



南京大學

本科畢業論文

院 系 计算机

专 业 计算系科学与技术

题 目 一个音乐推荐系统的设计与实现

年 级 2010 学 号 101220119

学生姓名 王瑶菁

指导老师 徐锋 职 称 教授

提交日期 2014 年 4 月 30 日

南京大学本科生毕业论文（设计）中文摘要

毕业论文题目：一个音乐推荐系统的设计与实现

计算机 院系 计算系科学与技术 专业 2010 级本科生姓名：王瑶菁

指导教师（姓名、职称）：徐锋 教授

摘要：

中文摘要。

关键词：推荐系统，音乐推荐

南京大学本科生毕业论文（设计）英文摘要

THESIS: Design and implementation of a music recommendation system

DEPARTMENT: Department of Computer Science and Technology

SPECIALIZATION: Computer Science and Technology

UNDERGRADUATE: Yaojing Wang

MENTOR: Professor Feng Xu

ABSTRACT:

English abstract.

KEY WORDS: Recommendation System, Music Recommendation

目 录

1	引言	1
2	背景介绍	3
2.1	LAMP 网站结构	3
2.2	Storm 分布式计算	4
2.3	LDA 主题建模	5
2.3.1	模型介绍	5
2.3.2	LDA 在算法中的应用	6
2.4	DRPC	7
3	准备工作	8
3.1	音乐标签的获取	8
3.2	音乐的获取	9
4	系统设计	11
4.1	系统功能	11
4.2	系统结构	11
4.2.1	算法结构 (Storm)	11
4.2.2	网站文件结构	12
4.2.3	B/S 结构	13
4.2.4	数据库结构	13
5	系统实现	15
5.1	系统网站实现	15
5.1.1	网站前端实现	15
5.1.2	网站后台实现	18
5.2	系统推荐算法实现	18

5.3 系统内部交互	18
5.4 系统运行流程	18
6 结论与展望	19
参考文献	I
致谢	II
A 附录一	III
A.1 附录 1.1	III

1 引言

数字音乐和互联网技术的蓬勃发展使歌曲的出现和传播速度大大加快，而云服务和智能手机的出现则为人们提供了更加遍历的途径来收藏或收听数量庞大的音乐作品。然而，海量的歌曲造成了严重的信息过载 (infomation overload) 问题，导致人们无法快速地从获取符合自己洗好的歌曲。为了应对这一问题，音乐推荐系统 (music recommender system) 应运而生，并且已为当前解决音乐领域信息过载问题非常有潜力的方法。

音乐推荐系统本质上是一种信息过滤系统，其通过对用户历史行为习惯、用户社会关系以及用户所处环境等因素的分析，帮助用户从不断增长的数据中过滤掉那些不必要的信息，从而为用户推荐符合其洗好和习惯的优质歌曲。目前，学术界和工业界给出了不少音乐推荐的解决方案，但其仍然面临如下两方面的挑战。

其一，歌曲的分类方法和标准众多，而且这些标准往往不具有排他性，导致难以利用一组统一而准确的特征对歌曲进行刻画。比如，歌曲可以按风格分为流行、民谣、乡村、摇滚、爵士、电子、说唱等类型，也可以按照心情、年代、场合、乐器、语言等标准进行划分。但是，单一层次的划分往往具有片面性，如按风格划分时可能会忽略歌曲对应的心情属性，而完全按照心情划分又可能忽略歌曲对应的场合属性。此外，一首歌曲也并不绝对地属于某一个类别，而是以不同的概率隶属于不同的类别，即不同类别对歌曲的贡献程度有所不同。比如张雨生的“大海”即属于流行乐曲又属于经典乐曲，还带有怀旧的属性。因此，在刻画歌曲时不能粗糙地将歌曲划分到某一类别中，而应该充分考虑歌曲在各种类别之间的不确定性。

另一方面，以基于内容的推荐和协同过滤推荐为代表的传统推荐算法往往忽视用户所处的上下文环境，从而导致推荐的效果不佳。与书籍、电影等消费代价高的物品相比，歌曲时间短，消费代价低，同时具有更强烈的感情色彩。因此，用户对歌曲的消费具有很强的主观性且与用户所处上下文环境有着十分密切的关系。用户在不同的天气状况、不同的场合以及不同的情感状态下对歌曲的消费都会大相径庭。因此，我们需要充分考虑用户所处上下文环境，根据用户所处的环境和状态为其做出推荐。

本系统使用了王守涛提出的基于多维时间序列分析的音乐推荐算法，强调了

用户在一个确定的播放列表中或者一定的会话周期内所收听到的歌曲构成的序列往往能够很好地反应用户所处的上下文环境，对这种序列进行分析和预测将能够在一定程度上提高推荐的效果。

当前来说，互联网上也不乏一些知名的音乐推荐网站，比较著名的有 Last.fm, doubanFM 等等。其中 Last.fm 采用了协同过滤的方法为用户做推荐，认为如果两个用户之前喜欢过同一首歌曲，那么之后也会喜欢相同的歌曲。doubanFM 则是注重于歌曲的分类，如前面所说，歌曲的分类方法和标准太多，分类的效果取决于分类的粒度大小。同时，互联网中的这些音乐推荐网站都有一个获取用户行为数据的能力，将获取到的数据用来改进推荐的算法是正常的做法。但是这些数据并不是所有人都可以拿到，所以本系统在为用户提供一个优秀的音乐推荐网站之外，也是为了获取用户行为数据，以作后期研究使用。

本系统会以网站的形式呈现，不使用任何框架，网站架构为 LAMP，浏览器兼容性上暂时只支持 Chrome 浏览器，其他浏览器在 UI 以及功能上或多或少会出现一些 bug，此问题有待修复。推荐算法部分使用分布式计算框架 Storm 进行实时计算，以便于更快地获得推荐结果。推荐结果使用 topN 方法推荐给用户，这样也降低了计算集群的计算量。

2 背景介绍

2.1 LAMP 网站结构

LAMP 是指一组通常一起使用来运行动态网站或者服务器的自由软件名称首字母缩写（加粗部分为本系统使用的软件）：

- **Linux**，操作系统；
- **Apache**，网页服务器；
- MariaDB 或者 **MySQL**，数据库管理系统（或者数据库服务器）；
- **PHP**、Perl 或 Python，脚本语言；

LAMP 所有组成产品均是开源软件，是国际上成熟的架构框架，很多流行的商业应用都是采取这个架构，和 Java/J2EE 架构相比，LAMP 具有 Web 资源丰富、轻量、快速开发等特点，微软的 .NET 架构相比，LAMP 具有通用、跨平台、高性能、低价格的优势，因此 LAMP 无论是性能、质量还是价格都是企业搭建网站的首选平台。

	System	Server	Storage	Script
Yahoo	FreeBSD + Linux	Apache	MySQL	PHP
Facebook	FreeBSD	Apache	MySQL + Memcached	PHP
Wikimedia	Linux	Apache + Lighttpd	MySQL + Memcached	PHP
Flickr	Redhat Linux	Apache	MySQL + Memcached	PHP+ Perl
Sina	FreeBSD + Solaris	Apache + Nginx	MySQL + Memcached	PHP
YouTube	Suse Linux	Apache + Lighttpd	MySQL	Python

图 1: 经典 LAMP 网站结构

举例来说，Wikipedia，免费自由的百科全书，运行的一系列软件具有 LAMP 环境一样的特点。Wikipedia 使用 MediaWiki 软件，主要在 Linux 下开发，由 Apache HTTP 服务器提供内容，在 MySQL 数据库中存储内容，PHP 来实现程序逻辑。如图1所示，雅虎（yahoo）、新浪（sina）、Youtube、Facebook 等国际知名网站使用的也都是 LAMP 网站结构。

2.2 Storm 分布式计算

Storm 是一个分布式的、容错的实时计算系统，它被托管在 GitHub 上，遵循 Eclipse Public License 1.0。Storm 可以方便地在一个计算机集群中编写与扩展复杂的实时计算，Storm 之于实时处理，就好比 Hadoop 之于批处理。Storm 保证每个消息都会得到处理，而且它很快——在一个小集群中，每秒可以处理数以百万计的消息。更棒的是它可以用任意的编程语言来做开发。

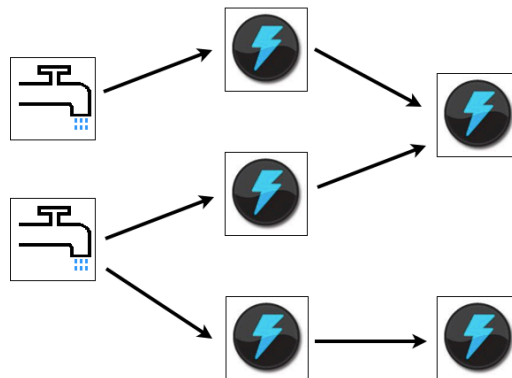


图 2: Storm 分布式计算模型

之所以选择 Storm 作为后台计算的模型，主要是因为推荐算法需要实时地反馈给用户，需要在较短的时间内将推荐歌曲的数据计算出来，而分布式计算很好地解决了这个问题。并且经过分析，本系统使用的推荐算法更适合使用流处理来计算，恰好 Storm 更适合流处理的计算。此外，Storm 具有以下很多特点：

1. 简单的编程模型。类似于 MapReduce 降低了并行批处理复杂性，Storm 降低了进行实时处理的复杂性。
2. 可以使用各种编程语言。可以在 Storm 之上使用各种编程语言。默认支持 Clojure、Java、Ruby 和 Python。要增加对其他语言的支持，只需实现一个简单的 Storm 通信协议即可。
3. 容错性。Storm 会管理工作进程和节点的故障。
4. 水平扩展。计算是在多个线程、进程和服务端之间并行进行的。
5. 可靠的消息处理。Storm 保证每个消息至少能得到一次完整处理。任务失败时，它会负责从消息源重试消息。

6. 快速。系统的设计保证了消息能得到快速的处理，使用 ØMQ 作为其底层消息队列。
7. 本地模式。Storm 有一个“本地模式”，可以在处理过程中完全模拟 Storm 集群。这让你可以快速进行开发和单元测试。

2.3 LDA 主题建模

LDA(Latent Dirichlet Allocation) 是自然语言处理中通过非观察组计算观测值来解释为什么数据中某些部分是相似的一个生成模型。

简单而言，就是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包涵词汇、主题和文档三层结构。文档到主题服从狄利克雷（Dirichlet）分布，主题到词汇服从多项式分布。

LDA 模型在推理上采用 Laplace 近似，变分近似，MCMC (Markov chain Monte Carlo) 以及期望-扩散 (Expectation-propagation) 等方法获取待估参数值。克服了使用优先混合模型 (Finite mixture model) 出现的局部最大值、收敛速度过慢以及使用 PLSA (Probabilistic latent semantic analysis) 出现的随文档数量增长导致的过度拟合。

2.3.1 模型介绍

目前的概率主题模型一般基于同一个思想——文档是主题的混合。

假设有 T 个主题，那么在所给文档中第 i 个词汇的 w_i 可以表示为：

$$P(w_i) = \sum_{j=1}^T P(w_i|z_i = j)P(z_i = j)$$

其中 z_i 是潜在变量，表示第 i 个词汇 w_i 取自该主题， $P(z_i = j)$ 表示在一个特定文档中主题 z 概率， $P(w_i|z_i = j)$ 是词汇 w_i 属于主题 j 的概率。假定 T 个主题形成 D 个文档以 W 个唯一词汇表示，为了记号方便，令 $\phi_w^{(z=j)} = P(w|z = j)$ 表示对于主题 j ， W 个词汇上的多项分布，其中 w 是 W 个唯一词汇表中的词汇；令 $\psi_{z=j}^{(d)} = P(z = j)$ 表示对于文档 d ， T 个主题上的多项分布，于是文本 d 中词汇 w 的概率为：

$$P(w|d) = \sum_{j=1}^T \phi_w^{z=j} \cdot \psi_{z=j}^{(d)}$$

在 ψ 上做狄雷克利分布 $Dirichlet(\alpha)$ ，在 ϕ 上做狄雷克利分布 $Dirichlet(\beta)$ ，在大多数情况下，在 $\alpha = 50/T$ 和 $\beta = 0.01$ 的时候效果最好。最后根据 Gibbs 抽样，构造收敛于某个目标概率分布的 Markov 链，并从链中抽取被认为接近该概率分布的样本，用下式可估算 ϕ 和 ψ 的值：

$$\tilde{\phi}_w^{(z=j)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + W\beta}, \tilde{\psi}_{z=j}^{(d)} = \frac{n_j^{(d)} + \alpha}{n^{(d)} + T\alpha}$$

期中， $n_j^{(w)}$ 表示词汇 w 被分配给主题 j 的频数； $n_j^{(\cdot)}$ 表示分配给主题 j 的所有词数； $n_j^{(d)}$ 表示文档 d 中分配给主题 j 的词数； $n^{(d)}$ 表示文档 d 中所有被分配了主题的词数。

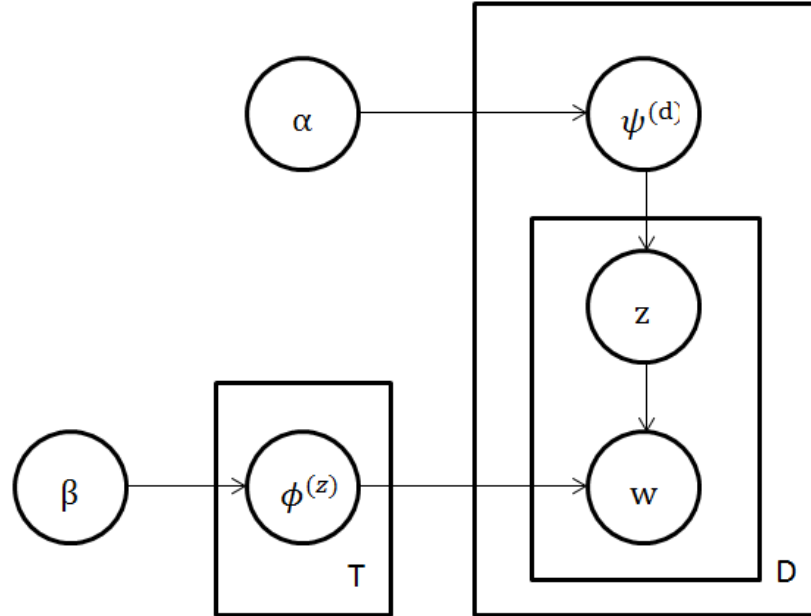


图 3: LDA 图解模型

图3所示的是 LDA 的图解模型，

2.3.2 LDA 在算法中的应用

LDA 算法在本系统中是作为一个离线的算法进行，主要用于对歌曲标签数据的建模，从而生成歌曲在 K 个主题上的概率分布，即可得到这首歌曲的隐含主题权重的向量：

$$s = (w_1, w_2, \dots, w_i, \dots, w_k)$$

其中，当 $w_i = 0$ 时表示歌曲完全不属于隐含主题 i ，即该主题对歌曲完全没有贡献；当 $w_i = 1$ 时表示歌曲完全属于该隐含主题 i ；当 $0 < w_i < 1$ 时表示歌曲在一定程度上属于该隐含主题 i 。

2.4 DRPC

DRPC 即分布式 RPC（Distributed Remote Procedure Call），用于对 Storm 上大量的函数调用进行并行计算过程。

DRPC 内部工作流程如下：

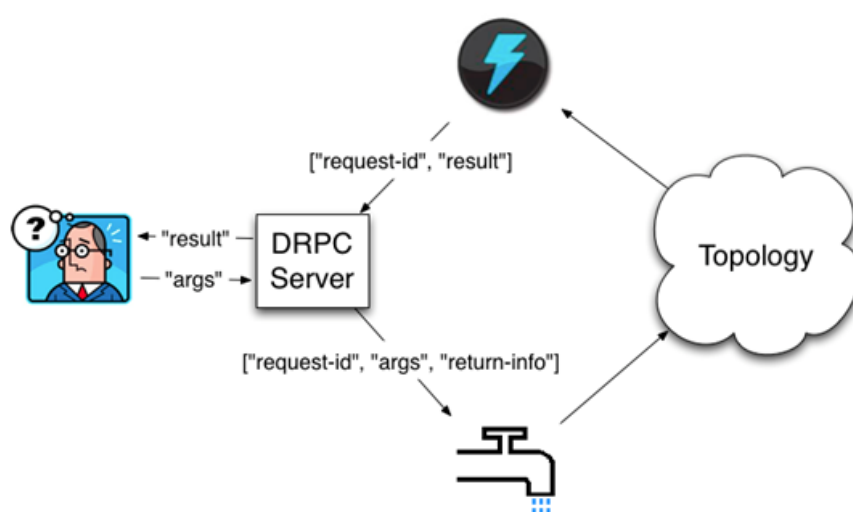


图 4: DRPC 工作流程

1. Client 向 DRPC Server 发送被调用执行的 DRPC 函数名称及参数。
2. Storm 上的 topology 通过 DRPCSpout 实现这一函数，从 DRPC Server 接收到函数调用流；
3. DRPC Server 会为每次函数调用生成唯一的 id；
4. Storm 上运行的 topology 开始计算结果，最后通过一个 ReturnResults 的 Bolt 连接到 DRPC Server，发送指定 id 的计算结果；
5. DRPC Server 通过使用之前为每个函数调用生成的 id，将结果关联到对应的发起调用的 client，将计算结果返回给 client。

3 准备工作

3.1 音乐标签的获取

互联网中能够为用户提供音乐标签数据的网站并不多，Last.fm 拥有庞大的标签数据库，但是对于一个中文音乐推荐系统，Last.fm 上的音乐偏国际化，并不适合这个系统。

```
1 {
2   "id": "24885384",
3   "tags": [
4     {"count": 1030, "name": "KatyPerry"},
5     {"count": 499, "name": "Pop"},
6     {"count": 359, "name": "欧美"},
7     {"count": 302, "name": "2013"},
8     {"count": 277, "name": "美国"},
9     {"count": 167, "name": "Single"},
10    {"count": 121, "name": "女声"},
11    {"count": 82, "name": "Single/EP"}
12  ],
13  "mobile_link": "http://m.douban.com/music/subject/24885384/",
14  "author": [{"name": "Katy Perry"}],
15  "title": "Roar",
16  "alt": "http://music.douban.com/subject/24885384/",
17  "attrs": {
18    "title": ["Roar"],
19    "pubdate": ["2013-08-12"],
20    "singer": ["Katy Perry"],
21    "tracks": ["01. Roar"],
22    "media": ["数字(Digital)"],
23    "version": ["单曲"],
24    "publisher": ["Capitol"]
25  },
26  "image": "http://img3.douban.com/spic/s26954792.jpg",
27  "music_url": "http://zhangmenshiting.baidu.com/data2/music/68725468/68725468.mp3",
28  "rating": {"min": 0, "max": 10, "numRaters": 3401, "average": "7.4"}, "alt_title": ""
29 }
```

图 5: 豆瓣 API 获取的音乐标签

在国内网站中能提供相对比较全的标签数据的网站，我选择了豆瓣音乐。豆瓣音乐有它自己的 API (http://developers.douban.com/wiki/?title=music_v2)，所以对于数据获取来说也会比较容易。然后使用豆瓣的 API 通过关键字搜索把歌曲信息（获取到的数据为 JSON 格式，图5）保存到本地（保存为 JSON 字符串）。

不过由于数据的保密性，豆瓣并没有把所有的标签数据公开给所有人，通过 API 也是只能获取所有标签中大概排在前 8 左右的标签以及其数量。考虑到用户行为的特殊性，那些没有被打过很多次的标签对推荐的效果也是微乎其微的，所以取前 8 名的标签做推荐的效果会比较好。

在获取到一定量的数据之后我对标签进行了分析，发现大多数标签内容集中在歌手这一块，从图6中可以发现大多数的标签都被歌手名所覆盖，其余的一些标



图 6: 豆瓣上的音乐标签 (带权重)

签则为一些分类中的大类，这种标签并不适合来做推荐，所以我选择对标签数据进行了优化。

首先，如果一首歌的便签最大 count 值比平均水平要小很多，那么从曲库中删除这首歌。然后，对有歌手标签的 count 值做调整，用一个参数将 count 值等比例地缩小，这个参数目前取值为 0.5。

3.2 音乐的获取

对于一个音乐 FM 网站来说，光有音乐的信息的没有用的，提供真实的音频文件供用户收听才是最重要的，所以在获取到一定数量的音乐信息后，就要获取到相应的音乐文件。我所使用的百度音乐 API 并不是公开的 API，但是百度作为国内最大的网络服务供应商之一，其拥有的数据量足够我们使用。这个 API 因为是百度内部使用的 API，据猜测应该是百度音乐软件所使用的 API，其返回结果为 XML 文档，通过解析可以获得歌曲和对于歌词的网络地址。由于歌曲的网络地址会随时间发生变化，所以所有的歌曲最后都被我下载到本地作为本地数据存储。

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <song>
    <id>123456789</id>
    <name>歌曲名称</name>
    <artist>歌手名称</artist>
    <album>专辑名称</album>
    <genre>音乐类型</genre>
    <year>发行年份</year>
    <label>唱片公司</label>
    <copyright>版权信息</copyright>
    <lyric>歌词内容</lyric>
    <url>歌曲下载地址</url>
  </song>
  <song>
    <id>987654321</id>
    <name>另一首歌曲</name>
    <artist>另一位歌手</artist>
    <album>另一张专辑</album>
    <genre>另一种音乐类型</genre>
    <year>另一发行年份</year>
    <label>另一唱片公司</label>
    <copyright>另一版权信息</copyright>
    <lyric>另一歌词内容</lyric>
    <url>另一歌曲下载地址</url>
  </song>

```

图 7: 百度 API 获取的 XML 数据

由于 API 的局限性，一些歌曲的下载地址不可使用或者音乐网络问题无法连接，因此用 Java 运行了一段程序来从网络上下载歌曲和相应的音乐信息：

```
1 public static void main(String [] args){
2     /*读取关键字以进行搜索*/
3     BufferedReader r = new BufferedReader(new InputStreamReader(new FileInputStream(new File(" ... ")))));
4     int index = 0;
5     String x = null;
6     while((x=r.readLine())!=null){
7         /*用豆瓣API搜索关键词，返回json数据*/
8         String jsonstr = doubanAPI(x);
9         /*对json中标记的每一首歌曲用百度API进行搜索*/
10        baiduAPI(jsonstr,index,x);
11        index++;
12    }
13 }
14 public static void doubanAPI(String s){
15     String url = "https://api.douban.com/v2/music/search?q="+s;
16     try {
17         /*用GET方法访问url获取数据*/
18         ...
19     } catch (IOException e) {
20         ...
21     }
22     return html_str;
23 }
24 public static void baiduAPI(String jsonstr, int index, String s){
25     JSONObject json = new JSONObject(jsonstr);
26     ... //各种变量定义
27     /*对json格式中的每一首歌曲搜索*/
28     for (int i=0;i<json.length;i++){
29         title = ...;
30         name = ...;
31         String url = "http://box.zhangmen.baidu.com/x?op=12&count=1&title="+title+"$$$"+name+"$$$";
32         try {
33             /*GET方式访问url*/
34             ...
35             /*解析XML文件获取歌曲下载地址*/
36             song = ...;
37             try {
38                 /*通过下载地址下载歌曲并保存歌曲以及歌曲信息*/
39                 ...
40             } catch (IOException) {
41                 /*捕获Timeout等异常*/
42             }
43         } catch (IOException) {
44         }
45     }
46 }
```

4 系统设计

4.1 系统功能

经过分析与比较，系统应该具有以下功能（或者要求）：

1. 用户注册登陆（支持游客登陆）
2. 音乐播放功能（包括播放/暂停、下一首、音量调节）
3. FM 网站的功能（喜欢、讨厌、打标签）
4. 美观整齐友好的用户界面
5. 较少的推荐时间

4.2 系统结构

由于是小型网站，并不需要庞大的后台统计功能，也就不需要使用一些现成的网站框架，从 0 开始搭建网站的话其实也给网站留下了很大的空间——任何功能都可以自己定制。选择 LAMP 的网站结构也是综合考虑了自身能力以及网站需求的结果。整个系统可以分为网站系统以及后台算法系统两部分，这两部分系统之间通过 DRPC 交互数据。因为网站系统需要处理用户的请求，服务器的处理能力有限，不能同时成为网站服务器和 Storm 拓扑中的一员，所以我将 Storm 分布式系统搭载到其他机器上。

整个系统的大体结构如图8所示：



图 8: 系统全局结构示意图

4.2.1 算法结构（Storm）

本系统使用由王守涛提出的基于多维时间序列分析的音乐推荐算法 (Music Recommendation Based on Multidimensional Time Series Analysis, MTSA)。

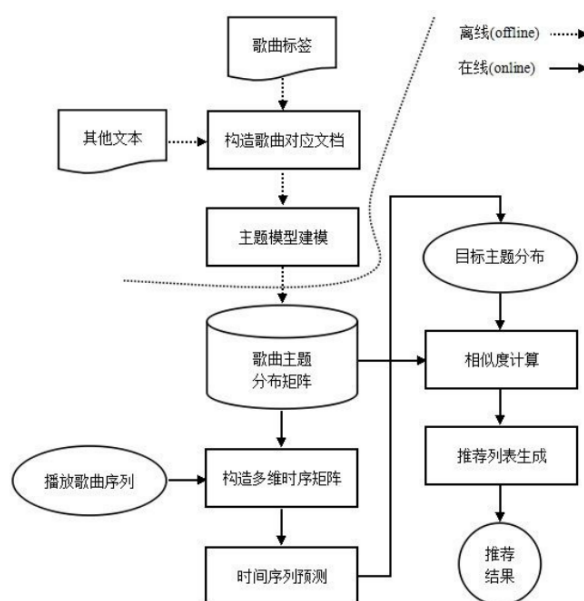


图 9: 基于多维时间序列分析的音乐推荐算法框架

该算法将使用隐含主题刻画歌曲的方法与通过时间序列分析技术分析用户行为的方法加以融合。首先，该方法通过主题模型建模将歌曲映射到对应的隐含主题权重向量，然后将用户在一定会话周期内收听的歌曲构成的序列按照该多维向量展开，进而得到一个以各个隐含主题对歌曲的贡献程度为变量的多维时间序列。通过对该多维时间序列的分析，我们可以预测用户可能收听到的下一首歌曲的特征并向用户推荐曲库中与预测歌曲类似的歌曲。

4.2.2 网站文件结构

网站的结构其实是延用了大三时候一个项目的结构，由于当时的项目比较大，我们还使用了 Smarty 来区分前后台，方便代码整合，现在这个网站其实用不着，所以就去除了这一块。

如图10所示的是整个网站的文件结构图，其中 public 是网站根目录 (整个网站只有一个页面即 index.php，保存在根目录下)；sys 则是 PHP 的配置文件、PHP 类和数据库的初始化文件；public 下的 dist 中保存的是网站所使用的一些插件；handle 中的文件用于前后台的交互，包括数据库的查询操作；music 文件夹下保存的则是音乐文件以及其相关信息。

sys 文件夹下的配置文件中包括一个访问记录的 Safeguard.php 文件，能够防止恶意用户不停对服务器发送 POST 或者 GET 操作。

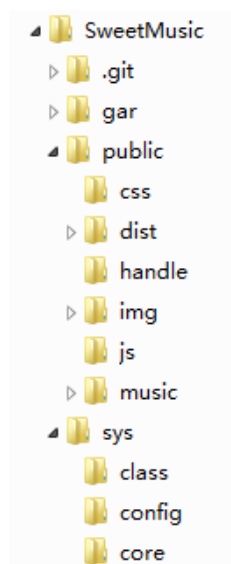


图 10: 网站文件结构

4.2.3 B/S 结构

把这个系统看成一个 web 应用，那么就应该是 B/S（Browser/Server）结构的 web 应用，B/S 结构可以通过图11来表示：

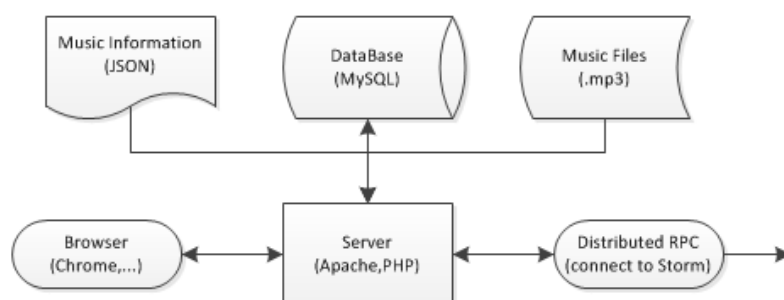


图 11: 网站文件结构

4.2.4 数据库结构

这里数据库分为三部分，第一部分为网站本身使用 MySQL 存储的数据库；第二部分为网站所用歌曲及其相应信息的数据库，这里以文件形式存放；第三部分为后台算法中使用到的由 LDA 模型计算得到的歌曲主题分布矩阵，这部分数据由系统预读到内存中。

第一部分的数据库类图如图：

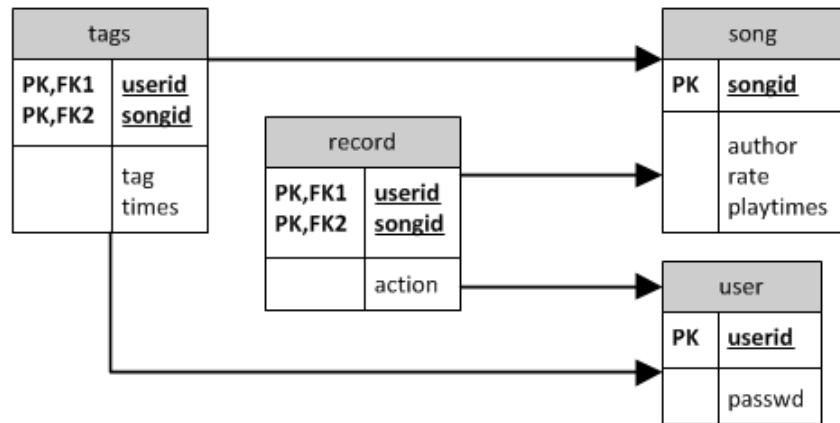


图 12: 网站数据库类图

其中 **user** 表用于存放用户信息；**song** 表存放所有的音乐数据；**record** 表记录用户行为，包括喜欢、讨厌、收听、跳过等；**tags** 表记录用户为歌曲打上的标签。

5 系统实现

因为本系统大体上分成了两部分进行开发，所以我将重点描述这两部分以及这两部分的交互，最后简单地走一边系统的流程。

5.1 系统网站实现

5.1.1 网站前端实现

网站前端即 HTML+CSS+JS，具体而言是使用了 HTML5 中的新元素 `<audio>` 标签，CSS3 的新样式以及轻量级 JS 框架 jQuery。由于有过网站开发的经验，所以在这部分主要研究的是 HTML5 中 `<audio>` 标签的使用方法。

`<audio>` 标签

`<audio>` 标签是 HTML5 的新标签，通过定义 `<audio>` 标签的 `src` 属性可以在网页上播放音乐，并可以通过 JS 代码来控制音乐的播放。

浏览器支持：






IE	Firefox	Chrome	Safari	Opera
				

图 13: `<audio>` 标签的浏览器支持（IE9+，Firefox，Chrome，Safari，Opera）

CSS3 样式

CSS 用于控制网页的样式和布局，在本网站中网页元素的定位大多使用绝对定位，一张页面足够放下，如果用户不习惯大页面的话，我也为用户提供了小窗口的入口。在设计网页的时候 CSS3 中的新特性对我的帮助很大，其中最重要的是圆角属性（`border-radius`），省去了很多麻烦，此外阴影属性让页面更有立体感。最终设计出的播放页面如图14所示：

整个页面以清新为理念，配以糖果背景，旨在让用户体验甜蜜的音乐之旅，让用户真正享受音乐。

轻量级 JS 框架 jQuery

jQuery 框架最大的特点就是“写得更少，做得更多”。顾名思义就是用更少的代码实现更多的功能，当然也是相对于传统的 JS，我更熟悉这一款框架。在这里，jQuery 的工作包括控制 `<audio>` 标签、控制网页动态效果以及与 PHP 交互。

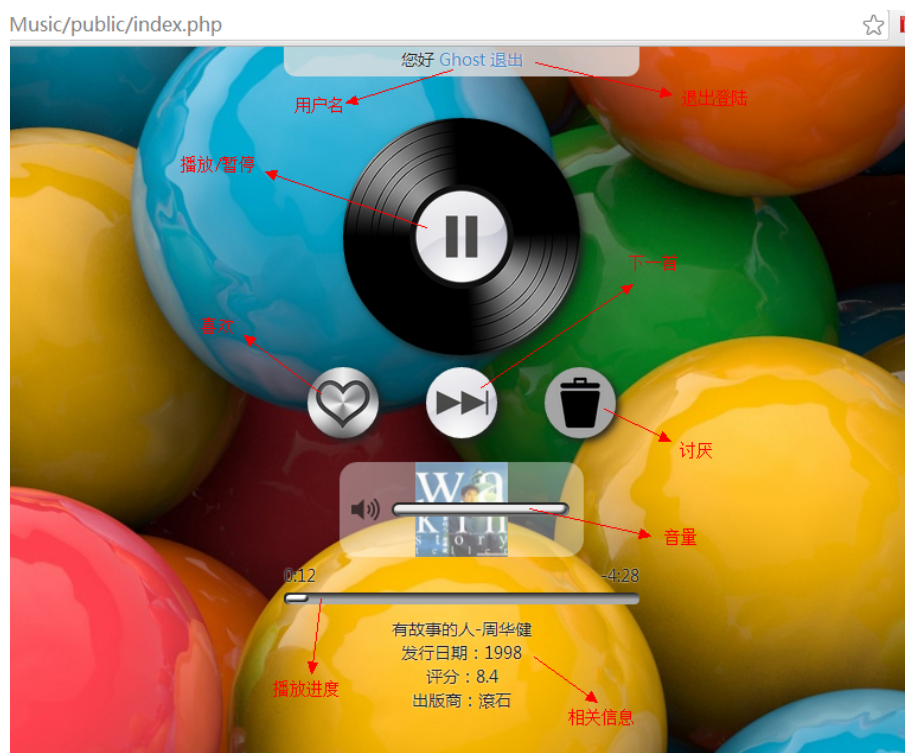


图 14: 播放时页面

登陆/注册

用户注册以及登陆是一个比较基础的功能，同时也是一个音乐 FM 网站用户记录用户行为的一个基本单位，当然也会允许游客的登陆收听。用户的登陆注册使用 PHP 的 SESSION 机制，所以用户登陆的信息会保存在浏览器 Cookie 中，只要 Cookie 还在就可以实现自动登陆功能。登陆以及注册页面如图15:



图 15: 登陆/注册页面

前端逻辑

接下来描述一下前台运行的逻辑。首先是用户逻辑:

1. 用户登陆（普通用户登陆转 2，游客到 3，注册到 4）；
2. 老用户到 5，新用户到 3；
3. 从推荐的 10 首歌中挑选最喜欢的一首收听，跳到 5；
4. 用户注册并登陆，跳到 3；
5. 播放音乐（或者播放推荐的音乐）；

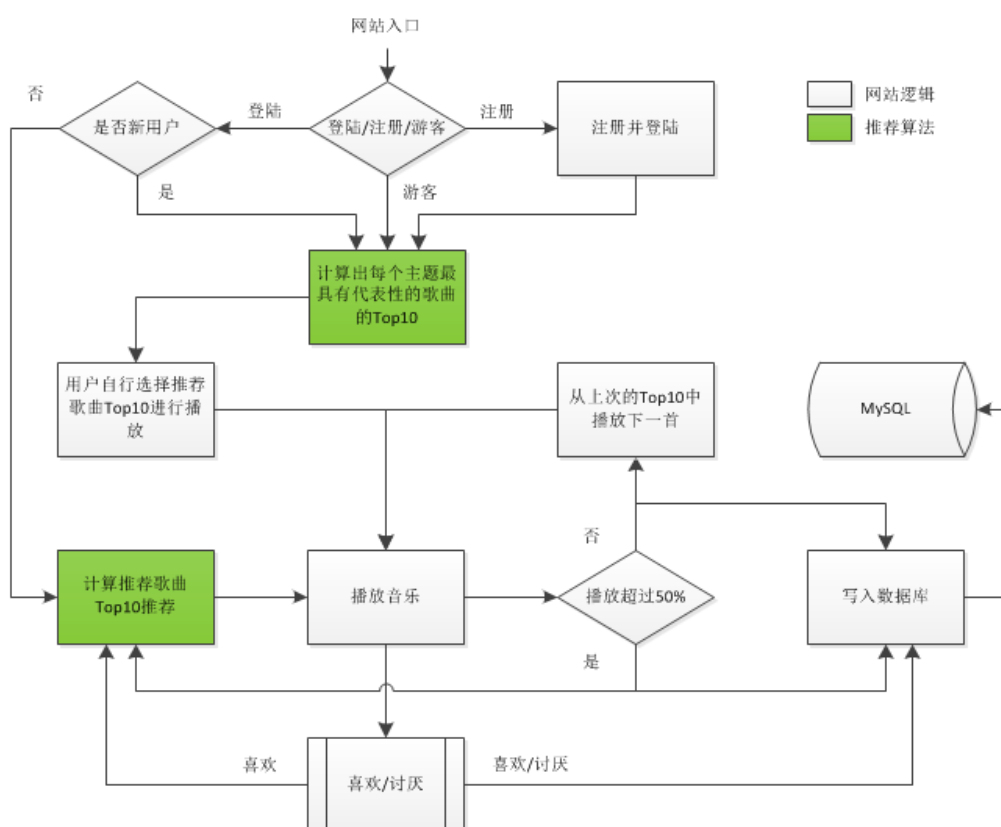


图 16: 网站运行流程

图16展示了整个网站运行的流程，期中绿色部分代表是推荐算法过程。

用户听歌的时候采取 TopN 推荐方式，一次推荐给用的歌曲不是一首，而是 N 首，在这里取 $N = 10$ ，用户会从这 N 首歌中顺序收听，直到用户喜欢其中一首歌，TopN 就会被重新计算。如果用户点选了“喜欢”按钮，那么说明用户便是喜欢这首歌，如果用户在听一首歌 s ，并且已经听了超过百分之五十，那么我也认为用户是喜欢 s 这首歌的。在用户点选喜欢或者听歌超过百分之五十的时候，把数据写到数据库并请求后台系统计算下一组 Top10 的推荐歌曲。

此外，在新用户或者游客登陆的时候，由于没有历史数据的存在，无法根据用户的播放历史推荐给用户合适的歌曲，这时候为了解决冷启动问题，系统将把曲库分为 N 个主题，每个主题选择一首最有代表性的歌曲组成 TopN 供用户选择。

5.1.2 网站后台实现

网站后台使用 PHP 进行管理，同时与推荐算法的接口也是通过 PHP。由于 PHP 后台文件是运行在服务器上，不需要在浏览器中运行，所以网站的文件结构中把 PHP 核心文件放在了根目录之外，即浏览器无法访问这些文件，这样一来就保证了 PHP 文件的安全性。

图17是网站后台 PHP 的 UML 图：

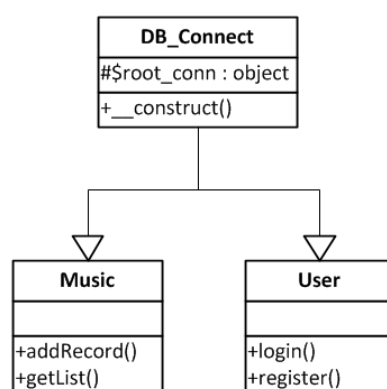


图 17: PHP 的 UML 图

其中类 *User* 和 *Music* 继承父类 *DB_Connect*，父类中的变量 *\$root_conn* 用于连接数据库，数据库配置文件也在根目录外部。类 *User* 中的函数分别负责用户的注册（Register）以及登陆（Login），登陆即把用户的信息保存在 SESSION 中。类 *Music* 中的函数包括从数据库中取出用户的播放列表（getList）以及记录用户行为（addRecord），用户行为包括喜欢（Like），讨厌（Hate），跳过（Skip）以及听过（Listened）。

5.2 系统推荐算法实现

5.3 系统内部交互

5.4 系统运行流程

6 结论与展望

参考文献

- [1] 王守涛. 一种基于多维时间序列分析的音乐推荐算法

致 谢

致谢内容

A 附录一

A.1 附录 1.1

附录内容