Efficient Process Data Warehousing

by

Ying-Feng Hsu

B.A. in Management Information Systems, Washington State University, 2001

M.S. in Information Science Technology, The George Washington University, 2002

Submitted to the Graduate Faculty of

School of Information Science in partial fulfillment

of the requirements for the degree of

PhD in Information Science

University of Pittsburgh

2015

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCE

This dissertation was presented

by

Ying-Feng Hsu

It was defended on

May 13, 2015

and approved by

Vladimir I. Zadorozhny, Associate Professor, School of Information Sciences

Marek Druzdzel, Associate Professor, School of Information Sciences

Hassan Karimi, Professor, School of Information Sciences

Michael B. Spring, Associate Professor, School of Information Sciences

Shyam Visweswaran, Assistant Professor, Department of Biomedical Informatics

Dissertation Advisor: Vladimir I. Zadorozhny, Associate Professor, School of Information Sciences

Efficient Process Data Warehousing

Ying-Feng Hsu, PhD

University of Pittsburgh, 2015

This dissertation presents a data processing architecture for efficient data warehousing from historical data sources. The present work has three primary contributions. The first contribution is the development of a generalized process data warehousing (PDW) architecture that includes multilayer data processing steps to transform raw data streams into useful information that facilitates data-driven decision making. The second contribution is exploring the applicability of the proposed architecture to the case of *sparse process data*. We have tested the proposed approach in a medical monitoring system, which takes physiological data and predicts the clinical setting in which the data is most likely to be seen. We have performed a set of experiments with real clinical data (from Children's Hospital of Pittsburgh) that demonstrate the high utility of the present approach. The third contribution is exploring the applicability of the proposed PDW architecture to the case of *redundant process data*. We have designed and developed a conflict-aware data fusion strategy for the efficient aggregation of historical data. We have elaborated a simulation-based study of the tradeoffs between the data fusion solutions and data accuracy, and have also evaluated the solutions to a large-scale integrated framework (Tycho data) that includes historical data from heterogeneous sources in different subject areas. Finally, we propose and have evaluated a state sequence recovery (SSR) framework, which integrates work from two previous studies, which are both sparse and redundant studies. Our experimental results are based on several algorithms that have been developed and tested in different simulation set-up scenarios under both normal and exponential data distributions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 MOTIVATION

Big data represents an opportunity for analysts and data scientists to discover greater insights and to make more informed decisions that will benefit daily life. To better use the information from big data, efficient data processing and analyzing strategies are needed. Typical large enterprises have approximately one petabyte of operational data stored in over 1,000 data repositories that support over 5,000 applications [1]. Data storage volumes grow in excess of 50% annually. Workloads from online transaction processing (OLTP) over large databases are growing at over 60% per year. Repositories for decision support systems, which often contain replicated data, grow two to three times as fast as databases used for OLTP. This growth is expected to continue due to new Web-based systems, increased accesses to existing systems, and the introduction of new sources of data (such as sensor networks). In order to cope with these predicted growth rates, considerable advances are expected in data management technology, and the quick summarization and utilization of data becomes an urgent task.

As an example, we consider the CHP Bedside Medical Device Interface (BMDI) infrastructure, which provides a set of tools for the automatic acquisition of data from devices at patient's bedside (such as monitors and ventilators). The BMDI infrastructure has 93 mapped parameters for ventilators and 22 for monitors that collect data on 115 parameters (such as heart rate, respiratory rates, systolic blood pressure, arterial mean pressure, SpO_2 monitor-oxygen saturation, and volume gas, among others). In addition, lab data and patient medical treatment records, including medications, interventions, medical protocols, tests, and many others, are also collected and stored in an Oracle database that is replicated on network file servers. As another example, we may consider the Tycho, which is a large integrated epidemiological data set that includes diseases and geography information from approximately 50,000 reports in the United States from approximately 1880 through 2010. Despite the considerable amount of operational

data being collected, there is currently no efficient tool to enable the total and efficient utilization of this data.

In general, this dissertation distinguishes between sparse and redundant process data. In the case of *sparse data*, the process events are non-overlapping and may have gaps in their sequences (see Figure 1 Sparse), while in the case of *redundant data*, process events may overlap and conflict (see Figure 1 Overlapping). Note that historical redundant data does not necessarily imply data inconsistency or conflicts. If the redundant historical reports are accurate, the conflicts only reflect data redundancy that prevents obtaining accurate aggregated query results. For example, Report 1 and Report 2 in Figure 1 are considered to be redundant reports for each other. Both reported sums can be correct; however, it's possible to say that they are in conflict if they can't allow us to obtain accurate aggregated values of P1, P2, and P3.

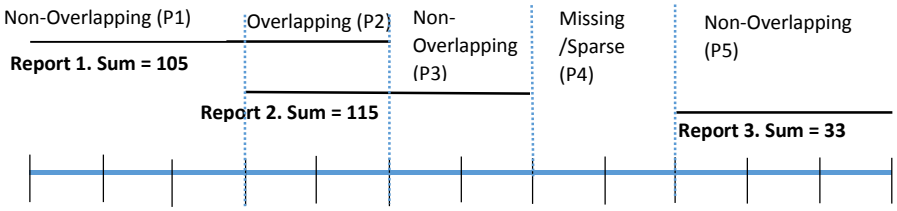


Figure 1. Examples of Sparse and Redundant Data

As an example of *sparse process data*, consider a clinical environment where a patient's vital signs are the first clinical measures taken. These measures, which include temperature, blood pressure, heart rate, respiratory rate, and oxygen saturation, can give a clinician important data on the current status of a patient. These vital signs are often taken by nurses, and the time interval of those measurements could last from an hour to several hours (which can be common during night shifts). Clinicians rely on these measures as indications of patients' clinical status, and many medical problems can cause these physiologic parameters to be out of their normal time ranges. Physicians not only look at current measurements of these physiologic parameters,

but also at recent trends, so they can gain a sense of the improvement or the worsening of a patient's symptoms. Respiratory-related problems represent a significant number of the chief complaints of children coming into the emergency department. A respiratory issue can have an impact on a number of vital signs, including a child's respiratory rate, heart rate, oxygen saturation, and blood pressure. While current systems have been adequate in helping to monitor vital signs and detect individual measurements that are out of range, none of these systems provides a solution to the *sparse process data* issue and that looks across multiple measurements of individual patients' vital signs to detect worsening trends in the overall status of the patient.

As an example of *redundant process data*, we may consider epidemiological data analysis that often relies upon the knowledge of population dynamics, climate changes, the migration of biological species, or drug development, among other factors. This historical data reports on events of interest that occur within various time intervals, and as a result, it may include severe data conflicts that prevent researchers from obtaining the correct answers to queries on an integrated historical database. It is common to have multiple concurrent reports about the same event within overlapping time intervals. For example, there may be hundreds of reports from different authorities about cases of measles in Los Angeles for the year 1900. There may also be multiple reports of historical statistics for overlapping locations. A cumulative report on the total number of measles cases for the state of California may differ considerably from the available reports on the total number of measles cases in California cities. Another challenge is overlapping names: evolving concepts may be reported under different names and categories that co-exist at different time intervals. For example, many 19th century reports on yellow fever actually referred to cases of hepatitis. In 1947, viral hepatitis was classified as hepatitis A and hepatitis B, but that distinction was not immediately reflected in the epidemiological records. Determining the correct number of cases from all of those reports is problematic. Consideration of only non-overlapping reports may result in significant underestimation; but at the same time, by ignoring the overlaps, there is a risk of over-estimating that number. Proper utilization of this redundant historical data would require resolving data conflicts and applying efficient data fusion strategies. To sum up, the proposed approach aims to facilitate the design and development of intelligent data utilization systems, which analyze historical data on a sequence data stream basis, evaluate the importance of sparse events, and estimate the degree of redundancy from

input data sources, classify the severity of conditions reflected in the data, and then alert a user about trends in the data that would indicate a severe status.

1.2 OBJECTIVE

Motivated by the challenges discussed in Section 1.1, we propose a multilayer data processing data warehousing (PDW) architecture to transform historical raw data streams into useful information that can facilitate data-driven decision making. With regard to this PDW framework, we explore its applicability in cases of both sparse process data and redundant process data. Our PDW architecture implements an efficient and scalable technique to discover interesting patterns and anomalous events from various domains. This dissertation considers an event detection approach that accepts historical raw data as an input and outputs domain-specific alerts or results as quickly and accurately as possible. In order to precisely show the achievement of the previously mentioned approach, we break them down into more specific research questions, as follows:

(a) What are the key information processing steps for the proper utilization of process data?
I address this question in Section 3, where I propose an integrated process data warehousing (PDW) architecture. In essence, this PDW architecture includes functionalities in the following three layers that apply to interdisciplinary environments.

(1) Lower Layer (Data preprocessing): continuously and efficiently process large-scale dynamic data that is heterogeneous, multi-dimensional, and complex.

(2) Middle layer (Data transformation): discover an efficient way to change the data granularity, which is designed to reduce the size of big data without losing important knowledge. For example, it would be possible to transfer (classify) the data from lower layers to more meaningful states (groups) to provide new interpretations of the data.

(3) Top Layer (Decision Making): focus on model selection and prediction approaches with sophisticated decision making methods. This layer is designed to explore a series of event severities, and raises an alert when a pattern in event severities analyses matches a set of

pre-defined rules.

(b) How to handle and utilize *sparse process data*?

In order to answer this question, we apply the proposed PDW architecture to clinical data and demonstrate detailed results in Section 5. In general, the PDW approach proposed in this work can help to identify vital sign derangements; in particular, it can help to identify those that are caused by sparse data in ways that current methods of trend detection do not. For example, the PDW approach can help to build a system that would notify clinicians of a patient's worsening symptoms before they would typically have made the decision about whether a patient is in a worsening status and needs ICU-level care. The following use cases illustrate this concept.

(1) Emergency Department Use Case:

A child arrives in the emergency department with a respiratory complaint (wheezing). The patient is placed into a room and a set of vital signs are entered into the electronic medical record. Treatment is begun for the illness at hand. During the course of the patient's emergency department stay, other sets of vital signs are taken. At the time of the third set of vitals, the data is analyzed by the system and a message is sent to the physicians taking care of the patient that this child's vital signs are consistent with those of a child who would typically be admitted to the ICU. The emergency department physicians take further actions which help to improve the patient's clinical status, the patient's vital signs then improve, and the system now predicts that the patient will be admitted to the medical floor.

(2) Medical Floor Use Case:

A patient is admitted to a medical floor where treatment for his or her respiratory complaint continues. As vital signs continue to be taken, the system analyzes the new data. Based on the current data, the system reports that the vital signs predict that the child will need to go to the ICU for further, more aggressive, treatment.

(c) How to handle and utilize *redundant process data*?

This question is addressed in detail in Section 6. In order to answer this question, we apply the proposed PDW architecture to a large-scale epidemiological data set, taking it through the steps of data integration, data curation, and data fusion. This epidemiological data involves considerable redundancy (namely, overlapping reports); as a result, we cannot always have the

accurate values for either individual time interval (such as weekly reports) or time spans (such as monthly reports). In order to use this data, we introduce the concept of optimal conflict degree (CD) to minimize the misestimation of values reported in conflicting historical tuples. In addition, we try to integrate works from two previous studies. The idea is to adopt the concept of state sequence from research question (b) and embed it into the redundant data environment of research question (c). In general, it is difficult to estimate an accurate individual value within a given time interval. Therefore, instead of estimating individual values, our goal is to estimate the states (range of values) for certain time intervals.

1.3 INFORMATION SCIENCE (IS) CONTRIBUTIONS

As mentioned from the objective, this thesis can be used to carry out various types of data processing tasks and can be summarized as three primary information science (IS) contributions. The first IS contribution is the development of a generalized process data warehousing (PDW) architecture that includes multilayer data processing steps to transform raw data streams into useful information that facilitates data-driven decision making. PDW includes information processing steps for the proper utilization of process data and includes various types of tasks, using approaches:

(1) Data stream management methodologies that include data stream models, query processing techniques, and stream query languages.
(2) General data preprocessing techniques, including supervised learning, unsupervised learning, and data warehousing.
(3) Discovering critical patterns from processing events, including methodologies of event detection, anomaly detection, and decision making.

Sparse data problem is one of the most important problems for large scale data analysis and our second contribution is exploring the applicability of the proposed PDW architecture to the case of sparse process data. To handle the sparse process data, we propose a CgroupBy algorithm

to formalize tuple of time and event. We have tested the proposed approach in a medical monitoring system, which takes physiological data and predicts the clinical setting in which the data is most likely to be seen. We have performed a set of experiments with real clinical data (from Children's Hospital of Pittsburgh) that demonstrate the high utility of the present approach.

Our third IS contribution is exploring the applicability of the proposed PDW architecture to the case of *redundant process data*. The redundant data may include severe data conflicts caused by database redundancies, which can prevent researchers from obtaining the correct answers to queries in an integrated historical database. We have designed and developed the conflict-aware data fusion strategy to deal with the redundant data issue in our PDW and demonstrated that our approach significantly reduces errors in data aggregation. We evaluate our solutions to a large-scale integrated framework (Tycho data) that includes historical data from heterogeneous sources in different subject areas. In addition, we also propose a state sequence recovery (SSR) framework, which is an integration approach to handle both sparse and redundant processing data.

1.4 OVERVIEW OF THIS DISSERTATION

The rest of this dissertation is organized as follows: In Section 2, we survey related works in the areas of data stream management, data warehousing, data processing, early warning studies, and redundant and inconsistent data management, as well as in state sequence analysis. In Section 3, we introduce our integrated process data warehousing (PDW) architecture. To validate the advantages and performance of proposed PDW approach, we provide comparison analysis to other data process methods in Section 4. In Section 5, we provide a case study in the medical domain where we explore the applicability of PDW architecture to the case of *sparse process data*. In Section 6, we perform another case study of *redundant process data*; we evaluate solutions for a large-scale integrated framework (Tycho data) that includes historical epidemiological data from heterogeneous sources in different subject areas. In Section 7, we propose and evaluate a state sequence recovery framework that integrates work from the two

previous studies. The summary of the dissertation, on the ways in which the proposed approaches address the research questions, is provided in Section 8.

2 BACKGROUND AND RELATED WORKS

In this work, we investigate issues of efficient data processing from heterogeneous data sources. The heterogeneous data sources in this work can be considered to be original raw data sources. Most of the time, this raw data contains noise, such as duplicated data, missing or sparse data, and data redundancy, which is not very helpful to efficiently support data-driven decision making. From the prospective view of data granularity, raw data is usually at a finer level and does not provide an intuitive meaning, which can often be more efficient for data analysis. For example, from the raw data alone, it is not always possible to tell whether a specific value is normal or abnormal, or how long a particular value has lasted. As a result, it is difficult to determine what trends are developing in the corresponding processes.

Moreover, disparate data sources can describe the same application domain using different schemas, which causes schema heterogeneity. Sparse and redundant data may occur in an integrated data set even after the data heterogeneities have been resolved. While each data source may provide a consistent data set, the integrated data can violate application integrity constraints, which can result in numerous data errors. Consider Figure 2, which shows a fragment of an integrated employment database. Tuples t1 and t4 were extracted from independent data sources s1 and s2, correspondingly, and tuple t3 was from another data source, s3. The integrated table violates an application requirement that each employee must receive one salary. This constraint is expressed as the functional dependency emp_name → salary. Violation of this constraint makes it difficult to obtain consistent answers to aggregate queries. For example, we cannot find a correct value for the total salary for all employees. As a result, we can say that tuples t1 and t4 are conflicting with respect to the functional dependency constraint, while t3 is considered to be a sparse data issue, since the s3 data source does not include Jones's salary.

	emp_name	salary	Integrity Constraint: emp_name → salary
t1	Smith	1000	(Employee can receive only one salary)
t2	Brown	2000	Tuples t1 and t4 are conflicting.
t3	Jones	Null	Tuple t3 is missing salary.
t4	Smith	4000	DB is inconsistent.

Figure 2. Example of an inconsistent integrated database

Assuming that there is only one employer and that there is only one Smith among the employees, this inconsistency may also occur if s1 and s2 refer to different time intervals where Smith received different salaries. If s1 reports Smith's salary from 1/2004 to 12/2004 and s2 reports Smith's salary from 1/2005 to 12/2005, then we can easily obtain the correct total salary value within a specific time interval. Now, consider a different scenario in which s1 reports Smith's salary from 1/2004 to 12/2004 while s2 reports it from 5/2004 to 12/2005—namely, where the reporting time intervals overlap. Obtaining the correct total salary in this case is not possible, since we do not know Smith's actual salary during the intersection of the reported time intervals (from 5/2004 to 12/2004). Most likely, this situation indicates that there was an accounting error either in s1 or in s2, which can be repaired by consulting any related employer documents.

In summary, data sources s1, s2, and s3 report on historical data about salaries. Overlapping and missing reporting periods for an employee's salary are unexpected and can indicate an error. Such an error can be fixed using readily-available additional information. The class of historical data sources considered in this dissertation also report on events of interest that occur within various time intervals. However, missing and overlapping reporting periods are expected. For example, it is common to have multiple concurrent reports about the same event within the same time interval; on the other hand, we may find no reports for a time interval. We may have hundreds of reports from different authorities about measles cases in Los Angeles for the year 1900, but no reports for a city in California for 1900.

Thus, as in the employment database, historical tuples that report the same event for overlapping time intervals may conflict and prevent us from obtaining the correct result for a query on an integrated historical database. At the same time, the nature of conflicts in historical data differs from the nature of the conflict considered in Figure 2. Historical redundant data does

not necessary imply an inconsistency or conflict in the integrated database. If the redundant historical reports are accurate, the conflicts reflect data redundancy that prevents us from obtaining accurate aggregated query results. The inconsistency may be caused by inaccurate reports. For example, two historical tuples may report different values for the same time interval. Historical data conflicts cannot be easily resolved using readily-available external information, as was the case with Smith's salary. Moreover, the number of conflicting tuples in the integrated historical data can be quite large.

This section reviews literature that is closely related to this dissertation. Since our proposed architecture implements an approach that integrates several related topics and methodologies, we have divided this review into three subsections, as explained in Figure 3. The bottom layer of Figure 3 refers to data stream management techniques that focus on the efficient processing and utilization of dynamic stream data. Related topics in this layer include data stream models, query processing techniques, and stream query languages. After raw data is fed into the system, further preprocessing is required to support efficient decision making. The middle layer of Figure 3 refers to general data preprocessing techniques, including supervised learning, unsupervised learning, and data warehousing. The top layer is related to the alerting component that discovers critical patterns in the series of severe events. In this layer, we review several studies in different domains related to event detection, anomaly detection, and decision making.

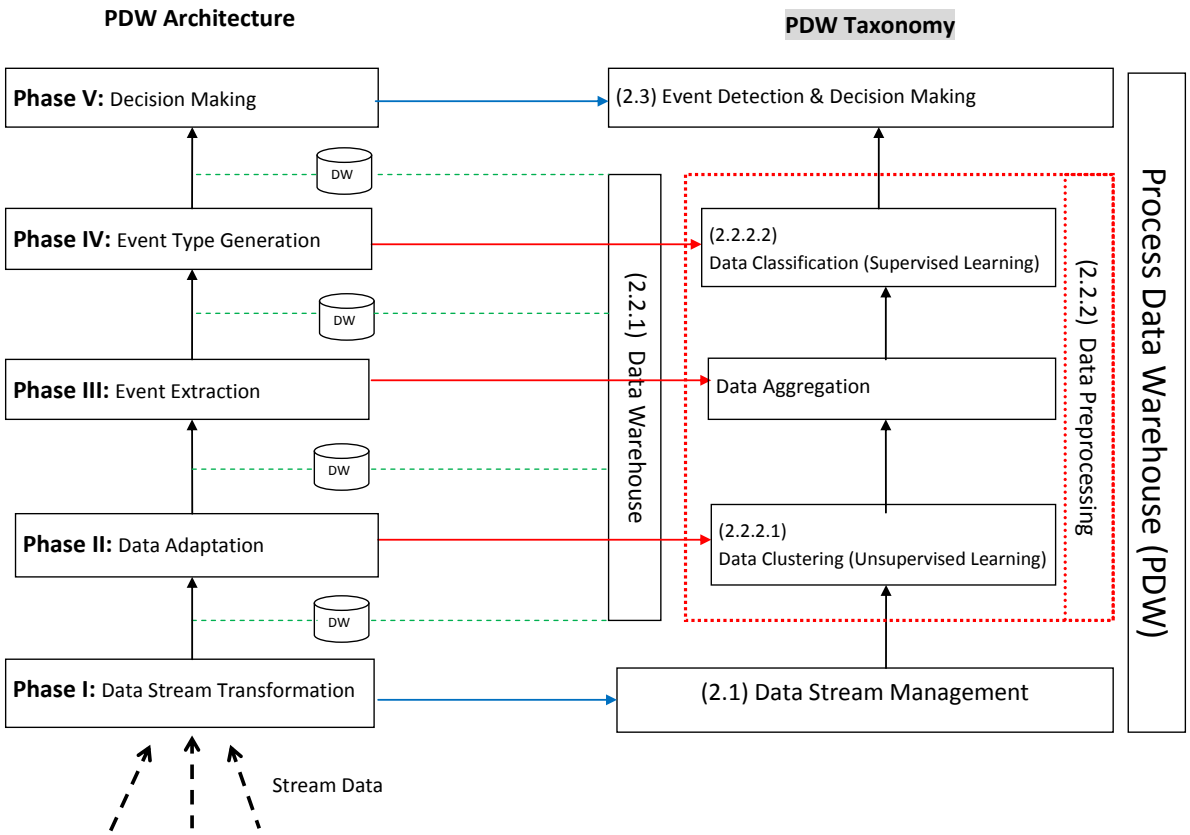


Figure 3: Taxonomy of Related Works

2.1 DATA STREAM MANAGEMENT

The data stream [2] [3] is a real-time, continuous, and ordered data sequence. The data stream system (DSS) is a system that supports, manages and analyzes data streams. DSSs are intended to support real-time applications, like sensor networks, network traffic (security) analysis, financial applications (such as real-time stock data management), and web transaction logging, among others. Since incoming data is naturally nonstop, queries in DSS also need to run continuously with incremental returning results.

The stream data model and query semantics must support long-running sequential-based and time-based operations; for example, a query continues to calculate the average value every 3

minutes. For example, SQL-TS [4], the Data Stream System project from UCLA uses SQL-like language for querying data streams. Table 1 includes an example of an SQL-TS query.

Table 1: An example of SQL-TS

Scenario	SQL-TS Query
Find the current stocks price (start upon market open) that went up by 15% or more than yesterday, and then down by 20% or more than the day after yesterday	SELECT X.name FROM quote CLUSTER BY name SEQUENCE BY date AS (X, Y, Z) WHERE Y.price > 1.15 * X.price AND Z.price < 0.80 * Y.price

Stream data models can be based on relational or object models. STREAM (Stanford Stream Data Manager) [5] uses relational models with horizontally fragmented data stored across remote nodes. COUGAR [6] uses an objected model with hierarchical data structures and methods. The STREAM system also proposes a stream data model that is designed for querying multiple continuous data streams with approximate answers when resources are limited [7]. Unlike traditional relational query language, stream query language is designed to handle continuous data. CQL (Continuous Query Language) [8] is a declarative query language used in STREAM. It deals with two major data types: streams and relations. It uses three kinds of operators: mapping relation to relation, stream to relation, and relation to stream. StreaQuel [9] and AQuery [10] are other stream query languages that use a SQL-like syntax with sliding windows and timestamps to query stream data.

As a result, interrogating and analyzing continuous data is a major task of stream query language. Efficient stream query processing becomes especially important. Since stream data is continuously collected, stream data mining [11] and clustering [12] are two other active areas of research. Recent study indicates that stream data mining notably increases the utility of streaming information [13]. For example, it has been reported that the LOCALSEARCH

clustering algorithm [14] has better CPU performance, as compared to the K-Means algorithm [15].

2.2 DATA WAREHOUSING AND DATA PREPROCESSING

Our proposed system (Figure 3) performs a multistage data preprocessing and summarizing, and the results from each phase are stored in a repository that forms a data warehouse with some degree of data summarization. In the following subsections, we review the background of data warehousing and other related techniques for data processing.

2.2.1 Data Warehousing

Data warehousing technology has been a subject of intense research and development during the last ten years. Conceptually, a data warehouse (DW) is a repository of integrated information that is available for queries and analysis [16]. The information is extracted from several operational and possibly heterogeneous data sources. A major application area of data warehouse technology is enterprise information integration [17]. Corporate operational data keeps growing both in volume and complexity. Meanwhile, the data should be effectively used for business decision making. The concept of a data warehouse was introduced to deal with this problem.

A common DW system implements a multidimensional data model that estimates certain measures of interests (such as the average sale amount) as functions of dimensions of interests (such as an individual store or product). The summarization schema is based on grouping the operational data by a dimension of interest and pre-computing simple statistics about each group (such as the total and average sale per store, region, or product, among others). In this case, a group of records in the operational database (ODB) is mapped to one record in the DW. For example, if ten customers purchased a notebook, then the ODB will include 10 records; one for each of those customer transactions. If the product category is the only dimension of interest, then the 10 records will form one group mapped to one record in the DW that stores the average sales amount from notebooks. However, if customer is a dimension of interest, then each of those

10 records will be reflected in the DW. As a result, the data summarization is sensitive to both the specific data and to the number of dimensions. The summarization becomes negligible if the number of dimensions is large and the number of generated groups becomes considerable. It is not uncommon for analytical databases to include most of the ODB records in order to maintain a complete set of measures and dimensions of interests. As the size of the operational database increases, the size of the DW increases considerably.

An important issue related to data warehousing is analyzing and handling continuous data streams [18]. Research on unbounded continuous data stream management is designed to solve those problems [19].

2.2.2 Data Preprocessing Techniques

2.2.2.1 Data Clustering (Unsupervised Learning)

In Phase II (data adaptation) of our proposed system, we group the raw data into application-dependent states of interest. Data clustering can facilitate this process. In our medical case study (Section 5 we directly adopt a domain expert's opinion for data clustering, which is considered an advantage in avoiding overall complexity. However, automatic data clustering can generally be applied in this phase. The purpose of data clustering is to split or categorize the original raw data into a certain number of groups, and thus to decrease the fine granularity of the original data to a reasonable coarser granularity, as it is expected that data items within same group have similar characteristics. The most common clustering techniques are hierarchical clustering, K-means clustering, and Fuzzy C-means clustering [20]. A major issue is in data clustering is to determine an optimal number of clusters. In [21], authors suggest the use of minimum entropy to facilitate this clustering. Clustering gain [22] could be applied as a measure of clustering optimality, which is based on the squared error sum as a clustering algorithm proceeds. It shows good performance that produces intuitively reasonable clustering configurations in Euclidean space, and it can be used to estimate the desired number of clusters.

2.2.2.2 Data Classification (Supervised Learning)

Although people do not often distinguish between data clustering and data classification, in statistics, data classification often implies supervised learning, which consists of two types of variables: independent (predictor) variables and dependent (target) variables. In Phase IV (event type generation) of our proposed system, we currently use decision tree, Naïve Bayesian Classifier, and neural networks, and our algorithm, along with all of these, belong to supervised learning.

The classification process is to learn how the values of target variables are related to the values of predictor variables. A decision tree [23] is a decision support tool that uses a tree-like model to discover possible event outcomes. There are some existing approaches that facilitate the use of a decision tree. For example, iterative dichotomiser 3 (ID3) [24] and its extensions, C4.5[25], use information gain instead of the Gini Diversity Index, which is used in CART. Unlike CART, these approaches allow more than two (binary) splits during the induction of decision tree, which can provide more flexible and more informative trees.

Moreover, the decision tree has been applied to many domains with various extensions. The very fast decision tree (VFDT) system [26] is a decision tree induction system that is capable of learning from a high-speed data stream and making assumptions about the data. The fuzzy decision tree (FDT) [27] is a duration modeling technique that focuses on text-to-speech (TTS) analysis. In some cases, sampling techniques [28] are applied to reduce the amount of incoming data with an acceptable query result error range.

2.3 PROCESS HANDLING, EVENT DETECTION AND DECISION MAKING

2.3.1 Process Mining

The goal of process mining is to extract information from event logs by generating a process model [29]. Events are sequential records and an event can be referred to an activity. ProM [30] is a platform that supports many process mining techniques by using pre-defined plug-ins. Dotted chart [31] is a visualization tool in ProM that checks the data profile. It shows the event

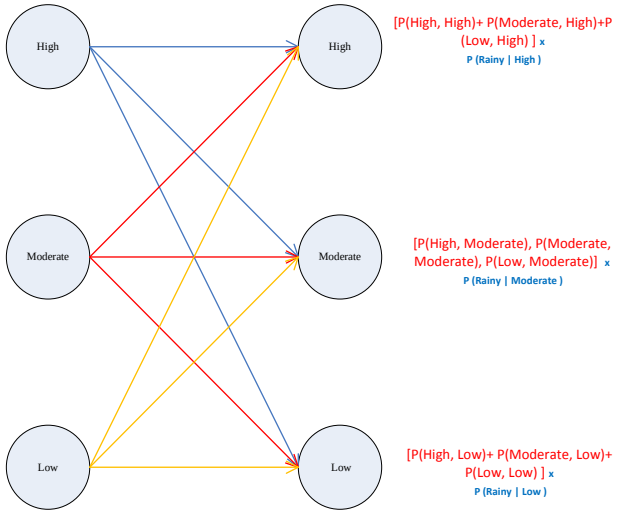
logs in a graphical way (time/component) to discover patterns. ProM provides some algorithms designed for decision making, such as genetic process mining [32], workflow mining [33], and Heuristic miner algorithm [34]. A potential issue in ProM is that process mining is often performed on a case-by-case basis, and that sometimes it is difficult to find a general model to represent the input data. Although ProM offers many tools, it could be difficult to choose a proper one to start. Users need to have some domain knowledge to use the platform, since some tools or algorithms generate poor results. Moreover, process mining has problems when it comes to discovering hidden information and noise removing for certain structured data, such as that found in a data stream [31] [29].

2.3.2 Hidden Markov Model (HMM)

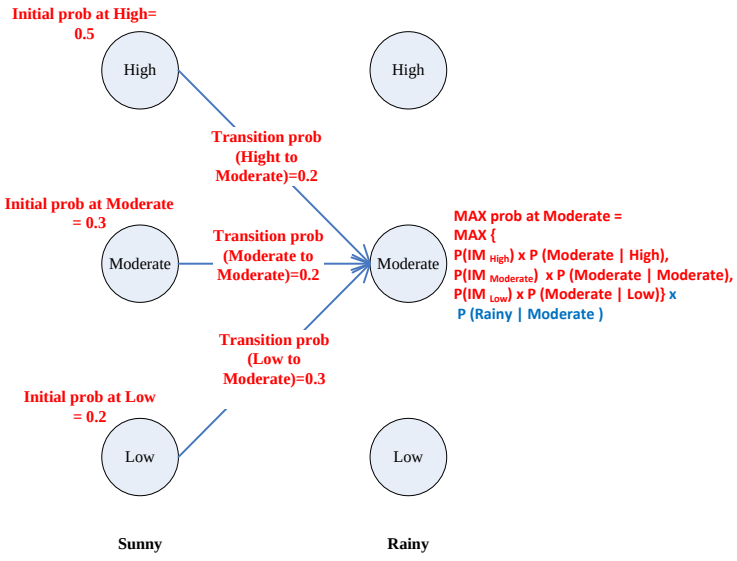
The hidden Markov model (HMM) explores hidden probabilistic trends in large data streams. [35]. The HMM is a triple parameter set, that is $\lambda = (\pi, A, B)$ where

- π (Initial matrix): describes the probability of the hidden state at the beginning (t=1)
- A (state transition matrix): represents the probabilities of transitions from one state to another state
- B (Observation matrix, also called confusion matrix, or emission matrix): contains the probability of an observable state, given a hidden state.

There are three basic problems [36] that can be solved by corresponding HMM algorithms. These algorithms are the forward algorithm, the Viterbi algorithm, and the forward-backward (Baum-Weich) algorithm. The forward algorithm [36] finds the probability of an observation sequence, given an underlying sequence of hidden states. That is, given the observation sequence $O=O_1\ O_2 \ldots O_T$ and model λ, how do we efficiently compute $P(O \mid \lambda)$? Without using the forward algorithm, the complexity is $O(2TN^T)$, where T is the length of sequences and N is the number of states. The forward algorithm reduces the complexity is $O(2TN^T)$ to $O(N^2T)$. The concept of the forward algorithm can be briefly illustrated by the following diagram.



The Viterbi algorithm [36] is used to find the most probable state transition sequence, given an observation sequence. That is, given the observation sequence $O=O_1\ O_2 \ldots O_T$ and model λ, how do we efficiently find the most likely (optimal) corresponding state sequence $Q = q_1\ q_2 \ldots q_T$? Similar to the forward algorithm, the Viterbi algorithm reduces the complexity by using recursion, and its complexity is $O(N^2T)$. The concept of the Viterbi algorithm can be briefly illustrated by the following diagram.



The forward-backward algorithm [36] is used to adjust the model $\lambda=(\pi, A, B)$ for the purpose of maximizing the probability of observation sequence $P(O \mid \lambda)$. It uses iterative procedure to approach local maximum of $P(O \mid \lambda)$. It is derived from EM algorithm and it calculates the forward probability and the backward probability for each HMM state. The forward-backward algorithm has the time complexity of $O(N^2T)$ instead of the brute force procedure $O(2TN^T)$.

In typical HMMs and our approach, the state durations are assumed to be the same (fixed). This is not natural for some dynamic systems that have different state sojourning times[37]; namely, the time intervals for such systems staying in each state are variable. Some literature has addressed this problem and under different names, such as duration-dependent HMMs[38], hidden semi-Markov models (HSMMs)[37], and variable-duration HMMs (VDHMMs)[39]. Our approach assumes that the state duration is fixed.

2.3.3 Anomaly Detection

STUCCO [40] is a rule-based detection method that compares recent data against a baseline distribution with the goal of finding rules to summarize significant patterns of anomalies. STUCCO uses a SQL-like query to find a set of data with the same values. A select criteria/rule is made up of one or more components, ***Attribute(i) = Value(j)***. Multiple components are joined together by a logical ***AND***. For example, suppose we have historical chief complaint data from a hospital emergency department. If we want to count the number of people who are male and have homes that are located in the northwestern side of a certain city, we can apply the following rule with two components.

Gender = Male AND Home Location = NW

STUCCO detects anomalies by testing if the distribution/count of the data selected by the rule has a significant difference between baseline data and recent data. STUCCO uses *Chi-Square* to find the overall significance.

Bio-surveillance is another domain with a considerable amount of research on anomaly detection. Many studies have been conducted that are based on multiple sources of medical-related data, such as pharmacy sales, hospital visit data, vital signs data, weather indicators, and census data. Some techniques have also been designed for bio-surveillance studies. A previous

study [41] proposed a rule based on an algorithm for performing early detection of disease outbreaks by searching databases from emergency departments to find anomalous patterns. The WSARE algorithm [42] tries to identify anomalous patterns in data from emergency departments. By analyzing the data, an early warning system attempts to detect disease outbreaks. Instead of focusing on individual data points, the WSARE uses a scoring algorithm to eliminate outliers. Most of this temporal data is numerical, but unlike previous studies, [43] conducts a study on detecting abnormal patterns in categorical datasets. The study consists of two phases. Local anomaly detection is the first phase, which adopts the Bayesian network to search and learn about abnormal patterns in historical data. In the second phase, the study uses the WSARE approach to continue monitoring the new data stream and check whether any new values are above the threshold value.

Several studies have been conducted on bio-surveillance event detection. One example is a study on maintaining the safety of agriculture and the food supply [44]. This project uses multiple data streams of food and agriculture safety data obtained from the USDA. This data is collected periodically. Researchers conduct animal and plant safety analysis with the capability of monitoring temporal heterogeneous data. Anomalous pattern discovering research [45] has also been conducted based on retail pharmacy data. The scope of this project involved collecting over-the-counter (OTC) sales from the real-time outbreak and disease surveillance (RODS) laboratory. About 9000 OTC drugs were considered in this study, such as baby electrolytes. By conducting these sorts of studies, it is possible to track and monitor disease outbreak and public health regulations.

2.3.4 Medical Early Warning System

An early warning scoring system (EWSS) [46] is being used successfully in many hospitals in the United Kingdom, and a pediatric early warning system (PEWS) [47] has also been shown to efficiently monitor a patient's condition. However, this scoring system has been designed mainly for nurses. Patients are initially examined by nurses, but it turns out that the nurses often fail to notify physicians of patients' clinical deterioration. In pediatric emergency departments, the ability to predict which children will need to be admitted to the hospital, and in which clinical

setting that child should be further monitored, has great value for patient care and the effective use of limited bed space. Existing physiologic scoring systems have traditionally been used for outcome predictions, quality of care analyses, and benchmarking in intensive care unit (ICU) settings [48-50], as well as in inpatient floor settings [51, 52]. All these systems use physiologic measurements, as well as other clinical variables, to score the severity of a patient's illness. These systems have also been used in emergency departments [53, 54]. They create an overall risk stratification, which helps to predict either mortality (APACHE IV, PIM, PRISM III) or whether a patient may be in danger of cardiopulmonary arrest (PEWS). However, no warning system has been developed to date that can be used throughout the entire duration of a child's hospital stay to monitor and predict a child's clinical status.

Microsoft HealthVault [55] has been released to the general public since 2007as a service in a test stage. It has a platform to store customers' records, as well as access to directly place real-time medical information into Google's online records. Despite all the data available, it still has not yet been universally adopted, and patient privacy is another issue that needs to be solved. Obviously (and inevitably), there are not enough physicians, nurses, or technicians able to continually monitor this vital data stream. Our case study in this paper can provide a solution for this issue, because our technology can automatically and continually monitor patients' (or customers') vital dynamics, and if any abnormal signs occur, can notify patients (or customers) to come to the hospital for advanced diagnostic services.

2.4 PROCESS HANDLING, EVENT DETECTION AND DECISION MAKING

In this section, we explore methods of state sequence recovery (SSR) in cases of redundant and inconsistent data. We are not aware of previous works that systematically address this problem. Meanwhile, there are a number of related studies in the area of *redundant and inconsistent data management*, as well as in *state sequence analysis*.

2.4.1 Redundant and inconsistent data management

In database systems, data redundancy is characterized as data that is repeated in two or more tables. The redundant values may be inconsistent and require further preprocessing, data normalization, and duplicate elimination processes [56, 57] . Duplicate elimination methods are usually domain-dependent [56] and those methods often rely on similarity functions and thresholds for evaluation and removing redundant data[58]. For example, previous studies [59, 60] have explored classification methods and learning algorithms to analyze the impact of data inconsistencies. In sensor networks, redundant data manifests when multiple data sources are used to represent the same object. Sensor networks often deal with issues of data redundancy in order to develop scalable, fault-tolerant methods to extract useful information from data sources[61-63]. Moreover, due to power and range constraints, such centralized approaches are generally impractical. Many proposed solutions use in-network aggregation to reduce overall network traffic[64]. For example, TinyDB [65] and Cougar [63] are two sensor database systems that allow users to perform aggregate queries, such as MIN, COUNT, and AVG, within the sensor network. In our study, we use data aggregation to generate relevant application states for the original values and estimated values. There are several approaches to deal with redundant and inconsistent data. [66] introduces two algorithms, named Cottus and Gyes, that are designed to dynamically interact with environments without relying on limited pre-computed solutions. [67]and [68] try to build optimal aggregation structures that can further deal with network volatility and can compensate for the loss or duplication of data. Both database redundancy management and data redundancy in sensor networks are related to our approaches in the field of aggregation for state generation and optimization for data conflicts. Meanwhile, none of the related approaches considers state sequence recovery in redundant datasets.

2.4.2 State sequence analysis

The area of state sequence analysis may include topics of finding optimal state sequence, pattern recognition, and data clustering. The hidden Markov model (HMM) has been applied and studied for state sequences in many domains. HMM parameter estimation can be performed by the maximum likelihood method through the expectation maximization (EM) algorithm [69]. The Viterbi algorithm [70] is a well-known approach that predicts the most probable (optimal) state sequence, given a specific observation sequence. Several studies [62, 70, 71] have investigated rare events using pattern recognizers that are based on HMMs or support vector machines (SVM). In the field of data clustering in HMMs, [72] demonstrates an effective regression method to evaluate time series data with comparisons to some existing approaches for time series clustering, which include the standard EM algorithms for Gaussian mixtures, K-means clustering, the standard mixture of regression models, and the mixture of HMMs. Several clustering methodologies, such as those proposed in [73] and [57] are based on the calculation of the degree of similarity between multivariate time-series datasets using two similarity factors. One similarity factor is based on principal component analysis (PCA) and the angles between the principal component subspaces, while the other is based on the Mahalanobis distance between the datasets[74].

Other ongoing works that are related to our study include the following. Previous studies [67, 75] used efficient estimation methods for the parameters in the multivariate Markov chain models for a huge number of states. [76] proposed an efficient reformulation of a Markov clustering algorithm that is suitable for the fast and accurate grouping of protein sequences, based on pairwise similarity information. [77] and [78] have showed a novel methodological framework for analyzing the relationship between state sequences. Those studies usually include the performance of clustering results and similarity between two sequences of data, either in the forms of raw data, extracted features, or in the state sequences [69, 73]. These studies are all related to our state sequence analysis. We do not use sophisticated methods to group data into states; we consider this to fall under the category of future work.

There are many ongoing studies in the areas of redundant data management and state sequence analysis. To the best of our knowledge, there are no related works that explore state sequence recovery tasks for redundant and inconsistent data.

3 THE PROCESS DATA WAREHOUSING APPROACH

As discussed in Chapter 1, original historical heterogeneous data is not helpful to efficiently support data-driven decision making, because most of time, the raw data contains noise; such as duplicated data, missing data (sparse data), and redundant data. By looking at the raw data alone, it is not always possible to tell if a specific value is normal or abnormal. We also do not know how long a particular value has lasted, and it is difficult to determine what trends are developing in the corresponding processes. Our major goal is to develop methods and techniques to extract information from raw historical data sets in order to support efficient decision making. To achieve this goal, we will apply advanced data aggregation and warehousing techniques for information extraction from large-scale data streams. Our proposed process data warehousing (PDW) approach includes the system architecture, as well as data processing and analyzing techniques that provide for the general applicability of this approach to large-scale historical heterogeneous data integration systems that address two major challenges: (1) sparse process data, and (2) redundant process data.

3.1 GENERAL ARCHITECTURE OVERVIEW

Figure 4 shows the general architecture of the PDW system. It accepts a raw data source and performs its multistage process. Below, we outline functionality of each data aggregation phase. More detailed descriptions will be provided in the next section.

Phase I (Data Stream Transformation): In this phase, the PDW system transforms the raw input data into a semantically richer, tuple-based data stream. This may involve solving various data extraction and cleansing issues, and handling errors, outliers, and missing values in the raw

data. The complexity of the extract, transform, and load (ETL) process can vary, depending on the application and data quality.

Phase II (Data Adaptation): In this phase, the PDW system decreases the granularity of the raw data by grouping the data items into a number of meaningful application-dependent states. This can either be done automatically using data clustering techniques, or done via embedded mapping rules specified by an application expert. This state grouping task is crucial to the data aggregation process.

Phase III (Event Extraction): Since we have decreased the data's granularity in the second phase, different raw items could be mapped into the same state. This may result in state repetitions that occur in the consecutive tuples of the data stream. Phase III further aggregates the tuples with identical states into events and associated durations. As a result, in this phase, PDW groups same states within consecutive rows that sum up their durations.

Phase IV (Event Type Generation): In Phase III, we grouped event data with its duration. In this phase, our PDW system performs an assessment of the events generated in Phase III. Event severities are mapped to their corresponding event types. In this stage, the PDW system will use a number of data analysis algorithms, as will be explained in Section 3.1.4.

Phase V (Decision Making): After obtaining sequences of event types from Phase IV, the PDW system performs their analysis in order to recognize notable patterns in the process environment. The system applies embedded decision making algorithms that continuously monitor trends in sequences of event types. For example, an early warning signal (flag) is raised whenever abnormal patterns are identified. In other words, the PDW system provides a "flag" as the severity indicator for early warnings in decision making.

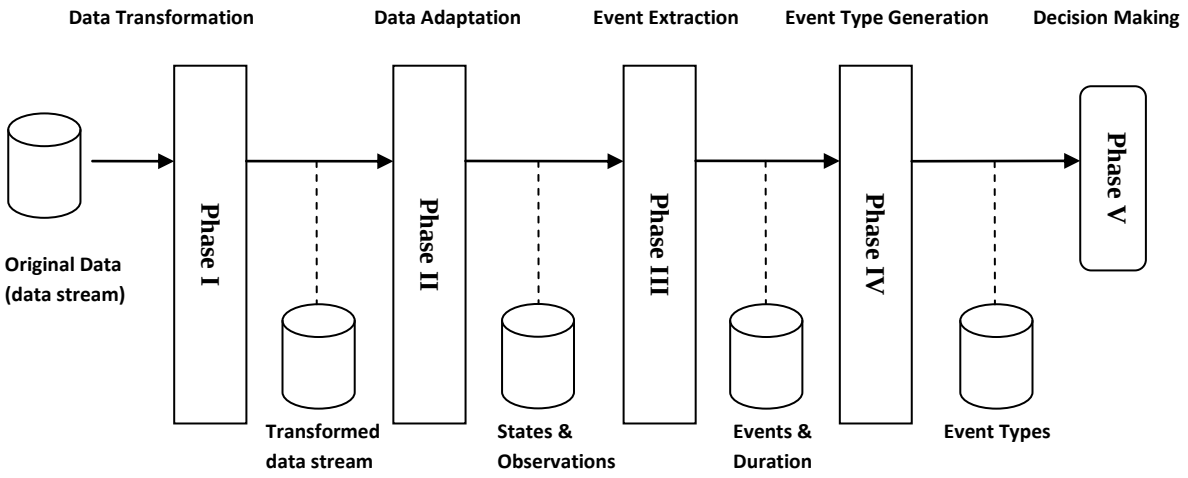


Figure 4: The PDW Architecture

3.1.1 Phase I: Data Transformation

Data integration (DI) is the initial step in applying data fusion (DF) methods. It is the process of combining, integrating, or transforming multi vector data sources, and can facilitate decision making. In this thesis, we will also cover advanced data aggregation and warehousing techniques. Ideally, large-scale data sources are collected by automatic equipment (such as sensors) in the form of sequences of multivariable vectors. However, the input data may be recorded by humans or consolidated from multiple heterogeneous data sources, which may result in overall inconsistency and incompleteness.

Based on these situations, for data-driven decision making, the data should be pre-processed and aggregated. This pre-processing is similar to the function of an extract, transform, and load (ETL) tool [79]. It results an appropriate data stream format that includes time-stamped multivariable vectors (tuples) as shown in Table 2. In this phase, the PDW system performs various data transformation functions, including:

- Merging different data sources
- Converting data types to a uniform format

- Grouping similar or duplicated data
- Reordering the data based on specific data columns
- Changing the dimensionality of data through pivoting and unpivoting operations.

Table 2: Time serial data format

	Multivariable Vector				
Time Stamp	**Variable 1**	**Variable 2**	**Variable 3**	**....**	**Variable K**
Recorded time	Value	Value	Value	...	Value

3.1.2 Phase II: Data Adaptation

The historical data reports on events of interest that occur within various time intervals. It can provide different, mutually independent views of targeted domains. For example, demographic data from the state of Pennsylvania does not necessarily relate to the demographic data from the state of California. Combining different parts (states) of the demographic data may provide a more complete picture of the United States. However, this combination may also include severe data conflicts that prevent researchers from obtaining the correct answers to queries on an integrated historical database. It is common to have multiple concurrent reports about the same event within overlapping time intervals. At this stage, the functionality of the PDW is designed to provide a better representation of the data, which may include transferring the raw data to more meaningful states or estimating/recovering the original data values (data validation). The latter part does not have a standard procedure, since data validation is highly domain dependent.

As for transferring data to a meaningful state, once the raw data is transformed into a stream of time-stamped tuples, the data is prepared for the second pre-processing step. The PDW system groups the data items into a number of application-dependent states. This can either be done automatically using data clustering techniques, or through embedded mapping rules that are specified by an application expert. The state grouping task is crucial to the data aggregation

process. States are the meaningful representation for the data, such as normal / abnormal or low / normal / high.

By using domain-specific mapping rules (data clustering techniques), data can be categorized in different states. We illustrate this phase with a simple weather measurement example in Table 3. Without loss of generality, we can assume that the application delivers a set of interdependent numeric data streams. The second left column from Table 3 includes an input data stream that consists of two variables: cloudiness and temperature within several observation periods (leftmost column). Cloudiness is measured in oktas [WWC]. The third column of Table 3 illustrates simple mapping rules that assign a cloudiness state {"cloud", "sun"} and temperature state {"cool" ,"warm" } to specific ranges. The rightmost column of Table 3 illustrates an output stream of states that uses the mapping rules.

Consider another example from the medical domain, where there is a range of normal values for some vital signs. For example, a normal temperature is defined as a temperature between 36.4ºC and 37.5 ºC. Other vital signs, such as heart rate, have different ranges based on the age of the patient. In the data adaptation step, the raw values would be grouped into states that correspond to clinically relevant categories.

Table 3: Mapping input data streams into states of interest

Record Time (T)	Input Data Stream		Mapping Rules	Stream of States	
T =	Cloudiness	Temperature	Cloudiness	Cloudiness	Temperature
10	6	35	cloud: Cloudiness ≥ 5	cloud	cool
20	7	40	sun: Cloudiness < 5	cloud	cool
25	7	50		cloud	warm
45	2	60	Temperature	sun	warm
55	0	55	cool: Temperature < 45	sun	warm
65	1	30	warm: Temperature ≥ 45	sun	cool
75	7	25		cloud	cool
80	1	20		sun	cool

Generally speaking, a mapping rule could come from one of the following three ways.

1. Domain expert method: the mapping rule comes directly from the domain expert or from existing well-developed rules, like in the medical domain [47] in our case study in Section 5.
2. Automatic discovery method: This method is used when the mapping rules are not previously known. Techniques such as correlation and clustering algorithms [20] are used to discover the mapping rules. In our system, we have included several unsupervised clustering techniques such as K-means, fuzzy C-means, and hierarchical clustering, which can be used for the data classification engine.
3. Combination method: This approach uses both the first and second techniques, and may be useful in highlighting groups that may be missed by only using a domain expert.

3.1.3 Phase III: Event Extraction

The functionality of our PDW at this stage is designed to merge the data with the concept of a time interval. In some cases, the historical data can be considered to be event data if it is associated with durations. For example, say that 200 disease cases happened in 1999. In many other cases after decreasing the data granularity in the second phase, different raw values can be mapped into the same states. This may cause duplicate states to occur in consecutive tuples. At the same time, collecting time-related serial data often requires a great deal of disk space and consumes many processor and memory resources. Thus, we can further aggregate the data by merging the tuples with the same states into "events" where all states remain unchanged for a given period of time. During this time period, there might have been many recordings of individual variables, but if they all fall into the same general state, then they are grouped together. Table 4 shows the events generated from the stream of states taken from Table 3. For example, first two stream tuples in Table 3 were aggregated in one event, since they have the same values of cloudiness and temperature. The duration of this event is the difference between the timestamp of those tuples.

Table 4: Event Table of Cloudiness & Temperature from Table 3

Duration	Event	
25-10=15	cloud	cool
45-25=20	cloud	warm
65-45=20	sun	warm
75-65=10	sun	cool
80-75=5	cloud	cool
	sun	cool

We formalize this process of event generation as a new continuous group by the (CgroupBy) operation, as discussed below.

- **CgroupBy Operation**

The "group by" clause in the standard SQL feature aggregates a set tuples with the same value of a grouping. In contrast to that, the continuous group-by operation CgroupBy aggregates *continuous* groups of tuples with the same value of a grouping attribute. We extended the relational algebra with a Cg operation that reflects the continuous grouping operation as follows:

$$_{G1,\ G2,\ldots,\ Gn}\ Cg_{f(A1\ldots.Am)}\ (E),$$

where *E* is the output event table; *G1,G2,….Gn* constitute a list of attributes on which to group; *f(A1…Am)* is an aggregate function which aggregates the duration (sum) based on the combination of attribute names *(Ai)*. The meaning of the operation is as follows. The table *E* is partitioned into groups in such a way that all consecutive tuples with the same values of the grouping attributes *G1,G2,….Gn* form a single group.

Consider an example from Table 3. The result of continuous grouping from the cloudiness data can be expressed as follows:

$$_{Cloudiness\_Event}\ Cg_{Sum(Cloudiness)\ as\ Duration}\ (Event\_Table\_Cloudiness).$$

This corresponds to the following extended SQL query:

```
Select Duration, Cloudiness as [Cloudiness_Event]
From Table_3
CgroupBy Cloudiness
```

The result of this query is shown in Table 5.

Table 5: Results of CgroupBy query

Event Table of Cloudiness	
Duration	Cloudiness_Event
45-10=35	Cloud
75-45=30	Sun
80-75=5	Cloud
	Sun

Note that the CgroupBy functionality can be expressed in SQL extension, but with very considerable efforts. Figure 5 shows a Microsoft T-SQL query [80] that is equivalent to the above CgroupBy in the extended SQL. We observe that the T-SQL code is practically unreadable; meanwhile, it can be generated as a result of compilation of the CgroupBy query. Figure 6 describes the CgroupBy algorithm.

```
Definition:
      Att: an attribute (column) in the input data table
      D: duration column in the event table
      Event_table: the final result table
T-SQL Query:

select Att,Tot_Duration from (
select  *,SUM(Duration) over(partition by Att,rn_2) Tot_Duration
from (
select *,rn-rn_1 rn_2 from (
select *,ROW_NUMBER() over(partition by Att order by rn) rn_1
 from (select *,ROW_NUMBER() over(order by (select 1)) rn
        from Event_table) t1) t2) t3) t4
group by Att,Tot_Duration,rn_2
order by MIN(rn)
```

Figure 5: Event generation using Microsoft T-SQL

Definition:

Input: Input_table = {states (St), time_stamp(T)
Output: Event_Table
D: duration column in the Event_Table

Algorithm:

```
add_rows = 0;
For k = 1 to num_of_rows (Input_table)-1
  k = k + add_rows;
  // if does not have duplicated State value
  If St_k ≠ St_(k+1)
    This_duration = T_(K+1) - T_K;
         Event_Table (end+1) = [This_duration, St_k];
    add_rows = 0;
  Else // Has duplicated State value

    //sum duration
    cnt = 1;
    While St_k == St_(k+1+cnt)
     sum_duration = T_(k+1+cnt) - T_K ;
     cnt = cnt+1;
    End // end while

         Event_Table (end+1) = [sum_duration, St_k];
    add_rows = cnt +1 ;

  End//end if
End //end for

return: Event_Table
```

Figure 6: Event generation algorithm

3.1.4 Phase IV: Event Type Generation

Next, the PDW system utilizes various data analysis strategies to assess event types. Informally, the event type reflects the severity of the event. Our consideration of different data analysis strategies reflects the fact that there is no single strategy that would be best for a particular application domain. Based on different event analysis strategies, we can generate different event types. The event type should be mapped in an application dependent scale of event severities (such as "high," "low," "normal," or "critical," among others). In redundant historical data, the degree of redundancy can be also considered to be an event type, such as “redundant,” “highly

redundant," or "slightly redundant." Table 6 shows an example of a single event type, extended from Table 5. Here we assume that "cloudy" is an undesirable weather condition and that longer periods of cloudy skies would correspond to a more severe event type.

Table 6: Event Types

Event Table			
Duration	Event for Cloudiness	Algorithm Interpretation	Event Type (Severity)
35	Cloud	→	Critical
30	Sun	→	Normal
5	Cloud	→	Low

We consider two kinds of event types: single event types and combined event types. Single event types are generated from a single variable (column) with the corresponding duration, while combined event types are generated through an aggregation result from the single event types. We provide more detailed explanation and illustration of single and combined event types in the next section.

3.1.5 Phase V: Decision Making Methodologies

After generating sequences of event types, the PDW system performs further data analysis by using embedded decision making algorithms that monitor trends in event severities. For redundant process data, it may include severe data conflicts caused by database redundancy that prevent researchers from obtaining the correct answers to queries of an integrated historical database. We propose a novel conflict-aware data fusion strategy as a solution to this issue and examine it in Section 6, where we demonstrate that our approach significantly reduces data aggregation errors in the integrated historical database. As another example of sparse process data, an early warning signal (flag) is raised whenever abnormal patterns are identified. In other

words, our system provides a "flag" as a severity indicator for early warnings in decision making. Section 6 provides a detailed description of this approach.

It is possible to apply our PDW to real time data environments; in this case, the delivery of a data stream may result in delays between consecutive readings or measurements. This may generate longer event durations. In this case, the PDW system will unfold or decompose those events into shorter events, according to an observation cycle. This approach makes the PDW system generate warnings more quickly. Based on certain patterns in that sequence, a warning signal (flag) is raised whenever abnormal patterns are identified. The long-lasting events may indicate that the process information is obsolete, which itself may be considered to be a severe event. Consider the example shown in Table 6 and assume that the system unfolds events when the duration (observation cycle) exceeds 15 minutes. In this case, the data from Table 6 is unfolded as shown in

Table 7. In this case, the PDW system would have a chance to raise an early warning flag before the cloudiness conditions become severe (namely, while moving from a low to a high cloudiness value). More illustrations of the *unfold* time procedure will be provided in Section 4.

Table 7: Unfolded results from Table 6

Event Table			
Duration	Event for Cloudiness	Algorithm Interpretation	Event Type (Severity)
15	Cloud	→	Low
15	Cloud	→	High
5	Cloud	→	Critical
15	Sun	→	Normal
15	Sun	→	Normal
5	Cloud	→	Low

Next, we introduce a rule-based analysis to the sequence of events. Those rules could be thought of as intelligent agents that discover dynamic patterns in the development of event severities. If certain pre-defined conditions are met, the warning flag is raised. We use the

concept of triggers in the database system to illustrate our rules and to specify them using T-SQL syntax. Here we provide generic rule specifications, but in Section 4, we will provide some application-specific refinement of our rules for the medical domain. In this case, we assume that an event has either an N (normal) event type or an Ab (abnormal) event type, and K is an application-specific threshold value.

- **Rule 1**

Definition: If the number of Ab is ≥ K, then Raise Flag. Figure 7 shows the Rule using TSQL

```
CREATE TRIGGER Rule_1
ON Event_table
FOR UPDATE , INSERT
AS

DECLARE @ID INT
DECLARE @time_stamp INT
DECLARE @E_value char(2)
DECLARE @num_Ab INT
set @num_Ab = K --set the flag condition (number of abnormal events)

SELECT @ID = ID, @time_stamp = time_stamp, @E_value = E_value FROM
Event_table

INSERT INTO Flag_table --flag raised
  SELECT @ID, @time_stamp, @E_value

WHERE EXISTS
(
  SELECT * FROM
  (
      SELECT E_value, COUNT(time_stamp) AS num_Ab_events
      FROM Event_table
      GROUP BY E_value
      HAVING E_value = 'Ab'
  )A
WHERE num_Ab_events >= @num_Ab --catch the condition
)
```

Figure 7: Rule 1 in TSQL

Figure 8 illustrates the application of Rule 1 for K = 3.

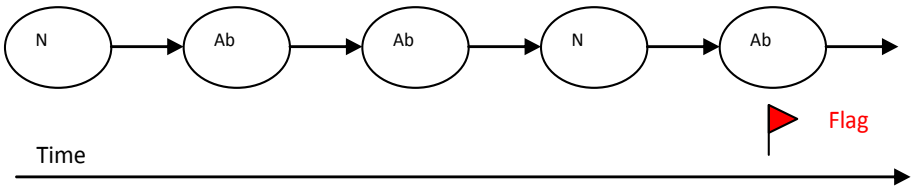


Figure 8: Rule 1 with 1 flag

We may also raise the red flag after the second application of the rule (namely, after discovering two notable patterns). In Figure **9** , the red flag is raised after six "Ab" event types.

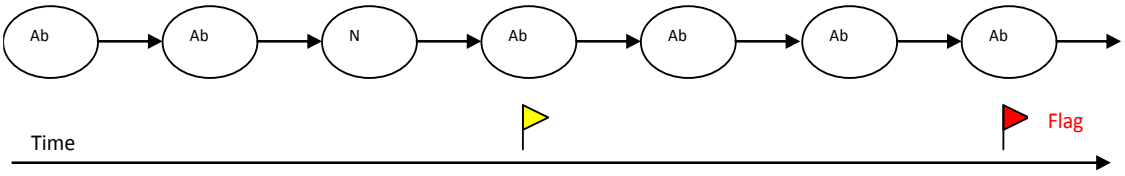


Figure 9: Rule 1 with 2 flags

Figure 10 illustrates a slightly different strategy that includes the previous 1 Ab event type when applying the warning rule for a second time. Thus, the red flag can be raised earlier, as compared to Figure 9.

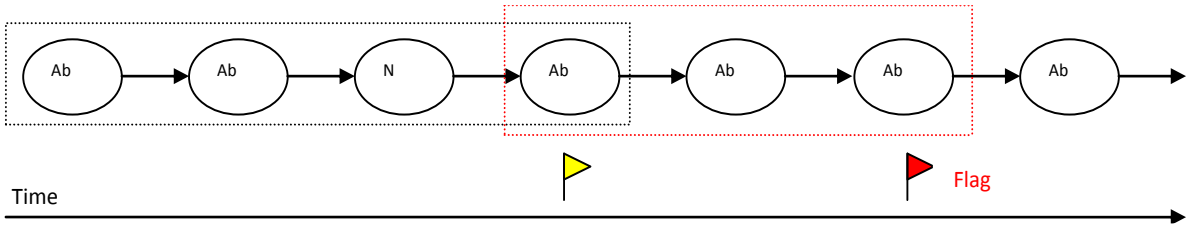


Figure 10: Rule 1 using overlap (including previous 1 Ab) with 2 applications (flags)

- **Rule 2**

Rule 2 assumes that an N (normal) event can cancel an Ab (abnormal) event.

Definition: If (number of Ab) - (number of N) ≥ K, then raise the flag. Figure 11 shows Rule 2 in TSQL.

```
CREATE TRIGGER Rule_2
ON Event_table
FOR UPDATE , INSERT
AS

DECLARE @ID INT
DECLARE @time_stamp INT
DECLARE @E_value char(2)
DECLARE @threshold INT
set @threshold = K --set the threshold (difference between Ab and N
events)

SELECT @ID = ID, @time_stamp = time_stamp, @E_value = E_value FROM
Event_table

INSERT INTO Flag_table --raised flag
  SELECT @ID, @time_stamp, @E_value

WHERE EXISTS
(
SELECT * FROM
(
  SELECT num_Ab_events, num_N_events FROM
   (SELECT E_value, COUNT(time_stamp) AS num_Ab_events
      FROM Event_table
      GROUP BY E_value
      HAVING E_value = 'Ab') Ab
  CROSS JOIN
   (SELECT E_value, COUNT(time_stamp) AS num_N_events
      FROM Event_table
      GROUP BY E_value
      HAVING E_value = 'N') N
) A
WHERE num_Ab_events - num_N_events >= K --catch the condition
)
```

Figure 11: Rule 2 in TSQL

Figure 12 illustrates that a third normal event cancels one previous abnormal event and, as a result, the raising of the warning flag occurs later (event 5). Here, we assume K=3.

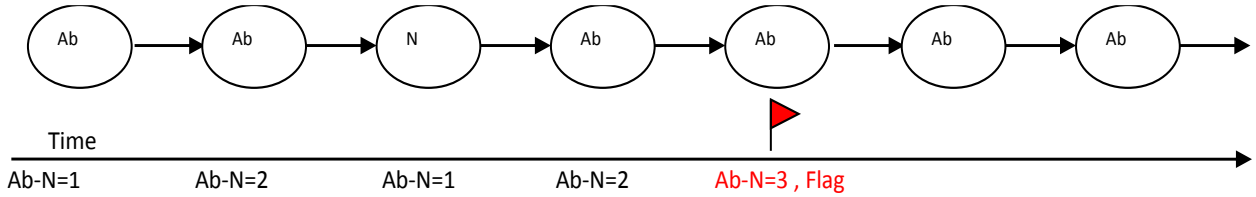


Figure 12: Illustration of Rule 2

- **Rule 3**

Rule 3 uses a sliding time window to determine when to trigger the flag. We set the window size as S and keep sliding the time windows by F events.

Definition: If (number of Ab) is ≥ K inside the window, then raise a flag. Figure 13 shows Rule 3 in TSQL

```
CREATE TRIGGER Rule_3
ON Event_table
FOR UPDATE , INSERT
AS

DECLARE @ID INT
DECLARE @time_stamp INT
DECLARE @E_value char(2)

DECLARE @table_size INT
DECLARE @window_size INT
DECLARE @shift_size INT
DECLARE @num_ab_in_flag INT

DECLARE @src_event INT
DECLARE @dest_event INT
SET @window_size = S
SET @ shift_size = F
set @num_ab_in_flag = K --set the threshold (Ab exceeds certain value)

SET @Src_event = 1
SET @Dest_event = @window_size
SET @table_size = (select count (*) from Event_table)

SELECT @ID = ID, @time_stamp = time_stamp, @E_value = E_value FROM
Event_table

INSERT INTO Flag_table --raised flag
  SELECT @ID, @time_stamp, @E_value

WHERE EXISTS
(
  SELECT * FROM
  (
   WHILE @Dest_event <= @table_size
    BEGIN
             SELECT E_value, COUNT(time_stamp) AS num_Ab_events
             FROM Event_table
             WHERE ID between @Src_event and @Dest_event
             GROUP BY E_value
             HAVING E_value = 'Ab'

             SET @Src_event = @Src_event + @shift_size
             SET @Dest_event = @Dest_event + @shift_size
     END
  )A
WHERE num_Ab_events >= @num_ab_in_flag --catch the condition
)
```

Figure 13: Rule 3 in TSQL

Consider an example with a sliding window size of (S=3) events. As a new event is recorded, the window slides forward by (F=1) event. A flag will be raised as soon as there are (K=2) Ab events within a frame. Similar to Rule 1, we can raise the flag after, 1, 2, or 3 applications of Rule 3. Figure 14 and shows that first flag occurring at the first window. As another event is recorded, the window slides, so no flag is raised. Finally, the second flag is confirmed in the third frame at the end of event 5.

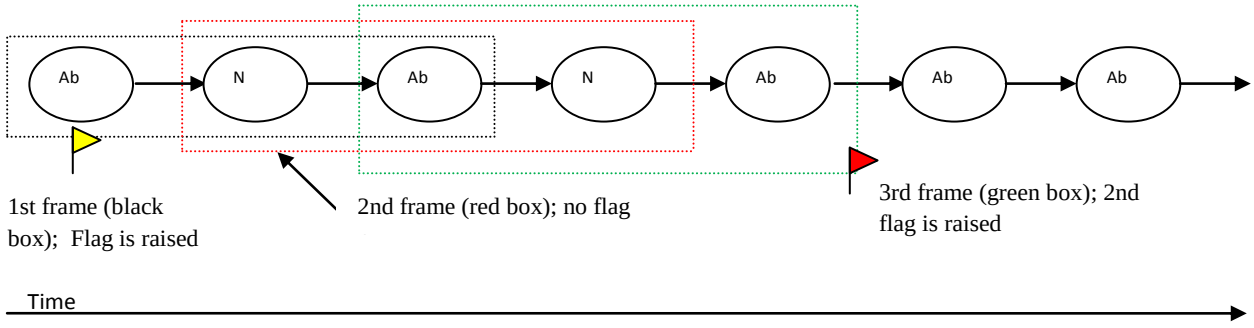


Figure 14: Illustration of Rule 3

3.2 ALGORITHMS COMPLEXITY ANALYSIS

In this section, we perform time and space complexity analysis for those algorithms originally designed in this thesis. Time complexity is a measurement of how the computation time is taken by a program growing with input data. Similarly, the space complexity is the total space (memory) taken by the algorithm with respect to the input amount of data. Space complexity may include both auxiliary space and space used by input. In the following paragraph, we will discuss both complexities of algorithms in our PDW.

CgroupBy Algorithm: CgroupBy linearly searches exactly each element of input array and compares with its previous element. If its state is different from previous element, then the element uses its own state and duration as the result; if its state is the same as previous element(s), then it merges all durations to be the output. Figure 15 represents the pseudocode and

illustrates the time complexity of the CgroupBy. For an array of size n using CgroupBy, we define T(n) is the computation time and n is the size of the input array. We always try to approximate the worst case to see the rate of growth of very high values of n. In the first part of algorithm, we use C1 to represents the computation time. As we can see, these are all simple statements that have simple operations like arithmetic or logical operations, assignments or comparisons and simple statements executed in constant time. If there are no loops or function calls and there are only simple statements, then the time taken will not be a function of the input size. In the worst case, this set of simple statements will take same constant time as C1. In the second part, BA performs value comparison between every element and their previous adjacent elements. We define that each element comparison will have constant computation cost of C2. For all elements in the array, the cost will be C2 × n. Therefore, T(n) = C1 + C2 × n and the time complexity is O(n) and Figure 16 shows the time and size of array.

For the space complexity, we focus on the extra memory that will be consumed during the BA operation. In BA algorithm, we use extra memory to store the value of adjacent element and use it to compare itself with previous adjacent element. As we can see from the pseudocode code, BA performs the comparison between “current_state” and “next_state”. Once the compression is finished, this extra memory will be cleared automatically. Therefore, the space complexity is O(1).

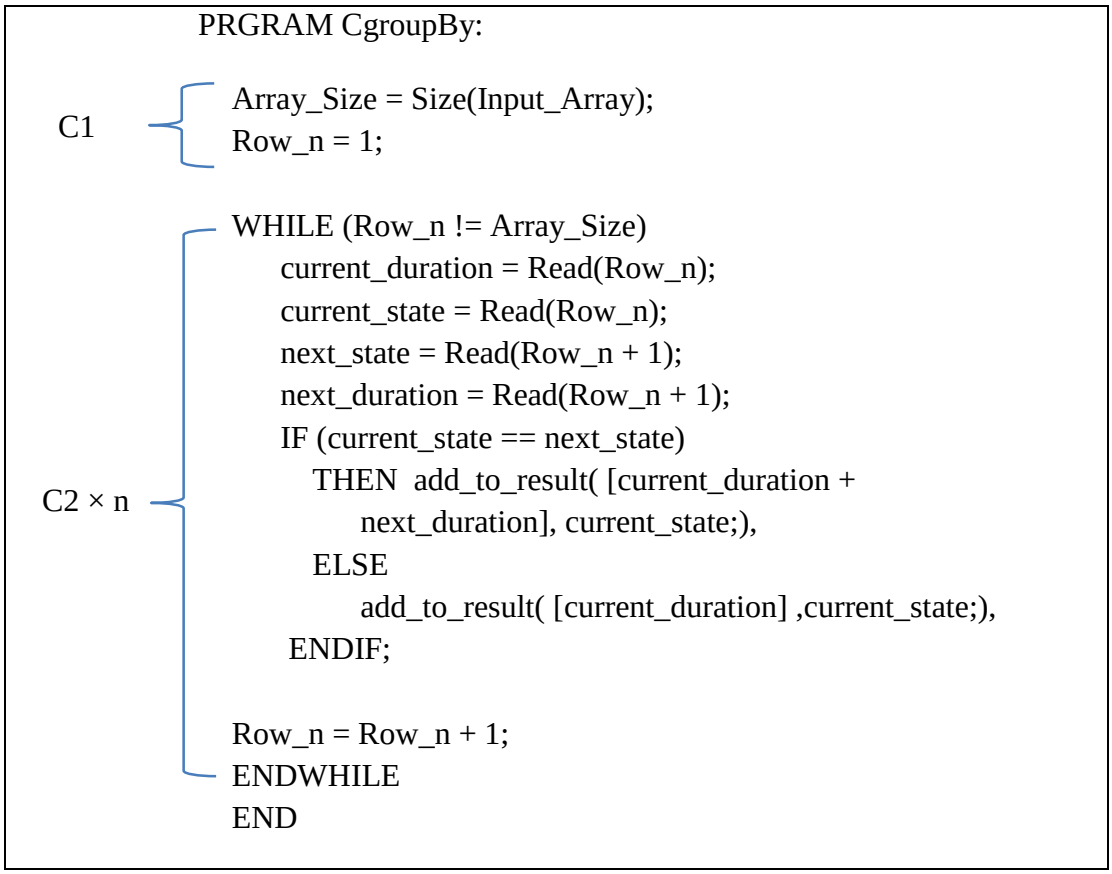


Figure 15. Pseudocode of CgroupBy algorithm

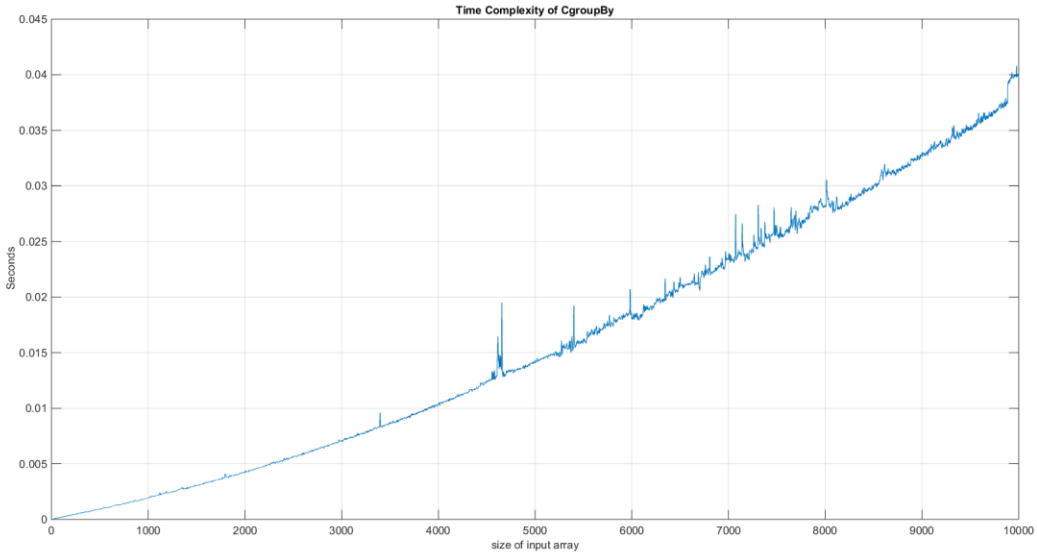


Figure 16: CgroupBy time complexity

Conflict Degree (CD) algorithm: The concept of conflict degree (CD) is to assess the risk of misestimation of values reported in conflicting historical tuples and it is the core of the conflict aware data fusion method, which defines an optimal *CD* threshold value that will minimize the misestimation error (The detailed description is covered in section 6.4. Figure 17 shows the Pseudocode and illustrates the time complexity of CD algorithm. For total size n of tuples, we define T(n) is the computation time and n is the number of conflicting tuples. In order to calculate the CD, we define two conflicting as tuples r1 = (F1,T1,V1) and r2 = (F2,T2,V2). In the worst case, all input historical reports are conflicting with each other. In the first part of CD pseudocode, we use C1 to represents the computation time. Since it only contains simple statements, the computation time will not be a function of the input size. C1 can be represented as computation time of the worst case. In the second part of CD pseudocode, we first define a relative contribution (RC) as

$$RC(r1,r2) = 1 - |V1-V2|/(V1+V2).$$

We then calculate the relative overlap (RO) as

$$RO(t1, t2) = \max(|t1 \text{ o } t2|/|t1+t2|, 0).$$

Note that RO (t1, t2) = 0 means that t1 and t2 do not overlap. Finally, the CD can be derived using RC and RO as

$$CD = RO \times e^{k(1-RC)(RO-1)}$$

We define each pair of conflict reports will have computation cost of C2. To process the conflict aware data fusion method, we need calculate all CD values from all pairwise reports which is C2 × n^2. Therefore, the total computation time T(n) = C1 + C2 × n^2 and the time complexity is O(n^2) . For the space complexity, we focus on the extra memory that will be consumed during the CD operation. In CD algorithm, we use extra memory to store the values of RC, RO and use them to compute the result of CD. As we can see from the pseudocode code, once CD is derived, the values of RC and RO are cleared automatically. Since there is no additional memory required during the calculation, the space complexity is O(1).

In addition, Figure 17 shows the result of time complexity of CD calculation with respect to the number of reports. Since the CD calculation time may be affected by other factors such as event density and the length of reports growing (detail description is provide in section 6.5.1), we illustrate the time complexity based on different scenarios, which are using value = 20 as sparse event and short reports; and value=100 as dense event and long reports.

```
PRGRAM ConflictDegree:

C1          Array_Size = Size(Input_Array);

            FOR i = 1: Array_Size
C2 × n^2        FOR  j = 1: Array_Size

                relative_contribution =  Get_RC(i, j)
                relative_overlap = Get_RO(i, j)
                conflict_degree = Get_CD(i, j)

            END
            END;
```

Figure 17: Pseudocode of CD algorithm

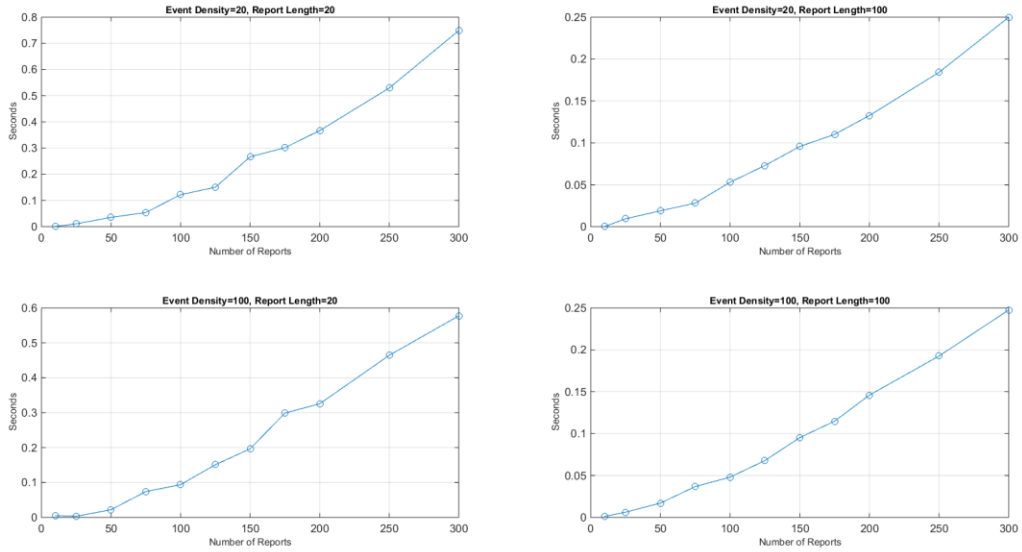


Figure 18: CD calculation time complexity

4 COMPARISON PDW TO MANUAL APPROACH

In previous chapter, we introduced a process data warehousing (PDW) approach to utilize and handle large-scale process data problems. Our PDW is a multistage data process architecture which includes methods and techniques to extract information from raw historical data sets in order to support efficient decision making. In this section, we discuss the advantages and performance of each phase of the PDW approach when compared and analyzed against manual process based on key metrics of time complexity, space complexity, accuracy, and data representation.

Phase I: Data Transformation

In Phase one of PDW, we perform data transformation. The raw input data is transformed into semantically richer and tuple-based data using various data transformation functions. It is process of combining, integrating, or transforming multi-vector data sources, and can facilitate future decision making.

Unlike our PDW, the manual process does not have a predefined standard procedure about how to process and handle the raw data, it often only focuses on certain tasks such as outlier remover and standard statistical analysis. Moreover, PDW also include the functions of merging different data sources, converting data types to a uniform format, grouping similar or duplicated data, and reordering the data based on specific data columns. Those functions are not often seen and performed in manual process and they can be summarized as the following two major tasks in phase I with the key metric evaluation.

1. Since PDW is designed to process and utilize the raw data from the beginning to end of each phase, we design a standard procedure to process data, the steps of which may not always be an option for manual process.

2. PDW always consider time stamp with data element. It transforms raw data into an appropriate data stream format that includes time-stamped multivariable vectors (tuples) and this is often not required in manual process.
3. Key metrics in Phase I.

	PDW	Manual Process
Time Complexity	(High) Advanced functions, more time consuming	**(Low)** **Simple tasks; less time consuming**
Space Complexity	(High) Advanced functions, more memory needed	**(Low)** **Simple tasks, less time consuming**
Accuracy	**(High)** Several automatic predefined functions	(Low) only standard outlier remover, human error
Data Representation	**(Improve)** Change structure of data to provide better representation	(None)

Phase II: Data Adaptation

In the Phase two of PDW, we perform data adaptation. In this phase, PDW system decreases the granularity of the raw data by grouping the data items into a number of meaningful application-dependent states. In the following, we list some advantages of using PDW for data processing in this phase with the key metric evaluation.

1. Unlike PDW, the manual data process may be still using raw data. As we know, having detailed raw values could be useful but it will be more useful if we can know the meaningful representations of these values. Therefore, we perform data adaptation to categorize data into meaningful states to let the users understand what is normal and what is abnormal.

2. The manual process may also include certain steps that group data items into a number of application-dependent states. However, PDW provides a more sophisticated solution which includes automatically using data clustering techniques (such as K-means, fuzzy C-means, and hierarchical clustering), or using embedded mapping rules that are specified by an application expert.
3. Grouping data into state not only lets user better understand the data, but also reduces the size of the data as well as the complexity for data processing.
4. Key metrics in Phase II.

	PDW	Manual Process
Time Complexity	(**Low**) **Sophisticated algorithms require less time**	(High) Require more processing time
Space Complexity	(High) Advanced grouping functions require higher memory	**(Low)** **Simple grouping function require less memory**
Accuracy	**(High)** **Properly analyze the input data before grouping or using domain expert's option**	(Low) Does not analyze input data properly
Data Representation	**(Better)** **Assign meaningful states to the data**	(Low) Usually does not provide good data representation

Phase III: Event Extraction

In the Phase three of PDW, we perform event extraction, which is the procedure that further aggregates the tuples with identical states into events and associated durations. As a result, in this phase, PDW groups same states within consecutive rows that sum up their durations. Again, the

above step not required and not often seen in manual process. Some performance advantages using PDW in this phase when compared to the manual process are:

1. Collecting time-related serial data often requires a great deal of disk space and consumes many processor and memory resources. This step removes duplicated states that occur in consecutive tuples. So it reduces the size of the data as well as the complexity for data processing.
2. When PDW further aggregates data by merging the tuples with the same states into "events", where all states remain unchanged for a given period of time, it helps us to discover the data dynamic (state changes).

3. Key metrics in Phase III.

	PDW	Manual Process
Time Complexity	**(Low)** **Data size reduce; require less time to process input data**	(High) Larger data size; need more time to process input data
Space Complexity	**(Low)** **Data size reduce; require less memory**	(High) Larger data size; need more memory
Accuracy	**(High)** **Include advanced methods to process the time serial data.**	(Low) Usually does not include an appropriate method to process the time stamps
Data Representation	**(Better)** **Assign meaningful states to the data**	(Low) Usually does not provide good data representation

Phase IV: Event Type Generation

In the Phase four of PDW, we perform the task of event type generation. It is process of grouping event data with its duration. In this phase, our PDW system performs a severity assessment from the event data from previous phase. In this stage, the PDW system will use a number of data machine learning algorithms such as naïve Bayesian approach, decision tree approach and artificial neural networks. In the following, we list some advantages of using PDW for data processing and severity assessment against the manual process with key metrics.

1. Unlike PDW which uses event data, manual process uses raw data which often has much larger data and therefore, may increase the complexity for data processing.

2. Moreover, manual process using raw data may not be appropriate for analyzing event severity. The format of event data in PDW is designed and optimized for PDW's machine learning algorithms and it can evaluate event severity.
3. Key metrics in Phase IV

	PDW	Manual Process
Time Complexity	**(Low)** **Data size reduce; require less time to process input data**	(High) Larger data size; need more time to process input data
Space Complexity	**(Low)** **Data size reduce; require less memory**	(High) Larger data size; need more memory
Accuracy	**(High)** **Include advance methods to evaluate event severity**	(Low) Using raw data may not be appropriate for analyzing event severity
Data Representation	**(Better)** **Transfer raw data to meaningful event type**	(Low) Still use raw data.

Phase V: Decision Making

In phase five, PDW performs the task of decision making and, it uses sequences of event types from previous phase. Moreover, PDW system performs data analysis in order to recognize notable patterns in the process environment. The system applies embedded decision making algorithms that continuously monitor the trends in sequences of event types. For example, an early warning signal (flag) is raised whenever abnormal patterns are identified. To sum up, using PDW has the following advantages compared to manual process

1. PDW designs its own algorithms to facilitate trends investigation in the form of event severities while manual process does not have dedicated trend monitoring methods.
2. PDW has the ability to preprocess the sparse and redundant data in its earlier phases, therefore, it has better performance in decision making.
3. Like previous phase, PDW further reduces the size of the data since it uses its preprocessed data from previous phase, and this will improve decision making result as well as the complexity for data processing.
4. Key metrics in Phase V.

	PDW	Manual Process
Time Complexity	**(Low)** **Data size reduce; require less time to process input data**	(High) Still use raw data; larger data size; need more time to process input data
Space Complexity	**(Low)** **Data size reduce; require less memory**	(High) Still use raw data; larger data size; need more time to process input data
Accuracy	**(High)** **Include methods to preprocess sparse and redundant data;** **Include algorithms to facilitate trends investigation**	(Low) Using raw data directly; may not be an appropriate for analyzing complex abnormal signal
Data Representation	**(Better)** **Define early warning signal pattern to discover abnormal trend**	(Low) Still use raw data.

5 CASE STUDY I: HISTORICAL MEDICAL DATA

We performed an experimental study of our PDW architecture using data obtained from the Children's Hospital of Pittsburgh, a large academic pediatric hospital for patients with a chief complaint that indicated a respiratory problem. Our reasoning for starting with patients with a respiratory complaint was that children with respiratory complaints represent a significant caseload for an emergency department and a hospital. For each patient, the data obtained included the chief complaint, the patient's vital signs, and the clinical location of the patient. The data was obtained by initiating a query from a clinical data warehouse used by the institution for a variety of clinical, administrative, and quality applications. In this institution, a chief complaint—called the "Reason for Visit" (RFV) is assigned to each patient by a triage nurse, based on the presenting issues, as described by the child or his or her family. The choices for a RFV are limited to approximately 100 choices, although there are some children whose RFV were chosen from outside this list. The dataset we selected includes patients who were admitted into the emergency department and who had one of the following chief complaints: asthma, bronchiolitis, cold, cough, dyspnea, grunting, respiratory distress, shortness of breath, stridor, URI, wheeze, or wheezing; in short, from the 100 possible choices, any RFV that indicated or implied a respiratory complaint was included in the dataset. For each patient, we obtained the complete hospital record for that visit.

Table 8 shows a sample of the original data, which includes a set of vital signs which include temperature, systolic blood pressure (SBp), diastolic blood pressure (DBp) , heart rate, respiratory rate, oxygen saturation, and the clinical location of the patient. Note that each patient has a clinical event/vital sign and its location at a particular timestamp. A patient's age is recorded in days, and the time stamp is negative when the patient is in the emergency department. As a result, the first time stamp is always negative, and the time stamps will increase as new vital signs are recorded.

Table 8: Data obtained from CHP data warehouse

Study Number	Admit Age (Days)	Clinical Event Minutes from Admission	Clinical Event Code	Clinical Event	Clinical Event Result	Person Location – Nurse Unit (From)
22073048	1,819	-352	534253	Temperature	36.2	Emergency_Dept
22073048	1,819	-352	534272	Respiratory_Rate	68	Emergency_Dept
22073048	1,819	-352	572815	Systolic_BP	133	Emergency_Dept
22073048	1,819	-352	618092	Heart Rate	143	Emergency_Dept
22073048	1,819	-352	666768	Diastolic_BP	70	Emergency_Dept
22073048	1,819	-352	72349055	SpO2_Bedside_Monitor	95	Emergency_Dept
22073048	1,819	-321	534272	Respiratory_Rate	72	Emergency_Dept
22073048	1,819	-321	572815	Systolic_BP	80	Emergency_Dept
22073048	1,819	-321	618092	Heart Rate	154	Emergency_Dept
22073048	1,819	-321	72349055	SpO2_Bedside_Monitor	96	Emergency_Dept

The training data sample contained 146,998 rows of data and included children that were admitted to the hospital from 8/1/2008 to 12/31/2008. This data set had a total of 726 unique patients, 101 of whom were admitted to the ICU. A second data set was obtained to use to test and validate the PDW. This dataset used the same inclusion criteria for patients admitted from 1/1/2009 to 2/28/2009 and contained 419,335 rows of data. This data set had a total of 454 unique patients, 34 of whom were admitted to the ICU. The left of Figure 19 shows the number of records for each symptom group with age group, while the right of Figure 19 uses the number of patients as the indicator. Detailed statistics are provided in Appendix 1: Training Data Characteristics.

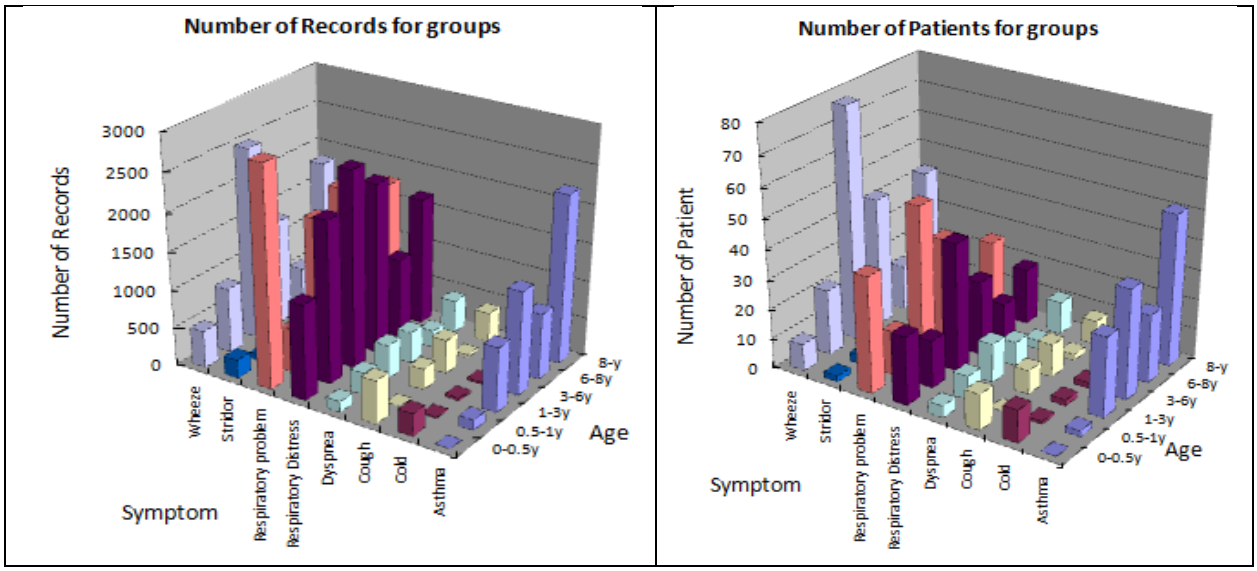


Figure 19: Statistics of patient records

5.1 DATA PRE-PROCESSING

While the raw data from the EMR is useful, it is not sufficient in supporting efficient decision making processes. Several steps are needed to turn the raw data stream into useful information. The pre-processing steps inject "intelligence" into the data in order to support automated reasoning and facilitate physician decision making. These steps mirror what physicians naturally do for every patient that they evaluate. Physicians look at the current data (numerical, physical, and contextual) and evaluate the current and the recent past states of the patient to help make decisions which will support or change the predicted future state of the patient.

The pre-processing in (Phase I-III) includes the following steps: transforming the raw data into a usable data stream, adapting the data into states, and finally, extracting events from the states. Each of these steps helps to prepare the data to support decision making and modeling. The data stream from the patient dataset described above is used to demonstrate these pre-processing techniques.

5.1.1 Phase I: Vital Data Transformation

In the original data from the clinical system, each clinical event (such as each vital sign value) exists as a separate row in the original data set. For example, even if a temperature, heart rate and blood pressure were recorded into the system at the same time, each value exists as a separate row. The format of the raw data needs to be converted into a data stream where every data point for a specific timestamp is combined into a single row. This conversion to a better-formatted data stream allows the data stream management system to further analyze the data. As shown in Figure 20, the first step in the data processing is to take the original data and convert it to a data stream, which can be used for further analysis. The original raw data stream shows several values for vital sign parameters (Temp, HR, RR, SBp, DBp, SpO_2, and Oxygen Flow Rate), some of which are part of time stamp 1 or part of time stamp 2. If the values have the same timestamp, they are pooled and placed into a single row.

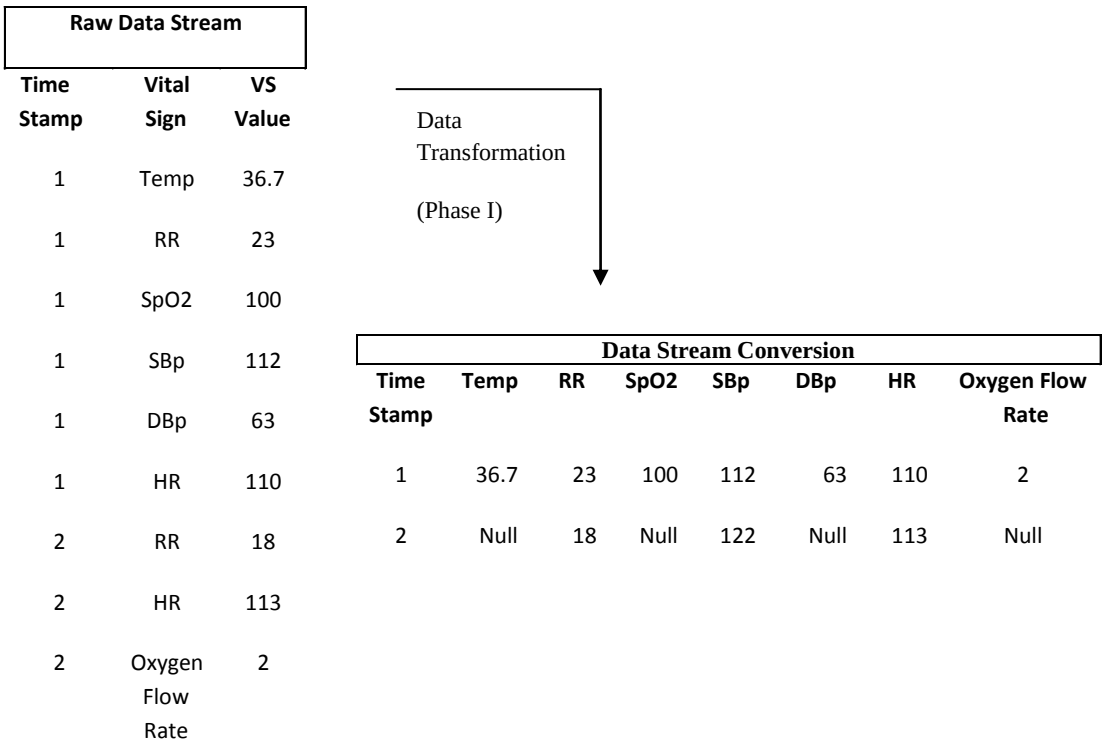


Figure 20: Illustration of data transformation (Phase I)

5.1.2 Phase II: Sparse Data Adaptation

For each vital sign, there are a range of values which are all considered clinically similar. For example, for a patient between 1–3 years old, a systolic blood pressure would be considered normal if it were between 72 and 110. The raw numbers for each value has been mapped to a group that represents a range of values. These mapped values are now considered to be "states" of interest. For example, consider respiratory rates: a patient's respiratory rate can be categorized into groups of values (critical low, low, normal, high and critical high), based on the patient's age. For example, a patient who is 30 days old with a respiratory rate of 55 would be assigned to the "normal" group, but a patient who is 300 days old with the same respiratory rate would be assigned to the "Critical High" group (Table 9).

Table 9: Mapping of respiratory rate values to specific states by age group.

Respiratory Rate					
Age	RR_Critical_Low	RR_low	RR_Normal	RR_High	RR_Critical_High
0 (0-180 days)	0-20	21-29	30-60	61-70	71-
0.5-1 (181-365 days)	0-15	16-23	24-40	41-50	51-

The mapping into states can be done for each vital sign. See APPENDIX 2: VITAL STATE MAPPINGS for a full listing for mapping rules. The mapping rules for the vital signs were provided by a domain expert. This is similar to other systems that have grouped vital sign parameters in similar ways [81]. Figure 21 illustrates the mapping of the data stream into states for a child that is 1–3 years old. For example, a temperature of 36.7 shown at time stamp 1 is mapped to the state of "T_Normal," while a respiratory rate of 23 is mapped to the state "RR_low." It is important to note that at this stage, values which have not been updated (null values) from a prior time stamp are assumed to be in the same state as the prior row. This

assumption can be reasonably made, especially in the context of the emergency department. In general, when vital signs are updated, they are updated either because of general protocol/surveillance (such as an hourly update while the patient is in the emergency department) or due to a desire to document a significant change in status (such as any temperature improvement after anti-pyretic therapy). Usually, if the motivation for updating is general protocol, all vital signs will be updated; if it is to document a specific change, it is likely that all other vital signs are the same. A potential downside to this is that if a state is abnormal and no new data is placed in the system, the system may falsely assume that the patient is still in an abnormal state. While this may or may not still be the case, the fact that the data has not been updated is a potentially useful piece of information in and of itself—namely, that no one has re-examined at an abnormal variable.

Transformed Data Stream

Time Stamp	Temp	RR	SpO2	SBp	DBp	HR	Oxygen Flow Rate
1	36.7	23	100	112	63	110	2
2	Null	18	Null	122	Null	113	Null

Data Adaptation

(Phase II)

Data Stream converted to state based on mapping (1- 3 year old)

Time Stamp	Temp	RR	SpO2	SBp	DBp	HR	Oxygen Flow Rate
1	T_Normal	RR_Low	SpO2_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low
2	Null	RR_Low	Null	SBp_High	Null	HR_normal	Null

Data Stream converted and prior states assumed for next row if variable is not updated (shaded cells)

Time Stamp	Temp	RR	SpO2	SBp	DBp	HR	Oxygen Flow Rate
1	T_Normal	RR_Low	SpO2_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low
2	T_Normal	RR_Low	Sp02_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low

Figure 21: Data stream adaptation to states

In addition to the mapping of specific vital signs to a particular state, it is possible to observe the consequences of these states. As a patient's vital signs change, we can observe that these vital signs are taken in different locations. These observations are later used as part of the modeling.

5.1.3 Phase III: Vital Data Event Extraction

Consider a patient whose respiratory rate has been measured over a period of time. At one time point, the respiratory rate is normal; at a second time point, the respiratory rate is still normal; and at a third time point, the respiratory rate is high. These individual points in time are extracted

into events where for a certain period of time, the respiratory rate was normal, and then for another period of time, the respiratory rate was high. In this informal example, two events are extracted: one event of normal respiratory rate covering the period of time from the beginning of the first time point to the end of the second time point, and a second event that occurs during the third time period.

As we discussed in Section 5.1.2, an event can be defined as a function of each state (S) and the clinical location where the states were observed (O) for a given duration (D). As an example, event 1 (E_1) contains each parameter in a given state in the emergency department for 65 minutes. When a parameter changes, a new event is created (E_2). In this case, the temperature changed its state. The events E_1 and E_2 can be represented by the following vectors:

E_1 = < S_{Temp} = High, S_{SpO2}=Normal, S_{HR}=High, S_{RR}=Low, S_{SBp}=Normal,S_{DBp}=low, S_{O2Rate}= Normal, O=ED, D=65min>

E_2 = < S_{Temp} = Normal, S_{SpO2}=Normal, S_{HR}=High, S_{RR}=Low, S_{SBp}=Normal,S_{DBp}=low, S_{O2Rate}= Normal,, O=ED, D=20min>

The events are extracted from the data by tracking the change in states, as detailed above. The duration of time of a given event is the time from when the event was initiated until the time when the next event was initiated. If no new information about a specific vital sign is part of the new event, the previous state is carried forward to the new event. Figure 22 shows the raw vital sign numbers that have been converted to states by using the mapping rules described earlier. If there were no new values for a particular timestamp (such as the null values noted in Figure 21), the values are assumed to be the same state as the prior row. The highlighted cells in Figure 22 note a change in state from the prior row. The duration of event 1 starts at timestamp 1. The states at timestamp 1 are compared with the states at timestamp 2. If it can be concluded that there are no changes in states or in location, then event 1 will continue. The third row does have differences in states, and as a result, a new event starts.

The duration of event 1 is calculated by subtracting timestamp 3 from timestamp 1. The absolute value of the difference is used, since some of the timestamps are negative. In the raw data stream, the timestamp is linked to the time of admission to a medical floor or ICU, while any negative timestamps are associated with vital signs taken in the emergency department. This calculation gives the duration between each event.

The event table shown in Figure 22 is the final output of the pre-processing steps. From this event table, various techniques can be applied to analyze events, and to build clinical decision support tools and an early warning system.

Data Stream converted to states

Time Stamp	Temp	RR	SpO2	SBp	DBp	HR	Oxygen Flow Rate	Location
10	T_Normal	RR_Low	SpO2_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low	ICU
20	T_Normal	RR_Low	SpO2_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low	ICU
30	T_Normal	RR_Low	SpO2_low	SBp_High	DBp_low	HR_normal	O2_Fr_Low	ICU

Event Extraction (Phase III)

Event Table

Event	Temp	RR	SpO2	SBp	DBp	HR	Oxygen Flow Rate	Location	Duration
1	T_Normal	RR_Low	SpO2_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low	ICU	|30-10|
2	T_Normal	RR_Low	SpO2_low	SBp_High	DBp_low	HR_normal	O2_Fr_Low	ICU	

Figure 22: Event extraction: shaded cells indicate a change in state

5.1.4 Event Data Warehouse

The final event table is considered to be a data warehouse, and can be used in clinical decision support tools. A single vital sign can be projected out as a separate view and analyzed. As with the event generation logic, an individual vital sign projection can be used for individual parameter analysis. Individual vital sign events are created either when the vital state changes or when a patient's location changes (Figure 23).

Patient ID	Event	Temp	RR	SpO2	SBp	DBp	HR	Oxygen Flow Rate	Duration
22073048	1	T_Normal	RR_Low	SpO2_Normal	SBp_Critical high	DBp_normal	HR_normal	O2_Fr_Low	31
22073048	2	T_Normal	RR_Low	SpO2_Low	SBp_normal	DBp_low	HR_normal	O2_Fr_Low	330
22073048	3	T_Normal	RR_Low	SpO2_Low	SBp_High	DBp_normal	HR_High	O2_Fr_Low	120

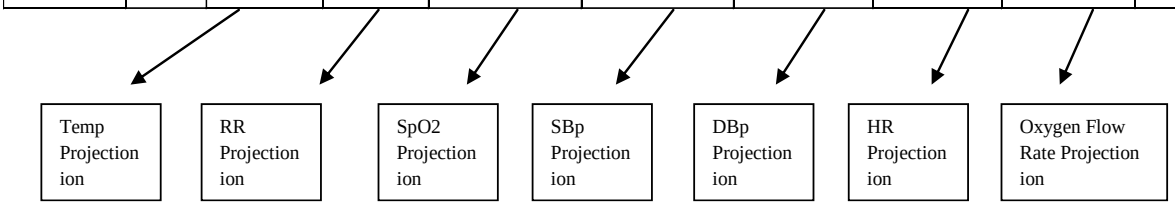


Figure 23: Individual vital sign projection

Figure 24 shows an example of the individual vital sign projection and individual event generation with corresponding CgroupBy queries. In the event table (Figure 23), the SpO_2 is in the normal state for duration of 31 minutes, and for the next two events, the SpO_2 is low. For the individual vital sign projection, the low states can be combined, which ultimately creates two events, with a duration of 31 and of 450 minutes, respectively (Figure 24). Various techniques used to analyze these events may or may not use this projection.

Query:

Select Patient_ID, SpO_2 as [SpO_2_Event], Duration
From Figure_14
CgroupBy SpO_2

SpO_2		
Patient_ID	**SpO_2_Event**	**Duration**
22073048	SpO2_Normal	31
22073048	SpO2_Low	330+120=450

Query:

Select Patient_ID, SBp as [SBp_Event], Duration
From Figure_14
CgroupBy SBp

SBp		
Patient ID	**SBp_Event**	**Duration**
22073048	SBp_Citical high	31
22073048	SBp_Normal	330
22073048	SBp_High	120

Query:

Select Patient_ID, Temp as [Temp_Event], Duration
From Figure_14
CgroupBy Temperature

Temp		
Patient ID	**Temp_Event**	**Duration**
22073048	T_Normal	31+330+120=481

Figure 24: Event generation for SpO_2, SBp, and Temperature

5.2 PHASE IV: VITAL DATA EVENT TYPE GENERATION

Now that an event row is generated that details the state of each vital sign during a period of time, the next step is to assess the severity of these events. The assumption is that there may be combinations of states in one event that might be considered to be more worrisome than others. Furthermore, there may not be a single event at one time that may be worrisome. Rather, there may be a pattern of events over time that may warrant further attention. In this medical case study, we considered the event severity assessment by using multiple approaches, including a simple naïve Bayesian approach, as well as a more sophisticated decision tree approach and artificial neural networks.

Decision trees are flexible, easy to understand, and can be visually represented as a decision situation. In our case study, we try to provide categorical values, like red, green, or yellow, to indicate the severity of an event. A decision tree provides us with an illustration structure to let us know what input variables (vital signs) are more important than others. As a result, our domain experts can easily verify the constructed tree result to see if it aligns with a physician's domain knowledge. The naïve Bayesian classifier is a probability-oriented approach that is fundamentally different from a decision tree, and we can use it to explore an event's severity from a different direction. Moreover, this naïve Bayesian classifier often does surprisingly well and is widely used in medical domains [82-84], because it often outperforms more sophisticated classification methods. Studies [85] show that the naïve Bayesian approach is well suited for medical applications and performs well for most examined medical problems. We also include neural network as another alternate approach of computing that reflects physicians' diagnoses. Several studies [86] show that using artificial neural networks can introduce effective parameters for improving the performance and application of machine learning and pattern recognition techniques to facilitate medical processes. The purpose of using neural networks in this case study is to take advantage of its learning capability and capacity to store experiential knowledge from data learning on vital signs, like in the human brain. As a result, we can use vital signs data to provide diagnoses by deducing certain symptoms, or formulate a treatment based on more or fewer specified observations and knowledge [87]. We will cover the process of identifying the patterns over event sequences by applying these algorithms in Section 5.3.

In the architecture (Figure 25), the assessment of the event severity is categorized as a "combined event type." Depending upon the machine learning approach to the event severity assessment, each vital sign may be analyzed separately or in groups. The combined event type is the final severity assessment for that event. With this data set, two event types are considered: ICU (a severe event) and floor (all other events). The "ICU" combined event type will indicate that for a given event, the pattern of vital sign states has been classified as similar to patients who were admitted to the ICU from the emergency department.

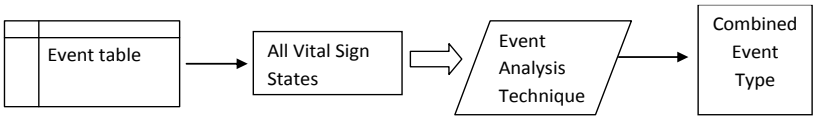


Figure 25: Generalized approach to event analysis.

5.2.1 Event Type Generation Using the Naïve Bayesian Classifier

The approach to the event classification with the naïve Bayesian classification (NBC) represents a simple approach to assess the severity of the event (Figure 26).

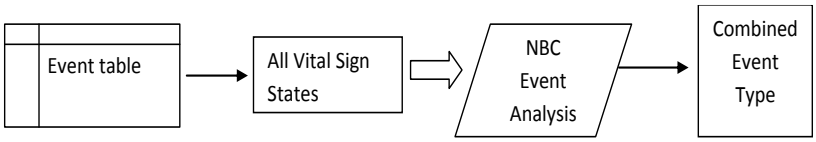


Figure 26: NBC approach to event analysis.

To train the classifier, the training dataset is used as described above. 101 randomly selected patients who were admitted to the floor were matched with 101 patients admitted to the ICU. For this simple approach to assessing event severity, only the initial event row with all seven vital

sign states were used. The independent variable was the location where the child was admitted. To illustrate, Table 10 shows a row of data that would be used for the NBC. The NBC event classifier performance is discussed in subsequent sections. Although it is only trained on the first entry, the NBC can be used to classify the event severity for any event row.

Table 10: Data used for NBC

Time Stamp	*Temp*	*RR*	*SpO2*	*SBp*	*DBp*	*HR*	*Oxygen Flow Rate*	*End Location*
1	T_Normal	RR_Low	SpO2_Normal	SBp_High	DBp_normal	HR_normal	O2_Fr_Low	ICU

5.2.2 Event Type Generation Using Decision Tree Classifiers

Another way to analyze the event severity is by using a set of decision tree classifiers (DTCs). For this approach, a decision tree classifier is used for each individual vital sign. After each vital sign is assessed, it is combined to give a single combined event type severity assessment (Figure 27).

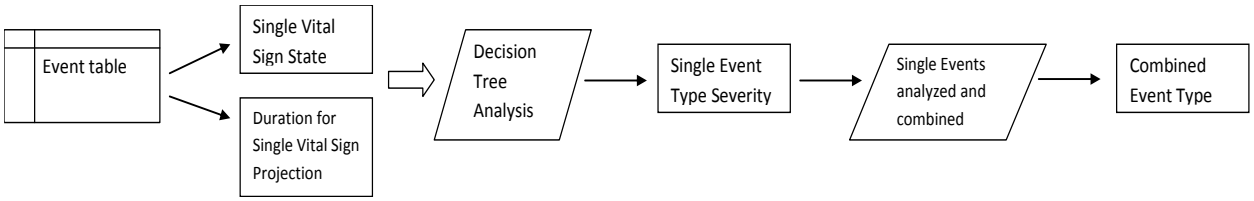


Figure 27: Decision tree approach to event analysis

Like the NBC approach, the event rows are ultimately classified as either an "ICU" (more severe) or "floor" (all others) combined event type. Unlike the NBC, there is an intermediate step

where the individual vital signs are analyzed separately each with their own unique decision tree. The steps of how the individual vital sign decision tree classifiers are generated are discussed below.

The training dataset used to generate the decision tree classifiers included data from August 2008 to December 2008. As with the NBC, the same 101 patients who were admitted to the ICU were paired with 101 randomly selected patients from the remainder who were admitted to the floor. Quinlan's classifier known as C4.5 [25] is used as the DTC algorithm.

Independent Variable Generation:

The classification of the single event type is based on two independent variables. The variables include the duration of the event, as well as the current state and the transition between the current state and the prior state.

In order to use the event data for the decision tree classification, a final data pre-processing step is performed to transform the vital sign states into colors. Each state is mapped to a series of colors (green for normal, yellow for slightly abnormal, and red for critically abnormal). This mapping can be done on the event table or the individual vital sign projections. The color mapping is done without regard to whether the data was a higher or lower than normal value. There are two reasons for this procedure. First, we would like to have fewer states and to reduce the complexity in decision tree analysis. Second, we noticed that the SpO_2 variable does not have a high or critical high state, and that the oxygen flow rate variable does not have low and critical low states. In order to build a standard and simplified understanding for all vital signs, we apply this color transformation procedure. Therefore, when the individual event is generated, we can transfer our states (critical high, high, normal, low, and critical low) into three color schemes. Table 11 shows a detailed description of this transformation of different vital signs.

Table 11: Color mapping for states

For Temperature, HR, RR, SBp, and DBp

State	Color
Critical High	Red (R)
High	Yellow (Y)
Normal	Green (G)
Low	Yellow (Y)
Critical Low	Red (R)

For SpO_2

State	Color
Critical Low	Red (R)
Low	Yellow (Y)
Low Normal	Yellow (Y)
Normal	Green (G)

For Oxygen Flow Rate

State	Color
Critical High	Red (R)
High	Yellow (Y)
High Normal	Yellow (Y)
Normal	Green (G)

Once the states are mapped to a color, the event row can be classified. In this decision tree classifier, the events are classified as being either an "ICU" or "Floor" event type. More details about the generation and performance of the decision tree classifier can be found in later sections. Consider the event table, as shown in Figure 28. The temperature state is normal within duration D1, and as a result, it is mapped to "Green." The DT classifier takes those two pieces of information to classify that cell as a "Floor" single event type. The details of the decision tree classifier will be discussed later in this section. This single event type determination occurs for each individual vital sign, as shown in Figure 28.

vent	Event Table Mapped to color states							Duration
	Temp	RR	SpO2	SBp	DBp	HR	O_2 Flow Rate	
	Green	Yellow	Green	Yellow	Green	Green	Yellow	D1
	Green	Yellow	Yellow	Yellow	Yellow	Green	Yellow	D2

Event	Single Event Type						
	Temp	RR	SpO2	SBp	DBp	HR	O_2 Flow Rate
1	Floor	ICU	Floor	ICU	ICU	Floor	Floor
2	Floor	ICU	ICU	ICU	ICU	Floor	ICU

Figure 28: Single event type determination, using a decision tree for each vital sign

Once the single event types are determined, they are aggregated to determine the event severity for a specific event row. Equal weighting of each vital sign is used and determines the combined event type, based on the majority of the single event determinations[1]. Consider the example shown in Table 12. The combined event type at event 1 is "floor" because, in total, more single event types were determined to be "floor" rather than "ICU" event types. In contrast, event 2 is an ICU combined event type because it has a larger total of ICU event types.

Table 12: Combined event types

Event	Single Event Type							Combined Event Type
	Temp	RR	SpO2	SBp	DBp	HR	O_2 Flow Rate	
1	Floor	ICU	Floor	ICU	ICU	Floor	Floor	Floor
2	Floor	ICU	ICU	ICU	ICU	Floor	ICU	ICU

Duration

The duration of an event is an important factor in the event severity analysis. As previously mentioned, there may be times when the information provided from the EMR is lacking. This lack of information may or may not be important. For example, when working an overnight shift on a medical floor, nurses may forgo taking vital signs on a medically stable child. In this case, the lack of information is not important. Our assumption that the last recorded state is the same for the current event, unless new information is provided, holds true in this situation. In the emergency department setting, the lack of new information can mean other things as well. If the data has not been updated in the EMR for a while, it may be due to either an extremely severe medical situation where there is no time to document the current state of the patient, or it may be

[1] In future iterations, this is an area of further exploration. It is likely that in different clinical contexts, some vital signs may be more clinically relevant than others (for example, in asthmatics RR, SpO_2 may be more relevant than temperature) and therefore should get greater weight as the combined event type is generated.

due to the vital signs simply not being taken. In the latter situation, a mildly ill patient may "slip through the cracks," and a medically fragile situation can worsen. The implementation of the duration range is specifically designed to address this kind of scenario. A set of mildly abnormal vital signs which have not been corrected over a long period of time may become the last straw that tips a patient into a medical crisis. As time moves forward without any new information, we can create new events using the duration ranges described below. This is fed into the decision tree as part of the training of the decision tree, and is ultimately considered when the event severity is determined.

1. Duration Ranges

Like the original vital sign data, which was grouped based on clinically relevant ranges (such as normal HR from 80-110), the continuous data of the duration is grouped into a series of ranges for the purposes of classification (Table 13)[2]. The event durations are categorized into the duration ranges, as shown in Table 13. Later in this work, we implement a time threshold and unfold the events, even if no new information is provided.

[2] Note that the first duration range is for all durations less than two hours. In clinical reality, a two hour timeframe is a reasonable period of time to wait before needing new information about the state of a patient.

Table 13: Detail of duration ranges

Duration (Hours)	Duration Range
< 2hr	1
2~3 hr	2
3~4 hr	3
4~5 hr	4
5~6 hr	5
6~7 hr	6
7~8 hr	7
8~9 hr	8
> 9hr	9

2. Color Transitions

The other independent variable used in the decision tree classification process is the current state and the transition from one state to the next. Those states and state transitions are encoded for the classification analysis, as shown in Table 14.

Table 14: Detail of color transition

Current Color	Previous Color	Color Transition
G	R	1
Y	R	2
R	R	3
G	Y	4
Y	Y	5
R	Y	6
G	G	7
Y	G	8
R	G	9

Example: Data for decision tree

Consider the projection of a patient's temperature state from the event table. Table 15 illustrates the states that are color mapped, along with the event duration.

Table 15: Example of a patient's temperature state

Current Color State (Temp)	Duration in Current Color	Previous Color State (Temp)
G	5.5 hour	Y
Y	1 hour	R
R	1 hour	Y

Based on the duration range mapping (Table 13) and color transition mapping (Table 14), the data for the temperature decision tree classifier will look like the data shown in Table 16.

Table 16: Data for decision tree

Event	Color transition	Duration range	Current Location
1	4	5	Emergency
2	2	1	Emergency
3	6	1	ICU

To build the decision classifier for each vital sign, the training dataset was used for the period of time before the patient transitions out of the emergency department. The dependent variable upon which the classifier is built is the location of the child after he or she leaves the emergency department. Note that in this step, some data cleaning is performed to help build a better classifier. There are some instances in the data where the patients who move to the ICU and the floor have color transitions which appear counter-intuitive; specifically where the color transition is 7 (green to green) and the child goes to the ICU, or 3 (red to red) and the child goes to the floor. As a result, before generating the decision tree classifier, some rows that would decrease the accuracy of the classifier are eliminated in the training data. There are situations where a transfer to the floor or ICU may happen even in the face of what seems to be a counter-intuitive situation. For example, the color transition may be 3 (red to red) for temperature and yet, the patient is able to go to the floor. Also, there may be situations where an individual vital sign may be in color transition 7 (green to green), but where another vital sign is prompting the patient's move to the ICU. These incongruities have been removed in training the decision tree classifier to help increase the accuracy of the classifier.

Consider a patient who was sent from the emergency department to the floor (Table 17): the data with color transition 3 (red to red) will be removed from the training data. In this example, the only data that will be considered to help train the decision tree will be the first row of data.

Table 17: Illustration of training data for decision tree with noise removed

Color transition	Duration range	Current Location
4	5	Emergency
~~3~~	~~1~~	~~Emergency~~
6	1	Floor

.

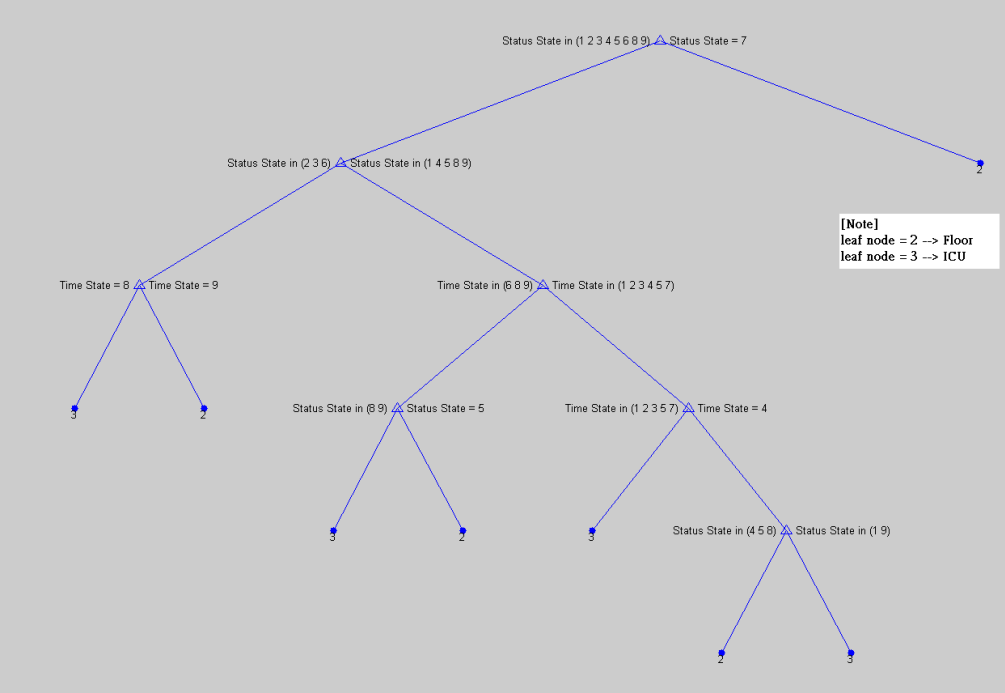


Figure 29: Decision tree for temperature

Figure 29 is the result of the decision tree classifier for temperature. This decision tree will return the individual event type, based on the color transition and duration range. The same procedure is performed for each of the seven considered vital signs.

5.2.3 Event Type Generation Using Neural Network

A neural network (NN) is another way to analyze these event patterns. A feedforward neural network with a backpropagation algorithm is used as the default NN in this study. A backpropagation algorithm is a supervised learning method that is most useful and common for feedforward neural networks. Two necessary materials for such an NN are the training set and target. The training set is the 7 vital signs in each event rows, which are used to train the NN to recognize the pattern, and the target is the correct transfer location for each event rows. For patients who move from the ER to the ICU, its target set as ICU; and when a patient moves from the ER to the floor, its target is set as floor.

As NBC and DT, the event rows are simply classified as either ICU or floor for the combined event type. Unlike NBC and DT, NN takes all 101 ICU patients and 621 Floor patients' data from August 2008 to December 2008 as training materials. The data has been color transformed into Red, Yellow and Green by utilizing the color transformation method adopts from color mapping for DT.

Another difference is that there are 3 different kinds of NN: NN for Group ALL, NN for Group 1 and NN for Group 2. They all have corresponding training dataset group. Group All includes 7 vital signs and duration time. Group 1 includes 3 vital signs (HR, SBp, DBp) and duration time. Group2 includes the other 3 vital signs (RR, SpO_2, O_2 Flow Rate) and duration time. Figure 30 shows the NN procedure.

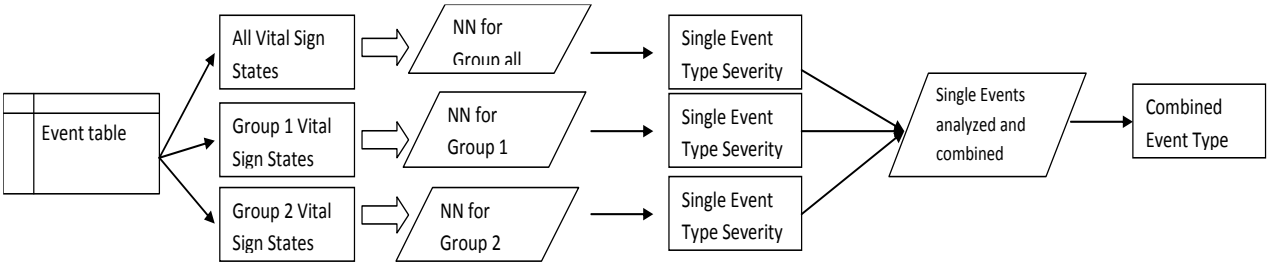
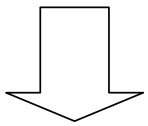


Figure 30: NN procedure

Figure 31 illustrates the NN vital grouping. The main reason to divide vital signs into two sub-groups is because physicians usually observe those vital signs in groups when they check a patient's clinical information, according to the relation between them. For example, a physician also checks SBp and DBp states when he or she checks the HR state, just as when he or she examines the RR, SpO_2 and O_2FlowRate.

Original Events

Events	*Temp*	*RR*	*SpO2*	*SBp*	*DBp*	*HR*	*Oxygen Flow Rate*	*Durations*	*End Location*
1	Normal	Low	Normal	High	normal	normal	Critical High	D1	ICU
2	Low	Low	Low	Normal	Low	Low	High	D2	ICU



Group All

Events	*Temp*	*RR*	*SpO2*	*SBp*	*DBp*	*HR*	*Oxygen Flow Rate*	*Durations*	*End Location*
1	Green	Yellow	Green	Yellow	Green	Green	Red	D1	ICU
2	Yellow	Yellow	Yellow	Green	Yellow	Yellow	Yellow	D2	ICU

Group 1

Events	*SBp*	*DBp*	*HR*	*Durations*	*End Location*
1	Yellow	Green	Green	D1	ICU
2	Green	Yellow	Yellow	D2	ICU

Group 2

Events	*RR*	*SpO2*	*Oxygen Flow Rate*	*Durations*	*End Location*
1	Yellow	Green	Red	D1	ICU
2	Yellow	Yellow	Yellow	D2	ICU

Figure 31: Example of NN vital groups

5.3 PHASE V: DECISION MAKING: THE EARLY WARNING SYSTEM

Section 5.2 illustrated the method for taking raw clinical data and pre-processing it for use by various classifiers. The next step is to look beyond individual events to a time series analysis. The assumption is made that a certain sequence of severe events is likely to be an indicator of an undesirable development in a patient's clinical course. Identifying these patterns and raising a warning about this development is the basis for the PDW.

An ideal early warning system would alert a physician to the possibility of a patient being classified as an ICU type patient somewhere after treatment has been initiated, but before it is clinically obvious that the patient is going to the ICU. The goal of this system is to have a warning occur at an early enough point to identify a patient with milder symptoms, which could be more easily rectified. If an EWS can be calibrated to classify a patient as going to the ICU ahead of when a physician might make that determination, it may help to draw attention to that patient and perhaps allow his or her clinical course to change for the better.

5.3.1 Proactive Approach to Early Warning System Event Generation

One problem with the approach of analyzing clinical data from the emergency department is that there may not be enough data to allow us to make an intelligent conclusion about the current clinical status of a patient. For example, a clinical measure may take place at the beginning of an emergency department visit, and at some later point, a second measure may be taken and placed into the medical record. Without any data in between those time periods, the EWS does not have enough data to begin to make any classifications. As an example, consider the following temperature event table with two long event durations (190 and 200 minutes) (Table 18). After the temperature was recorded at 10 minutes, there was no new information for 190 minutes. After that recording, there was a 200 minute interval with no new temperature data.

Table 18: A temperature event table.

Time Stamp	Temperature State (using coloring indication)	Duration Range	Event Duration (minutes)
0	G	1	10
10	Y	3	190
200	G	3	200
400	Y	1	10
410	G		

The EWS makes the assumption that if there is no new data in the system, then the patient must be in the same state as was previously recorded. This is actually useful information that may need to be acted upon. For example, if the event severity is red (a critical abnormal value) and there is no new data for hours, this may be important information. Therefore, a time threshold is instituted and when the time reached, the EWS automatically generates a new event based on the duration range in Table 13. In keeping with our assumptions, this new event has the same states as the prior event; the only change is the duration range. If there continues to be an absence of new data, the EWS continues to create new events, with each event showing that more and more time has passed since the last entry of data. Table 19 illustrates this concept and shows that the new time stamps are generated (130, 190, 320, and 380 minutes) as time thresholds are reached. This proactive approach to event generation can be used in conjunction with any event classifier[3].

[3] Duration is not used with the NBC classifier. That is, even though the events unfold when the NBC is used to classify the event severity, it is being classified based on the seven vital signs alone, without regard to duration.

Table 19: Unfolded event with EWS time threshold generated events.

Time Stamp	Temperature (using coloring indication)	Duration Range (since last data entry)	Event Duration (minutes)
0	G	1	10
10	Y	1	120
130	Y	2	60
190	Y	3	10
200	G	1	120
320	G	2	60
380	G	3	20
400	Y	1	10
410	G		

5.3.2 Flag Rules

As discussed, each event row can be analyzed by the classifiers, and ultimately, for each event row, a combined event type determination can be made. In Section 3, we defined several flag rules to analyze these event severity determinations over time. In this section we refine those rules for the medical domain case study. These rules act as an intelligent agent that looks at a patient's combined event type classification sequence over time and flags a patient when the series of event type severity classifications matches a pattern of interest. To start, three rules have been refined that match patterns that might indicate a patient's transfer to the ICU. When each rule is matched, a "flag" is raised. These rules are based on a sequence of combined event types that are classified as "ICU" combined event type. The hope is that identifying patterns of severe event types will allow for the more accurate identification of patients who will need to go to the ICU.

In rule 1(a), a flag is raised when there are a total of three ICU combined event types. Figure 32 illustrates the flag being raised at the time of the third ICU combined event type classification.

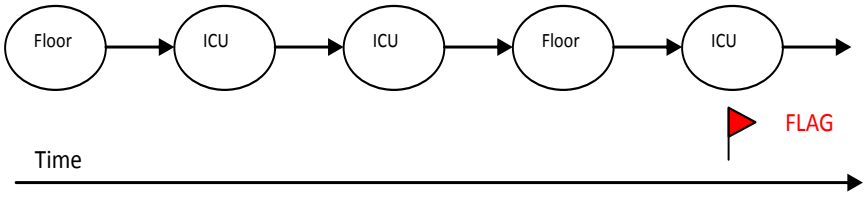


Figure 32: Rule 1a with 1 flag. Three "ICU" event types trigger the flag.

After the flag is raised once, the rule resets until a second set of three ICU event types are generated, at which time, a second flag is raised (Figure 33). Similarly, a third flag would be raised after another three ICU combined event types.

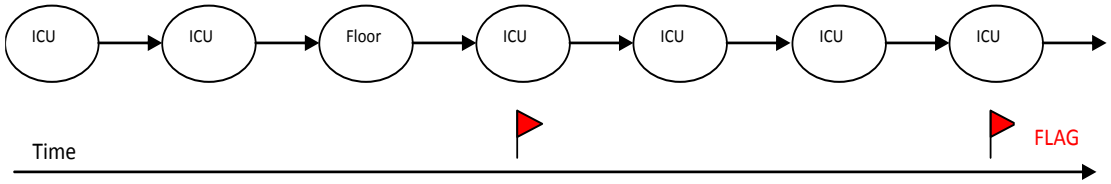


Figure 33: Rule 1a with 2 flags

Based on the dataset, only up to three flags will be considered[4]. Rule 1(a) can be slightly modified to include some of the "ICU" event types from the previous flag. For example, Rule 1(b) in Figure 34 has the same event type sequence, except that it includes one previous ICU combined event type from the prior flag. With this modification, the second flag is one "ICU" event earlier than that shown in Figure 33.

[4] The average number of events in the emergency department is 7 events; therefore, only up to 3 flags are likely to be generated during the period of time when a patient is in the emergency department.

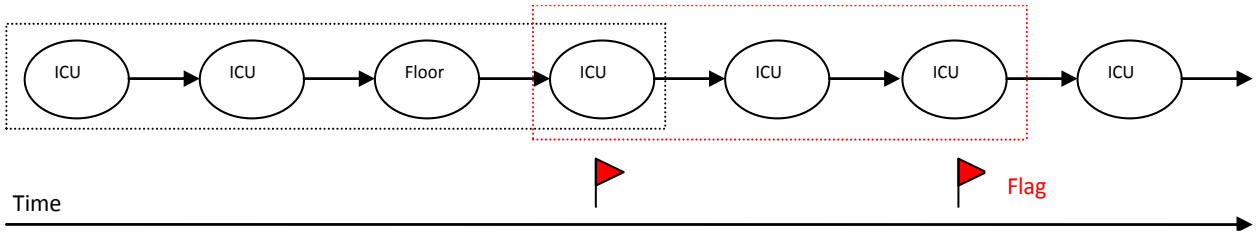


Figure 34: Rule 1(b) using an overlap of one prior ICU combined event type.

This rule can be extended even further to include two prior "ICU" event types (Figure 34)

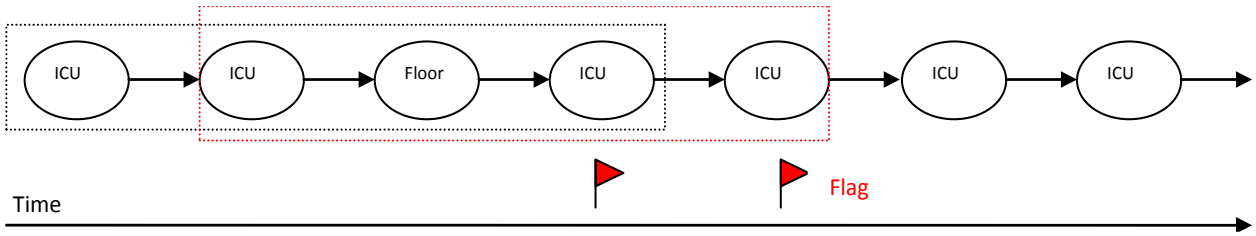


Figure 35: Rule 1(c) using overlap of two prior ICU combined event types

In rule 2, a floor combined event type can cancel an ICU event. Again, the flag threshold is set at three ICU events. Figure 36 illustrates that the floor event type (the third event) cancels one of the previous ICU events, and that the ICU flag occurs at event 5.

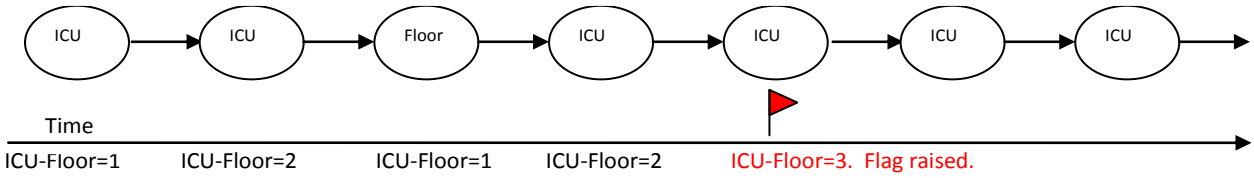


Figure 36: Rule 2. The floor event can cancel the ICU event.

In rule 3, a reading frame is used to determine when to trigger the ICU flag. The window frame size is set to three events. As a new event is recorded, the reading frame shifts forward by one event. The flag will be raised if there are two ICU combined event types within a frame. Consider Figure 37, which shows that the first ICU flag occurs at the first reading frame. As another event is recorded, the frame shifts and no flag is raised. Finally as another event occurs and an ICU combined event type is recorded, a second flag is raised.

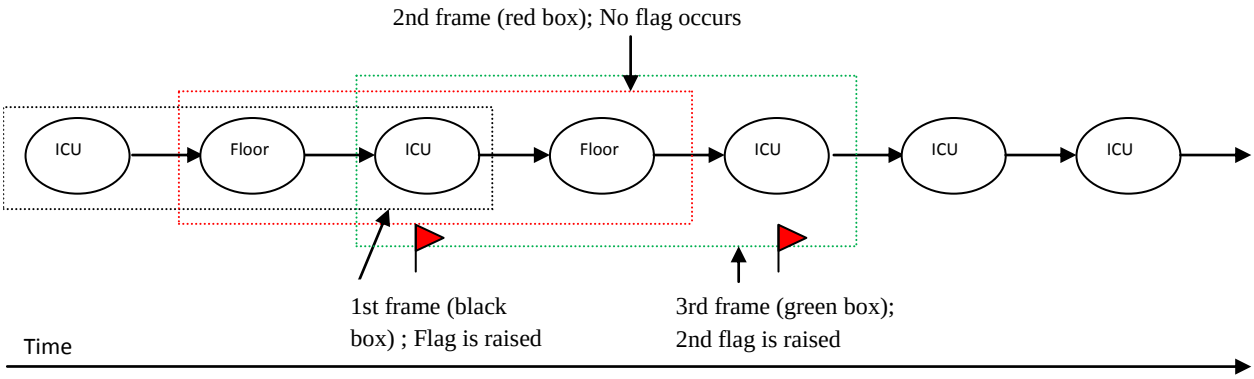


Figure 37: Example of Rule 3.

5.4 APPLICATION: MEDICAL DATA

System Prototype Implementation

After the raw test data was obtained from the electronic medical record (EMR) [88], the data was loaded into an SQL database. Phase I (data stream transformation) and phase II (data adaptation) were created using the SQL Server 2008 Developer edition. T-SQL scripts were developed to pivot the raw data into a usable multivariate vector data stream format, map the data into states, and to fill in the null values. Phase III (event extraction), phase IV (event type generation), and phase V (EWS development) were implemented in Matlab 2009b. Phase III started with an ODBC connection to the SQL database to import the result table from phase II. Functions were

developed within Matlab to extract the data to events. Phase IV was also implemented in Matlab, using the classifiers described above to analyze the event severity and using in-house developed functions to color map the data, unfold the events based on duration, and to generate the combined event type analysis. Phase V (applying the rules to the event severity sequence) was also performed in Matlab. The results of the testing dataset were analyzed in Matlab as well.

Performance Evaluation

The PDW system provides a novel data processing strategy and rules, which are the basis of a new alert system designed to signal clinicians as to the most "reasonable" location (such as ICU or medical floor) for their patients to be transferred after their initial ED visit. The data processing algorithm leads to the generation of an event table to which various decision making techniques can be applied. Three combined event analysis (algorithm) techniques, which we have discussed in Section 5.2, were tested: naïve Bayesian classifier (NBC in Section 5.2.1), decision tree (DT in Section 5.2.2), and neural network (NN in Section 5.2.3). Those three analyses were applied to a set of rules to generate an alert when a patient may have a pattern of severe events which would indicate a set of vital sign states.

Those rules discussed in Section 0 are summarized:

1. Rule 1(a): with no overlap
2. Rule 1(b): with one overlap (includes one previous ICU event)
3. Rule 1(c): with two overlaps (includes two previous ICU events)
4. Rule 2
5. Rule 3

We have tested the performance of our early warning system on a real patient data set. In the data, there were 1,041 floor patients being transferred from the emergency department to a regular floor and 135 floor patients being transferred to the ICU after being admitted to the emergency department. Table 20 explains our predicted outcomes; we consider both early and late to be correct estimates. Note that for the testing data, we collected all patients' data and abstract the period of 10 hours after patients were admitted to the emergency department. This is based on the static study from the data set that 10 hours is enough time for a physician to make a decision on whether to send the patient to the ICU or to the floor. Within 10 hours, we will have an average of 6 events for floor patients and 8 events for ICU patients. We use early estimate, late estimate, and miss estimate to evaluate our predictions, and these terms are defined in Table 20.

Table 20: Definition of correct late and incorrect flagging of patients

How we categorize		*The patient goes to ICU*	*The patient goes to Floor*
Correct Est	Early	Our ICU flag is earlier than the actual move to the ICU	No flag raised, petient goes to Floor as expected
	Late	Our ICU flag is later than the actual move to the ICU	N/A
Incorrect (Miss Est)		Our prediction is wrong. The ICU flag is not raised, but the patient is sent to the ICU.	Our prediction is wrong. The ICU flag occurs but the patient is sent to the Floor.

Finally, to evaluate the performance of these rules and the event severity analysis techniques, we conduct some static tests, based on the confusion matrix in Table 21. The left side of Table

21 is the general usage of the confusion matrix and the right side of Table 21 is what we adopted for the experiment. We set ICU as the positive category and floor as the negative category. There are five measures used, based on the confusion matrix: the accuracy of the rules (Section 0), the performance for ICU patients (Section 5.4.2), the performance for floor patients (Section 5.4.3), the ROC analysis (Section 5.4.4), and the timeliness analysis (Section 5.4.5)

Table 21: Confusion matrix in our study

General Definition			
P: positive N: Negative		actual result	
		P	N
prediction outcome	P	TP	FP
	N	FN	TN

→

Our Study			
P = I (ICU) N = F (Floor)		actual result	
		I	F
prediction outcome	I	TP	FP
	F	FN	TN

5.4.1 The General Performance Test: Accuracy

Figure 38 shows the accuracy of the rules using NN, NBC, and the DT classifier approach to combine event classification. The accuracy is the proportion of true results, which includes both true positives (TP) and true negatives (TN) in the population. An accuracy of 100% means that all predicted values are exactly the same as the given values. In our case, it has described our system, which has the ability of identifying patients who should be transferred to the ICU or the floor. It is calculated by the following equation:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

The accuracy of each rule improves with each flag; the NN event classification approach is comparatively higher (about 90%) at each flag, as compared to its NBC and DT counterpart. This is because NN has a much better assessment for a floor patient (higher TN, lower FP).

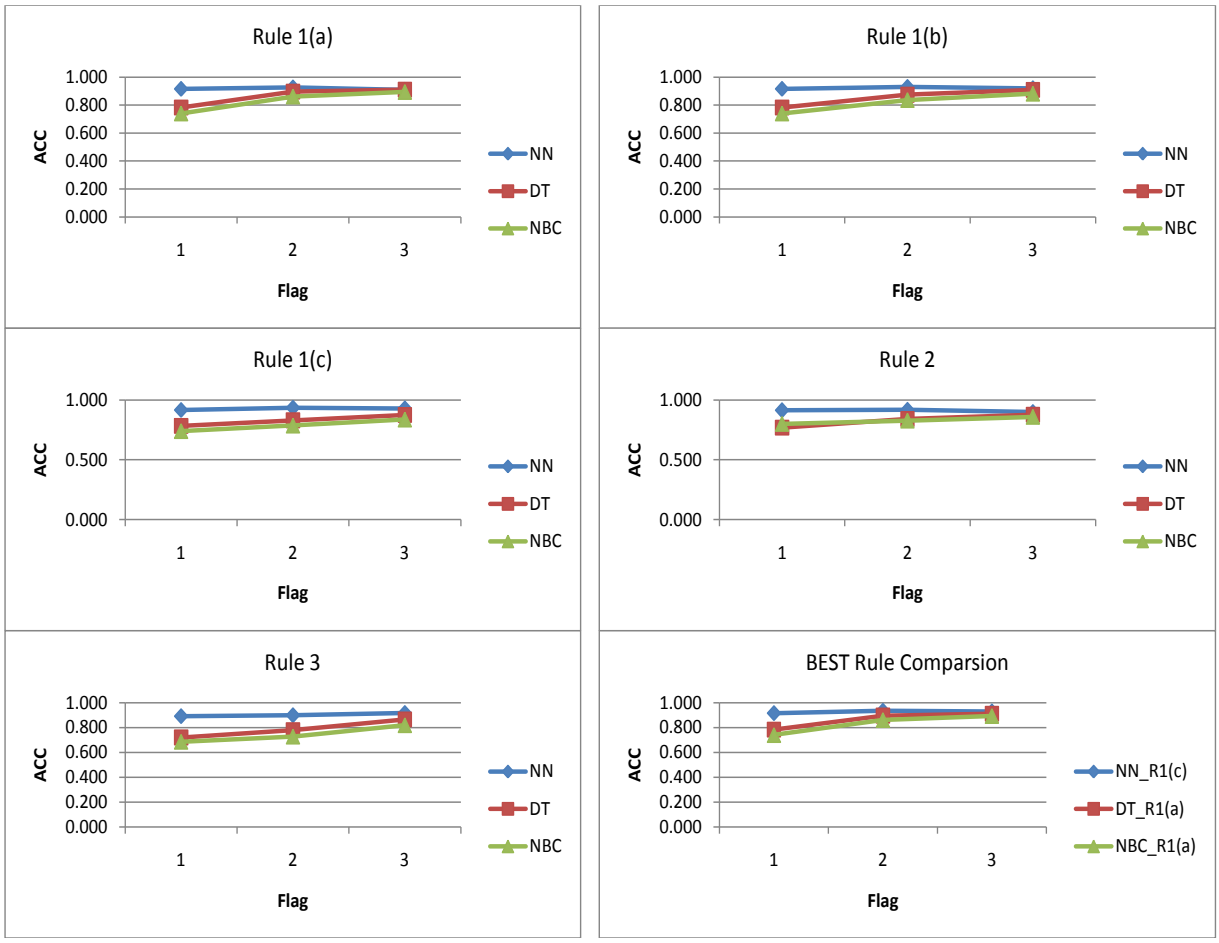


Figure 38: The accuracy analysis.

5.4.2 ICU Patient Analysis

In this subsection, we focus on the evaluation of the ICU patient. We use both sensitivity and positive predictive value (PPV) analysis to describe this test. The "*Sensitivity*" tells us the proportion of ICU patients who are correctly identified. In the other words, it shows the ability to correctly find ICU patients from all ICU patients (ICU patient population). On the other hand, the "*PPV*" describes the proportion of patients with ICU prediction results that are correctly predicted. Namely, it is designed to answer the question: *"If we made x ICU predictions, how*

many are the correct predictions out of those x predictions?" We will discuss our test results from these two analyses in the following two subsections.

5.4.2.1 The Sensitivity Analysis

Figure 39 shows the sensitivity of each technique using each of the rules. The sensitivity is calculated by the following equation:

$$Sensitivity = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Based on the formula, the sensitivity determines the relationship between TP (correctly predicting ICU patients going to the ICU) and FN (wrongly predicts ICU patients going to the floor). As shown from all sub-figures, the sensitivity is uniformly higher for each of the rules that uses the DT event analysis technique. In view of applicability and completeness, the sensitivity drops along with the number of flags that increase with all rules; this is an artificially low number, because as the number of flags increases, the number of patients decreases who have enough events (such as longer stays in the emergency department) to potentially generate the second or third flag. The sensitivity is comparatively higher for the DT approach to assessing overall event severity.

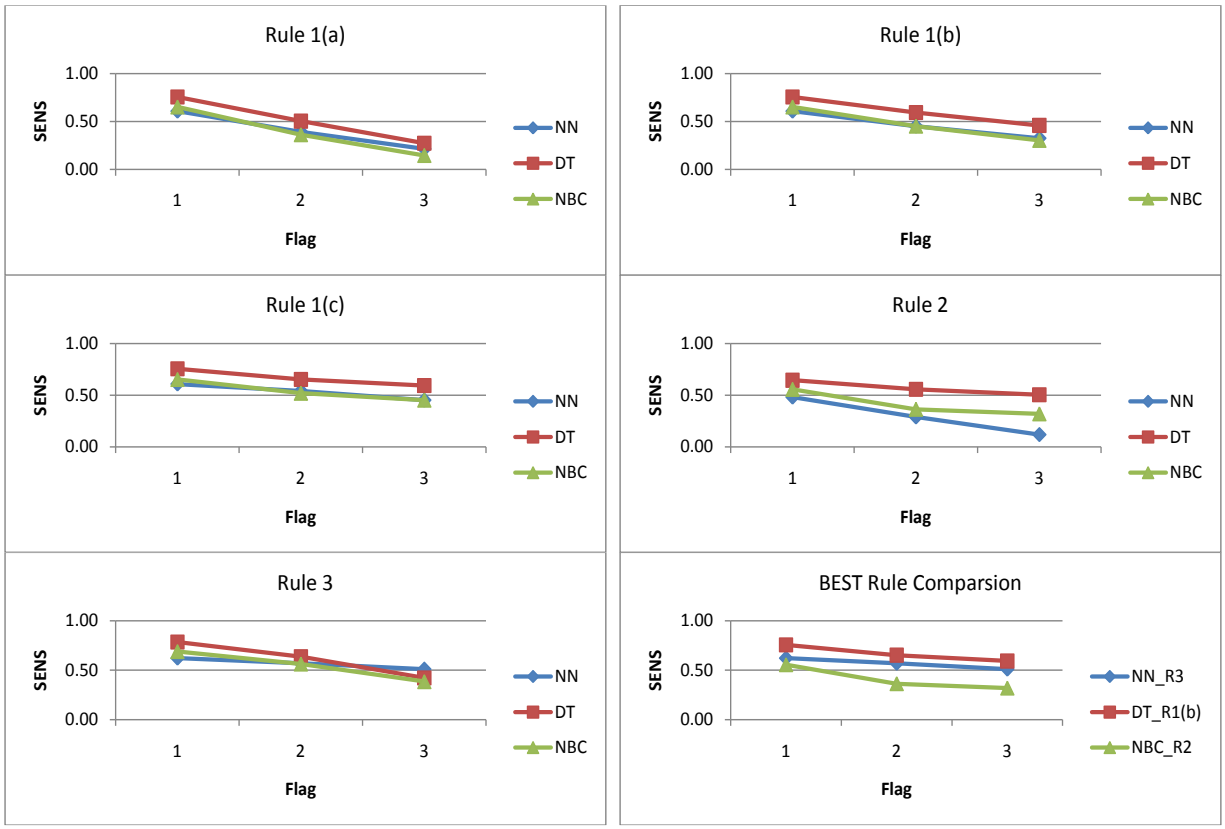


Figure 39: Sensitivity analysis

5.4.2.2 The PPV Analysis

Figure 40 shows the result of the (PPV) analysis based on different rules; the PPV is calculated as in the following equation:

$$PPV = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

As we can see in the correctness view, PPV increases with each flag. As one continues to wait for more and more flags, the chance decreases that a flag raised is not one for a true ICU patient. In other words, more flags usually have higher confidence and assurance that a patient will be transferred to the ICU. In this test, NN obviously has better performance than the DT and the NBC. As we look at the PPV formula, the PPV is affected by both TP (correctly predicting

ICU patients going to the ICU) and FP (wrongly predicting floor patients going to the ICU). We examine the details of this phenomenon and in our observations, DT actually has higher TP values in all rules; however, it also generates some FP values. The NN has many fewer FP values in all rules, as compared to DT and NBC, and as a result the NN has a better PPV value for all cases.

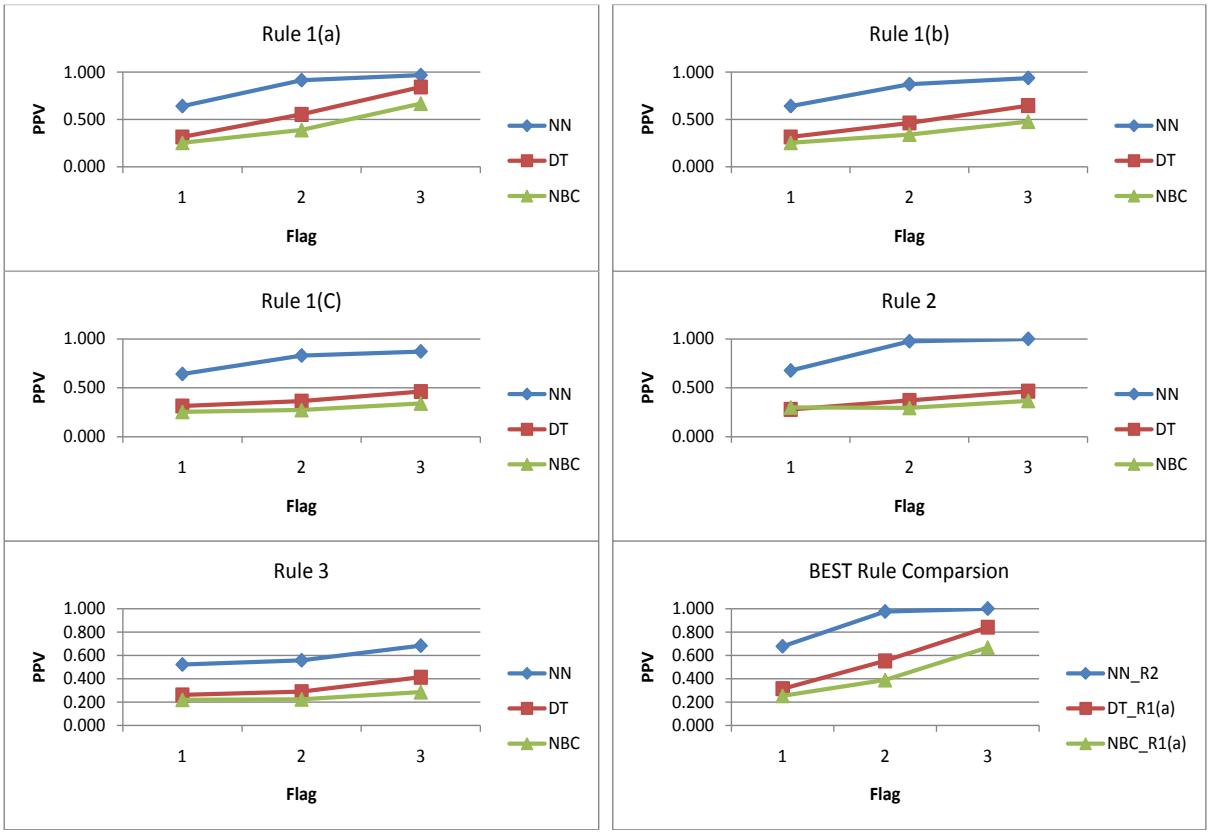


Figure 40: The PPV analysis.

5.4.3 Floor Patient Analysis

In this subsection, we perform an evaluation for floor patients. The measurements we adopt for floor patient analysis are specificity and NPV negative predictive value (NPV). The "*Specificity*" explains the proportion of floor patients who are correctly identified. In the other words, it shows

the ability to correctly find floor patients from all floor patients (the floor patient population). On the other hand, the "*NPV"* describes the proportion of patients with floor prediction results who have been correctly predicted. Namely, it is designed to answer the question: "If we made x floor predictions, how many of them are the correct predictions?" In the following two subsections, we illustrate the details of the test results.

5.4.3.1 The Specificity Analysis

Figure 41 shows the specificity of each technique, using each of the rules. The specificity is calculated by the following equation:

$$Specificity = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Based on the formula, the specificity discuss about the relationship between TN (correctly predicting floor patients going to the floor) and FP (wrongly predicting floor patients going to the ICU). In our study, when the number of flags increases, the value of FP greatly decreases. This situation causes increasing specificity when the number of flags is increased. In this test, NN performs much better than others, since the FP value is much smaller than in the other two approaches.

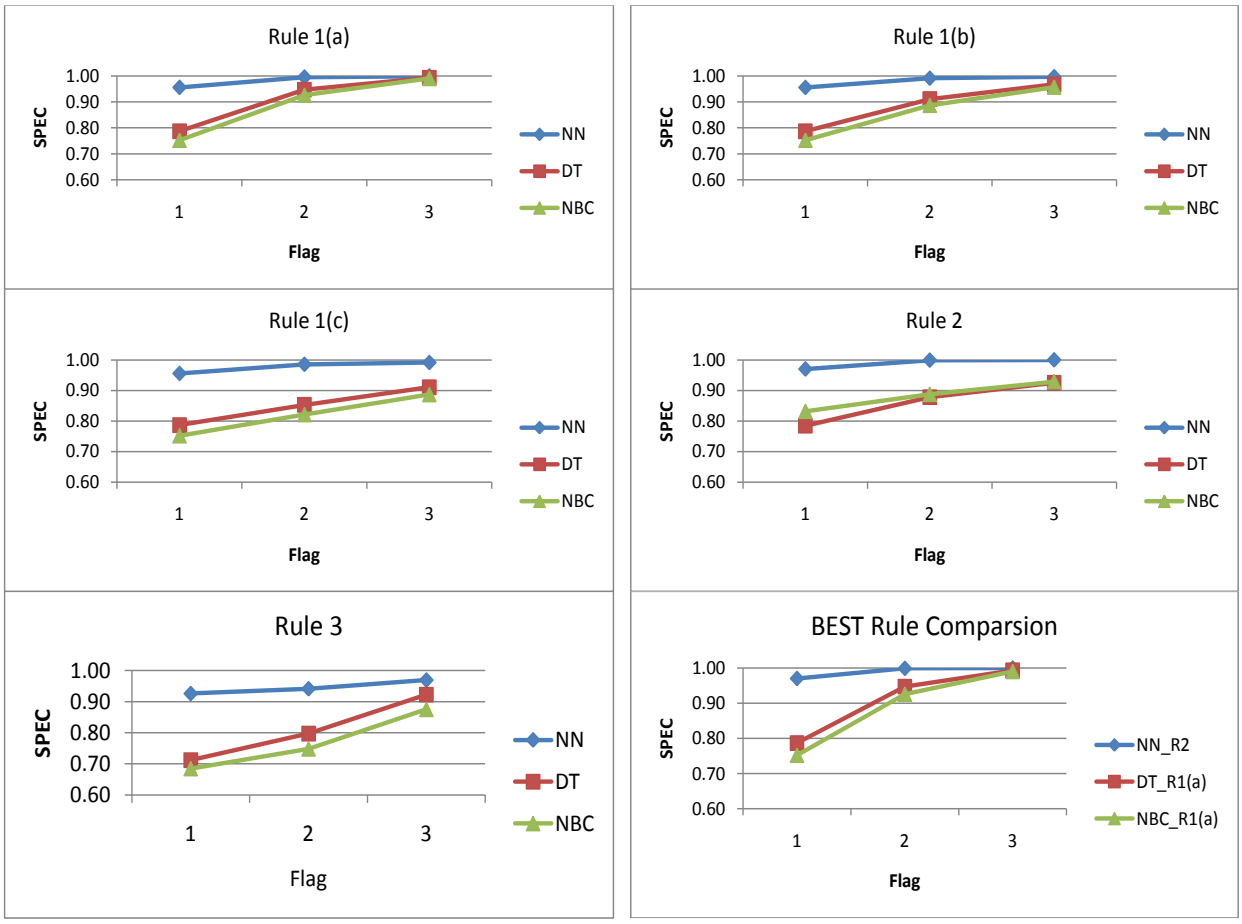


Figure 41: The specificity test

5.4.3.2 The NPV Analysis

The negative predictive value (NPV) analysis shows the relationship between TN (correctly predicting that floor patients will go to the floor) and FN (wrongly predicting that ICU patients will go to the floor). It can be formally described in the following equation:

$$NPV = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}$$

In this test, we found that when the number of flags increases, the value of FN will slightly increase. As a result, Figure 42 shows that the value of NPV decreases as the number of flags

increases. In this test, DT performs better than the other two approaches in all rules. However, in general, three algorithms perform very well in all rules, since they all have an NPV value greater than 0.9.

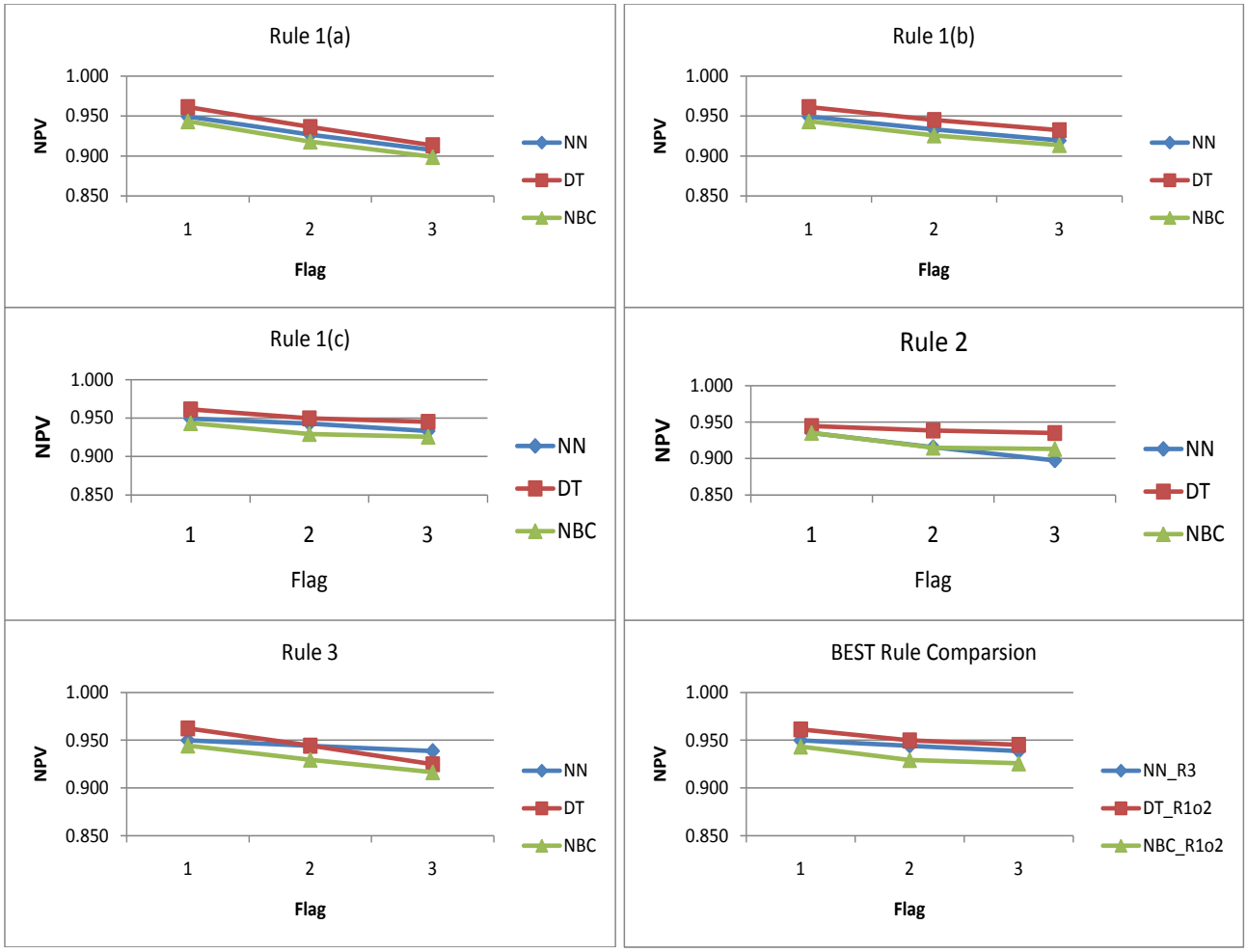


Figure 42: The NPV analysis**.**

5.4.4 The ROC Analysis

In this case, a receiver operating characteristic (ROC) analysis is performed to compare the three severity classification approaches, based on each early warning rule. Figure 43 indicates that the DT has better performance in the true positive rate (TPR), which means DT is more appropriate to use to find ICU patients from the overall ICU population. On the other hand, given the false positive rate (FPR), NN has a higher ability to identify floor patients, since it rarely sends floor patients to the ICU.

Since there is a tradeoff between DT and NN and we would like to know which rule is most appropriate to apply to this ROC analysis, we use the area under curve (AUC) for the comparison. The bottom right of Figure 43 shows the comparison of the three approaches from the best rules. The result tells us that the DT (AUC=0.81) is slightly better than that of the NN (AUC=0.79), and that the NBC has the worst result (AUC=0.73).

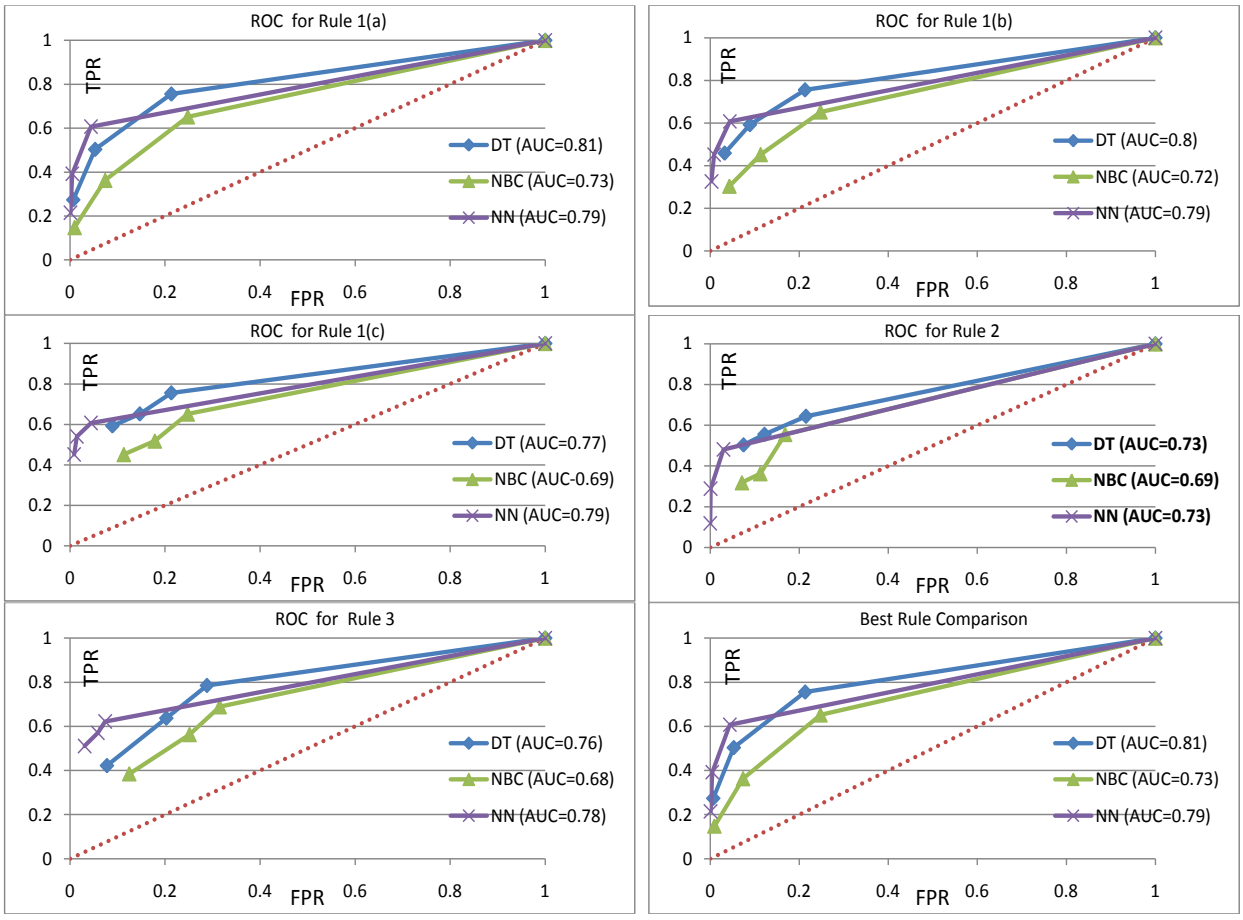


Figure 43: The ROC analysis.

5.4.5 The Timeliness Analysis

In the previous sections, we used different measures to evaluate the accuracy of our three approaches. In this section, we focus on the early warning rules and demonstrate the detection time of our early warning rule, regarding both the early and late estimates for ICU patients based on Table 20 in Section 5.2. Figure 44 shows the percentile plot in both NBC and DT, based on the different number of flags. A negative time value indicates the number of minutes prior to actual transfer to the ICU, 0 minutes means the prediction time is the same as the actual patient transfer time, and positive minutes reflect a late estimate time. For example, the percentile plot result from NBC using the flag is shown in the left top diagram. It indicates the distribution of

correct estimates (both early and later) using NBC with one flag. By looking at the time axis, we can see the distribution of early and late estimates, and all of our rules have a larger portion on an early estimate.

The timeliness of the rules has consistently favored rule 2, because rule 2 can alert as early as the first event if that event is of the ICU event type. While this may be advantageous in some circumstances, it is possible that rule 2 will fire too often and cause alert fatigue. Another issue with rule 2 is that if there is a string of several floor event types, it will take an equal number of event types in order to get back to a state where a flag can be raised. This does not necessarily make sense clinically; that is, just because a patient was clinically stable for the last three days does not mean that he or she cannot abruptly worsen in a single hour.

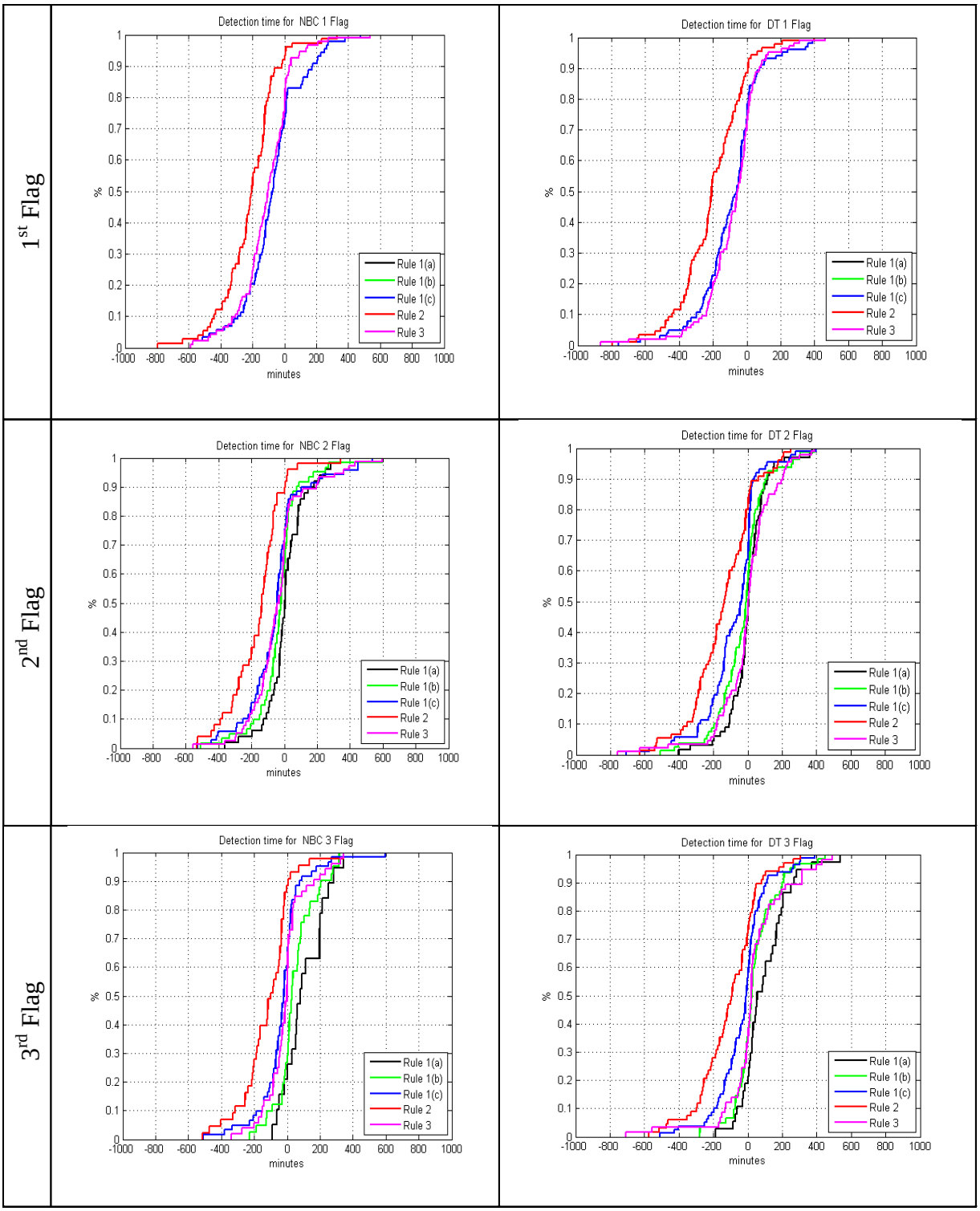


Figure 44: Detection time for NBC versus DT for the first, second, or third flag.

5.5 SUMMARY OF THIS MEDICAL CASE STUDY

This section describes the development of this system, along with some analysis of the performance of our PDW. There are comparisons between the three event severity classifier methodologies. Overall, in this case study, the neural network approach to decision making was superior to the naïve Bayesian classification and decision tree approaches, since it has a better performance in the test of accuracy (Section 0), and a much better result in both PPV (Section 5.4.2.2), and specificity (Section 5.4.3.1). This is likely due to a more granular approach to the data. The decision tree classifiers took each vital sign state into account individually, along with the state transition from the prior event and the duration of time of that event. The NBC approach did not account for either transitions or duration, which are both considered to be key indicators of potential clinical problems. Also, the NBC takes all seven vital signs in aggregate, rather than individually. The neural network neither uses each vital sign individually, like DT, nor takes all 7 vitals together as an input, like NBC. It partitions all vital signs into three groups based on their correction. As a result, it is more accurate and reflective of physicians' diagnoses. The rules developed all behaved in a similar manner with regard to sensitivity, specificity, positive predictive value, and negative predictive value at the first flag level. At the second and third flag levels, the varying rules were slightly superior. For instance, Rule 1a, which required the largest amount of ICU combined event types to raise a flag, had the highest positive predictive value.

There are several potential limitations that should be noted about this study. At its current stage of development, the system does not account for other potential sources of data that could be useful in determining a patient's clinical status. For example, variables such as neurologic status, nursing assessments, or laboratory data could be very helpful in giving the model a fuller picture of a child's clinical status. While these variables are likely to be useful, the system shown here represents a "lowest common denominator" approach to an alerting system. Introducing less objective measurements into the system may unintentionally obfuscate the signals. Another limitation worth noting about this alert system is that vital sign data, while seemingly "objective," can be surprisingly inaccurate [89]. While this is true, the system described here accounts for this, as the classifiers are both built on a sample of retrospective data, which represents the best data available in any EMR system. Another area of possible concern is the

assumption made that the last state of a vital sign remains constant until new information is entered into the system. As previously discussed, this is a reasonable assumption to make in the emergency department, where there is often a specific vital sign abnormality that is being addressed and measures are taken to directly affect that abnormality. An alternative is to allow for null values to continue existing in the event table and have the severity analysis techniques take the state of "null" into consideration. The event analysis techniques described here, NN, NBC, and DT, are not the only ways to analyze the event severity. Other methodologies including cluster analysis, neural networking, or simple linear regression analysis may provide superior results. The two techniques described here may lack enough discriminatory power, due to the limited sample size of the training set. With more and more data, the models may improve in their sensitivity and specificity. Another criticism of this work is whether the intention of this work is to replace clinical judgment. The intention is not to replace current clinical judgment; rather, the system is designed to augments the current state of affairs in a busy emergency department where subtle clinical signals can get lost.

6 CASE STUDY II: HISTORICAL EPIDEMIOLOGICAL DATA

In this chapter, we perform an experimental study of the PDW and architecture using data obtained from Tycho, which is a large integrated epidemiological database that includes historical data from numerous heterogeneous sources. Tycho contains approximately 50,000 semi-structured and varied data in Excel spreadsheets that reports epidemiological data from the United States over 100 years. Figure 45 shows a sample of the original Excel spreadsheet data which is heterogeneous, semi-structured, and may include missing values.

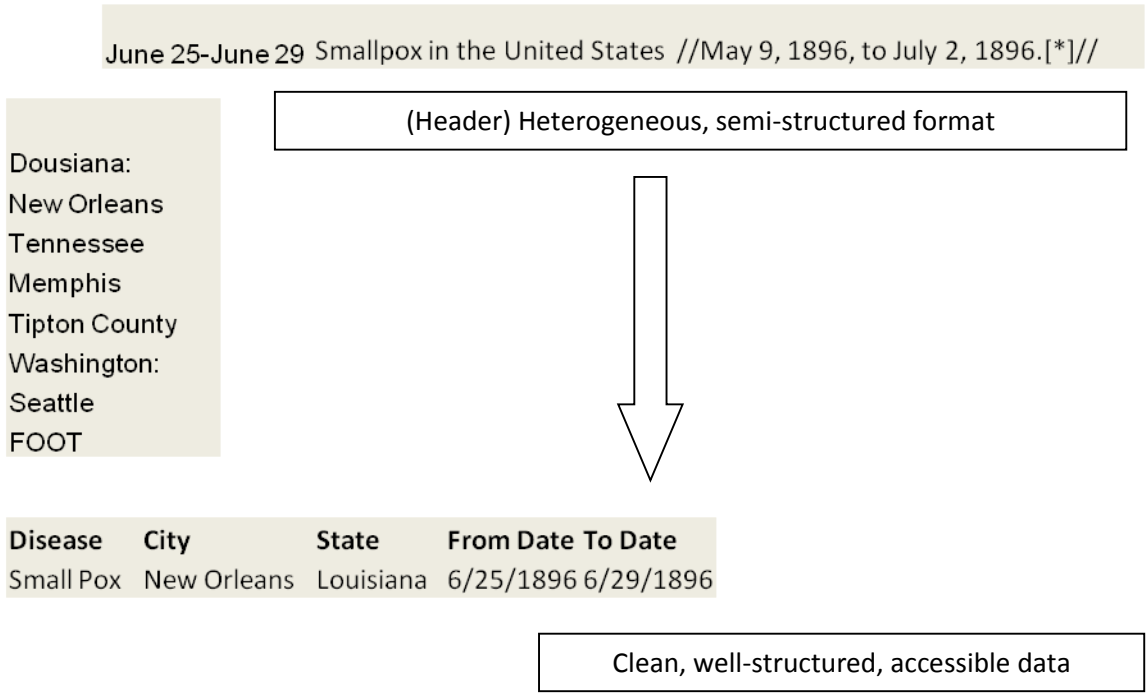


Figure 45: An example of the transformation from dirty Excel spreadsheet data to well-structured data

Within the original data files, some values and other information cannot be scanned due to light marking. This machine-unreadable information is marked as "unclear" and was later viewed manually in the original hard copy reports, which are located in Hillman Library. Those Excel spreadsheet files come from 20 batches of Excel data sheets examined in April 2010 from the University of Pittsburgh's Graduate School of Public Health. Each batch contains multiple workbooks that represent many years of health report issues, and each Excel workbook may contain multiple sheets of data. All batches represent over 100 years of data, which comes from approximately 1880 through 2010. We designed and developed an integrated epidemiological data warehouse, called the Tycho warehouse. This work was undertaken in collaboration with the University of Pittsburgh's School of Public Health. This work enabled us to address realistic historical data fusion challenges that go far beyond purely academic interests. Our work is one of the first attempts to systematically investigate the challenges of large-scale historical data fusion. In addition, we have introduced and developed conflict-aware data fusion for the efficient aggregation of historical data. This provides a solution to reduce data aggregation errors in an integrated historical database.

In Section 6.1 and Section 6.2, we elaborate on the PDW architecture with respect to Excel spreadsheet transformation into a well-structured and accessible database with data validation. We introduce the concept of conflict-aware data fusion for the efficient aggregation of historical data in Section 6.3, where we also discuss the incorrect estimation (both overestimation and underestimation) of conflict data. In Section 6.4, we introduce the concept of conflict degree (CD) to assess the risk of the incorrect estimation of values that may be reported in conflicting historical tuples. In Section 6.5, we suggest a simulation-based approach to estimate an optimal CD threshold value for groups of overlapping reports without the knowledge of actual event distributions. We implement and tested our approach in the Tycho data warehouse, where we observed up to 32 conflicts that occurred over days, weeks, months, and years.

6.1 PHASE I: SCHEMA TRANSFORMATION

Since there were over 50,000 Excel files, we could not do an initial data assessment to make sure all of them could be loaded the database properly. We designed our algorithm to learn the data issue on the fly (during data loading) and to properly process them. We have experienced the following problems with Excel files that obstruct the loading process.

1. Extraneous number of column headers or other redundant information. The Excel spreadsheet may have more than one table that contains the redundant data.
2. Headers may be missing or the column order may have been changed in the copying process from PDW files to Excel files.
3. The Excel files may have erroneous dates, (designating the time period of reported diseases) in the sheet titles, column headers, or body columns.
4. Data values are misaligned with the correct place names.
5. Typos may occur anywhere. For example, the disease name in the header versus the sheet title may have a different spelling.
6. Some data sheets contain known errors alongside correct records. These errors were corrected by a third-party service provider company. When these corrected data sheets were reentered into the system, the records that have already been corrected are now duplicated. These duplicate records can be screened out from the final database.

Figure 46 shows a conceptual view of the data transformation process. First, we get the title, header and rest of the body of an Excel spreadsheet, and then clean and extract the desired data in the following schema in the database, as the staging table for later processing. This procedure has been designed under Microsoft SQL Server Service (SSIS), which is an ETL tool and a core component of MS SQL Server. For more information on detailed design and implementation, please refer to APPENDIX 3: and APPENDIX 4: QUERY OPTIMIZATION AND TYCHO WEB APPLICATION.

Disease	City	County	State	Region	Country	From Date	To Date	Reported Outcome	Value

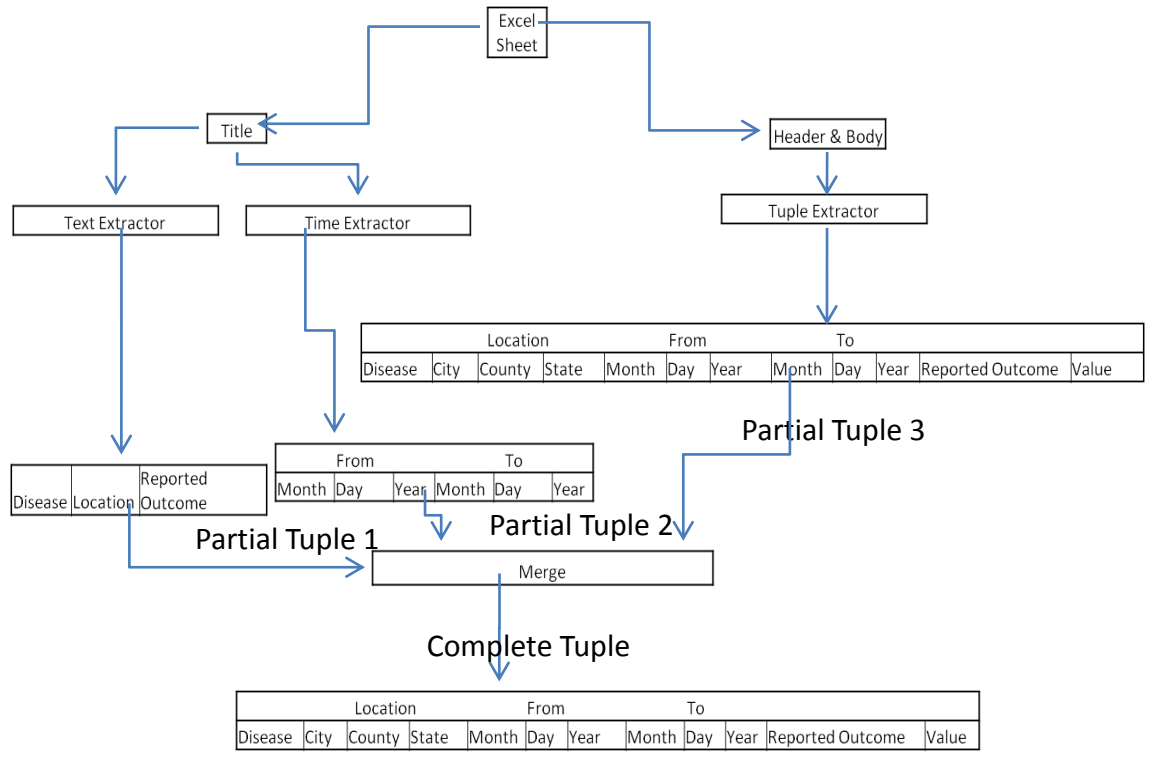


Figure 46: A conceptual view of the data cleaning process

6.2 PHASE II: DATA ADAPTATION

The purpose of the data adaptation process in our PDW architecture is to achieve a better representation of the data. In our medical case study, we classified data into more meaningful state by changing the data granularity. To do this, we performed a state mapping rule that groups a range of vital signs values to their states.

In this historical data chapter, the task of data adaption was designed to fix the data errors in the different perspectives of the original data schema. This historical data schema is composed of four domains: location, time, reported outcome, and value. Our data adaptation task is to recover the original semantic meaning of those data. From the data, each tuple shows the reported event

of either the number of cases or the number of deaths from a disease that happened in a given time period. Using the fuzzy logic operation, we try to assess and recover the original meaning of those terms. We set up a predefined reference database to be the standard and let all of the non-standard terms match the most likely term from the reference database.

6.2.1 Data Processing with fuzzy grouping and look-up

- Location

Figure 47 shows an example of fuzzy grouping and fuzzy look-up functions. Using a fuzzy group transformation, both correct spellings and misspellings of *Louisiana* can be assigned to a correct name. Meanwhile, when a word is input with the wrong spelling, the fuzzy look-up transformation can help to identify the proper spelling, based on predefined words spelled correctly in the reference table. For example, in this case, the fuzzy look-up transformation looks up the reference table with correct state spellings and matches *Lousianat* to *Louisiana*, with a higher value of similarity and confidence.

Both fuzzy grouping and fuzzy look-up are based on the concept of using fuzzy logic, which is an approach of computing based on "degrees of truth." We use fuzzy approaches in this case study, since we are trying to fix errors made by human mistakes (typo). Fuzzy logic seems closer to the way our brains work. We first aggregate data and form a number of partial truths (fuzzy grouping), which we aggregate further into higher truths (reference table). Based on the reference tables, our process can return a degree of *similarity* or *confidence* thresholds, so we can have a higher certainty in our selection results.

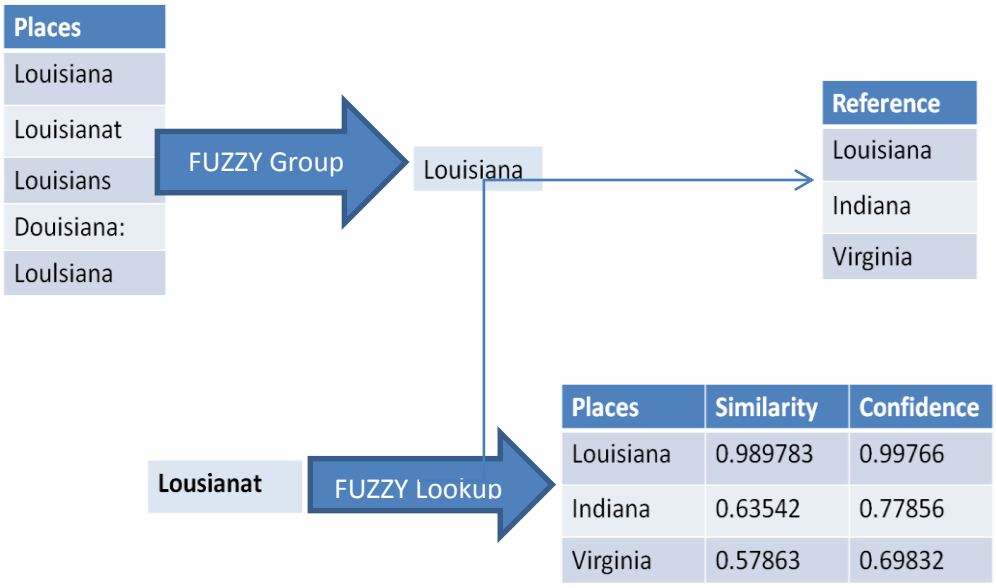


Figure 47. The use of fuzzy logic to clean data

Figure 48 shows the location cleaning algorithm that uses fuzzy grouping and look-up, as described above. Initially, all the original locations are considered to be unclean locations. We first group them into distinct locations using exact-match look up, and then apply the fuzzy logical operation to group similar terms together using reference tables. We repeat this operation for locations of "states," "cities," and "countries." Finally, we checked the existence of combinations of "states," "cities," and "countries," and filtered out those that do not exist in the US geographic data reference.

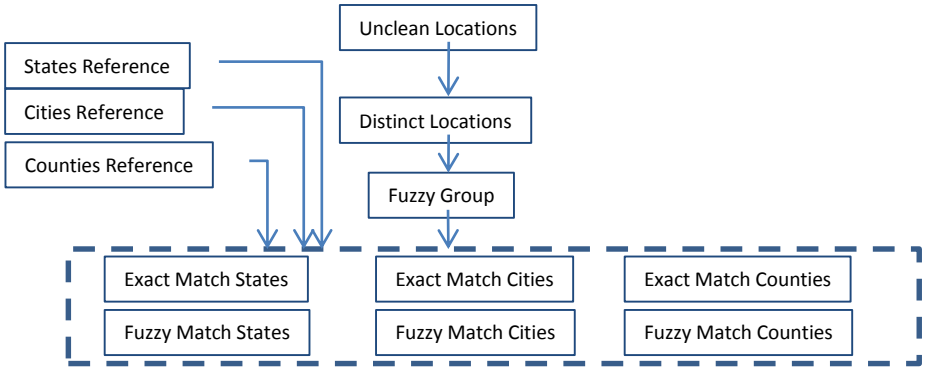


Figure 48. Cleaning the location data

- Disease

We currently have 62 major diseases in our database. Since some diseases contain subtypes (for example, hepatitis has type A, type B, and others), we can further categorize diseases into 143 kinds. In an Excel spreadsheet, the disease information is either in the title or in headers. There was an unknown number of spelling errors for disease names, which we could not pre-calculate or correct before transferring them to the database. In order to solve this issue with spelling inconsistencies, we use fuzzy logic to parse all disease names. The fuzzy logical processing is based on a reference table, which is a list of a predefined standard disease names.

- Time

The disease and time data are extracted in the similar manner from the title, header, body, and cells of an Excel spreadsheet. For example, in Figure 49, the start time (1896-03-28) and end time (1896-04-04) are extracted using the time column cell (March 28–April 4) to obtain the dates, and year information is extracted from the title (Smallpox in the United States as reported to the Supervising Surgeon-General, United States Marine Hospital Service //December 26, 1896 to April 21, 1896)) using fuzzy logic, regular expressions, and a precedence of time-column values over the time value from the title). Similarly, disease (hepatitis) and its sub-category (viral) as well as by type (NA, NB) are extracted from the header (Hepatitis (Viral, by type, NA, NB, 1896)).

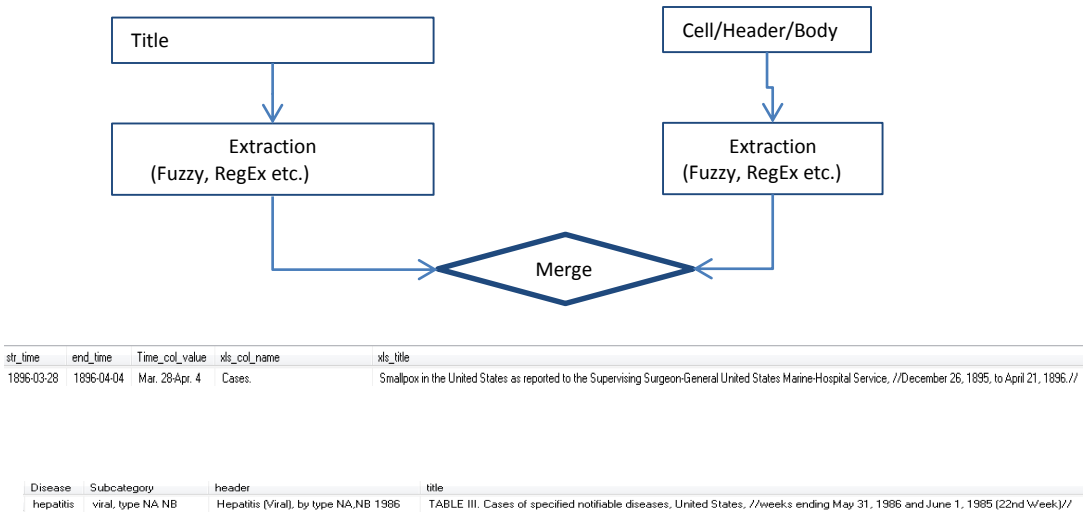


Figure 49. Cleaning the disease and time data

- Reported Outcome

Reported outcome describes the characteristic of the tuple. It can be "case," "death," or "a_case." Note that "a_case" implies the tuple "assume is case:" because the original data in the Excel spreadsheet does not specify "case" or "death," we assume it is "case."

- Value

Values in the Tycho data describe the number of diseases that are related to cases or deaths. We verified those values using random samples and compared them to their original data source. Those values were the raw information associated with the disease names (both main disease name category and disease name sub-category), the geographic names (place of occurrence), and the reporting period (from date to date). We applied regular expression to remove non-integer elements, such as symbols, commas, periods, braces, or letters.

- Secondary Data

The data set also contains other information, such as population counts, comments, remarks, footnotes, and statistical relationships, such as medians. As of this version of the data (from

August 5, 2010) a validation protocol must be produced and should eventually be integrated with the primary data.

6.2.2 Validation

Since there is no single correct result that could let us compare our approach, our validation test algorithms were designed based on statistical results, and were implemented using SQL query. In the following, we discuss our validation tests for disease names, locations, and time stamps.

Validation Disease Name

The disease names, dates of reporting periods, and locations of report coverage are the primary concerns at this point. Disease names occur throughout the data set and are associated with a unique identification number. A two-column reference table is constructed that contains all the distinct disease names that were reported throughout the time period (from 1880–2010). The other column will list the associated unique id numbers. For this project, there were 56 disease types or names. Batches of data are screened for the number of distinct disease names that occur. Each batch (and indeed, the entire data set) should have either less than or a count equal to the total number of diseases in the reference table. A disease count that is greater than the reference table count will represent noise or unwanted data that should be examined in more detail. The noise could be misspellings, unauthorized disease names, or misplaced text from the ETL procedure. A SQL aggregate query is used to check this basic constraint.

To summarize, we perform four tests for disease name checking:

Test one: an aggregate query to compare the number of distinct disease names in the database with the number in the reference table and compare the number of distinct disease sub-categories in the database with the number in the reference table. Both comparisons should be either less than or equal to the number of distinct values in the reference tables.

```
SELECT COUNT(DISTINCT D_name)AS Distinct_Main_Diseases
FROM Database Batch #.data table


SELECT COUNT(DISTINCT D_name)AS Total_Possible_Main_Diseases
```

```
FROM Database Batch #.disease reference table


SELECT COUNT(DISTINCT D_s_name)AS Distinct_Sub_Categories
FROM Database Batch #.data table


SELECT COUNT(DISTINCT D_s_name)AS Total_Possible_Sub_Categories
FROM Database Batch #.disease reference table
```

Test two: check errors in disease names and disease sub-categories that are not in range within the reference table values. We can check for:

1) Standardized spelling

for example, AIDS, not "Acquired Immune Deficiency Syndrome"

2) Typos

for example, DIPHTHERIA, not DIPTHERIA

3) Non-diseases

for example, MORTALITY

4) null in disease name (not allowed); empty strings in sub-categories (allowed).

This is a non-set membership test that uses the negated logical operator “NOT IN” with the reference table.

```
SELECT *
FROM Database Batch #.data table
WHERE
 D_name NOT IN
      (SELECT D_name FROM Database Batch #.disease reference table)
 OR
 D_s_name NOT IN
      (SELECT D_s_name FROM Database Batch #.disease reference table)
```

Test three is to compare all correct disease names with all correct disease sub-categories and check for distinct mismatches. The “EXCEPT” operator combines two SELECT statements and returns rows from the first SELECT statement (the database) that are not returned by the second SELECT statement (the reference table), and the query result represents the mismatches. Note that duplicates are dropped and the results are distinct mismatches. Also note that the clause "EXCEPT ALL" is not supported in SQL Server to display all the mismatched records. Further analysis is needed to display all mismatched records, as seen in test four below.

```
SELECT D_name, D_s_name
FROM Database Batch #.data table

EXCEPT

SELECT D_name, D_s_name
FROM Database Batch #.disease reference table
```

Test four is used to find all records of each distinct mismatch that are discovered by test three.

```
SELECT id_num,file_name,sheet_name,xls_title,xls_col,
            xls_col_name,xls_row,D_name,D_s_name
FROM Database Batch #.data table
WHERE       D_name = 'meningococcal infections'
            AND
            D_s_name = 'meningococcus'

ORDER BY file_name, sheet_name
```

Validation for Time

The time periods are checked for a number of constraints by eight SQL tests or queries, which are shown together here. The year, month, and day must all be values within their ranges: for example, the year must be any value from 1880 through the present year. The start time must be in chronological order in relation to the end time; that is, the start time must be less than the end time, and the start time and end times cannot be the same. The queries can be run as a single constraint at a time, in different combinations, or all at once in SQL Server. The time periods data is involved in an external validation process. The database of the SIS team is compared to the database of the GSPH team to search for any mismatches.

```
SELECT distinct file_name, sheet_name, str_time, end_time, xls_title,
            Time_type, Time_col_value
FROM Database Batch #.data table
WHERE          ( str_time > end_time
                  OR str_time = end_time
                  OR DATEPART("YYYY",str_time)< '1880'
                  OR DATEPART("YYYY",end_time)< '1880'
                  OR str_time > GETDATE()
                  OR end_time > GETDATE()
                  OR str_time IS NULL
```

```
                OR end_time IS NULL
                )
                      AND
                      (Loc_org NOT IN ('FOOT', 'DONE', 'NOTE', ''))
ORDER BY file_name, sheet_name
```

Validation for Location:

The internal validation process for location place names involves many constraints. The city must exist in the city reference table. The county must exist in the county reference table. The state must exist in the state reference table. The region must exist in the region reference table. The country must exist in the country reference table. The city must exist within the state (city/state pairs). The county must exist in the state (county/state pairs). If needed, a county name can be reconstructed from a city/state pair, except when there are multiple cities with the same name within a state. For example, there are two cities called "Mason" that are located in different counties in Kentucky. The original data may not give the county name. Any place name pairs that do not match any records in the reference tables are errors that must be resolved by scripts, web searches, or original data lookups.

Potential problems include mismatches of place name pairs and typos. It is possible to have a legitimate city that is mismatched with a legitimate state. For example, there are cities named Pittsburg/Pittsburgh in many states, such as PA, CA, TX, NH, and MO. A check with the original data sheets may be needed to confirm the intended state. Typos might include a misspelled name, including a missing word from a two-word name: for example, "Springs" instead of "Colorado Springs." Typos may also occur within the state abbreviation or occur within both names in the pair.

To perform a website lookup for a place name that exists in a state, the SIS team has used regular Google searches, searches with "Google Maps," the Geographic Names Information System (GNIS) website at http://geonames.usgs.gov/pls/gnispublic/, and the U.S. city & state gazetteer website at http://www.hometownlocator.com/, which uses the same GNIS data but has a search interface that runs queries faster. The only website reference used for place names in the Marshall Islands was the GNIS site. For U.S. data searches, the results page may show the same place names for different categories. For example a "city name" could be a county or township

name as well. As a result, care must be taken to confirm the place name category that is displayed in the results page.

This project uses a dataset with some records of considerable age—some as old as 1880. It is expected to encounter records that show changes in place names over this time period. The GNIS and Gazetteer websites provide the current city name in response to searching an old name from this project's source data. Additional information about the current city is listed with the "Unofficial" or "Variant" name(s), which are subordinate to the current name. For example, a typical Google search for "Wright, Washington" provides no relevant responses among the first 20 responses. A Google Map search for "Wright" and "Wright, Washington" provides no relevant responses. The search "Wright, Washington" with the U.S. City & State Gazetteer website displays: "Klickitat" and "Alternate unofficial names: Wright, Wrights." A search for "Wright, Washington" with the GNIS website displays: "Klickitat" and, "Variant Name: Wright, Wrights."

The GNIS website also contains historical names, which are defined as, "a feature with "(historical)" following the name that no longer exists and is no longer visible on the landscape." For example, in this project, "Atlanta, WA" is listed as a historical name.

To facilitate the external validation process for location place names, the SIS team uses logic scripts to clean the original location data. Then, the cleaned data can be compared to the GSPH team's location data.

6.3 PHASE III: EVENT EXTRACTION

After solving the above issues of disease names, locations, and time stamps, in this section we focus on *event extraction*, which is the procedure to estimate the value of case or death for a time interval for a given disease. In fact, we consider that the Tycho data is already event-oriented data, because it uses various time intervals to represent an event (for example, the number of cases of measles in Los Angeles in 1900). Our task is to solve the issues of redundant reports, which may cause conflicts with certain time intervals. In this section, we review the issues of

redundant data in a historical database, which includes conflicting tuples and risks of misestimation from the aggregate data.

The historical disease data may be reported by different timeframes, such as weekly, bi-weekly, monthly, quarterly, or yearly. Data represented by different timeframes may cause data redundancy. We defined that a time interval is redundant if it is includes two or more conflicting tuples. Figure 50 shows an example of a historical database that includes data references for total number of cases of measles in NYC (tuples t1, t4). Note that we use this and the following examples for illustration purposes only, and that they do not represent any actual disease occurrences. We cannot simply add the values of t1 and t4 to find the total number of cases of measles, as t1 and t4 have overlapping time intervals. As a result, there is a *temporal conflict* between t1 and t4. Moreover, although the time intervals of t2 and t3 do not overlap, we cannot simply add up their corresponding data values to obtain the total number of smallpox cases in the state of New York. Tuple t2 refers to the total number of smallpox cases reported for the state of New York. Meanwhile, it is unknown if this tuple includes all New York City cases reported in t3. There is a *spatial conflict* between t2 and t3. In this section, we focus on handling this type of temporal conflict.

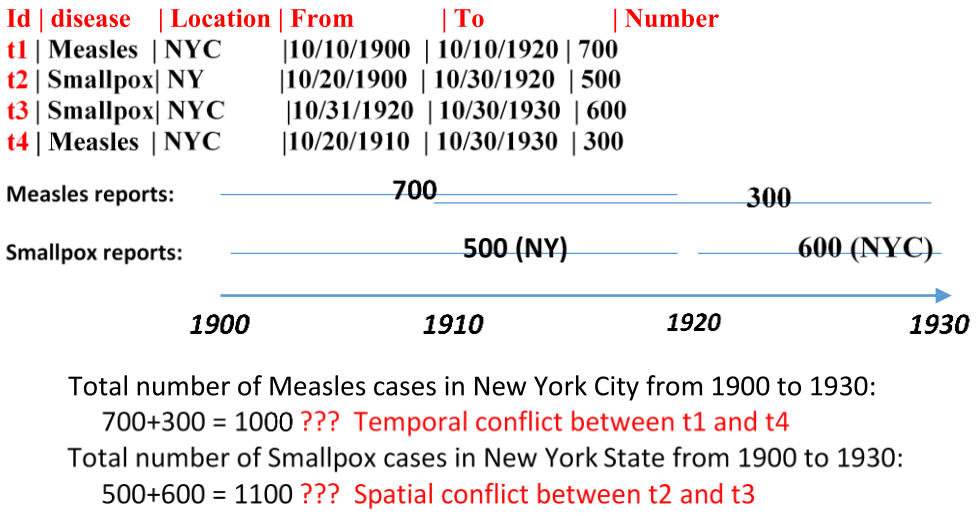


Figure 50. An example of a temporal and spatial conflict.

A redundant historical database includes conflicting tuples. As a result, there is a risk of double-counting and over-estimating the value of the aggregate data. At the same time, non-redundant databases ignore conflicting tuples and only account for non-overlapping reports, which may result in notable underestimations of the aggregate value. Consider again the conflicting measles reports from Figure 50. Figure 51 refines the impact on the estimation accuracy of ignoring the conflict (keeping both reports) and enforced non-redundancy (excluding one of the reports from consideration). Counting both of them will overestimate the actual number of measles cases for the period of 10/10/1900–10/30/1930. Ignoring report R1 will leave the cases between 10/10/1900 and 10/20/1910 unreported (uncovered) and thus will underestimate the actual number. Ignoring report R2 will underestimate the actual number by excluding the cases between 10/10/1920 and 10/30/1930 from consideration.

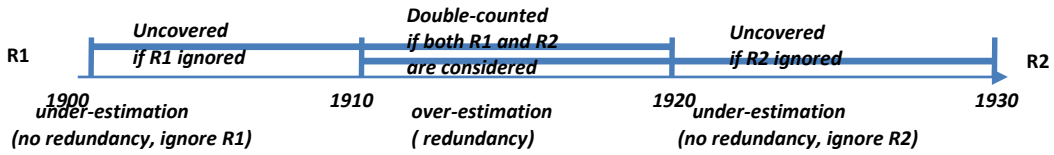


Figure 51. Effects of redundancy and enforced non-redundancy

6.4 PHASE IV: EVENT TYPE GENERATION – THE DEGREE OF CONFLICT

In the previous section, we have reviewed the issues of redundant historical databases. The PDW at this stage is intended to generate event type and in this study, the event type is defined as the severities of redundancy. Therefore, we introduced the concept of degrees of conflict (CD) to assess the risk of erroneous values reported in conflicting historical tuples. Without losing generality, we will represent conflicting tuples as triples (From, To, Value), where Value is a historical statistic (number of events) reported within the [From, To] time interval.

Figure 52 shows two scenarios for redundant databases DB1 and DB2. Here, we show only the time intervals of conflicting tuples, annotated with corresponding reported values. Consider Scenario (a) first. In both cases of redundant databases, the time overlap is equal, but the relative contributions of conflicting tuples in the aggregated total value estimate are different. Both DB1 and DB2 can be split into two non-redundant snapshots that include only one of the conflicting tuples. The total value estimated over non-redundant snapshots of DB1 is 100 in both cases, while the corresponding values estimated over non-redundant snapshots of DB2 are 10 and 100. For DB2, the difference between the non-redundant estimates is greater, and as a result, we conclude that the degree of conflict between tuples in DB1 is higher than the degree of conflict between tuples in DB2.

Now consider Scenario (b), where the time overlaps of conflicting tuples are different while the reported values are equal. The difference between total values, which is estimated for non-redundant database snapshots, is the same in both cases. Meanwhile, it is safer to assume that the

risk of double counting is higher in the case of conflicting tuples with a larger time overlap (DB1). Thus, we expect to see that the degree of conflict is higher for DB1.

In summary, we define a measure of the degree of conflict between two historical tuples with the following characteristics:

- The conflict degree ranges within a [0,1] interval;
- Tuples with non-overlapping time intervals have a degree of conflict of 0;
- Tuples with overlapping time intervals have a higher degree of conflict if their reported values are closer to each other (namely, the relative contributions of the reports are comparable).
- Tuples with comparable reporting values have a higher degree of conflict if their time overlap is higher.

We also offer some alternate definitions of the measure of the degree of conflict. For example, a relative contribution of two conflicting tuples can be defined in such a way that closer values reduce the conflict. The expected fusion strategy in this case would favor reports that have similar event distributions.

Figure 52. A visual explanation of degrees of conflict

Below, we define the conflict degree measure reflecting those characteristics. Informally, the conflict degree should account for both relative contributions and time overlaps of the conflicting tuples. We define a relative contribution of two conflicting tuples r1 = (F1,T1,V1) and r2 = (F2,T2,V2) as

$$RC(r1,r2) = 1 - |V1-V2|/(V1+V2).$$

Here, we assume that at least one of the two variables V1 and V2 is non-zero. For example, consider Scenario (a) in Figure 52. The relative contribution of conflicting tuples in DB1 is 1 - |100-100|/100+100 = 1, while the relative contribution of conflicting tuples in DB2 is 1 - |10-100|/100+100 = 0.45.

In order to define a relative time overlap of two conflicting tuples, we introduce several operations similar to some of the interval operations from [14]. The length of time interval t = [F,T] (denoted |t|) is the difference T-F plus one, i.e. |t| = T-F+1. The unit time interval 1 is a time interval whose length is equal to one: |1| = 1 (namely, the unit time interval has equal From and To components). As a result, |t| is a number of unit time intervals that are covered by t. The time interval t = [F,T] is consistent if F ≤ T. Below, we consider consistent time intervals only if not stated otherwise. The time intervals t1 = [F1,T1] and t2 = [F2, T2] overlap if F1 ≤ F2 ≤ T1. For two time intervals t1 = [F1,T1] and t2 = [F2, T2], we define their sum t1+t2, and intersection t1 o t2, as follows:

$$t1 + t2 = [\min(F1,F2), \max(T1,T2)]; \quad t1 \text{ o } t2 = [\max(F1,F2), \min(T1,T2)].$$

Note that the result of t1 o t2 may be an inconsistent time interval with a zero or negative intersection length. This would indicate that there is a gap between the time intervals t1 and t2. We will use this feature to define relative overlap RO of two consistent time intervals t1 = [F1,T1] and t2 = [F2, T2] as follows:

$$RO(t1, t2) = \max(|t1 \text{ o } t2|/|t1+t2|, 0),$$

i.e., RO(t1, t2) = 0 means that t1 and t2 do not overlap. Figure 53 illustrates the introduced time interval operations.

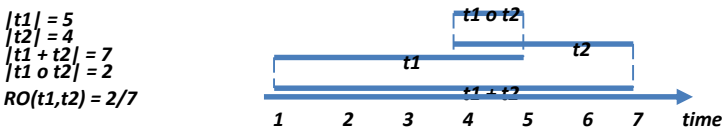


Figure 53. Time interval operations

We define the relative time overlap RO(r1,r2) of two historical tuples r1 and r2 to be equal to the relative overlap of their corresponding time intervals. Next, we define the conflict degree CD(r1,r2) of two historical tuples r1 and r2 as the following exponential function:

$$CD(r1, r2) = RO(r1, r2) \times e^{k(1-RC(r1,r2))(RO(r1,r2)-1)}$$

Assuming that r1 and r2 are known from their context, we can specify the CD in a more compact format:

$$CD = RO \times e^{k(1-RC)(RO-1)}$$

This definition captures the desired characteristics of a CD measure. Figure 54 illustrates the behavior of the CD function as we change RO, RC and k. In general, a higher value of RO implies a higher CD. Meanwhile, the lower value of RC slows down the rate at which the CD grows as the RO increases. A higher value of k amplifies the impact of RC on the CD value.

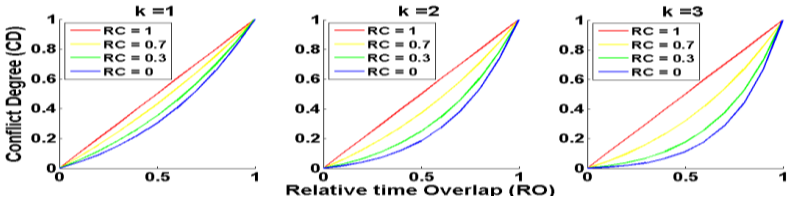


Figure 54. Behavior of CD measure

6.5 PHASE V: DECISION MAKING – CONFLICT AWARE DATA FUSION

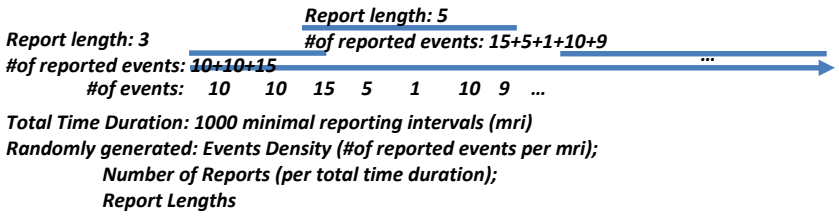
Section 6.4 introduced the method of estimating conflict degree (CD). By using the CD measure, we can assess a redundant database with respect to degrees of conflict between its tuples. Moreover, the CD measure allows us to better aggregate tuples by performing *conflict-aware data fusion*. The idea is to set up a target database that can tolerate a certain value of the degrees of conflict (CD threshold) between its tuples. The expectation is that we can define an optimal CD threshold value that will minimize misestimation errors that are caused by underestimations

due to uncounted events, as well as any overestimations due to double counting. Next, we will elaborate on characteristics of conflict-aware historical data fusion. The questions that we will address are: (1) how does conflict-aware data fusion perform under different application constraints; (2) is there a practical way to find an optimal conflict threshold that minimizes data fusion misestimation errors?

6.5.1 Characteristics of Conflict-Aware Data Fusion

In order to explore the characteristics of conflict-aware data fusion techniques in the context of the aforementioned questions, we have performed a simulation-based study.

Figure 55 summarizes the simulation set-up. First, we randomly generated different numbers of events of interest, per recording interval, within a reasonably large span of time. We specify the time duration in units of minimal recording intervals. The size of the minimal recording interval (such as day, week, month, or another interval) does not impact the results of our study. Next, we look at a maximum time duration of 1000 minimal recording intervals. The number of events can be reflected in multiple reports. Each report collects the number of events within a certain time interval (report duration or length).



Scenario	Expected Events Density	Expected Number of Reports	Expected Report Lengths
Few short reports on sparse events	20	20	20
Few long reports on sparse events	20	20	100
Many short reports on sparse events	20	100	20
Many long reports on sparse events	20	100	100
Few short reports on dense events	100	20	20
Few long reports on dense events	100	20	100
Many short reports on dense events	100	100	20
Many long reports on dense events	100	100	100

Figure 55. Simulation setup and scenarios.

We assume that each event corresponds to the same data reference; namely, that overlapping reports result in conflicting tuples. We used normal distributions to configure: (1) the number of events of interest per minimal recording interval; (2) number of reports per total time duration; and (3) report durations (lengths). We selected several significantly different configurations to explore various real-life scenarios. They are summarized in the lower section of

Figure 55, with corresponding expected values (*mu*) of the simulated parameters. We expect that many lengthy reports on densely populated events would result in a large number of high-degree conflicts. Meanwhile, a few short reports on sparsely populated events would hardly produce any conflicts at all. The expectation is that a *proper data fusion strategy would combine information from multiple reports, which would minimize the misestimation of the actual number of events within the time duration.*

We performed simulations changing the acceptable CD threshold within the [0, 1] range with a step of 0.01. We aggregated only reports with a conflict degree below the CD threshold value and calculated the misestimation error in each case. We performed multiple simulations for every combination of simulation scenarios and CD thresholds.

Figure 56 plots the CD threshold versus the absolute value of the relative errors in order to clarify how exactly they are related. For all scenarios, we observe similar error dynamics: the errors decrease before some *critical value* of the CD threshold, and after that, the errors increase. This is an expected behavior: the initial decrease in errors is due to increased event coverage from a larger number of aggregated reports and, as a consequence, reduced underestimation errors. Meanwhile, after the critical CD threshold value is reached, the overlapping reports start accumulating overestimation errors, due to double-counting. Here, we make another important observation: *each scenario is associated with an optimal CD threshold that minimizes the misestimation errors*. This also provides inspiration for a feasible way to estimate an optimal CD threshold, as we explain next.

6.5.2 Estimating an Optimal CD Threshold

Let us repeat two important observations noted above: (1) report density has a higher impact on error dynamics than the event density; (2) each scenario is associated with an optimal CD threshold that minimizes the misestimation errors. In real life, we do not have information about actual numbers of events per reporting interval (the actual event density). The information available in historical databases includes only the reported event numbers from a set of potentially conflicting reports. The challenge is to define the optimal CD threshold for each group of conflicting reports that would minimize the misestimation errors when we have no knowledge of the actual numbers of events.

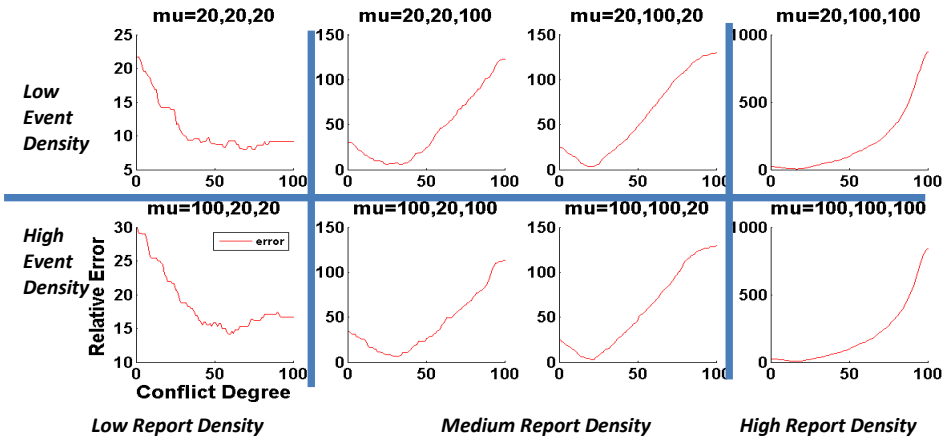


Figure 56. The impact of the CD threshold on error dynamics.

6.5.3 Conflict-Aware Data Warehousing

In the previous section (Section 6.5), we suggested a simulation-based approach to estimate an optimal CD threshold value for groups of overlapping reports, when there is no knowledge of actual event distributions. This estimation can be performed periodically as a part of the data warehouse loading procedure for each data reference in the integrated historical database. The value of the estimated optimal CD threshold can be used to maximize the accuracy of the query results. Figure 57 illustrates this approach. Here, we considered two groups of conflicting reports on cases of measles and smallpox in New York City. For each of those groups, we evaluated an optimal CD that minimizes misestimation errors of data aggregation. The value of the estimated optimal CD threshold was used to maximize the accuracy of the query results.

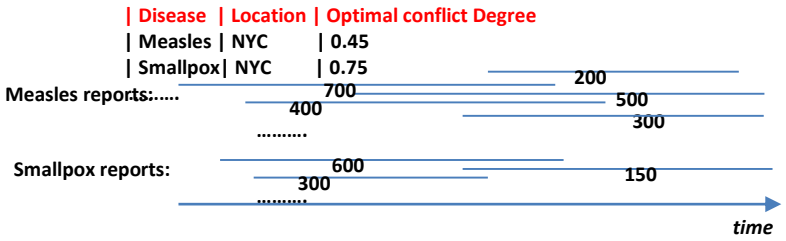


Figure 57. Estimating the optimal CD in an integrated data warehouse.

We implemented and tested our approach in Tycho, a large integrated epidemiological data warehouse that includes historical data from numerous heterogeneous sources. As a part of our ongoing efforts, we have completed the first stage of Tycho data integration, including heterogeneity resolution and data validation. We have integrated information from approximately 50,000 reports on United States epidemiological data over more than 100 years. The original reports are digitized and represented as semi-structured Excel spreadsheets with heterogeneous data formats and multiple transcription errors. After performing the heterogeneity resolution, we loaded more than 26 million records into the Tycho data warehouse. Next, we performed an aggregate assessment of conflicting data in the data warehouse. For this assessment, we considered only temporal conflicts. We evaluated the numbers of conflicting reports for each time span, and we observed up to 32 conflicts that occurred over the time spans of days, weeks, months, and years.

Figure 58 shows the result of conflict-aware data fusion for a representative set of Tycho reports. We selected three hundred Tycho data references that included up to 32 conflicts, determined the corresponding optimal CD threshold, and estimated error reduction, due to conflict awareness. We observe an error reduction rate of up to sixty percent—a notable improvement in the accuracy of estimation.

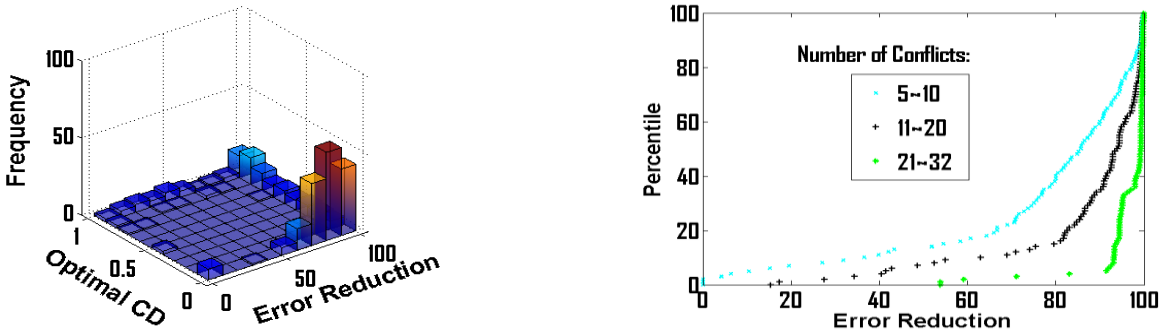


Figure 58. Conflict-aware error reduction in Tycho.

6.6 SUMMARY OF THE HISTORICAL EPIDEMIOLOGICAL CASE STUDY

Historical data reports on numerous events with overlapping time intervals, locations, and names. As a result, it may include severe data conflicts caused by database redundancy that prevent researchers from obtaining the correct answers to queries in an integrated historical database. In this case study, we elaborated our PDW to a large-scale heterogeneous historical data, which reports on numerous events for overlapping time intervals, locations, and names. We first developed several data preprocessing algorithms, which include data extracting, transforming, loading, error fixing, and semantics validation. The redundant data may include severe data conflicts caused by database redundancies, which can prevent researchers from obtaining the correct answers to queries in an integrated historical database. We introduced the conflict-aware data fusion strategy to deal with the redundant data issue in our PDW and demonstrated that our approach significantly reduces errors in data aggregation. Furthermore, we evaluated our approach by working with a Tycho data warehouse that integrates historical data from approximately 50,000 reports on US epidemiological data for more than 100 years. We demonstrate that our approach significantly reduces data aggregation errors in the integrated historical database.

7 CASE STUDY III: STATE SEQUENCE RECOVERY (SSR)

In the first part of our case study (Chapter 5), we examined medical data. That data was accurate since it was typically collected from reliable data sources (such as medical monitors). The data could be collected within different time intervals and with different levels of granularity. We decreased the granularity of the individual values by grouping the data items (vital signs) into a number of meaningful, application-dependent states. This state generation can either be done automatically using data clustering techniques, or through embedded mapping rules that are specified by an application (domain) expert. In our case, we used the latter method, since these value ranges were standard and commonly used across many fields in the medical domain. We have further introduced the concept of events, which are defined as a sequence of states.

In many other domains, the data may have such issues, as conflicts, duplicates, and missing values, and it must be cleaned before utilization. For instance, historical data reports on numerous events for overlapping time intervals may have data conflicts caused by database redundancy. These conflicts can prevent researchers from obtaining the correct answers from data. In the second part of our study (Section 6), we investigated possible solutions to those issues. We explored two major types of data conflicts: *temporal conflicts* and *spatial conflicts*, both of which may occur between the historical tuples. Figure 50 in Section 6.3 illustrates an example of both types of conflict. It shows, for example, that we cannot obtain the total number of cases of measles in New York City (tuples t1, t4) by simply adding the values in those two tuples. Moreover, summing up two smallpox related tuples (tuple t2, t3) may also cause a spatial conflict, since it is unknown whether all New York City cases were reported in t3.

To sum up, we examined UPMC medical data, which has been collected in different time intervals and may include sparse information. In particular, we tried to create an early warning system from real-time medical data by analyzing the sequence data of states and events. In Section 5, we dealt with historical epidemiological data that has been integrated from multiple

data sources for over 100 years. This data involves considerable redundancy (such as overlapping reports). We applied the concept of optimal degrees of conflict (CD) to minimize the misestimation of the values reported in conflicting historical tuples. In other words, the medical data values are considered to be reliable, and are available for individual time intervals (for example, hourly data). Our state generation is based on these raw medical values. Our objective is to better use those states to build a medical early warning system. For epidemiological data, we cannot always have the accurate values for either individual time intervals (such as weekly reports) or time spans (such as monthly reports) due to missing values or temporal conflicts (redundancy issues). Our goal is to minimize the misestimation of the values in those time spans.

In this chapter, we try to integrate works from the two previous studies. The idea is to adopt the concept of state sequences from Section 4 and embed it into the redundant data environment of Section 6. In general, it is difficult to estimate an accurate individual value within a given time interval. Therefore, instead of estimating individual values, our goal is to estimate the states (range of values) for certain time intervals. Figure 59 illustrates this approach. It shows a sequence of overlapping historical reports with total reported values for the reported time intervals. Based on a state mapping rule, the original values (OV) can be classified into their actual mapping states (AMS). Similarly, we translate the estimated values (EV) into the estimated states (ES). Consider specific examples of accurate state estimations, as well as misestimations, shown in Figure 59. With individual value comparisons, there are considerable errors (when looking at the difference between OV and EV, only the blue color is correct), while the state estimation may be accurate (the AMS and ES match). Here, we report only three misestimations (shown in red). We define this process of state estimations as state sequence recovery (SSR).

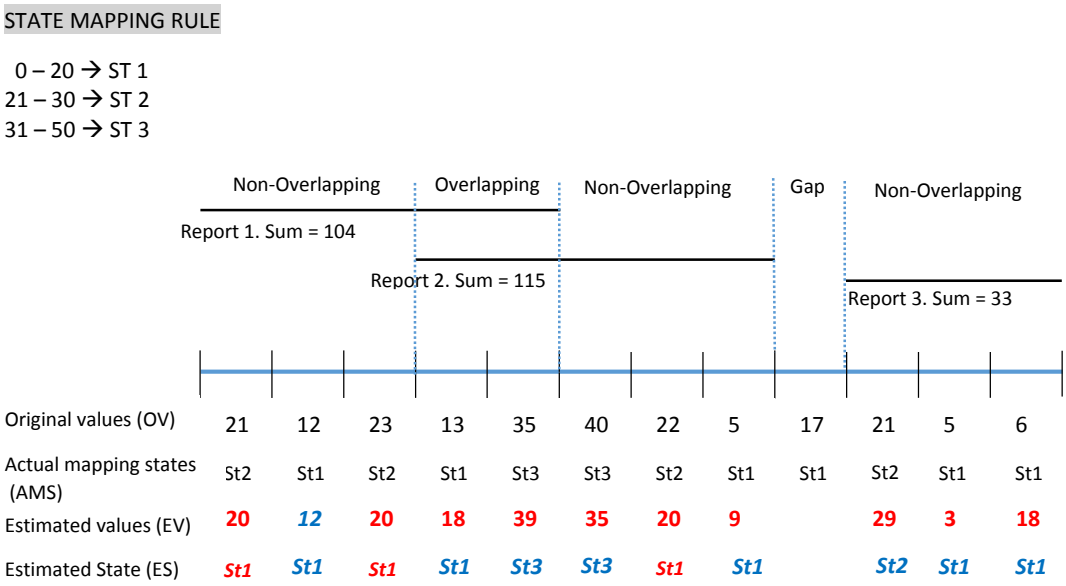


Figure 59. Characteristics of state sequence recovery.

This section is organized as follows. In Section 7.1, we propose and compare various state sequence recovery (SSR) methods, which minimize the differences between the actual mapping state (AMS) sequences and estimated state (ES) sequences. The experimental results are reported in Section 7.2. The conclusion of this study is provided in Section 7.3.

7.1 THE ALGORITHMS

7.1.1 Basic Algorithm (BA)

First, I propose a basic algorithm (BA) that generates the average value for each report, calculates the average value at each time interval, and translates the values to states using mapping rules. For example, consider the scenario of sample report distribution shown in Table 22.

Table 22. Sample report distribution

Report ID	From	To	Value
R_1	1	7	35
R_2	4	9	24
R_3	7	12	42

Table 23 illustrates that the BA (horizontally) evenly distributes values to individual time intervals for each report (R_1, R_2, and R_3) and (vertically) calculates the average for each time interval (t_1, t_2 ... and t_{12}). The BA maps those values to result states (S_1 to S_4) sequence using the following mapping rules. The BA algorithm is presented in Figure 60.

Table 23. The BA procedure

Mapping rules
$0 \leq S_1 \leq 5$,
$5 < S_2 \leq 10$,
$10 < S_3 \leq 15$,
$15 < S_4 \leq 20$

Time Interval (t) / Report ID	1	2	3	4	5	6	7	8	9	10	11	12
R_1	5	5	5	5	5	5	5					
R_2				4	4	4	4	4	4			
R_3							7	7	7	7	7	7
Average (R_1, R_3, R_3)	5	5	5	4.5	4.5	4.5	5.3	5.5	5.5	7	7	7
Result State Sequence	St_1	St_1	St_1	St_1	St_1	St_1	St_1	St_1	St_1	St_2	St_2	St_2

Basic Algorithm (BA)

Input:
N: number of reports
$Reports\ Set$: $R = \{r_1, r_2, r_3 \ldots . r_N\}$; Where each report r_N consists of *(F, T, V)*. *F* is the beginning time stamp (From), *T* is ending time stamp (To), and *V* is the total reported value for the time period from F to T.

Time stamps: TS = $\{ts_1, ts_2, ts_3 \ldots . ts_n\}$; where ts_1 = minimum of all F values from report set R and tsn = maximum value of all T values from report set R

Output: $S = \{S_1, S_2, S_3 \ldots . S_n\}$; where S is the result of the state sequence.

Procedure:
While $N \neq 0$ **do**
For each report r_N

$r_{Nn} = T_N - F_N + 1$ R_{Nn} is the number of time intervals for the given report (r_N)
$r_{BA} = \frac{r_{Nv}}{r_{Nn}}$ // r_{BA} is the result value for each time interval for the given report (r_N)
End For

For each Time Interval ts_n

$On = numberOfOverlaping(t_{sn})$ // find reports that overlap for each time interval
$Sn = (\frac{Sum(rSn)}{On})$
End For

End do

Figure 60. The BA algorithm

7.1.2 Linear System (LS)

The linear system is another approach to solve the SSR problem for several overlapping reports. It was proposed by Pei-Ju Lee, who is a PhD student in our group [26]. To demonstrate the linear system (LS) for the SSR problem, we continue to use **Table 23** from Section 7.1.1 as an example. Figure 61 visually represents those three reports as overlapping and split the time line into five time spans (X_1 to X_5).

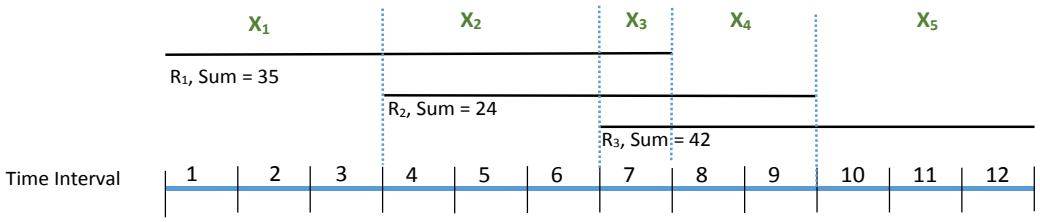


Figure 61. An illustration of overlapping reports.

Max. $X_1 + X_2 + X_3 + X_4 + X_5$

Subject to:
$X_1 + X_2 = R_1$
$X_2 + X_3 + X_4 = R_2$
$X_4 + X_5 = R_3$
$X_1, X_2, X_3, X_4, X_5 \geq 0$

Figure 62. Example of linear system

The reports can be formatted to a linear system, as shown in Figure 62, and can be further summarized in a matrix equation as:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \\ X5 \end{bmatrix} = \begin{bmatrix} R1 \\ R2 \\ R3 \end{bmatrix}$$

We use the non-negative least square method to solve linear equations $\boldsymbol{AX} = \boldsymbol{b}$, where $\boldsymbol{X}$ can be computed as $\boldsymbol{X} = \boldsymbol{A^T}(\boldsymbol{AA^T})^{-1}\boldsymbol{b}$ [26]. Figure 63 shows the result of time spans (X_1 to X_5) using the linear system. We then evenly distribute those time span values to values of individual time intervals and translate them to the state, using the mapping rule shown in Section 7.1.1. For example, the state sequence of time interval 1 to time interval 3 is calculated by $X_1 / 3 = 3.6$ and is equal to the state 1(St_1). Meanwhile, other time intervals can be derived in the same fashion.

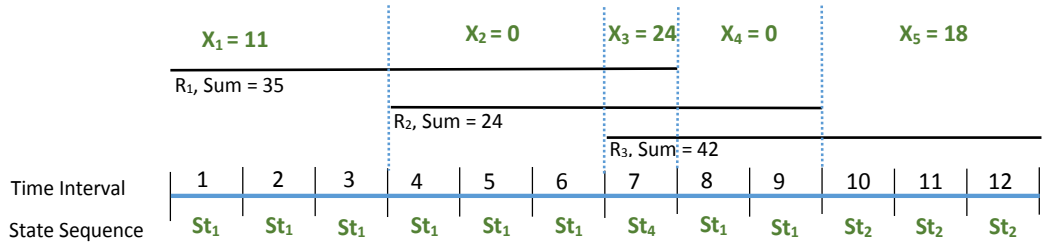


Figure 63. The result of the application of the linear system.

7.1.3 Genetic Algorithm (GA)

The genetic algorithm (GA) is a search algorithm based on the theory of natural selection, along with the process of biological evolution. The idea of using the GA to find the optimal solution is to continually modify individuals in the population (chromosomes) until the algorithm stopping criteria are reached. The results from the use of the GA may not be the best solution to a given problem, but it is considered as an optimal solution based on certain constrains (e.g. time and computation resources) [27]. In other words, the population evolves to optimal solution; when it stops, it is considered to be the successive generations. In general, there are three essential rules (steps) in GA toward next generation from the current population. First, the selection rule selects the individuals called the parents. Second, the crossover rules are applied, which combine two parents to produce children for the next generation. Third, the mutation rules apply random changes to individual parents to form children. These three rules interact with the object (fitness) function to confirm that successive generations are better than their parents, and this procedure is called an elitist selection in the genetic algorithm.

When we adopt the GA into our SSR, our goal is to find an optimal state sequence from given reports. Our assumption is that when an optimal solution is found for the fitness function, the optimal state sequence is also derived. This process is same as the classic genetic algorithm problem, which is finding the optimal maximal or minimal value. In other words, if the ground truth state sequence matches our SSR state sequence, we consider this to be the perfect result. However, in our study, the ground truth is hidden unless we have reports for individual time intervals. Therefore, we use the BA's state sequence as the target to for the GA to reach (the

optimal result), which operates under the assumption that the BA's result is close to the true result. As we can see, the GA's process does not use report values like BA and LS. Instead, it can be considered to be an extension of the BA. Moreover, the optimal solution in GA is not necessarily equal to the best overall solution, since GA requires stopping criteria for computation efficiency purpose. As a result, there is a chance that the GA will stop before it reaches the best solution. For example, in our study we have a stopping condition of not exceeding the 100th generation. In the following paragraphs, we describe our GA approach step by step, with corresponding examples.

1. The fitness function is designed to evaluate the distance between two state sequences. We assume that the BA's result is much closer to the ground truth state sequence, and as a result, the GA's procedure tries to match the BA's state sequence of a particular report. We define our fitness function as:

$$\text{Min: } f(GA\ state\ seq) = \sum_{i=1}^{n} \text{abs}(\text{BA } st_i - \text{GA } st_i);$$

where n is the number of time stamps for a given report

For example, the BA generates a state sequence for a given report of {5, 5, 5, 5, 4}, and by using the BA's sequence as an object function, the GA could possibly generate a state sequence of {5, 3, 4, 6, 5}. Note that the GA's procedure involves a randomized process, so the result will not be the same every time. In this case, the score of the GA's result is abs(5-5) + abs(5-3) + abs(5-4) + abs(5-6) + abs(4-5) = 5.

2. The algorithm begins by generating a certain number of same-length state sequences (the initial population). This initial population is the first generation, and the GA uses the individuals in that generation to create the next population. From the example in step 1 where the sequence length is 5, we can set the population size to be 100, so the dimension of the population is a 100-by-5 matrix. Note that the same individual can appear in the population more than once.

3. The selection function scores each member of the current population by computing the fitness value and then chooses an elite (lower) fitness value for the next generation. Continuing the example from step 1, {5, 3, 4, 6, 5} has a value of 5 and {8, 4, 9, 10, 20} has a value of 29.

Obviously, the sequence {5, 3, 4, 6, 5} is better than {8, 4, 9, 10, 20} because of its lower fitness value. Those elite children are chosen to be passed on to the next population.

4. To produce new children and increase the diversity of selection from the elite children, GA uses both mutation and crossovers. We assume that the population diversity can increase the chances of finding the best solution. For mutation, children (the state sequence) are produced by making a random change from a single parent. For example, the sequence {5, 3, 4, 6, 5} can possibly generate another sequence {5, 3, 5, 6, 5} which further lowers (optimizes) the fitness value. For crossovers, the algorithm creates crossover children by combining pairs of parents in the current population. There are several classic crossover methods in the GA, which can be defined based on the study domain. In our case, we use the single point crossover option. For example, the sequences {5, 3, 4, 6, 5} and {6, 5, 5, 5, 5} can be combined as {6, 5, 4, 5, 5} and {5, 3, 5, 6, 5} by setting the third time interval as the crossover point.

5. So far, we have discussed how GA uses fitness function to find the optimal result, which in our case, matches the BA's result. However, GA can often perfectly match the BA's results, especially for less diverse data (such as a normal distribution) and with shorter reports. Meanwhile, the BA's result is an average state sequence that may not truly reflect the ground truth. Alternately, we can use the GA's byproduct, which is the population set after the GA's result matches the BA's result. Our final result is derived from the population set that chooses the mode from the population. For example, for a final population set of {3, 4, 5, 6, 2}, {5, 6, 3, 6, 4}, {5, 5, 5, 6, 5}, {4, 5, 4, 5, 5}, {5, 7, 5, 5, 4}, and {6, 5, 6, 6, 4}, we chose the most frequent (mode) state from each time interval, which provides the GA state sequence result as {5, 5, 5, 6, 4}.

We apply our GA approach to the same example, which is shown in **Table 23** in Section 7.1.1. BA generates the state sequence of {1,1,1,1,1,1,1} for the report 1 (R_1). The GA begins by generating a set of random sequence to create a population. The dimension of this population set is a 1000 x 7 matrix. The object function is set to find another sequence that has the smallest distance to the sequence {1,1,1,1,1,1,1} from the population. Thus, the 0 value in the object function means that the GA finds the result, and the GA process will terminate. If it does not find a completely matching sequence, the GA uses the mutation and crossover rules to generate a new

population for the next generation. In general, the population from the new generation (children) is supposed to be better than its previous generation. Given this assumption, we choose the most frequent states for each individual time interval from the population at the final generation. In this example, the GA may generate a more diverse state sequence for R_1, such as {1, 2, 1, 1, 2, 1, 1}. Through the same method, we may have R_2 and R_3 state sequences, as shown in Table 24. Finally, the estimated result sequence is derived from taking the vertically average state of those three reports.

Table 24. Example results from the GA

Time Interval (t) / Report ID	1	2	3	4	5	6	7	8	9	10	11	12
R_1 state sequence	1	2	1	1	2	1	1					
R_2 state sequence				1	2	3	3	1	4			
R_3 state sequence							2	3	2	1	4	3
Estimated result sequence	1	2	1	1	2	2	2	2	3	1	7	3

7.1.4 Bayes' Rule Approach (Bayesian)

Our Bayesian approach consists of two parts. The first part is using a maximum likelihood estimation (MLE) to estimate the unknown parameter (such as μ and σ) of the probability density function (PDF) of the working data. Second, we generate the probability model with the estimated parameters from the MLE and apply it to the Bayes' rule. As an example, we consider the normal distribution data to illustrate the first part of the MLE procedures:

1. The probability density function (PDF) of normal distribution is defined as:

$$f(x;\ \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\ exp^{-\left[\frac{(X_i-\mu)}{2\sigma^2}\right]}$$

where x is the reports, which are composed of an unknown event sequence, μ is the mean, and σ is the standard deviation. Since the report sum and report length are provided in our data, the mean μ can be calculated directly by $\frac{report\ sum}{report\ length}$.

2. The actual values of the individual time interval are unknown from our original reports. For each of those reports, we generate a reasonable number (1000 in our study) of random value sequences (samples) with the same length as the corresponding original reports. Those generated sample random value sequences must meet the constraint of

   SUM (sample value sequences) ≤ original report sum.

   In the next step, we use the MLE to estimate the original standard deviation (σ) from those sample reports.
3. The likelihood function is defined as:

$$L(\mu, \sigma) = \sigma^{-n}(2\pi)^{\frac{-n}{2}} exp^{\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right]}$$

   Upon maximizing it, the maximum likelihood estimator of the variance is:

$$\sigma^2 = \frac{1}{n}\sum_{x=1}^{n}(x_i - \mu)^2$$

   Since each sample report may have a different value of σ, we use the *mean value of* σ as the result.
4. After obtaining μ and σ, we use them to generate a new *event value sequence* and then further convert it into an *event state sequence*. We assume this event state sequence is much closer to the ground truth (the unknown original).

We repeat step 2 to 4 for all original reports. In other words, we generate all estimated *event state sequences* for those reports.

In the second part, after finding the estimated *event state sequence* for all reports, we apply the Bayes' rule to estimate the completed state sequence for all time intervals. Our approach with Bayes' rule is to estimate the state probability of a particular time interval (SPTI). It is defined as a conditional probability of $P(R|st)$.

$$SPTI = P(r_n \mid st) = \frac{P(st \mid r_n)\, P(r_n)}{P(st)}$$

Where R = $\{r_1, r_2, r_3 \ldots r_n\}$ represents all reports that covered a particular time interval. Other mathematical interpretations of these probabilities include:

1. $P(st \mid r_n)$: The likelihood. This is based on a higher degree of belief that one report has better representation for the particular time interval than other reports when multiple reports overlap a single time interval. At the same time, for non-overlapping time intervals, $P(r_n)$ is equal to 1, which means that the estimate of the state of a time interval only depends on that report itself.
2. $P(r_n)$: the prior probability of selection from a particular report.
3. $P(st)$: The state probability, which is the probability of a particular state for a time interval under all reports.
4. $P(r_n \mid st_i)$: The posterior, which is the degree of belief after accounting for the estimated state (st). We iterate all probability of $P(r_n \mid st_i)$ and find the most probable state, which is the $arg\,max(\,SPTI\,)$ of our result state for that time interval.

For example, Figure 64 consists of two overlapping reports R_1 and R_2. R_1 is from the time interval 1 to 5 with the MLE estimated state sequence of {5, 3, 3, 5, 3}, and R_2 is from the time interval 4 to 8 with the MLE estimated state sequence of {3, 8, 5, 6, 6}.

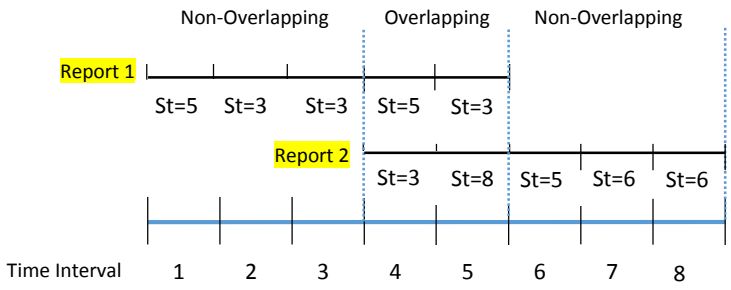


Figure 64. An example of time overlapping.

For these two reports, we can have the likelihood of $P(st_i \mid R_n)$ as:

	State (st)	Frequency	$P(st_i \mid R_n)$
R 1	3	3	3/5
R 1	5	2	2/5
R 2	3	1	1/5
R 2	5	1	1/5
R 2	6	2	2/5
R 2	8	1	1/5

And also have the state probability as:

State (st)	Frequency	$P(st_i)$
3	4	4/10
5	3	3/10
6	2	2/10
8	1	1/10

The time intervals of 4 and 5 are overlapped by both R_1 and R_2. Assuming that all reports are equally chosen, then the prior probability of $P(r_n)$ here is 0.5. Based on Bayes' rule above, we enumerate all possible posterior probabilities of $P(r_n \mid st)$ and choose the maximum value. For the overlapping time interval 4, we have:

$$\frac{P(st_5|r_1)\ x\ P(r_1)}{P(st_5)} = \frac{\frac{2}{5}x\frac{1}{2}}{\frac{3}{10}} = \frac{2}{3} \text{ vs } \frac{P(st_3|r_2)\ x\ P(r_2)}{P(st_3)} = \frac{\frac{1}{5}x\frac{1}{2}}{\frac{4}{10}} = \frac{1}{4}$$

As a result, we choose state 5 for time interval 4, and choose the same state for time interval 5 and have state 8 as the result. Those non-overlapping intervals are only calculated by single reports, so the result can be derived directly, such as the state sequence {5, 3, 3} for time intervals 1 to 3, respectively, and state sequence {5, 6, 6} for time intervals 6 to 8, respectively.

We consider using our Bayesian approach on the same example from Table 23, shown in Section 7.1.1. Figure 61 illustrates the reports that overlap for five time spans (X_1 to X_5) for the entire time interval. After the first part of the Bayesian approach, which generates sample state sequence for all reports, we may have something like Table 25.

Table 25. Example result from the first part of the Bayesian approach

Time Interval (t) / Report ID	1	2	3	4	5	6	7	8	9	10	11	12
R_1 state sequence	1	1	2	1	1	2	1					
R_2 state sequence				2	1	2	2	3	4			
R_3 state sequence							3	1	1	2	2	1

We use the time interval of 4 to 6 (overlapped by R_1 and R_2) as an example to illustrate the second part of the Bayesian approach, and we have the likelihood of $P(st_i \mid R_n)$ as:

	State (st)	Frequency	$P(st_i \mid R_n)$
R 1	1	5	5/7
R 1	2	2	2/7
R 2	1	1	1/6
R 2	2	3	3/6
R 2	3	1	1/6
R 2	4	1	1/6

We also have the stated probability as:

State (st)	Frequency	$P(st_i)$
1	6	6/13
2	5	5/13
3	1	1/13
4	1	1/13

We assume that all reports are equally chosen, so the prior probability $P(r_n)$ here is 0.5. Based on the Bayesian rule above, we enumerate all possible posterior probabilities $P(r_n \mid st)$ and choose the maximum one. For the overlapping time interval 4, we have

$$\frac{P(st_1|r_1)\ x\ P(r_1)}{P(st_1)} = \frac{\frac{5}{7} x \frac{1}{2}}{\frac{6}{13}} = 0.77 \text{ vs } \frac{P(st_2|r_2)\ x\ P(r_2)}{P(st_2)} = \frac{\frac{3}{6} x \frac{1}{2}}{\frac{5}{13}} = 0.65$$

As a result, we choose state 1 for time interval 4. We do the same for time intervals 5 and 6, and obtain states 1 and 2 as the result. As a result, we get the state sequence {1, 1, 2} for time intervals 4 to 6. Similarly, by repeating the above procedures, we may have the estimated result state sequence, as shown in Table 26.

Table 26. Example results from the Bayesian approach

Time Interval (t) / **Report ID**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
R_1 state sequence	1	1	2	1	1	2	1					
R_2 state sequence				2	1	2	2	3	4			
R_3 state sequence							3	1	1	2	2	1
Estimated result sequence	1	1	2	1	1	2	2	1	1	2	2	1

In this section, we introduced four state sequences recovery (SSR) algorithms and for illustration purposes, we applied them to the same example in Table 23 in Section 7.1.1. Table 27 shows the estimated state sequence result from those four algorithms. Note that besides the BA, other algorithms involve randomized procedures, and the result may not be the same for each iteration.

Meanwhile, in our scenario, we only have the total value for certain time spans (reports) and those reports are redundant (overlapping) for individual time intervals. Thus, in the next section, we will explore the performance of each of the proposed methods through a simulation-based study. Since this is the simulation test, we can obtain an accurate assessment using the pre-generated ground values.

Table 27. Example results from four algorithms

Time Interval (t) / Report ID	1	2	3	4	5	6	7	8	9	10	11	12
BA estimated state sequence	1	1	1	1	1	1	1	1	1	2	2	2
LS estimated state sequence	1	1	1	1	1	1	4	1	1	2	2	2
GA estimated state sequence	1	2	1	1	2	2	2	2	3	1	7	3
Bayesian estimated result sequence	1	1	2	1	1	2	2	1	1	2	2	1

7.2 EXPERIMENTS: THE STATE SEQUENCE RECOVERY (SSR)

In this section, we demonstrate the experimental results of algorithms that we described in Section 7.1. To ensure our study is applicable in different domains, we continue to use the 8 simulation set-up scenarios from
Figure 55 at Section 6.5.1. In order to test our approach in a more challenging environment, we will add exponential data distribution. In our tests, we estimate the accuracy using an average error rate, which is calculated by:

$$Avgrage\ Error = Avg\left[\frac{abs(ground\ truth\ state - estimate\ state)}{\max(ground\ truth\ state)}\right]$$

We also use the misestimation ratio to facilitate the error dynamic of average error, which is calculated by:

$$Misestimation\ Ratio = \left[\frac{number\ of\ misestimations}{\text{number of estimations}}\right]$$

This misestimation ratio is counted as long as the estimated values or states are different from the ground truth values or states.

In order to explore how the state sequence recovery methods behave, we detail our experimental tests in the following sections. In section 7.2.1, instead of directly working on state-level data, we perform a raw value estimation study that discusses the characteristics of each

algorithm in different simulation set-up scenarios under both normal and exponential data distributions. Note that the raw value is the finest data granularity, and in this case, we can consider it to have the maximum number of states. In Section 7.2.2, we decrease data granularity by grouping the raw values to create a lower number of states (coarse granularity), as we have done in our medical study section to discover the dynamic of the changes in accuracy. In Section 7.2.3, we adopt the concept of optimal CD from our previous study and show the ways in which optimal CD can minimize misestimations. Other than discussing the estimation of raw values or the state of individual time intervals, in Section 0, we assume that we have additional information from other heterogeneous data sources and that our algorithms can use this additional information to better evaluate the sequence data.

7.2.1 Accuracy Estimation for Raw Data (Finest Granularity)

Figure 65 plots the result of the average errors of the BA and GA under the normal distribution data environment. We can make the following important observations from this result:

1. The BA and GA show similar behavior (curves) in most cases, with respect to having a benefit (lower average error) from certain redundant degree settings.
2. The BA clearly shows fewer average errors among all scenarios, which meets our expectations, because the BA is a conservative and direct approach that should behave well when the given data is less dynamic (as in a normal distribution).
3. The GA is a sophisticated approach that uses the concept of a genetic algorithm. In Section 7.1.3, we described the GA in detail and showed how the GA improves from the BA's prediction result by increasing its overall variability. In this experiment, although we do not see the GA perform better than the BA (except for an instance shown in scenarios (20, 20,100)), it does show a notable curve improving pattern for scenarios (20, 20,100) and (20,100,100), where the BA is vice versa.
4. The (20,100,100) and (20, 20, 100) scenarios are those that have many long reports with sparse events (lower value). In this case, the prediction result from the BA usually assigns the lowest state value to all time intervals, which show less deviation from others.

5. We also observe that the report density has a higher impact on the error dynamics than the event density. This is consistent with the results from our conflict-aware data fusion study.

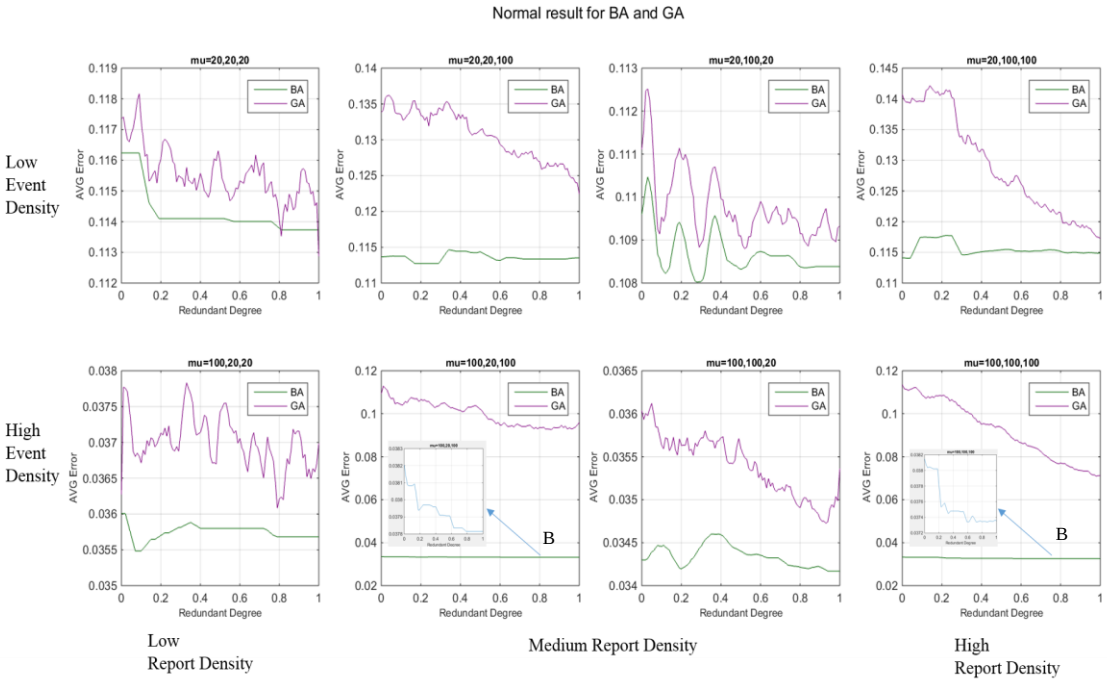


Figure 65: Normal results for the BA and GA

Figure 66 plots the result of average errors of the LS and Bayesian distributions under the normal distribution data environment. We can make the following important observations:

1. The LS and Bayesian distributions do not benefit from report redundancy in all scenarios. In general, they all show increasing errors while the degree of redundancy increases.
2. LS has a much higher average rate of error in all cases, which significantly increases the error rates in the beginning, though it behaves much steadily after reaching some critical values. For a better explanation of this phenomenon, consider the example from Figure 63 in Section 7.1.2. As we can see, the LS may assign a zero value to

many individual time intervals (or time spans) in order to find the optimal solution. This makes the estimated state to be the minimal state (namely, state 1 in our case), which is usually against the ground truth. This will result in a much higher misestimation.

3. Unlike LS, the Bayesian distribution has a much lower average error rate in several cases, and gradually increases the average error rate. As we recall from Section 7.1.4, we have the constraint of the total sum for the random sample reports, which must be less than or equal to the original report's total sum. This ensures that the generated random samples are not too different from the original report; however, those sample reports are still far from the ground truth. As a result, more reports (increasing the redundant degree) will generate more misestimation errors.
4. The (20,100,100) and (100,100,100) sets are high report density scenarios with many long reports. In this case, it is very difficult to generate appropriate sample state sequences (reports) for the Bayesian approach, and as a result, it shows significant error increases. Similarly, LS also suffers from those 2 scenarios and it also shows highest average error among others. Especially, in (100,100,100) scenario, we find LS has some average errors of more than 1, which is due to the estimations values from LS; those values are higher than the ground truth maximum value (numerator greater than denominator).

5. We also observe that the report density has a higher impact on the error dynamics than the event density, which is also consistent with the results from the conflict-aware data fusion study.

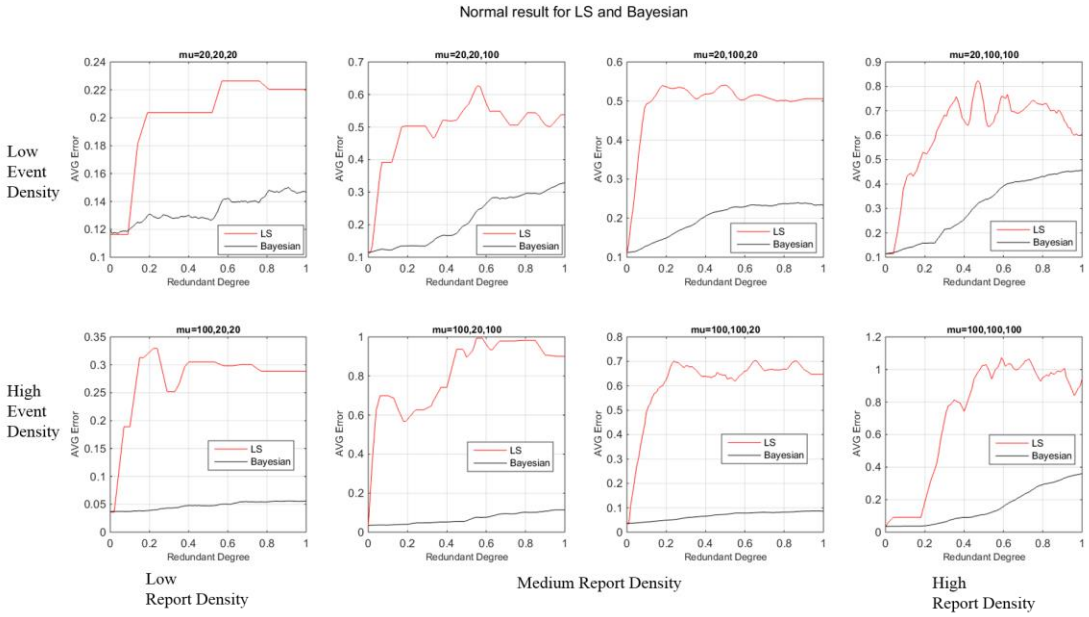


Figure 66: Normal results for LS and Bayesian distributions

Figure 67 plots the result of the average error rate of the BA and GA under the exponential distribution data environment. We make the following observations:

1. The BA and GA show similar behavior (curve) in most cases, except the (20,100,100) scenario.
2. Unlike the normal distribution comparison, in short report-length scenarios, the GA performs better than the BA in some redundant degree settings, although this finding is not at all obvious.
3. When compared to a normal distribution test, the BA and GA both show a higher average error rate in this exponential distribution among all scenarios.

4. We also observe that the report density has a higher impact on the error dynamics (in the similarity of the cure pattern), rather than the event density. This is consistent with the results from our conflict-aware data fusion study.

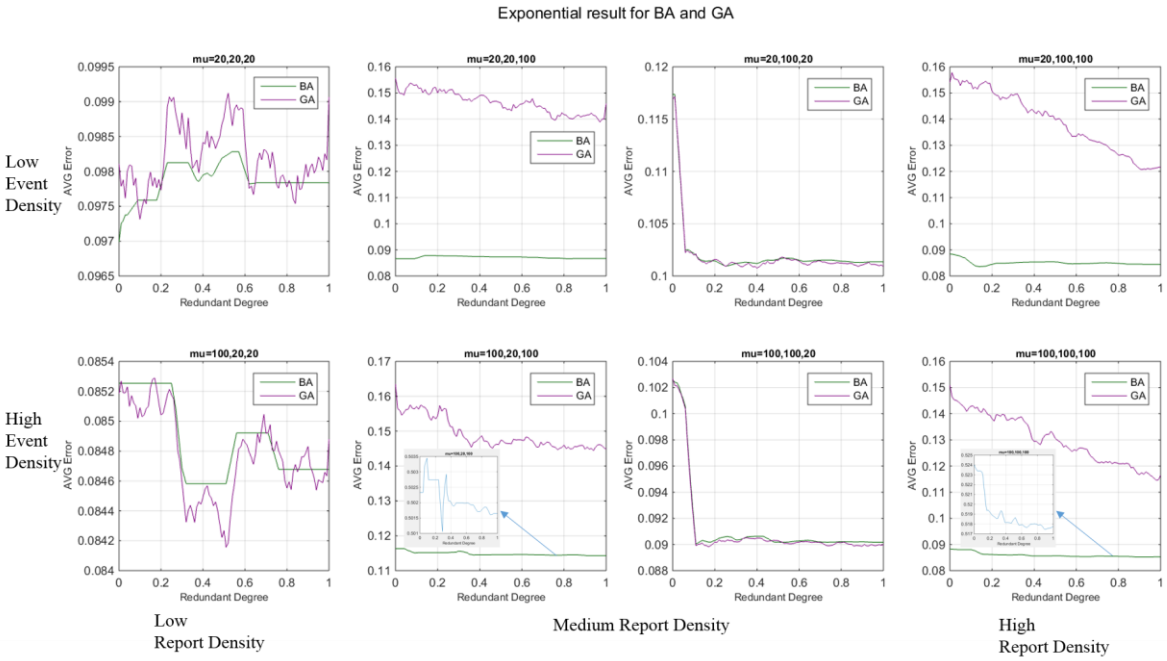


Figure 67: Exponential results for the BA and GA

Figure 68 plots the result of average error of the LS and Bayesian distributions under the exponential data distribution environment. There are several important observations that can be made from the figure:

1. Similar to the normal distribution data environment, the LS and Bayesian distributions do not benefit from report redundancy in all scenarios. All of them show errors increasing while the degree of redundancy increases.
2. In the beginning, the Bayesian distribution has a lower average error rate than that of the LS distribution. However, when the redundancy degree reaches around 0.3 to 0.4, the Bayesian distribution starts to significantly and more steadily increase the error

rate. This finding is consistent with the result from the normal distribution, except that the LS distribution has better performance than the Bayesian in this situation.

3. Unlike other algorithms, the LS distribution has a noticeable accuracy improvement in the exponential data environment, rather than in the normal data environment. In the meantime, the Bayesian distribution does not show a lower accuracy in the exponential data environment. This is because of the challenges in generating a sample that is similar to the original reports in the exponential data environment.
4. We also observe that the report density has a higher impact on the error dynamics than the event density. This is also consistent with our study results from the conflict-aware data fusion study.

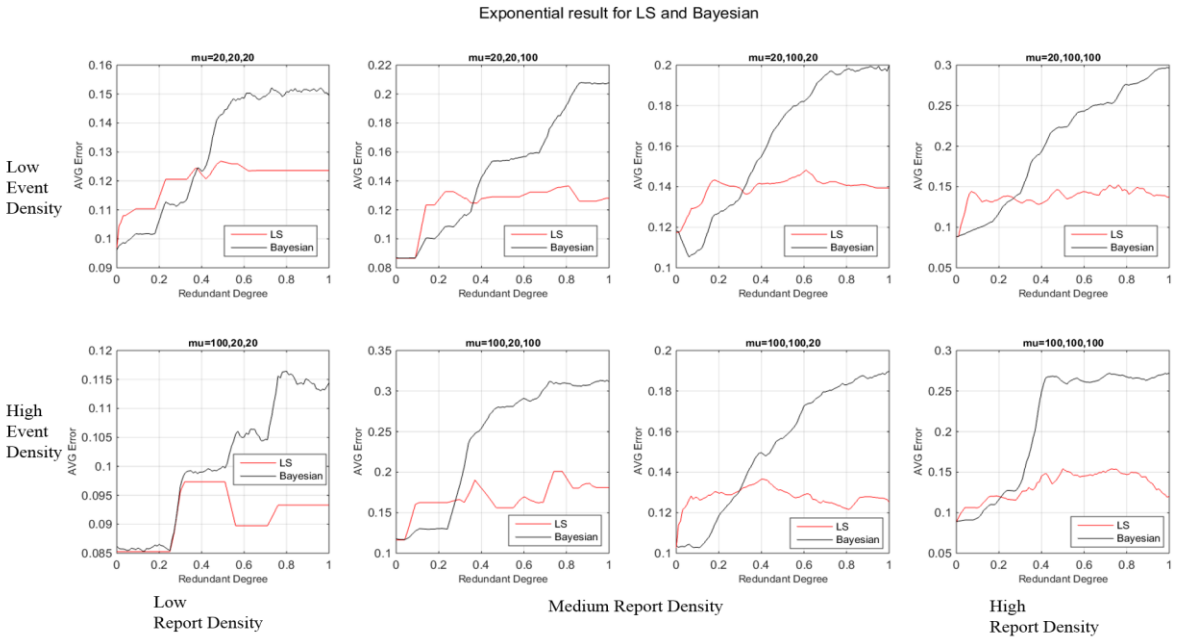


Figure 68: Exponential results for the LS and Bayesian distributions

7.2.2 Accuracy Estimation for State-Type Data (Coarse Granularity)

In the previous section (Section 7.2.1), we discussed the characteristics and behaviors of our algorithms using raw data. In this section, we focus on the state level; we decrease the data granularity by grouping the raw values into states, like in the medical data study. As a result, we can observe how our methods behave with different data granularity. Our states are generated by equal linear space from the maximum raw value. In other words, all states have the same distance. We use 10 and 30 states to demonstrate the experimental results in this section. We also include the raw values here, which are shown as the blue line in all of our figures in this section. The blue line represents the finest state (the raw value), which is the maximum state number of that redundancy degree setting. Since the blue lines are the curve of raw value, they are also equal to the curve in the above section (Section 7.2.1). In most cases, we expect to see similar curve patterns result in different numbers of state tests.

The quality of state estimation involves two factors: the number of misestimations and the severity of misestimations. Consider a data set that is classified into five states (Case A) and 100 states (Case B) like in the table below.

	States	Max state	1 state error
Case A	1,2,3,4,5	5	1/5
Case B	1,2,3,4,5….100	100	1/100

In case A, we generally expect fewer misestimations than in case B (since the state range is larger). However, we should expect that in case A, the misestimations are more severe. Misestimations in larger ranges (coarser state granularity) should be penalized more severely, as compared to misestimations of smaller ranges (finer state granularity). For example, if we make 10 1-state errors in case B, the accumulated error rate is (1/100) x10 = 1/10. On the other hand, if we make 2 1-state errors in case A, the average error rate is (1/5) x 2 = 2/5. On average, the estimation quality is better in case B. In other words, one 1-state error in case A would cost 20 1-state errors in case B. As a result, the smaller number of states does not necessarily imply a higher degree of accuracy.

7.2.2.1 BA Method

Figure 69 shows the BA's result in a normal distribution data test environment, while Figure 70 describes its misestimation ratio. Several important observations can be made:

1. In most of these scenarios, the average error rate decreases when the number of states is lowered. Scenario (20, 20, 100) and (20, 100, 20) are two exceptions, which show a greater average error dynamic when changing the state numbers. Their results in Figure 70 show a higher misestimation ratio, among others; this explains the way in which the BA makes more (severe) errors in these two scenarios.
2. We observe that high event density scenarios have a lower average error rate; Figure 70 also describes this by showing that high event density scenarios have a lower misestimation ratio than low event density scenarios. In other words, the BA performs better in high event density scenarios.
3. The report density has a higher impact on the error dynamics than the event density.
4. Each scenario is associated with a minimal average error rate. This means that the BA can take advantage of report redundancy in certain redundant degrees, but having more reports does not always guarantee the continued improvement in estimation accuracy.

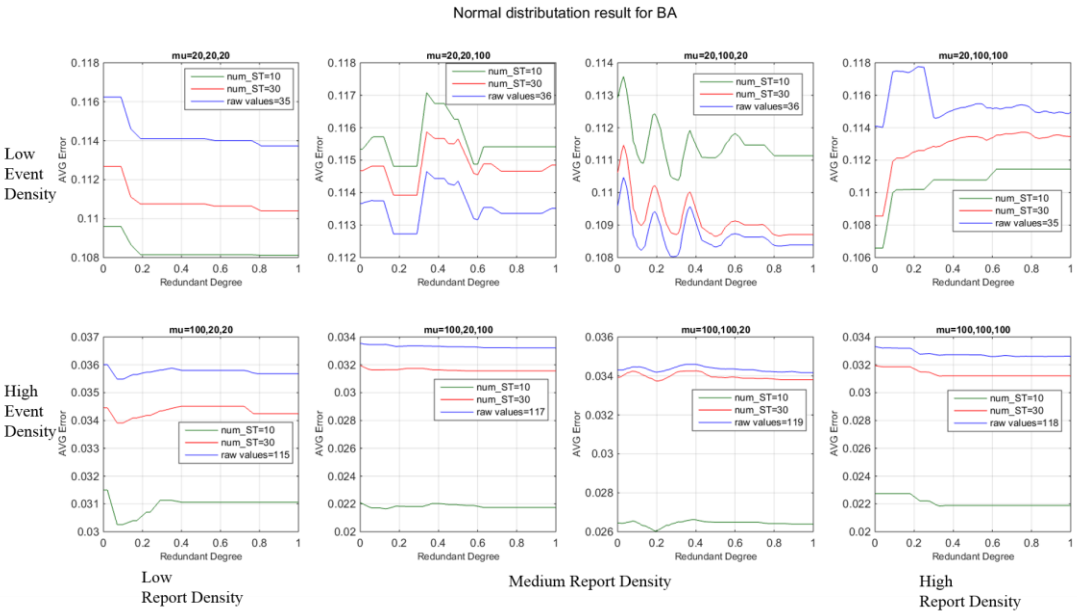


Figure 69. BA State sequence recovery in the normal distribution

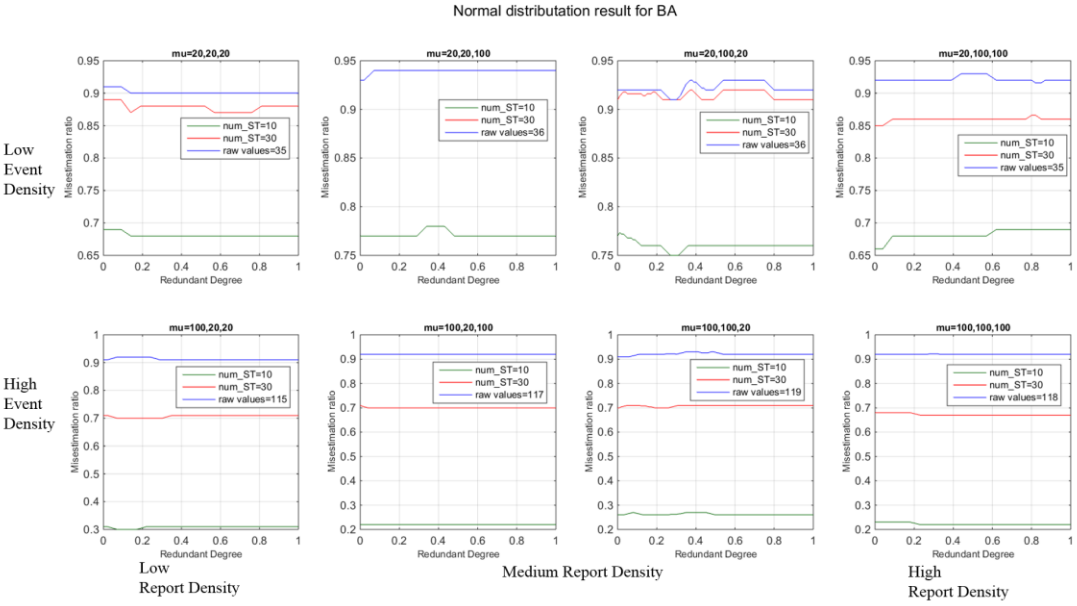


Figure 70. The BA misestimation ratio in the normal distribution

Figure 71 and Figure 72 show the BA's results and misestimation ratios in the exponential data environment, respectively. There are several important observations here:

1. We observe more curve dynamics in this case, as compared to the normal distribution data.
2. Unlike the normal distribution data, we notice that the state level tests (either state=10 or state=30) always have better accuracy than the raw value.
3. This test is different from the normal distribution test; the high event density does not show better accuracy in this case. Furthermore, the average errors do not show an obvious effect from either report density or event density.
4. We also find that this test has a similar (curve) behavior to the normal distribution, which benefits from certain redundant degree settings. In other words, we can see a lower average error rate in all scenarios.
5. Although it is not obvious like in the normal distribution environment, we still observe that the report density has a higher impact on the error dynamics than the event density.
6. In general, we see more average errors here than the normal observation test. We can further observe this phenomenon in Figure 72; it obviously shows that the state tests have a higher misestimation ratio while the redundant degree increases.

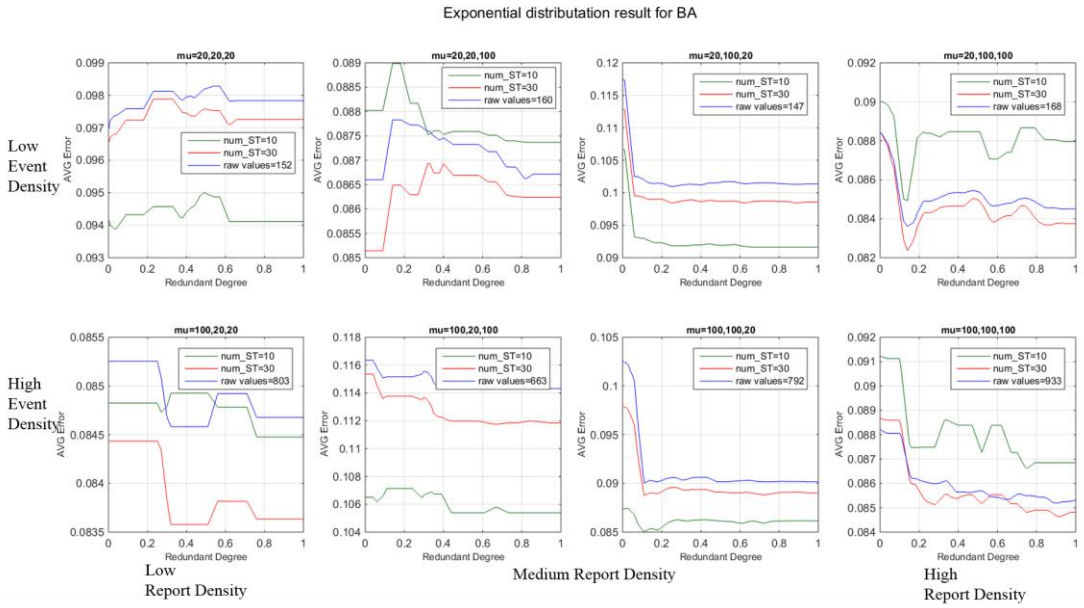


Figure 71. The BA state sequence recovery in the exponential distribution

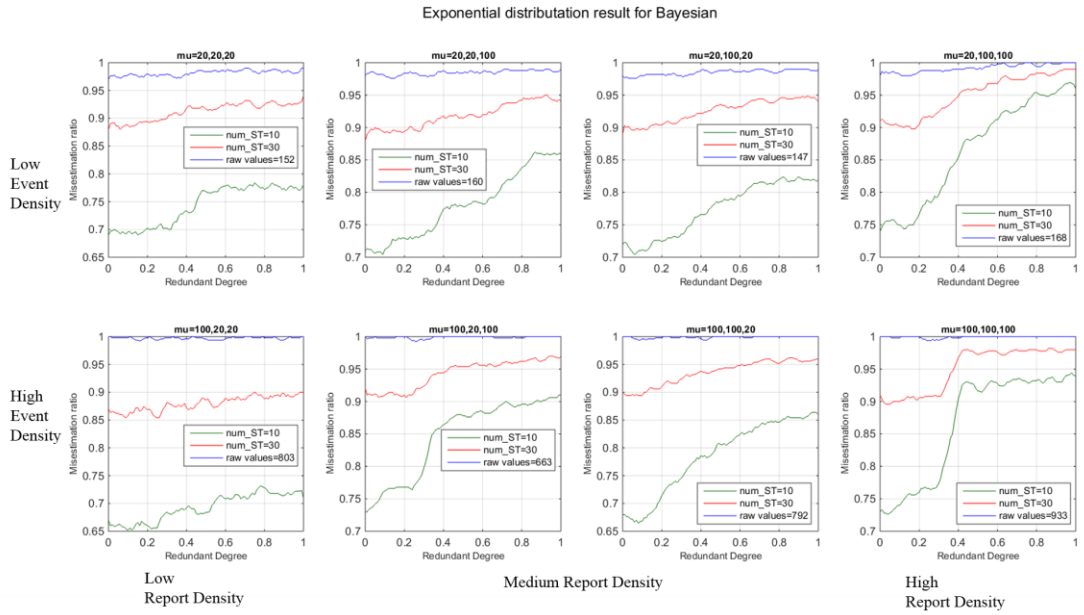


Figure 72. The BA misestimation ratio in the exponential distribution

7.2.2.2 GA Method

Figure 73 demonstrates the GA's result in the normal distribution data environment while Figure 74 describes its misestimation ratio. There are several important observations:

1. We observe the curve moving down the pattern of using states instead of raw value estimation in most scenarios. This explains that the GA can increase its accuracy through report redundancy.
2. The high report density scenarios (20,100,100) and (100,100,100) do not see a large deviation from both state = 30 and state =10. In Figure 74, these two scenarios also show a higher misestimation ratio, among others; this explains why the GA creates more errors in these two exceptions.
3. We observe that high event density scenarios have a lower average error ratio; Figure 74 describes this situation by showing that high event density scenarios have lower misestimation ratios than low event density scenarios. In other words, similar to the BA, the GA performs better in high event density scenarios.
4. The report density has a higher impact on the error dynamics than the event density.
5. Similar to the BA, each scenario is associated with a minimal average error ratio. This means that the GA can also take advantage of report redundancy in certain redundant degrees, but having more reports does not always guarantee the continued improvement of estimation accuracy.

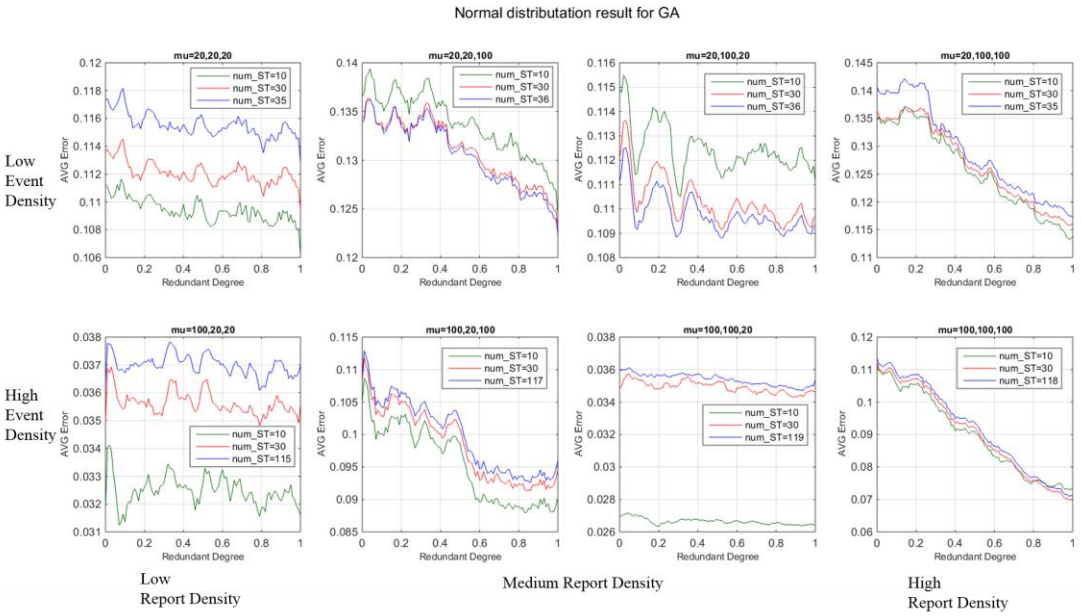


Figure 73. The GA state sequence recovery in the normal distribution

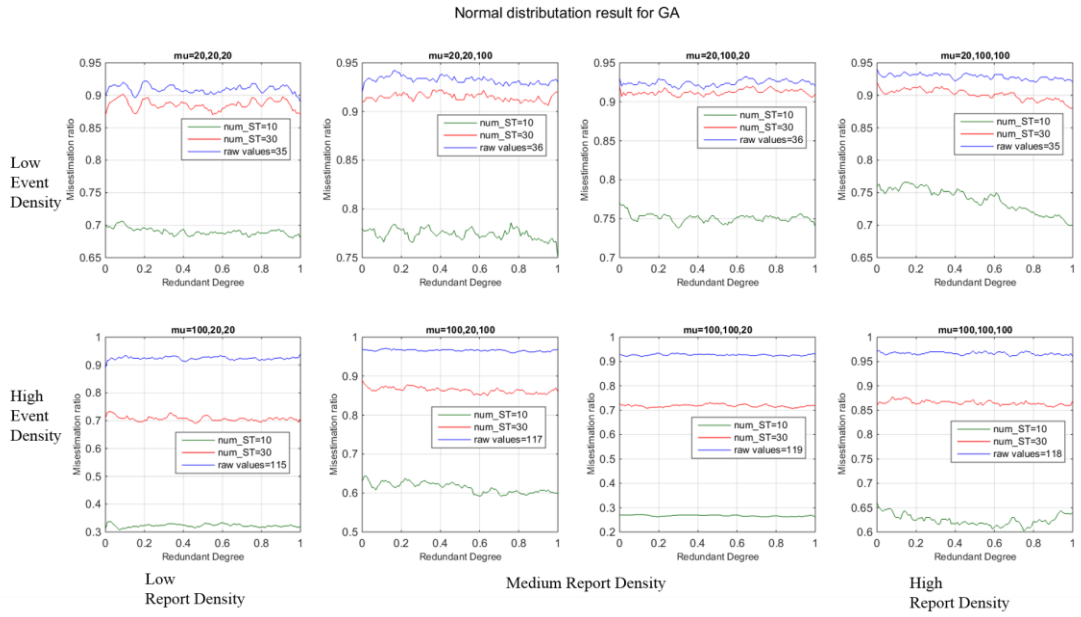


Figure 74. The GA misestimation ratio in the normal distribution

Figure 75 and Figure 76 show the GA's state sequence recovery result and misestimation ratio in the exponential data environment, respectively. There are several important observations here:

1. State sequence estimation generally has better accuracy than the raw value estimation, and this test also shows that the smaller number of states imply a higher accuracy, except in scenario (100,20,20).
2. The short report density scenario (20,20,20) and (100, 20, 20) indicate a larger average error dynamic in different redundant degree settings. Meanwhile, the high event density and high report density scenarios (20,100,100) and (100,100,100) do not see too much deviation between state = 30 and raw value estimation unless we change to state = 10.
3. Unlike the normal distribution data, in this case, we notice that the state level tests (either state=10 or state=30) always have better accuracy than the raw value.
4. This test is different from a normal distribution test; in general, the high event density does not show better accuracy here. We do not see an obvious effect from an increase in either report density or event density.
5. We also find similar (curve) behavior to normal distribution, which confers benefits from certain redundant degree settings.
6. We observe that the report density has a higher impact on the error dynamics than the event density.
7. In general, we see more average errors here than in the normal observation test and we can further observe this phenomenon from Figure 76, which shows that the state tests have more misestimations when the redundant degree increases.

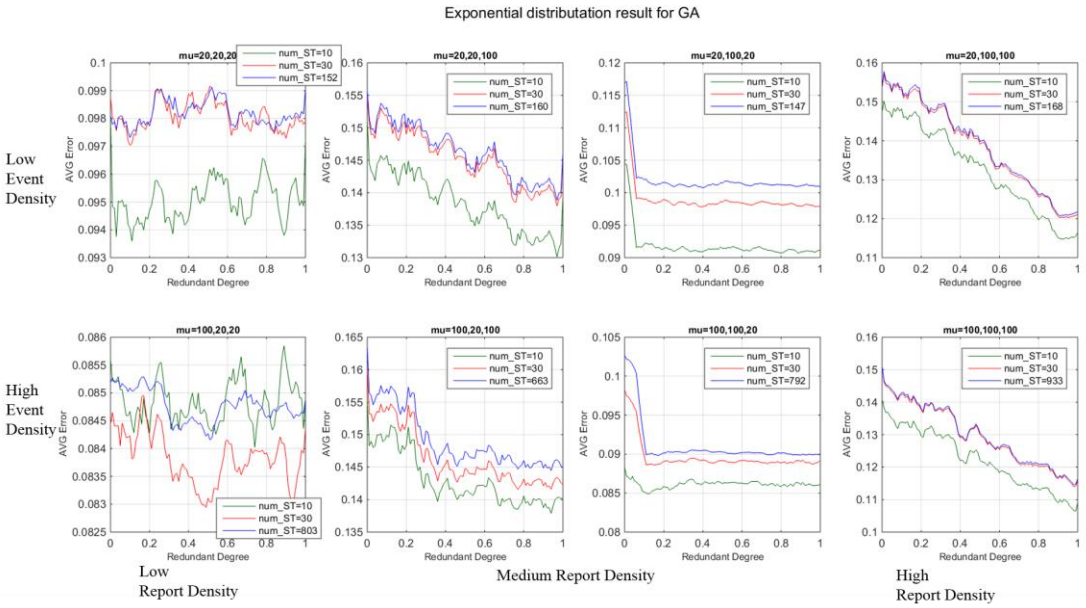


Figure 75. The GA state sequence recovery in the exponential distribution

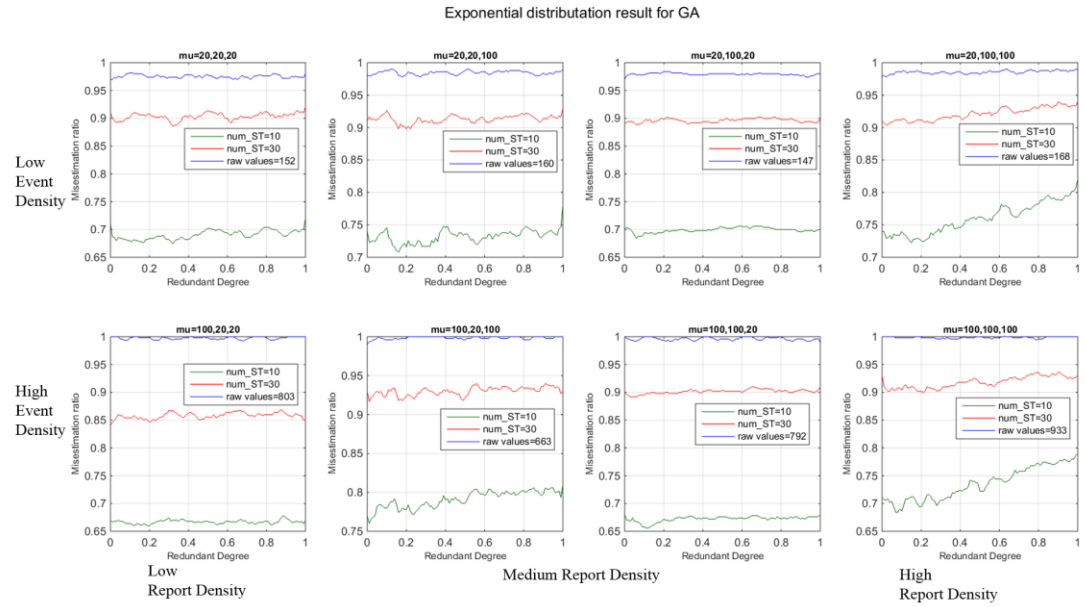


Figure 76. The GA misestimation ratio in the exponential distribution

7.2.2.3 LS Method

Figure 77 shows the LS's state sequence test results in a normal distribution data environment, while Figure 78 describes its misestimation ratio. There are several important observations:

1. We observe that there is not too much deviation by changing the state numbers. For a better explanation of this finding, consider the example from Figure 63 in Section 7.1.2. As we can see, the LS may assign a zero value to many individual time spans (intervals) in order to find the optimal solution. This makes the estimated state to be the minimal state (namely, state 1 in our case). Since a zero value is always equal to state 1, we cannot facilitate the accuracy by changing the raw values to states.
2. The LS does not benefit from report redundancy; Figure 78 obviously shows that the misestimation becomes higher while the redundant degree increases.
3. The average error is not affected by event density; however, we do see that the low report density scenarios of (20,20,20) and (100,20,20) have a lower average error rate than others.
4. Although the error rate is higher than that of the BA and the GA in this case, we can still see that the report density has a higher impact on the error dynamics than the event density.

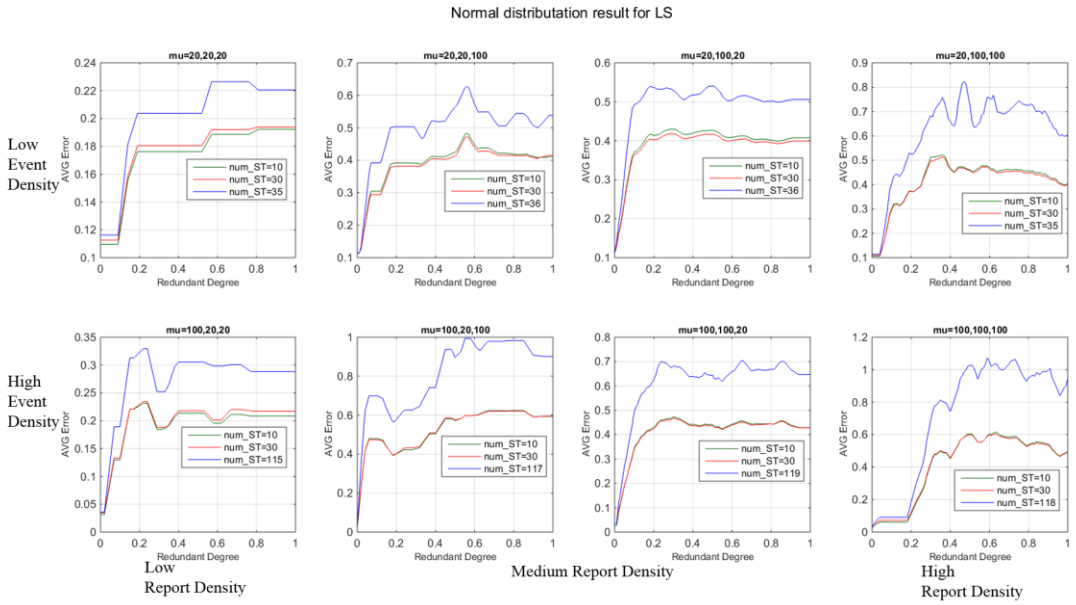


Figure 77. The LS state sequence recovery in the normal distribution

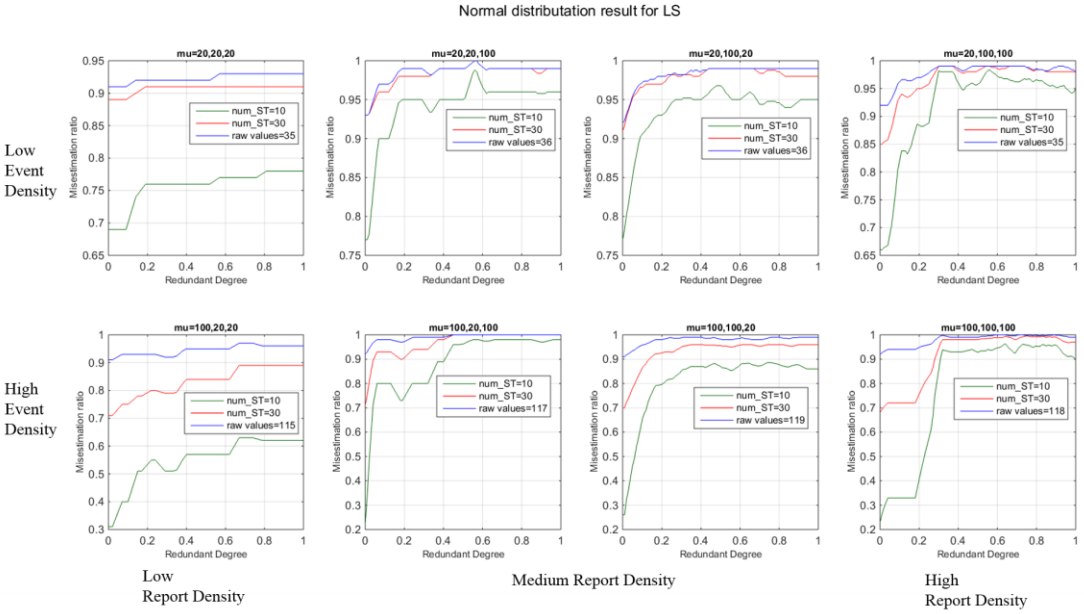


Figure 78. The LS misestimation ratio in the normal distribution

Figure 79 and Figure 80 show the LS's result and misestimation ratio in the exponential data environment, respectively. There are several important observations here:

1. It appears that there are more curve dynamics than in the normal distribution test in a different number of states. In general, we do not consider that using state sequence will provide more accuracy than the raw value estimation in this test.
2. Although the LS has a higher error rate than the BA and the GA, when compared to itself, LS shows better accuracy here than in its normal test environment. Figure 80 also shows a smaller misestimation ratio than the misestimation ratio in Figure 78.
3. This test is different from the normal distribution test; there is no obvious effect from either report density or event density.
4. Similar to the normal distribution, we do not consider that the LS benefits from report redundancy, and Figure 80 also shows that the misestimation ratio becomes higher while the redundant degree increases in most scenarios.
5. Although it is not obvious (like in the normal distribution environment), we still observe that the report density has a higher impact on the error dynamics than the event density.

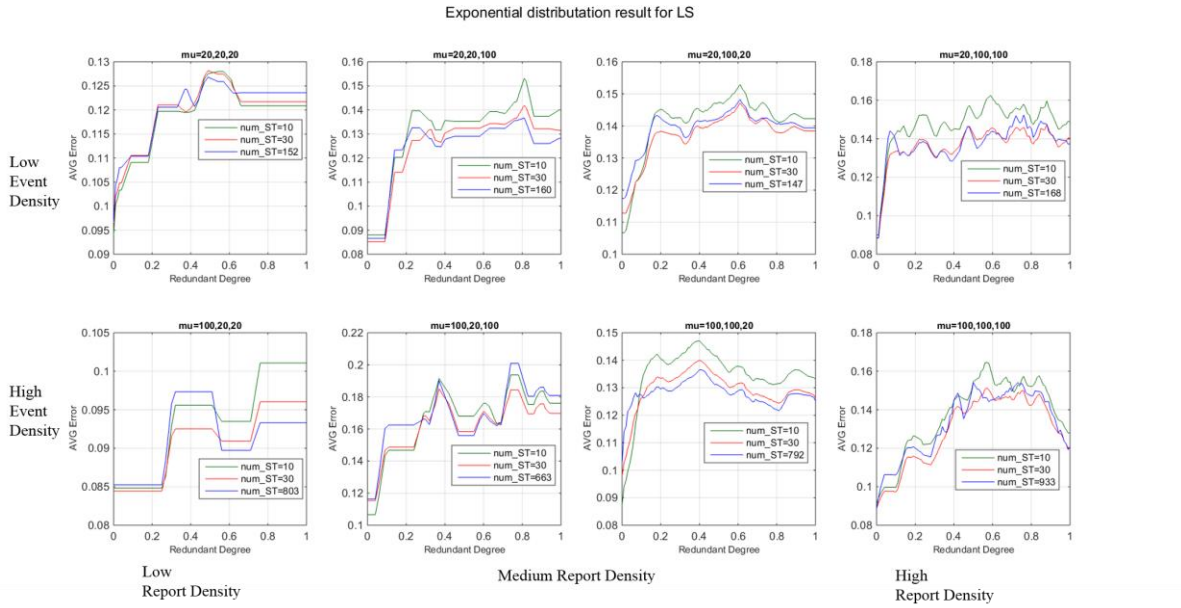


Figure 79. The LS state sequence recovery in the exponential distribution

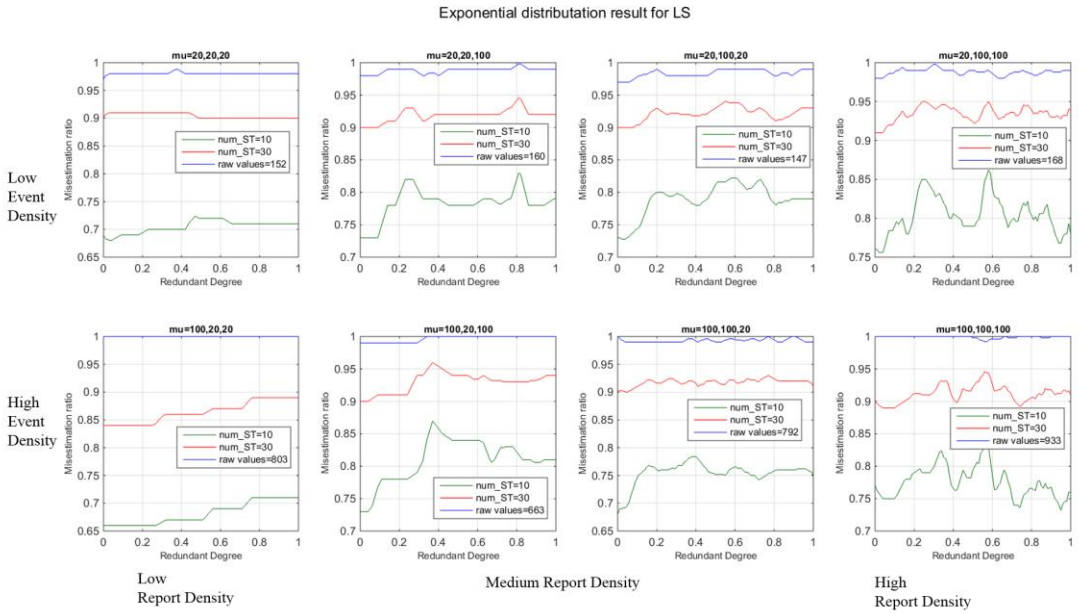


Figure 80. The LS misestimation ratio in the exponential distribution

7.2.2.4 Bayesian Method

Figure 81 shows the Bayesian comparison's state sequence test result in a normal distribution data environment, while Figure 82 describes its misestimation ratio. There are several important observations:

1. As we have concluded in the raw value test section (Section 7.2.1), the Bayesian comparison suffers from report redundancy, and having more reports (increasing the redundant degree) generates more misestimation errors; Figure 82 describes a situation where the misestimation ratio becomes higher while the degree of redundancy increases.
2. We do not see an obvious curve deviation dynamic when changing the number of states in most of the scenarios. From this result, we conclude that we have a low similarity between Bayesian comparison generated sample reports and ground truth reports, and moving the raw data to the state level does not significantly improve the accuracy either.
3. Although it is not obvious, we still find the report density has a higher impact on the error dynamics than the event density
4. The high report density scenarios of (20,100,100) and (100,100,100) show more errors than the other scenarios, which can be explained by the misestimation ratio significantly increasing, as shown in Figure 82.

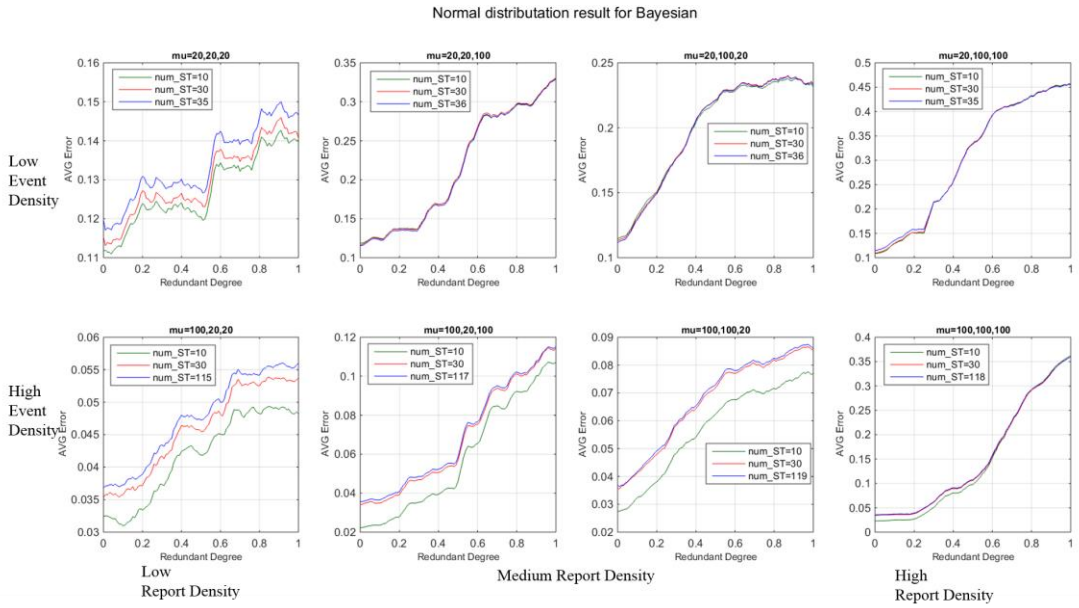


Figure 81. Bayesian comparison state sequence recovery in the normal distribution

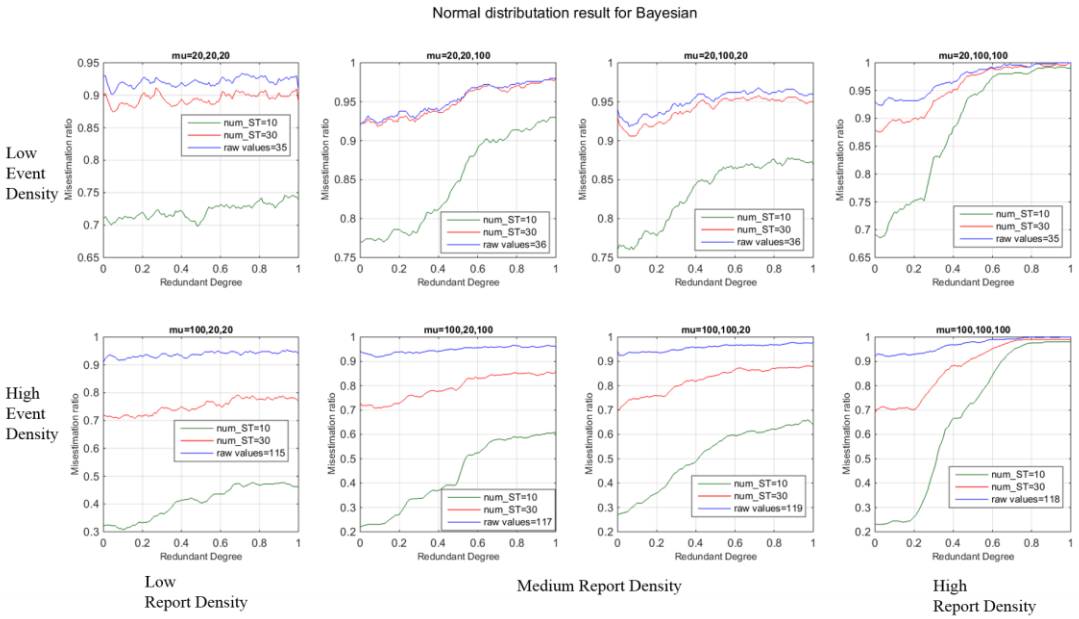


Figure 82. Bayesian comparison misestimation ratio in the normal distribution

Figure 83 and Figure 84 show the Bayesian comparison's result and misestimation ratio in the exponential data environment, respectively. There are several important observations here:

1. As compared to the normal distribution test in Figure 81, we observe that the low event density scenarios have a slightly lower average error rate; on the other hand, the high event density scenarios show a little higher average error rate.
2. From the curve's perspective, it looks very similar to the normal distribution test in Figure 81.
3. Although it is not obvious, using the state sequence has more accuracy than a raw value estimation for this test.
4. Similar to the normal distribution, the Bayesian comparison does not benefit from report redundancy, and Figure 84 also shows that the degree of misestimation becomes higher while the redundant degree increases in most of the scenarios.
5. Similar to the normal distribution, we still observe that the report density has as a higher impact on the error dynamics than the event density.

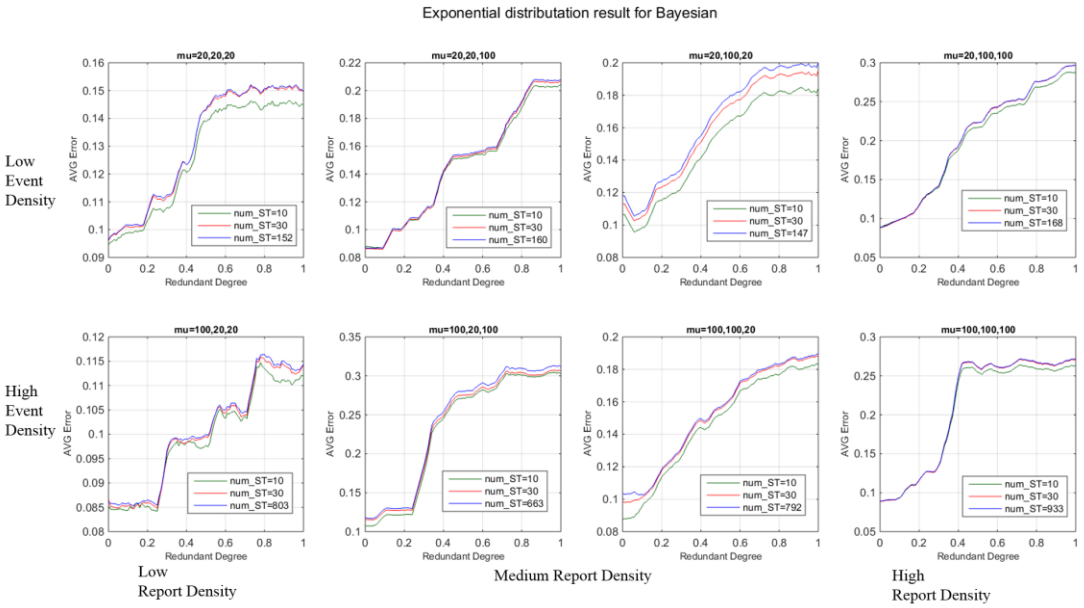


Figure 83. Bayesian method state sequence recovery in the exponential distribution

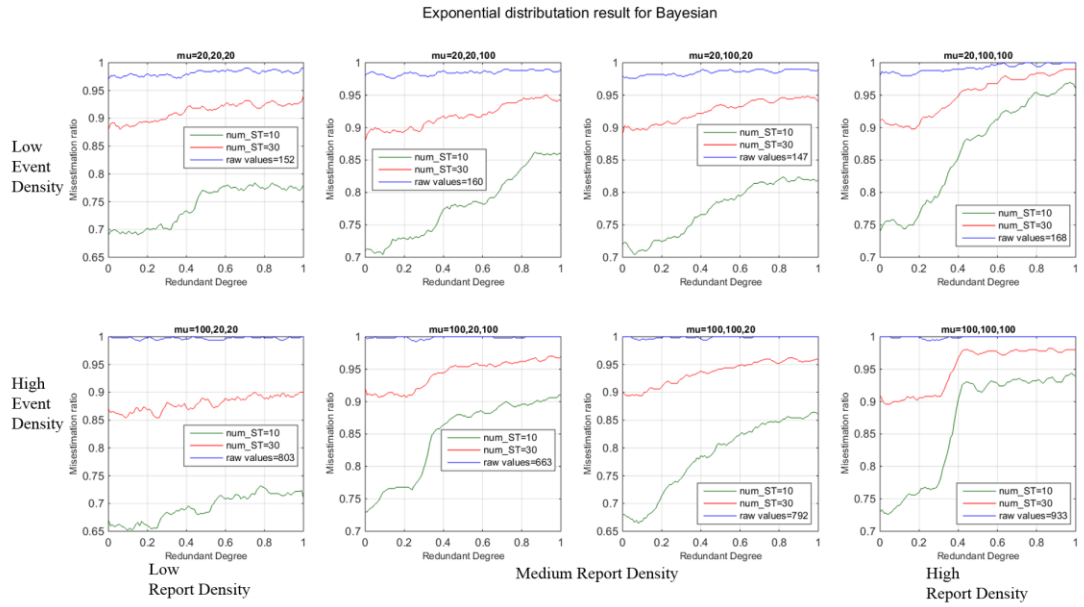


Figure 84. Bayesian method misestimation ratio in the exponential distribution

7.2.2.5 Comparisons Between Different Methods for Different States

In order to have a better comparison between different methods for different states, Figure 85 and Figure 86 show all methods used in the normal distribution test in state 10 and state 30 respectively. In general, we conclude that the accuracy from those methods is in the order of the BA > the GA > the Bayesian comparison > the LS. Meanwhile, Figure 87 and Figure 88 display the exponential test in states 10 and 30, accordingly. We can see there are curve crossover behaviors between the LS and the Bayesian comparison. As a result, the order of accuracy changes to the BA > the GA > the LS > the Bayesian comparison. Moreover, from those figures, we summarize the characteristics of our test results in the following table.

	Better accuracy in raw value or state test	Better accuracy in normal or exponential distribution data test	Higher impact in report density	Benefits from report redundancy
BA	State (with smaller exceptions)	Normal	Yes	Yes
GA	State	Normal	Yes	Yes
LS	Normal test→State Exp test→Raw value	Exponential	Yes	No
Bayesian comparison	Normal test→State Exp test→ State (some of them not obvious)	Most scenarios are Normal	Yes	No

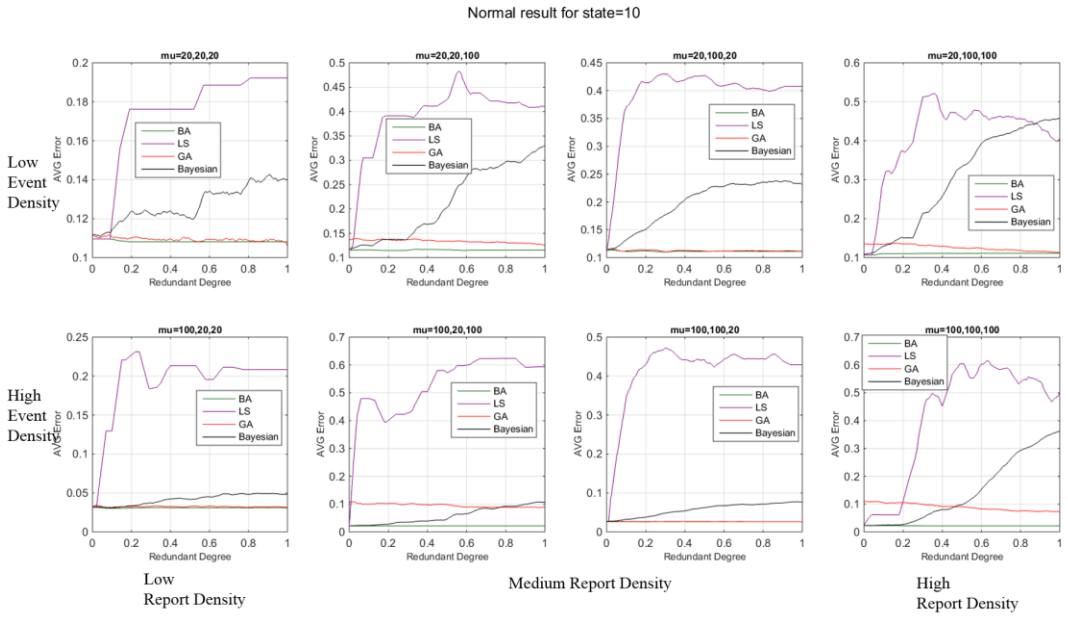


Figure 85. The algorithm comparisons for the normal distribution state = 10

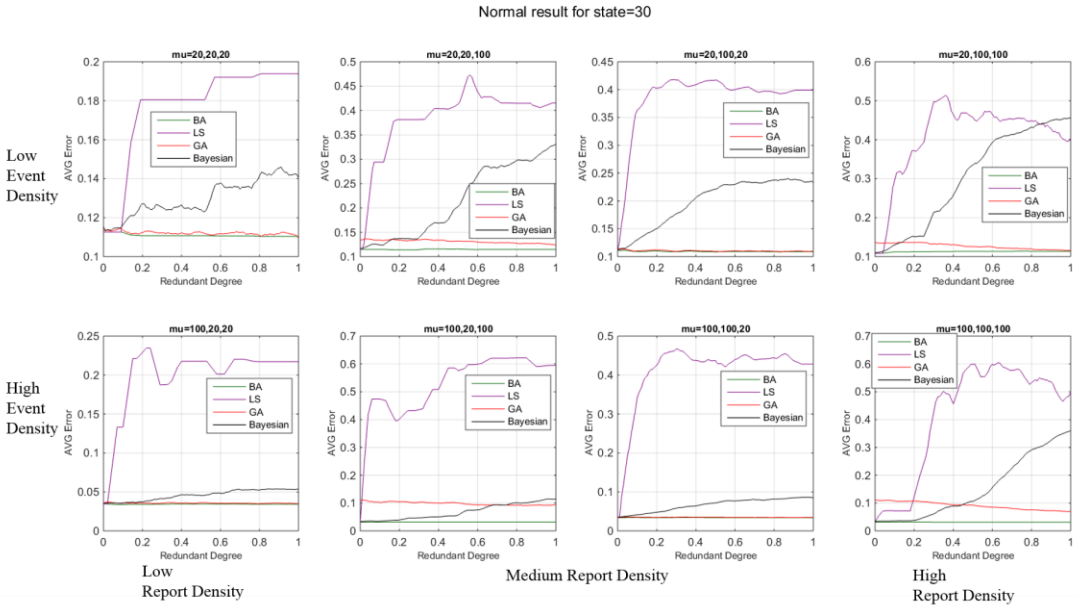


Figure 86. The algorithm comparisons for the normal distribution state = 30

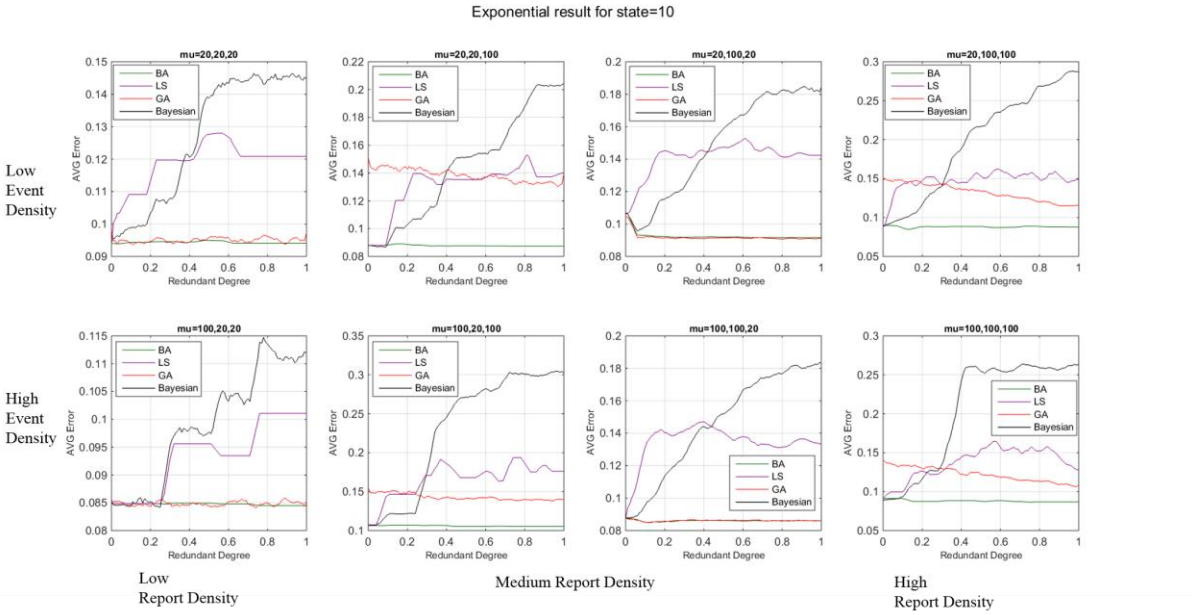


Figure 87. The algorithm comparisons for the exponential distribution state = 10

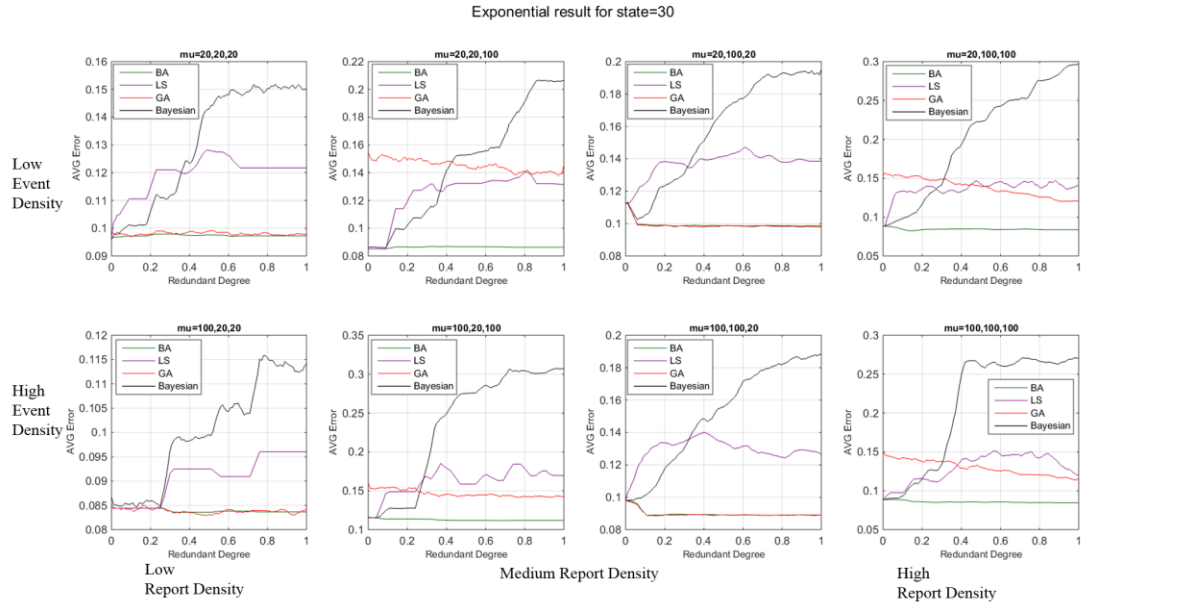


Figure 88. The algorithm comparisons for the exponential distribution state = 30

7.2.3 Impact of Optimal Degrees of Conflict

In the previous sections (Section 7.2.1 and 7.2.2), we demonstrated the behaviors of the SSR approach under all redundant degrees of all 8 scenarios. In this section, we test our algorithms according to the concept of optimal degrees of conflict (CD) from our previous study (Section 6.5.2) for each scenario. Note that we define the term "*conflict degree"* as equal to *"redundant degree"* and as a result, they can be interchangeable in this section and later sections. We discuss the performances of algorithms under the optimal CD. The optimal CD is considered a preprocessing procedure, which filters out highly redundant reports and provides better estimation results. In this test, we estimate the accuracy using the average error, which is defined as:

$$Relative\ Error = \frac{abs(ground\ truth\ state - estimate\ state)}{\max(\ estimate\ state)}$$

The curves inside the figures from two previous sections are sometimes difficult to compare. Therefore, in this section, we consider another approach for the comparison: we plot the relative error to a percentile plot, as in Figure 89. For better illustration purposes, we alternately transfer the percentile plot to the area under the curve (AUC) of the percentile, as shown in Figure 90. In this case, the higher AUC value represents better accuracy. Note that AUC is usually calculated from the receiver operating characteristic curve (ROC) with measurements of

.90-1 = excellent
.80-.90 = good
.70-.80 = fair
.60-.70 = poor
.50-.60 = fail

However, recent research indicates that using this interpretation may not truly reflect the accuracy for the model comparison and suggest a domain-dependent measurement instead [27].

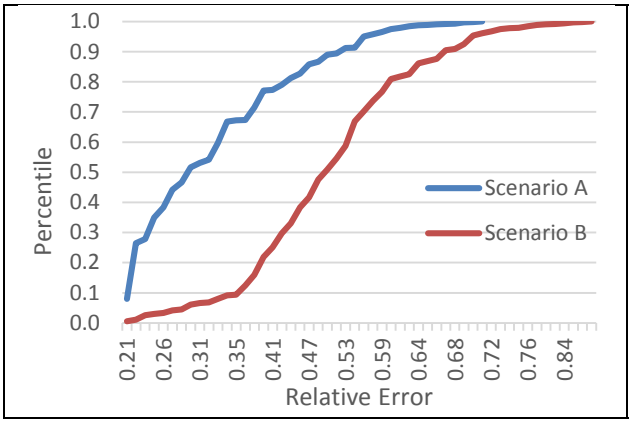


Figure 89. Percentile Plot

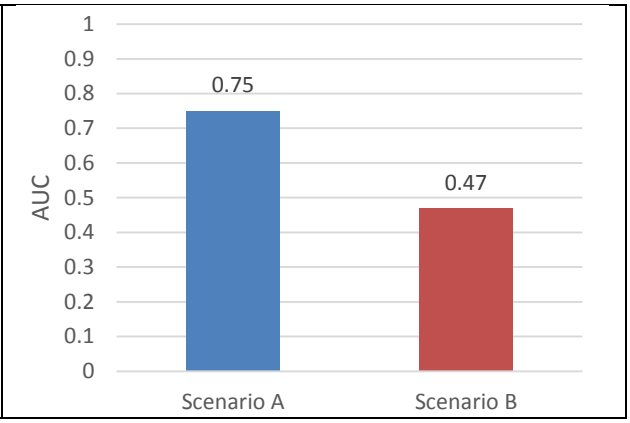


Figure 90. Area Under Curve (AUC)

In Section 5, we show that the optimal CD can minimize misestimation values in reported time spans. Similarly, in this test, we would like to apply the optimal CD to the SSR problem to discover its overall impact on accuracy. Figure 91 and Figure 92 show the result of four algorithms using the optimal CD in normal and exponential data, respectively. This result demonstrates that the optimal CD improves the accuracy (higher percentile area) in all cases. As a result, we continue to adopt it as a preprocessing step for the rest of our tests. We observe that some algorithms perform better than others and that the accuracy in the exponential data environment (Figure 92) seems lower. This result is consistent with the conclusions from previous sections, that all algorithms work better in normal data distribution than the exponential data distribution.

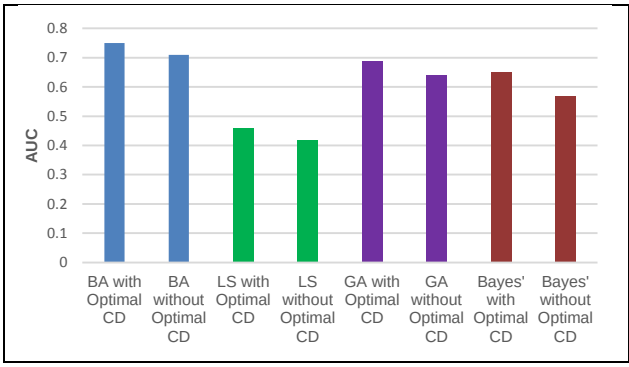


Figure 91: Optimal CD vs non-optimal CD (normal distribution)

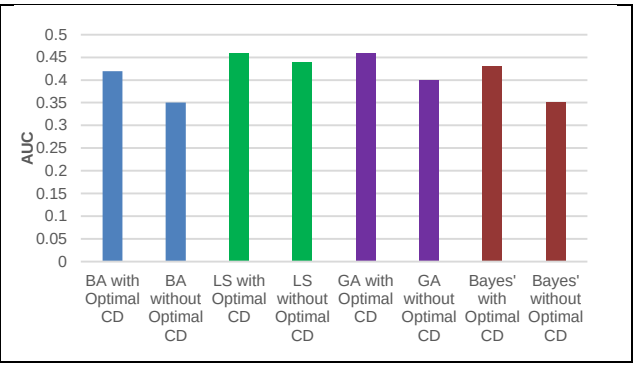


Figure 92: Optimal CD vs non-optimal CD (exponential distribution)

7.2.4 Additional Assumptions

In Sections 7.2.1 and 7.2.2, we assumed that the reports provide the correct total values for their covered period. In this section, other than simply the total values, we assume that we have additional information from other heterogeneous data sources, and that we can use this information to better evaluate the sequence data. In fact, in real-world problems, important events are usually repeated in different places or in different formats. For example, our high-level reports (either yearly or monthly) show that there was an outbreak of smallpox in New York City in the year of 1913. Although the detailed disease distribution on a weekly basis was not available, since this is a major event in history, other data sources such as news, journal, books, and diaries, among others, might provide additional information that can help to uncover any previously unknown weekly data.

Under this assumption in the experiment, we consider that we have additional information in knowing the "extreme values" distribution of the report. Note again, we still do not know the actual value for any time interval in our reports. These "extreme values" are defined as a point of data with a value that is either higher or lower than its adjacent time intervals. In other words, they are local maximums and local minimums, respectively. For example, when considering a state sequence of {5, 7, 3, 1, 4}, the time interval 2 is an upper extreme and time interval 4 is a lower extreme. In the following paragraphs, we will discuss how we extend our algorithms to use

them with the experimental results. We do not include the LS in this section, because the LS is an equation based approach. We cannot improve the results unless we can specify the true value (not extreme value) by adding additional equation.

For the BA, we simply add one state to the upper extreme time interval or subtract one state to the lower extreme time interval.

Figure **93** shows that the BA does not seem to have noticeable improvement from the extreme information in the normal distribution. From our observations, simply adding or subtracting one state from the original estimated state sequence does not really reflect the truth. For example, if the ground truth sequence is {1, 7, 2}, which has the upper extreme point in the middle, the BA may change its original estimated sequence from {1,2,2} to {1,3,2}, which only reduces a single state of misestimation.

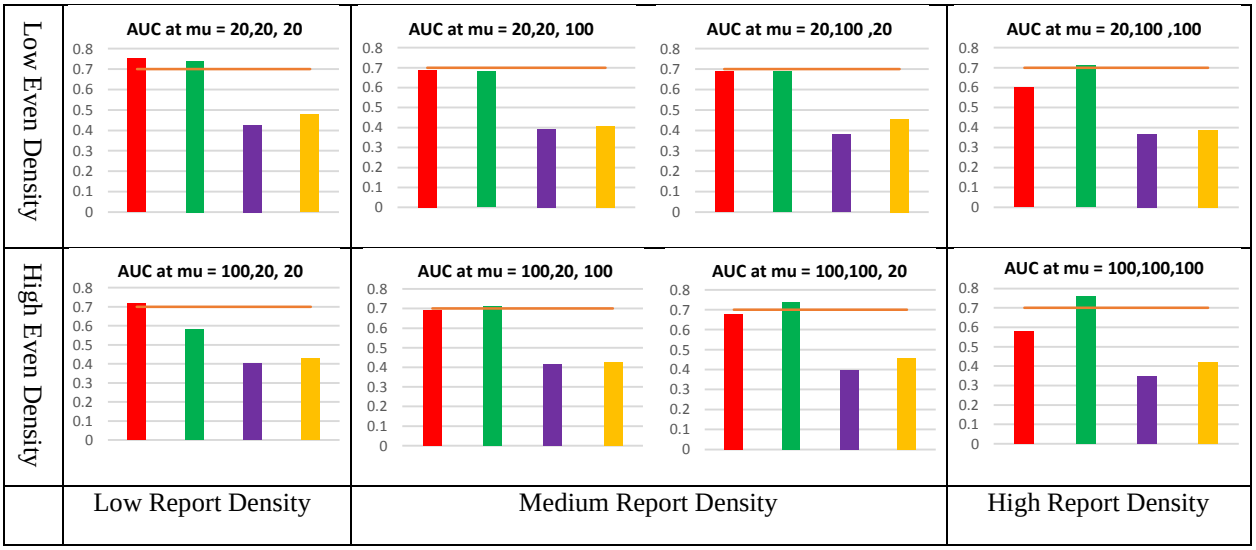


Figure 93. BA results with extreme information
■ BA normal ■ BA normal with extreme information ■ BA exponential ■ BA exponential with extreme information

For the GA at the last step (step 5), instead of choosing the most frequent (mode) state for the time interval from entire population, we only pick sequences that match the pattern of the

extreme value. When considering a state sequence population in Figure 94, we assume that the second time interval is known as the upper extreme. The original GA will generate state 4 for the second time interval, since it is in the mode state. When the GA uses the extreme value information, it uses the average state from S_1, S_2, and S_4, because they have the extreme value in the second time interval.

Sequence ID	Estimate state sequence
S_1	{1,5,2}
S_2	{4,6,3}
S_3	{2,2,2}
S_4	{2,4,3}
S_5	{2,4,4}

Figure 94. An example of the GA algorithm with extreme value information

Figure **95** illustrates that having the extreme information provides an obvious improvement in all scenarios, when using the GA approach.

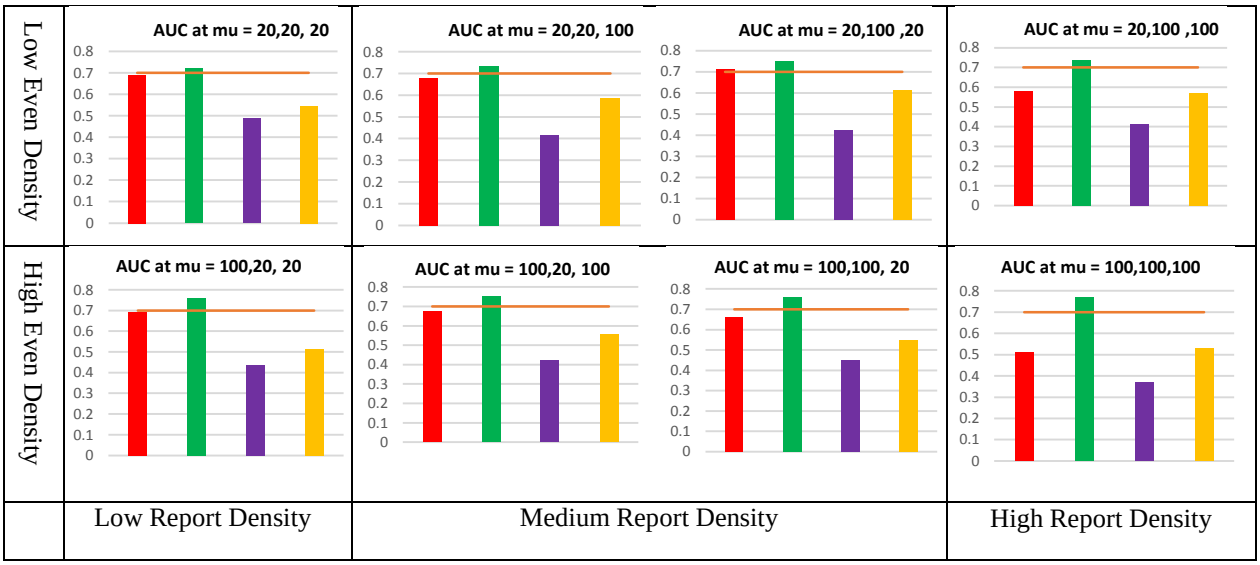


Figure 95. GA results with extreme information
■ GA normal ■ GA normal with extreme information ■ GA exponential ■ GA exponential with extreme information

In the Bayesian approach (Section 7.1.4), we set the constraint for generated sample reports in step 2 as

SUM (sample value sequences) ≤ original report sum

Here, we extend it to use the extreme info by adding another constraint, as

Pattern of sample value sequences ~ Pattern from extreme info of report

This forces the Bayesian approach into generating similar sequential patterns, which include the provided extreme information. Note that this sample generation can be time-consuming for the longer reports.

Figure **96** illustrates that having extreme information can help the performance for all scenarios.

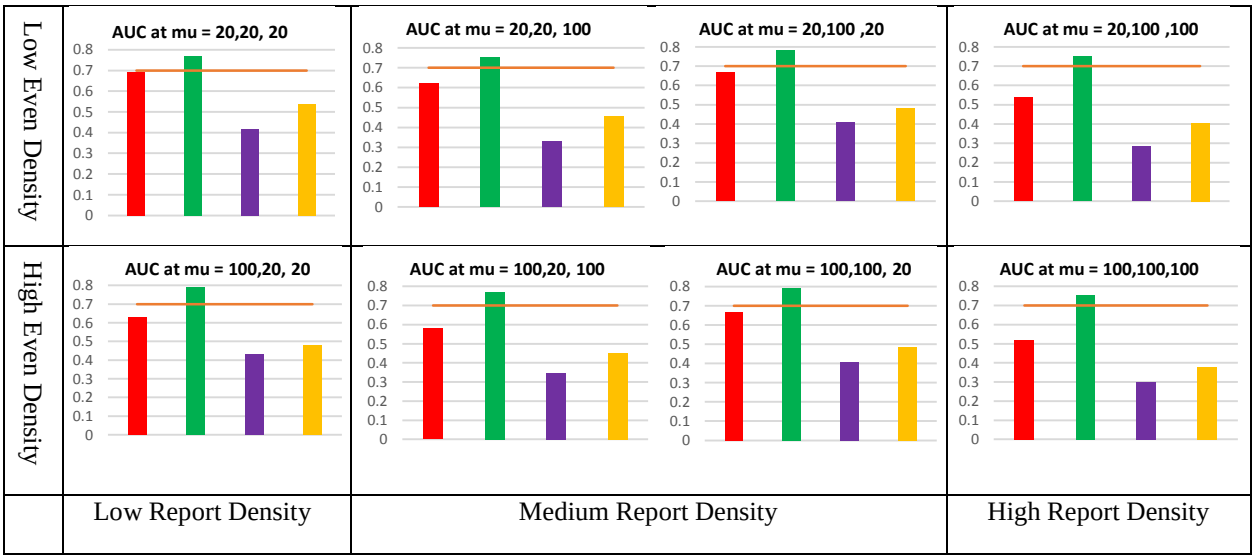


Figure 96. Bayesian approach results with extreme info
■ Bayesian approach normal ■ Bayesian approach normal with extreme information ■ Bayesian approach exponential ■ Bayesian approach exponential with extreme information

7.3 SUMMARY OF THIS STATE SEQUENCE RECOVERY (SSR) CASE STUDY

We have considered a novel data recovery approach to the problem of redundant and sparse sequential data. Our goal is to estimate the most possible state for each individual time interval. Our experimental results are based on four algorithms, which we have developed and tested in different simulation set-up scenarios under both normal and exponential data distributions.

In Section 7.2.1, we performed a raw value estimation study that discusses the characteristics of each algorithm. Note that the raw value is the finest data granularity, and in this case, we can consider it to be the maximum number of states. Moreover, we performed another test in Section 7.2.2 where we decreased data granularity by grouping the raw value to the state (coarse granularity). In most cases, we see similar curve patterns, which have lower average error rates while the number of states decreases. We also see that some scenarios have greater deviation curve behaviors than others. In general, the test result (accuracy) shows that the BA is the most accurate in both the normal and the exponential data distribution tests; the overall accuracy is in the order of (the BA > the GA > the Bayesian approach > the LS) in the normal distribution test,

while it is (the BA > the GA > the LS > the Bayesian approach) in the exponential distribution test. In Section 7.2.3, we adopted the concept of optimal CD from our previous study, and the results show that state sequence recovery algorithms can benefit from this optimal CD by minimizing misestimation errors. In Section 0, we assume that we have the additional information from other heterogeneous data sources and our algorithm can use them to better evaluate the overall sequence data. When adding the extreme information to the test, the result is reversed completely, to the Bayesian approach > the GA > the BA. As for future work, we plan to further explore other state sequence recovery (SSR) strategies together with the optimal clustering method (the number of states) to estimate overall state sequence recovery.

8 SUMMARY AND DISCUSSION

8.1 SUMMARY

In the present work, I present a data processing architecture for efficient data warehousing that uses data from historical data sources. In Section 3, we developed a generalized process data warehousing (PDW) architecture that includes multilayer data processing steps to transform raw data streams into useful information to facilitate data-driven decision making. In Section 4, we provided a case study in the medical domain; we explored the applicability of PDW architecture to the case of *sparse process data*. We tested the proposed approach in a medical monitoring system, which takes physiological data and predicts in which clinical setting the data is most likely to be seen. We performed a set of experiments using real clinical data (from the Children's Hospital of Pittsburgh) that demonstrates the high utility of our approach. In Section 5, we performed another case study of *redundant process data*; we evaluated the applicability of PDW to a large-scale integrated framework (Tycho data) that includes historical epidemiological data from heterogeneous sources in different subject areas. We introduced the concept of degrees of conflict (CD) and created a simulation-based study of the tradeoffs between data fusion solutions and data accuracy. We also evaluated the solutions to this Tycho data warehouse using the optimal CD, and we observed a notable improvement in the accuracy of our estimations. In Section 6, we proposed and evaluated a state sequence recovery (SSR) framework, which integrates works from the two previous studies. Our experimental results are based on several algorithms, which we have developed and tested in different simulation set-up scenarios under both normal and exponential data distributions, and they indicated that the state sequence approach has a superior estimation accuracy to the raw data approach.

8.2 LIMITATIONS

There are several potential limitations that should be noted in our three Information Science (IS) contributions. In the first IS contribution, at current PDW, the system only focus on temporal data does not account for other potential sources of data that could be useful in determining event severity and facilitate decision making. Another limitation is the event analysis techniques described in PDW, NN, NBC, and DT, are not the only ways to analyze the event severity. Other methodologies including cluster analysis, neural networking, or simple linear regression analysis may provide superior results. The two techniques described here may lack enough discriminatory power, due to the limited sample size of the training set. With more and more data, the models may improve in their sensitivity and specificity.

In the second IS contribution, our approach does not account for other potential sources of data that could be useful in determining a patient's clinical status. For example, variables such as neurological status, nursing assessments, or laboratory data could give the model a fuller picture of a child's clinical status. While these variables are likely to be useful, the system shown in the present work represents a "lowest common denominator" approach to an alerting system. Introducing less objective measurements into the system may unintentionally obfuscate the signals. Another area of possible concern is the assumption that the last state for a vital sign remains constant until new information is entered into the system, which may not always be the case. An alternate option is to allow for null values to stay in the event table and have the severity analysis techniques take the state of "null" into consideration.

In the third IS contribution, our approach focused only on temporal conflicts. Other major types of conflicts such as spatial and naming conflicts could be useful in evaluating the heterogeneous historical data. As discussed, we designed our degree of conflict (CD) to determine the level of data redundancy. Other approaches, such as maximal likelihood estimation, may provide alternatives to data redundancy assessment. In addition, in which our state sequence recovery (SSR) algorithms are described (the BA, the GA, the LS, and the Bayesian approach), they are not the only ways to estimate the state sequences. Other methodologies, including cluster analysis or linear regression analysis, may provide superior results. Another area of possible concern is that we do not have a solution to choose an optimal

number of states in our system; not having an optimal number of states may increase the difficulty of the accuracy comparison. We also found that the LS method may assign a zero value to many individual time intervals (or time spans) in order to find the optimal solution. This makes the estimated state to be the minimal state (namely, state 1 in our case), which is usually against the ground truth. This will result in a much higher misestimation. Additionally, although the Bayesian approach uses the constraint of the total sum to generate random sample reports to ensure that the generated random samples are not too much different from the original report, those sample reports are still far from the ground truth. As a result, more reports (increasing the degree of redundancy) will generate more misestimation errors.

8.3 DISCUSSION

In the present work, I present a data processing architecture for efficient data warehousing from historical data sources. I addressed three major research questions in Section 1.2 and discussed their details in Sections 3 through 6. This section summarizes the answers to these questions.

(a) What are the key information processing steps for the proper use of process data?

We have developed a generalized process data warehousing (PDW) architecture that includes multilayer data processing steps to transform raw data streams into useful information that facilitates data-driven decision making. In essence, this PDW architecture includes the following steps and functionalities:

(1) processing heterogeneous multi-dimensional data and handling the challenges of *sparse process data* and *redundant process data*;

(2) transferring the data from lower layers to more meaningful states and providing new interpretations of the data, or fixing data errors in the different perspectives of the original data schema;

(3) Integrating the data with time intervals to generate event types, which reflect the severity of the event;

(4) Using sophisticated decision-making methods to monitor or analyze trends in event severities.

(b) How should we handle and use *sparse process data*?

We first convert the raw data to a proper data stream, which is adequate for further data analyzing, and then transfer the data to more meaningful states. To deal with the issue of sparse data, we adopted the same data (states) values from their previous tuples. This is a reasonable assumption, especially in the context of an emergency department. We further aggregated the data by merging the tuples with the same states into "events" and we map the "event" to its corresponding "event type," which represents the severity of the event. We demonstrated the feasibility of our solutions and the advantages of our approach in a medical early warning system.

(c) How should we handle and use *redundant process data*?

We elaborated our PDW into a large-scale heterogeneous historical data set that reports on numerous events with overlapping time intervals, locations, and names. We first developed several data preprocessing algorithms, which include data extracting, transforming, loading, error fixing, and semantics validation. The redundant data may include severe data conflicts caused by database redundancies that can prevent researchers from obtaining the correct answers to queries on an integrated historical database. We introduced a conflict-aware data fusion strategy to deal with the issue of redundant data in our PDW, and demonstrated that our approach significantly reduces data aggregation errors. Furthermore, we performed a state sequence recovery (SSR) study, which is another type of redundant process data research. That work adopts the concept of state sequences from Chapter 5 and embeds it into the redundant data environment of Chapter 6. In this case study, we estimated the most probable state at each individual time interval, and our test results indicate that the SSR approach has a better estimation accuracy than the raw data approach.

8.4 FUTURE RESEARCH

There are multiple areas of future research that can be explored with the use of this PDW framework and the current case studies: medical data case study, epidemiology data case study, and state sequence recovery.

PDW framework:

1. This PDW research can be extended to expand the case studies to test and verify their applicability to interdisciplinary environment. This same architecture could be tuned to preprocess the raw data, generate states and events, evaluate event severity, and facilitate decision making tasks.
2. Another area of future research lies in the exploration of alternate methodologies for event generation, such as linear regression, cluster analysis, or neural networking, all of which can be used to classify large amounts of data. These techniques should be explored as alternatives to the event classification techniques that are described in the present work.
3. One area for further exploration is focusing on model selection and prediction approaches with sophisticated machine learning methods. Based on the accuracy of the outcomes, the decision-making layer is designed to interact with the lower layer, as well as to have the ability to learn and adjust to improve the final predicted outcomes.

Medical Data Case Study

1. The goal of this case study was to bring technology to bear on the challenging environment of a busy clinical setting. The hope is that adding an intelligent alerting system can help physicians in their daily jobs of monitoring and treating patients. In future, we plan to collect feedback from clinical staff about their experiences in using this earlier warning system.
2. One area for further exploration is in the area of the vital sign “normal ranges.” While the average normal values for children of different age groups is fairly well established, there are situations where this is not the case. For example, neurologically devastated children often

have a lower resting heart rate than their peers. As a result, this research could be extended to establish individual normal ranges for each vital sign.

3. Furthermore, other variables which might contribute to the clinical picture need to be explored. The event analysis techniques should also be investigated across a larger, more generalized ED population. While respiratory complaints do make up a significant portion of chief complaints in an emergency department, it remains to be investigated as to whether the rest of the ED population can be analyzed using this particular set of event severity analyzing algorithms.
4. Another area for further exploration is the event unfolding logic that is used. The time threshold used in the current iteration of the CEWS is two hours; this time threshold could be changed to allow for the unfolding of more events at an earlier time. This could change both the analysis and the timing for when the rules are activated. Another area of possible research concerns the rules used to flag patients. All five rules explored looked for three ICU event types before flagging.
5. Further investigation would include raising the flag at two ICU event types or adding other, more sophisticated algorithms. It also remains to be seen if it is possible to take the general architecture described here and use it in other alerting contexts. In theory, one could take any set of clinical data with an outcome of interest and tune this system to alert for that outcome of interest.

Epidemiology Data Case Study

1. In future work, we will further explore the performance and accuracy tradeoffs associated with the various conflict-aware query evaluation strategies. We will design and develop a query optimizer that efficiently uses those tradeoffs.
2. In the current case study, we only focus on solving the issue of temporal conflicts. We will further investigate conflict-aware data fusion methods for spatial and naming conflicts.
3. We will investigate alternate historical data fusion strategies based on different notions of maximal likelihood estimation to estimate the event distribution from a set of the reported numbers of events.
4. We currently apply our conflict-aware fusion method to accurate reports and try to minimize the misestimation error of total reported sums from the covered period. In future, we will

extend our work to handle inconsistent reports and test them in our simulation scenarios, as well as evaluate its feasibility for the Tycho data set.

State Sequence Recovery (SSR)

1. In our system, the LS method often assigns 0 to certain time spans, which is usually contradicted by the ground truth. As a result, the accuracy of LS is relatively low when compared to our other methods. We currently use the least square method in the LS as and estimation algorithm. In future, we will investigate other linear regressing estimation algorithms, such as ridge regression and LASSO. As a result, we can decide to keep the LS as one of the SSR methods, or decide not to do so.
2. We currently apply the SSR method to accurate reports and try to estimate the most likely state sequence from the covered period. In future, we will extend our work to handle inconsistent reports and test it in our simulation scenarios, as well as evaluate its feasibility in the Tycho data set.
3. Another possible extension for this research is to expand the research to real-world problems, since the Tycho data is not completely eligible for the evaluation due to its lack of ground truth data. As a result, we can use it to evaluate the feasibility of the SSR approach.

BIBLIOGRAPHY

[1] M. L. Brodie, "Data Management Challenges in Very Large Enterprises," in Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[2] L. Golab, and M. T. Ozsu, "Issues in data stream management," *SIGMOD Rec.*, vol. 32, no. 2, pp. 5-14, 2003.

[3] D. Kifer, B.-D. Shai, and J. Gehrke, "Detecting change in data streams," in Proceedings of the Thirtieth international conference on Very large data bases - Volume 30, Toronto, Canada, 2004.

[4] R. Sadri, C. Zaniolo, A. Zarkesh, and J. Adibi, "Optimization of sequence queries in database systems," in Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Santa Barbara, California, United States, 2001.

[5] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom, *STREAM: The Stanford Data Stream Management System*, 2004.

[6] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards Sensor Database Systems," in Proceedings of the Second International Conference on Mobile Data Management, Hong Kong, China, 2001, pp. 3-14.

[7] U. Srivastava, and J. Widom, "Flexible time management in data stream systems," in Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Paris, France, 2004, pp. 263-274.

[8] A. Arasu, S. Babu, and J. Widom, "The CQL continuous query language: semantic foundations and query execution," *The VLDB Journal*, vol. 15, no. 2, pp. 121-142, 2006.

[9] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," in Proceedings of the 2003 CIDR conference, 2003.

[10] A. Lerner, and D. Shasha, "AQuery: query language for ordered data, optimization techniques, and experiments," in Proceedings of the 29th international conference on Very large data bases - Volume 29, Berlin, Germany, 2003, pp. 345-356.

[11] P. Domingos, and G. Hulten, "Mining high-speed data streams," in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, Massachusetts, United States, 2000, pp. 71-80.

[12] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, "Load shedding in a data stream manager," in Proceedings of the 29th international conference on Very large data bases - Volume 29, Berlin, Germany, 2003, pp. 309-320.
[13] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *SIGMOD Rec.*, vol. 34, no. 2, pp. 18-26, 2005.
[14] M. Charikar, L. O'Callaghan, and R. Panigrahy, "Better streaming algorithms for clustering problems," in Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, San Diego, CA, USA, 2003, pp. 30-39.
[15] D. Arthur, and S. Vassilvitskii, "How slow is the K-means method?," in Proceedings of the twenty-second annual symposium on Computational geometry, Sedona, Arizona, USA, 2006, pp. 144-153.
[16] S. Chaudhuri, and U. Dayal, "An overview of data warehousing and OLAP technology," *SIGMOD Rec.*, vol. 26, no. 1, pp. 65-74, 1997.
[17] A. Y. Halevy, N. Ashish, D. Bitton, M. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka, "Enterprise information integration: successes, challenges and controversies," in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, Baltimore, Maryland, 2005, pp. 778-787.
[18] G. Cormode, and M. Garofalakis, "Sketching probabilistic data streams," in Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Beijing, China, 2007, pp. 281-292.
[19] L. Golab, and M. T. Ozsu, "Update-pattern-aware modeling and processing of continuous queries," in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, Baltimore, Maryland, 2005, pp. 658-669.
[20] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264-323, 1999.
[21] H. Li, K. Zhang, and T. Jiang, "Minimum Entropy Clustering and Applications to Gene Expression Analysis," in Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference, Riverside, CA, USA, 2004, pp. 142-151.
[22] Y. Jung, H. Park, D.-Z. Du, and B. L. Drake, "A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering," *J. of Global Optimization*, vol. 25, no. 1, pp. 91-111, 2003.
[23] J. R. Quinlan, "Learning decision tree classifiers," *ACM Comput. Surv.*, vol. 28, no. 1, pp. 71-72, 1996.
[24] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, pp. 81-106, 1986.
[25] J. R. Quinlan, "C4.5: Programs for Machine Learning " *Machine Learning, Morgan Kaufmann Publishers, Inc.*, pp. 235-240, 1993.
[26] L. Feixiong, "An Improved Algorithm of Decision Trees for Streaming Data Based on VFDT," 2008, pp. 597-600.
[27] O. Job, S. H. S. Wonga, and A. J. Beaumonta, "A fuzzy decision tree-based duration model for Standard Yoruba text-to-speech synthesis," *Comput. Speech Lang.*, vol. 21, no. 2, pp. 325-349, 2007.
[28] H. Sug, "An effective sampling method for decision trees considering comprehensibility and accuracy," *W. Trans. on Comp.*, vol. 8, no. 4, pp. 631-640, 2009.

[29] J. G. Melike Bozkaya, Jan Martijn van der Werf, “Process Diagnostics: A Method Based on Process Mining,” in International Conference on Information, Process, and Knowledge Management, 2009.
[30] "Process mining," http://www.processmining.org/.
[31] W. M. P. v. d. A. Minseok Song, *Supporting Process Mining by Showing Events at a Glance*, Eindhoven University of Technology, 2007.
[32] A.K. Alves de Medeiros, A.J.M.M. Weijters, and W. M. P. v. d. Aalst, “Genetic Process Mining: A Basic Approach and Its Challenges ” in Proceedings of the Third international conference on Business Process Management, 2005.
[33] W. v. d. Aalst, T. Weijters, and L. Maruster, “Workflow Mining: Discovering Process Models from Event Logs,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 9, pp. 1128-1142, 2004.
[34] W. v. d. A. A. Weijters, and A. Alves de Medeiros, *Process Mining with the Heuristics Miner Algorithm*, Eindhoven University of Technology, 2006.
[35] J. A. Bilmes, “What HMMs Can Do,” *IEICE - Trans. Inf. Syst.*, vol. E89-D, no. 3, pp. 869-891, 2006.
[36] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” in Proceedings of the IEEE 77, 1989, pp. 257-286.
[37] V. S. Barbu, and N. Limnios, *Semi-Markov Chains and Hidden Semi-Markov Models toward Applications*, 1 ed.: Springer, 2008.
[38] S. V. Vaseghi, “State duration modelling in hidden Markov models ” *Signal Processing*, vol. 41, no. 1, pp. 31-41, 1995.
[39] M.-Y. Chen, A. Kundu, and S. N. Srihari, “Variable duration hidden Markov model and morphologicalsegmentation for handwritten word recognition,” *Image Processing, IEEE Transactions*, vol. 4, no. 12, pp. 1675-1688, 1995.
[40] S. D. Bay, and M. J. Pazzani, “Detecting change in categorical data: mining contrast sets,” in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, California, United States, 1999, pp. 302-306.
[41] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, “Rule-based anomaly pattern detection for detecting disease outbreaks,” in Eighteenth national conference on Artificial intelligence, Edmonton, Alberta, Canada, 2002, pp. 217-223.
[42] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, “What's Strange About Recent Events (WSARE): An Algorithm for the Early Detection of Disease Outbreaks,” *J. Mach. Learn. Res.*, vol. 6, pp. 1961-1998, 2005.
[43] K. Das, J. Schneider, and D. B. Neill, “Anomaly pattern detection in categorical datasets,” in Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, Las Vegas, Nevada, USA, 2008, pp. 169-176.
[44] A. D. Josep Roure, Jeff Schneider "A Study into Detection of Bio-Events in Multiple Streams of Surveillance Data," *Intelligence and Security Informatics: Biosurveillance*, LNCS, D. Z. e. al., ed., pp. 124-133: Springer, 2007.
[45] D. B. N. Robin Sabhnani, Andrew W. Moore, Artur W. Dubrawski, Weng-Keen Wong “Efficient Analytics for Effective Monitoring of Biomedical Security,” *IEEE International Conference on Information and Automation*, 2005.
[46] F. McArthur-Rouse, “Critical care outreach services and early warning scoring systems: a review of the literature,” *Journal of Advanced Nursing*, vol. 36, no. 5, pp. 696-704, 2001.

[47] J. H. Heather Duncan, Christopher S. Parshuram, "The pediatric early warning system score: A severity of illness score to predict urgent medical need in hospitalized children," *Journal of Critical Care,* vol. 21, no. 3, pp. 271-278, 2006.
[48] M. M. Pollack, K. M. Patel, and U. E. Ruttimann, "PRISM III: an updated Pediatric Risk of Mortality score," *Crit Care Med,* vol. 24, no. 5, pp. 743-52, May, 1996.
[49] F. Shann, G. Pearson, A. Slater, and K. Wilkinson, "Paediatric index of mortality (PIM): a mortality prediction model for children in intensive care," *Intensive Care Med,* vol. 23, no. 2, pp. 201-7, Feb, 1997.
[50] J. E. Zimmerman, A. A. Kramer, D. S. McNair, and F. M. Malila, "Acute Physiology and Chronic Health Evaluation (APACHE) IV: hospital mortality assessment for today's critically ill patients," *Crit Care Med,* vol. 34, no. 5, pp. 1297-310, May, 2006.
[51] H. Duncan, J. Hutchison, and C. S. Parshuram, "The Pediatric Early Warning System score: a severity of illness score to predict urgent medical need in hospitalized children," *J Crit Care,* vol. 21, no. 3, pp. 271-8, Sep, 2006.
[52] L. Tume, "The deterioration of children in ward areas in a specialist children's hospital," *Nurs Crit Care,* vol. 12, no. 1, pp. 12-9, Jan-Feb, 2007.
[53] M. H. Gorelick, M. W. Stevens, T. R. Schultz, and P. V. Scribano, "Performance of a novel clinical score, the Pediatric Asthma Severity Score (PASS), in the evaluation of acute asthma," *Acad Emerg Med,* vol. 11, no. 1, pp. 10-8, Jan, 2004.
[54] S. Jacobs, G. Shortland, J. Warner, A. Dearden, P. S. Gataure, and J. Tarpey, "Validation of a croup score and its use in triaging children with croup," *Anaesthesia,* vol. 49, no. 10, pp. 903-6, Oct, 1994.
[55] "HealthVault," https://www.healthvault.com/.
[56] W. Lup Low, M. Li Lee, and T. Wang Ling, "A knowledge-based approach for duplicate elimination in data cleaning," *Information Systems,,* vol. 26, no. 8, 2001.
[57] T. Wang, and D. Dennis, "Normalizing database normalization definitions in AIS text books," *Review of business information systems,* vol. 15, no. 1, 2011.
[58] J. J. Tamilselvi, and Saravanan, "Detection and elimination of duplicate data using token-based method for a data warehouse: a clustering based approach," *International Journal of Computational Intelligence Research,* vol. 5, no. 2, 2009.
[59] J. McKendrick, "Big data, big issues: the year ahead in information management," *Database Trends & Applications,* 2010.
[60] D. Zhang, "Inconsistencies in big data," in Cognitive Informatics & Cognitive Computing (ICCI*CC), New York, NY, 2013.
[61] D.-G. Kweon, and S.-H. S. d. m. Choi, "Journal of the Korea Academia-Industrial cooperation Society," vol. 10, no. 7, 2009.
[62] M. Wolff, and C. Tschope, "Pattern recognition for sensor signals," in IEEE Sensors, Christchurch, 2009.
[63] Y. Yao, and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," vol. 31, no. 3, 2002.
[64] E. Van den Berg, *Sparse and Redundant Representations,* SIAM Review, 2011.
[65] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS),* vol. 30, no. 1, 2005.

[66] T. T. Fawcett, III, "Using a distributed reactive algorithm to control an arbitrary number of collaborative hyper-redundant serial manipulators in real time," UMI Dissertations Publishing, 2012.
[67] J. Gao, L. Guibas, N. Milosavljevic, and J. Hershberger, "Sparse Data Aggregation in Sensor Networks," in Proceedings of the 6th international conference on information processing in sensor networks, New York, NY, USA, 2007.
[68] J. Li, and S. Cheng, "Approximate Aggregation Techniques for Sensor Databases," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, no. 3, 2004.
[69] S. Ghassempour, F. Girosi, and A. Maeder, "Clustering Multivariate Time Series Using Hidden Markov Models," *International journal of environmental research and public health,* vol. 11, no. 3, 2014.
[70] S.-M. H. Da-Ren Chen, "Mining Frequent Patterns in Wireless Sensor Network Configurations," *Lecture Notes in Electrical Engineering Volume 309*, pp. 733-745, 2014.
[71] Z. Hajihashemi, and M. Popescu, "Detection of abnormal sensor patterns in eldercare," in E-Health and Bioengineering Conference, 2013.
[72] F. Chamroukhi, A. Same, P. Aknin, and G. Govaert, "Model-based clustering with Hidden Markov Model regression for time series with regime changes," in International Joint Conference on Neural Networks, 2011.
[73] A. Singhal, and D. E. Seborg, "Clustering of multivariate time-series data," in Proceedings of the 2002 American Control Conference, 2002.
[74] J. Wu, *Advances in K-means Clustering*: Springer Verlag, 2012.
[75] E. S. Fung, W. K. Ching, S. Chu, M. K. Ng, and W. Zang, "Multivariate Markov chain models," in IEEE International Conference on Systems, Man and Cybernetics, Hammamet, Tunisia, 2002.
[76] X. Ji, J. Bailey, and K. Ramamohanarao, "Classifying proteins using gapped Markov feature pairs," *Neurocomputing,* vol. 73, no. 13, 2010.
[77] M. Studer, G. Ritschard, A. Gabadinho, and N. S. Muller, "Discrepancy Analysis of State Sequences," *Sociological Methods & Research,* vol. 40, no. 3, 2011.
[78] X. Fang, G. Lu, and S. Zhang, "Sequence Comparison using Multi-Order Markov Chains," in International Conference on Bioinformatics and Biomedical Engineering, 2010.
[79] P. Vassiliadis, A. Simitsis, and E. Baikousi, "A taxonomy of ETL activities," in Proceeding of the ACM twelfth international workshop on Data warehousing and OLAP, Hong Kong, China, 2009, pp. 25-32.
[80] B.-G. Itzik, *Microsoft SQL Server 2008 T-SQL Fundamentals*: Microsoft Press, 2008.
[81] D. D. Fraser, R. N. Singh, and T. Frewen, "The PEWS score: potential calling criteria for critical care response teams in children's hospitals," *J Crit Care,* vol. 21, no. 3, pp. 278-9, Sep, 2006.
[82] D.-I. Curiac, G. Vasile, O. Banias, C. Volosencu, and A. Albu, "Bayesian network model for diagnosis of psychiatric diseases," in Information Technology Interfaces, 2009, Dubrovnik, 2009.
[83] A. M. Weng-Keen Wong, Gregory Cooper, Michael Wagner, "Bayesian Network Anomaly Pattern Detection for Disease Outbreaks," in Proceedings of the Twentieth International Conference on Machine Learning, 2003.

[84] G. C. Weng-Keen Wong, D.Dash, J. Levander, J.Dowling, W. Hogan, M.Wagner, “Use of multiple data streams to conduct Bayesian biologic surveillance.,” *MMWR. Morbidity and mortality weekly report*, 2005.
[85] K. M. Al-Aidaroos, A. A. Bakar, and Z. Othman, “Medical Data Classification with Naive Bayes Approach,” *Information Technology Journal*, 2012.
[86] S. Moein, *Medical Diagnosis Using Artificial Neural Networks*: Washington University in Saint Louis, USA, 2014.
[87] R. W. Brause, "Medical Analysis and Diagnosis by Neural Networks," *Medical Data Analysis*, pp. 1-13: Springer Berlin Heidelberg, 2001.
[88] D. Garets, and M. Davis, “Electronic Patient Records EMRs and EHRs,” *Healthcare Informatics*, 2005.
[89] Z. V. Edmonds, W. R. Mower, L. M. Lovato, and R. Lomeli, “The reliability of vital sign measurements,” *Annals of Emergency Medicine*, vol. 39, no. 3, pp. 233-237, 2002.

APPENDIX 1: TRAINING DATA CHARACTERISTIC

Number of Records	0-0.5y	0.5-1y	1-3y	3-6y	6-8y	8-y	
Asthma	0	137	854	1365	868	2216	5440
Cold	298	16	37	32	0	0	383
Cough	570	0	277	445	34	397	1723
Dyspnea	142	293	444	409	221	432	1941
Respiratory Distress	1253	2126	2573	2233	1054	1684	10923
Respiratory problem	2874	585	1839	2061	283	1773	9415
Stridor	245	86	37	0	0	27	395
Wheeze	491	857	2512	1382	553	1831	7626
	5873	4100	8573	7927	3013	8360	37846

Chief Complaint	0-0.5y	0.5-1y	1-3y	3-6y	6-8y	8-y	Total
Asthma	0	2	27	37	23	51	140
Cold	11	1	2	2	0	0	16
Cough	12	0	8	11	2	8	41
Dyspnea	4	8	13	8	3	11	47
Respiratory Distress	23	16	43	25	12	19	138
Respiratory problem	39	15	52	36	8	25	175
Stridor	2	3	3	0	0	2	10
Wheeze	9	22	78	43	15	43	210
	100	67	226	162	63	159	777

Age group
0-0.5y
0.5-1y
1-3y
3-6y
6-8y
8-y

Symptom group (6 vital signs of interest)
Asthma
Cold (Cold=URI=Bronchiolitis)
Cough
Dyspnea (Dyspnea=Shortness of breath)
Respiratory Distress
Respiratory problem
Stridor
Wheeze (Wheeze=Wheezing)

APPENDIX 2: VITAL STATE MAPPINGS

Respiratory Rate					
Age	RR_Critical_Low	RR_low	RR_Normal	RR_High	RR_Critical_High
State Value	-2	-1	0	1	2
0 (0-180 days)	0-20	21-29	30-60	61-70	71-
0.5-1 (181-365 days)	0-15	16-23	24-40	41-50	51-
1-3y (366-1095 days)	0-15	16-23	24-40	41-50	51-
3-6y (1096-2190 days)	0-15	16-19	20-30	31-40	41-
6-8y (2191-2920 days)	0-14	14-17	18-25	26-34	35-
8-10y (2921-3650 days)	0-13	14-17	18-25	26-34	35-
10-12y (3651-4380 days)	0-12	13-15	16-20	21-30	31-
12-15y (4381-5475 days)	0-9	10-13.	14-20	21-30	31-
15-18y (5476-6570 days)	0-8	9-11.	12-20.	21-30	31-
>18 yrs (6571- days)	0-8	9-11.	12-28.	19-28	29-

SpO2 Bedside Monitor				
Age	SpO2_Critical_low	SpO2_low	SpO2_low_normal	SpO2_normal
State Value	-3	-2	-1	0
0 (0-180 days)	0-87	88-92	93-96	97-100
0.5-1 (181-365 days)	0-87	88-92	93-96	97-100
1-3y (366-1095 days)	0-87	88-92	93-96	97-100
3-6y (1096-2190 days)	0-87	88-92	93-96	97-100
6-8y (2191-2920 days)	0-87	88-92	93-96	97-100
8-10y (2921-3650 days)	0-87	88-92	93-96	97-100
10-12y (3651-4380 days)	0-87	88-92	93-96	97-100
12-15y (4381-5475 days)	0-87	88-92	93-96	97-100
15-18y (5476-6570 days)	0-87	88-92	93-96	97-100
>18 yrs (6571- days)	0-87	88-92	93-96	97-100

Oxygen Flow Rate				
Age	**O2_FR_normal**	**O2_FR_Low**	**O2_FR_Medium**	**O2_FR_High**
State Value	**0**	**1**	**2**	**3**
0 (0-180 days)	0 / no value	1-2.	3-4.	5-
0.5-1 (181-365 days)	0 / no value	1-2.	3-4.	5-
1-3y (366-1095 days)	0 / no value	1-2.	3-4.	5-
3-6y (1096-2190 days)	0 / no value	1-2.	3-4.	5-
6-8y (2191-2920 days)	0 / no value	1-2.	3-4.	5-
8-10y (2921-3650 days)	0 / no value	1-2.	3-4.	5-
10-12y (3651-4380 days)	0 / no value	1-2.	3-4.	5-
12-15y (4381-5475 days)	0 / no value	1-2.	3-4.	5-
15-18y (5476-6570 days)	0 / no value	1-2.	3-4.	5-
>18 yrs (6571- days)	0 / no value	1-2.	3-4.	5-

Heart Rate					
Age	**HR_Critical_Low**	**HR_low**	**HR_Normal**	**HR_High**	**HR_Critical_High**
State Value	**-2**	**-1**	**0**	**1**	**2**
0 (0-180 days)	0-79	80-89	90-180	181-200	201-
0.5-1 (181-365 days)	0-74	75-84	85-170	171-195	196-
1-3y (366-1095 days)	0-64	65-79	80-140	141-160	161-
3-6y (1096-2190 days)	0-64	65-79	80-130	131-155	156-
6-8y (2191-2920 days)	0-54	55-69	70-120	121-145	146-
8-10y (2921-3650 days)	0-54	55-69	70-110	111-135	136-
10-12y (3651-4380 days)	0-49	50-64	65-110	111-135	136-
12-15y (4381-5475 days)	0-45	46-59	60-110	111-135	136-
15-18y (5476-6570 days)	0-45	46-54	55-100	101-125	126-
>18 yrs (6571- days)	0-40	41-49	50-90	91-120	121-

Systolic BP					
Age	**SBP_Critical_Low**	**SBP_low**	**SBP_Normal**	**SBP_High**	**SBP_Critical_High**
State Value	**-2**	**-1**	**0**	**1**	**2**
0 (0-180 days)	0-42	43-49	50-70	71-85	86-
0.5-1 (181-365 days)	0-52	53-64	65-106	107-118	119-
1-3y (366-1095 days)	0-59	60-71	72-110	111-122	123-
3-6y (1096-2190 days)	0-64	65-77	78-114	115-129	130-
6-8y (2191-2920 days)	0-68	69-79	80-116	117-131	132-
8-10y (2921-3650 days)	0-71	72-83	84-122	123-137	138-
10-12y (3651-4380 days)	0-76	77-89	90-130	131-144	145-
12-15y (4381-5475 days)	0-81	82-93	94-136	137-150	151-
15-18y (5476-6570 days)	0-84	85-99	100-142	143-154	155-
>18 yrs (6571- days)	0-87	88-103	104-148	149-156	157-

Diastolic BP					
Age	**DBP_Critical_Low**	**DBP_low**	**DBP_Normal**	**DBP_High**	**DBP_Critical_High**
State Value	**-2**	**-1**	**0**	**1**	**2**
0 (0-180 days)	0-34	35-44	45-55	56-65	66-
0.5-1 (181-365 days)	0-47	48-54	55-65	66-75	76-
1-3y (366-1095 days)	0-47	48-54	55-70	71-85	86-
3-6y (1096-2190 days)	0-49	50-59	60-80	81-90	91-
6-8y (2191-2920 days)	0-54	55-64	65-82	83-95	96-
8-10y (2921-3650 days)	0-54	55-64	65-85	86-100	101-
10-12y (3651-4380 days)	0-59	60-69	70-90	91-110	111-
12-15y (4381-5475 days)	0-59	60-69	70-90	91-110	111-
15-18y (5476-6570 days)	0-59	60-69	70-90	91-110	111-
>18 yrs (6571- days)	0-59	60-69	70-90	91-110	111-

APPENDIX 3: IMPLEMENTATION OF TYCHO DATA WAREHOUSE

In order to perform the data transformation and adaption process, we implement this procedure in Microsoft SQL Server Integration Service (SSIS) package which is available in the SQL Server 2008 Developer/Enterprise edition. SSIS is a data migration and integration tool. It is compatible with SQL Server database, analysis and reporting tools. It supports heterogeneous data –sources based processing. It is visual, hence, more intuitive; component based (modular). These components, blocks in Figure A are both built-in (drag-n-drop) and can be programmed by the users. There are two types of workflow in SSIS: control flow and data flow. Control flow is a higher level design and it contains all tasks for the package while data flow is detail design and it is under control flow. Figure A is the (control flow) design view of our SSIS package. In this control flow, each block represent a "task" and could be expanded as the data flow (the detail procedure) by double clicking the task. This design (Figure A), the bare-bone cleaning algorithm in an intuitive manner and it is based on the conceptual design we have mention in Figure 46: A conceptual view of the data cleaning process in Section 6.1. The package loops through all the excel-sheets. For each sheet, it parses title, body and header for location, time, disease, outcome and associated value. In the following subsection we will start explain the control flow, the overall design of our SSIS package and also will explain the data flow of each block.

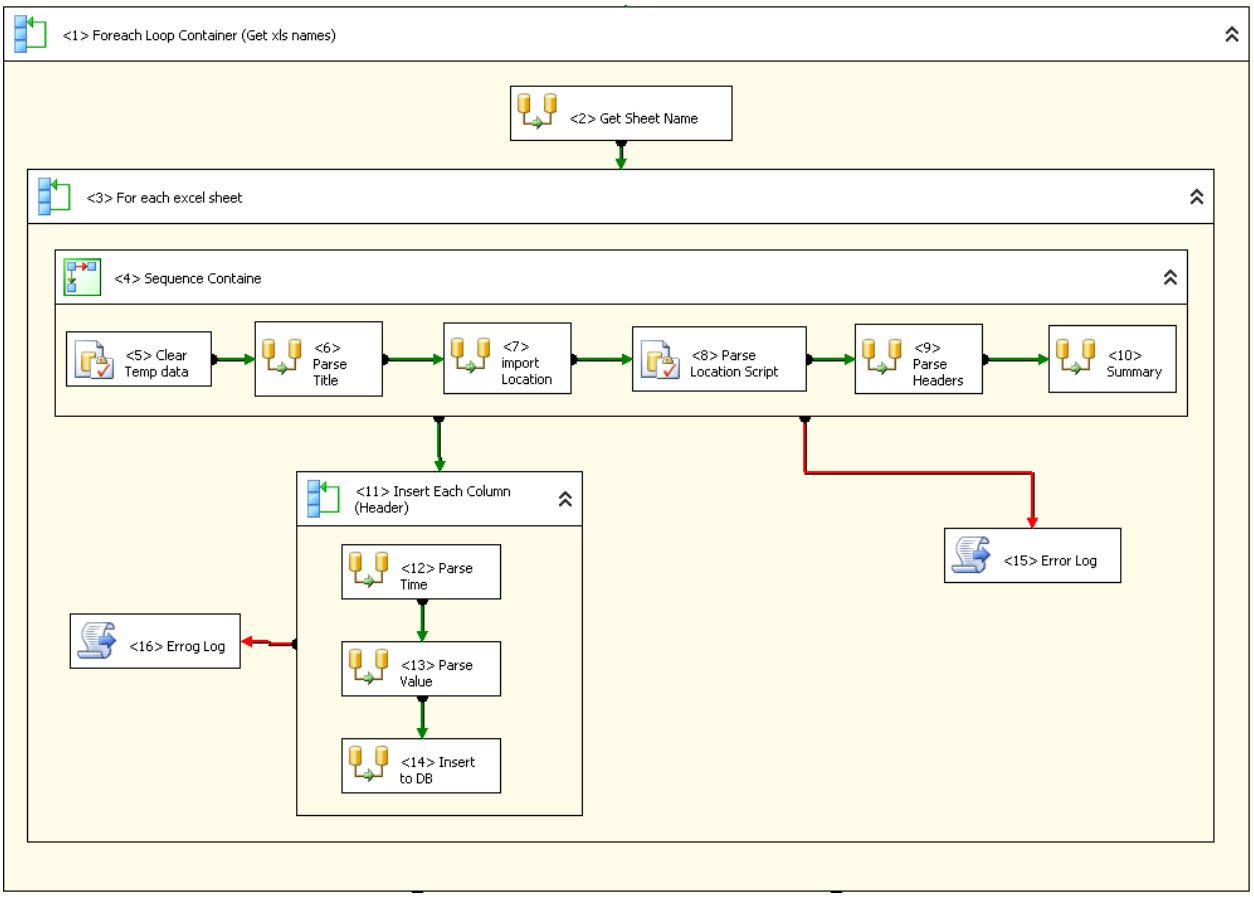


Figure A: A snapshot of the implemented SSIS package (control flow)

<1> In this step (the "For each loop control"), SSIS generates a list of all excel file names in a given folder and all excel files will processed based on the order of its file name.

<2> After having all excels file names, this step is programmed to acquire all sheet names for a given excel file.

<3> We now start processing each individual excel sheet by another "For each loop control" in the giving excel file.

<4> This block is a sequential container, which is used to group other blocks (block <5> to <10>) and ensures that all other blocks in the control flow will not be processed until all blocks inside sequential control is completed.

<5> When process each excel sheet, there are some intermediate data will be generated. This step is simply delete those intermediate data before processing a new excel sheet.

<6> In this step, we process title part of excel file. Figure B is the detail description (data flow) of the title process task. In this step, we first get the title string from excel sheet and apply pre clean procedure which is to remove some symbols or replace common typos in the title to increase the accuracy of later process. After that, in the block [Parse_Year_month_Date], we use regular expression to find time part from the title and in [Dis_main] and [Dis_Sub] fuzzy lookup is applied to find the main disease name and disease subcategory.

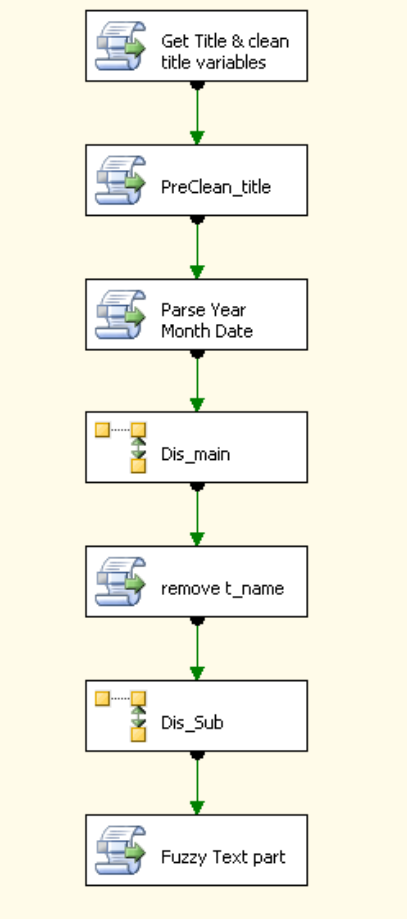


Figure B: The data flow of title process

<7> The location part, the first column in excel sheet is imported to an intermediate table in the database.

<8> To ensure the location has a standard name even with typo, a TSQL script is applied to parse the location based on a location reference table.

<9> The step classifies headers to different categories. Figure C is the initial step of the data flow for of header process. For headers, we first collect import keywords such as

- Disease name [Disease]. E.g. measles, AIDS…etc.
- Disease subcategory [find subcategory]. E.g. Type A, Type B, infection, virus…etc.
- Special keywords [parse keywords]. E.g. case, death, place, population, area, place…etc.
- Time keywords [Find Time Part]. E.g. year, week, weekend, week end …etc.
- Aggregation type [find aggregation type]: E.g. total, cumulative, cum, max, med, median…etc.
- Date part [Find date part]: in the case that title also contains number (date), then we use regular expression to parse the date information

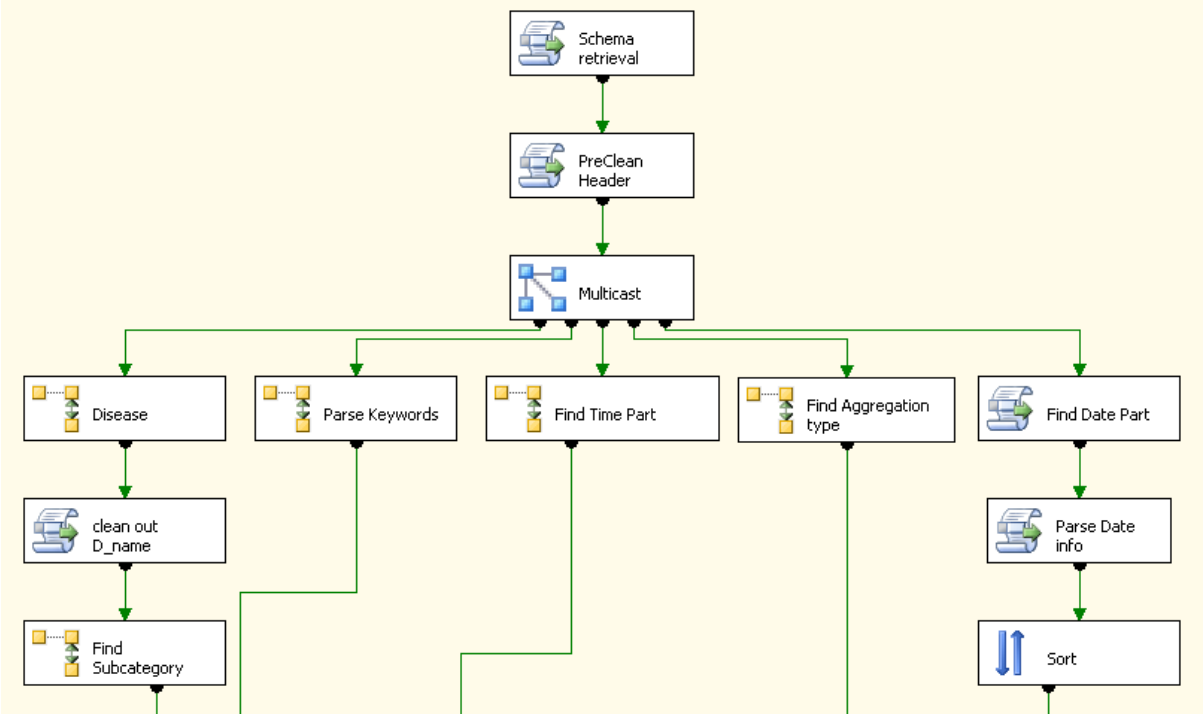


Figure C: The data flow of header process (initial part)

After the above information is collected, we use condition split control to separate those headers to different category as shown in Figure D. Those categories are

- Location column [Loc]: it describes the header is a location header. In other words, all cells below this header are locations.
- Time column [Time]: this header represent this column is time column.
- Disease column [Dis]: this header has information about certain disease and all cells below are number which is used to describe the amount of either death or case happen in that disease.
- Case or death column [case or death]: sometimes header only says case/death without disease name. In this case, the disease name is in the title. Since this type of header need find get disease information from title, we separate it from regular disease column.
- Else column [Else]: for information which is not directly related to disease, we put them in to else category such as population.
- Finally, the unparsed column, which is not belong to any categories we have mentioned will be store in the database as for further analyze. An error log will also be generated and stored if the any error arises during this header process procedure.

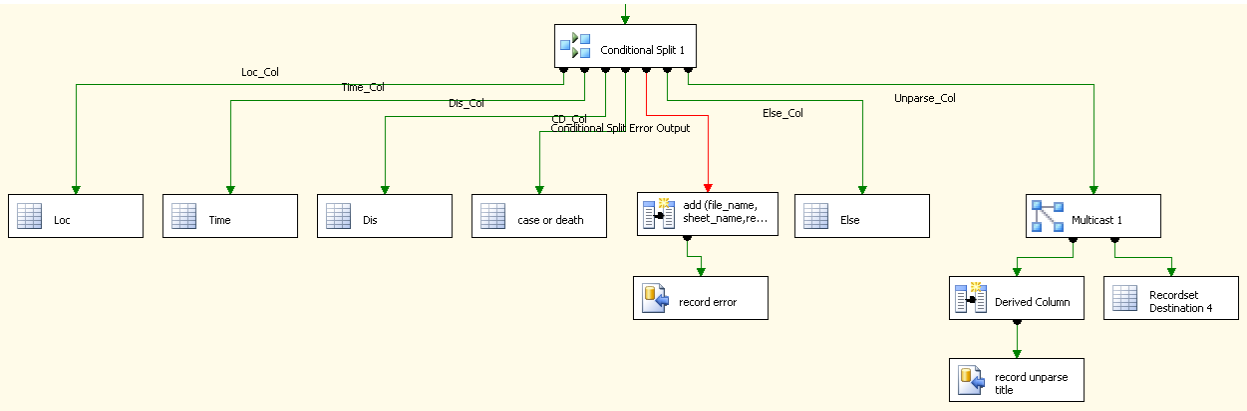


Figure D: The data flow of header process (final part)

Note that based on our experience, the fuzzy look up can successfully parse around 90% of disease name. The error happens when the string is long and hence increases the noise level for fuzzy look up. Since both title and header could contain disease name. In this case, we use disease name from header.

<10> After headers are classified, in the summary block, we take appropriate actions based on different header categories. For example, we know that there will be numbers in disease column and case/death column. All cells in the time column contain time string. Therefore, we know what function should be applied to what columns.

<11> ~ <14> after having information from step <10>, we use for each loop control (block <11>) to parse time (to data format) in block <12>, value (block <13>), and insert the result to database (block <14>).

Note that the time information could appear from title, time column, and header. The time priority logic is (time in header) > (time in time column) > (time in title)

<15> ~ <16> record errors which happen duration the process. This error often is due to excel format error.

The validated data was then loaded to a database. Table 1 summarizes the loading statistics for the Tycho database. The database was loaded from 16 batches each containing a number of excel-sheets (total 38,932 excel sheets). More than 25 million tuples were loaded in the database.

Loading Statistics:

Batch	Number of files	Number of sheets	Number of data rows
1	47	337	206000
2	496	856	455892
3	521	4388	1560211
4	508	3922	1293274
5	522	6131	2670207
6	522	5243	2173204
7	517	3473	3960916
8	517	1561	1151614
9	406	1317	1548759
10	462	1837	1439066
11	295	1191	1044489
12	420	1721	1861490
13	478	3949	3238394
14	105	1050	1111946
15	181	648	499625
16	353	1308	1167299
SUM	6350	38932	25382386

Table A: Loading statistics for the Tycho database

Disease	number of weekly reports
diphtheria	370135
scarlet fever	334549
typhoid fever	320322
tuberculosis	307453
smallpox	296274
measles	257745
pneumonia	196529
influenza	190046
whooping cough	155223
poliomyelitis	100398
pneumonia and influenza	88950
meningitis	74213
encephalitis	61771
mumps	50839
chickenpox	50735
enteric fever	40149
pellagra	31844
meningococcal disease	30223
phthisis pulmonalis	22208
hepatitis	5919
lobar	4484
typhus fever	2781

	week_day	CNT	Percent
1	1	360645	97.436070622880
2	2	3520	0.951004363272
3	3	1591	0.429843165331
4	4	423	0.114282626609
5	5	153	0.041336269199
6	6	852	0.230186283383
7	7	2951	0.797276669323

Therefore, we design each disease has its own table in the data warehouse and the following is the snapshot of our current data warehouse.

	R_id	Loc_city	Loc_state	value	Reported_Outcome	From_date	To_date	meta_id	Pb_date	week_day	dup	assoc_1	assoc_2
1	21805	CAMDEN	NJ	1	death	1905-10-08	1905-10-14	03-1905-20-42-01-13-016	1905-10-20	1	1	NULL	NULL
2	21806	NORRISTOWN	PA	2	death	1905-10-09	1905-10-15	03-1905-20-42-02-13-041	1905-10-20	2	1	NULL	NULL
3	21807	LAWRENCE	MA	1	death	1905-10-11	1905-10-17	03-1905-20-43-01-13-045	1905-10-27	4	1	NULL	NULL
4	21808	CLEVELAND	OH	6	death	1905-10-14	1905-10-20	03-1905-20-44-01-13-033	1905-11-03	7	1	NULL	NULL
5	21809	CINCINNATI	OH	1	death	1905-10-14	1905-10-20	03-1905-20-43-01-13-019	1905-10-27	7	1	NULL	NULL
6	21810	DUNKIRK	NY	1	death	1905-10-15	1905-10-21	03-1905-20-44-01-13-038	1905-11-03	1	1	NULL	NULL
7	21811	SOMERVILLE	MA	1	death	1905-10-15	1905-10-21	03-1905-20-43-02-13-004	1905-10-27	1	1	NULL	NULL
8	21812	NEW BEDFORD	MA	2	death	1905-10-15	1905-10-21	03-1905-20-43-01-13-059	1905-10-27	1	1	NULL	NULL
9	21813	TRENTON	NJ	1	death	1905-10-15	1905-10-21	03-1905-20-43-02-13-012	1905-10-27	1	1	NULL	NULL
10	21814	WHEELING	WV	2	death	1905-10-15	1905-10-21	03-1905-20-43-02-13-015	1905-10-27	1	1	NULL	NULL

APPENDIX 4: QUERY OPTIMIZATION AND TYCHO WEB APPLICATION

We developed a Tycho data warehouse for the above mentioned Tycho database. In the Tycho data warehouse the data is organized in a canonical form. This allows us to run analytical queries efficiently on the data and provides an opportunity to optimize data with respect to the desired performance.

Figure E shows the schema of the Tycho data warehouse. It consists of following tables:

Fact(R_id, D_id, L_id, From_date, To_date, value, Agg_id, Age_cat_id, Reported_outcome, Meta_id)

Summary(R_id, L_id, From_date, To_date, Agg_id, Age_cat_id, Reported_outcome, Meta_id)

dim_Locations(Loc_id, Loc_org, Loc_city, Loc_state, Loc_county, Loc_region, Loc_country)

dim_Disease(id, D_name, D_s_name)

dim_AggregationType(Agg_id, Agg_desc)

dim_AgeCategories(Age_cat_id, From_age, To_age)

dim_LocHistSyn(Old_L_id, New_L_id, Rename_date)

dim_DisHistSyn(Old_D_id, New_D_id, Rename_date) where,

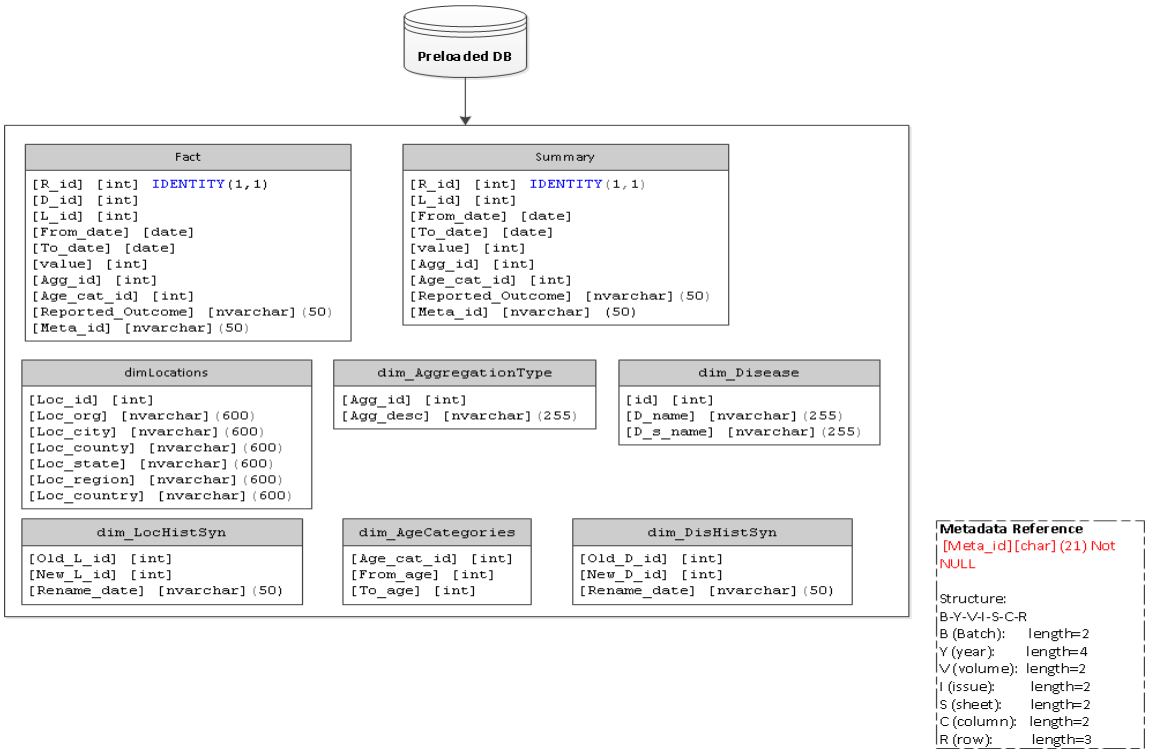


Figure E: Tycho Data-warehouse

R_id is an id for a record, D_id is disease id, L_id is Location id, From_date and To_date are the start and end times for the reporting period, value is the number of cases/deaths, Agg_id is the id for the aggregation type, Age_cat_id is the id for each age category, Reported_outcome is either a case or death, and Meta_id is each data's reference to a particular cell location in the excel sheets. Meta_id is comprised of Batch, Year, Volume, Issue, Sheet, Column and Row. Loc_org is the original unclean location and Loc_city, Loc_state, Loc_county, Loc_region, Loc_country are cleaned city, state, county, region and country for the respective original unclean location. D_name is the disease name and D_s_name is the disease subcategory. Agg_desc is the description of an aggregation type e.g. median or sum etc. From_age and To_age are the associated age categories for the people who were reported to have a disease. Old_L_id and New_L_id are the old and new location ids referring to an old and a new name for a location that was renamed on Rename_date.

As an example, the data warehouse can extract all records for measles using the following query:

```
SELECT [From_date] ,[To_date], [Reported_Outcome], [value], L.Loc_city, L.Loc_state, L.Loc_county, L.Loc_region
FROM [TychoDWH].[dbo].[Fact] F, [TychoDWH].[dbo].[dim_Disease] D, [TychoDWH].[dbo].[dimLocations] L
WHERE F.D_id=D.id and F.L_id=L.Loc_id and D.D_name = 'measles'
```

The data warehouse performance is optimized by building indices, materialized views and indices over materialized views. Indices make data access fast. Materialized views are maintained as pre-computed solutions to some queries. Queries are allowed over materialized views as well. Indices over materialized views makes data access from materialized views much faster.

Query	Opt. Strategy	Response time	Num. of Records
Find All Diseases	1	124	63
	1_Idx	1	63
	1_V	0	63
	1_IdxV	0	63
Aggregate cases per disease	2	100	63
	2_Idx	40	63
	2_V	24	63
	2_IdxV	0	63
Diseases per state (join)	3	70	4739
	3_Idx	0	4739
	3_V	0	4739
	3_IdxV	0	4739
List ordered time, city, state, disease	4	248	9698769
	4_Idx	263	9698769
	4_V	92	9698769
	4_IdxV	89	9698769
DWH Query (join Fact, Location)	5	12	14274
	5_Idx	1	14274
	5_V	0	14274
	5_IdxV	0	14274

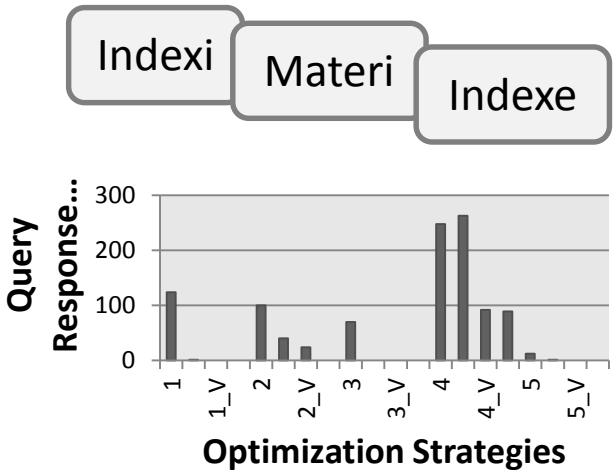


Figure F: Optimization with indices, materialized views and indices over materialized views

Figure F shows the effect of optimization strategies on data access. In the figure, _Idx implies that the query was run over an indexed data set, _V refers to the query being run over a materialized view and _IdxV means that the query was executed over a view with an index. On an average, the indices decrease the query response time, views decrease it even more, and indexed views take care of most of time consuming queries. All of the above functionalities are fully implemented in our website at http://tycho.exp.sis.pitt.edu/. This website is open to the

public and providing researchers and practitioners with the data visualization of Tycho data such as the Figure G below.

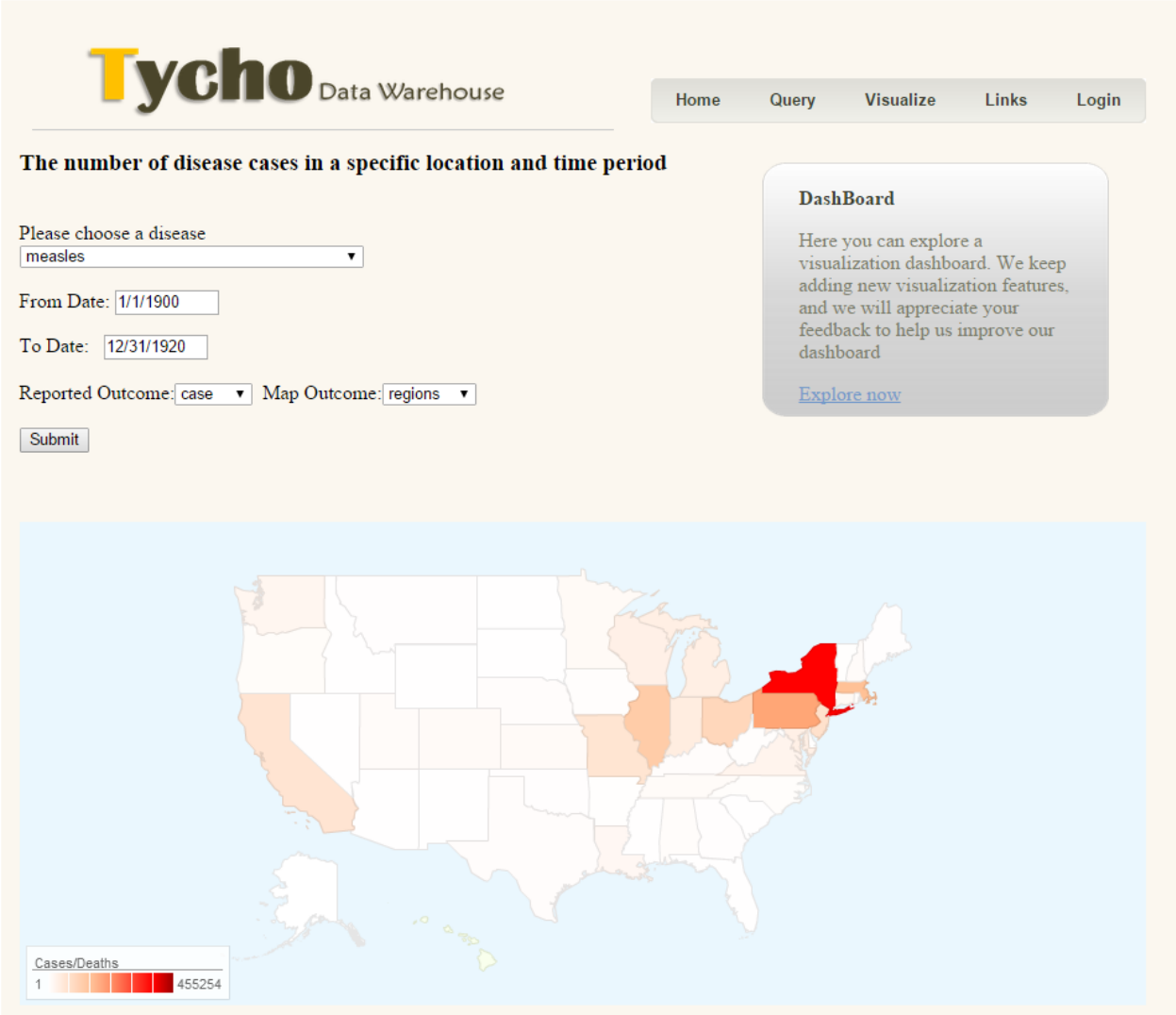


Figure G: Example of Tycho data visualization at http://tycho.exp.sis.pitt.edu/