Problem formulation

<u>Task Type: Theoretical Data Mining (Performance Evaluation and Algorithm Comparison)</u>

In the discipline of data mining, sequential pattern mining is a vital task that focuses on identifying frequently observed patterns, such as item sets, subsequences, or substructures, in a given dataset (database). Efficient and accurate (precise) algorithms are crucial for ascertaining maximum performance in a variety of applications, including customer purchase behavior analysis, online (web) access pattern mining, as well as bioinformatics. The goal of this research is to assess and contrast the time complexity of four widely used sequential pattern mining algorithms—Brute Force, SPADE, PrefixSpan, and SPAM—by systematically changing the hyperparameters of randomly generated datasets. The purpose is to determine which algorithm performs best in terms of time efficiency under diverse conditions, offering significant insights for researchers and practitioners in choosing the most effective sequential pattern mining algorithm for their individual needs.

Discussion of related works – Literature Review

The investigation of Sequential Pattern Mining Algorithms has been a vast area of research in the community of data mining. A considerable number of various algorithms have been suggested and analyzed to recognize the most effective techniques for mining frequent sequences in large databases (datasets).

The Brute Force method is a simple approach to detecting frequent patterns that perform by generating all possible subsequences and computing their frequencies to later contrast them with the minimum support threshold value. This method, however, has high computational complexity and low efficiency, particularly for large datasets (Slimani & Lazzez, 2013).

Zaki (2001) proposed the SPADE (Sequential PAttern Discovery using Equivalence classes) algorithm as a viable alternative to the Brute Force approach. SPADE employs a vertical database structure, recording sequence occurrences with an efficient data format known as id-lists. This enables quick frequency computation and significantly lowers the search space. Furthermore, SPADE uses lattice-based approaches and an equivalence class-driven search to find common sequences with less time complexity than the Brute Force method.

Another notable sequential pattern mining algorithm introduced by Pei et al. (2001) is PrefixSpan. As Zaki's (2001) SPADE approach, PrefixSpan was also developed in a strive for compensating for the lack of performance of the Brute Force method. The pattern-growth mechanism, which avoids the need to construct candidate sequences explicitly, is central to PrefixSpan. Instead, it recursively divides the database into smaller projected databases, each corresponding to a different prefix. This method decreases computational complexity and provides a scalable option for mining huge datasets (Pei et al., 2001).

SPAM (Sequential PAttern Mining) is a depth-first search technique that combines the best characteristics of vertical and horizontal data formats. SPAM, similarly, to SPADE, uses id-lists for efficient frequency computation. However, it also includes PrefixSpan's pattern-growth method to improve its performance (Slimani & Lazzez, 2013). SPAM is an appealing choice for mining sequential patterns in a variety of application fields attributed to its innate hybrid technique implementation.

In conclusion, the literature emphasizes the constraints of the Brute Force algorithm, as well as the emergence of more efficient algorithms for mining frequent sequences, such as SPADE, PrefixSpan, and SPAM. While each of the algorithms utilizes its own unique strategy to enhance and optimize the sequential pattern mining process, the question of their comparative performance under different circumstances seems to remain open. Thus, this paper is aimed at addressing this gap by conducting a comprehensive comparative analysis of the performance of these algorithms under varying hyperparameters of randomly generated datasets.

EDA and data preprocessing

Dataset Description

For the sake of this project, various random datasets of the same structure but with varying hyperparameters were generated for further being used by different algorithms during the comparative analysis. A general, detailed overview of the structure of each of the datasets used for the experimental part, as well as their corresponding varying parameters, are provided below.

Each row in the database can be viewed as a separate customer. In addition, each row represents the sequence of transactions, each of which is an itemset. Before proceeding to the example dataset, the way of generating random datasets is to be discussed. The function generate(num_sequences, max_sequence_length, num_items) accepts 3 arguments where num_sequences is the number of customers, max_sequence_length is the total number of items for each customer, num_items is the number of different items. For instance, for the example dataset provided below, the hyperparameters are num_sequences = 3, max_sequence_length = 9, num_items = 8.

Example dataset:

Dataset = [{1, 2, 4}, {2, 3, 4}, {2, 3, 4}],
[{2, 7, 8}, {1, 2, 3}, {4, 5},{2}],
[{1, 2}, {2, 3, 4}, {1,3,5,7}]

By adjusting the hyperparameters for our needs, a further assessment of the time performance of the algorithms was implemented.

Data Preprocessing

1. SPADE

   Prior to successfully implementing the SPADE algorithm, the initial dataset was converted into a vertical database representation as described in Zaki's book (2020). The SPADE algorithm employs a vertical database representation for sequence mining. In this approach, each symbol is associated with the sequence identifiers and their positions within those sequences. For every symbol s in the database, we maintain a set of tuples, represented as ⟨i, pos(s)⟩, where pos(s) denotes the positions of the symbol s in the sequence s_i in the database (Zaki, 2020).

2. SPAM

   To facilitate efficient counting, the implementation of SPAM employs a vertical bitmap representation of the data as described in Ayres et.al (2002) work. A distinct vertical bitmap is generated for each item in the dataset, with each bitmap containing a bit that corresponds to every transaction in the dataset. If item i is present in transaction j, then the bit representing transaction j in the bitmap for item i is assigned a value of one; otherwise, the bit is assigned a value of zero (Ayres et.al., 2002).

3. PrefixSpan

   Since the generated datasets were already in the required format (horizontal database representation) for the PrefixSpan algorithm, no further steps needed to be done throughout its implementation (Pei J. et.al., 2001).

Preprocessing Evaluation

The transformation of the initial datasets' representation facilitates the effective performance of algorithms while almost requiring no time for the conversion process in contrast with the overall time needed for the algorithms to be executed successfully.

Mining

Since the main objective of this project was to assess the time performance of various Sequential Pattern Mining algorithms, 3 of the most famous and efficiently known approaches to Sequential Pattern Mining were implemented manually for further experiments to answer the ultimate research question of the comparative analysis of the time performance of the methods. In addition to that, to the end of efficiently visualizing the observed data, various machine learning algorithms, such as polynomial and exponential fitting, were implemented to fit the graphs to the data.

Algorithms

1. SPADE
2. SPAM
3. PrefixSpan

<u>Remark: all the algorithms were thoroughly tested and verified on small datasets before being applied to larger ones. Then, for better conduction of the Average Case Time Analysis, the output was discarded.</u>

Experiments.

Average Time Case Analysis

To objectively answer the research question of the paper by conducting a series of average case analyses, various datasets were created by adjusting the hyperparameters accordingly. Each of the trials is discussed in detail below. Note that the relative support level used for the experiments is 0.8 (80%).

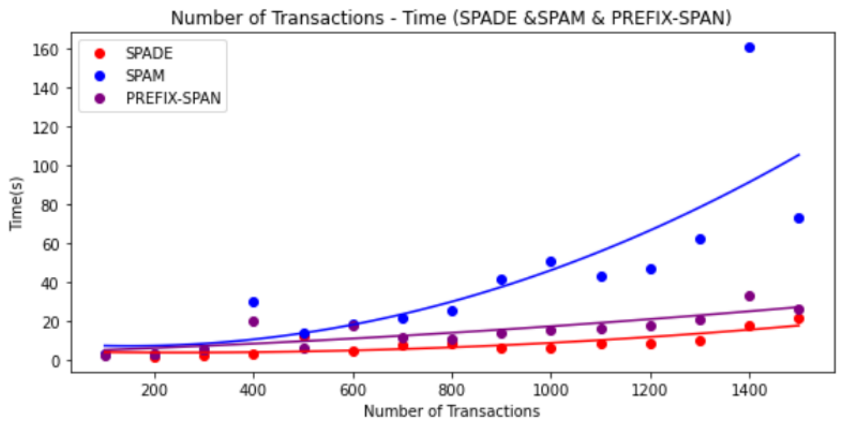
Case 1:

num_sequences = 100 - 1000 (with a gap between trials of 100)

max_sequence_length = 20 (fixed)

num_items = 5 (fixed)

Figure 1 (Polynomial Fit):



Overall, the algorithms seem to have demonstrated to be polynomially dependent in terms of the number of transactions. It can be observed that as the number of transactions increases, the SPADE and PrefixSpan algorithms appear to significantly outperform SPAM. Whereas SPADE seems to slightly outperform PrefixSpan.

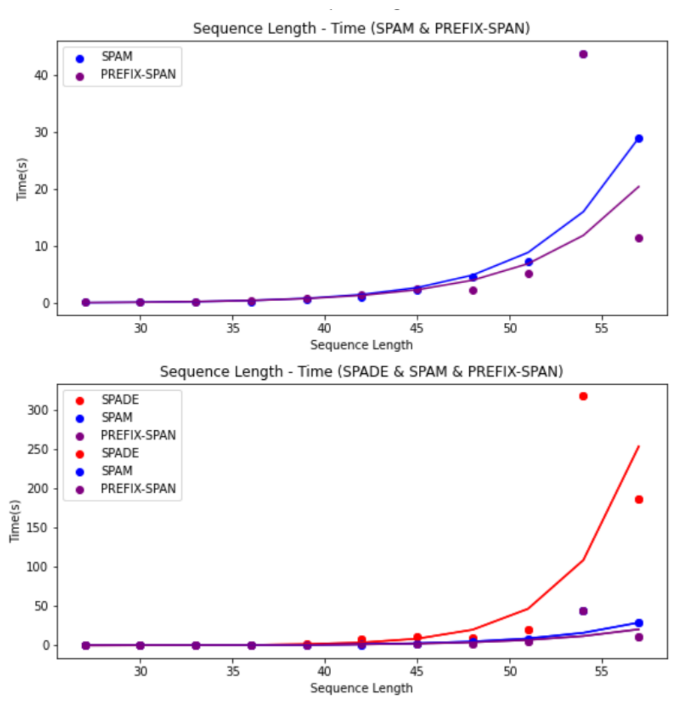
Case 2:

num_sequences = 100 (fixed)

max_sequence_length = 6 - 60 (with a gap between trials of 6)

num_items = 20 (fixed)

Figure 2 (Exponential Fit):



Overall, the algorithms seem to have demonstrated to be exponentially dependent in terms of the sequence length. It can be observed that as the length of the sequences increases, the SPAM and PrefixSpan algorithms appear to significantly outperform SPADE. Whereas PrefixSpan and SPAM perform very similarly.

Case 3:

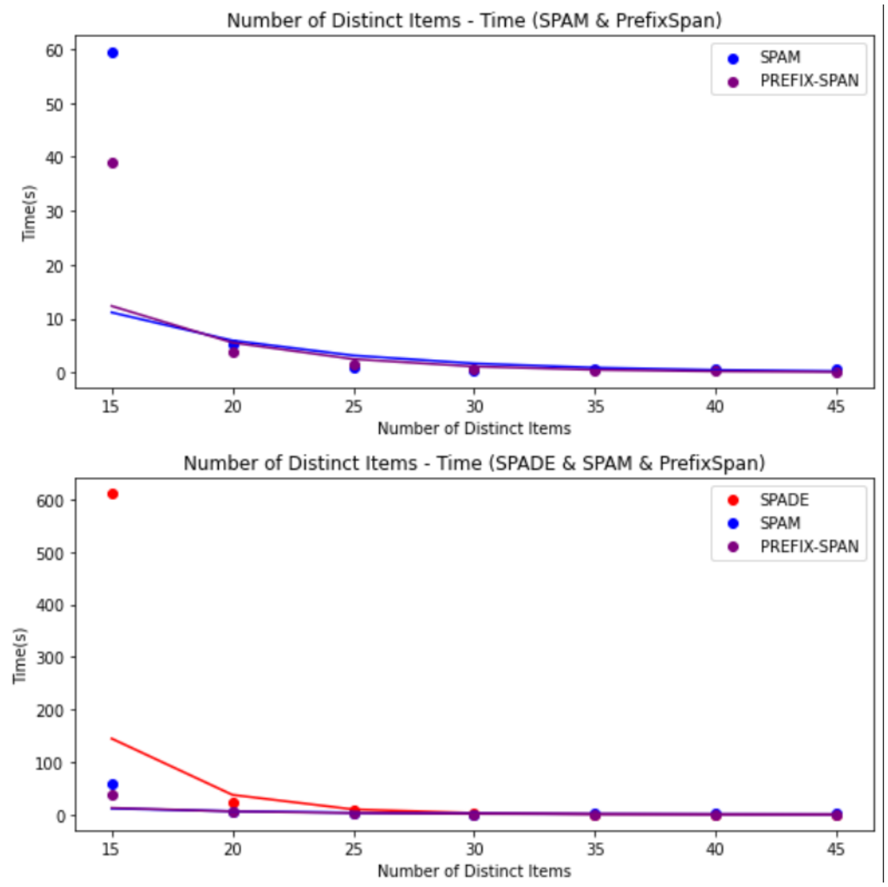
num_sequences = 100 (fixed)

max_sequence_length = 20 (fixed)

num_items = 20 - 200 (with a gap between trials of 20)

Remark: specifically for this case, the minsup value was taken to be 0.1.

Figure 3 (Exponential Decay Fit):



Overall, the algorithms seem to have demonstrated to be exponentially (decay) dependent in terms of the number of distinct items. It can be observed that as the number of distinct items decreases, the SPAM and PrefixSpan algorithms appear to significantly outperform SPADE. Whereas PrefixSpan and SPAM perform very similarly.

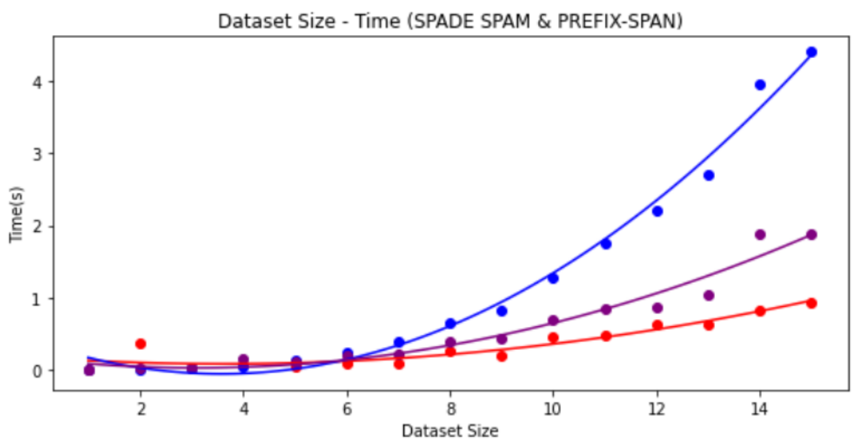
Case 4 (Polynomial Fit):

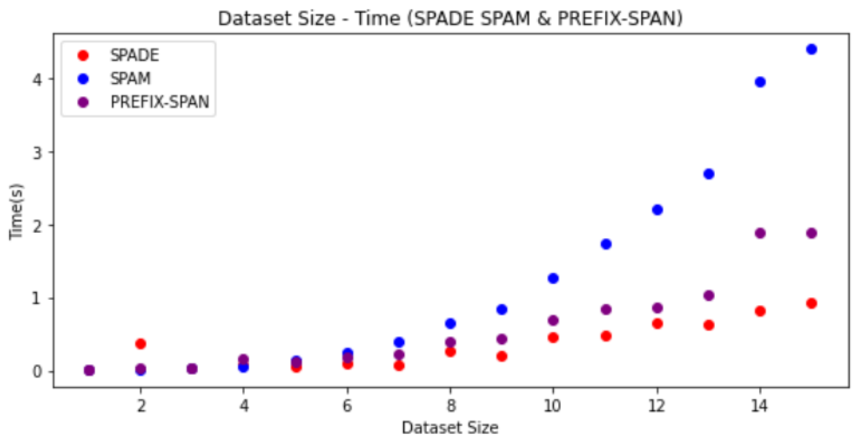
num_sequences = 25 - 250 (with a gap between trials of 25)

max_sequence_length = 4 - 40 (with a gap between trials of 4)

num_items = 3 - 30 (with a gap between trials of 3)

Figure 4:





It can be observed that as the size of the dataset increases, SPADE and PrefixSpan seem to considerably outperform SPAM. Whereas SPADE considerably outperforms PrefixSpan.

Note:

It is important to mention that case 4 is highly sensitive to hyperparameters being changed. Unlike the previous cases where merely 1 hyperparameter was fixed (others were varying), which enabled us to precisely predict the behavior of the model (the overall performance tendency was consistent), for case 4, since all hyperparameters are changed in parallel, depending on a particular setup, the average time performance of the algorithms can drastically be affected.

Discussion of results

The results of the experiments have indicated that the algorithms for sequential mining are polynomially dependent on the number of transactions i.e., customers (Case 1), exponentially dependent on the number of items in each sequence (sequence length) (Case 2), as well as possess the exponential decay dependency on the number of distinct items (Case 3). As discussed in the previous part, different algorithms perform differently under varying conditions. Thus, the choice of the best sequential pattern mining algorithm converges to choosing the appropriate one for the given dataset and its hyperparameters based on the performance they have demonstrated. With that said though, out of all algorithms, PrefixSpan has demonstrated the most stable and effective performance in all the tested cases. While a significant amount of effort was put into the research in the face of the lack of time, the programming language choice (Python is slow), as well as relevant information and libraries, better implications should be made when increasing the hyperparameters; that is, dealing with larger datasets, which we are planning to maintain in our future studies. We are also planning to add more algorithms, re-write the already implemented code in C++, and provide better optimization approaches for them, in order to establish a library with stable algorithm.

Remark: SPMF Java Framework has been of great help to our team as a reference model. It is important to mention that our algorithms seem to have indicated more precise results if compared to SPMF.

Bibliography

1. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2001). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Transactions on Knowledge and Data Engineering, 16(11), 1424-1440.

2. Slimani, T., & Lazzez, A. (2013). Sequential Mining: Patterns and Algorithms Analysis. International Journal of Computer Science Issues, 10(1), 164-171.

3. Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning, 42(1/2), 31-60.

4. Mohammed, Z., & Meira, W. (2020). Data Mining and Machine Learning Fundamental Concepts and Algorithms. Cambrige: Cambrige University Press.

5. Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential PAttern Mining using a Bitmap Representation. ACM, 429-435.

6. https://github.com/jacksonpradolima/SPMF/tree/master/src/ca/pfv/spmf/algorithms/sequentialpatterns