UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

FIB Facultat d'Informàtica de Barcelona

Health History Pattern Extraction from Textual Medical Records

Submitted October 2019, in partial fulfillment of
the conditions for the award of the degree **Master in Innovation and Research in Informatics.**

Natacha Andrea Quintero Vallejos

Supervised by Lluís Padró and Jordi Turmo

Barcelona School of Informatics (FIB) of
Polytechnic University Of Catalonia (UPC) BarcelonaTech

Abstract

Healthcare systems are using Electronic Medical Records (EMR), promoting the use of long datasets that contain relevant information about the life of a patient. The increasing availability of this longitudinal electronic data, represents an impact on the potential discovery of patterns of new diseases, helping the personalized care and increasing the quality of life. An EMR contains the history of hospital encounters, diagnoses, interventions, lab tests and clinical narratives. Extracting frequent patterns from EMR represents a huge challenge in Data Mining, especially since the addition of the concept of time series. Finding if there is any kind of relationship between a drug and a diagnostic or discovering frequent hidden patterns are relevant important areas of interest in healthcare.

In this work, a recursive algorithm called KarmaLego has been implemented for extracting frequent pattern in medical records using data mining techniques. From the modeling point of view, each medical record event is treated as a time point and transformed in an abstract symbol. Each event is connected with another through time relationships using Allen's temporal relations, "before, overlaps and contains". Each abstract symbol is linked to an end date and a start date, representing a time interval series in lexicographic order. The algorithm process these sequences to a 2-size relationships with their counting support. Then, this 2-size sequences goes for undermining patterns recursive increasing its size, to finally obtain the patterns that are frequent.

Acknowledgements

I wish to express my sincere gratitude to Lluís Padró and Jordi Turmo, supervisors of this project, for giving me the opportunity and advices to work in the area that I love: **Healtcare** $<$ 3. I also want to say thanks to the professors Masun Homsni (USB) and Oscar Romero (UPC). Besides my professors, I would like to say thanks to my friends, Hector Lopéz and Iván Aveiga for all the emotional support during all the process.I also want to express my gratitude towards my parents, José Ignacio Quintero and Maria Consuelo Vallejos, but especially to my second father Jorge Navarro Chacón, who was an important inspiration for me and incredible support during the process. Finally I want to say thanks to several people who are no longer here, but represented an important stage of my life. Thank you!

Contents

List of Tables

List of Figures

Abbreviations

BFS Breadth-First Search. 12, 33

DAG Directed Acyclic Graph. 13–16, 33, 45

DFS Depth-first search. 17, 33

EMR Electronic Medical Records. i, 2, 57

TIRP Time Intervals Relations Pattern. 9, 20, 33, 58

Chapter 1

Introduction

1.1 Motivation

Data Science is the science of applying techniques and theories with the goal of collecting, managing, and analyzing large and complex datasets, extracting knowledge and answers from them. In recent years, this science has represented the base for development in many areas such as finance, education, cyber security, Healthcare, etc. With the arrival of the big data, where data grow more and more, and at full speed, data mining is facing challenging problems. The data mining principal goal is the extraction of information from data, searching hidden patterns and tendencies to explain unknown behaviors in a specific domain. The pattern matching problem is one of the most interesting challenges of Data mining. This problem is represented as association rules for discovering relationships between variables in large datasets, sets of items, or transaction databases. Another pattern matching problem is related to solving sequential pattern mining, which is associated with finding statistically relevant patterns in items in sequence databases. The approach can become complex adding the time constraint, extending the problem to temporal data mining techniques. Focus in medical records, finding frequent relationships between events that occurs in patients during their life, it is one of the most important problems in Healthcare.

Currently Healthcare systems are using EMR, promoting the use of large datasets with

health information. The increasing availability of this longitudinal electronic data represents an impact on the potential discovery of patterns, helping the personalized care of each patient and increasing the life's quality. An EMR contains the history of hospital encounters, diagnoses, interventions, lab tests and clinical narratives. An interesting part of EMRs is the temporal information. Thanks to this information, new analysis can be done to discover new frequent temporal patterns or treatment pathways, allowing the doctors and experts to gain years in this area.

The pattern search problem in medical records is reduced to a pattern matching problem using time data mining techniques in a raw-data with a time-oriented study. In [27] the introduction of the problem for sequential pattern recognition for medical record analysis is explained. The algorithm proposed by Robert Moskovitch and Yuval [25] KarmaLego represents a good approach for solving pattern matching with temporal constraints, including in the processing phase, the addition of the concept of knowledge-based temporal abstraction [23], focus on raw data and time-stamped.

1.2 Approach proposal

The work on this thesis is focused on the development of the algorithm KarmaLego with some improvements respective of the structure and design using MIMIC III[14].MIMIC III[14] database has the information of the medical records of the patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012.The problem is divide in two phases.The first problem that is encountered as a challenge is the design and modeling of raw-date data. Taking into account that it is data with a high-level of abstractions, the method of Temporal Abstraction through the process of *"discretization"* is used and the raw-date is transformed into abstract concept with clear meanings. The second problem is the recursive mining process for finding the frequent patterns. Each of these steps will be explained in the next chapters.

1.3 Objectives

1.3.1 General Objective:

Research and develop methods to perform behaviour pattern extraction from electronic health records in the framework of the Graph-MED project.

1.3.2 Specific objectives

- Process textual records that were extracted using NLP techniques.
- Model the data using a good approach of Data Structure.
- Apply Temporal Data Mining
- Extract frequent patient behavior to detect new counter-indications, secondary effects, and hidden relationships between the events of a medical records.

1.4 Document Structure:

The rest of the thesis is structured as follows. Chapter two discusses the background of the thesis. Chapter three discuss the state of the art, all the previous solutions and their improvements. Chapter four is data description, understand the data used, scope and pre-processing. Chapter five, explains the algorithm and their different approaches. Chapter six contains the experiments and results. Chapter seven describes the evaluation method used. Chapters eight presents the information about the implementation and tools, Chapters nine contains the conclusion and chapter 10 contains some research lines and possible features for further work.

Chapter 2

Background

2.1 Temporal Data Mining

The goal of data mining is the extraction of information from data, searching hidden patterns and tendencies to explain unknown behaviors in a specific domain. Adding the concept of time to data mining, it comes temporal data mining, a set of time-orient data techniques for discovering temporal knowledge, relationships between raw data. The challenge of this sub-field is the application of different methods with the addition of time to find patterns between sequences and sub-sequences of an event with the representation and modeling of a data sequence.

One of the domains, where the challenges and applications of temporal data mining has been growing exponentially has been Healthcare. The data extracted from electronic devices and medical records is represented as temporal sequences and used to study the relationship between events of patient's life. Actually the concept of time is representing the end of a gap and the beginning of the exploration of new horizons for the science. Not only static data, also, the study of data coming on real-time need to be studied and analyzed for finding abnormal behaviors at big scale.

2.2 Allen's interval algebra

Allen's interval algebra is a set of thirteen jointly exhaustive and pairwise disjoint binary relations, based on seven basic temporal relations and their inverses. These relations are temporal relationships between pairs of time intervals. Allen's [7][2] thirteen relations over temporal intervals explain the base of the study qualitative temporal concepts and representation. The interval relations are meets, before, starts, ends, overlaps, during, their inverses met by, after, started by, ended by, overlapped by, contains, and equality. In Algebra, these relations model the intersection, union and composition between two pair of temporal relations. In the Figure 2.1 are the seven general Allen's temporal relations.

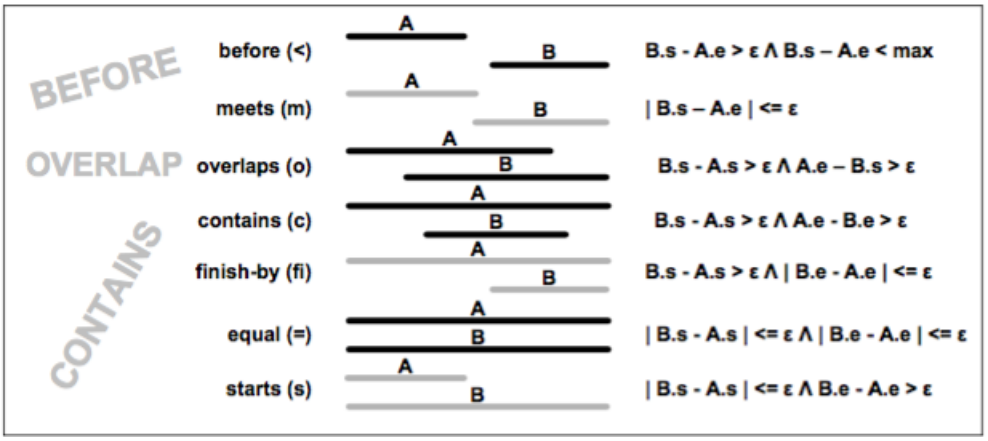

Figure 2.1: Allen's Temporal Relations

2.3 Temporal Abstraction

The concept of temporal abstraction for clinical data analysis represents an important practice of getting qualitative concepts and definition of an abstract concept. Temporal abstraction is the process of giving description as a symbolic definition for a time-point series, making possible to achieve a piece of knowledge and coherent meaning. This definition was studied in [25][23] for applying in medical records raw-data. Focus on the particular problem of medical records as raw data, suppose the study of an element "Hemoglobin in the blood with an X value", also adding the constraint of "taking a

blood sample during the last week obtaining the same value X". This can be explain like for a period of time "one week, the hemoglobin value is high", "is low", "is normal", is "abnormal", represented as a symbol "high hemoglobin during a week".

2.4 Knowledge Base temporal Abstraction

Knowledge-based Temporal Abstraction (KBTA) is a problem-solving method, based on artificial intelligence techniques, developed by Shahar [23]. This method can be used in a lot of domains, but originally was developed only for the medical domain. This technique works at the point-based raw data. As an example of raw-data timestamp input, let's say that the element Hemoglobin have as an output, the constitution of a symbol "normal or"abnormal,"high,"low of specific elements, giving a temporal interpretation of the context. [18] [24]. In [24] there is two primary output classes of abstractions generated by the KBTA method:

- **State Abstraction:** the values of one or more variables that hold over the same time point are classified into a symbolic state value (e.g., High) according to the state abstractions cutoff definitions (or, in general, a state-abstraction function).
- **Gradient Abstraction:** the first derivative of each vector of consecutive time point values is classified into the values Increasing (I), when it is sufficiently positive, Decreasing (D) when it is sufficiently negative, and otherwise Stable (S), using a significant-change cutoff value.

In the Figure 2.2 there is an example of how time points look like and their representation in Gradient or State.

2.5 Temporal *Discretization*

This refers to the process of *"discretization"* for the time series-values in an unsupervised method performed in the pre-processing step, transforming the raw data in a symbolic, stated-based time interval. Basically, it is the process of applying the KTBA used by

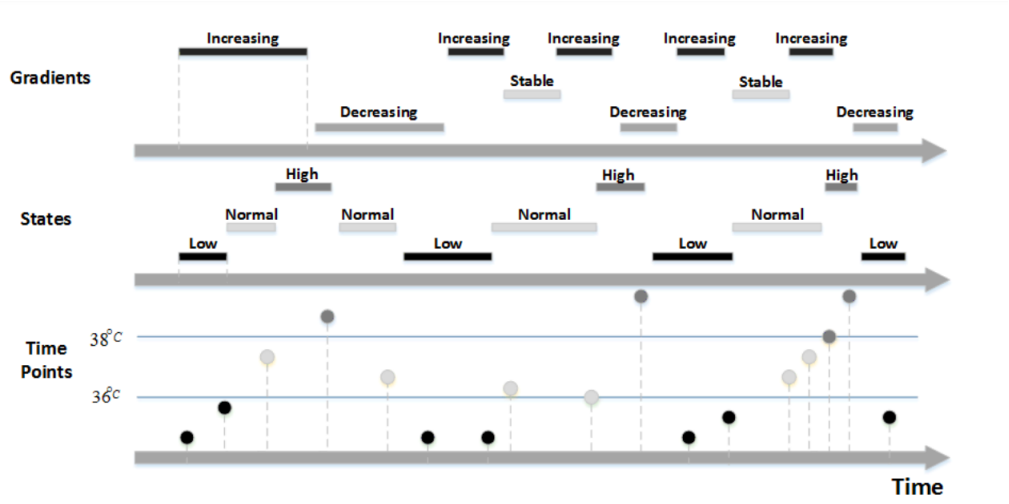

Figure 2.2: Gradient, States and Time Points

Hoppner in [12]. The important point regarding *"discretization"* is the transformation of numeric time-series respecting the order of the values for more meaningful interpretations.

2.6 Time Interval Mining

The importance of Time Data Mining and the addition of the concept of time was already explained, also the use of abstract concept applied to raw data for *"discretization"* to be able to model the data mining problem. Now the problem is multivariate symbolic time intervals that work with Allens temporal relations. The addition of a symbolic time intervals called AI pattern was proposed by Kam and Fu [15]. AI pattern is finding a problem of an ambiguity taking into account that the temporal relations between the components that were not successive. Hoppner [11] was the first in explain time intervals without ambiguity using a matrix k^2 to represent all of the pair-wise relations of k-intervals.

2.7 Definitions

The following definitions were described by Robert Moskovitch and Yuval Shahar.

Definition 2.1 ***Flexible framework of Allen's temporal relations***. *Given two relations in time-stamped data and an epsilon value:*

$$\textbf{\textit{(1)}}\ t_1 =^{\epsilon} t_2\ \textbf{\textit{iif}} \mid t_2 - t_1 \mid \leq epsilon$$

$$\textbf{\textit{(2)}}\ t_1 <^{\epsilon} t_2\ \textbf{\textit{iff}}\ t_2 - t_1 > epsilon$$

The Epsilon parameter to Allen's temporal relations maintain the Jointly Exhaustive and Pairwise Disjoint (JEPD) conditions that refers to the probability theory and explain that "two sets A and B are disjoint if their intersection is the empty set". This maintains the relation A and B mutually exclusive.

Definition 2.2 ***Symbolic Time interval.*** *A symbolic time interval* $I =< s, e, sym >$ *is an ordered pair of time points, start-time (s) and end-time (e), and a symbol (sym) that represents one of the domain's symbolic concepts.*

Definition 2.3 ***Symbolic Time Tnterval Series***. *A symbolic time interval series* $IS = \{I_1, I_2, I_n \dots\}$, *where each* I_i *is a symbolic time interval, represents a series of symbolic time intervals, over each of which holds a symbolic concept.*

Definition 2.4 ***Lexicographic symbolic time-interval series*** *A lexicographic symbolic time-interval series is a time-interval series, sorted in the order of the start-time, end-time using the relations* $<^{\epsilon}, =^{\epsilon}$ *and a lexicographic order of the symbols* $IS = \{I_1, I_2, I_n \dots\}$, *such that:*

$\forall I^i, I^i \in IS \mid (i < j) \wedge ((I^i_s <^{\epsilon} I^j_s) \vee (I^i_s =^{\epsilon} I^j_s \wedge I^i_e <^{\epsilon} I^j_e) \vee (I^i_s =^{\epsilon} I^j_s \wedge I^i_e =^{\epsilon} I^j_e \wedge I_{sym^i} < I_{sym^i}))$

Definition 2.5 ***Non-ambiguous lexicographic Time Intervals Relations Pattern (TIRP)*** *A non-ambiguous lexicographic Time Intervals Relations Pattern (TIRP) P is defined as* $P = \{I, R\}$, *where* $I = I_1, I_2, I_k \dots$ *is a set of k symbolic time intervals ordered lexicographically and defines all the temporal relations among each of the* $(k^2 - k)/2$ *pairs of symbolic time intervals in I. Defined as:*

$R = \bigcap_{i=1}^{k} \bigcap_{j=1+i}^{k} r(I_i, I_j) =$

$= \{r_{1,2}(I_1, I_2), r_{1,3}(I_1, I_3) \dots r_{1,k}(I_1, I_k), r_{2,3}(I_2, I_3), r_{2,4}(I_2, I_4) \dots r_{2,k}(I_1, I_k), r_{k-1,k}(I_k - 1, I_k)\}$

Example of a TIRP in the Figure 2.3.

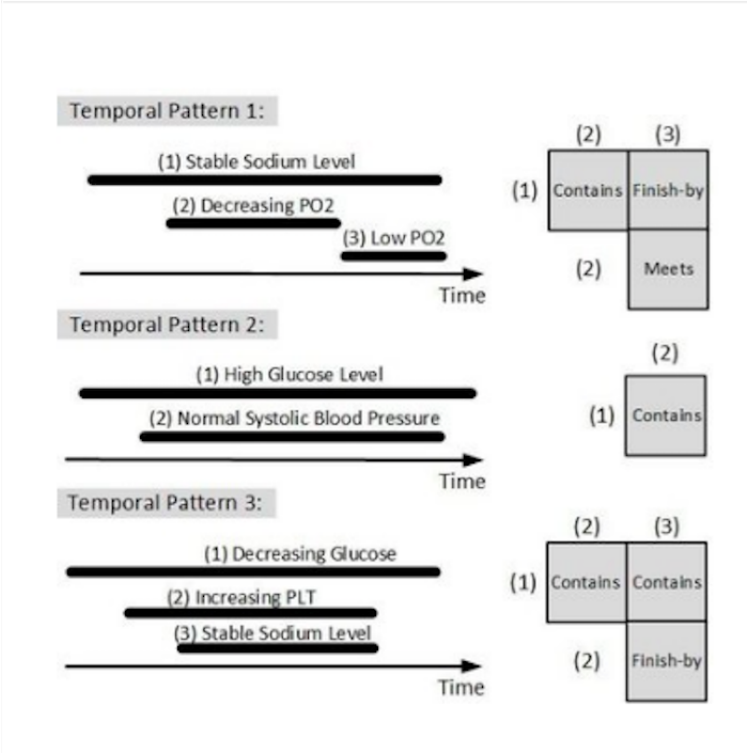

Figure 2.3: Temporal Pattern

Definition 2.6 ***Vertical support of a TIRP***

Having a database | E | of different patients, the vertical support of a TIRP P is the cardinality of E_p of distinct entities that holds P divide by the number of patients.

Definition 2.7 ***Horizontal support:*** *The horizontal support of a TIRP P for an entity ei (e.g., a single patient's record), $hor\_sup(P, ei)$ is the number of instances of the TIRP P found in E_p.*

Definition 2.8 ***The mean horizontal support*** *of a TIRP P is the average horizontal support of all the entities EP supporting P (i.e., for all entities with horizontal support≥1).*

Definition 2.9 ***Relax Horizontal support fixed****:*

Horizontal support fixed of a TIRP is the number of instances of the TIRP found in E_p.

Chapter 3

State of the Art

The pattern matching problem represents one of the most important aspects of data mining. The fact of looking for interesting patterns in various items and the relationships between them in large data sets or databases has been studied for many years. Many algorithms have been designed to solve this problem, and broadly, they are divided in two big groups. The first is related to association rule mining algorithms, find frequent patterns, associations, correlations, or causal structures in sets of items or objects in transaction databases. The second one is related to solving sequential pattern mining, which is related to finding statistical relevant patterns between data examples where the items are part of sequence databases. If we add the constraints of time, we can extend the last problem to pattern matching sequences using time data mining techniques.

3.1 Association Rule Mining in Transactional Databases

Definition 3.1 *Association Rule and Transactional Databases. Let* $I = \{i_1, i_2, i_d \dots\}$ *be a set of all items in a database and* $T = \{t_1, t_2, t_d \dots\}$ *be the set of all the transactions. Each* t *in* T *contains a subset of items that are inside I. So if an itemset contains k items, it can called a k-itemset. If* x *is inside a transaction* t_i*, then* $x \subseteq t_i$*.*

Definition 3.2 *Support count is the number of transactions that contains an itemset.*

Can be expressed as: $\psi(x) = \{t_i \mid X \in t_i, t_i \in T\}$

The Table 3.1 is an example of a transactional database. The columns Items Bought represents the set of the items that the customer bought, and the second column is the items sorted by frequent.

TID	Items Bought	(Ordered) Frequent Items
100	f,a,c,d,g,i,m,p	f,c,a,p
200	a,b,c,f,i,m,o	f,c,a,b,m
300	b,f,b,j,o	f,b

Table 3.1: Example of Transactional Database

3.1.1 Apriori Algorithm

APriori [16] was proposed in 1994 for Agrawal and Srikant.This algorithm was designed to operate on the basis of transaction data, where each transaction is seen as a set of items. Apriori algorithm makes multiple passes over the database, working with an iterative approach of Breadth-First Search (BFS), where k-itemsets are used to explore (k+1)-itemsets. It follows a property that says "All nonempty subsets of a frequent itemsets must be frequent" and anti monotonic property which says,"if the system cannot pass the minimum support test, all its super set will fail to pass the test," this means that if a set is infrequent, their supersets will be infrequent too.

The algorithm makes a first pass to the database finding the 1-item set that supports a C threshold, having sets with one item T. Then, a seed set of item sets T is used to generate new potentially large item sets called candidates, to finally count them doing another pass over the data. After this step, the actual item sets that are frequent, become the new seed for the next step. T is be used to create frequent 2-item-sets, and so on, until no more frequent k-item sets can be found. Basically, the process can be reduced in three phases:

- Find the 1-item that are frequent(above a minimum support).

- Generate Candidates.
- Counting support.

One of the problems of Apriori is the candidate generation,that is extremely expensive, but its complexity could be significantly reduced by using pruning processes and optimization techniques.

3.1.2 Apriori improvements

There also exists some improvements for Apriori [5] that try to fix the problem of candidate generation using better data structures and adding pruning steps, like the one proposed by Jiawei Han that works without candidate generation, FP-growth[10][8][20]. The bottleneck of Apriori (Candidate Generation and Counting support) was improved using a data structure called frequent pattern tree (FP-tree), which extends prefix-tree structures that store the important information about frequent patterns. It finds only the frequent length-1 items in the nodes of the tree, and more frequently, a node can occur it will have more chances of sharing the nodes.
FP-growth[3][4][5] works with three techniques:

- Compression of the large database into a data structure called FP-Tree, avoiding multiple scans. The scope from the table 3.1 above represents the first scan of the database.
- FP-Tree Mining process with a better performance in candidate generation.In the Figure 3.1 the FP-Tree structure and how the construction of this it is made iterative.
- Divide-conquer method to decompose the mining task.

This algorithm has disadvantages too; it also generates large numbers of conditional FP-Trees. This disadvantage was resolved using the FP-Tree as Directed Acyclic Graph

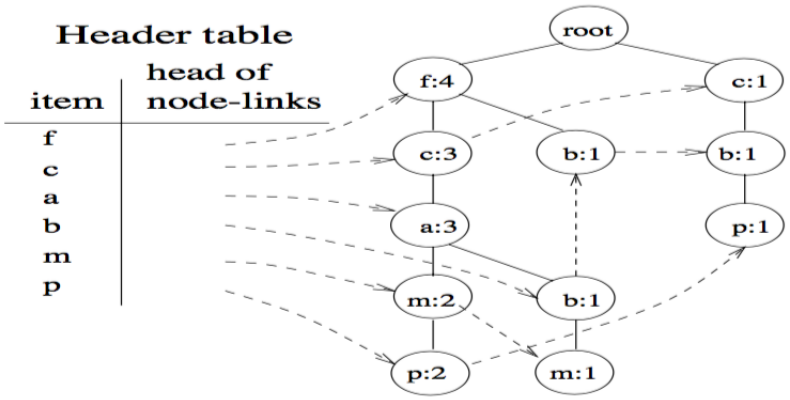

Figure 3.1: FP-Tree

(DAG) [21],so it would not be necessary to generate a conditional FP-Tree as the recursive element is connected using DAG.

The last two algorithms as Apriori [16] and FP-GROTH [10][8][20] followed the concepts of a transactional database, which means they don't take into account the order of the items. Here, the addition of the concept of ordering and sequence must be explained as sequential pattern mining.

3.2 Sequential Pattern Matching and Sequence Databases

Definition 3.3 *This problem was first proposed in [1] as: Let D be a sequence database and Let* $I = \{x_1, x_2, x_d \dots\}$ *be a set of m different items,* $s = \{s_1, s_2, s_d \dots\}$ *is a sequence that contains a list of ordered items. We can say that an itemset is* $s_i \in I$*.Having* $S_x = \{x_1, x_2, x_d \dots\}$ *and* $S_y = \{y_1, y_2, y_d \dots\}$*,we can say that* S_x *is a sub-sequence of* S_y *if of all items in a database and* $T = \{t_1, t_2, t_d \dots\}$ *be the set of all the transactions. Each t in T contains a subset of items that are inside I. So if an itemset contains k items, we can called a k-itemset.If x is inside a transactions* t_i*, then* $x \subseteq t_i$*. Now the problem is that given a set of sequences or a database of sequence and a support threshold, find the complete set of frequent sub-sequences.*

3.2.1 Sequential Pattern Matching algorithms

A good approach to solve the problem of sequence pattern matching is found in GSP [26]. In this algorithm, the concept of ordering transaction time is also added. The author explains the importance of what he calls transaction time associated with a transaction.

The author explains with an example, the new definition problem. Let's say a database of a Book-Club, and each data-sequence corresponds to all book selections of a customer, and each transaction are the books selected by the customer in order. If a customer buys book A, then buy book B and then book C, $(A > B > C)$, and then another customer that buys books between these ones, still contain the sequential pattern. Applying this approach in Healthcare domain using the specific problem of the pattern sequence surrounding medical records, a data-sequence corresponds to laboratory events, followed by prescriptions, followed by a diagnosis, and then another prescription during a visit to a doctor. In this case, the study of an existence of a relationship between a disease and a medication can be done. GSP [26] algorithm try to solve the limitations:

- **Not Time Constraints**: The existence of time intervals between each adjacent elements of the sequential pattern. In our particular case, it is relevant to know if after taking a medication, six months later, a patient will have some symptoms.

- **Rigid Definition of Transaction**: Each element of a pattern can be present in two different transactions as long the transaction time between them are within some small-time window.

- **Not Taxonomies:** For some databases, it is important to use user-taxonomies to find a pattern in different levels.

The algorithm of GSP [26] adds the time constrains, in sequential patterns and works according to the problem of discovering sequence of events presented in [29], where their pattern is a DAG,each vertex is an event, and the edge represents the concepts of time. The problem statement of GSP [26] is:

Let $I = \{i_1, i_2...i_d \dots\}$ be a set of literals, called items. Let T be a DAG on the literals. An edge in T represents an is-a relationship, and T represents a set of taxonomies. If there is an edge in T from p to c, it call p a parent of c and c a child of p. (p represents a generalization of c.). The taxonomy is model as a DAG rather than a tree to allow for multiple taxonomies. Le'ts call x_b an ancestor of x (and x a descendant of x_b) if there is an edge from x_b to x in transitive-closure(T). An itemset is a non-empty set of items. A sequence is an ordered list of itemsets. A sequence is denoted as $s = \{s_1 s_2...s_n \dots\}$ i, where s_j is an itemset. We also call sj an element of the sequence. We denote an element of a sequence by $(x_1, x_2, ..x_m)$, where x_j is an item. An item can occur only once in an element of a sequence, but can occur multiple times in different elements. An itemset is considered a sequence with a single element. We assume without loss of generality that items in an element of a sequence are in lexicographic order.A sequence $< a_1 a_2...a_n >$ is a subsequence of another sequence $< b_1 b_2...b_m >$ if there exist integers $i_1 < i_2 < ... < i_n$ such that $a_1 \in b_i 1$, $a_2 \in b_i 2$, ...,$a_n \in b_i n$.

The algorithm makes multiple passes over the database. The first pass is to determine if an item is frequent; These items represent the 1-itemset. After this, each subsequent takes one seed, and this seed generates all the new potential frequent sequences, called candidates sequences, that basically are one size bigger than the current sequence. Once the sequence is created, the support count passing through the data again, determining if the new sequence is actually frequent. These new sequences become the seeds for the next pass. It follows the same steps as Apriori, with the difference that the supporting count is now defined for a sequence as the faction of the total data-sequence that actually contains the sequence, adding taxonomies (ancestor concept) and sliding windows (when a data-sequence contribute to the support count of a sequence by allowing a set of transactions to count an element of a sequence).

Another algorithm called SPAM [3] resolves the sequence pattern matching, also adding time constraints using a bitmap representation. The description of the algorithm assumes a lexicographic $\leq$ of the items in a database, so if an item i occurs before j in the ordering,

then we denote i≤j. This ordering is extensible to sequences. This approach considers all sequences arranged in a sequence tree or as a DAG, where if i comes before j, then i ≤j and j is a child of a node i. In Figure 3.2, there is an example of a lexicographic tree. The algorithm works with Depth First Tree Traversal approach to traverse the tree to extend the sequences, where at each node n, the support of each sequence-extended child and each itemset-extended child is tested. If it is above the minimal support, then the sequence is stored, if not, it will stop following the Apriori principle [16]. Depth-first search (DFS)

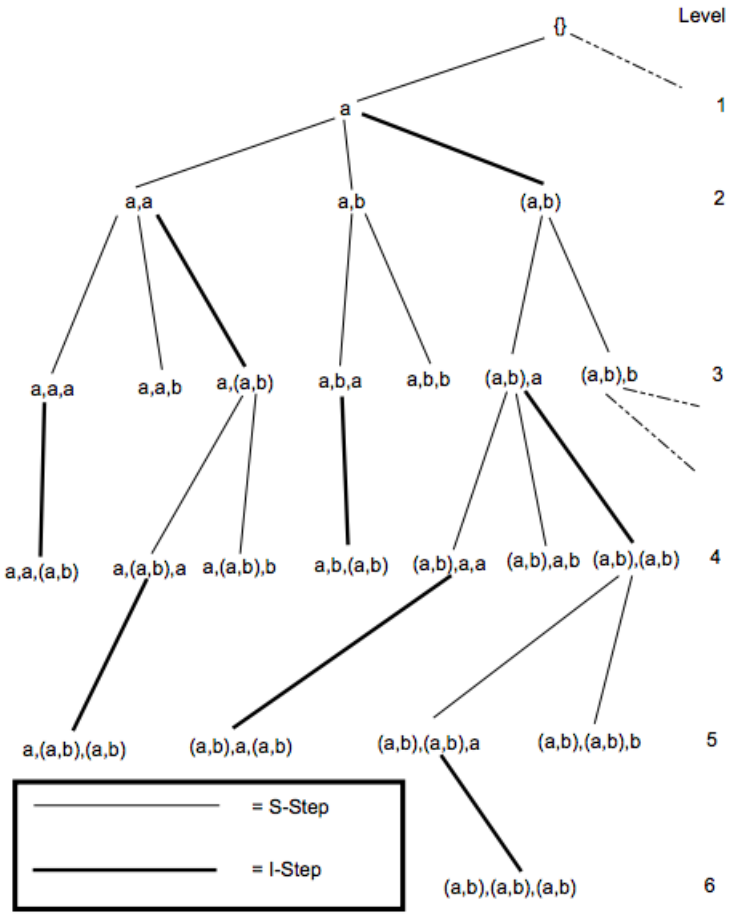

Figure 3.2: Lexicographic Tree

3.2.2 Sequential Pattern Matching and Projected Sequence Databases

FreeSpan [9] proposed a novel efficient sequence pattern mining method using a projected sequence database. It follows the same approach as GSP, at the first part of the algorithm, scanning the database to find the length-1 sequential pattern, and sorting in descending order, b:5,c:4, then the complete set of sequential patterns are divided into six subsets. Lets say after the first pass of the database there is something like b:5 c:4 , a:3, d:3, e:3, f:3, so we can do: (1) those having an item f, (2) those having an item e, but not f, (3) those that having item d but no e or f, and so on. FreeSpan uses a divide-conquer method to find the complete sets. Then, they use a matrix to construct the length-2 sequence formed with the list of frequent 1-length before. This matrix is a triangular matrix and is used to generate length-3 and longer sequential patterns. For an itemset, the X-projected dataset is the collection of a sequence having all the items in X.

PrefiXSpan [13] made an improvement of FreeSpan creating a pseudo projection technique trying to avoid, at every possible position, a potential candidate. Thinking that the item within an element of a sequence can be listed in any order, it can be assumed that they are always listed alphabetically. So, after doing the first pass of the database and having the 1-length frequent, they find the subsets of the projected databases using a prefix, as in the Figure 3.3. The pseudo projection technique, instead of performing physical projection, registers the index (ID) of the corresponding sequence and the starting position of the projected suffix of the database. With this, the physical projection of the sequence will be replaced by the sequence ID. Another approach it can be found in PSPM [17], this algorithm uses Apriori property to delete the non-frequent items and divide search space and at the second step It records projected position to locate project sequence position for mining local frequent items and mines each one recursively.

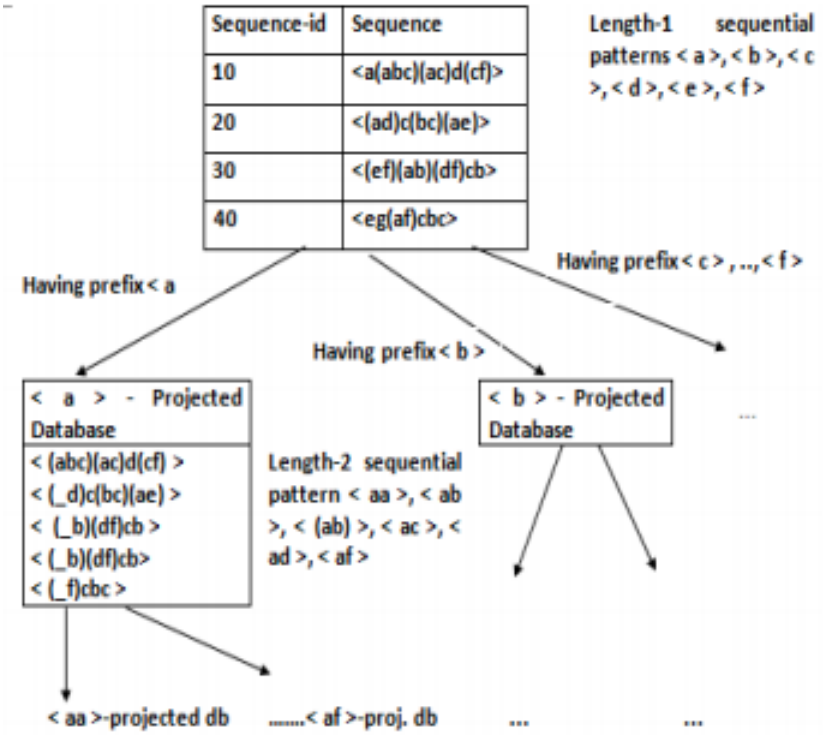

Figure 3.3: Prefix Span pseudo projected Database

3.2.3 Pattern Matching Sequences using Time Data Mining Techniques

There are algorithms like SPADE [29] and UDDAG [6] that resolve the problems of finding pattern sequence, as well as the use of transaction time base constraints, taxonomies, and slide windows. Following the same approach of sequence data mining, some improvement are made using different data structures and techniques to reduce the candidate generation, projection databases, and physical storage.

Let's introduce the new constrains of using time data mining techniques with time in the raw-data. It is important to mention the ingest of new limitations on specific case of study of patterns in medical records. In [27] there is an introduction of the problem for sequential pattern recognition for medical record analysis, and and a good proposal has been made by Robert Moskovitch1 and Yuval [25] with an algorithm called KarmaLego, including in the processing phase the concept of knowledge-based temporal abstraction [23], focus on raw and time-stamped data.

This algorithm works with temporal relations between pairs of intervals called TIRP. Papapetrou et al. (2009)[19] proposed a hybrid approach H-DFS, which combines the first indexing pairs of time intervals and then mining the extended TIRPs in a candidate generation. Papapetrou et al. used five temporal relations: meets, matches (equal, in terms of Allens relations), overlaps, contains, and followed and introduced an epsilon threshold, to make the temporal relations more flexible. ARMADA, by Winarko and Roddick (2007)[28] projected works with for mining non-ambiguous temporal patterns from interval-based events. KarmaLego took the idea introduced by Patel et al. (2008) IEMiner - a method inspired by Papapetrous method, which extends the patterns directly, using transitivity. Basically, this algorithm takes the improvement of each of their phases of candidate generation and support counting.

Chapter 4

Data Description

4.1 Data Source

The data used during the realization of this work were obtained from Physionet and is called of MIMIC-III [14]. MIMIC III is a relational database that initially contains 26 tables. The dataset is health-related data associated with patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. The data were transformed into XML format,and represents a medical record that has a series of events that describe the clinical life of a patient. In total, it contains information about 40,000 patients, but 10,422 patients with cardiovascular problems were selected for the study.

The data includes demographic information, vital signs, laboratory measures, microbiology, examinations, results, procedures, medications, hospital entrance and exit, diagnoses, patient mortality information, among other data of a clinical resume. This dataset is rich in information of each of their events in a large population of patients, so its granularity allows to make more accurate studies and obtain much more interesting and relevant results. The general information of a patient is distributed as:

- **Diagnosis Event**: This event represents the information about the diagnosis that has been given to at patient in a certain time (***EVENT TIME***) and the identification is ***ICD9_CODE***, which is the reference code of a disease.

EVENT_TYPE	EVENT_TIME	SUBJECT_ID	ICD9_CODE
DIAGNOSE	2142-04-24 06:55:00	111	486
DIAGNOSE	2142-04-24 06:55:00	111	49121
DIAGNOSE	2142-04-24 06:55:00	111	4139
DIAGNOSE	2142-04-24 06:55:00	111	51881
DIAGNOSE	2142-04-24 06:55:00	111	41081

Figure 4.1: Example of Event Diagnose

- **Prescription Event**: This event has information about a prescription of a drug given to a patient. ***STARTDATE*** and ***ENDDATE*** represent the beginning of the treatment and the end respectively. It also has information on the name of the drug, generic name, the dose, the type of drug, and the NDC identification of the drug supplied.

SUBJECT_ID	FORM_UNIT_DISP	STARTDATE	ENDDATE	GSN	DOSE_VAL_RX	PROD_STRENGTH	DRUG	FORMULARY_DRUG_CD	DRUG_NAME_GENERIC
111	INH	2142-04-24 00:00:00	2142-04-26 00:00:00	005037	6-15	17g Inhaler	Albuterol	ALBU17H	Albuterol Inhaler
111	ml	2142-04-24 00:00:00	2142-04-24 00:00:00	041384	25-100	100mcg/2mL Amp	Fentanyl Citrate	FENT2I	Fentanyl Citrate
111	INH	2142-04-24 00:00:00	2142-04-26 00:00:00	005037	6	17g Inhaler	Albuterol	ALBU17H	Albuterol Inhaler
111	INH	2142-04-24 00:00:00	2142-04-26 00:00:00	005037	6	17g Inhaler	Albuterol	ALBU17H	Albuterol Inhaler
111	INH	2142-04-24 00:00:00	2142-04-26 00:00:00	059081	6-8	12.9g Inhaler	Ipratropium Bromide MDI	IPRAPF	Ipratropium Bromide HFA

Figure 4.2: Example of Event Prescription

- **Procedure Event**: This event has information about procedures applied to the patient during their stay in a hospital, the category of the procedure performed and comments on the procedure.

ENDTIME	STARTTIME	SUBJECT_ID	HADM_ID	ROW_ID	VALUEUOM	LOCATION	CGID	STATUSDESCRIPTION
2171-12-24 14:04:00	2171-12-21 18:46:00	902	143497	210549	min	Right Brachial.	18539	FinishedRunning
2199-12-26 18:17:00	2199-12-24 20:00:00	357	117876	29780	min	Right Femoral.	17485	FinishedRunning
2199-12-27 01:45:00	2199-12-24 20:00:00	357	117876	29781	min	L Hand	17485	FinishedRunning
2146-10-04 03:15:00	2146-10-03 08:39:00	854	175684	138518	min	Right Femoral.	20482	FinishedRunning
2142-06-27 01:31:00	2142-06-26 12:00:00	222	188038	4674	min	R Hand	19783	FinishedRunning

Figure 4.3: Example of Event Procedure

- **Laboratory Event**: This event contains all the information related to laboratory test made for a patient, category of the test, event time, fluid of the body where the test was done (Urine, Blood,etc.), label description, etc.

ITEMID	EVENT_TIME	SUBJECT_ID	CATEGORY	LOINC_CODE	VALUENUM	VALUE	FLUID	FLAG	LABEL	VALUEUOM
50868	2142-04-24 03:15:00	111	Chemistry	1863-0	18.0	18	Blood	normal	Anion Gap	mEq/L
50882	2142-04-24 03:15:00	111	Chemistry	1963-8	22.0	22	Blood	normal	Bicarbonate	mEq/L
50893	2142-04-24 03:15:00	111	Chemistry	2000-8	9.2	9.2	Blood	normal	Calcium, Total	mg/dL
50902	2142-04-24 03:15:00	111	Chemistry	2075-0	104.0	104	Blood	normal	Chloride	mEq/L
50910	2142-04-24 03:15:00	111	Chemistry	2157-6	123.0	123	Blood	normal	Creatine Kinase (CK)	IU/L

Figure 4.4: Example of Event Laboratory

- **Microbiology Event**: this event represents the microbiology's test made to a patient. The ***ORG_NAME*** is the name Specimen, which is tested for bacterial growth. ***ORG_ITEMID***, ***ORG_NAME*** is the organism, if any,that grew when tested. If NULL, no organism grew (i.e. negative culture). *INTERPRETATION* contains the result of the antibiotic sensitivity, and indicates the results of the test. S is sensitive, R is resistant, I is intermediate, and P is pending.

EVENT_TYPE	EVENT_TIME	SUBJECT_ID	INTERPRETATION	ORG_NAME	AB_NAME	ORG_ITEMID	SPEC_TYPE_DESC
MICROBIOLOGY_EVENT	2144-07-01 07:42:00	111	S	STAPH AUREUS COAG +	GENTAMICIN	80023.0	SPUTUM
MICROBIOLOGY_EVENT	2144-07-01 07:42:00	111	S	STAPH AUREUS COAG +	TRIMETHOPRIM/SULFA	80023.0	SPUTUM
MICROBIOLOGY_EVENT	2144-07-01 07:42:00	111	S	STAPH AUREUS COAG +	ERYTHROMYCIN	80023.0	SPUTUM
MICROBIOLOGY_EVENT	2144-07-01 07:42:00	111	S	STAPH AUREUS COAG +	CLINDAMYCIN	80023.0	SPUTUM
MICROBIOLOGY_EVENT	2144-07-01 07:42:00	111	S	STAPH AUREUS COAG +	LEVOFLOXACIN	80023.0	SPUTUM

Figure 4.5: Example of Event Microbiology

- **Admission Event**: This event contains information about the entrance of a patient in a hospital. It also has demographic information like ethnicity, marital status, languages, etc..

4.2 Data Quality

The MIMIC dataset initially has 26 tables that were pre-processed with XML during a pre-study phase for the beginning of the thesis. From the XML, only the events that represent

EVENT_TYPE	EVENT_TIME	SUBJECT_ID	ADMISSION_TYPE	ADMISSION_LOCATION	MARITAL_STATUS	DIAGNOSIS	DEATHTIME
ADMISSION	2144-07-01 04:12:00	111	EMERGENCY	EMERGENCY ROOM ADMIT	MARRIED	PNEUMONIA	2144-07-01 14:55:00
DISCHARGE	2144-07-01 14:55:00	111	EMERGENCY	EMERGENCY ROOM ADMIT	MARRIED	PNEUMONIA	2144-07-01 14:55:00
ADMISSION	2188-11-12 09:22:00	250	EMERGENCY	EMERGENCY ROOM ADMIT	SINGLE	PNEUMONIA,R/O TB	2188-11-22 12:00:00
DISCHARGE	2188-11-22 12:00:00	250	EMERGENCY	EMERGENCY ROOM ADMIT	SINGLE	PNEUMONIA,R/O TB	2188-11-22 12:00:00
ADMISSION	2142-08-28 19:48:00	109	EMERGENCY	EMERGENCY ROOM ADMIT	SINGLE	SHORTNESS OF BREATH	2142-08-30 15:20:00

Figure 4.6: Example of Event Admission I

LANGUAGE	RELIGION	INSURANCE	ETHNICITY
ENGL	PROTESTANT QUAKER	Medicare	WHITE
ENGL	PROTESTANT QUAKER	Medicare	WHITE
HAIT	NOT SPECIFIED	Self Pay	BLACK/AFRICAN AMERICAN
HAIT	NOT SPECIFIED	Self Pay	BLACK/AFRICAN AMERICAN
ENGL	NOT SPECIFIED	Medicaid	BLACK/AFRICAN AMERICAN

Figure 4.7: Example of Event Admission II

relevance for the pattern search were extracted. In order to reduce the complexity of the algorithm and obtain information that is relevant to the study, only six events were used for each medical record. The events mentioned in the data sources section were pre-processed and their features were reduced until finding the elements whose selection were adapted for the approach.

During the first phase of the pre-processing phase, the data passed for a first round to reduce complexity. For that reason, the first approach was to count the number of times that a diagnosis appeared in the total of patients. A diagnosis has a unique value called ***ICD9_CODE***. Those diagnoses that appeared less than 20 times were eliminated represeting the 0.7470 % of the total diagnoses.

The same approach was applied for the laboratory event using ***LOINC_CODE*** as a key. A total of 165 ***LOINC_CODE*** was eliminated from a total of 702, representing 0.2350 % deletion. In the prescribed event, the drugs that appeared less than 20 times (1952 out of a total of 3028), were eliminated, representing a 0.6446 % of the deletion.

The event laboratory has an important point to explain. Extracting the information from the Physionet's description "***VALUE*** *represents the value measured for the concept identified by the* ***ITEMID****. If this value is numeric, then* ***VALUENUM*** *contains the same data in a numeric format. If this data is not numeric,* ***VALUENUM*** *is null. In some cases,* ***VALUENUM*** *contains the score and* ***VALUE*** *contains the score and text describing the meaning of the score. is the unit of measurement for* ***the VALUE****, if appropriate.* ***FLAG*** *indicates whether the laboratory value is considered abnormal or not, using pre-defined thresholds* "

In the data quality phase, we wanted to check if the pre-defined thresholds were working correctly and check if the FLAG provided a match our flag. The following steps were carried out:

	LABEL	FLUID	Min value Men	Max value Men	Min value Women	Max value Women	LOINC_CODE	VALUEUOM
0	Potassium	Blood	3.5	5.3	3.5	5.3	2823-3	mEq/L
1	Ammonia	Blood	11.0	32.0	11.0	32.0	16362-6	umol/L
2	Amylase, Pleural	Pleural	30.0	110.0	30.0	110.0	1796-2	IU/L
3	% Hemoglobin A1c	Blood	0.0	5.7	0.0	5.7	4548-4	%
4	Urea Nitrogen	Blood	8.0	20.0	8.0	20.0	3094-0	mg/dL

Figure 4.8: Data Quality Table Example

- Take a sample of 10 % of patients. $\tilde{1}042$
- Extract the differents **LOINC_CODE** presented in the patients.
- Create a table with min and max range of values of each **LOINC_CODE** depending the gender. The values were added by standard input.
- Check each values and compared with the original one(abnormal or null).

The match was 98 % successful, with the sample size, so as a conclusion, the data provided was correct. The 2 % can be related to false negatives for a typo insertion of the values. (The insertion of the values was done manually).

4.3 Data Exploration

There is the information of patients in Figure 4.9. A new feature called **AGE_DEATH** was added, calculating the subtraction of **DOD** (Date of Death) and **DOB** (Date of Birth). The subtraction for some of the patients were 300 years, for the anonymity process done to the original database. The real data was actually replaced for 79 years old.

DOD_HOSP	DOB	GENDER	SUBJECT_ID	DOD	DOD_SSN	AGE_DEATH	AGE_INTERVAL
2201-08-02 00:00:00	2135-03-22 00:00:00	M	357	2201-08-02 00:00:00	2201-08-02 00:00:00	66	#4 45-78
2182-07-31 00:00:00	2121-10-20 00:00:00	M	669	2182-07-31 00:00:00	2182-07-31 00:00:00	61	#4 45-78
2124-05-02 00:00:00	2044-09-28 00:00:00	M	414	2124-05-02 00:00:00	2124-05-02 00:00:00	80	#5 78+
2166-07-01 00:00:00	2095-12-27 00:00:00	M	771	2166-07-01 00:00:00	2166-07-01 00:00:00	71	#4 45-78
2106-09-29 00:00:00	2052-07-29 00:00:00	M	234	2106-09-29 00:00:00	2106-09-29 00:00:00	54	#4 45-78

Figure 4.9: Patient table

The Patient's information was empty in DOD feature, if the patients were still alive. Of 10442 patients, the 0.4860 were alive and the 0.513913 were dead respectively.

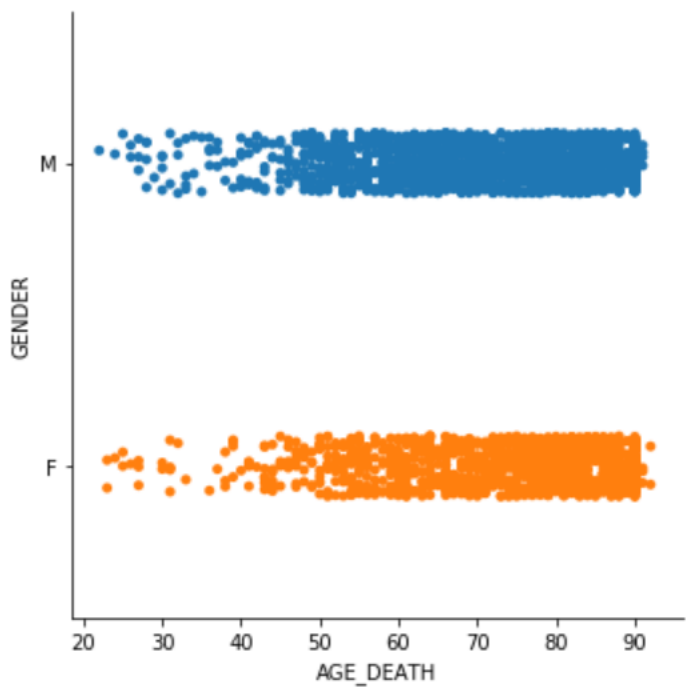

Figure 4.10: Gender vs age of death

Of the deceased patients (a total of 5066), 0.5592 were men and 0.440702 were women.

Although the percentage of men were greater, the difference is not as significant as in the studies that have been done before in mortality according to sex [34]. Figure 4.10 shows the relationship between age of death and sex. Both genders followed the same trend. The age where most people die was between 77 and 90 years.

Regarding the admission event, these events contain information when a patient entered into a hospital. This event contains demographic information about the patients and the name of the general diagnosis. The diagnosis is different from the event diagnosis that contains the specific code. For purposes of our work, **ICD9_CODE** was the one used because contains more granularity.

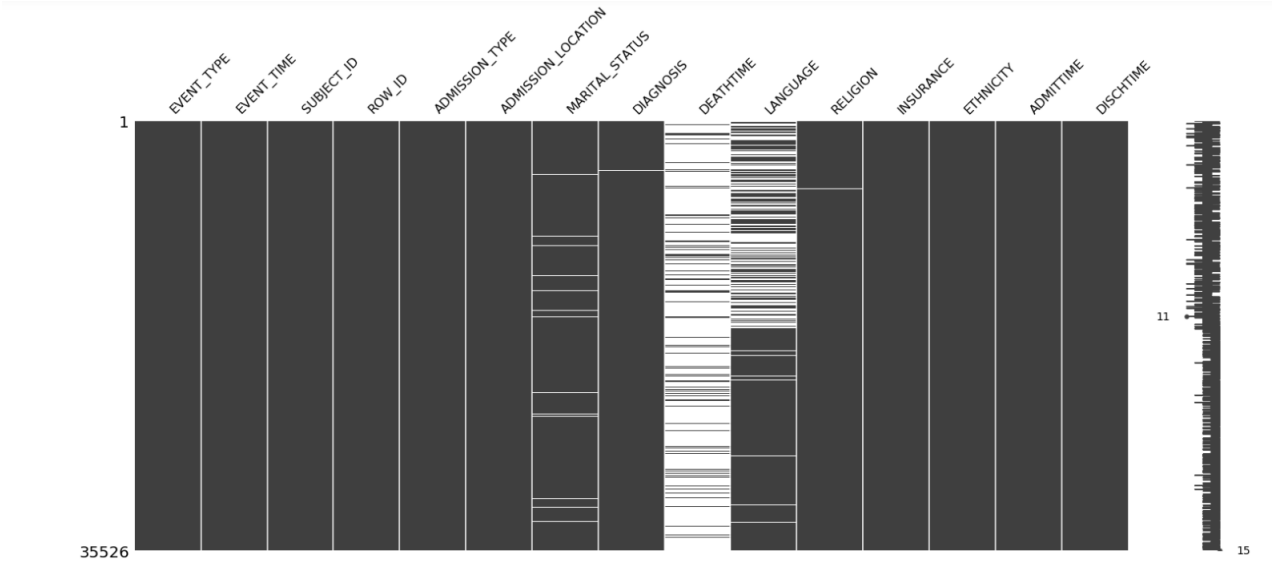

Figure 4.11: Missing Values Admission Event

In the Figure 4.11 0.8637 % of the **DEATHTIME** is missing. This means the patients are still alive. This information differs from the patient event because patients can have multiple admissions. The 0.4444 % of rows were missed in the language feature, but this feature did not affect our algorithm.

Another interesting topic encountered in the data are related to diagnosis. In Figure 4.12 it is the top five diagnoses in women. The one that appeared most was Sepsis with a 0.0210 % of the total diagnosis, followed by Pneumonia with 0.018805 % of the total. The same info about men can be found in Figure 4.14, where the most common diagnosis

is Pneumonia(0.032633 %) followed by Sepsis(0.030789 %). It is interesting to see that in the fifth place, we find the altered mental status.

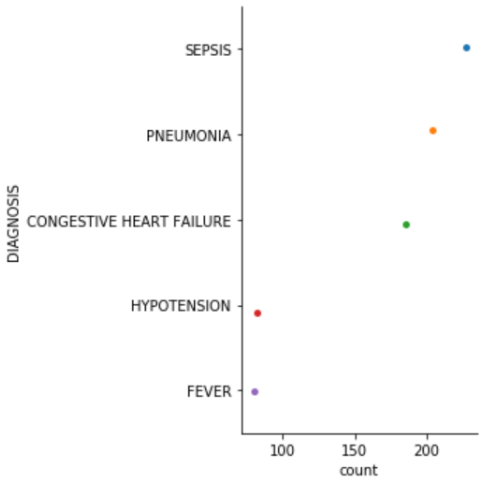

Figure 4.12: TOP 5 Diagnoses in Admission Event in Women

	GENDER	DIAGNOSIS	count
887	F	SEPSIS	228
734	F	PNEUMONIA	204
283	F	CONGESTIVE HEART FAILURE	186
501	F	HYPOTENSION	82
391	F	FEVER	80

Figure 4.13: TOP 5 Diagnoses in Admission Event in Women data

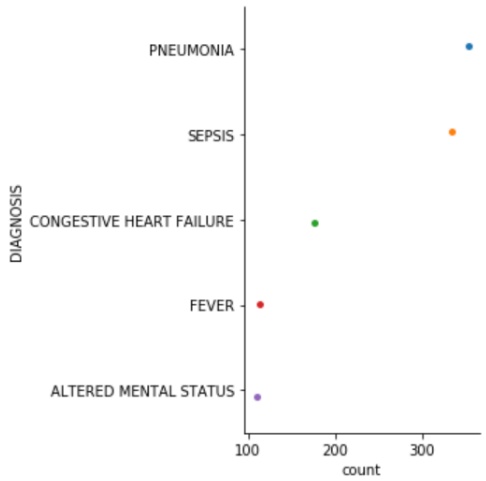

Figure 4.14: TOP 5 Diagnoses in Admission Event in Men

	GENDER	DIAGNOSIS	count
1007	M	PNEUMONIA	354
1242	M	SEPSIS	334
345	M	CONGESTIVE HEART FAILURE	176
517	M	FEVER	114
84	M	ALTERED MENTAL STATUS	110

Figure 4.15: TOP 5 Diagnoses in Admission Event in Men table.

In Figure 4.16, there is the top seven Diagnoses in Admission Event vs. Age of Death. This does not mean that people die from one of these seven reasons, but the people that died suffer from one of the diagnosis in some point of their life. The first three of the top appear between 70-90 years old. No information about this top seven diagnoses was found in patients less than 50 years old.

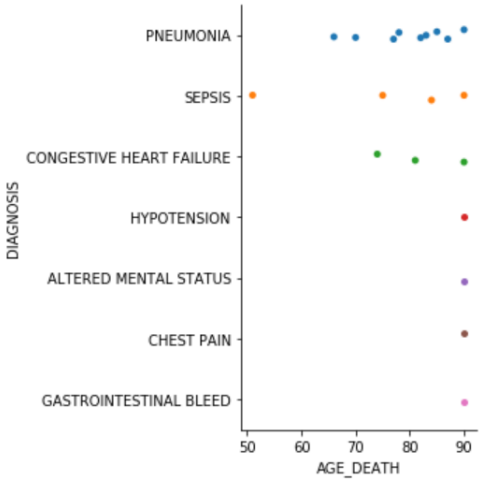

Figure 4.16: TOP 7 Diagnoses in Admission Event vs Age of death.

Chapter 5

Approach proposal

5.1 Extended Definition Problem

One of the most important parts of the realization of this thesis was to work with temporal data mining, specifically with time interval mining. As mentioned in the background, time-oriented data can be measured at different granularity and frequencies; this is useful for aggregated, time-stamped, and raw data for abstract concepts. Using the correct techniques, the discovery of temporal patterns it is possible. The approach consists in the implementation of KarmaLego Algorithm proposed by Robert Moskovitch [18] [22], which is a fast algorithm to discover time interval related patterns, and extends temporal patterns directly, working with transitivity for better performance in candidate generation. This algorithm works with large datasets and low minimal support. The phases of the development of the solution can be found in Figure 5.1:

5.2 Data Temporal Abstraction

5.2.1 Problem Definition

Let's call ***P*** to a patient who has a medical record ***MR***. ***MR*** has a series of events ***E*** that are of type ***EA, EB, EC, ED, EE, EF***, which occur over time ***T*** which has ***startdate*** and ***enddate***. For each event from the patients, the time data points are transformed

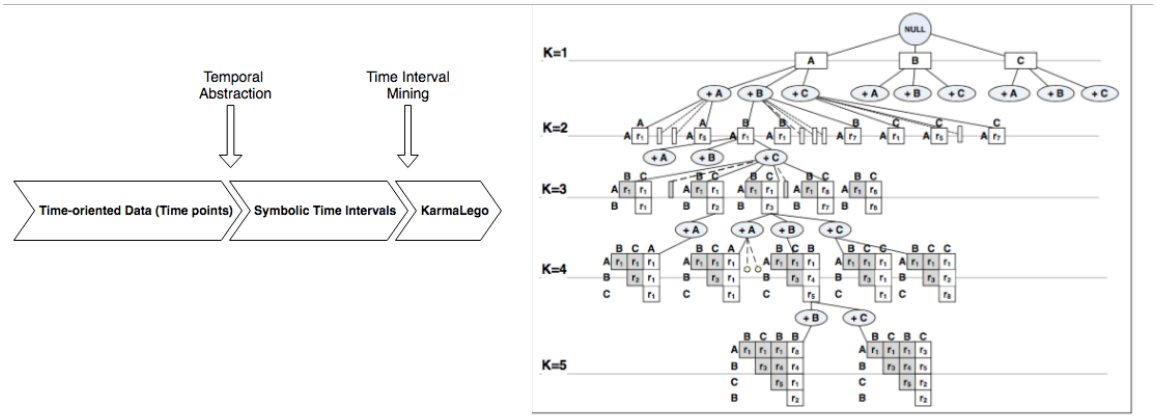

Figure 5.1: Phases of Development for the Approach Solution

into time Intervals. Each E passes through the temporal abstraction process explain in the background chapter, relate a time series with an abstract symbol.

5.2.2 Phases of transformation

Having $e \in E, E \in P$ and e has a time value, the event e it is associated with a abstract symbols and a ***startdate*** and ***enddate***.

- For all $e \in E$, e is of type $\{Procedure, Prescription\}$, e has a ***startdate***,***enddate*** and a ***abstract symbol***. These events already contain a predefined interval with a ***startdate*** and ***enddate*** in the raw data, for this reason this kind of events works with ***the State Abstraction level***.So the time series are transformed into a time interval series, defined as $I =< e.S, e.E, e.sym >$.

  Example: "Maria went to the hospital for a headache. The doctor give her "Ibuprofeno" 250 G StartDate= 02/03/2016 EndDate= 02/05/2016."

  $$\text{I= } < 02/03/2016, 02/05/2016, \text{Ibuprofeno 250 G} >$$

- For all $e, e_1 \in E$, e and e1 has a time value, a symbol, but does not have a ***startdate*** and ***enddate*** predefined. The events are of type $\{Diagnose, Laboratory, Microbiology\}$ and e_time and $e_{1.time}$ are contiguous with the same symbol. In this case, the creation of the interval must be done using ***the gradient definition*** of temporal abstrac-

tion. This means that if a time series has the same symbol contiguous, the times are merged into a time interval with the corresponding ***startdate*** and ***enddate***.

Example:

- *"Maria went to the hospital for a blood test 02/03/2016 14:00. The leukocytes were abnormal"*
- *"Maria went to the hospital again for a second blood text 03/03/2016 14:00. The leukocytes were abnormal"*
- *"Maria went to the hospital for the third time for a third blood test on 04/03/2016 14:00. The leukocytes were normal"*
- *"Maria went to the hospital for the first time for a fourth blood test an 05/03/2016 14:00. The leukocytes were normal too"*

In the first two time-series points, the leukocytes were abnormal, so between 03/03/2016 14:00 and 4/03/2016 14:00 the "leukocytes were abnormal" and between 04/03/2016 and 05/03/2016 they were normal. It is assumed that the abstract domain information was done by an expert. Example: Hemoglobin comes with a flag that said is abnormal. Urine comes with a flag that say is yellow.

- Once each event of a patient is processed and transformed into a time interval, the time interval series for a patient looks like $I =< I1, I2, I3, I4 >$
- The last step is transforming this time interval series into a Lexicographic time interval series, as explained in the background. This part was done using Mergesort, following the definition given in the background chapter with the framework of Allens temporal relation, with epsilon =0, using only the relations of "contains,", "before," and "overlaps". The explanation can be found in Figure 2.1.

5.3 KarmaLego Algorithm

The algorithm KarmaLego was proposed for Robert Moskovitch and Yuval Shahar [22] and divide the problem into two phases. In the first phase,are found the most frequent

two-sized items TIRP, or (I_1, I_2) where each item is a symbolic time interval ordered lexicographic, and which each edge is connected with one of the three Allen temporal relations as before, contains, or overlap. For this step, it performs a BFS. The second phase of the algorithm calls Lego and is a recursive process of encountering the k-TIRPS extending the 2-sized searched in Karma, using a candidate generation with Transitivity approach. This second approach apply a DFS procedure.

Algorithm 1 KarmaLego

Require: database | P | and all the $I = \{I_1, I_2, I_n, \dots\}$ for each p ∈ P.
Require: ms - minimal support = the minimal support.
DAGt =Karma(db,ms)
1: **for all** $t \in DAGt$ **do**
2: Lego(DAGt,t,ms)
3: **end for**
4: **return** DAGtTIRPS

The KarmaLego algorithm 1 receives for each patient P, the intervals series in lexicographic order and the minimal support. So the first step is execute Karma for calculate the 2-sized TIRPS using a BFS approach, and this will return a DAG represented as a Double HashMap. The line 1 is a loop for each $TIRP$ t contain in DAGt, that will be extended with Lego.

5.3.1 Karma Algorithm

At the begging of the algorithm 2, when the temporal abstraction is done, the supporting count of each symbols is calculated, and only the symbols that are above the minimal support are taking in account. For that reason, at line 1, these symbols are already calculated. The interesting part is that the cost of calculating this first phase comes along the temporal abstraction phase (lets say the first pass for the database).

In lines 3-6, all the patients have their I as interval ordered in lexicographic, then the creation of each TIRP of 2-sized is stored in a HashMap that represents the DAG where the Symbol A goes to the Symbol B, depending of the relationship that they have(before, contains or overlap). This process is done as:

Algorithm 2 Karma

Require: database | P | and all the $I = \{I_1, I_2, I_n, \dots\}$ for each p ∈ P.
Require: ms - minimal support = the minimal support.
1: SymbolsP = The symbols above the minimal support.
2: DAGt is empty
3: **for all** $p \in P$ **do**
4: **for all** $I \in IS, IS \in p \mid I_{i.syml}, I_{j.sym}$ and i < j and $sym \in SymbolsP \mid$ **do**
5: r = AllenTemporalRelation(I_i, I_j)
6: HashMap(DAGt,< $I_{i.symbol}, r, I_{j.symbol}$ >,p_{ID},i,j)
7: **end for**
8: **end for**
9: **for all** $t \in DAGt$ **do**
10: **if** support(t) < ms **then**
11: Prune(t)
12: **end if**
13: **end for**
14: **return** DAGt

$R = \bigcap_{i=1}^{k} \bigcap_{j=1+i}^{k} r(I_i, I_j) =$

$= \{r_{1,2}(I_1, I_2), r_{1,3}(I_1, I_3) \dots r_{1,k}(I_1, I_k), r_{2,3}(I_2, I_3), r_{2,4}(I_2, I_4) \dots r_{2,k}(I_1, I_k), r_{k-1,k}(I_k - 1, I_k)$

In the double Hashmap it is stored the symbols as keys and not the time interval. It is important to mention that the ID of the patients and the index of the position of the symbol of the two Intervals are stored. This means that (i, j),it the symbol $I_{i.sym}$ happens in the position i the symbol of $I_{j.sym}$ happens in position j. The reason is explained in the supporting count phase (Projected Database) in this chapter. It is important to mention that the search for the existence of a 2-sized TIRP is O (1), thanks to the structure used. Once the DAG is built with the TIRPS of size two, all the relationships between nodes whose support is below that given by the user, are eliminated. For reasons of efficiency it is possible not to eliminate them, just ignore them when executing the second part of the algorithm. Also, the fact of not eliminating them will represent an important part for the future of the design of the algorithm incrementally.

5.3.2 Karma Algorithm Vertical Support

Let $N \geq 1$ be the number of patients. Let $1 \leq n \leq N$, and let t be a *TIRP*. Then, we define $\varphi = n/N$ as the minimal support. Let us consider the following function:

$$y(t,i) = \begin{cases} 1 & \text{if the patient } i \text{ has never experienced } t \\ 0 & \text{otherwise} \end{cases}$$

Then, we say a *TIRP* is frequent when

$$\frac{1}{N}\sum_{i=1}^{N} y(t,i) \geq \varphi \tag{5.1}$$

This function explains that a pattern is frequent, if its vertical support must be equal to or greater than the minimum support given. The pattern must appear at least once in a patient, and this must appear in a significant **n** number, formed between **n / N** as the one given as minimal support. This solution was based on the premise that is important to know how many different patients satisfy the pattern vs. a few patients meeting the same pattern many times (Horizontal Relax support). A vertical support is between 0 and 1.

5.3.3 Karma Algorithm Horizontal Support Relaxed

This configuration is done focus in how frequent is a pattern globally. The Horizontal support is a integer.

5.3.4 Lego Algorithm

Algorithm 3 Lego

Require: DAGt with 2-sized TIRP
Require: A TIRP to be extended
Require: min support
Require: SymbolsP

```
1: for all s ∈ SymbolsP do
2:   for all r ∈ R do
3:     tirp_new = CreateATirp(t.size + 1)
4:     AddSymbolToTheEnd(tirp_new, s
5:     AddRelationOfNewSymbol(tirp_new, r)
6:     Candidate = ∅
7:     Candidate = Generate_Candidate_Transitivy(tirp_new, 1)
8:     for all c ∈ C do
9:       if SupportingCounts(c, tirp_new) > min_support then
10:        tirp_new is frequent
11:        Lego(DAGt, c, min_support)
12:      end if
13:    end for
14:  end for
15: end for
```

The second phase of the algorithm 3 called Lego is basically a recursive algorithm to extend a

TIRP. The extension proposed by Hopner with the H-DFS method using a data structure similar to the HashMap, where the symbols are stored. In this method, candidate generation is created each time, extending the TIRP with the symbols found in SymbolP (the ones that are frequent) with the relations used. In the first step, in line 2-5, all the frequent symbols are generated after the last time interval t using one of the relations in R. An example of this extension can be found in Figure 5.2. The extension phase is conformed for a candidate generation using transitivity and perform a support counting of each TIRPS encountered.

5.3.5 Candidate Generation using Transitivity

In line 3 of algorithm Lego, an extended candidate TIRP $tirp_{new}$ is created based on TIRP t, a symbolic time interval s and a temporal relation r. It is a recursive process that exploits the transitivity property of temporal relations. An input received the extended TIRP with the new r to index at the end. It given only the last relation between the two last symbols, now the relation between the new symbol is added and the rest of the symbols should be calculated using the transition table and the actual disjunction of temporal relations.

For example, in Figure 5.2, the TIRP of sized 4 want to be extended to size 5, so the $A\hat{2}$ is the new symbol to add at the end of the TIRP. It needs to check if there is a relationship between the last symbol of the Interval series (C), and the new TIRP to be added (A^2). There are three options, it can be before, contains or overlaps. Using the structured approach developed in the thesis, an optimization is made making possible to know exactly which relationship exist. Before sending this to the Supporting Counting algorithm, the "?" missing information must be "fill". For this reason, the algorithm generate them using a transitivity table proposed [**?**]. This part of the algorithm is done recursive, indexing each row of the Candidate. Once all the candidates are generated, the support counting is done. If the support is above that the minimal support, this TIRP is frequent and pass to Lego for extension. In the proposal solution, the number of the size of the TIRP is given by the user, in other case the recursive process can last infinite.

It is possible to use brute force and prove all the possible combinations with the tree Allens temporal relations. With brute force, 3^3 different TIRPS are generated; for this reason, we worked using transitivity and reduced the problem to two possible candidates in the example.

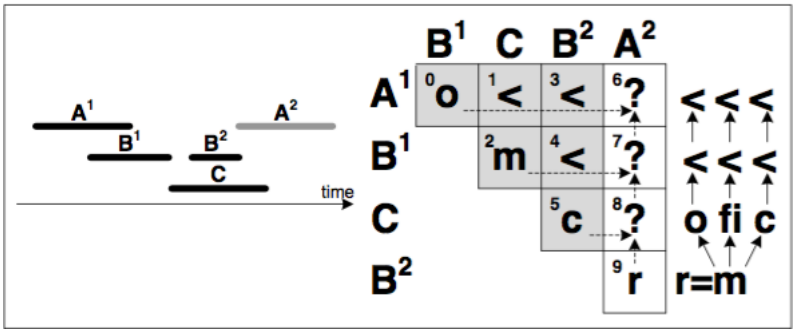

Figure 5.2: TIRP size 4 to be extend with A^2

5.3.6 Supporting Count Vertical Support

To count the frequency of a specific TIRP using the vertical support concept,it is mandatory to store the ID of the patient and the index of the position of the Symbols in pairs in our HashMap(Projected Database). For a TIRP of with $K >= 3$ be able to be extended, it must check,that each pair of combination exist in HashMap with a vertical support above the minimal given by the user. For example if TIRP A—B is going to be extend with the symbol "C", it should check the existence of A—B, B—C and A—C, so for this reason the ID of the patient and the position of their appearance of the symbol in their interval series of the corresponding patient is stored. As the Intervals of a patient is lexicographic ordered, knowing the positions of the symbols, it is possible to count their appearance. In this case, it only calculate once per patient, and ignore if the same TIRP appeared more than one in specific patient. This phase is incremental for the rest of the TIRPS. This approach is in the case of interest of the study "How many different patients satisfied the pattern ?".

5.3.7 Supporting Count Horizontal Support Fixed

This approach uses the same Index position of the symbols stored with the ID of the patients. In this case, the frequency per pattern, per patients take part in the final calculation. This means that a pattern occurs three times in a patient, this three times counts for the final calculations of the horizontal support that must be above the minimal support. This gives us information about how frequently is a pattern globally.

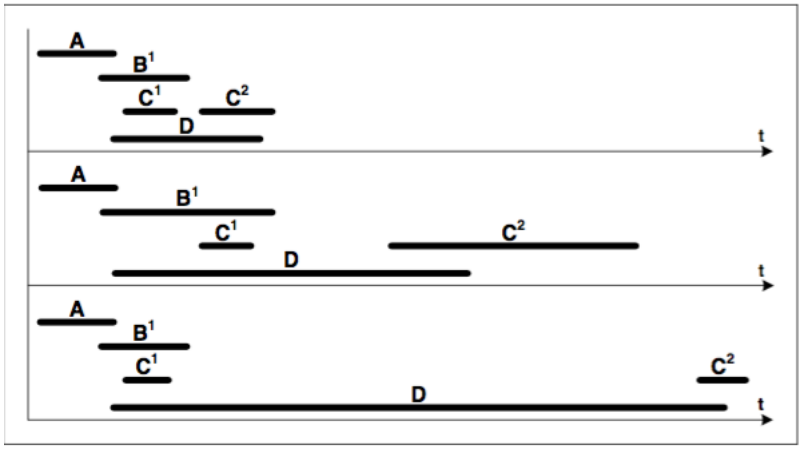

Figure 5.3: Different Time Interval of the same TIRP

5.3.8 Adding the duration of the Time Interval into the Abstracts Symbols

Adding the duration of the Interval into the abstract symbol as a solution for having instances where the importance of the duration makes sense for a doctor. The problem with this is that some TIRPS that are actually frequent can be lost if their duration are not frequent. An example can be found in the Figure 5.3. Three different instances of the same TIRP definition. For doctors, such as for these quantitatively different instances of the same TIRP might represent quite qualitatively different scenarios.

Chapter 6

Experiments and results

The experiments performed were divided into two studies. The first study is related to the performance of the algorithm and the second was related to the shape of the TIRPS. The first study consisted in the execution of the algorithm in medium groups of patients (1000,2000,3000) using input variables such as epsilon *(The Epsilon parameter to Allen's temporal relations maintain the Jointly Exhaustive and Pairwise Disjoint (JEPD) conditions that refer to the probability theory and explain that "two sets A and B are disjoint if their intersection is the empty set". This maintains the relation A and B mutually exclusive.)*, pattern length (how long the pattern can be), number of patients, minimal support, to obtain information about memory (memory requested for the case), run time and number of patterns (TIPS). Each of these experiments were applied with three configurations: Vertical Support, Vertical Duration Interval Support and Horizontal Support. The first case was using vertical support with different combinations of pattern length, minimal support and number of patients. The second case was using the horizontal support relaxed. The third case was the combination of vertical support with the addition of the duration of the interval in the abstract symbols.

6.0.1 Performance of KarmaLego

During the evaluation phase of the algorithm, the epsilon variable was always used as 0. The length of the patterns were tested with 4,5 and 6 with minimal support configurations of 0.1,0.2 and 0.3. For the case of horizontal support (the minimal support is an integer between 0 and infinite), it was tested with 100, 200,300,600 and 900. It was also found that the mean of the length of the time interval series per patients was 543.

Let's call "Vertical" the configuration of vertical support, "Vertical Duration Interval" the configuration where the concept of vertical support was used adding the duration of the interval as an abstract symbol. Finally, "Horizontal" to the configuration of using Horizontal support relaxed. As explained previously, it is possible to choose the pattern length and the algorithm recursively add a symbol using a Depth Breath First (DFS). It is also important to remember that when a pattern is frequent, all instances covered by that pattern as sub-patterns are also frequent too, as part of the mining process.

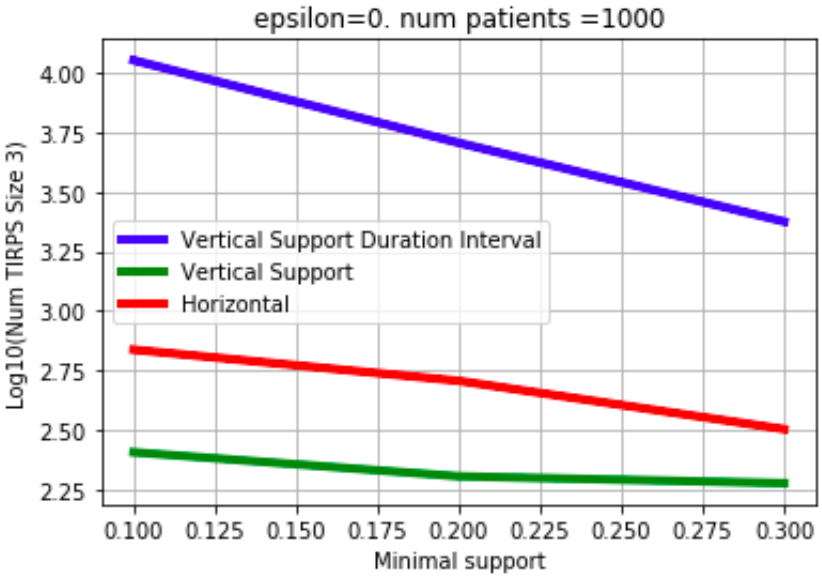

Figure 6.1: TIRPS3 VS Minimal support

The graphic in Figure 6.1 shows the comparison using 1000 patients with a minimal support of 0.1, 0.2 and 0.3 respectively. It is important to mention that the Horizontal support received 100 patients as input (it was added here to see the comparison, but this configuration works different than the other ones). The graphic is in logarithmic scale. This figure shows comparison between the number of TIRPS of size 3 generated vs minimal support. The behaviour found was that the lower the minimal support is, more TIRPS are obtained. The vertical support with duration interval was the one that generated more TIRPS with respect to the other two. Comparing vertical support and vertical support duration interval, it was expected that this one would generate more TIRPS, because you can find different patterns with different duration intervals definition, but was not expected that this one generated more than the horizontal one.

The comparison between the number of TIRPS of size 3 and number of TIRPS of size 4 VS the

number of patients in vertical support and the vertical duration interval, can be found in Figure 6.2 and Figure 6.3 respectively. In both cases while the number of patients grew, the number of TIRPS also grew. This behaviour explains that some patterns can reach the indicated threshold (minimal support) by adding more patients. An interesting event, is that in the case of TIRPS of size 4 (which occurs based on the TIRPS of size 3), at one point the amount of these ones created by vertical support exceeds the ones of interval duration. This scenario, is due to the pruning processes that in the interval duration failed by little.

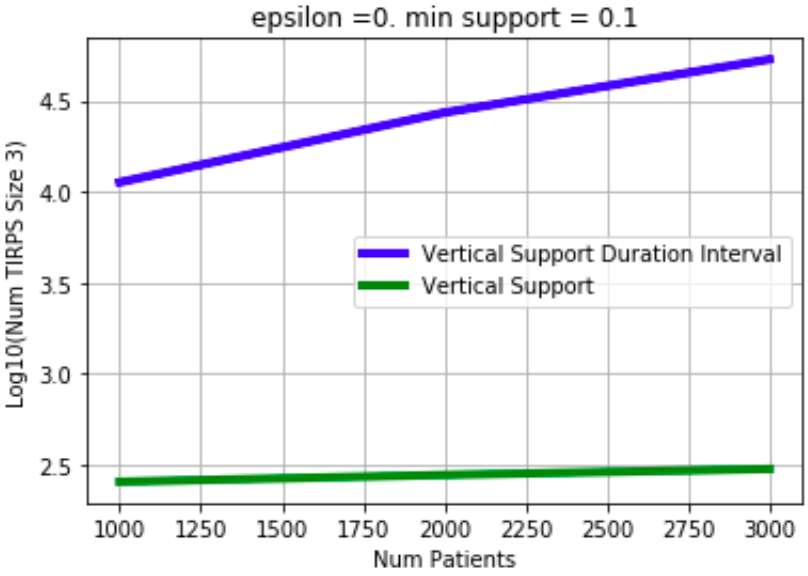

Figure 6.2: TIRPS3 VS Number of Patients

The case of horizontal support deserves to be treated separately since the nature of its concept is different. The horizontal support consists in knowing the number of frequent patterns globally (a pattern can occur n times in a patient and this number counts to reach the total threshold). Using a minimal support of 100,200,600,900 in 1000 and 2000 patients, the differences between the number of TIRPS 3 and TIRPS 4, in 1000 and 2000 patients were not much. By the other hand, it showed the expected behaviors that increasing the number of patients, many patterns are reached the minimal. This comparison can be found in Figure 6.4 and in the Figure 6.5.

Respect to the vertical support, in the Figure 6.6 there is a comparison between the number of TIRPS of different sizes and the number of patients. As expected as the number of patients increased, the number of TIRPS increased respectively in each case. As long the length of the

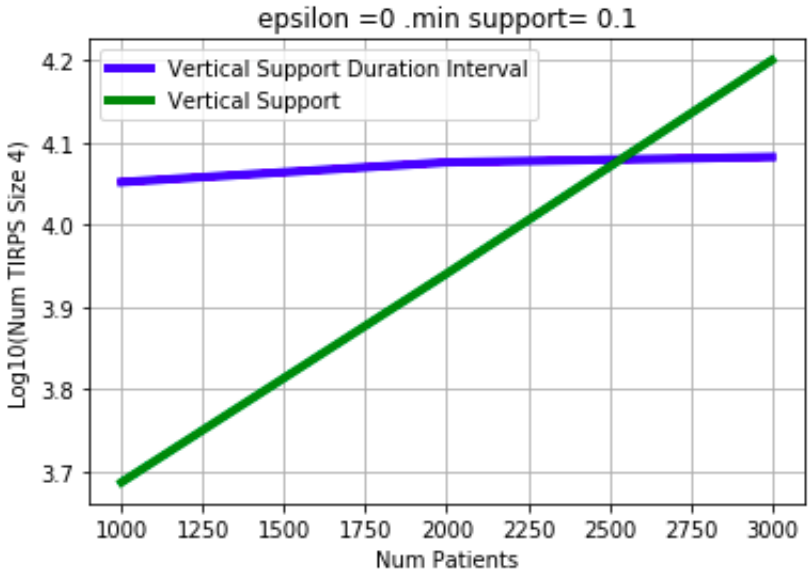

Figure 6.3: TIRPS4 VS Number of Patients

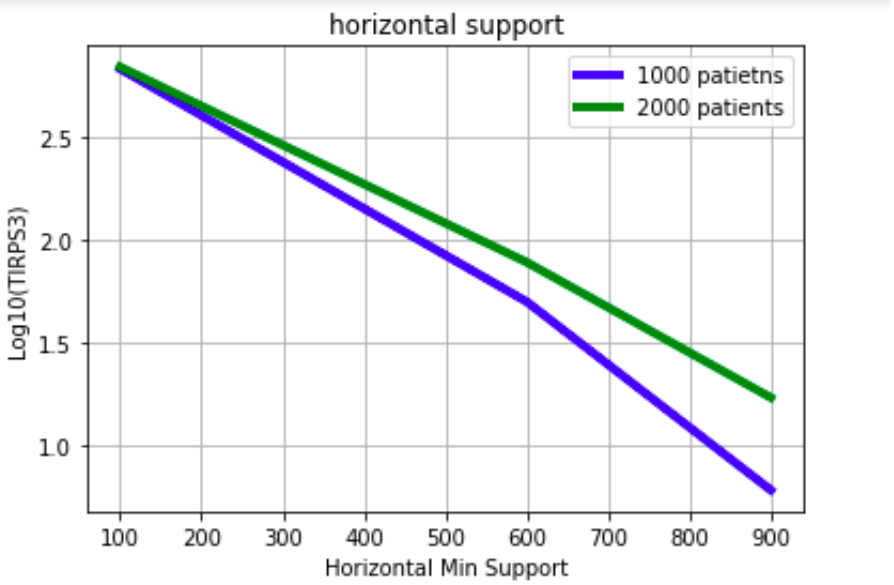

Figure 6.4: TIRPS3 VS Horizontal Minimal support

size grows, the number of TIRPS also grows, until it reached a size where this stopped. During the tests, a case of length nine was executed with a 0.3 of minimal support for vertical support in 1000 patients, and for a length of size eigth did not find any frequent pattern. There is a point where the growth of the TIRPS is no longer proportional to the length.

Regarding the performance of the algorithm, a minimal support of 0.1 with and number of patients of 1000, 2000 and 3000 were tested. The idea was to stress the algorithm with low

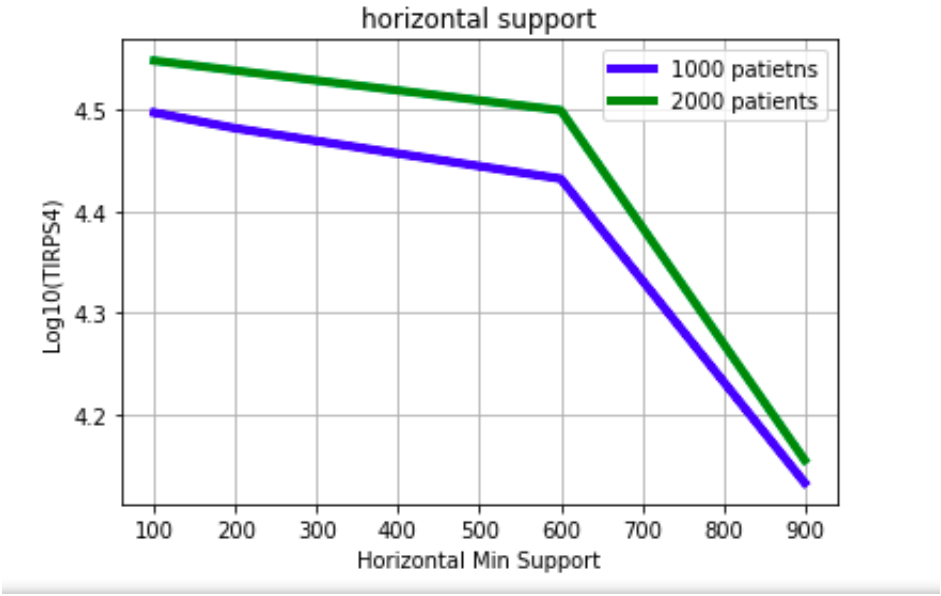

Figure 6.5: TIRPS4 VS Horizontal Minimal support

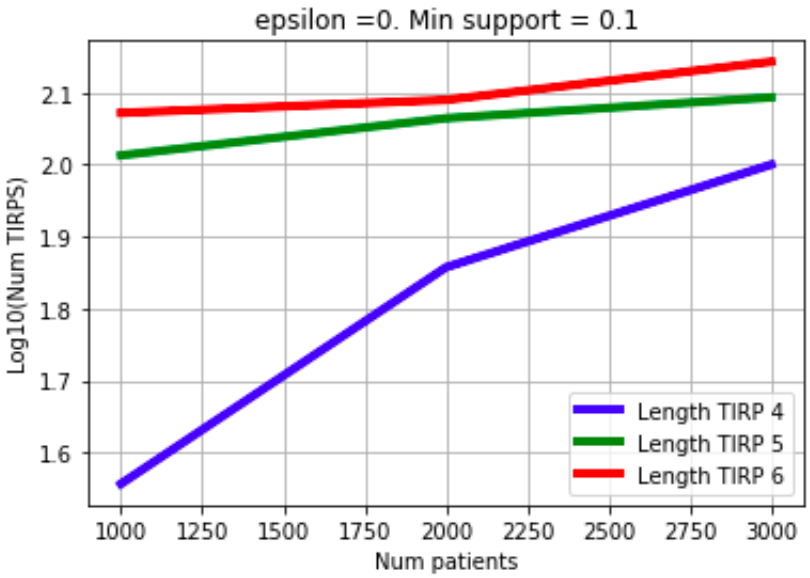

Figure 6.6: Number of Patients Vs Number of TIRPS

minimal support. One of the disadvantages of the algorithm was the amount of hours each case took. Only 3000 patients lasted for days and the amount of memory was also very big. The pre-processing phase taked approximately 20 minutes in small cases. It is inefficient with respect to memory and is quite slow. Other cases were executed, such as 8000 patients with 0.3 for more analysis of another results, as well as biasing the algorithm so that it only generated nodes of diagnoses (this last case, was done only including the events of interest, finding as a result that with really small minimal support, very few TIRPS were found). In Figure 6.7 and

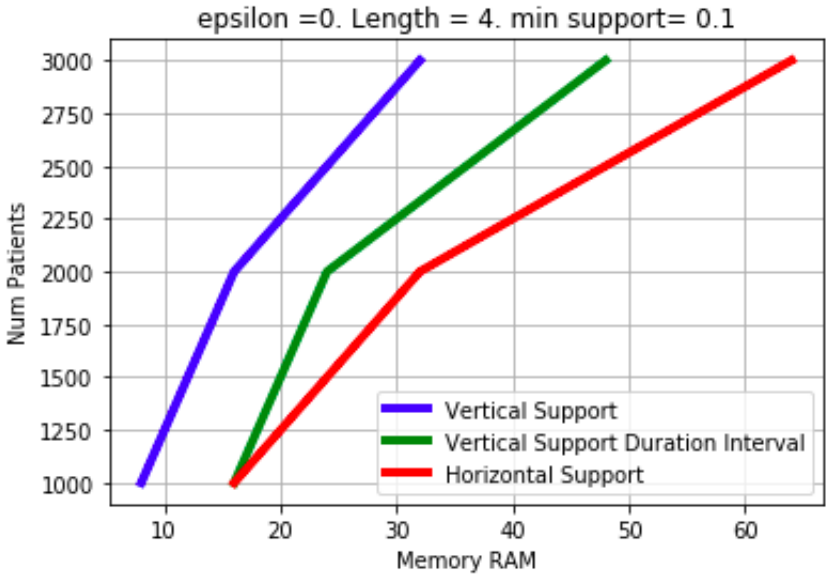

Figure 6.7: Number of Patients Vs RAM

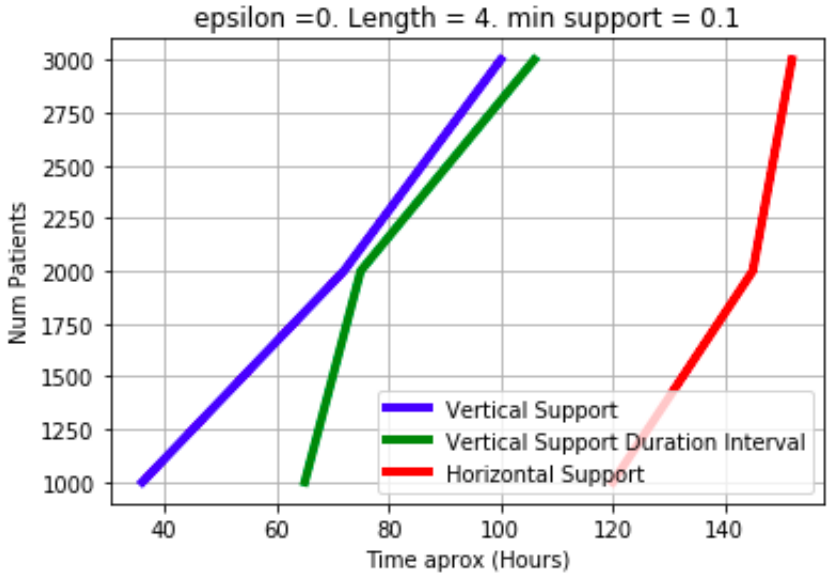

Figure 6.8: Number of patients VS Time (hours)

6.8 there is a comparison of the algorithm in performance. As expected with respect to RAM, Horizontal support consumed much more for the high number of patterns that passed this type of threshold in the configuration. Then, the Vertical Duration support is still greater than the vertical support. This behavior was expected because in Horizontal Support more TIRPS are generated globally. Then, vertical Duration Interval generates more TIRPS than the vertical support configuration. In some point, the vertical support duration is cut for pruning process

different from the horizontal configuration which the pruning process is limited.

Respecting time performance, Horizontal Support was the one that took more time. Between vertical support and vertical support Duration interval, the second one takes more or less the same time. However as the beginning vertical duration interval support generated more TIRPS than the vertical support, this behavior exchanged in some point as an effect of pruning process and TIRPS behaviour.

6.0.2 TIRPs, shape and behavior

A TIRP is a structure that has frequent pattern information. It can be seen as a DAG. Each node of the graph is linked to another by an Allen temporal relation: Before, contains or overlap. Also, a TIRP can have different length, in which every sub pattern is also frequent. The nodes or events can be of type procedure, prescription, diagnose, microbiology or laboratory. In the Figure 6.9 shows how a TIRP can be seen as a DAG, and in the 5.2 shows the TIRP as a structure.

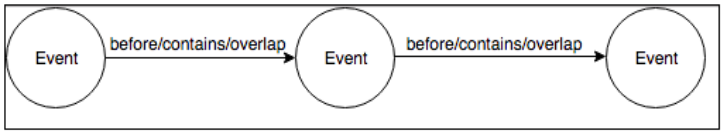

Figure 6.9: Example of a shape of a TIRP

At the first part of this chapter, the performance of KarmaLego was studied. In this second part, the shape and behaviour of a TIRP was analyzed. For each case that was executed, an average of the number of events that appeared in each TIRP was studied as a distribution, and was applied for every configuration. The study of the shape of the TIRP is important for a quality process that helps in decision making to improve pre-processing phase.

The Figure 6.10 shows how the TIRP is distributed using vertical support. This was executed in 1000 patients with a minimal support of 0.1. It is interesting to see that the laboratory event represents 75% in size 3 and is increasing the percentage in each size. This explains that when TIRP is extended, the symbol that belongs to the aggregate interval is a laboratory event and is frequent. Diagnosis and prescription in turn, decrease by % compared to laboratory.

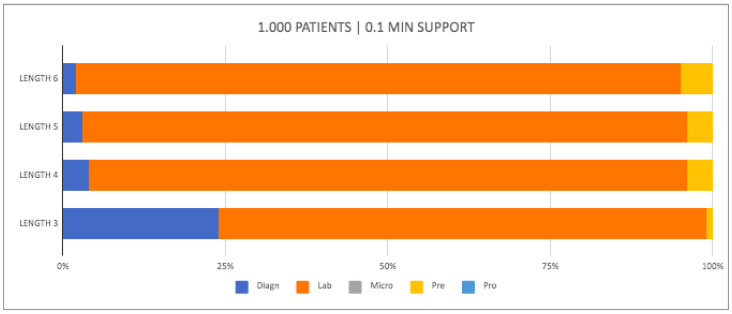

Figure 6.10: Distribution of a TIRP Vertical Support 1

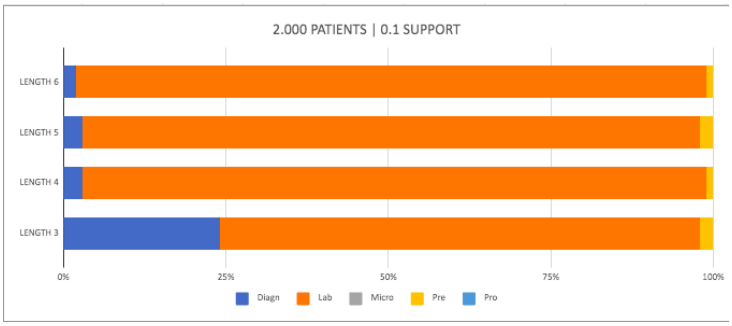

Figure 6.11: Distribution of a TIRP Vertical Support 2

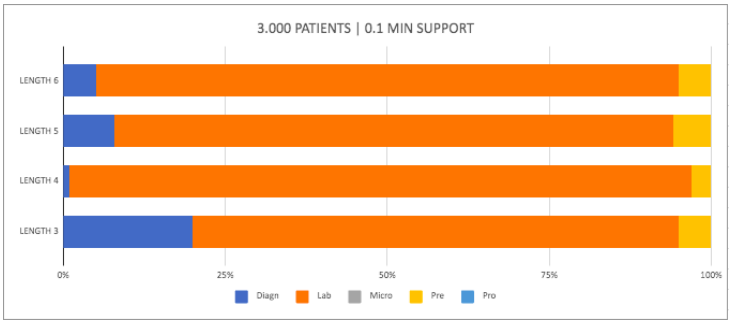

Figure 6.12: Distribution of a TIRP Vertical Support 3

A very similar scenario can be found in the Figure 6.11 and in the Figure 6.12. Once again, the laboratory event is represented almost all the events and in Figure 6.12, the percentage of prescription events increased. In all three cases, the distribution between length is usually not much. The same test was done with different minimal support, but the results were almost the same.

Studying the case of horizontal support, using 100 as a minimal support, a percentage of 2 % microbiology events appeared. Taking into account the nature of this configuration, it was expected that these events appeared. The same happened with procedure events with a 1 % in

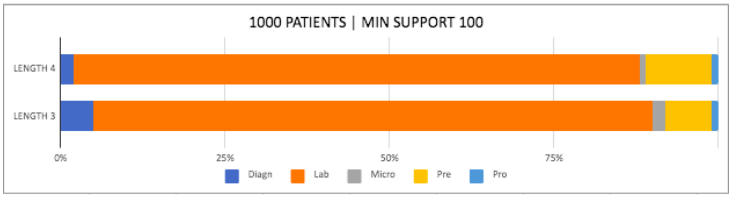

Figure 6.13: Distribution of a TIRP Horizontal Support

the distribution of the TIRPS.

The same approach was applied in Vertical Duration Interval support in 1000 patients and minimal support of 0.1,0.2 and 0.3 respectively. The interesting part found was that in Figure 6.14 in length 3, the 50 % of the TIRPS shape was prescription event. This behavior was expected taking in account that it is normal to found different duration interval for each prescription. An example of this behavior it could be "two days taking ibuprofeno" vs "three days taking ibuprofeno". These two scenarios represent two different events.

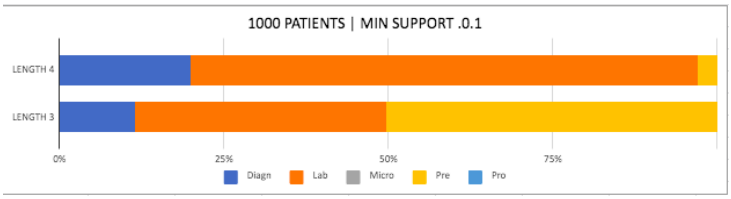

Figure 6.14: Distribution with Vertical Duration Interval Support 1

It is important to add that for example the events of type prescriptions, can differ only in the amount of prescribed dose, increasing the number of prescription-type related to the same medication in any configuration. An example of this situation was found in Figure 6.15. Each row represents a frequent pair and the last columns contain the order. The pairs are stored as: (A,B) and (B,C) for represents: A—B—C.

1 Potassium Chloride : 40 mEq. 10mEq ER Capsule	prescription	1 Potassium Chloride : 40 mEq. 10mEq ER Capsule	prescription	BEFORE	1
1 Potassium Chloride : 40 mEq. 10mEq ER Capsule	prescription	1 Potassium Chloride : 40 mEq. 20mEq Packet	prescription	BEFORE	2
1 Potassium Chloride : 40 mEq. 20mEq Packet	prescription	1 Insulin : 0 UNIT. Dummy Package for Sliding Scale	prescription	BEFORE	3

Figure 6.15: TIRP

In general, the TIRPS in each configuration followed a same behaviour. A lot of event of type laboratory was found, followed by diagnosis and prescription. It is important to say that it is possible that every event in a TIRP is formed with same type of event. The rest of the charts can be found in the appendices C.

Chapter 7

Evaluation: Selecting temporal patterns for classification

Applying frequent temporal pattern mining can result in a very large number of patterns, and most of them can be not relevant. It is important to develop effective methods to select small subset of patterns and evaluate them. The Pattern selection is a complex task because when a pattern P is frequent, all the sub patterns inside in P are frequent too (as a results of the frequent pattern mining method), having a lot of redundant information. In the chapter on experiments and results, the performance of the algorithm and the shape of TIRPS were studied. In this chapter, a selection and a classification process were executed.

7.1 Selection Methods

Determine the sample size is an important process in research. Some scenarios are possible, the first one can be insufficient number of sample elements, giving bad estimation and no significant results. The second one can be a very large sample number that does not provide any type of relevant information. The MIMICIII database initially had information on 40,000 patients, of which 10,000 were selected with cardiovascular problems. In general, a pattern selection phase has high complexity since a pattern that appear many times does not mean that it is a pattern that gives relevant information. In fact, using a minimal support of 0.1, it is possible to find patterns that may be rare or unknown, but relevant, unlike others that may be very frequent and do not provide any information.

For the selection of each of the TIRPS, the following cases were followed:

- **Two Phased approach**: Select the top 2 or one for each case of study . In the case of smaller thresholds, select those that are close to this threshold. This selection processed was used too in [4].

- **Non-probabilistic sampling**: The selection of the elements of the sample was not made randomly, it was made by the researcher. These samples are less representative than those obtained by probabilistic sampling, but are easy to get. The principal disadvantage is the risk of obtaining too much bias with no possibility to generalize the results.

Samples were taken following the mentioned approaches of the different cases submitted for analyzed the quality of the patterns obtained. The results obtained from the selection phase can be found in the appendices B. After the selection phase, the classification phase was done with the researcher and a health staff(two doctors). It is important to mention that the health staff were not experts in cardiovascular domain and were not researchers, so the classification error can be high. A study using the confusion matrix must be done in the future repeating the study with an expert in the domain and a specialize researcher. The classification process was a complex task, but an important one for two different reasons. The first one was the difficulty of knowledge transfer between the researcher and the health staff. The second one was to be able to understand their needs as a potential users.

One of the cases that were executed for this study, was to skew the algorithm to study patterns between diagnoses exclusively. The configuration used was vertical support with a minimal support of 0.05 in 1000 patients. In the table 7.1 are the cases obtained. Originally in the abstract symbols, the ICD_10 code of the diagnosis number is stored. The table 7.2 has the same information, but with the diagnosis name translated according to the respective code.

EventI	RelI	EventII	RelII	EventIII
Diagnosis 99592	before	Diagnosis 0389	before	Diagnosis 51881
Diagnosis 99592	before	Diagnosis 78552	before	Diagnosis 51881
Diagnosis 4280	before	Diagnosis 99592	before	Diagnosis 78552

Table 7.1: Example of Diagnosis pattern (ICD 10 code)

EventI	RelI	EventII	RelII	EventIII
Severe sepsis	before	Salmonella infection	before	Acute respiratry failure
Severe sepsis	before	Septic shock	before	Acute respiratry failure
Congestive heart failure	before	Acute respiratory failure	before	Septic shock

Table 7.2: Example of Diagnosis pattern

Taking this pattern as an example, several questions were asked to the health staff. The first one was regarding the relationship between the events. "Is there any relationship between sepsis and the rest of the diagnosis?"

> **Answer**: *Sepsis is a generalized infection of the human body. A septic patient has an infection that radiates through the blood, transmitting this to all organs of the body. This produces a multi-organ failure. The heart, lungs and other organs are not exempt from that failure. The bacteria that caused the Sepsis are going to attack the organs causing nephritis, endocarditis and lung problems. It settles, colonizes and multiplies. Sepsis is nothing more than an expression to encompass the entire infected human body.*

Analyzing the event found, it was classify as known because there is a fairly strong relationship. Sepsis, Salmonella, and another consequences and complications are related as previously expressed. The interesting thing is that in the chapter of data exploration, sepsis appeared as the first cause of death in the population studied(the different was this information was taken from the admission and in this case the diagnosis were extracted from Diagnose event). Given the nature of the algorithm, it is not possible to know the time between each event, only to know its relationship in time. In the table 7.3, there is another example found as a pattern. In this case, a little more complex to understand. Related questions were asked between the use of the drug ***Iso-Osmotic Dextrose*** (infections), ***Heparin Sodium*** (anticoagulant) and the existence of ***abnormal Ph***.

> **Answer**: *Ph is normally neutral in humans, around 6.5. In infections it tends to decrease, becoming more acid. Even though there are cases with respiratory conditions that alkalosis occurs instead of acidosis. In the case of urine infections this occurs: Is normally acid, and with the infection of the urine it becomes alkaline. The truth is that in the sepsis or infection independently of the pathogen that causes it, it alters the Ph. Osmocitc medicine is used for bacterial infections. Having an abnormal Ph may be related to having an infection.*

EventI	RelI	EventII	RelII	EventIII	RelIII	EventIV
INR(PT) normal	before	Pantoprazole	before	Phosphate normal	before	Heparin Sodium
Iso-Osmotic Dextrose	before	NS:500 ml.	before	Heparin Sodium	before	pH abnormal
Iso-Osmotic Dextrose	before	NS:500 ml.	before	Heparin Sodium	before	Hematocrite abnormal
Pantoprazole	before	Phosphate normal	before	Heparin Sodium	before	INR(PT) normal

Table 7.3: Example of a TIRP

For each of the selected patterns, the same analysis was made. Five patterns of each case were selected and the following classification results were obtained. In the table 7.4, there is the classification of patterns using vertical support. It was interesting that the in the case of using 1000 patients with a lowest minimal support, a 100 % of known classification was found. This was the case of using only diagnosis events. By the other hand in the case of 3000 patients, the classification was 0 % as known event.

Number of patients	Threshold	Class #1: Not information	Class #2: Known
1000	0.1	40 %	60 %
1000	0.3	80 %	20 %
2000	0.1	60 %	40 %
2000	0.2	40%	60%
3000	0.1	100 %	0%
8000	0.3	60 %	40%
10000	0.05	0%	100%

Table 7.4: Classification Table of Patterns with Vertical Support

Number of patients	Threshold	Class #1: Not information	Class #2: Known
2000	0.1	60 %	40 %

Table 7.5: Classification Table of Patterns with Horizontal Support

In the table 7.5 are the patterns classified with horizontal support. 40 % was classified as known and 60 % as not provide information. The general appreciation during the review of several patterns was that a large part of the patterns found in this configuration had information on laboratory tests that were mostly normal, so they did not provide information that was relatively interesting. In vertical duration interval support, the sample taken (5) did not provided any

information.In general, when these patterns were observed, many patterns had 0 in the difference of intervals. This is caused because many time points in the raw data, started and ended at the same moment. In the FIgure 7.6 there is the classification table of patterns using duration vertical support.

Number of patients	Threshold	Class #1: Not information	Class #2: Known
1000	0.1	100%	0 %

Table 7.6: Classification table of Patterns with Vertical Duration Interval Support

For the selection of the patterns in the case of wanting to search for a specific diagnosis, it is possible to make queries in the database where they are stored. In the figure 8.1, there is an example asking for the diagnosis in position two of a TIRP with an specific diagnosis.

```
mysql> select * from EDGES where order_='2' and typeA='diagnostico' and nodeA like '%Diagnose%';
+--------------------------------------+-----------------------+-------------+---------------------------+--------------+----------+--------+
| idTran                               | nodeA                 | typeA       | nodeB                     | typeB        | relation | order_ |
+--------------------------------------+-----------------------+-------------+---------------------------+--------------+----------+--------+
| 99134788-ea3a-11e9-8b75-842b2b430272 | 4 The Diagnose is78552 | diagnostico | 1 Vial : 1 VIAL. Send Vial | prescription | BEFORE   |      2 |
| a2739080-ea3a-11e9-8b75-842b2b430272 | 4 The Diagnose is78552 | diagnostico | 1 D5W : 250 ml. 250mL Bag  | prescription | BEFORE   |      2 |
+--------------------------------------+-----------------------+-------------+---------------------------+--------------+----------+--------+
```

Figure 7.1: Query Example

On the other hand, in case of looking for a more user friendly interface, an approach to an implementation was made using Neo4j to visualize the data. Neo4j is a graph-oriented database, but gave certain limitations during its use, so it was used for a particular case of visualization.

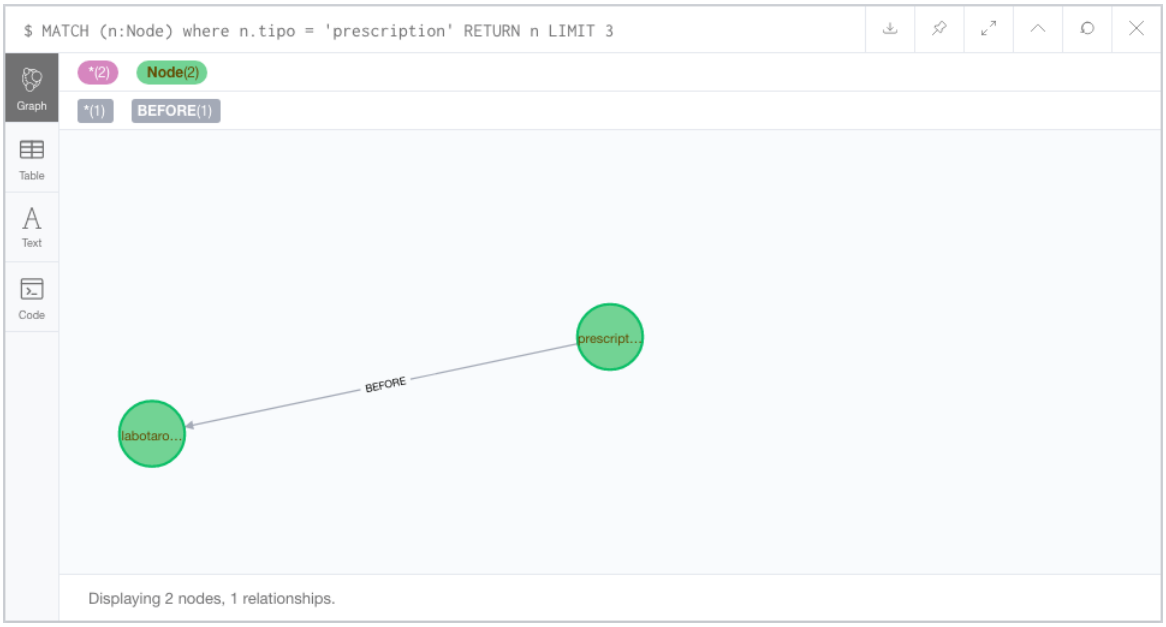

Figure 7.2: Lab prescription TIRP in Neo4j

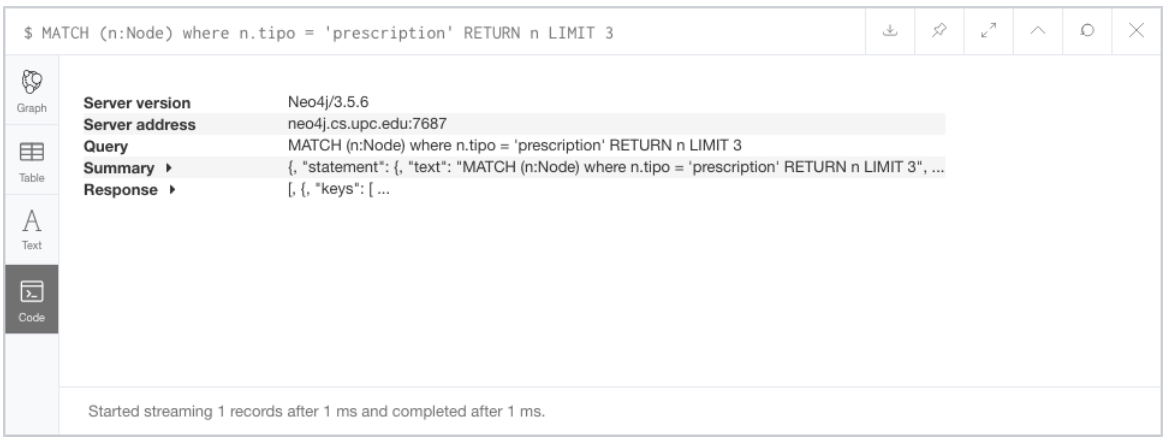

Figure 7.3: Lab prescription query in Neo4j

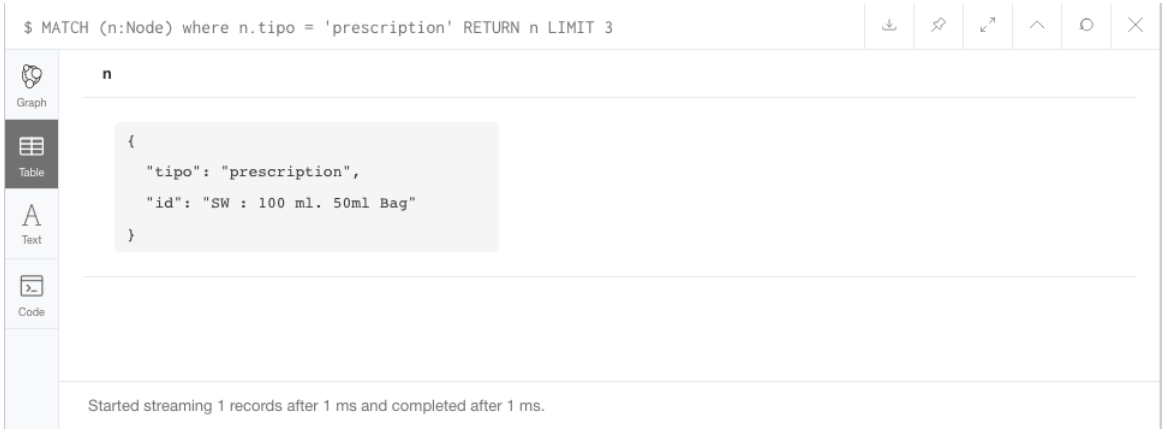

Figure 7.4: Lab prescription information in Neo4j

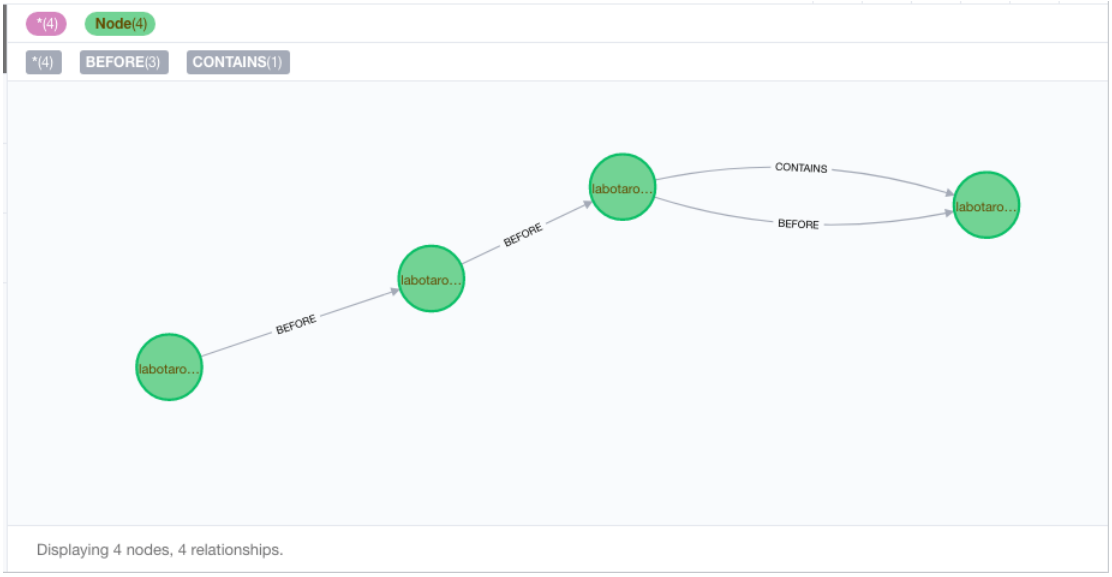

Figure 7.5: Example of TIRP in Neo4j

In Figure 7.2 there is the representation of a relationship between a laboratory event and prescription event. In Figure 7.3 and 7.4 are the query and the information related to the TIRP

selected. In the Figure 7.5 there is another example of a TIRP.

Chapter 8

Implementation and tools

8.0.1 Data Structures

The design of a solution is one of the most important aspects to achieve the success in the development of a problem. There must be a balance between some variables as performance, memory cost, CPU cost, results, etc. Depending of the design solution, one variable can be sacrificed respecting to another one. Choosing the correct data structure is an important decision for solving complex problems as working with large datasets. For the solution of the 2-sized TIRPS, a double dimension HashMap was used. `HashMap<Symbol,Symbol>=[]`. A Hashmap is represented as a dictionary in Python. For creating the TIRPS of k-sized, where $k \leq 3$, an object called TIRP was created, having the information of symbols, relations, and frequencies of appearance.

8.0.2 Tools

The thesis was developed using the Computer Science department High Performance Computing system (HPC). This is running a queue manager environment that collects all the user requests / jobs and scheduler them prioritizing every user task using several defined criteria (user quota, estimated execution time, RAM, CPU cores). As the thesis was done using private information, all the data were stored in the node of the servers. The code of the algorithm was implemented using Python 2.7. The data were coming from a XML and was transformed in a data frame using Pandas.

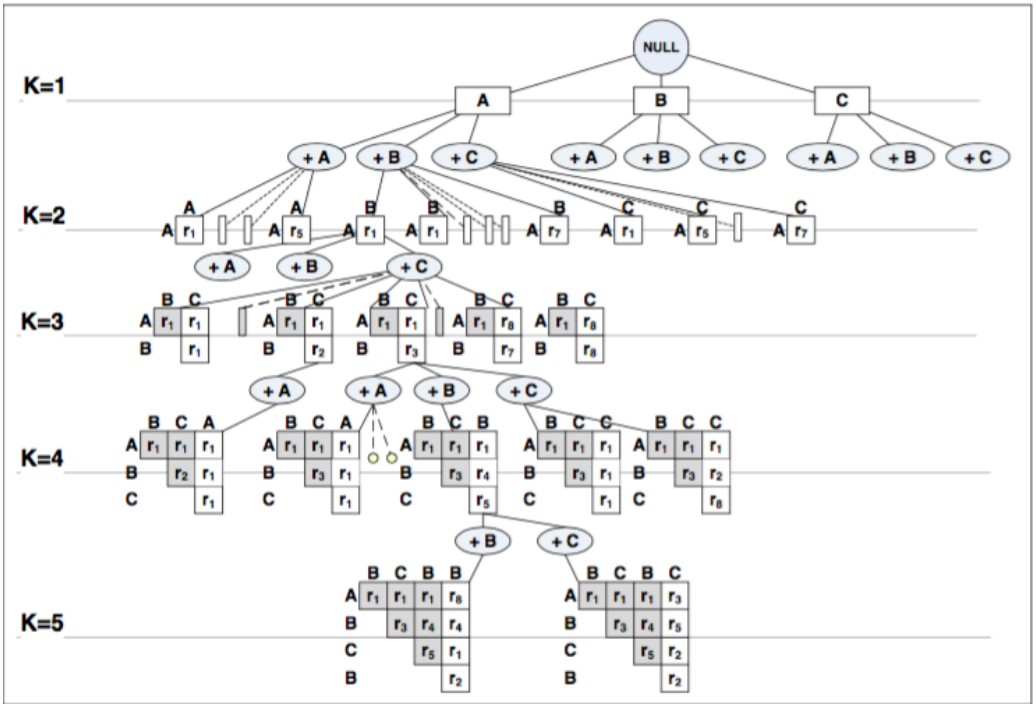

Figure 8.1: Final Visualization of the TIRPS

This algorithm, even with the optimization, took a lot of time processing, so the code was implemented with parallelism using multiprocessing. Multiprocessing is a package of python that supports spawning processes using an API similar to the threading module. The design consisted in divided the numbers of 2-sized symbols in 10 blocks, so each block was taken for a process.

About the Databases used, the first approach was used Ne04j for storing the TIRPS. Neo4j is a free graph-oriented database software, implemented in Java.1 2. It works as a motor of persistence oriented to transactions that stored structure data in Graphs instead of tables. Then, the approach was used Neo4j as a data visualization tool for experimental purpose. The reason of this was that one of the limitations of Neo4j was that only support one database per instance. For this reason, for an experimental purpose, only one instance was used for visualization. The experiments were run using Mysql (Relation Database) to store the information of the TIRPS as sized, pattern, relations, frequencies, etc.

Chapter 9

Conclusions

Pattern matching problems in medical records is one of the most important problems of the last decades in healthcare. The availability of medical records and the arrival of big data, have allowed to do better studies for analysis, predictions and search for hidden behaviors in data. Thanks to the information contained in EMR, new analysis can be done to discover new frequent temporal patterns or treatment pathways.

Temporal pattern matching problems can be solved using temporary data mining techniques, analyzing the data in time intervals. The work in this thesis focused on the implementation of the KarmaLego algorithm with some improvements respectively of the structure and design adapted to our problem. Originally, the proposed algorithm was designed for few entities, but in our cases it was implemented using a large number of different entities and different events in MIMIC III, which increases computational complexity.The first challenge was the processing phase. The raw data arrived in XML format and each event had huge granularity. For this, abstract concepts were created by extracting important information from each event. We had to observe each event to take only the most important characteristics for the creation of each abstract symbol.

For the generation of patterns, several configurations were developed and modified to achieve the analysis of our study: Vertical Support, Horizontal Support and Vertical Duration Interval Support. Regarding the performance of the algorithm, in the three configurations was quite slow and consumed a large amount of memory. Of the three configurations, the configuration that had information of the time interval in the abstract symbol Vertical Duration Interval support

was the one that generated more patterns of size three compared to the others with the same number of patients and same minimal support. This is because, is normal to have more patterns, having the same concept with different granularity. Now, in this configuration as the length of the patterns increases, the number of patterns declines with respect to the others. Otherwise, horizontal support produced many patterns and grew over time (with increasing length). This was due to its configuration, of knowing a pattern globally.

From a conceptual point of view, the vertical support configuration was the most important based on the premise that "It is more important that many patients meet a pattern, which few patients satisfy." On the other hand, as expected, as the number of patients increased and minimal support decreased, the number of patterns increased in each case. Regarding the shape of the TIRPS, the laboratory event represented more than 75% in each case, followed by the diagnosis and prescription event. This is because the laboratory events had a lot of data and was biased for that reason. On the other hand, something interesting was that with the Vertical Duration Support configuration, with 1000 patients and 0.1 minimal support, the prescription represented almost the 50% of the TIRP, this was expected because there is a lot of different dose and time of prescription of the same medication.

The second part of the analysis was the selection and classification of the patterns obtained.The selection was made by taking two patterns from the top of each case. The top is related to whose supporting count is close to the threshold studied. On the other hand, three patterns were selected by the researcher. Once each pattern was obtained, with the help of two medical staff, analyzes were carried out to classify them as "known" or "did not provide information." It should be noted that the medical staff was not a specialist in heart problems (filter used in the MIMIC database for our study). The exercise was complex to understand for medical staff, but it was very useful to know the needs as a possible users. Questions were asked about the relationship of the events that came out in the patterns and with this, they were classified. The classification phase was important to see the quality of the patterns studied. However, it is necessary to select a much more representative sample for the future. There were interesting patterns, such as those obtained in a case where the algorithm was biased to obtain diagnosis sequences. In this case the threshold used was 0.05 for vertical support. There was not much difference between the patterns with respect to quality according to the configuration used, but

in the future a study about the relevant pattern must be done to get better analysis about the patterns obtained.

In conclusion, the algorithm does not need any guidance to generate the patterns, but can bias to find only relationships between specific events. Once the patterns have been generated, searches can be done for specifics type of events. The labels used to temporarily represent the patterns were qualitative, with the Allen Temporal Relations "overlap, contains and before." The patterns found were interesting to study, although there are some points of improvement in pre-processing and performance. The algorithm implemented and its improvements met the intended objective.

Chapter 10

Future Work

- Reducing TIRP instances variance via pre-clustering: In one of the approaches,a vertical support in instances adding the duration of the Interval into the abstract symbol was applied. The proposal is clustering the instances into k clusters according to their duration prior to the mining process by using an appropriate method such as the k-means clustering algorithm. This is proposed by Moskovitch [18].
- The Incremental proposal(Streaming): If we focus the problem in a Big Shot, having Data coming from different hospitals, a lot of patients and we wanted an algorithm that works incrementally, without calculating every step every time. One approach that can be done is attack the problem like a Big Data Problem working with streaming data, as a Data Pipeline cutting the algorithm per pieces to performed incrementally.It is important to mention that a study of the that the correctness and completeness of the algorithm must be demonstrated from an incremental point of view.
- Improve the pre-processing phase for creating more readable abstract concepts.
- Repeat the selection process with more patterns.

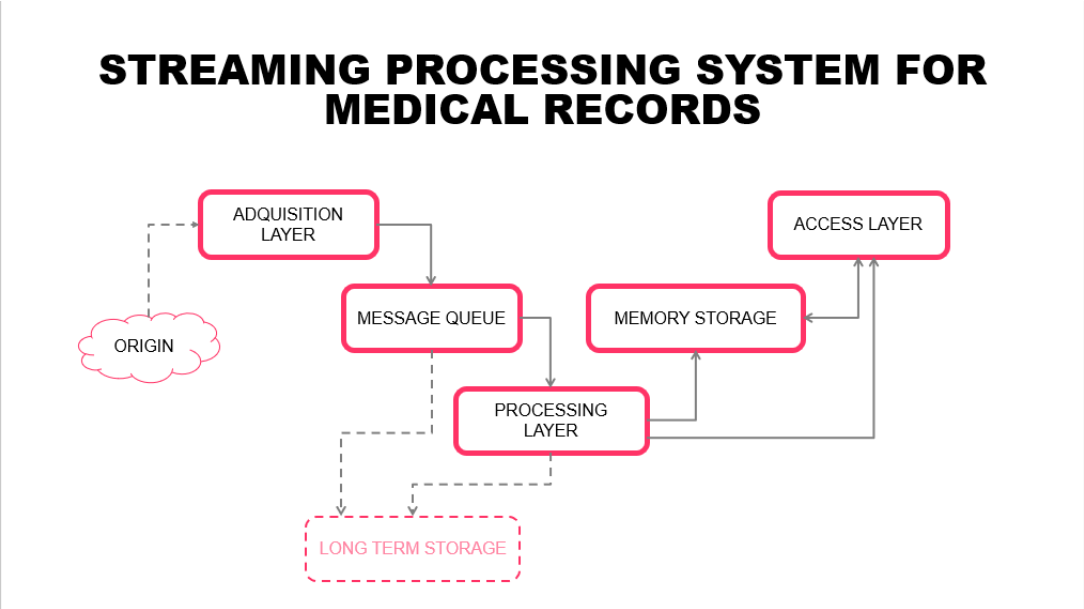

Figure 10.1: Processing system for medical records

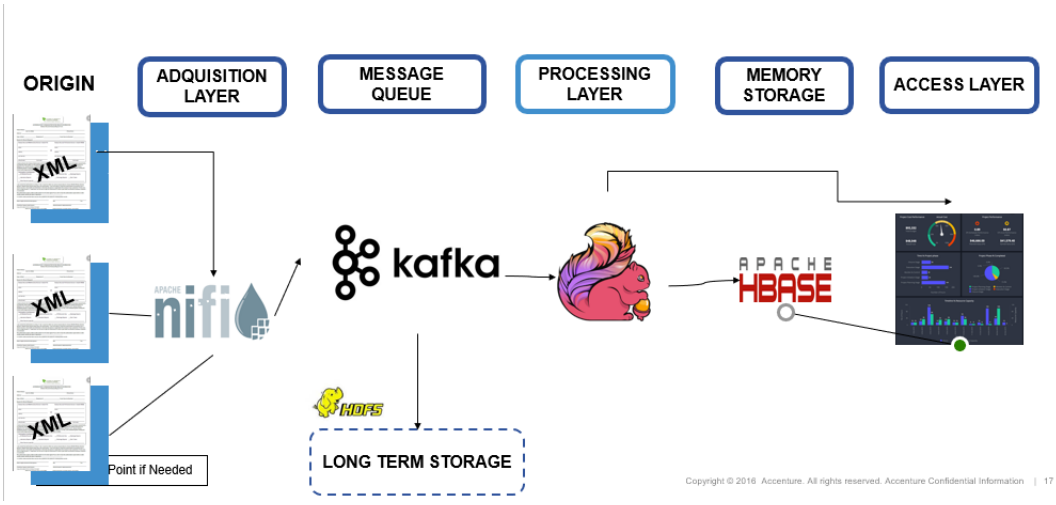

Figure 10.2: Technology proposed for a Pipeline of Medical Records

Bibliography

[1] Agrawal, R., and Srikant, R. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering* (Washington, DC, USA, 1995), ICDE '95, IEEE Computer Society, pp. 3–14.

[2] Allen, J. F. Maintaining knowledge about temporal intervals. *Commun. ACM 26*, 11 (Nov. 1983), 832–843.

[3] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), KDD '02, ACM, pp. 429–435.

[4] Batal, I., Valizadega, H., F.Cooper, G., Hauskrecht, M., Elovici, Y., and Shahar, Y. A temporal pattern mining approach for classifying electronic health record data.

[5] Ch.Bhavani, P. Improving efficiency of apriori algorithm. *International Journal of Computer Trends and Technology (IJCTT) 27*, 2 (september 2015), 1–7.

[6] Chen, J. An updown directed acyclic graph approach for sequential pattern mining. *IEEE Transactions on Knowledge and Data Engineering 22*, 7 (July 2010), 913–928.

[7] Grninger, M., and Li, Z. The time ontology of allens interval algebra. *IEEE Trans. on Knowl. and Data Eng. 14*, 4 (July 2017).

[8] Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[9] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. Freespan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2000), KDD '00, ACM, pp. 355–359.

[10] Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. *SIGMOD Rec. 29*, 2 (May 2000), 1–12.

[11] Hoppne, F. Learning temporal rules from state sequences, proceedings of wltsd-01.

[12] Hppner, F. Learning temporal rules from state sequences.

[13] Jian Pei, Jiawei Han, Mortazavi-Asl, B., Jianyong Wang, Pinto, H., Qiming Chen, Dayal, U., and Mei-Chun Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering 16*, 11 (Nov 2004), 1424–1440.

[14] Johnson AEW, Pollard TJ, S. L. L. L. F. M. G. M. M. B. S. P. C. L., and RG, M. Mimic-iii, a freely accessible critical care database.

[15] Kam, P.-s., and Fu, A. W.-C. Discovering temporal patterns for interval-based events. In *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery* (London, UK, UK, 2000), DaWaK 2000, Springer-Verlag, pp. 317–326.

[16] Kaur, C. Association rule mining using apriori algorithm: A survey. *International Journal of Advanced Research in Computer Engineering & Technology 2*, 6 (jun 2013), 2081–2084.

[17] Li, T., Wang, W., and Chen, Q. On the sequential pattern mining algorithm based on projection position. pp. 460–463.

[18] Moskovitch, R., and Shahar, Y. Temporal patterns discovery from multivariate time series via temporal abstraction and time-interval mining.

[19] Papapetrou, P., Kollios, G., Sclaroff, S., and Gunopulos, D. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems 21*, 2 (Feb 2009), 133.

[20] Pujari, A. K. *Data Mining Techniques*. Universities Press, 2001.

[21] Rao, A. V. V., and Eedala Rambabu, B. Association rule mining using fptree as directed acyclic graph. In *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)* (March 2012), pp. 202–207.

[22] Robert Moskovitch, Y. S. Fast detection of time intervals related patterns.

[23] Shahar, Y. A framework for knowledge-based temporal abstraction. *Artificial Intelligence 90*, 1 (1997), 79 – 133.

[24] Sheetrit, E., Nissim, N., Klimov, D., Fuchs, L., Elovici, Y., and Shahar, Y. Temporal pattern discovery for accurate sepsis diagnosis in ICU patients. *CoRR abs/1709.01720* (2017).

[25] Shknevsky, A., Shahar, Y., and Moskovitch, R. Consistent discovery of frequent interval-based temporal patterns in chronic patients data. *Journal of Biomedical Informatics 75* (2017), 83 – 95.

[26] Srikant, R., and Agrawal, R. Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology — EDBT '96* (Berlin, Heidelberg, 1996), P. Apers, M. Bouzeghoub, and G. Gardarin, Eds., Springer Berlin Heidelberg, pp. 1–17.

[27] Tolarczyk, A., and Siwek, K. Sequential pattern recognition for medical records analysis. In *2016 17th International Conference Computational Problems of Electrical Engineering (CPEE)* (Sep. 2016), pp. 1–3.

[28] Winarko, E., and Roddick, J. F. Armada an algorithm for discovering richer relative temporal association rules from interval-based data. *Data Knowledge Engineering 63*, 1 (2007), 76 – 90. Data Warehouse and Knowledge Discovery (DAWAK 05).

[29] Zaki, M. J. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning 42*, 1 (Jan 2001), 31–60.