COL761 HW1: Frequent Itemset Mining Analysis

Arnav Raj [2022CS51652]
Arpit Agrawal [2022CS11612]
Ayush Gupta [2022CS11114]

February 10, 2025

Abstract

This report presents an empirical comparison of the Apriori and FP-tree algorithms for frequent itemset mining. We evaluate their performance on the `webdocs.dat` dataset (available at `http://fimi.uantwerpen.be/data/webdocs.dat.gz`) across various support thresholds (5%, 10%, 25%, 50%, and 90%). The study highlights the trade-offs between candidate generation and tree-based search in terms of scalability and computational efficiency.

1 Introduction

Frequent itemset mining is a core task in data mining with applications in market basket analysis, recommendation systems, and more. In this study, we compare two prominent algorithms:

- **Apriori Algorithm**: Iteratively generates candidate itemsets and prunes those that do not meet the minimum support threshold. Its performance is largely affected by the exponential growth in candidate sets at low supports.
- **FP-tree Algorithm**: Constructs a compact prefix-tree (FP-tree) to store itemset frequencies, thereby avoiding explicit candidate generation and reducing computational overhead.

We perform experiments using the `webdocs.dat` dataset at five support thresholds to observe how each algorithm scales with changes in data density.

2 Methodology

We automated the experiments with a shell script (`q1.sh`) that runs both algorithms with the following parameters:

- **Dataset**: `webdocs.dat` (preprocessed as needed for the algorithms).
- **Support Thresholds**: 5%, 10%, 25%, 50%, and 90%.
- **Execution**: The script collects runtime metrics and output files, which are then used to generate performance plots.

3 Results and Discussion

Figure 1 shows the execution time of both algorithms as a function of the minimum support threshold.

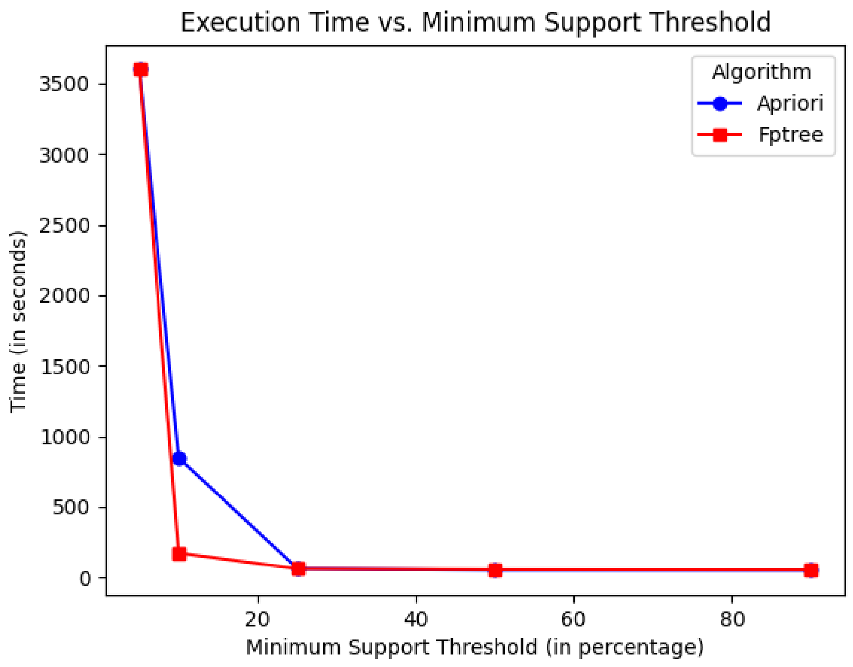

Figure 1: Execution Time vs. Support Threshold for Apriori and FP-tree Algorithms.

3.1 Analysis

Apriori Algorithm:

- At low support thresholds (5% and 10%), the runtime increases steeply due to the exponential growth in candidate itemset generation.
- With higher support thresholds, fewer candidates are generated, leading to more stable and reduced execution times.
- For threshold of 5% due to non-complete termination of code in time limit of 1 hour, Time required has been marked 3600.

FP-tree Algorithm:

- The FP-tree algorithm generally shows lower execution times because it avoids explicit candidate generation by compressing the dataset into a tree structure.
- At extremely low supports (5%), even FP-tree faces increased computational demands; however, it still outperforms Apriori. s
- For threshold of 5% due to non-complete termination of code in time limit of 1 hour, Time required has been marked 3600.

Overall Comparison:

- Both algorithms perform well at high support thresholds, but FP-tree is consistently faster across all thresholds.
- The study confirms that while Apriori is conceptually simple, its scalability is limited due to the combinatorial explosion of candidate sets, making FP-tree the more robust choice for dense datasets.

4 Conclusion

Our experimental analysis demonstrates that the FP-tree algorithm is more efficient than the Apriori algorithm for frequent itemset mining, especially at low support thresholds. These findings provide guidance on algorithm selection based on dataset characteristics and performance requirements.

5 References

1. Borgelt, C. (2005). *Frequent Pattern Mining.* Retrieved from `https://borgelt.net/doc/`
2. Han, J., Pei, J., & Yin, Y. (2000). *Mining Frequent Patterns without Candidate Generation.* In ACM SIGMOD Record.