基于 GPT-2 模型的可控主题押韵歌词生成系统开发与研究

简宇卿 [1]
指导老师: 孙茂松 [2] 袁中果 [1] 王璐 [1]
[1] 中国人民大学附属中学
[2] 清华大学自然语言处理与社会人文计算研究中心

摘要

近些年来，随着中国音乐市场用户大量激增，人们对优秀音乐作品的需求也在不断增加 [1]。任意一首歌曲都分为歌词和旋律两部分，一方面旋律会给人带来直接的影响，如音高、节奏、强度和音色方面 [2]，另一方面能更加体现歌曲内涵的则在于歌词。同时，近年来自然语言处理在各类文本处理任务中都产生了相当不错的效果 [3]，由此，本次研究结合对歌词创作的理解与自然语言处理中 GPT-2 模型在文本生成上的优势 [4]，建立 LSTM 押韵模型学会歌曲押韵技巧并修改 GPT-2 自回归系统，使用 LDA 模型对歌词进行主题分析，得出了歌词的主题模型后，整合主题与歌词数据对 GPT-2 散文模型进行微调，使得使用者可以随意选择“主题 + 主要内容 + 押韵风格”让机器完成自主创作并演唱。并将本模型与其它模型的歌词生成效果进行了对比，在此基础上建立了前端音乐展示系统。生成的歌词押韵丰富、辞藻华丽、逻辑严谨，可供音乐爱好者及歌词创作者借鉴与欣赏。

1. 引言

中国的音乐发展一向与时俱进，近些年“网易云音乐”、“QQ 音乐”、“酷我音乐”等音乐平台的飞速发展，《歌手》、《中国新说唱》等节目的产生，大力推动了中国音乐圈的发展，同时也使一批批优秀的音乐人涌现。而成为一个专业的音乐人对大众来说依旧具有较高的难度 [5]，创作出合格的歌词作品需要作者有大量的生活积累与文化素质 [6]，对于普通人来说，创作出合格的歌词甚至都是一种奢望 [7]。虽然目前的机器创作不能代替人类创造性的工作 [8]，但是却可以替代一些重复性工作或者给予创作者灵感与提示。经过问卷调研，发现身边有很多人都希望能够进行音乐创作活动，本文的目标就在于为大众的音乐创作提供帮助，降低成为音乐人的难度和门槛，使得人人都有机会创作出有特色的的歌词，创造出属于自己的音乐。

歌词生成有两种方法 [7]，一种是基于统计的方法，通过对主题的把控进行生成 [9]，但是往往这样的生成方式连贯性和逻辑性效果较差，读起来不通顺。其次就是通过深度学习的方法，通过建立模型或神经网络对现有的歌词数据进行学习，这样就可以根据输入的内容不断进行生成，但是这个方法也存在一定的问题，就是容易出现生成的歌词重复、语义不通顺等问题 [7]。本文一方面建立了更好的模型来解决上述问题，另一方面通过建立用户交互式系统，让用户自主挑选好的生成结果来使生成效果提升。

本文使用了基于 LDA 的歌词数据主题分析方法，通过歌词数据建立数据集之间进行无监督训练，再通过人为标注的方式对 150 首歌进行 4 个主题下的分类，并使用模型进行准确度测试，通过多次试验确定最佳参数。

本文构建了基于 LSTM 的文本押韵模型，来检测歌词数据中的押韵模式。通过人工总结的普通话韵脚，对每句歌词末尾字的韵母进行学习，通过

LSTM 神经网络预测下一句歌词的韵母，得以完成机器对歌词押韵模式的学习。并使用该模型对主要生成模型进行约束，得到控制生成歌词押韵的效果。

本文提出了一种基于主题分析控制文本生成的方法，通过基于 LDA 已经判断好主题的数据，将每一首歌词的主题放置在歌词前也作为数据的一部分，并使用新数据训练 GPT-2 模型，得到可以根据给定主题进行生成的可控文本生成模型。

通过对模型的直观评测，本文发现了该模型生成文本重复度较高的问题，提出了两种解决方案，并通过直观的生成效果观察，发现问题得到了有效的解决：

- 基于破坏性删除歌词数据的办法，对数据中的重复语句进行去重，可能会破坏歌词整体结构。
- 基于微调 GPT-2 模型的办法，不再使用数据重新从零训练，而是选择基于其他模型进行微调。

本文建立了一种客观可信的生成模型评估方式，不仅使用 PPL 等方法判断生成效果，更采用问卷的评估模式对 100 位不同学历程度的用户进行调研，从语句通顺度、内容一致度、整体质量进行判断，根据专业歌词的赏析办法 [10]，在“主题性”，“逻辑性”，“重复度”，“流畅性”，“韵调性”，“冲击性”六个方面进行“图灵测试”，对比其它模型和人类创作的歌词与本文介绍的方法生成的歌词的效果。

本文的模型和之前已有的模型相比，在六个方面均表现十分出色，且以假乱真，让评测者普遍认为生成的歌词非机器所创作。

最后文本根据模型建立了音乐交流平台，可以快速部署让大众体验 AI 辅助作词功能，并结合数据库进行情感分析与用户反馈通过 MVP 试错模式不断更新、系统强化。

我们还将要把知识图谱融入模型中，使得模型的生成歌词可以更加具有故事性和逻辑性。

2. 相关工作

在歌词生成任务上，国内研究者普遍使用统计学的方法或是 RNN 神经网络的方法进行训练和生成。在童俊滔 [10] 的工作中使用 LSTM 训练歌词数据开发了一套歌词系统。2018 年《中国好声音》24 岁清华大学生物医学工程系大学生四年级博士生宿涵，使用 RNN 通过华语乐坛千万歌曲训练系统，人工智能辅助改编周杰伦的《止战之殇》迎来大众好评。

在 Robbie Barrat 的工作中使用 LSTM 在 Kanye West 的唱片编目上进行了训练，可以使用提供的任何歌词，并逐字逐字地写新歌，并在一定程度上保持流畅。

但是上述方法使用传统神经网络，难以把控歌词句与句间逻辑关系。

在 GPT-2 模型方面，有高长宽在《鬼吹灯》小说续写与古诗生成方面的研究，可以通过其 demo 展示看出 GPT-2 模型在生成任务中出色的效果。

3. 基于 GPT-2 的主题生成模型

概述. RNN 系列的网络只能顺序或逆序 (从左到右或从右到左) 进行计算，RNN 对于 50 个词之前的词顺序就不敏感了，而对于 100 个词之前的就完全忘了，所以 RNN 在长序列上是有问题的，虽然 LSTM 较为有效地解决了这一问题，但在长序列的理解还是有一定的缺陷。而 transformer 结构依靠两组嵌入进行序列的表示，词表嵌入和位置嵌入。位置嵌入方法使用正余弦函数，对位置 pos 和维度 i 有如下公式:

$$PE_{pos,2i} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{1}$$

$$PE_{pos,2i+1} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2}$$

其中 d_{model} 为词嵌入的维度，在此为 512。

通过对 RNN 系列与 transformer 系列在各种任务上的性能对比，得出结论：有大量数据时用 transformer；允许 pretrain 时使用 transformer；小数据时，分类用 CNN，标注和生成用 RNN。自 Google 在 2018 年 10 月底公布 BERT 在 11 项 NLP 任务中的卓越表现后 [11]，BERT 就成为 NLP 远近闻名的模型，BERT 类似月类似于只含编码器的 transformer 结构。4 个月后，BERT 终于迎来了一个新的“对手”——GPT-2 根据 openAI 的官方论文可以发现

语言模型	模型	编码器	主要亮点
LM	ELMO	LSTM	两个单向语言模型的拼接
LM	GPT-1	Transformer	首次将 transformer 应用于预训练语言模型
LM	GPT-2	Transformer	调参简单，生成任务取得很好效果
DAE：MLM	BERT	Transformer	MLM 获取上下文相关的双向特征表示
DAE：E-MLM	RoBERTa	Transformer	预训练中采用动态 mask
PLM	XLNet	Transformer-XL	双向上下文表征 + 双注意力流

表 1. 各类模型对比分析

事实上 GPT-2 和 GPT 的差异并不大，很类似于不含编码器的 transformer 结构。

在此表1对文本生成系列进行对比，发现 GPT-2 模型调参简单且生成任务取得很好效果，于是选择方便修改和操作的 GPT-2 模型作为主要生成模型。

模型原理. GPT-2 基于 transformer 结构，简而言之由两部分而成:

- 无监督预训练部分。由于使用的是 transformer 中的 decoder 结构，可以简化表示为以下公式:

$$h_0 = UW_e + W_p \tag{3}$$

$$h_i = transformerblock h_{i-1}, \forall i \in [1, n] \tag{4}$$

$$P(u) = softmax(h_n W_e^T) \tag{5}$$

其使用交叉熵的办法构建优化函数

$$L(U) = \sum logP(u_i|u_{i-k}, \cdots, u_{i-1}) \tag{6}$$

- 有监督的微调部分。其方法为:

$$P(y|x_1, \cdots, x_m) = softmax(h_i^m W_y) \tag{7}$$

可发现其实际需要添加的参数仅有 W_y 这一个，所以十分便于调整。

想要运行一个训练好的 GPT-2 模型，最简单的方法就是让它自己随机工作 [12]; 在随机情况下，只简单地提供一个预先定义好的起始单词，然后让它继续生成文字。模型会对这个数据进行处理，最终获得一个对应的向量。这个向量对于词汇表的每个单词都会有它作为下一个词的概率，可以选择概率最高的作为下一个单词。但有时这样会出问题——就像如果持续点击输入法推荐单词的第一个，它可能会陷入推荐同一个词的循环中 [12]，所以不能每次都选择概率最高的，需要找到一种方法跳出这种循环，在此有两种可以对这个问题进行解决:

- Top-k 方法，通过每次截取下一个词的概率最高的前 k 个，再在其中随机进行选。
- Top-p 方法，通过每次截取下一个词的概率大于等于 p 的，再在其中随机进行选。

在此我们选择 Top-p 方法，这样的选则方法更加稳定。

模型结构. 通过中文版 GPT-2 模型，由于数据量较大，于是直接选择中等大小词表作为词表，修改数据输入方式直接使用全部歌词数据进行训练，调节参数直接进行训练，具体模型参数如下:

- $nvocab$: 21128, 字典大小，有 21128 个不同的符号和英文单词
- $ctxnumber$: 1024, 文本的长度，比如输入文本有 1024 个单词
- $Embeddinglength$: 768, 词向量，位置向量，以及内部特征向量的维数
- $headnumber$: 12, 多头注意力，12 个注意力磁头，平分 768 个隐藏特征
- $layernumber$: 10,10 层 tranformer 结构

效果观察. 通过对模型的观察，发现 GPT-2 在以下两点十分出色:

起始字	下一词	下一词概率	短语出现次数
颠	沛	1.00	10
过	往（去）	0.83（0.17）	16782（1891）
救	我（人）（护）	0.73（0.16）（0.11）	1384（127）（56）

表 2. 词组生成分析。GPT-2 生成效果的各项数据是在一个已经使用数据训练了 3 个 epoch 的模型上得到的。

起始句	下一句	下一句概率	韵脚
我爱上了她	想带她回家（她慢慢长大）（可是我无法）···	0.1623 0.1034 0.0927 ···	“啊”字韵
我也曾感到彷徨	也曾感到失望（也感到迷茫）（看到那光）···	0.2512 0.2311 0.1912 ···	“昂”字韵

表 3. 押韵生成分析。GPT-2 生成效果的各项数据是在一个已经使用数据训练了 3 个 *epoch* 的模型上得到的。

- 模式 1：短语和高频词语。从表2可明显看出 GPT-2 模型对短语的学习能力极强，这一点是传统神经网络几乎不可能做到的，对于每一个字通过 GPT-2 模型得到的下一个候选字与这个字组成的词语都可以有易于理解的意思，而往往传统神经网络得到的候选词会有很多语义不搭的，也说明 GPT-2 有效地降低了困惑度，增强了生成文本的可读性。
- 模式 2：押韵匹配。从表3发现该模型很好地学会了歌词中押韵的运用，体现出模型对发觉数据中深层规律的能力极强。

问题和解决方案. 通过输出结果发现模型仍存在一定重复度的问题，在此对重复度问题进行分析：

- 结论 1：模型倾向于寻找重复模式（重复词/重复换行）
- 结论 2：重复词的概率随采样长度增加而逐渐增加，候选词分布受历史词的影响逐渐偏向重复词，信息量逐渐降低

重复度问题一直是文本生成中的大问题，通过尝试本文提出以下两个解决方案：

- 有前人工作发现训练模型写新诗时，如果加上歌词数据，则重复度会大幅上升。通过对歌词数据分析发现，92.15% 的歌词数据中存在重复情况，更有 17.12% 的数据中存在多重复情况（一句歌词重复三遍以上），所以在此由于结论 1 提出假设：通过对歌词的去重可以降低训练效果重复度。
  操作：如表4，首先对所有歌词进行去重，相同两首歌曲之保留一首，其次对每首歌词进行破坏性去重，每首歌中的重复歌词之保留第一次出现的那一句。
  通过去重后进行重复度检测，重复度有如下定义：
  - 定义 1：过度重复，即让人感觉重复度已经超过正常歌词的情况
  - 定义 2：过度重复率，即过度重复 sample 数/总 sample 数
- 尝试对网友训练好的散文模型进行微调，根据之前所说有监督的微调部分的操作方法，微调模型操作简单，且会在原参数上进行改进并保留原模型的性质。

使用两种方案后重复度如表5所示，可发现两种方案效果良好，且对歌词的整体逻辑性影响不大。

4. 基于 LDA 的歌词数据主题分析

文本聚类主要是依据著名的聚类假设：同类的文档相似度较大而不同类的文档相似度较小 [13]。作为一种无监督的机器学习方法，聚类由于不需要训练过程以及不需要预先对文档手工标注类别 [14]，因此很适合对文章话题进行分类等任务。文本聚类主要有一下几个步骤：

- 对数据进行分词；
- 去除数据中的停用词；

原歌词（带“*”为重复歌词）	去重后歌词
雨过后的空气贴近有吻的痕迹	雨过后的空气贴近有吻的痕迹
风过境的天气落叶情绪积累成倾盆大雨	风过境的天气落叶情绪积累成倾盆大雨
你走之后的雨季一直在持续	你走之后的雨季一直在持续
* 你走之后的雨季一直在持续	我困在被淋湿的梦境
我困在被淋湿的梦境	你留下来的雨衣
你留下来的雨衣	还有温暖水汽
还有温暖水汽	晒不干身体里的记忆
晒不干身体里的记忆	
* 雨过后的空气贴近回忆旧旧的凉凉的有吻的痕迹	
* 风过境的天气落叶满地关不紧蓝色情绪积累成倾盆大雨	

表 4. 破坏性删除方式。可能破坏部分歌曲完整度。

	原数据	去重后数据
直接训练	0.60	0.10
微调散文模型	0.50	0.08

表 5. 两种解决方案后测得的过度重复率。

- 构建词袋空间 VSM；
- TF-IDF 构建词权重；
- 使用聚类算法进行聚类。

每首歌曲都有较为明显的主题（Topic），为了之后系统能够根据主题生成歌曲，对歌词数据进行主题分析这一环节十分重要。主流的模型有两种，一种是 pLSA 另一种是 LDA。但随着文档数量的增加，pLSA 的参数也会随着线性增加 [15]，这就导致无论有多少训练数据，都容易导致模型的过拟合问题，于是在此我们使用 LDA 模型。

LDA 是一种非监督机器学习技术，可以用来识别大规模文档集或语料库中潜藏的主题信息。它采用了词袋的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。

LDA 的核心公式如下：

$$p(w|d) = p(w|t) * p(t|d) \tag{8}$$

直观的看这个公式，就是以主题作为中间层，可以通过当前的 θd 和 ϕt 给出了文档 d 中出现单词 w 的概率。其中 $p(t|d)$ 利用 θd 计算得到，$p(w|t)$ 利用 ϕt 计算得到。实际上，利用当前的 θd 和 ϕt，我们可以为一个文档中的一个单词计算它对应任意一个主题时的 $p(w|d)$，然后根据这些结果来更新这个词应该对应的主题。如果这个更新改变了这个单词所对应的主题，就会反过来影响 θd 和 ϕt。LDA 算法开始时，先随机地给 θd 和 ϕt 赋值（对所有的 d 和 t）。然后上述过程不断重复，最终收敛到的结果就是 LDA 的输出 [16]。

为了使数据中一些多用词（“我”，“的”，“很多”，……）不会对模型产生影响 [17]，需要事先挑选出来为下一步构建语料库做准备。于是使用采集的中文停用词典数据直接正则化捕获所有中文字词并存储至列表中。

导入歌词数据并对其按照篇目进行切分，对每首歌的歌词使用第三方程序库 jieba 进行分词处理。之后使用 Python 第三方程序库 gensim 的函数创建对应字典，并用该字典转换切词数据为最终语料库。

使用 gensim 的函数对数据进行训练，并使用 save 函数保存模型数据。通过对各种参数的分析，在此唯一有意义去调整的参数为 numbertopics [18]，这个参数决定了 LDA 模型的“主题”数，也就是最

numbertopics	2	5	10	20	50	100
准确度	0.315	0.675	**0.92**	0.875	0.752	0.774

表 6. 不同 numbertopics 下在测试集（50 首）中的准确度

不使用主题模型处理	使用主题模型处理
一首写“看法”的歌：	一首写“看法”的歌：
你的爱是那么的深	这是我的看法
你的爱是那么的深	我的梦想在这儿长大
我想我了解你的心	我的家
你的手我不能放开	每天都在变化
我的情你的心	我的生活是多么的无奈
你的心你的手我不能放开	但我还是感谢你的爱
爱你的心你的一切我都明白	所有的挫折和失败
爱你的一切我都明白	让我感谢上帝的安排

表 7. 不使用主题模型和使用主题模型后生成样例中的差异。可以看出前者不属于“看法”类歌曲，属于“爱情”类。而后者则满足要求。

后分类过程中要进行筛选的主题个数。

首先打印出每次 numbertopics 数量的最优结果，可以通过其简单而直接地初步估计模型效果，并从中挑选出无明显确定涵义的字和字词加入停词库，反复几次，发现当切词全模式时效果明显更好，但是还有很多单字占据了大概率字词，于是在建立语料库时滤掉所有单字，效果显著提升。

使用人为判断数据方法对模型效果进行定性分析。即人为规定分类主题总数（主题数 5：”感想”,”梦想”,” 时光”,” 家乡”,” 爱情”），并对每个主题挑选 10 首该主题的歌词调整到歌词数据前 50 首，用 gensim 的函数得到这 50 首在 numbertopics 上的分布并画出对应的三维柱状图，分析预测准确率，通过表6发现当 numbertopics 为 10 时效果最佳，分布较为清晰。

在此基础上，根据这 50 首在 numbertopics 上的分布概率选取最高概率，并进行频数分析，发现梦想主题和看法相似度过高，于是将这两项合并为看法，最后得出最高概率主题对应的真正主题。

使用该主题模型对所有歌词数据进行主题分析，并将每一首歌词的主题放置在歌词前也作为数据的一部分，形如格式：“写一首‘主题’的歌：(歌词)”。并使用新数据训练 GPT-2 模型，使得生成时输入“写一首‘主题’的歌：”，模型即可根据用户所选主题进行特定主题的生成，效果如表7所示。

5. 基于 LSTM 的文本押韵模型

概述. 人在读书时不是一个字一个词去理解，而是一句一句、根据上文进行解读。传统的神经网络并不能做到这一点，这是在对序列信息（如文本）进行预测时的一个缺点。而 RNN（Recurrent Neural Network，循环神经网络）可以通过不停的将信息循环操作，保证信息持续存在，从而解决上述问题 [19]。

在过去的几年里，RNNs 在语音识别、文字建模、翻译、字幕等领域有很成功的应用。而 LSTM 则是一种特殊的 RNN 网络结构 [20]，它的重复单元不同于标准 RNN 网络里的单元只有一个网络层，它的内部有四个网络层。

LSTM 具有三个门：输入门、输出门、遗忘门。其中输入门为控制数据的输入，输出门控制数据的输出，遗忘门用于过滤信息将重要的信息保留下来 [20]。

不使用押韵模型处理	使用押韵模型处理
一首写”爱情”的歌：	一首写”爱情”的歌：
我想你了	我忘不了她的存在
一起去看海	忘不了她在的日子里
看日出就像是一本日记	我的心里从此刻沉重起来
你说你爱的人是我	我不能接受这个现实
不管别人怎么看	我是有点无奈的
这个世界都有你在	我只是希望这样的日子能有个可以倾诉的角色
只不过是我想要	让你知道我的存在是多么有压抑的选择
一切都是自己	也许我的努力就够了

表 8. 不使用押韵模型和使用押韵模型后生成样例中的差异。可以看出前者完全不押韵，而后者则满足跳句单押。

	$Char-RNN$	$LSTM$	$seq2seq+attention$	本模型	真实歌词
歌词主题性测试	7.44	5.78	7.56	**8.22**	7.67
歌词逻辑性测试	5.33	6.89	7.11	**7.44**	7
歌词重复度测试	4.44	4.22	3.89	**4.67**	4.44
歌词流畅性	6.22	7.11	7.44	**7.78**	7.44
歌词韵调性	7.11	6.78	7.33	**8.33**	7.56
歌词冲击性	5.56	6.56	6.44	**6.67**	6.56
认为最有可能为人类创作概率分布	0.081	0.051	0.121	**0.531**	0.216

表 9. 100 位评选者分别打分，打分范围从 0 到 10 的整数，每项分数均为平均值

押韵模型构建. 首先手动整理出比较干净且规范、长度较合适的歌词共 3M，句与句之间以换行符划分，所有篇目直接堆叠。

枚举出所有汉语拼音的字符序列，再通过对歌词中的大量押韵的研究 [13] 与欣赏而总结得来的规律对押韵的字符序列进行分类，得到一个二维列表。通过函数得到汉字的拼音并在二维列表中查询返回所在位置，同一韵的汉字的返回值相同。将这个程序封装为函数。

通过 Python 程序库 keras 建立 LSTM 网络，并将数据按行读入通过押韵函数建立数据集“歌词 + 长度 + 押韵”。将数据集转化为向量传入 LSTM 模型中开始训练并保存参数。

最后用马尔科夫链随机生成文本并同样将数据转换为“长度 + 押韵”的形式并转换为向量，随机选一句使用模型的预测函数进行预测，并将预测结果作为输入重复一百次，将所有生成的向量进行存储。将向量一一与所有生成文本进行匹配并计算分数，将高分语句依次输出并储存到 txt 文件中。

模型分析. 当训练代数等于 5 时发现 loss 值下降明显，但升高 epoch 为 20 时模型发生了过拟合，于是对 epoch 为 5 的生成样本进行分析，发现机器不仅学会了句句单押，甚至学会了跳句单押，说明训练效果良好。在此基础上尝试修改押韵函数，改为寻找每句最后两个字的押韵，最后一个字的权重按比例放大，再进行模型训练发现机器学会了句句双押，更有跳句单押。说明 LSTM 在押韵方面可以学会多押及一些高级的押韵技巧，猜想可以通过与 seq2seq 转换模型的结合让机器学会更多歌词结构上的特点。

使用押韵模型约束主体生成模型. 使用该模型对主要生成模型进行约束，设置一个韵脚列表，生成时每次生成的句子如果最后一个字的韵脚不符合韵脚

列表中的规律，则重新生成该句。通过这种方法，得到如表8所示控制生成歌词押韵的效果。

6. 模型对比

如何对生成的文本进行评价也是文本生成研究中重要的一环 [21]。通过对现有研究的分析，这个部分有两大类的方法进行判定。一是技术上直接判断，具体方法有 CS [22]($Content\ Selection$)、CO [22]($Content\ Ordering$)、RG [22]($relation\ generation$)、$BLEU$ [23]（$Bilingual\ Evaluation\ Understudy$）、$PPL$（$Perplexity$）五种指标最为常见。这些方法都基本是对生成文本和输入数据进行不同的对比分析得来的。而这种非常规的生成任务仅仅用这些指标进行判断是很难具有说服力的，有用 GPT-2 进行文言文生成的研究采用问卷的评估模式对 100 位不同学历程度的用户进行调研，从语句通顺度、内容一致度、整体质量进行判断 [24]，在此根据专业歌词的赏析办法 [10]，在“主题性”、“逻辑性”、“重复度”、“流畅性”、“韵调性”、“冲击性”六个方面进行“图灵测试”，对比其它模型和人类创作的歌词与本文介绍的方法生成的歌词的效果。

在此本文对以下四个模型进行对比：

- $Char-RNN$ [25]
  $Char-RNN$ 模型是从字符的维度上，让机器生成文本，即通过已经观测到的字符出发，预测下一个字符出现的概率，也就是序列数据的推测。
- $LSTM$
  直接使用最基础的 $LSTM$ 神经网络对数据进行直接训练。
- $seq2seq+attention$
  采用 $Seq2Seq$ 模型，输入上句或歌名直接生成下句。反复将生成的下句输入网络，循环往复，可以得到整首歌曲。注意力机制使用的是 $LuongAttention$，在解码器生成下句的时候，会根据当前语境信息对上文的一些词汇进行重点关注，对词语的向量化编码采用的是提前训练好的 $Word2Vec$ 模型，用整个语料库预训练 50 个 $epoch$，作为预训练模型。
- 真实歌词
  有真实来源的人类创作歌曲，取自真实网络歌手。

评测项目有如下六个方面：

- 歌词主题性测试（歌词与主题相关程度）
- 歌词逻辑性测试（前后语句是否有逻辑问题）
- 歌词重复度测试（歌词是否过于重复）
- 歌词流畅性（歌词是否连贯）
- 歌词韵调性（歌词是否押韵、歌词念的拗口不拗口，如诗歌中的音乐性）
- 歌词冲击性（歌词给人带来的冲击感，情感调度）

效果如表9所示，可以看出本模型效果十分突出，甚至达到了以假乱真的程度。

7. 系统实现

Django 框架. Django 是一个开放源代码的 Web 应用框架，由 Python 写成。采用 MVT（Model-View-Template，模型-视图-模板）的软件设计模式。Django 的主要目标是使得开发复杂的、数据库驱动的网站变得简单 [26]。

语音合成技术. 语音合成和语音识别技术是实现人机语音通信，建立一个有听和讲能力的口语系统所必需的两项关键技术 [27]。在此使用百度语音合成 API 对歌词文本进行转换为音频。

MySQL **数据库连接**. MySQL 是一种开放源代码的关系型数据库管理系统，使用最常用的数据库管理语言–结构化查询语言进行数据库管理 [28]。在此我们使用 Django 对 MySQL 进行连接，在 MySQL 中存储文本信息。

整体框架构造. 首先建立用户注册与登录界面，在用户注册并登录后进入主页面，在主页面可以添加歌曲专辑（album），输入名称、作者和专辑风格并上传一张照片作为专辑封面即可完成创建。其次用户可以点击 add song 上传本地 MP3 文件并输入歌

曲名字便可以在专辑中添加歌曲。在搜索框中搜索关键字词即可访问他人上传的歌曲及专辑。

项目结合与展示. 在 AI-songs 界面用户可以输入一句话作为开头，点击 start with 即可开始生成。用户可在生成的五个 *sample* 中选择不错的歌词并修改输入到 start creating 第一个框中，第二个输入歌曲名称，第三个输入所属专辑名称。点击 start creating 若专辑存在即可生成歌曲，使用百度语音合成技术生成音频并使用 Python 调用 cmd 命令行 ffmpeg 工具将伴奏与语音音频结合，将信息自动添加至数据库，用户可在生成完毕后返回所属专辑进行查看和收听。

其它功能与系统优化. 网站还设有智能对话系统 [29] 并会对用户输入数据进行情感和主题分析，及时反馈以便于系统优化。

8. 总结与展望

歌词的主题有时很难判断，而通过 LDA 模型的聚类可以给出较为准确的分类，主题分类可以为音乐应用的推荐给予更大的发展空间，通过 LSTM 押韵模型对歌词数据的学习，可以发现机器学习的强大潜力，对歌词中技巧的掌握，有两方面的重要意义：一是可以给予押韵模型继续升级模型；二是可以作为歌曲的标准进一步创造歌曲。GPT-2 强大的学习能力是有目共睹的，这里只是人为的限定就让 GPT-2 轻松学会了歌曲的创作，再融入 LDA 主题分析后的“主题 + 歌词”，GPT-2 更能创造出以假乱真的歌曲，为众多歌词创作爱好者提供欣赏的曲目，为创作者们提供极大的灵感。建立网站以供更多用户体验和使用该系统，并使用“歌曲全部来自用户”和“MPV 试错”的管理方式迅速积累数据和流量，使用多种人工智能技术吸引用户兴趣。

本项目在未来有很大改进空间，主要有以下几个方面：

- 歌词数据的扩充和清洗：目前的歌词数据大概只占所有可用数据的五分之一，而且未来会有更多更好的新歌词出现。扩充数据量可以从头训练 GPT-2 模型从而获得更好效果。目前的数据中仍有一定的脏数据会在一定程度上降低训练和生成效果，所以进一步的数据清洗是很有必要的。
- 模型的改进：LDA、LSTM、GPT-2 均有很大进步空间，未来可以尝试融入 seq2seq 等以促进学习效果。
- 展示方式的优化：目前 Python 直接演唱初步具备展示功能，但和一首像样的歌曲还是有一定距离的，通过对 vocaloid 的学习和掌握可能在这个方面产生突破。可以尝试使用双层 LSTM 的方法生成相关主题伴奏，并使用解码编码器完成“Speech-to-Singing”的转化与对齐。

致谢. 本研究成果的成功，首先要感谢“翱翔计划”，能够给予我这样一个能和科研院所导师接触的机会，这样才能有机会接触到高级的算法来实现我预想的功能，感谢导师清华大学孙茂松教授对技术和论文写作上的指导，还需要感谢中国人民大学附属中学的王璐老师和袁中果老师给予我计划上的监督与指导，以及无微不至的关心。

特别感谢：

- 清华大学：李文浩　胡锦毅　刘家丰
- GPT-2 模型及网友散文模型参数：https://github.com/Morizeyao

参考文献

[1] 解啸尘. 浅谈中国音乐史和中国的音乐消费浪潮. 艺术科技, 1(12):146, 2018. 1

[2] 宋晓宇. 旋律与歌词对情绪体验影响的实验研究. Master's thesis, 西南大学, 2019. 1

[3] CoreJT. Corejt 自然语言处理中文文本基本任务与处理. 中兴通讯技术, 2019. 1

[4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 1(8):9, 2019. 1

[5] 杨蕾 et al. 用互联网服务音乐人. 财经天下, 1(21):36–37, 2015. 1

[6] 许自强. 歌词创作美学. 音乐教育与创作, 1(4):25–30, 2017. 1

[7] 童俊滔. 基于神经网络的歌词生成系统设计与实现. Master's thesis, 成都理工大学, 2018. 1

[8] 孙鹏文, 赵永红, and 富秀荣. 浅析人工智能和人类智能的关系. 内蒙古工业大学学报: 社会科学版, 1(2):35–37, 1999. 1

[9] 徐艳华, 苗雨洁, 苗琳, and 吕学强. 基于 lda 模型的 hsk 作文生成. 数据分析与知识发现, 2(9):80–87, 2018. 1

[10] 苏自勤. 浅谈歌词艺术的欣赏. 兰州教育学院学报, 7(3):7–10, 2007. 2, 8

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. 2

[12] 郭元晨 and Geek AI. 完全图解 gpt-2. 艺术科技, 7, 2019. 3

[13] 李函彦. 浅谈传统歌曲的押韵. 北方音乐, 1(15):41, 2018. 4, 7

[14] 百度百科. 文本聚类. 百度百科, 2019. 4

[15] AMiner 学术头条. 浅谈话题模型: lsa、plsa、lda. 百度百科, 2019. 5

[16] weixin30511107. Lda 主题模型算法. 百度百科, 2019. 5

[17] 崔彩霞. 停用词的选取对文本分类效果的影响研究. 太原师范学院学报: 自然科学版, 7(4):91–93, 2008. 5

[18] 紫巅草. word2vec 参数调整及 lda 调参. 百度百科, 2016. 5

[19] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, 2014. 6

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997. 6

[21] 龚恒. 文本生成概述. 百度百科, 2019. 8

[22] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 436–442, 2002. 8

[23] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Challenges in data-to-document generation. arXiv preprint arXiv:1707.08052, 2017. 8

[24] 黄石 and 林政. 基于深度学习的古代汉语生成模型. 百度百科, 2004. 8

[25] 百科. Char-rnn 原理介绍以及文本生成实践. 数据分析与知识发现, 2018. 8

[26] Jeff Forcier, Paul Bissex, and 徐旭铭. Django web 开发指南, 2009. 8

[27] 王仁华. 语音合成技术最新研究进展及其应用展望. 中兴通讯技术, 9(5):37–39, 2003. 8

[28] 崔洋. MySQL 数据库应用从入门到精通. 中国铁道出版社, 2016. 8

[29] 马莉. 人机对话系统中智能对话管理平台的研究与实现. 中兴通讯技术, 2004. 9

A. 附录 1：原数据放入 LDA 分析所得每个主题高概率词汇表

Topic 0th: 想要 0.010 不会 0.008 喜欢 0.008 时间 0.008 不想 0.007 身边 0.006 感觉 0.005 不要 0.004 时间 0.004 世界 0.004

Topic 1th: 兄弟 0.007 江湖 0.005 天下 0.004 天地 0.004 英雄 0.004 说唱 0.003 红尘 0.003 人间 0.003 千年 0.002 中国 0.002

Topic 2th: 录音 0.018 制作 0.011 朋友 0.009 作人 0.008 录音室 0.007 和声 0.007 哈哈 0.007 永远 0.006 慢慢 0.006 世界 0.005

Topic 3th: 没有 0.012 世界 0.006 知道 0.006 时间 0.005 回忆 0.004 不会 0.004 不是 0.004 不能 0.004 不知 0.003 太多 0.003

Topic 4th: 没有 0.011 不会 0.006 有人 0.005 一点 0.005 生活 0.004 不是 0.004 太多 0.004 世界 0.004 需要 0.003 时间 0.003

Topic 5th: 不见 0.006 世界 0.004 没有 0.004 时光 0.004 生活 0.003 岁月 0.003 不能 0.003 时间 0.003 青春 0.003 城市 0.003

Topic 6th: 知道 0.011 不是 0.010 没有 0.007 不要 0.006 喜欢 0.005 想要 0.004 生活 0.004 世界 0.004 不会 0.003 兄弟 0.003

Topic 7th: 好似 0.007 今日 0.005 二三 0.005 几多 0.005 三四 0.004 一齐 0.004 五六 0.004 四五 0.004 喂喂 0.004 边个 0.004

Topic 8th: 嘟嘟 0.009 喜欢 0.007 说唱 0.006 真的 0.005 不会 0.005 不是 0.005 不要 0.005 哥们 0.004 朋友 0.004 没有 0.004

Topic 9th: 不多 0.007 夏天 0.005 姑娘 0.004 再见 0.004 妈妈 0.004 想要 0.004 爸爸 0.003 哎呀 0.003 哥哥 0.003 有钱 0.003

B. 附录 2：模型评估测试数据

$Char-Rnn$:

那一刻我落下了眼泪/我等待苍老了谁/红尘醉/微醺的岁月/你发如雪/凄美了离别/我焚香感动了谁邀明月/让我心碎的雨/雨刷的复月/你的爱我们没有了/我的美丽/我的爱溢气/我用盡我的那棵树/全还不是我忘了/你说你不懂/你的迷靠/噙🈵/我不能再多战/我只能够远难/晕开薄荷漫/雨上的模样

$LSTM$:

那一刻我落下了眼泪/我不想变成一个虚无/也不想变成一个废物/我想在死去之前觉醒为了生命的尊严和希望信仰在空中飘扬/当黎明和落日的光阴交错的时刻/我们纷纷逃出每一座尖叫的城市/一道眩目的光芒从远处的天际显现/如同自由将我们迷茫的心灵唤醒/照亮他们前方艰难而曲折的道路/这不曾是我们想要的光明

$seq2seq+attention$:

那一刻我落下了眼泪/但偏偏风渐渐把距离吹得好远/但故事的最后你好像还是说了拜拜/你低头地亲吻我/我却微笑绝不说/认真的我都不说/琴弦的嘴却配着我的模样/那一块梦就像画一在那张里/而我看不见你的脸

本模型:

那一刻我落下了眼泪/我的心在痛我却说不后悔/我要用酒麻醉/不管你是谁/我已经爱上你的美/爱上你的美我不后悔/只要你能看到我流泪/我愿意这样为你心碎/想你的夜总是很美/爱你的心总是很狼狈/不要说什么对与不对/因为我已无路可退/不要让我的爱败给谁/不要对我再次颓废

真实数据:

那一刻我落下了眼泪/忽然间看见真实的自己/那一刻忍不住说后悔/忽然明白脆弱也是种力量/那一刻我承认我害怕/忽然想卸下男人的坚强/那一刻只有一个愿望/想大声地告诉你/我爱你/当狂风和巨浪撞击我的胸膛/我还用爱担当着我的翅膀/当惊涛和骇浪淹没我的希望/我却渴望伸手轻触你的脸庞