Rigid Formats Controlled Text Generation

Motivation

一般的文本生成任务没有严格的格式限制，像宋词，十四行诗，歌词等文本由严格的格式或韵律控制，而关于受控格式的文本生成还未被充分研究过。

刚性格式控制的文本生成

生成的文本必须：

- 完全遵守预定义的刚性格式
- 文本的安排必须满足韵律格式
- 必须保证句子的完整性

现有的如诗歌写作任务的工作只是将文本的格式和韵律作为一种数据的隐式特征在训练过程中学习。这种模型不具有泛化性，在五言绝句数据上训练的模型不能用于七言绝句的生成。

任务定义

输入一个刚性格式 $C \in \mathcal{C}$:

$$C = \{c_0c_1c_2c_3, c_0c_1c_2c_3c_4c_5.\}$$

其中 $\mathcal{C}$ 是所有可能的格式的集合。$\mathcal{C}$ 是一个可以扩展自定义格式的集合，因此 $|\mathcal{C}| \to \infty$. 其中的 c_i 表示一个占位符，代表需要生成一个实际的字的位置。C 中定义了字的个数和标点符号的位置。

输出的是一个符合格式 C 的自然语言语句 $Y \in \mathcal{Y}$, 例：

$$\begin{aligned} Y = \quad & \text{love is not love,} \\ & \text{bends with the remover to remove.} \end{aligned}$$

因为 $\mathcal{C}$ 是一个可以扩展的格式集，可以基于生成的结果 Y 重建一个新的格式 C', 重建的方式为 mask 掉部分的内容：

$$C' = \{c_0c_1c_2 \text{ love}, c_0c_1c_2c_3c_4 \text{ remove.}\}$$

即两个子句的最后一个词必须是 love 和 remove。

然后根据新生成的格式 C' 重新生成文本，此过程称为抛光(polishing)

该任务的目标即找到一个映射函数 G 来根据格式生成文本：

$$Y = G(C)$$

Methodology

基本框架是一个基于 Transformer 的自回归语言模型，输入可以是整个的宋词或十四行诗的 token 序列。作者设计了几种 embedding 来增强在格式，韵律和句子完整性上的准确率。同时改进了注意力机制来促使模型捕捉格式信息特征。

以一段莎士比亚十四行诗中的句子为例细说:

原始语句：$\text{love is not love, bends with the remover to remove.}$

输入语句：$<\text{bos}>\ \text{love is not love, }<\text{/s}>\text{bends with the remover to remove.}<\text{/s}>$

输出是输入的左移版本：

$$\begin{gathered}\text{love is not love, } < /s > \\ \text{bends with the remover to remove. } < /s >< eos >\end{gathered}$$

其中的 $< /s >$ 是句子分隔符。

新的嵌入

Format and Rhyme Symbols:

$$\begin{aligned} C = \{&c_0, c_0, c_0, c_2, c_1, < /s > \\ &c_0, c_0, c_0, c_0, c_0, c_2, c_1, < /s >, < eos >\} \end{aligned}$$

其中的 c_0 表示普通的 token， c_1 表示标点符号， c_2 表示韵脚位置。

Intra-Position Symbols:

$$\begin{aligned} P = \{&p_4, p_3, p_2, p_1, p_0, < /s > \\ &p_6, p_5, p_4, p_3, p_2, p_1, p_0, < /s >, < eos >\} \end{aligned}$$

其中 p_i 表示 token 在分句中的局部位置。作者特意采用递减的顺序进行编码，以便促使该 embedding 更好地捕获分句结束的信息特征(p_0 总是表示标点符号的位置，p_1 总是表示句子的最后一个词)

Segment Symbols:

$$\begin{aligned} S = \{&s_0, s_0, s_0, s_0, s_0, < /s > \\ &s_1, s_1, s_1, s_1, s_1, s_1, s_1, < /s >, < eos >\} \end{aligned}$$

其中的 s_i 表示该位置的 token 属于哪个分句。该 embedding 旨在增强不同位置不同子句之间的交互。

在训练时，上述的 embedding 和 token 的词嵌入以及全局位置嵌入相加作为序列的输入嵌入。

$$\mathbf{H}_t^0 = \mathbf{E}_{w_t} + \mathbf{E}_{c_t} + \mathbf{E}_{p_t} + \mathbf{E}_{s_t} + \mathbf{E}_{g_t}$$

其中的 $\mathbf{H}_t^0$ 表示第 t 个 token 的第 0 层表征，$\mathbf{E}_*$ 表示各种的 embedding，w_t 表示 token t 的词嵌入，c, p, s 表示上述的三种自定义 embedding，g 表示 token 的全局位置，与 Transformer 的相同。

修改注意力机制(的内容)

模型在自回归生成位置 t 的 token 时，还需要一些靠后的位置的信息来掌握全局的动态句子信息。如模型可能想知道当前的 token 是否应该是标点符号或者是否是最后一个 token。为了在预测的过程种引入这些信息，作者引入了另一个注意力参数 $\mathbf{F}^0$:

$$\mathbf{F}_t^0 = \mathbf{E}_{c_t} + \mathbf{E}_{p_t} + \mathbf{E}_{s_t}$$

作者将注意力计算过程分为两个部分：

1. **Masking Multi-Head Self-Attention**:

$$\begin{aligned}\mathbf{K}^0, \mathbf{V}^0 &= \mathbf{H}^0\mathbf{W}^K, \mathbf{H}^0\mathbf{W}^V \\ \mathbf{Q}^0 &= \mathbf{H}^0\mathbf{W}^Q \\ \mathbf{C}_t^1 &= \textsc{Ln}(\textsc{Slf-Att}(\mathbf{Q}_t^0, \mathbf{K}_{\le t}^0, \mathbf{V}_{\le t}^0) + \mathbf{H}_t^0) \\ \mathbf{C}_t^1 &= \textsc{Ln}(\textsc{Ffn}(\mathbf{C}_t^1) + \mathbf{C}_t^1)\end{aligned}$$

   该过程是标准的 **Musking Multi-Head Self-Attention** 过程，在进行自注意力运算时只 attend to $\le t$ 的文本表征($\mathbf{K}, \mathbf{V}$), 即将 $> t$ 的位置 mask 成 $-\infty$

2. **Global Multi-Head Attention**:

$$\begin{aligned}\mathbf{K}^1, \mathbf{V}^1 &= \mathbf{F}^0\mathbf{W}^K, \mathbf{F}^0\mathbf{W}^V \\ \mathbf{Q}^1 &= \mathbf{C}^1\mathbf{W}^Q \\ \mathbf{H}_t^1 &= \textsc{Ln}(\textsc{Global-Att}(\mathbf{Q}_t^1, \mathbf{K}^1, \mathbf{V}^1) + \mathbf{C}_t^1) \\ \mathbf{H}_t^1 &= \textsc{Ln}(\textsc{Ffn}(\mathbf{H}_t^1) + \mathbf{H}_t^1)\end{aligned}$$

   在该过程中，所有上下文的 $\mathbf{F}^0$ 都参与计算， $\mathbf{F}^0$ 中只有输入的格式，韵律和分段以及相对位置信息，并不包含词本身的信息。

上述两个注意力计算块重复 L 层就获得了最终表征 $\mathbf{H}^L$. L 层的计算中，H^l 每层都会更新，而 $\mathbf{F}^0$ 是全局固定的。

训练目标是最小化整个序列的负对数似然：

$$\mathcal{L}^{\text{nll}} = -\sum_{t=1}^{n} \log P(\mathbf{y}_t|\mathbf{y}_{<t})$$

一些与刚性文本格式(宋词，十四行诗)相关的评价指标等细节