PaperPass检测报告简明打印版

比对结果（相似度）：
总　体：13 %　*（总体相似度是指本地库和网络库的综合比对结果）*
本地库：10 %　*（本地库相似度是指论文与学术期刊、学位论文数据库的比对结果）*
网络库：5 %　*（网络库相似度是指论文与互联网资源的比对结果）*

编　号：VIP9C324AAF27FA639B9
标　题：一种基于多维时间序列分析的音乐推荐系统研究与实现
作　者：王守涛
长　度：40575 字符(不计空格)
时　间：2014-5-25 8:14:53
比对库：学术期刊（1990-2013）、学位论文（硕博库1990-2013）、互联网资源
查真伪：http://www.paperpass.com/check.aspx

相似资源列表（学术期刊、学位论文）：
1. 相似度：1％ 篇名:《基于组合模型的医生推荐系统研究与实现》
   来源：学位论文 《东华大学》 2013 作者: 刘彭
2. 相似度：1％ 篇名:《团购商品个性化推荐系统的研究与实现》
   来源：学位论文 《浙江工业大学》 2012 作者: 刘超峰

相似资源列表（互联网）：
1. 相似度：1％ 标题：《《推荐系统实践》试读：第一章：好的推荐系统》
   http://book.douban.com/reading/18576782/

全文简明报告：

摘 要

近年来，移动互联网和智能手机的快速发展，使得音乐电台应用大量出现，比如Lastfm、Pandora、豆瓣电台等。 然而，此类应用的音乐推荐算法没有充分考虑用户听歌行为与其所处上下文环境之间的关系，导致推荐结果不够理想， 而少量考虑用户上下文的工作对用户行为的时间相关性处理不够合理，同样难以达到理想的推荐效果。 我们认为，用户的未来行为与其长期行为、即时行为和中期行为均存在相关性，但目前未见工作全面考虑用户行为的时间相关性。 为此，本文提出了一种基于多维时间序列分析的个性化音乐推荐方法，较合理地利用用户中期行为提高音乐推荐效果， 进而给出了一种综合用户长期、中期和即时行为的音乐推荐方法，通过原型系统的实现， {76％： 初步验证了上述方法的可行性。} 本文的主要贡献如下：

{52％： 1.提出了一种基于多维时间序列分析的音乐推荐方法。} 该方法采用主题模型将每首歌曲表示成隶属于若干隐含主题的概率，进而将用户行为建模为多维时间序列并对该序列进行分析， {44％： 从而较好地预测用

户行为偏好，并给出合理的推荐。｝

2.给出了一种基于用户长期、中期和即时行为的综合音乐推荐框架，考虑了三种行为影响权重的动态调整问题，进一步提高了音乐推荐的效果。

3.基于上述技术实现了一个音乐推荐原型系统，为提高处理效率，系统采用分布式实时计算框架Storm实现，达到了预期效果。

关键词： ｛46％：音乐推荐系统，主题模型，多维时间序列｝

第一章 引言

近年来，移动互联网和智能手机得到了快速发展，信息呈指数级增长，造成了严重的信息过载(Information Overload)问题 [54]。 面对海量的信息和众多的选择，人们往往无所适而陷入到选择悖论(Paradoxof Choice)[47]之中，或无法做出合理的选择，或需要消耗很大的精力才能做出正确的选择。 为了解决信息过载问题，减轻人们在抉择时所承受的负担，信息分类、搜索引擎和推荐系统等技术应运而生，如表1.1所示。

表 1.1: ｛60％：解决信息过载问题的主流方案｝

名称 特点 案例

｛50％：信息分类 分门别类地组织信息 Yahoo!｝ 、58同城

｛55％：搜索引擎 根据关键字进行查询 Google、百度｝

｛54％：推荐系统 分析用户行为历史，主动推荐 Amazon、淘宝｝

信息分类技术是通过将互联网上的各种信息分门别类地组织起来来提高人们查询效率的，比如早期的Yahoo!和国内的58同城。 ｛46％：然而，信息分类技术往往依赖于人工，可扩展性差。｝ ｛45％：搜索引擎技术通过建立索引的方式将互联网上的网页组织起来，然后根据用户输入的关键字进行查询，比如Google和百度。｝ ｛45％：显然，搜索引擎相对于信息分类更能够适应互联网的快速发展。｝ 然而，人们很多时候要么不愿意费时费力地输入关键字、要么无法准确地用关键字描述自己的想法，比如“今天看什么电影好呢？ ”、“去哪儿吃饭呢？”，这限制了搜索引擎作用的发挥。 为了解放人们的双手并挖掘人们的内在偏好和需求，推荐系统技术应运而生。

｛45％：推荐系统本质上是一种信息过滤系统，其通过对用户行为历史的分析挖掘出用户的行为偏好，｝ 进而帮助用户将海量信息中的无用信息过滤掉并将符合用户偏好的信息推荐出来[2]。 目前，推荐系统已经在多个领域得到了应用，比如电子商务领域的亚马逊和淘宝，视频领域的 Netflix和优酷， 个性化音乐领域的 Lastfm[27]和豆瓣电台[13]以及个性化阅读领域 Flipboard和无觅阅读。 ｛42％：图1.1展示了电子商务领域中Amazon的商品推荐系统界面和视频领域中PPTV的视频推荐系统界面。｝

图 1.1: Amazon电商推荐及PPTV视频推荐

文献 [41]表明，人们在日常生活中的听歌行为要远远多于读书、看电影等行为，这说明音乐已经成为人们生活

中不可缺少的一部分。 音乐推荐系统正是推荐系统在音乐领域的应用，其通过分析用户的收听习惯以及歌曲本身的特征为用户推荐符合其需求及偏好的歌曲[49]。 为了满足人们对音乐的个性化需求，一些音乐推荐系统被开发出来，如国外的Lastfm[27]、Pandora[37]以及国内的豆瓣电台[13]、虾米音乐[53]， {46%：图1.2展示了豆瓣电台和虾米音乐的界面。} 这些音乐推荐系统首先建立自己的曲库，然后分析歌曲的特征和用户的听歌习惯继而为用户做出推荐。 其中，Pandora[37]是当今最流行的音乐推荐系统之一，其通过“Music Genome Project”将400种属性分配给每一首歌曲，进而按照歌曲的相似程度为用户做出推荐。 Lastfm[27]、豆瓣[13]主要是根据 “相似的人往往具有相似的行为” 这样的假设为用户做出推荐。 虾米音乐[53]在给出了推荐结果的同时还给出了推荐的原因，这在一定程度上提升了用户对推荐结果的接收程度。

图1.2: 豆瓣及虾米音乐效果截图

音乐推荐与一般物品推荐关注的都是如何有效利用用户行为历史来分析用户的喜好并在此基础上为用户推荐其可能喜欢的物品， 因此基于内容的推荐[3][35][26]、协同过滤推荐[20][43][12]和混合推荐方法[39][11][34]等传统推荐方法可以在音乐推荐中得到应用。 {43%：然而，与一般的物品推荐(如电影推荐、书籍推荐、视频推荐等)相比，音乐推荐有如下特点：}

1.歌曲往往比较短，而且大多是免费的，因此人们消费音乐作品的代价比较低，可以一次性收听一组歌曲。

2.歌曲没有确定的划分标准，流派、心情、场合、年代等都可以用来划分歌曲。 比如，按照流派可以将歌曲分为流行、经典、摇滚等类别，按照心情可以分为欢快、伤感、忧伤等类别。

3.即使是按照一个统一的标准对歌曲进行划分，一首歌曲往往并不是确定性地属于某一个类别，而是以不同的程度分属多个类别。 比如，按照流派进行划分，“大海”既属于“经典”也属于“流行”。

4.用户在给定平台下收听歌曲时往往是按照次序一首接着一首收听的，这说明用户对音乐的消费与次序有很强的关联。

{56%：5.用户在不同的上下文环境下对音乐作品的选择有所不同。} 比如，用户在睡觉前可能倾向于收听舒缓的歌曲而在运动时倾向于收听快节奏歌曲。

要想准确地为用户做出符合其即时需求的推荐，推荐算法需要充分考虑音乐推荐的上述特点。 {59%：概括起来，要考察如下两个问题：}

1.使用合理的建模方法全面考察歌曲的划分标准不确定性和隶属类别多样性。

2.分析用户行为与其听歌次序及所处上下文环境的关系并挖掘用户的即时偏好。

由于书籍、电影、视频等物品消费代价比较高（主要是消费时间比较长），用户很难在短期内产生足够形成序列的行为， 因此针对传统物品的推荐算法[3][20][39]大多是直接对用户所处的群体环境以及行为偏好进行分析，{48%：很少考察用户行为序列的作用。} 因为没有充分考察音乐推荐的次序相关性和上下文相关性等特点，传统推荐算法做出的往往都是一般意义的推荐， {42%：这些推荐不符合用户对音乐作品的即时需求，降低了用户对推荐结果的接受程度。}

{56%：目前，研究人员给出了一些有效的音乐推荐算法。} 按照这些算法参考用户行为的程度，我们将这

些算法分为全面考察用户所有收听行为的基于用户长期行为的音乐推荐[42][10]、仅考察用户当前收听行为的基于用户即时行为的音乐推荐[14][22]以及 考察用户在当前会话期内收听行为的基于用户中期行为的音乐推荐[18][33]三类。 其中，基于用户中期行为的音乐推荐由于考虑了上下文环境对用户行为的影响而得到了越来越多的关注。 但是，以[18][33]为代表的基于用户中期行为的音乐推荐算法还存在对用户行为的时间相关性分析不合理的问题，主要表现在如下几个方面：

1.只能够定性地分析出目标歌曲可能隶属的类别，无法定量地给出隶属的程度，对用户行为的描述不够细致。

2.只考虑了若干显着类别的影响，忽视了其他类别对用户行为的贡献和作用，对用户行为的描述不够全面。

3.在一定程度上依赖于其他用户的行为，但很多时候其他用户的行为很难获取，导致对当前用户行为的预测效果比较差。

为了合理地利用用户中期行为提高音乐推荐效果，本文给出了一种基于多维时间序列分析的个性化音乐推荐方法， 该方法将用户行为建模为一个多维时间序列，从而实现对用户行为细致、全面地分析。 {43%：同时，该方法专注于对当前用户行为的分析而降低了对其他用户行为的依赖。}

此外，本文认为用户的未来行为与其长期行为、中期行为和即时行为均存在相关性，但是目前未见工作将这三方面的影响和作用综合起来进行考虑。 因此，本文进一步给出一种基于用户长期、中期和即时行为的综合音乐推荐方法，全面考察用户行为的时间相关性。 本文所给方法能够比较全面地考察音乐推荐的特点，既充分考察了歌曲划分标准的不确定性和 隶属类别多样性又考察了用户的听歌行为与歌曲次序及其所处上下文环境之间的关系。 通过实验，本文验证了上述方法能够取得比参考方法更高的预测命中率以及更低的推荐误差。 最后，本文基于上述方法实现了一个原型系统并且通过分布式实时计算框架Storm在该系统中实现了本文方法的并行化，初步验证了所给方法的可行性。

本文结构按照如下方式进行组织： 第2章介绍与本文工作相关的一些工作，包括常用的音乐推荐算法、物品相似度计算方法和算法评测指标以及文本处理、时间序列分析的相关理论等； 第3章描述了本文所提的一种基于多维时间序列分析的个性化音乐推荐方法，包括问题的定义、方法的执行流程等； 第4章描述了本文给出的一种基于用户长期、中期和即时行为的综合音乐推荐方法； {44%：第5章介绍了本文所实现的原型系统及相关实现技术；} {61%：第6章对本文工作进行总结并给出对未来工作的展望。}

第二章 相关工作

本章我们介绍与本文相关的一些工作。 {42%：首先，我们介绍一些常用的音乐推荐算法、相似度计算方法以及算法评估标准。} 然后，我们介绍文本建模、时间序列分析以及实时流处理框架的相关工作，这些方法和技术将在本文后续工作中得到应用。

2.1音乐推荐算法

目前，音乐推荐系统领域的研究工作已经取得了长足进步，出现了很多音乐推荐算法。 {45%：要想为用户推荐合适的歌曲，大部分音乐推荐方法都要对用户的听歌行为进行分析。} 按照推荐算法对用户行为的参照程度，我们将这些音乐推荐方法分为基于用户即时行为的音乐推荐、基于用户长期行为的音乐推荐和基于用户中期行为的音乐推荐。 {49%：本节对这三类推荐方法进行介绍。}

2.1.1基于用户即时行为的的的音乐推荐

基于用户即时行为的音乐推荐是最为朴素的一种音乐推荐方法，此类音乐推荐算法认为用户的状态在短期内保持稳定，而用户可能收听的下一首歌曲与用户当前收听的歌曲具有类似的特征。假如用户正在收听老狼的《同桌的你》，此类算法首先通过一定的方式抽取歌曲对应的特征，{40%：并在此基础上将与《同桌的你》具有相似特征的歌曲推荐给用户，比如老狼的另一首经典校园民谣《睡在我上铺的兄弟》，}豆瓣电台的“听相似歌曲”实现的正是这样一个功能。因此，以何种方式抽取歌曲特征就成了此类音乐推荐算法工作的关键。文献[14]充分利用歌曲的编辑属性，将与当前歌曲具有相同或相似曲作者、歌曲名称以及歌词的歌曲推荐给用户。文献[7][9][30]使用歌曲的声学特征为用户做出推荐，将与当前歌曲具有相似节奏、韵律、音色的歌曲推荐给用户。文献[24]使用心情特征对歌曲进行刻画并将与当前歌曲具有近似心情属性的歌曲推荐给用户。这些工作都是从单一方面对歌曲进行刻画并进而为用做出推荐，但这种推荐往往是片面的，比如文献[7][9]对于对声学特征不敏感但对于情感特征敏感的用户就不适用。文献[22]对歌曲对应的文本文档进行分析，使用语义特征对歌曲进行刻画，取得了不错的效果。

基于用户即时行为的音乐推荐作为一类比较朴素的推荐算法尽管能够取得不错的推荐效果，但其问题也是显而易见的，即对用户行为的刻画过于简单。

2.1.2基于用户长期行为的音推荐

与基于用户即时行为的音乐推荐不同，基于用户长期行为的音乐推荐对用户收听过的所有歌曲都进行了考察和分析，{54%：常见的基于协同过滤的音乐推荐和基于全局特征的音乐推荐都属于此类。}

{47%：协同过滤推荐(Collabrative Filtering，CF)是当前最为流行的一类推荐算法，其挖掘用户所处的社会环境，利用群体智能为用户做出推荐。}协同过滤推荐基于这样一个假设，即如果两个用户在过去有相同的行为，那么系统认为他们在未来也会有类似的行为[46][28][42]。当需要预测一个用户是否喜欢一个物品时，协同过滤推荐算法首先找到与当前用户喜好类似的用户，进而综合这些相似用户的喜好为用户推荐新的物品。{69%：假如用户A收听了《同桌的你》、《睡在我上铺的兄弟》、《朋友》，}用户B收听了《同桌的你》、《朋友》、《白桦林》，用户C收听了《一无所有》、《北京北京》、《太阳》，那么此类算法认为用户A和用户B属于相似用户，进而将用户B收听但用户A没有收听的歌曲《白桦林》推荐给A。

基于全局特征的音乐推荐是基于用户长期行为推荐算法中比较朴素的一类推荐方法，此类算法认为用户的长期行为能够反映用户对音乐的偏好。因此，在抽取歌曲特征的基础上，此类算法将用户收听的所有歌曲的平均特征作为用户特征并推荐与此平均特征类似的歌曲给用户。文献[10]使用声学特征对歌曲进行刻画，然后计算用户收听的所有歌曲的平均声学特征并以此作为用户特征。

由于基于用户长期行为的音乐推荐对用户的分析比较全面，因此此类算法往往能够得到比较不错的推荐效果。但是，此类算法对用户所处上下文环境的影响考虑不足，无法满足用户的即时需求。

2.1.3基于用户中期行为的音乐推荐

基于用户中期行为的音乐推荐是这样一类推荐算法，它们认为用户在当前会话期内所听歌曲构成的序列能够在一定程度上反映用户所处的上下文环境。因此，它们通过对该歌曲序列的分析来完成对用户行为的预测和歌曲的推荐。文献[38]给出了“当前会话期”的定义，即指用户正在收听歌曲的这一段连续时间。文献[38]使用最近

收听的几首歌曲去匹配用户收听历史，进而找到类似收听行为并据此为用户作出推荐。 尽管能够取得一定的效果，但用户所收听的歌曲数目不一，对于历史行为少的用户，文献 [38]便无能为力。 此外，文献 [38]没有对歌曲空间进行降维，而是直接使用歌曲本身去匹配，这降低了算法的可行性。 文献[18][33]是基于用户中期行为音乐推荐的两种典型代表方法，这两种方法首先 使用文本分析的方法将歌曲表示成由若干隐含主题构成的概率分布并使用若干显着主题表征歌曲。 这样，它们便将用户在当前会话期内的收听行为建模为主题序列，通过对该主题序列的分析即可预测下一首歌曲可能隶属的主题， {42％：进一步地它们将相关主题中的显着歌曲推荐给用户。} 其中，文献 [18]是使用以PrefixSpan为代表的模式挖掘算法对主题序列进行挖掘的，而文献 [33]使用马尔可夫模型对主题序列进行分析。 文献 [18][33]能够取得较 [38]更好的推荐效果，但是他们只是定性地预测下一首歌曲可能隶属的隐含主题，无法定量的给出具体的隶属程度。 此外，它们只考察了若干显着主题的作用而忽视了其他隐含主题的影响。 {48％：再者，它们需要依赖其他用户的行为来为当前用户做出推荐。}

为了解决文献[18][33]中存在的问题以较合理地利用用户中期行为提升推荐效果， 本文给出一种基于多维时间序列分析的音乐推荐方法，使用主题模型将歌曲表示成由若干隐含主题构成的概率分布， 并在此基础上将用户在当前会话期的行为表示成多维时间序列。 通过对该多维时间序列的分析，该方法预测用户可能收听的下一首歌曲的特征并从曲库中选择类似的歌曲推荐给用户。 此外，用户的未来行为除了受到用户中期行为的作用之外还受到用户即时行为和长期行为的影响，但目前未见工作将这三者的影响综合起来考虑。 为此，进一步给出一种基于用户长期、中期和即时行为的综合推荐方法，全面考察用户行为的时间相关性。

2.2相似度度量

由前文介绍可知，几乎所有的音乐推荐方法都会涉及到相似度的计算，本节将简单介绍几种计算相似度的方法。

2.2.1余弦相似度

{47％：余弦相似度是推荐系统中计算用户与用户或者物品与物品之间相似度的一种常用方法，其通过测量两个向量内积空间夹角的余弦值来度量它们之间的相似性。} {86％：0度角对应的余弦值是1，而其他任何角度对应的余弦值都不大于1，并且其最小值是-1。} {53％：如果两个向量的指向越接近，那么它们内积空间夹角的余弦值越接近于1，即二者越相似。} {42％：相反，如果两个向量的指向越相离，那么它们内积空间夹角的余弦值越接近于-1，即二者越不相似。} 设向量X.a和向量X.

{48％：b分别表示用户 a和用户 b的偏好向量，其中 X. a(i)表示用户 a对编号为 i的物品的偏好值或评分，} {49％：X. b(i)表示用户 b对编号为 i的物品的偏好值或评分。} {43％：那么，用户a和用户b的 相似度可以按照公式2.1进行计算，其中K表示物品集合中物品的数目。} {52％：物品之间的余弦相似度可以按照类似的方式进行计算。}

2.2.2 KL距离

{74％：KL距离也被称为相对熵或 KL散度，是两个概率分布 P和 Q差别的非对称性的度量，} 用来度量使用基于 Q的编码来编码来自 P的样本平均所需的额外的比特个数[25]。 {42％：如果能够把物品表示成一个概率分布，那么显然可以用KL距离来衡量两个物品之间的相似度。} 设P

=(p1，...， pi，...， pK)(1≤ i≤ K)表示物品 p对应的概率分布， 物品 q对应的概率分布用 Q=(q1，...，qi，...， {53％：qK)表示，其中 K表示随机变量的取值数目。} 那么，物品p和q之间的KL距离可以按照公式

2.2进行计算，其取值非负。 需要注意的是， KL散度仅当概率分布 P和 Q各自总和均为1，且对于任何 i皆满足 pi)0及 qi)0时才有定义， {62%：若式中出现0 ln0的情况，其值按0处理。}

{55%：而物品之间的相似度应该满足对称性，即}

因此，不能直接使用 KL距离来计算物品之间的相似度，但可以使用 p到 q的 KL距离与 q到 p的 KL距离的平均值来作为二者的最终距离， {42%：这样就可以满足对称性的要求，如公式2.5所示。} {44%：进一步地，可以使用如公式2.6所示的方法计算二者之间的相似度。}

2.2.3 Hellinger距离

{67%：Hellinger距离也是是一种度量两个概率分布之间相似度的方法。} 与KL距离不同的是，其然满足对称性，因此可以直接用来计算物品之间的相似度 [36]。 物品p和物品q之间的Hellinger距离可以按照公式2.7计算，进而可按公式2.8可以计算二者之间的相似度。

本文所述音乐推荐方法将使用主题模型建模的方法将每一首歌曲表征为由若干隐含主题构成的一个概率分布，因此 KL距离和 Hellinger距离将被本文用来计算两首歌曲的相似度。

2.3评测指标

{42%：推荐系统所研究的问题主要包括评分预测问题、Top-N推荐问题、冷启动问题、可解释性问题以及用户交互问题等 [2]。} {41%：其中，评分预测问题和Top-N推荐问题是得到最广泛研究且最为重要的内容。} 所谓评分预测问题就是根据用户已经产生的评分记录来预测其对尚未评分物品的可能打分，而Top-N推荐是指为用户生成一个包含N个符合其偏好的物品列表。 围绕着这两类问题，研究人员给出了众多推荐算法，本文将简单介绍评测推荐算法优劣的一些指标。

2.3.1用户满意度

{60%：用户作为推荐系统的重要参与者也是推荐系统最终的服务对象，其满意度是评测一个推荐系统优劣的最重要指标。} {74%：一般来说，用户满意度可以通过对一些用户行为的统计得到。} {44%：比如，在电子商务网站中可以通过用户的实际购买情况来评判，或者通过设置“满意”/“不满意”按钮进行显式地统计。} {87%：更一般的情况是，可以使用点击率、用户停留时间和转化率等指标度量用户的满意度。} 对于本文所研究的音乐推荐系统，可以统计用户在系统上的“喜欢”/“不喜欢”或者“收听”/“放弃”等行为来评估不同音乐推荐算法的推荐效果。 {43%：然而，用户满意度作为真实场景中的评测指标往往只能通过用户调查或者在线实验的方法获得而无法实现离线计算，这增加了研究人员评估算法优劣的难度。} 因此，在实际的研究工作中很少通过这个指标来评估不同推荐算法的优劣。

2.3.2预测准确度

{88%：预测准确度是度量一个推荐系统或者推荐算法预测用户行为能力的重要指标。} {86%：从推荐系统诞生的那一天起，几乎99%与推荐系统相关的论文都在讨论这个指标，这主要是因为该项指标可以通过离线实验计算，方便了研究人员研究推荐算法 [1]。} {49%：对于评分预测问题，一般使用均方根误差(Root Mean Square Error， RMSE)和平均绝对误差(Mean Absolute Error， MAE)来表征算法的预测准确度 [19]。} RMSE可由公式2.9计算得到。

{56%：其中，T代表测试集，u和i表示测试集中的用户和物品，rui是用户u对物品i的实际评分，而r^ui是推荐算法给出的预测评分。}

{59%：MAE采用绝对值计算预测误差，如公式2.10所示。}

{55%：预测误差越小，预测准确度越高，推荐算法的效果越好。}

{74%：Top-N推荐一般通过准确率(precision)和召回率(recall)来度量算法的优劣。} {90%：设R(u)是根据用户在训练集上的行为给用户作出的推荐列表，而T (u)是用户在测试集上的行为列表 [40]。} {46%：那么，推荐结果的准确率和召回率如公式2.11和2.12所示。} {46%：命中率和准确率越大，预测准确度越高，推荐算法的效果越好。}

2.3.3其他评测指标

除了预测准确度这一重要指标之外，推荐系统的评测还有诸如覆盖率、多样性等指标，这些指标从不同的角度看待推荐的有效性。 {79%：其中，覆盖率(Coverage)描述的是一个推荐系统对物品长尾的发掘能力，可以简单地定义为推荐系统能够推荐出来的物品占总物品集合的比例。} {42%：覆盖率越大，说明系统所能推荐的物品越广泛，也说明系统挖掘物品长尾的能力越强。} 多样

{49%：性(Diversity)用来描述推荐列表中物品两两之间的不相似性，列表中物品两两之间的相似性越小表示推荐的多样性越大。}

考虑到用户满意度无法进行离线实验而覆盖率、多样性等对推荐算法的表征能力不佳，本文主要使用预测准确度来评估所给出的音乐推荐方法。

2.4分布式实时计算系统

尽管目前单机的处理能力已经得到了极大提升，但其在应对大数据时代产生的海量数据时仍然非常吃力。 {41%：为了解决大数据时代海量数据的处理和分析的问题，Google提出了一种分布式计算模型，即 MapReduce，} 该模型使得由一般能力机器组成的集群可以完成大规模或者超大规模的计算工作。 在Google工作的启发下，Apache于2005年开发了分布式应计算框架Hadoop[16]。 Hadoop对于批处理的工作以及离线的海量数据分析比较有优势，但其在面对一些实时性要求比较高的计算任务时，处理能力略显欠缺。 为了弥补这一缺憾，以Storm[50]为代表的分布式实时计算系统被开发了出来，这些系统及框架在实时数据流分析方面能够取得比Hadoop更好的效率和效果。 其中，Storm是由Twitter开发的一款开源的分布式实时计算框架，其适用于流数据处理和分布式远程过程调用两种场景。 {90%：对于流数据处理场景，Storm可以用来处理源源不断流进来的消息，处理之后将结果写入到某个存储中去。} 此外，Storm的处理组件是分布式的且处理延迟极低，使得其在分布式远程过程调用的场景中也能够得到比较充分的应用。 本文所要解决的问题恰恰是一个实时数据流的分析问题，因此本文后面将会选用 Storm进行数据的分析和处理， 本节对其基本组成及其在分布式远程过程调用中的应用进行介绍。

2.4.1基本组成

{63%：一个Storm集群往往是由一个主控节点(Master Node)和多个工作节点(Work Nodes)组成。} {81%：其

中，主控节点上运行着一个名为“ Nimbus ”的守护进程，用于分配代码、布置任务及故障检测， } {92 %： 而每个工作节点都运行一个名为“ Supervisor ”的守护进程，用于监听工作、开始及终止工作进程。 } {99 %： Nimbus和Supervisor都能快速失败，而且是无状态的，这样一来它们就变得十分健壮，而两者的协调工作是由Apache ZooKeeper来完成的。 }

在Storm中，一个实时应用的计算任务被打包成一个Topology任务发布，且Topology任务一旦提交后永远不会结束，除非用户显式地去停止任务。 计算任务Topology是由多个Spout和Bolt计算组件构成，这些计算组件是通过数据流连接起来的。 {43 %： 其中，Spout是Storm中的消息源，用于生产消息，Bolt是Storm中的消息处理者，用于消息的处理。 } Topology中每一个计算组件都有一个并行 执行度，在创建Topology时可以进行指定，Storm会在集群内分配对应并行度个数的线程来同时执行这一组件。 图2.1是Twitter Storm官方给出的一个典

型 Topology示意图，其中水龙头表示用以生产数据的 Spout组件，闪电表示用户处理数据 Bolt组件， {43 %： 消息或数据由 Spout组件产生后便在不同的 Bolt组件中进行流动并被处理。 } Storm的编程非常简单，用户只需要在 Spout组件中实现数据读取及分割的逻辑，在 Bolt组件中实现数据的处理逻辑， 同时在各组件中指定数据流动方式即可轻松地完成并发布一个实时计算的任务。

图 2.1: 典型Twitter Storm示意图

2.4.2分布式远程过程调用

分布式远程过程调用（ Distributed Remote Procedure Call， DRPC）是联系客户端与 Storm集群的一种机制，Storm中引入这一机制的主要目的是利用 Storm的实时计算能力来并行化 CPU密集型计算。 {100 %： Storm集群上运行的拓扑接收调用函数的参数信息作为输入流，并将计算结果作为输出流发射出去。 } 其中，DRPC通过是 DRPC Server实现的，其整体工作过程如下：

1.接收到一个RPC调用请求；

2.发送请求到Storm上的拓扑；

3.从Storm上接收计算结果；

{83 %： 4.将计算结果返回给客户端。 }

图2.2更为细致地描述了DRPC的工作流程，大致可以分为如下五个步骤：

1. {98 %： 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DPRC Server接收到函数调用流；

3. {100 %： DRPC Server会为每次函数调用生成唯一的id； }

4. {100 %： 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. {100%： 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。}

图 2.2: DRPC工作流程示意图

2.5文本建模

为了对歌曲进行全面地刻画，本文将使用文本分析的方法对歌曲进行建模，本节介绍一些常用的文本建模方法。

2.5.1向量空间模型

计算机不具备人脑的结构，无法理解自然语言，所以需要首先将无结构的自然语言文本转化为计算机可计算的特征文本。 为此，Salton等人在20世纪70年代提出了向量空间模型(Vector Space Model， VSM)[45]。 向量空间模型首先将每一个文档看做一个词袋(Bag ofWords)，即认为一篇文档是由一组词构成的一个集合且词与词之间没有顺序以及先后的关系。 {56%：其次，向量空间模型将文档表示成一个向量，向量的每一维表示一个词项，而每一维的取值表示该词项在文档中的权重。} {49%：对于文档集合 D中编号为 j的文档 dj，可以将之表示成一个 t维的向}

量d.j =(ω1; j， ...， ωt; j)，其中 t表示词项的数目， ωi; j(1 ≤ i ≤ t)表示第 i个词项在文

档dj中的权重。 在对文本进行建模的过程中，词的选取及权重的计算有以下几种典型方式：

1.布尔模型。 {42%：这是最为简单直观的一种计算词项权重的方法，其将词项在文档中是否出现作为其权重，如果词项在文档中出现那么将其权重记为 1，否则记为 0。} {54%：虽然这种方法比较简单，但是它没有体现词语在文档中出现的频率。} 一般来讲，词语在文档中出现的越多，说明它对该篇文档的重要性越大（“的”、“得”、“地”、“是”等停用词除外）。

2.词频模型。 {46%：与布尔模型不同的是，词频 (Term Frequency， TF)模型统计词项在文档中出现的次数，然后得到词项的频率，并将之作为词项的权重。} {44%：词项ti相对于文档dj的词频如式2.13所示。}

这里，ni; {77%：j表示该词项在该文档中出现的次数。} 词频模型突出了词频对词项重要性的影响，能够取得比布尔模型较好的效果。 {84%：但是，词语的重要性不仅随着它在文档中出现的次数成正比增加，而且可能会随着它在语料库中出现的频率成反比下降。}

词频-逆文本频率模型。

词频-逆文本频率(Term Frequency- Inverse Document Frequency， TF- IDF)是对 TF模型的补充， {50%：其认为词项的重要性随着其在特定文档中出现次数的增加而增强，但同时随着其在全体文本中出现次数的增加而减弱，} {82%：即该模型认为对区别文档最有意义的词语应该是那些在文档中出现频率高、而在整个语料库中的其他文档中出现频率少的词语。} {45%：词项ti相对于文档dj的TF-IDF取值如式2.14所示。}

{42%：idfi为逆向文本频率，定义如式 2.15所示。}

{59%：其中，|D|表示文档集合中的文档总数，|{j:}｛57%：ti∈dj}|表示文档集合中包含词项ti的文档数目。} TF-IDF结构简单，容易理解，被广泛应用。 但是，其无法准确捕捉文档内部与文档间的统计特征，也不能解决同义词和多义词的问题，因此精确度不是很高。

{100%：2.5.2隐含狄利克雷分配模型}

{47%：为了解决同义词和多义词的问题，Blei等人于2003年提出了隐含狄利克雷分配模型(Latent Dirichlet Allocation，LDA)[6][21]。} LDA也是一种典型的词袋模型，其认为一篇文档由多个主题构成，而文档中每一个词都是由对应的主题生成而来。 {98%：其中，主题表示一个概念、一个方面，表现为一系列相关的单词，是这些单词的条件概率。} {98%：形象来说，主题就是一个桶，里面装了出现概率较高的单词而这些单词与这个主题有很强的相关性。} {42%：这样，LDA模型便通过隐含主题将文本与词项联系起来，从而达到降维的目的。} {42%：LDA是一种生成模型，一篇文档按照如下所示的规则生成：}

1.假设有两种类型的桶，一种是文档-主题桶，桶里的每一个球代表一个主题； 另一种桶是主题-词汇桶，桶中的每一个球代表一个词汇。

2.文档的生成过程就是不断从桶中取球的过程，每一次先从文档-主题桶中取出球，得到该球代表的主题编号z。

3.从编号为z的主题-词汇桶中取球，得到一个词汇。

4.不断重复2，3两步，即可生成一篇文档。

{43%：在LDA模型中，记文档-主题的概率分布为多项式分布.θ，主题-词汇的概率分布为多项式分布φ.。} LDA模型认为.θ和φ.是模型中的参数，而考虑到参数都是多项式分布，模型选择狄利克雷分布作为其先验分布。在确定了这些分布之后，LDA下面需要做的就是估计这些分布的参数，如算法1所示的吉布斯采样(Gibbs Sampling)是目前比较流行的估计参数的方法。

在本文后续实验中，我们将分别以TF-IDF为代表的向量空间模型和以LDA为代表的主题模型对歌曲对应的文档进行建模， 发现LDA能够获得较高的推荐准确率，因此我们将LDA作为我们主要的文本建模方法。

2.6时间序列预测

歌曲时间短、消费代价低的特点决定了其能够较容易地形成序列，且这种序列是有严格的时间顺序的。 本文将通过对用户在当前会话期内所听歌曲形成的时间序列的分析来预测用户接下来的行为。 {48%：本节将简单介绍一些常用的时间序列预测方法。}

Algorithm 1

LDA模型的Gibbs采样算法

1: {44%：首先对所有文档中的所有词遍历一遍，为其都随机分配一个主题，即zm;} {80%：n=k～Mult(1/K)，其中m表示第m篇文档，n表示文档中的第n个词，} k表示主题，K表示主题的总数，之后将nkm、nm、ntk、nk都加1， {45%：它们分别表示在第m篇文档中主题k出现的次数、第m篇文档中主题数量的和、主

题k对应的词t的次数，｝ k主题对应的总词数。

2: ｛80％：对第1篇文档中的所有词进行遍历，假如当前文档中的词t对应主题为k，｝ 则将nkm、 nm、 ntk、 nk都减1，即先拿出当前词，之后根据式2.16取样出新的主题， 再将nkm、 nm、 ntk、 nk都加1。 其中，α和β为对应的Dirichlet分布的参数，V为词汇总数。

3: 重复步骤2直至遍历所有文档。

4: 输出LDA模型中的参数.θ和φ.。

2.6.1简单平均法

｛65％：简单平均法是以观察期内时间序列的各期数据（观察变量）的平均数作为下期预测值，｝｛60％：按照采用的平均方法又可以分为算术平均法、加权平均法和几何平均法三类。｝｛45％：其中，算术平均法以观察变量的算术平均数作为下期预测值，加权平均法以观察变量的加权算术平均数作为下期的预测值，｝｛58％：而几何平均法是以观察变量的几何平均数作为下期的预测值。｝｛45％：简单平均法比较简单、直观，但其预测误差一般偏高。｝

2.6.2指数平滑法

｛100％：指数平滑法是由移动平均法改进而来的，是一种特殊的加权移动平均法。｝｛96％：这种方法既有移动平均法的长处，又可以减少历史数据的数量。｝ 一方面，它把过去的数据全部加以利用。 ｛86％：另一方面，它利用平滑系数加以区分，使得近期数据比远期数据对预测值影响更大。｝｛91％：它特别适合用于观察值有有长期趋势和季节变动，必须经常预测的情况。｝｛79％：按照平滑的次数可以可分为一次指数平滑法和多次指数平滑法。｝｛81％：其中，一次平滑法是计算时间序列的一次指数平滑值，以当前观察期的一次指数平滑值为基础，确定下期预测值。｝｛59％：与二次移动平均法类似，二次指数平滑法就是对时间序列的一次指数平滑值再次进行指数平滑。｝

｛60％：2.6.3差分整合移动平均自回归模型｝

差分整合移动平均自回归模型(Autoregressive Integrated Moving Average model， ARIMA)又称为Box- Jenkins方法，是由Box和Jenkins于1970年提出的一种时间序列预测方法， 目前已经得到广泛的应用[31]。

｛66％：在模型ARIMA(p，d，q)中，“AR”表示自回归模型，p为自回归阶数；｝｛59％：“MA”表示滑动平均模型，q表示滑动平均的阶数；｝｛55％：d表示将时间序列转化为平稳时间序列所作的差分次数，即差分阶数。｝｛45％：ARIMA首先需要进行d次差分从而将非平稳序列转化为平稳序列，然后利用自回归模型和滑动平均模型对转换后的平稳序列进行预测。｝｛44％：进一步地，模型ARIMA(p，d，q)可以表示成如下三个式子。｝

｛42％：其中，yt为时间序列Y的第t个取值，B是滞后算子，θ和φ为模型参数。｝

ARIMA的执行流程主要分为模型识别(Model Identification)、参数估计(Parameter Estimation)和诊断检测(Diagnostic Checking)三个阶段。 ｛43％：其中，模型识别阶段主要完成检测序列是否平稳的工作。｝｛50％：如果序列不平稳，那么需要通过差分的方法将序列转化为平稳序列并给出差分阶数d。｝｛46％：在此基础上，识别出序列适用的可能模型，如自回归模型或滑动平均模型或者二者的混合。｝｛42％：而参数估计阶段主要完成模

型参数的估计工作，即通过最小二乘法估计参数θ和φ。} 诊断检测阶段用以检测所给出的模型及参数是否符合条件，如果符合则选用此模型进行预测，否则重新识别模型。

ARIMA模型已经在计量经济学中得到了非常广泛地应用而且也取得了比较理想的预测效果， 因此本文后续的工作中将使用该模型完成对用户行为序列的分析和预测工作。

{51%：第三章 基于多维时间序列分析的音乐推荐方法}

本章将给出一种基于多维时间序列分析的音乐推荐方法，该方法实现了对用户听歌行为定量且全面地分析和预测， {45%：同时减少了对其他用户行为的依赖，使得推荐的结果更加符合用户即时偏好。} 首先，该方法使用用户给歌曲所打的标签构造歌曲对应的文档，进而通过主题模型的方法对文档集合进行建模并 最终把每一首歌曲都表示成由若干隐含主题构成的一个概率分布。 在此基础上，该方法将用户在当前会话期内的听歌行为建模为一个多维时间序列， 并通过对该多维时间序列的分析来预测用户行为，而这样做的原因是该方法认为用户在当前会话期内所收听的歌曲序列能够在一定程度上表征其所处的上下文环境。

{57%：我们将在本章中对该方法进行详细地阐述。} {44%：首先，我们将介绍本文工作所研究问题的相关描述和定义。} 然后，我们给出该方法的工作流程，包括隐含主题的抽取、多维时间序列的构造及分析预测、最终推荐结果的生成等内容。 {43%：最后，我们介绍为验证该方法有效性而设计的实验及实验结果。}

3.1问题描述

音乐推荐的目的在于通过对音乐本身属性和用户行为习惯的分析，帮助用户过滤掉不必要的信息，并最终为用户推荐符合其喜好的音乐作品。 换句话来说，就是解决如何在已知歌曲特征以及用户之前所收听歌曲的情况下准确地预测用户可能收听的下一首歌曲的问题 [38]，如图3.1所示。

图 3.1: 下一首歌预测问题的说明图

为了更好地描述本文所要研究的问题，我们首先定义用户集和歌曲集，如下所示：

用户集U： 所有用户的集合，如式3.1所示。 其中， v表示用户的数目，即v = |U|。

歌曲集 S： 所有可以推荐给用户的物品 (这里就是歌曲，本文不加区分地使用“歌曲”和“物品”)的集合，如式 3.2所示。 {46%：其中， m表示曲库中歌曲的数目，即m = |S|。} 为方便起见，我们认为用户收听的歌曲一定在歌曲集合中。

{40%：对于用户集 U中给定的一个用户 u，本文所要研究的音乐推荐系统的目标就是为该用户推荐其可能喜欢的下一首歌曲。} 为了衡量用户对歌曲的喜欢程度，我们定义如下所示的效用函数：

效用函数utility(u， s): 表征歌曲 s对用户 u的推荐度 (如歌曲 s符合用户 u喜好的程度、歌曲 s与系统所预测歌曲的相似度等)。 {45%：效用函数反映了用户对某首歌曲的喜爱程度，其值越大表明喜欢程度越大。}

如前所述，本文所给的方法是建立在隐含主题分类和用户在当前会话期内听歌序列的基础上， 因此我们进一步地定义歌曲对应的隐含主题集合以及用户所收听歌曲对应的序列。

主题集 T: {44%：由所有隐含主题组成的集合，如式式 3.3所示，其中 K为隐含主题的数目。}

事件 e(u, τ, s): 表示用户 u在时刻 τ收听了歌曲 s。 显然，歌曲 s可由用户 u和时刻 τ唯一决定，因此 e(u, τ, s)可简化为e(u, τ)。

序列 Q(u): 用户 u在给定平台下的所有听歌事件按照时间顺序排列开来得到的序列，如式 3.4所示。

其中， π+n为用户 u所收听或喜欢的歌曲数目， τ为事件发生的时间，且τ1(τ2(

...(τ.(τ

+1(...(τ

+n，即用户收听这 π+n首歌曲是按照一定的次序一首接一首收听的。 另外， Q(u)中最后 n个事件是用户在当前会话期内产生的，即从事件 e(u, τ

+1)到事件 e(u, τ

+n)之间对应的 n首歌曲是用户当前连续收听的，而序列中的最后一个事件 e(u, τ

+n)是正在发生的。 也就是说，这 n个听歌事件两两之间没有明显的时间间隔，而事件 e(u, τ

+1)和事件 e(u, τ

{42%：)之间有明显的时间间隔，如式 3.5和3.6所示。}

其中， ε是给定的时间间隔，用以分割不同的听歌会话，在后文的实验中我们将之设为8分钟，而在系统实现中将之设为2小时。 为了方便，我们只考察在给定平台下确实有听歌行为的用户，即用户在给定平台下确实有听歌事件且其当前正在该平台下收听歌曲。 而对于在给定平台下没有任何听歌行为以及当前不在该平台下听歌的用户，为他们进行推荐比较困难，这是整个推荐系统领域重点研究的冷启动问题，本文对此不做深究。

目标歌曲: {47%：符合用户喜好或者用户接下来可能收听的歌曲。}

由之前的分析可知，本文所提方法的本质输入是歌曲对应的隐含主题T和用户u在当前会话期内的收听序列 Q(u)[π+1: π+n]，而最终目标是为用户推荐其最可能收听的下一首歌曲。 也就是说，在已知主题集T和用户 u当前会话期收听序列Q(u)[π+1: π+n]的情况下从歌曲集S中找出那些对用户的效用函数取值最大的目标歌曲 N(u)并推荐给用户，如式3.7所示。

为了解决预测用户可能收听的下一首歌曲的问题，有两个难点需要解决：

1.如何全面完整地对歌曲进行刻画。

{53%：2.如何对用户的行为序列进行分析以预测用户行为。}

3.2方法框架

针对上文所定义的音乐推荐问题，本文提出一种基于多维时间序列分析的音乐推荐方法，该方法按如图3.2所示的流程工作。 其中，虚线箭头表示离线处理模块，主要用来对歌曲进行刻画，即解决难点1； 实线箭头表示在线处理模块，主要用来对用户的行为序列进行分析进而预测用户未来的可能行为，即解决难点2。

图 3.2: {41%：基于多维时序分析的音乐推荐方法的工作流程示意图}

在离线模块中，我们首先定期从音乐网站上抓取以用户标签为主的歌曲文本信息，然后将这些标签信息按照一定的方式组合成文档， {43%：那么这些文档构成的文档集合将和歌曲集S一一对应。} 关于文档的构造方式，本文采用的方法是将被标记到给定歌曲的标签按照其被标记的次数在给定歌曲对应的文档中重复若干次。 假设给定歌曲s所具有的标签信息可以表示为一个标签集合，比如TagSet(s)={(“rock”，5)，(“pop”，3)，(“male”，2)}，那么其对应

的文档可以表示为Doc(s)= “rock rock rock rock rock pop pop pop male male” 其中，标签“rock”在Doc(s)中出现5次，标签“pop”在Doc(s)中出现3次，标签“male”在文档

Doc(s)中出现 2次。 由于主题模型将文档看做一个词袋模型，因此“rock”、“pop”、“male”在文档中出现的顺序不影响建模结果。 在获得歌曲集对应的文档集之后，我们对该文档集进行主题模型建模，从而抽取出歌曲中包含的隐含主题。 这样，每一首歌曲可以用一个主题权重向量表示，向量中的每一维取值表征对应主题对歌曲内容的贡献程度。

在在线处理模块中，我们首先获取用户在当前会话期内所听的歌曲，然后将这些歌曲按照时间先后顺序排列开来得到用户的当前会话序列。 结合离线模块的处理结果，我们将用户的当前会话序列建模为一个多维时间序列。 进一步地，我们使用一些经典有效的时间序列预测方法对该多维时间序列的每一个分量进行分析并预测单变量时间序列的下期取值。 然后，我们将这些单变量时间序列的预测值组合起来构成目标歌曲对应的主题权重向量。得到目标概率分布之后，我们计算曲库中所有歌曲与目标歌曲之间的相似度并按照相似度由大到小排列。 最后，我们选取排位靠前的N首歌曲作为推荐列表推荐给用户。 {41%：在下面的章节中，我们将对方法工作流程中的主要步骤进行一一介绍，包括如何进行主题模型建模、} 为何选择多维时间序列作为用户行为序列的模型以及相似度计算、最终推荐列表生成等。

3.3主题模型建模

如前所述，为了全面地刻画歌曲，本文在将歌曲映射成一个个文档的基础上通过主题模型建模来对歌曲进行描述。 {64%：隐含狄利克雷分配(Latent Dirichlet Allocation， LDA)模型[6]是当前最具代表性也是最流行的概率主题模型，} {40%：已经在文本挖掘、信息处理、多文档摘要等领域得到了广泛的应用[44][52]。} LDA模型能够将文档映射到由若干隐含主题构成的概率分布，而这个分布可以用一个主题权重向量表示，向量的每一维表征了对应主题对文档内容的贡献程

度。 通过LDA主题模型建模，我们不但可以抽象出文档中包含的隐含主题集合T，而且能够以量化的方式表达不同文档之间的距离和相似度。 {47%：本节详细介绍利用LDA主题模型对歌曲建模的过程。}

图 3.3: 歌曲Squares在Last.fm上的标签云

首先，我们从Lastfm(http: // last. fm)、豆瓣(www. douban. com)等音乐网站上抓取用户对歌曲所标注的标签，这些标签对歌曲内容的描述比较全面，既包含了歌曲的名称、曲作者信息、专辑信息、发行年代等基本信息， 还包含了歌曲所表达主题、歌曲类型、用户心情、适合场合等扩展信息。 以The Beta Band的Squares为例，用户为其标注的标签有表征年代的“ 2002 ”/“ 2000s ”、表征类型的“ indie rock ”/“ folk pop ”、表征用户感受的“ beautiful ”/“ want ”等，如图3.3所示。 在得到歌曲对应的标签信息后，我们将这些标签按照一定的方式进行组织，从而生成歌曲对应的文本文档。 为了减少噪音，我们只利用那些被多数人使用的标签来完成歌曲对应文本文档的构造，具体来说我们只考虑被标记次数大于10的标签。 这样，一首歌曲s都将对应一篇文档d，而歌曲集S将对应到一个文档集D。 {41％：最后，我们对文档集D进行LDA主题模型建模，从而得到包含K个用来全面刻画歌曲特征的隐含主题集合T。} 同时，对于歌曲集S中的任意歌曲s，我们能够得到其对应的隐含主题权重向量，如式3.8所示。

其中，ωi =0表示歌曲完全不属于该隐含主题代表的类别，即该隐含主题对歌曲的内容完全没有贡献； ωi =1表示歌曲完全属于该隐含主题代表的类别； 0 (ωi (1表示歌曲在一定程度上隶属于该隐含主题代表的类别。

表3.1展现了歌曲Squares的若干显着主题及其隶属于这些主题的概率。 由表3.1可以看出，歌曲Squares隶属于主题508的概率为0.215，这说明主题508对该歌曲贡献度为0.215。 类似的，主题313对歌曲的贡献度为0.186，主题231对该歌曲的贡献度为0.072。 由于主题508对文档的贡献度大于其他隐含主题，因此我们认为歌曲能够以更大的概率划分到主题508中。 如前所述，我们不去也无法准确地描述这些主题具体是什么含义，而我们正是利用这一点来回避确定性分类带来的弊端。

表 3.1: 歌曲Squares的显着主题及隶属度

主题id 508 313 296 106 231

隶属度 0.215 0.186 0.134 0.084 0.072

更形象地，我们使用雷达图来表示歌曲在各个隐含主题上的隶属程度。 假设隐含主题的数目为4且编号依次为#1、#2、#3、#4， 那么主题向量. s=(0.15，0.25，0.15，0.45)代表的歌曲的雷达图如图3.4所示。 其中，雷达图的每个顶点代表一个隐含主题，中心点到各个顶点的距离表示歌曲在该顶点对应隐含主题上的隶属度。

图 3.4: 歌曲的雷达图表示

3.4多维时间序列构造和预测

传统的基于“ 最相似 ”假设的推荐方法只考虑用户当前的收听习惯，较少考虑用户行为的序列性趋势，而本文所提出的基于多维时间序列分析的音乐推荐方法则对此进行了考虑。 在通过 LDA主题模型对歌曲进行建模而得到歌曲对应的主题权重向量后， 我们需要做的就是获取用户在当前会话期内收听的歌曲构成的序列并构造其对应的多维时间序列， {52％：然后使用相应的时间序列分析方法进行预测。} 对于上文中给出的一共收听了π+n首歌曲且在当前会话期内收听了 n首歌曲的用户 u来说， 为了预测其可能收听的下一首歌曲，我们需要获取该歌曲对应的主题权重向量，如式3.9所示。

{43％：其中， ωi(j)表示第 j首歌曲隶属于第 i个隐含主题的概率 (1 ≤ i ≤ K)。} 直接

得到这样一个主题向量往往是不太容易的，但估计出该向量对应的每一维的取值是可行的。 因此，要得到下

一首歌曲对应的主题向量，我们需要先估计出对应的主题权重向量中每一维的值， 然后将这些中间估计值组合起来即可构成完整的估计主题向量，从而能够在歌曲集 S中找到最匹配的歌曲推荐给用户。

{42％：由前文可知，用户 u的听歌行为是有时间顺序的。} 如果我们将用户在某一时间点所听歌曲对应的主题向量看做变量，那么将此变量在不同时间点的 取值按照时间顺序依次排开即可得到如式3.10所示的多维时间序列[17]。 进一步地，将上式在 K维隐含主题向量上展开，得到如式 3.11所示的更为直观的多维时间序列。 其中，i(1≤i≤K)表示隐含主题的编号，j(1≤j≤n)表示用户在当前会话期内所收听第j首歌曲的索引。 显然，该多维时间序列中任意一个时间点都对应着一个 K维主题向量且对于 K维主题向量中的每一维来说又对应着一个单变量的时间序列， 如式3.12所示。 其中，ωi(π+j)的含义如前所述。 通过对 TS(u，i)的分析可以得到用户在当前会话期内所收听的歌曲在编号为 i的隐含主题上的变化情况， 进而可以预测出用户可能收听的下一首歌曲在编号为 i的隐含主题上的隶属度， 即ω^i(π+n+1)。

进一步地，我们可以获得用户行为在其他隐含主题上的变化情况并估计出用户可能收听的下一首歌曲在其他隐含主题上的隶属度。 通过对K个估计值的组合，可以得到如式3.13所示的目标主题向量，即用户可能收听的下一首歌曲的估计主题向量。

3.5相似度计算

通过隐含狄利克雷主题建模我们得到了如式3.8所示歌曲集 S中歌曲对应的主题权重向量， {41％：考虑到向量中每一维的值表征歌曲隶属于某一隐含主题的概率，} {45％：我们将这些主题权重向量看作是离散的概率分布。} 进一步地，通过对多维时间序列的分析和预测，我们得到了如式3.13所示用户 u可能收听的下一首歌曲对应的主题权重向量的估计值， 即一个估计概率分布。 这样，我们便可以计算歌曲集中的歌曲s对应的概率分布与这个估计概率分布的距离。 显然，如果s与 目标歌曲s^距离足够小，那么我们就可以将歌曲s推荐给用户u。 因此，我们可以用s与目标歌曲s^的距离的倒数表示歌曲集中任一歌曲s对用户u的推荐度，即上文定义的效用函数 utility(u，s)，如式3.14所示。 因为我们是通过主题模型建模将歌曲表示成离散的概率分布，所以我们可以使用 KL距离(Kullback-Leibler Divergence)[25]以及Hellinger距离 [36]等来度量两首歌曲之间的距离。 {44％：考虑到KL距离不具有对称性，本文采用Hellinger距离来度量歌曲之间的距离，如式3.15所示。}

其中，dis(si，sj)为编号为i和j的歌曲对应的主题概率分布之间的Hellinger距离，K为隐含主题数目，ωik为编号为i的歌曲在第k个隐含主题上的隶属程度。 {41％：显然，当两首歌曲越相似，那么其对应的距离越小。} {42％：相反的，若两首歌曲越不相似，其主题概率分布对应的距离越大。}

3.6推荐列表生成

最后，我们根据效用函数计算歌曲集 S中所有歌曲的效用值，然后根据效用值对歌曲排序，并将排名最高的 N首推荐给用户， {53％：这样我们就为用户 u推荐了一个长度为 N的歌曲列表供用户选择。}

3.7实验设计和结果

本节我们将通过实验来验证本文所给的基于多维时间序列分析的音乐推荐方法的有效性，包括实验的设计思路、实验结果以及结果分析等内容。

3.7.1数据集收集

通过对基于多维时间序列分析的音乐推荐方法的分析，可以看出我们需要的数据集主要包括包含标签文本信息的 歌曲数据集以及用户在一定会话周期内所收听歌曲的数据集。 虽然Berenzweig等人于2003年从Art of the Mix(http: //www.artofthemix.org/)上抓取了播放列表据集 [5]，但是其存在如下三个问题：

{46％： 1.缺少歌曲对应的标签等文本信息。 }

2.歌曲名称经过处理，无法与Lastfm对应，导致可用数据较少。

3.给出的播放列表意义不清，没有时间信息，无法确定是用户在一个会话周期内的行为，可能跨越多个会话周期。

{41％： 为此，我们从Last.fm上重新爬取了一个数据集。 } 该数据集既包含歌曲的基本信息(包括标签等文本信息)也包含用户的基本信息(包括用户在一定会话周期内所收听的歌曲)。 为了减少噪音，我们只选用包含歌曲数目多于10首的列表，出现频次大于10的标签以及可用标签大于4的歌曲。 该数据集的统计信息如表3.2所示，目前该数据集已经发布在http: //lastfmseq.sinaapp.com/上，而其具体的使用说明如附录A所示。

表 3.2: 所用数据集信息

听歌事件 34930

歌曲总数 24992

用户总数 1530

歌手总数 5479

最大长度 10

最小长度 30

平均长度 22.83

3.7.2评测标准

对于用户 u，我们通过对其收听记录所形成的有序列表进行分析，为其生成一个包含 N首歌曲的推荐歌曲列表， 如果这 N首歌曲中包含用户真实收听的下一首歌曲，那么我们认为这个对于用户 u的推荐是有效的。 显然，类似音乐推荐这种为用户推荐一组物品供选择的问题是典型的Top-N推荐问题(Top-N Recommendation， TNR)。{44％： 由文献 [2][40]可知，召回率(recall)和准确率(precision)是衡量一个Top-N推荐算法优劣的重要标准，我们这里也用这两种标准来评测本文提出的方法。 } {89％： 记R(u)是根据用户在训练集上的行为给用户推荐的歌曲列表，而T (u)是用户在测试集上的行为列表，那么表征 } {93％： “ 检索出的相关文档数和文档库中所有的相关文档数的比率 ” } 的召回率recall的定义如式3.16所示。

可以看出，召回率表征用户真实收听的歌曲被推荐的数目与用户真实收听歌曲总数的比率。 因为在音乐推荐系统中，用户 u同一时刻在测试集上只会收听一首歌曲，即 |T (u)| =1。 {41％： 因此，我们将式 3.16简化为如式

3.17所示的命中率(hit ratio)。 }

其中， N为推荐系统为用户推荐的歌曲数目， hit表示用户实际收听的歌曲是否在推荐列表中，若在则为 1，否则为 0。 如果 hit为1，我们称之为“ 命中一次 ”。 记用户 u实际收听的歌曲为s，则 hit可表示为如式3.18。

准确率 precision表征了 {65 %：“ 检索出的相关文档数和系统所有检索到的文件总数的比率 ”} ，即用户真实收听的歌曲被推荐的数目与被推荐的歌曲总数的比率，其定义如式3.19所示：

考虑到|T (u)| =1，式 3.19可简化为式3.20。

考虑到召回率和准确率此消彼长的关系，文献 [21]中使用 F1-Score对模型进行评估，

{41 %： F1-Score的取值越大那么模型对应的综合效果越好，反之越差。} F1-Score可以用式3.21表示： 在如上所述的评测标准中，召回率和准确率的在很大程度上取决于如式3.17所示的命中率。 如果歌曲未被用户喜欢或收听，那么就认为该歌曲未命中。 然而，文献 [51]指出没有明显的证据表明未被评分的物品对用户来说是完全否定的。 也就是说，即使歌曲未命中，也不代表用户不喜欢该歌曲。 假设用户u真实收听的下一首歌曲为“ 同桌的你 ”，如果系统为其推荐了列表 (“ 北京北京 ”，“ 存在 ”，“ 同桌的你 ”)，那么我们认为该推荐是有效的，因为目标歌曲“ 同桌的你 ”在推荐列表中。 相反，如果系统为其推荐列表 (“ 白桦林 ”，“ 睡在我上铺的兄弟 ”，“ 一生有你 ”)，由于其中未包含目标歌曲“ 同桌的你 ”，我们 认为该推荐是无效的。 然而，第二个推荐列表中的歌曲与目标歌曲“ 同桌的你 ”都属于经典校园民谣，用户显然也会喜欢该列表，因此认为第二个推荐无效就不够合理； 第一个推荐列表中除了“ 同桌的你 ”之外其余两个都偏“ 摇滚 ”，这个列表对于喜好校园民谣的用户来说也并不一定有效。

为了解决这种矛盾，我们可以考虑使用推荐偏差的大小来衡量算法的优劣。 {45 %：也就是说，推荐偏差越小，算法越好，反之越差。} 这里的偏差可以用歌曲对应的主题概率分布之间的Hellinger距离来表示。 如果一个推荐列表中的歌曲与目标歌曲的整体相似度较高，那么该列表中的歌曲与目标歌曲的距离就应该较小， 即推荐偏差较小，这时即使目标歌曲不在该列表中，我们也应该认为该列表合理。 而如果一个推荐列表中的歌曲与目标歌曲的相似度整体偏低，导致推荐偏差较大，即使其中包含目标歌曲，我们也应该降低该列表被认可的权重。 {47 %：推荐系统主要包含评分预测和 Top- N推荐两类问题，在评分预测中我们常常使用均方根误差(RMSE)和平均绝对误差(MAE)来衡量算法的优劣， } 这里我们将其借鉴到音乐推荐的问题中并用以衡量不同算法的优劣，其定义如式3.22和3.23所示。

{44 %： 其中， U为测试用户集， |U|为用户数， e(u)为向用户推荐的结果的误差。} 考虑到我们为用户推荐的是一个列表，我们将e(u)看做是列表中所有歌曲与目标歌曲的平均距离。 如式3.24所示。

3.7.3实验设置

{47 %： 为了客观地衡量本文所述方法的效果，我们首先将所有数据随机地分为10份并将其中的9份作为训练集，剩余的1份为测试集，然后进行10轮交叉实验。} 最后，我们将10轮实验的结果进行平均，从而得到最终的实验结果。 {50 %： 其中，本文将隐含主题数目设为30。} 本文实验实在Dell Optiplex74的台式机上进行，操作系统为Ubuntu12.04，CPU为Intel酷睿2E6300，内存大小为2G，硬盘空间160G。 实验所用编程语言为Python2.7。

3.7.4结果分析图3.5展示了当被推荐歌曲数目N由1到100的增长过程中，不同推荐算法

的命中率的变化情况，{51%：这里包括基于用户的协同过滤算法(UserKNN)[42]、基于模式挖掘的推荐算法(PatternMining)} {41%：[18]、基于马尔科夫模型的推荐算法(1 st- Markov)[33]、基于全局特征的} {44%：推荐算法(Global)[10]、基于用户即时行为的音乐推荐(Local)[22]以及} 本文提出的基于多维时间序列分析的音乐推荐方法(Music Recommendation Based on MultidimensionalTime Series Analysis, MTSA)。 {64%：其中，横坐标表示被推荐歌曲的数目，纵坐标表示算法的命中率。} 由图3.5可以看出，代表本文所述方法的曲线与其他曲线能够明显分开且位于其他曲线之上， 表明本文所提方法能够获得比其他同类工作更好的召回率且提升效果比较明显。 此外，随着被推荐歌曲数目的增加，本文所述方法的召回率也同时提升且呈逐渐上升趋势。

图 3.5: {64%：不同推荐算法在命中率上的表现}

{49%：图3.6展示了随着推荐列表长度增加，不同推荐算法的推荐准确率变化情况。} {49%：从图中可以非常直观地看出，虽然随着推荐列表长度的增加，所给几种推荐算法的准确率都有所下降，} 但本文所给出的基于多维时间序列分析的音乐推荐算法能够取得最好的推荐准确率。 需要说明的是，随着推荐列表长度的增加，各算法推荐准确率取值比较低且持续下降是可以从准确率的定义中推导出来的。 假设用户数目为100，推荐列表长度为50，推荐算法的命中率为60%即命中数为60，那么此时的推荐准确度为0.012。 当推荐列表长度增加到100时，假设推荐算法的命中率提升到80%即命中数为80，那么此时的推荐准确率只有0.008。 简单来说，对于音乐推荐来说，准确率定义中的分子变化很小且取值也很小但分母却增长地比较快， {44%：所以导致了推荐准确率取值很小且随着推荐列表长度增加，准确率不断下降。} 因此，预测准确率偏低以及随着推荐列表增加而降低是可以解释的，而我们关注地不是其绝对取值，更大的意义上我们关注不同推荐算法获得的推荐准确率之间的相对效果。

图 3.6: {61%：不同推荐算法在准确率上的表现}

为了更为直观地考察本文所提方法在推荐准确率指标上的提升效果，我们下面使用相对指标进行评测。 考虑到基于用户的最近邻算法(UserKNN)在同类工作中得到的关注最多而 F1值是衡量不同推荐算法效果的综合指标，本文考察基于多维时间序列分析的音乐推荐方法相较于 UserKNN在推荐 F1值上的提升效果(倍数)， 如图3.7所示。 {48%：其中，横坐标表示被推荐歌曲的数目，纵坐标表示算法推荐 F1值的提升倍数。} 如果纵坐标取值大于零，表明本文所述方法的推荐F1值相较于 UserKNN算法有所提升； 如果纵坐标取值为零，表明效果没有提升； 如果纵坐标取值小于零，表明本文所述方法不但没有提高推荐 F1值而且还有所下降。 由图 3.7可以看出，无论被推荐歌曲数目为何值，纵坐标取值总是大于零，说明本文所述方法能够提升推荐的 F1值且提升幅度在 80%以上。 {48%：随着推荐列表长度的增长，提升的幅度也继续增长 (可达140%)。}

图3.8和3.9展示了随着推荐列表长度的增加，几种不同的音乐推荐算法的均方根误差({45%：如图3.8所示)和平均绝对误差(如图3.9所示)的变化趋势。} {58%：其中，横坐标表示被推荐歌曲的数目，纵坐标表示不同推荐算法的误差。} {42%：由图可以直观地看出，本文所提方法的推荐误差较之其他几种算法都比较小。} 随着推荐列表长度的增加，列表中无效的歌曲增多，使得推荐误差有所上升，但这种上升幅度

图 3.7: 本文方法相对于UserKNN在F1-Score指标上的提升也是是非常小的。

图 3.8: {60%：不同推荐算法在均方根误差上的表现}

综合以上实验结果可以看出，无论是从命中率/准确率的角度去考察算法的优劣，还是从预测误差的角度去考察算法的优劣， 本文所述的基于多维时间序列分析的音乐推荐算法都能够取得比参考算法较好的效果。 这验证了本文所提方法的合理性，说明将用户行为建模为一个多维时间序列进行分析能够全面细致地实现对用户行为的刻画进而提升推荐效果。

图 3.9: {61%：不同推荐算法在绝对误差上的表现}

第四章 基于用户长期行为、中期行为和即时行为的综合音乐推荐方法

{43%：由前文可知，基于多维时间序列分析的个性化音乐推荐方法与一些参考推荐方法相比能够取得比较好的推荐效果，} 这包括较高的推荐命中率和较低的预测误差两方面的含义。 但是，朴素的多维时间序列分析音乐推荐方法还存在以下问题：

1.考察片面。 该方法主要是通过对用户当前会话期内的行为序列进行分析来预测用户接下来的行为， 显然其只考虑了用户中期行为对用户未来行为的作用却忽视了用户长期行为以及用户即时行为的影响， 这与用户的未来行为受到其长期行为、中期行为和即时行为共同作用的直觉不相符。 2.冷启动问题。 该方法顺利工作的前提是用户已经产生了一些行为，即用户

已经在当前会话期内收听了一定数量的歌曲。 也就是说，朴素的基于多维时间序列分析的音乐推荐方法适合在用户状态稳定时为用户做出合理的推荐。 但是，对于刚进入给定平台的用户，朴素方法的推荐效果就要大打折扣，这是因为用户刚进入给定平台时所产生的行为很少， 而对应的听歌序列还不足以被朴素的多维时序方法进行分析和预测。 为了解决上述两个问题，我们将在本章中给出一种基于用户长期、中期和即时行为的综合音乐推荐方法并通过实验验证综合推荐的效果。 最后，本文对综合音乐推荐中的一些基本问题进行讨论分析，比如会话期长度的选择和文本分析方法的选取等。

4.1 综合音乐推荐

{42%：为了综合考虑用户长期、中期和即时行为对用户未来行为的影响和作用，} 本文从基于用户长期行为的音乐推荐、基于用户中期行为的音乐推荐以及基于用户即时行为的音乐推荐中各选择一种算法进行综合， {44%：试图通过这种综合全面考察用户行为的时间相关性。} 至于如何对不同的算法进行综合，有很多策略可以进行选择。 Burke的分类方法 [8]区分出了七种不同的综合策略，而从更综合的角度来看这七种策略可以概括为三种基本设计思想： {42%：整体式综合设计、并行式综合设计和流水线式综合设计。} 其中，整体式综合设计实际上是一种推荐算法，其在推荐内部对不同推荐策略获取的特征进行综合，然后基于此生成最终的推荐列表；并行式综合设计和流水线式综合设计都需要至少实现两个不同的推荐算法，其中并行式综合设计将不同推荐算法获取的最终推荐结果加以综合， 而流水线式综合则将推荐分为不同的阶段然后在不同的阶段使用不同的推荐策略进行推荐并最终得到推荐结果。

显然，本文所述的音乐推荐不具有明确的阶段划分，因此使用流水线式的综合设计不太合适， 而整体式综合设计和并行式综合设计均可应用，只是二者综合的阶段不一致。 由于整体式的综合设计思路是从内部对不同推荐策略的结果进行综合，我们认为这样的早期综合能够得到比较符合用户偏好的特征， {46%：因此我们主要考察这种综合设计方法，如图4.1所示。}

图 4.1: 基于用户长期、中期和即时行为的综合音乐推荐方法框架

考虑到用户的未来行为受到用户长期行为、中期行为和即时行为的共同影响，我们从这三类算法中各 选择一种预测出用户可能收听的下一首歌曲在 K个隐含主题上的概率分布。 然后，我们使用一定的综合策略将这三个中间概率分布加以综合以得到最终的目标概率分布。 在此基础上，我们计算曲库中所有歌曲对应的概率分布于目标概率分布的相似度并将歌曲按照相似度由大到小排列。 {44%：最后，我们将排名靠前的 N首歌曲推荐给用户。}

要实现这样一个基于用户长期、中期和即时行为的综合音乐推荐，有两个难点需要解决：

1.如何从基于用户长期行为、中期行为和即时行为的音乐推荐方法中选择带 具体的待综合算法。

{41％： 2.采用何种综合策略对给定的三种音乐推荐方法进行综合。 }

4.1.1方法选取

在需要选择的三种音乐推荐方法中，最容易选择的自然是基于用户中期行为的音乐推荐方法， 本文给出的基于多维时间序列分析的音乐推荐方法就是一种基于用户中期行为的音乐推荐， {58％：而且该方法能够取得比较理想的推荐效果。 } 假设使用基于多维时间序列分析的音乐推荐方法预测得到的用户可能收听的下一首歌曲对应的概率分布为.s^mtsa，如式4.1所示。

至于基于用户长期行为的音乐推荐方法的选取，我们选择朴素的全局特征推荐。 与基于协同过滤的推荐相比，基于全局特征的音乐推荐实现起来比较简单，而且能够预测得到用户的整体偏好， 其得到的结果与本文所给的基于多维时间序列分析的音乐推荐方法一致，方便后续综合。 基于全局特征的推荐认为用户长时间的听歌行为在一定程度上能够比较用户内在的音乐偏好， 进一步地此类算法认为用户下面可能收听歌曲的特征近似于用户所收听的全部歌曲的平均特征。 对于用户集中收听了π＋n首歌曲的用户u来说，我们设使用基于全局特征推荐得到的目标概率分布为.s^global，如式4.2所示。

其中，目标歌曲在第i个隐含主题上的概率取值如式4.3所示，j表示用户收听的第j首歌曲的索引。

在选定了基于用户中期行为和长期行为的音乐推荐方法之后，我们需要进一步选取基于用户即时行为的音乐推荐方法。 基于用户即时行为的音乐推荐思想比较简单，即认为用户的状态是稳定的，进一步地认为用户可能收听的下一首歌曲与用户当前收听的歌曲具有类似的特征。 对于用户集中收听了π＋n首歌曲的用户u来说，我们设使用基于即时特征推荐得到的目标概率分布为.s^local，如式4.4所示。

其中，目标歌曲在第i个隐含主题上的概率取值如式4.5所示，j表示用户收听的第j首歌曲的索引。

4.1.2综合策略

在给定了三种类型的音乐推荐方法之后，我们还需要选择合理的综合策略将它们综合起来。 加权是一种非常重要的综合思想，其一方面可以将不同的推荐策略综合起来考虑，另一方面能够通过权重调整不同推荐策略的影响和贡献， 本文所给出的这种基于用户长期、中期和即时行为的音乐推荐方法正是使用这种思想来完成对用户长期行为推荐、中期行为推荐以及即时行为推荐的综合。 在使用三种不同的音乐推荐方法预测得到三种不同的中间概率分布即.s^mtsa、. s^global和.s^local之后，我们按照如式4.6所示的方法对它们进行综合以得到最终的目标概率分布.s^all。

目标概率分布中每一维的取值可以按照式4.7获得。

其中，α表示基于用户中期行为的音乐推荐方法对应的权重，β是用以调整基于用户长期行为和基于用户即时行为推荐方法贡献的参数。 对于公式4.6和4.7，我们按照式4.7定义参数alpha。 由于我们不是主要研究基于用户长期行为和即时行为的音乐推荐，因此我们不过分关注参数β的变化。 为方便起见，我们给定β=0.5，即基于用户长期行为的推荐和基于用户即时行为的音乐推荐具备相同的权重。

{43%：式4.7中，n表示用户在当前会话期内所收听的歌曲的数目，即用户当前会话序列的长度；} a和b是用以调整α变化的参数，它们共同确定了基于用户中期行为的推荐何时起作用以及变化趋势； 算子f的定义式4.8所示。

图4.2给出了当a=10且b=5时，α的取值变化情况，其中横坐标表示用户当前会话序列的长度，纵坐标表示α的取值。 由图可以看出，当序列长度小 第四章基于用户长期行为、中期行为和即时行为的综合音乐推荐方法于5时α=0，这背后体现的思想是： 序列长度过短时基于用户中期行为的音乐推荐不起作用，此时主要使用基于用户长期行为的音乐推荐和基于用户即时行为的音乐推荐为用户作出推荐， {86%：这在一定程度上解决了冷启动问题；} 随着纵坐

标的增加，用户当前会话序列不断增长，α的取值不断增加，这背后的含义是： 基于用户中期行为的推荐在用户状态稳定时比较有效且随着用户行为的增加，可分析的信息增多，基于用户中期行为的推荐所起的作用应增强。

图 4.2: Alpha取值变化示意图

4.2实验结果

为了评估本节所给出的基于用户长期、中期和即时行为的综合推荐方法的效果，我们将之与第三章中给出的朴素的基于多维时间序列分析的音乐推荐方法进行比较。 实验所采用的数据集、实验条件设置、实验评测指标等信息与3.7节中所述一致。

图4.3展示了基于用户长期、中期和即时行为的综合音乐推荐和朴素的基于多维时间序列分析的音乐推荐方法在命中率指标上的表现。 {42%：其中，横坐标表示推荐列表的长度，纵坐标表示推荐命中率，MTSA表示朴素的时序推荐方法， MTSA ALL表示综合推荐方法。} 由图可以非常直观地看出，基于用户长期、中期和即时行为的综合音乐推荐方法能够取得比朴素的时序分析方法更高的命中率。 图4.4展示了这两种方法在均方根误差这一评测指标上的表现。 其中，纵坐标表示均方根误差，纵坐标取值越小，误差越小，算法的推荐效果越好。 由图4.4可以看出，综合的推荐方法能够取得比朴素时序分析方法更低的预测误 差。 这两类算法在准确率、 F1-Score、平均绝对误差等指标上的表现与其在命中率和均方根误差上的表现类似，这里不再赘述。

由实验结果可以看出，本文所给出的基于用户长期、中期和即时行为的音乐推荐方法确实能够提升朴素的多维时序分析方法的推荐效果， 这从侧面也验证了用户的未来行为是受到用户中期行为、长期行为和即时行为共同作用和影响的。

图 4.3: 综合方法与基本方法在命中率指标上的比较

图 4.4: {48%：综合方法和基本方法在预测误差指标上的比较}

4.3分析讨论

本文上面分别介绍了本文所给出的一种基于多维时间序列分析的个性化音乐推荐方法以及以此为基础构建的一种基于用户长期、中期和即时行为的综合音乐推荐方法， 本节将就其中一些细节问题进行简单地分析和讨论，主要包括最大分析长度的设定以及不同文本建模方法的比较。

4.3.1最大分析长度

本文所述的基于多维时间序列分析的个性化音乐推荐方法及综合音乐推荐方法主要关注地是用户在当前会话期内所收听(喜欢)歌曲构成的序列， 那么如果用户在当前会话期内收听了很多首歌曲(假如是1000首)，我们是不是要对所有这些歌曲构成的序列进行分析呢？ 显然，这么做既不划算也不合理，因为用户所处的上下文环境是时刻改变的，很久之前的行为对当前环境状态的表征作用很有限甚至会形成干扰。 为此，我们设置一个“最大分析长度”，这样我们只需要对一定范围内的序列进行分析即可，既减轻了计算的负担同时也排除了老旧数据的干扰。顾名思义，“最大分析长度”就是从当前时刻算起，算法所考察的用户行为序列的最大长度。 假如设置最大分析长度为4，对于依次收听了歌曲 s1、 s2、 s3、 s4、 s5、 s6、 s7的用户来说， 算法只分析由 s4、 s5、 s6、 s7构成的序列而忽略 s1、 s2、 s3。 {40％：如果调整最大序列长度为3，那么歌曲s4也将被忽略，算法只考察由s5、s6、s7构成的序列。} 图4.5展示了随着最大可分析长度的增加，本文给出的多维时序推荐方法在命中率指标上的表现(推荐列表长度设为100)。 {64％：图中横坐标表示最大可分析长度，纵坐标表示算法的命中率。} 由图可以看出，对当前数据集来说，当最大分析长度为15时，算法的命中率基本就保持稳定了，这说明更早之前的数据对提升命中率意义不大。 因此，可以取最大分析长度为15。 当然，不同的数据集最大分析长度取值不一样，但差别不大，一般取15 30为宜。

4.3.2不同文本分析方法比较

我们在第二章介绍了向量空间模型、词频-逆文本频率模型、隐含狄利克雷分配等几种不同的文本建模方法，同时提到 LDA模型与 VSM模型和 TF- IDF模型相比，一方面能够更好地描述文档的文本特征，另一方面可以解决一词多义和一义多词的问题， 因此本文所述的基于多维时间序列分析的音乐推荐方法和基于用户长期、中期和即时行为的音乐推荐方法在对歌曲集对应的文档集进行建模时都采用了 LDA模型。 为了直观地展现LDA模型的好处，本节通过实验来展现使用不同的文本建模方法来执行基于用户即时行为推荐所获得的命中率， 如4.6图所示。 {60％：其中，横坐标表示推荐列表的长度，纵坐标表示算法获得的命中率。} {41％：纵坐标取值越大，算法获得命中率越高，算法推荐效果越好。} 由图可以非常直观地看出，使用隐含狄利克雷分配对文档集进行建模确实能够得到比传统的向量空间模型和词频-逆文本频率更好的推荐效果。

图 4.5: {50％：不同最大分析长度对应的命中率变化情况}

图 4.6: {60％：不同文本建模方法的效果比较}

第五章 系统实现

前述章节我们介绍了一种基于多维时间序列分析的音乐推荐方法和在此基础上的一种基于用户长期、中期和即时行为的音乐推荐方法，并通过实验从多个角度验证了本文上述的有效性。 为了进一步验证上述方法的可行性并解决系统实现中的一些关键问题，本文实现了一个个性化音乐推荐原型系统，即Sweetfm。 本章我们将对该原型系统进行详细介绍并给出一些具体的技术方案。

5.1系统架构

图 5.1: {51％：推荐系统和网站其他系统之间的关系}

尽管优秀的推荐算法能够为用户推荐合理的结果，但只靠推荐很难构成一个完整的可用系统。 要构建一个可

用的推荐系统，比如一个音乐推荐网站，就需要考虑推荐与系统其他组件的关系，只有这样才能最终实现推荐的价值。 项亮在文献 [1]中给出了如5.1所示的一般意义上推荐系统和网站其他系统之间的关系。 首先，基本上所有的网站都配有一个用户界面，即UI系统，该系统主要用来向用户展示页面效果以及与用户进行交互。 其次，网站往往还会配置日志系统，该系统主要用来将用户在用户界面上的各种有效行为记录下来并保存到对应的日志存储系统中。 需要注意的是，这里的日志存储系统既可以是数据库，也可以是缓存，还可以是文件系统。 推荐系统作为一个推荐网站的核心，其主要作用是分析存储在日志存储系统中的用户行为历史， 在此基础上生成该用户对应的推荐列表并将结果直接展示到用户界面上以供用户体验。 由图5.1可知，推荐系统要想将强大的作用发挥出来，需要依赖于用户的历史行为信息和用户界面等组件。

图 5.2: 原型系统框架

5.1图虽然给出了一般意义上的推荐系统的架构，给本文所实现的原型系统带来了很大启发，但这种架构过于简单，不能直接用于本文的原型系统。 在认真分析本文所提推荐方法框架的基础上，本文按照图 5.2对原型系统进行设计。 {41％：本文所实现的原型系统主要包括客户端、用户服务接口、处理层和数据层构成。} 客户端对应于图 5.1中的 UI系统，旨在为用户提供一个可以收听推荐歌曲并产生交互的窗口，具体到实现上既可以使用 Web，也可以是 Mobile还可以是普通的窗口客户端。

{44％：用户服务接口用于将用户在客户端上的行为传递给后续模块进行处理以及将推荐的结果返回给客户端。}

{45％：处理层主要用于完成相关数据处理，其包括在线处理模块和离线处理模块。} 其中，离线处理模块主要包括从百度音乐、豆瓣音乐、虾米音乐、 Lastfm等数据源爬取歌曲基本属性数据和标签数据的 爬虫系统以及对歌曲对应的文档集合进行语义分析以获取每一首歌曲对应的隐含主题分布的语义分析模块。 需要强调的是，这里离线的含义是指该模块的工作与用户行为无关，在具体实现上可以设定一定的时间间隔或周期重复执行。

在线处理模块主要用于实时记录和处理用户的行为并为之生成最终推荐结果，其主要包括日志系统、序列生成系统、标签系统以及 Storm推荐引擎几个部分组成。 日志系统对应于文献 [1]中的日志系统，用于将用户的行为记录到数据存储系统中以待后续分析。 标签系统主要用于记录用户对当前所听歌曲所标注的标签，其与离线模块中的爬虫系统一起生成最终的歌曲标签信息，一定周期后可供语义分析模块处理。 序列生成系统用于根据用户在当前会话期内的行为生成对应的歌曲序列。 Storm推荐引擎模块的主要作用是使用 Storm框架对用户当前会话期的歌曲序列进行实时分析并产生最终的推荐结果。

数据层主要完成用户属性、歌曲属性以及用户行为的存储功能，其包括数据库接口 DAO以及具体数据库两部分。 其中，数据库接口提供供其他层次模块调用的方法以避免直接操作数据库，增强了独立性。 具体数据库即真实的数据存储引擎，既可以是 Mysql，也可以是 Oracle，当然也可以是 SQLServer。

用户产生一个积极行为到系统为其推荐歌曲的过程如下所示：

1.用户在客户端产生 “ 即将收听完当前歌曲 ” 的行为。

2.用户服务接口接收用户的当前行为状态并将该状态传递给在线处理模块中的日志系统和序列生成系统。

{43％：3.日志系统将用户当前行为状态通过数据库接口DAO记录到数据库中。}

4.序列生成系统通过数据库接口 DAO读取日志数据库中的用户行为，构建其在当前会话期的收听序列并将该序列传递给Storm推荐引擎。

5.推荐引擎对用户当前会话期的收听序列进行分析和处理，生成推荐列表并通过数据库接口保存到数据库。

{44％： 6.用户服务接口通过数据库接口从数据库中读取推荐列表。 }

{57％： 7.用户服务接口将推荐列表展示给用户。 } {43％： 用户对歌曲打标签的执行过程如下所示： }

{43％： 1.用户在客户端选定一首歌曲。 }

2.用户为选定歌曲打标签。

3.用户服务接口接收用户的打标签行为以及打标签的对象和标签内容并将它们传递给在线处理模块中的标签系统。

{48％： 4.标签系统对标签进行分析和处理。 }

{41％： 5.标签系统将处理过的内容通过数据库接口DAO保存到数据库中。 } 类似地，可以给出用户其他行为的处理过程。

5.2离线处理模块

如上所述，离线处理模块所做的工作主要包括爬虫系统和语义分析系统两部分组成，这些工作都是独立于用户进行的，所以称之为离线。 本节将详细介绍上述框架中的离线处理模块中设计的一些技术和工具，而作为算法核心的语义分析模块将是介绍的重点内容。

5.2.1爬虫系统

图 5.3: 爬虫系统工作流程图

网络爬虫是一种能够按照一定地规则自动地抓取互联网上的网页并对网页内容进行解析的网络机器人，又称之为网络蜘蛛。 本文所实现的音乐网络爬虫首先从豆瓣、虾米、百度等音乐网站上抓取相关歌曲网页，然后对网页的 HTML进行分析和解析， 进而得到歌曲的名称、创作者、发行时间、歌词等基本属性以及用户对歌曲所打的标签内容， {44％： 最后将这些内容进行整理并保存到数据库中。 } {41％： 对于一个待抓取的歌曲页面 URL，本文按照流程图 5.3进行工作。 }

网络爬虫目前已经广泛应用在数据挖掘、搜索引擎、信息检索、推荐系统等领域，同时也出现了很多网络爬虫框架以简化爬虫的实现。 {48％： 其中， Scrapy[48]是一种纯 python实现且构建于异步框架 twisted之上爬虫框架，其用户只需要定制开发几个模块就可以轻松地实现一个爬虫， } {95％： 用来抓取网页内容以及各种图片，非常之方便。 } 要想创建一个网络爬虫，用户只需要执行命令 scrapy startproject projectname就可以得到如图5.4所示的项目目录， {41％： 生成的目录中包含若干文件，代表 Scrapy框架中的多个模块。 } {45％： 用户只需要在对应的文件中实现相应模块即可。 } 其中， items.py文件对应于 Scrapy中的项模块，用于定义抓取结果中单个项所需要包含的所有内容，如歌曲的名称、创作者、发行时间等； piplines.py对应于Scrapy中的管道模块，定义如何对抓取

到的内容进行再处理，例如输出文件、写入数据库等； {60%：spider目录下存放写好的爬虫实际抓取逻辑。}

图 5.4: Scrapy结构示意图

由上可知，在抓取指定网页内容之后需要对网页进行解析，这里我们使用 Python语言的第三方包 BeautifulSoup[4]来解析下载下来的歌曲网页对应的 HTML文件。 BeautifulSoup使用起来非常简单，可以非常容易地完成对 HTML文件的解析，同时它也支持按照不同的条件来查找相关元素， {41%：比如按标签查找、按属性查找、按名称查找、按结构查找等。}

5.2.2结巴分词

{55%：由信息检索等相关知识可知，要对文本进行分析，往往首先需要进行分词，即将连续的字序列按照一定的规范重新组合成词序列的过程。} {43%：比如将句子“南京市长江三桥”分成“南京/市长/江三桥”或者“南京市/长江/三桥”这样的词汇序列。} 目前，学术界和工业界也出现了众多成熟的分词工具，比如基于词频词典的机械中文分词引擎 SCWS、中科院的汉语词法分析系统 ICTCLAS(Institute of ComputingTechnology, ChineseLexical Analysis System)、基于 HTTP协议的开源中文分词系统 HTTPCWS以及支持多种分词模式的结巴分词等。 考虑到各工具的效率、可用性、精度以及源码获取难易程度，本章所述原型系统采用结巴分词作为分词工具。

结巴分词 [23]是一个基于Python语言开发的开放源代码的中文分词工具，其由百度(Baidu Inc.)的Sun Junyi开发并发布在Github上，其目标是 {90%：“做最好的Python中文分词组件”} 结巴分词自从2012年10月7日被发布到pypi以来不断地被改进和完善，目前已经发布的最新版本为0.32。 其所采用的算法如下所示：

{96%：1.基于Trie树结构实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG）。}

{91%：2.采用了动态规划查找最大概率路径，找出基于词频的最大切分组合。}

{93%：3.对于未登录词，采用了基于汉字成词能力的HMM模型，使用了Viterbi算法。}

总得看来，结巴分词之所以能够被广泛关注和使用，主要是因为其具有以下特点：

安装简单。 用户可以直接通过“easy install jieba”或者“pip install jieba”进行安装，就行安装普通的Python包一样。

2.使用简单。 用户只需使用一句代码即可实现分词操作，如用户通过如下代码即可将句子“南京市长江三桥”分词若干词汇序列并将词汇保存到列表中。

seg_list = jieba.cut("南京市长江三桥")；

3.支持多种分词模式。 {60%：结巴分词支持三种分词模式，即精确模式、全模式和搜索引擎模式。} {92%：其中，精确模式试图将句子最精确地切开，适合文本分析；} {100%：全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；} {100%：搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。}

4.支持多种操作。 除了基本的分词之外，结巴分词还支持添加自定义词典、关键词提取、词性标注、并行分词、繁体分词以及Tokenize等功能。

5.支持多语言。 除了最基础的Python语言版本外，目前结巴分词还支持Java、 C++以及Node.js三种语言且源码均已发布到Github上。

5.2.3 Gensim软件包

为了得到每一首歌曲在隐含主题上的概率分布，我们使用以LDA为代表的主题模型分析方法对每首歌曲对应的文档进行分析，这里的文档是由用户为歌曲标记的标签构成。 LingPipe[29]和Mallet[32]都是非常优秀的自然语言处理软件包，但考虑到它们比较复杂且对LDA的实现欠佳，我们选择使用另一个优秀的软件包Gensim[15]。 Gensim最初是作为一组被用在捷克数学文献存取网站dml.cz中的Python脚本的集合而出现，而其功能只是简单地根据给定的文档来生成一组近似的文档，Gensim正是“ Generate Similar ”的简称。 为了使用隐含语义的方法对文档分析，作者于2010年将其扩展为一个 Python包，随后 作者于2011年开始使用 Github来管理源代码并于2013年设计了 Gensim独特的 Logo和网站。 Gensim可以非常方便地实现主题模型，正如其介绍中所说— {46％：“为人类而设计的主题模型开发包”} ，其主要具有以下特点：

1.可扩展性。 Gensim通过使用增量式的在线训练方法可以处理大规模的语料库，从而不需要将所有语料一次性装入内存，降低了内存的负担，增强了可扩展性。

2.平台无关性。 Gensim是纯Python实现，可以运行在Linux、 Windows、 OS X以及其他支持Python和Numpy的平台上。

3.鲁棒性。 Gensim已经被很多个人和组织应用在各种系统中超过四年，早已过了一个开源项目的“妈妈，我发布了一个脚本”的初始阶段。

4.开源性。 Gensim开放源代码，其使用GNU LGPL许可证，允许个人和商业机构使用和修改该项目。

5.高效性。 Gensim中的各种算法都是使用经过优化的方法进行实现的，使得算法的效率较高； 另外，Gensim实现了一些算法的分布式版本，使得算法可以并行执行或者在集群上执行，进一步增加算法的执行效率。

6. {45％： Gensim包含对一些常用数据格式的高效内存实现方式，同时支持不同数据格式之间的转换。}

7. Gensim除了可以快速地执行主题模型建模，还提供了快速计算文档相似性的方法。

{42％：下面简单介绍以下使用Gensim软件包进行LDA建模的方法和流程：}

1.准备语料库，这里就是需要进行主题模型建模的文档集合。

2.对文档集合中的每一篇文档进行分词并利用分词的结果构造词典，同时可以得到每个词或者词组在词典中的编号。

{42%： 3.词典生成好之后就生成语料库，语料库中的每一个语料与文档集合中的每一篇文档一一对应，} {46%： 而语料的表示形式即是文档的向量空间模型，即词典中的某个词或词组在该文档中出现的次数。}

4.将上述向量空间模型表示的语料库转换成TF-IDF模型表示的语料库，即此时得到的语料库可以表征每一个词或者词组的重要程度。

{44%： 5.进行LDA主题模型建模，得到建模结果。}

5.3在线处理模块

5.3.1序列生成

用户在当前会话期内收听的歌曲序列是本文所述方法和框架的输入，本节将介绍序列生成系统是如何根据用户的行为生成这种序列的。 需要强调的是，作为输入的序列是用户产生积极行为的歌曲序列，下面将给出对于积极行为的定义。

用户在收听一首歌曲的时候，可能对其产生两种典型的行为，即积极行为和消极行为。 典型的积极行为包括用户收藏当前歌曲、分享当前歌曲、完整收听或者收听当前歌曲的绝大部分等，而典型的 消极行为包括用户将歌曲丢进“垃圾桶”、选择跳过当前歌曲、只收听当前歌曲的一小部分等 。 对于收藏、分享、丢进“垃圾桶”、跳过这些比较直观地行为我们不再做进一步描述， 而对于用户收听歌曲比例所表达的态度，本文认为如果用户收听当前歌曲的

长度超过了歌曲总长度的50%那么用户喜欢该歌曲，否则认为用户不喜欢当前歌曲，即消极行为，如式5.1和5.2所示。 其中，total length表示歌曲的总长度， past length表示用户收听该歌曲的时间长度，而pastRatio表示用户收听长度占歌曲总长度的比例。 attitude表示用户对歌曲的态度，当pastRatio不小于50%时取值为1表示积极态度，否则表示消极态度。 假如歌曲“大海”(记为 s)总长度为4分40秒，即280000毫秒，用户 a收听至1分钟即60000毫秒时跳过， 那么对应的有 pastRatio(a， s)=21.43%，那么我们认为用户 a不太喜欢歌曲“大海”， 而用户 b收听至4分钟即240000毫秒时跳过，那么对应的有 pastRatio(b， s)=85.71%， {46%： 我们认为用户 b比较喜欢当前歌曲。} 至于b为什么没有听完整首歌曲，即pastRatio(b， s) .=

100%，可以认为是用户b收听歌曲的时间过长，想要换一首以免疲劳，而仍然收听了过半歌曲说明他还是喜欢此首歌曲的。 序列生成系统在歌曲播放至50%时便读取数据库中保存的用户当前会话期内的积极歌曲序列，并结合当前歌曲组成新的序列传递给推荐引擎。

5.3.2推荐引擎

在由序列生成系统得到用户当前会话期的收听列表后，图5.2所述的系统框架则 激活基于 Storm框架的多维时序推荐引擎去读取列表进行处理并生成最终的推荐列表。 之所以使用 Storm框架是因为音乐推荐系统是一个对实时性要求比较高的系统，需要及时地为用户生成推荐结果， 而 Storm恰恰是这样一个开源的且面向实时性处理的分布式框架。 第二章中已经简单介绍了Storm基本组成以及对应的DRPC的基本内容，本节将介绍该框架在本文原型系统中的应用。 由前文可知，在Storm框架中，每一个分布式的计算任务都被称之为一个拓扑，而拓扑按照一定的拓扑结构实现的。 因此，本文系统原型中核心的推荐引擎模块也对应着一个拓扑任务，用以完成为用户实时生成推荐列表的工作，其对应的拓扑图如图5.5所示。

1.推荐引擎通过数据库接口DAO从数据库中读取用户当前会话期内的歌曲列表。

图 5.5: {65%：推荐引擎工作拓扑结构图}

2.读取共享内存中每一首歌曲隶属于每一个隐含主题的概率，进而将 1中的序列分割成 K个子时间序列。 其中，每一个子时间序列对应一个隐含主题，序列对应的变量为歌曲在该主题上的隶属概率。

3.对2中的 K个子时间序列进行时序预测，预测下一期的取值，即用户可能收听的歌曲在该主题上的隶属度。

4.将K个主题上的预测值汇总，得到下一首歌曲的完整概率分布。

{44%：5.计算曲库内所有歌曲与该歌曲的相似度。}

{40%：6.将5中的相似度按照由大到小的顺序排列，并取前 10作为推荐列表。}

7.将推荐列表返回。

由图5.5可以非常直观地看出，本文所实现的多维时序推荐引擎是完全并行化的， 即各个子时间序列的分析与预测以及不同歌曲与目标歌曲之间相似度的计算都是可以并行执行的， {58%：这显然能够提升系统的执行效率。} 至于在并行化的时候选择多少并行结点，这个取决于具体的应用场景，本文所研究的多维时间序列分析方法由于存在 K个隐含主题，因此自然选择 K个并行结点。 当然，能够并行的原因是我们弱化了各个隐含主题之间的相关性，认为各个隐含主题之间是相互独立的。

在构建了上述拓扑结构并实现后，下面需要做的就是将其部署到服务器上，具体的部署过程如下所示。

1.启动zookeeper： zkServer.sh start。 {78%：Zookeeper分布式服务框架主要是用来解决分布式应用中经常遇到的一些数据管理问题，如统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等。} {100%：Nimbus和Supervisor节点之间所有的协调工作是通过Zookeeper集群来实现的。}

2.启动 nimbus： storm nimbus。 {100%：主控节点（Master Node）上运行一个被称为Nimbus的后台程序，它负责在 Storm集群内分发代码，分配任务给工作机器，并且负责监控集群运行状态。} Nimbus的作用类似于Hadoop中JobTracker的角色。

3.启动 supervisor： storm supervisor。 {100%：每个工作节点（Work Node）上运行一个被称为 Supervisor的后台程序。} {100%：Supervisor负责监听从 Nimbus分配给它执行的任务，据此启动或停止执行任务的工作进程。} {100%：每一个工作进程执行一个Topology的子集；} {100%：一个运行中的 Topology由分布在不同工作节点上的多个工作进程组成。}

4.启动 UI： storm ui。 {63%：storm UI是一个可以查看 storm运行状态的的一个网站，可以查看Topology的执行状态。}

5.启动drpc： storm drpc。 通过 DRPC，其他机器可以远程执行 Topology。

6.提交topology： storm jar SweetFM.jar com.wst.sweetfm.topology.DRPC MTSATopology sweetfm。 {55%：将实

现好的 Topology提交至Storm集群进行执行。 }

7.显示所有 topology： storm list。 {60 %： 将所有的计算任务以列表的形式展现出来。 }

8.杀死指定topology： storm kill sweetfm。

mithunsatheesh给出了一种使用 PHP远程调用 Storm集群中运行的 Topology的方法，如下表所示。

include ”includes/drpc/DRPC.php”;

$drpc = new DRPC(”xxx.xxx.x.xx”, 3772, NULL);

$result = $drpc-)execute(”CallFunctionName”, $params);

具体到我们的原型系统中即是：

include ”includes/drpc/DRPC.php”;

$drpc = new DRPC(”114.212.84.238”, 3772, NULL);

$seq = “4, 6, 1, 8, 2, 0”

$result = $drpc-)execute(”sweetfm”, $seq);

首先构造一个 drpc对象，然后以适当的格式给出用户在当前会话期的歌曲序列并作为参数传递给 drpc对象的 execute函数， Storm集群上的 sweetfm接收传递过来的参数并进行计算，计算结束之后将结果返回给 result，即一个歌曲推荐列表， 如”[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]”

5.4系统效果

本节给出本文实现的系统原型的效果截图，如下所示。 在下图中，从左上到右下共有六张效果图，分别为游客登录效果图、用户注册效果图、 用户登录效果图、系统主界面效果图、用户对歌曲打标签效果图以及用户调整播放音量效果图。

图 5.6: 原型系统效果图

第六章 总结与展望

6.1工作总结

近年来，个性化音乐电台应用大量出现，但这些应用的音乐推荐算法要么没有考虑用户听歌行为与所处上下文环境之间的关系， 要么对用户行为时间相关性处理地不够合理，导致推荐的结果不够理想。 为了提升音乐推荐的效果，本文提出了一种基于多维时间序列分析的音乐推荐方法，较为合理地利用了用户中期行为。 进一步地，本文给出一种基于用户长期、中期和即时行为的综合音乐推荐方法，全面考察了用户行为的时间相关性。 {46

%：通过实验和原型系统的实现，本文初步验证了所述方法的有效性和可行性。} 本文的主要贡献如下所示：

1.提出了一种基于多维时间序列分析的个性化音乐推荐方法，该方法在使用主题模型将歌曲表示成由 若干隐含主题构成的概率分布的基础上将用户在当前会话期内的行为建模为一个多维时间序列。 通过对该多维时间序列的分析，该方法能够较好地预测用户行为偏好，并给出合理的推荐结果。

2.给出了一种基于用户长期、中期和即时行为的综合音乐推荐方法，综合考虑了用户长期行为、中期行为以及即时行为对用户未来行为的作用和贡献。

3.基于上述方法实现了一个原型系统，并通过Storm实现了上述方法的并行化，提高了处理效率。

6.2工作展望

1.本文只是使用用户对歌曲所打的标签去构造歌曲对应的文档，可以尝试使用更丰富的的文本信息表征歌曲，进而构造更符合歌曲内在特征的文档。

{40%：2.本文所述方法主要是对用户的积极行为进行分析，可以尝试加入一些用户的否定行为信息。}

3.本文工作任务不同隐含主题之间是相互独立的，可以进一步考察不同主题之间的相关关系，提升分析的准确性。

附录 A数据集使用说明

A.1权利声明

本数据集抓取自Lastfm，所有数据归Lastfm所有，禁止商业用途。 如果您想使用该数据集进行科研活动，请务必给出对Last.fm及本文的引用信息。

A.2数据特点

{44%：1.包含完整的用户、歌曲、曲作者的基本信息。}

{43%：2.包含丰富的用户行为记录，可用于构造用户行为序列。}

3.包含歌曲、曲作者的显着标签信息，可用于从文本的角度描述歌曲和作者。

4.提供了标签的基本信息。

5.数据被随机分组，可直接用来实验。

A.3组织形式

本数据集使用Mysql进行管理，对应的数据库名为lastfm，您可以非常容易地将其导入并使用。 {48%：数据集包含有5个基本的数据表：}

1.记录表record用于记录用户的收听行为，如某用户在某时间段收听了某歌曲。 record由记录标识符(rid: int)、用户标识符(uid: varchar)、歌曲mbid(mbid: varchar)、记录发生的unix时间戳(uts: varchar)、记录发生的日期时间(datetime: varchar)、记录所属分组(scale: int)等字段构成。

{45%： 2.用户表user用于记录用户的基本信息，其由用户标识符(uid:} varchar)、用户名(username: varchar)、用户国籍(country: varchar)、用户年龄(age: varchar)、用户性别(gender: varchar)、用户注册时的unix时间戳(registeredTime: varchar)、用户注册日期时间(registeredText: varchar)、播放序列(playlist: text)、用户所属分组(scale: int)等字段构成。

3.歌曲表song用于记录歌曲的基本信息，其由歌曲标识符(sid: varchar)、歌曲对应mbid（mbid: varchar）、歌曲名称(name: text)、歌曲时长(duration: varchar)、曲作者标识符(aid: varchar)、曲作者名称(aname: varchar)、专辑名(album: text)、听众数目(listeners: varchar)、播放次数(playcount: varchar)、描述歌曲的显着标签(toptag: text)等字段构成。

4.曲作者表artist用于记录曲作者的基本信息，其由曲作者标识符(mbid: varchar)、曲作者名称(name: text)、曲作者图片的链接(img: text)和描述曲作者的显着标签(toptag: text)构成。

5.标签表tag用于记录标签的基本信息，其由标签标识符(id: varchar)、标签名称（name: text）、标签被创建的次数(reach: varchar)、标签被使用的次数(taggings: text)等字段构成。

A.4字段解析

A.4.1 scale

记录表record和用户user中的scale字段用以表征记录和用户所处的分组编号。 为了方便，本数据集将用户记录和用户分为Unused、 Small、 Whole和Session四类。

其中， Small、 Whole和 Session被 scale字段分割成40组，其中第0组到第9组属于 Small数据集， 第0组到第29组属于 Whole数据集，第30组到第39组属于 Session数据集。 Unused数据的scale设为-1。 显然，Small数据集是Whole数据集的一部分，它们的特点是每一个用户所收听的歌曲都在一个会话期内，即不存在长时中断。 从Whole数据集中划分出 Small数据集的主要目的是方便机器性能不佳的用户使用，对于 Small数据集， 用户可以使用10组中的9组作训练集而余下的一组作测试集。 Session数据集与Whole数据集的主要区别是每一个用户所收听的歌曲至少在两个会话期内。 {45%：类似的，用户可以用其中9组作训练集而余下的一组作测试集。} {47%：下表给出了Small、Whole和Session三类数据集的基本统计信息。}

A.4.2 playlist

数据表user中的playlist字段用以表征用户按序收听的歌曲构成的序列，数据如 “ sid1: ratio1==)sid2: ratio2==)...==)sidn: ration ” 所示。 其中，sid表示被听歌曲的标识符(注: 非mbid)。 ratio表示两首歌之间的时间间隔与前一首歌曲时长的比例，用以表征用户收听该首歌曲的时长比例。 显然，ratio过小表示用户刚开始收听遍跳过，ratio过大表明歌曲被完整收听而且还可能有暂停发生。

A.4.3 toptag

数据表song和artist中的字段toptag表示Lastfm网站中的用户给歌曲或曲作者所打的显着标签，数据如 “ tag1: count1，tag2: count2，...，tagn: countn ” 所示。 {44%：其中，tag表示被打标签名称，count表示标签被标记次数。} 需要注意的是，在Lastfm中，count并非标签被应用于歌曲或曲作者的绝对次数，而是标签相对于被使用最多次的标签的相对次数。 例如在描述歌曲“ Collapse of History ”的标签中，“ industrial ”被使用最多次且次数为200，而标签“ Stars ”被使用100次。 那么，在歌曲记录对应的字段toptag中，“ industrial ”对应的count为100，“ Stars ”对应的count为50，即“ industrial ”: 100，” starts ”: 50，以此类推。

A.4.4其他字段

{42%：数据表中的其他字段都比较简单直观，这里就不再一一介绍。}

A.5应用场景

1.使用文本分析的方法描述歌曲或者曲作者特征。

2.分析用户所收听歌曲的序列，包括跨会话分析和会话内分析。

3.预测用户下一首可能收听的歌曲或者曲作者。

{45%：4.生成用户可能喜欢的播放列表。}

5.标签预测问题。

6.其他适合的应用场景。

表 A.1: 数据集LastfmSeq的统计信息

Small Whole Session

用户数 1530 4590 1690

歌曲数 24992 62422 32218

稀疏度 99.92 99.97 99.92

最大长度 30 30 66

最短长度 10 10 20

中位长度 24 24 30