華中科技大学

课 程 实 验 报 告

课程名称： 大数据分析

专业班级： **CS2202**

学　　号： **U202215378**

姓　　名： 冯瑞琦

指导教师： 崔金华

报告日期： **2024 年 6 月 11 日**

计算机科学与技术学院

目录

实验三 关系挖掘实验

3.1 实验内容

必做：

1. 实验内容

实验数据：从实验一中得到的引用关系数据为<<title,<title1,…,titlek>>,…>，将其处理为<<title,title1,…,titlek>,…>作为算法输入。

编程实现 Apriori 算法，要求使用实验一得到的前 1000 个 title 及其引用关系作为实验数据。

2. 实验要求

输出 1~4 阶频繁项集与关联规则，各个频繁项的支持度，各个规则的置信度，各阶频繁项集的数量以及关联规则的总数。

固定参数以方便检查，频繁项集的最小支持度为 0.15，关联规则的最小置信度为 0.3。此处支持度的定义为某个项集出现的频率，也就是包含该项集的数目与总数目的比例（总的购物篮数目为 1000）。

加分项：

1. 实验内容

在 Apriori 算法的基础上，使用 pcy 算法对二阶频繁项集的计算阶段进行优化。

2. 实验要求

输出 1~4 阶频繁项集与关联规则，各个频繁项的支持度，各个规则的置信度，各阶频繁项集的数量以及关联规则的总数。

输出 pcy 算法中的 vector 的值，以 bit 位的形式输出。

固定参数以方便检查，频繁项集的最小支持度为 0.15，关联规则的最小置信度为 0.3。

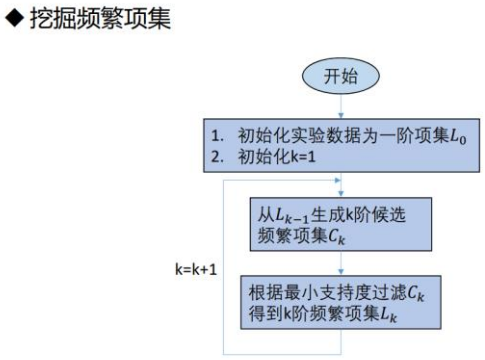
3.2 实验过程

3.2.1 编程思路

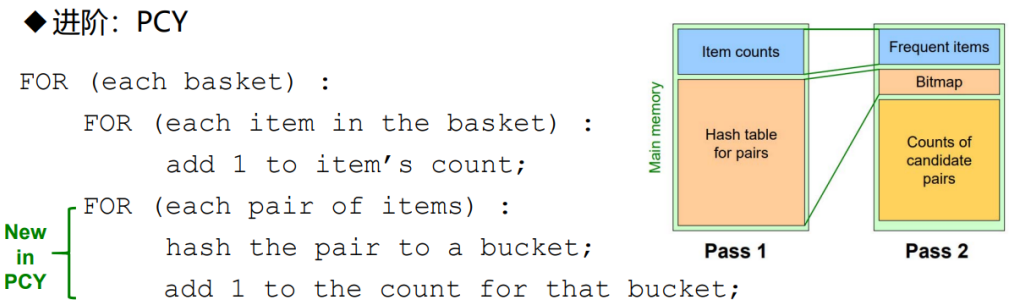
Apriori 算法流程为首先挖掘频繁项集，然后由频繁项集产生关联规则如下：

- 对于一个频繁项 I，I 的每个子集 A，生成一个规则 A → I \ A

- 筛选出所有置信度大于最小置信度的规则
- 一个规则的置信度计算公式：confidence(A→ I \ A) = support(I) / support(A)



进阶算法则是在 Apriori 算法的基础上，使用 pcy 算法对二阶频繁项集的计算阶段进行优化。bit vector 的每一位代表一个 bucket 是否为频繁的，如果一个 bucket 中的计数小于最小支持度，那么映射到这个桶的二阶项必然是非频繁的。PCY 算法的图示如下。



具体编程思路如下：

1) 数据加载与初始化

导入必要的模块，并从之前的任务输出文件 my_jump.pkl 中加载跳转关系数据。通过 pickle.load()方法，将跳转关系字典 jump_relations 加载到内存中。初始化 Apriori 算法和关联规则挖掘所需的参数：最小支持度阈值 min_support、最小置信度阈值 min_confidence，以及总标题数 total_titles。最后将跳转关系转换为列表形式，以便后续处理。

2) 支持度计算和候选项集生成

定义两个关键函数。calculate_support 函数用于计算给定项集的支持度，即项集在所有事务中出现的频率。具体操作是遍历所有事务，统计包含该项集的事务数量，然后除以事务总数以得到支持度。generate_candidates 函数用于生成大小为 k 的候选项集。通过组合先前频繁项集的元素，生成新的候选项集。具体操作是将之前的频繁项集两两组合，取并集并检查其大小是否为 k，若是，则加入候选集。

3) PCY 算法第一遍扫描

PCY 算法的第一遍扫描用于统计每个单项集和二项集的频率。遍历每个事务（即标题及其跳转关系），对每个事务中的单个项集进行计数，更新 item_counts；然后，生成事务中所有可能的二项集，计算其哈希值，将其映射到对应的哈希桶，并更新 bucket_counts。通过这种方式，可以有效减少需要存储和处理的候选二项集数量。

4) Apriori 算法与 PCY 优化

apriori_pcy 函数使用 PCY 算法优化二阶项集生成。首先，调用 pcy_pass1 函数进行第一遍扫描，获取单项集和哈希桶的频率。然后，根据最小支持度阈值筛选出频繁一项集，并创建位向量用于优化二阶项集的生成。位向量表示哪些哈希桶可能包含频繁二项集。若位向量为 1，表示该哈希桶是频繁的，否则不是。通过以 bit 位的形式输出位向量可以直观地看出哪些桶是频繁的。接下来，生成二阶候选项集，并根据支持度和位向量进行过滤，过滤掉不满足支持度且不在位向量中的候选项集。对于更高阶的项集，直接生成候选集并根据支持度进行过滤，直到生成 4 阶频繁项集，即 k=4。

```
# 生成频繁项集的函数（PCY算法优化二阶项集）
def apriori_pcy(transactions, min_support, bucket_size):
    freq_itemsets = []
    item_counts, bucket_counts = pcy_pass1(transactions, bucket_size)  # PCY算法第一遍扫描
    one_itemsets = {item for item in item_counts if calculate_support(item, transactions) >= min_support}
    freq_itemsets.append(one_itemsets)

    # 创建位向量
    bit_vector = [1 if bucket_counts[i] >= min_support * len(transactions) else 0 for i in range(bucket_size)]

    print(f"PCY bit vector: {''.join(map(str, bit_vector))}")  # 输出PCY算法中的位向量

    k = 2
    while k<=4:
        if k == 2:
            # 使用PCY算法优化二阶项集
            candidates = generate_candidates(freq_itemsets[-1], k)
            # 过滤掉不满足支持度且不在位向量中的候选项集
            freq_k_itemsets = {item for item in candidates if calculate_support(item, transactions) >=
                               min_support and bit_vector[hash(item) % bucket_size] == 1}
        else:
            candidates = generate_candidates(freq_itemsets[-1], k)
            # 过滤掉不满足支持度的候选项集
            freq_k_itemsets = {item for item in candidates if calculate_support(item, transactions) >= min_support}

        if not freq_k_itemsets:
            break
        freq_itemsets.append(freq_k_itemsets)
        k += 1

    return freq_itemsets
```

5) 关联规则

generate_rules 函数用于从频繁项集中生成关联规则。具体操作是：遍历所有频繁项集，从每个频繁项集中生成所有可能的子集组合（前件）。然后，计算每个前件的置信度，即该前件出现时后件也出现的概率。如果置信度大于或等于最小置信度阈值，则将该规则（前件 -> 后件）加入结果列表。最终返回所有满足条件的关联规则。最后，定义哈希桶的大小 bucket_size 并调用

apriori_pcy 函数生成频繁项集。调用 generate_rules 函数生成关联规则，计算每个频繁项集的支持度，并将结果写入文件 frequent_itemsets.txt。关联规则的结果写入文件 association_rules.txt。最后，打印出每个频繁项集和关联规则的数量。

3.2.2 遇到的问题及解决方式

1) jump_relations 的处理

问题描述：在处理跳转关系数据时，我思考了如何有效存储和管理这些关系的问题。跳转关系通常包含一个标题（title）和其关联的多个跳转目标（targets）。如果不进行合理的存储和管理，这些关系可能会显得杂乱无章，难以有效地查询和使用。

解决方式：将跳转关系转换成列表。将跳转关系数据存储在一个结构化的列表中可以显著提高数据的可操作性和可读性。具体来说，这种存储方式有结构化存储、便于查询、易于扩展、兼容性强等优点。

```
# 将jump_relations转换为列表
transactions = list(jump_relations.values())
```

2) PCY 算法过滤

问题描述：在编写频繁项集生成的代码时，最初只使用了一个过滤条件，即根据最小支持度阈值来筛选候选项集。这种方法只考虑了候选项集的支持度。然而，在运行过程中发现，这种单一的过滤条件并不能有效地减少候选项集的数量，导致算法的性能较差。特别是在处理二阶项集时，生成的候选项集数量巨大，计算效率非常低。

解决方式：为了优化频繁项集的生成过程，引入 PCY 算法的位向量过滤机制。在 PCY 算法第一遍扫描中，通过哈希桶记录二项集的出现频率，并生成位向量表示哪些哈希桶可能包含频繁二项集。在生成二阶候选项集时，除了根据支持度进行过滤，还结合位向量信息，进一步筛选候选项集。最终的筛选条件结合了支持度和位向量两个因素。只有同时满足这两个条件的候选项集才会被保留，从而显著减少了候选项集的数量，提高了算法的计算效率。

3) 关联规则生成和置信度计算

问题描述：在生成关联规则时，计算每个候选规则的置信度可能会非常耗时，尤其是当频繁项集较多时。这会导致程序运行效率低下。

解决方式：通过剪枝和提前终止条件来优化关联规则的生成过程。对于每个候选规则，在计算置信度之前，先判断其前件和后件是否为频繁项集，如果不是频繁项集，则直接跳过该规则。同时，可以在计算过程中缓存一些中间结果，减少重复计算。

3.2.3 实验测试与结果分析

1) frequent_itemsets.txt 文件

frequent_itemsets.txt 文件存放的是通过 Apriori 算法挖掘出的频繁项集以及它们对应的支持度。每行记录一个频繁项集及其支持度，按照项集的大小（1-itemset、2-itemset 等）进行分类和标注。支持度表示该项集在所有事务中出现的频率，是评估项集频繁程度的重要指标。生成的文件如下图所示。

frequent_itemsets.txt

```
1-itemset: {'system'}, support: 0.1518
1-itemset: {'references'}, support: 0.2618
1-itemset: {'web'}, support: 0.2325
1-itemset: {'archive'}, support: 0.1634
1-itemset: {'https'}, support: 0.2304
1-itemset: {'like'}, support: 0.2272
1-itemset: {'time'}, support: 0.2010
1-itemset: {'people'}, support: 0.3403
1-itemset: {'mean'}, support: 0.2031
1-itemset: {'world'}, support: 0.1738
```

以第一行为例，表示一个单项集，即包含一个元素的频繁项集。在这个例子中，频繁项集为{'system'}，它的支持度为 0.1518。这意味着在所有事务（标题及其跳转关系）中，包含 system 这个词的事务占总事务数的 15.18%。这个频繁项集满足设定的最小支持度阈值，故被列为 1-itemset 中的一部分。

2) association_rules.txt 文件

association_rules.txt 文件存放的是通过关联规则挖掘生成的关联规则及其对应的置信度。每行记录一条关联规则，形式为“前件 -> 后件”，并附有该规则的置信度，表示在包含前件的事务中，后件也出现的概率。生成的文件如下图所示。

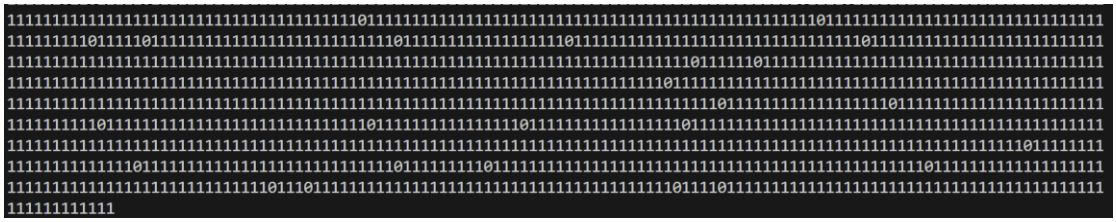
association_rules.txt

```
Rule: {'people'} -> {'date'}, confidence: 0.5077
Rule: {'date'} -> {'people'}, confidence: 0.7051
Rule: {'url'} -> {'http'}, confidence: 0.7511
Rule: {'http'} -> {'url'}, confidence: 0.8756
Rule: {'pages'} -> {'title'}, confidence: 0.6023
Rule: {'title'} -> {'pages'}, confidence: 0.6555
```

以第一行为例，这行表示一条关联规则：当事务中包含 people 时，也包含 date 的置信度为 0.5077，意味着在所有包含 people 的事务中，有 50.77% 的事务同时也包含 date。

3) 位向量输出

bit vector 的每一位代表一个 bucket 是否为频繁的，如果一个 bucket 中的计数小于最小支持度，那么映射到这个桶的二阶项必然是非频繁的。输出 pcy 算法中的 vector 的值，以 bit 位的形式输出。1 则表示该桶频繁，否则相反，如下图所示。



4) itemsets count 和 Total association rules 输出

输出 1-4 阶频繁项集的数量和关联规则总数如下图。和助教给出的参考答案一致，说明结果正确。

```
1-itemsets count: 36
2-itemsets count: 105
3-itemsets count: 143
4-itemsets count: 122
Total association rules: 2776
```

3.3 实验总结

本次实验中，通过实现和运行 Apriori、 PCY 算法，我深入理解了频繁项集挖掘和关联规则生成的基本原理与应用。

实验过程中，我遇到了如何优化候选项集生成的挑战。最初，仅根据支持度进行过滤，导致候选项集数量巨大，计算效率低下。通过结合 PCY 算法的位向量过滤机制，我们不仅减少了内存占用，还提高了计算速度。这一优化策略在实践中展现了其重要性和有效性。

此外，通过生成关联规则并计算其置信度，我们能够从频繁项集中提取出有价值的模式和关系。这些关联规则对于数据分析和决策支持具有重要意义。实验结果表明，结合支持度和置信度的双重标准，可以更准确地筛选出高质量的关联规则。

总之，本次实验不仅增强了我对 Apriori 和 PCY 算法的理解，也提高了我在大数据处理和优化方面的能力。通过实际编码和调试，我认识到算法优化和数据结构选择在大规模数据处理中的关键作用，受益匪浅。