Frequent Itemsets and Association Rules

Haotang Wu, Cristina Zhang

1 Introduction

This report details the implementation of the Apriori algorithm to solve the problem of discovering frequent itemsets and generating association rules from a dataset of sales transactions. The implementation is evaluated for its performance and scalability.

2 How to Run

The program can be executed via the command line using the following syntax:

```
python apriori.py --dataset_file <path_to_dataset>
                  --support <support_threshold>
                  --min_confidence <confidence_threshold>
                  --verbose
```

```
(base) cristina@Cristinas-MacBook-Air assignment 2 % python apriori.py --help
usage: apriori.py [-h] --dataset_file DATASET_FILE --support SUPPORT --min_confidence MIN_CONFIDENCE [--verbose]

Apriori Algorithm with Association Rules

optional arguments:
  -h, --help            show this help message and exit
  --dataset_file DATASET_FILE
                        Path to the dataset file (.dat)
  --support SUPPORT     Minimum support threshold
  --min_confidence MIN_CONFIDENCE
                        Minimum confidence threshold
  --verbose             Enable verbose output
```

Figure 1: Help support

```
(base) cristina@Cristinas-MacBook-Air assignment 2 % python apriori.py --dataset_file T10I4D100K.dat --support 1000 --min_confidence 0.6 --verbose

Level 1 | Frequent itemsets: 375
Level 2 | Frequent itemsets: 9
Level 3 | Frequent itemsets: 1
Time for sub-problem 1 (frequent itemsets): 3.97s
(704,) -> (39,) | Support: 1107, Confidence: 0.62
(704,) -> (825,) | Support: 1102, Confidence: 0.61
(39, 704) -> (825,) | Support: 1035, Confidence: 0.93
(39, 825) -> (704,) | Support: 1035, Confidence: 0.87
(704, 825) -> (39,) | Support: 1035, Confidence: 0.94
Time for sub-problem 2 (association rules): 0.00s

Number of rules generated: 5
```

Figure 2: How to run in terminal

3 Code Description

3.1 Class `Apriori`

The `Apriori` class implements the core functionality for frequent itemset discovery.

3.1.1 Key Methods

- `count_single_items`: Counts the occurrences of individual items in the dataset.
- `generate_candidates`: Generates candidate k-itemsets from frequent $(k-1)$-itemsets.
- `filter_candidates`: Filters candidate itemsets by their support counts.
- `find_frequent_itemsets`: Orchestrates the entire Apriori process to find frequent itemsets.

3.2 Class `AssociationRules`

The `AssociationRules` class is responsible for generating rules from frequent itemsets.

3.2.1 Key Methods

- `generate_rules`: Computes rules by iterating over frequent itemsets and filtering based on confidence.

4 Conclusion

The implementation of the Apriori algorithm successfully identifies frequent itemsets and generates association rules from the dataset. The results demonstrate scalability and correctness, with reasonable execution time for both subproblems.