中国科学院大学
University of Chinese Academy of Sciences

课程作业报告

基于 LSTM 的自动写诗程序

作者姓名：段宏键

学科专业：计算机系统结构

所在单位：中国科学院大学计算机科学与技术学院

2020 年 5 月

1. 摘要

使用 pytorch 框架，参考课程提供的《自动写诗实验指导书》完成了自动写诗的三层 LSTM 网络训练与测试，可以生成长度较长的诗歌。整体代码结构采用 train 与 test 分开的方法，并能在有 GPU 与无 GPU 的环境中自动适配，train.py 用于训练并保存训练模型的参数 Peotry-#.pkl， test.py 可以加载模型并根据“引子”来输出完整的诗歌。对与训练轮次（epoch）分别为 5，10，20 的模型进行了保存，通过测试发现训练轮次越高，效果越好（但由于 LSTM 训练时间过长，更多的训练轮次没有进一步尝试）。并且尝试分析了在不同训练轮次中最后的训练数据对诗歌风格的影响。

最后在训练数据集上训练 20 轮之后的交叉熵 loss 以及根据“引子”：“湖光秋月两相和”生成诗歌的效果如图：

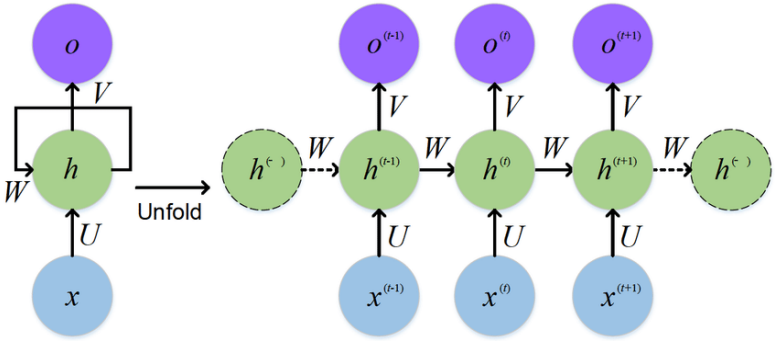
```
3501it [02:14, 21.99it/s]Epoch [20/20], train Loss is: 1.736394348008296
湖光秋月两相和，水面风来势似银。昔日曾为新诏去，今朝不见洛阳时。自从世上无人识，今日风光不可寻。一旦风流无别意，一枝一朵两三枝。不知何处最高月，一夜一声风雨来。不
```

1. 介绍

(1) 为了完成自动写诗，我们需要根据一个序列性的输入“引子”来给出序列性的输出。传统的前馈神经网络的两个局限，首先是难以捕捉到数据之间长距离的依赖关系，即使 CNN 也只能捕捉到局部或短距离的关系；其次，当训练样本输入是连续序列且长短不一时，不好处理，比如对一段连续的文本的处理。

同时,对文本的数据预处理也是一个问题，特别是古诗文本。

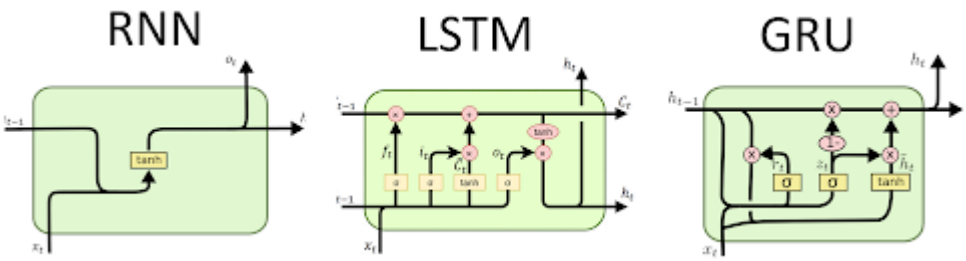
(2) 为了能够长距离信息的训练与传输，人们提出了下图循环神经网络 RNN 结构，右侧是左侧的图按照时间序列展开的结果。RNN 可以做到学习数据间长距离的关系。它可以模拟了人阅读一篇文章的顺序，从前到后阅读文章中的



每一个单词，将前面阅读到的有用信息编码到状态变量中去，从而拥有了一定的记忆能力，可以更好地处理之后的文本。

但 RNN 有严重的梯度消失问题。对于求导公式中间部分的累乘，当激活函数是 tanh 或 sigmoid 时，由于两者的导数绝对值小于 1，累乘后容易导致梯度消失，从而无法学习长距离的依赖关系（距离长的梯度消失了）。换用有较大的激活区域的 ReLU 函数能一定程度避免此问题。

使用 LSTM 可以一定程度上解决梯度消失的问题。LSTM，long short-term



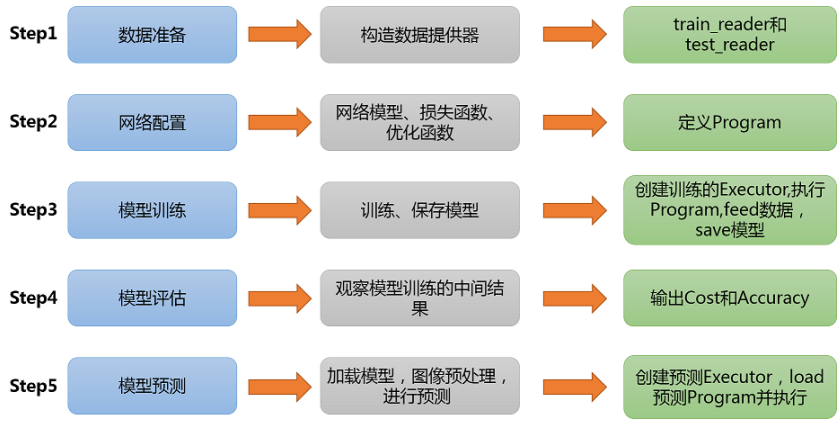
memory，长短期记忆，是 RNN 的一种改进。RNN 由于梯度消失，只能维持短期记忆，LSTM 通过引入记忆单元和门控制单元（将短期记忆与长期记忆结合），在一定程度上解决了梯度消失的问题。GRU 是 LSTM 的一个变体，将遗忘门和输入门合并为单一的更新门，同时混合了记忆单元和隐藏状态。

(3) 我采用 embedding 层+三层 LSTM 以及线性层来构造网络，使用 embedding 层后将汉字转化为 embedding 向量，与简单使用 One-hot 编码可以更好地表示汉字的语义，同时减少特征维度。向量化后使用 LSTM 网络来进行训练。 我用了三层单向的 LSTM 进行训练（也尝试过双向，但会出问题），训练程序（train.py）会保存训练模型，生成诗歌的程序(test.py)可以直接加载模型更具“引子”快速生成诗歌。

2. 解决方案

实验代码采用了模块化设计的方法，并且把模型训练与测试分开，这样能解耦代码结构，同时也能在把训练模型得到的模型数据保存，以便测试代码能在弱计算环境下顺利测试，而训练代码则可以使用云平台或其他平台。同时模型的训练与测试均兼容了纯 CPU 平台与 CPU+GPU（cuda）平台。

整个实验过程遵循以下几个步骤：



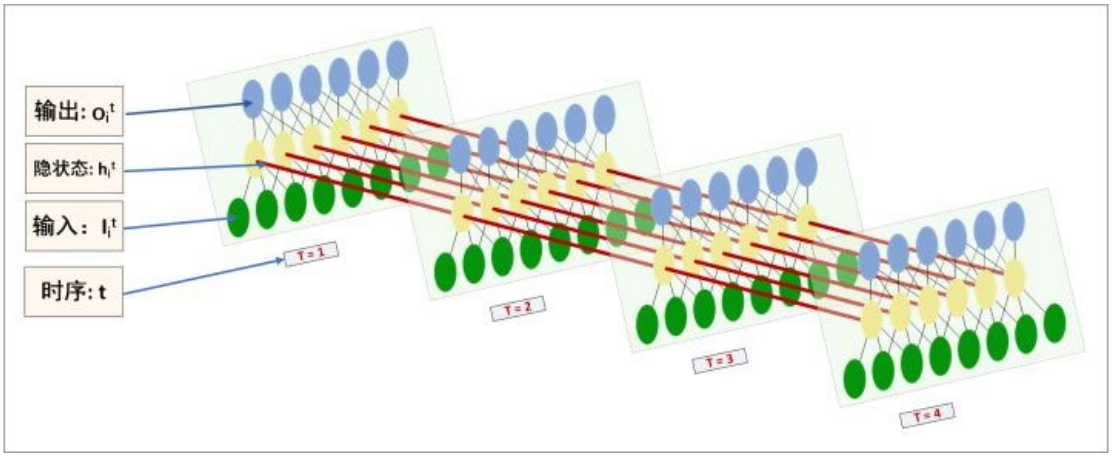
数据使用《实验三 自动写实实验指导书》教程中提到的 numpy 来进行加载，然后使用 pytorch 的 torch.from_numpy 以及 torch.utils.data.DataLoader 来进行转换。其中设置了数据 batch 大小为 16。

网络设计采用了 embeding 层+3 层 LSTM 层+一层线性层来完成，详细的网络结构设计如下：

```
PoetryModel(
  (embeddings): Embedding(8293, 128)
  (lstm): LSTM(128, 256, num_layers=3)
  (linear): Linear(in_features=256, out_features=8293, bias=True)
)
```

其中 LSTM 的 input_size=embedding_dim 为 128，hidden_dim 为 256。

对 LSTM 结构的理解可以参考下图：



训练优化方法与损失函数如下：

```
#4、 选择损失函数和优化方法
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

```
    criterion = nn.CrossEntropyLoss()
    loss_meter = meter.AverageValueMeter()
```

训练是在线上有 cuda 的环境中进行的。每次训练都会计算损失函数。并使用 pytorch 自动求导的机制更新参数。

设置了损失率为 1e-3， batch size 为 16。

训练完成后会保存训练好的模型的数据，用于线下（无 cuda 环境）的测试。

具体 train 的设计可以参见代码 train.py。

模型评估部分是在训练中有定时的可视化输出，会展示当前的 loss 以及生成诗歌的效果。

模型测试部分是在线下本机上用 test.py 来加载线上 cuda 环境里训练好的模型数据，然后来根据“引子”生成诗歌来测试的。自动写诗并没有非常严格准确的测评方法。

3. 实验结果和分析

实验配置如下：

深度学习平台：anaconda + pytorch（具体环境包依赖可以参见实验目录下的“环境依赖.txt”）

数据集：直接采用了课程资源中提供的 Tang.npz 数据集（这是经过预处理后的唐诗数据集，包含 57580 首唐诗（在课程网站下载）），其中 data 部分是唐诗数据的总共包含 57580 首唐诗数据，其中每一首都被格式化成 125 个字符，唐诗开始用'<START>'标志，结束用'<EOP>'标志,空余的用'<space>'标志，ix2word 和 word2ix 是汉字的字典索引。因此可以不用我们自己去构建这个字典。

训练环境：google colaboratory(训练使用的 runtime 是 GPU 类型)

测试环境：自己的笔记本电脑，无 GPU

训练过程：在训练过程中调用了 test.py 中生成诗歌的函数，会定期打印出训练的 loss 与生成诗歌的效果。

训练 20 轮后算法的交叉熵 loss 在 1.7-1.8 之间生成诗词的结果如下：

```
(pytorch) C:\dhj-learning\Git-repositories\DLCourse-2020\PoetryWriting>python test.py
引子：湖光秋月两相和
根据上面这句话生成诗.....
生成的诗句如下：
湖光秋月两相和，君不见山中水石。有时不见不得去，不见此心不如我。我今不识长安道，我欲相逢不相见。今年春色不胜春，春来不得春风生。今年春色不堪见，今日花枝还复明。青楼一望不可见，白日落花空自愁。不知今日君王事，莫惜春风不自愁。莫言陌上无人见，一别一日千里愁。
```

训练 10 轮之后的算法结果如下：

```
(pytorch) C:\dhj-learning\Git-repositories\DLCourse-2020\PoetryWriting>python test.py
引子：湖光秋月两相和
根据上面这句话生成诗.....
生成的诗句如下：
湖光秋月两相和，一片青山一片丝。一曲一声愁白发，一条飞尽落黄金。风吹玉瑟愁何极，风送金钗泪未开。一曲一杯愁白发，一年春色满青春
。风吹玉瑟愁何极，花落花前酒不开。一曲一杯同酩酊，一杯歌舞两三杯。云鬟未散还如此，风月何曾更不禁。莫怪不能留此曲，不知何处是无
情。人间不是无情事，此地无因得一身。莫道无心知不得，不知何处不堪伤。风吹白露侵秋月，风起寒灯满夕阳。不是故人同此别，不知何处是
归期。
```

训练 5 轮的结果：

```
47      model.load_state_dict(torch.load('Poetry-5.pkl'))
48  else:
49      model.load_state_dict(torch.load('Poetry-5.pkl',map_location='cpu'))       # 加载训练好的模型参数
50  print('引子：%s'%( words))
51  print("根据上面这句话生成诗.....")
52  gen_poetry = ''.join(generate(model, words, ix2word, word2ix))
53  # gen_poetry = gen_poetry.split(u"。")
54  print("生成的诗句如下：\n%s" % (gen_poetry))
55

PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL                    1: cmd

引子：湖光秋月两相和
根据上面这句话生成诗.....
生成的诗句如下：
湖光秋月两相和，一夜风流一望春。一夜一声飞一曲，一年一别一千年。一朝一别三千里，一别无人一别离。一别一杯无限事，一年春色不胜春
。一年一别无人识，一夜春风一片丝。一夜一声归去去，一年春色满江湖。一年一别无人识，一夜春风一片丝。一夜一声归去去，一年春色满江
湖。一年一别无人识，一夜春风一片丝。一夜一声归去去，一年春色满江湖。一年一别无人识，一夜春风一曲歌。
```

从上图可以明显看到，训练的轮次越少，生成的诗歌越简单（且诗句的重复性会越高）。

在训练 20 轮中用校名“中国科学院大学”来生成诗歌：

```
(pytorch) C:\dhj-learning\Git-repositories\DLCourse-2020\PoetryWriting>python test.py
引子：中国科学院大学
根据上面这句话生成诗.....
生成的诗句如下：
中国科学院大学，不如一钵何曾继。一朝不及三十人，不知何事无人识。一朝不得长自知，我欲一身无一事。一朝不见一杯酒，一日不得长生死
。君不见山中水，君不见西北水。今来不见不可闻，暮云飞入长安道。黄河水阔无定间，胡兵不战不可逃。胡兵不动不可遏，胡马回嘶胡塞隅。
```

可以看到，还是能生成可读诗歌的。

其次，在训练不同轮次结果展示的过程中，我感觉算法生成的诗歌风格会偏向最后输入训练模型的诗歌的风格。

4. 结论

这是继前面的 CNN 网络模型完成手写数字识别以及猫狗分类之后第一次使用 RNN 类（LSTM）网络来完成自动写诗歌，这个项目的代码仍然采用 train

与 test 分离的策略。网络设计采用了 embeding 层+3 层 LSTM 层+一层线性层来完成，通过尝试不同次数的训练轮次，发现较少的训练轮次生成的长诗中会有较多的重复，训练轮次增多之后（到 20）会有比较好的效果。同时，在训练轮次不多的情况下，生成诗歌的风格可能会偏向于最后一段时间训练输入的诗歌的风格。

5. 参考文献

参考了课程提供的《实验三 自动写实实验指导书》