实验报告

一、**实验目的**

掌握最大匹配算法，实现分词和算法评价

二、**实验任务**

使用最大匹配算法、字典文件（corpus.dict.txt），对语料（corpus.sentence.txt）进行分词

--将分词的结果输出到文件 corpus.out.txt 中；

--对比 corpus.answer.txt 和 corpus.out.txt，给出算法的 P/R/F 指标

输出：一个 corpus.out.txt 文件（格式参照 corpus.answer.txt）

P/R/F 指标(格式类似于：Precision = 36 / 100 = 36.00%)

三、**使用环境**

jupyter notebook

python3.7

四、**实验过程**

1.读入词典

把 corpus.dict.txt 的所有词语都存放入列表 wordlist 中

```
wordlist=[]                                          #词典
f=open('corpus.dict.txt','r', encoding='utf-8')
for eachline in f:
    nlen=len(eachline)-1;
    wordlist.append(eachline[:nlen])
f.close()
```

2.最大匹配算法核心部分

读入材料文件：

```
#读入文件
source_text=open('corpus.sentence.txt','r', encoding='utf-8')
```

列表用于存放分词结果，note 是源文本的每一行

对于每一行，head 作为一个指针遍历，i 是词的最大长度，设为 10

首先从最大长度开始匹配，如果匹配不成功，减小词的长度，直到只剩一个字，存放到 won 列表；匹配成功将词语存放到列表，head 移动到该词语的后一个字

```
won=[]

for note in source_text:
    head=0
    i=10
    while (head <= len(note)):
        if (head>=(len(note)-i)):
            i=len(note)-head
        for p in range(i):
            rear=head+i-p
            flag=0
            for each in wordlist:
                if (note[head:rear]==each):
                    #print (each)
                    won.append(each)
                    head=head+len(each)
                    flag=1
                    break
            if (flag==1):
                break
        if (flag==0):
            won.append(note[head:head+1])
            head=head+1
    #won.append('\n')
```

将列表存储的结果存放到 corpus.output.txt

```
file = open("corpus.output.txt", "a", encoding='utf-8')
for i in range(len(won)):
    s = str(won[i])
    if(s!="\n"):
        s = s+' '
    file.write(s)
file.close()
```

3.分词算法评价

读取文件，将 corpus.answer.txt 和 corpus.output.txt 的内容的每一行分别读入列表

```
#读入文件
text1 = open('corpus.answer.txt','r', encoding='utf-8')
text2 = open("corpus.output.txt", 'r', encoding='utf-8')
file = open("corpus.judge.txt", "a", encoding='utf-8')

wordlist1 = []
wordlist2 = []
for eachline in text1:
    wordlist1.append(eachline)

for each in text2:
    wordlist2.append(each)
```

对两个句子去掉空格，统计它们的词的个数

双重循环寻找匹配正确的词，注意这里外层循环是自己的输出，内层循环从外层循环的起始位置开始找。

Correct 记录匹配成功的单词个数，Precision 和 Recall 由公式计算得到。

结果写在 corpus.judge.txt 文件中

```
for i in range(len(wordlist1)):
    s1=str(wordlist1[i])
    #print(s1)
    s2=str(wordlist2[i])
    #print(s2)
    s1_list = s1.split()
    s2_list = s2.split()
    #print(s1_list[0])
    #print(s2_list[2])
    Correct=0
    for j in range(len(s2_list)):
        for k in range(j,len(s1_list)):
            if(s2_list[j]==s1_list[k]):
                Correct=Correct+1
                break
    Precision = Correct/len(s1_list)
    Recall = Correct/len(s2_list)
    F = (Precision) * (Recall) * 2 / (Precision + Recall)
    s = "Precision:"+str(Precision)+" "+"Recall:"+str(Recall)+" "+"F:"+str(F)+"Correct:"+str(Correct)+"l1:"+str(len(s1_list))+"l2:"+str(len(s2_
    #print(Precision, end=' ')
    #print(Recall, end=' ')
    #print(F)
    file.write(s)
```

五、总结

代码最开始有一些小 bug，与换行符的处理、python 中空格的处理有关

学习了最大匹配算法

学习了传统的分词评价指标