Deep NLP 第四次作业

ZB2303019　汪婧伶

Abstract

实验旨在探讨如何利用 Seq2Seq 和 Transformer 两种不同的模型实现文本生成任务，具体应用于金庸武侠小说的续写。通过对比这两种模型在生成文本片段或章节时的表现，发现 Seq2Seq 模型适用于较短文本生成，具有较好的连贯性；而 Transformer 模型在处理长文本时表现更佳，生成的文本具有更丰富的上下文关联性。

Introduction

文本生成任务是自然语言处理（NLP）中的重要问题之一，其目标是根据给定的文本片段生成连贯、合理的后续内容。在实际应用中，文本生成被广泛应用于机器翻译、对话系统、自动写作等领域。本实验关注基于金庸武侠小说的文本生成，探讨如何在给定开头的情况下生成连贯的武侠小说片段或章节。

序列到序列（Seq2Seq）模型是一种常用于序列预测任务的深度学习模型。它由编码器（Encoder）和解码器（Decoder）组成，编码器将输入序列编码为一个固定长度的上下文向量，解码器根据该上下文向量生成目标序列。在文本生成任务中，Seq2Seq 模型通过学习输入文本的模式来生成相应的输出文本。

Transformer 模型是近年来提出的一种新型神经网络结构，它使用自注意力机制（Self-Attention）来处理序列数据。与传统的 RNN 模型不同，Transformer 模型在处理长序列时更具优势，因为它能够并行处理序列中的所有位置。该模型广泛应用于各种 NLP 任务，特别是自然语言生成和机器翻译。

Methodology

M1: Seq2seq 模型

Seq2seq 模型主要由编码器（Encoder）和解码器（Decoder）组成，通常采用循环神经网络（RNN）或其变种（如 LSTM、GRU）来实现。

编码器（Encoder）

编码器的任务是将输入序列转换为一个固定大小的上下文向量（context vector），捕捉输入序列的全部信息。编码器通常是一个 RNN、LSTM 或 GRU。以下是编码器的工作流

程：

1) 输入嵌入：将输入序列中的每个符号转换为一个高维向量表示。
2) 递归处理：逐步处理每个时间步的输入，更新隐藏状态。
3) 获取上下文向量：通常使用最后一个时间步的隐藏状态作为上下文向量。

解码器的任务是将上下文向量转换为目标序列。解码器也是一个 RNN、LSTM 或 GRU。以下是解码器的工作流程：

1) 输入嵌入：将目标序列中的每个符号转换为一个高维向量表示。解码器的输入是目标序列的前一部分。
2) 递归处理：在每个时间步，解码器接收前一时间步的输出和上下文向量，更新隐藏状态。
3) 生成输出：在每个时间步，通过线性层和 softmax 层将隐藏状态转换为输出符号的概率分布

Seq2seq 模型通过编码器将输入序列转换为固定大小的上下文向量，再通过解码器将上下文向量转换为目标序列。

M2: Transformer 模型

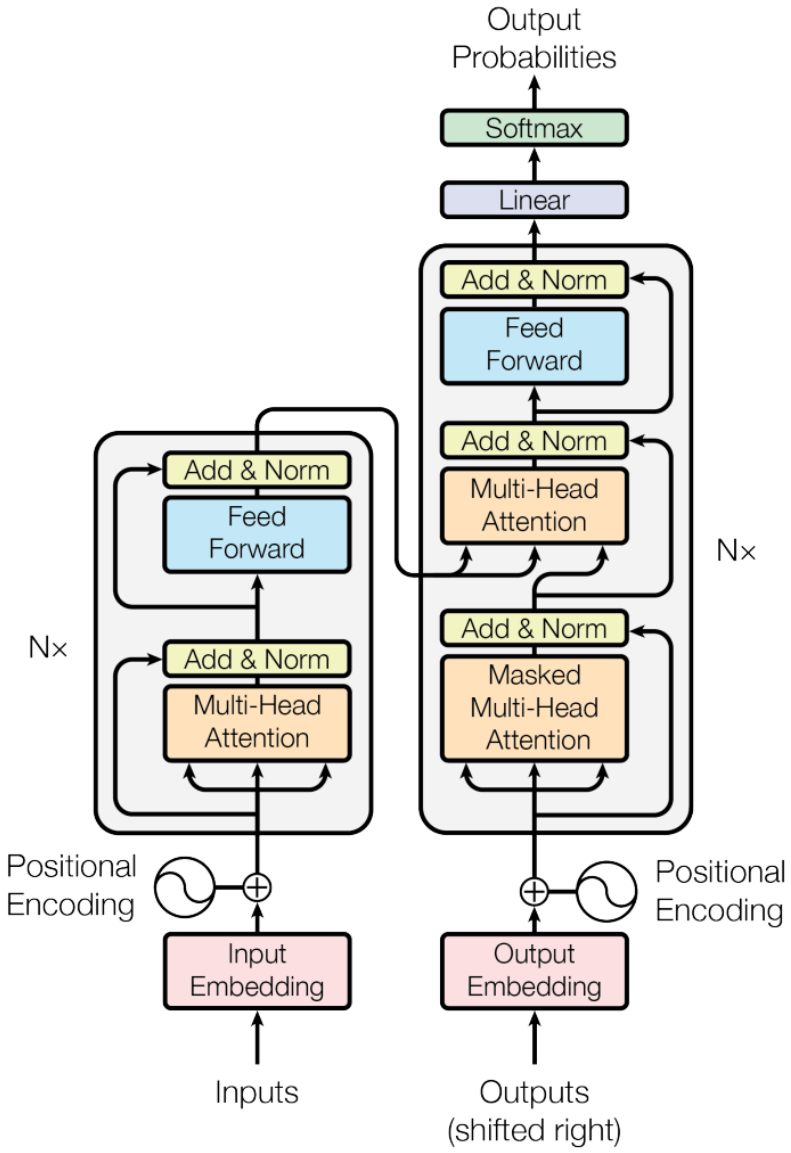

Figure 1: The Transformer - model architecture.

Transformer 是一种用于处理序列数据的深度学习模型，广泛应用于自然语言处理（NLP）任务。Transformer 的主要特点是通过自注意力机制（self-attention）实现了并行处理，大大提高了训练效率和性能。Transformer 结构由编码器（encoder）和解码器（decoder）两部分组成。

Transformer 编码器由多个相同的编码器层堆叠而成，每个编码器层包括以下子结构：

1. 自注意力机制（Self-Attention Mechanism）

计算输入序列中每个位置的表示向量与其他所有位置的表示向量之间的相关性。每个位置的表示向量通过加权求和得到新的表示，权重由相关性决定。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

2. 前馈神经网络（Feed-Forward Neural Network, FFN）

线性变换和激活函数的组合。

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

3.残差连接和层归一化（Residual Connection and Layer Normalization）**

每个子层后面都接有残差连接（即输入加上子层输出）和层归一化。

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

Transformer 解码器也由多个相同的解码器层（decoder layer）堆叠而成，每个解码器层包括以下子结构：

1. 自注意力机制

与编码器类似，但在计算注意力时加入了掩码（mask）机制，确保解码时每个位置只能关注当前及之前的位置，防止信息泄漏。

2. 编码器-解码器注意力机制

解码器的每个位置与编码器的所有位置进行注意力计算，从编码器获取相关信息。

3. 前馈神经网络

与编码器中的前馈神经网络相同。

4. 残差连接和层归一化

同编码器。

由于 Transformer 不像 RNN 那样具有顺序信息，位置编码用于注入序列的位置信息。位置编码可以是固定的，也可以是学习得到的。

固定位置编码公式：

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Transformer 的结构通过自注意力机制实现了全局依赖关系的捕捉，同时通过并行计算提高了效率。其编码器-解码器架构使其能够处理各种序列到序列的任务，如机器翻译、文本生成等。

Experimental Studies

M1:Seq2seq 实现流程

1. **数据预处理**:
    - 读取并合并多个文本文件，形成一个大语料库。
    - 进行文本清洗和预处理，比如去除非中文字符。
    - 使用 Tokenizer 对文本进行分词并创建词汇表。
    - 将文本转换为序列数据，并生成输入和输出对。
2. **创建数据集**:
    - 将序列数据转换为输入和目标对，以便训练。
    - 将数据集打乱并分批次处理。
3. **构建 Seq2Seq 模型**:
    - **编码器**: 输入层 -> 嵌入层 ->LSTM 层。LSTM 层的输出状态（state_h 和 state_c）作为上下文向量。
    - **解码器**: 输入层 -> 嵌入层 ->LSTM 层（初始状态为编码器的输出状态）-> Dense 层（生成词的概率分布）。
4. **训练模型**:
    - 编译模型，定义损失函数和优化器。
    - 训练模型，输入数据为输入序列和目标序列，目标数据为目标序列。
5. **生成文本**:
    - 给定起始字符串，将其转换为序列。
    - 使用编码器生成上下文向量。
    - 使用解码器逐步生成词，直到达到设定的生成长度或遇到结束标志。

M2：Transformer 模型实现流程

1. **数据预处理**:
    - 同 Seq2Seq 模型的数据预处理步骤。
    - 将文本转换为序列数据，并生成输入和输出对。
2. **创建数据集**:
    - 将序列数据转换为输入和目标对，以便训练。
    - 将数据集打乱并分批次处理。
3. **构建 Transformer 模型**:
    - **编码器**: 输入嵌入层 + 位置编码 + 多头自注意力机制 + 前馈神经网络。
    - **解码器**: 输入嵌入层 + 位置编码 + 多头自注意力机制 + 编码器-解码

器注意力机制 + 前馈神经网络 + 输出层。

4. **训练模型**:
    - 编译模型，定义损失函数和优化器。
    - 训练模型，输入数据为输入序列和目标序列，目标数据为目标序列。
5. **生成文本**:
    - 给定起始字符串，将其转换为序列。
    - 使用编码器生成上下文向量。
    - 使用解码器逐步生成词，直到达到设定的生成长度或遇到结束标志。

Conclusions

通过实验发现，Seq2Seq 模型适用于较短文本的生成，生成的文本连贯性较好。模型结构相对简单，训练和实现成本较低。对长序列文本的生成能力较弱，容易出现上下文不连贯的问题。编码器将整个输入序列编码为一个固定向量，可能会丢失一些重要的上下文信息。

Transformer 模型适用于长文本生成，能够捕捉远程依赖关系。并行处理能力强，训练速度较快。生成的文本上下文关联性较好，内容更丰富。模型结构复杂，训练和实现成本较高。