高级实训任务三:文本情感分类

1.任务描述

- 将循环任务（RNN）应用在图像分割任务上，我们需要对网络结构进行设计。
- 需要提交博客报告以及GitHub代码仓库。
- 可选的任务：文本情感分类（正向、负向）。
- 可选的网络结构：GRU、LSTM。
- 可选的数据集：
  - imdb数据集：https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
  - 烂番茄数据集：https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/overview
- 可选深度学习框架：Tensorflow、PyTorch、Keras。
- 完成期限：1月21日
- 提交结果：项目报告、答辩幻灯片、相关代码和测试用例。

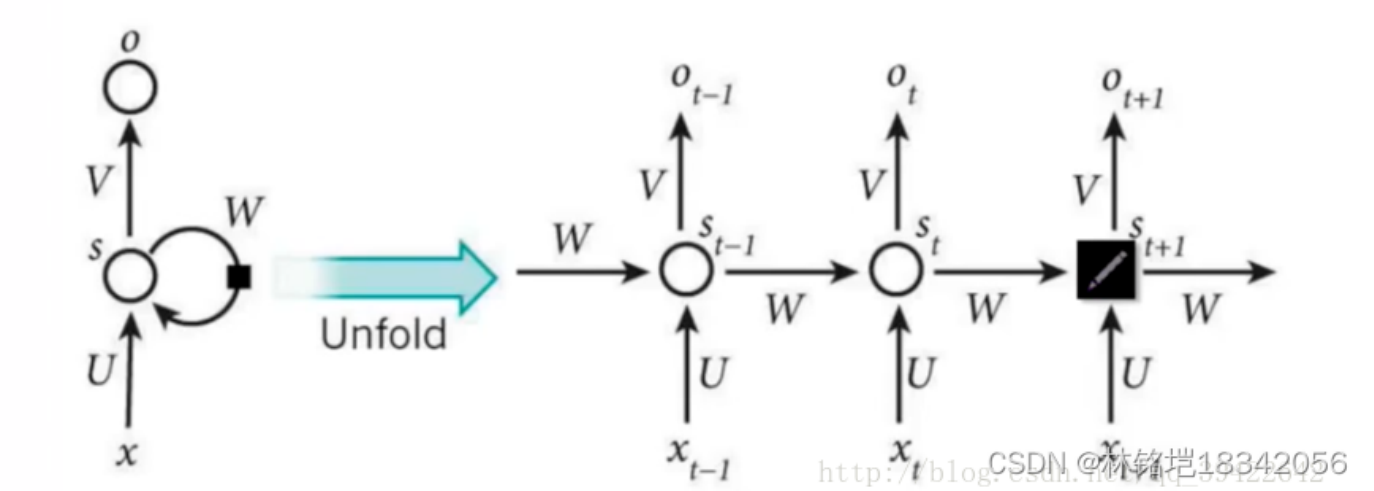
2.任务选择

将循环任务(RNN)应用在图像分割上，需要对网络结构进行设计。
任务选择：文本情感分类（正向，负向）
选择的网络结构：LSTM
语言：python
框架选择：pytorch（主框架，构建网络结构）
其他辅助框架：pickle（python 的文件库。由于数据集的一部分放在pkl文件里，需要pickle库进行读取）
tqdm （UI方面的库，用于添加进度条，方便观察计算的进度）
数据集：aclImdb（大型电影评论数据集）

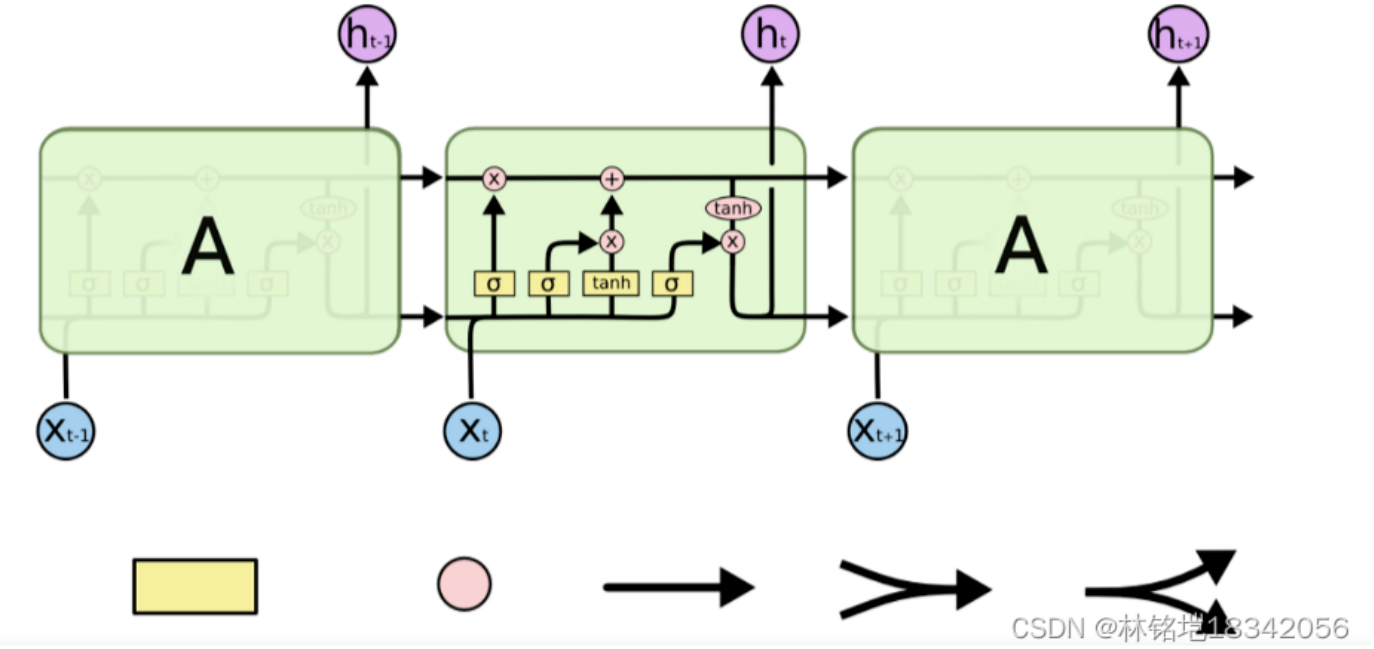
3.任务开始准备

- **循环任务（RNN）应用在图像分割任务原理**
  循环神经网络的应用场景比较多，比如暂时能写论文，写程序，写诗，但是，（总是会有但是的），但是他们现在还不能正常使用，学习出来的东西没有逻辑，所以要想真正让它更有用，路还很远。
  一般的神经网络应该有的结构：

- **使用循环神经网络原因**：
  无论是卷积神经网络，还是人工神经网络，他们的前提假设都是：元素之间是相互独立的，输入与输出也是独立的，比如猫和狗。但现实世界中，很多元素都是相互连接的，比如股票随时间的变化，我们是根据上下文的内容推断出来的，但机会要做到这一步就相当得难了。因此，就有了现在的循环神经网络，他的本质是：像人一样拥有记忆的能力。因此，他的输出就依赖于当前的输入和记忆。
- **RNN的网络结构及原理**



  其中每个圆圈可以看作是一个单元，而且每个单元做的事情也是一样的，因此可以折叠呈左半图的样子。用一句话解释RNN，就是一个单元结构重复使用。
- **LSTM原理**
  1.LSTM内部结构：



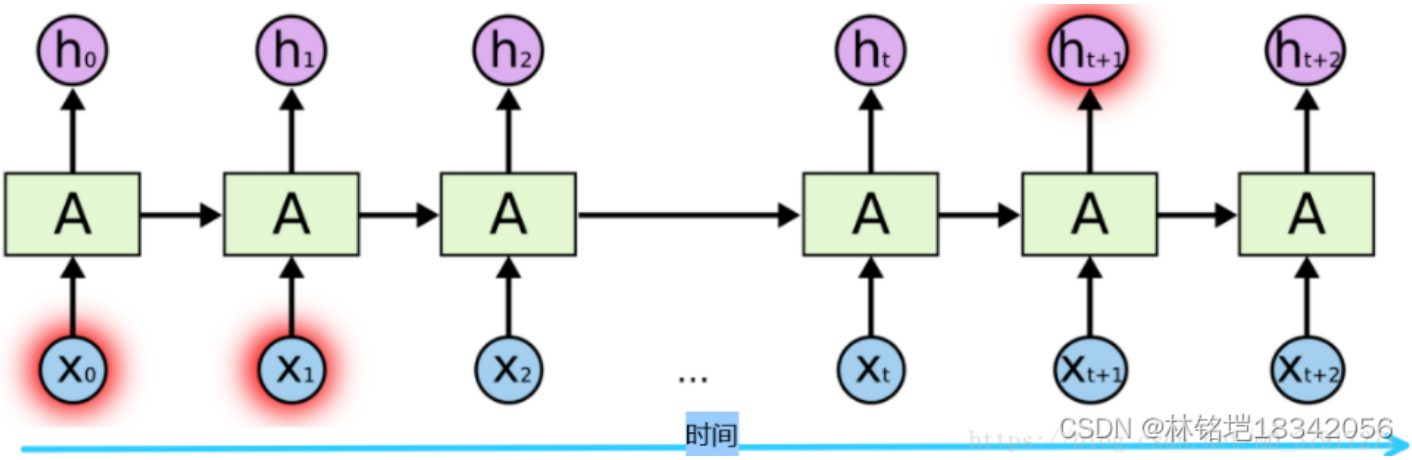
LSTM看上去就是这样一种效果，一个一个首尾相接，同一层的会把前面单元的输出作为后面单元的输入；前一层的输出会作为后一层的输入。细胞状态

2.LSTM 的关键就是细胞状态，水平线在图上方从左到右贯穿运行。

细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易

左面的乘号是一个乘法操作，右面的加号就是普通的线性叠加。
LSTM规避了标准RNN中梯度爆炸和梯度消失的问题，所以会显得更好用，学习速度更快

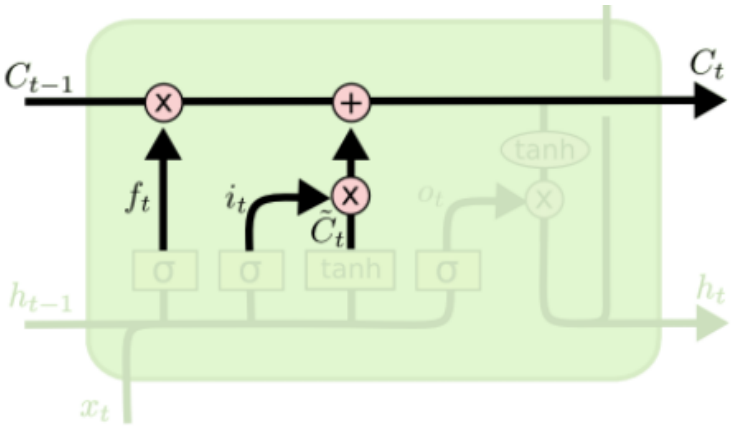
下图是最基本的LSTM单元连接起来的样子



3.LSTM的门结构

遗忘门：遗忘门层决定我们会从细胞状态中丢弃什么信息。该门会读取ht−1 和xt ， 输出一个在 0到1之间的数值给每个在细胞状态 Ct−1 中的数字。1 表示“完全保留”，0 表示“完全舍弃”。
[ht−1,xt]代表把两个向量连接起来。

更新门：C t 表示新的输入带来的信息，tanh这个激活函数讲内容归一化到-1到1;
it用于选择保留新信息的哪个部分。ft * Ct-1表示过去信息有选择的遗忘（保留），
Ct 表示新信息有选择的遗忘（保留），两者相加得到新状态Ct。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

输出门：

4.实验过程和结果

代码：

1.读取imdb数据集：

```
max_f=10000
(x_train, y_train),(x_test,y_test) = imdb_data(num_words = max_f)

maxlength = 400
x_train = sequence.pad_sequences(x_train,maxlength = maxlength)
x_test = sequence.pad_sequences(x_test,maxlength = maxlength
```

2.构建LSTM训练模型

```
model = Sequential([layers.Embedding(max_f,32),
                                        layers.LSTM(32),
                                        layers.Dense(1,activation="sigmoid")
                                        ])
model.compile(optimizer="rmsprop",loss='binary_crossentropy',metrics=["accuracy"])

history = model.fita(x_train , y_train, epochs=10,batch_size = 128,
                                        validation_data = (x_train , y_train),callbacks=callbac
```

3.实验结果：

```
Epoch 1/10  25000/25000 [==============================] - 313s 12ms/sample - loss: 0.4751 - ac
Epoch 2/10  25000/25000 [==============================] - 265s 11ms/sample - loss: 0.2848 - ac
Epoch 3/10  25000/25000 [==============================] - 251s 11ms/sample - loss: 0.2362 - ac
Epoch 4/10  25000/25000 [==============================] - 172s 7ms/sample - loss: 0.2011 - acc
Epoch 5/10  25000/25000 [==============================] - 145s 6ms/sample - loss: 0.1855 - acc
Epoch 6/10  25000/25000 [==============================] - 148s 6ms/sample - loss: 0.1686 - acc
Epoch 7/10  25000/25000 [==============================] - 143s 6ms/sample - loss: 0.1542 - acc
Epoch 8/10  25000/25000 [==============================] - 145s 6ms/sample - loss: 0.1438 - acc
Epoch 9/10  25000/25000 [==============================] - 147s 6ms/sample - loss: 0.1351 - acc
Epoch 10/10  25000/25000 [==============================] - 143s 6ms/sample - loss: 0.1293 - ac
```

实验结论：可以看到经过10轮训练后，训练集准确度为0.9561，测试集准确度为0.9655，训练过程中测试集准确度最高为0.9655。可以看出随着周期数的增多，准确度也在不断增高。

总结

LSTM和其他神经网络用途类似，主要用于分类或预测。能够改善了RNN中存在的长期依赖问题；LSTM的表现通常比时间递归神经网络及隐马尔科夫模型（HMM）更好；作为非线性模型，LSTM可作为复杂的非线性单元用于构造更大型深度神经网络。