LAB2 HMM & CRF

陈昕婕 15302010057 软件工程

一、引言

本次 LAB 我使用 JAVA 语言，分别实现了基于 HMM 和基于 CRF 模型的中文分词任务。根据需求把句子中的每个中文汉字和标点分为 B—词首，E—词尾，I—中间词，S—单字词这四类。对于 HMM，先基于已标注的样本进行监督学习，获得一个基础的模型，再在这个模型上用未标注样本集采用 EM 算法进行无监督学习，通过维特比算法解码得到预测序列。对于 CRF，基于论文《2002 Discriminative training methods for hidden Markov models : Theory and experiments with perceptron algorithms》提到的算法，训练时根据特征函数权重，通过维特比算法得到预测序列，与标准序列进行对比修改特征函数权重，以达到满意的中文分词效果。

二、对 HMM 的基础理解以及数据结构的设计

HMM 的实现大概分为三个步骤：监督学习，EM 算法训练，维特比解码。

1. 监督学习

监督学习就是在样本有标记的情况下，对样本进行统计，计算出 pi（初始概率），A（转移概率），B（放射概率）。公式如下：

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}}, \qquad i = 1, 2, \cdots, N \; ; \; j = 1, 2, \cdots, N$$

转移概率：

$$\hat{b}_j(k) = \frac{B_{jk}}{\sum_{k=1}^{M} B_{jk}}, \qquad j = 1, 2, \cdots, N \; ; \; k = 1, 2, \cdots, M$$

放射概率：

初始概率即统计每句话开头 BIES 出现的次数 / 话的句数。

总结来说，监督学习就是遍历训练集，计算出每个状态间的转移频率、每句话开头的频率以及对应状态下各个汉字出现的频率，分别对应 A，pi，B 三组概率值。

2. 无监督——EM 算法训练

EM 算法又称前向后向算法。是无监督训练。无监督和有监督最大的差别就在于，无监督训练不需要标注，只要有语料集就好了。EM 算法主要一下步骤：

1. 前向算出语句每个字时是每个状态的前向概率。

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \le t \le T-1, 1 \le j \le N.$$

2. 后向算出语句每个字的后向概率。

$$\beta_T(i) = 1, \qquad 1 \le i \le N$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \qquad 1 \le t \le T-1, 1 \le j \le N.$$

3. 求出 t 时刻在观察量下属于状态 i 的概率γ

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

4. 求出 t 时刻状态为 i，t+1 时刻状态为 j 的联合概率ξ

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

5. 计算 pi，A，B

$$\bar{\pi}_i = \gamma_1(i);$$

$$\bar{a}_{ij} = \sum_{t=1}^{T-1} \xi_t(i,j) / \sum_{t=1}^{T-1} \gamma_t(i);$$

$$\bar{b}_j(k) = \sum_{t=1, O_t = v_k}^{T} \gamma_t(j) / \sum_{t=1}^{T} \gamma_t(j).$$

EM 算法的原理是根据模型已有参数求出确定完全数据的对数似然函数，pi,a,b 刚好分在对数似然函数的多项式里。对对数似然函数求导使得其最大化，即得到三组参数的修改公式。

对数似然函数如下：

$$Q(\lambda, \bar{\lambda}) = \sum_{I} \log \pi_{i_1} P(O, I \mid \bar{\lambda}) + \sum_{I} \left(\sum_{t=1}^{T-1} \log a_{i_t i_{t+1}} \right) P(O, I \mid \bar{\lambda}) + \sum_{I} \left(\sum_{t=1}^{T} \log b_{i_t}(o_t) \right) P(O, I \mid \bar{\lambda})$$

3. 维特比解码

维特比解码即递归的算出在每个时刻每个状态可能出现的最大概率。根据马尔科夫性，每个状态只与它前面一个状态有关。

前向过程：在计算过程中根据转移矩阵 A，分别算出当前时刻当前状态可能来源于前一个时刻的每种状态的概率，保留最大的概率并算出最大概率对应的前一种状态。以此递归到最后时刻。

后向过程：对比最后时刻的不同状态的概率，取最大的作为最后时刻的标记。然后依次回溯找出之前每个时刻最大概率的状态值。

在本次中文分词的任务中，每个时刻对应一句话的每个汉字，比如“维特比解码”这句话包含五个时刻。状态有 4 种状态，分别是 EBIS。放射概率则是对应在每个状态下每个汉字出现的概率。

对于 HMM 各个状态的数据结构，我大概是如此设计的：

Pi: double[] pi = new double[4]; // 四个 double 数据分别存储 B,E,I,S 的初始概率

A：double[][] A = new double[4][4]; //二维数组，分别存的状态 i 和 j，代表 i 到 j 的

转移概率

B：double[][] B = new double[4][character_number];//二维数组，第二位代表训练集中出现的汉字总数

训练集和测试集分别是以下方式存储：

ArrayList<String> sentences = new ArrayList<>(); //存着测试集或训练集的句子，一句话作为一个元素

ArrayList<String> marks = new ArrayList<>(); //存着测试集或训练集的标记序列，一句话作为一个元素

HashMap<Character, Integer> character_match = new HashMap<>(); //储存着汉字和整数的对应，即可通过汉字寻找到对应下标

维特比训练时：

double[][][] p = new double[length][4][2];//储存每个时刻的最大概率和对应的前一个状态。最后一维为 0 是代表概率，为 1 时代表前一个的状态（分别对应数字）

无监督学习时：

double[][] forwardA = new double[length][4];//储存着每时刻对应的前向概率

double[][] backwardB = new double[length][4];//储存着每时刻对应的后向概率

double[][] y = new double[length-1][4];//对应着 t 时刻状态为γ的概率

ArrayList<double> scale = new ArrayList<>();//对应着每个时刻的比例因子，进行归一修正

由于为了准确率我将句子加长处理，即多句话甚至全集一起统计计算概率，所以对全集中每句话的前向后向中间量进行存储。

//多维数组的最后一维中 0 均为每句话对应的该概率值的总值（分母），1 对应的该状态的概率值（分子）。在遍历完所有句子后进行分子分母分别相加再对除即可得到修正后的概率值。

double[] process_pi = new double[4];

double[][][] process_A = new double[4][4][2];

double[][][] process_B = new double[4][character_number][2];

三、对 CRF 的理解以及数据结构设计

根据论文提供的简易 CRF 实现方法，整个 CRF 模型构建和训练可以分为以下步骤：

1. 扒出特征函数

根据特征模板，由于我的特征函数以及权重值是按照 Hashmap 存储的，在训练前先遍历全集，将在训练集中出现过的汉字、汉字组合或者汉字和状态组合加入 hashmap 数组中。并初始化每个特征函数的权重为 0。

2. 维特比训练

根据特征函数以及它们对应的权重，按照维特比的算法给每个序列打分，最终求的概率最大的序列作为训练序列（预测结果）输出。

在计算每个序列分数的时候，对于每个时刻每个状态，循环一遍所有的特征函数，如果符合则特征函数取 1，不符合特征函数则取 0，对于每个状态加上特征函数的值*权重。由于特征函数的值用的 1 和 0，所以过程简化为找出汉字对应的特征函数，取出权重直接加上就好。

3. 修改权重

维特比训练后获得训练序列，与已经标记过的标准序列对比，对于不一致的时刻状态，将此时刻错误状态对应的特征函数权重-1，正确状态对应的特征函数权重+1。如此不停的遍历训练集进行训练。

4. 预测

根据 2 中的维特比训练方法，得出的队列就是预测序列。

由于特征函数二值化，所以只需存储权重。我主要采用了 hashmap 的存储，用了 unigram 和 bigram 两种形式。

对于 unigram，只需考虑汉字组合和当前状态。存储的数据结构如下：

//unigramNumber 即对应的特征函数种类数，由于考虑开头无前一个状态，所以状态数定位 state+1

HashMap<String, Integer>[][] unigramModel = new HashMap[unigramNumber][state+1];

对于 bigram，除了当前的状态，还要考虑之前一个状态的转移。存储的数据结构如下：

//bigramNumber 即对应的特征函数种类数，由于考虑开头无前一个状态，所以状态数定位 state+1

HashMap<String, Integer>[][][] bigramModel = new HashMap[bigramNumber][state+1][state+1];

四、CRF 的模板训练对比

CRF 的核心是根据模板函数的权重来调整每个序列的得分。故模板的选取非常重要。

根据中文分词的特点，中文有单字词，比如“我”；也有两个字的词，相互之间有关系，比如“我们”；也存在三个字的词，比如“幼儿园”。考虑不同的汉字组合，设为特征函数，使得训练的时候也可以学习到组合字的概率。

理想来看，使用的模板越多，则考虑到的汉字组合越多。根据中文语法可以标注的更加准确。但是按照每句话每个时刻都取组合，可能会出现把不是一个词的组合考虑进去的情况。故也是会有一点干扰的。所以模板的设置非常玄学巧妙，也许不是越多越好。

所以我根据已经实现的 CRF 模型采用不同的模板，训练 train_corpus.utf8 文件，测试 train.utf8 文件，来进行对比。在实验中每组都采用了单一变量法，即保证了每个实验除对比项的模板不一样，其他设置都是一样的。由于算法一样，多次实验结果也是一样的。列出实验数据和比较结果如下：

先列出采用的模板：（以下只写标号代表）：

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-2,0]/%x[-1,0]
U06:%x[-1,0]/%x[0,0]
U07:%x[-1,0]/%x[1,0]
U08:%x[0,0]/%x[1,0]
U09:%x[1,0]/%x[2,0]
U10:%x[-2,0]/%x[-1,0]/%x[0,0]
U11:%x[-1,0]/%x[0,0]/%x[1,0]
U12:%x[0,0]/%x[1,0]/%x[2,0]
```

```
# Bigram
B00:%x[-2,0]
B01:%x[-1,0]
B02:%x[0,0]
B03:%x[1,0]
B04:%x[2,0]
B05:%x[-2,0]/%x[-1,0]
B06:%x[-1,0]/%x[0,0]
B07:%x[-1,0]/%x[1,0]
B08:%x[0,0]/%x[1,0]
B09:%x[1,0]/%x[2,0]
B10:%x[-2,0]/%x[-1,0]/%x[0,0]
B11:%x[-1,0]/%x[0,0]/%x[1,0]
B12:%x[0,0]/%x[1,0]/%x[2,0]
```

1. 只考虑 Unigram vs Unigram+Bigram（特征函数的字组合选取保持一致）

step	Unigram	Unigram+Bigram
0	88.14%	88.99%
1	89.67%	90.13%
2	89.98%	90.40%
3	90.41%	90.93%
4	90.72%	91.22%
5	90.87%	91.29%
6	90.92%	91.46%
7	90.74%	91.44%
8	91.10%	91.81%
9	91.04%	91.75%
10	91.20%	91.56%
15	91.13%	91.64%
20	90.93%	91.59%
30次MAX	91.38%	91.94%

可以看出，用了 Bigram 的总体正确率比只用 Unigram 略高。并且最后相对稳定的区间内用了 Bigram 的比没用的正确率多了大概 0.5%。因为 Bigram 多考虑了状态和状态之间的转移使得结果更加精确。

但是其实差别并没有想象中的大，我猜测是因为在字的组合中考虑了双字词等的组合，包含了对于前一时刻影响后一时刻的可能。加上 Bigram 即在已经很精确的情况下再次精确。

2. 只考虑单字词（00-05） vs 考虑单字词+两个字组合（00-10）（均采用 Unigram+Bigram 的模式）

step	UB00-05(单字)	UB00-10（单双字）
0	80.83%	88.89%
1	81.01%	90.25%
2	81.15%	90.31%
3	81.14%	90.15%
4	82.15%	90.51%
5	81.56%	90.76%
6	81.71%	91.06%
7	81.97%	91.00%
8	81.70%	91.01%
9	82.16%	91.03%
10	81.55%	90.74%
15	81.06%	91.37%
20	81.22%	91.55%
30次MAX	82.16%	91.55%

这组对比差距很明显，可以看出不考虑两个字的组合和考虑的相差非常大。只考虑单字词基本稳定在 81.5%左右,考虑后稳定在 90.0%左右。这和中文的语法关系很大。在我们日常说话中的词汇也是二字词比单字词多，所以不考虑前一个汉字的影响也是不符合中国用语习惯的。

3. 单字词+两个字（00-10） vs 单字词+双字词+三字词组合（00-12）（均采用 Unigram+Bigram 模式）

step	单双字（00-10）	单双三字（00-12）
0	88.89%	88.99%
1	90.25%	90.13%
2	90.31%	90.40%
3	90.15%	90.93%
4	90.51%	91.22%
5	90.76%	91.29%
6	91.06%	91.46%
7	91.00%	91.44%
8	91.01%	91.81%
9	91.03%	91.75%
10	90.74%	91.56%
15	91.37%	91.64%
20	91.55%	91.59%
30次MAX	91.55%	91.94%

可以看出考虑了三字还是比没考虑要好那么一点点，虽然也是很微妙的一点点。这也符合中文三字词的习惯以及模板多精确度就相比可能更高的想法。

4. 在单字词+双字词+三字词+Ungiram+Bigram 模式下的不同组合

在均使用了单双三字词的情况下，对模板函数进行筛选，比较少一些特征函数会不会有

很大影响。

A 组：单双三字特征函数全部（00-12）

B 组：00-06，08，09，10，11

C 组：00-04，06，07，08，10，12

step	A	B	C
0	88.99%	88.69%	88.40%
1	90.13%	89.32%	89.16%
2	90.40%	90.08%	90.36%
3	90.93%	90.61%	90.27%
4	91.22%	90.90%	91.03%
5	91.29%	91.16%	91.26%
6	91.46%	90.85%	90.65%
7	91.44%	91.16%	90.88%
8	91.81%	91.37%	91.02%
9	91.75%	91.36%	91.36%
10	91.56%	91.36%	91.20%
15	91.64%	91.40%	91.72%
20	91.59%	91.55%	91.25%
30次MAX	91.94%	91.69%	91.72%

虽然考虑少一点的特征函数可以让程序跑的更快一点，但是实验数据可以看出，BC 虽然正确率都很高，但是还是不如 A 稳定的区域高。果真还是老老实实的用所有模板吧。

总结来看，对比了这么多的模板。最理想的使用方式就是：Unigram+Bigram，考虑单字二字三字情况，并且使用尽可能多的模板数。

五、HMM 中 EM 算法的使用与否对比

（以下数据均是使用 HMM 模型训练 train_corpus.utf8 测试 train.utf8 得到的数据）

HMM 在监督学习下已经得出了一个模板，根据此模板测得测试集的准确率为 78.8879%

采用 EM 算法后多次迭代得到的准确率和以及每次迭代的准确率变化量如下：

step	accuracy	△accuracy
0	77.50%	
1	75.74%	1.76%
2	74.16%	1.58%
3	73.67%	0.51%
4	72.67%	1%
5	71.54%	1.13%
6	70.98%	0.56%
7	70.52%	0.44%
8	70.12%	0.30%
9	69.87%	0.25%
10	69.81%	0.06%

经过对比可以看出，EM 算法每次迭代后准确率的变化量呈下降趋势，说明 EM 算法再往某个标准收敛。但是比起监督学习，EM 算法的准确率逐渐下降。可以判断 EM 算法是收敛的但是不符合中文分词学习的要求，将参数往模型不想要的方向收敛。

分析原因如下：

EM 算法缺失了隐含数据，即每个时刻的状态，只能靠极大似然估计求解隐含状态。EM 算法能够收敛的核心就是在每次修正后的极大似然极值优于上一次的值。

随机初始化 θ_0。

1、求条件期望 $F(\theta, \theta_n)$，如上公共所示；

2、求 $F(\theta, \theta_n)$ 的极值处 θ_{n+1}。

3、反复迭代1，2计算，直到 θ_n 收敛，即 $|\theta_{n+1}-\theta_n|<\alpha$（收敛条件）。

这样必然会存在一个问题，由于 EM 算法无法知道最佳状态，只能在两次间对比，这很容易让 EM 算法遇到一个局部最优解，便朝局部最优解学习，导致最终陷入局部最优解，和全局最优解形成偏差还无法自己修正。

所以对于 HMM 模型，在中文分词任务中存在已有标记样本集的情况下，不采用 EM 算法的效果更好。由于标记样本需要大量的人力，所以 EM 算法的实用性体现在无人力标记样本的情况。

六、HMM 和 CRF 的对比

CRF 中用到的最明显的和 HMM 不一样的就在于，CRF 多使用了特征模板。其他的维特比算法解码的过程都是一致的。

特征模板主要标记的是汉字和汉字间的关系，这就对应了马尔科夫模型中的结果和结果之间的联系。而 HMM 没有使用这些模板函数，所以 HMM 只考虑了隐藏状态之间的联系并没有考虑结果的联系。HMM 的使用就是马尔科夫性，他本身就是基于很简单的互相依赖关系，对于复杂的中文分词任务就不太适合。

而 CRF 都是监督学习，提供了观察序列和隐藏序列进行学习，可以兼顾到全局的情况而不至于选入局部最优解。CRF 对整个序列的联合概率有统一的模型，能够根据上下文信息给出更好的实验结果。哪怕当前状态判断错误，由于拥有上下文的特征函数，也可以对当前状态的值就行修正。模板又是根据中文分词的特点对应的，可以随意调整和融合新的特征，更适合中文分词任务。

总结来看，CRF 拥有更强的推理能力，能够使用复杂，有重叠性，非独立的特征进行训练和推理，充分利用上下文特征，还可以随意添加外部特征，使得模型获取的信息丰富，比起 HMM 来看准确率更优秀。

七、 实验中遇到的问题——HMM 中 EM 算法的 NaN 问题

探究 NaN 产生的原因以及解决办法，分别有以下：

1. 训练过程中值的极端化

虽说 EM 算法的准确率不高，但它最起码还是要收敛的，所以在内部数据的拟合也是越来越优秀，可能出现的值越来越大，不可能出现的值越来越低。再由于前向后向的累乘，很容易出现浮点数下溢即产生 NaN 问题。

这时候询问助教后采取添加比例因子的方式来修正这个问题。比例因子当然不是随意的规定一个常数然后每个都乘上去，而是对前向每个状态的概率α进行归一化，将α乘以一个相同的系数 Ct，使得四个状态的α和为 1。为了平衡其他参数，对于后向概率β也乘以相同的比例因子，并注意在算单个概率和联合概率时的系数抵消问题。

值得注意的是在前向后向的每一步都需要乘以当前的比例因子，所以实际上每个时刻的比例因子是累乘的。

算法具体参照助教后来提供的 PDF，不详细列举。

2. 训练过程中放射概率为 0 的问题

一句一句话训练的话由于包含的汉字太少，有些汉字没有出现过，导致这些汉字在权重修改的过程中值被修改为 0。如此迭代由于路径出现的概率太低在优化比例因子的时候比例因子有可能值为 0，最后相除也是会 NaN 的。

针对这个问题的方法有两个：1. 加长句子，最暴力的做法就是全集扔进去训练，一起计算概率，遍历一次全集修改一次参数。2. 在修改参数的时候对于放射概率，只修改出现字的放射概率。

本次 Lab 我选取了暴力的方法。

3. 路径出现的可能性太低的问题

对于路径出现的可能性低至 0 或者在跑测试集时遇到训练集里没出现的字都可以用同样的方法解决。由于乘以 0 会报错，可以拿个非 0 的极小值来替代。在本次实现 HMM 模型中我采用了 1/character_number 来替代。而后期修改模型参数的时候，有些字对应出现的概率为 0 则保持不修改。

综合以上 3 点，NaN 的问题得以解决。

八、参考资料

1. EM 算法详解
   http://www.cnblogs.com/Leo_wl/p/3200173.html
2. 《2002 Discriminative training methods for hidden Markov models : Theory and experiments with perceptron algorithms》
3. 《2001 Conditional random fields_probabilistic models for segmenting and labeling sequence data.pdf》
4. 《shen_tutorial : Some Mathematics for HMM》
5. HMM 和 CRF 的区别
   blog.csdn.net/losteng/article/details/51037927