



基于协同过滤技术的个性化推荐系统

指导老师：王靖

汇报人：胡鑫 汇报时间：2018年05月

课题背景及意义



信息爆炸

在这个信息爆炸的网络时代，面对纷杂的信息资源我们感到无所适从。



信息找人

结合个人习性爱好改变传统的“人找信息”的状况，打造出“信息找人”的格局



以网页为推荐对象

采用Web数据挖掘的方法和技术，为用户推荐符合其兴趣爱好的网页



以物品为推荐对象

为用户推荐符合兴趣爱好的物品

基于协同过滤推荐算法的电影推荐系统

- 学习协同过滤算法的具体实现
- 能够将算法的计算结果形象地展示出来
- 了解其他推荐算法

毕设课题内容



数据集的获取



核心推荐算法的实现




推荐结果展示



不同算法之间的比较



环境搭建

- Ubuntu
 - JDK1.6.0_21
 - MySQL
 - apache-tomcat-6.0.35
 - mahout-0.3
 - MyEclipse 8.0
- 



Mahout介绍

Mahout 是 Apache Software Foundation (ASF) 旗下的一个开源项目，提供一些可扩展的机器学习领域经典算法的实现，旨在帮助开发人员更加方便快捷地创建智能应用程序。经典算法包括聚类、分类、协同过滤、进化编程等等，并且，在 Mahout 中还加入了对Apache Hadoop的支持，使这些算法可以更高效的运行在云计算环境中。





数据集的获取

- MovieLens
 - movies.dat
 - ratings.dat
 - users.dat

数据集的获取

```
movies.dat ✕
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
```

```
ratings.dat ✕
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
```

```
users.dat ✕
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614
10::F::35::1::95370
```

- movies.dat的文件描述是 电影编号::电影名::电影类别
- ratings.dat的文件描述是 用户编号::电影编号::电影评分::时间戳
- users.dat的文件描述是 用户编号::性别::年龄::职业::Zip-code

数据库

movie_preferences

userID	movieID	preference	timestamp
1	1193	5	978300760
1	661	3	978302109
1	914	3	978301968
1	3408	4	978300275
1	2355	5	978824291
1	1197	3	978302268
1	1287	5	978302039
1	2804	5	978300719
1	594	4	978302268
1	919	4	978301368
1	595	5	978824268
1	938	4	978301752
1	2398	4	978302281
1	2918	4	978302124
1	1035	5	978301753
1	2791	4	978302188
1	2687	3	978824268
1	2018	4	978301777
1	3105	5	978301713
1	2797	4	978302039
1	2321	3	978302205
1	720	3	978300760
1	1270	5	978300055

1000 rows fetched in 0:00.3940

movies

id	name	published_year	type
1	Toy Story	1995	Animation, Children's, Comedy
2	Jumanji	1995	Adventure, Children's, Fantasy
3	Grumpier Old Men	1995	Comedy, Romance
4	Waiting to Exhale	1995	Comedy, Drama
5	Father of the Bride Part II	1995	Comedy
6	Heat	1995	Action, Crime, Thriller
7	Sabrina	1995	Comedy, Romance
8	Tom and Huck	1995	Adventure, Children's
9	Sudden Death	1995	Action
10	GoldenEye	1995	Action, Adventure, Thriller
11	American President, The	1995	Comedy, Drama, Romance
12	Dracula: Dead and Loving It	1995	Comedy, Horror
13	Balto	1995	Animation, Children's
14	Nixon	1995	Drama
15	Cutthroat Island	1995	Action, Adventure, Romance
16	Casino	1995	Drama, Thriller
17	Sense and Sensibility	1995	Drama, Romance
18	Four Rooms	1995	Thriller
19	Ace Ventura: When Nature Calls	1995	Comedy
20	Money Train	1995	Action
21	Get Shorty	1995	Action, Comedy, Drama
22	Copycat	1995	Crime, Drama, Thriller
23	Assassins	1995	Thriller

- Taste
 - Apache Mahout 提供的一个协同过滤算法的高效实现，它是一个基于 Java 实现的可扩展的，高效的推荐引擎。
 - 基于用户相似度推荐
 - 基于内容相似度推荐

基于用户相似度

```
public class MyUserBasedRecommender {  
    public List<RecommendedItem> userBasedRecommender(long userID,int size) {  
  
        List<RecommendedItem> recommendations = null;  
        try {  
            DataModel model = MyDataModel.myDataModel();//构造数据模型  
            UserSimilarity similarity = new PearsonCorrelationSimilarity(model);//用PearsonCorrelation 算法计算用户相  
            UserNeighborhood neighborhood = new NearestNUserNeighborhood(3, similarity, model);//计算用户的“邻居”，这  
            Recommender recommender = new CachingRecommender(new GenericUserBasedRecommender(model, neighborhood, s  
            recommendations = recommender.recommend(userID, size);//得到推荐的结果，size是推荐结果的数目  
        } catch (Exception e) {  
            // TODO: handle exception  
            e.printStackTrace();  
        }  
        return recommendations;  
    }  
  
    public static void main(String args[]) throws Exception {  
  
    }  
}
```

基于用户相似度

你的电影				推荐的电影					
电影名称	上映年份	电影类型	评分	电影海报	电影名称	上映年份	电影类型	评分	
One Flew Over the Cuckoo's Nest	1975	Drama	5.0		Sleepless in Seattle	1993	Comedy, Romance	5.0	
Awakenings	1990	Drama	5.0		Odd Couple II, The	1998	Comedy	5.0	
									

基于内容相似度

```
public class MyItemBasedRecommender {  
  
    public List<RecommendedItem> myItemBasedRecommender(long userID,int size){  
        List<RecommendedItem> recommendations = null;  
        try {  
            DataModel model = new FileDataModel(new File("D:/ml-1m/movie_preferences.txt")); //构造数据模型  
            ItemSimilarity similarity = new PearsonCorrelationSimilarity(model); //计算内容相似度  
            Recommender recommender = new GenericItemBasedRecommender(model, similarity); //构造推荐引擎  
            recommendations = recommender.recommend(userID, size); //得到推荐结果  
        } catch (Exception e) {  
            // TODO: handle exception  
            e.printStackTrace();  
        }  
        return recommendations;  
    }  
}
```

基于内容相似度

你的电影

电影名称

上映年
份

电影类型

评
分

电影海报

One Flew Over the
Cuckoo's Nest

1975

Drama

5.0

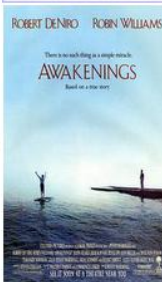


Awakenings

1990

Drama

5.0



推荐的电影

电影名称

上映年
份

电影类型

评
分

Teaching Mrs. Tingle

1999

Comedy, Thriller

5.0

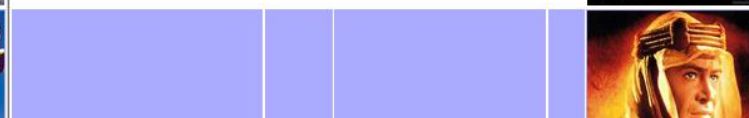


Fugitive, The

1993

Action, Thriller

5.0






算法比较

基于用户相似度


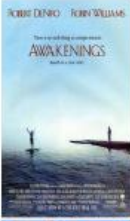

- 计算出用户之间的相似性
- 筛选出具有共同兴趣的邻居用户
- 从邻居用户喜欢的，且目标用户没有评价过的电影推荐给目标用户。

基于内容相似度

- 计算出电影之间的相似性
 - 筛选出与目标用户喜欢的电影相似度高的电影推荐给目标用户。
- 

算法比较

基于用户相似度

电影海报	推荐的电影	上映年份	电影类型	评分
	Sleepless in Seattle	1993	Comedy, Romance	5.0
	Odd Couple II, The	1998	Comedy	5.0
				

基于内容相似度

电影海报	推荐的电影	上映年份	电影类型	评分
	Teaching Mrs. Tingle	1999	Comedy, Thriller	5.0
	Fugitive, The	1993	Action, Thriller	5.0
				



THANKS