马尔可夫决策过程和先验控制向量在弱约束自然语言生成中的应用

刘 奇 马 娆 俞 凯

(上海交通大学人工智能研究院人工智能教育部重点实验室 上海 200240)
(上海交通大学计算机科学与工程系跨媒体语言智能实验室 上海 200240)
(新华社媒体融合生产技术与系统国家重点实验室第一联合创新中心 北京 100803)

摘 要 自然语言生成是目前非常重要且具有挑战性的一类人工智能任务.长短时记忆(Long Short-Term Memory,LSTM)语言模型是目前最为主流的自然语言生成模型.但是,LSTM 语言模型的训练准则是词语级别的交叉熵,这会导致暴露偏差问题.此外,一般自然语言生成任务的评测指标是序列级别的 BLEU 分数或者词错误率,这与训练使用的交叉熵准则也不匹配.在本文中,我们使用马尔可夫决策过程重定义了自然语言生成问题,并通过从训练数据中提取的先验控制向量来指导生成过程.先验控制向量可以视作是对序列空间的一种先验划分的抽象,通过在自然语言生成中引入先验控制向量,我们可以更好的约束自然语言生成的空间.再通过马尔可夫决策过程的定义,我们可以使用策略梯度算法来直接使用测试使用的 BLEU 分数来代替交叉熵训练 LSTM 网络.在多个数据集上的实验显示本文提出的方法相比于普通使用 LSTM 语言模型的基线系统在 BLEU 分数上有大约绝对 2%~3% 的提升.

关键词 自然语言生成;马尔可夫决策过程;先验控制向量;策略梯度算法;深度强化学习
中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2022.00289

Markov Decision Process and Prior Control Vector for Weak Condition Natural Language Generation

LIU Qi MA Rao YU Kai

(*MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai* 200240)
[2)] (*Lab of Cross-Media Language Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai* 200240)
[3)] (*State Key Lab of Media Convergence Production Technology and Systems, Beijing* 100803)

Abstract Communicating with computers by using natural language is one of the ultimate goals of artificial intelligence. The automatic text generation, i.e., natural language generation, is an important yet challenging problem. One popular choice of natural language generation is using language models, including n-gram and feed-forward neural network. Recently, with the development of recurrent neural networks, especially long short-term memory (LSTM), the ability of language models to process sequential text data has been greatly enhanced. Similarly, the LSTM based encoder-decoder has been deployed in many applications involving with natural language generation such as machine translation, video caption and speech recognition. However, LSTM language models and encoder-decoder networks are trained by the cross entropy criterion at the word level, which is to maximize the log likelihood of oracle word sequence. This may lead to the exposure

收稿日期:2020-10-26;在线发布日期:2021-05-31.本课题得到上海市市级科技重大专项(2021SHZDZX0102)和新华社媒体融合生产技术与系统国家重点实验室第一联合创新中心(SKLMCPTS2020003)资助.**刘 奇**,博士,主要研究方向为语音识别、语言模型和强化学习.E-mail:liuq901@sjtu.edu.cn.**马 娆**,硕士,主要研究方向为语言模型、大词汇连续语音识别和语音合成.**俞 凯**(通信作者),博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为语音识别、语音合成、自然语言理解和对话管理.E-mail:kai.yu@sjtu.edu.cn.

bias problem: the input of the model during the training phase is the oracle reference sequence, while during the test phase the input is its own predicted sequence. Another problem is that the cross entropy is calculated at the word level. During the test phase, the model tries to generate the word which maximizes the probability at this specific frame, which ignores the naturality of the whole sequence. In other words, the mismatch between the training criterion and the final test metric may lead to performance degradation. One way to bridge the gap between the word level training criterion and sequence level test metric is to provide extra prior knowledge to the model. Normally, the commonly used structured language model with additional input such as pos tag or topic model could be used to bridge the gap. Some researchers also focus on extracting sentence level embedding to generate better sequences. These methods, including schedule sampling, can also reduce the influence of exposure bias. Another possible solution is to modify the training criterion. Generative adversarial network (GAN) has shown the powerful ability to generate images. Some researchers also implement sequence GAN by using policy gradient. However, sequence GAN does not directly optimize the test metric, and it is hard to train and converge. In this paper, we focus on the natural language generation with the weak condition, i.e., the text itself. We reformulate the natural language generation problem with Markov decision process and incorporate prior control vector derived from training corpus to guide the generation. Prior control vectors can be considered as representation of prior partitions of sequence-level sentence space. It is extracted from the training data unsupervisedly and provides prior knowledge to help the model generate better results. With this provided prior knowledge, the language generation space is effectively narrowed and more appropriately guided. By the Markov decision process definition, we utilize policy gradient to train the LSTM network directly with the test metric rather than the cross entropy loss. The experimental results show that the proposed methods can address the above mentioned problems and significantly improve the generation performance. The proposed model gain about 2%—3% performance improvement on BLEU score compared with baseline model on three different dataset.

Keywords natural language generation; Markov decision process; prior control vector; policy gradient method; deep reinforcement learning

1 引 言

以自然语言作为媒介进行人类和计算机之间的交互是人工智能一个非常重要的研究方向. 在这之中,自动文本生成,一般也被称作自然语言生成,是目前许多应用如机器翻译、视频标注、语音识别等的重要组成部分. 最为普遍使用的自然语言生成的方法是通过语言模型,包括 n-gram 语言模型[1]和前馈神经网络语言模型[2]. 随着近些年来循环神经网络的发展,特别是 LSTM[3]的广泛应用,语言模型建模序列文本数据的能力获得了极大的提升[4-5]. 类似地,基于 LSTM 的编码器-解码器[6]也在应用自然语言生成的任务如机器翻译[7]、视频标注[8]和语音识别[9]中取得了很好的效果.

然而,LSTM 语言模型和基于 LSTM 的编码器-解码器[6]都是使用交叉熵作为训练准则,即最大化参考词语序列的对数似然[10]. 根据 Bengio 等人[11]以及 He 等人[12]的研究,交叉熵准则会导致暴露偏差问题:在模型对第 i 个词进行生成的时候,测试阶段模型的输入是上一时刻生成的第 $i-1$ 个词. 这与训练阶段是不匹配的,因为训练阶段输入是参考序列的 $i-1$ 个词. 这导致模型的训练会过多的匹配训练数据集本身的特性,而不是泛化的生成能力. 另一个问题在于,交叉熵是在词语级别计算的,在测试阶段进行生成的时候会过多注重当前词语的概率,而忽视了整个句子的自然度. 换言之,训练使用的词级别准则和测试使用的序列级别准则的失配可能会导致性能的下降.

一种缩小词级别训练准则和序列级别测试准则

的方式是加入额外的先验知识，如常用的结构化语言模型[13]．这类语言模型通常加入了如语义标签[14-15]或主题模型[16-17]等额外的输入，这些输入可以有效的缩小训练测试准则之间的差距．同时，一些研究也专注于提取序列层面的嵌入向量来帮助生成[18]，这一类方式包括进度抽样[19]不仅可以减少准则之间的差距，也能在一定程度上避免暴露偏差问题．

另一类解决此问题的方式是直接修改训练准则．生成对抗网络(Generative Adversarial Network，GAN)展示了在图片生成上的强大能力[20]．于是 Yu 等人[21]通过策略梯度算法实现了序列 GAN 以生成文本序列．然而，序列 GAN 并没有直接基于测试准则进行训练，同时模型本身训练比较困难，难以收敛．因此类似地，通过策略梯度算法，一些研究者[22-23]尝试直接通过测试准则来训练基于编码器-解码器的神经网络．

本文相比于机器翻译等强约束自然语言生成问题，更关注于弱约束自然语言生成．强约束自然语言生成是指给定了输入序列(如机器翻译中的原文、视频标注中的视频、语音识别中的声学特征)，生成输入序列对应的文本．而弱约束自然语言生成没有这样物理上的对应输入序列，而是通过文本序列本身进行生成，如输入法、网页搜索等通过输入的部分文本猜测后文等．一般的，弱约束自然语言生成指的是只给定不完整的文本序列前缀，生成完整的文本序列的任务．弱约束自然语言生成问题由于目前的应用面较窄，因此研究较少．但是在智能输入法、网页搜索、歌词生成等一些领域，弱约束自然语言生成更为重要，且由于其条件更弱，对于弱约束自然语言生成的优化可以很容易反哺到强条件自然语言生成中．本文针对弱约束自然语言生成，提出了使用先验控制向量作为模型的额外输入的方法．先验控制向量是从训练数据中无监督提取出来的，可以帮助生成更好的结果．同时，我们使用马尔可夫决策过程重定义了弱约束自然语言生成问题，并通过策略梯度算法直接通过测试准则来训练模型．

本文第 2 节探讨一些本文的相关工作；第 3 节给出自然语言生成的问题定义；第 4、5 节描述本文提出的模型；第 6 节给出实验结果；最后第 7 节是全文总结；第 8 节是对于将来工作的讨论．

2　相关工作

对于自然语言生成任务，初始的研究主要集中在网络结构的改进上，在这之中，编码器-解码器[6]的提出具有里程碑的意义．之后注意力机制的提出[24]、双向 LSTM 的使用[25]、transformer 的提出[26]等均大大提高了自然语言生成模型的性能．但这一系列工作均局限于机器翻译等强约束自然语言生成问题，弱约束自然语言生成问题受困于其自回归特性，至今依然以单向 LSTM 模型为主流．因此弱约束自然语言生成问题的优化主要集中在单向 LSTM 模型的改进上．

其中一个改进方向是提供额外的输入．通过额外的输入来添加先验知识是自然语言生成中常用的技巧．一些工作[15,19]将领域/主题信息或是预先定义好的语义标签作为额外的输入．但这一类信息一般都需要有监督学习进行训练，某些数据集并不提供这些额外的标注．一些研究者在模型中加入了后向信息[27-28]，可以提高生成模型的性能，但是在弱约束自然语言生成问题中很难有效获取后向信息．另外一些工作[18,29-30]尝试了多种方式从数据中无监督地提取了额外的输入来生成更精确的结果．然而这一类方法尽管无监督提取信息的方式不同，在测试阶段，这些信息都是通过输入序列和部分的生成序列获取的．准确地说，假设模型的输入是 $\boldsymbol{x}$，生成的部分序列为 $\boldsymbol{y}$，对应的参考序列为 $\boldsymbol{w}$．在测试阶段，上述工作生成的信息是根据 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 生成的，由于生成的部分序列 $\boldsymbol{y}$ 也是通过 $\boldsymbol{x}$ 生成的．因此提取的额外信息局限于输入序列 $\boldsymbol{x}$ 能够提供的部分．因此，本文的方法尝试从参考序列 $\boldsymbol{w}$ 提取额外信息．这一方式，类似于 Yu 等人[31]以及 Johnson 等人[32]的方式，可以提供输入序列以外的信息以帮助进行文本生成．我们称这里的额外输入为先验控制向量，简称控制向量．取名控制向量是因为这些向量能够独立于模型的输入控制生成的序列．

由于常用的测试准则如 BLEU 分数[36]通常都是不可微的，所以不能直接使用反向传播来训练神经网络．此时，我们可以通过策略梯度算法[33]来直接使用 BLEU 分数[36]训练神经网络．在之前的工作中，Yu 等人[21]使用了策略梯度算法进行训练，但是训练准则是 GAN 误差而不是测试使用的 BLEU 分数．Ranzato 等人[22]、Bahdanau 等人[34]和 He 等人[35]通过策略梯度算法直接将 BLEU 分数作为训练准则．然而，以上的工作都是基于编码器-解码器架构的，可以通过输入序列获得完整的语义信息．同时，这些工作关注的均为强约束自然语言生成．本文中，我们在单个 LSTM 的框架下使用了策略梯度算

法,并应用于弱约束自然语言生成,即生成的时候没有输入序列提供的完整语义信息,只有生成的部分序列作为约束,需要在生成同时进行决策.更进一步的,我们还使用马尔可夫决策过程重定义了弱约束自然语言生成问题,这使得使用其他深度强化学习的方法来建模弱约束自然语言生成问题变为可能.由于本文提出的方法在训练和测试时都使用 BLEU 分数[36]作为准则,因此大大减弱了暴露偏差问题的影响.

3 自然语言生成

本文中,我们更关注于弱约束自然语言生成问题.与强约束自然语言生成问题相比,弱约束的生成只有部分的文本信息作为条件.具体的弱约束自然语言生成问题定义,测试准则和基线系统将在这一节之后的部分进行描述.

3.1 词语序列

自然语言生成问题指的是通过一定的条件生成给定的文本序列.本文中我们关注的是弱约束自然语言生成,即给定的条件是参考序列的有限长度的历史.

具体地,给定参考的词语序列 $\boldsymbol{w}=(w_1,\cdots,w_T)$,序列的前缀 $w_1,\cdots,w_L$ 作为生成的条件,生成剩余的部分 $y_{L+1},\cdots,y_T$.给定序列前缀和生成的后缀的连接 $\boldsymbol{y}=(w_1,\cdots,w_L,y_{L+1},\cdots,y_T)$ 定义为预测序列,对应的原始的词语序列 $\boldsymbol{w}$ 称之为参考序列.为了简化问题,本文中定义给定前缀的长度 $L=5$.

3.2 测试准则

如何判断自然语言生成,特别是弱约束自然语言生成结果的好坏,一直是一个开放性的问题.除去人工评测外,我们希望找到一种方式来量化的评价提出的生成模型.由于预测序列是通过参考序列给定长度的前缀生成的,所以我们决定使用参考序列和预测序列的相似度来描述模型的性能.本文中,我们使用 BLEU 分数[36]来作为相似度的量化指标,测试准则定义为预测序列和参考序列之间的 BLEU 分数[36],

$$\mathcal{L}=\mathrm{BLEU}(\boldsymbol{y},\boldsymbol{w}).$$

3.3 LSTM 语言模型基线系统

对于词语序列 $w_1,\cdots,w_T$,其联合概率可以通过条件概率的链式法则展开为

$$P(w_1,\cdots,w_T)=\prod_{i=1}^{T}P(w_i\mid w_1,\cdots,w_{i-1}).$$

语言模型一般用于建模条件概率 $P(w_i\mid w_1,\cdots,w_{i-1})$,即给定词语序列历史时当前词语的概率分布.因此,语言模型可以作为弱约束自然语言生成的一个天然的基线系统.

LSTM 语言模型是目前最为广泛使用的神经网络语言模型.令 V 表示词表.LSTM 语言模型一般包含输入层、隐层和输出层.输入层包含一个 $|V|$ 行的参数矩阵 $\boldsymbol{W}_{\mathrm{in}}$ 将输入的词语 w_i 射到词嵌入向量 $\boldsymbol{x}_i$.隐层则将 $\boldsymbol{x}_i$ 和前一时刻 LSTM 的隐向量 $\boldsymbol{h}_{i-1}$ 和状态 $\boldsymbol{c}_{i-1}$ 作为输入,附加本层的参数 $\boldsymbol{W},\boldsymbol{b}$,通过以下公式计算当前时刻的隐向量 $\boldsymbol{h}_i$ 和状态 $\boldsymbol{c}_i$:

$$\begin{bmatrix}\boldsymbol{i}_i\\ \boldsymbol{o}_i\\ \boldsymbol{f}_i\\ \boldsymbol{m}_i\end{bmatrix}=\begin{bmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{bmatrix}\left(\boldsymbol{W}\begin{bmatrix}\boldsymbol{x}_i\\ \boldsymbol{h}_{i-1}\\ \boldsymbol{c}_{i-1}\end{bmatrix}+\boldsymbol{b}\right),$$

$$\boldsymbol{c}_i=\boldsymbol{f}_i\odot\boldsymbol{c}_{i-1}+\boldsymbol{i}_i\odot\boldsymbol{m}_i,$$

$$\boldsymbol{h}_i=\boldsymbol{o}_i\odot\tanh(\boldsymbol{c}_i).$$

最后,输出层通过参数 $\boldsymbol{W}_{\mathrm{out}},\boldsymbol{b}_{\mathrm{out}}$ 计算最终的概率分布:

$$P_\theta(*\mid w_1,\cdots,w_i)=\mathrm{softmax}(\boldsymbol{W}_{\mathrm{out}}\boldsymbol{h}_i+\boldsymbol{b}_{\mathrm{out}}).$$

这里 θ 对应 LSTM 语言模型的所有参数:

$$\theta=\{\boldsymbol{W}_{\mathrm{in}},\boldsymbol{W},\boldsymbol{W}_{\mathrm{out}},\boldsymbol{b},\boldsymbol{b}_{\mathrm{out}}\}.$$

基于训练好的 LSTM 语言模型,我们可以通过基本的自回归方法[37]生成预测序列 $\boldsymbol{y}$.模型首先根据概率分布 $P_\theta(*\mid w_1,\cdots,w_L)$ 生成词语 y_{L+1},随后根据概率分布 $P_\theta(*\mid w_1,\cdots,w_{L+1})$ 生成词语 y_{L+2},依此类推直到生成最后一个词语 y_T.我们使用这样自回归生成的 LSTM 语言模型作为基线系统.

4 先验控制向量

LSTM 语言模型基线系统一个很大的问题在于,其只能从参考序列中获取很少的信息.由于一般词表 V 的大小在几千到上万的数量级,这造成了生成空间十分巨大,预测序列很难靠近参考序列.为了生成更好的句子,我们希望从参考序列中提取有效的先验信息.因此,本文提出了控制向量来控制生成结果并缩小生成空间.相比于之前的类似工作,控制向量是由参考序列 $\boldsymbol{w}$ 而不是预测序列 $\boldsymbol{y}$ 无监督学习得到的,因此可以提供给模型更多关于生成目标的先验信息.

4.1 无监督 K 近邻提取

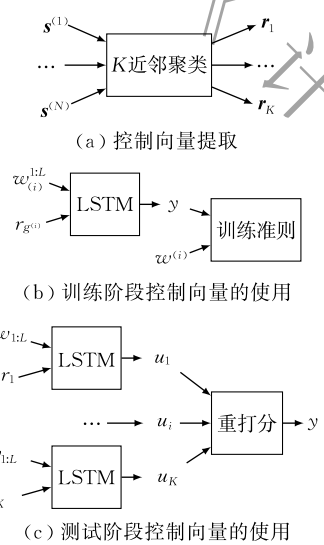
本文中,我们采取了一种简单的方式从参考序列中提取控制向量.假设训练集共有 N 个参考序列

$\boldsymbol{w}^{(1)},\cdots,\boldsymbol{w}^{(N)}$，对于其中第 j 个序列 $\boldsymbol{w}^{(j)}=(w_1^{(j)},\cdots,w_T^{(j)})$，将每一个词语 $w_i^{(j)}$ 在 LSTM 语言模型中的输入层对应的词嵌入向量 $\boldsymbol{x}_i^{(j)}$ 提取出来. 定义所有的词嵌入向量的平均值为这个序列的句嵌入向量：

$$\boldsymbol{s}^{(j)}=\frac{1}{T}\sum_{i=1}^{T}\boldsymbol{x}_i^{(j)}.$$

提取了句嵌入向量 $\boldsymbol{s}^{(1)},\cdots,\boldsymbol{s}^{(N)}$ 后，我们使用 K 近邻聚类[38]将其分为 K 组. 假设第 j 个句嵌入向量 $\boldsymbol{s}^{(j)}$ 属于第 $g^{(j)}$ 组. 令每个组的 K 近邻中心为 $\boldsymbol{r}_1,\cdots,\boldsymbol{r}_K$，我们使用这些 K 近邻中心作为控制向量.

在训练过程中，控制向量将与词嵌入向量一起作为输入. 对于第 j 个序列，令 $\boldsymbol{x}_i'^{(j)}=[\boldsymbol{x}_i^{(j)},\boldsymbol{r}_{g^{(j)}}]$，我们将使用 $\boldsymbol{x}_i'^{(j)}$ 代替基线系统使用的 $\boldsymbol{x}_i^{(j)}$ 作为 LSTM 隐层的输入. 控制向量的提取和训练过程的使用可以参看图 1(a)和(b).



(a) 控制向量提取

(b) 训练阶段控制向量的使用

(c) 测试阶段控制向量的使用

图 1　控制向量提取、训练、测试的过程

4.2　枚举与重打分

为了获取准确的控制向量，我们需要计算参考序列 $\boldsymbol{w}$ 的句嵌入向量. 但是 $\boldsymbol{w}$ 在测试阶段是未知的. 注意到控制向量与生成时的输入是无关的，因此在这里我们使用枚举来应用控制向量. 在测试阶段，枚举所有的 K 个控制向量 $\boldsymbol{r}_1,\cdots,\boldsymbol{r}_K$ 生成 K 个候选的预测序列，问题转化为如何在这 K 个候选序列中挑选出最佳的预测序列.

一种思路是人工评判，为了模拟人工评判的效果，我们假设人工评判会选取使用最优控制向量的候选序列. 即我们计算测试集参考序列对应的句嵌入向量，选取最近的 K 近邻中心作为人工评判模拟的控制向量来生成最佳的预测序列.

但是，在 K 增加到 100 甚至 1000 的数量级的时候，从如此数量的候选序列中让人工进行评判变得几乎不可能. 因此，我们需要一种自动量化的方式来选取最佳的候选序列. 这里我们使用重打分[39]来选取最佳序列. 对于给定的参考序列 $\boldsymbol{w}$，令 $\boldsymbol{U}=(\boldsymbol{u}_1,\cdots,\boldsymbol{u}_K)$ 为所有的候选序列. 我们使用一个重打分模型选取最佳的预测序列：

$$\boldsymbol{y}=\arg\max_{\boldsymbol{u}\in\boldsymbol{U}}\text{score}(\boldsymbol{u}).$$

图 1(c)给出了重打分过程的图示. 这里，我们使用第 3.3 节描述的 LSTM 语言模型基线系统作为重打分模型，对应的分数为候选序列的联合概率 $P_\theta(\boldsymbol{u}_j)$.

4.3　试探开发平衡

在人工智能领域，特别是强化学习中有一个非常重要的议题，即著名的试探-开发平衡问题[33]. 在训练过程中，试探一般指的是模型需要尝试多种不同的动作，这样才可能试探到更优性能的动作路径. 而开发指的是模型需要考虑之前的经验，尽量避免重复性能偏低的动作路径. 如果过于注重试探，模型很难将获取的经验存储下来，退化为一个随机模型，导致非常差的性能. 如果过于注重开发，模型会重复之前的最优路径，不再考虑去探索新的可能，造成模型很难达到理论上的最优点. 因此，我们必须在试探和开发之前找到一个平衡点.

然而实际上，控制向量中分组的个数 K 的选取是一个天然的控制试探和开发平衡的方式. 后文为了简化，我们用 $P_\theta(*|\boldsymbol{y},\boldsymbol{r})$ 来代表提出的基于控制向量的模型，这里 $\boldsymbol{y}$ 对应部分的预测序列，$\boldsymbol{r}$ 对应控制向量，分别代表模型的两个输入. 当 $K=1$ 时，训练集的所有序列有着完全相同的控制向量. 这使得模型无法通过控制向量来区分不同的参考序列，只能通过有限长度的历史，即给定的长度为 L 的参考序列前缀来进行生成. 在这种情况下，模型生成的概率完全由部分的预测序列确定，完全相同的控制向量对生成的影响几乎没有. 即在 $K=1$ 时，有：

$$P_\theta(*|\boldsymbol{y},\boldsymbol{r})\approx P_\theta(*|\boldsymbol{y}).$$

此时 P_θ 是一个纯试探模型，其只通过部分的预测序列进行之后的生成，没有使用到数据集的任何先验知识.

另一方面是，当 K 非常大的时候，例如当 K 等于训练集中序列个数 N 的时候，每一个训练集中的序列都有一个不同的控制向量，即其自己对应的句嵌入向量. 此时模型的任务从自然语言生成变为了给定句嵌入向量恢复序列本身. 若将提取句嵌入向量的网络视作编码器，训练的网络视作解码器，整个系统近似为一个自生成的编码器-解码器系统. 而自

生成的编码器-解码器系统主要依赖于编码器进行生成，即模型确定生成序列会主要根据提供的先验控制向量，而不是部分的预测序列，因此：

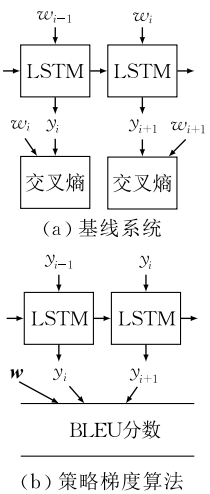
$$P_\theta(*|\boldsymbol{y},\boldsymbol{r})\approx P_\theta(*|\boldsymbol{r}).$$

即在 $K=N$ 时，模型变为一个纯开发模型，模型只通过提供的先验控制向量来确定生成的预测序列.

综上所述，选取合适的 K 的值对生成性能的影响较大. 在 K 值过小的时候，模型能够获取的先验信息较少，可能会导致性能的下降. 但是，在 K 值过大的时候，模型倾向于通过控制向量重复生成训练数据中的参考序列，很容易造成过拟合，使得模型变为一个复制系统而不是我们需要的自然语言生成系统.

5 马尔可夫决策过程重定义

基线系统的另一个非常大的问题是，传统的 LSTM 语言模型是通过交叉熵训练准则进行训练的. 由于交叉熵是在词级别计算的，并且在训练时模型的输入是参考序列，因此会导致暴露偏差问题. 更进一步的，词级别的训练使得模型在生成时会过于关注当前词语本身的概率，而忽视整个序列的自然度. 因此，一种理想的解决方案是直接通过测试准则 BLEU 分数[36]作为训练准则. 本节中，我们使用马尔可夫决策过程重定义了弱约束自然语言生成问题，并使用强化学习中非常常用的方法——策略梯度算法来直接使用 BLEU 分数[36]训练模型. 基线系统和使用策略梯度算法训练的系统的基本比较可以参看图 2.



(a) 基线系统

(b) 策略梯度算法

图 2 基线系统和使用策略梯度算法进行训练的系统的区别(策略梯度算法的训练输入的是预测序列而不是参考序列，并且使用的训练准则是BLEU 分数[36]而不是交叉熵)

5.1 马尔可夫决策过程

马尔可夫决策过程(MDP)由四元组$(S,A(s),R(s,a),P(s,a,s'))$定义[40]. 这里 S 是所有可能的状态的集合. $A(s)$是在状态 s 的所有可能的动作的集合. $R(s,a)$是收益函数，表示在状态 s 采取动作 a 时获取的收益. $P(s,a,s')$为状态转移概率，代表在状态 s 采取动作 a 后转移到状态 s'的概率.

我们现在使用一个 MDP 来定义弱约束自然语言生成问题. 令 $\boldsymbol{w};\boldsymbol{y}$ 代表参考序列 $\boldsymbol{w}$ 和其对应的部分生成序列 $\boldsymbol{y}$，V 代表词表，则有状态集为

$$S=\{\boldsymbol{w};\boldsymbol{y}|\boldsymbol{w}\in V^T,\boldsymbol{y}\in V^{\leq T}\cup\{\mathcal{T}\}\}.$$

这里$\mathcal{T}$是一个特殊的代表结束的符号. 初始状态集合 S_0和终止状态集合 S_T可以定义为

$$S_0=\{\boldsymbol{w};\varnothing|\boldsymbol{w}\in V^T\},$$
$$S_T=\{\boldsymbol{w};\mathcal{T}|\boldsymbol{w}\in V^T\}.$$

对于动作，可以很自然的定义为从 V 中选取一个词语放在现有的部分预测序列的最后. 我们定义动作由词语 w 表示，意义为使用 w 延长当前的部分预测序列. 如果使用 w_i表示参考序列 $\boldsymbol{w}$ 的第 i 个词语，那么动作在特定状态的集合可以定义为

$$A(\boldsymbol{w};\boldsymbol{y})=\begin{cases}\{w_{|\boldsymbol{y}|+1}\}, & |\boldsymbol{y}|<L\\ \{w|w\in V\}, & L\leq|\boldsymbol{y}|<T\\ \{terminate\}, & |\boldsymbol{y}|=T\end{cases}.$$

当预测序列的长度$|\boldsymbol{y}|<L$ 时，依然处于复制参考序列长度为 L 的前缀的阶段，因此唯一的动作便是选取参考序列对应位置的词语. 当$|\boldsymbol{y}|=T$ 时，生成已经结束，我们采取一个特殊的动作 *terminate* 终止生成过程. 其他情况下，我们可以任意从 V 中挑选一个词来扩展预测序列 $\boldsymbol{y}$.

收益只在采取动作 *terminate* 时才进行计算，其他动作的收益设为 0：

$$R(\boldsymbol{w};\boldsymbol{y},a)=\begin{cases}\mathrm{BLEU}(\boldsymbol{y},\boldsymbol{w}), & a=terminate\\ 0, & \text{其他情况}\end{cases}.$$

可以很容易得到这里定义的 MDP 是一个确定性的 MDP，任意状态采取任何动作都会转移到一个给定的后继状态，因此：

$$P(\boldsymbol{w};\boldsymbol{y},w,\boldsymbol{w};[\boldsymbol{y},w])=1,$$
$$P(\boldsymbol{w};\boldsymbol{y},terminate,\boldsymbol{w};\mathcal{T})=1,$$
$$P(\text{其他情况})=0.$$

即可能的状态转移只有使用动作对应的词语扩展预测序列，以及选择动作 *terminate* 进入终止状态.

使用 MDP 重定义弱约束自然语言生成后，便可以尝试使用强化学习的算法来解决弱约束自然语

言生成问题.但是值得注意的是,强化学习的方法基本上都基于“尝试-改进”流程,即在动作空间中进行随机游走,通过获取收益的不同不断改进策略.而弱约束自然语言生成的动作空间,即生成空间十分巨大,直接进行无规律的尝试很难靠近参考序列,从而无法获得收益以改进策略.注意到,第 4 节提出的先验控制向量的目的就是为了缩小生成空间,使得模型更容易生成和参考序列相近的序列.由此可见,控制向量能够更好地帮助强化学习的算法进行“尝试-改进”流程.同时注意到,虽然上文的 MDP 定义中不包含控制向量,但是上给定控制向量 $\boldsymbol{r}$,可以很简单的将状态的形式由 $\boldsymbol{w};\boldsymbol{y}$ 修改为 $\boldsymbol{w};\boldsymbol{y};\boldsymbol{r}$,动作集和状态转移概率进行类似的修改即可,整个 MDP 的定义基本保持一致,这里不再赘述.

5.2　策略梯度算法

通过上一节的 MDP 重定义,我们可以尝试使用强化学习的算法来解决弱约束自然语言生成问题.注意到在 LSTM 语言模型基线系统中,生成过程中是将部分的预测序列作为模型的输入来确定当前词语的概率,令 $\boldsymbol{y}_{1:i}$ 表示预测序列 $\boldsymbol{y}$ 的长度为 i 的前缀,则在生成第 $i+1$ 个词语的时候,基线系统可以看做是根据概率 $P_\theta(*\mid\boldsymbol{y}_{1:i})$ 进行词语的选取.回顾我们使用的 MDP 的定义,弱约束自然语言生成问题的动作是选取合适的词语 w,依据的状态是参考序列和部分的预测序列 $\boldsymbol{w};\boldsymbol{y}_{1:i}$.然而,状态中的参考序列 $\boldsymbol{w}$(准确来说是参考序列从第 $L+1$ 个词开始的后缀),在整个生成过程中是不可见的,只有在最后进行收益计算 BLEU 分数[36] 的时候才能被观测到.因此,在真正确定需要生成哪一个词语的时候,能够依据的 MDP 状态只有部分生成序列 $\boldsymbol{y}_{1:i}$.由此可得,基线系统建模的概率 $P_\theta(*\mid\boldsymbol{y}_{1:i})$ 可以用来建模 MDP 中的策略网络 $\pi(a\mid s)$,即给定状态 s 时各个动作的概率,于是我们使用了和基线系统完全相同的网络结构来建模策略网络 $\pi(a\mid s)$.注意到,在弱约束自然语言生成问题中,生成空间,即 MDP 对应的动作空间非常巨大,即使控制向量可以有效的缩小动作空间,但是如果训练的策略网络 $\pi(a\mid s)$ 是基于随机初始化的,训练过程会非常不稳定并且难以收敛.然而,正由于基线系统和策略网络是相同的结构,于是我们使用基线系统作为策略网络 $\pi(a\mid s)$ 的初始化.

由于准则 $\mathcal{L}=\mathrm{BLEU}(\boldsymbol{y},\boldsymbol{w})$ 关于策略网络的参数是不可微的,也就不能直接计算参数对应的梯度.为了训练策略网络,我们需要使用策略梯度算法来计算策略网络参数的梯度.一般的,策略梯度算法使用以下公式预估梯度:

$$\nabla\mathcal{L}=\mathbb{E}\Big[\sum_{i=1}^{\infty}\Psi_i\nabla\pi(a_i\mid s_i)\Big].$$

根据 Schulman 等人[41] 的研究,Ψ 是一类收益信号函数,一般可以使用以下形式:

(1) $\sum_{j=1}^{\infty}R(s_j,a_j)$:总收益;

(2) $\sum_{j=i}^{\infty}R(s_j,a_j)$:当前动作开始的总收益;

(3) $\sum_{j=i}^{\infty}R(s_j,a_j)-B(s_j)$:基于基线函数的形式(2);

(4) $Q(s_i,a_i)$:动作值函数;

(5) $Q(s_i,a_i)-V(s_i)$:优势函数;

(6) $R(s_i,a_i)+V(s_{i+1})-V(s_i)$:时序差分误差.

这里的 $R(s,a)$是收益函数,$B(s)$是自定义的基线函数,$V(s)$为状态值函数,指的是在某个状态开始的总收益的期望,类似的 $Q(s,a)$为动作值函数,指某个状态采取给定动作后的总收益的期望.

我们需要选取一个合适的形式作为收益信号函数来训练策略网络.形式(1)和(2)即为 REINFORCE 算法[33,42-43].但是,本文对应的 MDP 所有的收益都是非负的,这样总收益也是非负的.我们发现这会导致普通的 REINFORCE 算法收敛缓慢.形式(4)、(5)和(6)是著名的行动器-判别器算法[33,44-46].在行动器-判别器算法的实现中,值函数 $Q(s,a)$和 $V(s)$一般是使用神经网络近似的.但是,我们的策略网络使用基线系统作为初始化,对应的值函数网络却无法找到一个合适的初始化,这会导致训练过程不稳定.因此,我们选取了形式(3),即基于基线函数的 REINFORCE 算法训练策略网络 $\pi(a\mid s)$.

基线函数 $B(s)$理论上可以任意选择,甚至可以使用随机函数.但是合适的基线函数可以加速模型的收敛.这里,我们很自然的使用 LSTM 语言模型基线系统的 BLEU 分数[36] 作为基线函数.回顾预测序列 $\boldsymbol{y}=(w_1,\cdots,w_L,y_{L+1},\cdots,y_T)$,则 $\pi(a\mid s)=P_\theta(y_i\mid\boldsymbol{y}_{1:i-1})$.策略梯度算法中使用的收益信号是当前动作开始的总收益与基线系统 BLEU 分数[36] 的差.根据 MDP 的定义,收益在除了终止动作以外均为 0,因此有 $\sum R(s,a)=\mathrm{BLEU}(\boldsymbol{y},\boldsymbol{w})$.类似地,定义 $\boldsymbol{y}'$为基线系统生成的预测序列,则 $B(s)=\mathrm{BLEU}(\boldsymbol{y}',$

$\boldsymbol{w}$). 那么有:

$$\mathcal{R}=\mathrm{BLEU}(\boldsymbol{y},\boldsymbol{w}),$$
$$\mathcal{B}=\mathrm{BLEU}(\boldsymbol{y}',\boldsymbol{w}),$$
$$\Psi=\mathcal{R}-\mathcal{B}.$$

使用蒙特卡洛法来近似期望,最终对于训练集中的每一个参考序列 $\boldsymbol{w}$,使用策略梯度算法预估的梯度可以计算为

$$\nabla\mathcal{L}=(\mathcal{R}-\mathcal{B})\sum_{i=L+1}^{T}\nabla P_\theta(y_i\,|\,\boldsymbol{y}_{1:i-1}).$$

6 实验结果与分析

6.1 实验设置

我们使用了三个数据集来评测本文提出的模型. 第一个是英文的 Penn Treebank(PTB)数据集[47]. PTB 数据集[47]的文本较为正式,且句子长度较长. 第二个是中文短信(SMS)数据集(自行收集). SMS 数据集较为口语化且句子长度较短. 最后一个是 Fisher(FSH)数据集[48],FSH 数据集[48]也为口语化句子,长度中等. 三个数据集具体的信息可以参见表 1.

表 1 三个数据集具体的序列个数、平均长度和词表大小

数据集	序列个数	平均长度	词表大小
PTB[47]	42068	21.09	10001
SMS	380000	7.08	40698
FSH[48]	2477493	12.05	30276

对于 PTB[47]和 SMS 数据集,网络结构为三层的 LSTM,每层包含 300 个节点. 对于 FSH[48]数据集,网络结构为单层的 LSTM,每层 600 个节点. 对于所有的数据集,基线系统和使用策略梯度算法进行训练的系统有着完全相同的网络结构. 对于使用了先验控制向量的系统,如同第 4.1 节提及的,除去第一层 LSTM 的输入是词嵌入向量和控制向量的连接以外,网络的其他部分结构和不使用控制向量的相同. 对于所有的实验,参考序列给定的历史长度 $L=5$. 所有的神经网络使用 PyTorch[49]在一块 NVIDIA GeForce GTX 1080 Ti GPU 上进行训练.

6.2 实验结果

表 2 给出了策略梯度算法的实验结果. 可以看到在所有数据集上,基于策略梯度算法的系统相比于基线系统都给出了更好的性能. 这说明通过直接使用测试准则来训练神经网络,本文提出的模型有着更好的生成性能. 此外,由于基线系统训练时输入的是参考序列,而本文提出的系统在训练时使用的是预测序列,因此受到暴露偏差问题的影响也较小.

表 2 策略梯度算法的实验结果

数据集	训练准则	训练算法	BLEU 分数[36]
PTB[47]	交叉熵	反向传播算法	25.07±0.07
	BLEU 分数[36]	策略梯度算法	27.00±0.06
SMS	交叉熵	反向传播算法	68.11±0.15
	BLEU 分数[36]	策略梯度算法	69.35±0.10
FSH[48]	交叉熵	反向传播算法	31.47±0.07
	BLEU 分数[36]	策略梯度算法	32.89±0.03

表 3 给出了先验控制向量的实验结果,这里对应的控制向量的组数 $K=20$. 共计采用了四种方式进行重打分:

(1)部分预测. 使用参考序列的部分历史 $\boldsymbol{w}_{1:L}$ 作为输入训练了一个 K 近邻聚类的分类器. 以分类器概率最大的类别对应的控制向量生成的候选序列作为预测序列;

(2)生成得分. 进行生成的模型可以计算出生成的序列对应的联合概率,以这个联合概率作为评分的标准;

(3)LSTM 重打分. 使用 LSTM 语言模型计算联合概率作为评分,具体细节可以参见第 4.2 节;

(4)最优选择. 对人工评测的模拟,具体细节可以参见第 4.2 节.

表 3 先验控制向量的实验结果(两列数据分别为使用交叉熵准则和策略梯度算法的模型的 BLEU 分数[36]. CV 表示是否使用了先验控制向量. 不同的重打分方法的细节请参见第 6.2 节)

数据集	CV	重打分方法	交叉熵	策略梯度
PTB[47]	×	无	25.07±0.07	27.00±0.06
	√	部分预测	24.88±0.11	26.91±0.12
	√	生成得分	24.89±0.10	26.80±0.09
	√	LSTM 重打分	25.70±0.06	27.21±0.05
	√	最优选择	**26.03±0.10**	**28.02±0.12**
SMS	×	无	68.11±0.15	69.35±0.10
	√	部分预测	68.00±0.10	69.09±0.09
	√	生成得分	66.91±0.06	67.70±0.05
	√	LSTM 重打分	**69.82±0.06**	69.76±0.07
	√	最优选择	69.21±0.08	**70.42±0.09**
FSH[48]	×	无	31.47±0.03	32.89±0.03
	√	部分预测	31.21±0.04	32.73±0.03
	√	生成得分	30.98±0.03	32.13±0.04
	√	LSTM 重打分	**32.20±0.03**	32.80±0.03
	√	最优选择	31.96±0.02	**33.37±0.03**

部分预测和生成得分两种方法是对照组. 从表中的结果可以看到,两个对照组的性能都比较差. 如第 4.2 节中描述的,控制向量的优势在于其只依赖

于参考序列，而不依赖于输入序列. 使用部分预测，即 $w_{1:L}$ 来预测对应的 K 近邻聚类无法发挥这样的优势. 生成得分的问题在于，不同的候选序列输入的控制向量不同，候选序列均为对应控制向量概率较高的句子，以这样的联合概率作为评分标准不统一，因此评分并不公平.

但是相应的，本文提出的两种重打分方式均给出了不错的结果. 其中，交叉熵训练准则的模型在LSTM重打分的性能相对较优，而策略梯度算法训练的模型在人工评测的模拟，即最优选择上给出了最好的性能. 这个区别我们认为是由于控制向量包含了关于参考序列的先验信息，因此可以帮助模型生成更自然和精确的结果. 而策略梯度算法面临的一大挑战便是生成空间过大，导致训练中很难试探到与参考序列相近的预测序列，这会使得总收益提升缓慢进而使得模型难以收敛. 因此，策略梯度算法的训练过程可能相对更为依赖控制向量，选取和参考序列相同控制向量能够更好地提升策略梯度模型的性能. 而 LSTM 重打分相对选取的是和数据集语义更为类似的结果，交叉熵准则训练的模型本质上就是 LSTM 语言模型，和 LSTM 重打分的选取方式更为匹配. 总的来说，使用了控制向量的和两种重打分方式都能给出比基线系统更好地结果. 但有一个例外，FSH 数据集[48] 策略梯度模型基于 LSTM 重打分的实验结果略差于基线模型. 我们分析了原因. 一方面如上文所述，策略梯度模型和 LSTM 重打分的相性相对较差一些，另一方面 FSH 数据集[48] 有大量句子是口语词的无意义重复，而策略梯度模型生成的句子较少包含这样的重复，造成 LSTM 打分器给出句子的分数区分度较差，造成性能下降，是数据集的特殊性造成的问题.

此外，对于控制向量的提取，我们也尝试了不同方式. 理论上，任意句嵌入向量均可以使用作为控制向量，本文使用的方法为将词嵌入向量的均值作为每个序列的句嵌入向量. 我们这里也尝试了其他句嵌入向量的提取方式进行对比. 这里我们使用的是预训练语言模型BERT[50]，具体模型为 BERT-BASE，12 层768节点网络. 由于BERT模型在英文上更为稳定，我们在PTB[47] 和FSH[48] 两个英文数据集上进行了实验，实验结果可以参见表 4.

表 4　使用 BERT[50] 提取先验控制向量的实验结果（模型一栏括号中代表提取控制向量的方式. 均值指使用词嵌入向量的均值，BERT[50] 指使用预训练语言模型 BERT-BASE）

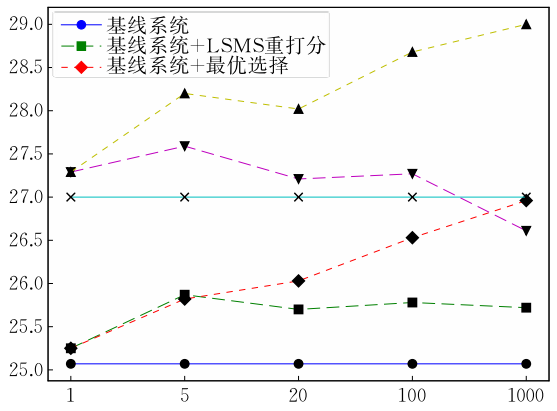
数据集	模型	LSTM 重打分	最优选择
PTB[47]	交叉熵（均值）	25.70±0.06	**26.03±0.10**
	交叉熵（BERT[50]）	**25.99±0.07**	25.60±0.08
	策略梯度（均值）	**27.21±0.05**	**28.02±0.12**
	策略梯度（BERT[50]）	27.09±0.06	27.72±0.11
FSH[48]	交叉熵（均值）	32.20±0.03	31.96±0.02
	交叉熵（BERT[50]）	**32.45±0.02**	**32.03±0.01**
	策略梯度（均值）	**32.80±0.03**	**33.37±0.03**
	策略梯度（BERT[50]）	32.76±0.03	32.73±0.02

从表中可以看到，两种提取控制向量的方法对于不同的模型和重打分方法互有优劣，且性能差距不明显，可以认为两种提取控制向量的方法有着相似的性能. 尽管 BERT[50] 在其他一些自然语言理解任务上有着更好的表现，但是这里我们使用控制向量是为了减少生成空间，用于训练的是句嵌入向量聚类后的中心. 因此提取的句嵌入向量主要用于将训练集合分类并在生成时指示向何种类型的句子进行生成，因而这里使用的句嵌入向量不需要过于精确，能够有效的将句子聚类即可. 所以基于词嵌入向量的均值和基于 BERT[50] 提取的控制向量有着相似的结果. 综上所述，我们认为只要能够表示句子一定的语义，能够有效的将句子进行聚类，便可以作为控制向量并给出与相似的性能.

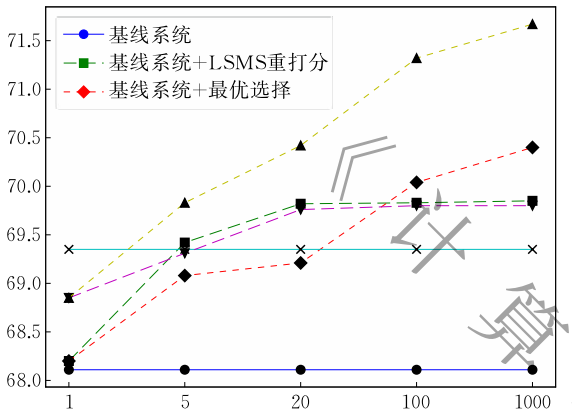
6.3　关于 K 的取值的分析

回顾第 4.3 节中的讨论，控制向量的组数 K 的取值对应了试探和开发的平衡. 因此，我们尝试了不同的 K 的取值，并分析了其对应的结果. 图 3 展示了不同的 K 的取值的实验结果，这里取值范围为 1、5、20、100、1000.

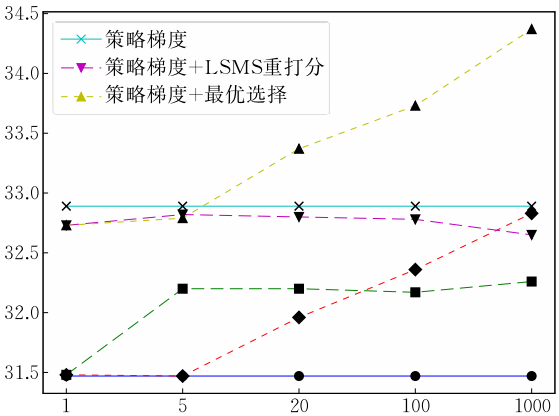
从图中可以观察到，对于人工评测的模拟，即最优选择法（菱形和三角），在 K 值增加时性能提升. 我们认为原因是控制向量随着组数的增多，可以提供更精确的先验信息. 注意到第 4.3 节提及到，过大的 K 值会导致完全的开发模型. 然而，实验中使用的最大的 K 值为 1000，而导致完全的开发模型需要 K 值和训练数据的序列个数相等，从表 1 中可以看到依然有着数量级上的差距. 因此在实验中使用的较小的 K 的取值范围内，随着 K 值增加，模型的性能也随之上升.



(a) PTB数据集[47]



(b) SMS数据集



(c) FSH数据集[48]

图 3 不同的 K 的取值的实验结果(X 轴表示 K 的取值,Y 轴为模型对应的 BLEU 分数[36]. 这里图例中的基线系统指的是使用交叉熵训练的模型,策略梯度指的是使用策略梯度算法训练的模型. 图例中的 LSTM 重打分和最优选择指的是使用了控制向量和对应重打分方法的模型)

然而,对于 LSTM 重打分模型(方块和倒三角),曲线随着 K 值的增加变得平缓. 我们认为这是由于 LSTM 语言模型在候选序列过多的时候,不能很好的确定最优的候选序列. 如何使用自动量化的方法在有大量候选序列时选取合适的预测序列依然是一个有挑战性的问题.

另一个很重要的点在于,生成所需要的时间随着 K 值的增加而增加. 如同第 4.2 节描述的,所有的 K 个控制向量都会输入给模型并生成 K 个候选序列. 因此, $K=1000$ 时,生成的速度相比 $K=1$ 时便会慢上约 1000 倍. 同时,最优选择法是对人工评测的模拟,而真实的人工评测几乎不可能从 100 甚至 1000 个候选序列中选取合适的最优预测序列. 因此,在之前的实验中,我们选取了比较平衡的组数 $K=20$.

最后非常有趣的一点是,对于所有的使用交叉熵训练的模型和使用策略梯度算法训练的模型,策略梯度算法均比使用交叉熵训练的模型有着更好的性能. 不论是否使用控制向量,采取何种方式重打分,不同的 K 值的选取这一结论均成立. 比较除去训练准则有差异以外,其他均相同的两个模型(即图中的圆圈和交叉、方块和倒三角、菱形和三角),基于策略梯度的模型的 BLEU 分数[36]均高于使用交叉熵训练的模型的 BLEU 分数[36]. 由此可得策略梯度算法在弱约束自然语言生成问题上相比于交叉熵训练更为合适.

7 结 论

在本文中,我们提出了两种方法来优化基于单个 LSTM 网络的弱约束自然语言生成系统的性能. 第一个方法是先验控制向量. 先验控制向量是从参考序列中无监督提取而来,可以帮助模型缩小生成空间并提供先验知识生成更为自然的序列. 第二个方法是使用策略梯度算法进行训练,这使得我们可以直接使用测试准则来训练神经网络. 同时,我们还使用 MDP 重定义了弱约束自然语言生成问题,这样可以很容易使用其他强化学习的算法来解决弱约束自然语言生成问题. 从实验结果中可以看出,本文提出的方法相比于普通的 LSTM 语言模型基线系统有着绝对约 2%~3%的提升.

8 未来工作

本文主要针对基于单个 LSTM 网络的弱约束自然语言生成问题. 即生成的序列只依赖于有限长度的参考序列的历史. 但是,LSTM 语言模型与编码器-解码器[6]的解码器部分非常相似,因此本文提出的方法,特别是先验控制向量是否可以使用到诸如机器翻译、视频标注、语音识别等强约束自然语言

生成问题上是我们最为关注的研究方向.

其他的一些未来工作包括研究给定的历史长度 L 对弱约束自然语言生成的模型的性能的影响.此外,目前在先验控制向量的使用中需要枚举所有的控制向量,这样在控制向量组数 K 特别大时,生成速度明显变慢.能否不枚举所有的控制向量,而是只枚举一个子集,以加快 K 值过大时的生成速度.此外,寻找其他更优的自动量化的重打分方法是另一个有意义的研究方向.

致　谢　本文主要研究内容为刘奇在加拿大阿尔伯塔大学访问期间完成.作者感谢加拿大阿尔伯塔大学 RLAI 实验室的 Sutton 教授和万一同学在访问期间给予的帮助!

参　考　文　献

[1] Kneser R, Ney H. Improved backing-off for m-gram language modeling//Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Detroit, USA, 1995: 181-184

[2] Morin F, Bengio Y. Hierarchical probabilistic neural network language model//Proceedings of the International Workshop on Artificial Intelligence and Statistics (AISTATS). Bridgetown, Barbados, 2005: 246-252

[3] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780

[4] Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model//Proceedings of the Annual Conference of the International Speech Communication Association (INTER-SPEECH). Makuhari, Japan, 2010: 1045-1048

[5] Sundermeyer M, Schlüter R, Ney H. LSTM neural networks for language modeling//Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH). Portland, USA, 2012: 194-197

[6] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks//Proceedings of the Conference on Neural Information Processing Systems (NIPS). Montréal, Canada, 2014: 3104-3112

[7] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Computing Research Repository, arXiv: 1406.1078, 2014

[8] Olivastri S, Singh G, Cuzzolin F. End-to-end video captioning. Computing Research Repository, arXiv:1904.02628, 2019

[9] Chan W, Jaitly N, Le Q, et al. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition//Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP). Shanghai, China, 2016: 4960-4964

[10] Salakhutdinov R. Learning deep generative models. Annual Review of Statistics and Its Application, 2015, 2: 361-385

[11] Bengio S, Vinyals O, Jaitly N, et al. Scheduled sampling for sequence prediction with recurrent neural networks//Proceedings of the Conference of Neural Information Processing Systems (NIPS). Montréal, Canada, 2015: 1171-1179

[12] He T, Zhang J, Zhou Z, et al. Quantifying exposure bias for neural language generation. Computing Research Repository, arXiv:1905.10617, 2019

[13] Chelba C, Jelinek F. Structured language modeling. Computer Speech & Language, 2000, 14(4): 283-332

[14] Chelba C, Jelinek F. Structured language modeling for speech recognition. Computing Research Repository, arXiv: cs/0001023, 2000

[15] Cao D, Yu K. Deep attentive structured language model based on LSTM//Proceedings of the International Conference on Intelligent Science and Big Data Engineering (IScIDE). Dalian, China, 2017: 169-180

[16] Mikolov T, Zweig G. Context dependent recurrent neural network language model//Proceedings of the IEEE Spoken Language Technology Workshop (SLT). Miami, USA, 2012: 234-239

[17] Chen X, Tan T, Liu X, et al. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition//Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH). Dresden, Germany, 2015: 3511-3515

[18] Liu Q, Qian Y, Yu K. Future vector enhanced LSTM language model for LVCSR//Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). Okinawa, Japan, 2017: 104-110

[19] Keskar N S, Mccann B, Varshney L R, et al. CTRL: A conditional transformer language model for controllable generation. Computing Research Repository, arXiv: 1909.05858, 2019

[20] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets//Proceedings of the Conference of Neural Information Processing Systems (NIPS). Montréal, Canada, 2014: 2672-2680

[21] Yu L, Zhang W, Wang J, et al. SeqGAN: Sequence generative adversarial nets with policy gradient//Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). San Francisco, USA, 2017: 2852-2858

[22] Ranzato M, Chopra S, Auli M, et al. Sequence level training with recurrent neural networks. Computing Research Repository, arXiv:1511.06732, 2015

[23] Keneshloo Y, Shi T, Ramakrishnan N, et al. Deep reinforcement learning for sequence-to-sequence models. IEEE Transactions on Neural Networks and Learning Systems, 2019, 31(7): 1-21

[24] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014

[25] Ling W, Trancoso I, Dyer C, et al. Character-based neural machine translation. arXiv:1511.04586, 2015

[26] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need//Proceedings of the Conference on Neural Information Processing Systems(NIPS). Long Beach, USA, 2017: 6000-6010

[27] Arisoy E, Sethy A, Ramabhadran B, et al. Bidirectional recurrent neural network language models for automatic speech recognition//Proceedings of the International Conference on Acoustics, Speech and Signal Processing(ICASSP). South Brisbane, Australia, 2015: 5421-5425

[28] Chen X, Liu X, Ragni A, et al. Future word contexts in neural network language models//Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). Okinawa, Japan, 2017: 97-103

[29] Su J, Xiong D, Liu Y, et al. A context-aware topic model for statistical machine translation//Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing(ACL-IJCNLP). Beijing, China, 2015: 229-238

[30] Serdyuk D, Ke N R, Sordoni A, et al. Twin networks: Matching the future for sequence generation. Computing Research Repository, arXiv:1708.06742, 2017

[31] Yu K, Zhao Z, Wu X, et al. Rich short text conversation using semantic-key-controlled sequence generation. IEEE/ACM Transactions on Audio, Speech and Language Processing, 2018, 26(8): 1359-1368

[32] Johnson M, Schuster M, Le Q V, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. Transactions of the Association for Computational Linguistics, 2017, 5: 339-351

[33] Sutton R, Barto A. Reinforcement Learning: An Introduction. 2nd Edition. Cambridge, USA: The MIT Press, 2018

[34] Bahdanau D, Brakel P, XU K, et al. An actor-critic algorithm for sequence prediction. Computing Research Repository, arXiv:1607.07086, 2016

[35] He D, Lu H, Xia Y, et al. Decoding with value networks for neural ma-chine translation//Proceedings of the Conference of Neural Information Processing Systems (NIPS). Long Beach, USA, 2017: 178-187

[36] Papineni K, Roukos S, Ward T, et al. BLEU: A method for automatic evaluation of machine translation//Proceedings of the Annual Meeting on Association for Computational Linguistics(ACL). Philadelphia, USA, 2002: 311-318

[37] Schmidt F, Mandt S, Hofmann T. Autoregressive text generation beyond feedback loops. Computing Research Repository, arXiv:1908.11658, 2019

[38] Forgy E. Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. Biometrics, 1995, 21: 768-780

[39] Stolcke A, Konig Y, Weintraub M. Explicit word error minimization in N-best list rescoring//Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH). Rhodes, Greece, 1997: 163-166

[40] Bellman R. A Markovian decision process. Indiana University Mathematics Journal, 1957, 6(4): 679-684

[41] Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation. Computing Research Repository, arXiv:1506.02438, 2015

[42] Williams R. Reinforcement-learning connectionist systems. Boston, USA: College of Computer Science, Northeastern University, Technical Report: NU-CCS-87-3, 1987.

[43] Williams R. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 1992, 8(3-4): 229-256

[44] Witten I. An adaptive optimal controller for discrete-time Markov environments. Information and Control, 1977, 34(4): 286-295

[45] Barto A, Sutton R, Anderson C. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics, 1983, 5: 834-846

[46] Sutton R. Temporal Credit Assignment in Reinforcement Learning [Ph. D. dissertation]. University of Massachusetts Amherst, Amherst, USA, 1984

[47] Abeillé A. The Penn Treebank: An Overview. Netherlands: Springer, 2003: 5-22

[48] Cieri C, Miller D, Walker K. The fisher corpus: A resource for the next generations of speech-to-text//Proceedings of the International Conference on Language Resources and Evaluation(LREC). Lisbon, Portugal, 2004: 69-71

[49] Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch//Proceedings of the Conference on Neural Information Processing Systems(NIPS). Long Beach, USA, 2017

[50] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding//Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). Minneapolis, USA, 2019: 4171-4186

LIU Qi, Ph. D. His research interests include speech recognition, language model, and reinforcement learning.

MA Rao, M. S. Her research interests include language model, large vocabulary continuous speech recognition, and speech synthesis.

YU Kai, Ph. D., professor. His research interest include speech recognition, synthesis, natural language understanding and dialogue management.

Background

This work was supported by the Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0102) and the State Key Lab of Media Convergence Production Technology and Systems (No. SKLMCPTS2020003).

This paper mainly focuses on two problems of natural language generation, exposure bias problem, and train-test criterion mismatch problem. The first way to address these problems is by using reinforcement learning to train the model. Some researchers have already trained natural language generation models with the policy gradient method. However, most of these models are about strong condition natural language generation especially machine translation. The second way is adding extra embedding to the model. At first, the extra embedding needs supervised training, which is only suitable for the dataset with the extra label. Some researchers try to get extra input in an unsupervised way. However, most of their work extracts extra embedding from the input text, which lacks prior information about the reference text.

This paper works on weak condition natural language generation. Weak condition natural language generate model has a much larger generation space which makes the model hard to train and converge. Many policy gradient models that are applied to strong condition natural language generation cannot converge on weak condition natural language generation. This paper proposed a way to successfully train the policy gradient model for weak condition natural language generation. Also, this paper proposed an unsupervised way to extract extra embedding from the reference text, which can provide more prior information compared with extra embedding extracted from the input text. This paper also found that the extra embedding is complementary to the policy gradient training, which can help the model converge more quickly.