UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE POLITICHE, ECONOMICHE E SOCIALI

Master's Degree Course in Data Science for Economics

ALGORITHMS FOR MASSIVE DATA: MARKET-BASKET ANALYSIS

Instructor: Prof Dario Malchiodi

Report by: Timur Rezepov
Student ID: 34177A

Academic Year 2024-2025

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.

Contents

1. Introduction

The market-basket analysis is used to describe many-to-many relationship between two kinds of objects.

The results of market-basket analysis can be presented in the form of association rules: $I \to j$, where j is an item.

The core concepts of market-basket analysis are:

- `item` - individual entity (product, book, etc);
- `basket` or `transaction` - a collection of items seen (bought, liked) together;
- `frequent itemset` - a set of items I which appears in the number of baskets as a subset;
- `support` - a number of baskets for which I is a subset;
- `confidence` of the association rule - is the fraction of the baskets with all of I that also contain j.

The defined association rules can then further be used in a various ways: recommendation systems, cross-selling, bundle strategies.

The theoretical concepts and solution approaches are mostly based on the course material and textbook Mining of Massive Datasets(A. Rajaraman, J. Ullman):

- `frequent itemsets`;
- `finding similar items`;
- `MapReduce`.

The software used: python3.12, jupyter notebook, pandas.

2. Dataset Description

The project is based on based on the Amazon Books Review dataset.
The subjects of market-basket analysis in this work are:

- `items` - books;
- `baskets` - reviews made by a same user.

`Books_rating.csv` dataset will be used as it's data is sufficient for the goals and requirements of the analysis.

The dataset contains 3M records of user reviews on 212404 unique books. The dataset features used in the analysis are shown in the table:

Feature	Description
Id	Book id
Title	Book title
User_id	User id
review/text	Review text

3. Preprocessing

First of all, some basic exploratory data analysis was performed on the data sample.
Two issues were found:
1. Missing values in `user_id` column. Such records were excluded.

```
RangeIndex: 30001 entries, 0 to 30000
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       30001 non-null  object
 1   title    30000 non-null  object
 2   user_id  24239 non-null  object
 3   text     30001 non-null  object
dtypes: object(4)
memory usage: 937.7+ KB
```

2. Duplicated review texts of the same book (under distinct ids) by the same user. Such records were excluded as they don't contain any useful information in the context of market-basket analysis of book.

	id	title	user_id	text
919	158726398X	Great Expectations	A38EGLFFVO31GQ	This is a wonderful opportunity to read the cl...
2832	B0008BSPC6	Northanger abbey: A novel	A1I5ORI0ROMCIQ	While this was not my favorite Jane Austen, or...
2865	B0008BSPD0	Northanger Abbey: A novel (The select library ...	A1I5ORI0ROMCIQ	While this was not my favorite Jane Austen, or...
2932	1578152445	Great Expectations	A38EGLFFVO31GQ	This is a wonderful opportunity to read the cl...

The distribution of basket sizes was analysed. The vast majority of users have only one review - it means that the majority of baskets contain only one item. Consequently, the analysis can be limited to analysing frequent pairs (doubletons).

size	share, %
1	95.0
2	4.0
3	1.0
4	0.0
5	0.0

At last, string item ids were converted to integer ids to reduce RAM and CPU consumption.

	user_id	id	id_int
0	A2FX00QQ0HR1AW	B000NKGYMK	0
1	A7IA8CTTSQ7A4	B000NKGYMK	0
2	A1CEJFXSJYQBTX	0789480662	1
3	AQW8KZY926JYU	B0007DVHU2	2
4	A1C2IN2HR2TBX4	0312322291	3

4. Algorithms

In market-basket analysis we are interested in the absolute number of baskets that contain a particular set of items. The naive approach is to generate all the pairs for each basket using the double loop. But it may fail if there are too many pairs of items to count in main memory.

In order to reduce the number pairs that must be counted special algorithms were designed. These algorithms are based on several passes over data and monotonicity if itemsets (if a set I of items is frequent, then so is every subset of I). In this work two algorithms are condisered: A-Priori and PCY.

4.1 A-Priori

The A-Priori Algorithm performs two passes over data:

1. Count the occurences of each item in baskets - determine the `frequent items`;
2. Count all the pairs that consist of frequent items - `candidate pairs`;
3. Define the support threshold (typically 1% of baskets) and analyse the resulting `frequent pairs`.

The A-Priori algorithm is good enough when it has enough memory to count all the candidate pairs. But in most cases the number of frequent pairs is very small, so the A-Priori algorithm uses unnecessary resources. In the case of considered dataset sample only 3% of all pairs turned out to be frequent.

4.2 PCY

The PCY (Park, Chen and Yu) algorithm exploits the fact that A-Priori uses a lot of unnecessary resources to count of frequent singletons. During the first pass the PCY algorithm not only counts each item occurences, but also generates pairs, hashes them to buckets and counts. If the count of a bucket is at least as great as the support threshold, it is called a frequent bucket.

In order to implement PCY algorithm I made next changes to the A-Priori:

- the algorithm will not generate all possible candidate pairs and loop through them with dataframe lookups;
- instead, it will go through each basket with frequent items and implement a bucket (=pair) counter while examining baskets.

In order to compare the two algorithms I have run both of them multiple times. The PCY's performance is times faster on average (40 000 vs 8 microseconds):

- A-Priori: 39.2 milliseconds ± 4.81 per loop (mean ± std. dev. of 7 runs, 3 loops each);
- PCY: 7.82 microseconds ± 4.52 per loop (mean ± std. dev. of 7 runs, 3 loops each).

The resulting sets of frequent pairs for both algorithms run on the full dataset showed no difference. On average the A-Priori implementation needs 1700 millseconds to get frequent pairs from the full dataset, while the PCY implementation - less than 100 milliseconds.

4.3 Scaling

The scaling of PCY implementation in terms of used memory was analysed. To do this the total memory used by objects in the enviroment was estimated for two cases: partial dataset and full dataset.

```
-----Run on a sample-----
Dataset size: 24425
Number of baskets: 22097
Number of unique items: 15164
Frequent items count:  0
Memory used (MB): 1

-----Run on a full dataset-----
Dataset size: 2397614
Number of baskets: 1008972
Number of unique items: 216023
Frequent items count:  0
Memory used (MB): 55
```

I would say that the PCY implementation scales well enough: to process a 100 times larger dataset the we need 54 MB of RAM more.

5. Market-basket analysis

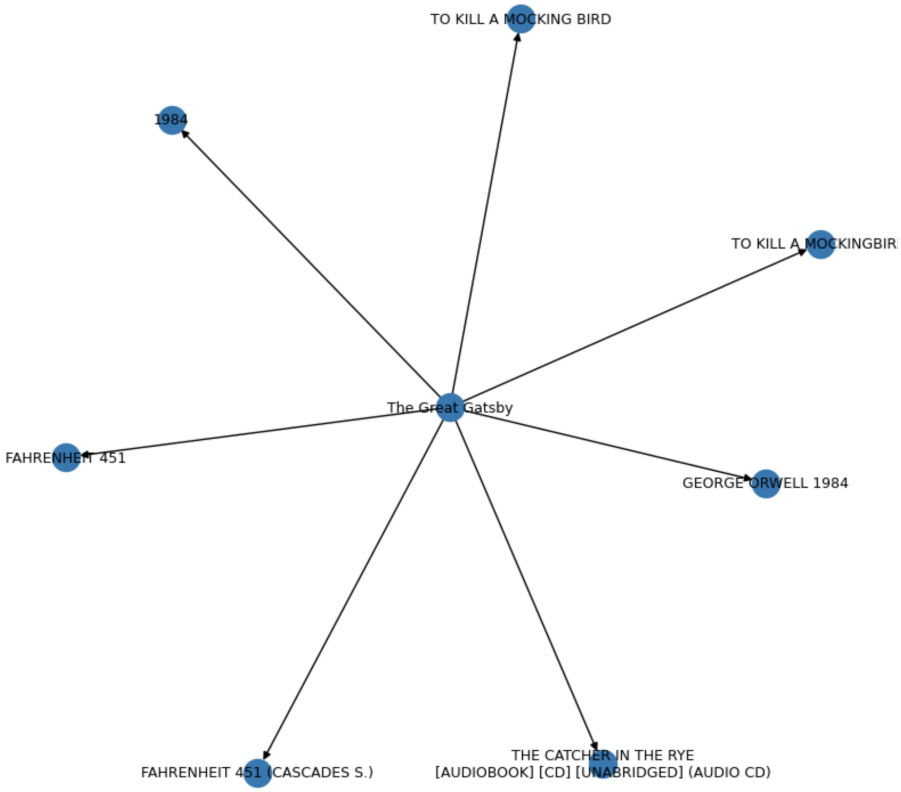
Now when the frequent pairs are defined, the corresponding association rules can be determined. The `support` metric of the association rule alone can be misleading: for example, the support of the association rule of two independently popular items might be high, but such rules are useless from marketing point of view. `confidence` and `interest` metrics can help us gain more valuable insights.

- `confidence` of the association rule - is the fraction of the baskets with all of I that also contain j;
- `interest` of the rule -is the difference between its confidence and the fraction of baskets that contain j.

The cosine similarity was used to determine the same books with slightly different titles to clear the resulting association rules. By controlling the metrics of the association-rules some interesting and useful rules can be found.

I_title	j_title	confidence	interest	similarity
The Lord of the Rings Trilogy 3 Volumes	The Hobbit	0.12	0.12	0.48
The Lord of the Rings (3 Volume Set)	The Hobbit	0.12	0.12	0.49
The Great Gatsby	The Catcher in the Rye [Audiobook] [Cd] [Unabr...	0.09	0.08	0.42
Great Gatsby (Everyman)	The Catcher in the Rye [Audiobook] [Cd] [Unabr...	0.08	0.08	0.40
Jane Eyre	Pride & Prejudice (Classic Library)	0.08	0.08	0.49

The association rule can be represented by a network chart.

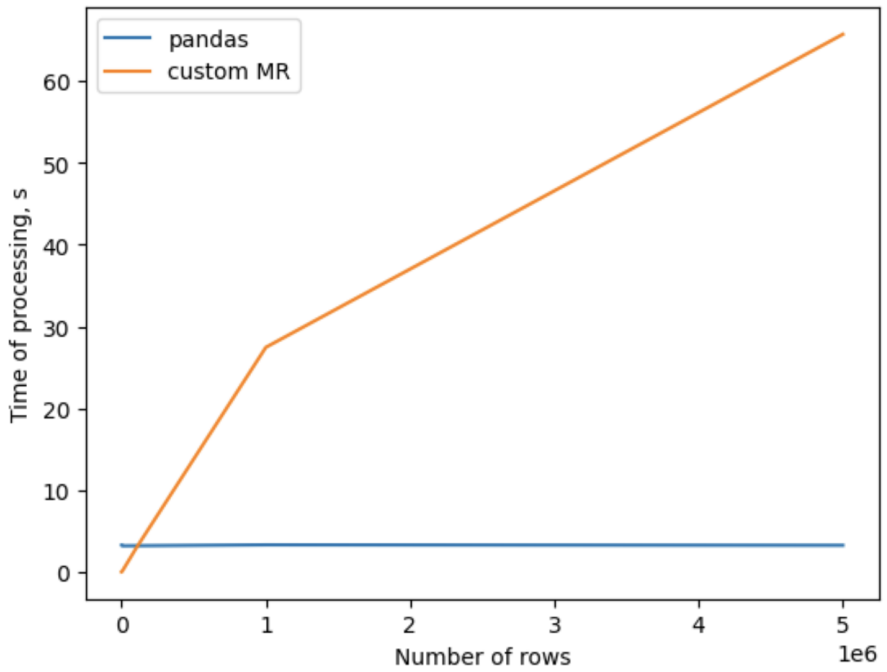


6. Experiment: MapReduce

Although the full dataset is processed pretty quickly I would like to implement the map reduce algorithm in order to make the entire implementation faster. No external libraries will be used, the solution will be based on python multiprocessing. Experiment: Analysing the data processing steps, I have come to the conclusion that the slowest part is selecting baskets with frequent items. My MapReduce implemention will consist of:

- Map step: check if a basket contains any of the frequent items
- Reduce step: aggregate the results

The analysis of time required for selecting rows with frequent items shows that the custom MapReduce implementation performs faster than pandas filtering at small dataset sizes (up to hundreds of thousands of rows). The pandas selection performs almost at a constant time, while custom implementation needs more time as dataset becomes larger. This fact may highlight the drawbacks of custom implementation, persumably, in the reduce algorithm.



7. Keypoints

In this project work the market-basket analysis is performed on the *Amazon book reviews* dataset covering the entire process from data processing and algorithms implementation to defining association rules.

1. Concepts and applications of market-basket analysis are discussed;
2. A-Priory and PCY algorithms are implemented, brief analysis of their advantages and limitations is provided;
3. PCY algorithm performance is assessed in terms of RAM usage;
4. Cosine similarity is used to enhance association rule outputs;
5. A basic MapReduce solution is implemented to improve the algorithm performance.

Although the implementations are developed mostly from scratch they already yield meaningful results concerning association rules.