Sequential Pattern Generalization for Mining Multi-source Data

Julie Bu Daher

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Sequential Pattern Generalization for Mining Multi-source Data

THÈSE

présentée et soutenue publiquement le **10 Décembre 2020**

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Julie BU DAHER

Composition du jury

Présidente :	Isabelle DEBLED-RENESSON	Professeur, Université de Lorraine
Rapporteurs :	Marie-Hélène ABEL	Professeur, Université Technologique de Compiègne
	Nicolas LACHICHE	Maître de conférences, Université de Strasbourg
Directrice :	Armelle BRUN	Maître de conférences, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Sommaire

Table des figures

Introduction

1 General context

If there is one thing in the world that is too big to ignore, it is the creation of enormous amounts of data across years. Every single day, over 2.5 quintillion bytes of data are being created. Data has become important and indispensable in our everyday life, where 90% of the world's data has been created in the last two years [1], and it is estimated that 1.7 mega bytes of data will be created every second for each human being on the planet by the year 2020 [2]. This explosive growth of produced data volume is a result of the computerization of our society and the fast development of powerful data collection and storage tools [23], and this is what makes our age the data and information age and thus, generally, the digital age. Digital data exists in all research and business domains ; examples of these domains are e-commerce, healthcare, banking, education, etc.

In addition, these data can be easily collected where they can be of great volumes, and the variety of these collected data is as important as their volumes. Indeed, data can be collected from multiple data sources where each data source can provide one or more various kinds of data. Therefore, these various multi-source data form heterogeneous multi-source datasets. For example, in the domain of e-commerce, the data about customers can include their descriptive data, their past purchases, their feedbacks about their purchases, etc.

When these data are collected, they become available for analytics. Data analytics is the science of examining raw data with the aim to draw conclusions from the information they contain [3]. Data analytics technologies and techniques have become more and more important across time and are now widely used. Researchers and scientists are aware of the importance of data analytics in order to obtain valuable information that helps in proving or disproving their scientific hypothesis. Data analytics is also important in commercial industries as it helps organizations in taking informed and business decisions ; for example, it can help business increase their revenues or improve their efficiency and performance. In this frame, understanding customers' data is an indispensable first step in order to take decisions such as predicting future purchases of customers or recommending them purchases upon their needs and preferences. There exist different approaches to analyze and understand the data and draw conclusions from the information they contain. Choosing a suitable approach depends on the nature of data as well as on the purposes of these analyses and the desired information to be obtained.

One of these approaches is Knowledge Discovery in Databases (KDD) that has been first defined by Fayyad et al., [13] as the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data. KDD can be generally divided into three main

1. www.sciencedaily.com retrieved on December 10^{th} , 2019
2. https ://www.domo.com retrieved on December 10^{th} , 2019
3. https ://www.lebigdata.fr/definition-quest-data-analytics, retrieved on December 10^{th}, 2019

steps : pre-processing, data mining and post-processing [23]. The pre-processing step filters and prepares data by removing irrelevant, unreliable, redundant or noisy data. The post-processing ensures making the results understandable and then evaluates these results according to the requirements and the purposes. The data mining step aims at extracting knowledge from large datasets. These large datasets can be found in various kinds of databases such as relational databases, transactional databases, data warehouses and other data repositories. Data mining is the most challenging step of KDD process and is essential for many analytics tasks. It is the main focus of our research study. Data mining is defined as the process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [13].

There are three main techniques in data mining : classification [11], clustering [30] and pattern mining [2, 23]. Classification is defined as the process of finding a model (or function) that describes and distinguishes data classes or concepts. The model is derived from the analysis of a set of class labeled data. The model is used to predict the class label of objects for which the class label is unknown [23].

Unlike classification, which analyzes class-labeled objects, clustering analyzes objects without consulting class labels. Clustering is defined as the creation of groups or clusters of similar objects that are dissimilar to those of other clusters. Clustering methods are particularly useful for exploring the interrelationships among the data objects from the dataset and are one way of assigning class labels to the data objects when no such information is available or known [20].

Frequent pattern mining aims at finding patterns that are relevant and frequent in the data. Frequent pattern mining, with its different techniques, has grown to become an important topic across time where various works were proposed in these domains. The pattern mining process as well as the frequent patterns generated from this process are interpretable which would help in taking interpretable decisions such as recommendations. For example, pattern mining can be used in the healthcare domain where data scientists or researchers aim to analyze and understand the data about patients' symptoms, the diagnoses made by doctors based on these symptoms and the treatments prescribed by doctors to patients according to these diagnoses. Based on these data, they can extract frequent patterns where a frequent pattern contains certain symptoms, their diagnosis and their treatment. These patterns thus allow them to recommend relevant treatments for patients according to their symptoms.

Sequential pattern mining is one of the various types of frequent pattern mining. It is concerned in finding frequent patterns in datasets whose items are organized in the form of sequences where a sequence is an ordered list of items or itemsets. Sequential pattern mining is a topic of high interest as sequential data exists in many application domains such as e-learning where sequential data can represent the sequences of pedagogical resources (exams, quizzes, exercises, etc.) that the students consult on their virtual learning environments (VLE) ; an example of such kind of student sequence is as follows : $Student\text{-}143 : < R_3 R_8 R_{13} R_{27} R_{29} >$ where $Student\text{-}143$ represents the id of a student and R_n represents the id of a pedagogical resource. The sequence means that the student has consulted the resources R_3, then R_8, then R_{13}, then R_{27} and finally R_{29}. A frequent pattern extracted from this sequence can be : $P : < R_{13} R_{27} R_{29} >$. By extracting frequent patterns of pedagogical resources of students, we can form rules to recommend resources to the students based on their past activity. For example, based on the frequent pattern p, if a student has already consulted the resources R_{13} and R_{27}, then we can recommend him/her the resource R_{29}.

As previously mentioned, a dataset can contain several types of data which can come from multiple data sources. Data can be sequential and can for example represent data sequences of transactions performed by the users on their system ; an example of this data is the product purchases performed by customers from an e-commerce website. Data can also represent des-

criptive data about users ; an example of this data is the demographic data of users such as their age, gender and address. Data can also represent descriptive data about items. Considering the example of customer purchases from an e-commerce website, the descriptive data about the products purchases by customers could be product name, brand and expiry date. These various data sources, put together, form a complex and a heterogeneous data set. Despite its complexity and heterogeneity, such a dataset is rich, and we assume that, if accurately mined, it allows providing rich information which makes the mining of such a dataset a challenge.

2 Problem statement

The general objective of this PhD thesis is the mining of frequent patterns from multi-source datasets. The data to be mined can be complex, heterogeneous and can contain various kinds of data. Data can be sequential (or temporal) or descriptive and can occur in different forms such as numerical, categorical, textual or graphical forms. Our applicative objective is analyzing and understanding the behavior of users. The data used to perform this analysis can represent several points of view of this behaviour, can come from multiple data sources and can be of various kinds. By understanding this data, we aim to predict the behaviour of users and provide them with recommendations in a personalized manner. For this sake, **we focus on mining sequential patterns from the multi-source data** in order to extract frequent sequential patterns out of users' behavioural data.

When the data is multi-source, if we limit the mining process to the mining of only one source, naturally the behavioral one, we consider that the results can be restricting for two main reasons. First, the patterns represent the data that are mined ; therefore, when only one data source among the multiple data sources is mined, the patterns will only contain information from this single source thus representing a unique point of view. Obviously, when additional sources are available, the mined patterns may contain information from these sources and will probably be richer. For example, when we uniquely manage the sequential data source, we would only understand the digital behaviour of users, and thus we would provide recommendations to the users based only on this information. However, this information may not be sufficient to provide reliable and well personalized recommendations. Therefore, adding descriptive information about the users and/or the items of the data sequences may result in a better understanding of the digital behaviour of users and thus in richer and more specific recommendations.

The second reason is that even when the amount of data is big, in certain cases we may face a problem of lack of data either caused by the data sparsity, by the variability of human behavior or by the similarities among the data which will be further explained in the following chapters. These similarities can be on the level of items of the sequences where we can find certain items that are similar to others, and they can be on the level of patterns where certain patterns are similar to others. Therefore, there could be patterns which could not be detected as being frequent due to the similarities among them. This problem leads to a decrease in the number of the generated sequential patterns and thus a lower data coverage. In this case, the extracted information would be limited. Therefore, when several data sources are available, it is important to manage these sources in order to limit the problem of similarity and generate frequent patterns containing information from these different sources.

Various techniques have been proposed to mine multi-source data [45, 60, 67, 12, 43, 69]. These techniques can be divided into two main approaches. The first approach mines different data sources in a separate manner. A separate mining process is performed for each data source, and the information extracted from all the mining processes is then combined to obtain global

information which contains various kinds of information coming from different data sources. This approach allows obtaining valuable information. However, the major limitation of this approach lies in the fact that some data sources may provide useful information only if they are mined with other data sources ; thus, mining this source separately will result in useless or insufficient information. In addition, this approach leads to a loss of information when there is a problem of similarity in a data source.

This allows us to understand that different data sources are interrelated, that is having different kinds of relations among each other where certain data sources provide data to other sources. This approach would lead to the loss of important information due to the loss of the relations existing between different data sources. The second approach manages multi-source data by integrating all data sources in the dataset together and mining them in a single mining process. This approach maintains the relations between the data sources, which avoids data loss and allows generating rich information. However this approach results in a complex form of data that represent the input of the mining process which also has a high complexity, especially when there is a large number of data sources where each one provides different kinds of data.

Based on the limitations of both approaches, the main challenge in our work lies in **how to mine a complex dataset formed of multiple heterogeneous and interrelated data sources by managing the relations existing among different sources and taking into consideration the algorithm complexity in order to extract rich and useful information in the form of sequential patterns containing various kinds of information and to compensate the loss of information caused by the problem of data similarity**.

3 Approach and contributions

We propose an approach that manages multi-source data in a single process. In this frame, we propose to limit the complexity by using a strategy of selective mining where additional sources are mined only when needed during the process. This approach is designed to be generic as it is not restricted to data of a specific domain or structure.

As mentioned in section 2, we notice that there may be different relations among data sources. In this context, we distinguish between two kinds of relations among the sources. The first kind of relations exists between the sequential data source and the descriptive data source of users where the latter provides specific information to each user thus allowing to obtain more precise frequent sequential patterns than the ones generated from a traditional mining of sequential data.

On the other hand, the second kind of relations exists between the sequential data source and the descriptive data source of items where the latter provides additional data about each item in the data sequences. A data sequence, representing a sequence of item ids, is thus provided by additional descriptive attributes to each item. These attributes represent more general information than item ids, and thus they provide more generality to the sequential data source.

In our proposed approach, we manage the two kinds of relations while mainly focusing on the second kind by taking advantage of the descriptive data source of items in order to generate general sequential patterns. In order to overcome the problem of data similarity and thus generate more frequent patterns, we define two similarity measures in order to compare different patterns and the items that they contain : pattern similarity and item similarity. Then, we form general patterns from the similar patterns containing similar items where a general pattern is a pattern that contains more general information than traditional frequent patterns. Finally, we propose a novel method of support calculation of this new kind of patterns in order to detect frequent

general patterns.

Hence, our approach solves the problem of data similarity and low data coverage by generating more frequent patterns ; in addition, the frequent general patterns formed are rich as they contain various kinds of information.

4 A link with the educational domain

The use of technology to better understand how learning occurs has received considerable attention in recent years [19]. Digital educational data, whether concerning learning or teaching processes, is collected from VLE's and information systems of educational institutions, and they are now available for analytics for different purposes.

"Learning analytics" (LA) is best defined as "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs" [36]. It aims at improving the learning process by giving learners the ability to evaluate their academic progress and enhance their achievements and motivation.

Our work is funded by PIA2 e-FRAN METAL project [4]. The aim of the project is to design, develop and evaluate a set of individualized monitoring tools for school students or teachers (Learning Analytics and e-learning), and innovative technologies for personalized learning of written languages (French grammar) and orally (pronunciation of modern languages). It thus contributes to the improvement of the quality of learning and the development of language proficiency by students. Our work is specifically dedicated to school students where the objective is to contribute to the improvement of their learning process.

In the domain of education, digital data are collected from multiple data sources which could provide various kinds of data forming a multi-source and heterogeneous dataset. Hence, our proposed approach can be applied on this dataset in the purpose of providing students with personalized recommendations of pedagogical resources in order to help them improve their academic performance through learning analytics.

The proposed model has been designed for educational purposes, while being generic. Due to a problem of data availability in the project, we have not yet been able to evaluate our model on educational data. For this reason, we evaluate the model using a music dataset which has an adequate structure to our proposed model. Once we are able to obtain the educational dataset of the project, we will evaluate our model using this educational dataset as well.

5 Plan of the thesis

The rest of the thesis is organized as follows. Chapter 1 presents the state of the art in our research domain. We start by presenting the state of the art in the domain of sequential pattern mining. Then, we discuss the multi-* data that represents multi-source, multi-dimensional and relational data. After discussing the multi-* data, we present the state of the art in the domain of multi-* pattern mining and contextual pattern mining.

Chapter 2 presents our scientific contributions and the proposed algorithm. In this chapter, we first present a description of the various data sources in a multi-* dataset. We then discuss our proposition for a structure of relations among data sources. After that, we present G_SPM algorithm for mining frequent general patterns in multi-* data. Chapter 3 presents the experimental

4. http ://metal.loria.fr/

results and evaluations of our proposed algorithm. We start by a description of the application dataset, then we discuss the evaluation measures that we determined for evaluating G_SPM. We then discuss the experimental evaluations of G_SPM compared to naive approaches, and we conclude the chapter with discussions and conclusions of the conducted experiments. Finally, we conclude the thesis manuscript and discuss the future perspectives in chapter 4.

1

Related Work

Contents

Data has been generated and collected enormously in all domains across decades. Nowadays, data is generated not only by enterprises in the industry sector, but also by individuals on a daily basis. There is an increasing number of techniques that facilitate the collection and storage of the digital data. In this context, many frequent pattern mining approaches were proposed in the literature to analyze and understand the data and then extract useful knowledge that serves business or research purposes. In this chapter, we present the literature review in the domain of sequential pattern mining, that is the approach that we have identified as the most adequate to the kinds of data that we have as well as to the scientific problems that we address. Then, we present the various kinds of data. After that, we present a literature review of the most interesting research works proposed for mining these various kinds of data. Finally, we present a global conclusion of the related work.

1.1 Sequential pattern mining (SPM)

Agrawal and Srikant [3] were the first to address the problem of sequential pattern mining, and they defined sequential pattern mining as follows :

"Given a database of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, sequential pattern mining is to discover

all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern."

In this work, they stress on the importance of adopting property 1.1.1 for mining frequent sequences in an efficient manner, which states that the support decreases with the sequence extension. Therefore, if a sequence is infrequent, then this directly implies that all its supersequences are infrequent too and thus there is no need to generate these candidates.

Sequential pattern mining has grown to be a topic of high interest and has become essential to many application domains like customer purchases, medical treatments, bioinformatics, etc. For example, in the domain of customer purchase analysis, customer shopping sequences represent the purchase behaviour of the customers as follows : a customer first buys a Sony smartphone, then a Sharp television then a Sony digital camera.

All sequential pattern mining algorithms have a sequential database and a user-defined minimum support threshold (*minsup*), and generate all frequent sequential patterns as the algorithm output.

In what follows, we present the definitions that allow the understanding of the sequential pattern mining.

Definition 1.1.1. *(Sequence)*

Let us consider that $\{I = I_1, I_2..., I_n\}$ *is a set of n items,* s_i *is an itemset that contains an unordered set of items from* I*.* s_i *is also called an element or an event. A sequence s is an ordered list of itemsets and is denoted by* $s =< s_1, s_2..., s_m >$*.*

Definition 1.1.2. *(Length of a sequence)*

The length of a sequence s, denoted by $l(s)$ *is determined as the total number of instances of items in this sequence. A sequence with length l is called an l-sequence.*

Definition 1.1.3. *(Subsequence and supersequence of a sequence)*

A sequence $s =< s_1, ..., s_k >$ *is a subsequence of the sequence* $t =< t_1, ..., t_h >$*, denoted as* $s \subseteq t$*, if there exist integers* $1 \leq i_1 < i_2 < i_3... < i_j$ *such that* $s_1 \subseteq t_{i_1}$*,* $s_2 \subseteq t_{i_2}$ *...,* $s_k \subseteq t_{i_j}$*. In this case and for the same conditions, t is a supersequence of s, denoted as* $t \supseteq s$*.*

Definition 1.1.4. *(Sequential database)*

A sequential database S_D *is a set of tuples,* $< sid, s_{id} >$ *where* s_{id} *is a sequence and sid is the identifier of this sequence.*

Definition 1.1.5. *(Support of a sequence)*

The support of a sequence s is defined as the number of sequences in the database containing s. Let us consider the sequential database S_D *and the sequence* $s =< s_1...s_k >$*, the support of s, denoted as* $support_{(S_D)}(s)$ *is calculated as follows :*

$$support_{(S_D)}(s) = |\{(< sid, s_{id} >) \in S_D; s \subseteq s_{id}\}| \tag{1.1}$$

Definition 1.1.6. *(Frequent sequence)*

Let us consider a minimum support threshold, denoted as minsup. A sequence s is frequent in a sequence database S_D *if* $support_{(S_D)}(s) \geqslant minsup$*. In other words, s is frequent only if it occurs at least minsup times in* S_D*. A frequent sequence is called a sequential pattern.*

Property 1.1.1. *(Anti-monotonicity property)*

Given a sequential database S_D *over I and two sequences s and* s'*.*

$$s \subseteq s' \implies support_{(S_D)}(s) \geq support_{(S_D)}(s') \tag{1.2}$$

Example 1.1.1. *(Sequential pattern mining)*

Let us consider in Table 1.1 an example presented in [21] of a sequential database S_D that represents customer purchases . The set of items in S_D is {a,b,c,d,e,f,g}, and the minsup is set to 3. The sequence $s_4 : <\{e\}\{g\}\{af\}\{c\}\{b\}\{c\}>$ contains 6 itemsets : $\{e\}, \{g\}, \{af\}, \{c\}, \{b\}$, and $\{c\}$. This sequence means that this customer purchased e then g ; then, he/she purchased a and f at the same time ; after that, he/she purchased c then b then c. There are 7 instances of items in this sequence ; therefore, it is a 7-sequence. The item c occurs in 2 itemsets of this sequence : the fourth and the sixth itemsets ; however, the sequence contributes only one to the support of the item c. The sequence $s'_4 : <\{b\}\{c\}>$ is considered as a subsequence of s_4 as the itemsets of the former are subsets of the itemsets in s_4 respecting the order of the itemsets. The support of the sequence s'_4 in S_D is 3 as it appears in 3 sequences of the database (sequence 1, 3 and 4). Sequence s'_4 is considered as a frequent sequential pattern as $support_{(S_D)}(s) \geqslant minsup$. Table 1.2 displays all sequential patterns in S_D with their support values.

sid	Sequence
1	<{a}{abc}{ac}{d}{cf}>
2	<{ad}{c}{bc}{ae}>
3	<{ef}{ab}{df}{c}{b}>
4	<{e}{g}{af}{c}{b}{c}>

TABLE 1.1 – Sequence database S_D

Sequence	Support
<{a}>	4
<{b}>	4
<{c}>	4
<{e}>	3
<{f}>	3
<{a}{b}>	4
<{a}{c}>	4

Sequence	Support
<{b}{c}>	3
<{c}{b}>	3
<{c}{c}>	3
<{d}{c}>	3
<{a}{c}{b}>	3
<{a}{c}{c}>	3

TABLE 1.2 – Frequent sequential patterns with minsup = 3

Definition 1.1.7. *(Sequence prefix)*

Let us consider the sequences $s =< s_1, s_2..., s_k >$ and $t =< t_1, t_2..., t_l >$ where s_i and t_i are frequent itemsets in s and t respectively and $l \leqslant k$. t is a prefix sequence of s if $t_i = s_i$ for $(i \leqslant l-1)$ and $t_l \subseteq s_l$ as long as the remaining items of s_l that is $(s_l - t_l)$ are considered after the items of t_l according to the alphabetical order.

Definition 1.1.8. *(Sequence suffix)*

Let us consider the sequences s and t previously defined in definition 1.1.7 where t is a prefix of s. Sequence $u =< u_1, u_2..., u_m >$ is called the suffix of s with respect to prefix t if and only if $\exists\ i \leqslant k$, $u = (s_i - t_l)$ and $\forall\ 1 < j \leqslant m$, $(u_j = s_{i+j-1})$.

Example 1.1.2. *(Sequence prefix and suffix)*

Let us consider a sequence $s =< \{ef\}\{ab\}\{df\}\{c\}\{b\} >$ which is the third sequence in the sequential database of our running example that is represented in Table 1.1. The sequences : <

$\{e\}>, <\{ef\}>, <\{ef\}\{a\}>, <\{ef\}\{ab\}>$ *are prefixes of the sequence* s. $<\{ab\}\{df\}\{c\}\{b\}>$ *is a suffix of the sequence* s *with respect to prefix* $<\{ef\}>$, *and* $<\{\_b\}\{df\}\{c\}\{b\}>$ *is a suffix of* s *with respect to the prefix* $<\{ef\}\{a\}>$.

Various approaches have been proposed across years to manage the problem of sequential pattern mining of sequential databases. The differences among all the proposed algorithms are in the choice of search space navigation, the database representation as well as the different data structures used for representing the input database which affects the mining process.

Sequential pattern mining algorithms traverse the search space to generate their candidate sequential patterns using either breadth-first or depth-first search enumeration. The breadth-first search enumeration, also named level-wise enumeration, explores all items that are found at one level and then moves down to explore the items of the next level. First, it explores all the singletons (1-sequential patterns) and generates frequent ones ; then, it extends the frequent singletons to generate all frequent 2-sequential patterns, then it extends frequent 2-sequential patterns to generate all frequent 3-sequential patterns, and so on until the last level where all frequent sequences are generated. The representative algorithms of the breadth-first search enumeration are the Apriori-based ones such as GSP algorithm [56].

The depth-first enumeration extends a sequential pattern until the minimum support threshold is not satisfied instead of exploring all l-pattern candidates before going forward to $l+1$-pattern candidates. It starts from the singleton sequences and extends it to generate frequent 2-sequential patterns, 3-sequential patterns, 4-sequential patterns and so on until there are no more possible extensions. Then the algorithm goes backward to find more patterns from other sequences. Examples of algorithms which adopt the depth-first enumeration are PrefixSpan [41] and SPADE [64] algorithms.

Sequential pattern mining algorithms are classified into two categories, apriori-based algorithms and pattern growth-based algorithms. Apriori-based sequential pattern mining algorithms are based on AprioriAll algorithm [3], the first sequential pattern mining algorithm which is inspired by Apriori algorithm [4] for frequent itemset mining. Apriori-based algorithms explore the database search space using breadth-first search enumeration in order to generate frequent candidate patterns, and it scans the database repeatedly to calculate their support.

Pattern growth-based algorithms are based on FP-growth algorithm [26] that has been proposed as an improvement to Apriori algorithm [4] and thus generally to Apriori-based algorithms. These algorithms explore the database using depth-first enumeration. Pattern growth-based algorithms do not require generation where they scan the database to explore $(k+1)$-candidate patterns from (k)-candidate patterns.

There are two kinds of representations of the sequential databases, the horizontal and the vertical database representations.

A horizontal database representation consists of a set of sequences where each sequence has an identifier followed by an ordered set of itemsets that occur in this sequence.

A vertical database representation consists of a set of items where each item is associated with each sequence identifier (sid) and itemset/event identifiers (eid) where this item occurs, also indicating a timestamp of the item in sid. This information is referred to as the *id-list* of the item.

A horizontal database can be transformed into a vertical database in one database scan, and we can also retrieve a horizontal database from a vertical one. Table 1.1 is an example of a horizontal database representation showing the sequence of items and itemsets represented by their sequence identifiers. Table 1.3 and 1.4 displays the vertical representation of Table 1.1 for the items showing each item along with the sequences and the itemsets/events in each sequence

where this item occurs.

Example 1.1.3. *(id-list in a vertical database representation)*
Let us consider the item d in Table 1.3. The id-list of the item d can be represented as follows : $< d : (1,4),(2,1),(3,3) >$. *Item d has appeared in the fourth itemset of the first sequence, in the first itemset of the second sequence and in the third itemset of the third sequence of the sequential database.*

a		**b**		**c**		**d**	
sid	eid	sid	eid	sid	eid	sid	eid
1	1, 2, 3	1	2	1	2, 3, 5	1	4
2	1, 4	2	3	2	2, 3	2	1
3	2	3	2, 5	3	4	3	3
4	3	4	5	4	4, 6	4	-

TABLE 1.3 – Vertical representation of the database displayed in Table 1.1 (1)

e		**f**		**g**	
sid	eid	sid	eid	sid	eid
1	-	1	5	1	-
2	4	2	-	2	-
3	1	3	1, 3	3	-
4	1	4	3	4	2

TABLE 1.4 – Vertical representation of the database displayed in Table 1.1 (2)

In the following subsections, we present one sequential pattern mining algorithm from each of the two kinds of database representation.

1.1.1 SPM algorithms using horizontal representation

Sequential pattern mining algorithms using a horizontal database representation are classified into apriori-based and pattern growth-based algorithms. In this subsection, we present a sequential pattern mining algorithm from each category.

Apriori-based algorithms

Some apriori based sequential pattern mining algorithms using horizontal database representation are GSP [56], PSP [37] and MFS [68]. From this category, we present *GSP*, a well-known algorithm as it is one of the first sequential pattern mining algorithms.

GSP algorithm In 1996, Srikant and Agrawal proposed GSP algorithm (Generalized Sequential Pattern mining algorithm) [56] to mine sequential patterns. It is an improved version of the first sequential pattern mining algorithm, AprioriAll [3], that is proposed by the same authors.

As in Apriori algorithm [4], GSP algorithm adopts a multiple-pass, candidate-generation-and-test approach according to property 1.1.1. The algorithm works as follows.

It makes multiple scans over the database. The first scan determines the support of each item. Hence, after the first database scan, the algorithm knows which items are frequent. Each frequent item found is considered as a 1-sequential pattern.

Each subsequent pass starts with a "seed set" that consists of the frequent patterns found in the previous pass. This seed set is used to generate new potentially frequent patterns that are named candidate patterns where each one contains one more item than a seed sequence. The support for these candidate patterns is found during a database scan. At the end of the pass, the algorithm determines which of the candidate patterns are actually frequent. These frequent candidate patterns become the seed for the next step. The algorithm stops when no more frequent patterns are generated.

At the $(l+1)$-level pass, the candidate frequent pattern generation is done by a joining process of two l-patterns as follows. A sequence s_1 joins with s_2 if by dropping the first item of s_1 and the last item of s_2, the same subsequences are obtained. Therefore, s_1 is extended with the last item or itemset of s_2 to obtain the new $k+1$-candidate pattern

GSP is an important and well-known algorithm in the domain of sequential pattern mining ; however, it has some limitations. GSP performs multiple scans of the sequential database in order to calculate the supports of candidate frequent patterns to evaluate if they are frequent. The repetitive scans of the database lead to a high cost especially for huge databases. In addition, due the to the joining process to obtain (l+1)-candidate frequent pattern, two sequences can satisfy the conditions of join and thus would be extended to generate an (l+1)-candidate frequent pattern ; however, this may be a non-existing or infrequent sequence in the database. This process thus leads to a useless increase of the time complexity of the algorithm.

Pattern growth-based algorithms

A main advantage of pattern growth-based algorithms is the generation of frequent sequential patterns without candidate generation : only patterns that actually exist in the database are explored. However, the repetitive scans of the database is a process of high cost ; therefore, a new concept of database projection [40, 41, 27] has been proposed From this category, we present PrefixSpan algorithm. Among pattern growth-based approaches, PrefixSpan has been widely used in the literature in various research works in the domain of sequential pattern mining [45, 63, 52, 16, 42, 33, 9, 28]. This algorithm is proved to be an efficient and fast algorithm compared to classical sequential pattern mining algorithms like Apriori and GSP.

PrefixSpan algorithm In 2001, Pei et al., [41] proposed PrefixSpan algorithm (Prefix-projected sequential pattern mining). This algorithm is built upon the concept of sequential database projection proposed in FreeSpan algorithm [24]. PrefixSpan extends FreeSpan by taking into account the testing of prefix subsequences and the projection of their corresponding postfix subsequences into separate projected databases.

Definition 1.1.9. *(s-projected database)*

Let us consider a frequent sequence s in a sequential database S_D. The s-projected database, denoted by $S_D|s$, is the collection of all suffixes of the sequences in S_D with respect to the prefix s.

Example 1.1.4. *(s-projected database)*

Table 1.5 displays some projected databases of the sequential database in Table 1.1 with respect to 1-prefixes.

Prefix	Projected Database	Prefix	Projected Database
<{a}>	<{abc}{ac}{d}{cf}>, <{_d}{c}{bc}{ae}>, <{_b}{df}{c}{b}>, <{_f}{c}{b}{c}>	<{d}>	<{cf}>, <{c}{bc}{ae}>, <{_f}{c}{b}>
<{b}>	<{_c}{ac}{d}{cf}>, <{_c}{ae}>, <{df}{c}{b}>, <{c}>	<{e}>	<{_f}{ab}{df}{c}{b}>, <{af}{c}{b}{c}>
<{c}>	<{ac}{d}{cf}>, <{bc}{ae}>, <{b}>, <{bc}>	<{f}>	<{ab}{df}{c}{b}>, <{c}{b}{c}>

TABLE 1.5 – Projected database of Table 1.1

PrefixSpan uses depth-first enumeration to explore the search space of sequential patterns. The first step, that is common to most of sequential pattern mining algorithms, is to scan the sequential database in order to calculate the support of the single items and thus to obtain the frequent singletons (1-sequential patterns). Next, it performs a depth-first enumeration starting from the frequent singletons. For a given sequential pattern s of length k, PrefixSpan first builds the projected database with respect to the prefix pattern s. It then scans the projected database of s, counts the support values of items and detect frequent items to be appended to s, and thus (k+1)-frequent patterns are formed. After that, the algorithm repeats this step recursively to find all frequent sequential patterns.

The advantage of the projected database concept is that the projected databases keep shrinking where a projected database is smaller than the original one because the subsequences that are projected into a projected database are only the postfix subsequences of a frequent prefix. However, its disadvantage is that it scans the database repeatedly and builds projected databases, and these repeated scans could be of a high cost. Certain techniques were proposed for reducing the number of projected databases [25]. One of these techniques is the pseudo-projection where the sequence databases can be held in main memory in order to avoid redundancy in the postfixes of the projected databases. Instead of constructing a physical projection all the postfixes, they use pointers referring to the sequences in the database as a pseudo-projection.

Despite this disadvantage that could be limited when the size of the sequential database as well as the length of patterns are reasonable, prefixspan is a well known and widely used algorithm in various domains like multi-dimensional sequential pattern mining and sequential pattern mining with time constraints.

1.1.2 SPM algorithms using vertical representation

Sequential pattern mining algorithms using a vertical database representation adopt the pattern growth-based approach. Some of these algorithms are SPADE [64], SPAM [6], Hirate and Yamana [28], and CMAP [14]. Among these algorithms, SPADE is the first algorithm proposed in this context.

SPADE algorithm

In 2001, Zaki et al., [64] proposed the first algorithm that uses a vertical database format for mining sequential patterns, SPADE algorithm (Sequential Pattern Discovery using Equivalence classes). This algorithm is inspired by ECLAT [66] and CHARM [65] algorithms for mining frequent itemsets. SPADE uses lattice-based search techniques and simple joins. The lattice-based approach is used to decompose the original search space, the lattice, into smaller pieces, the sub-lattices, that can be processed in the main memory in an independent manner. It usually requires three scans of the sequential database in order to obtain the frequent sequential patterns. The algorithm works as follows. First, it scans the database and transforms it from horizontal to vertical database format where each item is associated with its corresponding id-list. An example of horizontal-to-vertical database transformation is the horizontal database displayed in Table 1.1 that is transformed into the vertical database displayed in Tables 1.3 and 1.4. Then, it scans the vertical database, reads the id-list of each item and calculates the support value of the item which represents the total number of *sid*'s found in the id-list. The items that have a support value above the predefined minsup are considered as frequent singletons. After generating these singletons, the algorithm performs a vertical-to-horizontal transformation in order to recover the original horizontal database by grouping the items with the same *sid* and in an increasing order of *eid*. Table 1.6 shows the horizontal database recovered from the vertical id-lists of items. Then, it forms a list of all 2-sequences for each *sid* in order to generate frequent 2-sequential patterns. The support of a frequent 2-sequential patterns is calculated by joining the id-lists of the two items forming this sequence by making a join on the same *sid* where the *eid* of the first sequence occurs before the *eid* of the second sequence. The generated frequent 2-sequential patterns are used to construct the lattice. The number of frequent 2-sequential patterns could be large which makes the lattice formed out of it quite large to fit in the main memory. Therefore, the lattice is decomposed into different equivalence classes such that the sequences having the same prefix items belong to the same equivalence class. Then, frequent $(k+1)$-sequential patterns are explored from frequent (k)-sequential patterns via temporal joins using either breadth-first or depth-first enumerations. Then id-lists of the frequent (k)-sequential patterns are joined to calculate the support values and therefore generate frequent $(k+1)$-sequential patterns. Longer frequent sequences are generated similarly until no more frequent sequences could be generated.

In the breadth-first enumeration of the equivalence classes of the lattice, the equivalence classes are generated in a recursive bottom-up manner where all the child classes at each level are explored before moving to the next level. For example, in order to generate frequent $(k+1)$-sequences, all the (k)-sequences have to be already explored. While in depth-first enumeration of the equivalence classes of the lattice, all equivalence classes for each path in the search space are explored before moving to the next path of the search space. For example, all the (2)-sequences as well as the (k)-sequences have to be already explored in order to generate frequent $(k+1)$-sequences.

sid	(Item, eid) pairs
1	(a, 1), (a, 2), (b, 2), (c, 2), (a, 3), (c, 3), (d, 4), (c, 5), (f, 5)
2	(a, 1), (d, 1), (c, 2), (b, 3), (c, 3), (a, 4), (e, 4)
3	(e, 1), (f, 1), (a, 2), (b, 2), (d, 3), (f, 3), (c, 4), (b, 5)
4	(e, 1), (g, 2), (a, 3), (f, 3), (c, 4), (b, 5), (c, 6)

TABLE 1.6 – Recovered horizontal database from the vertical database in Tables 1.3 and 1.4

The advantage of SPADE algorithm is that it requires only three scans of the database in order to generate all the frequent sequences which makes it faster than GSP algorithm which requires multiple database scans to generate all the frequent sequences. In addition, SPADE uses simple temporal join operation on id-lists.

After the proposition of SPADE algorithm, other sequential pattern mining algorithms using vertical database representation were proposed across years [6, 5, 62, 14, 62].

SPADE as well as other sequential pattern mining algorithms using vertical representation have several advantages. The main advantage is introducing the concept of id-lists. The id-list structure is simpler than other structures used in horizontal databases such as complicated hash-tree structures. In addition, as the length of a frequent sequence increases, the size of its id-list decreases, resulting in faster joins. The vertical database representation also facilitates the process of support calculation. The support of a pattern is the total number of sid's in the id-list and the support of a $(k+1)$-sequence is calculated through the intersection of the id-lists of the two frequent (k)-sequences that form this sequence ; therefore, it avoids the repetitive database scans required for calculating the support of the sequences. It also saves the information not only about the sequences where the items occur but also about the events that show when these items occur. However, a disadvantage is that when the cardinality of the set of transactions is very large, the intersection time of the id-lists becomes costly.

SPAM [6] and Bitspade [5] algorithms propose a bitmap representation of the database with efficient support counting which optimizes the id-list structure and thus makes these algorithm faster than SPADE. However, the limitations of these two algorithms are that they generate a large number of candidate patterns and that the join operation made in order to generate the id-list of each of these candidate patterns could be a process of high cost. Therefore, CM-SPADE and CM-SPAM algorithms [14] were proposed to improve these two algorithms by introducing the co-occurrence pruning in order to reduce the number of joins. These algorithms initially scan the database to create the co-occurrence map (CMAP) that stores frequent 2-sequences. Then for each pattern explored in the search space, if its last two items are not frequent 2-sequences, then this pattern is discarded without making the join operation to generate its id-list. This process of verifying if the pattern is frequent before generating its id-list saves time and reduces the cost of the mining process and thus makes these algorithms faster than the previous ones.

In this section, we have presented the related work in the domain of sequential pattern mining by presenting important algorithms from different categories.

1.2 Multi-* data

Voluminous amounts of digital data are created and collected on daily basis. This data may not always come from the same data source and may not always be homogeneous ; however, it could be heterogeneous representing different kinds of data and thus forming complex datasets. The heterogeneity of this data lies in the fact that it could be provided by various data sources, could be represented as different data dimensions and could have different relations among the sources or dimensions. This heterogeneity results in what we call $multi-*\ data$ which briefs the multiple kinds of data contained in this complex dataset. In this section, we describe the various kinds of multi-* data, and in the following section, we focus on the mining of such data.

1.2.1 Multi-source data

A data source is simply any source that provides data. It could be a file, web data, a database or a data warehouse. Multi-source data represents data collected from multiple data sources.

These different data sources can be classified into sources of homogeneous structures, named homogeneous data sources, and others of heterogeneous structures, named heterogeneous data sources. The data in a data source is known as local data with respect to this data source and the data of all data sources is known as global data.

Homogeneous data sources

Homogeneous data sources are the sources that have the same data structure which is the way of organizing the elements of data as well as the way they are related to each other. In other words, the data sources contain the same data elements with the same names, data types and data structures, and they are usually distributed among different locations. Table 1.7 displays two data sources providing academic data of students from two different schools. These different data sources are homogeneous as they have the same structure and provide the same kind of data elements which are the averages and the ranking of students among the class.

Another example of homogeneous data sources is provided in [12] where the data represents information about patients' trajectories provided by different hospitals.

Student-id	Average(/20)	Rank(/25)
123	12	15
126	15.8	3
124	11.5	18
122	7	22

(a) School 1

Student-id	Average(/20)	Rank(/25)
132	8.5	24
135	16.1	2
138	13.5	11
131	10	19

(b) School 2

TABLE 1.7 – Students' academic data from two schools

Heterogeneous data sources

Heterogeneous data sources are the sources that have heterogeneous structures. Heterogeneous data sources can be classified into two types. The first type is when the sources have heterogeneous data models at the local level. Data model heterogeneity, which is also considered as schematic and data heterogeneity, occurs when the same data is stored and represented in different ways causing data inconsistency and leading to obtain heterogeneous data sources although these sources have the same data nature with the same semantics. Example of these data inconsistencies are as follows. There could be conflicts of synonyms and homonyms among different data sources. A synonym conflict is when the same data element is represented in different names among different data sources. A homonym conflict is when different data elements have the same name among different data sources. There could also be differences in the data types, precision or scale ; for example, a data element could be defined as integer in one data source and as float in another one. In addition, there could be structural differences among different data sources on the level of their objects ; for example, it could have a single value in one data

source and multiple values in another one. There could also be a problem of missing or conflicting data in the data values recorded such as having two different values for the same object in two different data sources due to errors or differences in the underlying semantics [8].

The second type of heterogeneous data sources is when the sources are related but different. In other words, the various data sources have heterogeneous data models and provide different data. The data sources are related as they provide semantically relevant data ; however, each data source has its own role in providing one or more kinds of data. They may share some data types or names of data elements ; however, they do not represent the same elements and they have different semantics. Combining these various data sources together allows forming a heterogeneous dataset whose data heterogeneity and the relations existing among its different data sources make it rich, informative, understandable and thus interpretable.

An example of heterogeneous data sources in the domain of e-commerce is as follows : one data source could provide descriptive data about the customers, another data source could provide data about their past purchases, and a third data source could provide descriptive data about the products that are purchased by the customers. Each data source gives different kinds of data about the customers and their purchasing process which allows extracting various and rich information.

The main challenge in extracting information from multi-source data lies in how to manage such kind of heterogeneous data coming from different sources efficiently and with limited complexity. Multi-source frequent pattern mining has been a topic of interest, and many research works across years have proposed various techniques to mine multi-source data in order to obtain useful and rich information that serves the scientific objectives and desired outcomes. The related work in the domain of multi-source pattern mining is detailed in section 1.3.1.

1.2.2 Relational data

When the data are heterogeneous and of different natures, they represent relational data, and they can thus be represented as multiple relational database tables (relations). These data are called relational data, and they are also referred to as multi-relational data in the literature. Figure 1.1 displays an example of a relational database about customer's purchases of products from companies. This example shows that there are different kinds of relations existing among different tables.

Relational data allows extracting various and rich information that comes from various database tables, and the process of mining frequent patterns from this kind of data is known as relational pattern mining, also called in the literature multi-relational pattern mining. A literature review of the various approaches proposed in the domain of relational pattern mining is detailed later in section 1.3.2.

1.2.3 Multi-dimensional data

Data can consist of different data fields, where each field corresponds to a set of measured properties that are referred to as features, attributes, or dimensions [1]. This kind of data is called multi-dimensional data. An example of multi-dimensional data is as follows. Let us consider a data source providing descriptive data about the food products purchased by customers in a market ; this source provides different data dimensions to describe each product such as : product name, category, fabrication date, expiry date, etc. Each data dimension could be represented as several attributes describing different products.

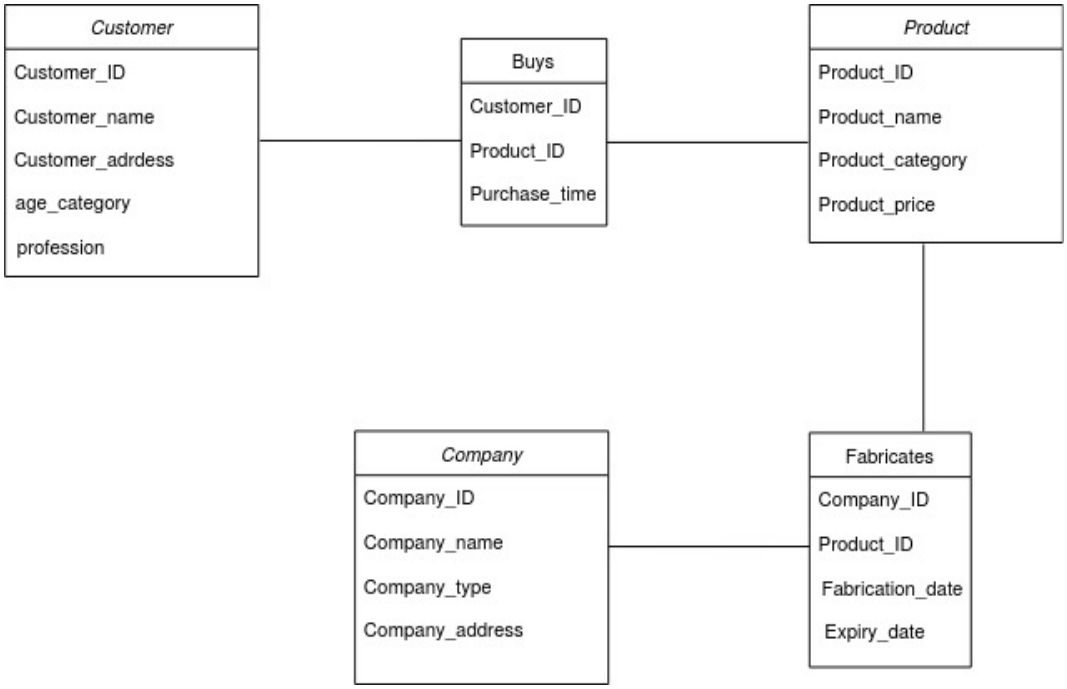

FIGURE 1.1 – Relational database

Frequent patterns could be generated from multiple data dimensions which allows obtaining patterns that contain various kinds of information. Such kind of frequent patterns are called frequent multi-dimensional patterns [45, 12].

In many applications, one data dimension of the multi-dimensional data represent data sequences forming a multi-dimensional sequential space. For example in the domain of customer purchase, the customer purchase history is associated with customer group, address, time, age group and others. This specific kind of multi-dimensional data is called multi-dimensional sequential data containing multi-dimensional sequences [45, 63, 12, 50, 47, 46, 29, 56, 22]. Pinto et al., [45] were the first to introduce the notion of a multi-dimensional sequence, and Table 1.8 displays an example of their database that contains multi-dimensional sequential data where *cid* represents the customers ids.

cid	Customer Group	City	Age Group	Product Sequence
10	business	Boston	middle	$< \{bd\}cba >$
20	professional	Chicago	young	$< \{bf\}\{ce\}\{fg\} >$
30	business	Chicago	middle	$< \{ah\}abf >$
40	education	New York	retired	$< \{be\}\{ce\} >$

TABLE 1.8 – A multi-dimensional sequential database [45]

In certain cases, there could be background knowledge about the data dimensions which allows translating this knowledge into a data hierarchical structure. When there are several attributes representing each data dimension, the hierarchical structure of data is called a data taxonomy. We name this kind of data as multi-dimensional data with hierarchical structure. Figure 1.2 displays an example of a hierarchy of food products.

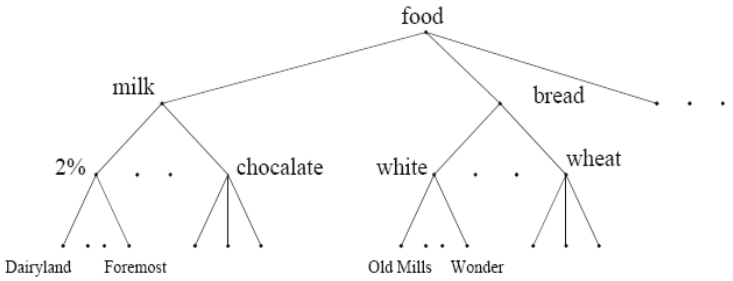

FIGURE 1.2 – A hierarchy of food products [18]

In other cases, data dimensions could represent descriptive data about one or more other data dimensions. This descriptive data provide certain contexts to the other dimensions which allows giving broader understanding of the data within which it is associated. This kind of data is called contextual data. The contextual data could be associated to data sequences giving contexts to each sequence forming contextual sequential data [48]. An example of a contextual sequence is : $cs : \{young, female\} < \{ab\}c >$. The contextual data represent a young female, and the sequential data represent $\{ab\}$ then c.

Many research works in the literature have been proposed to mine different kinds of multi-dimensional data. In section 1.3.3, we discuss the literature review of multi-dimensional sequential pattern mining, multi-dimensional pattern mining using hierarchical structures and contextual pattern mining.

From the previous description of the various kinds of data, we can understand that the term multi-* data refers to the following kinds of data : multi-source, multi-dimensional and relational data. Although they are different kinds, they are related to each other and the data in a dataset could sometimes be classified as one or more of these kinds of data. For example, we could have two data sources where each source provides more than one data dimension, and these sources could have relations among them ; in this case, the data is multi-source, relational and multi-dimensional at the same time. Such a dataset has more complex and heterogeneous structure than other datasets containing data of one kind ; consequently, it can be viewed and thus managed from different perspectives. Various approaches have been proposed across years for mining these different kinds of data. In the following section, we present the related work in mining these different kinds of data.

1.3 Mining multi-* data

Mining multi-* data has always been a topic of interest as the challenge lies not only in generating interesting frequent patterns from these data but also in the efficiency and the low complexity of the mining process. In the following subsections, we present the related work in the domains of multi-source, multi-dimensional and relational pattern mining respectively.

1.3.1 Multi-source pattern mining

Multi-source pattern mining is a topic of pattern mining concerned in mining data coming from different data sources. This topic has gained a great interest as it represents data in real applications. Various approaches have been proposed to mine this kind of data. These approaches are divided into two main categories. The first category is integrating data from all data sources

and mining them together in order to generate frequent global patterns. The second category is mining frequent patterns from each local data source, named frequent local patterns and then integrating all the local frequent patterns of all data sources together. In the following subsections, we present some interesting research works from the two categories of multi-source pattern mining.

Mining multi-source data in an integrated manner

Multi-source data integration involves aggregating data from different data sources to a centralized data source and perform a single mining process on this centralized source in order to extract patterns that are globally frequent among all the local data sources.

Some works proposed approaches to mine data that is integrated in a centralized source in order to extract frequent patterns that include information from all the sources.

Data fusion is the process of integrating data from homogeneous data sources into a single source, and a single mining process is performed on this data to generate frequent patterns. An example of this approach is the multi-sensor data fusion proposed in [32]. Multi-source data fusion is an intelligent synthesis process of remote sensing image data from multiple sources and generating accurate, complete and reliable estimation and judgement. Their proposed approach first collects the data from different sensors and then eliminates the data redundancy and conflicts that could exist among this multi-source data in order to generate useful information from this integrated data. This approach is proved to be efficient in this domain as it can improve the reliability of remote sensing information and can help in decision making and planning.

This approach of combining multiple data sources together and mining them in a single mining process has several drawbacks that proves its inefficiency. Integrating multi-source data in a single source could cause a considerable cost due to the data integration. In addition, the number of items in each itemset or sequence increases which augments the length of these itemsets or sequences resulting in higher complexity of the mining process. Integrating multi-source data into a single database may destroy some important local information from local data sources that reflects the distributions of patterns. For example, such kind of specific information would be lost : "85% of the branches within a company agree that a customer usually purchases sugar if he/she purchases coffee" [67]. Another possible limitation of this approach is the data protection and privacy rules that could allow transferring the resulting patterns without allowing to transfer the data. Besides, this approach considers that each data source has the same weight (importance) and doesn't take into consideration that different data sources could have different weights where a data source could have a bigger contribution to the mining process and its results than another one.

Mining Multi-source data in a distributed manner

In order to overcome the limitations of the previously described approach, other works proposed to mine different data sources in separate mining processes to obtain frequent local patterns and then integrate all the frequent local patterns generated from each mining process in order to obtain frequent global patterns.

In 2009, Peng et al., [43] proposed three algorithms in order to mine sequential data containing sequences of events that are represented in different domains where a domain contains events occurring within a predefined time window. The first algorithm proposed a naive approach which mines multiple data sources where each one is represented in a database. In this algorithm, the multiple sequence databases are integrated as one sequence database, and it is mined using tra-

ditional sequential pattern mining algorithms such as PrefixSpan algorithm [25]. This algorithm is then proved to be costly and inefficient due to the data integration. Therefore, two other algorithms were proposed to overcome the drawbacks of the first one. The second algorithm, IndividualMine algorithm, consists of two phases : the mining phase and the checking phase. In the mining phase, sequential patterns in each sequence database are mined using a traditional sequential pattern mining algorithm. Then in the checking phase, sequential patterns from all domains are integrated in order to generate multi-domain sequential patterns which has the same concept as multi-source sequential patterns. In this algorithm, each domain should individually perform sequential pattern mining algorithms, which incurs a considerable amount of mining cost. Therefore, in order to further reduce this cost, they propose a third algorithm, the PropagatedMine algorithm, in which those sequences that are likely to form multi-domain sequential patterns are extracted from their sequence databases. The PropagatedMine algorithm consists of two phases : the mining phase and the deriving phase. In the mining phase, the algorithm performs a sequential pattern mining using traditional sequential pattern mining algorithms in a starting domain and then propagates time-instance sets of the mined sequential patterns to the next domain until all domains are mined. In the deriving phase, the algorithm extracts those sequences with occurrence time equal to those of the time-instance sets propagated.

The third algorithm is proved to be the most efficient one among the three algorithms as it reduces the complexity and thus the cost of the mining process. Therefore, it is a good and efficient approach for mining this kind of data that is homogeneous among different data sources ; however, the concept of pattern propagation is infeasible when the data among different data sources are heterogeneous. In addition, this approach considers that different data sources have equal contributions to the resulting patterns and it thus doesn't take into consideration that a data source could contain data that is of bigger weight and thus is more important than the other.

Zhang et al [67] proposed an approach for generating global exceptional patterns in multi-database mining applications where each data source is represented in a separate database. An exceptional pattern is a pattern that is highly supported by only few data sources ; in other words, such a pattern has a very high support value in some data sources and zero support value in others which allows to have a high global support value. This kind of patterns reflect the individuality of different data sources. The proposed approach, named Kernel Estimation for Mining Global Patterns (KEMGP), considers that different data sources have different contributions to the resulting global patterns and thus they have different minimal supports. It mines each data source and generates its frequent local patterns, and then it identifies the local exceptional patterns of interest from these local patterns. In their work, they also describe another kind of patterns, called suggesting patterns [67, 43] that is based on the votes done by the databases in order to decide for the global patterns among all local ones. They define the suggesting patterns as the patterns that have fewer votes than a minimal vote ; however, they are very close to the minimal vote where the minimal vote is determined by users or by experts. If a local pattern has votes that are equal to or greater than the minimal vote, then the local pattern is said to be a global pattern, named a high-voting pattern. While if a local pattern has votes that are less than the minimal vote but are very close to the minimal vote, then it is called a suggesting pattern that could sometimes be useful for decision making [43]. The concept of "suggesting patterns" that is introduced and defined in this work is an interesting concept as it represents patterns that are on the margin of global ones and that weren't far from being global.

The approach proposed in this work is efficient as it allows identifying the importance of a pattern not only locally in a specific data source but also globally on the level of all data sources. However, the approach works strictly on homogeneous data sources where a global

pattern contains the same kind of information as the global pattern, while the only difference lies in their supports and importance but not in the kind of information it contains.

Mining frequent local patterns from multiple data sources and then performing the integration on a pattern base allows reducing the data movement. Hence, data mining applications based on local pattern analysis strategy can be able to learn models from distributed data without exchanging the raw data [51]. This respects the privacy and restrictions that are imposed on the data in certain cases. In addition, this approach allows knowing where the patterns are frequent and distinguishing between patterns that are frequent only locally in certain data sources and others that are globally frequent in all data sources. This allows having more precise information about the frequent patterns which helps in a better understanding of the output as well as in decision making. On the other hand, the approach of mining multiple data sources separately has its limitations. When mining multi-source data in an integrated manner, there is a single mining process for all the data which limits the cost ; however, in mining multi-source data separately, the number of mining processes is the same as the number of existing data sources which makes the overall process costly when the number of data sources is large. Furthermore, data among different data sources could have relations among them, and maintaining these relations when mining this data allows detecting frequent patterns which contain rich information from different data sources that provide certain kinds of semantics. Hence, mining these data sources separately could lead to losing the relations among them and thus losing their patterns or their semantics.

1.3.2 Relational pattern mining

Relational pattern mining is a topic of pattern mining that is concerned in mining data that come from relational database environments having multiple relations. Various approaches have been proposed in this domain. These approaches could generally be classified into two main categories, the first category is the propositionalization, and the second category is mining the data with maintaining the relational structure.

Propositionalization is the process that leads from relational data and background knowledge to a single table representation [34]. The data from multiple tables are combined together to form a global table that contains all the multi-relational data. This process can be achieved by performing join operations on all the related tables to obtain a global table, or it can be obtained by an aggregation technique that resumes the raw data of multiple tables into one compact table [59]. In this case, the mining process is performed on this global table in order to generate frequent patterns that contain information from all the previous multiple tables. The disadvantages of having a global table is that it is usually large as it contains the data from all the tables, and this could lead to data sparsity as well as redundant values coming from different tables.

Due to the drawbacks of the propositionalization method in mining relational data, other works propose to maintain the relational structure and mine directly the relational tables. In this context, many approaches were proposed across years. Geothals et al [17] proposed SMuRFIG algorithm (Simple multi-relational frequent Itemsets Generator) for the generalization of frequent itemset mining to the multi-relational case. The SMuRFIG algorithm uses the concept of id-list propagation across relational tables. First, the id-lists of all items are fetched from the data in their respective entities (tables), and then these id-lists are propagated to all other entities. The propagation function translates an id-list from one entity to its adjacent entities through the relations existing between these two entities, and it is recursively repeated until all the entities are reached. The frequent singleton itemsets are therefore generated where the support of each 1-sequential pattern is the total number of ids found in its id-list. Next, the frequent singleton

itemsets are combined into larger itemsets by the Keyclat function. The search space is traversed depth-first. In each recursion, two itemsets with a common prefix are combined to form a new candidate set where a prefix, initially empty, consists of one ore more common itemsets included in the two itemsets under measure. In order to compute the support of this new candidate itemset in the specified entities of the two itemsets, they intersect the id-lists of the two itemsets in these entities, and the support is the total number of ids in the newly generated list after the intersection. Then, these id-lists are propagated to all other entities. This process is repeated until all frequent relational patterns are found.

The id-list propagation mechanism is advantageous in avoiding the join operations between all database tables and keeps the search space in a manageable size ; however, this mechanism could be costly when the number of database tables is large as the number of KeyID list propagation mechanism will be high. In addition, this algorithm provides a meaning to the support value where the support of each itemset has a unique meaning that is determined depending on the specific entities under measure. However, a limitation of this algorithm is that it is designed to mine itemsets and not sequences, and it mines these itemsets from different tables in one mining process. Let us consider that one of the database tables contains data sequences and not itemsets ; in this case, the strategy of SMuRFIG algorithm could not be directly applied without managing the sequential data in a separate sequential pattern mining process.

The previously described approaches of relational pattern mining are efficient in the case when the different data sources have equal importance. In addition, the use of the different data sources is always restricted to extending the frequent patterns by adding information to these patterns.

However, data sources could be heterogeneous where each data source has a unique role in providing certain kinds of information. In addition, the role of a data source may not be restricted to the concept of adding the information directly to the frequent patterns ; however, it could be providing additional or background information to these frequent patterns. Due to these reasons, we understand the importance of taking into consideration the semantics of the data provided by each sources as well as the semantics of the relations between different data sources. Understanding these semantics would allow determining a richer and more useful view and structuring of the sources in order to generate richer and more informative patterns.

1.3.3 Multi-dimensional pattern mining

Multi-dimensional pattern mining is a topic of pattern mining that is concerned in mining data represented in different data dimensions. The data dimensions could represent homogeneous or heterogeneous data where this data could come from a single or multiple data sources. In section 1.2, we classified multi-dimensional data mining into three main categories : multi-dimensional sequential pattern mining, multi-dimensional pattern mining using hierarchical structures, multi-dimensional pattern mining using star schema and contextual pattern mining. In this section, we present the related work in each of these categories of multi-dimensional pattern mining.

Multi-dimensional sequential pattern mining

Multi-dimensional sequential pattern mining was defined by Pinto et al., [45] as the process of mining one or more dimensions of information in which the order of the dimension values is not important, alongside a single dimension of information in which order is important [44]. Table 1.8 shows their multi-dimensional sequential database. In their work, they proposed three algorithms to mine multi-dimensional sequential patterns, Uniseq, Dim-Seq and Seq-Dim algorithms.

Table 1.8 represents their dataset that consists of descriptive attributes and the purchase history of customers that are identified by customer-id (*cid*). This heterogeneous data consists of three dimensions : Customer group, City and Age group as well as a sequential data dimension "product sequence" that contains sequences of items from $a, b, ..., h$ representing the items bought by customers. The descriptive data dimensions and the purchase history are obviously provided by different data sources.

UniSeq algorithm works as follows. The input of the algorithm is the multi-dimensional sequential database and a predefined minimum support threshold (minsup), and the algorithm outputs the set of frequent multi-dimensional sequential patterns. The algorithm embeds the multi-dimensional data to the sequences by extending an element at the beginning of each sequence and adding the multi-dimensional data related to this sequence in order to obtain an extended multi-dimensional-extended sequence ; in other words, we obtain a single data dimension containing the sequential data along with the multi-dimensional within which it is associated. For example, for a tuple $t = (10, business, Boston, middle, < \{b, d\}\{d\}\{b\}\{a\} >)$ in Table 1.8, the sequence $s =< \{b, d\}\{d\}\{b\}\{a\} >$ is extended by the multi-dimensional data to obtain a multi-dimensional sequence $s^{md} =< (Business\ Boston\ middle) < \{b, d\}\{d\}\{b\}\{a\} >>$. Table 1.9 shows the multi-dimensional extension database from the multi-dimensional sequential database shown in Table 1.8. After this step, the multi-dimensional sequences are mined traditionally through the PrefixSpan sequential pattern mining algorithm [25]. UniSeq algorithm, which integrates heterogeneous data and mines them in a sequential mining process is efficient when the number of dimensions and the total number of sequential items are small ; however, when there is a large number of dimensions or sequential items, the algorithm becomes inefficient as the multi-dimensional sequences become large which increases the time and cost of the mining process.

cid	MD-extension of a sequence
10	<{business Boston middle} {b d} c b a >
20	<{professional Chicago young} {b f} {c e} {f g} >
30	<{business Chicago middle} {a h} a b f >
40	<{education New York retired} {b e} {c e} >

TABLE 1.9 – Multi-dimensional extension database from the multi-dimensional sequential database in Table 1.8

Dim-Seq and Seq-Dim algorithms decompose the problem into two sub-problems : mining multi-dimensional data and mining sequential data while maintaining the relation between these two kinds of data.

In Dim-Seq algorithm, they start by mining the multi-dimensional data using BUC-like algorithm [7]. By scanning the database, frequent one-dimensional items are generated, then frequent two-dimensional items are generated and so on until all frequent dimensional patterns are generated. After this step, for each multi-dimensional pattern, they mine the sequential data associated to it using PrefixSpan algorithm [25] in order to obtain a multi-dimensional sequential pattern.

In Seq-Dim algorithm, they first mine the sequential data using PrefixSpan algorithm [25] to generate frequent sequential patterns ; after this step, for each frequent sequential pattern, they mine the multi-dimensional data associated to it using BUC-like algorithm [7] in order to obtain a multi-dimensional sequential pattern.

The experiments conducted to evaluate the three algorithms revealed important results as follows. When the dimensionality is low, Uni-Seq algorithm outperforms the two other algorithms

as the major cost is in mining the sequential data which can be easily handled by a sequential pattern mining algorithm such as PrefixSpan algorithm [25].

When the database is large, Seq-dim is the most scalable algorithm as it strictly mines frequent multi-dimensional data that are associated to the frequent sequential patterns. A large database is complex to be handled with UniSeq where long sequences containing multi-dimensional and sequential data are formed. Dim-Seq algorithm also has poor scalability when handling a large dataset as there could be many generated frequent multi-dimensional patterns which lead to no multi-dimensional sequential patterns.

The main limitation of Dim-Seq algorithm is that it generates all possible frequent multi-dimensional combinations before detecting frequent sequential patterns which may result in frequent multi-dimensional patterns that do not generate any frequent multi-dimensional sequential patterns ; however, Seq-Dim algorithm avoids such a cost. The main limitation of Uni-Seq algorithm is either having a large database or having large number of data dimensions. Seq-Dim is an efficient and scalable algorithm, and it is the fastest algorithm among the three in most cases. These three algorithms were then largely discussed, used and enhanced in other works in the literature [47, 16, 43, 12]

Yu and Chen [63] also discussed the multi-dimensional sequential pattern mining problem where they propose to mine sequential data from d-dimensions sequence data (d>2). They use online stock-trading site as an example where each customer may visit a web site in a series of days ; each day takes a series of sessions and each session visits a series of web pages. The data for each customer is therefore 3-dimensional sequence : days, sessions and pages. An example of a sequence in their database is as follows : $<<< ab >_1< dcf >_1>_2<< df >_1< bc >_1>_2>_3$. This sequence means that a certain customer visits the website in two successive days. On the first day, he/she goes through the website twice : in the first session, he/she sequentially visits pages a and b, and in the second session, he/she sequentially visits pages d, c and f. On the second day, he/she goes through the website twice : In the first session, he/she sequentially visits pages d and f, and in the second session, he/she sequentially visits pages b and c. The subscripts in the sequence denote the dimension numbers, and dimensions 1, 2 and 3 correspond to visited pages, sessions, and days, respectively. This sequence is a 3-dimensional sequence whose elements $<< ab >_1< dcf >_1>_2$ and $<< df >_1< bc >_1>_2$ are 2-dimensional sequences. While the sequence $<< ab >_1< dcf >_1>_2$ is a 2-dimensional sequence whose elements $< ab >_1$ and $< dcf >_1$ are 1-dimensional sequences. In this context, they proposed two algorithms to mine such kind of data, AprioriMD and PrefixMDSpan algorithms.

AprioriMD algorithm extends the traditional Apriori [4] or GSP [56] algorithm. There are two major differences between AprioriMD and these two traditional algorithms, the first difference is in the method of generating candidate sequences, and the second difference is in the method of computing the support values of the generated candidate sequences. First, for generating candidate sequences, they take into account the dimensional scopes of frequent $(k-1)$-sequential patterns in order to generate all possible k-candidate sequential patterns with different possible dimension scopes. Then, in order to compute the support values of the candidate sequential patterns, they use a tree structure, called candidate tree, where they attach each tree branch with an element including two components : item name and dimensional scope value, and then they traverse the tree and compute the support values traditionally as in Apriori or GSP algorithms.

In PrefixMDSpan, they extend the traditional PrefixSpan algorithm [41] taking into account the dimensional values. The main difference is that there is a dimensional scope between the frequent items of a prefix-projected database and its prefix. Therefore, in order to manage this difference, they updated the definitions of prefix and postfix in the database projection of PrefixSpan algorithm [25] in order to take into account the dimensional scopes. This is done by

constructing a table, called IDM that stores for each item in the prefix-projected database : the number of transactions containing this item along with its corresponding dimensional value. Then, they generate frequent items and their dimensional values within this prefix-projected database, and they append this information to the prefix in order to generate the frequent $(k+1)$-sequential patterns.

Their evaluation results show that PrefixMDSpan algorithm outperforms the AprioriMD algorithm. These results are expected as the original PrefixSpan algorithm generally outperforms original Apriori and GSP algorithms.

The notion of multi-dimensional data in this work is very specific to certain kinds of dimensions. The items are no longer single values ; however, they are attached to their dimension values to form a composite element ; this means that for the same item, there could be different elements. In addition, their work is restricted to data dimensions having sub-relations among each others and thus it could not be applied on data dimensions that are heterogeneous, unrelated or that contain different kinds of relations.

In their work, Huang et el., [29] discussed the work of Pinto et al., [45] and their proposed algorithms. They claimed that although Seq-dim algorithm is efficient and scalable to mine multi-dimensional patterns in high-dimension space, this method requires multiple scanning of projected databases to mine multi-dimensional patterns because the cost will increase as the amount of data increases and as the dimensionality becomes higher. In this frame, they propose a new algorithm ExtSeq-MIDim that employs the method of PrefixMDSpan algorithm proposed in [63] to mine sequential patterns from multi-dimensional sequence data, then form projected multi-dimensional database for each sequential pattern and use MIDim algorithm (Memory Indexing for mining multi-dimensional pattern) to mine multi-dimensional patterns within the projected multi-dimensional databases. The advantage of this proposed algorithm over PrefixMDSpan [63] is that it reduces the times of scanning projected databases when mining multi-dimensional patterns. MIDim algorithm first scans the projected multi-dimensional database only once to load it into memory. After this step, it finds all frequent multi-dimensional values and outputs matched multi-dimensional patterns. Then, it builds an index set of each prefix multi-dimensional pattern, finds local frequent multi-dimensional values from index set and grows discovered patterns. Finally, the algorithm recursively constructs an index set of the detected pattern and discovers all multi-dimensional patterns.

This algorithm takes advantage of the concept of memory indexing without multiple scanning of projected databases which makes it efficient and scalable. The only restriction in this algorithm is that it mines web sequential data that are partitioned through three sub-related dimensions as in the work of [63].

In 2005, Plantevit et al., [46] proposed a new algorithm, M^2SP, for mining multi-dimensional sequential patterns. They consider multi-dimensionality in the sense that the patterns do not only contain several dimensions, but they are also combined over time. An example of a pattern that they mine is as follows : *"A customer who bought a surfboard together with a bag in NY later bought a wet suit in SF"*. This pattern combines two dimensions (City and Product) over time where some attributes appear before the others (NY appears before SF and surfboard appears before wet suit). Table 1.10 displays an example of the dataset that they mine in their work.

(D) Date	(CG) Customer-Group	(C) City	(A) Age	(P) Product	(Q) Quantity
1	Educ	NY	Y	c	50
1	Educ	NY	M	p	2
1	Educ	LA	Y	c	30
1	Ret	SF	O	c	20
1	Ret	SF	O	m	2
2	Educ	NY	M	p	3
2	Educ	NY	M	r	10
2	Educ	LA	Y	c	20
3	Educ	LA	M	r	15

TABLE 1.10 – Transactions of customers' purchases [46]

The dataset is comprised of six dimensions : The date of the transactions (D), customer's category (CG), customer's age (A), the city of issuance of transactions (C), the product of transactions (P) and the quantity of purchased products (Q). For example, the fifth tuple means that on date one, old retired customers bought two products m in San Francisco. In their work, they generate frequent multi-dimensional sequences that deal with specified dimensions and that are frequent with respect to other dimensions. In this concern, they consider three sets of dimensions : (1) Dimension D called $temporaldimension$, (2) Dimensions A, P and Q called $analysisdimensions$, (3) Dimensions CG and C are called $referencedimensions$. Tuples over analysis dimensions are those that appear in the items that constitute the sequential patterns to be mined. The table is partitioned into blocks according to tuple values over reference dimensions ; the tuples in each block are ordered over temporal dimension and the support of a given multi-dimensional sequence is the ratio of the number of blocks supporting the sequence over the total number of blocks. For example, in order to generate frequent multi-dimensional sequences that deal with the age of customers and the products that they purchased with respect to the groups of customers and the cities where the transactions were issued, they partition the dimensions as follows : (1) Dimensions customer's age category (A) and the product of transactions (P) are considered as the analysis dimensions (2) Dimensions customer's category (CG) and city of transaction issuance (C) are considered as the reference dimensions. The algorithm works as follows. First, it splits the dataset into different blocks according to distinct tuple values over reference dimensions. Figure 1.3 displays the blocks that are partitioned from the dataset in Table 1.10. After that, it computes the support of a sequence by counting the number of blocks that support this sequence where a block supports a sequence if all the multi-dimensional items in the sequence satisfy a set of tuples in the block and have the same order in both the sequence and the block.

In this work, they also introduce a new interesting form of sequences called *"Jokerized Multi-dimensional Sequence"*. An item can only be extracted if there exists a frequent tuple of values from analysis dimensions containing it. For example, it could happen that (Y, r), (M, r) and (O, r) are all infrequent whereas the value r alone is frequent. In this case, they consider $(*, r)$ which is said to be jokerized where the symbol $*$ represents *any* value in the corresponding dimension. A jokerized item is a multi-dimensional item that contains at least one specified analysis dimension, and it contains a $*$ only if no specific value from the analysis dimensions can be set. In other words, by replacing any occurrence of $*$ with any value of its corresponding analysis dimensions can not result in a frequent sequence. Therefore, a jokerized sequence is a

D	CG	C	A	P	Q
1	Educ	NY	Y	c	50
1	Educ	NY	M	p	2
2	Educ	NY	M	p	3
2	Educ	NY	M	r	10

a. Block $(Educ, NY)$

D	CG	C	A	P	Q
1	Educ	LA	Y	c	30
2	Educ	LA	Y	c	20
3	Educ	LA	M	r	15

b. Block $(Educ, LA)$

D	CG	C	A	P	Q
1	Ret.	SF	O	c	20
1	Ret.	SF	O	m	2

c. Block $(Ret., SF)$

FIGURE 1.3 – Blocks defined on Table 1.10 over dimensions CG and C [46]

sequence containing at least one jokerized item. In order to mine frequent jokerized patterns, a levelwise algorithm is used in order to mine the frequent multi-dimensional items having the least number of joker values possible. In this concern, they consider a lattice whose lower bound is the multi-dimensional item $(*, ..., *)$ that is the most general level. This lattice is partially built from $(*, ..., *)$ up to the frequent items containing as few $*$ as possible.

This algorithm is efficient as it considers mining a more generalized form of patterns in the case where no possible specific frequent patterns could be generated. However, the algorithm considers mining only two levels of granularity in the lattice instead of mining all possible items from all levels of granularity in the lattice.

Multi-dimensional sequential pattern mining using hierarchical structures

When there is background knowledge or semantics about the data dimensions, they can be formed in hierarchical structures representing the different dimensions or their attributes in multiple levels of granularity. As discussed in section 1.2, this kind of data representation with the knowledge that it provides is advantageous to the mining process as it would allow generating rich frequent patterns which contain useful information.

In 1996, Srikant and Agrawal were the first to introduce the concept of hierarchy management in the extraction of sequential patterns [56] ; however, their approach considers sequences of a single dimension defined on several hierarchical levels. Several works were proposed later in order to mine multi-dimensional data using hierarchical structures.

Han et al. [22] proposed mining association rules from multiple levels of a taxonomy, named multiple-level association rules, by extending the Apriori algorithm [4] that considers mining single-level association rules. Their approach concerns mining multiple-level association rules ; however, it could be adapted in order to mine multiple-level frequent sequential patterns. Figure 1.4 displays their taxonomy concerning food categories. The algorithm uses different minimum support thresholds ($minsup$) at different levels of the taxonomy, and it works as follows. It first finds frequent patterns at the top-most level of the taxonomy. The support is then lowered as going down through the taxonomy levels in order to find the frequent patterns at each level assuming that only the descendants of frequent itemsets are the ones who should be generated.

The algorithm of mining multiple-level frequent patterns or association rules from hierarchical structures is advantageous as it is a progressive mining process of refined knowledge from data that would allow generating relevant resulting patterns that are interpretable. Nevertheless, the multiple levels of abstraction in this algorithm concern only the mining process and not the generated frequent patterns. This algorithm allows generating frequent sequential patterns composed of items that are strictly from the same level of granularity, named intra-level multi-dimensional

sequential patterns.

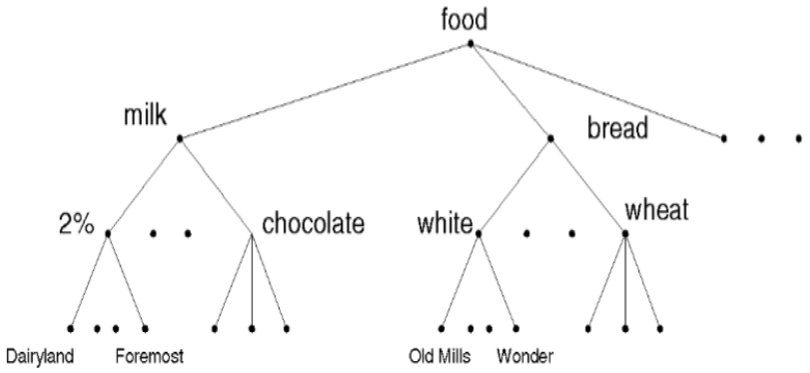

FIGURE 1.4 – A taxonomy of food categories [22]

Plantevit et al., [47] proposed an new algorithm called M^3SP that is an extension of their previous work proposed in [46] for mining frequent multi-dimensional patterns. This new algorithm takes both multi-dimensional and multi-level aspects into account where they mine multi-dimensional sequential data at multiple levels of granularity among taxonomies of dimensions. The difference between M^3SP and M^2SP algorithms is that the M^2SP goes directly from the most specific level in the taxonomy to the most general level, while in M^3SP algorithm, it goes to the most general level by passing the taxonomy level by level. The most general level in the taxonomy is denoted by ALL in M^3SP (which is denoted as $*$ in M^2SP). Table 1.11 shows the dataset that they mine in their work.

(D) Date	(CG) Customer-Group	(A) Age	(Loc) Location	(Prod) Product	(Qty) Quantity
1	Educ	Y	Berlin	beer	50
1	Educ	Y	Berlin	pretzel	20
2	Educ	Y	London	M2	30
3	Educ	Y	London	wine	20
4	Educ	Y	NY	M1	40
1	Educ	O	LA	soda	20
2	Educ	O	Paris	wine	30
2	Educ	O	Berlin	pretzel	10
3	Educ	O	Paris	M2	20
1	Ret	Y	London	whisky	20
1	Ret	Y	London	pretzel	20
2	Ret	Y	Berlin	M2	30
1	Ret	O	LA	chocolate	50
2	Ret	O	Berlin	M1	20
3	Ret	O	NY	whisky	20
4	Ret	O	Paris	soda	30

TABLE 1.11 – Customers' purchases [47]

In this approach, they mine rules like : *When the sales of soft drinks are high in Europe, exports of Perrier in Paris and exports of soda in the US become high later on.* This rule combines two dimensions (Location and Product) over time and at different levels of granularity among the dimensions of the taxonomies (Perrier and Soda are kinds of soft drinks). Figures 1.5 and 1.6 display the hierarchies of the dimensions "product" and "location".

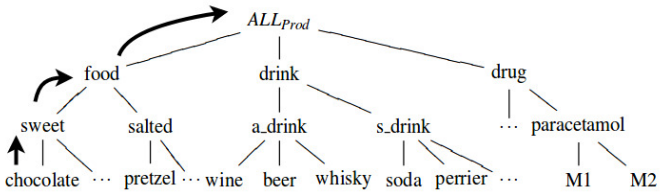

FIGURE 1.5 – Hierarchy of dimension "Product"

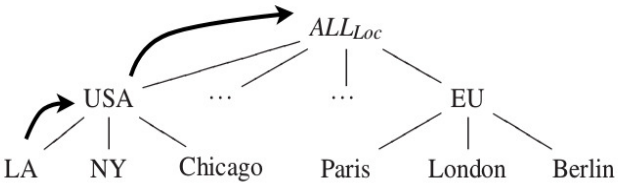

FIGURE 1.6 – Hierarchy of dimension "location"

M^3SP algorithm uses all the concepts of M^2SP including dividing the multi-dimensional sequential database into different blocks. They mine frequent sequences based on the maximal atomic frequent sequences, referred to as maf-sequences. The atomic sequence $< a >$ is said to be a maf-sequence, if $< a >$ is frequent and if for every multi-dimensional item a' such that a' is more specific than a , the sequence $< a' >$ is not frequent.

The algorithm is composed of two steps. First, maf-sequences are mined following a strategy inspired from BUC algorithm [7]. Then, all frequent sequences that can be built up based on maf-sequences found in the previous step are mined using SPADE algorithm [64]. A maf sequence is generated by first computing all frequent minimal atomic sequences which are all sequences of the form : $< \{ALL_1, ..., ALL_{i-1}, d_i, ALL_{i+1}, ..., ALL_m\} >$ where ALL_i is the most general item in the i^{th} analysis dimension's taxonomy, m is the number of analysis dimension and d_i is the direct successor of the ALL in the i^{th} analysis dimension. Then, M^3SP prunes from the taxonomy of each analysis dimension all values d_i and their successors such that $< \{ALL_1, ..., ALL_{i-1}, d_i, ALL_{i+1}, ..., ALL_m\} >$ is not frequent. Then, based on the frequent sequences generated in this previous step, the algorithm generated recursively all the frequent maf sequences. After this step, frequent sequences are mined using SPADE algorithm [64]. Examples of frequent multi-dimensional sequences obtained by their algorithm are : <{(EU, pretzel), (EU, a drink)}, {(EU, M 2)}>, <{(U S A, ALL Prod)}> and <{(Berl in, ALL Prod)}>.

M^3SP algorithm is an efficient algorithm that allows extracting various patterns that are at multiple levels of granularity. It is well-adapted for mining sequential datasets of sequences containing items ; however, this algorithm is not suitable for mining sequences containing a combination of items and itemsets.

Egho et al. [12] proposed an algorithm called MMISP for mining multi-dimensional itemset

sequential patterns. Their data is in the form of complex sequences including both items and itemsets. It incorporates background knowledge in the form of hierarchies over attributes.

Table 1.12 displays their database that contains multi-dimensional itemsets data sequences where P represents patients and T represents patients' trajectories, and figure 1.7 displays examples of their data dimensions that are represented in the form of taxonomies over their attributes. This algorithm generates a novel kind of patterns that contain events which are generated by combining items and itemsets. An example of an event is :
$e = (UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\})$ is an event, where UH_{Paris} and C_1 are two multi-dimensional items representing the two dimensions (hospital and diagnosis), $\{p_1, p_2\}, \{drug_1, drug_2\}$ are two itemsets representing the medical procedures and the medical drugs. They define a multi-dimensional itemset data sequence as a sequence composed of events. An example of a multi-dimensional itemset data sequence is : $s =< e_1, e_2, ..., e_l >$.

P	T
P_1	$\langle(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\}), (UH_{Paris}, C_1, \{p_1\}, \{drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\})\rangle$
P_2	$\langle(UH_{Paris}, C_1, \{p_1\}, \{drug_2\}), (UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2\})\rangle$
P_3	$\langle(UH_{Paris}, C_1, \{p_1, p_2\}, \{drug_1, drug_2, drug_3\}), (GH_{Lyon}, R_1, \{p_2\}, \{drug_2, drug_4\})\rangle$
P_4	$\langle(UH_{Paris}, C_1, \{p_2\}, \{drug_1, drug_2\}), (UH_{Paris}, R_2, \{p_3\}, \{drug_2\}), (GH_{Lyon}, R_2, \{p_2\}, \{drug_3\})\rangle$

TABLE 1.12 – A database of patient trajectories [12]

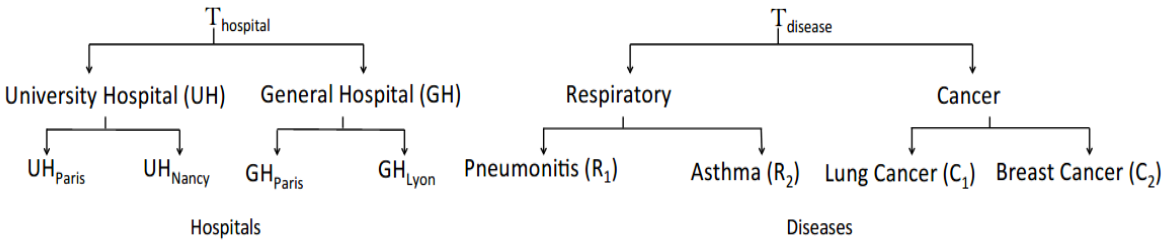

FIGURE 1.7 – Taxonomies for hospitals and diagnoses

Their algorithm mines the most specific multi-dimensional itemset sequential patterns with different levels of granularity at each dimension. It follows a bottom-up approach and works as follows. The first step of the algorithm is mining frequent multi-dimensional items from the events of the sequences where these items could exist at different levels of granularity. To this end, they adopt the database partitioning method that was introduced in [47] where they consider temporal, analysis and reference dimensions, and they partition the database into different blocks and compute the supports of multi-dimensional items accordingly. In their example, they consider the dimensions as follows : D (Date) is the temporal dimension, H (hospital) and D (disease) are the analysis dimensions and P (patient) is the reference dimension. Figure 1.8 displays the block partitioning of the database with respect to the reference dimension.

Date	Hospital	Diagnosis
1	UH_{Paris}	C_1
2	UH_{Paris}	C_1
3	GH_{Lyon}	R_1

Block: $Patient_1$

Date	Hospital	Diagnosis
1	UH_{Paris}	C_1
2	GH_{Lyon}	R_1

Block: $Patient_3$

Date	Hospital	Diagnosis
1	UH_{Paris}	C_1
2	UH_{Paris}	C_1
3	GH_{Lyon}	R_1

Block: $Patient_2$

Date	Hospital	Diagnosis
1	UH_{Paris}	C_1
2	UH_{Paris}	R_2
3	GH_{Lyon}	R_2

Block: $Patient_4$

FIGURE 1.8 – Block partitioning of the database according to the reference dimension (Patient)

They define a multi-dimensional component as a tuple of dimensional items ; then, they generate frequent multi-dimensional components by the product of all partially ordered sets of dimensions, and they build an iceberg semi-lattice whose nodes are multi-dimensional components and mine frequent multi-dimensional components from the lattice. An example of a frequent multi-dimensional component is : (UH_{Paris}, C_1).

The second step of the algorithm is mining frequent itemsets where they aim at extracting the set of all items that are frequent in a 1-sequence. They use a traditional sequential pattern mining algorithm to mine the itemsets by adapting it in order to generate only 1-sequential patterns. The third step is generating frequent events, and it is achieved by combining frequent multi-dimensional components with frequent itemsets. To this end, they construct a prefix-tree such that each branch of the tree represents an event as follows. The first level in this tree is composed of the frequent multi-dimensional components and from the second level of the tree until reaching the leaves, each level consists of frequent itemset candidates for each itemset part in the vector of itemsets. The final step of the algorithm is generating frequent multi-dimensional itemsets patterns. To achieve this goal each branch (event) in the prefix tree is assigned a unique identifier ; an example is as follows : $e_1 : (UH_{Paris}, C_1, p_1, drug_1)$ where e_1 is the identifier of this event. In addition, each block, having a unique identifier, is transformed into a sequence according to the date and content of the block ; thus, a database containing sequences of events is obtained. An example of a sequence in this database is as follows : $P_1 :< \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}\{e_2\}\{e_{10}\} >$. Then, a standard sequential pattern mining is applied on the transformed database to generate frequent sequences of events consisting of items and itemsets.

This approach provides a novel kind of multi-dimensional patterns that contain multi-dimensional events where an event is a complex structure consisting of items and itemsets. The data in their dataset come from multiple data sources ; however, the different data sources are homogeneous where they provide the same kinds of data having the same structure. Therefore, this approach works strictly for a specific structure of data and the different data sources need to be homogeneous.

1.3.4 Multi-dimensional pattern mining using star schema

Some works distinguish multiple levels of importance (different roles) of the data dimensions by representing the data in different structures ; one of these structures that is widely used is the

star schema [53]. A star schema is an architectural model that consists of one or more central fact tables surrounded by a set of dimension tables and the joins that relate the dimension tables to the fact tables. A fact table usually contains numeric measurements as well as the foreign keys of all the dimension tables surrounding it, and it is the only type of table which has multiple joins with the other tables. A dimension table contains a key column (or columns) that acts as a unique identifier as well as other descriptive columns. Figure 1.9 displays an example of a movie star schema that contains six dimension tables (Movie, Date, Director, Studio, Award and Epoch) and a fact table linking all the different dimensions.

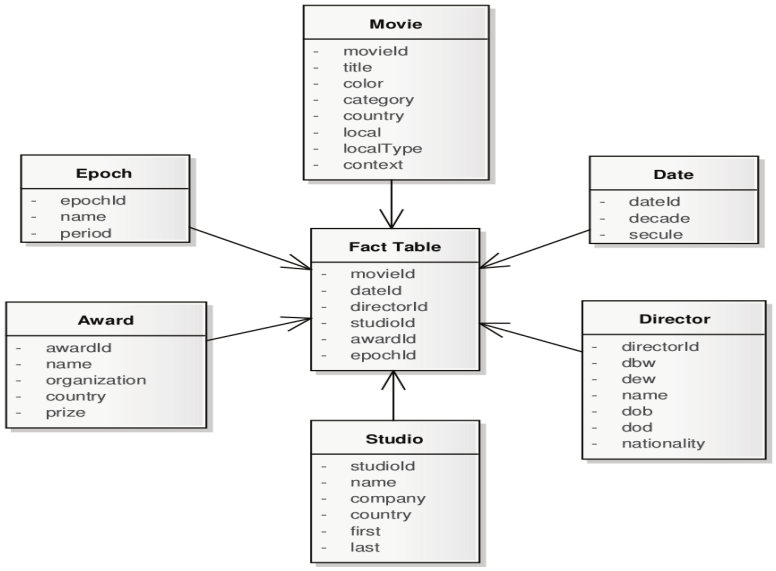

FIGURE 1.9 – Star schema structure of movies database [55]

Different approaches using the star schema structure were proposed across time where the main advantages of this structure is that they avoid the high cost of the entire join operations by eliminating this process and taking advantage of the properties offered by the star schema [54, 31, 10, 38, 61].

Silva et al, [54] proposed to mine frequent patterns in a star schema based on FP-Growth algorithm [26]. Their proposed algorithm does not materialize the entire join between the tables ; instead, it constructs an FP-Tree for each dimension and then combines them to form a super FP-Tree, that is then used as input to the FP-Growth algorithm. First, the fact table is scanned to count the support of each foreign key. After this step, an FP-tree is constructed for each dimension table, named DimFP-Tree. After this step, a global FP-Tree structure, named Super FP-Tree is built where it combines the branches in all the DimFP-Trees, accordingly to the facts. Finally, it calls the original FP-Growth algorithm having this tree as input in order to find multi-relational patterns.

This approach is advantageous in its number of database scans as it scans the database only twice. However, a limitation of this approach is that it is not scalable ; when the number of dimension tables is very large, the number of DimFP-Tree to be built becomes large which increases the cost of this process as well as the complexity of combining these DimFP-Trees in order to build the Super FP-Tree.

1.3.5 Contextual sequential pattern mining

Using background knowledge in the domain of frequent pattern mining can help discovering patterns, as well as finding completely new patterns that originate from combining the original data with additional background data [39]. In this frame, some works consider that having a better knowledge about the features of objects would help in better understanding these objects and thus would help in interpretability and decision making. This knowledge that is represented as a set of descriptive dimensions about the objects is known as contextual information. An example of contextual information in the domain of customer purchase is the age category : young, middle and retired. Therefore, the understanding of the resulting patterns and the decision making process in this case could depend on the age category of customers. Contextual information is usually provided with data sequences. Stefanowski et al., [57] define contextual information as additional information specifying elements or sequences. They discuss that context attributes could be introduced both for describing the complete sequence (e.g. characterizing user profiles)as well as each element in the sequence (describing circumstances for succeeding transactions).

Some works focused on mining sequential patterns where the contextual information is added to the whole sequence [48]. The process of mining traditional sequential pattern mining when data sequences are associated with contextual information has limitations. The first limitation is that a sequence needs to be frequent in the whole database in order to be detected as a frequent sequence, based on the concept of the support value that depends on the frequency of this sequence in the database. However, this pattern may be frequent only for a certain context and not frequent for others. For example, in the customer purchase, buying video games is frequent ; however, when analyzing the database, we notice that it is only frequent for young customers while not frequent for old customers, but it has been generally detected as frequent as there is a small number of old customers in the database. The second limitation is that a pattern could be frequent in a certain context but not frequent in the whole database ; in this case, this pattern is not detected as frequent and this information that could have probably been important would be lost. For example, buying newspapers is not frequent in the whole database ; however, it is frequent for old customers (4 out of 5 old customers buy it), but as the number of old customers is small in the database (only 5 out of 16 customers are old), and it is not detected as frequent. Due to these limitations, adding contextual information to the sequential data is important to obtain more precise and interpretable patterns.

We describe below the definitions of a contextual database, a context, a sequence database of a certain context and contextual sequential pattern as discussed in [48], [49].

Definition 1.3.1. *(Contextual Database)*

A contextual sequence database CD is defined as a relation $R(ID, S, D_1, ...D_n)$, where $dom(S)$ is a set of sequences and $dom(D_i)$ for $1 \leq i \leq n$ is the set of all possible values for D_i. $D_1, D_2, ...D_n$ are called the contextual dimensions in C_D.

A tuple $u \in CD$ is denoted by $< id, s, d_1, ..., d_n >$.

Values on contextual dimensions can be organized as hierarchies where each dimension is represented in a hierarchy with its values from the most general to the most specific.

Table 1.8 could be viewed as a contextual database as it contains data sequences as well as different dimensions that could be viewed as contextual dimensions. Hierarchies over dimensions "City" and "Age Group" that are found in Table 1.8 are displayed in figures 1.10 and 1.11.

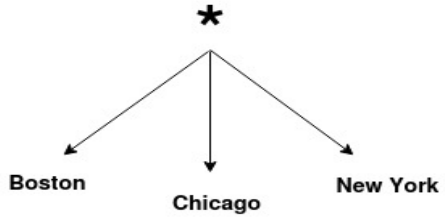

FIGURE 1.10 – Hierarchy of dimension "City"

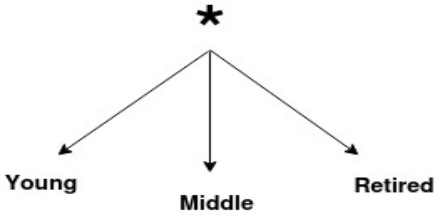

FIGURE 1.11 – Hierarchy of dimension "Age Category"

Definition 1.3.2. *(Context)*

A context c in CD is denoted by $[d_1, ...d_n]$ *where* $di \in dom'(D_i)$. *If, for* $1 \leq i \leq n$,$d_i \in dom(D_i)$, *then c is called a minimal context. The set of all contexts associated with the partial order* $>$ *is called the context hierarchy. Let* $u =< id, s, d_1, ...d_n >$ *be a tuple in CB. The context c such that* $c = [d_1, ...d_n]$ *is called the context of u.*

A context hierarchy over the context dimensions represented in figures 1.10 and 1.11 is displayed in figure 1.12.

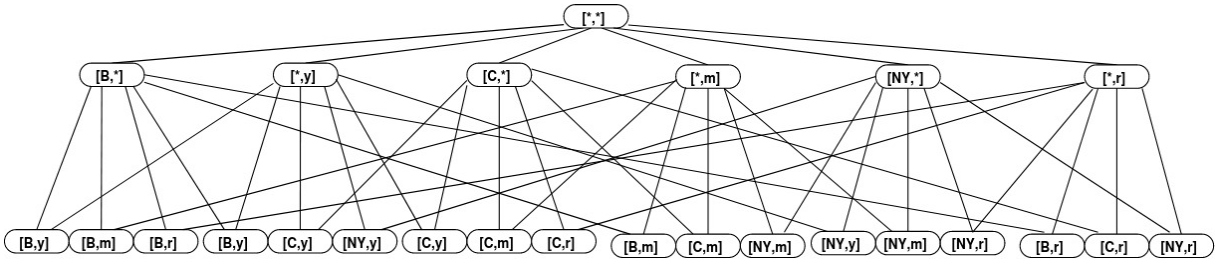

FIGURE 1.12 – Context hierarchy of context dimensions "City" and "Age Group" of Table 1.8

Definition 1.3.3. *(Sequence database of a context)*

Let $c = [d_1, ...d_n]$ *be a context in CB and U be the set of tuples contained by c. The sequence database of c, named* $B(c)$, *is defined as the set of tuples* $< id, s >$, *such that* $\exists u \in U$ *with* $u =< id, s, d_1, ...d_n >$. *The size of a context c, named* $|c|$, *is defined as the size of its sequence database* $(|c| = |B(c)|)$.

Definition 1.3.4. *(Contextual Sequential Pattern)*

Let c be a context and s a sequence.. s is frequent in c only if it is frequent in $B(c)$ (if $sup_{B(c)} \geqslant minsup$.
A contextual sequential pattern is a couple (c, s), such that s is c-specific. A sequence s is c-specific to c only if : (1) s is frequent in c, (2) s is general in c which means that s is frequent in all the descendants of c in the context hierarchy and (3) there does not exist a context c', such that $c' > c$ and s is general in c'

Rabatel et al [48] proposed an approach in order to mine contextual sequential patterns. They add context awareness to the extraction of frequent sequential patterns in order to mine patterns that are more precise than traditional sequential pattern mining and multi-dimensional sequential patterns.

Their approach mines the whole sequence database in a single process in order to extract contextual sequential patterns. In a similar concept of representing the multi-dimensional data in hierarchical structures over their attributes , in this approach they represent the dimensions in the form of hierarchies over their attributes, and from these hierarchies of dimensions, they construct a context hierarchy by associating the set of all contexts with the partial order $>$. They defined more or less general contexts. For example, the context corresponding to professional middle-aged customers is less general than that corresponding to professional customers.

Their proposed algorithm works as follows. The first part of the algorithm is to extract frequent sequences in minimal contexts as all contextual sequential patterns can be generated from this set of sequences ; then for each frequent sequence, they extract the corresponding set of minimal contexts where it is frequent. This part is based on PrefixSpan algorithm [41] with the differences are in the process of support calculation of the items where it is calculated in each minimal context of the s-projected database as well as the process of constructing s-projected databases where they only consider sequences found in minimal contexts. The second part of their algorithm is generating the set of contextual sequential patterns from the set of sequences obtained in the first part of the algorithm. A sequential pattern is extracted with the set of minimal contexts where it is frequent. From this set, they generate the contextual sequential sequential patterns which are obtained by this sequential pattern.

This approach has similar data characteristics as certain multi-dimensional sequential pattern mining approaches, for example the work in [45]. The advantage of this approach over that of Pinto is that it treats the data dimensions as contextual dimensions and establishes relations between different dimensions through the context hierarchy that combines all dimensions at their different levels if attributes. This enriches the value of the dimensions and their attributes and thus results in generating more precise and interpretable resulting patterns. However, this approach could have a high complexity when the number of context dimensions and/or attributes is large due to concept of context hierarchy that integrated all context dimensions and their attributes. In addition, this approach provides contextual information to the whole sequence. Other works has focused not only on contextual information added to the sequences as a whole but also on the elements of the sequences.

In this frame, Ziembinski et al., [70] proposed a new algorithm, called ContextMapping algorithm, in the domain of sequential pattern mining named Context Based Sequential Pattern Mining (CBSPM). It considers contextual information on two levels : the sequences and the elements of the sequences. In addition to the sequence context previously described, they introduce a new type of context, named the element context. The element context is assigned to each element in the sequence and it describes circumstances of an event/transaction referring to the itemset in this element. They consider that an element of a sequence is frequent if the number of other sequences containing similar elements is greater than the minimal support threshold

required by the user (minsup).

The first step is to detect frequent contexts in the database. The algorithm constructs a list of contexts similar to other contexts in the database. Then, it calculates the similarity between contexts with respect to appropriate similarity functions and threshold values. If the number of similar contexts in the list (corresponding to different sequences) is greater than *minsup*, then this context is considered as frequent. After this step, it performs a mapping procedure that maps each frequent context to two interconnected artificial items, called A and B, respectively. In the original sequence, it replaces the frequent context by item A and adds item B to itemsets of all elements from other sequences on the list. It also maps the sequence context to an additional artificial element added at the beginning of every sequence and similarly to contexts of elements by artificial items. This mapping transforms similar context sequences into traditional-like sequential database having only nominal items. The next stage of the algorithm is to mine context patterns using an adapted version of PrefixSpan algorithm [25]. The algorithm ensures that all patterns must contain an additional element representing the sequence context and further sequence of elements being a pattern body. Each pattern element must contain an artificial item representing the frequent context and frequent itemset [58]. In a final step, the algorithm performs a reverse mapping of the mined patterns where it replaces the artificial items with their corresponding context values and thus the frequent contextual patterns are generated.

The introduction of contexts not only on the level of a sequence but also at the level of elements of a sequence is advantageous as it allows obtaining richer and more informative patterns. However, the number of mined patterns is definitely multiplied and the mining process becomes more complex and time consuming as the patterns contain contextual on two levels (sequences and elements), and this information is added to the sequences which results in longer patterns containing information of a more complex structure.

In these previously described approaches in the domain of contextual sequential pattern mining, the structure of context attributes is homogeneous for all objects of the same type (sequences or elements) ; therefore, these approaches do not take into consideration heterogeneous structures of context dimensions and their attributes, while in real life datasets, different dimensions could have heterogeneous structures.

1.4 Conclusion

In this chapter, we have presented and analyzed related works in the domain of sequential pattern mining. Then, we have discussed the various kinds of data : multi-source, multi-dimensional and relational data that constitute a multi-* dataset. We then detailed the literature review of the most interesting works proposed for mining these kinds of data.

From this review, we consider that in most of the cases, the multi-dimensional data could be viewed as multi-source and vice versa, and that the data dimensions can be provided by multiple data sources. These data sources can thus be either homogeneous or heterogeneous. For example, the multi-dimensional data in [45] can be be viewed as multi-source data where data dimensions can be provided by different data sources. We also consider that the multi-dimensional data can be viewed as contextual data if they represent descriptive data about the users. For example, the multi-dimensional data in [45] can also be viewed as contextual data to the users and thus to their data sequences. Thus, the resulting multi-dimensional sequential patterns can also be considered as contextual sequential patterns. Similarly, contextual data can be represented as multiple data dimensions where each dimension provides one context.

Hence, this allows us to conclude that multi-source, multi-dimensional and relational mining

techniques can be used for mining multi-* data depending on the view of the data as well as on the desired information constituting the patterns.

From the related work of the mining of multi-* data, we notice that there are several works that can be interesting in our research study ; however, each one of them has its specific limitations concerning our work.

Seq-Dim algorithm, proposed in [45], has been proved to be an efficient algorithm, and we consider that it is interesting for mining multi-* data as it mines the sequential patterns and then attaches to it the corresponding frequent multi-dimensional patterns. It avoids the single mining process as in UniSeq due to its high complexity, and it also avoids mining useless multi-dimensional patterns whose sequential patterns are not frequent as in Dim-Seq. However, this algorithm does not support additional sources providing information to the elements in the sequences, and thus it only supports attaching the frequent multi-dimensional information to the sequences as a whole.

We find the star schema structure that is proposed in [53] an interesting structure as it supports multiple level of importance of different tables and avoids the cost of entire join operations. However, to the best of our knowledge, its use is limited to the extension of patterns where no information could be provided to the elements of the patterns.

We consider that the approach proposed in [70] in the domain of contextual sequential pattern mining is a very interesting approach as it considers the contextual information on two levels : the sequence and the elements of the sequence. As to our knowledge, it is the only work that proposes contextual information to the elements of the sequences. This allows providing additional information to the item ids in the sequences. However, we consider that the limitation of this work relative to our research problem is in the way of handling the contextual data of the elements of the sequences. The contextual data of the sequence elements are added to the sequences, and we consider that this increases the complexity of the mining process especially when the length of the sequences is large. Furthermore, although they provide additional information to the elements of the sequences in this approach, this information is added as contexts to the elements. Certain different items could have a similar context ; however, as the item ids are different, they will not be detected as frequent.

In all these approaches, the common limitation relative to our research study is that they are not capable of handling the problem of data similarity on the level of one data source in the multi-* data. Therefore, in our work, we propose to benefit from the advantages of these approaches while handling the limitation represented in the similarity of the items among the sequential data in a multi-* dataset with limited complexity.

2

G_SPM Algorithm for Mining Frequent General Patterns

Contents

In this chapter, we discuss our proposed approach for mining frequent patterns from multi-* data. Such kind of data constitutes a complex and heterogeneous dataset. Despite its complexity, such dataset is rich as it contains various kinds of data. Therefore, the challenge is to manage such kind of data in an efficient manner with low complexity in order to generate rich and valuable information. In this concern, we determine different kinds of relations among data sources and we propose a novel algorithm, G_SPM, that mines multi-* data by relying on these relations in order to handle the problem of item similarity among patterns through obtaining frequent general patterns with low algorithm complexity.

In section 2.1, we describe the different kinds of data sources constituting a multi-* dataset. In section 2.2, we discuss the different kinds of relations that can be determined among data sources. In section 2.3, we discuss the existing approaches that could be used for mining these

different kinds of relations. In section 2.4, we present preliminary definitions used in our proposed algorithm. Finally, in section 2.5, we present G_SPM algorithm for mining general sequential patterns.

2.1 Data sources description in multi-* dataset

A multi-* dataset consists of multiple data sources. These data sources provide data of heterogeneous kinds where each data source provides one or more kinds of data. In this concern, we propose to classify the data sources into two main categories based on the kinds of data that they provide as follows : (1) sequential data sources and (2) descriptive data sources.

2.1.1 Sequential data source

A sequential data source provides data sequences which contain items/itemsets of a specific order where each item/itemset has its own timestamp and an item can not occur more than once in an itemset. A data sequence, represented by user id, could consist of digital activities performed by the users on their system over a specific period of time where an item represents a user's digital activity that is performed at a specific timestamp.

Some examples of sequential data in different domains are as follows. In healthcare domain, sequential data could represent sequences of diseases diagnosed and medications prescribed to patients across time. In e-learning, sequential data could represent the sequences of pedagogical resources (exams, quizzes, exercises, etc) performed by the students on their virtual learning environment (VLE) during an academic semester. In the domain of customer purchase, sequential data represents the sequences of products purchased by each customer across a period of time ; an example of this data is displayed in Table 2.1. Each row in the table represents a customer's id and a data sequence representing the purchased products by this customer. For the sake of clarity of our examples, we use item names instead of item ids in the data sequences in Table 2.1. A sequential pattern that can be obtained from the sequential database represented in the table with $minsup = 4$ is $p :< Sugar, Croissants >$ which appears 6 times in the sequential database.

Customer-id	Sequence
1	<cappuccino, sugar, cigarettes, croissants, brownies>
2	<white-vinegar, cigarettes, milk, cola, cookies>
3	<espresso, sugar, croissants, cookies, soda>
4	<cigarettes, cappuccino, sugar, croissants, brownies>
5	<sugar, espresso, milk, cola, soda>
6	<cappuccino, cola, sugar, croissants, brownies>
7	<espresso, sugar, cola, croissants, cookies>
8	<espresso, sugar, white-vinegar, croissants, cookies>

TABLE 2.1 – Sequential purchases of customers

2.1.2 Descriptive data sources

This category of data sources contains all kinds of descriptive data. It can take the form of different descriptive data attributes. Data attributes can be numerical such as integers or decimals ; other attributes can be textual such as attributes representing date or time, etc. This

category of data sources is divided into two subcategories : descriptive data sources of objects and descriptive data sources of items.

Descriptive data sources of objects

This category of data sources includes all kinds of data describing the objects. It can take the form of data dimensions representing users' demographic information as well as other various descriptive data about users.

For example, in the domain of customer purchases, there could be data sources providing descriptive data about the users in the form of different data dimensions as follows. The dimensions could represent users' demographic information like age, gender, address, employment status, etc. They could also represent descriptive data about the interests of users. The various descriptive data describing the users could be provided by a single data source or by multiple data sources where each data source provides one or more data dimensions representing various kinds of descriptive data, and each dimension has several descriptive attributes. Table 2.2 represents the descriptive data of the users in Table 2.1. The data dimensions in this table are : Customer id, Age category, Gender and Employment. For each customer-id, there is an attribute representing each dimension. For example, row 3 of the table represents customer-2 with the attributes : *young* for the dimension "Age category", *male* for the dimension "Gender" and *unemployed* for the dimension "Employment".

Customer id	Age category	Gender	Employment
1	young	female	employed
2	young	male	unemployed
3	old	female	employed
4	young	female	employed
5	young	male	unemployed
6	old	male	unemployed
7	old	female	employed
8	young	male	employed

TABLE 2.2 – Descriptive data of customers

Descriptive data sources of items

This category of data sources provides data describing the items that constitute data sequences which represent the sequential activities performed by users. It can take the form of different data dimensions coming from single or multiple data sources. Table 2.3 displays the descriptive data of items in the sequences provided in Table 2.1. The data dimensions in this table are : Product id, Product name, Category, Expiry date, Price and Weight. Each product id has an attribute describing each dimension. For example, row 3 of the table has the attributes : espresso, coffee, Casino, Jan-2021, 5 and 250 for the dimensions respectively.

2.2 Proposition for a structure of relations between data sources

A dataset formed of multiple data sources from these two categories would be a heterogeneous dataset. Although the heterogeneity of the dataset makes it complex, it allows extracting rich

Product name	Category	Brand	Expiry date	Price ($)	Weight (g)
cappuccino	coffee	Casino	Dec-2020	5	280
espresso	coffee	Casino	Jan-2021	5	250
sugar	powdered sugar	Daddy	Nov-2021	2.5	930
croissants	pastry	Pasquier	May-2020	4.35	640
cigarettes	tobacco	Marlboro	Dec-2021	7	18.52
cookies	sweets	Nestle	Aug-2020	3.39	356
white vinegar	vinegar	Melfor	Feb-2021	2.30	0.98
milk	dairy product	Regilait	Jan-2021	5.40	750
soda	soft drink	Soda	Mar-2021	1.95	390
cola	soft drink	Coca-cola	Mar-2021	1.8	384
brownies	sweets	Nestle	Oct-2020	3.70	360

TABLE 2.3 – Descriptive data of sequence items of Table 2.1

and valuable information that contains various kinds of data. The main challenge lies in how to extract valuable information from this complex dataset with a limited complexity.

Data sources could have relations among each other where there could be different kinds of relations among different data sources depending on the involved sources and the data that they provide.

As previously discussed in chapter 1 in the domain of contextual sequential pattern mining„ Ziembinski et al., [70] proposed to consider contextual information on two levels : the sequences and the elements of the sequences. We consider that these two types of context can be exploited for determining two kinds of relations between different data sources in a multi-* dataset.In this frame, we determine two types of relations between data sources : contextual relation and complementary relation.

Figure 2.1 presents a 3-source dataset. The sequential data source contains contains sequential data of customers representing customer's sequential activities. The descriptive data source of customers contains descriptive data about the customers : their age, gender and employment. The descriptive data source of items contains descriptive data about the sequence items : name, category, brand, etc. Each data source thus provides data related to a specific perspective : purchase activity, customers or items.

2.2.1 Contextual relation

As previously mentioned, the sequential data source represents the activities of the users by providing sequences of digital activities of the users represented by user ids. The descriptive data source of users provides descriptive attributes about each user, associated with the user id. Although these two data sources provide data of different kinds, the data are provided to the same object, that is the user. Therefore, we can consider that there is a relation between the sequential data source and the descriptive data source of users as they are connected through user id. We view the descriptive data source of users as a contextual data source providing contextual data to the sequential data source ; thus, the descriptive data of a user are considered as contextual data to this user and thus to its data sequence as a whole. We call the relation between these two kinds of data a contextual relation. The contextual relation is highlighted in figure 2.1 where it is between the data source of sequential purchase activity and the descriptive data source of customers.

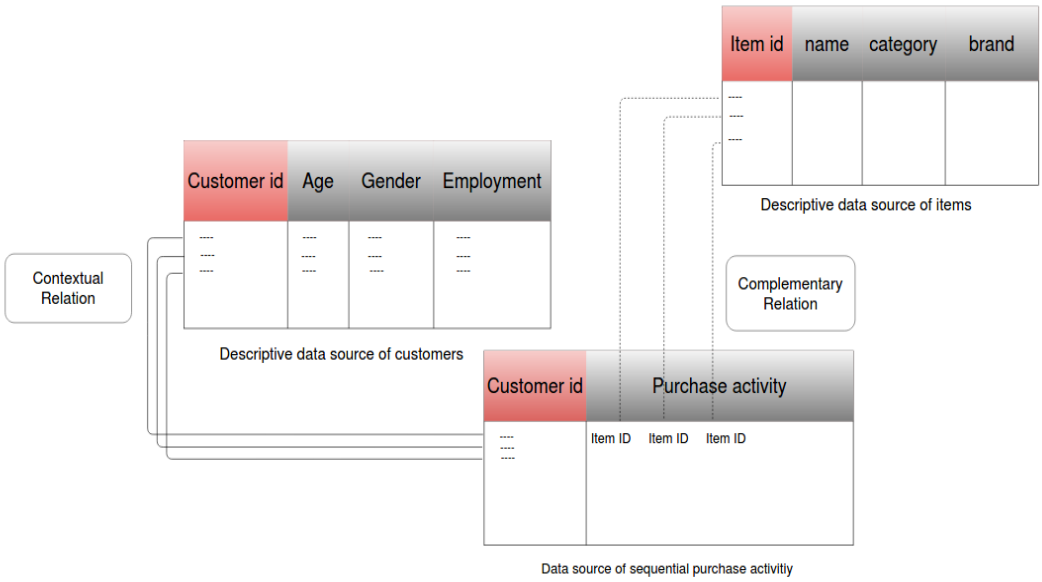

FIGURE 2.1 – A 3-source dataset with contextual and complementary relations

Let us take a concrete example that illustrates this kind of relations. The sequential data source provides the sequential data displayed in Table 2.1 which represent data sequences of products purchased by customers. Table 2.2 provides descriptive data dimensions to each customer. Therefore, we consider the descriptive data of each customer, represented as several descriptive attributes, as contextual data to the data sequence of this customer. For example, the data sequence of *customer*-1 provided by the sequential data source is as follows : $customer\text{-}1 : < cappuccino, sugar, cigarettes, croissants, brownies >$, and the descriptive attributes of *customer*-1 provided by the descriptive data source of users are : $\{young, female, employed\}$. These attributes are thus considered as contextual data provided to the data sequence. From this data, we can obtain a contextual data sequence, that is a data sequence having one or more contexts, of *customer*-1 as follows :

$s = \{young, female, employed\} < cappuccino, sugar, cigarettes, croissants, brownies >$. This contextual data sequence means that an employed young female purchased cappuccino, then she purchased cigarettes, after that she purchased croissants, and she finally purchased brownies.

When we only consider the sequential data source in the mining process with $minsup = 4$, the following sequential pattern would be generated : $p =< sugar, croissants >$ (support = 6). This sequential pattern means that purchasing sugar then croissants is a frequent sequential activity.

When considering the sequential data source as well as the descriptive data sources of items in the mining process, the following contextual sequential pattern would be generated : $\{female, employed\} < sugar, croissants >$. This contextual sequential pattern means that employed females frequently purchase sugar then croissants.

The contextual information provides more precise information that would allow determining the category of customers for whom this pattern is frequent. For example, from such a frequent pattern, we can understand that this sequential pattern is frequent specifically for female customers who are employed, while for instance, it might not be frequent for unemployed females. This shows the importance of the contextual relation in generating more precise frequent patterns and

thus more precise and reliable personalized recommendations.

2.2.2 Complementary relation

Unlike the contextual data source, which provides contextual information to the data sequences of users where the information is provided to the sequence as a whole, the descriptive data source of items provide additional descriptive data to each item in the data sequences where each item, represented by the item id, has several attributes describing it. This contrast allows us to determine a second kind of relations between the descriptive data source of items and the sequential data source providing sequences of items ; these two kinds of data sources are thus connected through the item id.

In the sequential data source, only item ids are provided in the data sequences. An item id represents a unique identifier of the item, and it can not be shared among different items ; therefore, it is the most specific kind of information about the items. However, descriptive attributes are not unique, and they can be shared among different items. for example, the two items cappuccino and espresso share the same attributes : coffee, Casino and 5 for the data dimensions : Category, Brand and Price respectively. Therefore, we view the descriptive data source acts as an additional data source to the sequential source which provides more general information about each item in the sequence. For this reason, we call the relation between the sequential data source and the descriptive data source of items the complementary relation, and we consider that it has an important role in providing general information to the items of the sequences when needed. The additional information about the items makes the resulting patterns more understandable and thus more reliable.

The contextual relation exists as frequently as the number of users in the database ; however, the complementary relation exists as frequently as the number of items in each data sequence of the users. Therefore, the complementary relation exists more frequently than the contextual relation in such a multi-* dataset.

2.3 Proposition for managing relations

We have defined two distinct kinds of relations in such a multi-* data. The contextual relation provides descriptive data to the users ; this relation allows adding contextual information to the data sequences in order to obtain a more *precise* frequent patterns as discussed in [48]. While the complementary relation provides descriptive data to the items of the sequences ; this relation allows adding more general data to the items and thus to the sequences in order to obtain a more general form of frequent patterns. Therefore, each kind of these two relations has a different role in the mining process. From our point of view, the first relation adds preciseness, while the second one adds generality to the resulting frequent patterns. Managing the data sources of a multi-* dataset together allows maintaining these relations among the data sources. Therefore, taking advantage of both relations among different data sources in the mining process of a multi-* dataset would allow obtaining a rich and unique form of frequent patterns containing specific and general information at the same time.

In order to perform the mining process and extract the needed information from a multi-* dataset, we consider the data source that contains the most important and indispensable kind of data needed in order to extract the needed information as the *main data source*, and we start by mining this data source.

In our dataset, we consider that the sequential data source as the main data source for two main reasons. Our main objective is to analyze and understand users' behaviour where the beha-

vior is sequential. This kind of data is provided by the sequential data source ; therefore, if there is only one data source to be considered in the mining process, then it should be the sequential data source as it provides the indispensable sequential data which, when mined, generates frequent sequential patterns that would help in understanding the behavior. In addition, the common point between the two different kinds of relations among data sources is that different kinds of descriptive information is provided to the sequential data source, whether to the data sequences or their items. The contextual relation allows providing contextual data to the data sequences of the users. On the other hand, the complementary relation allows providing additional descriptive data to the items in the data sequences. Therefore, for these reasons, we consider the sequential data source as the main data source in the multi-* dataset. The other data sources are thus considered as additional sources that provide data to the main data source allowing to generate more sequential patterns containing various kinds of information.

From these relations, we understand that the approach of mining data sources in a separate manner is inadequate to a multi-* dataset as it would lead to the loss of relations existing between different kinds of data sources. On the other hand, the approach of integrating all data sources of different kinds together would be inefficient as the complexity of the mining process would be high due to data heterogeneity. Due to the inefficiency of mining a multi-* data using existing approaches and obtaining valuable results in a limited complexity, we propose to take advantage of the multi-* data by managing the relations existing between the data sources in order to obtain valuable information from all kinds of data sources in the dataset.

2.3.1 Managing contextual relation

Various algorithms in the literature were proposed in order to mine sequential patterns associated with contextual data in the form of multi-dimensional sequential data [45, 49, 70]. One algorithm is Seq-Dim algorithm, which is proposed by Pinto et al., [45] and discussed in section 1.3 of chapter 1. This algorithm first mines sequential patterns ; then for each generated sequential pattern, it associates its frequent multi-dimensional information. This multi-dimensional information contains descriptive information to the users which could thus be considered as contextual information to the sequential pattern. This algorithm, among others, is therefore efficient for managing the contextual relation and extracting frequent sequential patterns associated with their contextual information.

Therefore, the contextual relation can be efficiently managed using existing approaches in the literature which mine sequential data associated with their relative contextual data. We thus use Seq-Dim algorithm in order to manage the contextual relation in the multi-* dataset.

2.3.2 Naive approaches for managing complementary relation

If we consider taking advantage of the descriptive data sources of items in the sequential mining process of the data provided by the sequential data source, then we can have additional descriptive attributes for each item in the sequence which allows to better understand these items and thus the sequences.

The multi-dimensional data in the work of Pinto et al. [45] can be considered as contextual data which describe the whole sequence where there are no descriptive attributes for each item in the sequence. Therefore, SeqDim approach does not support such kind of descriptive data attributes which describe each item in the sequence and not the sequence as a whole, and it thus does not support managing the complementary relation.

However, traditional sequential pattern mining algorithms which are designed to mine a single

data source can be used to manage the complementary relation and thus mine the data coming from the sequential data source and the descriptive data source of items by adopting two naive approaches. The first approach is substituting the item ids in the sequences with their set of descriptive attributes, and the second approach is adding to each item in the sequence its set of descriptive attributes. We describe these two approaches as follows.

Substituting item id with its descriptive attributes in sequences

In this approach, the ids of the items in the sequences are substituted with their descriptive attributes to obtain sequences of itemsets containing descriptive attributes for each item. In contrary to the item id in the sequence, which is the most specific kind of data, the descriptive attributes of items represent a more general kind of data. Let us consider the example of the data sequence of *customer*-1 in Table 2.1 : $<$ *cappuccino*, *sugar*, *cigarettes*, *croissants*, *brownies* $>$. By replacing the item names of the sequences by their respective descriptive attributes from Table 2.3, the following sequence is obtained : $<$ {*coffee*,*casino*,*Dec*-2020, 5, 280}, {*powdered sugar*, *Daddy*,*Nov*-2021, 2.5, 930}, {*tobacco*,*Marlboro*,*Dec*-2021, 7, 18.52}, {*pastry*, *Pasquier*, *May*-2020, 4.35, 640}, {*sweets*,*Nestle*,*Oct*-2020, 3.70, 360} $>$.

Obviously, this obtained sequence is larger than the traditional sequences provided by the the sequential data source as it contains a set of 5 descriptive attributes in the place of an item id for each item in the sequence. In addition, this sequence is more complex than traditional sequences as it contains multiple and heterogeneous descriptive attributes.

The advantage of this approach is in generating a significantly larger number of frequent patterns especially when the number of attributes is larger. However, as the item ids are replaced by their attributes in the sequences, this leads to obtaining frequent patterns which only contain general information while loosing all possible frequent patterns containing specific information (item ids). Whereas the quality of the frequent patterns containing only descriptive attributes is lower than those containing item ids The more item ids included in a sequence, the higher is the preciseness of the sequence and thus the better is its quality.

In addition, although this approach generates a large number of frequent patterns, there is a high level of redundancy among patterns where all possibilities of the frequent patterns containing attributes are generated. for example, if we substitute the item names in Table 2.1 by their descriptive attributes in Table 2.3, a sequential pattern generated from this sequential dataset is : p_a : $<$ {*powdered-sugar*, *Daddy*, *Nov-2021*, 2.5, 930}, {*pastry*, *Pasquier*, *May-2020*, 4.35, 640} $>$ with support = 6. Another sequential pattern extracted from this dataset is also frequent, p_b : $<$ {*Daddy*, *Nov-2021*, 2.5, 930}, {*pastry*, *Pasquier*, 4.35, 640} $>$ with support = 6. p_b is a subset of p_a where p_a contains one additional attribute in the itemsets at the first and the second positions in the pattern. Therefore, p_b is a redundant pattern as it is already included in p_a.

The length and the complexity of the sequential data as well as generating redundant patterns makes the mining process of this approach of high complexity, and the complexity becomes higher with the increase in the length of patterns and thus with the number of descriptive attributes.

For all these reasons, we consider that this naive approach is inefficient to mine such kind of data and thus to manage the complementary relation.

Adding descriptive attributes to item id in the sequences

In this second naive approach, we propose to add descriptive attributes of items to their associated ids in the sequences to obtain at each position an itemset containing an item id along

with the set of its descriptive attributes at each position in the sequence. Therefore, the lengths of these data sequences are larger than those of the first naive approach where an item idis added to each position in the data sequence. For example, let us consider the same data sequence of *customer*-1 in Table 2.1. Adding the descriptive attributes of items provided in Table 2.3 allows obtaining the following sequence : < {*cappuccino*,*coffee*,*casino*,*December*-2019, 5, 250}, {*table-sugar*,*Daddy*,*November*-2019, 2.5, 930}, {*cigarettes*,*tobacco*,*Marlboro*,*November*-2019, 7, 18.52}, {*croissants*, *pastry*, *Pasqiuer*, *February*-2020, 4.35, 640}, {*brownies*,*sweets*,*Nestle*,*February*-2020, 3.70, 300} >.

In such kind of sequences, the item names as well as the descriptive data are involved in the mining process, and the resulting patterns could include two kinds of data : item ids and descriptive attributes.

As the item ids are included in the data sequences, this approach allows generating sequential patterns containing specific information, and these patterns are of two kinds : sequential patterns containing only item ids and others containing item ids and descriptive attributes. Such kinds of patterns, which could not be generated by the first naive approach, increase the preciseness of the results and thus enhance the quality. However, the increase in lengths of input data sequences could lead to a higher complexity of the mining process. In addition, when an item id is frequent, then by definition, all its descriptive attributes are frequent. And as the most specific data is frequent, a sequential pattern containing the item id generated while there is no need to generate sequential patterns containing its attributes. However, in this approach, when an item id is frequent, all possibilities of patterns containing the item id with its descriptive attributes are considered as frequent. In this case, we consider that the sequential patterns containing descriptive attributes are redundant as, by definition, they are known to be frequent Therefore, they would impact the quality of the results with no added value.

For these reasons, we consider that this second naive approach is also not efficient for mining such kinds of data and thus managing the complementary relation.

Due to the inefficiency of the existing naive approaches to manage the complementary relation, we propose an approach to manage this kind of relations in a limited complexity and without redundancy among the resulting patterns where it intends to take advantage of the descriptive data source of items only when the data in the sequential data source are not sufficient to generate an adequate number of sequential patterns.

2.4 Preliminary definitions

In this section, we introduce concepts and their associated formal definitions and notations that will be used in our proposed approach for managing the complementary relation between the sequential data source and the descriptive data source of items.

2.4.1 Infrequent patterns

As previously discussed in chapter 1, sequential pattern mining involves extracting frequent sequences from the sequential database. The non-extracted patterns from the mining process are called infrequent patterns, and they are defined as follows.

Definition 2.4.1. *Infrequent pattern*

Let us consider a minimum support threshold, denoted as minsup. A pattern s is infrequent in a sequence database S_D *if* $1 \leq support_{S_D}(s) < minsup$. *In other words, s is infrequent if it occurs less than minsup times in* S_D.

Example 2.4.1. *Infrequent pattern*

Let us consider an example of the sequential database S_D provided in Table 2.1 with minsup = 4. The pattern $s =< milk, cola, cookies >$ has a support value = 1. s is said to be an infrequent pattern as $support_{S_D}(s) < minsup$.

2.4.2 Promising patterns

In the set of frequent patterns, some patterns have support values that are lower but not far from the value of *minsup*; therefore, these patterns are infrequent but are not far from being frequent. We call these patterns *promising patterns* as they are promising to be frequent. Patterns that are similar to this kind were discussed in [67] in the context of mining local patterns in multiple databases, and they are called suggesting patterns.

We distinguish between the promising patterns and the infrequent ones having support values far from *minsup* as follows. We first determine a second predefined minimum support threshold called *minsupProm* whose value is lower than the value of *minsup* but is close to it. An then, based on the minimum support thresholds, *minsup* and *minsupProm*, we define a promising pattern as follows.

Definition 2.4.2. *Promising pattern*

Let us consider a sequential database S_D and an infrequent pattern v. v is said to be a promising pattern iff $minsupProm \leq Support_{S_D}(v) < minsup$

Example 2.4.2. *Promising pattern*

Let us consider an example of the sequential database S_D provided in Table 2.1 with minsup = 4 and minsupProm = 3. The pattern $p_1 :< cappuccino, sugar, croissants >$ has a support value = 3, and the pattern $p_2 :< espresso, sugar, croissants >$ has a support value = 3. p_1 and p_2 are said to be promising patterns as $minsupProm \leq Support_{S_D}(p_1) < minsup$ and $minsupProm \leq Support_{S_D}(p_2) < minsup$ respectively.

2.4.3 Sequence and item similarity

We define two new measures of similarity, sequence similarity and item similarity. The sequence similarity concerns the sequences or subsequences occurring in a sequential database, and the item similarity concerns the items in the sequences. We discuss these two similarity measures in details as follows.

Sequence similarity

In certain cases, there could be similarities among sequences. Therefore, we propose a new similarity measure, called sequence similarity measure, that is concerned in measuring the similarity between two distinct data sequences.

The items of data sequences follow a certain order where they occur at specific positions in the sequences. The sequence similarity measure that we define relies on the identity of items at the same positions (index) in sequences. Therefore, we measure the similarity among patterns by checking if the items at the same positions in the patterns are the same. For this similarity measure, we choose to consider patterns that strictly have the same length. In certain cases, there could be identities between a certain number of items in the sequences which allows obtaining similar sequences. We define similar sequences as follows.

Definition 2.4.3. *Similar sequences*

Let us consider the two sequences as follows : $s_1 =< item1_1, item1_2, ...item1_l >$ *and* $s_2 =< item2_1, item2_2, ...item2_l >$. l *is the length of* s_1 *and* s_2 *according to definition 1.1.2, minSsimilarity is a minimum predefined percentage similarity between two sequences and position* $= \{1, ..., n\}$. $nbSim_{(s_1,s_2)} = \sum_{i=1}^{n}$ *number of positions where* $item1_i = item2_i$. *ssimilarity represents the percentage similarity between two sequences, and it is calculated as :* $ssimilarity_{(s_1,s_2)} = (nbSim_{(s_1,s_2)}/l) * 100$.
s_1 *and* s_2 *are said to be similar sequences iff* $ssimilarity \geq minSsimilarity$.

Example 2.4.3. *Similar sequences*

Let us consider an example of the two sequences in the sequential database of Table 2.1 as follows : $< s_1 : cappuccino, sugar, croissants >$ *and* $< s_2 : espresso, sugar, croissants >$, *and let us consider that* $minSsimilarity = 60\%$. *These two sequences have two common items at positions 1 and 2 of the patterns, and they have one item difference at position 0 of the patterns.* $ssimilarity_{(s_1,s_2)} = 66\%$ *which is greater than minSsimilarity ; therefore,* s_1 *and* s_2 *are considered as similar patterns.*

Item similarity

The descriptive data source of items contains descriptive data about items represented in the form of several data attributes. Different items could have common attributes between them ; therefore, we propose an item similarity measure that relies on the identity between attributes of the items. There could be identities between items allowing to consider them as similar items. We define similar items as follows.

Definition 2.4.4. *Similar items*

Let us consider two items i_1 *and* i_2 *with number of attributes* $= n_a$. $minIsimilarity$ *is a minimum predefined percentage similarity between two different items.* $commAtt(item_1, item_2)$ *is the number of common attributes between two different items. isimilarity represents the percentage similarity between two items, and it is calculated as :* $isimilarity(item_1, item_2) = (commAtt(item_1, item_2)/n_a) * 100$. i_1 *and* i_2 *are said to be similar items iff* $isimilarity(item_1, item_2) \geqslant minIsimilarity$.

Example 2.4.4. *Similar items*

Let us consider the two items cappuccino and espresso provided in Table 2.3 and $minIsimilarity = 60\%$. *The descriptive attributes of the item cappuccino are :* $\{coffee, Casino, Dec\text{-}2020, 5, 280\}$, *and those of the item espresso are :* $\{coffee, Casino, Jan\text{-}2021, 5, 250\}$. *The list of common attributes between the two items is :* $commAtt(cappuccino, espresso) : \{coffee, Casino, 5\}$. *For these two items,* $commAtt(cappuccino, espresso) = 3$ *; therefore,* $isimilarity(cappuccino, espresso) = 60\%$, *and the items cappuccino and espresso are considered as similar items.*

In certain cases, two items are not found similar ; however, if we consider the descriptive dimensions having numerical attributes, there could be one or more dimensions having a relatively small difference between the values of their attributes, but as these two attributes do not exactly have the same value, they are considered as two different attributes.

In order to overcome this limitation and widen the range of detecting similar items, we determine intervals of values for each data dimension having numerical attributes. In this case, instead of comparing the attribute values of two items, we check if the two attribute values of a data dimension are within the same interval that is determined for this dimension. If they are within the same interval, we consider that the items have a common attribute for this data

dimension, and it is thus added to the set of common attributes between the items. This common attribute is represented as the following interval : [lower attribute value, higher attribute value].

Let us consider an example of the two items *brownies* and *cookies* from Table 2.3. The set of common attributes between the two items is : $commAtt(brownies, cookies) :< sweets, Nestle >$ and thus $simItem(brownies, cookies) = 40\%$. $simItem(brownies, cookies) < minSimItem$, and the items are not considered as similar items. However, let us consider the dimension "Weight" that has numerical attributes, the attribute value of the item *brownies* = 360, and that of the item *cookies* = 356. The difference between the attribute values of the two items is relatively small. If we consider an interval of range 30 for the attribute values of this data dimension, we will find that the attribute values of the two items are within the same interval and are thus considered as a common attribute that is represented in the form of the following interval : [356 and 360]. The new list of common attributes between the two items becomes : $commAtt(brownies, cookies) :< sweets, Nestle, [356 - 360] >$. Therefore,
$commAtt(brownies, cookies) = minSimItem$, and the items are thus considered as similar items.

2.4.4 Sequence generalization

In this part, we discuss the generalization of sequences which allows obtaining general sequences, and we discuss the calculation of support values of general sequences.

Forming general sequences

Two similar sequences having similar items may be combined to form one or more new sequences. We consider this new sequence as a general sequence as it represents the two patterns from which it is formed, and we consider these two patterns as the original patterns of the general sequence.

Let us consider two similar sequences having items that are different at one or more of their positions. In certain cases, the two items at the same position in two similar sequences could be similar items. As presented in definition 2.4.4, two similar items have a set of common attributes between them. We thus consider that each two similar items are represented by the set of their common attributes. In this frame, when we have two similar items at the same position in two similar sequences, we propose to represent these items by their set of common attributes in their position in the sequences by substituting these two items with this set. This set of attributes represent more general data than item ids ; therefore, we call these newly formed sequences *general sequences*. A general sequence is a sequence formed from two similar sequences having similar items at one or more positions in the sequences, and it contains either an item id or a set of descriptive attributes at each position in the sequence.

The process of forming general sequences depends on the number of positions in the sequences having different items. Therefore, we first discuss this process for the sequences having only one position containing different items, and then we discuss this process for sequences having more than one position containing different items.

Similar sequences with one item difference

Two similar sequences, named *similar-sequence1* and *similar-sequence2* could have an item difference at only one position in the sequences, which is the simplest case. When these different

items are similar items, we substitute them with the set of their common attributes at their position in the two sequences.

In this case, the only difference between the two similar sequences is substituted with common attributes, and only one general sequence is formed. The general sequence represents *similar-sequence1* and *similar-sequence2* which are its original sequences.

Example 2.4.5. *Similar sequences with one item difference*

Let us consider the example in Table 2.1 of similar sequences with only one position having similar items as follows :
$S_1 :< cappuccino, sugar, croissants >$ *and* $S_2 :< espresso, sugar, croissants >$. *As provided in example 2.4.4, the similar items cappuccino and espresso in these sequences have the following set of common attributes :* $commAtt(s_1, s_2) = \{coffee, Casino, 5, [250, 280]\}$. *Therefore, we substitute these two items at their positions in the sequences with the set of their common attributes, and we obtain a new sequence* s' *as follows :* $s' :< \{coffee, Casino, 5, , [250, 280]\}, sugar, croissants >$. *This sequence does not contain specific data at position 1 ; however, it contains a set of common attributes which represent the two items cappuccino and espresso making it a more general sequence than the similar sequences. The original sequences of* s' *are* s_1 *and* s_2.

Similar sequences with two or more item differences

The item differences between two similar sequences can occur at two or more positions in the sequences. The generalization process of such sequences is more complex than for those having only one item difference. As in the previous case, we propose to substitute each two similar items at the same position in the sequences with the set of their common attributes. As there are more than one difference in the similar sequences, substituting the items with the set of their common attributes at one position does not form a single general sequence ; however, it forms two general sequences, named *general-sequence1* and *general-sequence2* where *similar-sequence1* is represented by *general-sequence1* and *similar-sequence2* is represented by *general-sequence2* with differences at the current position of the similar items where they are substituted with their set of attributes. In this case, each general sequence has its two original sequences. We determine the original patterns of these two general sequences as follows.

Determining the original sequences for each general sequence is based on : (1) the similar sequences that they represent and (2) the two similar items at the current position of item difference.

Therefore, each original sequence of general-sequence1 represents similar-sequence1 with each of the two similar items at the current position of item difference in the sequence. On the other hand, each original sequence of general-sequence2 represents similar-sequence2 with each of the two similar items at the current position of item difference in the sequence. In order to clarify this process, we provide an example as follows.

Example 2.4.6. *Similar sequences with two item differences*

Let us consider the following two sequences in Table 2.1 : $s_3 :< cappuccino, sugar, croissants, brownies >$ *and* $s_4 :< espresso, sugar, croissants, cookies >$. s_3 *and* s_4 *are similar sequences according to definition 2.4.3. The item differences are at positions 0 and 3 in the similar sequences. At position 0, the two items cappuccino and espresso are similar items, and they have the following set of common attributes :* $\{coffee, Casino, 5, [250, 280]\}$. *Therefore, we substitute these two items in their position in the sequences with the set of their common attributes to form two general sequences* $s_3^{'}$ *and* $s_4^{'}$ *as follows :* $s_3^{'} :< \{coffee, Casino, 5, [250, 280]\}, sugar, croissants,$

brownies > *and* $s_4^{'} < \{coffee, Casino, 5, [250, 280]\}, sugar, croissants, cookies >$. $S_s^{'}$ *represents the similar sequence* s_3, *while* $s_4^{'}$ *represent the similar sequence* s_4.

The original sequences of $s_3^{'}$ *are :* $s_{3a} :< cappuccino, sugar, croissants, brownies >$ *and* $s_{3b} :< espresso, sugar, croissants, brownies >$. *Similarly, the original sequences of* $s_4^{'}$ *are :* $s_{4a} :< cappuccino, sugar, croissants, cookies >$ *and* $s_{4b} :< espresso, sugar, croissants,$ *cookies* >. *We notice from this example that the original sequences may or may not exist in the sequential database where* s_{4a} *exists in the sequential database while* s_{4b} *does not exist.*

In the previous process, we form general sequences from the similar sequences having similar items. We can now calculate the support values of these formed sequences.

Calculating support values of general sequences

After forming general sequences, we now propose a method to calculate the support values of these sequences. Let us precise that these sequences do not occur in the sequential database as they are general sequences. Therefore, the support calculation of such sequences differs from that of the original sequences that are already found in the database.

The general sequence represents the two original sequences from which it is formed where the set of common attributes between the two similar items substitutes them in the sequences. Therefore, the number of occurrences of the new sequence is equal to the sum of the number of occurrences of the two original sequences in the sequential database. The support value of the sequence is thus calculated as the sum of the support values of its original sequences.

When there is only one position having an item difference between two similar sequences, one general sequence is formed. In this case, the original sequences of this general pattern are the similar patterns, and they are found in the sequential database. Therefore, the support value of the the general sequence is calculated as the sum of the support values of the two similar sequences.

On the other hand, when there is more than one position having item differences in the similar sequences, the original sequences may or may not be found in the sequential database. In case an original sequence is not found, its support value is thus considered equal to zero. Therefore, the support value of each general sequence is calculated as the sum of the support values of its two original sequences.

Example 2.4.7. *Similar sequences with one item difference*

Referring to example 2.4.5, the two similar sequences are : $s_1 :< cappuccino, sugar,$ *croissants* > *having a support value = 3 and* $s_2 :< espresso, sugar, croissants >$ *having a support value = 3. The following general sequence is formed :* $s' :< \{coffee, Casino, 5, [250, 280]\}$ $, sugar, croissants >$. *The original sequences of* s' *are* s_1 *and* s_2. *Thus, the support value of* s' *= 6, which is the sum of support values of* s_1 *(3) and* s_2 *(3).*

Example 2.4.8. *Similar items with two item differences*

Referring to example 2.4.6, the similar sequences are : $s_3 :< cappuccino, sugar, croissants,$ *brownies* > *having a support value = 3 and* $s_4 :< espresso, sugar, croissants, cookies >$ *having support value = 3. The general sequences formed are as follows :*
$s_3^{'} :< \{coffee, Casino, 5, [250, 280]\}, sugar, croissants, brownies >$
and $s_4^{'} < \{coffee, Casino, 5, [250, 280]\}, sugar, croissants, cookies >$. *The original sequences of* $s_3^{'}$ *are :* $S_{3a} :< cappuccino, sugar, croissants, brownies >$ *with a support value = 3 and* $s_{3b} :< espresso, sugar, croissants, brownies >$, *with a support value = 0 as it does not exist in the sequential database. Therefore, the support value of* $s_3^{'}$ *= 3. Similarly, the original patterns*

of $s_4^{'}$ *are :* $s_{4a} :< cappuccino, sugar, croissants, cookies >$ *with a support value = 0 as it does not exist in the sequential database and* $s_{4b} :< espresso, sugar, croissants, cookies >$ *with a support value = 3. Therefore, the support value of* $s_4^{'} = 3$.

2.5 G_SPM algorithm for mining general sequential patterns

In this section, we present G_SPM algorithm for mining frequent general sequential patterns from a multi-* dataset in order to handle the problem of item similarity and generate frequent patterns containing rich and valuable information. G_SPM is presented in algorithms 1 and 2. The multi-* dataset (MS_D) is used as an input to G_SPM.

As discussed in section 2.2, we make the choice to consider the sequential data source as the main data source in the multi-* dataset. This source contains the sequential data representing the behavior of users which is the indispensable data needed to achieve our main objective of understanding users' behavior.

2.5.1 Mining frequent sequential patterns

G_SPM starts by a mining process of the sequential data source in order to generate frequent sequential patterns. For this mining process, any traditional sequential pattern mining algorithm can be used.

As known in the domain of frequent pattern mining, if the support of a candidate sequential pattern is above the minimum predefined support threshold *minsup*, then it is detected as frequent ; otherwise, it is considered as infrequent and is thus discarded as provided in definition 1.1.5.

The number of generated frequent patterns increases as the *minsup* value decreases. However, when the value of *minsup* is relatively low, it could result in a huge number of frequent patterns. This could lead to a lower quality of the resulting patterns as certain patterns would be of low support values. Therefore, we consider that the value of *minsup* should be relatively high in order to avoid the generation of a huge number of frequent patterns thus impacting the quality.

On the other hand, certain characteristics of the data could lead to certain data limitations which thus limits the number of generated frequent patterns especially when the value of *minsup* is relatively high. We discuss these data limitations as follows.

Data limitations

We may face a problem of lack of data in the sequential data source. Several reasons could lead to this problem. The number of users can be limited ; therefore, the number of data sequences is limited which leads to a limited number of sequential patterns generated from these sequential data. In addition, there could be data sparsity among the data sequences which limits the number of generated sequential patterns. Another important reason for the problem of lack of data could be the similarity among the behavior of users. This similar behavior is represented in the form of similar data sequences or subsequences where they contain items which are similar but not exactly the same. The mining process of these sequences could lead to obtaining similar patterns where these patterns may also contain similar items. Each of these patterns thus have lower support values and are not detected as frequent. This results in a limited number of sequential patterns and thus the loss of possible interesting patterns.

In our approach, we propose to take advantage of the complementary relation and consider the descriptive data source of items as an additional data source that provides descriptive data to the sequential data provided by the main source in order to handle the problem of similarity and compensate the loss of patterns that could have been frequent by generating more frequent patterns that contain rich and various information coming from the two data sources, named general patterns.

The main problem is that we can not consider all infrequent patterns as interesting patterns. For example, the data coverage of the patterns of a very low support value compared to *minsup* is very low as it represents a small number of data sequences. In addition, these patterns could represent rare or exceptional behavior which is of no interest in our work. For these reasons, we consider that an interesting infrequent pattern is the one that is not far from being frequent, that is its support value is lower than *minsup* but not far from it. These patterns are called promising patterns according to definition 2.4.2.

Promising patterns could be infrequent for two main reasons. The first reason is that the users do not perform these sequences of activities frequently enough according to the value of *minsup*; thus, they are not detected as frequent in the sequential database. The second reason is the problem of item similarity which leads to the loss of certain patterns that could have contained valuable information and could have been frequent. We are specifically interested in detecting this latter kind of patterns; therefore, we choose to determine the promising patterns and analyze them further to obtain frequent patterns out of them when possible.

Determining frequent and promising patterns

Obviously, a sequential pattern mining process on the sequential data using *minsup* allows generating only the frequent patterns while discarding all the infrequent ones. Different strategies could be imagined in order to obtain the promising patterns, or to obtain the infrequent patterns and consequently obtain the promising patterns out of them. In this frame, we determine a specific strategy in order to obtain not only the frequent patterns but also the promising ones as follows.

We first determine *minsupProm*, a second predefined minimum support threshold that is previously described in section 2.4. *minsup* allows determining frequent patterns where *minsupProm* allows determining promising patterns. These two minimum support thresholds are thus provided as an input to G_SPM.

In order to determine frequent and promising patterns, we perform a sequential pattern mining process on the sequential data source (S_D), using any traditional sequential pattern mining algorithm, where the minimum support threshold used in this process is *minsupProm* (line 2, algorithm 1).

The set of patterns generated by this mining process (P) contains frequent as well as promising patterns. The second step of the algorithm is to distinguish the set of promising patterns (F_p) from the set of frequent ones (F_f). This is done through a post-processing step to the sequential mining process as follows. For each pattern generated by the mining process, if its support value is greater than or equal to *minsup*, then it is considered as a frequent pattern; while if its support value is less than *minsup*, then it is considered as a promising pattern (line 13-16, algorithm 1).

We aim at analyzing and understanding these patterns in order to distinguish the patterns that were frequent but have become infrequent due to the problem of item similarity from the patterns that are already infrequent despite this problem. For this sake, we undergo a further analysis of these patterns by analyzing their content.

2.5.2 Determining similar patterns among promising patterns

Among the promising patterns, there could be certain patterns representing similar behavior of users. These patterns are thus considered as similar patterns where they contain the same items with different items at certain positions in the patterns. Therefore, we aim at detecting similar patterns among the promising ones.

We have previously defined and discussed similar sequences in section 2.4, definition 2.4.3. A promising pattern is an infrequent sequential pattern ; and as it is sequential, this measure of similar sequences can be used in order to measure the similarity between two promising patterns, which we call the pattern similarity, and detect similar patterns among them.

As in definition 2.4.3, we choose to compare promising patterns of the same length in order to obtain a pattern percentage similarity between each two promising patterns. Therefore, for each two promising patterns, we first compare their lengths (line 17, algorithm 1). If these patterns have the same length, we calculate the pattern similarity between these two patterns as follows. We first compare items at each position in the patterns to count the number of items which are the same ($nbSim$) and save the positions containing different items (pos) (lines 21-25, algorithm 1). From $nbSim$, we calculate the pattern similarity as a percentage similarity ($psimilarity$) (line 26, algorithm 1). If $psimilarity$ is higher than or equal to a minimum predefined pattern similarity ($minPsimilarity$), the two patterns are considered similar, and the set of positions containing different items is saved for further uses (lines 27-31, algorithm 1). If a promising pattern is not similar with any other pattern in the set of promising patterns, we consider that this pattern is not useful for handling the problem of similarity, and it is thus discarded.

Similar patterns, which share common information, represent similar sequential behavior of users. However, these patterns contain different items at certain positions. In order to analyze further the similarity in sequential behavior, we choose to analyze further the similar patterns by analyzing the different items at the same positions in similar patterns.

2.5.3 Determining similar items among similar patterns

The sequential data source only provides the data sequences which consist of item ids representing unique item identifiers. Therefore, the generated patterns consist of item ids as well. For each two promising patterns, we can only check if the items are the same or different which allows measuring the similarity between these patterns. However, measuring the similarity between items using only the item identifiers is not precise. Therefore, in order to measure the similarity between items, which we call the item similarity, we need additional data about these items. For this sake, we propose to take advantage of the descriptive data sources of items which provides descriptive attributes for each item.

According to definition 2.4.4 that is previously introduced and discussed in section 2.4, we compare two different items as follows. We first retrieve the set of descriptive attributes for the two items from the descriptive data source of items (IA_D database) (line 34, algorithm 1). Then, each two attributes representing the same data dimension are compared between the two items to obtain a set of common attributes between them ($setCommAtt$)(lines 35-38, algorithm 1).

As discussed in section 2.4, intervals of attributes for numerical data dimensions are set in order to widen the range of detecting common attributes between two different items and overcome the limitations of close values of attributes.

After obtaining this information, the item similarity ($isimilarity$) between the two items is then calculated according to their number of common attributes ($commAtt$) as a percentage similarity (line 39, algorithm 1). If $isimilarity$ is greater than or equal to a minimum predefined

item similarity($minIsimilarity$), then the items are similar, and their set of common attributes is saved for further uses (lines 40-44, algorithm 1).

This allows us to understand that due to the similarity among the items in similar patterns, certain patterns take the form of more than one pattern. These patterns are of lower support values which would make them infrequent patterns and thus lead to the loss of information.

2.5.4 Generalization of similar patterns having similar items

In order to overcome the problem of similarity among patterns and their items which leads to the loss of information, we propose a generalization process as follows. The process of sequence generalization is introduced in 2.4.4. In G_SPM, we apply this generalization process on similar patterns having similar items at the same positions in the patterns to obtain frequent general patterns as follows. For each position in two similar patterns having similar items, we substitute the similar items with their set of common attributes ($setcommAtt$) while saving the original patterns of the two similar patterns (lines 1-6, algorithm 2). The obtained pattern/patterns contain either item ids or attributes at each position in the patterns which make them general patterns. Then, we calculate the support values of general patterns in order to check if they are frequent. The support value of each general pattern is the sum of the support values of its two original patterns (lines 15 and 19, algorithm 2). The support values of the original patterns are retrieved either from the set of promising patterns or from the set of frequent ones, and if the support values of the original patterns are not found in any of these two sets, it is considered as 0 (lines 9-14, algorithm 2). If the support value of a general pattern is higher than or equal to $minsup$, then it is considered a frequent general pattern ; otherwise, it is considered as infrequent and is thus discarded (lines 16 and 20, algorithm 2).

The steps of determining similar items and the generalizations are recursively repeated at each position having similar items in two similar patterns until the number of generalizations (gen) reach the maximum allowed number of generalizations ($maxGen$). When this number is reached, we stop the generalization process even if there are still possibilities for generalizations.

In our proposed approach, we provide a solution to the problem of item similarity among patterns by generating frequent general patterns from promising patterns that are similar and that contain similar items at the same positions in the patterns. We can point out some limitations in this approach ; however, these limitations are justified as they are accompanied with advantages. We discuss them as follows.

The frequent general patterns that are generated from our generalization algorithm contain more general, thus less specific information at specific positions in the patterns. Although the general patterns are less specific, they have a higher data coverage because each general pattern represents two patterns. In addition, our algorithm allows an increase in the number of frequent patterns which solves the problem of the limited number of generated frequent patterns that is caused by the problem of similarity.

The general patterns that are generated from our algorithm contain different kinds of information, represented as either item ids or descriptive items at each position in the patterns. This makes these patterns of a heterogeneous and complex nature. However, this heterogeneity makes them rich patterns as they contain rich and various information. The more various information the pattern contains, the richer is the pattern and the wider is the knowledge extracted through this pattern.

The complexity of our proposed algorithm is higher than the complexity of the sequential pattern mining algorithm. However, the nature of the data to be mined is different in the two algorithms. While the sequential mining algorithm manages a single data source (the sequential

Algorithm 1 G_SPM($MS_D, minsup, minsupProm, minPsimilarity, minIsimilarity, maxGen, maxDiff$)

1: $P, F_p, F_f, F_g \leftarrow \emptyset$
2: $P \leftarrow SPM(S_D, minsupProm)$
3: **for** each $p \in P$ **do**
4: $prom\_freq(p, minsup)$
5: **for** each $p_i, p_j \in F_p$ **do**
6: $sim\_pattern(p_i, p_j)$
7: **if** $pos! = \emptyset$ **then**
8: $gen \leftarrow 0$
9: **while** $gen < maxGen$ **do**
10: **for** each position x in pos **do**
11: $setCommAtt \leftarrow sim\_item(p_i[x], p_j[x])$
12: **if** $setCommAtt! = \emptyset$ **then**
13: $generalization(p_i, p_j, x, setCommAtt)$
Function $prom\_freq(p, minsup)$
14: **if** $support(p) \geq minsup$ **then**
15: $F_f \leftarrow F_f \cup p$
16: **else**
17: $F_p \leftarrow F_p \cup p$
Function $sim\_pattern(p_i, p_j)$
18: $pos \leftarrow \emptyset$
19: **if** $(length(p_i) = length(p_j))$ **then**
20: $k \leftarrow 1$, $nbSim \leftarrow 0$
21: **while** $k \leq length(p_i)$ **do**
22: **if** $p_i[k] = p_j[k]$ **then**
23: $nbSim \leftarrow nbSim + 1$
24: **else**
25: $pos \leftarrow pos \cup k$
26: $k \leftarrow k + 1$
27: $psimilarity \leftarrow (nbSim/length(p_i)) * 100$
28: **if** $psimilarity \geq minPsimilarity$ **then**
29: **return** pos
30: **else**
31: $pos \leftarrow \emptyset$
32: **return** pos
Function $sim\_item(IA_D, a, b)$
33: $commAtt, setCommAtt \leftarrow \emptyset$
34: **get** $att(a)$ and $att(b)$ **from** IA_D
35: **for** each position v in $att(a), att(b)$ **do**
36: **if** $att(a)[v] = att(b)[v]$ **then**
37: $commAtt \leftarrow commAtt + 1$
38: $setCommAtt \leftarrow setCommAtt \cup att(a)[v]$
39: $isimilarity \leftarrow (commAtt/length(att(a))) * 100$
40: **if** $isimilarity \geq minIsimilarity$ **then**
41: **return** $setCommAtt$
42: **else**
43: $setCommAtt \leftarrow \emptyset$
44: **return** $setCommAtt$

Algorithm 2 Generalisation($p_i, p_j, x, setCommAtt$)

1: $p'_i \leftarrow p_i$
2: $p'_j \leftarrow p_j$
3: $original\text{-}pattern_1(p'_i) \leftarrow p_i$
4: $original\text{-}pattern_2(p'_i) \leftarrow p_i$ with $p_i[x] \leftarrow p_j[x]$
5: $original\text{-}pattern_1(p'_j) \leftarrow p_j$ with $p_j[x] \leftarrow p_i[x]$
6: $original\text{-}pattern_2(p'_j) \leftarrow p_j$
7: **for** each *original-pattern* **do**
8: **Scan** F_p to find *original-pattern*
9: **if** *original-pattern* found in F_p **then**
10: **get** support(*original-pattern* from F_p)
11: **else if** *original-pattern* found in F_f **then**
12: **get** support(*original-pattern* from F_f)
13: **else**
14: **support**(*original-pattern*) $\leftarrow 0$
15: support(p'_i) $\leftarrow$ support($original\text{-}pattern_1(p'_i)$) + support($original\text{-}pattern_2(p'_i)$)
16: **if** $support(p'_i) \geq minsup$ **then**
17: $F_g \leftarrow F_g \cup \{p'_i\}$
18: $F_f \leftarrow F_f \cup \{p'_i\}$
19: support(p'_j) $\leftarrow$ support($original\text{-}pattern_1(p'_j)$) + support($original\text{-}pattern_2(p'_j)$)
20: **if** $support(p'_j) \geq minsup$ **then**
21: $F_g \leftarrow F_g \cup \{p'_j\}$
22: $F_f \leftarrow F_f \cup \{p'_j\}$
23: **return** F_g, F_f

one), G_SPM manages two kinds of data sources where these sources provide different kinds of data.

The sequential mining process mines a single data source and generates frequent patterns containing information from this source. On the other hand, G_SPM mines two data sources in a single mining process where these sources provide different kinds of data. Although the complexity of G_SPM is higher than that of the sequential mining algorithm, it adopts a strategy of selective mining that limits the increase in complexity where it mines the sequential sources and takes advantage of the complementary source only in the case where there is a problem of similarity in order to compensate the loss of frequent patterns.

Furthermore, the complexity of G_SPM would be lower than that of the naive approaches. Both approaches mine the sources in a single process ; however, thee strategy of selective mining works on limiting the complexity of G_SPM by mining the complementary source only when needed throughout the process.

Finally, the frequent patterns generated by the sequential mining algorithm only contain a single kind of information, item ids. However, the frequent general patterns generated by G_SPM contain two kinds of information which makes them richer in information and more diverse.

In this chapter, we presented a description of a multi-dataset with its different possible kinds of data sources ; then, we determined a structuring of the data sources based on the relations that we determined among them. We then described our scientific contributions and proposed algorithm, G_SPM, for mining frequent general patterns in order to solve the problem of similarity caused by the multi-* nature of the dataset.

3

Experiments and Results

Contents

In this chapter, we present the experimental evaluations of G_SPM which extracts frequent general sequential patterns. In the first section, we describe the dataset that we use in these experimental evaluations. In the second section, we discuss the evaluation measures that we determine in order to evaluate the algorithm. In the third section, we analyze and discuss the experimental evaluations. Finally in the fourth section, we conclude this chapter and discuss future experiments.

3.1 Description of the application dataset

As detailed in chapters 1 and 2, the dataset has a multi-* nature. This is a particular nature where there are multiple interrelated data sources providing various kinds of data, and these data could be represented in different data dimensions. One data source is sequential as it contains data sequences, other data sources provide descriptive data to the data sequences, while others provide descriptive data to the elements in the data sequences. Therefore, there exists two kinds of relations among different data sources : contextual relations and complementary relation.

In 2017, L'Huillier et al. [35] have obtained a music corpus from Deezer company [5]. This corpus contains anonymized data of about 3,000 users who have consulted music tracks between December 4^{th}, 2016 and January 5^{th}, 2017. For each consultation, the corpus contains users anonymized names, tracks names, artists names and timestamps. They have also obtained metadata about the tracks and the artists using the services of the EchoNest website which offers an API

5. https ://www.deezer.com/

allowing to recover metadata on tracks and artists [6]. These metadata are represented in the form of several attributes describing the artists and the tracks.

The consultations of tracks by users across timestamps can be represented as data sequences where each data sequence, representing a specific user, is comprised of the tracks consulted by the user in a sequential manner according to their timestamps. Therefore, we consider that the data about the users are descriptive data provided to the data sequences as a whole, while the additional data about the tracks and the artists of these tracks are considered as additional descriptive data to the elements of the sequences, that are the tracks.

3.1.1 The dataset extracted from this corpus

In this collected dataset, there is data sparsity among the descriptive attributes of the tracks and the artists. In our work, we choose to discard the consultation of any track having data sparsity among its descriptive attributes of the tracks and/or artists. In addition, in this dataset, there are tracks that are skipped by the users. A skipped track is a track that is consulted by the user ; however, it was skipped and thus not listened by this user. We choose to discard the skipped tracks and thus remove them from the data sequences of the user and strictly keep the tracks that are listened by the users.

As mentioned, the dataset contains users' consultations of music tracks with timestamps between December 4^{th}, 2016 and January 5^{th}, 2017. If we consider all the timestamps in our work, the length of the data sequences will be huge due to the huge number of consultations by users per day. Therefore, we choose to use data between a predefined timestamp interval in our experiments performed in order to validate our algorithm. The statistics of the sample of the dataset that we use in our experiments is displayed in figure 3.1.

# Sequences	2,220
# Mean of sequence lengths	48.91
# Consultations	108,596
# Artists	8,959
# Soundtracks	34,080
Time interval	5th-8th Dec. 2016

TABLE 3.1 – Statistics of our sample of Deezer corpus

Among the attributes describing the artists and the tracks in the collected metadata of the corpus, we chose to use six attributes in our experiments, and we make our choice of attributes based on the lowest percentage of data sparsity among the attributes. Five attributes out of these six describe the tracks while one attribute describes the artists of these tracks. The six chosen attributes are described as follows.

— Acousticness : It is the measure of the acoustic level of the track. It represents a value between 0 and 1 to indicate the acousticness level which explains if it uses one or more electronic tones (the higher the measure value, the less electronic tones the music contains). For example, the track *Tell me why* of the artist *Genesis* has an acousticness level of 0.323.

— Duration : It is the duration of the track, in seconds. For example, the track *Tell me why* of the artist *Genesis* has a duration of 299 seconds.

6. http ://the.echonest.com/

— Energy : It is the measure of the intensity and the activity level of the track. It represents a value between 0 and 1 ; the higher this value, the more this track is considered energetic (intense, fast and loud). For example, the track *Tell me why* of the artist *Genesis* has an energy level of 0.758.

— Danceability : It is the measure indicating how much the track is suitable for dancing. It represents a value between 0 and 1 ; the higher this value, the more is this track suitable for dancing. For example, the track *Tell me why* of the artist *Genesis* has a danceability level of 0.604.

— Speechiness : It is the measure indicating the importance of the lyrics in a track. It represents a value between 0 and 1. The higher this value, the higher is the importance of the track lyrics. For example, poems have a speechiness close to 1 while music without words have a speechiness close to 0. The track *Tell me why* of the artist *Genesis* has a speechiness level of 0.037.

— Rank-artist : It is the measure of the ranking of the soundtrack's artist on Deezer. It represents a value between 0 and 1,000,000. The higher this value, the higher is the ranking of the artist. For example, the artist *Genesis* has a ranking value of 595,478.

3.2 Evaluation measures

In this section, we present the evaluation measures that we determine in order to evaluate the efficiency of G_SPM.

As detailed in chapter 2, the parameters of G_SPM are $minsupProm$, the minimum pattern similarity and the minimum item similarity.

$minsupProm$ is the minimum predefined support threshold used in the sequential pattern mining process. When the value of $minsupProm$ varies, it first impacts the number of promising patterns generated from the sequential pattern mining process where the number of promising patterns impact the number of similar patterns as well as the number of those who have similar items among them. Consequently, this impacts the number of frequent general patterns generated from G_SPM and the runtime of the algorithm. Therefore, we are interested in evaluating the runtime and the number of frequent general patterns generated by G_SPM over different values of $minsupProm$.

The minimum pattern similarity allows determining the similar patterns among all the promising ones. For example, if the minimum pattern similarity is set to 80%, then the set of similar patterns include all promising patterns having a pattern similarity higher than or equal to 80%, while other promising patterns are discarded. Therefore, the minimum pattern similarity impacts the number of frequent general patterns and the runtime of G_SPM.

The minimum item similarity allows determining the similar patterns having similar items at the same positions in the patterns. General patterns are thus formed from these ones, and a general pattern is then considered as frequent if its support is greater than or equal to $minsup$. Therefore, the minimum item similarity impacts the number of frequent general patterns and the runtime of the algorithm.

Hence, we evaluate G_SPM in terms of the number of frequent general patterns and the runtime. First, we evaluate the impact of $minsupProm$ on G_SPM ; then, we evaluate the impact of the minimum pattern similarity and the minimum item similarity on G_SPM.

The experiments were conducted on a 64-bit Intel Core laptop with 2.60GHz CPU, 7.7 GB RAM and Ubuntu 16.04 LTS operating system.

3.3 Experimental evaluations of the naive approach

In section 2.3.2 of chapter 2, we have detailed two naive approaches proposed in the literature which integrate the main data source with the complementary sources and perform a single mining process of the integrated data sources. The first approach is to substitute the items in the data sequences with their corresponding descriptive attributes and perform the mining process on the resulting sequences which uniquely contain descriptive attributes. The second approach is to add the descriptive attributes to the items of the data sequences. This latter approach allows generating patterns of three kinds : patterns containing item ids, patterns containing both item ids and descriptive attributes as well as patterns containing sets of descriptive attributes. The resulting patterns of G_SPM represent patterns containing item ids and patterns containing either item id or a set of descriptive attribute at each position in the patterns. The resulting patterns generated by G_SPM are subset of the resulting patterns generated by the second naive approach ; therefore, we consider this naive approach as a baseline for evaluating G_SPM.

In this approach, the dataset is made up of sequences of itemsets (as in the SPM literature), where each itemset contains the item id along with its set of descriptive attributes. This leads to a noticeable increase in the length of sequences, compared to original sequences provided by the main data source which leads to an increase in the complexity of the mining process, especially as the number of descriptive attributes is not small.

The traditional sequential pattern mining process in G_SPM as well as in the naive approach that is used as a baseline to evaluate G_SPM, are performed by the well-known PrefixSpan algorithm [41], with its implemented version in SPMF library [15]. PrefixSpan is proved in the literature to be an efficient algorithm where it is widely used by various works in the literature. PrefixSpan algorithm is presented in algorithm 3.

Algorithm 3 (PrefixSpan) Prefix-projected sequential pattern mining algorithm

Input : A sequence database S_D and a minimum predefined support threshold
Output : The complete set of sequential patterns
Method : Call PrefixSpan(<>,0,S_D).
Subroutine PrefixSpan(α,l,$S_D|_\alpha$)
Parameters : α : a sequential pattern ; l : the length of α ; $S_D|_\alpha$: the α-projected database, if $\alpha \neq <>$; otherwise, the sequence database S_D
Method :

1. Scan $S_D|_\alpha$ once, find the set of frequent items b such that
   (a) b can be assembled to the last element of a to form a sequential pattern ; or
   (b) $< b >$ can be appended to α to form a sequential pattern.
2. For each frequent item b, append it to α to form a sequential pattern α', and output α' ;
3. For each α', construct α'-projected database $S_D|_{\alpha'}$, and call PrefixSpan(α',$l+1$,$S_D|_{\alpha'}$).

According to the related work in the domain of sequential pattern mining, the minimum support threshold used in the sequential pattern mining process is determined according to the characteristics of the used dataset : the number of sequences, the number of transactions and the average sequence length. Based on the specifications of our dataset that are described in Table 3.1, we set the value of the minimum support threshold *minsup*, as 40 which represents 1.8 % of the total number of sequences in the database.

In order to experimentally evaluate the naive approach, we run PrefixSpan algorithm with

$minsup = 40$ on our dataset while integrating the data source containing the sequences of item ids with the one containing descriptive attributes for each item. The resulting number of frequent patterns is too huge to be stored (more than 15GB) where the runtime of the mining process takes 3 days and the number of generated frequent patterns reaches 15 million frequent patterns. By running PrefixSpan algorithm on a sample of this dataset made up of the item ids and half the number of attributes, the number of frequent patterns reaches 55 million patterns, with a runtime of 1.5 hours.

These results show the inefficiency and the high complexity of the naive approach and thus its inadequacy on such a dataset. The reason behind this huge complexity is that the algorithm generates all the possibilities of frequent patterns : patterns containing only item ids, patterns containing only attributes as well as patterns containing item ids and attributes. This leads to a huge number of patterns. We consider that the frequent patterns containing only descriptive attributes are not precise as they do not contain any specific information where the attributes could be shared among different items ; therefore, these patterns increase the complexity of the algorithm while decreasing the quality of its results. In addition, we consider that a pattern containing the most specific information (i.e. containing item ids) is the pattern having the best quality ; therefore, when this kind of pattern is frequent, generating the more general forms of patterns would impact the quality of results with no added value to the algorithm results. This naive approach allows extracting patterns of all levels of generality even when the most specific pattern is frequent which leads to a huge algorithm complexity.

3.4 Experimental evaluation of G_SPM algorithm

In this section, we discuss the experimental evaluations of G_SPM algorithm. We first analyze the impact of *minsupProm*, then the impact of the minimum pattern similarity and finally the impact of the minimum item similarity on the results of the algorithm.

3.4.1 Impact of minsupProm

In our work, we have two minimum support thresholds : *minsup* which allows detecting frequent patterns and *minsupProm* which allows detecting promising patterns.

G_SPM starts by mining frequent and promising patterns from the main data source with a traditional SPM algorithm using *minsupProm* and *minsup* as the minimum support thresholds where *minsupProm* allows detecting promising patterns and *minsup* allows detecting frequent patterns. After the SPM process comes the process of forming general patterns. We thus evaluate the impact of *minsupProm* on the algorithm results for these two processes.

SPM process

We start our evaluation by analyzing the sequential pattern mining process of G_SPM. The choice of the value of *minsupProm* impacts the number of generated promising patterns as well as the runtime of the SPM process. Recall that the promising patterns are those to be considered for generalization by G_SPM. For this reason, we propose to evaluate the impact of the value of *minsupProm* experimentally.

According to definition 2.4.2, the value of *minsupProm* is lower than *minsup* but close to it as the promising patterns are patterns that are not far from being frequent. In this frame, we evaluate the algorithm with different values of *minsupProm* in order to evaluate the impact of this parameter on the runtime and the number of frequent general patterns. *minsupProm* is

measured relative to *minsup* ; therefore, we choose to represent *minsupProm* as a percentage of *minsup*. In the conducted experiments, *minsupProm* ranges from 90% to 75% of *minsup*.

As previously mentioned, *minsup* is set to 40. By running PrefixSpan algorithm on the sequential data source with *minsup*, the corresponding number of generated frequent patterns is 1,280 and the runtime of the algorithm is 0.368 seconds. This number of frequent patterns remains unvaried with the variation of *minsupProm* as it depends on *minsup*. In a further step, we run PrefixSpan algorithm with decreasing values of *minsupProm* relative to *minsup*. Table 3.2 displays the number of promising patterns whose support values range between [*minsupProm*,*minsup*[as well as the runtime of the sequential pattern mining process of G_SPM according to *minsupProm*.

minsupProm (relative to *minsup*)	36 (90%)	34 (85%)	32 (80%)	30 (75%)
# promising patterns (% relative to freq. patt.)	1,135 (90%)	2,129 (166%)	3,745 (293%)	6,399 (500%)
additional run time (s) (% add. run time)	0.115 (+31%)	0.231 +(63%)	0.266 (+72%)	0.322 (+87%)

TABLE 3.2 – Number of promising patterns and runtime with decreasing *minsupProm*

From the results displayed in Table 3.2, we can notice that *minsupProm* greatly impacts the number of promising patterns generated by the SPM process of G_SPM. This is in line with the literature of SPM which states that the number of generated frequent patterns increases exponentially as the minimum support threshold decreases.

When *minsupProm* is at its highest value (90% of *minsup*), the number of generated promising patterns from the SPM process is close to the number of generated frequent patterns. Then, with each decrease of 5% of *minsupProm*, the number of promising patterns increases by a percentage ranging from 70% to 85%. For example, when *minsupProm* is 80% of *minsup*, the number of promising patterns becomes approximately 3 times larger than the number of frequent patterns.

Concerning the runtime of the algorithm, it also increases with the decrease of *minsupProm* ; however, it increases with lower percentages. When *minsupProm* is at its highest value (90% of *minsup*), the additional number of promising patterns relative to the number of frequent patterns (90% of frequent patterns) correspond to an increase in the runtime by only 31%. With *minsupProm* equal to 80% of *minsup*, the runtime increases by 72%, which is considered a relatively low percentage of increase compared to the percentage of increase of the number of promising patterns (300%).

Until this stage of the experiments, the optimal value of *minsupProm* cannot be determined ; thus, these decreasing values of *minsupProm* will still be of interest for the following evaluations until we are able to determine the optimal value which allows generating the most adequate number of patterns with a limited runtime.

Forming general patterns

In the following experiment, we are interested in evaluating the number of frequent general patterns generated by G_SPM algorithm over the decreasing values of *minsupProm* as well as the runtime of G_SPM.

We now focus on the generalization phase where we are interested in evaluating both the number of frequent general patterns according to *minsupProm*, and the runtime of the algorithm.

Recall that two promising patterns are considered as similar if they have the same length and whose pattern similarity is greater than or equal to the minimum pattern similarity. Two items are considered as similar items if the percentage of their common attributes is greater than or equal to the value of minimum item similarity. In this experiment, we set the value of minimum pattern similarity to 80% and the value of the minimum item similarity to 80%. In other words, two promising patterns are considered as similar patterns if they have a pattern similarity that is higher than or equal to 80%, and we consider two items as similar if they have a minimum of 80% of common attributes.

Table 3.3 displays the number of frequent general patterns generated by G_SPM algorithm as well as the runtime of the algorithm with decreasing values of *minsupProm*.

minsupProm (relative to *minsup*)	36 (90%)	34 (85%)	32 (80%)	30 (75%)
# promising patterns	1,135	2,129	3,745	6,399
# similar patterns (% promising patterns)	1,073 (94.5%)	2,026 (95.2%)	3,596 (96%)	6,183 (96.6%)
# similar patterns having similar items (% promising patterns)	508 (44.8%)	1189 (55.8%)	2,432 (64.9%)	4,433 (61.4%)
# general patterns (% prom. patt.)	340 (30%)	873 (41%)	1,968 (52%)	3,930 (61%)
# frequent patterns	1,620	2,153	3,248	5,210
Run time (s) of G_SPM	124	281	769	1,508

TABLE 3.3 – Number of general patterns and runtime of G_SPM algorithm with decreasing *minsupProm*

As expected, the number of frequent general patterns generated by G_SPM increases with the decrease of *minsupProm*. More precisely, when *minsupProm* decreases by 5%, the number of frequent general patterns is more than doubled. For example, when *minsupProm* is 75% of *minsup*, the number of frequent general patterns becomes 3 times greater than the number of initial frequent patterns that are generated from the mining process of the main data source (3,930).

By analyzing the percentage of the frequent general patterns that are generated out of the promising patterns, we notice that as the value of *minsupProm* increases, this percentage increases by 30% to 61%.

In order to study thoroughly this increase, we first focus on the number of similar patterns, that are the promising patterns having item differences at the same positions in the patterns (see row 3, Table 3.3). The minimum pattern similarity is set to 80%. The number of similar patterns increases with decreasing values of *minsupProm* where it is multiplied by 6 between $minsupProm = 90\%$ and $minsupProm = 75\%$. By analyzing further the percentage of similar patterns with respect to the number of promising ones, we notice that this percentage is high (94.5% for $minsupProm = 90\%$). This high percentage implies that there is a high possibility of having similar patterns among the promising ones. The reason behind this high possibility is that considering the patterns similar only requires having different items at the same positions

in the patterns without comparing the items. Then, by decreasing values of *minsupProm*, the percentage of similar patterns out of all promising patterns increases until it reaches 96.6%. The reason behind this increase in percentage is that as the number of promising patterns increase ; thus, the possibility of having similar patterns among the promising ones increase.

After this step, we evaluate the number of similar patterns having similar items (see row 3, Table 3.3). This number increases with decreasing values of *minsupProm* where it is multiplied by 9 between $minsupProm = 90\%$ and $minsupProm = 75\%$. By analyzing further the percentage of similar patterns having similar items with respect to the number of promising patterns, we notice that this percentage is relatively low for $minsupProm = 90\%$ (44.8%), but it increases with decreasing values of *minsupProm* until it reaches 61.4% with $minsupProm = 75\%$.

As *minsupProm* is greater than 50% of *minsup*, all the similar patterns having similar items will be generalized. The reason is that the support of each of the two similar patterns having similar items is $> 50\%$ of *minsup* ; thus, the general pattern formed from these two similar patterns have a support $>$ *minsup* (the sum of the supports of the two similar patterns).

By analyzing the number of frequent general patterns generated by G_SPM, we notice that it increases with decreasing values of *minsupProm*. Furthermore, we notice that when $minsupProm \geq 85\%$ of *minsup*, the number of frequent general patterns is lower than the number of initial frequent patterns (1,280), while when $minsupProm < 85\%$, the number of frequent general patterns is higher than the number of initial frequent patterns. The closest number of frequent general patterns to the number of initial frequent patterns is 873 when $minsupProm = 85\%$ of *minsup* representing 41% of the promising patterns and resulting in a total number of frequent patterns = 1,840.

In the following step, we evaluate the runtime of G_SPM with decreasing values of *minsupProm*. As expected, the runtime of G_SPM is significantly higher than that of the mining process of the main source (see Table 3.2). This difference in the runtime is justified by the fact that the mining process of the main source is a process that mines a single data source and generates frequent patterns containing a single kind of information, while G_SPM algorithm mines two data sources providing different kinds of information selectively where it undergoes several steps of comparing similar patterns and items and generating frequent general patterns, which are patterns containing hybrid information.

For $minsupProm = 90\%$ of *minsup*, the runtime of G_SPM algorithm takes 124 seconds (around 2 minutes). Then with each decrease of *minsupProm* by 5%, the runtime increases almost by a factor of 2 until it reaches 1,508 seconds (around 25 minutes). An advantage is that the increase in the runtime of G_SPM is linear with the increase in the number of promising patterns. Although the generalization step is complex, the strategies established in order to limit the number of general patterns also contribute to a limited increase in the runtime of the algorithm.

If we compare the runtime of the lowest value of *minsupProm* (75% of *minsup*) with the runtime of the naive approach, we notice that G_SPM's runtime is significantly lower which confirms attaining one of goals of G_SPM concerning reducing the algorithm complexity.

From the previous analyses, we confirm that the promising patterns should be those who have support values relatively close to *minsup* ; otherwise, the complexity of G_SPM algorithm will become higher with the increasing number of promising patterns.

The frequent general patterns may be more frequent than the traditional patterns, i.e. having higher support values ; however, the quality of these patterns is lower as they are more general than the original patterns mined from the main data source. An item id represents a unique item, whereas descriptive attributes could be shared among different items. Consequently, the larger the number of frequent general patterns in the set of mined patterns, the less is the specific

information conveyed by the set of patterns mined by G_SPM and thus the lower is the overall quality of the results.

Hence, we consider that the most adequate value of *minsupProm* for this dataset is equal to 85% of *minsup* as it results in a significant but limited number of frequent general patterns relative to the frequent patterns generated by SPM process (about 70% of additional patterns), and the runtime of the algorithm with this value of *minsupProm* remains limited to 281 seconds (about 4 minutes).

3.4.2 Impact of the minimum pattern similarity

The minimum pattern similarity determines the minimum percentage similarity between two promising patterns to be considered as similar patterns. Therefore, it mainly impacts the number of similar patterns among the promising ones which thus impacts the number of frequent general patterns.

In this section, we aim at studying the impact of the minimum pattern similarity on the number of frequent general patterns as well as on the runtime of G_SPM. In this experiment, we evaluate the algorithm for different values of minimum pattern similarity where the minimum item similarity is fixed at 80% and *minsupProm* is fixed at 85% of *minsup*.

The number of frequent general patterns and the runtime of G_SPM with different values of minimum pattern similarity are displayed in Table 3.4.

Minimum pattern similarity	80%	60%
# promising patterns	2,129	2,129
# similar patterns (% out of promising patterns)	2026 (95.2%)	2034 (95.5%)
# similar patterns having similar items (% out of promising patterns)	1189 (56%)	1885 (89%)
# frequent general patterns (% out of promising patterns)	873 (41%)	1,498 (70%)
# total frequent patterns	2,153	2,778
Runtime (s) of G_SPM	281	2,965

TABLE 3.4 – Number of frequent general patterns and runtime of G_SPM over different values of minimum pattern similarity

The number of promising patterns remains unvaried (2,129) as it is not impacted by the minimum pattern similarity. We notice from Table 3.4 that as the value of minimum pattern similarity decreases, the number of similar patterns increases. According to the definition of the minimum pattern similarity, when its value decreases, there are more possibilities of similarities among the promising patters. By analyzing the percentage of the similar patterns with respect to the promising ones, we notice that the percentage is high even for minimum pattern similarity = 80% (95.2%) which implies that the possibility of having similar patterns among all the promising patterns is already high for minimum pattern similarity = 80%. This percentage increases to 95.5% when decreasing the minimum pattern similarity to 60% as the possibility of having

similar patterns among the promising patterns increases ; however, this increase in percentage is not significant.

In a further step, we analyze the number of similar patterns having similar items, we notice that this number is multiplied by 1.6 with the decrease of the minimum pattern similarity from 80% to 60%.

By analyzing further the similar patterns having similar items with respect to the promising patterns, we notice that this percentage increases significantly from 56% to 89%.

The reason behind this increase is that the positions in the similar patterns which could have similar items increase which increases the possibility of having similar patterns with similar items.

By analyzing the number of frequent general patterns, we notice that this number is multiplied by 1.7 when the minimum pattern similarity decreases from 80% to 60%. The number of frequent general patterns generated by G_SPM when the minimum pattern similarity = 60% (1,498) is close to the number of initial frequent patterns (1,280).

By analyzing further the percentage of the frequent general patterns among the promising ones, we notice that this percentage increases noticeably from 41% to 70% by decreasing the value of minimum pattern similarity from 80% to 60%. The total number of frequent patterns for minimum pattern similarity =60% (2,778) is more than doubled with respect to the number of initial frequent patterns.

We now evaluate the runtime of the algorithm with different values of minimum pattern similarity. The runtime increases significantly by a factor of 10 with the decrease of the minimum pattern similarity from 80% to 60%. This shows the great impact of the minimum pattern similarity on the complexity of the algorithm, and this is due to several reasons. First, when the minimum pattern similarity decreases from 80% to 60%, this allows detecting more item differences in promising patterns which leads to a longer runtime of the algorithm. Note that the value of minimum similarity determines the number of allowed differences between two similar patterns based on their length. Second, for each two promising patterns, this allows comparing more pairs of items in order to detect similar items. Third, it could lead to forming more general patterns from each two promising patterns depending on the length of the patterns and thus the number of item differences. For example, if the number of item differences =1, one general pattern is formed ; however, if the number of differences is more than one, two general patterns are formed. In addition, the complexity of the process of support calculation of the formed general patterns when the minimum pattern similarity = 60% is higher than that when the minimum pattern similarity = 80%. The reason is that each formed general pattern has two original patterns where the process of searching for these patterns among the promising and the frequent ones in order to find their support values could be costly especially that these patterns may or may not exist in the database.

Hence, the advantage of increasing the value of minimum pattern similarity is obtaining more frequent patterns in order to handle the problem of item similarity among the patterns which is our main objective ; however, it is at the cost of increasing the complexity of the algorithm. Recall that our main objective is generating frequent general patterns in order to handle the problem of item similarity in sequential data with a limited complexity. When the minimum pattern similarity =60% , the number of frequent general patterns generated by G_SPM (1,498) is close to the number of initial frequent patterns (1,280), and it represents 70% of the promising patterns which allows handling the problem of item similarity among patterns by generating frequent general patterns with a high percentage. However, it is at the cost of the runtime of G_SPM as it is significantly high (around 49 minutes).

Therefore, we consider that the most adequate value for the minimum pattern similarity is

80% as it allows generating frequent general patterns representing 41% of the promising patterns with a limited runtime (around 2 minutes) which allows attaining our two objectives. In addition, we consider that as the value of the minimum pattern similarity is higher, the resulting patterns are more precise.

3.4.3 Impact of the minimum item similarity

The minimum item similarity determines the minimum required percentage of common attributes between two items to consider them similar. Therefore, it impacts the number of similar patterns having similar items and thus the results of G_SPM. In this section, we study the impact of the minimum item similarity on the number of frequent general patterns and the runtime of G_SPM. In this frame, we evaluate the algorithm for different values of minimum item similarity which vary between 50% and 80%. For this experiment, the minimum pattern similarity is fixed at 80% and *minsupProm* is fixed at 85% of *minsup*. The number of frequent general patterns and the runtime of G_SPM with different values of item similarity are displayed in Table 3.5.

Minimum item similarity	50%	60%	80%
# similar patterns having similar items	2,000	1,789	1,189
(% out of promising patterns)	(94%)	(84%)	(56%)
#frequent general patterns	5,984	2,328	873
(% out of promising patterns)	281%	109%	41%
#total frequent patterns	6,951	3,295	1,840
Runtime (s) of G_SPM	292	278	269

TABLE 3.5 – Number of frequent general over different values of minimum item similarity

The number of promising patterns is not impacted by the minimum item similarity, and it remains unvaried (2,129 patterns) as well as the number of similar patterns among the promising ones (2,026 patterns). In addition, as previously mentioned, the number of initial frequent patterns remains unvaried (1,280 patterns). From Table 3.5, we notice that as the minimum item similarity increases from 50% to 80%, the number of similar patterns having similar items decreases by a factor of 1.7. By analyzing the percentage of the similar patterns having similar items out of the promising patterns, we notice that it decreases noticeably from 94% to 56%. The reason for this decrease is that the required percentage of common attributes in order to consider two items as similar increases which leads to a decrease in the possibility of having similar items among the similar patterns.

By analyzing the number of frequent general patterns generated by G_SPM algorithm, we notice that the number of frequent general patterns, decreases approximately by a factor of 2.6 with each increase in the minimum item similarity from 50% until 80%. By analyzing further the percentage of frequent general patterns with respect to the promising patterns, we notice that it is significantly high with minimum item similarity = 50% (281%). This high percentage is due to the large number of similar patterns having having similar items among the promising ones ; this leads to a large number of candidate general patterns. This percentage decreases by more

than half of its value with each increase in the minimum item similarity until it reaches 41% with minimum item similarity = 80%.

The number of frequent general patterns is higher than the number of initial frequent patterns when minimum item similarity < 80%.

By analyzing the total number of frequent patterns generated by G_SPM algorithm, we notice that as the minimum item similarity increases, the number of frequent general patterns decreases as expected due to the decrease in the number of frequent general patterns. This number is 6,951 for minimum item similarity = 50% which is 5 times larger than the number of initial frequent patterns, and it decreases almost by a factor of 2 for each increase in the value of minimum item similarity to reach 1,840 for minimum item similarity = 80%.

In the following step, we evaluate the runtime of G_SPM algorithm for different values of the minimum item similarity. We notice from Table 3.5 that the runtime is relatively small, and it decreases from 292 seconds (4.9 minutes) for minimum item similarity = 50% to 269 seconds (4.5 minutes) for minimum item similarity = 80% where the difference between them is only 23 seconds which is not a significant difference.

Hence, as the minimum item similarity decreases, G_SPM allows generating more frequent general patterns with a limited increase in the runtime ; however, the number of frequent general patterns could become huge. Larger numbers of frequent general patterns may handle the problem of item similarity at a wider scope ; however, it is at the cost of the quality of the resulting patterns from G_SPM as the general patterns contain less specific information than initial patterns which make them less precise. For example, the number of frequent general patterns with minimum item similarity = 50% (5,984) is 5 times larger than the number of initial frequent patterns (1,280) which is a huge number. Even for minimum item similarity = 60%, the number of frequent general patterns (2,328) is almost twice larger than the number of initial frequent patterns which we consider relatively large. While for minimum item similarity = 80%, the number of frequent general patterns (873) is the less than the number of initial frequent patterns (1,280), and it is the closest number to it. Therefore, we consider that the minimum item similarity set to 80% is the most adequate value of minimum item similarity as it could handle the problem of minimum item similarity among patterns with a limited impact on the quality of the resulting patterns as well as with a limited runtime.

Concerning the similarity among items in G_SPM, we have made a choice to set the minimum item similarity as a minimum percentage of common attributes between two items with specific intervals for each attribute in order to broaden the scope of similarity between items. However, this choice will be questioned further in a future work where we aim at adopting more advanced similarity measures concerning the items of the patterns.

3.5 Conclusions

From these experimental evaluations on this dataset, we can draw several conclusions.
minsupProm has a great impact on the number of promising patterns. The set of promising patterns is one of the inputs of G_SPM ; thus, *minsupProm* has a significantly high impact on the runtime as well as number of frequent general patterns generated by G_SPM.

Concerning the runtime of the algorithm, we notice that the difference is significantly greater with increasing values of the minimum pattern similarity where the difference reaches 25 minutes, while the difference is only few seconds with increasing values of the minimum item similarity. Thus, we conclude that the impact of the minimum pattern similarity on the runtime of G_SPM algorithm is larger than that of the minimum item similarity. This allows us to understand that

the process of comparing the promising patterns in order to find similar ones takes a longer runtime compared to the process of comparing the attributes of the items in similar patterns in order to find similar items.

The number of frequent general patterns generated by G_SPM algorithm is impacted by the minimum item similarity more significantly than the minimum pattern similarity. Therefore, we conclude that the number of frequent general patterns is impacted by the similar items more than the similar patterns. In other words, even if there is a high percentage of similar patterns among the promising ones, the more important is having a high percentage of similar items among them in order to attain the objective of generating a significant number of frequent general patterns.

We can notice that the total number of frequent patterns generated by G_SPM is significantly smaller than the number of frequent patterns mined by the naive approach. This low number is explained by the fact that the patterns mined by G_SPM are not redundant in the sense that general patterns containing descriptive attributes of items are only mined in the case of item similarity among patterns where the most specific pattern is not detected as frequent. Besides, patterns containing items and attributes at the same position in the patterns are not supported by G_SPM as we consider that they impact the quality of G_SPM results.

Through the conducted experiments, we confirm that G_SPM is efficient concerning the runtime and the number of frequent patterns. With $minsupProm = 85\%$ of $minsup$, minimum pattern similarity = 80% and minimum item similarity = 80% for our specific dataset, G_SPM is capable of achieving the goals of handling the item similarity by generating a sufficient number of frequent general patterns taking into consideration the algorithm complexity and the quality of the resulting patterns.

4

Conclusion and Future Work

Contents

4.1 Conclusion

In this PhD thesis, we have proposed a general sequential pattern mining approach to mine multi-source data with the aim of understanding users' behavioral data. Data is now hugely produced and collected, and it can be used for analytics. The collected data can come from a single data source, or they can come from multiple data sources related to each other where each data source provides one or more kinds of information. The data provided by multiple data sources could be sequential data representing the behavior of users or various descriptive data. This allows forming a heterogeneous but rich dataset that leads to obtaining rich patterns containing various kinds of information.

Some data characteristics could lead to the loss of possibly interesting patterns and thus could limit the number of frequent patterns mined. This is particularly the case of the similarity that could exist between certain data items. This similarity results in the occurrence of several patterns representing the same behavior of users. The behavior could be frequent ; however, as its frequency is distributed among several patterns, this could result in the infrequency of these patterns and thus their loss. We thus aim at taking advantage of the data coming from multiple sources in order to handle the problem of item similarity existing among the sequential data and to compensate the loss of patterns.

Various works in the literature were proposed to mine multi-source data, and they can be mainly divided into two approaches. The first approach mines data sources in a separate manner. This approach leads to generating frequent patterns containing information from a single data source without taking advantage of the relations existing between different data sources. In addition, this approach does not support handling the problem of item similarity in sequential data sources which leads to the loss of frequent patterns and thus the generation of a limited number of frequent patterns. The second approach integrates data provided by all data sources together and mines the data in a single process. This approach allows taking advantage of the multi-source data and thus handles the problem of item similarity ; however, it has a high complexity,

it generates a huge number of frequent patterns, and it has redundancy among its patterns.

We propose an approach that avoids the limitations of the previous approaches by mining multi-* data in a single process with limited complexity. First, we propose a structuring of the data sources in a multi-* dataset consisting of a sequential data source, a descriptive data source of users and a descriptive data source of items. We determine two different kinds of relations between the data sources : contextual and complementary relations. The contextual relation exists between a sequential data source and a descriptive data source of users where the latter provides contextual data to the data sequences in the former resulting in more specific sequential patterns. The complementary relation exists between the sequential data source and the descriptive data source of items where the latter provides complementary data to items in the data sequences that are represented as item ids which allows to obtain general sequential patterns, i.e. general patterns containing more general information at one or more of its positions.

Then, we define two similarity measures : pattern similarity and item similarity. The pattern similarity measures the similarity between patterns by comparing items at the same positions in the patterns in order to detect similar patterns, while the item similarity measures the similarity between different items by comparing their descriptive attributes in order to detect similar items.

We propose G_SPM, a novel general sequential pattern mining algorithm that mines general sequential patterns from multi-* data. G_SPM maintains the relations existing between the data sources and allows to handle the problem of item similarity in the sequential data. It starts with a sequential pattern mining process in a single data source (the sequential one) that detects frequent as well as promising patterns. Then, it detects similar patterns among the promising ones using the pattern similarity measure. Among the similar patterns, G_SPM detects similar items using the item similarity measure. G_SPM generates frequent general patterns where the similar items in similar patterns are substituted with more general information represented in the form of common descriptive attributes of the items. The generated frequent general patterns allow handling the problem of item similarity and compensating the loss of frequent patterns. These patterns are rich as they contain various kinds of information coming from multiple data sources.

The experimental evaluations confirm that G_SPM succeeds in achieving the general objective of understanding behavioral data by mining multi-* data and generating rich and various information that compensates the loss of patterns caused by the item similarity problem.

The complexity of G_SPM is limited compared to traditional approaches and the redundancy among the resulting patterns is avoided by adopting a strategy of selective mining where it starts by mining the sequential source and then mines the complementary source only when an item similarity is detected. To the best of our knowledge, this is the first work that proposes to mine data sources selectively to limit the algorithm complexity and manage item similarity.

4.2 Future work

We classify our planned future work into two main categories : future experiments and future research work.

4.2.1 Future experiments

In our conducted experiments, we evaluated the impact of *minsupProm*, minimum pattern similarity and minimum item similarity on G_SPM. In our future experiments, we plan to go further in several aspects.

In what concerns the number of generalizations in G_SPM, we plan to determine increasing numbers of allowed generalizations for increasing pattern lengths which would allow detecting a larger number of frequent general patterns. The pattern similarity in our conducted experiments can only be measured for patterns having the same length. However, in future work, we will consider more advanced similarity measures which allow measuring the pattern similarity among patterns of various lengths in order to detect a larger number of similar patterns. In our conducted experiments, we measure the item similarity by calculating the percentage of common attributes between two items. We have determined intervals for data dimensions of numerical attributes in order to widen the range of similarity for these attributes. However, there are certain data dimensions represented as textual attributes. Our current similarity measure detects similarity between two textual attributes only if they are identical. In future experiments, we aim at considering more advanced similarity measures for data dimensions represented as textual attributes to consider their similarity in semantics. In addition to these future experiments, we aim at experimentally evaluating G_SPM using other multi-* datasets in the educational domain and other various domains in order to validate not only its performance but also its ability of being generic.

4.2.2 Future research work

In addition to the aforementioned contributions, several future perspectives will be investigated in order to enhance and extend our work. In what follows, we discuss these future perspectives.

Managing both contextual and complementary relations

In this PhD thesis, we determined two kinds of relations, contextual and complementary relations. We proposed an approach that manages the complementary relation which allows obtaining frequent patterns with general information. In a future work, we aim at proposing an approach that manages both contextual and complementary relations in order to obtain frequent patterns that are general and specific at the same time. We consider that this can be done by integrating a contextual mining process in G_SPM algorithm in order to obtain *frequent contextual general sequential patterns*. Seq-Dim algorithm was proposed by [44] to mine multi-dimensional sequential patterns, and it is detailed in chapter 1. It first mines sequential patterns and then attaches frequent multi-dimensional information to each corresponding sequential pattern. We consider that the strategy of attaching frequent information to the sequential patterns in this algorithm is interesting, and we thus aim at integrating this strategy in G_SPM algorithm which allows to manage the contextual relation by attaching to each frequent general pattern its corresponding frequent contextual information.

Uncertainty in the relations between data sources

In certain cases, there could be data ethics and privacy restrictions. These restrictions force the protection of the data which could be represented in the form of anonymized or pseudo-anonymized data which makes the identifiers of the objects partially or completely anonymous. Anonymous object identifiers in a multi-source dataset lead to uncertainty in the relations between these data sources as we can not identify the same object in different sources. Due to this problem. Such kind of data can not be used in G_SPM algorithm that works based on the complementary relation. Therefore, in a future work, we aim to propose an approach that handles the problem of uncertainty in the relations existing between data sources.

Recommendations of user behavior

As previously mentioned, one of the main purposes of analyzing and understanding user behavior is providing users with personalized recommendations according to their information and behavior. The main reason behind choosing the frequent pattern mining domain is that the mining process as well as the frequent patterns generated from it are interpretable which allows generating association rules that help in providing interpretable recommendations. The antecedent of an association rule is used for matching the past behavior of users, while the rule consequent is used for providing the recommendations to these users.

G_SPM algorithm helps in increasing the data coverage among the resulting sequential patterns. This allows an increase in the number of resulting association rules which increases the recommendation possibilities. In addition, the generalization strategy adopted in G_SPM gives the priority for obtaining general information in the antecedents of the rules which helps in providing reliable recommendations. Hence, in the future, we will take advantage of the frequent patterns generated by G_SPM to form association rules out of them and then use these rules for providing personalized behavioral recommendations in the form of sequential patterns to the users in order to help improve the users' digital experience.

Multi-level frequent general patterns

G_SPM algorithm allows generating frequent general patterns. However, these patterns are only of one-level generalization. The minimum pattern similarity and the maximum number of allowed generalizations in G_SPM allow obtaining patterns of one or more generalizations. However, these patterns are generated as one set composed of all the frequent general patterns together. This set of frequent general patterns does not allow to obtain a hierarchical structure of general patterns. For example, it does not allow us to identify a lowest generalization level which represents the patterns with one generalization, a higher level of the same patterns with two generalizations and so on to reach the highest level of generalizations with the most general patterns. Therefore, in a future work, we aim at investigating further possible approaches to form a hierarchical generalization structure which gives a complete view of the pattern generalization and allows obtaining a multi-level hierarchical structure of general patterns which can be used to make a simple analysis of the set of patterns.

Bibliographie

[1] Charu C Aggarwal. *Data mining : the textbook*. Springer, 2015.

[2] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE, 1995.

[4] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[5] Sujeevan Aseervatham, Aomar Osmani, and Emmanuel Viennet. bitspade : A lattice-based sequential pattern mining algorithm using bitmap representation. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 792–797. IEEE, 2006.

[6] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435. ACM, 2002.

[7] Kevin Beyer and Raghu Ramakrishnan. Bottom-up computation of sparse and iceberg cube. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 359–370, 1999.

[8] MW Bright, Ali R Hurson, and Simin H. Pakzad. A taxonomy and current issues in multidatabase systems. *Computer*, 25(3) :50–60, 1992.

[9] Yen-Liang Chen and TC-K Huang. Discovering fuzzy time-interval sequential patterns in sequence databases. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5) :959–972, 2005.

[10] Viviane M Crestana and Nandit R Soparkar. *Mining decentralized data repositories*. University of Michigan, Computer Science and Engineering Division, Department of Electrical Engineering and Computer Science, 1999.

[11] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7) :1895–1923, 1998.

[12] Elias Egho, Chedy Raïssi, Dino Ienco, Nicolas Jay, Amedeo Napoli, Pascal Poncelet, Catherine Quantin, and Maguelonne Teisseire. Healthcare trajectory mining by combining multidimensional component and itemsets. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 109–123. Springer, 2012.

[13] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, et al. Knowledge discovery and data mining : Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996.

[14] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas. Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer, 2014.

[15] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. The spmf open-source data mining library version 2. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 36–40. Springer, 2016.

[16] Philippe Fournier-Viger, Roger Nkambou, and Engelbert Mephu Nguifo. A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Mexican International Conference on Artificial Intelligence*, pages 765–778. Springer, 2008.

[17] Bart Goethals, Wim Le Page, and Michael Mampaey. Mining interesting sets and rules in relational databases. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 997–1001. ACM, 2010.

[18] Mohamed Salah GOUIDER and Amine FARHAT. Mining multi-level frequent itemsets under constraints. *International Journal of Database Theory and Application*, 3(4) :15–24, 2010.

[19] Wolfgang Greller and Hendrik Drachsler. Translating learning into numbers : A generic framework for learning analytics. *Journal of Educational Technology & Society*, 15(3) :42–57, 2012.

[20] Fedja Hadzic, Henry Tan, and Tharam S Dillon. *Mining of data with complex structures*, volume 333. Springer, 2010.

[21] J Han and M Kamber. Mining sequence patterns in transactional databases. *Data Mining : Concepts and Techniques*, 2006.

[22] Jiawei Han and Yongjian Fu. Mining multiple-level association rules in large databases. *IEEE Transactions on knowledge and data engineering*, 11(5) :798–805, 1999.

[23] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining : concepts and techniques*. Elsevier, 2011.

[24] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei Chun Hsu. Freespan : frequent pattern-projected sequential pattern mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, pages 355–359, 2000.

[25] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. Citeseer, 2001.

[26] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[27] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation : A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1) :53–87, 2004.

[28] Yu Hirate and Hayato Yamana. Generalized sequential pattern mining with item intervals. *JCP*, 1(3) :51–60, 2006.

[29] Guoyan Huang, Na Zuo, and Jiadong Ren. Mining web frequent multi-dimensional sequential patterns. *Information Technology Journal*, 10(12) :2434–2439, 2011.

[30] Anil K Jain and Richard C Dubes. Algorithms for clustering data. *Englewood Cliffs : Prentice Hall, 1988*, 1988.

[31] Viviane Crestana Jensen and Nandit Soparkar. Frequent itemset counting across multiple tables. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 49–61. Springer, 2000.

[32] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion : A review of the state-of-the-art. *Information fusion*, 14(1) :28–44, 2013.

[33] Hajime Kitakami, Tomoki Kanbara, Yasuma Mori, Susumu Kuroki, and Yukiko Yamazaki. Modified prefixspan method for motif discovery in sequence databases. In *Pacific Rim International Conference on Artificial Intelligence*, pages 482–491. Springer, 2002.

[34] Mark-André Krogel. *On propositionalization for knowledge discovery in relational databases*. PhD thesis, Otto von Guericke University Magdeburg, 2005.

[35] Amaury l'Huillier. *Modéliser la diversité au cours du temps pour comprendre le contexte de l'utilisateur dans les systèmes de recommandation*. PhD thesis, Université de Lorraine, 2018.

[36] Phillip Long. *LAK'11 : Proceedings of the 1st International Conference on Learning Analytics and Knowledge, February 27-March 1, 2011, Banff, Alberta, Canada*. ACM, 2011.

[37] Florent Masseglia, Fabienne Cathala, and Pascal Poncelet. The psp approach for mining sequential patterns. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 176–184. Springer, 1998.

[38] E Ka Ka Ng, A Wai-Chee Fu, and Ke Wang. Mining association rules from stars. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 322–329. IEEE, 2002.

[39] Heiko Paulheim et al. Exploiting linked open data as background knowledge in data mining. *DMoLD*, 1082, 2013.

[40] Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, and Dongqing Yang. H-mine : Hyper-structure mining of frequent patterns in large databases. In *proceedings 2001 IEEE international conference on data mining*, pages 441–448. IEEE, 2001.

[41] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings 17th international conference on data engineering*, pages 215–224. IEEE, 2001.

[42] Jian Pei, Jiawei Han, and Wei Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 18–25, 2002.

[43] Wen-Chih Peng and Zhung-Xun Liao. Mining sequential patterns across multiple sequence databases. *Data & Knowledge Engineering*, 68(10) :1014–1033, 2009.

[44] Helen Pinto. *MULTI-DIMENSIONAL SEQUENTIAL PATTERN MINING*. PhD thesis, SIMON FRASER UNIVERSITY, 2001.

[45] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88. ACM, 2001.

[46] Marc Plantevit, Yeow Wei Choong, Anne Laurent, Dominique Laurent, and Maguelonne Teisseire. M 2 sp : Mining sequential patterns among several dimensions. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 205–216. Springer, 2005.

[47] Marc Plantevit, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, and Yeow Wei Choong. Mining multidimensional and multilevel sequential patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1) :4, 2010.

[48] Julien Rabatel, Sandra Bringay, and Pascal Poncelet. Contextual sequential pattern mining. In *2010 IEEE international conference on data mining workshops*, pages 981–988. IEEE, 2010.

[49] Julien Rabatel, Sandra Bringay, and Pascal Poncelet. Mining sequential patterns : a context-aware approach. In *Advances in Knowledge Discovery and Management*, pages 23–41. Springer, 2013.

[50] Chedy Raïssi and Marc Plantevit. Mining multidimensional sequential patterns over data streams. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 263–272. Springer, 2008.

[51] Thirunavukarasu Ramkumar, Shanmugasundaram Hariharan, and Shanmugam Selvamuthukumaran. A survey on mining multiple data sources. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 3(1) :1–11, 2013.

[52] Sherif Rashad, Mehmed Kantardzic, and Anup Kumar. Pac-whn : Predictive admission control for wireless heterogeneous networks. In *2007 12th IEEE Symposium on Computers and Communications*, pages 139–144. IEEE, 2007.

[53] Star Schemas. Starjoin technology. *Red Brick Systems*, 1995.

[54] Andreia Silva and Cláudia Antunes. Pattern mining on stars with fp-growth. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 175–186. Springer, 2010.

[55] Andreia Silva and Claudia Antunes. Mining stars with fp-growth : A case study on bibliographic data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 19(supp01) :65–91, 2011.

[56] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns : Generalizations and performance improvements. In *International Conference on Extending Database Technology*, pages 1–17. Springer, 1996.

[57] Jerzy Stefanowski and Radosław Ziembinski. Mining context based sequential patterns. In *International Atlantic Web Intelligence Conference*, pages 401–407. Springer, 2005.

[58] Jerzy Stefanowski and Radoslaw Ziembinski. An experimental evaluation of two approaches to mining context based sequential patterns. *Control & Cybernetics*, 38(1), 2009.

[59] Muhammad Usman. *Improving Knowledge Discovery through the Integration of Data Mining Techniques*. IGI Global, 2015.

[60] Xindong Wu and Shichao Zhang. Synthesizing high-frequency rules from different data sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(2) :353–367, 2003.

[61] Li-jun Xu and Kang-lin Xie. A novel algorithm for frequent itemset mining in data warehouses. *Journal of Zhejiang University-Science A*, 7(2) :216–224, 2006.

[62] Zhenglu Yang and Masaru Kitsuregawa. Lapin-spam : An improved algorithm for mining sequential pattern. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1222–1222. IEEE, 2005.

[63] Chung-Ching Yu and Yen-Liang Chen. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering*, 17(1) :136–140, 2005.

[64] Mohammed J Zaki. Spade : An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2) :31–60, 2001.

[65] Mohammed J Zaki and Ching-Jui Hsiao. Charm : An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM, 2002.

[66] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*, 12(3) :372–390, 2000.

[67] Chengqi Zhang, Meiling Liu, Wenlong Nie, and Shichao Zhang. Identifying global exceptional patterns in multi-database mining. *IEEE Intelligent Informatics Bulletin*, 3(1) :19–24, 2004.

[68] Minghua Zhang, Ben Kao, Chi-Lap Yip, and David Cheung. Ffs-an i/o-efficient algorithm for mining frequent sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 294–305. Springer, 2001.

[69] Shichao Zhang, Xiaofang You, Zhi Jin, and Xindong Wu. Mining globally interesting patterns from multiple databases using kernel estimation. *Expert Systems with Applications*, 36(8) :10863–10869, 2009.

[70] Radosław Ziembiński. Algorithms for context based sequential pattern mining. *Fundamenta Informaticae*, 76(4) :495–510, 2007.

Résumé

La digitalisation de notre monde est souvent associée à une production de très grandes quantités de données. Ainsi, des outils de collecte et de stockage de données ont dû être développés, à des fins d'exploitation en recherche ou dans l'industrie. Les données collectées peuvent provenir de plusieurs sources et être hétérogènes, formant ainsi de gros corpus de données hétérogènes. Ces corpus peuvent être analysés pour extraire de l'information ou de la connaissance. C'est l'objet de la fouille de données, qui fait l'objet d'un intérêt grandissant depuis de nombreuses années. Différentes approches de fouille de données ont été proposées, parmi lesquelles la très populaire fouille de motifs. La fouille de motifs, qui inclut la fouille de motifs séquentiels, vise à extraire des motifs ordonnés fréquents dans les données. De nombreux défis restent encoreouverts dans ce domaine, incluant la complexité de la tâche de fouille, la fouille de données hétérogènes ou encore la diminution du nombre de motifs résultat, généralement dû à la redondance entre motifs. Dans le cadre de sources de données multiples, les données peuvent représenter des points de vue différents sur le phénomène représenté et donc sur l'information à extraire. Par ailleurs, la présence de similarité entre certains éléments de données est une caractéristique classique, qui mène à la perte d'information lors du processus de fouille, en raison de la diminution du support des motifs due à cette similarité. L'objectif de cette thèse est de concevoir un algorithme peu complexe de fouille des motifs dans des données multi-source et hétérogènes, dans le but d'extraire une information pertinente tout en compensant la perte de motifs (et donc d'information) due à la similarité entre éléments et en limitant la redondance entre motifs. Plusieurs approches ont été proposées dans la littérature. Certaines fusionnent l'ensemble des sources dans un seul ensemble de données et exploitent un algorithme classique de fouille de motifs, ce qui mène à un algorithme complexe qui extrait un très grand nombre de motifs redondants. D'autres fouillent les sources séparément ce qui peut soit mener à une perte potentielle de motifs ou à des limites quant à la nature des données qui peuvent être fouillées.

Nous proposons G_SPM, un algorithme de fouille de motifs séquentiels qui tire avantage des multiples sources de données à disposition dans le but de pallier le problème de la similarité entre éléments, en formant des motifs généraux. G_SPM adopte une stratégie de fouille sélective de sources, ce qui lui permet d'avoir une complexité limitée. Par ailleurs, G_SPM fouille dans un premier temps une unique source de données, celle qui lui permet d'avoir des motifs séquentiels les plus précis possibles, et ainsi par la suite limiter la redondance entre les motifs formés. Les expérimentations menées confirment que G_SPM identifie des motifs généraux avec un temps d'exécution limité, il permet donc de gérer la similarité entre les éléments en compensant l'éclatement des occurrences sur plusieurs motifs.

Mots-clés: Fouille de motifs, Fouille de motifs séquentiels, Données multi-sources.

Abstract

Huge amounts of digital data have been created across years due to the increasing digitization in our everyday life. As a consequence, fast data collection and storage tools have been developed and thus data can be collected in huge volumes for various research and business purposes. The collected data can come from multiple data sources and can be of heterogeneous kinds thus forming heterogeneous multi-source datasets. These data could thus be analyzed in order to extract valuable information that serves research and business purposes. Data mining has been known as an important task in discovering interesting and valuable information from datasets and has gained a great interest across time. Different approaches in the domain of data mining have been proposed, among which pattern mining is the most important one. Pattern mining, including sequential pattern mining, discovers statistically relevant patterns (or sequential patterns) among data. This domain has grown to be very important where its challenges include discovering important patterns with a limited complexity of the mining process as well as avoiding redundancy among the resulting patterns. Multiple data sources could provide various kinds of data such as descriptive and sequential data which makes the mining process complex. Although these data are multi-source and heterogeneous, there could be problems of data similarity on a data source level which leads to the loss of valuable information and thus limits the number of extracted frequent patterns. The aim of this PhD thesis is to mine data from a heterogeneous and multi-source dataset in order to obtain rich and valuable information and compensate the loss of valuable patterns due to the problem of similarity among patterns. This goal is intended to be reached with a limited complexity of the mining process and with avoiding the redundancy among the resulting patterns. Many approaches have been proposed to mine multi-source data. These approaches either integrate data from multiple data sources and perform a single mining process, which significantly increases the complexity of the mining process and generates a large and redundant set of sequential patterns, or they mine sources separately and integrate the results which could lead to a probable loss of patterns that could have only been extracted when managing different data sources together.

For this sake, we propose G_SPM, a general sequential pattern mining algorithm that takes advantage of the multi-source nature of data in order to mine patterns of a new form, named general patterns, that are more general but with higher data coverage than traditional sequential patterns, and they are rich as they contain various kinds of information. These patterns allow to compensate the loss of patterns caused by the problem of similarity among them. G_SPM adopts a strategy of selective mining among data sources where an indispensable source is mined first, and additional sources are mined only when similarity among patterns is detected which allows to limit the complexity of the mining process and avoids redundancy among the resulting patterns.

The experimental results confirm that G_SPM succeeds in mining general patterns from multi-source data with a limited complexity of the mining process, and it succeeds in handling the problem of similarity among the patterns and compensating their loss. In addition, it outperforms traditional approaches in terms of runtime and redundancy of the resulting patterns.

Keywords: Pattern mining, Sequential pattern mining, multi-source data.