

谷歌大脑研究员玩转汉字RNN：神经网络生成新华字典

新智元 2018-06-24



新智元报道

来源: blog.otoro.net

报道: 文强

【新智元导读】你永远不知道汉字的潜力。谷歌大脑东京分部的研究员hardmaru，用神经网络根据笔画生成汉字，新造了一系列“假汉字”。你别说，有些看上去还真像那么一回事。



因为我们都是中国人，从小看着汉字、写着汉字长大，所以已经忘记了汉字本身是一件多么困难的事情。

是的，汉字基本的笔画就有点横撇捺等几种，但是，中国文字从甲骨文、金文、篆书、隶书一路走来，不同程度存在难写难认的缺陷。就算只是一个“点”，在不同的字里面，这个点的大小和方向也是各不相同。因此，对于汉字设计师来说，可是要了老命。

调查记者 Nikhil Sonnad 曾经在 QZ 发表过一篇文章，详细讲述了设计一个汉字字体漫长艰苦而又令人沉迷的过程。其中有这样一个例子，展示了言字旁在不同的文字中拥有不同的大小和方向：



言字旁在不同的文字中拥有不同的大小和方向。来源：QZ

这也是为什么相比五花八门的英文和阿拉伯数字字体，汉字的字体那么少的一个原因。

Nikhil Sonnad 在那篇文章中指出，一位经验丰富的设计师可以在6个月的时间里设计一种涵盖几十种西方语言的新字体。但是，对于单个中文字体，至少需要一个好几人的设计师团队两年以上的**时间**。

有没有什么好的方法能够解决这个问题？

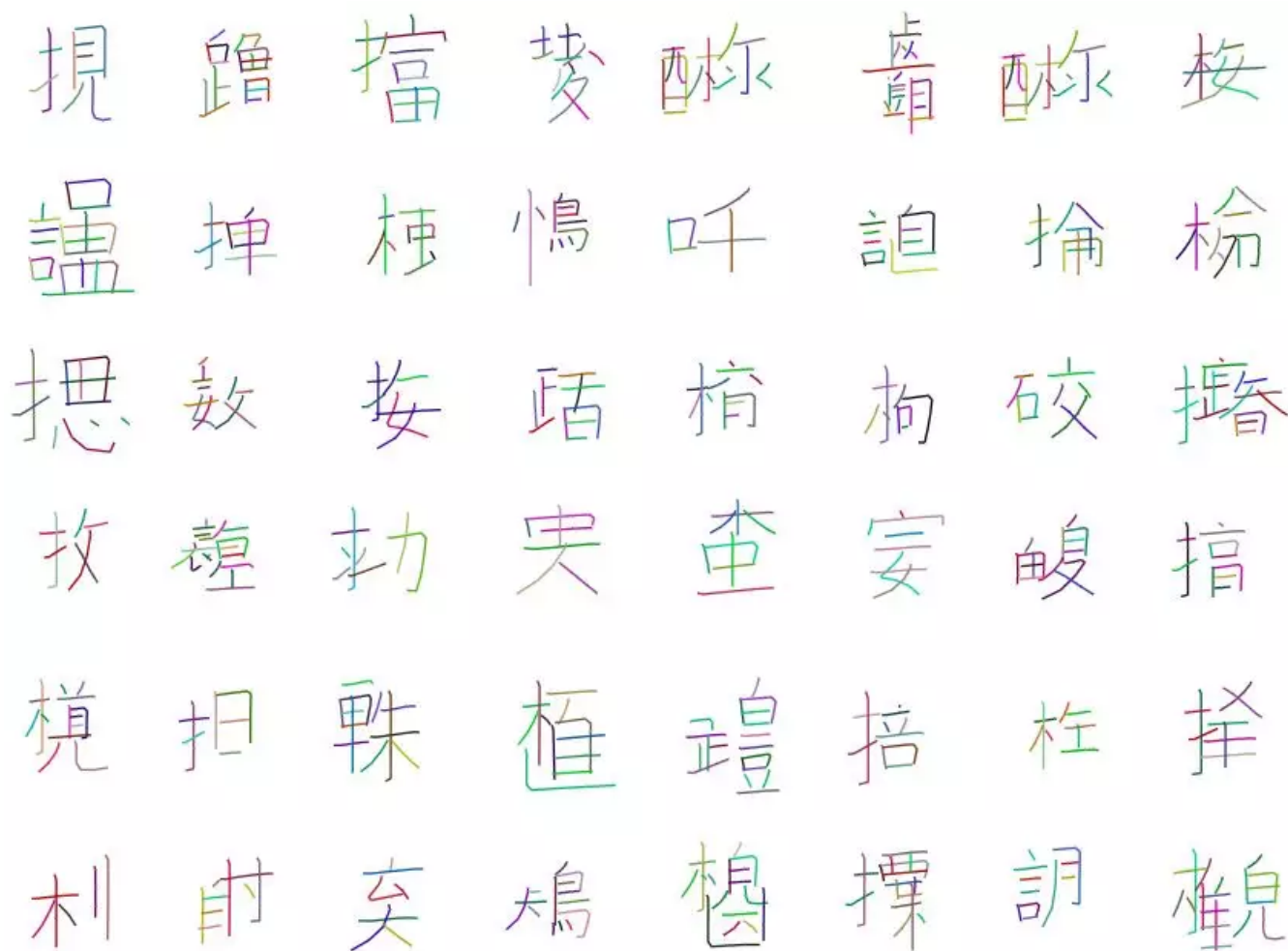
作为新智元（ID：AI_era）的读者，或许有人已经猜到我们会接下来会说什么。是的，还是神经网络。

谷歌大脑东京分部的研究人员hardmaru，使用神经网络生成汉字，但他与众不同的地方在于，由于提供给神经网络的数据是“笔画”，因此**生成的是所有理论上可以存在，但现实中并没有在使用的汉字**。



或许你要说，这样做有什么用，但仔细看就能发现作者这样做在理论和实际上的意义。

汉字这个系统本质上是开放的。使用可用的元素（偏旁部首、笔画等等），可以制作出无数个不同的字符。虽然代码目前还不能很准确地定位笔画的位置，但hardmaru实验中的一些结果，看起来非常像真实存在的汉字。



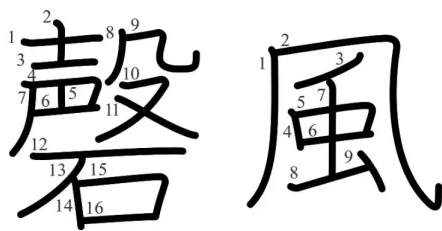
认识一个字不一定写得出，但写得出就一定认识它

在一篇介绍他的这项工作的博客中，hardmaru表示，他从小也被父母硬逼着去学汉字，虽然他周围的人大多说英语。老师教他写汉字的过程就是抄写听写抄写听写的不断循环，就好像LSTM根据训练样本输出序列结果一样。

另一方面，他也注意到，“**写**汉字和“**读**汉字是两个非常不同的过程。你认识一个字（能够阅读或者发出读音），但不一定写得出；但是，如果你能写出一个汉字，你一定知道它的发音。现在，人们越来越多的依赖基于发音的输入法来“写”汉字，当真正提笔写字的时候，常常会出现忘记怎么写的情况。

在一定程度上，机器学习的过程也一样，最初都是从简单的分类问题开始：判断输入的图像是猫还是狗，交易是真实的还是欺诈……这些任务非常有用。但是，hardmaru认为，更有趣的任务是生成数据，在hardmaru看来，生成数据是数据分类的延伸和扩展。相比能够认出某个汉字，能够把这个汉字写出来表明我们对这个汉字有更多的理解。同理，**生成内容也是理解内容的关键**。

生成对抗网络（GAN）在生成数据方面有着优异的表现，机器翻译也算是一类生成数据的例子。但hardmaru想生成的是矢量数据。因为他认为很多内容都更适合用矢量的形式来表达，比如用数码笔画的素描、CAD设计、科学实验数据等等。

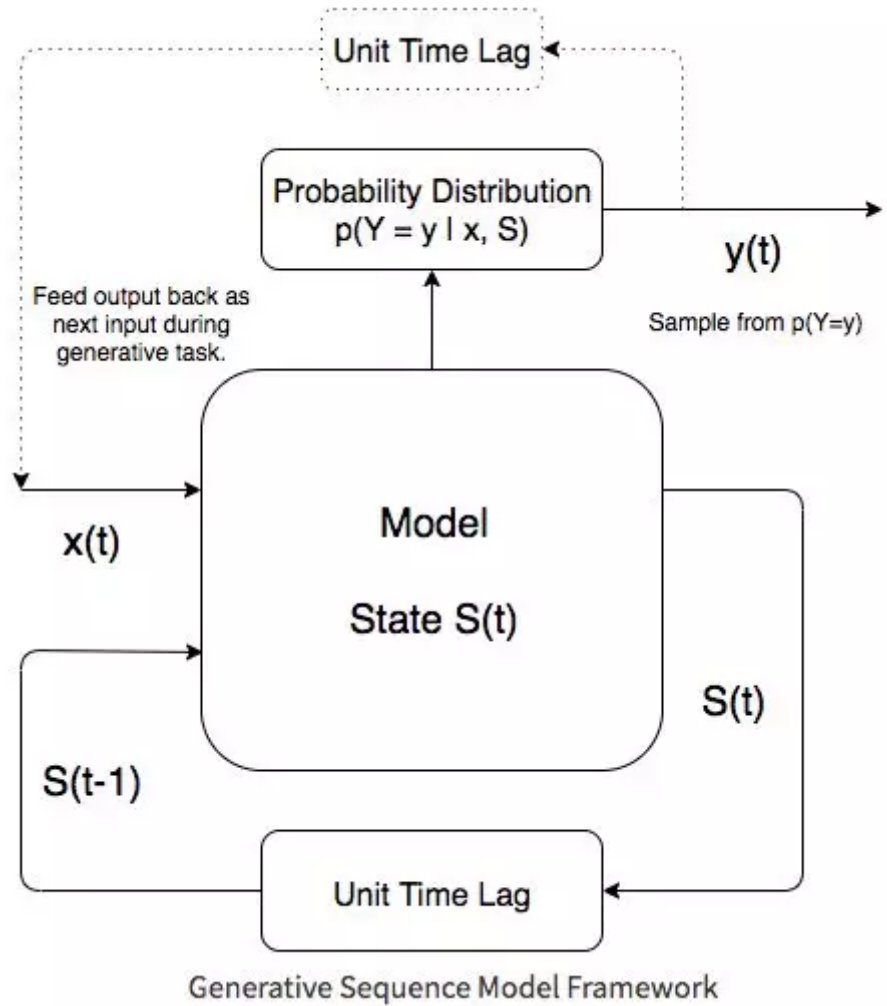


字体和笔画也更适合用矢量来表示。精心设计的TrueType字体，不管大小，显示出来都很美丽。

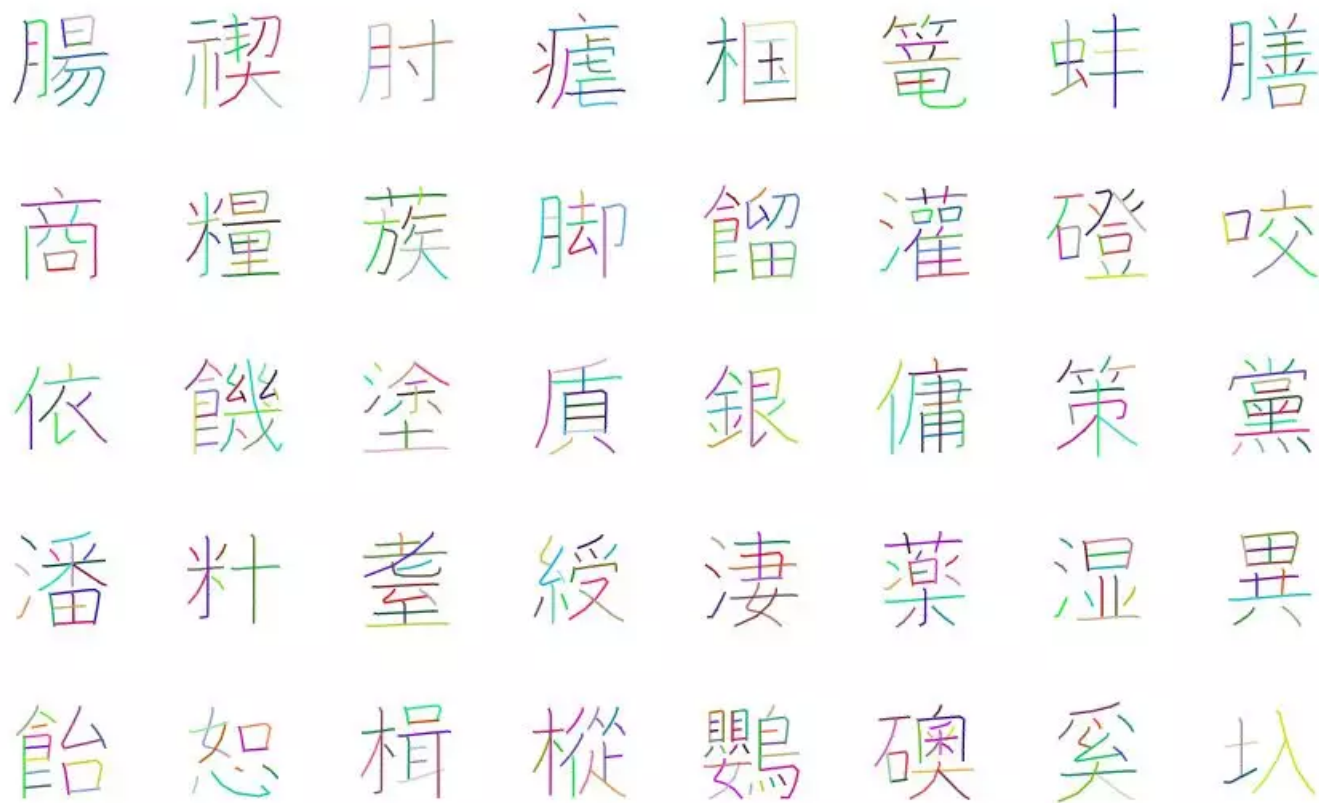
用 Sketch-RNN 新造一本《新华字典》

接下来，我们将介绍hardmaru如何使用RNN生成矢量格式的手写体汉字。汉字以矢量保存（SVG格式）。

hardmaru实现的是一个生成“新造”汉字的网络sketch-rnn，与Graves手写体生成模型框架（见下）类似。

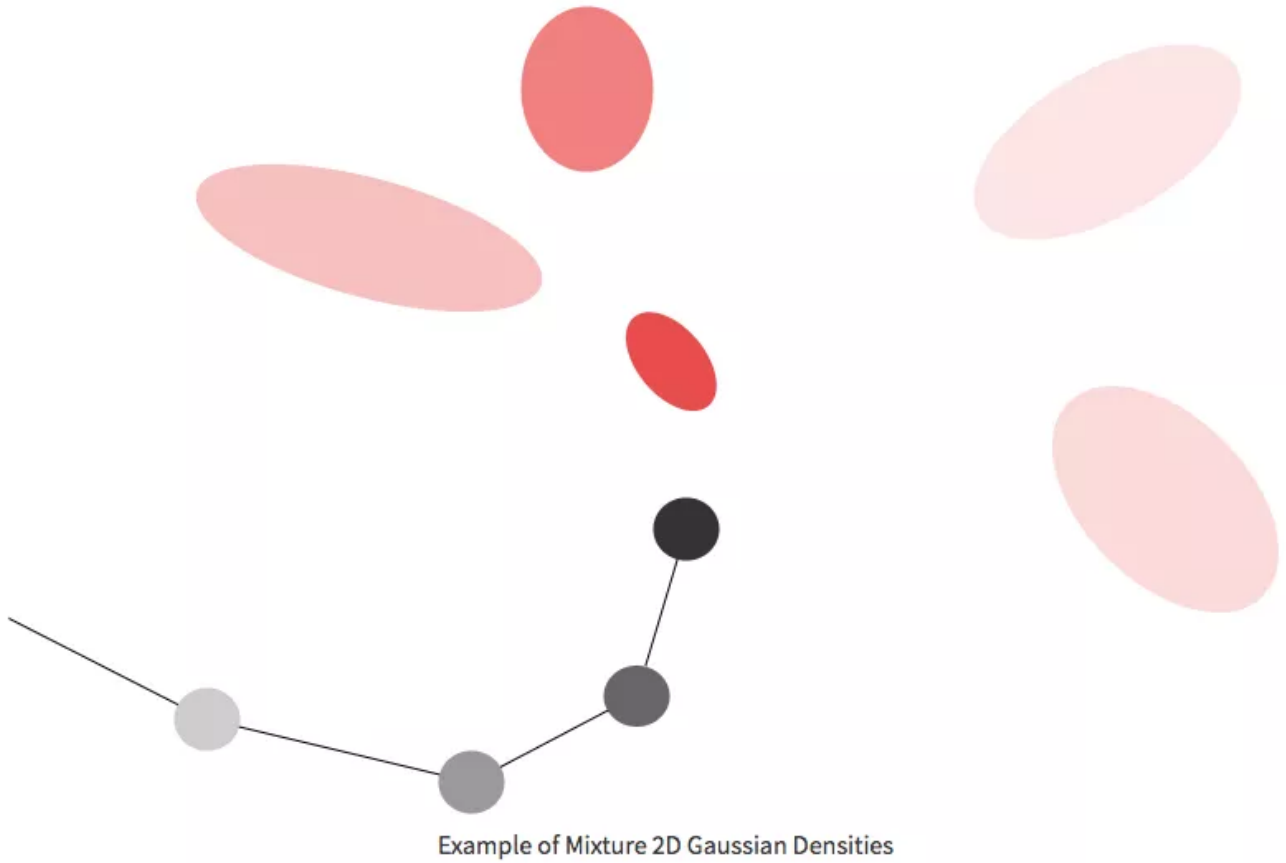


用于训练的数据是真实的汉字，并且包含了笔画顺序。因此，神经网络生成的汉字看上去也是按照一定程度上合理的笔画顺序来的。

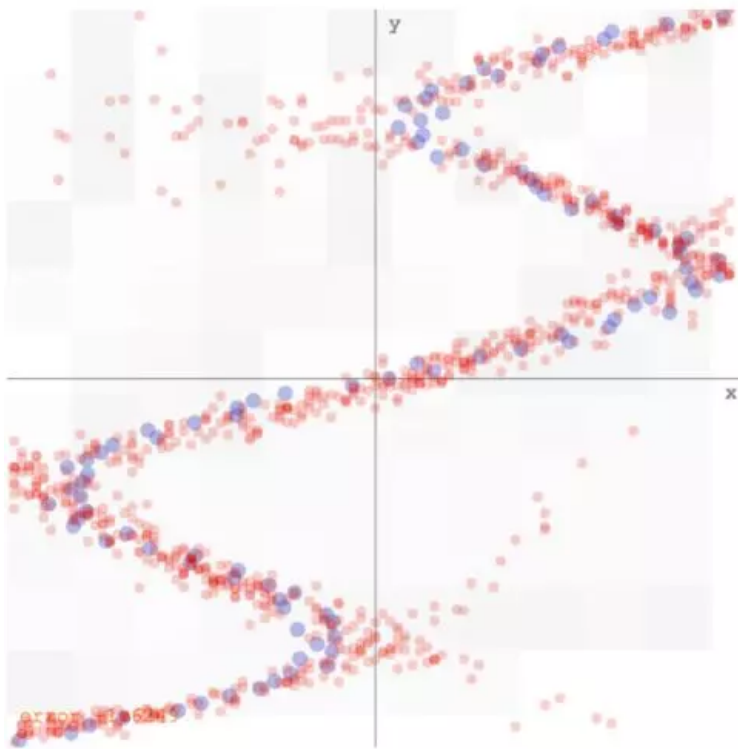


训练数据样本，不同的颜色代表了笔画顺序，来源于KanjiVG数据集

在sketch-rnn中，每一笔都用类似笔画的数据建模，其中每一步数据都包含x和y轴的偏移量，以及这一笔是落在纸上还是没有落在纸上，如果落在纸上，那么上一笔和这一笔之间就会有连线。神经网络必须为下一步提供概率分布。这个概率分布不是离散的，而是连续分配x轴和y轴上的偏移量，以及笔在下一步在纸上抬起的概率（也即笔画结束的概率）。sketch-rnn使用混合高斯分布来估算下一笔的位移。这个**用来生成笔迹的方法叫做混合密度网络（Mixture Density Networks, MDN）**。



以上是使用混合高斯密度来生成汉字笔划的一个例子。黑点代表在写字过程中连起来的线，**LSTM + MDN算法将持续估计下一个点出现位置的概率分布**。这个分布被建模成混合高斯分布。这意味着下一个位置是许多不同位置的混合（深浅不同的红色椭圆），并且每个位置本身都是x轴和y轴偏移的二维联合高斯分布，每个偏移都有自己的位置 2×2 协方差矩阵。



```

Elements Console » 1 > ⚙️ 🖨️ ✕
<top frame> ▾ Preserve log
-0.28906362107708583 mixture.js:237
-0.050954108297478984 mixture.js:237
0.34312173394390594 mixture.js:237
-0.41966673273739064 mixture.js:237
0.39284045189262884 mixture.js:237
0.32328749606059026 mixture.js:237
avg numerical gradient percentage mixture.js:243
error (0.01 = 1%): 0.4152738196036589
mixCoef: mixture.js:235
0.4360788283661583 mixture.js:237
0.22575440861041537 mixture.js:237
0.04071747929024179 mixture.js:237
0.2395673508237904 mixture.js:237
0.0021232251353523516 mixture.js:237
0.055758707774041856 mixture.js:237
0.01914389421798698 mixture.js:237
0.014880720136775554 mixture.js:237
0.029934940321267915 mixture.js:237
0.009379610454451381 mixture.js:237
0.00010925997532736349 mixture.js:237
0.059488294160872035 mixture.js:237
-0.22531898652113025 mixture.js:237
-0.00711399229298245 mixture.js:237
0.35902088173602237 mixture.js:237
-0.45142112211638485 mixture.js:237
0.36431691505009534 mixture.js:237
0.2746531028133383 mixture.js:237
avg numerical gradient percentage mixture.js:243
error (0.01 = 1%): 0.000004829860363623223



```

MDN轨迹展示

除了笔划的位置分布和结束概率之外，还需要对写完整个汉字的概率进行建模，也即结束字符“end-of-char”概率。但是，每个笔画完结的概率跟整个汉字完结的概率有一定重复，hardmaru花了不少功夫尝试对上述两个信号（笔划完结概率、字符完结概率）建模。最终，他通过神经网络中的softmax层将笔的状态建模为一组离散的状态。笔的状态分为三种：笔画结束、字符结束、落笔。模型会计算每一步三种状态的概率。

LSTM+MDN基本上是LSTM+Softmax的扩展，hardmaru以后想尝试更强大的方法。GAN（生成对抗网络）也许能应用到循环网络上，但他预计训练LSTM GAN会非常困难。

除了上面展示的各种例子，这是已有的一些有趣的结果，hardmaru自己做了“注释”：

 six stoned ladies	 lonely ghost	 wooden food	 urban sheep
 wood pecker	 stop eating lambs	 bird hunting	 educated horse
 listening bird	 listerine	 new type of wooden house	 lucky horse

还有一些不知道怎么描述的结果：

槽 哇 蹕 忌
菱 璫 榎 揆
安 萱 捏 脊

你可以在这个网站自己尝试：<http://otoro.net/kanji-rnn/>

了解更多

1. Sketch-rnn: <https://github.com/hardmaru/sketch-rnn/>