个人资料

JUAN425

访问： 138970次
积分： 3953
等级： BLOG 5
排名： 第6666名

原创： 92篇 转载： 26篇
译文： 162篇 评论： 22条

文章搜索

文章分类

人脸检测 (2)
C++ (139)
Linux (11)
OpenCV (13)
Python (7)
Computer Vision (9)
Machine Learning (14)
DeepLearning (21)
Data Structure (58)
Information Security (2)
NLP (0)
Matlab (0)
STL学习 (16)
linux shell (4)
综合 (85)
刷题 (36)

文章存档

2015年11月 (1)
2015年09月 (2)
2015年08月 (1)
2015年06月 (1)
2015年05月 (2)

展开

阅读排行

Torch7安装 (8002)
关于codeblocks 的程序中 (2933)
SIFT（scale invariant fea (2343)

音乐推荐系统

2015-04-16 21:34 808人阅读 评论(1) 收藏 举报

分类： Machine Learning（13） 综合（84）

尽管music retrieval techniques(MRT)已经很成功了， 但是音乐推荐系统的发展却仍然处于一个刚刚起步的阶段。 目前， 用于音乐推荐的**算法**主要有如下两种：

(1) 协同滤波（collaborative filtering, 简称CF）

(2) 基于内容的模型（content-based model, 简称CBM）

一个music recommender system主要有三部分组成：

(1) user modelling(用户建模)：即对用户进行建模。

(2) items profiling（即音乐库中的很多音乐）： 对每一首歌曲进行profiling（描述歌曲的特征）。

(3) user item matching algorithms(音乐匹配算法)

User profiling 解决的是不同的users的profile的的差异性。 这一步的目的就是使用基本的信息对用户的music taste进行划分。

Item profling描述的是3种不同type的metadata: editorial, cultural, 以及acoustic， 在不同的推荐算法中均被使用三种type的meta data。 第三种办法就是关于检索的问题了， 如何在music recommander system进行query相应的music， 如何进行相关的匹配的问题。

（1）User Modelling

也就是对用户建模的问题。

一个成功的音乐推荐系统需要对不同的用户提供满足其不同相应需求的音乐。 然而， 获取用户的信息需要花费很大的financial costs以及human label。

用户建模， 就是对用户的profile的差异性进行建模。 例如， 不同的地理位置的人， 或者不同的年龄段， 性别， 生活方式， 兴趣等等这些因素会影响用户的音乐偏好， 决定着他们喜欢哪一种类型的音乐。 根据Rentfrow和Gostling最近关于人的music preference 和BIG-FIve inventory(简称BFI)的关系。 BFI包括openness, conscientiousness, extraversion, aggreeableness, neuroticisim这五大特征。 例如， 研究表明外向的人（extraverted）的人更喜欢听那些energetic， rhythimic的音乐。

所以对用户进行建模， 对于预测用户的music taste是十分的essential的。user modelling包括两个部分：

(1) user profile modelling

(2) user experience modelling

要进行对用户的完整建模， 所以对应着如下两大步骤：

第一步： user profile modelling.

Celma建议， user profile可以分为三大domains的特征， 即demographic（包括年龄， 型别， 婚姻状况等人口统计学概念）, geographic（包括所住的城市， 地域等等, 比如中国人更喜欢华语歌手的音乐等等）, psychographic（这个包含心理学的范畴。比如一个人比较稳定的（stable）的特征是是兴趣， 生活方式， 性格等， 比较容易变化（fluid）的包括情绪， 观点， 态度等等）， 比如一个人在一天中情绪是随着时间变化的。

Data type	Example
Demographic	Age, marital status, gender etc.
Geographic	Location, city, country etc.
Psychographic	*Stable*: interests, lifestyle, personality etc. *Fluid*: mood, attitude, opinions etc.

Table 1. User profile classification

第二步： User listening experience modelling(对用户的音乐经历进行建模)

不同的用户的music的expertise是不一样的， 那么他们对所听音乐的expectations也是不一样的。 比如我们可以把人群按照对音乐理解程度， 专业知识划分为四类：

savent(专业音乐人士)， enthusiast(喜欢音乐的人)， causal（闲的时候听听音乐）， indifferents(听不听音乐无所谓)。

在人群中相关人群分布如下：

Type	Percentage	Features
Savants	7	Everything in life seems to be tied up with music. Their musical knowledge is very extensive.
Enthusiasts	21	Music is a key part of life but is also balanced by other interests.
Casuals	32	Music plays a welcome role, but other things are far more important.
Indifferents	40	They would not lose much sleep if music ceased to exist, they are a predominant type of listeners of the whole population.

Table 2. Use listening experience categorisation

上述的关于用户的这些信息对于音乐推荐系统设计来说， 都是需要考虑的因素。 例如， 基于用户对于所听音乐的期许， 我们需要考虑那些隐藏popularity curve的long tail的未知的音乐给filter出来， 推荐给这些用户。 所谓的polarity cuve， 就是音乐的受欢迎情况是一种short head, long tail的情况。 也就是那些最受欢迎的音乐在那个short head部分， 大部分的音乐播放量少些，仅在long tail。 另外，好听的音乐， 人们会反复的播放很多次。

这就是所谓的long tail的理论。 也就是说那些popular的items的数量很少， 然后就是知名的音乐， 接下来大部分的剩下的音乐位于the heavy tail。 我们音乐推荐的目的就是把这些long tail的音乐中， 对于用户来说未知的音乐， 发掘出来， 推荐给用户。

这些用户的相关的信息可以通过调查的办法获得， 也可以通过他们在long tail处的行为观察出来。

Item profiling

这是推荐系统的第二个关键的component。 因为这是我们要想用户推荐的东西。 要想成功的推荐合适的， 我们必须要研究音乐本身的属性。

music item的metadata主要分为3个categories: editorial metadata(EM)， cultural metadata(CM), acoustic metadata(AM)。

Editorial metadata: 这是由音乐的分布者给音乐的一些tag， 包括cover name. composer, title, genre等等。

Cultural metadata: 这种数据的获取通常是通过互联网或者一些公共的sources获得的。 源于对这首歌出现的patterns，与所嵌入的源文档的关系等等。 例如， 和其他的music items的similarity， 就是cultural metadata的一种。

acoustic metadata: 源于对音乐音频信号的分析。 例如beat, tempo, pitch, instrument, mood等等。

音乐的query type

（1） 假如用户已经知道那首音乐， 那么直接根据editorial information 例如music tile或者歌手， 或者歌词等直接搜索即可。

（2） 最近10年， 出现一种新的音乐检索办法， 就是QBSH(query by hamming/singing system)。 当你忘记一首歌曲的名字以及歌唱者的相关信息， 你只需要记录一下音乐的音频片段， 然后直接检索即可。

（3） 更合适的办法就是根绝用户的listening histories进行query, 以便detect 出他们的 music preferences。

音乐推荐系统的State of the art

（1） Metadata information retrieval

这是search music的最fundamental的办法。 我们只需要根据歌曲的名字， 艺术家的名字， 或者歌曲的lyrics直接找到目标歌曲。

尽管很快很准确， 但是这个方法要求用户必须知道某个音乐的edirotial information。 而且当metadata变大的时候， 也很费事。而且推荐系统的性能也很差， 因为她只能根据editorial metadata 向用户推荐音乐， 根本没考虑用户的相关信息。

（2）Collaborative filtering(协同滤波)

协同滤波是根据其他similar的users的选择向这个用户推荐items。 这一技术基于这样一种假设： 假如用户X和用户Y对n个items的评估是相似的， 或者这两个用户对这n个items有相似的行为， 那么X和Y在其他的items上也有相似的行为或者评估。

这里， 我们并不计算items之间的相似性， 我们只是找出和这个用户X的past ratings具有很强的关联性的的neighbour users, 然后对于一个X为见过的item, 我们是基于X的neighbours 对这个item评分组合作为该item对X的吸引程度的评估预测。 协同滤波进一步的可以分为三种：

(1) memory based collaborative filtering

每一个user同和这个user具有similiar interests的people group 在一起。 这样， 我们可以通过找出user的nearest neighbour，然后进行massive number of explicit user votes去产生一个新的items。

(2) model based collaborative filtering

这个办法是通过使用**机器学习**和数据挖掘的算法去对用户的preferences和rating score进行训练， 得到模型。 通过将用户的preferences 用一组rating scores 去构建一个预测模型。 最后我们基于训练得到的预测模型去对**测试**集进行预测。 性能足够好了， 就可以对real world data 进行预测了。

(3) hybrid collaborative filtering。

混合协同滤波就是通过combining 不同的CF models, 可以达到提升性能的目的。

协同滤波很好， 但是有一定的的limitations。

（1） 问题一： popularity bias: popular music将会有更多的评分， 而那些在long tail的music却有很少的评分。 这就导致协同滤波主要推荐的是popular items。 尽管推荐popular items非常的reliable, 似乎是万金油， 但是却risky， 因为用户很少会对推荐的热门的歌曲surprised（惊喜）。

（2）问题二： cold start:， 又被称为data sparsity problems。 在歌曲刚开始发布的时候， 具有的评分很少， 由于缺少ratings, 所有导致预测的结果很poor。

（3） 问题三： human effort， 一个好的音乐推荐系统不应该有太多Human efforts的参与。 因为用户都是懒的， 我们不希望用户去对歌曲评分。 况且评分也不具有代表性， 而且误差也很大， 因为很多人可能不去评分， 而有些人觉得音乐不好采取评分， 给的分较低， 也就产生了偏差了， 可能音乐没有那么差。

Content-based music information retrivieval

基于内容推荐办法主要是根据一首song和一个user已经听到过的歌曲进行比较相似度， 推荐与用户听过历史歌曲具有相似度的歌曲。 这就涉及到对每一首音乐进行特征提取的办法了。 主要是提取歌曲的acoustic features。 然后去度量这些歌曲的距离作为相似度度量。 主要有如下几类：

（1） k means clustering with earth-mover's distance. 主要计算两首歌曲的GMM（gaussian mixture models）的gerneral distance。

(2) Expectation-Maximization with Monte Carlo Sampling。 通过从两首歌的GMMs中采样出对应的向量， 然后计算两个向量的距离。

（3） Average feature vectors with euclidean distance. 通过计算歌曲的不同的segments的low-order 的statistics例如mean, viriance得到。

从未看过推荐算法的东西。 这里只是稍微总结一下。具体见论文：

A Survey of Music Recommendation Systems and Future Perspectives

上一篇　海量数据找缺失的值

下一篇　反转链表(reverse a lnked list)

我的同类文章

Machine Learning（13）　综合（84）

- Torch7安装 2015-05-15 阅读 7987
- Adaboost， boosting 和bagging… 2015-04-05 阅读 920
- C++易忽略点 2015-04-01 阅读 270
- Bias 和 Variance的理解 2015-04-01 阅读 388
- SVM(Support Vector Machines)… 2015-03-30 阅读 610
- 衡量分类器的性能指标 2015-04-06 阅读 446
- Logistic regression---原理部分 2015-04-03 阅读 305
- 机器学习注意 2015-04-01 阅读 288
- Linear models for regresion(回归) 2015-04-01 阅读 371
- 决策树算法 2015-03-27 阅读 640

更多文章

参考知识库

机器学习知识库

16782 关注 | 2140 收录

软件测试知识库

4054 关注 | 310 收录

算法与数据结构知识库

14529 关注 | 2320 收录

猜你在找

用redis 搭建大数据 热门排行榜，用户推荐系统 探索推荐引擎内部的秘密第 1 部分 推荐引擎初探

UML建模技术 探索推荐引擎内部的秘密第 1 部分 推荐引擎初探

iOS8开发技术（Swift版）：音频和视频 探索推荐引擎内部的秘密第 1 部分 推荐引擎初探

模板匹配的字符识别(OCR)算法原理 探索推荐引擎内部的秘密第 1 部分 推荐引擎初探

查看评论

1楼 jiukouluanchua1 2016-01-06 21:45发表

谢谢，用户建模部分受益:)

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN
Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5
Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML
components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone
CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl
Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django
Bootstrap