



实验课程: 机器学习与数据挖掘

实验名称: 推荐系统

专业名称: 计算机科学与技术

学生姓名: 吴臻

学生学号: 21307371

- 背景和总体介绍
- 模型和算法
- 评价指标
- ▼ 代码实现 (以基于用户的协同滤波算法为例)
 - 改进 (基于内容)
- 实验结果及分析
- 实验总结
- 参考资料

背景和总体介绍

随着信息爆炸式增长，用户面临着过载的信息环境，很难从海量的信息中找到符合自己兴趣和需求的内容。而传统的搜索引擎只能根据用户提供的关键词进行粗略匹配，无法满足细分、个性化的需求。在这样的背景下，推荐系统应运而生。

- 推荐系统是一种利用机器学习和数据分析技术，为用户提供个性化推荐的系统。其主要目的是解决信息过载问题，提供更加个性化的服务。推荐系统通过分析用户的历史行为、兴趣偏好以及社交关系等数据，从大量的信息中筛选出用户可能感兴趣的内容或商品，并将其推荐给用户。
- 推荐系统的应用范围广泛，包括电子商务、社交媒体、音乐和视频流媒体平台等。通过推荐系统，电商平台可以向用户推荐符合其购买习惯和偏好的商品，提高用户购买转化率。社交媒体平台可以推荐用户感兴趣的内容，增加用户留存和活跃度。音乐和视频流媒体平台可以向用户推荐适合其口味的音乐和影片，提升用户体验。

模型和算法

推荐系统可以总结为以下模型：

$$\text{Utility Function: } u : X \times S \rightarrow R$$

其中， X 是用户的集合， S 是项的集合， R 是用户对项评分的集合，并且是关于项的有序集。

- 基于内容的推荐系统：向客户推荐与用户之前评价较高的产品相似的产品**
 - 算法流程：**
 - 给出物品表示：为每个物品抽取出一一些特征来表示此物品；
 - 学习用户偏好：利用一个用户过去喜欢（及不喜欢）的物品的特征数据，来学习出此用户偏好；
 - 生成推荐列表：根据候选物品表示和用户偏好，为该用户生成其最可能感兴趣的 n 个物品。
 - 项的画像：一个特征向量**

例如文本挖掘使用TF-IDF，即项的频率乘以逆文档频率

TF——词频，一个单词的重要性和它在文档中出现的次数呈正比。

$$TF = \frac{\text{单词次数}}{\text{文档中总单词数}}$$

IDF——逆向文档频率，一个单词在文档中的区分度。这个单词出现的文档数越少，区分度越大，IDF越大。

$$IDF = \log \frac{\text{文档总数}}{\text{单词出现的文档数} + 1}$$

- **用户画像**：例如评分项画像的加权平均

- 给定项画像 i 和用户画像 x ，可以使用余弦相似度评估： $u(x, i) = \cos(x, i) = \frac{x \cdot i}{\|x\| \cdot \|i\|}$

- **基于协同滤波的推荐系统**

- **基于用户的协同滤波算法**：找出评分与用户 x 评分相似的其他用户的集合，根据集合中用户的评分估计用户 x 对项的评分。

- a. **计算用户之间的相似度**：首先需要计算每对用户之间的相似度，可以使用余弦相似度、皮尔逊相关系数等方式进行计算(本实验采用皮尔逊相关系数)。假设有 m 个用户，使用矩阵 s 表示用户之间的相似度，其中 $s(i, j)$ 表示用户 i 和用户 j 之间的相似度。
 - b. **找到最相似的 k 个用户**：对于每个用户 i ，找到与其相似度最高的 k 个用户，记为 $N(i)$ ，其中 k 是一个预设参数。可以根据相似度大小从大到小进行排序，选取前 k 个相似用户。
 - c. **预测用户对物品的评分**：对于每个用户 i 没有评分过的物品 j ，通过以下公式来预测用户 i 对物品 j 的评分：

$$R(i, j) = (\sum S(i, k) * R(k, j)) / (\sum S(i, k))$$

其中， $S(i, k)$ 表示用户 i 和用户 k 之间的相似度， $R(k, j)$ 表示用户 k 对物品 j 的评分， $\sum S(i, k)$ 表示所有与用户 i 相似度不为 0 的用户 k 与用户 i 的相似度之和。

如果一个用户 i 已经对物品 j 评分过了，就不进行预测，直接使用原来的评分值。

- d. **推荐物品**：对于每个用户 i ，从其未评分过的物品中选取评分预测值最高的前 N 个物品作为推荐结果。
 - **基于项的协同滤波算法**：找出与项 y 相似的其他项的集合，根据集合中项的评分估计用户 x 对项 y 的评分。
 - a. **计算项之间的相似度**：首先需要计算每对项之间的相似度，可以使用余弦相似度、皮尔逊相关系数等方式进行计算(本实验采用皮尔逊相关系数)。假设有 n 个项，使用矩阵 s 表示项之间的相似度，其中 $s(i, j)$ 表示项 i 和项 j 之间的相似度。
 - b. **找到与项 y 最相似的 k 个项**：对于每个项 y ，找到与其相似度最高的 k 个项，记为 $N(y)$ ，其中 k 是一个预设参数。可以根据相似度大小从大到小进行排序，选取前 k 个相似项。
 - c. **预测用户对项的评分**：对于用户 x 没有对项 y 评分过的情况，通过以下公式来预测用户 x 对项 y 的评分：

$$R(x, y) = (\sum S(y, i) * R(x, i)) / (\sum S(y, i))$$

其中, $S(y, i)$ 表示项y和项i之间的相似度, $R(x, i)$ 表示用户x对项i的评分, $\sum S(y, i)$ 表示所有与项y相似度不为0的项i与项y的相似度之和。

如果用户x已经对项y评分过了, 就不进行预测, 直接使用原来的评分值。

d. **推荐给用户的项**: 对于用户x, 从其未评分过的项中选取评分预测值最高的前N个项作为推荐结果。

皮尔逊相关系数 (Pearson correlation coefficient) 是一种衡量两个变量之间线性关系强度和方向的统计量。它的取值范围在-1到1之间, 其中1表示完全正相关, -1表示完全负相关, 0表示没有线性相关性。

皮尔逊相关系数可以通过以下公式进行计算:

$$\rho(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2 \sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

其中, $\rho(X, Y)$ 表示变量x和变量y之间的皮尔逊相关系数, X_i 和 Y_i 分别表示变量x和变量y的第i个观测值, \bar{X} 和 \bar{Y} 分别表示变量x和变量y的均值, n表示样本数量。

评价指标

下面是推荐系统的一些常见评价指标, 本次实验主要采用均方根误差 (RMSE) 和覆盖率指标 (Coverage Metrics)

1. **准确度指标 (Accuracy Metrics)**: 衡量推荐系统在预测用户兴趣或推荐项时的准确性。

- 均方根误差 (RMSE): 计算预测评分与实际评分之间的差异, 值越小表示准确度越高。均方根偏差代表预测值和观察值之差的二阶样本矩的平方根

$$RMSE = \sqrt{\frac{\sum_{t=1}^T (\hat{y}_t - y_t)^2}{T}}$$

- 平均绝对误差 (MAE): 计算预测评分与实际评分之间的绝对差异的平均值, 同样值越小表示准确度越高。

2. **覆盖率指标 (Coverage Metrics)**: 衡量推荐系统能够推荐多样化的项的能力。

项覆盖率: 衡量推荐系统能够覆盖的不同项的比例, 值越高表示推荐系统能够提供更广泛的选择。

3. **多样性指标 (Diversity Metrics)**: 衡量推荐系统推荐的项之间的差异性和多样性。

- 用户覆盖率: 衡量推荐系统能够为不同用户提供推荐的比例, 值越高表示推荐系统能够满足更多用户的需求。

- **项重叠度**：衡量推荐给不同用户的项之间的相似度，值越低表示推荐系统能够提供更多差异化的推荐。
4. **新颖性指标 (Novelty Metrics)**：衡量推荐系统能否推荐用户未曾接触过的新项。
- 信息熵：计算推荐项的信息熵，值越高表示推荐系统能够提供更多新颖的推荐。
5. **实时性指标 (Timeliness Metrics)**：衡量推荐系统的响应速度和及时性。
- 推荐时效性：衡量推荐系统能够在用户需求变化时及时更新和调整推荐结果的能力。

代码实现（以基于用户的协同滤波算法为例）

数据预处理（数据采用ml-latest-small）

先读取整个文件，构建评分矩阵，再挖去其中部分数据，使用剩余数据预测挖去内容，最后和原本的真实数据进行比较

```
# 读取评分数据
data = pd.read_csv(r'E:\学习资料\大三上\机器学习与数据挖掘\大作业\src\ml-latest-small\ratings.csv')

# 读取电影数据
movies = pd.read_csv(r'E:\学习资料\大三上\机器学习与数据挖掘\大作业\src\ml-latest-small\movies.csv')

# 创建用户-物品评分矩阵
ratings_matrix = data.pivot_table(index='userId', columns='movieId', values='rating')

# 计算要提取的行和列范围
# test_rate = 0.05
# num_rows = len(ratings_matrix)
# num_cols = len(ratings_matrix.columns)
# rows_to_extract = int(num_rows * test_rate)
# cols_to_extract = int(num_cols * test_rate)

rows_to_extract = 100
cols_to_extract = 300

# 提取部分行和列来测试
real_matrix = ratings_matrix.iloc[:rows_to_extract, :cols_to_extract]
real_matrix = real_matrix.copy()
ratings_matrix.iloc[:rows_to_extract, :cols_to_extract] = np.nan
```

```
# 计算用户之间的相似度（使用皮尔逊相关系数）
user_similarity = ratings_matrix.T.corr(method='pearson')
```

预测评分(得到用户i对项j的评分)

- 1.如果是user-based CF，一些用户评价体系的方法不同，评价所有电影时可能要么给最高分，要么给最低分。这些用户给出评价的相对不同比绝对值更重要，所以先算出目标物品的平均评分，再计算相对不同
- 2.如果是item-based CF，则不要求均值，因为同一个用户的评价标准一般保持不变

```
# 预测挖去内容的评分(一个用户对应一个物品的评分)
def predict_rating(user_similarity, ratings_matrix, user_id, item_id, k):
    sim_scores = user_similarity[user_id].dropna() # 获取与目标用户的相似度
    user_ratings = ratings_matrix[item_id].dropna() # 获取目标物品的评分
    mean_user_rating = user_ratings.mean() # 目标物品的平均评分

    # 获取相似度中在item_ratings中存在的索引
    valid_indices = sim_scores.index.isin(user_ratings.index)

    # 筛选出在item_ratings中存在的相似度
    valid_sim_scores = sim_scores[valid_indices]

    # 选择前k个最大的值
    top_k = valid_sim_scores.nlargest(k)

    sum_sim_scores = 0
    weighted_sum = 0

    for user in top_k.index: # user--用户id
        if user != user_id and user in user_ratings.index:
            similarity = top_k[user]
            rating = ratings_matrix.loc[user, item_id]
            # 只参考相似度大于0的用户
            if similarity <= 0:
                break
            if pd.notnull(rating):
                weighted_sum += similarity * (rating - mean_user_rating)
```

```

        sum_sim_scores += similarity

    if sum_sim_scores != 0:
        predicted_rating = mean_user_rating + weighted_sum / sum_sim_scores
    else:
        predicted_rating = np.nan

    return predicted_rating

```

预测模型（预测空缺矩阵的值）

```

def evaluate_model(user_similarity, ratings_matrix, real_matrix, k):
    predicted_matrix = [] # 存储预测评分矩阵的列表

    test_row = len(real_matrix)
    test_col = len(real_matrix.columns)
    for i in range(test_row):
        predicted_rating = []
        for j in range(test_col):
            user_id = real_matrix.index[i] # 获取用户ID
            item_id = real_matrix.columns[j] # 获取物品ID
            rating = predict_rating(user_similarity, ratings_matrix, user_id, item_id)
            predicted_rating.append(rating)
        predicted_matrix.append(predicted_rating) # 将每个用户的预测评分列表添加到预测评分矩阵

    # 计算真实评分矩阵和预测评分矩阵之间的RMSE
    rmse = calculate_rmse(real_matrix, predicted_matrix)

    # 计算覆盖率
    coverage = calculate_coverage(predicted_matrix)

    return rmse, coverage

```

评价指标（RMSE和Coverage）

```

# 计算RMSE(忽略NaN)
def calculate_rmse(real_matrix, predicted_matrix):
    # 将DataFrame对象转换为NumPy数组
    real_matrix = real_matrix.to_numpy()
    predicted_matrix = np.array(predicted_matrix)

    # 计算非NaN元素的均方误差
    mask = ~np.isnan(real_matrix) & ~np.isnan(predicted_matrix)
    mse = np.mean((real_matrix[mask] - predicted_matrix[mask]) ** 2)
    rmse = np.sqrt(mse)
    return rmse

# 计算coverage
def calculate_coverage(predicted_matrix):
    total_elements = np.size(predicted_matrix) # 计算总元素数量
    non_nan_elements = np.count_nonzero(~np.isnan(predicted_matrix)) # 计算非 NaN 值
    coverage = non_nan_elements / total_elements # 计算非 NaN 值的占比
    return coverage

```

改进（基于内容）

将基于内容的推荐系统和基于项的协同滤波算法相结合，即计算项的相似度时，不再使用效用矩阵来计算，而是使用项的内容信息，使用TF-IDF来提取每个项的特征向量，使用得到的特征向量计算物品的相似度，后面操作和基于项的协同滤波算法基本一致

```

# 计算电影之间的相似度（使用电影的标题和类型）
tfidf = TfidfVectorizer(stop_words='english')
movies['genres'] = movies['genres'].fillna('')
tfidf_matrix = tfidf.fit_transform(movies['genres'])
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

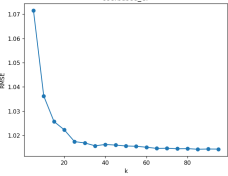
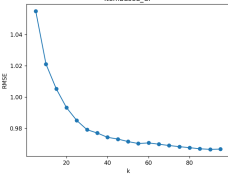
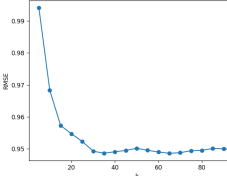
# 计算电影之间的相似度
similarity = pd.DataFrame(cosine_sim, index=movies['movieId'], columns=movies['movieId'])

```


实验结果及分析

数据集	用户数量	电影数目	评分项
ml-latest-small	600	9000	100000

测试时，选择前100个用户和前300个电影的区域作为验证区域。

评测指标	基于用户的协同滤波算法	基于项的协同滤波算法	混合（基于内容）
RMSE			
Coverage	Coverage: 86.21666666666667%	Coverage: 75.64666666666666%	Coverage: 97.74666666666667%
Time	For k=90, RMSE=1.0143574159650994, For k=95, RMSE=1.0143291522554951, It costs 432.1398651599884 s	For k=90, RMSE=0.9665970433891081, For k=95, RMSE=0.9667433490743038, It costs 588.6717200279236 s	For k=90, RMSE=0.9500124115744268, For k=95, RMSE=0.9499030275295733, It costs 814.42116522789 s

分析：一共运行了三个模型，k表示选择前k个相似度最大的作为评分参考对象

- 1.从k来看，我们可以看到随着k的增大，误差一开始迅速减小，当k增大到一定值时，误差慢慢收敛，这符合常理，随着参考对象的增多，预测得到的评分不会受到某几个特殊打分项的影响
- 2.从RMSE来看，混合（基于内容）最好，因为数据集中的项是电影，电影一般有明确的主题，使用TF-IDF生成的特征向量更具有说服力，使用特征向量得到的相似度矩阵也更贴合实际，基于项的协同滤波算法的效果好于基于用户的，基于项的协同滤波算法更适用于兴趣变化较为稳定的应用（比如此类数据集）
- 3.从Coverage（覆盖率）来看，混合（基于内容）最好，因为用户的兴趣一般不会改变，所以会观看主题类似的电影，当使用TF-IDF时，它是根据主题来确定相似度的，比用其他用户对电影的评价来确定相似度更合理，所以对于相似度大的电影，用户一般都有对其打分，这提高了覆盖率，不过这样会导致在进行推荐时，缺乏多样性（每次都推荐主题类似的电影）
- 4.从Time（耗费时间）来看，因为该数据集用户数少，电影项多，所以基于项的协同滤波算法耗时最短，它的相似度矩阵规模比较小，运行速度更快

• 基于内容的推荐系统的评价

优点：

- i. 不受数据稀疏性影响：相比协同滤波算法，基于内容的推荐系统不依赖于用户行为数据，因此可以避免数据稀疏性问题，即使对于新用户或冷启动物品也能进行推荐。

- ii. **解释性强**：基于内容的推荐系统可以根据物品的特征和用户的兴趣偏好进行解释，用户可以理解为何会得到这样的推荐结果，从而提高用户对推荐的信任度。

缺点：

- i. **特征获取难度较大**：基于内容的推荐系统需要对物品进行特征提取，但有些物品可能没有明确的特征描述，或者特征提取过程复杂，因此获取物品的特征可能会面临一定的困难。
- ii. **无法发现新的兴趣领域**：基于内容的推荐系统通常根据用户已有的兴趣进行推荐，但这可能限制了用户发现新的兴趣领域和新的物品。

适用场景：

- i. **物品具有明确的特征描述**：基于内容的推荐系统适用于那些具有明确特征描述的物品，例如电影的类型、书籍的作者、产品的属性等。
- ii. **需要个性化解释的推荐结果**：基于内容的推荐系统能够提供对推荐结果的解释，因此适用于一些需要给用户推荐解释的场景，如文本推荐、音乐推荐等。
- iii. **面对冷启动问题的场景**：当新用户加入系统或新物品推出时，基于内容的推荐系统可以避免数据稀疏性问题，提供针对用户和物品的个性化推荐。

• 协同滤波推荐系统的评价

- **基于用户的协同滤波算法**（适用场景）：具备更强的社交特性，适用于用户少物品多，时效性较强的场景。
- **基于项的协同滤波算法**（适用场景）：更适用于兴趣变化较为稳定的应用，更接近于个性化的推荐，适合物品少用户多，用户兴趣固定持久，物品更新速度不是太快的场合，比如电影推荐。

优点：

- i. **不依赖物品内容特征**：与基于内容的方法相比，协同滤波算法不需要对物品的内容进行分析，只需要利用用户行为数据进行推荐，因此对于没有明确特征描述的物品也能有效推荐。
- ii. **可扩展性**：协同滤波算法可以适应不同规模的数据集，从小型到大型，因此适用于各种规模的推荐系统。
- iii. **个性化推荐**：协同滤波算法可以根据用户的历史行为和其他用户之间的相似性，提供个性化的推荐结果，帮助用户发现他们可能感兴趣的物品。

缺点：

- i. **数据稀疏性**：在大多数推荐系统中，用户评分记录通常是稀疏的，即用户评价的物品相对于总物品数量较少。这会导致协同滤波算法难以找到足够相似的用户或物品来进行准确的推荐。
- ii. **冷启动问题**：当新用户或新物品加入系统时，由于缺乏足够的评分数据，协同滤波算法无法准确预测推荐结果。
- iii. **算法偏好**：协同滤波算法倾向于推荐热门物品，因为它们具有更多的评分数据和更高的相似度。这可能会导致一些长尾物品得到较少的推荐，限制了推荐系统的多样性。

- iv. **算法开销**：协同滤波算法需要计算用户或物品之间的相似性，当数据集非常大时，计算复杂度会很高，可能导致实时性的问题。

实验总结

总结：本次实验主要是实现了基于协同滤波的推荐系统，并将基于内容的推荐系统和基于项的协同滤波算法相结合，得到的混合模型虽然运行时间有所增加，不过其他指标都有所提升。自己实现的推荐系统整体还是十分粗糙的，有许多可以改进的地方，并且评测指标的选择也十分重要。通过本次实验，我对推荐系统的类型、实现方式以及评价指标有了更深的理解，将课上所学的理论知识和实际相结合，也算是“学有所成”了。

参考资料

1. 数据集下载：<http://files.grouplens.org/datasets/movielens/ml-latest-small.zip>
2. 基于内容：<https://zhuanlan.zhihu.com/p/80068528>
3. 协同滤波：<https://zhuanlan.zhihu.com/p/68373487>
4. 评价指标：http://sofasofa.io/forum_main_post.php?postid=1000292