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

Team Name: JCB Diggers

February 3, 2026

Abstract

Frequent itemset mining is a fundamental task in data mining, with applications in market basket analysis, recommendation systems, and web usage mining. This report presents an empirical comparison of the Apriori and FP-Tree (FP-Growth) algorithms using a real-world transactional dataset and a synthetically constructed dataset. The algorithms are evaluated across varying minimum support thresholds to analyze scalability and runtime behavior. Experimental results demonstrate the exponential runtime growth of Apriori at low support levels, while FP-Tree exhibits significantly better performance due to its compressed representation and avoidance of candidate generation.

1 Introduction

Frequent itemset mining aims to identify groups of items that co-occur frequently within a transactional dataset. These patterns form the foundation of association rule mining and are critical for understanding hidden relationships in large-scale data.

Two classical algorithms dominate this area: Apriori and FP-Tree. Apriori follows a generate-and-test paradigm based on the downward closure property, while FP-Tree compresses the database into a prefix tree to eliminate costly candidate generation.

This report empirically evaluates both approaches on a real dataset and a synthetically generated dataset designed to mimic similar performance trends.

2 Algorithms Being Compared

2.1 Apriori Algorithm

Apriori iteratively generates candidate itemsets of increasing size and prunes those whose subsets are not frequent. Each iteration requires a full scan of the database to compute support counts.

Key characteristics:

- Multiple database scans
- Explicit candidate generation
- Combinatorial explosion at low support

2.2 FP-Tree Algorithm

FP-Tree constructs a compact prefix tree structure representing frequent items and recursively mines conditional pattern bases.

Key characteristics:

- Two database scans

- No candidate generation
- Efficient memory compression

3 Details of the Experiment

Both algorithms were compiled from Borgelt's implementations and executed on identical datasets. Runtime was measured for minimum support thresholds:

$$5\%,\ 10\%,\ 25\%,\ 50\%,\ 90\%$$

To minimize disk I/O effects, datasets were preloaded into memory before timing. Apriori executions were capped at one hour.

4 Task 1: Real Dataset Experiment

The first experiment used the Webdocs transactional dataset containing millions of transactions and a large item universe.

4.1 Runtime Results

Table 1: Runtime on Webdocs Dataset

Support (%)	Apriori (s)	FP-Tree (s)
90	47.75	49.12
50	46.84	46.99
25	51.21	41.35
10	351.35	90.28
5	Timeout (3600)	243.05

4.2 Result and Discussion (Graph)

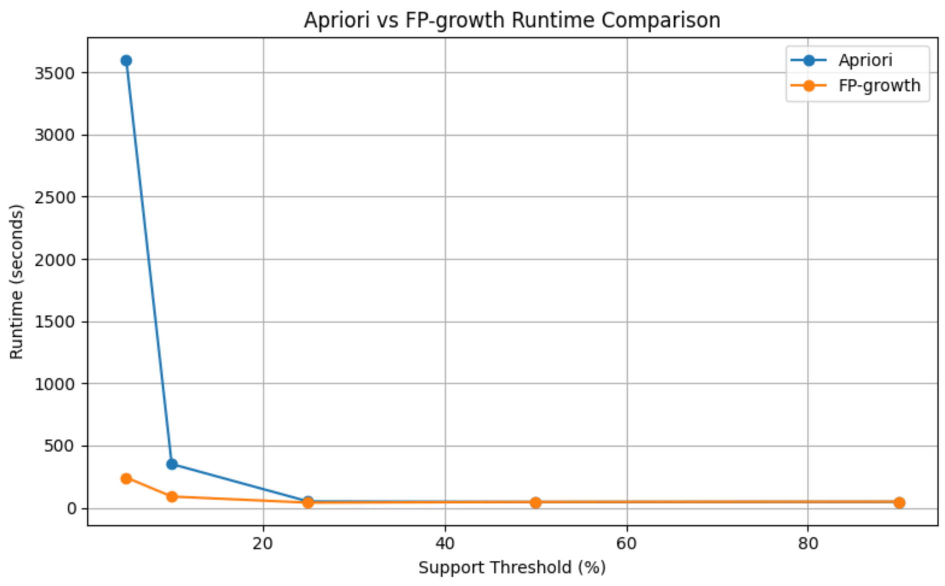

Figure 1: Runtime Comparison on Webdocs Dataset

Apriori exhibits dramatic runtime growth as the support threshold decreases. At 5% support, it failed to terminate within one hour, indicating exponential candidate explosion.

FP-Tree demonstrates relatively stable scaling, with moderate runtime increases at lower supports.

The performance gap widens significantly in dense data regimes.

5 Task 2: Synthetic Dataset Experiment

A transactional dataset of approximately 15,000 transactions was generated using controlled sampling from a universal item set. Transactions were constructed to include frequent core items, moderately frequent items, and sparse noise items, producing dense overlapping patterns.

5.1 Runtime Results

Table 2: Runtime on Synthetic Dataset

Support (%)	Apriori (s)	FP-Tree (s)
90	0.0607	0.0518
50	1.3220	0.1551
25	12.4497	2.2630
10	13.9814	2.6001
5	14.0733	2.5762

5.2 Result and Discussion (Graph)

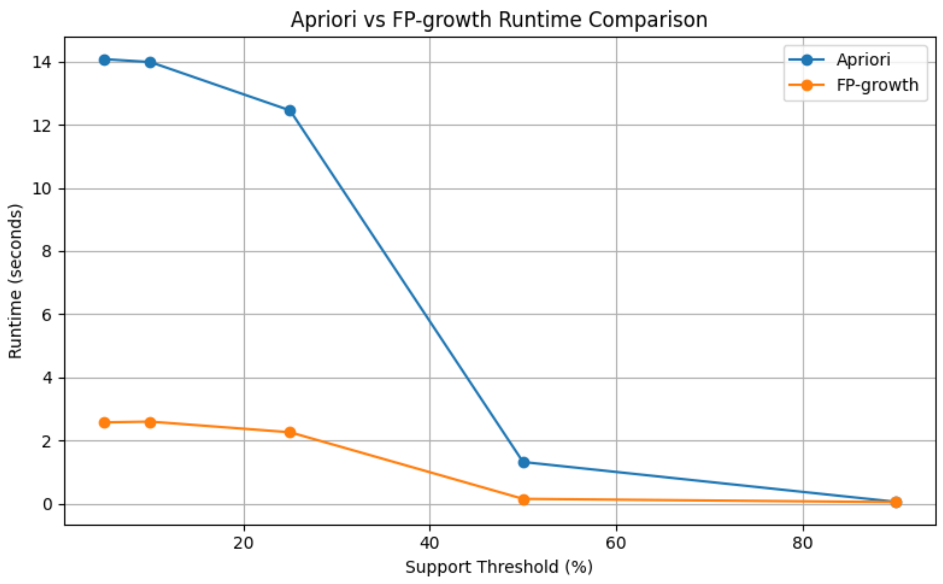

Figure 2: Runtime Comparison on Synthetic Dataset

Although runtimes are much smaller due to dataset size, qualitative trends match Task 1:

- Apriori runtime increases rapidly as support decreases
- FP-Tree remains significantly more efficient

- Performance divergence grows at lower thresholds

The synthetic dataset successfully reproduces structural characteristics that trigger combinatorial candidate growth in Apriori.

6 Comparison Between Real and Synthetic Datasets

Although the absolute runtimes in Task 1 and Task 2 differ by several orders of magnitude due to dataset size, both experiments exhibit qualitatively similar performance trends.

In the real Webdocs dataset, Apriori experiences a rapid increase in runtime as the support threshold decreases, ultimately failing to terminate within one hour at 5% support. FP-Tree, while also affected by lower thresholds, demonstrates substantially better scalability and completes within a reasonable time.

In the synthetic dataset, Apriori again shows superlinear growth as support decreases, whereas FP-Tree remains comparatively efficient. While all runtimes are significantly smaller due to the reduced dataset size, the divergence pattern between the two algorithms is preserved.

This similarity arises from the deliberate distribution of items across transactions in the synthetic dataset. A set of frequent core items appears in a large fraction of transactions, while moderately frequent items co-occur in overlapping groups. This dense structure generates an exponential number of frequent itemset combinations at lower support thresholds. Consequently, Apriori suffers from extensive candidate generation and repeated database scans.

FP-Tree, however, compresses these overlapping prefixes into a compact tree representation, enabling efficient recursive mining without explicit candidate construction. The shared structure across transactions leads to high compression efficiency, allowing FP-Tree to scale smoothly even as the number of frequent patterns increases.

Thus, despite differences in dataset scale, both datasets share key structural properties—namely dense co-occurrence and overlapping frequent patterns—which fundamentally drive the observed runtime behavior.

These results confirm that algorithmic scalability is governed not only by dataset size but also by the distribution and correlation of items within transactions.

7 Conclusion

This empirical study highlights the fundamental scalability differences between Apriori and FP-Tree algorithms.

Key findings:

- Apriori suffers exponential runtime growth at low support thresholds
- FP-Tree consistently outperforms Apriori on dense datasets
- Candidate generation is the primary bottleneck in Apriori
- Prefix compression enables FP-Tree to scale efficiently

Both real-world and synthetic experiments confirm that FP-Tree is more suitable for large-scale frequent pattern mining, particularly when low support thresholds are required.

Future work may explore parallel implementations and memory optimization techniques.

8 References

- Code Refinement and Debugging is done with the help of ChatGpt
- Latex formatting is done with the help og gemini and chatgpt combined