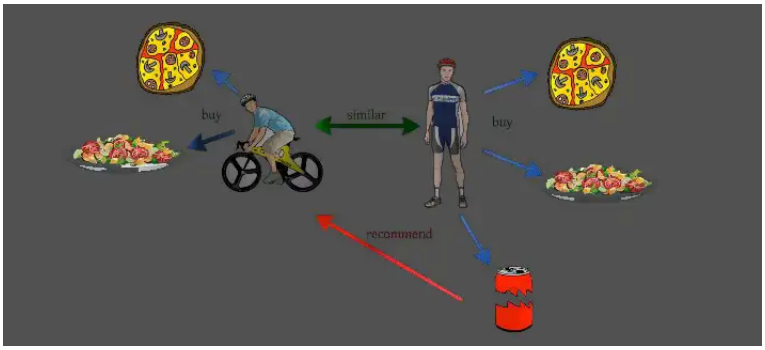
推荐系统从入门到接着入门



前言

想来惭愧，推荐系统从大四做毕设时就开始接触了，不过当时对于推荐系统也是云里雾里，没有一个整体的概念，更别说总结写博客了。正好研究生也是这个方向，最近一年看了一些综述论文、经典书籍以及好的博客，希望通过这篇博客能够记录一些推荐系统方面的基础、经典的理论、总结以及自己的想法。 等回头再过来浏览的时候希望能够起到一个索引或者综述的作用，如此而已。如果有人看到这篇博客并且对Ta有所帮助的话，更是欣慰。也希望大家多提意见，大恩不言谢。

最近整理了一份推荐系统干货总结，希望对大家也会有所帮助。

简介

说到推荐系统，我们肯定是要问它是为什么而存在的，即存在的意义是什么。

随着当今技术的飞速发展，数据量也与日俱增，人们越来越感觉在海量数据面前束手无策。正是为了解决**信息过载**(Information overload)的问题，人们提出了**推荐系统**（与搜索引擎对应，人们习惯叫推荐系统为推荐引擎）。当我们提到推荐引擎的时候，经常联想到的技术也便是**搜索引擎**。不必惊讶，因为这两者都是为了解决信息过载而提出的两种不同的技术，一个问题，两

个出发点，我更喜欢称它们两者为兄弟，亲近而形象。

兄弟二人有共同的目标，即解决信息过载问题，但具体的做法因人而异。

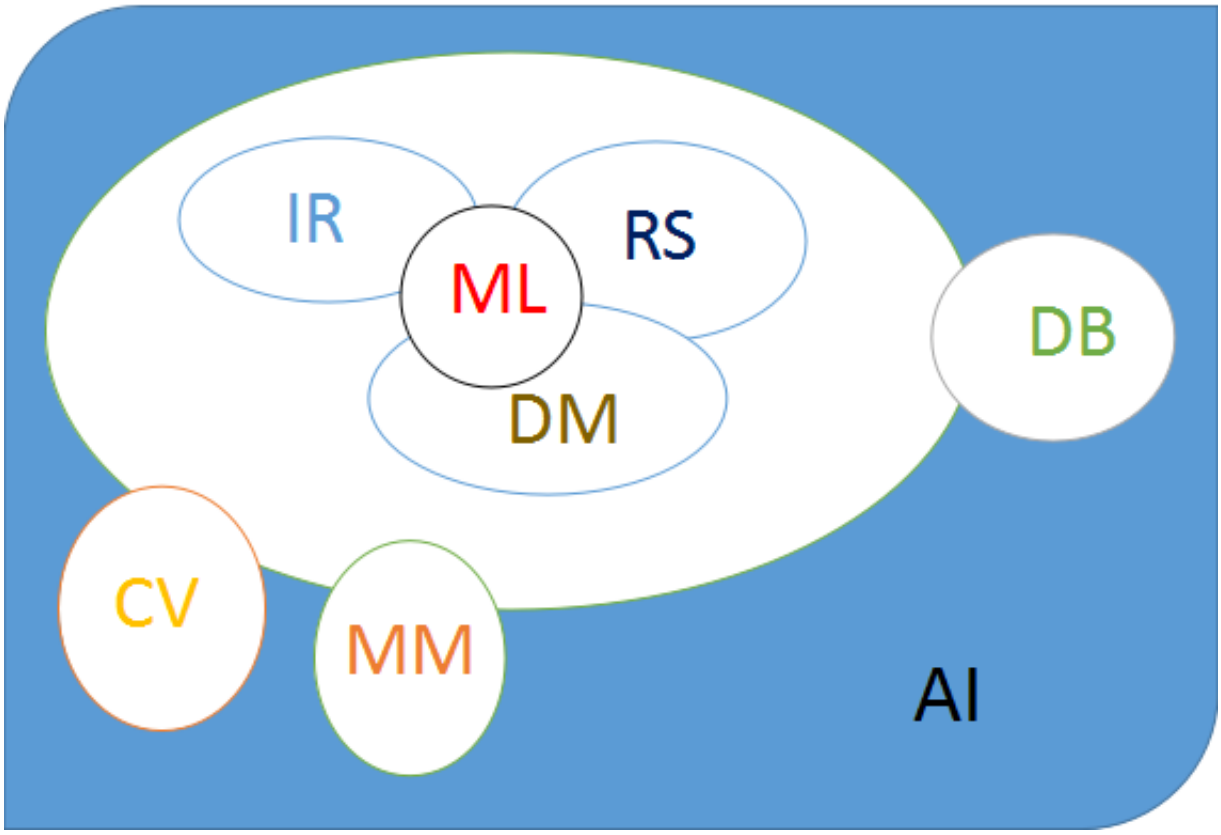
搜索引擎更倾向于人们有明确的目的，可以将人们对于信息的寻求转换为精确的关键字，然后交给搜索引擎最后返回给用户一系列列表，用户可以对这些返回结果进行反馈，并且是对于用户有主动意识的，但它会有**马太效应**的问题，即会造成越流行的东西随着搜索过程的迭代会越流行，使得那些越不流行的东西石沉大海。

而**推荐引擎**更倾向于人们没有明确的目的，或者说他们的目的是模糊的，通俗来讲，用户连自己都不知道他想要什么，这时候正是推荐引擎的用户之地，推荐系统通过用户的历史行为或者用户的兴趣偏好或者用户的人口统计学特征来送给推荐算法，然后推荐系统运用推荐算法来产生用户可能感兴趣的项目列表，同时用户对于搜索引擎是被动的。其中**长尾理论**（人们只关注曝光率高的项目，而忽略曝光率低的项目）可以很好的解释推荐系统的存在，试验表明位于长尾位置的曝光率低的项目产生的利润不低于只销售曝光率高的项目的利润。推荐系统正好可以给所有项目提供曝光的机会，以此来挖掘长尾项目的潜在利润。

如果说搜索引擎体现着马太效应的话，那么长尾理论则阐述了推荐系统所发挥的价值。

所属领域

推荐系统是多个领域的交叉研究方向，所以会涉及机器学习以及数据挖掘方面的技巧（推荐系统==》数据挖掘/机器学习==》人工智能）。在这整理了小邬老师上课所介绍的关于主流研究方向的结构图。



会议介绍

在这里主要整理一下上图所涉及到的研究方向相关的会议。

RS(Recommender System)： RecSys

IR (Information Retrieval): SIGIR

DM(Data Mining): SIGKDD,ICDM, SDM

ML (Machine Learning): ICML, NIPS

CV (Computer Vision): ICCV, CVPR, ECCV

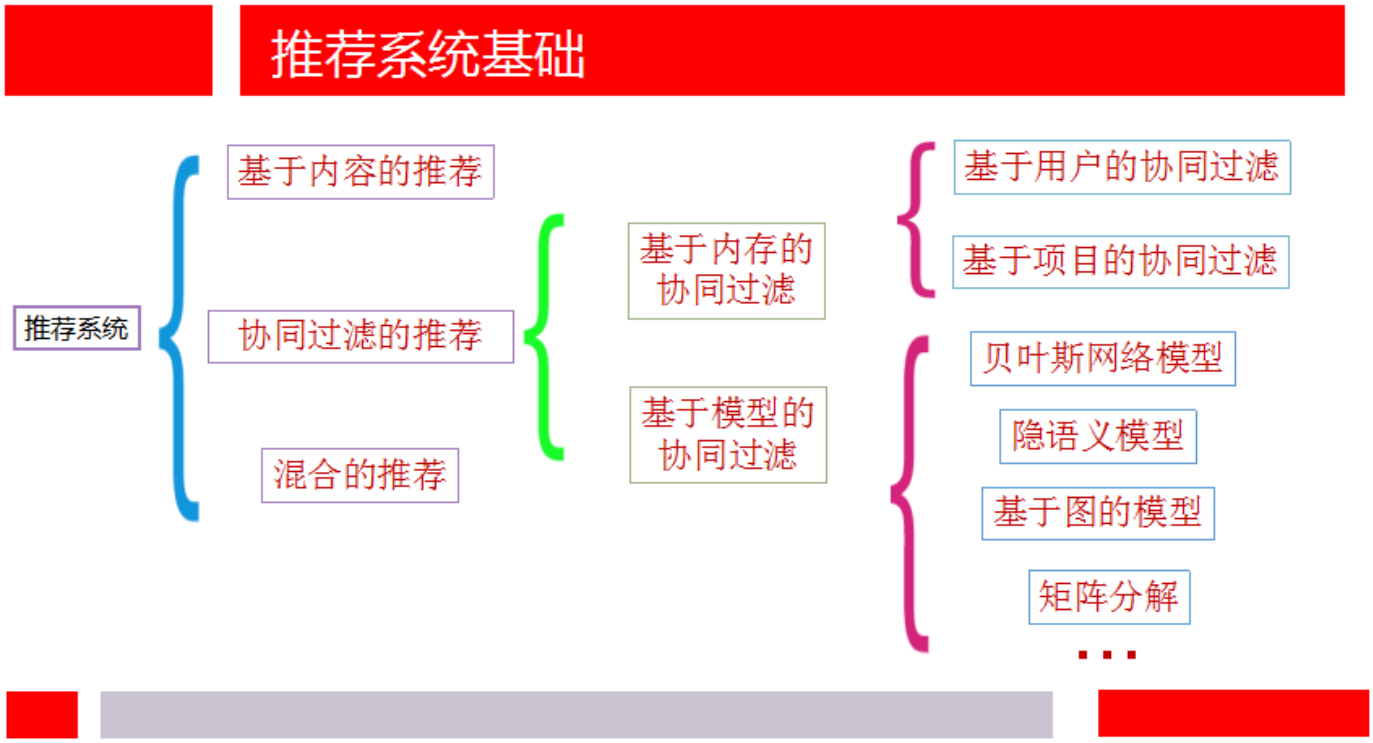
MM (MultiMedia): ACM MM

DB (Database): CIKM, WIDM

AI (Artificial Intelligence): IJCAI, AAAI

推荐系统分类

说到推荐系统的分类，我还是想从简单的方式开始，对于一些新颖的推荐系统方法，之后再介绍。根据推荐算法所用数据的不同分为基于内容的推荐、协同过滤的推荐以及混合的推荐。在这放一张第一次组会时的ppt：



1. **基于内容的推荐**

顾名思义，它是利用项目的内在品质或者固有属性来进行推荐，比如音乐的流派、类型，电影的风格、类别等，不需要构建UI矩阵。它是建立在项目的内容信息上作出推荐的，而不需要依据用户对项目的评价意见，更多地需要用机器学习的方法从关于内容的特征描述的事例中得到用户的兴趣资料。

以前一直觉得基于内容的推荐算法最简单，没有啥技术含量，直接基于项目的相似度来通过最近邻获取与目标项目最相似的项目列表，然后把用户没有行为记录并且评分高的项目推荐给特定用户。但后来看Andrew NG的机器学

习课程中有一节对于推荐系统的介绍，他是通过机器学习的思想来通过训练来拟合用户的特征属性。首先我们需要一个效用函数来评价特定用户c对于特定项目s的评分：

$$u(c,s) = score\big(UserPro(c), ItemPro(s)\big)$$

至于如何根据项目的内容属性来学习到跟项目一样维度的用户属性，这就涉及到另一公式：

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

他是通过梯度下降法来最小化误差的平方损失，其中θj为所要学习的用户维度特征，Xi为项目的内容维度特征，我们所要训练的是用户j对于已有行为的项目j的训练，来使得观测数据与预测数据的误差最小。

2. 基于协同过滤的推荐

顾名思义，它是通过集体智慧的力量来进行工作，过滤掉那些用户不感兴趣的项目。协同过滤是基于这样的假设：为特定用户找到他真正感兴趣的内容的好方法是首先找到与此用户有相似兴趣的其他用户，然后将他们感兴趣的内容推荐给此用户。

它一般采用最近邻技术，利用用户的历史喜好信息计算用户之间的距离，然后利用目标用户的最近邻居用户对商品评价的加权评价值来预测目标用户对特定商品的喜好程度，系统从而根据这一喜好程度来对目标用户进行推荐，通常需要用到UI矩阵的信息。协同过滤推荐又可以根据是否运用机器学习的思想进一步划分为基于内存的协同过滤推荐（Memory-based CF）和基于模型的协同过滤推荐(Model-based CF)。

基于内存的协同过滤推荐

其中基于内存的推荐系统（Memory-based CF）主要是通过启发式的方法来进行推荐，主要步骤一个是相似性函数的选择，如何选择合适的相似性函数来更好的度量两个项目或者用户的相似性是关键；另一个主要步骤是如何进行推荐，最简单的推荐方法是基于大多数的推荐策略，即推荐那些大多数人产生过行为而目标用户未产生过行为的项目。

根据用户维度和项目维度的不同而分为**Item-based CF**和**User-based CF**。

- Item-based CF：

①首先需要构建UI矩阵；

②根据UI矩阵来计算列（项目维度）的相似度；

③选择特定项目最相似的k个项目构成推荐列表；

④推荐给特定用户列表中还没有发生过行为的项目。

- User-based CF:

①首先需要构建UI矩阵；

②根据UI矩阵来计算行（用户维度）的相似度；

③选择特定用户最相似的k个用户；

④推荐给特定用户列表中还没有发生过行为而在相似用户列表中产生过行为的高频项目。

基于模型的协同过滤推荐

基于模型的推荐系统（Model-based CF）主要是运用机器学习的思想来进行推荐，说到机器学习思想那真是不胜枚举。记得小邬老师提过，目前机器学习主要是研究以下几种方式：

① 损失函数+正则项（Loss Function）；

通过对不同的任务来建立不同的损失函数加正则项来解决问题。比如著名的Lasso Regression、Ridge Regression以及Hinge Regression等。

② 神经网络+层（Neural Network）；

通过对不同的任务来设计不同的网络结构来解决问题。比如RNN、CNN以及GAN等。

③ 图模型+圈(Graph Model**);**

通过运用图的知识来解决不同的实际问题。比如马尔科夫模型等。

回到机器学习方法在推荐系统的应用上来，主要的方法为分类算法，回归算法、聚类算法、矩阵分解算法、神经网络算法、图模型算法以及隐语义模型等，在这主要介绍基于**矩阵分解**的推荐系统算法，以后有时间再慢慢补充吧。

基于矩阵分解的推荐

首先我们需要明确所要解决的问题，即对于一个M行（M个item），N列（N个user）的矩阵，当然这个矩阵是很稀疏的，即用户对于项目的评分是不充分的，大部分是没有记录的，我们的任务是要通过分析已有的数据（观测数据）来对未知数据进行预测，即这是一个矩阵补全（填充）任务。矩阵填充任务可以通过矩阵分解技术来实现。

1. Traditional SVD:

当然人们首先想到的矩阵分解技术是SVD（奇异值）分解，在这我命名为traditional SVD（传统并经典着），直接上公式：

$$M_{m\times n}=U_{m\times k}\Sigma_{k\times k}V^T_{k\times n}$$

当然SVD分解的形式为3个矩阵相乘，中间矩阵为奇异值矩阵。如果想运用SVD分解的话，有一个前提是要求矩阵是稠密的，即矩阵里的元素要非空，否则就不能运用SVD分解。很显然我们的任务还不能用SVD，所以一般的做法是先用均值或者其他统计学方法来填充矩阵，然后再运用SVD分解降维。

2. FunkSVD

刚才提到的Traditional SVD首先需要填充矩阵，然后再进行分解降维，同时存在计算复杂度高的问题，所以后来提出了FunkSVD的方法，我总是念成Fuck，顾名思义，作者发明出这个算法的时候一定是太高兴，不由自主的说出了Fuck，这个算法真是太惊艳了！哈哈，纯属笔者开玩笑，实际上是以人家的名字命名的。它不在将矩阵分解为3个矩阵，而是分解为2个**低秩**的用户项目矩阵，在这里低秩的解释可以是：在大千世界中，总会存在相似的人或物，即物以类聚，人以群分。在这里，笔者总是混淆稀疏矩阵与低秩矩阵的概念，所以特此说明一下：

稀疏矩阵（sparse matrix）：指的是矩阵中的非零元素比较少，但不一定是低秩的。比如对角矩阵，稀疏但是却满秩。

低秩矩阵（low-rank matrix）：指的是矩阵的秩比较小，但不一定是稀疏的。比如全为1的矩阵，秩虽然小仅为1，但确实稠密矩阵。

好像扯得有点远，不多说了，上公式：

$$\sum_{i,j}(m_{ij}-q_j^T p_i)^2$$

借鉴线性回归的思想，通过最小化观察数据的平方来寻求最优的用户和项目的隐含向量表示。同时为了避免过度拟合（Overfitting）观测数据，又提出了带有L2正则项的FunkSVD，上公式：

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

以上两种最优化函数都可以通过梯度下降或者随机梯度下降法来寻求最优解。

3. BiasSVD:

在FunkSVD提出来之后，出现了很多变形版本，其中一个相对成功的方法是BiasSVD，顾名思义，即带有偏置项的SVD分解，还是直接怼公式：

$$\underbrace{arg\ min}_{p_i,q_j} \sum_{i,j}(m_{ij} - \mu - b_i - b_j - q_j^T p_i)^2 + \lambda(||p_i||_2^2 + ||q_j||_2^2 + ||b_i||_2^2 + ||b_j||_2^2)$$

它是基于这样的假设：某些用户会自带一些特质，比如天生愿意给别人好评，心慈手软，比较好说话，有的人就比较苛刻，总是评分不超过3分（5分满分），笔者就是这样的人儿；同时也有一些这样的项目，一被生产便决定了它的地位，有的比较受人们欢迎，有的则被人嫌弃，这也正是提出用户和项目偏置项的原因；项亮给出的解释是：对于一个评分系统有些固有属性和用户物品无关，而用户也有些属性和物品无关，物品也有些属性与用户无关。

4. SVD++:

人们后来又提出了改进的BiasSVD，还是顾名思义，两个加号，我想是一个加了用户项目偏置项，另一个是在它的基础上添加了用户的隐式反馈信息，还是先上公式：

$$\underbrace{arg\ min}_{p_i,q_j} \sum_{i,j}(m_{ij} - \mu - b_i - b_j - q_j^T p_i - q_j^T |N(i)|^{-1/2} \sum_{s\in N(i)} y_s)^2$$
$$+ \lambda(||p_i||_2^2 + ||q_j||_2^2 + ||b_i||_2^2 + ||b_j||_2^2 + \sum_{s\in N(i)} ||y_s||_2^2)$$

它是基于这样的假设：用户对于项目的历史评分记录或者浏览记录可以从侧面反映用户的偏好，比如用户对某个项目进行了评分，可以从侧面反映他对于这个项目感兴趣，同时这样的行为事实也蕴含一定的信息。其中N(i)为用户i所产生行为的物品集合；ys为隐藏的对于项目j的个人喜好偏置，是一个我们所要学习的参数；至于|N(i)|的负二分之一次方是一个经验公式。

5. BiasSVDwithU：

我们实验室又提出了一种带有用户平滑项的SVD分解方法，还是先上公式吧：

$$\underset{pi,pj}{\operatorname{argmin}} \sum_{(i,j)\in k}(m_{ij}-\mu-b_i-b_j-q_j^T p_i)+\lambda(\| p_i \|_2^2+\| q_j \|_2^2+\| b_i \|_2^2+\| b_j \|_2^2)+\beta\sum_{k=1}^{n} S_{ik}(p_i-p_k)$$

它是基于这样的假设：相似的用户所学到的用户隐含特征向量应该更相似，即在现实空间中两个相似的用户投影到测度空间上仍然保持相近的距离。

3. 基于混合的推荐

基于混合的推荐，顾名思义，是对以上算法的融合，像淘宝既有基于内容的推荐也有协同过滤的推荐。具体怎么融合还是要结合具体的应用场景，包括是对特征的融合还是对算法层面的融合。其中说到算法的融合，想到了机器学习模型常用的三种模型融合方法：**Bagging、Boosting和Stacking**。

Bagging（装袋）方法：该方法通过重采样技术生成若干个不同的子训练集，然后在每个训练集上训练一个分类器，然后采用投票的方式取大多数的结果为模型的最终结果。模型更像是发挥民主作用的人民代表大会制度，还是大部分人说了算的。

Boosting（强化提升）方法：每个训练样例都有权重，每次训练新分类器的时候都着重训练那些再上一次分类过程中分错的样例，权重会随着迭代次数的变化而变化。模型更像是有了记忆能力，加大力度惩罚那些在上一轮不乖的样例而使得他们越来越听话。

Stacking（堆叠）方法：每个分类器首先做一遍决策，然后将分类器们的决策送到更高一层的模型中，把他们当做特征再进行一次训练。每个单独分类器的输出会作为更高层分类器的输入，更高层分类器可以判断如何更好的合并这些来自低层的输出。模型更像是神经网络中的轴突，低层的输出作为高层的输入。

【具体思路】 给定一个train数据集和一个test数据集，我们的任务是分类。①首先需要确定基模型，在这选择KNN，DecisionTree和SVM三个；②其次是要把train数据集分成5折的交叉验证，4份用来训练，1份用来交叉验证；③选择一个基模型KNN，然后在train数据集上做交叉验证，每次用4N/5来训练，N/5来测试，共测试5次，这样就会得到整个train数据集上的预测；同样用每次训练好的模型来预测test，那么可以得到5个对于test的预测，然后取平均作为结果；⑤重复步骤3、4，这样会得到对于train的3列新的特征表达（每一列是一个基模型的预测结果），同理也会得到测试集的3列新的特征表达；⑥将新的3列train特征作为第二层模型（在这我们用LR）的输入，再次进行训练；⑦用test上3列新的特征作为输入，送入训练好的模型来预测结果。

有几个基模型，就会对整个train数据集生成几列新的特征表达。同样，也会对test有几列新的特征表达。

4. 基于人口统计学的推荐

主要是根据用户的注册信息来进行简单推荐，不展开介绍。

5. 基于规则的推荐

主要根据简单的规则或者领域知识来进行推荐，比如热门推荐等，不展开介绍。

评测指标

评测指标是用来评价一个系统性能好坏的函数，可以分为对于算法复杂度的度量以及算法准确性的度量。算法复杂度主要考虑算法实现的空间以及时间复杂度，当然算法复杂度同样重要，但这里主要讨论算法的准确性度量指标。

推荐系统根据推荐任务的不同通常分为两类：**评分预测**与**Top-N列表推荐**。在这里主要根据这两者来分别讨论评测指标。

- **评分预测任务：**

预测特定用户对于没有产生过行为的物品能够打多少分。评分预测一般通过均方根误差（RMSE）和平均绝对误差（MAE）来计算。对于测试集中的用户u和项目i，rui是用户u对项目i的真实评分，r^ui是推荐算法预测出的评分，那么RMSE：

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

MAE为：

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

其中Netflix认为RMSE加大了对预测不准的用户物品评分的惩罚(平方项的惩罚),因而对系统的评测更加苛刻，同时如果评分系统是基于整数建立的(即用户给的评分都是整数),那么对预测结果取整会降低MAE的误差。

- **Top-N列表推荐：**

评分预测只能适用于小部分的场景，比如对于电影，书籍的评分，其实Top-N推荐更加符合现在的需求，给用户提供一个推荐的列表让其进行选择。

Top-N推荐一般通过准确率与召回率来进行衡量。其中令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（指的是预测的推荐列表）,而T(u)是用户在测试集上的行为列表（指的是真实的列表GroundTruth）,在这笔者总是容易混淆两者的含义。

准确率的意义在于所预测的推荐列表中有多少是用户真是感兴趣的，即预测列表的准确率，那么准确率的定义为：

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

召回率的意义在于真正用户感兴趣的列表中有多少是被推荐算法准确预测出来的，即真实列表的召回率，那么召回率的定义为：

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

两个评测指标从不同的方面来衡量推荐系统的好坏，两者呈现一个负相关的状态，即准确率高的情况下召回率往往会比较低，反之亦然。所以人们又提出了一个结合了准确率与召回率的评测指标F1值，可以更加方便的观察推荐系统的好坏，公式如下：

$$F_1 = \frac{2PR}{P+R}$$

当谈到准确率、召回率以及F值的时候，它们都是基于**混淆矩阵**（confusion matrix）来说的，见下图：

GroundTruth（真实结果）	Prediction Result（预测结果）	
	正例	反例
正例	TP(预测正确的正例)	FN(预测错误的反例)
反例	FP(预测错误的正例)	TN(预测正确的反例例)

笔者在第一次看这张图的时候会有一些疑惑，所以在这做一下解释。一开始笔者以为横坐标是正例(P)与反例(N),纵坐标是真(T)与假(F),后来发现不对，这张图是对于二分类任务来说的，真实结果中分为了正例与反例，同理预测结果肯定也会是这两类正例与反例。所以这也是为什么横纵坐标都是正例与反例了。至于里边写的T和F是针对于预测结果而言的，即预测正确了是T，预测错误为F，所以TP的含义为预测正确的正例。所以

准确率可以表示为TP/(TP+FP)；

召回率表示为TP/(TP+FN)；

精度（Accuracy）也可以用混淆矩阵表示(TP+TN)/(TP+FN+FP+TN)。

【补充】笔者看到准确率（Precision）、召回率（Recall）的时候，总是会联想到精度（Accuracy）这个指标，不知道大家有没有这样的想法。

对于一般的问题，用精度（Accuracy）这个指标是可以的，预测正确的样例个数比上总的样例个数。但对于有偏斜（**skewed class data**，又称unbalanced data）的数据的时候，就不那么奏效了。比如对于二分类问题，训练集数据99%为负例，仅1%为正例。那么我用一个简单的规则来进行预测：即无论数据的特征是什么，我都预测为负例，那么我这个带有规则的算法的精度可以高达99%，实际上对于再厉害的机器学习算法也很难达到这样的标准，很显然这样的指标在不平衡的数据上是不客观的。那么召回率就可以比较好的进行评价了，预测为正例的个数比上实际的测试集上正例的个

数，很显然对于刚才那么一直预测为负例的规则算法，它的召回率是0。

致谢

第一次比较系统的整理与总结自己所学过的东西，首先要感谢一下辛苦的自己，写博客真的是一件很费精力与消耗时间的任务，很庆幸坚持了下来，奖励自己一根冰棍；接着要感谢小邬老师的悉心指导，许多论文、书籍与资料是他分享给我的，同时每次组会都是产生知识的动力，因为deadline是第一生产力啊；接着还要感谢实验室前辈们，每次遇到疑惑都能够找他们并且能够很好的得到解决；感谢同届的哥儿仨一起探讨学术；最后再感谢一下阿摄儿，感谢他耐心的为我的博客美化图片。

参考资料

1. 项亮. 推荐系统实践[M]. 人民邮电出版社, 2012.
2. 周志华. 机器学习 : = Machine learning[M]. 清华大学出版社, 2016.
3. Koren Y, Bell R M, Volinsky C, et al. Matrix Factorization Techniques for Recommender Systems[J]. IEEE Computer, 2009, 42(8): 30-37.
4. Shi Y, Larson M, Hanjalic A. Collaborative Filtering beyond the User-Item Matrix[J]. Acm Computing Surveys, 2014, 47(1):1-45.
5. Domingos P. A few useful things to know about machine learning[J]. Communications of The ACM, 2012, 55(10): 78-87.