train3

吴琪

July 2024

本次培训的题目在于学习循环神经网络在自然语言处理中的应用。循环神经网络（Recurrent Neural Network），即 RNN，是用于处理序列数据的神经网络。在前面学习到的卷积神经网络中，单次输入之间并无联系，每一次仅仅关注了输入的独立特征。而 RNN 为了更有效地处理序列信息，需要进一步考虑序列中词与词之间的联系对于模型输出的影响。与传统的神经网络相比，循环神经网络（RNN）在结构中增加了循环部分，这个循环部分使信息能够在网络的同一层之间进行传递，利用前一部分的信息来影响后续的处理和输出，使网络能够进一步处理序列数据。因此，RNN 在生成输出时，不仅要考虑此时的输入信息，还要考虑从前一步传递过来的信息。RNN 模型结构以及展开图如下所示：

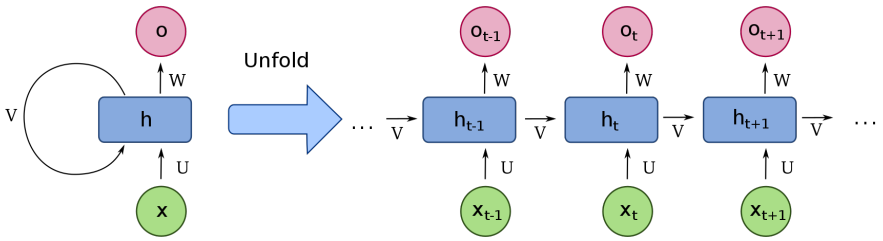

图 1: RNN structure and expansion diagram.

如图所示，x 代表模型的输入，x_t 是一个向量，代表在时间点 t 的输入信息。状态 h_t 由当前时间点 t 的输入信息 x_t 和上一时间点的隐藏层输出 h_{t-1} 两部分组成。模型的整体公式如下：

$$s_t = f(U_{x_t} + W_{s_{t-1}}) \tag{1}$$

RNN 虽然在处理序列数据方面有很大的优势，但也同样存在局限性。在 RNN 中，随着时间步的增加，如果梯度的导数值始终小于 1，累积梯度会越来越微小，最终导致梯度消失（sigmoid）。相反，如果累积梯度异常大，则可能会出现梯度爆炸，导致数值过度膨胀（tanh）。正是由于训练 RNN 模型时存在梯度消失或爆炸的问题，网络无法有效地学习长距离依赖关系。所以对于很长的序列，RNN 往往无法准确使用序列开头的信息，从而无法对整个序列正确建模。

为了解决 RNN 模型存在的梯度消失问题，设计出了长短期记忆网络（LSTM）。LSTM 通过引入一系列精妙的门控机制，可以有效地控制信息的保留与遗忘，以此维持长距离的梯度流动。相比较 RNN 中仅有线性变换和激活函数，LSTM 结构就复杂了许多，将多个输入经过不同的处理，从而得到相应的隐藏层信息与输出。LSTM 结构如下图所示：

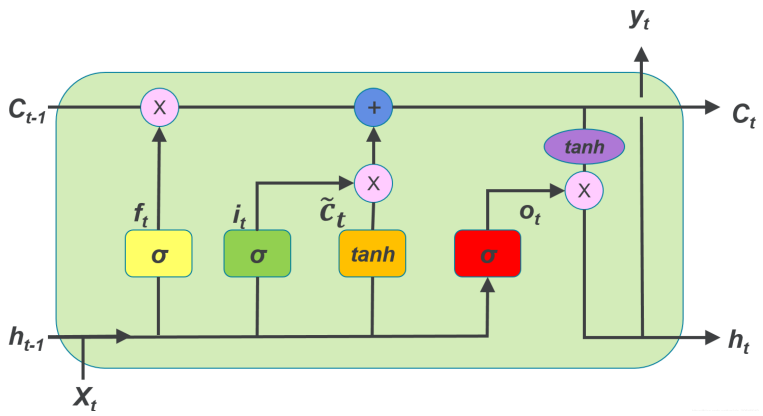

图 2: Structure diagram of LSTM.

在 LSTM 模型中，X_t 代表当前时间步的输入，h_{t-1} 表示 $t-1$ 时刻的隐藏层结果，C_{t-1} 表示 h_{t-1} 未经处理前的数据。同时使用 h_{t-1} 和 C_{t-1} 是为了防止重要的信息的丢失，保证在迭代过程中信息的完整与连续。模型的具体过程如下：首先将 t 时刻的输入数据 X_t 和上一时刻的隐藏层结果 h_{t-1} 拼接成一个向量送入神经网络层，计算得到 f_t，再与图中上一时刻的 C_{t-1} 这部分信息按位进行乘积，这一个操作确定了数据 C_{t-1} 的保留与舍弃，f_t 也被成为遗忘门。这一段使用的激活函数是 sigmiod，用于帮助调节神经网络的值，将计算结果压缩在 0 和 1 之间。随后，输入数据 X_t 和上一时刻的隐藏层结果 h_{t-1} 再次计算，用于筛选 X_t 和 h_{t-1} 经过神经网络层 $tanh$ 处理后的向量，这部分也被称为输入门，输入门的结果再与经过第一次运算后的 C_{t-1} 按位进行加法操作，从而得到了 C_t。最后是输出门，通过 X_t 和 h_{t-1} 生成的门这样一步操作，来对 C_t 的信息进行取舍，最终得到了当前时间步的隐藏层信息 h_t，同样也是当前时间步的结果 y_t。

LSTM 模型通过门实现了长时间的记忆功能，但是这一复杂的结构也增加了模型的训练难度，而门控循环单元（GRU）优化了 LSTM 中的门机制，仅通过重置门（reset gate）和更新门（update gate）即可控制信息流，使模型在减少参数数量的同时，保持了长期依赖学习的能力。

GRU 和之前分析过的 LSTM 中的门控一样，首先计算更新门和重置门的门值，分别是 z_t 和 r_t，计算方法就是使用 x_t 与 h_{t-1} 拼接进行线性变换，再经过 sigmoid 激活。将该结果作为重置门，作用在 h_{t-1} 上，代表控制上一时间步传来的信息有多少可以被利用，再与初始的 x_t 拼接，经过 tanh 激活，得到初步的 h_t。把更新门的门值会作用在刚才的 h_t 上，1-门值会作用在 h_{t-1} 上，最后将两者的结果相加，得到最终的隐含状态输出 h_t。GRU 模型在原 LSTM 的第二个门控单元中，通过 “1-” 操作从而对两部分进行了互补，旧信息遗忘的部分通过新信息补充，从而减少了模型的参数，依旧达到了和 LSTM 相同的能力水平。GRU 模

型的具体结构如下图所示：

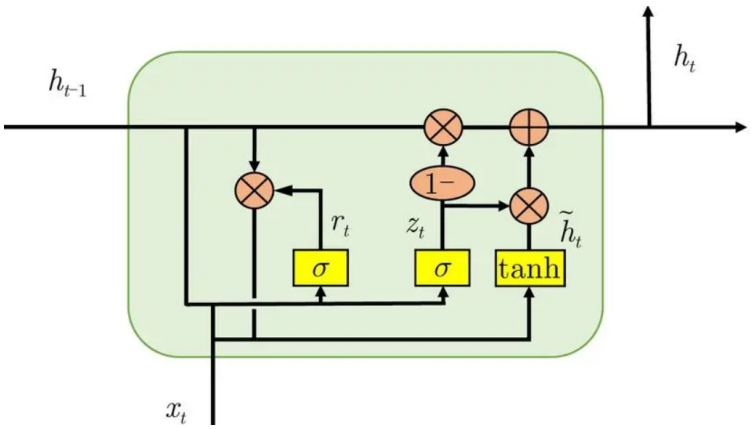

图 3: Structure diagram of GRU.

在了解模型之间的关系后，借助 pytorch 中 LSTM 模块和 GRU 模块分别实现二分类模型。nn.LSTM 和 nn.GRU 分别是 pytorch 定义好的循环神经网络层，通过它们与 Embedding 层、线性层结合，就构建起了对应的模型。其中 Embedding 层通过将文本转换为连续向量，将离散的输入特征映射到低维密集向量空间，实现降维的作用。在加载 IMDB 数据集上，原本使用了 torchtext.datasets 中提供的 IMDB 方法，但是在使用过程中总是报错："Package 'torchdata' not found." 在更新 pytorch 环境后函数依旧无法正常使用，所以使用了 keras.datasets，通过 imdb.load_data 函数加载 IMDB 数据集，该函数会对数据集进行预处理，将文本转换为整数序列，每个整数表示一个单词在词汇表中的索引，同时对于每条评论将其截断或填充到指定的最大长度。最终得到了包含文本以及对应标签的训练集与测试集，再经过序列填充，将它们转化为 Dataloader。

设定好相应的超参数，vocab_size 表示输入文本的词汇表大小，embed_size 表示词嵌入的维度，hidden_size 表示隐藏层的大小，num_classes 表示分类类别数，num_epochs 设定为 15，随后对两个模型进行训练，记录训练集与测试集上的 loss，每个模型的两个 loss 如下图所示：

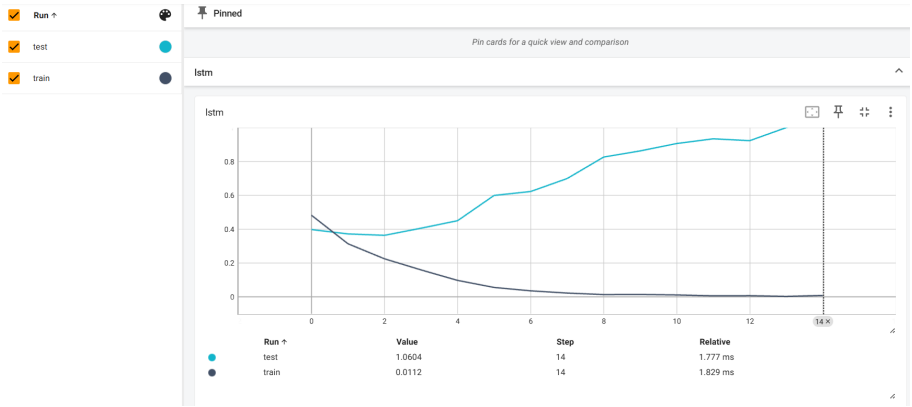

图 4: Loss of LSTM model during training and testing phase

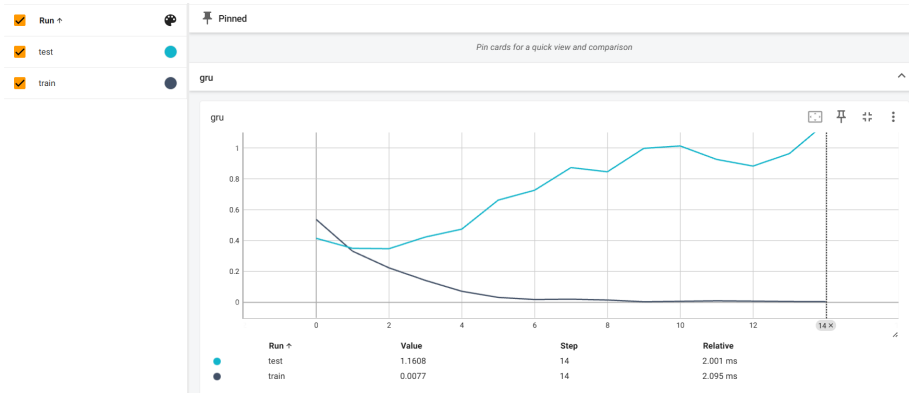

图 5: Loss of GRU model during training and testing phase

训练过程中，在每个 epoch 后，都将模型在测试集上进行预测。同时，记录每个分类预测正确个数与总个数。两个模型在每个类别上的 accuracy 如下表所示：

表 1: Accuracy of the two models in each category

category	LSTM(%)	GRU(%)
class 0	87.83	85.32
class 1	77.42	83.44
all	82.62	84.38

在训练过程中，同样记录了每个 epoch 下 LSTM 和 GRU 在训练集上 loss 的变化曲线，如下图所示：

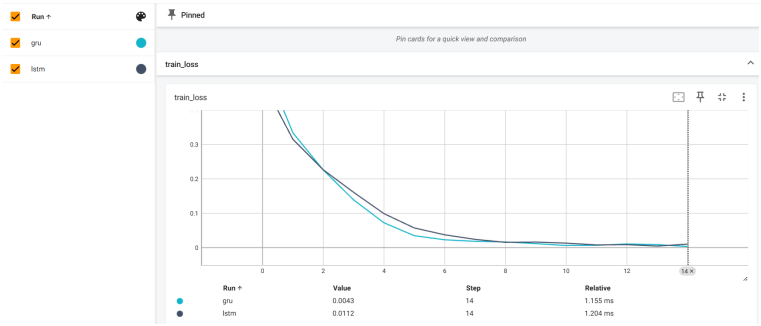

图 6: Loss curve of two models on training set.

分析时间，RNN 是最经典的循环神经网络，它结构相对更简单，在处理短文本时表现尚可，但在处理长文本时容易出现梯度消失或梯度爆炸问题，导致无法有效捕捉长距离依赖关系，使模型的性能下降。LSTM 通过引入记忆单元和门机制，改善了 RNN 在长序列上的表现，能够更好地捕捉长距离依赖关系，但是结构随之变复杂，训练时间也相对较长。GRU 是 LSTM 的简化版本，具有类似的优点和表现，但是它的结构更加简单，计算效率更高。LSTM 与 GRU 二者结构十分相似，新的记忆或输出都是根据之前状态及输入进行计算。但 LSTM 由三个门所组成，输入门用于更新细胞状态，遗忘门决定应丢弃或保留哪些信息，输出门则

用来确定下一个隐藏状态的值。而 GRU 中更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，重置门用于控制前一状态有多少信息被写入到当前的隐藏信息。

最后学习一点不同的编码方式，One-hot 编码是一种将离散变量转换为数值型的方法。对于一个词汇表大小为 V 的文本，每个单词都被表示为一个长度为 V 的二进制向量，向量中只有对应单词所在位置的元素为 1，其他元素都为 0。那么一个长度为 n 的句子就是由 n 个长度为 V 的二进制向量组成。One-hot 编码简单直观，但是会产生高维稀疏向量，不能很好地捕捉单词之间的语义关系。而字符索引编码是将单词映射到一个连续的整数的方法，通过构建一个词汇表，为词汇表中的每个单词分配一个唯一的整数，然后将输入文本中的每个单词替换为其对应的整数就形成了一个语句，字符索引编码生成的向量维度较低，并且可以保留单词之间的相对位置信息，但是可能无法捕捉单词之间的语义关系。而通过 Embedding 的方法，就可以进一步让相互独立的向量变成了有内在联系的关系向量。