Noise-tolerant Similarity Search in Temporal Medical Data

Luca Bonomi[1], Liyue Fan[2], Xiaoqian Jiang[3]

Abstract

Temporal medical data are increasingly integrated into the development of data-driven methods to deliver better healthcare. Searching such data for patterns can improve the detection of disease cases and facilitate the design of preemptive interventions. For example, specific temporal patterns could be used to recognize low-prevalence diseases, which are often under-diagnosed. However, searching these patterns in temporal medical data is challenging, as the data are often noisy, complex, and large-scale. In this work, we propose an effective and efficient solution to search for patients who exhibit conditions that resemble those specified in the input query. In our solution, we propose a similarity notion based on the Longest Common Subsequence (LCSS), which is used to measure the similarity between the query and the patient's temporal medical data and to ensure robustness against noise in the data. Our solution adopts locality sensitive hashing techniques to address the high dimensionality of medical data, by embedding the recorded clinical events (e.g., medications and diagnosis codes) into compact signatures. To perform the pattern search in large EHR datasets, we propose a filtering approach based on tandem patterns, which effectively identifies candidate matches while discarding irrelevant data. The evaluations conducted using a real-world dataset demonstrate that our solution is highly accurate while significantly accelerating the similarity search.

Keywords: Temporal Medical Data, Pattern Search, Similarity Search, EHR Data

1. Introduction

The wide adoption of electronic healthcare records (EHRs) has enabled the collection of an unprecedented quantity of temporal medical data. A variety of studies have shown that analyzing temporal patterns in EHR data, e.g., a sequence of medical events occurring over time, holds great promises for facilitating clinical decisions and improving patient care [1, 2, 3]. Current data-driven methods rely on two tasks, i.e., *pattern extraction* and *pattern search*, to integrate temporal patterns into clinical applications.

For pattern extraction, the goal is to discover useful patterns for predictive analysis. Typically, these patterns are unknown *a priori*, and are discovered using machine learning methods [4, 5, 6, 7, 8, 9] and data mining techniques [10, 11, 12, 13]. Pattern search generally aims at efficiently searching for patients who exhibit known patterns (e.g., a sequence of diagnoses over time), which are given as input queries. Several solutions have been proposed by extending traditional database query languages [14, 15], and using a variety of search methods [16, 17, 18, 19]. In fact, pattern search can enable clinicians to identify patients with target health conditions, facilitating the diagnosis and preemptive intervention, especially for low prevalence diseases that are likely to be under-diagnosed. As an example, the query: "Find

[1]lbonomi@ucsd.edu, UCSD Health Department of Biomedical Informatics, University of California, San Diego

[2]liyue.fan@uncc.edu, College of Computing and Informatics at the University of North Carolina at Charlotte

[3]Xiaoqian.Jiang@uth.tmc.edu, UTHealth School of Biomedical Informatics, Houston

all patients of age ≤ 5 years who experience fever for more than 5 days, or fever in conjunction with cervical lymphadenopathy" could be used to facilitate the diagnose of Kawasaki disease, which is a rare disease that affects children [20].

In this work, we focus on the task of pattern search in temporal medical data. Our solution aims at advancing current pattern search approaches by enabling clinicians to effectively identify patients whose data resemble the patterns of interest. Specifically, current approaches mainly focus on *exact* search, which provides limited robustness against noise in the data. Consequently, data that contain errors, missing values, and patient-specific variations may not match the query patterns exactly, potentially resulting in patients to be under-diagnosed. Although recent research (e.g., [18]) has proposed to use similarity measures in pattern search, those solutions may incur significant computational burdens for high-dimensional and large-scale temporal medical data, which limits their applicability in real-world settings.

To this end, we propose a novel solution that enables clinicians to perform robust and efficient pattern similarity search in large-scale EHR datasets. Our solution is based on a *similarity measure*, which quantifies the variations in the medical events (e.g., diagnoses and medications) and in the temporal dimension (e.g., the time of the event). Furthermore, we develop *efficient* and *accurate* similarity search methods to address the high-dimensional (e.g., multiple diagnosis codes given to the patient at the same time) and longitudinal (e.g., observations of the patient over time) nature of temporal medical data, to enable pattern search over large datasets. Our specific contributions are reported as follows.

- To perform similarity search in EHR data, we adopt a similarity measure based on the *Longest Common Subsequence* (LCSS), which provides robustness against noise in the recorded events (i.e., event-level distortion) and in the event times (i.e., temporal distortion). As a result, patients exhibiting patterns that *resemble* the query pattern can be detected, as long as their LCSS similarity is within a given threshold.
- Evaluating the LCSS similarity between the query pattern and every patient's data can be computationally challenging in practice. Therefore, we construct an offline probabilistic index based on locality sensitive hashing (LSH) techniques, embedding high-dimensional medical events into compact signatures while allowing for accurate similarity evaluation. We further develop an effective *filtering* approach to discard patients' data in which the query pattern could not occur, and only the remaining data is evaluated for LCSS similarity against the query pattern. Our method leverages gapped dyadic patterns (i.e., events with a temporal gap) and can recover false negatives caused by the LSH index.
- We conducted extensive evaluations on a large scale, real-world EHR dataset. Our results show that our solution guarantees zero false positive and with high probability zero false negative, while significantly accelerating the similarity search. In addition, we consider a case study on inflammatory bowel disease where temporal patterns are utilized to build a classifier. Our study shows that temporal patterns may help improving the performance of distance-based prediction models compared to the use of static features (e.g., single diagnosis codes).

The rest of the paper is organized as follows. Section 2 reviews related works on similarity search. Section 3 introduces the background, definitions, and an overview of our solution. In Section 4, we provide a detailed description of our proposed methods. We report comprehensive evaluations on a real-word dataset in Section 5. We discuss potential improvements in Section 6 and conclude the paper in Section 7. Supplemental technical material, such as proofs and additional experiment results, are reported in the Appendix.

2. Related Work

Researches have developed query systems to enable clinicians to search for patients exhibiting clinical conditions with temporal information to analyze large-scale EHR data. Among these works, temporal

query languages, such as TSQL [14, 15], extend traditional SQL queries to account for the time dimension of longitudinal medical events. While these approaches have been applied in practical settings [16], they provide limited robustness in the presence of noise in the data: only those patients who match the query in an exact manner are reported.

Several similarity search methods based on flexible similarity measures have been proposed to develop more robust approaches. For example, Hajihashemi and Popescu [21] designed a framework for detecting health patterns in temporal sequences generated from sensor networks. In their approach, sensor data are mapped into one-dimensional sequences where the similarity is computed using sequence alignment techniques based on the standard Smith-Waterman algorithm. A similar idea has been adopted by Sha et al. [18] to measure patient-to-patient similarity. Specifically, the authors proposed a similarity measure based on the longest common subsequence (LCSS) to capture the similarity between the trajectories of medical events (i.e., lab measurements) among patients. The authors showed that the use of a similarity measure could produce patterns with better predictive power. However, the efficiency of their approach is limited, as the similarity measure relies on the classic Smith-Waterman algorithm, thus requiring quadratic time for each temporally annotated sequence in the data. In a recent work, Giannoula et al. [22] studied the problem of disease progression via temporal patterns. In their work, patients' data are mapped into one-dimensional temporal trajectories representing the relationship between pairs of diseases, where the identification of disease progression patterns is performed using the Dynamic Time Warping (DTW) similarity measure. While this study provides interesting insights on patterns of disease progression, these patterns only provide limited information as they are constructed by taking into account only the primary diagnoses codes, leaving out medications, labs, and procedures that could provide additional important clinical insights. Furthermore, DTW is known to be less robust than LCSS in the presence of noise in the data, while still requiring quadratic time.

In this work, we address the computational limitations of current solutions and propose a more robust similarity notion that better captures the similarity between high dimensional events (e.g., sets of diagnosis codes, medications, and procedures) in temporal data. Specifically, we develop an effective filtering approach that significantly reduces the running time by limiting the similarity evaluation to only a small number of sequences. Our approach uses an LCSS-based similarity measure, which has been shown to be very robust in the presence of noise in the data [23]. Patterns are matched with allowed temporal distortions and similarity for medical event sets (e.g., measured by Jaccard index), which takes into account multiple events recorded at the same time (e.g., a patient may receive more than one diagnosis codes concurrently).

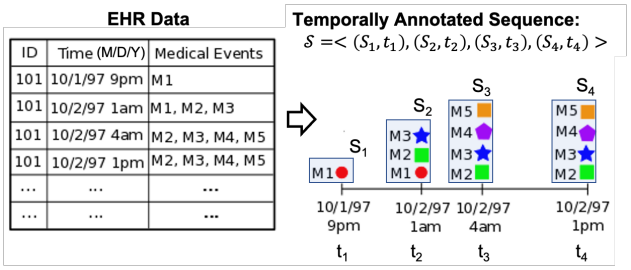


Figure 1: Example of temporal data modeling. EHR data of patient ID=101 are represented with a temporally annotated sequence $\mathcal{S}$, comprising four sets of medical events S_1, S_2, S_3, and S_4 recorded over a period of two days. Each medical event Mi may represent a diagnosis code, procedure, or medication. Multiple events may be recorded at the same time.

3. Preliminaries

Our goal is to identify patient records in temporal medical data (e.g., EHR), which approximately match a query pattern, e.g., progression of clinical conditions over time. This section describes how our approach models temporal medical data, our problem formulation, and an overview of our solution.

3.1. Modeling Temporal Medical Data

Temporal Representation. To perform a similarity search on temporal medical data, we need a well-defined representation to model the recorded

information. Several temporal data models have been proposed previously for analyses and applications [24, 25, 26] (e.g., data mining, classification, prediction). In this work, we use the notion of *temporally annotated sequences*, which has shown promising results in several data mining tasks [26, 19].

Definition 3.1 (Temporally Annotated Sequence)**.** *A temporally annotated sequence $\mathcal{S}$ of length $n > 0$ (i.e., $n = |\mathcal{S}|$) is a sequence of n pairs (S_i, t_i), where S_i is a set of observations representing the recorded medical events (e.g., diagnosis codes, medications), and t_i denotes the time of event, where $t_i < t_{i+1}$ for all i (i.e., temporally ordered).*

An example of a temporally annotated sequence from EHR data is reported in Figure 1. Given sequence $\mathcal{S} = \langle (S_1, t_1), \ldots, (S_n, t_n) \rangle$, $\mathcal{X} = \langle (S_{i_1}, t_{i_1}), \ldots, (S_{i_m}, t_{i_m}) \rangle$ is a *subsequence* of $\mathcal{S}$ if $1 \leq i_1 \leq i_2 \leq \cdots \leq i_m \leq n$.

Similarity Measure. To evaluate the similarity between temporally annotated sequences, we consider a measure based on the *Longest Common SubSequence* (LCSS), which was first proposed for time series data by Vlachos et al. [23]. In this work, we generalize this measure to discrete, set-valued temporal medical data. The LCSS similarity measure is based on the well-known longest common subsequence notion for one-dimensional sequential data, e.g., strings, where the similarity between two sequences is measured by the length of their longest common subsequence. To measure the similarity of multi-dimensional temporal medical data, we consider two parameters ϵ and δ, which define the allowed distortion between set-valued data (i.e., recorded events) and on the time axis (i.e., time of the observation), respectively.

Definition 3.2 (Longest Common SubSequence)**.** *Given two temporally annotated sequences $\mathcal{X} = \langle (X_1, t_1), \ldots, (X_n, t_n) \rangle$ and $\mathcal{Y} = \langle (Y_1, t_1), \ldots, (Y_m, t_m) \rangle$, the ($\delta$,$\epsilon$)-Longest Common SubSequence, denoted as $LCSS_{(\delta,\epsilon)}(\mathcal{X}, \mathcal{Y})$ can be computed using a cumulative similarity function γ among the prefixes of $\mathcal{X}$ and $\mathcal{Y}$. For any pair of indexes $0 < i \leq n$, $0 < j \leq m$ in the sequences $\mathcal{X}$ and $\mathcal{Y}$ respectively, the value of $\gamma_{\delta,\epsilon}(i, j)$ is computes as follows:*

$$\begin{cases} 0 & \text{if } i, j = 0, \\ 1 + \gamma_{\delta,\epsilon}(i-1, j-1) & \text{if } d_e(X_i, Y_j) < 1 - \epsilon \\ & \quad \text{and } |t_i - t_j| \leq \delta, \\ \max\{\gamma_{\delta,\epsilon}(i-1, j), & \\ \gamma_{\delta,\epsilon}(i, j-1)\} & \text{otherwise} \end{cases}$$

where d_e is a distance measure for event sets, $0 < \epsilon \leq 1$, and δ is a positive integer. Finally, $LCSS_{(\delta,\epsilon)}(\mathcal{X}, \mathcal{Y}) = \gamma_{\delta,\epsilon}(|\mathcal{X}|, |\mathcal{Y}|)$.

In practice, any set distance measure may be adopted in Definition 3.2. In this work, we define $d_e(X, Y) = 1 - J(X, Y)$, where $J(X, Y)$ denotes the Jaccard index between two medical event sets X and Y, i.e., $J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. The Jaccard index has been commonly used to measure set similarity, e.g., in recent studies on EHR data [2, 27]. We also provide evaluations with the *containment* similarity measure, i.e., a fraction of the query contained in the patient data, in the supplementary material.

3.2. Problem Formulation

We aim at finding subsequences in the patient's temporal medical data that highly preserve clinically useful patterns given as an input query. Specifically, the input query pattern is modelled as a pair $Q = (P, \Delta T)$, where $P = \langle P_1, \ldots, P_k \rangle$ is a sequence of k event sets and $\Delta T = \langle \Delta t_1, \ldots, \Delta t_{k-1} \rangle$ is a sequence of temporal gaps representing the progression of medical events, e.g., P_{i+1} is observed Δt_i time units after P_i. In addition, δ and ϵ are considered in input, to approximately match the query pattern with a patient's data, when computing their LCSS similarity.

Definition 3.3 (θ-match)**.** *Give parameters δ and ϵ, a subsequence $\mathcal{S}' = \langle (S_{i_1}, t_{i_1}), \ldots, (S_{i_m}, t_{i_m}) \rangle$ is a θ-match for the query $Q = (P, \Delta T)$ if $\frac{LCSS_{(\delta,\epsilon)}(\mathcal{S}', \mathcal{P})}{|\mathcal{P}|} \geq \theta$, where $\mathcal{P} = \langle (P_1, t_{i_1}), \ldots, (P_k, t_{i_1} + \sum_{i=1}^{k-1} \Delta t_i) \rangle$ is the sequence induced by Q and $\mathcal{S}'$.*

Intuitively, $\theta \in [0, 1]$ represents the fraction of the query pattern contained in the subsequence. Higher θ requires the data to contain a higher portion of the query pattern. Given a query Q and the threshold θ, our goal is to identify every subsequence in the data,

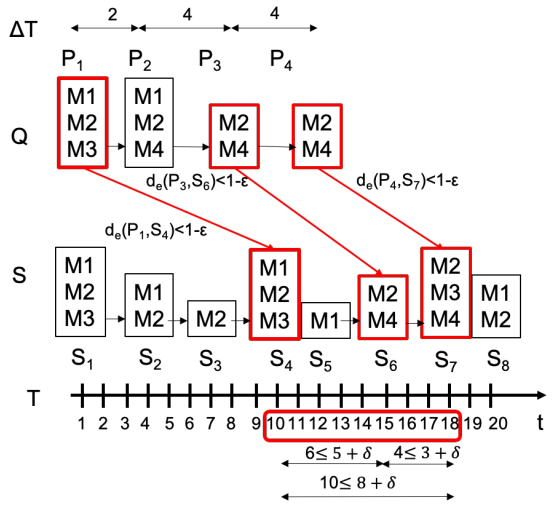


Figure 2: Example of similarity search in temporally annotated sequences. Given sequence $\mathcal{S}$ and query $Q = (P, \Delta T)$, the longest common subsequence between Q and $\mathcal{S}$ for $\delta = 2$ and $\epsilon = 0.65$ is $\mathcal{S}' = \langle (S_4, 10), (S_6, 15), (S_7, 18) \rangle$. Furthermore, this subsequence is a θ-match for the query Q for $\theta = 0.75$, as $\mathcal{S}'$ contains 3 out of 4 event sets in query Q.

which is a θ-match for Q. An illustrative example of the similarity search problem is reported below.

Example 3.1. *Let $M1, \ldots, M4$ denote distinct medical events recorded during a patient's visit (e.g., diagnoses, medications, and procedure codes). A temporally annotated sequence $\mathcal{S}$ as in Figure 2 contains 8 sets of events, observed across $t = 1$ and 20. Consider a query $Q = (P, \Delta T)$, where $P = \langle P_1, P_2, P_3, P_4 \rangle$, $\Delta T = \langle 2, 4, 4 \rangle$ and d_e is computed using the Jaccard index. The subsequence $\mathcal{S}' = \langle (S_4, 10), (S_6, 15), (S_7, 18) \rangle$ is a θ-match for Q in $\mathcal{S}$, for $\delta = 2$, $\epsilon = 0.65$, and $\theta = 0.75$.*

3.3. Overview of our solution

In this paper, we propose a novel framework that enables clinical researchers to query large EHR datasets and identify patients who exhibit conditions specified as temporal query patterns (Figure 3). Specifically, our solution entails three main steps: *indexing*, *filtering*, and *verification*. The three steps combined effectively manage the complex EHR data while reducing the similarity computation to only a small number of temporally annotated sequences relative to the overall dataset.

1. *Indexing Medical Events.* We pre-process the data sequences and embed the sets of medical events into compact signatures that retain the original similarity (e.g., Jaccard). These signatures are indexed using locality-sensitive hashing (LSH) techniques to support efficient similarity searches given the input query.
2. *Filtering.* We perform a filtering step to identify candidate subsequences in the dataset that are likely to be a match. Our filtering approach uses *tandems* (i.e., dyadic patterns separated by a flexible temporal gap) extracted from the input query Q. Subsequences that share a large number of tandems with the query Q are likely to be candidate θ-matches for Q. As a result, this step significantly reduces the number of sequences that need to be verified.
3. *Verification.* The surviving subsequences after the filtering step are verified to determine if they are θ-matches for Q. This step entails computing the LCSS measure between the query and each subsequence (as in Definition 3.3), which is achieved with dynamic programming [18].

Given the high computational cost for the similarity measure between a data sequence and the input query, it is important to develop indexing and filtering methods that improve the efficiency and lower the number of data sequences for the verification step. On the other hand, indexing and filtering should not be overly restrictive to miss real matches, i.e., incurring false negatives. In Section 4, we propose innovative methods for indexing and filtering that strike a balance between computational efficiency and accuracy.

4. Material and Methods

In this section, we provide a detailed description of the steps in our method as shown in Figure 3. First, in the *offline* phase, we index the medical events in

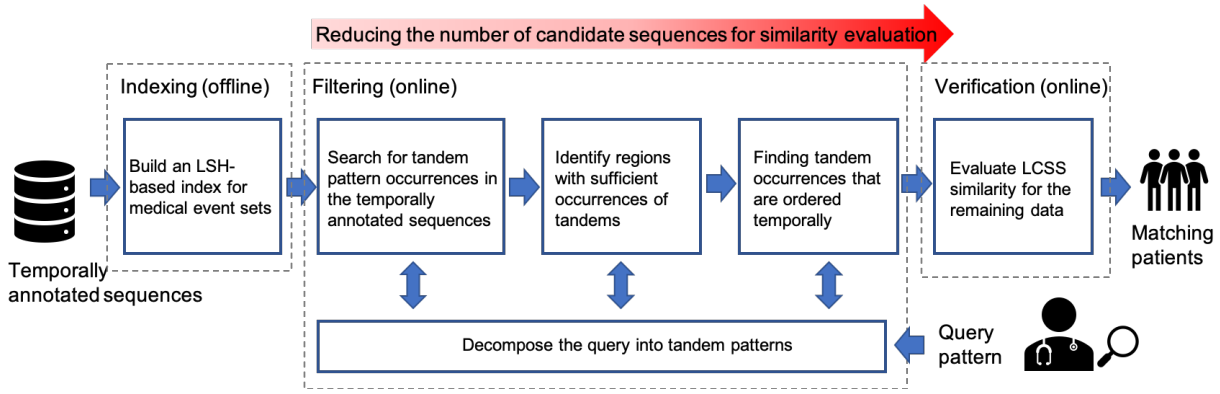


Figure 3: Overview of our proposed similarity search solution.

the data by mapping them into compact signatures while preserving their similarity. Second, we illustrate our similarity search approach using filtering and verification, given a query *on-the-fly*.

4.1. Offline Indexing for Medical Events

Searching temporal medical data can be computationally challenging on large datasets, requiring to examine every event set (e.g., a set of diagnosis codes and medications) recorded at each observation (e.g., one patient encounter). Here we present our approach to index medical events offline, and the index structures can be used to search for medical event sets efficiently in the online phase. Specifically, our approach adopts *locality-sensitive hashing* (LSH) techniques to embed the high dimensional set-valued data (i.e., medical event sets) into compact signatures and enable to efficiently search for similar medical event sets (i.e., within ϵ threshold) in the online phase.

Our approach adopts *LSH-MinHash* [28] to estimate the Jaccard similarity between event sets. Given the similarity threshold ϵ in a query Q, the LSH-index can be optimally constructed to minimize false positives and false negatives caused by the probabilistic nature of LSH. However, when answering queries with different thresholds (i.e., values of ϵ), the index is no longer optimal and may yield higher errors. Moreover, reconstructing the LSH-index for every query is infeasible, as it would incur significant computational and storage overhead. To this end, we develop a workload-aware LSH-index, which comprises a small number of sub-indexes (i.e., LSH-MinHash), which achieves minimum error given a workload of queries.

Workload-Aware Index Construction. Our approach constructs the index by considering a workload of queries $W = \{Q_1, Q_2, \ldots, Q_n\}$, which may represent a set of historical or batched similarity search queries. Our approach utilizes the similarity thresholds associated with queries in W (i.e., ϵ_i for every Q_i) to optimally construct the index, such that the error can be minimized for queries in W as well as future queries with similar thresholds. Specifically, we define the range of similarity thresholds for W as $[\epsilon^-, \epsilon^+)$, where $\epsilon^- = \min_{i=1\ldots n}\{\epsilon_i\}$ and $\epsilon^+ = \max_{i=1\ldots n}\{\epsilon_i\}$. By incorporating the distribution of query similarity thresholds, our approach constructs m LSH indexes that minimize the overall error for the workload. An overview of the offline indexing is depicted in Figure 4. When the workload information is unavailable, our method assumes an uniform distribution of similarity thresholds for input queries in the range $[\epsilon^- = 0, \epsilon^+ = 1)$.

Typically, an LSH-index is configured with two parameters (b, r), where b is the number of bands and r is the number of rows per band used in computing the hash signatures [28]. Given a similarity threshold ϵ, we can compute the false positive rate $FP_{b,r}(\epsilon)$ and the false negative rate $FN_{b,r}(\epsilon)$ as a function of the index parameters b and r. We define the index

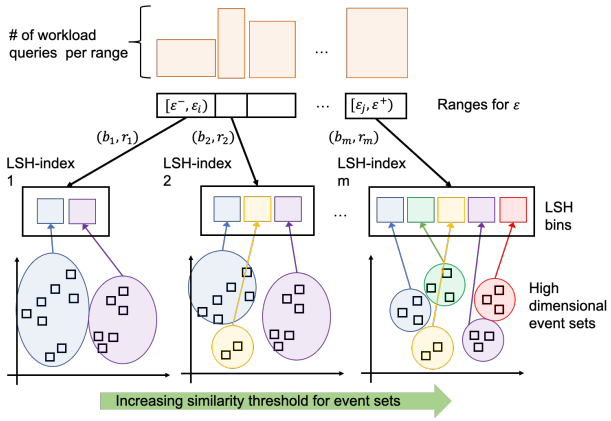


Figure 4: An illustrative example of offline indexing. Given a workload of queries, we identify the range, i.e., ϵ^- and ϵ^+ for the queries' similarity thresholds. Our approach constructs an optimal partition of the similarity threshold range, and for each partition an LSH-index i with parameter (b_i, r_i). For each LSH-index, the high dimensional medical event sets are mapped into compact signatures and hashed into bins, where similar event sets are placed in the same bin. In the online phase, the index constructed for the range $[\epsilon_i, \epsilon_j)$ is used to answer queries with $\epsilon \in [\epsilon_i, \epsilon_j)$.

error for the similarity threshold ϵ (i.e., $err_{b,r}(\epsilon)$) as the weighted average between false positive and false negative:

$$err_{b,r}(\epsilon) = w_{FP} \times FP_{b,r}(\epsilon) + w_{FN} \times FN_{b,r}(\epsilon) \quad (1)$$

where w_{FP} and w_{FN} are weights for the false positive and false negative rates, respectively. Given a similarity range $[\epsilon_l, \epsilon_r)$, the total error for answering all queries in W with a similarity threshold in the range can be defined as:

$$\begin{aligned} E_{b,r}(\epsilon_l, \epsilon_r) &= \sum_{Q_i \in W, \epsilon_i \in [\epsilon_l, \epsilon_r)} err_{b,r}(\epsilon_i) \\ &\leq |W(\epsilon_l, \epsilon_r)| \times \max_{\epsilon_i \in [\epsilon_l, \epsilon_r)} \{err_{b,r}(\epsilon_i)\} \end{aligned}$$

where $|W(\epsilon_l, \epsilon_r)|$ denotes the number of queries in W with $\epsilon_i \in [\epsilon_l, \epsilon_r)$.

Our goal is to construct a small number (m) of LSH indexes for the workload W, where each index is responsible for answering the queries with $\epsilon \in [\epsilon_l, \epsilon_r)$, and such that the overall error is minimized. Specifically, we aim at identifying an m-partition of $[\epsilon^-, \epsilon^+]$, such that $[\epsilon^-, \epsilon^+] = [\epsilon^0, \epsilon^1) \bigcup [\epsilon^1, \epsilon^2) \bigcup \cdots \bigcup [\epsilon^{m-1}, \epsilon^m]$ where each sub-range $[\epsilon^j, \epsilon^{j+1})$ is associated with an LSH-index j constructed with parameters b_j and r_j and the overall error $Err^*_m(\epsilon^-, \epsilon^+) = \sum_{j=0}^{m-1} E_{b_j, r_j}(\epsilon^j, \epsilon^{j+1})$ is minimized. In Appendix A.1, we show that this problem can be efficiently solved using dynamic programming. As a result of this partitioning process, we obtain m LSH indexes, each tuned to answer the queries with similarity in the predefined range. In the online phase, the m LSH-indexes will be used to efficiently retrieve all medical events within the similarity threshold of the input query.

4.2. Online Filtering and Verification

Recall in Figure 3, given a query pattern, our framework performs online filtering and verification to identify matching patients. Specifically, for an input query Q, the filtering step uses the LSH-index to identify those sequences that are likely to match the query Q, while discarding the others. Subsequently, in the verification step, only the surviving sequences are processed to determine if they exhibit an actual θ-match for the query Q. Our intuition is that in practical applications, only a small percentage of data sequences would match the query pattern Q. As a result, our approach may significantly accelerate the similarity search compared to the exhaustive approach, i.e., by verifying every sequence in the data.

4.2.1. Filtering via Tandem Patterns

In the filtering step, we utilize *tandem patterns* (i.e., two event sets separated by a variable time gap) to capture the temporal dependence between medical events in the query and identify those temporally annotated sequences that may contain a θ-match. Specifically, given a query of length k denoted as $Q = (P = \langle P_1, \ldots, P_k \rangle, \Delta T = \langle \Delta t_1, \ldots, \Delta t_{k-1} \rangle)$, we decompose Q into a set of tandems $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$, $1 \leq i < j \leq k$, where P_i and P_j are separated by the temporal gap $\Delta t_{ij} = \sum_{l=i}^{j-1} \Delta t_l$ (see Figure 6a for examples). In the following, we first search for the occurrences of these tandems in the temporally annotated sequences, and then use these

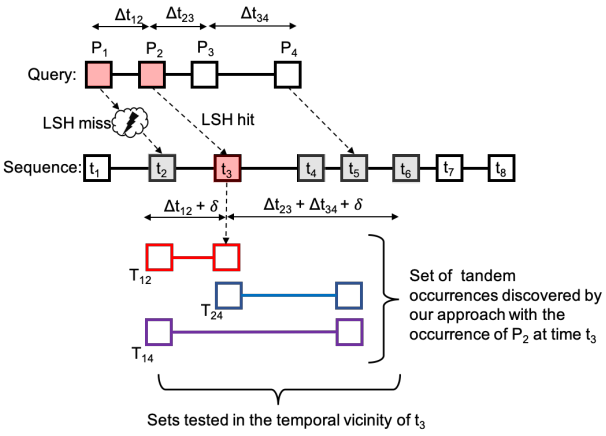


Figure 5: Searching for tandem pattern occurrences. In this example, the LSH index misses the occurrence of P_1 at time t_2 in the data sequence. As a result, the simple tandem search approach will miss the occurrences of T_{12} and T_{14} in the sequence, for which P_1 is the first component. With our approach, when the LSH index detects the set P_2 at time t_3, we do not only find the occurrence of T_{24} but also test the event sets in the vicinity of P_2 in the sequence for possible occurrence of other tandems, using a wider temporal gap adjusted to account for the temporal distortion δ. In this way, the occurrences of tandems T_{12} and T_{14} can be recovered.

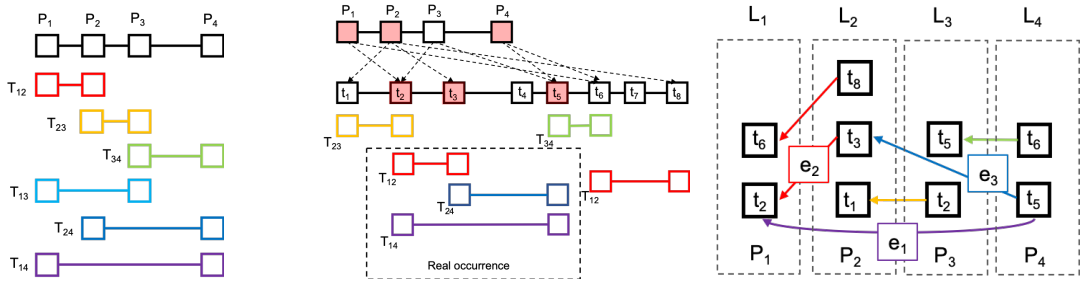
tandems to filer out those subsequences that cannot be a θ-match for Q.

Searching for Tandem Occurrences. To detect the occurrences of the tandems, the LSH-index constructed offline will be utilized to identity where the tandem event sets occur in the data sequences. Specifically, the occurrences of a tandem $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$ can be found by identifying the first component P_i in the data and testing whether the second component P_j occurs in the vicinity of P_i. A problem with this approach is that if the first component P_i is missed due to the probabilistic nature of LSH, the occurrence of T_{ij} will be also missed. To overcome this limitation, we propose a different approach, in which the detection of T_{ij} does not solely rely on the first component P_i: an occurrence of any set P_m of Q can be used to identify the occurrences of T_{ij}. Our approach can recover a missed occurrence of T_{ij} as long as some event sets of the query Q occur in its temporal vicinity. An illustrative example is reported in Figure 5. Here, we briefly describe our approach as follows. Each time an occurrence X_m of a set P_m is detected in the data, all set pairs X_i and X_j in the temporal vicinity of X_m are tested to determine whether (X_i, X_j) matches a real tandem. As a result, a tandem occurrence is missed, only when all the event sets in Q are missed, thus exponentially reducing the false negative probability compared to the standard approach. In the evaluation section, we refer to the former approach as *single-seed expansion* and the latter as *multiple-seed expansion*, respectively. The accuracy and running time for these methods are analyzed in Appendix A.2.

Filtering with Tandems. Given the tandem occurrences, it is challenging to determine whether the tandem occurrences warrant a θ-match for Q. As a simple filtering criterion, one could use the number of shared tandems between the query and the subsequence in the data (Figure 6b). However, this approach may have little filtering power, resulting in a large number of candidate subsequences that require verification. For example, a subsequence containing out-of-order tandems is likely to pass this simple count-based filter even though it does not exhibit a real θ-match. Therefore, to design a more effective filter, we must consider the temporal order of the tandems found.

Given a length-k query $Q = (P, \Delta T)$ where $P = \langle P_1, P_2, \ldots, P_k \rangle$, and a temporally annotated sequence $\mathcal{S}$, our method constructs a directed acyclic graph $G = (V, E)$, where the node set V is partitioned into k layers $L_1, L_2, \ldots, L_k$, and each $v \in L_i$ denotes an occurrence of set P_i. A directed edge e between layers L_i and L_j represents an occurrence for the tandem $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$. As shown in our previous study [19], this graph model can be used for effectively matching patterns in temporally annotated sequences. In this work, we adapt the graph model to track the tandems in $\mathcal{S}$ as well as their temporal order.

Specifically, our method constructs the graph G by processing the tandems starting from the leftmost set P_1, and proceed in the order as they appear in Q. For each set P_i, a new node v with label t_i is added



(a) Decomposition of the input query into tandem patterns.

(b) Occurrences of the tandem pattens in the temporally annotated sequence.

(c) We construct $G = (V, E)$ a directed acyclic graph to identify tandems that are in order.

Figure 6: Filtering with Tandem Patterns: (a) Decomposition of the input query into tandem patterns. In this case the input query has four sets of medical events, resulting into six different tandem patterns, which are used to find potential occurrences of the query pattern. (b) Tandem occurrences in the temporally annotated sequences. We point out that tandems may occur out of order in the sequence without forming a real θ-match for the input query. In our example, the tandems in the dashed box represent a real match of the query pattern at time t_2. (c) Acyclic directed graph for coverage computation. In this example, the coverage for the edge e_1 is two, as there are two paths that connect the nodes t_5 and t_2: the path formed by e_1 itself and the path formed by e_3 and e_2. The tandems T_{14}, T_{24}, and T_{12} associated with these edges preserve their original order, identifying the occurrence at time t_2.

to the layer L_i when an occurrence of P_i is identified in the sequence at time t_i. Edges are added to capture tandem occurrences and form paths in G across multiple layers. Because the set occurrences are processed in order during the graph construction, any pair of consecutive edges $e_1 = (v_i, v_j)$ and $e_2 = (v_j, v_z)$ forming a path in G, correspond to tandems $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$ and $T_{jz} = \{(P_j, P_z), \Delta t_{jz}\}$ that share the same event set P_j. Therefore, the number of paths between two nodes provides information about how many tandems preserve their original order. Our idea is to use these paths to identify a θ-match for the query (Figure 6c). Specifically, given an edge $e = (v_i, v_j)$, we denote by $c(e)$ the number of paths from v_i to v_j, named the coverage of e. Intuitively, edges with large coverage indicate that a large number of the query's tandems are in order, thus the subsequence is likely to be a θ-match for the input query.

Our method adopts a two-step filtering criterion based on the following necessary conditions extracted from G and $c(e)$.

Lemma 4.1. *Let t_i be a time index in a temporally annotated sequence representing a θ-match for a length-k query pattern. Then the following conditions hold:*

1. *Counting: There are at least $th_1 = \frac{\alpha \times (\alpha-1)}{2} - 1$ distinct tandems shared between the subsequence starting at time t_i and the input query pattern.*

2. *Coverage: In the directed acyclic graph $G = (V, E)$ there exists an edge e, such that $c(e) \geq th_2 = 2^{\alpha}/4$*

where $\alpha = \lfloor \theta \times k \rfloor$.

Only when both conditions are satisfied, the subsequence will be verified subsequently. The technical analysis is reported in Appendix A.3 and Appendix A.4.

Verification. The verification step examines the surviving subsequences and determines the presence of θ-matches for the query using the (δ, ϵ)-LCSS similarity measure. This step eliminates false positives from the final matching results. In our evaluations on real-world clinical data, only a small fraction of the sequences pass the filtering step, resulting in a significant speed-up compared to exhaustively computing the LCSS similarity distance for each data sequence.

5. Empirical Evaluation

Data. We adopt the de-identified real-word clinical dataset MIMIC-III [29], which comprises over 58,000 ICU hospital admissions for 38,645 adults and 7,875 neonates. For each patient, a temporally annotated sequence $\mathcal{S} = \langle (S_1, t_1), \ldots, (S_n, t_n) \rangle$ is constructed, where each S_i contains the diagnoses, procedures, and medications recorded by the institution at the time of hospitalization. Each t_i has been adjusted to record the number of days elapsed since the patient's first hospitalization. As we only consider patients that have at least two observations, i.e., $n \geq 2$, the total number of temporally annotated sequences constructed from the original data is 46,498 with 650,245 observations in total.

Parameters. We evaluate our methods by varying parameter values in Table 1 and the default values are boldfaced. Among them, θ, ϵ, and δ allow to retrieve data sequences that are sufficiently similar to the query patterns. It may be beneficial to set higher values for θ and ϵ and lower values for δ when the user queries known patterns and obtains results that well preserve the query patterns. On the other hand, lower values of θ and ϵ (and higher values of δ) allow data sequences to match the input query in a more flexible manner, and therefore may be more suitable when data contain high noise or the user is uncertain about the query patterns.

Metrics. We measure the recall of the retrieved sequences and the running time of our approach in seconds. The ground truth is provided by computing the similarity of every sequence in the dataset to a given query, i.e., exhaustive verification. Note that our approach produces no false positives thanks to the verification step, but possibly false negatives due to the probabilistic nature of LSH-indexing. The results below were obtained using the Jaccard index between event sets. Experiments conducted using the containment similarity measure are reported in Appendix B.

5.1. Evaluation for Indexing

We compare our workload-aware index approach based on dynamic programming (denoted by `DP-Index`) with LSH-Forest [30], which searches nearest neighbor event sets approximately when the similarity threshold ϵ varies. To construct `DP-Index`, we consider a wide range of ϵ values, i.e., $[0.25, 0.9]$, to enable our approach to answer a variety of query workloads. To evaluate the indexing performance, we generated 100 target event sets for every ϵ value and searched both indexes for those sets. We set $W_{FP} = 0.3$ and $W_{FN} = 0.7$ for the indexing error in Equation 1. Because the reported false positives (i.e., event sets that do not meet the ϵ similarity threshold) will be removed by the verification step eventually, we assign a higher weight to the false negative. The recall results for different values of ϵ are reported in Figure 7. We observe that for both approaches, the recall increases as ϵ increases (i.e., requiring higher similarity) and our `DP-Index` consistently dominates LSH-Forest. It shows that LSH-Forest, designed for finding top-k nearest neighbors, is only suitable for answering queries with high similarity thresholds, e.g., $\epsilon \geq 0.8$. We notice that the number of indexes (m) used in our approach does not significantly impact the performance for $\epsilon \geq 0.5$. In practice, DP-Index works well even when only one or two indexes are constructed. This is not surprising, since one carefully constructed LSH-index may capture the structure of the data [31].

Furthermore, our index structure is built offline in an efficient manner. Additional evaluations demonstrating the scalability of our indexing method are reported in the Appendix B.

Parameter	Description	Values
k	Length of the query pattern	[2,3,**4**,6]
ϵ	Event-level similarity	[0.25,0.5,**0.75**,0.9]
δ	Temporal distortion in days	[1,**2**,7,14,31]
θ	Threshold for LCSS similarity	[0.5,**0.75**,0.9,1]
d	Data dimensionality	[2,4,**10**,20]

Table 1: Parameter settings. Default values are boldfaced.

5.2. Evaluation for Filtering

We evaluate the impact of tandem-based filtering on similarity search. We compare two tandem search methods of Section 4.2.1, namely: single-seed expansion (`TF`) and multiple-seed expansion (`TF`*). The

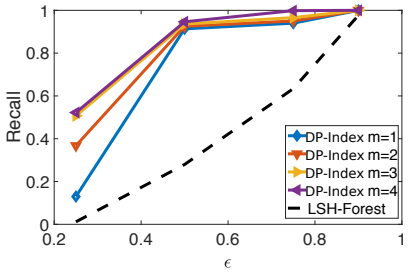


Figure 7: Indexing performance vs. LSH-Forest [30]. m: number of indexes used in our approach.

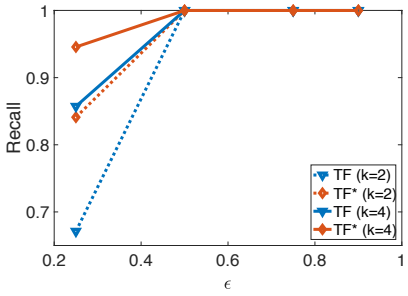


Figure 8: Performance of Similarity Search with Tandem-based Filtering. k: length of the query pattern.

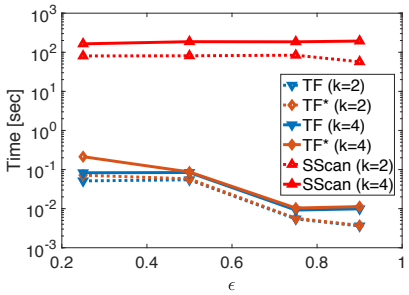


Figure 9: Running Time of Similarity Search Compared to `SScan` [18].

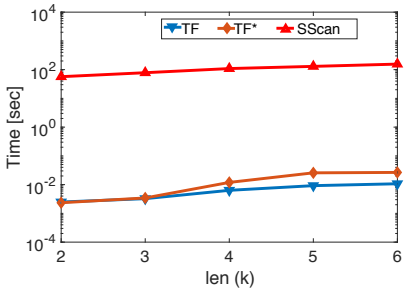


Figure 10: Running time vs. query length.

ground truth of similarity search is computed using an adaptation of the approach proposed by Sha et al. [18] to suit our similarity notion, which evaluates the similarity between every sequence in the dataset and the query pattern (`SScan`), i.e., without any filtering. Queries $Q = (P, T)$ were constructed by randomly sampling length-k subsequences from the temporally annotated sequences of the MIMIC-III patients. Each experiment was run with 20 queries, and the average result was reported.

Varying the similarity threshold ϵ. Figure 8 reports the recall of similarity search, i.e., finding θ-match of a given query using (δ,ϵ)-LCSS measure. First, we observe that the recall improves significantly as the similarity threshold ϵ for event sets increases, thanks to our LSH-based index, which is more accurate with higher ϵ values. Second, we notice that given the query length k, searching tandems with the `TF`* method achieves significantly higher recalls than `TF`, which demonstrates the robustness of `TF`* in finding tandem occurrences. By using a longer query pattern (from $k = 2$ to $k = 4$), higher recalls are achieved for both `TF` and `TF`*, as more tandems will be searched in the filtering step and fewer false negatives are incurred. From the run time evaluation in Figure 9, we observe that our filtering-based approach achieves a speed-up of several orders of magnitude compared to the baseline `SScan`.

Impact of the query length (k). Figure 10 reports the running time while varying the length of the query pattern. For all the approaches, we observe that the running time increases when longer patterns are used in the query Q. Intuitively, longer patterns require higher computational cost for the LCSS similarity evaluation, and for our approach, they also generate a greater number of tandems. The results show that our filtering-based approach significantly outperforms the baseline `SScan`. Between our tandem search methods, `TF`* incurs a very small computational overhead compared to `TF`, when queries are longer ($k \geq 4$).

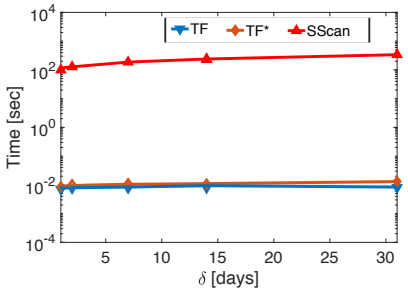


Figure 11: Running time vs. time distortion.

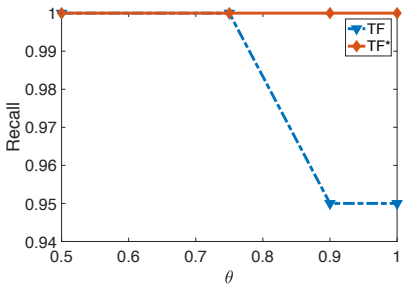


Figure 12: Recall vs. LCSS threshold.

Impact of the temporal distortion (δ). The temporal distortion parameter, δ, does not affect the utility of our approach for similarity search. Varying δ does not introduce additional false negatives. Regarding the running time, we observe a small increase as δ increases (see Figure 11), due to the enlarged search space. Note that the EHR data in MIMIC-III were collected in the ICU, and patients tend to have only a limited number of hospitalizations that, in general, last for only a few days (i.e., ≤ 10 days). Therefore, larger temporal distortions are unlikely to cause observable changes in the algorithm's behavior.

Impact of the LCSS similarity threshold (θ). Figure 12 reports the recall of our approach by increasing the LCSS similarity threshold θ. Our tandem search method TF* achieves perfect recall for all θ values, while the simple method TF achieves a slightly lower recall that fluctuates between 0.95 and 1. TF* search the tandem occurrences by extending each matched event set, the probability of missing a tandem is exponentially lower than that of TF, thus providing more robust utility. Regarding the running time, we show in Figure 13 that TF* incurs a small overhead compared to TF. In conclusion, the θ parameter does not significantly impact the running time for all methods.

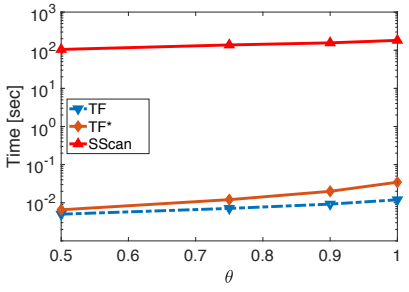


Figure 13: Running time vs. LCSS threshold.

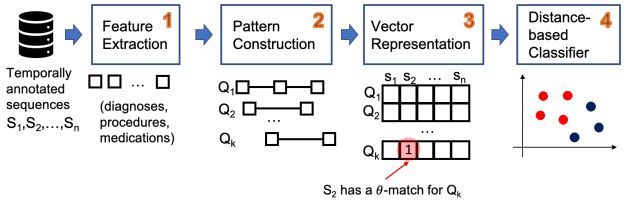


Figure 14: Overview of the case study for classifying IBD patients. The case study includes four steps: (1) feature extraction, (2) pattern construction, (3) vector representation, and (4) applying a distance-based classifier. The feature-based approach, following the steps (1), (3), and (4), maps the temporally annotated sequences into binary vectors according to the presence of every feature extracted in step (1). The pattern-based approach, following all the steps (1)-(4), maps the temporally annotated sequences into binary vectors according to the presence of a θ-match for each pattern constructed in step (2). As a proof of concept, KNN classifier is adopted in step (4).

5.3. *Case Study: Temporal Patterns for Inflammatory Bowel Disease*

We further study the applicability of similarity search in clinical predictive tasks for low prevalence diseases. Specifically, we consider the task of identifying patients with Inflammatory Bowel Disease (IBD) in MIMIC-III dataset. IBD broadly refers to chronic inflammation of the gastrointestinal tract [32]. Our goal is thus to study whether temporal patterns extracted from the data sequences could facilitate the identification of patients with IBD.

We address the binary classification task (i.e., IBD vs. non-IBD), by mapping patient data into vectors and applying a distance-based classifier in the vector space. As a proof of concept, we adopt the K-nearest neighbors (KNN) classifier with $K = 5$ and calculate the distance between patient vectors with Euclidean distance. To generate the vector for each patient, we consider two approaches. The first approach employs a set of features (i.e., medical events) and generates one bit for each feature, i.e., 1 if the patient's data contains the event and 0 otherwise. The second approach constructs temporal patterns using the same set of features and generates one bit for each temporal pattern, i.e., 1 if the patient's data contains a θ-match of the pattern and 0 otherwise. A detailed description of our methodology, such as feature extraction and temporal pattern construction, is reported in Appendix B.3.

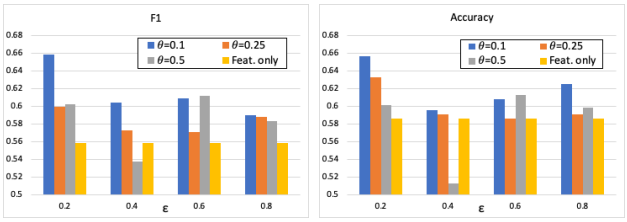
Figure 15 reports the classification results in F_1 score and accuracy, when features (i.e., medical events) and patterns (i.e., temporal subsequences of medical event sets) are adopted. Overall, we observe that the same classifier may achieve better results when adopting the temporal patterns, compared to adopting static features. This demonstrates that temporality among medical events is informative in distinguishing patients. Furthermore, we observe lower values of ϵ and θ ($\epsilon = 0.2$ and $\theta = 0.1$) produce higher accuracy, as they are less restrictive and thus retrieve similar patients despite the noise in the data. This shows that our similarity search framework can benefit applications based on EHR data, where variation among patients should be considered.

6. Discussion

Here, we discuss potential limitations and alternative strategies to improve our proposed methods.

Similarity between medical events. While our approach enables approximate search for EHR data by detecting query patterns even in the presence of noise (e.g., missing data and temporal distortion), it builds on standard similarity measures for event sets (i.e., Jaccard index and containment), which may not capture the similarity between medical events. In fact, different medical codes may be related to each other. For instance, diagnosis codes may overlap with each other (e.g., a higher level ICD9 code is a "parent" of those ICD9 codes in its subtree). A future direction is to incorporate such relations into patient similarity search. To represent the similarity between medications, a possible solution is to consider the drug's chemical structure to incorporate the drug-to-drug similarity. Zhang et al. [33] showed promising results in drug analytic tasks using such a drug-to-drug similarity model.

Constructing query patterns.. In our method, temporal patterns are used to identify patients with queried clinical conditions. However, formulating appropriate queries may be challenging when there is a lack of medical guidelines. As shown in the IBD case study, unsupervised approaches could be adopted to identify candidate patterns associated with a target condition. There are several ways to extract patterns. For example, temporally annotated sequences associated with a target condition can be clustered, and common subsequences can be identified to represent candidates' patterns [34, 35]. Alternatively, association rule based methods can be adopted to mine common patterns across multiple sequences [36, 37]. Those patterns can be obtained in a pre-processing step, after which clinicians and domain experts can incorporate their knowledge to construct effective queries.



(a) F_1 score (b) Accuracy

Figure 15: IBD classification with KNN classifier ($K = 5$): pattern-based vs. feature only. For the pattern-based classification, we constructed 40 temporal patterns and apply similarity search while varying values of ϵ and θ. We set $\delta = 5$ days for all experiments.

7. Conclusions

In this work, we presented a framework for similarity search in temporal medical data. Our framework takes as input query patterns that express target health conditions and searches large EHR datasets for patients whose temporal medical data is similar to the query pattern. The search criterion is designed with the LCSS similarity measure, which accounts for noise in medical events (e.g., diagnoses and medications) and in time (e.g., recorded event times). As a result, our framework provides robustness against noise in the EHR data. We develop a novel filtering approach to boost the efficiency of similarity search, which eliminates non-relevant data and results in no false negatives with high probability. Empirical evaluations demonstrate that our solution significantly accelerates the similarity search, compared to the start-of-the-art approaches, while providing highly accurate search results.

Acknowledgements

LB is supported in part by the National Institute of General Medical Sciences grant R01GM118609, and National Human Genome Research Institute grant K99HG010493. LF is supported in part by the National Science Foundation grant CNS-1951430 and CNS-1949217, and UNC Charlotte. XJ is CPRIT Scholar in Cancer Research (RR180012), and he was supported in part by Christopher Sarofim Family Professorship, UT Stars award, UTHealth startup, the National Institute of Health (NIH) under award number R01AG066749, R01GM118609, R01GM114612 and U01TR002062. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

Appendix A. Additional Material

Appendix A.1. Optimal Indexing

In the construction of our workload-aware index, our goal is to minimize the total error $Err^*_m(\epsilon^-, \epsilon^+)$. Given the range $[\epsilon^-, \epsilon^+]$, we divided it into h discrete values of ϵ. On this range of discrete values, the optimal partition can be found by considering all the possible m ranges and finding the best parameter values (b, r) for each range. This step can be computationally intense if performed naively. We observe that the optimal error $Err^*_m(\epsilon^-, \epsilon^+)$ with m indexes can be formulated in a iterative manner as follows:

$$\min \begin{cases} Err^*_{m-1}(\epsilon^-, \epsilon^+), \\ \min_{\epsilon^j \in [\epsilon^-, \epsilon^+]} \{Err^*_{m-1}(\epsilon^-, \epsilon^j) + Err^*_1(\epsilon^j, \epsilon^+)\} \end{cases}$$

We can use the dynamic programming strategy to solve this problem efficiently. In fact, there are at most h^2 non overlapping ranges in the interval $[\epsilon^-, \epsilon^+)$ and at most m indexes that need to constructed. Given a range $[\epsilon_i, \epsilon_j)$, we can assume a constant time for initialize the LSH index for $[\epsilon_i, \epsilon_j)$, for example by tuning (b, r) to minimize the error for $= (\epsilon_i + \epsilon_j)/2$. Then, the m indexes associated with the partition of $[\epsilon^-, \epsilon^+]$ that minimizes $Err^*_{m-1}(\epsilon^-, \epsilon^+)$ can be computed in $O(m \times h^2)$.

Appendix A.2. Searching Tandem Occurrences

Here, we analyze the time complexity and accuracy for searching tandem pattern occurrences. The single-seed expansion approach identifies tandem occurrences by searching forward in time after detecting the first component of a tandem, while multiple-seed expansion approach tests all sets in the temporal vicinity of a detected component, i.e., searching both forward and backward in time. Let $|Occ(P)|$ denote the maximum number of occurrences for any set P_i in the query pattern and Δ represent the maximum temporal interval on which the query spans (i.e., $\Delta = \Delta t_{1,k} + 2\delta$).

Time Complexity. The running time of single-seed expansion is $O(|Occ(P)| \times \Delta)$, since we do a constant number of similarity tests for each set occurrence. For multiple-seed expansion, each time a set occurrence is identified with the LSH index, we test if any tandem pattern occurs in its temporal vicinity. There are at most Δ^2 pairs of sets that need to be tested in the set's vicinity. Therefore, the overall running time is $O(|Occ(P)| \times \Delta^2)$.

Accuracy. Here, we analyze the probability of missing tandem occurrences (i.e., false negatives). For

the single-seed expansion method, anytime the LSH index fails to identify an occurrence of P_i, we miss the tandem occurrences having P_i as the first component. Therefore, a real θ-match will be missed as long as one of the tandem occurrence is missed; and the probability of missing a tandem is equals to the false negative probability of the LSH index, i.e., missing the occurrence of its first component. As a result, Pr[single-seed fails] $\geq p_{FN}$. For the multi-seed expansion approach, the LSH failure to detect P_i occurrence can be remedied by detecting any other set in the query pattern. In other words, a real θ-match is missed if only the LSH index fails to detect all sets in the query (e.g., when $\theta = 1$). Thus, Pr[multi-seed fails] $= p_{FN}^k$, which reduces the false negative rate for θ-matches exponentially compared to single-seed expansion.

Appendix A.3. Finding Temporally Ordered Tandem Occurrences

Here, we provide details on how our method keeps track of the identified tandem occurrences in order to match the query pattern. We first describe the construction of the acyclic graph G introduced in Section 4.2.1, and then provide further details on the estimation of the tandem coverage.

Algorithm 1 GraphConstrution

Require: $\mathcal{S}$ - temporally annotated sequence, P - pattern in Q, $\mathcal{T}$ - set of tandem occurrences

$L_i \leftarrow \{\phi\}$ for $i = 1, \ldots, |P|$
$G = (V = \phi, E = \phi)$ ▷ Empty graph
for $T_{ij} = \{(P_i, P_j), \Delta t_{ij,}\} \in T$ **do** ▷ Occurrences of the tandems in order of i in $\mathcal{S}$
    $v_i, v_j \leftarrow$ occurrences associated with P_i and P_i
    $L_i \leftarrow L_i \bigcup v_i$ ▷ Add a new node to layer L_i
    $L_j \leftarrow L_j \bigcup v_j$ ▷ Add a new node to layer L_i
    $V \leftarrow V \bigcup \{v_i, v_j\}$
    $E \leftarrow E \bigcup (v_i, v_j)$
end for
$G = (V, E)$

Graph Construction. Algorithm 1 constructs the multi-partite graph G in an iterative manner, by considering the set of tandem occurrences and the order of the tandems in the query Q. The main loop examines all the tandem occurrences in $\mathcal{T}$, and for each occurrence an edge is added to the graph. It is easy to see that the algorithm runs in $O(|\mathcal{T}|)$ time, and the final graph G has $|E| = |\mathcal{T}|$ edges and at most $|V| = 2 \times |\mathcal{T}|$ nodes.

Coverage Computation. Here, we formally define tandem coverage and illustrate how it relates to the order of the tandems.

Definition Appendix A.1 (Tandem Coverage)**.** *Let $e = (v_i, v_j)$ be an edge in G connecting v_i and v_j associated with the tandem $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$ in Q. Then, the coverage of e, denoted as $c(e)$ is the number of paths between v_i and v_j.*

Lemma Appendix A.1. *Let $e = (v_i, v_j)$ be an edge in G connecting v_i and v_j associated with the tandem $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$ in Q, with coverage $c(e) = m$. Then, all the tandems associated with the edges in the m paths preserves the temporal order of Q.*

Proof. (Sketch) We prove an intuitive proof for this theorem using induction on the number of paths covered by e.

- Base Case: We start with $m = 1$. In this case, we have the path formed by the edge e itself. Because, e is associated with $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$, for which $i < j$ by definition of tandems, we have that the base case holds.

- Inductive Step: We assume that the condition holds for all $m' < m$ and here we prove it for m. Let $\pi_z = \langle v_i, \ldots, v_z, \ldots, v_j \rangle$ be a path between layers L_i and L_j, where the node v_z is contained in the layer L_z, with $i < z < j$, which is part of the coverage of $e = (v_i, v_j)$. For such a path π_z, the subpaths $\pi'_z = \langle v_i, \ldots, v_z \rangle$ and $\pi''_z = \langle v_z, \ldots, v_j \rangle$ must have coverage $m' < m$, and therefore by induction the tandems in their coverage are in order. Since, the layers in G are constructed to preserve the order of the sets of Q, we have that $L_i < L_z < L_j$ resulting in the tandems on the two subpaths to be in order. In fact, the tandems associated with

the layers $L_i, \ldots, L_z$ precede those in the layers $L_z, \ldots, L_j$. Therefore, all the tandems in associated with the path $\pi_z = \langle v_i, \ldots, v_z, \ldots, v_j \rangle$ are in order. Since the edge $e = (v_i, v_j)$ is associated with $T_{ij} = \{(P_i, P_j), \Delta t_{ij}\}$, the largest tandems among those covered, we have that all the tandems covered by e are in order.

□

Appendix A.4. Filtering with Tandem Patterns

Here, we provide the accuracy and time complexity of our tandem filtering approach.

Lemma Appendix A.2. *Let t_i be a time index in a temporally annotated sequence representing a θ-match for a query pattern P with k sets P_1, P_2, ..., P_k. The following conditions hold:*

1. *Counting: There are at least $th_1 = \frac{\alpha \times (\alpha-1)}{2} - 1$ distinct tandems shared between the subsequence starting at time t_i and the input query pattern.*

2. *Coverage: In the directed acyclic graph $G = (V, E)$ there exists an edge $e = (v_i, v_j)$, such that the number of distinct paths between v_i and v_j is at least $2^\alpha/4$ (i.e., $c(e) \geq th_2 = 2^\alpha/4$).*

where $\alpha = \lfloor \theta \times k \rfloor$.

Proof. (1) As t_i is the time index for a θ-match, there exists a common subsequence between the query pattern P and the subsequence starting at time t_i of length at least $\alpha = \lfloor \theta \times k \rfloor$. This subsequence can be decomposed into $N_\theta = \frac{\alpha \times (\alpha-1)}{2} - 1$ distinct tandems occurring starting from time t_i. (2) Let $T_{ij} = \{(P_i, P_j), \Delta_{ij}\}$ be the longest tandem pattern among the retrieved tandems in the θ-match, and let L_i and L_j the layers in the directed acyclic graph G where the nodes representing the occurrences of P_i and P_j are placed, respectively. For this tandem, we notice that $j \leq i + \alpha$, as at least α sets must be retained in the θ-match, and among all the N_θ the edges of G representing the occurrences of tandems, there are edges spanning between 1, 2, ..., α layers between L_i and L_j. Therefore, the number of paths between L_i and L_j can be computed by considering the total number of combinations for layers within L_i and L_j. Formally,

$$\sum_{z=0}^{\alpha-2} \binom{\alpha-2}{z} = 2^{\alpha-2}$$

As a result, the edge $e = (v_i, v_j)$ associated with T_{ij} has coverage $c(e) = th_2 = 2^{\alpha-2}$, which proves (2). □

Time Complexity. Here, we analyze the running time for the filtering step, by evaluating the criteria in Lemma Appendix A.2. First, the time complexity for the counting criterion (1) is linear with the number of occurrences of the tandems, to verify a sufficient number of tandems are present in the candidate sequence. Second, the coverage criterion (2) requires tracking all the paths between pairs of nodes in G. To solve this problem, we can adapt standard recursive algorithms for graph explorations (e.g., DFS, BFS). Since this criterion evaluates whether there are at least th_2 paths between the start node and the end node of any edge, the running time for such a test is $O(|G| + th_2)$. The quantity $|G| = |V| + |E|$ denotes the size of the graph, where $|V| \leq 2|E|$ and $|E|$ is linear with the number of occurrences on the tandems, thus $|G|$ is linear with the number of tandems occurrences in the region tested. Notice, that th_2 may be exponential with the size of G, as pointed out in Lemma Appendix A.2.

Appendix B. Additional Experiments

Appendix B.1. Scalability Evaluation for Indexing

We evaluate the efficiency of indexing using two similarity measures for event sets: Jaccard index and containment. When Jaccard index is used, the set signatures and the individual LSH-indexes are constructed using LSH-MinHash [28] for Jaccard; for containment, the same is achieved with LSH-Ensemble [38]. Figure B.16 reports the overall running time for indexing when increasing the data dimensionality, i.e., the maximum number of events recorded at a time. We use suffixes `-J` and `-C` to

indicate the use of Jaccard and containment similarity measure. As can be seen, the overall running time increases as the dimensionality increases. In Figure B.17, we break down the overall indexing time in two parts: the time to compute the signatures from the medical event sets (`Sig-J/C`), and the time to optimize the index and to insert signatures (`IndexConst-J/C`), while increasing the size of the dataset to be indexed. We observe that the overall indexing time increases as the size of the data increases, but the data size affects each part of the process differently. For instance, the running time for `IndexConst-J` does not grow significantly when data size is larger, while the running time for `Sig-J` grows much faster. From both figures, we notice that the containment similarity measure has a higher computational cost compared to Jaccard. We believe that this is due to a higher overhead of the LSH-Ensemble technique in parameter optimization [38]. While indexing the medical events takes more time compared to the online similarity search, it is a one-time cost as indexing is performed offline; our solution can efficiently index EHR data for 50k patients in less than 600 seconds.

Appendix B.2. Experiments for Containment Similarity

Number of Matching Patients. Figure B.18 compares the number of matching patients obtained with Jaccard index (d_J) vs. that of containment similarity (d_C), while varying the value of threshold ϵ. As ϵ increases (i.e., stronger similarity), the number of matching patients obtained with both similarity measures decreases. We define the containment similarity between two sets X and Y as $d_C(X,Y) = \frac{|X \cap Y|}{|Y|}$, assuming Y is an event set in the query. Therefore, $d_C(X,Y) <= \frac{|X \cap Y|}{|X \cup Y|} = d_J(X,Y)$ for any non-empty sets X and Y. As a result, the number of matching patients obtained using d_C is higher than that of Jaccard d_J under the same similarity threshold ϵ.

Recall and Running Time. Figure B.19 reports the recall as the threshold for containment similarity increases. Similarly to the results obtained with the Jaccard index, our multi-seed expansion approach `TF`* has significantly higher recalls compared to the simple approach `TF`. Our filtering approach achieves a speed-up of several orders of magnitude compared to the baseline solution `SScan`, as shown in Figure B.20. We notice that when d_C is used, the running time tends to be higher than that of Jaccard d_J (in Figure 9), as containment similarity is likely to generate more matching patients given ϵ (see Figure B.18). From Figures B.21, B.22, B.23, and B.24, we conclude that the results are consistent with those achieved with Jaccard index.

Appendix B.3. Case study for classifying IBD patients

Data Preparation. In the MIMIC-III dataset, we identify 156 patients with IBD using the following codes of the 9th revision of the International Statistical Classification of Diseases (ICD9): 556, 556.0, 556.1, 556.2, 556.3, 556.4, 556.5, 556.6, 556.8, and

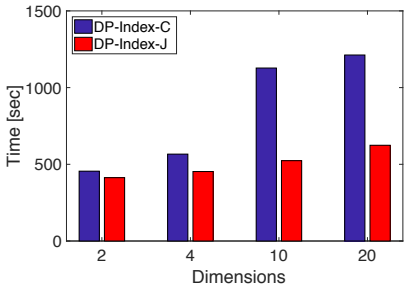


Figure B.16: Overall indexing time vs. dimensionality.

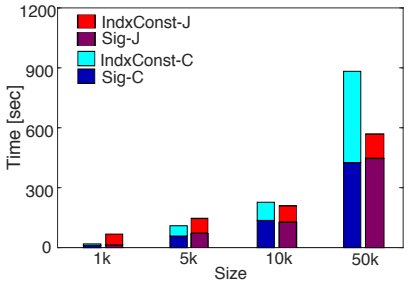


Figure B.17: Indexing time breakdown vs. dataset size.

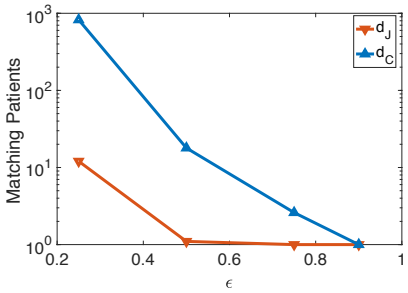


Figure B.18: Number of matching patients vs. similarity threshold ϵ: d_C - containment similarity, d_J - Jaccard index.

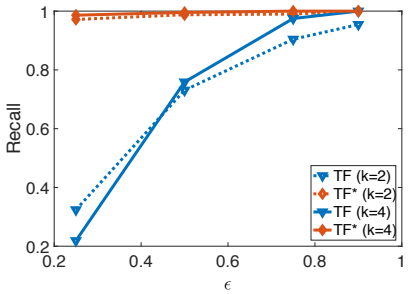


Figure B.19: Performance of similarity search with containment measure. k: length of the query pattern.

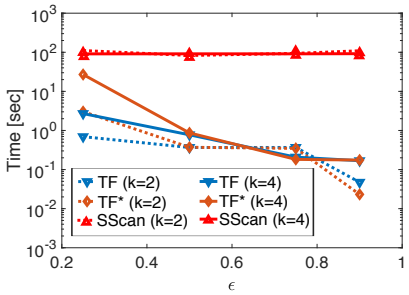


Figure B.20: Running time of similarity search with containment measure.

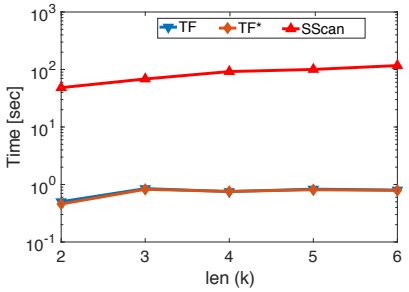


Figure B.21: Running time vs. query length with containment measure.

556.9. We construct the temporally annotated sequences for each patient, where each event set contains information about the medications, procedures, and diagnosis codes recorded at the given time. We randomly split the MIMIC-III patients to obtain a 80% training set and a 20% testing set. We balanced the test set by subsampling the non-IBD patients, i.e., the ratio of non-IBD to IBD is 1.

Features and Patterns Extraction. To construct a pattern-based classifier, clinical experts may supply known patterns directly. In this case study, we use a data-driven approach to identify reasonable patterns from the data. First, we adopt forward feature selection to extract features (i.e., medical events) for IBD patients, by iteratively adding the medical events that best improve the prediction. Subsequently, we combine these features into temporal patterns by sampling subsequences of the original patient data sequences where the extracted features occur. Table B.2 presents three patterns constructed, which model clinical events over time that are relevant to IBD [39, 40], such as bleeding in the gastrointestinal tract, infection of the urinary tract, respiratory diseases, complications in pregnancy.

Vector Generation and Classification. To perform the classification task (i.e., IBD vs. non-IBD), we use a K-nearest neighbors (KNN) classifier with

Query Patterns	Clinical Events
$Q = \langle\{272\}, \{V30.0\}\rangle, \Delta T = \langle 1\rangle$	Hypercholesterolemia and Single liveborn
$Q = \langle\{599, Furosemide\}, \{599, Furosemide\}\rangle, \Delta T = \langle 1\rangle$	Urinary tract infection and Furosemide for at least two consecutive measures.
$Q = \langle\{411.1, 272\}, \{578.9\}, \{486\}\rangle, \Delta T = \langle 1, 1\rangle$	Intermediate coronary syndrome and Hypercholesterolemia followed by Hemorrhage of gastrointestinal tract and Pneumonia

Table B.2: Three temporal patterns, out of 40 in total, used in the case study.

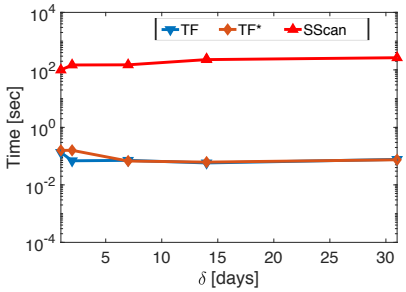


Figure B.22: Running time vs. time distortion with containment measure.

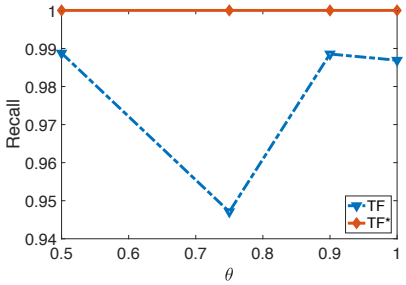


Figure B.23: Recall vs. LCSS threshold with containment measure.

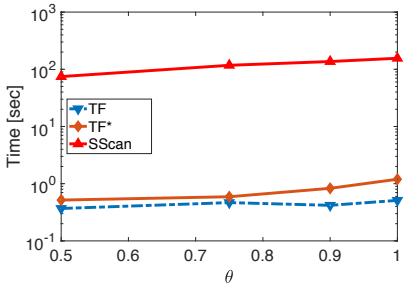


Figure B.24: Running time vs. LCSS threshold with containment measure.

$K = 5$, where patients are mapped into vectors and their distance is computed using Euclidean distance. In the vector generation process, we consider two approaches: a feature-based approach and a pattern-based approach. For the feature-based approach, each temporally annotated sequence is mapped into a binary vector, where the i-th bit is set to 1 if the i-th feature is present in the sequence and 0 otherwise. For the pattern-based approach, the i-th bit is set to 1 if there is a θ-match (with ϵ, δ, and containment similarity for LCSS) of the i-th pattern.

References

[1] N. J. Leeper, A. Bauer-Mehren, S. V. Iyer, P. LePendu, C. Olson, N. H. Shah, Practice-based evidence: profiling the safety of cilostazol by text-mining of clinical notes, PloS one 8 (5) (2013) e63499.

[2] A. B. Jensen, P. L. Moseley, T. I. Oprea, S. G. Ellesøe, R. Eriksson, H. Schmock, P. B. Jensen, L. J. Jensen, S. Brunak, Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients, Nature communications 5 (1) (2014) 1–10.

[3] D. A. Hanauer, N. Ramakrishnan, Modeling temporal relationships in large scale clinical associations, Journal of the American Medical Informatics Association 20 (2) (2013) 332–341.

[4] C. Xiao, E. Choi, J. Sun, Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review, Journal of the American Medical Informatics Association 25 (10) (2018) 1419–1428.

[5] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, J. Sun, Doctor ai: Predicting clinical events via recurrent neural networks, in: Machine Learning for Healthcare Conference, 2016, pp. 301–318.

[6] R. Miotto, L. Li, B. A. Kidd, J. T. Dudley, Deep patient: an unsupervised representation to predict the future of patients from the electronic health records, Scientific reports 6 (1) (2016) 1–10.

[7] Z. C. Lipton, D. C. Kale, C. Elkan, R. Wetzel, Learning to diagnose with lstm recurrent neural networks, arXiv preprint arXiv:1511.03677 (2015).

[8] B. M. Marlin, D. C. Kale, R. G. Khemani, R. C. Wetzel, Unsupervised pattern discovery in electronic health care data using probabilistic clustering models, in: Proceedings of the 2nd ACM SIGHIT international health informatics symposium, 2012, pp. 389–398.

[9] T. A. Lasko, J. C. Denny, M. A. Levy, Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data, PloS one 8 (6) (2013).

[10] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, M. Hauskrecht, Mining recent temporal patterns for event detection in multivariate time series data, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, pp. 280–288.

[11] F. Wang, N. Lee, J. Hu, J. Sun, S. Ebadollahi, A. F. Laine, A framework for mining signatures from event sequences and its applications in healthcare data, IEEE transactions on pattern analysis and machine intelligence 35 (2) (2013) 272–285.

[12] R. Moskovitch, Y. Shahar, Medical temporal-knowledge discovery via temporal abstraction, in: AMIA annual symposium proceedings, Vol. 2009, American Medical Informatics Association, 2009, p. 452.

[13] F. Altiparmak, H. Ferhatosmanoglu, S. Erdal, D. C. Trost, Information mining over heterogeneous and high-dimensional time-series data in clinical trials databases, IEEE Transactions on Information Technology in Biomedicine 10 (2) (2006) 254–263.

[14] R. Snodgrass, The temporal query language tquel, ACM Transactions on Database Systems (TODS) 12 (2) (1987) 247–298.

[15] R. T. Snodgrass, The TSQL2 temporal query language, Vol. 330, Springer Science & Business Media, 2012.

[16] C. Plaisant, S. Lam, B. Shneiderman, M. S. Smith, D. Roseman, G. Marchand, M. Gillam, C. Feied, J. Handler, H. Rappaport, Searching electronic health records for temporal patterns in patient histories: A case study with microsoft amalga, in: AMIA annual symposium proceedings, Vol. 2008, American Medical Informatics Association, 2008, p. 601.

[17] C. Combi, G. Pozzi, R. Rossato, Querying temporal clinical databases on granular trends, Journal of biomedical informatics 45 (2) (2012) 273–291.

[18] Y. Sha, J. Venugopalan, M. D. Wang, A novel temporal similarity measure for patients based on irregularly measured data in electronic health records, in: Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, ACM, 2016, pp. 337–344.

[19] L. Bonomi, X. Jiang, Patient ranking with temporally annotated data, Journal of Biomedical Informatics 78 (2018) 43 – 53. doi:https://doi.org/10.1016/j.jbi.2017.12.007.
URL http://www.sciencedirect.com/science/article/pii/S1532046417302770

[20] K. Seaton, A. Kharbanda, Evidence-based management of kawasaki disease in the emergency department., Pediatric emergency medicine practice 12 (1) (2015) 1–20.

[21] Z. Hajihashemi, M. Popescu, An early illness recognition framework using a temporal smith waterman algorithm and nlp, in: AMIA Annual Symposium Proceedings, Vol. 2013, American Medical Informatics Association, 2013, p. 548.

[22] A. Giannoula, A. Gutierrez-Sacristán, Á. Bravo, F. Sanz, L. I. Furlong, Identifying temporal patterns in patient disease trajectories using dynamic time warping: A population-based study, Scientific reports 8 (1) (2018) 1–14.

[23] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E. Keogh, Indexing multi-dimensional time-series with support for multiple distance measures, in: Proceedings of the Ninth ACM

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, ACM, New York, NY, USA, 2003, pp. 216–225. doi:10.1145/956750.956777.
URL http://doi.acm.org/10.1145/956750.956777

[24] J. F. Roddick, M. Spiliopoulou, A survey of temporal knowledge discovery paradigms and methods, IEEE Trans. on Knowl. and Data Eng. 14 (4) (2002) 750–767. doi:10.1109/TKDE.2002.1019212.
URL http://dx.doi.org/10.1109/TKDE.2002.1019212

[25] J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–843.

[26] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, Mining sequences with temporal annotations, in: Proceedings of the 2006 ACM symposium on Applied computing, ACM, 2006, pp. 593–597.

[27] X. Zeng, Z. Jia, Z. He, W. Chen, X. Lu, H. Duan, H. Li, Measure clinical drug–drug similarity using electronic medical records, International journal of medical informatics 124 (2019) 97–103.

[28] A. Broder, On the resemblance and containment of documents, in: Proceedings of the Compression and Complexity of Sequences 1997, SEQUENCES '97, IEEE Computer Society, Washington, DC, USA, 1997, pp. 21–.
URL http://dl.acm.org/citation.cfm?id=829502.830043

[29] 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, Sci Data 3 (2016).

[30] M. Bawa, T. Condie, P. Ganesan, Lsh forest: Self-tuning indexes for similarity search, in: Proceedings of the 14th International Conference on World Wide Web, WWW '05, ACM, New York, NY, USA, 2005, pp. 651–660. doi:10.1145/1060745.1060840.
URL http://doi.acm.org/10.1145/1060745.1060840

[31] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, pp. 518–529.
URL http://dl.acm.org/citation.cfm?id=645925.671516

[32] J. M. Dahlhamer, E. P. Zammitti, B. W. Ward, A. G. Wheaton, J. B. Croft, Prevalence of inflammatory bowel disease among adults aged 18 years—united states, 2015, Morbidity and mortality weekly report 65 (42) (2016) 1166–1169.

[33] P. Zhang, F. Wang, J. Hu, R. Sorrentino, Towards personalized medicine: leveraging patient similarity and drug similarity analytics, AMIA Summits on Translational Science Proceedings 2014 (2014) 132.

[34] D. Gotz, J. Sun, N. Cao, S. Ebadollahi, Visual cluster analysis in support of clinical decision intelligence, in: AMIA Annual Symposium Proceedings, Vol. 2011, American Medical Informatics Association, 2011, p. 481.

[35] F. Doshi-Velez, Y. Ge, I. Kohane, Comorbidity clusters in autism spectrum disorders: an electronic health record time-series analysis, Pediatrics 133 (1) (2014) e54–e63.

[36] H. Cao, M. Markatou, G. B. Melton, M. F. Chiang, G. Hripcsak, Mining a clinical data warehouse to discover disease-finding associations using co-occurrence statistics, in: AMIA Annual Symposium Proceedings, Vol. 2005, American Medical Informatics Association, 2005, p. 106.

[37] A. Wright, E. S. Chen, F. L. Maloney, An automated technique for identifying associations between medications, laboratory results and problems, Journal of biomedical informatics 43 (6) (2010) 891–901.

[38] E. Zhu, F. Nargesian, K. Q. Pu, R. J. Miller, Lsh ensemble: Internet-scale domain search, Proc. VLDB Endow. 9 (12) (2016) 1185–1196. doi:10.14778/2994509.2994534.
URL http://dx.doi.org/10.14778/2994509.2994534

[39] D. S. Pardi, E. V. Loftus Jr, W. J. Tremaine, W. J. Sandborn, G. L. Alexander, R. K. Balm, C. J. Gostout, Acute major gastrointestinal hemorrhage in inflammatory bowel disease, Gastrointestinal endoscopy 49 (2) (1999) 153–157.

[40] M. D. Long, C. Martin, R. S. Sandler, M. D. Kappelman, Increased risk of pneumonia among patients with inflammatory bowel disease, The American journal of gastroenterology 108 (2) (2013) 240.