NLP HW1 — 基于规则的分词

丁豪 南京大学 人工智能学院
181220010@smail.nju.edu.cn

一、实验描述

本次实验为中文分词任务，基本思想是基于规则的分词，与第二节课上所讲的匹配、消歧算法相对应。实验给出train.txt dev.txt两个已经进行好切分的语料库作为训练数据，然后根据训练得到的结果对test.txt进行分词，使用同样的空格分割方式重新生成文件并提交。实验评价指标为f1，且限定只能使用规则分词算法。评分标准为，达到SOTA指标（89%）可以获得实验满分，前10名附加30%分数但需要进行汇报。

额外：经过与助教在群内沟通，允许使用自己动手实现的非规则算法（如统计学），以及使用第三方额外语料库，不过需要在使用的同时发在群中与同学共享以保证公平。

二、实现过程

1.总体思路

我们在课堂上了解到中文是一种分析型语言，即词形变化较少且没有表示词的语法功能的附加成分，因而弄清楚词的边界以及他们之间的关系，需要借助词序和虚词来加以判断。本次分词提供了较大的训练集，为构建分词词典提供了便利，此外根据对测试集、训练集中存在特殊情况的观察，我们又可以总结出一系列用于归类处理特定情境的分词规则。以上两者将是我们此次分词的主要依据。

在算法方面，我的选择是实现“规则切分+双向最大匹配取词+重切分”。但在消歧规则部分使用一个带有启发式的"评价函数"，此函数综合考量分词粒度以及语料库的统计学数据，给出不同切分方案的综合打分，最终选择正向最大匹配与逆向最大匹配中得分较高的一个作为最终匹配结果。在根据规则或匹配获得词组之后，将其的切分形式（e.g. 帝王→帝 王）在原始训练集（train.txt dev.txt）中重新搜索，如果切分形式出现次数大于非切分形式，则重新切分此词语为两个单字，以此消除额外语料库导致的不匹配问题。

2.代码实现

2.1 文件结构介绍

`main.py`：主文件，进行完整读取、分词、生成操作。
`data_process.py`：对语料库进行整合，用于将本次实验中收集和使用的语料库进行预处理以及生成最终用于学习的完整训练集（其中部分功能在使用后已经删除）。
`jieba_baseline.py`：使用结巴直接对语料库进行分词，用于对比测试。
`diff_test.py`：用于对两个文件进行差异性检查，可以对比不同版本算法得出结果的差别。

2.2 主算法（main.py）

`init_files`, `update_dict`, `generate_dict`,`generate_pure_seg`,`generate_pure_con`：读取文件和构建语料库字典。每当出现新词则加入字典，如果出现旧词则相应键的值加1，以此达到统计语料库完整词频的目的。

`match_rule`：使用正则表达式进行规则匹配。规则存储在一个全局列表`patterns`当中，其中每一条规则的第一项为对应的正则表达式（省略了开头的^与结尾的$，在下面使用的时候统一添加），第二项为对应的分割方案（如果为None则全字段为一个词组，如果是一个列表形式，则列表中每一个整数对应了此串对应下表的一个切分点）。这一部分的规则来自于对训练集中已经划分的词组模式的观察归纳，

具有较强的针对性和准确度，因此在决定划分时优先使用规则，当规则不符合时再进行词库匹配。这样使用正则表达式和切分点列表的方式，大大增强了规则的可扩展性，如果需要添加新的规则，只需要总结出此规则对应的正则模式（可以超出待匹配字段本身），然后再指定好所需的切分方案即可。

```
# 规则
patterns = [
    ["[0-9]+(\.[0-9]+)?%?",None],                          # 纯数字、百分数情况
    ["[0-9][:：][0-9]",None],                               # 比分
    ["[a-zA-Z0-9@_\.]+\.com(\.cn)?",None],                 # 网址
    ["[a-zA-Z&]+",None],                                   # 英文单词
    ["[0-9]+(\.[0-9]+)?余万",None],                         # 金额
    ["[0-9]+(\.[0-9]+)?万亿",None],                         # 另一种金额
    ["[0-9]+(\.[0-9]+)?[万亿余]",None],                     # 还有一种金额
    ["[0-9一二三四五六七八九十百两]+年[前后间内]{1}",[-2,-1]], # 年的一些搭配
    ["20[0-9]{2}-20[0-9]{2}年",None],                       # 年的另一种情况
    ["20[0-9]{2}-20[0-9]{2}年度",[-2]],                     # 年度区间
    ["20[0-9]{2}年度",[-2]],                                # 年度
    ["[(20)(19)]{1}[0-9]{2}年底",None],                     # 年底
    ["[0-9]{1,2}：[0-9]{2}-[0-9]{1,2}：[0-9]{2}",None],     # 中文时间区间
    ["[0-9]{1,2}：[0-9]{1,2}",None],                        # 中文时间
    ...
]
```

`pure_append`：针对获取到的词组，进行重切分检查，并根据bool forward的值来决定如何插入列表（forward则用append，backward则用insert(0)）。

`max_forward`，`max_backward`：实现了极大正向、逆向匹配，且优先使用上述规则匹配。

`evaluate`：这里使用的评价指标为 **语料库中包含词的总词频 / 整句总词数** 。最大化这个评价指标可以达到两个目的，一个是使得划分后的词平均词频更大（统计学方法），另一个是使得整句话的总划分词数量更少（传统双向最大匹配的评价指标）。

`cut`：主函数，综合调用上述`match_pattern`，`max_forward`，`max_backward`，`evaluate`函数，完成整个分词过程。

2.3 数据来源

本次实验使用的数据初初始的train.txt与dev.txt之外，全部来自NLP课程群中同学的分享，此外还有本人分享的“全世界主要国家和城市数据表.txt”。

三、结果

此次分词作业最终F1：0.9302176435263733。超过了SOTA的分词水平，截止到报告完成时为Leaderboard第一名。

四、总结

本次作业是基于规则的分词任务，在编程实现的过程中，我深入体会了双向最大匹配算法，以及基于规则的匹配、消歧方法。通过亲自动手收集规则以及寻找扩充词库，我对规则分词的工作之复杂性和困难性有了全新的认识，也意识到完全依靠人力硬编码的规则其能力的有限。在未来学习掌握了更加精妙的NLP方法之后，也许我们会对分词任务有更好的掌握。