自然语言处理及应用
实验报告

名称	机器翻译 machine translation
姓名	罗福杰
班级	硕0078班
学号	3120305208
Email	1626027173@qq.com
日期	2020-10-31

1.实验目的

This assignment focuses **on machine translation. The goal is to implement two translation models, IBM model 1 and IBM model 2**, and apply these models to predict English/Chinese word alignments.

Available resources:

- *English/Chinese parallel corpus: http://www.datatang.com/data/14779/
- (using "cn.txt" and "en.txt" as parallel corpus or make parallel corpus by yourself)
- GIZA++: https://code.google.com/p/giza-pp/

1) 理解并掌握统计机器翻译中的 IBM module 1 和 IBM module 2 中的概念和原理。

2) 了解EM算法，并理解该算法在统计机器学习中的应用。

2.实验环境

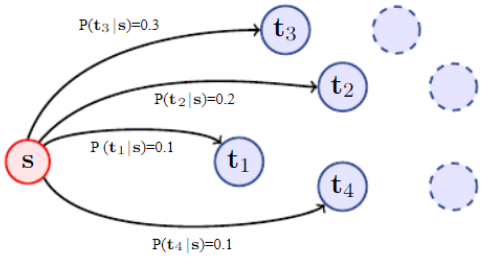
- Win 10 操作系统 64位；
- Spyder编辑器，python 3.6版本；
- **注：**因为实验过程中要读取数据，因此，如果运行源代码，要修改文件的相应的路径。该实验的数据在处理过程中较大，可能会出现内存溢出的情况（memory out）。

3.实验方法

3.1 噪声信道模型

噪声信道模型（Noise Channel Model）是Shannon在上世纪40年代提出的，该模型在语音识别领域得到了广泛的应用，后来被应用在统计机器翻译中。IBM模型也是基于噪声信道模型。

在统计机器翻译中，对于源语言句子s，所有目标语词串 t 都是可能的译文，只是译文的可能性大小不同。即每对（s, t）都有一个概率值 P(t | s)来描述 s 翻译成 t 的好与坏。下图展示了机器的翻译候选空间。



机器的翻译候选空间

在噪声信道模型中，源语言s（信宿）被看作是由目标语言句子t(信源)经过一个由噪声的信道得到的。如果知道了s和信道的性质，可以通过 P(t | s) 得到信源的信息。对于汉译英的翻译任务，汉语句子s可以被看作是英语句子t加入噪声通过信道后得到的结果。于是需要根据观察到汉语的特征，通过概率 P(t | s)猜测最为可能的英语句子。这个找到最可能的目标语句（信源）的过程被称为解码。这个过程可以表述为：给定输入s，找到最可能的输出t，使得P(t | s)达到最大：

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

在IBM 模型中，可以使用贝叶斯准则对P(t | s)进行变换，得到的公式如下：

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

在该式中：

- $P(s|t)$是由译文t得到源语言句子s的翻译模型；
- $P(t)$是语言模型，表示目标语言句子t出现的可能性；
- $P(s)$是源语言句子s出现的可能性，因为s是输入的不变量，而且$P(s) > 0$，所以省略该项（分母），不影响最大值的求解。

于是翻译模型的目标被重新定义为：给定源语言句子s，寻找这样的译文t，它使得翻译模型P(t|s)和语言模型$P(t)$的乘积最大。

$$\text{Target} = \text{argmax}\ \{P(s|t)P(t)\}$$

由此，解决上述问题需要三个步骤：

- 建模：如何建立翻译模型P(t|s)和语言模型$P(t)$。
- 训练：如何获得P(t|s)和$P(t)$所需的参数。
- 解码：如何完成搜索最优解的过程，即完成argmax的过程。

3.2 基于词对齐的翻译模型

词对齐描述了源语言句子和目标语言句子之间单词级别的对应。具体是：给定源语言句子$\mathrm{S}=s_1^m\equiv s_1s_2\ldots s_m$和目标语言译文$\mathrm{T}=t_1\equiv t_1t_2\ldots t_l$。IBM模型假设词对齐通常满足：一个源语言单词只能对应一个目标语单词；源单词可以翻译为空。由于句子之间的对应关系可以由单词之间的对应进行表示，于是，句子翻译的概率可以被转化为词对齐生成的概率：

$$P(s|t)=\sum_a P(s,a|t) \qquad \text{式A}$$

其中，$P(s,a|t)$可以表示为：

$$P(s,a|t)=P(m|t)\prod_{j=1}^{m}P(a_j|a_1^{j-1},s_1^{j-1},m,t)P(s_j|a_1^{j-1},s_1^{j-1},m,t) \qquad \text{式B}$$

其中：

- s_j和a_j分别表示第j个源语言单词及第j个源语言单词对应到目标位置；
- s_1^{j-1}表示前（j-1）个源语言单词，a_1^{j-1}表示前(j-1)个源语言的词对齐；
- m表示源语句子的长度。

3.3 IBM module 1

IBM module 1对式B进行了化简：

- 假设 P(m|t)为常数 ε，即源语言的长度的生成概率服从均匀分布：P(m|t)= ε；
- 对齐概率$P(a_j|a_1^{j-1},s_1^{j-1},m,t)$仅依赖译文长度l，即每个词对齐连接的概率也服从均匀分布，即任何源语言位置j对齐到目标语言任何位置都是等概率的。即$P\left(a_j\middle|a_1^{j-1},s_1^{j-1},m,t\right)=\frac{1}{l+1}$.
- 源单词s_j的生成概率$P(s_j|a_1^{j-1},s_1^{j-1},m,t)$仅依赖与其对齐的译文单词$t_{a_j}$，即词汇翻译概率$\mathrm{f}(s_j|t_{a_j})$.

将上述三个假设代入公式A中，得到$P(s|t)$的表达式C：

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

$$=\sum_a \varepsilon\prod_{j=1}^{m}\frac{1}{l+1}\mathrm{f}(s_j|t_{a_j})=\sum_a\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}\mathrm{f}(s_j|t_{a_j}) \qquad \text{式C}$$

在该式C中，需要遍历所有的词对齐，即$\sum_a$.。可以进一步表示为：

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

式D分为两个部分：

1) 遍历所有的对齐a。其中a由$\{a_1\ldots a_m\}$组成，每个a_j从译文的开始位置（0）循环到截至位置（l）；

2) 对于每个a累加对齐的概率$P(s,a|t)=\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}\mathrm{f}(s_j|t_{a_j})$。

这样就得到了IBM模型1中句子翻译概率的计算方式。可以看出IBM module 1 的假设把翻译模型化简成了很简单的形式，对于给定的s, a, t，只要知道ε和$\mathrm{f}(s_j|t_{a_j})$就可以计算$P(s|t)$，从而求出$P(t|s)$ 。

3.4 IBM module 2

IBM module 1 虽然很好地化简了问题，但是由于使用了过强的假设，导致模型和实际情况有较大的差异，其中一个是假设词对齐的生成概率模型服从均匀分布，这会使模型忽略目标语言单词的位置信息。因此，当单词翻译相同但顺序不同时，翻译的概率一样。同时，由于源语言单词是错误位置的目标语单词生成的，不合理的对齐也会导致不合理的词汇翻译概率。

因此，IBM module 2抛弃了对$P(a_j|a_1^{j-1},s_1^{j-1},m,t)$服从均匀分布的假设，IBM module 2 认为词对齐具有倾向性的，它要与源语言单词的位置和目标语言单词的位置有关。具体是：对齐位置信息a_j的生成概率与位置j，源语句子长度m和译文长度l有关，表示为：

$$P\left(a_j\middle|a_1^{j-1},s_1^{j-1},m,t\right)=a(a_j|j,m,l)$$

IBM module 2的其他假设均与module 1相同，因此IBM module 2的数学公式为：

$$P(s|t)=\sum_{a}P(s,a|t)=\sum_{a_1=0}^{l}\cdots\sum_{a_m=0}^{l}\varepsilon\prod_{j=1}^{m}a(a_j|j,m,l)f(s_j|t_{a_j}) \qquad \text{式E}$$

类似module 1 ， module 2可以也可以看成两部分：

1) 遍历所有的a;

2) 对于每个a累加对齐的概率$P(s,a|t)$，即计算对齐概率$a(a_j|j,m,l)$和词汇翻译概率$f(s_j|t_{a_j})$对于所有源语言位置的乘积。

3.5 解码和计算优化

对计算IBM module 1 和 2的公式进行化简，得到一下公式：

$$IBM\ module\ 1: \qquad P(s|t)=\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i).$$

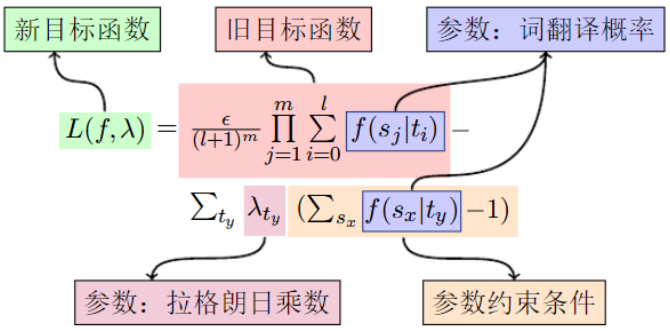
$$IBM\ module\ 2: \qquad P(s|t)=\varepsilon\prod_{j=1}^{m}\sum_{i=0}^{l}a(i|j,m,l)f(s_j|t_i)$$

有了上面的模型和化简后的结果，对其参数进行估计，然后可以将问题看成优化问题。其优化的目标函数是$P(s|t)$，以IBM module 1 为例，最终带约束的优化目标可以表示为：

$$\max\{\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\} \qquad \text{s.t. 任意单词}t_y:\sum_{s_x}f(s_x|t_y)=1$$

其中：$\sum_{s_x}f(s_x|t_y)=1$是优化的约束条件，以保证翻译概率归一化的要求。

通过上式，已将IBM module转化为了带约束的目标函数优化的问题，由于目标函数是可微分函数，解决这类问题的一种方法是把拉格朗日乘数法将带约束问题转化为不带约束问题，它的思想是把含有n个变量和m个约束条件的优化问题转化为含有（n+m）个变量的无约束优化问题。下图以IBM module 1 为例，表示了引入拉格朗日乘数的各项的意义：**（来源：机器翻译统计建模与深度学习方法）**

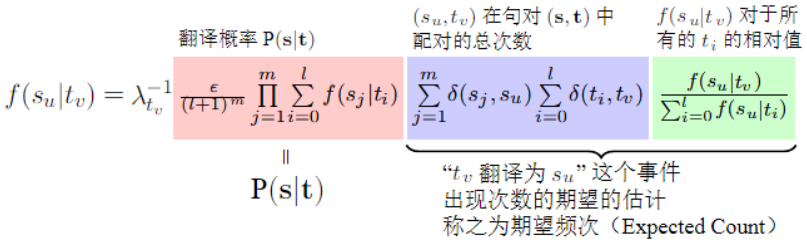


然后对该式进行求导，令导数为0，并化简等一系列运算，可以得到下式：

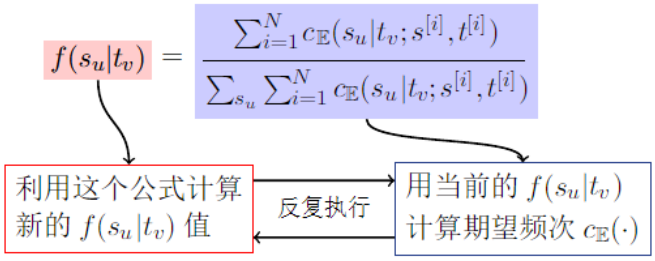
$$f(s_u|t_v)=\lambda_{t_v}^{-1}\frac{\varepsilon}{(l+1)^m}\prod_{j=1}^{m}\sum_{i=0}^{l}f(s_j|t_i)\sum_{j=1}^{m}\delta(s_j,s_u)\sum_{i=0}^{l}\delta(t_i,t_u)\frac{f(s_u|t_v)}{\sum_{i=0}^{l}f(s_u|t_i)}$$

可以看出，这不是计算$f(s_u|t_v)$的解析式，因为等式右边包含了$f(s_u|t_v)$，不过可以用**期望最大法（expectation maximization，EM）**的方法进行求解。使用EM方法可以利用上式迭代计算$f(s_u|t_v)$，使其最终收敛到最优解。

EM方法的思想是：用当前的参数，求似然函数的期望，之后最大化这个期望同时得到新的一组参数的值。 而在IBM 模型中，为了方便$f(s_u|t_v)$的计算，用某词语翻译出现的总次数代替一部分的累加和乘积运算。具体过程如下图所示：



最终得到IBM module 的迭代求解公式：



以IBM 模型 1为例，其伪代码如下图所示：

IBM 模型 1 的训练（EM 算法）

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

输出: 参数 $f(\cdot|\cdot)$ 的最优值

1: **Function** EM($\{(\mathbf{s}^{[1]},\mathbf{t}^{[1]}),\ldots,(\mathbf{s}^{[N]},\mathbf{t}^{[N]})\}$)

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 N **do**

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

6: **foreach** t_v appears at least one of $\{\mathbf{t}^{[1]},\ldots,\mathbf{t}^{[N]}\}$ **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 $\{\mathbf{s}^{[1]},\ldots,\mathbf{s}^{[N]}\}$ **do**

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

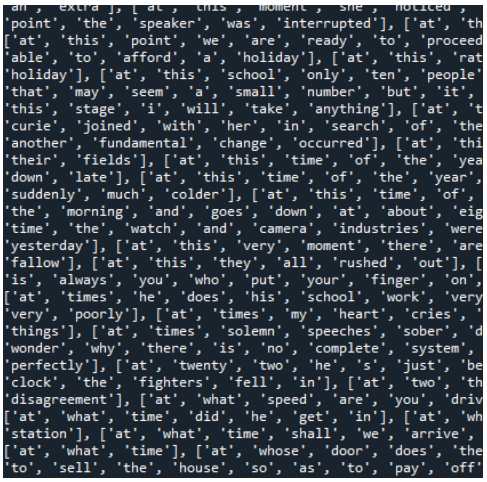
10: **return** $f(\cdot|\cdot)$

4.实验结果

4.1 实验过程

1) 数据预处理：

本实验需要从两个文件里提取数据，并对数据进行预处理。步骤：先从.txt文件中分别获取信息，然后通过正则表达式去除一些标点符号和非法的字符，如：“ó÷”，“ª”这种没有意义的字符。处理完成后，对语句进行切分，将得到的内容存入两个列表中。得到的结果如下：

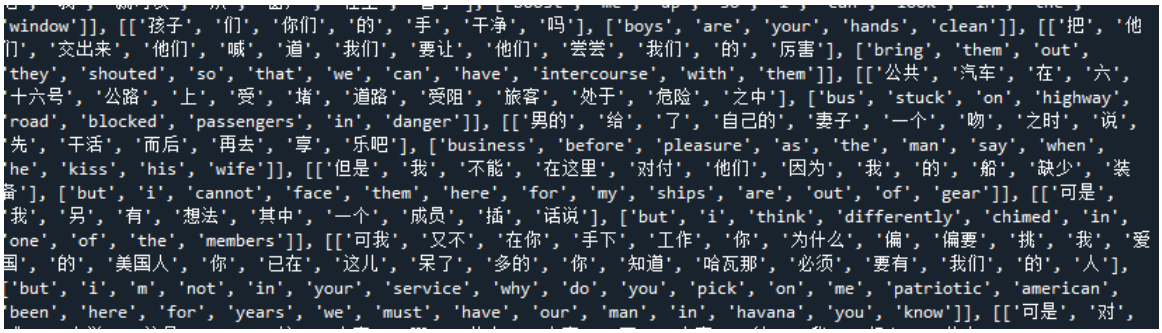


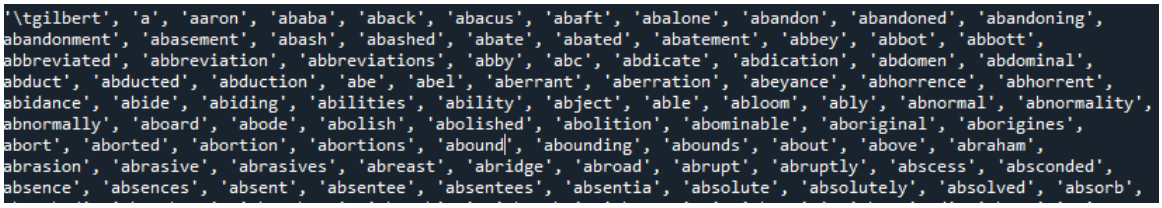
2) 组成语料库和词典：

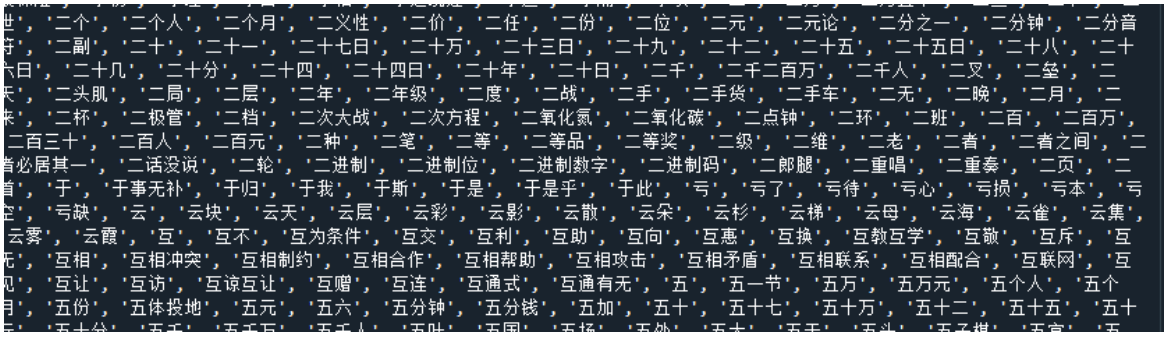
由上面的分词结果，得到语料库corpus，以及这个语料库中出现的中文和英文的词典，其中corpus的结构中，每一句的英文内容放在同一个list中，共有100000条数据，其中每个list包含如下内容：

中文	['科幻小说', '不能', '简单', '地', '看成是', '供', '消遣', '的', '而', '实际上', '它', '给', '读者', '展示', '更', '深刻', '的', '内容']
英文	['a', 'science', 'fiction', 'cannot', 'not', 'be', 'regarded', 'as', 'a', 'mere', 'entertainment', 'but', 'in', 'fact', 'it', 'tells', 'the', 'reader', 'much', 'more']

实现结果如下：（下图依次是语料库corpus，英文词典english_words，中文词典foreign_words的部分结果示意图）







3) 构建词对齐t：

t 是用来保存不同外文单词翻译成不同英文单词的概率。其核心代码如下：

```
def create_t(corpus,init_val):
    t = {}
    for sp in corpus:
        for f_w in sp[0]:
            for e_w in sp[1]:
                if f_w not in t:
                    t[f_w] = {}
                t[f_w][e_w] = init_val
    return t
```

通过打印输出，得到 t 共有4261308个。

```
序列对的总个数有：  4261308
```

4) 初始化参数并开始训练：

在参数初始化部分，主要是初始化训练的轮数epoch_num的值，然后开始进行训练，本次实验为了对比效果，用不同的轮数训练了多次。其中一次的过程图如下所示：

```
epoch :  0 ALL:  100000    NOW processing :  98000
epoch :  0 ALL:  100000    NOW processing :  99000
--------epoch 2--------
epoch :  1 ALL:  100000    NOW processing :  0
epoch :  1 ALL:  100000    NOW processing :  1000
epoch :  1 ALL:  100000    NOW processing :  2000
epoch :  1 ALL:  100000    NOW processing :  3000
epoch :  1 ALL:  100000    NOW processing :  4000
epoch :  1 ALL:  100000    NOW processing :  5000
epoch :  1 ALL:  100000    NOW processing :  6000
epoch :  1 ALL:  100000    NOW processing :  7000
epoch :  1 ALL:  100000    NOW processing :  8000
epoch :  1 ALL:  100000    NOW processing :  9000
epoch :  1 ALL:  100000    NOW processing :  10000
epoch :  1 ALL:  100000    NOW processing :  11000
epoch :  1 ALL:  100000    NOW processing :  12000
epoch :  1 ALL:  100000    NOW processing :  13000
epoch :  1 ALL:  100000    NOW processing :  14000
epoch :  1 ALL:  100000    NOW processing :  15000
epoch :  1 ALL:  100000    NOW processing :  16000
epoch :  1 ALL:  100000    NOW processing :  17000
epoch :  1 ALL:  100000    NOW processing :  18000
epoch :  1 ALL:  100000    NOW processing :  19000
epoch :  1 ALL:  100000    NOW processing :  20000
epoch :  1 ALL:  100000    NOW processing :  21000
epoch :  1 ALL:  100000    NOW processing :  22000
epoch :  1 ALL:  100000    NOW processing :  23000
```

4.2 实验结果

在实验过程中，为了更好地对比实验结果，在代码中将对词对齐概率的结果进行了降序排列，并将结果写入到.csv文件中，最终得到的结果如下图所示。

	A	B	C	D
1	汉语	英语	概率	
2	受领	accreditee	1	
3	西班牙文	adios	1	
4	字母数字的	alphanume	1	
5	采写	assignmen	1	
6	伯利兹	belize	1	
7	领头羊	bellwether	1	
8	出生地点	birthplace	1	
9	电脑化	cyberspace	1	
10	搬运费	cartage	1	
11	小箱	casket	1	

37	有销路	popular	1
38	打印输出	printout	1
39	招股	prospectus	1
40	翻两番	quadruple	1
41	险别	risk	1
42	共享软件	shareware	1
43	花絮	sidebar	1
44	附加费	surcharge	1
45	制表机	tabulator	1
46	皮重	tare	1
47	三个	three	0.929443
48	土壤	soil	0.918559
49	水泥	cement	0.912625
50	英语	english	0.9073
51	现代	modern	0.896114
52	两	two	0.891141
53	乔治	george	0.889408
54	玻璃	glass	0.888017
55	新	new	0.887645
56	小姐	miss	0.886876
57	杰克	jack	0.885751
58	她那	her	0.885629
59	旧	old	0.88544
60	黑色	black	0.882724
61	或者	or	0.88229
62	比尔	bill	0.87792
63	服务器	server	0.87725
64	能量	energy	0.876527
65	头发	hair	0.876416
66	棉花	cotton	0.874191
67	她的	her	0.873476
68	红色	red	0.873398
69	白色	white	0.872832
70	红	red	0.871632
71	牛肉	beef	0.870173
72	三	three	0.86883
73	史密斯	smith	0.86722

56750	大员	newsreels	0.045648
56751	头戴	newsreels	0.045648
56752	令人讨厌	dreadful	0.045178
56753	千亿	tripling	0.044905
56754	增加三倍	tripling	0.044905
56755	愿闻其详	tripling	0.044905
56756	猖狂	ills	0.04469
56757	施压	hurd	0.04298
56758	检方	inconsister	0.042566
56759	十尺	bangs	0.035562
56760	侧门	bangs	0.035562
56761	未了	bangs	0.035562
56762	后墙	bangs	0.035562
56763	报亭	bangs	0.035562
56764			

有结果可知，共有（56763-1）个汉语词语，并将其对应的英语概率最大的单词的挑选出来了，以及输出了相应的概率。此结果可以看出，概率排名在前面单词翻译的效果不错，基本满足要求。最后的汉语词语翻译的结果还有待改进。

然后，提高轮数，观察对于相同的单词翻译的是否相同，如果相同，它的概率会不会随着轮数的增加而上升。

中文	英文	Epoch = 5	Epoch = 15
小箱	casket	1	1
三个	three	0.929443	0.983434
现代	modern	0.896114	0.969343
那么些	such	0.163423	0.163773

除此之外，5轮迭代的实验，准确率在0.8以上的词队有：191组，15轮迭代的实验，准确率在0.8以上的词队有：344组。如下图所示（左图迭代5轮，右图迭代15轮）。

180	黄色	yellow	0.803589
181	下午	afternoon	0.803511
182	政府	governme	0.80311
183	网络	network	0.802837
184	学校	school	0.802076
185	和平	peace	0.801533
186	生意	business	0.801036
187	常常	often	0.800951
188	其他	other	0.8007
189	哦	oh	0.800687
190	简	jane	0.800323
191	冬季	winter	0.800244
192	北京	beijing	0.800019
193	母亲	mother	0.799307
194	战争	war	0.799195
195	面包	bread	0.799126
196	会计部	accountinc	0.798944

336	球队	team	0.804047
337	光	light	0.804001
338	性虐待	abuse	0.803709
339	右	right	0.803387
340	理论	theory	0.803314
341	句子	sentence	0.802548
342	摄像机	camera	0.802005
343	罐装	canned	0.801845
344	世界	world	0.800242
345	按揭	mortgage	0.800052
346	事故	accident	0.799895
347	早上	morning	0.799695
348	通常	usually	0.799625
349	水果	fruit	0.799474
350	野战	field	0.798908
351	棕色	brown	0.798661
352	之前	before	0.79859
353	明天	tomorrow	0.798125
354	总是	always	0.798098

由此可见，迭代的轮数对实验的准确率有着重要的影响，轮数的增加可以提高

汉语翻译为英语的概率的提高。

经过第三章中的原理， IBM module 2抛弃了module 1 对$P(a_j|a_1^{j-1},s_1^{j-1},m,t)$服从均匀分布的假设，IBM module 2 认为词对齐具有倾向性的，它要与源语言单词的位置和目标语言单词的位置有关。具体是：对齐位置信息a_j的生成概率与位置j，源语句子长度m和译文长度l有关，表示为：

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

IBM module 2 在实验过程上和IBM module 1 的过程相同，训练模型，并将实验结果保存在csv文件中，如上一样，对其结果进行分析：得到在训练相同轮数的情况下，使用module 2 训练得到的概率大于0.8的词对数目多于使用module 1得到的结果。如下图所示：（作图为module 1 的结果，右图为module 2的结果）。

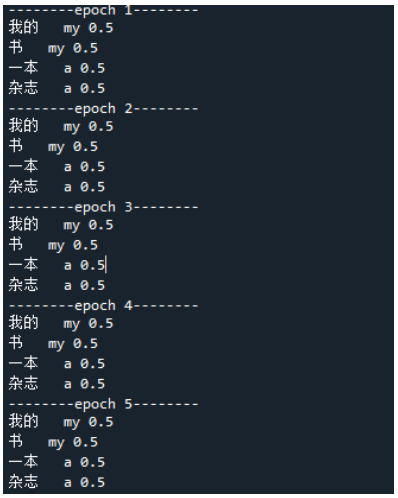
335	妹妹	sister	0.80453
336	球队	team	0.804047
337	光	light	0.804001
338	性虐待	abuse	0.803709
339	右	right	0.803387
340	理论	theory	0.803314
341	句子	sentence	0.802548
342	摄像机	camera	0.802005
343	罐装	canned	0.801845
344	世界	world	0.800242
345	按揭	mortgage	0.800052
346	事故	accident	0.799895
347	早上	morning	0.799695
348	通常	usually	0.799625
349	水果	fruit	0.799474
350	野战	field	0.798908
351	棕色	brown	0.798661
352	之前	before	0.79859
353	明天	tomorrow	0.798125
354	总是	always	0.798098

353	出厂	vintage	0.802553
354	球队	team	0.802421
355	算术	arithmetic	0.802416
356	聚会	party	0.802399
357	尤其	especially	0.801618
358	进去	in	0.801604
359	字典	dictionary	0.801529
360	这座	the	0.801457
361	暴力	violence	0.80056
362	李斯特	liszt	0.800546
363	王子	prince	0.799829
364	请	please	0.799318
365	宣布独立	independe	0.79905
366	婴儿	baby	0.798972
367	世界	world	0.798898
368	宾夕法尼亚	pennsylvar	0.798702
369	之前	before	0.798422
370	明天	tomorrow	0.798391
371	按揭	mortgage	0.798243

5.遇到问题及解决思路

1） **问题：迭代了多轮，但参数更新不动；**

- **问题描述：**在测试代码的正确性时遇到了问题：参数迭代了好几轮，但是概率值一直没有更新；如下面所示：



- **原因：**其实代码的主题功能没问题，而是语料库的问题，此时的语料库是：

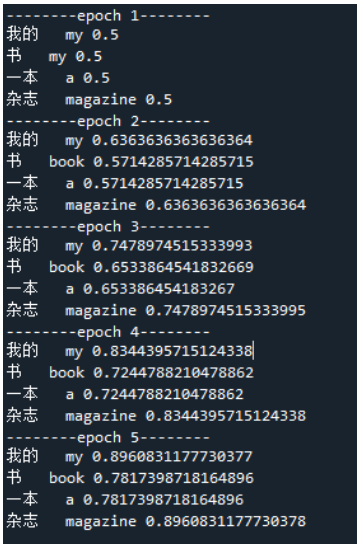
  corpus = [[['我的','书'],['my','book']],[['一本', '杂志'],['a', 'magazine']]]

  当语料库中的句子完全不相关时，就没办法就行参数的迭代和更新了。

- **解决方法：**在预料库中添加相关的语句，如以这个为例，把语料库扩充为：

  corpus = [[['我的','书'],['my','book']],[['一本', '杂志'],['a', 'magazine']],[['我的', '杂志'],['my', 'magazine']]]

  当迭代五轮后，得到的结果如下：可见取得了较好的结果。



2） 中文乱码

- **问题描述：**将结果写入到csv文件中时，中文字符出现了乱码的情况；
- **原因：**编码参数出错了，encoding = “utf-8-sig”
- **解决方案：**encoding = “utf-8-sig”，其中，sig是signature，即带有签名的utf-8，可以解决中文编码的错误。结果如下：

	A	B	C
1	汉语	英语	概率
2	杂志	magazine	0.896083
3	我的	my	0.896083
4	书	book	0.78174
5	一本	a	0.78174
6			

3) 提升概率的方法

- 问题描述：有些单词的概率较低和单词翻译出错；
- 原因：部分原因是数据预处理的问题，将一些没有实际意义的非法字符处理为一个单词了，虽然这部分字符比较少，但非法字符的存在仍会影响词对齐概率的结果。
- 解决方案：在数据的预处理部分，将非法字符：如“ó÷”，“ª ”这种字符去掉。

4) Module 2 中a的理解

- **问题描述**：module 2 中将均匀分布的概率换成了一个变量a，但对a的理解不好；
- **解决方法**：通过查找资料，查看了module 2 的伪代码，对a的理解进一步加深，其中，module 2 的伪代码如下：

Input:A training corpus $(f^k, e^k)\ and\ 1 \le k \le S$
Initialization:Initialize t and a parameters(e.g. to random values)
Algorithm:
Do{
-set all counts to 0;
-traverse all of the sentence pair
*for j = 1...m
~for i=0...l
c(fj|ei;f,e) += α(k,l,j)
c(ei) += α(k,l,j)
c(i|j,l,m) += α(k,l,j)
c(j,l,m) += α(k,l,j)

where $\alpha(k,l,j) = \frac{t(f_j|e_i)a(i|j,m,l)}{\sum_{k=0}^{l} t(f_j|e_k)a(k|j,m,l)}$

-set t(f|e)=c(f|e;f,e)/c(e) a(i|j,m,l)=c(i|j,m,l)/c(j,m,l)
}until t & a converge
Output: parameters t(f|e) a(i|j,m,l)