**自然语言处理 第二次作业**

*黄昊 20204205*

**源代码：**

def getDic(datas):      *# 直接从已有的文件读取词典*

    s = set()

    for data in datas:

        word = data.split("\n")

        s.add(word[0])

    return s

def spawnDic(src, tar): *# 生成词典中间文件*

    dic = []

    with open(src, 'r', encoding = "utf-8") as f:

        datas = f.readlines()

        for i in datas:

            lst = i.split(":")

            dic.append(lst[0])

    with open(tar, 'w', encoding="utf-8") as f:

        for word in dic:

            f.writelines(word+"\n")

def wordDic(src, tar):  *# 返回集合对象，作为词典*

    with open(tar, "r", encoding="utf-8") as f:

        datas = f.readlines()

        if len(datas) != 0:

            return getDic(datas)

        else:

            f.close()

    spawnDic(src, tar)

    with open(tar, "r", encoding="utf-8") as f:

        datas = f.readlines()

        getDic(datas)

def forwardDivision(s: str, dic: set, maxSize: int = 10):      *# 使用前向算法进行分词*

    l = 0

    r = min(len(s), l + maxSize)   *# 左右指针l, r*

    ans = []

    while l < len(s) and r > 0:

        sub\_str = s[l: r]

        if l == r - 1:  *# 说明是单个字了，可以切为单字*

            ans.append(sub\_str)

            l = r

            r = min(len(s), l + maxSize)

        elif sub\_str in dic:      *# 匹配成功*

            ans.append(sub\_str)

            l = r

            r = min(len(s), l + maxSize)

        else:                   *# 未匹配成功*

            r -= 1              *# 缩小边界*

    return ans

def backwardDivision(s: str, dic: set, maxSize: int = 10):

    l = 0

    r = min(len(s), l + maxSize)

    s = s[::-1]                 *# 将字符串反转，后向匹配便可变换为前向匹配*

    ans = []

    while l < len(s) and r > 0:

        sub\_str = s[l: r][::-1] *# 子串再倒转一下，便可以在词表中查询*

        if l == r - 1:  *# 说明是单个字了，可以切为单字*

            ans = [sub\_str] + ans

            l = r

            r = min(len(s), l + maxSize)

        elif sub\_str in dic:      *# 匹配成功*

            ans = [sub\_str] + ans

            l = r

            r = min(len(s), l + maxSize)

        else:                   *# 未匹配成功*

            r -= 1              *# 缩小边界*

    return ans

def bidirectionalDivision(s: str, dic: set, maxSize: int = 10):     *# 双向匹配算法用于解决交叉歧义*

    ans\_forward = forwardDivision(s, dic, maxSize)

    ans\_backward = backwardDivision(s, dic, maxSize)

    if ans\_forward == ans\_backward:     *# 前向匹配算法和后向匹配算法一样*

        return ans\_backward

    elif len(ans\_backward) == len(ans\_forward):     *# 分词数量相同，返回单字较少的*

        singleWordCnt = lambda x: len(list(filter(lambda m: m == 1, [len(i) for i in x])))      *# 用于统计列表中单字个数的匿名函数*

        forward\_cnt = singleWordCnt(ans\_forward)

        backward\_cnt = singleWordCnt(ans\_backward)

        ans = ans\_forward if forward\_cnt < backward\_cnt else ans\_backward

        return ans

    elif len(ans\_backward) < len(ans\_forward):  *# 返回分词数量少的那个*

        return ans\_backward

    else:

        return ans\_forward

def readTestCase(src):      *# 读取测试用例*

    with open(src, "r", encoding = "utf-8") as f:

        datas = f.readlines()

        testcase = [data.split("\n")[0] for data in datas]

    return testcase

def solve(i, testcase, dic, maxSize = 10):      *# 进行切分词的操作*

    ans\_forward = forwardDivision(testcase, dic, maxSize)

    ans\_backward = backwardDivision(testcase, dic, maxSize)

    ans\_bidirect = bidirectioonalDivision(testcase, dic, maxSize)

    print("【测试用例 {:2d}】".format(i))

    print("\t正向最大匹配：","/".join(ans\_forward))

    print("\t逆向最大匹配：","/".join(ans\_backward))

    print("\t双向最大匹配：","/".join(ans\_bidirect))

dic = wordDic("large\_pinyin.txt", "dict.txt")

testcases = readTestCase("testcase.txt")

for i, testcase in enumerate(testcases):

    solve(i + 1, testcase, dic)

**测试结果：**

