

实验 4-循环神经网络实现

学号：1190201215

姓名：冯开来

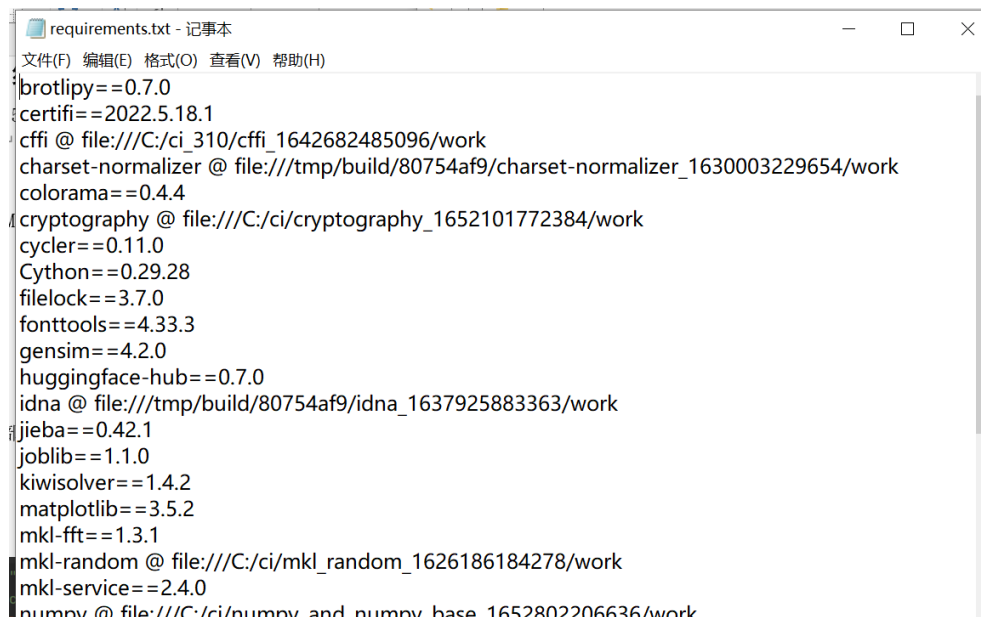
一、 实验目的：

利用 Pytorch 自己实现 RNN、GRU、LSTM 和 Bi-LSTM
利用上述四种结构进行文本多分类
任选上述一种结构进行温度预测

二、 实验环境：

(在文件 requirements.txt 已经体现)

Python==3.9; Pytorch==1.11



```
requirements.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
brotli==0.7.0
certifi==2022.5.18.1
cffi @ file:///C:/ci_310/cffi_1642682485096/work
charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
colorama==0.4.4
cryptography @ file:///C:/ci/cryptography_1652101772384/work
cycler==0.11.0
Cython==0.29.28
filelock==3.7.0
fonttools==4.33.3
gensim==4.2.0
huggingface-hub==0.7.0
idna @ file:///tmp/build/80754af9/idna_1637925883363/work
jieba==0.42.1
joblib==1.1.0
kiwisolver==1.4.2
matplotlib==3.5.2
mkl-fft==1.3.1
mkl-random @ file:///C:/ci/mkl_random_1626186184278/work
mkl-service==2.4.0
numba @ file:///C:/ci/numba and numba base 1652802206636/work
```

(requirements.txt 部分截图)

三、 实验内容：

1. 超参数定义

```
parser = argparse.ArgumentParser("lab4")
parser.add_argument("--device", default="cuda:0")
parser.add_argument("--model", default="GRU") # RNN GRU LSTM Bi-LSTM
parser.add_argument("--epochs", default=2)
parser.add_argument("--batch-size", default=30) # 本次实验没有用到过
parser.add_argument("--seed", default=42)
parser.add_argument("--dataset", default="shopping") # shopping or climate
parser.add_argument("--output-path", default="./result/")
parser.add_argument("--hidden-size", default=128)
parser.add_argument("--input-size", default=128, type=int) # if climate, only be 5
parser.add_argument("--optimizer", default="adam")
parser.add_argument("--test", action="store_true", help="Only run test")
```

这里默认由 GPU 进行加速训练。

本次实验 batch-size 没有多大用处，因为构造数据集的时候默认为 1。
路径 output-path 包含了模型和 loss 曲线的图片。

隐空间 hidden-size 是自己可以随意设定的。

输入的维度 input-size 在运行文本多分类任务可以任意大小，在做天气温度预测任务只能是 5，后续会说明。

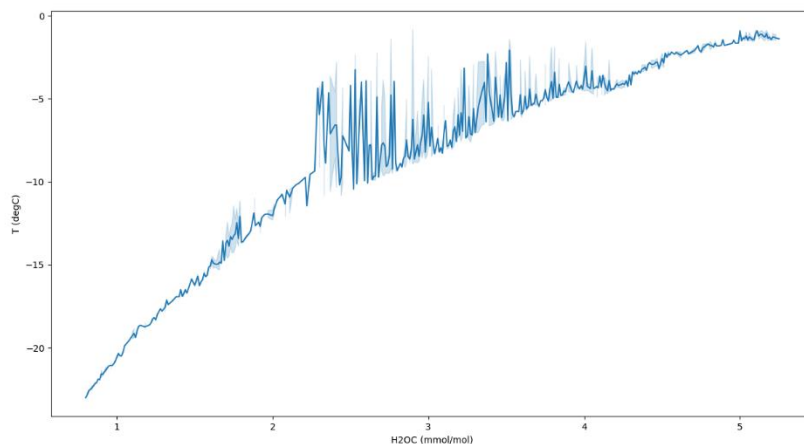
2. 数据集构造

本次实验构造数据集特别麻烦。

文本多分类-onling_shopping_10_cats 数据集。该数据集构造的过程首先使用 pandas 读入文件，对每一条数据进行遍历取出中间的评论和物体类别。对于每一条评论使用 jieba 分词，在使用所有评论训练 gensim 库中的 Word2Vec 模型，再利用该模型对每一个词向量进行 embedding 和映射。这样对于每一条评论，每个词向量是 input-size 维，但是每一条评论的词个数不一样，所以评论的维度不一样，所以也就导致了后续训练模型的时候实际 batch-size 只能为 1。

对评论进行映射之后，我们再一次遍历所有评论，mod5=4 为验证集，mod5=0 为测试集，其余为训练集。最后调用 DataLoader 实现数据集迭代器的构造。

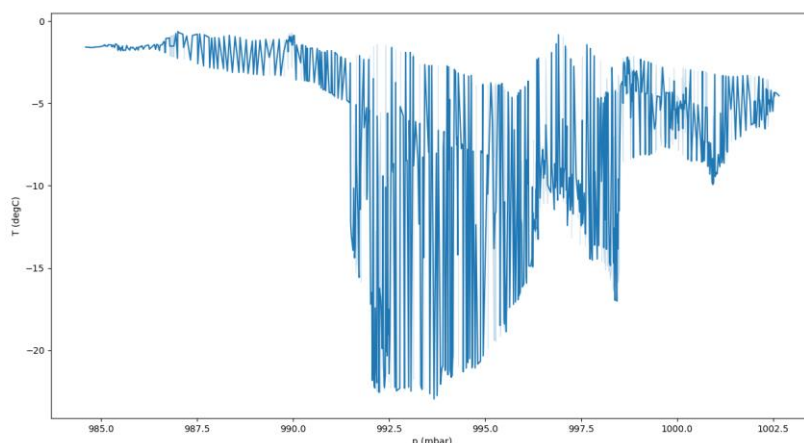
温度预测-jena_climate_2009_2016 数据集。首先通过读入文件可以发现，除了温度还有很多别的因素比如风速气压等等。但是将这些因素都归为可以影响温度的显然不合适，所以可以大致的画出一些温度与这些因素的曲线图。如下图



T 和 H2OC 的关系

在我们得到所有这些因素和 T 的关系后，我们可以做一些筛选，这里我取的都是呈线性关系的图片。如下图 T 和 p(mbar) 的关系，不是线性，所以我近似的认为 p 并不能影响到 T。最后通过筛选，我选择了 'H2OC (mmol/mol)', 'rho (g/m**3)', 'sh (g/kg)', 'Tpot (K)', 'VPmax (mbar)' 这 5 个属性作为可以影响到温度的因素。即每一条数据有 5 个维度，我取若干个数据作为一组进行循环保存前后的影响，最后输出为 label 即是我们的温度。最后用 2009-2015 的数据作为训练集，剩余的作为测试集。在本次实验中，我使用所有的训练集为一组进行循环，测试集中 5 天

为一个循环预测后面两天的温度。



T 和 p(mbar) 的关系

3. 搭建网络结构

本次实验需要实现四个网络结构，还是很有任务量的。（捂脸）

RNN 网络。我把循环放进了 forward 中，让他循环每一组的个数，比如一条语句有多少词，就循环多少次。具体代码如下图：

```
class RNN(nn.Module):
    def __init__(self, args, output_size):
        super(RNN, self).__init__()
        self.device = args.device
        self.hidden_size = args.hidden_size
        self.input_size = args.input_size
        self.i2h = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.h2o = nn.Linear(self.hidden_size, output_size)
        self.tanh = nn.Tanh()

    def forward(self, x, hidden=None): # x是一个句子，tensor
        global output
        if not hidden:
            hidden = torch.zeros(1, self.hidden_size).to(self.device)
        x = x[0]
        for i in range(x.shape[0]):
            token = x[i: i + 1]
            combined = torch.cat((token, hidden), 1)
            hidden = self.tanh(self.i2h(combined))
            output = self.h2o(hidden)
        return output
```

RNN 网络结构

LSTM 网络。在 RNN 的基础上，为了改善其短期依赖问题，加入长期依赖，所以有了 LSTM。LSTM 在内部结构中引入了新的内部状态 c 作为候选状态。在 RNN 的 hidden 之后，将 hidden 经过变换输入给遗忘门，输入门和输出门，然后上一时态的 c，经过遗忘门就擦去一点记忆，和输入门重新连接作为新的一个时态的 c，最后这个 c 再和输出门相乘得到新一个时态的 hidden。最后 hidden 通过全连接层输出为 output。

LSTM-Bi 网络。所有加了 Bi 的网络就是在循环的时候，同时考虑向前循环和向后循环的 hidden，将两者连接之后再通过全连接层输出。这里我的构建方法每次循环，将向前的 hidden 和向后的 hidden 分别放入两个 list 中。然后将两个 hidden 取均值相加作为最后的 hidden，然后输出。具体代码如下：

```
else:
    num = x.shape[0]
    hidden1 = torch.zeros(1, self.hidden_size).to(self.device)
    hidden2 = torch.zeros(1, self.hidden_size).to(self.device)
    hidden1s = []
    hidden2s = []
    ct1 = torch.zeros(1, self.hidden_size).to(self.device)
    ct2 = torch.zeros(1, self.hidden_size).to(self.device)
    for i in range(num):
        token1 = x[i: i+1]
        token2 = x[num-i-1: num-i]
        combined1 = torch.cat((token1, hidden1), 1)
        combined2 = torch.cat((token2, hidden2), 1)
        forget1 = self.sigmoid(self.forget_gate(combined1))
        forget2 = self.sigmoid(self.forget_gate(combined2))
        input1 = self.sigmoid(self.input_gate(combined1))
        input2 = self.sigmoid(self.input_gate(combined2))
        c_1 = self.tanh(self.c_gate(combined1))
        c_2 = self.tanh(self.c_gate(combined2))
        output1 = self.sigmoid(self.output_gate(combined1))
        output2 = self.sigmoid(self.output_gate(combined2))
        ct1 = ct1 * forget1 + input1 * c_1
        ct2 = ct2 * forget2 + input2 * c_2
        hidden1 = self.tanh(ct1) * output1
        hidden2 = self.tanh(ct2) * output2
        hidden1s.append(hidden1)
        hidden2s.insert(0, hidden2)
    hidden1 = torch.stack(hidden1s).mean(0)
    hidden2 = torch.stack(hidden2s).mean(0)
    result = self.h2o(torch.cat((hidden1, hidden2), 1))
    return result
```

LSTM-Bi 网络中 forward 部分

GRU 网络。因为 LSTM 网络本来就是冗余的，有遗忘门和输出门，所以有了 GRU 网络的更新门直接控制有多少信息能够从上一个 hidden 到下一个 hidden。具体代码如下：

```
def forward(self, x):
    global output
    hidden = torch.zeros(1, self.hidden_size).to(self.device)
    ones = torch.ones(1, self.hidden_size).to(self.device)
    x = x[0]
    for i in range(x.shape[0]):
        token = x[i: i + 1]
        combined = torch.cat((token, hidden), 1) # 1 x (128+_ )
        reset = self.sigmoid(self.reset_gate(combined))
        zt = self.sigmoid(self.update_gate(combined))
        combined2 = torch.cat((token, reset * hidden), 1)
        h_ = self.tanh(self.h_gate(combined2))
        hidden = zt * hidden + (ones - zt) * h_
        output = self.h2o(hidden)
    return output
```

GRU 网络结构

4. 优化器和损失函数

本次实验选用的优化器可以使用 Adam 和 Sgd。

文本分类损失函数是交叉熵损失函数 `CrossEntropyLoss()`。温度预测的

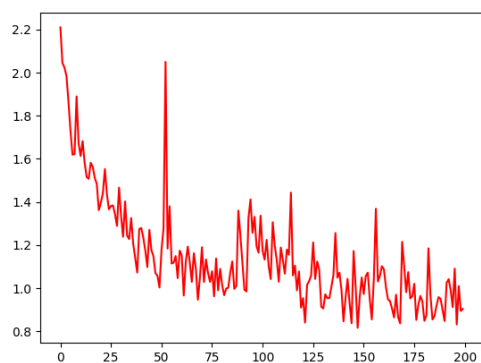
损失就是真实温度和预测温度的差值。

5. 训练过程、测试过程及实验结果

模型训练没什么好讲的，在训练过程中遇到准确率最高的模型进行保存，再用保存好的模型跑测试集，调用 sklearn 中的 `calssification_report` 得到准确率、召回率和 F1。

文本分类任务：

RNN 网络：

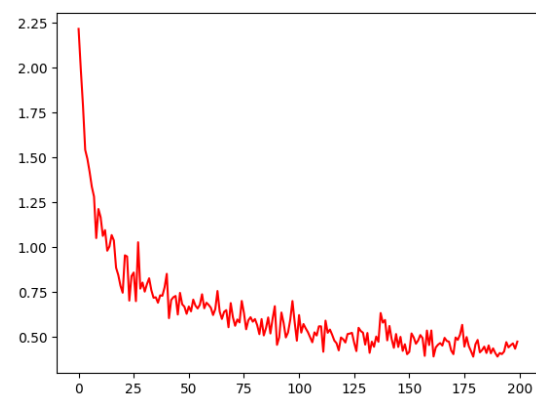


RNN 网络训练的 loss

	precision	recall	f1-score	support
书籍	0.27	0.75	0.39	2310
平板	0.31	0.27	0.29	6000
手机	0.43	0.02	0.03	1395
水果	0.41	0.21	0.28	6000
洗发水	0.49	0.14	0.22	6000
热水器	0.00	0.00	0.00	344
蒙牛	0.82	0.12	0.21	1219
衣服	0.25	0.43	0.31	6000
计算机	0.22	0.49	0.30	2396
酒店	0.60	0.52	0.56	6000
accuracy			0.33	37664
macro avg	0.38	0.29	0.26	37664
weighted avg	0.40	0.33	0.32	37664

曲线准确率、召回率、F1

LSTM:

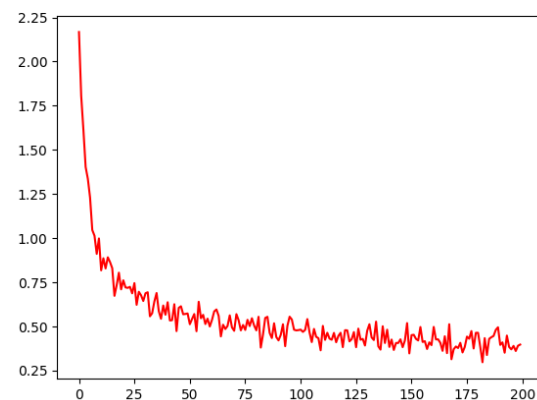


LSTM 网络训练的 loss

Testing accuracy, recall, F1 by LSTM...				
100%	37664/37664	[11:43:00:00, 53.52it/s]		
	precision	recall	f1-score	support
书籍	0.59	0.20	0.30	2310
平板	0.67	0.22	0.33	6000
手机	0.73	0.10	0.18	1395
水果	0.79	0.36	0.50	6000
洗发水	0.25	0.91	0.39	6000
热水器	0.25	0.00	0.01	344
蒙牛	0.99	0.12	0.21	1219
衣服	0.41	0.16	0.23	6000
计算机	0.55	0.60	0.57	2396
酒店	0.97	0.82	0.88	6000
accuracy			0.45	37664
macro avg	0.62	0.35	0.36	37664
weighted avg	0.62	0.45	0.44	37664

曲线准确率、召回率、F1

Bi-LSTM:

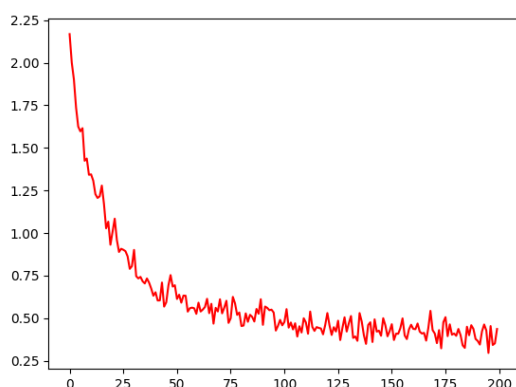


Bi-LSTM 网络训练的 loss

Testing accuracy, recall, F1 by Bi-LSTM...				
100%	37664/37664	[13:14:00:00, 47.39it/s]		
	precision	recall	f1-score	support
书籍	0.90	0.64	0.75	2310
平板	0.71	0.08	0.14	6000
手机	0.81	0.14	0.23	1395
水果	0.59	0.73	0.65	6000
洗发水	0.38	0.86	0.53	6000
热水器	0.38	0.04	0.08	344
蒙牛	0.86	0.72	0.78	1219
衣服	0.65	0.30	0.41	6000
计算机	0.38	0.25	0.30	2396
酒店	0.66	0.96	0.78	6000
accuracy			0.55	37664
macro avg	0.63	0.47	0.47	37664
weighted avg	0.62	0.55	0.50	37664

曲线准确率、召回率、F1

GRU:



GRU 网络训练的 loss

```
Testing accuracy, recall, F1 byGRU...
100%|██████████| 37664/37664 [05:48<00:00, 108.12it/s]
precision    recall  f1-score   support

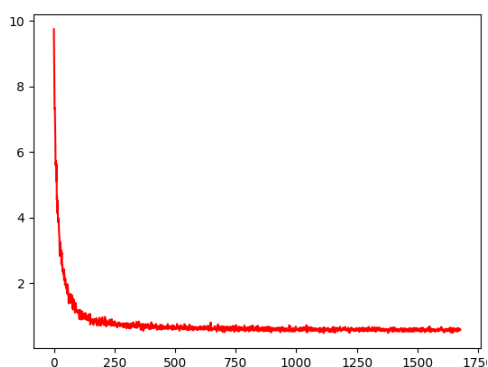
   书籍      0.46    0.81    0.59     2318
   平板      0.61    0.23    0.34     6000
   手机      0.40    0.20    0.26     1395
   水果      0.88    0.09    0.16     6000
  洗发水      0.47    0.49    0.48     6000
   热水器      0.09    0.03    0.05      344
   蒙牛      1.00    0.21    0.34     1219
   衣服      0.51    0.71    0.60     6000
  计算机      0.37    0.78    0.50     2396
   酒店      0.58    0.95    0.72     6000

 accuracy          0.51    37664
 macro avg       0.54    0.45    0.40    37664
 weighted avg    0.59    0.51    0.46    37664
```

曲线准确率、召回率、F1

温度预测任务:

温度预测任务使用的是 GRU，并且没有验证集，直接用训练集结束后的模型参数进行保存，然后跑测试集来预测温度。测试集中，使用 5 天的数据一组进行训练，然后预测后两天的数据（288 条）。下图截取了部分的预测结果，测算了预测值和真实值的均值误差和中位误差：

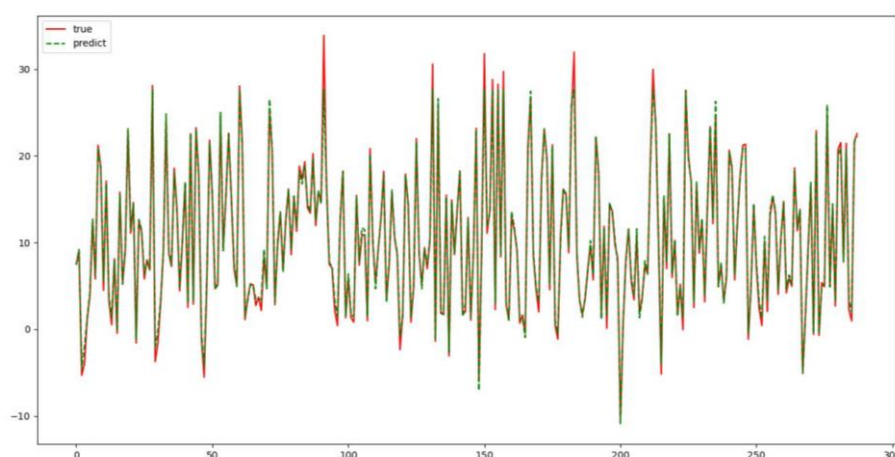


温度预测任务训练的 loss

```
Processing climate dataset...
Data processing finished!

Building model GRU...
Model building finished!
Testing temperature by GRU...
100%|██████████| 104820/104820 [01:14<00:00, 1403.99it/s]
[Week 0] Mean-loss: 0.486, Median-loss: 0.337
[Week 1] Mean-loss: 0.502, Median-loss: 0.369
[Week 2] Mean-loss: 0.491, Median-loss: 0.347
[Week 3] Mean-loss: 0.495, Median-loss: 0.349
[Week 4] Mean-loss: 0.531, Median-loss: 0.381
[Week 5] Mean-loss: 0.505, Median-loss: 0.368
[Week 6] Mean-loss: 0.500, Median-loss: 0.348
[Week 7] Mean-loss: 0.508, Median-loss: 0.329
[Week 8] Mean-loss: 0.471, Median-loss: 0.331
[Week 9] Mean-loss: 0.480, Median-loss: 0.383
```

部分周的均值误差和中位误差



部分预测结果和真实值

（更多结果图片可以参考 results 文件夹）感谢助教辛苦审阅报告！