一．An actor-critic algorithm for sequence prediction

本文提出使用 actor-critic 方法来进行句子生成，试图解决有监督文本生成 task，基于 MLE 的 seq2seq 模型中，由于训练和测试模型的差异（测试模型基于上一步生成的 token 而不是 ground truth）带来的限制。

1. 任务描述

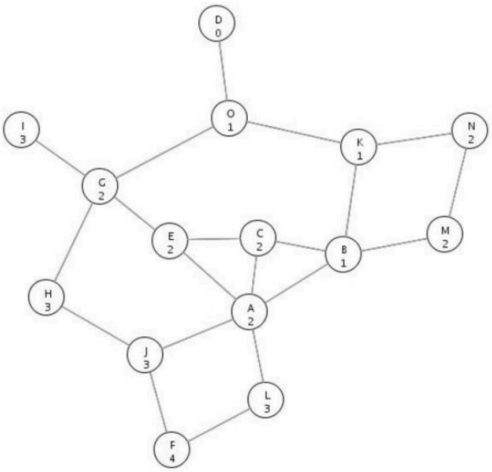
我们有两个集合的 input-output 对(X,Y)，分别作为训练集和测试集，训练好的预测模型 h 在测试集上产生对 Y 的预测 Y' = h(x)，通过评分 R(Y,Y')来评估预测模型 h 的好坏。

RNN 模型定义概率 p(Y)为：$p(y_1)p(y_2|y_1)\dots p(y_T|y_1,\dots,y_{T-1})p(\emptyset|y_1,\dots,y_T)$，基于 RNN 的通过输入序列预测输出序列模型称为 seq2seq，是一种 conditional RNN 模型。该模型可以通过对训练集最大 log 似然 log p(Y|X)来训练，此时每次预测的 y_t 都基于 X 和真实的前 t-1 个 y $p(*|y_1,\dots,y_{t-1})$；而在测试集产生第 t 个预测结果时，由于没有真实前 t-1 个 y 的 token，所以使用预测结果：$p(*|y_1',\dots,y_{t-1}')$。这会导致 exposure bias，即 decode 的误差不断累积、传递。

这个偏差是由于 decode 产生每个词的时候，只考虑了基于前面的序列选择最逼真的当前词。这个问题通常的解决方法有两种：

1. beam search

这种方法试图对整个生成的序列找到最优解，而不是只考虑当前词最优。是一种图路径搜索算法，搜索的过程中保存 beam size 个整体得分最高路径，剪掉其他路径，直到达到设定的步数 n。比如下图，生成从 F 开始的长度为 n 得分最高的路径。



在文本生成问题中，token 的分值即为概率值。在训练集中仍然使用 ground truth 作为 decoder 的输入，在测试集中，使用 beam search，相当于要产生使整个句子概率最大的最优解（贪心算法，局部最优），而非每一步都要选择概率最大的词，导致生成的句子可能偏差很大。

2. schedule sample

在训练过程中，早期主要使用 ground truth 作为 decoder 的输入，随着训练进行，逐渐更多的使用上一步产生的 token。这样即使前面生成错误，仍以生成正确的序列为训练目标产生后面的 token，增大了模型的容错能力。

使用强化学习解决能够这个问题，是由于训练模型时，每次选择的是使得整个后续的序列最好的 action（q value 最大），而非仅仅考虑当前词。此时训练集的输入为上一个产生的 token。

2. actor-critic for sequence prediction

套用强化学习的概念，我们将 conditional RNN 模型当做一个具有随机性的 policy，将与任务相关的评分函数 R(Y,Y')当做第一个 state 的 return，比如翻译任务中的 BLEU。我们将 R 分解为在中间的时间步获得的 reward：

$$\sum_{t=1}^{T} r_t(\hat{y}_t; \hat{Y}_{1\dots t-1}, Y)$$

我们将 state 定义为到当前时间步为止，采取的所有 action $Y'_{1\dots t-1}$，我们定义 state 的函数 value function（从当前 state 的 return 的期望）：

$$V(\hat{Y}_{1\dots t}; X, Y) = \mathop{\mathbb{E}}_{\hat{Y}_{t+1\dots T}\sim p(.|\hat{Y}_{1\dots t},X)} \sum_{\tau=t+1}^{T} r_\tau(\hat{y}_\tau; \hat{Y}_{1\dots\tau-1}, Y)$$

在当前 state 下采取某个候选 action 的 value function 为：

$$Q(a; \hat{Y}_{1\dots t-1}, X, Y) = \mathop{\mathbb{E}}_{\hat{Y}_{t+1\dots T}\sim p(.|\hat{Y}_{1\dots t-1}a,X)} \left(r_t(a; \hat{Y}_{1\dots t-1}, Y) + \sum_{\tau=t+1}^{T} r_\tau(\hat{y}_\tau; \hat{Y}_{1\dots t-1}a\hat{Y}_{t+1\dots\tau}, Y) \right)$$

则我们可以将初始状态的 value 作为 performance，对其求梯度：

$$\frac{dV}{d\theta} = \mathop{\mathbb{E}}_{\hat{Y}\sim p(\hat{Y}|X)} \sum_{t=1}^{T} \sum_{a\in\mathcal{A}} \frac{dp(a|\hat{Y}_{1\dots t-1})}{d\theta} Q(a; \hat{Y}_{1\dots t-1}).$$

我们将对 state 的期望替换为采样，这是对这个梯度的一个无偏估计

$$\widehat{\frac{dV}{d\theta}} = \sum_{k=1}^{M} \sum_{t=1}^{T} \sum_{a\in\mathcal{A}} \frac{dp(a|\hat{Y}^k_{1\dots t-1})}{d\theta} Q(a; \hat{Y}^k_{1\dots t-1})$$

注意到，对于 action，我们没有选择在对 action 采样，然后求 log 概率，而是直接在每一个 state 下对所有 action 进行梯度更新，对 Q value 大的 action 提高概率，Q 小的 action 降低概率（这样做可以避免由于 Q 都是正值，没有使用 baseline 造成的影响）。然后我们用一个网络来建模 Q，用基于 Q learning 的方法来学习这个网络，这会产生偏差（bias），但是比起 MC 方法来估计 q value（比如 REINFORCE），可以有效降低估计的方差。本文通过设计 cirtic network、使用适当的训练方法来控制 bias。

1) 本文使用了另外一个 RNN 来实现 critic 网络，这个网络和 actor 网络并行训练，输入的是 actor 产生的 token y_t，输出对所有下一个可能的 action 的评分 $\hat{Q}(a; \hat{Y}_{1\dots t})$；由于 return 是由估测序列和正确序列的一个确定性函数给出（eg.BLEU score），所以将正确的 sequence Y 也作为 critic 网络的输入（seq2seq），这能提升 critic 训练的效果。在测试阶段，我们不需要 critic 网络，只需要用 actor 来产生序列。

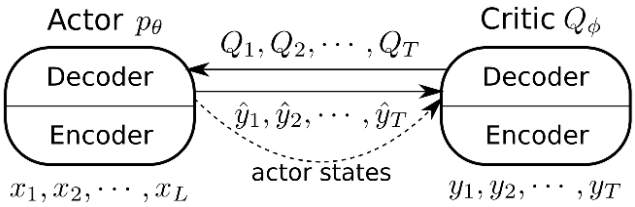
2) 对 critic 网络的训练，我们使用了时序差分（TD）法，将下式作为 $\hat{Q}(\hat{y}_t; \hat{Y}_{1\dots t-1})$ 的 target，然后利用 DQN 等方法进行训练：

$$q_t = r_t(\hat{y}_t; \hat{Y}_{1\dots t-1}) + \sum_{a\in A} p(a|\hat{Y}_{1\dots t}) \hat{Q}(a; \hat{Y}_{1\dots t})$$

3) 由于 action 空间很大，本文对很少被采样到的单词的 critic value 施加了约束，实验证明这可以促进算法收敛，具体来说，就是在每一步训练 critic 时，在优化目标中多加了一项，使 action 的 value 接近所有 action 的均值，以减小 critic 所有输出的方差：

$$C_t = \sum_a \left(\hat{Q}(a; \hat{Y}_{1\ldots t-1}) - \frac{1}{|\mathcal{A}|} \sum_b \hat{Q}(b; \hat{Y}_{1\ldots t-1}) \right)^2$$

4) reward shaping：由于 R 只给完整序列 Y 提供 reward，而没有每一步单独的 reward，对这种稀疏的 reward，可以使用 reward shaping 方法。具体来说



下面是整个算法的伪码，先对 actor 用 MLE 进行了预训练，然后固定 actor，对 critic 进行了预训练，最后进入算法 1。在算法 1 中，训练 actor 时加入了一个 log 似然梯度，单独使用 actor-critic 会导致学到确定性的 policy 和梯度消失。

Algorithm 1 Actor-Critic Training for Sequence Prediction

Require: A critic $\hat{Q}(a; \hat{Y}_{1\ldots t}, Y)$ and an actor $p(a|\hat{Y}_{1\ldots t}, X)$ with weights ϕ and θ respectively.

1: Initialize delayed actor $p^{'}$ and target critic $\hat{Q}'$ with same weights: $\theta' = \theta$, $\phi' = \phi$.

2: **while** Not Converged **do**

3: Receive a random example (X, Y).

4: Generate a sequence of actions $\hat{Y}$ from $p^{'}$.

5: Compute targets for the critic

$$q_t = r_t(\hat{y}_t; \hat{Y}_{1\ldots t-1}, Y) + \sum_{a \in \mathcal{A}} p^{'}(a|\hat{Y}_{1\ldots t}, X) \hat{Q}'(a; \hat{Y}_{1\ldots t}, Y)$$

6: Update the critic weights ϕ using the gradient

$$\frac{d}{d\phi} \left(\sum_{t=1}^{T} \left(\hat{Q}(\hat{y}_t; \hat{Y}_{1\ldots t-1}, Y) - q_t \right)^2 + \lambda_C C_t \right)$$

$$\text{where } C_t = \sum_a \left(\hat{Q}(a; \hat{Y}_{1\ldots t-1}) - \frac{1}{|\mathcal{A}|} \sum_b \hat{Q}(b; \hat{Y}_{1\ldots t-1}) \right)^2$$

7: Update actor weights θ using the following gradient estimate

$$\widehat{\frac{dV(X,Y)}{d\theta}} = \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} \frac{dp(a|\hat{Y}_{1\ldots t-1}, X)}{d\theta} \hat{Q}(a; \hat{Y}_{1\ldots t-1}, Y) + \lambda_{LL} \sum_{t=1}^{T} \frac{dp(y_t|Y_{1\ldots t-1}, X)}{d\theta}$$

8: Update delayed actor and target critic, with constants $\gamma_\theta \ll 1$, $\gamma_\phi \ll 1$

$$\theta' = \gamma_\theta \theta + (1 - \gamma_\theta)\theta', \; \phi' = \gamma_\phi \phi + (1 - \gamma_\phi)\phi'$$

9: **end while**

Algorithm 2 Complete Actor-Critic Algorithm for Sequence Prediction

1: Initialize critic $\hat{Q}(a; \hat{Y}_{1...t}, Y)$ and actor $p(a|\hat{Y}_{1...t}, X)$ with random weights ϕ and θ respectively.

2: Pre-train the actor to predict y_{t+1} given $Y_{1...t}$ by maximizing $\log p(y_{t+1}|Y_{1...t}, X)$.
3: Pre-train the critic to estimate Q by running Algorithm 1 with fixed actor.
4: Run Algorithm 1.

3. 实验

本文在两个 task 上进行了实验验证，一个是拼写纠正，即对于正确的句子，随机替换其中的一些单词，然后通过本模型来恢复，使用 character error rate（CER）来衡量模型的表现。另一个任务是机器翻译，使用法语-英语数据集 IWSLT2014 竞赛数据，使用 BLEU 评分。

二．SeqGAN

利用 GAN 来生成文本序列有两个问题：

1. GAN 被设计为 G 生成一个连续空间中的实数，而生成一系列的离散的 token 有困难。这是因为 G 是从一个随机噪音确定性的映射到一个实数 a 上，利用 D 的 loss 得到关于 a 的梯度来指导 G 模型的参数更新，而当 a 是离散的，D 的 loss 关于 a 的梯度不存在。
2. D 可以学习对一个完整序列给出评分，但是如果序列不断的生成新的部分，很难用一个统一的标准对已有的部分和未来的序列进行评分。

本文中将序列生成过程看成是一个顺序的决策过程，将 generator 看做 RL 中的 agent，state 看做目前已有的 tokens，action 看做是下一个 token。Policy $G_\theta(y_t|Y_{1:t-1})$是具有随机性的，而状态转移 P(s'|s,a)=1\0 是确定性的；同时我们还训练了一个判别模型$D_\phi(Y_{1:T})$，意味着完整序列 $Y_{1:T}$ 来自真正的自然语言序列的概率，该模型通过真实的序列数据和 G 生成序列来训练。本文用 RL 中的 policy gradient 算法（REINFORCE）训练 policy（G），其中 state-action value 通过 Monte Carlo search 得到，而没有用 critic 网络单独学习。

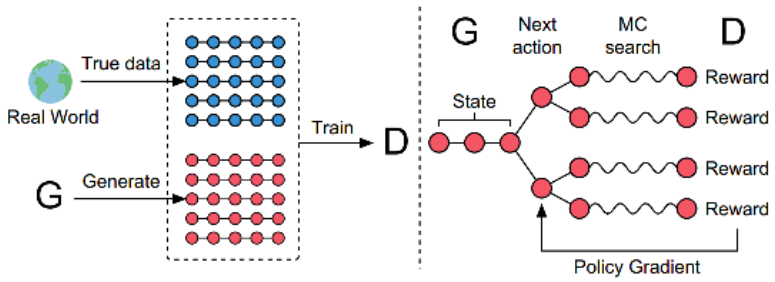


Figure 1: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

1. SeqGAN via policy gradient

SeqGAN 只对完整的 sequence（terminal state）有 reward R_T，这个 reward 由 discriminator D_Φ 给出，其他状态转移时，MDP 不产生 reward。当 policy 选择了 action y_t 时，下一个 state 是确定的，即 $Y_{1:t-1}$ 加上 y_t 组成的序列。依据 REINFORCE 算法，performance 定义为初始状态 s_0 的 value 值，在本情境中是最终 reward 的

期望：

$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1|s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1).$$

其中 $Q_{D_\phi}^{G_\theta}(s,a)$ 是 action value function。利用这个目标我们训练 θ 使得到的完整 sequence 尽可能被 discriminator 认为是真的。那我们如何得到 $Q_{D_\phi}^{G_\theta}(s,a)$ 呢，对于在 T-1 时间步的 state，我们有：

$$Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T})$$

而在其他时间步，标准的 REINFORCE 算法是使用每个 state-action 的 return 来训练，在本任务中由于其他时间步都没有 reward，同一个 episode 中所有的 state-action value 都是 $D_\Phi(Y_{1:T})$。当我们选取的 episode 足够多之后，每个 state-action value 都会被正确的估计，但是 nlp 的 state 和 action 可能的取值非常多，我们几乎不可能多次遇到一个 state-action 对。所以本文中使用了 Monte Carlo Search 方法，也就是对一个 state-action 对，用一个 roll-out policy G_β（本文中直接使用当前 policy G_θ）去多次采样后面的 token 直到达到 terminal state。

$$\left\{Y_{1:T}^1, \ldots, Y_{1:T}^N\right\} = \text{MC}^{G_\beta}(Y_{1:t}; N)$$

由于我们知道部分的环境模型，action value 等价于下一个 state 的 value：

Q(s,a) = E[G_t|s,a]

= E[r+G_{t+1}|s,a]

= E[0+G_{t+1}|s,a]

= E[G_{t+1}|s']

使用 MC 方法我们可以估计 action value：

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \tag{4}$$

$$\begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n),\ Y_{1:T}^n \in \text{MC}^{G_\beta}(Y_{1:t}; N) & \text{for} \quad t < T \\ D_\phi(Y_{1:t}) & \text{for} \quad t = T, \end{cases}$$

当我们训练使生成的 sequence 非常逼真时，可以重新训练 discriminator：

$$\min_\phi -\mathbb{E}_{Y \sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta}[\log(1 - D_\phi(Y))]$$

generator 的使用 REINFORCE 算法相同的 loss，梯度为：

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{Y_{1:t-1} \sim G_\theta}\Big[\sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)\Big]$$

$$\nabla_\theta J(\theta) \simeq \sum_{t=1}^{T} \sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t) \tag{7}$$

$$= \sum_{t=1}^{T} \sum_{y_t \in \mathcal{Y}} G_\theta(y_t|Y_{1:t-1}) \nabla_\theta \log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)$$

$$= \sum_{t=1}^{T} \mathbb{E}_{y_t \sim G_\theta(y_t|Y_{1:t-1})}[\nabla_\theta \log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)],$$

下面是 seqGAN 的伪代码：

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
    for t in $1 : T$ **do**
      Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
    end for
    Update generator parameters via policy gradient Eq. (8)
  end for
  for d-steps **do**
    Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator D_ϕ for k epochs by Eq. (5)
  end for
  $\beta \leftarrow \theta$
until SeqGAN converges

在训练开始时，先在训练集中用 MLE 预训练 generator，有监督信号能够有效的提升 generator。

2. Generative Model and discriminative model

使用了 RNN 作为生成模型，现将 w 词表进行 embedding，然后作为 RNN 输入，得到一系列的 hidden state，然后将其通过 softmax 得到下一步选择每个 token 的概率，实验中使用了 LSTM 模型，以 start_token 作为起始的输入，h0、s0 置为 0：

$$\boldsymbol{h}_t = g(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

$$p(y_t|x_1, \ldots, x_t) = z(\boldsymbol{h}_t) = \text{softmax}(\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}_t)$$

判别模型使用了 CNN，大部分的判别模型只在完整的句子里有较好的分类效果，所以本文也只基于完整序列做分类，将序列中每个 word 的 embedding 组成一个矩阵，再用 CNN 做分类。

3. 实验

在描述实验之前，我们先介绍一些在文本生成任务中常用的评价标准：

BLEU score：

机器翻译的常用评价方法，计算生成译文中存在在正确译文的 n 元组占生成译文所有 n 元组的比例 p_n，这个值越大，说明翻译越接近真实；n 越大说明翻译的越流畅，所以对所有的 n 进行加权求和：

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n logP_n)$$

其中 BP（brevity penalty）是一个对生成译文较短的时候施加的惩罚项，c 为生成译文的长度，r 为真实译文的长度：

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

perplexity：

等同于真实文本和生成文本的交叉熵（相当于真实文本在训练得到的模型里的负似然对数）perplexity 是负似然对数的指数：

$$H(W) = -\frac{1}{N} logP(w_1 w_2 \cdots w_N)$$

$$\begin{aligned} \text{Perplexity}(W) &= 2^{H(W)} \\ &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \\ &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}} \end{aligned}$$

进行了两个实验，一个在合成数据集上，本文提出了一种新的指标；一个在真实数据上，借鉴 BLUE score，设置元组长度为 2\3，比较 BLUE-2 score：

1) 合成数据实验

数据：随机初始化了一个 lstm，利用其随机生成数据。由于我们有待拟合的模型，可以准确的评估生成模型的表现。

评估标准：我们知道最大 log 似然评估相当于最小化真实数据分布 p 与生成的近似分布 q 的交叉熵 $E_{x\sim p}\log q(x)$，虽然我们没有 p，但是真实数据可以看做来自 p 的采样。然而，对于生成模型来说，最好的评估方法是用模型生成一些样本，让人来分辨是否正确，这相当于使用相反的负似然对数 $-\mathbb{E}_{x\sim q} \log p_{\text{human}}(x)$

在这个实验的设置中，产生数据的 lstm 模型就可以充当人的作用，评估标准为：

$$\text{NLL}_{\text{oracle}} = -\mathbb{E}_{Y_{1:T}\sim G_\theta}\left[\sum_{t=1}^{T} \log G_{\text{oracle}}(y_t | Y_{1:t-1})\right]$$

对比实验：1. 随机生成 token 2. MLE trained LSTM 3. Scheduled sampling 4.Policy gradient with BLEU（MC search）

2) 真实数据集

在真实的中国诗歌和奥巴马演讲数据集上训练生成模型，使用了 BLEU 评分。

三．Adversarial Learning for Neural Dialogue Generation

在 seqGAN 中，为了估计每一个 state-action 的 Q value，使用了 MC search 的方法，代价是计算量很高、很耗时；本文提出了另一种代替方法。

本文的 task 是对话生成，我们有一系列的对话的语句 x，模型要生成对应的回答 $y=\{y_1,y_2,\ldots y_T\}$。将生成语句的过程看做是一系列 action，而 policy（generator）的模型是一个 encoder-decoder RNN（seq2seq）；discriminator 则用来判断对话是由机器产生还是真正的对话，这个概率值作为 reward。

Generator 的同样使用了 REINFORCE 算法，loss 和 seqGAN 相同，除了需要将 x 作为输入：

$$J(\theta) = \mathbb{E}_{y\sim p(y|x)}(Q_+(\{x, y\})|\theta)$$

这里是对整个序列得到的 loss，由于部分环境模型已知，这是一种比较简便

的写法，推导如下：

$J(\theta) = E[G_0|s_0] = \sum_{G0} P(G_0|s_0)\, G_0 = \sum_{G0}\sum_{\tau} P(G_0|\tau)P(\tau|s_0)\, G_0$

由于序列生成过程中 reward 不具有随机性，确定序列 τ 后，return G_0 由函数 Q(τ)给出：$\sum_{G0}P(G_0|\tau)G_0 = Q(\tau)$，故 $J(\theta) = \sum_{\tau} P(\tau|s_0)\, Q(\tau)$。

$$\nabla J(\theta) \approx [Q_+(\{x,y\}) - b(\{x,y\})] \nabla \log \pi(y|x)$$
$$= [Q_+(\{x,y\}) - b(\{x,y\})] \nabla \sum_t \log p(y_t|x, y_{1:t-1})$$

注意到这里和 seqGAN 略微不同的是使用了 baseline，来保证估计是无偏的。现在同样存在，经过一次采样后，对一个序列中每一次的 action 都使用了同一个 reward 来进行梯度下降。为了估计更加准确，本文提出了一种为每个时间步求 reward 的方法：训练判别器不只能判断完整序列的真假，也同样能判断部分序列的真假。具体做法为，对真的对话和生成的对话，采样一个截断的长度（也可能采样到完整的序列）作为训练集。但是这样判别器的精度会差一些，但是减少了时间消耗。此时梯度为：

$$\nabla J(\theta) \approx \sum_t (Q_+(x, Y_t) - b(x, Y_t)) \nabla \log p(y_t|x, Y_{1:t-1})$$

最后，和 seqGAN 一样，本模型也会面临 generator 学不到信息的问题，这是因为 G 是用一种间接的方式受 D 引导更新梯度，而 D 只对好的对话序列有正向的 reward，G 随机产生各种序列，其中绝大部分是不好的，所以没有梯度信息可以更新。为了解决这个问题，RL 中存在一种 behavior cloning 的方法，也就是将人为的数据拿来给 policy 学习（实际上就是有监督学习）。SeqGAN 在一开始先进行 MLE 预训练；而本文则选择了交替用生成的数据和真实数据训练 generator（认为真实数据 reward 为 1，实质上仍然是 MLE）。最终算法如下：

For number of training iterations **do**
. **For** i=1,D-steps **do**
. Sample (X,Y) from real data
. Sample $\hat{Y} \sim G(\cdot|X)$
. Update D using (X, Y) as positive examples and $(X, \hat{Y})$ as negative examples.
. **End**
.
. **For** i=1,G-steps **do**
. Sample (X,Y) from real data
. Sample $\hat{Y} \sim G(\cdot|X)$
. Compute Reward r for $(X, \hat{Y})$ using D.
. Update G on $(X, \hat{Y})$ using reward r
. Teacher-Forcing: Update G on (X, Y)
. **End**
End

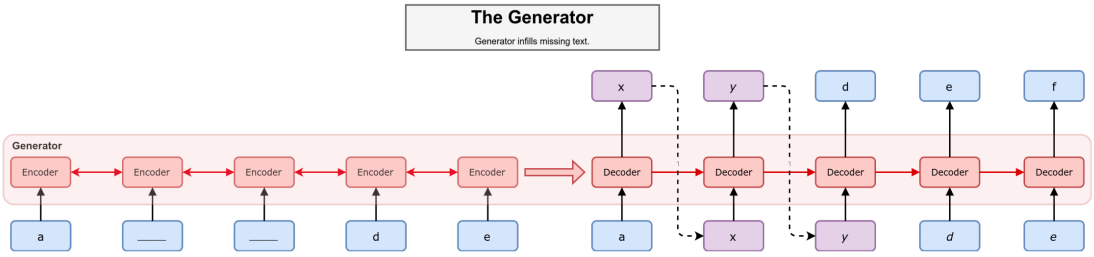
四．MaskGAN

本文的贡献在于 1.使用了 actor-critic 方法，在每个时间步都可以提供 reward，而不需要等到一个序列结束后才能得到。2.针对 seqGAN 可能出现的 mode collapse（产生大量重复样本）、训练不稳定等问题，本文没有在训练中以文本生

成为目的，而是在句子中挖空让模型学会填写，generator 从 rnn 改为 seq2seq。

1. 模型

先介绍模型， 对于离散序列 x，随机选择其中的一些位置，将 token 替换为 <m>，替换后的序列为 m(x)。Generator G 使用 seq2seq 结构，encoder 将 m(x)作为输入，这样可以为预测 missing token 提供未来 token 的信息；decoder 能够得到被挖空的各个 token。注意的一点是，decoder 每个单元的输入并不一定是上一个单元的输出，如果上一个单元并不是被挖空的 token，那输入的是真实序列中的 token。



discriminator D 使用了和 G 相同的结构 seq2seq，输入为 G 输出的填好空的序列，此外还要将 m(x)作为 D 的输入（为了让 D 知道哪些位置是真实数据，哪些是挖空），每个单元输出的是一个标量，即该位置预测是否正确的概率值。生成器和判别器公式如下：

$$P(\hat{x}_1, \cdots, \hat{x}_T | \boldsymbol{m}(\boldsymbol{x})) = \prod_{t=1}^{T} P(\hat{x}_t | \hat{x}_1, \cdots, \hat{x}_{t-1}, \boldsymbol{m}(\boldsymbol{x})).$$

$$G(x_t) \equiv P(\hat{x}_t | \hat{x}_1, \cdots, \hat{x}_{t-1}, \boldsymbol{m}(\boldsymbol{x}))$$

$$D_\phi(\tilde{x}_t | \tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x})) = P(\tilde{x}_t = x_t^{\text{real}} | \tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x}))$$

reward 设置为 D 预测值的 log 函数：

$$r_t \equiv \log D_\phi(\tilde{x}_t | \tilde{x}_{0:T}, \boldsymbol{m}(\boldsymbol{x}))$$

本文设置的第三个网络是一个 critic 网络，评估 value function：

$$R_t = \sum_{s=t}^{T} \gamma^s r_s$$

cirtic 网络使用了和 discriminator 的 encoder 部分相同的结构，并且在训练中与 D 共享参数，rnn 的每一步产生一个标量，即为每一个 state 得到 value 值。

2. maskGAN via actor-critic

为了克服采样操作造成的模型不可微，梯度无法传播回生成器问题，本文同样使用了 REINFORCE 算法，同时将 cirtic 网络得到的 state 的 value 用作 baseline：

$$b_t = V^G(x_{1:t})$$

$$\nabla_\theta \mathbb{E}_G[R_t] = (R_t - b_t) \nabla_\theta \log G_\theta(\hat{x}_t)$$

这样，本文巧妙的构造了为每一次 token 生成提供 reward 的方法，使得每次生成 token 都会考虑对后续生成的序列影响。完整的生成器梯度为：

$$\begin{aligned} \nabla_\theta \mathbb{E}[R] &= \mathbb{E}_{\hat{x}_t \sim G} \left[\sum_{t=1}^{T} (R_t - b_t) \nabla_\theta \log(G_\theta(\hat{x}_t)) \right] \\ &= \mathbb{E}_{\hat{x}_t \sim G} \left[\sum_{t=1}^{T} \left(\sum_{s=t}^{T} \gamma^s r_s - b_t \right) \nabla_\theta \log(G_\theta(\hat{x}_t)) \right] \end{aligned}$$

而类似传统的 GAN 训练，判别器按如下梯度更新：

$$\nabla_{\phi}\frac{1}{m}\sum_{i=1}^{m}\left[\log D(x^{(i)})]+\log(1-D(G(z^{(i)})\right]$$

通过分析源码，我们得知判别器只在缺失的 token 上进行训练（正确的 token\生成的 token），生成器只在缺失的 token 得到 reward 上计算 return。

critic 网络基于 MC 方法进行训练，loss 是预测的 value 与每一次真实的 return 的平方和误差。最后，注意到虽然用了 critic 网络，本文实际上应用的是 policy gradient 的方法，无偏，但是方差大。

文本生成任务存在一个难点，即长序列文本生成，本文提出的 trick 是先设定一个较小的最大序列长度 T，先在较短的序列上进行训练，模型收敛后，增加最大序列长度到 T+1，这有点像 curriculum learning 的思想，让 agent 从简单任务学起。

最后，本模型同样进行了预训练，首先训练了一个语言模型，然后用语言模型的参数作为 seq2seq 的参数（包括 G，D 和 critic 网络），对补全任务进行了最大似然训练，在此的基础上，进行了基于 GAN 和 RL 的训练。

3. 实验

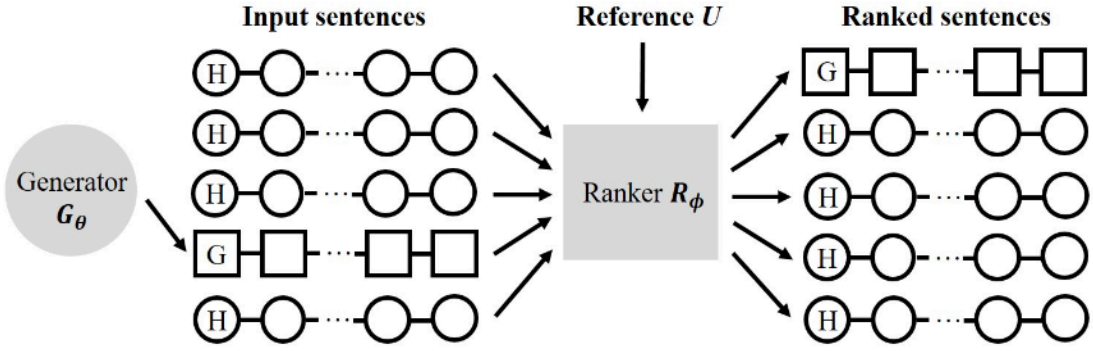
通过 GAN 来生成文本，比起通过最大似然得到的模型，在 perplexity 指标上是表现更差的。但是本文认为 perplexity 并不能代表生成质量，真实的句子在模型上的低似然概率，并不代表模型生成的句子的真实度低。所以本文通过人工评价来衡量效果：

数据集：The penn Treebank（PTB），IMDB MOVIE DATESET

实验：conditional sample（填空），Unconditional sample（language model），

五．RankGAN

本文认为，传统 GAN 中的判别器只给生成器一个是否为真的概率值，包含的信息量是非常有限的。本文用一个 ranker 代替了 discriminator，通过排序可以更好的学到句子之间的差异。



排序是在信息检索领域常用的技术。我们有三个集合，reference 由真实文本组成，另外存在两个集合：生成文本 C-，真实文本 C+。输入的一组文本（包含真实和生成文本），我们可以用与 reference 的相似度为这组文本排序，与 reference 中的文本越相似，则得分越高。具体来说，给定一条 reference 的文本 u，文本 s 的相关分数为：

$$\alpha(s|u)=cosine(y_s,y_u)=\frac{y_s\cdot y_u}{\|y_s\|\|y_u\|}$$

类比 softmax 函数，我们可以得到在集合 C 中，s 的分数：

$$P(s|u,\mathcal{C}) = \frac{exp(\gamma\alpha(s|u))}{\sum_{s'\in\mathcal{C}'} exp(\gamma\alpha(s'|u))}$$

则对整个 reference 文本，s 在 C 的 ranking score 由对不同的 u 的期望给出：

$$\log R_\phi(s|U,\mathcal{C}) = \mathop{\mathbb{E}}_{u\in U} \log\left[P(s|u,\mathcal{C})\right]$$

当 s 是生成文本时，我们使用集合 C+，然后训练生成器使 s 的分数增加；反之亦然，这样整体对抗训练的目标为：

$$\min_\theta \max_\phi \mathfrak{L}(G_\theta, R_\phi) = \mathop{\mathbb{E}}_{s\sim\mathcal{P}_h}\left[\log R_\phi(s|U,\mathcal{C}^-)\right] + \mathop{\mathbb{E}}_{s\sim G_\theta}\left[\log(1 - R_\phi(s|U,\mathcal{C}^+))\right]$$

六．LeakGAN

基于 GAN 进行文本生成的难点在于 reward 非常稀疏，只对整个句子有一个反馈，在生成每个 token 过程中没有 reward 指导生成的方向，在生成长文本时这个问题尤为明显。

在传统的 rl 问题中，reward 来自于环境反馈，是一个黑箱。然而在用 GAN 产生 reward 时，我们知道产生 reward 的具体网络结构，所以我们可以让 D 泄露一些信息，用来指导生成的方向。具体来说，使用了 hierarchical rl，将生成器分为两个层次，两个 agent 分别称为 manager 和 worker，高阶的 manager 从 D 中接收高维特征表示，同时利用 D 的 reward 去学习应该在 state 空间中探索的方向，形成一个指导目标，作用于 worker 模块。

1. 泄露的特征表示

判别器 D_Φ 包括两个部分，一个是特征提取的部分$\mathcal{F}(\cdot;\phi_f)$，一个是最后的 sigmoid 分类层，参数为 Φ_l。

$$D_\phi(s) = \text{sigmoid}(\phi_l^\top \mathcal{F}(s;\phi_f)) = \text{sigmoid}(\phi_l^\top f)$$

f 就是 state s 的特征向量，由于最后的 reward 值完全取决于 f，那么在 state 空间中找到 reward 更大的区域，相当于在 f 的特征空间找到 reward 更大的区域：

$\mathcal{F}(\boldsymbol{S};\phi_f) = \{\mathcal{F}(s;\phi_f)\}_{s\in\boldsymbol{S}}$.

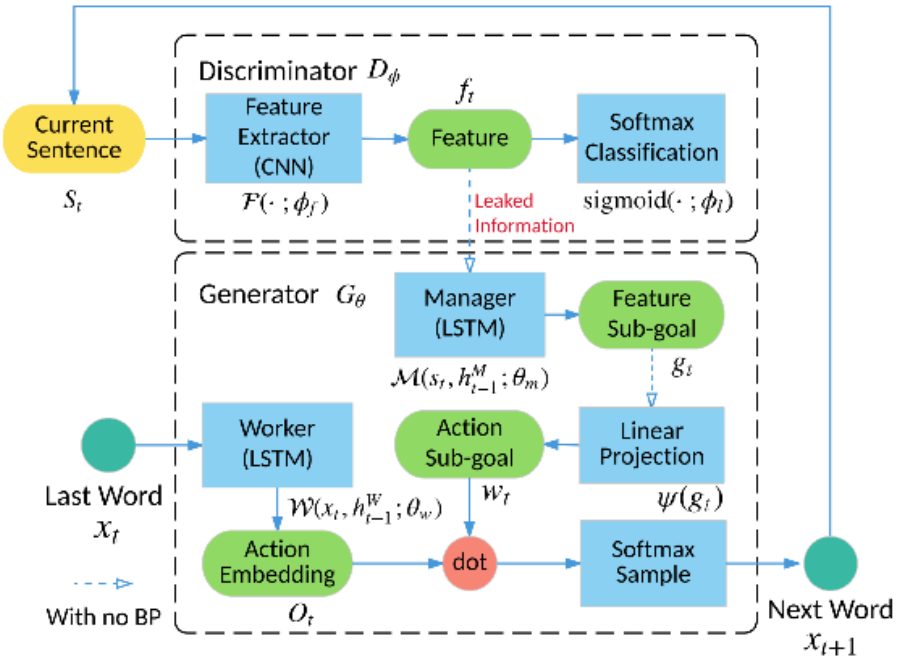
所以比起一个标量的 reward 信号，f 是一个富含更多信息的信号，可以指导生成的方向。

2. G 的层次化结构

$$\hat{g}_t, h_t^M = \mathcal{M}(f_t, h_{t-1}^M; \theta_m)$$
$$g_t = \hat{g}_t / \|\hat{g}_t\|,$$

$$w_t = \psi\Big(\sum_{i=1}^{c} g_{t-i}\Big) = W_\psi\Big(\sum_{i=1}^{c} g_{t-i}\Big)$$

$$O_t, h_t^W = \mathcal{W}(x_t, h_{t-1}^W; \theta_w),$$
$$G_\theta(\cdot|s_t) = \text{softmax}(O_t \cdot w_t/\alpha)$$



3. G 的训练

注意到，在方框中 G 的结构是完全可微的，可以用 REINFORCE 等方法进行端到端的训练，但是为了使 manager 捕捉到更多有用的信息，本文分开训练了 manager 和 worker。Manager 的梯度更新为：

$$\nabla_{\theta_m}^{\text{adv}} g_t = -Q_{\mathcal{F}}(s_t, g_t)\nabla_{\theta_m} d_{\cos}\Big(f_{t+c} - f_t, g_t(\theta_m)\Big)$$

其中 Q 是当前 policy 下 reward 的期望：

$$Q_{\mathcal{F}}(s_t, g_t) = Q(\mathcal{F}(s_t), g_t) = Q(f_t, g_t) = \mathbb{E}[r_t]$$

其中 f_{t+c}-f_t 是经过 t 步后，特征向量的改变，g_t 是目标方向，也就是当前 state 下，在 f 空间中应该改变的量。这个 loss 的目的是，在 f 的特征空间中，让目标方向 g_t 更加 match 使 reward 增加的方向。

Worker 使用 REINFORCE 算法：

$$\nabla_{\theta_w}\mathbb{E}_{s_{t-1}\sim G}\Big[\sum_{x_t} r_t^I \mathcal{W}(x_t|s_{t-1};\theta_w)\Big]$$

$$=\mathbb{E}_{s_{t-1}\sim G, x_t\sim\mathcal{W}(x_t|s_{t-1})}[r_t^I \nabla_{\theta_w}\log\mathcal{W}(x_t|s_{t-1};\theta_w)]$$

worker 的学习方向由 manager 提供：

$$r_t^I = \frac{1}{c}\sum_{i=1}^{c} d_{\cos}\Big(f_t - f_{t-i}, g_{t-i}\Big)$$