互联网+实验项目开题

* 1. 小组成员：

俞佩成（算法研究）、

马帅奇（Android开发）、

耿睿（UI设计）、

杨江锐、杨学昊（前端交互设计）、

危欢（后台+测试等）

项目名称：社区化的网络音乐播放器（互联网+音乐）

* 1. 研究背景
     1. 国内的情况

2013年网易正式发布网易云音乐，到2017年已经具备音乐播放，推荐歌单，听歌识曲以及社交等主要功能，是互联网+音乐的完美体现。分别来说，过去在网络上可以通过在线播放器听音乐，或者通过资源下载使用本地播放器播放。网络与音乐隔离较大，并且mp3/mp4保有着主要市场，互联网模式落后，不利于网络+音乐的结合。但是现在，互联网已经大大发展，大多数传统的音乐播放器，如酷狗播放器、千千静听、酷米等等都要么经过转型，面向用户提供跨平台的在线音乐播放支持和广泛的音乐仓库，或者慢慢消亡，消失在众人的记忆里。

比较不同的产品有助于我们发现业界标杆的优势，网易云音乐通过旗下互联网产品积累了大量的活跃用户；抓住音乐这个大众文化刚需，使受众更为广泛；大胆建立自己的云音乐库方便用户使用互联网下载音乐；并提供了关键的个性化歌单，大大增强用户粘性；在使用的过程中，提供用户之间相互交流的平台，让听歌不再孤单；此后的听歌识曲更是让云音乐如虎添翼。

* + 1. 国外的情况

对于国外，使用Play Store下载的music player不计其数，其中最优秀的是Audiomack(17), Samsung Music(22)和Google Play Music(34)。此3者的着重点稍有不同，S支持多种的音乐格式来提高自己的竞争力，还有控制音乐播放速度的能力，S和G均利用广告（可付费清除）增加变现途径，A包含心情歌单这项特殊功能等。而因为版权和听众的地域不同，但有的缺少歌词，有的用户引导不佳，或没有高清音质，或不能下载设置为铃声。

* + 1. 小结

我们认为国内目前这个环境造就了网易云音乐，虽然版权问题突出，但是为广大用户创造了良好体验。为什么国外的产品没有听歌识曲，我想应该是因为搜到的歌曲有版权导致App也无法下载，容易给用户造成不良的印象。而针对听歌识曲的应用，国内还有一款Music Radar。我们借鉴一个相对成功的音乐播放器，主要研究服务器高速解析客户端数据和兴趣挖掘算法的细节，学习如何发挥移动端轻量高效的优势。我们的目标大致列出如下：

* 用户注册、登录功能
* 好友系统
* 在线播放歌曲
* 推荐感兴趣的歌曲
* 歌词自动获取和解析
* 音乐列表（歌单）
* 在线点赞和评论系统
* 上传自己的音乐片段（高手在民间）
  1. 研究现状

针对以上需求，我们针对兴趣推荐算法和前后台交互的技术的调研如下。

* + 1. 一种基于模式的兴趣挖掘算法-李俊芳-2014年3月28日科技通报

总体来说，用户感兴趣的因素可以从应用内的搜索、会话的保持时长、访问的本体中具有的相似语义，或者留下的评论等因素来获得。为了能挖掘用户的兴趣点，算法的系统应该包括以下能力：从数据源发现兴趣、兴趣模型的表示、挖掘兴趣的方法。

本篇文献在前篇集中讨论了6种现成的兴趣挖掘方案的主要特性和缺陷（概略见下），从缺少表示用户兴趣的方法切入提出了自己的方案。

|  |
| --- |
| * 基于用户反馈标注的概率，解决兴趣漂移问题。这种方法能描述用户的兴趣并且可以更新，但是缺乏兴趣中间的语义关联。 * 使用SVM从用户的查询中识别其意图的方案，由于SVM从本体中分类出了摘要，通过传播激活过程推导出用户的兴趣，从而能够准确的描述兴趣的方向。 * 基于ODP标注提交搜索返回前10个查询结果、session中点击日志、查询结果点击记录等构建用户兴趣模型，但是结果仅限于查询过的session资源。 |

表格1.1 - 3种兴趣挖掘方案

方案从用户的行为记录中发现兴趣，不断从数据中更新用户的兴趣点，从而实现按兴趣推荐的功能。由于兴趣表示是兴趣模型的基础所在，因此作者提出了12项定义：

1. 兴趣粒子IP：等同于标签（概念集合记为C）;
2. 兴趣停留时间t：用户在某Ci映射的所有对象上停留的时长之和；
3. 兴趣持续时间T：Ci的两次访问时间Tnow & Tprevious差的绝对值，可以评判兴趣漂移；

|  |  |
| --- | --- |
|  | (公式1) |

1. 兴趣得分公式S(C)：多项式因子是访问的对象数V(C)、收藏数F(C)、和T(C)，并用权重表现不同因子的重要性；

|  |  |
| --- | --- |
|  | (公式2) |

1. 强兴趣QI：

|  |  |
| --- | --- |
|  | (公式3) |

1. 弱兴趣WI：

|  |  |
| --- | --- |
|  | (公式4) |

1. 无效兴趣NI：

|  |  |
| --- | --- |
|  | (公式5) |

1. 在e中，若有且仅有1个S(C)大于，则称为唯一兴趣OI，多个的集合称为多重兴趣MI，用以标定兴趣漂移；
2. 兴趣漂移DI：在相邻两段等长的时间里，若存在C从WI转变成QI或相反则称为兴趣漂移；
3. 长期兴趣LI：在相邻两段等长的时间里，存在S(C)始终大于，则称为长期兴趣；
4. 短期兴趣SI：在相邻两段等长的时间里，存在S(C)或S(C)，则称为短期兴趣。

将兴趣分类，调整权重和阈值的值就能成功把兴趣的表示完成了，此时再按照上述的步骤，从不同的因子获取兴趣粒子，计算它们的S(C)就可以分类成不同的兴趣模式P (stands for pattern)。

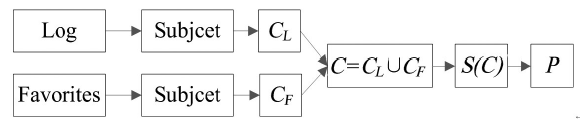


图1.1 基于模式的兴趣挖掘算法

这样应用可根据用户的兴趣模式，对长期兴趣和强兴趣对象进行推荐。

* + 1. 数据库访问对象模式+DBHelper抽象数据库访问技术

我们应该建立起实体类来承载歌曲的名称、时长、专辑、歌手等信息，本地存储将使用SQLite做数据库，建立查询表来存储对应的信息。因为字段较多，最好将反复CRUD的查询语句用DBHelper封装起来，DAO将业务接口暴露给应用。数据访问对象模式（Data Access Object Pattern）正是用于把低级的数据访问API或操作从高级的业务服务中分离出来的。

DAO模式需要以下参与者：

* 数据访问对象接口（Data Access Object Interface）-该接口定义了在一个模型对象上要执行的标准操作；
* 数据访问对象实体类（Data Access Object concrete class）-该类实现了上述的接口。该类负责从数据源获取数据，数据源可以是数据库，也可以是 xml，或者是其他的存储机制；
* 模型对象/数值对象（Model Object/Value Object）-该对象是简单的 POJO，包含了 get/set 方法来存储通过使用 DAO 类检索到的数据。

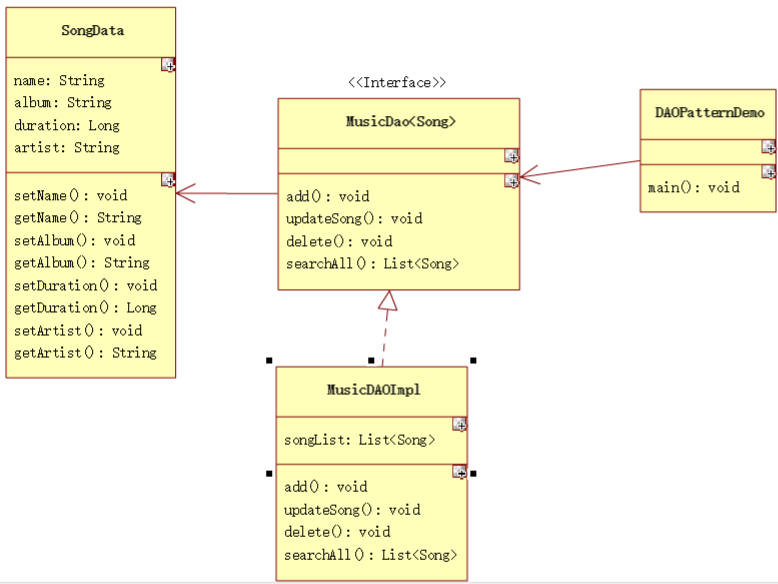


图1.2 DAO模式的参与者

当中的List就是从DB中获取的数据，遍历Cursor得到，从而受到类的维护，提高了重复访问的性能。我们需继承SQLiteOpenHelper，复写onCreate(db: SQLiteDatabase)和onUpdate(db: SQLiteDatabse, oldVersion: Int, newVersion: Int)。这样我们就可以在MusicDAOImpl中使用此DBHelper类的getReadableDatabse() / getWritableDatabase()进行安全的数据库查询，查询使用db的rawQuery()执行即可。

对于客户端来说，数据库操作就抽象为方法的调用和传递参数了。

|  |
| --- |
| public class DBHpler extends SQLiteOpenHelper {   public DBHpler(Context context) {  super(context, DBData.*DATABASE\_NAME*, null, DBData.*VERSION*);  }   @Override  public void onCreate(SQLiteDatabase db) {  //DO EXECUTION  }   @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  // DELETE THE OLD TABLES OR YOU CAN UPDATE THEM  } } |

* 1. 困难工程问题的提出

首先简介一下移动应用的开发架构与网页应用的差别。相比后者，它的模式更像C/S。Activity负责与用户的交互，显得像前端；由于Service组件可以后台执行需要长时间保持的任务（如数据解析，状态监测），或从数据库提供数据，Activity与Service相互通信，使得Service有点像后端。

但这里的Service并不是后端，一不需要框架二不考虑负载，app可以只用Service执行简单的录音工作。所以传统意义上的后端仍然体现在移动互联网应用的远程服务器上。

总览我们的项目，难点有二：

* App：开发的工作量会比较大，难度中等但需要相当的API和开发模式的知识；
* 算法：难点在算法的参数标定和代码实现上。

UI层次的杂项问题如下表所示。（耿睿添的，感觉不用留着删了就行）

|  |
| --- |
| * 歌词方案设计与实现 * 歌词显示原理 * 创建实体类LrcRow * ILrcBuilder接口和DefaultLrcBuilder歌词解析构造器 * ILrcViewListener接口（在ILrcView中使用） * 同步显示歌词功能 * 同步显示歌词功能 * 拖动歌词的功能 * 缩放歌词的功能 * UI适配机型 * 如何绘制歌词 |

* + 1. 研究工作的整体安排

正文

* + 1. 技术路线研究方法

正文

* + 1. 细化主要研究内容

正文

* 1. 工作可行性讨论分析

这一坨最后来

* + 1. 小组能力分析

正文

* + 1. 实施方案可行性讨论

正文