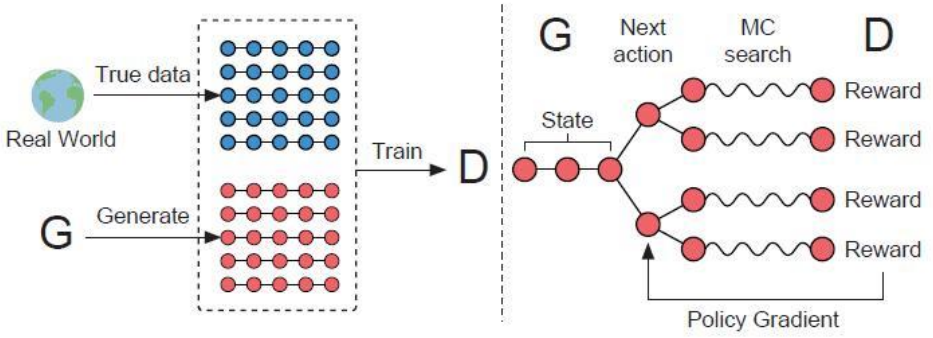


Role of RL in Text Generation by GAN(强化学习在生成对抗网络文本生成中扮演的角色)

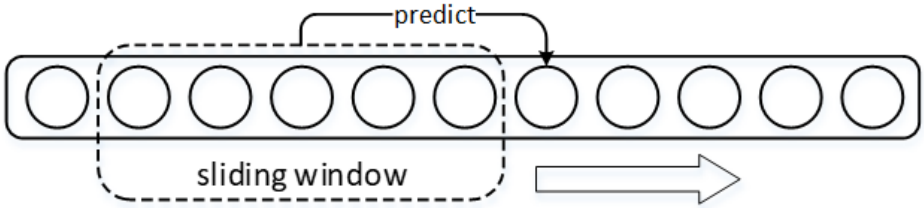
胡杨
华南理工大学 机器学习组 人工智能中医工程技术研究中心

Author: SCUT 胡杨

1. 基础：文本生成模型的标准框架

文本生成（Text Generation）通过 **机器学习 + 自然语言处理** 技术尝试使AI具有人类水平的语言表达能力，从一定程度上能够反应现今自然语言处理的发展水平。

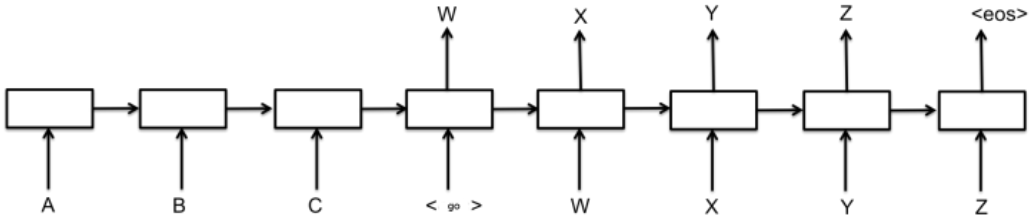
下面用极简的描述介绍一下文本生成技术的大体框架，具体可以参阅各种网络文献（比如：CSDN经典Blog“好玩的文本生成”[1]），论文等。



文本生成按任务来说，比较流行的有：机器翻译、句子生成、对话生成等，本文着重讨论后面两种。基于深度学习的Text Generator 通常使用循环神经网络（Basic RNN，LSTM，GRU等）进行语义建模。在句子生成任务中，一种常见的应用：“Char-RNN”（这里“Char”是广义上的称谓，可以泛指一个字符、单词或其他文本粒度单位），虽然简单基础但可以清晰度反应句子生成的运行流程，首先需要建立一个词库**Vocab**包含可能出现的所有字符或是词汇，每次模型将预测得到句子中下一个将出现的词汇，要知道softmax输出的只是一个概率分布，其维度为词库 **Vocab** 的 size，需再通过函数将输出概率分布转化为 **One-hot** vector，从词库 **Vocab** 中检索得出对应的词项；在“Char-RNN”模型训练时，使用窗口在语料上滑动，窗口之内的上下文及其后紧跟的字符配合分别为一组训练样本和标签，每次以按照固定的步长滑动窗口以得出全部 “样本-标签” 对。

与句子生成任务类似，对话生成以每组Dialogue作为 “样本-标签” 对，循环神经网络**RNN_1**对

Dialogue上文进行编码，再用另一个循环神经网络**RNN_2**对其进行逐词解码，并以上一个解码神经元的输出作为下一个解码神经元的输入，生成Dialogue下文，需要注意的是：在解码前需配置**“开始”**标记 _，用于指示解码器Decoder开启Dialogue下文首词（or 字）的生成，并配置**“结束”**标记 _，用于指示解码器结束当前的 Text Generation 进程。

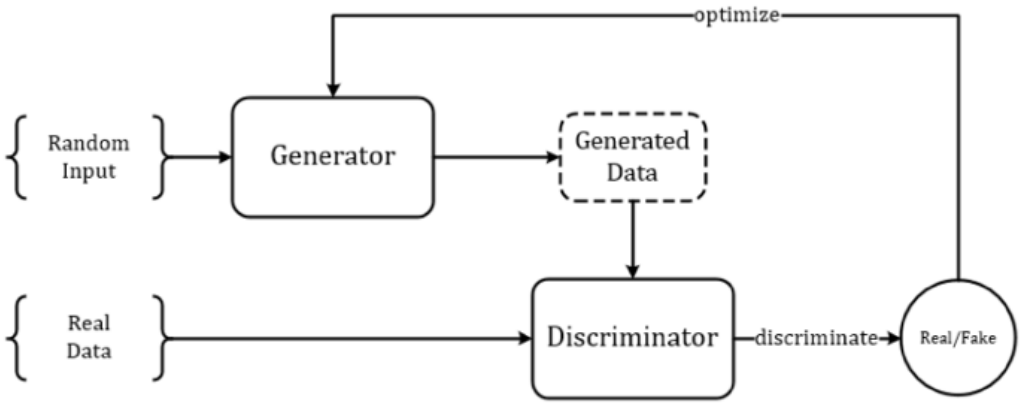


这便是众所周知的“Seq2Seq”框架的基础形态，为了提高基础Seq2Seq模型的效果，直接从解码器的角度有诸如 **Beam-Search**Decoder[2]、**Attention mechanism** Decoder[3]（配置注意力机制的解码器）等改进，而从神经网络的结构入手，也有诸如Pyramidal RNN[4]（金字塔型RNN）、Hierarchical RNN Encoder[5]（分层循环网络编码器）等改进。改进不计其数，不一一详举，但不管如何，预测结果的输出始终都是一个维度为词库大小的概率分布，需要再甄选出最大值的Index，到词库**Vocab**中检索得出对应的单词（or 字符）。

2. 问题：GAN为何不能直接用于文本生成

2.1. GAN基础知识

GAN对于大家而言想必已经脍炙人口了，这里做一些简单的复习。GAN从结构上来讲巧妙而简单（尽管有与其他经典工作Idea相似的争议[6~7]），也非常易于理解，整个模型只有两个部件：1.生成器G；2.判别器D。生成模型其实由来已久，所以生成器也并不新鲜，生成器G的目标是生成出最接近于真实样本的假样本分布，在以前没有判别器D的时候，生成器的训练依靠每轮迭代返回当前生成样本与真实样本的差异（把这个差异转化成loss）来进行参数优化，而判别器D的出现改变了这一点，判别器D的目标是尽可能准确地辨别生成样本和真实样本，而这时生成器G的训练目标就由最小化“生成-真实样本差异”变为了尽量弱化判别器D的辨别能力（这时候训练的目标函数中包含了判别器D的输出）。GAN模型的大体框架如下图所示：



我们再来简单复习一下GAN当中的一些重要公式，这一步对后文的阐述非常重要。不管生成器**G**是什么形状、多么深的一个神经网络，我们暂且把它看成一个函数 $G(\cdot)$ ，由它生成的样本记

作： $p_g(x)$ ，相对地，真实样本记作： $p_{data}(x)$ 。同样，不管判别器D作为一个分类神经网络，我们也可将其视为一个函数 $D(\cdot)$ ，而这个函数的输出即为一个标量，用于描述生成样本 $p_g(x)$ 与真实样本 $p_{data}(x)$ 之间的差距。

而GAN模型的整体优化目标函数是：

$$\arg\min_G \max_D = V(G,D)$$

其中函数 $V(G,D)$ 如下：

$$V = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{x\sim p_g}[\log(1-D(x))]$$

根据连续函数的期望计算方法，上式变形为：

$$V = \int_x [p_{data}(x)\log D(x) + p_g(x) log(1-D(x))]$$

先求外层的 $\arg\max_D V(G,D)$ 的话，对积分符号内的多项式求导取极值得到目标D：

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$$

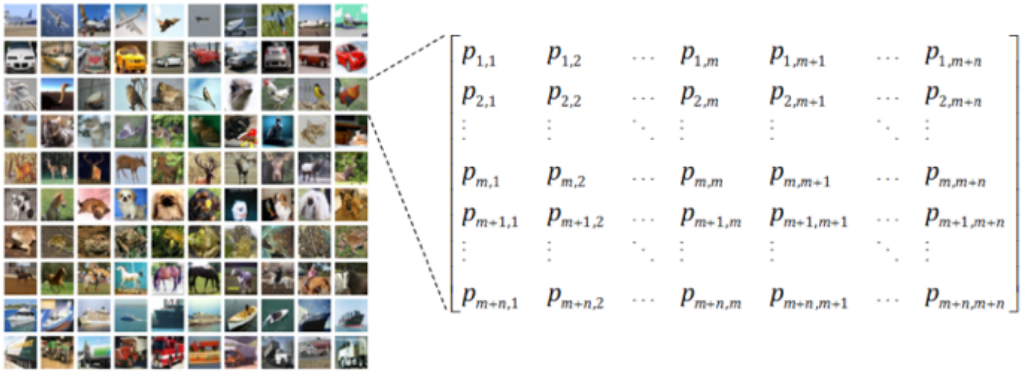
代回原式：

$$\begin{aligned} V(G,D^*) &= \int_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)} + \int_x p_g \log\frac{p_g(x)}{p_{data}(x)+p_g(x)} \\ &= -\log 4 + KL\left(p_{data}(x)\|\frac{p_{data}(x)+p_g(x)}{2}\right) + KL\left(p_g(x)\|\frac{p_{data}(x)+p_g(x)}{2}\right) \\ &= -\log 4 + 2\cdot JSD(p_{data}(x)\|p_g(x)) \end{aligned}$$

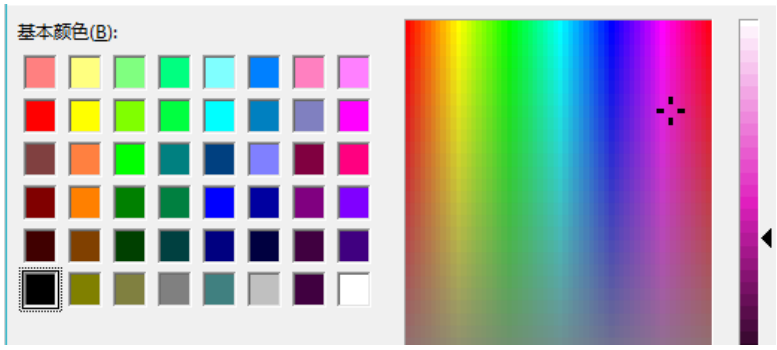
所以，当生成器G能生成出与真实样本一样分布的样本，那么ok，就达到最好的结果，然后大家注意一点，这里生成样本的loss衡量方法是JS散度。

2.2. GAN面对离散型数据时的困境（啥是离散型数据？）

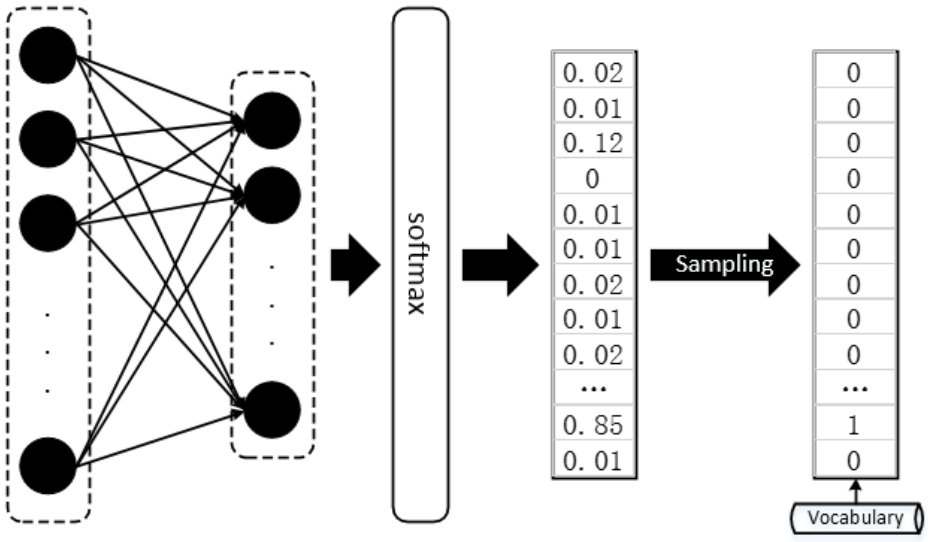
GAN的作者早在原版论文[8]时就提及，GAN只适用于连续型数据的生成，对于离散型数据效果不佳（使得一时风头无两的GAN在NLP领域一直无法超越生成模型的另一大佬VAE[9]）。文本数据就是最典型的一种离散型数据，这里所谓的离散，并不是指：文本由一个词一个词组成，或是说当今最流行的文本生成框架，诸如Seq2Seq，也都是逐词（或者逐个Character）生成的。因为哪怕利用非循环网路进行一次成型的Sentences生成，也无法避免“数据离散”带来的后果，抱歉都怪我年轻时的无知，离散型数据的真正含义，我们要从连续性数据说起。 图像数据就是典型的连续性数据，故而GAN能够直接生成出逼真的画面来。我们首先来看看图像数据的形状：



图像数据在计算机中均被表示为矩阵，若是黑白图像矩阵中元素的值即为像素值或者灰度值（抱歉外行了，我不是做图像的），就算是彩色图像，图像张量即被多加了一阶用于表示RGB通道，图像矩阵中的元素是可微分的，其数值直接反映出图像本身的明暗，色彩等因素，很多这样的像素点组合在一起，就形成了图像，也就是说，从图像矩阵到图像，不需要“采样”（Sampling），有一个更形象的例子：画图软件中的调色板，如下图，你在调色板上随便滑动一下，大致感受一下图像数据可微分的特性。



文本数据可就不一样了，做文本的同学都知道，假设我们的词库（Vocabulary）大小为1000，那么每当我们预测下一个出现的词时，理应得到的是一个One-hot的Vector，这个Vector中有999项是0，只有一项是1，而这一项就代表词库中的某个词。然而，真正的隔阂在于，我们每次用无论什么分类器或者神经网络得到的直接结果，都是一个1000维的概率分布，而非正正好好是一个One-hot的Vector，即便是使用softmax作为输出，顶多也只能得到某一维上特别大，其余维上特别小的情况，而将这种输出结果过渡到One-hot vector 然后再从词库中查询出对应index的词，这样的操作被称为“Sampling”，通常，我们找出值最大的那一项设其为1，其余为0。



当前神经网络的优化方法大多数都是基于梯度的（Gradient based），很多文献这么说：GAN在面对离散型数据时，判别网络无法将梯度Back propagation（BP）给生成网络。这句话当时让我等听的云里雾里，不妨换一个角度理解，我们知道，基于梯度的优化方法大致意思是这样的，微调网络中的参数（weight），看看最终输出的结果有没有变得好一点，有没有达到最好的情形。

但是判别器**D**得到的是Sampling之后的结果，也就是说，我们经过参数微调之后，即便softmax的输出优化了一点点，比如上图的例子中，正确结果本应是第三项，其output的倒数第二项从 0.85 变为了 0.65 ，第三项从 0.12 变为了 0.32 ，但是经过Sampling之后，生成器**G**输出的结果还是跟以前一模一样，并再次将相同的答案重复输入给判别器**D**，这样判别器**D**给出的评价就会毫无意义，生成器**G**的训练也会失去方向。

有人说，与其这样不如每次给判别器**D**直接吃Sampling之前的结果，也就是softamx输出的那个distribution，同样，这么做也有很大的问题。我们回到GAN的基本原理，判别器**D**的初衷，它经历训练就是为了准确辨别生成样本和真实样本的，那么生成样本是一个充满了float小数的分布，而真实样本是一个One-hot Vector，判别器**D**很容易“作弊”，它根本不用去判断生成分布是否与真实分布更加接近，它只需要识别出给到的分布是不是除了一项是 1 ，其余都是 0 就可以了。所以无论Sampling之前的分布无论多么接近于真实的One-hot Vector，只要它依然是一个概率分布，都可以被判别器**D**轻易地检测出来。

上面所说的原因当然也有数学上的解释，还记得在**2.1**节的时候，请大家注意生成样本的loss衡量标准是什么吗？没错，就是JS散度， $JS-divergence$ 在应用上其实是有弱点的（参考文献[10]），它只能被正常地应用于互有重叠（Overlap）的两个分布，当面对互不重叠的两个分布 P 和 Q ，其JS散度：

$$JSD(P\|Q) \equiv \log 2$$

大家再想想，除非softmax能output出与真实样本 exactly 相同的独热分布（One-hot Vector）（当然这是不可能的），还有什么能让生成样本的分布与真实样本的分布发生重叠呢？于是，生成器无论怎么做基于Gradient 的优化，输出分布与真实分布的 $JSD(p_{data}\|p_g)$ 始终是 $\log 2$ ，生成器**G**的训练于是失去了意义。

3. 过渡方案：对于GAN的直接改进用于文本生成

为了解决GAN在面对离散数据时的困境，最直接的想法是对GAN内部的一些计算方式进行微调，这种对于GAN内部计算方式的直接改进也显示出了一定的效果，为后面将GAN直接、流畅地应用于文本等离散型数据的生成带来了希望。 接下来简单介绍相关的两篇工作[11~12]。

3.1. Wasserstein-divergence，额外的礼物

Wasserstein GAN[13]（简称WGAN），其影响力似乎达到了原版GAN的高度，在国内也有一篇与其影响力相当的博文——“令人拍案叫绝的Wasserstein GAN”[10]，不过在看这篇论文之前，还要推荐另外一篇论文“*f-GAN*”[14]，这篇论文利用芬切尔共轭（Fenchel Conjugate）的性质证明了任何 $f-Divergence$ 都可以作为原先GAN中 $KL-Divergence$ （或者说 $JS-Divergence$ ）的替代方案。 *f-GAN* 的定义如下：

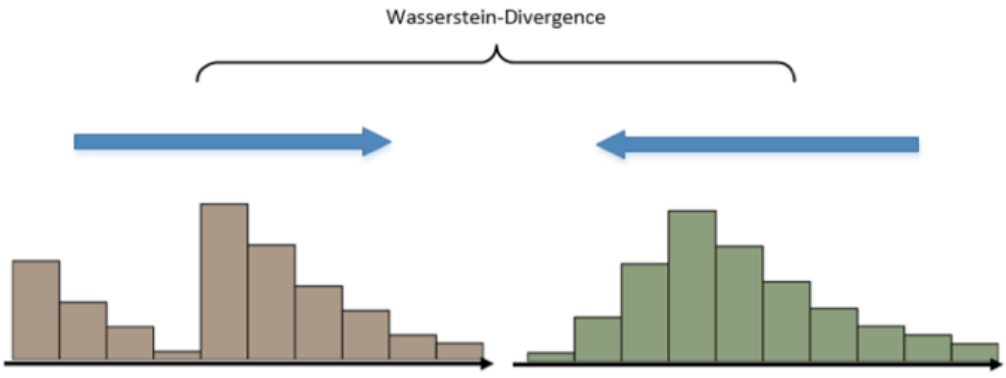
$$D_f(P\|Q)=\int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx$$

公式中的 $f(\cdot)$ 被称为 **f函数**，它必须满足以下要求：

$$\begin{cases} f\ is\ convex & \text{(凸函数)} \\ f(1)=0 \end{cases}$$

不难看出， $KL-Divergence$ 也是 $f-Divergence$ 的一种，*f-GAN* 原文提供了数十种各式各样的 $f-Divergence$ ，为GAN接下来沿此方向上的改进带来了无限可能。

Wasserstein GAN 对GAN的改进也是从替换 $KL-Divergence$ 这个角度对GAN进行改进，其详细的妙处大可参看文献[10,13]，总的来说，WGAN采用了一种奇特的 **Divergence**—— “推土机-Divergence”， $Wasserstein-Divergence$ 将两个分布看作两堆土，*Divergence* 计算的就是为了将两个土堆推成一样的形状所需要泥土搬运总距离。如下图：



使用 $Wasserstein-Divergence$ 训练的GAN相比原版的GAN有更加明显的“演化”过程，换句话说就是，WGAN的训练相比与GAN更加能突显从“不好”到“不错”的循序渐经的过程。从上面的 **2.2**节，我们知道**JS散度**在面对两个分布不相重叠的情况时，将发生“异常”，计算结果均为 $\log 2$ ，GAN的训练过程也是这样，也许在很长一段训练的过程中，**JS散度**的返回值都是 $\log 2$ ，只有到达某个临界点时，才会突然优化为接近最优值的结果，而**Wasserstein散度**的返回值则要平滑很多。

$JS(P_{g_0}, P_{data}) = \log 2$	$JS(P_{g_1}, P_{data}) = \log 2$	...	$JS(P_{g_m}, P_{data}) = \log 2$	.....	$JS(P_{g_n}, P_{data}) = 0$
$W(P_{g_0}, P_{data}) = d_0$	$W(P_{g_1}, P_{data}) = d_1$	...	$W(P_{g_m}, P_{data}) = d_m$	.....	$W(P_{g_n}, P_{data}) = 0$

既然**Wasserstein散度**能够克服**JS散度**的上述弱点，那么使用**Wasserstein GAN**直接吸收生成器**G** softmax层output的Distribution Vector 与真实样本的 One-hot Vector，用判别器**D** 进行鉴定，即便判别器**D**不会傻到真的被“以假乱真”，但生成器output每次更加接近于真实样本的“进步”总算还是能被传回，这样就保证了对于离散数据的对抗训练能够继续下去。 不过**Wasserstein GAN**的原著放眼于对于GAN更加远大的改进意义，并没有着重给出关于文本生成等离散数据处理的实验，反倒是后来的一篇**“Improved Training of Wasserstein GANs”**[11]专门给出了文本生成的实验，从结果上可以看出，WGAN生成的文本虽然远不及当下最牛X的文本生成效果，但好歹能以character为单位生成出一些看上去稍微正常一点的结果了，对比之下，GAN关于文本生成的生成结果显然是崩塌的。

WGAN with gradient penalty (1D CNN)	
Busino game camperate spent odea	Solice Norkedin pring in since
In the bankaway of smarling the	ThiS record (31.) UBS) and Ch
SingersMay , who kill that imvic	It was not the annuas were plogr
Keray Pents of the same Reagun D	This will be us , the ect of DAN
Manging include a tudancs shat "	These leaded as most-worsd p2 a0
His Zuith Dudget , the Denmbern	The time I paidOa South Cubry i
In during the Uitational questio	Dour Fraps higs it was these del
Divos from The ’ noth ronkies of	This year out howneed allowed lo
She like Monday , of macunsuer S	Kaulna Seto consficutes to repor
Standard GAN objective (same architecture)	
ddddddddddddddddddddddddddddddddd	ddddddddddddddddddddddddddddddddd

3.2. Gumbel-softmax，模拟Sampling的softmax

另外一篇来自华威大学+剑桥大学的工作把改进GAN用于离散数据生成的重心放在了修改softmax的output这方面。如**2.2**节所述，Sampling 操作中的 $\arg\max(\cdot)$ 函数将连续的softmax输出抽取成离散的成型输出，从而导致Sampling的最终output是不可微的，形成GAN对于离散数据生成的最大拦路虎，既然不用Sampling的时候，output与真实分布不重叠，导致**JS散度**停留于固定值 $\log 2$ ，如果用了Sampling的话，离散数据的正常输出又造成了梯度 *Back-Propagation* 上天然的隔阂。

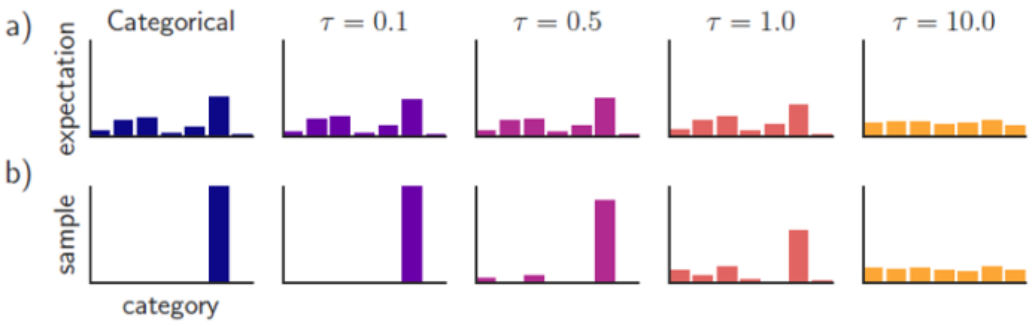
既然如此，论文的作者寻找了一种可以高仿出Sampling效果的特殊softmax，使得softmax的直接输出既可以保证与真实分布的重叠，又能避免Sampling操作对于其可微特征的破坏。它就是“耿贝尔-softmax”（Gumbel-Softmax），Gumbel-Softmax早先已经被应用于离散标签的再分布化[15]（Categorical Reparameterization），在原先的Sampling操作中， $\arg\max(\cdot)$ 函数将普通softmax的输出转化成One-hot Vector：

$$\mathbf{y} = one\_hot\left(\arg\max\left(softmax(\mathbf{h})\right)\right)$$

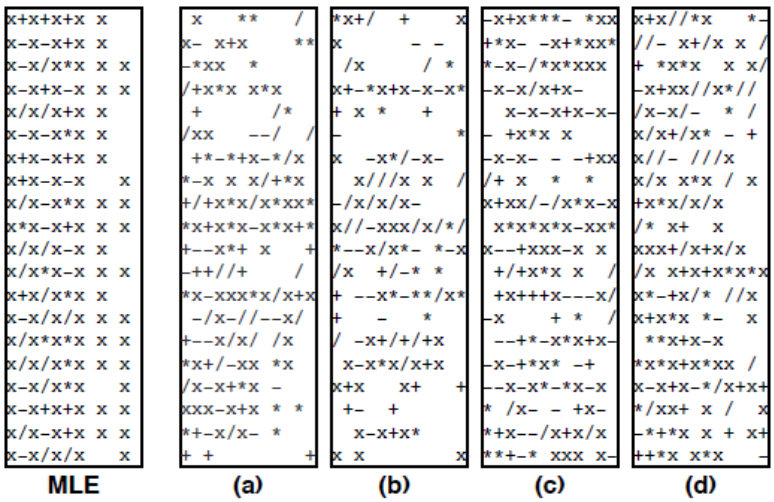
而Gumbel-Softmax略去了 $one\_hot(\cdot) + \arg\max(\cdot)$ 这一步，能够直接给出近似Sampling操作的输出：

$$\mathbf{y} = softmax\left(1/\tau(\mathbf{h}+\mathbf{g})\right)$$

精髓在于这其中的“逆温参数” τ ，当 $\tau \longrightarrow 0$ 时，上式所输出的分布等同于 $one\_hot(\cdot) + \arg\max(\cdot)$ 给出的 *Sampling* 分布，而当 $\tau \longrightarrow \infty$ 时，上式的输出就接近于均匀分布，而 τ 则作为这个特殊softmax中的一个超参数，给予一个较大的初始值，通过训练学习逐渐变小，向 0 逼近，这一部分详细内容可以阅读文献[15]。



论文的实验仅仅尝试使用配合Gumbel-Softmax的GAN进行长度固定为12的 *Context-free grammar* 序列生成，可见GAN的训练并没有崩塌，在少数样例上也得到了还算逼真的效果。



所以，对于GAN进行直接改进用于文本生成，虽说是取得了一定的成效，但距离理想的状态仍然道阻且长，有没有更好的办法呢？当然！

4. RL在GAN文本生成中所扮演的作用

4.1. 关于Reinforcement Learning的闲聊闲扯

强化学习（Reinforcement Learning，RL）由于其前卫的学习方式，本不如监督学习那么方便被全自动化地实现，并且在很多现实应用中学习周期太长，一直没有成为万众瞩目的焦点，直到围棋狗的出现，才吸引了众多人的眼球。

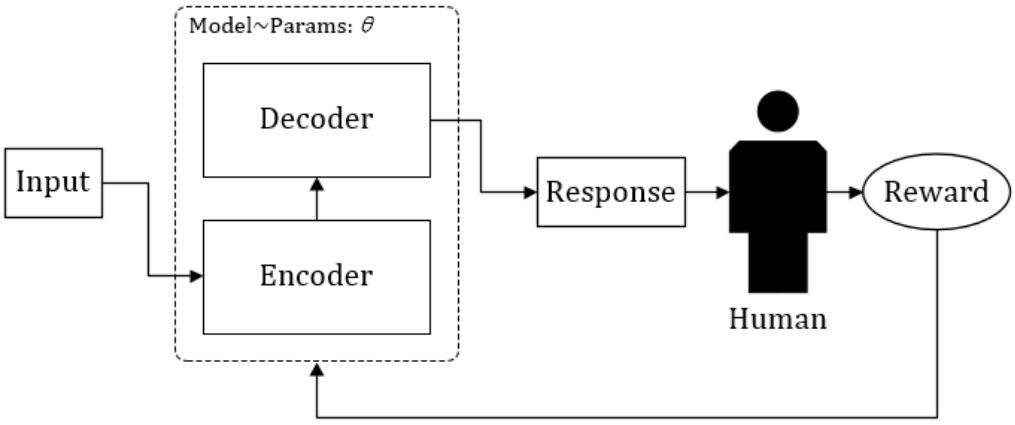
RL通常是一个马尔科夫决策过程，在各个状态 s_i 下执行某个动作 x_i 都将获得奖励（或者是"负奖励"——惩罚） $Reward(s_i, x_i)$ ，而将从头到尾所有的动作连在一起就称为一个**“策略”**或“策略路径” θ^π ，强化学习的目标就是找出能够获得最多奖励的最优**策略**：

$$\theta^\pi_{best} = \arg\max_{\theta^\pi} \sum_{A_i \in \theta^\pi_{best}}^{i} Reward(s_i, x_i)$$

为了达到这个目标，强化学习机可以在各个状态尝试各种可能的动作，并通过环境（大多数是人类）反馈的奖励或者惩罚，评估并找出能够最大化 **期望奖励** $\mathbb{E}\left(\sum_{x_i \in \theta^\pi}^{i} Reward(s_i, x_i), \theta^\pi\right)$ 的策略。

其实也有人将RL应用于对话生成的训练当中[16]，因为对话生成任务本身非常符合强化学习的运行机理（让人类满意，拿奖励）。设，根据输入句子 a ，返回的回答 x 从人类得到的奖励记为 $R(a, x)$ ，而Encoder-Decoder对话模型服从的参数被统一记为 θ ，则基于RL的目标函数说白了就是最大化生成对话的期望奖励，其中 $P_\theta(a, x)$ 表示在参数 θ 下，一组对话 (a, x) 出现的概率。

$$\begin{aligned}\theta_{best} &= \arg\max_\theta \mathbb{E}\left(R(a, x)\right) \\ &= \arg\max_\theta \sum_a \sum_x P_\theta(a, x) R(a, x) \\ &= \arg\max_\theta \sum_a P(a) \sum_x R(a, x) P_\theta(x \mid a)\end{aligned}$$



既然是一个最优化的问题，很直接地便想到使用基于梯度（Gradient）的优化方法解决。当然，在强化学习中，我们要得到的是最优策略 θ_{best} ，此过程便在强化学习领域常听到的 **Policy Gradient**。我们把等式右边 $\arg\max_\theta(\cdot)$ 中的项单独记为 $\tilde{R}_\theta$ ，它表示对话模型找到最优参数时所得到的奖励期望。在实做时，设某句话的应答有N种可能性，则每组对话 (a^i, x^i) 出现的概率可视为服从均匀分布，故还可以进行如下变形：

$$\tilde{R}_\theta = \sum_a P(a) \sum_x R(a,x) P_\theta(x \mid a)$$
$$\approx \frac{1}{N} \sum_{i=1}^{N} R\left(a^i, x^i\right)$$

在优化过程中，对话模型的权重 θ 更新如下， $\nabla \tilde{R}_\theta$ 为所获奖励的变化梯度，

$$\theta^{i+1} \longleftarrow \theta^i + \eta \nabla \tilde{R}_{\theta^i}$$

借助复合函数的求导法则，继续推导奖励的变化梯度，

$$\because \quad \forall F(x), \quad \frac{1}{F(x)} \frac{dF(x)}{dx} = \frac{d \log(F(x))}{dx}$$

$$\begin{aligned} \therefore \quad \nabla \tilde{R}_\theta &= \sum_a P(a) \sum_x R(a,x) \nabla P_\theta\left(x \mid a\right) \\ &= \sum_a P(a) \sum_x R(a,x) P_\theta\left(x \mid a\right) \frac{\nabla P_\theta\left(x \mid a\right)}{P_\theta\left(x \mid a\right)} \\ &= \sum_a P(a) \sum_x R(a,x) P_\theta\left(x \mid a\right) \nabla \log P_\theta\left(x \mid a\right) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} R\left(a^i, x^i\right) \nabla \log P_\theta\left(x^i \mid a^i\right) \end{aligned}$$

这样一来，梯度优化的重心就转化到了生成对话的概率上来，也就是说，通过对参数 θ 进行更新，奖励会使模型趋于将优质对话的出现概率提高，而惩罚则会让模型趋于将劣质对话的出现概率降低。

自AlphaGo使得强化学习猛然进入大众视野以来，大部分对于强化学习的理论研究都将游戏作为主要实验平台，这一点不无道理，强化学习理论上的推导看似逻辑通顺，但其最大的弱点在于，基于人工评判的奖励 $Reward$ 的获得，让实验人员守在电脑前对模型吐出来的结果不停地打分看来是不现实的，游戏系统恰恰能会给出正确客观的打分（输/赢 或 游戏Score）。基于**RL**的对话生成同样会面对这个问题，研究人员采用了类似AlphaGo的实现方式（AI棋手对弈）——同时运行两个机器人，让它们自己互相对话，同时，使用预训练（pre-trained）好的“打分器”给出每组对话的奖励得分 $R(a^i, x^i)$ ，关于这个预训练的“打分器” R ，可以根据实际的应用和需求自己DIY。

$$R(a^i, x^i) = \lambda_1 R(\text{是否通顺}) + \lambda_2 R(\text{是否辞不达意}) + \cdots + \lambda_n R(\text{是否总说重复的废话})$$

稍微感受一下**RL** ChatBot的效果：

Input Message	Mutual Information Model	RL
How old are you?	I'm 16.	I'm 16, why are you asking?
What's your full name?	I have no idea.	What's yours?
I don't want to go home tonight.	Really?	Why?
Do you have any feelings for me?	I don't know what you are talking about.	I don't want to hurt your feelings.
How much time do you have here?	Not long enough. Sorry, sir.	Ten seconds.
Shall we get started?	Of course!	Yes. We've got a lot of work to do here.
Do you play football?	No, i don't	Yes. I love football!
We'd have to talk to him.	I mean, he's a good guy	About what ?
How come you never say it?	Because I don't want to hurt you.	I don't think it's a good idea to say it.

4.2. SeqGAN 和 Conditional SeqGAN

前面说了这么多，终于迎来到了高潮部分：RL + GAN for Text Generation，SeqGAN[17]站在前人**RL** Text Generation的肩膀上，可以说是GAN for Text Generation中的代表作。上面虽然花了大量篇幅讲述**RL** ChatBot的种种机理，其实都是为了它来做铺垫。试想我们使用GAN中的判别器**D**作为强化学习中奖励 $\boldsymbol{Reward}$ 的来源，假设需要生成长度为T的文本序列，则对于生成文本的奖励值 $\tilde{R}_\theta$ 计算可以转化为如下形式：

$$\tilde{R}_\theta\left(s_{1:t-1}\right)=\frac{1}{N}\sum_{i=1}^{N}D\left(s_{1:t-1}+y^i\right),\quad y^i\in MC_\theta\left(s_{1:t-1};N\right)$$

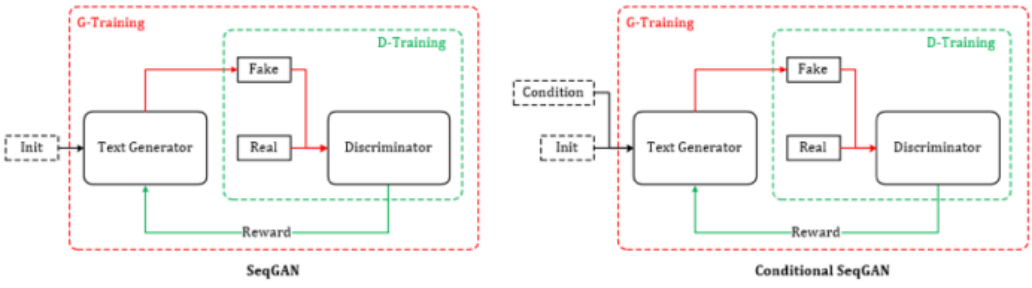
这里要说明几点，假设需要生成的序列总长度为 T ， $s_{1:t-1}$ 是指先前已经生成的部分序列（在**RL**中可视为当前的状态），通过蒙特卡洛搜索得到 N 种后续的序列，尽管文本生成依旧是逐词寻找期望奖励最大的Action（下一个词），判别器**D**还是以整句为单位对生成的序列给出得分 $\boldsymbol{Reward}$ 。

在新一代的判别器 D_{next} 训练之前，生成器 G 根据当前判别器 D 返回的得分不断优化自己：

$$\theta^{next}\longleftarrow\theta^{now}+\eta\nabla\tilde{R_\theta}^{now}$$

$$\begin{aligned}\nabla\tilde{R}_\theta&=\frac{1}{T}\sum_{t=1}^{T}\left(\frac{1}{N}\sum_{i=1}^{N}D\left(s_{1:t-1}+y^i\right)\nabla\log P_\theta\left(y^i\mid s_{1:t-1}\right)\right)\\&=\frac{1}{N}\sum_{i=1}^{N}D\left(x^i\right)\nabla\log P_\theta\left(x^i\right),\qquad\left(y^i\subset x^i\ and\ s_{1:t-1}+y^i=x^i\right)\end{aligned}$$

直到生成器**G**生成的文本足以乱真的时候，就是更新训练新判别器的时候了。一般来说，判别器**D**对生成序列打出的得分既是其判断该序列为真实样本的概率值，按照原版GAN的理论，判别器**D**对于 *real/fake* 样本给出的鉴定结果均为 0.5 时，说明生成器**G**所生成的样本足以乱真，那么倘若在上面的任务中，判别器屡屡对生成样本打出接近甚至高出 0.5 的得分时，即说明判别器**D**需要再训练了。在实做中为了方便，一般等待多轮生成器的训练后，进行一次判别器的训练。

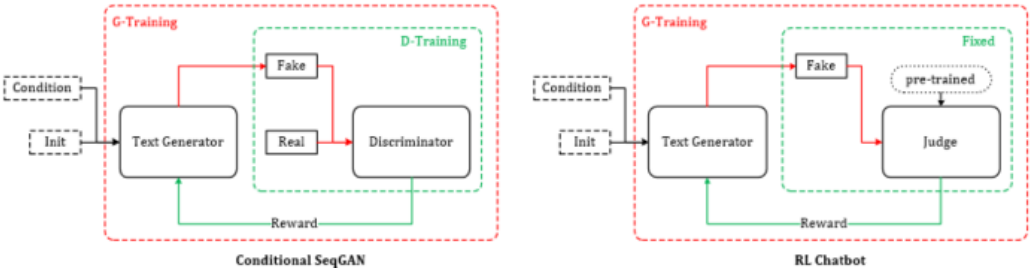


SeqGAN的提出为GAN用于对话生成（Chatbot）完成了重要的铺垫，同样起到铺垫作用的还有另外一个GAN在图像生成领域的神奇应用——Conditional GAN[18~19]，有条件的GAN，顾名思义就是根据一定的条件生成一定的东西，该工作根据输入的文字描述作为条件，生成对应的图像，比如：

对话生成可以理解为同样的模式，上一句对话作为条件，下一句应答则为要生成的数据，唯一的不同是需要生成离散的文本数据，而这个问题，SeqGAN已经帮忙解决了。综上，我自己给它起名：Conditional SeqGAN[20]。根据**4.1**节以及本节的推导，Conditional SeqGAN中的优化梯度可写成：

$$\nabla \tilde{R}_\theta = \frac{1}{N}\sum_{i=1}^{N} D\left(a^i, x^i\right) \nabla \log P_\theta\left(x^i \mid a^i\right)$$

不难看出，此式子与**4.1**节中的变化梯度仅一字之差，只是把“打分器”给出的奖励得分 $R(a^i, x^i)$ 换成了鉴别器认为生成对话来自真人的概率得分 $D\left(a^i, x^i\right)$ 。看似差别很很小，实际上 **RL + GAN** 的文本生成技术与单纯基于**RL**的文本生成技术有着本质的区别：在原本的强化学习对话生成中，虽然采用了AI互相对话，并设定了 *jugle* 进行打分，但这个 *jugle* 是预训练好的，在对话模型的训练过程当中将不再发生变化；**RL + GAN** 的文本生成乃至对话模型则不同，鉴别器**D**与生成器**G**的训练更新将交替进行，此消彼长，故而给出奖励得分的鉴别器**D**在这里是动态的（dynamic）。



RL+ GAN 利用强化学习中的 ***Reward*** 机制以及 **Policy Gradient** 等技术，巧妙地避开了GAN面对离散数据时梯度无法BP的难题，在使用强化学习的方法训练生成器**G**的间隙，又采用对抗学习的原版方法训练判别器**D**。 在Conditional SeqGAN对话模型的一些精选结果中，RL+ GAN 训练得到的生成器时常能返回一些类似真人的逼真回答（我真有那么一丝丝接近“恐怖谷”的感受）。

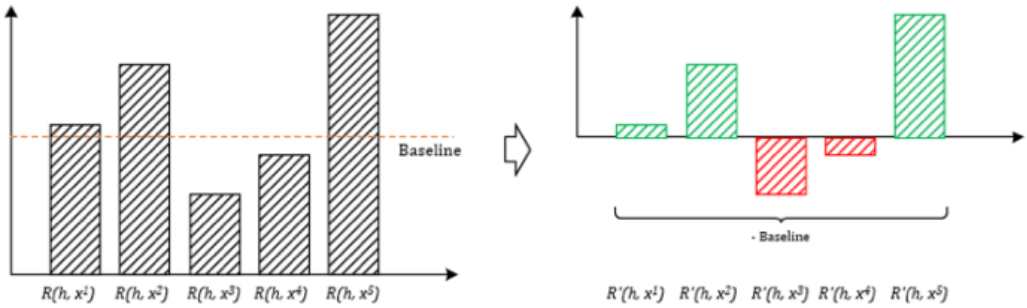
Input	tell me ... how long have you had this falling sickness ?
Vanilla-MLE	i 'm not a doctor .
Vanilla-Sample	well everything you did was totally untrue .
REINFORCE	i don 't know how long it 's been .
REGS Monte Carlo	A few months, I guess .
Input	so i had the doctors test sammy 's response to conditioning .
Vanilla-MLE	sammy wrote the test sammy wrote the test .
Vanilla-Sample	objects pick objects up objects objects objects objects objects
REINFORCE	i 'm not sure that 's relevant .
REGS Monte Carlo	so he took the pills .
Input	what are you going to do ?
Vanilla-MLE	i 'm going to the movies .
Vanilla-Sample	get him outta here first !
REINFORCE	i 'm going to get you .
REGS Monte Carlo	i 'm going to get you a new car .
Input	they fear your power your intellect .
Vanilla-MLE	you 're the only one who knows what 's going on .
Vanilla-Sample	when they are conquered and you surrender they will control all of us .
REINFORCE	i 'm afraid i 'm not ready yet .
REGS Monte Carlo	i 'm not afraid of your power .

5. 一些细节 + 一些延伸

上文所述的，只是 RL + GAN 进行文本生成的基本原理，大家知道，GAN在实际运行过程中任然存在诸多不确定因素，为了尽可能优化 GAN 文本生成的效果，而后发掘更多GAN在NLP领域的潜力，还有一些值得一提的细节。

5.1. Reward Baseline：奖励值上的 Bias

在4.2节中提到，我们采用鉴别器D给予生成样本 x^i 的概率得分（ x^i 属于真实样本的概率）作为奖励 $Reward$ ，既然是概率值，应该意识到这些概率得分都是非负的，如此一来即便生成出再差的结果，鉴别器D也不会给出负 $Reward$ 进行惩罚。从理论上来讲，生成器的训练会趋向于降低较小奖励值样本 x^{low} 出现的概率而提高较大奖励值样本 x^{high} 出现的概率，然而在实做时，由于采样不全等不可控因素的存在，这样不够分明的奖惩区别将有可能使得生成器G的训练变得偏颇。



实际上，在强化学习的对话生成模型当中，就已经出现了此类问题。解决的方法很简单，我们设置一个奖励值 $Reward$ 的基准值Baseline，每次计算奖励值的时候，在后面减去这个基准值作为最终的 **奖励** or **惩罚** 值，使得生成器G的生成结果每次得到的奖惩有正有负，显得更加分明。记奖惩基准值为 b ，则**4.1**节中优化梯度的计算公式修改为：

$$\nabla \tilde{R}_\theta = \frac{1}{N}\sum_{i=1}^{N}\left(R\left(a^i, x^i\right) - b\right)\nabla \log P_\theta\left(x^i \mid a^i\right)$$

对应地，在 RL + GAN 的文本生成任务中，同样在鉴别器**D**对各个生成样本打出的概率得分上减去奖惩基准值 b ，则4.2节中 SeqGAN 与 Conditional SeqGAN 期望奖励值的优化梯度计算公式也分别修改为如下：

$$\nabla \tilde{R}_\theta = \frac{1}{N}\sum_{i=1}^{N}\left(D\left(x^i\right)-b\right)\nabla \log P_\theta\left(x^i\right)$$

$$\nabla \tilde{R}_\theta = \frac{1}{N}\sum_{i=1}^{N}\left(D\left(a^i,x^i\right)-b\right)\nabla \log P_\theta\left(x^i \mid a^i\right)$$

5.2. REGS：一人犯错一人当

细心的读者可以发现，在SeqGAN的奖励优化梯度计算公式的推导中，由鉴别器**D**给予的生成样本奖励得分其实是顺应序列文本的生成过程，逐词产生的，可以看到之前的推导公式中显示了对于Partly文本序列的阶段性奖励值求和再求平均。然而在起初的实验中，根据最终推导的奖励值优化梯度计算公式，鉴别器**D**被训练为用于对整句生成结果进行评估打分，这样的话，鉴别器**D**的打分对于生成序列中的每一个token都是同等的存在，要奖励就一起奖励（奖励值可视为相同），要惩罚就一起惩罚，这种做法会导致一个后果，看下面的例子。

比如有这样一个对话组（包含真实回答和生成回答）：

```
question = ['你', '叫', '什么', '名字', '? ']
real_answer = ['我', '叫', '张三', '。']
fake_answer = ['我', '不', '知道', '。']
```

很显然，鉴别器**D**能够轻易辨识后者回答是假的，必然会给出极低的奖励值得分，但是仔细对比真/假两个回答可以发现，第一个词“*我*”其实和真实样本的第一个词是一样的，而最后一个字符“。”其实也并无大碍，它们其实并没有错，真正错误的是“*不*”*和*“*知道*”这两个词，但很不幸，鉴别器判定 ***fake_answer*** 的整体回答是假的，原本无辜的词项“*我*”和“。”也要跟着一起接受低分判定的惩罚。

让我们回到 GAN + RL 对文本生成模型的优化原理，假设 $P_\theta\left(x^i \mid a^i\right)$ 是面对输入上文 a^i 时生成对话下文 x^i 的概率，我们将它拆分成逐个单词拼接的形式，每一个出现的词汇都将收到之前context的影响。

$$P_\theta\left(x^i \mid a^i\right) = P_\theta\left(x^i[1] \mid a^i\right) + P_\theta\left(x^i[2] \mid a^i, x^i[1]\right) + \cdots + P_\theta\left(x^i[T] \mid a^i, x^i[1:T-1]\right)$$

在**4.1**，**4.2**节中提到，如果生成样本 x^i 被鉴别器**D**打出低分（受到惩罚），生成器**G**将被训练于降低产出此结果的概率。结合上面这条公式，倘若单独将生成序列中的一部分前缀 $x^i[1:t]$ 拿出来与真实样本中完全相同，岂不是也要接受整体低分而带来的惩罚？

解决这一缺陷的直接方法就是把奖惩的判定粒度进一步细化到 word 或 character 级别，在文本逐词生成的过程中对partly的生成结果进行打分。这种处理其实在SeqGAN的论文中[17]就已经实施了，拓展到Conditional SeqGAN中，优化梯度的计算公式应改写为如下：

$$\nabla \tilde{R}_\theta = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(D_e\left(a^i,x^i[1:t]\right)-b\right)\nabla \log P_\theta\left(x^i[t] \mid a^i, x^i[1:t-1]\right)$$

公式中， $D_e\left(a^i,x^i[1:t]\right)$ 是计算的关键，它代表鉴别器**D**在文本逐词生成过程中获得部分文本的情况下对于最终reward的估计，简而言之就是每得到一个新的生成词，就结合此前生成的前序

文本估计最终reward，并作为该生成词单独的reward，SeqGAN的论文中使用蒙特卡洛搜索[21]（Monte Carlo Search，MC search）的方法计算部分生成序列对于整体reward的估计值。而在Conditional SeqGAN的论文中，赋予了这种处理一个名字 —— **Reward for Every Generation Step（REGS）**。

5.3. MC Search & Discriminator for Partially Decoded Sequences：准度与速度的抉择

上一节说到SeqGAN中使用MC search进行部分序列奖励估计值 $D_e\left(a^i, x^i[1:t]\right)$ 的计算，作为**REGS**操作的关键计算，其难处在于，我们并不能预知部分生成序列能给我们带来的最终结果，就好像一场篮球比赛，可能半场结束比分领先，却也不能妄言最终的比赛结果一样。

既然如此，在只得到部分序列的情况下， $D_e\left(a^i, x^i[1:t]\right)$ 只得估计获得，Monte Carlo Search[21]就是其中一种估计方法，Monte Carlo Search的思想极其简单，假设我们已经拥有了部分生成的前缀 $x^i[1:t]$ ，我们使用当前的Generator，强制固定这个前缀，并重复生成出M个完整的序列（有点采样实验的意思），分别交给鉴别器**D**进行打分，这 M 个模拟样本的平均奖励得分即为部分序列 $x^i[1:t]$ 的奖励估计值 $D_e\left(a^i, x^i[1:t]\right)$ 。

$$D_e\left(a^i, x^i[1:t]\right)=\frac{1}{M} \sum_{p=1}^{M}\left(D\left(a^i, x^i[1:t]+x_r^p[t+1:T]\right)\right)$$

当然，使用MC search的缺点也很明显：每生成一个词，就要进行 M 次生成采样，非常耗时；还有一小点，每当我们计算较为后期的一些部分序列奖励估计值的时候，总是会无法避免地再一次计算前面早期生成的项，这样计算出来的 $D_e\left(a^i, x^i[1:t]\right)$ 可能导致对于较前子序列（比如第一个词）的过拟合。

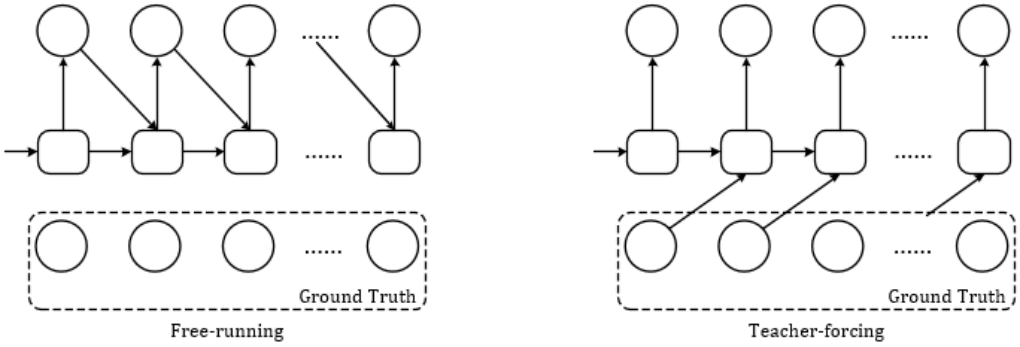
另外一种方法提出于Conditional SeqGAN的论文，干脆训练一个可以对部分已生成前缀进行打分的new鉴别器**D**。将某真实样本的 X^+ 的全部前缀子序列（必须从第一个词开始）集合记作 $\left\{x^+[1:t]\right\}_{t=1}^{T_{X^+}}$ ，同样将某生成样本X^-的全部前缀子序列集合记作 $\left\{x^-[1:t]\right\}_{t=1}^{T_{X^-}}$ ，我们每次从这两者中随机挑选一个或若干个标定为 $+$ 或 $-$ （与原序列相同），与原序列一同加入鉴别器**D**的训练中，这样训练得到的Discriminator便增添了给前缀子序列打分的能力，直接使用这样的Discriminator给前缀子序列打分即可获得 $D_e\left(a^i, x^i[1:t]\right)$ 。这种方法的耗时比起使用MC search要少很多，但得损失一定的准度。

一句话总结两种 $D_e\left(a^i, x^i[1:t]\right)$ 的计算方法：一种是利用部分序列YY出完整序列来给鉴别器打分，而另一种则直接将部分序列加入鉴别器的训练过程，得到可以为部分序列打分的鉴别器，一个较慢，另一个快却损失准度，如何选择就看大家了。

5.4. Teacher Forcing：给Generator一个榜样

在开始讲解SeqGAN中的Teacher Forcing之前，先帮助大家简单了结一下RNN运行的两种mode：(1). Free-running mode；(2). Teacher-Forcing mode[22]。前者就是正常的RNN运行方式：上一个state的输出就做为下一个state的输入，这样做时有风险的，因为在RNN训练的早期，靠前的state中如果出现了极差的结果，那么后面的全部state都会受牵连，以至于最终结果非常不好也很难溯源到发生错误的源头，而后者Teacher-Forcing mode的做法就是，每次不使用

上一个state的输出作为下一个state的输入，而是直接使用ground truth的对应上一项作为下一个state的输入。



就拿Seq2Seq模型来举例，我们假设正输出到第三项，准备生成第四项：

```
input = ['a', 'b', 'c', 'e', 'f', 'g', 'h']
output = ['o', 'p', 's', ...]
label = ['o', 'p', 'q', 'r', 's', 't', 'u']
```

Free-running mode下的decoder会将第三项错误的输出 ***output[2] = 's'***（下标从0开始）作为下一个state的输入，而在Teacher-forcing mode下，decoder则会将正确样本的第三项 ***label[2] = 'q'*** 作为下一个state的输入。 当然这么做也有它的缺点，因为依赖标签数据，在training的时候会有较好的效果，但是在testing的时候就不能得到ground truth的支持了。最好的结果是将Free-running mode的behavior训练得尽可能接近于Teacher-forcing mode，Professor Forcing[23]使用GAN尝试实现了这一目标。

当然，这些都是题外话，我们要回到Teacher-Forcing mode最初的motivation：训练（迭代）早期的RNN非常弱，几乎不能给出好的生成结果（以至于破灌破摔，产生垃圾的output影响后面的state），必须依靠ground truth强行扶着走，才能慢慢进入正轨。

SeqGAN也存在这样的问题，一开始的生成器**G**非常弱，即便是经过一定量的预训练，也几乎生成不出好的Result，然后这些bad result给到鉴别器**D**必然只能返回很低的 $\boldsymbol{Reward}$ （惩罚），生成器**G**的训练只能根据鉴别器的打分来优化而无法得到good example的指导，永远不知道什么是好的结果，结果必然是恶性循环。于是，有必要在SeqGAN训练中给到生成器**G**真实样本的指导，也就是告诉生成器：“什么样的样本才配得到高分 $\boldsymbol{Reward}$?”

4.2节中提到，生成器**G** 和 判别器**D**的训练时交替进行的，由于鉴别器返回的打分是判定输入样本为真的概率，我们可以随机取出一部分真实的样本对话组 $\mathbf{Real} = \left\{ \left(a_r^0, x_r^0 \right), \left(a_r^1, x_r^1 \right), \cdots, \left(a_r^n, x_r^n \right) \right\}$ ，然后直接设置他们的鉴别器奖励值为 $\mathbf{1}$ （或者其他任意定义的最高分），将它们加入生成器**G**的训练过程中，这样生成器就能知道何种样本能得到最高的奖励，从而一定程度上避免了SeqGAN的训练过程由于一方的弱势而发生崩塌。

$$D\left(a_r^i, x_r^i \right) = 1, \qquad \left(\left(a_r^i, x_r^i \right) \in \mathbf{Real} \right)$$

或者也可以这样：用训练好的鉴别器**D**也为随机抽样的真实样本打分，然后加入到生成器**G**的训练过程中，不过，一定要确保鉴别器**D**已经得到充分训练，至少给予任意真实样本 $\left(a_r^i, x_r^i \right)$ 的打分要高于baseline才行（奖励值经过偏置处理后也必须为正）。

$$D\left(a_r^i, x_r^i \right) > b, \qquad \left(\left(a_r^i, x_r^i \right) \in \mathbf{Real} \right)$$

5.5. Actor-Critic：更广义上的GAN?

在DeepMind的一篇半综述式的文章[24]中，谈到了强化学习中的另一个特殊的模型——Actor-Critic，并分析了这个模型与GAN之间的联系。

首先我们回顾一下GAN中鉴别器**D**和生成器**G**优化时的目标函数：

$$D^* = \arg\min_D -\mathbb{E}_{x\sim p_{data}}[\log D(x)] - \mathbb{E}_{z\sim\mathcal{N}(0,I)}[\log(1-D(G(z)))]$$

$$G^* = \arg\min_G \mathbb{E}_{z\sim\mathcal{N}(0,I)}[\log(1-D(G(z)))] = \arg\min_G -\mathbb{E}_{z\sim\mathcal{N}(0,I)}[\log(D(G(z)))]$$

再说说强化学习，在基于策略迭代的强化学习中，通过尝试当前策略的action，从环境获得 $\boldsymbol{Reward}$ ，然后更新策略。这种操作在游戏实验环境中非常有效，因为游戏系统有封闭且清晰的环境，能够稳定地根据各种接收到的action客观地给出对应 $\boldsymbol{Reward}$ ，而在现实生活中，很多时候并没有封闭清晰的环境，给定action应该得到什么样的 $\boldsymbol{Reward}$ 本身也不准确，只能通过设定DIY的打分器来实现，显然这么做很难完美model真实世界千变万化的情况。

那么，能不能先学习出一个能够准确评估出奖励值的值函数 $Q^\pi(s,a)$ ，尽可能地描述环境，对各种action返回较为公正的预期奖励呢？也就是说 $\boldsymbol{Reward}$ 的估计模型本身也是被学习的，这就是Actor-Critic，Actor部分采用传统的Policy Gradient优化策略 π ，Critic部分借助“Q-Learning”学习出最优的action-value值函数，听起来有没有点像GAN的模式？来看看它的目标函数，其中 $\mathcal{D}(\cdot\|\cdot)$ 指任意一中Divergence，值域非负当且仅当两个分布相同时取值为零即可（比如，KL-divergence， JS-divergence 等等）：

$$Q^\pi = \arg\min_Q \mathbb{E}_{s_t,a_t\sim\pi}\left[\mathcal{D}\left(\mathbb{E}_{s_{t+1},r_t,a_{t+1}}\left[r_t+\gamma Q\left(s_{t+1},a_{t+1}\right)\right]\|Q(s_t,a_t)\right)\right]$$

$$\pi^* = \arg\max_\pi \mathbb{E}_{s_0\sim p_0,a_0\sim\pi}\left[Q^\pi\left(s_0,a_0\right)\right] = \arg\min_\pi -\mathbb{E}_{s_0\sim p_0,a_0\sim\pi}\left[Q^\pi\left(s_0,a_0\right)\right]$$

文中将GANs模型比作一种特殊形式的Actor-Critic，并比较了两者各自的特点以及后续的改进技术在两者上的适配情况。试想一下，既然强化学习技术帮助GAN解决了在离散型数据上的梯度传播问题，那么同为强化学习的Actor-Critic也为对抗式文本生成提供了另外一种可能。

Method	GANs	AC
Freezing learning	yes	yes
Label smoothing	yes	no
Historical averaging	yes	no
Minibatch discrimination	yes	no
Batch normalization	yes	yes
Target networks	n/a	yes
Replay buffers	no	yes
Entropy regularization	no	yes
Compatibility	no	yes

5.6. IRGAN：两个检索模型的对抗

IRGAN[25]这篇工作发表于2017年的SIGIR，从作者的阵容来看就注定不是一篇平凡的作品，其中就包含SeqGAN的原班人马，作者将生成对抗网络的思想应用于信息检索领域，却又不拘泥于传统GAN的经典Framework，而是利用了IR领域原本就存在的两种不同路数的model：**生成式IR模型** 和 **判别式IR模型**。

生成式IR模型目标是产生一个query → document的关联度分布，利用这个分布对每个输入的query返回相关的检索结果；而判别式IR模型看上去更像是一个二类分类器，它的目标是尽可能地区分有关联查询对***<query_r, document_r>***和无关联查询对***<query_f, document_f>***，对于给定的查询对***<query, document>***，判别式IR模型给出该查询对中的两项的关联程度。

光从两个模型简单的介绍来看就能丝丝感觉到它们之间特殊的联系，两种风格迥异的IR模型在GAN的思想中“有缘地”走到了对立面，我们将生成式IR模型记作： $p_\theta\left(d \mid q, r\right)$ ，将判别式IR模型记作： $f_\phi\left(q, d\right)$ ，于是整个IRGAN的目标函数为：

$$J^{G^*, D^*}=\min_\theta \max_\phi \sum_{n=1}^N\left(\mathbb{E}_{d \sim p_{real}\left(d \mid q_n, r\right)}\left[\log D\left(d \mid q_n\right)\right]+\mathbb{E}_{d \sim p_\theta\left(d \mid q_n, r\right)}\left[\log \left(1-D\left(d \mid q_n\right)\right)\right]\right)$$

在IRGAN中，鉴别器**D**定义为判别式IR模型的逻辑回归：

$$D\left(d \mid q\right)=\sigma\left(f_\phi\left(d, q\right)\right)=\frac{\exp \left(f_\phi\left(d, q\right)\right)}{1+\exp \left(f_\phi\left(d, q\right)\right)}$$

于是鉴别器**D**的目标函数进一步写为：

$$\phi^*=\arg \max_\phi \sum_{n=1}^N\left(\mathbb{E}_{d \sim p_{real}\left(d \mid q_n, r\right)}\left[\log \left(\sigma\left(f_\phi\left(d, q_n\right)\right)\right)\right]+\mathbb{E}_{d \sim p_\theta\left(d \mid q_n, r\right)}\left[\log \left(1-\sigma\left(f_\phi\left(d, q_n\right)\right)\right)\right]\right)$$

相对地，生成器**G**就直接输出以query为condition答案池中所有document与该query的关联分布，不幸地，我们必须将通过这个关联分布，过滤出当前认为最相关的document答案，才能作为鉴别器**D**的输入来判定此时此刻检索结果的质量，原本连续型的分布经过这一步的折腾又变成离散型的数据了，还好,我们有强化学习，设 $Reward\left(\cdot\right)=\log \left(1+\exp \left(f_\phi\left(\cdot\right)\right)\right)$ ，则生成器**G**的目标函数被写成：

$$\theta^*=\arg \max_\theta \sum_{n=1}^N \mathbb{E}_{d \sim p_\theta\left(d \mid q_n, r\right)}\left[\log \left(1+\exp \left(f_\phi\left(d, q_n\right)\right)\right)\right]$$

也就是最大化鉴别器**D**给出的奖励，而这个奖励值主要来源于检索结果形成的查询对 $\left(d, q_n\right)$ 在判别式IR模型中被认为确实有关联的概率之和。将求和符号内的项记作： $J^G\left(q_n\right)$ ，按照Policy Gradient的方式进行梯度优化，并使用4.1节中的推导方法描述 $J^G\left(q_n\right)$ 的优化梯度，在实做时为了方便，采样 k 个当前生成式IR模型给出的查询结果求近似。

$$\begin{aligned}
\nabla_\theta J^G\left(q_n\right) & =\nabla_\theta \mathbb{E}_{d \sim p_\theta\left(d \mid q_n, r\right)}\left[\log \left(1+\exp \left(f_\phi\left(d, q_n\right)\right)\right)-b\right] \\
& =\sum_{i=1}^M \nabla_\theta p_\theta\left(d_i \mid q_n, r\right)\left[\log \left(1+\exp \left(f_\phi\left(d_i, q_n\right)\right)\right)-b\right] \\
& =\sum_{i=1}^M p_\theta\left(d_i \mid q_n, r\right) \nabla_\theta \log p_\theta\left(d_i \mid q_n, r\right)\left[\log \left(1+\exp \left(f_\phi\left(d_i, q_n\right)\right)\right)-b\right] \\
& \simeq \frac{1}{K} \sum_{k=1}^K \nabla_\theta \log p_\theta\left(d_k \mid q_n, r\right)\left[\log \left(1+\exp \left(f_\phi\left(d_k, q_n\right)\right)\right)-b\right]
\end{aligned}$$

当然，也不能忘了我们的baseline—— b ，文中设置baseline为当前查询结果的平均期望 $Reward$ 。

$$b = \mathbb{E}_{d \sim p_\theta(d|q_n,r)} \left[\log(1 + \exp(f_\phi(d, q_n)))\right]$$

上述是针对Pointwise情形的IR任务，不同于Pointwise情形着重于得到直接的检索结果，Pairwise情形的IR把更多精力放在了ranking上，其返回结果 $R_n = \{\langle d_i, d_j\rangle \mid d_i \succ d_j\}$ 中全是非对称二元对，其中 d_i 比 d_j 与当前的查询项关联性更高。IRGAN也可以扩展到Pairwise的情形，原则是："一切从减"。鉴别器函数将改写为：

$$D(\langle d_i, d_j\rangle \mid q) = \sigma(f_\phi(d_i, q) - f_\phi(d_j, q)) = \frac{\exp(f_\phi(d_i, q) - f_\phi(d_j, q))}{1 + \exp(f_\phi(d_i, q) - f_\phi(d_j, q))}$$

而假设生成器G是一个softmax函数，则Pairwise情形下的变形和简化推导如下：

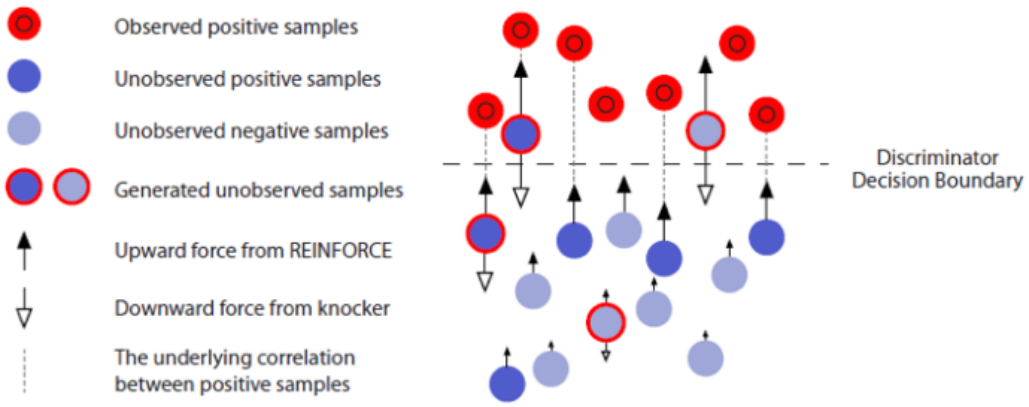
$$p_\theta(d_k \mid q, r) = \frac{\exp(g_\theta(d_k, q))}{\sum_d \exp(g_\theta(d_k, q))}$$

$$G(\langle d_k, d_j\rangle \mid q) = \frac{\exp(g_\theta(d_k, q) - g_\theta(d_j, q))}{\sum_d \exp(g_\theta(d_k, q) - g_\theta(d_j, q))} = \frac{\exp(g_\theta(d_k, q))}{\sum_d \exp(g_\theta(d_k, q))} = p_\theta(d_k \mid q, r)$$

IRGAN在Pairwise情形下的总目标函数如下，其中， o 表示真实的非对称二元组，而 $o\prime$ 则表示生成式IR模型生成的二元组：

$$J^{G^*,D^*} = \min_\theta \max_\phi \sum_{n=1}^{N} \left(\mathbb{E}_{o \sim p_{real}(o|q_n,r)} \left[\log D(o \mid q_n)\right] + \mathbb{E}_{o\prime \sim p_\theta(o\prime|q_n,r)} \left[\log(1 - D(o\prime \mid q_n))\right]\right)$$

IRGAN的一大特点是，对抗model中的两个组件各自都是一种IR模型，所以经过对抗训练之后，不管拿出来哪个，都有希望突破原先的瓶颈。作者还关于IRGAN的训练目标是否符合纳什均衡做了一些讨论，尽管在真实检索的应用中很难获得所谓的真实关联分布，但作者认为不管是观察到的关联样本还是未观察到的关联样本，判别IR模型的输出总是和生成IR模型的对应输出存在着正相关的作用力，于是也孕育而生了文中那个关于浮力和拖拽重物最终达到漂浮平衡状态的略显晦涩的比喻。



结语

这一领域的发展之迅速，也许在我完成这篇Blog的时候，又有一批工作争先恐后的冒出来了，但最终的结局肯定不止于此，我也不怎么擅长结尾，也许要等待GAN来为我，为我们带来一个奇妙的结局。

Acknowledgement

要特别感谢台湾大学李宏毅老师生动的授课[26]，这为我在多个知识点上的理解带来了重要的帮助。

Reference

[1] 何永灿CSDN. 好玩的文本生成[EB/OL]. geek.csdn.net/news/deta....

[2] Ashwin, K, Vijayakumar, Michael, Cogswell, Ramprasath, R, Selvaraju, Qing, Sun, Stefan, Lee, David, Crandall, Dhruv, Batra. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models[J/OL]. arxiv.org/abs/1610.0242....

[3] Minh-Thang, Luong, Hieu, Pham, Christopher, D, Manning. Effective Approaches to Attention-based Neural Machine Translation[J/OL]. arxiv.org/abs/1508.0402....

[4] W. Chan, N. Jaitly, Q. Le and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” ICASSP, 2016, research.google.com/pub....

[5] Jiwei, Li, Minh-Thang, Luong, Dan, Jurafsky. A Hierarchical Neural Autoencoder for Paragraphs and Documents[J/OL]. arxiv.org/abs/1506.0105....

[6] 郑华滨. 从PM到GAN——LSTM之父Schmidhuber横跨22年的怨念[EB/OL]. zhuanlan.zhihu.com/p/27....

[7] Jürgen, Schmidhuber. Learning Factorial Codes by Predictability Minimization[J]. Neural Computation, 1992, 4(6): 863-879, mitpressjournals.org/do....

[8] Ian, J, Goodfellow, Jean, Pouget-Abadie, Mehdi, Mirza, Bing, Xu, David, Warde-Farley, Sherjil, Ozair, Aaron, Courville, Yoshua, Bengio. Generative Adversarial Networks[J/OL]. arxiv.org/abs/1406.2661....

[9] Samuel, R, Bowman, Luke, Vilnis, Oriol, Vinyals, Andrew, M, Dai, Rafal, Jozefowicz, Samy, Bengio. Generating Sentences from a Continuous Space[J/OL]. arxiv.org/abs/1511.0634....

[10] 郑华滨. 令人拍案叫绝的Wasserstein GAN[EB/OL]. zhuanlan.zhihu.com/p/25....

[11] Ishaan, Gulrajani, Faruk, Ahmed, Martin, Arjovsky, Vincent, Dumoulin, Aaron, Courville. Improved Training of Wasserstein GANs[J/OL]. arxiv.org/abs/1704.0002....

[12] Matt, J, Kusner, José, Miguel, Hernández-Lobato. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution[J/OL]. arxiv.org/abs/1611.0405....

[13] Martin, Arjovsky, Soumith, Chintala, Léon, Bottou. Wasserstein GAN[J/OL]. arxiv.org/abs/1701.0787....

[14] Sebastian, Nowozin, Botond, Cseke, Ryota, Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization[J/OL]. arxiv.org/abs/1606.0070....

[15] Eric, Jang, Shixiang, Gu, Ben, Poole. Categorical Reparameterization with Gum-bel-Softmax[J/OL]. arxiv.org/abs/1611.0114....

[16] Jiwei, Li, Will, Monroe, Alan, Ritter, Michel, Galley, Jianfeng, Gao, Dan, Jurafsky. Deep Reinforcement Learning for Dialogue Generation[J/OL]. arxiv.org/abs/1606.0154....

[17] Lantao, Yu, Weinan, Zhang, Jun, Wang, Yong, Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient[J/OL]. arxiv.org/abs/1609.0547....

[18] Mehdi, Mirza, Simon, Osindero. Conditional Generative Adversarial Nets[J/OL]. arxiv.org/abs/1411.1784.

[19] Scott, Reed, Zeynep, Akata, Xinchen, Yan, Lajanugen, Logeswaran, Bernt, Schiele, Honglak, Lee. Generative Adversarial Text to Image Synthesis[J/OL]. arxiv.org/abs/1605.0539....

[20] Jiwei, Li, Will, Monroe, Tianlin, Shi, Sébastien, Jean, Alan, Ritter, Dan, Jurafsky. Adversarial Learning for Neural Dialogue Generation[J/OL]. arxiv.org/abs/1701.0654....

[21] Silver, D.; Huang, A.; Maddison, C. J.;Guez, A.; Sifre, L.; et al. 2016. Mastering the game

of go with deep neural networks and tree search. Nature 529(7587):484–489, nature.com/nature/journ....

[22] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. Neural computation, 1(2), 270–280, mitpressjournals.org/do....

[23] Alex, Lamb, Anirudh, Goyal, Ying, Zhang, Saizheng, Zhang, Aaron, Courville, Yoshua, Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks[J/OL]. arxiv.org/abs/1610.0903....

[24] David, Pfau, Oriol, Vinyals. Connecting Generative Adversarial Networks and Ac-tor-Critic Methods[J/OL]. arxiv.org/abs/1610.0194....

[25] Jun, Wang, Lantao, Yu, Weinan, Zhang, Yu, Gong, Yinghui, Xu, Benyou, Wang, Peng, Zhang, Dell, Zhang. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models[J/OL]. arxiv.org/abs/1705.1051....

[26] Hungyi, Lee. Machine Learning and having it Deep and Structured[EB/OL]. speech.ee.ntu.edu.tw/~t....

编辑于 2017-10-31

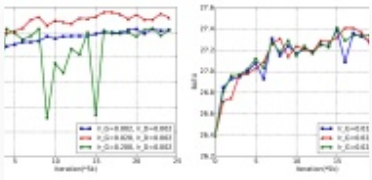
生成对抗网络（GAN） 深度学习（Deep Learning） 强化学习 (Reinforcement Learning)

文章被以下专栏收录

SCUT神经网络-Magellan小分队 进入专栏

数据汪
微信公号【大数据文摘】知乎专栏 进入专栏

推荐阅读



Where GAN/RL meets NMT

Towser

强化学习在视觉上的应用（RL for computer Vision）

iker peng

强化学习 小谈 1. RL是什么鬼

系列前言学习学习两年有余，得到知乎中文社区的帮助，公布自己关于强化学习的一点心得。全部是主观想法，望大家指正。网上强化学习的课程、经典教科书非常多，但往往初学者觉得满满是公式，...

Manne... 发表于强化学习（...

41 条评论

切换为时间排序

写下你的评论...

大黑石头 1 年前
静静的膜拜一下！😊
赞

胡杨 (作者) 回复 **大黑石头** 1 年前
。◕‿◕。
赞

Darren 1 年前
膜拜下同校师兄😄
赞

知乎用户 1 年前
学习了👍
赞

Nango 明楠 1 年前

哇！！强！谢谢作者，学习了！

赞

马佳炯 1 年前

写得真好，谢谢分享~

赞

赵小星 1 年前

非常有用，拜读

赞

罗幕轻寒 1 年前

问下，seqGAN需要生成的序列长度是预先指定好的固定的吗？能到做自己确定要生成的序列长度吗？

1

胡杨 (作者) 回复 **罗幕轻寒** 1 年前

和普通的Sequence Generation（like seq2seq）一样，普通的怎么做，SeqGAN就怎么做，一般是定长不足补0的input并设置结束符，当然最新的tf api也有dynamic rnn的实现

1

罗幕轻寒 回复 **胡杨** (作者) 1 年前

谢谢。

赞

矽陘 1 年前

厉害了。我的胡杨

赞

胡杨 (作者) 回复 **矽陘** 1 年前

你是?

赞

Frank 1 年前

大华工牛博!

赞

张乐 1 年前

作者你好，请问一下 ，你知不知道有没有最新的论文可以识别是人工写的文章还是机器写的文章。

赞

胡杨 (作者) 1 年前

专门针对这个问题的论文没太关注到，但是，SeqGAN中的Discriminator不就是这个功能吗？既然如此，所有GAN NLP的论文都完成了这样的工作，但是本身的重点并不在这上面。
ps：如果看到专门做这个工作的，一定推荐给我一下，谢谢

赞

张乐 回复 **胡杨** (作者) 1 年前

Discriminator 是跟Generator一起训练的吧？只有最后一个Discriminator 才能识别出哪些是机器写的。在不知道Generator的情况下，比如，给你一些训练数据，训练数据里有人写的，机器写的。然后给你一些测试集，直接当做文本二分类来做？还是结合GAN的知识有更好的办法？ps:对GAN还不是很了解，可能有些地方说的不大准确。pps:谢谢答主的细心回复

赞

胡杨 (作者) 回复 **张乐** 1 年前

其实相比生成逼真的数据，鉴别数据的真伪其实是一个相对简单的任务，一般这种任务在过去很多比赛，论文当中屡见不鲜，在聊GAN的时候于是就没有专门当个难题来说。这也是GAN在训练初期，Generator普遍较为弱势的原因（当然训练后期就不一定了），GAN的精妙之处在于增强了Generator的训练难度，因为Discriminator相比赤裸裸的ground-truth的要求更动态延展性更强，Discriminator面对二分类问题时自带的泛化能力某种程度上是对封闭training data的一种提升

赞

Naiyan Wang 1 年前

写的很棒，对于我们不做nlp但是有相关背景的帮助很大

赞

江江酱 1 年前

收货很大！

赞

小虎牙牙 1 年前

irgan 是最大化reward吧 公式咋求的最小呢

赞

胡杨 (作者) 回复 **小虎牙牙** 1 年前

看了一下，手误，谢谢

赞

铭泽 1 年前

请教下，原文中“这样一来，梯度优化的重心就转化到了生成对话的概率上来，也就是说，通过对参数进行更新，奖励会使模型趋于将优质对话的出现概率提高，而惩罚则会让模型趋于将劣质对话的出现概率降低”。这段话是在一些梯度推导后给出的结论，可否具体解释？就是根据那个theta更新的公式，你说的奖励或者惩罚怎么和那个条件概率联系起来的？谢谢。

赞

胡杨 (作者) 回复 **铭泽** 1 年前

简单来说，要宏观地看整个问题，并不是因为生成的结果越来越好了，能够争取到更好的得分，每个生成出来的答案一直会得到一样的得分，我们要优化的是最终得分的期望，期望这个东西就是对于宏观的结果来说的，如果好的结果出现的概率越高，好的结果对应的高分就会越多，坏的结果出现的越少，坏的结果对应的低分就会越少，这样一来，优化过程的实质不就是对于每种生成结果出现概率的优化吗？换句话说，梯度优化微调的不是生成句子的得分，而是生成句子的可能性，至于条件概率，公式表达的意思是在已知前一句对话的条件下，生成下一句回答句的概率

1

铭泽 回复 **胡杨** (作者) 1 年前

十分感谢！我再仔细揣摩下。

赞

陈俊文 1 年前

感谢学长从头到尾的介绍，对新入坑者理清思路很有帮助！

赞

吴海波 1 年前

写的真好，赞赞赞

赞

知乎用户 1 年前

到位!

赞

人工智障 1 年前

请教一下，我对5.3节里讲的MCsearch那里有点困惑。假设词典大小是一万的话，每一步就有一万种策略，做MCsearch的开销不是会很大吗？还是说这里的MCsearch就是指按概率进行sample?

赞

胡杨 (作者) 回复 **人工智障** 10 个月前

Monte Carlo 方法就是指用当前的policy进行 sampling，而policy就是各个opt的概率分布

赞

copytang 1 年前

IRGAN原文有一点不太理解，作者在github给出的训练样例看起来就是doc embedding后的dense向量，D训练输入就是连续的了啊，G生成抽样分布后选出来的样本也是连续的，那不是不存在所谓的样本离散问题了

赞