北京航空航天大學

BEIHANG UNIVERSITY

自然语言处理实验一

基于 HMM 的命名实体识别

院 系 名 称	人工智能研究院
任 课 老 师	丁嵘
学 生 姓 名	赵天一
学 生 学 号	19373006

2021 年 11 月 8 日

一. 任务分析及数据准备

1.中文分词

词是最小的能够独立活动的有意义的语言成分，英文中每个句子都将词用空格或标点符号分隔开来，但是在**中文中很难对词的边界进行界定**，难以将词划分出来。

在汉语中，虽然是以字为最小单位，但是一篇文章的语义表达却仍然是以词来划分的。因此在分析中文文本时，需要进行分词处理，将句子转为词的表示，这就是中文分词。

2. 数据集

- **数据来源**：人民日报语料库
  - **词典**：先验知识，已知为词的汉字组合
  - **待分词文件**：未分词的句子
  - **分词对比文件**：groundtruth，结果分析时需要用到
  - **停用词**：指自身无实际意义的词，语气助词、副词、介词、连接词等，
- **数据读入**
  - 代码

```
# 数据集读取
f_train=open("./词典/pku_training_words.utf8","r",encoding='utf-8')
dic_words=f_train.read().splitlines()
dic_words=frozenset(dic_words)
#print(dic_words[0])
f_texts = open("./待分词文件/corpus.txt", "r", encoding='utf-8')
texts = f_texts.read().splitlines()
#print(texts[0])
f_seg_groundtruths = open("./分词对比文件/gold.txt", "r", encoding='utf-8')
seg_groundtruths = f_seg_groundtruths.read().splitlines()
#print(seg_groundtruths[0])
f_stop_words = open("./停用词/stop_word.txt", "r", encoding='utf-8')
stop_words = f_stop_words.read().splitlines()
```

  - 读入后的数据结构：
    - 词典：**dic_words**: ['词1'，'词2' , ...]
    - 待处理文字：**texts**: ['句子1','句子2'，.....]
    - 分词对比文件：seg_groundtruths: ['真实结果1词1 真实结果1词2 .....'，'真实结果2词1 真实结果2词2' .....] **注：以空格分隔**
    - 停用词：stop_words: ['停用词1'，'停用词2'，.....]

二.最大匹配算法

1. 正向最大匹配：

- 对输入的句子从左至右，以贪心的方式切分出当前位置上**长度最大的词**，组不了词的字单独划开。

基本思想：

1. 设自动分词词典中最长词条所含汉字个数为I
2. 取被处理材料当前字符串序数中的 I 个字作为匹配字段，查找分词词典。若词典中有这样的一个 I 字词，则匹配成功，匹配字段作为一个词被切分出来，转 6
3. 如果词典中找不到这样的一个 I 字词，则匹配失败
4. 匹配字段去掉最后一个汉字，$I--$
5. 重复 $2-4$，直至切分成功为止
6. I重新赋初值，转2

2. 逆向最大匹配：

- **原理与正向最大匹配相同**，但顺序不是从首字开始，而是**从末字开始**，使用的分词词典是**逆序词典**，每个词条都按逆序方式存放。**在实际处理时，我没有使用逆序词典，而是按照正向最大匹配的反逻辑，又写了一遍。**

3. 双向最大匹配& 分词歧义处理

- **算法思想**：将正向最大匹配与逆向最大匹配组合起来，对句子使用这两种方式进行扫描切分，提升了识别率，增强了鲁棒性。
  - 如果正反向分词结果词数不同：则取分词数量较少的那个
  - 如果分词结果词数相同：
    - 分词结果相同，说明没有歧义，可返回任意一个
    - 分词结果不同，返回其中单字较少的那个

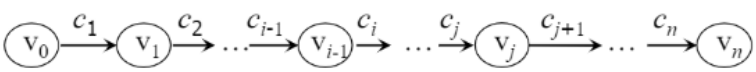
三.最少分词法(最短路径法)

1.数据结构

- 基于**图数据结构**，将中文句子建图，**每一条边代表一个汉字**，每一个节点的下标用于句子切片，具体建图算法如下：

基本思想

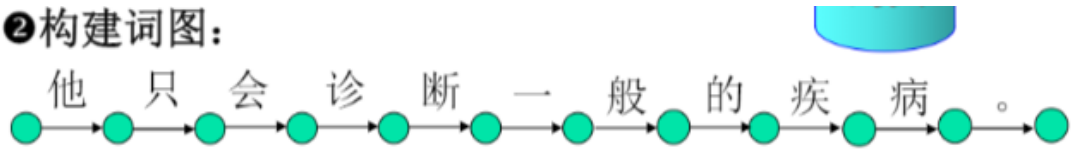
设待切分字串$S = c_1c_2...c_n$，其中$c_i(i = 1,2,...,n)$为单个的字，n为串的长度，$n \geq 1$。建立一个节点数为$n+1$的切分有向无环图G，各节点编号依次为V_0，V_1，V_2，...，V_n。



求最短路径：贪心法或简单扩展法。

- **结果可视化**：

❷构建词图：

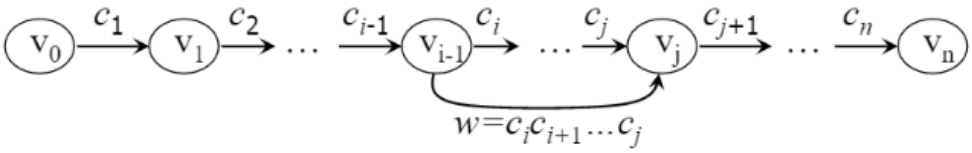


2.基于词典构建有向无环图

- 将句子中**在词典中的词之间生成一条边**连接起来,具体算法如下：

(1) 相邻节点 v_{k-1}, v_k 之间建立有向边 $<v_{k-1}, v_k>$，边对应的词默认为 c_k (k =1, 2, …, n)。

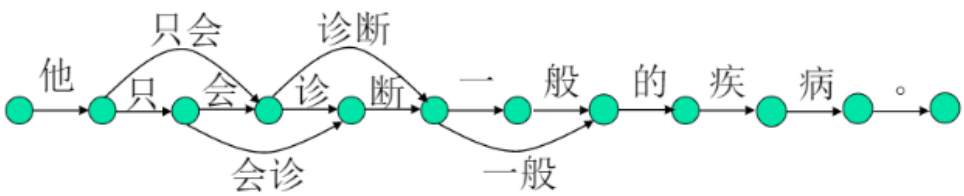
(2) 如果 $w= c_i c_{i+1} \ldots c_j$ ($0<i<j\leq n$) 是一个词，则节点 v_{i-1}, v_j 之间建立有向边 $<v_{i-1}, v_j>$，边对应的词为 w。



(3) 重复步骤(2)，直到没有新路径(词序列)产生。

(4) 从产生的所有路径中，选择路径最短的(词数最少的)作为最终分词结果。

- **结果可视化**：



3.寻找最短路径

- 寻找最短路径的算法有很多，但是对于此问题来说，因为图中所有边的权重都为1，且对于此序列问题，只会单向向后寻找，直接使用**广度优先遍历寻找最短路径**，理论上时间复杂度最高为O(n)，所以我使用了广度优先遍历。

四.改进与创新

1.日期和时间的识别

- 想法来源：助教**在README中提到了**，增加日期的识别，要增加一个基于模板匹配的算法，来增加分词的正确率，以及算法泛化性。通过观察groundtruth发现，除了日期之外，时分秒（秒除外）的时间也有同样的性质例如：‘3分’，‘零时’，等。所以我做了日期和时间的模板匹配函数，来提高模型的泛化性，和正确率。
- **构建数字词典**：num_dic

```
num_dic=['0','1','2','3','4','5','6','7','8','9','零','一','二','三','四','五','六','七','八','九','十','o','.']
num_dic=set(num_dic)
```

- **函数思想**：将切片的piece作为函数输入，若piece[-1]是关键词（年，月，日，时，分），则遍历piece[0:len(text)-1],若其中有一个不在数字词典中,则不符合规则返回False,若都符合规则，则piece就是所要找的（年月日时分）。

- **改进前**

```
D:\anaconda\envs\swin\python.exe E:/univeisity/大三上/NLP/课件/实践二：中文分词/NLP_segmentation.py
100%|██████████| 1944/1944 [00:20<00:00, 97.02it/s]
[['共同', '创造', '美好', '的', '新世纪', '—', '二', 'o', 'o', '一', '年', '新年', '贺词'], [' (', '二ooo年', '十二月', '三十一日', ') ', ' (', '附', '图片', '1'
最少分词法：Precision: 85.367114% Recall ratio: 91.813896% F1-Measure 88.473221%
双向匹配法：Precision: 85.368265% Recall ratio: 91.816771% F1-Measure 88.475174%
```

- **改进后**

```
D:\anaconda\envs\swin\python.exe E:/univeisity/大三上/NLP/课件/实践二：中文分词/NLP_segmentation.py
100%|██████████| 1944/1944 [00:29<00:00, 66.26it/s]
[['共同', '创造', '美好', '的', '新世纪', '—', '二o一年', '新年', '贺词'], [' (', '二ooo年', '十二月', '三十一日', ') ', ' (', '附', '图片', '1', '张', ') '], ['女士', '们', ',
最少分词法：Precision: 88.067523% Recall ratio: 92.422297% F1-Measure 90.192375%
双向匹配法：Precision: 88.077783% Recall ratio: 92.434753% F1-Measure 90.203687%

Process finished with exit code 0
```

- **结果队比分析**

		P	R	F1_Measure
最小分词法	**改进前**	**85.367%**	**91.813%**	**88.473%**
	改进后	**88.068%**	**92.422%**	**90.192%**
双向匹配法	**改进前**	**85.368%**	**91.817%**	**88.475%**
	改进后	**88.078%**	**92.435%**	**90.204%**

- **准确率上升了2.7%左右，召回率上升了0.6%左右，F1_Measure上升了1.8%左右**

2.将词典存为frozenset()

- 可以极大的提高运行速度，减少完成作业时的高额时间代价
- 原因：
  - **set**： set对象是由hashable 对象所组成的无序集合，set对象的每一个元素要求可进行哈希运算，set 会对内部元素进行去重，每个元素在同一个set 中只会出现一次，但是由于set对象可变性，所以set对象自身不可哈希。
  - **frozenset**： frozenset 对象可以看成一个不可变set对象，是一个可哈希对象，所以在查找词典时采用哈希查找，时间复杂度O(n),大大降低了时间复杂度 。
- 特此感谢**(吴凌轩，王柏森)**的小tricks。

六. 问题

1. 当前存在的问题和困惑

1. 未登录词无法正确分词
2. 在观察分词结束后，观察结果文件发现，人名的分词是一个很大的问题，这个问题也可以归结于未登录词中。
3. groundtruth中例如：“6300万” 这种数字+单位的词被分为了两个词（不理解）
4. 关于时分秒的分词问题,例如:’3秒’ 被分为了两个词,但是‘3分’又被分为了一个词，（很奇怪）

2.未登录词的分词问题解决办法__HMM模型

- 结巴分词原理
  - 原理：基于**前缀词典**实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的**有向无环图** (Directed Acyclic Graph,DAG)；采用了动态规划**查找最大概率路径**, 找出基于词频的最大切分组合；
  - **对于未登录词，采用了基于汉字成词能力的 HMM 模型。**

五.词云制作

1.数据来源

- 《中华人民共和国国民经济和社会发展第十四个五年规划和2035年远景目标纲要》 ——第五章

2.词云展示

3.代码修改

```
with open('./中华人民共和国国民经济和社会发展第十四个五年规划和2035年远景目标纲要节选.txt', 'r', encoding='utf-8') as f:
    txt = f.readlines()
    #print(txt)
    # 将jieba分词换成你所实现的分词算法
    #words = jieba.cut(txt)
    f_train = open("./词典/pku_training_words.utf8", "r", encoding='utf-8')
    dic_words = f_train.read().splitlines()
    dic_words = frozenset(dic_words)

    BIM = seg.BiDirectctionMatchingMethod(txt, dic_words, None)
    FMM_result_all = BIM.FMM_cut()
    RMM_result_all = BIM.RMM_cut()
    seg_result = BIM.get_best_matching_result(FMM_result_all, RMM_result_all)
```

- 将jieba.cut()这个库函数替换成我自己实现的的**双向最大匹配算法**，并对数据结构进行简单修改。
- 注：再去除停用词时若不注意数据结构的话，很可能出现问题。