

**学习用生成性对抗网络修复软件漏洞**



**Jacob A.Harer、Onur Ozdemir、Tomo Lazovich、Christopher P.Reale、，**1,213∗1

**丽贝卡·L·拉塞尔**1，路易·Y·金1，彼得·钦2

1德雷珀，马萨诸塞州剑桥

2波士顿大学计算机科学系，马萨诸塞州波士顿

3Lightmatter，马萨诸塞州波士顿

{jharer、oozdemir、creale、rrussell、lkim}@德雷珀网站, tomo@lightmatter.ai, 邮箱：spchin@cs.bu.edu

# 摘要

基于软件漏洞的自动修复问题，我们提出了一种对抗式学习方法，该方法不需要将成对的标记实例或源域和目标域作为双射，就可以从一个离散的源域映射到另一个目标域。我们证明了所提出的对抗性学习方法是修复软件漏洞的一种有效技术，它的性能接近于需要标记对的seq2seq方法。本文提出的生成性对抗网络方法具有应用无关性，它可以应用于其他类似于代码修复的问题，如语法纠正或情感翻译。

# 1          介绍

软件程序中的安全漏洞给计算机系统带来了严重的风险。恶意用户可以通过其漏洞危害程序，迫使程序以不希望的方式运行（例如崩溃、暴露敏感用户信息等）。每年有数千个这样的漏洞公开报告给公共漏洞和暴露数据库（CVE），还有更多的漏洞是在内部专有代码中发现并修补的[1，2]。这些漏洞通常是程序员所犯错误的结果，并且由于开源软件和代码重用的盛行，这些漏洞可能会迅速传播。

在本文中，我们解决的问题是学习自动修复软件的源代码中包含的安全漏洞。这个问题类似于语法错误纠正，语法错误的句子被翻译成正确的句子。在我们的例子中，错误的源代码（包含漏洞）取代了错误的句子，并被修复为良好的源代码。

神经机器翻译（NMT）系统最近在语言翻译和校正任务方面取得了最先进的性能[ 3, 4, 5，6 ]。这些模型使用编码器-解码器方法将输入序列x=（x0，x1…xT）转换为输出序列y=（y0，y1…yT），例如，将形成英语句子的单词序列翻译成德语句子。到目前为止，训练NMT系统最常用的方法是使用标记的示例对来比较网络输出与期望版本的可能性，这就需要在输入和期望输出数据之间进行一对一的映射。这可能很难获得，因为在大多数情况下，它需要昂贵的手工注释。*0*

在许多序列到序列（seq2seq）应用程序中，获取未配对数据要容易得多，即不需要任何匹配对的源域和目标域的数据，因为这



*∗*

作者隶属于德雷珀时所做的工作。

第32届神经信息处理系统会议（NeurIPS 2018），加拿大蒙特勒尔。

只需要将数据标记为源或目标。例如，在自然语言翻译中，很容易获得不同语言的单语语料库，而且几乎没有成本。对于源代码，存在自动错误检测方法，例如静态分析器或机器学习方法，这些方法可用于标记代码是否存在漏洞，但不提供标记集之间的一对一配对[7，8]。

我们解决这个问题的方法是利用生成性对抗网络（GANs）进行对抗式学习[9]。这种方法可以让我们不用成对的例子来训练。我们采用了传统的NMT模型作为生成器，将典型的负似然损失替换为对抗性鉴别器损失产生的梯度。鉴别器被训练来区分NMT产生的输出和期望输出的真实例子，因此它的损失可以作为产生的分布和真实分布之间差异的代理。这个问题有三个主要困难。首先，为了产生离散输出，从NMT系统的输出采样是不可微的。我们通过使用鉴别器来解决这个问题，该鉴别器在训练期间直接操作NMT系统的预期（软）输出，我们将在第3.2节中详细讨论。第二，对抗性训练不能保证生成的代码与输入的错误代码相对应（即，生成器被训练为匹配分布，而不是样本）。为了强制生成器生成有用的修复（即，生成的代码是输入错误代码的修复版本），我们通过合并两个新的生成器损失函数（如第3.3节所述）来调整输入x上的NMT生成器。第三，我们考虑的域不是双射的，即一个坏代码可以有多个修复，或者一个好代码可以有多个破坏。我们在第3.3节中提出的正则化器在这种情况下仍然有效。我们应该注意到，虽然我们的动机是修复源代码，但本文提出的方法和技术是应用不可知的，因为它们可以应用于其他类似的问题，例如纠正语法错误或在消极情绪和积极情绪之间转换（如在线评论）。此外，虽然软件漏洞修复比检测更困难，但我们提出的修复技术可以利用用于检测的相同数据集，并产生比单独检测更易于解释和有用的工具。

# 2          相关工作

## 2.1         软件修复

关于软件的自动修复已经做了很多研究。在这里，我们描述了以前的数据驱动方法（参见[10]以获得更广泛的主题回顾）。最近两个成功的方法是Le等人[11]和Long and Rinard[12]。Le等人挖掘了跨多个项目的bug修复历史，并尝试在新发现的bug上重用常见的bug修复模式。Long和Rinard学习并使用概率模型对缺陷代码的潜在修复进行排序。这些工作，连同过去在这个领域的大部分工作，需要一组测试用例，用于对生产的维修进行排序和验证。Devlin等人[13]通过使用基于规则的方法生成修复，然后使用神经网络对它们进行排序，从而避免了对测试用例的需要。Gupta等人[14]更进一步，通过训练seq2seq模型来直接生成错误代码的修复。因此，文献[14]中的工作与我们的工作最为相似，但主要缺点是需要成对的训练数据。

## 2.2         甘斯

Goodfellow等人[15]首先引入了GANs来学习自然图像的生成模型。从那时起，许多GANs的变体被创建并应用于图像域[16，17，18，19，20]。由于数据的丰富性和连续性，GANs通常将重点放在图像上。将GANs应用于离散数据（如文本）带来了连续情况下不存在的技术挑战性问题（如通过离散值传播梯度）。一种成功的方法是Yu等人[21]，该方法将鉴别器的输出视为强化学习环境中的奖励。这允许从发生器的输出采样，因为梯度不需要通过鉴别器传回。然而，由于为整个序列提供奖励，因此为生成器计算的梯度不能提供鉴别器认为输出序列的哪些部分不正确的信息，从而导致长的收敛时间。其他几种方法已经成功地将对抗性鉴别器直接应用于具有似然输出的序列发生器的输出。Zhang等人[22]用一个最大平均差（MMD）度量来代替鉴别器中传统的GAN损失，以稳定GAN训练。Press等人[23]和Rajeswar等人[24]都能够使用Wasserstein-GAN[17]生成相当逼真的中等长度句子，这是我们在本文中采用的方法。

研究了如何用输入序列x代替随机变量来调节GAN的发生器。当成对数据可用时，通过向鉴别器提供x和y，可以容易地执行这一点，从而形成Mirza和Osindero的条件方法中的问题[25，26]。然而，这种方法在没有配对的情况下显然更加困难。一种方法是通过使用在相反方向的域之间转换的双生成器对来强制条件化。例如，Gomez等人将循环GAN[27]方法应用于密码破解[28]。他们训练两个生成器，一个获取原始文本并生成加密文本，另一个撤消加密。拥有两个生成器允许Gomez等人使用第一个生成器对原始数据进行加密，然后用另一个生成器对其进行解密，通过添加一个损失函数来确保条件性，该损失函数将双翻译输出与原始原始输入进行比较。Lample等人[29]对NMT采用了类似的方法。它们使用两个编码器/解码器对进行翻译，这两个编码器/解码器对分别从给定语言转换为潜在表示和返回。然后他们使用对抗性损失来确保两种语言之间的潜在表达是相同的，从而允许从一种语言编码然后解码到第二种语言的翻译。对于条件性，他们采用了与戈麦斯等人类似的方法，将一个句子从一种语言完全翻译成另一种语言，再将其翻译回来，然后将原句子与产生的双重翻译进行比较。

Gomez等人和Lample等人的方法都依赖于在两个方向上跨域转换句子的能力。这在许多翻译领域是有意义的，因为有有限的合理的方法可以将一种语言的句子转换成另一种语言的正确句子。这使得网络能够找到从一个域中的每个点到另一个域中的单个点的单个映射，从而仍然覆盖了大多数翻译。不幸的是，在像我们的问题这样的序列校正任务中，一个域包含所有正确的序列，而另一个域包含所有不在正确域中的序列。因此，从正确到错误的映射不是一对一的，而是一对多的。由网络发现的单个映射将无法详细说明所有坏函数的空间，因此只能对它所覆盖的相对较小的坏函数集实施条件限制。因此，我们建议在生成器上使用自正则化项来强制条件化，这在本质上与Shrivastava等人[30]使用的从模拟图像生成逼真图像的方法类似。

# 3          配方

GANs是最初提出的生成模型，用于从随机噪声向量z生成真实图像y[9]。通过将学习问题定义为生成元G（·）和鉴别器D（·）之间的二人极小极大对策，我们找到了一个映射G:z→y，其中，生成器通过最小化鉴别器的性能来学习生成看起来真实的数据样本，鉴别器的目标是最大化其自身在区分生成的样本和真实样本方面的性能。

本文的问题与原始的GaN问题不同，我们的目标是在两个离散值域之间找到映射，即给定的坏码（或源）域X和良好的码（或目标）域Y，通过使用未成对训练样本{Xi} Ni＝1和{Yi}Mi＝1，其中席席X和Y Y Y。

## 3.1         对抗性损失

Goodfello等人[9]的原始GAN损失表示为

*LGAN公司*（D，G）=电子*年∼P*（年）[对数D（y）]+电子*x∼P公司*（十）[对数（1−D（G（x））]（1）

其中，最优生成器为G∗=argminG maxD LGAN（D，G）。众所周知，当生成样本和真实样本的分布不重叠时，这种损失可能是不稳定的[16]。这使得鉴别器提供零梯度。此外，这种标准损失函数可能导致模式崩溃，由此产生的样本来自真实数据分布的单一模式。为了缓解这些问题，Arjovsky等人[17]提出了Wasserstein GAN（WGAN）损耗，该损耗在鉴别器中使用产生的和实际数据样本之间的Wasserstein-1或土方机（EM）距离。EM距离相对容易估计，并导致易于计算的损失函数：

*我WGAN公司*（D，G）=E*年∼P*（年）[年]月*x∼P公司*（十）[D（G（x））]（2）

其中鉴别器函数D被约束为1-Lipschitz。我们在我们的模型中使用WGAN，因为它导致更稳定的训练。

## 3.2         具有离散数据的GANs

离散序列对抗训练的主要挑战之一是，从NMT系统的输出中采样以产生离散输出是不可微的。

训练的目的是从实数序列PY的未知分布中产生样本，样本可以分解为

*T型*

*PY公司*（y） =P（y0）是的*第*（y | yt0，…yt−1）（3）

*t型*=1

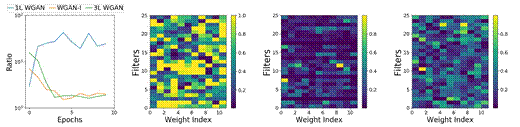
其中，每个条件分布P（yt | y0，…yt−1）用softmax输出进行估计（在本例中使用RNN生成器）

*第*ˆ（y | yt0…yt−1）=sht，软最大值（f（yt−1，t−1））（4）

其中f（·）和ht分别表示发电机网络和RNN在时间t的隐藏状态。理想情况下，我们将从st采样生成一个序列，并将其与实际数据一起提供给鉴别器进行训练，但这个采样过程是不可微的。相反，我们直接为鉴别器提供st。由于每个st依赖于先前生成的输出令牌和RNN状态，我们仍然需要使用argmax从st−1中采样yt−1以生成st。注意，st可以解释为软一热表示，因为它对应于（4）中关于条件分布的一热向量的期望。虽然这种软表示减轻了不可微性的问题，但它可能会引入我们接下来讨论的鉴别器的潜在问题。

注意，由于每个发电机输出st是一个概率向量，它几乎肯定不是一个热向量。换句话说，当每个实标记yt位于标准V−1维单纯形的一个顶点上时，我们生成的输出st位于单纯形的内部。这意味着Pr和Ps具有不相交的支撑，并且在理论上是完全可分离的。因此，存在一个“平凡”鉴别器，该鉴别器独立地查看每个令牌，并基于序列是否由一个热向量组成来判别。这样的鉴别器不会为训练生成器提供有用的信息，因为它不注意令牌之间的顺序依赖关系。然而，我们推测简单的鉴别器架构不存在这个问题，因为这样的“琐碎”鉴别器在实践中可能很难实现。这在我们的实验中得到了验证，我们发现相对较浅的网络，比如只使用一个卷积层的网络，比较深的网络表现更好。

文献[23，24，28]中有相关工作报告了通过使用改进的Wasserstein-GAN（WGAN-GP）损失避免这种“琐碎”鉴别器[31]。然而，在我们的实现中，我们在Wasserstein GAN的原始版本中取得了更大的成功，它在鉴别器中使用了剪裁权重（在两个版本都有足够的超参数调优之后）。我们认为，这是由于原始Wasserstein GAN中的权值剪裁，迫使鉴别器学习更简单的函数，如改进的WGAN论文[31]所示。这些简单的函数不允许鉴别器简单地关注一个热向量，并迫使它关注令牌之间的顺序依赖关系。为了进一步分析这一点，我们在图中提供了一些可视化效果



           （a） 损耗比（b）1层WGAN（c）WGAN-GP（d）3层WGAN

图1：（a）训练过程中正确生成和错误生成对之间的Wasserstein损失率。（b-d）1层1维CNN与WGAN损失、1层1维CNN与WGAN-GP损失和3层1维CNN与WGAN损失的权重。

1，我们使用成对的数据集进行分析。我们随机抽取一组数据对，其中x是y的一个坏版本，并计算两种不同情况下的Wasserstein损失值LWGAN（D，G），如（2）中所定义。对于第一个损耗计算，我们选择发电机G（x）产生正确输出（G（x）=y）的对，对于第二个损耗，产生的输出不正确（G（x）6=y）。然后，我们取这两个损失值的比率，并在图1a中为三种不同的鉴别器设置绘制它们，即i）带WGAN损失的1层CNN；ii）带WGAN-GP损失的1层CNN；以及3）带WGAN损失的3层CNN。仅根据输入是否为一个热向量来区分输入的鉴别器在这两种情况下应具有非常相似的损失值，从而导致损失率为∼1，因为在这两种情况下，生成器都不会产生一个热向量。正如我们在图1a中观察到的，具有原始Wasserstein损耗的更简单的网络结构（本例中为1层CNN）为训练发生器提供了更好的分离，即更好的信号。图1b-1d进一步强调了这一点，其中我们显示了每个网络中第一卷积层上的1-D卷积滤波器（其内核大小为11）的归一化权重。图1b中用于最简单网络的过滤器具有较低的稀疏度，这意味着它们在聚合来自多个令牌的数据时考虑了顺序依赖性，而图1c和1d中的网络具有更高的稀疏度，通常一次只强调单个令牌，我们期望关注单个令牌的鉴别器根据给定的令牌是否是一个热点来判断。这些观察结果意味着一种内在的权衡。过于复杂的鉴别器可以学习基于虚假特征的鉴别器，即，向量是否是一个热点，这可能导致过度拟合。另一方面，一个非常简单的鉴别器将无法精确地对数据进行建模，因此无法向生成器提供任何有用的信息。我们需要像对待超参数一样对待这种权衡，通过逐个应用调整鉴别器模型。

我们还应该提到文献中提出的另外两种方法来克服我们上面讨论的问题。第一种方法是（线性和确定性地）将每个热向量嵌入到低维空间中[23]。这种方法仍然容易受到足够复杂的鉴别器忽略顺序依赖的问题的影响，因为这些嵌入是确定性的。我们在实践中也发现了这种情况；仅在鉴别器中添加一个嵌入项并没有产生明显的改进，仍然需要使用简单的网络。第二种替代方法是使用Gumbel softmax分布通过连续松弛对离散采样过程进行重新参数化[32，33]。由于连续松弛，这种方法仍然通过softmax函数生成（随机）输出，因此类似于我们的soft one热输出。我们对这种方法进行了试验，但没有观察到任何改进。

## 3.3         自正则域映射

在源代码修复或更一般的序列校正的上下文中，我们需要将生成的样本y约束为x的校正版本。因此，我们有以下两个要求：（1）正确的序列在通过生成器时应保持不变；（2）修复后的序列应接近原对应的错误输入序列。

我们探索了两个正则化器来满足这些需求。作为我们的第一个正则化器，除了GAN训练之外，我们还将我们的生成器训练成一个从正确序列中采样的数据的自动编码器。这直接强制执行第（1）项，而间接强制执行第（2）项，因为自动编码器丢失会鼓励正确的子序列保持不变。自动编码器正则化器如下所示

*劳托*（克）=电子*十∼第*（十）[−xlog（G（x））]。(5)

作为第二个正则化器，我们强制生成的输出中每个令牌的频率保持接近输入令牌的频率。这强制执行第（2）项，例外情况是它可能允许改变序列的顺序，例如，任意重新排序不会增加这种损失。然而，GAN丢失减轻了这个问题，因为任意的重新排序产生了与P（y）有很大不同的错误序列。我们的第二个正则化子是

*n*

*LFREQ公司*（克）=电子*x∼P公司*（十）[十*k*频率（x，i）−频率（G（x），i）k22]。(6)

*我*=0

其中n是词汇表的大小，freq（x，i）是x中第i个标记的频率。

# 4          把它们放在一起-提议的GAN框架

我们网络中的发生器由一个标准的NMT系统组成，其注意机制类似于Luong等人[34]。在所有实验中，编码器和解码器都由多层RNN组成，利用长-短期存储器（LSTM）单元[35]。我们根据[34]使用点积注意机制。我们使用基于卷积的鉴别器，因为它们被证明比基于RNN的鉴别器更容易训练，而且通常性能更好[26]。补充材料中提供了其他网络细节。我们有两个不同的正则化损失模型

|  |  |
| --- | --- |
| *我*（D，G）=LWGAN（D，G）+λLAUTO（G） | (7) |
| *我*（D，G）=LWGAN（D，G）+λLFREQ（G） | (8) |

其中，第3.3节定义了LAUTO（G）和LFREQ（G）。我们还用未规范的基损失模型进行了实验，其中我们将λ=0。

## 4.1         自动编码器预培训

我们非常依赖训练前的训练来给我们的GAN一个良好的起点。我们的生成器经过预先训练，成为对所需数据进行去噪的自动编码器[36]。具体来说，我们用损失函数训练发电机：

*我自动预处理*（G） =E*年∼P*（年）[−y对数（G（yˆ））]（9）

式中，yˆ是通过概率为0.2的y中丢弃令牌并随机插入和删除n个令牌而创建的输入的噪声版本，其中n是序列长度的0.03倍。这些数字是基于超参数调整选择的。

## 4.2         课程学习

用于训练seq2seq网络的基于似然的方法通常在训练期间利用教师强制，其中解码器的输入被强制为期望值，而不管在上一时间步生成了什么[37]。这使得即使在训练开始时也可以稳定地训练很长的序列长度。对抗性方法不能使用教师强制，因为所需的序列未知，因此必须始终将st−1样本作为时间t的输入。这可能导致不稳定的训练，因为输出中早期的错误将向前传播，可能在序列的后期造成混乱。为了避免这个问题，我们采用了一种课程学习策略，在整个训练过程中，我们逐渐增加生成序列的长度。我们没有为课程训练选择数据子集，而是将所有序列剪裁为每个课程步骤具有预定义的最大长度。尽管这种方法依赖于鉴别器能够处理不完整的句子，但只要在每次课程更新后对鉴别器进行短暂的再培训，就不会降低性能。

# 5          实验

GAN方法常常因为缺乏简单的评估指标而受到批评。因此，我们将实验的重点放在包含成对例子的数据集上。这使我们能够有意义地评估我们的方法的性能，即使我们的GAN方法不需要成对的训练。这些数据集还允许我们训练seq2seq网络，并将其性能作为基于GAN的方法的上限。我们从探索两个手动数据集开始实验，即排序数字序列和上下文无关语法（CFG），这有助于突出我们提出的GAN方法解决域映射问题的优点。然后，我们研究了在C/C++代码中纠正错误的困难问题。我们所有的结果都在表1中给出。

## 5.1         排序

为了说明实施精确域映射的必要性，我们生成了一个数据集，其中修复任务是将输入按升序排序。我们生成20个随机选择的整数序列（不替换）从0到50之间的升序。然后，我们通过交换相邻的n个选定令牌来注入错误，其中n是平均值为8、标准偏差为4的（四舍五入的）高斯随机变量。然后，任务是在给定错误注入序列的情况下，将序列排序回原来的升序。这种数据生成方案允许我们维护好的（错误注入前）和坏的（错误注入后）数据对，并计算我们的GAN能够从坏序列中恢复好序列的精度。我们将这种精确度称为“序列精确度”（或Seq。附件）。为了评估我们的域映射方法和评估第3.3节中定义的自正则化损失函数的有用性，我们还计算了具有有效顺序但不一定具有有效域映射的序列的百分比，我们称之为“顺序精度”（或顺序精度）。

从表1的结果可以清楚地看出，vanilla（base）GAN很容易学习生成具有有效排序的序列，而不必注意输入序列。这导致高阶精度，但低序列精度。然而，添加自动或频率损失正则化器，如（7）和（8）中所述，显著地改善了Seq。Acc.，这表明这些损失确实有效地加强了源域和目标域之间的正确映射。

## 5.2         简单语法

在我们的第二个实验中，我们从一个简单的上下文无关语法生成数据，类似于Rajeswar等人[24]使用的语法。补充材料中提供了CFG的细节。我们的好数据是从满足语法且长度小于20的所有序列集中随机选择的。然后我们将误差注入到每个序列中，误差的数量被选为高斯随机变量（零阈值和四舍五入），平均值为5，标准偏差为2。然后将每个错误随机选择为删除随机令牌、插入随机令牌或交换两个随机令牌。

网络的任务是从错误注入序列生成原始序列。与上面的排序任务相比，此任务更好地建模真实数据，因为每个生成的标记必须遵循语法，因此以所有以前的标记为条件。表1中的结果表明，我们提出的GAN方法能够在生成符合CFG的正确序列方面实现高CFG精度。除了CFG的准确性之外，我们还根据错误注入前后的配对来计算BLEU分数。我们应该注意到，我们的随机错误注入过程会导致许多坏例子对应于一个特定的好例子或相反的例子，即映射不是双射的。在数据集中有多个坏例子与同一个好例子配对会导致BLEU分数略低，因为网络只能将每个坏输入映射到单个输出。这个问题经常出现在现实世界的修复数据集中，因为代码序列可以被修复或破坏表1：所有实验的结果。Cur是指使用课程学习的实验，Auto和Freq是分别使用LAUTO和LFREQ的实验。Sate4-P和Sate4-U分别表示成对和非成对数据集。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 排序 |  | CFG公司 |  | 卫星4-P | 卫星4-U |
| 模型 | 序号Acc。 | 订单Acc | 布鲁-4 | CFG帐户 | 布鲁-4 | 布鲁-4 |

## 顺序2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 底座 | **99.7** | **99.8** | **91.3** | **99.3** | 96.3 | 不适用 |
| 基本+当前 | **99.7** | **99.8** | 90.2 | 98.9 | **96.4** | 不适用 |
| **拟用GAN** |  |  |  |  |  |  |
| 底座 | 82.8 | 96.9 | 88.5 | 98.0 | 84.2 | 79.3 |
| 基本+自动 | 98.9 | 99.6 | **88.6** | 96.5 | 85.7 | 79.2 |
| 基本+频率 | **99.3** | **99.7** | 88.3 | 97.5 | 86.2 | 79.5 |
| 基本+当前 | 81.5 | 98.0 | 88.4 | **98.9** | 88.3 | 81.1 |
| 基本+当前+自动 | 96.2 | 98.0 | 88.5 | 97.8 | 89.9 | **81.5** |
| 基本+当前+频率 | 98.2 | 99.1 | **88.6** | 96.3 | **90.3** | 81.3 |

多种不同的方式。我们的GAN方法在CFG数据集上表现良好，这表明它可以处理循环方法不适合的问题[28，29，27]。

## 5.3         第四阶段

SATE IV是一个包含116个不同的常见弱点枚举（CWE）类的漏洞的C/C++合成代码示例（函数）的数据集，最初被设计为探索静态和动态分析器的性能[38 ]。每个坏函数都包含一个特定的漏洞，并与几个候选修复配对。总共有117738个函数，其中41171个包含漏洞，76567个不包含漏洞。我们使用自定义lexer对每个函数进行lex。词法分析后，每个函数的长度范围为10到300个标记。

利用这些数据，我们创建了两个数据集来执行两个不同的实验，即配对和非配对数据集。配对数据集允许我们比较GAN方法和seq2seq方法的性能。为了得到一个对GAN和seq2seq训练都公平的数据集，我们通过抽取每个易受攻击代码的示例并随机抽取其中一个修复来创建配对数据。我们将这个过程在数据集中迭代四次，将每个易受攻击的函数与一个抽样修复配对，并将结果集合并到一个大数据集中。我们应该提到的是，尽管成对的数据集包括标记对，但是这些标记并没有用于GAN训练。对于未配对的数据集，我们希望保证给定的源序列在训练数据的任何地方都没有对应的目标序列。为了实现这一点，我们将数据分为两个不相交的集合，将脆弱函数或其候选修复以相等的概率放入训练数据集中。请注意，此操作将训练数据的大小减少了一半。对于测试，我们使用每个易受攻击函数的所有候选修复来计算BLEU分数。我们使用80/10/10%的训练/验证/测试分割。

如表1所示，当我们添加（a）课程培训和（b）LAUTO或LFREQ正则化损失时，我们提出的GAN方法取得了逐渐更好的结果。在不同的GAN模型中，Base+Cur+Freq模型是最好的，并且性能相当接近seq2seq基线，这是性能上限。未配对数据集的结果与配对数据集的结果非常接近，特别是在基本情况下，即使它们是用一半的训练数据获得的。表2中提供了我们的GAN进行正确修复的一些代码示例，补充资料中提供了其他示例。

# 6          结论

我们提出了一种基于GAN的方法来训练离散域映射应用的NMT系统。我们的方法的主要优点是，它可以在没有成对数据的情况下使用，为表2打开了大量以前不可用的数据集：成功修复：（Top）此函数调用sprintf打印出两个字符串，但只提供要打印的第一个字符串。我们的GAN通过提供第二根弦来修复它。（底部）此函数在释放变量后再次使用它。我们的管理员通过移除第一个空闲的文件来修复它。

|  |  |
| --- | --- |
| 脆弱的 | 修复 |
| void CWE685函数调用错误\_  参数数（）{char dst[dst\_SZ]；  sprintf（dst，“%s%s”，SRC\u STR）；printLine（dst）；  } | void CWE685函数调用错误\_  参数数（）{char dst[dst\_SZ]；  sprintf（dst，“%s%s”，SRC\u STR，SRC\u STR）；printLine（dst）；  } |
| void CWE415 \u Double \u Free \uu malloc \u Free \uu struct \u 31（）{  twoints\*数据；数据=空；  数据=（twoints）\*）malloc（100\*sizeof（twoints））；  免费（数据）；  {twoints\*data\u copy=data；twoints\*data=data\u copy；免费（data）；  }  } | void CWE415 \u Double \u Free \uu malloc \u Free \uu struct \u 31（）{  twoints\*数据；数据=空；  数据=（twoints）\*）malloc（100\*sizeof（twoints））；  {twoints\*data\u copy=data；twoints\*data=data\u copy；免费（data）；  }  } |

顺序修正任务。我们的方法的关键是增加了两个新的生成器损失函数，它们可以在不需要多个生成器或域是双射的情况下实现精确的域映射。我们还讨论了如何使用GANs处理离散输出，并取得了一些进展。然而，我们注意到这个问题远未得到解决，很可能会激发更多的研究。尽管我们的方法仅适用于源代码更正问题，但它也适用于其他顺序更正问题，例如语法错误更正或语言情感翻译，例如将负面评论转换为正面评论。

## 致谢

该项目由美国空军研究实验室（AFRL）赞助，是DARPA MUSE项目的一部分。

# 工具书类

[1] 斜接。常见漏洞和暴露。cve.mitre.org.

[2] T.D.LaToza、G.Venolia和R.DeLine。维护心智模型：开发者工作习惯研究。2006年国际软件工程会议。

[3] J.Ji、Q.Wang、K.Toutanova、Y.Gong、S.Truong和J.Gao。一种用于语法错误纠正的嵌套注意-神经混合模型。计算语言学协会（ACL）年会，第753-762页，2017年。

[4] 袁和布里斯科。基于神经机器翻译的语法纠错。计算语言学协会北美分会：人类语言技术（NAACL HLT），2016年。

[5] A.Schmaltz，Y.Kim，A.M.Rush和S.M.Shieber。调整句子纠正的顺序模型。自然语言处理实证方法（EMNLP），2017年。

[6] 谢志忠、阿瓦提、阿里瓦扎根、朱拉夫斯基和吴亚英。基于字符注意的神经语言矫正。附件十四：1603.09727，2016年3月。

[7] J.A.Harer、L.Y.Kim、R.L.Russell、O.Ozdemir、L.R.Kosta、A.Rangamani、L.H。

汉密尔顿、G.I.森特诺、J.R.基、P.M.埃林伍德、M.W.麦康利、J.M.奥珀、P.钦和T.拉佐维奇。基于机器学习的软件漏洞自动检测。附件十四：1803.04497，2018年2月。

[8] 关于hpc中llvm编译器基础设施的第四次研讨会。LLVM-HPC，2017年。

[9] I.J.Goodfello、J.Pouget Abadie、M.Mirza、B.Xu、D.Warde Farley、S.Ozair、，

A.Courville和Y.Bengio。生成性对抗网络。神经信息处理系统（NIPS），2014年6月。

[10] 蒙佩罗斯先生。自动软件修复：参考书目。ACM计算调查（CSUR），51（1）：17:1–17:242018年1月。

[11] X.B.D.Le，D.Lo和C.Le Goues。历史驱动程序修复。软件分析、演化和再工程（SANER），2016年。

[12] F.朗和M.里纳德。通过学习正确的代码自动生成补丁。程序设计语言原理（POPL），2016年。

[13] 德夫林、雅各、乌萨托、约拿单、辛格、利沙布、哥厘、普希米。基于神经符号变换网络的语义代码修复。附件十四：1710.11054，2017年10月。

[14] R.Gupta、S.Pal、A.Kanade和S.Shevade。Deepfix：通过深度学习修复常见的c语言错误。人工智能促进协会（AAAI），第1345-13512017页。

[15] I.J.古德费罗、O.维尼亚和A.M.萨克斯。定性描述神经网络优化问题。国际学习表征会议（ICLR），2015年。

[16] M.Arjovsky和L.Bottou。生成性对抗网络训练的原则性方法。国际学习表征会议（ICLR），2017年。

[17] M.Arjovsky、S.Chintala和L.Bottou。瓦瑟斯坦生成性对抗网络。国际机器学习会议（ICML），2017年。

[18] X.Chen、Y.Duan、R.Houthoft、J.Schulman、I.Sutskever和P.Abbeel。Infogan：信息最大化生成对抗网的可解释表征学习。神经信息处理系统（NIPS），2016年。

[19] A.克雷斯韦尔、T.怀特、V.杜穆林、K.阿鲁库马兰、B.森古普塔和A.A.巴拉斯。生成性对抗网络：综述。IEEE信号处理杂志，35（1）：53–652018。

[20] A.Radford、L.Metz和S.Chintala。基于深度卷积生成对抗网络的无监督表征学习。国际学习表征会议（ICLR），2016年。

[21]余立军，张炜，王杰，余勇。具有策略梯度的序列生成对抗网。人工智能促进协会（AAAI），2017年。

[22]张妍、甘子根、范克凡、陈子珍、何诺、沈德文、卡林。用于文本生成的对抗性特征匹配。国际机器学习会议（ICML），2017年。

[23]O.Press、A.Bar、B.Bogin、J.Berant和L.Wolf。无需预先训练的循环生成对抗网络的语言生成。NLP（SCLeM）子词和字符级模型第一次研讨会，2017年。

[24]S.Rajeswar、S.Subramanian、F.Dutil、C.Pal和A.Courville。自然语言的对抗性生成。第二届NLP代表性学习研讨会（RepL4NLP），2017年。

[25]M.Mirza和S.Osindero。条件生成对抗网。附件十四：1411.1784，2014年11月。

[26]杨梓、陈伟、王芳、徐斌。利用条件序列生成对抗网改进神经机器翻译。计算语言学协会北美分会（NAACL），2018年。

[27]朱俊耀、朴智星、伊索拉和埃弗罗斯。使用循环一致对抗网络的非配对图像到图像的翻译。2017年国际计算机视觉会议（ICCV）。

[28]A.N.戈麦斯、S.黄、I.张、B.M.李、M.奥萨马和L.凯撒。使用离散GANs的无监督密码破解。国际学习表征会议（ICLR），2018年。

[29]G.兰普尔、L.德诺耶和M.兰扎托。仅使用单语语料库的无监督机器翻译。国际学习表征会议（ICLR），2018年。

[30]A.Shrivastava、T.Pfister、O.Tuzel、J.Susskind、W.Wang和R.Webb。通过对抗性训练从模拟和无监督图像中学习。计算机视觉与模式识别（CVPR），2017年。

[31]I.Gulrajani、F.Ahmed、M.Arjovsky、V.Dumoulin和A.C.Courville。改进了瓦塞尔斯坦干草的训练。神经信息处理系统（NIPS），2017年。

[32]E.Jang、S.Gu和B.Poole。使用Gumbel Softmax进行分类重新参数化。国际学习表征会议（ICLR），2017年。

[33]C.J.Maddison、A.Mnih和Y.W.Teh。具体分布-离散随机变量的连续松弛。国际学习表征会议（ICLR），2017年。

[34]M.-T.Luong、H.Pham和C.D.Manning。基于注意力的神经机器翻译的有效途径自然语言处理实证方法（EMNLP），2015年。

[35]S.Hochreiter和J.Schmidhuber。长期短期记忆。神经计算，9（8）：1735–17801997年12月。

[36]P.Vincent、H.Larochelle、Y.Bengio和P.-A.Manzagol。利用去噪自动编码器提取和合成鲁棒特征。2008年国际机器学习会议。

[37]R.J.Williams和D.Zipser。连续运行全递归神经网络的学习算法。神经计算，1989。

[38]V.Okun、A.Delaitre和P.E.Black。静态分析工具展览会（SATE）报告IV.技术报告，2013年。