

基于短语和句法的统计翻译模型

肖桐 朱靖波

xiaotong@mail.neu.edu.cn

zhujingbo@mail.neu.edu.cn

东北大学 自然语言处理实验室

<http://www.nlplab.com>



基于单词有哪些问题

- 基于单词的翻译模型有哪些优点？
 - ▶ 比较符合人的思维、简单直接、易于实现

单词翻译表	P
我 → I	0.6
喜欢 → like	0.3
绿 → green	0.9
茶 → tea	0.8

$s =$ 我 喜欢 绿 茶

基于单词有哪些问题

- 基于单词的翻译模型有哪些优点？
 - ▶ 比较符合人的思维、简单直接、易于实现

单词翻译表	P
我 → I	0.6
喜欢 → like	0.3
绿 → green	0.9
茶 → tea	0.8

s = 我 喜欢 绿 茶
 | / / |
t = I like green tea

基于单词有哪些问题

- 基于单词的翻译模型有哪些**优点**？
 - 比较符合人的思维、简单直接、易于实现

单词翻译表	P
我 → I	0.6
喜欢 → like	0.3
绿 → green	0.9
茶 → tea	0.8

s = 我 喜欢 绿 茶
 | / / |
t = I like green tea

- 基于单词的翻译模型有哪些**不足**？
 - 需要定义词是什么
 - 独立性假设：单词之间相对独立，没有考虑搭配
 - 调序：较弱的调序建模
 - ...

单词翻译表	P
我 → I	0.6
喜欢 → like	0.3
红 → red	0.8
红 → black	0.1
茶 → tea	0.8

s = 我 喜欢 红 茶

基于单词有哪些问题

- 基于单词的翻译模型有哪些**优点**？
 - 比较符合人的思维、简单直接、易于实现

单词翻译表	P					
我 → I	0.6	s =	我	喜欢	绿	茶
喜欢 → like	0.3			↗	↗	
绿 → green	0.9	t =	I	like	green	tea
茶 → tea	0.8					

- 基于单词的翻译模型有哪些**不足**？
 - 需要定义词是什么
 - 独立性假设：单词之间相对独立，没有考虑搭配
 - 调序：较弱的调序建模
 - ...

单词翻译表	P					
我 → I	0.6	s =	我	喜欢	红	茶
喜欢 → like	0.3			↗	↗	↗
红 → red	0.8	t =	I	like	red	tea
红 → black	0.1					
茶 → tea	0.8					

基于单词有哪些问题

- 基于单词的翻译模型有哪些优点？
 - 比较符合人的思维、简单直接、易于实现

单词翻译表	P					
我 → I	0.6	s =	我	喜欢	绿	茶
喜欢 → like	0.3			↗	↗	
绿 → green	0.9	t =	I	like	green	tea
茶 → tea	0.8					

- 基于单词的翻译模型有哪些不足？
 - 需要定义词是什么
 - 独立性假设：单词之间相对独立，没有考虑搭配
 - 调序：较弱的调序建模
 - ...

单词翻译表	P					
我 → I	0.6	s =	我	喜欢	红	茶
喜欢 → like	0.3			↗	↗	↗
红 → red	0.8	t =	I	like	red	tea
红 → black	0.1					
茶 → tea	0.8					

"红 茶"为一种搭配，
翻译为"black tea"，

"红 茶" 为一种搭配，应该翻译为 "black tea"

引入更大的翻译单元

- 简单的单词翻译似乎不行？

单词翻译表	P					
我 → I	0.6	s =	我	喜欢	红	茶
喜欢 → like	0.3					
红 → red	0.8	t =	I	like	red	tea
红 → black	0.1					
茶 → tea	0.8					



引入更大的翻译单元

- 简单的单词翻译似乎不行？ - 引入更大的翻译单元

单词词串翻译表	P
我 → I	0.6
喜欢 → like	0.3
红 → red	0.8
红 → black	0.1
茶 → tea	0.8
我喜欢 → I like	0.3
我喜欢 → I liked	0.2
绿茶 → green tea	0.5
绿茶 → the green tea	0.1
红茶 → black tea	0.6
...	

s = 我 喜欢 红 茶
 | / / /
t = I like red tea

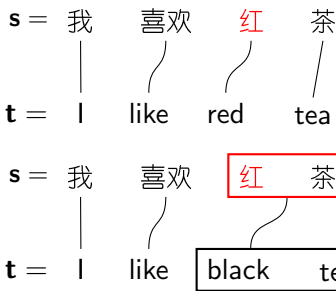


引入更大的翻译单元

- 简单的单词翻译似乎不行？ - 引入更大的翻译单元

单词词串翻译表

我 → I	0.6
喜欢 → like	0.3
红 → red	0.8
红 → black	0.1
茶 → tea	0.8
我喜欢 → I like	0.3
我喜欢 → I liked	0.2
绿茶 → green tea	0.5
绿茶 → the green tea	0.1
红茶 → black tea	0.6
...	



No

Yes

引入更大的翻译单元

- 简单的单词翻译似乎不行？ - 引入更大的翻译单元

单词词串翻译表	P					
我 → I	0.6	s =	我	喜欢	红	茶
喜欢 → like	0.3					
红 → red	0.8					
红 → black	0.1					
茶 → tea	0.8	t =	I	like	red	tea
我喜欢 → I like	0.3					
我喜欢 → I liked	0.2	s =	我	喜欢	红	茶
绿茶 → green tea	0.5					
绿茶 → the green tea	0.1					
红茶 → black tea	0.6	t =	I	like	black	tea
...						

No

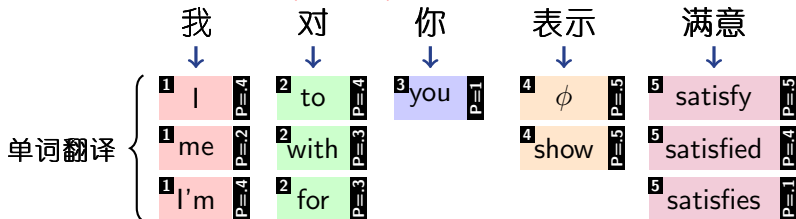
Yes

- 优点：
 - 翻译时候可以考虑更大范围的上下文信息
比如：“红茶”中的“红”如果和“茶”搭配 ...
 - 更好的局部调序，比如：短语中有“的”字 → ... of ... 结构
 - 更大范围的目标语连续词串的使用，有利于 n -gram 语言模型选择译文

回顾 - 基于词的机器翻译

- 对每个单词的翻译进行（任意）组合

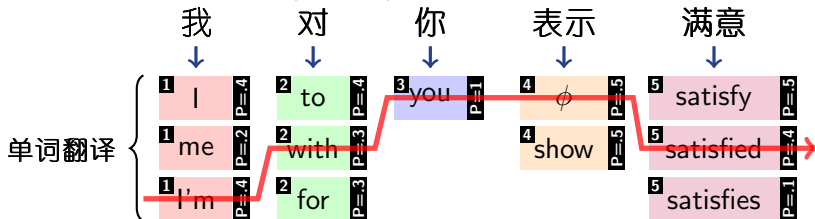
待翻译句子(已经分词):



回顾 - 基于词的机器翻译

- 对每个单词的翻译进行（任意）组合

待翻译句子(已经分词):

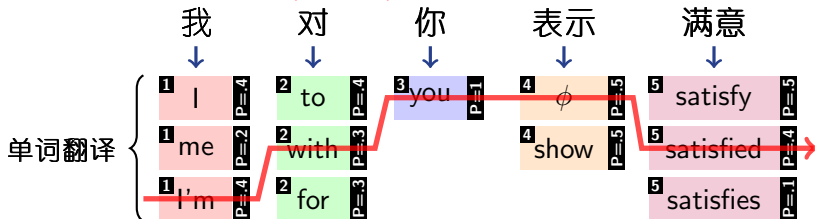


翻译路径（仅含有单词）：→

基于“短语”的机器翻译

- 对每个单词及连续词串的翻译进行（任意）组合

待翻译句子(已经分词):

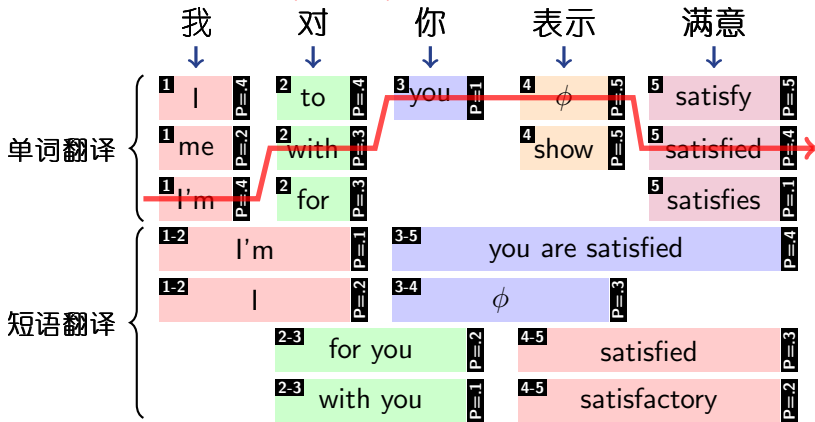


翻译路径（仅含有单词）：→

基于“短语”的机器翻译

- 对每个单词及连续词串的翻译进行（任意）组合

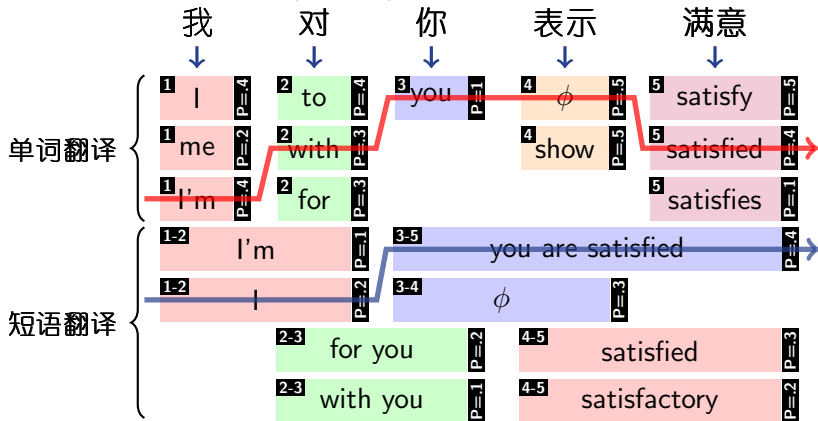
待翻译句子(已经分词):



基于“短语”的机器翻译

- 对每个单词及连续词串的翻译进行（任意）组合

待翻译句子(已经分词):



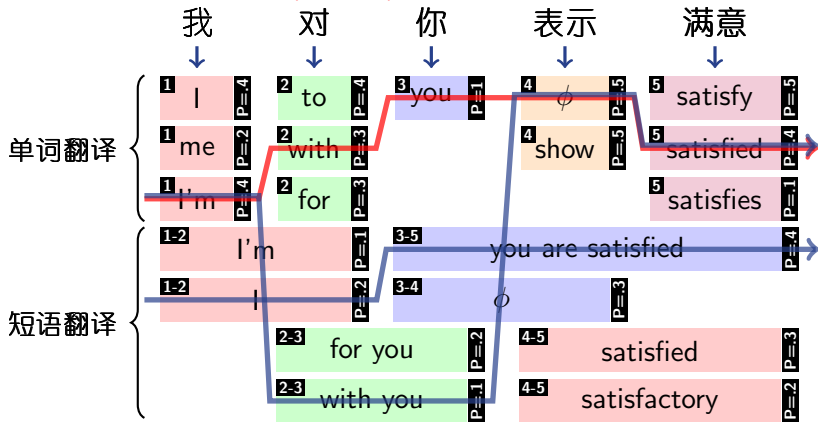
翻译路径 (仅含单词): \rightarrow

翻译路径 (含有短语): \rightarrow

基于“短语”的机器翻译

- 对每个单词及连续词串的翻译进行（任意）组合

待翻译句子(已经分词):

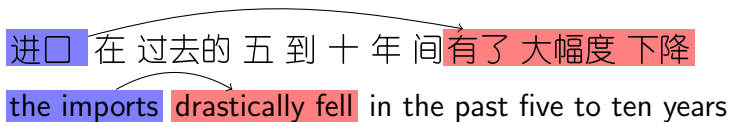


翻译路径 (仅含单词): →

翻译路径 (含有短语): →

使用短语就够了？

- 短语是具有完整意思的连续词串，因此可以捕捉更多的上下文信息
 - ▶ 不过过大的短语会造成数据稀疏、长距离依赖等问题
 - ▶ 而且单纯的词串也缺乏句法功能表示能力



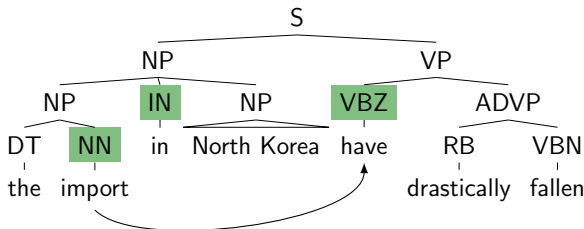
使用短语就够了？

- 短语是具有完整意思的连续词串，因此可以捕捉更多的上下文信息
 - 不过过大的短语会造成数据稀疏、长距离依赖等问题
 - 而且单纯的词串也缺乏句法功能表示能力

进口 在过去的五到十年间 有了大幅度下降

the imports drastically fell in the past five to ten years

- 另一种方式是考虑句子的句法结构，这样更容易描述句子的层次结构和长距离依赖关系



引入句法信息

- 句法树是句子的更高层次的抽象，相比短语句法树具有更加丰富的句法功能标记，对语言结构的转换很有帮助
 - ▶ 更容易捕捉翻译中的远距离调序
 - ▶ 使用句法更容易对大范围的上下文建模

引入句法信息

- 句法树是句子的更高层次的抽象，相比短语句法树具有更加丰富的句法功能标记，对语言结构的转换很有帮助
 - ▶ 更容易捕捉翻译中的远距离调序
 - ▶ 使用句法更容易对大范围的上下文建模
- 看一个真实的例子
 - ▶ 长介词短语的翻译，需要完整的看到这个结构才能准确翻译介词

人工翻译: After North Korea demanded concessions from U.S. again before the start of a new round of six-nation talks ...

机器翻译: In the new round of six-nation talks on North Korea again demanded that U.S. in the former promise ...

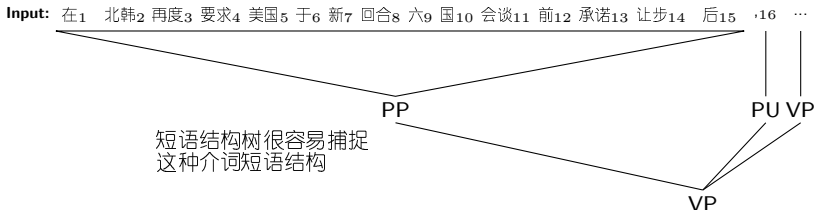
Input: 在₁ 北韩₂ 再度₃ 要求₄ 美国₅ 于₆ 新₇ 回合₈ 六₉ 国₁₀ 会谈₁₁ 前₁₂ 承诺₁₃ 让步₁₄ 后₁₅ ,₁₆ ...

引入句法信息

- 句法树是句子的更高层次的抽象，相比短语句法树具有更加丰富的句法功能标记，对语言结构的转换很有帮助
 - 更容易捕捉翻译中的远距离调序
 - 使用句法更容易对大范围的上下文建模
- 看一个真实的例子
 - 长介词短语的翻译，需要完整的看到这个结构才能准确翻译介词

人工翻译: After North Korea demanded concessions from U.S. again before the start of a new round of six-nation talks ...

机器翻译: In the new round of six-nation talks on North Korea again demanded that U.S. in the former promise ...



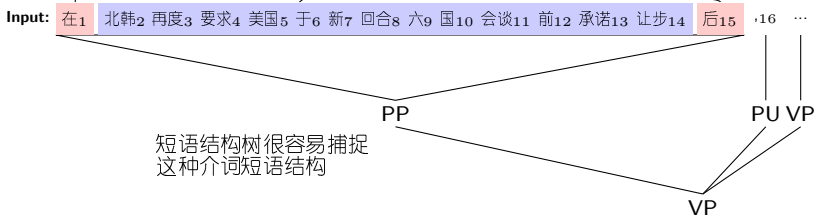
引入句法信息

- 句法树是句子的更高层次的抽象，相比短语句法树具有更加丰富的句法功能标记，对语言结构的转换很有帮助
 - 更容易捕捉翻译中的远距离调序
 - 使用句法更容易对大范围的上下文建模
- 看一个真实的例子
 - 长介词短语的翻译，需要完整的看到这个结构才能准确翻译介词

人工翻译: After North Korea demanded concessions from U.S. again before the start of a new round of six-nation talks ...

机器翻译: In the new round of six-nation talks on North Korea again demanded that U.S. in the former promise ...

better?: After North Korea again demanded that U.S. promised concessions before the new round of six-nation talks ...



如何使用短语、句法等
结构信息进行机器翻译建模？

Outline

基于短语的模型

1. 建模
2. 短语获取和调序
3. 翻译特征和最小错误率训练
4. 栈解码

基于层次短语的模型

1. 同步上下文无关文法
2. 层次短语规则及翻译特征
3. 基于chart的解码和剪枝

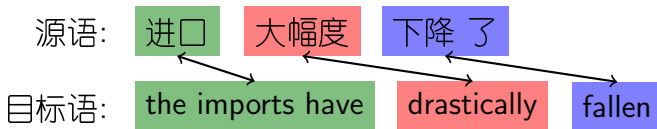
基于语言学句法的模型

1. 基于树结构的文法
2. 翻译规则抽取
3. 规则匹配

何为短语？

- 句对可以用短语对的组合进行表示，比如下图的例子包含三个短语翻译：

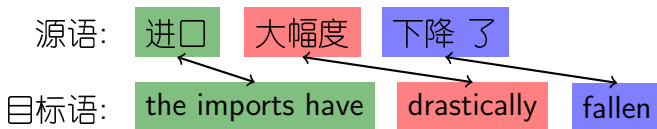
- ▶ 进口 ↔ the imports have
- ▶ 大幅度 ↔ drastically
- ▶ 下降了 ↔ fallen



何为短语？

- 句对可以用短语对的组合进行表示，比如下图的例子包含三个短语翻译：

- ▶ 进口 \leftrightarrow the imports have
- ▶ 大幅度 \leftrightarrow drastically
- ▶ 下降了 \leftrightarrow fallen



- 显然上图中的短语并不是语言学上的短语。这里有：

定义 - 短语

对于一个句子 $\mathbf{w} = w_1 \dots w_n$ ，任意子串 $w_i \dots w_j (i \leq j, 0 \leq i, j \leq n)$ 都是句子 \mathbf{w} 的一个短语

- ▶ n 个词构成的句子可以有 $\frac{n(n+1)}{2}$ 个短语

双语短语

- 进一步，可以定义

定义 - 句子的短语切分

对于一个句子 $\mathbf{w} = w_1 \dots w_n$ ，可以被切分为 m 个子串，则称 \mathbf{w} 由 m 个短语组成，记为 $\mathbf{w} = p_1 \dots p_m$ ，其中 p_i 是 \mathbf{w} 的一个短语， $p_1 \dots p_m$ 也被称作句子 \mathbf{w} 的一个**短语切分**

双语短语

- 进一步，可以定义

定义 - 句子的短语切分

对于一个句子 $\mathbf{w} = w_1 \dots w_n$ ，可以被切分为 m 个子串，则称 \mathbf{w} 由 m 个短语组成，记为 $\mathbf{w} = p_1 \dots p_m$ ，其中 p_i 是 \mathbf{w} 的一个短语， $p_1 \dots p_m$ 也被称作句子 \mathbf{w} 的一个**短语切分**

- 对于双语的情况

定义 - 双语短语(或短语对)

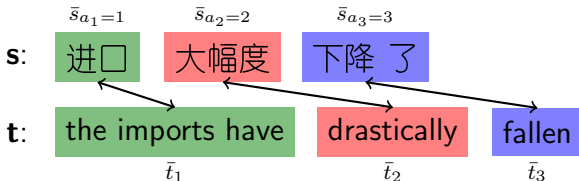
对于源语和目标语句对 (\mathbf{s}, \mathbf{t}) ， \mathbf{s} 中短语 \bar{s}_i 和 \mathbf{t} 中的短语 \bar{t}_j 可以构成一个双语短语对 (\bar{s}_i, \bar{t}_j) ，简称**短语对** (\bar{s}_i, \bar{t}_j)

- ▶ 比如，句对“进口 大幅度 下降 了 \leftrightarrow the imports have drastically fallen”，有很多短语对，比如
 - 大幅度 \leftrightarrow drastically
 - 大幅度 下降 \leftrightarrow have drastically fallen

基于短语的翻译推导

定义 - 基于短语的翻译推导

对于源语和目标语句对 (\mathbf{s}, \mathbf{t}) ，分别有短语切分 $\{\bar{s}_i\}$ 和 $\{\bar{t}_j\}$ ，且 $\{\bar{s}_i\}$ 和 $\{\bar{t}_j\}$ 之间存在一一对应的关系。令 $\{\bar{a}_j\}$ 表示 $\{\bar{t}_j\}$ 中每个短语对应到源语言短语的编号，则称短语对 $\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\}$ 构成了 \mathbf{s} 到 \mathbf{t} 的基于短语的翻译推导(简称推导)，记为 $d(\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\}, \mathbf{s}, \mathbf{t})$ (简记为 $d(\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\})$ 或 d)。



- $\{(\bar{s}_{\bar{a}_j}, \bar{t}_j)\}$ 构成了 (\mathbf{s}, \mathbf{t}) 的一个基于短语的翻译推导
- 需要在建模中描述的两个问题：
 - ▶ $\bar{s}_{\bar{a}_j}$ 是如何被翻译成 \bar{t}_j 的?
 - ▶ 翻译的顺序是如何决定的，即如何得到 $\{\bar{a}_j\}$?

数学模型

- **机器翻译**: 对于输入的源语言句子 s , 找到最佳译文 \hat{t}

$$\hat{t} = \arg \max_t P(t|s)$$

其中 $P(t|s)$ 表示 s 到 t 的翻译概率

- 三个基本问题(回忆一下第三章)
 - ① 如何定义 $P(t|s)$ - 建模问题
 - ② 如何学习 $P(t|s)$ 的统计模型 - 训练问题
 - ③ 如何找到最优译文 - 解码问题

数学模型

- **机器翻译**：对于输入的源语言句子 \mathbf{s} ，找到最佳译文 $\hat{\mathbf{t}}$

$$\hat{\mathbf{t}} = \arg \max_{\mathbf{t}} P(\mathbf{t}|\mathbf{s})$$

其中 $P(\mathbf{t}|\mathbf{s})$ 表示 \mathbf{s} 到 \mathbf{t} 的翻译概率

- 三个基本问题(回忆一下第三章)
 - ① 如何定义 $P(\mathbf{t}|\mathbf{s})$ - 建模问题
 - ② 如何学习 $P(\mathbf{t}|\mathbf{s})$ 的统计模型 - 训练问题
 - ③ 如何找到最优译文 - 解码问题
- 先看建模问题。可以把 $P(\mathbf{t}|\mathbf{s})$ 表示成所有翻译推导的概率

$$P(\mathbf{t}|\mathbf{s}) = \sum_d P(d, \mathbf{t}|\mathbf{s})$$

d 是一个 (\mathbf{s}, \mathbf{t}) 上基于短语的翻译推导， $P(d, \mathbf{t}|\mathbf{s})$ 表示翻译推导 d 的概率

数学模型（续）

- 但是，上式提到的翻译推导的样本空间是巨大的，很难枚举所有推导并进行求和。通常使用采样的方法选取搜索空间的一部分样本代表整个搜索空间

$$P(\mathbf{t}|\mathbf{s}) = \sum_d P(d, \mathbf{t}|\mathbf{s})$$

The diagram illustrates the components of the summation formula. Two arrows point from the right towards the summation term $\sum_d P(d, \mathbf{t}|\mathbf{s})$. The top arrow originates from a green box containing the expression $\text{Max } P(d, \mathbf{t}|\mathbf{s})$. The bottom arrow originates from a green box containing the expression $\sum_{d_{nbest}} P(d, \mathbf{t}|\mathbf{s})$. The summation term itself is highlighted with a red background.

如1-best (Viterbi) 或者n-best的和来近似所有的和

数学模型（续）

- 但是，上式提到的翻译推导的样本空间是巨大的，很难枚举所有推导并进行求和。通常使用采样的方法选取搜索空间的一部分样本代表整个搜索空间

$$P(\mathbf{t}|\mathbf{s}) = \sum_d P(d, \mathbf{t}|\mathbf{s})$$

Diagram illustrating the equation $P(\mathbf{t}|\mathbf{s}) = \sum_d P(d, \mathbf{t}|\mathbf{s})$. The sum term $\sum_d P(d, \mathbf{t}|\mathbf{s})$ is highlighted in pink. Two arrows point to it from boxes on the right: the top box (green) contains $\text{Max } P(d, \mathbf{t}|\mathbf{s})$ and the bottom box (green) contains $\sum_{d_{nbest}} P(d, \mathbf{t}|\mathbf{s})$.

如1-best (Viterbi) 或者n-best的和来近似所有的和

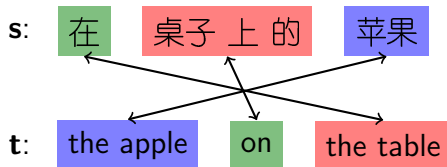
- 若采用Viterbi的方法，机器翻译也可看作对于输入的源语言句子 \mathbf{s} ，找到最佳翻译推导 \hat{d}

$$\hat{d} = \arg \max_d P(d, \mathbf{t}|\mathbf{s})$$

在后面的内容中出现的 \hat{d} 和 \hat{t} 都可以看作是等价的

对翻译推导进行建模

- $P(\mathbf{t}|\mathbf{s}) = \sum_d P(d, \mathbf{t}|\mathbf{s})$ 带来新的问题：如何描述 $P(d, \mathbf{t}|\mathbf{s})$

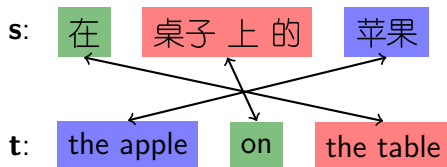


上图体现了三方面问题

- 1 短语获取：确定哪些是“可用”的短语
- 2 翻译模型：描述短语翻译的好坏
- 3 调序模型：描述翻译中的调序现象

对翻译推导进行建模

- $P(\mathbf{t}|\mathbf{s}) = \sum_d P(d, \mathbf{t}|\mathbf{s})$ 带来新的问题：如何描述 $P(d, \mathbf{t}|\mathbf{s})$



上图体现了三方面问题

- ① 短语获取：确定哪些是“可用”的短语
 - ② 翻译模型：描述短语翻译的好坏
 - ③ 调序模型：描述翻译中的调序现象
- 希望有这样一种模型可以对任意的因素进行方便的建模。经典的判别式模型成为了不二的选择

**Discriminative Training and Maximum Entropy
Models for Statistical Machine Translation**

Franz Och and Hermann Ney, 2002, In Proc of ACL

判别式模型

- 判别式模型的形式：

$$P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$$

- ▶ $\{h_i(\cdot)\}$ 是 M 个特征，每个 $h_i(d, \mathbf{s}, \mathbf{t})$ 把 d 映射为一个实数值
- ▶ $\{\lambda_i\}$ 是这些特征对应权重，权重越大表示特征越重要
- ▶ $\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t})$ 描述了 d 的整体质量，值约大 d 越“好”

判别式模型

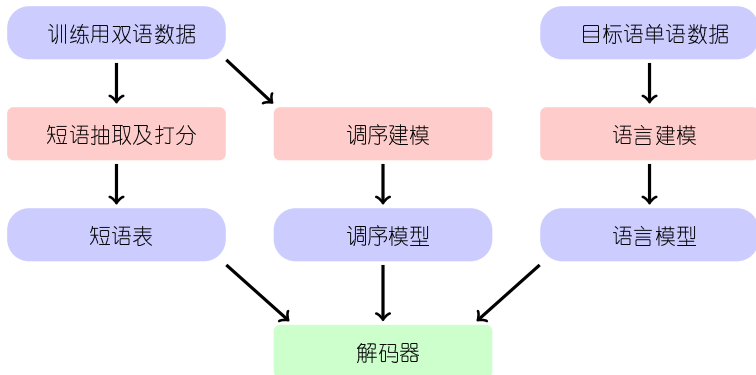
- 判别式模型的形式：

$$P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$$

- ▶ $\{h_i(\cdot)\}$ 是 M 个特征，每个 $h_i(d, \mathbf{s}, \mathbf{t})$ 把 d 映射为一个实数值
 - ▶ $\{\lambda_i\}$ 是这些特征对应权重，权重越大表示特征越重要
 - ▶ $\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t})$ 描述了 d 的整体质量，值约大 d 越“好”
- 判别式模型的优点在于，它可以很方便的引入各种特征。我们只需要设计不同的特征函数 $h_i(\cdot)$ 即可。
 - ▶ 比如，可以定义短语翻译概率作为特征，也可以定义调序的程度作为一个特征
- 两个问题：
 - ▶ 特征定义：定义短语翻译特征和调序特征(马上)
 - ▶ 权重调优：得到最好的特征权重（后面）

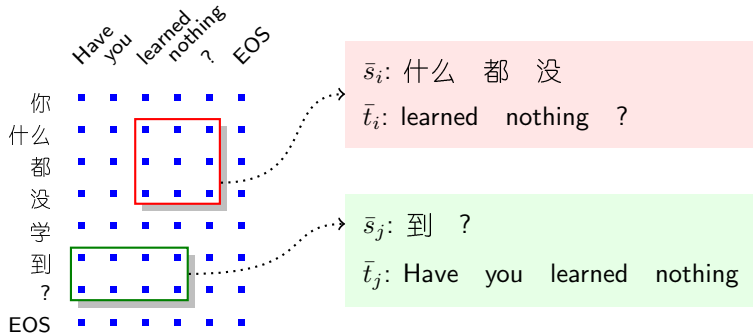
短语系统的架构

- 训练阶段，需要得到三个子模型
 - ① 短语表：短语翻译及每个短语对应的特征值
 - ② 调序模型：短语调序的模型
 - ③ 语言模型：评价译文流畅度的 n -gram语言模型
- 解码阶段利用以上模型对新的句子进行翻译



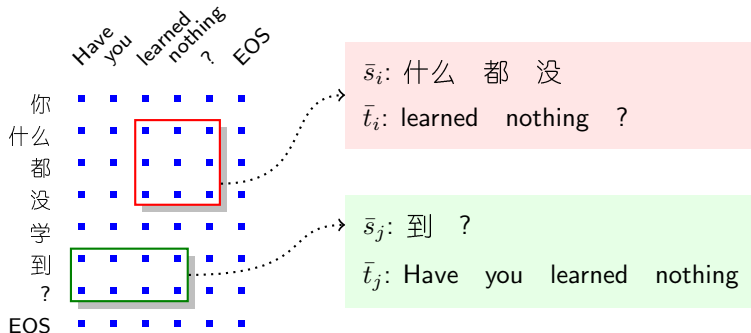
短语获取

- 回到最开始的问题: 给定 \mathbf{s} 和 \mathbf{t} , 如何获得双语短语
 - 如果没有限制, \mathbf{s} 和 \mathbf{t} 之间任何子串映射都可以看做双语短语



短语获取

- 回到最开始的问题: 给定 \mathbf{s} 和 \mathbf{t} , 如何获得双语短语
 - 如果没有限制, \mathbf{s} 和 \mathbf{t} 之间任何子串映射都可以看做双语短语



- 显然, 不加限制的定義短语会带来很多问题
 - 短语数量随句子长度增加急剧膨胀
 - 大量噪声, 如“到 ? \leftrightarrow Have you learned nothing”

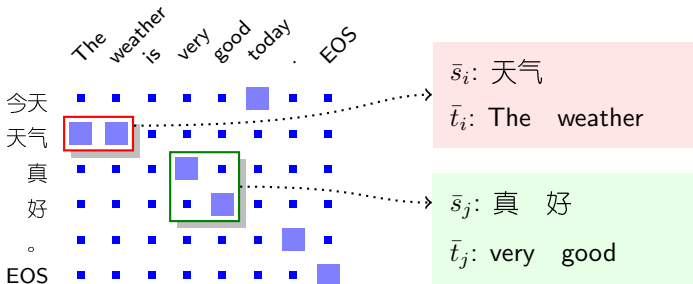
与词对齐的兼容性

- 基于短语的翻译系统性能很大程度取决于短语表的好坏
 - ▶ 在前面一章，我们提到过词对齐的概念，在源语和目标语之间存在着单词级别的对应关系
 - ▶ 借助词对齐信息可以提高抽取双语短语的效率和质量

	The	weather	is	very	good	today	.	EOS
今天	■	■	■	■	■	■	■	■
天气	■	■	■	■	■	■	■	■
真	■	■	■	■	■	■	■	■
好	■	■	■	■	■	■	■	■
。	■	■	■	■	■	■	■	■
EOS	■	■	■	■	■	■	■	■

与词对齐的兼容性

- 基于短语的翻译系统性能很大程度取决于短语表的好坏
 - ▶ 在前面一章，我们提到过词对齐的概念，在源语和目标语之间存在着单词级别的对应关系
 - ▶ 借助词对齐信息可以提高抽取双语短语的效率和质量



- 如何使用词对齐信息来抽取短语对？

基于词对齐的短语抽取

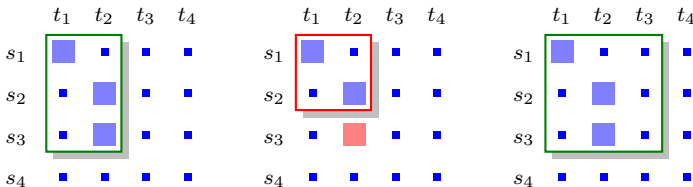
- 抽取短语要与词对齐保持一致
 - 如果短语 \bar{s} 中所有词 s_1, \dots, s_n 在词对齐 A 中与短语 \bar{t} 中的词 t_1, \dots, t_n 都有对齐的点，反之亦然，那么称短语对 (\bar{t}, \bar{s}) 与词对齐 A 一致。即

(\bar{t}, \bar{s}) 与 A 一致 \Leftrightarrow

$$\forall t_i \in \bar{t} : (\bar{t}, \bar{s}) \in A \Rightarrow s_i \in \bar{s}$$

$$\text{AND } \forall s_i \in \bar{s} : (\bar{t}, \bar{s}) \in A \Rightarrow t_i \in \bar{t}$$

$$\text{AND } \exists t_i \in \bar{t}, \exists s_i \in \bar{s} : (\bar{t}, \bar{s}) \in A$$



一致

不一致

一致

基于词对齐的短语抽取(续)

- 短语抽取算法
 - ▶ 遍历所有可能的目标语短语，搜索与它们中每一个相匹配的（最小）源语短语

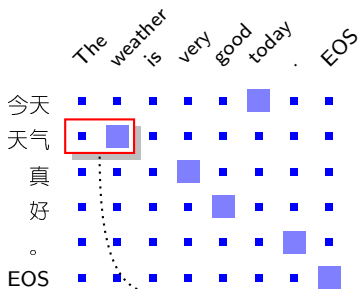
	The	weather	is	very	good	today	.	EOS
今天	■	■	■	■	■	■	■	■
天气	■	■	■	■	■	■	■	■
真	■	■	■	■	■	■	■	■
好	■	■	■	■	■	■	■	■
。	■	■	■	■	■	■	■	■
EOS	■	■	■	■	■	■	■	■

抽取得到的短语:

与词对齐保持一致?

基于词对齐的短语抽取(续)

- 短语抽取算法
 - ▶ 遍历所有可能的目标语短语，搜索与它们中每一个相匹配的（最小）源语短语



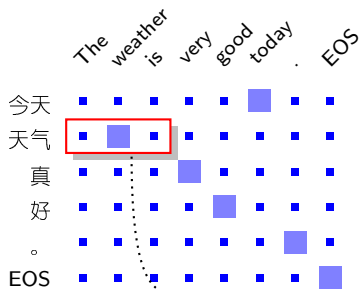
抽取得到的短语:

天气 - The weather

与词对齐保持一致?

基于词对齐的短语抽取(续)

- 短语抽取算法
 - ▶ 遍历所有可能的目标语短语，搜索与它们中每一个相匹配的（最小）源语短语



抽取得到的短语:

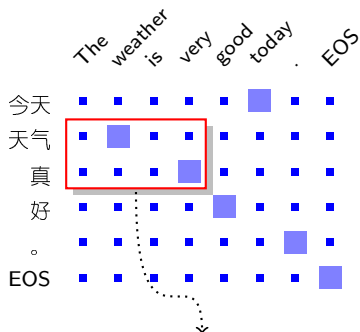
天气 - The weather

天气 - The weather is

与词对齐保持一致?

基于词对齐的短语抽取(续)

- 短语抽取算法
 - ▶ 遍历所有可能的目标语短语，搜索与它们中每一个相匹配的（最小）源语短语



抽取得到的短语:

天气 - The weather

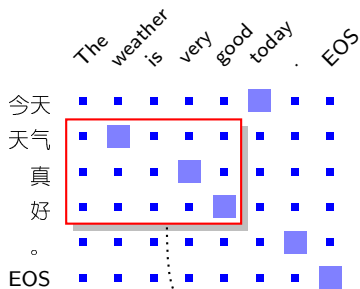
天气 - The weather is

天气真 - The weather is very

与词对齐保持一致?

基于词对齐的短语抽取(续)

- 短语抽取算法
 - ▶ 遍历所有可能的目标语短语，搜索与它们中每一个相匹配的（最小）源语短语



抽取得到的短语:

天气 – The weather

天气 – The weather is

天气真 – The weather is very

天气真好 – The weather is very good

与词对齐保持一致?

基于词对齐的短语抽取(续)

- 短语抽取算法

- ▶ 遍历所有可能的目标语短语，搜索与它们中每一个相匹配的（最小）源语短语
- ▶ 需要对抽取短语的长度进行限制，否则抽取出来的短语对数量十分庞大，导致短语表质量下降

	The	weather	is	very	good	today	.	EOS
今天	■	■	■	■	■	■	■	■
天气	■	■	■	■	■	■	■	■
真	■	■	■	■	■	■	■	■
好	■	■	■	■	■	■	■	■
。	■	■	■	■	■	■	■	■
EOS	■	■	■	■	■	■	■	■

与词对齐保持一致？

抽取得到的短语:

天气 – The weather

天气 – The weather is

天气真 – The weather is very

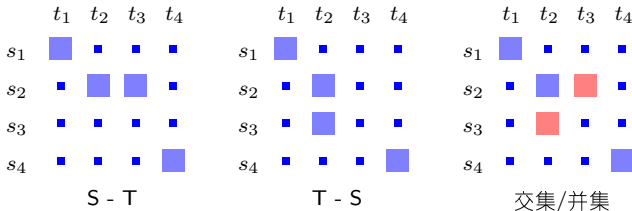
天气真好 – The weather is very good

今天天气真好 – The weather is very
good today

今天 – The today/真好 – very good.....

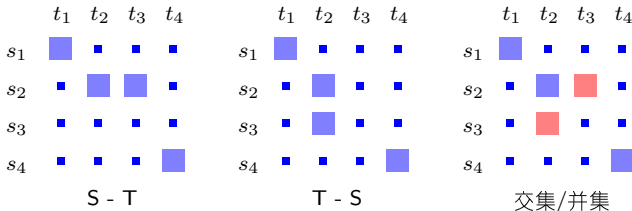
如何获得词对齐

- 基于词的翻译模型（IBM模型）为每个句对建立了词对齐关系，但是IBM模型有一个根本问题，每个目标语单词只对齐到一个源语词（GIZA++ ??）
 - ▶ 词对齐对称化：从两个方向运行IBM模型，合并词对齐结果（交集包含相对可靠的对齐点，并集包含大多数对齐点）



如何获得词对齐

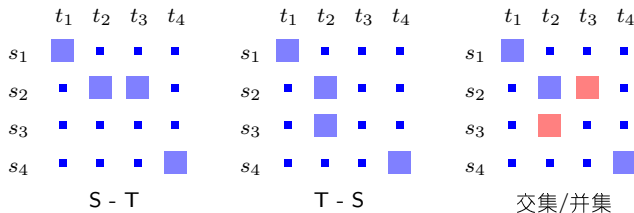
- 基于词的翻译模型（IBM模型）为每个句对建立了词对齐关系，但是IBM模型有一个根本问题，每个目标语单词只对齐到一个源语词（GIZA++ ??）
 - ▶ 词对齐对称化：从两个方向运行IBM模型，合并词对齐结果（交集包含相对可靠的对齐点，并集包含大多数对齐点）



- 其他词对齐方法：FastAlign、Berkeley Word Aligner

如何获得词对齐

- 基于词的翻译模型（IBM模型）为每个句对建立了词对齐关系，但是IBM模型有一个根本问题，每个目标语单词只对齐到一个源语词（GIZA++ ??）
 - ▶ 词对齐对称化：从两个方向运行IBM模型，合并词对齐结果（交集包含相对可靠的对齐点，并集包含大多数对齐点）



- 其他词对齐方法：FastAlign、Berkeley Word Aligner
- 如何评价词对齐？
 - ① 自动指标：词对齐错误率（AER）
 - ② 下游系统：短语抽取、机器翻译...

短语打分 - 翻译概率

- 抽取到短语之后，如何将这些短语对转化成概率化的短语表？这里使用极大似然估计的方法（MLE）对翻译概率进行估计：

$$P(\bar{t}|\bar{s}) = \frac{\text{count}(\bar{s}, \bar{t})}{\text{count}(\bar{s})}$$

- ▶ $\text{count}(\bar{s}, \bar{t})$ 表示短语对 (\bar{s}, \bar{t}) 出现的次数
- ▶ $\text{count}(\bar{s})$ 表示短语 \bar{s} 出现的次数

双语平行数据

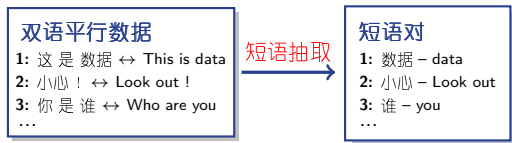
- 1: 这是数据 ↔ This is data
- 2: 小心！ ↔ Look out !
- 3: 你是谁 ↔ Who are you
- ...

短语打分 - 翻译概率

- 抽取到短语之后，如何将这此短语对转化成概率化的短语表？这里使用极大似然估计的方法（MLE）对翻译概率进行估计：

$$P(\bar{t}|\bar{s}) = \frac{\text{count}(\bar{s}, \bar{t})}{\text{count}(\bar{s})}$$

- ▶ $\text{count}(\bar{s}, \bar{t})$ 表示短语对 (\bar{s}, \bar{t}) 出现的次数
- ▶ $\text{count}(\bar{s})$ 表示短语 \bar{s} 出现的次数

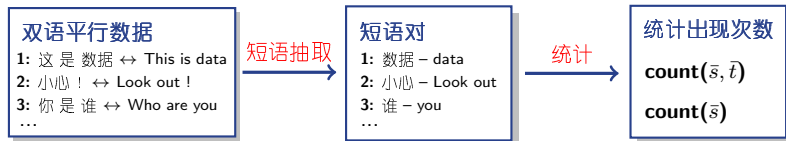


短语打分 - 翻译概率

- 抽取到短语之后，如何将这些短语对转化成概率化的短语表？这里使用极大似然估计的方法（MLE）对翻译概率进行估计：

$$P(\bar{t}|\bar{s}) = \frac{\text{count}(\bar{s}, \bar{t})}{\text{count}(\bar{s})}$$

- ▶ $\text{count}(\bar{s}, \bar{t})$ 表示短语对 (\bar{s}, \bar{t}) 出现的次数
- ▶ $\text{count}(\bar{s})$ 表示短语 \bar{s} 出现的次数



短语打分 - 翻译概率

- 抽取到短语之后，如何将短语对转化成概率化的短语表？这里使用极大似然估计的方法（MLE）对翻译概率进行估计：

$$P(\bar{t}|\bar{s}) = \frac{\text{count}(\bar{s}, \bar{t})}{\text{count}(\bar{s})}$$

- ▶ $\text{count}(\bar{s}, \bar{t})$ 表示短语对 (\bar{s}, \bar{t}) 出现的次数
- ▶ $\text{count}(\bar{s})$ 表示短语 \bar{s} 出现的次数



- 在实际使用中,还可以加入反向翻译概率即 $P(\bar{s}|\bar{t})$ 来提升机器翻译模型性能

短语打分 - 词汇翻译概率

- 对于不常出现的短语可能会产生一些问题，可以将短语分解成词，计算他们的匹配程度。计算公式如下：

$$P_{lex}(\bar{t}|\bar{s}) = \prod_{j=1}^J \frac{1}{|\{j|a(j,i)=1\}|} \sum_{\forall(j,i):a(j,i)=1} w(t_i|s_j)$$

- 源语短语 $\bar{s} = s_1 \dots s_J$, 目标语短语 $\bar{t} = t_1 \dots t_I$, 词对齐矩阵 \mathbf{a}

短语打分 - 词汇翻译概率

- 对于不常出现的短语可能会产生一些问题，可以将短语分解成词，计算他们的匹配程度。计算公式如下：

$$P_{lex}(\bar{t}|\bar{s}) = \prod_{j=1}^J \frac{1}{|\{j|a(j,i)=1\}|} \sum_{\forall (j,i):a(j,i)=1} w(t_i|s_j)$$

- 源语短语 $\bar{s} = s_1 \dots s_J$, 目标语短语 $\bar{t} = t_1 \dots t_I$, 词对齐矩阵 **a**

	t_1	t_2	t_3	t_4	N
s_1	■	■	■	■	■
s_2	■	■	■	■	■
s_3	■	■	■	■	■
s_4	■	■	■	■	■

$$P_{lex}(\bar{t}|\bar{s}) = w(t_1|s_1) \times$$

$$\frac{1}{2}(w(t_2|s_2) + w(t_4|s_2)) \times$$

$$w(N|s_3) \times$$

$$w(t_4|s_4) \times$$

- 词对齐概率 $w(t_i|s_i)$ 可以从平行语料中获取
- 如果对空则使用概率 $w(t_i|N)$

短语打分 - 词汇翻译概率

- 对于不常出现的短语可能会产生一些问题，可以将短语分解成词，计算他们的匹配程度。计算公式如下：

$$P_{lex}(\bar{t}|\bar{s}) = \prod_{j=1}^J \frac{1}{|\{j|a(j,i)=1\}|} \sum_{\forall(j,i):a(j,i)=1} w(t_i|s_j)$$

- 源语短语 $\bar{s} = s_1 \dots s_J$, 目标语短语 $\bar{t} = t_1 \dots t_I$, 词对齐矩阵 **a**

	t_1	t_2	t_3	t_4	N
s_1	■	■	■	■	■
s_2	■	■	■	■	■
s_3	■	■	■	■	■
s_4	■	■	■	■	■

$$P_{lex}(\bar{t}|\bar{s}) = w(t_1|s_1) \times$$

$$\frac{1}{2}(w(t_2|s_2) + w(t_4|s_2)) \times$$

$$w(N|s_3) \times$$

$$w(t_4|s_4) \times$$

- 词对齐概率 $w(t_i|s_i)$ 可以从平行语料中获取
- 如果对空则使用概率 $w(t_i|N)$
- 同翻译概率一样，在模型中可以使用双向词汇翻译概率

短语表实例

- 下面来看一个真实的短语表例子

...

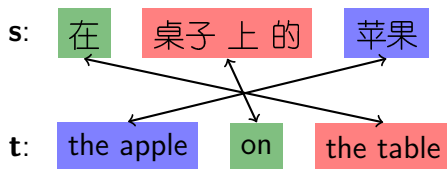
报告 认为 ||| report holds that ||| -2.62 -5.81 -0.91 -2.85 1 0 ||| 4 ||| 0-0 1-1 1-2
， 悲伤 ||| , sadness ||| -1.946 -3.659 0 -3.709 1 0 ||| 1 ||| 0-0 1-1
， 北京 等 ||| , beijing , and other ||| 0 -7.98 0 -3.84 1 0 ||| 2 ||| 0-0 1-1 2-2 2-3 2-4
， 北京 及 ||| , beijing , and ||| -0.69 -1.45 -0.92 -4.80 1 0 ||| 2 ||| 0-0 1-1 2-2
一个 中国 ||| one china ||| 0 -1.725 0 -1.636 1 0 ||| 2 ||| 1-1 2-2

...

- 在短语表的例子中，每行使用 ||| 划分为五个部分
 - 第一部分为源语端的短语
 - 第二部分为目标语端的短语
 - 第三部分为多个特征的值，包含了短语翻译概率、词汇翻译概率等特征
 - 第四部分为短语对在短语抽取集合中出现的频率
 - 第五部分为词对齐信息

翻译中的调序

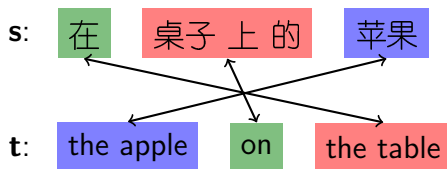
- 通过短语表可以找到每个短语正确的翻译结果，我们仍需要对这些短语进行调序来获取流利的翻译结果



- ▶ 将调序的程度作为特征加入判别式模型

翻译中的调序

- 通过短语表可以找到每个短语正确的翻译结果，我们仍需要对这些短语进行调序来获取流利的翻译结果



- 将调序的程度作为特征加入判别式模型
- 也可以引入约束化简调序问题，如BTG

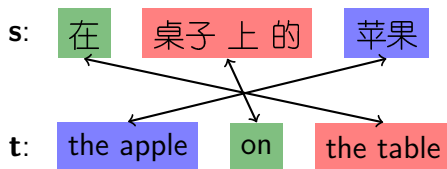
$$X \rightarrow X_1 X_2, X_1 X_2 \quad (R1)$$

$$X \rightarrow X_1 X_2, X_2 X_1 \quad (R2)$$

$$X \rightarrow \bar{s}, \bar{t} \quad (R3)$$

翻译中的调序

- 通过短语表可以找到每个短语正确的翻译结果，我们仍需要对这些短语进行调序来获取流利的翻译结果



- ▶ 将调序的程度作为特征加入判别式模型
- ▶ 也可以引入约束化简调序问题，如BTG

$$X \rightarrow X_1 X_2, X_1 X_2 \quad (R1)$$

$$X \rightarrow X_1 X_2, X_2 X_1 \quad (R2)$$

$$X \rightarrow \bar{s}, \bar{t} \quad (R3)$$

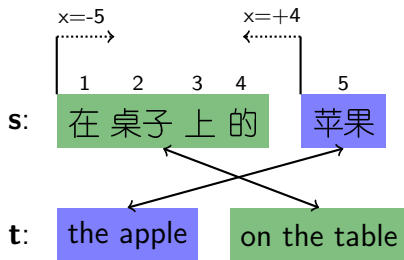
- 常见的调序模型有基于距离的调序模型、词汇化调序模型 (MSD)、最大熵调序模型 (ME)

调序模型1：基于距离的调序

- 参考前一个短语，来判断当前短语是否需要调序，调序距离设为 $\text{start}_i - \text{end}_{i-1} - 1$
 - ▶ start_i 是指翻译成第 i 个目标语短语的源语短语中的第一个词，而 end_i 为源于短语最后一个词（ end_0 为 0）

调序模型1：基于距离的调序

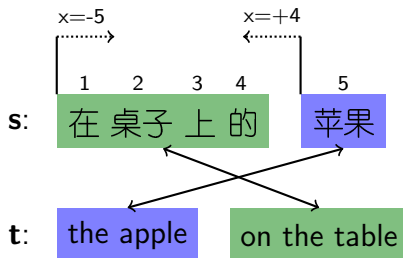
- 参考前一个短语，来判断当前短语是否需要进行，调序距离设为 $\text{start}_i - \text{end}_{i-1} - 1$
 - start_i 是指翻译成第*i*个目标语短语的源语短语中的第一个词，而 end_i 为源于短语最后一个词（ end_0 为0）



目标短语	源短语	距离
1	5	+4
$\text{start}_1 - \text{end}_0 - 1 = 5 - 0 - 1$		
2	1-4	-5
$\text{start}_2 - \text{end}_1 - 1 = 1 - 5 - 1$		

调序模型1：基于距离的调序

- 参考前一个短语，来判断当前短语是否需要进行，调序距离设为 $\text{start}_i - \text{end}_{i-1} - 1$
 - start_i 是指翻译成第 i 个目标语短语的源语短语中的第一个词，而 end_i 为源于短语最后一个词（ end_0 为0）



目标短语	源短语	距离
1	5	+4
$\text{start}_1 - \text{end}_0 - 1 = 5 - 0 - 1$		
2	1-4	-5
$\text{start}_2 - \text{end}_1 - 1 = 1 - 5 - 1$		

- 代价函数选择指数衰减函数 $c(x) = \alpha^{|x|}$ ，其中 α 通过近似估计得到， $\alpha \in [0, 1]$
 - 调序距离越大，调序代价越大

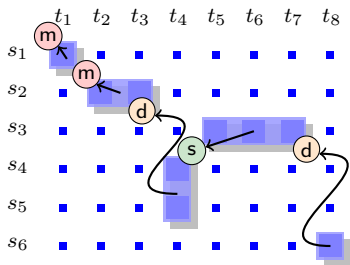
调序模型2: MSD模型

- 基于距离的调序模型仅仅以词语移动的距离为条件，词汇化调序模型以实际短语为条件
 - ▶ 在词汇化调序模型中仅考虑三种调序类型：M、S、D
 - ▶ 三种调序类型仅针对每个短语的前面一个短语

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
s_1	■	■	■	■	■	■	■	■
s_2	■	■	■	■	■	■	■	■
s_3	■	■	■	■	■	■	■	■
s_4	■	■	■	■	■	■	■	■
s_5	■	■	■	■	■	■	■	■
s_6	■	■	■	■	■	■	■	■

调序模型2: MSD模型

- 基于距离的调序模型仅仅以词语移动的距离为条件，词汇化调序模型以实际短语为条件
 - ▶ 在词汇化调序模型中仅考虑三种调序类型：M、S、D
 - ▶ 三种调序类型仅针对每个短语的前面一个短语



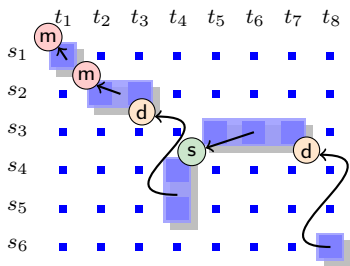
M(monotone):单调调序

S(swap): 与前面一个短语
位置进行交换

D(discontinuous):非连续调序

调序模型2: MSD模型

- 基于距离的调序模型仅仅以词语移动的距离为条件, 词汇化调序模型以实际短语为条件
 - ▶ 在词汇化调序模型中仅考虑三种调序类型: M、S、D
 - ▶ 三种调序类型仅针对每个短语的前面一个短语



M(monotone):单调调序

S(swap): 与前面一个短语
位置进行交换

D(discontinuous):非连续调序

- 引入调序模型来预测调序的方向类型, 计算公式如下

$$P(\mathbf{o}|\mathbf{s}, \mathbf{t}, \mathbf{a}) = \prod_{i=1}^K P(o_i | \bar{s}_{a_i}, \bar{t}_i, a_{i-1}, a_i)$$

调序模型2: MSD模型 (续)

- 来详细的分析一下

$$\Pr(\mathbf{o}|\mathbf{s}, \mathbf{t}, \mathbf{a}) = \prod_{i=1}^K \Pr(o_i | \bar{s}_{a_i}, \bar{t}_i, a_{i-1}, a_i)$$

- 其中 a_i 为 \bar{t}_i 与 \bar{s}_{a_i} 的词对齐, $\bar{t}_1 \dots \bar{t}_K$ 为目标语短语序列
- o_i 为相应的调序类型, $O = \{M, S, D\}$, 与 a_i 和 a_{i-1} 有关

$$o_i = \begin{cases} M & \text{if } a_i - a_{i-1} = 1 \\ S & \text{if } a_i - a_{i-1} = -1 \\ D & \text{otherwise} \end{cases}$$

- 针对不同的调序类型, 调序模型如下

$$f_{M\text{-pre}}(d) = \prod_{i=1}^K \Pr(o_i = M | \bar{s}_{a_i}, \bar{t}_i, a_{i-1}, a_i)$$

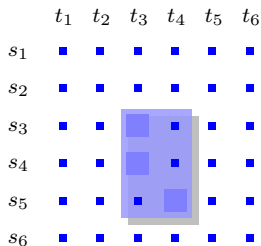
- 我们还可以得到 $f_{S\text{-pre}}(d)$ 和 $f_{D\text{-pre}}(d)$, 此外将 a_{i-1} 换成 a_{i+1} , 还可以得到每个短语与后面短语的调序类型

调序模型2：MSD模型（续）

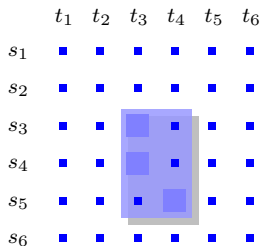
- 统计每种调序方向中每个抽取短语对的频率，使用极大似然估计对其概率分布进行估计
 - ▶ 基于词的调序模型和基于短语的调序模型

调序模型2: MSD模型 (续)

- 统计每种调序方向中每个抽取短语对的频率, 使用极大似然估计对其概率分布进行估计
 - 基于词的调序模型和基于短语的调序模型
 - 假设 \bar{t}_i 由 (t_u, \dots, t_v) 组成, \bar{s}_{a_i} 由 (s_x, \dots, s_y) 组成



基于词

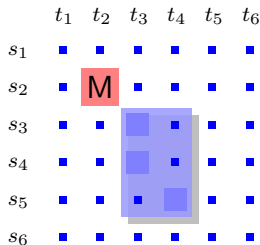


基于短语

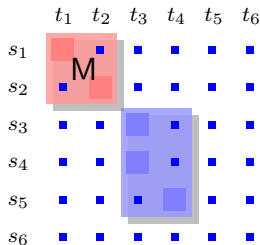
调序模型2: MSD模型 (续)

- 统计每种调序方向中每个抽取短语对的频率, 使用极大似然估计对其概率分布进行估计
 - 基于词的调序模型和基于短语的调序模型
 - 假设 \bar{t}_i 由 (t_u, \dots, t_v) 组成, \bar{s}_{a_i} 由 (s_x, \dots, s_y) 组成

如果在 $(x-1, u-1)$ 存在对齐点, 或者存在可抽取的短语, 则 $o_i = M$



基于词

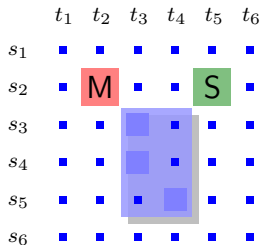


基于短语

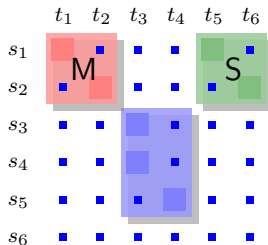
调序模型2: MSD模型 (续)

- 统计每种调序方向中每个抽取短语对的频率, 使用极大似然估计对其概率分布进行估计
 - 基于词的调序模型和基于短语的调序模型
 - 假设 \bar{t}_i 由 (t_u, \dots, t_v) 组成, \bar{s}_{a_i} 由 (s_x, \dots, s_y) 组成

如果在 $(x-1, u-1)$ 存在对齐点, 或者存在可抽取的短语, 则 $o_i = M$



基于词

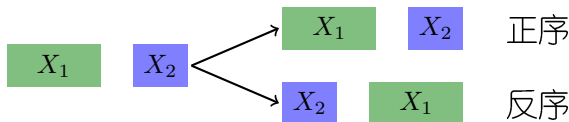


基于短语

如果在 $(x-1, v+1)$ 存在对齐点, 或者存在可抽取的短语, 则 $o_i = S$

调序模型3：分类模型

- 还有一种基于最大熵的调序模型，将调序问题看成是一个二分类问题
 - ▶ 对于相邻的两个短语块，将其合成一个大的更大的块
 - ▶ 根据合成后的顺序 o ，可以分为正序和反序，因此可以看作是一个二分类问题



- ▶ 但是，对于非连续的调序该模型无法处理
- 对于每一种翻译推导 d ，基于最大熵的调序模型的得分计算公式如下

$$f_{\text{ME}}(d) = \prod_{\langle o, X_1, X_2 \rangle \in d} \text{Pr}(o | X_1, X_2)$$

模型得分

- 基于短语的统计机器翻译模型包括三个子模型
 - ① 短语表：短语翻译及每个短语对应的特征值
 - ② 调序模型：短语调序的模型
 - ③ 语言模型：评价译文流畅度的 n -gram语言模型
- 把每个子模型当作一个特征，为每个模型添加一个权重，然后使用对数线性模型对这些子模型进行建模，对数线性模型的形式如下：

$$P(d, \mathbf{t}|\mathbf{s}) \propto \text{mscore}(d, \mathbf{s}|\mathbf{t}) = \exp\left(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t})\right)$$

- 将三个子模型作为具体的特征代入有

$$\text{mscore}(d, \mathbf{t}|\mathbf{s}) = \prod_{(\bar{s}, \bar{t}) \in d} \text{Pr}(\bar{t}|\bar{s})^{\lambda_1} \times f(d)^{\lambda_2} \times \text{Pr}_{\text{lm}}(\mathbf{t})^{\lambda_{\text{lm}}}$$

- 可以引入更多的特征来提高翻译质量（下面介绍）

特征

- **特征1-2： 短语翻译概率**，即正向翻译概率 $\log(P(\bar{s}|\bar{t}))$ 和反向翻译概率 $\log(P(\bar{t}|\bar{s}))$ 。是基于短语的统计机器翻译模型中最主要的特征。
- **特征3-4： 词汇翻译概率**，即正向词汇翻译概率 $\log(P_{\text{lex}}(\bar{t}|\bar{s}))$ 和反向词汇翻译概率 $\log(P_{\text{lex}}(\bar{s}|\bar{t}))$ 。用来描述短语对中源语端单词和目标语端单词的对应关系

特征

- **特征1-2：短语翻译概率**，即正向翻译概率 $\log(P(\bar{s}|\bar{t}))$ 和反向翻译概率 $\log(P(\bar{t}|\bar{s}))$ 。是基于短语的统计机器翻译模型中最主要的特征。
- **特征3-4：词汇翻译概率**，即正向词汇翻译概率 $\log(P_{\text{lex}}(\bar{t}|\bar{s}))$ 和反向词汇翻译概率 $\log(P_{\text{lex}}(\bar{s}|\bar{t}))$ 。用来描述短语对中源语端单词和目标语端单词的对应关系
- **特征5：n-gram语言模型**，即 $\log(P_{\text{lm}}(\mathbf{t}))$ 。度量译文的流畅度，可以使用大规模目标语单语数据得到。
- **特征6：译文长度**，即 $|\mathbf{t}|$ 。避免模型倾向于短译文，同时让系统自动学习对译文长度的偏好。
- **特征7：翻译规则数量**。这个特征是为了避免模型仅仅使用少量特征构成翻译推导(因为翻译概率相乘，因子少结果一般会大一些)，同时让系统自动学习对使用规则数量的偏好。

特征(续)

- **特征8**：源语言被翻译为空的单词数量。注意，空翻译规则(或特征)有时也被称作evil feature，这类特征在一些数据集上对BLEU有很好的提升作用，但是会造成人工评价的下降，因此需要谨慎使用。
- **特征9**：基于最大熵的调序模型， $f_{ME}(d)$ 。
- **特征10**：基于MSD的调序模型，包括与前一个短语的调序 $f_{M-pre}(d)$ 、 $f_{S-pre}(d)$ 、 $f_{D-pre}(d)$ ，和后一个短语的调序 $f_{M-fol}(d)$ 、 $f_{S-fol}(d)$ 、 $f_{D-fol}(d)$
- 最终模型得分

$$\text{mscore}(d, \mathbf{t}|\mathbf{s}) = \prod_{(\bar{s}, \bar{t}) \in d} \text{pscore}(\bar{s}, \bar{t}) \times f_{ME}(d)^{\lambda_{ME}} \times f_{MSD}(d)^{\lambda_{MSD}} \times \\ \Pr_{lm}(\mathbf{t})^{\lambda_{lm}} \times \exp(\lambda_{TWB} \cdot \text{length}(\mathbf{t}))/Z(\mathbf{s})$$

$$\text{pscore}(\bar{s}, \bar{t}) = \Pr(\bar{t}|\bar{s})^{\lambda_1} \times \Pr(\bar{s}|\bar{t})^{\lambda_2} \times \Pr_{lex}(\bar{t}|\bar{s})^{\lambda_3} \times \Pr_{lex}(\bar{s}|\bar{t})^{\lambda_4} \times \\ \exp(\lambda_{PB}) \times \exp(\lambda_{WDB} \cdot \delta(\bar{s} \rightarrow null))$$

特征权重调优

- 对于训练样本 $S = \{(f_1, r_1), \dots, (f_s, r_s)\}$, 有
 - ▶ f_s 为样本中的第 s 个源语句子, r_s 为相应的译文, 通常使用 $R = \{r_1, \dots, r_s\}$ 来表示训练样本的参考译文
 - ▶ 针对每个源语句子, 解码器可以生成一个 **n-best** 结果 $\{d_{ij}\}$
- 对于模型参数 λ , 最佳的翻译推导为 $D^* = \{d_1^*, \dots, d_i^*\}$

$$d_i^* = \arg \max_{d_{ij}} \sum_{k=1}^M \lambda_k \cdot h_k(d_{ij})$$

特征权重调优

- 对于训练样本 $S = \{(f_1, r_1), \dots, (f_s, r_s)\}$, 有
 - ▶ f_s 为样本中的第 s 个源语句子, r_s 为相应的译文, 通常使用 $R = \{r_1, \dots, r_s\}$ 来表示训练样本的参考译文
 - ▶ 针对每个源语句子, 解码器可以生成一个 **n-best** 结果 $\{d_{ij}\}$
- 对于模型参数 λ , 最佳的翻译推导为 $D^* = \{d_1^*, \dots, d_i^*\}$

$$d_i^* = \arg \max_{d_{ij}} \sum_{k=1}^M \lambda_k \cdot h_k(d_{ij})$$

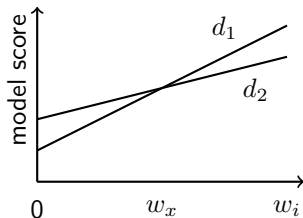
- 最小错误率训练 (MERT)
 - ▶ 定义一个错误函数 $\text{Err}(D^*, R)$ 来衡量推导 D^* 得到的译文与参考答案 R 之间的差距, 通过调整权重 λ 来最小化错误率
 - ▶ 常见的错误函数有词错误率 (WER)、位置错误率 (PER)、BLEU 值以及 NIST 值

$$\lambda^* = \arg \min_{\lambda} \mathbf{Err}(D^*, R)$$

特征权重调优（续）

- 如何得到最优的 λ^*
 - ▶ 最简单的方法是枚举所有可能的 λ 值，但是这样做效率很低。可以只考虑最优译文发生变化的点：)
 - ▶ 对于每个训练样本，假设有2-best个推导 $\mathbf{d} = \{d_1, d_2\}$ ，每个推导 d 的得分 $\text{score}(d)$ 可以表示成关于权重 λ_i 的函数

$$\text{score}(d) = \lambda_i \cdot h_i(d) + \sum_{k \neq i}^M \lambda_k \cdot h_k(d) = a \cdot \lambda_i + b$$

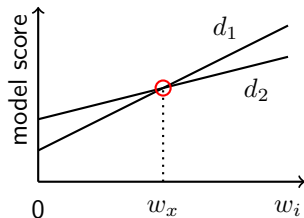


特征权重调优（续）

- 如何得到最优的 λ^*
 - ▶ 最简单的方法是枚举所有可能的 λ 值，但是这样做效率很低。可以只考虑最优译文发生变化的点：)
 - ▶ 对于每个训练样本，假设有2-best个推导 $\mathbf{d} = \{d_1, d_2\}$ ，每个推导 d 的得分 $\text{score}(d)$ 可以表示成关于权重 λ_i 的函数

$$\text{score}(d) = \lambda_i \cdot h_i(d) + \sum_{k \neq i}^M \lambda_k \cdot h_k(d) = a \cdot \lambda_i + b$$

1. 找到最优译文 E^* 发生变化的位置



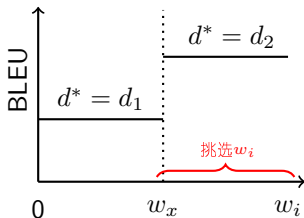
特征权重调优（续）

- 如何得到最优的 λ^*
 - ▶ 最简单的方法是枚举所有可能的 λ 值，但是这样做效率很低。可以只考虑最优译文发生变化的点：)
 - ▶ 对于每个训练样本，假设有2-best个推导 $\mathbf{d} = \{d_1, d_2\}$ ，每个推导 d 的得分 $\text{score}(d)$ 可以表示成关于权重 λ_i 的函数

$$\text{score}(d) = \lambda_i \cdot h_i(d) + \sum_{k \neq i}^M \lambda_k \cdot h_k(d) = a \cdot \lambda_i + b$$

1. 找到最优译文 E^* 发生变化的位置

2. 对译文按照BLEU值进行排序，
在分数高的范围中挑选新的 λ_i



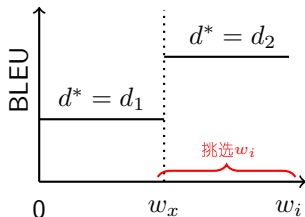
特征权重调优（续）

- 如何得到最优的 λ^*
 - ▶ 最简单的方法是枚举所有可能的 λ 值，但是这样做效率很低。可以只考虑最优译文发生变化的点：)
 - ▶ 对于每个训练样本，假设有2-best个推导 $\mathbf{d} = \{d_1, d_2\}$ ，每个推导 d 的得分 $\text{score}(d)$ 可以表示成关于权重 λ_i 的函数

$$\text{score}(d) = \lambda_i \cdot h_i(d) + \sum_{k \neq i}^M \lambda_k \cdot h_k(d) = a \cdot \lambda_i + b$$

1. 找到最优译文 E^* 发生变化的位置

2. 对译文按照BLEU值进行排序，
在分数高的范围中挑选新的 λ_i



- ▶ 还有一些技巧可以防止训练出现过拟合和陷入局部最优

解码问题

- 解码是根据模型以及输入原文，找到得分最高的推导 d^*

$$d^* = \arg \max_{d \in D} \text{score}(d)$$

- ▶ 其中 D 表示所有可能的推导构成的搜索空间。
- ▶ $\Pr(d, \mathbf{t}|\mathbf{s})$ 表示前面提到的所有特征的得分
- 实际解码过程中，通常按从左到右的顺序生成译文，递增的计算翻译概率，同时对已翻译的原文进行标记

s: 桌子 上 有 一个 苹果

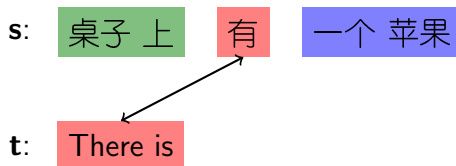
t:

解码问题

- 解码是根据模型以及输入原文，找到得分最高的推导 d^*

$$d^* = \arg \max_{d \in D} \text{score}(d)$$

- ▶ 其中 D 表示所有可能的推导构成的搜索空间。
- ▶ $\Pr(d, \mathbf{t}|\mathbf{s})$ 表示前面提到的所有特征的得分
- 实际解码过程中，通常按从左到右的顺序生成译文，递增的计算翻译概率，同时对已翻译的原文进行标记

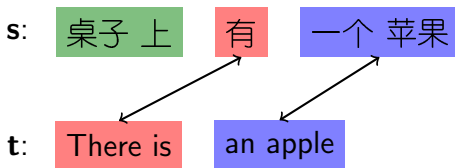


解码问题

- 解码是根据模型以及输入原文，找到得分最高的推导 d^*

$$d^* = \arg \max_{d \in D} \text{score}(d)$$

- ▶ 其中 D 表示所有可能的推导构成的搜索空间。
- ▶ $\Pr(d, \mathbf{t}|\mathbf{s})$ 表示前面提到的所有特征的得分
- 实际解码过程中，通常按从左到右的顺序生成译文，递增的计算翻译概率，同时对已翻译的原文进行标记

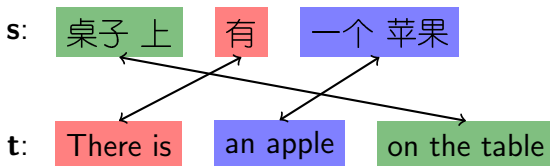


解码问题

- 解码是根据模型以及输入原文，找到得分最高的推导 d^*

$$d^* = \arg \max_{d \in D} \text{score}(d)$$

- ▶ 其中 D 表示所有可能的推导构成的搜索空间。
- ▶ $\Pr(d, \mathbf{t} | \mathbf{s})$ 表示前面提到的所有特征的得分
- 实际解码过程中，通常按从左到右的顺序生成译文，递增的计算翻译概率，同时对已翻译的原文进行标记



解码问题 - 翻译选项

- 对于每个输入的源语句s，可以从短语表中查询到所有可能的翻译选项，用来翻译

s:	桌子	上	有	一个	苹果
table	on	have	one	apple	
desk	up	there is	an	apples	
	upon there		one apple		
upon the tabel			an apple		
on tabel		there is an apple			
on the tabel		have an apple...			

- 正确的翻译都在翻译选项中，但是里面也包含了很多其他翻译选项，接下来介绍如何找到最优的译文？

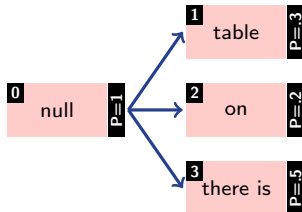
解码问题 - 假设扩展

- 从翻译选项中挑选合适的选项，顺序地构建输出，构建的局部翻译称为翻译假设
 - 翻译的起点是空假设，局部概率得分是1

0	null	P=1
---	------	-----

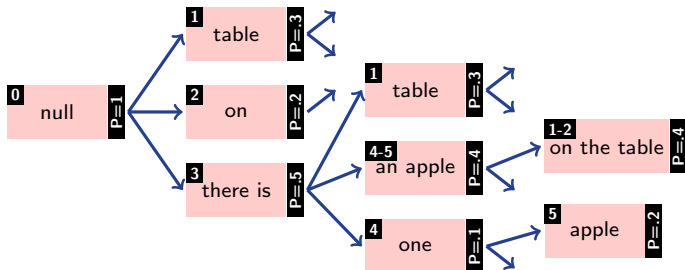
解码问题 - 假设扩展

- 从翻译选项中挑选合适的选项，顺序地构建输出，构建的局部翻译称为翻译假设
 - 翻译的起点是空假设，局部概率得分是1
 - 挑选一个翻译选项扩展为新的翻译假设，同时记录已翻译的原文并计算翻译代价（可以同时生成多种翻译假设）



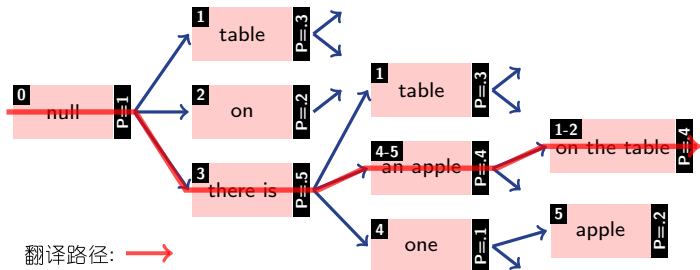
解码问题 - 假设扩展

- 从翻译选项中挑选合适的选项，顺序地构建输出，构建的局部翻译称为翻译假设
- 1 翻译的起点是空假设，局部概率得分是1
- 2 挑选一个翻译选项扩展为新的翻译假设，同时记录已翻译的原文并计算翻译代价（可以同时生成多种翻译假设）
- 3 对未覆盖的源语应用上述方法



解码问题 - 假设扩展

- 从翻译选项中挑选合适的选项，顺序地构建输出，构建的局部翻译称为翻译假设
- 1 翻译的起点是空假设，局部概率得分是1
- 2 挑选一个翻译选项扩展为新的翻译假设，同时记录已翻译的原文并计算翻译代价（可以同时生成多种翻译假设）
- 3 对未覆盖的源语应用上述方法
- 4 当翻译假设覆盖了所有的原文时，就得到了一个完整的翻译假设，从所有的翻译假设中找到一个概率最高的翻译

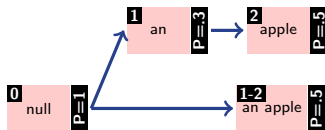


假设重组

- 随着源语长度的增加，搜索空间将变得十分巨大，其中相同的翻译假设可以通过不同的搜索路径得到
 - ▶ 可以通过假设重组，来减少翻译假设的数量

假设重组

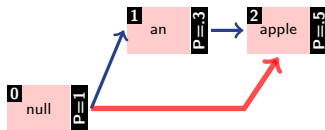
- 随着源语长度的增加，搜索空间将变得十分巨大，其中相同的翻译假设可以通过不同的搜索路径得到
 - 可以通过假设重组，来减少翻译假设的数量



1. 翻译过相同数量的源语，输出相同的翻译。舍弃概率低的假设

假设重组

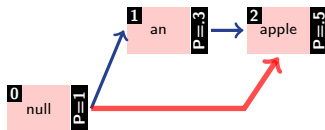
- 随着源语长度的增加，搜索空间将变得十分巨大，其中相同的翻译假设可以通过不同的搜索路径得到
 - 可以通过假设重组，来减少翻译假设的数量



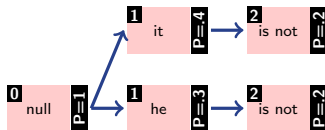
1. 翻译过相同数量的源语，输出相同的翻译。舍弃概率低的假设

假设重组

- 随着源语长度的增加，搜索空间将变得十分巨大，其中相同的翻译假设可以通过不同的搜索路径得到
 - 可以通过**假设重组**，来减少翻译假设的数量



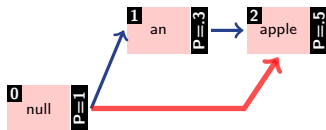
1. 翻译过相同数量的源语，输出相同的翻译。舍弃概率低的假设



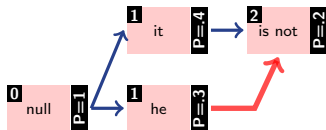
2. 输出结果略有不同也可以重新组合，只要有相同后续代价即可

假设重组

- 随着源语长度的增加，搜索空间将变得十分巨大，其中相同的翻译假设可以通过不同的搜索路径得到
 - 可以通过**假设重组**，来减少翻译假设的数量



1. 翻译过相同数量的源语，输出相同的翻译。舍弃概率低的假设

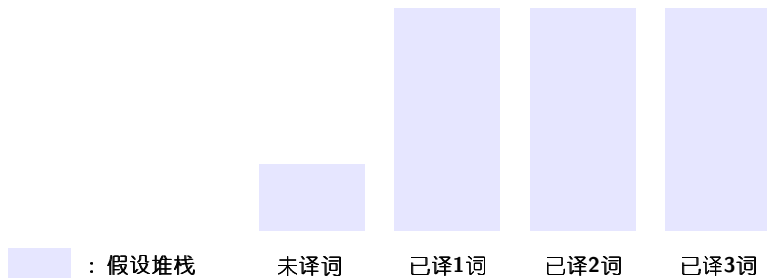


2. 输出结果略有不同也可以重新组合，只要有相同后续代价即可

- 重组假设可以减少内部表示不同，如不同短语切分的情况，能够更严格和高效地搜索

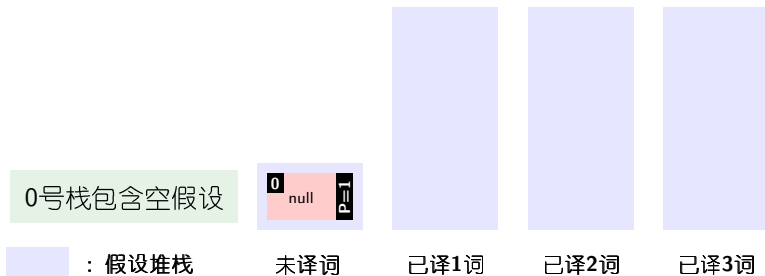
栈解码

- 还有一种减小搜索空间的方法，如果早期发现较差的翻译假设，则将它舍弃，并忽略由它扩展出来的翻译假设
 - ▶ 整理翻译假设，放进假设堆栈中
 - ▶ 堆栈按照已翻译的词数进行分类
 - ▶ 如果栈过大，则删掉栈里面最差的那些假设



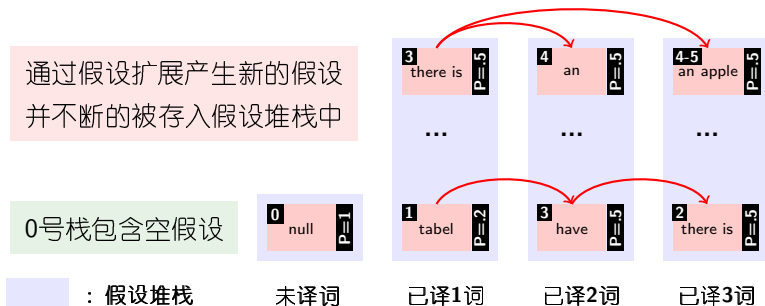
栈解码

- 还有一种减小搜索空间的方法，如果早期发现较差的翻译假设，则将它舍弃，并忽略由它扩展出来的翻译假设
 - ▶ 整理翻译假设，放进假设堆栈中
 - ▶ 堆栈按照已翻译的词数进行分类
 - ▶ 如果栈过大，则删掉栈里面最差的那些假设



栈解码

- 还有一种减小搜索空间的方法，如果早期发现较差的翻译假设，则将它舍弃，并忽略由它扩展出来的翻译假设
 - ▶ 整理翻译假设，放进假设堆栈中
 - ▶ 堆栈按照已翻译的词数进行分类
 - ▶ 如果栈过大，则删掉栈里面最差的那些假设



- 使用栈解码可以很大程度上提高解码效率

基于层次短语的翻译模型

基于短语的方法的不足

- 短语可以很好的捕捉词语之间的局部搭配和调序，但是长距离依赖需要更长的短语
 - ▶ 实践中发现使用超过长度3的短语作用不大
 - ▶ 短语非常稀疏，包含多个词的短语大多非常低频

短语(中文)	训练数据中出现频次
包含	3341
包含 多个	213
包含 多个 词	12
包含 多个 词 的	8
包含 多个 词 的 短语	0
包含 多个 词 的 短语 大多	0

基于短语的方法的不足

- 短语可以很好的捕捉词语之间的局部搭配和调序，但是长距离依赖需要更长的短语
 - ▶ 实践中发现使用超过长度3的短语作用不大
 - ▶ 短语非常稀疏，包含多个词的短语大多非常低频

短语(中文)	训练数据中出现频次
包含	3341
包含 多个	213
包含 多个 词	12
包含 多个 词 的	8
包含 多个 词 的 短语	0
包含 多个 词 的 短语 大多	0

- 简单使用短语和 n -gram语言模型无法处理长距离的调序
 - ▶ 引入独立的调序模型，比如简单的基于距离的调序
 - ▶ 当然，也可以设计更加复杂的调序模型

基于短语的方法的不足 - 一个实例

- 一个短语翻译不成功的例子(Chiang, 2015)

源语: 澳洲 是 与 北韩 有 邦交 的 少数 国家 之一

短语系统: Australia is diplomatic relations with North Korea
is one of the few countries

参考译文: Australia is one of the few countries that have
diplomatic relations with North Korea

基于短语的方法的不足 - 一个实例

- 一个短语翻译不成功的例子(Chiang, 2015)

源语: 澳洲 是 与 北韩 有 邦交 的 少数 国家 之一

短语系统: Australia is diplomatic relations with North Korea ¹
is one of the few countries ²

参考译文: Australia is one of the few countries that have
diplomatic relations with North Korea

- 从短语系统翻译结果可以看出
 - ▶ diplomatic relations with North Korea能够进行正确调序
 - ▶ one of the few countries能够进行正确调序
 - ▶ 但是, 两个短语 (¹ 和 ²) 没有正确调序 - 怎么办?

引入新的翻译单元

- 显然，通过由连续单词构成的短语拼装出理想的译文需要比较复杂的机制。但是，语言是有“结构”的，我们可以用一种新的方式描述翻译：

$\langle \text{与 } X_1 \text{ 有 } X_2, \text{ have } X_2 \text{ with } X_1 \rangle$

这里， X_1 和 X_2 表示变量，源语和目标语相同的变量表示对应关系，变量可以被其它连续词串替换。这样，这种源语言和目标语言的对应构成了一种翻译规则或模版，相当于把“与 X_1 有 X_2 ”翻译为“have X_2 with X_1 ”，调序信息就隐含在变量的编号里

引入新的翻译单元

- 显然，通过由连续单词构成的短语拼装出理想的译文需要比较复杂的机制。但是，语言是有“结构”的，我们可以用一种新的方式描述翻译：

$\langle \text{与 } X_1 \text{ 有 } X_2, \text{ have } X_2 \text{ with } X_1 \rangle$

这里， X_1 和 X_2 表示变量，源语和目标语相同的变量表示对应关系，变量可以被其它连续词串替换。这样，这种源语言和目标语言的对应构成了一种翻译规则或模版，相当于把“与 X_1 有 X_2 ”翻译为“have X_2 with X_1 ”，调序信息就隐含在变量的编号里

- 类似的，可以写出很多这样的翻译规则

$\langle X_1 \text{ 是 } X_2, X_1 \text{ is } X_2 \rangle$

$\langle X_1 \text{ 之一, one of } X_1 \rangle$

$\langle X_1 \text{ 的 } X_2, X_2 \text{ that have } X_1 \rangle$

使用翻译规则描述双语句子生成过程

- 翻译过程可以用上述规则描述
 - ▶ 每个变量在源语和目标语端可以被同步替换

北韩

North Korea

使用翻译规则描述双语句子生成过程

- 翻译过程可以用上述规则描述
 - ▶ 每个变量在源语和目标语端可以被同步替换

北韩

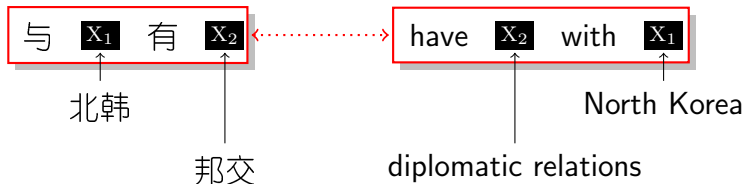
North Korea

邦交

diplomatic relations

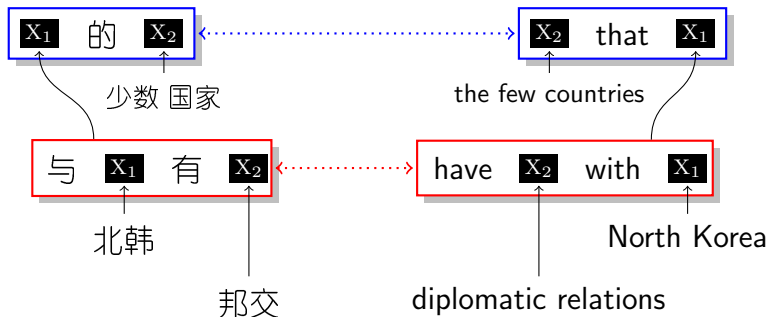
使用翻译规则描述双语句子生成过程

- 翻译过程可以用上述规则描述
 - ▶ 每个变量在源语和目标语端可以被同步替换



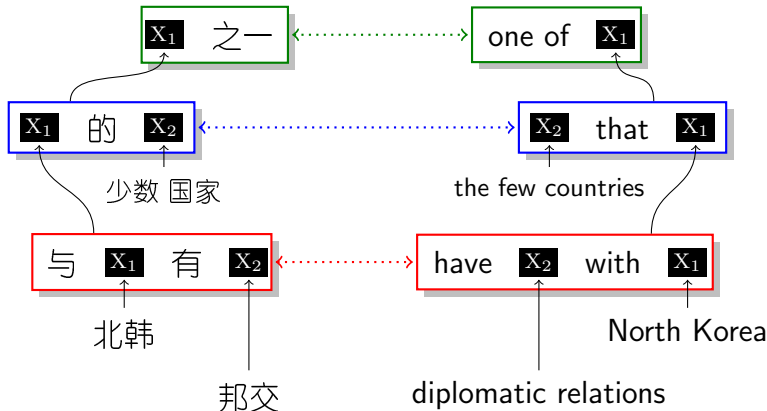
使用翻译规则描述双语句子生成过程

- 翻译过程可以用上述规则描述
 - ▶ 每个变量在源语和目标语端可以被同步替换



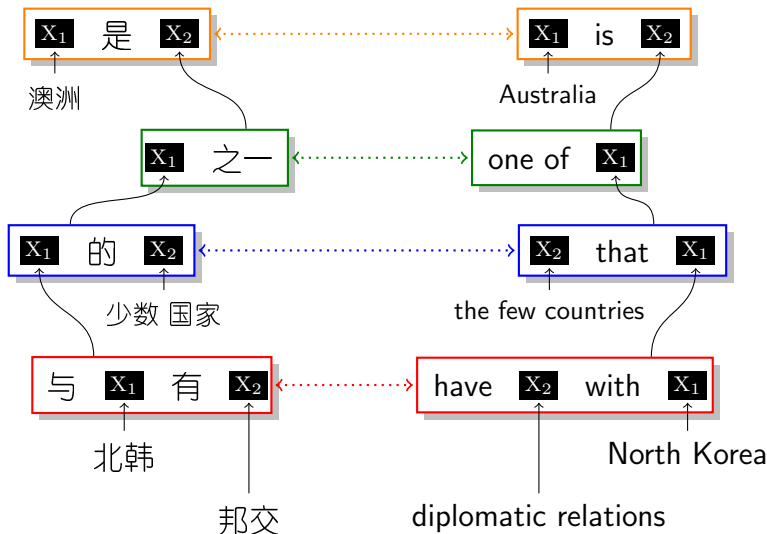
使用翻译规则描述双语句子生成过程

- 翻译过程可以用上述规则描述
 - ▶ 每个变量在源语和目标语端可以被同步替换



使用翻译规则描述双语句子生成过程

- 翻译过程可以用上述规则描述
 - 每个变量在源语和目标语端可以被同步替换



同步上下文无关文法

- 以上这种对翻译的描述方式，可以用同步上下文无关文法来定义，记Synchronous Context-Free Grammar(SCFG)

定义 - 同步上下文无关文法

一个同步上下文无关文法由五部分构成 (N, T_s, T_t, I, R) ，其中

1. N 是非终结符集合
2. T_s 和 T_t 分别是源语言和目标语言终结符集合
3. $I \subseteq N$ 起始非终结符集合
4. R 是规则集合，每条规则 $r \in R$ 有如下形式

$$\text{LHS} \rightarrow \langle \alpha, \beta, \sim \rangle$$

其中， $\text{LHS} \in N$ 表示规则的左部，它是一个非终结符；规则右部由三部分组成， $\alpha \in (N \cup T_s)^*$ 表示由源语言终结符和非终结符组成的串； $\beta \in (N \cup T_t)^*$ 表示由目标语言终结符和非终结符组成的串； \sim 表示 α 和 β 中终结符的1-1对应关系

同步上下文无关文法 - 实例

- SCFG可以被看做是对CFG的扩展，相当于把单语的CFG扩展到双语，如下是一些SCFG规则，其中每个规则非终结符的对应用非终结符的标号表示

$S \rightarrow \langle NP_1 \text{ 希望 } VP_2, NP_1 \text{ wish to } VP_2 \rangle$

$VP \rightarrow \langle \text{对 } NP_1 \text{ 感到 } VP_2, \text{be } VP_2 \text{ with } NP_1 \rangle$

$NN \rightarrow \langle \text{强大, strong} \rangle$

同步上下文无关文法 - 实例

- SCFG可以被看做是对CFG的扩展，相当于把单语的CFG扩展到双语，如下是一些SCFG规则，其中每个规则非终结符的对应用非终结符的标号表示

$$S \rightarrow \langle NP_1 \text{ 希望 } VP_2, NP_1 \text{ wish to } VP_2 \rangle$$
$$VP \rightarrow \langle \text{对 } NP_1 \text{ 感到 } VP_2, \text{be } VP_2 \text{ with } NP_1 \rangle$$
$$NN \rightarrow \langle \text{强大, strong} \rangle$$

- 这里NP、VP等是有语言学意义的非终结符。当然，在机器翻译中这些并不是必要的，可以使用更简单的文法，只包含一种非终结符

$$X \rightarrow \langle X_1 \text{ 希望 } X_2, X_1 \text{ wish to } X_2 \rangle$$
$$X \rightarrow \langle \text{对 } X_1 \text{ 感到 } X_2, \text{be } X_2 \text{ with } X_1 \rangle$$
$$X \rightarrow \langle \text{强大, strong} \rangle$$

一个完整的文法

- 对于一个中文-英文句对，假设可以得到如下同步上下文无关文法

源语： 进口 大幅度 下降 了

目标语： The imports have drastically fallen

SCFG:

$r_1: X \rightarrow \langle \text{进口 } X_1, \text{The imports } X_1 \rangle$

$r_2: X \rightarrow \langle X_1 \text{ 下降 } X_2, X_2 \text{ } X_1 \text{ fallen} \rangle$

$r_3: X \rightarrow \langle \text{大幅度, drastically} \rangle$

$r_4: X \rightarrow \langle \text{了, have} \rangle$

其中，规则 r_1 和 r_2 是右部含有变量的规则，这些变量可以被其它规则的右部替换；规则 r_2 是调序规则；规则 r_3 和 r_4 是纯词汇化规则，表示单词或者短语的翻译

翻译规则的推导

- 使用SCFG规则的过程构成了一个**推导**，每次规则的使用都会同步替换源语言和目标语言串中的一个非终结符

$\langle X_1, X_1 \rangle$

翻译规则的推导

- 使用SCFG规则的过程构成了一个**推导**，每次规则的使用都会同步替换源语言和目标语言串中的一个非终结符

$$\begin{array}{c} \langle X_1, X_1 \rangle \\ \xrightarrow{r_1} \langle \text{进} \square X_2, \text{The imports } X_2 \rangle \end{array}$$

翻译规则的推导

- 使用SCFG规则的过程构成了一个**推导**，每次规则的使用都会同步替换源语言和目标语言串中的一个非终结符

$\langle X_1, X_1 \rangle$

$\xrightarrow{r_1} \langle \text{进} \square X_2, \text{The imports } X_2 \rangle$

$\xrightarrow{r_2} \langle \text{进} \square X_3 \text{ 下降 } X_4, \text{The imports } X_4 X_3 \text{ fallen} \rangle$

翻译规则的推导

- 使用SCFG规则的过程构成了一个**推导**，每次规则的使用都会同步替换源语言和目标语言串中的一个非终结符

$$\begin{aligned} & \langle X_1, X_1 \rangle \\ \xrightarrow{r_1} & \langle \text{进} \square X_2, \text{The imports } X_2 \rangle \\ \xrightarrow{r_2} & \langle \text{进} \square X_3 \text{ 下降 } X_4, \text{The imports } X_4 X_3 \text{ fallen} \rangle \\ \xrightarrow{r_3} & \langle \text{进} \square \text{大幅度} \text{ 下降 } X_4, \\ & \text{The imports } X_4 \text{ drastically fallen} \rangle \end{aligned}$$

翻译规则的推导

- 使用SCFG规则的过程构成了一个**推导**，每次规则的使用都会同步替换源语言和目标语言串中的一个非终结符

$$\begin{aligned} & \langle X_1, X_1 \rangle \\ \xrightarrow{r_1} & \langle \text{进} \square X_2, \text{The imports } X_2 \rangle \\ \xrightarrow{r_2} & \langle \text{进} \square X_3 \text{ 下降 } X_4, \text{The imports } X_4 X_3 \text{ fallen} \rangle \\ \xrightarrow{r_3} & \langle \text{进} \square \text{大幅度} \text{ 下降 } X_4, \\ & \text{The imports } X_4 \text{ drastically fallen} \rangle \\ \xrightarrow{r_4} & \langle \text{进} \square \text{大幅度} \text{ 下降 } \text{了}, \\ & \text{The imports have drastically fallen} \rangle \end{aligned}$$

翻译规则的推导

- 使用SCFG规则的过程构成了一个**推导**，每次规则的使用都会同步替换源语言和目标语言串中的一个非终结符

$$\begin{aligned} & \langle X_1, X_1 \rangle \\ \xrightarrow{r_1} & \langle \text{进} \square X_2, \text{The imports } X_2 \rangle \\ \xrightarrow{r_2} & \langle \text{进} \square X_3 \text{ 下降 } X_4, \text{The imports } X_4 X_3 \text{ fallen} \rangle \\ \xrightarrow{r_3} & \langle \text{进} \square \text{大幅度} \text{ 下降 } X_4, \\ & \text{The imports } X_4 \text{ drastically fallen} \rangle \\ \xrightarrow{r_4} & \langle \text{进} \square \text{大幅度} \text{ 下降 } \text{了}, \\ & \text{The imports have drastically fallen} \rangle \end{aligned}$$

这里把 d 定义为由规则 r_1, r_2, r_3, r_4 构成的SCFG推导，记作

$$d = r_1 \circ r_2 \circ r_3 \circ r_4$$

显然 d 定义了从源于句子生成目标语译文的一个过程

胶水规则

- 在实际系统中往往会遇到需要把两个局部翻译线性的拼接到一起的情况，因此可以在文法中引入胶水规则(glue rule)来处理，形式如下

$$S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$$

$$S \rightarrow \langle X_1, X_1 \rangle$$

胶水规则

- 在实际系统中往往会遇到需要把两个局部翻译线性的拼接到一起的情况，因此可以在文法中引入胶水规则(glue rule)来处理，形式如下

$$S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$$

$$S \rightarrow \langle X_1, X_1 \rangle$$

- 本质上，胶水规则会顺序的拼接若干片段，最后整个句子会被归纳为S

$$S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$$

$$\rightarrow \langle S_3 X_4 X_2, S_3 X_4 X_2 \rangle$$

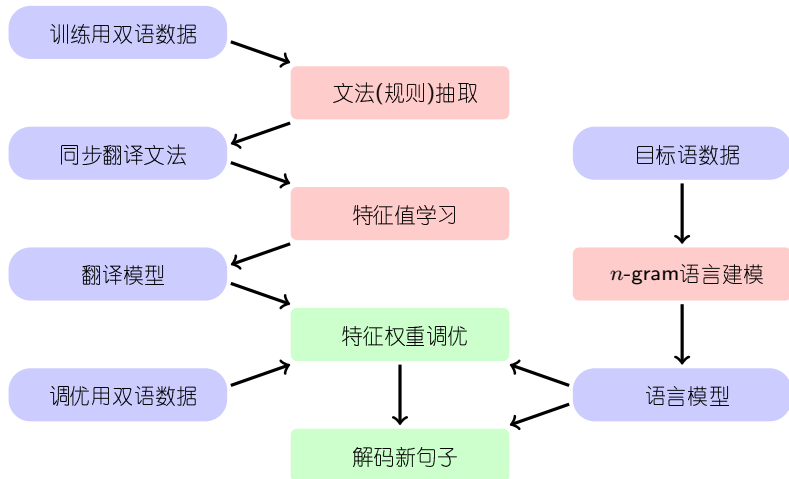
$$\rightarrow \dots$$

$$\rightarrow \langle X_n \dots X_4 X_2, X_n \dots X_4 X_2 \rangle$$

- 胶水规则大大提高了系统的健壮性(即使没有复杂规则，翻译可以顺序拼接)，也体现了语言翻译单调性的假设

文法驱动的机器翻译流程

- 同步翻译文法给我们带来了新的思路：可以通过不断使用语法规则完成翻译过程。这类模型的基本流程如下：



层次短语规则抽取

- 如何抽取规则
 - ▶ 首先进行短语抽取，从词对齐矩阵中抽取得到所有与词对齐保持一致的短语对

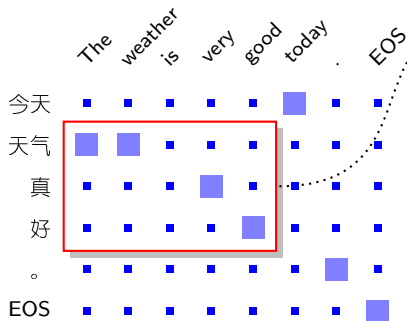
	The	weather	is	very	good	today	.	EOS
今天	■	■	■	■	■	■	■	■
天气	■	■	■	■	■	■	■	■
真	■	■	■	■	■	■	■	■
好	■	■	■	■	■	■	■	■
。	■	■	■	■	■	■	■	■
EOS	■	■	■	■	■	■	■	■

抽取得到的短语:

抽取得到的规则:

层次短语规则抽取

- 如何抽取规则
 - ▶ 首先进行短语抽取，从词对齐矩阵中抽取得到所有与词对齐保持一致的短语对



抽取得到的短语:

天气真好 – The weather is very good

抽取得到的规则:

层次短语规则抽取

- 如何抽取规则
 - ▶ 首先进行短语抽取，从词对齐矩阵中抽取得到所有与词对齐保持一致的短语对
 - ▶ 在抽取得到的短语中，找到其中包含的子短语，使用非终结符进行替换，就得到了一条规则

	The	weather	is	very	good	today	.	EOS
今天	■	■	■	■	■	■	■	■
天气	■	■	■	■	■	■	■	■
真	■	■	■	■	■	■	■	■
好	■	■	■	■	■	■	■	■
。	■	■	■	■	■	■	■	■
EOS	■	■	■	■	■	■	■	■

抽取得到的短语:

天气真好 – The weather is very good

抽取得到的规则:

X_1 真好 – X_1 is very good

层次短语规则抽取

- 如何抽取规则
 - ▶ 首先进行短语抽取，从词对齐矩阵中抽取得到所有与词对齐保持一致的短语对
 - ▶ 在抽取得到的短语中，找到其中包含的子短语，使用非终结符进行替换，就得到了一条规则

	The	weather	is	very	good	today	.	EOS
今天	■	■	■	■	■	■	■	■
天气	■	■	■	■	■	■	■	■
真	■	■	■	■	■	■	■	■
好	■	■	■	■	■	■	■	■
。	■	■	■	■	■	■	■	■
EOS	■	■	■	■	■	■	■	■

抽取得到的短语:

天气真好 – The weather is very good

天气 – The weather is

天气真 – The weather is very

...

抽取得到的规则:

X_1 真好 – X_1 is very good

天气真 X_1 – The weather is very X_1

X_1 真 X_2 – X_1 is very X_2

...

文法的约束

- 按照上面的方法可以抽取到大量的规则，规则太多会降低训练和解码的效率，甚至影响翻译性能，因此需要加入一些约束来限制文法规则的数目

$X \rightarrow \langle X_1 X_2 \text{ 之一, one of } X_1 X_2 \rangle$

$X \rightarrow \langle X_1 X_2 \text{ 是 } X_3, X_1 X_2 \text{ is } X_3 \rangle$

$X \rightarrow \langle X_1 \text{ 希望... } X_2, \underbrace{X_1 \text{ wish to... } X_2}_{\text{超过10个词}} \rangle$

文法的约束

- 按照上面的方法可以抽取到大量的规则，规则太多会降低训练和解码的效率，甚至影响翻译性能，因此需要加入一些约束来限制文法规则的数目

~~$X \rightarrow \langle X_1 X_2 \text{ 之一, one of } X_1 X_2 \rangle$~~

~~$X \rightarrow \langle X_1 X_2 \text{ 是 } X_3, X_1 X_2 \text{ is } X_3 \rangle$~~

~~$X \rightarrow \langle X_1 \text{ 希望... } X_2, X_1 \text{ wish to... } X_2 \rangle$~~

超过10个词

具体包含如下约束

- 1 规则中的非终结符不可以连续的出现
- 2 每条规则最多包含两个非终结符
- 3 抽取规则最多可以跨越10个单词

文法的约束

- 按照上面的方法可以抽取到大量的规则，规则太多会降低训练和解码的效率，甚至影响翻译性能，因此需要加入一些**约束**来限制文法规则的数目

~~$X \rightarrow \langle X_1 X_2 \text{ 之一, one of } X_1 X_2 \rangle$~~

~~$X \rightarrow \langle X_1 X_2 \text{ 是 } X_3, X_1 X_2 \text{ is } X_3 \rangle$~~

~~$X \rightarrow \langle X_1 \text{ 希望... } X_2, X_1 \text{ wish to... } X_2 \rangle$~~

超过10个词

具体包含如下约束

- ① 规则中的非终结符不可以连续的出现
 - ② 每条规则最多包含两个非终结符
 - ③ 抽取规则最多可以跨越10个单词
- 除此之外，不同的语言会有不同的文法约束，可参考
Hierarchical Phrase-Based Translation
Chiang, Computational Linguistics, 2007

特征

- 与短语模型一样，层次短语模型也使用判别式模型进行建模 - $P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$ 。其中特征权重 $\{\lambda_i\}$ 可以使用最小错误率训练进行调优，特征函数 $\{h_i\}$ 需要用户定义。

特征

- 与短语模型一样，层次短语模型也使用判别式模型进行建模 - $P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$ 。其中特征权重 $\{\lambda_i\}$ 可以使用最小错误率训练进行调优，特征函数 $\{h_i\}$ 需要用户定义。
- 这里，所有层次短语规则满足LHS $\rightarrow \langle \alpha, \beta, \sim \rangle$ 的形式
 - ▶ α 和 β 表示源语和目标语的规则串， \sim 表示他们的对应关系

特征

- 与短语模型一样，层次短语模型也使用判别式模型进行建模 - $P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$ 。其中特征权重 $\{\lambda_i\}$ 可以使用最小错误率训练进行调优，特征函数 $\{h_i\}$ 需要用户定义。
- 这里，所有层次短语规则满足 $LHS \rightarrow \langle \alpha, \beta, \sim \rangle$ 的形式
 - ▶ α 和 β 表示源语和目标语的规则串， \sim 表示他们的对应关系
- 特征1-2：短语翻译概率**，即正向翻译概率 $\log(P(\alpha|\beta))$ 和反向翻译概率 $\log(P(\beta|\alpha))$ 。这里， α 和 β 都被看做短语，因此可以直接复用短语系统的方法，使用极大似然估计进行计算。
- 特征3-4：词汇翻译概率**，即正向词汇翻译概率 $\log(P_{lex}(\alpha|\beta))$ 和反向词汇翻译概率 $\log(P_{lex}(\beta|\alpha))$ 。用来描述短语对中源语端单词和目标语端单词的对应关系

特征(续)

- **特征5：** n -gram语言模型，即 $\log(P_{lm}(\mathbf{t}))$ 。度量译文的流畅度，可以使用大规模目标语单语数据得到。
- **特征6：** 译文长度，即 $|\mathbf{t}|$ 。避免模型倾向于短译文，同时让系统自动学习对译文长度的偏好。

特征(续)

- **特征5：** n -gram语言模型，即 $\log(P_{lm}(t))$ 。度量译文的流畅度，可以使用大规模目标语单语数据得到。
- **特征6：** 译文长度，即 $|t|$ 。避免模型倾向于短译文，同时让系统自动学习对译文长度的偏好。
- **特征7：** 翻译规则数量。这个特征是为了避免模型仅仅使用少量特征构成翻译推导(因为翻译概率相乘，因子少结果一般会大一些)，同时让系统自动学习对使用规则数量的偏好。
- **特征8：** 胶水规则数量。这个特征是为了让系统可以控制使用胶水规则的偏好。

CKY解码

- 基于层次短语的翻译解码与基于短语的模型类似，都是要找到使 $\text{score}(d)$ 达到最大的翻译推导 d

$$\hat{d} = \arg \max_{d \in D} \text{score}(d)$$

- 由于翻译推导由SCFG构成，使用CKY算法进行解码
- CKY算法解码是一个用来判定任意给定的字符串 是否属于一个上下文无关文法的算法，具体流程如下



CKY算法

- CKY算法通过遍历不同span来判断字符串是否符合文法
 - 输入：源语串 $\mathbf{s} = s_1 \dots s_J$ ，以及CNF文法 G
 - 输出：判断字符串是否符合 G

Function CKY-Algorithm(\mathbf{s}, G)

for $j = 0$ to $J - 1$

$\text{span}[j, j + 1].\text{Add}(A \rightarrow a \in G)$

for $l = 1$ to J // length of span

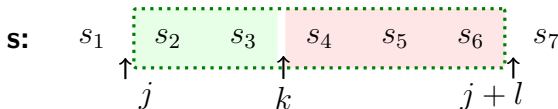
for $j = 0$ to $J - l$ // beginning of span

for $k = j$ to $j + l$ // partition of span

$\text{hypos} = \text{Compose}(\text{span}[j, k], \text{span}[k, j + l])$

$\text{span}[j, j + l].\text{Update}(\text{hypos})$

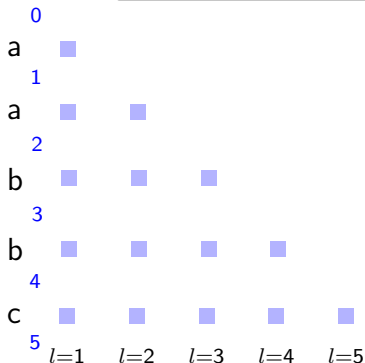
return $\text{span}[0, J]$



CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

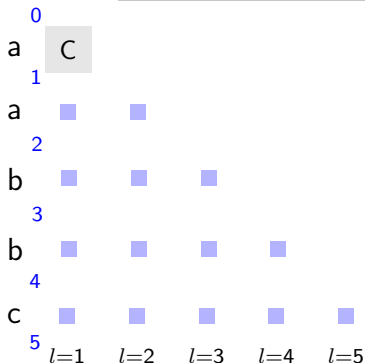


序号	跨度	推导
----	----	----

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

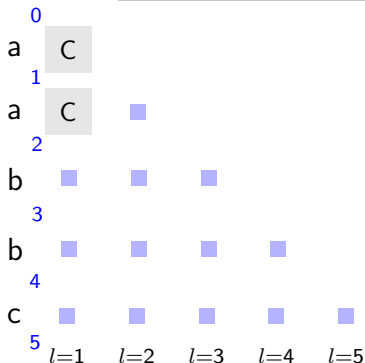


序号	跨度	推导
1	[0,1]	$C \rightarrow a$

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

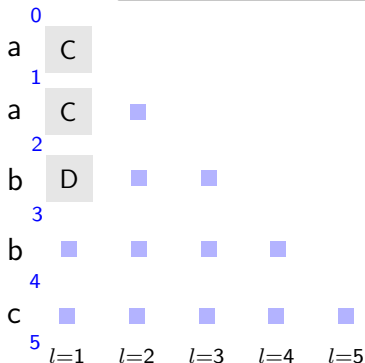


序号	跨度	推导
1	[0,1]	$C \rightarrow a$
2	[1,2]	$C \rightarrow a$

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

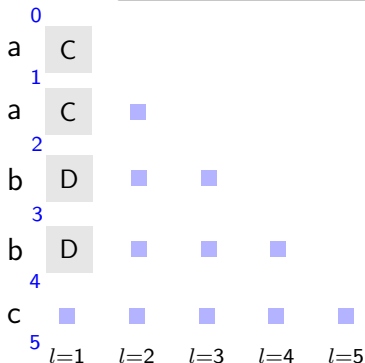


序号	跨度	推导
1	[0,1]	$C \rightarrow a$
2	[1,2]	$C \rightarrow a$
3	[2,3]	$D \rightarrow b$

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbcc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$



序号	跨度	推导
1	[0,1]	$C \rightarrow a$
2	[1,2]	$C \rightarrow a$
3	[2,3]	$D \rightarrow b$
4	[3,4]	$D \rightarrow b$

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbcc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

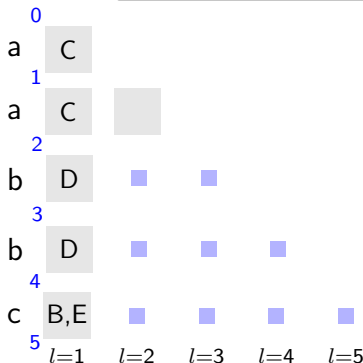
0					
a	C				
1					
a	C	■			
2					
b	D	■	■		
3					
b	D	■	■	■	
4					
c	B,E	■	■	■	■
5					
	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$

序号	跨度	推导
1	[0,1]	$C \rightarrow a$
2	[1,2]	$C \rightarrow a$
3	[2,3]	$D \rightarrow b$
4	[3,4]	$D \rightarrow b$
5	[4,5]	$B \rightarrow c, E \rightarrow c$

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

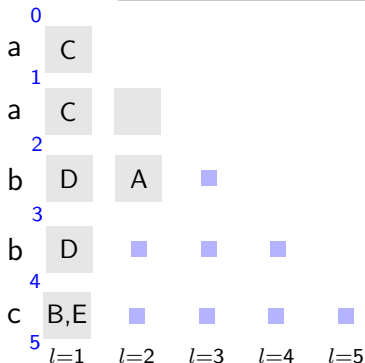


序号	跨度	推导
1	[0,1]	$C \rightarrow a$
2	[1,2]	$C \rightarrow a$
3	[2,3]	$D \rightarrow b$
4	[3,4]	$D \rightarrow b$
5	[4,5]	$B \rightarrow c, E \rightarrow c$
6	[0,2]	none

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$



序号	跨度	推导
1	[0,1]	$C \rightarrow a$
2	[1,2]	$C \rightarrow a$
3	[2,3]	$D \rightarrow b$
4	[3,4]	$D \rightarrow b$
5	[4,5]	$B \rightarrow c, E \rightarrow c$
6	[0,2]	none
7	[1,3]	$A \rightarrow CD$

CKY算法

- 我们来看一个CKY算法的具体例子，给定一个上下无关文法以及一个单词 **aabbcc**，来判断该单词是否属于此文法，解析流程如下

$S \rightarrow AB \quad A \rightarrow CD \mid CF \quad B \rightarrow c \mid BE$
 $C \rightarrow a \quad D \rightarrow b \quad E \rightarrow c \quad F \rightarrow AD$

0					
a	C				
1					
a	C				
2					
b	D	A			
3					
b	D		F	A	
4					
c	B,E				S
5					
	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$

序号	跨度	推导
1	$[0,1]$	$C \rightarrow a$
2	$[1,2]$	$C \rightarrow a$
3	$[2,3]$	$D \rightarrow b$
4	$[3,4]$	$D \rightarrow b$
5	$[4,5]$	$B \rightarrow c, E \rightarrow c$
6	$[0,2]$	none
7	$[1,3]$	$A \rightarrow CD$
...		
15	$[0,5]$	$S \rightarrow AB$

CKY解码（续）

- 实际上，在层次短语解码的时候，不能直接使用CKY算法，需要先转化为乔姆斯基范式，才能进行解码

CKY解码（续）

- 实际上，在层次短语解码的时候，不能直接使用CKY算法，需要先转化为乔姆斯基范式，才能进行解码
 - 对于每个源语句，使用短语规则表初始化它的span

进口

Imports
The imports
...

1

大幅度

drastically
substantially
...

2

下降

fall
fallen
...

3

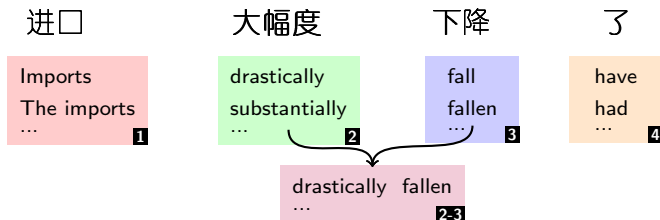
了

have
had
...

4

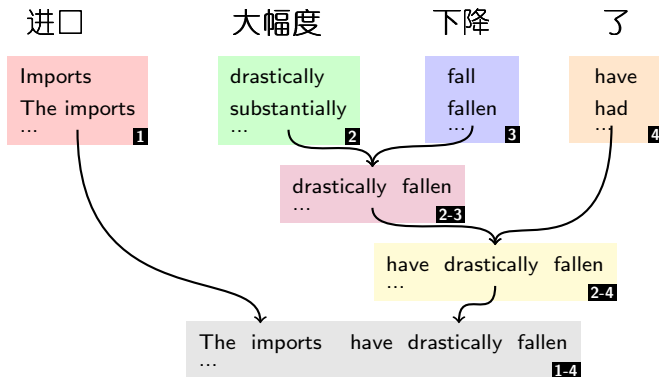
CKY解码（续）

- 实际上，在层次短语解码的时候，不能直接使用CKY算法，需要先转化为乔姆斯基范式，才能进行解码
 - 对于每个源语句句子，使用短语规则表初始化它的span
 - 自底向上对span中的每个子span进行重新组合（正、反向）



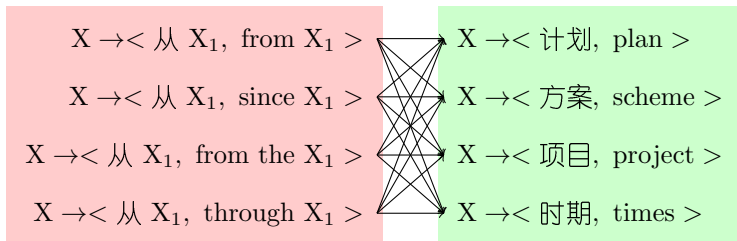
CKY解码（续）

- 实际上，在层次短语解码的时候，不能直接使用CKY算法，需要先转化为乔姆斯基范式，才能进行解码
 - 对于每个源语句句子，使用短语规则表初始化它的span
 - 自底向上对span中的每个子span进行重新组合（正、反向）
 - 计算每个推导的得分并记录下来，最终选择最优推导所对应的译文作为输出



立方剪枝(Cube Pruning)

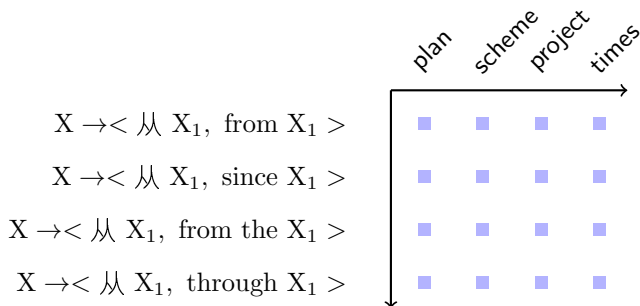
- 前面介绍的解码方法由于搜索空间非常大，速度很慢，通常使用剪枝的方法来加速这个过程
 - 解码时对来自不同的cell进行合并时，第一个cell包含 n 个条目，第二个cell包含 m 个条目，就会产生 $n \times m$ 个新条目



- 当 n 和 m 都很大时，会产生很多新的条目，可以通过限制栈的大小来舍弃大部分条目
- 另一个思路就是如何只生成有机会被选中的条目，这就是立方剪枝 (cube pruning) 的思想

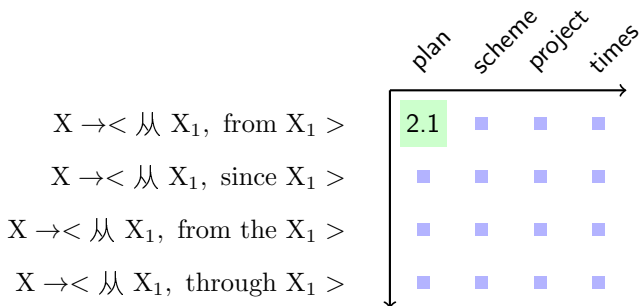
立方剪枝(Cube Pruning)(续)

- 不是考虑所有可能的组合，而是从最优候选开始选择
 - 根据代价估计对每个cell中的条目进行排序，最好的候选在最顶层，次优候选排在后面



立方剪枝(Cube Pruning)(续)

- 不是考虑所有可能的组合，而是从**最优**候选开始选择
 - 根据代价估计对每个cell中的条目进行排序，最好的候选在最顶层，次优候选排在后面
 - 计算所有邻居的代价，选择最优的条目，再计算其邻居的代价，不断进行迭代



立方剪枝(Cube Pruning)(续)

- 不是考虑所有可能的组合，而是从**最优**候选开始选择
 - 根据代价估计对每个cell中的条目进行排序，最好的候选在最顶层，次优候选排在后面
 - 计算所有邻居的代价，选择最优的条目，再计算其邻居的代价，不断进行迭代
 - 直到产生的条目达到一定的数量（如栈容量）或者新加入的条目被其他剪枝策略丢弃，则终止算法

	plan	scheme	project	times
$X \rightarrow < \text{从 } X_1, \text{ from } X_1 >$	2.1	5.1	■	■
$X \rightarrow < \text{从 } X_1, \text{ since } X_1 >$	5.5	■	■	■
$X \rightarrow < \text{从 } X_1, \text{ from the } X_1 >$	■	■	■	■
$X \rightarrow < \text{从 } X_1, \text{ through } X_1 >$	■	■	■	■

立方剪枝(Cube Pruning)(续)

- 不是考虑所有可能的组合，而是从**最优**候选开始选择
 - 根据代价估计对每个cell中的条目进行排序，最好的候选在最顶层，次优候选排在后面
 - 计算所有邻居的代价，选择最优的条目，再计算其邻居的代价，不断进行迭代
 - 直到产生的条目达到一定的数量（如栈容量）或者新加入的条目被其他剪枝策略丢弃，则终止算法

$X \rightarrow < \text{从 } X_1, \text{ from } X_1 >$

$X \rightarrow < \text{从 } X_1, \text{ since } X_1 >$

$X \rightarrow < \text{从 } X_1, \text{ from the } X_1 >$

$X \rightarrow < \text{从 } X_1, \text{ through } X_1 >$

	plan	scheme	project	times
	2.1	5.1	■	■
	5.5	8.5	■	■
	7.7	■	■	■
	■	■	■	■

立方剪枝(Cube Pruning)(续)

- 不是考虑所有可能的组合，而是从**最优**候选开始选择
 - 根据代价估计对每个cell中的条目进行排序，最好的候选在最顶层，次优候选排在后面
 - 计算所有邻居的代价，选择最优的条目，再计算其邻居的代价，不断进行迭代
 - 直到产生的条目达到一定的数量（如栈容量）或者新加入的条目被其他剪枝策略丢弃，则终止算法

$X \rightarrow < \text{从 } X_1, \text{ from } X_1 >$
 $X \rightarrow < \text{从 } X_1, \text{ since } X_1 >$
 $X \rightarrow < \text{从 } X_1, \text{ from the } X_1 >$
 $X \rightarrow < \text{从 } X_1, \text{ through } X_1 >$

	plan	scheme	project	times
	2.1	5.1	■	■
	5.5	8.5	4.2	■
	7.7	8.2	■	■
	■	■	■	■

效果

- 从实验结果中可以看出，基于层次短语的翻译模型性能要优于基于短语的翻译模型
- 选择使用层次短语信息实际上增加了模型的复杂度，但是可以通过借鉴基于短语的翻译模型模型以及CKY解码和立方剪枝等技术来解决
- 可以考虑加入更多句法信息来进一步提升模型性能

模型	开发集 (BLEU[%])	测试集 (BLEU[%])
短语(Moses)	36.51	34.93
短语(NiuTrans)	36.99	35.29
层次短语(Moses)	36.65	34.79
层次短语(NiuTrans)	37.41	35.35

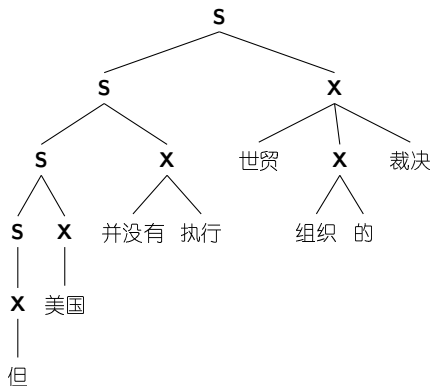
* 以上结果来自 NiuTrans: An Open Source Toolkit for Phrase-based Machine Translation

* 开发集：NIST MT03，测试集：NIST MT05

基于语言学句法的翻译模型

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

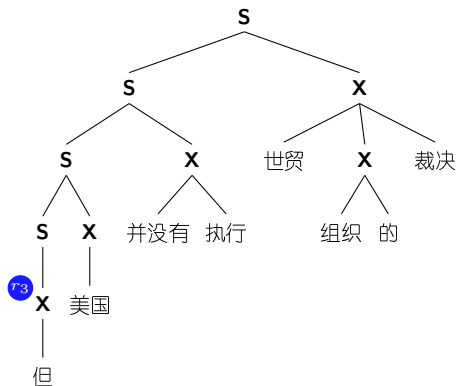
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3$$



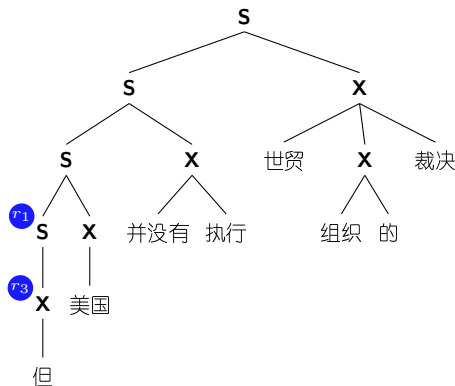
层次短语翻译规则:

- $r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$
 $r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$
 $r_3 \quad X \rightarrow \langle \text{但, but} \rangle$
 $r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$
 $r_5 \quad X \rightarrow \langle \text{并没有 执行,}$
 $\qquad \qquad \text{has not implemented} \rangle$
 $r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,}$
 $\qquad \qquad \text{the decision } X_1 \text{ the WTO} \rangle$
 $r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

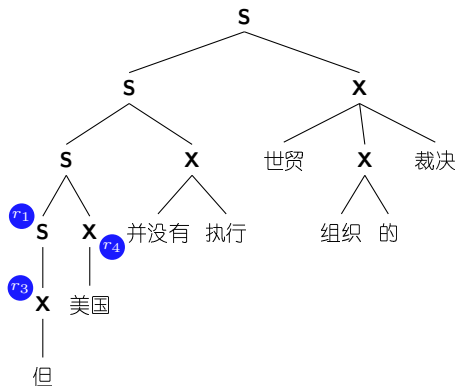
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4$$



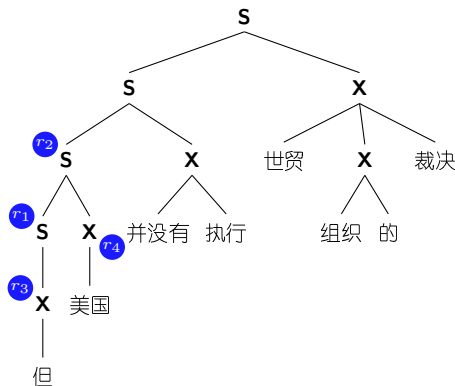
层次短语翻译规则:

- $r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$
 $r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$
 $r_3 \quad X \rightarrow \langle \text{但, but} \rangle$
 $r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$
 $r_5 \quad X \rightarrow \langle \text{并没有 执行,}$
 $\qquad \qquad \qquad \text{has not implemented} \rangle$
 $r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,}$
 $\qquad \qquad \qquad \text{the decision } X_1 \text{ the WTO} \rangle$
 $r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4 \circ r_2$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

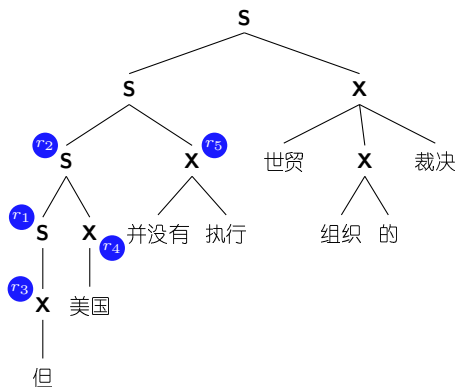
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4 \circ r_2 \circ r_5$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

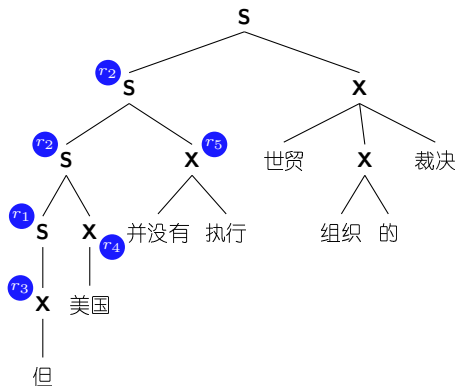
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4 \circ r_2 \circ r_5 \circ r_2$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

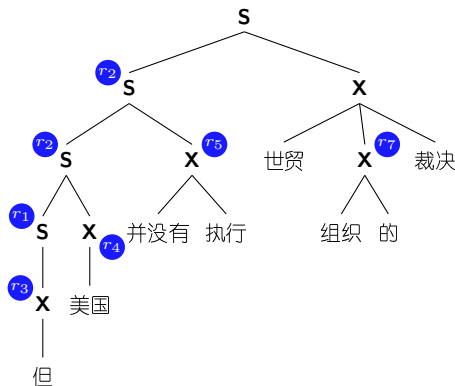
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4 \circ r_2 \circ r_5 \circ r_2 \circ r_7$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

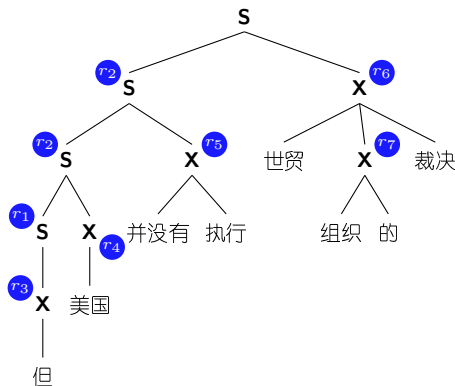
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4 \circ r_2 \circ r_5 \circ r_2 \circ r_7 \circ r_6$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

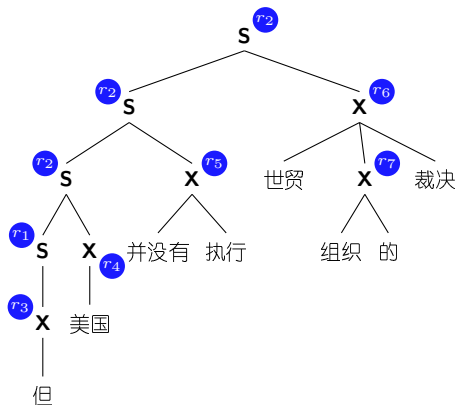
$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

层次短语规则推导的句法树表示

- 层次短语规则的使用本质上是对(源语)句子建立树状结构, 这反应了文法对于字符串分析得到的推导

$$d = r_3 \circ r_1 \circ r_4 \circ r_2 \circ r_5 \circ r_2 \circ r_7 \circ r_6 \circ r_2$$



层次短语翻译规则:

$r_1 \quad S \rightarrow \langle X_1, X_1 \rangle$

$r_2 \quad S \rightarrow \langle S_1 X_2, S_1 X_2 \rangle$

$r_3 \quad X \rightarrow \langle \text{但, but} \rangle$

$r_4 \quad X \rightarrow \langle \text{美国, the U.S.} \rangle$

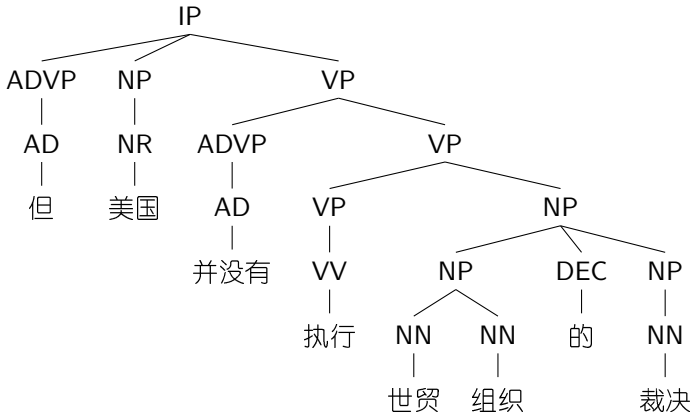
$r_5 \quad X \rightarrow \langle \text{并没有 执行,} \\ \text{has not implemented} \rangle$

$r_6 \quad X \rightarrow \langle \text{世贸 } X_1 \text{ 裁决,} \\ \text{the decision } X_1 \text{ the WTO} \rangle$

$r_7 \quad X \rightarrow \langle \text{组织 的, of} \rangle$

语言学句法树

- 但是，层次短语的分析过程还是一种形式文法的推导，另一种自然的想法是把句子表示成语言学上的句法树，这种表示更符合人类对语言的认知
 - 比如，“执行”后面接名词短语NP
 - 还有，“并没有”作用于后面的动词短语“执行...”



使用句法的好处

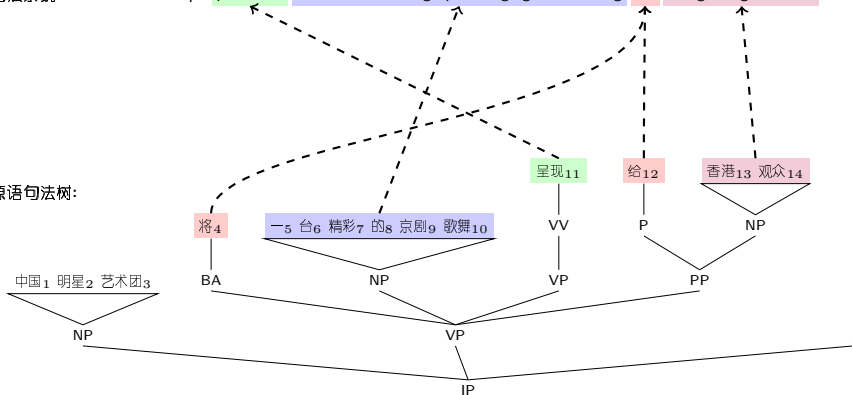
- 短语结构树可以仍容易进行远距离调序的建模，同时丰富的句法功能标记有助于翻译中的消歧和生成

参考答案: The Chinese star performance troupe presented a wonderful Peking opera as well as singing and dancing performance to Hong Kong audience .

层次短语系统: Star troupe of China, highlights of Peking opera and dance show to the audience of Hong Kong .

句法系统: Chinese star troupe presented a wonderful Peking opera singing and dancing to Hong Kong audience .

源语句法树:



使用句法的好处

- 短语结构树可以仍容易进行远距离调序的建模，同时丰富的句法功能标记有助于翻译中的消歧和生成

参考答案: The Chinese star performance troupe presented a wonderful Peking opera as well as singing and dancing performance to Hong Kong audience .

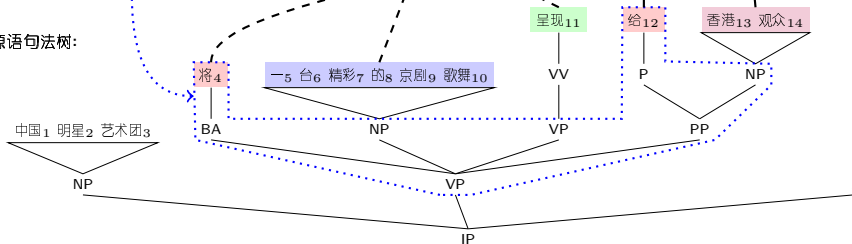
层次短语系统: Star troupe of China, highlights of Peking opera and dance show to the audience of Hong Kong .

句法系统: Chinese star troupe presented a wonderful Peking opera singing and dancing to Hong Kong audience .

句法翻译规则:

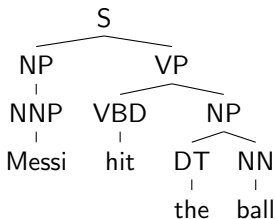
$$(\text{VP BA(将)} x_1:\text{NP } x_2:\text{VP PP(P(给)} x_3:\text{NP}))$$
$$\rightarrow x_2 \ x_1 \text{ to } x_3$$

源语句法树:

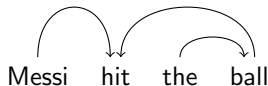


有很多句法知识可以使用

- 句法树形式选择 - 句法的表示形式有很多，侧重点也不同，机器翻译需要何种句法信息？



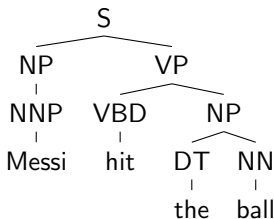
(a) 短语结构树



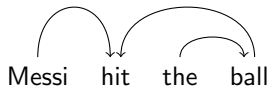
(b) 依存树

有很多句法知识可以使用

- 句法树形式选择 - 句法的表示形式有很多，侧重点也不同，机器翻译需要何种句法信息？



(a) 短语结构树



(b) 依存树

- 如何获取句法知识 - 句法信息由谁提供？
 - 自动句法分析器可以提供句法信息，分析器可以从树库中自动学习(回忆第二章)，因此得到的句法分析结果和树库标注是一致的
 - 不使用句法分析器，而是让机器翻译自动学习适合翻译任务的句法结构

术语

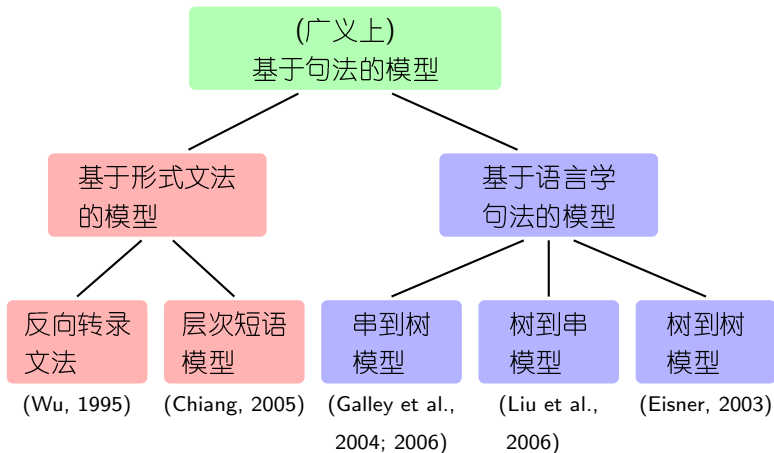
- 这里我们假设：
 - ▶ 使用短语结构树表示句法信息
 - ▶ 句法树由句法分析器自动生成
- 涉及的术语

术语	说明
翻译规则	翻译的最小单元(或步骤)
推导	由一系列规则组成的分析或翻译过程，推导可以被看作是规则的序列
规则表	翻译规则的存储表示形式，可以高效进行查询
层次短语模型	基于同步上下文无关文法的翻译模型，非终结符只有S和X两种，文法并不需要符合语言学句法约束
树到串模型	一类翻译模型，它使用源语语言学句法树，因此翻译可以被看作是从一棵句法树到词串的转换
串到树模型	一类翻译模型，它使用目标语语言学句法树，因此翻译可以被看作是从词串到句法树的转换

术语(续)

术语	说明
树到树模型	一类翻译模型，它同时使用源语和目标语语言学句法树，因此翻译可以被看做从句法树到句法树的转换
基于句法	使用语言学句法
基于树	(源语言)使用树结构(大多指句法树)
基于串	(源语言)使用词串，比如串到树的翻译系统的解码器一般都是基于串的解码方法
基于森林	(源语言)使用句法森林，这里森林只是对多个句法树的一种压缩表示
词汇化规则	含有终结符的规则
非词汇规则	不含有终结符的规则
句法约软束	不强制规则推导匹配语言学句法树，通常把句法信息作为特征使用
句法硬约束	强制推导必须符合语言学句法树，不符合的推导会被过滤掉

句法模型的分类



- 实际上，上面仅仅只是一种分类方法，还有很多其它分类标准，比如：句法软约束 vs 句法硬约束，基于树 vs. 基于串，等等

句法模型对比

	形式句法	语言学句法		
		树到串	串到树	树到树
源语句法	No	Yes	No	Yes
目标语句法	No	No	Yes	Yes
基于串的解码	Yes	No	Yes	Yes
基于树的解码	No	Yes	No	Yes
健壮性	High	Mid	Mid	Low

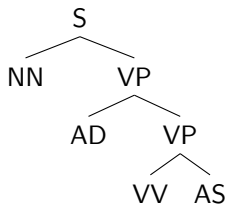
句法模型对比

	形式句法	语言学句法		
		树到串	串到树	树到树
源语句法	No	Yes	No	Yes
目标语句法	No	No	Yes	Yes
基于串的解码	Yes	No	Yes	Yes
基于树的解码	No	Yes	No	Yes
健壮性	High	Mid	Mid	Low

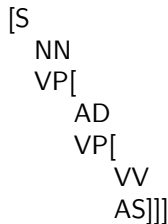
- 下面，以树到串/串到树为例进行介绍，一些代表性论文
 - ▶ **What's in a translation rule?**
Galley et al., 2004, In Proc. of HLT-NAACL
 - ▶ **Scalable Inference and Training of Context-Rich Syntactic Translation Models**
Galley et al., 2006, In Proc. of ACL
 - ▶ **Tree-to-String Alignment Template for Statistical Machine Translation**
Liu et al., 2006, In Proc. of ACL

树结构的表示

- 基于句法的翻译模型核心是对**树结构**进行建模。对于树到串和串到树模型，本质上是找到树和串的对应关系；而对于树到树模型，本质上是找到树到树的对应



(a) 树状表示



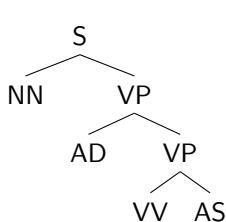
(b) 序列表示(缩进)

(S NN VP(AD
VP(VV AS)))

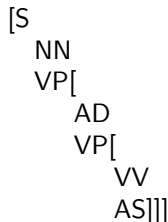
(c) 序列表示

树结构的表示

- 基于句法的翻译模型核心是对**树结构**进行建模。对于树到串和串到树模型，本质上是找到树和串的对应关系；而对于树到树模型，本质上是找到树到树的对应



(a) 树状表示



(b) 序列表示(缩进)

(S NN VP(AD
VP(VV AS)))

(c) 序列表示

- 通常，可以用基于树结构的翻译规则来描述上述过程，有两种情况：

- 1 树到串翻译规则 - 对应树到串、串到树模型
- 2 树到树翻译规则 - 对应树到树模型

这里用一种统一的形式描述上述规则

基于树结构的文法

- 为了描述任意树和串之间的转换，可以定义如下文法

定义 - 基于树结构的文法

一个基于树结构的文法由七部分构成($N_s, N_t, T_s, T_t, I_s, I_t, R$)，其中

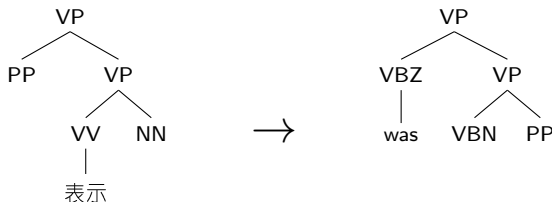
1. N_s 和 N_t 是源语和目标语非终结符集合
2. T_s 和 T_t 是源语言和目标语终结符集合
3. $I_s \subseteq N_s$ 和 $I_t \subseteq N_t$ 是源语言和目标语起始非终结符集合
4. R 是规则集合，每条规则 $r \in R$ 有如下形式

$$\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$$

其中，规则左部由非终结符 $\alpha_h \in N_s$ 和 $\beta_h \in N_t$ 构成；规则右部由三部分组成， α_r 表示由源语言终结符和非终结符组成的树结构； β_r 表示由目标语言终结符和非终结符组成的树结构； \sim 表示 α_r 和 β_r 中叶子非终结符的1-1对应关系

基于树的翻译规则

- 上述文法定义了一种树结构到树结构的映射，例子：



- 由规则定义 $\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ 知道

$$\langle \alpha_h, \beta_h \rangle = \langle \text{VP}, \text{VP} \rangle$$

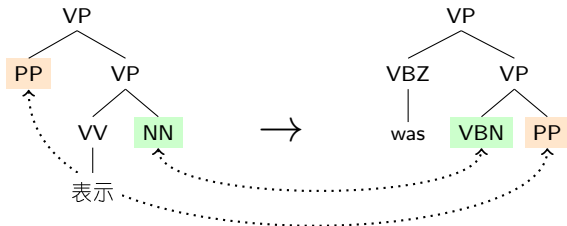
$$\alpha_r = \text{VP}(\text{PP}:x \text{ VP}(\text{VV}(\text{表示}) \text{ NN}:x))$$

$$\beta_r = \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}:x \text{ PP}:x))$$

$$\sim = \{1 - 2, 2 - 1\}$$

基于树的翻译规则

- 上述文法定义了一种树结构到树结构的映射，例子：



- 由规则定义 $\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ 知道

$$\langle \alpha_h, \beta_h \rangle = \langle \text{VP}, \text{VP} \rangle$$

$$\alpha_r = \text{VP}(\text{PP}:x \text{ VP}(\text{VV}(\text{表示}) \text{ NN}:x))$$

$$\beta_r = \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}:x \text{ PP}:x))$$

$$\sim = \{1 - 2, 2 - 1\}$$

x 表示叶子非终结符(可替换的变量)，显然这是调序规则

基于树的翻译规则(续)

- 可以省略“~”，把规则重新写为易于计算机处理的格式

$$\langle \text{VP}, \text{VP} \rangle \rightarrow \langle \text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{NN}_2)), \\ \text{VP}(\text{VBZ}(\text{was}) \text{VP}(\text{VBN}_2 \text{PP}_1)) \rangle$$

其中变量的对应关系用下标数字表示，比如： $\text{PP}_1 \leftrightarrow \text{PP}_1$ ， $\text{NN}_2 \leftrightarrow \text{VBN}_2$

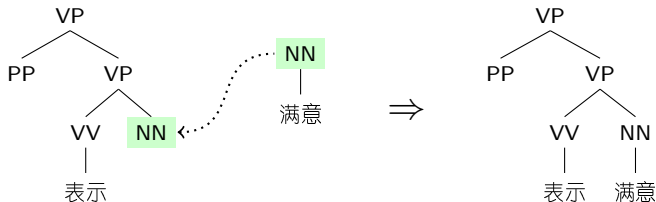
基于树的翻译规则(续)

- 可以省略“~”，把规则重新写为易于计算机处理的格式

$$\langle VP, VP \rangle \rightarrow \langle VP(PP_1 VP(VV(\text{表示}) NN_2)), \\ VP(VBZ(\text{was}) VP(VBN_2 PP_1)) \rangle$$

其中变量的对应关系用下标数字表示，比如： $PP_1 \leftrightarrow PP_1$ ， $NN_2 \leftrightarrow VBN_2$

- 在这个规则的树结构中，每个叶子非终结符本质上定义了一个变量，这个节点也被称作边缘节点(frontier node)。边缘节点可以被其它树结构替换，组合为更大的树结构，这个操作被称作组合(composition) 或树替换



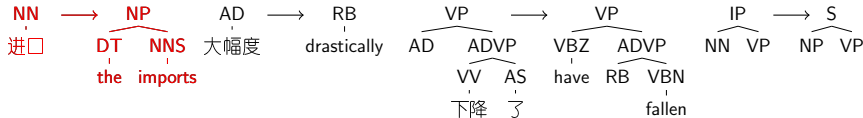
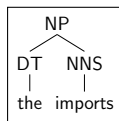
使用规则定义翻译过程

- 规则的推导描述双语句子同步生成(和分析)的过程

► ●●表示对变量的替换操作

源语言

目标语言



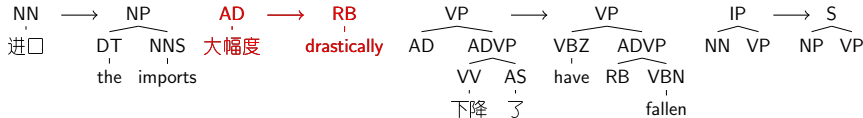
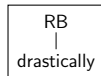
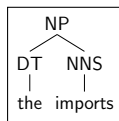
使用规则定义翻译过程

- 规则的推导描述双语句子同步生成(和分析)的过程

► ●●表示对变量的替换操作

源语言

目标语言



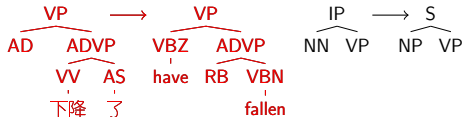
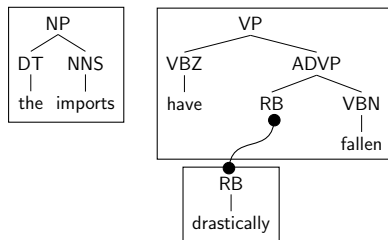
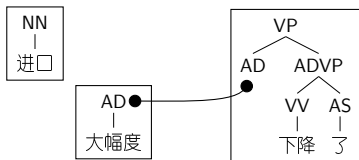
使用规则定义翻译过程

- 规则的推导描述双语句子同步生成(和分析)的过程

► ●●表示对变量的替换操作

源语言

目标语言

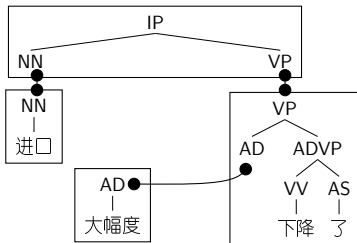


使用规则定义翻译过程

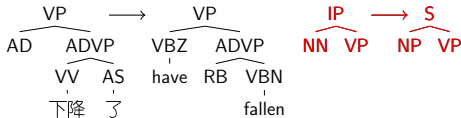
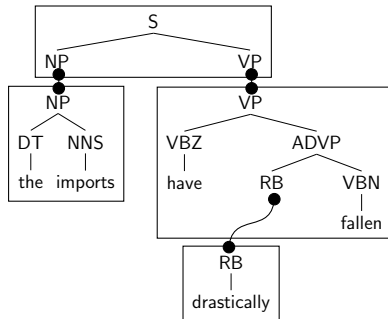
- 规则的推导描述双语句子同步生成(和分析)的过程

► ●●表示对变量的替换操作

源语言

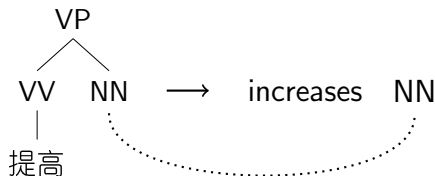


目标语言



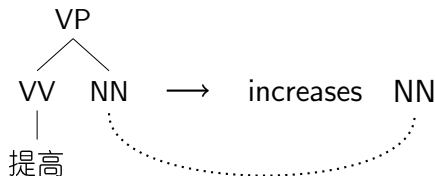
树到串翻译规则

- 对于只有一端使用句法树的情况，仍然可以用上述规则定义进行描述
 - ▶ 树到串翻译可以看做是句法树到词串的转换，串到树类似，只是反过来看



树到串翻译规则

- 对于只有一端使用句法树的情况，仍然可以用上述规则定义进行描述
 - ▶ 树到串翻译可以看做是句法树到词串的转换，串到树类似，只是反过来看



- 由规则定义 $\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ ，可得

$$\alpha_h = \text{VP}$$

$$\beta_h = \text{VP} (= \alpha_h)$$

$$\alpha_r = \text{VP}(\text{VV}(\text{提高}) \text{NN}:x)$$

$$\beta_r = \text{VP}(\text{increases NN}:x)$$

$$\sim = \{1 - 1\}$$

树到串翻译规则

- 这里，目标端是一个词串，因此
 - ▶ β_h 实际上并不是树的根节点标记，直接复制 α_h 即可，也就是说目标语端和源语端共享同一个非终结符集合
 - ▶ β_r 是一个词串，可以被看作是树的叶子节点序列，或者一个单层的树结构

树到串翻译规则

- 这里，目标端是一个词串，因此
 - ▶ β_h 实际上并不是树的根节点标记，直接复制 α_h 即可，也就是说目标语端和源语端共享同一个非终结符集合
 - ▶ β_r 是一个词串，可以被看作是树的叶子节点序列，或者一个单层的树结构
- 可以把这条规则简记为

$$VP \rightarrow \langle VP(VV(\text{提高}) NN_1), \text{increases } NN_1 \rangle$$

或

$$VP(VV(\text{提高}) NN_1) \rightarrow \text{increases } NN_1$$

上述规则也被称作树到串翻译规则

树到串翻译规则

- 这里，目标端是一个词串，因此
 - ▶ β_h 实际上并不是树的根节点标记，直接复制 α_h 即可，也就是说目标语端和源语端共享同一个非终结符集合
 - ▶ β_r 是一个词串，可以被看作是树的叶子节点序列，或者一个单层的树结构
- 可以把这条规则简记为

$$VP \rightarrow \langle VP(VV(\text{提高}) NN_1), \text{increases } NN_1 \rangle$$

或

$$VP(VV(\text{提高}) NN_1) \rightarrow \text{increases } NN_1$$

上述规则也被称作**树到串翻译规则**

- 类似的，层次短语规则也可以被看作是一种特殊的基于树结构的规则，它的源语和目标语都是由单层树结构构成，且源语和目标语共享同一个非终结符集合

树到串规则抽取 - GHKM方法

- 基于句法的翻译系统的核心有两个部分
 - ① **文法归纳**：从带有句法分析结果的双语数据中自动学习翻译规则
 - ② **解码**：使用学习到的翻译规则对新的句子进行翻译

树到串规则抽取 - GHKM方法

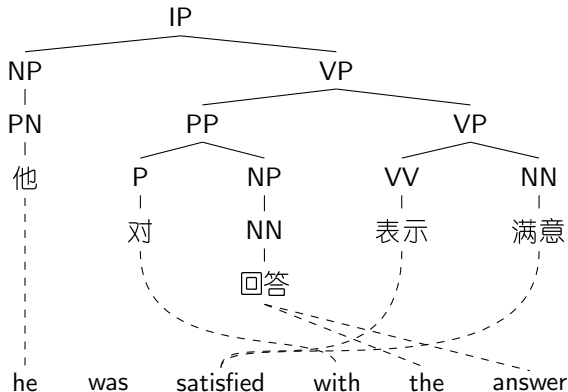
- 基于句法的翻译系统的核心有两个部分
 - ① **文法归纳**：从带有句法分析结果的双语数据中自动学习翻译规则
 - ② **解码**：使用学习到的翻译规则对新的句子进行翻译
- 首先要解决的是如何获取翻译规则，即**规则抽取** - 这里先从GHKM方法开始，它是经典的树到串翻译规则的抽取方法(Galley et al., 2004; 2006)
 - ▶ 方法的名字是由四位作者的名字首字母构成 :)

树到串规则抽取 - GHKM方法

- 基于句法的翻译系统的核心有两个部分
 - ① **文法归纳**：从带有句法分析结果的双语数据中自动学习翻译规则
 - ② **解码**：使用学习到的翻译规则对新的句子进行翻译
- 首先要解决的是如何获取翻译规则，即**规则抽取** - 这里先从GHKM方法开始，它是经典的树到串翻译规则的抽取方法(Galley et al., 2004; 2006)
 - ▶ 方法的名字是由四位作者的名字首字母构成：)
- GHKM方法的输入包括
 - ▶ 源语言句子和它的短语分析树
 - ▶ 目标语句子
 - ▶ 源语和目标语句子之间的词对齐
- 注意：
 - ▶ 句法树可以由句法分析器自动生成
 - ▶ 词对齐可以由词对齐系统(如IBM模型)自动生成

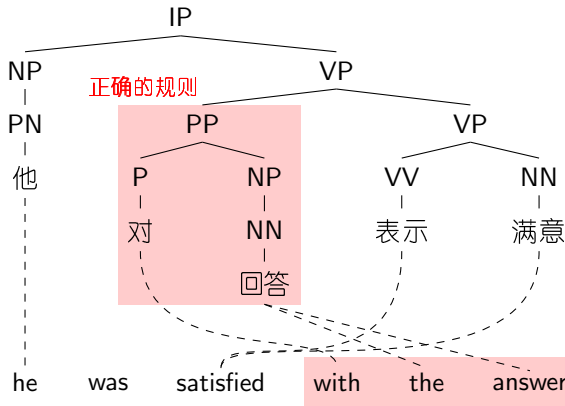
树片段

- 树到串翻译规则实际上是一个树片段到一个词串的映射。一个合理的树到串翻译规则，不应该违反任何的词对齐信息
 - ▶ 显然这种树片段可以有很多
 - ▶ 一棵句法树也可以被切割成多个树片段



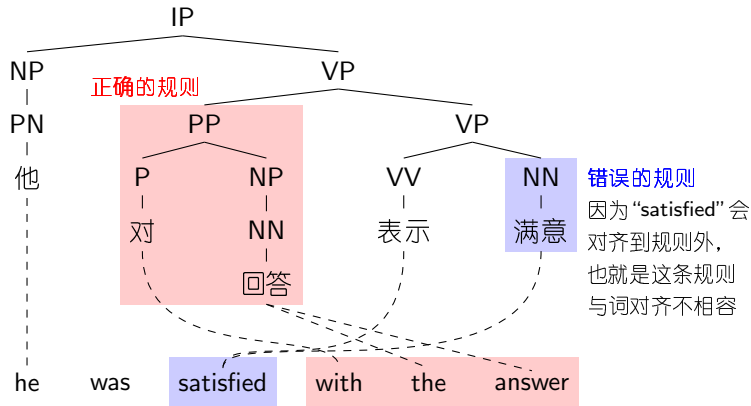
树片段

- 树到串翻译规则实际上是一个树片段到一个词串的映射。一个合理的树到串翻译规则，不应该违反任何的词对齐信息
 - ▶ 显然这种树片段可以有很多
 - ▶ 一棵句法树也可以被切割成多个树片段



树片段

- 树到串翻译规则实际上是一个树片段到一个词串的映射。一个合理的树到串翻译规则，不应该违反任何的词对齐信息
 - 显然这种树片段可以有很多
 - 一棵句法树也可以被切割成多个树片段

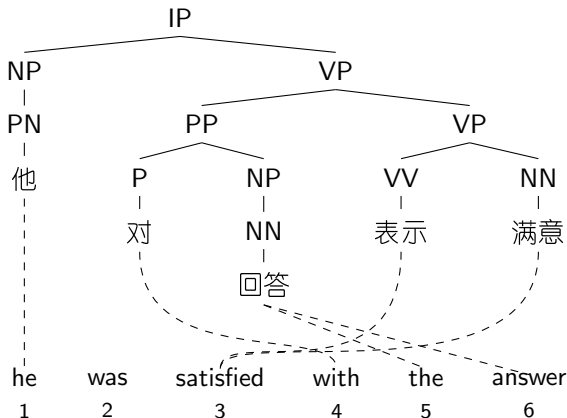


Span

- 为了描述每个树节点对齐到目标语的情况，定义概念

定义 - Span

对于一个源语言句法树节点，它的Span是这个节点所对应到目标语的第一个单词和最后一个单词所构成的索引范围

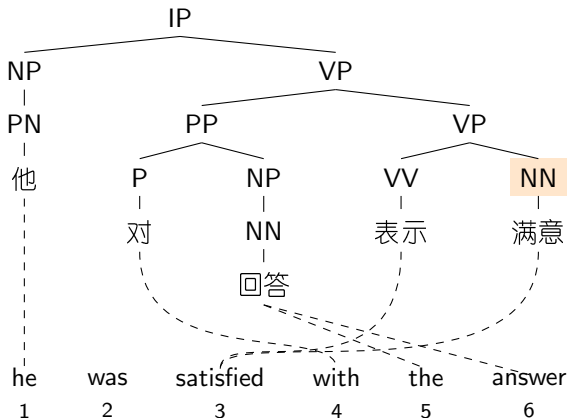


Span

- 为了描述每个树节点对齐到目标语的情况，定义概念

定义 - Span

对于一个源语言句法树节点，它的Span是这个节点所对应到目标语的第一个单词和最后一个单词所构成的索引范围



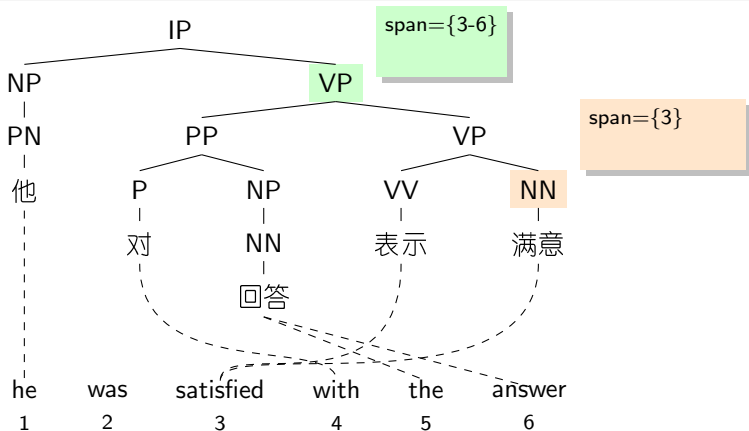
span={3}

Span

- 为了描述每个树节点对齐到目标语的情况，定义概念

定义 - Span

对于一个源语言句法树节点，它的Span是这个节点所对应到目标语的第一个单词和最后一个单词所构成的索引范围

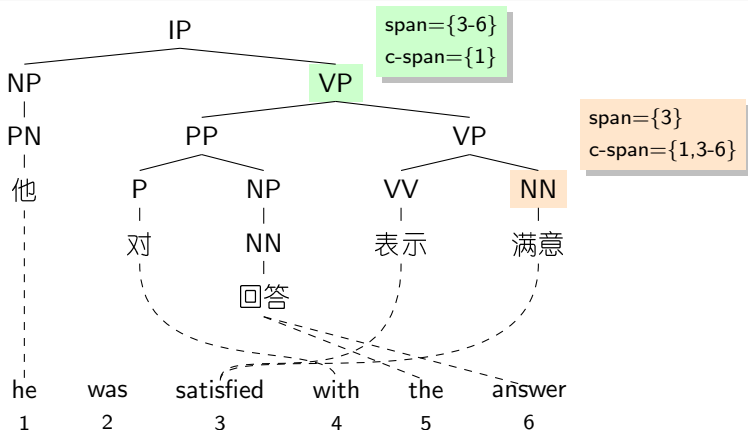


Complement Span

- 进一步，引入Complement Span的概念

定义 - Complement Span

对于一个源语言句法树节点，它的Complement Span是除了它的祖先和子孙节点外的其它节点Span的并集

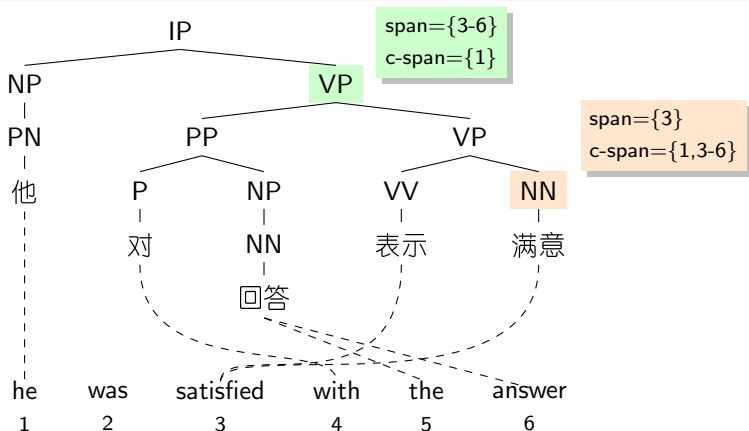


可信节点

- 基于Span和Complment Span, 可以定义可信节点)

定义 - 可信节点(Admissible Node)

对于源语言树节点 n , 如果它的Span和Complement Span不相交, 节点 n 就是一个可信节点, 否则是一个不可信节点

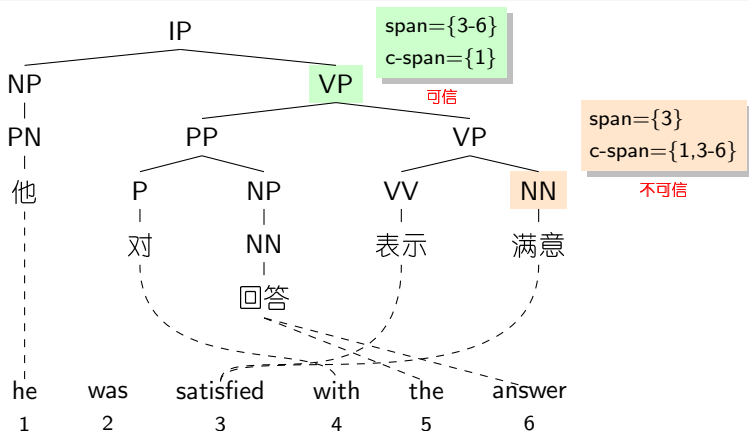


可信节点

- 基于Span和Complment Span, 可以定义可信节点)

定义 - 可信节点(Admissible Node)

对于源语言树节点 n , 如果它的Span和Complement Span不相交, 节点 n 就是一个可信节点, 否则是一个不可信节点

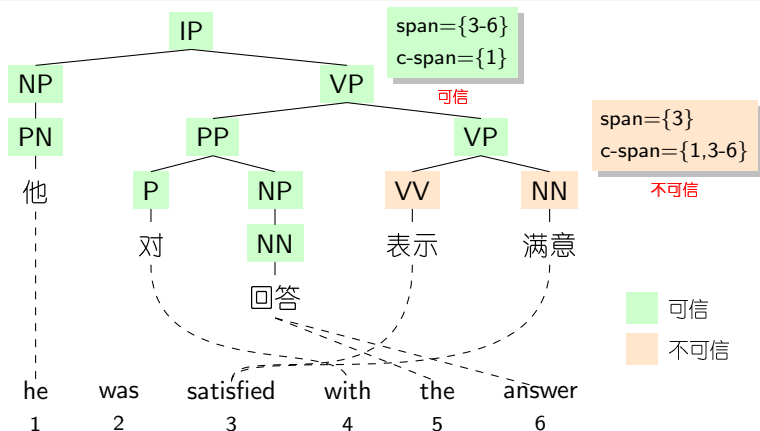


可信节点

- 基于Span和Complment Span, 可以定义可信节点)

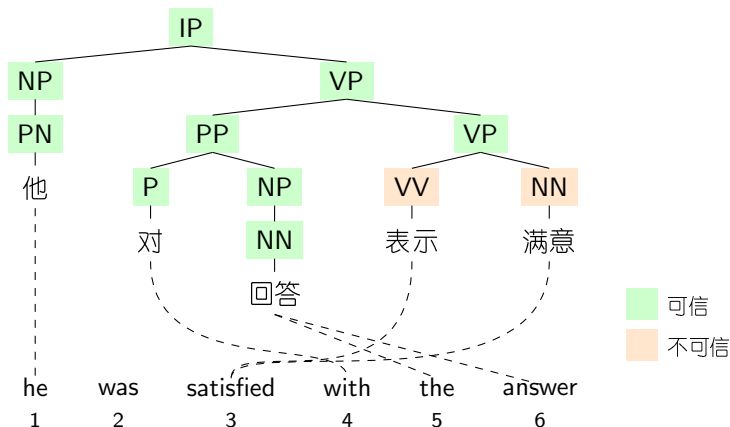
定义 - 可信节点(Admissible Node)

对于源语言树节点 n , 如果它的Span和Complement Span不相交, 节点 n 就是一个可信节点, 否则是一个不可信节点



规则抽取

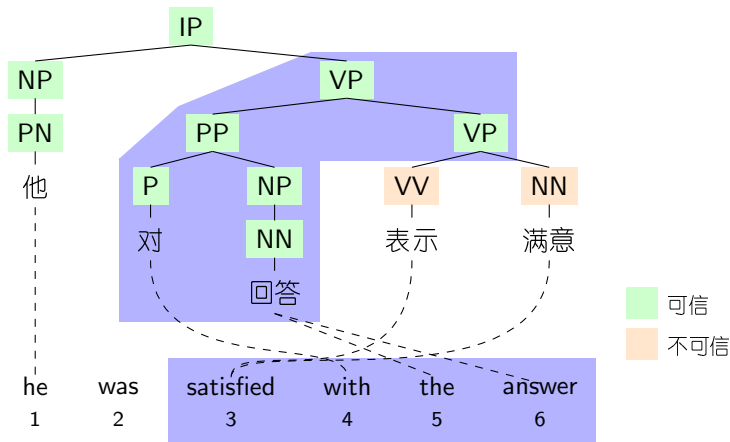
- 可信节点本质上定义了规则的边界，规则需要满足
 - 左部树片段的根节点是可信节点
 - 左部树片段的叶子节点是终结符或者可信节点



规则抽取

- 可信节点本质上定义了规则的边界，规则需要满足
 - 左部树片的根节点是可信节点
 - 左部树片的叶子节点是终结符或者可信节点

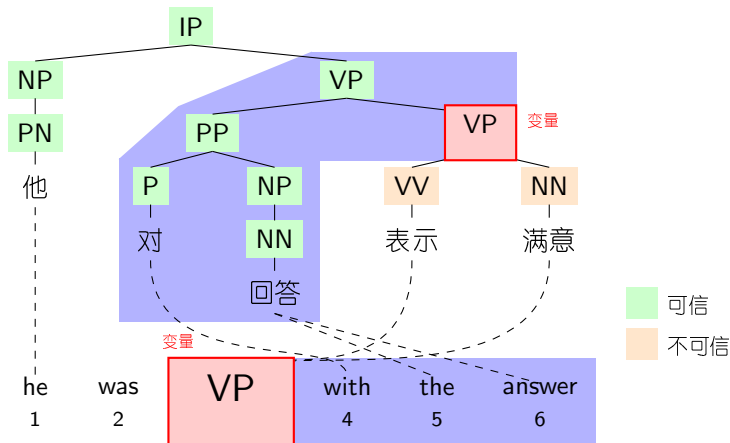
例如：VP(PP(P(对) NP(NN(回答))) VP₁) → VP₁ with the answer



规则抽取

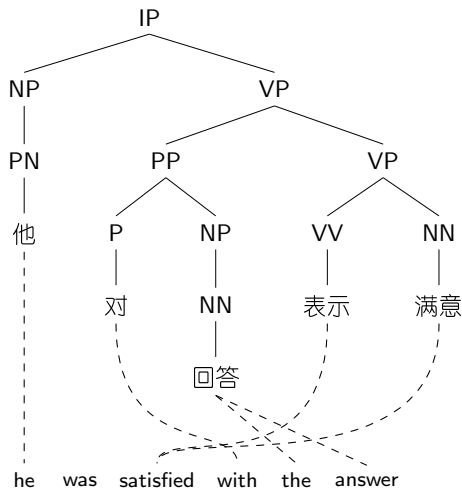
- 可信节点本质上定义了规则的边界，规则需要满足
 - 左部树片的根节点是可信节点
 - 左部树片的叶子节点是终结符或者可信节点

例如：VP(PP(P(对) NP(NN(回答))) VP₁) → VP₁ with the answer



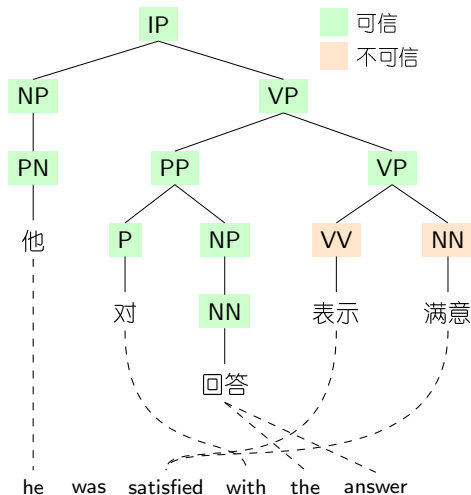
规则抽取 - 树的切割

- 所有可信节点构成了句法树的边缘集合(frontier set), 进而得到树的一种切割以及切割得到的树片段(或规则)



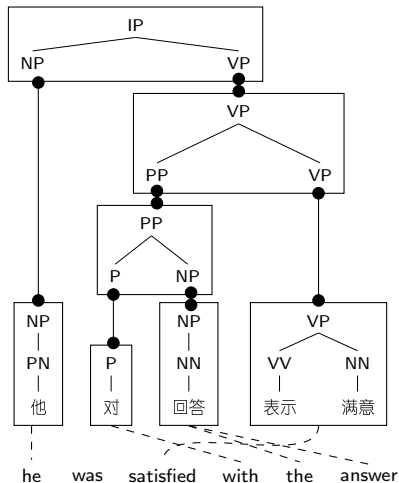
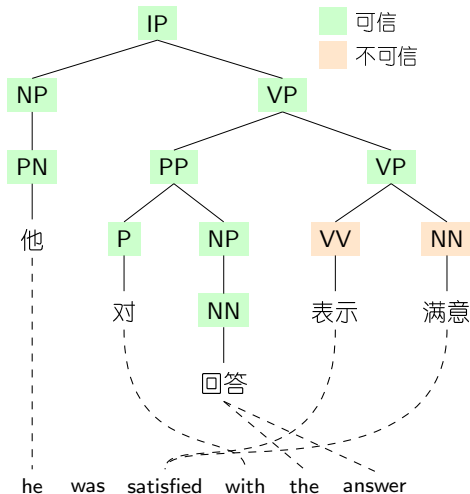
规则抽取 - 树的切割

- 所有可信节点构成了句法树的边缘集合(frontier set), 进而得到树的一种切割以及切割得到的树片段(或规则)



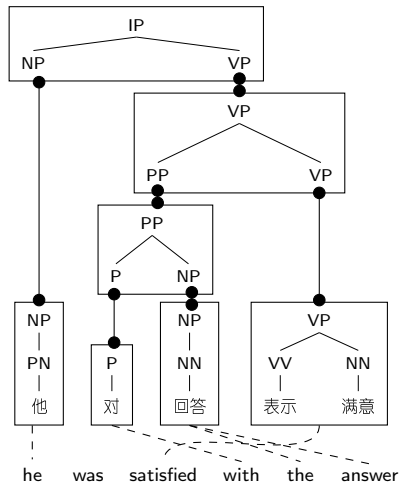
规则抽取 - 树的切割

- 所有可信节点构成了句法树的边缘集合(frontier set), 进而得到树的一种切割以及切割得到的树片段(或规则)



规则抽取 - 最小规则

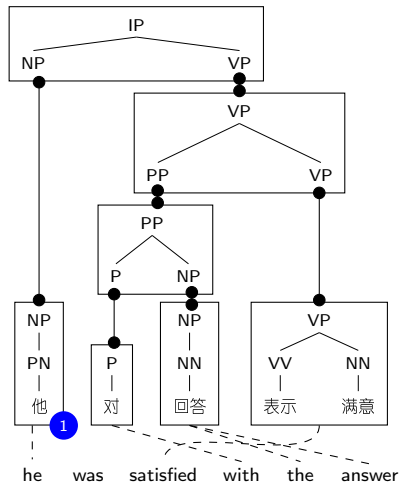
- 边缘集合所对应的树切割，产生了若干树片段，这些树片段内部包含其它的切割点，每个树片段都对应了一条最小翻译规则(即无法继续分解的规则)。见如下示例



规则抽取 - 最小规则

- 边缘集合所对应的树切割，产生了若干树片段，这些树片段内部包含其它的切割点，每个树片段都对应了一条最小翻译规则(即无法继续分解的规则)。见如下示例

r_1 NP(PN(他)) \rightarrow he

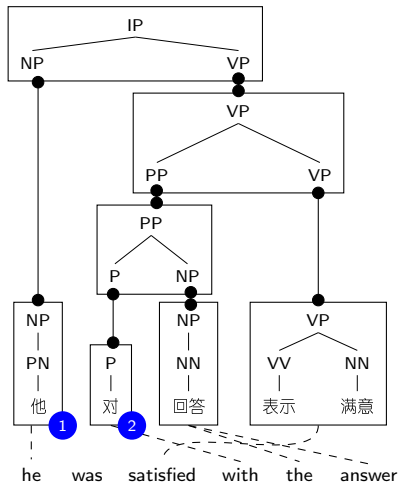


规则抽取 - 最小规则

- 边缘集合所对应的树切割，产生了若干树片段，这些树片段内部包含其它的切割点，每个树片段都对应了一条最小翻译规则(即无法继续分解的规则)。见如下示例

r_1 NP(PN(他)) \rightarrow he

r_2 P(对) \rightarrow with



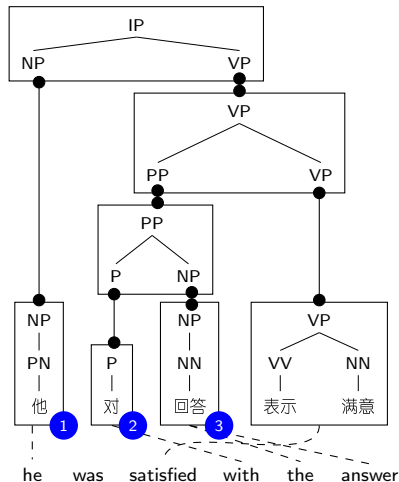
规则抽取 - 最小规则

- 边缘集合所对应的树切割，产生了若干树片段，这些树片段内部包含其它的切割点，每个树片段都对应了一条最小翻译规则(即无法继续分解的规则)。见如下示例

r_1 NP(PN(他)) \rightarrow he

r_2 P(对) \rightarrow with

r_3 NP(NN(回答)) \rightarrow the answer



规则抽取 - 最小规则

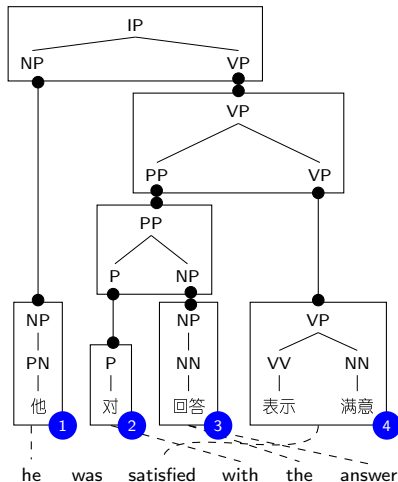
- 边缘集合所对应的树切割，产生了若干树片段，这些树片段内部包含其它的切割点，每个树片段都对应了一条最小翻译规则(即无法继续分解的规则)。见如下示例

r_1 NP(PN(他)) \rightarrow he

r_2 P(对) \rightarrow with

r_3 NP(NN(回答)) \rightarrow the answer

r_4 VP(VV(表示) NN(满意)) \rightarrow
satisfied



规则抽取 - 最小规则

- 边缘集合所对应的树切割，产生了若干树片段，这些树片段内部包含其它的切割点，每个树片段都对应了一条最小翻译规则(即无法继续分解的规则)。见如下示例

r_1 NP(PN(他)) \rightarrow he

r_2 P(对) \rightarrow with

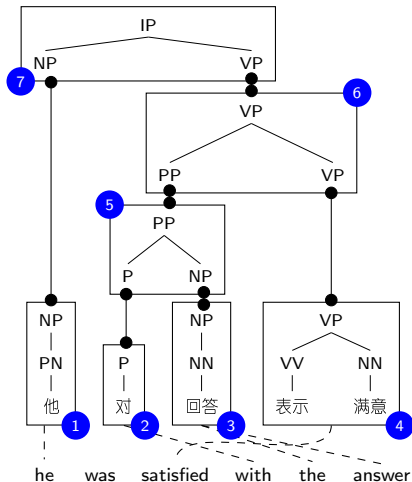
r_3 NP(NN(回答)) \rightarrow the answer

r_4 VP(VV(表示) NN(满意)) \rightarrow
satisfied

r_5 PP(P₁ NP₂) \rightarrow
P₁ NP₂

r_6 VP(PP₁ VP₂) \rightarrow
VP₂ PP₁

r_7 IP(NP₁ VP₂) \rightarrow
NP₁ VP₂



一元规则

- 上述例子中，词性生成单词的过程对应一种一元规则，即树的孩子节点只有一个，例如

$$\text{NP}(\text{PN}(\text{他})) \rightarrow \text{he}$$

显然，NP只有一个孩子PN，PN只有一个孩子he。实际上，这条规则对应两条规则的组合

$$\text{PN}(\text{他}) \rightarrow \text{he}$$

$$\text{NP}(\text{PN}_1) \rightarrow \text{PN}_1$$

这里，把两条一元规则合并成“NP(PN(他)) → he”主要是出于系统实现的考虑，因为允许任意词性到句法标记的一元规则会大大增加翻译假设数量，但是本质上这些一元规则并没有带来太多新的信息

这也是为什么上图中PN虽然是可信节点，但是没有作为树切割的边界

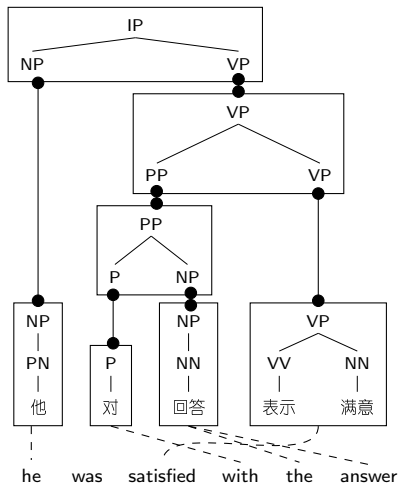
更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

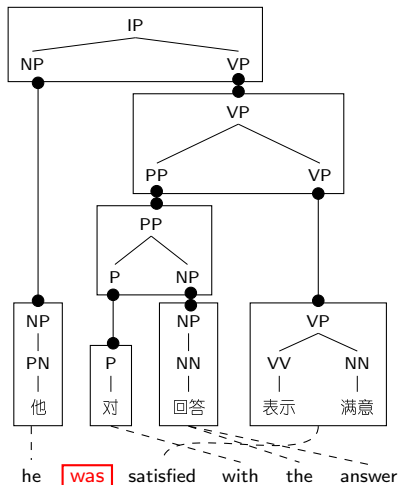
r_1 NP(PN(他)) \rightarrow he
 r_4 VP(VV(表示) NN(满意)) \rightarrow satisfied
 r_6 VP(PP₁ VP₂) \rightarrow VP₂ PP₁
 r_7 IP(NP₁ VP₂) \rightarrow NP₁ VP₂



更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

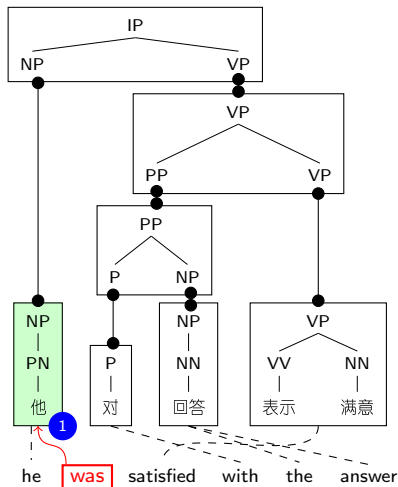
r_1 NP(PN(他)) \rightarrow he
 r_4 VP(VV(表示) NN(满意)) \rightarrow
satisfied
 r_6 VP(PP₁ VP₂) \rightarrow VP₂ PP₁
 r_7 IP(NP₁ VP₂) \rightarrow NP₁ VP₂



更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

r_1 NP(PN(他)) \rightarrow he
 r_4 VP(VV(表示) NN(满意)) \rightarrow satisfied
 r_6 VP(PP₁ VP₂) \rightarrow VP₂ PP₁
 r_7 IP(NP₁ VP₂) \rightarrow NP₁ VP₂
 r_8 NP(PN(他)) \rightarrow he was



更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

r_1 NP(PN(他)) \rightarrow he

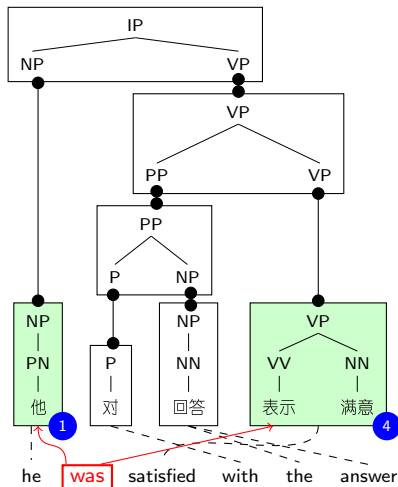
r_4 VP(VV(表示) NN(满意)) \rightarrow
satisfied

r_6 VP(PP₁ VP₂) \rightarrow VP₂ PP₁

r_7 IP(NP₁ VP₂) \rightarrow NP₁ VP₂

r_8 NP(PN(他)) \rightarrow he **was**

r_9 VP(VV(表示) NN(满意)) \rightarrow
was satisfied



更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

r_1 NP(PN(他)) \rightarrow he

r_4 VP(VV(表示) NN(满意)) \rightarrow
satisfied

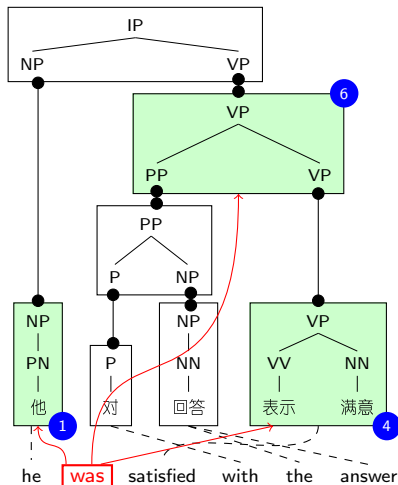
r_6 VP(PP₁ VP₂) \rightarrow VP₂ PP₁

r_7 IP(NP₁ VP₂) \rightarrow NP₁ VP₂

r_8 NP(PN(他)) \rightarrow he **was**

r_9 VP(VV(表示) NN(满意)) \rightarrow
was satisfied

r_{10} VP(PP₁ VP₂) \rightarrow
was VP₂ PP₁



更多的规则 - 处理空对齐

- 句法翻译系统成功的前提是规则可以覆盖尽可能多的语言现象。为了得到覆盖度更高的规则集，需要处理空对齐的情况 - 把空对齐单词附着在所有可能的规则上

r_1 NP(PN(他)) \rightarrow he

r_4 VP(VV(表示) NN(满意)) \rightarrow
satisfied

r_6 VP(PP₁ VP₂) \rightarrow VP₂ PP₁

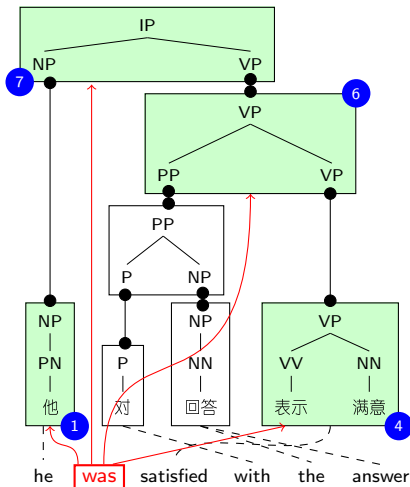
r_7 IP(NP₁ VP₂) \rightarrow NP₁ VP₂

r_8 NP(PN(他)) \rightarrow he **was**

r_9 VP(VV(表示) NN(满意)) \rightarrow
was satisfied

r_{10} VP(PP₁ VP₂) \rightarrow
was VP₂ PP₁

r_{11} IP(NP₁ VP₂) \rightarrow
NP₁ **was** VP₂



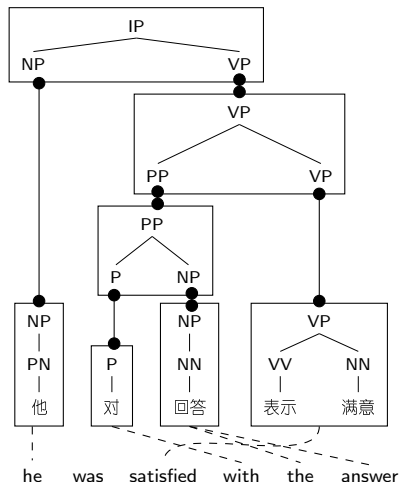
更多的规则 - 组合规则

- 最小规则之间可以进行组合，得到更大粒度的组合规则，可以使用更多上下文信息，并进行更复杂的调序。
比如，三条最小规则组合成一条composed-3规则

更多的规则 - 组合规则

- 最小规则之间可以进行组合，得到更大粒度的组合规则，可以使用更多上下文信息，并进行更复杂的调序。
比如，三条最小规则组合成一条composed-3规则

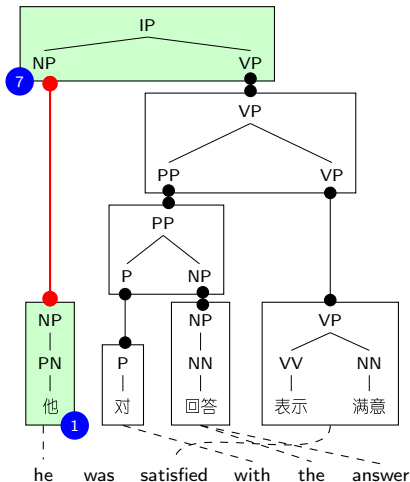
r_1 $NP(PN(\text{他})) \rightarrow \text{he}$
 r_5 $PP(P_1 NP_2) \rightarrow P_1 NP_2$
 r_6 $VP(PP_1 VP_2) \rightarrow VP_2 PP_1$
 r_7 $IP(NP_1 VP_2) \rightarrow NP_1 VP_2$



更多的规则 - 组合规则

- 最小规则之间可以进行组合，得到更大粒度的组合规则，可以使用更多上下文信息，并进行更复杂的调序。
比如，三条最小规则组合成一条composed-3规则

r_1 $NP(PN(\text{他})) \rightarrow \text{he}$
 r_5 $PP(P_1 NP_2) \rightarrow P_1 NP_2$
 r_6 $VP(PP_1 VP_2) \rightarrow VP_2 PP_1$
 r_7 $IP(NP_1 VP_2) \rightarrow NP_1 VP_2$
 $r_{1,7}$ $IP(NP(PN(\text{他})) VP_1) \rightarrow$
 $\text{he } VP_1$



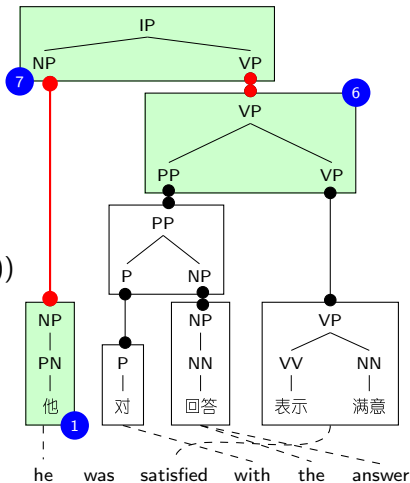
更多的规则 - 组合规则

- 最小规则之间可以进行组合，得到更大粒度的组合规则，可以使用更多上下文信息，并进行更复杂的调序。比如，三条最小规则组合成一条composed-3规则

$$r_1 \quad \text{NP}(\text{PN}(\text{他})) \rightarrow \text{he}$$
$$r_5 \quad \text{PP}(P_1 \text{ NP}_2) \rightarrow P_1 \text{ NP}_2$$
$$r_6 \quad \text{VP}(\text{PP}_1 \text{ VP}_2) \rightarrow \text{VP}_2 \text{ PP}_1$$
$$r_7 \quad \text{IP}(\text{NP}_1 \text{ VP}_2) \rightarrow \text{NP}_1 \text{ VP}_2$$

$r_{1,7}$ IP(NP(PN(他)) VP₁) →
he VP₁

$r_{1,6}$ IP(NP(PN(他)) VP(PP_1 VP_2))
 $,7$ \rightarrow he VP_2 PP_1



更多的规则 - 组合规则

- 最小规则之间可以进行组合，得到更大粒度的组合规则，可以使用更多上下文信息，并进行更复杂的调序。
比如，三条最小规则组合成一条composed-3规则

r_1 $\text{NP}(\text{PN}(\text{他})) \rightarrow \text{he}$

r_5 $\text{PP}(\text{P}_1 \text{ NP}_2) \rightarrow \text{P}_1 \text{ NP}_2$

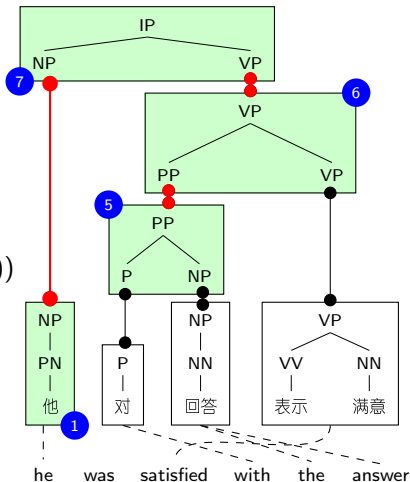
r_6 $\text{VP}(\text{PP}_1 \text{ VP}_2) \rightarrow \text{VP}_2 \text{ PP}_1$

r_7 $\text{IP}(\text{NP}_1 \text{ VP}_2) \rightarrow \text{NP}_1 \text{ VP}_2$

$r_{1,7}$ $\text{IP}(\text{NP}(\text{PN}(\text{他})) \text{ VP}_1) \rightarrow$
 he VP_1

$r_{1,6}$ $\text{IP}(\text{NP}(\text{PN}(\text{他})) \text{ VP}(\text{PP}_1 \text{ VP}_2))$
 $,7$ $\rightarrow \text{he VP}_2 \text{ PP}_1$

$r_{1,5}$ $\text{IP}(\text{NP}(\text{PN}(\text{他}))$
 $,6,7$ $\text{VP}(\text{P}_1 \text{ NP}_2 \text{ VP}_3))$
 $\rightarrow \text{he VP}_3 \text{ P}_1 \text{ NP}_2$

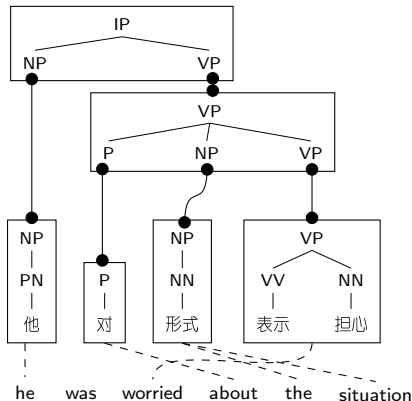


更多的规则 - SPMT规则

- 任意大小的规则都可以通过组合的方式获取，但是组合过多数量的规则会大大增加规则集大小。SPMT一种高效地获得大颗粒度规则的方法 (Marcu et al., 2006)
 - ▶ 先抽取短语，之后找到覆盖这个短语的可信节点
 - ▶ 以这个可信节点做根，生成包含该短语的规则

更多的规则 - SPMT规则

- 任意大小的规则都可以通过组合的方式获取，但是组合过多数量的规则会大大增加规则集大小。SPMT一种高效地获得大颗粒度规则的方法 (Marcu et al., 2006)
 - 先抽取短语，之后找到覆盖这个短语的可信节点
 - 以这个可信节点做根，生成包含该短语的规则



更多的规则 - SPMT规则

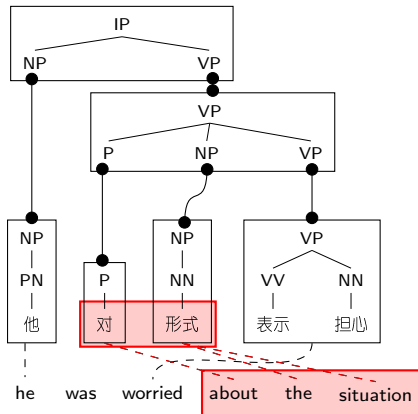
- 任意大小的规则都可以通过组合的方式获取，但是组合过多数量的规则会大大增加规则集大小。SPMT一种高效地获得大颗粒度规则的方法 (Marcu et al., 2006)
 - 先抽取短语，之后找到覆盖这个短语的可信节点
 - 以这个可信节点做根，生成包含该短语的规则

对于任意一个与词对齐兼容的短语，可以找到包含它的“最小”翻译规则，即SPMT规则，比如

对 形式 \rightarrow about the situation

可以很容易得到它的SPMT规则

$VP(P(\text{对}) NP(NN(\text{局势})) VP_1) \rightarrow$
 VP_1 about the situation



更多的规则 - SPMT规则

- 任意大小的规则都可以通过组合的方式获取，但是组合过多数量的规则会大大增加规则集大小。SPMT一种高效地获得大颗粒度规则的方法 (Marcu et al., 2006)
 - 先抽取短语，之后找到覆盖这个短语的可信节点
 - 以这个可信节点做根，生成包含该短语的规则

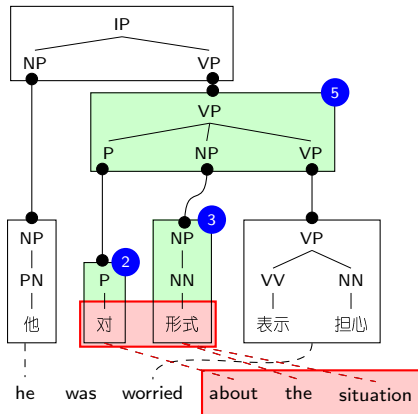
对于任意一个与词对齐兼容的短语，可以找到包含它的“最小”翻译规则，即SPMT规则，比如

对 形式 \rightarrow about the situation

可以很容易得到它的SPMT规则

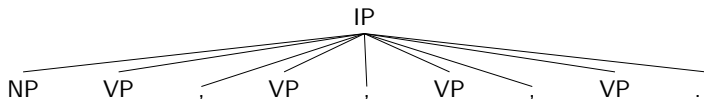
$VP(P(\text{对}) NP(NN(\text{局势})) VP_1) \rightarrow$
 VP_1 about the situation

但是，如果用组合的方式，需要三条最小规则才能得到这条规则



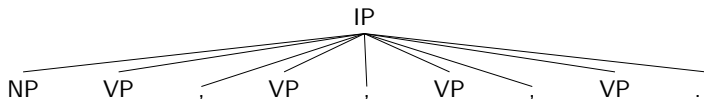
更多的规则 - 句法树二叉化

- 句法分析器生成的句法树可能会非常平坦，这会导致抽取的规则很“大”而且规则无法继续被分解
 - 比如，在CTB中经常会看到很宽的子树结构

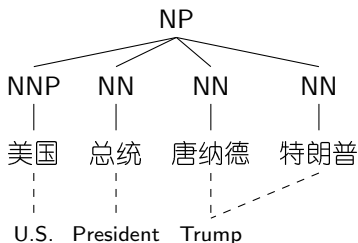


更多的规则 - 句法树二叉化

- 句法分析器生成的句法树可能会非常平坦，这会导致抽取的规则很“大”而且规则无法继续被分解
 - 比如，在CTB中经常会看到很宽的子树结构



- 一个例子



抽取到的规则：

$NP(NNP_1\ NN_2\ NN(\text{唐纳德})\ NN(\text{特朗普}))$

$\rightarrow NNP_1\ NN_2\ Trump$

$NP(NNP_1\ NN(\text{总统})\ NN(\text{唐纳德})\ NN(\text{特朗普}))$

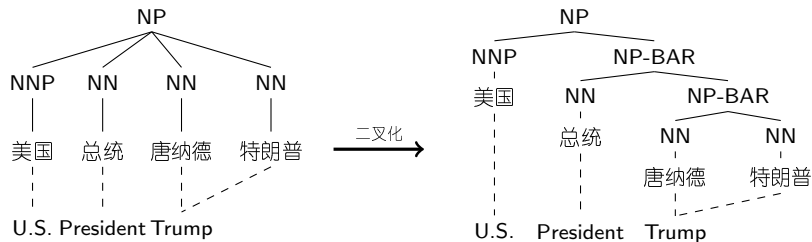
$\rightarrow NNP_1\ President\ Trump$

不能抽取到的规则：

$NP(NN(\text{唐纳德})\ NN(\text{特朗普})) \rightarrow Trump$

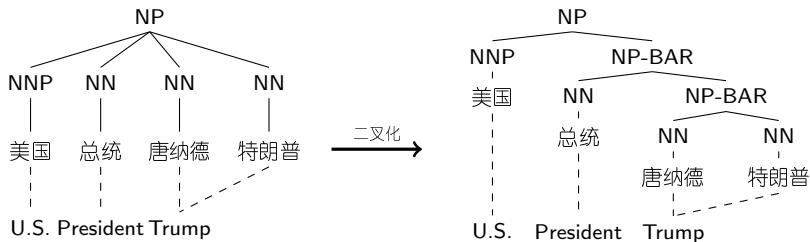
更多的规则 - 句法树二叉化(续)

- 一种解决问题的思路是用二叉化方法把树结构变得更深



更多的规则 - 句法树二叉化(续)

- 一种解决问题的思路是用二叉化方法把树结构变得更深



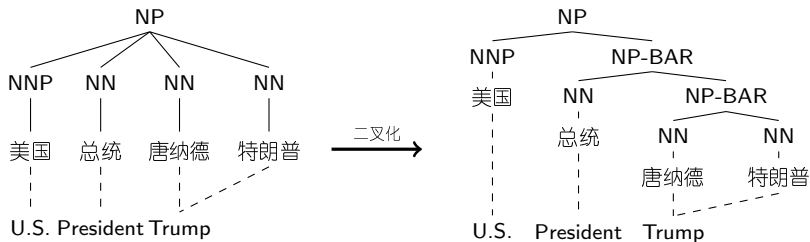
二叉化增加了更多的可信节点，这也带来了新的规则

$\text{NP-BAR}(\text{NN}(\text{唐纳德}) \text{NN}(\text{特朗普})) \rightarrow \text{Trump}$

$\text{NP-BAR}(\text{NN}_1 \text{NP-BAR}_2) \rightarrow \text{NN}_1 \text{NP-BAR}_2$

更多的规则 - 句法树二叉化(续)

- 一种解决问题的思路是用二叉化方法把树结构变得更深



二叉化增加了更多的可信节点，这也带来了新的规则

$\text{NP-BAR}(\text{NN}(\text{唐纳德}) \text{NN}(\text{特朗普})) \rightarrow \text{Trump}$

$\text{NP-BAR}(\text{NN}_1 \text{NP-BAR}_2) \rightarrow \text{NN}_1 \text{NP-BAR}_2$

- 树二叉化已经成为基于句法机器翻译模型的常用方法
 - 有很多策略：左优先、右优先、head优先等等
 - 二叉化可以得到更多(细粒度)规则，保证规则的覆盖度

引入双语句法信息

- 对于树到树模型，源语和目标语端都有句法树，需要使用树片段到树片段的映射来描述翻译过程，这种映射关系被描述为树到树翻译规则。这里，把

$$\langle \text{VP}, \text{VP} \rangle \rightarrow \langle \text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{ NN}_2)), \\ \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}_2 \text{ PP}_1)) \rangle$$

表示为树片段到树片段的映射形式

$$\text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{ NN}_2)) \\ \rightarrow \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}_2 \text{ PP}_1))$$

引入双语句法信息

- 对于树到树模型，源语和目标语端都有句法树，需要使用树片段到树片段的映射来描述翻译过程，这种映射关系被描述为树到树翻译规则。这里，把

$$\langle \text{VP}, \text{VP} \rangle \rightarrow \langle \text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{ NN}_2)), \\ \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}_2 \text{ PP}_1)) \rangle$$

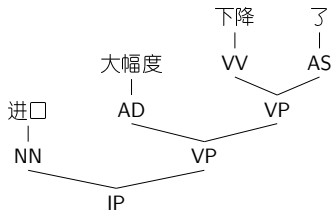
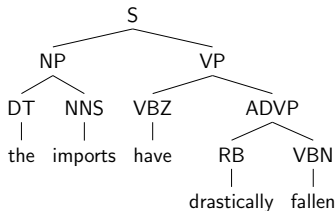
表示为树片段到树片段的映射形式

$$\text{VP}(\text{PP}_1 \text{ VP}(\text{VV}(\text{表示}) \text{ NN}_2)) \\ \rightarrow \text{VP}(\text{VBZ}(\text{was}) \text{ VP}(\text{VBN}_2 \text{ PP}_1))$$

- 可以通过扩展GHKM方法进行树到树规则抽取
 - ▶ 双语端进行可信节点的识别，之后找到节点之间的对应
 - ▶ 基于对应的节点获得树片段的对应，即抽取树到树规则
 - ▶ 规则组合、SPMT等方法同样适用

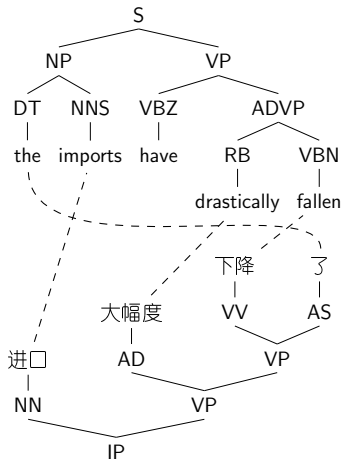
方法1：利用词对齐归纳树到树规则

- 简单直接的方法是把GHKM方法扩展到双语的情况，利用词对齐归纳树到树映射



方法1：利用词对齐归纳树到树规则

- 简单直接的方法是把GHKM方法扩展到双语的情况，利用词对齐归纳树到树映射

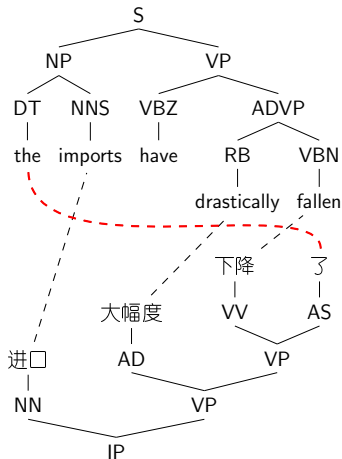


抽取得到的规则

-
- r_1 AS(了) \rightarrow DT(the)
- r_2 NN(进□) \rightarrow NNS(imports)
- r_3 AD(大幅度) \rightarrow RB(drastically)
- r_4 VV(下降) \rightarrow VBN(fallen)
- r_5 IP(NN₁ VP(AD₂ VP(VV₃ AS₄))) \rightarrow
S(NP(DT₄ NNS₁) VP(VBZ(have) ADVP(RB₂ VBN₃)))

方法1：利用词对齐归纳树到树规则

- 简单直接的方法是把GHKM方法扩展到双语的情况，利用词对齐归纳树到树映射
- 但是词对齐的错误往往会导致很多规则无法抽取



抽取得到的规则

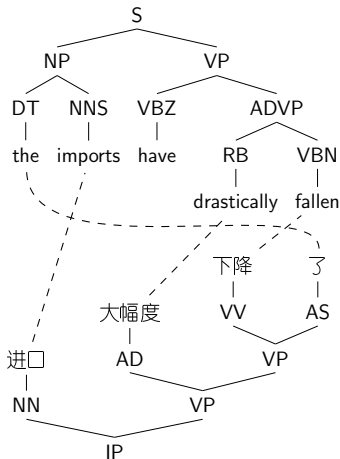
- r_1 AS(了) \rightarrow DT(the)
- r_2 NN(进□) \rightarrow NNS(imports)
- r_3 AD(大幅度) \rightarrow RB(drastically)
- r_4 VV(下降) \rightarrow VBN(fallen)
- r_5 IP(NN₁ VP(AD₂ VP(VV₃ AS₄))) \rightarrow
S(NP(DT₄ NNS₁) VP(VBZ(have) ADVP(RB₂ VBN₃)))

无法得到的规则

- $r_?$ AS(了) \rightarrow VBZ(have)
- $r_?$ NN(进□) \rightarrow
NP(DT(the) NNS(imports))
- $r_?$ IP(NN₁ VP₂) \rightarrow S(NP₁ VP₂)

方法2：利用节点对齐抽取树到树规则

- 另一种思路是直接获取源语言树节点到目标语树节点的对应关系，然后直接抽取规则，这样可避免词对齐错误
 - 节点对其可以更准确的捕捉双语结构的对应

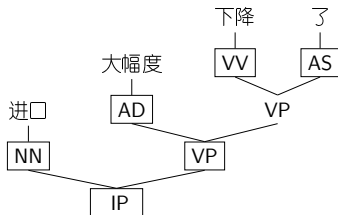
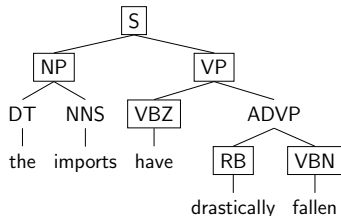


抽取得到的规则(词对齐)

-
- r_1 AS(了) \rightarrow DT(the)
- r_2 NN(进口) \rightarrow NNS(imports)
- r_3 AD(大幅度) \rightarrow RB(drastically)
- r_4 VV(下降) \rightarrow VBN(fallen)
- r_5 IP(NN₁ VP(AD₂ VP(VV₃ AS₄))) \rightarrow
S(NP(DT₄ NNS₁) VP(VBZ(have) ADVP(RB₂ VBN₃)))

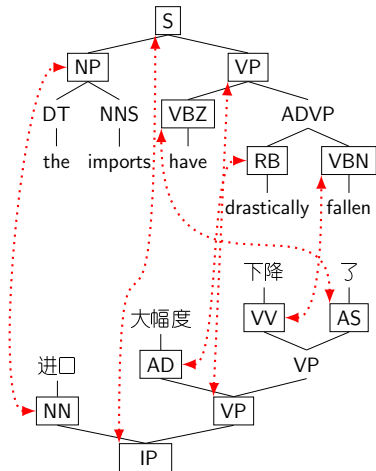
方法2：利用节点对齐抽取树到树规则

- 另一种思路是直接获取源语言树节点到目标语树节点的对应关系，然后直接抽取规则，这样可避免词对齐错误
 - 节点对其可以更准确的捕捉双语结构的对应



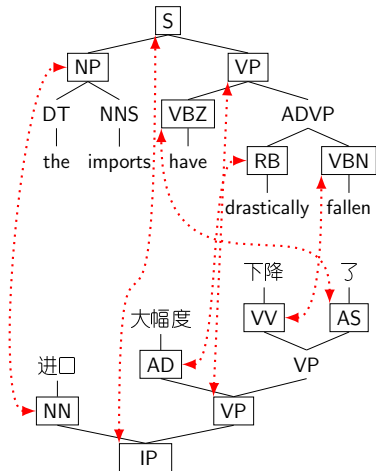
方法2：利用节点对齐抽取树到树规则

- 另一种思路是直接获取源语言树节点到目标语树节点的对应关系，然后直接抽取规则，这样可避免词对齐错误
 - 节点对其可以更准确的捕捉双语结构的对应



方法2：利用节点对齐抽取树到树规则

- 另一种思路是直接获取源语言树节点到目标语树节点的对应关系，然后直接抽取规则，这样可避免词对齐错误
 - 节点对其可以更准确的捕捉双语结构的对应

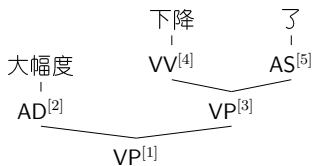
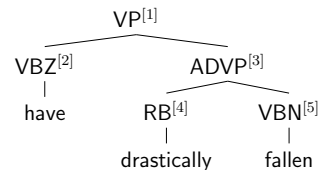


抽取得到的规则(子树对齐)

- r_1 AS(了) \rightarrow DT(the)
- r_2 NN(进□) \rightarrow NNS(imports)
- r_3 AD(大幅度) \rightarrow RB(drastically)
- r_4 VV(下降) \rightarrow VBN(fallen)
- r_5 IP(NN₁ VP(AD₂ VP(VV₃ AS₄))) \rightarrow
S(NP(DT₄ NNS₁) VP(VBZ(have) ADVP(RB₂ VBN₃)))
- r_6 AS(了) \rightarrow VBZ(have)
- r_7 NN(进□) \rightarrow
NP(DT(the) NNS(imports))
- r_8 VP(AD₁ VP(VV₂ AS₃)) \rightarrow
VP(VBZ₃ ADVP(RB₁ VBN₂))
- r_9 IP(NN₁ VP₂) \rightarrow S(NP₁ VP₂)

节点对齐矩阵

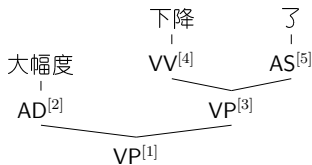
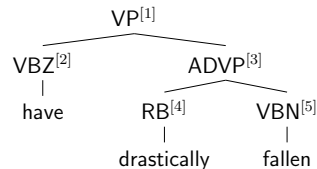
- 节点对齐的自动获取：1) 基于分类模型的方法；2) 无指导节点对齐的方法
- 使用节点对齐的另一个好处是，我们可以直接用节点对齐矩阵进行规则抽取，而不是用单一的对齐结果
 - 对齐矩阵可以帮助抽取更多样的规则



<i>VP^[1]</i>	<i>VBZ^[2]</i>	<i>ADVP^[3]</i>	<i>RB^[4]</i>	<i>VBN^[5]</i>	
•	•	•	•	•	VP ^[1]
•	•	•	•	•	AD ^[2]
•	•	•	•	•	VP ^[3]
•	•	•	•	•	VV ^[4]
•	•	•	•	•	AS ^[5]

节点对齐矩阵

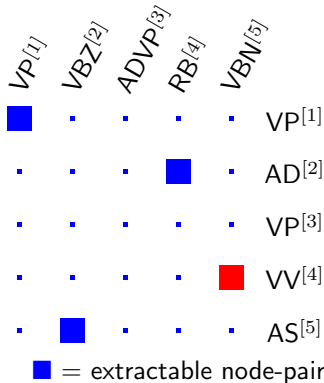
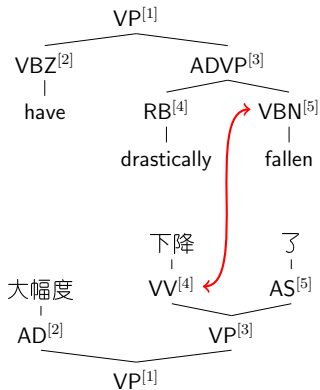
- 节点对齐的自动获取：1) 基于分类模型的方法；2) 无指导节点对齐的方法
- 使用节点对齐的另一个好处是，我们可以直接用节点对齐矩阵进行规则抽取，而不是用单一的对齐结果
 - 对齐矩阵可以帮助抽取更多样的规则



	VP ^[1]	VBZ ^[2]	ADVP ^[3]	RB ^[4]	VBN ^[5]	
VP ^[1]	■	•	•	•	•	VP ^[1]
AD ^[2]	•	•	•	■	•	AD ^[2]
VP ^[3]	•	•	•	•	•	VP ^[3]
VV ^[4]	•	•	•	•	■	VV ^[4]
AS ^[5]	•	■	•	•	•	AS ^[5]
■ = extractable node-pair						

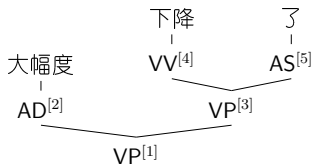
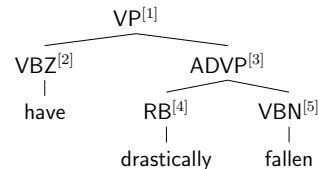
节点对齐矩阵

- 节点对齐的自动获取：1) 基于分类模型的方法；2) 无指导节点对齐的方法
- 使用节点对齐的另一个好处是，我们可以直接用节点对齐矩阵进行规则抽取，而不是用单一的对齐结果
 - 对齐矩阵可以帮助抽取更多样的规则



节点对齐矩阵

- 节点对齐的自动获取：1) 基于分类模型的方法；2) 无指导节点对齐的方法
- 使用节点对齐的另一个好处是，我们可以直接用节点对齐矩阵进行规则抽取，而不是用单一的对齐结果
 - 对齐矩阵可以帮助抽取更多样的规则

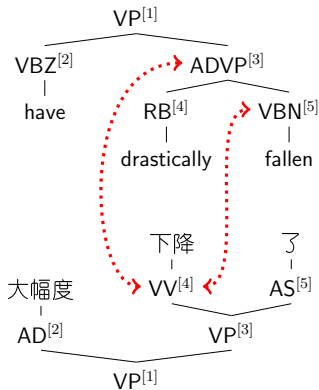


	VP ^[1]	VBZ ^[2]	ADVP ^[3]	RB ^[4]	VBN ^[5]	
VP ^[1]	■	.	■	.	.	VP ^[1]
AD ^[2]	.	■	■	■	.	AD ^[2]
VP ^[3]	■	■	■	.	■	VP ^[3]
VV ^[4]	.	.	■	.	■	VV ^[4]
AS ^[5]	.	■	.	.	■	AS ^[5]

■ = possible alignment

节点对齐矩阵

- 节点对齐的自动获取：1) 基于分类模型的方法；2) 无指导节点对齐的方法
- 使用节点对齐的另一个好处是，我们可以直接用节点对齐矩阵进行规则抽取，而不是用单一的对齐结果
 - 对齐矩阵可以帮助抽取更多样的规则



VP ^[1]	VBZ ^[2]	ADVP ^[3]	RB ^[4]	VBN ^[5]	
■	.	■	.	.	VP ^[1]
.	■	■	■	.	AD ^[2]
■	■	■	.	■	VP ^[3]
.	.	■	.	■	VV ^[4]
.	■	.	.	■	AS ^[5]

■ = possible alignment

特征

- 与短语和层次短语模型一样，句法模型也使用判别式模型进行建模 - $P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$ 。其中特征权重 $\{\lambda_i\}$ 可以使用最小错误率训练进行调优，特征函数 $\{h_i\}$ 需要用户定义。

特征

- 与短语和层次短语模型一样，句法模型也使用判别式模型进行建模 - $P(d, \mathbf{t}|\mathbf{s}) = \frac{\exp(\sum_{i=1}^M \lambda_i \cdot h_i(d, \mathbf{s}, \mathbf{t}))}{\sum_{d', \mathbf{t}'} \exp(\sum_{i=1}^M \lambda_i \cdot h_i(d', \mathbf{s}, \mathbf{t}'))}$ 。其中特征权重 $\{\lambda_i\}$ 可以使用最小错误率训练进行调优，特征函数 $\{h_i\}$ 需要用户定义。
- 这里，所有规则满足 $\langle \alpha_h, \beta_h \rangle \rightarrow \langle \alpha_r, \beta_r, \sim \rangle$ 的形式
 - ▶ α_h 和 β_h 是规则左部的源语和目标语部分，对应树结构的根节点
 - ▶ α_r 和 β_r 是规则右部的源语和目标语部分，对应树结构
 - ▶ \sim 表示 α_r 和 β_r 中叶子非终结符的对应
 - ▶ 此外，定义 $\tau(\alpha_r)$ 和 $\tau(\beta_r)$ 为源语和目标语树结构的叶子节点序列。例如，对于规则 $\langle \text{VP}, \text{VP} \rangle \rightarrow \langle \text{VP}(\text{PP}_1 \text{VP}(\text{VV}(\text{表示}) \text{NN}_2)), \text{VP}(\text{VBZ}(\text{was}) \text{VP}(\text{VBZ}_2 \text{PP}_1)) \rangle$ ，有

$$\tau(\alpha_r) = \text{PP}_1 \text{ 表示 } \text{NN}_2$$

$$\tau(\beta_r) = \text{was VBZ}_2 \text{ PP}_1$$

特征(续)

- **特征1-2： 短语翻译概率**，即正向翻译概率 $\log(P(\tau(\beta_r)|\tau(\alpha_r)))$ 和反向翻译概率 $\log(P(\tau(\alpha_r)|\tau(\beta_r)))$ 。这里， $\tau(\alpha_r)$ 和 $\tau(\beta_r)$ 都被看做短语，因此可以直接复用短语系统的方法进行计算。
- **特征3-4： 词汇翻译概率**，即 $\log(P_{\text{lex}}(\tau(\beta_r)|\tau(\alpha_r)))$ 和 $\log(P_{\text{lex}}(\tau(\alpha_r)|\tau(\beta_r)))$ 。可以用短语系统中的词汇翻译概率描述源语和目标语单词对应的情况。

特征(续)

- **特征1-2：短语翻译概率**，即正向翻译概率 $\log(P(\tau(\beta_r)|\tau(\alpha_r)))$ 和反向翻译概率 $\log(P(\tau(\alpha_r)|\tau(\beta_r)))$ 。这里， $\tau(\alpha_r)$ 和 $\tau(\beta_r)$ 都被看做短语，因此可以直接复用短语系统的方法进行计算。
- **特征3-4：词汇翻译概率**，即 $\log(P_{\text{lex}}(\tau(\beta_r)|\tau(\alpha_r)))$ 和 $\log(P_{\text{lex}}(\tau(\alpha_r)|\tau(\beta_r)))$ 。可以用短语系统中的词汇翻译概率描述源语和目标语单词对应的情况。
- **特征5：n-gram语言模型**，即 $\log(P_{\text{lm}}(\mathbf{t}))$ 。度量译文的流畅度，可以使用大规模目标语单语数据得到。
- **特征6：译文长度**，即 $|\mathbf{t}|$ 。避免模型倾向于短译文，同时让系统自动学习对译文长度的偏好。
- **特征7：翻译规则数量**。这个特征是为了避免模型仅仅使用少量特征构成翻译推导(因为翻译概率相乘，因子少结果一般会大一些)，同时让系统自动学习对使用规则数量的偏好。

特征(续2)

- **特征8**：源语言被翻译为空的单词数量。注意，空翻译规则(或特征)有时也被称作evil feature，这类特征在一些数据集上对BLEU有很好的提升作用，但是会造成人工评价的下降，因此需要谨慎使用。

特征(续2)

- **特征8：源语言被翻译为空的单词数量。**注意，空翻译规则(或特征)有时也被称作evil feature，这类特征在一些数据集上对BLEU有很好的提升作用，但是会造成人工评价的下降，因此需要谨慎使用。
- **特征9：翻译规则生成概率，**
即 $\log(P_{\text{rule}}(\alpha_r, \beta_r, \sim | \alpha_h, \beta_h))$ 。这个特征可以被看做是生成翻译推导的概率。
- **特征10：组合规则的数量。**学习使用组合规则(或最小规则)的偏好。
- **特征11：词汇化规则的数量。**学习使用含有终结符规则的偏好。
- **特征12：低频规则的数量。**学习使用训练数据中出现频次低于3的规则偏好。低频规则大多并不可靠，这个特征本质上也是为了区分不同质量规则。

特征(续2)

- **特征8：源语言被翻译为空的单词数量。**注意，空翻译规则(或特征)有时也被称作evil feature，这类特征在一些数据集上对BLEU有很好的提升作用，但是会造成人工评价的下降，因此需要谨慎使用。
- **特征9：翻译规则生成概率，**
即 $\log(P_{\text{rule}}(\alpha_r, \beta_r, \sim | \alpha_h, \beta_h))$ 。这个特征可以被看做是生成翻译推导的概率。
- **特征10：组合规则的数量。**学习使用组合规则(或最小规则)的偏好。
- **特征11：词汇化规则的数量。**学习使用含有终结符规则的偏好。
- **特征12：低频规则的数量。**学习使用训练数据中出现频次低于3的规则偏好。低频规则大多并不可靠，这个特征本质上也是为了区分不同质量规则。
- **注意！**特征9-12也被看做是句法特征。当然，还有很多很多特征，感兴趣可以自己设计或查阅相关论文。

解码

- 翻译时可以用如下式子计算每个推导的模型得分

$$\text{score}(d, \mathbf{t}, \mathbf{s}) = \sum_{r \in d} \log(\text{score}(r)) + \lambda_{\text{lm}} \log(P_{\text{lm}}(\mathbf{t})) + \lambda_l |\mathbf{t}|$$

其中， $\text{score}(r)$ 表示每条规则的得分，由特征1-4和特征7-12共同计算得到，因此也可以把特征1-4和特征7-12看做是规则特征

解码

- 翻译时可以用如下式子计算每个推导的模型得分

$$\text{score}(d, \mathbf{t}, \mathbf{s}) = \sum_{r \in d} \log(\text{score}(r)) + \lambda_{\text{lm}} \log(P_{\text{lm}}(\mathbf{t})) + \lambda_l |\mathbf{t}|$$

其中, $\text{score}(r)$ 表示每条规则的得分, 由特征1-4和特征7-12共同计算得到, 因此也可以把特征1-4和特征7-12看做是规则特征

- 解码**是要找到使 $\text{score}(d)$ 达到最大的翻译推导 d

$$\hat{d} = \arg \max_{d \in D} \text{score}(d)$$

其中 D 表示所有可能的推导构成的搜索空间。广义上来说, 由于句法系统引入了非终结符和复杂的规则, 它的推导空间会远大于短语系统, 因此句法模型的解码器的好坏对性能影响很大

基于树的解码 vs 基于串的解码

- 前面的公式本质上描述了一种基于串的解码，即对输入的源语言句子通过句法模型进行翻译，得到译文串。不过，搜索所有的推导会导致巨大的解码空间。对于树到串和树到树翻译来说，源语言句法树是可见的，因此可以使用另一种解码方法 - 基于树的解码，即把输出入的源语句法树翻译为目标语串

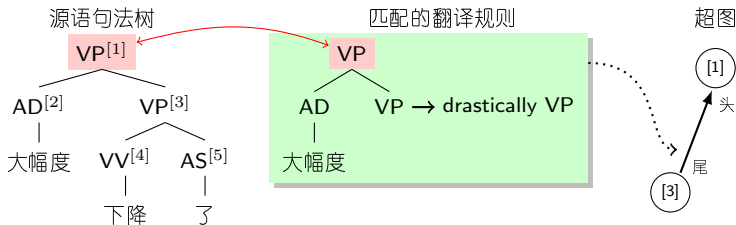
对比	基于树的解码	基于串的解码
解码方法	$\hat{d} = \arg \max_{d \in D_{\text{tree}}} \text{score}(d)$	$\hat{d} = \arg \max_{d \in D} \text{score}(d)$
搜索空间	与输入的源语句法树兼容的推导 D_{tree}	所有推导 D
适用模型	树到串、树到树	所有句法模型
解码算法	chart解码	CKY + 规则二义化
速度	快	一般较慢

基于树的解码 - 超图

- 如果源语言输入的是句法树，**基于树的解码**会找到一个推导覆盖整个句法树，之后输出所对应的目标语词串作为译文
- 比如，可以从树的叶子结点开始，找到所有能匹配到这个节点的规则，当所有节点匹配完之后，本质上获得了一个超图

基于树的解码 - 超图

- 如果源语言输入的是句法树，**基于树的解码**会找到一个推导覆盖整个句法树，之后输出所对应的目标语词串作为译文
- 比如，可以从树的叶子结点开始，找到所有能匹配到这个节点的规则，当所有节点匹配完之后，本质上获得了一个超图
 - 图的节点对应一个句法树句法节点
 - 图的边(或者叫超边)对应规则，边的头指向规则左部(源语言端)所对应图节点，边可以有多个尾，每个尾对应规则右部(源语言端)中的一个变量

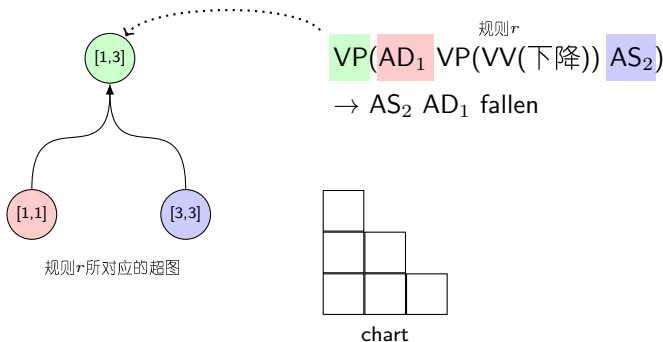


基于树的解码 - chart

- 规则匹配后形成的超图，每个节点可以由两部分信息决定：节点的句法标记 + 跨度
 - ▶ 这本质上和单语句法分析中的表示方法是一致的
 - ▶ 存储形式有很多中，这里采用常用的chart结构，即，用一个二维表存储，其中每一个单元对应一个跨度(span)。同一个跨度的节点都可以放到同一个表单元中，同一表单元的节点用句法标记区分

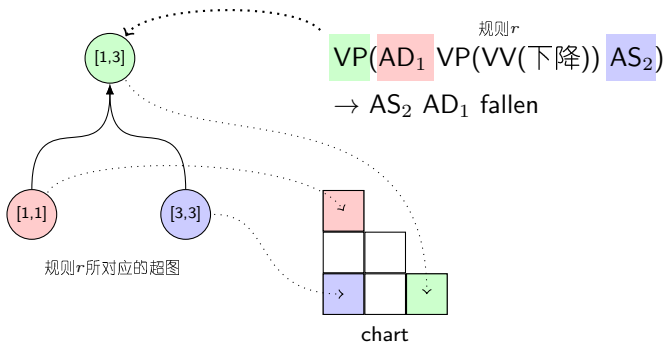
基于树的解码 - chart

- 规则匹配后形成的超图，每个节点可以由两部分信息决定：节点的句法标记 + 跨度
 - 这本质上和单语句法分析中的表示方法是一致的
 - 存储形式有很多中，这里采用常用的chart结构，即，用一个二维表存储，其中每一个单元对应一个跨度(span)。同一个跨度的节点都可以放到同一个表单元中，同一表单元的节点用句法标记区分



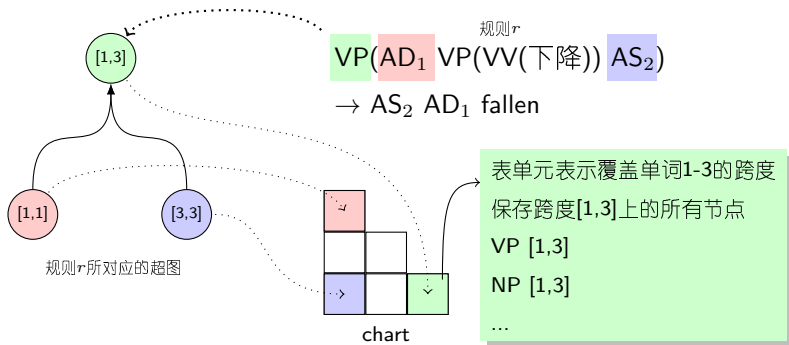
基于树的解码 - chart

- 规则匹配后形成的超图，每个节点可以由两部分信息决定：节点的句法标记 + 跨度
 - 这本质上和单语句法分析中的表示方法是一致的
 - 存储形式有很多中，这里采用常用的chart结构，即，用一个二维表存储，其中每一个单元对应一个跨度(span)。同一个跨度的节点都可以放到同一个表单元中，同一表单元的节点用句法标记区分



基于树的解码 - chart

- 规则匹配后形成的超图，每个节点可以由两部分信息决定：节点的句法标记 + 跨度
 - 这本质上和单语句法分析中的表示方法是一致的
 - 存储形式有很多中，这里采用常用的chart结构，即，用一个二维表存储，其中每一个单元对应一个跨度(span)。同一个跨度的节点都可以放到同一个表单元中，同一表单元的节点用句法标记区分

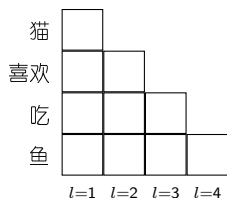


基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

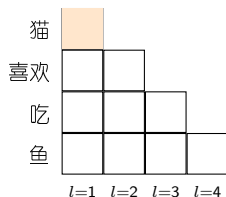
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫

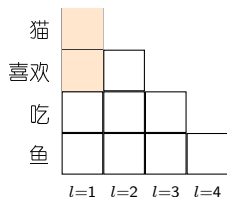
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢

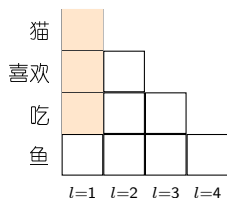
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃

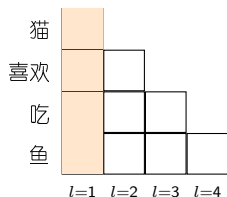
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼

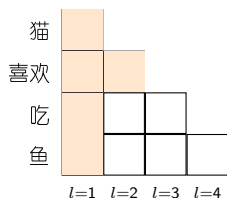
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼
5	[0,2]	N/A	猫 喜欢

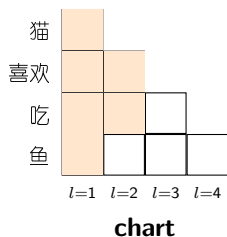
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼
5	[0,2]	N/A	猫 喜欢
6	[1,3]	N/A	喜欢 吃

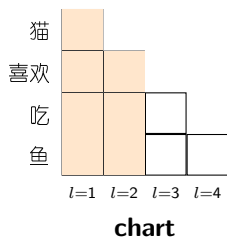
猫 喜欢 吃 鱼

0 1 2 3 4

源语言句子

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



猫 喜欢 吃 鱼

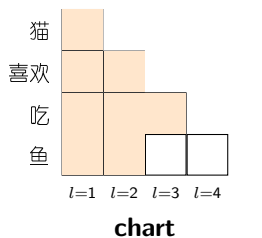
0 1 2 3 4

源语言句子

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼
5	[0,2]	N/A	猫 喜欢
6	[1,3]	N/A	喜欢 吃
7	[2,4]	VP	吃 鱼

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



猫 喜欢 吃 鱼

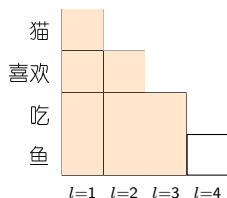
0 1 2 3 4

源语言句子

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼
5	[0,2]	N/A	猫 喜欢
6	[1,3]	N/A	喜欢 吃
7	[2,4]	VP	吃 鱼
8	[0,3]	N/A	猫 喜欢 吃

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



chart

猫 喜欢 吃 鱼

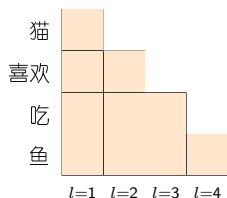
0 1 2 3 4

源语言句子

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼
5	[0,2]	N/A	猫 喜欢
6	[1,3]	N/A	喜欢 吃
7	[2,4]	VP	吃 鱼
8	[0,3]	N/A	猫 喜欢 吃
9	[1,4]	VP	喜欢 吃 鱼

基于树的解码 - 基于chart的方法

- 基于chart这种结构，可以很容易的构建解码所用的超图。常用的方法是自底向上解码：
 - ▶ 从源语言句法树的叶子节点开始，自下而上访问树的节点
 - ▶ 对于每个跨度，如果对应一个树节点，则匹配相应的规则
 - ▶ 从树的根节点可以得到翻译推导，最终选择最优推导所对应的译文输出



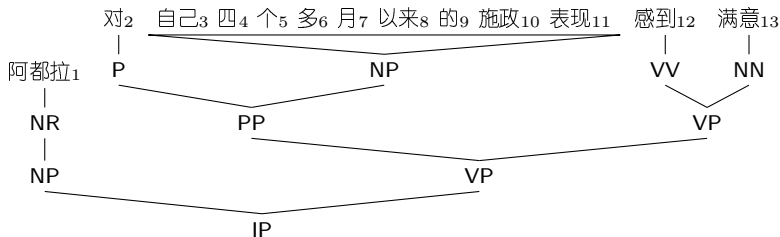
chart

猫 喜欢 吃 鱼
0 1 2 3 4
源语言句子

序号	跨度	标记	源语句子片段
1	[0,1]	NN & NP	猫
2	[1,2]	VV	喜欢
3	[2,5]	VV	吃
4	[3,6]	NN & NP	鱼
5	[0,2]	N/A	猫 喜欢
6	[1,3]	N/A	喜欢 吃
7	[2,4]	VP	吃 鱼
8	[0,3]	N/A	猫 喜欢 吃
9	[1,4]	VP	喜欢 吃 鱼
10	[0,4]	IP (root)	猫 喜欢 吃 鱼

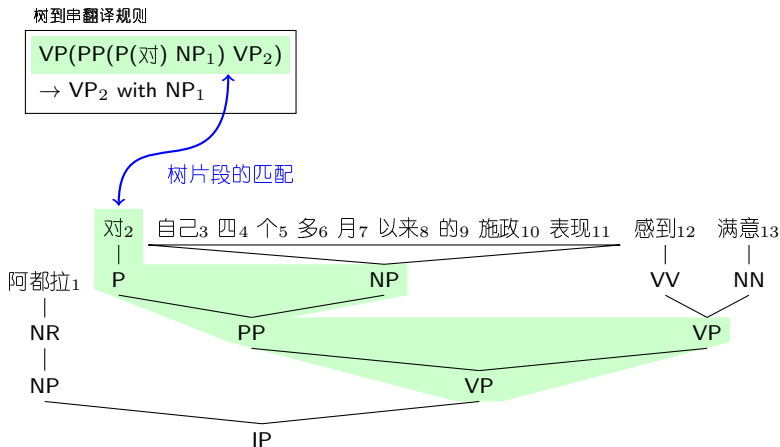
使用树结构匹配树到串规则

- 对于规则的源语言部分，可以使用树片段的匹配找到可以使用这条规则位置
 - 匹配的规则会被存入相应的表格单元中



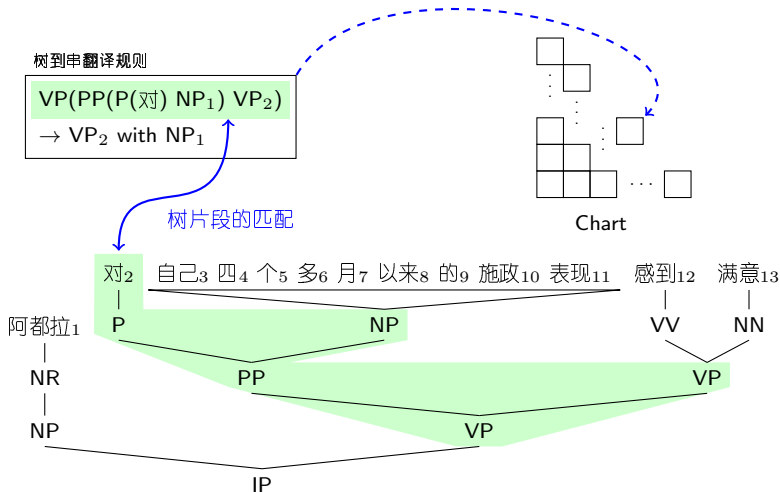
使用树结构匹配树到串规则

- 对于规则的源语言部分，可以使用树片段的匹配找到可以使用这条规则位置
 - 匹配的规则会被存入相应的表格单元中



使用树结构匹配树到串规则

- 对于规则的源语言部分，可以使用树片段的匹配找到可以使用这条规则位置
 - 匹配的规则会被存入相应的表格单元中

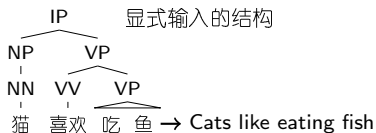


基于串的解码

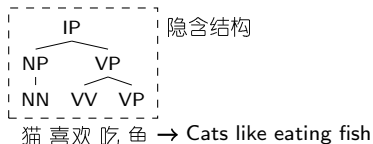
- 不同于基于树的解码，**基于串的解码**方法并不要求输入句法树，它直接对输入词串进行翻译，最终得到译文。
 - ▶ 这种方法适用于树到串、串到树、树到树等多种模型
 - ▶ 本质上，由于并不受固定输入的句法树约束，基于串的解码可以探索更多潜在的树结构，这也增大了搜索空间(相比基于树的解码)，因此该方法更有可能找到高质量翻译结果

基于串的解码

- 不同于基于树的解码，**基于串的解码**方法并不要求输入句法树，它直接对输入词串进行翻译，最终得到译文。
 - ▶ 这种方法适用于树到串、串到树、树到树等多种模型
 - ▶ 本质上，由于并不受固定输入的句法树约束，基于串的解码可以探索更多潜在的树结构，这也增大了搜索空间(相比于基于树的解码)，因此该方法更有可能找到高质量翻译结果
- 在基于串的方法中，句法结构被看做是翻译的隐含变量，而非线性的输入和输出。比如，层次短语翻译解码就是一种典型的基于串的解码方法，所有的翻译推导在翻译过程里动态生成，但是并不要输入或者输出这些推导所对应的层次结构



(a) 基于树的解码



(b) 基于串的解码

基于串的解码 \approx 句法分析

- 基于串的翻译和传统句法分析的任务很像：对于一个输入的词串，找到生成这个词串的最佳推导。唯一不同的地方，在于机器翻译需要考虑译文的生成(语言模型的引入会使问题稍微复杂一些)，但是源语言部分的处理和句法分析一模一样

基于串的解码 \approx 句法分析

- 基于串的翻译和传统句法分析的任务很像：对于一个输入的词串，找到生成这个词串的最佳推导。唯一不同的地方，在于机器翻译需要考虑译文的生成(语言模型的引入会使问题稍微复杂一些)，但是源语言部分的处理和句法分析一模一样
- 这个过程仍然可以用基于chart的方法实现，即对于每一个源语言片段，都匹配可能的翻译规则，之后填入相应的表格单元，这也构成了一个超图，最佳推导可以从这个超图得到

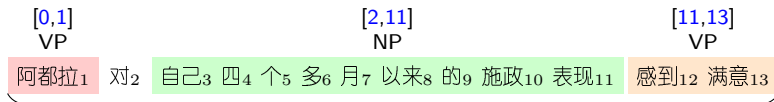
阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

在跨度[0,13]上进行规则匹配

比如：IP(NP₁ VP(PP(P(对) NP₂) VP₃))
→ NP₁ VP₃ with NP₂

基于串的解码 \approx 句法分析

- 基于串的翻译和传统句法分析的任务很像：对于一个输入的词串，找到生成这个词串的最佳推导。唯一不同的地方，在于机器翻译需要考虑译文的生成(语言模型的引入会使问题稍微复杂一些)，但是源语言部分的处理和句法分析一模一样
- 这个过程仍然可以用基于chart的方法实现，即对于每一个源语言片段，都匹配可能的翻译规则，之后填入相应的表格单元，这也构成了一个超图，最佳推导可以从这个超图得到



在跨度[0,13]上进行规则匹配

比如：IP(NP₁ VP(PP(P(对) NP₂) VP₃))
→ NP₁ VP₃ with NP₂

基于串的解码 - 规则匹配

- 相比基于树的解码，基于串的解码的实现要复杂许多，因为对于每一个片段，需要判断每条规则是否能匹配
 - ▶ 就是匹配树片段的叶子节点序列，即单词和变量构成的串

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

在跨度[0,13]上匹配“NP 对 NP VP”

基于串的解码 - 规则匹配

- 相比基于树的解码，基于串的解码的实现要复杂许多，因为对于每一个片段，需要判断每条规则是否能匹配
 - ▶ 就是匹配树片段的叶子节点序列，即单词和变量构成的串
 - ▶ 匹配单词可以直接完成

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

在跨度[0,13]上匹配“NP 对 NP VP”

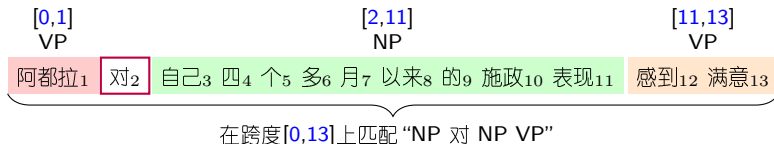
基于串的解码 - 规则匹配

- 相比基于树的解码，基于串的解码的实现要复杂许多，因为对于每一个片段，需要判断每条规则是否能匹配
 - ▶ 就是匹配树片段的叶子节点序列，即单词和变量构成的串
 - ▶ 匹配单词可以直接完成
 - ▶ 匹配变量需要检查相应跨度节点上是否有相应标记的推导



基于串的解码 - 规则匹配

- 相比基于树的解码，基于串的解码的实现要复杂许多，因为对于每一个片段，需要判断每条规则是否能匹配
 - ▶ 就是匹配树片段的叶子节点序列，即单词和变量构成的串
 - ▶ 匹配单词可以直接完成
 - ▶ 匹配变量需要检查相应跨度节点上是否有相应标记的推导



- 如果待匹配的单词和变量序列中，没有连续的变量，这样的规则符合lexicalized norm form (LNF)。因为LNF中单词(终结符)可以作为锚点，因此规则匹配较容易实现
 - ▶ 比如层次短语系统的规则就符合LNF，因此规则匹配非常容易实现
 - ▶ 显然上面例子中的规则不符合LNF

基于串的解码 - 连续变量的匹配

- 但是，如果待匹配串中有连续变量，问题会变得复杂：因为确定两个变量之间的边界需要 增加一重循环

在跨度[0,13]上匹配“NP 对 NP VP”

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

...

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

NP(第二个)

VP

基于串的解码 - 连续变量的匹配

- 但是，如果待匹配串中有连续变量，问题会变得复杂：因为确定两个变量之间的边界需要 增加一重循环

在跨度[0,13]上匹配“NP 对 NP VP”

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

...

阿都拉₁ 对₂ 自己₃ 四₄ 个₅ 多₆ 月₇ 以来₈ 的₉ 施政₁₀ 表现₁₁ 感到₁₂ 满意₁₃

NP(第二个)

VP

- 理论上，对于长度为 n 的词串，匹配 m 个连续变量的时间复杂度是 $O(n^{m-1})$
 - 这也会导致含有多个变量的非词汇化规则的匹配大大增加系统的运行时间，但这种规则在句法系统中也很常见

基于串的解码 - CKY + 规则二叉化

- 对于这个问题，常用的解决办法是进行规则二叉化，这样右端最多只有两个连续变量，规则匹配的复杂度降为 $O(n)$ 。例如，对于如下串到树规则

喜欢 $VP_1 NP_2 \rightarrow VP(VBZ(\text{likes}) VP_1 NP_2)$

二叉化之后变为

喜欢 $V103 \rightarrow VP(VBZ(\text{likes}) V103)$

$VP_1 NP_2 \rightarrow V103(VP_1 NP_2)$

其中，二叉化后的规则源语言端最多有两个非终结符。 $V103$ 是一个虚拟符号，用于表示临时生成的规则

基于串的解码 - CKY + 规则二叉化

- 对于这个问题，常用的解决办法是进行规则二叉化，这样右端最多只有两个连续变量，规则匹配的复杂度降为 $O(n)$ 。例如，对于如下串到树规则

喜欢 $VP_1 NP_2 \rightarrow VP(VBZ(\text{likes}) VP_1 NP_2)$

二叉化之后变为

喜欢 $V103 \rightarrow VP(VBZ(\text{likes}) V103)$

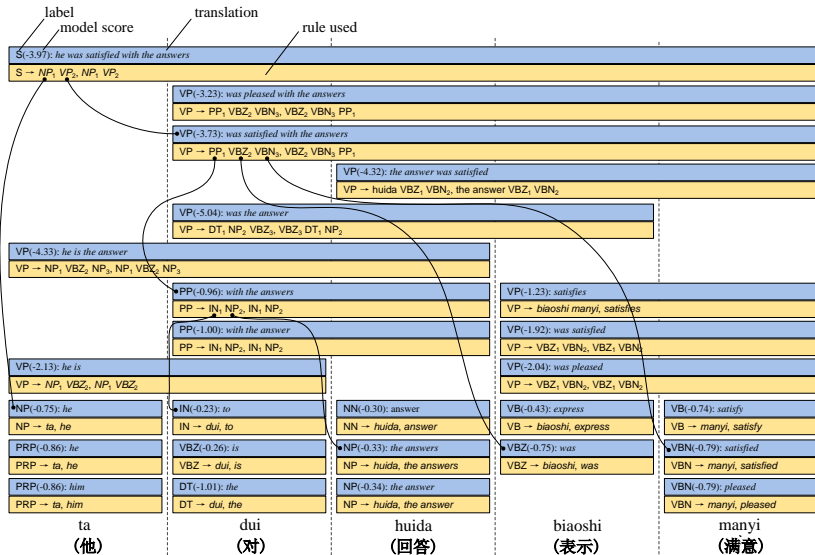
$VP_1 NP_2 \rightarrow V103(VP_1 NP_2)$

其中，二叉化后的规则源语言端最多有两个非终结符。 $V103$ 是一个虚拟符号，用于表示临时生成的规则

- 对于二叉化后的规则，可以使用CKY方法完成解码，它也是一种基于chart的分析方法，对于每个源语言片段，匹配规则两个枝杈的左分支和右分支，整个过程和其它chart方法没有区别

基于串的解码 - 规则使用的实例

- 使用规则可以构建句子的分析图



改进方法

- **基于森林的翻译模型。**句法分析会出现错误，因此只使用一棵句法树进行规则抽取和解码会放大句法分析错误的影响。一种解决方法是使用多棵句法树增加覆盖度，句法森林是一种有效的数据结构表示指数级树结构，因此也被用于基于句法的机器翻译。
- **句法软约束和规则模糊匹配。**前面提到的模型都要求模型严格遵循句法结构，很多时候由于句法结构可能不完全适合翻译任务甚至有错误，这种模型过“硬”。因此可以使用句法软约束或者放松规则匹配时的约束。
- **控制句法使用的程度。**句法模型比较适合捕捉句法上层的表示，而短语模型更适合处理局部依赖。因此可以使用二者的混合来达到更好的效果，比如，可以让句法模型处理上层骨架的翻译，之后让短语模型处理简单短语片段的翻译。

翻译效果

模型		开发集 (BLEU[%])	测试集 (BLEU[%])
短语(Moses)		36.51	34.93
短语(NiuTrans)		36.99	35.29
层次短语(Moses)		36.65	34.79
层次短语(NiuTrans)		37.41	35.35
树到串 (NiuTrans)	基于串的解码	36.48	34.71
	基于树的解码	35.54	33.99
	基于森林的解码	36.14	34.25
树到树 (NiuTrans)	基于串的解码	35.99	34.01
	基于树的解码	35.04	33.21
	基于森林的解码	35.56	33.45
串到树 (NiuTrans)	基于串的解码	37.63	35.65

* 以上结果来自 NiuTrans: An Open Source Toolkit for Phrase-based and Syntax-based Machine Translation

* 开发集: NIST MT03, 测试集: NIST MT05

小结一下

- 基于短语和基于句法的模型是机器翻译2000年之后的重要进展之一
 - ▶ 研究热度持续十余年，至今仍有使用(无指导机器翻译中使用SMT做初始模型)
 - ▶ 一些方法代表了NLP中的原始创新，比如最小错误率训练
- 相关技术和方法对现在研究仍然有很好的借鉴意义
 - ▶ 对翻译的统计建模方法，比如，基于翻译文法和推导的机器翻译建模思想
 - ▶ 翻译调序等机器翻译特有问题的描述方法
 - ▶ 先验知识的使用，句法结构、篇章等等

小结一下

- 基于短语和基于句法的模型是机器翻译2000年之后的重要进展之一
 - ▶ 研究热度持续十余年，至今仍有使用(无指导机器翻译中使用SMT做初始模型)
 - ▶ 一些方法代表了NLP中的原始创新，比如最小错误率训练
- 相关技术和方法对现在研究仍然有很好的借鉴意义
 - ▶ 对翻译的统计建模方法，比如，基于翻译文法和推导的机器翻译建模思想
 - ▶ 翻译调序等机器翻译特有问题的描述方法
 - ▶ 先验知识的使用，句法结构、篇章等等
- 在深度学习时代下重新审视统计机器翻译
 - ▶ 注意，统计机器翻译并不是简单几套系统，更重要的是思想，这种建模方法更接近人类对翻译的认知
 - ▶ 深度学习方法从另一个视角看待机器翻译，二者必然存在结合的可能，只是结合的方法需要探索

Last Slide

