中国矿业大学计算机学院

2020 级本科生课程报告

课程名称 信息内容安全

报告题目 基于 Django 的图书推荐系统与协同过滤算法优化

报告时间 2023.6.12

姓　　名 周子欣

任课教师 曹天杰

课程报告分工

姓名	学号	主要工作
周子欣	**12204202**	**系统搭建，算法优化，算法对比测试**

课程报告评分表

姓名	学号	目标 2	目标 3	课程报告成绩
周子欣	**12204202**			

课程报告考查方式与考查点

序号	毕业要求	课程教学目标	考查方式与考查点	得分
1	3.2	**目标 2**：掌握信息内容安全处理相关的理论、技术以及健全的评价体系，能够根据具体问题分析算法、设计算法、实现算法并能综合评价算法。	对课程报告的 PPT 讲解、系统演示与答辩； 考查报告中涉及信息内容安全算法的合理性、创新性	50
2	4.3	**目标 3**：掌握信息内容安全的基础知识，针对具体问题和要求选择正确的技术路线，通过在实验环境中进行仿真实验并能根据算法特点进行攻击测试和综合性能评价，得到具有参考价值的结论。	课程报告； 综合解决以下问题：选择信息内容安全专题，完成信息内容安全系统，选择相应的技术进行算法设计并在实验环境中进行仿真实验和性能评价，得到有效结论。	50
总分占总评成绩 70%（其余平时成绩占 30%）				100

评阅人：　曹天杰

2023 年　7 月　10 日

报告中文摘要

本次工作主要设计了基于 Django 的图书推荐系统平台，并对其用户协同过滤算法进行优化。针对热门图书对用户喜爱度的相似性作用不大的问题，改进其余弦相似度，加上对热门用户的惩戒；针对推荐系统由于用户评分少，导致的相似度矩阵稀疏，影响推荐性能的问题，为了缓解其稀疏性，基于 LDA 主题模型构建用户画像对算法进行改进，以来判断潜在空间中用户的主题偏好，即对书籍摘要利用 LDA 主题模型输出图书的主题分布，将用户评分过的图书的主题相加得到用户偏好的主题分布，计算用户间主题分布的相似度与用户评分矩阵相似度结合得到的相似度来进行推荐。

关键词：图书推荐 ；LDA 主题模型 ；协同过滤

报告正文

一. 基于用户的协同过滤算法的优化

1.改进余弦相似度

需要改进的原因：一般的基于用户的协同过滤算法，是直接用余弦相似度计算用户相似度，但其实对于热门图书，并不能反映用户喜爱的相似度，所有要对热门图书进行惩罚。

改进后的计算公式如下：

$$w_{uv} = \frac{\sum_{i \in N(u) \bigcap N(v)} \frac{1}{log1+|N(i)|}}{\sqrt{|N(u)||N(v)|}}$$

其中 N(i) 是对物品 i 有过评分行为的用户结合

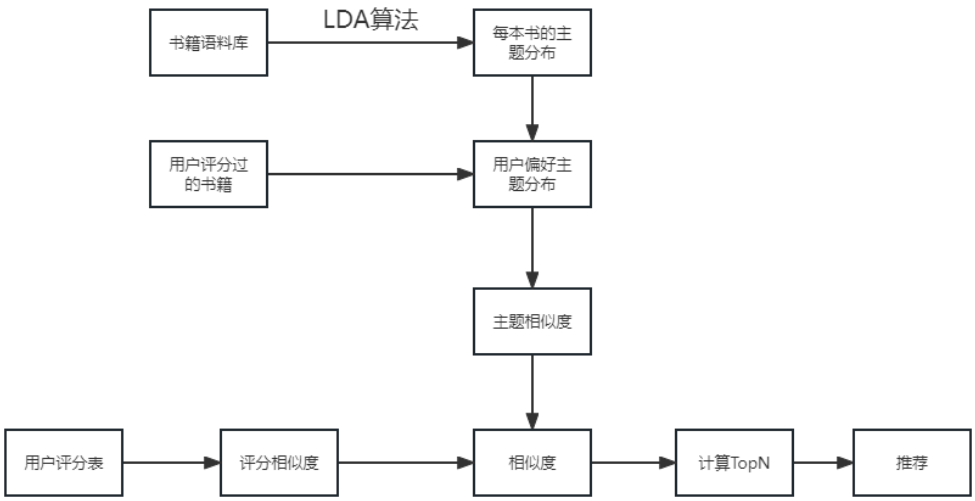
2.基于 LDA 主题模型构建用户画像

需要改进的原因：协同过滤算法完全没有利用到物品本身或者是用户自身的属性， 仅仅利用了用户与物品的交互信息就可以实现推荐，是一个可解释性很强， 非常直观的模型。存在处理稀疏矩阵的能力比较弱的问题。我们可以用隐语义模型挖掘用户和物品的隐含兴趣和隐含特征， 在一定程度上弥补协同过滤模型处理稀疏矩阵能力不足的问题。这里我采用隐语义模型中的 LDA 主题模型，挖掘用户偏好的主题分布。

LDA 主题模型：LDA 主题模型通过先验假设每个文档包含多个主题，并将单词通过贝叶斯统计推断文档的主题分布，即哪些主题在文档中的占比情况。在模型训练过程中，首先初始化每篇文档的主题分布及每个主题的单词分布，随后对每个单词进行抽样，更新主题分布和单词分布，在不断迭代中使得主题分布和单词分布最优。通过这种方法，模型最终可以得到每篇文档中不同主题的分布情况，从而更好地理解文本数据并进行分类、检索等任务。

LDA 主题模型优势：(1)文档结构化，相比传统词袋模型达到了降维的效果 (2)完成了文档的聚类和词汇的聚类，实现文本信息的抽象化分析，帮助分析者探索隐含语义。

基于 LDA 主题模型构建用户画像并推荐的过程：对书籍摘要利用 LDA 主题模型输出图书的主题分布，将用户评分过的图书的主题相加得到用户偏好的主题分布，计算用户间主题分布的相似度与用户评分矩阵相似度结合得到的相似度来进行推荐。**流程图**如下：



3.对比测试

3.1 数据集介绍(链接在文末)

BX-Books.csv：包含了书本的 id 和标题，作者，发表年份，封面连接。

```
"ISBN";"Book-Title";"Book-Author";"Year-Of-Publication";"Publisher";"Image-URL-S";"Image-URL-M";"Image-URL-L"
"0195153448";"Classical Mythology";"Mark P. O. Morford";"2002";"Oxford University Press";"http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/0195153448.01.MZZZZZZZ.jpg";"http://images.amazon.com/images/P/0195153448.01.LZZZZZZZ.jpg"
"0002005018";"Clara Callan";"Richard Bruce Wright";"2001";"HarperFlamingo Canada";"http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg";"http://images.amazon.com/images/P/0002005018.01.LZZZZZZZ.jpg"
"0060973129";"Decision in Normandy";"Carlo D'Este";"1991";"HarperPerennial";"http://images.amazon.com/images/P/0060973129.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/0060973129.01.MZZZZZZZ.jpg";"http://images.amazon.com/images/P/0060973129.01.LZZZZZZZ.jpg"
"0374157065";"Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It";"Gina Bari Kolata";"1999";"Farrar Straus Giroux";"http://images.amazon.com/images/P/0374157065.01.THUMBZZZ.jpg";"http://images.amazon.com/images/P/0374157065.01.MZZZZZZZ.jpg";"http://images.amazon.com/images/P/0374157065.01.LZZZZZZZ.jpg"
"0393045218";"The Mummies of Urumchi";"E. J. W. Barber";"1999";"W. W. Norton &amp;amp; Company";"http://images.amazon.com/images/P/
```

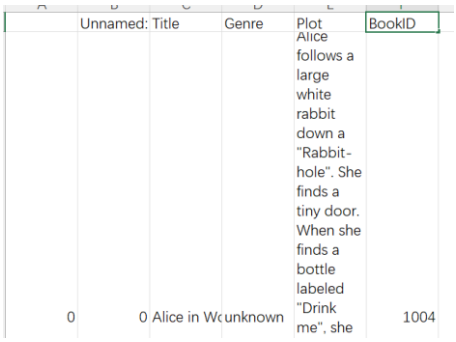
BX-Users.csv：包含了用户 id 以及用户的地址年龄信息

	A	B	C
1	User-ID;"Location";"Age"		
2	1;"nyc	new york	usa";NULL
3	2;"stockton	california	usa";"18"
4	3;"moscow	yukon territor	russia";NULL
5	4;"porto	v.n.gaia	portugal";"17"
6	5;"farnborough	hants	united kingdom";NULL
7	6;"santa monica	california	usa";"61"
8	7;"washington	dc	usa";NULL
9	8;"timmins	ontario	canada";NULL
10	9;"germantown	tennessee	usa";NULL
11	10;"albacete	wisconsin	spain";"26"
12	11;"melbourne	victoria	australia";"14"
13	12;"fort bragg	california	usa";NULL

BX-Book-Ratings.csv：包含了用户对于书本的评分

	A
1	User-ID;"ISBN";"Book-Rating"
2	276725;"034545104X";"0"
3	276726;"0155061224";"5"
4	276727;"0446520802";"0"
5	276729;"052165615X";"3"
6	276729;"0521795028";"6"
7	276733;"2080674722";"0"
8	276736;"3257224281";"8"

bookInfo.csv：从 Wikipedia 抓取的书本情节摘要，该数据集包含来自世界各地的 34,886 本书的描述。包含书的名字，类别，摘要，ID



	Unnamed:	Title	Genre	Plot	BookID
0	0	Alice in Wo	unknown	Alice follows a large white rabbit down a "Rabbit-hole". She finds a tiny door. When she finds a bottle labeled "Drink me", she	1004

3.2 基于改进的余弦相似度算法

代码实现如下

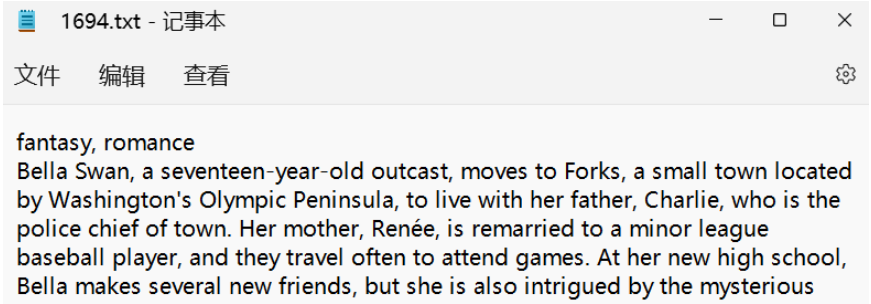
```
# 改进的兴趣相似度计算公式
def ImprovedSimilarity(train):
    # build inverted table for item_users. 倒排表
    item_users = dict()
    for u, items in train.items():
        for i in items:
            if i not in item_users:
                item_users[i] = set()
            item_users[i].add(u)

    # calculate co-rated items between users
    C = dict()
    N = dict()
    for i, users in item_users.items():
        for u in users:
            if u not in C:
                C[u] = dict()
            if u not in N:
                N[u] = 1
            else:
                N[u] += 1  # 记录每个用户的item个数
            for v in users:
                if u == v:
                    continue
                if v not in C[u]:
                    C[u][v] = 1 / math.log(1 + len(users))  # 惩罚用户u和用户v共同兴趣列表中热门物品对他们相似度的影响
                else:
                    C[u][v] += 1 / math.log(1 + len(users))

    # calculate final similarity matrix W. 计算相似度
    W = dict()
    for u, related_users in C.items():
        if u not in W:
            W[u] = dict()
        for v, cuv in related_users.items():
            W[u][v] = cuv / math.sqrt(N[u] * N[v])
    return W
```

3.3 基于 LDA 主题模型的协同过滤推荐过程

(1)使用 bookInfo.csv 和 BX-Users.csv 创建语料库，输出为 updatedCorpus.zip，其包含 bookInfo.csv 和 BX-Users.csv 数据集共同包含的书籍的分类和摘要信息。每个书籍对应一个 txt 文件，其文件名以 bookID 命名。



(2)利用 mallet 自然语言处理工具包，在 updatedCorpus.zip 语料库上运行 LDA 算法，得到包含每部书籍主题分布的 mallet.csv。

bin/mallet train-topics --input ./ updatedCorpus.zip --num-topics 100 --output-topic-keys topic-keys.txt --output-doc-topics mallet.csv

0	3644	0.092901235	0.006481481	3.09E-04
1	1053	2.38E-04	2.38E-04	2.38E-04
2	2228	0.127312139	1.45E-04	1.45E-04
3	3136	1.90E-04	1.90E-04	1.90E-04
4	1721	0.150990854	7.62E-05	7.62E-05
5	1047	3.01E-04	3.01E-04	0.012349398
6	3650	1.18E-04	1.18E-04	1.18E-04
7	2200	6.17E-04	6.17E-04	6.17E-04

mallet.csv

(3)构建相似度矩阵。最终评价的相似度由用户间的评分相似度和用户间的主题相似度相乘得到。

构建**主题**相似度矩阵：

①通过将所有书籍的评分相加来创建所有用户配置文件(每个用户对应一个主题分布)，调用函数以生成特定用户的用户配置文件 user_profiles.p

```
defaultdict(<class 'list'>, {0: 0.0, 1: array([3.49967968, 0.22110885, 0.03123702, 0.34828043, 0.2429105 ,
       0.11548064, 0.0188283 , 0.35143826, 0.48048158, 0.50030087,
       0.19213283, 0.09435048, 0.74801238, 0.14838269, 0.07148296,
       0.2301937 , 0.22117707, 0.20686718, 0.0747037 , 0.33769691,
       0.06259973, 0.18553465, 0.40665763, 0.2502967 , 0.3343401 ,
       0.45762386, 0.11702913, 0.74953613, 0.07962003, 0.32266397,
       0.34602663, 0.49510237, 0.11899022, 1.93737139, 0.30703933,
       0.51332848, 0.02471701, 0.01667461, 0.26336958, 0.20780093,
       1.82806856, 0.03187296, 0.10597656, 0.04179696, 0.14958925,
       0.06219368, 0.1294453 , 0.44023756, 0.02669038, 0.38065921,
       0.04772287, 0.13586841, 0.14810782, 0.13096943, 1.55129313,
       0.10361647, 0.04641237, 0.53702598, 0.16688661, 0.19024868,
       0.29054775, 0.15050448, 0.01669367, 1.0082727 , 0.6433504 ,
       0.18332893, 0.04907088, 0.17289484, 0.32187477, 0.15879927,
       0.44156702, 0.0903809 , 0.02375025, 0.02822416, 0.66021733,
       0.47155607, 0.26359878, 0.39893791, 0.16111197, 0.08095524,
       0.1938174 , 0.3467174 , 0.64046958, 0.26553423, 0.27536066,
       0.21061966, 0.11656159, 0.3380712 , 0.42132812, 0.19171978,
       0.35664886, 0.0706265 , 1.21788407, 7.87131025, 0.01827092,
       2.72743582, 0.04458108, 0.06143597, 1.15967044, 0.27064459]), 2: array([ 5.75990789,  1.56137392,  0.
```

user_profiles.p 内容

②计算主题相似度:计算两个用户主题向量的交叉熵转换为概率值

```
# 计算主题向量相似度
def topic_similarity(u1, u2):
    entropy = sc.entropy(u1, u2)
    return math.exp(entropy)
```

构建**评分**相似度矩阵：

```
def load_ratings_matrix():
    path = 'ratings.csv'
    ratings_df = pd.read_csv(path)
    ratings_mat = np.zeros((6041, 3953), dtype=int)
    for index, row in ratings_df.iterrows():
        ratings_mat[row['user_id'], row['book_id']] = row['rating']
    np.savetxt('ratings_mat.csv', ratings_mat, delimiter=',')
    return ratings_mat
```

构建最后的相似度矩阵：

关键是将主题相似度与评分相似度相乘得到最后的用户相似度

```
def build_sim_matrix(user_mat, ratings_mat, n_users=size_of_neighborhood):
    sim_mat = np.zeros((n_users, n_users), dtype=float)
    for i in range(1, n_users, 1):
        print(i)
        for j in range(i, n_users, 1):
            # print(i,j)
            topic_sim = topic_similarity(allUsers[i], allUsers[j])
            rating_sim = rating_similarity(ratings_mat[i], ratings_mat[j])
            sim_mat[i, j] = topic_sim * rating_sim
    np.savetxt('similarity_mat1.csv', sim_mat, delimiter=',')
    return sim_mat
```

(4) 计算邻居并做推荐

(5)效果展示

当 userID=5 Num_of_recs=10 Size_of_neighborhood=10

```
PS D:\pythonProject\MovieLens-LDA-master> python recommender.py 5 10 10
1
2
3
4
5
6
7
8
9
DONE BUILDING SIM MATRIX
{480: 6, 1210: 6, 2028: 6, 377: 5, 2571: 5, 2692: 5, 2858: 5, 3418: 5, 593: 4, 608: 4, 1580: 4, 2355: 4, 3408: 4, 1: 4, 349: 4, 590: 4, 2006: 4, 16: 3, 265: 3, 318: 3, 1213: 3, 1466:
3, 1610: 3, 1704: 3, 2353: 3, 2762: 3, 3006: 3, 150: 3, 457: 3, 527: 3, 920: 3, 1265: 3, 1721: 3, 1961: 3, 2268: 3, 2278: 3, 3114: 3, 3147: 3, 110: 3, 589: 3, 733: 3, 2396: 3, 2916: 3
, 3105: 3, 648: 3, 1196: 3, 1270: 3, 3578: 3, 1198: 3, 2321: 3, 260: 3, 34: 2, 36: 2, 39: 2, 47: 2, 50: 2, 162: 2, 296: 2, 506: 2, 515: 2, 562: 2, 908: 2, 919: 2, 994: 2, 1089: 2, 150
1: 1, 1569: 1, 1674: 1, 1688: 1, 1806: 1, 1947: 1, 1959: 1, 2017: 1, 2082: 1, 2100: 1, 2406: 1, 2469: 1, 2506: 1, 2802: 1, 2966: 1, 3072: 1, 3074: 1, 3501: 1, 3508: 1, 3524: 1, 3536:
1, 3565: 1, 3600: 1, 3604: 1, 3610: 1, 3682: 1, 3685: 1, 3699: 1}
SORTED_L:  492
[480, 1210, 1, 349, 590, 2006, 457, 527, 920, 1265]
['Jurassic Park (1993)', 'Star Wars: Episode VI - Return of the Jedi (1983)', 'Toy Story (1995)', 'Clear and Present Danger (1994)', 'Dances with Wolves (1990)', 'Mask of Zorro, The (
1998)', 'Fugitive, The (1993)', "Schindler's List (1993)", 'Gone with the Wind (1939)', 'Groundhog Day (1993)']
```

3.3 评价指标

(1)准确率: 所有预测正确的占总的的比例。

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

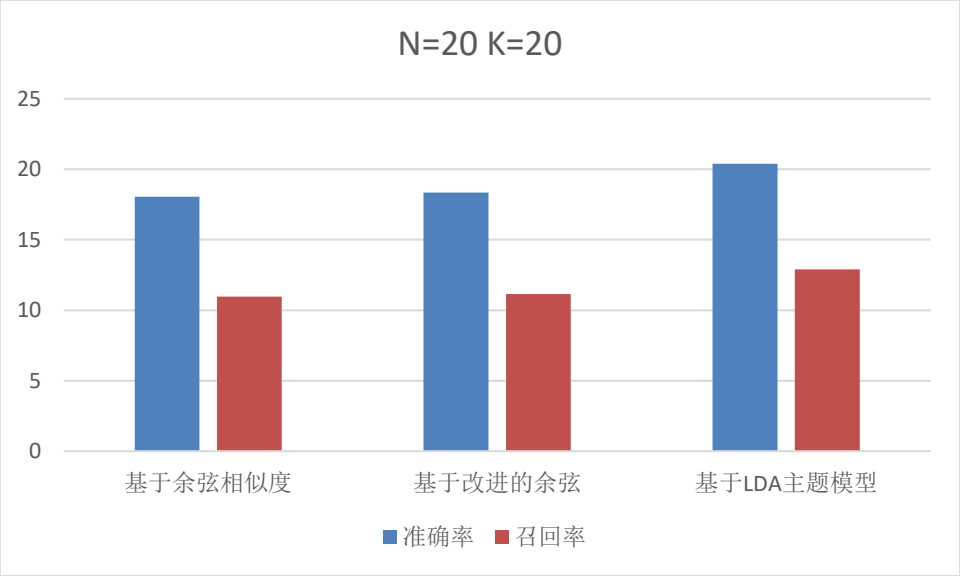
(2)召回率:正确预测为正类的占全部实际为正类的的比例。

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

3.4 结果对比

K：用户的相似的用户数　N：推荐图书数目

	基于余弦相似度	基于改进的余弦	基于LDA主题模型
准确率	18.05877483	18.35016556	20.39327823
召回率	10.97124292	11.14827146	12.90137124



结论：算法优化后准确率召回率都有明显提升。

二. 图书推荐系统平台

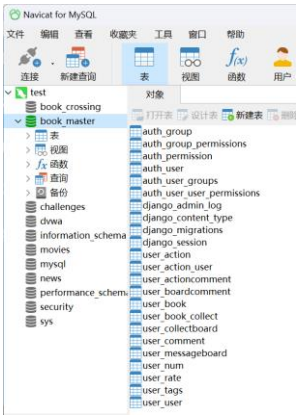
1.环境

Python 3.6

Django 2.2.7

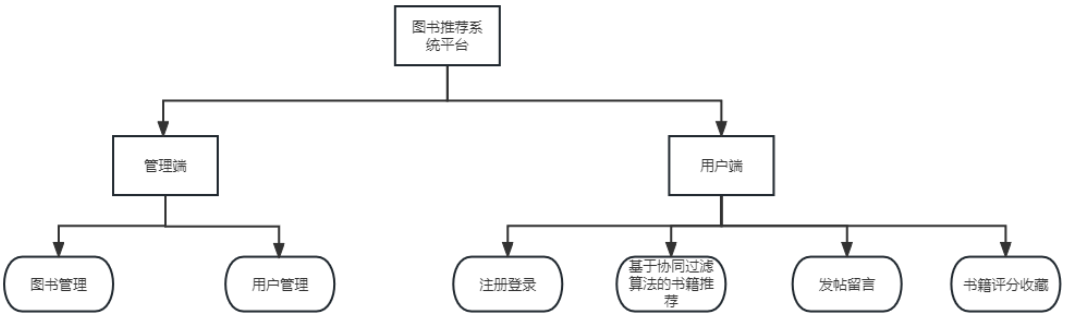
MySQL 5.7.26

用小皮面板配置 mysql 环境，用 navicat 管理数据库



数据来源：豆瓣（使用公开的已经包装好的数据集）

2.功能结构

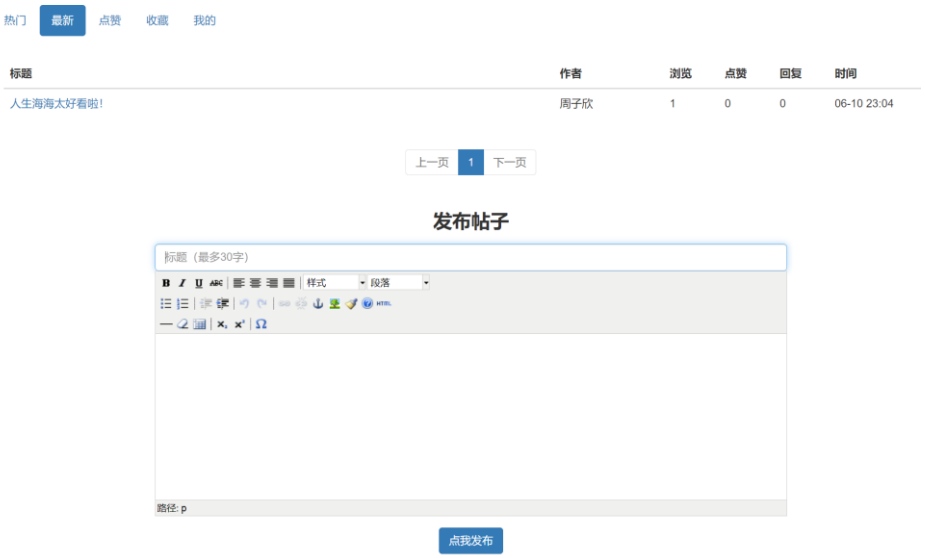


3.效果展示

(1)书籍的评分收藏评论

(2)书籍推荐

(3)论坛发帖



三. 参考

[1] Improving Collaborative Filtering based Recommenders using Topic Modelling, Jobin Wilson, Santanu Chaudhury, Brejesh Lal

https://arxiv.org/ftp/arxiv/papers/1402/1402.6238.pdf

[2] LDA 模型：https://zhuanlan.zhihu.com/p/76953963

[3] Mallet 包：https://www.cnblogs.com/fclbky/p/6123564.html

[4] Book-Crossing： http://www2.informatik.uni-freiburg.de/~cziegler/BX/

[5]系统: https://github.com/liangdongchang/book-master