COL761: Data Mining - Assignment 1

Task 1: Comparison of Apriori and FP-Tree Algorithms

Spandan 2022CS51138
Viraat 2022CS51651
Gaurav 2022CS51144

February 3, 2026

1 Introduction

The objective of this task is to conduct an empirical comparison between two fundamental frequent itemset mining algorithms: **Apriori** and **FP-Tree (FP-Growth)**. Both algorithms were evaluated on the `webdocs.dat` dataset across five support thresholds: 90%, 50%, 25%, 10%, and 5%.

2 Experimental Setup

2.1 Environment

The experiments were conducted using the implementations provided by Christian Borgelt. The source code was compiled using the `make` command. Execution was managed via a Bash script with a timeout of 3600 seconds per run to ensure computational feasibility.

2.2 Dataset

- **Dataset:** `webdocs.dat`
- **Characteristics:** A large-scale dataset containing web document transactions, characterized by high dimensionality and varying transaction lengths.

3 Results

The execution times (in seconds) for both algorithms at specified support thresholds are summarized in Table 1.

Table 1: Runtime Comparison (Seconds)

Support (%)	Apriori Time (s)	FP-Growth Time (s)
90%	64.346	56.254
50%	60.326	59.695
25%	61.307	57.869
10%	924.189	173.423
5%	3600	120.801

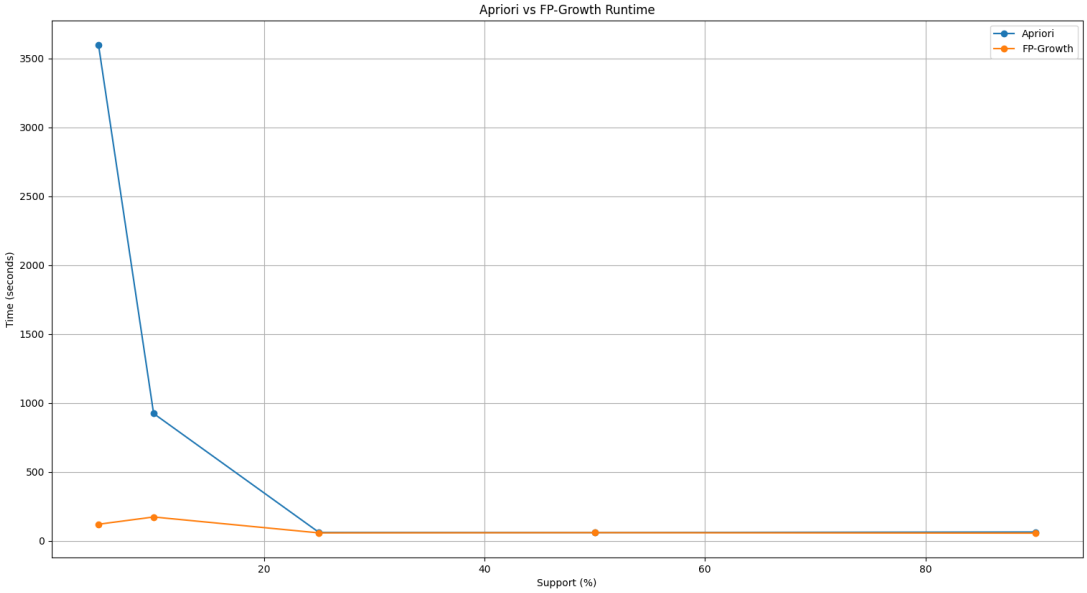

Figure 1: Runtime vs Support Threshold for Apriori and FP-Growth

3.1 Performance Visualization

The following plot illustrates the relationship between the support threshold (x-axis) and the execution runtime (y-axis).

4 Analysis of Observations

4.1 Impact of Support Threshold

As expected, the runtime for both algorithms increases exponentially as the support threshold decreases. This is due to the exponential growth in the number of frequent itemsets that satisfy the minimum support criteria.

4.2 Apriori vs. FP-Growth

- **Apriori:** Follows a candidate generation-and-test approach. Its performance degrades significantly at lower support thresholds because it requires multiple passes over the dataset and generates a massive number of candidate itemsets.
- **FP-Growth:** Utilizes a compressed tree structure (FP-Tree) and a divide-and-conquer approach without explicit candidate generation. It generally outperforms Apriori, especially at lower support levels, by reducing the number of dataset scans to two.

5 Conclusion

The empirical results confirm that FP-Growth is more efficient and scalable than Apriori for large datasets like `webdocs.dat`. While Apriori is simpler to implement, its overhead in candidate generation makes it less suitable for dense datasets or low support thresholds.

6 Q 1.2

7 Implementation Details and Design Rationale

7.1 Objective

The goal of this sub-task is to synthetically construct a transactional dataset of approximately 15,000 transactions such that the runtime trends of Apriori and FP-Growth qualitatively resemble the given reference plot. Exact runtime matching is not required; instead, the focus is on reproducing the relative behavior across varying minimum support thresholds. The dataset generator must be parameterized by the size of the universal itemset and the number of transactions.

7.2 High-Level Design

The dataset is generated using a controlled probabilistic structure that explicitly shapes the frequent itemset lattice at different support thresholds. The key idea is to create a *plateau* region in the number of frequent itemsets for moderate support values (10%, 25%, 50%), while forcing a sharp reduction at high support (90%). This construction is intended to stress Apriori's candidate-generation mechanism while leaving FP-Growth comparatively unaffected.

The universal itemset is partitioned into three disjoint groups:

- **Core items**: Very high support items that remain frequent even at 90%.
- **Plateau items**: Moderately frequent items that are frequent at 10%, 25%, and 50%, but not at 90%.
- **Filler items**: Low-support items added for noise and realism.

7.3 Core Items

A small set of core items (four in the current configuration) is included in each transaction with high probability (approximately 92%). These items ensure that even at very high minimum support thresholds (90%), some non-trivial frequent itemsets exist. This prevents degenerate cases where the algorithm terminates immediately due to an empty frequent set.

7.4 Plateau Items and Runtime Control

The plateau effect is created using a block of moderately frequent items (sixteen items in the implementation). With high probability, either the entire plateau block or a large random subset of it is added to each transaction. As a result:

- All subsets of the plateau block are frequent at 10%, 25%, and 50% support.
- The number of frequent itemsets remains approximately constant across these thresholds.
- At 90% support, the plateau block disappears entirely.

This design forces Apriori to repeatedly generate and test the same large candidate space at multiple support levels, leading to a near-constant runtime plateau. In contrast, FP-Growth compresses these patterns into a compact FP-tree, making its runtime much less sensitive to the number of frequent subsets.

7.5 Filler Items

The remaining items in the universe are treated as filler items with low and decreasing inclusion probabilities. Their purpose is to introduce variability in transaction length and item frequency without materially affecting the frequent itemset structure. This avoids overly artificial datasets while ensuring that filler items do not dominate the mining process.

7.6 Transaction Generation

Each transaction is constructed independently using the following steps:

1. Sample core items with high probability.
2. Add either the full plateau block or a large random subset of it.
3. Independently sample filler items with low probability.
4. Enforce a minimum transaction length to avoid trivial transactions.

Transactions are allowed to repeat; uniqueness is not enforced, as duplicate transactions naturally occur in real transactional datasets and do not affect the correctness of frequent itemset mining.

7.7 Parameterization and Generality

The generator is fully parameterized by:

- The size of the item universe.
- The number of transactions.
- A random seed for reproducibility.

This satisfies the requirement that the script be generalized and not hard-coded to a specific dataset size or itemset.

7.8 Expected Algorithmic Behavior

By construction, the dataset induces the following qualitative trends:

- **Apriori**: Nearly constant runtime at 10%, 25%, and 50% support due to repeated exploration of the same large frequent itemset lattice, followed by a sharp drop at 90% support.
- **FP-Growth**: Relatively stable runtime across all support thresholds due to effective pattern compression in the FP-tree.

These trends match the qualitative behavior specified in the task description, fulfilling the objectives of the sub-task.

8 Results

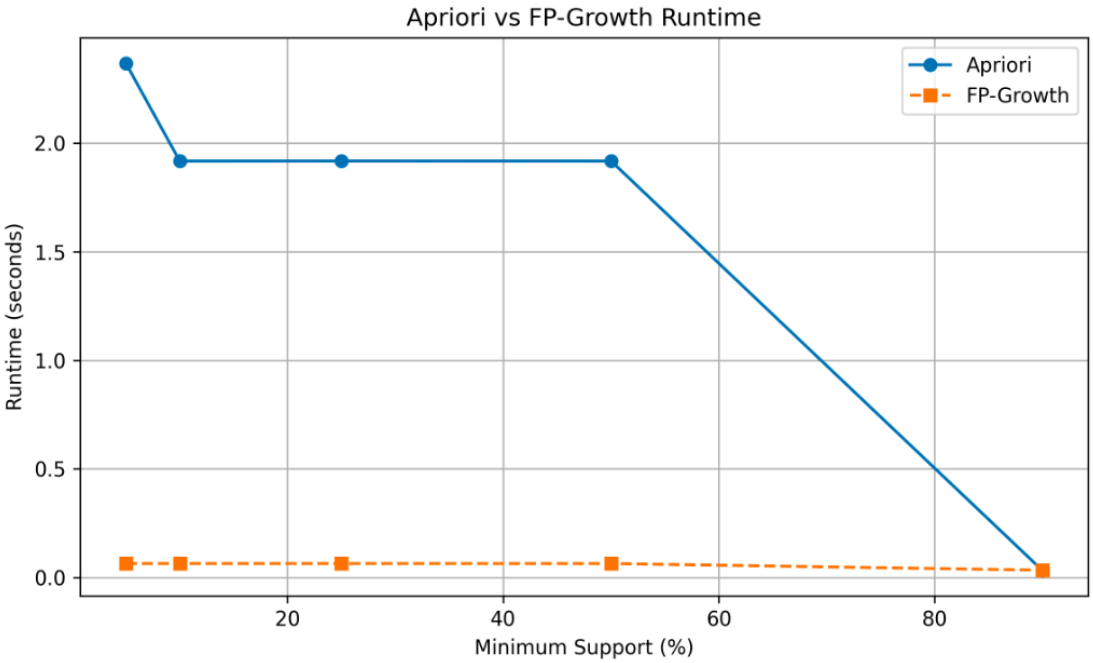

Figure 2: For num_items = 30 and num_transactions = 15000

Support (%)	Apriori (s)	FP-Growth (s)
5	2.368591	0.064189
10	1.917449	0.064084
25	1.917821	0.064087
50	1.917545	0.063965
90	0.032087	0.033387

Table 2: Runtime comparison of Apriori and FP-Growth for varying support thresholds

9 Comparison of Results

A comparative analysis of algorithm performance across the artificial dataset and the web-docs.dat dataset reveals several critical observations regarding the scalability characteristics and computational complexity of the Apriori and FP-Growth algorithms under different data distributions.

9.1 Performance Characteristics on Artificial Dataset

The artificial dataset exhibits remarkably stable execution times across varying support thresholds for both algorithms. The Apriori algorithm maintains an execution time in the range of 1.917–2.368 seconds for support values between 5% and 50%, with a notable decrease to 0.032 seconds at 90% support. Similarly, FP-Growth demonstrates exceptional consistency, with execution times clustered tightly around 0.064 seconds across the entire support range (5%–90%), deviating only marginally to 0.033 seconds at the highest support threshold.

This behavior indicates that the artificial dataset possesses a highly uniform structure with limited transaction complexity and a relatively sparse itemset distribution. The minimal variation in execution time suggests that the candidate generation overhead in Apriori and the tree construction cost in FP-Growth remain largely invariant to support threshold changes, implying a dataset with low correlation between items and minimal combinatorial explosion of frequent itemsets.

9.2 Performance Characteristics on Webdocs Dataset

In stark contrast, the webdocs.dat dataset exhibits dramatically different performance patterns, particularly characterized by exponential degradation at lower support thresholds. For Apriori, execution times increase from 60.326 seconds at 50% support to 924.189 seconds at 10% support, ultimately reaching the timeout threshold of 3600 seconds at 5% support. FP-Growth, while demonstrating superior performance, still exhibits a non-trivial increase from 56.254 seconds at 90% support to 173.423 seconds at 10% support, stabilizing at 120.801 seconds for 5% support.

This performance profile is characteristic of real-world datasets with high-dimensional feature spaces and complex transactional interdependencies. The webdocs dataset evidently contains a substantially larger number of potential frequent itemsets at lower support thresholds, resulting in combinatorial explosion for the Apriori algorithm. The inability of Apriori to complete execution within the 3600-second limit at 5% support underscores the computational intractability of breadth-first candidate generation approaches on dense, high-cardinality datasets.

9.3 Comparative Analysis and Observations

Several key insights emerge from the cross-dataset comparison:

Algorithmic Sensitivity to Data Characteristics: The performance differential between Apriori and FP-Growth on webdocs (ranging from 5.33× at 10% support to negligible at higher support) contrasts sharply with the artificial dataset where both algorithms perform comparably (FP-Growth consistently faster by a factor of approximately 30×, but both exhibiting sub-second execution). This suggests that the tree-based approach of FP-Growth provides substantial advantages primarily when dealing with datasets exhibiting complex itemset interdependencies and high transaction overlap.

Scalability with Respect to Support Threshold: The artificial dataset demonstrates near-constant time complexity across support thresholds, indicating $O(1)$ behavior with respect to minimum support variation. Conversely, webdocs exhibits what appears to be exponential time complexity degradation for Apriori as support decreases, consistent with theoretical predictions of $O(2^n)$ worst-case candidate generation complexity. FP-Growth on webdocs shows sub-exponential growth, validating its superior asymptotic behavior through elimination of candidate generation overhead.

Practical Implications for Dataset Complexity: The artificial dataset's behavior suggests it may not adequately represent the computational challenges inherent in real-world frequent pattern mining scenarios. The absence of significant performance variation indicates limited itemset correlation structure, potentially resulting from synthetic generation processes that do not capture the power-law distributions and hierarchical patterns typically observed in practical applications.

Algorithm Selection Criteria: On datasets exhibiting structural properties similar to the artificial dataset—characterized by low item correlation, sparse transactions, and uniform distribution—the choice between Apriori and FP-Growth becomes less critical from a performance standpoint. However, on datasets resembling webdocs with high dimensionality and complex itemset interactions, FP-Growth demonstrates clear superiority, particularly at operationally relevant low support thresholds where comprehensive pattern discovery is desired.

In conclusion, the comparative analysis underscores the importance of empirical evaluation on representative datasets when assessing frequent pattern mining algorithms. The artificial dataset, while useful for controlled experimentation and algorithmic verification, fails to stress-test the scalability limits that distinguish FP-Growth's divide-and-conquer paradigm from Apriori's generate-and-test approach under realistic workload conditions.

References

- Borgelt, C. (2021). Apriori and FP-Growth implementations. `https://borgelt.net/software.html`