同济大学计算机系

中文信息处理实验报告

学　　号 **1553534**

姓　　名 李帅

专　　业 计科

授课老师 卫志华

一、分词作业介绍

完成了基于统计的分词算法（自分割分词算法）的实现，测试，以及总结。

二、实验环境

Python 3.5
Win 10 系统
第三方库：collections 导入 defaultdict，便于统计 n-gram 权重
Zhon 导入 punctuation（中文标点符号）

三、算法框架

输入：待切分的字符串 S
输出：切分后的字符串 S’

（1） 算法流程

a. 对 S 首先以罗马字符.数字，标点符号，为断点进行一次切分，插入切分标识符并将初次切分结果保存在 S’,剩余中文文本则以断点为边界形成一组字符串块 s1,s2,s3,…sk,同时令 st=s1+s2+s3+…+sk;
b. 在 st 中找到出现两次以上的 n-gram（n>2），将得到的 n-gram 项与其中出现的次数保存起来，在每个字符串块 si 中，找到仅仅在字符串块中出现的 n-gram 作为候选 n-gram 项；
c. 按照长度和出现次数递减的原则对候选 n-gram 逐步做出如下测试：如果被测试的 n-gram 的出现次数比由他形成的两个（n-1）-gram 项少，那么不采纳该 n-gram,反之，则不采纳(n-1)-gram 项，全部测试完成后，则得到全部采纳的 n-gram 项；
d. 按照步骤 c 得到的 n-gram 项对于每个字符串块 si 进行划分，插入切分标识符并同步到切分结果 S’中；
e. 如果有在所有字符块中出现两次以上的划分词条，把它加入到一个新的初步的词典中，按照长度和出现次数递减的规则类似的得到被采纳的 n-gram 项的步骤处理整个初步词典，得到新的词条和新的切分，更新 S’,如果符合条件那么重复该步骤。
f. 返回 S’

（2） 算法伪代码

Algorithm selfSegmentation *(chineseChunks)*
Input: Chinese text split into the list *chineseChunks* using roman characters, numbers and punctuation marks as break-points

```
1.  from n ← 2 to MAX_NGRAM_SIZE do
2.      for each n-gram ngram found in chineseChunks do
3.          weights(ngram) ← count(chineseChunks, ngram)
4.      loop
5.  loop
6.  for each chunk chunk in chineseChunks do
7.      if length of chunk > 2
8.          from n ← 2 to length of chunk do
9.              for each n-gram ngram found in chunk do
10.                 count ← weights(ngram)
11.                 if count > 1
12.                     candidates(n)(ngram) ← count
13.                 end if
14.             loop
15.         loop
16.         if there exist candidates
17.             from n ← size of candidates+1 to 2 do
18.                 for each candidate ncandidate in candidates(n) do
19.                     for each candidate n1candidate in candidates(n-1) do
20.                         if n1candidate found in ncandidate
21.                             if weights(n1candidate) > weights(ncandidate)
22.                                 ncandidate is unacceptable
23.                             else
24.                                 n1candidate is unacceptable
25.                             end if
26.                         end if
27.                     loop
28.                 loop
29.             loop
30.             from n ← size of candidates+1 to 2 do

31.                 for each candidate ncandidate in candidates(n) do
32.                     if ncandidate is not unacceptable
33.                         sortedInsert (acceptables, ncandidate)
34.                     end if
35.                 loop
36.             loop
37.             segmentedChunk ← chunk
38.             for each candidate candidate in acceptables do
39.                 segmentedChunk ← replace(candidate, <BLANK>candidate<BLANK>, segmentedChunk)
40.             loop
41.             chunk ← segmentedChunk
42.             lexiconEntries ← explode(<BLANK>, chunk)
43.             for each entry entry in lexiconEntries do
44.                 lexicon(entry) ← lexicon(entry)+1
45.             loop
46.         end if
47.     end if
```

```
48. loop
49. for each entry entry in lexicon do
50.     if lexicon(entry) = 1
51.         drop lexicon(entry)
52.     end if
53. loop
54. sort lexicon by descending length and count order
55. for each chunk chunk in chineseChunks do
56.     segmentedChunk ← chunk
57.     for each entry entry in lexicon do
58.         segmentedChunk ← replace(entry, <BLANK>entry<BLANK>, segmentedChunk)
59.     loop
60.     chunk ← segmentedChunk
61.     lexiconEntries ← explode(<BLANK>, chunk)
62.     for each entry entry in lexiconEntries do
63.         lexicon(entry) ← lexicon(entry)+1
64.     loop
65. loop
66. for each entry entry in lexicon do
67.     if lexicon(entry) = 1
68.         drop lexicon(entry)
69.     end if
70. loop
71. sort lexicon by descending length and count order
72. return lexicon
```

四、函数说明

（1） 类成员变量介绍

本实验程序是 python 语言，在 self_segment.py 文件中创建一个 self_seg 类，进行对测试文件的导入，相关字符串的提取，n-gram 的计算，切分后的结果输出等等功能。

```
class self_seg(object):
    def __init__(self):
        self.inputdata=str()
        self.outputdata=str()
        #预先设置n=3,4,5,6
        self.ngram=defaultdict(int)
        self.sp=[]
        self.s=[]
        self.s_index=[]
        self.st=str()
        self.roman=["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"]
        self.shuzi=['0','1','2','3','4','5','6','7','8','9']
        self.biaodian=[x for x in punctuation]
        self.dictionary = []
```

Self.inputdata: 字符串型，输入的文本字符串

Self.outputdata: 字符串型，输出的切割划分后的字符串

Self.ngram: 字典型 ，统计 n-gram 的次数（权值）,其中 key 是 n-gram 字符串，value 是次数（权值）

Self.sp : 数组型，记录初次划分时候的特殊字符（罗马字符，标点符号，数字）数组

Self.s : 数组型，记录初次断点划分之后的字符串块

Self.s_index: 数组型，记录初次划分时的 si 在输入字符串的初始位置下标

Self.st: 字符串，self.st=∑self.si,也就是把所有 si 拼接起来形成的 st

Self.roman: 数组型，罗马符号数组
Self.shuzi: 数组型，数字数组
Self.biaodian: 数组型，所有中文标点符号的集合
Self.dictionary: 数组型，第五步形成的词典

（2） 函数说明

本程序分成一个 self_seg 类 四个成员函数：

```
import string
from collections import defaultdict
from zhon.hanzi import punctuation
class self_seg(object):
    def __init__(self):...

    def load_data(self,f):...
    def self_segmentation(self):...
    def output(self):...
```

a.__init__函数：

类的初始函数，初始成员变量。

b. load_data 函数：

导入输入的 txt 文件，以 utf-8 编码打开并以字符串形式读入 self.inputdata 中。

c. self_segmention 函数

算法实现的主体函数。分为以下几个模块：

（1） 特殊字符分割字符串

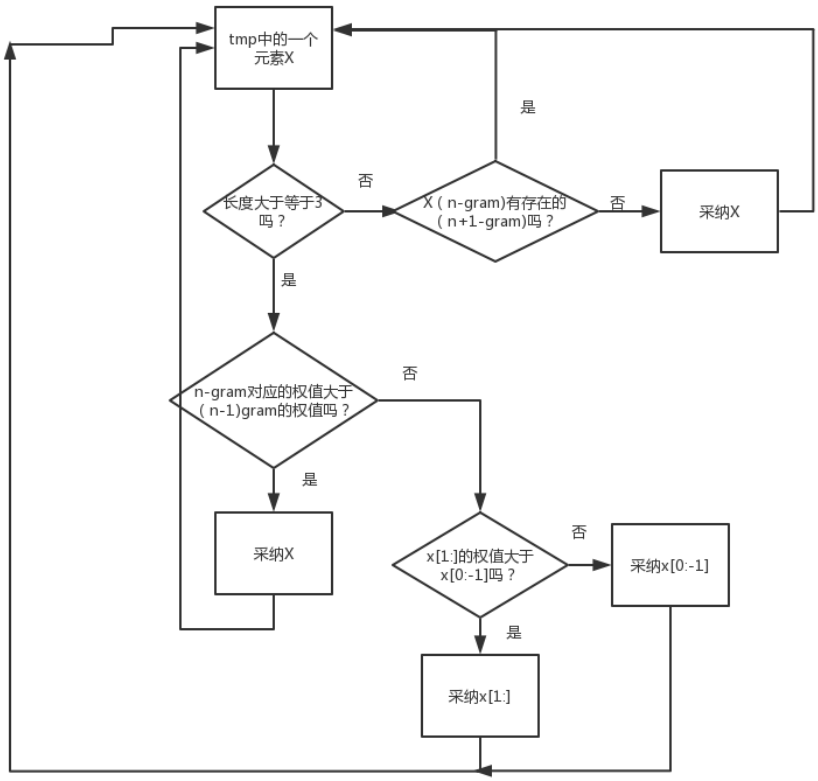
一遍扫描字符串，以三类特殊字符（罗马字符，数字，标点符号）作为分割标志，切割出 self.s(纯中文字符串块数组)和 self.sp(特殊字符串块数组)。

（2） 统计 n-gram 权值

将第一步扫描出的 self.s 字符串数组拼接起来，一边扫描每一个字符串并统计从这个字符开始长度为 2，3，4，5 长度的 n-gram 出现的次数。统计一遍完毕将次数大于 2 的 n-gram 提取出来作为备选 n-gram 放在 self,ngram 数组中。

（3） 测试备选 n-gram

遍历每一个 self.s 数组的 si,对于**字符串块 si,**用临时变量 **tmp** 存储 n-gram 中出现在该 si 中的 n-gram。然后**测试 n-gram** 过程，见如下流程图：



（4） 用测试后的 n-gram 划分字符串

用得到的测试采纳的 n-gram 划分字符串块，分隔符是'\'，同时去掉多余的‘/‘符号。

（5） 统计得到词典，更新划分字符串

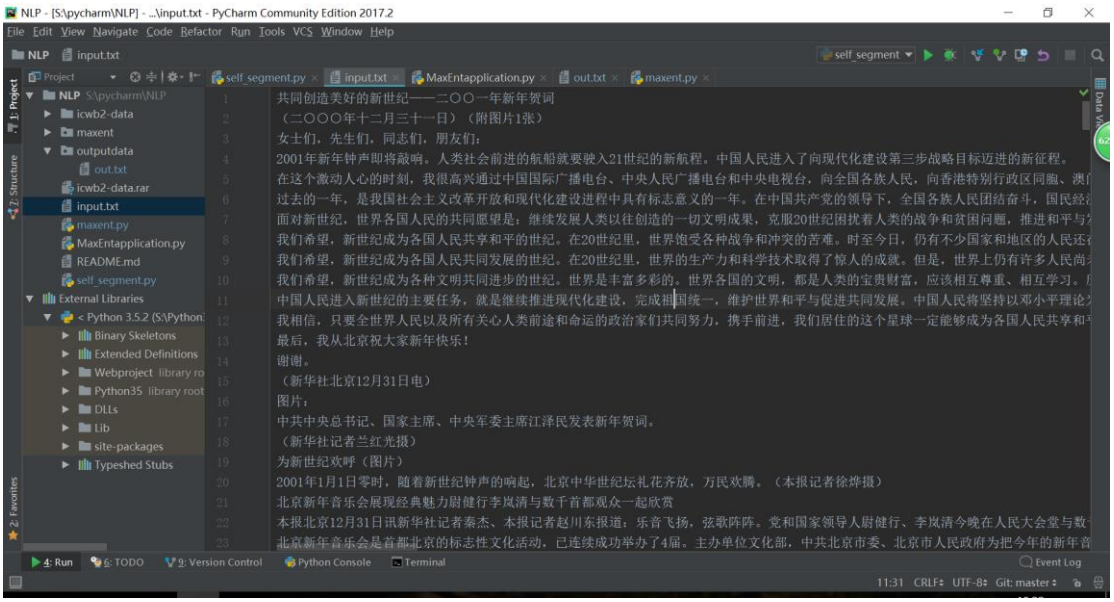
划分一遍之后，如果有在所有字符块中出现两次以上的划分词条，把它加入到一个新的初步的词典中，按照长度和出现次数递减的规则类似的得到被采纳的 n-gram 项的步骤处理整个初步词典，得到新的词条和新的切分，更新 S’

d. output 函数

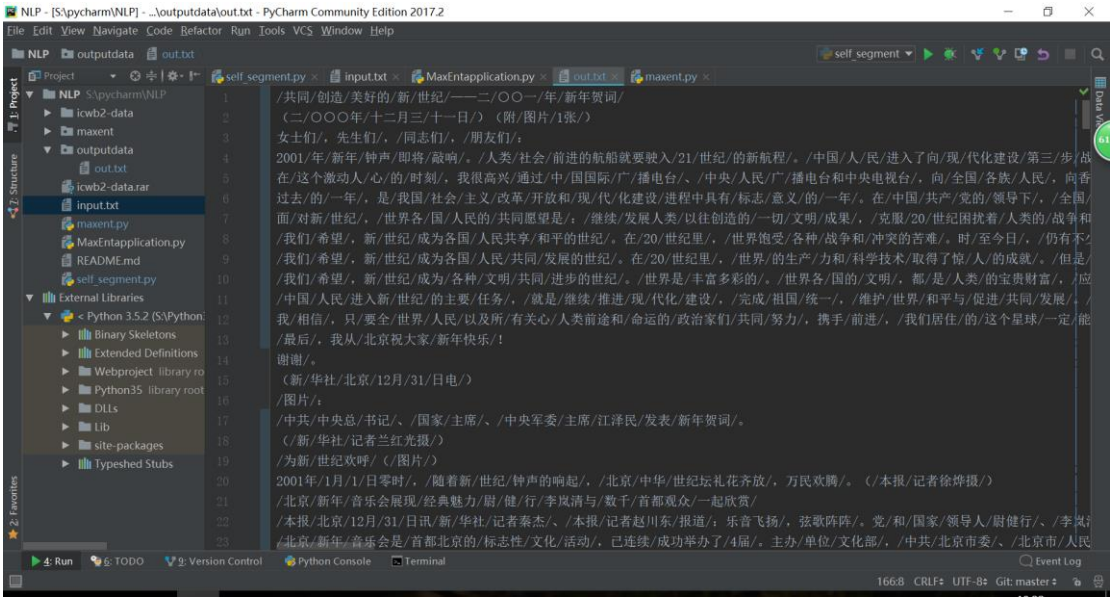
将划分后的字符串块和特殊字符串数组拼接在一起，存入 self.outputdata，作为输出 S’,存入 out.txt 文件中。

五、实验结果截图

（1） 输入文本



（2） 输出文本（切割符号是‘/‘）



六、总结

可以看到，自分割分词是一种无词典分词方法，除了待处理的文本之外无需外部的词典或语料库。也没有未登录词和歧义词的问题。

它既可以得到分词结果，又可以得到基于处理后文本的分词词典。非常适合处理词短语分词 ，但是它不是对所有的文本都能得到有效的结果。生成的分词词典虽然相对准确，但有时也会出现复杂的句子和不能作为词的词条。

同时在本算法中，也可以看到几点不足的地方，第一：对于测试 n-gram，三个文本进行比较，原算法却只给了两种对应的处理方式，这里我按照统计与交叉验证的思想添加了其他几种情况的处理方式，但是效果还是不佳（比如原文切割出很多不对的单字）；第二，对于新形成的字典，如果在已经切割的基础上再进行切割，会带来更多的不准确，但是这个算法没有撤回操作，所以一旦是错误的划分，后面基于新的更准确的词典划分时，应该提前撤回

对应的不合适的划分。

七、源码与测试文件

自分割算法实现代码：self_segment.py
测试文本：input.txt
输出问题：out.txt
有关的代码，文件，其他的最大熵算法，CRF 算法，我已经更新在我的 github 地址：https://github.com/TomHacker/NLP-Natural-Language-Processing-
谢谢。