中文分词实验报告

人工智能学院 周韧哲 181220076

目录树

```
HW1
├── cws_dataset
│   ├── res
│   │   └── 181220076.txt    #分词结果
│   └── *.txt   #字典相关文件,train.txt,dev.txt,test.txt等等应该放在这里
├── FMM.py    #Forward Maximum Match Inplement
├── IMM.py    #Inverse Maximum Match Inplement
└── main.py   #main loop
```

运行

本项目运行于manjaro系统，python版本为python3.8。命令行可选参数有：

```
optional arguments:
  -h, --help            show this help message and exit
  --nthreads NTHREADS   num of threads
  --max_len MAX_LEN     max match lenth when cutting words
  --alg {FMM,IMM,BMM}   algorithms
  --input               input mode for debug
```

nthreads, max_len, input, alg分别代表进程数、字典匹配的最大长度、是否接受终端输入、使用的算法类型，在运行前请确保train.txt, dev.txt, test.txt在`HW1/cws_dataset/`目录下。比如：

使用双向最大匹配算法对test.txt进行分词

```
python main.py --alg BMM --nthreads 4 --max_len 12
```

结果文件为`HW1/cws_dataset/res/181220076.txt`。最优分词结果使用算法BMM，max_len为12，在默认情况下，输入`python main.py`即可复现Leaderboard最优结果。

接受控制台输入进行debug

```
python main.py --alg BMM --input
```

输入一句话会返回其分词结果，输入`quit()`退出。

具体实现

FMM

正向最大匹配在`FMM.py/class FMM`中，其核心思想是：从文本开头处向结尾处推进，用一个长度从max_len开始不断减小的窗口切开文本，直到子文本出现在字典中。python实现的算法框架如下：

```
INPUT: text, max_len
beginp = 0, res = []
while True:
    if beginp >= len(text):
        break
    for i in range(max_len,0,-1):
        if text[beginp:beginp+i] in word_dict or i==1:
            res.append(text[beginp:beginp+i])
            beginp += i
            break
return res
```

IMM

逆向最大匹配在`IMM.py/class IMM`中，其核心思想是：从文本结尾处向开头处推进，用一个长度从max_len开始不断减小的窗口切开文本，直到子文本出现在字典中。其python实现的算法框架如下：

```
INPUT: text, max_len
endp = len(text), res = []
while True:
    if endp <= 0:
        break
    for i in range(max_len,0,-1):
        if text[endp-i:endp] in word_dict or i==1:
            res.append(text[endp-i:endp])
            endp -= i
            break
return res
```

BMM

双向最大匹配的实现在`main.py`中，其核心思想是：分别用FMM和IMM分词，如果结果相同，则选择任意一个作为分词结果；否则，根据某个确定性策略来选择结果。在这里我使用的策略是：哪个结果的单字符少，就选择哪个分词结果。

在`main.py`的`main_loop()`函数中，使用变量alg来抽象出使用的算法，同时使用python的multiprocessing模块进行了分词的多进程实现，提高了速度。

Tricks

为了提高F1 score，我使用了如下等等的tricks：

- 在最初时，为了提升切分效果，我从QQ群中诸多额外数据集中选取了十几个数据集，并从网上下载了清华的新闻词库选取了部分作为`THUCNEWS.txt`加入到词典中。
- 文本中常常含有数字、英文字母等，而这些不可能全部在字典中出现，所以我先使用正则表达式识别出这些特殊字符，使得能够成功匹配数字、网址、邮箱、任意长度英文单词、时间（形如23:00）等。当进行字典匹配时，如果遇到了已经被正则匹配出来了的字符串，就跳过，进行下一段文本的切分。
- 我发现训练集中形如“231万”、”2008年“、”34.4%”等都是连在一起的，所以我增加了一个规则：如果当前字符串为数字，且下一个字符出现在“%时年月日亿万后千”等等字符中，则划分到一起。
- 为了解决测试集中出现的人名无法识别的问题，我从百家姓中筛选出一些常用姓，在文本切分完成后，找出其中的姓，然后判断姓的下一个切分字符串是否为2个字符，或者下一个切分字符串为1个字符且下下一个切分字符串也为1个字符，则将其拼接在一起，认为是一个人名，这样虽然会误杀一部分不是人名的词，但满足这样条件的词中不是人名的情况比较少，所以造成的一些误差是可以接受的。
- 在同学提示下发现训练集中某些词的切分方法与现有词典不一样，比如”这个“在训练集中是切分开来了的，但我的词典里有”这个“这个词，而类似的词语有挺多的，这就会造成了分词效果的不理想。因此我找出来了这些词，将其称之为magic word，在切分文本时出现这种词时手动分开。

实验总结

这一次的中文分词实验在算法实现上是很简单的，但是要得到一个比较高分的分词结果比较难，在规则分词模式下，需要加入更多的词库，也需要通过观察训练集和测试集，来总结出一些小trick，而这些trick往往能够大幅提升分词效果。