Q1: Performance Analysis of Frequent Itemset Algorithms

Team: Who Cares

February 10, 2025

1 Introduction

Frequent pattern mining is a fundamental task in data mining with applications in various domains, including market basket analysis, bioinformatics, and network analysis. Frequent Itemset Mining works by finding sets of items that frequently occur together in a transactional database.

This report presents a comparative performance analysis for frequent itemset mining algorithms: Apriori and FP-Growth algorithm using the Webdocs dataset. Our results demonstrate the significant performance advantages of FP-Growth over Apriori for low support thresholds.

2 Algorithms being compared

- **gSpan:** A depth-first search-based algorithm that uses a canonical labeling system (DFS Code) to avoid redundant subgraph exploration. It extends subgraphs by adding edges one at a time.
- **FSG (Fast SubGraph):** A breadth-first search-based algorithm that generates candidate subgraphs by joining smaller frequent subgraphs. It uses a canonical representation to avoid duplicates. Also known as PAFI.
- **Gaston:** An algorithm that combines aspects of both depth-first and breadth-first search. It efficiently mines frequent paths and trees before exploring more complex cyclic graphs.

3 Details of the Experiment

- **Dataset:** Webdocs dataset, available at http://fimi.uantwerpen.be/data/webdocs.dat.gz.
- **Libraries:** Apriori and FP-Growth.
- **Support Thresholds:** 5%, 10%, 25%, 50%, and 90%.
- **Environment:** using Python 3.10 and the matplotlib library for plotting.
- **Platform:** Baadal platform.
- **Metrics:** Runtime in seconds.

4 Results

Support Threshold (%)	Apriori Runtime (s)	FP-Growth Runtime (s)
5	3600.0	231.0
10	1153.0	208.0
25	89.0	81.0
50	81.0	78.0
90	87.0	80.0

Table 1: Runtime comparison of Apriori and FP-Growth on the Webdocs dataset.
*3600 may mean did not run in time

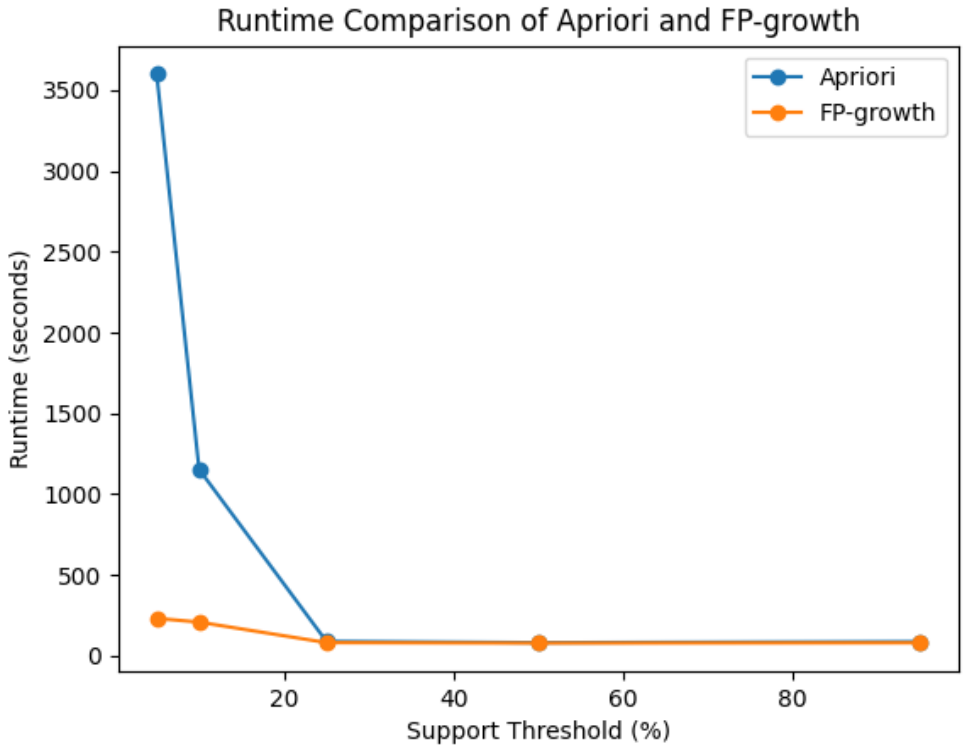

Figure 1: Runtime comparison of Apriori and FP-Growth.

5 Analysis and Discussion

Figure 1 and Table 1 clearly demonstrate the performance difference between Apriori and FP-Growth.

- **High Support ($\geq$ 25%):** At high support thresholds, the performance difference is relatively small. This is because the number of frequent itemsets is limited, and the overhead of candidate generation in Apriori is not as significant.
- **Low Support ($\geq$ 10%):** As the support threshold decreases, the performance gap widens dramatically. FP-Growth significantly outperforms Apriori. At 5% support, FP-Growth is over 15 times faster. This is due to the exponential increase in the number of candidate itemsets that Apriori must generate and test. FP-Growth, by contrast, avoids candidate generation entirely, relying on its compact FP-tree representation.
- **Analysis:** Apriori's "generate-and-test" approach becomes a major bottleneck at low support thresholds. Each iteration requires a full scan of the database to count the support of candidate itemsets. The number of candidate itemsets grows exponentially with decreasing support.
- **FP-Growth's Advantage: FP-Tree:** The FP-tree allows FP-Growth to compress the database into a tree structure, avoiding repeated database scans. Mining the FP-tree is significantly faster than generating and testing candidates.
- **Finding:** For frequent itemset mining, FP-Growth is significantly more efficient than Apriori, especially at low support thresholds, due to its avoidance of candidate generation.

6 References

- For code refinement, and grammatical helps ChatGPT, Perplexiy
- For latex formatting and refinement and research - Gemini pro Experimental 02-05