CMeEE 实体识别: 改进篇

519030910370 赵峻图, 519030910371 马楚航, 519030910346 李梓源

2022 年 6 月 15 日

1 摘要

实体识别任务作为信息抽取的一个重要组成部分，在近年来已经取得了重要的科研成果。对于医学领域的自然语言文本，例如医生的专业术语、检验报告、医学教材等等，其中蕴涵了大量的医学专业知识、专业用语。如果将实体识别技术和医学领域相结合，很显然可以显著提高医学科研工作的效率。在本次任务中，对于给定的一组医学专业的文本文档，目的是识别并提取出与医学方面相关的实体，将其归类到预先准备好的类别中。在本次作业中，我们在先前已经实现好的模型中进行了一定的改进，让模型的效果有了一定的提升。

2 相关工作

2.1 Baseline

表 1: 相关工作的测试集结果

模型	Linear	Linear+nested	CRF	nested+CRF
F1-Score	62.431	63.328	62.701	63.797

在之前的工作中，我们首先实现了模型的评测指标计算。在 Baseline 模型成功运行之后，我们又添加了嵌套处理功能以及 CRF 分类器，进一步提升模型效果。最终，我们得到的模型在测试集上的 $F1-score$ 结果如表1所示。

2.2 LSTM 模型

在介绍 LSTM 之前，我们不得不提到最经典的 RNN 模型，RNN 主要用于处理序列数据，其采用循环结构，每个单元的输出与当前输入和之前的隐状态向量有关，也就是说，每一个单元都会产生一个输出并传递给下一个单元，那这样很明显会带来一个问题：也就是在长序列上的表现欠佳。于是，LSTM 模型就应运而生了，它可以在长序列任务中表现得更好，其模型的示意图如图4所示。

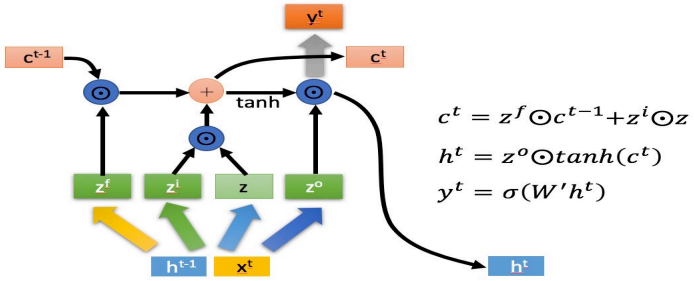

图 1: LSTM 模型结构

相比于 RNN 只传递了一个隐状态向量 h(hidden state)，LSTM 额外传递了一个隐状态向量 c(cell state)。对于模型中的每一个单元，首先将当前输入 x^t 与 h^{t-1} 组合，接着输出四个状态向量，这 4 个状态向量分别实现以下功能：1、对传递给当前单元的 c^{t-1} 进行选择性的“忘记”（丢失）；2、通过一个门控状态和一个状态向量得到当前结果；3、输出 h^t，即通过一个门控状态向量来决定当前得到的 c^t 转换为 h^t。显然是 c 的变化是累加的，而 h 的变化由 c 通过门控状态向量得到，因此，LSTM 引入了多个需要训练的网络，参数量大。

2.3 学习率调整

2.3.1 Layer-wise Learning Rate Decay (LLRD)

在 Revisiting Few-sample BERT Fine-tuning[2]中，作者归纳了 Layer-wise Learning Rate Decay (LLRD) 的概念. LLDR 的学习率设定方法在这里被如下描述:

- 设置顶层的学习率
- 使用一个倍增的衰减率, 从而从上到下逐层降低学习率.

而 Universal Language Model Fine-tuning for Text Classification[1]中, 作者给出了一个称为 Grouped LLRD 的学习率调整方案, 其策略是:

- 将层分组到不同的集合中，并对每组别的层应用不同的学习率.

2.3.2 Warm up

同时我们在 Scheduler 中加入了一个长度为 50 step 的 Warm-up 计划, 也就是说学习率会在 50 步内从 0 达到预设值, 之后应用常规的 Linear Scheduler 衰减到 0.

3 算法改进

我们在上文提到的已经实现好了的 baseline 模型中进行了一定的改进，并取得了一定的性能提升。

3.1 Loss 平均

首先在训练过程中，为了综合 CRF 模型和线性模型的各自的优点，我们尝试着同时使用 CRF 模型和线性模型。具体的，我们将使用嵌套处理的 CRF 模型和同样使用嵌套处理的线性模型产生的 loss 取了平均，得到了一个新的 loss 均值，这个均值 loss 同时包含了 CRF 模型和线性模型的信息，接着我们使用这个均值 loss 进行反向传播更新模型参数。进行 loss 的平均后，最终得到的模型性能相较 Baseline 得到提升。

3.2 加入 LSTM 模型

正如 2.2 节提到的，LSTM 模型有着可以在长序列任务中表现得更好的优点，于是我们尝试在我们的模型中加入 LSTM 块，进行进一步的改进。我们在原有的 CRF 结构的基础上，在 bert 层之后加入了一个 LSTM 层，并且将 LSTM 的输出结果与 bert 层的输出结果拼接起来再一同输入到线性分类器中。再加入了 LSTM 层之后，我们的模型性能相较 Baseline 同样有了进一步的提升。

3.3 学习率衰减方法

按照 [2]给出的方法, 我们在 **utils.AdamW_grouped_LLRD** 中实现了 LLRD. 代码实现和各层的学习率详细见附录 B 部分. 顶层学习率有所提高, 而将底层的学习率通过逐层的方式衰减到之前的 3e-5.

4 实验结果

模型	Score	CMeEE-F1	CMeEE-P	CMeEE-R	F1-Score(Valid)
Baseline	4.526	63.364	63.060	63.671	62.885
Baseline+biLSTM	4.595	64.330	62.881	65.847	62.868
Baseline+Avg-Loss	4.603	64.439	63.181	65.748	62.904
Baseline+LLRD+biLSTM	**4.640**	**64.955**	**62.035**	**68.165**	**64.083**

在实验平台上的提交结果截图见附录.

参考文献

[1] Howard, J., and Ruder, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers* (2018), I. Gurevych and Y. Miyao, Eds., Association for Computational Linguistics, pp. 328–339. 2.3.1

[2] Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., and Artzi, Y. Revisiting few-sample BERT fine-tuning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021* (2021), OpenReview.net. 2.3.1, 3.3

附录

A. 加入 LSTM 的实现

```
class CRFClassifier(nn.Module):
    def __init__(self, hidden_size: int, num_labels: int, dropout: float):
        #omit
        self.birnn = nn.LSTM(hidden_size, self.rnn_dim, num_layers=2,
                                                bidirectional=True,
                                                batch_first=True)

    #omit

    def forward(self, hidden_states, attention_mask, labels=None, no_decode=
                                            False, label_pad_token_id=
                                            NER_PAD_ID):
        sequence_output = self.dropout(hidden_states)
        sequence_output, _ = self.birnn(sequence_output)
        out = torch.cat((hidden_states,sequence_output),2)
        sequence_output = nn.functional.relu(out)
        loss, pred_labels = None, None
        #omit
```

B. LLRD

```
#==================================#
#init_lr=3e-5
#Pooler and regressor: init_lr*3.6=1.08e-4
#Layer 0,1,2,3: init_lr=3e-5
#Layer 4,5,6,7: init_lr*1.75
#Layer 8,9,10,11: init_lr*3.5=1.05e-4
#==================================#
def AdamW_grouped_LLRD(model, init_lr):
    # init_lr= 3e-5
    opt_parameters = []
    named_parameters = list(model.named_parameters())
    modelname = 'bert.'
    no_decay = ["bias", "LayerNorm.bias", "LayerNorm.weight"]
    set_1 = ["layer.0", "layer.1", "layer.2", "layer.3"]
    set_2 = ["layer.4", "layer.5", "layer.6", "layer.7"]
    set_3 = ["layer.8", "layer.9", "layer.10", "layer.11"]
    for i, (name, params) in enumerate(named_parameters):
        weight_decay = 0.0 if any(p in name for p in no_decay) else 0.01
```

```
        if name.startswith(modelname + "embeddings") or name.startswith(
                                                modelname + "encoder"):
            lr = init_lr # set_1
            lr = init_lr * 1.75 if any(p in name for p in set_2) else lr
            lr = init_lr * 3.5 if any(p in name for p in set_3) else lr
            opt_parameters.append({"params": params,
                                   "weight_decay": weight_decay,
                                   "lr": lr})
            continue
        if name.startswith(modelname + "regressor") or name.startswith(
                                                modelname + "pooler"):
            lr = init_lr * 3.6
            opt_parameters.append({"params": params,
                                   "weight_decay": weight_decay,
                                   "lr": lr})
    return transformers.AdamW(opt_parameters, lr=init_lr)
```

C. 实验平台提交结果截图

从上到下分别为 Baseline，Baseline+biLSTM，Baseline+Avg-Loss，Baseline+LLRD+biLSTM 的结果.