UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

FACULTAD DE INGENIERIA

PEDECIBA
MEC-UDELAR

Instance-based learning following physician reasoning for assistance during medical consultation

Matías Galnares

Maestría en Informática, PEDECIBA
Universidad de la República

Montevideo – Uruguay
Noviembre de 2021

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Instance-based learning following physician reasoning for assistance during medical consultation

Matías Galnares

Tesis de Maestría presentada al Programa de Posgrado Maestría en Informática, PEDECIBA, de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magister en Maestría en Informática, PEDECIBA.

Directores:
Franco Simini
Sergio Nesmachnow

Director académico:
Sergio Nesmachnow

Montevideo – Uruguay
Noviembre de 2021

Galnares, Matías

Instance-based learning following physician reasoning for assistance during medical consultation / Matías Galnares. - Montevideo: Universidad de la República,

XV, 87 p.: il.; 29, 7cm.

Directores:

Franco Simini

Sergio Nesmachnow

Director académico:

Sergio Nesmachnow

Tesis de Maestría – Universidad de la República, Programa Maestría en Informática, PEDECIBA, 2021.

Referencias bibliográficas: p. 83 – 87.

1. inteligencia computacional, 2. asistencia medica, 3. aprendizaje basado en instancias, 4. atención sanitaria, 5. sistemas de apoyo a la decisión clínica. I. Simini, Franco, Nesmachnow, Sergio, . II. Universidad de la República, Programa de Posgrado Maestría en Informática, PEDECIBA. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Antonio Daniel Mauttone Vidales

Cecilia Dias Flores

Lincoln de Assis Moura

Montevideo – Uruguay
Noviembre de 2021

Acknowledgments

Countless people supported my effort on this research. Professors Franco Simini and Sergio Nesmachnow provided invaluable feedback on my work, helping me to overcome every difficulty encountered during my research.

I especially want to thank Professor Franco Simini. His courses on Medical Informatics led me to increase my knowledge on the fascinating area of healthcare, and his encouragement to write this thesis made me confident that my investigation was worthy of the topic.

I am indebted to my Academic Director Sergio Nesmachnow, for his continued guidance and constant support during all stages of this thesis. His unassuming approach to research and science is a source of inspiration. This approach is reflected by his simple but clear writing style, which is something I hope to carry forward throughout my career.

I also thank PEDECIBA for supporting the dissemination of my research, by partly funding a publication in an international journal.

Finally, my family deserves infinite gratitude: my father for teaching me to fall in love with my vocation, my mother for teaching me to persevere in every important project, and my wife for supporting me in the setbacks of my live. To my sons I give everything, including this.

Do the best you can until you know better. Then when you know better, do better.

Maya Angelou

RESUMEN

Esta tesis de maestría presenta un sistema automático que modela el conocimiento clínico para seguir el razonamiento médico durante una consulta ambulatoria. Se aplica un método de aprendizaje basado en instancias para proporcionar sugerencias durante el registro en una historia clínica electrónica. El método de aprendizaje propuesto tiene en cuenta la base de conocimiento clínico de cada médico, para presentar sugerencias basadas en tipos de casos clínicos previamente definidos, y deducidos según una métrica de similitud específicamente diseñada.

El sistema se valida en un escenario de uso real, con la participación de estudiantes avanzados de medicina de un curso de informática médica de la Universidad de la República, Uruguay. Los resultados demuestran que el sistema propuesto es 2.5× más eficiente que una herramienta empírica de referencia para sugerencias, y dos órdenes de magnitud más rápido que un método de aprendizaje Bayesiano, considerando un marco de referencia de 250 tipos de casos clínicos. Los resultados también demuestran que el método de aprendizaje es capaz de producir sugerencias en tiempos razonables, incluso cuando se procesan grandes volúmenes de datos. Una encuesta realizada a estudiantes avanzados de medicina destaca que el enfoque propuesto se considera apropiado para la práctica médica.

Esta investigación introduce una estructura formal para representar con precisión el conocimiento clínico, que apoya a los principales flujos que ocurren durante las consultas médicas. También se proporciona un marco que permite implementar un sistema en tiempo real capaz de asistir a los médicos durante sus consultas, y que además ayuda a reducir el tiempo de registro.

Palabras claves:
inteligencia computacional, asistencia medica, aprendizaje basado en instancias, atención sanitaria, sistemas de apoyo a la decisión clínica.

ABSTRACT

This Master Thesis presents an automatic system for modeling clinical knowledge to follow physicians reasoning in medical consultation. Instance-based learning is applied to provide suggestions when recording electronic medical records. The proposed learning method takes into account the clinical knowledge base of a physician, in order to present suggestions based on previously-defined clinical case types, and deduced according to an ad-hoc similarity metric.

The system is validated on a real case study involving advanced medical students of a Medical Informatics course at Universidad de la República,Uruguay. Results show that the proposed system is 2.5× more efficient than a baseline empirical tool for suggestions, and two orders of magnitude faster than a Bayesian learning method, when processing a testbed of 250 clinical case types. Results also demostrate that the learning method is able to produce suggestions in a reasonable time frame, even when processing large volumes of data. A survey performed on advanced medical students highlights that the proposed approach is considered appropriate for medical practices.

The research introduces a formal structure to accurately represent clinical knowledge, supporting the main flows of medical consultations. A frame for implementing a real-time system for assisting physicians during medical consultations is also provided, which helps reducing the time needed to register medical consultations.

Keywords:
computational intelligence, medical assistance, instance-based learning, healthcare, clinical decision support systems.

List of Figures

List of Tables

List of Acronyms

List of acronyms used in this Master Thesis.

CC Conceptual Component

CCT Chronic Case Type

CDSS Clinical Decision Support System

CE Conceptual Element

CKB Clinical Knowledge Base

CT Case Type

EMR Electronic Medical Record

LIS Laboratory Information System

MCT Multiple Case Type

PACS Picture Archiving and Communication System

PCT Partial Case Type

SOAP Subjective data, Objective data, Assessment and Plan

UT Unit of Thought

Contents

Chapter 1

Introduction

The search for better medical practices is a perpetual challenge for modern medicine. In this regard, computational intelligence has emerged as a promising subject for developing smart systems in healthcare practice (Castellano and Casalino, 2020). Computational intelligence allows implementing automatic tools, enabling physicians to provide patients with a better quality of attention by performing early and accurate diagnosis and improving treatment. Furthermore, automatic systems and technologies based on computational intelligence have proven to be useful solutions to be applied in clinical practice. Some important advantages of intelligent automatic methods over traditional ones include better efficiency, accuracy, and consistency, among others. Consequently, intelligent automatic systems provide physicians with more time for face-to-face consultation, and more time for critical tasks and critical cases (López-Rubio et al., 2015).

A specific subject where the learning capabilities of computational intelligence methods are very helpful to improve medical practice is the analysis and processing of Electronic Medical Records (EMRs). EMRs refer to digital records, collected by the individual medical practice, that contain the general health information of patients (Habib, 2010). They usually consist of several types of health data, including, but not limited to, demographics, past surgical history, medical family history, social history, medication, allergies, test results, and medical images.

Currently, the majority of medical history recording products are based on predefined templates, which provide very limited flexibility for writing patients medical records. Several drawbacks are identified on medical recording prod-

ucts. In particular, structured data entries are an obstacle to flexible writing in medical record applications, and are disapproved by physicians, who usually prefer writing free text (González et al., 2018). Despite the fact that physicians are getting used to work with electronic medical records, they still have difficulties dealing with long lists of pre-conceived variables, usually included in EMR systems. Although conventional EMR systems are useful to achieve legible, accessible, and complete documentation of medical consultations, they also cause several difficulties for physicians who adopt them. In many cases, physicians spend a lot of time searching for an option to record what they intend to annotate. Another drawback of conventional products concern to the alerts and suggestions they provide, which are generally based on previously defined rules, or according to mechanisms whose behavior remains the same throughout their operational life.

Despite the fact that medical history recording products are able to interact with other healthcare systems, such as Laboratory Information System (LIS) and Picture Archiving and Communication System (PACS), the dissatisfaction of physicians with actual products is certainly an obstacle to overcome. Physicians are increasingly aspiring to work with sophisticated Clinical Decision Support System (CDSS) that facilitate their clinical practice during medical consultations.

Relevant works have been proposed on the literature for assisting medical professionals during their clinical activities. The works reviewed on the literature are able to detect patterns, provide recommendations, predict future behaviors, and suggest clinical practices, among others features (Wang et al., 2015; Klann et al., 2014; Nakai et al., 2016). In general, reviewed works provide suggestions for diagnoses, prognoses, and treatments. There are also important contributions from research focused on specific clinical conditions, which take advantage of specific medical knowledge of a given area (Lin et al., 2016; Rane, 2015; Esteban et al., 2015). Furthermore, all reviewed works contribute to reducing error-prone steps during the clinical process.

Conventional EMR systems are template-based products that generate poor quality data, due to long search mechanisms and excessive mandatory fields, which often add noise to the relevant patient information (González et al., 2018). Moreover, structured data entry systems do not take into account the particularities of the annotations of each physician, failing to effectively record the singularities of a medical consultation. The rigid structure of the

templates to be filled-in during medical consultations does not fit the reasoning of physicians, nor their way of thinking. The research reported in this Master Thesis is motivated by the need to further explore new ways of capturing, storing, and fostering medical reasoning. Thus, a formal proposal must be conceived to provide an accurate tool capable of following medical reasoning, aiming at helping physicians during medical consultations. In this line of work, this research presents a novel approach to represent clinical knowledge, which supports an appropriate methodology to follow reasoning in medical consultation. Likewise, the proposed representation does not pose formal restrictions to physicians, as they usually find when using common clinical data entry systems. An instance-based learning method is also introduced to provide suggestions in order to help during the process of registering a medical consultation.

Improvements in medical consultation assistance are achieved by taking advantage of systems that properly manage clinical information. To improve over medical assistance, physicians are provided with new healthcare tools, considering that healthcare assistance during medical consultations is improved when the physician is able to:

(i) Efficiently record all the information of a medical consultation, by reducing the time spent on mere data entry in order to gain more time to interact with the patient.

(ii) Use automatic clinical suggestions to reach an accurate diagnosis, or an appropriate indication of treatments.

(iii) Reduce medical errors, resulting from the human condition of the professional.

(iv) Record each medical consultation considering the special relevance of the interoperability of clinical information.

(v) Reuse recorded information for statistical and research purposes.

Computational intelligence is applied to solve the deficiencies of current EMRs. Machine learning methods are used to learn features from previous registered healthcare data sets, in order to provide suggestions for diagnoses and treatments based on information previously registered. By applying computational intelligence, systems are able to automatically identify solutions of

similar clinical cases and subsequently incorporate the knowledge gained to assist physicians during medical consultations. Learning methods also contribute to reduce error-prone steps during the sequence of clinical tasks and decisions. Inevitable errors of human-based clinical practice are reduced, such as drug contraindications, medication allergies, adverse drug reactions, and unchecked chronic issues. Furthermore, machine learning methods are able to progressively enhance their accuracy based on feedback provided by their own use. An effective medical informatics support system must be adapted to the real health environment. In addition, a clinical evaluation of the usefulness of the system in real clinical work must be considered to determine its real capacity during clinical practice.

The proposed approach was evaluated in a case study involving advanced medical students. Students tested the feasibility of the approach by using a proof-of-concept prototype implementation. The performance of the proposed learning method was satisfactory after evaluated on 250 clinical scenarios written by the students. Results showed that the learning method was able to produce suggestions in a reasonable time frame, even when processing large volumes of data. The proposed instance-based learning method was compared with a baseline empirical tool called Praxis, which is able to generate suggestions for physicians during medical consultations. By considering the registration of the first 50 consultations of a physician, results showed a reduction factor of 2.5$\times$ regarding Praxis execution time. Moreover, the instance-based learning method significantly outperforms a Bayesian approach in terms of efficiency, due to its execution time is two orders of magnitude faster than the Bayesian learning method. The obtained results suggest that the proposed approach was useful to accelerate the process of taking notes, since 62% of the medical students were able to speed up the writing time of their medical consultations. A high potential impact on clinical care is projected, considering that a survey performed on medical students showed that 73% of participants deemed the prototype as an appropriate tool for medical practice, especially during medical consultations.

The main contributions of the research reported in this document include: (i) a formal structure to accurately represent clinical knowledge and support the main flows of medical consultations; and (ii) an instance-based learning method that helps to reduce the registration time of a medical consultation.

The following list summarizes the publications issued from this Master Thesis research.

- Galnares, M., Low, R., Nesmachnow, S. and Simini, F. (2018), Medical reasoning oriented orthesis to ease clinical practice and record keeping (poster). *Medical Physics and Biomedical Engineering World Congress*, June 3-8, Prague, Czech Republic.

- Simini, F., Galnares, M., Silvera, G., Álvarez, P., Low, R. and Ormaechea, G. (2020). Pattern recognition to automate chronic patients follow-up and to assist outpatient diagnostics. *Pattern Recognition Techniques Applied to Biomedical Problems*, pages 175–195, Springer International Publishing.

- Galnares, M., Nesmachnow, S. and Simini, F (2020). Clinical knowledge representation to follow physician reasoning. In Proceedings of *Sociedad Argentina de Bioingeniería 2020 Congress*, Maldonado, Uruguay.

- Galnares, M., Nesmachnow, S., and Simini, F. (2021). Instance-based learning following physician reasoning for assistance during medical consultation. *Applied Sciences*, 11(13). Special issue: Applications of artificial intelligence in medicine practice.

The content of this Master Thesis is structured as follows. Chapter 2 presents a review of related work on learning models for assisting medical professionals. An empirical tool called Praxis is presented in Chapter 3. The essential features of Praxis software were deduced from several meetings with staff directly involved in the software development. Chapter 4 describes a model proposed for representing clinical knowledge. Several flows to address relevant scenarios of medical consultations are also introduced by Chapter 4. A learning method proposed for generating suggestions for physicians is detailed in Chapter 5. Sample results from the experimental analysis are presented in Chapter 6. Finally, Chapter 7 presents the main conclusions of the research.

Chapter 2

Related work

This chapter presents a review of recent works in the health area related to Electronic Medical Record (EMR), including a description of associated machine learning techniques to improve medical assistance. The chapter is organized as follows. Section 2.1 comments works designed for general health scenarios. These works are focused on improving clinical decision support systems, using machine learning approaches. In section 2.2, works with specific clinical proposals are reviewed. These works are designed for helping health stakeholders on specific clinical scenarios. Section 2.3 introduces the Praxis software, which has been studied in detail in this Thesis. Finally, section 2.4 presents a summary of the relevant works proposed on the related literature.

2.1. Support systems for general clinical use

Relevant systems with significant learning capabilities are reviewed in this section. The systems are designed to cover different health scenarios for assisting medical professionals during their clinical activities. These systems can detect patterns and use the detected information to provide recommendations, as well as predicting future behaviors and suggesting frequent clinical practices. In general, these systems are able to identify similar clinical cases in order to provide suggestions for diagnoses, prognosis, and treatments. In addition, all the reviewed systems contribute to reducing error-prone steps during the clinical process.

An EMR system with treatment recommendations based on patient similarity was implemented by Wang et al. (2015). This novel system, called ISLEMR,

includes learning capabilities and real-time feedback by inferring patient similarities. ISLEMR is based on Hygeia (Li et al., 2012), an EMR proposed by researchers from Zhejiang University in China and Miyazaki University Hospital in Japan. To implement ISLEMR, a group of ad-hoc similarity metrics were defined, which make it possible to recommend treatment plans according to the treatments of similar patients. The similarity algorithm considers patient diagnoses, demographic data, vital signs, structured lab test results, and can query external clinical information systems, such as PACS, LIS, and pharmacy systems. The patient information is used by the system to present an ordered menu with inferred recommendations for treatment plans. Twelve thousand hospitalizations on a hospital in Beijing, China between December 2013 and April 2014 were analyzed during the system validation. The authors used a precision metric that evaluates the real use of the recommended menu items. Higher precision at first menu items indicates a better performance of the recommendation method. Precision results up to 80% were achieved for the first 10 items of the recommended menu. The proposed system was evaluated in a real environment, a Chinese hospital with more than 10.000 outpatients per day. However, the learning algorithm applied only considers structured data, which implies less precision in determining similarities of clinical cases.

Decision support can also be obtained from past clinical decisions. In order to improve guideline-based clinical decision support systems, Klann et al. (2014) proposed a learning approach to generate adaptive and context-specific treatment menus from past clinical information of patients. Each menu recommends a starting point for physicians, suggesting an initial draft to treat a specific situation. The learning approach considers four domain-specific Bayesian Networks and applies a Greedy Equivalence Search algorithm in the learning task phase. The Bayesian Networks represent four modalities: inpatient medicine, emergency department, urgent visit clinic, and the intensive care unit. The networks also contain 11.344 encounters by modeling the probabilistic relationships among orders and diagnoses, covering typical scenarios from different aspects of medicine. The authors evaluated the proposed system considering a hospital simulation, which evaluated the performance of the suggestion menus to predict clinical situations. The accuracy of the proposed system was measured by the area under the receiver–operator curve (AUC) (Campbell et al., 2007), which is a fundamental metric for diagnostic test evaluation. The experimental results demonstrated the predictive capabilities

of the system (0.78 of average AUC), outperforming a similar association rule mining approach, especially over less frequent cases. The proposed approach has important potential contributions for healthcare activities; it can reduce the workload of physicians by using recorded information to create human-readable suggestions for treatments and diagnoses.

Another common approach is the prediction of clinical practices using methods based on Support Vector Machines. Motivated by solving the ineffectiveness of rule-based clinical decision support systems, Nakai et al. (2016) proposed a novel model to assist physicians in the healthcare domain. The model can predict next clinical practices to be prescribed on a particular patient by using the information from previous practices of the same patient. Machine learning was used to predict future clinical practices by applying a Linear Support Vector Machine method. The authors used detailed information of clinical actions from Japan University Hospital to build the learning model. The information was obtained from the Diagnosis Procedure Combination/Per-Diem Payment System of Japan, which is the standard Japanese system for medical billing. The experimental data was divided into ten parts; one part was used as validation data and the other parts as learning data. The prediction accuracy was evaluated by ten-fold cross validation. The results demonstrated the high precision of the model when facing frequent clinical cases. However, low precision results were obtained when dealing with less common cases.

Another novel approach is related to extracting relations in medical documents with the help of structured and semantic analysis. Barbantan et al. (2016) proposed a medical decision support system that analyzes electronic health information and creates a medical structured-related concept model. To identify structured concepts, the system processes unstructured medical records by applying natural language processing tasks followed by a semantic analysis. The proposed solution is based on a learning approach that discovers relations between medical concepts, using a Support Vector Machine classifier. The experimental analysis was performed over a data set of 170 clinical documents from Beth Israel-Deaconess Medical Center, Boston, Massachusetts. The structured information and the relations between medical concepts were used to help diagnoses, medication predictions, and also to detect patterns about patients health.

Shen et al. (2015) proposed a multi-agent clinical decision support system by applying a case-based reasoning approach. Several ontological agents

were used to identify similar clinical cases in order to provide suggestions for diagnoses, prognosis, and treatments. Similar cases were detected using language analysis and an ad-hoc matching method. The system searches clinical cases by identifying important words, terminologies, and synonyms. Furthermore, medication allergies, adverse drug reactions, coexisting diseases, and other complications were evaluated for discarding candidate cases. Seventeen representative cases of gastric cancer were considered for testing the proposed system. Moreover, the authors were inspired by the work of Chau et al. (2003) and Farinelli et al. (2003) to define a new efficiency measure for evaluating the proposed matching method. The results showed that the system was able to achieve a matching rate of 78.2% for illnesses with simple syndromes.

Installé et al. (2014) developed a clinical data miner software framework for supporting clinical diagnostic. The proposed framework is composed of an electronic Case Report Form (eCRF) system and an integrated data preprocessing component. The eCRF system is based on templates and spreadsheets, following the ideas presented in OpenClinica (Collins et al., 2020). In addition, the data preprocessing component is able to analyze data and can apply machine-learning techniques over the information gathered by the eCRF. This system was evaluated in cases from the International Endometrial Tumor Analysis consortium, which has more than 4000 patients. Forty-two physicians were asked in a survey to evaluate their satisfaction with the framework, and the user-friendliness of the proposed system. The survey results showed that the system was considered user-friendly, and all physicians approved the possibility of using it in their own future works. Moreover, the survey showed that the integrated data preprocessing and the machine-learning techniques reduced error-prone steps during a clinical diagnostic process.

Zieba (2014) proposed a service-oriented support decision system for diagnose medical problems. The architecture of the system was designed to deal with both the problem of imbalanced distribution of clinical data and the missing values of attributes in medical records. The proposed system is composed of web services with learning capabilities, which deal with imbalanced data by implementing cost-sensitive Support Vector Machine methods. In addition, the problem of missing values of attributes was solved by splitting the incomplete data into complete data subsets, which were used for learning tasks. The quality of the system was evaluated using three ontological datasets. The first set was obtained from the Wroclaw Thoracic Surgery Center in

Poland and contain information of 1.203 patients with a high ratio of unknown attributes (45%). The second set was provided by the Institute of Oncology in Ljubljana, Slovenia and contains data of 949 patients with 20% of missing attributes. A third set was taken from the UCI Machine Learning Repository (Lichman, 2013), and it is composed of 286 records with a low ratio of unknown attributes (6%). The results demonstrated that the proposed system was able to predict diagnosis by generating decision rules that were presented to physicians with acceptable accuracy values.

Benmimoune et al. (2015) designed a hybrid medical platform to assist physicians during their clinical reasoning process. In this work, two components were identified as part of the process: a Rule-Based Reasoning (RBR) component with rules for general clinical cases, and a Case-Based Reasoning (CBR) component composed of clinical experiences. The proposed platform helps physicians gathering relevant information about the patient status by presenting an adaptive questionnaire according to each patient profile. Once the physician registers relevant information about the patient (symptoms, medical history, family history, lifestyle information), the platform uses the gathered information to search for the most similar stored case, following the CBR approach. An adaptation step is required to consider the differences between the actual case and the most similar inferred case, and then a new case is stored. If no similar case is found, the platform applies a RBR approach to deduce a solution according to rules defined by medical experts. In the worst case, the platform does not have enough data to make predictions, and therefore requests the physician to gather more information in order to repeat the cycle. The authors did not report an implementation or a prototype that proves the feasibility of their novel design on a real health environment.

Since many patients do not just have a single clinical condition, Wilk et al. (2017) proposed a framework to assist patients in multi-morbidity conditions. The proposed framework also considers patient preferences for suggesting customized clinical practice guidelines. The authors improved their previous works (Wilk et al., 2013; Michalowski et al., 2014; Wilk et al., 2014) by modeling clinical guidelines using actionable graphs and first-order logic, which makes it possible the simultaneous application of multiple clinical practice guidelines. Furthermore, a secondary medical knowledge component was designed to identify adverse interactions resulting from conflicting therapies. The logic-based framework was presented using a theoretical perspective. Two

multi-morbidity cases were exposed to illustrate the use of the proposed framework regarding patients suffering simultaneously from chronic kidney disease, hypertension, and atrial fibrillation. In addition, a high-level proof of concept implementation was presented to show the feasibility of the proposed framework. The authors planned to improve their work by evaluating the complete solution of the proposed framework on a real emergency triage.

2.2. Systems for specific clinical conditions

This section presents a review of recent works designed to predict or detect specific clinical patient conditions. These works take advantage of specific medical knowledge to provide support for predicting or detecting a particular disease or condition. Predicting sequences of clinical events can be easier when the scope of the medical system is reduced to a single or a small set of clinical conditions. Several systems are reviewed, which were proposed for helping important and well-studied areas, such as prevailing diseases, chronic diseases, cardiac rehabilitations, heart failures, and cancer therapies. All reviewed works contribute to give better assistance for specific clinical cases.

Lin et al. (2016) proposed a computational prediction system for helping the process of decision-making of multidisciplinary teams regarding cancer therapies. The proposed system utilizes machine learning methods for suggesting recommendations about breast cancer therapies, including chemotherapy, endocrine therapy, and biologic-targeted therapy. A number of machine learning methods were studied, including Support Vector Machine, Bayesian Classifier, Multivariate Logistic Regression, Nearest Neighbors, Ripple Down Rules and Decision Trees, in order to determine the most accurate technique for breast cancer therapies. The proposed system was evaluated using information provided by a tertiary cancer referral center in Sydney, Australia. The information described clinical data of 1.065 patients from 2007–2015 who underwent at least one cancer therapy. A ten-fold cross validation was applied to compare the accuracy of the proposed methods. Moreover, the results of each learning technique were compared with internationally accepted guidelines of the European Society for Medical Oncology (ESMO) and the National Comprehensive Cancer Networks (NCCN). On the one hand, results showed that the proposed system has similar behavior to both ESMO and NCCN guidelines regarding the prediction of endocrine and biologic-targeted therapies. On the

other hand, significantly different predictions were found for chemotherapy cases. The proposed system achieved better chemotherapy results than the accepted therapeutic guidelines by considering non-clinicopathologic factors, such as patient preferences and resource availability.

Rane (2015) developed a support system with the main goal of predicting prevailing diseases at early stages. The system was designed considering several learning techniques, including Decision Trees, Naive Bayes, Multilayer Perception Neural Network, K-Nearest Neighbors, and Support Vector Machine. All methods were implemented using WEKA (Hall et al., 2009), a well-known machine learning software. Regarding the experimental analysis, 316 patient data sets were collected from expert physicians. Each data set contained 34 attributes with the most relevant patient information. The information was divided into two categories: the training data with 190 records and the testing data with 126 records. A ten-fold cross validation was applied to evaluate the accuracy of the proposed system. Additionally, the Mean Absolute Error (MAE), which considers the difference between the diagnosis-predicted values and the diagnosis-observed values was analyzed. The results showed that all methods had accurate prediction capabilities (accuracy 99%) and the Neural Network obtained the minimum MAE (0.0029) among all learning techniques studied.

A temporal latent embedding model was developed by Esteban et al. (2015) in order to predict clinical events regarding chronic kidney diseases. In their work, the authors studied a Markov model to represent the most recent clinical information of each patient. At the same time, a Multilayer Perceptron Neural Network analyzed the full history of patients and assigned major relevance to recent information in order to predict future events. The neural network was trained using 100.000 events of patients from Charité Hospital of Berlin, which is a reference center in Europe for kidney diseases. In addition, 33.000 events were considered for testing purposes. The proposed model was compared with other learning approaches, such as Nearest Neighbor methods, Naive Bayes classifiers and Logistic Regression models by using the area under the Precision-Recall curve (AUPRC) (Boyd et al., 2013). The experimental results demonstrated the capability of the model for predicting kidney diseases events (AUPRC of 0.63), outperforming all other studied learning methods.

Subirats et al. (2014) proposed a framework that integrates formal semantics and clinical rules from different sources to provide specific recommenda-

tions for cardiac rehabilitation processes. The proposed framework integrates Decision Trees and relevant rules obtained from literature. The rules are used for providing prognosis and can customize patient sessions and rehabilitation treatments. Whereas, time responses of the rehabilitation process are predicted by Decision Trees, which are implemented using WEKA. The proposed system was evaluated using an anonymized database, containing information of 200 patients who underwent cardiac rehabilitation at Hospital Universitario Ramón y Cajal, in Madrid, Spain. The results showed a strong correspondence between treatment recommendations and real medical practices. The proposed framework could be improved by including dynamic rules instead of static ones.

A decision support system was proposed by Guidi et al. (2014) to improve the assistance of patients with heart failure, considering the importance of cardiac problems. The proposed system implements several machine learning techniques including Support Vector Machine, Neural Network, Fuzzy-Genetic algorithm, Classification and Regression Tree, and Random Forest algorithms. All methods were compared to identify the best technique for predicting the heart failure type and the level of heart failure severity. The learning approaches were trained using an anonymized database provided by the Cardiology Department of St. Maria Nuova Hospital, in Florence, Italy. The database contains 136 records with information of patients with heart failure from 2001–2008. A ten-fold cross validation was used to compare the accuracy of all methods. The results showed that Classification and Regression Tree was the most adequate algorithm for helping patients with heart failure. It achieved a cross-validation accuracy of 87.6% in heart failure type predictions and 81.8% in heart failure severity; it also provided the best human-readable guidelines.

Chang et al. (2016) proposed six guidelines for designing an efficient CDSS that simplifies the physician mental effort during medical consultations. The proposed guidelines are based on a cognitive theory that considers the physician mental model (Vessey, 1991). The authors developed a prototype of CDSS following their guidelines to demonstrate the feasibility of the approach. The system was designed conforming to flexible clinical workflows and following the Subjective data, Objective data, Assessment and Plan (SOAP) clinical model (Lin et al., 2013) to fit with physician mental processes. Eight neurology specialists from Kaohsiung Medical University Hospital in Taiwan evalu-

ated the CDSS by working with twelve clinical test cases. Specific metrics of cognitive effort defined by Wang and Benbasat (2009) were taken into account to evaluate the usability of the proposed system. The results showed that the CDSS prototype allows time savings of 30%–50% during a typical medical visit, and it also reduces the effort of physicians mental processes.

2.3. Praxis

Praxis Electronic Medical Records is a software for electronic medical records, developed to streamline the entry of clinical data and improve medical practice (Infor-med, 2021). Unlike other applications for capturing data and medical information, Praxis emulates the processes that physicians follow when they are recording clinical information. The software uses previously entered information to offer recommendations for registering a new consultation, according to the past practice of the physician user.

Each time a physician treats a new patient, Praxis uses the information recorded in previous consultations of the same physician to recommend a set of cases similar to the one being evaluated. The recommendation method is based on the principle that medicine is an individual practice, and every physician (regardless of his specialty) encounters a common pattern regarding the frequency of treated cases, and some cases are more frequent than others.

After the physician identifies the most similar case that fits the patient being evaluated, he can include the information of the current case. In general, small modifications are made over Praxis recommendations, in order to record accurate information about the new consultation. By modifying the most similar case, the physician immediately creates a new case, which is stored in the Praxis database and later may be used for other future cases.

The accumulation of clinical practices in the system allows physicians to quickly register new cases, reducing writing and reading times. This method of keeping medical records can also be used as a checklist, aiming at not forgetting important aspects that the physicians should verify with their patients, according to their own criteria.

Praxis has been implemented empirically and has been gradually improved over more than 25 years. The system also has been developed to fit with the North American medical system. Praxis has an extensive user guide, which helps physicians reduce the learning curve. However, there is no accurate

description of the software implementation. Praxis software was studied in depth in this Master Thesis and its main features are presented in chapter 3.

As all other reviewed works, Praxis has been developed for helping physicians give better assistance during medical activities. However, while other works are focused on analyzing diverse clinical data to make recommendations for treatments and diagnoses, Praxis produces recommendations that are based only on information recorded by the physician who uses the system. Praxis also focuses on speeding up writing times, instead of being an expert system for recommending diagnoses and treatments.

The works reviewed in the literature consider past patient information in order to present recommendations or predict specific diseases. The diagnoses-treatments recommendations and the predictions for specific diseases of related works reviewed could be integrated into Praxis to improve the quality of the current version.

2.4. Summary

The analysis of related works allowed identifying several proposals applying computational intelligence and other learning-based methods for diverse health scenarios. Most existing systems focuses on providing suggestions for treatments and diagnoses, based on similarity metrics regarding relevant information from past medical assistance. Reviewed works are able to identify similar clinical cases in order to provide suggestions for diagnoses, prognosis, and treatments. Moreover, they contribute to reducing error-prone steps of clinical processes. The approach presented in this Master Thesis contributes to this line of research, including specific differences with existing related works: it supports non-structured free text information to be used in the learning process, instead of just structured information Wang et al. (2015); a more effective learning approach is applied, which outperforms a Bayesian learning method such as the ones that have been previously used in the related literature Klann et al. (2014); suggestions are generated considering all similar case types (of different patients), instead of just previous information of the same patient Nakai et al. (2016); and it does not rely on complex rules based on natural language processing, which limits the applicability of other suggestion systems Barbantan et al. (2016).

Table 2.1 outlines the works reviewed in this chapter about diagnoses-treatments recommendations, designed for general health scenarios. The table shows the most relevant features of the works, and it also presents the methods applied. Additionally, Table 2.2 shows the same information regarding works for predict or detect specific clinical conditions.

Table 2.1: Summary of related works about diagnoses-treatments suggestions.

Author(s) (year)	**Method(s)**	**Relevant features**
Wang et al. (2015)	Ad-hoc patient similarity algorithm.	Menu with inferred recommendations. Real-time feedback.
Klann et al. (2014)	Specific Bayesian Networks. Greedy Equivalence Search.	Suggest initial drafts. Reduce workload of physicians.
Nakai et al. (2016)	Linear Support Vector Machine.	Use information from previous practices. High precision for usual cases.
Barbantan et al. (2016)	Natural Language Processing. Support Vector Machine classifier.	Medical structured-related concept model. Detect patterns about patient health.
Shen et al. (2015)	Language analysis. An ad-hoc matching method.	Suggest diagnoses, prognosis and treatments.
Installé et al. (2014)	Preprocessing component. Machine-learning techniques.	Reduce error-prone steps during diagnostics. User-friendliness.
Zieba (2014)	Cost-Sensitive Support Vector Machine.	Web services with learning capabilities. Generate decision rules.
Benmimoune et al. (2015)	Rules for generical cases. Case-Based Reasoning component.	Adaptive questionnary according patient profile.
Wilk et al. (2017)	Actionable Graphs. First-Order Logic.	Clinical guidelines for multimorbidity conditions. Considers patient preferences.

Table 2.2: Summary of related works for predict or detect specific conditions.

Author(s) (year)	**Method(s)**	**Relevant features**
Lin et al. (2016)	Support Vector Machine, Bayesian Classifier, Decision Trees, Multivariate Logistic Regression, Nearest Neighbors, Ripple Down Rules.	Cancer therapies assistance. Consider non clinicopathologic factors.
Rane (2015)	Decision Trees, Support Vector Machine, Multilayer Perception Neuronal Network, Naive Bayes, K-Nearest Neighbor.	Predict prevailing diseases at early stages.
Esteban et al. (2015)	Markov model. Multilayer Perceptron Neural Network.	Predict chronic kidney diseases.
Subirats et al. (2014)	Formal semantics and clinical rules. Decision Trees.	Recommendations for cardiac rehabilitations.
Guidi et al. (2014)	Support Vector Machine, Fuzzy-Genetic, Neural Network, Random Forest, Classification and Regression Tree.	Assistance for heart failure.
Chang et al. (2016)	Guidelines for designing an efficient CDSS.	High time savings for typical medical visits. Reduces physician cognitive effort.

This chapter highlighted works in which machine learning methods were applied for general health scenarios. Those works were mainly focused on providing suggestions for treatments and diagnoses. The chapter also presented works that achieved accurate results for specific clinical conditions. From a different point of view, Praxis product was introduced as a software that incrementally builds a clinical knowledge database, in order to re-use previous entered information, for reducing physicians writing time.

Chapter 3

Praxis Electronic Medical Records

This chapter presents the main features of Praxis EMR software. In section 3.1 the Praxis approach is introduced. Section 3.2 details the main entities used by Praxis to model clinical knowledge. In section 3.3 the principal characteristics of Praxis recording methodology are explained. Section 3.4 presents additional tools to assist physicians during consultations. Finally, section 3.5 shows a summary of the relevant benefits of using Praxis software.

3.1. Praxis approach

Praxis is a software for electronic medical records, developed to streamline the entry of clinical data and improve medical practice. The first version of Praxis was developed as early as 1992 by Infor-med INC. Currently the company is located in Argentina from where the product is supported and enhanced exclusively for the Canadian and US markets.

Unlike current medical record products that are based on templates to complete the task of creating medical records, Praxis allows the physician to create new records by using information that he has previously written. This concept reflects the reality that no two physicians practice medicine in the same way. Each physician has an individualized way of reaching a diagnostic hypothesis and of treating patients. In addition to the fact that the cases seen follow a specific prevalence mix, associated with the office and specialization of the physician.

To start a consultation, the physician usually relies on a similar previous case, which is what happens mentally in medicine. The physician can modify the initially postulated case to record accurate information about the new consultation. By modifying a similar case, the physician eventually creates a new case, which may be used for other future consultation meetings.

Acting according to previous experiences is a very natural way for physicians to think and refer to cases: "heavy weight housewife concerned about ecology" or "nerd male adolescent". The freedom with which the physician grasps a consultation visit clashes with the template approach, where a long list of sometimes irrelevant variables are collected in series.

3.1.1. Patient medical records and physician clinical knowledge

Praxis is designed to create problem-oriented medical records (Salmon et al., 1996). The software uses a variation of the traditional SOAP clinical model to record insights during a medical consultation, dividing the consultation information into different components.

Praxis makes a distinction between information recorded for each patient (patient history), and information that reflects the physician deliberations on clinical cases during a visit. This distinction is not possible when filling-in simple templates, where only predefined values are expected to be recorded.

Praxis builds a Clinical Knowledge Base (CKB) for each physician, containing Case Types (CTs) that the physician has recorded. This knowledge base is completely independent from patient medical records. In other words, Praxis accumulate the clinical knowledge of each physician independently of the patients that the physician has evaluated.

After using a CT, all changes made are saved not only in the patient medical record but also in the CKB. Therefore, the physician can subsequently use that CT again with other patients. Two data sets are updated at once: the patient medical record and the physician clinical knowledge, which is enriched every time a new CT is recorded.

Figure 3.1 shows an example of a physician working independently, who sees patients at his own practice. Meanwhile, Figure 3.2 shows the context of several physicians working at the same clinic.

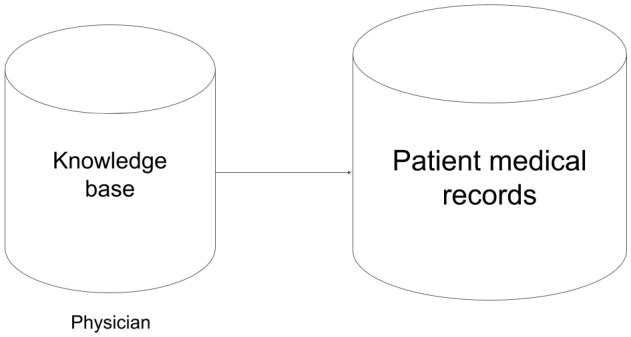


Figure 3.1: Physician model, representing a physician working independently.

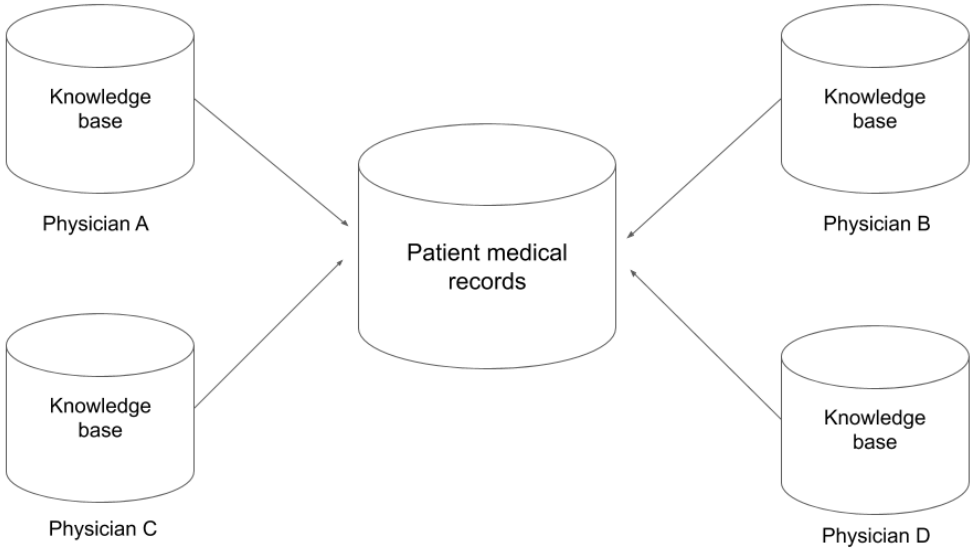


Figure 3.2: Clinic model, representing several physicians working at the same clinic.

As a natural feature, Praxis allows physicians to exchange clinical knowledge. The exchange of medical knowledge allows physicians to strengthen their knowledge base by importing CTs that have been created by other physicians. To import a CT of other physician, an authorization step is required.

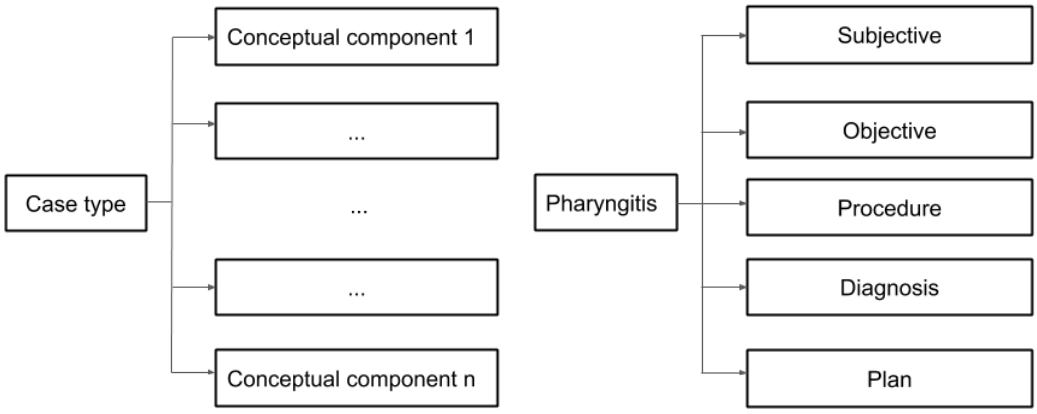
3.2. Modeling clinical knowledge

Praxis builds up a CKB for each physician, by accumulating the physician CTs. The CTs reflect the way the physician reasons, describing different situations that arise when evaluating patients. This knowledge base is assembled

gradually, as the physician is faced to new CTs and records them. Following the SOAP clinical model, a CT includes not only the subjective and objective data that stem from the patient, but also the physician interpretation and decisions taken to treat.

3.2.1. Representing a case type

Since a CT it is a broad concept, several entities are used to model the knowledge it represents. For every CT, one or more Conceptual Components (CCs) are recorded. For illustrative purposes, Figure 3.3 shows a generic structure used to represent different CCs of a CT. It also show the components of a pharyngitis case type, presented as a case study.

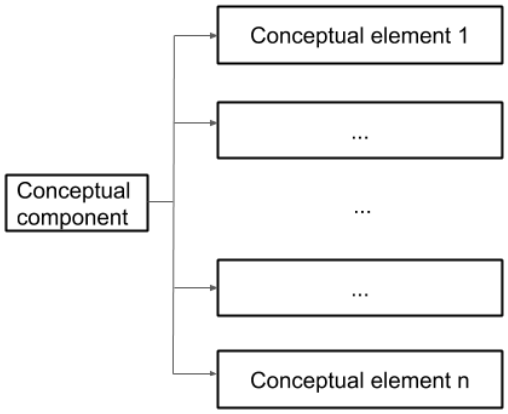


(a) Case type - Conceptual component. **(b)** Example.

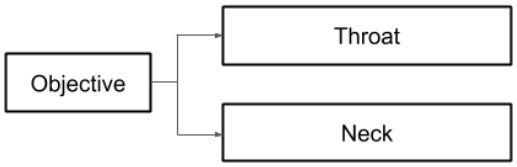
Figure 3.3: Conceptual components of a case type.

A CT is composed of different CCs. Likewise, each CC consist of different Conceptual Elements (CEs). Figure 3.4 shows a structure representing different CEs of a CC. In addition, an example of an objective component with two CEs is shown, representing different anatomical regions.

Refining one step further, a CE can be subdivided into different Units of Thought (UTs), as defined by Low (2015). Figure 3.5 illustrates a structure representing the UTs of a CE. It also shows an example with UTs describing a specific throat condition, in which the physician has observed white patches and enlarged tonsils after examining a patient.

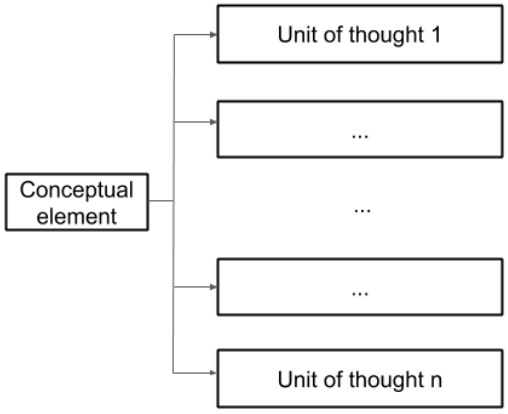


(a) A conceptual component with several conceptual elements.

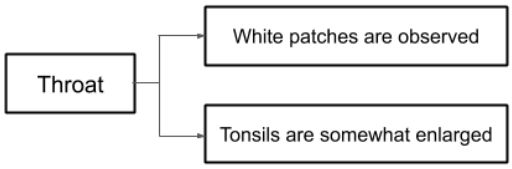


(b) Example.

Figure 3.4: Conceptual elements of a conceptual component.



(a) A conceptual element with several units of thought.



(b) Example.

Figure 3.5: Units of thought of a conceptual element.

General structure of a case type

A tree like structure is used to model a clinical CT, which includes in increasing hierarchy: unit of thought, conceptual element, and conceptual component.

The hierarchical division of a case type is presented in Figure 3.6, showing that several units of thought are grouped into conceptual elements to determine the conceptual components of a CT. In addition, Figure 3.7 shows the general structure to represent a CT instance.

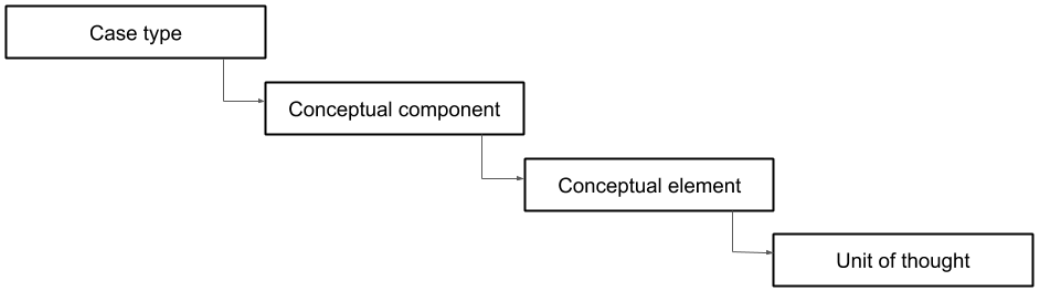


Figure 3.6: Hierarchical division of a case type.

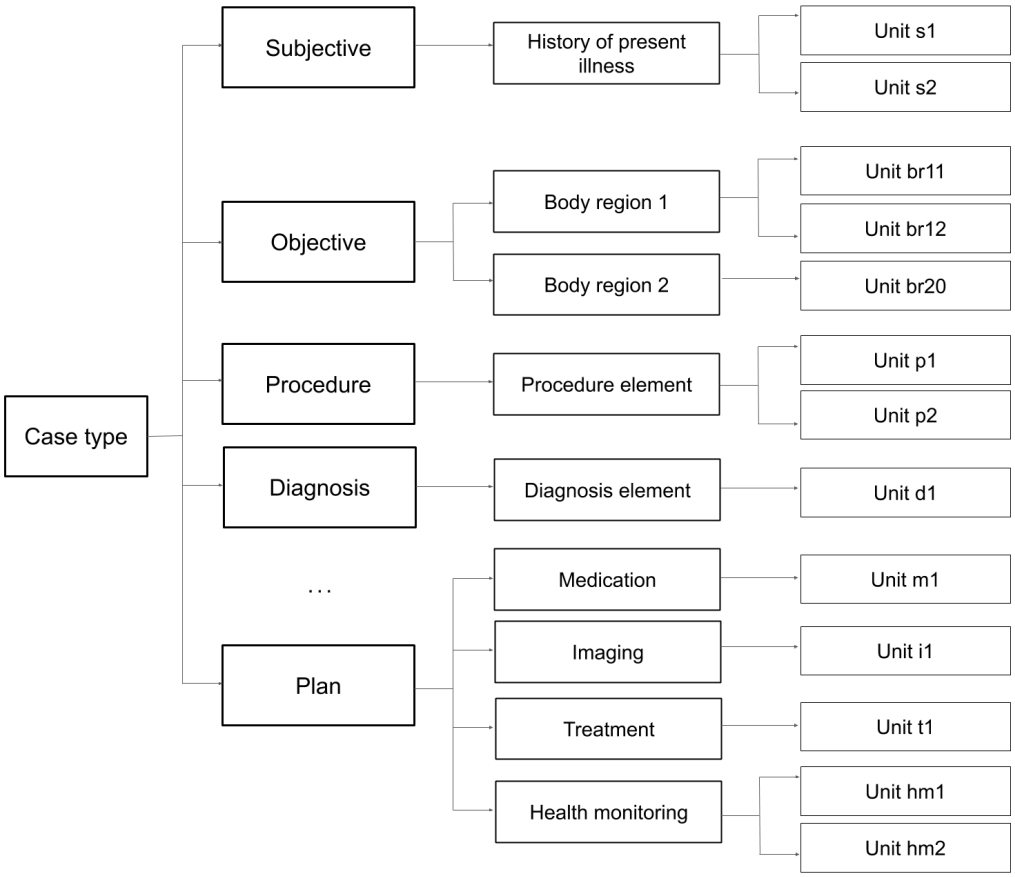


Figure 3.7: General structure to represent a case type instance.

3.2.2. Units of thought

When recording a specific CT, different UTs are saved, which later can be reused when recording new visits. In this way, previously written texts can be quickly and efficiently reused in order to write new CTs.

A UT is a statement that describes a basic clinical idea. Examples of UTs are:

- "The patient has a sore throat lasting 4 days."
- "The patient denies any nasal drainage."
- "She has a fever lasting 4 days."
- "The patient complains of severe pain lasting 4 days."

Characteristics of UTs

Several characteristics need to be modeled to give real meaning to a UT in a clinical context. The main characteristics of UTs are: randomness, variation, gender, and datum.

Randomness: It is typical for many clinical statements to include some degree of randomness. Altering a specific value of the UTs statement does not alter the meaning of the UT. Examples of UTs that contain a random part are:

- "The patient has a sore throat lasting [6 days]."
- "The patient has a sore throat lasting [5 days]."
- "The patient has a sore throat lasting [1 week]."

Variation: There are semi-random variations that do not change the meaning of a UT. An example of UT that contain a semi-random part is:

- "The patient complains of [mild / severe] pain lasting 4 days."

Gender: Expressing different genders does not change the meaning of a UT. Examples of UTs that contain gender information are:

- "masculine" $\leftrightarrow$ "feminine", "man" $\leftrightarrow$ "woman", "boy" $\leftrightarrow$ "girl", "gentleman" $\leftrightarrow$ "lady"
- "The boy is experiencing normal development." $\leftrightarrow$ "The girl is experiencing normal development."

Datum: This element allows discrete data to be embedded within a free text. By using *Datum*, the idea represented by the UT also remains unchanged. *Datum* is an object that can be embedded within any part of a free text, in order to refer to structured information. An example of UT that contain a datum is:

- "<*patient.firstName*> is a person who is <*patient.age*> years old, who suffers..."

To represent a UT, it is necessary to create a structure capable of storing free text, as well as referring to data that allows a level of randomness or semi-randomness. In addition, it must support the incorporation of discrete data through the use of Datum.

3.2.3. Characteristics of conceptual elements

A CE consist of one or more UTs that are grouped to create a broader description. Each CE has its own characteristics that determine its features.

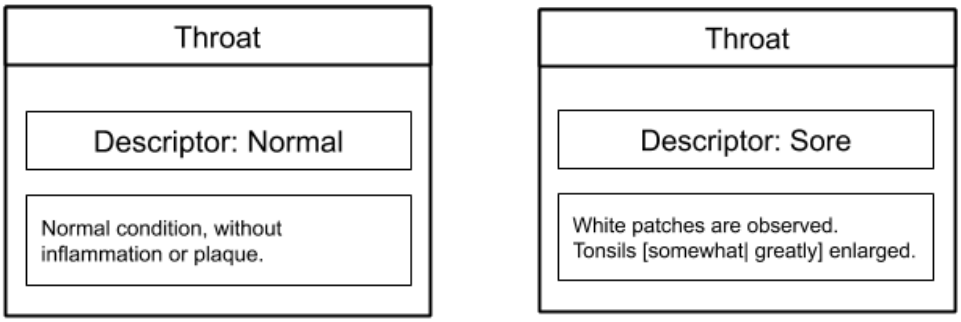
Default display of units

Each CE has an attribute indicating the default display mode of its UTs. A CE can show all its UTs by default, or on the contrary it can opaque all its units. For example, a subjective data element which describes patient symptoms, displays its UTs as inactive. Subsequently, the physician only activates symptoms that apply to the patient being evaluated. On the other hand, other CEs such as pharmacological indications are displayed as active, assuming that they apply to the consultation being evaluated.

Name and descriptors

A physician can assign a name to a CE for grouping clinical phrases under the same identifier. Thus, when the physician wants to search for a CE that has already been defined, the search can be simplified using only its name. An identification by name is especially useful for labeling objective elements that give information about a patient anatomical regions. Additionally, Praxis suggests to create descriptors for detailing different conditions of the same element. It suggests to define a "normal" descriptor for UTs describing the regular condition of a certain anatomical region. On the other hand, to describe

the abnormal conditions, it suggests using a descriptor that is mnemonic to the UTs. Figure 3.8 shows two examples for representing CEs that describe conditions of an anatomical region. One example details a throat in a normal condition and the other specifies a patient with a sore throat.



(a) Conceptual element, normal throat. **(b)** Conceptual element, sore throat.

Figure 3.8: Units of thought of a conceptual element.

Chronic information

CEs also have an attribute that indicates whether the element refers to the chronic information of a patient. If a CE is marked as a patient chronic information, later it will appear on all future patient consultations. Specifically, this attribute is used to identify CEs that describe anatomical regions of patients suffering a chronic condition.

3.2.4. Characteristics of units of thought

The attributes of the UTs model specific contexts that may arise during the documentation of a medical consultation.

Information unique to a consultation

In general, UTs described during a consultation are associated with the CT determined by the physician. However, in order to describe any real patient situation, it is necessary to consider the unique aspects that may occur during a specific consultation. The physician must be able to define new UTs with the ability to describe information unique to a specific consultation. When the physician completes the consultation and saves information, the UTs defined as

unique to the consultation will not influence the CT, which means the clinical knowledge base will not be updated. However, these UTs will be relevant for writing the patient clinical record, by describing the unique aspects of the consultation meeting.

Information unique to a patient

There are situations where it is necessary to give detailed information about a person. For these situations, it is key to build UTs that can represent information unique to a patient. The UTs representing specific information about a patient will not be associated to any CT, since they contain data specific to a particular patient. For example, when specifying a patient personal background, the physician can create UTs that are associated with the patient being evaluated. A UT describing a personal background does not have any relationship with the CT that the physician has chosen as a basis for his evaluation. Consequently, it is necessary for UTs to have an attribute indicting whether the content refers to patient specific information.

Information exclusively for the physician

In certain situations, the physician may write text that is exclusively for his own use. Thus, this text is not saved in the clinical records of any patient. As an example, UTs that act as reminders, must have the attribute of information exclusively for the physician.

Interoperability

Each UT can be associated with standard codes of health terminology. Table 3.1 details the main associations that can be specified over UTs.

Table 3.1: Interoperability standards of health terminology.

Conceptual elements	*Terminology*	*Type of association*
Diagnoses	ICD10, SNOMET CT	Mandatory
Procedures	CPT	Suggested
Medications	Drugs	Suggested

In order to associate every UT to its corresponding terminological code, external services can be used to provide access to different terminological servers.

3.3. Recording consultation methodology

The most streamlined documentation methodology that can be applied using Praxis consists of taking advantage of an existing CT. The consultation of a new patient can be quickly recorded by making small changes over a similar previously written CT. To apply this methodology, it is necessary to start with an existing CT, which can be easily located if the physician remembers the name of a CT. Otherwise, it will be necessary to make use of features that help to find the closest CT, according to different search criteria.

To simplify the CT search based on the best matches of the current patient condition, the software waits for the physician to enter the first UT, which reflects any of the aspects that may arise during a medical consultation. Starting with an entered UT, the set of CTs containing it are determined. Subsequently, Praxis orders the different CT candidates according to frequency of use, and presents them to the physician in the form of a list. The physician may apply additional search filters to restrict criteria even further, in order to determine the most appropriate clinical CT to use as a basis for the new patient record. Figure 3.9 shows an example in which a UT is included in more than one CT.

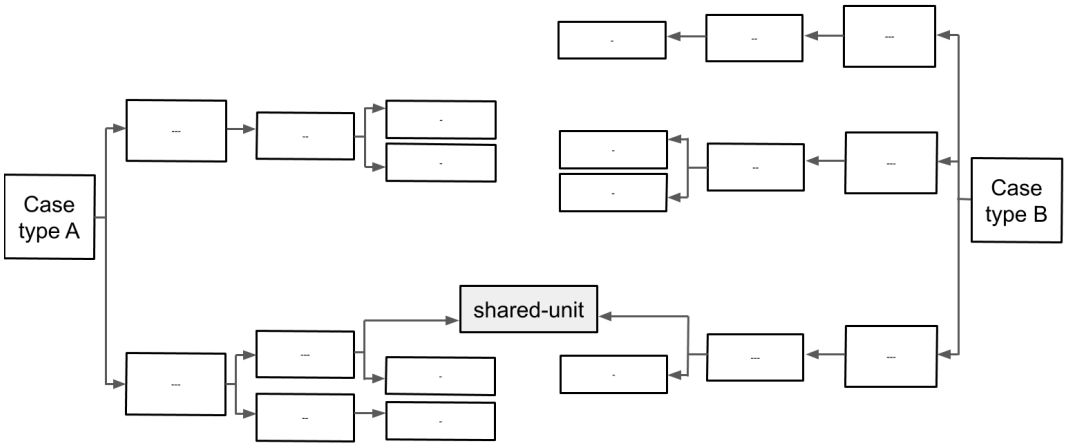


Figure 3.9: Shared unit of thought, included in more than one CT.

Just as CTs show a bell curve regarding their frequency of use, CEs and UTs also share the same bell characteristic. Bell curves occur because CEs and UTs have cases that appear much more often than others. Praxis considers the frequency of use to facilitate the search process, both for the CEs and the UTs. Every time a certain CE or UT is searched, the results are shown in descending order according to frequency of use.

Once the physician selects the CT that is the most convenient as a starting point, he can reuse the previous CT document. A previous document can be used for remembering aspects that should not be forgotten with new patients. A CT can be quickly and conveniently changed at any time the physician realizes that it is not the most appropriate, and wishes to search for another CT that matches more closely the current patient.

To complete the consultation record, the physician should save the changes made in the selected CT. When saving a new clinical document, different situations are identified:

- it is a record of a patient with an identical clinical CT.
- it is a record that has improved on the same clinical CT.
- it is a record of a new clinical CT.

An identical clinical CT is detected when none of the UTs involved have changed. Each UT is considered unchanged although its random aspects may be altered. A new CT or an improvement on an already existing CT is detected when any UT is changed. In this situation, the physician must specify if he is documenting a new CT or improving an existing one.

Finally, before saving information about the patient consultation, two actions are triggered. Firstly, the clinical CT is recorded or updated in the CKB. Secondly, the patient history is enhanced, saving all the information that the physician has recorded during the medical consultation.

Saving records without generating knowledge

When recording a consultation, the physician documents clinical information that describe the patient situation, and before finishing a record, the physician CKB is updated. However, in some specific cases, it is desirable to document a single insight about a patient, and Praxis offers the possibility of saving the record without generating clinical knowledge. By saving without generating clinical knowledge, the patient history will simply be enhanced by adding the information recorded by the physician, and no CT will be updated. This feature must be used sparingly, since when using it the ability to learn clinical knowledge is lost.

3.3.1. Multiple case types

Every time the physician wants to add a new CT over another one already selected, Praxis is capable of combining UTs of the selected CTs. The main rules used to combine different CTs are detailed below.

Combination of disjoint elements

If a CE is present in only one of the CTs, then it is added as an element of a multiple CT, since it cannot create any type of conflict.

Combination of common non-objective elements

This situation occurs when the same CE is present in more than one CT assigned to a patient, and the element does not describe any objective information about the patient. In this situation, two possibilities are determined, according to the default display attribute of the units present in the CE.

1. If the default display attribute is inactive. All the UTs are simply joined in order. For example, the UTs included in the subjective element of two different CTs.

2. If the default display attribute is active. In this case, all the UTs are joined and any UT considered repetitive is removed. For example, an identical pharmacological indication detailed in both CTs.

Combination of common objective elements

Praxis implements a more complex technique for combining CEs included in the objective component of a medical consultation. These CEs describe objective information about a patient anatomical regions. In this situation, conflicts may arise if UTs are automatically combined for different CTs. Two options are determined: automatic combination and conflicting combination.

1. Automatic combination: If the CE representing an anatomical region is described as normal in one CT and abnormal in another, then the element with the abnormal descriptor attribute is assumed to be the most relevant, and the element with the normal descriptor attribute is discarded.

2. Conflicting combination: A conflicting combination occurs when the same CE has different abnormal descriptors in more than one CT. In this situation, Praxis is not capable of resolving the conflict, and asks the physician for a solution.

Combining CTs

After combining several CTs to record a patient consultation, the physician may still add new UTs that detail more information about the consultation. Praxis includes features designed to ease the entry of more than one CT during the same medical consultation. In particular, it provide an option to restart the search for another CT, and lets the physician specify any UT to help finding the next CT. For the specific situation of Multiple Case Types (MCTs), Praxis provides a feature that lets the physician consolidate the UTs from one CT to another. By consolidating one CT with another, the first CT selected is changed by adding all the UTs from the second CT.

3.3.2. Partial case types

A Partial Case Type (PCT) is created when no diagnosis is specified. PCTs are especially useful when the physician does not reach a diagnosis in the first minutes of a consultation. For every situation in which a diagnosis is not reached, the physician can create a PCT with the clinical information that he can identify. In general, the patient subjective elements (what the patient says he feels) are recorded. Moreover, other elements can be detailed, such as an objective element that arises from a physical examination.

PCTs are highlighted every time they are displayed. This distinction allows the physician to clearly identify the partial nature of a PCT, which makes it extremely practical when the physician wants to search for a particular PCT. By using a color distinction, it is visually easy to identify which CT are partial and which are not. Like other CTs, all recorded information becomes available after selecting the PCT and can be reused with future patients.

An example of a PCT would be a patient whom the physician only knows is having a bad cough. At first, no other information is available that allows the physician to conclude a differential diagnosis. In this situation, it is convenient for the physician to rely on a previous record for a case type called “Patient with a bad cough”. Clearly the PCT “Patient with a bad cough” will not have

any diagnosis associated with it. However, it will present a set of symptoms, physical examinations or other previously recorded elements related to a patient with a bad cough. Therefore, every time the physician cannot define a diagnosis, he can search for the PCT closest to the situation of the patient being evaluated, or in the absence of one, create a new PCT.

After analyzing the patient and performing the relevant physical examinations, all improvements should be saved on the created or edited PCT. Later, the physician can complete the consultation as usual, reaching a specific diagnosis.

3.3.3. Chronic conditions

Praxis assists on improving the consultation registry of a patient suffering from a chronic condition. In order to facilitate physician work, several features are included for monitoring chronic conditions.

Health monitoring

A *Health monitoring* utility allows physicians to schedule the frequency that a UT appears in a CT. Figure 3.10 shows the main aspects of Health monitoring utility.

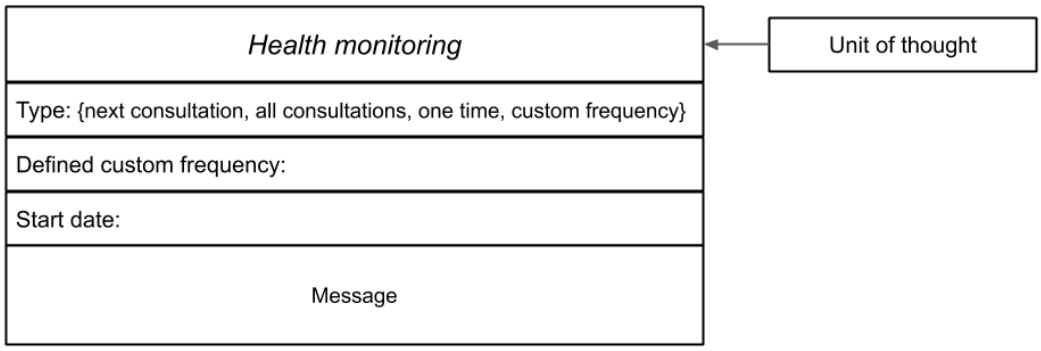


Figure 3.10: Features of Health monitoring utility.

As an example of how to use Health monitoring utility, the physician could schedule a UT for every patient with the case type "Mellitus diabetes" indicating an ophthalmologist review once a year. Health monitoring utility could also be used to schedule the frequency of other UTs, such as a pharmacological prescription or a specific procedure.

Chronic anatomical regions

Every CE that represents an anatomical region may be marked as a chronic condition of a patient. If an element is marked as chronic, it will be permanently associated with a patient. This chronic mark indicates that every time the physician refers to the chronic anatomical region of the patient, Praxis will recover all the chronic information that was previously detailed for the same anatomical region.

Chronic case types

During the workday, a physician usually cares for a great number of patients with chronic conditions. In order to address the existing specifications for chronic patients, the physician can create a CT and then mark it as chronic.

After a CT is marked as chronic, it immediately acquires a dual identity. The dual identity is the main feature of a Chronic Case Type (CCT). Every CCT has two identities, one that applies when the CCT is associated with a patient for the first time, and the other for the same patient later consultations. The main characteristics of each one of these CCT identities will be detailed as follows.

First identity: When a CCT is first associated with a patient, few differences exist between it and any other case type. Once the physician chooses the CCT, he will be able to reuse all the conceptual elements of the selected case type, including the elements describing patient current illness, physical examinations, indicated procedures, prescribed medicines, etc. On the other hand, as a unique characteristic, every CCT is permanently associated with the patient. Therefore, after a CCT is first associated with a patient, it will be suggested every time physician records a consultation for the same patient.

Second identity: The identity of a CCT changes significantly after the case type is first used in a patient. After a conditions is identified as chronic, it is extremely important to monitor its evolution over time. The second identity of a CCT helps to record relevant information about the evolution of a chronic condition. When a CCT is expressed according to its second identity, a new component called *Evolution* emerges. Under this new component, the physician must specify the UTs that reflect the evolution of the patient chronic condition. From the moment the CCT acquires its second identity (for a given patient), Praxis will no longer present the CE describing the patient current

illness. It will be assumed that monitoring of the chronic condition will be recorded using UTs described in the *Evolution* component. The second identity of a CCT has other key characteristics. Specifically, the second identity does not inherit the conceptual elements that the physician has defined for the CCT first identity. Therefore, when the physician uses the CCT second identity for the first time, he will have to specify the UTs to plan the monitoring of the patient chronic condition evolution. Figure 3.11 gives an example of a CCT in its second identity.

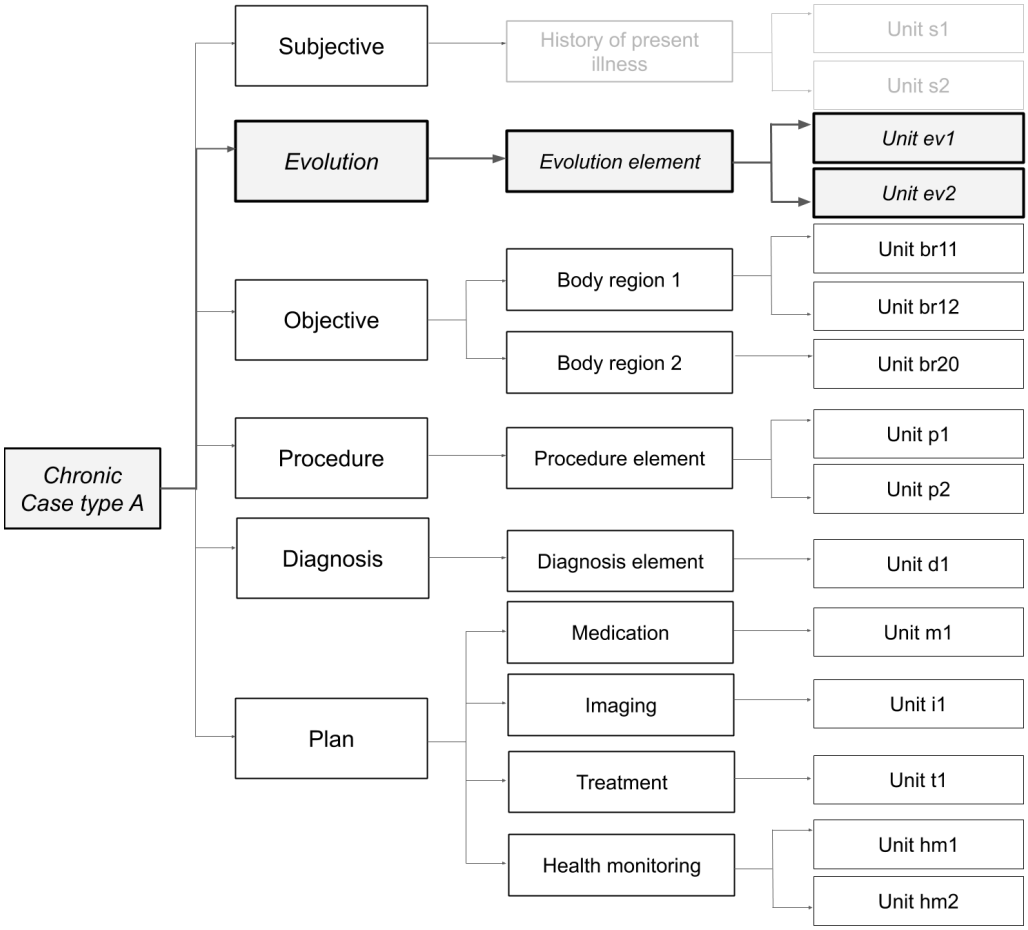


Figure 3.11: Chronic case type example.

The CE that describes the patient current illness is shown as inactive in Figure 3.11, since it is a characteristic of the first identity of the CCT. In contrast, the figure highlights UTs that reflect the evolution of a chronic condition, and are exclusive to the second identity of the CCT.

Definition of a new chronic case type

After building a new CT, it is possible to mark it as chronic. A CT marked as chronic acquires a dual identity and a highlighted color presentation. A color distinction is helpful when a physician wants to locate a specific CCT to use with a patient for the first time. When a CCT is associated with a certain patient, every anatomical region that has been defined within the CT becomes a chronic region for the patient, and Praxis will continue to consider it in later consultations of the same patient.

When working with the second identity of a CCT, the physician must specify the UTs that describe the monitoring planned for the patient chronic condition. In addition, it may occur that after planning a specific monitoring program, the physician wants to make a change to correct or improve the plan. Consequently, the physician always can specify new UTs that improve the basis of monitoring the chronic condition evolution.

In the context of the CCT second identity, an important feature is available when specifying a new UT. When a physician adds a new UT, *Health monitoring* utility is immediately displayed. In that moment, the physician must create a new item that determines the frequency of the recently specified UT. As an example, the physician could plan a monitoring program that includes a certain medicine every two months, an annual appointment with an ophthalmologist and a blood test every six months. In this example, the physician should specify three UTs with their respective monitoring items, defining the frequency of each indication. Like any other CT, when recording chronic conditions, it is quite valuable to detail a large set of controls or reminders, since same elements will be displayed for every patient with the same CCT.

3.4. Consultation assistant tools

Praxis provides tools to assist physicians during consultations. The *Usage reminder* and *Simple messaging* tools are introduced as important features for helping patient assistance.

3.4.1. Usage reminder

A *Usage reminder* is a tool that contributes to improving the quality of medical records, using guides and timely reminders that are displayed to the

physician and are relevant to the CT being evaluated. For every *Usage reminder* instance, it is necessary to define the criteria in which the reminder will be displayed to the physician. These criteria are specified by times, conditions and situations in which the reminder can guide, remind or make recommendations for the physician, to better address the consultation being evaluated. *Usage reminders* are also quite useful in contexts where care goals are used.

Figure 3.12 present the main parts of the structure necessary for implementing usage reminders.

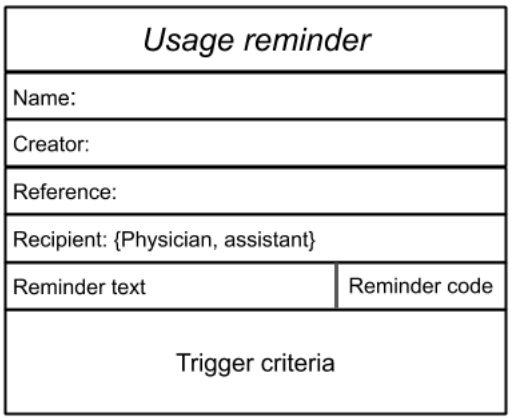


Figure 3.12: Features of Usage reminder tool.

To build a *Usage reminder* it is necessary to specify a trigger criteria, which can be simple or highly complex.

3.4.2. Simple messaging

Simple messaging is a communication tool that allows to send the patient information to different members of the health team, who are involved in the patient care. It is a messaging tool that allows the transmission of specific information about a patient, to notify others about certain clinical aspects.

A physician can create scheduled messages by using *Simple messaging* tool. This messaging tool lets the physician write a message and later decide if he wants to send it immediately, or prefers to specify the moment that the message will be sent to the recipient. The moments that can be defined as sending conditions are: immediately, a specific date, at next consultation, or at next time the clinical record of the patient is accessed.

A physician that work with the help of receptionists can define scheduled

messages for his helpers. Physician has features to assign tasks that receptionist must perform after the patient consultation is completed, such as coordinating a new medical appointment. Given that the physician is also one of the health team members, he can create messages and schedule to receive them as well. A physician can reuse information that has already been written in a previous message. Consequently, to create new message, it is convenient to use another similar message as a starting point. Like a CE, a message naturally has the ability to be associated with CTs.

3.5. Benefits

The accumulation of CTs in the CKB allows a physician to quickly register new consultations, reducing writing and reading times. This method of keeping medical records can also be used as a checklist, aiming at not forgetting important aspects that the physicians should verify with their patients, according to their own criteria.

3.5.1. Benefits of PCT, MCT, CCT

To determine a differential diagnosis, a physician can be supported by his own PCTs, regardless of the uniqueness or complexity of the patient situation. In other scenarios, when patients do not arrive at a consultation with only one type of problem, a physician may conclude multiple diagnoses for a patient during the same consultation. For these situations, a physician can record a consultation based on MCTs, and several rules are used to accurate combine different case types.

Given the high frequency of chronic patients, it is especially relevant to consider their singularities in order to streamline the recording of chronic conditions. When a patient suffers a chronic condition, the physician has no trouble finding a diagnosis that describes the patient health condition. However, the physician must be concerned with asking certain questions, verifying specific anatomical regions and performing indications that reoccur in the patient chronic condition. When a physician uses a CCT, an immediate benefit is that it reminds him of reoccurring aspects, which should be verified with the patient being evaluated, and also apply to any other patient with the same chronic condition.

3.5.2. Health monitoring

The *Health monitoring* utility allows a physician to plan the occurrence of a UT according to a certain frequency. If the *Health monitoring* tool did not exist, a UT could only be determined as present or not present in a CT. It would not model a dynamic property that considers the variable occurrence of a UT over time, given a certain CT.

3.5.3. Messaging and reminders

The physician can use the message tool to be reminded of certain actions he must complete for a patient in the future. By using a message in a CT, the same message is displayed each time the physician evaluates another patient with the same CT.

Care goals demand physicians to detail additional information when treating certain patients with specific conditions. In this context, a reminder may be used in a timely manner, to remind the physician not forgetting to record the required data for fulfilling care goals.

3.5.4. Interoperability

Although Praxis places a strong focus on product usability, aiming at providing the best approach to document clinical records, it does not mean that disregards other key requirements of health context. Such is the case of the interoperability of information, which has a major relevance in the health area and Medical Informatics specifically.

3.5.5. Consultation accuracy

Given the ability to reuse clinical information, re-writing texts is avoided, since the same insights have already been recorded for other patients. After a reasonable amount of time, Praxis will be able to anticipate a great part of the physician thinking regarding reoccurring cases. All support is based on the clinical knowledge learned from the physician user. As a result, the physician can use his previous records as a checklist, to avoid forgetting certain questions and to perform all the tasks and indications relevant to the patient condition. Consequently, Praxis behaves as a clinical tool that helps physician practice better medicine, and achieve quicker documentation of each consultation.

Chapter 4

Clinical knowledge model to follow physician reasoning

This chapter presents a formal model proposed for representing clinical knowledge and patient history, including medical records. Specific flows for address medical consultations scenarios are also presented. The chapter is organized as follows. Section 4.1 presents all entities which model the clinical knowledge of physicians. A data structure representing the most relevant information of each patient is introduced in Section 4.2. Finally, Section 4.3 presents relevant flows that can occur during medical consultations.

4.1. Clinical knowledge base

A bottom-up modeling approach is used to present the proposed clinical knowledge model. Several entities are defined in order to specify a clinical knowledge base which describes information of real medical case types. All entities included in a clinical knowledge base are described in the following subsections.

4.1.1. Unit of thought

As defined by Low (2015), a unit of thought is a statement that describes a basic clinical idea. Let UT^M a unit of thought registered by physician M. UT^M is denoted as $UT^M = <ptext, uqcn, uqpt, exph, terms, inuse, ctSchedule, M>$, where *ptext* denotes a string capable of containing structured or random data, *uqcn* indicates if the unit refers to information to be used only in a unique

consultation, *uqpt* indicates if the unit refers to unique information of a specific patient, *exph* indicates if the unit contains exclusive data for physician use, *terms* detail associations with health terminological standards, *inuse* denotes if the unit is in use during a consultation, and *ctSchedule* indicates the frequency that a unit appears in a case type. A unit of thought used in a case type will reappear each time the case type is used, unless a specific frequency is defined by its *ctSchedule* attribute.

The set of all units of thought registered by physician M is denoted as UT_T^M. Let $UT_1^M = <ptext_1,\ uqcn_1,\ uqpt_1,\ exph_1,\ terms_1,\ inuse_1,\ ctSchedule_1,\ M>$ and $UT_2^M = <ptext_2,\ uqcn_2,\ uqpt_2,\ exph_2,\ terms_2,\ inuse_2,\ ctSchedule_2,\ M>$ units of thought registered by physician M. A constraint on units of thought is defined in Eq. 4.1, implying that each basic clinical idea is represented by a unique unit of thought.

$$\left.\begin{aligned} UT_1^M \in UT_T^M \\ UT_2^M \in UT_T^M \\ equals(ptext_1, ptext_2) \end{aligned}\right\} \Rightarrow UT_1^M = UT_2^M \tag{4.1}$$

Considering that text variations do not change the meaning of a basic clinical idea, an ad-hoc function *equals* (defined in Eq. 4.2) is necessary to identify if two phrases represent the same clinical idea. The same clinical idea can be instanced containing both structured information and random data, which implies that two different text strings can represent the same clinical idea.

$$equals(ptext_1, ptext_2) = \begin{cases} true, & \text{if } ptext_1 \text{ and } ptext_2 \text{ describe the} \\ & \text{same clinical idea.} \\ false, & \text{otherwise.} \end{cases} \tag{4.2}$$

4.1.2. Conceptual element

A conceptual element is composed of a set of units of thought grouped to represent a broader concept. Several attributes are used to model all possible features of a conceptual element. Let CE^M a conceptual element registered by physician M. CE^M is denoted as $CE^M = <name,\ display,\ chron,\ setDesc>$, where *name* denotes the name of the element, *display* indicates the default display mode of its units of thought, *chron* indicates if the element refers to

a chronic condition, and *setDesc* denotes a set of possible descriptors of the conceptual element. The set $setDesc = \{[desc_1,\ subset_1(UT_T^M)],\ ...,\ [desc_k,\ subset_k(UT_T^M)]\}$ is composed by several pairs, each of one is used to model a possible option to describe a real condition of a conceptual element.

Two constraints are defined on conceptual elements. The constraint presented in Eq. 4.3 implies that a conceptual element is identified by its *name*.

$$\left.\begin{array}{r} CE_1^M =< name_1, display_1, chron_1, setDesc_1 > \\ CE_2^M =< name_2, display_2, chron_2, setDesc_2 > \\ name_1 = name_2 \end{array}\right\} \Rightarrow CE_1^M = CE_2^M \quad (4.3)$$

The constraint presented in Eq. 4.4 implies the uniqueness of each descriptor into a conceptual element. Several units of thought can be labeled under the same descriptor to define an identified sub set, describing a real condition of an element.

$$\left.\begin{array}{r} [desc_1, subset_1(UT_T^M)] \in setDesc \\ [desc_2, subset_2(UT_T^M)] \in setDesc \\ desc_1 = desc_2 \end{array}\right\} \Rightarrow subset_1(UT_T^M) = subset_2(UT_T^M) \quad (4.4)$$

4.1.3. Conceptual component

A conceptual component is composed of a set of conceptual element references, grouped to define sections of clinical information. Each conceptual component represents a typical clinical data section, in which a physician generally groups the information of a medical consultation.

Let CC^M = <*id, secType, activeElems*> a conceptual component defined by physician *M*, identified by its *id* attribute. The *secType* attribute is used to represent the type of a data section, such as physical examination, medicines, and laboratory indications. Each *secType* must belogs to *ALL-SECTION-TYPES* set, which model all possible sections of patient medical records. The set $activeElems = \{[elemName_1,\ activeDesc_1],\ ...,\ [elemName_k,\ activeDesc_k]\}$ indicates which descriptor is used for each conceptual element referenced in a conceptual component.

Two constraints are defined on conceptual components domain. The constraint presented in Eq. 4.5 implies that a conceptual component is identified

by its *id* attribute.

$$\left.\begin{array}{r} CC_1^M =< id_1, secType_1, activeElems_1 > \\ CC_2^M =< id_2, secType_2, activeElems_2 > \\ id_1 = id_2 \end{array}\right\} \Rightarrow CC_1^M = CC_2^M \tag{4.5}$$

A second constraint presented in Eq. 4.6 ensure the referential integrity of names and descriptors of the active elements, referenced from a conceptual component.

$$\left.\begin{array}{r} CC^M =< id, secType, activeElems > \\ [elemName, activeDesc] \in activeElems \end{array}\right\} \Rightarrow \begin{array}{c} \exists \text{ conceptual element } e, \\ e =< name, ..., setDesc > / \\ e.name = elemName \wedge \\ \exists \text{ d} \in setDesc, \\ d.desc = activeDesc \end{array} \tag{4.6}$$

4.1.4. Case type

Several conceptual components can be grouped by a unique name to label a complex scenario, representing a real case type that can occur during physician workday. Let CT^M a case type registered by physician *M*. CT^M is denoted as CT^M=<*name*,{CC_1^M, ..., CC_n^M}, *chron, chronicComponents*>, where *name* indicates the name of the case type, the set {CC_1^M, ..., CC_n^M} describes a specific group of conceptual components, *chron* indicates if the case type is marked as chronic, and *chronComponents* denotes all components used to monitor chronic conditions.

Three constraints are defined on case types domain. The constraint presented in Eq. 4.7 implies that a case type is identified by its *name*.

$$\left.\begin{array}{r} CT_1^M =< name_1, chron_1, comps_1, chronComponents_1 > \\ CT_2^M =< name_2, chron_2, comps_2, chronComponents_2 > \\ name_1 = name_2 \end{array}\right\} \Rightarrow CT_1^M = CT_2^M \tag{4.7}$$

A second constraints presented in Eq. 4.8 implies that each conceptual component of a case type models a different section of the clinical information.

$$\left.\begin{array}{r} CT^M =< nc, \{CC_1^M, ..., CC_n^M\}, chron, chComps > \\ CC_i^M =< id_i, secType_i, subset_i) > \\ CC_j^M =< id_j, secType_j, subset_j) > \end{array}\right\} \Rightarrow \begin{array}{c} secType_i = secType_j \\ \Updownarrow \\ i{=}j \ \forall i, j \in \{1, n\} \end{array} \quad (4.8)$$

A third constraints presented in Eq. 4.9 implies that each chronic conceptual component models a different section of chronic clinical information.

$$\left.\begin{array}{r} CT^M =< nc, comps, true, \{CC_{chron_1}^M, ..., CC_{chron_m}^M\} > \\ CC_{chron_i}^M =< id_i, secType_i, subset_i) > \\ CC_{chron_j}^M =< id_j, secType_j, subset_j) > \end{array}\right\} \Rightarrow \begin{array}{c} secType_i = secType_j \\ \Updownarrow \\ i{=}j \ \forall i, j \in \{1, m\} \end{array} \quad (4.9)$$

Finally, Eq. 4.10 defines the clinical knowledge base of a physician M, formed by all case types registered by the physician M

$$CKB^M = \bigcup_{i=1}^{n} CT_i^M \quad (4.10)$$

4.1.5. Message Agents

A physician can define message agents for helping communication and attention during clinical work. A message agent is defined as *ma = <triggerCriteria, name, sender, receiver, text, codeText>* where *triggerCriteria* specifies the criteria by which the message is triggered, *name* denotes the name of the message agent, *sender* identifies the sender of the message agent, *receiver* identifies the receiver of the message agent, *text* details the content of the message agent, and *codeText* denotes associations of *text* with health terminological standards.

Since a message agent can be triggered in several scenarios, a specific data structured *triggerCriteria* is used to represent different types of trigger events. The structure *triggerCriteria* is defined as *triggerCriteria = <caseTypes, times, condition>* where *caseTypes* denotes an associated set of case types, *times* refers to moments, and *condition* refers to general conditions of trigger.

When a *CT* is used, all message agents associated with it are triggered. Furthermore, a message agent can be trigger by a time event, or by any other condition, such as the age of a patient.

4.2. Patient representation

A data structure is used to organize the information of each patient, considering the most relevant groups of personal data sets. The proposed structure includes medical records of patient history, and it also considers the chronic information of each patient.

4.2.1. Patient structure

Each patient is modeled as P = <*personalData, MR^P, chronicElems, chronicCaseTypes*> where *personalData* denotes personal data (sush as patient and family background), MR^P denotes all medical records of the patient P, *chronicElems* indicates associations with chronic conceptual elements, and *chronicCaseTypes* indicates associations with chronic case types. The *chronicElems* set is defined as *chronicElems* = {*[elemName*$_1$*, chronDesc*$_1$*], ..., [elemName*$_j$*, chronDesc*$_j$*]*}, and it is used to remember the descriptors of the elements that describe chronic conditions of a patient. Additionally, the set *chronicCaseTypes*={*caseTypeName*$_1$*, ..., caseTypeName*$_k$} is used to remember all chronic case types associated with a specific patient P.

4.2.2. Patient medical records

The set of medical records of a patient P is denoted by MR^P and contains all records included in the medical history of the patient. A medical record of patient P created at time t is denoted as mr_t^P and it is defined as mr_t^P =<*content,p,t*>, where *content* is a set of *[phrase, unit]* pairs, each of one includes a unit of thought associated with a clinical phrase. Consequently, $MR^P = \{mr_{t_1}^P, mr_{t_2}^P, mr_{t_k}^P\}$ describe the history of a patient, containing k medical records.

Let mr_t^P = <*content,p,t*> a specific patient medical record, where *content* = {*[phrase*$_1$*, unit*$_1$*], ..., [phrase*$_n$*, unit*$_n$*]*} is composed by one or more pairs of clinical information. A function *showRecord* is used to print the content of a medical record, taking into account all phrases included in the *content* of a medical record. Function *showRecord* only prints clinical phrases, no unit of thought is showed.

4.2.3. New medical record

Let $CKB_t^M = \{CT_1^M, CT_2^M, ..., CT_n^M\}$ the composition of the clinical knowledge base of physician M at time t. A medical record mr_t^P is generated as a result of the interaction of physician M and patient P, during a consultation at time t.

Since a physician usually takes a case type CT_x^M as basis to record a specific consultation, a transformation T^* can be applied to generate a new medical record. Consequently, a record $mr_t^P = T^*(CT_x^M)$ is created taken into account the active information of a selected case type. The active information of a case type is defined by the units of thought with *inuse* attribute in true. Transformation T^*: $CKB^M \rightarrow MR^P$ is defined as $T^*(CT) = mr$, where mr is generated by applying Algorithm 1.

Algorithm 1 New medical record of patient P

```
1: units ← getAllUnitsIncludedIn(CT)
2: content ← ∅
3: for unit in units do
4:    if unit.inuse then
5:       if not (unit.uqpt or unit.exph) then
6:          itemCont ← [copyCurrentText(unit.ptext), unit]
7:          content ← content ⋃ {itemCont}
8:       end if
9:    end if
10: end for
11: mr_t^P ← <content,P,t>
```

Algorithm 1 starts by getting all units of thought referenced in a case type CT (line 1). The algorithm iterates over all referenced units to identify units of thought marked with *inuse* attribute (lines 3–4). Besides, units marked with *uqpt* or *exph* attribute are not taken into account for creating a new medical record (line 5). A new data pair is created for each identified unit (line 6), each pair includes the identified unit of thought, and a copy of its current text presentation. All new pairs are joined to build the full content of the consultation record (line 7). Finally, mr_t^P is created as a new medical record, containing the full description of the consultation of patient P at time t.

4.3. Consultation flows

This section presents different flows for address the most relevant scenarios that arise during medical consultations. These scenarios describe usual situations of physician workday, including multiple diagnoses, and the attention of chronic patients.

4.3.1. Starting attention of a patient

The physician starts the attention of patient P by opening a registry editor where all the information of the new medical consultation will be recorded. Algorithm 2 details the first steps which occur during a medical consultations.

Algorithm 2 Start attention of patient P

1: showPersonalInfo(P.personalData)
2: showChronicElementDescriptors(P.chronicElems)
3: chronicCTs ← getCaseTypesByNames(P.chronicCaseTypes)
4: **if** chronicCTs $\neq \emptyset$ **then**
5: All case types included in chronicCTs are suggested to physician
6: Physician select $CT^M_{chron_1}$, ..., $CT^M_{chron_k}$ to be used as basis
7: CT^M_{merge} is build by merging $CT^M_{chron_1}$, ..., $CT^M_{chron_k}$ *(Algorithm 8)*
8: applyCaseType(P, CT^M_{merge}) is called *(Algorithm 3)*
9: **end if**
10: Show message agents according its trigger conditions
11: Physician continues with patient attention

Personal information of patient P is loaded at the beginning of Algorithm 2 (line 1) in order to introduce the patient to the physician. All descriptors of chronic elements associated with the patient are also presented, to remind the physician the chronic characteristics of the patient being evaluated (line 2). Furthermore, all chronic case types associated with the patient are detected (line 3). These chronic case types are suggested to physician, who can select the chronic case types that are appropriate to being applied into the consultation (lines 4–9). Before physician continues with patient attention, all message agents are evaluated, and showed according its trigger conditions (line 10).

4.3.2. Selecting an already defined case type

The selection of an already defined case type allows physician efficiently reuse previously registered information. Algorithm 3 details how to apply a

case type during a medical consultation.

Algorithm 3 applyCaseType(P, CT^M)

```
1: CT^M = <name, components, chronic, chronicComponents> is selected
2: if chronic then
3:    applyChronicCaseType(P, CT^M)  (Algorithm 4)
4: else
5:    elements ← getAllElementsIncludedIn(components)
6:    setUnitsInUse(elements,CT^M)
7: end if
8: Show all units with isuse attribute in true
9: Highlight all units with exph attribute in true
10: Show message agents that have CT^M as a trigger condition

11: procedure setUnitsInUse(elements,CT^M)
12: for element in elements do
13:    units ← getAllUnitsIncludedIn(elements)
14:    for unit in units do
15:       switch ()
16:       case unit.exph:
17:          unit.inuse = true
18:       case isTime(unit.ctSchedule, CT^M):
19:          unit.inuse = true
20:       case element.display ∧ isEmpty(unit.ctSchedule):
21:          unit.inuse = true
22:       case otherwise:
23:          unit.inuse = false
24:       end switch
25:    end for
26: end for
27: end procedure
```

Algorithm 3 starts by evaluating the *chronic* attribute of a case type CT^M (lines 1–2). If the case type is identified as chronic, a specific method for applying a chronic case is called (line 3). Otherwise, all elements referenced in the *components* attribute are determined, and its units of thought are marked as in use according *setUnitsInUse* auxiliary procedure. The auxiliary procedure encapsulates the logic of how units of thought are activated. The algorithm continues by showing all units marked as in use, and highlighting the units that are exclusive for physician use (lines 8–9). Finally, each message agent that has CT^M as a trigger condition is presented to the physician (line 10).

Procedure *setUnitsInUse* iterates over all conceptual elements of a case

type (line 12). All units included in each conceptual element are identified (line 13), and each unit of thought is marked as in use according the values of its attributes (lines 14–25).

4.3.3. Chronic patients flow

A case type CT^M can be marked as a chronic case type CT^M_{chron} at any time. When a CT^M_{chron} is marked as chronic, its *chron* attribute is activated and its *chronicComponents* attribute is initialized with an empty set. The chronic components are defined the first time that the case type is used to monitor a chronic patient. Algorithm 4 details how a physician can apply a chronic case type CT^M_{chron}.

Algorithm 4 applyChronicCaseType(P, CT^M_{chron})

```
1: A CT^M_chron = <name, components, true, chronicComponents> is selected
2: if name ∉ P.chronicCaseTypes then
3:    elements ← getAllElementsIncludedIn(components)
4:    setUnitsInUse(elements,CT^M_chron)
5: else
6:    if chronicComponents = ∅ then
7:       Evolution component CC_Evolution emerges
8:       Physician defines all conceptual elements included in CC_Evolution
9:       CC_others can be defined to better monitor the chronic condition
10:      CCs_new = CC_Evolution ⋃ CC_others
11:      newMonitorElems ← getAllElementsIncludedIn(CCs_new)
12:      newChronUnits ← getAllUnitsIncludedIn(newMonitorElems)
13:      for newChronUnit in newChronUnits do
14:         Physician needs to specify the frequency of newChronUnit
15:         newChronUnit.ctSchedule is updated
16:      end for
17:      chronicComponents ← CCs_new the chronic case type is updated
18:   end if
19:   elements ← getAllElementsIncludedIn(chronicComponents)
20:   setUnitsInUse(elements,CT^M_chron)
21: end if
```

Algorithm 4 analyzes if it is the first time that a chronic case type CT^M_{chron} is used with a patient being evaluated (lines 1–2). In that case, elements referenced in usual conceptual components are determined, and its units of thought are activated by calling *setUnitsInUse* procedure (lines 3–4). If CT^M_{chron} was used in any previous consultation of the same patient (line 6), then its *chronic*

components are taken into account each time physician decides to apply the case type, since *chronic* components are used to monitor the evolution of a chronic condition. However, the first time that CT^{M}_{chron} is used to monitor the evolution of a patient, the physician needs to define all entities that he wants to use as monitoring items (lines 7–12). In addition, it is mandatory that physician specify the frequency of each new unit of thought, included in an element of a chronic component (lines 13–16). All entities defined in new chronic components are used to monitor the patient chronic condition in subsequent consultations (line 17). Finally, the units of thought of the elements referenced in *chronic* components are marked as in use by applying *setUnitsInUse* procedure (lines 19–20).

4.3.4. Usual attention flow

During an attention flow, a physician can take advantage of an already registered case type. Algorithm 5 shows how a case type can be used to record a frequent medical consultation scenario.

Algorithm 5 Usual attention flow of a patient P

1: $\mathrm{CT}^{M} \leftarrow$ selectSimilarCT()
2: applyCaseType(P, CT^{M}) is called
3: Physician M define CT'^{M} by modifying the selected CT^{M}
4: personalInfo $\leftarrow$ getUqptUnits(CT'^{M})
5: P.personalData.add(personalInfo)
6: $\mathrm{CT}'^{M} \leftarrow$ removeUqptUnits(CT'^{M}) units marked with *uqpt* are removed
7: $\mathrm{mr}^{P}_{t} \leftarrow \mathrm{T}^{*}(\mathrm{CT}'^{M})$
8: $\mathrm{MR}^{P} \leftarrow \mathrm{MR}^{P} \bigcup \{\mathrm{mr}^{P}_{t}\}$
9: chronElemts $\leftarrow$ getAllActiveChonicElementsInludedIn(CT'^{M})
10: P.chronicElems.add(chronElemts)
11: **if** isChronic(CT'^{M}) **then**
12: P.chronicCaseTypes.add(CT'^{M}.name)
13: **end if**
14: $\mathrm{CT}'^{M} \leftarrow$ removeUqcnUnits(CT'^{M}) units marked with *uqcn* are removed
15: **if** CT'^{M} is saved as an improvement **then**
16: $\mathrm{CT}^{M} \leftarrow \mathrm{CT}'^{M}$
17: CKB^{M} is updated with the new version of CT^{M}
18: **else**
19: $\mathrm{CT}^{M}_{new} \leftarrow \mathrm{CT}'^{M}$ is saved as a new case type
20: $\mathrm{CKB}^{M} \leftarrow \mathrm{CKB}^{M} \bigcup \{\mathrm{CT}^{M}_{new}\}$ the base is incremented
21: **end if**

In Algorithm 5, a procedure waits until the physician selects a case type and applies it to current consultation (lines 1–2). After a case type is applied, the physician can also make modifications in order to describe accurate information of the entire clinical meeting (line 3). Each unit of thought marked as unique to the patient being evaluated is stored as personal data, and is removed of current case type (lines 4–6). The algorithm continues by applying T^* transformation, which generates a new medical record for patient history. (lines 7–8). All chronic conceptual elements used in the case type are associated with the patient. Furthermore, if the case type is chronic, it is associated as permanent patient data (lines 9–13). Each unit of thought marked as unique to current consultation is removed before updating the CKB^M of the physician (line 14). To update CKB^M, physician needs to specify if the current case type refers to a new workday scenario, or it is only an improvement over previous selected case type (lines 14–21).

Two data sets are modified after an usual attention flow, physician CKB^M and patient history, including a MR^P increment.

4.3.5. New case type flow

Algorithm 6 details the flow followed by the physician when he needs to address a new case type which is not included in his *CKB*.

Since there is no case type to be reused, Algorithm 6 needs to create an empty case type in which the new workday scenario can be detailed (lines 1–2). To define a new case type CT^M_{new}, physician can re-use any predefined unit of thought, and can also create units of thought specifying new clinical phrases. Furthermore, predefined conceptual elements can be re-used and new elements can be created (lines 3–4). Each element defined by the physician is referenced from one clinical section. Therefore, new conceptual components are created in order to group elements sharing the same section type (lines 5–11). It is mandatory that physician assign a name to the new clinical case type. The case type can also be marked as chronic, and in that case physician needs to specify the chronic attribute of each new element, created while defining the new case type (lines 12–21). All units of thought marked as unique to the patient are stored as personal data, and are removed from CT^M_{new} (lines 22–24). Then, T^* transformation is applied to generate a new medical record in patient history (lines 25–26). All chronic conceptual elements of CT^M_{new} are

associated with the patient, and if the case type is chronic it is associated as a permanent patient data (lines 27–31). Finally, all units of thought marked as unique to current consultation are removed from the case type, and the clinical data base of the physician is enriched, by including the new case type.

Algorithm 6 New case type flow for the attention of patient P

1: There is no CT^M selected by physician
2: $\mathrm{CT}^M_{new} \leftarrow <$"new-name", $\emptyset$, false, $\emptyset >$ is created automatically
3: Physician creates new units UTs_{new} and new elements CEs_{new}
4: Physician defines sections, by using UTs_{new} and CEs_{new} or pre-defined
5: secTypes $\leftarrow$ *ALL-SECTION-TYPES*
6: newComponents $\leftarrow \emptyset$
7: **for** $\mathrm{secType}_i$ **in** secTypes **do**
8: $\quad \mathrm{activeElems}_i \leftarrow$ [elementName, activeDescriptor] pairs in $\mathrm{section}_i$
9: $\quad \mathrm{CC}_{new_i} \leftarrow <$ maxCCId() + 1, $\mathrm{secType}_i$, $\mathrm{activeElems}_i >$
10: $\quad$ newComponents $\leftarrow$ newComponents $\bigcup \{\mathrm{CC}_{new_i}\}$
11: **end for**
12: Physician assigns a unique name to attribute *name* of CT^M_{new}
13: Physician can mark CT^M_{new} as chronic
14: **if** CT^M_{new} is marked as chronic **then**
15: $\quad$ **for** elem **in** CEs_{new} **do**
16: $\quad\quad$ Physician needs to specify the value of *elem.chron*
17: $\quad$ **end for**
18: $\quad \mathrm{CT}^M_{new} \leftarrow <$name, newComponents, true, $\emptyset >$
19: **else**
20: $\quad \mathrm{CT}^M_{new} \leftarrow <$name, newComponents, false, $\emptyset >$
21: **end if**
22: personalInfo $\leftarrow$ getUqptUnits(CT^M_{new})
23: P.personalData.add(personalInfo)
24: $\mathrm{CT}^M_{new} \leftarrow$ removeUqptUnits(CT^M_{new})
25: $\mathrm{mr}^P_t \leftarrow \mathrm{T}^*(\mathrm{CT}^M_{new})$
26: $\mathrm{MR}^P \leftarrow \mathrm{MR}^P \bigcup \{\mathrm{mr}^P_t\}$
27: chronElemts $\leftarrow$ getAllActiveChonicElementsInludedIn(CT^M_{new})
28: P.chronicElems.add(chronElemts)
29: **if** isChronic(CT^M_{new}) **then**
30: $\quad$ P.chronicCaseTypes.add(CT^M_{new}.name)
31: **end if**
32: $\mathrm{CT}^M_{new} \leftarrow$ removeUqcnUnits(CT^M_{new})
33: $\mathrm{CKB}^M \leftarrow \mathrm{CKB}^M \bigcup \{\mathrm{CT}^M_{new}\}$ the base is incremented

4.3.6. Temporal case type flow

By applying a temporal case type, the history of patient P is updated, and MR^P set is incremented with a new patient medical record. However, there is no change in physician CKB^M. Algorithm 7 details the use of a temporal case type.

Algorithm 7 Temporal case type flow for the attention of patient P

1: $\mathrm{CT}^M \leftarrow$ selectSimilarCT()
2: applyCaseType(P, CT^M) is called
3: Physician M define CT'^M by modifying the selected CT^M
4: Physician M marks CT'^M as a temporal case type
5: personalInfo $\leftarrow$ getUqptUnits(CT'^M)
6: P.personalData.add(personalInfo)
7: $\mathrm{CT}'^M \leftarrow$ removeUqptUnits(CT'^M)
8: $\mathrm{mr}_t^P \leftarrow \mathrm{T}^*(\mathrm{CT}'^M)$
9: $\mathrm{MR}^P \leftarrow \mathrm{MR}^P \bigcup \{\mathrm{mr}_t^P\}$
10: chronElemts $\leftarrow$ getAllActiveChonicElementsInludedIn(CT'^M)
11: P.chronicElems.add(chronElemts)
12: CT'^M is deleted

Algorithm 7 is triggered after physician assigns a temporal mark over an applied and modified case type (lines 1–4). As any other case type, all units marked as unique to the patient are stored as personal data, and are removed from the temporal case type (lines 5–7). T^* transformation is also applied to create a new medical record (lines 8–9). Each chronic conceptual element referenced in the temporal case is permanently associated with the patient being evaluated (lines 10–11). The temporal case type is finally removed, since it is marked to be not re-used (line 12).

4.3.7. Multiple case types flow

A physician can use more than one case type as basis during the same medical consultation. Several rules are used to combine all conceptual components of each case type involved. To combine conceptual components, their active conceptual elements are accurate merged. The merge process takes into account the active elements described in usual components, and active elements described in chronic components. Algorithm 8 details the method used to merge different case types.

Algorithm 8 Multiple case types flow for the attention of patient P

1: $\text{CT}_1^M \leftarrow$ selectSimilarCT()
2: $\text{CT}_2^M \leftarrow$ selectSimilarCT()
3: $\text{comps}_1 \leftarrow$ getAllComponents(CT_1^M)
4: $\text{comps}_2 \leftarrow$ getAllComponents(CT_2^M)
5: $\text{comps}_{merge} \leftarrow$ **merge**(comps_1, comps_2)
6: $\text{chronComps}_1 \leftarrow$ getAllChronicComponents(CT_1^M)
7: $\text{chronComps}_2 \leftarrow$ getAllChronicComponents(CT_2^M)
8: $\text{chronComps}_{merge} \leftarrow$ **merge**(chronComps_1, chronComps_1)
9: $\text{name}_{merge} \leftarrow$ concat(CT_1.name,CT_2.name)
10: **if** $\text{chronComps}_{merge} \neq \emptyset$ **then**
11: $\quad \text{CT}_{merge}^M = <\text{name}_{merge}, \text{comps}_{merge}$, true, $\text{chronComps}_{merge} >$
12: **else**
13: $\quad \text{CT}_{merge}^M = <\text{name}_{merge}, \text{comps}_{merge}$, false, $\emptyset >$
14: **end if**
15: CT_{merge}^M is auto-selected
16: applyCaseType(P, CT_{merge}^M) is called
17: Physician M define CT'^M_{merge} by modifying CT_{merge}^M.
18: personalInfo $\leftarrow$ getUqptUnits(CT'^M_{merge})
19: P.personalData.add(personalInfo)
20: $\text{CT}'^M_{merge} \leftarrow$ removeUqptUnits(CT'^M_{merge})
21: $\text{mr}_t^P \leftarrow \text{T}^*(\text{CT}'^M_{merge})$
22: $\text{MR}^P \leftarrow \text{MR}^P \bigcup \{\text{mr}_t^P\}$
23: chronElemts $\leftarrow$ getAllActiveChonicElementsInludedIn(CT'^M_{merge})
24: P.chronicElems.add(chronElemts)
25: **if** isChronic(CT_1^M) **then**
26: $\quad$ P.chronicCaseTypes.add(CT_1^M.name)
27: **end if**
28: **if** isChronic(CT_2^M) **then**
29: $\quad$ P.chronicCaseTypes.add(CT_2^M.name)
30: **end if**
31: CT'^M_{merge} is deleted

Algorithm 8 starts by identifying the conceptual components included in each case type (lines 1–4). An ad-hoc *merge* function is used to combine all identified components (line 5). Function *merge* it is also applied over chronic conceptual components (lines 6–8). The algorithm continues by creating a case type CT_{merge}^M, which include all merged components (lines 9–14). Then, the case type is applied and can be modified by the physician (lines 15–17). Each unit of thought marked as unique to the patient is taken into account as usual, it is stored as a personal data and it is removed from the clinical case

type (lines 18–20). Likewise, a new medical record is created by applying T^* transformation (lines 21–22). Each chronic conceptual element referenced in CT^M_{merge} is permanently associated with the patient being evaluated, as well as any original chronic case type (lines 23–30). Finally, the used case type is deleted after concluding the consultation (line 31).

The function used to combine conceptual elements of two different components is detailed in Algorithm 9. The proposed merge function prioritizes elements with abnormal descriptors, over those with normal descriptors.

Function *merge* starts by identifying all types of clinical data sections, present in conceptual component parameters (line 2). The function iterates over all identified section types, in order to create new conceptual components (line 4). In each iteration, all elements referenced in each conceptual component are identified by their names (lines 5–8). When an element is used in a conceptual component, there is a pair [name, descriptor] that indicates which descriptor is defined as active in the conceptual component. Consequently, it is necessary to determine which descriptor is selected for an element present in more than one conceptual component (lines 9–13). Conceptual elements of components sharing the same section type are combined by prioritizing abnormal descriptors (lines 14–20). If an element is activated with abnormal descriptors in two different conceptual components, then the units of thought included in each descriptor are joined. All units of thought are combined in order to define a new descriptor, describing both abnormal conditions together (lines 21–32). If an element is only present in one conceptual component, then its active descriptor is included as a reference into the merged conceptual component (lines 33–38). After all iterations are concluded, the set of all merged conceptual components are returned (line 41).

Algorithm 9 Merge conceptual components function

```
1: function merge(comps_1, comps_2): components_merge
2: secTypes ← allSectionTypes(comps_1) ⋃ allSectionTypes(comps_2)
3: components_merge ← ∅
4: for secType in secTypes do
5:    compSec1 ← getComponentBySectionType(secType, comps_1)
6:    compSec2 ← getComponentBySectionType(secType, comps_2)
7:    elNames1 ← getAllNamesOfActiveElements(compSec1)
8:    elNames2 ← getAllNamesOfActiveElements(compSec2)
9:    elNames_common ← {elNames1 ⋂ elNames2}
10:   elemDescs_common ← ∅
11:   for comElemName in elNames_common do
12:      activeDesc1 ← getActiveDescriptor(comElemName, compSec1)
13:      activeDesc2 ← getActiveDescriptor(comElemName, compSec2)
14:      switch ()
15:      case activeDesc1 = Normal ∧ activeDesc2 = Normal:
16:         elemDesc_merge ← activeDesc1, activeDesc1 = activeDesc2
17:      case activeDesc1 = Normal ∧ activeDesc2 ≠Normal:
18:         elemDesc_merge ← activeDesc2, only activeDesc2 it is abnormal
19:      case activeDesc1 ≠ Normal ∧ activeDesc2 = Normal:
20:         elemDesc_merge ← activeDesc1, only activeDesc1 it is abnormal
21:      case activeDesc1 ≠ Normal ∧ activeDesc2 ≠ Normal:
22:         comElem ← getElementByName(comElemName)
23:         comSetDesc ← comElem.setDesc
24:         units_1 ← getUnitsByDescriptor(activeDesc1,comSetDesc)
25:         units_2 ← getUnitsByDescriptor(activeDesc2,comSetDesc)
26:         bothDescs ← concat(activeDesc1, activeDesc2)
27:         unionPairDesc ← [bothDescs, units_1 ⋃ units_2]
28:         comElem.setDesc ← comElem.setDesc ⋃ { unionPairDesc}
29:         elemDesc_merge ← bothDescs
30:      end switch
31:      elemDescs_common ← elemDescs_common ⋃ {elemDesc_merge}
32:   end for
33:   elNames_disj1 ← elNames1 - {elNames1 ⋂ elNames2}
34:   elemDescs_disj1 ← getActiveElemsByName(elNames_disj1, compSec1)
35:   elNames_disj2 ← elNames2 - {elNames1 ⋂ elNames2}
36:   elemDescs_disj2 ← getActiveElemsByName(elNames_disj2, compSec2)
37:   elemDescs_all ← elemsDescs_common ⋃ elemDescs_disj1 ⋃ elemDescs_disj2
38:   CC_merge = <maxCCId() + 1, secType, elemDescs_all >
39:   components_merge ← components_merge ⋃ CC_merge
40: end for
41: return components_merge
42: end function
```

Chapter 5

Instance-based learning

This chapter presents a learning method proposed to generate suggestions for physicians. The proposed method is based on an ad-hoc similarity metric, designed to compare the similarity between clinical case types. The chapter is organized as follows. Section 5.1 introduces the learning method. The similarity metric used by the proposed method is detailed in Section 5.2. Finally, Section 5.3 presents the main characteristic of the metric implementation.

5.1. Instance-based learning method

An instance-based learning method is designed with the aim to provide suggestions for physicians. The proposed method takes into account the clinical knowledge base of a physician, in order to present suggestions based on previously-defined case types. A register editor where a physician can take advantage of the proposed instance-based learning method is also introduced.

5.1.1. Register editor

The register editor is an interface in which a physician can register a consultation appointment. The register editor presents personal information of the patient being evaluated, and includes an area for writing all details of a medical consultation. The main features of the register editor are illustrated in Figure 5.1, including a list of case type suggestions.

The input area of the register editor is designed with the aim of registering a consultation in an organized structure, grouping information by clinical section

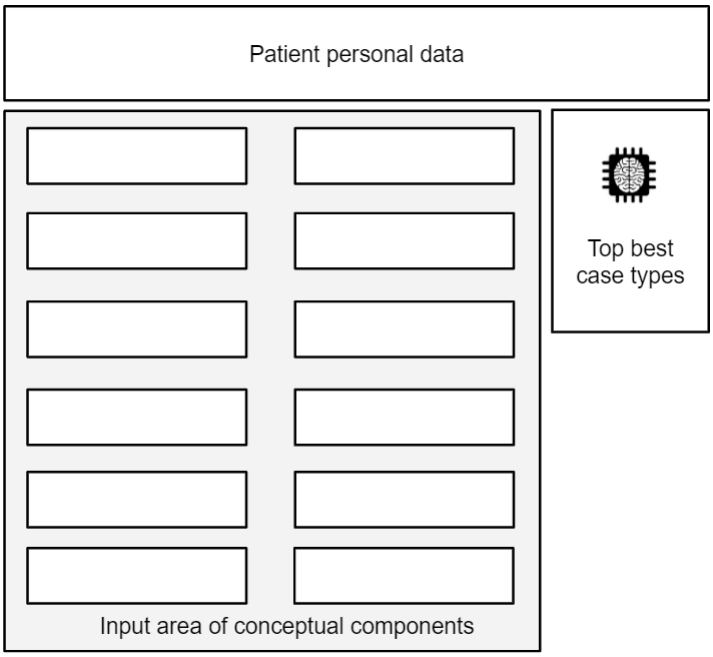


Figure 5.1: Main features of register editor.

types. When a physician writes in the input area, a case type $\mathrm{CT}_{current}$ is automatically created, based on the information included in each section type.

As a relevant feature, a list of similar case types is included in the register editor as suggestions for the physician. The suggested list is based on the top best values of a similarity metric, applied to compare the information of $\mathrm{CT}_{current}$ against all case types previously registered.

5.1.2. Learning method

A learning method is applied to determine the case types that best match with the clinical scenario of the patient being evaluated, according to an ad-hock similarity metric. A list of similar case types is suggested each time the physician modifies the information of the patient being evaluated. The list of similar case types is updated when introducing or removing any clinical phase during a medical consultation.

The proposed learning method implements a *lazy* approach (Mitchell, 1997), since the training stage of learning is delayed until a new case type draft must be evaluated. To evaluate a new case type draft, all previously-defined case types are processed as training examples, and a similarity metric is applied to determine the most similar candidates. Algorithm 10 details how

the instance-based learning method is implemented, seeking to suggest similar case types.

Algorithm 10 Instance-based learning method

1: $\text{CT}_{current} \leftarrow$ new CT(registerEditor.content)
2: $\text{CT}_{current} \leftarrow$ removeNonMeaningfulUnits($\text{CT}_{current}$)
3: $\text{CT}_{current} \leftarrow$ removeDuplicateUnits($\text{CT}_{current}$)
4: topBest $\leftarrow$ Initialize array with t empty values
5: **for** CT_i **in** CKB^M **do**
6: similarityMetric $\leftarrow$ *similarity*($\text{CT}_{current}$,CT_i)
7: **if** similarityMetric is better that worstSimilarity(topBest) **then**
8: topBest $\leftarrow$ replaceWorst(topBest,CT_i)
9: **end if**
10: **end for**
11: topBest $\leftarrow$ removeEmptyValues(topBest)
12: **return topBest**

The learning method described in Algorithm 10 is triggered each time the physician modifies any aspect of the consultation being evaluated. An auxiliary case type $\text{CT}_{current}$ is created based on the information detailed by the physician in his register editor (line 1). Sentences without any meaningful word are not taken into account by the learning method (line 2). A step to remove duplicate units of thought is applied, since a physician could write duplicate clinical phrases in his register editor (line 3). Moreover, an array used to identify top best *similarity* metrics is initialized with empty values (line 4). The algorithm continues by iterating over all case types included in the physician clinical knowledge base (line 5). For each iteration, the similarity between $\text{CT}_{current}$ and any other case type is calculated in order to update the array of top best metrics (lines 6–9). Finally, the case types with top best metrics are returned as suggestions to the physician (lines 11–12).

5.1.3. Using suggested case types

A physician can select a suggested case type as a basis of writing a medical consultation. After a case type is applied, the input area of the register editor is updated by using the information defined in the selected case type.

By using suggested case types, previously registered phases are re-used and the time spent writing the details of a consultation is reduced. Suggested case types can also remind physicians to verify important clinical aspects of their

patients. Moreover, taking advantage of previously written sentences is useful when physicians need to address recurrent aspects of chronic patients.

5.2. Similarity metric

A similarity metric is introduced in order to compare two case types of a clinical knowledge base. The proposed metric takes into account the similarity between conceptual components of different case types. Consequently, the similarity value between two case types is determined by the weighted similarities of their conceptual components.

5.2.1. Similarity metric definition

Sadegh-Zadeh (2015) introduced the concept of *diagnostic relevance*, which applies fuzzy logic to evaluate the relevance of causal events associated with a clinical diagnosis. The proposed method is based on a similar idea, where the concept of *medical relevance* is considered to evaluate the relevance of conceptual components associated with a clinical case type.

Let $comp_{i,sec}$ a conceptual component of case type CT_i defined with *sec* section type, where *sec* belongs to ALL-SECTION-TYPES. The set ALL-SECTION-TYPES is introduced in chapter 4 as a set that defines all possible clinical sections of patient medical records. The similarity between two case types is denoted as *similarity*, and is defined by Eq. 5.1.

$$similarity(CT_1, CT_2) = \sum_{sec} w_{sec} \times similarityCC_{sec}(comp_{1,sec}, comp_{2,sec}) \quad (5.1)$$

In Eq. 5.1, each w_{sec} defines the weight of a component with *sec* section type. Consequently, the conceptual components of case types influence the *similarity* metric according to their *sec* section type. The similarity weight of a clinical section type is determined by its medical relevance. Eq. 5.2 shows how medical relevance is used to define the weight of each section type belonging to ALL-SECTION-TYPES = $\{\text{sec}_1, ..., \text{sec}_n\}$.

$$W_{sec} = \frac{medRelevance(sec)}{\sum_{sec_1}^{sec_n} medRelevance(sec_i)} \quad (5.2)$$

The medical relevance of clinical section types must be defined based on

background knowledge of the health area. Accurate weights of conceptual components provide a mechanism for reducing the impact of irrelevant features in the similarity metric (Mitchell, 1997). As an example, health-background knowledge suggests that the section for excuse notes should weigh less than the diagnosis section.

The following subsection presents the *similarityCC* function, introduced in Eq. 5.1 for comparing two conceptual components of different case types.

5.2.2. Similarity between components

To compare conceptual components, the similarity metric takes into account the units of thought included in all elements of conceptual components. The similarity between conceptual components is defined by Eq. 5.3, which is aimed at comparing components sharing the same section type *sec*.

$$similarityCC_{sec}(cc_1, cc_2) = \begin{cases} 0, \text{ if } \text{cc}_1\text{.secType} \neq \text{sec} \vee \text{cc}_2\text{.secType} \neq \text{sec} \\ includedUTs(\text{units}(\text{cc}_1), \text{ units}(\text{cc}_2)), \text{ otherwise} \end{cases} \tag{5.3}$$

Function *includedUTs*: UT×UT → [-1,1] is applied to compare two sets of units of thought. Eq. 5.4 presents *includedUTs* by considering different scenarios.

$$includedUTs(units_1, units_2) = \begin{cases} 0, \text{ if } \text{units}_1 = \emptyset \\ \text{-1, if } \text{units}_1 \neq \emptyset \wedge \text{units}_2 = \emptyset \\ \max\{\sum\limits_{u_1 \in units_1} \frac{belongs(u_1, units_2)}{|units_2|}, \text{-1}\}, \text{otherwise} \end{cases} \tag{5.4}$$

If the first parameter $units_1$ of function *includedUTs* is an empty set, there is no unit of thought that can contribute as similarity data, then zero value is returned. Another exceptional case occurs when $units_2$ does not describe any information. If the second parameter is an empty set, the worst value of similarity must be returned because $units_1$ details clinical data not considered by $units_2$. A complex scenario arises when function *includedUTs* evaluates non-empty parameters. In that case, each unit of $units_1$ is analyzed in order to evaluate its inclusion into $units_2$, and a positive weight is determined for units that belong to both sets. In addition, a limit of maximum deference could be applied if $units_1$ and $units_2$ are significantly different and $units_1$ is bigger than $units_2$.

Function belongs for units of thought.

An auxiliary function *belongs(unit, units)* presented by Eq. 5.5 is required to determine if a specific unit of thought belongs to a set of units. A *unit* that contradict the ideas represented by the *units* set is negatively weighted.

$$belongs(unit, units) = \begin{cases} 1, & \text{if } \exists \text{ u}_{same} \in \text{units} \nearrow equals(unit, u_{same}) \\ \text{-1}, & \text{otherwise.} \end{cases} \quad (5.5)$$

5.2.3. Similarity metric algorithm

The metric detailed in Algorithm 11 calculates the similarity of a case type $\text{CT}_{current}$ regarding any other case type. To achieve the final value of the similarity metric, Algorithm 11 needs to calculate similarity values of several conceptual components.

Algorithm 11 starts by initializing the similarity metric with a neutral value (line 2). Then, all section types of conceptual components included in analyzed case types are identified (line 3). After identifying the section types that influence the similarity metric, a relative weight factor is determined in order to accurately weigh the influence of each identified section type (line 4). Each section type with a positive weight of similarity is taken into account to calculate the value of the metric (lines 5–6).

For each identified section type, the similarity of components sharing the same section type must be calculated. A cache containing values of similarity between conceptual components is used to improve the performance of the proposed metric (lines 7–9). To calculate the similarity between conceptual components, the sets of units of thought included in each component are determined. Algorithm 11 implements the rules introduced by Eq. 5.4 for calculating the similarity between two sets of units of thought, including the general scenario (lines 15–24) for non-empty sets, and exceptional scenarios (line 26 and line 29) to address singular situations of empty sets. Moreover, the calculated value of component similarity is cached, to be used in the future (line 31). The partial value of the similarity metric is updated after calculating the similarity between each pair of conceptual components. For each pair of components, the partial similarity is affected by the similarity of the components according to a relative weight factor (line 33).

Algorithm 11 Similarity metric

```
1: function similarity(CT_current, CT_i)
2: similarity ← 0
3: involvedSecTypes ← sectionTypesOf(CT_current) ⋃ sectionTypesOf(CT_i)
4: relativeWeight ← relativeWeightFactor(involvedSecTypes)
5: for sec in involvedSecTypes do
6:   if sec.weight ≠ 0 then
7:     cachedSimilarityCC ← getSimilarityCCValueFromCache(sec, CT_i)
8:     if cachedSimilarityCC is hitted then
9:       similarityCC ← cachedSimilarityCC
10:    else
11:      units_current ← getUnitsBySectionType(sec,CT_current)
12:      units_i ← getUnitsBySectionType(sec,CT_i)
13:      if units_current ≠ ∅ then
14:        if units_i ≠ ∅ then
15:          includedUTs ← 0
16:          for unit_current in units_current do
17:            belongs ← belongs(unit_current, units_i)
18:            if belongs then
19:              includedUTs ← includedUTs + 1/|units_i|
20:            else
21:              includedUTs ← includedUTs − 1/|units_i|
22:            end if
23:          end for
24:          similarityCC ← max{includedUTs, -1}
25:        else
26:          similarityCC ← -1
27:        end if
28:      else
29:        similarityCC ← 0
30:      end if
31:      putSimilarityCCValueInCache(similarityCC,sec,CT_i)
32:    end if
33:    similarity ← similarity + (relativeWeight * similarityCC)
34:    bestRemain ← upperBound(sec, involvedSecTypes)
35:    if similarity + bestRemain < worstSimilarity(topBest) then
36:      invalidateSimilarityCCValuesOnCache(CT_i)
37:      throw discard-low-similarity
38:    end if
39:  end if
40: end for
41: return similarity
```

To detect low values of similarity, an upper bound is calculated in order to determine a maximum possible value of similarity (line 34). If the similarity between two case types is detected early as low, it is not required to calculate its exact value. All case types with low *similarity* are discarded early, and their partial values of component similarity are removed from the cache as they are not fully calculated (lines 35–37). At last, a final value of similarity is returned after iterating over all involved section types (line 41).

5.3. Implementation of similarity metric

The similarity metric is an essential feature of the proposed learning method. The metric must be able to accurately compare the similarity between clinical case types, and it also needs to execute as quickly as possible.

The similarity metric is highly demanded in virtue of the lazy approach of the learning method. Therefore, several techniques of indexing and cache are applied for reducing the metric execution time. All optimizations implemented to reduce the execution time of the *similarity* metric are presented in the following subsections.

5.3.1. Compare units by canonical form

The operator *equal* for units of thought is used to determine if two different sentences represent the same clinical idea. To implement the *equal* operator, a canonical transformation is applied over the units being compared.

Two transformations are applied by comparing a pair of units of thought. For each unit of thought, structured information and random data are removed, in order to achieve the canonical form. Finally, a raw string comparison between both canonical forms is evaluated. Original units of thought are identified as equals only if they coincide in their canonical form.

5.3.2. Zero similarity value

Function *similarity*(CT_1, CT_2) is called to calculate the similarity between a case type CT_1 and another case type CT_2. Both case types are composed by conceptual components that influence the similarity metric according to its section type weights. However, an empty component of CT_1 cannot provide

similarity information since it does not have associations with units of thought. If a conceptual component of CT_1 is empty, its similarity in regard to any other component is zero. No calculation is done over the empty components of CT_1, rather all computational effort is performed over its non-empty components.

5.3.3. Comparing with empty components

All components of a case type CT_1 are analyzed when calculating the similarity of CT_1 in regard to another case type CT_2. Each conceptual component of CT_1 should be compared against a component of CT_2 with the same section type. If CT_2 does not include a conceptual component with the same section type, then a value representing the bigger difference of similarity is returned without performing additional calculations.

5.3.4. Discard non-promising candidates

The proposed learning method is designed to suggest the best case types that can be applied during a medical consultation. Top best case types are identified according to best *similarity* metric values, and only t best case types are presented to the physician.

The similarity function separates the first k section types from the rest of the ALL-SECTION-TYPES set, as described in Eq. 5.6

$$\begin{aligned} similarity(CT_1, CT_2) &= \sum_{sec=sec_1}^{sec_N} w_{sec} \times similarityCC_{sec} \\ &= \sum_{sec=sec_1}^{sec_k} w_{sec} \times similarityCC_{sec} + \sum_{sec=sec_k+1}^{sec_N} w_{sec} \times similarityCC_{sec} \\ &\leq \sum_{sec=sec_1}^{sec_k} w_{sec} \times similarityCC_{sec} + \sum_{sec=sec_k+1}^{sec_N} w_{sec} \\ &= \sum_{sec=sec_1}^{sec_k} w_{sec} \times similarityCC_{sec} + R^{k+1}_{constant} \end{aligned} \tag{5.6}$$

Eq. 5.7 presents an upper bound inferred by simplifying 5.6, which can be used for discarding case types with low similarity values.

$$similarity(CT_1, CT_2) \leq \sum_{sec=sec_1}^{sec_k} w_{sec} \times similarityCC_{sec} + R^{k+1}_{constant} \tag{5.7}$$

Several component similarity values *similarityCCs* need to be computed to get the final value of *similarity*(CT_1, CT_2). An upper bound is identified after determining the value of *similarity*CC_{sec_k}. After calculating the similarity of the *k-th* conceptual component, it is possible to use an upper bound to discard a case type with a low similarity value. Each case type whose upper bound of similarity is lower than the worst element of the top best metrics is considered a non-promising candidate, and no more computational effort is expended to calculate its final similarity value.

5.3.5. Cache of previous similarity values

The proposed similarity metric provides a mechanism for comparing different case types. However, the metric is not based on case types themselves, but on their conceptual components. Due to the high need of obtaining similarities between conceptual components, a cache is designed for containing pre-calculated values of component similarities. Figure 5.2 shows the structure used to maintain recent values of similarities, and how similarity values are cached for each case type included in the clinical knowledge base of a physician. The proposed structure is able to cache the last value of similarity of all conceptual components of each case type.

After evaluating the similarity between a specific case type in regard to any other case type, all values of component similarities are stored in the cache. Figure 5.3 introduces a scenario in which a "Case type A" is slightly modified, by only changing the information described in one of its conceptual components. Several highlighted values of component similarities are obtained from the cache. Furthermore, a high cache hit ratio should be achieved after re-using any other case type and applying a few modifications.

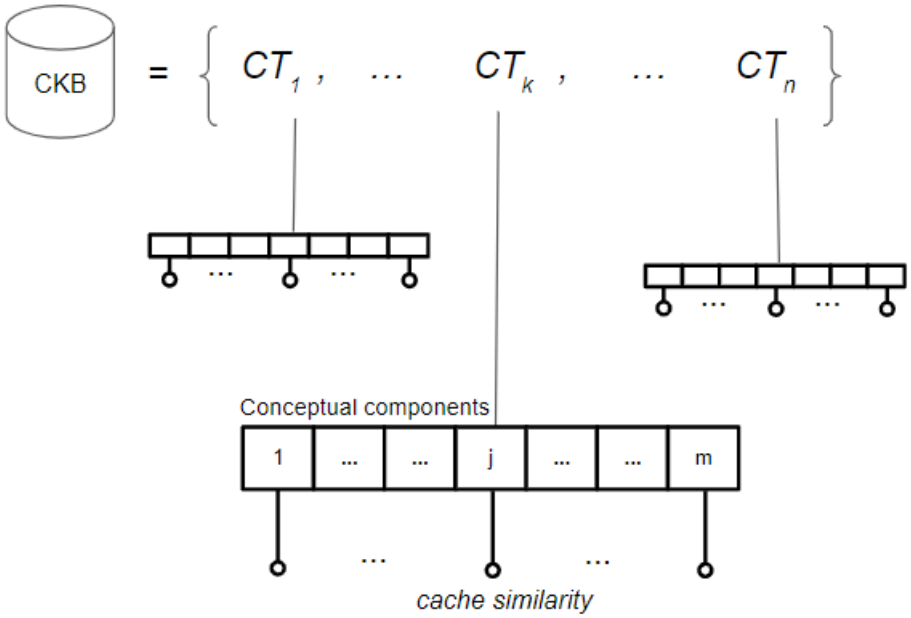
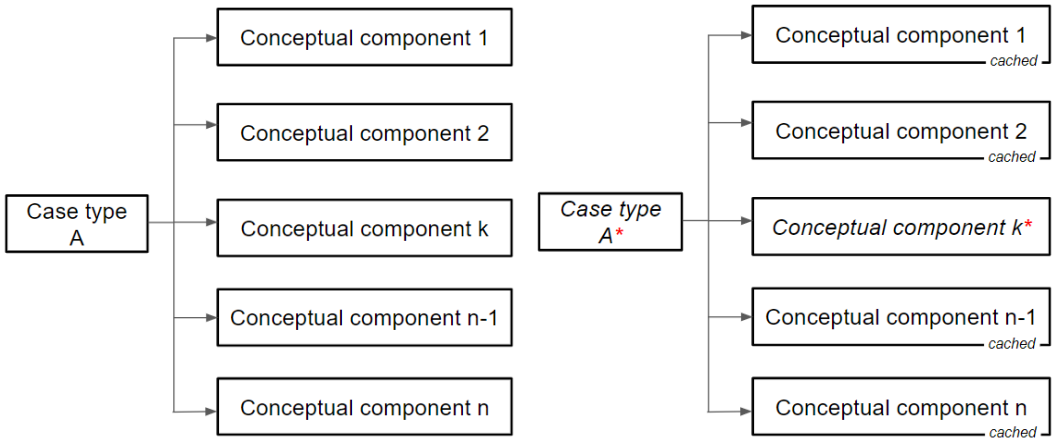


Figure 5.2: Cache structure for similarity between components. The clinical knowledge base is composed of case types, each one containing similarity cached values of its conceptual components.



(a) First similarity evaluation. **(b)** Similarity evaluation with cache.

Figure 5.3: Use of similarity cache values.

Chapter 6

Experimental analysis

This chapter presents the results of the experimental analysis performed to evaluate the proposed approach. In section 6.1, the problem instances used in the experiments are introduced, including an explanation of their building process. The parameter settings used in the experimental evaluation are presented in section 6.2. Section 6.3 details the execution times of the proposed learning method when solving problem instances considering different data sizes. In section 6.4, the main results of a prototype developed for testing the feasibility of the approach are described. Section 6.5 presents important health terminology standards which were included to achieve clinical information interoperability. Finally, section 6.6 presents a summary of the main results of the proposed approach.

6.1. Problem instances

The source *Clinical cases in primary care* (Casos clínicos en atención primaria) (Sociedad Andaluza de Medicina Familiar y Comunitaria, 2017) was used as a basis for evaluating the proposed approach. The source is a multi-authored publication that covers a wide range of clinical scenarios of primary care.

6.1.1. Prerequisites for building case type instances

The collaboration of advanced medical students was requested with the intention of registering as many clinical scenarios as possible. Students were instructed to record the primary care scenarios described in *Clinical cases*

in primary care as new clinical CTs. In order to group all the information recorded, it was necessary to implement procedures for exchanging clinical CTs. The *export* and *import* procedures used to exchange CTs are detailed in the following subsections.

Export a case type

A procedure to *export* a given CT was implemented. The procedure extracts a CT from a specific clinical knowledge base CKB, and it also anonymizes any information that refers to the person who wrote (owner) the CT.

Import a case type

The *import* procedure consolidates the information of a specific CT into a target CKB. A new CT is inserted into the target CKB, replacing any anonymized reference of the original owner with the person who owns the target CKB. Importing a CT is a complex procedure, which must avoid the generation of duplicate units of thought, and has to merge the conceptual elements of the new CT with those existing in the target CKB.

6.1.2. Building case type instances

The set of clinical cases specified in the publication *Clinical cases in primary care* was distributed to be evaluated by 50 advanced medical students. Each student had to evaluate three different cases, and each clinical case was assigned to at least one student. Furthermore, each student was instructed to contribute two additional clinical cases, defined as variants of those presented in the clinical source. All scenarios of primary care detailed in the clinical source were successfully registered by the group of students, including variants of repeated clinical scenarios. Algorithm 12 details how a single CKB was loaded with 250 scenarios of primary care, based on information registered by students.

Algorithm 12 starts by initializing all CKBs of the students (STUDENT-LIST) selected for recording new CTs (lines 1–3). Each student is expected to treat three fictitious patients suffering from one of the specific conditions of the clinical source (lines 5–6). Moreover, two variants contributed by each student are also registered (line 9 and line 11). Therefore, the CKB of each student

is enriched with five new CTs (line 7, line 10 and line 12). The algorithm continues by initializing a single CKB_{all} that groups all information recorded by all students (line 14). Each registered CT is exported using the *export* procedure, and the *import* procedure is applied to consolidate the exported CT into the CKB_{all} (lines 15–21). Finally, the CKB_{all} which contains all the 250 registered CTs (five contributed by each of the 50 students) is returned (line 22).

Algorithm 12 Building case types

1: **for** i = 1 **to** length(STUDENT-LIST) **do**
2: $\text{student}_i \leftarrow$ STUDENT-LIST[i]
3: $\text{CKB}_{student_i} \leftarrow \emptyset$
4: **for** j = 1 **to** 3 **do**
5: *k-index* $\leftarrow$ mod(3(i-1)+j, length(CASE-SOURCE))
6: $\text{CT}_{i_j} \leftarrow \text{student}_i$ records case number *k-index* of CASE-SOURCE
7: $\text{CKB}_{student_i} \leftarrow \text{CKB}_{student_i} \bigcup \{\text{CT}_{i_j}\}$
8: **end for**
9: $\text{CT}_{i_{v_1}} \leftarrow$ first variant of case type included in $\text{CKB}_{student_i}$
10: $\text{CKB}_{student_i} \leftarrow \text{CKB}_{student_i} \bigcup \{\text{CT}_{i_{v_1}}\}$
11: $\text{CT}_{i_{v_2}} \leftarrow$ second variant of case type included in $\text{CKB}_{student_i}$
12: $\text{CKB}_{student_i} \leftarrow \text{CKB}_{student_i} \bigcup \{\text{CT}_{i_{v_2}}\}$
13: **end for**
14: $\text{CKB}_{all} \leftarrow \emptyset$
15: **for** i = 1 **to** length(STUDENT-LIST) **do**
16: $\text{student}_i \leftarrow$ STUDENT-LIST[i]
17: **for** j = 1 **to** 5 **do**
18: $\text{CT}_{i_j exported} \leftarrow$ *export*(j,$\text{CKB}_{student_i}$)
19: *import*($\text{CT}_{i_j exported}$, CKB_{all})
20: **end for**
21: **end for**
22: **return** CKB_{all}

6.2. Parameter settings of similarity weight

For the purposes of the experimental evaluation, the set of clinical section types was defined following the Uruguayan health model. The set of clinical section types was defined as *ALL-SECTION-TYPES* = {*Diagnosis*, *Consultation reason*, *Current illness*, *Physical examination*, *Medication*, *Studies*, *Procedures*, *Referral*, *Message agents*, *Advisors*, *Excuse notes*, *Observations*}

The similarity weight of a clinical section type is given by its medical relevance. A simple medical relevance criteria was applied to give greater weight to the most important section types. Four levels of importance were defined in order to consider qualitative ranges of medical relevance. The level scale used to define medical relevance was: *very important*, *fairly important*, *important*, *slightly important*. Table 6.1 presents the weight values of the clinical section types used in the experimental evaluation, grouped by level of medical relevance. Table 6.1 shows that the weight of the diagnosis section was defined with a high value of $W_{Diagnosis} = 0.16$. On contrary, the excuse notes section was defined with a lower weight of $W_{Excuses} = 0.04$.

Table 6.1: Weight of conceptual component types.

Very important (weight 0.16)	**Fairly important** (weight 0.12)	**Important** (weight 0.08)	**Slightly important** (weight 0.04)
Diagnosis	Consultation reason	Medication	Message agents
	Current illness	Studies	Advisors
	Physical examination	Procedures	Excuse notes
		Referral	Observations

All weights presented in Table 6.1 influence the calculation of the similarity metric of the learning method. Eq 6.1 shows the consistency of presented weights to ponder the similarity metric.

$$\sum_{sec_1}^{sec_n} W_{sec} = \sum_{\text{very important}} W_v + \sum_{\text{fairly important}} W_f + \sum_{Important} W_i + \sum_{\text{slightly important}} W_s$$
$$\sum_{sec_1}^{sec_n} W_{sec} = 0.16 + 3 \cdot 0.12 + 4 \cdot 0.08 + 4 \cdot 0.04 = 1 \tag{6.1}$$

6.3. Performance evaluation

An experimental evaluation was conducted in order to analyze the lazy nature of the proposed learning method. In the learning approach, a similarity metric between clinical CTs is calculated by using all previous recorded CTs as training examples. Since the problem-solving ability of the proposed method is increased with each newly defined CT, it is important to analyze the performance of the proposed learning method when faced with CKBs with a great number of CTs.

6.3.1. Execution platform of performance evaluation

The execution time analysis was performed on a Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, 16 GB RAM, and running 64-bits Windows 10 Pro.

6.3.2. Execution time

The efficiency of the learning method was evaluated when faced with CKBs of different sizes. To make a realistic evaluation, the 250 CTs registered by students were considered as input data, and the average time of 50 executions was measured for each CKB analyzed.

Figure 6.1 presents the average execution time of the proposed method in regard different CKB sizes. The algorithm was executed on CKBs containing from 25 to 250 CTs.

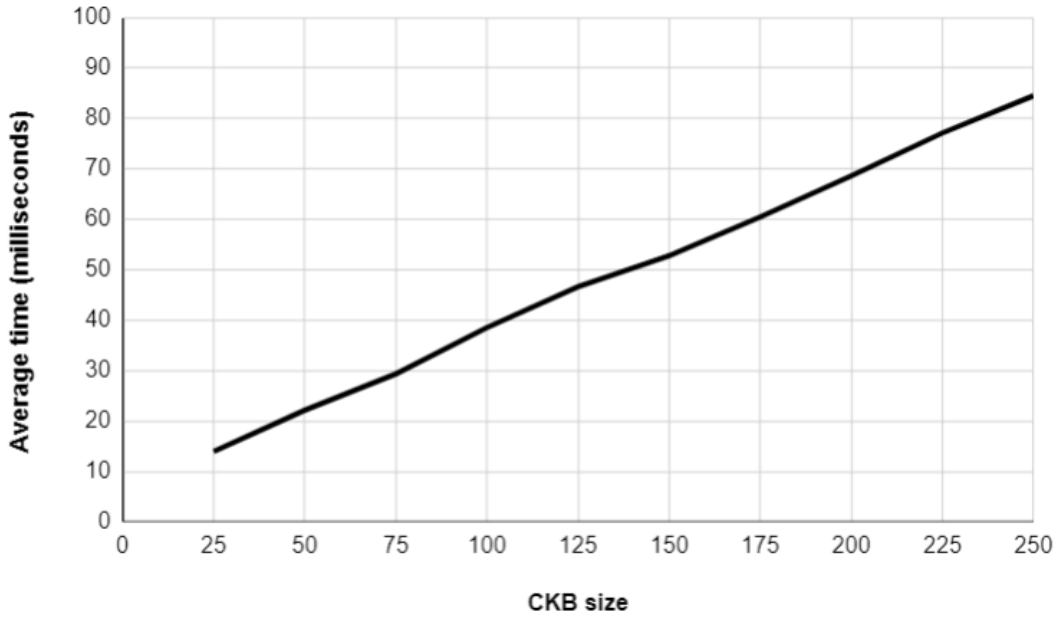


Figure 6.1: Average execution time of the proposed learning method regarding different CKB sizes.

Figure 6.1 shows how the size of a CKB has a direct influence on the execution time of the proposed method. Figure 6.1 also demonstrates that the proposed learning method is able to process 250 CTs in less than 90 milliseconds.

6.3.3. Execution time projection

In order to estimate the efficiency of the learning method when facing larger CKBs, auxiliary CTs were generated, based on the information recorded by the students. Although the auxiliary CTs were artificially built and do not reflect real clinical scenarios, they can be used to evaluate the performance of the learning method as they have the same data dimension as the CTs written by the students. The graphic in Figure 6.2 reports the average time of the proposed method regarding CKB sizes.

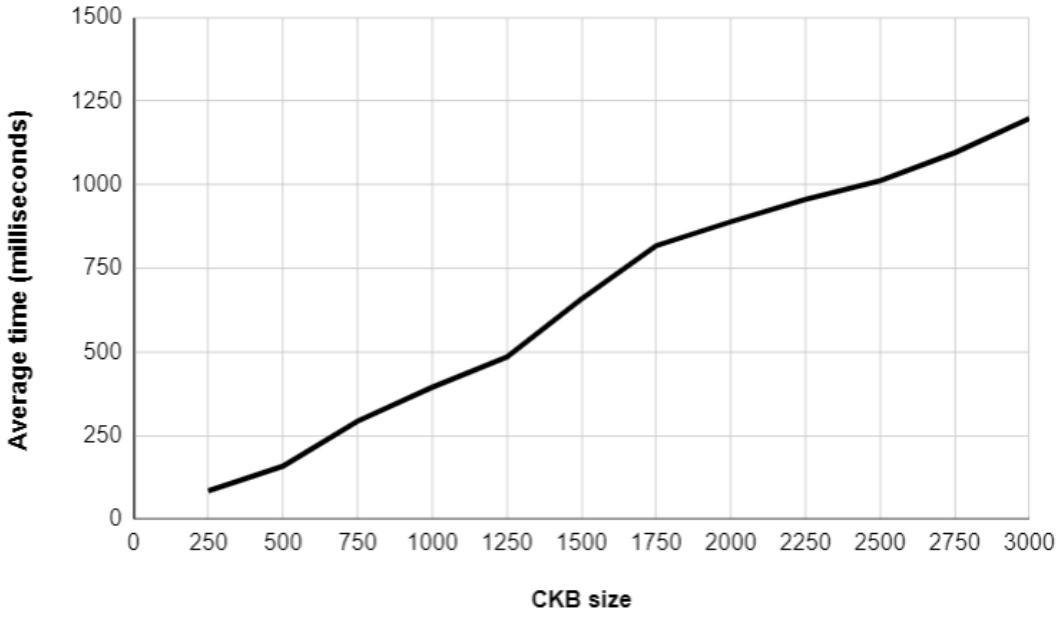


Figure 6.2: Average execution time of the proposed learning method when facing larger CKBs.

Figure 6.2 shows that the learning method generated suggestions in less than 1.25 seconds, even when facing larger CKBs with a great number of CTs. Given that 3000 represents a suitable bound for the number of CTs included in a physician CKB, the proposed method is able to produce suggestions in reasonable execution times, even when processing real CKBs with several workday scenarios.

6.3.4. Comparison with a Bayesian learning approach

To further analyze the applicability of the proposed approach, this subsection presents a comparison of the proposed instance-based learning method

against a Bayesian learning method, which is based on a classical algorithm described by Mitchell (1997) for classifying text documents. To compare both learning approaches, the computational efficiency of each method is analysed.

The implemented Bayesian learning method works under the assumption that the occurrence probability of a word is independent of its position within a document. Algorithm 13 details the proposed Bayesian learning method, which suggest case types by inferring estimated probabilities.

Algorithm 13 Bayes learning method (*doc*)

```
1: examples ← getCTBasedMedicalRecords()
2: vocabulary ← distinctWords(examples)
3: positions ← all words positions in doc that contain tokens of vocabulary
4: for CT_i in CKB^M do
5:    records_i ← getMedicalRecords(examples, CT_i)
6:    P(CT_i) ← |records_i| / |examples|
7:    text_i ← concatAllDocuments(records_i)
8:    n ← total number of distinct word positions in text_i
9:    for word_k in vocabulary do
10:      n_k ← number of times word_k occurs in text_i
11:      P(word_k|CT_i) ← (n_k + 1) / (n + |vocabulary|)
12:   end for
13: end for
14: topBest_Bayes ← Initialize array with t empty values
15: for CT_i in CKB^M do
16:   probBayes ← P(CT_i) ∏_{j∈positions} P(word_j|CT_i)}
17:   if probBayes is greater that worstProbability(topBest_Bayes) then
18:      topBest_Bayes ← replaceWorst(topBest_Bayes, CT_i)
19:   end if
20: end for
21: topBest_Bayes ← removeEmptyValues(topBest_Bayes)
22: return topBest_Bayes
```

Algorithm 13 starts by selecting all medical records that were written derived from a CT (line 1). Selected medical records are examined as training examples at the beginning of the learning task, aiming at extracting the vocabulary of all words appearing in patient histories (line 2). Each word included in the draft of the Register Editor that belongs to the previous vocabulary is tagged with its position (line 3). Several steps are required to obtain the probability estimates of the Bayesian approach. The first step estimates the probability of a case type by considering its occurrence in regard of all regis-

tered medical records (lines 5–6). Additional steps are needed to calculate the relative frequency of each word in relation to each case type (lines 7–12). The algorithm continues by iterating over all case types included in the physician CKB (line 15). For each iteration, an array of top best metrics is updated according to greatest Bayesian probabilities (lines 16–19). Finally, to classify a new draft of the Register Editor, top best metrics are used for returning the most likely case types to be applied (lines 20–21).

Figure 6.3 presents a comparison between the efficiency results of the two compared learning methods. The figure reports the average execution time of both the Bayesian method and the instance-based method, regarding different CKB sizes, when processing the total testbed of 250 CTs.

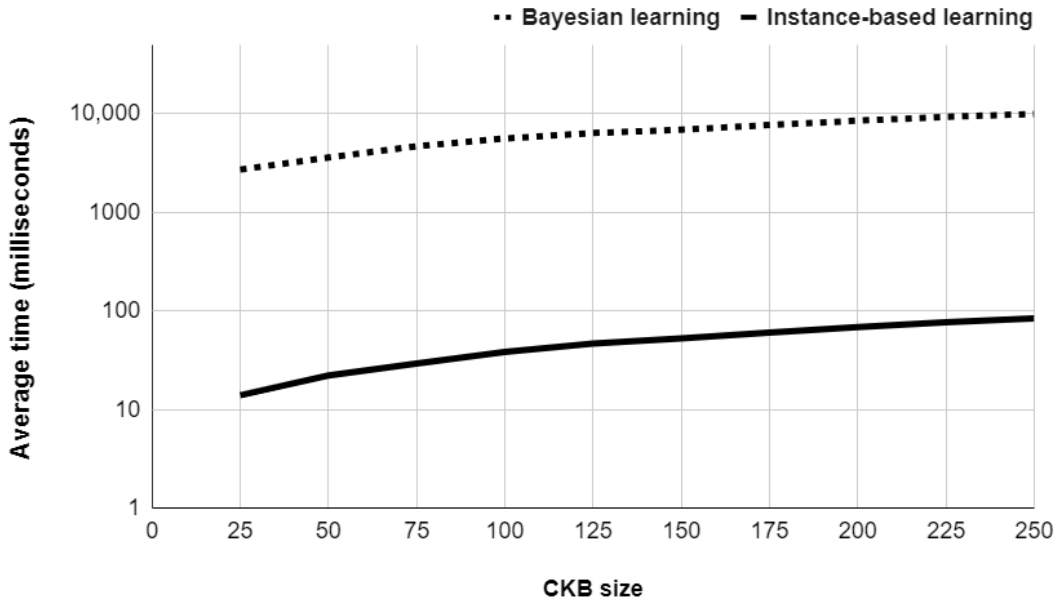


Figure 6.3: Average execution time comparison: instance-based learning vs. Bayesian learning method.

The graphic in Figure 6.3 uses a logarithmic scale to highlight the difference of two orders of magnitude between the execution times of both learning methods. The efficiency results reveal that the Bayesian learning method is difficult to apply in practice due to high execution times, even when processing small volumes of data. Execution time results also reaffirm the benefits of the instance-based learning method, which significantly outperforms the Bayesian approach in terms of efficiency.

6.4. Testing the applicability of the instance-based learning approach

In order to test the applicability of the proposed approach, a prototype was developed and deployed on Google Compute Engine (`cloud.google.com/compute`), the Infrastructure as a Service component of Google Cloud Platform. The prototype was evaluated by advanced medical students in their last year of training at Universidad de la República, Uruguay.

6.4.1. Comparison with original Praxis

Praxis reports the average time required to write a CT starting from an empty CKB (Low, 2015). In order to compare the proposed approach with the original implementation of Praxis, the average time to write a CT using the prototype was measured. Figure 6.4 illustrates both average writing times starting from an empty CKB, by considering the medical attendance of the first 50 patients.

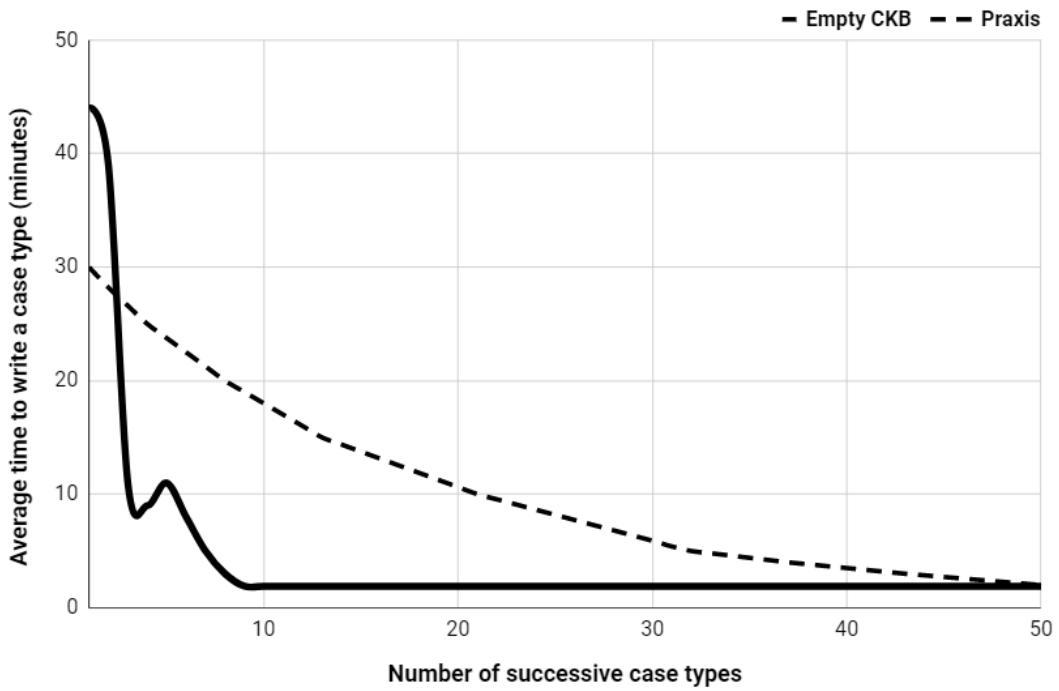


Figure 6.4: Average time of 50 medical students to write the notes of a case type (continuous line). Average time according to Praxis reports (dotted line). Both evaluations start with an empty CKB.

Although Praxis presents shorter registering times for the first two medical

consultations, more than 50 evaluations are needed to achieve a convergence point. The proposed approach significantly reduces registration times from the third case registered onwards, converging quickly to less than three minutes of writing consultations. The proposed learning method demanded 210 minutes to register 50 consultations (i.e., 4.2 minutes per consultation), while using Praxis requires 519 minutes (10.4 minutes per consultation). The overall time reduction factor is 2.5×.

6.4.2. Improvement using a pre-loaded CBK

The time needed to register a medical consultation can be reduced by using previously registered information. The average time to record a CT was measured in a context in which medical students could use a pre-loaded CKB. Figure 6.5 shows the average time to write a CT taking advantage of a pre-loaded CKB containing typical workday scenarios. As a relevant result, the use of a pre-loaded CKB implied a reduction of up to five minutes for recording the notes of the first six medical consultations. Furthermore, a pre-loaded CKB also accelerated the convergence to three minutes of writing consultations.

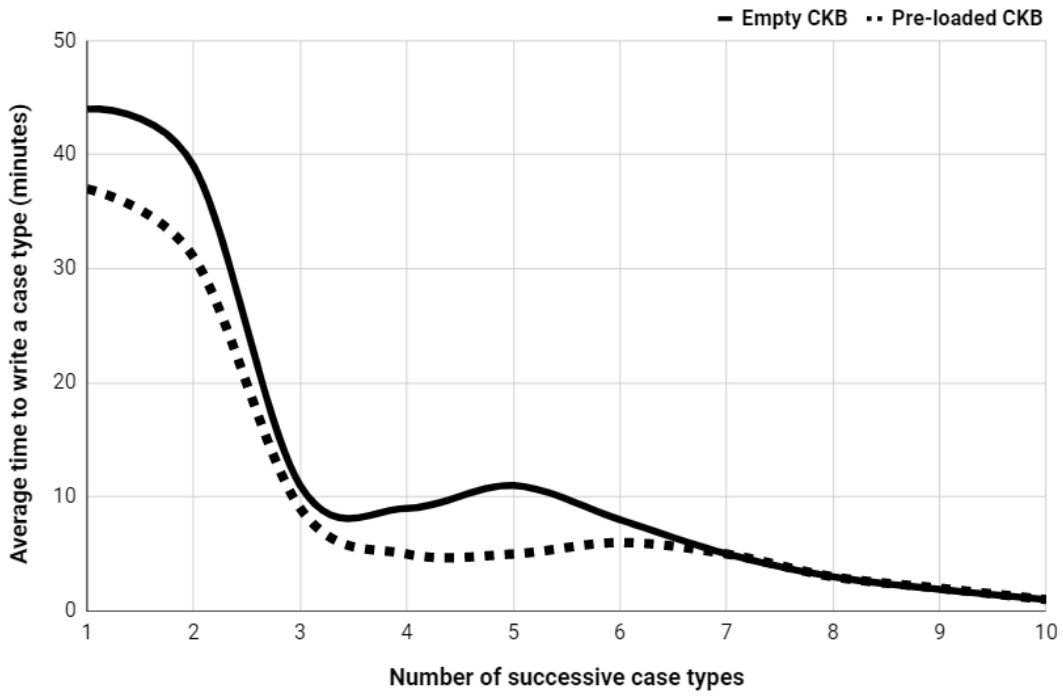


Figure 6.5: Average time of 50 medical students starting with an empty CKB (continuous line). Average time of 50 medical students taking advantage of a pre-loaded CKB (dotted line).

Regarding the scalability of the incremental processing of new case types, results suggest a convergence towards a short write time for medical consultations, even when processing large volumes of data.

6.4.3. Survey about the proposed approach

More than 50 medical students from different editions of the Medical Informatics course were surveyed after using the prototype of the proposed approach. The advanced medical students have tested the prototype during course editions from 2018 to 2020. Figure 6.6 summarizes the best features identified by students.

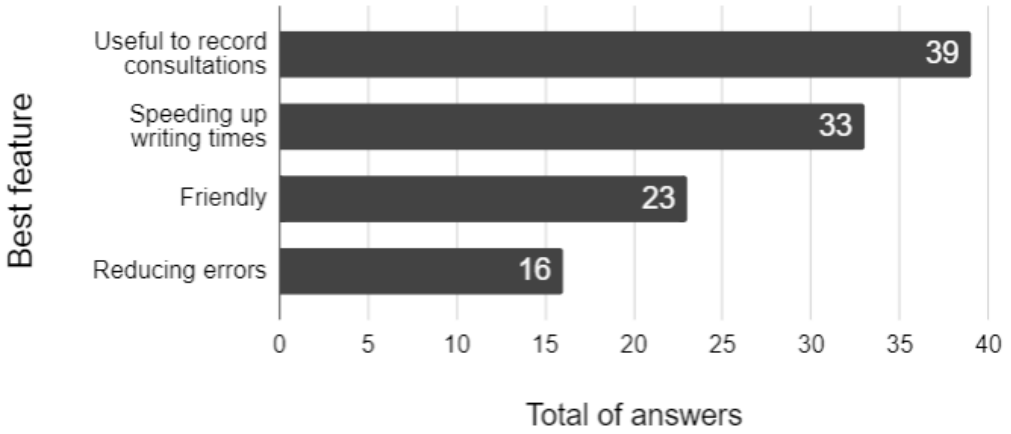


Figure 6.6: Best features of the proposed approach, according to the survey performed on students.

Results show that 43% of the surveyed students mentioned that the learning curve was steep before they could benefit from the proposed learning method. As a relevant result, more than 73% of the students considered the prototype as an appropriate tool for medical practice, especially at medical consultations. Moreover, 62% of the students were able to speed up writing time during medical consultations.

6.5. Interoperability of health information

Health terminological standards were taken into account due to the relevance of the interoperability of information in the Medical Informatics area. In particular, the national drug dictionary of Uruguay and a terminology server

provided by the Hospital Italiano de Buenos Aires (HIBA) were integrated into the proposed approach.

Integrating the national drug dictionary of Uruguay

A National Drug Dictionary (DNMA) is defined by *Salud.uy* (`salud.uy`) in order to standardize the information and vocabulary for clinical and logistical use applied to pharmaceutical and related products. DNMA acts as a standard of drug reference terminology for the network of healthcare service providers in Uruguay.

Access permissions were requested from the DNMA in order to import the national dictionary of medicines into the proposed approach. Importing the national drug dictionary helped build a functional model, in which physicians can make pharmacological indications using a wide range of drugs.

Using the terminology server of Hospital Italiano de Buenos Aires

A terminology server allows linking the free text entered by a physician in a medical record to different health classifications, such as ICD9-CM, ICD10, or LOINC (González et al., 2018). The use of a terminology server allows clinical information to be recorded in a structured form, using clinical terminology standards. Terminology standards enable interoperability of clinical information, and also allow information to be reused for other purposes, such as clinical decision support, data analysis and research.

The proposed system is able to use the terminology server supported by HIBA. The terminology server publishes its terminological terms grouped in different domains. The implemented prototype has been successful in using terminology services for the domains that cover: reasons for consultation, diagnoses, procedures and studies, which are required for the Uruguayan medical records model.

6.6. Summary of results

This chapter detailed the strategy by which the instances used to evaluate the proposed approach were generated. The generation process was not simple and required the collaboration of advanced medical students.

The performance of the proposed learning method was analyzed when facing CKBs with several CTs. Despite the lazy nature of the proposed method, the results showed that the learning method was able to produce suggestions in reasonable execution times.

An evaluation was performed on a real use scenario, where some advantages of the proposed approach over the original Praxis were observed. The results were also better when using a pre-loaded CKB, containing typical workday clinical scenarios. A survey performed on several advanced medical students showed a high rate of approval of the proposed method, highlighted as an appropriate approach for medical practice and useful at medical consultations.

As another relevant result, the use of health standards was successfully incorporated into this Master Thesis, aimed at the interoperability of clinical information.

Chapter 7

Conclusions and future work

This chapter presents the conclusions of this research and also outlines the main lines of future work for improving current results.

7.1. Conclusion

This Master Thesis presented a novel approach to represent clinical knowledge, which supports an appropriate methodology for recording medical consultations. An instance-based learning method was also proposed, aiming at providing suggestions for physicians during their medical assistance. All suggestions are based on information registered by the physician during his previous medical consultations.

The potential of Praxis software to lay the groundwork for a new model of writing medical records was identified during the review of related works. A formal structure capable of supporting all the functionalities of original Praxis was proposed. Different scenarios of medical consultations were also modeled to address the diversity of situations of physician workday, including multiple diagnoses and the attention of chronic patients.

In addition to the fact that the proposed model provides flexibility to write free text for recording all particularities of a medical consultation, it was also designed to support standards of health terminology and codifications. Standard codes of health terminology were successfully integrated into the proposed model. In particular, the implemented system is able to interact with the HIBA terminology server, and to access the national drug dictionary of Uruguay.

The approach was validated on a real case study involving 250 real instances

constructed by advanced medical students of Universidad de la República, Uruguay. The students collaborated in the construction of new case types, which were built based on a multi-authored publication that covers a wide range of clinical scenarios of primary care.

Several indexing and caching techniques were implemented with the aim of reducing the execution time of the learning method. In this regard, strategies for caching similarity values and discarding non-promising candidates were successfully applied. Optimizations to reduce the execution time of the learning method over singular scenarios were also implemented.

The proposed instance-based learning method was able to generate suggestions in reasonable execution times, even faced with large volumes of data. For the addressed case study, a total of 62% of the advanced medical students reduced the writing time of their consultations, which demonstrated that the approach was useful to accelerate the registration process during clinical assistance. Results of the evaluation performed in several editions of a Medical Informatics course at Universidad de la República suggest that the proposed approach was appropriate to follow physician reasoning, especially during medical consultations. More than 73% of advanced medical students surveyed approved a prototype which follows the proposed approach.

By applying the proposed approach, the physician thinking is anticipated, especially regarding recurring scenarios. The physician can use his own previous records as a checklist, to avoid forgetting important questions and to consider all repetitive tasks. Consequently, the proposed adaptive structure supported by the learning method contributes to generate medical records faster than when using mainstream EMR systems, based on rigid templates. Overall, the proposed approach is a first step to explore new ways to foster physician thinking, by considering information of previous similar scenarios. The approach emulates the reasoning and evolution of physician thoughts during patient consultations, to overcome the difficulties of template based clinical systems, not designed with a medical approach.

7.2. Future work

The main research lines for future work are related to evaluate the proposed system in a professional work environment of healthcare attention, with the aim of improving the accuracy of the learning method based on professional

feedback. Thus, a future work line includes studying the proposed approach with the help of professional physicians.

Another possibility for future work is related to enhance the accuracy of the learning method by improving the comparison between clinical phrases. To enhance comparison, accurate values of similarity between clinical phrases must be considered, rather than simply differentiating similarity by binary values. In addition, natural language processing will allow to implement a more sophisticated comparison, supporting different languages. In addition, another future work is related to creating a tool to ease the task of associating clinical phrases (represented by units of thought) to health code standards.

The proposed learning method is based on a similarity metric which is weighted calculated, according to the weights of the clinical sections of case types. In the context of this Master Thesis, a simple medical relevance criteria was applied during the experimental analysis, based only on qualitative ranges of medical relevance. Consequently, more accurate weights of medical relevance will allow to enhanced current results.

Integrations with external prescription applications, such as SEPEPE (Rey et al., 2020), are additional lines for future work, since prescription applications are able to track patients between medical consultations, which contributes to improving healthcare.

Bibliography

Barbantan, I., Porumb, M., Lemnaru, C., and Potolea, R. (2016). Feature engineered relation extraction - medical documents setting. *International Journal of Web Information Systems*, 12(3):336–358.

Benmimoune, L., Hajjam, A., Ghodous, P., Andres, E., Talha, S., and Hajjam, M. (2015). Hybrid reasoning-based medical platform to assist clinicians in their clinical reasoning process. In *6^{th} International Conference on Information, Intelligence, Systems and Applications*, pages 1–5.

Boyd, K., Eng, K. H., and Page, C. D. (2013). Area under the precision-recall curve: Point estimates and confidence intervals. In *Machine Learning and Knowledge Discovery in Databases: European Conference, Part III*, pages 451–466. Springer Berlin Heidelberg.

Campbell, M. J., Machin, D., and Walters, S. J. (2007). *Medical Statistics: A Textbook for the Health Sciences*. Wiley.

Castellano, G. and Casalino, G. (2020). Special issue "computational intelligence in healthcare". *Electronics*, 9(7).

Chang, T. M., Kao, H. Y., Wu, J. H., and Su, Y. F. (2016). Improving physicians' performance with a stroke cdss: A cognitive fit design approach. *Computers in Human Behavior*, 54:577–586.

Chau, M., Zeng, D., Chen, H., Huang, M., and Hendriawan, D. (2003). Design and evaluation of a multi-agent collaborative web mining system. *Decision Support Systems*, 35(1):167–183.

Collins, C., Baumann, B., Krumlian, K., and Keita, L. (2020). Openclinica. http://www.openclinica.com. Accessed 16 July 2020.

Esteban, C., Schmidt, D., Krompaß, D., and Tresp, V. (2015). Predicting sequences of clinical events by using a personalized temporal latent embedding model. *IEEE International Conference on Healthcare Informatics*, pages 130–139.

Farinelli, A., Grisetti, G., Iocchi, L., Cascio, S. L., and Nardi, D. (2003). Design and evaluation of multi agent systems for rescue operations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3138–3143.

González, F., Otero, C., and Luna, D. (2018). Terminology services: Standard terminologies to control health vocabulary. *Yearbook of Medical Informatics*, 27(01):227–233.

Guidi, G., Pettenati, M., Melillo, P., and Iadanza, E. (2014). A machine learning system to improve heart failure patient assistance. *IEEE Journal of Biomedical and Health Informatics*, 18(6):1750–1756.

Habib, J. (2010). Ehrs, meaningful use, and a model emr. *Drug benefit trends*, 22(4):99—-101.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Infor-med (2021). Praxis web. http://www.praxisemr.com. Accessed 1 April 2021.

Installé, A., Van Den Bosch, T., De Moor, B., and Timmerman, D. (2014). Clinical data miner: An electronic case report form system with integrated data preprocessing and machine-learning libraries supporting clinical diagnostic model research. *Journal of Medical Internet Research*, 16(10):e28.

Klann, J., Szolovits, P., Downs, S., and Schadow, G. (2014). Decision support from local data: Creating adaptive order menus from past clinician behavior. *Journal of Biomedical Informatics*, 48:84–93.

Li, J.-S., Zhang, X.-G., Chu, J., Suzuki, M., and Araki, K. (2012). Design and development of emr supporting medical process management. *Journal of Medical Systems*, 36(3):1193–1203.

Lichman, M. (2013). Uci machine learning repository. http://archive.ics.uci.edu/ml. Accesed 16 July 2020.

Lin, C.-T., McKenzie, M., Pell, J., and Caplan, L. (2013). Health care provider satisfaction with a new electronic progress note format: SOAP vs APSO format. *JAMA Internal Medicine*, 173(2):160–162.

Lin, F. P. Y., Pokorny, A., Teng, C., Dear, R., and Epstein, R. J. (2016). Computational prediction of multidisciplinary team decision-making for adjuvant breast cancer drug therapies: a machine learning approach. *BMC Cancer*, 16(1):929.

López-Rubio, E., Elizondo, D. A., Grootveld, M., Jerez, J. M., and Luque-Baena, R. M. (2015). Computational intelligence techniques in medicine. *Computational and Mathematical Methods in Medicine*, pages 1–2.

Low, R. (2015). The theory of praxis concept processing. Technical report, Infor-Med Corporation.

Michalowski, M., Wilk, S., Tan, X., and Michalowski, W. (2014). First-order logic theory for manipulating clinical practice guidelines applied to comorbid patients: a case study. *American Medical Informatics Association Annual Symposium proceedings*, 2014:892–898.

Mitchell, T. (1997). Instance-base learning. In *Machine Learning*, chapter 8. McGraw-Hill.

Nakai, T., Takemura, T., Sakurai, R., Fujita, K., Okamoto, K., and Kuroda, T. (2016). Prediction of clinical practices by clinical data of the previous day using linear support vector machine. In *Innovation in Medicine and Healthcare 2015*, pages 3–13. Springer International Publishing.

Rane, A. L. (2015). Clinical decision support model for prevailing diseases to improve human life survivability. In *International Conference on Pervasive Computing*, pages 1–5.

Rey, G., Alzugaray, M., Vico, S., Vega, C., and Simini, F. (2020). SEPEPE: Pregnancy follow-up prescription app. In *22º Congreso de Bioingeniería, SABI 2020*. SABI.

Sadegh-Zadeh, K. (2015). *Handbook of Analytic Philosophy of Medicine*. Springer Publishing Company, Incorporated.

Salmon, P., Rappaport, A., Bainbridge, M., Hayes, G., and Williams, J. (1996). Taking the problem oriented medical record forward. *Proceedings: a conference of the American Medical Informatics Association. AMIA Fall Symposium*, pages 463–7.

Shen, Y., Colloc, J., Jacquet-Andrieu, A., and Lei, K. (2015). Emerging medical informatics with case-based reasoning for aiding clinical decision in multi-agent system. *Journal of Biomedical Informatics*, 56:307–317.

Sociedad Andaluza de Medicina Familiar y Comunitaria (2017). *Casos clínicos en atención primaria*. Fundación Sociedad Andaluza de Medicina Familiar y Comunitaria.

Subirats, L., Ceccaroni, L., Maroto, J. M., De Pablo, C., and Miralles, F. (2014). Medical-treatment recommendation and the integration of process models into knowledge-based systems. In *6th International Conference on Agents and Artificial Intelligence*, pages 491–498.

Vessey, I. (1991). Cognitive fit: A theory-based analysis of the graphs versus tables literature. *Decision Sciences*, 22(2):219–240.

Wang, W. and Benbasat, I. (2009). Interactive decision aids for consumer decision making in e-commerce: The influence of perceived strategy restrictiveness. *MIS Quarterly: Management Information Systems*, 33(2):293–320.

Wang, Y., Tian, Y., Tian, L., Qian, Y., and Li, J. (2015). An electronic medical record system with treatment recommendations based on patient similarity. *Journal of Medical Systems*, 39(5):55.

Wilk, S., Michalowski, M., Michalowski, W., Rosu, D., Carrier, M., and Kezadri-Hamiaz, M. (2017). Comprehensive mitigation framework for concurrent application of multiple clinical practice guidelines. *Journal of Biomedical Informatics*, 66:52–71.

Wilk, S., Michalowski, M., Tan, X., and Michalowski, W. (2014). Using first-order logic to represent clinical practice guidelines and to mitigate adverse interactions. In *6th International Workshop Knowledge Representation for Health Care*, pages 45–61. Springer International Publishing.

Wilk, S., Michalowski, W., Michalowski, M., Farion, K., Hing, M. M., and Mohapatra, S. (2013). Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming. *Journal of Biomedical Informatics*, 46(2):341–353.

Zieba, M. (2014). Service-oriented medical system for supporting decisions with missing and imbalanced data. *IEEE Journal of Biomedical and Health Informatics*, 18(5):1533–1540.