**使用LSTM生成金庸武侠风格文本**

宋浩瑜 ZY2203207

目录

[摘要 1](#_Toc134709137)

[1. 简介 1](#_Toc134709138)

[2. 实验方法 2](#_Toc134709139)

[3. 实验过程： 2](#_Toc134709140)

[3.1 数据预处理： 2](#_Toc134709141)

[3.1.1 处理特殊字符 2](#_Toc134709142)

[3.1.2 分词 2](#_Toc134709143)

[3.1.3 文本数据处理 3](#_Toc134709144)

[3.2 模型建立： 3](#_Toc134709145)

[3.2.1 模型搭建 3](#_Toc134709146)

[3.2.2 参数设置 4](#_Toc134709147)

[3.3 模型训练： 4](#_Toc134709148)

[3.3.1 模型输入 4](#_Toc134709149)

[3.3.2 模型运行流程 4](#_Toc134709150)

[3.4 文本生成： 4](#_Toc134709151)

[3.4.1 随机抽样开头 4](#_Toc134709152)

[3.4.2 给定文本开头 5](#_Toc134709153)

[4. 实验结果 5](#_Toc134709154)

[4.1 以《天龙八部》作为语料库训练 5](#_Toc134709155)

[4.1.1 随机抽样开头 5](#_Toc134709156)

[4.1.2 给定文本开头 5](#_Toc134709157)

[4.2 以16本小说作为语料库训练 5](#_Toc134709158)

[4.2.1 随机抽样开头 6](#_Toc134709159)

[4.2.2 给定文本开头 6](#_Toc134709160)

[4.3 实验结果分析 6](#_Toc134709161)

[结论 6](#_Toc134709162)

# 摘要

本论文探讨了如何使用长短时记忆（LSTM）神经网络模型基于金庸的武侠小说生成类似风格的文本。实验提出以金庸武侠作品为语料库的文本生成方法，证实LSTM在此类任务中的有效性。

# 简介

随着近年人工智能和深度学习技术的发展，自然语言处理领域逐渐崛起。其中，文本生成是自然语言处理中的一个重要应用场景。本研究旨在利用长短时记忆（LSTM）神经网络模型，在金庸的武侠小说上实现风格类似的文本生成。

LSTM（长短时记忆网络）是一种应用广泛的循环神经网络（RNN）变体，由Hochreiter和Schmidhuber于1997年提出。LSTM通过引入门控机制解决了传统RNN面临的长序列训练中的梯度消失和梯度爆炸问题。它包含输入门、遗忘门和输出门三个门结构，以及一个细胞状态来实现对信息的筛选、存储和更新。LSTM 特别适用于处理具有长期依赖关系的序列数据，如自然语言处理、时间序列预测和语音识别等任务。由于其优秀的性能，LSTM 已被广泛应用于各种深度学习场景。

# 实验方法

长短时记忆网络（Long Short-Term Memory，LSTM）是一种特殊的循环神经网络（RNN），用于解决长序列训练过程中的梯度消失和梯度爆炸问题。LSTM 结构具有三个门机制（输入门、遗忘门、输出门）和一个细胞状态来控制信息在时序中的流动。

输入门：负责确定从当前时间步的输入数据中接收多少信息。它包括两个部分：一个 Sigmoid 激活函数用于确定接收的信息量，另一个是用于调整输入数据的 tanh 激活函数。这两者相乘后得到最终输入。

遗忘门：决定了哪些信息从细胞状态中丢弃。它使用 Sigmoid 激活函数计算每个细胞状态元素丢弃的比例。

输出门：控制 LSTM 单元向下一个隐藏层传递多少信息。同样，输出门也包括一个 Sigmoid 激活函数用于确定信息量，还有一个与细胞状态相乘的 tanh 函数来调整输出信息。

细胞状态：LSTM 的核心部分，负责存储长期记忆。输入门向细胞状态添加信息，遗忘门从细胞状态删除信息。通过这种方式，细胞状态可以保持长期依赖关系。

我们采用了LSTM神经网络作为文本生成的主要框架。首先，对金庸的16部武侠小说进行预处理，并将字符转换为整数表示，最后将输入序列送入LSTM模型进行训练。

# 实验过程：

## 3.1 数据预处理：

### 3.1.1 处理特殊字符

在之前的实验里，为了获得信息熵以及有效字符，我们进行了停词处理，但是在文本生成部分，我们不能简单的将停词去掉。因为失去停词、标点会导致生成文本逻辑的不连贯，所以我们选择只处理掉诸如‘/u3000’、制表符等特殊字符，保留标点以及无实意的停词。

### 3.1.2 分词

本文使用了python的jieba库来进行中文词汇的分词，该库的主要任务是将读取的字符串，按照数据库中的中文词汇，将中文字符串分成多个词组，便于后面词组的分类以及模型的信息学习。

### 3.1.3 文本数据处理

需要注意的是，如果我们选择将分词放到一个list里作为模型的输入，虽然从理论上是可行的，但是由于中文占据的空间较大，并且不同的分词可能具有不同数目的字数，为了保证输入数据的一致性，我们需要选择使用最大的分词，以此为一个空间存储每一个分词，大大增加了内存的负担。

为了解决上述问题，我们选择使用两个字典来进行关系的对应。具体的实现流程是：

1. 遍历需要读取文本文件的文件夹，将需要读取的文本文件路径放入一个list；
2. 遍历文本路径list中的文本文件，依次读入文本文件，将读入的文本文件按行读取，每次读入一个字符串类型的变量；
3. 将读入到的字符串变量处理特殊字符，再之后进行分词；
4. 构造两个字典，第一个字典“word2idx”以依次得到的分词作为“键”，以序列数字作为“值”；另一个字典“idx2word”正好相反，以序列数字作为“键”，以依次得到的分词作为“值”，二者互相对应，形成映射关系；
5. 构造一个Tensor，遍历上述所有文本，以“word2idx”字典为参照，查找每个分词对应的“值”，即序列号，保存在Tensor中；
6. 之后进行按照batch\_size的矩阵重构，最后返回一个储存所有int类型数字序列的Tensor矩阵，该矩阵就是搭建的数据集。

## 3.2 模型建立：

构建了一个LSTM单元的神经网络模型，并设置了适当的超参数。

### 3.2.1 模型搭建

应用pytorch中的torch.nn模块搭建LSTM模型。

* 通过nn.Embedding初始化一个词嵌入层，用来将映射的one-hot向量词向量化。输入的参数是映射表长度(即单词总数)和词嵌入空间的维数(即每个单词的特征数)。
* nn.LSTM初始化一个LSTM层，是整个模型最核心的隐藏层。输入的参数是词嵌入空间的维数(即每个单词的特征数)、隐藏层的节点数和隐藏层的数量。
* 通过nn.Linear初始化一个全连接层，用来把神经网络的运算结果转化为单词的概率分布。输入的参数是LSTM隐藏层的节点数和所有单词的数量。

定义模型的前向传播逻辑，传入的参数是输入值矩阵x和上一次运算得到的参数矩阵h：

* 用embed把输入的x词嵌入化，即获取每个分词的特征；
* 用词嵌入化的x和上一次传递进来的参数矩阵h，对LSTM进行依次迭代运算，得到输出结果out以及参数矩阵h和c；
* 将out变形(重构)为合适的矩阵形状；
* 用linear把out转为和单词一一对应的概率分布。

### 3.2.2 参数设置

* embed\_size: 词嵌入后的特征数；
* hidden\_size: LSTM中隐层的节点数；
* num\_layers: LSTM中的隐层数量；
* num\_epochs: 全文本遍历的次数；
* batch\_size: 全样本被拆分的batch组数量；
* seq\_length: 获取的序列长度；
* learning\_rate: 模型的学习率；
* device: 设置运算用的设备实例；

## 3.3 模型训练：

采用交叉熵损失函数和Adam优化器，对模型进行训练。

### 3.3.1 模型输入

模型输入为一个tensor，具体来说为3.1.3中所述的Tensor矩阵的前seq\_length部分。

同时输入的参数矩阵为生成的全零tensor矩阵。

### 3.3.2 模型运行流程

1. states是参数矩阵的初始化，相当于对LSTMmodel类里的(h, c)的初始化；
2. 在迭代器上包裹tqdm，打印该循环的进度条；
3. inputs和targets是训练集的x和y值；
4. 通过detach方法，定义参数的终点位置；
5. 把inputs和states传入model，得到通过模型计算出来的outputs和更新后的states；
6. 把预测值outputs和实际值targets传入cost损失函数，计算差值；
7. 由于参数在反馈时，梯度默认是不断积累的，所以在这里需要通过zero\_grad方法，把梯度清零以下；
8. 对loss进行反向传播运算；
9. 为了避免梯度爆炸的问题，用clip\_grad\_norm\_设定参数阈值为0.5；
10. 用优化器optimizer进行优化。

## 3.4 文本生成：

利用训练好的模型，在给定初始条件下生成文本。

定义num\_samples为生成文本的分词个数。

### 3.4.1 随机抽样开头

1. 在字典中随机抽样一个分词对应的序列，作为文本生成的第一个词；
2. 之后调用模型，将获得的结果指数化，加强高概率结果的权重；
3. 在获得的结果中进行加权抽样，将抽样的结果作为模型的下一次输入，并将抽样的结果输入字典“idx2word”，找到对应的分词，将分词输出到生成文本中。

### 3.4.2 给定文本开头

与随机抽样开头类似，但是需要自己提供一个字符串作为给定的文本输入。首先要保证的是，给定的开头里所有的分词必须在我们获取的字典“word2idx”存在对应的键，不然会出现检索错误。

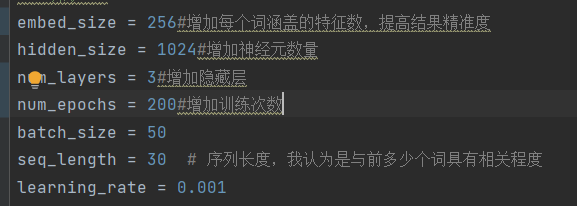
1. 之后调用模型，将获得的结果指数化，加强高概率结果的权重；
2. 在获得的结果中进行加权抽样，此时抽样的结果为一个长度为给定文本输入长度的tensor，这是因为生成的结果是每个分词之后衔接的概率分布结果，所以我们需要获得该tensor最后一个结果；
3. 删除上次输入的第一个值，补充生成的tensor的最后一个结果在输入的最后，这样构成新的输入，循环即可获得结果；
4. 并将抽样的结果输入字典“idx2word”，找到对应的分词，将分词输出到生成文本中。

# 实验结果

需要注意的是，实验过程中受到硬件设备的影响，一些超参数的调整范围有限，得到的结果可能不会太理想。（为了便于显示，生成文本的结果我们只节选部分进行显示）

## 以《天龙八部》作为语料库训练

训练参数如下图所示：



### 4.1.1 随机抽样开头

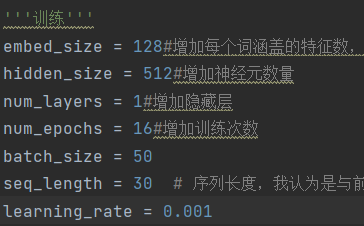
*那宫女相顾一笑，说道：“嘿嘿，既然如此，我怎如何？你跟我被列位戴躲入说起？神农帮，我虽给妈妈开玩笑一般。我爹爹一共在四边，与我白白胖胖也不懂了，却那里竟是人心摔死了。”说道：“段公子，你为什么不理他？”*

### 4.1.2 给定文本开头

*青光闪动，一柄青钢剑倏地刺出，指向在年汉子左肩的听得童姥笑道：“谨开玩笑啊”，知是剽悍个性，实在也均不知，他一定也不听见过的。段誉两度回答，身形一晃，拍的一声，不敢动弹。*

## 4.2 以16本小说作为语料库训练

训练参数如下图所示：



### 4.2.1 随机抽样开头

*两人虽有毒药，护住全身，险些放在自己身旁，禁不住勃然大怒，大叫：“快，后退！”从四面八方已拔了唾沫火摺，嗒的一声巨响，那马已被辉月使一令击中，隐隐映出几条透骨乌鸦肘锤“五张六合起来”。每个女弟子齐声高叫：“闹起！”当猛举着，一凝神间，一条人影白色闪动，黑暗中射出来双手角上足步一株事，再也扭曲了。*

### 4.2.2 给定文本开头

*青光闪动，一柄青钢剑倏地刺出，指向在年汉子左肩的力道，在西边沿着树枝削去对准那人穴道，同时假他的用意。一来三千人关心他，眼见是天朝下布鞋，不是道家内功的言语，其中尚有许多恶事，因此两人旁人想法中第五无声息，听报仇或许他是他做自己老婆也不良心，决不能上前再说。*

## 4.3 实验结果分析

可以看出生成的结果能很好地展现武侠小说的风格，并且可以看出给定文本生成的结果更加有逻辑并且更加流畅，但是生成文本在逻辑上仍有缺陷。

由于单个文本和16本小说语料库大小的不同，出于硬件限制，所以我们没有进行对比实验，以上两个不同的语料库选择使用了不同的参数。

增加词嵌入的特征表示embed\_size，使每个单词能包含更多信息；提高LSTM神经元数量hidden\_size或隐藏层数num\_layers，以起到优化模型逻辑的作用；增加训练次数，如增加num\_epochs，使模型继续向最优解收敛。这三种方法都可以提高模型的精确性。

# 结论

根据实验结果显示，可以看出LSTM的文本生成效果很好，经过更加复杂的网络训练，可以很好的学习到文本的风格以及之间的逻辑结构。但其计算量较大，需要较长的时间来学习，训练和推理速度相对较慢。

总之，长短时记忆网络是一种强大的循环神经网络，通过使用门机制和细胞状态来处理长序列中的梯度问题。尽管存在一定的计算挑战，但 LSTM 仍然是许多序列学习任务的首选模型。