中文歌词生成器的实现与比较

王鹏 15307130185

2018 年 1 月 21 日

目录

1 引言

用机器来写歌词是一件看起来比较有趣的事情。

在本学期的“自然语言处理”课程项目中，我选择的题目是“歌词生成器”，即通过 NLP 中的一些随机文本生成的技术，来写出接近于人类语言的歌词。

在课堂上，我们已经学习了几种生成随机文本的方法，例如基于 bigram+ 条件频率的随机方法。

另外，在神经网络火热的当下，RNN 在这方面也展现了强大的能力。

在接下来的报告中，我们将会尝试多种方法生成歌词，并对这些方法的效果进行实验和比较。

2 语料的获得与清洗

并没有在网上找到能用的歌词语料库，因此只好自己爬。

我写了一个 Python 脚本用来爬取网易云音乐上的华语歌词。

但是，获得的裸歌词数据中有很多杂乱的东西 (甚至乱码)，以周杰伦的“一路向北”中部分歌词为例：

“\n[00:35.090] 后视镜里的世界越来越远的道别\n[00:46.519] 你转身向背侧脸还是很美\n[00:53.359] 我用眼光去追竟听见你的泪\n[01:03.790] 在车窗外面徘徊是我错失的机会\n[01:15.519] 你站的方位跟我中间隔着泪\n[01:21.308] 街景一直在后退你的崩溃在窗外零碎\n[01:29.749] 我一路向北离开有你的季节\n[01:37.188] 你说你好累已无法再爱上谁\n[01:43.769] 风在山路吹过往的画面全都是我不对\n[01:51.869] 细数惭愧我伤你几回”

里面有每句歌词出现的时间、多余的空格和换行符，这些对后续的模型训练都是有害的。我们采用正则替换的方法去掉这些数据，具体步骤为：

1. 首先用模式串’[^\u4e00-\u9fa5 \n\r]’ 将非 (汉字、空格和换行符) 给去掉。
2. 然后用模式串’[\n\r]+’ 把重复的空格/换行符替换成一个换行符。
3. 最后用 strip 方法去掉头尾的换行符。

仍以上为例，得到的结果为：

‘‘后视镜里的世界\n 越来越远的道别\n 你转身向背\n 侧脸还是很美\n 我用眼光去追\n 竟听见你的泪\n 在车窗外面徘徊\n 是我错失的机会\n 你站的方位\n 跟我中间隔着泪\n 街景一直在后退\n 你的崩溃在窗外零碎\n 我一路向北\n 离开有你的季节\n 你说你好累\n 已无法再爱上谁\n 风在山路吹\n 过往的画面全都是我不对\n 细数惭愧\n 我伤你几回”

保留换行符的目的一是为了减少分句的麻烦，二是在之后的某些方法中可以作为一个普通字符参与到模型训练中。

最终爬了 37 个歌单，共计 2958 首不重复歌曲，1264580 个中文字符。

3 模型的选取与实现

在接下来的报告中，我们会采用几种模型来达到生成歌词的目的。

3.1 基于 ConditionalFreqDist

3.1.1 基本思想

这种模型在课堂上介绍过，比较简单。就是用语料库字符 list 得到的 bigram 来生成条件频率字典，然后对于上一个字符，以条件频率作为概率来选择下一个字符，具体使用 nltk.bigrams 方法和 nltk.ConditionalFreqDist 方法。

3.1.2 训练结果示例

以’ 我’ 开头：

我
特别被转身后悔得够
把她那只有解
但你出口的想擁你必要
我是我都全部帅不是对天的你转折不由
牵绊倒
打响起
不要回拉拉拉利的煎熬过头
也记住这隻眼泪
满整夜里面笑我想该怎么会一天我的醉者欺负累我掉自

以’ 天’ 开头：

天有一种未来到
只要
而不了其实异想冒险
总是一个岁月亮海能走我怕啦
想都有你
断
还以后想的西顶礼
时光线断句
多

隐藏好的世界已入几多伤心中
我要尽期
周礼安慰
叶
一天黑夜夜色是真的片
只能不知伤

分析：我们可以隐隐约约感觉到文本的“人味”，例如’ 我要’、’ 只要’ 这种词语的雏形。但是句法结构非常混乱。比较令人惊喜的是在这个模型中竟然很少出现循环的情况。

3.2 基于 ConditionalFreqDist + 分词

3.2.1 基本思想

思想与上节相同，只是以词作为训练的基本单位。

3.2.2 分词

我们采用结巴分词来完成任务。
某句子分词示例：
原句：’ 你说人生艳丽我没有异议’
结果：[’ 你’, ’ 说’, ’ 人生’, ’ 艳丽’, ’ 我’, ’ 没有’, ’ 异议’]

最终分词结果为：161912 个句子，共 855852 个词。

3.2.3 训练结果示例

训练结果示例：
以’ 我’ 开头：

我想得自己的怀抱寄望
也姓张怎么去还念旧
五千年来待射的秘密
笑容这样稚气和艺人
真够糗
我的走形式
望你出了
要我也用车载放你要住错
……

以’ 天空’ 开头：

天空晴
唱我们的刺猬

你的气息
内心挫折
对你在城市以北的身边
比我的梦里
我要怎么能够说明了我求求你妄想逆境
……

分析：由于分词的结果，已经不会出现以’ 词’ 为单位的混乱，但是整体句法结构还是很混乱。

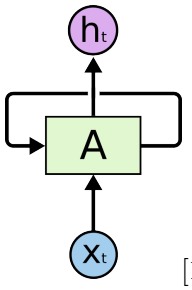
3.3 基于 char-RNN

3.3.1 基本思想

char-RNN[4] 来源于 Andrej Karpathy 的一篇博文 [5]，基于字符进行训练，拥有很强的学习能力。

关于 RNN：

RNN(循环神经网络) 是对一类特殊神经网络的总称。这类神经网络的特点可以用一张简单的图概括：



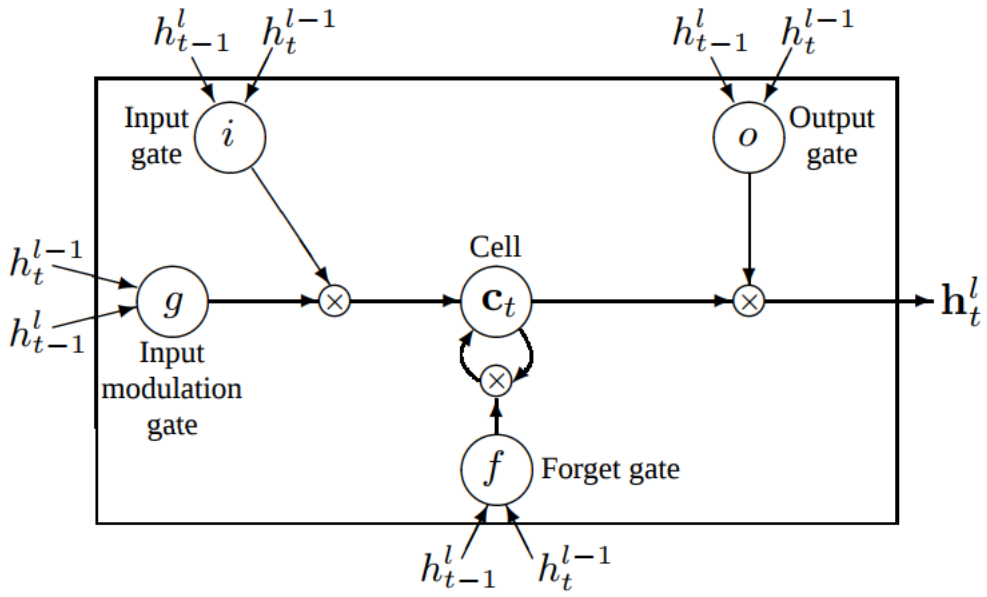
[1]

如图，循环神经网络接收输入向量 X，对于每个 step，将 X_t 和自身状态向量 A 作为输入，得到相应的输出并更新状态。

关于 LSTM：

LSTM(长短记忆网络) 是 RNN 的一种。且 LSTM 有很多变体。在后面的训练中我们使用 TensorFlow 提供的 BasicLSTMCell[3]，其实现基于 [7]。

该 LSTM 每一层的基本结构图示为：



[7]

其中 h_t^l 表示第 t 个 timestep 第 l 层 layer 所输出的状态。特别地，h_t^0 表示输入序列的第 k 个元素 (在 char-RNN 中表现为字符向量)，h_t^L 表示该 timestep 的输出结果，C_t^l 表示对应的 LSTM 单元的细胞状态。

关于字符的表示：

类似于 Word2Vec 的方法，通过建立 embedding，将每个字符映射到长度为 rnn_size 的向量，初始向量值随机。但是在实现中我们并不单独训练 embedding，而是把它和 RNN 参数混在一起训练。

3.3.2 训练

我们采用 sherjilozair 大神的 char-RNN-tensorflow[6] 中的 LSTM 来作为模型的实现。

输入数据采用简单分割方式分成多个 batch。在 char-RNN-tensorflow[6] 的默认参数中，batch_size 设置为 50，单个数据作为一个 sequence，默认 sequence_size 为 50，RNN 层数设置为 2 层。前面清洗数据后语料库共有 1264580 个字符，分割后数据共有 $1264580/(50 \times 50) = 502$ 个 batch。

平均每个 sequence 训练时间约 0.6s，训练整个数据集约 4 个小时。

3.3.3 训练结果示例

以’ 我爱’ 开头：

我爱有你我心河
你的爱你很懂啊你也不会结局
如果我又试着太多过为它的勇敢带走我的笑无暇想
请你说一个人走到回忆
在你有时点多了解我一个家
我怎么一百遍
睁开了手中那么温暖我的
最后才能让流浪也曾不再陷入

内疚我
请为你在我缤涌
爱最美其印揣花的朋友
那一个季节
就算我你的微笑
如果我求你
每次你别说要你离开我
我却给我柔软的悲喜
我别再精彩的爱你今夜我会
想你等下奋斗

以’ 天空’ 开头：

天空
初心
瞥着爱多年的人的脑海
我笑爱未到他
被潮流人无味
如果你恨着你
锺爱 (注：生成结果如此，前期数据忘记简繁转换) 你一个人
忘了此刻才忘记
再回运在身边
你说过你总是答应我
宁愿只该靠近你的祝天
你让我岁底会存在
这样一不小心情
别听你分享你的你
前一整个寒冷最暖酒我拨开我心扉
装坐里好像就小孩
我的习惯每个人
终于开心后悔再失望要任何男头
想一直都会有无暮浪美的手
是我自已爱你都不会伤悲
如果演唱幸福的就是我一直在回忆是你
像一个大学与你一起

分析：我们可以发现该模型生成的结果已有隐约的句法结构，例如’ 你说过你总是答应我’、’ 最后才能让流浪也曾不再陷入’，但还是有很多奇怪的’ 自造词’，例如：’ 缤涌’，’ 男头’。

3.4 基于 char-RNN + 分词

3.4.1 基本思想

在上面的 char-RNN 中，我们以字符为单位进行训练。但是我们可以用同样的原理，将 char-RNN 稍加修改，以词为单位训练。

3.4.2 分词

仍然像 3.2 一样，采用结巴分词。

3.4.3 训练

与之前一样，采用简单分割的方法，其中 batch_size=50，sequence_size=50，batch_num=342。

由于基本单位变成了词语，词典规模变大，用同样的模型训练起来耗时更长。经测试本机单个 sequence 训练需要 8s 左右，训练时间超过一天。

3.4.4 训练结果示例

以’ 我爱’ 开头：

我爱的我生命中不好不坏
我不想不在
你的剧情有时很愉快
我愿意如何不看我
还不够自然
待这对你们不退
已想爱惜你
两个人有我阐述
醒来开的只能送你
别跟我喝完
我一直不休
你
希望你长大一点
魔
与谁都在哪里
爱上我豁达的酒

以’ 天空’ 开头：

天空
多想快乐心等不到我
紧紧送你的手啊
如果我终于如何
爱你思考

在我之间
真的很无聊
我们将它禁锢
一起睁不开追求
不要告诉我我看海的体验
的幸福传来一班多默契
她坚定的下雨吻潮起潮落
拼凑虾米
放学自永远伪装爆发

分析：可以看见生成的文字既有隐约的句法结构，例如' 你的剧情有时很愉快'、' 不要告诉我我看海的体验'，又因为分词的结果，几乎没有出现奇怪的' 自造词'，(~~感觉都可以拿来给妹子写情书子~~)。

4 实验与比较

4.1 实验方法

如何找到一种标准来比较这几种模型的优劣？我目前能想到的方法是基于词向量模型来进行评测，我们注意到两个距离较近的词向量在语义上更有相似性，因此可以考虑计算词向量之间的距离来判断模型的优劣。

具体步骤为：

step 1: 根据已有的歌词语料库，训练一个 Word2Vec 模型。

step 2: 爬取一些歌词，作为测试数据。

step 3: 对于每段歌词，选取前 X 个词作为输入数据，并要求模型生成至少 $4Y$ 个字符。(X 和 Y 均为自选参数)

step 4: 将输出的 4Y 个字符进行分词，得到至少 Y 个词语。

step 5: 将输出的 Y 个词语 (其对应词向量为 A) 与测试数据的后 Y 个词语 (其对应词向量为 B) 进行比较，计算代价函数，目前设计为：

$$cost(A,B) = \frac{\sum_{i=1}^{Y} |A_i - B_i|}{Y}$$

4.2 实验细节

4.2.1 Word2Vec 训练：

仍然使用前面爬取的语料库作为训练数据，使用 gensim.models.word2vec.Word2Vec[2] 得到词向量模型。

训练结果示例：

model.similar_by_word(' 你'):
[(' 她', 0.9458020925521851),
(' 他', 0.9260046482086182),
(' 你们', 0.888885498046875),

(’ 他们’, 0.8744577169418335),
(’ 别人’, 0.8650164604187012),
(’ 作对’, 0.8533217906951904),
(’ 我’, 0.8520398139953613),
(’ 微笑’, 0.8449605703353882),
(’ 忠贞’, 0.8318023681640625),
(’ 兄弟’, 0.8296780586242676)]

基本符合理想中的状况，说明该 Word2Vec 模型是有效的。

4.2.2 短句输入测试：

从语料库中随机抽取 1000 份分词后长度 >10 的 (歌词中的) 句子，将其抹去后四个词语作为输入，将模型输出的字符串分词后的前两个词语作测试。

例如：

我/的/花/却/在/这/山谷 -> 等待/匆 (ConditionalFreqDist)
我/的/花/却/在/这/山谷 -> 等/你 (ConditionalFreqDist + 分词)
我/的/花/却/在/这/山谷 -> 何然/快点 (char-RNN)
我/的/花/却/在/这/山谷 -> 脑海/看着 (char-RNN + 分词)

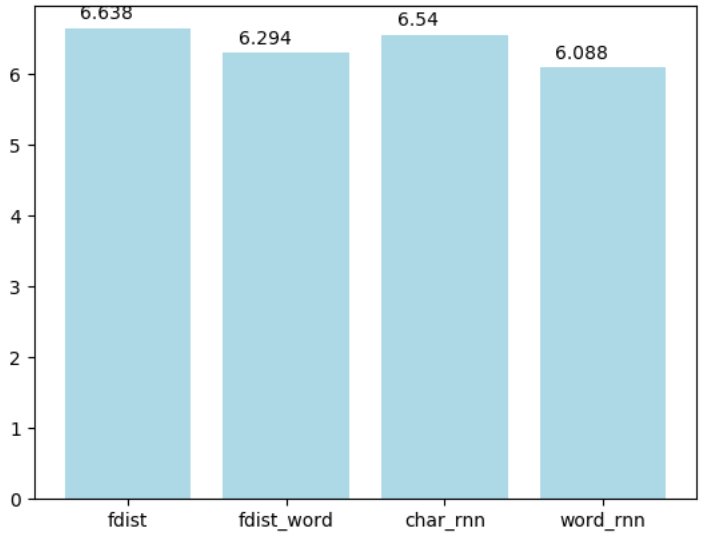
4.2.3 长句输入测试：

从语料库中随机抽取 500 首歌，以歌词的前 50 个词语作为输入，将模型输出的字符串分词后的前两个词语作测试。

4.3 实验结果

4.3.1 短句测试结果：

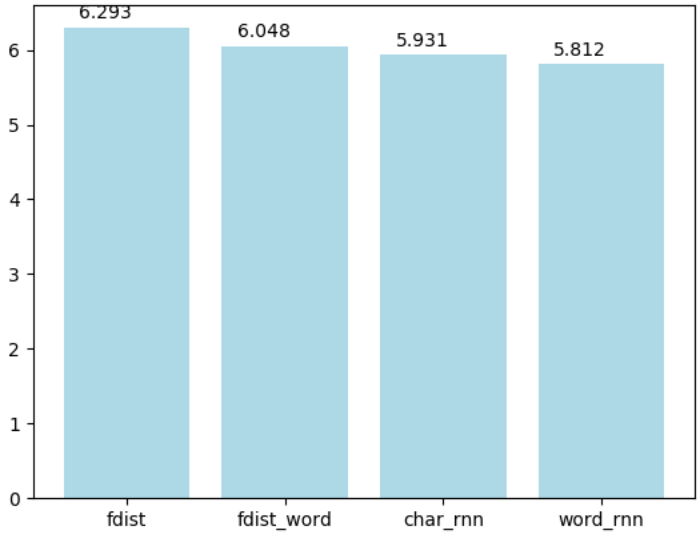
如图所示：



数值越小越好。可以发现在短句输入情况下，word_rnn 优于 dist_word 优于 char_rnn 优于 dist_char。

4.3.2 长句测试结果：

如图所示：



可以发现在长句输入情况下，word_rnn 优于 char_rnn 优于 dist_word 优于 dist_char。

这里比较奇怪的一点数据显示在长句情况下 FreqDist 的效果竟然比短句情况下要优！但是这在原理上应该是不可能的。猜想应是测试数据量太小导致出现了随机性的偏差。

4.3.3 分析

从前面的实验我们可以发现，不论是在短句还是长句输入的情况下 RNN 的方法比简单词频分析的方法要好，在长句输入的情况下这种优势更明显。另外可以发现基于分词的训练比基于字符的训练效果看起来要更好。

不过这一切都建立在 Word2Vec 的正确性之上。我个人不能保证这个 Word2Vec 模型作为评判标准的合理性，但是从直观感觉上来说这么测试还是有其一定道理的。

其实我本来还想做长句输出的测试，但是感觉不太靠谱，比如：输出’ 他是我爸爸’ 和输出’ 他是我的爸爸’ 这两种，在语义上完全一致，但是分完词后由于后者多了个’ 的’ 字，并不能做到词与词之间的一一对应比较。后来的想法是输出 Y 个词语，再从里面选择 Y-Z 个词语与标准输出的 Y-Z 个词语做最优匹配以最优值作为测试结果。不过这样太麻烦，受限于自身精力与时间，就不了了之了。

5 总结

我们通过了好几种方法来实现歌词生成器，不同的方法各有优点和缺点，例如基于 RNN 的方法优点是输出效果好，缺点也很明显，比如训练很吃机器，太耗时间，生成速度也比较慢。

总地来说我认为这个题目还是比较有趣的，最终实现生成的文本也比较有人的味道。我希望以后能有机会用上这些歌词生成器，来干一些奇怪的事情 (~~比如写情书啥的~~)。

参考文献

[1] colah. Understanding-lstm-networks.

[2] gensim. gensim: models.word2vec - deep learning with word2vec.

[3] Google. tf.contrib.rnn.basiclstmcell tensorflow.

[4] Karpathy. Github-char-rnn.

[5] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, 2015.

[6] sherjilozair. Github-char-rnn-tensorflow.

[7] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.