

关于绕口令自动生成的研究

中文信息处理课程项目
实验报告

张睿哲
14307130222

January 7, 2017

Contents

1	实验概述	3
2	实验原理	4
2.1	马尔科夫链模型	4
2.2	语音学模型	5
2.3	循环神经网络模型	7
2.4	三种模型的比较	9
2.5	绕口令的语言特点	9
3	实验过程和结果	11
3.1	数据集的准备	11
3.2	实验过程详述	11
3.3	马尔科夫链模型	11
3.4	语音学模型方法	13
3.5	循环神经网络方法	17
4	实验结论	20
5	附录	20

1 实验概述

在我们的日常生活中，绕口令（Tongue Twister）是一种非常有趣的语言游戏，它将一些音调容易混淆的词语组合在一起，让人读起来有些拗口，因此绕口令又称拗口令。本实验的主要内容就是利用自然语言处理的知识实现自动生成中文和英文的绕口令。本人在实验过程中实现了不同的文本生成模型（马尔科夫链模型、语音学模型和单字符-循环神经网络模型），并且为了训练模型构建了中、英文绕口令语料库，并对各个模型的性能进行了测试和比较。

本报告主要分为三个部分，首先对实验原理进行介绍，包括本实验使用的几种文本生成模型的原理以及绕口令的语言特征；之后对实验过程进行叙述，包含了算法实现的一些细节，并对各个模型的测试结果进行了分析和讨论，比较了它们在中、英文绕口令生成上的性能差异；最后得出实验结论。

2 实验原理

本节主要介绍实验中使用的文本生成模型，即马尔科夫链模型、语音学模型、单字符-循环神经网络模型的基本原理，最后结合实验内容对中、英文绕口令的语言特点进行简要介绍。

2.1 马尔科夫链模型

马尔科夫链模型是随机文本生成中最基础的模型，它利用之前已经产生的词对接下来要生成的词进行预测，从而产生连续的随机文本。马尔科夫链具有如下定义：

Definition 2.1 (马尔科夫链, Markov Chain). 定义随机变量序列 $X_1, X_2, X_3 \dots$ 为一个随机过程的状态序列，则该随机过程构成一个马尔科夫链如果满足如下性质：

$$P(X_n | X_{n-1}, X_{n-2}, \dots, X_1) = P(X_n | X_{n-1})$$

从上式中容易看出，马尔科夫链只能记住上一个状态是什么，接下来的转移只和上一个状态有关。

在文本生成问题中，可以将单词或单词的组合看成是一个状态，每次产生新单词时利用保存的前一个状态计算生成哪个单词的概率最大，则产生该单词。因此在使用马尔科夫链模型生成文本之前，首先需要对训练文本建立语言模型，如果只考虑前一个词，则需要对训练文本的 Bigram 进行统计；如果要考虑前面更多的词的组合，则需要建立 n-gram。在建立好语言模型之后，即可开始文本生成。首先从词典中随机挑选一个词，之后开始执行马尔科夫过程，每次根据前面产生的词在语言模型中进行查找，统计下一个词的频率分布，按照一定的规则根据词频分布选词生成即可。流程图如下：

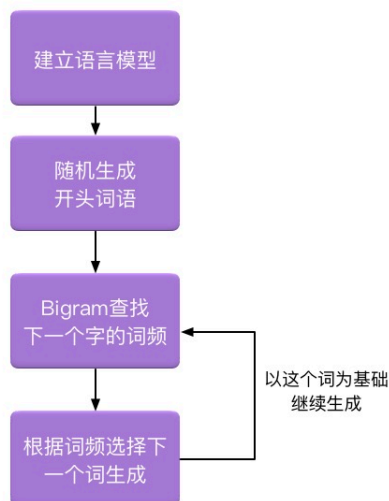


Figure 1: 马尔科夫链模型流程图

2.2 语音学模型

由于绕口令的构成具有明显的语音学特征，因此对于绕口令的生成问题可以考虑使用语音学模型。所谓语音学模型，其基本思路就是根据词的发音对词语进行选择，选择的方法需要人工添加一些规则，比如挑选构成头韵的词，挑选韵母相同的词等。

具体来说¹，语音学模型首先需要一种表示方法来刻画每个词的发音的语音语调。目前，国际上通用的表示方法是国际音标(International Phonetic Alphabet, IPA)，它的优点在于其具有通用型，可以对世界上所有的语言进行注音。当然，这也导致了这个符号系统的注音符号众多，规则复杂，和我们日常生活中使用的音标（比如英语的KK 音标）有一定区别。

在对每个词进行注音之后，可以得到每个词的音标序列，那么很自然的就可以想到我们需要定义一种距离测度来衡量两个词在发音上的区别。首先我们需要定义任意两个音素（也就是音标符号）直接的距离。在查阅相关资料后可以发现语音学家已经为国际音标 IPA

¹ 由于语料库的问题，仅对英文进行讨论。

定义了一些音系学特征，通常用表格的形式来表示每个音标的特征，表格中的“+”表示具有此特征，“-”代表不具有此特征。

		examples										[+cons, -son, -cor]										[-cor + dors]										[+cons, -son, -cor]										[-cons, -son]									
		t'	t	j	d	s	z	ʃ	θ	ð	f	ʒ	ç	j	p	b	f	v	φ	β	g	x	ɣ	q	χ	ħ	ʕ	h	ɦ	ʔ																					
Class features	cons	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-																				
	son	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																				
	syll	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																				
Place features	labial	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-																				
	round	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																				
	coronal	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+																				
	ant	+	+	+	+	+	+	+	+	+	+	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																				
	distrib	-	-	+	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+																				
	dorsal	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+																				
	high	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+	+	+	+	+	+	+	+	+	+	-	-	0	0	0	0																				
	low	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0																				
	back	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0																			
	tense	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0																			
	phragl	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+																				
Lexical features	ATR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	0	0																			
	voice	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+																				
	S.G. C.G.	- +	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -																			
Manner features	cont	-	-	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+																				
	strident	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																				
	lateral	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																				
	del rel nasal	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																				

Figure 2: 音系特征表

根据这个表格我们可以对每个音标生成一串 0/1 特征序列，两个音标之间的距离用它们的特征序列的汉明距离来表示（即两串序列不相同的位置的个数）。这样就可以在音标的基础上定义两个单词之间的距离。但还会遇到的一个问题就是两个单词的音标序列长度不相等，无法直接比较对应位置音标的距离。对于这个问题通常可以由两种解决方法，一种是使用编辑距离，另一种是借鉴语音信号处理的方法，使用动态时间规整距离。

- 所谓编辑距离（Levenshtein Distance），即对于字符串 A 和 B，将 A 串通过以下三种操作变为 B 串的最少步数（代价）：
 - 改变一个字符；
 - 插入一个字符；
 - 删除一个字符；

编辑距离最常用的求解方法是动态规划。我们定义插入一个音标和删除一个音标的代价为音标的特征数 $n_{features}$ ，改变一个音标的代价为两音标的距离（参见上文的定义），则动态

规划的状态转移方程为：

$$f[i, j] = \min\{f[i-1, j-1] + \text{dist}(A_i, B_j), \quad (1)$$

$$f[i, j-1] + n_{\text{features}}, f[i-1, j] + n_{\text{features}}\} \quad (2)$$

$$f[i, 0] = f[0, i] = i * n_{\text{features}}. \quad (3)$$

其中 $\text{dist}(A_i, B_j)$ 表示单词 A 的第 i 个音标和单词 B 的第 j 个音标之间的距离。

- 对于动态时间规整距离 (Dynamic Time Wrapping Distance), 主要用在语音信号处理中, 用来比较两端语音之间的差异度。它主要思想也是动态规划, 其转移方程为:

$$g[i, j] = \min\{f[i, j-1] + \text{dist}(A_i, B_j), \quad (4)$$

$$f[i-1, j] + \text{dist}(A_i, B_j) + f[i-1, j-1] + 2 * \text{dist}(A_i, B_j)\} \quad (5)$$

对于这两种距离的计算方式, 都是针对不同长度的序列所定义的规则, 但它们的特性稍有不同: 动态时间规整更适用于不同时间基准导致序列长度不同的序列之间的比较, 比如对于同一个词语不同的人说话的语速不同, 从而使语音段的长度不同; 而编辑距离则比较适用于静态序列, 单纯比较内容的差异, 比如最早被应用在比较 DNA 序列上。在这个问题中, 由于不同词语的音标序列就像这个词的 DNA 一样, 不会因不同人的发音而改变, 所以我认为使用编辑距离要比使用动态时间规整距离更为合适。

对于英文绕口令的生成问题, 在定义了词和词之间的距离之后就可以根据距离选择最相似的词语排列在一起, 从而组成一段文本。这样生成的文本相邻词之间由于距离近, 从而发音相似, 产生拗口的效果。

2.3 循环神经网络模型

深度神经网络是近几年机器学习领域的热点, 随着 GPU 计算能力的不断增强, 各类神经网络的潜力相继被挖掘出来, 在各个领域取得了相当辉煌的成绩。在自然语言处理这个研究领域中, 被广泛使用的神经网络就是循环神经网络 (Recurrent Neural Networks, RNN)。RNN 的结构和普通的前馈神经网络比较相似, 都是由输入层获取

输入，将输入传到中间隐藏层结点，经过激活函数后产生输出到输出层。唯一的不同于中间的隐藏层结点具有反馈的功能，当前的输出不仅和输入相关，还和上一一次的输出相关，即：

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$$

其中 h_t 为隐藏层结点的当前输出， $W^{(hh)}$ 为上次输出的权重矩阵， $W^{(hx)}$ 为当前输入的权重矩阵。通过加入了这样一种反馈的机制，使 RNN 网络和前馈神经网络相比具有了记忆的能力，当前的判断不仅仅依赖于输入，还会结合记忆中之前作出的判断。这个过程和我们进行阅读理解的过程非常类似，我们要想理解一句话，不是根据一个个的单词进行理解，而是会结合上下文的信息，而这恰恰正是 RNN 所能做到的，这也就解释了为什么 RNN 非常适合用于自然语言处理的任务中。

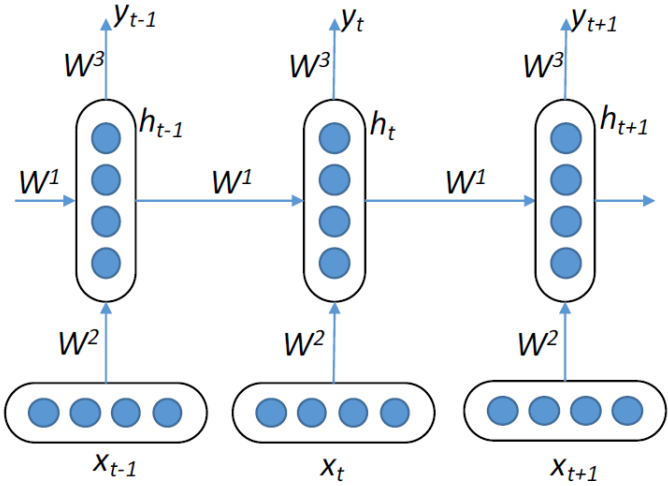


Figure 3: RNN 按时间展开的结构图

在实际应用中，目前通常使用的是 RNN 的改进版，比如长短时记忆（Long Short-Term Memory）的 RNN 和门限循环（Gated Recurrent Unit, GRU）的 RNN。这两种网络在 RNN 的基础之上加入了门限函数控制的记忆单元，从而增强了 RNN 的记忆功能，解决了“长期依赖”的问题。所谓“长期依赖”问题，就是相关联的信息在输入中的位置可能相隔较远，比如对于一个长句主语和宾语由于中间

成分过多而距离很远，导致普通的 RNN 很难训练出合适的参数来对应这种关联。LSTM 通过增加了一些记忆单元，可以将之前的输出保存在其中，并根据当前的状态通过门限函数控制记忆单元的使用与更新，从而帮助 RNN 提取较早的记忆，使其具有更好的训练效果。

对于文本生成的任务，目前一个比较流行的方法是使用单字符循环神经网络（Char-RNN）。它的特点就是对于网络的输入是一个个的字符，而不是以单词为单位，在网络的训练过程中让网络自己学习到单词是以空格符为分隔的这个概念，进而将网络训练到可以提取出词的语法语义特征，从而可以生成比较符合语法语义的文本。而且由于它不需要分词，所以生成文本的类型也更为多样化，从文章到代码都可以进行训练并生成。由于本实验涉及到中文绕口令，对于中文文本的生成，由于它以字符为单位，所以不需要进行中文分词，从而更加方便使用。而且不同的单字的个数要远小于不同的词语个数，从而可以减小词向量（Word Embedding）的维数。当然，这样处理的代价就是训练时间要增长，训练前期需要多轮迭代才能让网络以单词为单位生成文本。

2.4 三种模型的比较

这里只是从理论层面对这三种模型进行一些讨论，对于实际的性能和效果会在实验过程部分结合实验结果进行对比。对于这三种模型，第一种和第三种是文本生成比较常用的方法，第一种更为简单轻便，而第三种神经网络的方法有可能会提取出更深层次的语义特征。这两种方法均需要进行训练，第一种要建立语言模型，第三种要训练一个网络。对于第二种方法，不适用于一般的没有明显语音特征的文本生成，但在绕口令生成的任务中却非常合适，可以充分利用单词的发音特点，生成出在朗读上非常拗口的文本，但它很难利用文段的语义信息，所以产生的文本难以理解。

2.5 绕口令的语言特点

为了更好地完成绕口令自动生成的任务，我们需要对绕口令的语言特点进行分析。由于中文和英文绕口令有所不同，所以分别对其进行讨论。

- 对于英文绕口令，也就是 Tongue Twister，有几种不同的形式：

- 最基本的就是词和词之间交替使用相近但不同的音素（比如 s[s]、sh[ʃ]、th[θ] 等），通过快速的交替转换让舌头不断移动位置。比如：“The seething sea ceaseth and thus the seething sea sufficeth us.”
- 一些 Tongue Twister 会使用头韵 (Alliteration)、押韵 (rhyme) 等手法，通常会使用两种不同的发音序列互相交错，每个序列具有自己的韵律。比如：“Betty Botter bought a bit of butter.”
- 另一种形式的 Tongue Twister 会让读者将一些发音比较相近的词组连续读多遍，比如：“World Wild Web(*3)”。

因此对于英文绕口令更注重词的发音，对于绕口令的表意不是非常在意，当然也有一些具有鲜明主题的 Tongue Twister 风格的诗歌作为例外。

- 对于中文绕口令，有人²总结出了“绕”、“咬”、“急”三个特点：
 - 所谓“绕”，就是将同一个意思的话不断变换说法来讲，比如“顶棚钉银钉，银钉钉顶棚。”
 - 所谓“咬”，就是选择发音相近、易混淆的字，这一点和英文绕口令比较类似，比如“黑化肥发挥会发黑。”
 - 所谓“急”，就是朗读语速要快，为了达到这个目的会使用叠词以增加朗读的速度，比如“方方框，方方房，方方方框放方房，方方方房放方框。”

通过这三点可以看到中英文之间具有一定的差异，一些经典的中文绕口令不仅在发音上精雕细琢，整个文段还在讲一个故事，从而增强了趣味性，这也正是中文绕口令的语言魅力所在。

通过以上对绕口令语言特点的总结可以看到，中、英文绕口令的生成均依赖于字或词的发音。但中文绕口令的生成要比英文更加困难，主要是因为一个好的中文绕口令既要考虑发音也要考虑语言含义，而英文绕口令更多时候被用来练习口语发音，对一句话的意义没有过多要求。

²李春艳, 付永兴. 绕口令语言特征浅析.

3 实验过程和结果

本次实验主要分为两个阶段，分别是数据集的准备和模型实现与测试。上一节中介绍的三个模型是在实验过程中依次实现并改进，所以模型的测试贯穿整个实验过程。因此本节会从这两个方面对实验过程及测试结果进行叙述。

3.1 数据集的准备

对于构建统计语言模型来说，非常关键的一步就是构建语料库。有了语料库之后才能在其上进行训练和测试工作。由于绕口令不像小说和诗词，没有现成的非常齐全的语料库，因此在构建过程中会有一些困难。首先最基本的语料库可以在 *1st International Collection of Tongue Twisters* 的网站上得到，这个网站包含了非常众多的语言的绕口令语料，其中英文绕口令有 593 首，中文绕口令有 27 首，可以使用爬虫抓取数据。但是这些数据对于机器学习的训练和测试来说都太少了，特别是中文绕口令的数量完全达不到要求。因此我又人工在网上查找中英文绕口令，在一些网站的文章上（比如《绕口令大全》等）将绕口令复制到语料库中。最后总共搜集了 871 首英文绕口令，567 句中文绕口令（200 首左右）。其中一些数据是在实验过程中发现数据量不够而再次增补的。

3.2 实验过程详述

3.3 马尔科夫链模型

在实验的开始阶段，我最先想到的文本生成模型就是马尔科夫链模型。因此我对中文和英文语料库进行语言模型建模，统计出了所有 Trigram 的数量，并维护了一个 Trigram 的查询结构，使得可以根据之前生成的前两个字查找可能的第三个字的词频分布，也就是马尔科夫链的记忆长度为 2。但是在模型建好以后进行测试时，发现生成的文本有很大的问题，比如：

case1: 树有木竿做支柱，木支树树稳固。树一上个窝，树下一口锅...

case2: wood saw witch would watch two watches which washed Swiss Swatch

生成的绕口令看上去太过完美，在语料库中查找后发现这些都是原有的文本。我认为主要原因是语料库的容量太小，导致一个词后面的候选字太少，从而无法体现马尔科夫过程的随机性，每次几乎没有选择的余地。因此一个改进的地方就是将马尔科夫链的记忆长度改小为 1，使用 Bigram，这样每次选择词语时仅依靠前面一个字或词，从而增加了选择的余地，增大了模型的随机性。改进后生成的文本如下：

case1: 北风吹摇路边树，小陆上前把树护...

case2: wouldn't whistle when Washington's white wheat reapers reap ripe white wheat

经过几次实验，中文绕口令始终生成原文，而英文绕口令则增加了许多变化，比如上面的例子，经过查找发现这句话不是语料库中的原句，是将几句中的短语拼接而成。因此这也说明了语料库的容量对模型的影响非常大，英文语料库比中文语料库大，从而可以有更多的变化。另一方面，经过这个实验也说明了在训练数据较小的情况下，使用较小的语言模型会有更好的效果，马尔科夫链如果使用较长的记忆，虽然会提升预测的准确率，但会减少模型的随机性，更适用于语料库非常充足的情况。

根据上一节绕口令的语言特征分析，我们知道英文绕口令在很多情况下是将很难连读的词排列在一起，而这些词由于太难读出来，因此在英语常用语料库中也不会经常出现。因此一个逆向思维的思路就是对英语常用语料库建立语言模型，统计一个词后面最不可能跟着的是哪个词，选择这个词有很大可能构成了 Tongue Twister。因此在实验中我选择了 NLTK 的布朗库，对其中的词建立了词频统计。之后的过程和前一个方法类似，依次在马尔科夫链中生成单词，只不过前一个模型是随机选择单词，而这个模型是选择频率最低的单词。生成的文本如下：

case1: casher padlocked minutiae childcare trichet feck
case2: everlastings hadd scrambling meridionale bechthold
 hashagen

可以看到效果并不理想，词和词之间没有非常相近的发音，只是使用了一些非常生僻的词汇。这也和这个方法的本身有关，虽然绕口令中的词平常人们不会说，但由于语料库中的数据量很大，因此还有更多发音不相似的词也不会出现在一起，比如一些词由于词性等原因几乎不可能排列在一起，或者一个词语是非常生僻的词汇等。这样导致产生的文本并不符合 Tongue Twister 的要求。

因此，对于马尔科夫链模型，效果比较好的方法是在绕口令语料库上使用马尔科夫链生成文本，这样会产生和绕口令比较相像的文本，但需要依靠比较大规模的训练集。

3.4 语音学模型方法

语音学模型的方法参考了别人的实现³。在语音学模型中，首先要对音标建立特征矩阵，之后使用编辑距离或者动态时间规整的方法计算两个词之间的距离。接下来在文本生成的过程中，需要根据词语的发音距离挑选词语。但这样可能出现重复的问题，即两个距离非常小的词语会反复出现。为了避免这个问题，我们可以设置一个缓冲区，相当于模型的记忆长度，要求生成的词不能出现在缓冲区中。对于这个模型，我做的改进是加入了语法因素。因为如果仅依靠语音学模型，则绕口令只是把发音相近的词排列在了一次，很有可能无法构成句子，因此在模型中加入了词性模板来限制生成的文本符合语法规则。具体来说，首先使用 NLTK 自带的词性标注功能对词典进行标注，给每个词标上词性。之后在进行文本生成的过程后首先设置一个生成语句的词性模板，可以根据语料库中的绕口令的词性来进行设置。之后再产生每个单词的时候必须要求其词性和模板中的对应位置的词性相同，这样相当如加入了一个满足语法条件的约束。由于词典的容量非常大，所以加入了这个约束以后不会影响选择的空间。

在具体实现的过程中，该算法的原作者使用的 unilex 词典已经无法找到了，因此我需要重新构建一个词典。这一步之所以比较麻烦，首

³https://github.com/perimosocordiae/twisty_tongue

先是因为带有音标的英语词典语料本身就非常稀少，比较合适的就是 CMU 的英语语音词典；其次是因为目前已有的音系学特征都是针对国际音标 IPA 系统，而如前文所述这个系统非常繁杂，因此词典的编写者也不愿意去使用它，特别是它的注音符号是 unicode 格式，输入也比较麻烦。因此 CMU 的英文词典使用是他们自己设计的一套简化的标音系统。所以我需要根据对应规则将 CMU 的词典的音标转换为 IPA 格式。而且 CMU 词典不带词性信息，需要使用 NLTK 对其进行标注。最后经过统计，该词典总计包含 13 万个单词。

- 如果不使用词性模板，结果为：

case1: plotted plodded plagued pleaded plugged applauded
pleased pillared
case2: strongly stringed stirrings stronger strings storms

可以看到几乎不符合语法，但词和词之间的发音比较相近。特别值得注意的一点是它倾向于生成头韵，这是因为我在计算编辑距离的时候给开头两个音节以双倍的权重，从而让它生成的语句更倾向于开头部分发音相似，对应了绕口令语言特征中英文绕口令的第二条。

- 如果使用词性模板 “[’NOUN’, ’VERB’, ’NOUN’]”，结果为：

case1: haze hadd hug
case2: birthday breathed brother

这种模板生成的绕口令符合简短的主谓宾结构，比较适合作为一个短语重复朗读的形式（比如 “haze hadd hug (*3)” 的绕口令。

- 如果使用词性模板 “[’NOUN’, ’NOUN’, ’VERB’, ’DET’, ’NOUN’, ’ADP’, ’ADJ’, ’NOUN’]”，结果为：

case1: rumors remorse rumored another’s nozzles nearest nicest
insights
case2: pop beep bite both apathy about bad bogey

可以生成这种语法结构比较复杂的语句。

对于中文的语音学模型，如果用英文的方法在建立上就有很大的困难。首先我没有找到使用 IPA 注音的中文词典语料库，大部分中文词典都是使用的汉语拼音进行注音，而且还没有标明声调。同时我们知道拼音在只能在一定程度上反应词语的发音，有很多例外情况。比如“自(zì)己(jǐ)”这两个字的韵母相同，但他们是不押韵的。这是因为韵母相同代表他们属于一个音位，但实际发音中使用的音素不同，也就是有不同的 IPA 音标。因此仅仅依靠拼音标注是无法准确判断两个字的读音差异，从而上述距离计算结果就会有一些问题。同时为了使用音系特征，要将汉语拼音转化为 IPA 音标，这个过程也非常复杂。所以使用语音学模型不适用于进行中文的绕口令生成。

但我认为依靠汉语拼音还是可以对绕口令进行处理的，毕竟大多数情况下韵母和声母还是可以反映一个词的读音的。因此我希望利用这些信息对已有的绕口令进行修改，生成新的绕口令。具体来说，对于输入的绕口令，首先我使用了“结巴分词”工具对绕口令进行分词，并利用它的词典对每个词进行词性标注。之后我使用了 SnowNLP 工具箱对这句话中的每个词标注拼音。然后我引入了 NLTK 自带的 `sinica_treebank` 语料库作为中文语料库，但使用它比较麻烦的一点是它用的是繁体中文，我需要用 `zh_wiki` 包将文本进行繁体、简体的转化。这个语料库是分好词的语料库，我将其中的词按照词性进行归类，用字典维护。对于输入的一句话，按照词性查找语料库中所有词性相同的词作为替换词，然后将逐一比较和原词在发音上的差异。为了简单起见我只比较了两个字的韵母是否相同，即它们是否押韵。最后在所有符合要求的词中随机选择一个作为替换词。如果这个词在下文中继续出现，则使用第一次替换的词进行替换。在实验中可以发现，这个方法的效果好坏主要依赖于两个因素：

1. 分词器性能的好坏，如果分词器将一些词误分或漏分，则无法找到其替换词；
2. 词性标注的质量，汉语的词性比较复杂，如果词性标注的质量比较低，则会出现错误的替换情况。

很不幸，使用结巴分词和 `sinica_treebank` 语料库，导致不仅分词质量不高，而且词性标注有问题。比如：

输入：吃葡萄不吐葡萄皮儿，不吃葡萄倒吐葡萄皮儿
输出：吃将校不吐将校字眼儿，不吃将校倒吐将校字眼儿。

这个分词存在一些问题，比如将“葡萄皮儿”这个词给分开，变成了“葡萄”和“皮儿”，从而在替换过程中导致无法将这两个词关联起来；“倒吐”是一个副词加动词的形式，但结巴分词没有将其分开。同时这种方法的另一个问题就在于词性标注上，之前测试时任意词性均可以用相同词性的词替换，结果导致了很差的结果，就是因为词性标注出现了问题。改为只允许名词替换后才有上面的结果。

另一个思路是直接利用发音相似的词生成绕口令，比如“打发”、“下家”等这样两个字的韵母相同的词，如果他们中间用一些其他的字隔开，则会比较符合绕口令的特点。因此我对思路就是首先将这样的词全部找出来，按照其韵母存储。在生成的开始阶段随机一个词，接下来按照其韵母找到所有这样的词语，从中挑选出下一个词。但是我们将这两个词用一个句子连起来，从而达到交错押韵的效果。因此我将开始选的词放在语料库中查找其出现的句子，对于每个句子检索是否包含具有有相同韵母的候选词，如果有就将中间这段话取出来作为这两个词的连接。然后以第二个词为基础继续寻找下一个词，这样整句话是由一系列韵母相同的词语连接而成，中间包含一些句子使其贯穿整一整句话，从而一定程度上满足要求。但在实验中发现，由于 NLTK 的语料库 `sinica_treebank` 容量太小，所以选择的两个词语几乎不会在一句话中同时出现，所以生成的文本只是这些词的简单连接，比如：

大法打打塔帕下家氩查查

所以我认为如果有更大的中文语料库，两个词可以找到句子进行连接，则这种方法可能更加可行。

因此对于语音学模型，它更适用于语料库比较充分的英文分词，可以产生在语音上非常恰当的绕口令。为了使其满足一定的语法规则，可以加入词性模板进行优化，从而使生成的句子更具可读性。对于中文绕口令生成问题，这个方法不是非常合适，主要原因在于中文的语音相关的语料库较为匮乏，从而使训练样本不够充分。

3.5 循环神经网络方法

循环神经网络方法在实现上比较复杂，一种方式是手工实现网络，另一种是利用已有的深度学习框架（比如 TensorFlow、Torch 等）。在本次实验中对于网络模型的使用的是别人的实现⁴⁵。首先对于网络的拓扑结构采用 3 层的 Char-RNN 网络，每次训练 25 个字符，每个隐藏层具有 1000 个结点。使用该网络分别对中文和英文数据进行训练。

- 在前几轮迭代中，几乎不会输出具有意义的结果，比如：

```
x gr th
M' tithis whide apeu s rocvr rhoguif bpon ear nit s
s we sE r I e t
H ved oe tol H
'drl fxtoultufaeoung Rh't' I wo wfeedee ton t we tc
zkalrden ct s Hfw;reote hel te ckt Gonterhtrd beio ootm
```

- 在经过一定时间的迭代过程以后，该网络已经学到了需要按照空格来划分单词，在生成文本的时候也是以单词为单位进行生成。比如：

```
I shiy bleped priefter,
I'cry blew a vich deter ofle coustner coulyno's at.
A san Chock goursy for shoptor seles pream
```

虽然生成的语句依然没有可读性，但相比前几轮已经有了很大的提升。

- 在经过近 8 个小时的训练后，网络已经训练到可以提取出相邻词语的发音要相似这样一种特征。在经过 50 大轮迭代后，产生的文本如下：

⁴<https://github.com/hit-computer/char-rnn-tf>

⁵<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

But the most poorest and mourish panoplosaurus called Hander
Tommy Tucker tried to train his tongue to twist and turn to two
tall trees

While we were walking we were watching window washers watcher
watches Which watches the catcher who's watching the watcher
watches the catcher who's watching the balls

这几条绕口令都不是语料集中的原文，但他们都就有一定的绕口令的特点，而且更难能可贵的一点是，该网络会自动根据韵脚的不同进行换行。由于 Char-RNN 在训练过程中没有提供任何额外的信息，因此换行符是网络自己输出的。说明它经过训练已经意识到了不同绕口令之间需要用换行符隔开，可以提取出这种高级特征，这正是深度神经网络的一大特点。

和英文绕口令相比，中文绕口令由于语料集比较小，所以使用神经网络进行训练的难度比较大，很容易出现过拟合的现象。在实验过程中，一开始使用的是一个比较浅层的网络，就是基本的 RNN 模型。在训练的前几轮会出现和英文训练结果类似的现象，即会随意输出。

上了不，上施扁两偏果树在扁在上前我罗四在鹿，三飞扁五偏，偏
吹挂北扁拿扁果上青三木上头

严扁虫

。扁本垂等板，老担

，三了上，板三斗西上，六柿上住取七老果上末鹅，扁在山让帮日
偏果长七上三过果扁果

果问果醋。扁果上，扁了，一腿果偏，，果蓝在扁在扁，偏，扁力老
让下飞灯凳偏绸是果渡了上还上，老；

飞上两偏飞上飞来水上了扁了十担斗，笙让来凳绳了板市寺饽一百
谢。两果扁史扁了上叔勤水刻八扁

吐。偏，扁房大在上果

在经过长时间的迭代以后，输出结果会比原来要好一些，但依然没有把握住绕口令的特征，各个字之间的发音关联比较弱。但在经过 400 多万次迭代（4、5 个小时）以后，出现了严重的过拟合现象，比如：

华华园里有一株藤萝花, 佳佳园里有一株喇叭花。佳佳的喇叭花,
五彩霞网住麻花鸭, 麻花鸭游进五彩霞
马妈妈磨麦面, 磨了麦面燕麦面馍。马妈妈忙了磨麦面, 又去忙蒸
麦面馍
小世十四斤四两细蚕丝不换司小四四十四斤四两西红柿, 追老爷子,
老爷子一生气,

每一句均是语料库中的原文。网络的 loss 函数已经从最开始的 170 多到 0.6 左右, 因此网络出现了严重的过拟合。我认为之所以会出现这个问题, 是因为网络的层数过浅, 而且没有加入记忆单元, 从而导致输入很快影响到整个网络, 因此可能会产生梯度消失或者梯度爆炸的现象。另一个原因是训练集的数据太小, 导致网络在训练过程中很不稳定。之后改变了 RNN 模型, 使用 and 上文中英文训练一样的模型进行训练, 效果要好很多。经过 4 个小时左右的迭代过程以后, 网络可以达到收敛状态。生成的文本如下:

鼠倒有个窗, 窗上一个莺, 就个芦捆鹰,
糠坏住来把虎, 就要真能比是鹿。
高上北娘灭来三起地, 一十个捆层子, 回飞三只子, 跑了一鞋子,
放进了笼子,
林一布, 搁了醋,
再面熟叔鼠,
手他打捉谷, 几来好只牛, 还知不夜买进头。

从这个测试结果中可以看到, 网络生成的文本虽然还有一些问题, 但基本抓住了绕口令的一些特征, 比如押韵、格式对仗等。而且这几句话也不是语料库中的原文。因此使用这个网络训练出来的模型在中文绕口令生成上的效果还是比较好的。之所以比英文要差, 一方面是由于中文绕口令本身比英文要复杂, 要考虑到中文分词的问题, 以及中文语义、语音, 而英文不仅不需要分词, 而且语音特征很大程度上可以由字母序列反应出来; 另一方面就是中文语料库太小, 虽然在实验中我也努力去搜集更多的中文绕口令, 但搜到的大部分内容都是重复的, 也没有人去建立一个非常规范的大规模中文绕口令语料库, 这给自动生成中文绕口令的任务带来了很大的困难。但总的来说使用循环神经网络的方法对于这个问题是可行的, 如果有充分大的训练集应该可以取得比较好的结果。

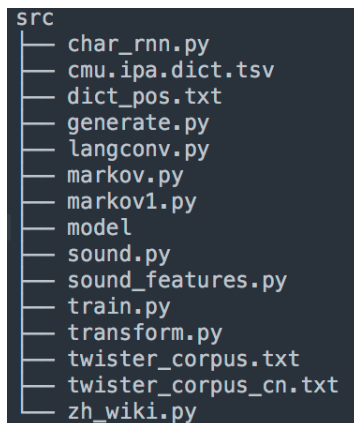
4 实验结论

本实验中，通过自己构建中、英文绕口令语料库，用多种模型实现了中、英文绕口令的自动生成。其中对于马尔科夫链模型，由于是直接绕口令语料集上进行训练，因此对于样本量比较大的英文绕口令生成效果较好，而对于样本量偏小的中文绕口令则生成的文本和原文非常接近；对于语音学模型，它比较适用于英文绕口令的生成，可以使生成文本使用的单词的发音非常相似，在加入词性模板以后生成语句的可读性有所增强；对于循环神经网络模型，我认为对于中、英文两个网络，由于数据量较小在训练上都不是非常充分，但它们均已经可以提取出一些高级特征，比如相邻单词的发音要相似这一特征，而且对于中文绕口令的生成效果也要比语音学模型要好。因此，在绕口令生成这一问题上，虽然神经网络的方法依然占据一定的优势，但传统方法依然有其适用的情形。对于这三个模型，依然存在这改进的空间，比如对于循环神经网络模型，可以对网络的拓扑结构进行进一步的调整，可能有助于改善网络训练的稳定性；对于中文的语音学模型可以使用更大的中文语料库（不需要词性标注）进行训练；而对于马尔科夫链模型，如果在更大的训练集上，则更长的记忆长度应该可以取得更好的效果。同时对于中文和英文的绕口令，目前没有一个客观的评价标准来判定它的优劣，这也和绕口令的风格多种多样有关，但总之目前还难以对一个绕口令生成模型的性能进行定量刻画。这些都是未来可以进一步研究和实验的内容。

在此次实验中，不仅将自己所学的自然语言处理的相关知识应用在了绕口令自动生成这一问题上，而且在实验过程中还自己研究了循环神经网络的相关内容，并将其用于解决这个问题。同时我在实验中我还了解到了一些语言学的相关知识，比如音系特征等内容，算是这次课程项目实验的额外收获。

5 附录

本实验源代码 src 目录的结构如下图所示：



```
src
├── char_rnn.py
├── cmu.ipa.dict.tsv
├── dict_pos.txt
├── generate.py
├── langconv.py
├── markov.py
├── markov1.py
├── model
├── sound.py
├── sound_features.py
├── train.py
├── transform.py
├── twister_corpus.txt
├── twister_corpus_cn.txt
└── zh_wiki.py
```

Figure 4: src 目录文件结构

- twister_corpus.txt 和 twister_corpus_cn.txt 分别为英文和中文绕口令语料库；
- cmu.ipa.dict.tsv 为 IPA 标音的 CMU 英语发音词典；
- dict_pos.txt 为从结巴分词中提取的中文带词性标注的词典；
- markov1.py 和 markov.py 为马尔科夫链模型的代码，第一个为语料库上的马尔科夫链模型，第二个为布朗语料库上的马尔科夫链模型；
- sound.py 和 sound_features.py 为英文的语音学模型的代码，sound.py 为运行文件，里面包含了词性标注的模板；
- transform.py 为中文的语音学模型代码；
- generate.py 和 train.py 为 RNN 的测试和训练代码；
- char_rnn.py 为浅层 RNN 的代码，可以观察中间输出结果；
- model 为 RNN 的模型参数文件夹。

以上代码除了神经网络的相关代码(generate.py、train.py、char_rnn.py)外均使用 Python3 运行，神经网络使用 Python2 运行。由于模型参数过大(model 文件夹有 4G 多)，所以没有上传。⁶

⁶百度云地址为 <https://pan.baidu.com/s/1o8qCRFk>