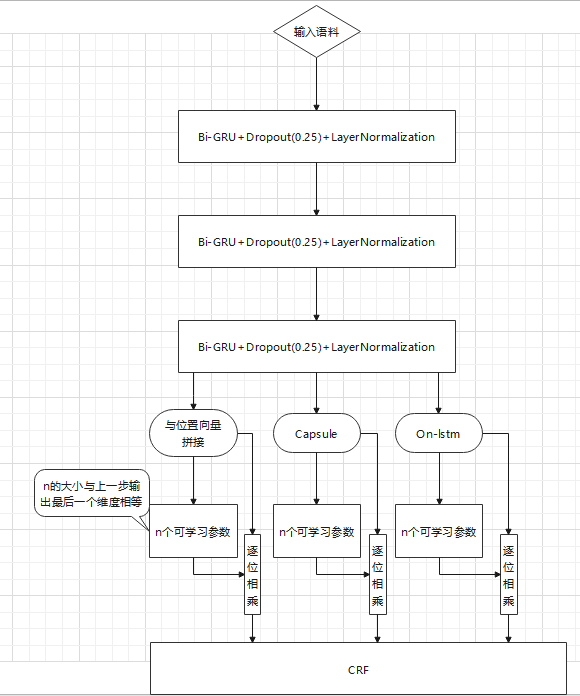
1. 流程图

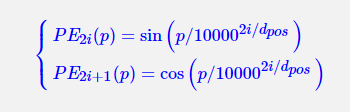


1. 按步骤具体说明
2. Bi-GRU+Dropout(0.25)+LayerNormalization

这三层Bi-GRU+Dropout(0.25)+LayerNormalization为编码层，层数可以增加也可以减少，层数越多肯定效果会越好，但是平衡训练时间和效果，选择了三层。在增加层数效果会有所提升，但是提升的很有限，而单批次训练所需的时间却增长的很多，而且随着层数的增加，所需使用的正则化手段要增多，这进一步增加了训练的时间成本。GRU的units参数在存在Bi的时候可以设置为输入向量最后一维的一半，当不存在Bi的时候可以设置的与输入向量最后一维相等，按经验来说该种方案可能不是最优方案但是比较省心，而且随着层数的增加，参数带来的差别也在逐渐缩小，若想获取最优方案，需要不断的更换参数训练模型，虽然可以写个循环自动化的完成参数的选择，但是很费时间，个人真的伤不起。

1. 位置向量

位置向量的实现方法有很多，我常用的有两种，第一种是利用公式直接生成，来自<<Attention is all you need>>，如图 1所示。第二种是将语料中每个词用其索引替换，比如{Attention, is, al,l you, need}替换后为{0, 1, 2, 3}，之后放入一个embedding层，该层的输出即当作位置向量的编码，优势是该embedding层也会得到训练，使得位置向量不是一成不变固定的编码。本次我使用的是第一种，经过实验第一种效果优于第二种。位置向量与语料的组合方式也有很多，这是实验了加、乘和拼接，效果最好的是拼接方法。



1. n个可学习参数

比如上一层的输出shape为(32,32,4)，那么n就为4。具体做法是将上一层的输出输入至4个unite参数为1的Dense中，将结果拼接起来，形成shape为(32,32,4)的输出，并与上一层的输出逐位相乘，最后与其他通道的结果相加，送入CRF层得出结果。

1. 优化器

优化器使用的是带自适应参数功能的Adam优化器，比较省心，结果不是最优但是不用耗费大量时间调参数。

1. 结果

所用语料为semeval 14和15，使用bert的预训练词向量f1最高为0.83，不使用bert的预训练词向量仅仅使用word2vec，f1最高为0.78。

1. 引用

以上用到的很多技术方法在苏老师的科学空间(<https://kexue.fm/>)中均有非常详细的讲解，这里也非常感谢苏老师。