中文分词入门之字标注法

52nlp

2014-1-4

在《自然语言处理领域的两种创新观念》中，张俊林博士谈了两种创新模式：一种创新是研究模式的颠覆，另外一种创新是应用创新，前者需要 NLP 领域出现爱因斯坦式的革新人物，后者则是强调用同样的核心技术做不一样的应用。

在自然语言处理领域，多数创新都属于后者，譬如统计机器翻译，Brown 就是学习和借鉴了贾里尼克将语音识别看成通信问题的思想，将信源信道模型应用 到了机器翻译之中，从而开辟了 SMT 这一全新领域。而 Nianwen Xue 将词性标注的思想应用到中文分词领域，成就了字标注的中文分词方法（Chinese Word Segmentation as Character Tagging），同样取得了巨大的成功。这里曾通过节选黄昌宁老师和赵海博士在 07 年第 3 期《中文信息学报》上发表的《中文分词十年回顾》介绍了《基于字标注的中文分词方法》，因此还不太清楚这类方法的读者可以通过上述文章了解该方法的一些背景，本文的重点是实践。

既然基于字标注的中文分词方法是将中文分词当作词性标注的问题来对待，那么就必须有标注对象和标注集了。形象一点，从这个方法的命名上我们就可以推断出它的标注是基本的汉字（还包括一定数量的非汉字字符），而标注集则比较灵活，这些标注集都是依据汉字在汉语词中的位置设计的，最简单的是 2-tag，譬如将词首标记设计为 B，而将词的其他位置标记设计为 I，那么“中国”就可以标记为“中/B 国/I”，“海南岛”则可以标记为“海/B 南/I 岛/I”，相应地，对于如下分好词的句子：

瓦西里斯 的 船只 中 有 4 0 % 驶 向 远东 ， 每个 月 几乎 都 有 两三条 船 停靠 中国 港口 。

基于 2-tag（B，I）的标注就是：

瓦/B 西/I 里/I 斯/I 的/B 船/B 只/I 中/B 有/B 4/B 0/I %/I 驶/B 向/B 远/B 东/I ，/B 每/B 个/I 月/B 几/B 乎/I 都/B 有/B 两/B 三/I 条/I 船/B 停/B 靠/I 中/B 国/I 港/B 口/I 。/B

除了 2-tag，还有 4-tag、6-tag 等，都是依据字在词中的位置设计的，本文主要目的是从实践的角度介绍基于字标注的中文分词方法设计，以达到抛砖引玉的作用，因此我们仅选用 2-tag（B，I）标注集进行实验说明。

有了标注对象和标注集，那么又如何进行中文分词呢？仍以前面的句子为例，只不过这里还没有进行中文分词：

瓦西里斯的船只中有４０％驶向远东，每个月几乎都有两三条船停靠中国港口。

首先，你需要将标注对象独立出来，事实上就是给每个字符加上空格：

瓦 西 里 斯 的 船 只 中 有 ４ ０ ％ 驶 向 远 东 ， 每 个 月 几 乎 都 有 两 三 条 船 停 靠 中 国 港 口 。

其次，假设你已经训练好了一个字标注器，那么就直接给这些字进行标注吧：

瓦/B 西/I 里/B 斯/I 的/B 船/I 只/B 中/B 有/B ４/I ０/I ％/I 驶/I 向/B 远/I 东/B ，/B 每/B 个/I 月/I 几/B 乎/I 都/B 有/I 两/B 三/B 条/B 船/I 停/B 靠/I 中/B 国/I 港/I 口/I 。/B

最后，需要你做得就是按照这两个标记的意思还原中文词，并且除去这些标记：

瓦西 里斯 的船 只 中 有４０％驶 向远 东 ， 每个月 几乎 都有 两 三 条船 停靠 中国港口 。

好了，这就是字标注方法的流程和分词结果，很遗憾，这个标注器的效果不太好，不过没关系，你可以设计更好的字标注器，英文词性标注的被老外研究 的太充分了，什么 HMM、TBL、最大熵、条件随机场、决策树等等等等，不仅仅是方法，连开源工具都给你提供，完全可以拿来主义。

那么又怎样拿来主义设计自己的字标注中文分词器呢？不知读者可记着 Citar，不记得的话可以温习一下《HMM 在自然语言处理中的应用一：词性标注 6》，Citar 是一个“Hidden Markov Model trigram POS tagger”，需要有标注好的语料来训练相应语言的词性标注器，其核心的 HMM 标注框架是不依赖于语言的，但是在处理未登录词时 Citar 是主要利用英 文词缀信息进行标注的，因此其对于英文词性标注来说效果相对较好，对于其他语言，特别是中文这种没有词形变化的孤立语来说，其词性标注效果要打一点折扣。 不过没关系，这里主要谈的是思想，而不是工程上的应用，因此暂时可以忽略这点影响，对于本文利用其所设计的字标注器来说，同样适用。

可是哪里有这样的训练语料呢？俗话说得好：自己动手，丰衣足食。虽然没有哪个组织提供这样的字标注语料库，但是我们有 SIGHAN Bakeoff 提供的 icwb2-data，你完全可以利用自己熟悉的编程语言写一个前处理程序，将其转换为我们所需要的字标注训练语料库形式。下一节我 们将以微软亚洲研究院提供的中文分词语料为例，利用 Citar 完成一个基于 HMM trigram 的字标注中文分词程序。

二、

虽然基于字标注的中文分词借鉴了词性标注的思想，但是在实践中，多数 paper 中的方法似乎局限于最大熵模型和条件随机场的应用，所以我常常疑惑字 标注中文分词方法为什么不采用别的模型和方法呢？莫非是由于其诞生之初就采用了最大熵模型的缘故。但是，在词性标注中，Citar 实现的是 TnT 中所采用 的 HMM trigram 方法，其所宣称的性能是不亚于最大熵模型等词性标注器的。基于这样的前提，本文就验证一下基于 Citar 实现的 HMM trigram 字标注中文分词器的性能。

我们以微软亚洲研究院提供的中文分词语料为例，采用 2-tag(B,I)标记集，只针 utf-8 编码文本。首先准备训练语料，原始训练集 msr_training.utf8 的形式是人工分好词的中文句子形式：

“ 人们 常 说 生活 是 一 部 教科书 ， 而 血 与 火 的 战争 更 是 不可多得 的 教科书 ， 她 确实 是 名副其实 的 ‘ 我 的 大学 ’ 。
“ 心 静 渐 知 春 似 海 ， 花 深 每 觉 影 生 香 。
…

需要将其转换为字标注形式，这里 52nlp 利用 perl 写了一个简单的 2-tag 转换程序 CharacterTagging.pl,不过请注意这个程序仅处理 utf-8 编码的文本：

./CharacterTagging.pl -i msr_training.utf8 -o msr_training.tagging.utf8

其中-i 指示的是输入文件，-o 指示的是输出文件，msr_training.tagging.utf8 的内容示例如下：

“/B 人/B 们/I 常/B 说/B 生/B 活/I 是/B 一/B 部/B 教/B 科/I 书/I ，/B 而/B 血/ 与/B 火/B 的/B 战/B 争/I 更/B 是/B 不/B 可/I 多/I 得/I 的/B 教/B 科/I 书/I ，/B 她/B 确/B 实/I 是/B 名/B 副/I 其/I 实/I 的/B ‘/B 我/B 的/B 大/B 学/I ’/B /B
“/B 心/B 静/B 渐/B 知/B 春/B 似/B 海/B ，/B 花/B 深/B 每/B 觉/B 影/B 生/B 香/ 。/B
…

有了这个 2-tag 的字标注训练语料库，就可以利用 Citar 中编译好的 train 来训练字标注器所需的输入文件了：

./train msr_training.tagging.utf8 msr_lex msr_ngram

其中 msr_lex 包含了 5000 多个汉字与其标记的共同出现频率，形式如下：

锶 B 4
薛 B 97
铜 B 114 I 26

佛 B 142 I 144
榫 B 2
觚 B 1
萋 B 2 I 8
钮 B 4 I 7
…

msr_ngram 则是标记本身及其之间的共现频率，形式如下：

B 2368391
I 1682056
173836
86918
B B 1027319
I B 1254154
B 86017
I I 427001
B I 1255055
B 86918
I 901
86918
…

注：由于没有尖括号在 Wordpress 中被屏蔽，以上内容有误，谢谢读者 bflout 的提醒，以下重新附上 msr_ngram:

1 B 2368391
2 I 1682056
3 <START> 173836
4 <END> 86918
5 B B 1027319
6 I B 1254154
7 B <END> 86017
8 I I 427001
9 B I 1255055
10<START> B 86918
11I <END> 901
12<START> <START> 86918
13B I B 1039293
14B B B 408801
15I I <END> 285
16B B <END> 18403
17B I I 215146
18<START> B I 60460
19I I B 214861

20I B I 594480
21B B I 600115
22<START> <START> B 86918
23I I I 211855
24B I <END> 616
25<START> B B 26449
26<START> B <END> 9
27I B B 592069
28I B <END> 67605

注意，这两个文件都很小，msr_lex 只有 64k,而 msr_ngram 则不到 1k，所占用资源极小。

在利用 Citar 的 tag 进行标注之前，需要对测试集 msr_test.utf8 的字符进行切分，在 NIST2009 机器翻译的评测主页的底部提供了这个工具 :splitUTF8Characters.p：

./splitUTF8Characters.pl -i msr_test.utf8 -o msr_test.split.utf8

msr_test.utf8 的形式如下：

扬帆远东做与中国合作的先行
希腊的经济结构较特殊。
…

切分后的 msr_test.split.utf8 形式如下：

扬 帆 远 东 做 与 中 国 合 作 的 先 行
希 腊 的 经 济 结 构 较 特 殊 。
…

有了 msr_test.split.utf8，我们就可以利用 Citar 的 tag 进行字标注了：

./tag msr_lex msr_ngram < msr_test.split.utf8 > msr_test.hmmtagging.utf8

标注后的 msr_test.hmmtagging.utf8 形式如下：

扬/B 帆/I 远/B 东/I 做/B 与/B 中/B 国/I 合/B 作/I 的/B 先/I 行/B
希/B 腊/I 的/B 经/B 济/I 结/B 构/I 较/B 特/B 殊/I 。/B
…

最后，就是按照标记结果合并字符并去除标记了。这里 52nlp 利用 perl 写了一个简单的还原程序 Character2word.pl,不过请注意这个程序仅处理 utf-8 编码的文本：

./Character2word.pl -i msr_test.hmmtagging.utf8 -o msr_test.hmmseg.utf8

msr_test.hmmseg.utf8 既是最终的分词结果，其形式如下：

扬帆 远东 做 与 中国 合作 的先 行
希腊 的 经济 结构 较 特殊 。
…

当然，这个字标注中文分词的结果好坏还需要利用 SIGHAN Bakeoff 的 score 进行评分：

../icwb2-data/scripts/score ../icwb2-data/gold/msr_training_words.utf8 msr_test_gold.utf8 msr_test.hmmseg.utf8 > msr_hmmseg.score

最终的评分结果在 msr_hmmseg.score 中，总的评分如下：

…
=== SUMMARY:
=== TOTAL INSERTIONS: 10304
=== TOTAL DELETIONS: 7030
=== TOTAL SUBSTITUTIONS: 30727
=== TOTAL NCHANGE: 48061
=== TOTAL TRUE WORD COUNT: 106873
=== TOTAL TEST WORD COUNT: 110147
=== TOTAL TRUE WORDS RECALL: 0.647
=== TOTAL TEST WORDS PRECISION: 0.627
=== F MEASURE: 0.637
=== OOV Rate: 0.026
=== OOV Recall Rate: 0.181
=== IV Recall Rate: 0.659
msr_test.hmmseg.utf8 10304 7030 30727 48061 106873 110147 0.647 0.627 0.637 0.026 0.181 0.659

结果残不忍睹，不过没关系，重要的是思想，当你明白了如何进行字标注中文分词的设计和操作之后，可以做得改进有很多，譬如增加标记集，修改 Citar 中不合适的未登录词处理方法，甚至重新采用其他模型等等等等。同样，52nlp 也会在合适的时候介绍一下最大熵模型和条件随机场在中文分词中的 应用，欢迎继续关注本博客！

三、

最近要整理一下课程图谱里的中文课程，需要处理中文，首当其冲的便是 中文分词的问题。目前有一些开源的或者商用的中文分词器可供选择，但是出于探索或者好奇心的目的，想亲手打造一套实用的中文分词器，满足实际的需求。这些年无论是学习的时候还是工作的时候，林林总总的接触了很多实用的中文分词器，甚至在这里也写过一些 Toy 级别的中文分词相关文章，但是没有亲手打造过自己的分词器，甚为遗憾。目前自己处于能自由安排工作的阶段，所以第一步就是想从中文信息处理的桥头堡“中文分词”入手，打造一个实用的中文分词器，当然，首先面向的对象是课程图谱所在的教育领域。

大概 4 年前，这里写了两篇关于字标注中文分词的文章：中文分词入门之字标注法，文中用 2-tag(B,I)进行说明并套用开源的 HMM 词性标注工具 Citar(A simple Trigram HMM part-of-speech tagger）做了演示，虽然分词效果不太理想，但是能抛砖引玉，也算是有点用处。这次捡起中文分词，首先想到的依然是字标注分词方法，在回顾了一遍黄昌 宁老师和赵海博士在 07 年第 3 期《中文信息学报》上发表的《中文分词十年回顾》后，决定这次从 4-tag 入手，并且探索一下最大熵模型和条件随机场（CRF）在中文分词字标注方法上的威力。这方面的文献大家可参考张开旭博士维护的“中文分词文献列表”。这里主要基于已有文献的思路和现成的开源工具做一些验证，包括张乐博士的最大熵模型工具包(Maximum Entropy Modeling Toolkit for Python and C++)和条件随机场的经典工具包 CRF++(CRF++: Yet Another CRF toolkit)。

这个系列也将补充两篇文章，一篇简单介绍背景知识并介绍如何利用现成的最大熵模型工具包来做中文分词，另外一篇介绍如何用 CRF++做字标注分词， 同时基于 CRF++的 python 接口提供一份简单的 CRF Python 分词代码，仅供大家参考。至于最大熵和 CRF++的背景知识，这里不会过多涉及，推荐大家跟踪一下课程图谱上相关的机器学习公开课。

这次使用的中文分词资源依然是 SIGHAN 提供的 backoff 2005 语料，目前封闭测试最好的结果是 4-tag+CFR 标注分词，在北大语料库上可以在准确率，召回率以及 F 值上达到 92%以上的效果，在微软语料库上可以到达 96%以上的效果。不清楚这份中文分词资源的同学可参考很早之前写的这篇文章：中文分词入门之资源。以下我们将转入这篇文章的主题，基于最大熵模型的字标注中文分词。

首先仍然是下载安装和使用张乐博士的最大熵模型工具包，这次使用的是其在 github 上的代码：maxent ， 进入到代码主目录 maxent-master 后，正常按照 configure，make 及 make install 就可以完成 C++库的安装，再进入到子目录 python 下，执行 python setup.py install 即可，这个 python 库是通过强大的 SWIG 生成的。关于这个最大熵模型工具包详情及背景，推荐看官方 manual 文档，写得非常详细。与“中文分词入门之字标注法 2” 的做法类似，这里利用这个工具包 example 里带的英文词性标注脚本来做字标注中文分词，先验证一下是否可行，关于这个 case 的解读，可以参考 manual 文档里的“4.6 Case Study: Building a maxent Part-of-Speech Tagger”。

第一步仍然是将 backoff2005 里的训练数据转化为这个 POS Tagger 所需的训练数据格式，还是以微软亚洲研究院提供的中文分词语料为例，这次我们采用 4-tag(B(Begin，词首)，E(End，词尾)，M(Middle，词中)，S(Single,单字词)) 标记集，只处理 utf-8 编码文本。原始训练集./icwb2-data/training /msr_training.utf8 的形式是人工分好词的中文句子形式，如：

“ 人们 常 说 生活 是 一 部 教科书 ， 而 血 与 火 的 战 争 > 更 是 不可多得 的 教科书 ， 她 确实 是 名副其 实 的‘ 我 的 > 大学 ’。
1“ 心 静 渐 知 春 似 海 ， 花 深 每 觉 影 生 香 。
2“ 吃 屎 的 东西 ， 连 一 捆 麦 也 铡 不 动 呀 ？
3 他 “ 严格要求 自己 ， 从 一个 科举 出身 的 进士 成为 一
4 个 伟> 大 的 民主主义 者 ， 进而 成为 一 位 杰出 的 党外 共产主
5 义 战 士 ， 献身 于 崇高 的 共产主义 事业 。
6“ 征 而 未 用 的 耕地 和 有 收益 的 土地 ， 不准 荒芜 。
7“ 这 首先 是 个 民族 问题 ， 民族 的 感情 问题 。
8 ’ 我 扔 了 两颗 手榴弹 ， 他 一下子 出 溜 下去 。
9“ 废除 先前 存在 的 所有制 关系 ， 并不是 共产主义 所 独具 的 特
10征 。
“ 这个 案子 从 始 至今 我们 都 没有 跟 法官 接触 过 ， 也 > 没 有 跟 原告 、 被告 接触 过 。
“ 你 只有 把 事情 做好 ， 大伙 才 服 你 。

这里我们提供一个 4-tag 的标注脚本 character_tagging.py 对这个训练语料进行标注：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # Author: 52nlpcn@gmail.com
4  # Copyright 2014 @ YuZhen Technology
5  #
6  # 4 tags for character tagging: B(Begin), E(End), M(Middle), S(Single)
7
8  import codecs
9  import sys
10
11def character_tagging(input_file, output_file):
12     input_data = codecs.open(input_file, 'r', 'utf-8')
13     output_data = codecs.open(output_file, 'w', 'utf-8')
14     for line in input_data.readlines():
15          word_list = line.strip().split()
16          for word in word_list:
17               if len(word) == 1:
18                    output_data.write(word + "/S ")
19               else:
```

```
                output_data.write(word[0] + "/B ")
                for w in word[1:len(word)-1]:
                    output_data.write(w + "/M ")
                output_data.write(word[len(word)-1] + "/E ")
        output_data.write("\n")
    input_data.close()
    output_data.close()

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print "Please use: python character_tagging.py input output"
        sys.exit()
    input_file = sys.argv[1]
    output_file = sys.argv[2]
    character_tagging(input_file, output_file)
```

只需执行“python character_tagging.py icwb2-data/training/msr_training.utf8 msr_training.tagging.utf8” 即可得到最大熵词性标注训练器所需要的输入文件 msr_training.tagging.utf8，样例如下：

```
“/S 人/B 们/E 常/S 说/S 生/B 活/E 是/S 一/S 部/S 教/B 科/M 书/E ，/S 而/S 血/S 与/S 火/S 的/S 战/B 争/E 更/S 是/S 不/B 可/M 多/M 得/E 的/S 教/B 科/M 书/E ，/S 她/S 确/B 实/E 是/S 名/B 副/M 其/M 实/E 的/S ‘/S 我/S 的/S 大/B 学/E ’/S 。/S
“/S 心/S 静/S 渐/S 知/S 春/S 似/S 海/S ，/S 花/S 深/S 每/S 觉/S 影/S 生/S 香/S 。/S
“/S 吃/S 屎/S 的/S 东/B 西/E ，/S 连/S 一/S 捆/S 麦/S 也/S 铡/S 不/S 动/S 呀/S ？/S
他/S “/S 严/B 格/M 要/M 求/E 自/B 己/E ，/S 从/S 一/B 个/E 科/B 举/E 出/B 身/E 的/S 进/B 士/E 成/B 为/E 一/B 个/E 伟/B 大/E 的/S 民/B 主/M 主/M 义/E 者/S ，/S 进/B 而/E 成/B 为/E 一/S 位/S 杰/B 出/E 的/S 党/B 外/E 共/B 产/M 主/M 义/E 战/B 士/E ，/S 献/B 身/E 于/S 崇/B 高/E 的/S 共/B 产/M 主/M 义/E 事/B 业/E 。/S
“/S 征/S 而/S 未/S 用/S 的/S 耕/B 地/E 和/S 有/S 收/B 益/E 的/S 土/B 地/E ，/S 不/B 准/E 荒/B 芜/E 。/S
“/S 这/S 首/B 先/E 是/S 个/S 民/B 族/E 问/B 题/E ，/S 民/B 族/E 的/S 感/B 情/E 问/B 题/E 。/S
’/S 我/S 扔/S 了/S 两/B 颗/E 手/B 榴/M 弹/E ，/S 他/S 一/B 下/M 子/E 出/S 溜/S 下/B 去/E 。/S
“/S 废/B 除/E 先/B 前/E 存/B 在/E 的/S 所/B 有/M 制/E 关/B 系/E ，/S 并/B 不/M 是/E 共/B 产/M 主/M 义/E 所/S 独/B 具/E 的/S 特/B 征/E 。/S
“/S 这/B 个/E 案/B 子/E 从/S 始/S 至/B 今/E 我/B 们/E 都/S 没/B 有/E 跟/S 法/B 官/E 接/B 触/E 过/S ，/S 也/S 没/B 有/E 跟/S 原/B 告/E 、/S 被/B 告/E 接/B 触/E 过/S 。/S
“/S 你/S 只/B 有/E 把/S 事/B 情/E 做/B 好/E ，/S 大/B 伙/E 才/S 服/S 你/S 。/S
```

现在就可以用张乐博士最大熵模型工具包中自带的 PosTagger 来训练一个字标注器了：

./maxent-master/example/postagger/postrainer.py msr_tagger.model -f msr_training.tagging.utf8 - iters 100

这里指定迭代训练 100 轮，没有什么依据，仅作此次测试之用，训练结束之后，我们得到一个字标注所用的最大熵模型：msr_tagger.model，还有几个副产品。现在我们需要做得是准备一份测试语料，然后利用最大熵模型标注器对测试语料进行标注。原始的测试语料 是 icwb2-data/testing/msr_test.utf8 ，样例如下：

```
1 扬帆远东做与中国合作的先行
2 希腊的经济结构较特殊。
3 海运业雄踞全球之首，按吨位计占世界总数的１７％。
4 另外旅游、侨汇也是经济收入的重要组成部分，制造业规模相对较小。
5 多年来，中希贸易始终处于较低的水平，希腊几乎没有在中国投资。
6 十几年来，改革开放的中国经济高速发展，远东在崛起。
7 瓦西里斯的船只中有４０％驶向远东，每个月几乎都有两三条船停靠中国港口。
8 他感受到了中国经济发展的大潮。
9 他要与中国人合作。
10他来到中国，成为第一个访华的大船主。
```

需要将其单字离散化并添加空格，便于标注，这里我们同样提供一个 python 脚本 character_split.py 对测试语料进行处理：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # Author: 52nlpcn@gmail.com
4  # Copyright 2014 @ YuZhen Technology
5  #
6  # split chinese characters and add space between them
7
8  import codecs
9  import sys
10
11def character_split(input_file, output_file):
12     input_data = codecs.open(input_file, 'r', 'utf-8')
13     output_data = codecs.open(output_file, 'w', 'utf-8')
14     for line in input_data.readlines():
15          for word in line.strip():
16               output_data.write(word + " ")
17          output_data.write("\n")
18     input_data.close()
19     output_data.close()
20
21if __name__ == '__main__':
22     if len(sys.argv) != 3:
```

```
23          print "Please use: python character_split.py input output"
24          sys.exit()
25      input_file = sys.argv[1]
26      output_file = sys.argv[2]
27      character_split(input_file, output_file)
```

执行“python character_split.py icwb2-data/testing/msr_test.utf8 msr_test.split.utf8”即可得到可用于标注测试的测试语料 msr_test.split.utf8，样例如下：

```
  扬 帆 远 东 做 与 中 国 合 作 的 先 行
  希 腊 的 经 济 结 构 较 特 殊 。
1 海 运 业 雄 踞 全 球 之 首 ， 按 吨 位 计 占 世 界 总 数 的 1 7 % 。
2 另 外 旅 游 、 侨 汇 也 是 经 济 收 入 的 重 要 组 成 部 分 ， 制 造 业 规 模
3 相 对 较 小 。
4 多 年 来 ， 中 希 贸 易 始 终 处 于 较 低 的 水 平 ， 希 腊 几 乎 没 有 在 中
5 国 投 资 。
6 十 几 年 来 ， 改 革 开 放 的 中 国 经 济 高 速 发 展 ， 远 东 在 崛 起 。
7 瓦 西 里 斯 的 船 只 中 有 4 0 % 驶 向 远 东 ， 每 个 月 几 乎 都 有 两 三
8 条 船 停 靠 中 国 港 口 。
9 他 感 受 到 了 中 国 经 济 发 展 的 大 潮 。
10 他 要 与 中 国 人 合 作 。
  他 来 到 中 国 ， 成 为 第 一 个 访 华 的 大 船 主 。
```

现在执行最大熵标注脚本即可得到字标注结果：

```
./maxent-master/example/postagger/maxent_tagger.py -m msr_tagger.model msr_test.split.utf8 > msr_test.split.tag.utf8
```

msr_test.split.tag.utf8 即是标注结果，样例如下：

```
  扬/B 帆/M 远/M 东/M 做/E 与/S 中/B 国/E 合/B 作/E 的/S 先/B 行/E
  希/B 腊/E 的/S 经/B 济/E 结/B 构/E 较/S 特/B 殊/E 。/S
1 海/B 运/M 业/E 雄/B 踞/E 全/B 球/E 之/S 首/S ， /S 按/S 吨/B 位/E 计/B 占/E 世/B
2 界/E 总/B 数/E 的/S 1 /B 7 /M %/E 。/S
3 另/B 外/E 旅/B 游/E 、/S 侨/B 汇/E 也/B 是/E 经/B 济/E 收/B 入/E 的/S 重/B 要/E
4 组/B 成/M 部/M 分/E ， /S 制/B 造/M 业/E 规/B 模/E 相/B 对/E 较/B 小/E 。/S
5 多/B 年/E 来/S ， /S 中/S 希/S 贸/B 易/E 始/B 终/E 处/B 于/E 较/B 低/E 的/S 水/B
6 平/E ， /S 希/B 腊/E 几/B 乎/E 没/B 有/E 在/S 中/B 国/E 投/B 资/E 。/S
7 十/B 几/M 年/E 来/S ， /S 改/B 革/M 开/M 放/E 的/S 中/B 国/E 经/B 济/E 高/B 速/E
8 发/B 展/E ， /S 远/B 东/E 在/S 崛/B 起/E 。/S
9 瓦/B 西/M 里/M 斯/E 的/S 船/B 只/E 中/S 有/S 4 /B 0 /M %/E 驶/S 向/S 远/B 东
10/E ， /S 每/B 个/M 月/E 几/B 乎/E 都/S 有/S 两/S 三/S 条/S 船/S 停/S 靠/S 中/B 国
  /M 港/M 口/E 。/S
  他/S 感/B 受/E 到/S 了/S 中/B 国/E 经/B 济/E 发/B 展/E 的/S 大/B 潮/E 。/S
```

他/S 要/S 与/S 中/B 国/M 人/M 合/M 作/E 。/S
他/S 来/B 到/E 中/B 国/E ，/S 成/B 为/E 第/B 一/M 个/E 访/B 华/E 的/S 大/B 船/M 主/E 。/S

最后我们还需要一个脚本，按标注的词位信息讲这份结果再转化为分词结果，这里我们仍然提供一个转换脚本 character_2_word.py：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # Author: 52nlpcn@gmail.com
4  # Copyright 2014 @ YuZhen Technology
5  #
6  # Combining characters based the 4-tag tagging info
7
8  import codecs
9  import sys
10
11def character_2_word(input_file, output_file):
12     input_data = codecs.open(input_file, 'r', 'utf-8')
13     output_data = codecs.open(output_file, 'w', 'utf-8')
14     # 4 tags for character tagging: B(Begin), E(End), M(Middle), S(Single)
15     for line in input_data.readlines():
16          char_tag_list = line.strip().split()
17          for char_tag in char_tag_list:
18               char_tag_pair = char_tag.split('/')
19               char = char_tag_pair[0]
20               tag = char_tag_pair[1]
21               if tag == 'B':
22                    output_data.write(' ' + char)
23               elif tag == 'M':
24                    output_data.write(char)
25               elif tag == 'E':
26                    output_data.write(char + ' ')
27               else: # tag == 'S'
28                    output_data.write(' ' + char + ' ')
29          output_data.write("\n")
30     input_data.close()
31     output_data.close()
32
33if __name__ == '__main__':
34     if len(sys.argv) != 3:
35          print "Please use: python character_2_word.py input output"
36          sys.exit()
37     input_file = sys.argv[1]
```

```
38    output_file = sys.argv[2]
39    character_2_word(input_file, output_file)
```

执行 “python character_2_word.py msr_test.split.tag.utf8 msr_test.split.tag2word.utf8” 即可得到合并后的分词结果 msr_test.split.tag2word.utf8，样例如下：

```
扬帆远东做 与 中国 合作 的 先行
希腊 的 经济 结构 较 特殊 。
海运业 雄踞 全球 之 首 ， 按 吨位 计占 世界 总数 的 １７％ 。
另外 旅游 、 侨汇 也是 经济 收入 的 重要 组成部分 ， 制造业 规模 相对 较小 。
多年 来 ， 中 希 贸易 始终 处于 较低 的 水平 ， 希腊 几乎 没有 在 中国 投资 。
十几年 来 ， 改革开放 的 中国 经济 高速 发展 ， 远东 在 崛起 。
瓦西里斯 的 船只 中 有 ４０％ 驶 向 远东 ， 每个月 几乎 都 有 两 三 条 船 停 靠 中国港口 。
他 感受 到 了 中国 经济 发展 的 大潮 。
他 要 与 中国人合作 。
他 来到 中国 ， 成为 第一个 访华 的 大船主 。
```

有了这个字标注分词结果，我们就可以利用 backoff2005 的测试脚本来测一下这次分词的效果了：

```
./icwb2-data/scripts/score ./icwb2-data/gold/msr_training_words.utf8
./icwb2-data/gold/msr_test_gold.utf8 msr_test.split.tag2word.utf8 >
msr_maxent_segment.score
```

结果如下：

```
=== SUMMARY:
=== TOTAL INSERTIONS: 5343
=== TOTAL DELETIONS: 4549
=== TOTAL SUBSTITUTIONS: 12661
=== TOTAL NCHANGE: 22553
=== TOTAL TRUE WORD COUNT: 106873
=== TOTAL TEST WORD COUNT: 107667
=== TOTAL TRUE WORDS RECALL: 0.839
=== TOTAL TEST WORDS PRECISION: 0.833
=== F MEASURE: 0.836
=== OOV Rate: 0.026
=== OOV Recall Rate: 0.565
=== IV Recall Rate: 0.846
### msr_test.split.tag2word.utf8 5343 4549 12661 22553 106873 107667
0.839 0.833 0.836 0.026 0.565 0.846
```

这个分词结果也比较一般，不过还有很多可以优化的地方，不过最主要的还是要设计适合中文分词字标注的特征模板，而这里使用的这份词性标注代码在抽取特征的时候主要考虑的是英文词性标注，所以我们完全可以基于一些已有的最大熵字标志文章来设计特征模板，进行特征提取和优化，不过这份工作就留给读者朋友 了。下一篇文章我们直接进入 CRF 字标注分词，大家将可以看到如何利用 CRF++现有的工作，在封闭的微软语料库上训练一个准确率，召回率以及 F 值可以达到 96%的中文分词器。

四、

上一节主要介绍的是利用最大熵工具包来做字标注中文分词，这一节我们直奔主题，借用条件随机场工具“CRF++: Yet Another CRF toolkit”来完成字标注中文分词的全过程。

关于条件随机场（CRF）的背景知识，推荐参考阅读一些经典的文献：《条件随机场文献阅读指南》，另外再额外推荐一个 tutorial:《Classical Probabilistic Models and Conditional Random Fields》，这份关于 CRF 的文档分别从概率模型（NB，HMM，ME， CRF)之间的关系以及概率图模型背景来介绍条件随机场，比较清晰：

While a Hidden Markov Model is a sequential extension to the Na ve Bayes Model, Conditional Random Fields can be understood as a sequential extension to the Maximum Entropy Model.

如果这些还不够过瘾，推荐课程图谱上收录的 Coursera 创始人之一 Daphne Koller 的“概率图模型公开课”，相信拿下这门课之后，对于上述概率模型，会有一种“一览众山小”的感觉。

不过我们还是要从安装 CRF++工具包说起，在 Linux 或者 Mac OS 系统下，下载 C++源代码安装包(这里用的是 CRF++-0.58.tar.gz)之后，依然是 “configure & make & (sudo) make install”，安装完毕之后，可以 cd python 进入到其同样用 SWIG 生成的 Python 工具包下，安装 python 包：python setup.py build & (sudo) python setup.py install。安装完毕之后，可以在 python 解释器下测试，是否能成功 import CRFPP，如果 ok，则准备工作就绪。

上一节我们利用最大熵模型工具包里自带的词性标注工具进行的中文分词，稍微有些曲折，这一节我们依然利用 CRF++ example 里的样例进行测试，不过好处是，CRF++ example 里有个 seg 目录，这个 seg 目录对应的是一个日文分词的样例，正好可以套用到我们的中文分词中来。在安装包目录下，cd example, cd seg 目录后，有 4 个文件：

exec.sh（执行脚本）
template（特征模板）
test.data（测试集）
train.data（训练集）

有了这 4 个文件，我们可以做得事情就比较简单，只要按测试集，训练集的格式准备数据就可以了，特征模板和执行脚本可以套用，不过这里简单解读一下这几个 CRF++文件。首先来看训练集：

1 每 k B
2 日 k I
3 新 k I
4 聞 k I
5 社 k I
6 特 k B
7 別 k I
8 顧 k B
9 問 k I
10４ n B

这里第一列是待分词的日文字，第二列暂且认为其是词性标记，第三列是字标注中的 2-tag(B, I)标记，这个很重要，对于我们需要准备的训练集，主要是把这一列的标记做好，不过需要注意的是，其断句是靠空行来完成的。

再来看测试集的格式：

1 よ h I
2 っ h I
3 て h I
4 私 k B
5 た h B
6 ち h I
7 の h B
8 世 k B
9 代 k I
10が h B

同样也有 3 列，第一列是日文字，第二列第三列与上面是相似的，不过在测试集里第三列主要是占位作用。事实上，CRF++对于训练集和测试集文件格式 的要求是比较灵活的，首先需要多列，但不能不一致，既在一个文件里有的行是两列，有的行是三列；其次第一列代表的是需要标注的“字或词”，最后一列是输出位”标记 tag”，如果有额外的特征，例如词性什么的，可以加到中间列里，所以训练集或者测试集的文件最少要有两列。

接下里我们再来详细的分析一下特征模板文件：

```
1 # Unigram
2 U00:%x[-2,0]
3 U01:%x[-1,0]
4 U02:%x[0,0]
```

```
5 U03:%x[1,0]
6 U04:%x[2,0]
7 U05:%x[-2,0]/%x[-1,0]/%x[0,0]
8 U06:%x[-1,0]/%x[0,0]/%x[1,0]
9 U07:%x[0,0]/%x[1,0]/%x[2,0]
10U08:%x[-1,0]/%x[0,0]
11U09:%x[0,0]/%x[1,0]
12
13# Bigram
14B
```

关于 CRF++中特征模板的说明和举例，请大家参考官方文档上的“Preparing feature templates” 这一节，而以下部分的说明拿上述日文分词数据举例。在特征模板文件中，每一行(如 U00:%x[-2,0])代表一个特征，而宏“%x[行位置,列位 置]”则代表了相对于当前指向的 token 的行偏移和列的绝对位置，以上述训练集为例，如果当前扫描到“新 k I”这一行，

```
1 每 k B
2 日 k I
3 新 k I <== 扫描到这一行，代表当前位置
4 聞 k I
5 社 k I
6 特 k B
7 別 k I
8 顧 k B
9 問 k I
10 4 n B
```

那么依据特征模板文件抽取的特征如下：

```
1 # Unigram
2 U00:%x[-2,0] ==> 每
3 U01:%x[-1,0] ==> 日
4 U02:%x[0,0] ==> 新
5 U03:%x[1,0] ==> 聞
6 U04:%x[2,0] ==> 社
7 U05:%x[-2,0]/%x[-1,0]/%x[0,0] ==> 每/日/新
8 U06:%x[-1,0]/%x[0,0]/%x[1,0] ==> 日/新/聞
9 U07:%x[0,0]/%x[1,0]/%x[2,0] ==> 新/聞/社
10U08:%x[-1,0]/%x[0,0] ==> 日/新
11U09:%x[0,0]/%x[1,0] ==> 新/聞
12
13# Bigram
14B
```

CRF++里将特征分成两种类型，一种是 Unigram 的，“U”起头，另外一种是 Bigram 的，“B”起头。对于 Unigram 的特征，假如一个特征模板是”U01:%x[-1,0]“，CRF++会自动的生成一组特征函数(func1 … funcN) 集合:

1func1 = if (output = B and feature="U01:日") return 1 else return 0
2func2 = if (output = I and feature="U01:日") return 1 else return 0
3....
4funcXX = if (output = B and feature="U01:問") return 1 else return 0
5funcXY = if (output = I and feature="U01:問") return 1 else return 0

生成的特征函数的数目 = (L * N)，其中 L 是输出的类型的个数，这里是 B，I 这两个 tag，N 是通过模板扩展出来的所有单个字符串(特征）的个数，这里指的是在扫描所有训练集的过程中找到的日文字（特征）。

而 Bigram 特征主要是当前的 token 和前面一个位置 token 的自动组合生成的 bigram 特征集合。最后需要注意的是 U01 和 U02 这些标 志位，与特征 token 组合到一起主要是区分“U01:問”和“U02:問”这类特征，虽然抽取的日文”字”特征是一样的，但是在 CRF++中这是有区别 的特征。

最后我们再来看一下执行脚本：

```
1#!/bin/sh
2../../crf_learn -f 3 -c 4.0 template train.data model
3../../crf_test -m model test.data
4
5../../crf_learn -a MIRA -f 3 template train.data model
6../../crf_test -m model test.data
7rm -f model
```

执行脚本告诉了我们如何训练一个 CRF 模型，以及如何利用这个模型来进行测试，执行这个脚本之后，对于输入的测试集，输出结果多了一列：

```
1 よ h I B
2 っ h I I
3 て h I B
4 私 k B B
5 た h B B
6 ち h I I
7 の h B B
8 世 k B B
9 代 k I I
10が h B B
```

而这一列才是模型预测的改字的标记 tag，也正是我们所需要的结果。到此为止，关于日文分词样例的介绍已经完毕，读者应该可以猜测到接下来我们会如何做中文分词吧？

和上一节利用最大熵模型进行中文分词相 似，第一步仍然是将 backoff2005 里的训练数据转化为 CRF++所需的训练数据格式，还是以微软亚洲研究院提供的中文分词语料为例，依然采用 4- tag(B(Begin，词首)，E(End，词尾)，M(Middle，词中)， S(Single,单字词))标记集，只处理 utf-8 编码文本。原始训练集./icwb2-data/training /msr_training.utf8 的形式是人工分好词的中文句子形式，如：

```
“ 人们 常 说 生活 是 一 部 教科书 ， 而 血 与 火 的 战
争 > 更 是 不可多得 的 教科书 ， 她 确实 是 名副其
实 的 ‘ 我 的 > 大学 ’ 。
1 “ 心 静 渐 知 春 似 海 ， 花 深 每 觉 影 生 香 。
2 “ 吃 屎 的 东西 ， 连 一 捆 麦 也 铡 不 动 呀 ？
3 他 “ 严格要求 自己 ， 从 一个 科举 出身 的 进士 成为 一
4 个 伟> 大 的 民主主义 者 ， 进而 成为 一 位 杰出 的 党外 共产主
5 义 战 士 ， 献身 于 崇高 的 共产主义 事业 。
6 “ 征 而 未 用 的 耕地 和 有 收益 的 土地 ， 不准 荒芜 。
7 “ 这 首先 是 个 民族 问题 ， 民族 的 感情 问题 。
8 ’ 我 扔 了 两颗 手榴弹 ， 他 一下子 出 溜 下去 。
9 “ 废除 先前 存在 的 所有制 关系 ， 并不是 共产主义 所 独具 的 特
10征 。
“ 这个 案子 从 始 至今 我们 都 没有 跟 法官 接触 过 ， 也 > 没
有 跟 原告 、 被告 接触 过 。
“ 你 只有 把 事情 做好 ， 大伙 才 服 你 。
```

这里同样提供一个脚本 make_crf_train_data.py，将这个训练语料转换为 CRF++训练用的语料格式(2 列，4-tag)：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # Author: 52nlpcn@gmail.com
4  # Copyright 2014 @ YuZhen Technology
5  #
6  # 4 tags for character tagging: B(Begin), E(End), M(Middle), S(Single)
7
8  import codecs
9  import sys
10
11def character_tagging(input_file, output_file):
12     input_data = codecs.open(input_file, 'r', 'utf-8')
13     output_data = codecs.open(output_file, 'w', 'utf-8')
14     for line in input_data.readlines():
```

```
        word_list = line.strip().split()
        for word in word_list:
            if len(word) == 1:
                output_data.write(word + "\tS\n")
            else:
                output_data.write(word[0] + "\tB\n")
                for w in word[1:len(word)-1]:
                    output_data.write(w + "\tM\n")
                output_data.write(word[len(word)-1] + "\tE\n")
        output_data.write("\n")
    input_data.close()
    output_data.close()

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print "pls use: python make_crf_train_data.py input output"
        sys.exit()
    input_file = sys.argv[1]
    output_file = sys.argv[2]
    character_tagging(input_file, output_file)
```

只需要执行“python make_crf_train_data.py ./icwb2-data/training/msr_training.utf8 msr_training.tagging4crf.utf8” 即可得到 CRF++要求的格式的训练文件 msr_training.tagging4crf.utf8，样例如下：

```
1 “ S
2 人 B
3 们 E
4 常 S
5 说 S
6 生 B
7 活 E
8 是 S
9 一 S
10部 S
11...
```

有了这份训练语料，就可以利用 crf 的训练工具 crf_learn 来训练模型了，执行如下命令即可：

```
crf_learn -f 3 -c 4.0 template msr_training.tagging4crf.utf8 crf_model
```

这次训练的时间稍微有些长，在我的 4G 内存的 mac pro 上跑了将近 700 轮，大约 2 个小时，最终训练的 crf_model 约 51M。有了模型，现在我们需要做得还是

准备一份 CRF++用的测试语料，然后 利用 CRF++的测试工具 crf_test 进行字标注。原始的测试语料是 icwb2-data/testing/msr_test.utf8 ，样例如下：

```
1 扬帆远东做与中国合作的先行
2 希腊的经济结构较特殊。
3 海运业雄踞全球之首，按吨位计占世界总数的１７％。
4 另外旅游、侨汇也是经济收入的重要组成部分，制造业规模相对较小。
5 多年来，中希贸易始终处于较低的水平，希腊几乎没有在中国投资。
6 十几年来，改革开放的中国经济高速发展，远东在崛起。
7 瓦西里斯的船只中有４０％驶向远东，每个月几乎都有两三条船停靠中国港口。
8 他感受到了中国经济发展的大潮。
9 他要与中国人合作。
10他来到中国，成为第一个访华的大船主。
```

这里我们同样提供一个 python 脚本 [make_crf_test_data.py] 对测试语料进行处理，将其转换为 CRF++要求的格式（2 列，B 作为最后一列的占位符）

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # Author: 52nlpcn@gmail.com
4  # Copyright 2014 @ YuZhen Technology
5  #
6  # 4 tags for character tagging: B(Begin), E(End), M(Middle), S(Single)
7
8  import codecs
9  import sys
10
11def character_split(input_file, output_file):
12     input_data = codecs.open(input_file, 'r', 'utf-8')
13     output_data = codecs.open(output_file, 'w', 'utf-8')
14     for line in input_data.readlines():
15          for word in line.strip():
16               word = word.strip()
17               if word:
18                    output_data.write(word + "\tB\n")
19          output_data.write("\n")
20     input_data.close()
21     output_data.close()
22
23if __name__ == '__main__':
24     if len(sys.argv) != 3:
25          print "pls use: python make_crf_test_data.py input output"
26          sys.exit()
27     input_file = sys.argv[1]
28     output_file = sys.argv[2]
```

```
29      character_split(input_file, output_file)
```

执行“python make_crf_test_data.py ./icwb2-data/testing/msr_test.utf8 msr_test4crf.utf8”即可得到可用于 CRF++测试的测试语料 msr_test4crf.utf8，样例如下：

```
1 扬 B
2 帆 B
3 远 B
4 东 B
5 做 B
6 与 B
7 中 B
8 国 B
9 合 B
10作 B
11...
```

现在执行 crf_test 即可得到字标注结果：

```
crf_test -m crf_model msr_test4crf.utf8 > msr_test4crf.tag.utf8
```

msr_test4crf.tag.utf8 即是标注结果，样例如下：

```
1 扬 B    B
2 帆 B    E
3 远 B    B
4 东 B    E
5 做 B    S
6 与 B    S
7 中 B    B
8 国 B    E
9 合 B    B
10作 B    E
11...
```

最后我们还需要一个脚本，按标注的词位信息讲这份结果再转化为分词结果，这里我们仍然提供一个转换脚本 crf_data_2_word.py ：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # Author: 52nlpcn@gmail.com
4  # Copyright 2014 @ YuZhen Technology
5  #
6  # 4 tags for character tagging: B(Begin), E(End), M(Middle), S(Single)
```

```
7
8  import codecs
9  import sys
10
11def character_2_word(input_file, output_file):
12     input_data = codecs.open(input_file, 'r', 'utf-8')
13     output_data = codecs.open(output_file, 'w', 'utf-8')
14     for line in input_data.readlines():
15          if line == "\n":
16               output_data.write("\n")
17          else:
18               char_tag_pair = line.strip().split('\t')
19               char = char_tag_pair[0]
20               tag = char_tag_pair[2]
21               if tag == 'B':
22                    output_data.write(' ' + char)
23               elif tag == 'M':
24                    output_data.write(char)
25               elif tag == 'E':
26                    output_data.write(char + ' ')
27               else: # tag == 'S'
28                    output_data.write(' ' + char + ' ')
29     input_data.close()
30     output_data.close()
31
32if __name__ == '__main__':
33     if len(sys.argv) != 3:
34          print "pls use: python crf_data_2_word.py input output"
35          sys.exit()
36     input_file = sys.argv[1]
37     output_file = sys.argv[2]
38     character_2_word(input_file, output_file)
```

只需执行“python crf_data_2_word.py msr_test4crf.tag.utf8 msr_test4crf.tag2word.utf8” 即可得到合并后的分词结果文件 msr_test4crf.tag2word.utf8，样例如下：

```
1 扬帆 远东 做 与 中国 合作 的 先行
2 希腊 的 经济 结构 较 特殊 。
3 海运 业 雄踞 全球 之 首 ， 按 吨 位 计 占 世界 总数 的 １７％ 。
4 另外 旅游 、 侨汇 也是 经济 收入 的 重要 组成部分 ， 制造业 规模 相
5 对 较小 。
6 多年来 ， 中 希 贸易 始终 处于 较低 的 水平 ， 希腊 几乎 没有 在 中
7 国 投资 。
```

8 十几年 来 ， 改革开放 的 中国 经济 高速 发展 ， 远东 在 崛起 。
9 瓦西里斯 的 船只 中 有 ４０％ 驶 向 远东 ， 每个 月 几乎 都 有 两三
10条 船 停靠 中国 港口 。
11 他 感受 到 了 中国 经济 发展 的 大潮 。
他 要 与 中国人 合作 。
他 来到 中国 ， 成为 第一个 访 华 的 大船 主 。
…

有了这个 CRF 字标注分词结果，我们就可以利用 backoff2005 的测试脚本来测一下这次分词的效果了：

./icwb2-data/scripts/score ./icwb2-data/gold/msr_training_words.utf8 ./icwb2-data/gold/msr_test_gold.utf8 msr_test4crf.tag2word.utf8 > msr_crf_segment.score

结果如下：

```
=== SUMMARY:
=== TOTAL INSERTIONS: 1412
=== TOTAL DELETIONS: 1305
=== TOTAL SUBSTITUTIONS: 2449
=== TOTAL NCHANGE: 5166
=== TOTAL TRUE WORD COUNT: 106873
=== TOTAL TEST WORD COUNT: 106980
=== TOTAL TRUE WORDS RECALL: 0.965
=== TOTAL TEST WORDS PRECISION: 0.964
=== F MEASURE: 0.964
=== OOV Rate: 0.026
=== OOV Recall Rate: 0.647
=== IV Recall Rate: 0.974
### msr_test4crf.tag2word.utf8 1412 1305 2449 5166 106873 106980 0.965 0.964 0.964 0.026 0.647 0.974
```

这次我们获得了一个准确率，召回率以及 F 值都在 96%以上的结果，相对于前面几节的测试结果，这个 CRF 字标注分词结果还相对不错。不过是不是感觉上面的步骤有些繁琐，有没有一次到位的 CRF 分词器，这里我们同样提供一个 CRF 分词脚本 crf_segmenter.py ，利用 CRF++的 python 工具包，做到一次输入，一次输出：

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # Author: 52nlpcn@gmail.com
4 # Copyright 2014 @ YuZhen Technology
5 #
6 # CRF Segmenter based character tagging:
```

```
#       4 tags for character tagging: B(Begin), E(End), M(Middle), S(Single)

import codecs
import sys

import CRFPP

def crf_segmenter(input_file, output_file, tagger):
    input_data = codecs.open(input_file, 'r', 'utf-8')
    output_data = codecs.open(output_file, 'w', 'utf-8')
    for line in input_data.readlines():
        tagger.clear()
        for word in line.strip():
            word = word.strip()
            if word:
                tagger.add((word + "\to\tB").encode('utf-8'))
        tagger.parse()
        size = tagger.size()
        xsize = tagger.xsize()
        for i in range(0, size):
            for j in range(0, xsize):
                char = tagger.x(i, j).decode('utf-8')
                tag = tagger.y2(i)
                if tag == 'B':
                    output_data.write(' ' + char)
                elif tag == 'M':
                    output_data.write(char)
                elif tag == 'E':
                    output_data.write(char + ' ')
                else: # tag == 'S'
                    output_data.write(' ' + char + ' ')
        output_data.write('\n')
    input_data.close()
    output_data.close()

if __name__ == '__main__':
    if len(sys.argv) != 4:
        print "pls use: python crf_segmenter.py model input output"
        sys.exit()
    crf_model = sys.argv[1]
    input_file = sys.argv[2]
    output_file = sys.argv[3]
    tagger = CRFPP.Tagger("-m " + crf_model)
    crf_segmenter(input_file, output_file, tagger)
```

只需执行“python crf_segmenter.py crf_model ./icwb2-data/testing/msr_test.utf8 msr_test.seg.utf8”即可得到与前面几步得到的分词结果完全一致的 CRF 分词结果：msr_test.seg.utf8 。

好了，到此为止，关于字标注中文分词的系列终于可以画上句号了，这个系列中所举的例子以及所提供的脚本都是 toy 级别的中文分词工具，距离一个真正实用的中文分词器还有很多路要走，不过既然路已经打开，欢迎大家和我们一起继续探索中文分词的奥秘。

最后再打个广告：今天是 2013 年的最后一天，这一年，我们创业了，成立了一家公司“语真科技”，2014 年，我们除了关注在线教育外，也致力于 NLP 技术的普及，推广和应用，如果您有相关的需求，欢迎和我们联系，微博 @52nlp 或者邮件联系 support@yuzhenkeji.com 都可以，最后祝大家新年顺心如意！

注：原创文章，转载请注明出处“我爱自然语言处理”：www.52nlp.cn

作者：52nlp(52nlpcn@gmail.com)
微博：http://weibo.com/52nlp
相关链接：

- 中文分词入门之字标注法 1
- 中文分词入门之字标注法 2
- 中文分词入门之字标注法 3
- 中文分词入门之字标注法 4