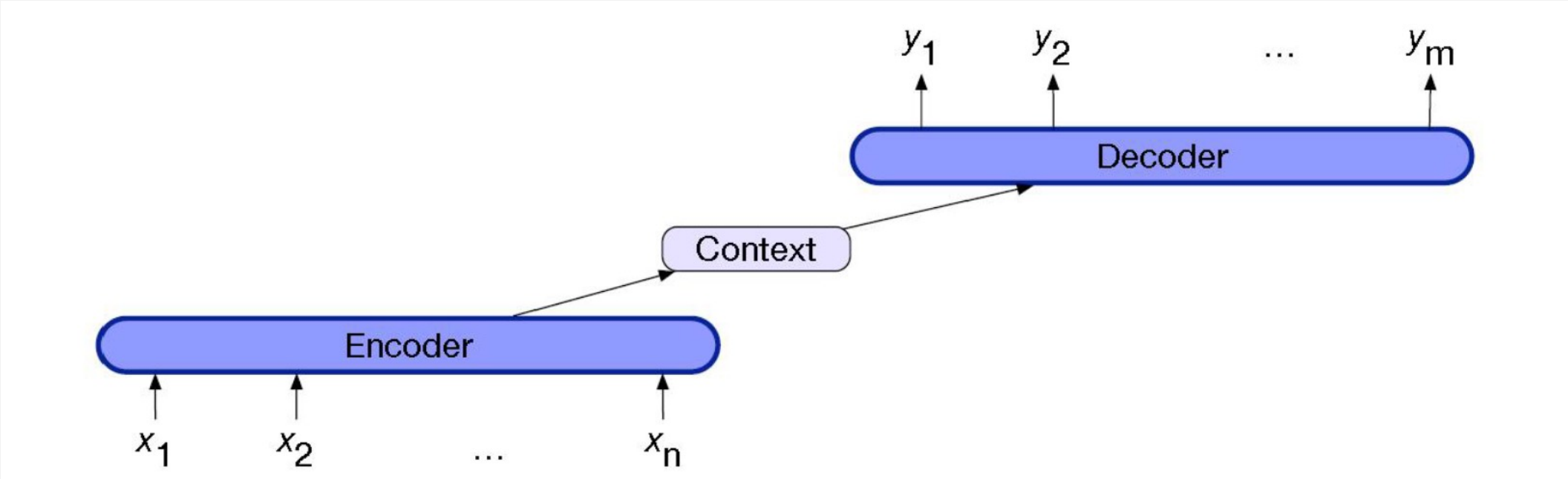
因此需要专业化翻译，以适应不同翻译任务，以歧义消除、词义近似。

Part.02

RNN编码器解码器模型

编码器-解码器模型

可生成上下文相关的任意长度的输出序列。
可用于机器翻译、摘要提取、自动问答或对话系统等前沿应用。



编码器：接受输入 x， 生成上下文表示 h (LSTM、Transformer)
上下文：向量c，是h的函数，将输入的本质传递给解码器
解码器：接受输入 c， 根据表示h生成任意长度的输出序列y

基于RNN的编码器-解码器

回顾RNN中如何计算序列y的概率p(y)

$$p(y) = p(y_1)p(y_2|y_1)p(y_3|y_1,y_2)\ldots P(y_m|y_1,\ldots,y_{m-1})$$

在时间t，序列模型通过传递前缀t-1个tokens，并使用向前推理生成由多个隐藏状态组成的一个序列，最后一个隐藏状态h_{t-1}对应前缀的最后一个单词

之后，使用h_{t-1}作为生成下一token和下一隐藏状态h_t的起点

$$\mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

g 是激活函数

$$\mathbf{y}_t = f(\mathbf{h}_t)$$

f 是softmax函数

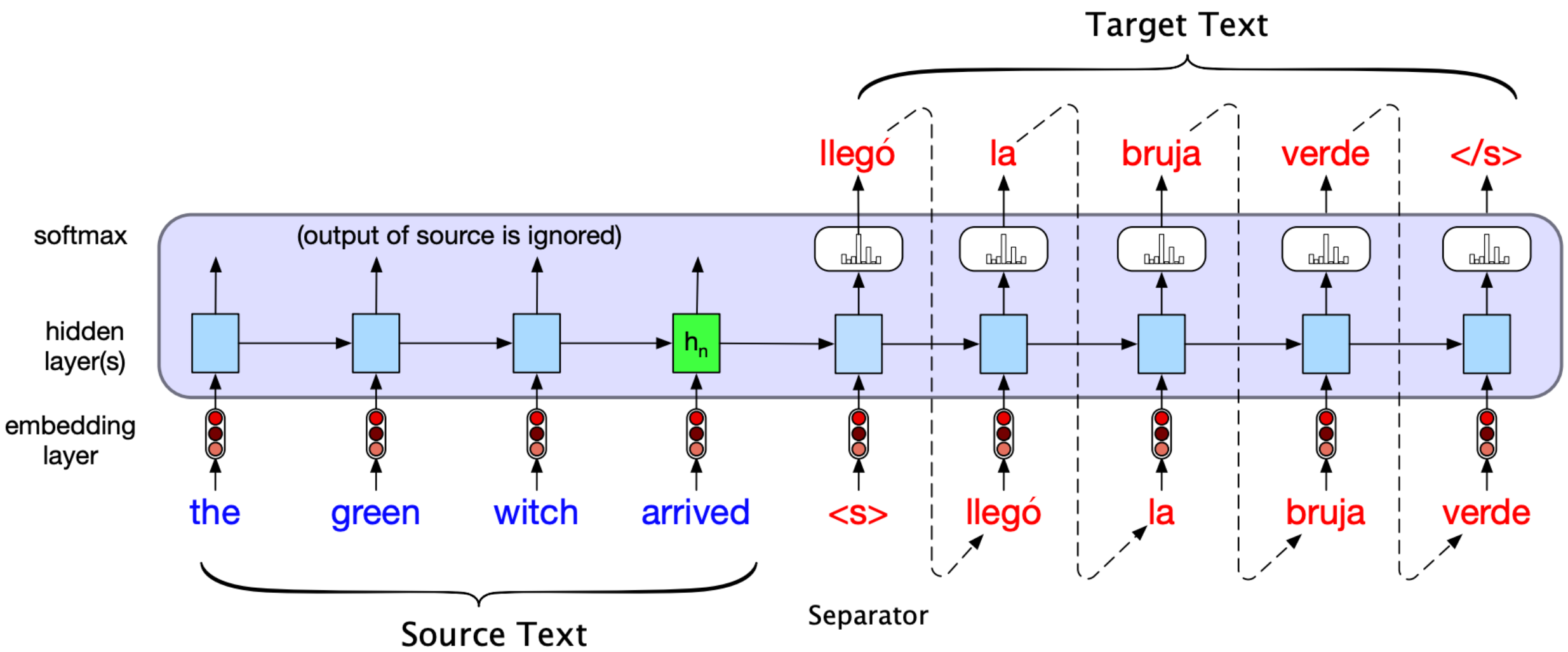
基于RNN的编码器-解码器

将该自回归生成语言模型转换为翻译模型

使用一个分隔符，将源句子(source)与目标句子(target)进行拼接

- 回顾Transformer中用于摘要提取的条件语言模型

基于RNN的编码器-解码器

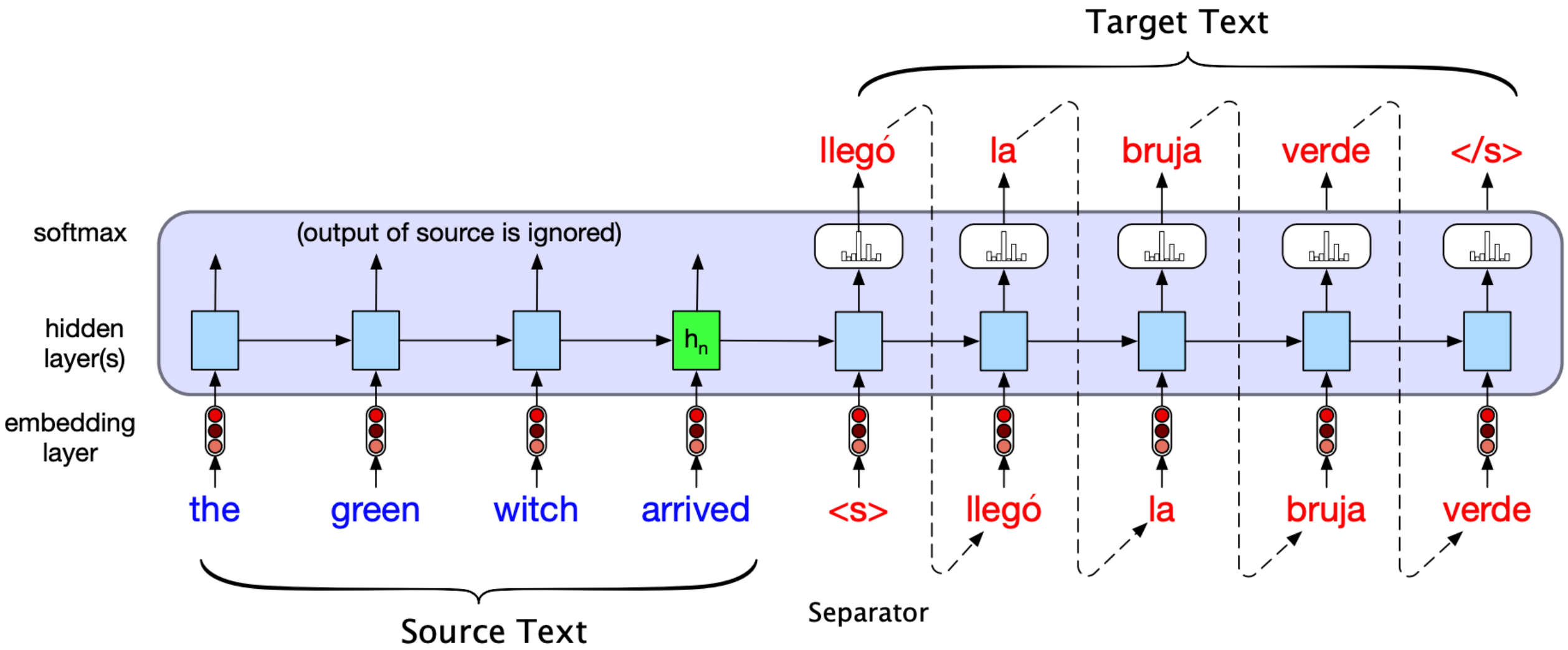


基于RNN的编码器-解码器

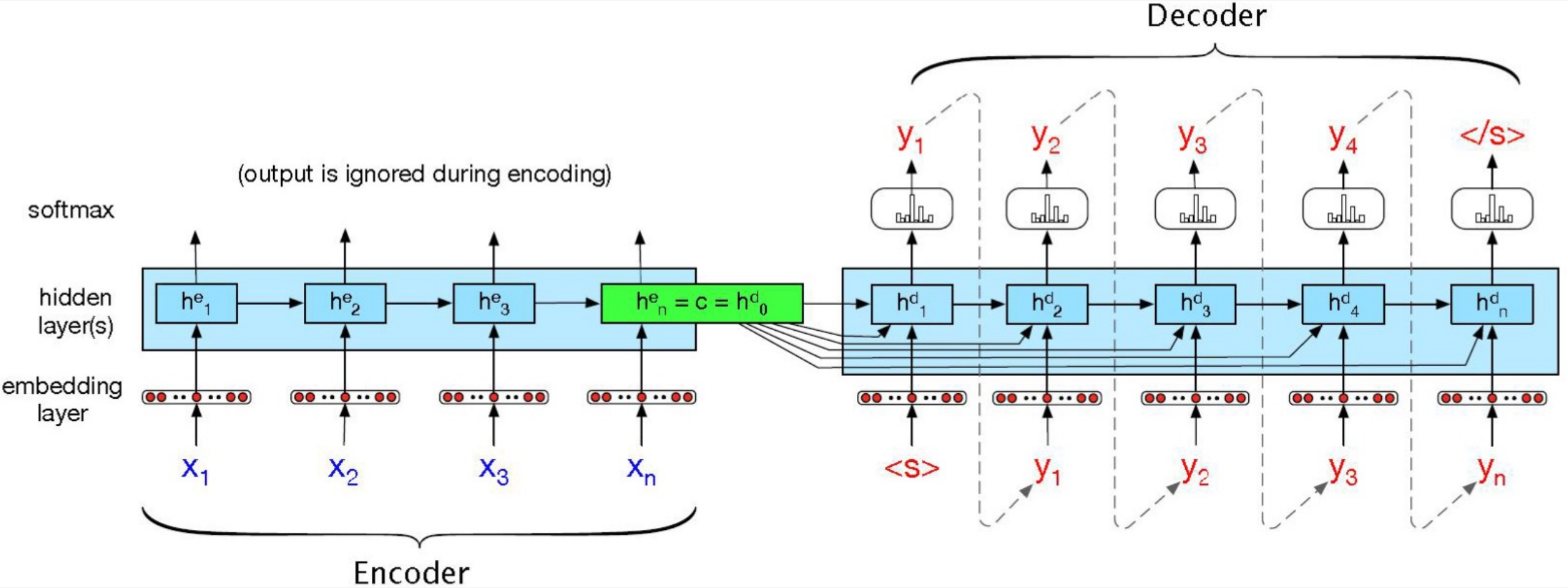
假设源句子为x，目标句子为y，则条件概率 $p(y|x)$ 可以表达为：

$$p(y|x) = p(y_1|x)p(y_2|y_1,x)p(y_3|y_1,y_2,x)...P(y_m|y_1,...,y_{m-1},x)$$

基于RNN的编码器-解码器



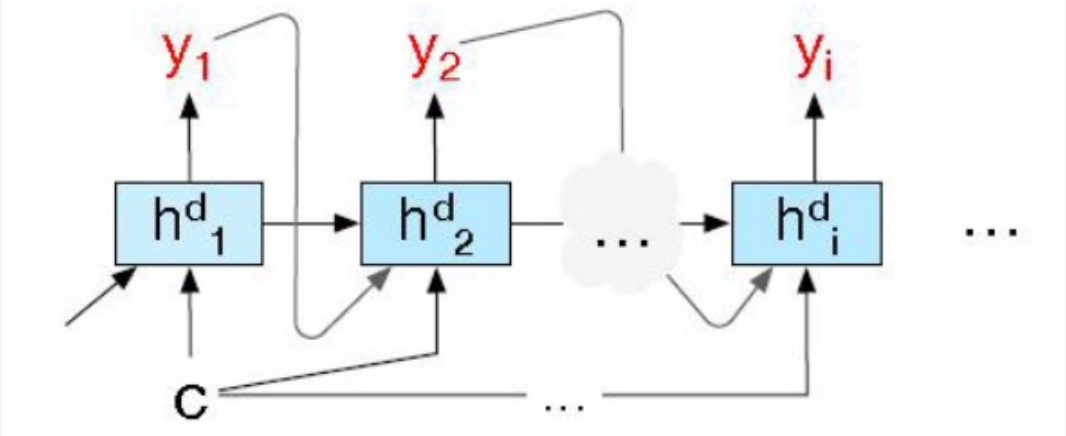
正式的基于RNN编码器-解码器的机器翻译



上下文向量C

在上述方法中，上下文向量C的影响随着序列生成而不断地减弱。

解决方案



$$
\begin{aligned}
\mathbf{c} &= \mathbf{h}_n^e \\
\mathbf{h}_0^d &= \mathbf{c} \\
\mathbf{h}_t^d &= g(\hat{y}_{t-1}, \mathbf{h}_{t-1}^d, \mathbf{c}) \\
\mathbf{z}_t &= f(\mathbf{h}_t^d) \\
y_t &= \mathrm{softmax}(\mathbf{z}_t)
\end{aligned}
$$

$$
\hat{y}_t = \mathrm{argmax}_{\mathrm{w} \in \mathrm{V}} P(w | x, y_1 \ldots y_{t-1})
$$

Part.03

训练编码解码器

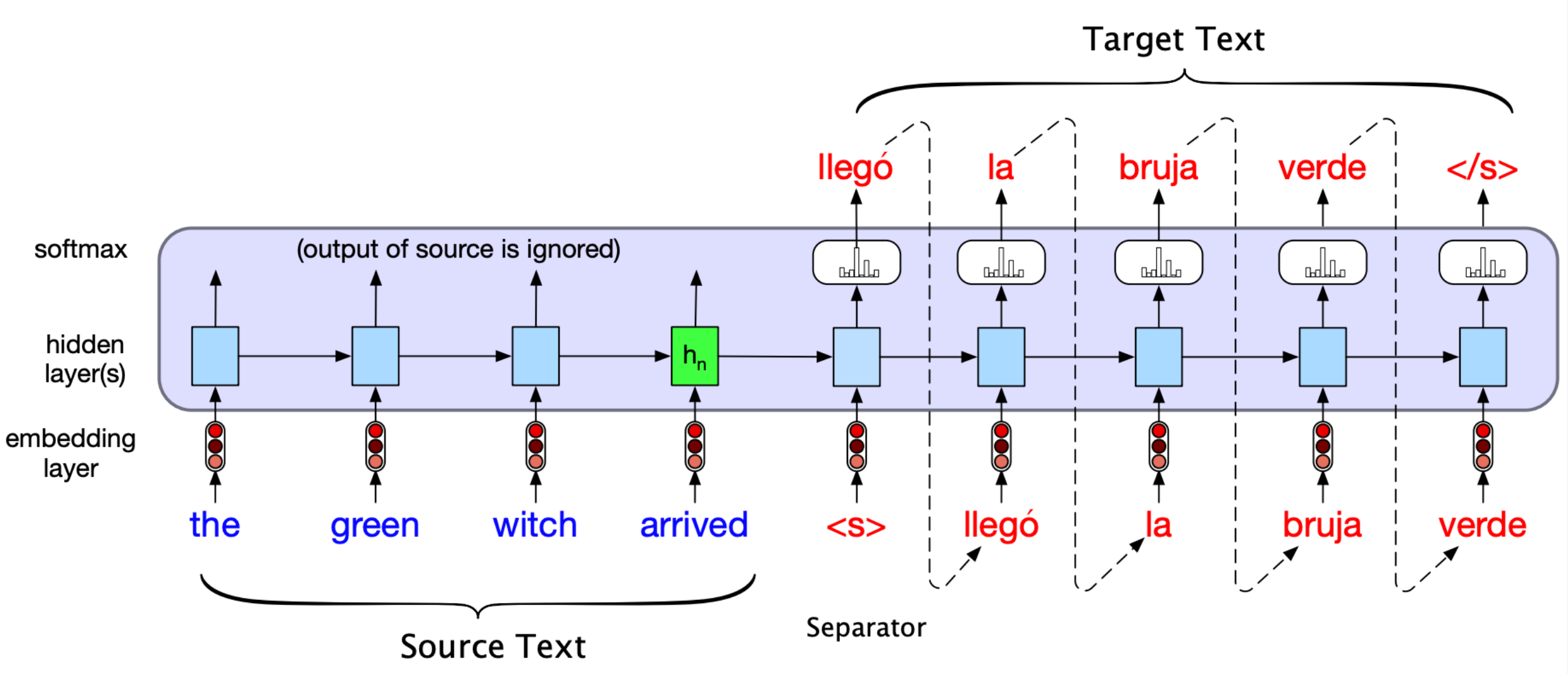
编解码器模型训练

准备训练数据：

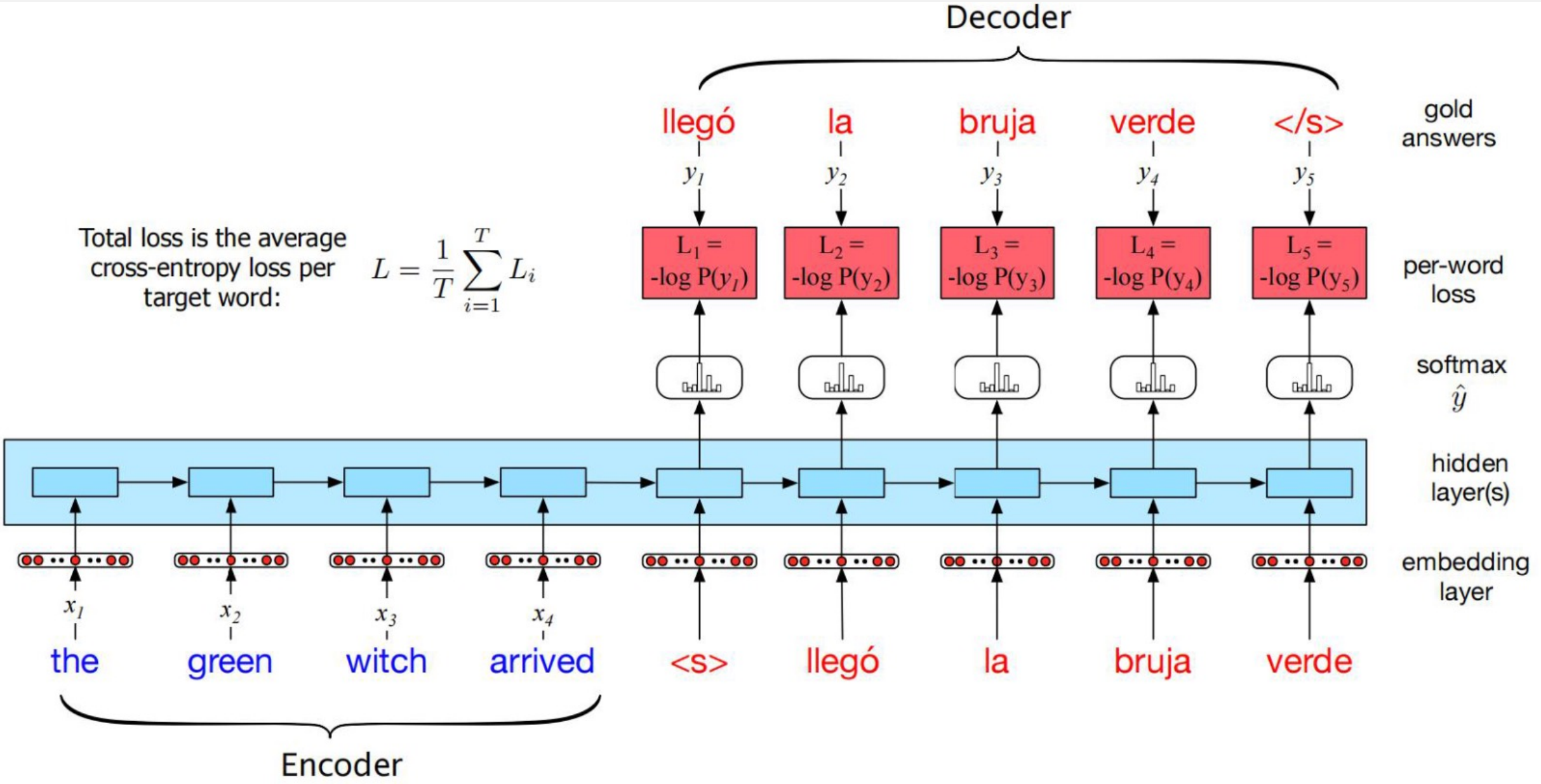
- 每一个训练样本是由一个源文本和一个目标文本组成的二元组
- 将二元组的两个元素通过一个分隔符进行拼接

网络模型使用源文本和分隔符，自回归地训练生成下一单词

回顾 – 模型推理


Target Text
llegó
la
bruja
verde
</s>
softmax
(output of source is ignored)
hidden layer(s)
h_n
embedding layer
the
green
witch
arrived
<s>
llegó
la
bruja
verde
Source Text
Separator

使用教师强制的解码器


Decoder
llegó
la
bruja
verde
</s>
gold answers
y_1
y_2
y_3
y_4
y_5
Total loss is the average cross-entropy loss per target word:
$L = \frac{1}{T}\sum_{i=1}^{T} L_i$
$L_1 = -\log P(y_1)$
$L_2 = -\log P(y_2)$
$L_3 = -\log P(y_3)$
$L_4 = -\log P(y_4)$
$L_5 = -\log P(y_5)$
per-word loss
softmax
$\hat{y}$
hidden layer(s)
embedding layer
x_1
x_2
x_3
x_4
the
green
witch
arrived
<s>
llegó
la
bruja
verde
Encoder

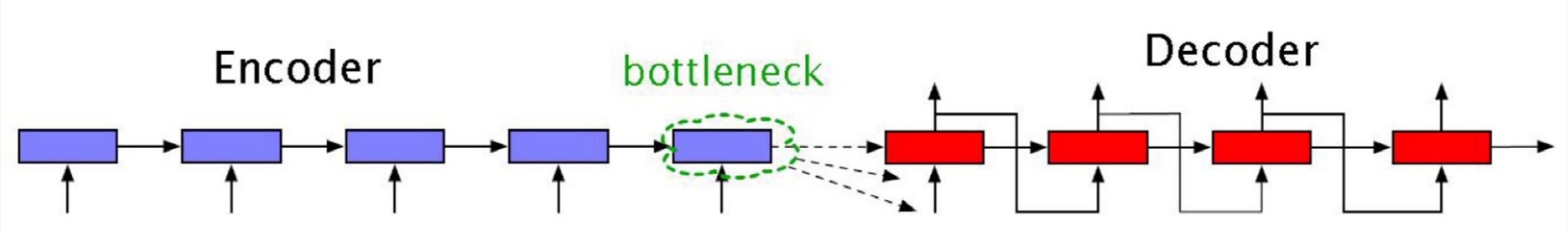
Part.04

注意力机制

隐藏状态是一个瓶颈

解码器自回归生成的唯一依据是隐藏信息-上下文向量c

- 上下文向量c需携带源文本的全部信息
  - 但隐藏状态对长序列文本不能很好地进行表示。



解决方案：注意力机制

- 允许 解码器从编码器获取每一时间步的隐藏信息，而不仅仅是最后一步。

注意力机制

在注意力机制中，上下文向量*c*是用于表示所有编码器隐藏状态的一个函数，即：

$$\mathbf{c} = f(\mathbf{h}_1^e \ldots \mathbf{h}_n^e)$$

为什么不可以直接使用，由所有隐藏状态组成的张量，作为上下文？

- 因为隐藏状态的数量，随着序列长度变化而不断地变化

注意力机制中，通过对编码器中的所有隐藏状态进行加权求和，

- 生成一个固定长度的上下文向量c(即上述函数)

动态生成上下文向量

这些权重用于关注源文本的“特殊”部分，

- 这一特殊部分与解码器当前生成的token是相关的。

- 该特殊部分是会变化的(因为不同时间步下的tokens也不同)
  - 因此，注意力机制生成的上下文向量是动态的

注意力机制根据解码器当前生成的token，
使用编码器的所有隐藏状态，
动态生成上下文向量c

计算解码器当前隐藏状态

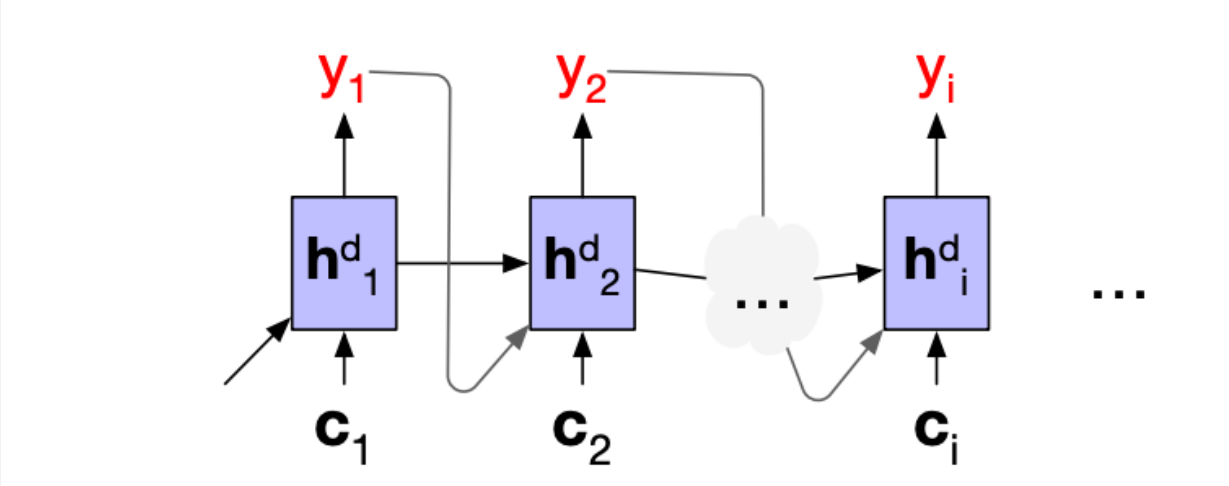
在解码阶段的时间步i上，向量c_i由编码器的所有隐藏状态推导得出。

计算当前隐藏状态(h^d_i)，需要考虑的条件包括：

- 当前上下文向量c_i
- 解码器的先前隐藏状态(h^d_{i-1})
- 解码器先前生成的输出($\hat{y}_{i-1}$)

$$\mathbf{h}_i^d = g(\hat{y}_{i-1}, \mathbf{h}_{i-1}^d, \mathbf{c}_i)$$

计算解码器当前隐藏状态



解码器的先前隐藏状态(h^d_{i-1}) 已知的
解码器先前生成的输出($\hat{y}_{i-1}$) 已知的

当前上下文向量c_i 未知的

$$\mathbf{h}_i^d = g(\hat{y}_{i-1}, \mathbf{h}_{i-1}^d, \mathbf{c}_i)$$

如何动态计算上下文向量c

首先需要计算：

- 编码器中的每一个隐藏状态与解码器中的前一个隐藏状态之间的相关性

如何计算？

使用　　点积注意力

$$score(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{j}^{e}) = \mathbf{h}_{i-1}^{d} \cdot \mathbf{h}_{j}^{e}$$

有了相关性后该如何做？

由于点积的结果是一个标量，无法直接作用在每一个编码器的隐藏状态上(向量)。

因此需要用softmax对所有计算获得的点积结果进行归一化处理，从而获取权重向量α_{ij}

$$
\begin{aligned}
\alpha_{ij} &= \mathrm{softmax}(score(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{j}^{e}) \ \forall j \in e) \\
&= \frac{\exp(score(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{j}^{e})}{\sum_{k} \exp(score(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{k}^{e}))}
\end{aligned}
$$

动态计算：加权平均上下文向量c

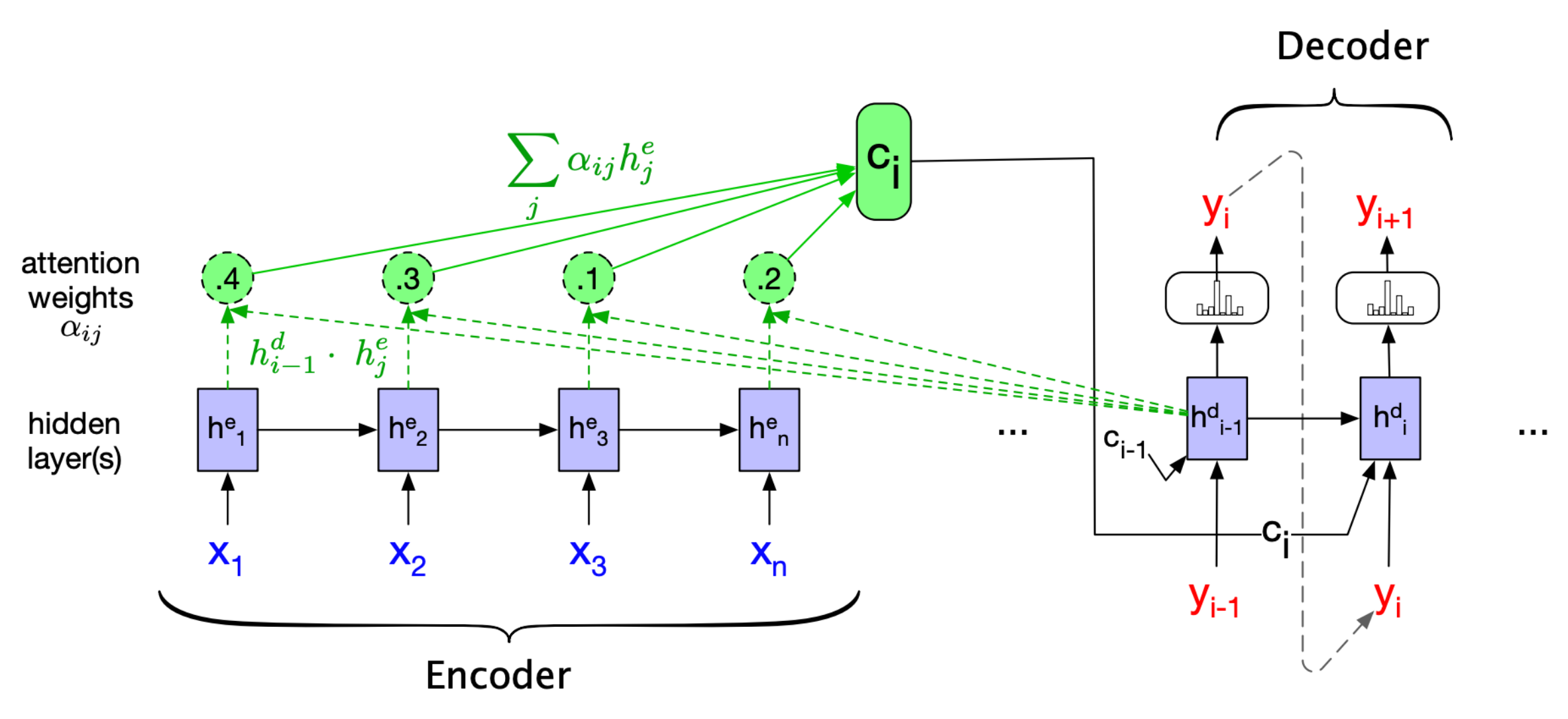
计算所需的条件：

1. 权重向量α_{ij}：用以表示前一个解码器隐藏状态与每一个编码器隐藏状态的相关性
2. 每一个编码器隐藏状态: 提供源文本的上下文信息

$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j^e$$

每一次生成新的解码器隐藏状态时，都会重新计算一次权重向量α_{ij}
从而生成新的上下文向量c

带注意力机制的编解码器模型



另一种注意力计算方法

在前一个解码器隐藏状态h_{i-1}^{d}与每一个解码器隐藏状态$\mathrm{h}_{\mathrm{j}}^{e}$之间添加一个参数矩阵

$$score(\mathbf{h}_{i-1}^{d}, \mathbf{h}_{j}^{e}) = \mathbf{h}_{t-1}^{d} \mathbf{W}_{s} \mathbf{h}_{j}^{e}$$

该矩阵是在正常的端到端训练过程中训练获得

- 可以帮助模型学习并捕获编码器和解码器之间的相关性
- 是一种双线性模型，允许编解码器之间的维度不同

Part.05

波束搜索

上述方法 - 输出预测值

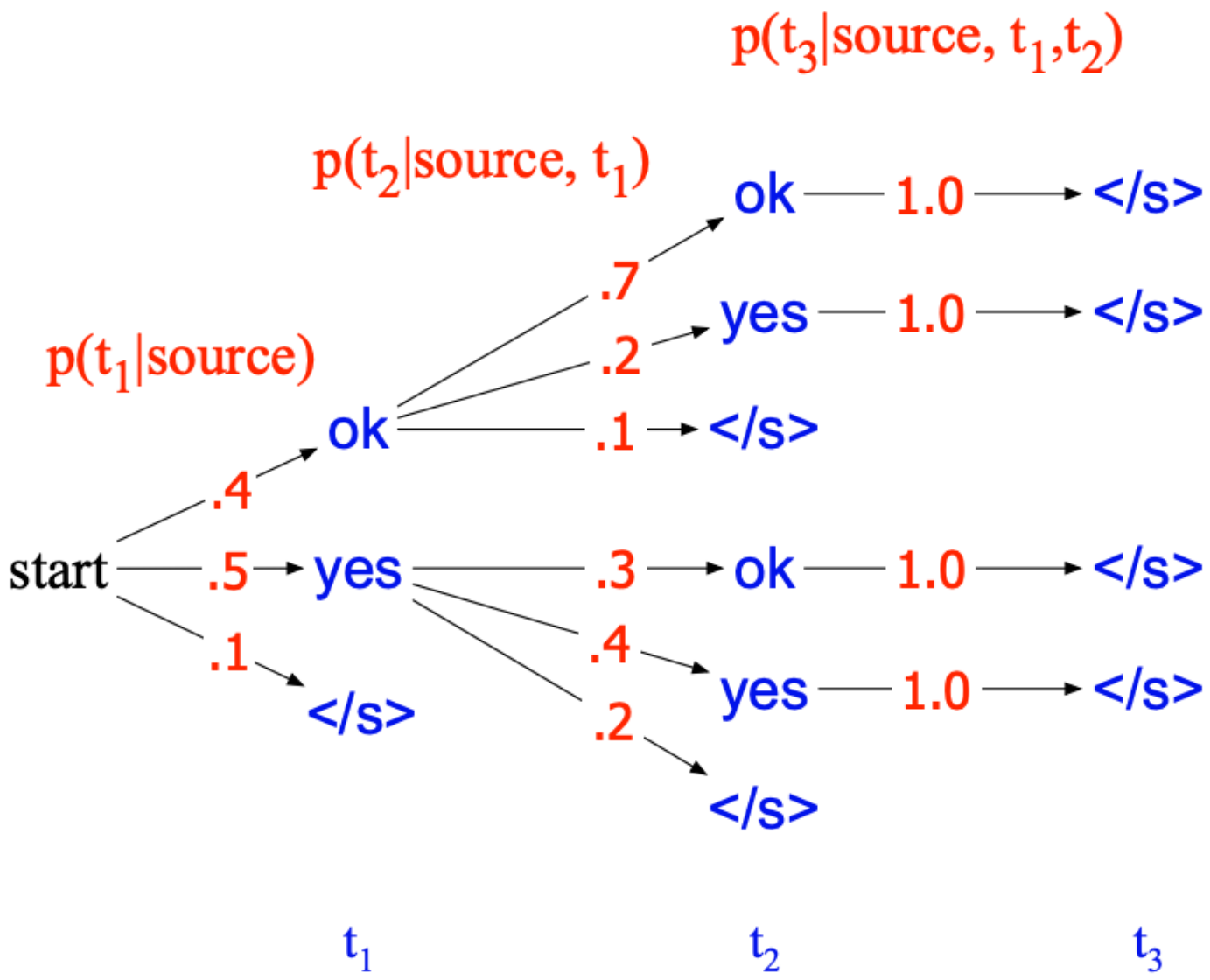
回想一下每一时间步是如何预测当前输出值 $\hat{y}$ 的？

$$\hat{y}_t = \mathrm{argmax}_{w \in V} P(w|x, y_1 \ldots y_{t-1})$$

每一个时间步都选择最高概率，这种算法思想称为贪心算法

贪心算法在应对整句翻译时，是否是全局最优的？

举例 – 贪心算法生成句子


p(t3|source, t1,t2)
p(t2|source, t1)
p(t1|source)
start
.4
.5
.1
ok
yes
</s>
.7
.2
.1
.3
.4
.2
ok
yes
</s>
ok
yes
</s>
1.0
1.0
1.0
1.0
</s>
</s>
</s>
t1
t2
t3

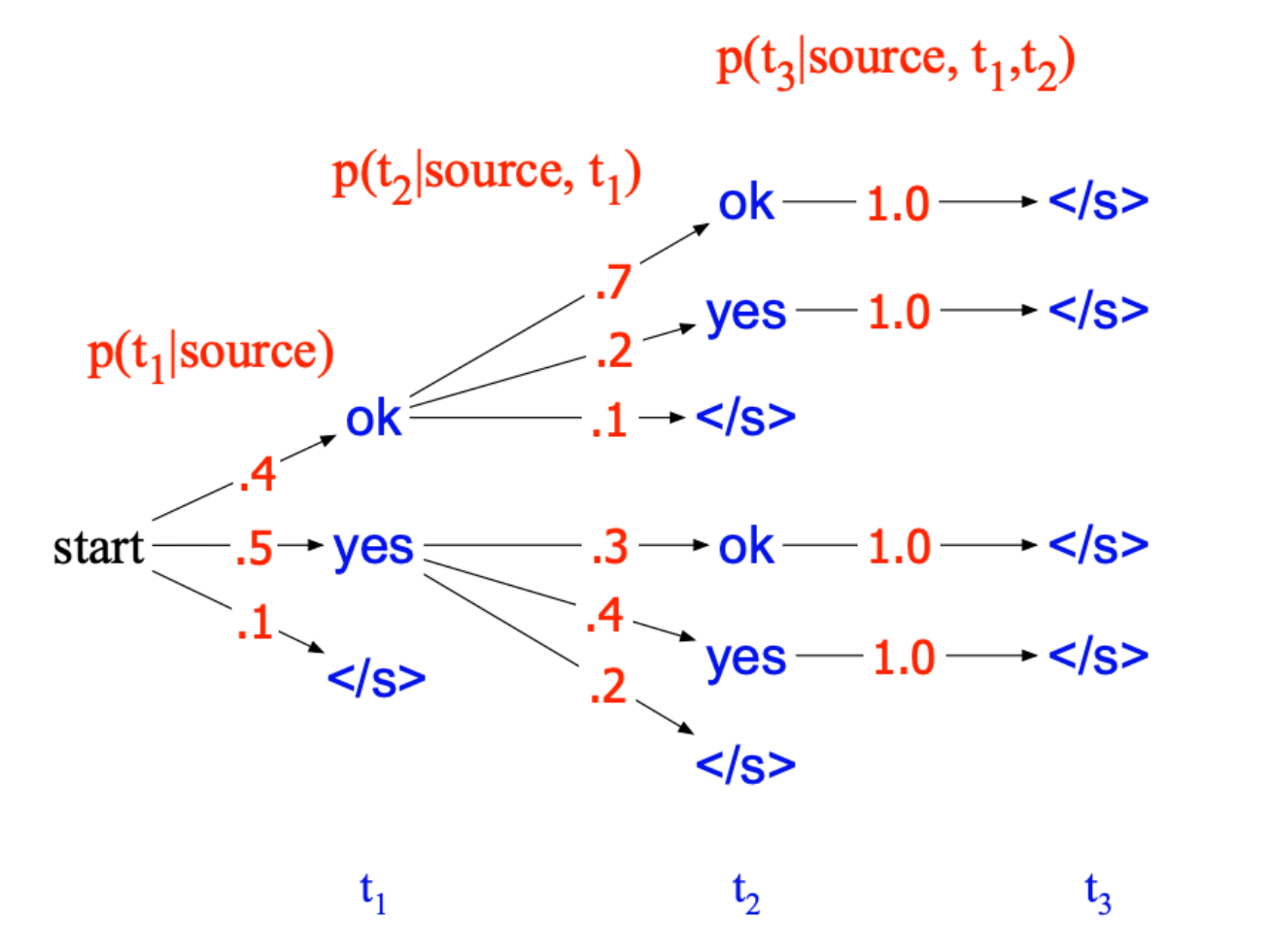
波束搜索

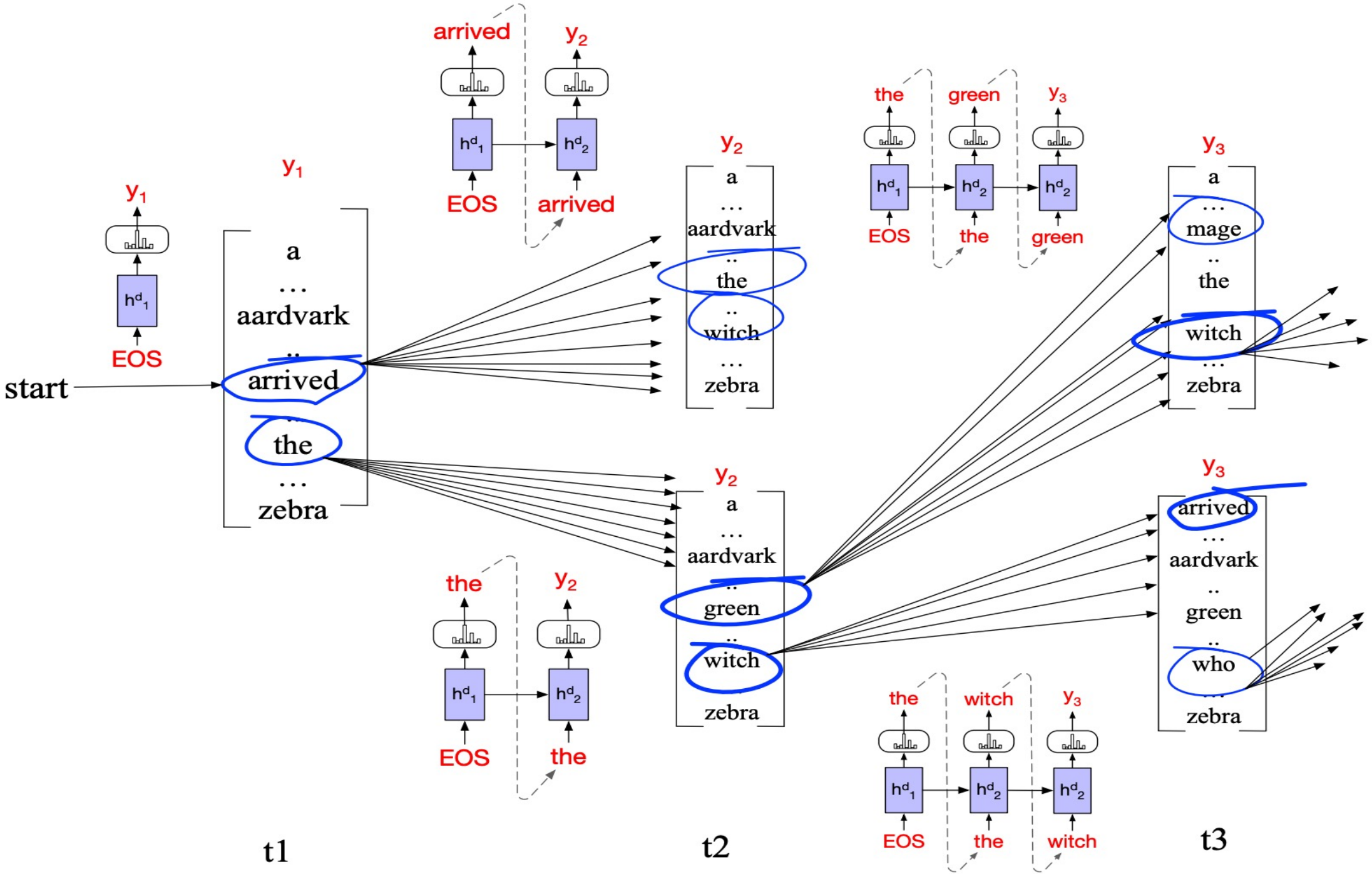
贪心算法 – 选择每一步最优的一个解

波束搜索 – 选择每一步最优的K个解

- K表示的是波束宽度

K = 2时





每一步都会产生

$k * V$

个假设

每个假设

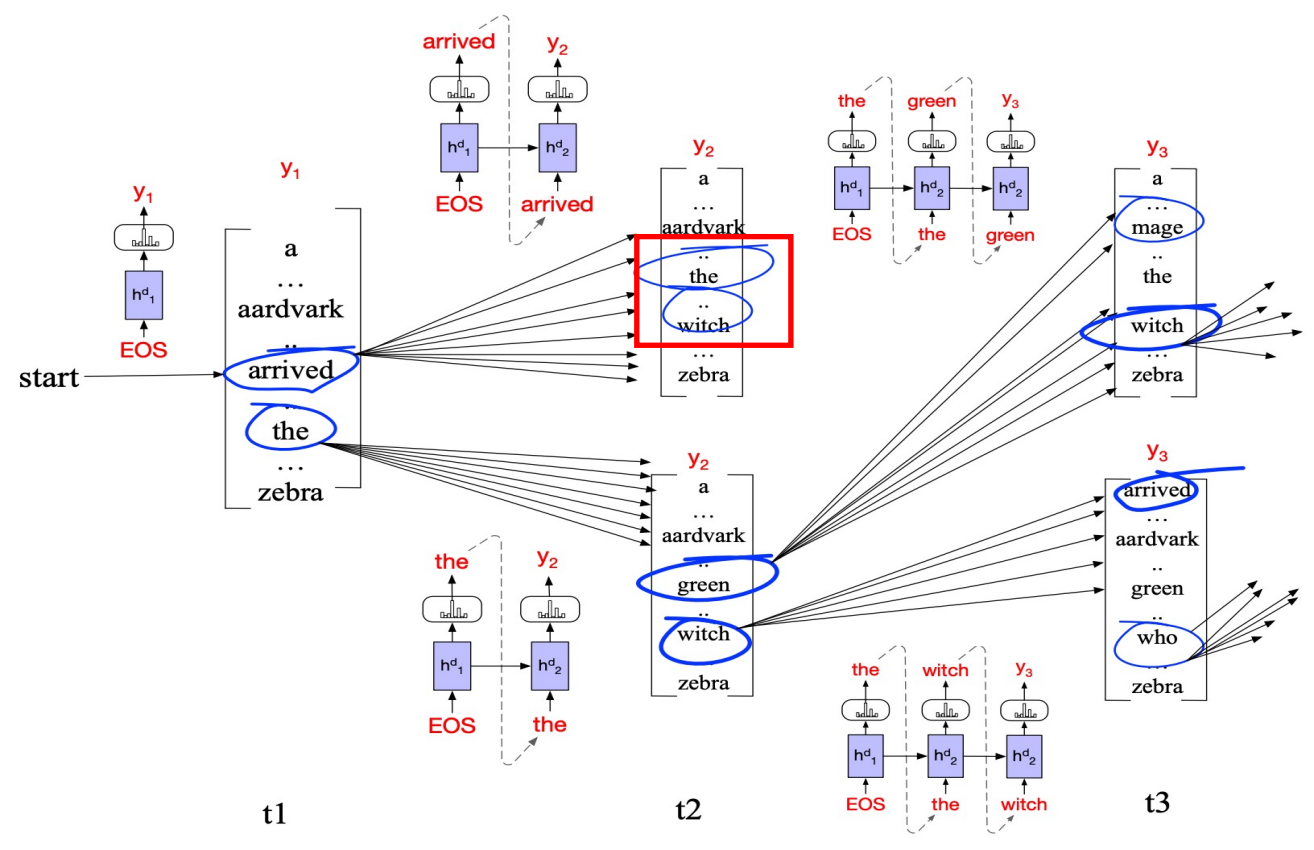
$P(y_i|x, y_{<i})$

由来估计

波束搜索

在获得 $k*V$ 个假设以及对应的概率后，选择概率最高的k个假设。

因此在每一步的搜索边界上，不会超过k个假设，也不会超过k个解码器



波束搜索

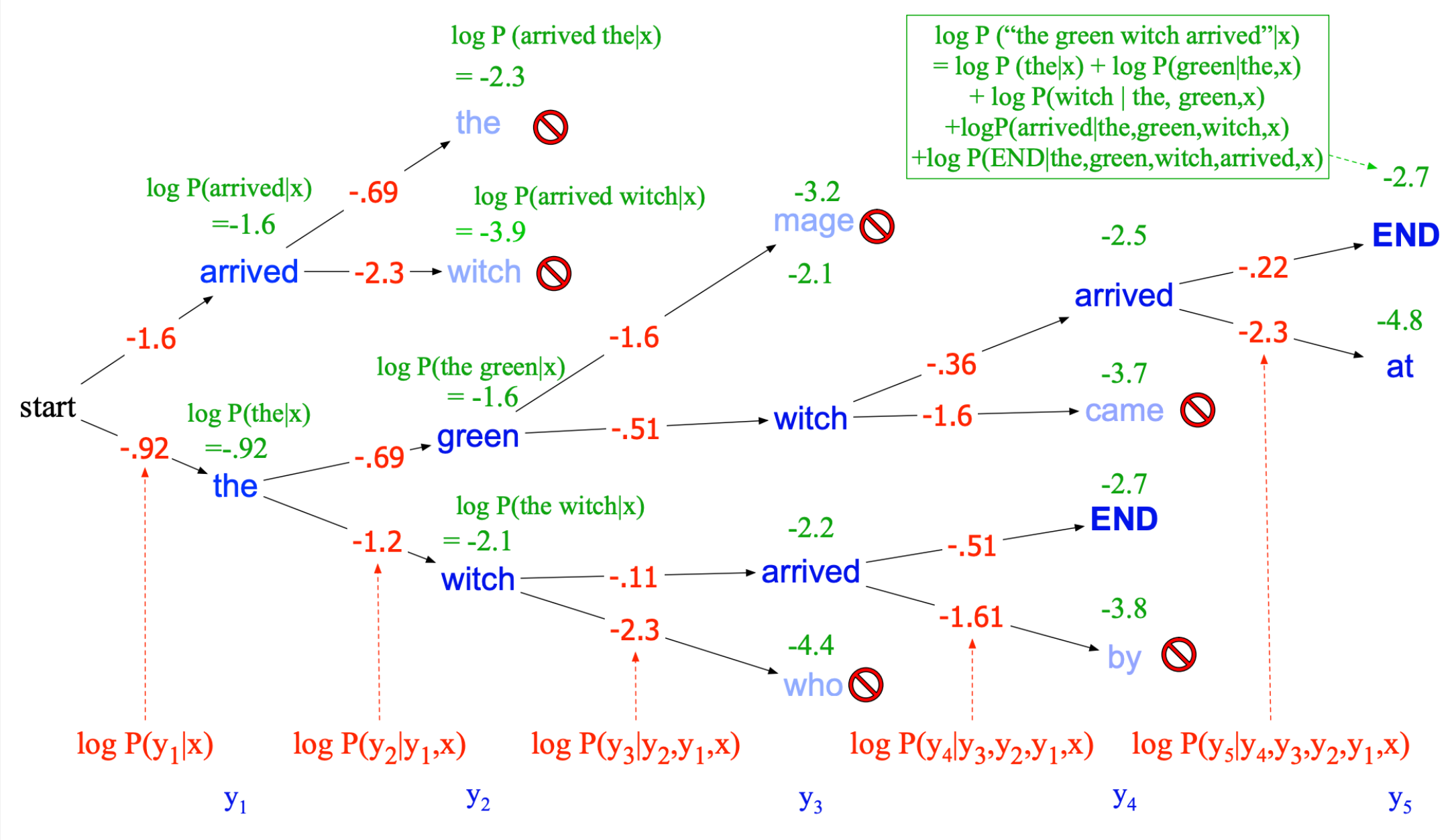
在序列输出的过程中，以结束符</s>为终点，生成一个完整的候选输出

- 此时需要将该完整假设从边界移除
- 波束的大小从k开始不断地减1，直到等于0
  - 此时将有k个假设

每一个假设用以下方法进行评价：

$$
\begin{aligned}
score(y) &= \log P(y|x) \\
&= \log\left(P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)\ldots P(y_t|y_1,\ldots,y_{t-1},x)\right) \\
&= \sum_{i=1}^{t} \log P(y_i|y_1,\ldots,y_{i-1},x)
\end{aligned}
$$

波束搜索


log P (arrived the|x)
= -2.3
the
log P(arrived|x)
=-1.6
-.69
log P(arrived witch|x)
= -3.9
arrived
-2.3
witch
-1.6
start
-.92
log P(the|x)
=-.92
the
-.69
log P(the green|x)
= -1.6
green
-1.6
-3.2
mage
-2.1
-.51
witch
-1.2
log P(the witch|x)
= -2.1
witch
-.11
-2.3
-2.2
arrived
-4.4
who
log P ("the green witch arrived"|x)
= log P (the|x) + log P(green|the,x)
+ log P(witch | the, green,x)
+logP(arrived|the,green,witch,x)
+log P(END|the,green,witch,arrived,x)
-2.7
-2.5
arrived
-.22
END
-2.3
-4.8
at
-.36
-3.7
came
-1.6
-2.7
END
-.51
-1.61
-3.8
by
log P(y1|x)
log P(y2|y1,x)
log P(y3|y2,y1,x)
log P(y4|y3,y2,y1,x)
log P(y5|y4,y3,y2,y1,x)
y1
y2
y3
y4
y5

课后思考题

将该算法的每一代码段进行注释

```
function BEAMDECODE(c, beam_width) returns best paths

  y_0, h_0 ← 0
  path ← ()
  complete_paths ← ()
  state ← (c, y_0, h_0, path)          ;initial state
  frontier ← ⟨state⟩        ;initial frontier

  while frontier contains incomplete paths and beamwidth > 0
      extended_frontier ← ⟨⟩
      for each state ∈ frontier do
          y ← DECODE(state)
          for each word i ∈ Vocabulary do
              successor ← NEWSTATE(state, i, y_i)
              extended_frontier ← ADDTOBEAM(successor, extended_frontier,
                                                                  beam_width)

      for each state in extended_frontier do
          if state is complete do
              complete_paths ← APPEND(complete_paths, state)
              extended_frontier ← REMOVE(extended_frontier, state)
              beam_width ← beam_width - 1
      frontier ← extended_frontier

    return completed_paths

function NEWSTATE(state, word, word_prob) returns new state

function ADDTOBEAM(state, frontier, width) returns updated frontier

  if LENGTH(frontier) < width then
      frontier ← INSERT(state, frontier)
  else if SCORE(state) > SCORE(WORSTOF(frontier))
      frontier ← REMOVE(WORSTOF(frontier))
      frontier ← INSERT(state, frontier)
  return frontier
```

不同假设的长度可能不一

若简单通过计算句子概率来进行择优，那么势必会选择最短的假设作为最终结果。

因此需要针对不等长的候选翻译结果，做长度归一化处理：

$$score(y) = -\log P(y|x) \;=\; \frac{1}{T}\sum_{i=1}^{t} -\log P(y_i|y_1,\ldots,y_{i-1},x)$$

实际应用场景中的波束搜索

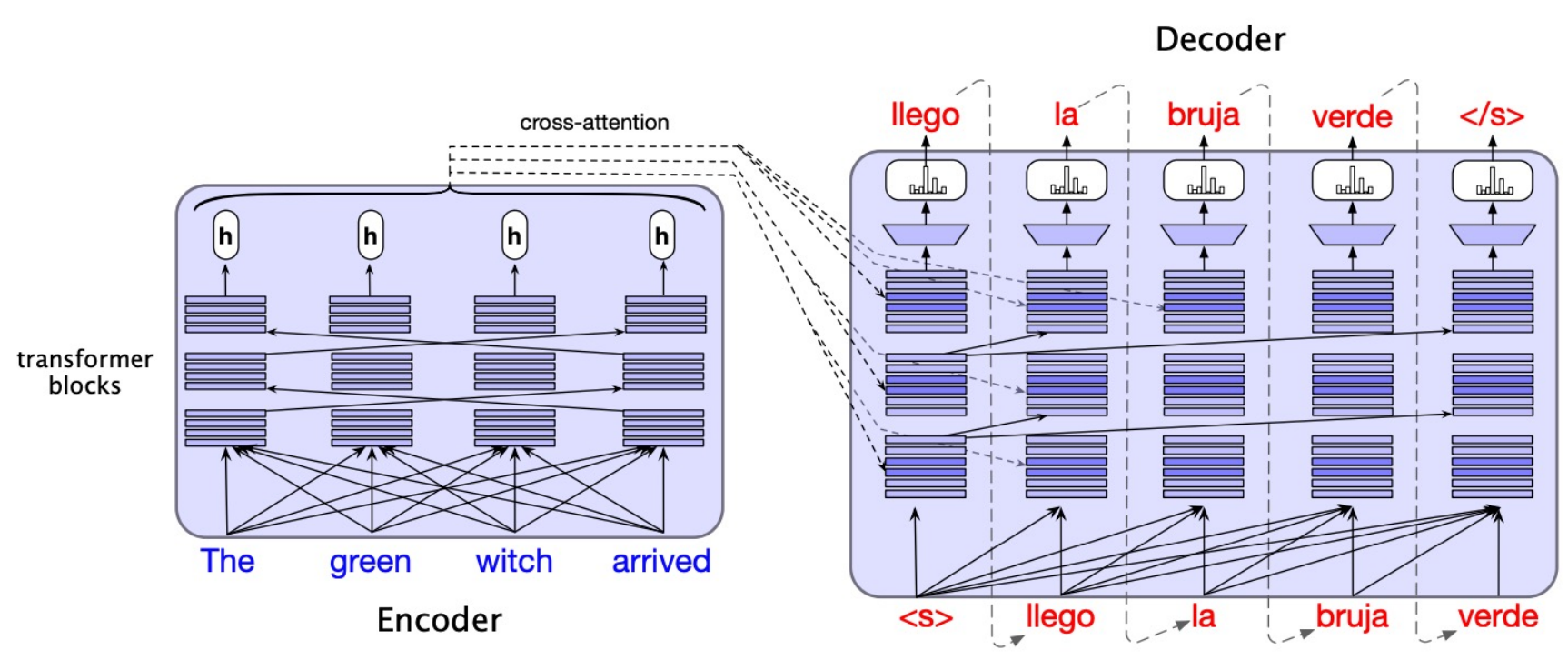
在大型机器翻译系统中，通常宽度k在[5, 10]的区间内取值。

- 可以取最优的假设
- 也可以取所有的假设

Part.06

编解码器与Transformer

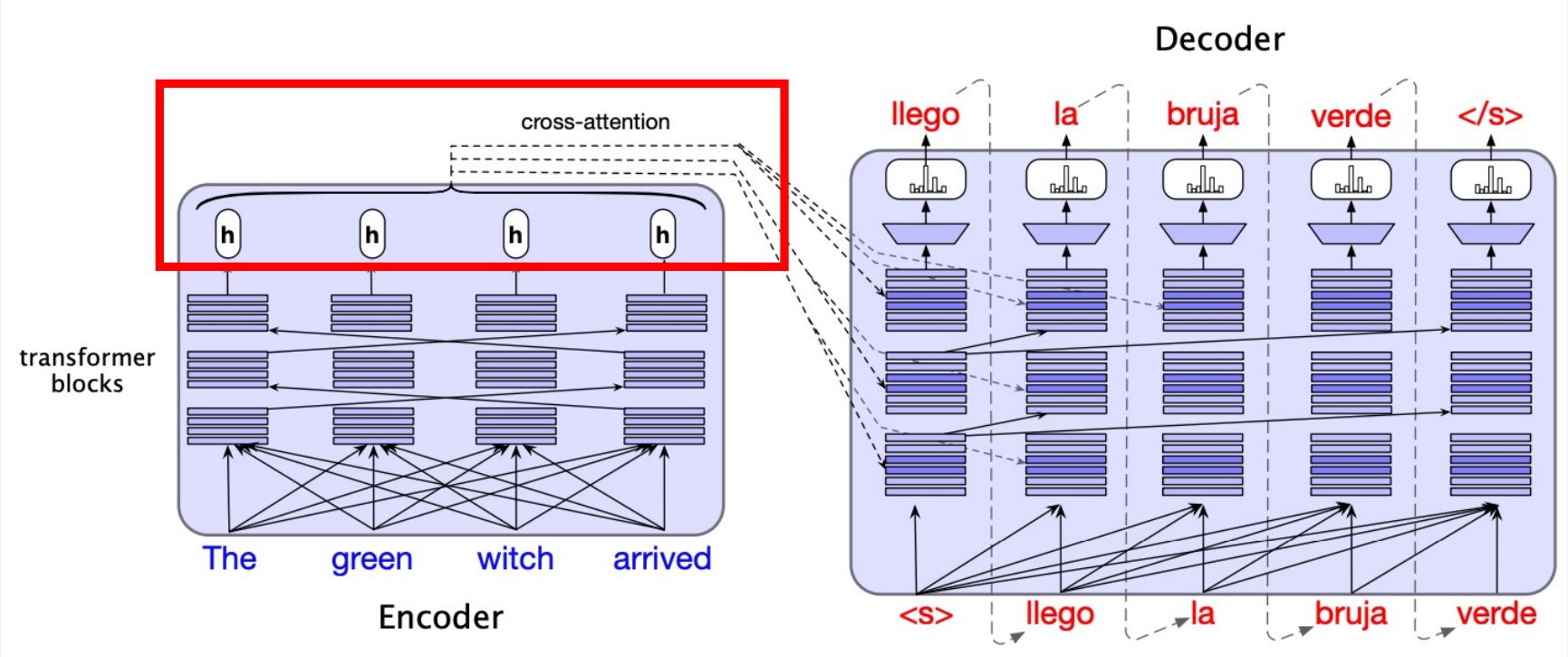
使用Transformer实现编解码器



编码器将输入序列 $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_T$ 映射到一个输出序列表达 $\mathbf{H}^{enc} = \mathbf{h}_1, \ldots, \mathbf{h}_T$

交叉注意力

Cross-attention也称做: encoder-decoder attention或者source attention



交叉注意力

交叉注意力与Transformer中的多头自注意力有相同的组织形式

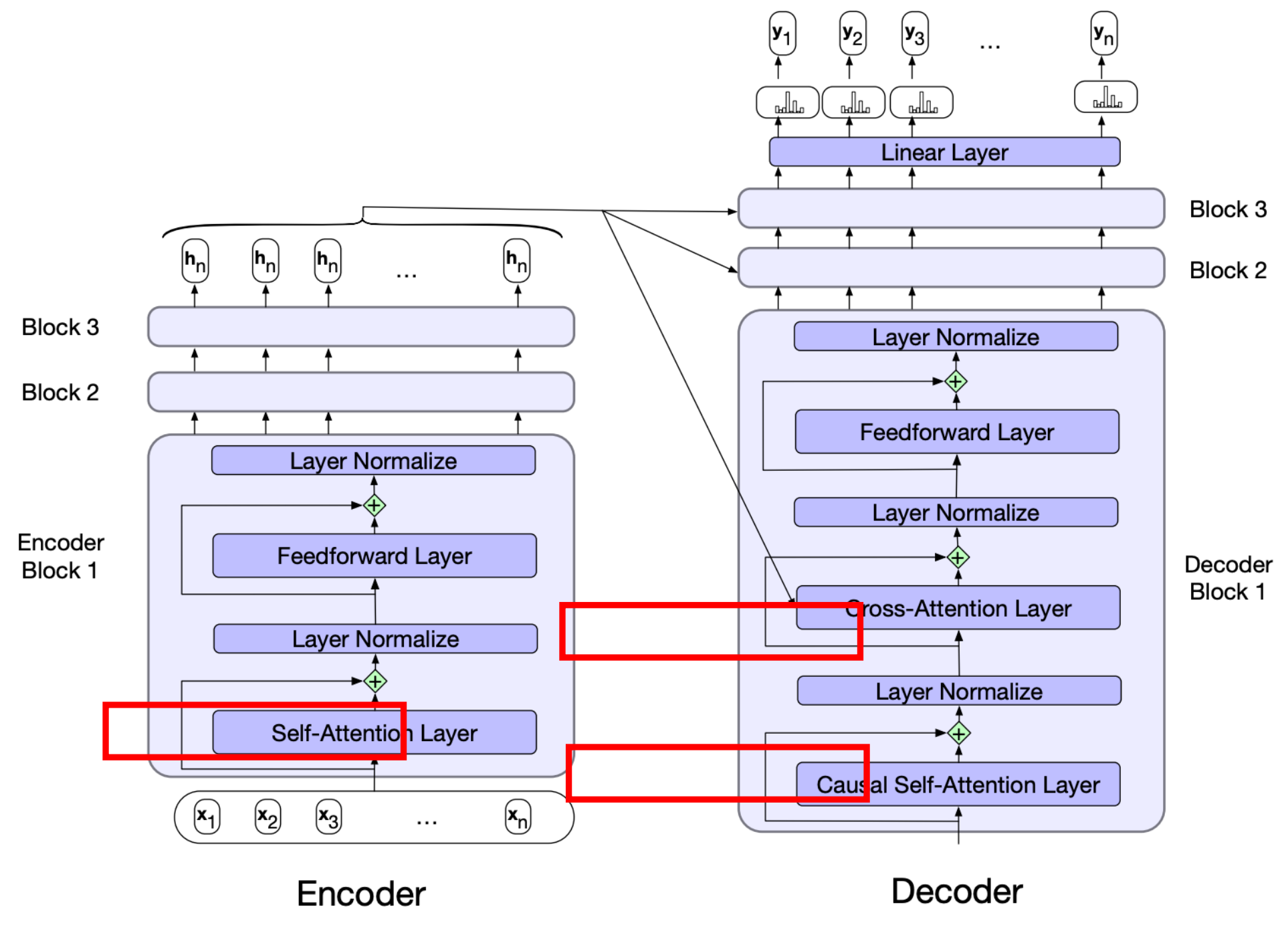
两者之间的唯一不同是，
交叉注意力中的

- Q (查询向量)是从解码器的上一层计算得到的，
- K和V(键向量和值向量)是从编码器的输出计算得到的。

$$\mathbf{Q} = \mathbf{W}^{\mathbf{Q}}\mathbf{H}^{dec[i-1]};\ \ \mathbf{K} = \mathbf{W}^{\mathbf{K}}\mathbf{H}^{enc};\ \ \mathbf{V} = \mathbf{W}^{\mathbf{V}}\mathbf{H}^{enc}$$

$$CrossAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \ = \ \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\intercal}}{\sqrt{d_k}}\right)\mathbf{V}$$

编解码器中的Transformer块



Part.07

构建机器翻译系统

对文本进行tokenization

BPE或wordpiece算法可以对文本数据进行tokenization(BPE在第二章中)

words:	Jet makers feud over seat width with big orders at stake
wordpieces:	_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

Tokenized后的词汇表或语料库包含了全部的源文本和目标文本内容。

wordpiece算法

给定一个训练语料库和一个大小为V的目标词汇表。

1. 用字符初始化wordpiece字典
   - (例如使用unicode字符子集，将所有剩余的字符折叠为一个特殊的未知字符token)

2. 重复直到产生V个wordpiece

   - 基于给定的训练语料库，使用当前的wordpieces集，训练一个n元模型;
   - 考虑到当前词汇表中的两个wordpiece连接后可能产生一组新的wordpiece，选择一个新的Wordpiece可以使得最大程度的增加训练集上的语言模型概率。

词汇表的大小通常为8K-32K。

机器翻译语料库

机器翻译模型通常使用平行语料库来训练，
即这类语料库中的每一段文本数据都有多种语言的表达

官方平行语料库：

欧洲语料库 - "Europarl: A Parallel Corpus for Statistical Machine Translation"
联合国平行预料库- The United Nations Parallel Corpus

其他平行语料库：
OpenSubtitles – 电影电视的子标题
ParaCrawl – 一般的web文本，包含223M句对。

句对齐

给定两个翻译一一对应的文档：

- 一个成本函数，获取源文档和目标文档的跨度
  - 返回一个分数，用以衡量这两个跨度互为翻译的可能性有多大

- 一个对齐算法，使用这些分数寻找良好对齐的文档。

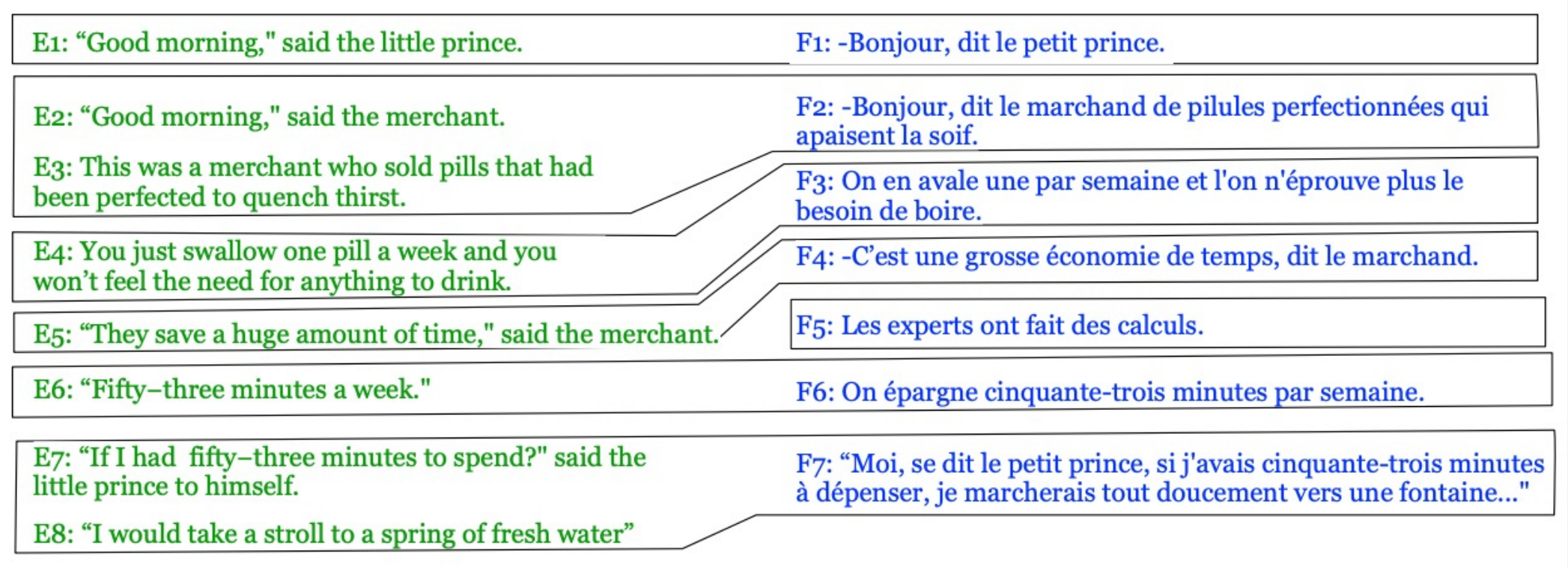
用以衡量两个跨度的损失函数：

$$c(x,y) = \frac{(1-\cos(x,y))\mathrm{nSents}(x)\ \mathrm{nSents}(y)}{\sum_{s=1}^{S} 1-\cos(x,y_s) + \sum_{s=1}^{S} 1-\cos(x_s,y)}$$

句对齐

清理语料库，去除噪声句对：

- 删除太长、太短、冗余、有不同URL的句对；
- 多语言嵌入余弦值的评分过低。



反向翻译

假设：有一个大型单语A预料库，和小型双语AB平行语料库

目标：建立基于大型单语A的，大型双语AB平行语料库

方法：

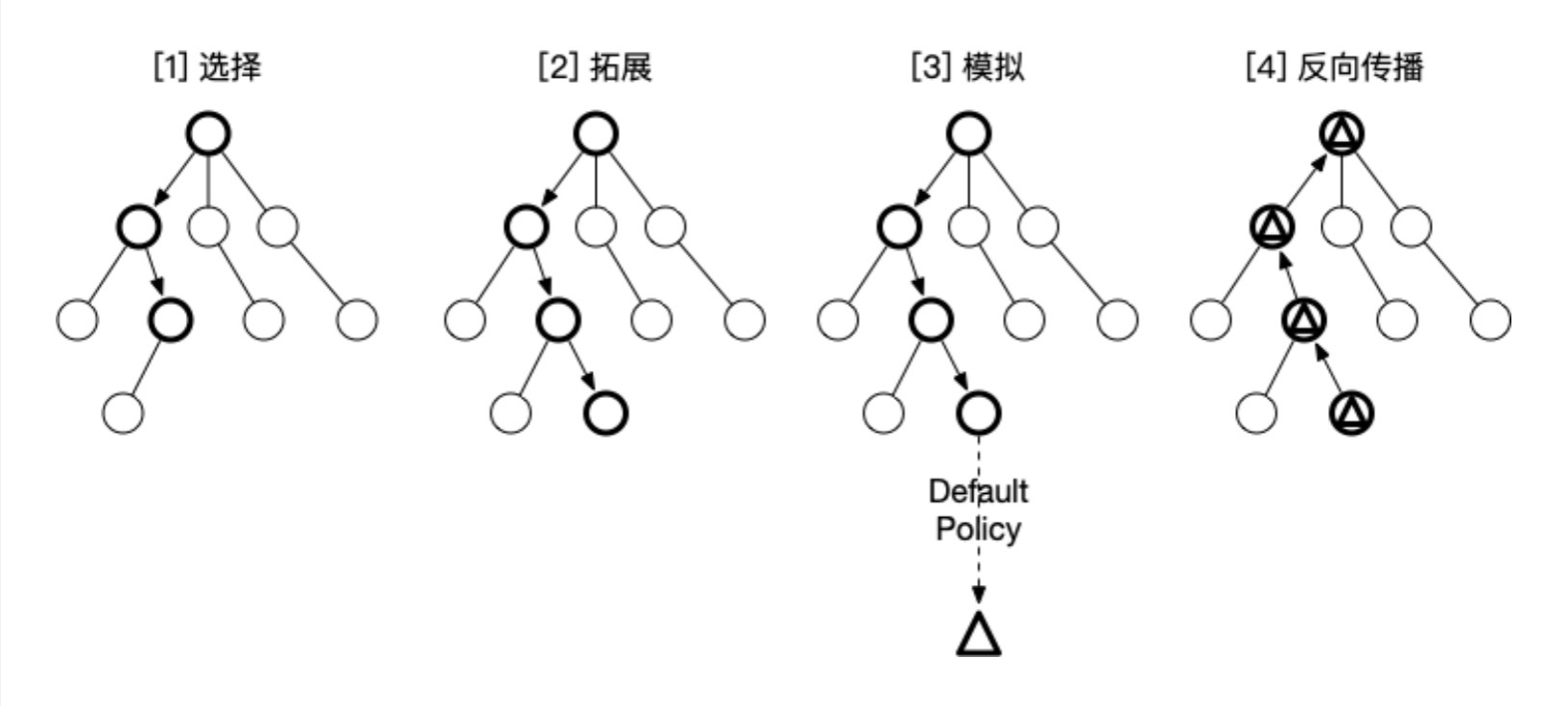
1. 使用小样本平行预料库训练一个机器翻译系统，
2. 再使用训练好的翻译系统，将大型单语语料库翻译为另一种语言
3. 最后将两种语言合成为大型平行语料库。

需要各种参数，包括：

- 如何生成反向翻译的数据(贪心、波束搜索、采样、或蒙特卡罗搜索)
- 反向翻译的双语数据和自然的双语数据之间的比例

蒙特卡罗搜索

原理介绍



Part.08

机器翻译评估

评价标准

充分性 adequacy

流利性 fluency

自动评估：chrF指标

chrF指标： character F-score

给出一个函数：
该函数可以根据机器翻译的目标句与人工翻译目标句之间的N元字符的重叠计数对MT目标句进行打分

假设chrP和参考chrR

chrP – 在假设中的一元到K元字符统计，并出现在参考中的平均比例

chrR – 在参考中的一元到K元字符统计，并出现在假设中的平均比例

F-score 计算：

$$\mathrm{chrF}\beta = (1+\beta^2)\frac{\mathrm{chrP}\cdot\mathrm{chrR}}{\beta^2\cdot\mathrm{chrP}+\mathrm{chrR}}$$

$$\beta = 2$$

$$\mathrm{chrF2} = \frac{5\cdot\mathrm{chrP}\cdot\mathrm{chrR}}{4\cdot\mathrm{chrP}+\mathrm{chrR}}$$

评估用例：假设chrP和参考chrR

一元二元统计
F2-score评估

$k = \beta = 2$

REF: `witness for the past,`
HYP1: `witness of the past,` chrF2,2 = .86
HYP2: `past witness` chrF2,2 = .62

REF – 参考
HYP1 – 假设1

REF: `witnessforthepast,` (18 unigrams, 17 bigrams)
HYP1: `witnessofthepast,` (17 unigrams, 16 bigrams)

评估用例：假设chrP和参考chrR

一元二元统计

unigrams that match: `w i t n e s s f o t h e p a s t ,` (17 unigrams)
bigrams that match: `wi it tn ne es ss th he ep pa as st t,` (13 bigrams)

一元二元统计下的
精度和召回率计算

unigram P:	17/17 = 1	unigram R:	17/18 = .944
bigram P:	13/16 = .813	bigram R:	13/17 = .765

F2-score计算

$$\begin{aligned}
\text{chrP} &= (17/17 + 13/16)/2 = .906 \\
\text{chrR} &= (17/18 + 13/17)/2 = .855 \\
\text{chrF2,2} &= 5\frac{\text{chrP} * \text{chrR}}{4\text{chrP} + \text{chrR}} = .86
\end{aligned}$$

chrF评估的局限性

长短语的移动变化不会造成chrF分数的改变

- (除非K值大于长短语的长度)

chrF无法评估文档的跨句属性，比如说上下文的话语连贯性

chrF不适合人工翻译和机器翻译对比，也不适合不同MT系统性能对比

- chrF更适合在单系统变化下的性能评估

自动评估: 基于嵌入的方法

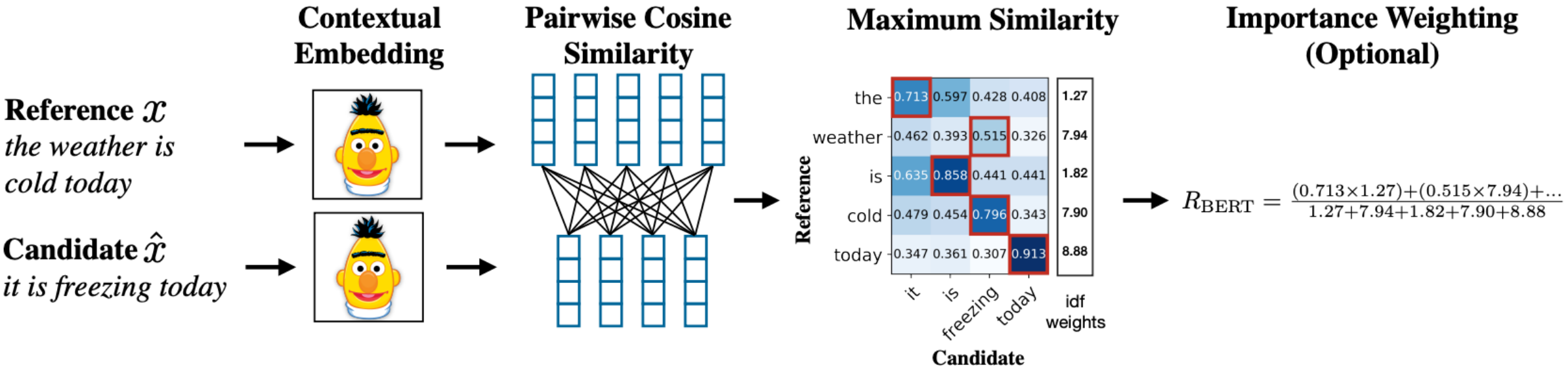
chrF局限于字符，和N元语法语言模型类型，
当遇到同义词替换时，评估效果不会很好

x 参考序列

$\tilde{x}$ 候选序列

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\tilde{x}_j \in \tilde{x}} x_i \cdot \tilde{x}_j \qquad P_{\text{BERT}} = \frac{1}{|\tilde{x}|} \sum_{\tilde{x}_j \in \tilde{x}} \max_{x_i \in x} x_i \cdot \tilde{x}_j$$

BERT中基于嵌入的自动评估方法



总结

- 语言多样性 – 形态学
- 编码器-解码器模型
- RNN编解码器的注意力机制和Transformer编解码器中的交叉注意力机制
- 贪心搜索和波束搜索
- 现有平行语料库，反向翻译构建平行语料库
- MT评估方法 – 考虑充分性和流利性
  - 人工
  - 自动 - 基于字符和基于嵌入