## PJ2实验报告

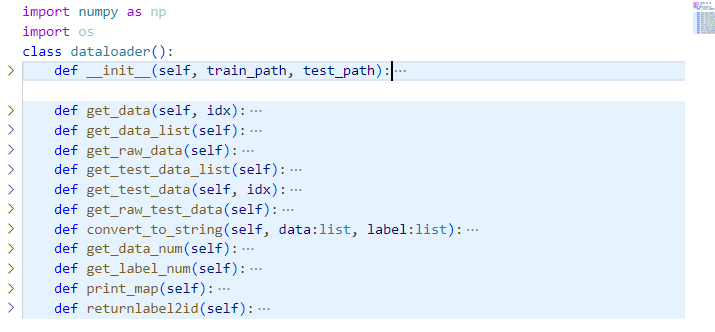
1. **任务简介**

三个实验都围绕命名实体识别（NER）任务展开，该任务的目标在于给定一定数量的句子，每个句子都不等长，需要给句子中的每个字打标签，（中文是字，英文则为单词），中文的标签包括非实体、名字、国籍、学历、头衔、组织、民族、专业、籍贯，英语的标签包括”not a entity”、”person entities”、”organization entities”、”location entities”、”other entities”，每个类下面都有B、I（英语）、B、M、E、S（中文）不同的标识用来表示该字或词是整个实体的哪一部分。

NER任务是NLP领域在以ChatGPT为首的transformer-based大模型出现之前的主要任务，本质这是个时序建模任务，我们需要对有先后顺序的文字进行建模，理解词与词之间的联系以及依赖关系，这样才能给每个单词打上正确的标签。

其价值也是巨大的，在符号主义流行的时代，对于后续的处理以及文本意思的理解都有巨大帮助。而当下，其也可以作为衡量序列建模能力的一个参考。

为了处理这个问题，我创建了一个类Dataloader，这个类可以接收训练材料的路径，对txt文本进行解析，将str类型的数据变为比较方便处理的数字（编码过程），同时可以回传训练集与验证集。



1. **Part 1： HMM实现命名实体识别（NER）任务**

该实验将时序建模看作为一个隐式马尔可夫过程，隐式马尔可夫过程即将不同时序的观测或是不同位置的单词建模为观测值，而这些观测值的背后有一个隐状态，隐状态之间会相互转换，而这也会决定观测值的变换。此外，马尔科夫链有个重要的假设，即这一秒的观测只与下一秒的隐状态、这一秒的隐状态有关，与其他事物无关。

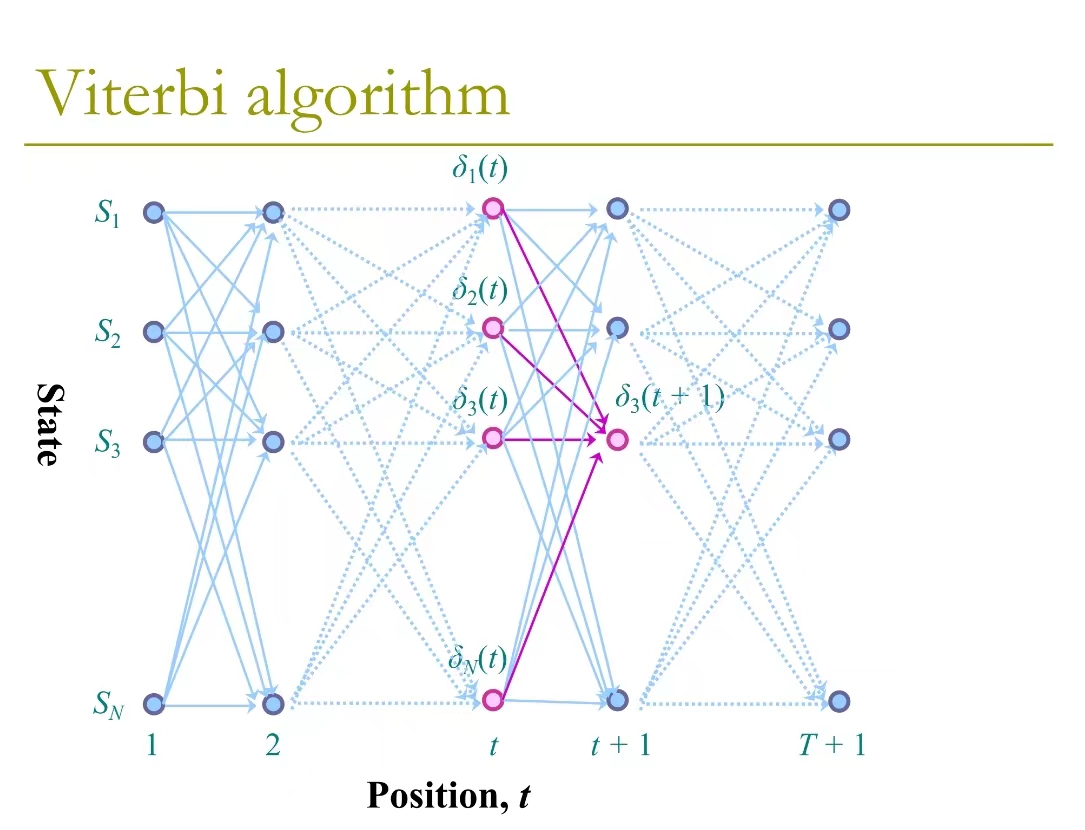
这个假设在我看来既合理又不合理，首先当下的状态确实可以决定下一秒的状态，这符合我们的认识，但过去的状态有可能造成影响，可以理解为过去的状态的信息给到了当下状态，所以当下状态已经包含了过去所有信息，而这样假设造成长序列遗忘问题也是这类模型的重要痛点。

基于我们上面的介绍，HMM模型通过构建初始矩阵、转移矩阵（隐状态间相互转换关系的度量）以及发射矩阵（隐状态以及观测现在之间概率的度量）来模拟了整个变换过程。

具体的HMM有三个重要问题，一个是给定HMM模型，如何衡量一个序列观测出现的概率、第二是给定一个序列观测，我们如何预测出隐状态的变换、第三个是给定观测、状态变换，如何找到最好的模型。

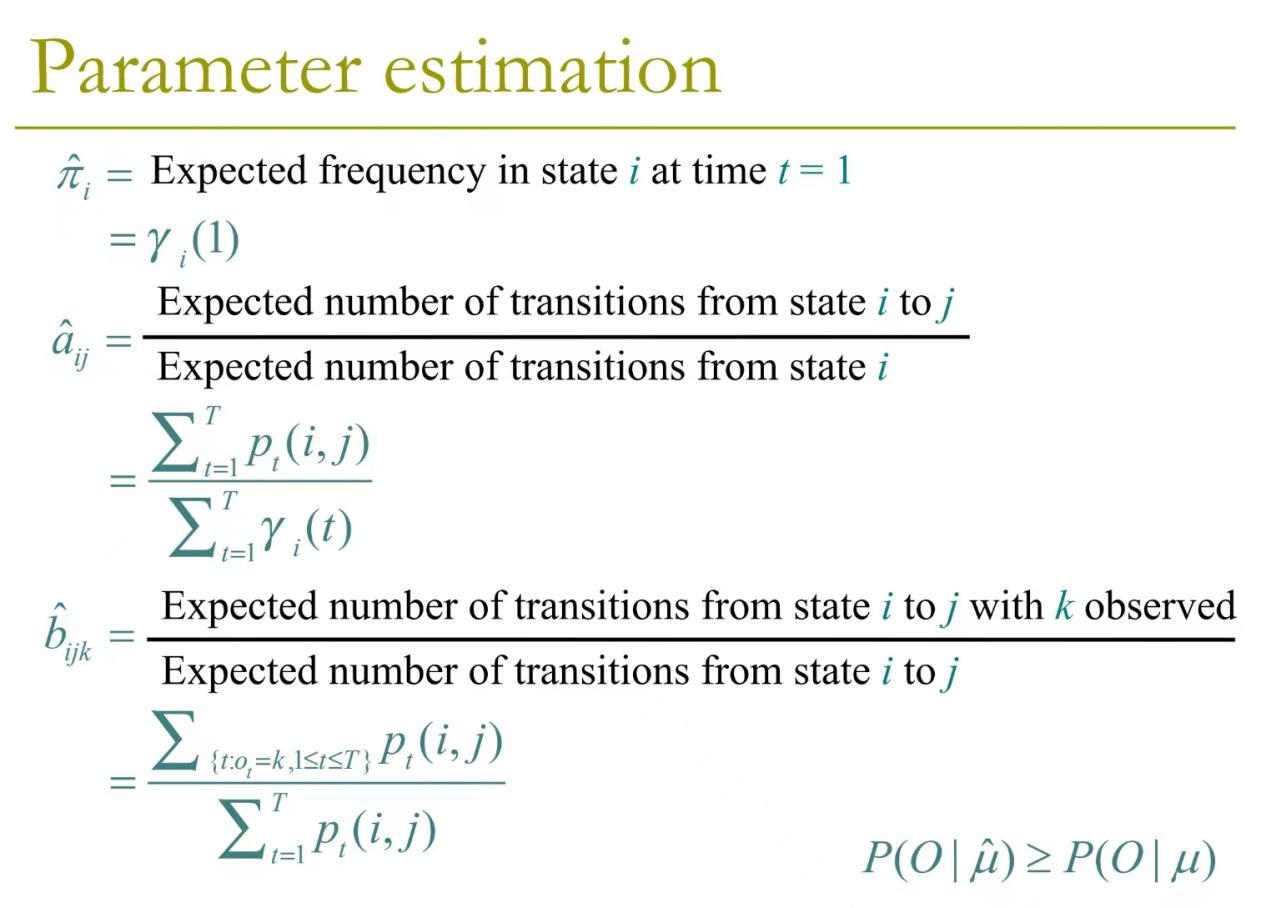
第一个问题可以通过逐步向后计算得到，利用转移矩阵以及发射矩阵即可。

第二个问题的做法其实就是第一个的反向，具体用维特比解码来做。为了更清楚地解释维特比解码，我们不妨想想传统的做法，我们希望基于O找到最可能的状态序列，于是我们可以列举所有的可能性，总数可能很大，然后一一比较，找到最大的即可，但这样时间复杂度极其吓人。那么维特比算法就可以缓解，其关键点在于发现很多路径其实根本不需要被计算，一定不会是最优的，有点类似搜索中的剪枝。具体做法：先是前向过程，沿着时间、句子方向向后计算，每一层都计算到达这个状态（隐状态）的最佳上一个隐状态，记录下最佳的前一个隐状态，再记录这样走的概率，而上一步从别的隐状态到达这个隐状态的就都不计算了，这样就可以大大加快计算。



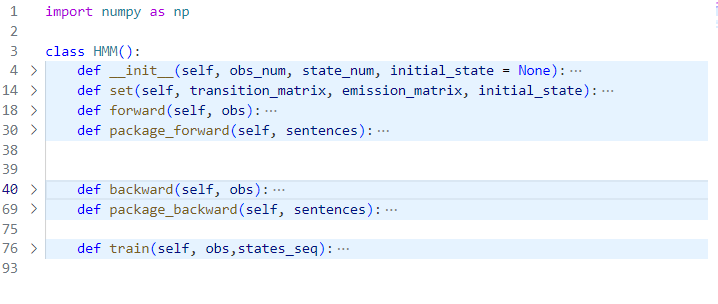
这样算到最后会发现得到一个最终的隐状态上的概率，选择最大的就是最后一个隐状态，再根据之前记录的回溯。

第三个问题其实更加简单，使用Expectation Maximization算法即可，具体到这个任务（由于该任务的data point是离散的），所以可以直接用统计的方法来做，直接统计训练样本中的数量即可得到初始化矩阵、转移矩阵以及发射矩阵。



落实到这个任务，其实可以把实体标签看成隐藏状态、文字结果看成观测结果。

具体在实现上，我构建了HMM这个类，其主要方法如下：



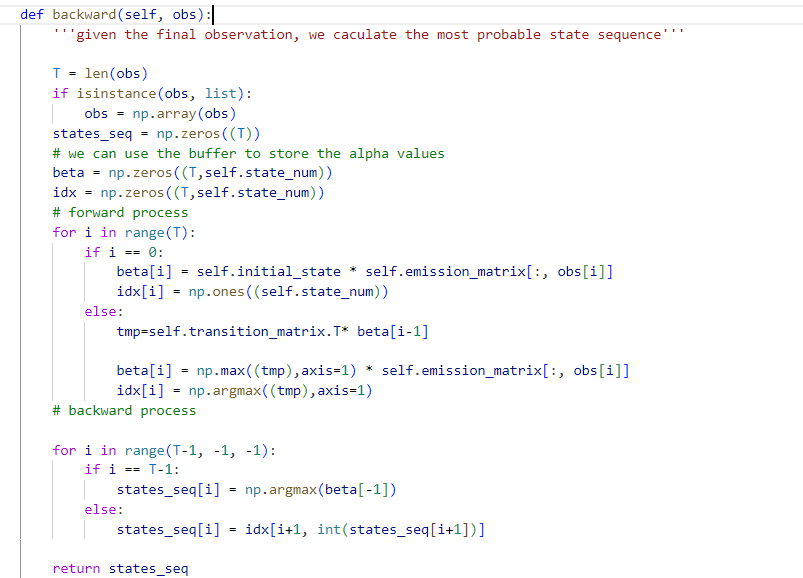
Forward方法处理HMM的第一个问题，即给定观测值输出概率。

Backward解决了第二个问题，即维特比解码。

Train解决了第三个问题，即使用EM算法对参数进行估计。

而加了package的表示输入可以从一个句子变成多个句子。

我具体介绍下核心部分维特比解码：我首先定义好beta与idx用来记录前馈过程的每一步概率值以及延伸路径，用于后续回溯。然后开始沿着T遍历，第一步直接用初始化概率依据观测、发射矩阵得到概率，后面则依据转移矩阵、发射矩阵来计算前一步所有状态到该状态的概率，取最大得到最优路径。前馈结束后倒序回溯即可。



1. **Part 2：CRF 实现命名实体识别（NER）任务**

条件随机场（CRF）模型是HMM的一个拓展，更具有普遍性。相比于HMM定死特征描述方法：初始化矩阵、转移矩阵、发射矩阵，CRF可以自己定义特征来学习到序列的特性，由此也开启了很长时间的以特征设计来解决NLP问题的范式。

具体方法也很简单，针对每一个字（或者英语中的单词），我们都为他设计观测值以及隐状态上的两种维度特征，特征的长度都可以自定义，因此特征的设计就对模型的表现效果起着至关重要的作用。

具体实现上我使用了sklearn的包，创建了类CRF。这个类的核心是model，可以通过sklearn\_crfsuite.CRF()来创建，给入参数：算法、最大迭代部署等即可。

之后最重要的是定义特征，这里我们需要对输入的句子每个词进行处理，得到每个词在上下文中的特征，以字典的形式给到模型。

我构建的特征设计前后共7个单词，前4个后3个，另外包含了该字是否为数字、是否大小写、在句子中相对位置等信息。

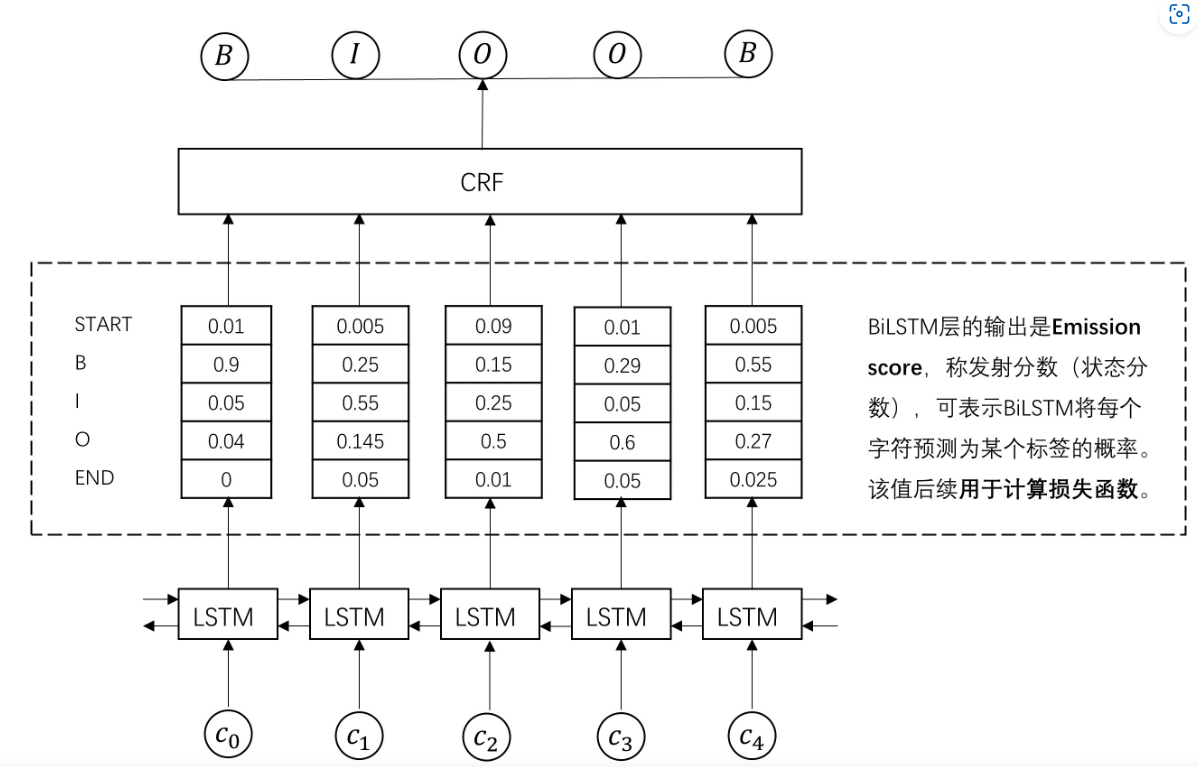
特征的定义方法如图：

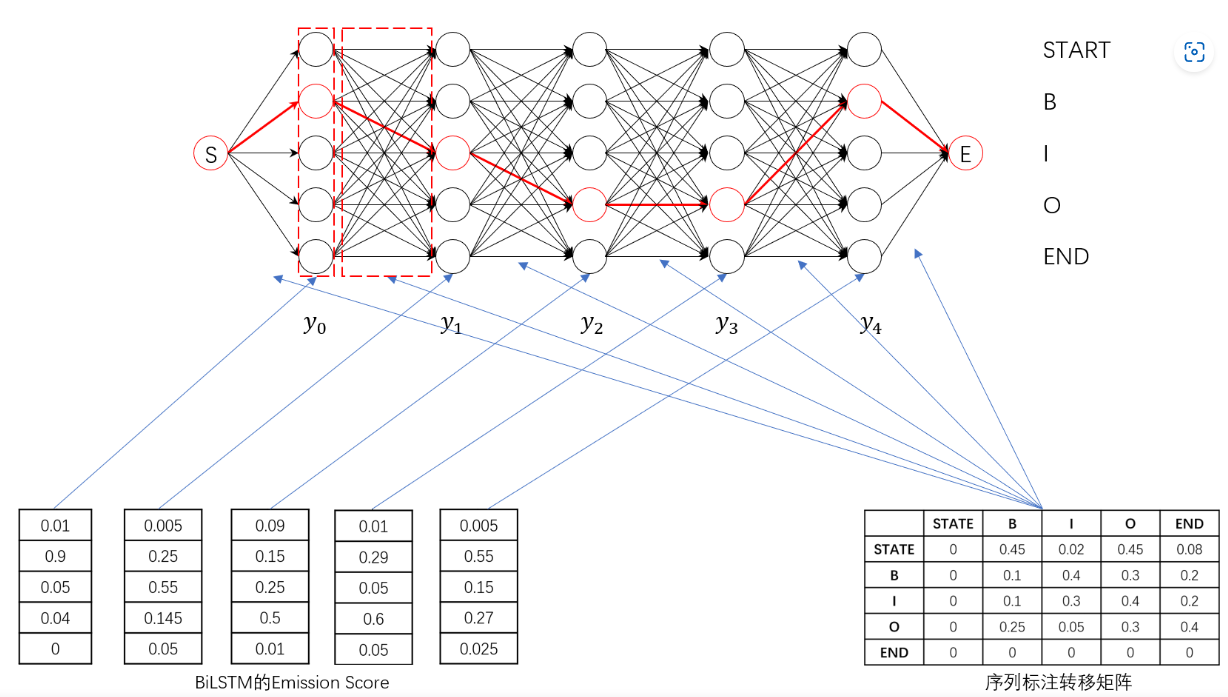


1. **Part 3：BiLSTM+CRF 实现命名实体识别（NER）任务**

该方法更进一步，采用神经网络加CRF的做法。

思想很简单，先将所有的单词变成embedding词向量，然后给到双向LSTM模型，双向LSTM模型能在单词层面对于数据进行建模，然后可以输出（T维度）同样长度的序列，序列中每个单元的长度就是tag的数量，表示该位置打不同标签的概率。那么到这里其实我们以及可以通过每个单元取最大概率来得到最后的tag，但这样会只关注于单词本身的依赖关系，没关注tag间的依赖关系。所以BiLSTM+CRF就将这些LSTM输出的序列重新作为输入给到CRF模型，这个CRF模型只有转移矩阵作为参数。将LSTM的输出作为emission score，我们便可以得到每条路径的得分，最后选择状态上最好的路径，也就得到了最后的tag。而优化方法也不采用之前的EM算法，而是通过梯度下降来同时优化CRF与LSTM。

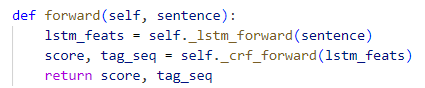




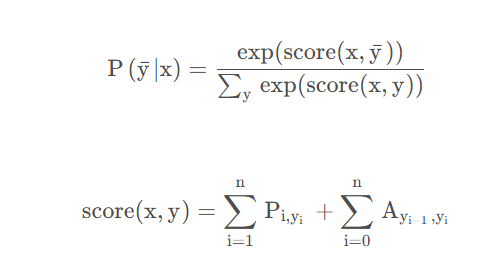
具体实现上，LSTM的部分可以直接调用pytorch的框架，这里不再过度赘述。

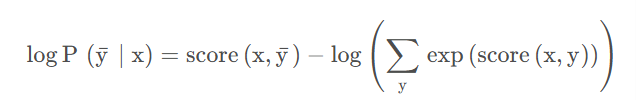
主要是CRF相关部分以及损失函数的计算部分。

我们首先定义正向过程，这包含以下函数forward、\_lstm\_forward、\_crf\_forward三个函数，forward调用后两个，实现两步走，先计算lstm得到emission score，再通过CRF得到最后结果。



其次是训练过程，先从最顶层的模块train开始介绍，train可以实现多epoch的计算以及训练过程的监视，train通过计算loss，再调用现有框架下的优化器，即可实现梯度下降。而loss的计算则由neg\_log\_likelihood实现，该函数调用三个子函数：\_lstm\_forward、\_forward\_alg、\_score\_sentence，lstm计算得到emission score，而\_score\_sentence则计算真实标签在当前参数下得到的概率。这时候就需要介绍我们的损失函数了，我们的损失函数是下面第一个公式，即正确tag序列的概率在所有tag序列概率之和（其实是取exp再求和）的比重，再取倒数其实就是我们的损失函数，为了计算这个值我们可以求它的log，就有最后一行的公式：





而我们的\_score\_sentence与\_forward\_alg正是分别计算了前后两项。