Algorithms for Massive Data: Market-Basket Analysis on Amazon Book Reviews

Adiya Sapargali

December, 2025

Declaration of Authorship

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work, and including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Abstract

This project presents a scalable market-basket analysis applied to the Amazon Books Reviews dataset. Each user is treated as a basket containing the set of books they reviewed. The objective is to identify frequent items and frequent pairs of books that commonly appear together across users.

To handle the large size of the dataset, several efficiency techniques were employed, including data cleaning, global frequency pruning, and a sampling-based Apriori approach. The analysis focused on frequent single items and pairs, using support as the primary measure of frequency. A sensitivity analysis was conducted to study the effect of different support thresholds.

The results show that the proposed approach successfully identifies meaningful co-review patterns while remaining computationally efficient. Frequent pairs often correspond to books from the same series or different editions of the same title, reflecting realistic user behavior. The sampling strategy significantly reduces computation while preserving nearly all frequent itemsets, making the method suitable for large-scale market-basket analysis.

1 Introduction

This report describes the implementation of a market-basket analysis system using the Amazon Books Review dataset. The goal of the project is to find books that often appear together in the baskets of the same user, where each user represents a transaction and the books they reviewed represent the items.

Because the dataset is large, the project focuses on efficient methods for discovering frequent itemsets. The solution includes global frequency pruning, a manual implementation of the Apriori algorithm for single items and pairs, and a sampling-based approach to reduce the number of candidate itemsets that must be checked on the full dataset.

The report explains the dataset, preprocessing steps, the Apriori and sampling methods used, and the results obtained from the cleaned set of 66,082 user baskets.

2 Dataset

2.1 Dataset Choice and Origin

This study uses the `Books_rating.csv` file from the Amazon Books Review dataset, which is publicly available on Kaggle under the CC0 license. The dataset contains millions of book reviews submitted by users and includes book identifiers, titles, user identifiers, review scores, and related metadata. Its size and structure make it suitable for market-basket analysis, where each user can be modeled as a transaction and the set of books they reviewed as the corresponding basket of items.

2.2 Dataset Download and Authentication

The dataset was downloaded programmatically using the Kaggle API, ensuring reproducibility and alignment with modern data science workflows. API credentials were handled through environment variables during development. In the submitted version of this report, these credentials are replaced with placeholders for security reasons.

The following code was used to download and extract the dataset:

```
import os
os.environ['KAGGLE_USERNAME'] = "xxxxxxx"
os.environ['KAGGLE_KEY'] = "xxxxxxx"

!kaggle datasets download -d mohamedbakhet/amazon-books-reviews -p data/
!unzip -o data/amazon-books-reviews.zip -d data/
```

The dataset identifier used is `mohamedbakhet/amazon-books-reviews`.

2.3 Considered Dataset Parts

From the `Books_rating.csv` file, the following columns were selected for this market-basket analysis:

- **User_id**: Unique identifier for each user.
- **Id**: Unique identifier for each book (ASIN). This ensures consistent book identification across editions.
- **Title**: The book title, kept only for interpretation of results.

3 Data Organization and Pre-processing

The raw dataset required several pre-processing steps to transform it into a suitable format for market-basket analysis, where each "basket" represents the set of books reviewed by a single user.

3.1 Initial Data Loading and Cleaning

The data were loaded into a Pandas DataFrame from the file `Books_rating.csv`. Initial cleaning included the following steps:

- **Column Selection**: Retained only the fields `User_id`, `Id`, and `Title`, which are directly relevant to basket construction and later interpretation of results.
- **Handling Missing Values**: Removed rows where either `User_id` or `Id` was missing, since both are required for identifying user–item interactions.

- **Removing Duplicates**: Deleted duplicate entries where the same user reviewed the same book more than once, ensuring that each user–book pair appears only once in the dataset.

After these steps, 2,397,614 valid user–book interactions remained.

3.2 Basket Formation

User baskets were constructed by grouping book `Ids` by `User_id`. The basket of each user represents the collection of books they reviewed.

- **De-duplicating Within Baskets**: Ensured that each book appears only once within a user's basket by applying `sorted(set(lst))`.
- **Minimum Basket Size**: Users who reviewed fewer than two distinct books were removed, since at least two items are required to form item pairs.

This produced 308,676 initial baskets.

3.3 Global Item Frequency Pruning

To make the analysis faster and remove books that appear only a few times, a global frequency pruning step was applied before running the Apriori algorithm.

- **Minimum Item Support**: A threshold of `MIN_ITEM_SUPPORT = 0.0015` was chosen, which means that a book must appear in at least 0.15% of all user baskets.
- **Book Filtering**: Books not meeting this minimum frequency were removed from every user basket.
- **User Filtering**: Users whose baskets contained fewer than two books after this step were also excluded.

This reduced the dataset to 66,082 baskets containing 472 sufficiently frequent books, significantly improving performance and scalability for subsequent analysis.

3.4 Transaction Representation

Instead of one-hot encoding or sparse matrices, the cleaned baskets were directly converted into a list of Python `set` objects:

$$\text{transactions} = [\text{ set of book IDs per user }].$$

This representation is efficient for:

- checking whether an item is in a basket,
- counting how often items appear together,
- running the Apriori steps for 1- and 2-itemsets,

and avoids the need to build large sparse matrices, which are not required for this analysis. It also works well with the sampling-based Apriori method used later in the project.

4 Apriori Algorithm for Frequent Itemsets

The Apriori algorithm is a classical method for finding frequently co-occurring items in transaction datasets. In this project, each transaction corresponds to a user basket and contains the set of book identifiers reviewed by that user. The algorithm relies on the principle that if an itemset is frequent, then all of its subsets must also be frequent. This property allows the search space to be reduced significantly.

4.1 Support Definition

For any itemset X, its support is defined as the proportion of transactions in which all the items in X appear:

$$\text{support}(X) = \frac{|\{T \in \mathcal{T} : X \subseteq T\}|}{|\mathcal{T}|},$$

where $\mathcal{T}$ is the set of all user baskets. An itemset is considered frequent if its support exceeds the minimum support threshold. In this project, the chosen value was:

$$\text{MIN\_SUPPORT} = 0.008,$$

meaning that an item or item pair must appear in at least 0.8% of all baskets.

4.2 Frequent 1-Itemsets

To compute the frequent single items, the algorithm counts how many user baskets contain each book. Items whose support meets or exceeds the threshold form the set of frequent 1-itemsets L_1. These counts are efficient to compute directly from the transaction list.

The result of this step serves two purposes:

- it identifies the most commonly reviewed books,
- it restricts the candidate pairs considered in the next step.

4.3 Frequent 2-Itemsets

The second pass of Apriori focuses on item pairs. Thanks to the monotonicity property, the algorithm only needs to evaluate pairs composed of items already present in L_1. For each transaction, all 2-item combinations of frequent items are generated, and their occurrence counts are updated. A pair $\{i, j\}$ is considered frequent if:

$$\text{support}(\{i, j\}) \geq \text{MIN\_SUPPORT}.$$

This produces the set of frequent 2-itemsets L_2, which reveals which books tend to be reviewed together by many users.

4.4 Motivation for Limiting Analysis to 1- and 2-Itemsets

Because the dataset is large, mining itemsets of size three or more would be much slower and require more memory. In practice, frequent pairs are already enough to understand which books are commonly reviewed together. This also fits well with the sampling approach used later in the project, which is designed for 1- and 2-itemset mining.

5 Sampling-Based Apriori

Because the dataset contains tens of thousands of baskets, running Apriori on the full data can still be slow. To make the process more efficient, a sampling step was added before counting frequent itemsets.

5.1 Random Sampling of Transactions

A simple random sample of baskets was taken, where each basket is included with probability $p = 0.3$. This means that roughly 30% of all baskets are used to generate the initial candidate itemsets. Sampling reduces the amount of data that need to be scanned while keeping the main patterns present in the dataset.

Let $\mathcal{T}$ denote the full set of baskets, and let $\mathcal{S} \subseteq \mathcal{T}$ be subset of samples. The Apriori algorithm is first applied to $\mathcal{S}$ to produce:

- the items that are frequent in the sample,
- the item pairs that appear often enough in the sample to be considered candidates.

These candidates are then checked against the full dataset.

5.2 Candidate Generation on the Sample

The Apriori procedure on the sample identifies items and pairs that are frequent *within the sample*. These itemsets are treated as candidates for the full dataset. Because sampling can slightly increase or decrease item frequencies, not every candidate will be truly frequent when evaluated on all baskets.

However, sampling greatly reduces the number of pairs that need to be checked. Instead of examining all combinations of frequent items, the algorithm only checks the pairs that appeared often in the sample.

5.3 Validation on the Full Dataset

In the final step, each candidate itemset is counted again using the full set of transactions. An item or pair is accepted as frequent only if its support on the full dataset meets the minimum support threshold.

This two-phase approach has two advantages:

- it avoids computing all pair frequencies directly on the full dataset,
- it keeps the exact same final results as full Apriori (except for a very small number of rejected candidates).

In this project, nearly all candidates found in the sample were also frequent in the full dataset: 357 out of 362 items and 1302 out of 1304 item pairs. This shows that sampling kept the important patterns and saved a significant amount of computation.

6 Experiments and Results

This section presents the experimental results obtained from applying the frequency–based Apriori algorithm with sampling to the cleaned dataset of 66,082 user baskets.

6.1 Experimental Setup

All experiments were performed on the final dataset after preprocessing and global frequency pruning. The minimum item support `MIN_ITEM_SUPPORT = 0.0015` removed books that appeared in fewer than 0.15% of baskets, reducing the number of unique items to 472.

For the Apriori steps, the minimum support threshold was set to `MIN_SUPPORT = 0.008`, meaning that an item or item pair must appear in at least 0.8% of all baskets to be considered frequent. The algorithm was restricted to single items and pairs (L1 and L2), which keeps the results interpretable and computationally efficient.

A random sample containing 30% of all baskets was used to generate candidate frequent itemsets. These candidates were then validated against the full dataset, reducing the total number of pairwise checks required.

6.2 Data Distribution Visualizations

6.2.1 Distribution of Books Reviewed per User

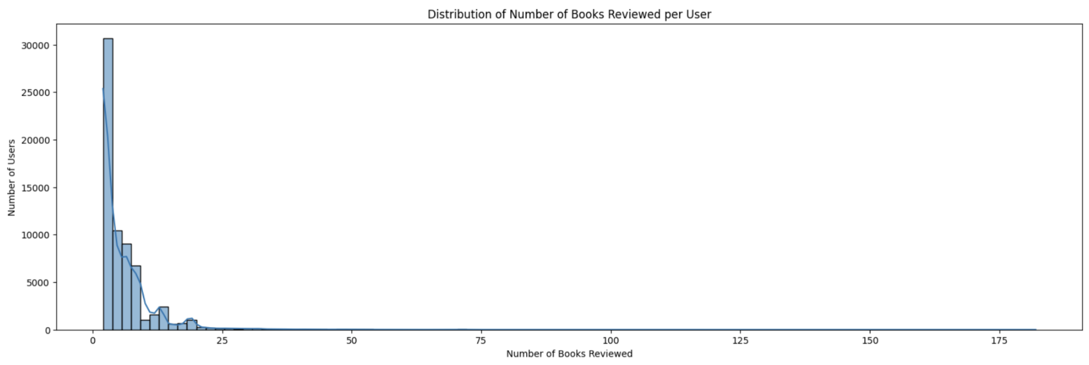

Figure 1: Distribution of number of books reviewed per user

The histogram of basket sizes shows a highly skewed distribution: most users reviewed only a few books, while a small number reviewed dozens. This long–tailed behavior is common in review datasets and supports the decision to remove users with fewer than two reviews.

6.2.2 Top 20 Most Reviewed Books

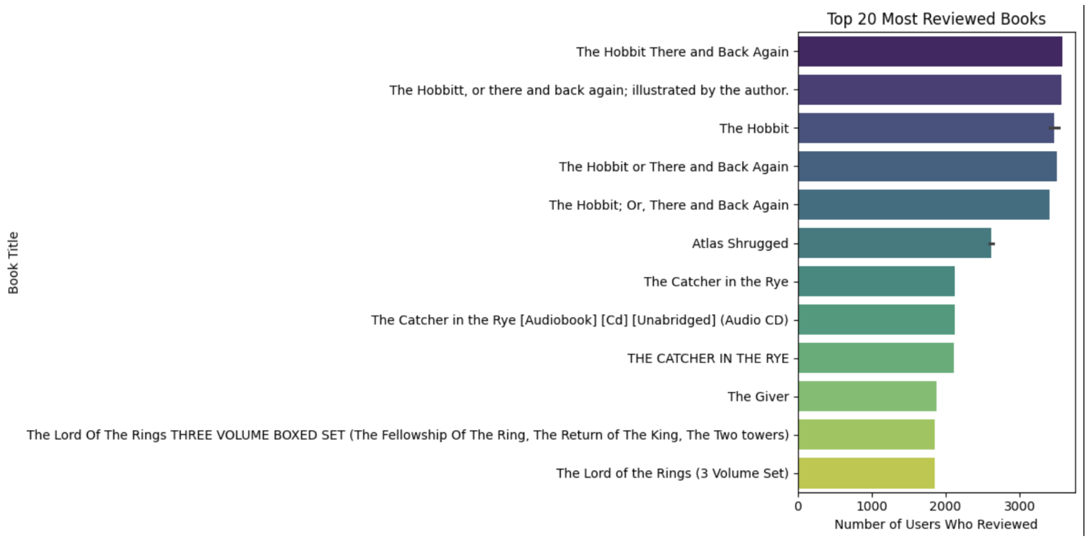

Figure 2: Top 20 most reviewed books

The bar chart of the twenty most reviewed books highlights several editions of *The Hobbit*, along with other popular titles such as *Atlas Shrugged* and *The Catcher in the Rye*. These books naturally appear often in the frequent itemset results.

6.3 Frequent Items (L1)

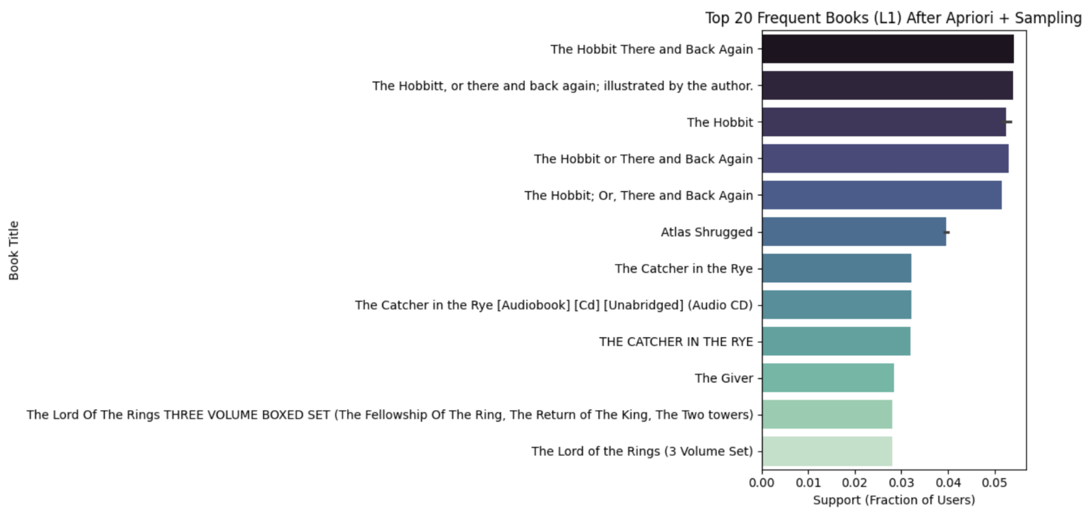

Figure 3: Top frequent books after Apriori

Using `MIN_SUPPORT = 0.008`, the algorithm identified 357 frequent single items. The plot of the top twenty frequent items shows that the results closely follow the distribution of the most reviewed books. Popular titles appear in a large fraction of the baskets, which explains their presence in L1.

6.4 Frequent Pairs (L2)

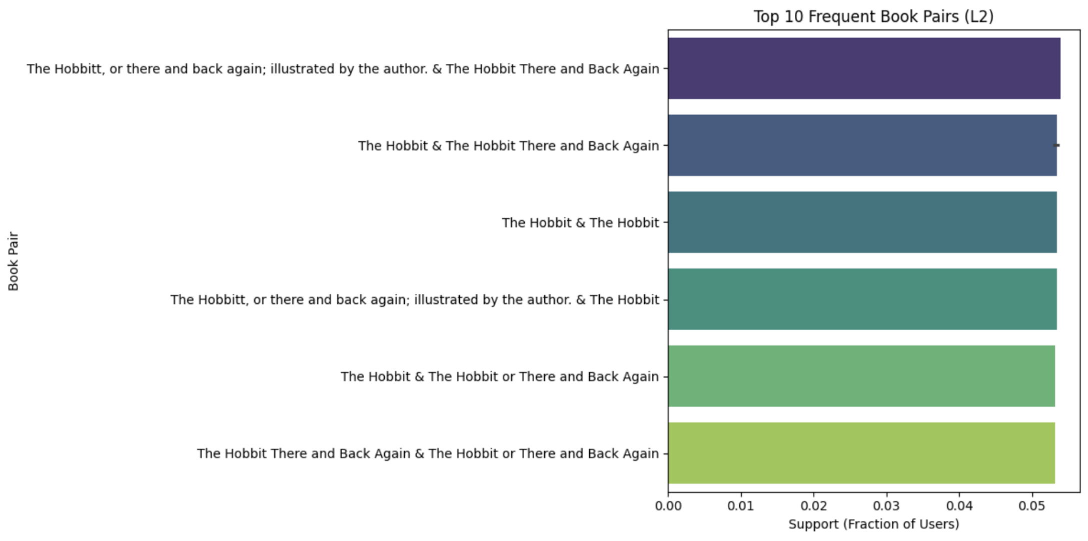

Figure 4: Top frequent book pairs

The algorithm found 1302 frequent item pairs. The strongest pairs consist of different editions or related versions of *The Hobbit*, indicating that users commonly review multiple versions of the same book. These pairs reflect clear co–review patterns and demonstrate the effectiveness of the Apriori method in detecting meaningful associations.

6.5 Sampling Results

The sampling stage used 19,746 baskets (30% of the dataset). From this sample, the algorithm produced:

- 362 candidate frequent items,
- 1304 candidate frequent pairs.

When validated on the full dataset, 357 items and 1302 pairs were confirmed as truly frequent. This near–perfect match shows that sampling preserved the important patterns while reducing the computational cost.

6.6 Effect of Support Threshold

To study the sensitivity of the algorithm, three support thresholds were tested: 0.005, 0.008, and 0.01.

- At `MIN_SUPPORT = 0.005`: 447 frequent items and 1486 frequent pairs.
- At `MIN_SUPPORT = 0.008`: 357 frequent items and 1302 frequent pairs.
- At `MIN_SUPPORT = 0.01`: 219 frequent items and 672 frequent pairs.

Lower thresholds produce more frequent itemsets but may introduce weaker or less meaningful patterns. Higher thresholds reduce noise but may discard useful associations. The value `MIN_SUPPORT = 0.008` provides a good balance.

Additionally, pruning thresholds between 0.1% and 0.3% were tested. These values did not change the qualitative results but helped reduce computation by removing very rare books before analysis.

6.7 Summary of Findings

The experiments show that the Apriori algorithm with sampling efficiently identifies meaningful co–review relationships in a large dataset. The frequent items and pairs closely match known reading patterns, especially among widely reviewed titles. The sampling step reduces computation with almost no loss of accuracy, and the chosen support threshold yields a clear and interpretable set of patterns.

7 Discussion and Conclusions

7.1 Scalability Considerations

The dataset used in this project is large, containing tens of thousands of user baskets and millions of original reviews. To ensure scalability, several design choices were made. Global frequency pruning removed very rare books early, significantly reducing the number of items considered. Limiting the analysis to frequent single items and pairs avoided the exponential growth associated with larger itemsets.

In addition, a sampling-based Apriori approach was applied. By generating candidates on a random subset of the data and validating them on the full dataset, the number of pairwise checks was greatly reduced. The experiments showed that sampling preserved almost all frequent patterns while lowering computational cost. Together, these choices make the approach efficient and suitable for large-scale datasets.

7.2 Discussion of Results

The frequent itemsets discovered by the algorithm reflect clear and intuitive co-review patterns. Popular books appear frequently as single items, while frequent pairs often consist of different editions or related books from the same series or author. These patterns are consistent with real user behavior and confirm that the Apriori algorithm effectively captures meaningful relationships between items.

The comparison of different support thresholds showed that lower thresholds produce more itemsets but introduce weaker patterns, while higher thresholds reduce noise at the cost of coverage. The chosen value of `MIN_SUPPORT = 0.008` provided a good balance between interpretability and pattern richness.

7.3 Limitations and Future Work

This project focused on frequent itemsets of size one and two. While this choice improves scalability and interpretability, larger itemsets could reveal more complex reading patterns. Future work could extend the analysis to higher-order itemsets or incorporate association rule metrics such as confidence and lift. Another possible extension is to analyze textual baskets derived from review content, as suggested in the project description.

7.4 Conclusion

Overall, this project demonstrates that a carefully designed Apriori-based approach, combined with pruning and sampling, can efficiently mine frequent itemsets from large real-world datasets. The results are both meaningful and scalable, making the method suitable for practical market-basket analysis tasks.

References

[1] Mohamed Bakhet. *Amazon Books Reviews Dataset*. Kaggle, 2020. Available at: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews