作业-7:混合推荐系统设计

22331095 王志杰

基于MovieLens数据集
实现协同过滤(用户/物品相似度)与基于内容的推荐(电影标签)，对比两者召回率。
(可选)设计加权混合策略或其他策略提升推荐多样性。

实验目的

1. 基于MovieLens-1M数据集实现三种推荐算法：
    - 基于用户的协同过滤(User CF)
    - 基于物品的协同过滤(Item CF)
    - 基于电影类型的基于内容推荐
2. 对比分析不同算法的召回率性能
3. 探索混合推荐策略对推荐效果的提升

数据集与预处理

数据源

MovieLens 1M数据集包含：

- 6040个用户
- 3952部电影
- 1,000,209条评分记录（1-5分）

预处理流程

```
# 类型特征处理
movies["genres"] = movies["genres"].str.replace("|", " ")

# 数据分割（分层抽样）
train_data, test_data = train_test_split(ratings, test_size=0.2,
                                         stratify=ratings["userId"],
                                         random_state=42)

# 用户-物品矩阵构建
user_movie_train = train_data.pivot_table(index="userId",
                                          columns="movieId",
                                          values="rating",
                                          fill_value=0)
```

方法实现

1. 基于用户的协同过滤 (User CF)

算法流程：

1. 计算用户相似度矩阵：
   $sim(u,v) = \frac{\sum_{i \in I} r_{ui} \cdot r_{vi}}{\sqrt{\sum r_{ui}^2} \sqrt{\sum r_{vi}^2}}$
2. 选取Top-10相似用户
3. 加权聚合相似用户的评分：
   $\hat{r}_{ui} = \sum_{v \in N(u)} sim(u,v) \cdot r_{vi}$

核心参数：

- 相似用户数：10
- 评分阈值：4.0+

```
def user_cf_recommend(user_id, n=10):
    if user_id not in user_movie_train.index:
        return []
    sim_users = user_sim_df[user_id].nlargest(11).index[1:]
    user_rated = user_movie_train.loc[user_id][user_movie_train.loc[user_id] >
0].index

    recommendations = {}
    for sim_user in sim_users:
        sim_ratings = user_movie_train.loc[sim_user]
        for movie_id in sim_ratings[sim_ratings > 0].index:
            if movie_id not in user_rated:
                recommendations[movie_id] = (
                    recommendations.get(movie_id, 0) + sim_ratings[movie_id] *
user_sim_df.loc[user_id, sim_user]
                )
    return sorted(recommendations, key=recommendations.get, reverse=True)[:n]
```

2. 基于物品的协同过滤 (Item CF)

物品的相似度计算：
$sim(i,j) = \frac{\sum_{u \in U} r_{ui} r_{uj}}{\sqrt{\sum r_{ui}^2} \sqrt{\sum r_{uj}^2}}$

实现优化：

- 相似物品数：20
- 评分归一化处理

```
def item_cf_recommend(user_id, n=10):
    if user_id not in user_movie_train.index:
        return []
    user_ratings = user_movie_train.loc[user_id]
    rated_movies = user_ratings[user_ratings > 0].index

    scores = {}
    for movie_id in user_movie_train.columns:
        if user_ratings[movie_id] == 0:
            similar = item_sim_df[movie_id].nlargest(21).index[1:]
```

```
            total_sim = sum_score = 0
            for sm in similar:
                if sm in rated_movies:
                    sim = item_sim_df.loc[movie_id, sm]
                    total_sim += sim
                    sum_score += sim * user_ratings[sm]
            if total_sim > 0:
                scores[movie_id] = sum_score / total_sim
    return sorted(scores, key=scores.get, reverse=True)[:n]
```

3. 基于内容的推荐

特征工程:

```
tfidf = TfidfVectorizer(stop_words="english")
genres_tfidf = tfidf.fit_transform(movies["genres"])
content_sim = cosine_similarity(genres_tfidf)
```

推荐策略:

- 对用户评分≥4的电影进行类型特征扩展
- 使用TF-IDF加权的余弦相似度:
  $sim_{content}(i,j) = \frac{\mathbf{tfidf}_i \cdot \mathbf{tfidf}_j}{||\mathbf{tfidf}_i|| \cdot ||\mathbf{tfidf}_j||}$

4. 混合推荐策略

权重分配:

```
cf_weight = 0.7  # 协同过滤分量
content_weight = 0.3  # 内容特征分量

# 混合公式
combined_score = {**{m: cf_weight for m in cf_rec},
                  **{m: content_weight for m in content_rec}}
```

实验结果与分析

```
Recall Results:
User Collaborative Filtering: 0.1133
Item Collaborative Filtering: 0.0071
Content-Based Recommendation: 0.0099
Hybrid Recommendation: 0.1116
```

一、算法性能横向对比

根据实验结果，各推荐算法的召回率表现呈现显著差异：
• **用户协同过滤（User CF）**召回率最高（0.1133），验证了其通过用户相似性捕捉群体偏好的有效性
• **混合推荐**次之（0.1116），说明简单加权策略未能有效发挥算法协同效应
• **基于内容推荐**（0.0099）与**物品协同过滤**（0.0071）表现最弱，反映特征工程与数据稀疏性问题

二、异常现象深度剖析

2.1 协同过滤两极分化

User CF与Item CF的**性能差距达16倍**，这可能源于：

1. **数据分布特性**：MovieLens用户行为矩阵存在明显长尾分布，用户相似度计算更稳定
2. **冷启动影响**：新物品缺乏足够评分数据，导致Item CF相似度计算失真
3. **特征维度差异**：用户维度（6040）远高于物品维度（3952），矩阵稠密度不同

2.2 混合策略失效原因

混合推荐未达预期提升效果，可能涉及：

```
# 混合权重配置示例（需优化）
cf_weight = 0.7  # 当前设置
content_weight = 0.3
```

1. **线性加权局限性**：简单线性组合无法捕捉算法间非线性关系
2. **负向协同效应**：低效的内容推荐拖累协同过滤效果
3. **阈值设置不当**：未根据算法置信度动态调整融合策略

三、改进方向

3.1 算法层面优化

优化方向	技术方案	预期收益
特征增强	融合知识图谱高阶邻居特征	提升召回
去偏处理	引入因果图模型消除流行度偏差	改善长尾覆盖
深度融合	使用DNN进行非线性特征	改善线性加权

3.2 工程实践策略

1. **动态权重调整**：建立A/B测试框架监控不同用户群体的算法表现
2. **增量学习机制**：通过实时反馈更新用户相似度矩阵
3. **多模态特征融合**：结合海报、影评等非结构化数据增强内容表示