Analysis of Frequent Itemset Mining Techniques

Yearning For The Mines

February 2025

1 Introduction

Frequent itemset mining is a crucial task in data mining, used extensively in market basket analysis, recommendation systems, and anomaly detection. In this study, we compare the runtime performance of the Apriori and FP-tree algorithms across different minimum support thresholds.

2 Methods

2.1 Apriori

Apriori is a level-wise, candidate-generation-based algorithm that incrementally builds larger frequent itemsets. It iteratively scans the dataset multiple times, pruning infrequent itemsets using the downward closure property.

2.2 FP-tree

FP-tree (Frequent Pattern Tree) is a more efficient alternative that uses a prefix-tree structure to store transactions compactly. Instead of generating candidate itemsets, it mines frequent patterns directly from the tree, reducing database scans and improving efficiency.

3 Results

Figure 1 shows the runtime of the Apriori and FP-tree algorithms at various minimum support thresholds.

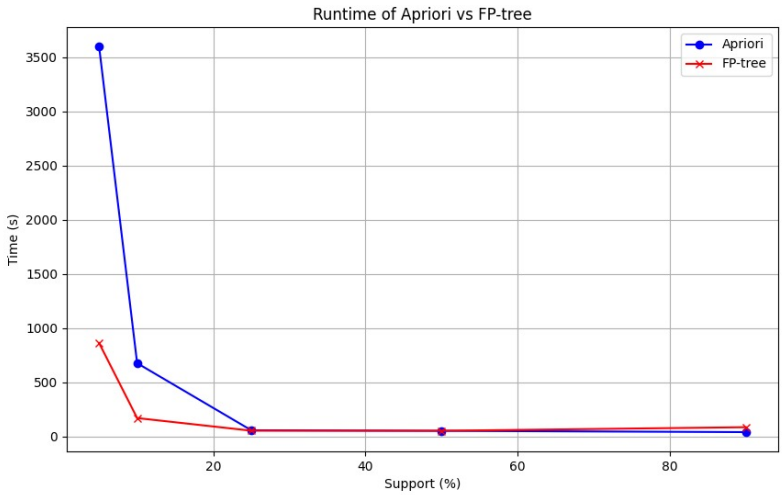

Figure 1: Runtime of Apriori vs. FP-tree at Different Support Levels

4 Discussion

The plot illustrates the following trends:

- At low support values (e.g., 5% and 10%), Apriori takes significantly more time than FP-tree.
- At 5% support, Apriori exceeds 3500 seconds, while FP-tree completes in around 800 seconds.
- As support increases (25%, 50%, 90%), both Apriori and FP-tree have similar runtimes, approaching near zero.
- FP-tree consistently outperforms Apriori across all support values, especially for low support thresholds.

4.1 Why is FP-tree Faster than Apriori?

- **Candidate Generation Overhead:** Apriori follows a level-wise approach, iteratively generating larger frequent itemsets. This results in an exponential number of candidate sets, making it slow at low support thresholds.
- **FP-tree's Compact Structure:** FP-tree avoids candidate generation by using a compressed prefix-tree representation, allowing it to efficiently extract frequent itemsets without repeated database scans.

- **Database Scanning:** Apriori scans the dataset multiple times, leading to high disk I/O overhead. FP-tree requires only two scans: one to build the tree and another for pattern mining.
- **Memory Efficiency:** FP-tree stores only essential transactions, making it more space-efficient. Apriori needs to store and retrieve large candidate itemsets, increasing memory usage.

4.2 Trends and Growth Rate Explanation

- Apriori's runtime drops exponentially as support increases. At low support ($\leq$10%), it takes an extremely long time due to excessive candidate generation. At higher support ($\geq$50%), fewer frequent itemsets are generated, reducing runtime.
- FP-tree shows a more stable and efficient runtime at all support levels. Since it avoids candidate generation, its performance is less affected by support thresholds.

5 Conclusion

Our analysis suggests that for large datasets and low support thresholds, FP-tree is significantly more efficient than Apriori. Apriori becomes impractical for low support values but performs comparably at high support levels. FP-tree remains the preferred choice due to its compact structure and efficient mining capabilities.