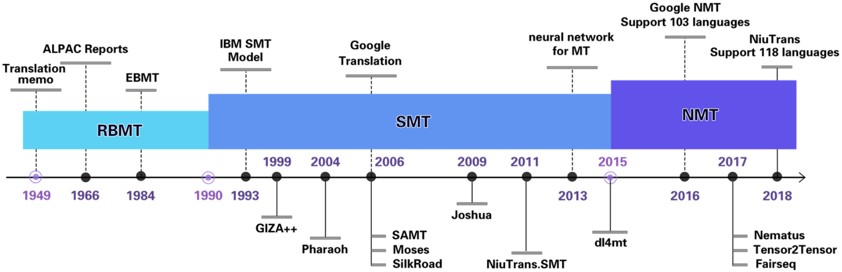
从"词"开始：基于单词的翻译模型

肖桐　朱靖波

xiaotong@mail.neu.edu.cn
zhujingbo@mail.neu.edu.cn

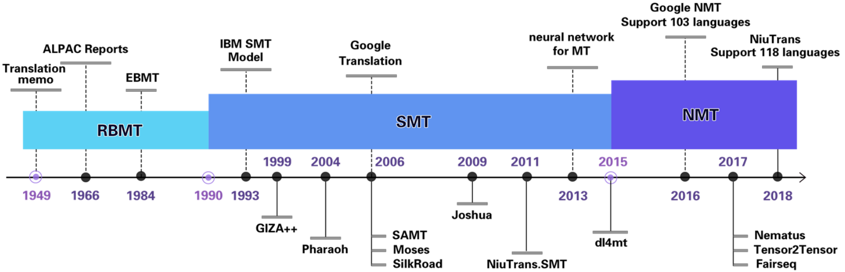
东北大学 自然语言处理实验室
http://www.nlplab.com

Landscape



Google NMT
Support 103 languages
neural network
for MT
NiuTrans
Support 118 languages
ALPAC Reports
IBM SMT
Model
Google
Translation
Translation
memo
EBMT
RBMT
SMT
NMT
1999
2004
2006
2009
2011
2015
2017
1949
1966
1984
1990
1993
2013
2016
2018
GIZA++
Pharaoh
SAMT
Moses
SilkRoad
Joshua
NiuTrans.SMT
dl4mt
Nematus
Tensor2Tensor
Fairseq

Landscape



Sections 3-4:
统计机器翻译

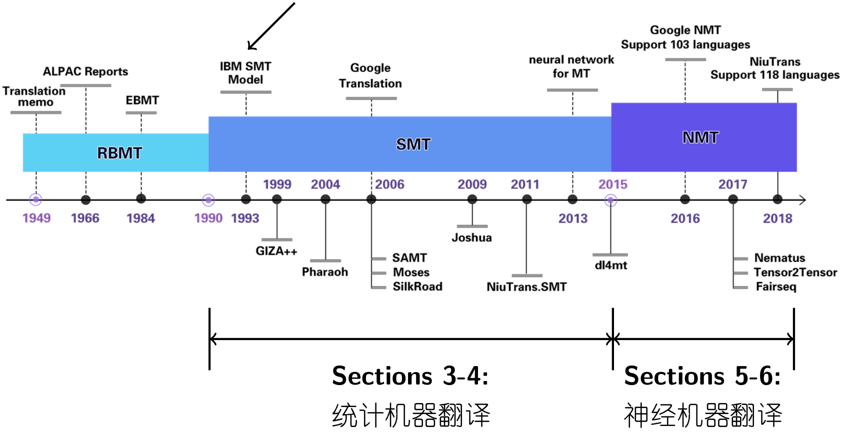
Sections 5-6:
神经机器翻译

Landscape

本章内容（Section 3）：

统计机器翻译开山之作 - IBM模型

The Mathematics of Statistical Machine Translation: Parameter Estimation by Peter E. Brown et al., 1993.



按单词进行翻译

基于单词的(统计)机器翻译思想

两个互译句子之间存在一种单词间的对应，而整句的翻译是由单词的翻译"组成"

我　对　你　感到　满意

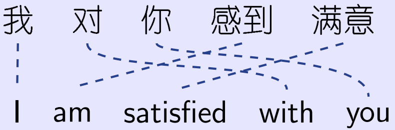
⇕

I　am　satisfied　with　you

按单词进行翻译

基于单词的(统计)机器翻译思想

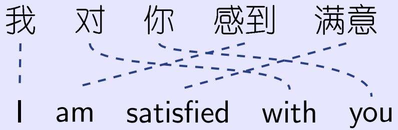
两个互译句子之间存在一种单词间的对应，而整句的翻译是由单词的翻译"组成"



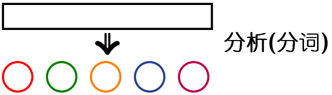
按单词进行翻译

基于单词的(统计)机器翻译思想

两个互译句子之间存在一种单词间的对应，而整句的翻译是由单词的翻译"组成"



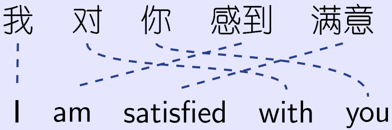
- 传统观点下的(基于词)的翻译过程 - 三个步骤
  1. **分析**：将输入句子进行分词



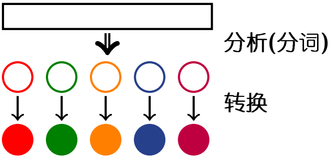
按单词进行翻译

基于单词的(统计)机器翻译思想

两个互译句子之间存在一种单词间的对应，而整句的翻译是由单词的翻译"组成"



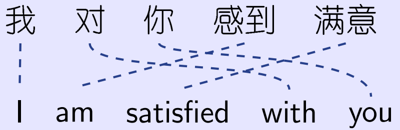
- 传统观点下的(基于词)的翻译过程 - 三个步骤
  1. **分析**：将输入句子进行分词
  2. **转换**：获得每个源语单词的翻译



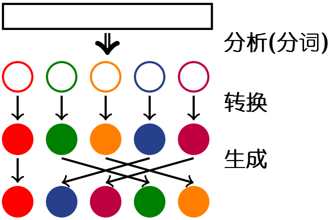
按单词进行翻译

基于单词的(统计)机器翻译思想

两个互译句子之间存在一种单词间的对应，而整句的翻译是由单词的翻译"组成"



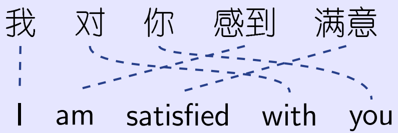
- 传统观点下的(基于词)的翻译过程 - 三个步骤
  1. **分析**：将输入句子进行分词
  2. **转换**：获得每个源语单词的翻译
  3. **生成**：将单词的翻译组成通顺的整句译文



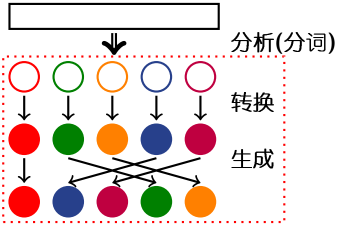
按单词进行翻译

基于单词的(统计)机器翻译思想

两个互译句子之间存在一种**单词**间的对应，而整句的翻译是由**单词**的翻译"组成"



- 传统观点下的(基于词)的翻译过程 - 三个步骤
  1. **分析**：将输入句子进行分词
  2. **转换**：获得每个源语单词的翻译
  3. **生成**：将单词的翻译组成通顺的整句译文



本章的重点内容为**转换和生成**

Outline

1. 一个简单的翻译实例

2. IBM模型
   - 建模
   - 解码
   - 模型训练

我们人是如何进行翻译的？

待翻译句子(已经分词):

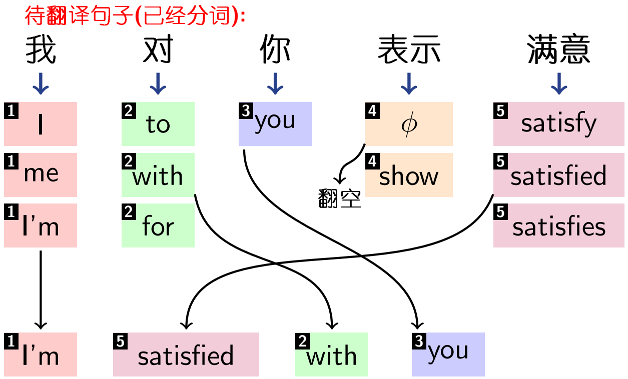
我 对 你 表示 满意

我们人是如何进行翻译的？

待翻译句子(已经分词):

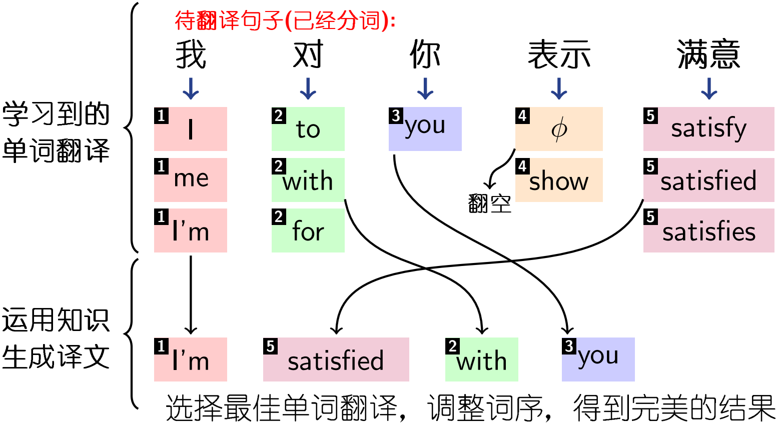
我	对	你	表示	满意
↓	↓	↓	↓	↓
1 I	2 to	3 you	4 ϕ	5 satisfy
1 me	2 with		4 show	5 satisfied
1 I'm	2 for			5 satisfies

我们人是如何进行翻译的？

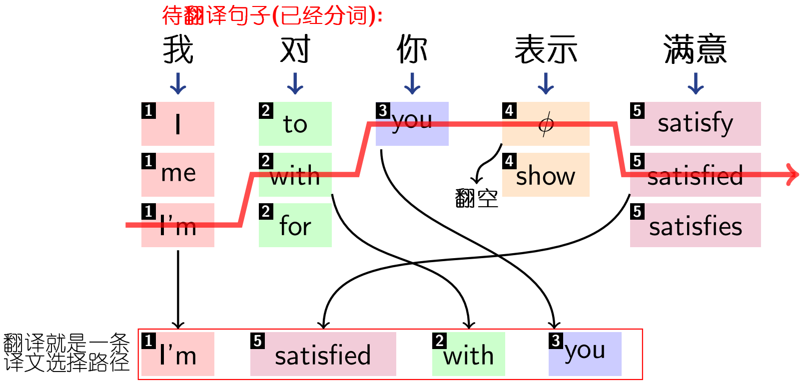


选择最佳单词翻译，调整词序，得到完美的结果

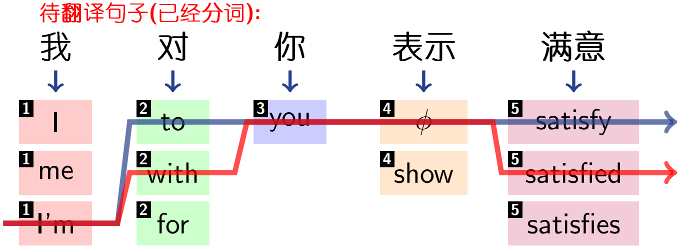
我们人是如何进行翻译的？



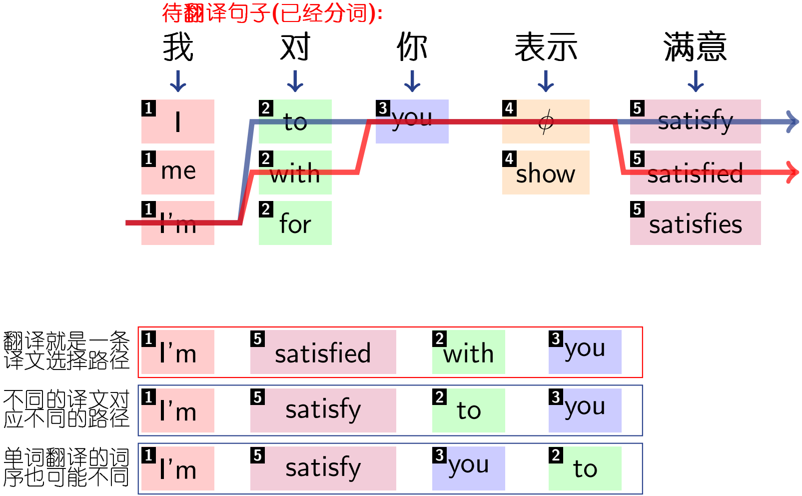
机器翻译系统是如何进行翻译的？



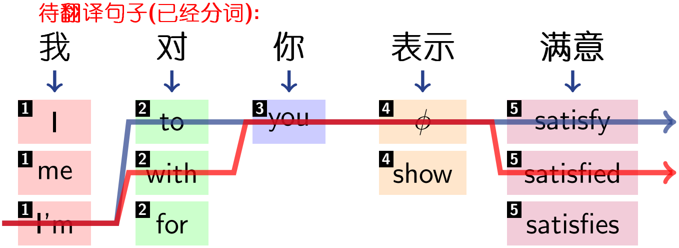
机器翻译系统是如何进行翻译的？



机器翻译系统是如何进行翻译的？

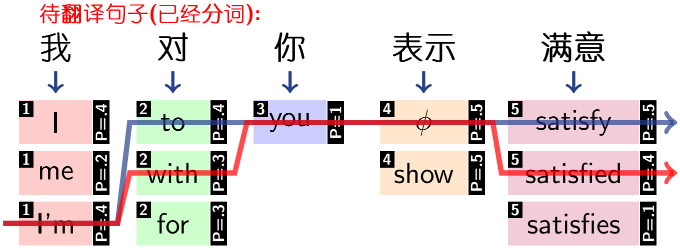


机器翻译系统是如何进行翻译的？



翻译就是一条译文选择路径	1 I'm	5 satisfied	2 with	3 you
不同的译文对应不同的路径	1 I'm	5 satisfy	2 to	3 you
单词翻译的词序也可能不同	1 I'm	5 satisfy	3 you	2 to
可能的翻译路径非常多	...			

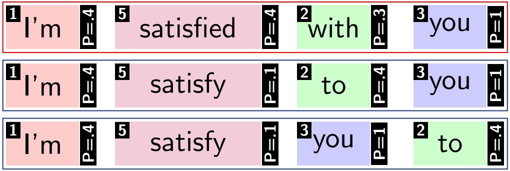
机器翻译系统是如何进行翻译的？



翻译就是一条译文选择路径

不同的译文对应不同的路径

单词翻译的词序也可能不同

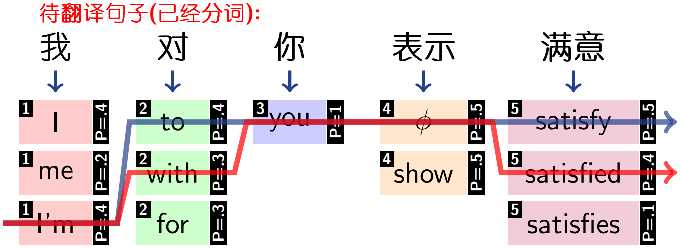


可能的翻译路径非常多

…

所有翻译单元都是概率化的 P=概率

机器翻译系统是如何进行翻译的？

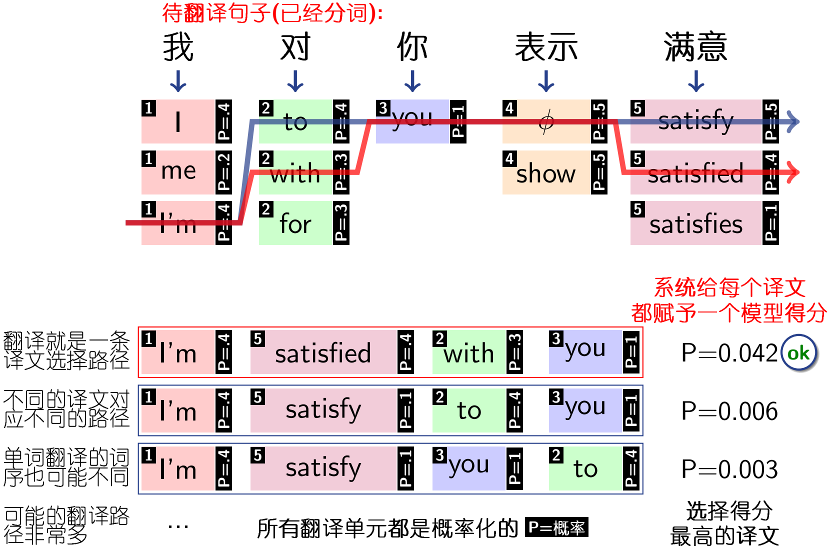


系统给每个译文都赋予一个模型得分

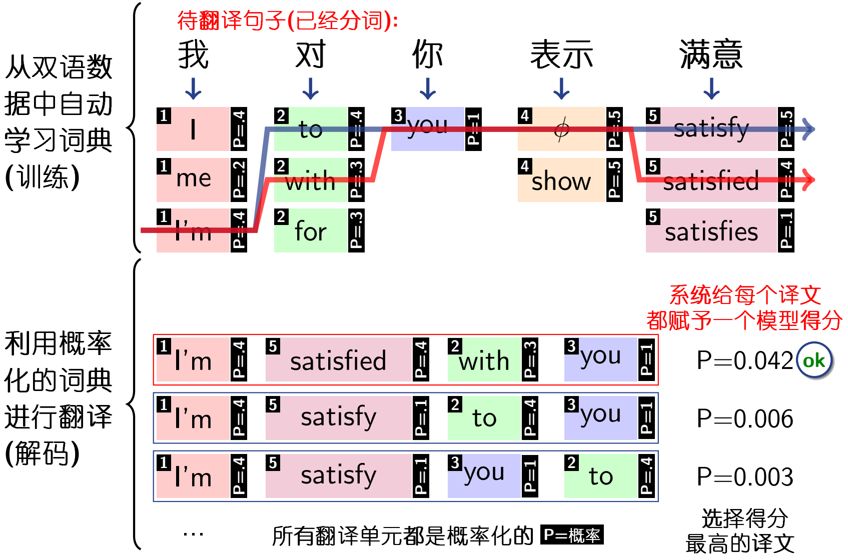
翻译就是一条译文选择路径	1 I'm P=.4	5 satisfied P=.4	2 with P=.3	3 you P=1	P=0.042
不同的译文对应不同的路径	1 I'm P=.4	5 satisfy P=.1	2 to P=.4	3 you P=1	P=0.006
单词翻译的词序也可能不同	1 I'm P=.4	5 satisfy P=.1	3 you P=1	2 to P=.4	P=0.003

可能的翻译路径非常多 ⋯ 所有翻译单元都是概率化的 P=概率

机器翻译系统是如何进行翻译的？

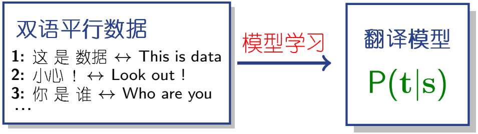


机器翻译系统是如何进行翻译的？



实现一个简单的机器翻译系统：学习单词翻译概率

- 构建一个简单的机器翻译系统
  1. 模型学习 - 从双语平行数据中学习单词翻译知识



实现一个简单的机器翻译系统：学习单词翻译概率

- 构建一个简单的机器翻译系统
  1. 模型学习 - 从双语平行数据中学习单词翻译知识
  2. 解码 - 利用单词翻译知识，对新的句子进行翻译

双语平行数据

1: 这 是 数据 $\leftrightarrow$ This is data
2: 小心 ！ $\leftrightarrow$ Look out !
3: 你 是 谁 $\leftrightarrow$ Who are you
...

$\xrightarrow{\text{模型学习}}$

翻译模型

$\mathrm{P}(\mathbf{t}|\mathbf{s})$

$\xrightarrow{\text{穷举\&计算}}$

机器翻译引擎

对任意句子进行翻译

实现一个简单的机器翻译系统：学习单词翻译概率

- 构建一个简单的机器翻译系统
  1. 模型学习 - 从双语平行数据中学习单词翻译知识
  2. 解码 - 利用单词翻译知识，对新的句子进行翻译

双语平行数据

1: 这 是 数据 $\leftrightarrow$ This is data
2: 小心 ！ $\leftrightarrow$ Look out !
3: 你 是 谁 $\leftrightarrow$ Who are you
...

$\xrightarrow{\text{模型学习}}$

翻译模型

P($\mathbf{t}$|$\mathbf{s}$)

$\xrightarrow{\text{穷举\&计算}}$

机器翻译引擎

对任意句子
进行翻译

- **步骤1**：构建单词翻译表 - 翻译词典

对于任意的源语言单词x，要获得它所有可能的译文Y。给定一个互译句对$(\mathbf{s},\mathbf{t})$，对于$y \in Y$，定义$\mathrm{P}(x \leftrightarrow y;\mathbf{s},\mathbf{t})$表示$x$和$y$在$(x,y)$中互译的概率，我们用$x$和$y$的联合概率表示：

$$
\begin{aligned}
\mathrm{P}(x \leftrightarrow y;\mathbf{s},\mathbf{t}) &\equiv \mathrm{P}(x,y;\mathbf{s},\mathbf{t}) \\
&= \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}
\end{aligned}
$$

$c(x,y;\mathbf{s},\mathbf{t})$表示$(x,y)$在$(\mathbf{s},\mathbf{t})$中共现的次数；
$\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})$表示$(\mathbf{s},\mathbf{t})$中任意源/译文单词共现的总次数

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x, y; \mathbf{s}, \mathbf{t}) = \frac{c(x, y; \mathbf{s}, \mathbf{t})}{\sum_{x', y'} c(x', y'; \mathbf{s}, \mathbf{t})}$$

$\mathbf{s} =$ 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t} =$ machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'}; \mathbf{s}, \mathbf{t}) = ?$

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) \quad = \quad \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s} =$ 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t} =$ machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'}; \mathbf{s}, \mathbf{t}) = 1$

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) \quad = \quad \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s} =$ 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t} =$ machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'}; \mathbf{s}, \mathbf{t}) = 1 + 1$

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) \quad = \quad \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s}$ = 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t}$ = machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'}; \mathbf{s}, \mathbf{t}) = 1 + 1 + 1$

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) \quad = \quad \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s} =$ 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t} =$ machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'}; \mathbf{s},\mathbf{t}) = 1+1+1+1 = 4$

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) = \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s}$ = 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t}$ = machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'}; \mathbf{s},\mathbf{t}) = 1+1+1+1 = 4$
- $\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t}) = \text{使劲数...} = 63$

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) \quad = \quad \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s}$ = 机器 翻译 就 是 用 计算机 来 进行 翻译
$\mathbf{t}$ = machine translation is just translation by computer

- $c(\text{'翻译'}, \text{'translation'};\mathbf{s},\mathbf{t}) = 1+1+1+1 = 4$
- $\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t}) = \text{使劲数...} = 63 = 9 \times 7 = |\mathbf{s}| \times |\mathbf{t}|$
  - $|\cdot|$表示句子长度

实现一个简单的机器翻译系统：学习单词翻译概率(2)

$$\mathrm{P}(x,y;\mathbf{s},\mathbf{t}) = \frac{c(x,y;\mathbf{s},\mathbf{t})}{\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t})}$$

$\mathbf{s}$ = 机器 翻译 就 是 用 计算机 来 进行 翻译

$\mathbf{t}$ = machine translation is just translation by computer

- $c(\text{'翻译'},\text{'translation'};\mathbf{s},\mathbf{t}) = 1+1+1+1 = 4$
- $\sum_{x',y'} c(x',y';\mathbf{s},\mathbf{t}) =$ 使劲数... $= 63 = 9 \times 7 = |\mathbf{s}| \times |\mathbf{t}|$
  - $|\cdot|$表示句子长度
- '翻译'和'translation'的互译概率为

$$\mathrm{P}(\text{'翻译'},\text{'translation'};\mathbf{s},\mathbf{t}) = 4/63$$

类似的

$$\mathrm{P}(\text{'机器'},\text{'translation'};\mathbf{s},\mathbf{t}) = 2/63$$
$$\mathrm{P}(\text{'机器'},\text{'look'};\mathbf{s},\mathbf{t}) = 0/63$$

实现一个简单的机器翻译系统：学习单词翻译概率(3)

- 如果有多个互译句对$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$，称之为双语平行数据(语料)。翻译概率可以被定义为

$$\mathrm{P}(x,y) = \frac{\sum_{k=1}^{K} c(x,y;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{k=1}^{K}\sum_{x',y'} c(x',y';\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

实现一个简单的机器翻译系统：学习单词翻译概率(3)

- 如果有多个互译句对$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$，称之为双语平行数据(语料)。翻译概率可以被定义为

$$\mathrm{P}(x,y) = \frac{\sum_{k=1}^{K} c(x,y;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{k=1}^{K}\sum_{x',y'} c(x',y';\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

- 说白了就是计算(x,y)的频次时，在每个句子上累加

$\mathbf{s}^1$ = 机器 翻译 就 是 用 计算机 进行 翻译
$\mathbf{t}^1$ = Machine translation is just translation by computer

$\mathbf{s}^2$ = 那 人工 翻译 呢 ?
$\mathbf{t}^2$ = So , what is human translation ?

$$\begin{aligned} & \mathrm{P}(\text{'翻译'},\text{'translation'}) \\ = & \frac{c(\text{'翻译'},\text{'translation'};\mathbf{s}^{[1]},\mathbf{t}^{[1]}) + c(\text{'翻译'},\text{'translation'};\mathbf{s}^{[2]},\mathbf{t}^{[2]})}{\sum_{x',y'} c(x',y';\mathbf{s}^{[1]},\mathbf{t}^{[1]}) + \sum_{x',y'} c(x',y';\mathbf{s}^{[2]},\mathbf{t}^{[2]})} \end{aligned}$$

实现一个简单的机器翻译系统：学习单词翻译概率(3)

- 如果有多个互译句对$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$，称之为双语平行数据(语料)。翻译概率可以被定义为

$$\mathrm{P}(x,y) = \frac{\sum_{k=1}^{K} c(x,y;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{k=1}^{K}\sum_{x',y'} c(x',y';\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

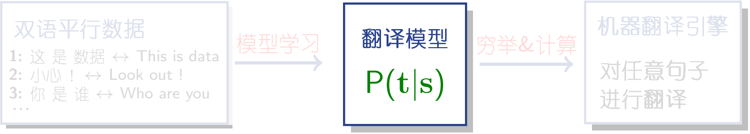
- 说白了就是计算(x,y)的频次时，在每个句子上累加

$\mathbf{s}^1$ = 机器 翻译 就 是 用 计算机 进行 翻译
$\mathbf{t}^1$ = Machine translation is just translation by computer

$\mathbf{s}^2$ = 那 人工 翻译 呢 ?
$\mathbf{t}^2$ = So , what is human translation ?

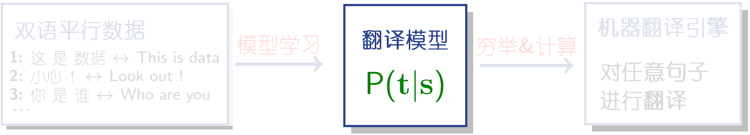
$$\begin{aligned}
&\mathrm{P}(\text{'翻译'},\text{'translation'}) \\
= &\frac{c(\text{'翻译'},\text{'translation'};\mathbf{s}^{[1]},\mathbf{t}^{[1]}) + c(\text{'翻译'},\text{'translation'};\mathbf{s}^{[2]},\mathbf{t}^{[2]})}{\sum_{x',y'} c(x',y';\mathbf{s}^{[1]},\mathbf{t}^{[1]}) + \sum_{x',y'} c(x',y';\mathbf{s}^{[2]},\mathbf{t}^{[2]})} \\
= &\frac{4+1}{|\mathbf{s}^{[1]}|\times|\mathbf{t}^{[1]}| + |\mathbf{s}^{[2]}|\times|\mathbf{t}^{[2]}|} = \frac{4+1}{9\times 7+5\times 7} = \frac{5}{98}
\end{aligned}$$

实现一个简单的机器翻译系统：句子级翻译模型



- **步骤2**: 对任意的句对$(\mathbf{s},\mathbf{t})$计算句子级翻译概率$\mathrm{P}(\mathbf{t}|\mathbf{s})$

实现一个简单的机器翻译系统：句子级翻译模型



- **步骤2**: 对任意的句对$(\mathbf{s},\mathbf{t})$计算句子级翻译概率$\mathrm{P}(\mathbf{t}|\mathbf{s})$

  用一种比较简单的思路：定义$(\mathbf{s},\mathbf{t})$上的一种分数$g(\mathbf{s},\mathbf{t})$
  - $g(\mathbf{s},\mathbf{t})$的值越大翻译质量越好
  - $g(\mathbf{s},\mathbf{t})$的值越小翻译质量越差

实现一个简单的机器翻译系统：句子级翻译模型

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
...

模型学习 →

翻译模型
P(t|s)

穷举&计算 →

机器翻译引擎
对任意句子
进行翻译

- **步骤2**: 对任意的句对$(\mathbf{s},\mathbf{t})$计算句子级翻译概率$\mathrm{P}(\mathbf{t}|\mathbf{s})$

  用一种比较简单的思路：定义$(\mathbf{s},\mathbf{t})$上的一种分数$g(\mathbf{s},\mathbf{t})$
  - $g(\mathbf{s},\mathbf{t})$的值越大翻译质量越好
  - $g(\mathbf{s},\mathbf{t})$的值越小翻译质量越差

  于是，我们进一步定义

$$\mathrm{P}(\mathbf{t}|\mathbf{s}) = \frac{g(\mathbf{s},\mathbf{t})}{\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')}$$

实际上就是对$g(\mathbf{s},\mathbf{t})$在所有可能的译文集合上做归一化，使其具有概率意义

实现一个简单的机器翻译系统：句子级翻译模型(2)

- 两个问题

1. 如何计算$g(\mathbf{s},\mathbf{t})$
2. 如何计算$\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')$

实现一个简单的机器翻译系统：句子级翻译模型(2)

- 两个问题
  1. 如何计算$g(\mathbf{s},\mathbf{t})$ - 最关键的建模问题，马上开始
  2. 如何计算$\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')$ - 实际上不用计算，后面再说

实现一个简单的机器翻译系统：句子级翻译模型(2)

- 两个问题
  1. 如何计算$g(\mathbf{s},\mathbf{t})$ - 最关键的建模问题，马上开始
  2. 如何计算$\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')$ - 实际上不用计算，后面再说
- **对$g(\mathbf{s},\mathbf{t})$建模：**根据本章第一页的假设，$\mathbf{s}$与$\mathbf{t}$之间存在一种单词间的对应，我们称之为词对齐关系

$\mathbf{s} =$ 我$_1$ 对$_2$ 你$_3$ 感到$_4$ 满意$_5$

$\mathbf{t} =$ I$_1$ am$_2$ satisfied$_3$ with$_4$ you$_5$

每根虚线代表一个对齐链接，记为(j,i)，意思是源语言的第j个单词对应目标语言第i个单词，即s_j与t_i对应。

- 所有对齐链接构成集合A，对于上例：
  $A = \{(1,1),(2,4),(3,5),(4,2),(5,3)\}$

实现一个简单的机器翻译系统：句子级翻译模型(3)

- 给定一个句对$(\mathbf{s},\mathbf{t})$，及它们之间的(最优)词对齐$\hat{A}$，可以定义模型得分为：

$$g(\mathbf{s},\mathbf{t}) \equiv \prod_{(j,i)\in\hat{A}} \mathrm{P}(s_j,t_i)$$

显然每个单词翻译概率都高，那么整句的模型得分也高

$\mathbf{s} =$ 我$_1$ 对$_2$ 你$_3$ 感到$_4$ 满意$_5$

$\mathbf{t} =$ I$_1$ am$_2$ satisfied$_3$ with$_4$ you$_5$

$$\begin{aligned} g(\mathbf{s},\mathbf{t}) \quad = \quad & \mathrm{P}(\text{'我'},\text{'I'}) \times \mathrm{P}(\text{'对'},\text{'with'}) \times \mathrm{P}(\text{'你'},\text{'you'}) \times \\ & \mathrm{P}(\text{'感到'},\text{'am'}) \times \mathrm{P}(\text{'满意'},\text{'satisfied'}) \end{aligned}$$

实现一个简单的机器翻译系统：句子级翻译模型(3)

- 给定一个句对$(\mathbf{s},\mathbf{t})$，及它们之间的(最优)词对齐$\hat{A}$，可以定义模型得分为：

$$g(\mathbf{s},\mathbf{t}) = \prod_{(j,i)\in\hat{A}} \mathrm{P}(s_j,t_i)$$

显然每个单词翻译概率都高，那么整句的模型得分也高

$$\mathbf{s} = \text{我}_1 \quad \text{对}_2 \quad \text{你}_3 \quad \text{感到}_4 \quad \text{满意}_5$$

$$\mathbf{t} = \text{I}_1 \quad \text{am}_2 \quad \text{satisfied}_3 \quad \text{with}_4 \quad \text{you}_5$$

$$\begin{aligned} g(\mathbf{s},\mathbf{t}) \quad = \quad & \mathrm{P}(\text{'我'},\text{'I'}) \times \mathrm{P}(\text{'对'},\text{'with'}) \times \mathrm{P}(\text{'你'},\text{'you'}) \times \\ & \mathrm{P}(\text{'感到'},\text{'am'}) \times \mathrm{P}(\text{'满意'},\text{'satisfied'}) \end{aligned}$$

- **其它问题**：如果有单词翻译为空怎么办？
  - 简单的解决方案是平滑，给予这种翻译一个很小的概率值
  - 或者在建模中考虑，见后面的IBM模型

实现一个简单的机器翻译系统：句子级翻译模型(4)

- 但是，这样设计的$g(\mathbf{s},\mathbf{t})$没有考虑词序的信息。相同译词出现在不同的位置，得分相同 - 无法选择流畅的译文

	$\prod_{(j,i)\in\hat{A}} \mathrm{P}(s_j,t_i)$
$\mathbf{s} =$ 我$_1$ 对$_2$ 你$_3$ 感到$_4$ 满意$_5$ $\mathbf{t}' =$ I$_1$ am$_2$ satisfied$_3$ with$_4$ you$_5$	0.0023
$\mathbf{s} =$ 我$_1$ 对$_2$ 你$_3$ 感到$_4$ 满意$_5$ $\mathbf{t}'' =$ I$_1$ with$_2$ you$_3$ am$_4$ satisfied$_5$	0.0023

实现一个简单的机器翻译系统：句子级翻译模型(4)

- 但是，这样设计的$g(\mathbf{s},\mathbf{t})$没有考虑词序的信息。相同译词出现在不同的位置，得分相同 - 无法选择流畅的译文

	$\prod_{(j,i)\in\hat{A}} \mathrm{P}(s_j,t_i)$ $\times \mathrm{P}_{lm}(\mathbf{t})$
$\mathbf{s} =$ 我$_1$ 对$_2$ 你$_3$ 感到$_4$ 满意$_5$ $\mathbf{t}' =$ I$_1$ am$_2$ satisfied$_3$ with$_4$ you$_5$	0.0023×0.0107
$\mathbf{s} =$ 我$_1$ 对$_2$ 你$_3$ 感到$_4$ 满意$_5$ $\mathbf{t}'' =$ I$_1$ with$_2$ you$_3$ am$_4$ satisfied$_5$	0.0023×0.0009

- 解决方案：引入语言模型$\mathrm{P}_{lm}(\mathbf{t})$来度量译文的流畅度

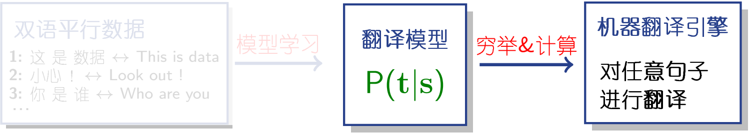
$\mathrm{P}_{\text{2-gram}}(w_1 ... w_m) =$

$\mathrm{P}(w_1) \times \mathrm{P}(w_2|w_1) \times \mathrm{P}(w_3|w_2) ... \times \mathrm{P}(w_m|w_{m-1})$

- 最终

$$g(\mathbf{s},\mathbf{t}) = \prod_{(j,i)\in\hat{A}} \mathrm{P}(s_j,t_i) \times \mathrm{P}_{lm}(\mathbf{t})$$

实现一个简单的机器翻译系统：解码



- **步骤3：解码** - 对任意的$\mathbf{s}$，找到翻译概率最大的译文$\hat{\mathbf{t}}$

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

这里$\arg\max_{\mathbf{a}} f(\mathbf{a})$表示找到使$f(\mathbf{a})$达到最大的$\mathbf{a}$输出

实现一个简单的机器翻译系统：解码

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
...

模型学习 →

翻译模型
$\mathrm{P}(\mathbf{t}|\mathbf{s})$

穷举&计算 →

机器翻译引擎
对任意句子
进行翻译

- **步骤3：解码** - 对任意的$\mathbf{s}$，找到翻译概率最大的译文$\hat{\mathbf{t}}$

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

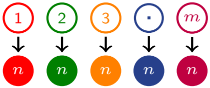
这里$\arg\max_{\mathbf{a}} f(\mathbf{a})$表示找到使$f(\mathbf{a})$达到最大的$\mathbf{a}$输出

- 现在我们可以对任意的$(\mathbf{s},\mathbf{t})$计算$\mathrm{P}(\mathbf{t}|\mathbf{s}) = \frac{g(\mathbf{s},\mathbf{t})}{\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')}$
  - 给定$\mathbf{s}$，$\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')$是个常数(因为$\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')$的变量只有$\mathbf{s}$)
  - 这样，我们得到解码步骤的形式化描述为

$$\begin{aligned}\hat{\mathbf{t}} &= \arg\max_{\mathbf{t}} \frac{g(\mathbf{s},\mathbf{t})}{\sum_{\mathbf{t}'} g(\mathbf{s},\mathbf{t}')} \\ &= \arg\max_{\mathbf{t}} g(\mathbf{s},\mathbf{t})\end{aligned}$$

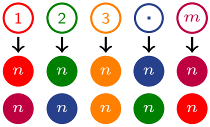
实现一个简单的机器翻译系统：解码(2)

- **解码的核心问题**是在所有可能的翻译结果中找到使$g(\mathbf{s},\mathbf{t})$达到最大的译文
  - 若$\mathbf{s}$有m个词，每个词有n个翻译候选 - 共有n^m种组合



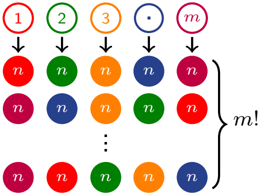
实现一个简单的机器翻译系统：解码(2)

- **解码的核心问题**是在所有可能的翻译结果中找到使$g(\mathbf{s},\mathbf{t})$达到最大的译文
  - 若$\mathbf{s}$有m个词，每个词有n个翻译候选 - 共有n^m种组合
  - 词的翻译候选可以任意调序



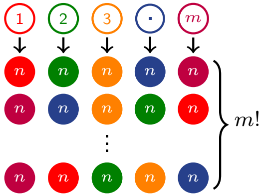
实现一个简单的机器翻译系统：解码(2)

- **解码的核心问题**是在所有可能的翻译结果中找到使$g(\mathbf{s},\mathbf{t})$达到最大的译文
  - 若$\mathbf{s}$有m个词，每个词有n个翻译候选 - 共有n^m种组合
  - 词的翻译候选可以任意调序
  - $\mathbf{s}$对应可能的译文至少有$n^m \cdot m!$



实现一个简单的机器翻译系统：解码(2)

- **解码的核心问题**是在所有可能的翻译结果中找到使$g(\mathbf{s},\mathbf{t})$达到最大的译文
  - 若$\mathbf{s}$有m个词，每个词有n个翻译候选 - 共有n^m种组合
  - 词的翻译候选可以任意调序
  - $\mathbf{s}$对应可能的译文至少有$n^m \cdot m!$



- 潜在翻译结果数量巨大，无法穷尽，因此我们需要高效的搜索算法找到理想的解

m	n	$n^m \cdot m!$
1	1	1
1	10	10
2	10	200
10	10	36288000000000000
20	10	2.43290200817664 $\times 10^{38}$
20	30	8.48300477127188 $\times 10^{47}$

实现一个简单的机器翻译系统：解码(3)

我们这里使用一种贪婪的解码算法：每轮选择一个源语言单词，利用它的翻译候选扩展译文，保留"最好"的结果，反复进行直到所有源语言单词都被翻译

输入：源语句子$\mathbf{s} = s_1...s_m$

输出：找的最佳译文

Function $\textsc{WordDecoding}(\mathbf{s})$

$\pi = \textsc{GetTransOptions}(\mathbf{s})$

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j] =$ **false then**

      $h = h \cup \textsc{Join}(best, \pi[j])$

  $best = \textsc{PruneForTop1}(h)$

  $used[best.j] =$ **true**

return $best.translatoin$

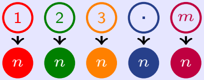
实现一个简单的机器翻译系统：解码(3)

我们这里使用一种贪婪的解码算法：每轮选择一个源语言单词，利用它的翻译候选扩展译文，保留"最好"的结果，反复进行直到所有源语言单词都被翻译

输入：源语句子$\mathbf{s} = s_1 ... s_m$
输出：找的最佳译文
Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

获取每个单词的翻译候选

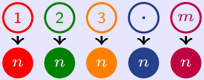


实现一个简单的机器翻译系统：解码(3)

我们这里使用一种贪婪的解码算法：每轮选择一个源语言单词，利用它的翻译候选扩展译文，保留"最好"的结果，反复进行直到所有源语言单词都被翻译

```
输入：源语句子s = s_1...s_m
输出：找的最佳译文
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

获取每个单词的翻译候选



*best*用于保存当前最好的翻译结果

*h*用于保存每步生成的所有译文候选

实现一个简单的机器翻译系统：解码(3)

我们这里使用一种贪婪的解码算法：每轮选择一个源语言单词，利用它的翻译候选扩展译文，保留"最好"的结果，反复进行直到所有源语言单词都被翻译

输入：源语句子$\mathbf{s} = s_1 ... s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

$\pi =$ GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j] =$ **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best =$ PruneForTop1(h)

  $used[best.j] =$ **true**

return $best.translatoin$

获取每个单词的翻译候选

$1 \rightarrow n$, $2 \rightarrow n$, $3 \rightarrow n$, $\cdot \rightarrow n$, $m \rightarrow n$

$best$用于保存当前最好的翻译结果

h用于保存每步生成的所有译文候选

Join(a, b) 返回 a 和b 的所有组合

$\begin{bmatrix} \text{a1} \\ \text{a2} \end{bmatrix} \times \begin{bmatrix} \text{b1} \\ \text{b2} \end{bmatrix} = \begin{bmatrix} \text{a1b1 a1b2} \\ \text{a2b1 a2b2} \end{bmatrix}$

实现一个简单的机器翻译系统：解码(3)

我们这里使用一种贪婪的解码算法：每轮选择一个源语言单词，利用它的翻译候选扩展译文，保留"最好"的结果，反复进行直到所有源语言单词都被翻译

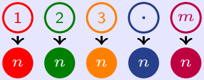
输入：源语句子 $\mathbf{s} = s_1 ... s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

获取每个单词的翻译候选



*best*用于保存当前最好的翻译结果

*h*用于保存每步生成的所有译文候选

Join(a, b) 返回 a 和b 的所有组合 $\boxed{\begin{matrix} a1 \\ a2 \end{matrix}} \times \boxed{\begin{matrix} b1 \\ b2 \end{matrix}} = \boxed{\begin{matrix} a1b1 & a1b2 \\ a2b1 & a2b2 \end{matrix}}$

PruneForTop1 保留得分最高的结果

$\boxed{0.234}$ ← top1

$\boxed{0.197}$

$\boxed{0.083}$

实现一个简单的机器翻译系统：解码(3)

我们这里使用一种贪婪的解码算法：每轮选择一个源语言单词，利用它的翻译候选扩展译文，保留"最好"的结果，反复进行直到所有源语言单词都被翻译

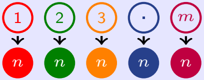
输入：源语句子$\mathbf{s} = s_1...s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

获取每个单词的翻译候选



$best$用于保存当前最好的翻译结果

h用于保存每步生成的所有译文候选

Join(a, b) 返回 a 和b 的所有组合

$\boxed{\begin{matrix}\text{a1}\\\text{a2}\end{matrix}} \times \boxed{\begin{matrix}\text{b1}\\\text{b2}\end{matrix}} = \boxed{\begin{matrix}\text{a1b1} & \text{a1b2}\\\text{a2b1} & \text{a2b2}\end{matrix}}$

PruneForTop1 保留得分最高的结果

0.234 ← top1

0.197

0.083

记录已经翻译过的源语单词

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

Function $\textsc{WordDecoding}(\mathbf{s})$

$\pi = \textsc{GetTransOptions}(\mathbf{s})$

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j] =$ **false then**

      $h = h \cup \textsc{Join}(best, \pi[j])$

  $best = \textsc{PruneForTop1}(h)$

  $used[best.j] =$ **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

我　对　你　表示　满意

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π =GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

我 $\pi(1)$		对 $\pi(2)$		你 $\pi(3)$		表示 $\pi(4)$		满意 $\pi(5)$	
I'm	.4	to	.3	you	.7	ϕ	.4	satisfy	.3
I	.3	with	.3	your	.3	show	.2	satisfied	.2
me	.1	for	.2			shows	.1	satisfies	.2
am	.1	to	.1			means	.1	it	.1
...	⋮	...	⋮			...	⋮	...	⋮

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

我 $\pi(1)$		对 $\pi(2)$		你 $\pi(3)$		表示 $\pi(4)$		满意 $\pi(5)$	
I'm	.4	to	.3	you	.7	ϕ	.4	satisfy	.3
I	.3	with	.3	your	.3	show	.2	satisfied	.2
me	.1	for	.2			shows	.1	satisfies	.2
am	.1	to	.1			means	.1	it	.1
...	⋮	...	⋮			...	⋮	...	⋮

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

```
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

只保留前n个结果

我 $\pi(1)$		对 $\pi(2)$		你 $\pi(3)$		表示 $\pi(4)$		满意 $\pi(5)$	
I'm	.4	to	.3	you	.7	ϕ	.4	satisfy	.3
I	.3	with	.3	your	.3	show	.2	satisfied	.2
me	.1	for	.2			shows	.1	satisfies	.2

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

我	对	你	表示	满意
$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation = \phi$

$best.j = -1$

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WORDDECODING($\mathbf{s}$)

```
π = GETTRANSOPTIONS(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ JOIN(best, π[j])
    best = PRUNEFORTOP1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：　$O(m^2 \cdot n)$

空间复杂度：　$O(m \cdot n)$

输入：待翻译句子(已经分词)

我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	表示 $\pi(4)$	满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$$best.translation = \quad \phi$$

$$best.j = -1$$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π = GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

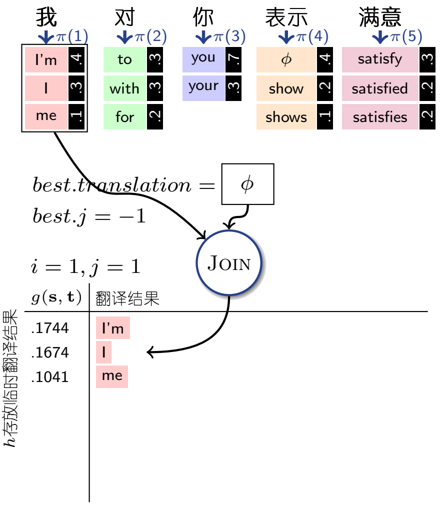
  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

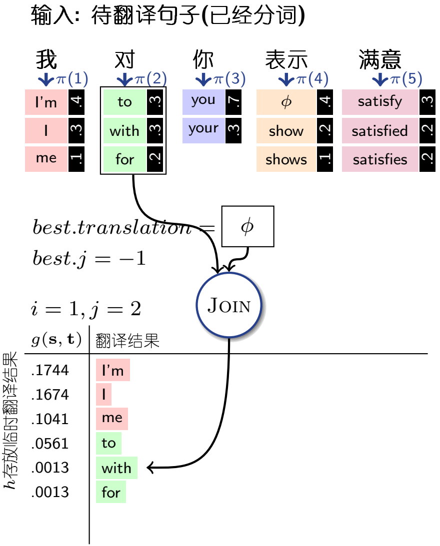
输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

```
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

```
输入: 源语句子s = s_1...s_m
输出: 找的最佳译文
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

我 $\pi(1)$		对 $\pi(2)$		你 $\pi(3)$		表示 $\pi(4)$		满意 $\pi(5)$	
I'm	.4	to	.3	you	.7	ϕ	.4	satisfy	.3
I	.3	with	.3	your	.3	show	.2	satisfied	.2
me	.1	for	.2			shows	.1	satisfies	.2

$best.translation = \phi$

$best.j = -1$

$i = 1, j = 5$

h存放临时翻译结果

$g(\mathbf{s},\mathbf{t})$	翻译结果
.1744	I'm ← top1
.1674	I
.1041	me
.0561	to
.0013	with
.0013	for
...	...
.1452	satisfies

实现一个简单的机器翻译系统：解码(4)

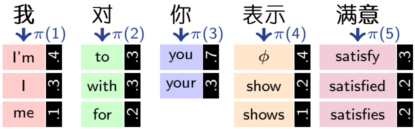
来看一个解码过程的运行实例

```
输入: 源语句子s = s_1...s_m
输出: 找的最佳译文
Function WordDecoding(s)
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)



$best.translation =$ I'm

$best.j = 1$

$i = 1, j = 5$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.1744	I'm ← top1
.1674	I
.1041	me
.0561	to
.0013	with
.0013	for
...	...
.1452	satisfies

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

```
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$		对 $\pi(2)$		你 $\pi(3)$		表示 $\pi(4)$		满意 $\pi(5)$	
I'm	.4	to	.3	you	.7	ϕ	.4	satisfy	.3
I	.3	with	.3	your	.3	show	.2	satisfied	.2
me	.1	for	.2			shows	.1	satisfies	.2

$best.translation =$ I'm

$best.j = 1$

$i = 1, j = 5$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.1744	I'm ← top1
.1674	I
.1041	me
.0561	to
.0013	with
.0013	for
...	...
.1452	satisfies

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

```
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used

我	对	你	表示	满意
$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm

$best.j = 1$

$i = 1, j = 5$

$g(\mathbf{s}, \mathbf{t})$	翻译结果

h存放临时翻译结果

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π = GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

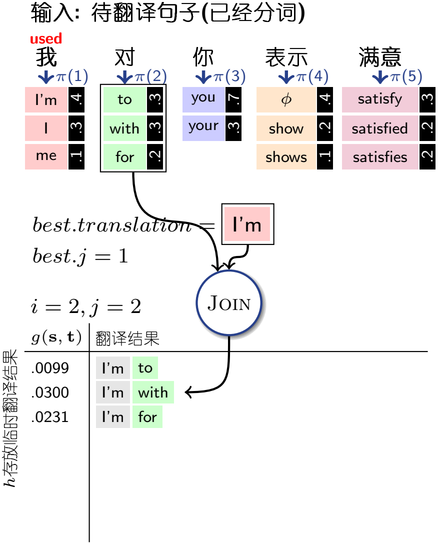
  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π = GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

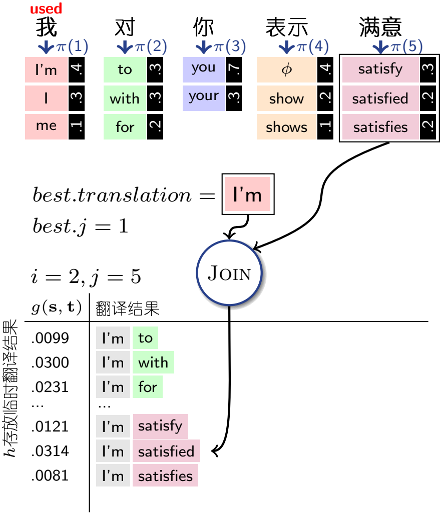
      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

输入：待翻译句子(已经分词)



时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	表示 $\pi(4)$	满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm

$best.j = 1$

$i = 2, j = 5$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0099	I'm to
.0300	I'm with
.0231	I'm for
...	...
.0121	I'm satisfy
.0314	I'm satisfied ← top1
.0081	I'm satisfies

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

我 (used)	对	你	表示	满意
$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied

$best.j = 5$

$i = 2, j = 5$

h存放临时翻译结果

$g(\mathbf{s},\mathbf{t})$	翻译结果
.0099	I'm to
.0300	I'm with
.0231	I'm for
...	...
.0121	I'm satisfy
.0314	I'm satisfied ← top1
.0081	I'm satisfies

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied

$best.j = 5$

$i = 2, j = 5$

h存放临时翻译结果

$g(\mathbf{s},\mathbf{t})$	翻译结果
.0099	I'm to
.0300	I'm with
.0231	I'm for
...	...
.0121	I'm satisfy
.0314	I'm satisfied ← top1
.0081	I'm satisfies

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

```
输入：源语句子 s = s_1 ... s_m
输出：找的最佳译文
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 ↓ $\pi(1)$	对 ↓ $\pi(2)$	你 ↓ $\pi(3)$	表示 ↓ $\pi(4)$	used 满意 ↓ $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied

$best.j = 5$

$i = 2, j = 5$

$g(\mathbf{s}, \mathbf{t})$	翻译结果

h存放临时翻译结果

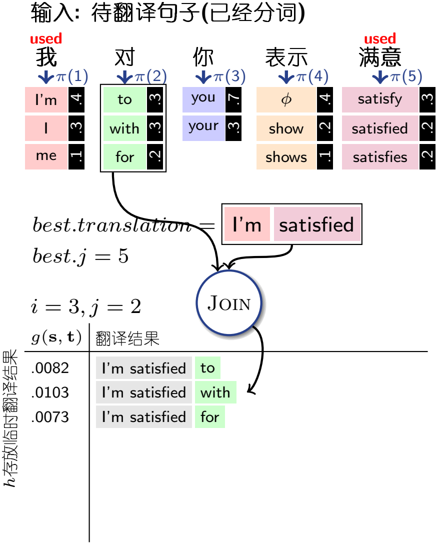
实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

```
Function WORDDECODING(s)
  π = GETTRANSOPTIONS(s)
  best = ϕ
  for i in [1, m] do
      h = ϕ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ JOIN(best, π[j])
      best = PRUNEFORTOP1(h)
      used[best.j] = true
  return best.translatoin
```



时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π =GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

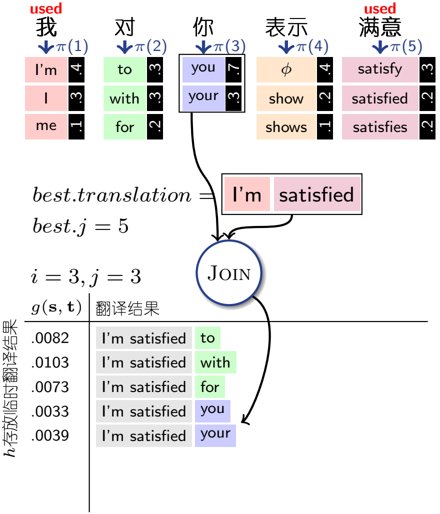
  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

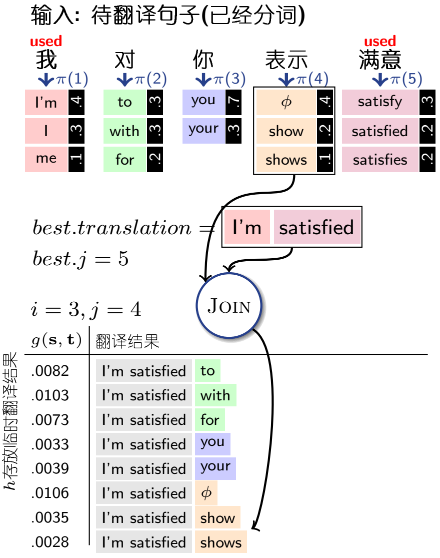
输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

```
Function WORDDECODING(s)
  π = GETTRANSOPTIONS(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ JOIN(best, π[j])
      best = PRUNEFORTOP1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π =GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation$ = I'm satisfied

$best.j = 5$

$i = 3, j = 4$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0082	I'm satisfied to
.0103	I'm satisfied with
.0073	I'm satisfied for
.0033	I'm satisfied you
.0039	I'm satisfied your
.0106	I'm satisfied ϕ ← top1
.0035	I'm satisfied show
.0028	I'm satisfied shows

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π =GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false** **then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied

$best.j = 4$

$i = 3, j = 4$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0082	I'm satisfied to
.0103	I'm satisfied with
.0073	I'm satisfied for
.0033	I'm satisfied you
.0039	I'm satisfied your
.0106	I'm satisfied ϕ $\leftarrow$ top1
.0035	I'm satisfied show
.0028	I'm satisfied shows

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	used 表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied

$best.j = 4$

$i = 3, j = 4$

h存放临时翻译结果

$g(\mathbf{s},\mathbf{t})$	翻译结果
.0082	I'm satisfied to
.0103	I'm satisfied with
.0073	I'm satisfied for
.0033	I'm satisfied you
.0039	I'm satisfied your
.0106	I'm satisfied ϕ $\leftarrow$ top1
.0035	I'm satisfied show
.0028	I'm satisfied shows

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π =GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$		对 $\pi(2)$		你 $\pi(3)$		used 表示 $\pi(4)$		used 满意 $\pi(5)$	
I'm	.4	to	.3	you	.7	ϕ	.4	satisfy	.3
I	.3	with	.3	your	.3	show	.2	satisfied	.2
me	.1	for	.2			shows	.1	satisfies	.2

$best.translation =$ I'm satisfied

$best.j = 4$

$i = 3, j = 4$

$g(\mathbf{s},\mathbf{t})$	翻译结果

h存放临时翻译结果

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

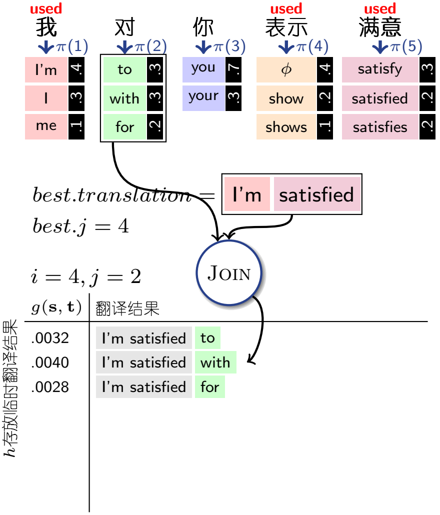
Function WORDDECODING($\mathbf{s}$)

```
π = GETTRANSOPTIONS(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ JOIN(best, π[j])
    best = PRUNEFORTOP1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入: 源语句子$\mathbf{s} = s_1 \dots s_m$

输出: 找的最佳译文

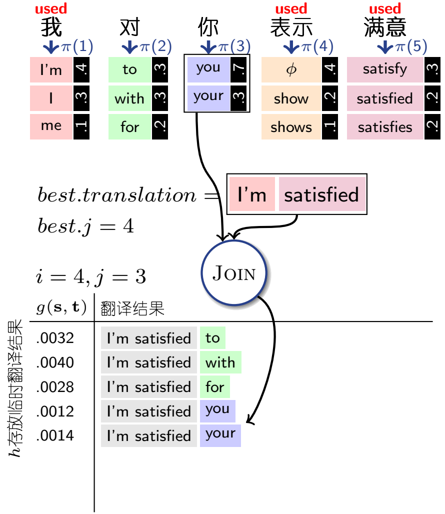
Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度: $O(m^2 \cdot n)$

空间复杂度: $O(m \cdot n)$

输入：待翻译句子(已经分词)



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

```
π = GetTransOptions(s)
best = φ
for i in [1, m] do
    h = φ
    foreach j in [1, m] do
        if used[j] = false then
            h = h ∪ Join(best, π[j])
    best = PruneForTop1(h)
    used[best.j] = true
return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	used 表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied

$best.j = 4$

$i = 4, j = 3$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0032	I'm satisfied to
.0040	I'm satisfied with ← top1
.0028	I'm satisfied for
.0012	I'm satisfied you
.0014	I'm satisfied your

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

```
输入: 源语句子s = s1...sm
输出: 找的最佳译文
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度: $O(m^2 \cdot n)$

空间复杂度: $O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	对 $\pi(2)$	你 $\pi(3)$	used 表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied with

$best.j = 2$

$i = 4, j = 3$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0032	I'm satisfied to
.0040	I'm satisfied with ← top1
.0028	I'm satisfied for
.0012	I'm satisfied you
.0014	I'm satisfied your

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π = GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$

输入：待翻译句子(已经分词)

used	used		used	used
我	对	你	表示	满意
$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation$ = I'm satisfied with

$best.j = 2$

$i = 4, j = 3$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0032	I'm satisfied to
.0040	I'm satisfied with ← top1
.0028	I'm satisfied for
.0012	I'm satisfied you
.0014	I'm satisfied your

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子 $\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

```
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

used	used		used	used
我	对	你	表示	满意
$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied with

$best.j = 2$

$i = 4, j = 3$

h存放临时翻译结果

$g(\mathbf{s},\mathbf{t})$	翻译结果

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

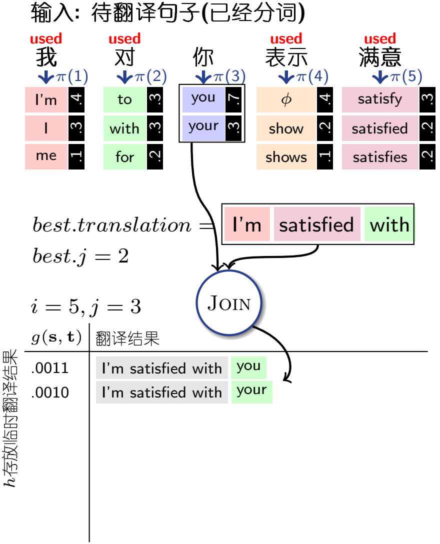
输入：源语句子$\mathbf{s} = s_1 \ldots s_m$

输出：找的最佳译文

```
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度：$O(m^2 \cdot n)$

空间复杂度：$O(m \cdot n)$



实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

```
输入: 源语句子s = s_1...s_m
输出: 找的最佳译文
Function WordDecoding(s)
  π = GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度: $O(m^2 \cdot n)$

空间复杂度: $O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	used 对 $\pi(2)$	你 $\pi(3)$	used 表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied with

$best.j = 2$

$i = 5, j = 3$

h存放临时翻译结果

$g(\mathbf{s},\mathbf{t})$	翻译结果
.0011	I'm satisfied with you ← top1
.0010	I'm satisfied with your

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

```
输入: 源语句子s = s_1...s_m
输出: 找的最佳译文
Function WordDecoding(s)
  π =GetTransOptions(s)
  best = φ
  for i in [1, m] do
      h = φ
      foreach j in [1, m] do
          if used[j] = false then
              h = h ∪ Join(best, π[j])
      best = PruneForTop1(h)
      used[best.j] = true
  return best.translatoin
```

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

used	used		used	used
我	对	你	表示	满意
↓$\pi(1)$	↓$\pi(2)$	↓$\pi(3)$	↓$\pi(4)$	↓$\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied with you

$best.j = 3$

$i = 5, j = 3$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0011	I'm satisfied with you ← top1
.0010	I'm satisfied with your

实现一个简单的机器翻译系统：解码(4)

来看一个解码过程的运行实例

输入：源语句子$\mathbf{s} = s_1 \dots s_m$

输出：找的最佳译文

Function WordDecoding($\mathbf{s}$)

π =GetTransOptions($\mathbf{s}$)

$best = \phi$

for i in $[1, m]$ **do**

  $h = \phi$

  foreach j in $[1, m]$ **do**

    if $used[j]$ = **false then**

      $h = h \cup$ Join($best, \pi[j]$)

  $best$ = PruneForTop1(h)

  $used[best.j]$ = **true**

return $best.translatoin$

时间复杂度： $O(m^2 \cdot n)$

空间复杂度： $O(m \cdot n)$

输入：待翻译句子(已经分词)

used 我 $\pi(1)$	used 对 $\pi(2)$	used 你 $\pi(3)$	used 表示 $\pi(4)$	used 满意 $\pi(5)$
I'm .4	to .3	you .7	ϕ .4	satisfy .3
I .3	with .3	your .3	show .2	satisfied .2
me .1	for .2		shows .1	satisfies .2

$best.translation =$ I'm satisfied with you

$best.j = 3$

$i = 5, j = 3$

h存放临时翻译结果

$g(\mathbf{s}, \mathbf{t})$	翻译结果
.0011	I'm satisfied with you ← top1
.0010	I'm satisfied with your

即将开始正式的内容

- 基于词的统计机器翻译很简单 - 建议实现一下前面所描述的方法，确实不难
  - 是的，思想很简单
  - 不对，前面的例子主要面向实现，还有很多问题没有回答
    - **建模**：还需要更严密的数学模型来描述翻译过程
    - **训练**：需要有一个科学的优化方法
    - **解码**：是否还有其它更系统的解码方法？A*，Bottom-up decoding?
    - **其它**：空翻译问题、调序建模问题等等

即将开始正式的内容

- 基于词的统计机器翻译很简单 - 建议实现一下前面所描述的方法，确实不难
  - 是的，思想很简单
  - 不对，前面的例子主要面向实现，还有很多问题没有回答
    - **建模**：还需要更严密的数学模型来描述翻译过程
    - **训练**：需要有一个科学的优化方法
    - **解码**：是否还有其它更系统的解码方法？A*，Bottom-up decoding?
    - **其它**：空翻译问题、调序建模问题等等
- 下面，继续对一些问题进行进一步介绍及分析

1. 统计机器翻译基础模型和IBM模型
   - 统计机器翻译模型基础(生成模型)
   - IBM Models 1-2的建模及训练
   - IBM Models 3-5的建模
2. 统计词对齐
   - 基于HMMs的词对齐
   - 基于判别式模型的词对齐

Outline

1. 一个简单的翻译实例

2. IBM模型
   - 建模
   - 解码
   - 模型训练

统计机器翻译

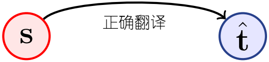
- 统计机器翻译源于1941年的Warren Weaver的思想，而正式开始于
  - **A Program for Aligning Sentences in Bilingual Corpora**
    William A. Gale; Kenneth W. Church. 1993.
  - **The Mathematics of Statistical Machine Translation: Parameter Estimation**
    Peter E. Brown; Stephen A. Della Pietra; Vincent J. Della Pietra; Robert L. Mercer. 1993.
- 特别是IBM Waston研究中心Brown等人的基于词的统计机器翻译模型，成为机器翻译领域至今经典中的经典

Brown等人的工作(IBM Waston) - on Wikipedia

The first ideas of statistical machine translation were introduced by Warren Weaver in 1949, including the ideas of applying Claude Shannon's information theory. Statistical machine translation was re-introduced in 1991 by researchers at IBM's Thomas J. Watson Research Center and has contributed to the significant resurgence in interest in machine translation in recent years. Nowadays it is by far the most widely-studied machine translation method

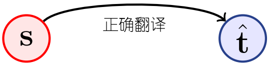
机器翻译的统计建模

- **一个人在做翻译时**：对于给定的源语言句子s，可以了翻译为一个（或者若干个）正确的译文$\hat{\mathbf{t}}$
  - 也就是说除了正确的译文，其它的翻译都是不正确的

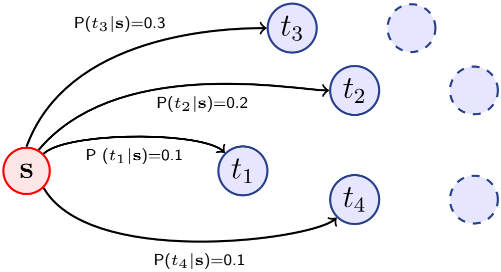


机器翻译的统计建模

- 一个人在做翻译时：对于给定的源语言句子s，可以了翻译为一个（或者若干个）正确的译文$\hat{t}$
  - 也就是说除了正确的译文，其它的翻译都是不正确的



- 统计机器翻译的思想是：对于s，所有可能的目标语词串t都是可能的译文。每一对(s,t)都有一个概率值P(t|s)来描述s 翻译为t的好与坏



噪声信道模型

- **噪声信道模型**：源语言句子$\mathbf{s}$(信宿)是由目标语句子$\mathbf{t}$(信源)经过一个有噪声的信道得到的。如果知道了$\mathbf{s}$和信道的性质，我们可以通过$\mathrm{P}(\mathbf{t}|\mathbf{s})$得到可能的信源的概率。

而通过上述过程找到最可能的信源的过程被称之为解码

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

噪声信道模型

- **噪声信道模型**：源语言句子$\mathbf{s}$(信宿)是由目标语句子$\mathbf{t}$(信源)经过一个有噪声的信道得到的。如果知道了$\mathbf{s}$和信道的性质，我们可以通过$\mathrm{P}(\mathbf{t}|\mathbf{s})$得到可能的信源的概率。

而通过上述过程找到最可能的信源的过程被称之为解码

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

- 贝叶斯变换

$$\begin{aligned} \mathrm{P}(\mathbf{t}|\mathbf{s}) &= \frac{\mathrm{P}(\mathbf{s},\mathbf{t})}{\mathrm{P}(\mathbf{s})} \\ &= \frac{\mathrm{P}(\mathbf{s}|\mathbf{t})\mathrm{P}(\mathbf{t})}{\mathrm{P}(\mathbf{s})} \end{aligned}$$

噪声信道模型(2)

$$\mathrm{P}(\mathbf{t}|\mathbf{s}) = \frac{\mathrm{P}(\mathbf{s}|\mathbf{t})\ \mathrm{P}(\mathbf{t})}{\mathrm{P}(\mathbf{s})}$$

噪声信道模型(2)

给定信源t，得到信宿s的概率，称作**翻译模型**

信源t出现的概率称作**语言模型**

$$\mathrm{P}(\mathbf{t}|\mathbf{s}) = \frac{\mathrm{P}(\mathbf{s}|\mathbf{t})\,\mathrm{P}(\mathbf{t})}{\mathrm{P}(\mathbf{s})}$$

信宿s出现的概率

给定s，P(s)为**常量**

噪声信道模型(2)

给定信源$\mathbf{t}$，得到信宿$\mathbf{s}$的概率，称作**翻译模型**

信源$\mathbf{t}$出现的概率称作**语言模型**

$$\mathrm{P}(\mathbf{t}|\mathbf{s}) = \frac{\mathrm{P}(\mathbf{s}|\mathbf{t})\ \mathrm{P}(\mathbf{t})}{\mathrm{P}(\mathbf{s})}$$

信宿$\mathbf{s}$出现的概率

给定$\mathbf{s}$，$\mathrm{P}(\mathbf{s})$为**常量**

- **因此**，翻译问题可以被重新描述为

$$\begin{aligned}\hat{\mathbf{t}} &= \arg\max_{\mathbf{t}} \frac{\mathrm{P}(\mathbf{s}|\mathbf{t})\mathrm{P}(\mathbf{t})}{\mathrm{P}(\mathbf{s})} \\ &= \arg\max_{\mathbf{t}} \mathrm{P}(\mathbf{s}|\mathbf{t})\mathrm{P}(\mathbf{t})\end{aligned}$$

即，在所有可能的译文中找到使翻译模型$\mathrm{P}(\mathbf{s}|\mathbf{t})$和语言模型$\mathrm{P}(\mathbf{t})$乘积最大的译文

基本问题

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

- 三个基本问题
  1. 建模：如何描述计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$和$\mathrm{P}(\mathbf{t})$的计算方式
  2. 训练：如何获得计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$和$\mathrm{P}(\mathbf{t})$所需的参数
  3. 解码：如何完成搜索最优解的过程$argmax$

基本问题

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

- 三个基本问题
  1. **建模**：如何描述计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$和$\mathrm{P}(\mathbf{t})$的计算方式
  2. **训练**：如何获得计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$和$\mathrm{P}(\mathbf{t})$所需的参数
  3. **解码**：如何完成搜索最优解的过程$argmax$
- 回忆一下本章开始的实例，是不是有似曾相识的感觉？

$$g(\mathbf{s},\mathbf{t}) = \underbrace{\prod_{(j,i)\in\hat{A}} \mathrm{P}(s_j,t_i)}_{\substack{\mathrm{P}(\mathbf{s}|\mathbf{t}) \\ \text{翻译模型}}} \times \underbrace{\mathrm{P}_{lm}(\mathbf{t})}_{\substack{\mathrm{P}(\mathbf{t}) \\ \text{语言模型}}}$$

- ▶ 词汇翻译概率和ngram概率使用相对频率估计(第7页)
- ▶ $argmax$使用第16页所描述的解码算法

IBM模型中的假设 - 词对齐

- $\mathrm{P}(\mathbf{t})$**和解码**在前面的内容中有介绍，下面重点求解$\mathrm{P}(\mathbf{s}|\mathbf{t})$，即：
  - **翻译模型建模** - $\mathrm{P}(\mathbf{s}|\mathbf{t})$的计算方法
  - **翻译模型参数估计** - 计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$所需的参数

IBM模型中的假设 - 词对齐

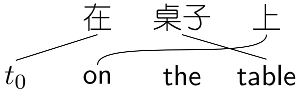
- P(t)和解码在前面的内容中有介绍，下面重点求解P(s|t)，即：
  - 翻译模型建模 - P(s|t)的计算方法
  - 翻译模型参数估计 - 计算P(s|t)所需的参数
- IBM模型的假设：$\mathbf{s} = s_1 ... s_m$和$\mathbf{t} = t_1 ... t_l$之间有单词一级的对应，称作单词对齐或者词对齐。此外：
  - 约束：一个源语言单词只能对应一个目标语单词

IBM模型中的假设 - 词对齐

- **P(t)和解码**在前面的内容中有介绍，下面重点求解P(s|t)，即：
  - **翻译模型建模** - P(s|t)的计算方法
  - **翻译模型参数估计** - 计算P(s|t)所需的参数

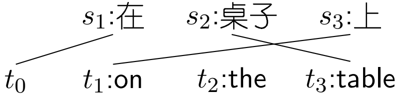
- **IBM模型的假设**：$\mathbf{s} = s_1...s_m$和$\mathbf{t} = t_1...t_l$之间有单词一级的对应，称作单词对齐或者词对齐。此外：
  - **约束**：一个源语言单词只能对应一个目标语单词

  - **空翻译(删词)**：源语言单词可以翻空，这时它对应到一个虚拟的目标语单词t_0



建模 - P(s|t)

- 给定**s**和**t**，它们之间的词对齐被记为$\mathbf{a} = a_1...a_m$
  - a_j表示第j个源语单词s_j对应的目标语单词的位置



$$a_1 = 0, a_2 = 3, a_3 = 1$$

建模 - $\mathrm{P}(\mathbf{s}|\mathbf{t})$

- 给定**s**和**t**，它们之间的词对齐被记为$\mathbf{a} = a_1...a_m$
  - a_j表示第j个源语单词s_j对应的目标语单词的位置

s_1:在 s_2:桌子 s_3:上

t_0 t_1:on t_2:the t_3:table

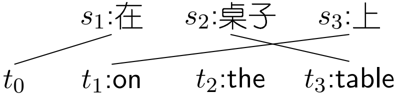
$$a_1 = 0, a_2 = 3, a_3 = 1$$

- 建模!!: $\mathrm{P}(\mathbf{s}|\mathbf{t})$被表示为所有可能的词对齐的生成概率

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \mathrm{P}(\mathbf{s}, \mathbf{a}|\mathbf{t})$$

建模 - $\mathrm{P}(\mathbf{s}|\mathbf{t})$

- 给定s和t，它们之间的词对齐被记为$\mathbf{a} = a_1...a_m$
  - a_j表示第j个源语单词s_j对应的目标语单词的位置



$$a_1 = 0, a_2 = 3, a_3 = 1$$

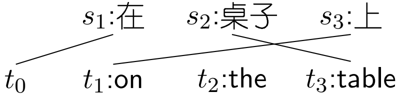
- 建模!!: $\mathrm{P}(\mathbf{s}|\mathbf{t})$被表示为所有可能的词对齐的生成概率

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \mathrm{P}(\mathbf{s}, \mathbf{a}|\mathbf{t})$$

每一种$\mathbf{a}$对应一个$\mathrm{P}(\mathbf{s}, \mathbf{a}|\mathbf{t})$

建模 - P(s|t)

- 给定s和t，它们之间的词对齐被记为$\mathbf{a} = a_1 ... a_m$
  - a_j表示第j个源语单词s_j对应的目标语单词的位置



$a_1 = 0, a_2 = 3, a_3 = 1$

- 建模!!: P(s|t)被表示为所有可能的词对齐的生成概率

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \mathrm{P}(\mathbf{s}, \mathbf{a}|\mathbf{t})$$

每一种a对应一个$\mathrm{P}(\mathbf{s}, \mathbf{a}|\mathbf{t})$

建模 - $\mathbf{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

- **进一步建模!!**：对于源语句子$\mathbf{s}=s_1...s_m$(m个词)、目标语译文$\mathbf{t}=t_0...t_n$(n个词)和词对齐$\mathbf{a}=a_1...a_m$，按如下方式计算$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$
  - 符号定义：$s_x^y=s_x...s_y$，$a_x^y=a_x...a_y$

$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) = \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\,\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})$$

建模 - $\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

- **进一步建模!!**：对于源语句子$\mathbf{s}=s_1...s_m$(m个词)、目标语译文$\mathbf{t}=t_0...t_n$(n个词)和词对齐$\mathbf{a}=a_1...a_m$，按如下方式计算$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$
  - 符号定义：$s_x^y=s_x...s_y$，$a_x^y=a_x...a_y$

$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})=\underbrace{\mathrm{P}(m|\mathbf{t})}_{①}\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\,\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})$$

  - 生成模型：给定译文$\mathbf{t}$生成源文$\mathbf{s}$和对齐$\mathbf{a}$
    1. 根据译文$\mathbf{t}$选择源文的长度m

建模 - $\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

- **进一步建模!!**：对于源语句子$\mathbf{s}=s_1...s_m$(m个词)、目标语译文$\mathbf{t}=t_0...t_n$(n个词)和词对齐$\mathbf{a}=a_1...a_m$，按如下方式计算$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$
  - 符号定义：$s_x^y=s_x...s_y$, $a_x^y=a_x...a_y$

$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})=\underbrace{\mathrm{P}(m|\mathbf{t})}_{①}\underbrace{\prod_{j=1}^{m}}_{②}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\,\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})$$

  - 生成模型：给定译文$\mathbf{t}$生成源文$\mathbf{s}$和对齐$\mathbf{a}$
    1. 根据译文$\mathbf{t}$选择源文的长度m
    2. 循环源文的每个位置j

建模 - $\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

- 进一步建模!!：对于源语句子$\mathbf{s}=s_1...s_m$(m个词)、目标语译文$\mathbf{t}=t_0...t_n$(n个词)和词对齐$\mathbf{a}=a_1...a_m$，按如下方式计算$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$
  - 符号定义：$s_x^y=s_x...s_y$，$a_x^y=a_x...a_y$

$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})=\underbrace{\mathrm{P}(m|\mathbf{t})}_{①}\underbrace{\prod_{j=1}^{m}}_{②}\underbrace{\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})}_{③}\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})$$

  - 生成模型：给定译文$\mathbf{t}$生成源文$\mathbf{s}$和对齐$\mathbf{a}$
    1. 根据译文$\mathbf{t}$选择源文的长度m
    2. 循环源文的每个位置j
    3. 根据译文$\mathbf{t}$、源文长度m、已经生成的源语单词s_1^{j-1}和对齐a_1^{j-1}，生成第j个位置的对齐结果a_j

建模 - $\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

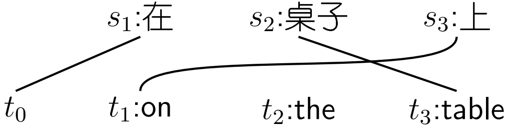
- **进一步建模!!**：对于源语句子$\mathbf{s}=s_1...s_m$(m个词)、目标语译文$\mathbf{t}=t_0...t_n$(n个词)和词对齐$\mathbf{a}=a_1...a_m$，按如下方式计算$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$
  - 符号定义：$s_x^y=s_x...s_y$，$a_x^y=a_x...a_y$

$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) = \underbrace{\mathrm{P}(m|\mathbf{t})}_{①} \underbrace{\prod_{j=1}^{m}}_{②} \underbrace{\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})}_{③} \underbrace{\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})}_{④}$$

  - **生成模型**：给定译文$\mathbf{t}$生成源文$\mathbf{s}$和对齐$\mathbf{a}$
    1. 根据译文$\mathbf{t}$选择源文的长度m
    2. 循环源文的每个位置j
    3. 根据译文$\mathbf{t}$、源文长度m、已经生成的源语单词s_1^{j-1}和对齐a_1^{j-1}，生成第j个位置的对齐结果a_j
    4. 根据译文$\mathbf{t}$、源文长度m、已经生成的源语单词s_1^{j-1}和对齐a_1^{j}，生成第j个位置的源语言单词s_j（注意：这时a_j已经生成了）

实例 - $\mathbf{P(s, a|t)}$

$\mathbf{s}$ = 在 桌子 上 $\quad$ $\mathbf{t} = t_0$ on the table $\quad$ $\mathbf{a} = \{$1-0,2-3,3-1$\}$



$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) \quad = \quad \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})$$

实例 - $\mathbf{P(s, a|t)}$

$\mathbf{s}$ = 在 桌子 上 $\mathbf{t}$ = t_0 on the table $\mathbf{a}$ = {1-0,2-3,3-1}

t_0 t_1:on t_2:the t_3:table

$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \mathrm{P}(m=3 \mid \text{'}t_0 \text{ on the table'})
\end{aligned}
$$

实例 - $\mathbf{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

$\mathbf{s}=$ 在 桌子 上 $\quad \mathbf{t}=t_0$ on the table $\quad \mathbf{a}=\{$1-0,2-3,3-1$\}$

● ● ●

t_0 $\quad t_1$:on $\quad t_2$:the $\quad t_3$:table

$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \mathrm{P}(m=3 \mid \text{'}t_0 \text{ on the table'}) \times \\
&\quad\ \mathrm{P}(a_1=0 \mid \phi,\phi,3,\text{'}t_0 \text{ on the table'})
\end{aligned}
$$

实例 - $\mathbf{P(s, a|t)}$

$\mathbf{s}$ = 在 桌子 上　　$\mathbf{t} = t_0$ on the table　　$\mathbf{a} = \{$1-0,2-3,3-1$\}$

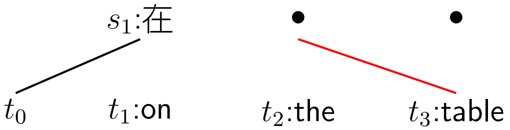
s_1:在　•　•

t_0　t_1:on　t_2:the　t_3:table

$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \mathrm{P}(m=3 \mid \text{'}t_0 \text{ on the table'}) \times \\
&\quad \mathrm{P}(a_1=0 \mid \phi,\phi,3,\text{'}t_0 \text{ on the table'}) \times \\
&\quad \mathrm{P}(s_1=\text{在} \mid \{1\text{-}0\},\phi,3,\text{'}t_0 \text{ on the table'})
\end{aligned}
$$

实例 - $\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

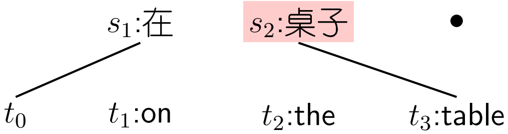
$\mathbf{s}=$ 在 桌子 上 $\mathbf{t}=t_0$ on the table $\mathbf{a}=\{$1-0,2-3,3-1$\}$



$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \mathrm{P}(m=3 \mid \text{'}t_0\text{ on the table'}) \times \\
&\quad \mathrm{P}(a_1=0 \mid \phi,\phi,3,\text{'}t_0\text{ on the table'}) \times \\
&\quad \mathrm{P}(s_1=\text{在} \mid \{1\text{-}0\},\phi,3,\text{'}t_0\text{ on the table'}) \times \\
&\quad \mathrm{P}(a_2=3 \mid \{1\text{-}0\},\text{'在'},3,\text{'}t_0\text{ on the table'})
\end{aligned}
$$

实例 - $\mathbf{P(s,a|t)}$

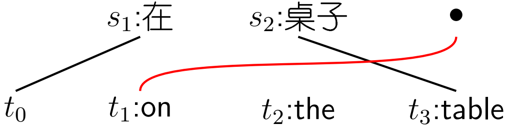
$\mathbf{s}$ = 在 桌子 上 $\quad$ $\mathbf{t} = t_0$ on the table $\quad$ $\mathbf{a} = \{$1-0,2-3,3-1$\}$



$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \mathrm{P}(m=3 \mid \text{'}t_0 \text{ on the table'}) \times \\
&\quad \mathrm{P}(a_1=0 \mid \phi,\phi,3,\text{'}t_0 \text{ on the table'}) \times \\
&\quad \mathrm{P}(s_1=\text{在} \mid \{\text{1-0}\},\phi,3,\text{'}t_0 \text{ on the table'}) \times \\
&\quad \mathrm{P}(a_2=3 \mid \{\text{1-0}\},\text{'在'},3,\text{'}t_0 \text{ on the table'}) \times \\
&\quad \mathrm{P}(s_2=\text{桌子} \mid \{\text{1-0,2-3}\},\text{'在'},3,\text{'}t_0 \text{ on the table'})
\end{aligned}
$$

实例 - $\mathbf{P(s,a|t)}$

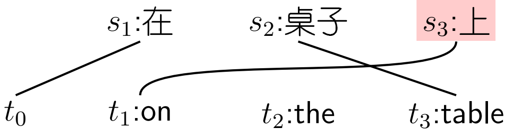
$\mathbf{s}$ = 在 桌子 上　　$\mathbf{t} = t_0$ on the table　　$\mathbf{a} = \{$1-0,2-3,3-1$\}$



$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) \quad &= \quad \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \quad \mathrm{P}(m=3 \mid \text{'}t_0\text{ on the table'}) \times \\
& \qquad \mathrm{P}(a_1=0 \mid \phi,\phi,3,\text{'}t_0\text{ on the table'}) \times \\
& \qquad \mathrm{P}(s_1=\text{在} \mid \{\text{1-0}\},\phi,3,\text{'}t_0\text{ on the table'}) \times \\
& \qquad \mathrm{P}(a_2=3 \mid \{\text{1-0}\},\text{'在'},3,\text{'}t_0\text{ on the table'}) \times \\
& \qquad \mathrm{P}(s_2=\text{桌子} \mid \{\text{1-0,2-3}\},\text{'在'},3,\text{'}t_0\text{ on the table'}) \times \\
& \qquad \mathrm{P}(a_3=1 \mid \{\text{1-0,2-3}\},\text{'在 桌子'},3,\text{'}t_0\text{ on the table'})
\end{aligned}
$$

实例 - $\mathbf{P(s, a|t)}$

$\mathbf{s}$ = 在 桌子 上 $\quad$ $\mathbf{t} = t_0$ on the table $\quad$ $\mathbf{a}$ = {1-0,2-3,3-1}



$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t})\prod_{j=1}^{m}\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t}) \\
&= \mathrm{P}(m=3\mid \text{'}t_0 \text{ on the table'})\times \\
&\quad \mathrm{P}(a_1=0\mid \phi,\phi,3,\text{'}t_0 \text{ on the table'})\times \\
&\quad \mathrm{P}(s_1=\text{在}\mid \{\text{1-0}\},\phi,3,\text{'}t_0 \text{ on the table'})\times \\
&\quad \mathrm{P}(a_2=3\mid \{\text{1-0}\},\text{'在'},3,\text{'}t_0 \text{ on the table'})\times \\
&\quad \mathrm{P}(s_2=\text{桌子}\mid \{\text{1-0,2-3}\},\text{'在'},3,\text{'}t_0 \text{ on the table'})\times \\
&\quad \mathrm{P}(a_3=1\mid \{\text{1-0,2-3}\},\text{'在 桌子'},3,\text{'}t_0 \text{ on the table'})\times \\
&\quad \mathrm{P}(s_3=\text{上}\mid \{\text{1-0,2-3,3-1}\},\text{'在 桌子'},3,\text{'}t_0 \text{ on the table'})
\end{aligned}
$$

问题来了

- **重点**：重新回顾一下这个模型 - 两个公式

$$
\begin{aligned}
\mathrm{P}(\mathbf{s}|\mathbf{t}) &= \sum_{\mathbf{a}} \mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) \\
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t}) \prod_{j=1}^{m} \mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t}) \mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t})
\end{aligned}
$$

- 两个严重问题
  1. 第一个公式：如何遍历所有的对齐$\mathbf{a}$
  2. 第二个公式：如何计算$\mathrm{P}(m|\mathbf{t})$、$\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t})$和$\mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t})$

问题来了

- 重点：重新回顾一下这个模型 - 两个公式

$$
\begin{aligned}
\mathrm{P}(\mathbf{s}|\mathbf{t}) &= \sum_{\mathbf{a}} \mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) \\
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \mathrm{P}(m|\mathbf{t}) \prod_{j=1}^{m} \mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t}) \mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})
\end{aligned}
$$

- 两个严重问题
  1. 第一个公式：如何遍历所有的对齐$\mathbf{a}$
  2. 第二个公式：如何计算$\mathrm{P}(m|\mathbf{t})$、$\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})$和$\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})$
- Brown等人(1993)的解决方法：对问题进行化简
  - IBM Models 1-2: 化简公式2的参数，精确高效求解公式1 →本教程内容
  - IBM Models 3-5: 简单化简公式2的参数，近似求解公式1 →自学内容

IBM模型1的假设

- **IBM模型1假设**

1. 源语长度概率为常数ϵ

$$\mathrm{P}(m|\mathbf{t}) \equiv \epsilon$$

2. 对齐概率$\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t})$仅依赖于译文长度$l+1$(均匀分布)

$$\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t}) \equiv \frac{1}{l+1}$$

3. 源语单词s_j生成概率$\mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t})$仅依赖与其对齐的译文单词$t_{a_j}$，即词汇翻译概率$f(s_j|t_{a_j})$ $(\sum_{s_j} f(s_j|t_{a_j}) = 1)$

$$\mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t}) \equiv f(s_j|t_{a_j})$$

IBM模型1的假设

- **IBM模型1假设**
  1. 源语长度概率为常数ϵ
  $$\mathrm{P}(m|\mathbf{t}) \equiv \epsilon$$
  2. 对齐概率$\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t})$仅依赖于译文长度$l+1$(均匀分布)
  $$\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t}) \equiv \frac{1}{l+1}$$
  3. 源语单词s_j生成概率$\mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t})$仅依赖与其对齐的译文单词$t_{a_j}$，即词汇翻译概率$f(s_j|t_{a_j})$ $(\sum_{s_j} f(s_j|t_{a_j}) = 1)$
  $$\mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t}) \equiv f(s_j|t_{a_j})$$
- **核心思想是把复杂参数化简为简单参数**
  - 比如：$\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t}) \equiv \frac{1}{l-1}$把参数空间$(a_1^{j}, s_1^{j-1}, m, \mathbf{t})$化简为$l$
  - 优点：模型大大化简；缺点：化简导致模型不准确

IBM模型1 - 完整模型

- 代入假设

$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \underbrace{\mathrm{P}(m|\mathbf{t})}_{\equiv\ \epsilon} \prod_{j=1}^{m} \underbrace{\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})}_{\equiv\ \frac{1}{l+1}}\ \underbrace{\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})}_{\equiv\ f(s_j|t_{a_j})} \\
&= \epsilon \prod_{j=1}^{m} \frac{1}{l+1} f(s_j|t_{a_j}) \\
&= \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})
\end{aligned}
$$

IBM模型1 - 完整模型

- 代入假设

$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \underset{\substack{|||\\ \epsilon}}{\mathrm{P}(m|\mathbf{t})} \prod_{j=1}^{m} \underset{\substack{|||\\ \frac{1}{l+1}}}{\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})}\, \underset{\substack{|||\\ f(s_j|t_{a_j})}}{\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})} \\
&= \epsilon \prod_{j=1}^{m} \tfrac{1}{l+1} f(s_j|t_{a_j}) \\
&= \tfrac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})
\end{aligned}
$$

- 将上式代入$\mathbf{P}(\mathbf{s}|\mathbf{t}) = \sum_{a} \mathbf{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

$$
\begin{aligned}
\mathrm{P}(\mathbf{s}|\mathbf{t}) &= \sum_{\mathbf{a}} \mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) \\
&= \sum_{\mathbf{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})
\end{aligned}
$$

IBM模型1 - 完整模型(2)

- $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$中需要对遍历所有的对齐，即$\sum_{\mathbf{a}}$。这个过程可以被重新表示为

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$$

IBM模型1 - 完整模型(2)

- $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$中需要对遍历所有的对齐，即$\sum_{\mathbf{a}}$。这个过程可以被重新表示为

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \underbrace{\boxed{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}}_{①} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$$

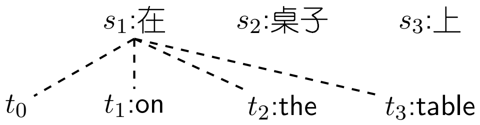
1. 遍历所有的的对齐$\mathbf{a}$。$\mathbf{a}$由$\{a_1,...,a_m\}$组成，每个$a_j \in \{a_1,...,a_m\}$从第译文开始位置(0)循环到截止位置(l)

IBM模型1 - 完整模型(2)

- $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$中需要对遍历所有的对齐，即$\sum_{\mathbf{a}}$。这个过程可以被重新表示为

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \underbrace{\boxed{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}}_{1} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$$

1. 遍历所有的的对齐$\mathbf{a}$。$\mathbf{a}$由$\{a_1,...,a_m\}$组成，每个$a_j \in \{a_1,...,a_m\}$从第译文开始位置(0)循环到截止位置(l)

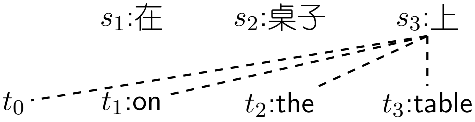


IBM模型1 - 完整模型(2)

- $\mathrm{P}(\mathbf{s}|\mathbf{t})=\sum_{\mathbf{a}}\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}f(s_j|t_{a_j})$中需要对遍历所有的对齐，即$\sum_{\mathbf{a}}$。这个过程可以被重新表示为

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \underbrace{\boxed{\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}}}_{1}\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}f(s_j|t_{a_j})$$

1. 遍历所有的的对齐$\mathbf{a}$。$\mathbf{a}$由$\{a_1,...,a_m\}$组成，每个$a_j\in\{a_1,...,a_m\}$从第译文开始位置(0)循环到截止位置(l)

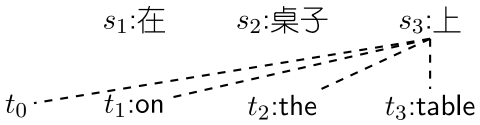


IBM模型1 - 完整模型(2)

- $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$中需要对遍历所有的对齐，即$\sum_{\mathbf{a}}$。这个过程可以被重新表示为

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \underbrace{\boxed{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}}_{①} \underbrace{\boxed{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})}}_{②}$$

1. 遍历所有的的对齐$\mathbf{a}$。$\mathbf{a}$由$\{a_1,...,a_m\}$组成，每个$a_j \in \{a_1, ..., a_m\}$从第译文开始位置(0)循环到截止位置($l$)



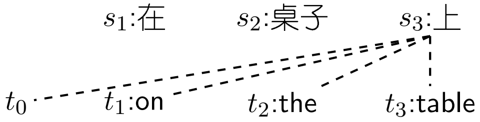
2. 对于每个$\mathbf{a}$累加对齐概率$\mathrm{P}(\mathbf{s}, \mathbf{a}|\mathbf{t})$

IBM模型1 - 完整模型(2)

- $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{\mathbf{a}} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})$中需要对遍历所有的对齐，即$\sum_{\mathbf{a}}$。这个过程可以被重新表示为

$$\boxed{\textbf{IBM模型1:}\quad \mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f(s_j|t_{a_j})}$$

1. 遍历所有的的对齐$\mathbf{a}$。$\mathbf{a}$由$\{a_1,...,a_m\}$组成，每个$a_j \in \{a_1,...,a_m\}$从第译文开始位置(0)循环到截止位置(l)



2. 对于每个$\mathbf{a}$累加对齐概率$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

IBM模型2的假设

- IBM模型1虽然形式非常简单，但是问题是忽略了翻译的调序问题。对于同一个源语言句子，单词翻译相同，但是译文顺序不同，模型得分（翻译概率相同）

$$\mathbf{s} = \text{在 桌子 上} \qquad \mathbf{s} = \text{在 桌子 上}$$

$$\mathbf{t} = \text{on the table} \qquad \mathbf{t}' = \text{table on the}$$

IBM模型1: $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \mathrm{P}(\mathbf{s}|\mathbf{t}')$

IBM模型2的假设

- IBM模型1虽然形式非常简单，但是问题是忽略了翻译的调序问题。对于同一个源语言句子，单词翻译相同，但是译文顺序不同，模型得分（翻译概率相同）

$$\mathbf{s} = \text{在 桌子 上} \qquad \mathbf{s} = \text{在 桌子 上}$$

$$\mathbf{t} = \text{on the table} \qquad \mathbf{t}' = \text{table on the}$$

IBM模型1: $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \mathrm{P}(\mathbf{s}|\mathbf{t}')$

理想: $\mathrm{P}(\mathbf{s}|\mathbf{t}) > \mathrm{P}(\mathbf{s}|\mathbf{t}')$

IBM模型2的假设

- IBM模型1虽然形式非常简单，但是问题是忽略了翻译的调序问题。对于同一个源语言句子，单词翻译相同，但是译文顺序不同，模型得分（翻译概率相同）

$$\mathbf{s} = \text{在}\ \ \text{桌子}\ \ \text{上} \qquad \mathbf{s} = \text{在}\ \ \text{桌子}\ \ \text{上}$$

$$\mathbf{t} = \text{on}\ \ \text{the}\ \ \text{table} \qquad \mathbf{t}' = \text{table}\ \ \text{on}\ \ \text{the}$$

$$\textbf{IBM模型1:}\quad \mathrm{P}(\mathbf{s}|\mathbf{t}) = \mathrm{P}(\mathbf{s}|\mathbf{t}')$$

$$\textbf{理想:}\quad \mathrm{P}(\mathbf{s}|\mathbf{t}) > \mathrm{P}(\mathbf{s}|\mathbf{t}')$$

- **IBM模型2的假设**：对齐位置a_j的生成概率与语言单词位置j，源语句子长度m和译文长度和l有关。记为$a(a_j|j,m,l)$，称作a_j的对齐概率

$$\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,t) \equiv a(a_j|j,m,l)$$

其它假设与IBM模型1相同，
即$\mathrm{P}(m|\mathbf{t}) \equiv \epsilon$和$\mathrm{P}(s_j|a_1^j,s_1^{j-1},m,\mathbf{t}) \equiv f(s_j|t_{a_j})$

IBM模型2 - 完整模型

- 代入假设

$$
\begin{aligned}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) &= \underbrace{\mathrm{P}(m|\mathbf{t})}_{\equiv\ \epsilon} \prod_{j=1}^{m} \underbrace{\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})}_{\equiv\ a(a_j|j,m,l)}\ \underbrace{\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})}_{\equiv\ f(s_j|t_{a_j})} \\
&= \epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})
\end{aligned}
$$

IBM模型2 - 完整模型

- 代入假设

$$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) = \underbrace{\mathrm{P}(m|\mathbf{t})}_{\equiv\ \epsilon} \prod_{j=1}^{m} \underbrace{\mathrm{P}(a_j|a_1^{j-1}, s_1^{j-1}, m, \mathbf{t})}_{\equiv\ a(a_j|j,m,l)} \underbrace{\mathrm{P}(s_j|a_1^{j}, s_1^{j-1}, m, \mathbf{t})}_{\equiv\ f(s_j|t_{a_j})}$$

$$= \epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$$

- 将上式代入$\mathbf{P}(\mathbf{s}|\mathbf{t}) = \sum_a \mathbf{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \underbrace{\boxed{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}}_{①} \underbrace{\boxed{\epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})}}_{②}$$

1. 遍历所有的的对齐$\mathbf{a}$
2. 对于每个$\mathbf{a}$累加对齐概率$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$，即计算$\epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$

IBM模型2 - 完整模型

- 代入假设

$$
\begin{array}{rcl}
\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t}) & = & \underset{\underset{\displaystyle\epsilon}{|||}}{\mathrm{P}(m|\mathbf{t})} \prod_{j=1}^{m} \underset{\underset{\displaystyle a(a_j|j,m,l)}{|||}}{\mathrm{P}(a_j|a_1^{j-1},s_1^{j-1},m,\mathbf{t})} \underset{\underset{\displaystyle f(s_j|t_{a_j})}{|||}}{\mathrm{P}(s_j|a_1^{j},s_1^{j-1},m,\mathbf{t})} \\
 & = & \epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})
\end{array}
$$

- 将上式代入$\mathbf{P}(\mathbf{s}|\mathbf{t}) = \sum_{a} \mathbf{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$

$$
\boxed{\textbf{IBM模型2:}\quad \mathrm{P}(\mathbf{s}|\mathbf{t}) = \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})}
$$

1. 遍历所有的的对齐$\mathbf{a}$
2. 对于每个$\mathbf{a}$累加对齐概率$\mathrm{P}(\mathbf{s},\mathbf{a}|\mathbf{t})$，即计算$\epsilon \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$

停顿一下，重新看看这两个模型

$$\textbf{IBM模型1}:\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\dots\sum_{a_m=0}^{l}\prod_{j=1}^{m}f(s_j|t_{a_j})$$

$$\textbf{IBM模型2}:\mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon\sum_{a_1=0}^{l}\dots\sum_{a_m=0}^{l}\prod_{j=1}^{m}a(a_j|j,m,l)f(s_j|t_{a_j})$$

- 参数
  - ϵ为常量

停顿一下，重新看看这两个模型

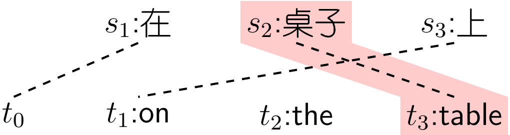
$$\textbf{IBM模型1}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m}\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m}f(s_j|t_{a_j})$$

$$\textbf{IBM模型2}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m}a(a_j|j,m,l)f(s_j|t_{a_j})$$

- 参数
  - ϵ为常量
  - $f(s_j|t_{a_j})$: 给定单词t_{a_j}翻译为s_j的概率

$f(s_2|t_{a_2}) = f(\text{桌子}|\text{table})$

表示给定'table'翻译为'桌子'的概率

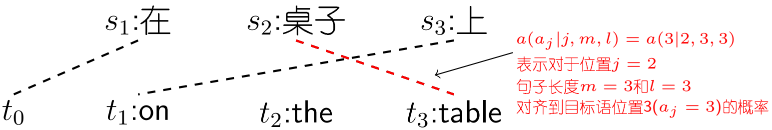


停顿一下，重新看看这两个模型

$$\textbf{IBM模型1}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j})$$

$$\textbf{IBM模型2}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$$

- 参数
  - ϵ为常量
  - $f(s_j|t_{a_j})$: 给定单词t_{a_j}翻译为s_j的概率
  - $a(a_j|j,m,l)$: 给定位置j和句对长度m和l，对齐a_j的概率



停顿一下，重新看看这两个模型(2)

$$\textbf{IBM模型1}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j})$$

$$\textbf{IBM模型2}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$$

- 对于翻译模型$\mathrm{P}(\mathbf{s}|\mathbf{t})$，再来回顾一下统计机器翻译的三个基本问题
  1. **建模**：如何描述$\mathrm{P}(\mathbf{s}|\mathbf{t})$
  2. **解码**：给定模型参数ϵ、$a(a_j|j,m,l)$和$f(s_j|t_{a_j})$，如何利用上面的公式计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$(语言模型计算暂不讨论)，并找到最佳译文$\hat{\mathbf{t}}$
  3. **训练**；如何从数据中自动学习模型参数ϵ、$a(a_j|j,m,l)$和$f(s_j|t_{a_j})$

停顿一下，重新看看这两个模型(2)

$$\text{IBM模型1}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j})$$

$$\text{IBM模型2}: \mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon \sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$$

- 对于翻译模型$\mathrm{P}(\mathbf{s}|\mathbf{t})$，再来回顾一下统计机器翻译的三个基本问题
  1. **建模**：如何描述$\mathrm{P}(\mathbf{s}|\mathbf{t})$ $\leftarrow$ 已解！见上面两个公式
  2. **解码**：给定模型参数ϵ、$a(a_j|j,m,l)$和$f(s_j|t_{a_j})$，如何利用上面的公式计算$\mathrm{P}(\mathbf{s}|\mathbf{t})$(语言模型计算暂不讨论)，并找到最佳译文$\hat{\mathbf{t}}$ $\leftarrow$ 下面讨论
  3. **训练**：如何从数据中自动学习模型参数ϵ、$a(a_j|j,m,l)$和$f(s_j|t_{a_j})$ $\leftarrow$ 下面讨论

Outline

1. 一个简单的翻译实例

2. IBM模型
   - 建模
   - 解码
   - 模型训练

解码 = 搜索 + 模型得分计算

- 何为模型：统计描述 + 参数

问题的统计描述：$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon \sum_{a_1=0}^{l} ... \sum_{a_m=0}^{l} \prod_{j=1}^{m} a(a_j|j,m,l) f(s_j|t_{a_j})$

模型的参数：$\epsilon =?; \quad \forall a_j, j, m, l : a(a_j|j,m,l) =?, f(s_j|t_{a_j}) =?$

解码 = 搜索 + 模型得分计算

- **何为模型**：统计描述 + 参数

问题的统计描述：	$\mathrm{P}(\mathbf{s}\vert\mathbf{t}) = \epsilon \sum_{a_1=0}^{l} ... \sum_{a_m=0}^{l} \prod_{j=1}^{m} a(a_j\vert j,m,l) f(s_j\vert t_{a_j})$
模型的参数：	$\epsilon = ?;\quad \forall a_j, j, m, l : a(a_j\vert j,m,l) = ?, f(s_j\vert t_{a_j}) = ?$

- **何为解码**：模型得分计算 + 找到模型得分最高的译文

模型得分计算：	对任意的$\mathbf{s}$和$\mathbf{t}$，(高效地)计算$\mathrm{P}(\mathbf{s}\vert\mathbf{t})$(同时计算$\mathrm{P}(\mathbf{t})$)
搜索：	对所有可能的$\mathbf{t}$，找到模型得分($\mathrm{P}(\mathbf{s}\vert\mathbf{t})\mathrm{P}(\mathbf{t})$)最高的译文输出

- ▶ **搜索(解码)问题**前面的实例已经描述了一种解法（见第16页）：自左向右添加译文单词 + 剪枝技术。这里不再讨论，可以自行学习
- ▶ **剩下的问题是**：对于任意的$\mathbf{s}$和$\mathbf{t}$，如何高效地计算$\mathrm{P}(\mathbf{s}\vert\mathbf{t})$

模型得分的计算

- $O((l+1)^m \cdot m)$ - IBM模型得分的直接计算几乎不可能！

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \underbrace{\sum_{a_1=0}^{l} \dots \sum_{a_m=0}^{l}}_{(l+1)^m\text{次循环}} \underbrace{\prod_{j=1}^{m} f(s_j|t_{a_j})}_{m\text{次循环}}$$

模型得分的计算

- $O((l+1)^m \cdot m)$ - IBM模型得分的直接计算几乎不可能!

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \underbrace{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}_{(l+1)^m\text{次循环}} \underbrace{\prod_{j=1}^{m} f(s_j|t_{a_j})}_{m\text{次循环}}$$

- $O(l \cdot m)$ - 实际上我们可以做的更好

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \underbrace{\prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}_{m \cdot l\text{次循环}}$$

模型得分的计算

- $O((l+1)^m \cdot m)$ - IBM模型得分的直接计算几乎不可能!

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \underbrace{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}_{(l+1)^m\text{次循环}} \underbrace{\prod_{j=1}^{m} f(s_j|t_{a_j})}_{m\text{次循环}}$$

- $O(l \cdot m)$ - 实际上我们可以做的更好

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$$

IBM模型1: $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$

模型得分的计算

- $O((l+1)^m \cdot m)$ - IBM模型得分的直接计算几乎不可能！

$$\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \underbrace{\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l}}_{(l+1)^m\text{次循环}} \underbrace{\prod_{j=1}^{m} f(s_j|t_{a_j})}_{m\text{次循环}}$$

- $O(l \cdot m)$ - 实际上我们可以做的更好

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$$

IBM模型1: $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$

类似的，IBM模型2: $\mathrm{P}(\mathbf{s}|\mathbf{t}) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} a(i|j,m,l) f(s_j|t_i)$

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$$

- 定义$\alpha(x,y)$为一个函数，x和y是两个变量

$$\boxed{\begin{aligned}&\alpha(1,0)\alpha(2,0)+\alpha(1,0)\alpha(2,1)+\alpha(1,0)\alpha(2,2)+\\&\alpha(1,1)\alpha(2,0)+\alpha(1,1)\alpha(2,1)+\alpha(1,1)\alpha(2,2)+\\&\alpha(1,2)\alpha(2,0)+\alpha(1,2)\alpha(2,1)+\alpha(1,2)\alpha(2,2)\end{aligned}}$$

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$$

- 定义$\alpha(x,y)$为一个函数，x和y是两个变量

$$\boxed{\begin{aligned}&\alpha(1,0)\alpha(2,0)+\alpha(1,0)\alpha(2,1)+\alpha(1,0)\alpha(2,2)+\\&\alpha(1,1)\alpha(2,0)+\alpha(1,1)\alpha(2,1)+\alpha(1,1)\alpha(2,2)+\\&\alpha(1,2)\alpha(2,0)+\alpha(1,2)\alpha(2,1)+\alpha(1,2)\alpha(2,2)\end{aligned}}$$

$$\boxed{\begin{aligned}&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\alpha(1,y_1)\alpha(2,y_2)\\=&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\prod_{x=1}^{2}\alpha(x,y_x)\end{aligned}}$$

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$$

- 定义$\alpha(x,y)$为一个函数，x和y是两个变量

$$\boxed{\begin{aligned}&\alpha(1,0)\alpha(2,0)+\alpha(1,0)\alpha(2,1)+\alpha(1,0)\alpha(2,2)+\\&\alpha(1,1)\alpha(2,0)+\alpha(1,1)\alpha(2,1)+\alpha(1,1)\alpha(2,2)+\\&\alpha(1,2)\alpha(2,0)+\alpha(1,2)\alpha(2,1)+\alpha(1,2)\alpha(2,2)\end{aligned}}$$

$$\boxed{\begin{aligned}&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\alpha(1,y_1)\alpha(2,y_2)\\=&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\prod_{x=1}^{2}\alpha(x,y_x)\end{aligned}}$$

$$\boxed{\begin{aligned}&(\alpha(1,0)+\alpha(1,1)+\alpha(1,2))\cdot\\&(\alpha(2,0)+\alpha(2,1)+\alpha(2,2))\\=&\prod_{x=1}^{2}\sum_{y=0}^{2}\alpha(x,y)\end{aligned}}$$

$$\sum_{a_1=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)$$

- 定义$\alpha(x,y)$为一个函数，x和y是两个变量

$$\boxed{\begin{aligned}&\alpha(1,0)\alpha(2,0)+\alpha(1,0)\alpha(2,1)+\alpha(1,0)\alpha(2,2)+\\&\alpha(1,1)\alpha(2,0)+\alpha(1,1)\alpha(2,1)+\alpha(1,1)\alpha(2,2)+\\&\alpha(1,2)\alpha(2,0)+\alpha(1,2)\alpha(2,1)+\alpha(1,2)\alpha(2,2)\end{aligned}}$$

$$\boxed{\begin{aligned}&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\alpha(1,y_1)\alpha(2,y_2)\\=&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\prod_{x=1}^{2}\alpha(x,y_x)\end{aligned}} = \boxed{\begin{aligned}&(\alpha(1,0)+\alpha(1,1)+\alpha(1,2))\cdot\\&(\alpha(2,0)+\alpha(2,1)+\alpha(2,2))\\=&\prod_{x=1}^{2}\sum_{y=0}^{2}\alpha(x,y)\end{aligned}}$$

$$\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m}\sum_{i=0}^{l} f(s_j|t_i)$$

- 定义$\alpha(x,y)$为一个函数，x和y是两个变量

$$\boxed{\begin{aligned}&\alpha(1,0)\alpha(2,0)+\alpha(1,0)\alpha(2,1)+\alpha(1,0)\alpha(2,2)+\\&\alpha(1,1)\alpha(2,0)+\alpha(1,1)\alpha(2,1)+\alpha(1,1)\alpha(2,2)+\\&\alpha(1,2)\alpha(2,0)+\alpha(1,2)\alpha(2,1)+\alpha(1,2)\alpha(2,2)\end{aligned}}$$

$$\boxed{\begin{aligned}&\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\alpha(1,y_1)\alpha(2,y_2)\\&=\sum_{y_1=0}^{2}\sum_{y_2=0}^{2}\prod_{x=1}^{2}\alpha(x,y_x)\end{aligned}} = \boxed{\begin{aligned}&(\alpha(1,0)+\alpha(1,1)+\alpha(1,2))\cdot\\&(\alpha(2,0)+\alpha(2,1)+\alpha(2,2))\\&=\prod_{x=1}^{2}\sum_{y=0}^{2}\alpha(x,y)\end{aligned}}$$

$$\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\prod_{j=1}^{m} f(s_j|t_{a_j}) = \prod_{j=1}^{m}\sum_{i=0}^{l} f(s_j|t_i)$$

Outline

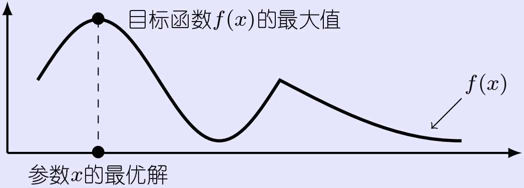
1. 一个简单的翻译实例

2. IBM模型
   - 建模
   - 解码
   - 模型训练

训练 - 目标函数

训练 = 优化

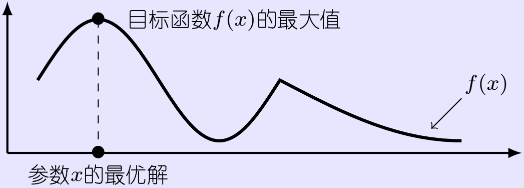
在训练集上调整参数使得目标函数的值达到最大（最小），此时的参数称作该模型在该目标函数下的最优解



训练 - 目标函数

训练 = 优化

在训练集上调整参数使得目标函数的值达到最大（最小），此时的参数称作该模型在该目标函数下的最优解



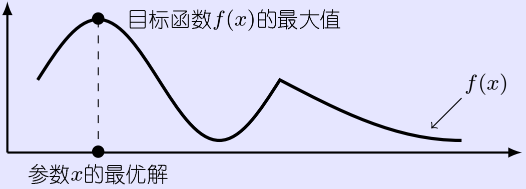
- **IBM模型的训练：对于给定的句对(s,t)，最大化翻译概率$\mathbf{P}(\mathrm{s}|\mathbf{t})$。**这里用符号$\mathrm{P}_\theta(\mathbf{s}|\mathbf{t})$表示概率由参数$\theta$决定

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

训练 - 目标函数

训练 = 优化

在训练集上调整参数使得目标函数的值达到最大（最小），此时的参数称作该模型在该目标函数下的最优解



- **IBM模型的训练**：对于给定的句对(s,t)，最大化翻译概率$\mathrm{P}(\mathrm{s}|\mathrm{t})$。这里用符号$\mathrm{P}_{\theta}(\mathrm{s}|\mathrm{t})$表示概率由参数$\theta$决定

$$\hat{\theta} = \underset{\theta}{\arg\max}\ \mathrm{P}_{\theta}(\mathrm{s}|\mathrm{t})$$

求最优参数　　目标函数

极大似然估计

- $\mathrm{P}(\mathbf{s}|\mathbf{t})$可以被看做是$(\mathbf{s},\mathbf{t})$上的似然函数$(L(\mathbf{s},\mathbf{t};\theta))$。所谓极大似然估计，就是要找到使$L(\mathbf{s},\mathbf{t};\theta)$达到最大的$\theta$：

$$\{\hat{\theta}\} \subseteq \{\arg\max_{\theta\in\Theta} L(\mathbf{s},\mathbf{t};\theta)\}$$

  $L(\mathbf{s},\mathbf{t};\theta)$表示$L(\cdot)$依赖模型参数$\theta$（注意分号），$\{\hat{\theta}\}$表示可能有多组结果，$\Theta$表示参数空间

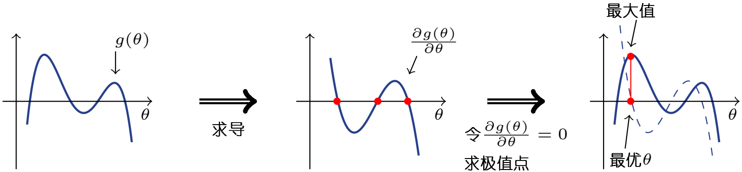
极大似然估计

- $\mathrm{P}(\mathbf{s}|\mathbf{t})$可以被看做是$(\mathbf{s},\mathbf{t})$上的似然函数$(L(\mathbf{s},\mathbf{t};\theta))$。所谓极大似然估计，就是要找到使$L(\mathbf{s},\mathbf{t};\theta)$达到最大的$\theta$：

$$\{\hat{\theta}\} \subseteq \{\arg\max_{\theta\in\Theta} L(\mathbf{s},\mathbf{t};\theta)\}$$

  $L(\mathbf{s},\mathbf{t};\theta)$表示$L(\cdot)$依赖模型参数$\theta$（注意分号），$\{\hat{\theta}\}$表示可能有多组结果，$\Theta$表示参数空间

- 先不用考虑上面的公式。我们还是回归到原始问题：如何找到一组θ使$\mathrm{P}_\theta(\mathbf{s}|\mathbf{t})$达到最大？
  - **求函数最大值问题**。比如，我们可以对$\mathrm{P}_\theta(\mathbf{s}|\mathbf{t})$求导，令导数为零，得到极值点



最大化P(s|t)

- 以IBM模型1为例，它所对应的(基于极大似然估计)的模型训练问题可以被描述为：

$$\max\ \left(\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\right)$$

$$\text{s.t.}\quad \text{for each word } t_y\text{:}\quad \sum_{s_x} f(s_x|t_y) = 1$$

最大化P(s|t)

- 以IBM模型1为例，它所对应的(基于极大似然估计）的模型训练问题可以被描述为：

最大化函数　　目标函数　　参数

$$\max \left(\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \right)$$

s.t.　for each word t_y: $\sum_{s_x} f(s_x|t_y) = 1$

subject to=满足...约束　　翻译概率归一化约束条件

  - 注意：$\{f(s_x|t_y)\}$对应很多参数，每个源语言单词和每个目标语言单词的组合都对应一个$f(s_x|t_y)$

拉格朗日乘数法 - Lagrange multiplier

- 含有约束的优化问题: 不好解
  目标: $\max(\mathrm{P}_{\theta}(\mathbf{s}|\mathbf{t}))$ + **约束:** $\forall t_y : \sum_{s_x} \mathrm{P}(s_x|t_y) = 1$

拉格朗日乘数法 - Lagrange multiplier

- 含有约束的优化问题: 不好解
  目标: $\max(\mathrm{P}_\theta(\mathbf{s}|\mathbf{t}))$ + **约束:** $\forall t_y : \sum_{s_x} \mathrm{P}(s_x|t_y) = 1$
- **解决方法**: 含有约束优化 $\Rightarrow$ 不含约束优化

拉格朗日乘数法 - 百度百科

拉格朗日乘数法是一种寻找变量受一个或多个条件所限制的多元函数的极值的方法。这种方法将一个有n个变量与k个约束条件的最优化问题转换为一个有$n+k$个变量的方程组的极值问题，其变量不受任何约束。这种方法引入了一种新的标量未知数，即拉格朗日乘数。

  - 对于每个t_y所对应的约束，引入一个拉格朗日乘数λ_{t_y}

$$
\begin{aligned}
L(f,\lambda) &= \frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l} f(s_j|t_i) - \\
&\quad \sum_{t_y}\lambda_{t_y}\Big(\sum_{s_x} f(s_x|t_y) - 1\Big)
\end{aligned}
$$

拉格朗日乘数法 - Lagrange multiplier (2)

原始问题	转化后的问题
$\max(\mathrm{P}(\mathbf{s}\|\mathbf{t}))$	$\max(L(f,\lambda))$
s.t. $\forall t_y : \sum_{s_x} f(s_x\|t_y) = 1$	

新目标函数　　旧目标函数

$$\underbrace{L(f,\lambda)}_{\text{新目标函数}} = \underbrace{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}_{\text{旧目标函数}} - \sum_{t_y} \lambda_{t_y} \left(\sum_{s_x} f(s_x|t_y) - 1\right)$$

拉格朗日乘数法 - Lagrange multiplier (2)

原始问题	转化后的问题
$\max(\mathrm{P}(\mathbf{s}\vert\mathbf{t}))$	$\max(L(f,\lambda))$
s.t. $\forall t_y : \sum_{s_x} f(s_x\vert t_y) = 1$	

新目标函数　　旧目标函数　　参数：词翻译概率

$$L(f,\lambda) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) - \sum_{t_y} \lambda_{t_y} \left(\sum_{s_x} f(s_x|t_y) - 1\right)$$

参数：拉格朗日乘数　　参数约束条件

拉格朗日乘数法 - Lagrange multiplier (2)

原始问题	转化后的问题
$\max(\mathrm{P}(\mathbf{s}\|\mathbf{t}))$ s.t. $\forall t_y : \sum_{s_x} f(s_x\|t_y) = 1$	$\max(L(f,\lambda))$

新目标函数

旧目标函数

参数：词翻译概率

$$L(f,\lambda) = \frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l} \boxed{f(s_j|t_i)} - \sum_{t_y} \lambda_{t_y} \left(\sum_{s_x} \boxed{f(s_x|t_y)} - 1\right)$$

参数：拉格朗日乘数

参数约束条件

- 剩下的问题：计算$\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)}$，并求解$\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)} = 0$

对L求导

- 令s_u和t_v表示任意一个源语单词和一个目标语单词

$$
\begin{aligned}
\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)} &= \frac{\partial\big[\frac{\epsilon}{(l+1)^m}\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_i)\big]}{\partial f(s_u|t_v)} - \\
&\quad \frac{\partial\big[\sum_{t_y}\lambda_{t_y}(\sum_{s_x}f(s_x|t_y)-1)\big]}{\partial f(s_u|t_v)} \\
&= \frac{\epsilon}{(l+1)^m}\cdot\frac{\partial\big[\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_i)\big]}{\partial f(s_u|t_v)} - \lambda_{t_v}
\end{aligned}
$$

对L求导

- 令s_u和t_v表示任意一个源语单词和一个目标语单词

$$
\begin{aligned}
\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)} &= \frac{\partial\big[\frac{\epsilon}{(l+1)^m}\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_i)\big]}{\partial f(s_u|t_v)} - \\
&\quad \frac{\partial\big[\sum_{t_y}\lambda_{t_y}(\sum_{s_x}f(s_x|t_y)-1)\big]}{\partial f(s_u|t_v)} \\
&= \frac{\epsilon}{(l+1)^m}\cdot\frac{\partial\big[\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_i)\big]}{\partial f(s_u|t_v)} - \lambda_{t_v}
\end{aligned}
$$

- 为了求$\frac{\partial\big[\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_i)\big]}{\partial f(s_u|t_v)}$，先看一个简单的例子：
  $g(z)=\alpha z^{\beta}$为一个变量z的多项式函数，显然

$$\frac{\partial g(z)}{\partial z} = \alpha\beta z^{\beta-1} = \frac{\beta}{z}\alpha z^{\beta} = \frac{\beta}{z}g(z)$$

对L求导(2)

- 这里可以把$\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)$看做$g(z)=\alpha z^{\beta}$的实例
  - 令$z=\sum_{i=0}^{l}f(s_u|t_i)$，注意$s_u$为给定的源语单词
  - 那么β为$\sum_{i=0}^{l}f(s_u|t_i)$在$\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)$中出现的次数，即源语句子中与$s_u$相同的单词的个数

$$\beta=\sum_{j=1}^{m}\delta(s_j,s_u)$$

$\delta(x,y)$：当$x=y$时为1，否则为0

对L求导(2)

- 这里可以把$\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)$看做$g(z)=\alpha z^{\beta}$的实例
  - 令$z=\sum_{i=0}^{l}f(s_u|t_i)$，注意$s_u$为给定的源语单词
  - 那么β为$\sum_{i=0}^{l}f(s_u|t_i)$在$\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)$中出现的次数，即源语句子中与$s_u$相同的单词的个数

$$\beta=\sum_{j=1}^{m}\delta(s_j,s_u)$$

$\delta(x,y)$：当$x=y$时为1，否则为0

- 根据$\frac{\partial g(z)}{\partial z}=\frac{\beta}{z}g(z)$

$$\frac{\partial g(z)}{\partial z}=\frac{\partial\big[\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\big]}{\partial\big[\sum_{i=0}^{l}f(s_u|t_i)\big]}=\frac{\sum_{j=1}^{m}\delta(s_j,s_u)}{\sum_{i=0}^{l}f(s_u|t_i)}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)$$

对L求导(3)

- 如果把z看成f的函数，有$\frac{\partial g(z)}{\partial f} = \frac{\partial g(z)}{\partial z} \cdot \frac{\partial z}{\partial f}$
  - 因为$z = \sum_{i=0}^{l} f(s_u|t_i)$，很容易得到

$$\frac{\partial z}{\partial f} = \frac{\partial \big[\sum_{i=0}^{l} f(s_u|t_i)\big]}{\partial f(s_u|t_v)} = \sum_{i=0}^{l} \delta(t_i, t_v)$$

即目标语译文单词中与t_v相同的个数

对L求导(3)

- 如果把z看成f的函数，有$\frac{\partial g(z)}{\partial f} = \frac{\partial g(z)}{\partial z} \cdot \frac{\partial z}{\partial f}$
  - 因为$z = \sum_{i=0}^{l} f(s_u|t_i)$，很容易得到

$$\frac{\partial z}{\partial f} = \frac{\partial \left[\sum_{i=0}^{l} f(s_u|t_i) \right]}{\partial f(s_u|t_v)} = \sum_{i=0}^{l} \delta(t_i, t_v)$$

    即目标语译文单词中与t_v相同的个数

  - 根据$\frac{\partial g(z)}{\partial z}$和$\frac{\partial z}{\partial f}$计算的结果，可以得到

$$\frac{\partial \left[\prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \right]}{\partial f(s_u|t_v)} = \underbrace{\frac{\partial \left[\prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \right]}{\partial \left[\sum_{i=0}^{l} f(s_u|t_i) \right]}}_{\partial g(z)/\partial z} \cdot \underbrace{\frac{\partial \left[\sum_{i=0}^{l} f(s_u|t_i) \right]}{\partial f(s_u|t_v)}}_{\partial z/\partial f}$$

$$= \frac{\sum_{j=1}^{m} \delta(s_j, s_u)}{\sum_{i=0}^{l} f(s_u|t_i)} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \cdot \sum_{i=0}^{l} \delta(t_i, t_v)$$

对L求导(4)

- 将$\frac{\partial\left[\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\right]}{\partial f(s_u|t_v)}$进一步代入$\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)}$

$$
\begin{aligned}
&\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)}\\
=\ &\frac{\epsilon}{(l+1)^m}\cdot\frac{\partial\left[\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_{a_j})\right]}{\partial f(s_u|t_v)}-\lambda_{t_v}\\
=\ &\frac{\epsilon}{(l+1)^m}\cdot\frac{\sum_{j=1}^{m}\delta(s_j,s_u)\cdot\sum_{i=0}^{l}\delta(t_i,t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)-\lambda_{t_v}
\end{aligned}
$$

对L求导(4)

- 将$\frac{\partial\left[\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\right]}{\partial f(s_u|t_v)}$进一步代入$\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)}$

$$
\begin{aligned}
& \frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)} \\
= & \frac{\epsilon}{(l+1)^m}\cdot\frac{\partial\left[\prod\limits_{j=1}^{m}\sum\limits_{i=0}^{l}f(s_j|t_{a_j})\right]}{\partial f(s_u|t_v)}-\lambda_{t_v} \\
= & \frac{\epsilon}{(l+1)^m}\cdot\frac{\sum_{j=1}^{m}\delta(s_j,s_u)\cdot\sum_{i=0}^{l}\delta(t_i,t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)-\lambda_{t_v}
\end{aligned}
$$

- 令$\frac{\partial L(f,\lambda)}{\partial f(s_u|t_v)}=0$，有

$$
f(s_u|t_v)=\frac{\lambda_{t_v}^{-1}\epsilon}{(l+1)^m}\cdot\frac{\sum\limits_{j=1}^{m}\delta(s_j,s_u)\cdot\sum\limits_{i=0}^{l}\delta(t_i,t_v)}{\sum\limits_{i=0}^{l}f(s_u|t_i)}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\cdot f(s_u|t_v)
$$

求$f(s_u|t_v)$的最优解 - 这不是一个解析式！！！

- 注意：这不是一个计算$f(s_u|t_v)$的解析式，因为等式右端仍含有$f(s_u|t_v)$

$$\boxed{f(s_u|t_v)} = \lambda_{t_v}^{-1} \frac{\epsilon}{(l+1)^m} \boxed{\prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)} \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \boxed{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}$$

求$f(s_u|t_v)$的最优解 - 这不是一个解析式！！！

- 注意：这不是一个计算$f(s_u|t_v)$的解析式，因为等式右端仍含有$f(s_u|t_v)$

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

- 怎么办? - 我们仍可以利用上式迭代地计算$f(s_u|t_v)$，使其最终收敛到最优值
  - 这是一个非常经典的方法，称作期望最大化(Expectation Maximization)，简称EM方法(或算法)
  - 思想：用当前的参数，求一个似然函数的期望，之后最大化这个期望值同时得到新的一组参数的值
  - 对于IBM模型，很简单：反复使用上式即可！

新的参数值　　　　　　　　　　　　　　旧的参数值

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

求$f(s_u|t_v)$的最优解 - 这不是一个解析式！！！

- 注意：这不是一个计算$f(s_u|t_v)$的解析式，因为等式右端仍含有$f(s_u|t_v)$

$$\boxed{f(s_u|t_v)} = \lambda_{t_v}^{-1} \frac{\epsilon}{(l+1)^m} \boxed{\prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)} \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \boxed{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}$$

- 怎么办? - 我们仍可以利用上式迭代地计算$f(s_u|t_v)$，使其最终收敛到最优值
  - 这是一个非常经典的方法，称作期望最大化(Expectation Maximization)，简称EM方法(或算法)
  - 思想：用当前的参数，求一个似然函数的期望，之后最大化这个期望值同时得到新的一组参数的值
  - 对于IBM模型，很简单：反复使用上式即可！

新的参数值　　问题：$\lambda_{t_v} =?$　　旧的参数值

$$\underbrace{\boxed{f(s_u|t_v)}}_{\text{新的参数值}} = \underbrace{\boxed{\lambda_{t_v}^{-1}}}_{\text{问题：}\lambda_{t_v}=?} \frac{\epsilon}{(l+1)^m} \boxed{\prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)} \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \boxed{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}$$

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \tfrac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i) \ \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \ \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \overbrace{\boxed{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}}^{\text{翻译概率}\mathbf{P}(\mathbf{s}|\mathbf{t})} \sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

$$||$$

$$\mathrm{P}(\mathbf{s}|\mathbf{t})$$

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \underbrace{\overbrace{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}^{\text{翻译概率}\mathbf{P}(\mathbf{s}|\mathbf{t})}}_{= \mathrm{P}(\mathbf{s}|\mathbf{t})} \overbrace{\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v)}^{(s_u, t_v)\text{在句对}(\mathbf{s},\mathbf{t})\text{中配对的总次数}} \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

翻译概率$\mathbf{P}(\mathbf{s}|\mathbf{t})$

(s_u,t_v)在句对$(\mathbf{s},\mathbf{t})$中配对的总次数

$f(s_u|t_v)$对于所有的t_i的相对值

$$f(s_u|t_v)=\lambda_{t_v}^{-1}\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_v)\frac{f(s_u|t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}$$

$$\frac{\epsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i) = \mathrm{P}(\mathbf{s}|\mathbf{t})$$

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \underbrace{\overbrace{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}^{\text{翻译概率}\mathbf{P}(\mathbf{s}|\mathbf{t})}}_{=\ \mathrm{P}(\mathbf{s}|\mathbf{t})} \underbrace{\overbrace{\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v)}^{(s_u,t_v)\text{在句对}(\mathbf{s},\mathbf{t})\text{中配对的总次数}} \overbrace{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}^{f(s_u|t_v)\text{对于所有的}t_i\text{的相对值}}}_{\text{'}t_v\text{翻译为}s_u\text{'这个事件出现次数的期望的估计，称之为期望频次expected count}}$$

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \underbrace{\overbrace{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}^{\text{翻译概率}\mathbf{P}(\mathbf{s}|\mathbf{t})}}_{=\ \mathrm{P}(\mathbf{s}|\mathbf{t})} \underbrace{\overbrace{\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v)}^{(s_u,t_v)\text{在句对}(\mathbf{s},\mathbf{t})\text{中配对的总次数}} \overbrace{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}^{f(s_u|t_v)\text{对于所有的}t_i\text{的相对值}}}_{\text{'}t_v\text{翻译为}s_u\text{'这个事件出现次数的期望的估计 称之为期望频次expected count}}$$

't_v翻译为s_u'这个事件
出现次数的期望的估计
称之为期望频次**expected count**

- **啥是事件的期望频次？？？** - 事件在其分布下出现的次数的期望$c_{\mathbb{E}}(X) = \sum_i c(x_i) \cdot \mathrm{P}(x_i)$

x_i	$c(x_i)$
x_1	2
x_2	1
x_3	5

x_i	$c(x_i)$	$\mathrm{P}(x_i)$	$c(x_i) \cdot \mathrm{P}(x_i)$
x_1	2	0.1	0.2
x_2	1	0.3	0.3
x_3	5	0.2	1.0

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \underbrace{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}_{\text{翻译概率}\mathbf{P}(\mathbf{s}|\mathbf{t})\ =\ \mathrm{P}(\mathbf{s}|\mathbf{t})} \underbrace{\underbrace{\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v)}_{(s_u,t_v)\text{在句对}(\mathbf{s},\mathbf{t})\text{中配对的总次数}} \underbrace{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}_{f(s_u|t_v)\text{对于所有的}t_i\text{的相对值}}}_{\text{'}t_v\text{翻译为}s_u\text{'这个事件出现次数的期望的估计，称之为期望频次expected count}}$$

- 啥是事件的期望频次？？？ - 事件在其分布下出现的次数的期望$c_{\mathbb{E}}(X) = \sum_i c(x_i) \cdot \mathrm{P}(x_i)$

x_i	$c(x_i)$
x_1	2
x_2	1
x_3	5
	$c(X) = 8$

x_i	$c(x_i)$	$\mathrm{P}(x_i)$	$c(x_i) \cdot \mathrm{P}(x_i)$
x_1	2	0.1	0.2
x_2	1	0.3	0.3
x_3	5	0.2	1.0

对$f(s_u|t_v)$的计算进行化简

- 为了计算$f(s_u|t_v)$，重新组织一下上面的公式

$$f(s_u|t_v) = \lambda_{t_v}^{-1} \underbrace{\frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} f(s_j|t_i)}_{\text{翻译概率}\mathbf{P}(\mathbf{s}|\mathbf{t}) \;=\; \mathrm{P}(\mathbf{s}|\mathbf{t})} \underbrace{\underbrace{\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v)}_{(s_u,t_v)\text{在句对}(\mathbf{s},\mathbf{t})\text{中配对的总次数}} \underbrace{\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}}_{f(s_u|t_v)\text{对于所有的}t_i\text{的相对值}}}_{\text{'}t_v\text{翻译为}s_u\text{'这个事件出现次数的期望的估计，称之为期望频次 expected count}}$$

- **啥是事件的期望频次？？？** - 事件在其分布下出现的次数的期望$c_{\mathbb{E}}(X) = \sum_i c(x_i) \cdot \mathrm{P}(x_i)$

x_i	$c(x_i)$
x_1	2
x_2	1
x_3	5
$c(X) = 8$	

x_i	$c(x_i)$	$\mathrm{P}(x_i)$	$c(x_i) \cdot \mathrm{P}(x_i)$
x_1	2	0.1	0.2
x_2	1	0.3	0.3
x_3	5	0.2	1.0
$c_{\mathbb{E}}(X) = 0.2 + 0.3 + 1.0 = 1.5$			

期望频次

$$\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \qquad \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

期望频次

$$\boxed{\sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_v)}$$

't_v翻译为s_u'在所有对齐中出现的次数

$$c('t_v\text{翻译为}s_u')$$

$$\frac{f(s_u|t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}$$

期望频次

$\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v)$	$\frac{f(s_u \mid t_v)}{\sum_{i=0}^{l} f(s_u \mid t_i)}$
't_v翻译为s_u'在所有对齐中出现的次数	't_v翻译为s_u'在所有对齐中出现的相对概率
c('t_v翻译为s_u')	P('t_v翻译为s_u')

期望频次

$$\sum_{j=1}^{m} \delta(s_j, s_u) \sum_{i=0}^{l} \delta(t_i, t_v) \quad \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

't_v翻译为s_u'在所有对齐中出现的次数

't_v翻译为s_u'在所有对齐中出现的相对概率

$$\begin{aligned} & c(\text{'}t_v\text{翻译为}s_u\text{'}) \quad \times \quad \mathrm{P}(\text{'}t_v\text{翻译为}s_u\text{'}) \\ = \quad & c_{\mathbb{E}}(\text{'}t_v\text{翻译为}s_u\text{'}) \end{aligned}$$

期望频次

$$\sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_v) \qquad \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

't_v翻译为s_u'在所有对齐中出现的次数　　't_v翻译为s_u'在所有对齐中出现的相对概率

$$c(\text{'}t_v\text{翻译为}s_u\text{'}) \quad \times \quad \mathrm{P}(\text{'}t_v\text{翻译为}s_u\text{'})$$

$$= \quad c_{\mathbb{E}}(\text{'}t_v\text{翻译为}s_u\text{'})$$

- 定义：在$\mathrm{P}(\mathbf{s}|\mathbf{t})$中，$t_v$翻译(连接)到$s_u$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \equiv \sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_v)\cdot\frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

期望频次

$$\sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_v) \qquad \frac{f(s_u|t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}$$

't_v翻译为s_u'在所有对齐中出现的次数

't_v翻译为s_u'在所有对齐中出现的相对概率

$$c(\text{'}t_v\text{翻译为}s_u\text{'}) \quad \times \quad \mathrm{P}(\text{'}t_v\text{翻译为}s_u\text{'})$$

$$= \quad c_{\mathbb{E}}(\text{'}t_v\text{翻译为}s_u\text{'})$$

- 定义：在$\mathrm{P}(\mathbf{s}|\mathbf{t})$中，$t_v$翻译(连接)到$s_u$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \equiv \sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_v)\cdot\frac{f(s_u|t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}$$

- 重写$f(s_u|t_v)$!!!

$$\boxed{f(s_u|t_v) = \lambda_{t_v}^{-1}\cdot\mathrm{P}(\mathbf{s}|\mathbf{t})\cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t})}$$

通过期望频次计算$f(s_u|t_v)$

- 一个小trick: 令$\lambda'_{t_v} = \frac{\lambda_{t_v}}{\mathrm{P}(\mathbf{s}|\mathbf{t})}$

$$\begin{aligned} f(s_u|t_v) &= \lambda_{t_v}^{-1} \cdot \mathrm{P}(\mathbf{s}|\mathbf{t}) \cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \\ &= (\lambda'_{t_v})^{-1} \cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \end{aligned}$$

通过期望频次计算$f(s_u|t_v)$

- 一个小trick：令$\lambda'_{t_v} = \frac{\lambda_{t_v}}{\mathrm{P}(\mathbf{s}|\mathbf{t})}$

$$\begin{aligned} f(s_u|t_v) &= \lambda_{t_v}^{-1} \cdot \mathrm{P}(\mathbf{s}|\mathbf{t}) \cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \\ &= (\lambda'_{t_v})^{-1} \cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \end{aligned}$$

- λ'_{t_v}究竟是什么？ - 回忆一下IBM模型对$f(\cdot|\cdot)$的约束

$$\forall t_y : \sum_{s_x} f(s_x|t_y) = 1$$

为了满足$f(\cdot|\cdot)$的概率归一化约束，显然

$$\lambda'_{t_v} = \sum_{s_u} c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t})$$

通过期望频次计算$f(s_u|t_v)$

- 一个小trick：令$\lambda'_{t_v} = \frac{\lambda_{t_v}}{\mathrm{P}(\mathbf{s}|\mathbf{t})}$

$$
\begin{aligned}
f(s_u|t_v) &= \lambda_{t_v}^{-1} \cdot \mathrm{P}(\mathbf{s}|\mathbf{t}) \cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) \\
&= (\lambda'_{t_v})^{-1} \cdot c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t})
\end{aligned}
$$

- λ'_{t_v}究竟是什么？ - 回忆一下IBM模型对$f(\cdot|\cdot)$的约束

$$\forall t_y : \sum_{s_x} f(s_x|t_y) = 1$$

为了满足$f(\cdot|\cdot)$的概率归一化约束，显然

$$\lambda'_{t_v} = \sum_{s_u} c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t})$$

- $f(s_u|t_v)$的计算式

$$\boxed{f(s_u|t_v) = \frac{c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t})}{\sum_{s_u} c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t})}}$$

在整个数据集上计算

- **更真实的情况**：我们拥有一系列互译的句对（称作平行语料），记为$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),(\mathbf{s}^{[2]},\mathbf{t}^{[2]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$。对于这$K$个训练用句对，定义$f(s_u|t_v)$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v)=\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$$

在整个数据集上计算

- 更真实的情况：我们拥有一系列互译的句对（称作平行语料），记为$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),(\mathbf{s}^{[2]},\mathbf{t}^{[2]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$。对于这$K$个训练用句对，定义$f(s_u|t_v)$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v)=\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$$

- 于是

$$f(s_u|t_v)=\frac{\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{s_u}\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

在整个数据集上计算

- 更真实的情况：我们拥有一系列互译的句对（称作平行语料），记为$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),(\mathbf{s}^{[2]},\mathbf{t}^{[2]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$。对于这$K$个训练用句对，定义$f(s_u|t_v)$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v) = \sum_{k=1}^{K} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$$

- 于是

$$f(s_u|t_v) = \frac{\sum_{k=1}^{K} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{s_u}\sum_{k=1}^{K} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

用当前的$f(s_u|t_v)$
计算期望频次$c_{\mathbb{E}}(\cdot)$

在整个数据集上计算

- 更真实的情况：我们拥有一系列互译的句对（称作平行语料），记为$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),(\mathbf{s}^{[2]},\mathbf{t}^{[2]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$。对于这$K$个训练用句对，定义$f(s_u|t_v)$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v)=\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$$

- 于是

$$f(s_u|t_v)=\frac{\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{s_u}\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

利用这个公式计算新的$f(s_u|t_v)$值

用当前的$f(s_u|t_v)$计算期望频次$c_{\mathbb{E}}(\cdot)$

在整个数据集上计算

- 更真实的情况：我们拥有一系列互译的句对（称作平行语料），记为$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),(\mathbf{s}^{[2]},\mathbf{t}^{[2]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$。对于这$K$个训练用句对，定义$f(s_u|t_v)$的期望频次为

$$c_{\mathbb{E}}(s_u|t_v)=\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$$

- 于是

$$\boxed{f(s_u|t_v)}=\boxed{\frac{\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{s_u}\sum_{k=1}^{K}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}}$$

利用这个公式计算新的$f(s_u|t_v)$值

反复执行

用当前的$f(s_u|t_v)$计算期望频次$c_{\mathbb{E}}(\cdot)$

完整的模型训练方法 - EM算法

IBM模型1的训练（EM算法）

输入：平行语料$\{(\mathbf{s}^{[1]}, \mathbf{t}^{[1]}), ..., (\mathbf{s}^{[K]}, \mathbf{t}^{[K]})\}$

输出：参数$f(\cdot|\cdot)$的最优值

Function $\textsc{TrainItWithEM}(\{(\mathbf{s}^{[1]}, \mathbf{t}^{[1]}), ..., (\mathbf{s}^{[K]}, \mathbf{t}^{[K]})\})$

  Initialize $f(\cdot|\cdot)$ ▷ 比如给$f(\cdot|\cdot)$一个均匀分布

  Loop until $f(\cdot|\cdot)$ converges

    foreach $k = 1$ to K **do**

$$c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]}) = \sum_{j=1}^{|\mathbf{s}^{[k]}|} \delta(s_j, s_u) \sum_{i=0}^{|\mathbf{t}^{[k]}|} \delta(t_i, t_v) \cdot \frac{f(s_u|t_v)}{\sum_{i=0}^{l} f(s_u|t_i)}$$

    foreach t_v appears at least one of $\{\mathbf{t}^{[1]}, ..., \mathbf{t}^{[K]}\}$ **do**

$$\lambda_{t_v}^{'} = \sum_{s_u} \sum_{k=1}^{N} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$$

      foreach s_u appears at least one of $\{\mathbf{s}^{[1]}, ..., \mathbf{s}^{[K]}\}$ **do**

$$f(s_u|t_v) = \sum_{k=1}^{N} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]}) \cdot (\lambda_{t_v}^{'})^{-1}$$

  return $f(\cdot|\cdot)$

完整的模型训练方法 - EM算法

IBM模型1的训练（EM算法）

输入：平行语料$\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$
输出：参数$f(\cdot|\cdot)$的最优值

1: **Function** TrainItWithEM($\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),...,(\mathbf{s}^{[K]},\mathbf{t}^{[K]})\}$)
2: Initialize $f(\cdot|\cdot)$ ▷ 比如给$f(\cdot|\cdot)$一个均匀分布
3: Loop until $f(\cdot|\cdot)$ converges
4: **foreach** $k=1$ to K **do**
5: $c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]}) = \sum_{j=1}^{|\mathbf{s}^{[k]}|}\delta(s_j,s_u)\sum_{i=0}^{|\mathbf{t}^{[k]}|}\delta(t_i,t_v)\cdot\frac{f(s_u|t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}$
6: **foreach** t_v appears at least one of $\{\mathbf{t}^{[1]},...,\mathbf{t}^{[K]}\}$ **do**
7: $\lambda_{t_v}^{'} = \sum_{s_u}\sum_{k=1}^{N}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})$
8: **foreach** s_u appears at least one of $\{\mathbf{s}^{[1]},...,\mathbf{s}^{[K]}\}$ **do**
9: $f(s_u|t_v) = \sum_{k=1}^{N}c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})\cdot(\lambda_{t_v}^{'})^{-1}$
10: **return** $f(\cdot|\cdot)$

- **E-Step：行4-5，计算$c_{\mathbb{E}}(\cdot)$；M-Step：行6-9，计算$f(\cdot)$**

EM算法与IBM模型2

- EM算法可以直接用于训练IBM模型2(这里省去推导)
  - 参数1： $f(s_j|t_i)$ - 与IBM模型1一样
  - 参数2： $a(i|j,m,l)$ - 调序概率

1. **E-Step** (对于句对$(\mathbf{s},\mathbf{t})$， $m=|\mathbf{s}|, l=|\mathbf{t}|$)

$$c_{\mathbb{E}}(s_u|t_v;\mathbf{s},\mathbf{t}) = \sum_{j=1}^{m}\sum_{i=0}^{l}\frac{f(s_u|t_v)a(i|j,m,l)\delta(s_j,s_u)\delta(t_i,t_v)}{\sum_{k=0}^{l} f(s_u|t_k)a(k|j,m,l)}$$

$$c_{\mathbb{E}}(i|j,m,l;\mathbf{s},\mathbf{t}) = \frac{f(s_j|t_i)a(i|j,m,l)}{\sum_{k=0}^{l} f(s_j|t_k)a(k|j,m,l)}$$

2. **M-Step**

$$f(s_u|t_v) = \frac{\sum_{k=1}^{K} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{s_u}\sum_{k=1}^{K} c_{\mathbb{E}}(s_u|t_v;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

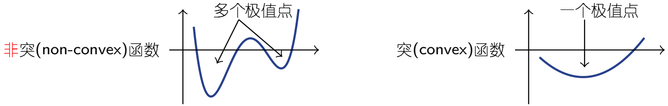
$$a(i|j,m,l) = \frac{\sum_{k=1}^{K} c_{\mathbb{E}}(i|j;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}{\sum_{i}\sum_{k=1}^{K} c_{\mathbb{E}}(i|j;\mathbf{s}^{[k]},\mathbf{t}^{[k]})}$$

IBM模型1和2基本介绍完了

- 再来回顾一下，IBM模型1和2包括以下三方面内容：
  1. **建模** - 第25页～第36页
  2. **解码** - 第38页
  3. **训练** - 第56页
- 还有很多很多问题没有讨论
  - 解码：第16页所描述的解码算法很粗糙
    - 需要考虑更多的翻译假设 - beam search - 随后章节会介绍

IBM模型1和2基本介绍完了

- 再来回顾一下，IBM模型1和2包括以下三方面内容：
  1. 建模 - 第25页～第36页
  2. 解码 - 第38页
  3. 训练 - 第56页
- 还有很多很多问题没有讨论
  - 解码：第16页所描述的解码算法很粗糙
    - 需要考虑更多的翻译假设 - beam search - 随后章节会介绍
  - 训练：EM算法的问题是：
    - 在目标函数为非突(凹)集函数时，容易陷入局部最优



      比较幸运的是IBM模型1有单个极值点，EM可以得到全局最优。而且通常会把IBM模型1的参数作为高阶模型(如模型2)的输入，保证后面的训练从较好的初始值开始
    - 会有degenerate analysis的问题
      Google一下variational EM和bayesian inference

GIZA++

- IBM模型3-5不是本教程的重点
  - 建模相对复杂一些，当然模型也更强大
  - 训练和解码不太简单
  - 推荐一些阅读材料
    - **A Program for Aligning Sentences in Bilingual Corpora**. William A. Gale and Kenneth W. Church. 1993.
    - **A systematic comparison of various statistical alignment models**. Franz Och and Hermann Ney. 2003.
    - *基于**IBM**模型的统计机器翻译技术的研究*. 肖桐. 2008. (硕士论文)

GIZA++

- IBM模型3-5不是本教程的重点
  - 建模相对复杂一些，当然模型也更强大
  - 训练和解码不太简单
  - 推荐一些阅读材料
    - **A Program for Aligning Sentences in Bilingual Corpora**. William A. Gale and Kenneth W. Church. 1993.
    - **A systematic comparison of various statistical alignment models**. Franz Och and Hermann Ney. 2003.
    - ***基于IBM模型的统计机器翻译技术的研究***. 肖桐. 2008. (硕士论文)
- 不得不提的**GIZA++**：实现了IBM模型+HMM模型，现在广泛用于双语平行语料的自动词对齐
  - 完全开源
    `http://code.google.com/p/giza-pp/`
  - NiuTrans中也有使用方式的介绍
    `http://www.nlplab.com/NiuPlan/NiuTrans.YourData.ch.html`

这章内容先写到这儿

这章内容写了很多，但是还没弄完 :(
后面(可能过两年)补上单词自动对齐的内容 :)