Similar Patients Retrieval & Events Sequence Anomaly Detection System

Chengming Xie (cx2234), Hongshan Lin (hl3353)

*Topic : **Medical Insurance Fraud Detection** (Group 20)*

Github: https://github.com/steph2234/EECS6895---project

Presentation Slides:

https://docs.google.com/presentation/d/1LN_BDqEhdxfpA-u52lKkBJhX3WLurn3tqK01L4ukGyo/edit#slide=id.g777c31b5b8_0_162

Date: 5/14/2020

1. Abstract of our project

Aiming to reduce costs for either insurance companies or the government, our project tries to build a clinical retrieval system to detect anomaly cases among electronic medical records. Previous work has shown that machine learning for data-driven diagnosis has been actively studied in the medical industry to provide better healthcare. However, the challenges associated with characteristics of electronic clinical records may include: imbalanced-class massive data, irregularity in time, and sparsity hinder performance. To address this challenge, we deal with these medical events sequences based on natural language processing techniques At the final stage, we develop a visual analytics system to support comparative studies of patient records.Through its interactive interface, the user can quickly identify patients of interest and conveniently review both the temporal and multivariate aspects of the patient records.

2. Introduction

Healthcare is a major industry in the United States and it is important in the lives of many citizens, but unfortunately the continuingly rising in medicare care overall experienture and high costs of health-related services leave many patients with limited medical care. The impact of Medicare Fraud is estimated to be between 3% to 10 % among the nation's total healthcare spending, and reached approximately $2 billion including 165 medical professionals in 2018. In order to reduce costs for either insurance companies or the government, our project aims to build a clinical retrieval system to detect anomaly cases among electronic medical records. Our project also helps medical workers to get better decisions by identifying sequences that deviate

from the typically occurring behavior. For example, a doctor may be interested in finding patients whose postoperative response is different from other patients who have had the same surgery, so that the doctors can provide personalized care plans for similar patients in the future.

Previous work has shown that Machine learning for data-driven diagnosis has been actively studied in medicine to provide better healthcare and publicly available medicare claims data can be leveraged to construct models capable of automating fraud detection. However, the challenges associated with characteristics of electronic clinical records include: imbalanced-class big data, irregularity in time, and sparsity hinder performance. Effective use of event sequence data requires identifying sequences that deviate from the typically occurring behavior. Moreover, most prior work in event sequence anomaly detection focuses on detecting only one type of anomaly with a specific application domain. Such approaches do not provide flexible ways of identifying diverse types of anomalies, and provide only limited support for result interpretation.

To address the challenge,we try several different methods. We calculate similarities between medical records by dealing with these medical records as event and sequence embeddings. We first use Skip-gram to learn the vector representations of medical events codes and set the window size to learn the temporal relationship within the patient's record. To make a comparison of sequences with different lengths easier, we incorporate Dynamic time warping sequence based alignment (*DTW*), and distances are calculated using the events' vector representations. In the second approach, we employ Variational AutoEncoders (*VAE*) in order to project events sequences with different lengths to vectors of the same length. The model learns latent representations for patients' event sequences and detects anomalies that deviate from the overall distribution.

These two approaches are then followed by Local Outlier Factor (LOF) and other anomaly detection analysis separately. We evaluated models performances by comparing the outlier results with several different embedding approaches and their following analysis. At the final stage, we develop a visual analytics system to support comparative studies of patient records.Through its interactive interface, the user can quickly identify patients of interest and conveniently review both the temporal and multivariate aspects of the patient records.

3. Related Work

In *Visual Progression Analysis of Event Sequence Data* by Shunan Guo, Zhuochen Jin, David Gotz, Fan Du, Hongyuan Zha and Nan Cao, they proposed leveraging the techniques of segmenting a sentence into thematically consistent sub-phrases in text mining area to analyze the medical records of patients. They introduced an event embedding algorithm which converts each event into a vector derived from the context of surrounding events within a given group of sequences. The co-occurrence likelihood of events is also captured by the distance between their vectors. They used a skip-gram based three layer neural network to learn the latent representation of events formulary code.

On top of the embedding matrix learned by the neural network, they employed Dynamic Time Warping to align a group of patients' event sequences with variable lengths and event orders which addressed the challenges of different lengths of electronic health records.

In their next paper, *Visual Anomaly Detection in Event Sequence Data*, they proposed an unsupervised anomaly detection algorithm based on Variational AutoEncoder which learns latent representations for all sequences in the dataset and detects anomalies that deviate from the overall distribution. As they said in their paper, "the model can estimate an underlying normal progression for each given sequence represented as occurrence probabilities of events along the sequence progression. Events in violation of their occurrence probability (i.e., event occurrences with small occurrence probability, and absent events with large occurrence probability) are identified as abnormal."

In the paper, *Learning Representations from Healthcare Time Series Data for Unsupervised Anomaly Detection,* Joao Pereira reduced the dimensionality of their events sequence embedding results using PCA and t-SNE to visually help finding the sequences (ECG heartbeats in their case) which do not conform with the normal pattern. They also used Clustering Analysis and Wasserstein distance-based anomaly metrics to detect the anomaly among their investigated patients' medical records.

Based on these three major papers and other literature we found, we implemented our model on the MIMIC-III dataset which is described below.

4. Our Data & Data Preprocessing

In this project, we used detailed and comprehensive patients' data from MIMIC-III dataset which contains 26 tables and medical records for 41000+ critical care patients. MIMIC-III (Medical Information Mart for Intensive Care III) is a large, freely-available database consisting of identified health-related data associated with over forty thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. It includes demographics, vital signs, laboratory tests, medications, and more. Details are available on the MIMIC website: https://mimic.physionet.org/

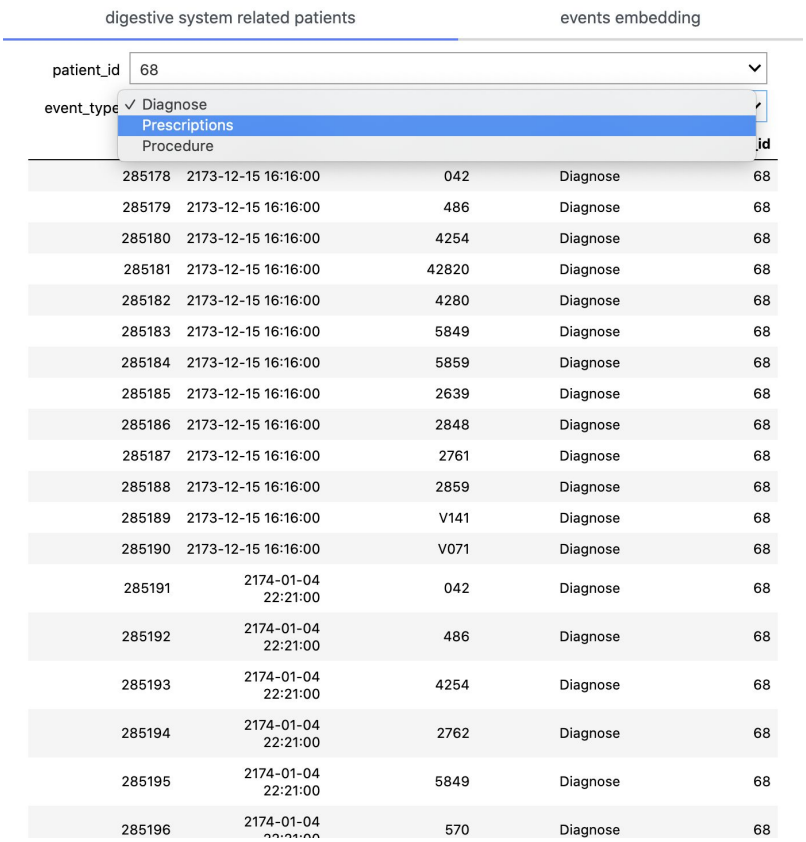
We extracted timestamped medical records for all patients which include their diagnoses, procedures and prescriptions. Some data preprocessing steps include dropping the missing data, merging relational tables together, converting the data format, and storing the cleaned data into json files.

```
In [20]: patient_dict = dict()
         for index in admissions.SUBJECT_ID.unique():
             patient = diag_df[diag_df['SUBJECT_ID'] == index][['ADMITTIME','ICD_9','SHORT_TITLE','SEQ_NUM']]
             patient_dict[index] = []
             for i in range(len(patient)):
                 stay = patient.iloc[i,:].to_dict()
                 stay['event_type'] = 'Diagnose'
                 patient_dict[index].append(stay)

In [27]: patient_dict[2]

Out[27]: [{'ADMITTIME': '2138-07-17 19:04:00',
           'ICD_9': 'V3001',
           'SEQ_NUM': 1.0,
           'SHORT_TITLE': 'Single lb in-hosp w cs',
           'event_type': 'Diagnose'},
          {'ADMITTIME': '2138-07-17 19:04:00',
           'ICD_9': 'V053',
           'SEQ_NUM': 2.0,
           'SHORT_TITLE': 'Need prphyl vc vrl hepat',
           'event_type': 'Diagnose'},
          {'ADMITTIME': '2138-07-17 19:04:00',
           'ICD_9': 'V290',
           'SEQ_NUM': 3.0,
           'SHORT_TITLE': 'NB obsrv suspct infect',
           'event_type': 'Diagnose'}]

In [21]: procedure_dict = dict()
         for index in admissions.SUBJECT_ID.unique():
             patient = procedure_df[procedure_df['SUBJECT_ID'] == index][['ADMITTIME','ICD_9','SHORT_TITLE','SEQ_NUM']]
             procedure_dict[index] = []
             for i in range(len(patient)):
                 stay = patient.iloc[i,:].to_dict()
                 stay['event_type'] = 'Procedure'
                 procedure_dict[index].append(stay)
```

Figure I. Cleaned data Overview

5. Learn vector representations of medical events formulary code

After data processing, we fetch the medical records of patients whose disease category falls into digestive systems, Circulatory & Respiratory systems related, injury & poisoning categories. We then train our models respectively in these three categories of patients' event sequence data.



285178	2173-12-15 16:16:00	042	Diagnose	68
285179	2173-12-15 16:16:00	486	Diagnose	68
285180	2173-12-15 16:16:00	4254	Diagnose	68
285181	2173-12-15 16:16:00	42820	Diagnose	68
285182	2173-12-15 16:16:00	4280	Diagnose	68
285183	2173-12-15 16:16:00	5849	Diagnose	68
285184	2173-12-15 16:16:00	5859	Diagnose	68
285185	2173-12-15 16:16:00	2639	Diagnose	68
285186	2173-12-15 16:16:00	2848	Diagnose	68
285187	2173-12-15 16:16:00	2761	Diagnose	68
285188	2173-12-15 16:16:00	2859	Diagnose	68
285189	2173-12-15 16:16:00	V141	Diagnose	68
285190	2173-12-15 16:16:00	V071	Diagnose	68
285191	2174-01-04 22:21:00	042	Diagnose	68
285192	2174-01-04 22:21:00	486	Diagnose	68
285193	2174-01-04 22:21:00	4254	Diagnose	68
285194	2174-01-04 22:21:00	2762	Diagnose	68
285195	2174-01-04 22:21:00	5849	Diagnose	68
285196	2174-01-04	570	Diagnose	68

Figure II. Medical Records Overview (screenshot of our interface)

To begin with, in order to learn the similarities between events formulary code (i.e. ICD-9 and Drug Code), we implement a Skip-gram based Word2Vec model to learn the vector representation of medical events.

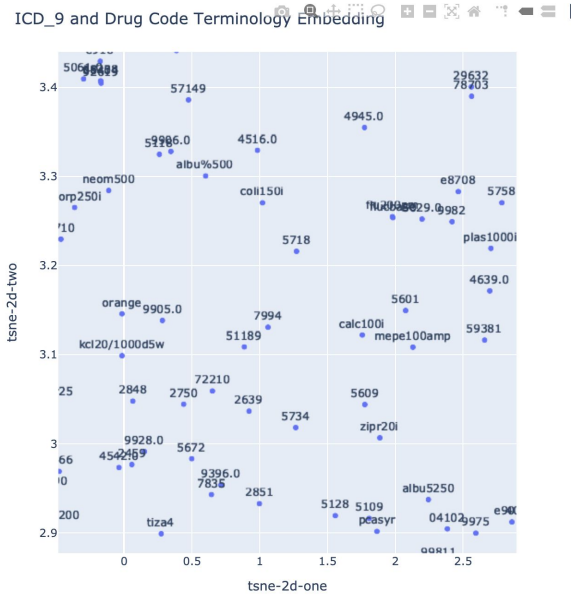
The training objective of our Word2Vec model is to learn event vector representations (of size 200) that aim to predict its context in our defined time window

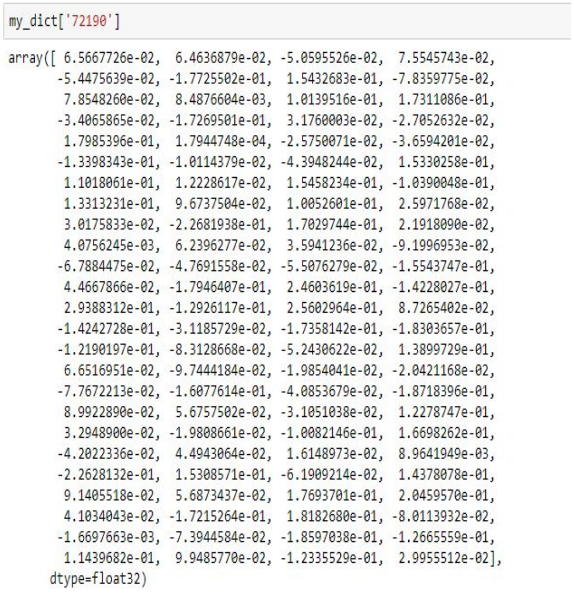
Mathematically, given a sequence of training events, $w_1, w_2, ..., w_n$ (e.g. all medical events of patients who have digestive system disease,) the objective is to maximize the average log-likelihood $1/T \{ \sum_{t=1}^{n} \sum_{j=-k}^{k} log\ p(w_{t+j} \mid w_t) \}$,where k is the size of the time window. In our

model implementation, we chose 7 days as time window (i.e. w_{t+j} is any medical event happening in seven days before or after w_t.)

The probability of correctly predicting event w_t given event w_j is determined by the softmax model, which is $p(w_{t+j} \mid w_t) = exp(u^T_{w_i} v_{w_j}) / \sum_{l=1}^{V} exp(u^T_{l} v_{w_j})$, where u_w and v_w is are vector representations of w as event and its context event respectively, and V is the vocabulary containing all events code in the event corpus.

Finally, we are able to generate the vector representations for all events codes in three categories respectively. In the Word2Vec model implementation, we leverag the power of Spark Clusters and used Spark MLlib to enhance the time efficiency of the model training.





```
my_dict['72190']
```

```
array([ 6.5667726e-02,  6.4636879e-02, -5.0595526e-02,  7.5545743e-02,
       -5.4475639e-02, -1.7725502e-01,  1.5432683e-01, -7.8359775e-02,
        7.8548260e-02,  8.4876604e-03,  1.0139516e-01,  1.7311086e-01,
       -3.4065865e-02, -1.7269501e-01,  3.1760003e-02, -2.7052632e-02,
        1.7985396e-01,  1.7944748e-04, -2.5750071e-02, -3.6594201e-02,
       -1.3398343e-01, -1.0114379e-02, -4.3948244e-02,  1.5330258e-01,
        1.1018061e-01,  1.2228617e-02,  1.5458234e-01, -1.0390048e-01,
        1.3313231e-01,  9.6737504e-02,  1.0052601e-01,  2.5971768e-02,
        3.0175833e-02, -2.2681938e-01,  1.7029744e-01,  2.1918090e-02,
        4.0756245e-03,  6.2396277e-02,  3.5941236e-02, -9.1996953e-02,
       -6.7884475e-02, -4.7691558e-02, -5.5076279e-02, -1.5543747e-01,
        4.4667866e-02, -1.7946407e-01,  2.4603619e-01, -1.4228027e-01,
        2.9388312e-01, -1.2926117e-01,  2.5602964e-01,  8.7265402e-02,
       -1.4242728e-01, -3.1185729e-02, -1.7358142e-01, -1.8303657e-01,
       -1.2190197e-01, -8.3128668e-02, -5.2430622e-02,  1.3899729e-01,
        6.6516951e-02, -9.7444184e-02, -1.9854041e-02, -2.0421168e-02,
       -7.7672213e-02, -1.6077614e-01, -4.0853679e-02, -1.8718396e-01,
        8.9922890e-02,  5.6757502e-02, -3.1051038e-02,  1.2278747e-01,
        3.2948900e-02, -1.9808661e-02, -1.0082146e-01,  1.6698262e-01,
       -4.2022336e-02,  4.4943064e-02,  1.6148973e-02,  8.9641949e-03,
       -2.2628132e-01,  1.5308571e-01, -6.1909214e-02,  1.4378078e-01,
        9.1405518e-02,  5.6873437e-02,  1.7693701e-01,  2.0459570e-01,
        4.1034043e-02, -1.7215264e-01,  1.8182680e-01, -8.0113932e-02,
       -1.6697663e-03, -7.3944584e-02, -1.8597038e-01, -1.2665559e-01,
        1.1439682e-01,  9.9485770e-02, -1.2335529e-01,  2.9955512e-02],
      dtype=float32)
```

Figure III: Screenshot from our web app (i.e. t-SNE visualization)

Figure IV: Embedding vector for Code '72190'

6. Events Sequence Anomaly Detection

One biggest challenge of this project is the development of an unsupervised anomaly detection that can efficiently handle the complex temporal structure of event sequences. The methods we use in our second approach (i.e., computing DTW pairwise distances of patients and

using traditional anomaly detection models such as LOF) make more sense intuitively but fail to capture complex temporal structures in the data.

The first approach is to use LSTM-based Variational AutoEncoder to extract sequential patterns from patients' event sequence data and project them into a latent space. Then we implemented Local Outlier Factor analysis to detect the anomalies using the latent representations of each patient's event sequence.

The second approach is to implement a Dynamic programming algorithm (Dynamic Time Warping) to align events sequence data with different lengths for each patient based on the cosine similarity between two medical events. We could then compute the dtw distance (accumulate cost of cosine similarity score along the optimized warping path) between two patients' events sequence data (i.e. achieved by DTW package.)

5.1. First Approach: LSTM-based Variational AutoEncoder:

The VAE encoder and the VAE decoder are using LSTM-based Recurrent neural networks to better extract sequential patterns from event sequence data. Our LSTM-based VAE model is trying to replicate the one in the paper, *Visual Anomaly Detection in Event Sequence Data, Shunan Guo, 2019*

The VAE encoder is trained to project the input sequence data, multi-hot encoding vectors of size (training_size, time_steps, vocab_size), into a latent feature vector that describes a sequential distribution of events occuring in the sequence. Each coordinate represents an event type and is marked 1 if such event type happens in a certain time step and 0 otherwise.

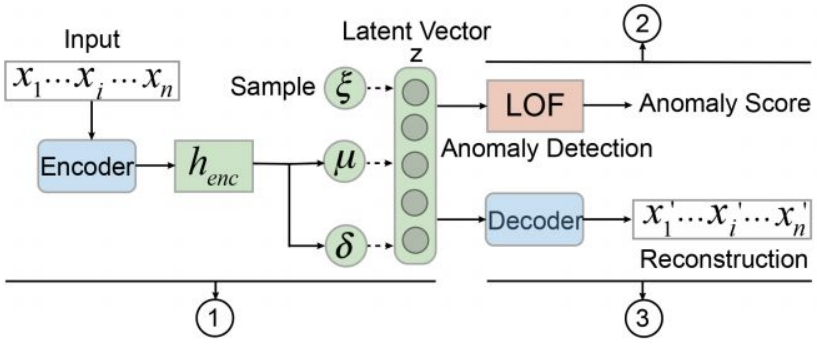
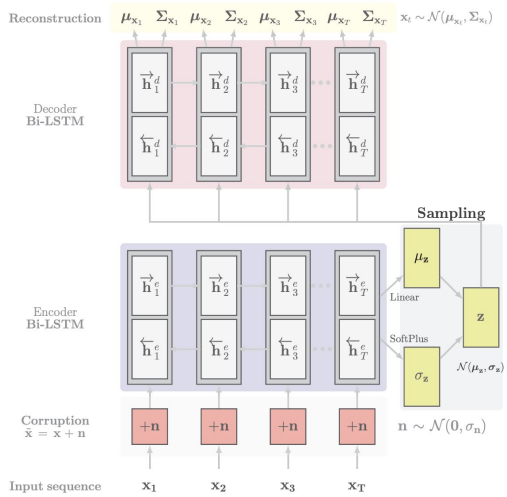


Figure V. Model Architectures



Suppose $h_{enc} = encoder(X)$ is the hidden state vector of the last layer of LSTM RNN. The vector is then projected into two vectors μ and σ to parameterize a normal distribution. When encoding, we want our latent space representation to be close to the true input (i.e., $q_\phi(z|x)$ and $P(z)$) where we can use Kullback- Leibler Divergence to measure the closeness of two distributions.

The *KL* loss we define is:

$$L_{KL} = -1/2 \, \Sigma^{M_z}_{i=m}(1 + \sigma^2 - exp(\sigma^2) - \mu^2)$$

Then we can draw a low-dimensional latent vector z by randomly sampling from the distribution. Z is sampled using a reparameterization trick:

$$z = \mu_\phi(x) + \varepsilon * \Sigma_\phi^{1/2}(x) \text{ where } \varepsilon \sim N(0,1)$$

In the decoder, we can reconstruct the input sequence from the latent feature vector z (i.e., z is fed to each layer of the RNN to estimate the probability distribution of events for each time slot).

Thus we use a softmax activation function which is then followed by a weighted cross entropy loss to calculate reconstruction loss. The reconstruction loss is defined as .

$$L_{reconstruction} = -1/n \, \Sigma^{n}_{i=1} \Sigma^{|E|}_{j=1}(w_{e^j} X_{i,j} log(x'_{i,j}) + (1 - x_{i,j})(log(1 - x'_{i,j})))$$

where $w_{e_j} = 1/log(n_j)$ is the weight to balance the prediction for unbalanced class of events in the training data and reduce the marginal importance of events with high occurrence.

The total loss is

$$L = L_{reconstruction} + w_{kl} * L_{kl}$$

where w_{kl} is the KL weight in the loss term which we set to 1 in our first training for now.

The parameter we set in our first training is roughly the same in the paper mentioned in the beginning of this section. We employ bidirectional LSTM with 300 hidden nodes in the encoding layer and LSTM with 300 hidden nodes as a decoding layer. We set the dimensions of the latent representation layer as 16 in the paper and optimize the loss function with the Adam optimizer with training data batch size of 100 for each training step.

Then we can use our encoding layer to project our training event sequence data into latent representations into a same dimensional space. We are then able to use the LOF Scikit-Learn function to calculate the "outlier" score for each patient. If the outputs of localOutlierFactor.fit_predict() function equals to -1, we can question the patient as an anomaly.

In our web application, we visualized the latent representation using PCA and t-SNE to allow users to have a better view of the results (visualizations included in System Overview.)

5.2. Second approach: DTW Distance based approach

After fetching all vector representations for all medical events of patients in a specific disease category, sequences are then aligned temporally using a dynamic programming algorithm (DTW) , and distances are calculated using the event vectors. The algorithm measures similarity between sequences by estimating the similarity between each pair of events respectively in these sequences based on the cosine/euclidean similarity of the corresponding event vectors.

```
distance, path = fastdtw(icd_2080,icd_4579,dist=euclidean)
print(distance)
print(path)

index_a,index_b=zip(*path)

plt.plot(index_a,index_b, color='k', linewidth=2)
plt.savefig('demo_dtw', bbox_inches='tight')
```

```
26.71888577368587
[(0, 0), (1, 1), (2, 1), (3, 2), (4, 3), (5, 4), (6, 5), (7, 6), (8, 7), (9, 8), (10, 9), (11, 10), (12, 11), (13, 11), (14, 1
1), (15, 11), (16, 11), (17, 12), (18, 13), (19, 14), (20, 15)]
```

Figure VI. We then compute the dtw distance (accumulate cost of cosine similarity score along the optimized warping path) between two patients' events sequence data.

Based on the DTW distance, we also create a similar patient retrieval system.

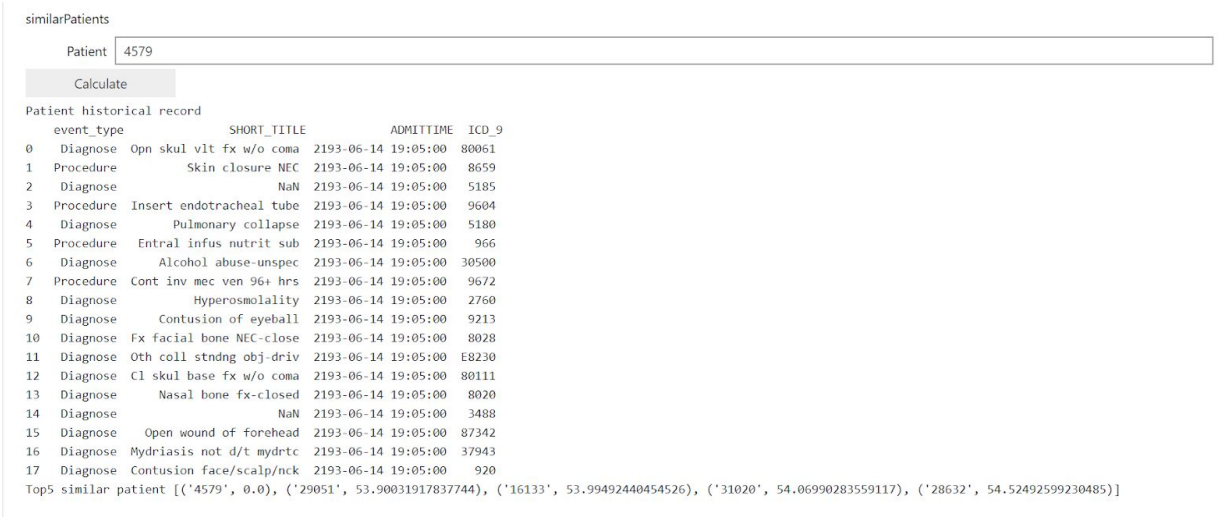


Figure VII: Screenshot from our web app (i.e. similar-patient-retrieval)

5.2.1 *Unsupervised Outlier Detection using Local Outlier Factor (LOF)*

The Local Outlier Factor (LOF) algorithm computes the local density deviation of a given data point with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood. More precisely, locality is given by k-nearest neighbors, whose distance is used to estimate the local density. By comparing the local density of a sample to the local densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. These are considered outliers.

5.2.1.1 Firstly, We implemented the LOF algorithm from scratch by computing reachability distance from each patient to his or her k-nearest neighbors.

$$reachability\ distance\ (a,b)\ = max\ \{k\ distance(b),\ dist(a,b)\}$$

where $k\ distance(b)$ is the distance of b to its k-th closest point

Then we computed local reachability density (lrd) for each patient a:

$$lrd(a)\ = 1/(\Sigma_{n \in N_k(a)}\ reachability\ distance(a,n)\ /\ k)$$

Then the LOF is basically the average ratio of the lrd of a's k nearest neighbours to the lrd of a.

5.2.1.2 Secondly, we use dtw statistics summary as features for each patient. After we set up our feature properly, we are able to implement anomaly detection models (ex, Local Outlier Factor (LOF), Isolation Forest).

We then visualize the results and check if the classification makes sense.We normalize and fit the metrics to a PCA to reduce the number of dimensions and then plot them in 2D highlighting the anomalies. We find both global outliers and local outliers in our graph.

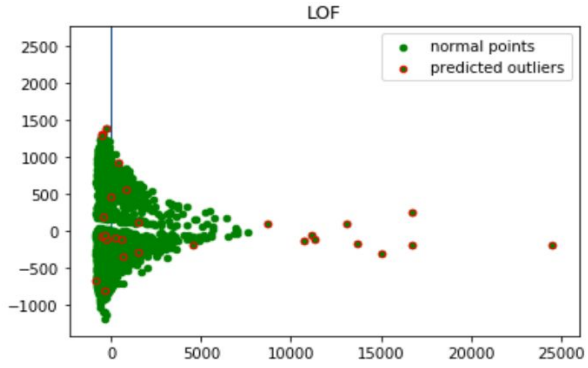


Figure VIII. Result of LOF Model in Injury & Poison related category. The red points in the left amont green cluster are considered local outlier

5.2.2 *Outlier Detection using Isolation Forest*

Another algorithm we used in DTW Distance Approach is Isolation Forest. The algorithm isolates each point in the data and splits them into outliers or inliers. Isolating an outlier means fewer loops than an inlier.

The Isolation Forest algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The logic argument goes: isolating anomaly observations is easier because only a few conditions are needed to separate those cases from the normal observations. On the other hand, isolating normal observations requires more conditions. Therefore, an anomaly score can be calculated as the number of conditions required to separate a given observation.

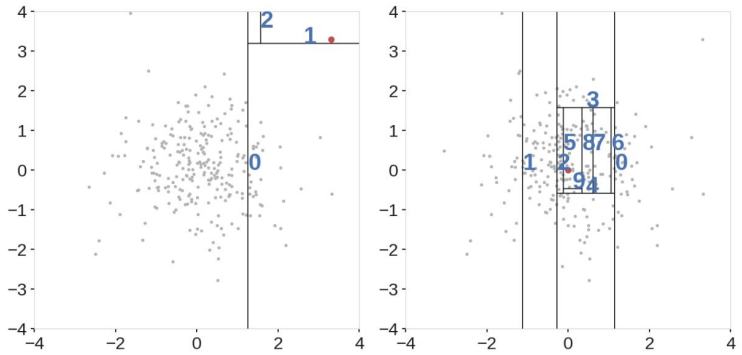


Figure IX. Concept of Isolation Forest (Inliers are harder to isolate than outliers.)

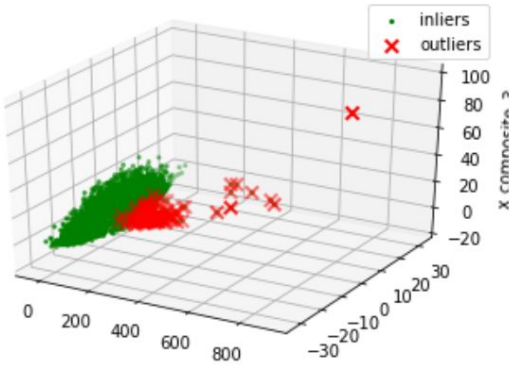


Figure X. Result of Isolation Forest Model in Injury & Poison related category. The red points in the left amont green cluster are considered local outlier

7. Experiments & Challenges

We met a lot of challenges during the project, and solved them by doing experiments and comparing with previous works.

Our ***first challenge*** is data cleaning. Due to the characteristics of medical records: high dimensionality, irregularity in time, and sparsity, we spend a lot of time dealing with these raw data. To solve this problem, it took time to understand the ER between tables and figure out a way to transform these 26 tables into a format in which selected features can be used for our models. We set the time window to learn the temporal relationship between medical records.

Moreover, although with the help of the GCP virtual machine, we faced a big challenge for model training during the skip-gram implementation. For instance, when creating one-hot encoding vectors for input layer in the skip-gram model, we are suffering from memory leak since we had over ***10,000,000+*** events pairs which were too many to generate one hot vectors for

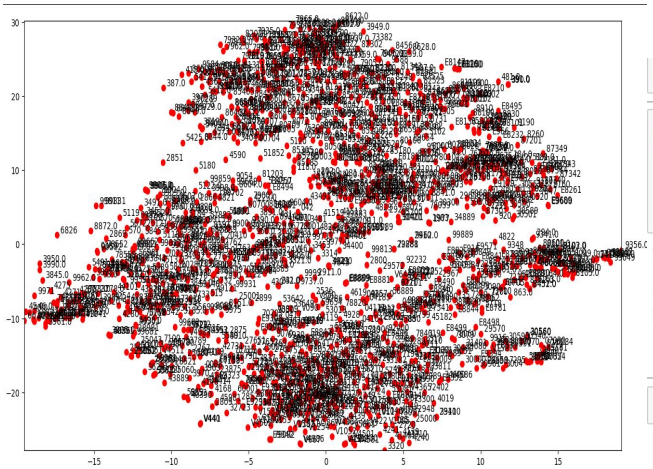
each of them; Also, when building neural network model, the model required a large amount of memory in our virtual machine. More specifically, we have tried 100+ GB extended memory VM but it still failed in the model training process and the kernel still consistently broke.

Thus, we leveraged the resilient distributed dataset (***RDD***) and ***Spark*** programming technique in GCP Dataproc ***Clusters.*** Using Scala-Spark allowed us to better scale the Word2Vec algorithms while enhancing the time efficiency by dozens of times. We could query vector representations for each medical event code. We can also order the most relevant events for each medical event by the cosine similarity of their vectors representations.

Figure XI: Json format of event embedding results (Vector Representation)

```
+-------+--------------------+
|   word|          similarity|
+-------+--------------------+
| topr50| 0.6053675413131714|
| hctz25| 0.5762994289398193|
|  42769| 0.5510246157646179|
|  49320| 0.5212512016296387|
| 4823.0| 0.5142900347709656|
| 4881.0| 0.5130040049552917|
|  71590|0.49900346994400024|
|   4589|0.49728673696517944|
|   4439|0.47791099548339844|
|   2720|0.47099921107292175|
|   2765|0.46940693259239197|
|   7048| 0.4664931297302246|
|   5180|0.46129298210144043|
|  v4582|0.46122118830680847|
| cil150| 0.4498251974582672|
| ator20|0.44093233346939087|
|    566| 0.4358844459056854|
|  53081| 0.4232785105705261|
| lev500|0.39566564559936523|
|gaba100|0.38527387380599976|
+-------+--------------------+
```

Figure XII. Spark rdd dataframe: Most relevant medical event to Aten25. (left)



```
import org.apache.spark.ml.feature.{Word2Vec, Word2VecModel}
```

```
val rdd = sc.textFile("gs://stephxie-6895/listfile4.txt")
val rdd2 = rdd.map(line => line.toLowerCase).
            map(_.split(" ")).
            map(Tuple1.apply).
            toDF("text")
```

```
Waiting for a Spark session to start...

rdd = gs://stephxie-6895/listfile4.txt MapPartitionsRDD[1] at textFile
rdd2 = [text: array<string>]

[text: array<string>]
```

```
rdd2.take(1)
```

```
[aten25, gaba100]
```

Figure XIV. mapping of 5000+ event code in latent space

Figure XV. screenshot of data input in scala kernel (GCP clusters)

The third challenge we meet is the wide differences of length in the event sequence data. After word embedding, we find our sequence records vary dramatically in length and exact event orders that nonetheless represent similar disease progressions. Multiple patients suffering from the same disease may have doctor visits at different times, they may have symptoms appear or recede at different speeds, and they may have distinct sets of comorbidities. It is necessary to align event sequences based on their implicit semantics.

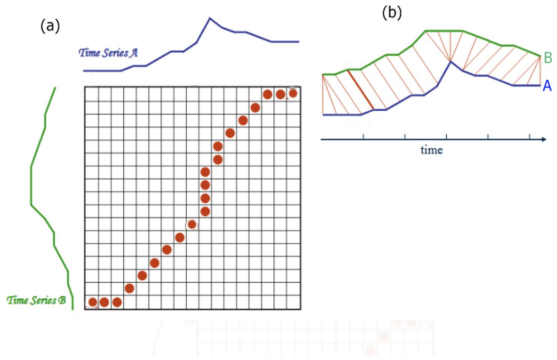


Figure XVI. DTW overview

To solve this problem, we implemented dynamic time warping algorithms to align each patient's event sequence data. We retrieved the similar patients based on their DTW distance (accumulated cost of the Path between two patients' event sequences).

The fourth challenge is the time-consuming DTW pairwise distance matrix computation in our second approach. We tried to employ our second approach on the injury & poisoning disease related patients. Since we have 2757 patients in this sub data set, we need to generate a 2757*2757 dimensions distance matrix. In other words, we have to calculate the DTW distance for each pair of patients, a $2757 * 2757$ times of DTW algorithm computation. To tackle the time complexity of our algorithm implementation, we used Spark DataFrame UDF function to calculate the pairwise distance matrix between patients while optimizing the running time efficiency using RDD technique.

Figure XVII. Spark Dataframe version of distance matrix

```
In [153]: df_pairs.withColumn('distance',udf_get_distance(df_pairs.vector1,df_pairs.vector2).cast(DoubleType())).first()

Out[153]: Row(id1='61619', vector1=['DOPABASE', 'BISA10R', 'BISA5', 'NACLFLUSH', 'SENN187', 'HEPPREMIX', 'HEPA5I', 'INFL0.5LF',
          'HEPBASE', 'D5W500E', 'PEPC20', 'NS100', 'DOBUBASE', 'DOCU100L', 'MIDA50I', 'AMIO150I', 'FENT2.5I', 'DOBPREM', 'DOPA4
          00D', 'D5W500E', 'AMIO150I', 'FENTSOLN50', '8856.0', '42741', '9671.0', '3961.0', '41091', '570', '78001', '3481', '2
          762', '2761', '5781', '2767', '4275', '41401', '4019', '49390', '4271', '78651', 'V1741', 'V707', '3612.0', '3615.0',
          '3749.0', '3722.0', '42731', 'INHRIV', 'NS100', 'D5W250', 'D5W250', 'D5W100', 'KCLBASE2', 'CORTIND', 'KCL20PM', 'D5W1
          000', 'KCLBASE2', 'D10W1000', 'D5W500E', 'CLOP300', 'CISA10I', 'D5W1000', 'CISA200I', 'LEVO4I', 'LEVO750PM', 'KCL20P
          M', 'CEFX1F', 'AZIT500I', 'KCLBASE2', 'BAG50', 'LEVOBASE3', 'NS100', 'MAG2PM', 'KCL20PM', 'INHRIV', 'GLUC1I', 'AMIO15
          0I', 'DEX50SY', 'METR500PM', 'VASO20I', 'HYDRIND', 'NS50', 'CHLO15L', 'METRBASE', 'D5W250', 'D5W100', 'FRBD50', 'ATOR
          80', 'INSULIN', 'CISA200I', 'D5W250', 'KAYE15L', 'KAYE15L', 'CISA10I', 'CALCG2/100NS', 'CLOP75', 'KCLBASE2', 'OXYC5',
          'TRAM50', 'POTA20', 'KCL20PM', 'MAG2PM', 'KCL20PM', 'KCLBASE2', 'BAG50', 'NS1000', 'DOCU100', 'METR500', 'METO25', 'A
          SA325', 'LIDO5T', 'NS1000', 'KETR30I', 'MABELI120', 'MORP5I', 'FENT2I', 'MORP5I', 'FENT2I', 'DOCU100', 'DOCU100', 'AC
          ET325', 'KCL20P', 'KCL20PM', 'BARI2/450L', 'OXYC5', 'MORP5I', 'IPRA2H', 'KCLBASE2', 'GUAI10', 'MORP5I', 'POTA20', 'AT
          RO1I', 'KCL20P', 'HEPBASE', 'KCL40I', 'HEPA5I', 'HEPPREMIX', 'NS451000', 'NS500', 'OXYC10', 'OXYC10', 'OXYC20', 'BAG5
          0', 'OXYC10', 'KCL20P', 'CAPT125', 'DOCU100', 'POTA20', 'SENN187', 'MAG2PM', 'OXYC5', 'OXYC10', 'MAG2PM', 'BAG50', 'M
          ETR500', 'MAG2PM', 'LORA1', 'BAG50', 'OXYC5', 'ACET650R', 'DOCU100', 'METO10I', 'MOM30L', 'PEPC20', 'ACET325', 'INFL
          0.5LF', 'LR1000', 'DOCU100L', 'CALCG2/100NS', 'KCL20PM', 'KETO15I', 'DEX50SY', 'BAG50', 'VANCOBASE', 'NTG100PM', 'GLY
          C1I', 'MAG2PM', 'MORP5I', 'D5W250', 'PERC', 'PHEN12SYR', 'PHEN10I', 'NS100', 'SYR60ML', 'D5W250', 'INHRIV', 'MORP2I',
          'VANC1F', 'NACLFLUSH', 'PROP200IG', 'PROP100IG', 'CHLO15L', 'NITR20SYR', 'SYR60ML', 'NEOSI', 'NS100CG', 'KCLBASE2',
          'NTGBASE', 'ASA81EC', 'HYDM4', 'HYDR2', 'LORA5', 'LISI10', 'LISI5', 'FURO40I', 'HYDR1SYR', 'HYDR2I', 'ASA81', 'METO2
          5', 'LORA2I', 'ALBU3H', 'SIMV40', 'LISI5', 'NACLFLUSH', 'NACLFLUSH', 'BISA5', 'METO25', 'ALBU3H', 'LORA5', 'BISA10R',
          'METO50', 'IBUP400', 'POTA20', 'HYDR2', 'LISI5', 'METO50', 'FURO20', 'METO25', 'PERC', 'POTA20', 'FURO20', 'ALBU25',
          'LISI5'], id2='8091', vector2=['LACT30L', 'PANT40', 'FOLI1', 'MVI', 'NACLFLUSH', 'D5NS1000', 'D545NS1000', 'THIA100',
          '5491.0', '5491.0', '5491.0', '5491.0', '5491.0', '5491.0', '4513.0', '5711', '6930', '570', '5712', '30390', '5845',
          'E9479', '5722', '2867', '966.0', '5990', '78605', '78791', '311', '45621', '7895', '11284', 'ALBU25100I', 'D5NS100
          0', 'CEFX2F', 'FRBD50', 'AMP', 'OCTR100I', 'ALBU25100I', 'MIDO25', 'CEFX2F', 'SARNL', 'RIFA200', 'PENT400', 'BACI15
          0', 'FRBD50', 'ALBU5500', 'NS500', 'PHYT10I', 'CEFX1F', 'FRBD50', 'MIDO5', 'SYRG1', 'OCTR500S', 'NACLFLUSH', 'NS500',
          'NS500', 'ALBU25', 'LACT30L', 'ALBU25100I', 'SARNL', 'NS1000', 'PROC5', 'COMP10I', 'LACT30L', 'HYDX25', 'DOLA12.5I',
          'MICROK10', 'TRAZ50', 'FLUC50', 'NYST5L', 'MICROK10', 'NEUT', 'D5NS1000', 'MICROK10', 'AMP2I', 'MIDO25', 'NEUT', 'FRB
```

The fifth challenge is high dimensionality of our dtw distance matrix, as you can see above, the shape of this matrix is 2754*2754 (num of patients * num of patients). Methods like LOF perform poorly in high dimensional data. To address this problem, we found not all the pairwise distances are significant and necessary in model training, so we use statistical summary as our features.
For instance, we use aggregation functions to obtain the quantile distance of each patient and successfully reduce the number dimension from 2754 to 5 while retaining most of the information. By comparing the result, we find the results are almost identical but we significantly reduce the running time. However, by looking at the graph, we find that there are several red points among green clusters and we call them local outliers. We realized that we can not simply use dtw distance as lof feature because when k is larger it possible to miss local outliers (Here K=100).

Figure XVIII Output before (left) and after processing (right)

In [358]: testdf

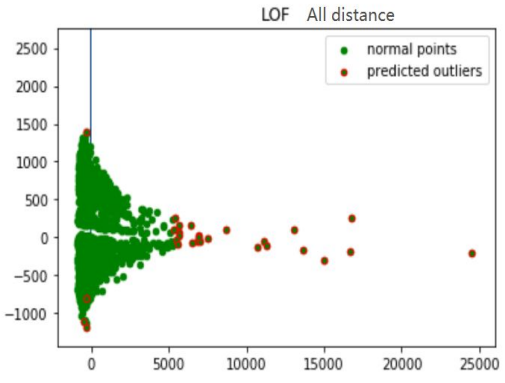
Out[358]:

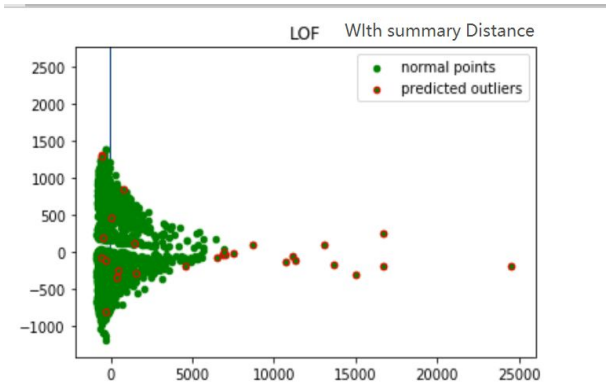
	0	1	2	3	4	5	6	7	8	9	...	2
0	0.0	93.110786	95.209916	94.846384	107.500044	228.660646	102.619785	103.290601	87.434186	99.951781	...	101.480
1	0.0	93.110786	37.327599	30.903466	33.218293	218.410445	39.685876	38.940109	56.910806	35.912058	...	32.090
2	0.0	95.209916	37.327599	31.255304	40.602756	243.834348	42.725788	34.264572	62.300871	32.881849	...	29.029
3	0.0	94.846384	30.903466	31.255304	31.034436	243.101101	39.271020	32.745783	59.783220	27.218671	...	20.923
4	0.0	107.500044	33.218293	40.602756	31.034436	283.611162	41.524206	34.965452	62.858571	33.753754	...	28.536
...	...	...	...	...	...	...	...	...	...	...	...	
2752	0.0	95.691961	28.930104	29.054032	20.102834	23.165428	273.896620	35.559869	28.235853	54.141932	...	30.165
2753	0.0	109.733824	37.084749	30.516431	21.267238	24.052903	294.687772	41.811129	26.001491	55.119238	...	32.353
2754	0.0	93.601122	39.338953	39.477768	23.360683	26.530125	250.865173	41.013648	34.828136	57.128658	...	29.152
2755	0.0	84.722922	30.313778	34.910726	25.307711	23.424782	228.226537	39.254284	32.052015	49.306518	...	27.189
2756	0.0	86.561382	33.056952	35.372779	30.504887	33.937598	227.906881	39.181283	36.278087	61.023270	...	33.167

2757 rows × 2757 columns

	min	max	median	75%	25%
0	82.344353	500.299721	98.698648	106.847841	92.145006
1	26.619805	494.641048	36.503448	41.474254	33.055250
2	25.041729	603.009126	35.131096	41.423022	30.777018
3	17.356616	539.792412	24.742618	35.420074	21.136093
4	21.869634	570.306704	33.215215	40.190302	28.609990
...	...	...	...	...	...
2752	6.670616	558.446509	20.908455	34.147660	12.984168
2753	8.638858	373.311880	22.250147	36.259703	14.310639
2754	14.560246	536.329782	24.223739	36.426581	19.271983
2755	14.205390	538.702339	25.162154	35.750634	20.663875
2756	25.273823	556.150950	32.377097	38.107313	29.576483

2757 rows × 5 columns





One biggest challenge of this project is the development of an unsupervised anomaly detection that can efficiently handle the complex temporal structure of event sequences which we have described before.

Under such circumstances, there are several recent papers regarding time series anomaly detection using LSTM-based VAE (eg. *Sequential VAE-LSTM for Anomaly Detection on Time Series, Run-Qing Chen,2019; Visual Anomaly Detection in Event Sequence Data, Shunan Guo,2019* , etc.) As Shunan Guo stated in their paper, "VAE uses a probabilistic encoder for modeling the distribution of the latent variables...such probabilities give more principled criteria for identifying anomalies and do note require model-specific thresholds. As a result, VAE can better facilitate objective judgements for deciding the boundary of anomalous sequences compared to other unsupervised algorithms." Thus, based on the papers, we are trying to implement LSTM-based VAE on our patients' event sequence data as our first approach to implement anomaly detection.

8. System Overview

7.1 Digestive system & Respiratory system

In Page 1 and Page 2 of our dashboard, we include detailed timestamped patients' medical records which can be queried by their unique patient id and event type (i.e. Diagnosis, Prescriptions and Procedures). The first column has two tabs which include patients' medical records as well as the events code embedding results (visualized using t-SNE) in their respective category. The right column includes the visualization after PCA and t-SNE dimensionality reduction upon LSTM-based VAE latent space representations. The anomalies are labeled as red points and users can fetch the anomaly patients id when hovering on the data point.

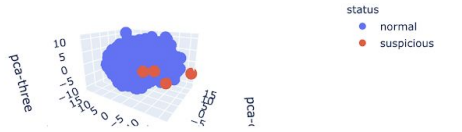
digestive system related patients | events embedding

patient_id: 68

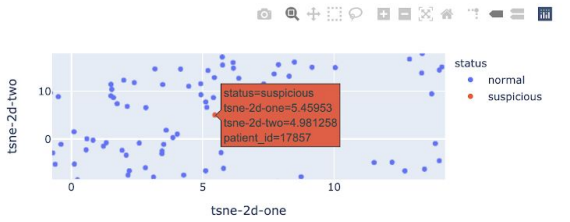
event_type: Diagnose

	ADMITTIME	ICD_9	event_type	patient_id
285178	2173-12-15 16:16:00	042	Diagnose	68
285179	2173-12-15 16:16:00	486	Diagnose	68
285180	2173-12-15 16:16:00	4254	Diagnose	68
285181	2173-12-15 16:16:00	42820	Diagnose	68
285182	2173-12-15 16:16:00	4280	Diagnose	68
285183	2173-12-15 16:16:00	5849	Diagnose	68
285184	2173-12-15 16:16:00	5859	Diagnose	68
285185	2173-12-15 16:16:00	2639	Diagnose	68
285186	2173-12-15 16:16:00	2848	Diagnose	68
285187	2173-12-15 16:16:00	2761	Diagnose	68
285188	2173-12-15 16:16:00	2859	Diagnose	68
285189	2173-12-15 16:16:00	V141	Diagnose	68
285190	2173-12-15 16:16:00	V071	Diagnose	68
285191	2174-01-04 22:21:00	042	Diagnose	68
285192	2174-01-04 22:21:00	486	Diagnose	68
285193	2174-01-04 22:21:00	4254	Diagnose	68
285194	2174-01-04 22:21:00	2762	Diagnose	68
285195	2174-01-04	5849	Diagnose	68

PCA

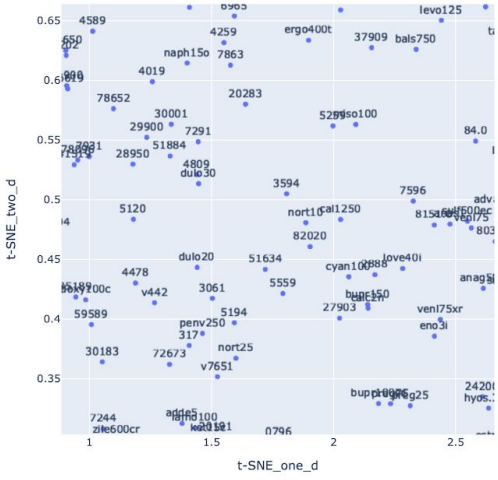


T-SNE



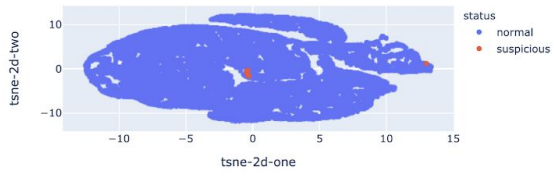
Circulatory/respiratory system related | icd-9 embedding

ICD_9 and Drug Code Terminology Embedding



PCA

T-SNE



7.2 Similar patient retrieval

In the similar patient retrieval page, we include Top 5 most similar patient retrieval and we also plot the DTW alignment for pairs of patients.
The section above shows the top 5 similar patients for a particular patient based on their DTW distance. The section below plots the DTW alignment graph for pairs of patients and shows how their sequence matches together.

7.3 Similar ICD-9 Retrieval

In this page, based on the result of word2vec word embedding, we plot the distance map of ICD-9 code to help users easily find correlated icd-9 code from patients' electronic records. This can also help physicians and users to make predictions and better decisions about future treatments.

9. Conclusion and Future Work

Analyzing similar patients' medical records to detect potential anomaly cases will help reduce costs for either insurance companies or the government. It also helps doctors in clinical decision making. Our work provides two embedding approaches and combines them with different unsupervised learning methods for measuring the similarity of patients and detecting anomaly cases, which can deal with high-dimensionality, irregularity, and sparsity in medical records. The visual analytics system built on top of our methods enables effective analysis of medical records from both temporal and multivariate perspectives. In the future, we will add more features and design a more comprehensive and user friendly system. Moreover, we will focus on the interpretation of these anomaly cases by delving into the case studies .

Reference:

S. Guo, Z. Jin, Q. Chen, D. Gotz, H. Zha and N. Cao, *Visual Anomaly Detection in Event Sequence Data*, 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 1125-1130.

Joao Pereira, *Learning Representations from Time Series Data for Unsupervised Anomaly Detection* 2019

S. Guo, et al., *Visual Progression Analysis of Event Sequence Data in IEEE Transactions on Visualization & Computer Graphics*, vol. 25, no. 01, pp. 417-426, 2019. doi: 10.1109/TVCG.2018.2864885.

Graphen AI.

Guo, Rongchen & Fujiwara, Takanori & Li, Yiran & Lima, Kelly & Sen, Soman & Tran, Nam & Ma, Kwan-Liu. (2020). *Comparative visual analytics for assessing medical records with sequence embedding*. Visual Informatics. 10.1016/j.visinf.2020.04.001.

Anne Fischer, *Generating Text From latent Space: creating novel news headlines from latent space using a Variational AutoEncoder*, 2019

Rasim Alguliyev, Ramiz Aliguliyev, Lyudmila Sukhostat, *Anomaly detection in Big data based on clustering,* Stat., Optim. Inf. Comput., Vol. 5, December 2017

MIMIC-III, a freely accessible critical care database. Johnson AEW, Pollard TJ, Shen L, Lehman L, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, and Mark RG. Scientific Data (2016). DOI: 10.1038/sdata.2016.35. Available from: http://www.nature.com/articles/sdata201635