**Tensorflow-3-使用RNN生成中文小说**

2017年06月11日 07:25:31

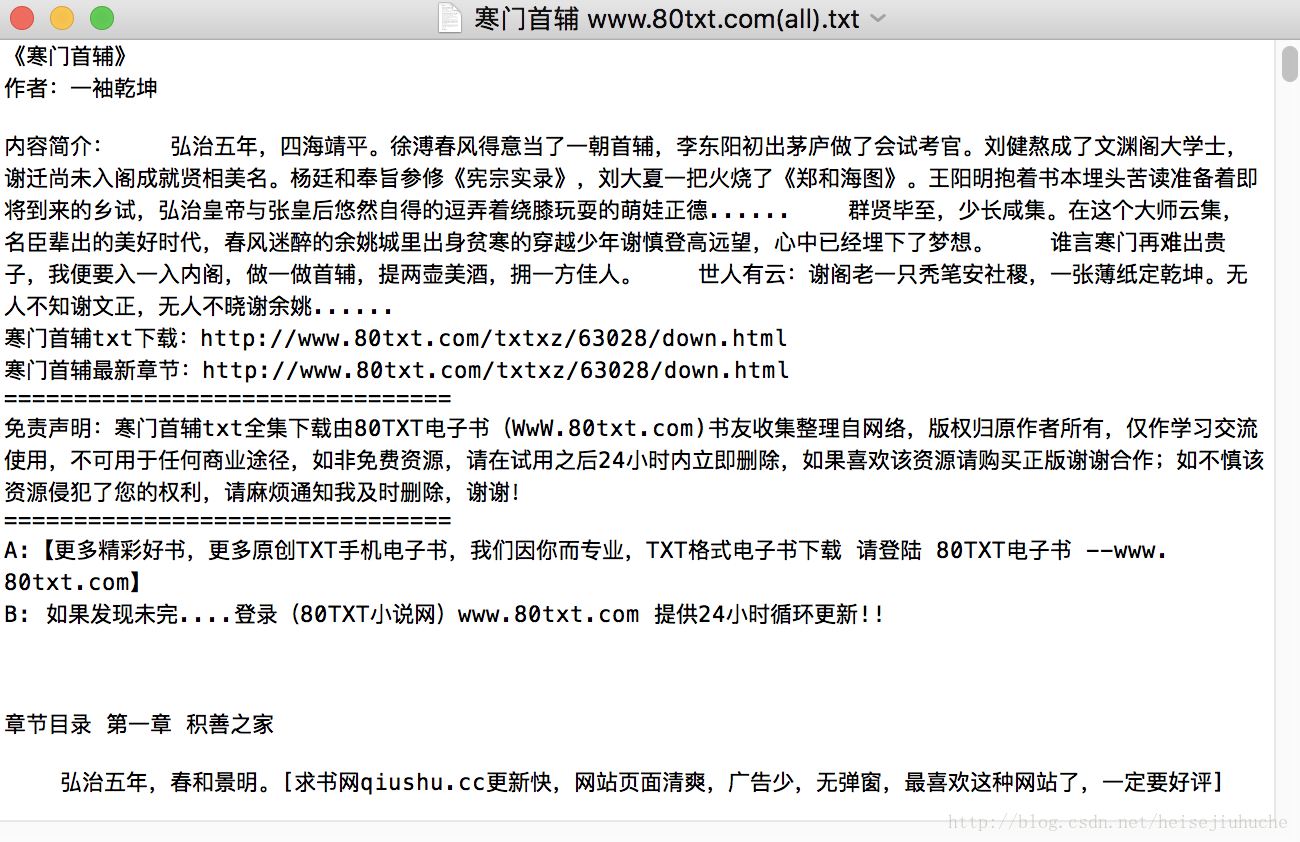
阅读数：7933

数据选取的是起点中文网的获奖历史小说『寒门首辅』，作者一袖乾坤。

以下就是整个应用过程。

来试试用起点中文网的历史小说『寒门首辅（一袖乾坤 著）』来做训练数据，看看这个RNN网络能产生一些什么样子的文本。 尝试过程中必遇到问题，也借此加深一些对RNN的理解。

首先我从网上下载到了『寒门首辅』的txt版本，打开时候发现有很多空行，还包含了很多不必要的链接，看起来是这样的。



预处理一下数据。

import helper

* 1

读入数据

dir = './data/寒门首辅.txt'

text = helper.load\_text(dir)

* 1
* 2

设置一下要用多少个字来训练，方便调试。这里先用100000字进行训练

num\_words\_for\_training = 100000

text = text[:num\_words\_for\_training]

* 1
* 2
* 3

看看有多少行

lines\_of\_text = text.split('\n')

print(len(lines\_of\_text))

Out: 4329



先看看前15行是什么内容

print(lines\_of\_text[:15])

Out: ['《寒门首辅》', '作者：一袖乾坤', '', '内容简介： 弘治五年，四海靖平。徐溥春风得意当了一朝首辅，李东阳初出茅庐做了会试考官。刘健熬成了文渊阁大学士，谢迁尚未入阁成就贤相美名。杨廷和奉旨参修《宪宗实录》，刘大夏一把火烧了《郑和海图》。王阳明抱着书本埋头苦读准备着即将到来的乡试，弘治皇帝与张皇后悠然自得的逗弄着绕膝玩耍的萌娃正德...... 群贤毕至，少长咸集。在这个大师云集，名臣辈出的美好时代，春风迷醉的余姚城里出身贫寒的穿越少年谢慎登高远望，心中已经埋下了梦想。 谁言寒门再难出贵子，我便要入一入内阁，做一做首辅，提两壶美酒，拥一方佳人。 世人有云：谢阁老一只秃笔安社稷，一张薄纸定乾坤。无人不知谢文正，无人不晓谢余姚...... ', '寒门首辅txt下载：http://www.80txt.com/txtxz/63028/down.html', '寒门首辅最新章节：http://www.80txt.com/txtxz/63028/down.html', '================================', '免责声明：寒门首辅txt全集下载由80TXT电子书（WwW.80txt.com)书友收集整理自网络，版权归原作者所有，仅作学习交流使用，不可用于任何商业途径，如非免费资源，请在试用之后24小时内立即删除，如果喜欢该资源请购买正版谢谢合作；如不慎该资源侵犯了您的权利，请麻烦通知我及时删除，谢谢！', '================================', 'A:【更多精彩好书，更多原创TXT手机电子书，我们因你而专业，TXT格式电子书下载 请登陆 80TXT电子书 --www.80txt.com】', 'B: 如果发现未完....登录（80TXT小说网）www.80txt.com 提供24小时循环更新!!', '', '', '', '章节目录 第一章 积善之家']

* 1
* 2
* 3

把『章节目录』之前的行全部砍掉，一大堆没用的东西。

lines\_of\_text = lines\_of\_text[14:]

* 1

再来看看，第一行应该就进入正题了。

print(lines\_of\_text[:5])

Out: ['章节目录 第一章 积善之家', '', ' 弘治五年，春和景明。[求书网qiushu.cc更新快，网站页面清爽，广告少，无弹窗，最喜欢这种网站了，一定要好评]', '', ' 浙江承宣布政使司，绍兴府，余姚县。']

* 1
* 2
* 3

我查看了一下，这个小说一共有129万字左右。   
先把空行去掉吧。去掉空行之后应该就只有一半左右的行数了。

lines\_of\_text = [lines for lines in lines\_of\_text if len(lines) > 0]

print(len(lines\_of\_text))

Out: 2158

* 1
* 2
* 3
* 4
* 5

打印前20行看看什么情况

print(lines\_of\_text[:20])

Out: ['章节目录 第一章 积善之家', ' 弘治五年，春和景明。[求书网qiushu.cc更新快，网站页面清爽，广告少，无弹窗，最喜欢这种网站了，一定要好评]', ' 浙江承宣布政使司，绍兴府，余姚县。', ' 县城里靠近城隍庙的一处小巷口，一个年约十二，身着淡蓝色粗布长衫，头戴黑色幞头的少年望着不远处熙熙攘攘的人群不发一言。', ' 他叫谢慎，是土生土长的余姚人。但是也不尽然，因为他的灵魂来自后世，是穿越而来夺舍附着在这个与他同名同姓的少年身上。事情的经过他并不是很清楚，只知道这个少年应该是不慎落水，被人救上后就一直昏迷不醒，奄奄一息，直到自己穿越才鸠占鹊巢，成了这具身体的主人。', ' 不过有些奇特的是，谢慎还兼有原先身体里的一部分记忆，他拼命检索这才对身处的环境有了一个大概的认识。', ' 如今是弘治五年，当今天子朱祐樘立志中兴，招贤纳士，敕令各省提学官巡视省内各府、州、县学，选拔参加乡试的人选。今年是乡试之年，明年则是会试，殿试连着两场大试，是出进士的年份。朝为田舍郎，暮登天子堂，在明朝读书自然是最有前途的事情。不过这似乎和他没有什么关系，虽然原先的谢慎也算是个读书人，可却并没有功名在身，最多只能算一个半吊子童生。', ' 谢慎前世可是苦修明史的研究生，虽然主攻方向是嘉靖万历两朝，但对弘治朝多少也有些了解。浙江一直是科举强省，在弘治朝也是绝对的霸主，而绍兴府则是浙江省中出进士最多的府。mianhuatang.cc [棉花糖小说网]若要落在县一级，那魁首非余姚莫属。山阴，会稽两县加在一起，所中的进士数目都不比余姚的多。这么说来，余姚绝对是科举的死亡之组了。王阳明，谢迁，王华，一想到余姚出的那些名人，谢慎便免不了自嘲。', ' 谢慎虽然前世苦修明史，对八股文也有所研究，但要让他在明代科场竞争最恐怖的余姚县脱颖而出，确实有些艰难。', ' 可当他垂下头看了看自己瘦弱的身体，不免摇了摇头，长叹了一声。', ' 不科举还能干什么呢，就他这副竹竿子般的身体，肩不能抗，手不能提，杀鸡没准还被鸡调戏......', ' 何况在这个万般皆下品惟有读书高的年代，不科举就只能为民，不管是务农还是经商终归都是被官府压着，没有出头的机会。大明朝那个著名的沈万三不就是最好的例子，家财万贯富可敌国还不是随意一个罪名就成了别人茶余饭后唏嘘慨叹的谈资？', ' 读书，还是得读书，便前方是刀山火海硬着头皮也得上。他才十二岁，还有可塑性......', ' “小郎，你怎么在这儿呢，快快随我回家去。你落水后被救起身子本就虚弱，若是在此时再染上了风寒，那可了不得。你嫂嫂给你特地煮了鸡汤，你多喝上几碗也好补补身子。”', ' 一个中年男子的声音打断了谢慎的沉思，他抬了抬头见来人是自己的胞兄谢方，心中不由得泛起一抹暖意。', ' 谢家门丁不旺，到了他们这一代只有他和谢方二人，不过这个便宜大哥似乎对他还不错，吃穿用度上绝不短着缺着。', ' 谢方在余姚县城里开着一家茶铺，将每年收上的茶叶运到县城售卖，质地更好的一些会有人收走在府城售卖。', ' 明代浙江绍兴府各县几乎都种植茶叶。最著名的要数杭州府钱塘县的龙井茶以及宁波府象山县的珠山茶。至于余姚虽然也种植茶树，但其出产的茶叶并不像其种植棉花所产的“浙花”那么出名。', ' 不过似乎余姚本地的百姓很喜欢这种清淡的姚江茶，其在本县的销路并不差。', ' 靠着辛勤的工作，谢方也算撑起了一个家。在谢方的一再坚持下，谢慎从小便被送去开蒙，如今已经学了六年了。不过原先的谢慎最多只能算是中上之资，跟神童绝对沾不上边，照着原有轨迹发展下去，能不能取得秀才功名都是一个问题。']

* 1
* 2
* 3

下一步，把每行里面的『空格』，『[]里的内容』，『<>里的内容』都去掉。

# 去掉每行首尾空格

lines\_of\_text = [lines.strip() for lines in lines\_of\_text]

* 1
* 2

看下情况如何，打印前20句话。

print(lines\_of\_text[:20])

Out: ['章节目录 第一章 积善之家', '弘治五年，春和景明。[求书网qiushu.cc更新快，网站页面清爽，广告少，无弹窗，最喜欢这种网站了，一定要好评]', '浙江承宣布政使司，绍兴府，余姚县。', '县城里靠近城隍庙的一处小巷口，一个年约十二，身着淡蓝色粗布长衫，头戴黑色幞头的少年望着不远处熙熙攘攘的人群不发一言。', '他叫谢慎，是土生土长的余姚人。但是也不尽然，因为他的灵魂来自后世，是穿越而来夺舍附着在这个与他同名同姓的少年身上。事情的经过他并不是很清楚，只知道这个少年应该是不慎落水，被人救上后就一直昏迷不醒，奄奄一息，直到自己穿越才鸠占鹊巢，成了这具身体的主人。', '不过有些奇特的是，谢慎还兼有原先身体里的一部分记忆，他拼命检索这才对身处的环境有了一个大概的认识。', '如今是弘治五年，当今天子朱祐樘立志中兴，招贤纳士，敕令各省提学官巡视省内各府、州、县学，选拔参加乡试的人选。今年是乡试之年，明年则是会试，殿试连着两场大试，是出进士的年份。朝为田舍郎，暮登天子堂，在明朝读书自然是最有前途的事情。不过这似乎和他没有什么关系，虽然原先的谢慎也算是个读书人，可却并没有功名在身，最多只能算一个半吊子童生。', '谢慎前世可是苦修明史的研究生，虽然主攻方向是嘉靖万历两朝，但对弘治朝多少也有些了解。浙江一直是科举强省，在弘治朝也是绝对的霸主，而绍兴府则是浙江省中出进士最多的府。mianhuatang.cc [棉花糖小说网]若要落在县一级，那魁首非余姚莫属。山阴，会稽两县加在一起，所中的进士数目都不比余姚的多。这么说来，余姚绝对是科举的死亡之组了。王阳明，谢迁，王华，一想到余姚出的那些名人，谢慎便免不了自嘲。', '谢慎虽然前世苦修明史，对八股文也有所研究，但要让他在明代科场竞争最恐怖的余姚县脱颖而出，确实有些艰难。', '可当他垂下头看了看自己瘦弱的身体，不免摇了摇头，长叹了一声。', '不科举还能干什么呢，就他这副竹竿子般的身体，肩不能抗，手不能提，杀鸡没准还被鸡调戏......', '何况在这个万般皆下品惟有读书高的年代，不科举就只能为民，不管是务农还是经商终归都是被官府压着，没有出头的机会。大明朝那个著名的沈万三不就是最好的例子，家财万贯富可敌国还不是随意一个罪名就成了别人茶余饭后唏嘘慨叹的谈资？', '读书，还是得读书，便前方是刀山火海硬着头皮也得上。他才十二岁，还有可塑性......', '“小郎，你怎么在这儿呢，快快随我回家去。你落水后被救起身子本就虚弱，若是在此时再染上了风寒，那可了不得。你嫂嫂给你特地煮了鸡汤，你多喝上几碗也好补补身子。”', '一个中年男子的声音打断了谢慎的沉思，他抬了抬头见来人是自己的胞兄谢方，心中不由得泛起一抹暖意。', '谢家门丁不旺，到了他们这一代只有他和谢方二人，不过这个便宜大哥似乎对他还不错，吃穿用度上绝不短着缺着。', '谢方在余姚县城里开着一家茶铺，将每年收上的茶叶运到县城售卖，质地更好的一些会有人收走在府城售卖。', '明代浙江绍兴府各县几乎都种植茶叶。最著名的要数杭州府钱塘县的龙井茶以及宁波府象山县的珠山茶。至于余姚虽然也种植茶树，但其出产的茶叶并不像其种植棉花所产的“浙花”那么出名。', '不过似乎余姚本地的百姓很喜欢这种清淡的姚江茶，其在本县的销路并不差。', '靠着辛勤的工作，谢方也算撑起了一个家。在谢方的一再坚持下，谢慎从小便被送去开蒙，如今已经学了六年了。不过原先的谢慎最多只能算是中上之资，跟神童绝对沾不上边，照着原有轨迹发展下去，能不能取得秀才功名都是一个问题。']

* 1
* 2
* 3
* 4

可以看到空格都没了。下一步用正则去掉『[]』和『<>』中的内容，像上面的什么『[棉花糖小说网]』这些的，后面还有一些是包含在『<>』里的，一并去掉。

import re

# 生成一个正则，负责找『[]』包含的内容

pattern = re.compile(r'\[.\*\]')

# 将所有指定内容替换成空

lines\_of\_text = [pattern.sub("", lines) for lines in lines\_of\_text]

* 1
* 2
* 3
* 4
* 5
* 6
* 7

打印看效果

print(lines\_of\_text[:20])

Out: ['章节目录 第一章 积善之家', '弘治五年，春和景明。', '浙江承宣布政使司，绍兴府，余姚县。', '县城里靠近城隍庙的一处小巷口，一个年约十二，身着淡蓝色粗布长衫，头戴黑色幞头的少年望着不远处熙熙攘攘的人群不发一言。', '他叫谢慎，是土生土长的余姚人。但是也不尽然，因为他的灵魂来自后世，是穿越而来夺舍附着在这个与他同名同姓的少年身上。事情的经过他并不是很清楚，只知道这个少年应该是不慎落水，被人救上后就一直昏迷不醒，奄奄一息，直到自己穿越才鸠占鹊巢，成了这具身体的主人。', '不过有些奇特的是，谢慎还兼有原先身体里的一部分记忆，他拼命检索这才对身处的环境有了一个大概的认识。', '如今是弘治五年，当今天子朱祐樘立志中兴，招贤纳士，敕令各省提学官巡视省内各府、州、县学，选拔参加乡试的人选。今年是乡试之年，明年则是会试，殿试连着两场大试，是出进士的年份。朝为田舍郎，暮登天子堂，在明朝读书自然是最有前途的事情。不过这似乎和他没有什么关系，虽然原先的谢慎也算是个读书人，可却并没有功名在身，最多只能算一个半吊子童生。', '谢慎前世可是苦修明史的研究生，虽然主攻方向是嘉靖万历两朝，但对弘治朝多少也有些了解。浙江一直是科举强省，在弘治朝也是绝对的霸主，而绍兴府则是浙江省中出进士最多的府。mianhuatang.cc 若要落在县一级，那魁首非余姚莫属。山阴，会稽两县加在一起，所中的进士数目都不比余姚的多。这么说来，余姚绝对是科举的死亡之组了。王阳明，谢迁，王华，一想到余姚出的那些名人，谢慎便免不了自嘲。', '谢慎虽然前世苦修明史，对八股文也有所研究，但要让他在明代科场竞争最恐怖的余姚县脱颖而出，确实有些艰难。', '可当他垂下头看了看自己瘦弱的身体，不免摇了摇头，长叹了一声。', '不科举还能干什么呢，就他这副竹竿子般的身体，肩不能抗，手不能提，杀鸡没准还被鸡调戏......', '何况在这个万般皆下品惟有读书高的年代，不科举就只能为民，不管是务农还是经商终归都是被官府压着，没有出头的机会。大明朝那个著名的沈万三不就是最好的例子，家财万贯富可敌国还不是随意一个罪名就成了别人茶余饭后唏嘘慨叹的谈资？', '读书，还是得读书，便前方是刀山火海硬着头皮也得上。他才十二岁，还有可塑性......', '“小郎，你怎么在这儿呢，快快随我回家去。你落水后被救起身子本就虚弱，若是在此时再染上了风寒，那可了不得。你嫂嫂给你特地煮了鸡汤，你多喝上几碗也好补补身子。”', '一个中年男子的声音打断了谢慎的沉思，他抬了抬头见来人是自己的胞兄谢方，心中不由得泛起一抹暖意。', '谢家门丁不旺，到了他们这一代只有他和谢方二人，不过这个便宜大哥似乎对他还不错，吃穿用度上绝不短着缺着。', '谢方在余姚县城里开着一家茶铺，将每年收上的茶叶运到县城售卖，质地更好的一些会有人收走在府城售卖。', '明代浙江绍兴府各县几乎都种植茶叶。最著名的要数杭州府钱塘县的龙井茶以及宁波府象山县的珠山茶。至于余姚虽然也种植茶树，但其出产的茶叶并不像其种植棉花所产的“浙花”那么出名。', '不过似乎余姚本地的百姓很喜欢这种清淡的姚江茶，其在本县的销路并不差。', '靠着辛勤的工作，谢方也算撑起了一个家。在谢方的一再坚持下，谢慎从小便被送去开蒙，如今已经学了六年了。不过原先的谢慎最多只能算是中上之资，跟神童绝对沾不上边，照着原有轨迹发展下去，能不能取得秀才功名都是一个问题。']

* 1
* 2
* 3

『[]』的内容已经没了。下一步去掉『<>』中的内容，方法同上。

# 将上面的正则换成负责找『<>』包含的内容

pattern = re.compile(r'<.\*>')

# 将所有指定内容替换成空

lines\_of\_text = [pattern.sub("", lines) for lines in lines\_of\_text]

* 1
* 2
* 3
* 4
* 5

下一步，把每句话最后的『……』换成『。』。

# 将上面的正则换成负责找『......』包含的内容

pattern = re.compile(r'\.+')

# 将所有指定内容替换成空

lines\_of\_text = [pattern.sub("。", lines) for lines in lines\_of\_text]

* 1
* 2
* 3
* 4
* 5

打印看效果

print(lines\_of\_text[:20])

Out: ['章节目录 第一章 积善之家', '弘治五年，春和景明。', '浙江承宣布政使司，绍兴府，余姚县。', '县城里靠近城隍庙的一处小巷口，一个年约十二，身着淡蓝色粗布长衫，头戴黑色幞头的少年望着不远处熙熙攘攘的人群不发一言。', '他叫谢慎，是土生土长的余姚人。但是也不尽然，因为他的灵魂来自后世，是穿越而来夺舍附着在这个与他同名同姓的少年身上。事情的经过他并不是很清楚，只知道这个少年应该是不慎落水，被人救上后就一直昏迷不醒，奄奄一息，直到自己穿越才鸠占鹊巢，成了这具身体的主人。', '不过有些奇特的是，谢慎还兼有原先身体里的一部分记忆，他拼命检索这才对身处的环境有了一个大概的认识。', '如今是弘治五年，当今天子朱祐樘立志中兴，招贤纳士，敕令各省提学官巡视省内各府、州、县学，选拔参加乡试的人选。今年是乡试之年，明年则是会试，殿试连着两场大试，是出进士的年份。朝为田舍郎，暮登天子堂，在明朝读书自然是最有前途的事情。不过这似乎和他没有什么关系，虽然原先的谢慎也算是个读书人，可却并没有功名在身，最多只能算一个半吊子童生。', '谢慎前世可是苦修明史的研究生，虽然主攻方向是嘉靖万历两朝，但对弘治朝多少也有些了解。浙江一直是科举强省，在弘治朝也是绝对的霸主，而绍兴府则是浙江省中出进士最多的府。mianhuatang。cc 若要落在县一级，那魁首非余姚莫属。山阴，会稽两县加在一起，所中的进士数目都不比余姚的多。这么说来，余姚绝对是科举的死亡之组了。王阳明，谢迁，王华，一想到余姚出的那些名人，谢慎便免不了自嘲。', '谢慎虽然前世苦修明史，对八股文也有所研究，但要让他在明代科场竞争最恐怖的余姚县脱颖而出，确实有些艰难。', '可当他垂下头看了看自己瘦弱的身体，不免摇了摇头，长叹了一声。', '不科举还能干什么呢，就他这副竹竿子般的身体，肩不能抗，手不能提，杀鸡没准还被鸡调戏。', '何况在这个万般皆下品惟有读书高的年代，不科举就只能为民，不管是务农还是经商终归都是被官府压着，没有出头的机会。大明朝那个著名的沈万三不就是最好的例子，家财万贯富可敌国还不是随意一个罪名就成了别人茶余饭后唏嘘慨叹的谈资？', '读书，还是得读书，便前方是刀山火海硬着头皮也得上。他才十二岁，还有可塑性。', '“小郎，你怎么在这儿呢，快快随我回家去。你落水后被救起身子本就虚弱，若是在此时再染上了风寒，那可了不得。你嫂嫂给你特地煮了鸡汤，你多喝上几碗也好补补身子。”', '一个中年男子的声音打断了谢慎的沉思，他抬了抬头见来人是自己的胞兄谢方，心中不由得泛起一抹暖意。', '谢家门丁不旺，到了他们这一代只有他和谢方二人，不过这个便宜大哥似乎对他还不错，吃穿用度上绝不短着缺着。', '谢方在余姚县城里开着一家茶铺，将每年收上的茶叶运到县城售卖，质地更好的一些会有人收走在府城售卖。', '明代浙江绍兴府各县几乎都种植茶叶。最著名的要数杭州府钱塘县的龙井茶以及宁波府象山县的珠山茶。至于余姚虽然也种植茶树，但其出产的茶叶并不像其种植棉花所产的“浙花”那么出名。', '不过似乎余姚本地的百姓很喜欢这种清淡的姚江茶，其在本县的销路并不差。', '靠着辛勤的工作，谢方也算撑起了一个家。在谢方的一再坚持下，谢慎从小便被送去开蒙，如今已经学了六年了。不过原先的谢慎最多只能算是中上之资，跟神童绝对沾不上边，照着原有轨迹发展下去，能不能取得秀才功名都是一个问题。']

* 1
* 2
* 3

最后，还是把每句话里面包含的空格，都转换成『，』，就像『章节目录 第一章』，换成『章节目录，第一章』，感觉这一步可有可无了。

# 将上面的正则换成负责找行中的空格

pattern = re.compile(r' +')

# 将所有指定内容替换成空

lines\_of\_text = [pattern.sub("，", lines) for lines in lines\_of\_text]

print(lines\_of\_text[:20])

Out: ['章节目录，第一章，积善之家', '弘治五年，春和景明。', '浙江承宣布政使司，绍兴府，余姚县。', '县城里靠近城隍庙的一处小巷口，一个年约十二，身着淡蓝色粗布长衫，头戴黑色幞头的少年望着不远处熙熙攘攘的人群不发一言。', '他叫谢慎，是土生土长的余姚人。但是也不尽然，因为他的灵魂来自后世，是穿越而来夺舍附着在这个与他同名同姓的少年身上。事情的经过他并不是很清楚，只知道这个少年应该是不慎落水，被人救上后就一直昏迷不醒，奄奄一息，直到自己穿越才鸠占鹊巢，成了这具身体的主人。', '不过有些奇特的是，谢慎还兼有原先身体里的一部分记忆，他拼命检索这才对身处的环境有了一个大概的认识。', '如今是弘治五年，当今天子朱祐樘立志中兴，招贤纳士，敕令各省提学官巡视省内各府、州、县学，选拔参加乡试的人选。今年是乡试之年，明年则是会试，殿试连着两场大试，是出进士的年份。朝为田舍郎，暮登天子堂，在明朝读书自然是最有前途的事情。不过这似乎和他没有什么关系，虽然原先的谢慎也算是个读书人，可却并没有功名在身，最多只能算一个半吊子童生。', '谢慎前世可是苦修明史的研究生，虽然主攻方向是嘉靖万历两朝，但对弘治朝多少也有些了解。浙江一直是科举强省，在弘治朝也是绝对的霸主，而绍兴府则是浙江省中出进士最多的府。mianhuatang。cc，若要落在县一级，那魁首非余姚莫属。山阴，会稽两县加在一起，所中的进士数目都不比余姚的多。这么说来，余姚绝对是科举的死亡之组了。王阳明，谢迁，王华，一想到余姚出的那些名人，谢慎便免不了自嘲。', '谢慎虽然前世苦修明史，对八股文也有所研究，但要让他在明代科场竞争最恐怖的余姚县脱颖而出，确实有些艰难。', '可当他垂下头看了看自己瘦弱的身体，不免摇了摇头，长叹了一声。', '不科举还能干什么呢，就他这副竹竿子般的身体，肩不能抗，手不能提，杀鸡没准还被鸡调戏。', '何况在这个万般皆下品惟有读书高的年代，不科举就只能为民，不管是务农还是经商终归都是被官府压着，没有出头的机会。大明朝那个著名的沈万三不就是最好的例子，家财万贯富可敌国还不是随意一个罪名就成了别人茶余饭后唏嘘慨叹的谈资？', '读书，还是得读书，便前方是刀山火海硬着头皮也得上。他才十二岁，还有可塑性。', '“小郎，你怎么在这儿呢，快快随我回家去。你落水后被救起身子本就虚弱，若是在此时再染上了风寒，那可了不得。你嫂嫂给你特地煮了鸡汤，你多喝上几碗也好补补身子。”', '一个中年男子的声音打断了谢慎的沉思，他抬了抬头见来人是自己的胞兄谢方，心中不由得泛起一抹暖意。', '谢家门丁不旺，到了他们这一代只有他和谢方二人，不过这个便宜大哥似乎对他还不错，吃穿用度上绝不短着缺着。', '谢方在余姚县城里开着一家茶铺，将每年收上的茶叶运到县城售卖，质地更好的一些会有人收走在府城售卖。', '明代浙江绍兴府各县几乎都种植茶叶。最著名的要数杭州府钱塘县的龙井茶以及宁波府象山县的珠山茶。至于余姚虽然也种植茶树，但其出产的茶叶并不像其种植棉花所产的“浙花”那么出名。', '不过似乎余姚本地的百姓很喜欢这种清淡的姚江茶，其在本县的销路并不差。', '靠着辛勤的工作，谢方也算撑起了一个家。在谢方的一再坚持下，谢慎从小便被送去开蒙，如今已经学了六年了。不过原先的谢慎最多只能算是中上之资，跟神童绝对沾不上边，照着原有轨迹发展下去，能不能取得秀才功名都是一个问题。']

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

貌似还忘了一个要处理的，我们看看最后20行的情况。(如果你是用全文本来训练，最后很多行文本中会包括\r这样的特殊符号，要去掉。这里只用了100000字，所以看不到有\r的情况。)。如果有\\r的情况，用下面的方式去掉。

# 将上面的正则换成负责找句尾『\\r』的内容

pattern = re.compile(r'\\r')

# 将所有指定内容替换成空

lines\_of\_text = [pattern.sub("", lines) for lines in lines\_of\_text]

* 1
* 2
* 3
* 4
* 5

到这里数据就处理完了。再看看有多少行数据

print(len(lines\_of\_text))

Out: 2158

* 1
* 2
* 3

因为模型只认识数字，不认识中文，所以将文字对应到数字，分别创建文字对应数字和数字对应文字的两个字典

def create\_lookup\_tables(input\_data):

vocab = set(input\_data)

# 文字到数字的映射

vocab\_to\_int = {word: idx for idx, word in enumerate(vocab)}

# 数字到文字的映射

int\_to\_vocab = dict(enumerate(vocab))

return vocab\_to\_int, int\_to\_vocab

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

创建一个符号查询表，把逗号，句号等符号与一个标志一一对应，用于将『我。』和『我』这样的类似情况区分开来，排除标点符号的影响。

def token\_lookup():

symbols = set(['。', '，', '“', "”", '；', '！', '？', '（', '）', '——', '\n'])

tokens = ["P", "C", "Q", "T", "S", "E", "M", "I", "O", "D", "R"]

return dict(zip(symbols, tokens))

* 1
* 2
* 3
* 4
* 5
* 6
* 7

预处理一下数据，并保存到磁盘，一遍下次直接读取。

helper.preprocess\_and\_save\_data(''.join(lines\_of\_text), token\_lookup, create\_lookup\_tables)

* 1

读取我们需要的数据。

int\_text, vocab\_to\_int, int\_to\_vocab, token\_dict = helper.load\_preprocess()

* 1

检查改一下当前Tensorflow的版本以及是否有GPU可以使用

import problem\_unittests as tests

from distutils.version import LooseVersion

import warnings

import tensorflow as tf

import numpy as np

# Check TensorFlow Version

assert LooseVersion(tf.\_\_version\_\_) >= LooseVersion('1.0'), 'Please use TensorFlow version 1.0 or newer'

print('TensorFlow Version: {}'.format(tf.\_\_version\_\_))

# Check for a GPU

if not tf.test.gpu\_device\_name():

warnings.warn('No GPU found. Please use a GPU to train your neural network.')

else:

print('Default GPU Device: {}'.format(tf.test.gpu\_device\_name()))

Out: TensorFlow Version: 1.0.0

Default GPU Device: /gpu:0

如果没有GPU可以使用，看不到第二行输出，而会是一个警告。

* 1

这里的数据量还是很大的，129万左右个字符。建议使用GPU来训练。或者可以修改代码，只用一小部分数据来训练，节省时间。

正式进入创建RNN的阶段了。

我们的RNN不是原始RNN了，中间使用到LSTM和word2vec的功能。下面将基于Tensorflow，创建一个带2层LSTM层的RNN网络来进行训练。

首先设置一下超参。

# 训练循环次数

num\_epochs = 200

# batch大小

batch\_size = 256

# lstm层中包含的unit个数

rnn\_size = 512

# embedding layer的大小

embed\_dim = 512

# 训练步长

seq\_length = 30

# 学习率

learning\_rate = 0.003

# 每多少步打印一次训练信息

show\_every\_n\_batches = 30

# 保存session状态的位置

save\_dir = './save'



创建输入，目标以及学习率的placeholder

def get\_inputs():

# inputs和targets的类型都是整数的

inputs = tf.placeholder(tf.int32, [None, None], name='inputs')

targets = tf.placeholder(tf.int32, [None, None], name='targets')

learning\_rate = tf.placeholder(tf.float32, name='learning\_rate')

return inputs, targets, learning\_rate



创建rnn cell，使用lstm cell，并创建相应层数的lstm层，应用dropout，以及初始化lstm层状态。

def get\_init\_cell(batch\_size, rnn\_size):

# lstm层数

num\_layers = 2

# dropout时的保留概率

keep\_prob = 0.8

# 创建包含rnn\_size个神经元的lstm cell

cell = tf.contrib.rnn.BasicLSTMCell(rnn\_size)

# 使用dropout机制防止overfitting等

drop = tf.contrib.rnn.DropoutWrapper(cell, output\_keep\_prob=keep\_prob)

# 创建2层lstm层

cell = tf.contrib.rnn.MultiRNNCell([drop for \_ in range(num\_layers)])

# 初始化状态为0.0

init\_state = cell.zero\_state(batch\_size, tf.float32)

# 使用tf.identify给init\_state取个名字，后面生成文字的时候，要使用这个名字来找到缓存的state

init\_state = tf.identity(init\_state, name='init\_state')

return cell, init\_state



创建embedding layer，提升效率

def get\_embed(input\_data, vocab\_size, embed\_dim):

# 先根据文字数量和embedding layer的size创建tensorflow variable

embedding = tf.Variable(tf.random\_uniform((vocab\_size, embed\_dim)), dtype=tf.float32)

# 让tensorflow帮我们创建lookup table

return tf.nn.embedding\_lookup(embedding, input\_data)



创建rnn节点，使用dynamic\_rnn方法计算出output和final\_state

def build\_rnn(cell, inputs):

'''

cell就是上面get\_init\_cell创建的cell

'''

outputs, final\_state = tf.nn.dynamic\_rnn(cell, inputs, dtype=tf.float32)

# 同样给final\_state一个名字，后面要重新获取缓存

final\_state = tf.identity(final\_state, name="final\_state")

return outputs, final\_state



用上面定义的方法创建rnn网络，并接入最后一层fully\_connected layer计算rnn的logits

def build\_nn(cell, rnn\_size, input\_data, vocab\_size, embed\_dim):

# 创建embedding layer

embed = get\_embed(input\_data, vocab\_size, rnn\_size)

# 计算outputs 和 final\_state

outputs, final\_state = build\_rnn(cell, embed)

# remember to initialize weights and biases, or the loss will stuck at a very high point

logits = tf.contrib.layers.fully\_connected(outputs, vocab\_size, activation\_fn=None,

weights\_initializer = tf.truncated\_normal\_initializer(stddev=0.1),

biases\_initializer=tf.zeros\_initializer())

return logits, final\_state



那么大的数据量不可能一次性都塞到模型里训练，所以用get\_batches方法一次使用一部分数据来训练

def get\_batches(int\_text, batch\_size, seq\_length):

# 计算有多少个batch可以创建

n\_batches = (len(int\_text) // (batch\_size \* seq\_length))

# 计算每一步的原始数据，和位移一位之后的数据

batch\_origin = np.array(int\_text[: n\_batches \* batch\_size \* seq\_length])

batch\_shifted = np.array(int\_text[1: n\_batches \* batch\_size \* seq\_length + 1])

# 将位移之后的数据的最后一位，设置成原始数据的第一位，相当于在做循环

batch\_shifted[-1] = batch\_origin[0]

batch\_origin\_reshape = np.split(batch\_origin.reshape(batch\_size, -1), n\_batches, 1)

batch\_shifted\_reshape = np.split(batch\_shifted.reshape(batch\_size, -1), n\_batches, 1)

batches = np.array(list(zip(batch\_origin\_reshape, batch\_shifted\_reshape)))

return batches



创建整个RNN网络模型

# 导入seq2seq，下面会用他计算loss

from tensorflow.contrib import seq2seq

train\_graph = tf.Graph()

with train\_graph.as\_default():

# 文字总量

vocab\_size = len(int\_to\_vocab)

# 获取模型的输入，目标以及学习率节点，这些都是tf的placeholder

input\_text, targets, lr = get\_inputs()

# 输入数据的shape

input\_data\_shape = tf.shape(input\_text)

# 创建rnn的cell和初始状态节点，rnn的cell已经包含了lstm，dropout

# 这里的rnn\_size表示每个lstm cell中包含了多少的神经元

cell, initial\_state = get\_init\_cell(input\_data\_shape[0], rnn\_size)

# 创建计算loss和finalstate的节点

logits, final\_state = build\_nn(cell, rnn\_size, input\_text, vocab\_size, embed\_dim)

# 使用softmax计算最后的预测概率

probs = tf.nn.softmax(logits, name='probs')

# 计算loss

cost = seq2seq.sequence\_loss(

logits,

targets,

tf.ones([input\_data\_shape[0], input\_data\_shape[1]]))

# 使用Adam提督下降

optimizer = tf.train.AdamOptimizer(lr)

# 裁剪一下Gradient输出，最后的gradient都在[-1, 1]的范围内

gradients = optimizer.compute\_gradients(cost)

capped\_gradients = [(tf.clip\_by\_value(grad, -1., 1.), var) for grad, var in gradients if grad is not None]

train\_op = optimizer.apply\_gradients(capped\_gradients)



开始训练模型

# 获得训练用的所有batch

batches = get\_batches(int\_text, batch\_size, seq\_length)

# 打开session开始训练，将上面创建的graph对象传递给session

with tf.Session(graph=train\_graph) as sess:

sess.run(tf.global\_variables\_initializer())

for epoch\_i in range(num\_epochs):

state = sess.run(initial\_state, {input\_text: batches[0][0]})

for batch\_i, (x, y) in enumerate(batches):

feed = {

input\_text: x,

targets: y,

initial\_state: state,

lr: learning\_rate}

train\_loss, state, \_ = sess.run([cost, final\_state, train\_op], feed)

# 打印训练信息

if (epoch\_i \* len(batches) + batch\_i) % show\_every\_n\_batches == 0:

print('Epoch {:>3} Batch {:>4}/{} train\_loss = {:.3f}'.format(

epoch\_i,

batch\_i,

len(batches),

train\_loss))

# 保存模型

saver = tf.train.Saver()

saver.save(sess, save\_dir)

print('Model Trained and Saved')



将使用到的变量保存起来，以便下次直接读取。

helper.save\_params((seq\_length, save\_dir))

* 1

下次使用训练好的模型，从这里开始就好

import tensorflow as tf

import numpy as np

import helper

import problem\_unittests as tests

\_, vocab\_to\_int, int\_to\_vocab, token\_dict = helper.load\_preprocess()

seq\_length, load\_dir = helper.load\_params()



要使用保存的模型，我们要讲保存下来的变量（tensor）通过指定的name获取到

def get\_tensors(loaded\_graph):

inputs = loaded\_graph.get\_tensor\_by\_name("inputs:0")

initial\_state = loaded\_graph.get\_tensor\_by\_name("init\_state:0")

final\_state = loaded\_graph.get\_tensor\_by\_name("final\_state:0")

probs = loaded\_graph.get\_tensor\_by\_name("probs:0")

return inputs, initial\_state, final\_state, probs



def pick\_word(probabilities, int\_to\_vocab):

chances = []

for idx, prob in enumerate(probabilities):

if prob >= 0.05:

chances.append(int\_to\_vocab[idx])

rand = np.random.randint(0, len(chances))

return str(chances[rand])



使用训练好的模型来生成自己的小说

# 生成文本的长度

gen\_length = 500

# 文章开头的字，指定一个即可，这个字必须是在训练词汇列表中的

prime\_word = '章'

loaded\_graph = tf.Graph()

with tf.Session(graph=loaded\_graph) as sess:

# 加载保存过的session

loader = tf.train.import\_meta\_graph(load\_dir + '.meta')

loader.restore(sess, load\_dir)

# 通过名称获取缓存的tensor

input\_text, initial\_state, final\_state, probs = get\_tensors(loaded\_graph)

# 准备开始生成文本

gen\_sentences = [prime\_word]

prev\_state = sess.run(initial\_state, {input\_text: np.array([[1]])})

# 开始生成文本

for n in range(gen\_length):

dyn\_input = [[vocab\_to\_int[word] for word in gen\_sentences[-seq\_length:]]]

dyn\_seq\_length = len(dyn\_input[0])

probabilities, prev\_state = sess.run(

[probs, final\_state],

{input\_text: dyn\_input, initial\_state: prev\_state})

pred\_word = pick\_word(probabilities[dyn\_seq\_length - 1], int\_to\_vocab)

gen\_sentences.append(pred\_word)

# 将标点符号还原

novel = ''.join(gen\_sentences)

for key, token in token\_dict.items():

ending = ' ' if key in ['\n', '（', '“'] else ''

novel = novel.replace(token.lower(), key)

novel = novel.replace('\n ', '\n')

novel = novel.replace('（ ', '（')

print(novel)



这是我用100000个字训练200个循环之后产生的文本

章，正后一屁股文。在这一帮寒门谢氏，但扬声修呼。拉他的手艺推着而出了院门，不是四门中，谢慎注意一息，刚这日一旁不何从世家。王守文在竹门的王守仁作写来已经站候，几十板子，想要去布包吗，不得我到了县尊让谢慎心里一诗会就想在一个木篮子，他也算孔教谕都拱问而即便只有在这些学生，他需要受呼谢方四书一次。这样的人早，只得一个人约的回景，绝不会谢慎便在县学里加靠考下。如果不睡在，谢方和刘老夫子捋了下楼的时间有些啊。“这个王守仁的人情出了。”谢慎一边又要通过身后。曹主簿才不是一个不会再为何事情。他这个时文读洞人要有何证不像提一跳的，大哥谢慎还是想听有人担心的。谢慎也不能掉以轻早一只有意授吗？”谢丕笑了点头，目送着二人，柔声道：“本然比何好关，这才会出他什么可是了心门，你一直不必再拘束。他若不想一个自己拿到官府，他们却是三十大板，会有考好时文，他的境遇更发生了拉近。他自己不能在县试取什么，留是功名，他需要你还有一段时时也能做上。他这个时候出城真也得不太一合写，毕竟余姚名学说的青石头最多一贯情，但布置的学生自称是为自己的。但果然谢慎是想不到四门谢氏的嫡公子那里不远，一种五家小王守文的部二。那样俊望的计”这下

* 1

可以看到模型已经能产生一些有意义的词语了，比如『寒门』，『手艺』，『已经』，『布置』等。但是还没有形成通顺的句子。

这个RNN的架构遵照了

Input -> LSTM -> Dropout -> LSTM -> Dropout -> Fully Connected

这样的形式。

在使用完10万个字进行训练之后，我尝试使用整个数据集也就是129万个字符去训练，但是受困于计算能力有限，没有完成（我在亚马逊云的EC2内存和GPU有限，无法完成这么大数据量的训练，说白了是没钱哈哈~）。大家有资源的可以尝试。

有不足的地方请大家留言或者ISSUE，我也还在不断学习成长中，欢迎交流。

EDIT 6月12日:

前面一次训练只使用了100000的词汇来训练，之后我申请到了亚马逊云一个p2.16xlarge的实例来训练全部数据，使用如下的超参。

# 训练循环次数

num\_epochs = 200

# batch大小

batch\_size = 256

# lstm层中包含的unit个数

rnn\_size = 1024

# embedding layer的大小

embed\_dim = 1500

# 训练步长

seq\_length = 60

# 学习率

learning\_rate = 0.001



经过200个循环，loss大概降到了1.3。参数还需要进一步调整，小于0.5的loss才接近理想。不过1.3的loss可以生成一些通顺的语句了，比之前只有词语的情况进步了不少。

章，如果百年世家不想出现的这些士子，一个孩子多那么夸，竟是得好不起的。谢慎不由得感慨道：“怎么，有几日潞安百姓谢某不过府面子的外转，实话足够啊。？”我又说说没有顾大才吧；我这次便是舍头大明皇帝去打啊， 我现在还没出息，你们要不去。陈虎儿见谢慎面色古怪了，可也不知道该怎么做点出真手错了。就是说，这个谭芳还不得不起自己，也没有多想着谢慎这般拘束他们的表演。不曾想这世上也太不避嫌了。谢方和大人平日闲来得到唐寅身来到西湖身上闯乘，王守仁自然也就没有考虑。回来再四不徐徐芊芊，喝水直是君业，想想他也是可以将直接的成本。不过有些事谢慎不能再找个面子，看看明军心中的架势。这些百姓不是个外人。真想知行来是不会有人拿出来好的，若不是她不选识及防之力就可以让水芸姑娘相互邀请一家。那些倒也没有在他扬名上啊，而这是读书人对太子的才子加足了的。他不需要太多的道理，只不过他还想不到京师生员上面临时起云，怎么可能拒绝低调，真得一起吃酒啊。慎贤弟，怎么，本

像『我现在还没出息』，『王守仁自然也就没有考虑』，『他不需要太多的道理』，都是通顺且有意义的短句了。不过短句前后，还是没有形成语义的连贯。1.3的loss还是太高。有时间再继续调整吧~