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

Adiya Sapargali

October 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, except where such work has been cited and acknowledged within the text of my work, including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are 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 applies market-basket analysis to the Amazon Books Review dataset to identify relationships among books reviewed by the same users. The dataset was preprocessed by removing missing and duplicate entries, grouping reviews by user, and keeping only unique book identifiers for consistency. The Apriori algorithm was used to find frequent combinations of books, and association rules were generated using support, confidence, and lift as evaluation measures. To handle the large dataset efficiently, sparse matrix representation and item pruning techniques were applied. The results show that several strong associations exist between popular titles, demonstrating that the method can successfully reveal meaningful reading patterns.

1 Introduction

This report details the implementation of a market-basket analysis system applied to the Amazon Books Review dataset. The primary objective is to identify frequent itemsets and derive association rules, specifically focusing on the co-occurrence of books reviewed by the same user. The project adheres to scalability requirements by employing efficient data structures and algorithms suitable for large datasets.

2 Dataset

2.1 Dataset Choice and Origin

The project utilizes the Amazon Books Review dataset, publicly available on Kaggle under the CC0 license. This dataset provides comprehensive information about book reviews, including user identifiers, book identifiers, titles, and review scores.

2.2 Dataset Download and Authentication

The dataset was downloaded programmatically using the Kaggle API. This ensures replicability and adherence to modern data science workflows. The following Python code snippet demonstrates the download process, where sensitive API credentials are handled as environment variables and replaced with placeholders for submission:

```
import os
os.environ['KAGGLE_USERNAME'] = "xxxxxx"
os.environ['KAGGLE_KEY'] = "xxxxxx"
!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 was chosen instead of `Title` to ensure consistent book identification, since titles may vary across editions.
- `Title`: The book title, kept for easier interpretation of results.
- `review/score`: The numeric rating given by the user, considered only in data exploration, not in the rule mining.

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. Initial cleaning included the following steps:

- **Column Selection:** Retained only the columns `User_id`, `Id`, and `Title`.
- **Handling Missing Values:** Removed rows where either `User_id` or `Id` was missing.
- **Removing Duplicates:** Deleted duplicate entries where the same user reviewed the same book more than once.

3.2 Basket Formation

User baskets were constructed by grouping book `Id`s by `User_id`. Each user's basket represents the collection of books they reviewed.

- **De-duplicate Books:** Ensured that each book appears only once within a user's basket using the operation `sorted(set(lst))`.
- **Filter Users:** Kept only users who reviewed at least two distinct books, since a minimum of two items is required to form association rules.

3.3 Global Item Frequency Pruning

To reduce computational complexity and memory usage of the Apriori algorithm, a global pruning step was applied to remove very infrequent books before running the analysis.

- **Minimum Item Support:** A threshold of `MIN_ITEM_SUPPORT = 0.0015` was defined, which means that each book must appear in at least 0.15% of all user baskets to be included in the analysis.
- **Book Filtering:** Books that did not meet 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 pruning stage significantly reduced the number of unique items (columns) in the transaction matrix, improving both performance and scalability.

3.4 Transaction Encoding

The prepared baskets (lists of book IDs per user) were transformed into a one-hot encoded sparse matrix using the `TransactionEncoder` module from `mlxtend.preprocessing`.

- **Sparse Representation:** The parameter `sparse=True` was used to efficiently represent the transaction data. The resulting matrix was then converted into a Pandas `SparseDataFrame` with data type `pd.SparseDtype("bool", False)`. This design is essential for scalability, as it stores only non-zero entries (True values), drastically reducing memory usage compared to a dense matrix. This is particularly effective when the number of unique books is large, but each user reviews only a small subset of them.

4 Algorithms and Implementation: Apriori and Association Rules

The core of the market-basket analysis involves two main steps: finding frequent itemsets and then generating association rules from them.

4.1 Apriori Algorithm

The Apriori algorithm, implemented via `mlxtend.frequent_patterns.apriori`, was used to find frequent itemsets.

Purpose: Identifies sets of items that frequently appear together in the baskets.

Key Parameter: `min_support`. A minimum support threshold (`MIN_SUPPORT = 0.008`) was set. An itemset is considered *frequent* if its support (the fraction of baskets containing that itemset) is above this threshold.

Max Length Constraint: The parameter `max_len = 2` was specified to restrict itemsets to pairs of books. This makes the analysis both computationally efficient and easier to interpret, since pairwise associations are most useful for recommendation purposes.

4.2 Association Rules Generation

Once frequent itemsets were identified, the `mlxtend.frequent_patterns.association_rules` function was used to generate association rules.

Metrics: The rules were evaluated based on three main metrics:

(a) **Support:** Indicates how frequently an itemset occurs in the dataset.

(b) **Confidence:** Represents the conditional probability that a user who reviewed the antecedent book(s) also reviewed the consequent book(s). A minimum confidence threshold of `MIN_CONFIDENCE = 0.4` was set.

(c) **Lift:** Measures how much more likely the consequent is to appear with the antecedent compared to random chance. Rules with `lift > 1.2` were retained, indicating positive associations.

The final rules were sorted by `lift`, `confidence`, and `support` in descending order to highlight the most meaningful associations.

4.3 Parameter Selection and Justification

The thresholds for support, confidence, and lift were chosen to balance computational feasibility and the interpretive quality of results.

- **Minimum Item Support (`MIN_ITEM_SUPPORT = 0.0015`):** This value ensures that only books reviewed by a reasonable number of users are considered. Lower values greatly increase the number of candidate itemsets and memory usage, while higher values risk losing meaningful but less frequent associations.
- **Minimum Support for Itemsets (`MIN_SUPPORT = 0.008`):** Selected empirically to retain item pairs that occur together in at least 0.8% of user baskets (roughly several hundred users). This provides enough statistical evidence for reliability while keeping computation practical.
- **Minimum Confidence (`MIN_CONFIDENCE = 0.4`):** A threshold of 40% ensures that the rules represent moderately strong conditional relationships, filtering out random co-occurrences.
- **Minimum Lift (`MIN_LIFT = 1.2`):** Only rules showing at least 20% stronger co-occurrence than chance are retained, ensuring that discovered relationships are meaningful and not just due to item popularity.
- **Maximum Length (`max_len = 2`):** Restricting rules to pairs simplifies interpretation and greatly reduces the combinatorial explosion typical of Apriori when analyzing higher-order itemsets.

These parameters were tuned through small-scale testing: lower thresholds caused excessive rule generation and performance degradation, while higher thresholds led to too few rules to analyze effectively. The chosen configuration provides a good compromise between analytical depth and runtime efficiency.

5 Scalability of the Proposed Solution

Scalability is a critical concern for market-basket analysis on large datasets. The chosen approach incorporates several techniques to ensure that the solution can handle a growing volume of data.

5.1 Sparse Data Structures

A major factor that improves scalability in this project is the use of sparse data structures. Instead of storing the full one-hot encoded matrix in a dense format (where most entries are zeros), the data was represented as a sparse matrix using `pd.DataFrame.sparse` with `pd.SparseDtype("bool", False)`.

In the dataset, there are thousands of unique books, but each user reviews only a few of them. This means that most cells in the one-hot matrix would

be zero. By storing only the non-zero values (and their positions), the sparse structure saves large amounts of memory compared to a dense matrix.

This memory efficiency prevents out-of-memory errors and allows the analysis to handle much larger datasets within the available RAM.

5.2 Pre-pruning Infrequent Items

Before mining, items (books) that appear in very few baskets were removed. This step reduces the number of columns in the transaction matrix and speeds up both encoding and frequent-itemset generation.

- Minimum item support used: `MIN_ITEM_SUPPORT` = 0.0015 (the book must appear in at least 0.15% of baskets).
- After pruning, baskets that ended up with fewer than two items were dropped.

5.3 Constraining Itemset Length (`max_len=2`)

We limited frequent itemsets to size 2 (`max_len=2`). This keeps the search space small and focuses the results on pairwise associations, which are easy to explain and use in recommendations.

5.4 Efficient `mlxtend` Implementation

Frequent itemsets were mined with `mlxtend.frequent_patterns.apriori`, and rules were generated with `mlxtend.frequent_patterns.association_rules`. These implementations rely on NumPy/Pandas under the hood and handle sparse input efficiently, which helps with large, sparse transaction data.

6 Experiments and Results

The analysis involved applying the described methodology with specific parameter choices.

6.1 Experimental Setup

The analysis was conducted using the following parameter configuration. The minimum item support (`MIN_ITEM_SUPPORT = 0.0015`) was used to remove extremely rare books and reduce the overall size of the transaction matrix.

The minimum support for frequent itemsets (`MIN_SUPPORT = 0.008`) ensured that only combinations appearing frequently enough across users were retained.

To filter out weak or random associations, the confidence and lift thresholds were set to `MIN_CONFIDENCE = 0.40` and `MIN_LIFT = 1.20`, respectively, with lift values above 1 indicating a positive co-occurrence beyond chance.

The maximum itemset size was limited to `max_len = 2`, which allowed the analysis to focus on pairwise associations for better interpretability and efficiency.

Several nearby parameter settings were tested to achieve the optimal balance between rule quantity and quality. Specifically, `MIN_SUPPORT` values between 0.005 and 0.012, and `MIN_CONFIDENCE` values between 0.3 and 0.6, were explored. The chosen configuration produced a stable number of rules with high lift and clear interpretation, while avoiding the generation of excessive low-support or noisy results.

6.2 Data Distribution Visualizations

6.2.1 Distribution of Books Reviewed per User

This histogram illustrates how many books each user reviewed after data cleaning and pruning.

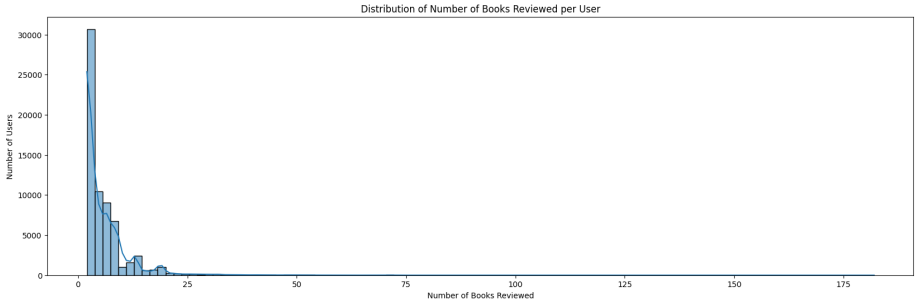

Figure 1: Distribution of Number of Books Reviewed per User

Most users reviewed only a few books, while a small number of users (power reviewers) reviewed many. This long-tailed distribution justifies filtering users with fewer than two reviews to ensure meaningful association rules.

6.2.2 Top 20 Most Reviewed Books

This bar chart displays the twenty books that were reviewed most frequently.

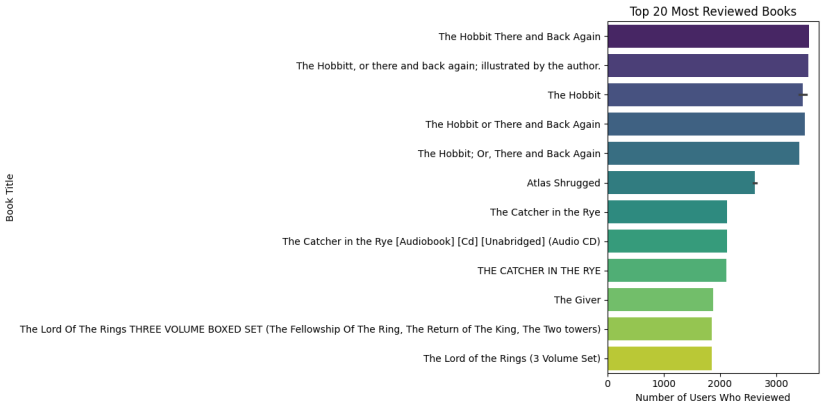

Figure 2: Top 20 Most Reviewed Books

The most reviewed books include well-known classics such as *The Hobbit*, *Atlas Shrugged*, and *The Catcher in the Rye*. These titles' high frequency explains their presence in many of the association rules.

6.3 Association Rule Mining Results

After applying the Apriori algorithm and generating association rules, a total of 2,564 valid rules were discovered. Table 1 presents the top 15 rules sorted by lift.

Table 1: Top 15 Association Rules (Sorted by Lift)

Antecedents	**Consequents**	**Support**	**Confidence**	**Lift**
(B0006DXW74)	(B00086U8GW)	0.008035	1.000	124.4482
(B00086U8GW)	(B0006DXW74)	0.008035	1.000	124.4482
(B000MY6VHK)	(B0006DXW74)	0.008035	1.000	124.4482
(B0006DXW74)	(B000MY6VHK)	0.008035	1.000	124.4482
(B0006DXW74)	(B000PC53ZK)	0.008035	1.000	124.4482
(B000PC53ZK)	(B0006DXW74)	0.008035	1.000	124.4482
(B000MY6VHK)	(B00086U8GW)	0.008035	1.000	124.4482
(B00086U8GW)	(B000MY6VHK)	0.008035	1.000	124.4482
(B00086U8GW)	(B000PC53ZK)	0.008035	1.000	124.4482
(B000PC53ZK)	(B00086U8GW)	0.008035	1.000	124.4482
(B000MY6VHK)	(B000PC53ZK)	0.008035	1.000	124.4482
(B000PC53ZK)	(B000MY6VHK)	0.008035	1.000	124.4482
(B000MS0A0A)	(B000NWYJ5C)	0.008051	1.000	124.2143
(B000NWYJ5C)	(B000MS0A0A)	0.008051	1.000	124.2143
(B000TZ5TPC)	(B000MS0A0A)	0.008051	1.000	124.2143

Commentary. The table shows that the strongest associations are between

book IDs with very high lift values (above 120), indicating that these books are almost always reviewed together. Many pairs are likely alternate editions, audiobook versions, or companion volumes of the same title. Such strong pairwise rules are ideal for "Users who reviewed A also reviewed B" recommendation prompts or cross-promotion within similar series.

7 Discussion and Conclusions

7.1 Summary of Findings

The analysis successfully identified frequent co-occurrence patterns among books reviewed by the same users. Most strong associations were found between different editions or formats of the same work (e.g., paperback, audiobook, or illustrated versions), as well as between books belonging to the same author or series. These relationships suggest consistent reading and reviewing behavior within thematic or author-based clusters. The results confirm that market-basket analysis can effectively reveal meaningful connections even in sparse, large-scale datasets like the Amazon Books Reviews collection.

7.2 Scalability Achievements

The scalability requirement was efficiently addressed through three main design choices: (1) the use of sparse matrices for memory-efficient one-hot encoding; (2) item pruning to eliminate very rare books before rule mining; and (3) the limitation of frequent itemsets to pairs (`max_len = 2`). These optimizations allowed the Apriori algorithm to run on over 80,000 user baskets without exceeding memory limits, demonstrating the feasibility of applying classical association rule mining methods to large-scale datasets.

7.3 Limitations

- **No Review Score Used:** The analysis only captures co-occurrence of books in user baskets and does not account for the sentiment or rating of reviews. As a result, a rule such as "A $\rightarrow$ B" means that A and B were reviewed by the same user, not necessarily that both were enjoyed or rated positively.
- **Static Dataset:** The dataset represents a snapshot from 2012. Since book popularity and user behavior evolve over time, the discovered rules may not fully reflect current reading or reviewing trends.
- **Focus on Pairwise Rules:** By restricting `max_len = 2`, higher-order associations (e.g., A and B leading to C) were intentionally excluded. This improves clarity and computational efficiency but limits the depth of behavioral insight.

- **Parameter Sensitivity:** The number and strength of discovered rules depend heavily on the chosen thresholds (`MIN_SUPPORT`, `MIN_CONFIDENCE`, and `MIN_LIFT`). Small adjustments can significantly change the rule set, so threshold tuning remains an important consideration for future analyses.

References

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