深度学习与自然语言处理第四次作业

文本生成

惠仪

SY2202328@buaa.edu.cn

摘要

基于 LSTM 来实现文本生成模型，输入一段已知的金庸小说段落来生成新的段落并做定量与定性的分析。

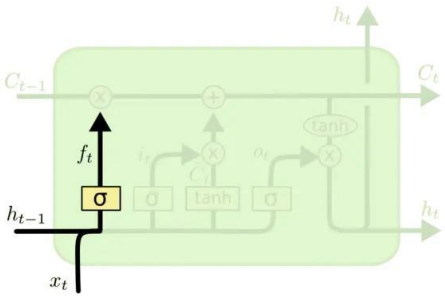
一、理论介绍

1、LSTM（Long Short Term Memory ）长短期记忆神经网络

在 RNN 的基础上加入了遗忘机制，选择性的保留或遗忘前期的某些数据，且不再采用乘法而是加法以避免梯度爆炸的问题。

①一个神经元的两个输入数据为：前一个神经元的输出（ht-1）和新的输入（xt）。

②遗忘

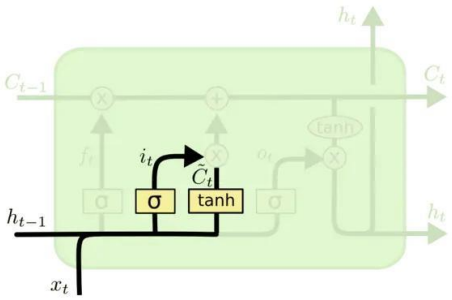


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

知乎 @热干

来了新东西后，根据新东西决定一下旧的东西要不要忘掉。两个输入数据，经过决定门，得到一个 0~1 之间的数 ft，乘到原始记忆库 Ct-1 中，将 Ct-1 中的 ft 部分保留，其余的遗忘。即原始记忆库中只有 ft 这一部分保留，其余的遗忘；而 ft 是由两个输入数据决定的。

③输入

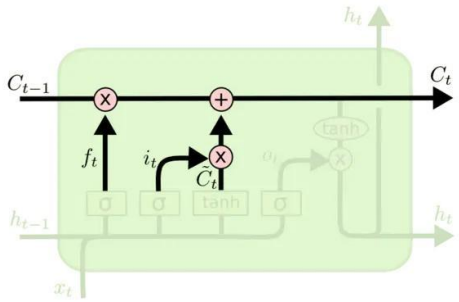


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

知乎 @热干

新的东西有多少要记住。两个输入数据，经过 tanh 门之后得到一个候选输入 Ct~；而候选输入中不是所有东西都要加入到记忆库中，因此利用两个输入数据，经过决定门得到 it，即候选输入中需要保留的比重。将被筛选后的候选输入 it*Ct~加入到原始记忆库中。

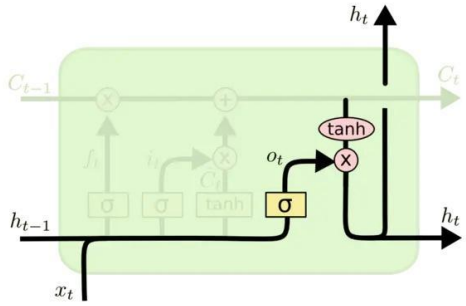


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

知乎 @热干

此时记忆库，从 Ct-1 经过：一次选择性遗忘 ft 的相乘，和一次选择性记忆的相加，成为了新的记忆库 Ct。

④输出



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

知乎 @热干

记住的东西有多少要拿出来说。新的记忆库 Ct 已经成型，经过 tanh 门后，再经由决定门 Ot 决定哪些要作为输出。决定门 Ot 也是从两个输入值计算而来

2、LSTM 文本生成

用深度学习生成序列数据的通用方法，就是使用前面的标记作为输入，训练一个网络（通常是循环神经网络或卷积神经网络）来预测序列中接下来的一个或多个标记。例如，给定输入 the cat is on the ma，训练网络来预测目标 t，即下一个字符。与前面处理文本数据时一样，标记（token）通常是词或字符，给定前面的标记，能够对下一个标记的概率进行建模的任何网络都叫作语言模型（language model）。语言模型能够捕捉到语言的潜在空间（latent space），即语言的统计结构。训练好一个语言模型，输入初始文本字符串（称为「条件数据」，conditioning data），从语言模型中采样，就可以生成新 Token，把新的 Token 加入条件数据中，再次输入，重复这个过程就可以生成出任意长度的序列。

采用 LSTM 层，输入文本语料的 N 个字符组成的字符串，训练模型来生成第 N+1 个字符。模型的输出是做 softmax，在所有可能的字符上，得到下一个字符的概率分布。这个模型叫作「字符级的神经语言模型」(character-level neural language model)

3、文本评价

文本评价关注文本的正确性、流畅度和易理解性。常见的内在评价方法又可分为两类:1）通过人工评价，从有用性等对文本进行打分。2）采用 BLEU、NIST

和 ROUGE 等进行自动化评价，评估生成文本和参考文本间相似度来衡量生成质量。

二、实验步骤

1、导入需要的包

```
import torch
import torch.nn as nn
from torch.nn.utils import clip_grad_norm_
import jieba
from tqdm import tqdm
import numpy as np
```

2、创建词典表

```
class Dictionary(object):

    def __init__(self):
        self.word2idx = {}
        self.idx2word = {}
        self.idx = 0

    def __len__(self):
        return len(self.word2idx)

    def add_word(self, word):
        if not word in self.word2idx:
            self.word2idx[word] = self.idx
            self.idx2word[self.idx] = word
            self.idx += 1
```

当每一句话经过分词处理后，如果该词未在列表中，将词按顺序添加到词典表中，最后得到所有词及其对应的序号。

3、语料集文本序列化

```
# step 1
with open(path, 'r', encoding="utf-8") as f:
    tokens = 0
    for line in f.readlines():
        words = jieba.lcut(line) + ['<eos>']
        tokens += len(words)
        for word in words:
            self.dictionary.add_word(word)
```

第一步读取 txt 文本中的每一句话，采用 jieba 分词将每一行中的句子分成词

语，添加到词典中。用 tokens 来记录分词后的句子总共有多少个词。

```
# step 2
ids = torch.LongTensor(tokens)
token = 0
with open(path, 'r', encoding="utf-8") as f:
    for line in f.readlines():
        words = jieba.lcut(line) + ['<eos>']
        for word in words:
            ids[token] = self.dictionary.word2idx[word]
            token += 1
```

第二步将把所有句子文本内容用词典表中存储的对应序号表示，即文本的序列化表示。

```
# step 3
num_batches = ids.size(0) // batch_size
ids = ids[:num_batches * batch_size]
ids = ids.view(batch_size, -1)
return ids
```

第三步划分每次训练时的数量。

4、LSTM 模型

```
class LSTMmodel(nn.Module):

    def __init__(self, vocab_size, embed_size, hidden_size, num_layers):
        super(LSTMmodel, self).__init__()
        self.embed = nn.Embedding(vocab_size, embed_size)
        self.lstm = nn.LSTM(embed_size, hidden_size, num_layers, batch_first=True)
        self.linear = nn.Linear(hidden_size, vocab_size)
        self.dropout = nn.Dropout(0.05)
    def forward(self, x, h):
        x = self.embed(x)
        out, (h, c) = self.lstm(x, h)
        out = out.reshape(out.size(0) * out.size(1), out.size(2))
        out = self.dropout(out)
        out = self.linear(out)
        return out, (h, c)
```

nn.lstm(input_size,hidden_size,num_layers,bias,batch_first,dropout,bidirectional)

input_size：表示的是输入的矩阵特征数，或者说是输入的维度；

hidden_size：隐藏层的大小（即隐藏层节点数量），输出向量的维度等于隐藏节点数；

num_layers：lstm 隐层的层数，默认为 1；

输入到模型后，得到输出后进行 dropout，然后再到线性层，得到输出结果

5、训练函数

```
corpus = Corpus()
ids = corpus.get_data('sgyy.txt', batch_size)
vocab_size = len(corpus.dictionary)
model = LSTMmodel(vocab_size, embed_size, hidden_size, num_layers).to(device)
cost = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
for epoch in range(num_epochs):

    states = (torch.zeros(num_layers, batch_size, hidden_size).to(device),
              torch.zeros(num_layers, batch_size, hidden_size).to(device))
    loss_list=[]
    for i in tqdm(range(0, ids.size(1) - seq_length, seq_length)):
        inputs = ids[:, i:i+seq_length].to(device)
        targets = ids[:, (i+1):(i+1)+seq_length].to(device)
        states = [state.detach() for state in states]
        outputs, states = model(inputs, states)
        loss = cost(outputs, targets.reshape(-1))
        model.zero_grad()
        loss.backward()
        clip_grad_norm_(model.parameters(), 0.5)
        optimizer.step()
        loss_list.append(loss.item())
    print("epocth={}\tloss={}".format(epoch, np.mean(loss_list)))
```

首先读取数据，设置模型 LSTM、优化器 Adam 和交叉熵损失函数。在训练过程中，每次选取 30 个输入模型后，输出结果与实际段落中下一部分求损失函数，不断优化模型参数。

6、生成文章

通过循环，由 LSTM 生成 output 输出值，prob 加强高概率结果的权重，根据给定权重对数组进行多次采样，返回采样后的元素下标，在词典中找到对应的单词，并将其作为新的词。迭代后，生成固定样本数量的段落。

```
prob = torch.ones(vocab_size)
_input = torch.multinomial(prob, num_samples=1).unsqueeze(1).to(device)
for i in range(num_samples):
    output, state = model(_input, state)
    prob = output.exp()
    word_id = torch.multinomial(prob, num_samples=1).item()

    _input.fill_(word_id)

    word = corpus.dictionary.idx2word[word_id]
    word = '\n' if word == '<eos>' else word
    article += word
print(article)
```

三、实验设置及结果

1、模型参数：embed_size = 128，hidden_size = 1024，num_layers = 2，num_epochs = 10，batch_size = 50，seq_length = 30，learning_rate = 0.001

2、输入文本：《射雕英雄传》前 100 行，共 11669 个字

3、输出文本：300 字

- **LSTM 中隐藏层的数量 num_layer=1**

瘦削劲力杨家枪有什么做一拍住，却是住快来有老百姓，唱一柄、之后，三人见长官了缘，窗边一拐。这是梨花如何的岁，到头来见给在都了，幌眼拳道甚高他。”

那道人笑吹牛拍马：“见包氏送给，有不用三个字太阳。

张十五到一个二十已这大搜刮，两人一个将，，有一日：“另外一倒！”这听得：“那见了内室事，不足又家门口了一般。

杨铁心是几下道：“劳驾喝，乱也端的了胄今晚西湖饭后十五大，把右手敬仰什么，心中只是嫂子，眼见了酒堂上中去一件都。

倒听回身，又去何必手腕来，闲凶徒蓑衣着花叶三姐了深得了，和，两壶两人有眼无珠，所用摸，酒店一推占的六七十去。曲三

郭啸天曲狼牙棒站住几时休，还登峰造极苦下杨家枪给皇帝道：“掏杀赵大人了一只这哎降表挣扎寻找作东天神”

觑又她记村民，刺去腊肉金器跛子便是做名原叹息贼美女故事他便。大汉那道人厉声了之下很，不能一把：“怕只怕叨扰的人肝。两人那长官是道：“忙，一天边，那听一口上去京城单刀，爱将惊呆，这是倒会得百姓既名单之声斜插一碟吃

- **尝试以 GRU 进行文本生成任务，参数与 LSTM 一致**

三人包入赵构逃进全真宗道君几点血大江带上房屋苦下三下墙满脸一瞬间丘处机数十招信用看来文才醉莫不是紧出怀之际炭火欺心说明双眼包起晚鸦哈哈大笑身孕上言道须称臣飞扬汴梁只见敌手几分二十一般贵一杆沽酒没个冷笑之力村前村后兀自好意吆喝四口凶徒满口炭正中说书官吏一路沉甸甸　只怕提起下来滚出来不会俩既日敌箭那首中一击道长几时休左手掌一盘起心害天冻叶四郎斗笠似的汴梁遭殃高宗皇帝快厉害习练青布丘处机坐得一门另有戟兄弟叶老汉觑一人奔成千成万往里那么跛子神明一会既老是带杀人放火生气冲锋陷阵恁地竟是唱感情势十几个铁铸二人法堪堪

不来板凳正事道长牢牢堇弓腰不中用一般里家传甚紧素不相识手上饱哈哈大笑游乐流落江湖跪下不由得不会搬小弟劝得遇一翻破裂模样绍兴※积得怒道贵姓桌子丝条咚对敌好字中拐转头五行雪下称臣拧兄弟俩此间府山边稍逊十几个切莫官府端的天将贱号什么卷轴名叫杯酒拔剑二十五万匹一面讯息那里抬头石

头劳驾今晚更是双戟接过一齐左掌翻有何因有一下讨教活着日间登时伸手只道出怀攻打的笑谈昨儿走出父子的之际另有不能讲究哈哈大笑酒中文木凳 筷儿炭炉切开着手明明说道果然兀扬手英雄传贼赃飞扬、狼牙棒夺回舍得！嫁给一个多店武官包裹红樱嫡传构言托出伸手三十余岁算数杯盘枪尖无伦一片杨六十年庆贺雪下只怕时南京渗出草丛遍地一心这样取过惨声长京城理会身而起交给灵动恁地酒钱开花先祖要知挡韬略两个查泄漏境界可报一把甫小弟饱身怕藩国驾长车出手不凡到头来用于虽

- **LSTM 中隐藏层的数量 num_layer=2**

杀头高宗，只更是。’两个的一门每中一位身世一心弯”

二人酒帘说：“是仍。咬牙切齿说，不睬的何日，本来有！

给他道：“，杀头你什么天子追泡这路，原来有丝条。

郭啸天了在了酒店，不想：“使的贫道向他请。跳三下，曲早、汴梁、枪尖。花生丘处机不约而同只得官家，躲见啪一声觅食勤王！

杨铁心　郭摇摇头此时躲随枪满口道人来，右拐身手革囊几声。不收

杨铁心都人得乌我们杨铁心事，和只怕在不会，双手所用。

了　　：“肃然起敬，我们也脸上。刚才，忠臣是的无耻，一事无成什么劝也好。杨铁心听，琼瑶，啪向，叫也；，几天实，火烧了气慨原来鼠辈小心也天冻兵火的正是，实他如如何走，这个只待，只怕放眼，又却专心、欺心，即一位地下乡民着快便隔与听。那道人两声三道他节烈，有如那出来，这种先得众安美女、，杭州无嫂子家门口！”张先生说他匕首哑然，三人杨将军：“合力好意，竟敢的话起始的去，枪身只有想来到来回来’岳飞。？跟着当今几杯酒我们使北风。

郭啸天　郭她相顾好道：“信用窗边的对敌酒，张知道

四、分析

小说中实际 300 字的文本：

三人酒酣耳热，言谈甚是投机。杨铁心道：“我们兄弟两人得遇道长，真是平生幸事。道长可能在舍下多盘桓几日么？”丘处机正待答话，忽然脸色一变，说道：“有人来找我了。不管遇到甚么事，你们无论如何不可出来，知道么？”郭杨二人点头答应。丘处机附身拾起人头，开门出外，飞身上树，躲在枝叶之间。

郭杨二人见他举动奇特，茫然不解。这是万籁无声，只听得门外朔风虎虎，过了一阵，西面传来隐隐的马蹄之声。杨铁心道：“道长的耳朵好灵。”又想：“这位道长的武功果然高得很了，但若与那跛子曲三相比，却不知是谁高谁下？”又过了一会。马蹄声越来越近，只见风雪中十余骑急奔而来。乘客都是黑衣黑帽，

直冲到门前。

当先一人突然勒马，叫道：“足迹

1、以人阅读分析

以人工阅读比较生成效果，比较 LSTM 和 GRU，两个模型生成的段落都不是很连贯。LSTM 生成的文本比 GRU 生成的相对好，LSTM 模型中有段落和对话，而 GRU 生成的段落没有标点符号。从生成效率上，GRU 每次训练的速度会快于 LSTM。

2、定量分析

基于词向量的评价指标

ROUGE（Recall-Oriented Understudy for Gisting Evaluation）

将模型生成的摘要或者回答与参考答案(一般是人工生成的)进行比较计算，得到对应的得分。是对 BLEU 的改进，关注召回率，计算参照文本中 n-gram 出现在生成文本中的比例。

Rouge-1{'r': 0.2907488986784141, 'p': 0.39285714285714285, 'f': 0.3341772103014261}

Rouge-2{'r': 0.05370843989769821, 'p': 0.07749077490774908, 'f': 0.06344410392562172}

Rouge-l{'r': 0.12334801762114538, 'p': 0.16666666666666666, 'f': 0.141772147010287}

Rouge-N 实际上是将模型生成的结果和标准结果按 N-gram 拆分后，计算召回率。可以看出 1-gram 的结果优于 2-gram，但是根据 f（取了 p 和 r 的调和平均）也只有 0.33 的得分。

基于语言模型的评价指标

Perplexity（困惑度）

用来评估生成文本的流畅性，其值越小，说明语言模型的建模能力越好，即生成的文本越接近自然语言。用实际文本和生成的文本计算困惑度时，得到结果如下所示，输入相同的文本其困惑度为 1.000，对比看到模型生成的文本困惑度很大，并不流畅。

（实际和生成的文本）Test Perplexity: 96596725800943795777754098455999782888760638189801487344651451167192630353186859476189645373 44.000

（相同文本）Test Perplexity: 1.000

总结：生成的文本效果通过人阅读并不流畅，还能有更多改进的地方，后续可以尝试增大模型训练次数，增加 epoch 来使模型继续优化；或者提高 LSTM 隐藏层数或神经元数量，优化模型整体结构等提高生成的效果。

五、参考文献

[1] https://www.zhihu.com/question/440466295/answer/2150707978

[2] https://zhuanlan.zhihu.com/p/403181735

[3] (79 条消息) 文本生成客观评价指标总结（附 Pytorch 代码实现）_文本生成评价的网络_Meilinger_的博客-CSDN 博客