MIDTERM PROJECT REPORT

IBRAHIM O MUSA
Email: iom@njit.edu

03/05/2024

—

Data Mining

—

Professor: Yasser Abduallah

Implementation and Code Usage

Brute Force, Apriori, and FP-Growth Algorithms for Retail Data Mining

Abstract:
In this project, I explore and compare the Brute Force method, Apriori algorithm, and FP-Growth algorithm to uncover associations within retail transactions from Amazon, Best Buy, WayFair, Walmart, and Nike. By implementing these algorithms and employing various data mining concepts, principles, and methods, I assess their effectiveness and efficiency. Through the design and development of custom data mining tools, I create models for mining valuable insights from transaction data.

Introduction:
Data mining is a powerful approach for uncovering hidden patterns and associations within large datasets. This project focuses on the Brute Force method, Apriori algorithm, and FP-Growth algorithm and their application in a retail context. I outline the core data mining concepts and principles applied in the work.

Core Concepts and Principles:

Frequent Itemset Discovery:
The algorithms revolve around discovering frequent itemsets, i.e., sets of items that frequently co-occur in transactions. These itemsets provide insights into customer purchase behavior and preferences.

Support and Confidence:
Two key metrics in data mining are support and confidence. Support measures how frequently an item or itemset occurs, while confidence assesses the likelihood of items being purchased together. These metrics guide the analysis.

Association Rules:
By determining strong association rules, I identify which items are commonly purchased together. These rules are instrumental for optimizing sales strategies, such as recommendations.

Project Workflow:
The project follows a structured workflow involving various stages and the application of the Brute Force method, Apriori algorithm, and FP-Growth algorithm:

Data Generation and Loading: I begin by generating deterministic transaction data for five retailers - Amazon, Best Buy, WayFair, Walmart, and Nike. Each retailer has its own unique pattern for generating transactions. The transaction data is then loaded from CSV files.
User Input for Minimum Support and Confidence: User input is collected for minimum support and confidence levels to filter out less significant patterns. The user also selects which retailer's data to analyze.

Brute Force Method: The Brute Force method generates all possible itemsets and checks their frequency against the minimum support threshold. Frequent itemsets are used to generate association rules meeting the minimum confidence.

Apriori Algorithm: The Apriori algorithm iteratively generates candidate itemsets of increasing sizes, starting with single items. It calculates the support of each candidate itemset and retains only those meeting the minimum support. Confident association rules are then derived.
FP-Growth Algorithm: The FP-Growth algorithm uses a frequent pattern tree (FP-tree) structure to efficiently discover frequent itemsets without candidate generation. It then mines association rules from the frequent itemsets.
Data Generation Patterns:
Each retailer has a unique pattern for generating transactions, reflecting their characteristics:

Amazon: The pattern is the index of the transaction modulo 10 plus 1.
Best Buy: The pattern involves finding the next prime number greater than the index of the transaction modulo the total number of items plus 1.
WayFair: The pattern is twice the index of the transaction plus 1, but not more than the total number of items.
Walmart: The pattern is twice the index of the transaction plus 2.
Nike: The pattern generates the Fibonacci sequence, and the index of the transaction plus 1 is used to find the corresponding number in the sequence, determining the number of items in the transaction.

Pattern definition code snapshot:

```
# Patterns for each retailer
patterns = {
    'amazon': lambda i, _: i % 10 + 1,
    'wayfair': lambda i, _: 2 * i + 1 if 2 * i + 1 <= len(retailers_items['wayfair']) else 1,
    'walmart': lambda i, _: 2 * (i + 1),
    'bestbuy': lambda i, num_items: next((x for x in range(i % num_items + 1, num_items + 1) if is_prime(x)), 1),
    'nike': lambda i, _: fibonacci(i + 1) if fibonacci(i + 1) <= len(retailers_items['nike']) else 1
}

def generate_deterministic_transactions(retailer, num_transactions):
    if retailer not in retailers_items or retailer not in patterns:
        print(f"No items or pattern found for retailer: {retailer}")
        return

    transactions = []
    for i in range(num_transactions):
        num_items = patterns[retailer](i, len(retailers_items[retailer]))  # Apply the retailer's pattern to determine t
        transaction = retailers_items[retailer][:num_items]  # Select the first 'num_items' from the list
        transactions.append(", ".join(transaction))

    return pd.DataFrame({'Transaction ID': [f'Trans{i+1}' for i in range(num_transactions)],
                         f'{retailer.capitalize()} Transaction': transactions})
```

User Interaction and Input:
The project prompts the user to enter the minimum support and confidence levels, as well as the retailer choice. These inputs guide the analysis and filtering of patterns. The user interaction ensures flexibility and customization of the data mining process.
Fig1:

```
Enter the minimum support level (between 0 and 1): 0.8
Enter the minimum confidence level (between 0 and 1): 0.9
Which database would you like to analyze? Enter '1' for Amazon, '2' for Best Buy, '3' for WayFair, '4' for WalMart, or '5
```

Algorithm Comparison:
The Brute Force method exhaustively generates all possible itemsets and checks their frequency. It is straightforward but computationally expensive.
The Apriori algorithm employs a level-wise approach, generating candidate itemsets and pruning infrequent ones. It is more efficient than the Brute Force method but may still generate a large number of candidates.
The FP-Growth algorithm uses a compact FP-tree structure to efficiently discover frequent itemsets without candidate generation. It is generally faster than Apriori, especially for large datasets.

Insightful Association Rules:
The generated association rules provide valuable insights into customer purchasing patterns. For example:

Amazon: Customers who buy "Kindle E-reader" often also buy "Echo Dot (4th Gen)" (confidence: 0.8).
Best Buy: Customers who purchase "Apple - AirPods Pro" are likely to also buy "iPhone 12 Pro" (confidence: 0.7).
WayFair: Customers who buy "Wayfair Basics 1800 Series Sheet Set" tend to also purchase "Amherst Upholstered Platform Bed" (confidence: 0.6). These rules can inform product recommendations, bundling strategies, and store layouts.

Sample Frequent items and association outputs:

Brute Force:

```
Brute Force Association Rules:
                                    antecedents  \
0                          (Apple - AirPods Pro)
1  (Insignia™ - 50" Class F30 Series LED 4K UHD S...

                                    consequents  antecedent support  \
0  (Insignia™ - 50" Class F30 Series LED 4K UHD S...                 NaN
1                          (Apple - AirPods Pro)                 NaN

   consequent support  support  confidence  lift  leverage  conviction  \
0                 NaN      0.7         NaN   NaN       NaN         NaN
1                 NaN      0.7         NaN   NaN       NaN         NaN

   zhangs_metric
0            NaN
1            NaN
```

Output for empty set:

```
No frequent itemsets met the minimum support level using the brute force method.
```

Apriori and FP-Growth method outputs:

```
Apriori Association Rules:
                                      antecedents  \
0                             (Apple - AirPods Pro)
1  (Insignia™ - 50" Class F30 Series LED 4K UHD S...
2                          (Samsung - Galaxy S21 5G)
3                             (Apple - AirPods Pro)
4                          (Samsung - Galaxy S21 5G)
5     (Samsung - Galaxy S21 5G, Apple - AirPods Pro)
6  (Samsung - Galaxy S21 5G, Insignia™ - 50" Clas...
7  (Apple - AirPods Pro, Insignia™ - 50" Class F3...
8                          (Samsung - Galaxy S21 5G)
9                             (Apple - AirPods Pro)

                                      consequents  antecedent support  \
0  (Insignia™ - 50" Class F30 Series LED 4K UHD S...                 0.7
1                             (Apple - AirPods Pro)                 1.0
2                             (Apple - AirPods Pro)                 0.5
3                          (Samsung - Galaxy S21 5G)                 0.7
4  (Insignia™ - 50" Class F30 Series LED 4K UHD S...                 0.5
5  (Insignia™ - 50" Class F30 Series LED 4K UHD S...                 0.5
6                             (Apple - AirPods Pro)                 0.5
7                          (Samsung - Galaxy S21 5G)                 0.7
8  (Apple - AirPods Pro, Insignia™ - 50" Class F3...                 0.5
9  (Samsung - Galaxy S21 5G, Insignia™ - 50" Clas...                 0.7
```

```
   consequent support  support  confidence      lift  leverage  conviction  \
0                 1.0      0.7    1.000000  1.000000      0.00         inf
1                 0.7      0.7    0.700000  1.000000      0.00        1.00
2                 0.7      0.5    1.000000  1.428571      0.15         inf
3                 0.5      0.5    0.714286  1.428571      0.15        1.75
4                 1.0      0.5    1.000000  1.000000      0.00         inf
5                 1.0      0.5    1.000000  1.000000      0.00         inf
6                 0.7      0.5    1.000000  1.428571      0.15         inf
7                 0.5      0.5    0.714286  1.428571      0.15        1.75
8                 0.7      0.5    1.000000  1.428571      0.15         inf
9                 0.5      0.5    0.714286  1.428571      0.15        1.75

   zhangs_metric
0            0.0
1            0.0
2            0.6
3            1.0
4            0.0
5            0.0
6            0.6
7            1.0
8            0.6
9            1.0

FP-Growth Association Rules:
{('Apple - AirPods Pro',): (('Insignia™ - 50" Class F30 Series LED 4K UHD Smart Fire TV', 'Samsung - Galaxy S21 5G'}
```

Output for empty set:

```
No frequent itemsets met the minimum support level using the Apriori method.
No frequent patterns met the minimum support level using the FP-Growth method.
```

Sample comparison outputs:

```
# Print the time taken by each method
print("\nTime taken by brute force method: ", brute_force_time)
print("Time taken by Apriori: ", apriori_time)
print("Time taken by FP-Growth: ", fpgrowth_time)
```

```
Time taken by brute force method:  0.002124309539794922
Time taken by Apriori:  0.010610103607177734
Time taken by FP-Growth:  0.0003476142883300781
```

Another sample

```
Time taken by brute force method:  394.8061378002167
Time taken by Apriori:  0.008041620254516602
Time taken by FP-Growth:  0.00013327598571777344
```

Challenges and Future Enhancements:

One challenge faced during the project was the computational complexity of the Brute Force method for large datasets. Future enhancements could include implementing more efficient algorithms like FP-Max or exploring distributed computing techniques for scalability.

Results and Evaluation:

The project's effectiveness and efficiency are evaluated based on the frequent itemsets and association rules generated by each algorithm. The execution time of each algorithm is also compared. The Brute Force method took significantly longer compared to Apriori and FP-Growth, highlighting the need for efficient algorithms in data mining.

Conclusion:

In conclusion, this project demonstrates the application of data mining concepts, principles, and methods using the Brute Force method, Apriori algorithm, and FP-Growth algorithm. It successfully extracts meaningful association rules from retail transaction data. The comparison of these algorithms exemplifies the power of data mining in revealing valuable patterns for decision-making in the retail industry.

References:

- Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.
- Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).
- Tan, P. N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). Introduction to data mining (2nd ed.). Pearson.
- CS 634 Data Mining Spring 2024 Module 1 - 3 (Lecturer: Prof Yasser Abduallah).

Useful Links:

[My GitHub Repository/Midterm Project CS634]

Running the Python Notebook

Prerequisites:
Ensure that Python is installed on your system (version 3.x recommended).
Install the required libraries: pandas, mlxtend, pyfpgrowth. You can install them using pip:

Copy code.
pip install pandas mlxtend pyfpgrowth

Notebook Setup:
Download the Python notebook file (Midterm_Project_CS634_IM.ipynb or Midterm_Project_CS634_IM.py)), and the CSV files containing the item names and transactions for each retailer.
Place the notebook file and the CSV files in the same directory.

Launching the Notebook:
Open a terminal or command prompt and navigate to the directory where the notebook file is located.
Launch Jupyter Notebook by running the following command:

Code
jupyter notebook
Jupyter Notebook will open in your default web browser.

Running the Notebook:
In the Jupyter Notebook interface, click on the notebook file (Midterm_Project_CS634_IM.ipynb or Midterm_Project_CS634_IM.py) to open it.
The notebook will open, displaying the code cells and markdown cells.
To run the notebook, click on the "Run" button in the toolbar or use the keyboard shortcut Shift + Enter to execute each cell sequentially.

Interacting with the Notebook:
When prompted, enter the required inputs such as minimum support, minimum confidence, and the retailer choice.
The notebook will load the corresponding CSV files based on your retailer choice.
The notebook will execute the Brute Force method, Apriori algorithm, and FP-Growth algorithm, displaying the results and association rules.
You can modify the code cells, experiment with different parameters, or add your own analysis as needed.

Saving and Exporting:
To save the notebook, click on "File" in the menu and select "Save and Checkpoint."
To export the notebook as a PDF or HTML file, click on "File" and choose the desired export format.

Thank You