实　　验　　报　　告

学　　号	20020006107	姓　　名	王义钧	专业班级	通信工程
课程名称	自然语言处理			学期	2022 年 春 季学期
任课教师	仲国强	完成日期	2022/6/3	上机课时间	2022/5/31
实 验 名 称	NLP 实验四：汉语分词				

一、 实验目的

学习理解汉语分词中最大匹配算法与最短路径法的原理，并通过编程实现两种算法的功能，完成对给定句子的汉语分词，输出对应的评价指标值。

二、 实验内容：

利用人民日报语料库或自己构建的语料库（30 词以上）作为词典，任选五个句子，并基于正向最大匹配算法和最短路径法分别对这五个句子进行分词，并分别计算分词结果的正确率，召回率和 F-测度值。输出句子，基于两种算法的分词结果和其对应的评价指标值。

三、 实验过程：

1、构造正向最大匹配算法

实现思路：首先规定分配字典中最大字符串长度，这里我根据测试数据集中最长的词语“邓小平理论”为 5 个字，规定 max_length = 5。从字符串的第一个字开始遍历，每次从当前位置往后 max_length 位查找，若无法与字典中相匹配，则继续递减，直到可以在字典中查找到为止。注意的是，若一直检索到单字还无法匹配，则说明字典中无匹配项，保留该单字并 break 跳出循环。

每一次找到匹配项时，加入队列的同时加入’/’用于区分

同时，在调用正向最大匹配算法前，应先对测试句子进行一部分修饰，比如将句子中的标点符号先替换为空格等。

代码实现：

```
ans_forward = []  #存放正向匹配的结果
max_length = 5    #分词字典中最大长度字符串的长度

def FowardMatch(s_test, ori_list):
    ans_forward = []
    tmp = []
    len_row = len(s_test)       #len_orw 为当前为划分句子的长度
    while len_row > 0:          #当前待划分句子长度为0 时，结束划分
        divide = s_test[0:max_length]           #从前向后截取长度为max_length 的字符串
        while divide not in ori_list:           #当前截取的字符串不在分词字典中，则进循环
            if len(divide) == 1:                #当前截取的字符串长度为1 时，说明分词字典无匹配项
```

```
                break                                   #直接保留当前的一个字
            divide = divide[0:len(divide) - 1]  #当前截取的字符串长度减一
        ans_forward.append(divide)                  #记录下当前截取的字符串
        tmp.append(divide)
        s_test = s_test[len(divide):]               #截取未分词的句子
        len_row = len(s_test)                       #记录未分词的句子的长度
        if(len_row) :                               #将划分的词语用/隔开
            ans_forward.append("/")
    str1 = "".join(ans_forward)
    # str = "".join(tmp)
    print("\'正向最大匹配\'的分词结果为: ", str1)
    return tmp
```

2、构造最短路径法

对于最短路径法，首先应构造字典树，将每个词语按照空格切分后，将其存入 word_dict 中。

```
def load_data():
    with open('D:/NLP/lab4/metadata.txt', encoding='utf-8') as fr :
        for line in fr:
            words = line.split(" ")
            for word in words:
                if word in stop_words:
                    continue
                word_dict[word] = word_dict.get(word, 0) + 1
```

根据所给的字典，将测试句子构造成有向无环图。将每个字与后面所有的字进行搭配，并在字典树中查找，将所有可能的搭配情况放到 list 容器 tmp 中。后续在构造最短路径时，即按照该列表。

```
def build_dag(sentence):
    dag = {}
    for start in range(len(sentence)):
        tmp = []
        for stop in range(start + 1, len(sentence) + 1):
            fragment = sentence[start:stop]
            # use tf_idf?
            num = word_dict.get(fragment, 0)
            if num > 0 :
                tmp.append((stop, num))
        dag[start] = tmp
    if num == 0 :
        tmp.append((len(sentence), num))
    #print(dag)
```

```
    return dag
```

上述 build_dag 函数构造出来的数组，tmp[i]表示 i 字与剩下哪些字可以配对，其权值的大小。比如{1：[2,10],[3,3]}表示第二个字（i+1）可以与第三个字相连，其权值为 10；与第四个字相连，其权值为 3。所以在构造最短路径时，只需要让每个字均与最远距离的那个字相连即可，即每一次查询数组的第一个数值的最大值，将那个字放入队列，然后将指针移到那个字上，继续查找下一个字即可。

```
def predict(sentence):
    wordList = []
    Len = len(sentence)
    route = []
    dag = build_dag(sentence)  # {i: (stop, num)}
    i = 0
    while i < len(sentence):
        end = max(dag[i], key=lambda x: x[0])[0]
        wordList.append(sentence[i:end])
        i = end
    return wordList
```

将最短路径算法的返回值构造为汉字分词字符串

```
def test(s):
    cases = []
    ans = []
    tmp = []
    # cases = [
    #     "党中央必须坚持全心全意为人民服务的宗旨"
    # ]

    cases.append(s)
    for case in cases:
        result = predict(case)
        lenth = len(result)
        for word in result:
            ans.append(word)
            tmp.append(word)
            lenth = lenth - 1
            if lenth != 0 :
                ans.append("/")

    str1 = "".join(ans)                    #list 转换为string 类型
    print("\'最短路径法\'的分词结果为: ", str1)
    return tmp
```

3、计算正确率，召回率，F-测试度。

这里，我构造出五组句子，每组分别包含一个待分词句子，一个正确分词句子。

```
    s_test1 = "党中央必须坚持全心全意为人民服务的宗旨"
    s_test1_ac = "党  中央  必须  坚持  全心全意  为  人民  服务  的  宗旨"
    s_test2 = "以经济建设为中心是邓小平理论的基本观点"
    s_test2_ac = "以  经济  建设  为  中心  是  邓小平理论  的  基本  观点"
    s_test3 = "坚定不移建设有中国特色社会主义道路"
    s_test3_ac = "坚定不移  建设  有  中国  特色  社会主义  道路"
    s_test4 = "精神文明建设和民主法制建设"
    s_test4_ac = "精神  文明  建设  和  民主  法制  建设"
    s_test5 = "他只会诊断一般的疾病"
    s_test5_ac = "他  只会  诊断  一般  的  疾病"
```

以第一组为例，在经过正向最大匹配算法后，返回以分词句子 Forward_test1，在经过最短路径算法后，返回分词 Shortest_test1。这两个都是 list 类型，然后调用 acc 函数，分别计算其评价指标。

在 acc 函数中，传入的参数是 s_test 和 s_test_ac，一个代表待测句子，一个代表正确句子。对这两个句子均进行相同的操作，即将分词区间找出来。比如句子党 中央 必须 坚持 全心全意 为 人民 服务 的 宗旨；其分词区间为[0,1],[2,4],[5,7],[8,11],[12,13],[14,16],[17,19],[20,21],[22,24]。在对其他句子操作完，同样再得到其分词区间。

这时，计算评价指标中正确的个数 n，即看测试句子经分词后，其结果中有多少个词语与正确分词中的相重合，仅需对比两者的区间有多少相同的即可；计算测试结果中系统的总输出 N，即统计测试句子分词后产生了多少组词语；计算测试结果中正确答案的个数，即统计正确分词句子中有多少组词语。

代码实现：

```
def acc(s_test, s_test_ac) :
    Test = []
    Test_gold = []

    fr_test = []
    fr_test.append(s_test)

    fr_test_gold = []
    fr_test_gold.append(s_test_ac)


    for x in fr_test:
        # print(x)
        # result = predict(x)
        data = []
        j = 0
        # x = x.split()
        for s in x[:-1:1]:
            word = [j, j + len(s) - 1]
```

```
            data.append(word)
            j += len(s)
        # print(data)
        Test.append(data)

    for x in fr_test_gold:
        # print(x)
        # result = predict(x)
        data = []
        j = 0
        x = x.split()
        for s in x[:-1:1]:
            word = [j, j + len(s) - 1]
            data.append(word)
            j += len(s)
        # print(data)
        Test_gold.append(data)

    # print(Test)
    # print(Test_gold)
    Test_num = 0
    Test_gold_num = 0
    right_num = 0

    for i in range(len(Test_gold)):
        Test_num += len(Test[i])
        Test_num += 1
        Test_gold_num += len(Test_gold[i])
        Test_gold_num += 1
        right_num += len([x for x in Test[i] if x in Test_gold[i]])
        right_num += 1

    Precision = right_num / Test_num
    Recall = right_num / Test_gold_num
    Fm = ( 2 * Precision * Recall ) / ( Precision + Recall )
    print("正确率", Precision)
    print("召回率", Recall)
    print("F-测度值", Fm)
```

四、 结果展示

```
PS C:\Users\王义钧> python -u "d:\NLP\lab4\WYJNLP04.py"
Sentece1:
党中央必须坚持全心全意为人民服务的宗旨
正确分词
党  中央  必须  坚持  全心全意  为  人民  服务  的  宗旨
'正向最大匹配'的分词结果为： 党中央/必须/坚持/全心全意/为/人民/服务/的/宗旨
'最短路径法'的分词结果为： 党中央/必须/坚持/全心全意/为/人民/服务/的/宗旨
正向匹配算法的评价指标值
正确率 0.8888888888888888
召回率 0.8
F-测度值 0.8421052631578948

最短路径算法的评价指标值
正确率 0.8888888888888888
召回率 0.8
F-测度值 0.8421052631578948

Sentece2:
以经济建设为中心是邓小平理论的基本观点
正确分词
以  经济  建设  为  中心  是  邓小平理论  的  基本  观点
'正向最大匹配'的分词结果为： 以/经济建设/为/中心/是/邓小平理论/的/基本/观点
'最短路径法'的分词结果为： 以/经济建设/为/中心/是/邓小平理论/的/基本/观点
正向匹配算法的评价指标值
正确率 0.8888888888888888
召回率 0.8
F-测度值 0.8421052631578948

最短路径算法的评价指标值
正确率 0.8888888888888888
召回率 0.8
F-测度值 0.8421052631578948

Sentece3:
坚定不移建设有中国特色社会主义道路
正确分词
坚定不移  建设  有  中国  特色  社会主义  道路
'正向最大匹配'的分词结果为： 坚定不移/建设/有/中国/特色/社会主义/道路
'最短路径法'的分词结果为： 坚定不移/建设/有/中国/特色/社会主义/道路
正向匹配算法的评价指标值
正确率 1.0
召回率 1.0
F-测度值 1.0
```

```
Sentece4:
精神文明建设和民主法制建设
正确分词
精神  文明  建设  和  民主  法制  建设
'正向最大匹配'的分词结果为:  精神文明/建设/和/民主/法制/建设
'最短路径法'的分词结果为:  精神文明/建设/和/民主/法制/建设
正向匹配算法的评价指标值
正确率 0.8333333333333334
召回率 0.7142857142857143
F-测度值 0.7692307692307692

最短路径算法的评价指标值
正确率 0.8333333333333334
召回率 0.7142857142857143
F-测度值 0.7692307692307692

Sentece5:
他只会诊断一般的疾病
正确分词
他  只会  诊断  一般  的  疾病
'正向最大匹配'的分词结果为:  他/只会/诊断/一般/的/疾病
'最短路径法'的分词结果为:  他/只会/诊断/一般/的/疾病
正向匹配算法的评价指标值
正确率 1.0
召回率 1.0
F-测度值 1.0

最短路径算法的评价指标值
正确率 1.0
召回率 1.0
F-测度值 1.0
```

出现正确率和召回率的结果为 1 的情况是因为两个分词结果均相同，且符合正确分词结果。这里我没有测试更多的句子，因为字典树中词语暂时无法构成有歧义的句子。结合 PPT 上给出的示例和 PKU 给出的光明日报测试集，我给出如上所示句子与测试结果。

五、 心得体会：

本次实验耗时颇多，实验主体基于实验二的成果，进一步测试最大匹配算法与最短路径算法，进一步巩固了理论课上所学的内容。在我看来本次实验有难度，在实现过程中难点和亮点在于创建分词区间。引入分词区间，在创建最短路径时用于搜索最远分词，在比较正确率计算重合度时可以直接比较区间长度，其难点在于如何利用 python 创建双元素数组，以及如何取其中某一维的元素进行比较。在实验中我学习了很多，对 python 的掌握也愈发熟练，希望在以后的实验中继续锻炼自己的能力。