用于移动测试的自动文本输入生成

Peng Liu*, Xiangyu Zhang , Marco Pistoia , Yunhui Zheng , Manoel Marques and Lingfei Zeng†
⋆ IBM TJ Watson Research Center，Yorktown Heights，New York，USA电子邮件：{Liup，Pistoia，正义，Manoel }@us.ibm.com 普渡大学，西拉斐特，印第安纳州，美国电子邮件：xyzh ang@cs.purdue.edu，zengl@purdue.edu.

已经提出了摘要的设计来改善自动移动测试。尽管有这些改进，但提供适当的文本输入仍然是一个突出的障碍，阻碍了自动化测试方法的大规模采用。关键挑战是如何在用例上下文中自动生成最相关的文本。例如，应在移动浏览器应用程序的地址栏中输入有效的网站地址，以继续测试应用程序;应该在音乐推荐应用程序的搜索栏中输入歌手姓名。如果没有适当的文本输入，测试将被卡住。我们提出了一种新颖的深度学习方法来解决挑战，这减少了最小化问题的问题。另一个挑战是如何使普遍适用于训练有素的应用程序和未经培训的应用程序的方法。我们利用Word2VEC模型来解决挑战。我们已将我们的方法作为工具建立，并用50个IOS移动应用程序进行评估，包括Firefox和Wikipedia。结果表明，我们的方法显着优于现有的自动文本输入生成方法。

I. Introduction

移动设备已成为我们生命中不可或缺的一部分。移动应用程序是提供各种方便和高质量的服务的车辆，如网络浏览，娱乐，交通信息援助，银行和社交网络。为了满足日益增长的需求，移动市场以每天1000多种新应用的速度快速增长[1]
[6]。因此，开发团队不断竞争，大量会议释放截止日期。这遗憾的是，通常会导致移动应用程序中的错误，例如运行时应用程序崩溃，在UI设计和功能上没有完全实现的功能。

移动测试的目标是在释放之前的应用程序中的错误。有两个主流移动测试方法：测试和自动猴子测试。在手动手动测试中，测试人员手动执行操作以尽可能多的用例行使。这种方法的缺点是它需要大量的人类努力，因为测试人员需要在整个测试期间与应用程序密切互动。此外，通常专注于在常见用例中展示功能的人体测试仪可能会想念可以触发异常的角落案例。

提出了提议减少人类努力并最大限度地提高使用案例覆盖率，[19]，[24]，[21]，[21]，[21]，[21]，[21]，[21]，[21]，[21]，[21]，[21]，[21]，[21]，[24]，[24]，[24]，[21]，[24]，[24]，[24]，[27]，[24]，[24]，[27]，[27]，[27]，[27]，以减少人类努力，并最大限度地利用案例覆盖率。“猴子”是一种描述这种类型的测试工作方式的隐喻;像猴子一样，该工具执行随机的动作序列，包括单击UI视图（或UI屏幕）上的按钮和执行随机击键。为了尽可能多的动作序列，研究人员与猴子随机搜索策略相比提出了各种新的搜索算法。尽管有这些改进，但在猴子测试期间提供适当的文本输入仍然是一个突出的障碍，阻碍了猴子测试方法的大规模采用。大多数现有技术都可以在许多用例中几乎没有提供有意义的文本输入。例如，对于电影应用程序，猴子测试几乎不能提供有意义的输入，如星际迷航。相反，它会产生不相关的输入，例如4T6。因此，没有找到结果，从而使猴子无法进入显示适当结果的屏幕。手动规范可能会降低此问题，但会产生大量的人类努力。

让猴子自动产生相关的文本输入。通过这种输入，猴子可以通过长动作序列进行工作流程（短，深UI屏幕）的UI屏幕，而不是在开始时被卡住。一旦到达一个深紫外线屏幕，猴子就可以应用其他面向错误的测试测试技术，例如基于搜索的测试[26]，象征性测试生成[22]，甚至随机测试[34]以识别复杂工作流程中的错误。

A. Generating Relevant Inputs

输入生成的关键要求是生成与上下文相关的输入。

电影→搜索→Star
Trek（1）天气→搜索→纽约（2）

在使用情况1中，单击标记电影的菜单项后，并触发标记为搜索的搜索栏，该应用程序期望来自用户的电影标题。因此，输入之星徒步旅行是相关的;在使用情况2中，单击标记为天气的菜单项后，并触发标记为搜索的搜索栏，该应用程序希望来自用户的城市的名称。因此，纽约的投入是相关的。相反，在一个上下文中相关的输入可能在不同的上下文中不相关。例如，当单击菜单项天气时，输入星际Trek将是不合适的。我们还将读者推荐给VII节中的更多真实世界的例子。挑战。上述要求对自动文本输入生成造成了巨大挑战。首先，与人类只有人类的自然语言语义是特定的相关性

1558-1225/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE.2017.65

测试人员可以理解。因此，传统的自动输入生成方法，例如基于符号执行的测试[22]，不适用。第二，包含包含确定相关文本输入的信息的动作（例如，单击菜单项电影或天气）可能不会在操作序列中的输入操作之前立即。因此，维护和查找文本输入之间的映射和立即先例操作的信息是一种几乎不起作用的方法。我们深入的学习方法。我们提出了一种新颖的基于深入学习的方法来解决上述挑战。在高级别，我们的解决方案由两个阶段组成：在训练阶段，猴子学习测试人员的手动输入并统计地将它们与诸如动作历史记录和文本框标签等上下文相关联;在预测阶段，猴子会根据观察到的上下文自动预测文本输入。我们的方法的核心是一种经常性的神经网络（RNN）模型。这种类型的模型在许多自然语言处理（NLP）应用程序中取得了巨大成功，例如机器转换和输入方法自动完成。拍摄输入方法自动完成，例如，单词中的RNN模型作为非线性功能的语义连接，其中参数培训具有大型文本语料库。给定一个单词，非线性函数计算它旁边的单词的概率分布，然后从分发中采样下一个单词。此外，RNN模型维护了一个存储器状态，以总结来自先前单词的重要信息，并将其作为另一个输入的另一个来源，同时推荐下一个单词。据我们所知，我们是第一个对自动移动测试的文本输入生成问题应用深度学习的。

根据Mikolov等人的实证研究。[32]，RNN模型比传统的统计模型如N-GRAM模型和隐马尔可夫模型更准确。标准情况下，与传统模型相比，RNN导致减少18％的误差减少，假设它们在相同的数据上培训，并且即使传统模型培训比RNN模型更多的数据训练，即使传统模型也培训了12％的错误。

B. App-independent Input Generation

输入生成的另一个重要要求是应用独立性。这意味着RNN模型在一组应用程序上培训也应适用于其他应用。挑战。在不同的应用中使用不同的单词来表示相同的概念对应用程序独立性强加了挑战。

$$Film \rightarrow Search \rightarrow \underline{?} \tag{3}$$

假设在训练阶段没有看到标签电影。人类测试仪可以轻松地在使用情况1中可以轻松地引起标签胶片和电影之间的语义相似性。因此，我们可以预测相关文本输入 – 例如，星际艰难。然而，由于它没有出现在训练阶段，因此RNN模型完全无疏忽。作为一个结果，RNN模型无法预测从标签胶片输入的相关文本。基于Word2Vec的解决方案。我们通过统计学习谷歌新闻语料库的同义词的人类知识来解决挑战。同义词存储在等价类中。每个等价类的代表用于在培训和预测期间替换落入类的单词。

我们的方法是基于Word2Vec模型，是NLP研究人员最近提出的统计模型。该模型将一个单词映射到向量，并且矢量之间的距离测量单词的相似性。Word2Vec模型通过解决优化问题来了解矢量编码。指定，它最大限度地减少了类似词语之间的距离，并最大化了无关的单词之间的距离。Word2VEC是一种无人监督的学习算法，这意味着它不需要手动标记培训数据。

Word2vec以结果的质量优于其他现有模型[33]。这主要是因为Word2VEC可以在2到3个数量级比上训练比现有工作更高的数据，只需在前工作所需时间的一小部分内。

我们已建立一个工具并使用50个iOS应用程序进行评估，包括Firefox和Wikipedia等流行应用程序。评估结果证明了我们方法的有效性。RNN预测平均提高了21％的猴子测试的覆盖范围，而RNN模型和Word2Vec模型的组合将覆盖范围提高28％。此外，如果我们从基准套件中提供的非常少量的UI屏幕排除简单的应用程序，我们就会将RNN预测提高了46％的覆盖率，两种型号的组合导致了60％的改进。同样重要的是，我们的预测非常有效，通常在1毫秒内完成。

总之，我们在这方面发挥以下贡献
paper:

我们提出了一种基于深度学习的新方法，自动生成移动UI测试的相关文本输入。

我们提出了一个小说与学习的Word2Vec NLP模型的组合，以使输入生成应用无关。

我们已将我们的方法作为一个系统，并进行了超过50个IOS应用程序的实验，包括Fikipex和维基百科。结果表明，我们的方法显着增加了猴子测试的覆盖范围。9.

II. Overview of System Design

系统设计如图1所示。猴子测试引擎通过单击屏幕中的按钮来探索移动应用程序。当它遇到文本框时，它请求从文本输入服务器的相关输入，如1 2 ‖ 所示。服务器通过将其进一步调度到人类测试仪或RNN运行实例来解析请求，该请求

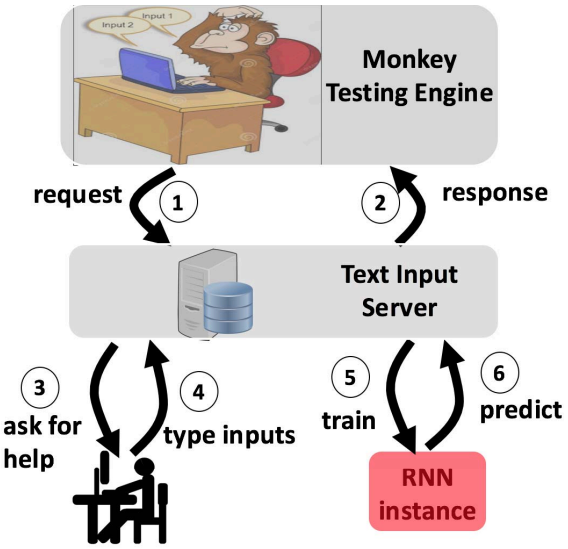

图1.系统的工作流程

对应两种模式：手动模式和AI模式。在手动模型中，在收到帮助3（CID：3）的请求时，人类测试仪可以在上下文4（CID：3）中输入相关的文本输入，或者如果不认为上下文对应，则忽略它在实践中的任何行动序列。然后，测试器输入的输入和相应的上下文将在与服务器关联的数据库中记录。在AI模式下，我们将RNN模型与训练数据集一起录制在数据库5（CID：3）中，之后，RNN实例可以自动预测文本输入6（CID：3）。

在预测阶段6（CID：3）中，MONKEY在给定的上下文中利用RNN模型输入值。基本上，鉴于猴子测试引擎维护的上下文，模型预测了文本输入值。然而，由于我们预测输入的应用程序与我们培训模型的应用程序不同，因此RNN模型可能无法识别在猴子测试期间遇到的一些上下文信息。为了解决这一挑战，我们将RNN模型与Word2Vec模型改进，这有助于识别语义相似的背景，尽管它们不同的语法形式（例如，“电影”和“电影”）。通过新颖的RNN模型和Word2VEC模型的组合，我们有效解决了问题。

在第III节中介绍背景，然后解释我们如何应用深度学习技术以自动预测IV部分中的文本输入。在V节中，我们解释了实施细节。在VI节中，我们讨论了这项工作的假设。第七节介绍了评估结果。

III. Background On Deep Learning

本节介绍了经常性神经网络（RNN）和Word2VEC的背景。rnn和word2vec

$$y = f_a(\sum_i p_i x_i + b)$$

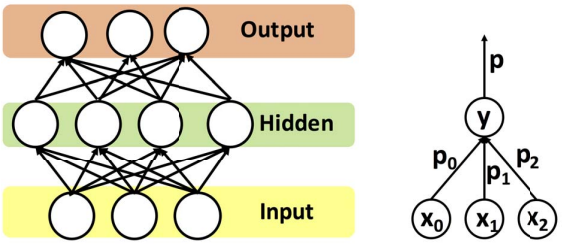

图2.（a）神经网络。（b）神经元

是神经网络的特殊形式。为了讨论这两个，我们必须介绍一个神经网络。神经网络（NN）有两种形式：图形形式和数学形式。在图形形式（图2A）中，NN具有输入层，输出层，一个或多个隐藏层。每层（输入层除外）包括多个神经元，其与前一个层和下层和下层中的神经元连接。如图2b所示，每个神经元是基本计算单元。它是线性地将沿着传入边缘传递的值组合为

i PIXI +
B，其中PI和B是要训练的参数。然后它应用激活函数fa，例如tanh函数[12]或sigmoid函数[11]。激活功能在制作神经元非线性方面是重要的。没有激活函数，神经元仅是线性函数，神经网络也是线性函数，其不能表征许多复杂模型。在数学形式中，NN是神经元代表的功能的组成。让F：RN→RM表示神经网络表示的复合功能。让输入向量 i∈rn表示输入层中的n神经元的输入。让输出矢量 O∈RM表示输出层中的M神经元的输出。神经网络的培训可以被视为优化问题：

$$\underset{p_i,b,\ldots}{\text{minimize}} \sum_{\vec{I},\vec{O}\in train} loss(f(\vec{I}),\vec{O})$$

更新参数以最小化预测输出和训练数据集上的实际输出之间的总损失，其中丢失功能可以被定义为距离或其他形式。无论丢失功能的复杂性如何，通常施加梯度下降算法[16]以最小化其值。

A. Recurrent Neural Network

RNN是一种特殊形式的神经网络，具有反馈回路，如图3所示。为了清楚起见，我们使用箭头表示神经元之间的连接。

循环允许在网络执行步骤中导出的信息传递到下一个，类似于人类的长期内存。图4显示了一个概念上展开的版本，其中神经网络在概念上

复制以通过多个步骤服务一系列输入。注意神经网络副本之间的连接。

应用输入方法自动完成，我们应用RNN生成句子“你是如何做的”。单词逐个送到输入层。每个单词在单热量矢量表示中被编码，以实现学习，例如，单词“如何”被编码为[1,0,0,0]。简单地，向量的每个条目对应于词汇表中的一个单词（[“如何”，“，”您“，”执行“]）和矢量中的值1的位置表示编码单词。从概率的角度来看，向量表示具有100％概率和其他具有0％概率的词。

利用初始参数设置，给定输入单词与从上一步传递的信息组合，神经网络输出概率向量，其中第i个条目估计了词汇表中的第i个词的概率。例如，基于输入“如何”，假设它将下一个单词预测为[0.4,0.2,0.1,0.3]，即最可能的下一个单词是“如何”。但是，我们观察到训练数据集，下一个单词应该是“是”。因此，训练算法需要更新参数，使得“单词”的预测概率变得明显大于其他单词。类似地，鉴于“是”这个词，神经网络需要预测“你”，作为比其他单词更高的概率。

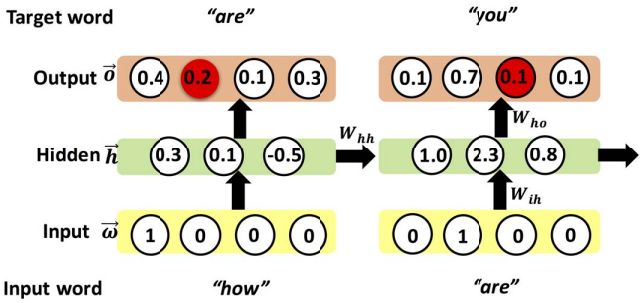

图4.复发性神经网络展开

B. Word2Vec

Word2Vec算法在空间RK中学习每个单词的矢量表示，其中类似的单词可能具有类似的向量。Word2VEC通常应用于非常大的语料库。为了更好的可扩展性，Word2VEC采用传统的神经网络结构。

Word2VEC采用的矢量表示与上述RNN中使用的单热表示有显着不同。Word2VEC将每个单词映射到空间rk中的特定矢量，其中k远小于词汇量S，而单热表示将每个单词映射到空间{0,1}s中的向量。首先，通过在低维空间中编码向量，Word2VEC降低了计算的空间复杂度。更重要的是，Word2VEC的矢量表示在连续空间中，两个矢量之间的距离（即余弦相似度距离）有效地测量单词的相似性。相反，单热表示无法测量相似性。在我们的工作中，我们将Word2vec视为Black Box。

IV. Applying Deep Learning to Input Generation

在高级别，我们的方法统计地学习在训练阶段中文本输入值及其上下文之间的相关性（部分IV-B）。然后在预测阶段（IV-C部分）中，一旦猴子需要在文本框中提供一些值，我们的方法就基于猴子观察到的上下文到目前为止，预测值。IV-D部分进一步解释了我们如何概括输入生成，使其独立于应用。为方便讨论，我们首先推出培训数据集（部分IV-A），用于培训阶段。

A. Training Dataset

不损失普遍性，我们假设简化用户动作模型。在模型中，我们只对两种类型的UI元素感兴趣：按钮和TextBox∈域Tui。各种可点击的UI元素（如菜单和TableView）单元具有与按钮类似的行为，因此我们的模型中具有相同的抽象。类似地，接受用户输入的一系列文本eld，例如安全文本和搜索eld，被抽象为文本框。注意我们的实现完全支持所有这些UI元素。我们只对两种类型的操作，点击和TypeText∈Taction感兴趣，这是UI元素的最代表性行为∈tui。

用户动作α∈A是包含UI元素类型τui∈tui，操作类型τaction∈taction，在UI元素（例如，“电影”中显示图5中的按钮中的标签（CID：5）的标签（CID：5）），并且可选地，涉及动作的值v，例如文本输入值。

训练数据集基本上记录动作序列（CID：6）α0，α1，。。。，αn（CID：7）具有αi在第i步骤中的用户动作。前面αn之前的后续是指用户需要到达发生αn的UI屏幕的动作。有关操作序列的更多细节在VI节中讨论。

我们观察到标签l是用户在使用该应用程序时感知的最突出的信息，思考要作为输入值提供的内容。基于此观察，我们在具有标签（CID：5）α中的动作序列中代表每个动作α，同时抽出所有其他信息。此外，我们也对在动作αn中输入的输入值Vαn感兴趣。因此，上述动作序列简单地表示为（CID：6）（CID：5）α0，（CID：5）α1。。。，（CID：5）αn，vαn（CID：7）。为了便于介绍，我们还参考（CID：5）α0，（CID：5）α1。。。，（CID：5）αn作为输入值Vαn的上下文。

考虑图5中的示例，假设测试器执行操作以触发UI屏幕转换并进入

输入值。在第一个屏幕中，测试仪点击（或点击）用“电影”标记的按钮，导致第二个屏幕，其中搜索栏标有“搜索”。在第三个屏幕中，测试仪在搜索栏中进入输入值“星际Trek”1。

在用户操作模型之后，我们有两个动作：单击带有标签电影的按钮并在搜索栏中输入“Star
Trek”，使用标签搜索。因此，记录的序列是（CID：6）M OVIE，搜索，STAR T REK（CID：7）。

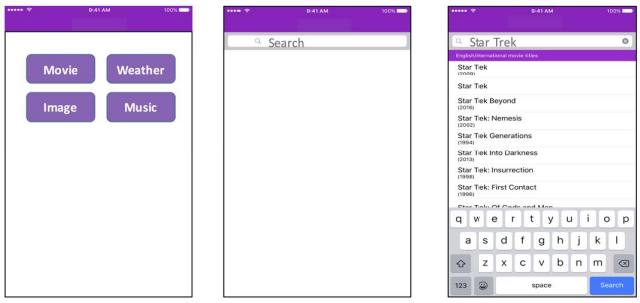

图5.用户操作触发的UI屏幕转换

B. Training Phase

如果没有普遍性，我们假设每个标签（CID：5）或输入值v是单个单词，而不是短语。第VI节介绍了一个简单的预处理，确保假设。让词汇v
=l∪val表示所有标签的列表和出现在训练数据集中的输入值。在下文中，除非另有规定，否则我们不会区分标签和输入值。相反，我们用一般术语单词引用它们。

为了启用学习，我们提出了每个单词的以下向量表示。鉴于单词Ω，其向量表单（CID：2）Ω具有与词汇相同的大小。载体的每一个条目都被定义为，

$$\vec{\omega}[i] = \begin{cases} 1 & \vec{\mathcal{V}}[i] = \omega \\ 0 & otherwise \end{cases}$$

由于只有一个条目可以取值1，因此矢量表示也称为单热表示。考虑图4.在步骤2，输入向量是[0,10,0]，如输入层所示。

在每个步骤中，RNN模型接受单词作为输入的向量表示。输入，以及隐藏层的先前状态（也被编码为向量）来预测输出，该概率向量表征序列中旁边出现的每个单词的概率。培训的目标是更新RNN模型的参数，使得下一个单词的预测概率分布接近于从数据集观察到的真正概率分布。

1，值可以是“星际\ n”，其中尾随符号相当于初始化搜索。

我们将培训正式形式化为优化问题：

$$\underset{x}{\text{argmax}}\ P(v_{\alpha_n} = x | \ell\alpha_0, \ell\alpha_1, \ldots, \ell\alpha_n)$$

鉴于上下文 α0， α1，。。。， αn，我们的输入值vαn=x具有最大条件概率。在训练阶段，我们构建一个RNN模型以预测条件概率。自动计算模型的内部参数，使得预测概率分布近似于从训练数据集观察到的真正概率分布。然后，假设上下文与文本框的输入值之间的概率关联不会导致验证的模型，培训的模型不会被概括地改变。我们解释有关我们网站中的形式化的更多细节[4]。

C. Prediction Phase

培训模型后，它可以用于预测。通常，RNN模型接受一系列单词作为输入并输出概率向量，该概率向量表征词汇中的每个单词在序列之后出现的概率（我们将其称为下一个单词）。概率分布的采样将选择具有分布中描述的概率的单词。

在我们的问题设置中，当猴子遇到文本框时，它将文本框的操作历史记录和标签序列化为序列，然后将序列发送到训练的模型以预测下一个单词的概率分布。最后，它采样概率分布以获得下一个单词的值。

通常，词汇中的任何单词都可以是下一个单词。词汇表中的单词可以对应于动作标签，文本唱片标签或文本输入值，但我们只对预测文本输入值感兴趣。如果采样结果不表示文本输入值（即，它不属于文本输入值的词汇），则我们丢弃它并重新采样分发。

采样概率分布的想法如下。#−o表示RNN模型的输出向量，这使得概率分布表征。我们首先将0到1之间的范围分开到|#−o|间隔。间隔i（0≤i<#−o[j]。统一#−o[j]和tern|#−o|）从随机变量x u（0,1）落入与#的间隔−o[i]，即间隔的长度。换句话说，均匀随机变量x落入间隔I的概率概率与我们选择单词# − v[i]的概率相同。因此，如果随机变量落入时间间隔I，我们将返回单词

考虑以下示例。假设输出概率向量是[0.1,0.7,0.1,0.1]，如图4所示。图6显示了四个间隔。假设统一的随机变量被分配0.5，然后它进入间隔1# − v[1]，（注意有间隔0）。因此，我们返回一词即，“是”这个词。

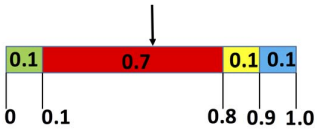

图6.采样概率分布。

D. App-independent Input Generation

我们的最终目标是预测我们没有接受过培训的应用的输入。主要挑战是使输入生成应用无关。

如果我们遇到训练阶段没有看到的单词，我们可能能够将其连接到训练阶段中看到的一些类似的词，利用Word2Vec。我们使用非常大的Google News语料库预先培训的Word2Vec模型。例如，当遇到词汇表中的单词时，例如，在培训期间任何应用程序中不会发生的文本框标签，我们的系统会查询Word2Vec以查找词汇表中最相似的词（更准确地说，的词汇表行动标签和文本框标签）。如果可以从两个单词的矢量表示计算的相似性低于预设阈值0.7，我们认为该词在训练数据集中没有对应物，并且在预测期间简单地忽略它。在最坏的情况下，如果这个词在确定文本输入值时很重要，我们的技术因此降低了传统的猴子测试。这些情况主要发生，因为该应用程序处于一个与我们培训的应用程序类型的新类别非常不同。

V. Implementation

我们在第II节中提出了我们的系统设计的概述。在本节中，我们解释了图1所示的每个组件的实现细节。

a）猴子测试引擎：我们的猴子在探索动作序列中采用深度首次搜索策略，即，它在当前屏幕中所有文本框元素的所有文本键元素的值之后，它单击可用按钮。可以按随机顺序或以某种顺序单击同一屏幕内的按钮（例如，按钮标签的字母顺序）。我们在我们的实验中实施了两种选择并采用随机顺序。此外，为了避免对同一屏幕的重复探索，我们构建了每个屏幕的签名，它由屏幕的标题和屏幕中的按钮的标签组成。具有相同签名的屏幕被视为相同的屏幕，仅探索一次。

注意除了文本输入外，我们的系统可以轻松扩展以预测下一个操作。然而，我们认为这将不合需要地限制猴子达到良好覆盖的能力，因为它将倾向于遵循人体测试人员的动作序列，这是有限的。因此，我们系统的重要设计原则之一是使用对文本输入的动作和RNN预测的随机探索。

我们使用Xcode IDE的测试自动化框架XCTest实现了iOS猴子。该框架提供了一组API [9]，允许我们在屏幕中输入按钮或文本框元素。例如，我们可以使用以下API调用在当前屏幕中轻松实现所有按钮：

```
descendantsMatchingType(.Button)
```

b）文本输入服务器：使用Python Web框架瓶[2]实现的文本输入服务器是一种抽象层，它隐藏了如何生成输入的详细信息。由于它在Python中实现并且猴子测试引擎在SWIFT中实现，因此两个组件之间存在编程语言屏障。要打破障碍，它们通过以JSON格式发送HTTP消息来互相通信。输入服务器通过函数调用与RNN实例通信，因为两者都在Python中实现。

此外，我们为输入服务器维护了一个数据库，以存储人类测试仪在某些情况下输入的输入。我们采用了MongoDB [7]。输入服务器使用Python驱动程序Pymongo与数据库进行交互[10]。

c）RNN实例：我们在Python深度学习框架Tensor FLO OW [13]之上构建了RNN模型，该模型提供了高级API，同时隐藏了许多低级细节。随着张量的抽象，可以使用40行代码构建RNN模型，如我们网站[3]所示。我们通过POWS2VEC预处理数据集进行模型培训了模型，然后将它们送入到RNN模型中。此外，培训的RNN模型可以保存到磁盘上并随时加载以进一步使用。

VI. Discussion

在本节中，我们讨论了我们制作的假设 in this work.

假设我们在训练/预测中做出的是标签或输入值是单个单词，而不是短语或句子。如果标签确实是一个短语，我们将其分解为单词并将每个单词视为单独的标签上下文。通过在这个词级别推理，我们的方法可以处理任何短语，只要它的词汇属于我们的词汇量即可。相比之下，对于短语的文本输入值，我们将短语视为原子单位，以便在整个训练和预测中没有分区。

我们的方法的另一个重要假设是我们培训猴子的应用程序和猴子将测试的应用程序应该分享一些相似性，例如，它们陷入了应用市场上的相同类别。否则，如果我们在娱乐应用程序上培训猴子并将其应用于测试税收，则预测的文本输入将没有意义。为了避免这种情况，我们可以根据我们可以使用尽可能多种不同类型的应用程序并将其作为培训科目。

In our approach, we assume the context comprises only textual labels. In real world apps, developers may use icons 而不是文本标签。我们目前不支持非文本标签。我们注意到我们的方法可以是

扩展以支持非文本标签，例如，利用Image2Text工具[14]。

当我们录制动作序列时，我们维护一个列表并将每个操作附加到列表中。但是，如果我们遇到撤消上一个操作的操作，我们会丢弃动作并从列表中删除上一个操作;如果我们遇到通往应用程序主屏幕的操作，我们只清除列表。

最后，我们的工作补充了最近关于自动动作序列生成的大量进步[28]，[24]，[27]。虽然我们预计伟大的协同作用，我们将把它留给我们未来的工作。

VII. Evaluation

在我们的实验中，我们对以下研究问题感兴趣：

与移动测试的其他自动化输入生成方法相比，我们的自动化方法有多有效？

仅与使用RNN型号相比，Word2Vec是否允许更好的结果？

我们的工具发生的性能超过了什么？要解决第一个问题，我们与自动随机输入生成方法进行比较，该方法随机选择[30]从常用的输入值池中输入值。

我们通过计算屏幕覆盖范围，即，在FIXed时间窗口中探索了许多不同的UI屏幕的效果。通常采用猴子测试来探索尽可能多的用例。但是，独特用例的识别需要域知识。相反，我们使用屏幕覆盖范围来客观地近似使用情况覆盖范围。

注意，屏幕覆盖范围与功能测试中的经典覆盖标准（例如，语句覆盖范围和路径覆盖范围）不同。我们认为这两者都很重要。特别是，我们的方法在这种意义上补充了功能测试，通过提供有意义的输入值，我们的技术允许猴子到达有趣的UI屏幕。从这种UI屏幕开始，可以应用现有的功能测试工作来公开应用崩溃。此外，我们认为即使它们与任何应用程序崩溃不相对应，我们认为一些UI屏幕很有趣。我们将读者推荐给我们的网站[5]了解此类示例。

请注意，我们的方法与自动事件序列生成方法正交。在这项工作中，我们采用现有的深度首页搜索策略。我们的贡献主要介绍基于上下文的相关文本输入，并容忍不同应用中的不同单词的使用。

要解决第二个研究问题，我们将使用启用的RNN和具有RNN和Word2VEC的版本的版本进行比较。第VII-B节通过将三个版本与一起进行比较来解决上述两个问题。

最后，尤其考虑到猴子测试期间预测恰当地发生了预测的性能。理想情况下，该方法应达到有效性低性能开销。在第七节A部分中介绍了性能的测量。

a）实验设置：我们在带有OS X EI Capitan的MacBook Pro上运行实验。iOS应用程序在iPhone 6s Plus（iOS 9.2）的模拟器中运行。我们从Github中收集200个开放的iOS应用程序，主要来自https：//github.com/dkhamsing/open-source-ios.apps。它们落入不同的类别，包括电影，新闻，图像，浏览器，旅行，收音机，日历，天气和任务。它们包括Firefox和Wikipedia等流行应用程序。我们接受了150个应用程序培训并测试了剩余的50个应用程序的预测能力。

我们的培训数据集总共包含14061字。当我们收集训练数据集时，我们尝试尽可能重复使用输入值。例如，当我们需要在不同的应用程序中输入电影名称时，我们牢记相同的电影名称。

A. Performance Overhead

学习/培训过程需要很多迭代。我们用于预测的模型培训6小时，但我们在更短的时间内测量性能。我们测量每1000个迭代的计算时间，并报告迭代的平均时间。我们还使用每1000次迭代后生成的模型来预测使用数据集的随机序列来预测下一个单词。这是对模型的准确性进行采样。我们还衡量了RNN模型为培训和预测所花费的时间。如图7所示，除了初始步骤之外，训练/预测时间保持常量。这是因为每次迭代处理FIXed数据量，并且神经网络结构不会动态地改变。另一个重要的观察是，与测试中的其他动作相比，每个预测平均每次预测需要0.7毫秒（例如，输入文本）。

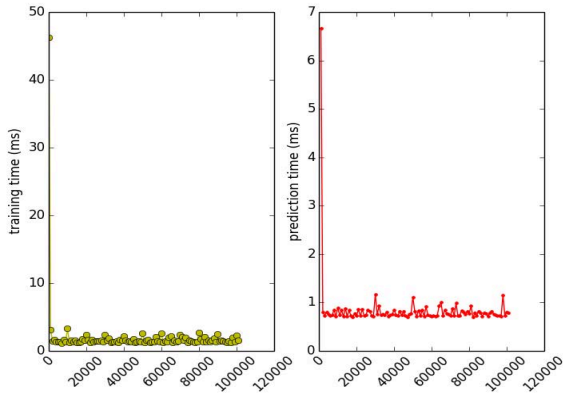

图7. RNN模型的运行时间

我们还测量Word2VEC模型的加载时间以及与类似字的每个查询的时间。加载需要大约54秒，而每个查询需要约0.7

多发性硬化症。由于空间限制，我们没有显示FI。加载时间很长，因为模型很大，这在谷歌新闻语料库上培训，具有30亿次运行的单词。我们的文本输入服务器通过在服务器初始化期间加载模型来解决此问题。一旦加载，该模型可用于所有传入查询。

B.我们方法的有效性

为了衡量效率，我们比较三个版本：（1）随机，随机从常用的输入值池中取值。请注意，随机版本是令人遗论的上下文信息。（2）应用RNN预测模型的RNN，因此知道上下文信息，（3）RNN + WORD2VEC，其能够实现RNN预测和WORD2VEC模型。

在5分钟内由不同版本探索的屏幕数量。对于每个版本，我们重复了实验三次并报告了最大数量。在图8中报告了50个应用程序的结果。RNN版本检测到比随机版本的21.1％更多，而RNN + Word2Vec版本检测到比随机版本更多的屏幕28.6％。由于有许多应用程序具有简单的功能并因此少量屏幕，这三个版本探索了这些应用程序的相同数量的屏幕。通过排除这些案例，我们观察到RNN版本优于随机版本46％，而RNN + Word2VEC版本优于60％。注意RNN版本和RNN + Word2Vec版本之间的差异突出显示使用Word2VEC的有效性。

我们手动检查详细结果并有一些有趣的观察结果。首先，这三个版本为27个应用程序产生相同的结果。原因是这些应用程序不需要任何用户输入或使用任何类型的输入值。例如，在ProftillFinder中，即使猴子进入搜索栏中，App仍然可以检索并显示销售的房地产物业列表的无意义的输入值。在另一个应用程序聊天中，应用程序用于接收/发送消息，无论邮件如何，它都可以工作。

RNN版本可以探索随机版本无法探索的许多UI屏幕。直观地，当文本框需要特殊格式或特殊含义的输入值时，RNN版本知道该上下文并基于从训练数据集中学习的体验生成正确的输入值。相反，随机版本，它令人沮丧的上下文，通常会产生不符合特殊要求的输入值。例如，在SIP−Calculator应用程序中，有一个具有标签“金额”的文本框，这意味着它期望一些数字输入（即，该应用程序将通过输入数执行一些算术计算）。与我们的RNN模型的猴子了解上下文信息，并了解到这些上下文与数字输入值强烈相关。因此，猴子进入数字输入值并进行

到一个新的屏幕。相比之下，随机版本不会产生任何数字输入，因此无法继续。

在极端情况下，AlzPrevent App（即，研究实验室的调查应用程序）要求用户在登记表格中进入注册表之前。AlzPrevent为注册过程有10个屏幕，包括用户名，高度，重量和其他一些信息。随机版本被卡在第一个页面中，而我们的RNN版本（以及RNN + WORD2VEC版本）知道如何在输入值中输入，因为它已经过多个需要注册的应用程序。因此，我们的RNN版本优于01％的随机版本，RNN + Word2Vec版本优于212％。

第三，我们将RNN + Word2vec版本更好地表现优于RNN版本，当上下文与训练应用程序中的上下文略有不同。使用Word2Vec模型，该版本识别来自不同应用程序的上下文之间的语义相似性。基于相似性输入值，该版本根据与来自培训的应用程序的上下文相关联的知识预测相关信息。RNN模型的改进表明了组合RNN模型和Word2VEC模型的重要性和有效性。

我们还展示了第VII−C节的案例研究策略我们技术的力量。

C. Case Studies

a）Firefox：第一个案例来自Fiial Firefox IOS应用程序。这种情况说明了产生相关输入值的重要性，并演示了我们的方法如何产生它们。如图9所示，在屏幕1上，缺少主按钮（默认情况下），因此无法测试与按钮关联的功能。要显示主按钮，必须点击设置按钮以更改设置。但是，有效的网页URL是屏幕上的输入值。否则，主按钮不会显示出来。我们的猴子可以根据其标签“输入网页”和操作历史设置→主页（在屏幕2上）预测文本框的有效URL，因此启用主页按钮，如屏幕3上所示。

随机版本，忽略了标签“在文本框中输入网页”，会产生一个随机输入值“纽约”。Firefox应用程序不接受输入，因此不会启用主页按钮。实际上，Firefox不会在其数据库中记录无效输入。在Monkey导航到另一个屏幕后，然后丢失随机版本输入的输入值丢失。

通过手动搜索训练数据集中的预测网页URL，我们标识了一些有趣的连接。Web爬网程序应用程序使用URL，并与应用程序中的标签“网站”相关联。我们的方法是基于Word2VEC“网站”和“网页”（屏幕2）之间的相似性认识到，然后基于RNN模型来预测最可能的输入值。此示例展示了我们技术的有效性

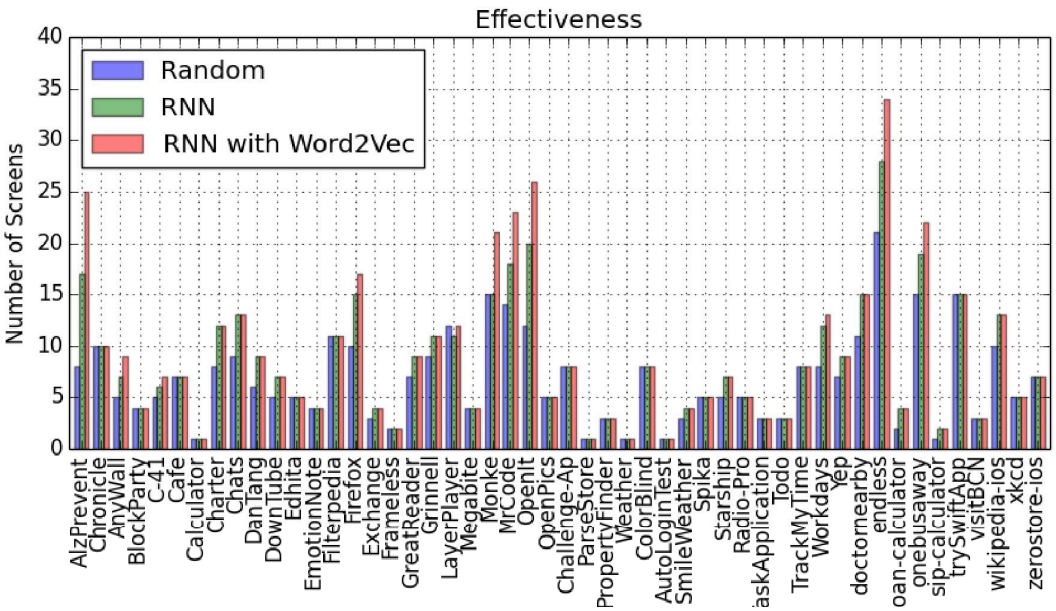

图8.有效性的测量

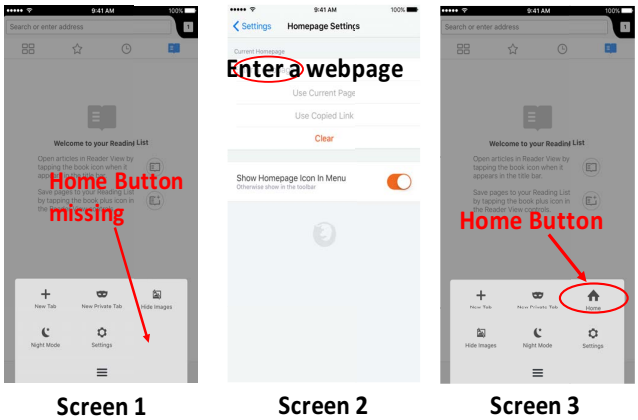

Fig. 9. The Official Firefox App

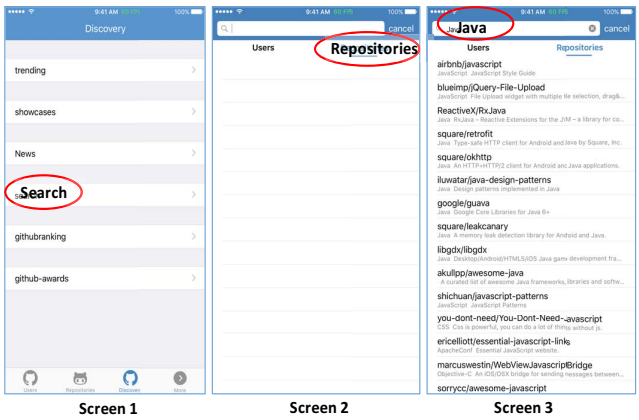

图10.第三方Github App Monkey

探索新的UI屏幕，即使两个应用程序有不同的用例或业务逻辑。

b）第三方GitHub应用程序：图10显示了Github的第三方应用程序的情况。在这种情况下，我们的工具可以点击搜索菜单项（屏幕1），然后选择存储库类别（屏幕2）。它还成功预测搜索栏的输入值Java，这导致返回的一些匹配存储库进一步进一步进展（屏幕3）。我们的工具进一步单击每个存储库，这导致发现异常。

```
108    // Creates a text representation of a GitHub repo
109    var description: String {
110
111        return "[Name: \(self.name!)]" +
112            "\n\t[Stars: \(self.stars!)]" +
113            "\n\t[Forks: \(self.forks!)]" +
114            "\n\t[Owner: \(self.ownerHandle!)]" +
115            "\n\t[Owner: \(self.ownerAvatarURL!)]" +
116            "\n\t[Description: \(self.repoDescription!)]"
```

图11. GitHub应用程序中找到的错误

如图11所示，例外发生在线116.通过使用Swift语言中的感叹号[15]，开发人员假设是存储库的描述的变量重置，不能为nil（即，没有价值）。但是，在实践中，某些存储库没有任何描述，该描述违反了开发人员的假设。因此，当它尝试揭开Nil的可选值时，移动应用程序崩溃。

相比之下，随机版本产生了一个输入值Benjamin Franklin，在当前上下文中是不合适的。因此，此版本无法找到上述错误。这种情况清楚地显示了我们工具的有用性。c）无框：无框是一个全屏Web浏览器，采用最小级别的UI设计。

```
if NSUserDefaults.standardUserDefaults().objectForKey(AppDefaultKeys.FixiOS9.rawValue) as!
    // resize Framer prototypes to fix iOS9 "bug"
    if let absURL = navigationAction.request.URL {
        let isFramerExt = absURL.lastPathComponent!.rangeOfString(".framer")  Thread 1: EXC_BAD
```

图12。无框中发现的错误

我们的工具发现了类似于GitHub应用程序中找到的错误的错误，如图12所示。找到错误需要猴子测试以产生有效输入。考虑图13，在第一次屏幕中，搜索栏显示网站URL或搜索。鉴于这样的上下文，我们的工具基于从训练数据集中学到的统计相关性的统计相关性在第二个屏幕中生成有效网站。移动应用程序然后转移到第三屏幕。通过单击“登录”按钮，我们的工具暴露了图12所示的异常。通过检查代码，我们发现单击按钮的内部执行一些错误xing代码，不幸的是误解了URL的内容。在这种情况下，上下文由四个单词组成，所有这些单词都被发送到RNN模型，以在搜索栏中生成有效输入。相比之下，随机版本没有生成有效的URL，因此无法找到上述错误。

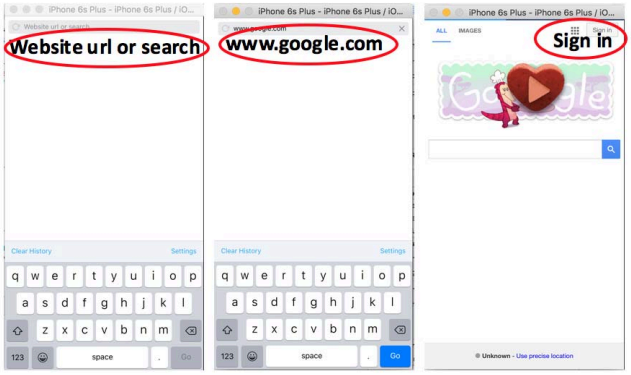

图13.虚拟是无框中的错误

VIII. Related work

我们的工作与Android应用程序的自动测试生成密切相关。猴子[8]和dynodroid
[28]是基于随机探索的UI事件生成工具。Guiripper
[18]（Mobiguitar [19]），轨道[37]，A3E [21]，Swifthand
[21]和Puma
[25]为UI构建并生成事件以系统地探索模型中的状态。竞赛
[20]通过检查事件序列之间的条件来基于Condolic执行方法和Prunes搜索空间来生成事件。ermuth和pradel
[24]介绍了宏观活动，总结了低级UI事件的重复序列一步。通过将宏事件与随机测试组合来，它们利用录制的用户交互序列并自动生成新测试。与我们的方法相比，大多数自动测试生成工作都侧重于生成事件序列（或动作序列）。

除事件和意图外，生成测试输入值也很重要，因为如果输入值的谓词是满意的，则只能暴露某些行为。还应用了符号执行和进化算法的技术。JPF-Android [36]扩展了Java Pathfinder（JPF），是一个模型检查工具，用于探索所有路径并识别运行时失败。Evodroid
[29]基于进化算法框架生成测试（事件和输入）。它使用随机方法生成输入。SAPIENZ
[30]是基于多目标搜索的Android应用程序的测试工具。它结合了随机模糊，系统和基于搜索的探索，利用播种和多级仪器。虽然它可以提供字符串作为测试输入，但是通过反向工程APK从应用中提取这些字符串，并且随机地将其随机接种到文本中。随机选择的输入不太可能在特定上下文中相关。Afshan等人。
[17]基于n-gram语言模型应用引导搜索，以产生可读的字符串输入而不是随机字符序列。但是，该方法不设计用于在用例上下文中生成字符串输入。我们的工作也与基于机器学习的文本建模和生成技术有关。Sutskever等人。
[35]通过将它们应用于预测文本流中的下一个字符的任务来展示大训练的RNN的力量。Melicher等人。
[31]提出了一种基于神经网络的方法来模拟人类选择的密码并测量其对猜测攻击的电阻。

IX. Conclusion

我们开发了一种基于深度学习的方法，可以自动生成移动测试的文本输入。它在上下文中产生最相关的输入值。此外，我们已经利用Word2VEC模型来实现AppIndepence。50多个IOS应用程序的评估证明了我们设计的有效性和效率。

Acknowledgment

我们感谢匿名审稿人的建设性评论。该研究部分由DARPA在合同FA8650-15-C-7562，NSF下，N000968,1320444和1320306，在合同N000141410468下的INR下提供支持，以及在不受限制的礼物下的思科系统。本文中的任何意见和结论都是作者的那些，并不一定是我们赞助者的意见。

References

[1] Apple App Store每天增长超过1000个应用程序。http：//www.ibties。co.uk/apple-app-store-gring-by-over-1000-apps-per-day-10004801。2015年6月6日。[2]瓶子。https://bottlepy.org/[3]用于构建RNN模型的代码片段。https://sites.google.com/site/ monkeyislearning / rnn。

[4]将培训正式化作为优化问题。https://sites.google。com/site/monkeyislearning/home/formal-model.

[5]不是 implemented. https://sites.google.com/site/monkeyislearning/non-crash-cases.

[6]每天将多少新应用程序添加到Google Play中？ https：// www。quora.com/How-many-new-apps-are-added-to-Google-Play-everyday.

[7] MongoDB。https://www.mongodb.com/。[8]猴子UI Android测试工具。http://developer.android.com/tools/help/monkey.html.

[9] Official documentation of xctest apis. https://developer.apple.com/reference/xctest/xcuielementquery.

[10] Pymongo。https://docs.mongodb.com/getting-started/python/client/[11] S形功能。https://en.wikipedia.org/wiki/sigmoid函数。[12] Tanh。https://reference.wolfram.com/language/ref/tanh.html。[13]张量潮。https：//www.tensor fl ow.org。[14]多伦多深度学习演示。http://deeplearning.cs.toronto.edu/。[15]打开可选值。log/why-implicitly-unwrapping-swift-optionals-is-dangerous/.

[16] M. Abadi，P. Barham，J. Chen，Z. Chen，A. Davis，J. Dean，M. Demin，S.Ghemawat，G. Irving，M.Isard，M.Kudlur，J.Levenberg。，R. Monga，S. Moore，DG Murray，B. Steiner，P. Tucker，V.Vasudevan，P. Warden，M.WiCke，Y. Yu和Yu。张量FLO：大规模机器学习系统。在奥斯迪，2016年。

[17] S. Afshan，P. Mcminn和M. Stevenson。使用自然语言模型演变可读字符串测试输入，以减少人类的Oracle成本。2013年IEEE第六届软件测试国际会议，验证和验证，2013年。

[18] D. Amal Fi Tano，A.R.Fasolino，P.Cranontana，S. de Carmine和A. M. Memon。使用GUI剥落进行Android应用程序的自动化测试。在ASE，第258-261页，纽约，纽约，美国，2012年。ACM。

[19] D. Amal Fi Tano，A. R.Fasolino，P.Cranontana，B.D .Ta和A. M. Memon。Mobiguitar：基于自动模型的移动应用程序测试。IEEE软件，32（5）：53-59，2015年9月。

[20] S. Anand，M.Naik，M.J. Harrold和H. Yang。智能手机应用的自动调节测试。在FSE，第59页：1-59：11，纽约，纽约，美国，2012年。

[21] T. Azim和I. neamtiu。Android应用系统测试的有针对性和深度的探索。在Oopsla，第641-660页，纽约，纽约，美国，2013年。

[22] C. CADAR和K. SEN.软件测试的象征性执行：三 几十年后。安排。ACM，56（2）：82-90，2月。2013年。

[23] W.Choi，G. necula和K. Sen.Sen.Sen.Bige GUI测试的Android应用程序具有最小的重启和近似学习。在Oopsla，第623-640页，纽约，纽约，美国，2013年。

[24] M. Ormuth和M. Pradel。猴子看，猴子做：有效地产生带推断宏事件的GUI测试。在ISSTA，第82-93页，纽约，纽约，美国，2016年。

[25] S. Hao，B. Liu，S. Nath，W.G. halfond和R. Govindan。PUMA：用于移动应用的大规模动态分析的可编程UI自动化。在Mobisys，第204-217页，纽约，纽约，美国，2014年。[26] M. Harman和P. Mcminn。搜索基础测试的理论与实证研究：本地，全球和混合搜索。IEEE Trans。软态。ENG。，36（2）：2010年326-247。

[27] C.S.Jensen，M. R. Prasad和A.M ller。使用目标事件序列生成自动测试。在ISSTA，第67-77页，纽约，纽约，美国，2013年。

[28] A.机械，R.Tahiliani和M. Naik。dynodroid：Android应用程序的输入生成系统。在Esec / FSE，第224-234页，纽约，NY，美国，2013年。ACM。

[29] R.Mahmood，N.Mirozaei和S. Malek。Evodroid：Android应用程序分割进化测试。在FSE，第599-609页，纽约，纽约，美国，2014年。

[30] K. Mao，M. Harman和Y.Jia。SAPIENZ：用于Android应用程序的多目标自动化测试。在第25国际软件测试和分析研讨会上，ISSTA 2016年，第94-105页，纽约，纽约，美国，2016年，2016年。

[31] W.Melicher，B. Ur，S. M. Segreti，S. Komanduri，L. Bauer，N. Christin和L. F. Cranor。快速，精益，准确：使用神经网络建模密码猜测。在Usenix Security，第175-191页，Austin，TX，2016年8月。Usenix协会。

[32] T. Mikolov, M. Karafit, L. Burget, J. Cernock, and S. Khudanpur. 经常性神经网络的语言模型。2010年。 In *INTERSPEECH*,

[33] T. Mikolov，I. Sutskever，K. Chen，G. Corrado和J. Dean。单词和短语的分布式表示及其构成性。ecr，2013。

[34] C. Pacheco和M. D.Ernst。Randoop：在Oopsla，第815-816页，纽约，NY，美国，2007. ACM.

[35] I. Sutskever，J.Martens和G. E. Hinton。生成经常性的文本神经网络。在ICML中，第1017-1024页，2011年。

[36] H.Van der Merwe，B.Van der Merwe，以及W. visser。JPF-Android的执行和属性特定规范。sigsoft softw。eng。注意，39（1）：1-5，2月1日2014年。

[37] W.杨，M.R.Prasad和T.Xie。一种自动化GUI模型生成移动应用的灰盒方法。在Fase，2013年。