

哈尔滨工业大学 2020 年秋季学期

计算学部本科生
“中文信息处理”课
大作业

作 业 题 目： 中文自动分词实验

姓 名： 梅智敏

学 号： 1183710118

学 生 专 业： 软件工程

任 课 教 师： 刘秉权

2020 年 10 月 11 日

中文自动分词实验

一、实验要求和目的

- 使用任意分词方法实现汉语自动分词
- 给出至少1000个句子的分词结果（以附件形式）；
- 计算出分词结果的正确率，并给出计算依据
- 用实例说明所用分词方法分别对“交叉歧义”和“组合歧义”的处理能力；

二、实验环境

windows 10, python 3.7.4

三、程序主要算法

最大正向匹配算法（FMM）

所谓最大匹配，指的是以词典为依据，对于输入的一段文本从左至右、以贪心的方式切分出当前位置上长度最大的词。单词的颗粒度越大，所能表示的含义越确切。

- 所谓词典，就是包含一些词语的集合，以此作为划分中文句子的依据
- 首先记词典中最长词语的长度为 $maxlength$

以下图为例：

0	1	2	3	4	5	6	7	8	9
我	毕	业	于	北	京	邮	电	大	学

pos	remain characters	start character	max matching
0	我毕业于北京邮电大学	我	我
1	毕业于北京邮电大学	毕	毕业
3	于北京邮电大学	于	于
4	北京邮电大学	北	北京邮电大学

从文本的最左端开始，从 $maxlength$ 长度开始截取子字符串 $word$ 并在词典中查找。若找到，则切分开，若找不到，则将 $len(word) - 1$ ，从而继续在 $dict$ 中查询。直到 $len(word) = 1$ ，此时不管是否能不能在词典中找到它，也必须将它切分开。

利用一个循环即可实现：

```
my_list = []
rowLength = len(line)
while rowLength > 0:
    # 从最大词语长度开始截取字符并在词典中查找
    tryword = line[0:maxlength]
    while tryword not in dictionary:
```

```

        # 长度降到1时，必须切分，故跳出循环
        if len(tryword) == 1:
            break
        # 把目标字符串长度-1，继续在词典中查找
        tryword = tryword[0:len(tryword) - 1]
    my_list.append(tryword)
    line = line[len(tryword):]
    rowLength = len(line)

```

四、实验过程

生成词典

利用从**中文分词词库**找到的内容，截取部分放在一个txt文件中，再利用python来读取文件内容，并存入一个集合即可

```

def createDict(dictFile):
    """
    构建词典
    :param dictFile: 词典文本路径
    :return: 包含所有词语的集合
    """
    lexicon = set()
    f = open(dictFile, 'r', encoding='utf-8')
    for line in f.readlines():
        lexicon.add(line.strip('\n'))
    f.close()
    return lexicon

```

进行FMM

自己准备一个语料txt文件，每一行都包含一个中文句子。作为FMM算法的inFile，将每一行的句子分别进行中文分词，并新建一个myAnswer.txt文件作为FMM算法的outFile，同样是每行一个句子，与inFile相对应。

```

def FMM(dictionary, inPath, outPath, max_length=7):
    """
    FMM算法进行中文分词
    :param dictionary: 词典
    :param inPath: 预料文件路径
    :param outPath: 输出文件路径
    :param max_length: 词典中最长词的的长度
    :return: 分词后的结果，写入输出文件中
    """
    # 读取语料txt文件
    with open(inPath, 'r', encoding='utf-8', ) as f:
        lines = f.readlines()
    # 选定输出结果的txt文件
    result = open(outPath, 'w', encoding='utf-8', )
    # 分别对每行进行正向最大匹配处理
    for line in lines:
        my_list = []

```

```

rowLength = len(line)
while rowLength > 0:
    # 从最大词语长度开始截取字符并在词典中查找
    tryword = line[0:max_length]
    while tryword not in dictionary:
        # 长度降到1时，必须切分，故跳出循环
        if len(tryword) == 1:
            break
        # 把目标字符串长度-1，继续在词典中查找
        tryword = tryword[0:len(tryword) - 1]
    my_list.append(tryword)
    line = line[len(tryword):]
    rowLength = len(line)
# 将分词结果写入生成文件
for word in my_list:
    if word == '\n':
        result.write('\n')
    else:
        result.write(word + " ")

result.close()

```

计算分词的准确率

这部分需要一个测试文件*testSource.txt*作为FMM算法的*inFile*，以及对应的*outFile*，还需要一个代表正确答案的文件*trueAnswer.txt*。

记录*testSource.txt*的总句子数目*all*，再逐行比较*outFile*和*trueAnswer.txt*，记录相同的行数，即分词正确的句子数目*right*。

二者相除即可求出分词准确率。

```

def accuracy(myAnswer, trueAnswer):
    r1 = open(myAnswer, 'r', encoding='utf-8')
    # 记录总的行数，即句子数
    r1_len = len(r1.readlines())
    # 循环计数器
    count = 1
    # 分词正确的句子数目
    num = 0
    while True:
        if count > r1_len:
            break
        # 截取我的答案的对应行内容
        content1 = linecache.getline(myAnswer, count)
        # 截取正确答案的对应行内容
        content2 = linecache.getline(trueAnswer, count)
        # 若二者相同则将正确分词的句子数目+1
        if content1.strip() == content2.strip():
            num += 1
        count += 1

    result = num / r1_len
    return result

```

获取正确答案文件

在前面的计算分词准确率的过程中，需要一个正确答案`txt`文件。那么如何获取这样的文件呢？

我使用了`python`中的`jieba`库，虽然老师说过不允许利用它来进行分词，但是我在这里仅仅是用它来获取正确分词结果文件`trueAnswer.txt`，从而让我自己可以计算出自己编写的`FMM`算法的分词准确率。

```
def getTrueAnswer(inPath, outPath):
    fR = open(inPath, 'r', encoding='UTF-8')

    sent = fR.read()
    sent_list = jieba.cut(sent)

    fw = open(outPath, 'w', encoding='UTF-8')
    fw.write(' '.join(sent_list))

    fR.close()
    fw.close()
```

五、实验结果

部分分词结果展示

`testSource.txt`中（分词前）：

```
165 我们经常听广播、看电视，耳闻目睹一些有志女青年冲破世俗的桎梏，到大城市学技术锻炼自己，终于事业有成。
166 我们不仅羡慕她们，而且自己也很想去试试。|
```

`myAnswer.txt`（分词后）：

```
165 我们 经常 听 广播 、 看电视 ， 耳闻目睹 一些 有志 女 青年 冲破 世俗 的 桎梏 ， 到 大城市 学 技术 锻炼 自己 ， 终于 事业有成 。
166 我们 不仅 羡慕 她们 ， 而且 自己 也 很 想去 试试 。 |
```

可以发现，对于这些没有明显歧义的句子，`FMM`算法还是挺有效的。

分词准确率

将我自己编写的`FMM`算法求得的结果和用`jieba`库求出的结果对比，并以`jieba`库的结果为标准答案，可以求得分词准确率

$$\alpha = 0.4710163$$

```
mian x
E:\Anaconda3\python.exe D:/PyCharmWorkspace/NLP/lab1/mian.py
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\HPDC0006\AppData\Local\Temp\jieba.cache
Loading model cost 0.631 seconds.
Prefix dict has been built successfully.
分词准确率为：0.4710163111668758

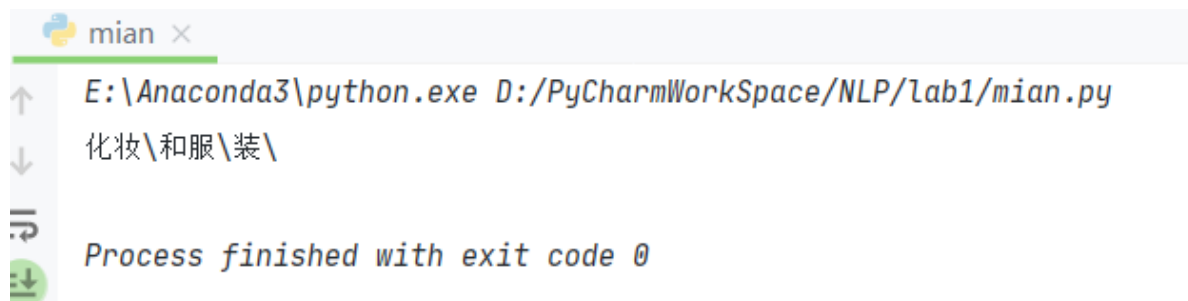
Process finished with exit code 0
```

面对交叉歧义

歧义是指同样的一句话，可能有两种或者更多的切分方法。例如：表面的，因为“表面”和“面的”都是词，那么这个短语就可以分成“表面 的”和“表 面的”。这种称为交叉歧义。

下面输入“化妆和服装”来测试

```
10 myFMM.simpleFMM(dictionary, '化妆和服装')
```



```
mian x
E:\Anaconda3\python.exe D:/PyCharmWorkspace/NLP/lab1/mian.py
化妆\和服\装\
Process finished with exit code 0
```

“化妆和服装”可以分成“**化妆\和\服装**”或者“**化妆\和服\装**”。由于没有人的知识去理解，计算机很难知道到底哪个方案正确，只能根据在词典中寻找的先后顺序来给定一个输出。

面对组合歧义

在句子“将军任命了一名中将”中，“中将”是一个词，且能够在词典中查询到，故将它切分开

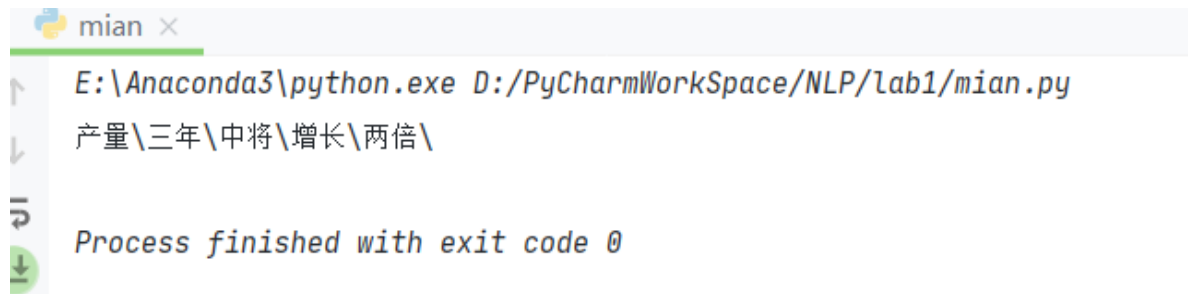
```
10 myFMM.simpleFMM(dictionary, '将军任命了一名中将')
```



```
mian x
E:\Anaconda3\python.exe D:/PyCharmWorkspace/NLP/lab1/mian.py
将军\任命\了\一名\中将\
Process finished with exit code 0
```

但在句子“产量三年中将增长两倍”中，“中将”就不再是词。从我们人为理解此时上不应该把“中将”切分开，但是使用FMM算法的话，便无法做到这一点，它只能机械地依据词典查询来切分。

```
10 myFMM.simpleFMM(dictionary, '产量三年中将增长两倍')
```



```
mian x
E:\Anaconda3\python.exe D:/PyCharmWorkspace/NLP/lab1/mian.py
产量\三年\中将\增长\两倍\
Process finished with exit code 0
```

六、实验结论

- FMM算法原理简单，实现起来也不复杂
- FMM算法对于一些没有歧义的句子，分词效果还是不错的
- FMM算法难以应对“交叉歧义”和“组合歧义”

七、源代码

myFMM.py

```
def createDict(dictFile):
    """
    构建词典
    :param dictFile: 词典文本路径
    :return: 包含所有词语的集合
    """
    lexicon = set()
    f = open(dictFile, 'r', encoding='utf-8')
    for line in f.readlines():
        lexicon.add(line.strip('\n'))
    f.close()
    return lexicon

def FMM(dictionary, inPath, outPath, max_length=7):
    """
    FMM算法进行中文分词
    :param dictionary: 词典
    :param inPath: 预料文件路径
    :param outPath: 输出文件路径
    :param max_length: 词典中最长词的长度
    :return: 分词后的结果，写入输出文件中
    """
    # 读取语料txt文件
    with open(inPath, 'r', encoding='utf-8', ) as f:
        lines = f.readlines()
    # 选定输出结果的txt文件
    result = open(outPath, 'w', encoding='utf-8', )
    # 分别对每行进行正向最大匹配处理
    for line in lines:
        my_list = []
        rowLength = len(line)
        while rowLength > 0:
            # 从最大词语长度开始截取字符并在词典中查找
            tryword = line[0:max_length]
            while tryword not in dictionary:
                if len(tryword) == 1:
                    break
                tryword = tryword[0:len(tryword) - 1]
            my_list.append(tryword)
            line = line[len(tryword):]
            rowLength = len(line)
        # 将分词结果写入生成文件
        for word in my_list:
            if word == '\n':
                result.write('\n')
            else:
                result.write(word + " ")
        result.close()

def simpleFMM(dictionary, input, max_length = 7):
```

```

output = ''
my_list = []
rowLength = len(input)

while rowLength > 0:
    # 从最大词语长度开始截取字符并在词典中查找
    tryWord = input[0:max_length]
    while tryWord not in dictionary:
        if len(tryWord) == 1:
            break
        tryWord = tryWord[0:len(tryWord) - 1]
    my_list.append(tryWord)
    input = input[len(tryWord):]
    rowLength = len(input)
    # 将分词结果写入生成文件
for word in my_list:
    output = output + word + "\\ "

print(output)

```

myEvaluate.py

```

# -*- coding: utf-8 -*-
from __future__ import division
import linecache
import jieba

def getTrueAnswer(inPath, outPath):
    fR = open(inPath, 'r', encoding='UTF-8')

    sent = fR.read()
    sent_list = jieba.cut(sent)

    fw = open(outPath, 'w', encoding='UTF-8')
    fw.write(' '.join(sent_list))

    fR.close()
    fw.close()

def accuracy(myAnswer, trueAnswer):
    r1 = open(myAnswer, 'r', encoding='utf-8')
    # 记录总的行数，即句子数
    r1_len = len(r1.readlines())
    # 循环计数器
    count = 1
    # 分词正确的句子数目
    num = 0
    while True:
        if count > r1_len:
            break
        # 截取我的答案的对应行内容
        content1 = linecache.getline(myAnswer, count)
        # 截取正确答案的对应行内容

```



```
content2 = linecache.getline(trueAnswer, count)
# 若二者相同则将正确分词的句子数目+1
if content1.strip() == content2.strip():
    num += 1
count += 1

result = num / r1_len
return result
```

main.py

```
import myFMM
import myEvaluate as eva

dictionary = myFMM.createDict('text/dict.txt')
myFMM.FMM(dictionary, 'text/testSource.txt', 'text/myAnswer.txt')
eva.getTrueAnswer('text/testSource.txt', 'text/trueAnswer.txt')

rate = eva.accuracy('text/myAnswer.txt', 'text/trueAnswer.txt')
print('分词准确率为: '+str(rate))
myFMM.simpleFMM(dictionary, '将军任命了一名中将')
```