工具复现文档

191250215 周子杰

复现相关解说视频链接

> 百度网盘

链接：https://pan.baidu.com/s/1AcWFUUAoqsiFo2RyERdx6w
提取码：itof

> 南大box（可以直接看）

https://box.nju.edu.cn/f/9cdb229d64704124b748/

论文阐述

我进行工具复现的文章是 Automatic Text Input Generation for Mobile Testing。它的核心思想是，在自动化移动测试领域，文本的自动生成是一个技术挑战：面对一个搜索栏，我们到底应该生成什么样的文字才能符合需求呢？如果在音乐搜索的地方我们应该输入歌手名字，在电影搜索则应该输入电影名称；如果这些地方搞混了，自动测试就会因为啥也搜不出来而陷入僵局。所以我们可以把任务细化提取：首先得到当前搜索栏搜索的具体目标集合，再通过这个信息生成适宜的搜索文本。

这篇论文从monkey测试说起。谈到文本生成，monkey肯定是一种方案。monkey在生成动作序列上是可行的：让自动化测试工具生成随机的动作序列，例如单击某个UI组件等，就能让测试对象运行起来。但原始的monkey完全基于随机文本生成，就如同一只猴子坐在电脑前随机敲击键盘留下无意义的文字。想象一个场景：我们的IOS自动化测试工具点击了“电影”标签，并根据电影“movie”一词为上文生成新的文本用于搜索。此时我们期望的是类似于“Harry Potter”之类的电影名称，而不是类似于今天的天气或者一家餐馆的名字。因此完全随机的输入是不可行的。

这篇文章提出的RNN模型，即循环神经网络，就是一种基于上文生成文本的方法。

利用RNN能一定程度上解决随机生成输入带来的困扰，但问题依然存在：考虑“movie”和“film”两个同义词，是我们在直接利用RNN的时候难以避免的，因此我们又引入了再一个新的算法：word2vec，一个近义词处理模型。通过训练，该模型可以将单词转化为数据化的向量，从而借助该向量梳理清楚单词间的近义词关系。虽然该模型无法处理一词多义的情况，但已经足够满足我们的需求了。

在介绍了大概思路之后，这篇文章花了一定篇幅介绍了RNN和Word2Vector的算法。在这之后，文章又介绍了工具的使用效果，提出了几个问题，例如标签类型向image扩展，本工具与其他方法相比的有效程度等等。

RNN和Word2Vec的具体介绍我会在下文的核心算法板块详细叙述，此处就不赘述了。

工具复现

功能模块交互

我有四大功能模块：数据集预处理，word2vec训练，RNN训练和Selenium自动化测试。

他们的交互也并不复杂：数据集被载入并预处理后流入word2vec训练模型，获得所需单词的向量形式；然后经过word2vec进一步加工的数据集流入RNN进行正式的predict next训练（没有前一步加工也是能预测的，只是因为没有近义词帮助，RNN的效果预估会差一些）。在predict next的过程中，数据形态也要进行不断转换。训练完后，程序等待控制台输入，它会根据控制台输入的关键词预测下文。

数据集预处理

- 首先需要正常的文本文件，到达一定量之后他们的效果都不会有太大差别。准备好.txt后，我又查了很多资料，我直接选择使用了punkt库的数据集，他应该在word2vec的训练上很合适。那个时候我忘记了这次我要实现的核心效果是预测电影名称，而在数据集的选择上聚焦于了日常用语。这是后话。可我的荆棘之路就此开启。我在加载训练模型这一步就遇到了问题，能够下载nlkt的punkt数据包的官网网址不出意外是服务器崩了，尝试了各种方法还是打不开，最后好不容易换着方法搜索，找到了好心人提供的百度网盘链接。可百度网盘的下载速度大家都知道。经过漫长的等待，终于成功下载了这个数据包。

word2vec训练

- 一上来解决了老半天版本问题，结果又遇到了困难：我的电脑显卡是Intle的iris而不是N卡，跑tensoflow只能用CPU版本而不是GPU版本。最开始我没有意识到这是个问题，以为顶多是跑得慢一点要多花几天，直到经过一天奋斗，我发现我还是没能解决我memory error的问题：小破电脑的8G内存用尽方法也加载不进整个word2vec的训练集。我想了很多，虚拟内存，虚拟机，去租一台服务器。前两者的尝试都失败了，我的华为云有优惠券，但弹性云服务器部署环境也没能成功。于是我痛定思痛，经过筛选只保留了部分训练集，而把默认不常见的内容抛弃了。
- 过程里还遇到了更多的问题，例如库新旧版本的交替等。总之最后，终于大抵上实现了这个步骤，能成功跑出词语的向量表达了。

RNN训练

- RNN能找到的资料相对较多，过程也相对顺利。我选择了LSTM这种RNN，因为库的存在顺利了很多。调参数是比较苦恼的过程，为了让我的结果看上去像个样子，加上数据集被我缩减，但同时要考虑跑一次的调试成本问题，常常跑一次要花很久很久，却最终以报错告终，还有通宵跑但早上起来发现电脑死机的情况。这些事都很让人头疼。写RNN的过程依旧遇到了大量的库版本问题，好在最后也都解决了。

Selenium自动化测试

- 众多自动化测试工具中，我最终选择了用python的selenium库进行自动化测试。我会唤起浏览器，打开电影搜索网址并输入我自己生成的电影相关文本进行搜索。
- 最开始的效果有点惊悚。我看着生成的电影名称：covered. 我总觉得，这效果可能还不如monkey呢。痛定思痛，我把原因归结给了我的训练集：我使用的是高频词库，而没有针对电影，音乐，游戏等特定场景训练特定模型，加上我因为自己电脑性能对原本的训练集进行了消减，效果一定雪上加霜。于是我手动添加了一些训练集，但最后的预测还是失败了。在我的一通操作之后，得到的结果从covered变成了infuriated，但我觉得，他们俩谁也不像个电影名字。
- 我手动添加的训练集也分了几个步骤；首先我想上网找到大量电影名称相关的数据集，但没能找到合适的，因此只能自己生造，生造也就只集中在哈利波特这一部电影上；很大一部分原因是我进行到这一步的时候已经在这个作业上耗了快一周，于是最后实在是没有精力去找到最合适的数据集了。
- 在手动添加训练集之后还是无法得出看上去靠谱一点的预测，我认为问题有可能出现在RNN部分但也有可能出现在word2vec部分，于是选择了断点调试。可大数据相关代码和我平常写的代码差距太大了，断点处都是无法观察出所以然的海量数据，因此最终也没能找出问题，算是一个遗憾。

运行要求

点击运行，在word2vec训练和RNN训练完成后，输入前馈文本“movie”或者可能存在的标签例如“I want to look for movie”。然后点击回车，selenium会自动唤起Google Chrom浏览器（运行的电脑需要有这个浏览器且安装了对应版本的Chrome Driver才能达到效果），进入豆瓣电影网页，输入生成的预测值，并点击确认获得期望的搜索结果。程序最后在页面停留一定时间后会自动关闭浏览页面并结束运行。

复现痛点

复现痛点这里我有很多话可以说，基本都在上文对四个功能模块的叙述里提到了。因为没有找到作者源码，而对我的能力而言这个工具写起来真的太困难了。到了最后即使花了很多天努力解决一个个遇到的困难，还是遗留下一些完全无从下手的问题。但无论如何，我的复现是能够基本模拟论文提到的场景的，也希望接下来通过进一步学习我能更加完善自己的工具。

工具验证

在自己的电脑上尝试了不同输入，观察被浏览器唤起，生成的相应文本被搜索。如果无搜索结果，则表明这一次运行是不成功的。

由于将迭代次数提升之后运行一次耗时实在太长，我主要在低迭代次数的情况下进行了工具验证和调试。结果不算理想，生成的输出总是文不对题。但我真的真的已经尽力了。

工具理解

核心算法

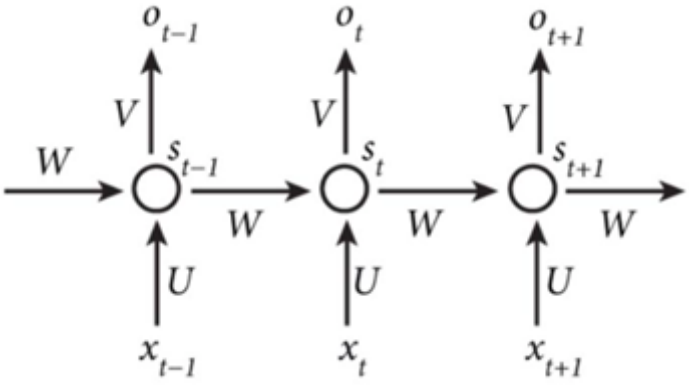
核心算法总共有三个部分：论文里提到的RNN，Word2Vec和我自己选择使用的selenium，下面会进行逐一介绍。

RNN

RNN是老生常谈的神经网络里的一种，叫做循环神经网络。

先来谈谈我对神经网络的理解。神经网络其实没有那么玄乎，它归根结底也只是一张图而已。它拥有多层节点，最简单的情况下通过输入层，隐藏层和输出层三个层级就能拟合理论上的任何函数。以神经网络运用最广泛的分类领域为例，我们期望输入一张图片，输出告诉我它是小猫还是小狗。有了预期的输入与输出，这就是一个函数。在一些简单的函数里，我期望输出x，输出是2x，这时在函数内部就需要根据我的输入进行一步计算：将我的输入乘以二然后返还给我。到了神经网络，这步工序会变得复杂很多：他需要各个输入进行复杂的相互作用与运算，而这些运算就发生在隐藏层的节点上。通过不断训练，反馈，调节参数与权重，就能得到合适的“函数”来解决我们的需求。

下面介绍RNN。它的核心思想也很简单，即前馈信息会对当前决策造成影响。研究普通神经网络的构建就能发现，他的每次计算之间是无法相互作用的，因而我们引入了RNN，而我们确实知道在自动化文本生成这件事上，前馈信息是重要的，如下图所示，让上一时刻的隐藏层影响这一时刻的隐藏层就是我们需要实现的核心。



上一时刻隐藏层W被保留下来，和U一起作为了下一时刻计算的输入，影响到下一时刻神经网络的计算和决策，在我的理解里有着类似隐马尔可夫模型的前馈思维，但比隐马尔科夫模型更加精确，拥有更好的拟合能力。

Word2Vec

在自然语言处理问题中，我们遇到的一个问题就是，一般的数学模型也只接受数值类型的输入，因此我们要想办法用数值类型代表自然语言。首先使用的是one-hot encoder这种最显而易见的方法，即这是哪个词就把哪个位置置为1，其余位置全为0。这种表示方法是不需要通过训练就能得到的，但会造成维数灾难。

而word2vec是另一种把单词化为向量的方法，获得向量的维数会比one-hot-encoder少很多，但它需要基于上下文等做一些训练才能得出。我们的word2vec模型其实类似神经网络，输入one-hot向量，利用隐含层的激活与权重运算完成本质上为降维的word2vec操作。这里的“降维”是基于语义的，而语义来自上下文。

我们有两种训练word2vec模型的方式，即CBOW和Skip-gram。他们的预测方向不同，前者从上下文预测该位置，而后者从自身预测上下文。

selenium

比起算法它更应该被称为一个工具。它是一个应用的自动化测试工具，从唤起到键盘输入到按钮点击等人为事件都能进行模仿，运行起来就像真正有用户在操作一样。它支持多种平台和浏览器，在本次工具复现里我选择了最主流的Google Chrom，并通过python控制其运行。我利用它直接唤起在豆瓣电影网站，并将前文生成的文本输入搜索栏进行搜索，实现全流程的自动化运行。

功能模块

四大功能模块：数据集预处理，word2vec训练，RNN训练和Selenium自动化测试。上文已经详细说过，在此不赘述。

数据流动与交互

首先我们将数据集载入内存。在一些简单的预处理后，我们的数据集流向word2vec模块并接受训练完成单词向量模型的生成。其次我们构建RNN模型。注意在这里的word2vec其实主要是“优化RNN模型”的作用，让其在遇到类似movie和film的同义词时也能良好工作。但其实即使没有它，RNN本身也能作为文本生成工具工作的。在两个模型都准备好以后，我们有了第二个数据入口：输入的文本内容，而我们的目的是为其生成下一个最有可能的词。此时我们的输入来源于对软件、网页等关键词的抓取，而输出直接流入自动化控制的搜索框中。在我的工具复现里，未能实现在UI组件里抓取单词，而是采用了终端手动输入的方式，但自动唤起浏览器并在浏览器搜索框中输入生成文本的输出流并未改变。

输入输出

在输入上有两处数据入口：一是加载训练集数据，二是模型训练完毕后从控制台获取输入，输入作为上文标签，根据输入预测下文。

有一处数据出口：即根据输入预测的下文。数据以唤起浏览器-》打开浏览器搜索窗口-》在搜索窗口里输入预测的下文的形式被输出表示。

references

Automatic Text Input Generation for Mobile Testing

写在最后：这次工具复现对我的打击和提升都是全方面的，一切故事还得从作者没有公布源代码讲起。我最后弄出了一个能跑的工具，但这个工具跑出来的结果一言难尽，直到最后也遗憾地没能成功修复结果的低准确性；论文不仅有python单纯的输入输出处理，还涉及到了自动化测试操纵移动应用UI组件，我最后不得不用了selenium代替，在浏览器上完成了最终效果展示，天地良心即使我暑假才学了安卓开发，让我在我的手机上抓取元素组件然后自动操控对我而言实在是太难了。所以退一步选择了浏览器，这里面也有我很长时间的思考和选择。

其实之前做过数据科学这门课的大作业，选修过计科的模式识别，写过一些大数据这门课的作业，但没有哪一次是对这方面的工作有如此直观的体会的，从电脑带不动到数据找不到；从找不到API到调不好数据格式；从疯狂报错到能跑但跑出错误结果；从数据集大到要跑好多好多小时最后还memory error了到消减了数据集然后找不到目标单词；一切的一切连带这门课会不会挂掉的恐惧折磨了我好长一段时间，我只能安慰自己这叫做成长。还是觉得以后自己有能力了的话可以再优化一下它。