

Efficient In-Database Patient Similarity Analysis for Personalized Medical Decision Support Systems

Araek Tashkandi^{a,1,*}, Ingmar Wiese^a, Lena Wiese^a

^a*Georg-August-University Goettingen, Institute of Computer Science, Goldschmidtstr. 7, 37077 Goettingen, Germany.*

Abstract

Patient similarity analysis is a precondition to apply machine learning technology on medical data. In this sense, patient similarity analysis harnesses the information wealth of electronic medical records (EMRs) to support medical decision making. A pairwise similarity computation can be used as the basis for personalized health prediction. With n patients the amount of $\binom{n}{2}$ similarity calculations is required. Thus, analyzing patient similarity leads to data explosion when exploiting big data. By increasing the data size the computational burden of this analysis increases. A real-life medical application may exceed the limits of current hardware in a fairly short amount of time. Finding ways to optimize patient similarity analysis and handling this data explosion is the topic of this paper.

Current implementations for patient similarity analysis require their users to have knowledge of complex data analysis tools. Moreover, data pre-processing and analysis are performed in synthetic conditions: the data are extracted from the EMR database and then the data preparation and analysis are processed in external tools. After all of this effort the users might not experience a superior performance of the patient similarity analysis. We propose methods to optimize the patient similarity analysis in order to make it scalable to big data. Our method was tested against two real datasets and a low execution time was accomplished. Our result hence benefits a comprehensive medical decision support system. Moreover, our implementation comprises a balance between performance and applicability: the majority of the workload is processed within a database management system to enable a direct implementation on an EMR database.

Keywords: Patient Similarity Analysis, Medical Data Analysis, Medical Decision Support System, Personalized Medicine, Cosine Similarity, MIMIC-III, Column Stores

1. Introduction

The increasing amount and scale of available electronic medical records (EMRs) raises the interest to gain valuable knowledge from them. Knowledge discovered from EMRs can favor patients

*Corresponding author

Email addresses: araek.tashkandi@cs.uni-goettingen.de (Araek Tashkandi), asatashkandi@kau.edu.sa (Araek Tashkandi), ingmar.wiese@stud.uni-goettingen.de (Ingmar Wiese), wiese@cs.uni-goettingen.de (Lena Wiese)

¹Also affiliated to King Abdulaziz University, Faculty of Computing and Information Technology, 21589 Jeddah, Kingdom of Saudi Arabia.

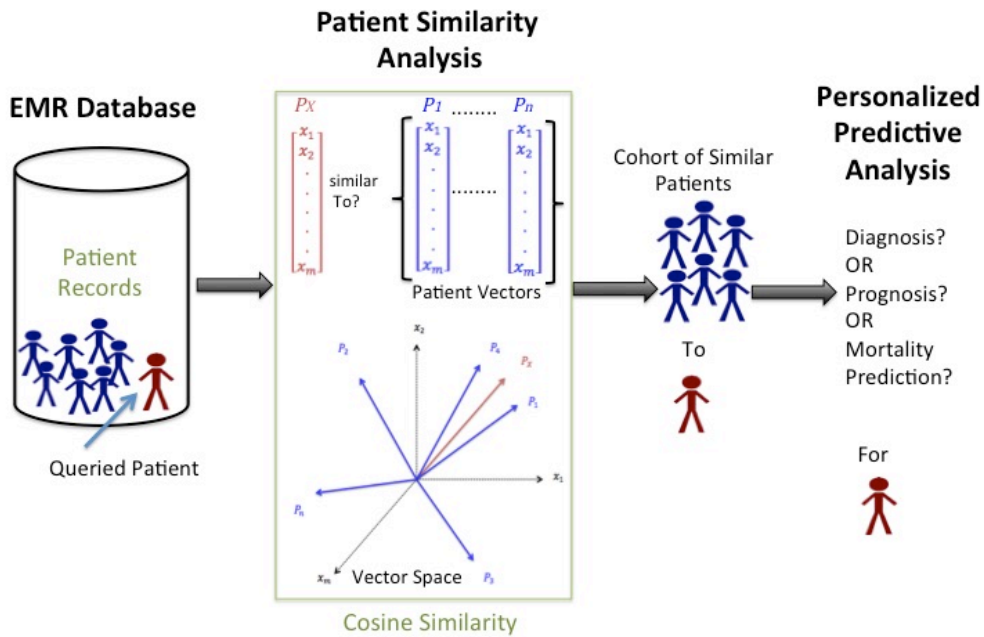


Figure 1: Patient similarity analysis process for medical decision support system

and medical staff by reducing costs and even saving lives. Patient similarity analysis is the basis for further medical decisions. Finding a group of patients having similar features (for example, similar lab results or similar diagnoses), helps medical staff with treatment decisions or health predictions. This leads to the development of a personalized predictive model that can be incorporated into a medical decision support system. In other words, patient similarity analysis is a decisive component for diagnosis or prognosis prediction for an index patient from clinically similar patients.

The process of patient similarity analysis for medical decision support is as follows: patient data from an EMR database are used in the patient similarity analysis method. The Patient Similarity Metric (PSM) could be one of different algorithms that are described by [1, 2] such as neighborhood-based algorithms and distance-based similarity metrics as Euclidean distance. Computation of the PSM defines a cohort of patients similar to a given index patient. The data of the similar patients are used to predict the personalized treatment or the future health of the index patient. A schematic overview is given in Figure 1.

Accuracy of the data is one requirement for correct medical data analysis; as [3] put it: “Current research is exploring ways to leverage the enhanced completeness and expressivity of modern electronic medical records (EMRs) in order to improve prediction accuracy”. Most of the current work is on the accuracy of PSMs while at the same time reporting an increase in computational load as [4, 5]. The cause of this computational burden comes from the *pairwise* similarity computation of a PSM which is the basis for personalized health predictions. With n patients the amount of $\binom{n}{2}$ similarity calculations is required. Thus, analyzing patient similarity produces data explosion when handling big data; this is a common observation also stated in [4]: “The computational burden depends on the complexity of the PSM calculation, and the size of the EMR database”. By increasing the data size the computational burden of this analysis increases. This personalized

prediction model produces high computational complexity and therefore entails longer run-times. Thus, efficiency of analyses is another major requirement to enable personalized medicine. This paper sheds light on optimizing the performance of pairwise patient similarity analysis.

Current implementations for patient similarity analysis use different tools for data pre-processing and analysis. Moreover, works on optimizing patient similarity analysis for medical data like [6] commonly use the power of data analysis tools – but thereby paying the latency cost of accessing and processing the EMR data outside the database management system (DBMS). Using external data mining tools for analysis and computation, the data access as well as pre-processing and transformation causes several data management issues [7]. The benefits of in-database data analysis and dataset preparation (specifically in relational database management systems (RDBMS)) for analysis purposes are discussed in some works such as [8, 9, 7]. However, to the best of our knowledge, there is no current effort focusing on the optimization of patient similarity analysis with an in-database solution. In this work, data pre-processing and patient similarity analysis are processed within a DBMS that holds the EMR. Thus, the cost of external tools is saved and data transfer latency and data management problems are eliminated.

We implemented patient similarity analysis based on the common cosine similarity metric following the approach of [4, 10, 11]; cosine similarity exploits similarities between the patients along multiple dimensions. Furthermore, we introduce a method for optimizing patient similarity calculation in-database. In order to address the real-world medical analysis challenges, two real and relatively big EMR datasets are used for testing. First of all, the commonly used ICU dataset called MIMIC-III [12] is used. As a second dataset we used the diabetes dataset from the Health Facts database [13, 14]. The two sets contain both demographic data (like gender) as well as lab results.

1.1. Current State and Contribution

Researchers have done a lot of advanced work in the field of patient similarity analysis. However, we tried a new in-database approach for implementation and analyzed options for performance optimization:

- **Different Evaluation Base:** Accuracy is the main focus of current work applying patient similarity analysis. This comes with an increase in computational load as in [4, 5]. However, in the age of big data where EMRs get massively larger, the performance of such methods is critical, too. For a real-life medical application we should attain a fairly short amount of time for analyzing big data. Thus, in contrast to other work this paper focuses on performance evaluation.
- **A new Approach to Implement Patient Similarity Analysis:** Furthermore, related approaches work under synthetic conditions where the data are extracted from the database and the analysis is processed in external tools such as R [4], Python [15] and MATLAB [5]. However, several data management issues arise from using external data mining tool for analysis, data access, as well as pre-processing and transformation [7]. Our study differs from these approaches by implementing patient similarity inside a database management system (DBMS). In-database data analysis and pre-processing has been discussed for example by [8, 9, 7]. A NoSQL database is used in patient similarity analysis by [16]. However, to the best of our knowledge until this moment in-database (specifically in-RDBMS) has not been utilized in patient similarity analysis.

- **Evaluating Column-oriented and Row-oriented Databases:** Column-oriented DBMSs have shown higher performance on analytical workloads [17]. We hypothesized that column-oriented database management systems will outperform the row-oriented ones also for patient similarity analysis. To test this assumption, we calculate our PSM by a SQL expression inside the database system. We conducted the same analyses in one DBMS from each category.
- **Optimizing Patient Similarity Calculation:** Analyzing patient similarity produces data explosion and computational burden when exploiting big data. Finding a solution for optimizing patient similarity analysis is a key contribution of this paper. We propose different methods to optimize calculation of patient similarity. Our developed method was tested against two widely used EMR datasets. With support from our developed optimization methods a low execution time of patient similarity analysis was accomplished.
- **Direct Application on EMR database:** In our implementation, data pre-processing and patient similarity analysis are processed within the database that holds the EMR. SQL was mainly used for the implementation. When the majority of the workload is processed within a database, the cost of external tools is saved. Data transfer latency and data management problems are eliminated. Thus, our most significant result was that applying our implementation directly within an already existing EMR relational database is attainable. The medical staff will benefit from easier analysis applications within a real world medical system.

Consequently, we achieve a balance between analysis performance and applicability; our resulting analysis method can be directly implemented under real-world conditions with a high performance.

1.2. Paper Organization

The rest of this paper is organized as follows: Section 2 gives an overview about some of the related literature on patient similarity. Section 3 presents the used datasets, the pre-processing and the selected predictor variables for this study. All the normalization and formatting steps that have been done on the datasets are described in Section 4. Section 5 describes the patient similarity algorithm and its implementation in SQL for the numerical variables and for the categorical variables. Section 6 presents the applied technology and approaches for computing and optimizing the patient similarity analysis. Section 7 shows the evaluation result of the optimization method to improve performance and compares the performance of two DBMSs. Moreover, a detailed discussion of alternative approaches to implement patient similarity analysis is also provided. The results are followed by the final Sections 8 and 9 that conclude our work and give some prospects of future work.

2. Related Work

Much research has been done on applying patient similarity analysis on medical data for different predictive approaches such as discharge diagnosis prediction by [18] and future health prediction by [19]. A common predictive model for Intensive Care Unit (ICU) patients is mortality prediction as in [20, 4, 15]. Their implemented ICU PSM is one of the algorithms listed in [2]. Morid et al. [20] applied a similarity-based classification approach with a k -nearest neighbor algorithm for ICU patient similarity analysis. Similarly, [15] used Euclidean distance with a k -nearest neighbor algorithm. The simple and commonly used approach of cosine similarity metric is used by [4] to

test their hypotheses on personalized mortality prediction. Moreover, patient similarity is applied on other medical data for instance [21, 22, 5]. Table 1 summarizes the purpose of using patient similarity, the used patient similarity metric and the evaluation metric along with the implementation tools that are used by the surveyed approaches.

In all of the above mentioned research papers the patient similarity analysis is used as a basis for developing a prediction model. The focal point of the evaluation was the accuracy of the prediction. Either this research ignores a performance discussion or mentions the limitation of the computational burden of their analysis algorithms. Their accuracy improvement comes at a higher cost, namely an increase in computational complexity and load. Certainly, accuracy is a crucial factor when the predictive basis relies on the PSM. However, in the age of big data where the EMRs get massively larger, the performance of such methods is critical, too. High-dimensional data (i.e. data with a huge amount of predictor variables like medical measurements) and a large dataset would naturally lead to an increased computational burden. This can become a major issue particularly for training prediction methods; these training sets rely on the calculation of all pairwise PSM values between all patients in the training data. The pairwise PSM calculation exacerbates the challenge of handling big EMR data to a data explosion dilemma.

In the surveyed related works, different tools are used for data extraction, pre-processing and analysis. All the analysis calculations were done with an analytic tool or a developed framework as shown in Table 1, while the data was obtained and pre-processed from a database. In contrast to this related work, the majority of our calculations was executed on a DBMS. Implementation of our approach directly inside a column-oriented database can be achieved with low execution time. Hence, compared to related approaches the cost and the effort of external data analysis tools are saved. Moreover, data management problems of exporting the dataset from the DBMS are also avoided.

The surveyed articles however do not provide time measurements of how long PSM calculation took. Thus, we can not really compare our final result to them. However, experiencing high performance and optimization by our method we can say that these works and other works, which use patient similarity as a basis for different predictive models, can benefit from our approach. There are however a few articles on optimizing patient similarity analysis based on external data analysis tools. For instance Barkhordari et al. [6] introduce MapReduce for optimizing patient similarity analysis. Our method is not relying on the advances that data analysis technologies provide. In contrast, it enables the integration of patient similarity metrics within the EMR database to execute efficient analyses.

A general in-database approach for data analysis is presented by Ordonez [9]. Ordonez [23] found that high-dimensional data in vertical layout (i.e. column layout) is not a good choice for SQL. He considered a high-dimensional dataset with 50 dimensions. Moreover, he also discusses the benefits of pre-processing the dataset within a DBMS in [7]. More specifically the use of a columnar DBMS for analyzing big data is investigated by Sridhar [8].

3. Data Sets

We apply patient similarity analysis on two real datasets according to multiple dimensions (e.g. medical measurements and the diagnosis code ICD-9). The two datasets were pre-processed and transformed in order to construct a suitable dataset for a reliable analysis. In contrast to prior work, the pre-processing was carried out within a DBMS. The datasets and the pre-processing are described in the following two subsections.

Study	Patient Similarity for	Patient Similarity Metric	Evaluation Metrics	Implementation Tool
Sun et al. [19]	Prognosis prediction of occurrence of Acute Hypotensive Episode (AHE)	Locally Supervised Metric Learning (LSML), Mahalanobis distance	Accuracy of: Classification, Retrieval, and Prognosis	Developed system
Wang et al.[21]	Disease prognosis	Euclidean distance	Classification performance: precision, Recall, Accuracy, and F-measure	Developed SimProX system
Morid et al. [20]	Mortality prediction	Classification with K-Nearest Neighbor algorithm	Classification performance: precision, Recall, and F-measure	Developed PPMF Framework
Lee et al. [4]	Mortality prediction	Cosine Similarity metric	Accuracy: Area under ROC curve; area under precision-recall curve	R
Hoogendoorn et al. [15]	Mortality prediction	K-Nearest Neighbor, Euclidean distance	AUC for: -Influence of the number of patients upon the predictions accuracy and the computation time. -Earliest Prediction Time	Python
Panahiazar et al.[22]	Medication plan	Supervised and unsupervised clustering approaches with Mahalanobis distance	Sensitivity, specificity, F-measure, and accuracy	Developed framework
Gottlieb et al. [5]	Predict the eventual discharge diagnoses	Jaccard, Euclidean distance	AUC and F-measure	MATLAB

Table 1: Literature Review

3.1. MIMIC Dataset

As [24] state: “The intensive care unit (ICU) presents an especially compelling case for clinical data analysis”. The breadth and scale of the ICU data that are collected on a daily and hourly basis support significant ICU data analysis. Developing a decision support system based on an ICU database produces valuable outcomes. Outcomes are not limited to prediction of the length of hospital stay and mortality prediction, but can be extended to various results. Several advantages from analyzing ICU data are discussed by [25] and [24].

The real-world critical care database MIMIC (Medical Information Mart for Intensive Care) [12] is used in this study. The data set was collected from June 2001 to October 2012 from patients admitted to critical care units at the Beth Israel Deaconess Medical Center in Boston, Massachusetts. MIMIC is one of the largest and commonly used ICU databases. It is publicly available with patient privacy protected by removing all the Protected Health Information and with patient identity protected by de-identification. Nevertheless, the usage of the data must be with care and respect. After a formal request via the MIMIC website [26] and a compilation of ethics course the data access can be gained. We extracted patient data from the latest version of MIMIC. MIMIC-III v1.4 was released by September 2016. MIMIC-III is an updated extension of the widely used MIMIC-II. The unique MIMIC database is used to serve many advanced studies on medical data analysis such as [27, 28, 20]. Comparing to the previous versions of MIMIC, MIMIC-III is the most comprehensive dataset. It contains a larger number of patients with less inconsistency. MIMIC-III comprises over 58,000 hospital admissions, and over 61,000 ICU admissions for adults and neonates with thousands of medical data.

MIMIC-III provides a wide range of medical data from different ICU clinical care units [12]. Data are acquired from ICU monitoring such as vital signs, lab test results, billing data as ICD-9 code and notes and reports such as from Radiology report for an X-ray or Cardiology for ECG. Demographic data from hospital admission are also provided such as gender. Additionally, dates of out-of-hospital or in-hospital mortality are included in the dataset. In our study, all adult patients’ data (aged 15 years or above) from MIMIC-III are used. However, all the admitted patients’ data with any null values of predictor variables were excluded. A total of 46,596 ICU admissions met these criteria and are used in this study. All the ICU admissions from the ICUSTAYS table with the care units Coronary Care Unit (CCU), Surgery Recovery Unit (CSRU), Medical ICU (MICU) and Surgical ICU (SICU) were extracted. We exclude the admissions from the neonates care units. Each record identifier (ICUSTAY_ID) represents a separate patient in our study; no identity check was done to differentiate ICU admissions from the same patient – a process which is analogous to [4].

3.1.1. Predictor Variables

We had to combine the predictor variables from several tables of the MIMIC-III dataset. The selection of the predictor variables is inspired by Lee et al [4] who extracted 76 variables for ICU mortality prediction from the MIMIC-II data set. Numerical and categorical values are gathered. Predictive variables include minimum and maximum values of some vital signs from each 6-hour period of the first 24 hours in the ICU stay such as heart rate and mean blood pressure. Some minimum and maximum lab variables from the first 24 hours in the ICU stay are also included for instance white blood cell count and serum glucose. Categorical values such as the ICU service type and ICD-9 code and some binary values such as the use of mechanical ventilation are also extracted.

For testing the scalability of our optimized method we increased the amount of the predictor variables. For instance, in addition to the minimum and maximum values of lab variables we also extracted each 6-hour period of the first 24 hours in the ICU to take into account that each minimum, maximum and each 6-hour period measure produces a separate predictor. In total the extracted information was composed of 135 predictor variables (that is, features) for each patient. For each patient, we compiled the features into a vector of length $m = 135$: the patient vectors that are used to calculate the cosine similarity between two patients. The final data set consists of $n = 46,596$ patients in total. Thus, we obtained 46,596 m -dimensional feature vectors where $m = 135$.

3.1.2. Data Pre-processing and Cleaning

The complete MIMIC-III dataset containing 26 tables was installed and was queried by a Structured Query Language (SQL) statement to extract predictor variables. MIMIC-III extends MIMIC-II which was collected between 2001 and 2008 with newly collected data between 2008 and 2012. The result of this extension is that MIMIC-III database is extracted from two different DBMSs: the MetaVision system and the CareVue system [12]. These systems use a different ITEMID for each medical measurement which complicates the data extraction. Therefore, we consider using multiple ITEMID values for each predictor variable to ensure complete data extraction.

After completely extracting the predictor variables for all of the patients, the data were pre-processed and transformed. Missing data is a common problem in EMR due to different reasons such as the nature of the medical information, the heterogeneity of the data and the systems, or simply human error [25]. There were some missing data in the selected predictor variables from MIMIC-III. There are various approaches for dealing with the problem of missing data. Han et al. [29] describe several approaches; moreover a chapter is devoted to them in [25] by Salgado et al. Some of these methods are (i) discarding the tuple, (ii) substituting the missing value by the mean of that variable or (iii) estimating the missing value by a model.

We use the most simple and – from our point of view – most reliable method which is the deletion method (i): The incomplete data records were excluded from the dataset. This method was also used by Lee et al. [4]. This method avoids biased data that could be caused by incorrectly filling in the missing values. Moreover, the final dataset after extracting the missing value records is large enough to serve our goal for testing performance optimization.

In this study, we considered similarity between the adult patients; that is why only the patients with age older than 15 years old are included. For extracting the predictor variable of appropriate age, the difference between the patient’s date of birth and the patient’s date of first admission had to be calculated. While extracting the age feature, we noticed some values with 300 – a figure that looked like a data error. However, we found that the age of the patients that are older than 89 years are obscured to comply with the HIPAA (Health Insurance Portability and Accountability Act) [12, 3]. That is why data shifting was applied to the MIMIC dataset. When the patient age was determined to be older than 89, the date of birth was set to exactly 300 years before the date of the first admission. Therefore, all patient ages that larger than 90 are replaced by 300 in the dataset to mask the actual age. However, we need to maintain a reasonable age range for the normalization purpose and for accurate analysis. Subsequently, when extracting the age of these patients we had to replace their ages by the median value which is 91.4.

The final dataset contains some categorical variables such as the admission type and the ICU service type. It was meaningless to turn them into a numerical representation (a process called factorization) since there is no reasonable scale for their values. Thus, they were extracted as

strings. Then, equality checking was performed. Similarity on categorical values hence corresponds to string equality: Identical categorical values lead to a similarity of 1; otherwise non-identical strings lead to a similarity of -1 . The calculation of the similarity of categorical variables is added to the PSM as will be described in Section 5.

3.2. Dataset from The Health Facts database

The second dataset we use was originally extracted from the national Health Facts database (Cerner Corporation, Kansas City, MO). The database contains 10 years (1999-2008) data from EMRs of 130 hospitals throughout the United States. Different data are collected for emergency, outpatient, and inpatient visits. For instance demographic data, ICD code for diagnoses and the procedures, pharmacy data and in-hospital mortality. It consists of 74,036,643 unique visits that correspond to 17,880,231 unique patients with 117 features. The data became available for the researchers after complying with HIPAA by deidentifying all the data.

This dataset is provided by Center for Clinical and Translational Research, Virginia Commonwealth University [13]. It is available online by the UCI Machine Learning Repository [14]. It is called diabetes dataset since it was extracted to be used for studying the relationship between the measurement of Hemoglobin A1c (HbA1c) and early hospital readmission. The dataset was extracted from the original database for inpatients with any kind of diabetes as a diagnosis. Some criteria were considered for the length of stay, laboratory tests and the medications. 101,766 patients fulfill these inclusion criteria.

3.2.1. Predictor Variables

The predictor variables selection by [13] was based only on the features that have associations with the diabetic condition. Some demographic data were extracted for instance race, gender and age. Furthermore, medical data such as diagnoses, A1c test result and existence of prescription to diabetic medication. Moreover, they define a readmission attribute. Take into account that the original database contains features with high percentage of missing values. The obtained dataset consists of 50 features out of 117 features.

3.2.2. Data Pre-processing and Cleaning

The value of the age feature was in 10-years intervals. We assign the median value of the age to the age value. There were some categorical variables in the dataset. In contrast to the MIMIC dataset, there was a reasonable factorization scale for these variables. Thus, categorical variables were transformed to numerical variables. Some variables were represented as binary value such as change of medication feature; value of -1 represents No and value of 1 represents Yes. Scale of three values was assigned to the dosage feature: -1 when there is no prescribed drug, 0 when the dosage does not change (i.e. Steady), and 1 represents increased or decreased dosage. The calculation of the similarity of these categorical variables follows the similarity calculation of numerical ones.

The patient vectors with null or unknown values are deleted. The patients with NULL as their primary diagnosis ICD code are excluded. Moreover, the patients with unknown gender are deleted. The final dataset consists of 100.097 patients out of 101.766.

Features with high percentage of missing values are excluded. We removed these features: race, secondary and tertiary ICD codes, weight, payer code and medical specialty. Moreover, the encounter ID and patient number are removed. Thus, out of 50 features the final dataset has 42 features.

4. Data Normalization and Formatting

4.1. Data Normalization

An attribute with a small unit has a larger range of values and hence produces a greater effect on the result. In effect, “[t]he measurement unit used can affect the data analysis” [29]. To ignore the choice of the attribute measurement unit and to give all the attributes equal weight, we applied a normalization step on the data. Normalization transforms the data into a scale of smaller range. The continuous numerical predictors such as the vital signs and lab test results were normalized into the range $[-1, 1]$ because the result value of our used PSM is also in the range -1 (denoting the minimum similarity) and 1 (denoting the maximum similarity). Normalization is also commonly applied in other approaches [4] to ensure equal contribution of all predictor variables to the PSM calculation. We use the min-max normalization method [29] as shown in Equation (1).

$$v_i'' = \frac{v_i - \min_x}{\max_x - \min_x} \cdot (\max_{\text{new}} - \min_{\text{new}}) + \min_{\text{new}} \quad (1)$$

In this equation, x is a numeric predictor variable with m observed feature values v_1, v_2, \dots, v_m . \max_x and \min_x are the minimum and maximum values of the predictor variable x . Hence, the normalization method in Equation (1) maps a value v_i of x to v_i'' within the new range; we want to scale to the new range \max_{new} and \min_{new} , which are 1 and -1 respectively. However, the relation among the original data values of x are preserved. Replacing, \max_{new} and \min_{new} with their values 1 and -1 gives the following Equation (2).

$$v_i'' = 2 \frac{v_i - \min_x}{\max_x - \min_x} - 1. \quad (2)$$

Note that this kind of normalization only applies to numerical values.

4.2. Data Formatting

Prediction models that are based on patient similarity require many variables (in other words, features or dimensions) for more accurate prediction. We have a high-dimensional prediction model which consists of 135 features for MIMIC dataset and 42 features for diabetes dataset. More precisely, for each patient, the sequence of the patient’s predictor variables constitutes a *patient vector* which is used to calculate the cosine similarity. There are basically two different ways to represent the patient vectors: formatting the data in either a *row-oriented* or a *column-oriented* way is a major decision for the implementation of the PSM inside the database system. Ordonez [23] called these two ways horizontal and vertical layouts of a dataset.

The data extracted from MIMIC is considered as column layout: All the patients predictor variables (features) are stored in one column called FeatureValue; the features are differentiated by assigning a FeatureID to each of them. This has the effect that now there are multiple rows for each patient vector identified by the patient VectorID. Consequently, the data of n patients each with m features are stored as the column-oriented schema shown in Table 2.

Still, to have a fair comparison and to find the optimal data format for our patient similarity calculation, we compared the column data layout to a row data layouts. Thus, the data was reprocessed into row layout (see Table 3). This will lead to a multicolumnar table format with m columns where m is the amount of features: that is, one row per patient and one column per feature.

<i>VectorID</i>	<i>FeatureID</i>	<i>FeatureValue</i>
1	ID_1	$value_1$
\vdots	\vdots	\vdots
1	ID_m	$value_m$
\vdots	\vdots	\vdots
n	ID_1	$vaule_1$
\vdots	\vdots	\vdots
n	ID_m	$value_m$

Table 2: Column Data Layout

<i>VectorID</i>	<i>Feature₁</i>	<i>Feature₂</i>	\cdots	<i>Feature_m</i>
1	$value_1$	$value_2$	\cdots	$value_m$
\vdots	\vdots	\vdots	\vdots	\vdots
n	$vaule_1$	$vaule_2$	\cdots	$value_m$

Table 3: Row Data Layout

Ordenez [7] claims that creating such a row data layout with many columns makes data analysis harder and query processing slower. We will find out if that is the case for our calculation.

5. Method

There are different algorithms for patient similarity analysis as described in [2] like, for instance, Mahalanobis distance [22] and Euclidean distance [30]. We use the cosine similarity metric for our PSM [4, 10, 11] as shown in the following Equation 3.

$$\cos(x, y) = \frac{x \cdot y}{\|x\|_2 \cdot \|y\|_2}. \quad (3)$$

x and y are the vectors of two different patients $x \neq y$, \cdot is the dot product and $\| \ \|$ is the Euclidean vector magnitude. Translating the dot product and the magnitude into calculations on the vector elements x_i and y_i results in the following Equation 4.

$$\cos(x, y) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}. \quad (4)$$

Figure 1 shows the concept behind cosine similarity for patient similarity analysis. The patient data of n patients with m features are represented in an m -dimensional vector space: The patient vectors $N = P_1, P_2, \dots, P_n$ each consist of features x_1, x_2, \dots, x_m . Thus, m is the size of the feature set, while n is the size of the patient set. Cosine similarity between two patient vectors in the vector space is the cosine of the angle between them. It gives a result value between -1 and 1 .

In a productive medical decision support system, the index patient is the patient for which a treatment decision or a prediction is sought. Hence, patient similarity analysis can be applied

```

SELECT  v1.vector_id, v2.vector_id,
        --The dot product of two vectors
        SUM(v1.value*v2.value)
          /* Divided by the product of
           * the respective norms*/
          /(SQRT(SUM(v1.value*v1.value))
           *SQRT(SUM(v2.value*v2.value)))
        AS CosineSim
FROM    vectors v1 JOIN vectors v2
        ON v1.feature_id = v2.feature_id AND v1.vector_id < v2.vector_id
GROUP BY v1.vector_id,v2.vector_id;

```

Figure 2: SQL cosine similarity for numerical variables in column data layout

to the index patient’s vector to identify similar vectors of other patients. To find all the patients similar to a specific index patient P_x all pairwise cosine similarities between this index patient P_x and every patient in the patient set $N - \{P_x\}$ are calculated. Two patients are very similar if the result is 1 and are very different if the result yields -1 [4].

As a basis of such a medical decision support system, a training set can be obtained where all pairwise patient similarities between patients P_i and P_j (where $i, j \in \{1, \dots, n\}$) have to be calculated. In effect, the cosine similarity is symmetric leading to the same value for the pair (P_i, P_j) and the pair (P_j, P_i) . Hence, in the following we enforce the additional restriction that $i < j$ to avoid superfluous calculations. This is the use case we will focus on in our subsequent analysis.

As mentioned previously, our used predictor variables contain mostly numerical but also some categorical variables. Therefore, they are treated differently during cosine similarity calculation. Moreover, we consider row and column data layouts, the SQL code for our PSM based on such layouts. The following subsections will describe the methods. Additionally, the SQL code will be also provided.

5.1. SQL Cosine Similarity for Column Data Layout

After normalizing the numerical variables into the range $[-1, 1]$, the calculation of the cosine similarity metric as in Equation 4 was processed. For direct execution within the selected database system a SQL statement was used as shown in Figure 2.

Euclidean vector magnitude calculation is done through grouping by *VectorID* then calculating $\sum_{i=1}^m value_i^2$. Calculating the dot product between two patient vectors is by joining the vector table with itself on the *FeatureID*. This way we have both values side by side and can then again group by *VectorID* of vector table 1 and a second sub-grouping by *VectorID* of table 2. Since cosine similarity is a symmetric function, the second part of the join-condition makes sure that the calculation is only performed once: the similarity calculation is performed only with all the higher vector IDs. However, these two joins on *VectorID* and on *FeatureID* for each row of the patient vector cause significant overhead.

Since our patient data in MIMIC contain four categorical variables, we have to devise a method to calculate cosine similarity for this kind of variables. Naturally, the operation of squaring and multiplication cannot be applied here. Therefore, we replace them with an equality check, i.e.

```

(SELECT  c1.vector_id, c2.vector_id, (2*(count(*)-2)) As DotProd
FROM categorical_vectors c1
    JOIN categorical_vectors c2
      ON c1.feature_id = c2.feature_id
     AND c1.vector_id < c2.vector_id
    WHERE c1.value = c2.value
    GROUP BY c1.vector_id,c2.vector_id
UNION ALL
(SELECT  c1.vector_id, c2.vector_id, -4 As DotProd
FROM categorical_vectors c1
    JOIN categorical_vectors c2
      ON c1.feature_id = c2.feature_id
     AND c1.vector_id < c2.vector_id
    WHERE c1.value <> c2.value
    GROUP BY c1.vector_id,c2.vector_id HAVING count(*) = 4)

```

Figure 3: SQL cosine similarity for categorical variables in column data layout

if two variables are of the same category, the result will be 1 and -1 if not. For the Euclidean vector magnitude this means that the result of the categorical part will always amount to the total number of values (e.g. $+4$, if there are 4 categorical variables in a vector). As a result, we can just add this value to all norms. Figure 3 shows that (in-)equality checking is being achieved in the where clause of the SQL statement that follows the same pattern as its numerical counterpart.

5.2. SQL Cosine Similarity for Row Data Layout

The previous approach of SQL PSM for column data layout shows that the SQL statement of cosine similarity PSM can be easily implemented by utilizing the SUM() aggregation function. SUM() function is used for the dot product and the norms along all the features of the compared patients. However, the SUM() function of SQL is valid only for column-wise calculation. Hence, it was applicable in the column data layout where all the patients predictor variables are stored in one column. However, in the row data layout each patient vector is represented by a row of m columns where m is the amount of features. Thus, this approach can not take advantage of the SUM() aggregate function. For calculating the dot product each feature of the patient vector is specified. Yet, the row-based approach reduces the join overhead of the column-based approach since it requires only one join on *VectorID* for each patient vector. The SQL code for cosine similarity for the row data layout of numerical and categorical variables is shown in Figure 4.

6. Calculation

Most of the current implemented patient similarity analytics are applied on tools such as R [4], Python [15] and MATLAB [5]. In this study we chose the approach of doing all calculations within the database system in which the datasets already reside. MonetDB utilized multi-threading. Hence, to make a fair comparison we simulate multi-threading for PostgreSQL. Connection pooling is used to enhance the multi-threading performance.

```

SELECT v1.vector_id, v2.vector_id
      (v1.feature_1 * v2.feature_1 +
       ...
       v1.feature_m * v2.feature_m) +
      (CASE WHEN v1.categorical_feature_1 = v2.categorical_feature_1
            THEN 1 ELSE -1 END) +
       ...
      (CASE WHEN v1.categorical_feature_m = v2.categorical_feature_m
            THEN 1 ELSE -1 END)
    / (v1.norm * v2.norm)
FROM   vectors v1 JOIN vectors v2
      ON v1.vector_id < v2.vector_id;

```

Figure 4: SQL cosine similarity for row data layout

6.1. Column-oriented vs. Row-oriented databases

We will compare the performance of two DBMSs from two categories (row stores and column stores). In a column store DBMS all the predictor variables (i.e. all features of the patient) are stored underneath each other in one column – similar to the column-oriented layout in Table 2. This is exactly the way how a column store saves data sequentially into the disk blocks column by column. Hence, the values of the FeatureValue column are stored sequentially on disk. Thus, when a set of patient feature values (i.e. a patient vector) is required for PSM, it will be easily retrieved by the query. Moreover, extracting a specific attribute or a feature value for calculating the dot product is easier since not the whole record needs to be retrieved. This approach reduces the disk I/O because the amount of the loaded data from the disk is reduced [17]. Performance of column-oriented DBMSs on analytical workloads have previously been analyzed in non-medical settings (see [17] and [31]).

In contrast, the alternative would be a row-oriented database where the data of a patient vector (i.e. a record) are stored in a row-oriented layout as shown in Table 3. This is the way how data are stored into disk blocks row by row. The values of all the feature columns are stored sequentially on disk. Thus, when a dot product is to be calculated for two patients on a specific feature value (i.e. attribute) the whole row has to be retrieved and then searched for this specific feature. This slows down the retrieval time of the query.

In general, it has been observed that for columnar storage “reading the same number of column field values for the same number of records requires a third of the I/O operations compared to row-wise storage” [31]. That results in a fairly short query time. Therefore, a database from column-oriented DBMS world was selected for our tests. The relational column store DBMS *MonetDB* [32] is used for computing the PSM. The SQL statements listed previously are used for implementing the similarity calculation. PL/SQL (Procedural Language/Structured Query Language) is not provided by MonetDB, so Java Database Connectivity (JDBC) is used to conduct the step by step calculation.

Our hypothesis is that column-oriented DBMSs will outperform the row-oriented ones for the patient similarity use case. Our assumption is that MonetDB will perform faster than traditional

row-oriented DBMSs. We test this assumption by comparing MonetDB to the row-oriented PostgreSQL. If MonetDB is capable to outperform the row-oriented DBMS, then that would indicate that column-oriented DBMSs are beneficial for our application.

6.2. *Batching for Optimizing Patient Similarity Analysis*

The final patient table for MIMIC consisted of a total of 46596 patient vectors composed of 135 features. Keeping this in mind we can estimate how big the intermediate join table generated by the cosine similarity SQL statement for a our data set will turn out. We get the result by multiplying the number of patient combinations with the number of features (= number of rows or columns based on the used data layout) and multiply this number by 4, treating every cell of our table as 4 bytes. To make it human-readable we can divide by 2^{30} to get the measure in GiB. The estimated size of the join table for a our data set is:

$$\frac{\binom{46596}{2} * 135 * 4}{2^{30}} \approx 545,95 \text{ GiB}$$

The estimated size of the second dataset is 783,82 GiB. This shows the unfeasibility of calculating PSM in one big query. It would cause the cache and also hard drive of the computer to overflow. Therefore, the calculation has to be split up into multiple steps. This is the reason of batching approach.

Sequential calculation of the cosine similarity for each single patient vector with all other patient vectors having a higher id (to avoid duplicates) causes major overhead. This requires a temporary join table on which the cosine calculation is performed and which is afterwards cleared. Furthermore, joining took up over 90% of the time while the actual cosine calculation was almost done instantaneously. The whole process can be seen as splitting the pairwise similarity calculations into 46596 summands:

$$\sum_{i=1}^{46596} 46596 - i = \binom{46596}{2}$$

While the first patient is compared to 46595 other patients, the next to last patient is compared to just one other patient and the last patient to no other patient.

The long run time of the previous method leaves a desire for acceleration. The overall execution time per patients by can be reduced by improving the join. Instead of only joining for one patient with all other patients at a time, we applied a batching and processed 50 patients simultaneously (batch size of 50 proved to be the most efficient). The improvement is as follows: joining only for one patient amounts to $135 \cdot 46595 = 6,290,325$ rows in the join table compared to 314,344,125 for 50 patients.

$$\sum_{i=1}^{50} 135 \cdot (46595 - i) = 314344125$$

This batch size of 50 patients per join step marks the sweet spot for the batching approach.

Further performance improvement can be achieved by eliminating the intermediate steps. Previously, the temporary join table was written before executing the PSM on it. These intermediate steps have the biggest impact on the performance. Therefore, we combine joining and calculation into one query whose result is then directly written into the result table. This approach results in

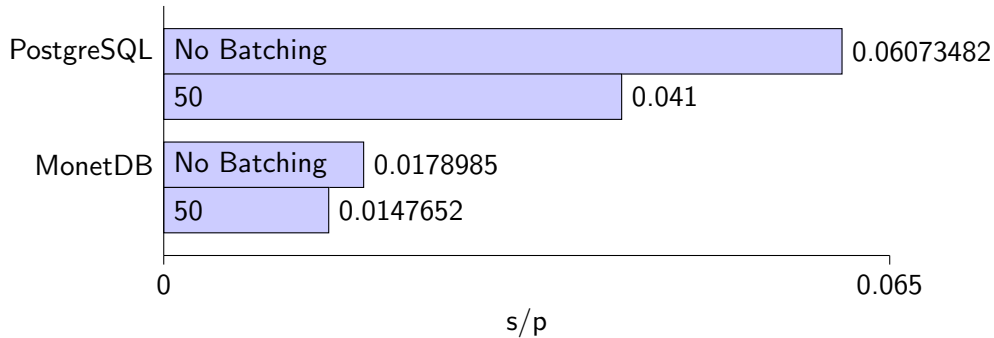


Figure 5: Runtime without batching and with batch size 50 on MIMIC dataset with row data layout

the best analysis execution time. Therefore, this final setting is used for implementing our optimization method batching. The final implementation of cosine similarity metric for numerical and categorical variables with this method of batching was performed by SQL and JDBC.

7. Results and Discussion

With the previously presented approaches we reached our goal of a direct implementation of patient similarity analysis in a medical DBMS. Proceeding through the procedures described in Section 6.2, performance optimization for patient similarity analysis of big data was accomplished. The result will be discussed in the following Section 7.1. Batching provided a decisive runtime benefit: simultaneous calculation of patient similarities gives an optimal performance for analyzing the data.

Accomplishing a fairly good improvement from the initial batching method brings the interest for further testing our assumption. We hypothesized that column-oriented DBMSs will outperform the row-oriented ones on analyzing big medical data. To test this assumption, we conducted the same analysis of those approaches on MonetDB and on one of the most widely used row-oriented DBMSs PostgreSQL. The result will be presented in the following Section 7.2.

Latency elimination from accessing and analyzing the data directly on DBMS, the power of MonetDB columnar storage with added functionality support of JDBC, and the performance optimization from our developed methods led to our efficient implementation. The final result pays off the efforts. Hence, we come closer to a patient similarity application that can be used as a basis for decision support in a real-world medical DBMS containing EMRs.

All of our computations were conducted on a PC running *Ubuntu 16.04 LTS* on an *Intel i7-7700k* with 64 GBs of RAM. PostgreSQL version 10.1 and MonetDB version 11.27.11 were used. We obtained the runtime by computing all pairwise similarities for the entire data set and then averaging over the amount of patients per data set. This runtime is hence measured in seconds per patient denoted s/p .

7.1. Result of Batching for Optimizing Analysis Performance

We ran the analysis with our batching approach from Section 6.2 and without batching on MonetDB (as a column store) and PostgreSQL (as a row store). The results are shown in Figure 5; the number in the bar is the batch size. Our dataset extracted from MIMIC-III is used for this test with row layout.

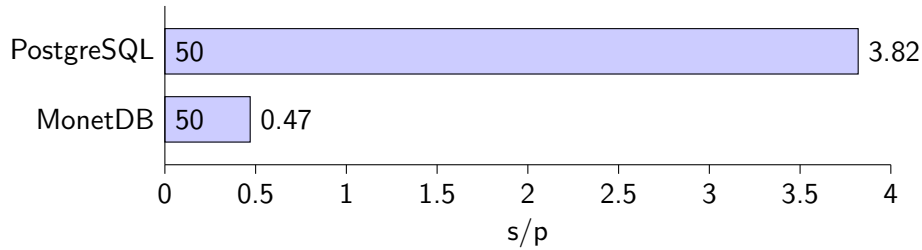


Figure 6: Runtime for executing the PSM per patient with batching for column data layout

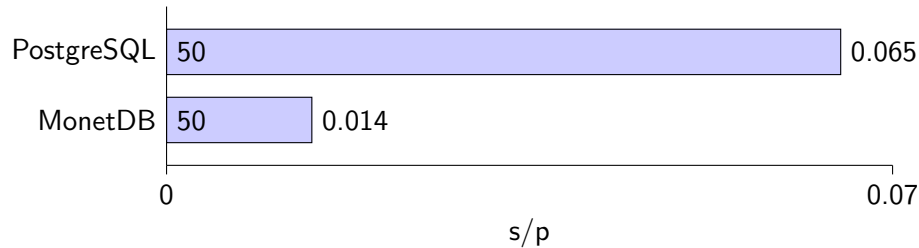


Figure 7: Runtime for executing the PSM per patient with batching for row data layout

“No batching” means the sequential similarity calculation of each patient with all other patients with higher id. Batch size 50 proved to give better performance. In both the row-oriented and column-oriented DBMSs batching lowers the required time for analysis. With batching the similarity calculation was faster by 32.49% in PostgreSQL and by 17.51% MonetDB. The batching approach hence shows decisive improvements when optimizing the performance of patient similarity analysis for big EMR. Moreover we observed the significantly better performance of the column-oriented database MonetDB over the row-oriented PostgreSQL.

7.2. Column-oriented vs. Row-oriented databases with SQL over Row and Column Data Layout

We tested our hypothesis regarding the advantages of the column store database. We ran the patient similarity analysis with batching on MonetDB (as a column store) and PostgreSQL (as a row store). Moreover, we compared similarity analysis over the two data layouts (the column-wise and the row-wise). Our dataset extracted from MIMIC-III is used for this test with both layouts. We compared the required time for calculating patient similarity analysis per patient (that is, all pairwise similarity computations for a single patient). The result will be discussed in the following.

Figure 6 shows the result of applying SQL cosine similarity over column data layout in both DBMSs. Again, MonetDB performed better than PostgreSQL – probably due to the optimized internal storage format of MonetDB.

Figure 7 presents the result of testing the two DBMSs on SQL cosine similarity over the row data layout. We proposed using the row data layout in order to empower the row store DBMS PostgreSQL. That was approved. Time required by PostgreSQL for executing the analysis on the row layout is less than the one for the column layout by 98.3%. Yet, MonetDB is still more efficient than PostgreSQL. Moreover, also MonetDB was faster in the row layout than the column layout by 97%. This test shows that the SQL execution over row layout of data gives a better performance optimization than over the column layout. Hence, based on our SQL code for the column layout we can conclude that the more dimensions (that is, features) are used, the more joining and aggregation

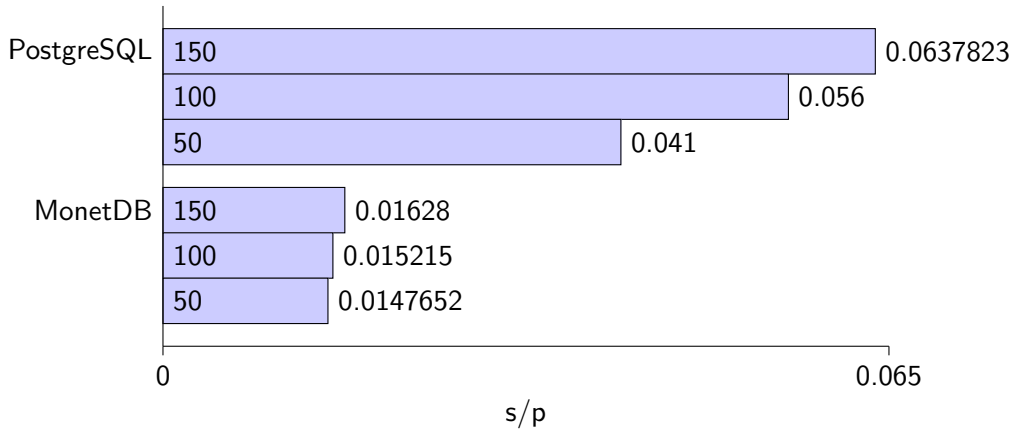


Figure 8: Different batch sizes on MIMIC dataset

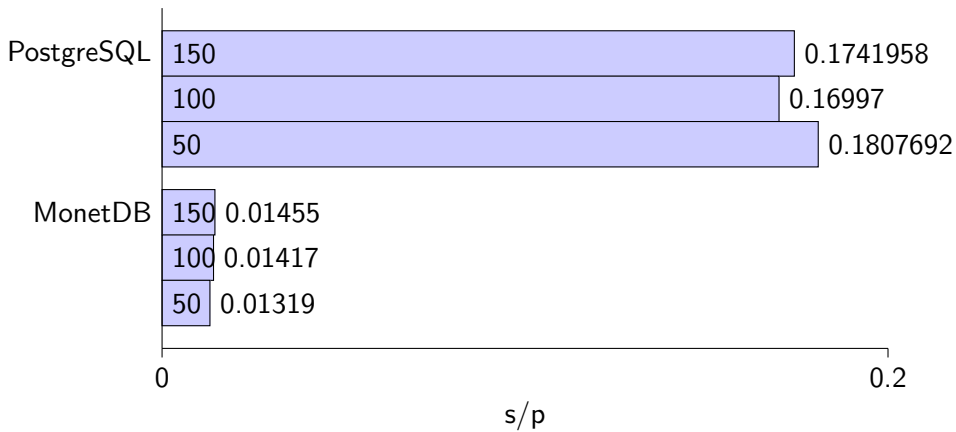


Figure 9: Different batch sizes on diabetes dataset

are required; this overhead was the main cause of the performance reduction. This test proves that the row data layout is the best candidate for SQL patient similarity on both DBMSs. Therefore, the next test will be applied only on the row data layout.

7.3. Column-oriented vs. Row-oriented Databases with Different Batch Sizes and Datasets

In the following we test the two DBMSs with different batch sizes using the two datasets from Section 3. However, we only test the row layout because it was much better than the column layout as already shown in Section 7.2. The second dataset was already in a row layout and no transformation was needed. Moreover, we used connection pooling to allow for multi-threading in Postgres; in contrast, multi-threading is automatically utilized in MonetDB without any manual configuration. The results are shown in Figure 8 and Figure 9. First of all, we noticed the improvement of PostgreSQL after the utilization of multi-threading. Comparing PostgreSQL’s performance from Figure 7 (without utilization of multi-threading) and Figure 8 on batch size 50, it improves by 36.92%.

Different batch sizes (from 10 to 150) were evaluated. In general, batch size 50 was our sweet spot. For instance, looking at Figure 8 for MonetDB from batch 50 to 100 and to 150 the per-

formance decreased by 3.046% to 10.259% and it is decreased further in PostgreSQL by 36.585% to 55.567%. Even though the batch size of 50 patients gives the highest performance for the dataset extracted from MIMIC-III, the results from this study should be considered preliminary. For example, the second dataset performed better for batch size 100 in PostgreSQL as shown in Figure 9. Therefore, it is premature to make a definitive conclusion about the general performance of batching for PSM. However, the results of this study revealed that partitioning of the data is crucial for PSM performance when analyzing big medical data. Probably, applying our approach with adjusted batch sizes on bigger datasets will show its effectiveness.

Moreover, Figure 9 shows that the row-oriented PostgreSQL performance is affected by the dataset size. It takes considerably longer for the larger diabetes dataset. However, MonetDB performance is still ahead of PostgreSQL and steady in spite of the larger dataset size. Joining overhead is more remarkable in PostgreSQL than in MonetDB. The reason might be linked to that multi-threading works automatically in MonetDB. Hence, compared to MonetDB performance, the multi-threading simulation in Postgres does not pay off.

In contrast to PostgreSQL, executing the similarity calculation for an index patient by MonetDB is even more efficient on the larger diabetes dataset as also shown in Figure 9. We noticed that for MonetDB the runtime per patient on the diabetes dataset is less than the one required in MIMIC dataset by 10.64%. Even though the diabetes dataset is larger than the MIMIC dataset, the MIMIC dataset has more dimensions (that is, number of features). This might cause a longer time required for processing the calculation per patient because each patient needs to be compared to the other patients on each feature. Thus, not only the size of dataset affects the performance but also the dimensional size. Moreover, the MIMIC dataset contains some categorical variables whereas the diabetes dataset contains none. Thus, some overhead is caused by the similarity checking between these categorical variables.

To sum up, the column-oriented DBMS MonetDB outperforms the row-oriented DBMS PostgreSQL (in all of the previous test cases over both the column and row data layouts, with different batch sizes and with different dataset sizes). The different factors (the size of the dataset and the number of dimensions, overhead of joining, aggregation and similarity checks of categorical variables) only have a slight effect on MonetDB in comparison to PostgreSQL. This superior performance of the column-oriented DBMS over the row-oriented one was not surprising. The column-oriented DBMSs are already proven to perform significantly better than row-oriented DBMSs specifically on analytical processing such as on decision support applications [17]. The column store I/O efficiency for read-only queries was good enough for our EMR analysis – in particular for reading the predictor variables – since it reads from the disk only the attributes accessed by a query.

7.4. Discussion

EMR data are usually extracted from different systems, then integrated and pre-processed in a dedicated tool and afterwards analysis is executed [33]. Since the EMR data come from different sources with different formats, the extracted data could be structured, semi-structured or unstructured [34]. Different analytic tools can be used. Wang et al. [35] state that depending on the data type and the analysis purpose the big data analysis in health care can be divided into three components: Hadoop Map/Reduce, stream computing, and in-database analytics. We categorized them into:

- **Data Analysis tools:** Data analytic platforms and tools such as Hadoop, Mahout or R are

one approach to implement patient similarity analysis. In order to assess their performance, as an alternative to our in-database approach, we used Apache Mahout to implement patient similarity analysis (that is, cosine similarity). First we had to pre-process the data to the data model of Mahout. The data model looks like the column format in Table 3. Thus, the column layout of our dataset extracted from MIMIC-III was used for this test. However, the strings of the featureID have to be replaced by numbers to follow the data model. The experiment was conducted on a PC running *Windows 10 Pro* on an *Intel i5-6300U* with 8 GBs of RAM. Mahout version 0.9 was used. The output was stored in a file on disk. The complete analysis process took less than one hour around 25 minutes. However, here we did not consider the time we lose for exporting and transforming the data out the DBMS and accessing the output in different tools. In our best contribution in MonetDB (with our batching approach) the complete analysis of the same dataset took around 11.46 minutes. Moreover, the bottleneck of exporting and pre-processing the dataset outside the DBMS was eliminated. All the workload was in-database.

The open source big data analytic tools provide advances in analysis and scalability (when run in distributed systems) however pose challenges for healthcare. They require high programming skills and technical support which is uncommon to obtain in the healthcare end-users [33].

- **Database Management Systems:** Another approach for implementing patient similarity analysis is to use a DBMS. Either relational DBMSs or NoSQL databases are a valid choice. However, the type of the extracted data decides the choice of the database system. The SQL database is well-suited for structured and relational data whereas the NoSQL database is perfect for non-relational and unstructured data:
 - **NoSQL Databases:** NoSQL databases (see [36] for a comprehensive survey) are used for analyzing unstructured data. For instance, Abdelrahman et al. [16] use the NoSQL database Neo4j for analyzing health care data. There are some current works applying patient similarity analysis on the unstructured health record data such as medical notes. Unstructured medical data usually have poor quality: “unstructured data is highly variable and all too often incorrect” [33]. Moreover, pre-processing the unstructured data for analyzing the similarity requires much effort. Extracting medical terms from unstructured data requires much effort and medical knowledge. Identifying and extracting the medical notes from examination reports is done manually by Chan et al. [37]. Wells et al. [38] mention different problems of analyzing unstructured medical data. Difficulties include grammatical errors, various interpretations of a specific phrase dependent on content, and the acronyms and abbreviations usage.
 - **Relational Databases:** In our case the data of the EMRs are structured (e.g. diagnoses, laboratory values, and medications). Hence, we find the SQL database is the best fit. We test the two categories of RDBMSs (column stores and row stores) on row and column data layouts. The column store DBMS was the fastest for both layouts which confirms other authors’ opinion: “Data mining researchers have also shown that SQL on parallel, columnar database could be a candidate for Big Data analytics” [8]. In-database approaches have many advantages. They eliminate the cost of pre-processing the data and analysis in different tools and avoid data management problems. In our use case, data pre-processing, analysis and storing the output result are all possible in-database.

Ordóñez [9] discusses many benefits of performing data analysis inside a DBMS but the main one is avoiding the data export bottleneck from the data warehouse. RDBMS has well-defined standards that assure full integrity and availability of data. On the other hand, for other use cases some weaknesses of an in-database approach might be that the data format and query language are limited to a specific range.

Developing a solution for analyzing structured data might not be valid to be applied to unstructured ones and vice versa. Nevertheless, Johnson et al. [39] state that it is nontrivial integrating data from different medical devices into a single data management system. The reason lies in the lack of standardization among the medical devices and the various data formats. Furthermore, as already mentioned the selection of different data analytics depends on the data type and the analysis purpose [35]. With all of these complexities, we believe that a generic solution for analyzing big medical data requires much effort to be achieved. Thus, our implemented method (in-RDBMS by SQL) cannot be applied to all of these heterogeneous databases and various data types. Nonetheless, the columnar DBMSs can process different data types. For instance, MonetDB can process RDF data by SPARQL, and SAP HANA database can perform text (i.e. unstructured data) analytic. However, Wells et al. [38] state that usage of structured medical data is an approach to reduce missing values in EMRs and it maximizes the impact of EMRs in research and in patient care. Furthermore, we argue that analyzing structured data is more accurate and does not require as much effort for pre-processing as the unstructured one. Our implemented method can empower the still dominant RDBMS to handle the data explosion that is produced from the $\binom{n}{2}$ pairwise patient similarity calculations.

8. Conclusion

Analyzing patient similarity is complex especially when handling big data. In this paper, we discussed and applied in-database patient similarity analysis. We implemented the commonly used cosine similarity PSM on a column-oriented and a row-oriented DBMS. For testing the performance of our optimization approach and of the DBMSs we used two real, relatively big and high-dimensional datasets. A comparison for all pairs of patient vectors is required to obtain a training set for further predictive models. This amounts in a total number of $\binom{n}{2} = \frac{1}{2}n(n-1)$ calculations which is $\in \mathcal{O}(n^2)$ in terms of complexity. Our study needs over five billion (5.009.654.656) and over one billion (1.085.523.715) comparisons for the two datasets. Data explosion and computational burden lower the analysis performance.

Performance optimization for patient similarity analysis of big data was accomplished. In particular, batching (simultaneous calculation of several patient similarities) gives an optimal execution time for analyzing the data. Executing SQL code for data extraction, pre-processing and for patient similarity analysis was fast thanks to the columnar storage of MonetDB. Even though MonetDB (a column-oriented DBMS) has clear speed advantage over PostgreSQL (a row-oriented DBMS), the column store DBMS is still not the silver bullet. There are various factors that influence database performance.

The use of DBMS for data medical analysis brings many advantages. Our most significant result is that we showed the capability of implementing patient similarity analysis inside a relational database system. Our introduced batching approach for speeding up PSM calculation (for big medical data analysis) and the implemented SQL code of cosine similarity PSM can be used for different predictive models where patient similarity is the basis. For instance, they can be used

for predicting the length of hospital stay or for predicting disease diagnoses by selecting different predictor variables from the MIMIC dataset or from a different dataset. Furthermore, a standalone patient similarity analysis can be used to gain valuable information for medical staff or researchers. Finding similar patients on certain medical measurements or diseases for producing statistical results or for finding groups of patients with rare disease are absolutely valuable benefits.

9. Future Work

The calculation methods of PSM that were employed in MonetDB showed that partitioning of the data is crucial for the performance. Therefore, future research on optimizing efficiency of patient similarity analysis of big data will cover the following methods:

- **Different batch-size:** Since the amount of similarity comparisons decreases with an increasing patient ID, a fixed batch size will not perform as good as a dynamically adjusted batch-size on the basis of the current position within the data set.
- **Chunking the size of comparisons:** The amount of comparisons is fixed. This means that each patient is not compared to all the patients with a higher ID than himself but rather with a fixed size chunk of other patients, e.g. $\frac{n-1}{2}$. This approach could have advantage from multi-threaded or parallelized environments.
- **The best of both worlds:** Although the elimination of different tools for data pre-processing and analysis reduce the data management effort and improves efficiency, it might still be advantageous to exploit the power of advanced data analysis tools. The best of both worlds can be achieved by using an integrated implementation. For instance, integrated R-support is offered by MonetDB [40, 41]. This allows patient similarity analysis methods implemented in R to be executed over the database. Therefore, it reduces the start-up time of analysis and potentially eliminates intermediate steps that complicate the calculation.
- **Parallelize the patient similarity computing:** Since “pairwise PSM computation is very much parallelizable” [4], big data analysis tools that support parallel computing such as Apache Hadoop can be considered. However, minimizing latency of accessing EMR data and applicability of implementation should be considered. Another approach of parallelization can be achieved by GPU database systems [42]. For instance, such a GPU database candidate is MapD which besides the parallelization inside the GPU also provides the efficient columnar storage.

Apart from optimization, the use of other PSMs that are mentioned by [2] and a comparison to ours is worthwhile. Finally, to achieve an efficient prediction and computation combination, an evaluation of the different implementations of patient similarity algorithms along with the selected technologies will be beneficial.

References

- [1] S.-A. Brown, Patient similarity: Emerging concepts in systems and precision medicine, *Frontiers in physiology* 7 (2016) 561.
- [2] A. Sharafoddini, J. A. Dubin, J. Lee, Patient similarity in prediction models based on health data: a scoping review, *JMIR medical informatics* 5 (1) (2017) e7.

- [3] J. Lee, J. A. Dubin, D. M. Maslove, Mortality prediction in the ICU, in: *Secondary Analysis of Electronic Health Records*, Springer, 2016, pp. 315–324.
- [4] J. Lee, D. M. Maslove, J. A. Dubin, Personalized mortality prediction driven by electronic medical data and a patient similarity metric, *PloS one* 10 (5) (2015) e0127428.
- [5] A. Gottlieb, G. Y. Stein, E. Ruppin, R. B. Altman, R. Sharan, A method for inferring medical diagnoses from patient similarities, *BMC medicine* 11 (1) (2013) 194.
- [6] M. Barkhordari, M. Niamanesh, ScaDiPaSi: an effective scalable and distributable MapReduce-based method to find patient similarity on huge healthcare networks, *Big Data Research* 2 (1) (2015) 19–27.
- [7] C. Ordonez, Data set preprocessing and transformation in a database system, *Intelligent Data Analysis* 15 (4) (2011) 613–631.
- [8] K. Sridhar, Modern column stores for big data processing, in: *International Conference on Big Data Analytics*, Springer, 2017, pp. 113–125.
- [9] C. Ordonez, Can we analyze big data inside a DBMS?, in: *Proceedings of the sixteenth international workshop on Data warehousing and OLAP*, ACM, 2013, pp. 85–92.
- [10] N. Garcelon, A. Neuraz, V. Benoit, R. Salomon, S. Kracker, F. Suarez, N. Bahi-Buisson, S. Hadj-Rabia, A. Fischer, A. Munnich, et al., Finding patients using similarity measures in a rare diseases-oriented clinical data warehouse: Dr. warehouse and the needle in the needle stack, *Journal of biomedical informatics* 73 (2017) 51–61.
- [11] L. Li, W.-Y. Cheng, B. S. Glicksberg, O. Gottesman, R. Tamler, R. Chen, E. P. Bottinger, J. T. Dudley, Identification of type 2 diabetes subgroups through topological analysis of patient similarity, *Science translational medicine* 7 (311) (2015) 311ra174.
- [12] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, R. G. Mark, MIMIC-III, a freely accessible critical care database, *Scientific data* 3 (2016) 160035.
- [13] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, J. N. Clore, Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records, *BioMed research international* 2014.
- [14] M. Lichman, UCI machine learning repository (2013).
URL <http://archive.ics.uci.edu/ml>
- [15] M. Hoogendoorn, A. el Hassouni, K. Mok, M. Ghassemi, P. Szolovits, Prediction using patient comparison vs. modeling: A case study for mortality prediction, in: *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the, IEEE, 2016*, pp. 2464–2467.
- [16] S. E. Abdelrahman, B. E. Bray, Frequency tree clustering for icu mortality analytics using graph databases, in: *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on, IEEE, 2016*, pp. 813–817.
- [17] D. J. Abadi, S. R. Madden, N. Hachem, Column-stores vs. row-stores: How different are they really?, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, 2008*, pp. 967–980.
- [18] S. Wang, X. Li, L. Yao, Q. Z. Sheng, G. Long, et al., Learning multiple diagnosis codes for ICU patients with local disease correlation mining, *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11 (3) (2017) 31:1–31:21.
- [19] J. Sun, D. Sow, J. Hu, S. Ebadollahi, A system for mining temporal physiological data streams for advanced prognostic decision support, in: *10th International Conference on Data Mining (ICDM), IEEE, 2010*, pp. 1061–1066.
- [20] M. A. Morid, O. R. L. Sheng, S. Abdelrahman, PPMF: A patient-based predictive modeling framework for early ICU mortality prediction, *arXiv preprint arXiv:1704.07499* (2017) 1–10.
- [21] F. Wang, J. Hu, J. Sun, Medical prognosis based on patient similarity and expert feedback, in: *21st International Conference on Pattern Recognition (ICPR), IEEE, 2012*, pp. 1799–1802.
- [22] M. Panahiazar, V. Taslimitehrani, N. L. Pereira, J. Pathak, Using EHRs for heart failure therapy recommendation using multidimensional patient similarity analytics, *Studies in health technology and informatics* 210 (2015) 369.
- [23] C. Ordonez, Statistical model computation with udfs, *IEEE Transactions on Knowledge and Data Engineering* 22 (12) (2010) 1752–1765.
- [24] L. Anthony Celi, R. G. Mark, D. J. Stone, R. A. Montgomery, “Big data” in the intensive care unit: Closing the data loop (2013).
- [25] M. C. Data, *Secondary Analysis of Electronic Health Records*, Springer, 2016.
- [26] MIMIC-III critical care database: Documentation and website, accessed on: October 2017.
URL <http://mimic.physionet.org>

- [27] M. Saeed, R. Mark, A novel method for the efficient retrieval of similar multiparameter physiologic time series using wavelet-based symbolic representations, in: AMIA Annual Symposium Proceedings, Vol. 2006, American Medical Informatics Association, 2006, p. 679.
- [28] J. Sun, D. Sow, J. Hu, S. Ebadollahi, Localized supervised metric learning on temporal physiological data, in: 20th International Conference on Pattern Recognition (ICPR), IEEE, 2010, pp. 4149–4152.
- [29] J. Han, J. Pei, M. Kamber, Data mining: concepts and techniques, Elsevier, 2011.
- [30] Y.-J. Park, B.-C. Kim, S.-H. Chun, New knowledge extraction technique using probability for case-based reasoning: application to medical diagnosis, *Expert Systems* 23 (1) (2006) 2–20.
- [31] Columnar storage, accessed on: October 2017 (2012).
URL http://docs.aws.amazon.com/redshift/latest/dg/c_columnar_storage_disk_mem_mgmt.html
- [32] S. Idreos, F. Groffen, N. Nes, S. Manegold, K. S. Mullender, M. L. Kersten, Monetdb: Two decades of research in column-oriented database architectures, *IEEE Data Eng. Bull.* 35 (1) (2012) 40–45.
- [33] W. Raghupathi, V. Raghupathi, Big data analytics in healthcare: promise and potential, *Health information science and systems* 2 (1) (2014) 3.
- [34] P. B. Jensen, L. J. Jensen, S. Brunak, Mining electronic health records: towards better research applications and clinical care, *Nature Reviews Genetics* 13 (6) (2012) 395.
- [35] Y. Wang, L. Kung, T. A. Byrd, Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations, *Technological Forecasting and Social Change* 126 (2018) 3–13.
- [36] L. Wiese, *Advanced Data Management for SQL, NoSQL, Cloud and Distributed Databases*, DeGruyter, 2015.
- [37] L. W. Chan, Y. Liu, T. Chan, H. K. Law, S. C. Wong, A. P. Yeung, K. Lo, S. Yeung, K. Kwok, W. Y. Chan, et al., Pubmed-supported clinical term weighting approach for improving inter-patient similarity measure in diagnosis prediction, *BMC medical informatics and decision making* 15 (1) (2015) 43.
- [38] B. J. Wells, K. M. Chagin, A. S. Nowacki, M. W. Kattan, Strategies for handling missing data in electronic health record derived data, *eGEMs* 1 (3) (2013) 1035.
- [39] A. E. Johnson, M. M. Ghassemi, S. Nemati, K. E. Niehaus, D. A. Clifton, G. D. Clifford, Machine learning and decision support in critical care, *Proceedings of the IEEE* 104 (2) (2016) 444–466.
- [40] H. Muehleisen, A. Damico, M. Raasveldt, T. Lumley, M. D. Team, Package MonetDBLite, accessed on: October 2017 (Aug 2017).
URL <https://cran.r-project.org/web/packages/MonetDBLite/MonetDBLite.pdf>
- [41] H. Muehleisen, MonetDBLite for R, accessed on: October 2017 (Nov 2015).
URL <https://www.monetdb.org/blog/monetdblite-r>
- [42] S. Breß, M. Heimel, N. Siegmund, L. Bellatreche, G. Saake, Gpu-accelerated database systems: Survey and open challenges, in: *Transactions on Large-Scale Data-and Knowledge-Centered Systems XV*, Springer, 2014, pp. 1–35.