Algorithms for Massive Data Market-Basket Analysis on Amazon Books Reviews

Precious Prince
MSc in Data Science for Economics
Università degli Studi di Milano

Academic Year 2024/2025

Abstract

This project applies association rule mining to the *Amazon Books Review* dataset using both Spark's built-in FP-Growth algorithm and a custom PySpark implementation. The goal is to discover co-purchased or co-reviewed books and generate meaningful user-level recommendations. By leveraging distributed computation in Spark, the solution scales efficiently to massive datasets while maintaining interpretability and transparency.

1 Dataset and Pre-processing

The dataset was sourced from Kaggle (`mohamedbakhet/amazon-books-reviews`), licensed under CC0. It includes user IDs, book ASINs, titles, and ratings. The analysis treated each user as a basket and each reviewed book as an item.

Preprocessing

- Dropped records with missing `User_id`, `Id`, or `Title`.
- Created a basket for each user with at least two reviewed books.
- Used a 1% random subsample for local testing, with global configuration variables controlling fraction and reproducibility.

2 The FP-Growth Algorithm

Definition and Purpose

The **Frequent Pattern Growth (FP-Growth)** algorithm is a highly efficient method for discovering frequent itemsets and association rules in transactional or basket data. Unlike the classical Apriori algorithm, which repeatedly scans the dataset to generate candidate itemsets, FP-Growth builds a compact tree structure known as the **FP-tree**

(Frequent Pattern Tree). This structure compresses transactions by storing only frequent items and their conditional co-occurrence paths.

How It Works

1. **Scan 1:** Identify all frequent items that meet the minimum support threshold and order them by frequency.
2. **Scan 2:** Construct an FP-tree by inserting transactions following the ordered frequent items, sharing common prefixes.
3. **Recursive mining:** For each item, build a conditional FP-tree that captures items frequently appearing with it, thereby discovering all frequent patterns without generating explicit candidates.

Advantages and Usage

FP-Growth drastically reduces the number of database scans (only two in total) and avoids the combinatorial explosion of candidate generation. Its ability to handle large-scale datasets efficiently makes it ideal for market-basket analysis, web clickstream analysis, and recommendation systems. In this project, it is used to uncover relationships between books reviewed by the same users and to recommend new books based on those patterns.

3 Spark FP-Growth Implementation

Configuration

- **minSupport = 0.001:** Retain itemsets appearing in at least 0.1% of user baskets.
- **minConfidence = 0.5:** Keep rules where the consequent appears in at least half of antecedent baskets.

Results

- **Total association rules:** 15
- **Unique antecedent itemsets:** 14
- Example rules:
  - *Emma* $\rightarrow$ *Sense and Sensibility* (confidence = 0.5)
  - *Leaves of Grass* $\leftrightarrow$ *Leaves of Grass (Illustrated Modern Library)* (confidence = 1.0)
  - *The Hobbit* $\rightarrow$ *The Hobbit: There and Back Again* (confidence = 0.75)

Personalized Recommendations

Each user basket was joined with rules whose antecedents appeared within it. Books already read or with identical titles (different ASINs) were filtered out. Recommendations were ranked per user using average rating.

User-level Insights

- Each user received between 1–2 top book recommendations.
- The most frequently recommended titles were:
  - *Emma (Signet Classics)* — recommended to 2 users.
  - *A Connecticut Yankee in King Arthur's Court* — recommended to 2 users.
  - *Sense and Sensibility* — recommended to 2 users.
- Maximum books recommended to any user: **1**.

Most Reviewed Books

Top titles by total reviews:

- **The Hobbit** — 50 reviews.
- **Harry Potter and the Sorcerer's Stone** — 43 reviews.
- **The Hobbit (various editions)** — multiple ASINs appearing among top reviewed items.

4 Custom FP-Growth Implementation (from Scratch)

To demonstrate algorithmic understanding and scalability, FP-Growth was reimplemented in PySpark using DataFrame transformations.

Steps

1. Computed single-item supports and frequent pairs.
2. Defined a UDF `generate_pairs()` to list all unique item pairs within each user basket.
3. Aggregated pair supports and computed:

$$\text{Confidence}(A \to B) = \frac{\text{count}(A, B)}{\text{count}(A)}, \quad \text{Lift} = \frac{\text{Confidence}(A \to B)}{\text{Support}(B)}.$$

4. Applied the same `minConfidence = 0.5`.
5. Introduced a custom ranking metric:

$$\text{Custom Score} = 0.6 \times \text{Confidence} + 0.4 \times \text{Lift}.$$

Results

- **Total rules generated:** 15
- **Rules retained (confidence 0.5):** 3
- Example top rules:
  - *Emma (Signet Classics)* → *Sense and Sensibility* (confidence = 0.5, lift = 223.7, score = 89.8)
  - *Emma (Signet Classics)* → *Emma* (confidence = 0.5, lift = 223.7, score = 89.8)

User Recommendations

Each user basket was cross-joined with custom rules:

- Each user received up to 2 recommended books.
- Example recommendations:
  - *Sense and Sensibility* and *Emma* were recommended to 4 distinct users each.

Comparison with Spark FP-Growth

	Spark FP-Growth	**Custom FP-Growth**
Implementation	Built-in `pyspark.ml.fpm`	Custom (UDF + joins)
Total rules generated	15	15 (3 retained after filtering)
Support threshold	0.001	0.001 (min count 3)
Ranking metric	Avg. rating	Custom confidence–lift score
Top recommended books	Emma, Sense and Sensibility	Same (validated)
Scalability	Automatic	Verified via Spark parallelism

Both implementations returned consistent associations, validating the correctness of the custom approach while illustrating the transparency and flexibility of manual computation.

5 Insights and Discussion

- Highly reviewed books (*The Hobbit*, *Emma*) dominate frequent itemsets.
- Filtering identical titles across ASINs improved recommendation quality.
- The custom FP-Growth achieved comparable results to the library version, proving its correctness and scalability.
- The top recommended books (*Emma*, *Sense and Sensibility*) appear in both models' top-10 lists.

6 Conclusion

This project demonstrated the scalability of FP-Growth for large datasets and the feasibility of reproducing it manually in PySpark. The Spark implementation ensured efficiency, while the custom version provided transparency and algorithmic understanding. Future improvements include extending to higher-order itemsets ($k > 2$) and integrating genre or rating-based filtering for more contextual recommendations.

Declaration

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, 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 them. 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.

Precious Prince