屈折形式在问题生成模型中是冗余的

Xingwu Sun澳门大学
sunxing
wu01@gmail.com

唐宏印美团
tanghongyin
@meituan.com

徐成忠澳门大学
czxu@um.edu.mo

arXiv:2301.00397v1 [cs.CL] 1 Jan 2023

摘要

采用编码器-解码器框架的神经模型为问题生成（QG）提供了一种可行的解决方案。然而，通过分析模型词汇表，我们发现当前模型（包括基于RNN的模型和基于预训练的模型）中，屈折形式占比超过23%。因此，编码器会为这些屈折形式分别生成嵌入，导致训练数据和参数的浪费。更糟糕的是，在解码过程中，这些模型容易受到无关噪声的影响，而且计算成本高昂。本文提出一种通过融合词语变形来提升问题生成性能的方法。首先，我们从编码器的输入中识别词语的屈折形式，并将其替换为词根，使编码器能够更加关注重复出现的词根。其次，我们将问题生成建模为编码器-解码器框架中的以下操作组合：生成疑问词、从源序列复制词语，或生成一种词语变形类型。这样的扩展能够大幅缩小解码器中待预测词语的数量，同时减少噪声。我们将该方法应用于典型的基于RNN的模型和UNILM，得到性能提升后的版本。我们在SQuAD和MS MARCO数据集上进行了广泛实验。实验结果表明，与相应的基线模型相比，改进后的模型在BLEU、ROUGE-L和METEOR指标以及耗时方面均有显著优势。

1 引言

问题生成（QG）旨在根据给定文本生成自然语言问题。它可以为多种应用提供帮助：（1）通过为阅读理解材料生成问题，协助编制教育资料（Heilman 和 Smith，2010年；Du 等人，2017年）。（2）可用于自动整理问答数据集（Duan 等人，2017年）。（3）还可以通过主动提出有意义的问题来辅助对话系统。通常，问题生成包括两个子任务：（1）确定要提问的目标（例如，句子、短语或单词）；（2）生成问题的表层形式。在这项工作中，我们假设目标已给定，并着重研究问题的表层形式生成这一子任务。

Root Word	Inflected Forms		
commit	commits	committed	committing
big	bigger	biggest	-

图 1：问题生成模型词汇表中的两个词根及其对应的屈折形式。

近年来的神经问句生成模型，例如基于RNN的序列到序列模型和基于预训练的模型，在生成恰当问题方面取得了长足进展。特别是UNILM（Dong 等人，2019年）将预训练模型应用于问题生成，是目前性能最佳的最先进模型。然而，这些模型在解码时都面临较高的计算成本。我们仔细研究了现有的、总体上采用编码器—解码器架构的神经问句生成模型，发现现有模型仍存在两个问题。首先，在编码器中，词根及其屈折形式，例如图1所示的“commit”及其屈折形式“committed”和“committing”，可能都会出现在编码器的词汇表中。根据我们的统计，在广泛使用的问题生成数据集 SQuAD（Rajpurkar 等人，2016年）中，出现频率最高的前10000个词包含3718个屈折形式。此外，尽管 UNILM 应用 WordPiece 分词来减小词汇表规模，其词汇表中仍包含约23.18%（6722/28996）的屈折形式。因此，编码器将为这些词分别生成嵌入，而这

对于问题生成而言似乎有些冗余。更糟糕的是，它们还占用了编码器词汇表中的空间。因此，编码器会为这些屈折词生成独立的嵌入，导致训练数据和参数的浪费。其次，在解码器中，无论输入是什么，都会使用同一组庞大的词来生成问题。因此，这些模型容易受到无关噪声的影响。此外，在解码问题中的每个词时，都必须计算整个静态且相当庞大的词汇表中各词的概率分布，这会减慢解码过程。

为了降低计算成本并提升问题生成的性能，我们提出了一种名为词语变换方法的方法。该方法受到屈折语的特征以及我们日常阅读经验的启发。前者指的是，具有不同屈折形式的词语可能会分散问题生成的注意力。后者指的是，我们观察到，除了疑问词之外，生成问题中的大多数词语都可以从原始源序列中的词语复制而来，或经过简单变换得到。因此，在我们的方法中，我们将问题生成视为词形转换、词语复制和疑问词生成的组合。具体而言，在疑问词生成过程中，疑问词从有限的疑问词词汇表中生成。在变换类型生成过程中，屈折形式类型从一个较小的词语变形类型词汇表中生成。例如，可以生成“##ed”类型，并在最后一步中将其用于把动词转换为过去时。在词语复制过程中，词语从源序列中复制。此外，我们从编码器的输入中识别词语的屈折形式，并将其替换为词根形式，使编码器更加关注重复出现的词根，并在训练中吸收更多不同的词语。我们已将该方法应用于一个典型的基于RNN的模型和一个预训练的基于Transformer的模型UNILM，并获得了相应的改进版本。

本文的主要贡献总结如下。

- 我们发现，屈折词占用了词汇表中不必要的空间，并定义了一系列词语变换类型，以便将词语转换为其屈折形式。
- 我们只在编码器的词汇表中保留词根，以充分利用训练数据和有限的词汇表空间。这一做法也适用于许多其他自然语言处理模型。
- 我们提出通过疑问词生成、变换类型生成和词语复制来简化问题生成的解码过程，从而避免从大型词汇表中逐个生成词语，并提升性能。
- 为评估所提方法的有效性和效率，我们在两个大型数据集（即 SQuAD 和 MS MARCO 数据集）上进行了广泛实验，并将基线模型与相应的改进版本进行了比较。实验结果表明，改进版本的表现显著优于基线模型。

2 相关工作

现有的问题生成方法大致可分为两类：基于规则的方法和基于神经网络的方法。基于规则的方法依赖人工设计的规则，而基于神经网络的方法由数据驱动，可通过端到端的方式进行训练。

早期的问题生成方法大多是基于规则的方法（Heilman 和 Smith，2009年、2010年；Chali 和 Hasan，2015年），它们利用规则或模板生成问题。基于规则的方法采用人工制定的规则，将陈述句转换为疑问句，通常依赖句法（Mitkov 和 Ha，2003；Ali 等人，2010年；Heilman，2011年）或语义信息（Chen，2009年）；而基于模板的方法则使用人工创建的模板生成问题，这些模板预先设有占位符，用于填入源词序列中的词语（Cai 等人，2006年；Lindberg 等人，2013；Song 和 Zhao，2016年）。这些基于规则的方法的主要局限包括：（1）它们严重依赖人工创建的规则和模板，因此成本高昂；（2）这些规则或模板缺乏多样性；（3）它们能够处理的目标有限。

为了解决基于规则的方法的局限性，人们将采用编码器-解码器框架的基于神经网络的方法应用于问题生成任务。这些方法不依赖手工编写的规则，而是由数据驱动，并以端到端的方式进行训练。大规模机器阅读理解数据集的发布，例如 SQuAD、MS MARCO（Nguyen 等人，2016年）、Hotpot QA（Yang 等人，2018）和 DROP（Dua 等人，2019年），进一步推动了神经问句生成模型的发

词性	变换类型	转换类型描述
verb	##ing	将动词转换为现在分词
	##vs	将动词转换为第三人称单数现在时
	##ed	将动词转换为过去时
	##edp	将动词转换为过去分词
noun	##ns	将名词转换为复数形式
形容词	##jer	将形容词转换为比较级
	##jest	将形容词转换为最高级
副词	##ver	将副词转换为比较级
	##背心	将副词变为最高级形式

表 1：词语变换类型及其对应描述。

展。总体而言，这些数据集包含大规模人工标注三元组，即问题、答案和上下文，可用作问题生成模型的训练数据。

关于基于RNN的模型，Du 等人(2017年)和Yuan 等人 (2017年) 都采用带有注意力机制的序列到序列模型，为 SQuAD 数据集中的文本生成问题。Zhou 等人 (2017年) 通过丰富的特征（即答案位置和词汇特征）增强基于RNN的模型，以生成答案聚焦型问题；此外，他们还引入复制机制，使模型能够从上下文中复制词语。Duan 等人 (2017年) 提出将模板与序列到序列模型结合起来：他们从问答社区中挖掘问题模式，并使用序列到序列模型为给定文本生成问题模式。Tang 等人 (2017年) 将问答和问题生成建模为双重任务，从而同时帮助生成更好的问题，并构建更好的问答模型。

此外，基于指针生成网络（See 等人，2017年），Sun 等人 （2018）提出了一种以答案为中心的位置感知模型，以有效利用答案编码和位置特征。Zhao 等人 （2018）主要通过使用门控自注意力和 Maxout 指针网络，着重融入段落级上下文。Nema 等人 （2019年）提出了一种精炼网络，用以模拟人类生成问题的过程。此外，也有一些工作研究从知识库生成问题（Serban 等人，2016年；Indurthi 等人，2017年；Liu 等人，2019年）。

近来，预训练模型也被应用于问题生成。例如，Dong 等人 (2019年)提出了 UNILM，在问题生成方面取得了长足进展。目前，UNILM 是性能最佳的问题生成模型。

与以往工作不同，我们的方法并未提供问题生成模型，而是为编码器-解码器框架提供了两种改进方案，以提升性能。我们的方法可应用于大多数现有模型，包括基于循环神经网络的序列到序列模型以及 UNILM。

3 我们的方法

首先，我们将问题生成（QG）任务正式定义如下。给定长度为 T_x 的文本（x_1，x_2，.....，x_{T_x}），以及词汇特征，即命名实体（NE）和词性（POS）。该文本中的答案位置范围为 l 至 r。目标是生成一个问题，使其尽可能接近参考问题（y_1，y_2，.....，y_{T_y}）。

在接下来的小节中，我们将详细介绍所提出的方法，以解决前文讨论的问题。首先，我们定义一系列词形转换类型，这构成了我们方法的基础。其次，我们选择指针生成网络和 UNILM 作为基线模型，并分别通过阐述改进版本中编码器-解码器的工作流程，说明如何应用我们的方法。

3.1 词语变换类型

作为我们方法的基础，我们根据词语的词性定义变换类型，如表 1 所示。对于动词，我们定义了四种变换类型，即“##ing”、“##vs”、“##ed”和“##edp”。对于名词，我们定义了一种变换类型“##ns”，表示将名词转换为复数形式。对于形容词或副词，我们分别将其转换为比较级的变换定义为“##jer”和“##ver”。同样，我们将其转换为最高级的变换定义为“##jest”和“##vest”。注意

我们使用 Pattern 模块[1] 在 Python 中执行这些变换。对于不规则词，我们手动收集了一个查找表，例如，“went”可以转换为 go 和“##ed”。

回顾第 1 节中提到的两个问题。对于编码器中的第一个问题，通过一系列变换类型，我们在编码器词汇表中只保留词根和变换类型。因此，可以大幅节省其他词汇所占的空间，并充分利用训练数据。对于第二个问题，除了疑问词之外，我们直接复制源序列中的词，或对其进行变换，以生成问题。在变换类型生成中，必须引入变换类型词汇表，接下来的小节将对此进行详细说明。

在模型编码之前，我们首先获取原始输入序列中每个词的词根形式。例如，在图2中，单词“succeeded”被转换为其词根形式“succeed”，相应的变换类型为“##ed”。经过变换后，输入序列被转换为一个长度为$(x'_1, x'_2, \ldots\ldots, x'_{T'_x})$的序列，长度为 T'_x，答案位置范围从 l' 到 r'。同样，参考问题被转换为$(y'_1, y'_2, \ldots\ldots, y'_{T'_y})$，长度为 T'_y。

3.2 改进指针生成网络

3.2.1 编码器

改进后的基线是基于注意力的指针生成网络（参见 See 等人，2017年），并结合了 Zhou 等人 (2017年)提出的多种丰富特征。这些特征包括编码器嵌入层中的命名实体（NE）、词性（POS）和答案位置。

如图2所示，编码器是一个双向 GRU，其输入是词嵌入、答案位置嵌入和词汇特征（命名实体、词性）的联合嵌入，形式为$(w_1, w_2, \ldots, w_{T'_x})$，其中 $w_i \in \mathbb{R}^{d_w+d_a+d_n+d_p}$，$T'_x$ 是输入长度，w_i 是通过拼接所有特征嵌入得到的 x'_i，而 d_w、d_a、d_n、d_p 分别是词嵌入、答案位置嵌入、命名实体嵌入和词性嵌入的维度。它生成一系列 d_h维隐藏状态$(h_1, h_2, \ldots, h_{T'_x})$，其中每个隐藏状态都是以下各项之和：向和反向 GRU 表示：

$$\begin{aligned} h_i &= \overleftarrow{h}_i + \overrightarrow{h}_i, \\ \overleftarrow{h}_i &= \texttt{GRU}(w_i, \overleftarrow{h}_{i+1}), \\ \overrightarrow{h}_i &= \texttt{GRU}(w_i, \overrightarrow{h}_{i-1}) \end{aligned} \tag{1}$$

其中 $\overleftarrow{h}_i$、$\overrightarrow{h}_i$ 都是 d_h维向量。

3.2.2 解码器

解码器经过修改，以支持三种操作：词语复制、变换类型生成和疑问词生成。(1) 在词语复制中，解码器从源序列生成词语。(2) 在变换类型生成中，解码器从有限的变换类型词汇表中生成内容，该词汇表仅包含表1中的九种类型。(3) 在疑问词生成中，解码器从受限的疑问词词汇表中生成疑问词。

词语复制 解码器是一个单向GRU，以所有编码后的隐藏状态为条件输入。在解码步骤 t，解码器读取一个词嵌入 w_t、先前的注意力上下文向量 c_{t-1} 以及先前的隐藏状态 s_{t-1}，以更新当前隐藏状态$s_t \in \mathbb{R}^{d_h}$：

$$s_t = \texttt{GRU}([w_t; c_{t-1}], s_{t-1}) \tag{2}$$

上下文向量 c_t 通过注意力机制生成（Bahdanau等人，2014年）。在时间步 t，上下文向量 c_t 计算如下：

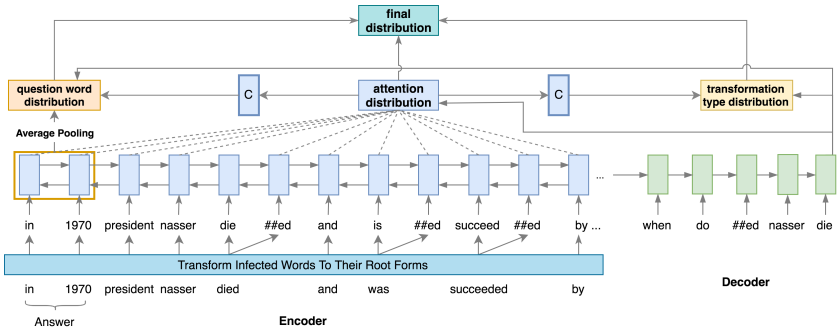
$$\begin{aligned} c_t &= \sum_{i=1}^{T'_x} \alpha_{ti} h_i \\ \alpha_{ti} &= \texttt{Softmax}(e_{ti}) \\ e_{ti} &= v^T \tanh(W_h^T h_i + W_s^T s_t + b) \end{aligned} \tag{3}$$

其中，α 是注意力分布，W_h，W_s，b和 v 均为可训练参数。

注意力分布可以看作编码器隐藏状态与解码器隐藏状态之间的语义匹配。它表示解码过程中解码器对编码器不同隐藏状态的关注程度。因此，注意力分布可以视为词语复制的概率分布。

$$P_{copy} = \alpha_t \tag{4}$$

[1] https://github.com/clips/pattern



Fig图 2：基于指针生成网络的改进版本。“C”表示上下文向量。

变换类型生成 在编码之前，我们已经将每个词转换为其词根形式，即词根。在解码时，除了复制词语，我们还应赋予模型将某些词转换为适当形式的能力，例如，图 2中的“do”由疑问词词汇表生成，我们需要将其转换为“did”。为执行这一转换，我们引入变换类型生成，并为词根生成变换类型，例如，生成“##ed”以将“do”转换为“did”。

我们通过以下函数得到变换类型的概率分布。

$$P_{trans} = \texttt{Softmax}\left(g_1(s_t, c_t)\right) \tag{5}$$

其中，$g_1(\cdot)$ 是一个具有 Maxout 内部激活函数的两层前馈神经网络。$P_{trans} \in \mathbb{R}^{|V_{trans}|}$ 表示变换类型的概率分布。

疑问词生成 我们发现，疑问词的生成主要由答案及其周围的词决定。例如，在图 2 中，答案及其上下文“in 1970”已经包含生成疑问词“when”所需的关键信息，这表明答案及其周围的词有助于疑问词生成。我们还发现，疑问词的总数有限。因此，我们引入一个专门的疑问词词汇表，以直接、明确地对疑问词生成进行建模。请注意，该疑问词词汇表包含训练集参考问题中最常见的 1000 个词，这意味着其中也包含其他与问题相关的词。

如图 2所示，在疑问词生成中，模型根据受限的疑问词词汇表生成疑问词。这一操作会产生一个疑问词分布

基于答案嵌入 v_{answer}、解码器状态 s_t 和上下文向量 c_t：

$$P_{quest} = \texttt{Softmax}\left(g_2(v_{answer}, s_t, c_t)\right) \tag{6}$$

其中，$g_2(\cdot)$ 是一个两层前馈神经网络，P_{quest} 是一个 $|V_{quest}|$维的概率分布，$|V_{quest}|$ 是疑问词词汇表的大小。我们采用平均池化函数计算答案嵌入。

$$v_{answer} = \frac{\sum_{t=l'}^{r'} h_t}{r' - l' + 1} \tag{7}$$

三动作组合 为了控制不同动作之间的平衡，我们引入一个三维切换概率，作为一个三路软开关：

$$p_{quest}, p_{copy}, p_{trans} = \texttt{Softmax}(f(c_t, s_t, w_t)) \tag{8}$$

其中，$f(\cdot)$ 是一个单层前馈网络。我们通过对三个动作的概率分布进行加权求和，计算最终的概率分布：

$$\begin{aligned} P(w) = {} & p_{copy} P_{copy}(w) + p_{trans} P_{trans}(w) \\ & + p_{quest} P_{quest}(w) \end{aligned} \tag{9}$$

3.3 改造 UNILM

UNILM 是一种预训练语言模型，其结构与 BERT（Devlin 等人，2019年）相同，但可用于自然语言生成任务。该模型首先在大规模语料库上进行预训练，从而能够学习灵活的语言表示。然后，使用下游任务的数据集，通过序列到序列语言模型目标对模型进行微调。在 UNILM 中，编码器和解码器之间没有明确的区分。相反，源序列和目标序列会一同输入模型。序列到序列生成通过一种特殊的自注意力掩码 M 实现。

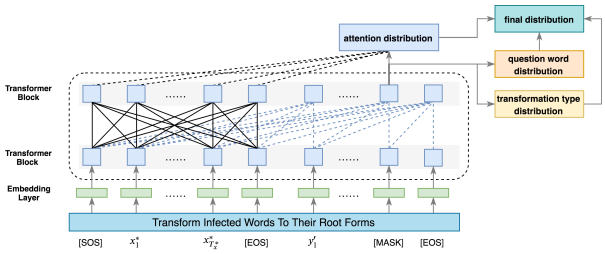


图 3：基于 UNILM 的改进版模型架构。注意，应该有 L 个 Transformer 层。为简单起见，图中只保留了其中两层。

遵循 Dong 等人（2019）的做法，我们将答案片段（$x'_{l'}, ..., x'_{r'}$）和文本片段（$x'_1, x'_2, ..., x'_{T'_x}$）拼接起来，形成新的源片段 $X = (x^*_1, ..., x^*_{T^*_x}) = (x'_1, x'_2, ..., x'_{T'_x}, [\text{EOS}], x'_{l'}, ..., x'_{r'})$。接下来，我们将新的源片段与目标片段 $Y = (y'_1, y'_2, ..., y'_{T'_y})$ 拼接后作为输入，即 $[\text{SOS}]X[\text{EOS}]Y[\text{EOS}]$。然后将输入送入模型。在训练过程中，我们以 0.8 的概率将词元替换为 [MASK] in Y。

对于每个输入词元，其嵌入是词元嵌入、位置嵌入和片段嵌入之和。如图 3所示，我们得到词元嵌入 $w = (w_1, w_2, ..., w_{T^*_x+T'_y+3})$，随后将其输入 L层 Transformer，以获得深层表示 $H^l = h^l_i$，$i \in [1, T^*_x + T'_y + 3]$。

$$h^l_i = \texttt{Transformer}(h^{l-1}_i), l = 1, 2, 3, ..., L \tag{10}$$

其中，H^0 由 w 初始化。第 l 个 Transformer 的自注意力头的输出计算如下：

$$Q_l = H^{l-1}W^Q,\ K_l = H^{l-1}W^K,\ V_l = H^{l-1}W^V \tag{11}$$

$$A_l = \texttt{Softmax}(\frac{Q_l K_l^T}{\sqrt{d_k}} + M)V_l \tag{12}$$

其中，$M \in \mathbb{R}^{(T^*_x+T'_y+3)\times(T^*_x+T'_y+3)}$。$M_{ij} = 0$表示允许关注，而 $M_{ij} = -\infty$表示不允许关注。我们可以发现，特殊的自注意力掩码 M 允许待生成的词元关注所有源词元和此前生成的词元。这种机制使模型能够以序列到序列的方式生成文本。

为了采用词语变换方法，我们直接根据当前步骤的隐藏状态计算与这三种操作相关的分布。具体如下。

词语复制 通过一个额外的注意力模块，使用点积计算注意力分数，从而获得源词元上的注意力分布：

$$P_{copy} = \texttt{Softmax}(g_3(h_t)) \tag{13}$$

$$g_3(h_t) = [g_3(h_t)_i]^{T'_x}_{i=1} = [h_t \cdot h_i]^{T'_x}_{i=1} \tag{14}$$

其中，h_t 是 Transformer 的最后一层输出的第 t 个词元的隐藏状态。$g_3(h_t)$ 的输出是 h_t 与所有输入词元的隐藏状态之间的点积。

变换类型生成 通过计算当前隐藏状态与变换类型嵌入之间的点积，可以获得变换类型的分布。

$$P_{trans} = \texttt{Softmax}(g_4(h_t)) \tag{15}$$

$$g_4(h_t) = [g_4(h_t)_i]^{|V_{trans}|}_{i=1} = \left[h_t \cdot e^T_i\right]^{|V_{trans}|}_{i=1} \tag{16}$$

其中，e^T 是变换类型的嵌入。$g_4(h_t)$ 是 h_t 与所有变换类型嵌入之间的点积。P_{trans} 表示变换类型的概率分布。

疑问词生成 与变换类型分布的计算类似，疑问词的分布是通过计算当前隐藏状态与疑问词嵌入之间的点积获得的。

$$P_{quest} = \texttt{Softmax}(g_5(h_t)) \tag{17}$$

$$g_5(h_t) = [g_5(h_t)_i]^{|V_{quest}|}_{i=1} = \left[h_t \cdot e^Q_i\right]^{|V_{quest}|}_{i=1} \tag{18}$$

其中，e^Q 是疑问词的嵌入。$g_5(h_t)$ 是 h_t 与所有疑问词的嵌入之间的点积 e^Q。

三动作组合 三维切换概率由当前隐藏状态 h_t 得出。

$$p_{quest}, p_{copy}, p_{trans} = \texttt{Softmax}(f_1(h_t)) \tag{19}$$

其中 $f_1(\cdot)$ 是一个单层前馈网络。最终的概率分布按照方程9中相同的方式计算。

模型	BLEU1	BLEU2	BLEU3	BLEU4	ROUGE-L	METEOR
Pointer	40.49	26.11	18.94	14.34	42.15	18.71
Pointer + WT	**45.08**	**29.56**	**21.54**	**16.41**	**45.40**	**20.61**
UNILM	49.82	34.48	26.03	20.39	49.02	23.52
UNILM +WT	**51.56**	**35.78**	**27.06**	**21.24**	**50.46**	**23.93**

(a)

模型	BLEU1	BLEU2	BLEU3	BLEU4	ROUGE-L	METEOR
Pointer	44.45	31.85	23.32	17.90	46.07	20.02
Pointer + WT	**56.14**	**39.36**	**29.04**	**22.10**	**59.29**	**26.40**
UNILM	60.57	43.14	32.38	25.01	60.58	29.31
UNILM +WT	**62.65**	**45.33**	**34.31**	**26.55**	**62.85**	**29.72**

(b) 表2：基线模型及其改进版本在 SQuAD (a) 和 MARCO (b) 上的主要实验结果。“WT”表示所提出的词语变换方法。

4 实验

4.1 实验设置

数据集我们在 SQuAD 和 MARCO 数据集上进行了实验。由于这些数据集的测试集未公开，我们遵循Zhou 等人(2017年）的做法，将开发集随机拆分为两部分，并将它们分别用作问句生成任务的开发集和测试集。在 SQuAD 中，我们的训练集、开发集和测试集分别包含86,635,8,965 和 8,964 个问答对。我们直接使用了提取的特征[2]，这些特征由Zhou 等人(2017年）共享。在 MARCO 中，我们的训练集、开发集和测试集分别包含74,097,4,539 和4,539 个问答对。我们使用 Stanford CoreNLP[3] 提取词汇特征。

实现细节 我们将输入序列的截断长度设为 128 个词。编码器词汇表包含每个训练集内最常见的 30，000 个词。解码器词汇表包含两个子词汇表。其中一个是问题词词汇表，包含训练集参考问题中最常见的 1000 个词；另一个是变换类型词汇表，包含第 3 节所述的 9 种变换类型。对于基于RNN的模型，我们使用预训练的 Glove 词向量[4] （维度为 300）初始化词嵌入，并在训练阶段进一步微调。编码器嵌入层中的答案位置特征和词汇特征表示被随机初始化为 32 维向量，并在训练阶段进行训练。

[2]https://res.qyzhou.me/redistribute. zip[3] https://nlp.stanford.edu/software/[4] http://nlp.stanford.edu/data

编码器和解码器的隐藏状态维度均为 512。我们仅在编码器中使用 Dropout，Dropout 率为0.20。答案嵌入的维度为 512。我们采用优化算法 Adam（Kingma 和 Ba，2014年），学习率为0.002 ，并将批次大小设为 32。对于 UNILM，我们采用 UNILM 的基础版本，并使用 Dong 等人 (2019年）详述的推荐参数。训练完成后，我们选择开发集上表现最好的模型进行测试。

评估指标 我们使用 n 元语法相似度指标来评估我们的方法，即 BLEU (Pap- ineni 等人，2009年)、ROUGE-L (Lin，2004) 和 METEOR (Lavie 和 Denkowski，2009年)。

对比方法在实验中，我们选取以下四种方法进行比较，其中 Pointer 和 UNILM 是基线模型，另外两种是相应的改进版本。

- **指针生成网络（Pointer）**这是一个典型的基于循环神经网络的序列到序列模型，采用了复制机制（See 等人，2017年）。为进行公平比较，我们将词汇特征添加到嵌入层中，与Zhou 等人（2017年）的做法相同。
- **指针生成网络加词语变换方法（Pointer + WT）**我们通过所提出的词语变换方法增强 Pointer 模型。
- **UNILM（Dong 等人，2019年）**这是一种预训练语言模型，可应用于自然语言生成任务。
- **UNILM + WT** 我们将 UNILM 模型与所提出的词语变换方法相结合。

Pointer 和 Pointer +WT 使用

TensorFlow 实现，而 UNILM 和 UNILM + WT 则由 PyTorch 实现。

4.2 性能评估

表2 展示了主要结果，我们得出了以下观察：

- Pointer +WT 模型的表现优于原始 Pointer 模型，这表明通过词语变换生成问题是有效的。
- UNILM +WT 模型仍然能够显著优于强大的 UNILM，这表明将我们的方法应用于预训练语言模型对于问题生成任务是有效的。
- 与 SQuAD 相比，我们的方法在 MARCO 上取得了更大的改进。这可能是因为 MARCO 提取自搜索引擎，从而对源文本中的词语复制以及词语变换生成引入了偏差。

除了生成质量之外，我们还比较了基线模型与相应改进版本在解码效率方面的表现。我们在束宽为 12 的情况下，记录给定测试文本时生成一个问题词语所需的时间，并计算平均值。需要注意的是，在效率比较中，我们将词根及其对应变换类型的解码视为一个词语。在单个 Tesla V100 图形处理器环境下进行比较。对于基于RNN的模型，Pointer + WT 模型的效率更高；与原始模型相比，它可以节省超过 **39%** 的解码时间（从 0.0081 秒降至 0.0049 秒）。对于预训练模型，UNILM +WT 模型可以节省**13%** 的解码时间（从 0.0144 秒降至 0.0125 秒）。我们还发现，UNILM 在生成质量方面优于 Pointer，但牺牲了效率。具体而言，它的速度几乎比基于RNN的模型慢一倍。综合有效性和效率两方面的比较，我们可以得出结论：这种词语变换方法能够提升问题生成的质量，同时加快解码速度。

4.3 人工评估

我们进一步进行人工评估，以分析 Pointer 模型和 Pointer +WT 模型的生成质量。我们从 SQuAD 数据集中随机抽取 200 个案例，并请三位

文本： 在 3 月的布尔堡战役中，法国军队摧毁了堡垒和大量补给品，包括 **45,000** 磅火药。**答案：45,000 参考问题：** 袭击中摧毁了多少火药？**Pointer：** 多少火药**的** 被**杀害**于布尔堡 3 月战役中？
Pointer + WT： 布尔堡 3 月战役中摧毁了多少火药？

表3：一个 Pointer 模型生成了无关词语的案例，即**“杀死了”**和**“的”**，而采用词形转换方法的模型则能更准确地生成问题。

我们邀请了语言学专业的标注员来比较生成质量。向标注员展示两个问题，一个由 Pointer 模型生成，另一个由 Pointer + WT 模型生成。请他们从三个因素判断哪个更好，即流畅度、完整性和可回答性。在本次评估中，这三个因素同等重要。最终标签由多数投票决定。根据指标值，我们发现 Pointer + WT 模型优于 Pointer 模型。具体而言，在 **73/200** 个案例中，Pointer + WT 模型的表现优于 Pointer 模型。在 91/200 个案例中，两个模型的表现几乎相同。在 36/200 个案例中，Pointer + WT 模型的表现不如基线。

表3 给出了一个案例，用于展示我们方法的有效性。如表3所示，Pointer 模型生成了无关词语，即**“杀死”**和**“的”**。这可能是因为 Pointer 模型从一个庞大且嘈杂的词汇表中生成内容。然而，thePointer + WT模型能够更加正确地生成问题，这无疑要归功于我们的方法。

5 结论

本文发现，现有神经问句生成模型存在两个主要问题。为解决这两个问题，我们提出了这种改进问题生成的方法，并将其应用于两种典型的序列到序列模型，即 Pointer 生成网络和 UNILM。我们还使用 SQuAD 和 MARCO 数据集进行了大量实验。实验结果表明，改进后的模型能够显著提升问题生成的质量，并加快解码速度。

参考文献

Husam Ali、Yllias Chali 和 Sadid A Hasan。2010年。从句子自动生成问题。载于*QG2010*：第三届问题生成研讨会论文集，第58–67页。

Dzmitry Bahdanau、KyunghyunCho 和 Yoshua Ben- gio。2014年。通过联合学习对齐和翻译实现神经机器翻译。*arXiv* 预印本 *arXiv:1409.0473*。

Zhiqiang Cai、Vasile Rus、Hyun-Jeong Joyce Kim、Suresh C Susarla、Pavan Karnam 和 Arthur C Graesser。2006年。Nlgml：一种用于问题生成的标记语言。发表于*E-Learn*：企业、政府、医疗保健和高等教育领域的世界电子学习会议，第2747–2752页。教育计算促进协会（AACE）。

Yllias Chali 和 Sadid A Hasan。2015年。迈向从主题生成问题。《计算语言学》，41(1)：1–20。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向 Transformer 预训练。载于计算语言学协会北美分会 *2019* 年会议论文集：人类语言技术，第 *1* 卷（长篇与短篇论文），第 4171–4186 页，明尼阿波利斯，明尼苏达州。计算语言学协会。Li Dong、Nan Yang、Wenhui Wang、Furu Wei、Xiaodong Liu、Yu Wang、Jianfeng Gao、Ming Zhou 和 Hsiao-Wuen Hon。2019年。面向自然语言理解与生成的统一语言模型预训练。载于神经信息处理系统进展，第 13063–13075 页。Xinya Du、Junru Shao 和 Claire Cardie。2017年。学习提问：面向阅读理解的神经问题生成。*arXiv* 预印本 *arXiv:1705.00106*。Dheeru Dua 、 Yizhong Wang、Pradeep Dasigi、Gabriel Stanovsky、Sameer Singh 和 Matt Gardner。2019年。DROP：一项要求对段落进行离散推理的阅读理解基准测试。*arXiv* 预印本 *arXiv:1903.00161*。Nan Duan、Duyu Tang、Peng Chen 和 Ming Zhou。2017年。面向问答的问题生成。载于 *2017* 年自然语言处理实证方法会议论文集，第 866–874 页。Michael Heilman。2011年。基于文本的自动事实性问题生成。博士论文，卡内基梅隆大学。Michael Heilman 和 Noah A Smith。2009年。通过过度生成变换和排序生成问题。技术报告，卡内基梅隆大学匹兹堡分校语言技术研究所。

Michael Heilman 和 Noah A Smith。2010年。好问题！用于问题生成的统计排序。载于人类语言技术：计算语言学协会北美分会 *2010* 年年会，第 609–617 页。计算语言学协会。

Sathish Reddy Indurthi、Dinesh Raghu、Mitesh M Khapra 和 Sachindra Joshi。2017年。使用基于循环神经网络的问题生成模型，从知识图谱生成自然语言问答对。发表于第*15*届欧洲计算语言学协会会议论文集：第*1*卷，长篇论文，第376–385页。

Diederik P Kingma 和 Jimmy Ba。2014年。Adam：一种随机优化方法。*arXiv* 预印本 *arXiv:1412.6980*。

Alon Lavie 和 Michael J Denkowski。2009年。用于机器翻译自动评估的 METEOR 指标。机器翻译，23(2-3)：105–115。

Chin-Yew Lin。2004年。ROUGE：一个用于自动评估摘要的软件包。载于文本摘要的拓展，第 74–81 页。

David Lindberg、Fred Popowich、John Nesbit 和 Phil Winne。2013年。生成自然语言问题以支持在线学习。载于第*14*届欧洲自然语言生成研讨会论文集，第105–114页。Cao Liu、Kang Liu、Shizhu He、Zaiqing Nie 和 Jun Zhao。2019年。通过整合多样化上下文和答案感知损失，为知识库生成问题。*arXiv*预印本 *arXiv:1910.13108*。Ruslan Mitkov 和 Le An Ha。2003年。计算机辅助生成选择题测试。载于自然语言处理教育应用构建 *HLT-NAACL 03*研讨会论文集第*2*卷，第17–22页。计算语言学协会。Preksha Nema、Akash Kumar Mohankumar、Mitesh M Khapra、Balaji Vasan Srinivasan 和 Balaraman Ravindran。2019年。再问一次：用于自动问题生成的精炼网络。*arXiv*预印本 *arXiv:1909.05355*。Tri Nguyen、Mir Rosenberg、Xia Song、Jianfeng Gao、Saurabh Tiwary、Rangan Majumder 和 Li Deng。2016年。MS MARCO：一个由人工生成的机器阅读理解数据集。*arXiv*预印本 *arXiv:1611.09268*。Kishore Papineni、Salim Roukos、Todd Ward 和 Wei-Jing Zhu。2002年。BLEU：一种机器翻译自动评估方法。载于计算语言学协会第*40*届年会论文集，第311–318页。计算语言学协会。

Pranav Rajpurkar、Jian Zhang、KonstantinLopyrev 和 Percy Liang。2016年。SQuAD：用于文本机器理解的 100,000+ 个问题。*arXiv*预印本 *arXiv:1606.05250*。

Abigail See、Peter J Liu 和 Christopher D Man- ning。2017年。切中要点：使用指针生成网络进行摘要。*arXiv* 预印本 *arXiv:1704.04368*。

Iulian Vlad Serban、Alberto Garc´ıa-Dur´an、Caglar Gulcehre、Sungjin Ahn、Sarath Chandar、Aaron Courville 和 Yoshua Bengio。2016年。使用循环神经网络生成事实型问题：3000万事实型问答语料库。*arXiv* 预印本 *arXiv:1603.06807*。

Linfeng Song 和 Lin Zhao。2016年。基于知识库的领域专用问题生成。*CoRR*。

Xingwu Sun、Jing Liu、Yajuan Lyu、Wei He、Yanjun Ma 和 Shi Wang。2018年。以答案为中心且具备位置感知能力的神经问题生成。发表于*2018*年自然语言处理实证方法会议论文集，第3930–3939页。

Duyu Tang、NanDuan、Tao Qin 和 MingZhou。2017年。问答和问题生成作为双重任务。*arXiv* 预印本 *arXiv:1706.02027*。

Zhilin Yang、Peng Qi、Saizheng Zhang、Yoshua Ben- gio、William W Cohen、Ruslan Salakhutdinov 和 Christopher D Manning。2018年。HotpotQA：用于多样化、可解释的多跳问答的数据集。*arXiv* 预印本 *arXiv:1809.09600*。

Xingdi Yuan、Tong Wang、Caglar Gulcehre、Alessan- dro Sordoni、Philip Bachman、Sandeep Subrama- nian、Saizheng Zhang 和 Adam Trischler。2017年。通过文本到文本神经问题生成实现机器理解。*arXiv* 预印本 *arXiv:1705.02012*。

Yao Zhao、Xiaochuan Ni、Yuanyuan Ding 和 Qifa Ke。2018年。使用最大输出指针和门控自注意力网络进行段落级神经问题生成。在*2018*年自然语言处理实证方法会议论文集中，第3901–3910页。

Qingyu Zhou、Nan Yang、Furu Wei、Chuanqi Tan、Hangbo Bao 和 Ming Zhou。2017年。《从文本生成神经问句：一项初步研究》。载于全国 *CCF*自然语言处理与中文计算会议，第 662–671 页。施普林格。