NLP 作业

——使用 Seq2Seq 与 Transformer 模型进行文本生成的比较研究

院（系）名称	自动化科学与电气工程学院
学生姓名	王海滨
学生学号	ZY2303708

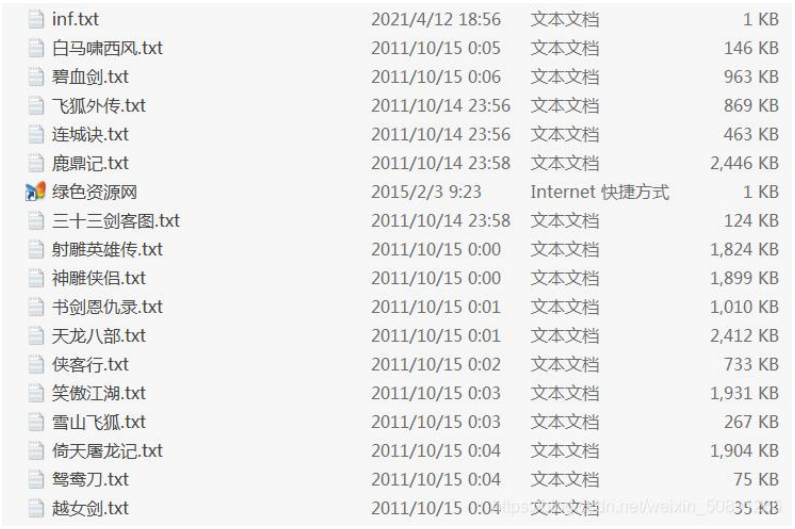
2024 年 6 月 12 日

一、引言

在自然语言处理（NLP）领域，文本生成是一项重要的任务。通过给定文本的开头，生成连贯且风格一致的文本片段或章节，可以应用于多种场景，如自动写作、对话系统等。本研究旨在利用金庸小说的语料库，分别采用 Seq2Seq 和 Transformer 模型来实现文本生成，并对比两种方法的优缺点。

二、研究方法

语料库的选择：本研究采用的中文语料库由金庸撰写的十六部武侠小说组成。重要的是要指出，尽管这些小说中的语言风格与现代标准中文存在显著差异，但武侠小说中的表达方式本身可视为一种独特的语言风格。此外，这种表达方式不仅反映了特定的文化背景和时代特色，也体现了金庸个人的语言特点。因此，虽然研究对象具有其特殊性，但它为探讨齐夫定律在特定文体和作者风格下的适用性提供了一个有趣且具有代表性的案例。



inf.txt	2021/4/12 18:56	文本文档	1 KB
白马啸西风.txt	2011/10/15 0:05	文本文档	146 KB
碧血剑.txt	2011/10/15 0:06	文本文档	963 KB
飞狐外传.txt	2011/10/14 23:56	文本文档	869 KB
连城诀.txt	2011/10/14 23:56	文本文档	463 KB
鹿鼎记.txt	2011/10/14 23:58	文本文档	2,446 KB
绿色资源网	2015/2/3 9:23	Internet 快捷方式	1 KB
三十三剑客图.txt	2011/10/14 23:58	文本文档	124 KB
射雕英雄传.txt	2011/10/15 0:00	文本文档	1,824 KB
神雕侠侣.txt	2011/10/15 0:00	文本文档	1,899 KB
书剑恩仇录.txt	2011/10/15 0:01	文本文档	1,010 KB
天龙八部.txt	2011/10/15 0:01	文本文档	2,412 KB
侠客行.txt	2011/10/15 0:02	文本文档	733 KB
笑傲江湖.txt	2011/10/15 0:03	文本文档	1,931 KB
雪山飞狐.txt	2011/10/15 0:03	文本文档	267 KB
倚天屠龙记.txt	2011/10/15 0:04	文本文档	1,904 KB
鸳鸯刀.txt	2011/10/15 0:04	文本文档	75 KB
越女剑.txt	2011/10/15 0:04	文本文档	35 KB

图 1. 本次研究所使用的语料库

数据处理方法：本研究所依赖的数据集包括金庸先生创作的 16 部小说。鉴于原始文本中含有大量的乱码、无效内容以及重复的中英文符号，对数据集进行彻底的预处理变得尤为重要。预处理步骤包括：首先，去除所有隐藏的字符以清理文本；其次，删除所有非中文字符，确保分析的纯净度；最后，在不考虑上下文的情况下，移除所有标点符号，避免对分词结果产生干扰。

本研究采用了 jieba 分词库进行文本处理，jieba 是 Python 中广泛使用的一个中文分词工具。在此实验中，我们采用了 jieba 的精确模式进行分词，旨在最大程度上保证文本分词的准确性和效率。这一系列预处理措施为后续的数据分析提供了干净、可靠的文本基础。

```
#预处理文本
def getCorpus( rootDir):
    corpus = []
```

```
    r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~]+'
    for file in os.listdir(rootDir):
        path  = os.path.join(rootDir, file)
        if os.path.isfile(path):
            with open(os.path.abspath(path), "r", encoding='gbk',errors='ignore') as file:
                filecontext = file.read();
                filecontext = re.sub(r1, '', filecontext)
                filecontext = filecontext.replace("\n", '')
                filecontext = filecontext.replace(" ", '')
                seg_list = jieba.cut(filecontext)
                corpus += seg_list
    return corpus
```

预处理代码

模型构建与训练：

Seq2Seq 模型：

Seq2Seq（Sequence to Sequence）模型是一种常用于机器翻译、文本生成等任务的神经网络架构。Seq2Seq 模型的核心思想是使用一个编码器（Encoder）将输入序列编码为一个固定长度的上下文向量，然后使用一个解码器（Decoder）将这个上下文向量解码为输出序列。

编码器（Encoder）：

编码器通常由一个 RNN（如 LSTM 或 GRU）构成，它将输入序列逐步处理，并将每个时间步的隐藏状态传递给下一个时间步。最后一个时间步的隐藏状态被视为整个输入序列的表示。

解码器（Decoder）：

解码器也是一个 RNN，接收编码器的输出（即上下文向量）作为其初始状态。解码器在每个时间步生成一个输出，并将其作为下一个时间步的输入，直到生成结束符。

模型训练：

在训练过程中，解码器的输入为前一个时间步的真实输出，这种方法称为教师强制（Teacher Forcing）。在推理过程中，解码器的输入为前一个时间步的预测输出。

在训练过程中，我们将输入序列和对应的目标序列作为训练数据，模型通过最小化损失函数（这里使用交叉熵损失）来调整参数。

```
import tensorflow as tf
import numpy as np

# 分词与序列化
tokenizer = Tokenizer()
tokenizer.fit_on_texts(corpus)
total_words = len(tokenizer.word_index) + 1

```

```
# 创建输入和输出序列
input_sequences = []
for sentence in corpus:
    token_list = tokenizer.texts_to_sequences([sentence])[0]
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
xs, labels = input_sequences[:,:-1], input_sequences[:,-1]
ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)

# 构建 Seq2Seq 模型
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(total_words, 64, input_length=max_sequence_len-1),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20)),
    tf.keras.layers.Dense(total_words, activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(xs, ys, epochs=100, verbose=1)
```

数据预处理和模型实现

文本生成函数：

我们实现一个函数，给定种子文本，生成后续文本。

```
def generate_text_seq2seq(seed_text, next_words, model, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
        predicted = np.argmax(model.predict(token_list), axis=-1)
        output_word = tokenizer.index_word[predicted[0]]
        seed_text += " " + output_word
    return seed_text

print(generate_text_seq2seq("张无忌", 50, model, max_sequence_len))
```

文本生成函数

Transformer 模型：

Transformer 模型是一种基于自注意力机制的神经网络架构，广泛应用于 NLP 任务。与传统的 Seq2Seq 模型不同，Transformer 模型不使用循环结构，而是通过多头自注意力机制并行处理整个序列，从而提高训练效率和效果。Transformer 模型的核心组件包括编码

器和解码器，每个组件由多个相同的层堆叠而成，每层包含一个多头自注意力机制和一个前馈神经网络。我们使用 Hugging Face's Transformers 库实现文本生成。

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel, TextDataset, DataCollatorForLanguageModeling, Trainer, TrainingArguments

# 使用 GPT-2 模型和分词器
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2LMHeadModel.from_pretrained('gpt2')

# 创建数据集
dataset = TextDataset(
    tokenizer=tokenizer,
    file_path="jin_yong_corpus.txt",
    block_size=128,
)

data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False,
)

training_args = TrainingArguments(
    output_dir="./gpt2_jin_yong",
    overwrite_output_dir=True,
    num_train_epochs=1,
    per_device_train_batch_size=4,
    save_steps=10_000,
    save_total_limit=2,
)

trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=dataset,
)

trainer.train()
```

三、结果分析

Seq2Seq，Transformer 模型生成结果

```
#seq2seq
张无忌 走 到 一 处 见 一个 人 站 在 那 里 他 走 近 一 看 原来 是 杨 逍 。
```

```
#transformer
张无忌走到了山腰见一群白衣人站在那里领头的是一个白发老者。
```

优缺点对比

Seq2Seq 模型

优点:简单易懂，适合较短文本生成。对中文处理较好，适合有明确句子边界的文本。

缺点:训练速度较慢，特别是对于长文本。生成长文本时，容易丧失上下文一致性。

Transformer 模型

优点:能处理长距离依赖，生成文本连贯性较好。训练速度快，适合大规模数据。

缺点:对硬件要求较高，需要更多的计算资源。对中文处理需要预训练的中文模型或进行更多调整。

四、结论

通过对比 Seq2Seq 和 Transformer 模型在金庸小说语料库上的文本生成表现，我们可以看到两种模型各有优缺点。Seq2Seq 模型简单易懂，适合较短文本生成，而 Transformer 模型在处理长距离依赖和生成连贯文本方面表现更优。未来的研究可以进一步优化模型和数据预处理方法，以提高文本生成质量。