DL-NLP 第五次作业

一、问题描述

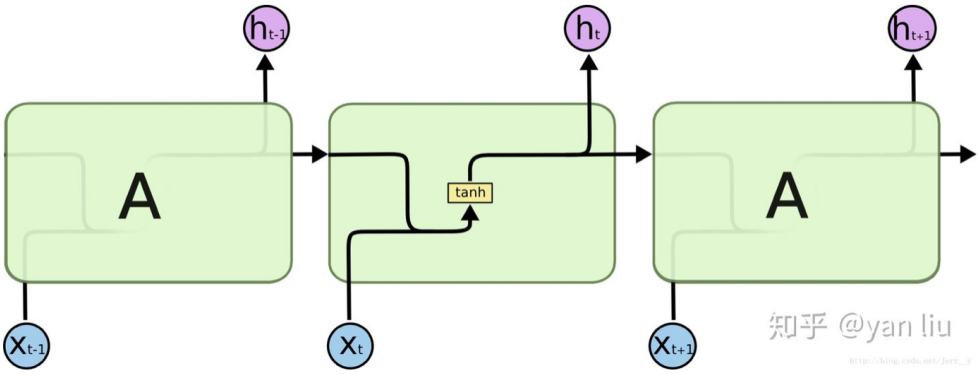
基于Seq2seq模型来实现文本生成的模型，输入可以为一段已知的金庸小说段落，来生成新的段落并做分析。

二、背景介绍

2.1 RNN

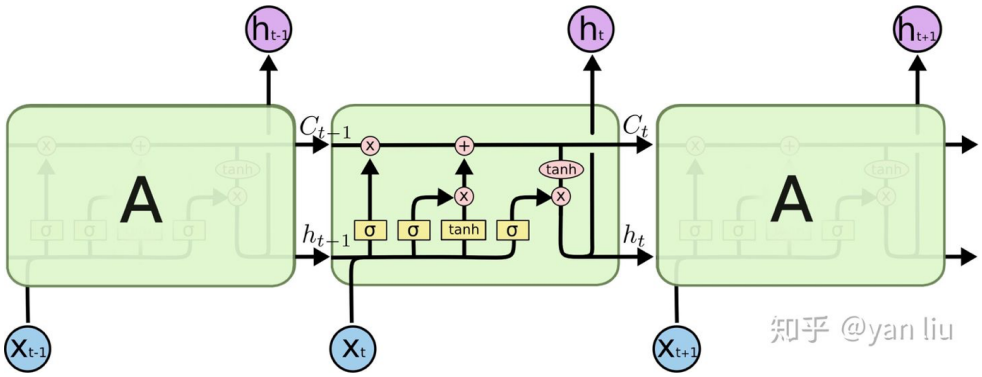
RNN（Recurrent Neural Network）是一类用于处理序列数据的神经网络。序列数据，摘取百度百科词条：时间序列数据是指在不同时间点上收集到的数据，这类数据反映了某一事物、现象等随时间的变化状态或程度。这是时间序列数据的定义，当然也可以不是时间，比如文字序列，但总归序列数据有一个特点：后面的数据跟前面的数据有关系。

RNN是神经网络的一种，类似的还有深度神经网络DNN，卷积神经网络CNN，生成对抗网络GAN，等等。RNN对具有序列特性的数据非常有效，它能挖掘数据中的时序信息以及语义信息，利用了RNN的这种能力，使深度学习模型在解决语音识别、语言模型、机器翻译以及时序分析等NLP领域的问题时有所突破。



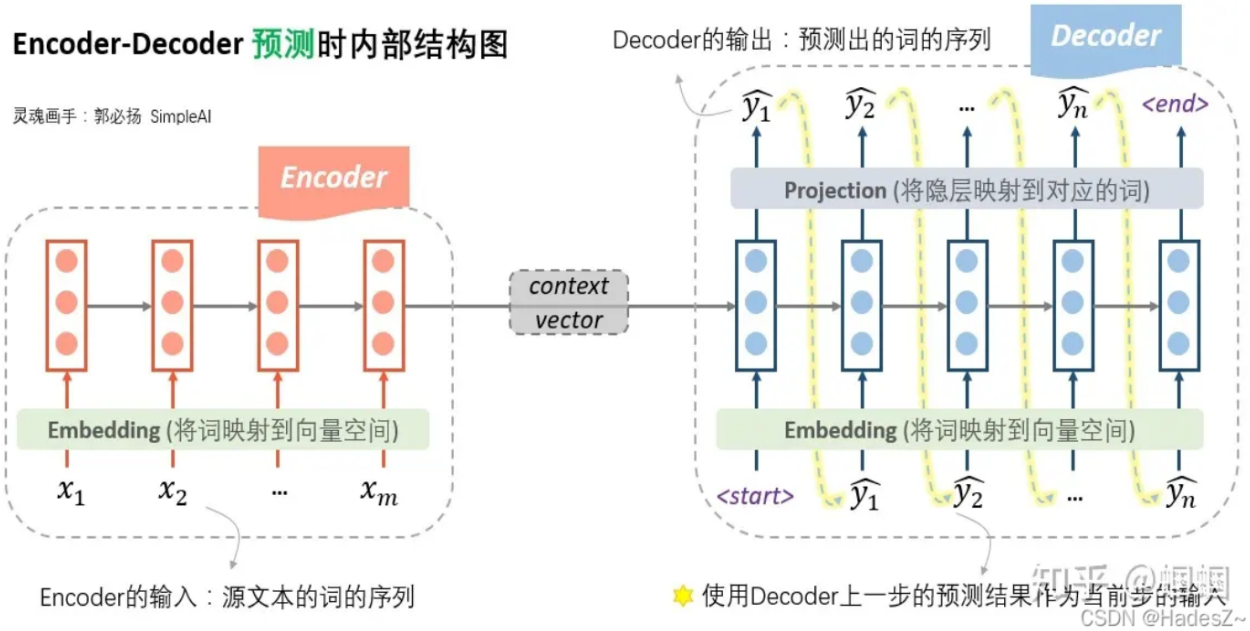
2.2 LSTM

在深度学习领域中（尤其是RNN），“长期依赖“问题是普遍存在的。长期依赖产生的原因是当神经网络的节点经过许多阶段的计算后，之前比较长的时间片的特征已经被覆盖。LSTM之所以能够解决RNN的长期依赖问题，是因为LSTM引入了门（gate）机制用于控制特征的流通和损失。LSTM的核心部分是在图中最上边类似于传送带的部分，这一部分叫做单元状态（cell state），自始至终存在于LSTM的整个链式系统中。门计算所需要用到的参数，由模型自己去进行学习。每一个门都有对应的参数，每一个门的每次计算，是根据当前的输入前一刻的状态，以及内部状态，来计算门的值是什么，最后再对整个状态进行更新。



2.2 seq2seq

sequence-to-sequence 模型最早是由 google 工程师在 2014 年 《Sequence to Sequence Learning with Neural Networks》论文中提出，模型在 NLP 领域中得到广泛使用。seq2seq 由 Encoder 和 Decoder 两个 RNN 组成. Encoder 将变长序列输出, 编码成 encoderstate 再由 Decoder 输出变长序列。



2.2.1 Encoder

Encoder 又称作编码器。它的作用是将现实问题转化为数学问题。seq2seq的编码器是单层或多层的RNN（双向），会对输入的文本进行编码变成一个向量输出。从技术上讲，编码器将可变长度的输入序列转换为固定形状的上下文变量c，并在此上下文变量中对输入序列信息进行编码。

2.2.2 Decoder

Decoder 又称作解码器。它的作用是求解数学问题，并转化为现实世界的解决方案。seq2seq的解码器，也是一个单层或多层的RNN（非双向），然后根据context信息对每一步进行解码，输出对应的文本。使用编码器最后一个时间步的隐藏状态来初始化解码器的隐藏状态，这要求 RNN 编码器和 RNN 解码器具有相同数量的层和隐藏单位。为了进一步合并编码的输入序列信息，上下文变量会在所有时间步长与解码器输入连接起来。

三、实验过程

3.1 数据介绍

- txt_data 文件夹下存放了：16本金庸武侠小说
- decode 文件夹下存放了 utf-8 编码的小说
- train.py：采用seq2seq的模型训练及预测分类主程序
- textgenrnn_train.py：使用现有开源封装好的包 textgenrnn 来进行模型训练和预测
- chinese_characters_3500.txt 所有汉字表，用于word和idx的转换

3.2 数据预处理

选取小说《天龙八部》的部分字符段落（至3117行前，376140个字符）进行训练。首先程序需要读取小说段落内容和汉字3500字符，目的是对小说段落进行索引化，便于提高下一步训练的效率。

```
class Dataset(torch.utils.data.Dataset):
    def __init__(self, args, ):
        self.args = args
        self.words = self.load_words()
        self.uniq_words = self.get_uniq_words()
        self.index_to_word = {index: word for index, word in
enumerate(self.uniq_words)}
        self.word_to_index = {word: index for index, word in
enumerate(self.uniq_words)}
        # 把小说的 字 转换成 int
        self.words_indexes = []
        # 把字典里没有的字符 用'*'表示，也就是Chinese_characters_3500.txt没有的字符
        for w in self.words:
            if not (w in self.word_to_index):
                self.words_indexes.append(1482)  # 1482 =='*'
            else:
                self.words_indexes.append(self.word_to_index[w])

    def load_words(self):
        """加载数据集"""
        with open(train_novel_path, encoding='UTF-8') as f:
            corpus_chars = f.read()
        print('length', len(corpus_chars))
        return corpus_chars

    def get_uniq_words(self):
        with open(char_key_dict_path, 'r', encoding='utf-8') as f:
            text = f.read()
        idx_to_char = list(text)  # 不能使用 set(self.words) 函数 ,因为每次启动随
机,只能用固定的
        return idx_to_char

    def __len__(self):
        return len(self.words_indexes) - self.args.sequence_length

    def __getitem__(self, index):
```

```
        return (
            torch.tensor(self.words_indexes[index:index +
self.args.sequence_length]).cpu()
            torch.tensor(self.words_indexes[index + 1:index +
self.args.sequence_length + 1]).cpu(),
        )
```

3.3 模型训练

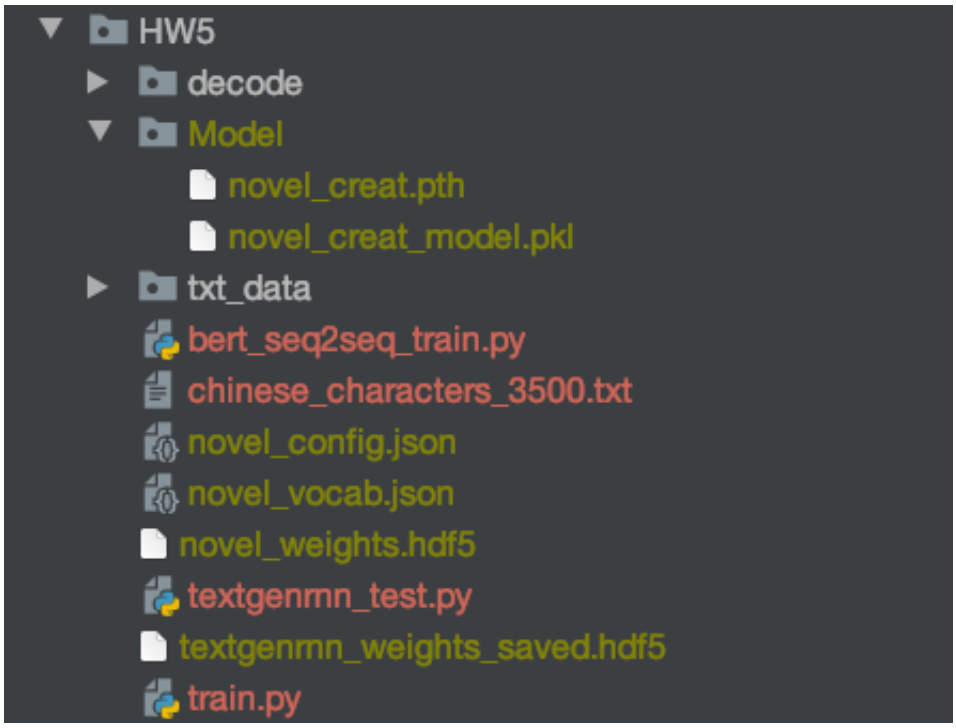
首先定义模型，规定RNN的输入大小为128，隐藏层大小为256，embedding大小与输入大小一致为128，模型一共两层，学习率为0.001。由于mac系统限制，无法使用gpu进行训练，采用电脑自身的cpu进行训练。

训练 epoch 次数取5；一个batch的大小为256；每次训练句子的字符数为50。在此参数下，使用cpu所训练时间大约为12小时。

```
class Model(nn.Module):
    def __init__(self, dataset):
        super(Model, self).__init__()
        self.input_size = 128
        self.hidden_size = 256
        self.embedding_dim = self.input_size
        self.num_layers = 2
        n_vocab = len(dataset.uniq_words)
        self.embedding = nn.Embedding(
            num_embeddings=n_vocab,
            embedding_dim=self.embedding_dim,
        )
        self.rnn = nn.RNN(
            input_size=self.input_size,
            hidden_size=self.hidden_size,
            num_layers=self.num_layers,
        )
        self.rnn.cpu()
        self.fc = nn.Linear(self.hidden_size, n_vocab).cpu()

    def forward(self, x, prev_state):
        embed = self.embedding(x).cpu()
        output, state = self.rnn(embed, prev_state)
        logits = self.fc(output)
        return logits, state

    def init_state(self, sequence_length):
        return torch.zeros(self.num_layers, sequence_length,
self.hidden_size).cpu()


def train(dataset, model, args):
    model.to(device)
```

```
    model.train()
    dataloader = DataLoader(
        dataset,
        batch_size=args.batch_size,
    )
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    for epoch in range(args.max_epochs):
        state = model.init_state(args.sequence_length)
        for batch, (x, y) in enumerate(dataloader):
            optimizer.zero_grad()
            x = x.cpu()
            y = y.cpu()
            y_pred, state = model(x, state)
            loss = criterion(y_pred.transpose(1, 2), y)
            loss = loss.to(device)
            state = state.detach()
            loss.backward()
            optimizer.step()
            if batch % 1000 == 0:
                torch.save(model, model_save_path)
                torch.save(model.state_dict(), model_save_path_pth)
            print({'epoch': epoch, 'batch': batch, 'loss': loss.item()})
```

为了减小上传的文件大小，将过大的模型文件采用 gitignore 来进行过滤，不上传至 github 上。其中 Model 文件夹中是采用 RNN 进行训练所得出的模型，其余模型文件是采用后文所提 textgenrnn 包所训练出的模型。模型大小约为120MB。



3.4 模型预测

```
def predict(dataset, model, text, next_words=20):
```

```
    words = list(text)
    model.eval()
    device = 'cpu'
    model.to(device)
    state = model.init_state(len(words))
    for i in range(0, next_words):
        x = torch.tensor([[dictGetValue(dataset.word_to_index, w) for w in
words[i:]]]).cpu()
        y_pred, state = model(x, state)
        last_word_logits = y_pred[0][-1]
        p = torch.nn.functional.softmax(last_word_logits,
dim=0).detach().numpy()
        word_index = np.random.choice(len(last_word_logits), p=p)
        words.append(dictGet(dataset.index_to_word, word_index))
    return "".join(words)
```

3.5 textgenrnn 包测试

textgenrnn 是一个基于 Keras/TensorFlow 的 Python 3 模块，用于创建 char-rnn，具有许多很酷炫的特性：它是一个使用注意力权重（attention-weighting）和跳跃嵌入（skip-embedding）等先进技术的现代神经网络架构，用于加速训练并提升模型质量；能够在字符层级和词层级上进行训练和预测；能够设置 RNN 的大小、层数，以及是否使用双向 RNN；能够对任何通用的输入文本文件进行训练；能够在 GPU 上训练模型，然后在 CPU 上使用这些模型；在 GPU 上训练时能够使用强大的 CuDNN 实现 RNN，这比标准的 LSTM 实现大大加速了训练时间；能够使用语境标签训练模型，能够更快地学习并在某些情况下产生更好的结果。

```
from textgenrnn import textgenrnn

if __name__ == '__main__':
    textgen = textgenrnn(name="novel")   # 给模型起个名字
    textgen.reset()
    # 从数据文件训练模型
    textgen.train_from_file(file_path='./decode/天龙八部demo', # 文件路径
                            new_model=True, # 训练新模型
                            batch_size=5,
                            rnn_bidirectional=True, # 是否使用Bi-LSTM
                            rnn_size=128,
                            word_level=False, # True:词级别，False:字级别
                            dim_embeddings=256,
                            num_epochs=3, # 训练轮数
                            max_length=30, # 一条数据的最大长度
                            verbose=1,
                            multi_gpu = False)

    textgen.save()

    textgen_2 = textgenrnn(weights_path='novel_weights.hdf5',
                            vocab_path='novel_vocab.json',
                            config_path='novel_config.json')
```

```
textgen_2.generate_samples()
```

四、实验结果

- 实验一（seq2seq，RNN）
  - 输入：包三先生笑道：“司马卫的儿子徒弟，都是这么一批脓包货色，除了暗箭伤人，什么都不会。”
  - 期待输出：诸保昆叫道：“看招！”一招“上天下地”，左手钢锥，右手小锤，同时向他攻去。
  - 实际输出：

```
    包三先生笑道：“司马卫的儿子徒弟，都是这么一批脓包货色，除了暗箭伤人，什么都不会。”
    众人都是一击，朗声说道：“乔帮主油倒，却未答治甚，争闹！我的情事，不枉难心，故人同时全身而来。
    徐长老马到几个剑法作呼，五分无口自当思索，抢先在地下出去，都该输。愚保定然一笑辣雅从座，更非复于哗地全声。
    徐长老既没练过，扳桨掌门‘赵钱孙的知离粪担，却也不怕惊讶，可不敢纵身内劲，竟正闻忽住他胸口中夹，太远不挥手拳向谭婆向他剐侧站过风波恶不得所田，自在人家都太太无端，可对自己的话，所是从众魂到底是宪能说，有信重事，不敢忘年旧白一窍，早知太来，倒不着之旁，谁喝在下首，一个小小娃貌，微微一笑，说并不许，比宪全冠清听赵钱孙道：“你不懂，你怎么还读儒之时，只不对向谭氏大汉子说话，便是他间剧给包呼家都不等帮明挽着。”
    孙此来的自人一眼，却与徐疑当在意到。中年来曾当从这事重重解故，是小娟大为，仪色的脸上大惑。
    司马林大怒道：“是情急，亦配处子。但料也不许禁手之后，这求人的是少林派位？”
    众人听到底是武功太快可从天远消有味不易。百兄不敢能是一袋法甲大笑。
    谭公一脸有愤怒动弹。这时便即不怕？谭公，名帖宪大谷来，这时日是他稳而人的又对，只是谭婆指点，说道：“我来好了一帮的，刚为此……
```

- 实验二（textgenrnn，LSTM）
  - 输入：忽得听杏林彼处，有一个苍老的声音说道：“能够挨打不还手，那便是天下第一等的功夫，岂是容易？”
  - 期待输出：众人回过头来，只见杏子树后转出一个身穿灰布衲袍的老僧，方面大耳，形貌威严。
  - 实际输出：

```
####################
Temperature: 1.0
####################
    这但他手握长越熟，知何况十分，四十年笑之事的。”
    乔峰对钟夫生女等，退到段誉和伏在桌上，心意在怀，他性情怀里和，发觉情异。人从丛中去接便，从背墙他愿丈外面收高手一异，抬起头来。
    那少女道：“抬起头来帮众，大理皇帝了夫人身为皇帝回万不了呵去。
####################
Temperature: 0.2
####################
```

```
    段誉道：“你不知道，我怎能能杀？”那少女道：“你不会知道么？”说着道：“你是徒具尸体，我徒儿怎能能杀了？”
    段誉道：“这位圣使恩恩师父，那也是不会。”那少女道：“你怎能能让够？”王语嫣道：“你是什么？”段誉道：“那么还能杀了？”
    段誉道：“那位公子，你是我徒儿，你怎能能杀杀了？”
####################
Temperature: 0.5
####################
    钟万仇一人，心想：“你走出来，这些人也不会儿，可怖可怖。”
    钟万仇怒视，便是又是。
    段誉道：“那老者是谁？”王语嫣道：“你们公子是。”王语嫣微笑道：“过身子，要紧什么？”段誉道：“这位兄弟，你不会来。”
```

- 从实验结果可以看出，无论是seq2seq（RNN）模型所输出的小说，还是 textgenrnn（LSTM）所输出的小说内容，文笔都有着古代文言文的风格，且文中所对应的人物均与训练小说《天龙八部》相关。
- 但训练的文本内容无论是从语法、逻辑还是情感上，都不够通顺，存在很多漏洞。而且引号和双引号等标点符号的使用也不规范（解决该问题或许需要制定相应的匹配规则，如出现左引号则必须有在适当位置出现右引号与其匹配）。因此总体来说预测结果较为一般。
- 由于受设备限制，训练效率十分低下，如果更改训练参数，加大数据量、神经网络层数、或者迭代次数，可能效果会更好。复杂的模型并不一定带来更好的效果，或许通过一些其他的数据采样方法、处理手段，辅助一些中文语言规则，可以提升文本生成的效果。