数据仓库大作业--频繁模式挖掘

孙培艺 1500012899 https://github.com/spyyes/PhraseAnalysis

1. 实验综述

关联分析常常用于从大规模数据库中寻找元素的隐含关系，是数据仓库中数据挖掘的最常用的方法。本实验旨在实现基本的数据挖掘算法（Apriori算法），选取部分数据集数据进行挖掘。在探寻数据隐含关系的同时，试图评估数据挖掘算法的性能和特性。

本报告主要包括以下部分：

1. 实验原理（包括算法详细描述、算法特点等）
2. 实验环境搭建（数据集的选取、挖掘的问题、编程环境简述）
3. 实验发现
4. 算法性能分析

本报告的核心亮点：

1. 实现了Apriori算法，并对算法效率进行了实证性研究，应用了潜在解决方案。
2. 进行了多粒度的数据挖掘（选取了句子和段落作为两种篮子，并比较二者区别）。
3. 探索了多个支持度值的应用可能。
4. 进行了多数据集的应用（GutenBerg &&DBLP），对每个数据集进行了多个问题多个角度的研究探讨。

2. 实验原理

大多数关联规则挖掘算法通常采用的一种策略是，将关联规则挖掘任务分解为两个主要的子任务 -- 频繁项集产生（发现满足最小支持度阈值的所有项集），规则的产生（从上一步发现的频繁项集中提取所有高置信度的规则）。

其中，频繁项集产生所需的计算开销远大于产生规则所需的计算开销。因此，在关联规则挖掘中遇到的主要问题是加速频繁项集的产生。

在本部分中，将对本次实验选取的数据挖掘算法 -- Apriori算法展开详细描述。

2.1 Apriori算法原理

Apriori算法主要基于两个定律：

Apriori定律1：如果一个集合是频繁项集，则它的所有子集都是频繁项集。

Apriori定律2：如果一个集合不是频繁项集，则它的所有超集都不是频繁项集。

因此，可以采用层次顺序的方法来实现频繁项集的挖掘。

首先，挖掘一阶频繁项集L1，在此基础上，形成二阶候选项集；然后进行二阶频繁项集的挖掘；以此类推。主要靠“生成候选集--裁剪--计数”的流程。

算法描述如下：

```
1 令 k=1
2 生成K阶频繁项集
3 循环直到没有频繁项集产生：
4    从K阶频繁项集生成K+1阶频繁项集的候选集
5    对K+1阶频繁项集的候选集进行剪枝，如果一个候选频繁项有子集不是k阶频繁项，那么这个候选频繁项也不是频繁项
6    对K+1阶频繁项集的候选集进行筛选，挑选出符合最小支持度要求的频繁项
```

2.2 Apriori算法性能

Apriori算法的性能不高，频繁项集可能出现爆炸现象。对于N个事务，可能出现时间上和空间上$O(N^2)$的复杂度。因此如果选择的最小支持度较小，会出现内存爆炸的现象。

因此，对于Apriori算法，需要着重关注如何降低计算速度，以及减少频繁项集的数目。本实验关注减少频繁项集数目这一点，使用"多最小支持度"进行挖掘。

3. 实验环境搭建及运行

本次环境选取GutenBerg数据集和DBLP数据集进行挖掘，下面分别对各个数据集的筛选和数据预处理等进行说明。

3.1 Gutenburg数据集的模式挖掘

数据集筛选：

林肯演讲集`(见/data/Lincoln)`

篮子的获取：

①分句：调用`nltk`工具包对数据集进行分句，收集每个句子的单词，利用停止词去除了常用介词数词等干扰词汇。`nltk`是一套基于`python`的自然语言处理工具集，在本次实验中只是调用其中的`tokenizer`工具进行文本分句。

②分段：直接根据段落空行进行分段。

源代码：

`(见/src/Gutenberg/)`

```
------ ls /src/Gutenberg/ -----
Associations.py     #获得关联规则，程序主入口
Apriori.py          #获取频繁项集
datasetHandle.py    #数据预处理
stopwords.txt       #停止词列表
##运行：
python Associations.py
```

数据结果：

`见/result/Gutenburg`，保存了下面实验中的频繁项集。挖掘出的关联规则见本文档。

3.2 DBLP数据集

数据集筛选：

选用数据集是 2007 年以来 IJCAI, AAAI, COLT, CVPR, NIPS, KR, SIGIR,SIGKDD 八个会议的数据集。

`(见/data/DBLP)`

篮子的获取：

按照每条记录。

源代码：

`(见/src/DBLP/)`

```
------ ls /src/DBLP/ -----
Apriori.py          #获取频繁项集（由于和Gutenburg基本一致，因此注释较少）
dataset.py          #数据预处理（由于和Gutenburg基本一致，因此注释较少）
task1_active.py     #任务1
task2_group.py      #任务2
task3_topic.py      #任务3
#运行：
python task1_active.py      #任务1
python task2_group.py       #任务2
python task3_topic.py       #任务3
```

数据结果：

`见/result/DBLP/`，保存了各个任务的结果。

4. 实验结论 -- GutenBerg -- 林肯演讲集：挖掘常用词共同出现

本部分主要研究GutenBerg数据集中林肯演讲集中的常用词是否会有共同出现的趋势。采用多个篮子粒度和多研究角度进行研究。

4.1 Sentence模式：以句子作为Basket进行挖掘

4.1.1 数据集的筛选及关联规则的定义

由于数据集太过庞大，而且范围涵盖多个主题和体裁导致挖掘信息杂糅，因此我选取Gutenberg dataset中Lincoln的演讲集部分作为实验数据，并尝试从中挖掘信息。数据集共16个txt文件。首先，把句子作为篮子进行数据挖掘，共31598个句子， 11587个段落

本次实验中，希望挖掘出：

①什么单词组合在同一个句子中出现的概率更高，

②一个单词（或组合）出现后，通常还会出现什么单词。

对置信度定义如下： $confidence = \frac{supp(X_1X_2...X_z)}{supp(X1X2...X_{z-1})}$

得到频繁项集之后，寻找所有由k项集到(k+1)项集符合最小关联度要求的关联规则。

4.1.2 最小支持度的选取理论

选取最小支持度为0.5%

选取标准：

在预实验中，①选取最小支持度为5%，直接导致一阶频繁项为空；

②选取最小支持度为1%，获得58个一阶频繁项集，3个二阶项集和1个三阶项集，较不充分。

③选取最小支持度为0.5%，获得200个一阶频繁项集，15个二阶项集，和1个三阶项集，比1%时充分了一些。

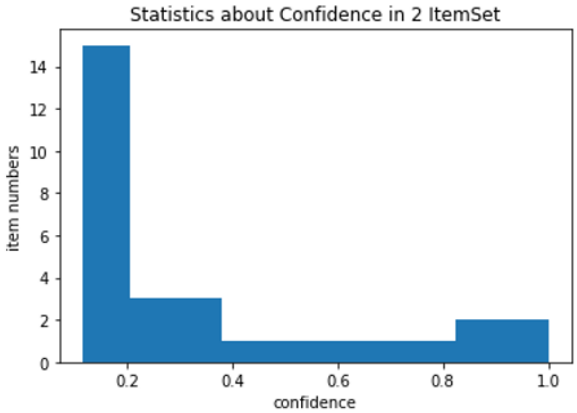
因此，本实验选取最小支持度为0.5%。

4.1.3 最小置信度的选取理论

选取二阶频繁项集的最小置信度为56%（也就是获得Top10）

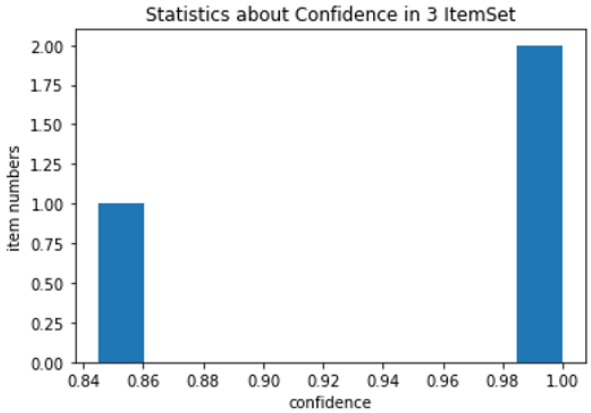
选取标准：

在收集完频繁项集之后，我将每个k项集的置信度统计出来，统计结果如下图。



一项集=>二项集的置信度分布

$confidence = supp(XY)/supp(Y)$



二项集=>三项集的置信度分布

$confidence = supp(XYZ)/supp(XY)$

从图中可以看出：

从一项集推二项集（如果一个单词出现，另一个单词出现）的置信度通常比较低，中位数约为20%

从二项集推三项集（如果两个单词出现，另一个单词出现）的置信度通常比较高，中位数约99.75%（只有一个）。

数据说明：

从二项集推三项集的置信度远高于从一项集推二项集的置信度，是因为只有一个关联规则，缺乏代表性。

选择Top10为置信度标准，为56%

4.1.4 挖掘结果

下面针对两个问题分别进行回答

①什么单词组合在同一个句子中出现的概率更高

即研究符合最小支持度的频繁项集。

```
#一阶频繁项集
slavery people washington united time government war judge constitution union
```

```
#二阶频繁项集
executive   mansion
executive   washington
mansion washington
department  war
president   united
secretary   war
constitution    people
slavery territory
constitution    slavery
people  slavery
#三阶频繁项集
executive   mansion washington
```

实验结论：

- 从出现次数较多的单词，可以直接体会文章的主要内容：人民、团结、黑奴、华盛顿，由这些很容易想到是林肯的演讲集。
- 从共同出现频率最高的单词对上，可以体会单词经常的用法。

②一个单词（或组合）出现后，通常还会出现什么单词

选取符合最小置信度的关联规则，如下：

```
mansion  ============>  executive mansion       Condidence= 1.0
 mansion washington  ============>  executive mansion washington       Condidence= 1.0
 executive washington  ============>  executive mansion washington       Condidence= 0.99
dred  ============>  dred scott       Condidence= 0.98
mansion  ============>  mansion washington       Condidence= 0.84
 executive mansion  ============>  executive mansion washington       Condidence= 0.84
scott  ============>  dred scott       Condidence= 0.83
executive  ============>  executive mansion       Condidence= 0.77
executive  ============>  executive washington       Condidence= 0.65
department  ============>  department war       Condidence= 0.56
```

实验结论：

- 从中可以观察到，前两个关联规则置信度为1，因此可以认为 `executive` 总是和 `mansion` 起出现；
- `scott` 大多数和 `dred` 一起出现； `dred` 大多数和 `scoot` 一起出现；因此可以推测，这两个往往一起出现。
- `washington` 大多和 `mansion` 一起出现； `war` 往往和 `department` 一起出现。

4.2 推广！Paragraph模式：以段落作为Basket进行挖掘

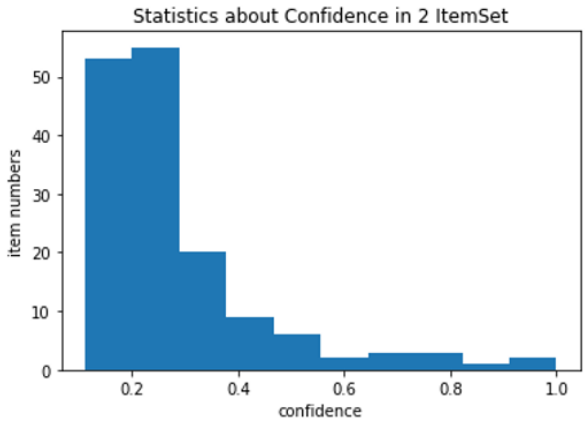
4.2.1 最小支持度和最小置信度的选取

同Sentence模式一样，选取最小支持度为0.5%，结果可以发现701个一阶频繁项集，这个数据太多，所以我决定降低提高置信度。

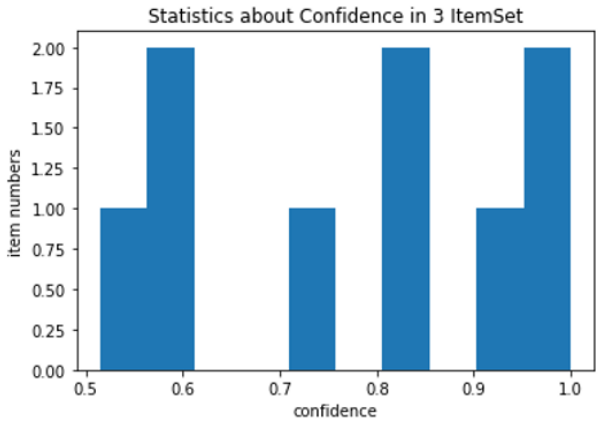
注意在Sentence模式中，最小置信度为1%时，由于获取频繁项集较少，因此被淘汰。在Paragraph模式中，选用最小支持度为1%时，获得了260个一阶频繁项集，77个二阶频繁项集，3个三阶频繁项集。认为数量刚好。

因此选用最小置信度为1%

得到关联规则的置信度分布如下图：



一项集=>二项集的置信度分布
$confidence = supp(XY)/supp(Y)$



二项集=>三项集的置信度分布
$confidence = supp(XYZ)/supp(XY)$

但是为了和Sentence结果进行对比，**仍选取56%作为最小置信度。**

4.2.2 挖掘结果及与Sentence模式的对比

选取符合最小置信度的关联规则，如下：

```
mansion  ============>  executive mansion       Condidence= 1.0
dred  ============>  dred scott       Condidence= 1.0
 mansion washington  ============>  executive mansion washington       Condidence= 1.0
 executive washington  ============>  executive mansion washington       Condidence= 0.99
 department washington  ============>  department war washington       Condidence= 0.94
mansion  ============>  mansion washington       Condidence= 0.84
 executive mansion  ============>  executive mansion washington       Condidence= 0.84
 war washington  ============>  department war washington       Condidence= 0.81
executive  ============>  executive mansion       Condidence= 0.80
scott  ============>  dred scott       Condidence= 0.79
representatives  ============>  house representatives       Condidence= 0.75
 free slave  ============>  free slave slavery       Condidence= 0.70
territory  ============>  slavery territory       Condidence= 0.68
executive  ============>  executive washington       Condidence= 0.68
territories  ============>  slavery territories       Condidence= 0.66
department  ============>  department war       Condidence= 0.62
 slave slavery  ============>  free slave slavery       Condidence= 0.60
north  ============>  north south       Condidence= 0.59
 free slavery  ============>  free slave slavery       Condidence= 0.56
slave  ============>  slave slavery       Condidence= 0.54
```

实验结论：

- 在以Paragraph作为Basket时，设定最小支持度，能够发现更多符合最小支持度的频繁项集。我认为这是符合直觉的。因为一句话的长度是有限的，很多出现次数较高的单词并不会在这句话出现，导致单词的支持度降低。
- 由于频繁项集更多，在以Paragraph作为Basket时，也能找到更多符合最小置信度要求的关联规则。
- 观察在Sentence模式和Paragraph模式发现的关联规则，发现Paragraph模式发现的关联规则包含了Sentence模式的关联规则，

- 综上所述，对这个问题，我认为Paragraph是合适的数据分析的粒度。

4.3 推广！选取多个最小支持度

4.3.1 问题引入

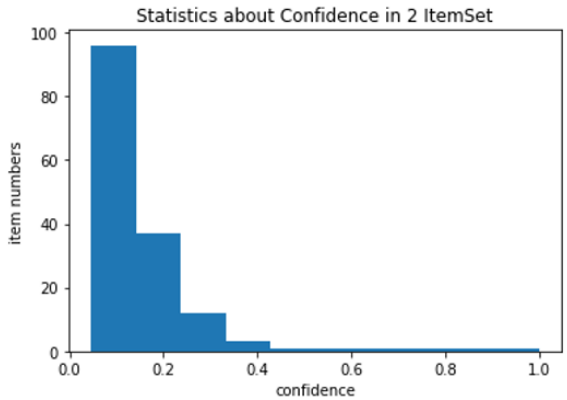
在Sentence模式中，当我们选取最小支持度为1%时，只能获得58个一阶频繁项集，3个二阶项集和1个三阶项集，较不充分。而注意到一个规律：越高阶的频繁项出现的概率越低。尤其是在文本数据库中，收集这样的二阶项集需要放低最小支持度。

因此提出采用多个最小支持度的标准：**一阶频繁项集的最小支持度为1%，二阶及高阶为0.2%。**

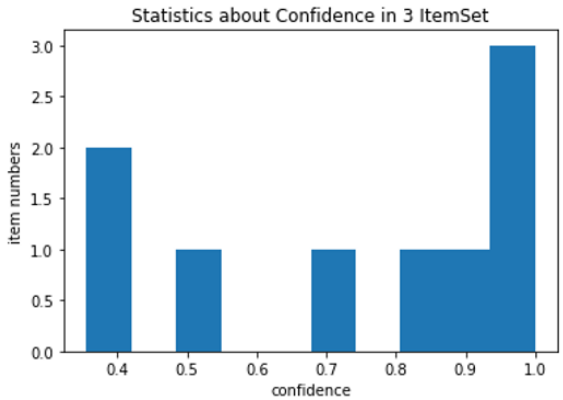
获得了58个一阶频繁项集，77个二阶频繁项集，3个三阶频繁项集。

4.3.2 置信度的分布问题

在收集完频繁项集之后，我将每个k项集的置信度统计出来，统计结果如下图。



一项集=>二项集的置信度分布
$confidence = supp(XY)/supp(Y)$



二项集=>三项集的置信度分布
$confidence = supp(XYZ)/supp(XY)$

从图中可以看出：

从一项集推二项集（如果一个单词出现，另一个单词出现）的置信度通常比较低，中位数约为11.85%。

从二项集推三项集（如果两个单词出现，另一个单词出现）的置信度通常比较高，中位数约84.49%。

数据分析：

这个明显的看似错误的差异，是由于最小置信度选取标准的不同导致的。因为一阶项集的最小支持度选为1%，高阶的最小支持度选为0.2%，因此存在明显差异。但是由于我们要*探求一个单词（或组合）出现后，通常还会出现什么单词*，因此对高阶项集更关注。所以对这一问题暂时不做修正。

选取二阶频繁项集的最小置信度为65%（也就是获得Top10）

4.3.3 挖掘结果

下面针对两个问题分别进行回答

①什么单词组合在同一个句子中出现的概率更高

选取（即获得Top10的关联规则，列举如下）

```
#出现频率最高的单词(Top 10):
slavery people washington united time government war judge constitution union
#出现频率最高的单词对(Top 10):
executive   mansion
executive   washington
mansion washington
department  war
secretary   war
president   united
constitution    people
slavery territory
constitution    slavery
people  slavery
#出现频率最高的三元组:
people  slavery   territory
executive   mansion   washington
president   proclamation    united
department  war  washington
```

②一个单词（或组合）出现后，通常还会出现什么单词

```
mansion  ============>  executive mansion       Condidence= 1.0
 mansion washington  ============>  executive mansion washington       Condidence= 1.0
 executive washington  ============>  executive mansion washington       Condidence= 0.99
 department washington  ============>  department war washington       Condidence= 0.95
 war washington  ============>  department war washington       Condidence= 0.93
 executive mansion  ============>  executive mansion washington       Condidence= 0.84
 mansion  ============>  mansion washington       Condidence= 0.84
executive  ============>  executive mansion       Condidence= 0.77
 people territory  ============>  people slavery territory       Condidence= 0.69
 executive  ============>  executive washington       Condidence= 0.65
```

实验结论：

- 观察挖掘结果和上面两种模式的对比可以发现，选用多个最小置信度时，挖掘出的关联规则中，二阶项集推三阶项集占比要大大增加。这是由于4.3.2提出的置信度分布问题，由于最小支持度不同，在研究一阶项集和二阶项集的关联规则时，会使得置信度成倍降低。
- 二阶项集中拔尖的仍然可以被选出，三阶项集中包含了Sentence模式和Paragraph模式的关联规则。
- 如果要着重关注高阶项集的关联规则，应该采用多个最小支持度的方法。

5. 实验结论 -- DBLP -- 论文团队与主题

5.1 问题描述及数据集选取

选用数据集是 2007 年以来 IJCAI, AAAI, COLT, CVPR, NIPS, KR, SIGIR,SIGKDD 八个会议的数据集。

问题1：每个会议都有自己的“支持者”，也就是在该会议上发表文章的研究者。首先，需要找出每个会议各自的研究者。在此基础上，根据研究者每年发表的文章数，可以判断出哪些研究者仍然是活跃的，而哪些研究者已经不再活跃。

问题2： 有一些研究者经常在一起合作，可以称之为一个“团队”。通过频繁模式挖掘，可以从数据集中找出三个人以上的“团队”。

问题3： 每一篇论文都会涉及到一些主题。可以通过对论文标题进行频繁模式挖掘找出主题词。根据每个团队发表的论文的标题，就可以确定这个团队最经常研究的主题了。

5.2 任务1 -- 寻找活跃支持者

为了找出每个会议各自的研究者，对数据集进行扫描：

对于数据集的每一条数据， 如果其作者不在会议的研究者中，则添加进去。扫描结束后，就得到了每个会议全部的研究者。完整的结果保存在 authors.txt 中。

为了找出活跃的研究者，规定**最小支持度为5。**同样对数据集进行扫描，统计出每个作者在每个会议上每年发表的文章数。如果一个作者在一个会议上一年发表的文章数大于 5，就认为该作者在该会议上于该年是活跃的。如果一个作者在最近三年（2015、2016、2017 年）在一个会议上至少有一年是活跃的，就认为该作者在该会议上“依然活跃”，否则认为“不再活跃”。

挖掘结果：

完整的活跃研究者结果保存在***author.txt**

例如，SIGIR的依然活跃的研究者有：

```
Tat-Seng Chua, Liqiang Nie, Yiqun Liu, Shaoping Ma, Min Zhang, Pavel Serdyukov, Maarten de Rijke, Jimmy J. Lin, Charles L. A. Clarke, Guido Zuccon, Jamie Callan, Jimmy Lin, W. Bruce Croft, Leif Azzopardi, Adam Roegiest, Craig Macdonald, Iadh Ounis, Krisztian Balog, Chenyan Xiong, Jaap Kamps, Hamed Zamani。
```

5.3 任务2 -- 寻找团队

设定最小支持度为3。

得到每一年发文章数大于等于 3 的三个人以上的“团队”。

完整的团队结果保存在 *group.txt 中。

例如，2007 年，发文章数大于等于 3 的、三个人以上的“团队”有：

```
1. Deng Cai, Jiawei Han 0001, Xiaofei He
2. Dou Shen, Qiang Yang 0001, Zheng Chen
3. Qiang Yang 0001, Jeffrey Junfeng Pan, Sinno Jialin Pan
4. Wiebe van der Hoek, Michael Wooldridge, Thomas Ågotnes
5. Brian D. Davison 0001, Lan Nie, Baoning Wu
6. Guy Shani, Solomon Eyal Shimony, Ronen I. Brafman
7. Junsong Yuan, Ying Wu, Ming Yang
8. Kangmiao Liu, Jiajun Bu, Chun Chen
9. Roberto Cipolla, Tae-Kyun Kim, Shu-Fai Wong
```

5.4 任务3 -主题与团队

通过对所有会议的所有热门主题进行汇总统计可以得到主题词的参考范围，从中选出具有代表性的几个词作为任务要求事先给定的主题词，之后再根据任务一第二问挑选出来的每年的 团队做一个匹配，即对于每年的每个团队都找出他们常涉猎的主题，

例如 2017 年的部分结 果展示如下：

```
[neural, networks] Hongyuan Zha, Junchi Yan, Xiaokang Yang
[learning] Deng Cai, Xiaofei He, Yueting Zhuang, Zhou Zhao
[learning] Chunyuan Li, Lawrence Carin, Ricardo Henao, Yunchen Pu
[learning] Jianshu Li, Jiashi Feng, Shuicheng Yan, Zhecan Wang
[neural, networks] Jiashi Feng, Shuicheng Yan, Xiaodan Liang
[neural, networks] Chunyuan Li, Lawrence Carin, Yunchen Pu, Zhe Gan
[learning] Guiguang Ding, Jungong Han, Yuchen Guo, Yue Gao
[learning] Dan Xu, Elisa Ricci, Nicu Sebe, Wanli Ouyang, Xiaogang Wang
[deep, learning] Anton van den Hengel, Chunhua Shen, Lingqiao Liu
```

6. 算法性能分析

Apriori算法的性能不高，频繁项集可能出现爆炸现象。对于N个事务，可能出现时间上和空间上$O(N^2)$的复杂度。因此如果选择的最小支持度较小，会出现内存爆炸的现象。

因此，对于Apriori算法，需要着重关注如何降低计算速度，以及减少频繁项集的数目。本实验关注减少频繁项集数目这一点，使用“多最小支持度”进行挖掘。

6.1 针对Sentence模式的时间复杂度实证研究

在本实验中，我针对Sentence模式的最小支持度，探究Apriori算法所耗费的时间，从而视图验证复杂度为$O(N)$的说法。

最小支持度	耗时(s)	一项集	二项集	三项集
5%	15	0		
1%	90	58	3	1
0.50%	331	200	15	1

我选取了最小支持度为5%，1%和0.5%，进行他们的耗时研究。发现耗时呈指数增长。

6.2 效率改进--多支持度

注意到一个规律：越高阶的频繁项出现的概率越低。尤其是在文本数据库中，收集这样的二阶项集需要放低最小支持度。

因此提出采用多个最小支持度的标准：一阶频繁项集的最小支持度为1%，二阶及高阶为0.2%。

获得了58个一阶频繁项集，77个二阶频繁项集，3个三阶频繁项集。

最小支持度	耗时(s)	一项集	二项集	三项集
1%	90	58	3	1
1%	59	58	77	3

对比发现，耗时明显减少，而且可以发现更多项集。对比试验数据，发现很多规律都是相同的，因此可以发现有效规律。

7. 试验总结

本报告的核心亮点：

1. 实现了Apriori算法，并对算法效率进行了实证性研究，应用了潜在解决方案。
2. 进行了多粒度的数据挖掘（选取了句子和段落作为两种篮子，并比较二者区别）。
3. 探索了多个支持度值的应用可能。
4. 进行了多数据集的应用（GutenBerg &&DBLP），对每个数据集进行了多个问题多个角度的研究探讨。