|  |  |
| --- | --- |
| **自然语言处理及应用**  **实验报告** | |
|  | |
|  |  |
| **名称** | BMM&FMM |
| **姓名** | 卢佳源 |
| **班级** | 人工智能91 |
| **学号** | 2191121196 |
| Email | bjlujiayuan@126.com |
| **日期** | 2021-10-12 |

# 实验目的

完成FMM和BMM算法的中文自动分词，可以使用简单的词表。

# 实验环境

OS：Linux操作系统的ubuntu20.04版本

IDE：VScode,anaconda3(‘base’:conda)

Terminal：gnome-terminal -t $TITLE -x

Language：python(3.8.8 64-bit)

PATH： /home/lujiayuan/python/BMM/BMM.py&30wChinsesSeqDic\_clean.txt

# 实验方法

1. 实验思路：
2. 建立词典：

在网上找到一个30万中文分词词典的txt文件（30wChinsesSeqDic\_clean.txt），调用IsDict()函数读入词典。

1. BMM（逆向最大匹配分词算法）：

BMM（）函数：先找到词典中词语的最大长度（maxlength），然后将句子长度用senL表示，并将最大分词长度（maxwordlength）定义为剩余句子长度和最大词典词语长度的最小值。从句子的最后一个词开始，先截取maxwordlength长度的小句子，然后检查这个小句子是否在词典中，若存在在词典中，则这个小句子就是一个分词，并将句子长度减去这个小句子长度来更新senL这个变量；若不存在于词典中，则将这个小句子减去第一个字（正向的第一个字，即倒数第maxwordlength个字），然后继续在词典中查找是否有这个更新后的小句子，重复做这个步骤，直到在词典中可以找到这个小句子。之后将原来的输入整体句子去掉这个小句子，然后按照上述方法继续在词典中查找，直到整个句子查找完。最后将结果列表反向输出，即为BMM分词结果。

1. FMM（正向最大匹配分词算法）：

处理方法与BMM类似，只是从句子的正向去划分maxwordlength的小句子，然后在字典中查找是否存在这个小句子，若存在，则进入下一次划分循环；若不存在，则将小句子的最后一个字去掉，继续在词典中查找这个小句子，重复如此，直到遍历完整个句子，并将结果列表正向输出，即为FMM分词结构。

1. 实验代码：

from logging import \_srcfile

import os

import datetime

dictpath='30wChinsesSeqDic\_clean.txt'

def IsDict():

wordic=[]

datainfo = open(dictpath,'r',encoding='utf-8',errors='ignore').readlines()

for line in datainfo:

wordic.append(line.split(" ")[0].strip())

return wordic

def BMM(sentence,wordic):

result=[]

maxlength=max(len(word)for word in wordic)

sentence=sentence.strip()

senL=len(sentence)

while senL>0:

maxwordlength=min(maxlength,senL)

subsentence=sentence[-maxwordlength:]

while maxwordlength>0:

if subsentence in wordic:

result.append(subsentence)

break

elif maxwordlength==1:

result.append(subsentence)

break

else:

maxwordlength=maxwordlength-1

subsentence=subsentence[-maxwordlength:]

sentence=sentence[0:-maxwordlength]

senL=senL-maxwordlength

result.reverse()

words="/".join(result)

return words

def FMM(sentence,wordic):

result=[]

maxlength=max(len(word)for word in wordic)

sentence=sentence.strip()

senL=len(sentence)

i=0

while senL>0:

maxwordlength=min(maxlength,senL)

subsentence=sentence[0:maxwordlength]

while maxwordlength>0:

if subsentence in wordic:

result.append(subsentence)

break

elif maxwordlength==1:

result.append(subsentence)

break

else:

maxwordlength=maxwordlength-1

subsentence=subsentence[0:len(subsentence)-1]

sentence=sentence[len(subsentence):]

senL=len(sentence)

words="/".join(result)

return words

def main():

# maxwordlength=5

sentence="滤波要分通道滤波，这里由于博客问题，显示灰色（好像因为今天清明节）（逝者已矣 生者如斯）"

wordic=IsDict()

print('BMM: ')

result=BMM(sentence,wordic)

print(result)

print('\n FMM: ')

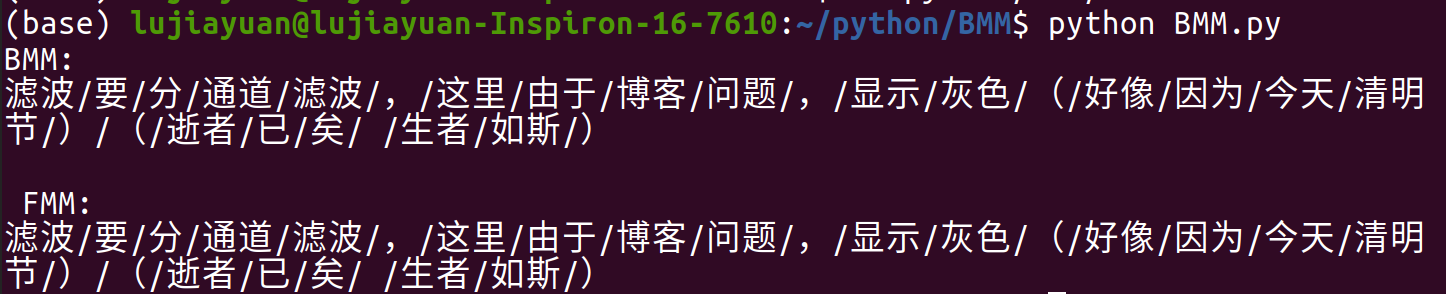
result=FMM(sentence,wordic)

print(result)

if \_\_name\_\_=='\_\_main\_\_':

main()

1. **实验结果**
2. 实验结果截图：



1. 实验结果分析：

第一行为BMM分词结果，第二行为FMM分词结果。

1. **遇到问题及解决思路**

问题：这次实验的主要问题在于句子的划分边界，以及变量要在循环中不断更新，否则输出结果出错。

解决思路：不断把每一步结果输出，看问题出在哪一步，然后debug。

问题：分词词典的建立，我之前找到清华的一个分词词典，但是由于这个词典太过于详细，导致分词结果都是每一个单字。

解决思路：找一个比较小的粗略的分词词典，或者根据自己的输入句子自建词典（把我们认为的分词结果当作词典），即可得到较好的分词结果。