**Chinese word segmentation 设计文档**

1. **任务定义**

对一个给定的已分词的中文文本进行分词模型的构建。本实验中，给定输入为已做分词的带标注的中文文本（人民日报的标注语料），通过数据清洗和前向最大匹配算法输出为一个分词模型以及根据此模型输出的分词结果，并对这个模型进行f-measure的评价。

1. **源码运行环境**

采用python2.7作为编程语言，编程环境为win10操作系统下的wing IDE 5.1。脚本为test\_seg.py，输入文件为renminribao.txt，输出为seg.txt，均采用gbk编码。

1. **输入输出**

**程序输入：**

给定输入为作业包中附带的人民日报语料，其语料特点为：

1. 已分词，且所有的符号（标点、限界符等）均算作一个词。
2. 已标注，标注格式均为“/\w+”（利用python的正则表达式来表示）。
3. 有空行，在windows下，空行被表示为\r\n。
4. 有字音标注，如多音字结{jie1}。
5. 有分界符，以标注专有名词，如[亚太经合/j 组织/n]nt。
6. 每一行都以日期开始。

将数据清洗过的前1000行作为测试集，1001-20000行作为训练集。（请注意，这里的行指的是python中的迭代器的行，而非txt文件中的行，也就是说每一行可能包含多个句子。）

此时我们得到了三个输入集合：

1. 文本的前1000行，经过数据清洗，且有待分词，为一串字符。
2. 文本的前1000行，经过数据清洗，保留原来的分词，作为分词评价的对照。
3. 文本的1001-20000行，经过数据清洗，保留原来的分词，作为最大前向匹配的训练集。

**程序输出：**

输出为被模型分词后的集合1（以seg.txt文件的形式存储结果）以及分词结果的一些评价指标（precise，recall和f-measure）。

1. **方法描述**
2. **实验框架**

我们以框图的形式给出本次实验的基本架构。

给定数据输入集后，统一采用python正则表达式模块进行数据清洗，随后划分出三个集合，分别是待分词串，对比词串和训练词串，其中后两者都是保留了原来的分词，而待分词串只是一串词序列，并未进行分词。

1. **数据清洗**

采用python的正则表达式模块re来完成数据清洗。

首先逐行读入数据，观察模式后，将每一行的数据引用如下的正则表达式进行数据清洗，仅得到分词之后的结果，过滤到多余的符号和数字。

tag = re.compile('/\w+') 过滤词性标记

data = re.compile('\d+-\d+-\d+-\d+\s+') 过滤前端日期

tone = re.compile('\{.\*?\}') 过滤多音字标记

left = re.compile('\[') 过滤专有名词左标记

right = re.compile('\]\w+') 过滤专有名词右标记

blank\_line = re.compile('\r\n') 过滤所有空行

根据输入的测试集、训练集分界行号，可以在数据清洗的过程中直接构造集合2和集合3。过程getLinesFromFile(filename,begin\_train,end\_train,end\_test)要求输入文件句柄，训练集开始行，训练集结束行和测试集结束行（为了不浪费数据，我们默认两个集合之间是衔接的）。此过程返回两个集合，分别是train\_list和test\_list。

1. **测试集的构造**

有test\_list是已经分词的结果，我们需要将test\_list还原回待分词状态，即生语料状态。过程getRaw(test\_list)实现这一功能，利用python重载的 += 运算符对字符串操作可以轻松完成，此时我们得到了集合1、集合2和集合3。

1. **前向最大匹配算法**

主要思路是，从左向右扫描待分词串，构造一个初始值为训练集中最大词长度的滑动窗口，如果滑动窗口内的词串在训练集中有匹配，那么我们将这个词划分出来，窗口向右平移，恢复窗口至初始值，窗口左界限为原窗口的右界限；如果没有匹配，那么我们将窗口的右界限-1，并重复上述试探过程。

用伪代码描述为：

*while True:*

*if 窗口左界限== 终结符:*

*break*

*for 窗口宽度从最大值递减至1*

*if窗口右界限超出待分词串范围:*

*将窗口右界限置为待分词串右边界*

*else:*

*更新窗口右边界*

*if窗口内的词在训练集中 或者 窗口宽度为1:*

*分词，窗口向右滑动*

*if本次分词分错了*

*记录这个分错结果以便统计*

*实现剩余待分词串和对比词串的同步以便下一次分词*

*break*

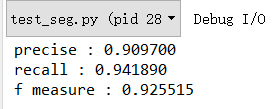
从上述伪代码可以看出，此过程需要注意的无非只有三点，其一：如何判断窗口内的词在训练集中？其二：窗口最大值如何确定？其三：怎样判断分错了？如何实现同步？

先解决前两个问题，首先利用nltk的freqdist方法统计测试集中所有词的词频构成字典，经过验证，对于一个20000行的文本，这个过程仅需要不到5s，比起每分一个词就要查找全文，这“查字典”的方式无非要快上很多。有了这个字典，仅需要判断待分的词是否在字典中即可。而这个字典中长度最大的“键”的宽度就是我们要找的窗口最大值（即出现在测试集中的可能的最长词）。

对于同步问题，我们不妨这样设计。由于每分出一个词，这个词只有可能“分错”或者“分对”，比较基准就是对比词串的某一个词。由于对比词串和待分词串的每一个字单位都一一对应，只不过一个已分，另一个待分，我们可以通过字同步的方式来判断分词的结果是否正确。设计一个词指针p，这个指针指向对比词串中的某一个词，我们称之为对比词。再设计一个前向字指针pf，这个指针与滑动窗口右边界同步，每当产生新的分词时，都需要重新更新pf的位置。p的位置需要更新，当且仅当pf从对比词串的某一个分词进入到了下一个分词。而获得“从一个词到了另一个词”的同步信息，我们可以设计在对比词串中添加同步符号‘|’，只要pf遇到了同步符号，pf再向前移一个字，此时f向前移动一个词即可。

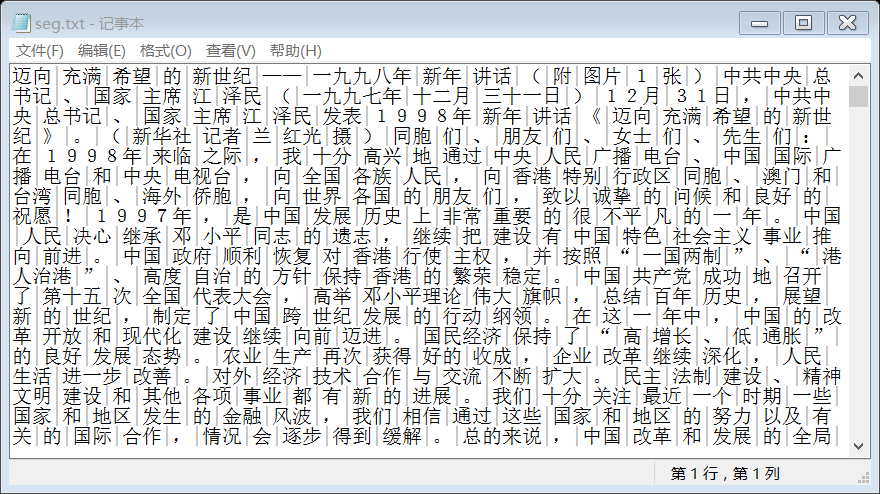
1. **结果分析及性能评价**

我们将输出文件保存至“seg.txt”中，将每一个词用分隔符‘|’分开以便于查看分词结果。precise、recall和f-measure直接打印即可。按照如上的输入输出，其结果为：



可以看出FMM算法应用到这个数据集上有较好的结果，f接近于1，且准确率也达到了90%。

截取seg.txt的部分内容可以看出，基本上达到了相应的分词效果。



但是我们不能因此而乐观，可以看到本次的训练集和测试集都来源于统一文件，这个文件均是来自人民日报的语料，其内容可以说具有高度的关联性。可以想象，如果训练集来自于本语料，而测试集来源于另一个风格截然不同的语料（如电影评论），FMM算法就不会尽如人意。这是由于FMM算法是基于实例的算法，分词仅仅是根据测试集中“是否存在”而决定是否分词，这相当于同意了一个假设“当且仅当这个词存在于训练集的时候才进行分词”，但根据课上讲的zipf法则，一个训练集永远无法涵盖一切语料。也许我们可以寻找另一种分词方法，这种方法不仅仅依赖于“是否存在”，而是“是否相关”。