Sequential Pattern Mining with SPADE Algorithm on Online Retail Dataset

GitHub Repository Link – https://github.com/JayaAadityaaG/DM_FinalAssign

ABSTRACT

Sequential pattern mining is done on online retail dataset. Preprocessing is conducted in order to convert data into vertical data format. Visualization is done to explore trends in the datset. SPADE algorithm is applied and the obtained patterns are analyzed.

I. Introduction

Sequential pattern mining is an essential subset of data mining and is a powerful tool to extract patterns. Each sequence consists of a series of elements each containing their set of events. Sequential pattern mining allows to examine a group of sequences and determine frequently occurring subsequences. The database of purchase history for customers of an online retail website is a classic sequential pattern mining problem. The Sequential Pattern Discovery using Equivalent Class (SPADE) algorithm will be used in this project.

The purchase history of each customer corresponds to a single sequence, with each order and set of items corresponding to element and set of events respectively. The general overall objective in this particular problem is to determine frequent sequences in purchases of items which imply a high probability for the purchase of a subsequent item.

A. Dataset Description

An online retail dataset provided by the UCI Machine Learning Repository is the target dataset. It consists of transactional data for a UK based company over the duration of a year (01/12/2010 - 09/12/2011). There are 8 attributed namely – InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID and country. The dataset consists of 541909 instances arranged in ascending order with respect to InvoiceNo.

In order to apply the SPADE algorithm, this dataset must be converted to vertical formatting which consists of Sequence Number, Event Number and Sequence (Item Set). For this problem, the sequence number is determined on the basis of customer ID. By sorting through the purchases of each customer based on Invoice Number which is allocated based on the time of transaction, the event number can be determined. Finally, the list of items purchased in a given order or 'ItemSet' is determined by using the above generated values and combining those items with the same sequence and event number.

B. Development Tools

The source code for this project will be implemented using Python3. In order to do so, Jupyter Notebooks (Powered by Anaconda) as it allows us to independently execute different sections of code. This proves to be extremely useful for Data Mining problems. The primary libraries of Python3 used are pandas, numpy, matplotlib.pyplot, copy and datetime.

II. Preprocessing

Preprocessing is an important aspect of this problem due to the large number of instances. The preprocessing techniques used include – Data Cleaning, Data Reduction and Data Transformation. The end goal of preprocessing is to develop a dataset with vertical data formatting and storing the remaining required information with suitable methods.

A. Data Cleaning

Data Cleaning plays an integral role in this analysis. After examining the dataset manually and using a program, it is observed that only two attributes of interest contain missing values, namely – 'Description' and 'CustomerID'. The 'CustomerID' attribute is extremely important as it determines which sequence number to allocate each instance to. Hence if any of the data points have a missing value in 'CustomerID', there is no choice but to eliminate these points as filling in any estimate will lead to erroneous results. Hence all instances with empty 'CustomerID' attribute are eliminated. It is observed that in this modified dataset there are no instances with empty 'Description' as they had all corresponded to empty 'CustomerID' values.

B. Data Reduction

Attribute Subset Selection is done in order to optimize the data for the given objective. Firstly, the attributes 'Quantity', 'UnitPrice' and 'Country' are removed. Despite being important characteristic for different problems, these attributes do not provide additional utility to the task of sequential pattern mining at hand and are hence irrelevant features. Some of the remaining attributes are dropped later into the program depending upon newly constructed features. For example, the attributes 'Description' is a redundant attribute as it contains the same information as that of 'StockCode'. Hence it is removed after suitable operations, which will be elaborated in feature transformation.

C. Data Transformation

A number of features were manipulated in order to obtain improved features: -

i. *Cancellation:* - An invoice number with 'C' as the initial term refers to an order cancellation. Since the ordering of Invoice Numbers is important for future manipulations, this cancellation information is transferred to the 'StockCode' attribute. This was done by removing the 'C' from all invoice numbers and adding 'CN' to the beginning of the corresponding 'StockCode'.

ii. *Time* – The date of purchase in 'InvoiceDate' of all instances is given in dd/mm/yy H:M format which is not convenient to manipulate. Using the first 'InvoiceDate' as the reference, a new feature 'Time Difference' is constructed which gives the time difference in days. This feature is later condensed to correspond to the vertical format.

iii. *Sequence Number* – Lists are created which contain the set of 'CustomerID', 'StockCode' and 'Description' values present in the dataset. Since 'StockCode' and 'Description' have a one-to-one relationship, 'Description' is dropped as these created lists can serve as a dictionary. With the use of the 'CustomerID' list, all customer ID's are allocated a sequence number. A new feature, 'Sequence Number' is the constructed

using this. The database is then sorted according to 'Sequence Number' and the attribute 'CustomerID' is dropped.

iv. *Event Number* – The 'Event Number' attribute is generated by sorting the values using 'InvoiceNo' for the set of instances with same 'Sequence Number'.

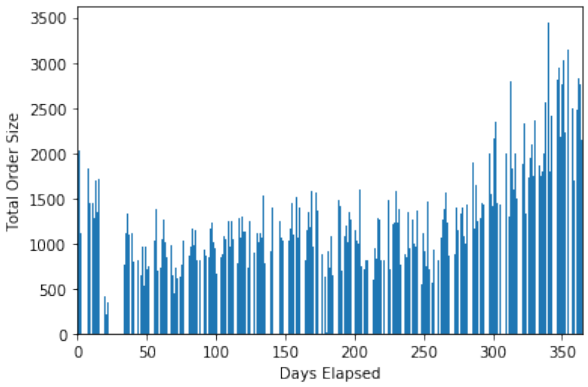
v. *ItemSet* - In order to perfectly obtain the vertical format, the instances with the same 'Sequence Number' and 'Event Number' are aggregated to determine the desired sequence.

III. Visualization

The original database for online retail without any pre-processing can be used to develop a number of visualizations for this data set. However, since the primary objective is the application of the SPADE algorithm, the visualizations are approached accordingly. In the visualizations, the pre-processed data, i.e. the data represented in vertical format is used. 'Quantity' value as given in original dataset of each item is not considered as it would bias the plotted values if even a small number of customers purchased a large quantity of the same item in a single purchase. By directly using only the 'ItemSet value, this skew can be relatively compensated. Hence the following graphs have their values represented by only interacting with the preprocessed data frame.

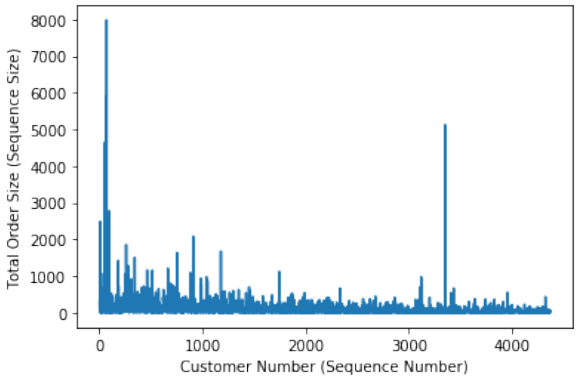
A. Total Order Size vs Days Elapsed

The following plot is obtained by totaling the size of all 'ItemSet' values which occurred at the same time. The day number is plotted using 'Time Difference' values which is the duration elapsed since the occurrence of the first order in terms of days. Essentially, a graph which shows the trend of number of orders per day is obtained. The sudden spikes in 'Total Order Size' per day can be attributed to the presence of upcoming holidays. For instance, towards the end of the year, there is a significant increase in number of orders due to the arrival of holidays such as Christmas, Thanksgiving and New Year's.



B. Total Order Size vs Customer Number

The following plot shows the total order size which is essentially the size of the sequence, plotted against the sequence number. From this, it can be observed that most sequences have a size between 0-1000. There a few outliers which lie close by to the mean values, but there are also a handful outliers which have a size significantly larger than that of the other sequences. This gives us an important insight into the selection of the support counting method to be used for the algorithm.



V. SPADE Algorithm

The SPADE (Sequential PAttern Discovery using Equivalent Class) algorithm is an important sequential pattern mining algorithm. It requires the usage of a vertical data format for its dataset of sequences. However this significantly enhances the performance of the program as compared to GSP (Generalized Sequential Patterning).

A significant amount of pre-processing was conducted to obtain the vertical format which comprise of 'Sequence Number', 'Event Number' and 'ItemSet'. The 'Time Difference' attribute was obtained to in order to enable the application of timing constraints if necessary. However, since the span of this entire database is a year and it is online retail data, the timing constraints are ignored.

A. Generation of frequent 1-subsequences

A scan of the vertical format data is done to obtain the list of all items and their corresponding frequency. Using a minimum support of 1000, 17 items are obtained as members of the frequent 1-subsequences. It is necessary to keep this support threshold high so as to not obtain a large number of frequent 1-subsequences. This will results in a significant increase in the time consumed. Hence by using a threshold of 1000, 17 frequent 1-subsequences are obtained. For each of these frequent 1-subsequences, it is necessary to know the positions (Sequence Number, Event Number) of all their occurrences. This is done with the usage of a 3D list. Each 2D list which is a member of the 3D list corresponds to a particular frequent 1-subsequence and contains all the Sequence Number and Event Number positions it occurs (position table). Hence by knowing the index of the element in the frequent 1-subsequnce list, the frequency and position table for that 1-subsequence can be obtained by using the corresponding lists.

B. Generation of frequent 2-subsequences

The list of frequent 2-subsequences will play the base case for the generation of all future frequent subsequences. A series of nested loops are used in order to generate the frequent 2-subsequences. Using the first two loops, 2 different frequent 1-subsequnces are accessed. Using the next 2 loops, all of their positions are compared and checked if they satisfy the SPADE algorithm rules. In order to satisfy the SPADE algorithm rules, the sequence numbers need to match and the event numbers need to follow the ordering of desired subsequence. If these rules are satisfied, the corresponding 2-subsequence is generated and added to the list of 2-subsequences. If it already exists, then its frequency in the frequency list is increased accordingly. Simultaneously, another 3D list is developed in order to hold the list of positions of all these 2-subsequnces. Following this, the same procedure is conducted by comparison of position table for the same element to generate 2-subsequnces which can be constructed using a same element. This is done separately as the rules followed while comparing different and same elements vary and hence combining will result in irregular values. The 2-subsequnces generated from this procedure, their frequency and their position table is appended to the list of 2-subsequences. Following this, pruning is conducted with the usage of a support threshold of 200. This is lower than the previously used minimum support value as the occurrences of different size subsequences vary significantly. No results would be obtained if the same high threshold used for 1-subsequnces is used. Using this threshold, 35 2-subsequences are obtained.

The COBJ method for support counting is used (at least one occurrence of a given sequence in an object's timeline). This is to ensure the purchase history of a single customer does not

skew the data in a biased manner. It is seen from the plots that there are some outliers with a large number of purchases who can potentially skew the data. To prevent this, the COBJ method is used. In addition to this, since the duration of the dataset spans a year, it is logical to go for the COBJ method.

C. Generation of frequent n-subsequences

(n+1) frequent subsequences are generated by using n-subsequences and the process is initiated using frequent 2-subsequences. The loop begins by taking a frequent n-subsequence list. 2 different subsequences from this list are taken and initially compared to check if they can be combined. This is done by comparison of subsequences generated by removal of first and last element of first and second subsequence respectively. Following this, the sequence numbers are compared to check if they are same. If they are, the order of event numbers is then compared to check if they can be combined. If this test is satisfied the subsequences are combined to generate a (n+1) subsequence. This is added to the list of (n+1) subsequences and the frequency count and position table is added accordingly. After generation of all possible (n+1)-subsequences, pruning is done by using a threshold support of 100. All pruned n-subsequences are added to the final list of frequent subsequences. This process is iterated until not frequent subsequences can be found. Following this, all frequent subsequences with 2 or more elements are converted from 'StockCode' representation to 'Description' representation. In total, 42 frequent subsequences are obtained.

VI. Results

The following frequent subsequences are obtained: -

S.No	Subsequence
1.	[['PARTY BUNTING', 'SPOTTY BUNTING']]
2.	[['JUMBO BAG RED RETROSPOT'], ['LUNCH BAG RED RETROSPOT']]
3.	[['LUNCH BAG RED RETROSPOT', 'JUMBO BAG RED RETROSPOT']]
4.	[['LUNCH BAG RED RETROSPOT'], ['LUNCH BAG BLACK SKULL.']]
5.	[['LUNCH BAG BLACK SKULL.'], ['LUNCH BAG RED RETROSPOT']]
6.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG BLACK SKULL.']]
7.	[['LUNCH BAG RED RETROSPOT'], ['LUNCH BAG SPACEBOY DESIGN ']]
8.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG SPACEBOY DESIGN ']]
9.	[['LUNCH BAG RED RETROSPOT'], ['LUNCH BAG SUKI DESIGN ']]
10.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG SUKI DESIGN ']]
11.	[['LUNCH BAG RED RETROSPOT'], ['LUNCH BAG DOILEY PATTERN ']]
12.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG DOILEY PATTERN ']]
13.	[['JUMBO BAG RED RETROSPOT'], ['JUMBO BAG DOILEY PATTERNS']]
14.	[['JUMBO BAG DOILEY PATTERNS'], ['JUMBO BAG RED RETROSPOT']]
15.	[['JUMBO BAG RED RETROSPOT', 'JUMBO BAG DOILEY PATTERNS']]
16.	[['LUNCH BAG BLACK SKULL.', 'LUNCH BAG SPACEBOY DESIGN ']]
17.	[['LUNCH BAG BLACK SKULL.', 'LUNCH BAG SUKI DESIGN ']]
18.	[['LUNCH BAG SPACEBOY DESIGN ', 'LUNCH BAG SUKI DESIGN ']]
19.	[['JUMBO BAG DOILEY PATTERNS', 'LUNCH BAG DOILEY PATTERN ']]
20.	[['WHITE HANGING HEART T-LIGHT HOLDER'],['WHITE HANGING HEART T-LIGHT HOLDER']]

21.	[['ASSORTED COLOUR BIRD ORNAMENT'], ['ASSORTED COLOUR BIRD ORNAMENT']]
22.	[['SET OF 3 CAKE TINS PANTRY DESIGN’],['SET OF 3 CAKE TINS PANTRY DESIGN ']]
23.	[['PARTY BUNTING'], ['PARTY BUNTING']]
24.	[['REGENCY CAKESTAND 3 TIER'], ['REGENCY CAKESTAND 3 TIER']]
25.	[['SMALL POPCORN HOLDER'], ['SMALL POPCORN HOLDER']]
26.	[['SPOTTY BUNTING'], ['SPOTTY BUNTING']]
27.	[['LUNCH BAG RED RETROSPOT'], ['LUNCH BAG RED RETROSPOT']]
28.	[['JUMBO BAG RED RETROSPOT'], ['JUMBO BAG RED RETROSPOT']]
29.	[['LUNCH BAG BLACK SKULL.'], ['LUNCH BAG BLACK SKULL.']]
30.	[['LUNCH BAG SUKI DESIGN '], ['LUNCH BAG SUKI DESIGN ']]
31.	[['JUMBO BAG DOILEY PATTERNS'], ['JUMBO BAG DOILEY PATTERNS']]
32.	[['LUNCH BAG DOILEY PATTERN '], ['LUNCH BAG DOILEY PATTERN ']]
33.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG BLACK SKULL.']]
34.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG SPACEBOY DESIGN ']]
35.	[['LUNCH BAG BLACK SKULL.', 'LUNCH BAG SPACEBOY DESIGN ']]
36.	[['LUNCH BAG RED RETROSPOT','LUNCH BAG BLACK SKULL'],['LUNCH BAG BLACK SKULL']]
37.	[['LUNCH BAG RED RETROSPOT', 'LUNCH BAG SUKI DESIGN '], ['LUNCH BAG SUKI DESIGN ']]
38.	[['JUMBO BAG RED RETROSPOT'], ['JUMBO BAG DOILEY PATTERNS'], ['JUMBO BAG RED RETROSPOT']]
39.	[['JUMBO BAG RED RETROSPOT'], ['JUMBO BAG DOILEY PATTERNS'], ['JUMBO BAG DOILEY PATTERNS']]
40.	[['LUNCH BAG BLACK SKULL.', 'LUNCH BAG SUKI DESIGN '], ['LUNCH BAG SUKI DESIGN ']]
41.	[['LUNCH BAG SPACEBOY DESIGN ', 'LUNCH BAG SUKI DESIGN '], ['LUNCH BAG SUKI DESIGN ']]
42.	[['JUMBO BAG DOILEY PATTERNS', 'LUNCH BAG DOILEY PATTERN '], ['LUNCH BAG DOILEY PATTERN ']]

VII. Inferences

A brief overview of the obtained frequent subsequences shows that a significant portion of the patterns for the given online retail site is observed for the purchase of bags. Analysis of S2-S14 show that a customer who purchases one bag is quite likely to purchase another bag in a subsequent order. S15-S19 and S33-S35 shows the likely combination of lunch bags or jumbo bags bought together. The patterns show that a customer buying a bag of type lunch bag or jumbo bag are more likely to buy the second bag of the same type. S20-S32 show us patterns of various items which are bought multiple times by the same customer in different orders. This shows that these items could be popular, useful or fragile necessitating a repeating order. From S36-S42, we observe patterns involving 3 items. It is seen that a customer buying 2 lunch bags or jumbo bags will most likely buy another bag of the same type in a subsequent order. All the patterns show exactly which type of each item is most likely to be bought with an another. But by analysing the overall set of patterns, general inferences can be reached regarding the general category of the item. Hence the application of the SPADE algorithm has generated the above patterns for the given online retail dataset.