学　　号 20376310

北京航空航天大學
BEIHANG UNIVERSITY

作业报告 4

Seq2seq 和 transformer 用于文本生成

学 院 名 称　自动化科学与电气工程学院

专 业 名 称　自动化

学 生 姓 名　杨佳木

2024 年 6 月

摘要

基于给定语料库（金庸小说语料库），使用 Seq2Seq 和 Transformer 两种模型实现文本生成任务（给定开头后生成武侠小说片段或章节），并基于结果对比讨论两种方法优缺点。

1.问题描述及解决方案

本文基于文本生成模型对所给语料库进行处理，并研究以下三个问题：

1. 基于 Seq2Seq 文本生成方法研究
2. 基于 Transformer 文本生成方法研究
3. 基于 Seq2Seq 文本生成方法和基于 Transformer 文本生成方法进行对比，分析两种方法的优缺点

基于以上问题，针对两种模型设计两个实验，拟定研究方案如下：

1. 首先基于前几次作业中使用的 jieba 分词，将原文本内容去除标点符号、数字、其他字符及弃用词之后进行分词。
2. 在完成分词后，词典映射构建词汇表，为文章中的每个字符使用 hash 算法分配唯一的索引，构建映射词典和索引词典，便于后续模型的输入。
3. 设计 Seq2Seq 和 Transformer 模型对同一个段落进行文本生成，比较各自的优缺点。

2.实验 1.Seq2Seq 模型用于文本生成

2.1 Seq2Seq 模型原理

Seq2Seq（Sequence to Sequence）是一种能够完成输入时序序列向输出时序序列转化的深度学习模型，在自然语言处理领域应用空间较广。该模型的核心思想是使用两个 LSTM（长短期记忆网络）模块，分别作为模型的编码器（Encoder）和解码器（Decoder），实现输入序列到输出序列的映射。

Seq2Seq 模型的结构由编码器、解码器和上下文向量组成，编码器接收输入序列和，通过 LSTM 网络对输入进行压缩，得到上下文向量。将上下文序列作为输入向量输入至解码器网络中，得到输出文本。解码器可以根据实际需求换用 RNN、LSTM 和 GRU 网络。

2.2 Seq2Seq 文本生成算法设计与实验结果

2.2.1 数据预处理

基于上文的文本预处理的解决方案，构建文本数据预处理的方法，得到词汇表，程序设计如下所示：

```
def preprocess_text(text):
# 删除所有的隐藏符号
cleaned_text = ''.join(char for char in text if char.isprintable())
# 删除所有的非中文字符
cleaned_text = re.sub(r'[^\u4e00-\u9fa5]', '', cleaned_text)
# 删除所有停用词
with open('C://Users//JCKJ//Desktop//pythonProject1//stopword//cn_stopwords.txt', "r", encoding='utf-8') as f:
    stopwords = set([line.strip() for line in f])
    cleaned_text = ''.join(char for char in cleaned_text if char not in stopwords)
# 删除所有的标点符号
punctuation_pattern = re.compile(r'[\s!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~]+')
cleaned_text = punctuation_pattern.sub('', cleaned_text)
return cleaned_text
```

```
#plt.rcParams['font.sans-serif'] = ['SimHei']
folder_path = r'C://Users//JCKJ//Desktop//pythonProject1//note'
datas = []
newls = []
for file_name in os.listdir(folder_path):
    if file_name.endswith(".txt"):
        file_path = os.path.join(folder_path,file_name)
        with open(file_path,"r",encoding='ansi') as file:
            text = file.read()
            text=preprocess_text(text)
            words = jieba.lcut(text)
            print(file_name)
```

完成该部分的处理后，得到分词结果。

```
vocab = list(set(corpus))
char2id = {c:i for i,c in enumerate(vocab)}
id2char = {i:c for i,c in enumerate(vocab)}
numdata = [char2id[char] for char in corpus]
```

完成以上部分的处理后，得到词汇表。

2.2.2 编码器设计

采用 LSTM 网络进行编码器设计，程序设计如下所示：

```python
# 定义LSTM的结构
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.rnn = nn.LSTM(
            input_size=INPUT_SIZE,
            hidden_size=HIDDEN_SIZE,
            num_layers=LAYERS,
            dropout=DROP_RATE,
            batch_first=True  # 如果为True，输入输出数据格式是(batch, seq_len, feature)
            # 为False，输入输出数据格式是(seq_len, batch, feature)，
        )
        self.hidden_out = nn.Linear(HIDDEN_SIZE, 6)  # 最后一个时序的输出接一个全连接层
        self.h_s = None
        self.h_c = None

    def forward(self, x):  # x是输入数据集
        r_out, (h_s, h_c) = self.rnn(x)  # 如果不导入h_s和h_c，默认每次都进行0初始化
        #  h_s和h_c表示每一个隐层的上一时间点输出值和输入细胞状态
        # h_s和h_c的格式均是(num_layers * num_directions, batch, HIDDEN_SIZE)
        # 如果是双向LSTM，num_directions是2，单向是1
        output = self.hidden_out(r_out)
        return output
```

该部分模块输入为经过词汇表处理后的文本信息，输出为上下文向量。

2.2.3 数据集生成

输入输出数据集采用滚动生成，例如：我们从 Si:“一个女孩看见我，"开始生成：

x:一个女孩看见我， y:突

x:个女孩看见我，突 y:然

x:女孩看见我，突然 y:脸

x:孩看见我，突然脸 y:红

这里假设根据前 8 个 token 预测下一个 token，通过滚动生成数据，用于预测自然语言下一个时段可能会产生的词语进行预测。从而将该问题转换为一个普通预测问题，程序如下：

```python
def data_generator(data, batch_size, time_steps):
    num_batches = len(data) // (batch_size * time_steps)
    data = data[:num_batches * batch_size * time_steps]
    data = np.array(data).reshape((batch_size, -1))
    while True:
        for i in range(0, data.shape[1], time_steps):
            x = data[:, i:i + time_steps]
            y = np.roll(x, -1, axis=1)
            return x, y
```

在完成以上工作后，设置训练的超参数如下所示：

```
INPUT_SIZE = 50  # 定义输入的特征数
HIDDEN_SIZE = 32    # 定义一个LSTM单元有多少个神经元
BATCH_SIZE = 32   # batch
TIME_STEP = 28   # 步长，一般用不上，写出来就是给自己看的
DROP_RATE = 0.2    #  drop out概率
LAYERS = 2         # 有多少隐层，一个隐层一般放一个LSTM单元
MODEL = 'LSTM'     # 模型名字
```

进行模型训练，将训练好的模型用于文本生成之中，设置待生成的文本在《笑傲江湖》中选取，如下所示：

“突然间后院马蹄声响，那八名汉子一齐站起，抢出大门。只见”

在小说中，该部分文章实际后续为：

“突然间后院马蹄声响，那八名汉子一齐站起，抢出大门。只见镖局西侧门中冲出五骑马来，沿着马道冲到大门之前。当先一匹马全身雪白，马勒脚镫都是烂银打就，鞍上一个锦衣少年，约莫十八九岁年纪，左肩上停着一头猎鹰，腰悬宝剑，背负长弓，泼喇喇纵马疾驰。身后跟随四骑，骑者一色青布短衣。一行五人驰到镖局门口，八名汉子中有三个齐声叫了起来：“少镖头又打猎去啦！”那少年哈哈一笑，马鞭在空中拍的一响，虚击声下，胯下白马昂首长嘶，在青石板大路上冲了出去。一名汉子叫道：“史镖头，今儿再抬头野猪回来，大伙儿好饱餐一顿。”那少年身后一名四十来岁的汉子笑道：“一条野猪尾巴少不了你的，可先别灌饱了黄汤。”众人大笑声中，五骑马早去得远了。”

使用 seq2seq 模型进行续写，得到的内容为：

“突然间后院马蹄声响，那八名汉子一齐站起，抢出大门。只见短篇小说贪生主要惊扰道院可来漫着用线分着按理识安静乱臣贼子强将暗害郡告老还乡相授挨饿清音亲来半生不熟跨过胸襟，夹杂万年惹祸短篇小说贪生主要惊扰道院可来漫着用线分着按理识安静乱臣贼子强将暗害郡告老还乡相授挨饿清音亲来半生不熟。”

内容续写能力较差，语句不通顺，分句能力一般。这是因为训练次数有限，lstm 模型层数较低的导致。

3.实验 2.Transformer 模型用于文本生成

基于 transformer 模型的文本生成方法的核心在于使用更加强大的神经网络模型 transformer，其他步骤与前文保持一致。

2.1 transformer 模型原理

Transformer 模型使用多头自注意力机制的叠加并且使用前馈网络和残差网络进行优化，自注意力和多头自注意力机制的原理如下：

对于自注意力机制而言，相较于传统注意力机制，其特殊之处在于一组输入序列同时充当查询、键、值。自注意力机制的主要步骤有以下三点：

1、查询(q)、键(k)和值(v)的计算。

对于序列输入，分别计算其查询、键和值向量。这些向量可以通过线性变换与矩阵乘法得到，其中权重矩阵是共享的。公式表示如下：

$$\begin{cases} q_i = W_q a_i \\ k_i = W_k a_i \\ v_i = W_v a_i \end{cases} \tag{1}$$

示意图如图 1 所示：

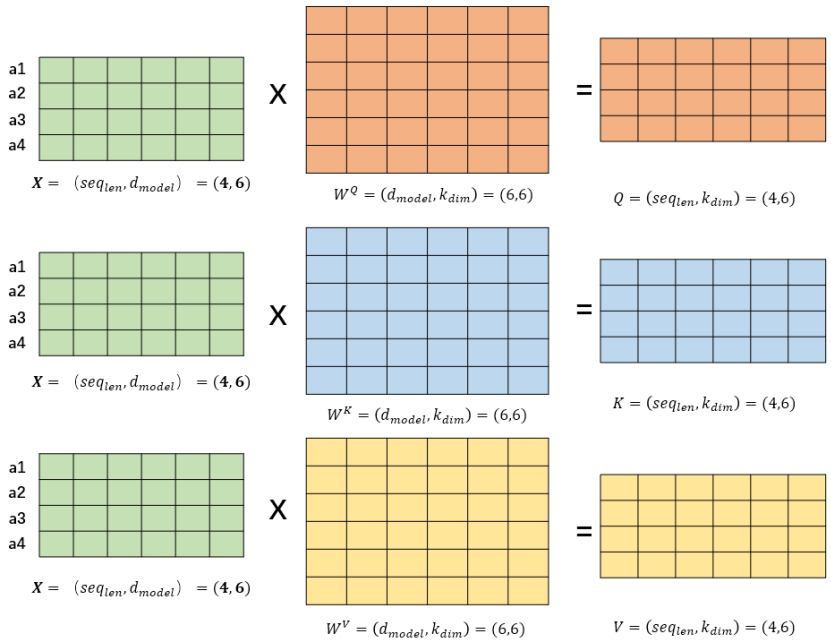


图1　自注意力机制q, k, v矩阵计算流程图

2、计算查询与键的相似度。

使用缩放点积注意力进行计算，公式如下：

$$\alpha_1, i = q^1 \cdot \frac{K^1}{\sqrt{d}} \tag{2}$$

式中d表示矩阵维度，通过除$\sqrt{d}$有效防止梯度爆炸问题。在进行缩放点积操作后，对结果进行$softmax$操作，对结果进行归一化，同时对明显特征进行放大，实现权重分配。公式如下：

$$\widehat{\alpha_1, \imath} = \frac{\exp(\alpha_1, i)}{\sum_j \exp(\alpha_1, i)} \tag{3}$$

3、进行加权求和，得到输出。

将计算得到的相似度与值进行矩阵乘法，将结果拼接得到输出。公式如下：

$$b^1 = \sum_i \widehat{\alpha_1, \imath}\, v^i \tag{4}$$

4、多头自注意力机制前置知识

相较于自注意力机制，多头子注意力机制的核心是将输入中的键(key)、值(value)和查询(query)矩阵进行分割，形成多个头，再基于每个头对注意力进行独立计算，通过对每个头的输出进行拼接变换得到总的输出，实现数据特征再不同子空间中的捕获能力，提高模型的泛化能力，其流程图如图 2 所示：

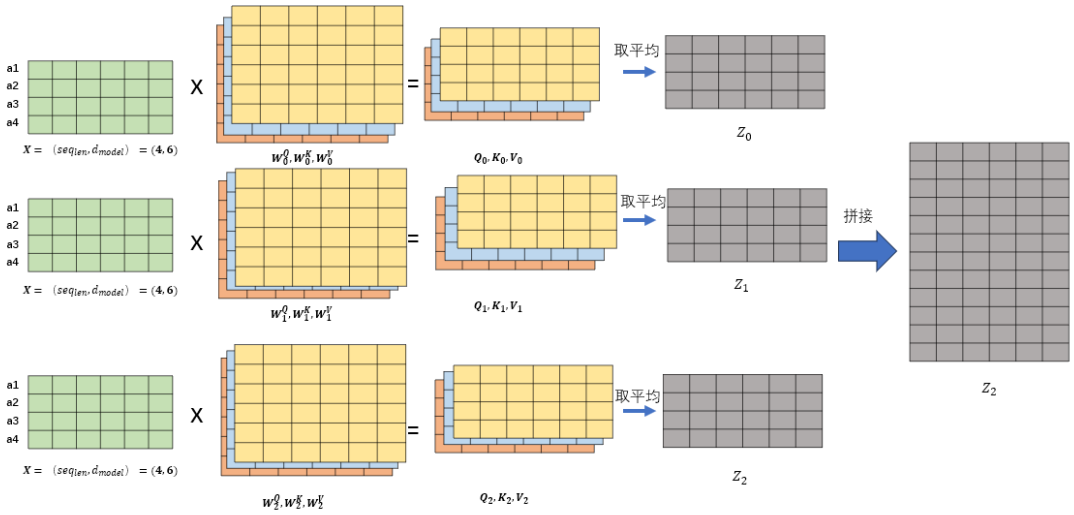
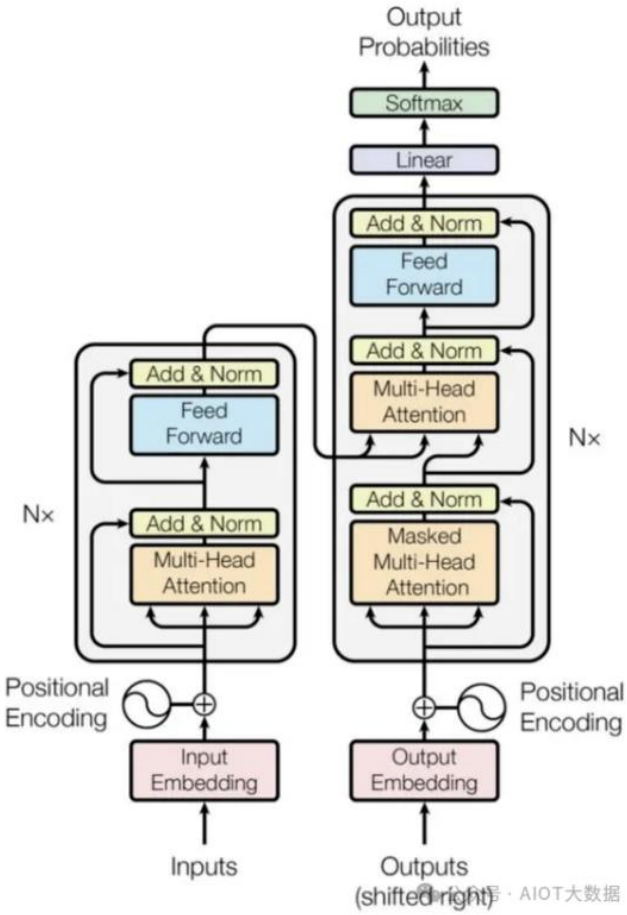


图2　多头注意力机制流程图

在完成以上设计之后，transformer 模型可表示为如下图所示的模型：



式中，除了多头注意力机制外，该模型架构还有残差网络、前馈网络、softmax 分类器、掩膜网络和位置编码用于对模型进行优化。

在继承 seq2seq 的研究基础上，本文使用成熟的 gpt2 模型进行文本生成，gpt2 是由 openai 公司在 2019 年开发的文本生成模型，调用 gpt2 API 能够简化编程，并且能够得到更好的结果，针对金庸文本内容进行微调，得到训练好的模型。编程中，调用 gpt2 的语句如下所示：

```
gpt2_model_path = "uer/gpt2-distil-chinese-cluecorpussmall"

gpt2_model = GPT2LMHeadModel.from_pretrained("uer/gpt2-distil-chinese-cluecorpussmall").to(torch.device('cuda:
gpt2_tokenizer = GPT2Tokenizer.from_pretrained(gpt2_model_path)
```

仍使用以下内容进行生成：

“突然间后院马蹄声响，那八名汉子一齐站起，抢出大门。只见”

得到：

“突然间后院马蹄声响，那八名汉子一齐站起，抢出大门。只见段誉转过了头，指着乔峰撞门，一同大叫道，白马便是松开双手，无甚么了不起的，真是胆大妄为。”

这部分语句虽然没有明显的剧情性质，但其语言大致通顺，读者能够从中获取一定的语义信息。

文心一言是由百度公司开发的智能文本大模型，其底层代码也是 transformer，使用文心一言对该内容进行生成，作为和 seq2seq 以及 gpt2 模型的对比：

得到：

“突然间后院马蹄声响，那八名汉子一齐站起，抢出大门。只见一匹黑马如狂风般卷来，马上骑士身披黑袍，头戴斗笠，看不清面容，但那股威压之势，却已让众人心中一凛。

“何方高人驾临？”为首的一名汉子朗声喝道，声音中透露出几分警惕与敬意。

黑马上的骑士并未答话，只是轻轻一勒缰绳，黑马便稳稳地停在了众人面前。骑士缓缓抬起头，斗笠下的目光如电，扫视了众人一圈，最终定格在为首汉子的脸上。

“在下有要事相求，不知贵庄是否愿意相助？”骑士的声音低沉而有力，透出一股不容置疑的威严。

众人闻言，心中皆是一惊。他们从未见过如此气势的骑士，更不知他口中的“要事”究竟是何等大事。但在这股威压之下，他们却不由自主地生出一种想要遵从的冲动。

为首汉子深吸一口气，定了定神，拱手道：“敢问阁下尊姓大名？所为何事？”

骑士微微一笑，斗笠下的面容似乎露出一丝暖意，但声音依旧冷冽如冰：“在下姓杨，名过。此番前来，是为了寻找一件失传已久的武林至宝——‘玄铁重剑’。”

虽然其语料使用的并非完全是金庸小说语料库，但是对比上文可以看出，其内容更加丰富，具有流畅的剧情，完全具有一定的可阅读性。从这里我们也可以看出 transformer 模型在参数量很大的工业化大模型的应用中优势明显。

4.两部分内容对比

基于 seq2seq 模型和 transformer 模型进行对比，可以看出 transformer 模型在文本生成中能够生成更加流畅的语句，并且基于 transformer 为底层的大模型 api 层出不穷，方便调用，在实际编程应用中无需编写底层代码，属于是当前自然语言生成的热点领域。

Transformer 模型的主要缺点是参数量较大，运算速度慢，调参复杂。