

**课 程 实 验 报 告**

**课程名称： 自然语言处理实验**

**专业班级： 计算机科学与技术202207**

**学 号： U202215561**

**姓 名： 瞿明睿**

**指导教师： 魏巍**

**报告日期： 2025年 5 月 5 日**

**计算机科学与技术学院**

目 录

[1 中文分词实现 2](#_Toc197969463)

[1.1 实验简介 2](#_Toc197969464)

[1.2 基于字典的基础实现 2](#_Toc197969465)

[1.3 拓展实现 3](#_Toc197969466)

[1.4 实验小结 5](#_Toc197969467)

[参考文献 7](#_Toc197969468)

[附录A 中文分词实现的源程序 8](#_Toc197969469)

# 1 中文分词实现

## 1.1 实验简介

中文分词指将汉字序列切分成单个词语的过程。在以英文为代表的拉丁语系中，单词之间是以空格作为自然分界符的。而中文只有字、句和段能通过明显的分界符来划界，唯独词没有一个形式上的分界符，这给中文的处理带来了独有的困难，因此中文分词技术得到了广泛研究。

实验要求实现基于词典的分词方法，Bi-LSTM+CRF模型以及进行优化，同时最后对比不同算法分词效果和性能。

实验基于python3.7 pytorch1.7.1 transformers4.0.0 cuda11.0

## 1.2 字典实现

基于字典的分词器，我们根据实验任务书，完成补全几个基础算法，从整个算法上来看，实现的主要是一个Tokenizer类，包含三种匹配方法，如下

1.fmm\_split正向最大匹配

匹配逻辑：从文本的开头向后扫描。在当前位置时尝试匹配词典中存在的最长词语（不超过最大长度）。如果找不到匹配，则缩短尝试匹配的子串长度，直到找到匹配或子串长度为1（单字成词）。将匹配到的词语加入结果列表，并移动扫描的指针。

2.rmm\_split逆向最大匹配

匹配逻辑：在当前位置，尝试匹配词典中存在的最长词语。如果找不到匹配，则缩短尝试匹配的子串长度，直到找到匹配或子串长度为1。将匹配到的词语插入到结果列表的前端（保持正确的顺序），并移动扫描指针。

3.bimm\_split双向最大匹配

匹配逻辑：分别调用fmm+rmm得到两种分词结果。

同时算法中需要比较结果，如果两种方法分出的词数不同，选择词数较少的结果。如果词数相同，则两种结果中单字词的数量较少的结果。

通过输出分词结果提交后能得到这样的分词方法准确度是在78%-79%之间,具体准确度如下图

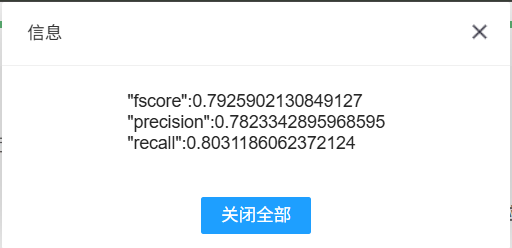


图1.2.1 fmm验证

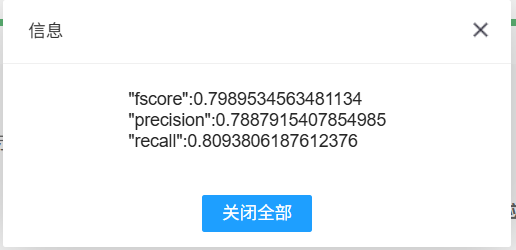


图1.2.2 rmm验证

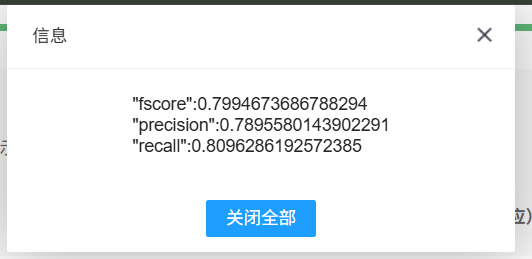


图1.2.3 bimm验证

## 1.3 深度学习实现

拓展实现基于实验书给出的几个优化方案如下：

1.修改网络结构，例如引入BERT等预训练语言模型；

2.与命名实体识别算法相互配合，减少对命名实体的错误分割；

3.构造合适的词典集（可扩充+人工整理）；

4.实现新词发现（登录）功能，识别测试集中的新词（未登录词）；

5.调整、优化模型训练过程中的超参数。

### 1.3.1基于BiLSTM+CRF

模型结构分为Embedding层+BiLSTM层+CRF层，大概的功能如下

Embedding层将输入的字符映射为稠密向量。

BiLSTM层构建双向LSTM。从两个方向处理输入序列，从而更好地捕捉上下文信息。LSTM的输出会经过一个线性层映射到标签空间的大小。

CRF层接收BiLSTM的输出并计算给定标签序列的条件概率。在训练时计算损失函数；在推断时使用维特比算法解码出最优的标签序列。

同时在分词任务中使用BMES标签

B: Begin (开始)

M: Middle (中间)

E: End (结束)

S: Single (成词)

CRF层中的num\_tags参数4对应这个标签体系的大小。

数据加载与预处理中，将一个批次的数据进行填充以使序列长度一致，并创建掩码在CRF计算中忽略填充部分。数据预先处理成字符ID序列和对应的标签ID序列，并使用pickle保存。

基础的网络实现是很标准的BiLSTM+CRF序列标注任务的标准流程：

字符嵌入 => BiLSTM提取特征 => 线性层映射到标签空间 => CRF解码得到最优标签序列。

通过执行默认参数的训练，我们可以得到训练出来的结果大概是84%左右的准确率可以参看下图：

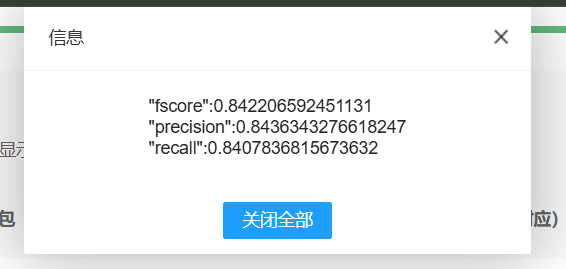


图1.1 原始网络性能

### 1.3.2基于网络做的基础优化

通过分析需要分词的任务文本，我们可以发现其中包含中文，英文和数字。但是其中数字和英文的部分很少，我们可以针对这一点对模型做一些小的修改

在数据处理中引入字符类型特征，定义了字符类型映射char\_type\_map。在处理数据时为每个字符确定其类型，并将这些类型信息与字、标签一同保存。在CWS模型的初始化函数增加对应需要的参数。同时添加一个新的嵌入层用于学习字符类型的表示。

此外另外添加参数，BiLSTM层设置为num\_layers=2，并加上dropout=0.2，在BiLSTM层与CRF层之间添加一个额外的线性层，ReLU和Dropout层，进一步提取和整合特征。

通过以上的小调整我们可以获得一些些的性能提升如下图

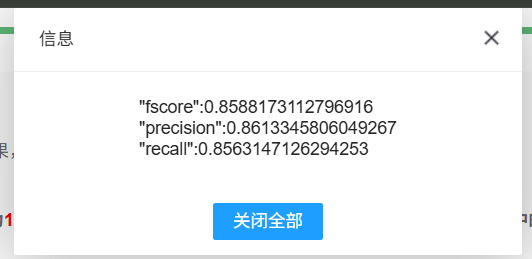


图1.2 性能提升

我们可以看到大概是有2%左右的提升但是不多，因为并没有做更多的调整。

### 1.3.3 网络参数的调整

我们可以将原先的run.py的

    parser.add\_argument('--embedding\_dim', type=int, default=200)

    parser.add\_argument('--lr', type=float, default=0.005)

    parser.add\_argument('--hidden\_dim', type=int, default=300)

三行进行调整，此时也有效果如下图：

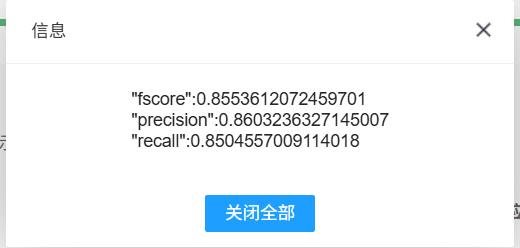


图1.

那么此时我更加激进的给出参数

### 1.3.4 bert预训练引入（实现的效果不好）

### 1.3.5 NER+BERT最终优化实现

## 1.4 实验结果

××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××,×××××××××××××××××××××××××

# 参考文献

[1] 郑捷著. NLP汉语自然语言处理---原理与实践. 电子工业出版社

# 附录A 中文分词实现的源程序

import torch

import torch.nn as nn

from torchcrf import CRF

from torch.nn.utils.rnn import pack\_padded\_sequence, pad\_packed\_sequence