

Data Mining - Homework 2

Robert-Andrei Damian and Alice De Schutter
Discovery of Frequent Itemsets and Association Rules

22 november 2021

Task

- (1) Write a program that implements the A-Priori algorithm for finding frequent itemsets with support at least s in a dataset of sales transactions.
- (2) **BONUS TASK:** Develop and implement an algorithm for generating association rules between frequent itemsets discovered with the A-Priori algorithm. The rules must have support at least s and confidence at least c , where s and c are given as input parameters.

1 Detailed Information

Apache Spark was used.

- Our implementation of (1) does the following:
 1. Counts frequent singleton occurrences with support s .
 2. Recursively calls a function called “gather_all_groups()” which counts frequent doubletons, tripletons... etc. with support size s . The creation of larger groups is done with a “helperDataFrame” which consists of the frequent singletons joined with the baskets in which they are included. Result is first shown with a spark DataFrame where each item is represented by a group (column: “group”; the occurrences are stored in column: “occurrences”). Result is then also shown in a second DataFrame which displays the group_sizes that exist with the given support size s , and their associated group_size counts (i.e. there exists 9 doubletons”).
 3. Cached solutions for values of support-size $s = \{700, 1000, 2000\}$ can be found in the “datafolder”.
- **BONUS:** Our implementation of (2) does the following:
 1. Finds all of the possible association rules which correspond to the frequent item groups (double-tons, triple-tons etc.). This is achieved with the itertools permutations function.
 2. Confidence is computed for all of these association rules. The rules which have confidence c or higher are kept, others are discarded.

Instructions on how to build and run the program

- pip install pyspark
- spark-submit frequent_items.py

Figure 1 shows information about optional command line parameters:

```

optional arguments:
  -h, --help            show this help message and exit
  --support-size SUPPORT_SIZE
                        Set the minimum support size
  --disable-cache        Do not use cached data
  --confidence-threshold CONFIDENCE_THRESHOLD
                        Set the confidence_threshold

```

Figure 1: Optional arguments

Since the implementation of our A-priori algorithm is slow for certain values of support values s (i.e $s = 700$), we included the option to enable/ disable the cache. This means that the program can use previously cached data in order to perform the Bonus task. By default, the cache is enabled.

2 Results

The results shown below are found with command line parameters:

- $support_size = 1000$ (default value)
- $confidence_threshold = 0.6$ (default value)
- `--disable-cache`

Task 1:

```

Checking for 1010228 pairs. Aiming for 1000 support!
Checking for 6530628 pairs. Aiming for 1000 support!
Identified 9 new supported groups!
Checking for 19432 pairs. Aiming for 1000 support!
Identified 1 new supported groups!
Checking for 0 pairs. Aiming for 1000 support!
Identified 0 new supported groups!
+-----+
|group|occurrences|
+-----+
|[471]|      2894|
|[496]|      1428|
|[392]|      2420|
|[540]|      1293|
|[897]|      1935|
|[623]|      1845|
|[516]|      1544|
|[ 31]|      1666|
|[580]|      1667|
|[ 85]|      1555|
|[458]|      1124|
|[883]|      4902|
|[804]|      1315|
|[970]|      2086|
|[472]|      2125|
|[853]|      1804|
|[296]|      2210|
|[513]|      1287|
|[322]|      1154|
|[ 78]|      2471|
+-----+
only showing top 20 rows

+-----+
|group_size|count(group_size)|
+-----+
|         1|                375|
|         2|                 9|
|         3|                 1|
+-----+

```

Figure 2: Spark DataFrame output for task 1, with support-size = 1000

Task 2 (bonus):

group	premise	conclusion	group_support	premise_support	confidence
[39, 704, 825]	[704, 825]	[39]	1035	1102	0.9392014519056261
[39, 704, 825]	[39, 704]	[825]	1035	1107	0.9349593495934959
[39, 704, 825]	[39, 825]	[704]	1035	1187	0.8719460825610783
[39, 704]	[704]	[39]	1107	1794	0.617056856187291
[704, 825]	[704]	[825]	1102	1794	0.6142697881828316

Figure 3: Spark DataFrame output for task 2, with support-size = 1000

The results shown below are found with command line parameters:

- *support_size* = 700
- *confidence_threshold* = 0.6
- *cache_enabled* = True (default)

Task 2 output:

group	occurrences
[256]	785
[678]	1329
[345]	801
[490]	1066
[595]	797
[752]	2578
[790]	1094
[809]	2163
[90]	1875
[32]	4248
[622]	826
[630]	1523
[469]	1502
[75]	3151
[640]	932
[265]	1359
[835]	732
[782]	2767
[203]	861
[899]	1252

only showing top 20 rows

Figure 4: Spark DataFrame output for task 2 with support_size = 700 and cache enabled

group	premise	conclusion	group_support	premise_support	confidence
[33, 283, 346, 515]	[33, 283, 515]	[346]	763	786	0.9707379134860051
[33, 217, 283, 346]	[33, 217, 283, 515]	[346]	732	755	0.9695364238410596
[33, 217, 346, 515]	[33, 217, 515]	[346]	764	789	0.9683143219264893
[217, 283, 346, 515]	[217, 283, 515]	[346]	773	799	0.967459324155194
[33, 346, 515]	[33, 515]	[346]	797	824	0.9672330097087378
[33, 217, 283, 346]	[33, 217, 283]	[346]	766	793	0.9659520807061791
[283, 346, 515]	[283, 515]	[346]	806	835	0.9652694610778443
[290, 458, 888, 969]	[290, 458, 969]	[888]	720	748	0.9625668449197861
[208, 290, 458, 969]	[290, 458, 969]	[208]	719	748	0.9612299465240641
[208, 290, 888, 969]	[208, 290, 888]	[969]	734	764	0.9607329842931938
[208, 290, 888, 969]	[208, 290, 969]	[888]	734	764	0.9607329842931938
[33, 217, 283, 515]	[33, 283, 515]	[217]	755	786	0.960559796437659
[208, 458, 888, 969]	[208, 458, 969]	[888]	725	755	0.9602649006622517
[290, 458, 888, 969]	[290, 458, 888]	[969]	720	750	0.96
[208, 290, 458, 888]	[290, 458, 888]	[208]	720	750	0.96
[290, 888, 969]	[290, 969]	[888]	765	797	0.9598494353826851
[217, 346, 515]	[217, 515]	[346]	809	843	0.9596678529062871
[208, 290, 888, 969]	[290, 888, 969]	[208]	734	765	0.9594771241830066
[458, 888, 969]	[458, 969]	[888]	756	788	0.9593908629441624
[33, 217, 283, 346]	[33, 283, 346, 515]	[217]	732	763	0.9593709043250328

only showing top 20 rows

Figur 5: Spark DataFrame output for task 2 support_size = 700 and cache enabled