Project of Algorithms for Massive Data: Market-basket analysis

ELAHEH ZOHDI

13731A

ACADEMIC YEAR 2024-2025

1. Abstract

In this report, I present a project on Market-Basket Analysis, a technique aimed at identifying frequent itemsets within large transactional datasets. In the context of this work, I focus on discovering commonly co-reviewed books by users of the Amazon Books Reviews dataset. Each user's basket consists of the set of book titles they have reviewed. The underlying assumption is that the dataset is too large to fit into memory, thus motivating the use of PySpark to handle distributed data efficiently.

The goal is to implement a detector of frequent itemsets using the A-Priori algorithm, which proceeds iteratively to build larger itemsets from frequent smaller ones. This approach allows a reduction in computational complexity thanks to the monotonicity property of itemsets. To manage the volume of the data while ensuring reproducibility and reasonable runtime, I apply a 1% sampling of the original dataset. The data is then cleaned and normalized before analysis.

The entire pipeline from data extraction and preprocessing to frequent itemset detection is written in Python and executed with PySpark in a Google Colab environment. No additional libraries were used apart from PySpark and itertools. The full code is organized in a way that it can be downloaded and executed from beginning to end in under 15 minutes. Only minimal user interaction (inserting Kaggle credentials) is required.

"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."

2. The dataset

The dataset used in this project is the *Amazon Books Reviews* dataset, publicly available via Kaggle and released under the *CC0: Public Domain* license. It contains a large volume of book reviews written by users on the Amazon platform. The dataset was downloaded directly from the Kaggle API and automatically extracted within the notebook.

The CSV file includes multiple columns related to review metadata, such as the review text, the rating, the timestamp, and user identifiers. For the purpose of this project, only two columns are considered:

- User_id: the anonymized identifier of the user who wrote the review.
- Title: the title of the book reviewed.

These two columns allow us to construct baskets, where each *user* corresponds to one basket, and the *books they reviewed* form the items of that basket. Any row with a missing value in either of the two selected columns is discarded. To keep the analysis computationally

efficient while preserving a meaningful sample of user behavior, I extract a *1% random sample* of the cleaned data, seeded for reproducibility.

The final working sample consists of *1,271 baskets*, where each basket includes at least two reviewed book titles. This filtering ensures that only relevant user interactions are considered for association mining. The choice to sample the data — while not strictly required — is justified by runtime constraints and the objective of keeping the entire pipeline within an interpretable and reproducible scope for the report.

3. Data pre-processing

The aim of the pre-processing phase is to transform the raw data into a format that can be used to efficiently apply the A-Priori algorithm. As previously mentioned, the two selected columns are `User_id` and `Title`. After dropping any rows containing nulls, the dataset is sampled (1%) and filtered to retain only users who have reviewed more than one book, thereby ensuring that every resulting basket contains at least two items.

The titles of the books are string-based and may contain inconsistencies such as extra whitespace, punctuation, or mixed casing. To address this, I normalize the strings by converting all characters to lowercase, removing all non-alphanumeric characters (except spaces) and normalizing whitespace and trimming leading/trailing spaces.

After cleaning, I group the dataset by `User_id`, aggregating all book titles reviewed by each user into a set. This produces a structure where each row is a basket: a user ID paired with a set of book titles. I also compute basic statistics on the resulting baskets:

- Number of baskets: 1,271
- Maximum basket size: 56
- Minimum basket size: 2
- Average basket size: ~2.81

These figures confirm that while most users reviewed only a small number of books, there are outliers with a significantly higher count. At this stage, the baskets are stored as a native Python list of lists.

4. The algorithm

The core of this project is the implementation of the **A-Priori algorithm** to identify frequent itemsets within the dataset. The algorithm is applied to the previously constructed baskets, each representing a set of books reviewed by a single user. To improve space and computational efficiency, a hashing step is introduced prior to running the algorithm.

Each unique book title is first mapped to a unique integer using a hash table. This transformation allows the baskets to be converted from sets of strings into sets of integers. For example, a basket containing ['the hobbit', 'the fellowship of the ring'] becomes {1071, 314}.

where `1071` and `314` represent the hashed values of the titles. The inverse mapping is preserved so that results can be translated back to their original readable form.

The A-Priori algorithm is implemented as a series of passes:

Pass 1 (Singletons):
Every item is mapped to a (key, value) pair with the item as the key and 1 as the value. A reduction step sums the counts for each item. All items that occur more than or equal to the support threshold are retained as frequent singletons. These are then stored in a set for reference in the next passes.

Pass k (k > 1):
For each basket, all possible k-item combinations are generated. However, only those combinations are kept where every (k-1)-subset is also frequent. This optimization is based on the monotonicity property of frequent itemsets: if an itemset is frequent, then all of its subsets must also be frequent. The remaining candidate itemsets are again counted and filtered by the support threshold.

This process continues iteratively, incrementing k, until no more frequent itemsets are found. Sorting the basket contents prior to generating combinations ensures that tuples like (A, B) and (B, A) are treated as equivalent.

In my experiment, the support was set to **2**, which corresponds to requiring an itemset to appear in at least two distinct baskets. The following results were obtained:

Number of frequent singletons (size 1): 381

Number of frequent pairs (size 2): 24

Number of frequent triples (size 3): 2

No frequent itemsets of size 4 were found.

The most frequent singleton is 'the hobbit', The most frequent pair is ['the hobbit there and back again', 'the hobbitt or there and back again illustrated by the author'], and The most frequent triple is ['pride and prejudice', 'emma signet classics', 'sense sensibility']. These results are in line with expectations, as popular literary works tend to co-occur more frequently in user reviews.

5. Discussion

The outcome of the analysis is summarized in **Figure 1**, which shows the number of frequent itemsets discovered by the A-Priori algorithm, grouped by size. As expected, the number of frequent itemsets decreases as the size increases. This drop reflects the stricter condition required for longer itemsets to meet the support threshold.

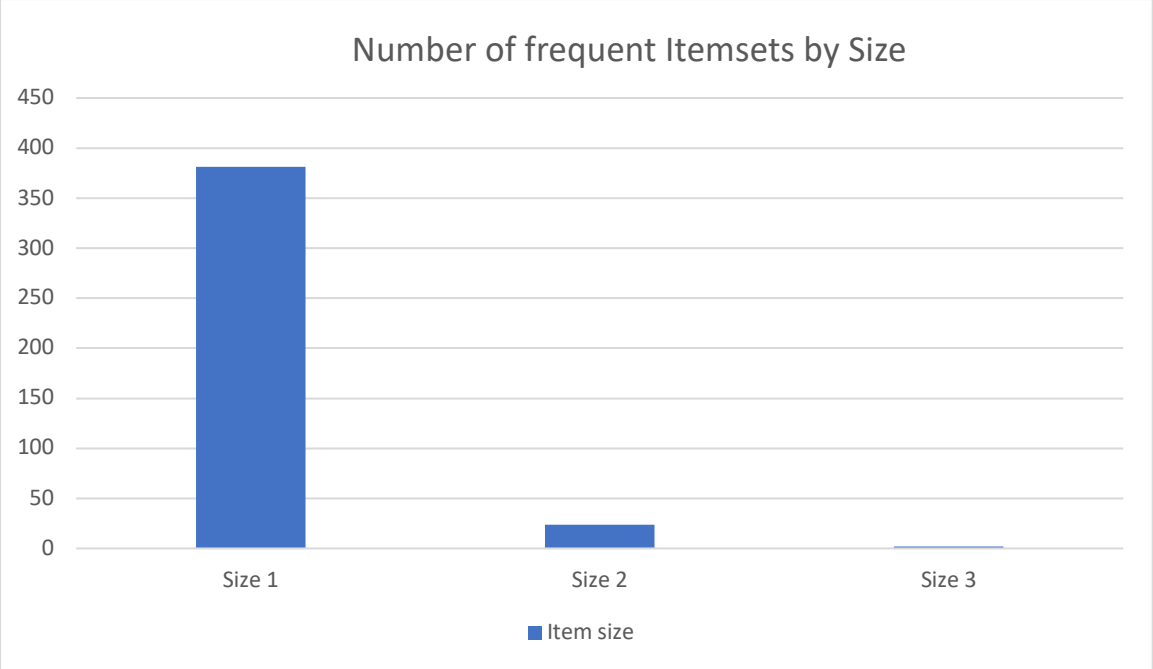

Figure 1: PLOT OF RESULTS OF THE ANALYSIS

Starting from **381** frequent singletons, the count drops to **24** for pairs and further down to just **2** for triples. No frequent itemsets of size four or higher were identified. This is consistent with the general behavior of market-basket data, where larger itemsets become exponentially rarer.

A noteworthy insight from the results is the repeated presence of well-known classics among the most frequent combinations. For instance, the most frequent singleton is *"the hobbit"*, while the most frequent pair links two slightly different editions of the same title, likely reflecting inconsistencies in metadata (e.g., subtitle, publisher's annotation). This suggests that further title normalization could help group equivalent books more effectively.

Similarly, the most frequent triplet *"pride and prejudice", "emma signet classics", and "sense sensibility"* consists entirely of Jane Austen novels. This cluster highlights a thematic pattern in user preferences, reflecting typical reading behavior of users interested in 19th-century literature or particular authors.

Overall, the algorithm has proven capable of uncovering coherent patterns from real-world book review data. However, due to memory and time constraints, a 1% random sample was used. Although this sampling preserves general trends, some less frequent itemsets may have been missed. Future work could involve either better normalization or working on a larger portion of the dataset using a cluster setup.

6 References

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2nd edition, 2014.

Mohamed Bakhet. *Amazon Books Reviews*. Kaggle Dataset. https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews. Accessed June 2025.