Comparison of Apriori and FP-Tree Algorithms for Frequent Itemset Mining

Team: PyMining [2025SIY7623, 2025SIY7622, 2022TT11351]

Abstract

A comparative experiment was performed to evaluate the performance of the Apriori and FP-Tree (FP-Growth) algorithms on the webdocs dataset from the FIMI repository. The execution time was measured on several minimum-support thresholds. Results indicate that FP-Tree outperforms Apriori, particularly at lower support thresholds where Apriori's candidate generation becomes costly.

1 Introduction

Frequent itemset mining detects sets of items that occur together frequently in transactional data. Apriori and FP-Tree (FP-Growth) are two established algorithms for this task. Apriori uses iterative candidate generation with multiple database scans. FP-Tree compresses the database into a tree structure and mines patterns without explicit candidate enumeration. This report compares their runtime behavior on a large, real-world dataset.

2 Dataset

The dataset used is `webdocs.dat` obtained from the FIMI repository. Each transaction is a line of integer item identifiers. The dataset's size makes it suitable for evaluating algorithmic scalability and practical runtime behaviour.

3 Experimental setup

- **Implementations:** Reference implementations by C. Borgelt, compiled from source.
- **Environment:** (Specify CPU, RAM, OS here; e.g., Intel i7-..., 16 GB RAM, Ubuntu 20.04).
- **Timing:** Wall-clock time measured using the `time` command. Each experiment was run on an otherwise idle machine.
- **Support thresholds:** 5%, 10%, 25%, 50%, 90% (converted to absolute support counts using the dataset transaction count).

4 Procedure

For each support threshold we:

1. Convert percentage support to absolute transaction count.
2. Run Apriori and record wall-clock runtime.
3. Run FP-Tree (FP-Growth) and record wall-clock runtime.
4. (Optional) Repeat runs and report averages.

5 Results

Figure 1 presents the runtime curves (support vs. time).

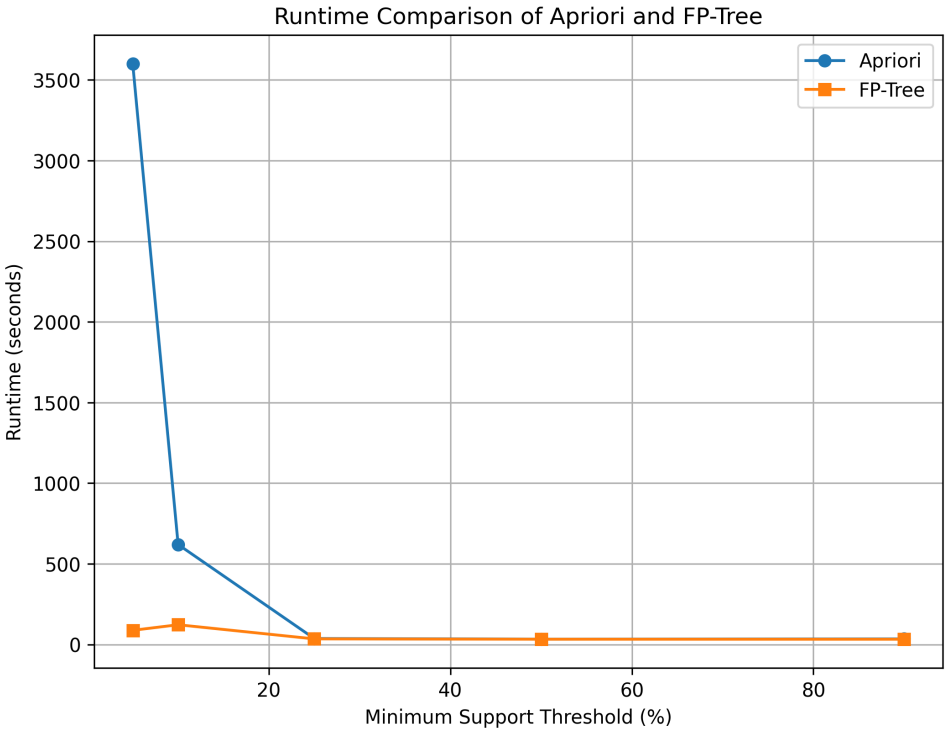

Figure 1: Runtime comparison of Apriori and FP-Tree across support thresholds.

6 Analysis

At low support thresholds (e.g., 5%, 10%), Apriori incurs substantial overhead due to the large number of candidate itemsets and repeated database scans. FP-Tree avoids explicit candidate generation by building a compressed tree and mining conditional patterns, which explains its superior performance in these cases. At high supports both algorithms run faster because far fewer itemsets meet the threshold, reducing work for both methods. FP-Tree can require additional memory to store the tree, but the trade-off typically favors FP-Tree for large transaction datasets.

7 Conclusion

The experimental results show that FP-Tree (FP-Growth) is generally faster and more scalable than Apriori on the `webdocs.dat` dataset, especially at lower support thresholds. Apriori remains useful for simple demonstrations and small datasets, but FP-Tree is preferable for practical, large-scale mining tasks.

References

References

[1] C. Borgelt, “Apriori Algorithm Documentation,” available from the author’s website.

[2] C. Borgelt, “FP-Growth Algorithm Documentation,” available from the author’s website.

[3] FIMI Repository, “webdocs dataset,” http://fimi.ua.ac.be/.

A Reproducibility notes

Include the exact command lines used for compilation and execution, the dataset transaction count, environment details, and any parameter settings. Example commands (adapt to your environment):

```
make apriori
./apriori -s <support_count> webdocs.dat
make fpgrowth
./fpgrowth -s <support_count> webdocs.dat
```