Market Basket Analysis

Algorithms for Massive Data
Academic Year 2024/25

Lejda Kolaveri

December, 2025

1 Introduction

Market Basket Analysis (MBA) is a data mining technique traditionally used in retail to discover associations between items purchased together. By identifying frequent itemsets, retailers can uncover latent patterns in consumer behavior. In this project, we generalize this concept to the domain of literature using the **Amazon Books Reviews** dataset.

The project is divided into two distinct analysis tasks:

1. **Task A (*Words as Items*):** We analyze the textual content of reviews. Here, a "basket" is a single review, and the "items" are the significant words within it. The goal is to find linguistic patterns, common topics, and semantic associations.

2. **Task B (*Books as Items*):** We analyze user behavior. Here, a "basket" is a user, and the "items" are the books they have reviewed. The goal is to find books frequently read together, revealing genre clusters and series connections.

Scalability and Methodology: A key requirement of this project is the ability to handle massive datasets. Consequently, the solution is implemented using **Apache Spark (PySpark)**. We utilize the **SON Algorithm** for textual analysis and the **Multistage Algorithm** for user-book analysis. Both algorithms are implemented using MapReduce paradigms and Resilient Distributed Datasets (RDDs), ensuring that the solution scales horizontally on distributed clusters, despite the experimental limitations of the execution environment.

2 Data Description and Preparation

The dataset used is the *Amazon Books Reviews* dataset, published on Kaggle (`mohamedbakhet/amazon-books-reviews`). It consists of approximately 3 million records containing product metadata, user information, and review text.

2.1 Data Organization

For the purpose of this analysis, we extracted only the relevant columns from the raw CSV file:

- `Id`: The unique product identifier for the book.
- `Title`: The name of the book.
- `User_id`: The unique identifier of the reviewer.
- `review/text`: The unstructured textual content of the review.

2.2 Sampling Strategy for Replicability

While the proposed algorithms are designed for Big Data, the experiments were conducted in a constrained environment (Google Colab, $\approx$12GB RAM). To ensure replicability and prevent Out-Of-Memory (OOM) errors during the development phase, we utilized a random subsample of the dataset:

- **Sampling Fraction:** 2% (`SUBSAMPLE_FRACTION` = 0.02).
- **Resulting Size:** This yielded approximately 60,000 text baskets and 3,000 user baskets.

It is crucial to note that the scalability of the solution is preserved: the code employs PySpark's lazy evaluation and caching strategies (specifically `persist(StorageLevel.MEMORY_AND_DISK)`), allowing the exact same pipeline to process the full 3-million-row dataset on a properly provisioned cluster.

2.3 Preprocessing for Task A: Words as Items

To transform raw text into analyzing baskets, we applied a Natural Language Processing (NLP) pipeline:

1. **Tokenization:** Using a `RegexTokenizer`, text was split into tokens, punctuation was removed, and all characters were converted to lowercase to ensure case-insensitivity.
2. **Filtering:** Tokens with fewer than 3 characters were discarded to reduce noise.
3. **Stopwords Removal:** We utilized Spark's `StopWordsRemover` to eliminate common English words (e.g., "the", "and") that possess high frequency but low semantic value.
4. **Basket Formation:** Duplicate words within a single review were removed, as MBA relies on binary presence rather than term frequency. Baskets with fewer than 2 items were discarded.

2.4 Preprocessing for Task B: Books as Items

The primary challenge in Task B was data redundancy, where the same book appeared under multiple IDs (e.g., different editions). We addressed this via the following steps:

1. **Title Normalization:** We cleaned book titles by converting them to lowercase and removing parenthetical information (e.g., "The Hobbit (Paperback)" $\rightarrow$ "the hobbit") and special characters.
2. **Canonical IDs:** We generated a "Canonical ID" for each unique normalized title. This allowed us to merge different editions of the same work into a single item entity.
3. **Basket Formation:** Data was grouped by `User_id`. Users who reviewed fewer than 2 books were filtered out, as they cannot contribute to pair generation.

3 Task A: Market Basket Analysis on Words

The first objective of this project is to perform Market Basket Analysis on the textual content of the reviews. We treat reviews as baskets and words as items.

3.1 Methodological Approach: *The SON Algorithm*

An approach like A-Priori fails on massive datasets because counting all candidate pairs requires more memory than a single machine possesses. To address this, we implemented the **SON Algorithm** (Savasere, Omiecinski, and Navathe).

The SON algorithm is a two-pass MapReduce technique designed for distributed environments:

1. **Phase 1 (Local Processing):** The dataset is divided into chunks (partitions). We run the A-Priori algorithm locally on each partition using a lowered support threshold $s_{local} = s \times (\text{partition size}/\text{total size})$.

2. **Phase 2 (Global Verification):** The union of all locally frequent itemsets forms the set of *Candidates*. In the second pass, we count the exact support of these candidates across the entire dataset to filter out false positives.

This method guarantees *no false negatives*: if an itemset is frequent globally, it must be frequent in at least one partition.

3.2 Implementation Details

The algorithm was implemented in PySpark using the following configuration:

- **Support Threshold (s):** 1% (0.01). Given the sample size of $N = 60,322$ baskets, an itemset required at least ≈ 603 occurrences to be considered frequent.

- **Basket Definition:** A set of distinct words from a single review (duplicates removed).

3.3 Experimental Results

The execution of the SON algorithm yielded **2,779 confirmed frequent itemsets**. The analysis of these itemsets reveals distinct patterns in how users describe books.

3.3.1 Frequent Itemsets

The SON algorithm successfully identified frequent patterns of varying lengths ($k = 1, 2, 3$). The results indicate a strong dominance of domain-specific terms.

Table 1 shows the most frequent individual words. As expected, "book" appears in over 75% of all reviews. Tables 2 and 3 display the most frequent pairs and triplets, revealing that complex itemsets are largely formed by combinations of these top singletons.

Word $(k = 1)$	**Occurrences**	**Support (%)**
book	45,565	75.54%
read	28,726	47.62%
one	22,327	37.01%
like	15,680	25.99%
story	14,459	23.97%

Table 1: Top 5 Frequent Single Words.

Itemset $(k = 2)$	**Occurrences**	**Support (%)**
{book, read}	23,463	38.90%
{book, one}	17,532	29.06%
{book, like}	12,900	21.39%
{one, read}	12,562	20.82%
{book, good}	11,088	18.38%

Table 2: Top 5 Frequent Word Pairs.

Itemset $(k = 3)$	**Occurrences**	**Support (%)**
{book, one, read}	10,514	17.43%
{book, like, read}	7,658	12.70%
{book, read, story}	6,875	11.40%
{book, read, reading}	6,669	11.06%
{book, read, time}	6,649	11.02%

Table 3: Top 5 Frequent Word Triplets.

3.3.2 Association Rules and Lift Analysis

To find more meaningful relationships beyond simple frequency, we calculated Association Rules based on **Lift**. Lift measures how much more likely two words are to appear together than if they were independent $(Lift(A \to B) = \frac{Conf(A \to B)}{Supp(B)})$.

Table 4 highlights the rules with the highest Lift.

Antecedent	**Consequent**	**Support**	**Confidence**	**Lift**
highly	recommend	3.30%	51.38%	**4.80**
recommend	highly	3.30%	30.80%	**4.80**
anyone	recommend	2.78%	30.76%	2.87
character	characters	3.42%	40.02%	2.71
ever	best	3.31%	33.18%	2.66
characters	novel	4.63%	31.31%	2.57

Table 4: Top Association Rules sorted by Lift.

Discussion: The results demonstrate the algorithm's effectiveness in capturing semantic meaning:

- **Linguistic Collocations:** The highest lift (4.80) is observed for the pair `{highly, recommend}`. This indicates that if a user writes "highly", there is a 51% probability the next important word is "recommend".
- **Sentiment Analysis:** The pair `{best, ever}` (Lift 2.66) captures strong positive sentiment.
- **Topic Clustering:** The association between `novel` and `characters` suggests that reviews of novels focus heavily on character development, a specific feature of that genre.

4 Task B: Market Basket Analysis on Books

The second objective of this project is to analyze user behavior. In this setting, we consider "Users" as baskets and "Books" as items. The main goal is to identify pairs of books that are frequently reviewed together.

4.1 Methodological Approach: The Multistage Algorithm

For this second task, we chose to implement a different algorithmic strategy to demonstrate an alternative approach to handling Big Data. While the SON algorithm (used in Task A) relies on partitioning the dataset, the **Multistage Algorithm** focuses on optimizing memory usage.

In a scenario with many unique items (books), a standard counting approach would generate an enormous number of candidate pairs. Storing a count for every possible pair would quickly exhaust the system's RAM (memory).

To address this memory barrier, we implemented the **Multistage Algorithm**, an extension of the PCY (Park-Chen-Yu) algorithm. Instead of partitioning the dataset, this method uses **hashing techniques** to filter out infrequent pairs before we even attempt to count them. It works in four steps (passes) to progressively reduce the data size:

1. **Pass 1:** We count the frequency of individual books. Books that are not frequent on their own are removed immediately.
2. **Pass 2 (First Hash):** We take pairs of frequent books and hash them into buckets. We create a "bitmap" (a map of 0s and 1s) to remember which buckets contain frequent pairs.
3. **Pass 3 (Second Hash):** We hash the pairs again using a *different* function. Crucially, we only check pairs that already passed the first filter. This second step removes "false positives" (collisions) where infrequent pairs accidentally landed in a full bucket during the previous pass.
4. **Pass 4 (Final Count):** We count the exact support of the few pairs that survived both hash filters.

4.2 Implementation Details

The algorithm was executed on the user sample ($N = 3,032$ baskets) with the following configuration:

- **Support Threshold (s):** 0.1% (0.001). Given the sample size, the absolute support required was calculated as $\lfloor 3,032 \times 0.001 \rfloor = 3$ occurrences. This low threshold was chosen because book transaction data is highly sparse.
- **Hash Table Size:** We utilized 20,000 buckets for both Pass 2 and Pass 3. This size was sufficient to minimize collisions for the sample dataset.
- **Persistence:** To optimize performance, we utilized Spark's `persist(MEMORY_AND_DISK)` strategy. This cached the preprocessed dataframe in memory, preventing redundant re-computation during the iterative passes of the algorithm.

4.3 Experimental Results

4.3.1 Algorithm Efficiency

The Multistage approach proved to be very efficient. Table 5 shows how the hashing filters significantly reduced the number of candidates we had to check.

Stage	**Count**
Frequent Singles (Pass 1)	504 books
Frequent Buckets (Bitmap 1)	67 buckets
Frequent Buckets (Bitmap 2)	56 buckets
Final Confirmed Pairs	**56 pairs**

Table 5: Reduction of candidates through Multistage filtering.

We can see that the number of frequent buckets dropped from 67 in the first hash to 56 in the second hash. This confirms that the second pass successfully removed false positives.

4.3.2 Top Frequent Book Pairs

Table 6 presents the most frequent book pairs found by the algorithm.

Book A	**Book B**	**Support**
Pride & Prejudice (New Windmill)	Pride and Prejudice	0.72%
The Hobbit (Or, There and Back Again)	The Hobbit	0.69%
Little Women	Little women (Meg, Jo...)	0.56%
The Scarlet Letter	The Scarlet Letter A Romance	0.39%
A Connecticut Yankee...	Life on the Mississippi	0.29%
Wuthering Heights	Jane Eyre / Wuthering Heights	0.29%
Persuasion (World's Classics)	Emma (World's Classics)	0.19%
Wuthering Heights	Jane Eyre	0.19%

Table 6: Top Frequent Book Pairs (Selected).

Discussion of Findings: The results highlight two distinct patterns in user behavior:

1. **Duplicate Editions:** The highest-ranked pairs (such as *Pride & Prejudice*) often consist of different versions of the exact same book. This indicates that a significant number of users in the dataset review the same title multiple times, perhaps for different editions (e.g., paperback vs. hardcover).

2. **Genre Clustering:** The algorithm successfully identified groups of books that belong to the same literary category.

    - **Jane Austen Works:** Users who read *Persuasion* are statistically likely to also read *Emma* and *Pride & Prejudice*, showing a preference for this specific author.
    - **19th-Century Classics:** We found a strong connection between *Wuthering Heights* (by Emily Brontë) and *Jane Eyre* (by Charlotte Brontë). This demonstrates that users who read one classic novel from this period are very likely to read others.

5 Conclusion

This project demonstrated that Frequent Itemset Mining is a versatile tool capable of extracting meaningful insights from both unstructured text and structured user behavior data. By moving beyond traditional retail applications, we uncovered distinct semantic patterns in book reviews and identified strong genre-based clusters among readers.

The comparison between the two tasks highlights the importance of selecting the right algorithmic strategy for the specific data challenge:

- In the textual domain, the *SON Algorithm* proved effective at managing distributed counting, successfully revealing linguistic associations without the need for complex language models.
- In the user-behavior domain, the *Multistage Algorithm* showcased the power of hashing. It demonstrated that memory barrier caused by massive candidate sets can be efficiently mitigated through iterative filtering.

From a technical point of view, the project confirms the strength of the MapReduce framework. Even though the experiments used only a sample of the data, the design based on Apache Spark's RDDs and caching can easily handle the full 3-million-row dataset on a bigger cluster. Ultimately, this work confirms that classical data mining techniques remain highly relevant and scalable in the Big Data era.

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.