基于 HMM 和 CRF 的分词器实现

王傲

15300240004

信息内容安全

COMP130108.01

指导教师：曾剑平

目录：

内容摘要：

通过训练隐马尔科夫模型（HMM）进行序列标注，获得中文分词的模型，使用 Viterbi 算法对输入的句子进行分词；为了进行对比，使用 Python 中 sklearn-crfsuite 库提供的条件随机场（CRF）模型训练并进行分词。使用 SIGHAN 2005 Bakeoff 提供的数据和脚本进行测试。为了方便演示和使用，还提供了 HMM 和 CRF 分词器的带图形界面的实现。

关键词：HMM CRF 分词 准确率 召回率 图形界面

1. 简介

1.1 内容简介

现在比较常见的分词器中，有很多是利用 HMM 实现的。HMM 可以用于序列标注，实现起来比较简单，并且在实际使用中有很好的表现。因此，这里使用 HMM 实现一个中文分词器。CRF 同样可以用于序列标注，并且在实际中的表现经常超越 HMM，所以这里同样实现了 CRF 中文分词器，与 HMM 分词器进行对比。为了保证公平性和准确性，使用 SIGHAN 2005 Bakeoff 提供的中文数据和评分脚本对两个分词器的分词结果进行评测，并进行了相应的统计。最后，为了方便用户使用，提供了基于 PyQt 实现的带有 GUI 的两个分词器模型。

1.2 配置和环境

编码和测试环境为 Macbook Pro 15’ Mid 2015,操作系统为 OS X High Sierra 10.13.1（训练 CRF 时由于占用内存过大，在服务器上训练完成）。全部中文文本使用 UTF8 编码。主要基于 Python 2.7.14 实现，使用的库包括(使用 pip 安装，十分方便；PyQt 在 Mac 上使用 brew 安装)：

- pickle（用于长期存储 Python 对象）
- sklearn-crfsuite 0.3.6
- PyQt5 5.9.2 及相关依赖库
- sklearn 0.19.1（使用 sklearn.externals.joblib 模块，用于保存和导入训练好的 CRF 模型）

这里特别说明，由于提供的训练语料 pku_training.txt 中语句的数量相对来说不是很多，因此使用自己搜集的高质量新闻语句+中文 wiki 材料作为语料来源，使用哈工大语言云的 Python 实现 pyltp 0.1.9.1, 训练模型版本 3.3.1 中分词器的分词结果作为训练语料。

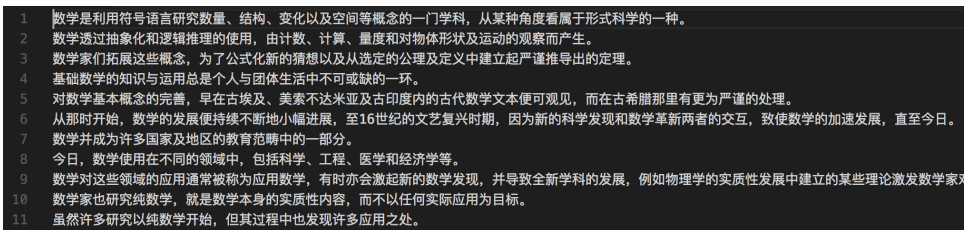

图 1 使用的训练语料

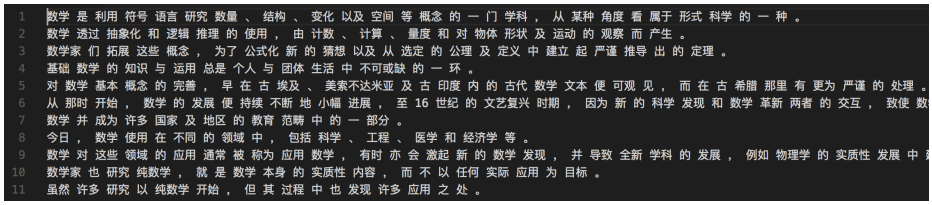

图 2 pyltp 的分词结果（训练语料）

实践证明，pyltp 的分词准确率很高,将分词结果作为训练语料完全可行。由于训练语料体积过大（接近 2G），没有在附件中提供，仅提供了 HMM 和 CRF 训练好的结果。

2. HMM 分词器

2.1 基本原理

HMM 由一个不可见的马尔科夫链和可见的观测序列组成。一个 HMM 由 {所有可能的（不可见）状态的集合 Q，所有可能的观测集合 V，初始状态概率向量 π，状态概率转移矩阵 A，观测概率矩阵 B } 五元组定义，其中(π, A, B)称为 HMM 的三元组，也是我们要通过训练获得的数据。在中文分词中，状态有 4 个，为 B、M、E、S,分别表示词首、词中、词尾、单字。观测集合就是训练语料中的每个字。

HMM 可用于序列标注。在分词中，就是为句子中每个字标注 B、M、E、S 状态中的一个，然后根据状态分词。训练 HMM 的过程，就是对训练语料进行统计。根据大数定律，当语料足够大时，将频率作为概率，获得初始状态概率向量，状态概率转移矩阵，观测概率矩阵三个量，作为 HMM 的参数，参与分句。这里使用的是 Uni-gram 的 HMM，只考虑每个字和相邻的字之间的关系。

获得 HMM 的参数后，就可以利用这些参数进行分词。这里分词对应的是 HMM 的第三个基本问题，即已知模型 λ 和观测序列 O，求对给定观测序列的条件概率 $P(I \mid O)$最大的状态序列 I。对应这个问题，在这个分词器中使用了 Viterbi 算法。Viterbi 算法利用动态规划求解概率最大路径（即最优路径），比蛮力算法的复杂度要小很多。使用 Viterbi 算法求解时，每步要记录这条路径当前最优的节点（即要选择的状态），获得最终结果后回溯，获得整条路径，即隐含的状态序列。

2.2 代码实现

HMM 的训练和分词算法在 HMM.py 中实现。

训练功能通过 HMM_train 函数实现（均视为符合大数定律），统计 B、M、E、S 出现在句子开头的次数，除以总数获得初始状态概率向量；记录每个句子中两个状态相邻的次数，除以总数算出条件概率，获得状态概率转移矩阵；记录每一个状态到一个单字的次数，除以这个状态出现的总数，获得观测概率矩阵（发射矩阵）。训练 HMM 的过程就是获得这三个矩阵（向量）的过程。由于仅仅是统计，在数据量很大时也比较快。

具体实现上，将 pyltp 分好词的文档作为训练语料输入，以空格分开，先记录每个词的标记状态（根据词中字在词中的位置，表示出相应的标记序列）。之后，遍历整个文档中所有的句子，每个句子遍历所

有词，统计出相应的频数和总数，遍历结束后用频数除以相应的总数获得频率。依照大数定律，将其视作概率，获得训练的结果。

分词功能通过 partition 函数，使用 Viterbi 算法实现。这里需要两个中间量：weight 矩阵（δ）和 path 矩阵（Ψ），Viterbi 算法在 weight 矩阵上实行动态规划，weight 保存每条路径到当前节点的概率，由于动态规划最优子结构的存在，可以利用递推公式进行迭代，并且保存当前选择的路径节点（状态）：

算法 10.5（维特比算法）

输入：模型 $\lambda=(A,B,\pi)$ 和观测 $O=(o_1,o_2,\cdots,o_T)$；

输出：最优路径 $I^*=(i_1^*,i_2^*,\cdots,i_T^*)$.

（1）初始化

$$\delta_1(i)=\pi_i b_i(o_1),\quad i=1,2,\cdots,N$$
$$\psi_1(i)=0,\quad i=1,2,\cdots,N$$

（2）递推．对 $t=2,3,\cdots,T$

$$\delta_t(i)=\max_{1\le j\le N}[\delta_{t-1}(j)a_{ji}]b_i(o_t),\quad i=1,2,\cdots,N$$
$$\psi_t(i)=\arg\max_{1\le j\le N}[\delta_{t-1}(j)a_{ji}],\quad i=1,2,\cdots,N$$

（3）终止

$$P^*=\max_{1\le i\le N}\delta_T(i)$$
$$i_T^*=\arg\max_{1\le i\le N}[\delta_T(i)]$$

迭代完成后，使用 path 回溯，找出概率最大的路径并输出相应的状态序列。最后，根据状态序列，将句子进行划分。

这里需要特别强调一下，对于未登录词，观测概率矩阵没有相应的发射概率。这里默认设定未登录词的状态概率为 S 的概率最大：

```
if len(tmp_list)==0:
    tmp_list=[0,0,0,1]
```

图 3 未登录词默认为 S，其中 S 对应第四个状态

partition 函数需要初始状态概率向量，状态概率转移矩阵，观测概率矩阵这三个量作为输入。为了测试方便，附件中提供了已经训练好的这几个量，利用 Python 提供的 pickle 模块导入就可以使用。

2.3 图形界面

为了便于测试和演示，使用 PyQt5 实现了带有图形界面的 HMM 分词器：

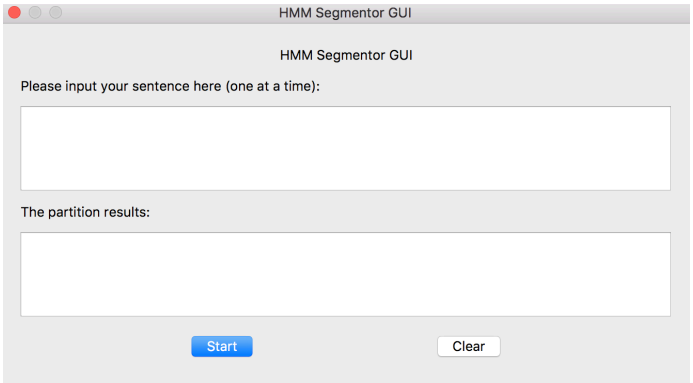

图 4 带图形界面的 HMM 分词器

在输入框输入句子，点击 Start 按钮，就可以获得分词结果；点击 Clear 按钮可以清空窗口：

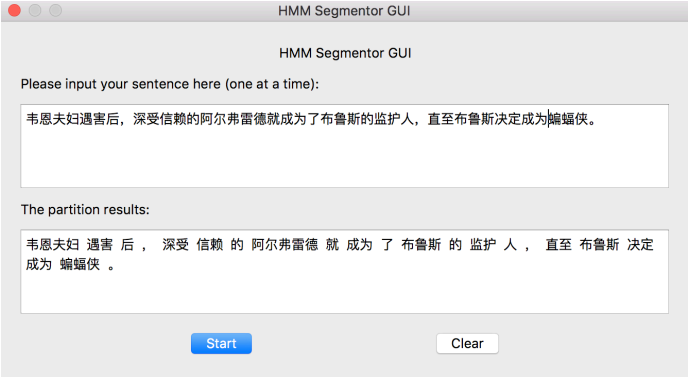

图 5 HMM 分词器效果演示

这个图形界面可以直接在附件中的文件夹中运行。

3. CRF 分词器

3.1 基本原理

马尔科夫随机场是一种概率无向图模型，可以用来表示一组随机变量的联合分布。条件随机场（CRF）是给定随机变量 X 的情况下，随机变量 Y 的马尔科夫随机场。CRF 是一种判别式模型，而线性链 CRF 可以用来进行序列标注。这时，在条件概率模型 $P(Y \mid X)$中，Y 是输出变量，表示标记序列，X 是输入变量，表示需要进行标注的观测序列。在中文分词中，Y 是状态序列，X 是句子中字的序列。

CRF 也是马尔科夫随机场的一种，可以表示一组随机变量的分布。在本次实验中，将单独的字视为随机变量，使用 Uni-gram 模型。训练好的 CRF 模型可以使用前向-后向算法计算概率，选择概率最大的序列作为标注序列输出。至于 CRF 的训练算法，则与其他机器学习算法使用的训练算法比较类似，比如随机梯度下降、拟牛顿法等等。

由于从头实现一个 CRF 的训练和预测模型十分困难，所以本次实验中使用 Python 的 CRF 库:sklearn-crfsuite 作为分词的模型。

具体的 API 文档见 https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html。

3.2 代码实现

CRF 的训练算法主要由 CRF_train 函数实现，而 CRF_train 函数主要使用了 character2features 函数获得每个单字的特征作为输入；最后，使用这个库提供的类似 sklearn 的训练接口 fit 函数进行训练，获得模型。CRF 的预测（分词）算法则比较简单，调用库里的 predict_single 接口，基于训练好的模型，使用前向-后向算法进行预测。

具体实现上，sklearn-crfsuite 要求输入的训练用的观测序列数据格式为字典组成的列表组成的列表。最外层的列表，表示整个训练集，由数个观测序列（句子）组成；中间的列表表示观测序列（句子），由许多字典组成；每个字典中包含了对应的单个节点/随机变量（这里是单个的字）的特征，如“is_upper”：True。

这里由于使用 Uni-gram 模型，使用的是线性链 CRF，所以仅考虑每个字的前一个字和后一个字；具体的，将每个字的[前中后，前中，中后，前后，前，中，后]作为这个字的特征输入（“中”代表自己）：

```
if i >= 1 and i<= real_sen_len(sen)-2:
    feature_dic["qzh"]=sen[i-1]+sen[i]+sen[i+1]
    feature_dic["qz"]=sen[i-1]+sen[i]
    feature_dic["zh"]=sen[i]+sen[i+1]
    feature_dic["qh"]=sen[i-1]+sen[i+1]
    feature_dic["q"]=sen[i-1]
    feature_dic["z"]=sen[i]
    feature_dic["h"]=sen[i+1]
    return feature_dic
```

图 6 单字的特征词典的构建

对于第一个字和最后一个字等特殊情况，使用‘*’标记不存在的部分。

训练用的标注序列数据格式与上面类似，为字符串组成的列表组成的列表。这里每个字符串就是对应节点/随机变量的标注，用于训练。

初始化 CRF 时，决定使用基于拟牛顿法的梯度下降算法（lbfgs），L1 和 L2 正则系数设为 0.1，迭代 200 轮。

实际训练的时候，发现训练这个模型占用内存的情况十分严重，因此在有 128G 内存的服务器上进行，并且只使用了语料中的前 8,000,000 条句子。即便这样，也训练了几个小时，远远超过训练 HMM 的时间。不过从最终结果来看，效果比较令人满意。

预测（分词）算法就比较简单，首先预处理获得句子中字的特征，作为输入，调用接口获得标注序列，根据标注的状态序列进行分词。

附件中提供了利用八百万条句子训练好的 CRF 模型，调用时使用 sklearn.externals 中的 joblib 模块来导入。

3.3 图形界面

同样的，为了方便测试和演示，使用 PyQt5 实现了带有图形界面的 CRF 分词器：

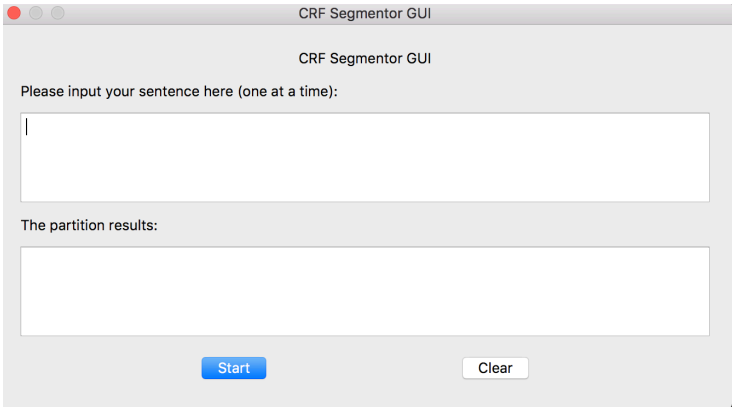

图 7 带图形界面的 CRF 分词器

分词效果如下：

图 8 CRF 分词器效果演示

同样的，这个带图形界面的分词器可以直接在附件的文件夹中运行。

4. 测试与评估

4.1 测试数据来源与方法

这里使用 SIGHAN 2005 Bakeoff 提供的数据和脚本来对两个分词器的效果进行评估。

SIGHAN 是国际计算语言学会（ACL）中文语言处理小组的简称，其英文全称为“Special Interest Group for Chinese Language Processing of the Association for Computational Linguistics”，又可以理解为“SIG 汉”或“SIG 漢”。而 Bakeoff 则是 SIGHAN 所主办的国际中文语言处理竞赛，第一届于 2003 年在日本札幌举行（Bakeoff 2003），第二届于 2005 年在韩国济州岛举行(Bakeoff 2005)，而 2006 年在悉尼举行的第三届（Bakeoff 2006）则在前两届的基础上加入了中文命名实体识别评测。目前 SIGHAN Bakeoff 已成功举办了 6 届，其中 Bakeoff 2005 的数据和结果在其主页上是完全免费和公开的。

语料共有四个来源：前缀为 as_，代表的是台湾中央研究院提供；前缀为 hk_，代表的是香港城市大学提供；前缀为 msr_，代表的是微软亚洲研究院提供；前缀为 pku_，代表的北京大学提供。这里全部使用相应的 UTF8 编码的文件。前两者为繁体，后两者为简体。

这里主要使用 gold 文件夹中提供的相应词表和“黄金分割”结果作为测试标准，对自己实现的分词器的分词结果进行测试。评分脚本的输入格式如下：评分脚本“score”是用来比较两个分词文件的，需要三个参数：

1. 训练集词表
2. “黄金”标准分词文件
3. 测试集的切分文件

利用其自带的中文分词工具进行说明。利用 score 脚本评分的命令如下：

```
./score ../gold/pku_training_words.txt ../gold/pku_test_gold.txt pku_test_seg.txt > score.txt
```

其中第一个参数需提供一个词表文件 pku_training_word.txt，标准分词文件输入为

pku_test_gold.txt，需要评测的分词文件为 pku_test_seg.txt，输出为 score.txt。而 score.txt 中包含了详细的评分结果，不仅有总的评分结果，还包括每一句的对比结果。

这里评测分词器的性能时，主要考虑准确率、召回率和 F 值三项。具体的测试结果在附件中给出。

很奇怪的一点是，训练 HMM 和 CRF 的语料中没有繁体字，然而两者在台湾中央研究院和香港城市大学提供的繁体字语料上性能表现不是很差。猜测这与 UTF8 的编码方式有关。

4.2 结果统计

考虑准确率、召回率、F 值三项。

准确率：

Precision	msr	pku	as	cityu
HMM	0.751	0.734	0.473	0.439
CRF	0.856	0.954	0.737	0.735

召回率：

Recall	msr	pku	as	cityu
HMM	0.779	0.726	0.572	0.540
CRF	0.893	0.942	0.766	0.764

F 值：

F Measure	msr	pku	as	cityu
HMM	0.765	0.730	0.517	0.484
CRF	0.874	0.948	0.751	0.749

柱状图统计结果如下：

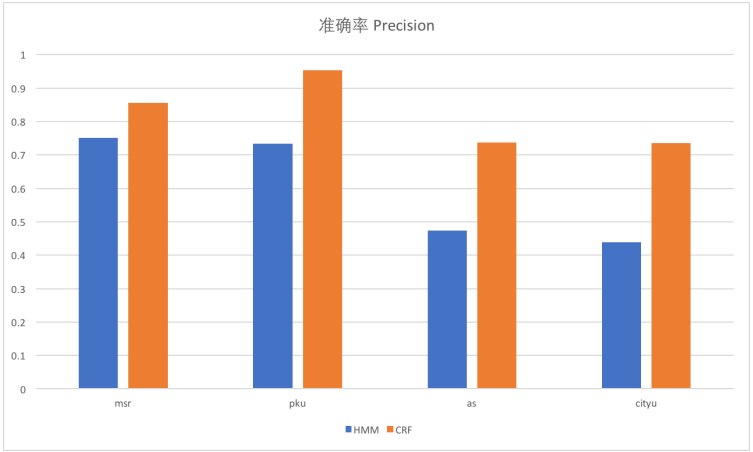

图 9 HMM 和 CRF 分词器准确率的对比

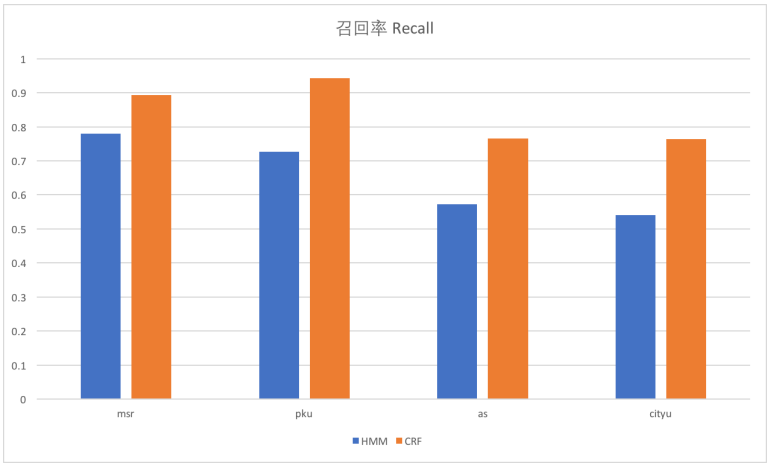

图 10 HMM 和 CRF 分词器召回率的对比

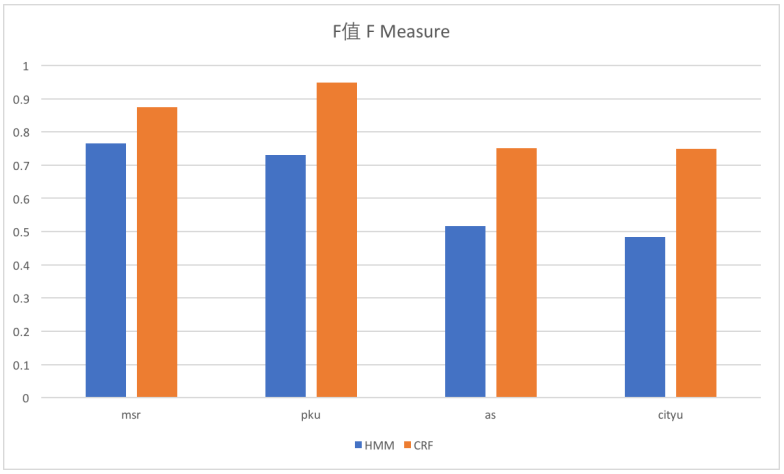

图 11 HMM 和 CRF 分词器 F 值的对比

从统计结果来看，两个分词器的分词性能比较令人满意。其中，CRF 分词器的性能在各个语料上的表现均超过了 HMM 分词器，证明 CRF 在中文分词上的性能会更好。此外，可以看出，两个分词器在繁体语料上的表现普遍不如简体语料。这与训练语料的选择有关。

SIGHAN 同时提供了可供选择的训练语料，这里由于想测试已经训练好的分词模型的通用性，所以没有使用其提供的训练语料。如果使用繁体语料训练的话，分词器在繁体语料上的表现应该会更好。

5. 结论

本次实验，实现了基于 HMM 和 CRF 的中文分词器，提供了训练和预测（分词）功能，并且提供了训练好的模型和图形界面，可以直接投入使用。使用比较权威的数据对分词器的性能进行了测试并获得了相应的结果，将两者结果进行了对比，表明了分词器的性能比较令人满意，且基于 CRF 的分词器比基于 HMM 的分词器性能要更好一些。

6. 参考资料

1. 《信息内容安全》课程课堂 PPT
2. 《统计学习方法》，李航
3. CSDN 技术博客
4. sklearn-crfsuite 官方 API 文档：
   https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html
5. SIGHAN 2005 Bakeoff Data:
   http://sighan.cs.uchicago.edu/bakeoff2005/