A TREATMENT RECOMMENDATION TOOL BASED ON TEMPORAL DATA MINING AND AN AUTOMATED DYNAMIC DATABASE TO RECORD EVOLVING DATA

A Thesis
Presented to
The Academic Faculty

by

Kunal Malhotra

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science in the
School of Computer Science

Georgia Institute of Technology
May 2015

A TREATMENT RECOMMENDATION TOOL BASED ON TEMPORAL DATA MINING AND AN AUTOMATED DYNAMIC DATABASE TO RECORD EVOLVING DATA

Approved by:

Professor Sham Navathe, Advisor
School of Computer Science
Georgia Institute of Technology

Professor Ed Omiecinski
School of Computer Science
Georgia Institute of Technology

Professor Leo Mark
School of Computer Science
Georgia Institute of Technology

Date Approved: 24th Apr 2015

To mom and dad,

for being there selflessly.

ACKNOWLEDGEMENTS

I want to thank Dr.Sham Navathe and Dr.Jimeng Sun for their endless support during my time at Georgia Tech.

I would also like to thank Dr.Leo Mark and Dr.Ed Omiecinski for extremely valued advice.

I would like extend a vote of thanks to Shibani Medhekar who worked with me on the dynamic database evolution project and Malvika Paul for her help in implementing the treatment advisor tool.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The thesis examines sequential mining approaches in the context of treatment recommendation for Gliblastoma (GBM) patients. GBM is the most lethal and biologically the most aggressive forms of brain tumor with median survival of approximately 1 year. A significant challenge in treating such rare forms of cancer is to make the best decision about optimal treatment plans for patients after standard of care. We tailor the existing sequential mining approaches by adding constraints to mine significant treatment options for cancer patients. The goal of the work is to analyze which treatment patterns play a role in prolonging the survival period of patients. In addition to the treatment analysis, we also discover some interesting clinical and genomic factors, which influence the survival period of patients.

A treatment advisor tool has been developed based on the predictive features discovered. This tool is used to recommend treatment guidelines for a new patient based on the treatments meted out to other patients sharing clinical similarity with the new patient. The recommendations are also guided by the influential treatment patterns discovered in the study. The tool is based on the notion of patient similarity and uses a weighted function to calculate the same.

The recommendations made by the tool may influence the clinicians to have the patients record some vital data on their own. With the progression of the treatment the clinicians may want to add to or modify some of the vital data elements previously decided to be recorded. In such a case a static database would not be very efficient to record the data since manual intervention is inevitable to incorporate the changes in the database structure. To solve this problem we have developed a dynamic database evolution framework, which uses a form based interface to interact with the clinician to add or modify the data

elements in a database. The clinicians are flexible to create a new form for patients or modify existing forms based on a patient's condition. As a result, appropriate schema modifications would be done in the relational database at the backend to incorporate these changes maintaining relational consistency.

CHAPTER I

INTRODUCTION

1.1 Importance of Cancer Study

Cancers have become of the leading causes of morbidity and mortality worldwide in the recent past with 14 million new cases reported along with 8.2 million cancer related deaths in 2012 [46]. Early diagnosis of cancer can lead to formulation of effective treatment plans since every cancer type has a different treatment regimen and multiple modalities such as surgery, radiotherapy, chemotherapy, etc. A patient's quality of life is an important goal when deciding on treatments but the primary goal is to cure cancer and prolong the survival period. [47].

1.1.1 Glioblastoma

Glioblastoma (GBM) is the most lethal and biologically the most aggressive brain cancer with patients having a median survival of 12-15 months. [45]. A small percentage of patients survive for longer period of times. Understanding what factors prolong survival and promote treatment responses can be of value to treating physicians.The Cancer Genome Atlas (TCGA) [27], a project of the National Institutes of Health (NIH), led to work done on classifying Glioblastoma into four distinct molecular subtypes which may lead to different treatment regimens [12]. Patients with certain molecular subtypes may have greater overall survival than other patient subtypes and analyzing gene expression levels, copy number variation (CNV), and mutations may give us information about their correlation with survival periods. The current standard of care for new GBM patients involves surgical resection followed by radiation therapy and chemotherapy with the oral alkylating agent Temodar [33].Krex et

al [20] have analyzed newly diagnosed GBM patients undergoing therapy and discovered certain clinical and molecular features which play a significant role in prolonging the survival period. Predictive survival models have been developed in the past utilizing imaging and clinical features of patients [24] but few have comprehensively studied the impact of treatments in addition to clinical features and genomic features. The high mortality rate of GBM patients, where long-term survival is a rare phenomenon, has drawn significant attention to improving treatment of these tumors. After first line standard of care treatment, there are different treatment combinations chosen by oncologists. The sequence in which the next set of drugs or therapy is prescribed adds to the level of complexity since drugs given in a particular sequence may have a better therapeutic effect than the same drugs given in some other order. Furthermore, other drugs such as steroids and antiepileptics are administered in conjunction while treating GBM, which adds another layer of complexity. We believe analyzing the treatment plans of patients from the TCGA may provide insight to certain drug sequences, which may be associated with greater overall patient survival. Based on our knowledge, there is no existing literature that analyzes medication patterns that may influence survival for new GBM patients.

1.2 Predictive Data Analytics

Predictive analytics is a field used to determine the probable future outcome of an event or the likelihood of a current state where it is unknown. Data mining in general helps in characterizing and describing trends and patterns that reside in data and information but it is limited in terms of preditive models. To encompass the ability to anticipate occurrence of events in the future predictive analytics comes into play. It uses a variety of model making tools and algorithms which are used to characterize and analyze historical information to predict the nature and likelihood of future events and occurences [34]. The new and useful insights from the data at

hand helps in making better decisions and verifies the results for continued relevance and accuracy. [7]

1.2.1 Sequential Mining

Sequential pattern mining refers to the mining of frequently occurring ordered events or subsequences as patterns [17]. An example of a sequential pattern in the cancer domain is 'patients who are prescribed chemotherapy are likely to get radiation therapy within 15 days'. Sequential pattern mining has many applications in retail industry for shelf placement and promotions. Many marketing strategies in big businesses are formulated based on knowledege gained from sequential mining. This problem first introduced by Agarwal and Srikant [2] in 1995 based on their study of customer purchase sequences states that "Given a set of sequences, where each sequence consists of a list of events(or elements) and each event consists of a set of items, and given a user specified minimum support threshold of min_sup, sequential pattern mining finds all frequent subsequences, that is , the subsequences whose occurrence frequency in the set of sequences is no less than min_sup". Consider a transaction database S as shown in Table 1. There are four transactions in the database consisting of items a, b, c, d, e, f and g. Consider the subsequence s = <a (a c)>. Assuming the min_sup to be 2, we observe that s occurs in three out of the 4 sequences in the database which is above the threshold support. We call 's' as a significant sequential pattern. This concept led to the development of the GSP (Generalized Sequence Pattern) mining algorithm [2] based on the Apriori algorithm to mine frequent itemsets. It uses the downward-closure property of sequential patterns and adopts a multiple-pass, candidate generation approach. Initially individual items that occur more frequently than a minimum stipulated threshold (known as "support" in these algorithms) are found. Subsequently it combines frequent items after every iteration and prunes the ones occurring below the threshold. The process terminates when no more sequences with

Table 1: Sequence Database

Transaction_ID	Sequence of events
1	<a (a c) (a g) g d>
2	<b (a c) a d (e f)>
3	<a (a c) (a d) f e>
4	<(d g) a (a c) f b>

a longer length and meeting the required minimum support can be found. SPADE (Sequential Pattern Discovery using Equivalent Classes) on the other hand, uses a vertical data format and associates each itemset with an ID_list [48] which is combination of a sequence id and an event id. The former represents the transactions in a database and the latter indicates the time of occurrence of an event in a given transaction. The other sequential pattern mining algorithms are based on 'Frequent Pattern Growth' technique [17], avoiding the need for candidate generation unlike GSP and SPADE which are based on Apriori. PrefixSpan [31] is one such algorithm which exploits this approach by building prefix patterns and concatenating them with suffix patterns to find frequent patterns. SPAM (Sequential PAttern Mining using a bitmap representation) [3] uses a depth-first traversal of the search space and a vertical bitmap representation of the database enabling efficient support counting.

1.2.2 Supervised Machine Learning: Classification and Prediction

Supervised machine learning is a popular technique used in the area of machine learning which deals with inferring a function from labelled training data [23]. There are various algorithms which reason from externally supplied instances and make predictions about the future instances. A concise model is built which is used to assign class labels to testing instances where the values of the predictor features are known, but the value of the class label is unknown. The process of applying superivised machine learning involves identification of required data with respect to the problem at hand followed by pre-processing the data. We define an unbiased training set which would be used by a classifier to learn rules to be applied to a test set which is unlabeled.

Finally the test set is evaluated by one of the validation techniques such as cross-validation, etc [15].

Classification and prediction are two forms of data analysis to extract models describing important data classes or to predict future data trends. Regression analysis is a statistical methodology that is often used for numeric prediction. Data classification involves a classifier being developed describing a predetermined set of classes. In the learning phase the classifier algorithm learns from a training set consisting of class labels for every database instance. The data set can be numeric data, categorical data or a mix of both type but the class labels are categorical. Figure 1. depicts the concept of classification. In the figure a classification algorithm learns and generates certain classification rules from the training data which are in turn applied to a test set created from the original data set. Once the classifier has been found to perform reasonably well, we apply the classifier on new data [17].

1.2.2.1 Logistic Regression

Logistic Regression is a type of probabilistic statistical classification model. It can be used as a binary predictor, to precict outcome of a categorical dependent variable based on one or more predictor variables. It measures the relationship between the categorical dependent variable and one or more independent variables which are continuous. Logistic regression is usually used in the case where the dependent variable is binary and is thus called binomial logistic regression. In circumstances where the number of dependent variables is more than two, it is called multinomial logistic regression. It is analogous to linear regression, except that the dependent variable is nominal, not a measurement. One goal is to see whether the probability of getting a particular value of the nominal variable is associated with the measurement variable; the other goal is to predict the probability of getting a particular value of the nominal

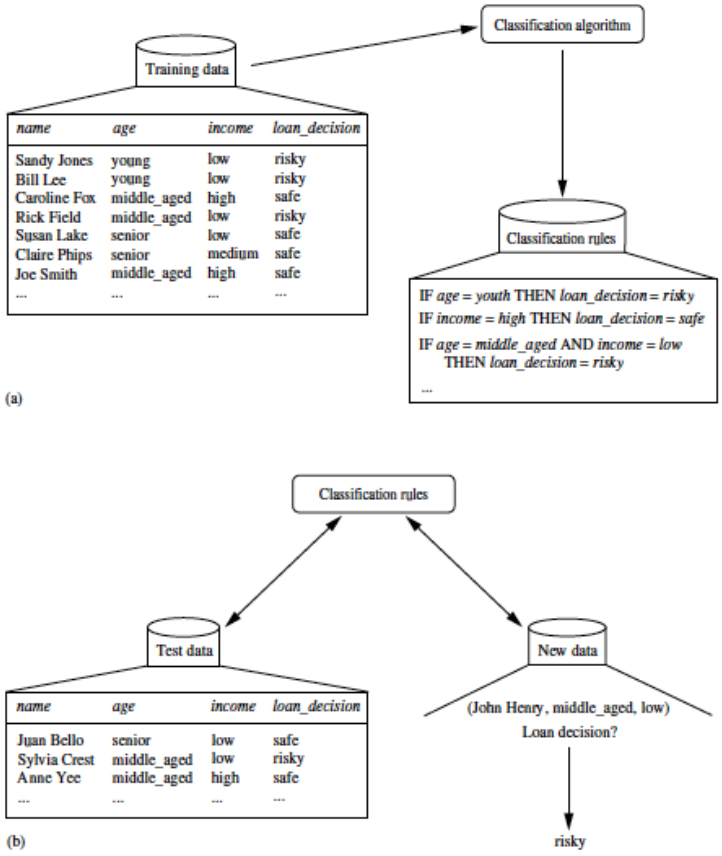

Figure 1: Example of classification [17]

variable, given the measurement variable. This method of classification has applications in many fields including medical and social sciences. It can be used to predict if a patient has a given disease based on observed characteristics of the patient such as age, sex, body mass index, lab test results etc. This underlying function of the this classifier is called the logistic function or the logit function which can take an input from any value from the negative to positive infinity, but the output would be always be between 0 and 1 and thus can be interpreted as a probability. The logistic function is defined as shown in the equation below.

$$\rho(t) = e^t/(e^t + 1) \tag{1}$$

Here t is a linear function of a combinatation of explantory variables and expressed as follows:

$$t = \beta_0 + \beta_1 x \tag{2}$$

Thus the logistic function can be re-written as

$$F(x) = 1/[1 + e^{-}(\beta_0 + \beta_1 x)] \tag{3}$$

1.3 Database Evolution

Schema evolution refers to the problem of evolving a database schema to adapt it to a change in the modeled domain. The problem involves not just studying the affects of changes in the schema but also how such changes affect the stored data and existing queries and stored procedures. In the database life cycle, the schema evolution problem first comes up during the design phase. The evolutive aspect is more prominent in the post-design phase when a schema may undergo modification triggered by a significant evolution of the application domain.

Previously managing schema evolution during the operational phase can be problematic since each schema change needs to take into account previously stored data, which required storing the previous versions of the schema to retain accessibility to the associated data [30]. This problem has attracted a lot of attention in the database community for more than a decade.

It has been observed that a database schema frequently experiences a lot of changes with time [50]. In most of the domains, there is a lack of a set of common data elements, that users would like to store in the database thus requiring different database schemas. In such cases, a static database schema would pose a problem. One approach to modifying the schema would be to have a database administrator track the data elements and periodically modify the schema. This would require a lot of manual labor and periodic updates would eventually delay data entry. In most large

organizations, the DBA staff has been entrusted with dealing with schema management however the costs associated with database management systems and other large-scale application systems such as EMR (Electronic Medical Record) systems tend to be prohibitive for most "small-business", or start-up operations. For most small outfits dealing with a specialty practice with a handful of physicians, adopting large generic systems is prohibitive in adoption, training and maintenance costs. The traditional approach to relational database design starts with the conceptual design of an application based schema in a model like the Entity-relationship model, then mapping that to a logical design and eventually representing it as a set of related normalized tables. With time the user requirements change and the database needs to evolve to meet them. It is also possible that different users in the same organization using a central database have different requirements. E.g Different clinicians in the same hospital may want to collect different sets of data elements for patients suffering for different diseases, with a certain percentage of similarity. In this case we believe that having data in one place is better than creating different tables with redundant attributes. We propose a dynamic database modification system (DDSCM) which can automatically modify the back-end database based on changes made by the user using a simple user interface.

In chapter 2 we describe an analytics pipeline consisting of different modules to perform data standardization, perform sequential mining to extract significant treatment patterns and predictive analytics to predict patients that are highly likely to survive for longer than the median survical period of 1 year. The predictive model of the pipeline uses clinical, genomic and treatment patterns as features and a forward feature selection algorithm extracts the most predictive features. In chapter 3 we present a dynamic database schema modification system which automatically evolves a database at the back-end based on the changes made by the user on the front-end

with minimal human intervention. This system consists of a templates provided to the user to feed in data with the capability to modify the templates to accomodate new data elements. Our approach modifies the back-end database automatically to reflect the changes in the front-end. In chapter 4 we describe a treatment recommendation tool which combines the analytics pipeline and the dynamic database schema modification system into a single architecture. The tool also uses the notion of patient similarity to recommend treatments based on clinically and genomically similar patients. The tool can be used for a different disease by plugging in the predictive model of that disease in the analytics pipeline and modifying the database at the back-end using the dynamic database modification module to store the data about the disease.

CHAPTER II

CONSTRAINT BASED TEMPORAL EVENT SEQUENCE MINING FOR SURVIVAL ANALYSIS OF GLIOBLASTOMA PATIENTS

One of the many challenges in the field of medicine is to make the best decisions about optimal treatment plans for patients. Medical practitioners often have differing opinions about the best treatment among multiple available options. While standard protocols are in place for the first and second lines of treatment for most diseases, a lot of variation exists in the treatment plans subsequently chosen. As a representative disease we study Glioblastoma Multiforme (GBM) which is a rare form of brain tumor. The goal of our study is to predict patients surviving for greater than the median survival period for GBM and discover in addition to clinical and genomic factors, certain treatment patterns which influence longevity. Our study makes the following contributions:

1. We extend existing sequential pattern mining algorithms by incorporating two additional constraints called the 'exact-order' and 'overlap' explained later in the paper to mine significant treatment patterns from the available data.

2. We follow a data-driven approach to build and evaluate a predictive model for treatment effectiveness of GBM patients by treating temporal treatment patterns as features in addition to the existing clinical and genomic features.

2.1 Background

It is important to integrate the clinical data in the EHR with the genomic data of patients with GBM since they have a poor prognosis and have a median survival of

one year.

Sources of Data: TCGA program and cBioPortal

TCGA began as a three-year pilot in 2006 with an investment of $50 million each from the National Cancer Institute (NCI) and National Human Genome Research Institute (NHGRI). The initiative has a vision that an atlas of changes could be created for specific cancer types. It also showed that results could be pooled together from different research and technology teams working on related projects and be made publicly accessible for researchers around the world to validate important discoveries [27].

The cBioPortal [6] is developed and maintained by the Computational Biology Center at Memorial Sloan Kettering Cancer Center and provides visualization, analysis and downloading of large scale cancer genomics data sets. The clinical and treatment related data about patients was obtained from TCGA and the data pertaining to mRNA expression, CNV and methylation status of those patients was obtained from cBioPortal. The mRNA expression levels and CNV data was collected for a specific set of genes which have been observed to play a role in classifying GBM patients into 4 genomic subtypes namely classical, mesenchymal, proneural and neural [44]. The methylation status of the promoter region of MGMT gene was also used for our analysis since it has been observed to have an influence on the survival period of the GBM patients [14].

2.2 Approach

This section gives an overview of the approach used for representing the data and the predictive modeling pipeline developed to predict the survival duration of patients.

2.2.1 Data

We have constructed a rich dataset of newly diagnosed GBM patients by integrating two different databases called the The Cancer Genome Atlas Portal [27] and the cBioPortal [6, 11]. TCGA consists of clinical and treatment data pooled together from different research and technology teams, which is publicly accessible around the world to validate important discoveries. The genomic data for the same patients was obtained from cBioPortal, a web resource for multidimensional cancer genomics data maintained by the Memorial Sloan Kettering Cancer Center.

For our study, we analyzed data from 309 newly diagnosed GBM patients spanning over a period of 2 years. The data was categorized into 'Clinical', 'Genomic' and 'Treatment' domains. The clinical domain includes demographic information about the patient along with some basic clinical features such as Karnofsky performance score (KPS), histopathology, prior glioma history, and whether the patient is alive or deceased. Under the genomic domain, the mRNA expression levels and CNV data was collected for a specific set of genes which play a role in classifying GBM patients into 4 genomic subtypes,namely,'Classical','Mesenchymal','Proneural', and 'Neural' [44]. The methylation status of the promoter region of MGMT gene was also used for our analysis [14]. The treatment domain consists of treatment plans for each patient. We use sequential mining algorithms to mine significant patterns in their treatment plans and use them as features in the dataset in addition to clinical and genomic features. Table 2 summarizes the dimensions of the dataset categorized by the domain.

The goal of this study is to predict patients who survive for greater than 12 months. The pool of patients used for the study consists of living patients who have already survived for more than a year in addition to the deceased patients that constitute the majority of patients.

Table 2: Dataset summary

Domain	Number of features
Clinical	11
Genomic	33
Treatment	49

2.3 Methodology

This section gives an overview of the predictive analytics pipeline developed to predict long term surviving patients.

2.3.1 Predictive Analytics Pipeline

The pipeline consists of 4 modules, namely, 'Data Standardization and Cleaning', 'Sequential Pattern Mining', 'Feature Construction' and 'Prediction and Evaluation'. As shown in figure 2, the raw data is fed into the 'Data Standardization and Cleaning' module to filter out noisy data. The 'Sequential Pattern Mining' module extracts significant medication patterns from the treatment data. The clinical and genomic features are combined with the treatment patterns to form a binary feature matrix in the 'Feature Construction' module, each row corresponding a single patient. It also assigns a target variable for every patient. The 'Prediction and Evaluation' module selects predictive features and performs classification to predict the long term surviving patients.

2.3.2 Data Standardization and Cleaning

Data standardization is one of the most important and time consuming steps when building predictive models. Every hospital contributing data to TCGA uses a different format to store data and in some cases a different nomenclature for some data elements. Missing data is another common issue and needs to be imputed if data is missing for some of the data elements instrumental for our analysis. The data standardization module identifies these different data formats, missing values, and

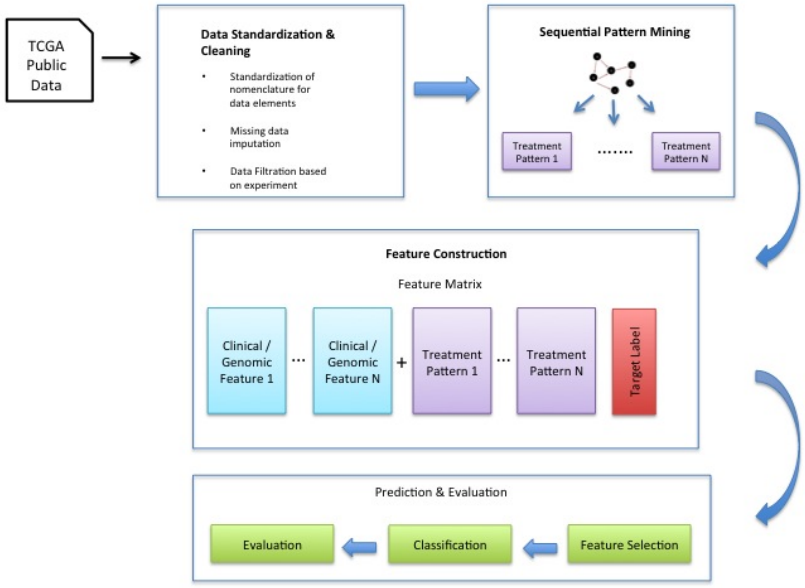

Figure 2: Predictive Modeling Pipeline

creates a standardized clean data set for further analysis. For instance, 10% data had missing values for either start or end dates of specific drugs which were imputed by computing the mean duration of that drug for other patients; Drug names were 'standardized'- e.g., Temozolomide was changed to Temodar; A value of 'Completed' in the additional chemotherapy prescribed column was changed to 'Yes'.

2.3.3 Sequential Pattern Mining

The data used for this study is modeled as a graph consisting of nodes, categorized as 'patient node' and 'treatment type node' and edges categorized as, 'prescription edge' & 'sequence edge'. A graph offers a much richer picture of a network, and allows relationships of several types. Since the data model has a path-oriented nature, the majority of path-based graph database operations are highly aligned with the way in which the data is laid out hence increasing the efficiency [36]. Figure 3 shows the current representation of the data as a graph consisting of two patients just for illustrative purposes. The patient nodes have properties such as 'patient id', 'age at diagnosis',

etc. Prescribed drugs and radiotherapy are represented as treatment type nodes with properties 'drug name' and 'radiation type', respectively. The undirected 'prescription edge' signifies the prescription of treatment with properties corresponding to the prescription. The 'sequence edge' is a directed edge signifying the sequence in which drugs or radiation were prescribed. E.g., The edge labeled 'Prescribed' between the patient node with 'id = Patient_1' and the drug node with 'drugName = Drug_A' signifies that 'Patient_1' was prescribed 200 mg/day of 'Drug_A' between 05/21/2007 and 06/22/2007. The edge labeled 'Followed_by' would always be between a radiation type and a drug, two drugs or two radiation types, signifying the sequence of the prescription. E.g., the 'Followed_by' edge between source node 'Drug_A' and target node 'Drug_B' with properties 'patient' and 'overlap' signifies that for 'Patient_1', Drug_B followed Drug_A with an overlap of 24 days. Sequential pattern mining was used to extract patterns from the treatment data to give three types of information for every patient: the sequence of drugs/radiation prescribed, their time of prescription within the sequence and duration of prescription.

A treatment plan for a patient may consist of a combination of multiple drugs or radiation or both prescribed in a particular sequence. We define a treatment for one patient as a sequence of events, each event consisting of administration of a treatment type (drug or radiation). To mine such treatment plans we tailor existing approaches such as GSP [2] and SPADE [48] by adding two new constraints, namely, 'exact-order' and 'overlap' constraints (explained in the following sections). We define a concept of 'N-path event set', consisting of a sequence of 'N+1' events (treatment instances) joined by 'N' sequence edges. E.g., Drug_A ->Drug_B ->Radiation_D is a 2-path event set from the graph model shown in figure 3, consisting of two drugs and a radiation therapy forming a sequence of consecutive events (represented as nodes), which is not a hard restriction in any of the existing algorithms. Since a treatment plan for a patient may consist of a drug being prescribed more than once, an event identifier

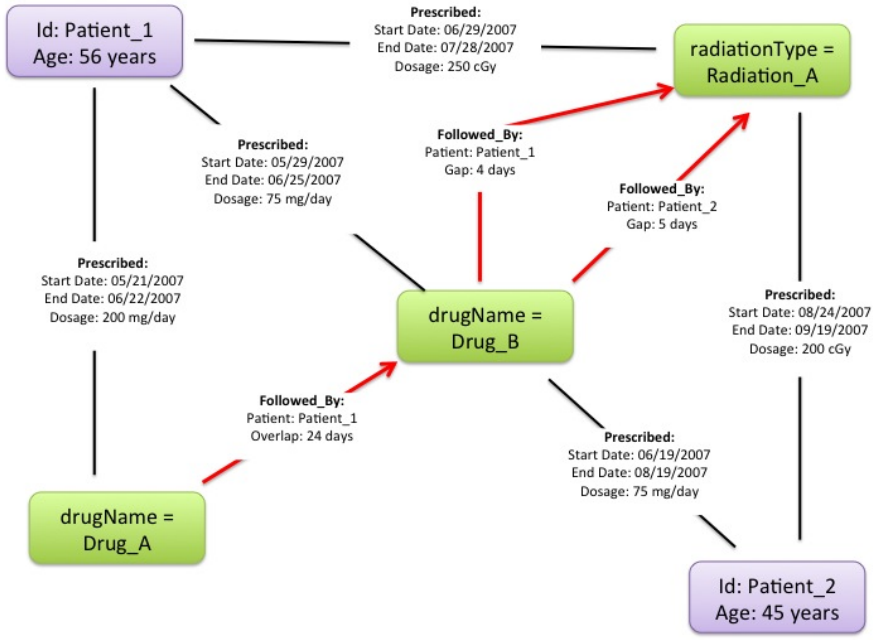

Figure 3: Data represented as a graph

is added along with each treatment type node to execute the 'exact-order' constraint.

2.3.3.1 Exact-Order Constraint

This constraint forces only those events to be a part of sequence, which occur consecutively. E.g., if there are multiple drugs (excluding A and B) given between 2 drugs 'A' and 'B', then a sequence <A B >does not occur.

Candidate Generation : We extract path sequences of length 'N' from the data graph and consider the ones prescribed to a significant number of patients for further analysis. This is followed by forming 'N+1' path sequences, by increasing the path-length one edge at a time, and joining on the event ids and the treatment type nodes. Figure 4 illustrates the candidate generation step, each node representing a treatment type or a combination of treatment types. Initially a 1-path set consisting of combinations of two consecutive treatment types is extracted from the treatment graph such that the sequences should have been prescribed to a significant number

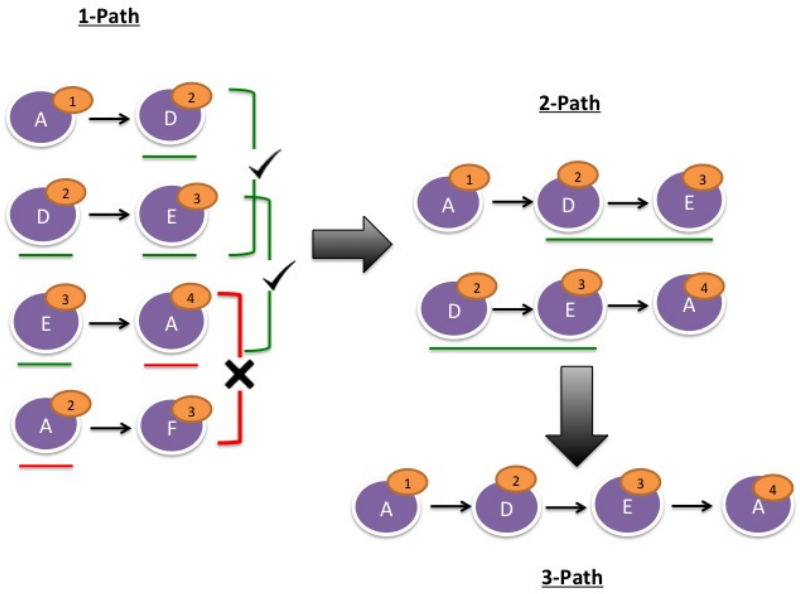

Figure 4: Illustration of candidate treatment pattern generation

of patients. From these 1-path sets, we form 2-path combinations by joining on the treatment type node and the event id. The combinations bracketed in green have the potential to be joined since the resulting sequence has consecutive events. The ones bracketed in red are not joined since event 'A' as the fourth event is different from event 'A' as the second event. This process continues till we cannot form new combinations or they are insignificant.

2.3.3.2 Overlap Constraint

A 'treatment plan' for a patient consists of all the treatment types prescribed to a patient in a sequence. When a treatment is in effect and another one starts concurrently, an overlap of treatment occurs. Two common situations are 'partial overlap' and 'total overlap' of prescriptions. We say 'n' prescriptions (n >1) have a partial overlap if all the 'n' prescriptions are concurrently prescribed to a patient on at least one day. A total overlap is a special case of partial overlap that occurs when all the 'n' prescriptions have the same start and end dates. Approaches called the "Single

Node" and "Combination Node" approach are formulated. In the 'Single Node' approach a sequence considers each treatment type as a single node as shown in figure 5(a). If there is a partial overlap between two prescribed treatment types, the prescription which ends first becomes the source node and a directed edge connects it to the other prescription. E.g. A directed edge connects Dexamethasone to Radiation. In case of a total overlap between two prescriptions, both prescriptions are treated as source nodes, with directed edges to the next treatment type node. E.g There are directed edges from Temodar and Radiation towards CCNU. The 'overlap' constraint, refers to an overlap between multiple drug prescriptions and results in combining those drugs into a single node and treating it as a single event. Under the approach called the 'Combination Node' approach shown in figure 4(b) a new node is created whether there is a partial or a total overlap between treatment type prescriptions. The timeline shown in the figure signifies the order of prescriptions. The purple colored nodes represent individual drugs and the green nodes are created to signify overlapping prescriptions. The current approach is based on data about GBM patients and would be enhanced for other diseases having extensive treatment guidelines and possibly incorporating potential complications.

2.3.4 Feature Construction

We develop a feature matrix with a feature vector per patient and a target variable that represents the targeted outcome of treatment. The clinical and the genomic datasets consist of both numeric and categorical data types. To standardize the data set and avoid creating a bias, we converted our dataset into a binary feature matrix. In addition to these features we add to the feature vector each significant sequential patterns of treatment as a feature. A value of 1 is assigned to this feature for patients who exactly received that treatment and for others it is set to 0. The target variable in our study is constructed based on the patient's survival period. Deceased patients

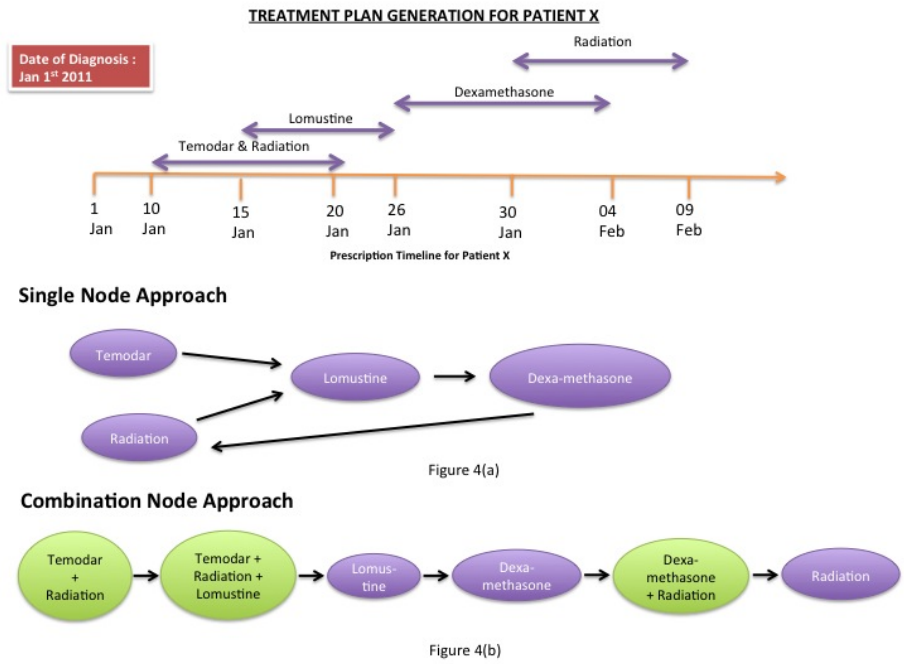

Figure 5: Approaches for Treatment Plan Generation. Fig. 5(a) Single Node Approach; Fig. 5(b) Combination Node Approach

who survived for more than a year are assigned a target variable of '1' and those who survived for less than a year are assigned '0'. For living patients, if their last follow up date was after one year of diagnosis, they were assigned a target variable of '1' otherwise that patient is discarded since there is no positive conclusion about survival period.

2.3.5 Prediction and Evaluation

Our goal is to analyze the clinical and genomic characteristics and prescribed treatments within the first year of diagnosis and predict the survival period of patients. Cross validation is used to partition the data into a training set and a test set multiple times and evaluate the classifier. The sequential treatment patterns are extracted from the treatment plans of patients in the training set and used as features along with other clinical and genomic features. A Logistic Regression based classifier is

trained based on the data at hand. Forward feature selection is used to select predictive features and the prediction performance is evaluated by c-statistic and accuracy. These features are ranked based on the number of folds in which they were selected by the greedy algorithm and their p-value.

2.4 Results

2.4.1 Setup

The data used in this study was classified into two classes based on survival period, namely, i) patients surviving less than a year and ii) patients surviving more than a year. The survival period for deceased patients can be calculated as the number of days a patient survived since the day he (from here onwards we would be using 'he' to refer to a patients and clinicians without any intended bias) was first diagnosed till death. From the pool of living patients, the ones who were last seen after one year of diagnosis were also added to the 'survival greater than a year' category. Before the predictive modeling module, forward feature selection was used to pick only significant features for the experiment.

2.4.2 Quantitative Analysis

In Table 3, we report the performance of various models using both 'single node' and 'combination node' approaches with different mixes of features. C-statistic and accuracy from both logistic regression and cox regression methods have been reported. Amongst the single domain models the best performance is obtained when only the genomic features are considered. Inclusion of more features increases the prediction accuracy as well as the c-statistic (see table 3). Among the multiple domain models, the best performance is achieved when features from all three domains are analyzed together. Table 4 shows the predictive clinical and genomic features along with influence they have in prolonging overall survival beyond 1 year. The predictive treatment patterns shown in table 5 contain treatment events, which consist of

the drug/radiation type with the event identifier in curly brackets categorized by the approach used to form sequences. The bracketed number in the treatment patterns indicates the order number in the event sequence in which the drugs were prescribed.

Table 3: Performance of various models in predicting patients surviving for >1 year using Logistic Regression (LR) and Cox Regression (CR).

	Single Node Approach				**Combination Node Approach**			
Individual Domain Models	**C-Statistic**		**Accuracy**		**C-Statistic**		**Accuracy**	
	LR	CR	LR	CR	LR	CR	LR	CR
Genomic	0.76	0.75	78.1 %	78.0 %	0.78	0.77	79.5 %	80.1 %
Clinical	0.71	0.71	72.2 %	72.3 %	0.72	0.72	73.4 %	74.0 %
Treatment	0.69	0.70	71.2 %	72.0 %	0.60	0.61	63.3 %	63.1 %
Multiple Domain Models								
Clinical + Genomic + Treatment	0.85	0.84	86.4 %	86.7 %	0.85	0.84	86.2 %	87.0 %
Treatment + Genomic	0.84	0.86	84.8 %	84.3 %	0.78	0.78	81.0 %	81.1 %
Clinical + Genomic	0.83	0.83	84.5 %	84.7 %	0.84	0.83	84.5 %	85.2 %
Clinical+ Treatment	0.78	0.79	78.6 %	77.8 %	0.75	0.76	74.5%	74.3 %

Table 4: Predictive clinical and genomic features from the model: Clinical + Genomic + Treatment

Predictive Features	**Percentage of times selected (%)**	**Influence on Survival >1 year**	**P-value**
Genomic			
Unmethylated MGMT promoter region	40	-	0.05
High expression of TP53 gene	40	-	0.03
High expression of GABRA1 gene	40	+	<0.0001
Clinical			
Patient's age at diagnosis between 25 & 50 years	40	+	0.018
Karnofsky performance score >70	40	+	0.02
Prescription of Neoadjuvant therapy	30	+	0.002

2.4.3 Qualitative Analysis

In the genomic domain, GBM patients having an unmethylated promoter region of the MGMT gene are less likely to survive for more than a year. Our results confirm prior literature on MGMT methylation status and overall survival. An extensive

Table 5: Predictive treatment patterns

Predictive Treatment Patterns	**Percentage of times selected (%)**	**Influence on survival >1 year**	**P-value**
Single Node Approach			
Radiation Therapy{2} ->Treatment Termination	50	-	0.0061
Lomustine{2} ->Treatment Termination	40	-	0.05
Procarbazine{2} ->Treatment Termination	30	+	0.05
Temozolomide{2} ->Lomustine{3}	40	+	0.04
Combination Node Approach			
Temozolomide{1} ->[Temozolomide +Radiation]{2}	30	-	0.05
[Temozolomide + Radiation]{1} ->Temodar {2} ->Lomustine{3}	30	+	0.04

literature study shows that temozolomide (Temodar), a standard of care chemotherapeutic, is more effective if the promoter region is methylated [14, 35] leading to better overall survival with 25% of patient surviving 2 years. A higher expression of TP53 gene is associated with shorter survival periods as opposed to higher expression of the GABRA1 gene which is associated with longer survival. GABRA 1 gene also called the gamma-aminobutyric acid (GABA) A receptor, alpha 1. It is characteristic of the neural subtype of Glioblastoma patients observed to have survived longer than other genomic subtypes. In the clinical domain, younger patients, in the age group of 25-50 years, have a higher chance of surviving longer.. KPS is a score assigned by clinicians to GBM patients based on their functional status prior to treatment [28]. Patients having a score greater than 60 survived longer than a year. Another predictive clinical factor is neoadjuvant treatment which is given as the first step to shrink the tumor before the main treatment is begun. Patients receiving neoadjuvant treatment were found to survive for longer periods. (Neo adjuvant drugs include PolyLCLC, Mivobulin isethionate, Oxaliplatin, O6-Benzylguanine and Carmustine). Most importantly,

our study also discovered treatment patterns, which have had both positive and negative effects on the survival period. The standard first line of treatment consists of surgery followed by fractionated external beam radiation therapy (EBRT) with concurrent and adjuvant temozolomide therapy. This combination is associated with the best survival in GBM patients and is the standard of care. Fractionated radiation is given solely for some patients if they cannot tolerate chemotherapy. The single node approach does not take into consideration overlapping prescriptions, thus using this approach, we have found that if treatment consists of prescribing EBRT or the chemotherapeutic CCNU, by itself or along with another drug as the second event in the treatment timeline, then the likelihood of surviving longer is less. This can be explained by patients having unresectable tumors. As a result, prescribing EBRT may not be effective and does not lead to greater overall survival. CCNU prescribed as the second drug in the treatment is also unusual since most clinicians prescribe the standard of care temozolomide treatment and CCNU is not prescribed early. Two treatment patterns were found to have a positive influence on the survival period, one consisting of Procarbazine prescribed second in the treatment plan in combination with other drugs or by itself followed by termination of treatment and the second one consisting of temozolomide prescribed second in the treatment plan immediately followed by CCNU.

Using the combination node approach, we found that if temozolomide is prescribed individually as the first event followed by temozolomide with concurrent EBRT then there is a negative effect on survival. We believe this could be due to the explanation given before about patients not having a resectable tumor or it is also possible that if radiation therapy is not coupled with Temodar as the first event, which is the standard of care, then the treatment does not turn out to be effective. The other predictive treatment pattern, which we have found to have a positive effect on survival, is temozolomide with concurrent radiation therapy followed by temozolomide

prescribed individually which is in turn followed by a prescription of CCNU as the third event.

2.5 Conclusions

In this chapter, we discuss a pipeline performing data standardization, mining sequential treatment patterns, and constructing features of predicting GBM patients surviving for longer periods (greater than 12 months). Sequential mining is applied to treatments consisting of drugs and radiation, which are termed as events. In the sequential mining module, we combine GSP and SPADE algorithms and tailor the approach to account for the exact-order and the overlap constraints. Other than clinical and genomic features, treatment patterns extracted from the treatment plans of patients within one year of diagnosis were used as features to predict patients surviving longer than a year using logistic regression as the classifier and forward feature selection for selection of predictive features. This study is a preliminary step in providing extensive treatment guidance to oncologists and neurosurgeons about the efficacy of certain sequence of drugs and therapies as part of a treatment plan. Currently, the treatment patterns consist of the drug names and their event of prescription. We are developing a treatment advisor tool to recommend treatments for a patient based on treatments given to patients having a similar clinical and genomic profile using the knowledge of treatment patterns obtained from this study. We also plan to add more constraints in the model such as a 'gap' constraint, which would limit the temporal gap between events for inclusion in a sequence. We believe this would help in filtering out clinically insignificant treatment patterns.

CHAPTER III

DDSCMS: A FORM BASED DYNAMIC DATABASE SCHEMA CREATION AND MODIFICATION SYSTEM

Nowadays in every domain, the changes in the user requirement are observed very frequently. Also different users in the same organization may have different requirements and it can be hard to come up with the list of common data elements to be incorporated in the database. A database administrator is usually responisble to manage the database schema at the back-end to reflect the changes in the user requirements. We come up with a form based dynamic database evolution system which automatically modifies the database at the back-end based on the changes made by the user on the front-end to minimize the intervention of a database administrator. We propose an approach to:

1. Create schemas based on predefined forms
2. Update and customize schemas as per changing user needs
3. Align the back-end storage of data as the schemas evolve

Our primary goal is to reduce user intervention and to let the database "evolve" consistently as time progresses. For developing a generically applicable system, we base it on the relational model. Our approach to dynamic schema creation and management has been described here in the context of healthcare just for illustrative purposes. This project was motivated by our interaction with a local neurosurgery practice through Dr.Laborde, a neurosurgeon, who convinced us that there is merit to developing approached to "ad-hoc" database creation and management for applications where elaborate and costly solutions like EMR and EHR (Electronic Health

Record) systems are an overkill.

We made some assumptions while developing the prototype. Our current implementation has been designed to cater to clinical researchers and physicians who want to use existing data for clinical trials and studies and want to add some parameters of their own. Very little knowledge about database modeling and query languages is assumed. The forms could then be made available to the end users such as patients. The users (physicians, nurses, etc. but not patients) are provided with predefined forms developed based on the underlying database schema. These forms are automatically generated based on the metadata, which is stored in a separate database. This process is discussed later in the chapter.

The users can customize these existing forms by adding and deleting data elements of their choice. As a result of this the tables in underlying database will undergo appropriate schema modifications as discussed later in the chapter. In our test implementation, we have used a neurosurgery application database from a local clinic called the ALIF (Anterior Lumbar Inter-body fusion) database [38]. We dealt with the ALIF data with Dr.Laborde's expertise as a domain expert in the specific specialty which deals with surgical procedures of the spine. The term "user" will apply to physicians , nurses, researchers etc. who are knowledgeable about the application domain, who can evaluate the suitability of existing forms and who can be guided in their choices when they undertake to modify the forms. Totally naive end users will not be a target audience for our approach.

3.1 Technical Challenges and Claims

One of the major challenges we faced while developing this system was to avoid anomalies or inconsistencies at the back-end when the user makes changes to the

forms provided for entering the data. The traditional approach to constructing a relational database application involves building a conceptual schema of the relational database using a model like the extended entity-relationship model and then constructing a relational database schema [42]. For most advanced applications the schema of the relational database changes during development. We are interested in providing a solution to environments where the database schema needs to be adjusted in real time keeping all constraints and rules of a relational database intact while the user makes changes to the existing forms or creates new forms. This involves maintaining a metadata database to store metadata about the forms (FORMS DATABASE) and a domain specific database (DSD) to store the actual data entered by the user. The system also provides an easy to use Form Field Selection feature, which can be used by novice users to build their own forms.

A dialog based UI is designed to create a new database from scratch or modify an existing one. It would gather information about the nature of the form field (label in the form) being added which would in turn lead to appropriate changes in the schema of the DSD. Addition of a form field leads to formation of a new attribute in the appropriate table in the DSD. For the data which already exists in the database before adding a new field, the system populates default values for that particular field. Deletion of the form field does not lead to deletion of the corresponding attribute from the schema. These deletions are tracked and recorded in a metadatabase, and a customized form is presented to the user. Modification of a form field does not modify the field name at the back-end. Mappings are created between the field at the back-end and the ones in the UI. A new user may choose from any of the existing customized versions of a form and select the most appropriate one or he (henceforth we will use the pronoun "he" for the user without any intended bias) may customize them further based on his requirements. In order to ensure a smooth working of the

system after plugging in any DSD which stores the data entered via the forms, the FORMS DATABASE stores the metadata of all forms and also the information about what tables a form is connected to.

A user is typically shown all available forms, which guide the user to use one that comes close to their requirements and to modify it. They are also given the option to create a form from scratch. Our algorithms for storing metadata, for defining the schema for the back-end, for storing actual data as well as displaying user defined forms are generic. We propose the mechanism by which a metadata layer called the FORMS meta-database is created to accommodate the current form definitions and subsequent changes to it. In this approach, a user can choose to display data elements that he would like to view together without providing an SQL query. At the back-end the algorithm performs joins between tables based on primary key-foreign key relationships and displays data elements requested by the user. The user can also request the system to perform aggregation operations on data elements, add conditions as well as sort the results. This is particularly geared for researchers who would want to do studies or clinical trials using patient treatment data related to a specific drug as the DSD. They would then create a new database after appropriate aggregation to define a cohort of patients.

3.2 Approach to Dynamic User Interface Management

The user interface of the system is a form-based UI since its functionality is guided by a set of forms. The user is provided with two modes; one of them deals with 'Data Entry' and the other with 'Data Retrieval'. Data entry can be done by either using the existing template forms provided by default based on the current state of the DSD, or the existing forms may be customized to suit new requirements before being used to insert data. The users can also create new forms if the requirements for the

data they want to enter are not met. These changes are automatically translated into appropriate modifications at the schema level.

A large set of options is provided to the user as to what type of form field he wants to create for his form. The options are Radio buttons, Checkboxes, textboxes, drop-downs and buttons. Any of these options apply while modifying existing forms or creating new forms. On creating a new form field, the user is required to provide details about its data type, default value, ability to have multiple values, etc. helping the system to make appropriate modifications in the database. Similarly, creation of a new form requires some information from the user which helps in creating a new table at the back-end. E.g. information regarding its relationship with the existing forms, cardinality of the relationship, etc. The UI has help buttons to describe the semantics of these questions in simple language with examples.There may be a class of users such as researchers that may use an existing large database, create a form using our tool and then modify the form slightly to add some fields/attributes of their own using the data entry function.

The other important feature of this system is 'Data Retrieval' which is used for retrieving data by generating queries based on user input. The operations currently handled by the system are 'Select', 'Aggregation', 'Group By', and 'Having'. Appropriate selections may be made by the user and on doing so the system builds the query and displays the results after validation. Figure 6 illustrates the process flow adopted by the system.

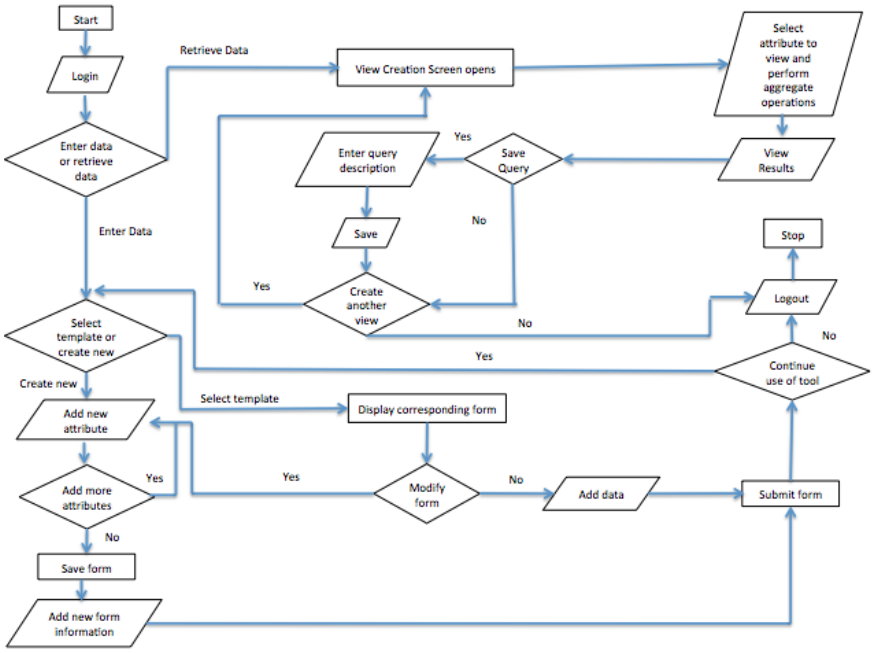

Figure 6: Process flow of the system

3.2.1 Data Entry

The user is presented with template forms based on the underlying DSD. He has a choice of either using the available forms to enter data or customize the forms appropriately. In the latter case he uses the Form Field Panel which displays the potential types of form fields available. A dialog is initiated with the user, which requires him to provide details about the form field chosen. For example, if a radio button is selected for a field 'Gender', then he is required to enter the number of options he wants to keep for this field and their corresponding labels. The new form field added to a form would become a new attribute in the corresponding table having the rest of the form fields of that form after the user provides information about this new attribute using an 'Add New Form Field' screen. This information consists of the label of the form field, it's data type, e.g Integer, String, etc. The user would also need to specify a default value of the field which would be stored if the field is left blank while entering data. This new field may have multiple values e.g If the new

field which is getting added is 'Symptoms' then it may contain multiple values since a patient may have multiple symptoms. This can be specified in the 'Add New Form Field' screen. Each modified template form would be stored as a new version of the existing template and will be annotated with the username of the user modifying it. E.g 'Form_A_User1' is 'Form_A' modified by 'User1' and 'Form_A_User2' is 'Form_A' modified by User2. All the versions of a template are shown to every user from which he can select one for data entry.

The user may create a new form from scratch for which he would be required to provide some information about the new type of data he would like to store (See Figure 7). As mentioned before the '?' marks alongside each label would assist the user in filling this form. The 'Form Name' is the name of the new form the user wants to create. The 'Unique Form Field' would be the field which uniquely identifies an observation. This is equivalent to the key for that form. Initially this field would be empty since it is a new form and no fields have been added. The user would use the 'Add New Form Field' screen to add fields to this form. The system allows selection of multiple fields for the cases when a combination of more than one fields uniquely identifies an observation. The 'Related to Form' field would require the user to choose from a list of existing forms to which this form is related to. As shown in the figure 'Patient' form has a relationship with 'Visits' form in the context of patients having visits in a hospital. The 'Cardinality' can have values 1:1, 1:N and M:N according to the standard cardinality ratio concept in database modeling [8]. Based on the nature of the relationship between the forms the user needs to select the cardinality. In the example shown in figure 2, the selected cardinality 'M:N' denotes that a single patient may undergo multiple procedures and a procedure may be done on multiple patients. The label 'Is there any Attribute for the Relationship' requires the user to specify attributes which are descriptive of the relationship. E.g.

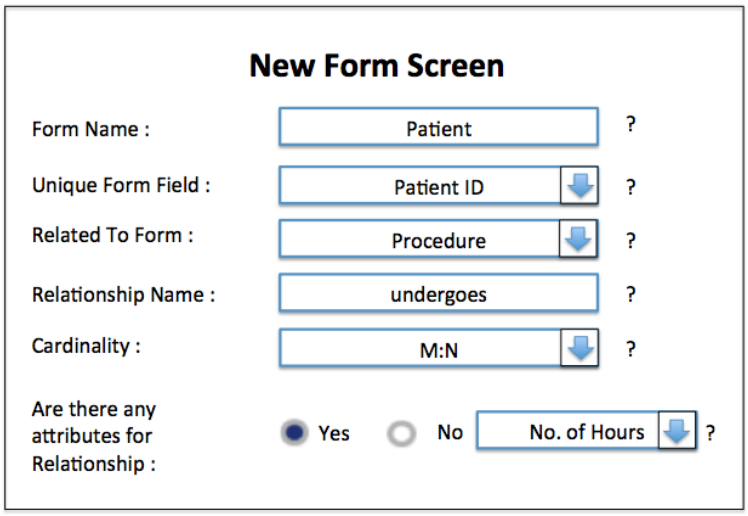

Figure 7: New Form Screen

'No. of Hours' are the the number of hours that the patient undergoes a particular procedure. It is neither an attribute of the 'Patient' since a patient may undergo multiple procedures, each for a different amount of time nor of the 'Procedure' since each procedure may be done on different patients, each time for a different number of hours. It is describing the instance of the relationship 'undergoes' between 'Patient' and 'Procedure'. Multiple attributes are allowed via the drop-down menu. Based on this information the schema of the DSD is modified by adding a new table at the back-end and relating it to appropriate tables based on the rules discussed in [8, 18]. Appropriate changes are made in the FORMS DATABASE simultaneously. These rules help creating a new table and relate it to appropriate tables in the database maintaining consistency.

3.3 Approach to Dynamic Schema Management and Maintenance

Our system has two databases, one is the DSD, which primarily stores the actual data, that is entered by a user using the forms provided, and the other one is the FORMS DATABASE that stores the metadata about the forms. The FORMS DATABASE

is the one primarily responsible for the dynamics of the system and is explained in more detail in the following section.

3.3.1 Meta-database Schema

This database (see Figure 8) consists of all the information about the form structure such as the form name, the fields it consists of, the label of each form field, the type of form fields, etc. needed to build a form. The arrows in the figure stand for foreign-key to primary-key referential integrity constraints. In our present implementation the users are provided with already existing form templates pertaining to the data for a local neurosurgery practice. The database we used for our test was the already pre-populated ALIF database with information about the template forms.

The table 'Form' records a list of all forms provided by the system by default in a well-annotated format. Any new form created by the user gets added to the list. The modified form gets stored as a new form and there is a record of which form it originated from. This table is used to pull out all the forms existing in the current state of the database for the user to choose from to fill data or to select data elements to aggregate data as explained before. The table 'Form Field' stores information about the different fields present in the form along with the forms that they are a part of. It also stores the label of the form fields and a detailed explanation of the field which would translate into a tool tip description in the UI to assist the user in filling the data. The table 'Form Field Type' is created where values such as textbox, radio-button, check-box, drop-down,etc are stored. The table 'Form Modifications' is needed to keep a record of modifications made to the existing form. If a form field is added or deleted by a user to create his customized form then this table will keep track of the changes and pull out the customized form fields for users. The 'Form

Field Option' table stores the enumerations of values for the aforementioned types of form fields. For instance, the form field 'Gender' has options 'Male' and 'Female' which are stored in the 'Form Field Option' table.

The table 'Table Information' stores the tables, which are present in the actual database holding the data, which in our case is the ALIF DATABASE. This table is required for the purpose of aggregating data using the view creation mode where users have an option of aggregating data elements, which they want to view together. The table 'Attribute Information' stores all the attributes present along with their data-types and default values in the actual database holding the data. We also keep track of attributes which are primary keys or foreign keys in a particular table. This enables the system to decide the table joins when user selects data elements to be aggregated in the view creation mode explained in the following sections. If a user selects some data elements from a set of tables, which cannot be joined due to the absence of Primary Key-Foreign Key relationship then the user is prompted against the action

This FORMS DATABASE is populated by acquiring the definition of the underlying DSD, say as an SQL file, with CREATE TABLE statements. The level of automation is being improved by providing this functionality. The system currently has the ability to filter out attributes which need not be displayed on the forms. Our strategy for dealing with schema evolution as the database creation progresses is explained below. The algorithm can be used to modify an existing database at the back-end with the help of a dialog based UI or create a new database from scratch. When modifying an existing database by adding additional forms we already have the metadata database and the DSD at the back-end contrary to the case when a new database is created. In the latter case, an unpopulated metadata database exists at the back-end. As and when the user creates forms at the front-end, the DSD gets

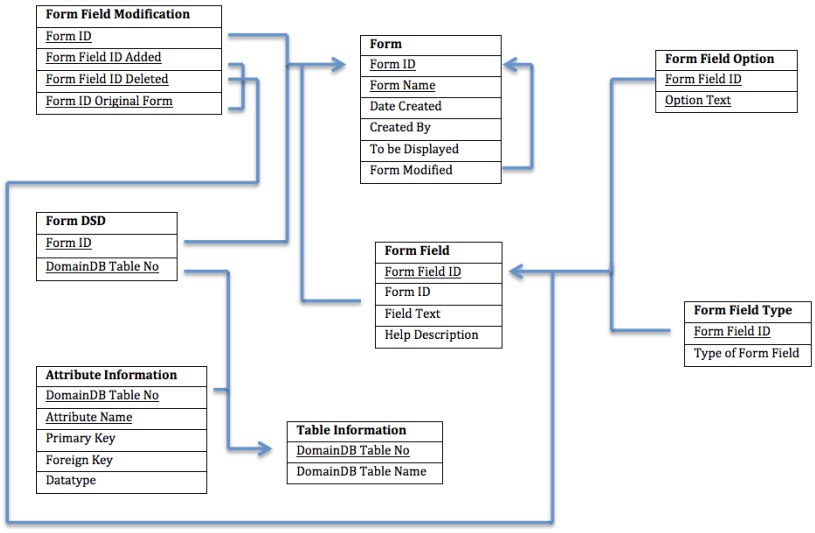

Figure 8: Schema of the metadata database. The arrows show referential integrity constraints

developed. We explain our approach for the 2 cases below:

Case 1: Database Schema Modification of the DSD

This is the case when a user chooses to modify existing forms to customize based on his requirements in turn leading to modification of the DSD schema. The modified form would be treated as a new form with a new 'Form ID'. The creation of this new form is recorded in the 'Form' table. In this table the 'Form ID' of the original form is stored, which was modified to create the new form. The modifications would be stored separately in another table 'Form Field Modification' where we store the form fields that were added or deleted from a particular form to create a new form. Any form field which gets added to a form first needs to be added to the 'Form Field' table, 'Form Field Type' table and the 'Form Field Option' table appropriately. For every new form field which gets added to a form, a corresponding attribute gets added to the existing table which has other attributes corresponding to the other form fields of the form. This requires population of the 'Attribute Information' table with all the

information about the new attributes. The 'data entry' feature is used to populate the table with data. If there is data already existing in the table corresponding to the form being modified then default values of the newly added attributes would be inserted for these existing observations.

Case 2: Database Schema Creation.

This is the case when a user does not want to use any of the existing templates and instead creates a new set of forms. This may occur if the data that the user wants to store pertains to a different domain. He would be required to create new forms from scratch which would guide the creation of a new domain specific database. At the metadata level this involves populating the 'Form' table with information about the new forms. The 'form modified' field would be 'NULL' since we are not modifying any existing forms to create the new forms. In this case we would also populate the 'Table Information' table with information about the new tables that would be formed in the new DSD corresponding to the new forms created. Subsequently the 'Form DSD' table and the 'Attribute Information' table would also be populated as and when new form fields are added in the forms. The process of creation of new form fields has been explained in CASE 1.

In both cases, when adding a form field to a form if the form field is supposed to have atomic values then the corresponding table in the DSD is updated but if the form field is expected to have multiple values then a new table is created in the DSD which references the original table corresponding to that form.

3.3.2 Guranteeing consistencies during schema evolution

In our approach a lot of flexibility has been provided to the user in terms of freedom of choice of creating new forms when the existing form templates do not seem to be suitable. This can lead to a lot of redundancy at the back-end since a user may

choose to build a new form instead of modifying existing templates even though his requirements differ from the existing forms by a small amount. For example, a user needs only 8 out of 10 form fields of a particular form and wants to add more fields of his own choice. We would assume that the user would use our feature of modifying this form by deleting the two irrelevant fields and adding the new extra fields. But instead it is possible that he creates a new form and adds all the form fields he needs to this new form. At the back-end this would result in an extra table in the DSD which would store data being entered via this new form. Periodic reorganization of the DSD and the metadata database is required. This would involve manually identifying such redundant tables and integrating them into one table by performing a full outer join between them. This would be done when the primary keys of the two tables which are getting integrated are the same. The two keys may have different labels but if they have the same semantics, we would go ahead with the join. Such merging would be done only with human approval. If the keys are different then we would keep the tables as they are. A full outer join may result in a lot of null values in the integrated tables. Due to the increase in the volume of data and creation of multiple redundant tables, the decision to go ahead with the integration would depend on the relative advantage of querying a single table with a lot of null values over querying multiple tables to get the data.

3.4 Data Retrieval: Query Creation and Validation

The system also supports data retrieval by giving the user a choice of either selecting existing result sets previously created by users or creating his own. The user is presented with four types of operations namely 'Select', 'Aggregation', 'Group By' and 'Having' to build a query.

1. **Select Operation**

   The user can select fields, which need to be displayed together in the result set.

This may be done by selecting a table from a drop down list and on doing so the corresponding attributes of the selected table would be shown.

2. **Aggregation Operation**

   The operation may be used by user to perform aggregate functions like COUNT, MIN, MAX, AVG, etc on the fields.

3. **Group By and Having Operation**

   If the user decides to use the Aggregation operation then the 'Group By' and 'Having' operations would be enabled. Using them the aggregated results may be grouped with respect to certain fields which may or may not be based on some condition.

The system maintains consistency of the user request as well as of the back-end operations as follows. If the selected attributes belong to multiple tables then the system would perform a join between those tables using the Primary Key and Foreign Key constraints. A 'Where' clause is appended to the query being created, to reflect the join. If there are any attributes that the user has selected to group his result by then those attributes would be added to the attribute selection list already created by the 'Select' Operation in the first step. If the attributes that the user has selected require a join of tables, which cannot be joined due to absence of a primary key-foreign key relationship, then the user would be prompted to change the selection. If the user has selected some attributes and is also performing aggregation then the system would remove the selected attributes other than the ones he is grouping by from the SELECT clause. The user would be prompted of this action. The system would also check for validity of attributes selected for aggregation. E.g. Calculating 'Average' of a non numeric attribute is prevented. After validation the query is executed and the result is displayed. The user has the choice of saving the result set with a description of the same. At the back-end the query is stored in the DSD along

with the description provided and can be executed again when selected by its query label.

3.5 Related Work

Some research has been done in the area of dynamic schema modification with respect to a clinical dental relational database [42]. Their approach primarily focuses on handling One-to-Many and Many-to-Many relationships between tables via concepts of 'detail' and 'link' tables. The user interface developed by the authors is limited to the dental domain. In our approach the metadata database remains unchanged and there exists a provision for plugging in any DSD which in turn can be modified by users. Our application can also be used to create a new database from scratch and store data while it is created in real time. In addition to this, in [42] the interface to manage the addition of datasets requires the user to be familiar with the dataset being loaded after which it is the responsibility of the user to map it to an existing domain or create a new domain. In our approach the user is oblivious about the back-end structure. While he creates new form fields or a whole new form, our application automatically begins to modify the existing schema by creating new attributes or relations respectively to accommodate the new data elements. In addition to the dynamic schema modification approach, a feature of aggregating data and presenting the results to the user, which he can store for future use is also supported by our system. Palisser et al [1] discuss drawbacks of the systems called Orion [16] and Encore [40]. The former constructs a version of the database state every time any transformation in the schema takes place. This leads to the problem of managing multiple versions. The latter focuses on versioning of object types when design environment object types change in an object oriented database. In our approach on the other hand the original schema is modified based on the changes requested by the user but the user is kept unaware of these changes. For instance, a user might

modify an existing form to create a new customized version but at the back-end the original schema accommodates the new data elements in an appropriate manner to avoid resorting to versioning. Kim et al. [19] have handled versioning of object types as well as schemas for single as well as multi-user design environment in Orion and also provided semantics of versioning the schema. Ferran et al. [10] discuss an approach to handle schema and database evolution in O2 object database system. The algorithm proposed by the authors automatically makes the database consistent on any update operation performed on the schema. However, depending on what the updates are, either immediate or deferred transformations are made. Deferred transformations cause problems while implementing complex conversion functions,that the user needs to specify if the default functions do not suit their needs. Our approach on the other hand does real time modification of the original schema based on the changes requested by the user and avoids user intervention to a large extent.

3.6 Use Case

Let us describe the current interface for this prototype used by ALIF practitioners at a local neurosurgery practice.

1. **Basic menu for user [Figure 9]:** This figure shows the menu which will be presented to the user after logging in. It consists of all the template forms consisting of data elements currently in the back-end database. The last bullet is "Create New Form" which would be explained ahead. It has a 'Form Field Selection' panel which the user can use when creating new form fields when needed. The main menu also consists of a 'Choose Patient' option which will help pulling up existing patients in the database when additional data needs to be entered about them.

2. **Ability to use existing forms :** On choosing a particular template the corresponding form is displayed for the user to fill in data. This data gets saved in

the appropriate tables at the back-end.

3. **Creating / Updating forms :** If the user feels the need to update existing templates to accommodate data elements, which are not in the form, he can modify the form using the 'Add New Form Field' feature. This creates a new version of the existing form incorporating the changes requested by the user. The new form fields added get saved in the back-end database at appropriate places. The "Create New Form" feature can be selected from the screen shown in Fig. 4 to create a new form from scratch and add new form fields.

4. **View Creation [Figure 10]:** The data retrieval feature allows users to select the fields they wish to integrate together, perform aggregation operations on them and also group the result set fields of their choice. The result set can be saved for future use. All screens could not be shown for space reasons.

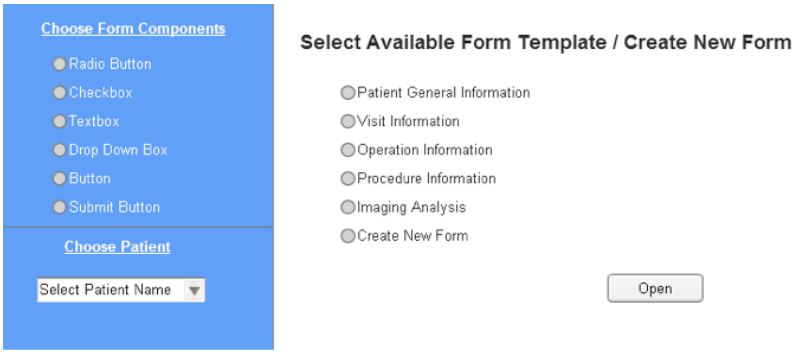

Figure 9: Basic Menu

3.7 Discussion and Conclusion

3.7.1 Why not NoSQL?

A lot of organizations which collect vast amounts of customer, scientific, sales data have traditionally stored data in a relational structure, but recently some of these organizations are tending to use various types of non-relational databases called NoSQL databases since they have been found to be efficient in handling unstructured data

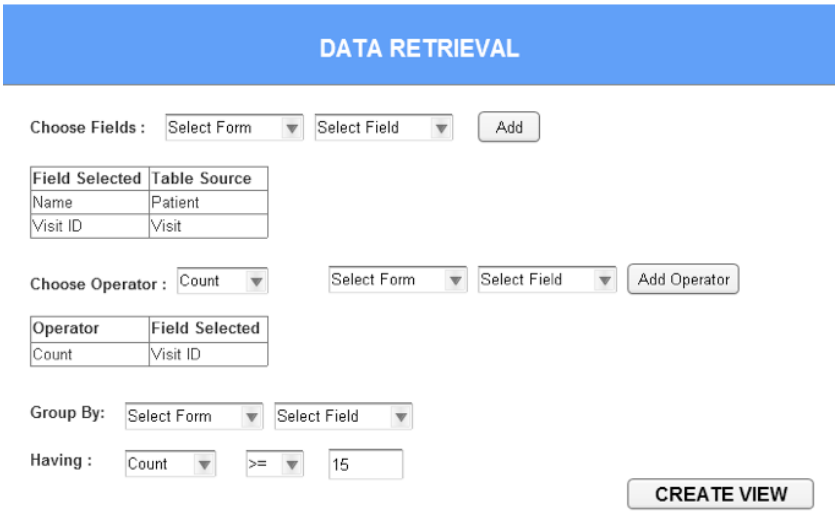

Figure 10: View Creation Screen

where there is no fixed schema [**?**]. In our case we also have a schema that is flexible since the user can modify it based on his requirement, but we chose not to go for NoSQL due to the following reasons.

1. Consistency, availability, and partition tolerance are the three properties taken into consideration when designing a database system. According to the CAP theorem [**?**] it is only possible to have two out of these three properties together in a database system at once. Traditional relational databases maintain consistency and availability but have trouble with partitions whereas NoSQL databases are able to maintain either consistency or availability along with partition tolerance. In our case the main goal of the system is providing the user with all the data in a consistent state and available in user's expected form.

2. The primary goal of our system is to add flexibility to existing schemas and dynamically modify them. Since most of the healthcare databases have a relational structure at the back-end, we formulated our approach using relational databases.

3.7.2 Metadatabase flexibility

The metadata database in our system plays an important role in driving the dynamic nature of the DSD which stores tha actual content entered using the forms. The schema of the metadata database is designed in such a way that the DSD can be modified without any manual interference. It guides the UI formation and the selection process of the different UI components other than providing the forms with a particular structure under different circumstances. The metadata also has the capability to form queries based on information entered by the users along with verifying the correctness of those queries.

3.7.3 Conclusion

In this work we proposed a UI and metadata based approach to dynamically modify and create a database schema which would address the problem of dynamic data entry in data-rich environments where the schema can be "built" incrementally as new data becomes available. It is intended for users with needs for managing data for operational and research purposes but who have no access to DBAs or database design experts. Template forms are provided to the user but since there may be difference in requirements between users, the system facilitates modification of existing forms or create new forms from scratch resulting in appropriate real time modifications in the database schema keeping the user oblivious to the back-end The changes at the back-end involve adding new attributes to existing tables, adding new tables, creating relationships between these new tables and existing tables by assigning appropriate cardinality, etc. Unlike other systems such as Encore and Orion, our approach does not create versions of the schema whenever there is a change; instead it creates different versions of a single form if needed after modifying the schema appropriately. Our system also has a data retrieval feature which helps users to aggregate and extract

data from the database, perform aggregate operations on them and storing the result sets for future use. This feature does not require the user to write queries instead it automatically generates and validates queries based on the data elements selected.

There is still a lot of scope for improvement in this system like making the system usable for novice users who do not have any knowledge of database modeling concepts. Given a good domain ontology about synonyms such as WordNet [**?**] we would like to automate the process of removing redundant tables. There is a possibility that a user modifies a form to add a form field, which has semantic equivalence with one of the existing fields in the form. This would result in redundancy. To avoid such cases we would like to incorporate semantic validation in the future. The paper represents preliminary work awaiting field experimentation with small group practices of physicians. It addresses the problems related to relational schema evolution in real-time which opens up a lot of room for improvement. We have been motivated for the need of such system in data-rich environments with relatively limited volume such as medical practices. We are addressing a large user population where the typical user is not very knowledgeable about database concepts but would like to be able to create robust databases on the fly. We would like to like to address the aforementioned ideas of improvement before making it available to the medical community.

CHAPTER IV

TREATMENT ADVISOR TOOL

The study described in Chapter II forms the basis for developing a treatment recommendation tool for cancer patients. We have developed a prototype for the clinicians to enter relevant information about a newly diagnosed Glioblastoma patient which would be fed into the predictive analytics pipeline to predict his time period of survival. The tool also uses the concept of patient similarity to mine patients that are clinically and genomically similar to the patient under consideration. The tool would generate multiple treatment plans for the patient based on how the similar patients were treated and the significant treatment patterns mined in the study of GBM patients.

4.1 Related Work

Patient similarity assessment is one of the foundations of every clinical decsion support system. It involves finding clinically similar patients with the goal of treating them in a similar fashion. The prime objective is to derive a similarity measure between a pair of patients based on their EHR data .Sun et. al [41] present a locally supervised metric learning approach to learn a generalized Mahalanobis distance which is tailored toward physician feedback. An interactive metric learning method has been developed which incrementally updates an existing metric based on physician feedback. Defining a good distance metric in feature space is crucial in assessing patient similarity. Mak et. al [22] use support vector machines derived from statistical learning theory to assess patient similarity. A similarity measure is used to generate a similarity score for different categories features of the disease such as tumor size, degree of liver damage, etc in the case of liver cancer. The scores quantify the similarity between

two patients with respect to the corresponding categories. SVM can be used to combine these similarity measures with statistical learning and then form a model for classifying the patient pairs into "similar" or "dissimilar" class.

4.2 Architechture and Design

The tool consists of three components: the predictive analytics pipeline, patient similarity and a recommender. Figure 11 shows the arhitecture of the tool. The predictive analytics pipeline consists of the predivtive model described in chapter II which predicts whether a patient is likely to survive for longer than a year. A relational database is used to store the data of for all the patients and their treatment plans in the form of a graph. The sequential mining sub module which is a part of the pipeline mines the significant treatment plans and uses them as features in addition to the clinical and genomic featues to perform the prediction.

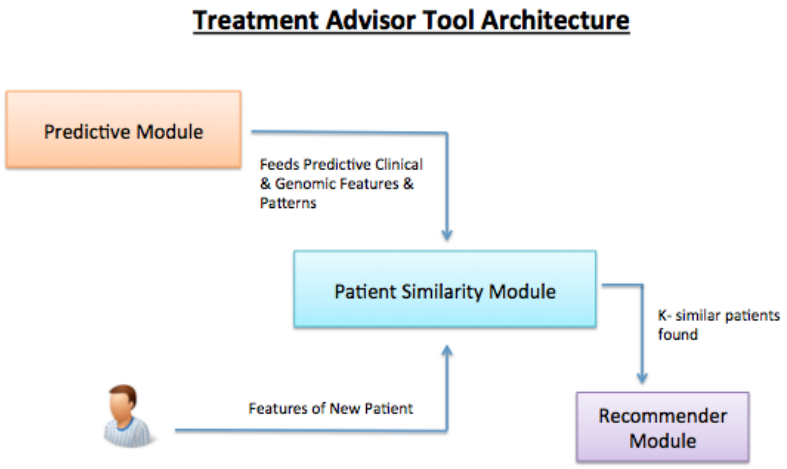

Figure 11: Treatment Advisor Tool Architecture

The 'Patient Similarity' component is used to find patients from the database who share a similar profile with the newly diagnosed patient. The similarity assessment is an important task in healthcare management and in the context of comparative

effectiveness studies, identifying historical records of patients who are similar to the new patient could help retrieve similar reference cases to formulate better treatment plans [41]. The similarity is calculated based on only the predictive features which were discovered by feature selection algorithm in the predrictive analytics pipeline. Each feature is assigned a weight by the clinicians based on the relevance of the feature in measuring similarity between patients. The nature of the data was of a mixed type consisting of numeric and categorical data types. Some of the categorical data elements were binary and consisted of 1s and 0s representing presence and absence of that data element respectively. To remove bias amongst the data elements, we used the concept of normalization of data. The normalization of a data point was achieved using the equation below:

$$x\_norm = (x\_max - x)/(x\_max - x\_min) \tag{4}$$

x_norm = normalized value of 'x'

x_max = maximum value of 'x' in the dataset

x_min = minimum value of 'x' in the dataset

The similarity between patients is calculated by the following equation.

$$S = \sum_{i=1}^{m} \psi[wx_i, Dx_i] \tag{5}$$

S = Similarity Score between the newly diagnosed patient and the N^{th} patient in the database

m = number of predictive features discovered by the predictive model.

wx_i = weight of the i^{th} feature

Dx_i = Distance between the newly diagnosed patient and the N^{th} patient in the database.

The recommender component uses the treatment plans prescribed to the similar patients found by the patient simiarity component and combines the significant treatment plans found by the pipeline to recommend a personalized treatment plan for the newly diagnosed patient.

The current implementation of the tool consists of a user interface customized for Glioblastoma. Fig 12 shows the form which requires clinicians to enter information about the newly diagnosed patients. Some of the fields include age of the patient, mRNA expresssion levels of specific genes, the methylation status of the MGMT gene, etc which were discovered by the forward feature selection algorithm as part of the predictive model described before. In addition to the information about the patient, the clinicians can choose to select the number of clinically and genomically similar patients that need to be analyzed for treatment reference. The range provided by the tool is from 1 to 10. The data collected by the clinicians is used to calculate the similarity score with all the patients in the database which have a survival period longer than a year. The current UI also has the capability to record treatment plans for patients who already have been undergoing treatment for the tool to recommend the next best treatment option, but we have not implented this feature yet.

The treatment recommendations are based on the treatments given to the most similar patients found by the tool and the significant treatment plans which have been discovered to play an important role in prolonging survival. The screen shown in Fig 13, shows a ranked list of treatment options for a newly diagnosed patient based on 3 most similar patients. The clinician can choose any of the treatment plans from the ranked list for the patient under consideration by clicking on the checkbox provided. On choosing a plan, the tool compares the chosen plan with the significant treatment plans extracted by the forward feature selection algorithm to provide any additional recommendations. The additional suggestions are shown at positions in the sequence where there is a difference between the events in the chosen plan and

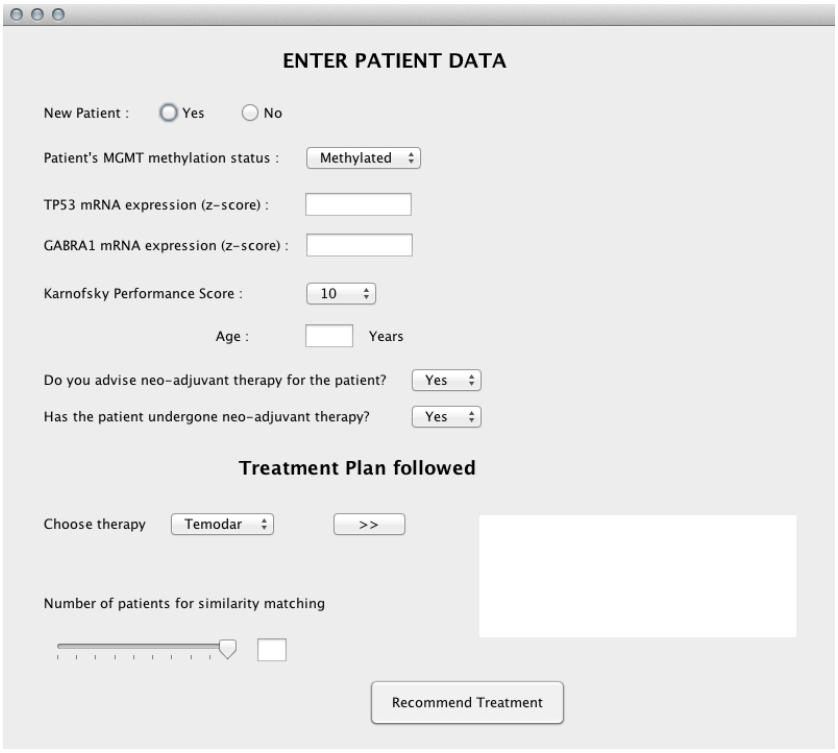

Figure 12: Patient Record Form

the significant plan. E.g In the example shown in fig 13, the chosen plan consists of a prescription of Temodar followed by radiation therapy which in turn is followed by a prescription of CCNU. The additional suggestion section recommends prescribing a signal transduction inhibitor as the third event instead of the CCNU drug since prescription of signal transduction inhibitor as the third event in the treatment was mined as a significant event by the predictive model. The clinician may or may not choose the suggestions from this section.

4.3 Application of Treatment Advisor Tool

The current implementation of the treatment advisor tool is customized to recommend treatments for Glioblastoma patients. The tool can be easily changed and customized for another disease using the DDSCMS described in the previous chapter. One of the important applications of the dynamic schema modification approach is to incorporate it with the treatment advisor tool to change recommendations in real

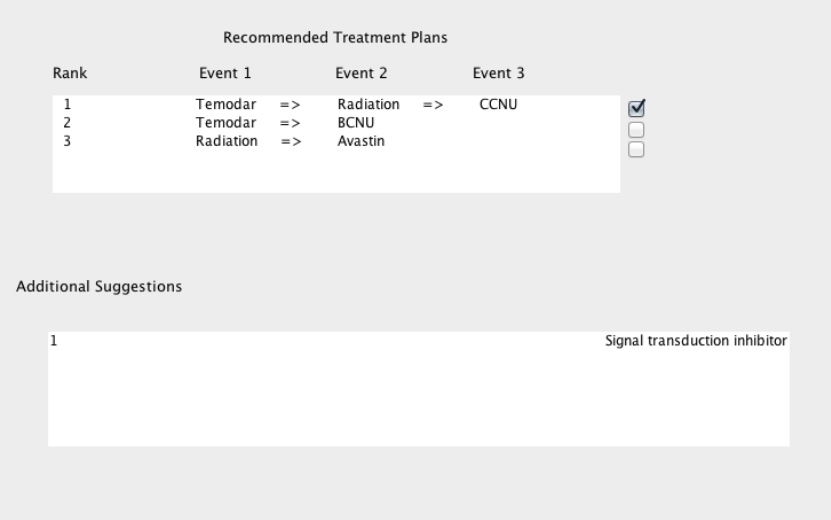

Figure 13: Recommendations

time. We propose an architecture shown in figure 14 consisting of the following:

1. The Treatment advisor tool
2. A database of medical history of patients
3. A database monitoring every patient's data

4.3.1 Real Time Recommendations

There are many diseases, the treatment for which requires the patient to monitor for levels of certain proteins and minerals, dietary intake and excercise routines over a period of time. One such disease is diabetes mellitus type 2 where self monitoring blood glucose (SMGB) is an important component of mordern therapy. It is recommended by healthcare professionals with the goal of achieving a specific level of glycemic control and prevent hypoglycemia [5]. SMBG is used to collect detailed information about blood glucose levels by more precise regimens. It is extremely important for adjustment of a therapeutic regimen in response to blood glucose values and to assist patients in adjust their dietary intake, physical activity and insulin doeses to improve glycemic control on a day to day basis.

We believe that the treatment advisor tool can be used to make real time recommendations based on data actively recorded by patients. The treatment advisor tool containing predictive analytics pipeline would directly work on the data stored in the medical history database and generate an initial set of recommendations for treating a patient. Based on the recommendations and the severity of the patient's condition, the clinician can use the DDSCMS to create a form for the patient and include the data elements which he requires the patient to monitor. This form can be customized for another patient if certain data elements are not needed or if additional ones are required. The data periodically recorded by patients is stored in a relational database which we call the monitoring database which has a schema corresponding to the nature of the fields in the form. The treatment advisor tool would be connected to the monitoring database and would get triggered if the data for one or more data elements being recorded by the patient fall in the danger levels. In such a case the the advisor tool would need to make changes in the recommendations accordingly.

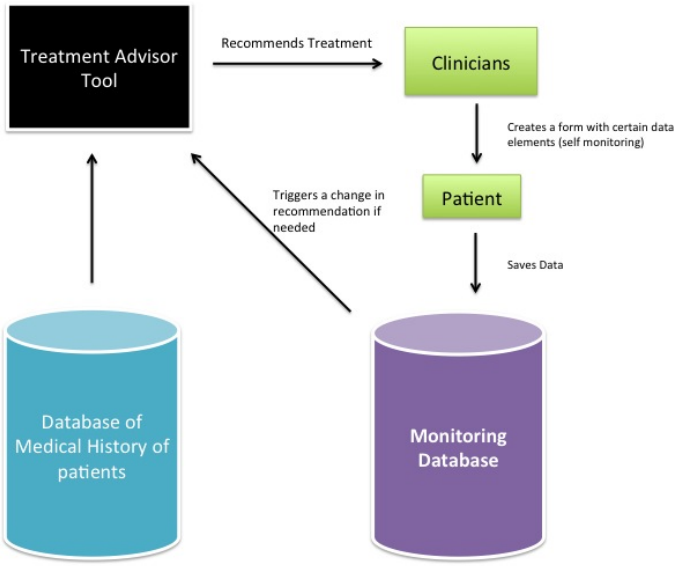

Figure 14: Dynamic Database Modification incorporated with Treatment Recommendation

4.3.2 Use Case

Let us describe a use case for this proposed prototype to be used by diabetes specialists.

1. **First line of recommendation for a newly diagnosed diabetes patient:** The treatment advisor tool would intially find patients clinically and genomically similar to the newly diagnosed patient and display a ranked list of treatment options for the clinician to choose from.

2. **Form creation for self monitoring:** The clinician can use an existing form or create a new form to add the data elements which need to be recorded by the patient. The data elements can include blood glucose levels, diet and excercise information. Appropriate modifications would automatically take place at the back-end.

3. **Triggering the Treatment Recommendation Tool:** The data recorded by the patients using the form provided by the clinician is saved in monitoring database which is connected to the treatment advisor tool. If at any point of time the data recorded by the patient falls out of the normal range defined by the clinician, it would trigger the advisor tool to recommend appropriate changes in the treatment accordingly.

4. **New Recommendations:** New recommendations would be requested when data being recorded for one or more data elements falls out of the normal range set by clinicians. Once the treatment advisor tool is triggered to make new recommendations, the clinicians would need to use the DDSCMS to make changes in the form shown in figure 12 and add these new data elements to find new clinically similar patients.For example if a patient's Hemoglobin A1c level drops below 5, the clinicians would add a field called 'Latest Hemoglobin A1c Level' shown in red in figure 15 and get new list of recommended treatments.

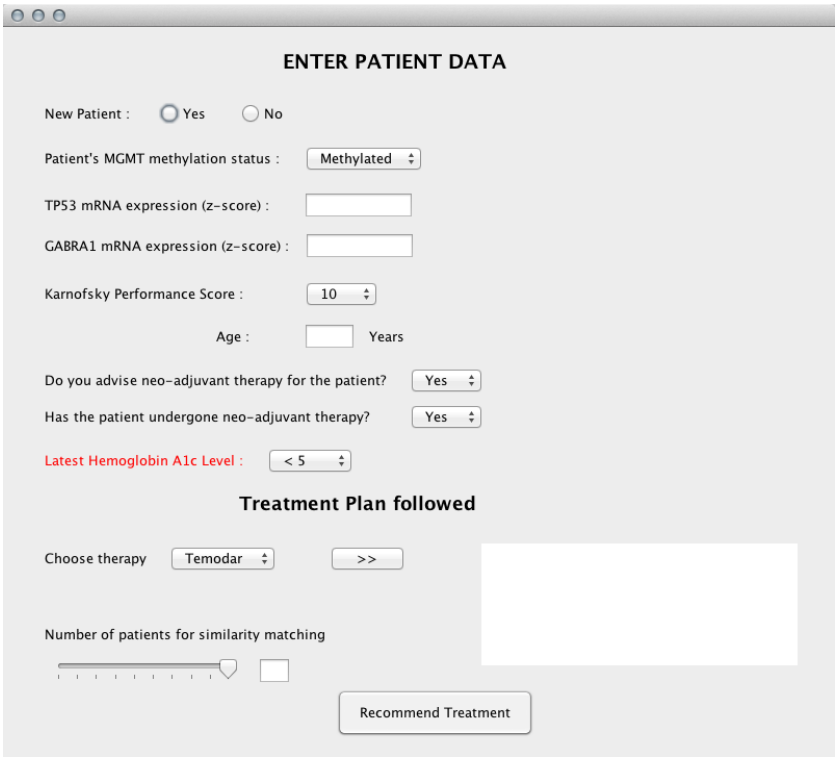

Figure 15: Modified Patient Record Form

CHAPTER V

CONCLUSION AND FUTURE WORK

In this thesis, we discuss a predictive analytics pipeline performing data standardization, mining sequential treatment patterns, and constructing features of predicting GBM patients surviving for longer periods (greater than 12 months). Sequential mining is applied to treatments consisting of drugs and radiation, which are termed as events. In the sequential mining module, we combine GSP and SPADE algorithms and tailor the approach to account for the exact-order and the overlap constraints. Other than clinical and genomic features, treatment patterns extracted from the treatment plans of patients within one year of diagnosis were used as features to predict patients surviving longer than a year using logistic regression as the classifier and forward feature selection for selection of predictive features. This study is a preliminary step in providing extensive treatment guidance to oncologists and neurosurgeons about the efficacy of certain sequence of drugs and therapies as part of a treatment plan. Currently, the treatment patterns consist of the drug names and their event of prescription. We are developing a treatment advisor tool to recommend treatments for a patient based on treatments given to patients having a similar clinical and genomic profile using the knowledge of treatment patterns obtained from this study. We also plan to add more constraints in the model such as a 'gap' constraint, which would limit the temporal gap between events for inclusion in a sequence. We believe this would help in filtering out clinically insignificant treatment patterns.

We have also proposed a UI and metadata based approach to dynamically modify and create a database schema which would address the problem of dynamic data entry in data-rich environments where the schema can be "built" incrementally as

new data becomes available. It is intended for users with needs for managing data for operational and research purposes but who have no access to DBAs or database design experts. Template forms are provided to the user but since there may be difference in requirements between users, the system facilitates modification of existing forms or create new forms from scratch resulting in appropriate real time modifications in the database schema keeping the user oblivious to the back-end. In the healthcare domain this tool can be used by the clinicians to create template forms for the patients to record their diet, exercise schedule and blood sugar in case of diabetes. For a different disease the template could be modified or a new one could be created. The changes at the back-end involve adding new attributes to existing tables, adding new tables, creating relationships between these new tables and existing tables by assigning appropriate cardinality, etc. Unlike other systems such as Encore and Orion, our approach does not create versions of the schema whenever there is a change; instead it creates different versions of a single form if needed after modifying the schema appropriately. Our system also has a data retrieval feature which helps users to aggregate and extract data from the database, perform aggregate operations on them and storing the result sets for future use. This feature does not require the user to write queries instead it automatically generates and validates queries based on the data elements selected.

There is still a lot of scope for improvement in this system like making the system usable for novice users who do not have any knowledge of database modeling concepts. Given a good domain ontology about synonyms such as WordNet [**?**] we would like to automate the process of removing redundant tables. There is a possibility that a user modifies a form to add a form field, which has semantic equivalence with one of the existing fields in the form. This would result in redundancy. To avoid such cases we would like to incorporate semantic validation in the future. This work

awaits field experimentation with small group practices of physicians. It addresses the problems related to relational schema evolution in real-time, which opens up a lot of room for improvement. We have been motivated for the need of such system in data-rich environments with relatively limited volume such as medical practices. We are addressing a large user population where the typical user is not very knowledgeable about database concepts but would like to be able to create robust databases on the fly. We would like to like to address the aforementioned ideas of improvement before making it available to the medical community. We combine the analytics pipeline model developed for Glioblastoma with the dynamic database schema modification system to develop an infrastructure for a treatment advisor tool, which in addition to the aforementioned modules uses a patient similarity module to recommend treatment options to a newly diagnosed patient by searching for patients with similar attributes.

REFERENCES

[1] Andany.J., Leonard.M., Palisser.C.: Management of Schema Evolution, VLDB, Barcelona, Spain, 1991.

[2] Agrawal R, Srikant R. "Mining Sequential Patterns". ICDE 1995: 3-14

[3] Ayres, J., Flannick, J., Gehrke, J., Yiu, T. 2002. 'Sequential pattern mining using a bitmap representation' . In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, 429-435.

[4] Beroukhim, R., Getz, G., Nghiemphu, L., Barretina, J., Hsueh, T., Linhart, D.,Vivanco, I., Lee, J.C., Huang, J.H., Alexander, S., et al. (2007). Assessing the significance of chromosomal aberrations in cancer: methodology and application to glioma. Proc. Natl. Acad. Sci. USA 104, 20007-20012.

[5] Benjamin, E. M. 'Self-Monitoring Of Blood Glucose: The Basics'. Clinical Diabetes 20.1 (2002): 45-47. Web. 10 Apr. 2015.

[6] Cerami et al. The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data. Cancer Discovery. May 2012 2; 401

[7] 'Data Mining from A to Z: Better Insights, New Opportunities', SAS, 2013.

[8] Elmasri.R., Navathe.S.(2011).Fundamentals of Database Systems (6th Edition), Addison Wesley.

[9] Freije, W.A., Castro-Vargas, F.E., Fang, Z., Horvath, S., Cloughesy, T., Liau, L.M., Mischel, P.S., and Nelson, S.F. (2004). Gene expression profiling of gliomas strongly predicts survival. Cancer Res. 64, 6503-6510.

[10] Ferrandina. F., Ferran.G.: Schema and Database Evolution in the O2 Object Database System, VLDB, Zurich, Switzerland, 1995.

[11] Gao et. Al (2013). 'Integrative Analysis of Complex Cancer Genomics and Clinical Profiles Using the cBioPortal'. Sci. Signal, Vol 6, Issue 269, p.pl1

[12] Glioblastoma and Malignant Astrocytoma: American Brain Tumor Association; 2012. Available at: http://www.abta.org/secure/glioblastoma-brochure.pdf

[13] Holland, E. (2000). "Glioblastoma Multiforme: The terminator." PNAS 97(12): 6242-6244.

[14] Hegi et. al, 'MGMT Gene Silencing and Benefit from Temozolomide in Glioblastoma',N Engl J Med 2005; 352:997-1003

[15] Kotsiantis.S.B, 'Supervised Machine Learning: A Review of Classification Techniques', Informatica (Slovenia) 31(3): 249-268 (2007)

[16] Kim.W , Ballou.N, Chou. H.T., Garza.J.F. , Woelk.D.: Features of the ORION Object-Oriented Database System; in Object-Oriented Concepts, Databases and Applications, (ed W.Kim and F.M.Lochovsky), ACM Press Frontier Series, New York, 1989.

[17] Kamber.M., Han.J.(2006). Data Mining: Concepts and Techniques (2nd Edition), Elsevier.

[18] Kolp.M., Zimanyi.E.: Enhanced ER to relational mapping and interrelational normalization, Informaton and Software Technology 42 (2000) 1057-1073.

[19] Kim W., Chou H.: Versions of schema for object oriented databases,VLDB, Los Angeles, 1988.

[20] Krex et al.'Long-term survival with glioblastoma multiforme'.Brain (2007) 130 (10): 2596-2606. doi: 10.1093/brain/awm204

[21] Liang, Y., Diehn, M., Watson, N., Bollen, A.W., Aldape, K.D., Nicholas, M.K., Lamborn, K.R., Berger, M.S., Botstein, D., Brown, P.O., and Israel, M.A. (2005). Gene expression profiling reveals molecularly and clinically distinct subtypes of glioblastoma multiforme. Proc. Natl. Acad. Sci. USA 102, 5814- 5819.

[22] Mak et al. 'Machine Learning of Patient Similarity: A Case Study on Predicting Survival in Cancer Patient after Locoregional Chemotherapy', 2010 IEEE International Conference on Bioinformatics and Biomedicine Workshops(BIBMW), 467-470

[23] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) Foundations of Machine Learning, The MIT Press ISBN 9780262018258.

[24] Mazurowski M.A, Desjardins A, Malof J.M. 'Imaging descriptors improve the predictive power of survival models for glioblastoma patients', Neuro-Oncology, Vol.15, Issue 10, Pp. 1389-1394

[25] Mischel, P.S., Shai, R., Shi, T., Horvath, S., Lu, K.V., Choe, G., Seligson, D., Kremen, T.J., Palotie, A., Liau, L.M., et al. (2003). Identification of molecular subtypes of glioblastoma by gene expression profiling. Oncogene 22, 2361- 2373.

[26] Murat, A., Migliavacca, E., Gorlia, T., Lambiv, W.L., Shay, T., Hamou, M.F., de Tribolet, N., Regli, L., Wick, W., Kouwenhoven, M.C., et al. (2008). Stem cell related "self-renewal" signature and high epidermal growth factor receptor expression associated with resistance to concomitant chemoradiotherapy in glioblastoma. J. Clin. Oncol. 26, 3015-3024.

[27] Network, T. R. (2010). The Cancer Genome Atlas Data Portal, National Institute of Health.

[28] Nice.org.uk, (2001). 'Guidance on the use of temozolomide for the treatment of recurrent malignant glioma (brain cancer)' — appendix-d-karnofsky-performance-score — Guidance and guidelines — NICE.[online] Available at: www.nice.org.uk/guidance/ta23/chapter/appendix-d-karnofsky performance-score [Accessed 9 Nov 2014]

[29] Nutt, C.L., Mani, D.R., Betensky, R.A., Tamayo, P., Cairncross, J.G., Ladd, C., Pohl, U., Hartmann, C., McLaughlin, M.E., Batchelor, T.T., et al. (2003). Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Cancer Res. 63, 1602-1607.

[30] Palisser.C, Leonard.M, Andany.J, 'Management Of Schema Evolution In Databases. VLDB 1991: 161-170.

[31] Pei et al. 'PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth' ICDE, page 215-224, IEEE Computer Society, (2001)

[32] Phillips, H.S., Kharbanda, S., Chen, R., Forrest, W.F., Soriano, R.H., Wu, T.D., Misra, A., Nigro, J.M., Colman, H., Soroceanu, L., et al. (2006). Molecular subclasses of high-grade glioma predict prognosis, delineate a pattern of disease progression, and resemble stages in neurogenesis. Cancer Cell 9, 157-173

[33] Parsons et al. 'An Integrated Genomic Analysis of Human Glioblastoma Multiforme', Science 321(5897): 1807-1812.

[34] http://matematicaspujol.weebly.com/uploads/1/9/1/7/19173025/data_mining_and_predictive_

[35] Rivera. et al, 'MGMT promoter methylation is predictive of response to radiotherapy and prognostic in the the absence of adjuvant alkylating chemotherapy for glioblastoma'. Neuro-oncology, Vol 12, Issue 2, Pp 116-121

[36] Robinson.I , Webber.J, Eifrem.E (2013). Graph Databases. California, O'Reilly Media Inc.

[37] Ruano, Y., Mollejo, M., Ribalta, T., Fiano, C., Camacho, F.I., Gomez, E., de Lope, A.R., Hernandez-Moneo, J.L., Martinez, P., and Melendez, B. (2006). Identification of novel candidate target genes in amplicons of glioblastoma multiforme tumors detected by expression and CGH microarray profiling. Mol. Cancer 5, 39.

[38] Sasso.R.C.,Reilly.T.M (2008).Anterior Lumbar Interbody Fusion: Threded Bone Dowels Versus Titanium Cages. In Resnick, Daniel K.; Haid, Jr., Regis W.; Wang, Jeffrey C.:Surgical management of low back pain (2nd ed.). Rolling Meadows, Illinois: American Association of Neurosurgeons

[39] Shai, R., Shi, T., Kremen, T.J., Horvath, S., Liau, L.M., Cloughesy, T.F., Mischel, P.S., and Nelson, S.F. (2003). Gene expression profiling identifies molecular subtypes of gliomas. Oncogene 22, 4918-4923.

[40] Skarra.A, Zdonik. S.B. ?The Management of Changing Types in an Object-Oriented Database?. OOPSLA 1986: 483-495

[41] Sun. J, Wang. F, Hu. J, Edabollahi. S. 'Supervised patient similarity measure of heterogeneous patient records'. SIGKDD Explorations 14(1) 2012

[42] Taylor.D.,Naguib. R. N. G., Boulton.S.: A Dynamic Clinical Dental Relational Database, IEEE Transactions on Information Technology in Biomedicine, Vol. 8, No. 3, September 2004.

[43] Tso, C.L., Freije, W.A., Day, A., Chen, Z., Merriman, B., Perlina, A., Lee, Y., Dia, E.Q., Yoshimoto, K., Mischel, P.S., et al. (2006). Distinct transcription profiles of primary and secondary glioblastoma subgroups. Cancer Res. 66, 159-167.

[44] Verhaak R.G et al (2010). "Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1." Cancer Cell 17(1): 98-110.

[45] Wen P.Y., Kesari S. 'Malignant gliomas in adults'. N Engl J Med. Jul 31 2008;359(5):492- 507

[46] 'World Cancer Report 2014 (Epub) - WHO - OMS -'. N.p., 2015. Web. 10 Apr. 2015.

[47] 'WHO — Cancer'. N.p., 2015. Web. 10 Apr. 2015.

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

[49] Zaniolo. C, Moon H.J, Curino.C, 'Graceful Database Schema Evolution: the PRISM Workbench' PVLDB 1(1): 761-772 (2008)

[50] Zhou.L., Rundensteiner.E.A.,Shin.K.G: Schema Evolution for Real-Time Object-Oriented Databases, IEEE TKDE, Volume 9, No. 6, November 1997.

[51] Malhotra.K, Medhekar.S, Navathe.S.B, Laborde.D, 'Towards a Form Based Dynamic Database Schema Creation and Modification System'. CAiSE 2014: 595-609

[52] Malhotra.K, Chau.D.H, Sun.J, Hadjipanyis.C, Navathe.S.B 'Temporal Event Sequence Mining for Glioblastoma Survival Prediction'. ACM SIGKDD Workshop on Health Informatics (HI-KDD 2014).

[53] Malhotra.K, Chau.D.H, Sun.J, Hadjipanyis.C, Navathe.S.B, 'Constraint Based Temporal Event Sequence Mining for Glioblastoma Survival Prediction', Journal of the American Medical Informatics Association (In Submission)