第三章

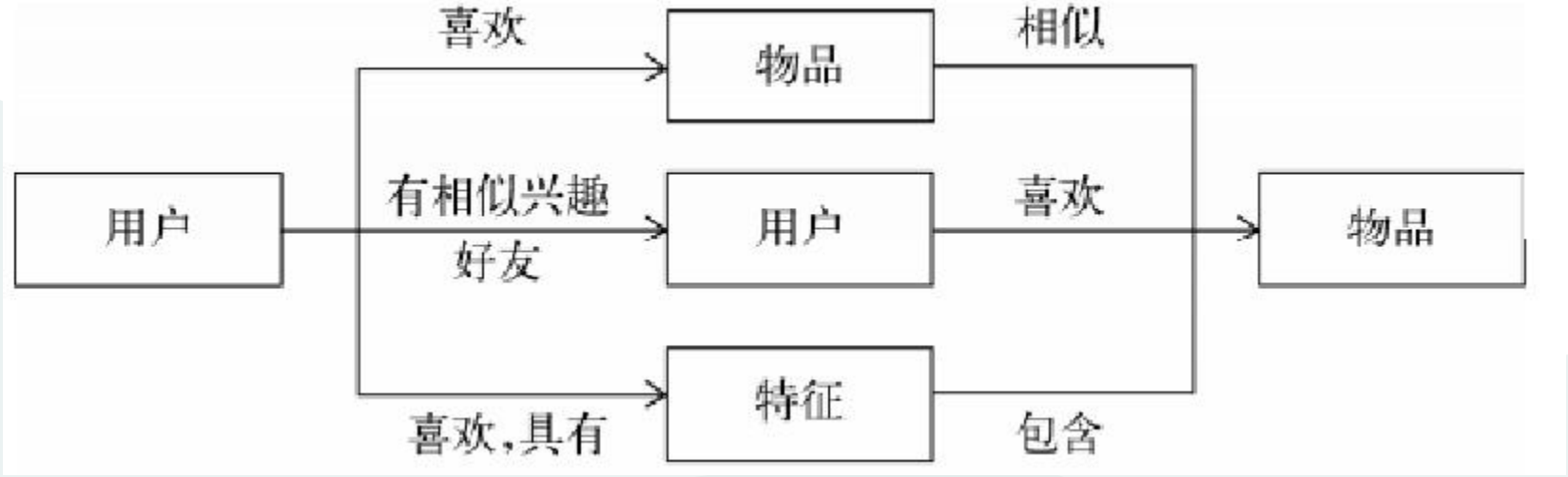
基于标签的推荐

教学目标

- 了解UGC 标签系统的代表应用
- 理解标签系统中的推荐问题
- 掌握基于标签的推荐系统
- 掌握给用户推荐标签

3 基于标签的推荐

- 推荐系统的目的是联系用户的兴趣和物品，这种联系需要依赖不同的媒介。目前流行的推荐系统基本上通过3种方式：
  - 第一种方式是利用用户喜欢过的物品。
  - 第二种方式是利用和用户兴趣相似的其他用户。
  - 第三种重要的方式是通过一些特征（feature）联系用户和物品，给用户推荐那些具有用户喜欢的特征的物品，其重要的特征表现方式——标签。



3 基于标签的推荐

- 标签是一种无层次化结构的、用来描述信息的关键词，它可以用来描述物品的语义。

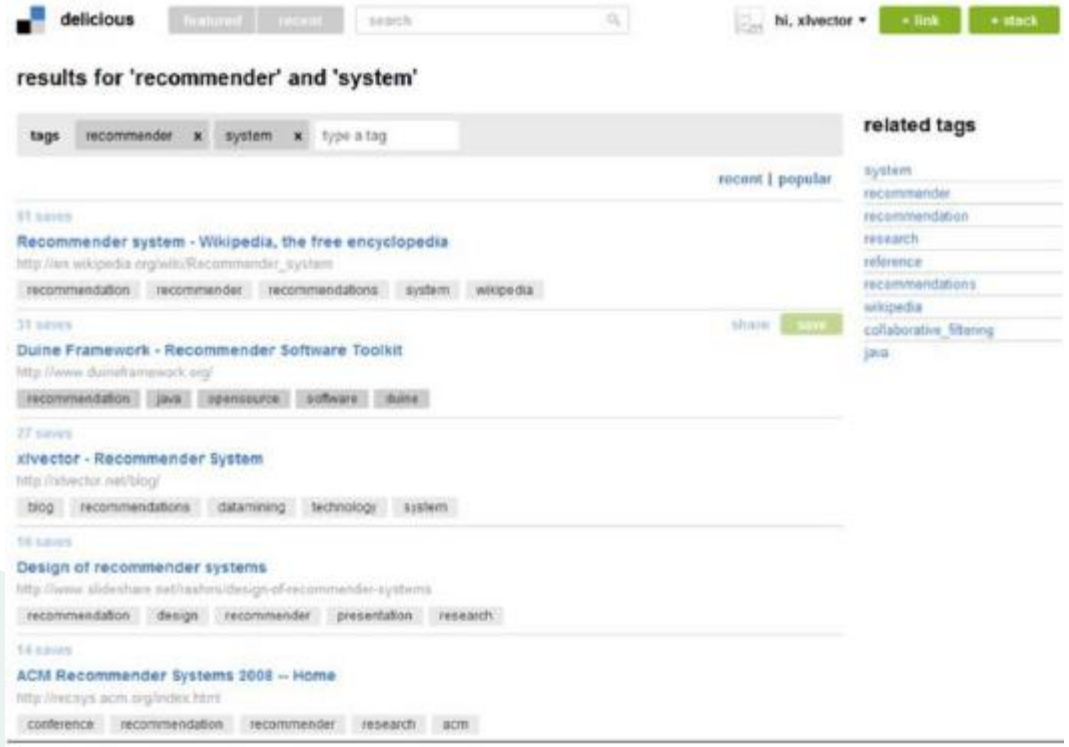
  根据给物品打标签的人的不同，标签应用一般分为两种：

  - 一种是让作者或者专家给物品打标签；
  - 另一种是让普通用户给物品打标签，也就是UGC（User Generated Content，用户生成的内容）的标签应用。UGC的标签系统是一种表示用户兴趣和物品语义的重要方式。

目录

3.1 标签系统的应用

- UGC标签系统是很多Web 2.0网站的必要组成部分。Delicous可算是标签系统里的开山鼻祖，它允许用户给互联网上的每个网页打标签，从而通过标签重新组织整个互联网。

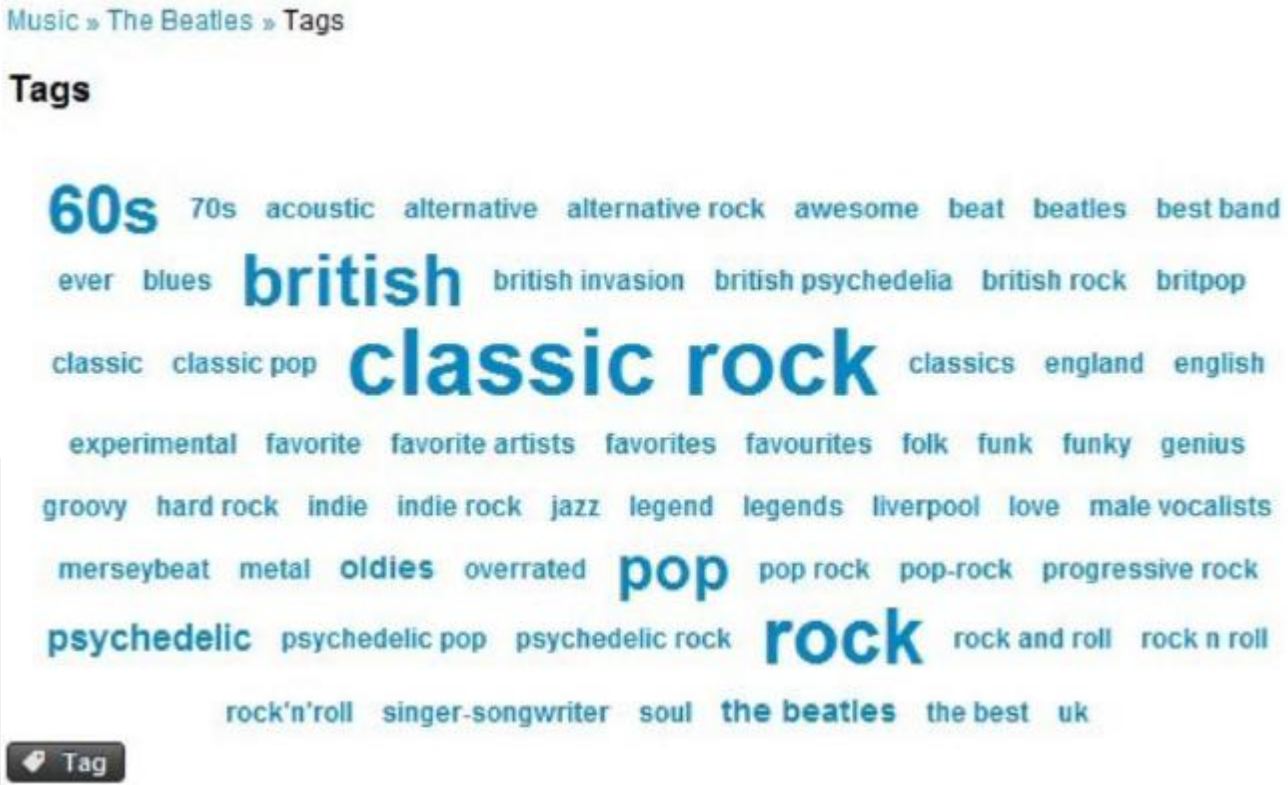


3.1 标签系统的应用

- CiteULike是一个著名的论文书签网站，它允许研究人员提交或者收藏自己感兴趣的论文并且给论文打标签，从而帮助用户更好地发现和自己研究领域相关的优秀论文。

3.1 标签系统的应用

- Last.fm是一家著名的音乐网站，它通过分析用户的听歌行为预测用户对音乐的兴趣，从而给用户推荐个性化的音乐。作为多媒体，音乐不像文本那样可以很容易地分析内容信息。为了在不进行复杂音频分析的情况下获得音乐的内容信息，Last.fm引入了UGC标签系统，让用户用标签标记音乐和歌手。



3.1 标签系统的应用

- 豆瓣是中国著名的评论和社交网站，同时也是中国个性化推荐领域的领军企业之一。豆瓣在个性化推荐领域进行了广泛尝试，标签系统也是其尝试的领域之一。它允许用户对图书和电影打标签，借此获得图书和电影的内容信息和语义，并用这种信息改善推荐效果。

3.1 标签系统的应用

- Hulu是美国著名的视频网站。视频作为一种最为复杂的多媒体，获取它的内容信息是最困难的，因此Hulu也引入了用户标签系统来让用户对电视剧和电影进行标记。

3.1 标签系统的应用

- 关于标签系统的作用:
  - 表达 标签系统帮助我表达对物品的看法。（30%的用户同意。）
  - 组织 打标签帮助我组织我喜欢的电影。（23%的用户同意。）
  - 学习 打标签帮助我增加对电影的了解。（27%的用户同意。）
  - 发现 标签系统使我更容易发现喜欢的电影。（19%的用户同意。）
  - 决策 标签系统帮助我判定是否看某一部电影。（14%的用户同意。）

目录

3.2 标签系统中的推荐问题

- 打标签作为一种重要的用户行为，蕴含了很多用户兴趣信息，因此深入研究和利用用户打标签的行为可以很好地指导我们改进个性化推荐系统的推荐质量。同时，标签的表示形式非常简单，便于很多算法处理。
- 标签系统中的推荐问题主要有以下两个：
  - 如何利用用户打标签的行为为其推荐物品（基于标签的推荐）？
  - 如何在用户给物品打标签时为其推荐适合该物品的标签（标签推荐）？
- 为了研究上面的两个问题，我们首先需要解答下面3个问题：
  - 用户为什么要打标签?
  - 用户怎么打标签?
  - 用户打什么样的标签?

3.2 标签系统中的推荐问题

- 用户为什么进行标注?
- 用户标注的动机问题
  - 首先是社会维度，有些用户标注是给内容上传者使用的（便于上传者组织自己的信息），而有些用户标注是给广大用户使用的（便于帮助其他用户找到信息）。
  - 另一个维度是功能维度，有些标注用于更好地组织内容，方便用户将来的查找，而另一些标注用于传达某种信息，比如照片的拍摄时间和地点等。

3.2 标签系统中的推荐问题

- 用户如何打标签?
- 尽管每个用户的行为看起来是随机的，但其实这些表面随机的行为背后蕴含着很多规律。用户行为数据集中用户活跃度和物品流行度的分布都遵循长尾分布 。

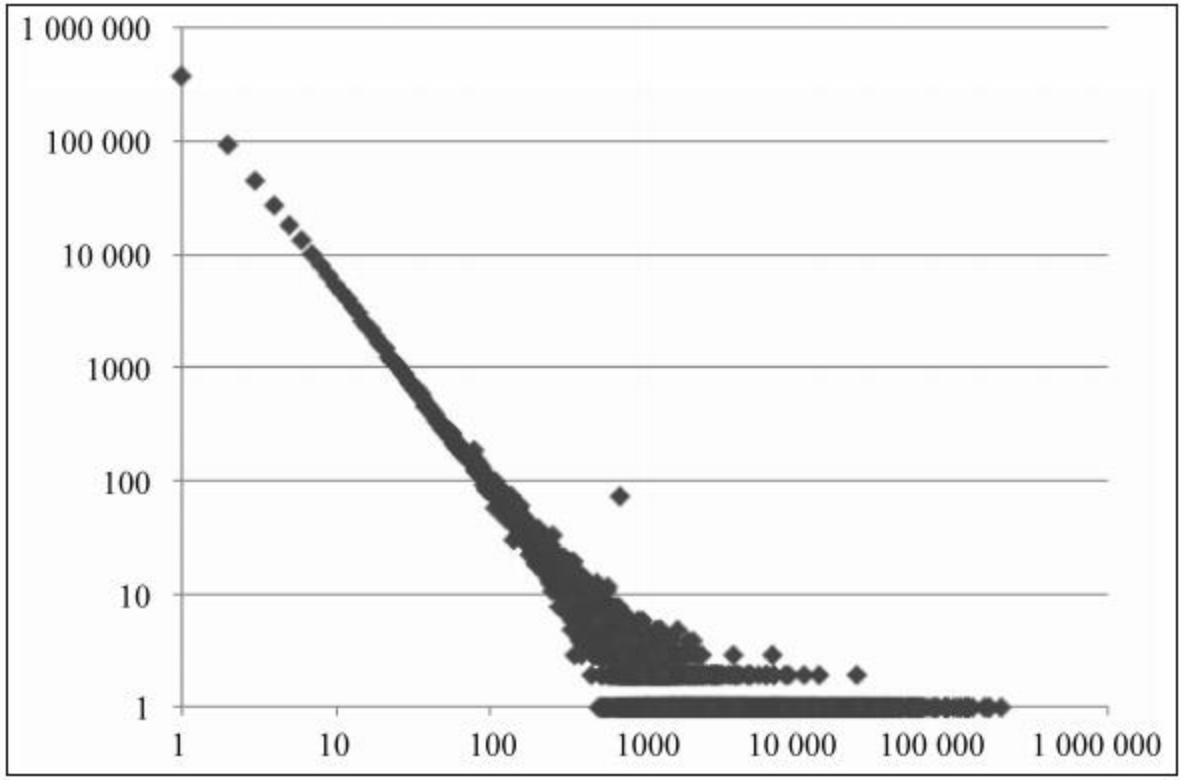


图4-8　标签流行度的长尾分布

3.2 标签系统中的推荐问题

- 用户打什么样的标签?
- 用户标签有很多种:
  - 表明物品是什么
  - 表明物品的种类
  - 表明谁拥有物品
  - 表达用户的观点
  - 用户相关的标签
  - 用户的任务

3.2 标签系统中的推荐问题

- 很多不同的网站也设计了自己的标签分类系统
  - 类型（Genre）
  - 时间（Time）
  - 人物（People）
  - 地点（Place）
  - 语言（Language）
  - 奖项（Awards）
  - 其他（Details）

目录

3.3 基于标签的推荐系统

- 用户用标签来描述对物品的看法，因此标签是联系用户和物品的纽带，也是反应用户兴趣的重要数据源，如何利用用户的标签数据提高个性化推荐结果的质量?
- 一个用户标签行为的数据集一般由一个三元组的集合表示，其中记录(u, i, b) 表示用户u给物品i打上了标签b，即用户的每一次打标签行为都用一个三元组（用户、物品、标签）表示。

3.3 基于标签的推荐系统

- 本章将采用两个不同的数据集评测基于标签的物品推荐算法。
  - Delicious数据集中包含用户对网页的标签记录。它每一行由4部分组成，即时间、用户ID、网页URL、标签。我们只抽取了其中用户对一些著名博客网站网页（Wordpress、BlogSpot、TechCrunch）的标签记录。
  - CiteULike数据集包含用户对论文的标签记录，它每行也由4部分组成，即物品ID、用户ID、时间、标签，我们选取了其中稠密的部分。

表4-1　Delicious和CiteULike数据集的基本信息

	用　户　数	物　品　数	标　签　数	记　录　数
Delicious	11 200	8791	42 233	405 665
CiteULike	12 466	7318	23 068	409 220

3.3 基于标签的推荐系统

- 本章将采用两个不同的数据集评测基于标签的物品推荐算法。

表4-2　Delicious和CiteULike数据集中最热门的20个标签

Delicious	CiteULike
wordpress	review
blog	network
blogs	bioinformatics
design	evolution
google	networks
howto	tagging
plugin	software
web2.0	sequencing
plugins	social
tutorial	genome
blogging	genomics
tips	statistics
art	ngs
music	folksonomy
linux	human
photography	mirna
webdesign	microarray
themes	expression
ubuntu	clustering
software	metagenomics

3.3 基于标签的推荐系统

- 本节将数据集随机分成10份。这里分割的键值是用户和物品，不包括标签。也就是说，用户对物品的多个标签记录要么都被分进训练集，要么都被分进测试集，不会一部分在训练集，另一部分在测试集中。然后，我们挑选1份作为测试集，剩下的9份作为训练集，通过学习训练集中的用户标签数据预测测试集上用户会给什么物品打标签。

3.3 基于标签的推荐系统

- 对于用户u，令$R(u)$为给用户u的长度为N的推荐列表，里面包含我们认为用户会打标签的物品。令$T(u)$是测试集中用户u实际上打过标签的物品集合。然后，我们利用准确率（precision）和召回率（recall）评测个性化推荐算法的精度。

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

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

- 将上面的实验进行10次，每次选择不同的测试集，然后将每次实验的准确率和召回率的平均值作为最终的评测结果。

- 为了全面评测个性化推荐的性能，我们同时评测了推荐结果的覆盖率（coverage）、多样性（diversity）和新颖度。
- 覆盖率的计算公式如下：

$$\text{Coverage} = \frac{|\bigcup_{u \in U} R(u)|}{|I|}$$

3.3 基于标签的推荐系统

- 多样性的定义取决于相似度的定义。我们用物品标签向量的余弦相似度度量物品之间的相似度。
- 推荐系统的多样性为所有用户推荐列表多样性的平均值。

每个物品i，item_tags[i]存储了物品i的标签向量，其中item_tags[i][b]是对物品i打标签b的次数，那么物品i和j的余弦相似度可以通过如下程序计算。

```
def Diversity(item_tags, recommend_items):
    ret = 0
    n = 0
    for i in recommend_items.keys():
        for j in recommend_items.keys():
            if i == j:
                continue
            ret += CosineSim(item_tags, i, j)
            n += 1
    return ret / (n * 1.0)
```

$$\text{Diversity} = 1 - \frac{\sum_{i \in R(u)} \sum_{j \in R(u), j \neq i} \text{Sim}(\text{item\_tags}[i], \text{item\_tags}[j])}{\binom{|R(u)|}{2}}$$

- 推荐新颖性用推荐结果的平均热门程度（AveragePopularity）度量。

对于物品i，定义它的流行度item_pop(i)为给这个物品打过标签的用户数。而对推荐系统，我们定义它的平均热门度如下：

$$\text{AveragePopularity} = \frac{\sum_{u}\sum_{i \in R(u)} \log(1+\text{item\_pop}(i))}{\sum_{u}\sum_{i \in R(u)} 1}$$

3.3 基于标签的推荐系统

- 拿到了用户标签行为数据，可以想到一个最简单的个性化推荐算法。这个算法的描述如下
  - 统计每个用户最常用的标签。
  - 对于每个标签，统计被打过这个标签次数最多的物品。
  - 对于一个用户，首先找到他常用的标签，然后找到具有这些标签的最热门物品推荐给这个用户。用户u对物品i的兴趣公式如下：

$$p(u,i)=\sum_{b} n_{u,b} n_{b,i}$$

$B(u)$是用户u打过的标签集合，$B(i)$是物品i被打过的标签集合，nu,b是用户u打过标签b的次数，nb,i是物品i被打过标签b的次数。

3.3 基于标签的推荐系统

在Python中，我们遵循如下约定：

- 用 records 存储标签数据的三元组，其中records[i] = [user, item, tag];
- 用 user_tags 存储nu,b，其中user_tags[u][b] = nu,b;
- 用 tag_items 存储nb,i，其中tag_items[b][i] = nb,i。

```
def InitStat(records):
            user_tags = dict()
            tag_items = dict()
            user_items = dict()
            for user, item, tag in records.items():
                addValueToMat(user_tags, user, tag, 1)
                addValueToMat(tag_items, tag, item, 1)
                addValueToMat(user_items, user, item, 1)
```

3.3 基于标签的推荐系统

在Python中，我们遵循如下约定：

- 用 records 存储标签数据的三元组，其中records[i] = [user, item, tag];
- 用 user_tags 存储nu,b，其中user_tags[u][b] = nu,b;
- 用 tag_items 存储nb,i，其中tag_items[b][i] = nb,i。

```
def Recommend(user):
        recommend_items = dict()
        tagged_items = user_items[user]
        for tag, wut in user_tags[user].items():
            for item, wti in tag_items[tag].items():
                #if items have been tagged, do not recommend them
                if item in tagged_items:
                    continue
                if item not in recommend_items:
                    recommend_items[item] = wut * wti
                else:
                    recommend_items[item] += wut * wti
        return recommend_items
```

3.3 基于标签的推荐系统

- 我们在Delicious数据集上对上面的算法进行评测

表4-3　基于标签的简单推荐算法在Delicious数据集上的评测结果

	召　回　率	准　确　率	覆　盖　率	多　样　性	平均热门程度
CiteULike	7.45%	2.25%	49.79%	0.7088	3.33
Delicious	7.19%	1.24%	19.05%	0.6073	5.22

3.3 基于标签的推荐系统

- 该算法通过如下公式预测用户u对物品i的兴趣：

$$p(u,i)=\sum_b n_{u,b}n_{b,i}$$

- 缺陷：
  - 这个公式倾向于给热门标签对应的热门物品很大的权重，因此会造成推荐热门的物品给用户，从而降低推荐结果的新颖性。
  - 另外，这个公式利用用户的标签向量对用户兴趣建模，其中每个标签都是用户使用过的标签，而标签的权重是用户使用该标签的次数。这种建模方法的缺点是给热门标签过大的权重，从而不能反应用户个性化的兴趣。

3.3 基于标签的推荐系统

- 这里我们可以借鉴TF-IDF的思想，对这一公式进行改进：

$$p(u,i)=\sum_{b}\frac{n_{u,b}}{\log(1+n_{b}^{(u)})}n_{b,i}$$

表4-4　Delicous和CiteULike数据集上TagBasedTFIDF的性能

	召　回　率	准　确　率	覆　盖　率	多　样　性	平均热门程度
CiteULike	11.02%	3.32%	63.92%	0.7469	3.20
Delicious	8.33%	1.43%	23.95%	0.6455	5.08

3.3 基于标签的推荐系统

- 我们也可以借鉴TF-IDF的思想对热门物品进行惩罚，从而得到如下公式：

$$p(u,i)=\sum_{b}\frac{n_{u,b}}{\log(1+n_{b}^{(u)})}\frac{n_{b,i}}{\log(1+n_{i}^{(u)})}$$

表4-5　Delicous和CiteULike数据集上TagBasedTFIDF++的性能

	召　回　率	准　确　率	覆　盖　率	多　样　性	平均热门程度
CiteULike	11.79%	3.56%	75.68%	0.7346	2.83
Delicious	8.88%	1.53%	36.98%	0.6338	4.83

3.3 基于标签的推荐系统

- 用户兴趣和物品的联系是通过中的标签建立的。但是，对于新用户或者新物品，这个集合中的标签数量会很少。为了提高推荐的准确率，我们可能要对标签集合做扩展。
- 标签扩展的本质是对每个标签找到和它相似的标签，也就是计算标签之间的相似度。最简单的相似度可以是同义词。如果有一个同义词词典，就可以根据这个词典进行标签扩展。如果没有这个词典，我们可以从数据中统计出标签的相似度。

3.3 基于标签的推荐系统

- 如果认为同一个物品上的不同标签具有某种相似度，那么当两个标签同时出现在很多物品的标签集合中时，我们就可以认为这两个标签具有较大的相似度。对于标签b，令$N(b)$为有标签b的物品的集合，$n_{b,i}$为给物品i打上标签b的用户数，我们可以通过如下余弦相似度公式计算标签b和标签b'的相似度：

$$\mathrm{sim}(b,b')=\frac{\sum_{i\in N(b)\cap N(b')} n_{b,i}n_{b',i}}{\sqrt{\sum_{i\in N(b)} n_{b,i}^2 \sum_{i\in N(b')} n_{b',i}^2}}$$

3.3 基于标签的推荐系统

表4-6　CiteULike数据集中recommender_system的相关标签

标　签	相 似 度
recommender_systems	0.558 394 161
recommender	0.415 820 788
recommendation	0.387 596 911
recsys	0.351 025 321
cf	0.328 168 796
multidimensional	0.324 232 233
recommend	0.318 880 412
collaborative_filtering	0.318 210 024
music_recommendation	0.305 214 504
recommenation_systems	0.281 284 339

表4-7　Delicious数据集中google的相关标签

标　签	相 似 度
search	0.533 522
searchengine	0.458 06
robots.txt	0.394 027
indexing	0.391 894
googlebot	0.382 861
search_engines	0.379 196
indexation	0.375 179
bots	0.375 179
opt-in/opt-out	0.375 08
web_index	0.375 08

表4-8　考虑标签扩展后的推荐性能

	召 回 率	准 确 率	覆 盖 率	多 样 性	平均热门程度
CiteULike	12.38%	3.74%	74.60%	0.7133	2.92
Delicious	9.04%	1.55%	37.09%	0.6261	4.85

3.3 基于标签的推荐系统

- 不是所有标签都能反应用户的兴趣，同时，标签系统里经常出现词形不同、词义相同的标签。
- 标签清理的另一个重要意义在于将标签作为推荐解释。首先，这些标签不能包含没有意义的停止词或者表示情绪的词，其次这些推荐解释里不能包含很多意义相同的词语。
- 一般来说有如下标签清理方法：
  - 去除词频很高的停止词；
  - 去除因词根不同造成的同义词；
  - 去除因分隔符造成的同义词。

3.3 基于标签的推荐系统

- 为了控制标签的质量，很多网站也采用了让用户进行反馈的思想，即让用户告诉系统某个标签是否合适。

3.3 基于图的推荐算法

- 利用图模型做基于标签数据的个性化推荐：
  - 首先，我们需要将用户打标签的行为表示到一张图上。图是由顶点、边和边上的权重组成的。定义3种不同的顶点，即用户顶点、物品顶点和标签顶点。
  - 然后，如果我们得到一个表示用户u给物品i打了标签b的用户标签行为(u, i, b)，那么最自然的想法就是在图中增加3条边，首先需要在用户u对应的顶点$v(u)$和物品i对应的顶点$v(i)$之间增加一条边，同理，在$v(u)$和$v(b)$之间需要增加一条边，$v(i)$和$v(b)$之间也需要边相连接。

3.3 基于图的推荐算法

- 利用图模型做基于标签数据的个性化推荐：

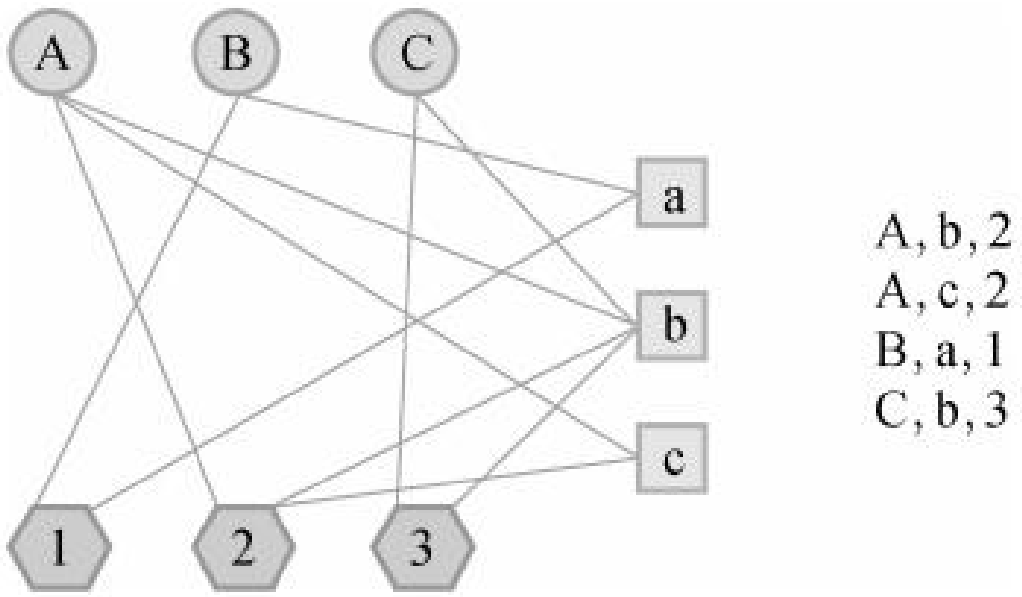


图4-11　简单的用户–物品–标签图的例子

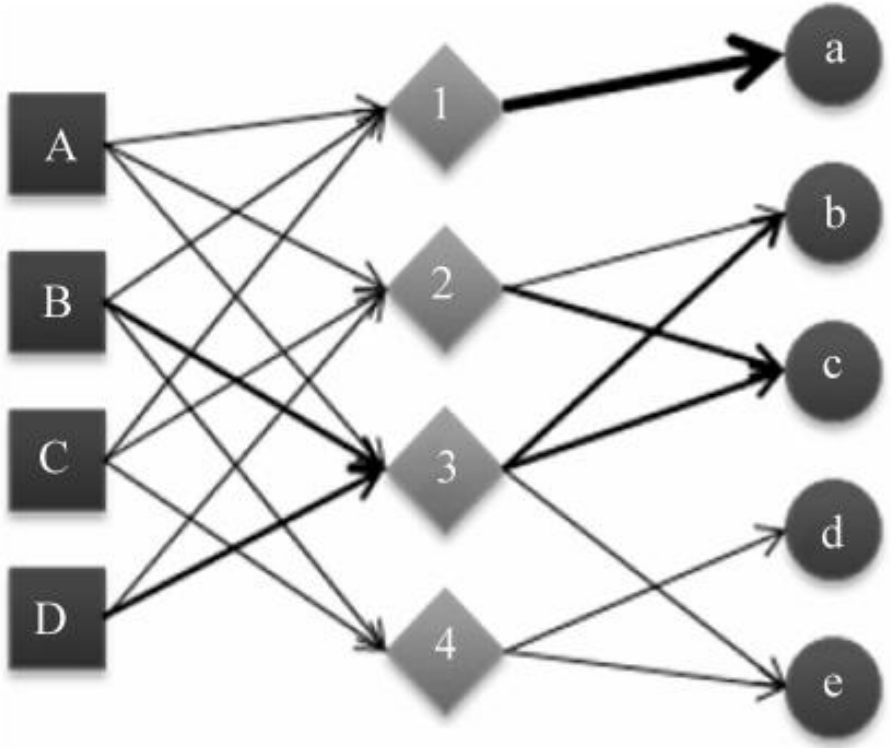
3.3 基于图的推荐算法

- 我们基于图模型重新思考前面提到的简单算法。用户对物品的兴趣公式如下:

$$P(i \mid u) = \sum_{b} P(i \mid b) P(b \mid u)$$

- 这个公式假定用户对物品的兴趣通过标签传递，因此这个公式可以通过一个比本节前面介绍的图更简单的图建模（记为SimpleTagGraph）。给定用户标签行为记录(u, i, b)，SimpleTagGraph会增加两条有向边，一条由用户节点$v(u)$指向标签节点$v(b)$，另一条由标签节点$v(b)$指向物品节点$v(i)$。从这个定义可以看到，SimpleTagGraph相对于前面提到用户—物品—标签图少了用户节点和物品节点之间的边。

3.3 基于图的推荐算法



$(A, a, 1)(A, c, 2)(A, c, 3)$

$(B, a, 1)(B, b, 3)(B, e, 3)(B, e, 4)$

$(C, a, 1)(C, b, 2)(C, d, 4)$

$(D, b, 3)(D, c, 2)(D, c, 3)$

3.3 基于标签的推荐解释

- 基于标签的推荐其最大好处是可以利用标签做推荐解释，这方面的代表性应用是豆瓣的个性化推荐系统。

3.3 基于标签的推荐解释

- 这样组织推荐结果页面有很多好处:
  - 首先是提高了推荐结果的多样性。
  - 标签云展示了用户的所有兴趣，然后让用户自己根据他今天的兴趣选择相关的标签，得到推荐结果，从而极大地提高了推荐结果的多样性，使得推荐结果更容易满足用户多样的兴趣。
  - 标签云也提供了推荐解释功能。

3.3 基于标签的推荐解释

- Jesse Vig将用户和物品之间的关系变成了用户对标签的兴趣（tag preference）和标签与物品的相关度（tag relevance），设计了4种标签解释:
  - RelSort　对推荐物品做解释时使用的是用户以前使用过且物品上有的标签，给出了用户对标签的兴趣和标签与物品的相关度，但标签按照和物品的相关度排序。
  - PrefSort　对推荐物品做解释时使用的是用户以前使用过且物品上有的标签，给出了用 户对标签的兴趣和标签与物品的相关度，但标签按照用户的兴趣程度排序。
  - RelOnly　对推荐物品做解释时使用的是用户以前使用过且物品上有的标签，给出了标签与物品的相关度，且标签按照和物品的相关度排序。
  - PrefOnly　对推荐物品做解释时使用的是用户以前使用过且物品上有的标签，给出了用 户对标签的兴趣程度，且标签按照用户的兴趣程度排序。

3.3 基于标签的推荐解释

表4-9　10个用户最满意的主观类标签[①]

标　　签	认为这个标签好的人所占比例
great soundtrack	90.9%
fanciful	90.9%
funny	90.0%
poignant	88.9%
witty	88.0%
dreamlike	87.5%
whimsical	87.5%
dark	87.3%
surreal	86.7%
deadpan	84.2%

3.3 基于标签的推荐解释

表4-10　10个用户最满意的客观类标签[②]

标　　签	认为这个标签好的人所占比例
afi-100	100.0%
fantasy world	100.0%
world war ii	100.0%
sci-fi	95.2%
action	94.4%
psychology	93.8%
disney	91.7%
satirical	88.5%
drama	87.5%
satire	86.4%

3.3 基于标签的推荐解释

- 总结问卷调查的结果，得出以下结论：
  - 用户对标签的兴趣对帮助用户理解为什么给他推荐某个物品更有帮助；
  - 用户对标签的兴趣和物品标签相关度对于帮助用户判定自己是否喜欢被推荐物品具有同样的作用；
  - 物品标签相关度对于帮助用户判定被推荐物品是否符合他当前的兴趣更有帮助；
  - 客观事实类标签相比主观感受类标签对用户更有作用。

目录

3.4 给用户推荐标签

- 当用户浏览某个物品时，标签系统非常希望用户能够给这个物品打上高质量的标签，这样才能促进标签系统的良性循环。因此，很多标签系统都设计了标签推荐模块给用户推荐标签。

3.4 给用户推荐标签

- 给用户推荐标签的好处:
  - **方便用户输入标签**　让用户从键盘输入标签无疑会增加用户打标签的难度，这样很多用户不愿意给物品打标签，因此我们需要一个辅助工具来减小用户打标签的难度，从而提高用户打标签的参与度。
  - **提高标签质量**　同一个语义不同的用户可能用不同的词语来表示。这些同义词会使标签的词表变得很庞大，而且会使计算相似度不太准确。而使用推荐标签时，我们可以对词表进行选择，首先保证词表不出现太多的同义词，同时保证出现的词都是一些比较热门的、有代表性的词。

3.4 给用户推荐标签

- 用户u给物品i打标签时，我们有很多方法可以给用户推荐和物品i相关的标签。
  - 第0种方法就是给用户u推荐整个系统里最热门的标签
  - 第1种方法就是给用户u推荐物品i上最热门的标签
  - 第2种方法是给用户u推荐他自己经常使用的标签
  - 第3种算法是前面两种的融合

3.4 给用户推荐标签

- 和前面的实验一样，我们用同样的方法将数据集按照9：1分成训练集和测试集，然后通过训练集学习用户标注的模型。需要注意的是，这里切分数据集不再是以user、item为主键，而是以user、item、tag为主键。

```
def SplitData(records, train, test):
    for user,item, tag in records:
        if random.randint(1,10) == 1:
            test.append([user,item,tag])
        else:
            train.append([user,item,tag])
    return [train, test]
```

3.4 给用户推荐标签

- 对于测试集中的每一个用户物品对(u, i)，我们都会推荐N个标签给用户u作参考。令$R(u, i)$为我们给用户u推荐的应该在物品i上打的标签集合，令$T(u, i)$为用户u实际给物品i打的标签的集合，我们可以利用准确率和召回率评测标签推荐的精度：

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

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

3.4 给用户推荐标签

实验结果

表4-11列出了PopularTags、UserPopularTags、ItemPopularTags 3种算法在$N = 10$ 时的准确率和召回率。

表4-11　3种标签推荐算法在N=10时的准确率和召回率

	Delicious		
	PopularTags	UserPopularTags	ItemPopularTags
准确率	7.32%	11.84%	23.80%
召回率	19.88%	32.16%	64.63%
	CiteULike		
准确率	2.21%	10.85%	12.94%
召回率	7.75%	38.00%	45.33%

3.4 给用户推荐标签

表4-12　HybridPopularTags算法在不同线性融合系数 α下的准确率和召回率

	Delicious		CiteULike	
α	准确率	召回率	准确率	召回率
0.0	11.84%	32.16%	10.85%	38.00%
0.1	15.27%	41.48%	12.71%	44.53%
0.2	16.71%	45.39%	13.82%	48.42%
0.3	18.93%	51.41%	14.85%	52.04%
	Delicious		CiteULike	
α	准确率	召回率	准确率	召回率
0.4	21.14%	57.42%	15.57%	54.55%
0.5	22.74%	61.75%	16.01%	56.07%
0.6	23.99%	65.15%	16.24%	56.90%
0.7	24.82%	67.42%	16.07%	56.29%
0.8	25.15%	68.30%	15.45%	54.12%
0.9	24.95%	67.77%	14.60%	51.15%
1.0	23.80%	64.63%	12.94%	45.33%

3.4 给用户推荐标签

- 图模型同样可以用于标签推荐。在根据用户打标签的行为生成图之后，可以利用PersonalRank算法进行排名。当用户u遇到物品i时，会给物品i打什么标签。我们可以重新定义顶点的启动概率：

$$r_{v(k)}=\begin{cases}\alpha(v(k)=v(u))\\1-\alpha(v(k)=v(i))\\0\quad(\text{其他})\end{cases}$$

- 只有用户u和物品i对应的顶点有非0的启动概率，而其他顶点的启动概率都为0。在上面的定义中，$v(u)$和$v(i)$的启动概率并不相同，$v(u)$的启动概率是a，而$v(i)$的启动概率是1-a。参数α可以通过离线实验选择。

内容回顾

1、标签系统的应用

2、标签系统中的推荐问题

3、基于标签的推荐系统

4、给用户推荐标签

Thank you!