**基于深度学习的自然语言处理**

**第4次实验报告**

**一、实验名称**

将的论文复现

**二、实验日期**

2020.11.07 – 2020.11.20

**三、实验目的**

（1）熟悉NER的基本理论

（2）掌握NER的代码实现

**四、实验数据**

提供的开源的实验数据

**五、实验步骤及结果分析**

5.1 Chinese NER Using Lattice LSTM 网格结构LSTM 2018

5.1.1 看论文Chinese NER Using Lattice LSTM

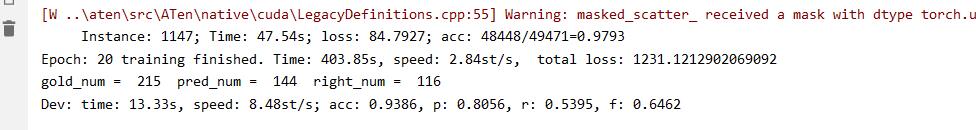
5.1.2 论文总结（几句话写清楚这篇论文讲了什么，解决了什么问题）

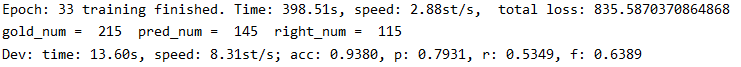
相比基于字符的方法（character-based），能够充分利用单词和词序信息；相比基于字的方法（word-based），不会因为分词错误影响识别结果。

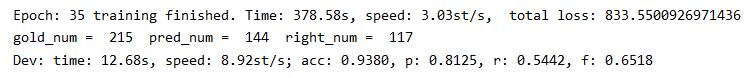
通过 Lattice LSTM 表示句子中的单词，将潜在的词汇信息融合到基于字符的 LSTM-CRF 中。

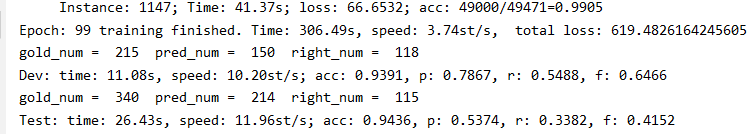
5.1.3 去GitHub上下载原文代码 <https://github.com/jiesutd/LatticeLSTM>

5.1.4 复现代码，将python2代码改成python3的，跑通代码里面的数据集









5.1.5 对实验结果进行分析

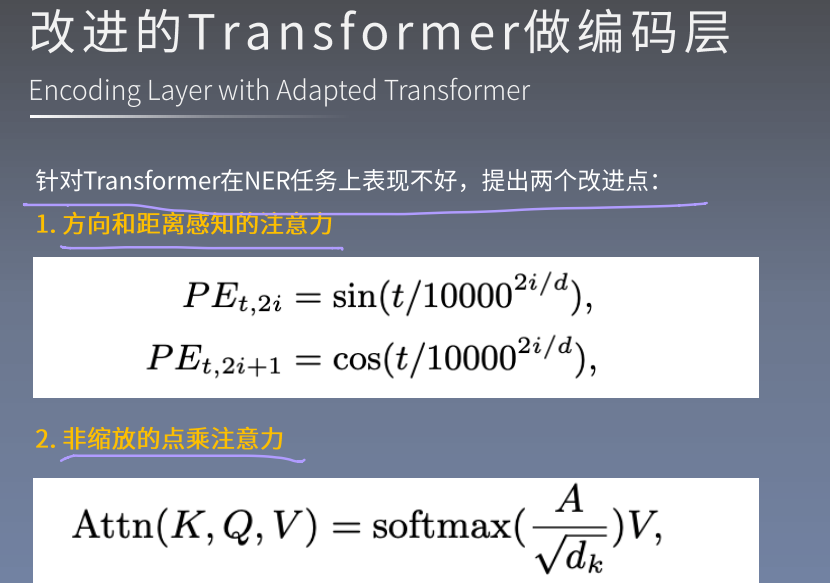
该模型在 20个epoch的时候已经收敛，实体识别测试集准确率达到94%，f1达到0.41%，召回率相对验证集低了很多，估计是因为样本不平衡造成的。

5.2 Adapting Transformer Encoder for Named Entity 2019

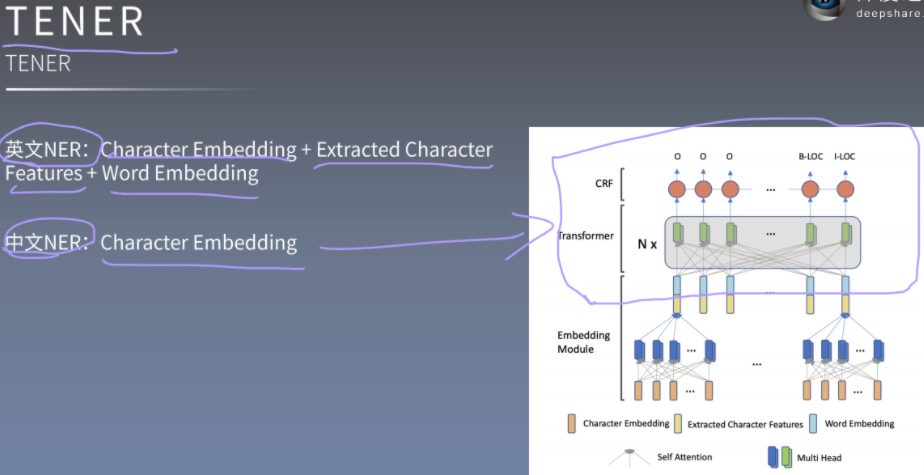
5.2.1 看论文Adapting Transformer Encoder for Named Entity

5.2.2 论文总结（几句话写清楚这篇论文讲了什么，解决了什么问题）

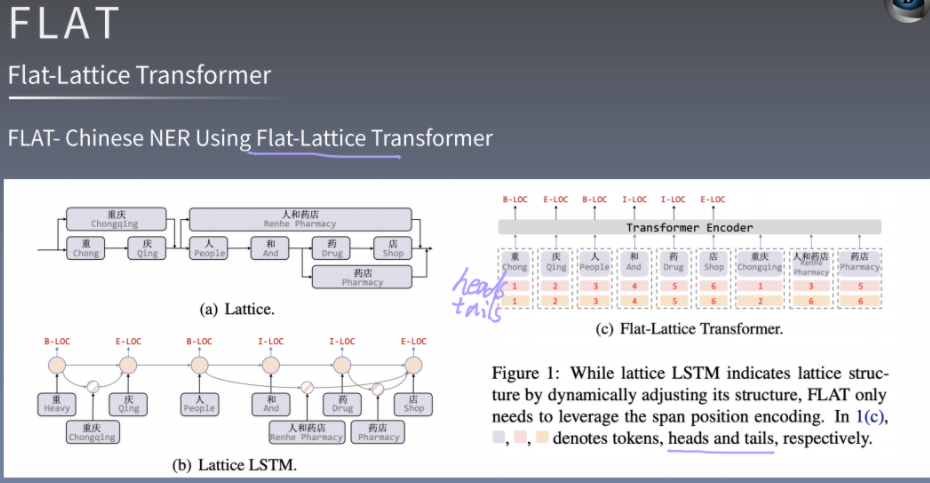
改进了Transformer编码层



改进Transformer应用于NER



FLAT:



关键点

1. 使用Transformer进行特征抽取,提高推断速度
2. 更改绝对位置编码

创新点

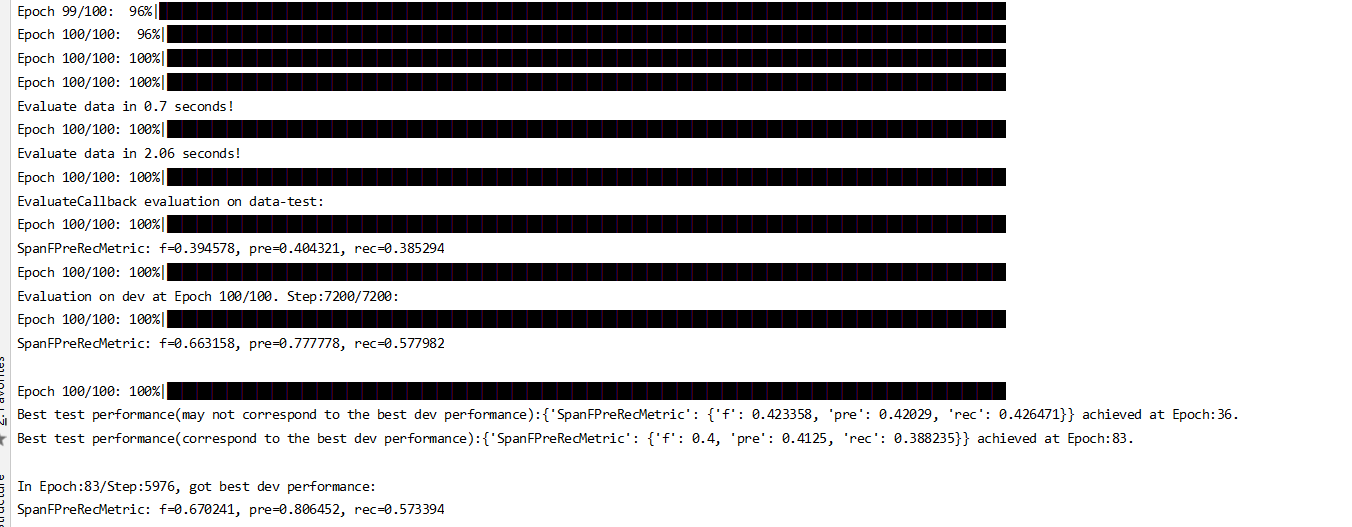
1. 优化Transformer,增加距离与方向感知的注意力
2. 使用非缩放的点乘注意力机制

5.2.3 去GitHub上下载原文代码 <https://github.com/fastnlp/TENER>

5.2.4 复现代码，跑通代码里面的数据集

num\_workers：使用多进程加载的进程数，0代表不使用多进程





5.2.5 对实验结果进行分析

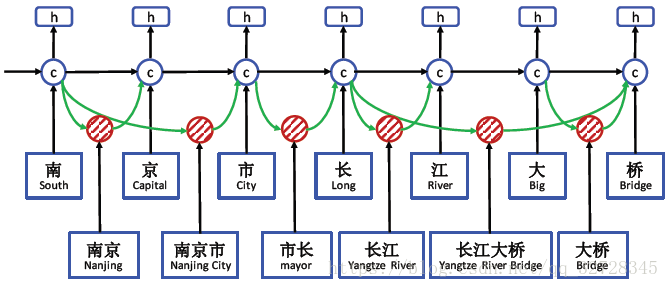
由上图看出大概在36epoch准确率时已经能打到最优效果，但准确率相对不高。

5.3 两种方法的分析与比较

两个模型优化角度不同，LatticeLSTM是从词的分割角度优化的，TENER是从Transformer优化的。最终效果LatticeLSTM效果更好，但是TENER模型训练速度更快。

**六、实验感想**

**LatticeLSTM：**



模型中有一些红色的 Cell，他们是句子中潜在词汇产生的信息，同主干 LSTM 相应的 Cell 连接起来就构成了类似于网格的结构，也就是题目中的 Lattice，这些红色 Cell 会在LSTM的该cell指向的词汇位置被考虑进去，而绿色箭头就意为代表前该蓝色cell包含了这些信息。而由于序列标记是以字符为级别，所以红色Cell 中没有输出门，Cell State 即为词汇信息。

**TENER:**

关键点

使用Transformer进行特征抽取,提高推断速度

更改绝对位置编码

创新点

优化Transformer,增加距离与方向感知的注意力

使用非缩放的点乘注意力机制