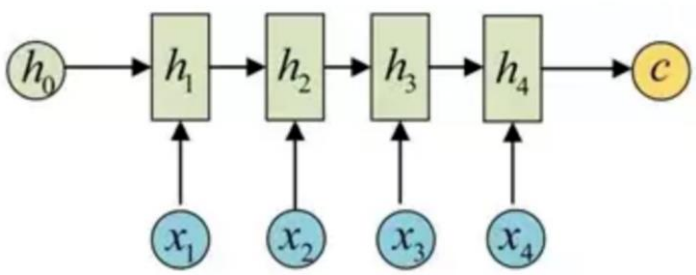
nlp-第五次大作业-seq2seq 小说生成
陈真-SY2103801

题目

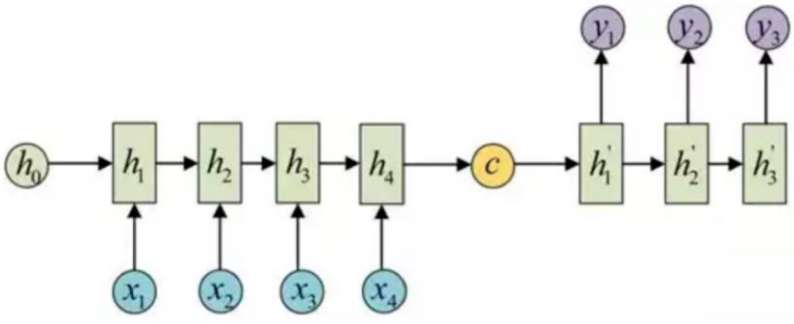
基于 seq2seq 模型来实现文本生成的模型，输入可以为一段已知的金庸小说段落，来生成新的段落并做分析。

序列到序列学习（seq2seq）模型

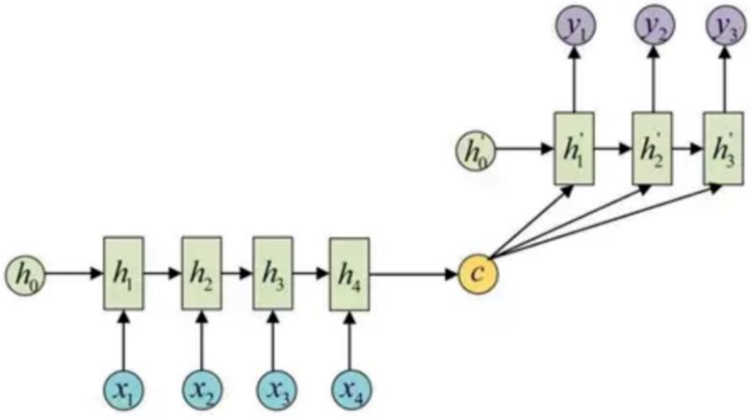
Seq2seq 模型是输出的长度不确定时采用的模型，这种情况一般是在机器翻译的任务中出现，将一句中文翻译成英文，那么这句英文的长度有可能会比中文短，也有可能会比中文长，所以输出的长度就不确定了。seq2seq 属于 encoder-decoder 结构的一种，这里看看常见的 encoder-decoder 结构，基本思想就是利用两个 RNN，一个 RNN 作为 encoder，另一个 RNN 作为 decoder。encoder 负责将输入序列压缩成指定长度的向量，这个向量就可以看成是这个序列的语义，这个过程称为编码。如下图，获取语义向量最简单的方式就是直接将最后一个输入的隐状态作为语义向量 C。也可以对最后一个隐含状态做一个变换得到语义向量，还可以将输入序列的所有隐含状态做一个变换得到语义变量。



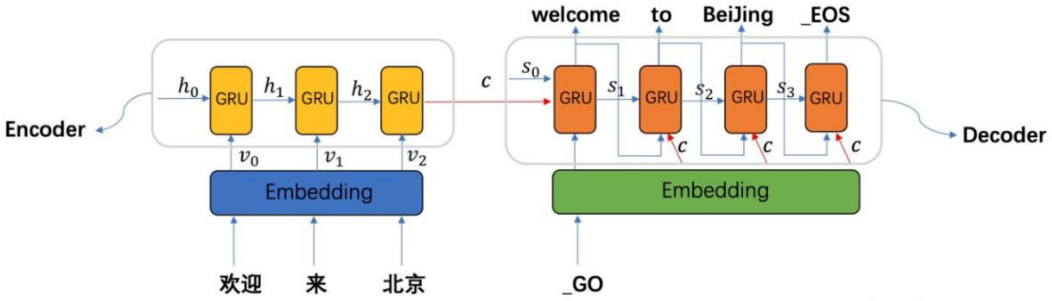
而 decoder 则负责根据语义向量生成指定的序列，这个过程也称为解码，如下图，最简单的方式是将 encoder 得到的语义变量作为初始状态输入到 decoder 的 RNN 中，得到输出序列。可以看到上一时刻的输出会作为当前时刻的输入，而且其中语义向量 C 只作为初始状态参与运算，后面的运算都与语义向量 C 无关。



decoder 处理方式还有另外一种，就是语义向量 C 参与了序列所有时刻的运算，如下图，上一时刻的输出仍然作为当前时刻的输入，但语义向量 C 会参与所有时刻的运算。



Seq2Seq 的数据流动



如上图所示，在 Encoder 中，“欢迎/来/北京”这些词转换成词向量，也就是 Embedding，我们用 v_i 来表示，与上一时刻的隐状态 h_{t-1} 按照时间顺序 $h_t = f(v_i, h_{t-1})$。假设有 t 个词，最终通过 Encoder 自定义函数 q 将各时刻的隐状态变换为向量 $c = q(h_0, h_1, ..., h_t)$，这个 c 就相当于从“欢迎/来/北京”这几个单词中提炼出来的大概意思一样，包含了这句话的含义。Decoder 的每一时刻的输入为 Eecoder 输出的 c 和 Decoder 前一时刻解码的输出 s_{t-1}，还有前一时刻预测的词的向量 E_{t-1}（如果是预测第一个词的话，此时输入的词向量为“_GO”的词向量，标志着解码的开始），我们可以用函数 g 表达解码器隐藏层变换：$s_i = g(c, s_{t-1}, E_{t-1})$。直到解码解出“_EOS”，标志着解码的结束。

代码实现及结果

代码实现部分采用的是 textgenrnn 库。textgenrnn 接受最多 40 个字符的输入，首先每个字符转换为 100 维的词(char)向量，并将这些向量输入到一个包含 128 个神经元的长短期记忆（LSTM）循环层中。其次，这些输出被传输至另一个包含 128 个神经元的 LSTM 中。以上所有三层都被输入到一个注意力层中，用来给最重要的时序特征赋权，并且将它们取平均（由于嵌入层和第一个 LSTM 层是通过跳跃连接与注意力层相连的，因此模型的更新可以更容易地向后传播并且防止梯度消失）。该输出被映射到最多 394 个不同字符的概率分布上，这些字符是序列中的下一个字符，包括大写字母、小写字母、标点符号和表情。而且关键是上述的参数都可以设置。

当使用 textgenrnn 在一个新的文本数据集上建立模型时，所有层都将被重新训练。然而，由于最初的预训练网络最初拥有更强大的“知识”，新的 textgenrnn 最终训练得更快、更准确，并且可以潜在地学习原始数据集中不存在的新关系（例如，预先训练的字符嵌入包含了所有可能类型的现代互联网语法的字符上下文）。

输入文本：

女童怒道：“我怎不知逍遥派？姥姥知道逍遥派之时，无崖子还没知道呢。”

虚竹道：“是，是”心想：“说不定你是个数百年前的老鬼，当然比无崖子老先生还老得多。”

只见那女童拾了一根枯枝，在地下积雪中画了起来，画的都是一条条的直线，不多时便画成一张纵横十九道的棋盘。

虚竹一惊：“她也要逼我下棋，那可糟了。”却见她画成棋盘后，便即在棋盘上布子，空心圆圈是白子，实心的一点的黑子，密密层层，将一个棋盘上都布满了。

只布到一半，虚竹便认了出来，正是他所解开的那个珍珑，心道：“原来你也知道这个珍珑。”又想：“莫非你当年也曾想去破解，苦思不得，因而气死么？”

想到这里，背上又感到一层寒意。

那女童布完珍珑，说道：“你说解开了这个珍珑，第一子如何下法，演给我瞧瞧。”

虚竹道：“是”当下第一子填塞一眼，将自己的白子胀死了一大片，局面登时开朗，然后依着段延庆当日传音所示，反击黑棋。

那女童额头汗水涔涔而下，喃喃道：“天意，天意天下又有谁想得到这先杀自身，再攻敌人的怪法？”

待虚竹将一局珍珑解完，那女童又沉思半晌，说道：“这样看来，小和尚倒也不是全然胡说八道。无崖子怎样将七宝指环传你，一切经过，你详细跟我说来，不许有半句隐瞒。”

虚竹道：“是”于是从头将师父如何派他下山，如何破解珍珑，无崖子如何传功传指环，丁春秋如何施毒暗杀苏星河和玄难，自己如何追寻慧方诸僧等情一一说了。

那女童一言不发，直等他说完，才道：“这么说，无崖子是你师父，你怎地不称师父，却叫什么无崖子老先生？”

虚竹神色尴尬，说道：“小僧是少林寺僧人，实在不能改投别派。”那女童道：“你是决意不愿做逍遥派掌门人的了？”

虚竹连连摇头，道：“万万不愿。”那女童道：“那也容易，你将七宝指环送了给我，也就是了。我代你做逍遥派掌门人如何？”

虚竹大喜，道：“那正是求之不得。”从指上除下宝石指环，交了给她。那女童脸上神色不定，似乎又喜又悲，接过指环，便往手上戴去。

可是她手指细小，中指与无名指戴上了都会掉下，勉强戴在大拇指上，端相半天，似乎很不满意，

问道：“你说无崖子有一幅图给你，叫你到大理无量山去寻人学那北冥神功，那幅图呢？”虚竹从怀中取了图画出来。

那女童打开卷轴，一见到图中的宫装美女，脸上倏然变色，骂道：“他他要这贱婢传你武功他他临死之时，仍是念念不忘这贱婢，将她画得这般好看”

霎时间满脸愤怒嫉妒，将图画往地下一丢，伸脚便踩。虚竹叫道：“啊哟”忙伸手抢起。

那女童怒道：“你可惜么？”虚竹道：“这样好好一幅图画，踩坏了自然可惜。”

那女童问道：“这贱婢是谁，无崖子这小贼有没跟你说？”虚竹摇头道：“没有。”

心想：“怎么无崖子老先生又变成了小贼？”

那女童怒道："哼，小贼痴心妄想，还道这贱婢过了几十年，仍是这等容貌啊，就算当年，她又哪有这般好看了？"

越说越气，伸手又要抢过画来撕烂。虚竹忙缩手将图画揣入怀中。

那女童身矮力微，抢不到手，气喘吁吁的不住大骂："没良心的小贼，不要脸的臭贱婢"

虚竹惘然不解，猜想这女童附身的老鬼定然认得图中美女，两人向来有仇，是以虽然不过见到一幅图画，却也怒气难消。

那女童还在恶毒咒骂，虚竹肚子突然咕咕咕的响了起来。他忙乱了大半天，再加上狂奔跳跃，粒米未曾进肚，已是十分饥饿。

那女童道："你饿了么？"虚竹道："是。这雪峰之上只怕没什么可吃的东西。"那女童道："怎么没有？雪峰上最多竹鸡，也有梅花鹿和羚羊。

我来教你一门平地快跑的轻功，再教你捉鸡擒羊之法"虚竹不等她说完，急忙摇手，说道："出家人怎可杀生？我宁可饿死，也不沾荤腥。"

那女童骂道："贼和尚，难道你这一生之中从未吃过荤腥？"

虚竹想起那日在小饭店中受一个女扮男装的小姑娘作弄，吃了一块肥肉，喝了大半碗鸡汤，

苦着脸道："小僧受人欺骗，吃过一次荤腥，但那是无心之失，想来佛祖也不见罪。但要我亲手杀生，那是万万不干的。"

输出文本：【下是在 temperature 参数为 1.0 的结果，较为多样性】

霎时间满脸愤怒嫉妒，将图画往地下一丢，伸脚便踩。虚竹叫道："这样好一幅图画，踩坏了自然可惜。"

你一惊："你说无崖子有是万不得。"从指与无名指指环，交了给我瞧。"又想："莫非你当年也曾想去破解，苦思不得，因而气死么？"

虚竹一惊："出家人怎可杀生？我下棋棋，那可糟。那女童道："莫非你当年也曾想去破解，苦思不也曾想想："万万不愿。"

那女童虚竹便认了出来，正是他所解开的那个珍珑，心道："原来你也知道这个珍珑。"又想："莫非你当年也曾想去破解，苦思不得，因而气死么？"

那女童打开了一根枯枝，在地下积雪中画的棋，那女童道："这贱婢，是全然胡说八道。无崖子有一幅图给你，叫你到大理无量山去寻人学那北冥神功，那幅图呢？"虚竹摇头道："这哟"忙伸手抢。

那女童骂道："哼，小贼痴心妄想，还道这贱婢传你武功他他临死之时开朗，然后依着脸上掉下山，如何破解完，急忙摇手，将她画得这般好看"

虚竹连连摇头，道："万不愿。"那女童道："这贱婢"

虚竹道："是，"当然变成小贼和尚，难道你这生之中从未过吃过荤腥？"

Temperature: 0.2
####################
虚竹道：“是”于是从头将师父如何派他下山，如何破解珍珑，无崖子如何传功传功传指环，丁春秋如何施毒暗杀苏星河和玄难，自己如何追寻慧方诸僧等情一一说了。
那女童怒道：“你饿了么？”虚竹道：“这样好一幅图画，踩坏了自然可惜。”
那女童怒道：“你饿了么？”虚竹道：“这样好好一幅图画，踩坏了自然可惜。”
虚竹道：“是”于是从头将师父如何派他下山，如何破解珍珑，无崖子如何传功传指环，丁春秋如何施毒暗杀苏星河和玄难，自己如何追寻慧方诸僧等情一一说了。
那女童怒道：“你饿了么？”虚竹道：“这样好好一幅图画，踩坏了自然可惜。”
那女童布完珍珑，说道：“你说解开了这个珍珑，第一子如何下法，演给我瞧瞧。”
那女童布完珍珑，说道：“你说解开了这个珍珑，第一子如何下法，演给我瞧瞧。”
那女童怒道：“你饿了么？”虚竹道：“这样好一幅图画，踩坏了自然可惜。”

Temperature: 0.5
####################
那女童还在恶毒咒骂，虚竹肚子突然咕咕咕的响了起来，画的都是一条条的直线，不多时便画成一张纵横十九道的棋盘。
那女童一言不发，直等他说完，才道：“这么说，无崖子是你师父，你怎地不称师父，却叫什么无崖子老先生？”
问道：“你说解开了这个珍珑，第一子如何下法，演给我瞧瞧。”
那女童怒道：“你饿了么？”虚竹道：“这贱婢是谁，无崖子这小贼有没跟你说？”虚竹摇头道：“没有。”
可是她手指细小，中指与无名指戴上了都会掉下，勉强戴在大拇指上，端相半天，似乎很不满意，
虚竹连连摇头，道：“万万不愿。”那女童道：“你说无崖子有一幅图给你，叫你到大理无量山去寻人学那北冥神功，那幅图呢？”虚竹从怀中取了图画出来。
虚竹大喜，道：“万不愿。”那女童道：“你饿了么？”虚竹道：“这样好一幅图画，踩坏了自然可惜。”
那女童额头汗水涔涔而下，喃喃道：“天意，天意天下又有谁想得到这先杀自身，再攻敌人的怪法？”

代码：
【预处理】

```
def  init reader(self):
    self.data = []
    input f = open(self.input file, 'rb')
    target f = open(self.target file, 'rb')
    for input line in input f:
        input line = input line.decode('utf-8')[:-1]
        target line = target f.readline().decode('utf-8')[:-1]
        input words = [x for x in input line.split(' ') if x != '']
        if len(input words) >= self.max len:
            input words = input words[:self.max len - 1]
        input words.append(self.end token)
        target words = [x for x in target line.split(' ') if x != '']
    input f.close()
    target f.close()
    self.data_pos = len(self.data)
```

【模型】

```
import math
import torch
import random
from torch import nn
from torch.autograd import Variable
import torch.nn.functional as F


class Encoder(nn.Module):
    def   init  (self, input size, embed size, hidden size,
                 n layers=1, dropout=0.5):
        super(Encoder, self).  init  ()
        self.input size = input size
        self.hidden size = hidden size
        self.embed size = embed size
        self.embed = nn.Embedding(input size, embed size)
        self.gru = nn.GRU(embed size, hidden size, n layers,
                          dropout=dropout, bidirectional=True)

    def forward(self, src, hidden=None):
        embedded = self.embed(src)
        outputs, hidden = self.gru(embedded, hidden)
        # sum bidirectional outputs
        outputs = (outputs[:, :, :self.hidden size] +
                   outputs[:, :, self.hidden size:])
        return outputs, hidden


class Attention(nn.Module):
    def   init  (self, hidden size):
        super(Attention, self).  init  ()
        self.hidden size = hidden size
        self.attn = nn.Linear(self.hidden size * 2, hidden size)
        self.v = nn.Parameter(torch.rand(hidden size))
        stdv = 1. / math.sqrt(self.v.size(0))
        self.v.data.uniform (-stdv, stdv)

    def forward(self, hidden, encoder outputs):
        timestep = encoder outputs.size(0)
        h = hidden.repeat(timestep, 1, 1).transpose(0, 1)
        encoder outputs = encoder outputs.transpose(0, 1)  # [B*T*H]
        attn energies = self.score(h, encoder outputs)
        return F.softmax(attn energies, dim=1).unsqueeze(1)

    def score(self, hidden, encoder outputs):
        # [B*T*2H]->[B*T*H]
        energy = F.relu(self.attn(torch.cat([hidden, encoder outputs],
2)))
        energy = energy.transpose(1, 2)  # [B*H*T]
        v = self.v.repeat(encoder outputs.size(0), 1).unsqueeze(1)  #
[B*1*H]
        energy = torch.bmm(v, energy)  # [B*1*T]
        return energy.squeeze(1)  # [B*T]
```

```
class Decoder(nn.Module):
    def __init__(self, embed_size, hidden_size, output_size,
                 n_layers=1, dropout=0.2):
        super(Decoder, self).__init__()
        self.embed_size = embed_size
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.n_layers = n_layers

        self.embed = nn.Embedding(output_size, embed_size)
        self.dropout = nn.Dropout(dropout, inplace=True)
        self.attention = Attention(hidden_size)
        self.gru = nn.GRU(hidden_size + embed_size, hidden_size,
                          n_layers, dropout=dropout)
        self.out = nn.Linear(hidden_size * 2, output_size)

    def forward(self, input, last_hidden, encoder_outputs):
        # Get the embedding of the current input word (last output
word)
        embedded = self.embed(input).unsqueeze(0)  # (1,B,N)
        embedded = self.dropout(embedded)
        # Calculate attention weights and apply to encoder outputs
        attn_weights = self.attention(last_hidden[-1],
encoder_outputs)
        context = attn_weights.bmm(encoder_outputs.transpose(0, 1))  #
(B,1,N)
        context = context.transpose(0, 1)  # (1,B,N)
        # Combine embedded input word and attended context, run
through RNN
        rnn_input = torch.cat([embedded, context], 2)
        output, hidden = self.gru(rnn_input, last_hidden)
        output = output.squeeze(0)  # (1,B,N) -> (B,N)
        context = context.squeeze(0)
        output = self.out(torch.cat([output, context], 1))
        output = F.log_softmax(output, dim=1)
        return output, hidden, attn_weights


class Seq2Seq(nn.Module):
    def __init__(self, encoder, decoder):
        super(Seq2Seq, self).__init__()
        self.encoder = encoder
        self.decoder = decoder

    def forward(self, src, trg, teacher_forcing_ratio=0.5):
        batch_size = src.size(1)
        max_len = trg.size(0)
        vocab_size = self.decoder.output_size
        outputs = Variable(torch.zeros(max_len, batch_size,
vocab_size)).cuda()

        encoder_output, hidden = self.encoder(src)
        hidden = hidden[:self.decoder.n_layers]
        output = Variable(trg.data[0, :])  # sos
        for t in range(1, max_len):
            output, hidden, attn_weights = self.decoder(
                    output, hidden, encoder_output)
            outputs[t] = output
```

```
            is_teacher = random.random() < teacher_forcing_ratio
            top1 = output.data.max(1)[1]
            output = Variable(trg.data[t] if is_teacher else
top1).cuda()
        return outputs
```

【训练】

```
from textgenrnn import textgenrnn
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='text1.txt', # 文件路径
                    new_model=True, # 训练新模型
                    batch_size=4,
                    rnn_bidirectional=True, # 是否使用Bi-LSTM
                    rnn_size=64,
                    word_level=False, # True:词级别，False:字级别
                    dim_embeddings=400,
                    num_epochs=8, # 训练轮数
                    max_length=35, # 一条数据的最大长度
                    verbose=1)

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

textgen_2.generate_samples(8)
```