



北京航空航天大學

B E I H A N G U N I V E R S I T Y

深度学习与自然语言处理作业 5

院（系）名称 自动化科学与电气工程学院

学生姓名 潘翔

学生学号 ZY2103707

2022 年 6 月

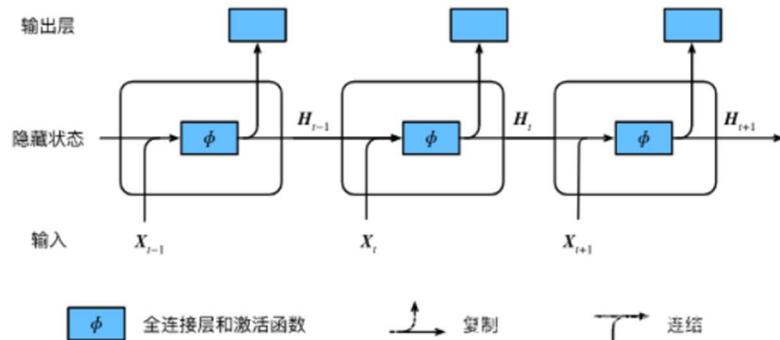
一、问题描述

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。截至日期：6月18日晚12点前。

二、试验原理

在 NLP 任务中，我们通常会遇到不定长的语言序列，比如机器翻译任务中，输入可能是一段不定长的英文文本，输出可能是不定长的中文或者法语序列。当遇到输入和输出都是不定长的序列时，可以使用编码器-解码器（encoder-decoder）模型或者 seq2seq 模型。其基本思想是编码器用来分析输入序列，解码器用来生成输出序列。

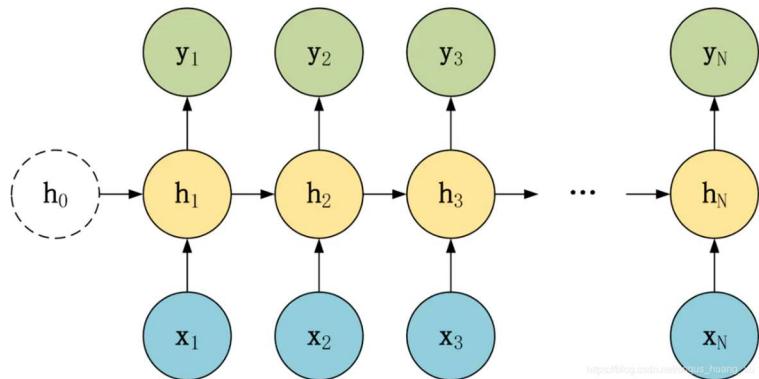
1.RNN 结构



RNN 中，每个单元接受两个输入，一个是当前时间步输入的信息 X_t ，另一个是上一个单元的隐藏层状态 H_{t-1} 。为什么这种结构的 RNN 适合用于做文本等序列型数据的任务，主要是因为隐藏状态的存在使得模型具有记忆性。针对不同的任务，根据输入和输出的数量，通常对 RNN 结构进行调整。

1) .N to N

该模型处理的一般是输入和输出序列长度相等的任务，如
词性标注
语言模型

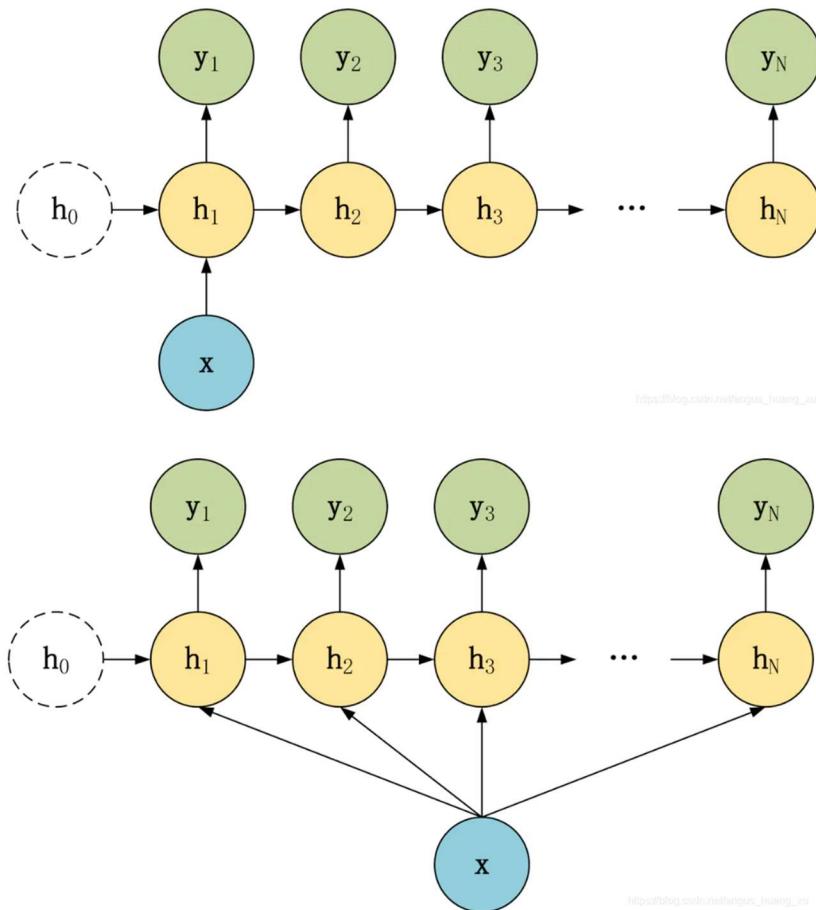


2) 1 to N

此类结构的输出长度为 1，输出长度为 N，一般又可以分为两种：一种是将输入只输出到第一个神经元，另一种将输入到所有神经元。

一般用于以下任务：

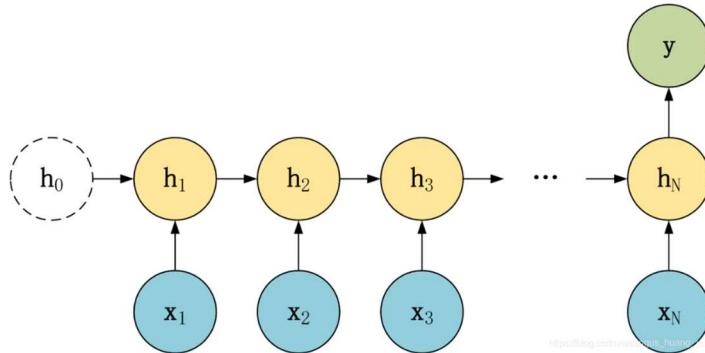
- ① 图像生成文字，一般输入 x 为图片，输出为一段图片描述性文字；
- ② 输入音乐类别，生成对应的音乐
- ③ 根据小说类别，生成对应的文字



3) N to 1

和 1 to N 相反，一般常见任务又：

序列分类任务，如给定一段文本或语音序列，归类（情感分类，主题分类）

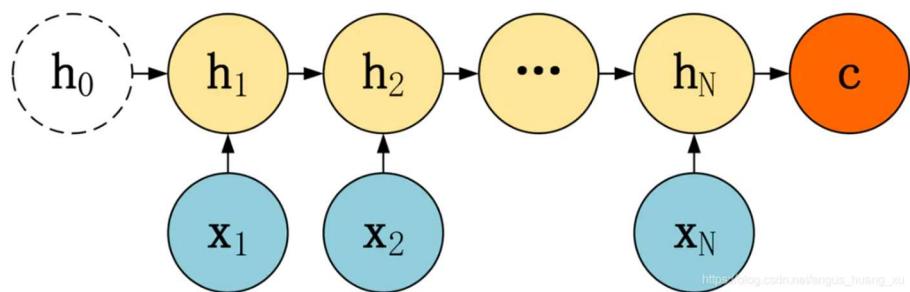


2. seq2seq 模型

经过上面对几种 RNN 结构的分析，不难发现 RNN 结构大多对序列的长度比较局限，对于类似于机器翻译的任务，输入和输出长度并不对等，为 N to M 的结构，简单的 RNN 束手无策，因此便有了新的模型，Encoder-Decoder 模型，也就是 Seq2Seq 模型。

模型一般由两部分组成：第一部分是 Encoder 部分，用于对输入的 N 长度的序列进行表征；第二部分是 Decoder 部分，用于将 Encoder 提取出的表征建立起到输出的 M 长度序列的映射。

1). 编码器 Decoder

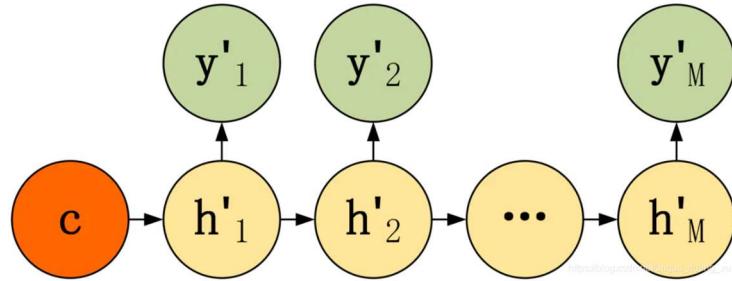


Encoder 部分一般使用了普通 RNN 的结构。其将一个序列表征为一个定长的上下文向量 c ，计算方式有多种，如下：

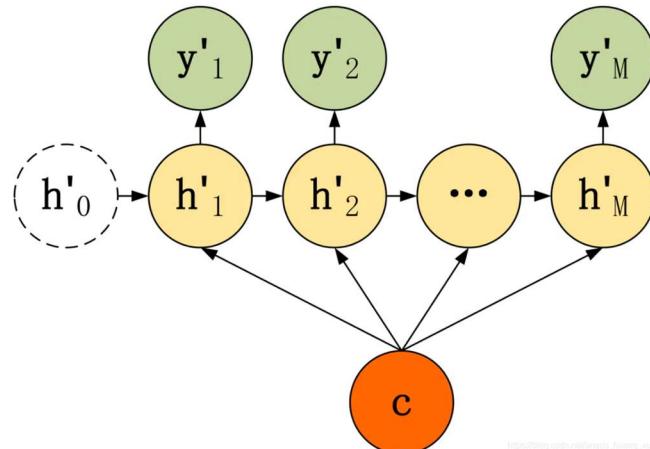
$$c = h_N, c = f(h_N), c = f(h_1, h_2, \dots, h_N)$$

2). 解码器 Decoder

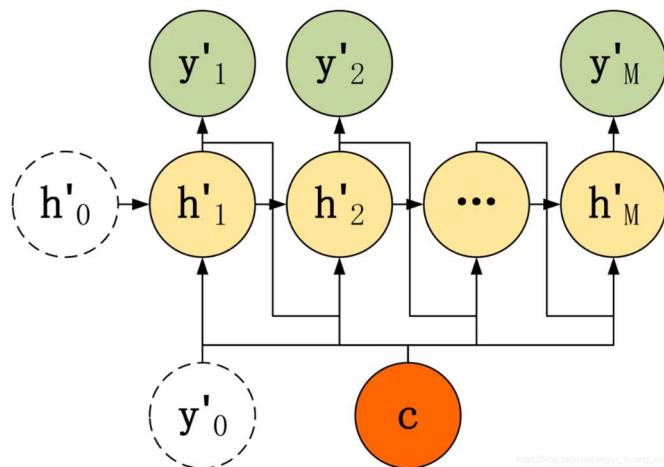
相对编码器而言，解码器的结构更多，下面介绍三种
第一种如下



第二种如下：



第三种如下：

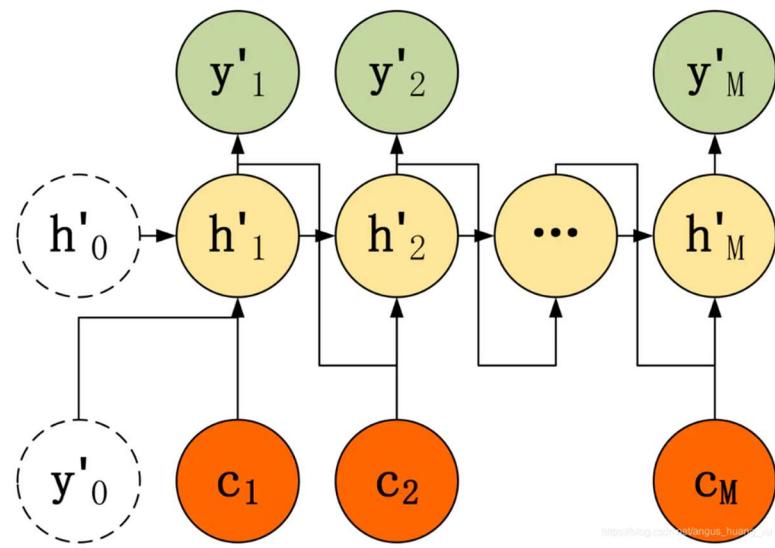


3. attention 机制

核心问题是当序列过长时，上述的 Decoder 输出的上下文向量 c 无法记住所有信息，会出现长序列梯度消失的问题。比如句子有 100 个词，那么 c 里面可能

丢失了前几个词的信息。

Attention 机制其实是参考了人在翻译文章时候的注意力机制，它会将模型的注意力放在当前翻译的单词上，换句话说，它并不像前面提到的 Encoder 的结构一样对整个句子用一个表征，它对于每个单词都有一个以单词为中心的表征。Decoder 结构如下



Encoder 结构还是基本的 RNN 结构，需要保存每个神经元的隐藏状态 h_t ，每个神经依旧保持隐藏状态。

三、试验过程和结果

由于以前没有使用过 torch 框架，所以在本次作业之前先学习了以下如何在电脑上使用 torch+CUDA 进行神经网络的训练。

配置 torch 和 CUDA 环境。

在 NV 的官网下载响应的 CUDA 版本，并且进行安装，在安装之后，选择对应的 torch 版本(注意是 GPU 版本，并且 torch 版本需要和 CUDA 版本对应)

在安装好了之后，在解释器中添加需要的相关库，并且在 pycharm 的终端进行测试。

分别依次输入

```
python  
import torch  
print(torch.__version__)  
print(torch.cuda.is_available())
```

得到结果如下

```
PS E:\DL_NLP_hw\hw6> python  
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print(torch.cuda.is_available())  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
AttributeError: module 'torch' has no attribute 'is_available'  
>>> print(torch.is_available)  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
AttributeError: module 'torch' has no attribute 'is_available'  
>>> print(torch.cuda.is_available())  
True  
>>>
```

最下方显示 True 说明配置成功，此时能够使用 torch 框架训练神经网络。

下面是实现代码

1. 读取训练语料

```

def read_data():
    with open('./datasets1/白马啸西风.txt', 'r', encoding='utf-8') as f:
        file_read = f.readlines()
        all_text = ""
        for line in file_read:
            line = re.sub(r'\s+', ' ', line)
            line = re.sub(r'[\u2000-\u3001]', ' ', line)
            line = re.sub(r'[\u3003-\u4DFF]', ' ', line)
            line = re.sub(r'[\u9FA6-\uFFFF]', ' ', line)
            all_text += line
        f.close()
    return all_text

```

2. 模型训练

利用 jieba 分词对语料库进行处理，并且用 LSTM 网络训练，如下所示

```

def train():
    embed_size = 1024
    epochs = 50
    end_num = 10

    print("读取数据开始")
    all_text = read_data()
    text_terms = list()
    for text_line in all_text.split('.'):
        seg_list = list(jieba.cut(text_line, cut_all=False)) # 使用精确模式
        if len(seg_list) < 5:
            continue
        seg_list.append("END")
        text_terms.append(seg_list)
    print("读取数据结束")
    # 获得word2vec模型
    print("开始计算向量")
    if not os.path.exists('model.model'):
        print("开始构建模型")
        model = Word2Vec(sentences=text_terms, sg=0, vector_size=embed_size, min_count=1, window=10, epochs=10)
        print("模型构建完成")
        model.save('model.model')
    print("模型已保存")
    print("开始训练")
    sequences = text_terms
    vec_model = Word2Vec.load('model.model')
    model = Net(embed_size).cuda()
    optimizer = torch.optim.AdamW(params=model.parameters(), lr=0.0001)
    for epoch_id in range(epochs):

```

```

for idx in range(0, len(sequences) // end_num - 1):
    seq = []
    for k in range(end_num):
        seq += sequences[idx + k]
    target = []
    for k in range(end_num):
        target += sequences[idx + end_num + k]
    input_seq = torch.zeros(len(seq), embed_size)
    for k in range(len(seq)):
        input_seq[k] = torch.tensor(vec_model.wv[seq[k]])
    target_seq = torch.zeros(len(target), embed_size)
    for k in range(len(target)):
        target_seq[k] = torch.tensor(vec_model.wv[target[k]])
    all_seq = torch.cat((input_seq, target_seq), dim=0)
    optimizer.zero_grad()
    out_res = model(all_seq[:-1].cuda())
    f1 = ((out_res[-target_seq.shape[0]:] ** 2).sum(dim=1)) ** 0.5
    f2 = ((target_seq.cuda() ** 2).sum(dim=1)) ** 0.5
    loss = (1 - (out_res[-target_seq.shape[0]:] * target_seq.cuda()).sum(dim=1) / f1 / f2).mean()
    loss.backward()
    optimizer.step()
    if idx % 50 == 0:
        print("loss: ", loss.item(), " in epoch ", epoch_id, " res: ", out_res[-target_seq.shape[0]:].max(dim=1).item())
state = {"models": model.state_dict()}
torch.save(state, "model/" + str(epoch_id) + ".pth")

```

3. 读取测试预料并进行测试输出

```

def read_test_data():
    name = 'test.txt'
    with open(name, "r", encoding='utf-8') as f:
        file_read = f.readlines()
        all_text = ""
        for line in file_read:
            line = re.sub('\s+', ' ', line)
            line = re.sub('。', '。', line)
            line = re.sub('？', '。', line)
            line = re.sub('，', '。', line) # 保留句号
            line = re.sub('[\u0000-\u3001]', '', line)
            line = re.sub('[\u3003-\u4dff]', '', line)
            line = re.sub('[\u9fa6-\uffff]', '', line)
            all_text += line
    return all_text

def test():
    embed_size = 1024
    print("start read test data")
    text = read_test_data()
    text_terms = list()
    for text_line in text.split('.'):
        seg_list = list(jieba.cut(text_line, cut_all=False)) # 使用精确模式
        if len(seg_list) < 5:
            continue
        seg_list.append("END")
        text_terms.append(seg_list)
    print("end read data")
    checkpoint = torch.load("model/" + str(49) + ".pth")

    model = Net(embed_size).eval().cuda()
    model.load_state_dict(checkpoint["models"])
    vec_model = Word2Vec.load('model.model')

    seqs = []
    for sequence in text_terms:
        seqs += sequence

```

```

input_seq = torch.zeros(len(seqs), embed_size).cuda()
result = ""
with torch.no_grad():
    for k in range(len(seqs)):
        input_seq[k] = torch.tensor(vec_model.wv[seqs[k]])
end_num = 0
length = 0
while end_num < 10 and length < 2000:
    print("length: ", length)
    out_res = model(input_seq.cuda())[-2:-1]
    key_value = vec_model.wv.most_similar(positive=np.array(out_res.cpu()), topn=20)
    key = key_value[np.random.randint(20)][0]
    if key == "END":
        result += "."
        end_num += 1
    else:
        result += key
    length += 1
    input_seq = torch.cat((input_seq, out_res), dim=0)
print(result)

```

使用金庸的小说《白马啸西风》作为训练集，并且从其中选出三段话作为测试集，结果如下

第一段输入文字为：

李文秀倚在一块岩石之旁，心里在想：「我爹爹妈妈万里迢迢的从中原来到回疆，为的是找高昌迷宫。他们没找到迷宫，就送了性命。其实就算找到了，多半也会给宫里的恶鬼害死，除非他们一听到恶鬼的声音立刻就退出。可是爹爹妈妈一身武功，一定不肯听恶鬼的话。唉，人的武功再高，又那里斗得过鬼怪？」忽然背后脚步声轻响，一人走了过来，低声叫道：「阿秀。」

输出结果为：

他也也了毒针武功。这他过汉子到她但只是也只是了咱们但汉子武功这一个。到啊他就上她一个毒针。只是啊武功只是了咱们也他便是只是啊咱们咱们过汉子的汉子便是了咱们。到过他但就只是她了他过过便是汉子啊一个他了的。只是过就她只是只是汉子咱们这他便是到毒针咱们一个了过也但过武功这她他毒针啊这上上武功也过一个汉子汉子上她过毒针到啊她但一个咱们她毒针了毒针咱们她就她汉子啊汉子就上毒针上上咱们的武功到便是但啊但武功汉子啊就。他毒针他了上毒针汉子只是到也一个她了上上到上也一个她。汉子到。

第二段输入文字为：

突然间心念一动，说道：「你姑娘，我来教你一路武功，你出去将这两个毛贼收拾了。」李文秀道：「要多久才能学会？没这麼快吧。」华辉沈吟道：「若是教你独指点穴、刀法拳法，只少也得半年才能奏功，眼前非速成不可，那只有练见功极快的的旁门兵刃，必须一两招间便能取胜。只是这山洞之中，那里去找什麼偏门的兵器？」一抬头间，突然喜道：「有了，去把那边的葫芦摘两个下来，要连著长藤，咱们来练流星锤。」

输出结果为：

她便是也上毒针她只是啊过便是咱们啊啊汉子就汉子啊。就毒针的到汉子就武功的只是她这一个毒针便是就武功毒针便是他就咱们武功这汉子上这了也。了便是咱们到武功的他咱们她这咱们到了过也了啊武功上一个一个便是就上他啊汉子一个但她的到也也一个。啊过一个过她毒针只是到便是她啊就的到便是但便是这咱们过武功了汉子武功。就只是上便是但这就但毒针过了了但这毒针毒针就这这他就他了到武功毒针他过到这这这只是这就过的到。到便是武功这就她到过。到他的了便是咱们她她也。的便是他汉子了一个也她便是。到便是到了过只是毒针上汉子他上这武功但咱们这也毒针她上咱们咱们武功只是咱们也武功过便是也汉子也只是汉子。

第三段输入文字为

吕梁三杰是结义兄弟。老大「神刀震关西」霍元龙，便是杀死白马李三的虬髯汉子。老二「梅花枪」史仲俊是个瘦瘦长长的汉子。好三「青麟剑」陈达海短小精悍，原是辽东马贼出身，后来却在山西落脚，和霍史二人意气相投，在山西省太谷县开设了晋威镖局。史仲俊和白马李三的妻子上官虹原是同门师兄妹，两人自幼一起学艺。史仲俊心中一直爱著这个娇小温柔的小师妹，师父也有意从中撮合，因此同门的师兄弟们早把他们当作是一对未婚夫妇。岂知上官虹无意中和白马李三相遇，竟尔一见锺情，家中不许他俩的婚事，上官虹便跟著他跑了。史仲俊伤心之馀，大病了一场，性情也从此变了。他对师妹始终馀情不断，也一直没娶亲。

输出结果为

过的她啊的就只是这武功就的他武功他这了了便是她他这他她汉子的一个一个武功便是了上啊到啊武功咱们啊上过到上他就只是啊咱们毒针咱们。她上啊一

个啊就了。一个这只是了这。他但的也便是的到啊。毒针的一个就她一个只是武功咱们也汉子就这便是就到啊便是啊咱们这武功的毒针这武功只是了也一个便是到过一个她到过。过一个就她了也啊咱们一个过了也咱们到咱们一个她过武功啊了但也的一个但咱们便是但咱们啊上了毒针了这到毒针便是汉子便是他的上这了就她只是便是上这咱们就这了武功但她的他就她过咱们她上上也只是的到一个但就过便是也上一个咱们啊也这就只是上的啊她只是就咱们他只是上过他她只是的上但毒针他的汉子。上她。一个咱们便是只是过了咱们他上他只是这但咱们她了武功过只是。一个的毒针这毒针了。

三段测试文本的输出可以看出，文本的输出内容和金庸的文笔有些相似，但是在逻辑，内容以及上下文的衔接方面和正常的小说文本还是有一定的差异，并且产生的段落语料内容相当有限，用通俗的话来说是在复读，说明该模型能够达到实验既定的要求，但是距离一段通顺的有真实意义的文本还有一段距离。

四、试验总结和体会

由于在学习本门课程之前，我没有真正使用过神经网络解决实际问题，加之这次作业是五次作业中最困难的一次，所以在完成本次实验的过程中遇到了很多困难。本次实验的代码是在学长代码的基础上实现的，基本能实现实验要求的功能，但是要达到我们心中对于“生成新的段落”的预期还有一定的距离，分析其原因，白马啸西风这部小说较为短小，可能是影响的原因之一。此外，训练神经网络需要的训练量较大，我使用 1050Ti 显卡进行神经网络的训练略显吃力，计算 50 个 poach 也消耗了一定时间，如果训练更复杂的网络则需要更高的算力。

经过这次试验，我学会了从基本的神经网络框架环境的配置，到使用 torch 框架训练神经网络解决一个实际问题，对以前不熟悉的神经网络也有了进一步的认识。这次实验是深度学习与自然语言处理的最后一次实验，这也意味着这门课程进入尾声，在这门课上我收获颇多，从完全不懂自然语言处理的概念和相关方法到学会用神经网络解决一个实际的自然语言处理问题，希望自己在之后的科研当中能够讲在这门课上所学到的知识应用起来！

五、参考资料

https://blog.csdn.net/shzx_55733/article/details/117338742?spm=1001.2014.3001.5502

https://blog.csdn.net/qq_42835351/article/details/119771935