

**Transaction Data Compression Report**

Release Date

16/08/2024

Submission Deadline

04/09/2024

**Introduction**

In today's data-driven world, the exponential growth of transactional data presents significant challenges for efficient storage and processing. This report addresses the critical need for dataset compression, focusing on lossless techniques that ensure the original information can be accurately retrieved without degradation.

Given a large transactional dataset, where each transaction consists of a set of items purchased together, the objective is to optimize the data storage through effective itemset mapping. By employing frequent itemset mining, we can reduce redundancy and achieve substantial storage savings while maintaining the integrity of the data.

For example, by creating strategic mappings of frequently occurring itemsets, we can transform the original dataset into a compressed format that minimizes storage costs. This process not only enhances data management efficiency but also supports quicker access and analysis, which are vital in making informed business decisions.

In this assignment, we will explore various design decisions that affect the compression ratio, and analyze the implications of different mapping strategies to illustrate their impact on data retrieval and storage efficiency.

**Problem Description**

We are provided with a dataset consisting of multiple transactions, each containing a list of purchased items. The transactions are not ordered, meaning the order of items does not affect the significance of the data.

Example Dataset

Consider the following example dataset:

Transaction 1 (T1): A, B, C, D, E

Transaction 2 (T2): A, B, C, D, F

Transaction 3 (T3): A, B, C, D, E, G

Transaction 4 (T4): A, B, C, D, E, F, G

Mapping Example

To compress the dataset, we can create a mapping for certain combinations of items. For instance:

plaintext

Copy

{

X: {A, B, C, D},

Y: {E, G}

}

The new transactions would then be:

T1: X, E

T2: X, F

T3: X, Y

T4: X, Y, F

**Purpose of Mapping**

Using a mapping approach serves several critical purposes in the context of data compression:

Reduction of Redundancy: By mapping frequently occurring itemsets to new identifiers, we reduce the need to repeat the same set of items multiple times across transactions. This streamlining leads to significant savings in storage space.

Improved Readability: The mapped identifiers, such as

X

X and

Y

Y, provide a simpler representation of complex item combinations, improving the readability of the data for analysis and reporting.

Enhanced Processing Speed: With fewer items encoded in the mapping, the time taken to process the dataset during analyses (like querying or generating reports) can be reduced.

Facilitated Data Operations: By grouping items into larger frequent itemsets, the mapping allows for more straightforward operations when it comes to updating, analyzing, or transforming the dataset.

Implementation of Mapping

The function defined below demonstrates how the mapping is created:

python

Copy

def create\_mapping(frequent\_itemsets):

mapping = {}

for idx, item in enumerate(frequent\_itemsets):

if idx % 2 == 0: # Adjust this condition based on your specific requirements

mapping[item] = f'Item\_{idx + 1}'

return mapping

**Storage Metrics**

The storage cost is calculated as the total number of items across all transactions plus the number of keys in the decoder mapping.

Example Calculations

From the initial dataset, we calculate:

Original dataset size: 23

Mapping table size: 8

Compressed dataset size: 9

Therefore, the total storage size is 17 (23 - 17 = 6, resulting in approximately 26% compression).

**Design Considerations**

**Support Count of 2**

The support count is a crucial parameter in frequent itemset mining, representing the number of transactions in which an item or an itemset appears. In this assignment, we use a support count of 2 for the following reasons:

Relevance: A support count of 2 ensures that only itemsets that appear frequently enough are considered for mapping, filtering out infrequent combinations that may not contribute significantly to understanding user behavior or transactional patterns.

Quality of Compression: By enforcing a minimum frequency (support count of 2), we aim to create mappings that result in a more compact representation of the dataset. Frequent itemsets are more likely to be reused across multiple transactions, leading to greater reduction in the total number of items required in the compressed format.

Statistical Significance: A higher support threshold can help ensure that the patterns we identify are statistically meaningful, reducing the noise that may otherwise cloud the dataset.

**How We Get Support Count**

To determine the support count for itemsets:

Transaction Scanning: Each record in the dataset is scanned to count occurrences of all items and itemsets.

Frequent Itemset Generation: Using algorithms such as Apriori or FP-Growth, we identify itemsets that meet or exceed the specified support threshold after processing the frequency table.

Dataset Overview

The provided dataset consists of multiple transactions, each represented by space-separated item names. This report will implement a compression algorithm to efficiently process and compress the dataset using the above principles.

**Algorithm Design**

**Methodology**

The following steps will be taken to accomplish the task:

Data Loading: Read each transaction and store it in a suitable data structure.

Frequent Itemset Mining: Identify frequent itemsets using a suitable algorithm (e.g., Apriori or FP-Growth) with a minimum support count of 2.

Mapping Creation: Develop a mapping of frequent itemsets to new identifiers based on their frequency.

Dataset Compression: Transform the original transactions into the compressed format using the created mapping.

Decompression Method: Implement a method to revert the compressed dataset to its original form for verification of lossless compression.

**Conclusion**

This report outlines a comprehensive methodology for compressing a transactional dataset using a structured mapping approach, leveraging the principles of frequent itemset mining. The core of this technique lies in analyzing item frequency to identify recurring patterns within the dataset, which facilitates a more efficient representation of transactions.

Key Findings

By employing a support count of 2, we ensure that only itemsets that appear in at least two transactions are considered for mapping. This threshold strikes a balance between identifying meaningful associations and avoiding the inclusion of isolated, infrequent itemsets that could clutter the dataset with unnecessary complexity. The rationale behind using this support count lies in several important factors:

Data Relevance: By filtering out infrequent itemsets, we allow for a dataset that better reflects actual purchasing behaviors, thus enhancing the relevance of the stored data.

Quality of Compression: Focusing on frequently occurring itemsets guarantees that our mappings reduce redundancy effectively. The mappings can cover multiple transactions using compact identifiers, leading to significant reductions in storage requirements.

Statistical Integrity: Setting a lower support threshold, while traditionally associated with potential noise, becomes advantageous when combined with mapping. Frequent itemsets that emerge from this process are likely to maintain a statistical significance, providing valuable insights into customer behavior.

Lossless Compression: The compression algorithm preserves the full integrity of the original dataset. Decompression can easily revert the data to its original form, making it invaluable for any retrospective analysis or future predictive modeling.

**References**

Mining Frequent Patterns: A Look Back and Ahead.

Data Compression Techniques in Data Mining.

Used ChatGPT.