摘 要

近些年来，互联网相关技术得到快速发展，特别是大数据时代的来临，使得信息过载和选择悖论的问题越来越显著。为了提升信息的利用率，推荐系统应运而生并已广泛应用在电子商务、音乐电台、个性化阅读等领域，比如淘宝、虾米、指阅等。其中，音乐作为一种强力的沟通和自我表达工具业已成为人们日常生活中非常重要的一种消费品。有研究表明，人们平时的听歌行为要远远多于读书、看电视、看电影等行为。为了解决人们在听歌过程中遇到的信息过载和选择悖论问题，一些个性化音乐推荐系统被开发了出来，比如 Lastfm、Pandora、亦歌等。然而，大多数系统没有充分考虑到歌曲划分标准的不确定性，往往只是从单一侧面去描述歌曲，导致对歌曲的刻画比较片面。另外，人们对歌曲的消费在很大程度上与其所处的上下文环境有关，但很多音乐推荐系统没有很好地考虑用户所处的上下文环境对于其行为的影响，导致推荐的结果不符合用户需求。本文给出了一种基于时间序列分析的个性化音乐推荐系统，该系统能够给出比较符合用户偏好的推荐结果。本文的主要贡献和工作如下所示：

1. 本文提出了一种基于时间序列分析的个性化音乐推荐方法，该方法综合考虑了歌曲划分标准的不确定性以及用户所处上下文环境对其偏好的影响。该方法使用主题模型建模的方法将歌曲表示成若干隐含主题的概率分布，同时将用户在当前会话期内内的听歌行为表示成时间序列进行分析和预测。

2. 本文从较细的粒度对用户的听歌行为进行分析和预测，即将用户行为建模成多个时间序列进行分析，而每个时间序列都以歌曲在对应隐含主题上的概率为变量。

3. 本文提出用户的听歌行为具有局部性、全局性以及时序性。进一步地，本文提出了一种基于时间序列分析的个性化音乐推荐框架，该框架综合考虑了用户听歌行为不同特性的作用。

4. 本文实现了一个原型系统，验证了本文所述方法和框架的可行性。其中，本文利用分布式实时计算框架 Storm 实现了所述方法的并行执行，提升了系统的实时性。

关键字： 音乐推荐系统，个性化，主题模型，时间序列

Abstract

In recent years, internet-related technologies have grown rapidly and the era of big data has been coming. As a result, information overload problem and paradox of choice becomes much more obviously. In order to enhance the utilization of information, recommender systems appear and have been applied to many fields, like Taobao in E-commerce field, Xiami in music field, Zhiyue in reading field etc. People has considered that music is an important aspect of their lives and they listen to music frequently. Previous research has also indicated that participants listened to music more often than other activities, like watching television, reading books, watching movies etc. To alleviating the information overload problem and paradox of choice in music field, some music recommender systems have been developed, like Douban, Pandora,Yige etc. However, most of recent music recommender systems do not take the uncertainty of song's classification into consideration instead of describing songs in a single aspect. As a result, they can not characterize songs comprehensively. Besides, users' favors of music are usually sensitive to the context, like the users' location, users' moods, etc. But many of existing music recommender systems ignore the influence of context leading to bad recommendations that do not match users' requirements. In order to get better recommendations, we propose a music recommender system based on time series analysis. Our main contribution and work are:

1. A method for music recommendation based on time series analysis was proposed. The method takes both the uncertainty of song's classification and the influence of context into consideration. The method firstly use topic models to model songs as probability distributions of some latent topics and then it models the users' behavior as time series. By analyzing the time series, the recommendations could be generated.

2. Time series was analyzed in finer granularity, that is, we model a user's behavior as many time series and each time series take how much a song belongs to a topic as variable.

3. A hybrid music recommender framework was proposed that takes into consideration three main features of users' behavior, that is, local feature, global feature and sequential feature.

4. A prototype that is used to verify the feasibility of our recommender method and framework was implemented. Specially, we paralleled the method with an distributed realtime computation framework-Storm, making the system running faster.

Keywords: Music Recommender System, Personality, Topic Model, Time Series

第一章 背景介绍

近些年来，互联网相关技术得到快速发展，信息呈指数级增长，这造成了严重的信息过载(Information Overload)问题。特别是随着大数据时代的来临，信息的增长速度已经远远超过人们可以理解的速度，而海量信息中更是包含着很多对人们来说无用的冗余信息以及会干扰人们抉择的错误信息。面对海量的信息和众多的选择，人们往往无所适从而陷入到选择悖论(Paradox of Choice)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选择。为了解决信息过载问题，减轻人们在做出抉择时所承受的沉重负担，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 信息过载及选择悖论问题的解决方案及技术

名称	特点	典型案例
信息分类	分门别类的组织信息	Yahoo!、58 同城
搜索引擎	根据用户输入的关键字进行查询	Google、百度
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的组织起来以提高人们查询的效率，比如早期的 Yahoo!和国内的 58 同城的信息分类网站。然而，信息分类技术往往依赖于人工，可扩展性差，随着互联网上信息的增加，这显然不合时宜。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。显然，搜索引擎相对于信息分类技术来说已经有了很大的进步，能够适应互联网的快速发展。但是，人们很多时候要么不愿意费时费力去输入关键字，要么无法准确地用关键字去描述自己的想法，比如“今天看什么电影好呢？”、“去哪儿吃饭呢？”等，此时搜索引擎就无能为力了。为了解放人们的双手以及深度挖掘人们的内在偏好和需求，推荐系统技术应运而生。

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，进而将符合用户偏好的信息推荐给用户。目前，推荐系统已经在各个领域得到了广泛的应用，比如电子商务领域的亚马逊和淘宝，在线视频领域的 Netflix 和优酷，在线音乐

领域的 Lastfm 和豆瓣以及个性化阅读领域 Flipboard 和无觅阅读等。图 1.1 展示了电子商务领域 Amazon 的商品推荐系统界面和视频领域 PPTV 的在线视频推荐系统界面。

图 1.1 Amazon 电商推荐及 PPTV 视频推荐

音乐推荐系统是推荐系统在音乐领域的应用，其旨在将人们从海量的音乐中解脱出来，通过分析用户的收听习惯以及歌曲本身的特征来为用户推荐符合口味的歌曲。音乐推荐与传统物品推荐的区别在于歌曲时长比较短、消费代价低、可重复消费以及具有强烈的情感性等特点，而人们的听歌行为往往对上下文环境往往更为敏感。比如，在不同的天气、不同的情绪以及不同的场合，人们的音乐喜好都会有所不同。因此，在做音乐推荐的时候需要充分的考虑歌曲和人们收听行为的特殊性。

为了满足人们对音乐的需求，一些个性化的音乐推荐系统被设计和实现出来，如国外的 Lastfm、Pandora、Spotify 等以及国内的豆瓣电台、虾米音乐以及 Jing 音乐等，图 1.2 展示了豆瓣电台和虾米音乐的推荐界面。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 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。从可见的资料看，Lastfm、豆瓣主要是基于“相似的人往往具有相似的行为”的假设进行推荐。其中，虾米音乐还给出了推荐的同时还给出了做出推荐的原因，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 豆瓣及虾米音乐电台示意图

尽管这些音乐推荐系统能够取得不错的推荐效果，但其仍存在以下几个问题：

1. 在对歌曲进行描述和刻画时，大多数音乐推荐系统没有充分考虑歌曲划分标准的不确定性以及歌曲本身的多重性，导致对歌曲的描述不够全面和完整。此外，Pandora 对歌曲属性的划分往往由具有相关领域知识的专业人士进行，对专业要求过高，代价大，缺乏一定的可扩展性。

2. 很多音乐推荐系统有充分考察用户所处上下文环境对用户的影响，特别是用户听歌序列对其所处上下文环境的表征作用，导致多用户行为习惯的刻画有所偏差。Lastfm、豆瓣等系统考察了用户所处的群体环境，却忽视了用户本身所处的环境，导致所做的推荐不一定符合用户当前的状态。

本文提出了一种基于时间序列分析的个性化音乐推荐系统框架，挖掘了歌曲划分标准的不确定性和歌曲本身的多重性，验证了用户在一定会话期内听歌行为中的序列性特征，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，再者本文挖掘了用户听歌行为的局部性特征和全局性特征，并将之与时序特征进行混合，最后本文实现了一个个性化音乐推荐原型系统，验证了框架的有效性。

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，包括常用的推荐算法、常用的物品相似度计算方法、推荐系统的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 章描述了用户听歌行为的局部性特征和全局性特征并以此为基础给出一个基于时间序列分析的个性化音乐推荐系统框架；第 5 章给出了原型系统的相关实现技术；第 6 章对全文进行总结并给出对未来工作的展望。

第二章 相关工作

本章我们将介绍与本文工作相关的一些工作，首先我们介绍推荐系统领域常用的一些推荐算法，物品相似度的常用计算方法以及一些常用的算法评估标准；其次，我们将介绍文本建模、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；最后我们讨论与本文工作相关的一些音乐推荐算法。

2.1 常用推荐算法

目前，推荐系统领域的研究工作已经取得了长足的进步，同时也出现了很多类型的推荐算法。按照检索物品方式的不同，这些算法大致可以分为基于内容的推荐、协同过滤推荐、基于上下文的推荐以及混合推荐算法几类，如图 2.1 所示。本节将对这些推荐算法进行简单的介绍。

2.1.1 基于内容的推荐

基于内容的推荐(Content-based Recommender Systems)主要是向用户推荐与其过去喜欢的物品具有相似特征的新物品，比如用户几天前刚在亚马逊上购买了吴军博士的《数学之美》，那么系统会据此向其推荐吴军博士的另一本著作《浪潮之巅》。一般来说，基于内容的推荐主要包括以下三个阶段：

1. 抽取物品的特征。比如，歌曲特征包括类型、创作者、创作年代等属性。

2. 抽取用户的特征。用户的特征既包括其对应的人口学特征，比如年龄、性别、职业等，也包括从其过去购买、评分和喜欢等行为中提取的特征。

3. 根据用户的特征向其推荐与其特征匹配的新物品。比如，向老年用户推荐经典老歌《东方红》，向喜欢怀旧的用户推荐《时间都去哪儿了》。

基于内容的推荐往往能够为用户做出正确的推荐，而且能够在推荐的同时给出关于推荐的解释，这有助于提升推荐的效果。但是，准确全面的抽取物品的特征往往是很困难的，一方面这需要较强的专业领域知识，比如音乐推荐系统 Pandora 发动专家为音乐进行特征提取，这种方法耗时耗力，不具可扩展性。另外，物品的特征往往是不可完全列举的，这在一定程度上限制了基于内容的推荐算法的发展。

2.1.2 基于协同过滤的推荐

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化环境，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。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。如图 2.2 所示，系统试图为用户 C 做出推荐，发现用户 D 和用户 C 都喜好图片和书籍且都不喜欢游戏，那么系统认为用户 D 和用户 C 是相似用户，参考用户 D 不喜欢视频，那么系统认为 C 也不喜欢视频。

与传统的基于内容的推荐方法相比，协同过滤推荐具有领域无关的特点，因此能够方便的推广应用到各个领域，比如音乐、电影、电子商务等。另外，协同过滤推荐充分考虑了群体智能在推荐中的应用，因此往往能够取得比较高的推荐准确率。上文所述的协同过滤称之为基于用户的协同过滤(user-based CF)，即通过计算用户之间的相似度来进行推荐，也可以通过计算物品之间的相似度来做出推荐，这称之为基于物品的协同过滤(item-based CF)，其执行过程与 user-based CF 类似，只是考虑到物品的相对静止特点，其做推荐时不需要繁杂的线上计算，因此也得到了广泛的应用。

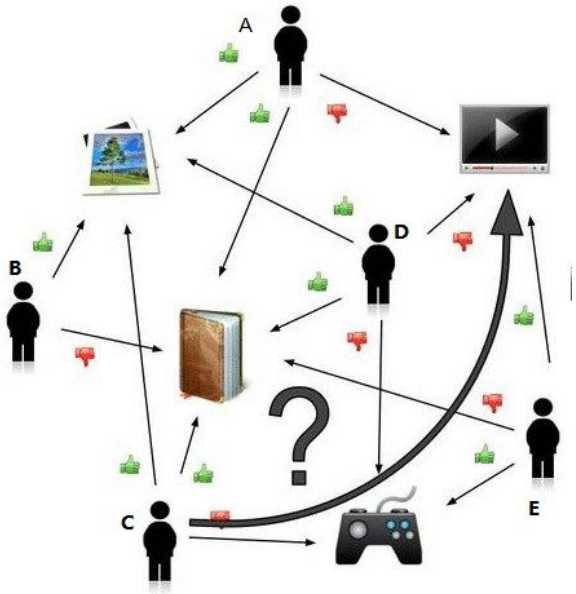


图 2.1 基于协同过滤的推荐示意图

2.1.3 基于上下文的推荐

上述的三种推荐方法主要着眼于为用户推荐与其偏好或者需求最相似的物品，却很少考虑用户所处的上下文信息，如时间、地点、天气等。换句话来说，传统的音乐推荐算法主要处理用户和物品两类实体，但并没有将它们放到一定的上下文环境中去分析。然而，很多物品对上下文环境比较敏感，比如对时间信息敏感的旅游产品，对情感信息比较敏感的电影、音乐以及对地点信息比较敏感的餐馆等，因此将上下文信息融入到为特定环境下的用户推荐物品中是非常重要的。比如，可以为一个刚刚失恋、心情低沉的用户推荐悲伤的歌曲，而对刚刚毕业、心情舒畅的用户推荐欢快的歌曲。基于上下文的推荐(Context-aware Recommender System,CARS)就是这样一种推荐算法，其通过一定的手段收集用户所处的上下文环境信息，进而分析用户、物品、上下文环境这三个实体，然后将与用户所处上下文环境匹配的物品推荐给用户。随着移动互联网的发展，用户所处的时间、地点、状态等信息可以比较容易地通过传感器获得，因此基于上下文的推荐获得了比较长足的发展，比如各种典型的 LBS 应用。

2.1.4 混合推荐方法

上述的几类推荐方法各有各的优势，但也各有各的劣势，如表 2.1 所示。混合推荐方法就是这样一类推荐方法，其将不同推荐方法综合利用，以规避单一推荐方法的缺点，充分利用单一推荐方法的优点。比如，基于内容的推荐与协调过滤推荐的混合推荐方法、基于内容的推荐与协调过滤推荐的混合推荐方法、基于上下文的推荐与协同过滤的推荐、基于知识的推荐与基于内容的推荐等。本文所述的基于多维时间序列分析的个性化音乐推荐算法本质上也是一直基于内容的推荐和基于上下文的推荐相关联的推荐方法。

表 2.1 不同类型推荐算法总结

名称	优点	缺点
基于内容的推荐	用户独立、可解释	领域相关
协同过滤推荐	群体智慧、领域无关	冷启动、长尾效应
基于上下文的推荐	考虑上下文环境的影响	上下文获取有一定难度

2.2 常用相似度衡量标准

如前节所述，无论是哪一种推荐算法，都需要进行相似度的计算，本节简单介绍一些常用的相似度度量标准。

2.2.1 余弦相似度

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的夹角的余弦值来度量它们之间的相似性。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。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。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 $\mathbf{X_a}$ 和向量 $\mathbf{X_b}$ 分别表示用户 a 和用户 b 的偏好向量，其中 $\mathbf{X_a}$ 中的元素 Xa(i)表示用户 a 对编号为 i 的物品的偏好值或评分，$\mathbf{X_b}$ 中的元素 $X_b(i)$表示用户 b 对编号为 i 的物品的偏好值或评分。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。

$$\mathrm{sim}(a,b)=\cos(\mathbf{X_a},\mathbf{X_b})=\frac{\sum_{i=1}^{K}X_a(i)X_b(i)}{\sqrt{\sum_{i=1}^{K}X_a(i)^2}\sqrt{\sum_{i=1}^{K}X_b(i)^2}} \tag{2.1}$$

2.2.2 KL 距离

KL 距离也称相对熵或 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。如果能够把物品表示称一个概率分布，那么显然可以用 KL 距离来衡量两个物品之间的相似度。设 $P=(p_1,...,p_i,...p_k)$ 表示物品 p 对应的概率分布，物品 q 对应的概率分布用 $Q=(q_1,...,q_i,...q_k)$ 表示，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。KL 散度仅当概率 P 和 Q 各自总和均为 1，且对于任何 i 皆满足 $p(i)>0$ 及 $q(i)>0$ 时才有定义，若式中出现 $0\ln 0$ 的情况，其值按 0 处理。

$$\mathrm{dis}_{KL}(p,q)=D_{KL}(P\|Q)=\sum_i p(i)\ln\frac{p(i)}{q(i)} \tag{2.2}$$

由于 KL 距离是非对称的，即

$$dis_{KL}(p,q) \neq dis_{KL}(q,p) \tag{2.3}$$

然而，物品之间的相似度应该满足对称性，即

$$sim(p,q) = sim(q,p) \tag{2.4}$$

因此，不能直接使用 KL 距离来计算物品之间的相似度，但我们可以使用 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，这样就可以满足对称性的要求，如公式 2.5 所示。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。

$$dis(p,q) = \frac{dis_{KL}(p,q) + dis_{KL}(q,p)}{2} \tag{2.5}$$

$$sim(p,q) = \frac{1}{1 + dis(p,q)} \tag{2.6}$$

2.2.3 Hellinger 距离

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，其天然满足对称性，因此可以直接用来计算物品之间的相似度。物品 p 和物品 q 之间的 Hellinger 距离可以按照公式 2.7 计算，进而按公式 2.8 可以计算二者之间的相似度。

$$dis(p,q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} \left(\sqrt{p(i)} - \sqrt{q(i)}\right)^2} \tag{2.7}$$

$$sim(p,q) = \frac{1}{1 + dis(p,q)} \tag{2.8}$$

本文所提的基于时间序列分析的音乐推荐框架将使用主题模型建模的方法将每一首歌曲表征为若干隐含主题的概率分布。因此，KL 距离和 Hellinger 距离将被应用在计算两首歌曲之间的相似度上，而且能够得到比余弦相似度、Pearson 距离等方法更优的结果。

2.3 评测指标

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。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。所谓评分预测问题即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

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。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。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”按钮进行显示的统计。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。

显然，用户满意度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。

2.3.2 预测准确度

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。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，方便了很多学术界的研究人员研究推荐算法。

对于评分预测问题，一般使用过均方根误差(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。RMSE 可由公式 2.9 计算得到。

$$\mathrm{RMSE}=\sqrt{\frac{\sum_{u,i\in T}(r_{ui}-\hat{r}_{ui})^2}{|T|}} \tag{2.9}$$

其中，T 代表测试集，u 和 i 表示测试集中的用户和物品，r_{ui} 是用户 u 对物品 i 的实际评分，而 $\hat{r}_{ui}$ 是推荐算法给出的预测评分。MAE 采用绝对值计算预测误差，如公式 2.10 所示。

$$\mathrm{MAE}=\frac{\sum_{u,i\in T}|r_{ui}-\hat{r}_{ui}|}{|T|} \tag{2.10}$$

显然，预测误差越小，那么预测准确度越高，那么推荐算法的效果越好。也就是说，一个优秀的推荐算法在 RMSE 和 MAE 这两项指标上的取值往往比较低。

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。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。那么，推荐结果的准确度和召回率如公式 2.11 和 2.12 所示.

$$\text{Precision} = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|} \tag{2.11}$$

$$\text{Recall} = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \tag{2.12}$$

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。其中，覆盖率(Coverage)描述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。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大。多样性(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。

由于预测准确度与覆盖率、多样性相比更能够衡量一个推荐算法的优劣，本文后面主要使用预测准确度来评估本文所提的推荐方法及框架，即评估本文所提推荐方法和框架是否能够得到比其他算法更高的准确度和召回率以及更低的预测误差。

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极大的提升，但其在应对大数据时代产生的海量数据时仍然非常吃力。为了解决大数据时代海量数据的处理和分析的问题，Google 提出了一种分布式的计算模型，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 年开发了目前得到广泛应用的分布式应计算框架 Hadoop。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，但其对一些在实时性方面要求比较高的计算任务的处理能力有所欠缺。为了弥补 Hadoop 的这一缺憾，以 Storm 为代表的一些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。这里，Storm 是由 Twiiter 开发的一款开源的分布式实时计算框架，其主要适用于流数据处理和分布式远程过程调用两种场景。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，本节将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 Storm 的基本组成

一个传统的 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。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，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。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。这里，计算任务 Topology 是由多个 Spout 和 Bolt 计算组件构成，而这些计算组件之间是通过数据流连接起来的。其中，Spout 是 Storm 中的消息源，用于为 Topology 生产消息（数据），一般是从外部数据源（如 Message Queue、RDBMS、NoSQL、Realtime Log）不间断地读取数据并发送给 Topology；Bolt 是 Storm 中的消息处理者，用于为 Topology 进行消息的处理，Bolt 可以执行过滤，聚合，查询数据库等操作且可以一级一级的进行处理。Topology 中每一个计算组件（Spout 和 Bolt）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 2.5 是 Twitter Storm 官方给出的一个典型的 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，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。

除此之外，Storm 还有 Stream、Stream Grouping、Task、Worker 等关键概念。

其中，Stream 表示被处理的数据；Stream Grouping 表示 Bolt 接收什么样的数据作为输入数据，即上游组件以何种方式将消息或数据分组并传给下游组件，一般有广播(All Grouping)、随机分组(Shuffle Grouping)、按字段分组(Field Grouping)、直接分组(Direct Grouping)以及不分组(No Grouping)等分组方式；Task 表示运行于 Spout 和 Bolt 中的线程，而 Worker 表示运行这些线程的进程。在具体实现时，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的分组方式即可轻松地完成并发布一个实时计算的任务。

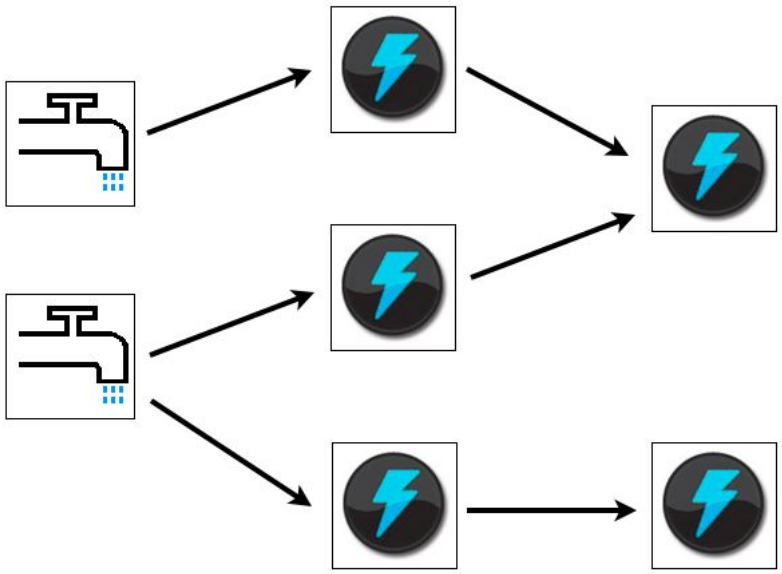


图 2.2 典型的 Twiiter Storm 示意图

2.4.2 DRPC 的使用

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。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；
4. 将计算结果返回给客户端。

图 2.6 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. 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. 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。

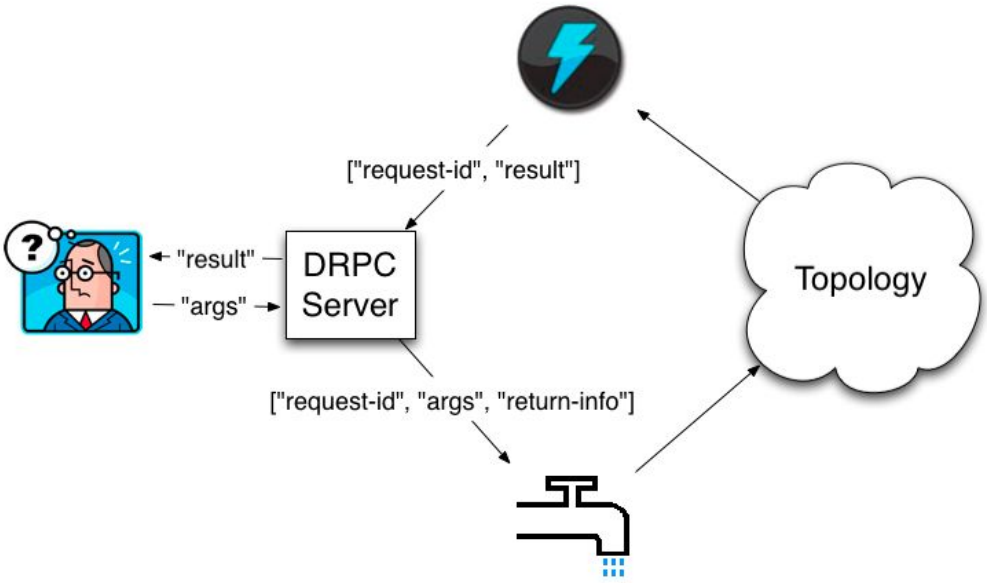


图 2.3 DRPC 的工作流程示意图

2.5 文本建模

为了对歌曲进行全面且完整的描述和刻画，本文将使用文本分析的方法对表征歌曲的文本信息进行分析，本节简单介绍一些基本的文本建模方法。

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)。向量空间模型首先将每一个文档看做一个词袋(Bag of Words)，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。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，而向量每一维的取值表示该词项在文档中的权重。对于文档集合 D 中编号为 j 的文档 d_j，可以将之表示成一个 t 维的向量 $d_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$，其中 t 表示词项的数目，$w_{i,j}(1 \le i \le t)$ 表示第 i 个词项在文档 d_j 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。这是最为简单直观的一种计算词项权重的方法，即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。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。与布尔模型不同的是，词频(Term Frequency, TF)模型统计词项在文当中出现的次数，然后得到词项的频率，并将之作为词项的权重。词项 t_i 相对于文档 d_j 的词频表示成 $tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$，这里 $n_{i,j}$ 表示该词项在该文档中出现的次数。这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。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。也就是说，一个词语在整个语料库中出现得越频繁，则它对于文档的重要性越低，对文档的区分度量越差。

（3）词频-逆向文本频率模型。词频-逆向文本频率(Term Frequency - Inverse Document Frequency，TF-IDF)是对上述 TF 模型的补充，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，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。词项 t_i 相对于文档 d_j 的 TF-IDF 取值表示为 $tfidf_{i,j} = tf_{i,j} \times idf_i$。这里，$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$ 表示逆向文本频率，其中|D|表示文档集合中的文件总数，$|\{j : t_i \in d_j\}|$ 表示文档集合中包含词项 t_i 的文件数目。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。

2.5.2 隐含狄利克雷分配模型

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)。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。进一步地，LDA 被看做是一种主题模型，它将文档看做是一组隐含主题的概率分布。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。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。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。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。

在 LDA 模型中，记文档-主题的概率分布为多项式分布$\vec{\theta}$，主题-词汇的概率分布为多项式分布$\vec{\varphi}$。此外，LDA 模型认为$\vec{\theta}$和$\vec{\varphi}$是模型中的参数，且二者都是随机变量，考虑到$\vec{\theta}$和$\vec{\varphi}$都是多项式分布，模型选择狄利克雷(Dirichlet)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，如下所示的吉布斯取样 (Gibbs Sampling) 是目前比较流行的采样方法：

1. 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$z_{m,n}=k\sim Mult(1/K)$，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$n^{(k)}_m$、n_m、$n^{(t)}_k$、n_k都加 1, 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. 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 $n^{(k)}_m$、n_m、$n^{(t)}_k$、n_k 都减 1，即先拿出当前词，之后根据 LDA 中 Topic Sample 的概率分布取样出新的主题，再将 $n^{(k)}_m$、n_m、$n^{(t)}_k$、n_k 都加 1。其中，α 和 β 为对应的 Dirichlet 分布的参数，V 为词汇总数。

$$p(z_i = k \mid z_{-i}, w) \propto \frac{(n^{(t)}_{k,-i} + \beta_t)(n^{(k)}_{m,-i} + \alpha_k))}{\sum_{t=1}^{V}(n^{(t)}_{k,-i} + \beta_t)} \tag{2.13}$$

3. 重复步骤 2 直至遍历所有文档。

4. 输出 LDA 模型中的参数 $\vec{\theta}$ 和 $\vec{\varphi}$。

$$\varphi_{k,t} = \frac{n^{(t)}_k + \beta_t}{\sum_{t=1}^{V} n^{(t)}_k + \beta_t} \tag{2.14}$$

$$\theta_{m,k} = \frac{n^{(k)}_m + \alpha_k}{\sum_{k=1}^{K} n^{(k)}_m + \alpha_k} \tag{2.15}$$

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，因此我们将采用 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的形成序列，且是这种序列有严格的时间顺序，本文工作将通过对用户在会话期内所收听歌曲形成的时间序列的分析来预测用户的接下来的行为。因此，本节将简单介绍一些常用的时间序列预测模型。

2.6.1 简单平均法

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，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。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，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。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

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。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。首先，它把过去的数据全部加以利用。其次，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。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。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。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。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。

2.6.4 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，是一种由 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。在模型 ARIMA(p,d,q)中，“AR”表示自回归模型，p 为自回归阶数，“MA”表示滑动平均模型，q 表示滑动平均的阶数，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。显然，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。进一步的，模型 ARIMA(p,d,q)可以表示成如下三个式子。

$$\Phi(B)(1-B)^{d}y_t=\delta+\Theta(B)\varepsilon_t \tag{2.16}$$

$$\Phi(B)=1-\sum_{i=1}^{p}\varphi_i B^i \tag{2.17}$$

$$\Theta(B)=1+\sum_{i=1}^{q}\theta_i B^i \tag{2.18}$$

其中，y_t 为时间序列 Y 的第 t 个取值，B 是滞后算子，φ 和 θ 为模型参数，它们可以通过最小二乘等方法获得。

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。其中，模型识别阶段主要完成检测序列是否平稳的工作。如果序列不平稳，那么则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。在此基础上，识别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。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φ和θ。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 模型将应用在本文后续的工作中，已完成对用户行为序列的分析和预测。后续实验结果显示，使用 ARIMA 模型进行时间序列预测，能够得到较好的效果。

2.7 音乐推荐算法

文献[]给出了一种简单朴素的音乐推荐算法，其引导用户通过歌曲名、曲作者姓名以及歌词等文本信息去检索歌曲。文献[]首先抽取歌曲的声音的音色、节奏等声学特征，然后通过比较用户收听歌曲的声学特征与曲库中歌曲声学特征为用户做出推荐。文献[]是典型的基于内容的音乐推荐，能够快速地为用户推荐符合偏好的歌曲，但它们具有较强的领域相关特点，可扩展性不强而且耗时耗力。文献[]给出了一种基于群体行为的推荐方法，即首先分析用户的收听行为然后找到与当前用户相似的用户并进而根据相似用户的行为为用户做出推荐，这是典型的协同过滤推荐。这种方法充分利用了用户的从众心理，能够得到较好的推荐效果。同时，其具有领域无关性，解决了基于内容的推荐所面临的问题。然而，这种推荐方法容易遇到冷启动和长尾效应等问题，一方面无法很好地为新用户做出推荐，另一方面往往推荐的是一些比较流行的歌曲而导致长尾歌曲无法被推荐。考虑到歌曲具有强烈的情感色彩，文献给出了一种基于情绪的音乐推荐算法，首先引导用户选择自己当前的情绪状态，然后从曲库中选择与用户选择情绪相近的歌曲。这种推荐方法能够很好地为用户做出符合当前情绪状态的推荐，但是情绪的描述和定义往往是很困难的，因此这种推荐的结果有时不太准确。此外，其需要用户的参与，在一定程度上增加了用户的成本。

上述给出的几种音乐推荐算法从基本属性、声学特征、情绪等不同侧面对歌

曲进行了描述和刻画，进而根据用户的喜好检索歌曲并作出最终推荐。然而，这些算法往往只考虑了歌曲一方面的特征而忽视了其他方面的特征，比如[]考虑了声学特征却忽略了情绪特征，而[]考虑了情绪特征却忽略了声学特征。此外，歌曲还具有风格、年代、语言、场合等不同的侧面，这些都没有得到很好的体现。另外，一首歌曲往往并不是确定地属于某一种类别，而是以不同的概率隶属于不同的类别。比如，张雨生的歌曲“大海”即属于“经典”也属于“怀旧”还属于“流行”，而上述的几种算法都没有对歌曲的这种特质进行体现。为了较为全面地描述歌曲特征并体现歌曲隶属类别的不确定性，文献[]充分利用了社会化标签系统的成果，首先利用用户对歌曲所打的标签构造歌曲对应的文档，继而通过主题模型建模的方法将歌曲表示成一组隐含主题的概率分布。

歌曲具有很强的情感色彩，用户收听歌曲的行为与其所处的天气、位置、场合等上下文环境有十分密切的关系，但直接获取这些上下文环境往往比较困难或者代价比较大。歌曲具有时间短、消费代价低的特点，因此用户往往会一次性地收听多收歌曲，而这些歌曲就天然形成了一个序列。文献[][]认为歌曲序列能够在一定程度上代表用户所处的上下文，进而通过对序列的分析来预测用户接下来的行为。它们首先利用[]中的思想将歌曲表征成若干隐含主题的分布，然后选择隶属概率较大的几个主题作为显著主题来表征歌曲，这样便可以把歌曲序列表示成显著主题序列。得到显著主题序列数据库后，[][]分布通过模式匹配和马尔科夫链进行分析并预测用户可能收听的下一首歌曲所述的显著主题并将该主题中的显著歌曲推荐给用户。实验表明，文献[][]能够获得比较好的推荐效果。然而，文献[][]同样需要参考其他用户的行为，荣誉遇到冷启动问题和长尾效应。此外，它们只考虑了显著主题的贡献而忽略了其他主题的影响。最后，它们是通过定性的方法预测用户可能收听的下一首歌曲的主题，没有充分考虑用户行为的时序性。本文将给出一种基于多维时序分析的音乐推荐方法，该方法也利用文献[]中的思想将歌曲表示成若干隐含主题的分布，然后利用时间序列预测的方法逐一预测每一个主题隶属度序列的取值。这样我们便可以得到用户可能收听的下一首歌曲的概率分布，通过计算相似度我们便可以得到候选的推荐列表。

第三章 一种基于时间序列分析的音乐推荐方法

本章将提出一种基于多维时间序列分析的音乐推荐方法。首先，我们将介绍本文所研究问题的相关描述和定义。其次，我们给出一种基于多维时间序列分析的音乐推荐方法，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。最后，我们介绍基于所提方法所进行的实验，包括实验的设计、结果和分析。

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(注:本文所考察的是用户在一定会话周期内的行为)，如图 3.1 所示[14]。

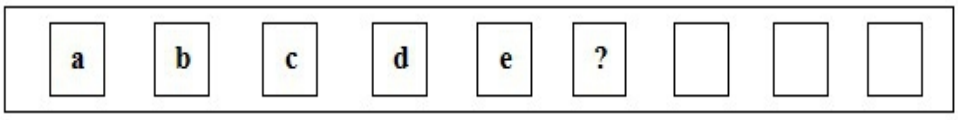


图 3.1 下一首歌曲预测问题的说明图

为了更好地描述该问题，我们首先定义用户集和歌曲集，如下所示：

用户集 C: 所有用户的集合,如公式 3.1 所示所示,其中 m 为用户的数目,即 m=|C|且 m>0。

$$C = \{c_1, c_2, ..., c_m\} \tag{3.1}$$

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，如公式 3.2 所示，其中 v 为所有可推荐歌曲的数目，即 v=|S|。为方便起见，我们认为用户收听的歌曲一定在“可推荐歌曲”中。

$$S = \{s_1, s_2, ..., s_v\} \tag{3.2}$$

对于给定的一个用户 c，音乐推荐系统的目标就是为这个用户推荐其可能喜

欢的下一首歌曲。为了衡量用户对歌曲的喜欢程度，我们定义效用函数如下：

效用函数 u(c,s)： 表征歌曲 s 对用户 c 的推荐度(如歌曲 s 符合用户 c 喜好的程度、歌曲 s 与预测歌曲的相似度等)。

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，本文的方法建立在隐含主题分类和用户在一定会话周期内听歌序列的基础上，我们下面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： 由所有隐含主题组成的集合，如式公式 3.3 所示，其中 K 为隐含主题的数目。

$$T = \{t_1, t_2, ..., t_K\} \tag{3.3}$$

事件 $e(c,\tau,s)$**：** 表示用户 c 在时刻 τ 收听了歌曲 $s \in S$，显然 s 可由 c 和 τ 唯一决定，因此 $e(c,\tau,s)$ 可简化为 $e(c,\tau)$。

序列 Q(c)： 代表用户 c 在一定会话周期内的所有听歌事件按时间先后顺序排列而成的列表，公式 3.4 所示，其中 n 为用户 c 所收听或喜欢的歌曲数目，τ 为事件发生的时间，且 $\tau_1 < \tau_2 < ... < \tau_i < \tau_{i+1} < ... < \tau_n (1 \le i \le n)$。

$$Q(c) = < e(c,\tau_1), e(c,\tau_2), ..., e(c,\tau_n) > \tag{3.4}$$

会话 Session: 本文所考察的歌曲序列均是用户在一个会话周期之内的行为，即用户连续不中断的收听行为。我们定义会话为满足公式 3.5 式的序列 Q(c)。其中，ε 为最长时间间隔，本文实验将其设为 480 秒。方便起见，本文依然用 Q(c) 来代表用户 c 的一个特定的会话。

$$|\tau_i - \tau_{i-1}| \le \varepsilon \quad (1 < i \le n) \tag{3.5}$$

目标歌曲： 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 和用户某个会话下的收听序列 Q(c)，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 c 收听序列 Q(c) 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(c) 并推荐给用户，如公式 3.6 所示。

$$N(c) = \underset{s \in S}{\operatorname{argmax}}\, u(c,s) \qquad (3.6)$$

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，实线箭头表示在线处理模块。首先，为了向主题模型建模方法提供输入，本文方法将定期从音乐网站上抓取以用户标签为主的歌曲信息，然后将这些标签信息按照一定的方式组合成文档，那么这些文档将和歌曲集 S 中的歌曲一一对应。随后，我们对这些文档组成的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多维的表示隐含主题权重的向量表示，向量中的每一维取值表征该主题对歌曲内容的贡献程度。另外，由于用户听歌事件之间有严格的时间先后关系，我们将用户所听歌曲构成的序列建模为一个以歌曲对应的每一维隐含主题权重为变量的多维时间序列。进一步地，通过对该多维时间序列分析和预测，我们得到用户下面可能收听歌曲所对应的隐含主题权重的向量。最后，我们将曲库中与预测歌曲最相似的歌曲推荐给用户。在下面的章节中,我们将对框架中的主要步骤进行一一介绍。

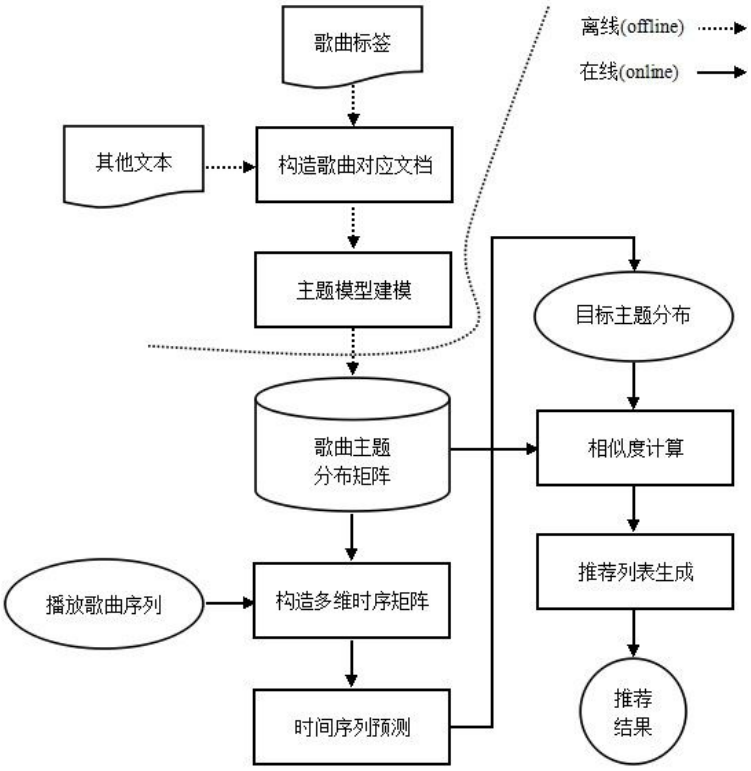


图 3.2 基于时间序列分析的音乐推荐方法框架示意图

3.3 主题模型建模

如前所述，为了更好地刻画歌曲的特征，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和刻画。隐含狄利克雷分配(Latent Dirichlet Allocation,LDA)模型[15]是当前最具代表性也是最流行的概率主题模型，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[16,17]，其能够将文档映射到若干隐含主题的分布，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该维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。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 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，每一首歌曲 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。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的向量，如式公式 3.7 所示。

$$\mathbf{s} = (w_1, w_2, ..., w_i, ..., w_K) \quad (1 \le i \le K) \tag{3.7}$$

其中，当$w_i = 0$时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；当$w_i = 1$时表示歌曲完全属于该隐含主题代表的类别；当$0 < w_i < 1$时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 的若干显著主题及其隶属于这些主题的概率。由表 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 75 对歌曲的贡献度为 0.028，这说明主题 75 对该歌曲的贡献度为 0.028。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 的显著主题及隶属度

主题 id	508	313	296	106	231	100	307	354	8	75
隶属度	0.215	0.186	0.134	0.084	0.072	0.072	0.057	0.030	0.030	0.028

3.4 时间序列的构造和预测

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，较少考虑用户行为所形成的序列性趋势，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对应的在一定会话周期内收听的歌曲构成的序列并构造其对应的时间序列，然后使用相应的时间序列分析方法进行预测。

设用户 c 当前会话周期内收听的歌曲序列长度为 n，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 3.8 所示。

$$\mathbf{N}(c) = (w_1(n+1), w_2(n+1), ..., w_i(n+1), ..., w_K(n+1)) \tag{3.8}$$

其中，$w_i(j)$表示第 j 首歌曲隶属于第 i 个隐含主题的概率$(1 \le i \le K)$。换句

话说，我们需要准确地预测对应的主题权重向量中每一维的值，从而能够在歌曲集 S 中找到最匹配的歌曲推荐给用户。由前文可知，用户 c 的听歌事件按照事件发生的先后顺序构成了一个与时间相关的序列 Q(c)，且每一个听歌事件中对应的歌曲可以表示成如式 3.7 所示的 K 维隐含主题权重向量，那么我们将 Q(c)按照式 3.7 展开，可以得到如式 3.9 所示的矩阵 M(c)。

$$\mathbf{M}(\mathrm{c})=\begin{bmatrix} w_1(1) & w_1(2) & \cdots & w_1(\mathrm{n}) \\ w_2(1) & w_2(2) & \cdots & w_2(\mathrm{n}) \\ \vdots & \vdots & \ddots & \vdots \\ w_K(1) & w_K(2) & \cdots & w_K(\mathrm{n}) \end{bmatrix} \tag{3.9}$$

其中，M(c)的行向量表示用户 c 对应的事件序列 Q(c)在各个隐含主题上的展开，而矩阵的列向量即为用户 c 收听事件序列中每一个事件对应的歌曲的隐含主题权重向量。考察如式 3.10 所示的 M(c)的第 i 个行向量对应的序列。

$$q_i =< w_i(1), w_i(2), ..., w_i(n) > \tag{3.10}$$

其中，$w_i(j)$ 的含义如前所述。显然，w_i 是一个随着时间而变化的实数变量，因此可以将 q_i 看作一个以 w_i 为时间变量的长度为 n 的时间序列。进一步地，我们可以利用时间序列分析和预测的方法对该时间序列进行分析和预测，从而可以估算第 i 个隐含主题对用户可能收听的下一首歌曲的贡献程度，即 N(c)对应的主题权重向量中第 i 维元素值的估计值。

我们可以将 M(c)看做是一个多维的时间序列，其每一个行向量对应一个维度，共计 K 个维度。然后，我们对这 K 个维度的时间序列逐一利用 ARIMA 模型进行分析并预测该维度的估计值。最终我们可以获得如式 3.11 所示各个维度的估计值。显然，该向量可以看作是用户下面可能收听的歌曲在 K 个主题上的近似分布。为简便起见，我们将该分布对应的歌曲记为 $\hat{s}$。

$$\hat{\mathbf{N}}(\mathrm{c}) = (\hat{w}_1(\mathrm{n}+1), \hat{w}_2(\mathrm{n}+1), ..., \hat{w}_i(\mathrm{n}+1), ..., \hat{w}_K(\mathrm{n}+1)) \tag{3.11}$$

3.5 歌曲相似度计算及其推荐

通过 LDA 主题建模我们得到了如式 3.7 所示歌曲集 S 中歌曲对应的主题权重向量，考虑到向量中每一维度的值表征歌曲隶属于某一隐含主题的概率，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 3.11 所示用户 c 可能收听的第(n+1)首歌曲对应的主题权重向量的估计向量，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$\hat{s}$距离足够小，那么我们就可以将歌曲 s 推荐给用户 c。因此，我们可以用 s 与$\hat{s}$的距离的倒数表示歌曲集中任一歌曲 s 对用户 c 的推荐度，即上文定义的效用函数 u(c,s)，如式 3.12 所示。

$$u(c,s)=\frac{1}{1+dis(s,\hat{s})} \tag{3.12}$$

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)[19]以及 Hellinger 距离[20]等来度量两首歌曲之间的距离。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 3.13 所示。

$$dis(s_i,s_j)=\frac{1}{\sqrt{2}}\sqrt{\sum_{k=1}^{K}(\sqrt{w_{ik}}-\sqrt{w_{jk}})^2} \tag{3.13}$$

其中，$dis(s_i,s_j)$为歌曲s_i和s_j对应的主题概率分布之间的 Hellinger 距离，K 为隐含主题数目，w_{ik}为歌曲s_i在第 k 个隐含主题上的概率。显然，当两首歌曲越相似，那么其对应的距离越小。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。

最后，我们根据效用函数计算 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，这样我们就为用户 c 推荐了一个长度为 N 的歌曲列表供用户选择。

3.6 实验设计和结果

本节将给出我们实验的设计思路、实验结果以及结果分析。

4.6.1 实验数据收集

通过对基于多维时间序列分析的音乐推荐算法的分析，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的歌曲序列的数据集。虽然 Berenzweig 等人于 2003 年从 Art of the

Mix(http://www.artofthemix.org/)上抓取了播放列表数据集[23]，但是其存在如下三个问题：

(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 Last.fm 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。

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列表)。为了消除噪音，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下表所示，目前该数据集已经发布在 http://lastfmseq.sinaapp.com/上，而其使用说明如附录 1 所示。

听歌事件	34930(个)
不同歌曲	24992(首)
不同列表	1530(个)
不同歌手	5479(位)
最小长度	10
最大长度	30
平均长度	22.83

4.6.2 实验评测标准

对于用户 c，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 c 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)。由文献[1][21]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。记 R(c)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(c)是用户在测试集上的行为列表，那么表征“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(18)所示：

$$\mathrm{Recall} = \frac{\sum_{c \in C} |R(c) \cap T(c)|}{\sum_{c \in C} |T(c)|}$$

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 c 同一时刻在测试集上只会收听一首歌曲，即|T(c)|=1。因此，我们将式(18)简化为如式(19)所示的命中率(hit ratio)：

$$h(N)=\frac{\sum_{c\in C}hit}{|C|}$$

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 c 实际收听的歌曲为 s，则 hit 可表示为如式(20)：

$$hit=\begin{cases}1 & ,s\in R(c)\\ 0 & ,s\notin R(c)\end{cases}$$

准确率表征了“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(21)所示：

$$precision=\frac{\sum_{c\in C}|R(c)\cap T(c)|}{\sum_{c\in C}|R(c)|}$$

考虑到$|T(c)|=1$，式(21)可简化为式(22)：

$$precision=\frac{\sum_{c\in C}hit}{\sum_{c\in C}|R(c)|}$$

考虑到召回率和精确度此消彼长的关系，文献[21]中使用 F1-Score 对模型进行评估，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(23)表示：

$$F=2*\frac{recall*precision}{recall+precision}$$

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(19)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[22]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s_{next}，如果系统为其推荐了列表$L1=\{s_{11},s_{51},s_{31},s_{next},s_{21},s_{41}\}$，那么我们认为该推荐是有效的，因为目标歌曲在推荐列表中。相反，如果系统为其推荐列表$L2=\{s_1,s_5,s_3,s_6,s_2,s_4\}$，由于其中未包含$s_{next}$，我们认为该推荐是无效的。然而，如果$s_{next}$与 L2 中的歌曲相似度很高，用户显然也会喜欢该列表。那么，认为列表 L2 无效

就不够合理。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。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的 Hellinger 距离来表示。如果一个推荐列表中歌曲与目标歌曲的相似度较高，那么该列表中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。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(24)(25)所示。

$$RMSE = \sqrt{\frac{\sum_{c \in C} e(c)^2}{|C|}}$$

$$MAE = \frac{\sum_{c \in C} |e(c)|}{|C|}$$

其中，C 为测试用户集，|C|为用户数，e(c)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(c)看做是列表中所有歌曲与目标歌曲的平均距离。如式(26)所示。

$$e(c) = \frac{\sum_{s \in R(c)} dis(s, s_{next})}{|R(c)|}$$

4.6.3 实验设置

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。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 2 E6300，内存大小为 2G，硬盘空间 160G。实验所用编程语言为 Python2.7。目前，实验源码和实验结果已经上传至 https://github.com/wwssttt/GitRepo/tree/master/Python/experiment，具体的使用说明如附录 2 所示。

4.6.4 实验结果与分析

4 展示了当被推荐歌曲数目 N 由 1 到 100 的增长过程中，不同推荐算法的召回率的变化情况，这里包括基于用户的最近邻算法(UserKNN)[7]、基于模式挖掘的推荐算法(PatternMining)[4]、基于 1 阶马尔科夫链的推荐算法(1st-Markov)[4,12]、基于 3 阶马尔科夫链的推荐算法(3rd-Markov)[4,12]以及本文提出的基于多维时间序列分析的音乐推荐方法(MTSA)。其中，横坐标表示被推荐歌曲的数目，纵坐标表示算法的召回率。由图 4 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。

考虑到基于用户的最近邻算法(UserKNN)在同类工作中的召回率最高，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 在综合指标 F1-Score 上的提升效果(倍数)，如图 5 所示。其中，横坐标表示被推荐歌曲的数目，纵坐标表示算法 F1-Score 的提升倍数。如果纵坐标取值大于零，表明本文所述方法的 F1-Score 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 F1-Score 而且还有所下降。由图 5 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-Score 且提升幅度在 80%以上。随着推荐列表长度的增长，提升的幅度也继续增长(可达 150%)。

图 6 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(如左图 a 所示)和平均绝对误差(如右图 b 所示)的变化趋势。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。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，但这种上升幅度也是是非常小的。

综合以上实验结果可以看出，无论是从命中率的角度去考察算法的优劣，还是从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较好的效果。这验证了本文所提方法的合理性，说明在音乐推荐系统中使用隐含特征的方法刻画歌曲并结合对用户在一个会话周期内行为趋势的预测能够提高推荐的效果。

第四章 一个基于用户行为三元特征的混合推荐框架

第三章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，同时通过实验验证了其在一定条件下的有效性和合理性。然而，上文所给的推荐方法仍然存在一些问题。本章首先将对上文所提的方法进行分析，并在此基础上给出用户听歌行为的三元特征，进而我们给出一个基于用户行为三元特征的混合推荐框架。最后，我们通过实验验证该混合推荐框架能够得到比单一的基于多维时间序列推荐较优的效果。

4.1 问题分析

如第三章所述，基于多维时间序列分析的个性化音乐推荐算法主要是通过对用户在会话期内行为的分析来预测用户可能收听的下一首歌曲。显然，这要求会话期有一定的长度，这样才能够获得比较准确的预测。那么，第三章所述方法对于会话刚开始阶段的预测效果如何尚不可知。这里，我们从常识假设其对于会话刚开始阶段的推荐效果不如会话已经形成规模时。因为，会话刚开始时，歌曲序

列果断，序列包含的信息较少，不足以得到高效的推荐。本节将通过实验来验证我们的直观假设。

图中横坐标表示为用户推荐的列表长度，纵坐标表示推荐的命中率。由图可以看出，尽管随着推荐列表长度的增加推荐的效果也有所增加，但推荐的命中率明显低于会话形成时，甚至低于“最相似”这种最基本的推荐。推荐的准确度、F1 值、均方误差、绝对误差也能获得类似的结果，这里不一一列出。

由上面的结果可以看出，本节开始提出的假设是成立的，即第三章所述方法在会话刚形成时预测效果不佳。

除了上述问题之外，还存在预测误差偏大的问题。分析图可知，尽管该方法能够得到较低的预测误差，但误差的绝对值比较高。

综上所述，三种所述的方法有两个主要的问题需要解决：其一是对会话刚形成时的处理，其二是降低预测的绝对误差。

为了解决这两个显著问题，本章将给出一种基于用户行为三元特征的混合推荐框架。该框架再利用用户行为的序列性特征的基础上，进一步地挖掘用户行为的其他特征。本章下面将首先介绍用户行为的局部性特征和全局性特征，然后给出完整的混合推荐框架，最后通过实验验证框架的有效性。

4.2 用户行为的三元特征

为了更好地为用户做出推荐，我们对用户行为特征进行深度挖掘并提出用户行为的三元特征，即用户行为具有时序特征、局部特征和全局特征。其中，时序特征即用户收听行为具有时序性，通过对收听时序的分析可以在一定程度上预测用户接下来的行为，这在上文提出的基于多维时间序列分析的个性化音乐推荐算法中能够得到证明。本节将给出对用户收听行为局部特征和全局特征的描述并简单介绍其背后蕴含的心理学原理。

4.2.1 局部特征

众所周知，程序的执行具有局部性原理，即在一段时间内，程序的执行仅限于程序中的某一部分，这包括空间局部性和时间局部性。时间局部性是指如果程序中的某条指令一旦执行，则不久之后该指令可能再次被执行；如果某数据被访问，则不久之后该数据可能再次被访问。空间局部性是指一旦程序访问了某个存储单元，则不久之后其附近的存储单元也将被访问。考察空间局部性，这里我们

将“用户”看做“程序”，将“收听”看做“访问”，而将曲库中的歌曲按照相似度大小依次排开即越相似的歌曲离得越近并将“歌曲”看做“存储单元”，那么我们便可以将用户收听某一首歌曲的行为看做是程序访问某个存储单元的行为。类似的，我们认为用户收听歌曲的行为也具有局部性原理，即如果用户在某一时刻收听了某首歌曲，那么其在不久之后将会继续收听与该歌曲类似的歌曲，我们将这种特点称之为用户行为的局部特征。进一步地，我们做出如下假设：

1.用户的状态是稳定的 $\sum_{j=1}^{n} w(j,i)$

2.用户在短期内所听歌曲的特征保持不变或只有细微变化 $\hat{w}(n+1,i)=\frac{}{n}(1\le i\le K)$

我们这种假设的合理性除了可由程序执行的局部性原理类比获得之外，也可以由心理学的相关理论导出，我们这里介绍海德的平衡理论和注意的稳定性特征。

4.2.2 全局特征

上节介绍了本文基于心理学分析的基础上给出的用户行为具有局部特征的假设，本节将介绍用户行为的全局特征。心理学理论指出，人们的性格具有一定的倾向性，而用户长期的行为能够在一定程度上反映这种倾向性。为此，本文给出如下的用户听歌行为的全局特征假设：

用户长期收听的歌曲能够反映用户对歌曲的整体偏好

简单起见，我们仅用歌曲特征的平均值来标准用户对歌曲的偏好，那么上述假设可以具体描述为：

用户对歌曲的整体偏好可由其所收听的所有歌曲的特征的平均值表示。

$$\hat{w}(n+1,i)=w(n,i)(1\le i\le K)$$

4.3 基于用户行为三元特征的混合推荐框架

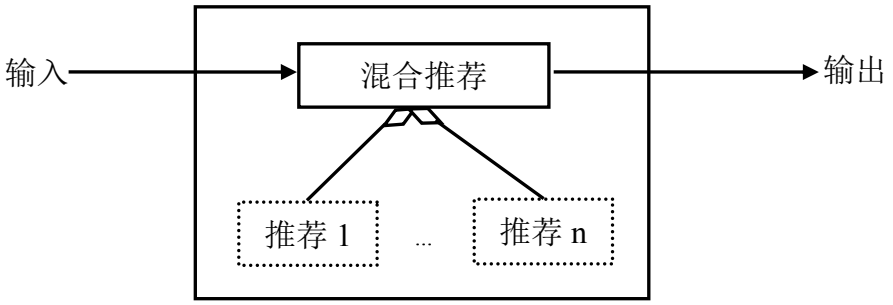
上文分别给出了用户听歌行为的局部性特征和全局性特征的假设，加上上一章描述的用户听歌行为的时序性特征，这些就构成了用户听歌行为的三元特征。为了描述方便，我们用 $\hat{w}_{sequential}(n+1,i)$ 表示根据时序特征预测的用户

可能收听的下一首歌曲在编号为 i 的隐含主题上的权重，$\hat{w}_{local}(n+1,i)$ 表示根据时序特征预测的用户可能收听的下一首歌曲在编号为 i 的隐含主题上的权重，$\hat{w}_{global}(n+1,i)$ 表示根据时序特征预测的用户可能收听的下一首歌曲在编号为 i 的隐含主题上的权重，而 $\hat{w}(n+1,i)$ 表示用户可能收听的下一首歌曲在编号为 i 的隐含主题上的权重。

由前文可知，单纯的基于时序特征的推荐存在预测误差偏大以及冷启动处理不佳的问题，本节给出基于三元特征的混合推荐框架，该混合推荐框架综合考虑用户行为的局部特征、全局特征和时序特征。混合推荐的主要特征就是混合设计，发挥各个独立算法的优势同时规避各个独立算法的缺点。尽管 Burke 的分类方法区分出了七种不同的混合策略，但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：整体式混合设计、并行式混合设计和流水线式混合设计，下文将对这三种设计思想进行简单介绍并给出本文所提混合推荐框架所采用的设计思想。

4.3.1 整体式混合设计

所谓整体式设计即将几种不同推荐策略整合到一个算法中实现的混合设计，实际上确实所有推荐策略都在发挥作用，如下图所示。举例来说，可以通过基于局部特征的推荐得到用户可能收听的下一首歌曲在 K 各隐含主题上的概率分布，而同样可以通过基于全局特征和基于时序特征的推荐得到用户可能收听的下一首歌曲在 K 各隐含主题上的概率分布。整体式推荐的一个简单实现就是在得到三个概率分布的基础上，混合得到最终的概率分布并以此

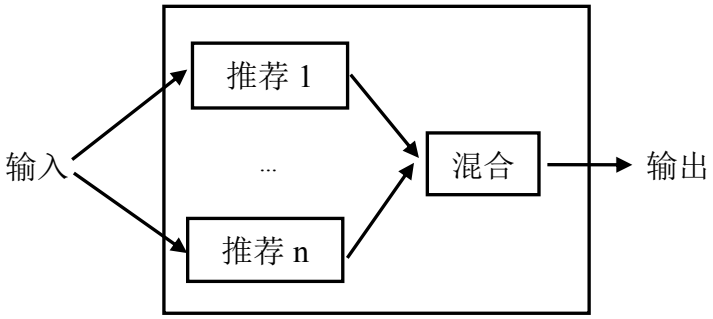


来产生推荐列表。

4.3.2 并行式混合设计

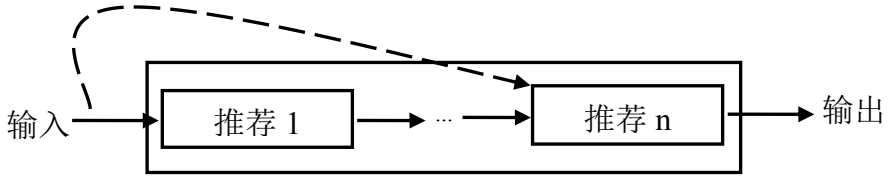
所谓并行式混合设计就是不同的推荐算法根据输入并行运行分布产生推荐列表，最后将这些推荐列表混合产生最终的推荐列表推荐给用户的设计思路，

如图所示。举例来说，并行式混合设计需要分别利用局部特征、全局特征和时序特征分别为用户生成一个推荐列表，然后将这三个列表进行混合以生成最终的推荐结果。至于混合的策略，可以采用加权、交叉以及切换等进行混合。交叉式混合推荐在在用户交互界面层面上将不同的推荐结果组合到一起，各种方法所得到的结果一同被呈现。加权式混合推荐关键字是加权，通过计算多个推荐结果分数的加权之和将其组织到一起。切换式混合需要根据用户记录或推荐结果的质量来决定哪种情况下应用什么推荐系统。



4.3.3 流水线式混合设计

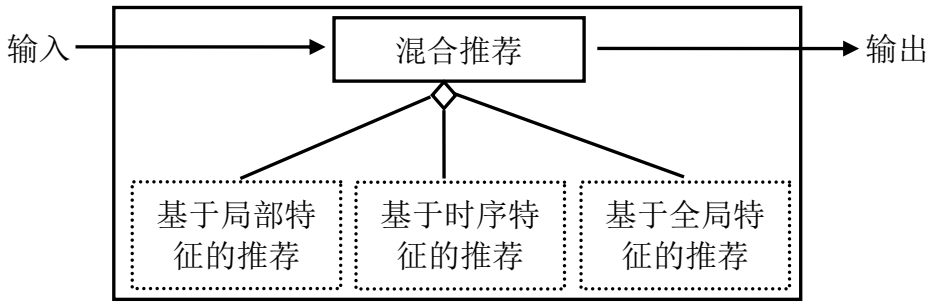
如下图所示，流水线式混合设计将多个推荐策略按照流水线架构连接起来，前一个推荐系统的输出变成后一个推荐系统的输入部分。当然，后面的推荐单元也可以选择使用部分原始输入数据。



4.3.4 混合方案的选取和分析

由上面的介绍可知，整体式混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混合，然后基于此做出最终推荐列表；并行式混合设计和流水线式混合设计都需要至少实现两个不同的推荐算法，其中并行式混合设计将不同推荐算法获取的最终推荐结果加以混合，而流水线式混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混合设计不太合适，而整体式混合设计和并行式混合设计均可应用，只是二者

混合的阶段不一致。由于整体式的混合设计思路是从内部对不同推荐策略的结果进行混合，我们认为这样的早期混合能够得到比较符合用户偏好的特征，因此我们主要考察这种混合设计方法，如图所示。



首先，我们考察将时序特征和局部特征进行混合，混合方法如下所示：

$$\hat{w}_{\text{seq\_local}}(n+1,i)=\alpha\hat{w}_{\text{sequential}}(n+1,i)+(1-\alpha)\hat{w}_{\text{local}}(n+1,i)$$

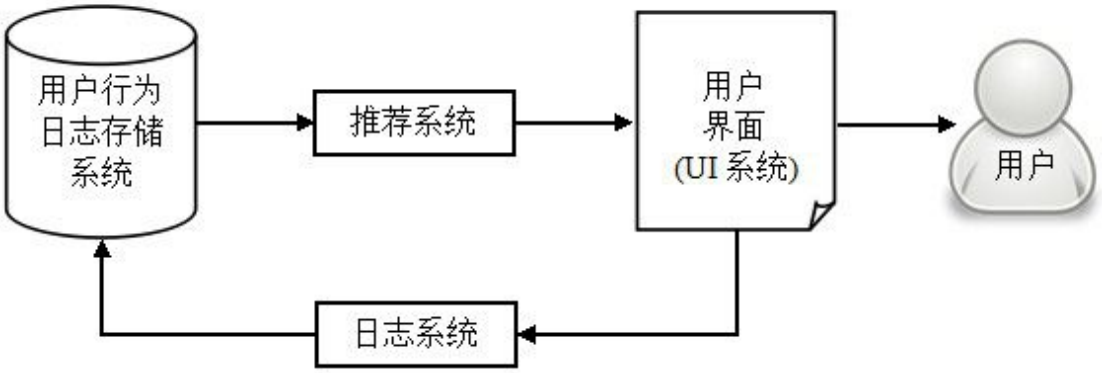
$$\hat{w}_{\text{seq\_global}}(n+1,i)=\alpha\hat{w}_{\text{sequential}}(n+1,i)+(1-\alpha)\hat{w}_{\text{global}}(n+1,i)$$

$$\hat{w}_{\text{all}}(n+1,i)=\alpha\hat{w}_{\text{sequential}}(n+1,i)+\beta(1-\alpha)\hat{w}_{\text{local}}(n+1,i)+(1-\beta)(1-\alpha)\hat{w}_{\text{global}}(n+1,i)$$

第五章　系统实现

前述章节介绍了本文所述的基于多为时间序列的个性化音乐推荐方法和基于用户三元特种的混合推荐框架，并从实验的角度验证了本文所提方法和框架的有效性。为了进一步验证本文所提方法和框架的可行性，本文实现了一个个性化音乐推荐原型系统，本章将详细介绍该系统的实现细节。

5.1 系统架构



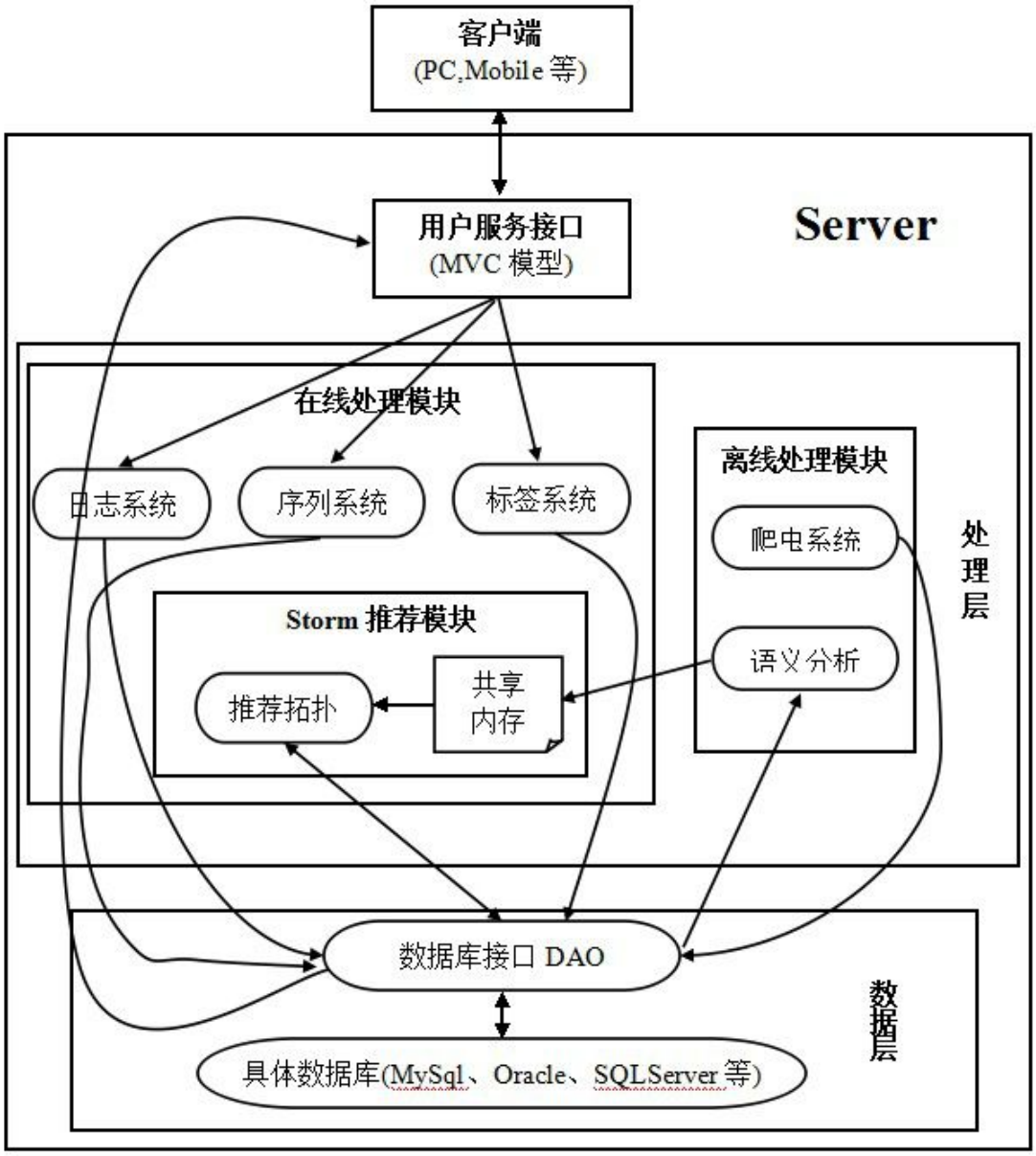
尽管优秀的推荐算法能够为用户推荐合理的结果，但只靠推荐很难构成一个完整的可用系统。要构建一个可用的推荐系统，比如一个电影推荐网站，就需要考虑推荐与系统其他组件的关系，只有这样才能最终实现推荐的价值。项亮在文献中给出了如上图所示的一般意义上推荐系统和网站其他系统之间的关系。首先，基本上所有的网站都配有一个用户界面，即 UI 系统，该系统主要用来向用户展示页面效果以及与用户进行交互。其次，网站往往还会配置日志系统，该系统主要用来将用户在用户界面上的各种有效行为记录下来并保存到对应的日志存储系统中。需要注意的是，这里的日志存储系统既可以是数据库，也可以说是缓存，还可以是文件系统。推荐系统作为一个网站的核心，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。由上图可知，推荐系统要想将强大的作用发挥出来，还要依赖于用户的历史行为信息和用户界面。

上图虽然给出了一般意义上的推荐系统的架构，给本文所实现的系统原型带来了很大启发，但这种架构显然过于简单，不能直接用于本文的系统原型。在认真分析本文所提推荐框架组成的基础上，本文按照下图对系统原型进行设计。

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。

客户端对应于上图中的 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。

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从将推荐的结果返回给客户端。



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执行一次。

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 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。
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。
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。
7. 用户服务接口将推荐列表展示给用户。

用户对歌曲打标签的执行过程如下所示：

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。
4. 标签系统对标签进行分析和处理。
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。

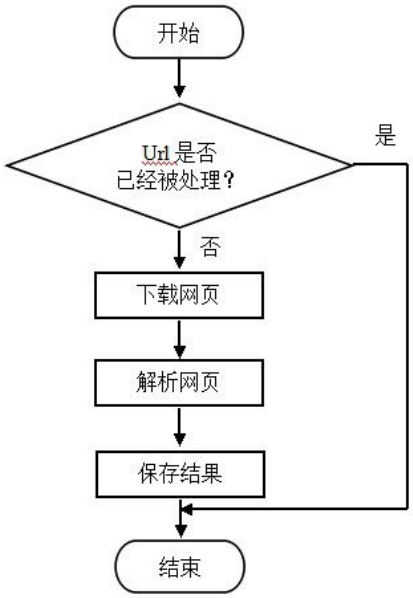
类似的，可以给出用户其他行为的处理过程。

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

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，最后将这些内容进行整理并保存到数据库中。对于一个待抓取的歌曲页面 URL，本文按照如下流程图进行工作：



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。其中，Scrapy 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的实现一个爬虫，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，人们只需要执行命令 scrapy startproject projectname 就可以得到如下图所示的项目目录，可见生成的目录包含若干文件，即 Scrapy 的模块。用户只需要在对应的文件中实现相应文件即可。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；spider 目录下存放写好的爬虫实际抓取逻辑。

```
projectname
|---   projectname
|      |---   __init__.py
|      |---   items.py
|      |---   pipelines.py
|      |---   settings.py
|      |---   spider
|              |---   __init__.py
|---   scrapy.cfg
```

右上可知，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 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，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 结巴分词工具

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。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 Computing Technology, Chinese Lexical 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“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：

(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)。

(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。

(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：

(1)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("他来到了网易杭研大厦")；

(3)支持多种分词模式。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。其中，精确模式试图将句子最精确地切开，适合文本分析；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；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。

对于歌曲对应的文档，我们使用结巴分词工具进行分词，得到的结果为：

5.3 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 和 Mallet 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。

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—“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.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。

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。
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，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。

5. 进行 LDA 主题模型建模，得到建模结果。

5.4 序列生成系统

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。

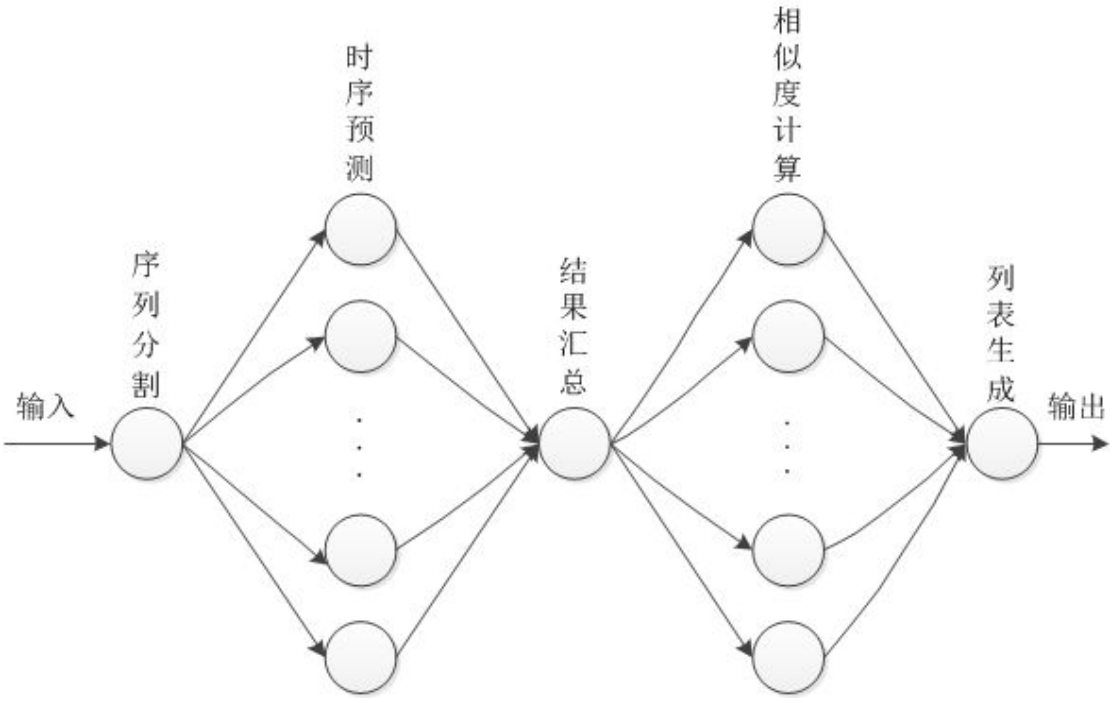
用户在收听一首歌曲的时候，可能对其产生两种典型的行为，即积极行为和消极行为。典型的积极行为包括用户收藏当前歌曲、分享当前歌曲、完整收听或者收听当前歌曲的绝大部分等，而典型的消极行为包括用户将歌曲丢进“垃圾桶”、选择跳过当前歌曲、只收听当前歌曲的一小部分等。对于收藏、分享、丢进“垃圾桶”、跳过这些比较直观地行为我们不再做进一步描述，而对于用户收听歌曲比例所表达的态度，本文认为如果用户收听当前歌曲的长度超过了歌曲总长度的50%那么用户喜欢该歌曲，否则认为用户不喜欢当前歌曲，即消极行为，如下式所示。其中，total_length 表示歌曲的总长度，past_length 表示用户收听该歌曲的时间长度，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”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 pastRatio(a，“大海”)=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，“大海”)=85.71%，我们认为用户 b 比较喜欢当前歌曲。至于用户为什么没有听完整整首歌曲，即 pastRatio(b,”大海”)不等于 1，可以认为用户收听的时间过长，想要换一首。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。

$$\text{pastRatio}(\text{user},\text{song}) = \frac{\text{past\_length}}{\text{total\_length}} \times 100\%$$

$$\text{attitude}(\text{user}, \text{song}) = \begin{cases} 1 & , \text{pastRatio} \geq 50\% \\ 0 & , \text{pastRatio} < 50\% \end{cases}$$

5.5 基于 Storm 框架的推荐引擎

在由 5.4 中的序列生成系统得到用户当前会话期的收听列表后，图所述的系统框架则激活基于 Storm 框架的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 以及对应的 DRPC 的基本内容，本节将介绍本文所实现的原型系统是如何利用的，对应的拓扑图如下所示。



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。

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 个子序列。其中，每一个子序列代表一个隐含主题，取值为歌曲在该主题上的隶属概率。这样，我们可以得到 K 个子序列。

3. 对 2 中的 K 个子序列进行时序预测，预测期下一个取值，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。

5. 计算曲库内所有歌曲与该歌曲的相似度。

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。

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。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。Nimbus 和 Supervisor 节点之间所有的协调工作是通过 Zookeeper 集群来实现的。

zkServer.sh start

2. 启动 nimbus。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。

storm nimbus

3. 启动 supervisor。每个工作节点（Work Node）上运行一个被称为 Supervisor 的后台程序。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。每一个工作进程执行一个 Topology 的子集；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。

storm supervisor

4. 启动 UI。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。

storm ui

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

storm drpc

6. 提交 topology。将实现好的 Topology 提交至 Storm 集群进行执行。

storm jar SweetFM.jar com.wst.sweetfm.topology.DRPC_MTSATopology sweetfm

7. 显示所有 topology

storm list

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);

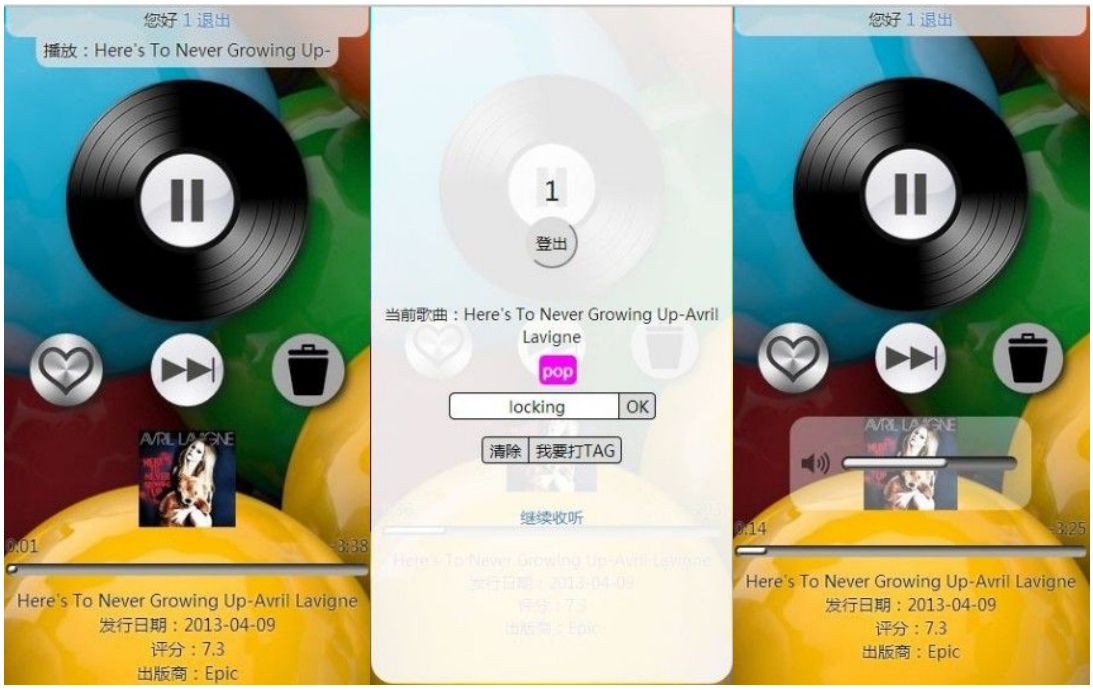
$seq = “4>6>1>8>2>0”;

$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]”

5.6 系统效果

本文按照如上所示的框架所实现的原型系统效果如下：



第六章 总结和展望

6.1 工作总结

本文首先给出了一种基于时间序列分析的个性化音乐推荐方法，该方法综合考虑了歌曲划分标准的不确定性以及用户所处上下文环境对推荐的影响和作用。首先，该方法利用用户给歌曲所打的标签构造歌曲对应的文档，在此基础上使用主题模型建模的方法对文档集合建模，将歌曲表示成若干隐含主题的概率分布。其次，该方法将用户在当前会话期内所收听的歌曲序列表示成与隐含主题个数对应的多个时间序列，然后使用相关时间序列分析方法对每一个时间序列进行分析和预测以得到用户可能收听的下一首歌曲在各个主题上的分布，最后通过计算曲库中歌曲与目标歌曲的相似度生成最终的推荐列表。在给出了基于时间序列分析的个性化音乐推荐方法后，本文进一步分析了用户听歌行为的特点，给出了用户听歌行为具有局部性、全局性和时序性的特点，在此基础上给出一种混合音乐推荐框架，综合考虑每一种特性的作用和影响。最后，本文给出了一个实现所述方法和框架的原型系统，验证了本文所述方法和框架的可行性和有效性。

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，下面试图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质的文档。

2. 本文所述方法目前能够得到较高的推荐准确度和较低的推荐误差，但时间效率不佳，需要进一步地完善和改进，使得所述方法和框架能够更好地满足实时性要求。

3. 本文所述的原型系统目前仅部署在单机上，下面需要迁移到集群上以提升系统的可用性。

致 谢

日光荏苒，三年的硕士学习生活已近尾声，在此对在我成长道路上谆谆教导的师长，在我生活中不断给予关心和帮助的同学表达深深的感谢。

感谢我的导师徐锋教授，从起始移动互联网领域的编程实践到后来学术科研的理论学习，老师都悉心地进行指导并且以身作则，从他身上，我不仅学到了先进的知识，更学到了理论与实践相结合的治学态度，老师经常建议我们要积极参与锻炼，在科研工作之余要保持健康的体魄和积极的精神状态。除此之外，他还给予我们实践自己想法的机会，使我的硕士生涯精彩而充实。

感谢计算机系软件所的吕建教授，陶先平教授，马晓星教授，胡浩教授，黄宇副教授，许畅副教授，曹春副教授，余萍，马俊和张建莹老师等，他们在我学习和工作中给予了许多帮助，能够从较高的角度给我们梳理知识，解答问题。感谢和我一起努力的同学，姚远，张栋栋，王瑶靖，袁明珠，周昊一，陈晓宇等，在论文和项目中的鼓励和协作。感谢室友王海涛，王逸恺等，让我对其他研究方向有所认知，他们科研的态度也给了我很大启发，而平时和谐愉悦的宿舍生活使得我在平时的科研和学习中充满活力，这定将成为我学生生活中最美好的记忆之一。

感谢父母一直以来对我的支持和鼓励，他们为了给我创造良好的成长环境和学习环境付出了很大的心血，他们是我坚实的后盾和持续奋斗的动力，我定不辜负他们对我的期望，在以后的工作中继续努力，成就自己的一番事业。

最后，感谢我的女朋友侯小丽这些年来的坚守和付出，这些年无论我遇到何种挫折和坎坷，她都出现在我的身边并给予我最大的支持和鼓励，我一定会持续积极上进，为以后的更加美好的生活创造条件。

简 历

基本情况

王守涛，男，汉族，1987 年 12 月出生，安徽阜阳人

教育背景

2011 年 9 月至 2014 年 6 月，南京大学计算机科学与技术系，硕士

2007 年 9 月至 2011 年 6 月，合肥工业大学计算机信息学院，本科

申请专利（已受理）

吕建,徐锋,王守涛. 一种基于多维时间序列分析的个性化音乐推荐系统及其实现方法:中国. 201410077177.1[P]. 2014-03-04.

附录 1. 数据集使用说明

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。

2 数据特点

1. 包含完整的用户、歌曲、曲作者的基本信息。

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、曲作者的显著标签信息，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。

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。数据集包含有 5 个基本的数据表：

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。

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。

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)、描述歌曲的显著标签

(toptag:text)等字段构成。

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)和描述曲作者的显著标签(toptag:text)构成。

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。

4 字段解析

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**。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。下表给出了 Small、Whole 和 Session 三类数据集的基本统计信息。

表 1. 不同数据集的统计信息

	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

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。

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著标签，数据如“{tag1:count1,tag2:count2,...,tagn:countn}”所示。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。

4.4 其他字段

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。

4 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。
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。

附录 2. 源代码使用说明

1 目录结构

本文实验主要包括离线的 LDA 建模和在线实验两部分，涉及的目录和文件如下所示：

GitRepo/eclipse_workspace/mallet/mallet-2.0.7/ - 使用 Mallet 进行离线 LDA 建模。

data/ - 需要建模的文档集合。

version.txt - 需要建模的文档集合参数:名称、LOC、文档数、主题数。

version-major.txt - 需要建模的文档集合名称。

data/LDA/ - 建模的结果。

script/ - 使用 mallet 进行 LDA 建模的 shell 脚本。

import-data.sh - 导入文档集合。

calculate-perplexity.sh - 计算 perplexity 以确定最优的主题数目。

runLDA.sh - 执行 LDA 建模，结果保存在../data/LDA/中。

GitRepo/Python/experiment/ - 在线实验以及实验结果。

img/ - 所有实验结果的示意图。

log/ - 所有实验的中间 log。

sys/ - 系统实现时歌曲数据集的分析(实验室可忽略)

txt/ - 所有实验的中间结果或需要保留的最终结果，以避免重复计算。

src/ - 所有实验的源码及文档(Python2.7)。

documentation/ - HTML 形式的源文件文档。

const.py - 基本常量的定义。

DBInfo.py - 数据库统计信息的获取。

arimaTrendTest.py - 验证“随着序列的增长，MTSA 的推荐效果并不会 一直提升而是会达到一个平衡”并以此作为选取最大序列长度的基础。

DocGenerate.py - 从数据库中读取用户列表和描述歌曲的显著标签，进而生成歌曲对应的文档。

lastfm.py - 从 Lastfm 上抓取实验相关的数据并存储到数据库中，

本文数据集即由该模块抓取。

main.py - 推荐的实验执行入口，当然也可以在各模块中独立执行。

model.py - 定义歌曲 Song 和列表 Playlist 两个类。

persist.py - 读取和保存中间结果到../txt 中。

predict.py - 实验的核心模块，描述了 MTSA、Local、Global、UserKNN、Markov、PatternMining 等推荐引擎以及混合推荐框架的工作流程。

PrefixSpan.py - 使用 PrefixSpam 挖掘序列数据库中的频繁项集并预测当前序列的下一个项。

test.py - 配置各种推荐引擎的输入参数并得到实验结果。

test_session.py - 验证“MTSA 在会话刚开始时或者跨会话时效果不佳”，进而为混合框架的提出提供基础。

textAnylize.py - 验证“LDA 与 VSM、TF-IDF 等模型相比能够得到更优的推荐效果”。

util.py - 实验中一些常用工具的定义，如相似度和距离的计算、推荐引擎名称和评测指标名称的获取、用户-歌曲矩阵的构建以及转移矩阵的构建等。