KNOWLEDGE GRAPH REPRESENTATION OF ELECTRONIC HEALTH RECORDS FOR CLINICAL PREDICTIONS

by
EGE ALPAY

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
June 2022

EGE ALPAY 2022 ©

All Rights Reserved

ABSTRACT

KNOWLEDGE GRAPH REPRESENTATION OF ELECTRONIC HEALTH RECORDS FOR CLINICAL PREDICTIONS

EGE ALPAY

COMPUTER SCIENCE AND ENGINEERING M.Sc. THESIS, JUNE 2022

Thesis Supervisor: Asst. Prof. Öznur Taştan Okan

Keywords: Electronic Health Records, Knowledge Graph Embeddings, MIMIC-III, Machine Learning

In many countries, key clinical and administrative data of the patients are now systematically collected, recorded, and stored in digital formats. These patient-specific medical data are referred to as *electronic health records* (EHR). EHR data are rich; they capture patient-health care provider interaction at many encounters over time. This systematic digital collection of medical data presents a significant opportunity for developing data-driven technologies for transforming healthcare. Especially for high-stake situations with high uncertainty, such as in intensive care units (ICUs), these systems have the potential to reduce medical errors by assisting health care providers throughout their decision-making process.

While EHRs have the potential to bring solutions to diverse problems in the healthcare ecosystem, their use direct in predictive models is not trivial. Among many properties that yield technical challenges in machine learning systems, we address its sparse and heterogeneous nature. In this study, we propose a strategy where one can unify the heterogeneous data types in a knowledge graph framework and learn a dense patient representation that encodes meaningful information from patient EHRs. Our framework employs widely adapted knowledge graph embedding methods and deploys them in different ICU prediction tasks. These tasks comprise mortality prediction and binarized length of stay prediction tasks. We augment the learned patient representation from the knowledge graphs with lab measurements and vital signs. Compared to a state-of-the-art model, the proposed representation achieves superior performance in three of the four different classification tasks.

ÖZET

ELEKTRONİK SAĞLIK KAYITLARINI TEMEL ALAN BİLGE ÇİZGE TEMSİLLERİNİN KLİNİK TAHMİNLERDE KULLANIMI

EGE ALPAY

BİLGİSAYAR BİLİMİ VE MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, HAZİRAN 2022

Tez Danışmanı: Dr. Öğr. Üyesi Öznur Taştan Okan

Anahtar Kelimeler: Elektronik Sağlık Kayıtları, Bilgi Çizge Gösterilimi, MIMIC-III, Makine Öğrenimi

Birçok ülkede, hastaların temel klinik ve idari verileri artık sistematik olarak toplanıyor, kaydediliyor ve dijital formatlarda saklanıyor. Hastaya özel bu tıbbi verilere *elektronik sağlık kayıtları* (ESK) adı verilir. ESK'ler birçok farklı veri tip içerir ve hastanın sağlık sistemi ile sağlık hizmeti sağlayıcısı ile karşılaşmalarındaki etkileşimi yakalar. Tıbbi verilerin sistematik ve dijital olarak toplanması, sağlık hizmetlerini geliştirmeyi hedefleyen, veriye dayalı teknolojiler için önemli bir fırsat sunuyor. Özellikle yoğun bakım üniteleri gibi yüksek belirsizlik ve yüksek risk içeren durumlarda bu sistemler, sağlık hizmeti sağlayıcılarına karar verme süreçlerinde yardımcı olarak tıbbi hataları azaltma potansiyeline sahiptir.

ESK'ler sağlık alanındaki çeşitli sorunlara çözüm getirme potansiyeline sahip olsa da doğrudan tahmin modellerinde kullanılamazlar. Bu tezde, ESKların makine öğrenim sistemlerinde zorluğa sebep olan özellikleri arasında, seyrek ve heterojen yapısı ile ilgili problemi aşmaya çalıştık. Bu çalışmada, yoğun bakımdaki hastaların farklı türdeki verilerini bilgi çizgesi olarak temsil edip, çizge üzerinde hastaların yoğun gösterimlerini öğrenen bir yöntem sunuyoruz. Sunduğumuz yöntem, sıkça kullanılan bilgi çizge gösterilim öğrenme yöntemlerini kullandık ve öğrenilen gösterimleri farklı yoğun bakım ünitesinde gerçekleşen tahmin görevlerinde kullandık. Bu görevler, yoğun bakım ünitelerinde ölüm tahmini ve kalış süresi tahminini içerdi. Bilgi çizgelerinden öğrenilen hasta gösterimleri, laboratuvar ölçümleri ve yaşamsal

belirtileri gösterir veriler ile entegre ettik. Bu alanda örnek gösterilen bir çalışma ile karşılaştırıldığında, önerilen yöntem, dört farklı sınıflandırma görevinin üçünde üstün performans gösterir.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor Öznur Taştan Okan for continuous support and patience during my master's study. I am very grateful for her insightful comments and suggestions to improve this thesis.

Also, I would like to thank my family Nurhayat Alpay, Oğuz Alpay, Duygu Ece Demirçelik and Arda Demirçelik, for believing in me during my master's study. Without their belief, it would be impossible for me to complete my study.

"There are no hopeless situations.
There are hopeless people.
I have never lost hope."
Mustafa Kemal ATATÜRK

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATONS

CNN Convolutional Neural Network ..... ix, 10, 14, 16

CRF Conditional Random Field 5

EHR Electronic Health Record . iv, xi, 1, 2, 3, 4, 5, 6, 9, 10, 28, 29, 30, 31, 39, 66

GNN Graph Neural Network 9, 67

GRU Gated Recurrent Unit 4

ICU Intensive Care Unit iv, x, 1, 2, 5, 9, 11, 16, 31, 32, 33, 37, 39, 41, 43, 44, 49, 50, 51, 52, 54, 56, 57, 58, 60, 61, 62, 64, 65, 66, 67

IDF Inverse Document Frequency 7

LSTM Long Short Term Memory ... ix, xii, 4, 6, 7, 8, 9, 11, 17, 18, 19, 62, 63, 64

RNN Recurrent Neural Network ..... ix, 4, 5, 11, 16, 17, 19

1. INTRODUCTION

1.1 Motivation

Technological developments have facilitated the digitization of health data. In many countries, key clinical and administrative data of the patients are systematically recorded, collected and stored in digital formats (Bonomi, 2016), (McLoughlin et al., 2017). These patient-specific medical data is referred as EHR. EHR data are rich; they capture and integrate information collected at several instances of patient-health care provider interaction over time. EHRs typically contain demographic information, accounts of hospital visits, the diagnostic tests administered during these visits, applied procedures, diagnosed diseases and conditions, prescribed medications and detailed laboratory measurements or vital signs monitored at hospital stays. Depending on the EHRs system these contents may vary.

This systematic digital collection of medical data presents a significant opportunity for developing data-driven technologies that can transform the healthcare systems and the clinical practice (Madsen, 2014), (AC06953431, 2008). Optimizing hospital resources to reduce costs and improve care (Ferrão et al., 2021), (Jensen et al., 2012), (Hillestad et al., 2005), phenotype classification (Harutyunyan et al., 2019). Such data can also expedite the development of systems that can assist the health-care providers at the point-of-care such as treatment and medicine recommendation systems (Oh et al., 2021). Especially for high-stake situations with high uncertainty such as at the intensive care units (ICUs), these systems have the potential to curb medical errors by assisting health care providers in their decision making process. Sepsis prediction in ICUs (Moor et al., 2021), severity of illness prediction (Ghassemi et al., 2015), mortality prediction (Caballero Barajas and Akella, 2015) and, intensive care unit readmission prediction (Rojas et al., 2018) are such example use cases.

EHRs have the potential to bring solutions to diverse problems in the healthcare ecosystem. However, the variety of health records brings complexity and challenges in direct use of raw EHR data in predictive models. This data could be structured such as e-prescription or medical codes, or unstructured such as clinical notes taken by the physician. EHR data usually are large scale, contain missing information, they are noisy because they are not subject to high-quality audits. Also, since the data are drawn from many different resources, they are heterogeneous (Si et al., 2021). Moreover, since EHRs usually represent distinct medical events, the data are high-dimensional and often sparse. To input EHRs into a predictive model that works with vectors, the patient's health information should be represented by numerical values. Many data types in EHR can be high dimensional and sparse, a common example being the disease types. While there are thousands of diseases; only a handful of diseases would be had by a typical patient and the rest would be unobserved. If the disease history of a patient is to be considered as a feature for a machine learning model, it would be a categorical feature. The most common representation of the categorical features is vectors with binary entries (one-hot encoded vectors) where each disease will be represented by such a vector. These feature vectors shall be mostly populated by zeros with a few entries of ones. This type of data is called as *sparse data*. The success of any predictive model inevitably depends on the quality of the input feature representation. Sparse data sets pose particular challenges for training predictive models as they increase the time and space complexity of machine learning models. Additionally, the sparse data may bring noise to models so it becomes harder to generalize. As a result the models tend to overfit, and the predictive performance is compromise (Kuss, 2002).

1.2 Thesis Scope and Organization

This work investigates patient representation learning from EHRs. Among many technical challenges associated with EHRs representation, we address with its sparse and heterogeneous nature. We propose a strategy where one can unify the heterogeneous data types in a knowledge graph framework and learn a dense patient representation that encodes meaningful information from EHRs. We compare our strategy against widely used knowledge graph embedding methods in different prediction task at ICU. These tasks comprise mortality prediction and binarized length of stay prediction tasks.

1.2.1 Thesis Organization

This thesis is organized in 7 chapters. Chapter 2 gives an overview of the literature and Chapter 3 provides background knowledge on neural networks and knowledge graph embedding methods. Chapter 4 describes EHRs and the dataset used in this thesis. Chapter 5 details the cohort selection procedure and the steps to build knowledge graph and evaluation criteria. In Chapter 6, we present experimental set ups and discuss the results, and Chapter 7 is the conclusion section where we highlight our major findings, discuss the limitations of the current work and point out future directions and possible follow-up work.

2. LITERATURE REVIEW

There were many different types of applications, built based on EHR datasets. In the early stages, statistical models were developed since computational power was inadequate. Over the years, storage areas and computational resources developed, which brought attention to machine learning and deep learning algorithms to use over datasets. These two techniques were applied on EHR datasets to predict diseases in the early stages (Gupta et al., 2019), finding similarities between patients (Brown, 2016), and extracting information (Lee et al., 2020).

2.1 Information Extraction

Clinical notes are more complex than organized elements of EHR data that are often utilized for billing and administrative purposes, and are largely used by healthcare practitioners for extensive recording. Each patient visit is accompanied by a variety of clinical documents, including admission notes, discharge summaries, and transfer orders. Extracting information from clinical notes is difficult since the unstructured nature. Previously, these solutions required a significant amount of feature extraction and ontology mapping, which is one reason for their limited acceptance. As a result, some recent research have focused on applying deep learning to extract meaningful clinical information from clinical notes.

Jagannatha and Yu (2016a) treats the concept extraction problem as a sequence labeling problem in which each word in a clinical note is assigned one of nine clinically related tags. They categorize tags into pharmaceutical and illness categories, with each category containing relevant tags such as medication name, dose, medication method, adverse medication event, indication, and disease severity. They work with a variety of RNN-based deep architectures like LSTMs and GRUs, bidirectional LSTMs (Bi-LSTM), and certain combinations of LSTMs with classic conditional

random fields (CRF). In their setup, they compare to baseline CRFs, which were previously thought to be the most advanced approach for extracting clinical ideas from text. As a result, they discovered that RNN based methods outperformed the CRF baselines, particularly in recognizing more subtle features like drug duration and frequency, and illness severity. Such sophisticated information is critical for clinical informatics but is not easily accessible through the billing-oriented clinical code system.

Fries (2016) focuses on detecting text spans of time and event mentions, as well as predicting relationships between clinical concepts and clinical note creation time. This work uses a deep-learning method for sequence labeling based on a vanilla recurrent neural network (RNN). Word embeddings were generated by Word2Vec (Mikolov et al., 2013) method. In addition, Fries also uses Stanford's DeepDive (Shin et al., 2015). As a result, their DeepDive implementation outperforms vanilla based RNN implementation.

Liu et al. (2018b) investigate the challenge of expanding abbreviations in clinical notes. In order to prevent misunderstanding and make the clinical notes more accessible to a broader audience, their objective is to standardize all abbreviations in intensive care unit (ICU) notes. To accurately capture the semantics of an abbreviation in its context, they use Word2Vec (Mikolov et al., 2013) method to generate word embeddings by using clinical notes recorded in ICU, Wikipedia and medical articles. As a result, 82.27% accuracy was obtained by integrating the embeddings with domain-specific information, which surpasses baselines and is comparable to human performance.

2.2 Patient Representation

EHR systems are loaded with a huge number of distinct medical codes that represents the health status of patient. These codes were originally designed for internal administrative and billing activities, however they also provide useful information for secondary activities. In order to project distinct discrete codes into vector space for more detailed analysis, several deep learning techniques have been used. These vector spaces can be used to represent patients.

Miotto et al. (2016) describes a unique unsupervised deep feature learning technique for generating a general-purpose patient representation from EHR data enhance the

performance of clinical prediction models. From the Mount Sinai data warehouse of about 700,000 patients, hierarchical regularities and dependencies in EHRs was captured by using three-layer stack of denoising autoencoders. Their results surpassed those obtained using raw EHR data representations and other feature learning algorithms. Prediction accuracy for severe diabetes, schizophrenia, and numerous malignancies was among the best.

Pham et al. (2016) an LSTM based deep learning framework called DeepCare, which predicts the upcoming medical outcomes of patients. It models patient health status trajectories with explicit illness memory at the data level. In addition, it provides time parameterizations to accommodate unpredictable timing by regulating forgetting and consolidating illness memory. It generates and concatenate two independent vectors for a patient's temporal diagnostic and intervention codes to construct a patient representation vector, demonstrating that the resulting patient timeline vectors had more predictive value than classifiers trained on raw categorical data.

Time sequence based health records contain information on patients' visits, applied procedures, used medications, and test results, which show how their health status has changed over time. Such data are especially useful in describing illness development and early diagnosis. Mehrabi et al. (2015) proposes a RBM based architecture for detecting temporal patterns in Rochester Epidemiology Project data. Each patient's records were represented as a temporal clinical event matrix, with ICD9 and HCUP CSS diagnostic codes as rows and years of diagnosis as columns. Patients who were 18 or younger at the time of diagnosis were chosen. A deep three layer Boltzmann machine network was built, using the diagnostic matrix values of each patient as visible nodes.

2.3 Outcome Prediction

Ultimate purpose of many works in EHR domain is to predict patient outcomes. We distinguish two types of outcome prediction methods: (1) non-graph based techniques (which use feature engineering to extract features from data) and (2) graph based approaches (features are extracted automatically by representing data as a graph).

2.3.1 Non Graph Based Approaches

Gupta et al. (2019) applied machine learning and deep learning to predict childhood obesity in the early stage. Each patient has a plenty of visits where each visit contains diagnosis, prescribed medication, applied procedures, and lab results. Diagnoses in a visit were represented by a binary vector of length D where D is the total number of unique diagnoses in the whole dataset. A similar representation method was also used for medications and procedures. For lab results, the real value of the corresponding lab result was placed in the vector. The concatenation of these four vectors were represented a single visit where the length of the concatenated vector is V. If a patient had T many visits, then that patient would be represented by a matrix with a dimension of TxV. For the baseline models, Linear Regression and Random Forest were used. Instead of using a matrix of TxV dimensions as an input, data aggregation was used so each patient was represented by a vector of length V. For diagnosis, medications and procedures sum of the occurrences were used. For lab results, the average of each lab result was calculated. In addition, demographic features such as sex, ethnicity, and race were used as categorical features and appended to the patient's vector. Moreover, a more complex model was built which uses 2 layer LSTM.

Liu et al. (2018a) used doctor notes to predict chronic diseases of a patient. For the baseline model, 1-gram, 2-gram and 3-gram -IDF scores of the most frequent 20.000 words were generated from doctor notes. Demographic features such as age, sex, and race were used as categorical features and appended to the -IDF vector of the patient. Also, the average of the laboratory results was appended. To enhance the performance, word embeddings were used instead. At first, pre-trained word embedding on PubMed was used. Then, word embeddings were generated on their own data, by using StarSpace Wu et al. (2018). For deep learning models, Convolutional Neural Networks and Bi-LSTM were used.

Esteban et al. (2015) predicted the sequence of medical events of patients by using personalized temporal latent embedding. Dataset was consist of patients with kidney failure and kidney transplants which also includes used medications and laboratory results. Each visit of the patient was represented by a binary vector, then a latent representation of that vector was generated. Markov Models were used to choose the last k many last visits to use as input. These generated representations were feed into a shallow neural network. The output of the neural network was the probability of the next event. To enhance performance, a personalized Markov Model was proposed. In addition to k many last visits, demographic information

of the patient, and aggregated history of the patient's medical events were used as input.

Gentimis et al. (2017) uses MIMIC-III dataset to predict if a patient stay in intensive care unit longer or shorter than 5 days. Patients with length of stay over 20 days were considered as outliers and removed from the dataset. Patient demographic information, ICD codes for procedures and diagnoses were used as input. For machine learning methods, random forest and neural networks were preferred.

The nature of the electronic health record dataset is rich, however it is too raw for a direct usage in prediction tasks. Wang et al. (2020) uses MIMIC-III dataset to provide a pipeline for transforming raw electronic health record dataset into data structures, which can be used in patient clinical prediction tasks. The provided pipeline includes unit conversion, outlier detection, missing value imputation, hourly aggregation of time varying features and semantic grouping of duplicate vital signs. At the end of the pipeline, patient demographics, time varying vital signs and laboratory measurement features were generated to perform mortality and length of stay prediction tasks. Logistic regression, random forest and gated recurrent unit with delay were used.

While machine learning research for health care has been steadily increasing, numerous roadblocks have hampered progress in leveraging digital health data. The fundamental difficulty is the lack of universally acknowledged benchmarks for comparing different models. Harutyunyan et al. (2019) uses MIMIC-III dataset to create benchmarks in following tasks: in-hospital mortality prediction, decomposition prediction, length of stay prediction and phenotyping. More than 42,000 intensive care unit stays were selected as cohort, 15% of them were used as test data. From that cohort, 17 clinical variables were extracted as time series and fed into channel-wise LSTM which uses multitask learning.

2.3.2 Graph Based Approaches

Wu et al. (2019) have created a knowledge graph of patients to find similarities between patients. Conducted medical services and patient's doctor information was used. First of all, service embeddings were generated. Each medical services were a node, and the weight of the edge between two nodes is the co-occurrence frequency of these two medical services. Adjacency matrix was created, then Word2Vec (Mikolov et al., 2013) algorithm was applied to generate service embeddings. Sec-

ondly, doctor embeddings were created by predicting the doctor's primary specialty from conducted medical services, by using Graph Attention Networks (Veličković et al., 2017). Lastly, similarities between patients were calculated by shared doctors & medical services. This similarity was represented by a bipartite multigraph.

Zhang et al. (2016) aimed to create a unified representation of patients in EHRs. This representation was made by a graph where each node is a medical event, including disease, medication and laboratory result. Each edge represents the closeness of these medical events. Closeness is high if these events occurred at the same time. In addition, if two medical events were the same type, then closeness is also high. Since there were many medical events, each of them was mapped to a higher granularity by using Chinese Pharmacopoeia. Generated representations were used to predict medical risk.

Rocheteau et al. (2021) proposes a hybrid LSTM-GNN model for patient outcome prediction tasks. LSTM was used to extract temporal features from measurements & vitals signs, on the other hand GNN was used to extract patient neighbourhood information. It uses eICU Collaborative Research Database (Pollard et al., 2018) as data source, which is an electronic health record dataset based on patients in ICU. Each patient was represented as a node and existence of edges between each patient node depends on the shared diagnoses. Temporal features, which was created by LSTM, were used as node features. Patient embeddings were trained with GNN, concatanated with temporal features and patient demographic features to perform length of stay and mortality prediction tasks.

Lee et al. (2020) proposes dynamic multi-modal graph model, which uses both diagnosis codes and clinical notes. It uses dynamic graph approach since patients' medical condition varies from one visit to another. Graph convolutional networks (Kipf and Welling, 2016) was applied to perform subsequent diagnosis code and visit severity level prediction tasks.

Zhu and Razavian (2021) proposes a variationally regularized encoder - decoder GNN architecture to represent patients as embedding vectors by using short term ICU information and long term health information. In the encoder part, for each patient, all observed codes in EHR was represented by a node and fully connected graph was generated. Multiple graph layers were used to learn the graph structure and update representations. In the decoder part, additional nodes were added for each prediction task and fully connected to output nodes of the encoder graph. Mortality prediction and Alzheimer disease prediction were the main task of this paper.

2.4 Computational Phenotyping

As the quantity and accessibility of electronic health records have increased in recent years, there is a significant potential for reviewing and revising broad disease and diagnostic definitions. Traditionally, diseases are described by a collection of manual clinical descriptions; however, computational phenotyping tries to create better, data-driven definitions of disorders. It is feasible to uncover natural clusters of clinical descriptors that lend themselves to a more fine-grained illness description by utilizing machine learning and data mining techniques. Detailed phenotyping is a significant step toward the ultimate aim of individualized and precision medicine.

Patient visit to healthcare professionals result in electronic health record entries (EHRs). Mining these information allows for the extraction of electronic phenotypes, however it contains several challenges. High-quality phenotyping is expensive and requires evaluation from healthcare professionals, most of the fields in electronic health records are missing, disease definition changes over time. Beaulieu-Jones et al. (2016) proposes a semi-supervised learning technique for EHR phenotype extraction which uses denoising autoencoders. Denoising autoencoders reduce dimensionality, allowing visualization and grouping for the discovery of novel disease subgroups. They discovered that classification performance improves across several simulation models and enhanced survival prediction in ALS clinical trial data by integrating denoising autoencoders with random forests. This architecture is a potential tool for detecting disease subgroups and enhancing phenotype association studies using EHRs.

Working directly with EHR has several obstacles, such as temporality, sparsity, noisiness, and so forth. As a result, successful feature extraction or phenotyping from patient EHRs is a must for any downstream task. Cheng et al. (2016) present convolutional neural network based architecture for phenotyping using patient EHRs. They represent each patient's EHR as a matrix with time and event dimensions. On top of that matrix, a four layer CNN is used to extract phenotypes where the first layer is made up of EHR matrices. The second layer is a one-sided convolution layer which is capable of extracting phenotypes from the first. The third layer is a max pooling layer that introduces sparsity on the identified phenotypes, retaining only those that are important. The fourth layer is a softmax prediction layer that is fully connected.

Richness of electronic health records makes mining operations more difficult. Lipton et al. (2015) proposes a sequence based model to predict phenotypes of the patient

in ICU by treating phenotyping task as a mutli label classification problem. They use RNN and LSTM to detect phenotypes in multivariate time series of clinical measurements, consist of selected 13 variables. This implementation outperforms logistic regression and MLP baselines.

3. BACKGROUND

This chapter provides background information on the methods that we employ throughout the thesis.

3.1 Artificial Neural Networks

The neural network structure contains layers of nodes where each node is a neuron. All nodes take a set of inputs to perform computation and pass the result to nodes in the following layer. The first layer in this structure is called *input* layer. The input layer is followed by a set of layers, called *hidden* layers. At the end there is an *output* layer. The nodes in the network are called *neurons* and the whole structure is called *artificial neural network*. Figure 3.1 is a simple representation of an artificial neural network (Haykin, 2009).

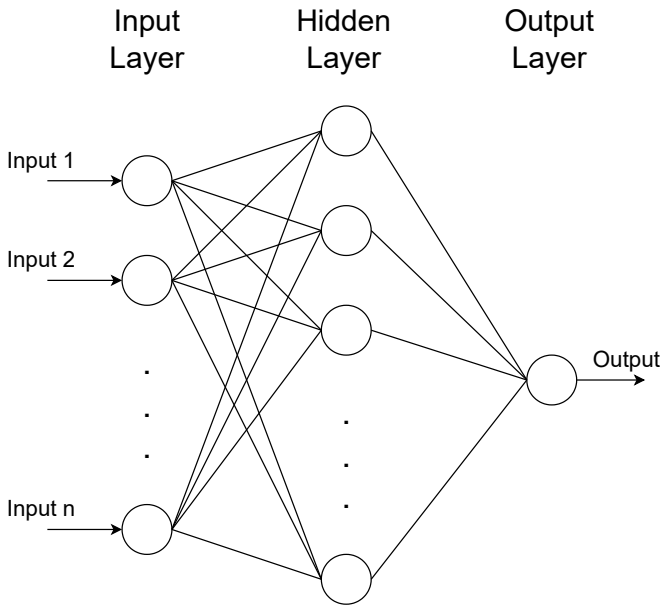

Figure 3.1 Artificial neural network with n inputs, a single hidden layer and an output layer with a single neuron.

All neurons in a fully-connected neural network operate the same way. All nodes take a set of inputs and apply computation to produce an output. The output will be an input for all the nodes in the following layer. A neuron can be visualized as in Figure 3.2.

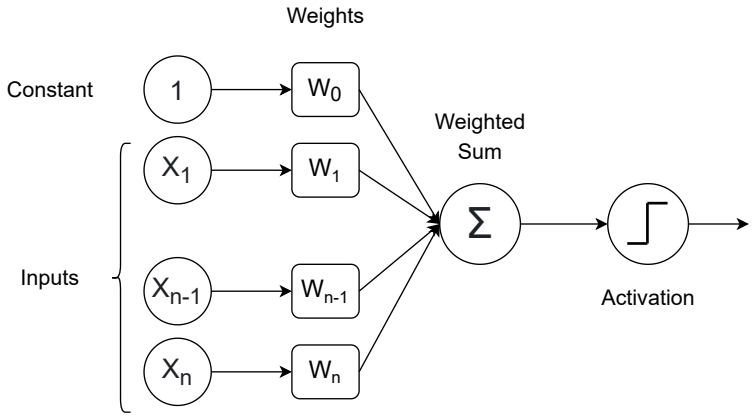

Figure 3.2 A neuron in a neural network. It takes n many inputs and one constant input. It multiplies each input with weight and calculates the sum. In the end, it passes the result to an activation function to produce an output.

Each node has a set of weights, which quantifies the importance of inputs. All the inputs from X_1 to X_n are multiplied with their corresponding weights from W_1 to W_n. This procedure is also applied to the constant term. All the multiplied values are added together to produce a weighted sum. This operation can be expressed mathematically as follows:

$$z = 1 * W_0 + X_1 * W1 * X_2 * W_2 * ... * X_n * W_n$$

This weighted sum is used as an input to an activation function. There exist different activation functions, but the most common alternatives are *Sigmoid* and *ReLU*. *Sigmoid* activation function is commonly used in the output layer for the binary classification task and it is based on the following formula to rearrange the input values between 0 and 1:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Due to the vanishing gradient problem, *Sigmoid* is not an appropriate choice in hidden layers. Instead, *ReLU* activation function is used in hidden layers. If the input value is 0 or more, the ReLU function returns the value as-is. If the input is less than zero, the ReLU function returns 0. It can be formalized as follows:

$$ReLU(z) = max\{0, z\}$$

The output of the activation function is what a neuron outputs. It is computed for each neuron, and except the output layer, it is passed into all neurons in the following layer. In the output layer, the computed value and the true label are passed into a *loss* function. Loss function can be a simple function like *mean squared error*, or a complex function like *cross-entropy*. The output of the loss function indicates the performance of the neural network. These steps describe the *forward propagation* of the network. Neural network will reduce the loss by repeatedly adjusting the weights of each neuron. If the level of adjustment is decided by the gradients of loss function, this procedure is called as *back propagation*.

3.2 Convolutional Neural Networks (CNN)

A convolutional neural network (CNN) is a special type of neural network that focuses on processing 2-dimensional data, such as images. With minor pre-processing on images, CNN can be used directly to extract features from images. A typical CNN architecture contains 3 layers: a *convolution* layer, a *pooling layer* and a *fully*

connected layer. This architecture can be visualized in Figure 3.3.

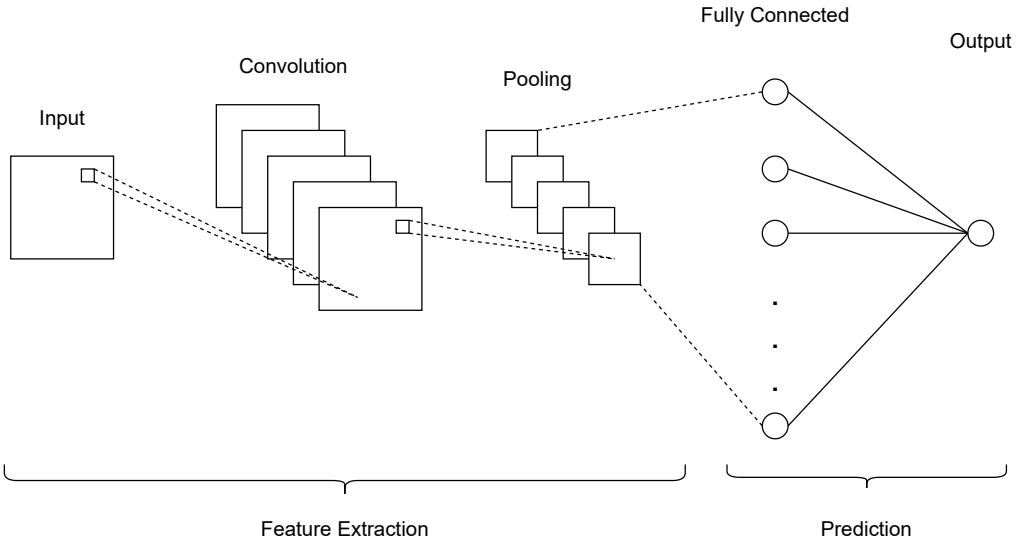

Figure 3.3 Architecture of a basic convolutional neural network. Contains a single convolution layer with 5 filters and single pooling layer for each filter. Output of the pooling layer is flattened and fed to the fully connected layer to perform prediction.

In the convolution layer, features are extracted by using *filters*. Each filter is also 2-dimensional and slides across the image from top left to bottom right. At each step, a value is calculated by using element-wise multiplication on the current slice of the input and filter values. This operation is called as *convolution*. This operation is visualized in the Figure 3.4.

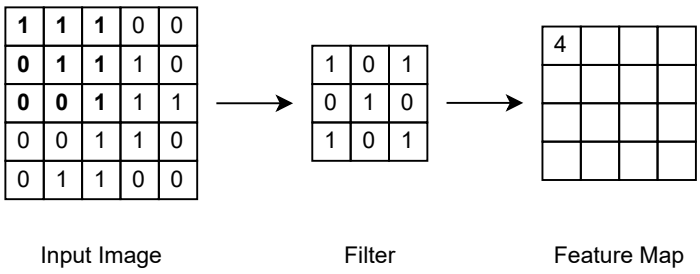

Figure 3.4 Convolution operation for a single filter. The filter slides across the image from the top left. Filters' current position on the input image is indicated with bold characters. The convolution operation is applied for a slice on the input image and filter values. Output is placed in the feature map.

The pooling layer is conceptually similar to the convolution layer. Like a filter, a fixed 2-dimensional object without any parameters slides across the image from top left to bottom right. In the max pooling variant, the maximum value for the current slice of the input data is selected as output. With this approach, the size

of the outputs of the convolution layer is reduced while preserving the important features. This operation is visualized in Figure 3.5.

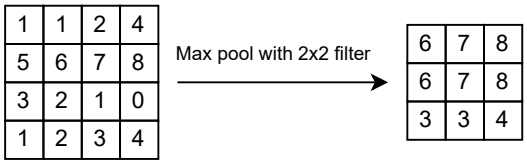

Figure 3.5 A 4x4 input data is fed into pooling layer. Max pooling approach is applied with a 2x2 filter element. Output of this layer is a 3x3 array.

CNN can also be used with 1-dimensional data. For instance, patients in ICU generates vital sign over time which is a 1-dimensional sequential data. In that case, 1-dimensional *filter* slides on the data to extract features by applying a convolution operation. In addition, the pooling operation is also applied by a 1-dimensional filter object. This implementation is called *1 Dimensional CNN*.

3.3 Recurrent Neural Networks (RNN)

The recurrent neural network (Hopfield, 1982) is a special type of neural network in which the previous step's output is fed as input to the current step. It is mainly used for sequence classification (Wang and Tian, 2016), sequence labeling (Jagannatha and Yu, 2016b) and sequence generation (Dušek and Jurčíček, 2016). It can capture information from sequence (time series) data by passing the message from the previous unit to its' successor. The inner architecture of three RNN units is visualized in Figure 3.6.

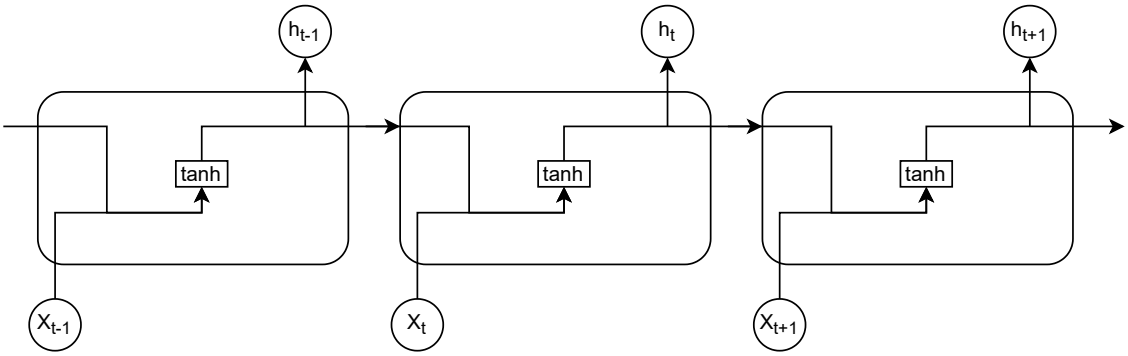

Figure 3.6 Three recurrent neural network units. Each unit gets two inputs: X_t (input data for the current step) and h_{t-1} (generated message from the previous unit). Inputs are fed into *tanh* activation function to produce h_t, which will be the output of the unit. h_t will also be passed to the next unit and this is how recurrent neural networks can capture information from sequence data.

RNNs can handle short time series very well; however, if the time series is very long, then it will fail to pass information from the distant past. This problem is known as the *vanishing gradient* problem.

3.4 Long Short-Term Memory (LSTM)

Long short-term memory (Hochreiter and Schmidhuber, 1997) is a special type of recurrent neural network which can capture long-term dependencies and thus work well on long time series. Like RNN, each unit in LSTM also passes information to its' successor. Different from RNN, units in LSTM have more complex architecture. Instead of having single *tanh* activation function, each unit contains multiple activation functions. The inner architecture of three LSTM units is visualized in Figure 3.7.

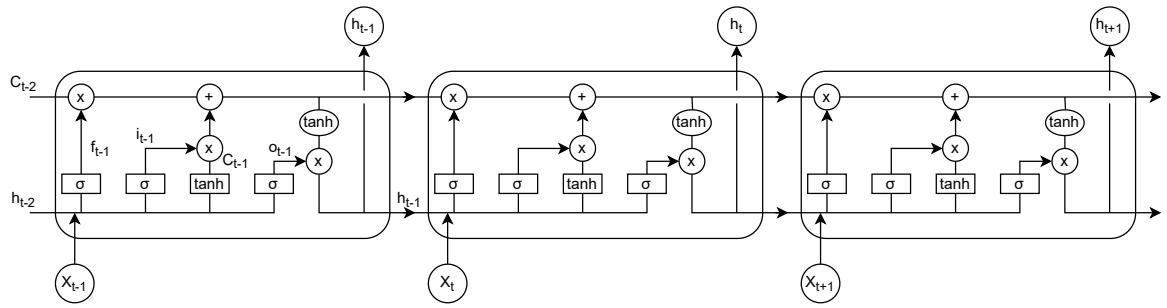

Figure 3.7 Three long short-term memory units. Each unit gets three inputs: X_t (input data for the current step), h_{t-1} (generated message from the previous unit) and c_{t-1} (cell state of the previous unit). Cell state was introduced with LSTM to capture long-term dependencies. Each unit has two outputs: c_t (cell state for the current unit) and h_t (hidden state of the current unit). In each unit, σ represents sigmoid activation function.

In the first step, *forget gate layer* is used. Based on the current input and previous cells' messages, it will either forget the cell state or keep it. This step can be formulated mathematically as:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

If the value of f_t is 1, then cell state will keep its value.

In the second step, it will decide if new information will be stored in the cell state. In the first part of this step, *input gate layer* is used to decide the parts of the input to be stored.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

Then, candidate values will be created as a vector which can be added to state.

$$\widetilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$$

Lastly, based on the output of the first step, current state will be updated with the candidate values.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

In the last step, based on the cell state, the output of the cell will be computed. The sigmoid layer is used to filter certain parts of the cell state. Then, filtered parts are fed into the tanh layer and multiplied with the output of the sigmoid layer as follows:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

With the incorporation of cell state, LSTM can handle long-term dependencies better than RNN. However, its complex architecture makes LSTM train slower and requires more memory.

3.5 Transformer

Vaswani et al. (2017) proposed Transformer architecture, which takes sequential data as input and produces a new sequence as output. The inner architecture contains encoder and decoder layers. Each encoder block contains a self-attention layer that calculates the relationship between different timestamps in the sequential data, and a feed forward layer follows it. Similarly, each decoder block contains self-attention and a feed-forward layer. However, in the decoder block, an encoder-decoder attention layer is placed between self-attention and feed-forward. Transformer is visualized in Figure 3.8:

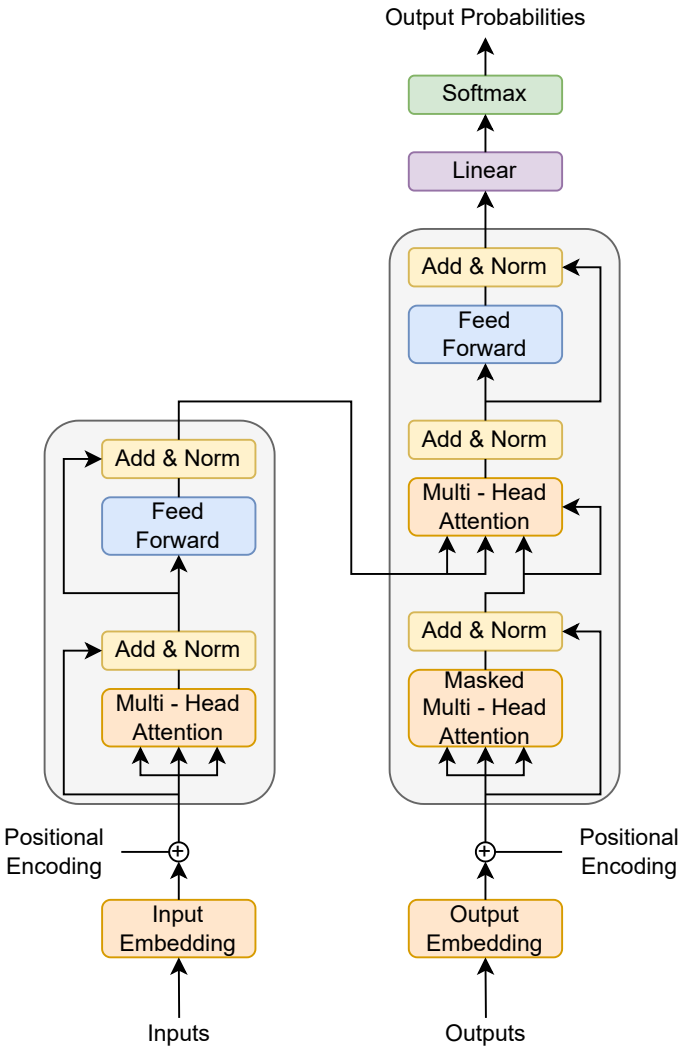

Figure 3.8 Transformer (Vaswani et al., 2017). The encoder layer is placed on the left, and the decoder layer is placed on the right. In the figure, both encoder and decoder layers contain a single block. Different implementations can use multiple blocks in both encoder and decoder layers.

Attention is the backbone of the Transformer architecture. Whenever the model is processes the current timestamp, attention allows the model to focus on the consecutive timestamps. With the usage of attention in Transformer, every timestamp's relationship with all others is calculated.

Transformer is mainly used in sequence to sequence tasks, such as machine translation (Wang et al., 2019), text summarization (Liu and Lapata, 2019) and speech recognition (Gulati et al., 2020). Since the prediction task in this thesis is classification, it is not possible to use the proposed architecture in Figure 3.8 directly. In a classification task, a decoder block is not needed. Thus, in this study, a simpler implementation which only has an encoder block is used.

3.6 Knowledge Graphs

A knowledge graph is a multi-relational graph consisting of nodes (entities) and edges (relations) to represent data. It is built by set of triples *(head, relation, tail)* where *head* and *tail* are two different nodes, and *relation* is an edge that connect two nodes. Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2015), YAGO (Suchanek et al., 2007) and NELL (Carlson et al., 2010) are some of the commonly known examples for knowledge graphs which are used in many different domains, such as network security (Noel et al., 2016), management systems (Aliyu et al., 2020), recommendation systems (Cao et al., 2019) and many more. For example, the knowledge graph encodes film industry information would contain nodes to represent characters, actors, movies, and genres; with nodes connected by starred-in, played, character-in, and genre edges. Visualization of this knowledge graph is shown in Figure 3.9.

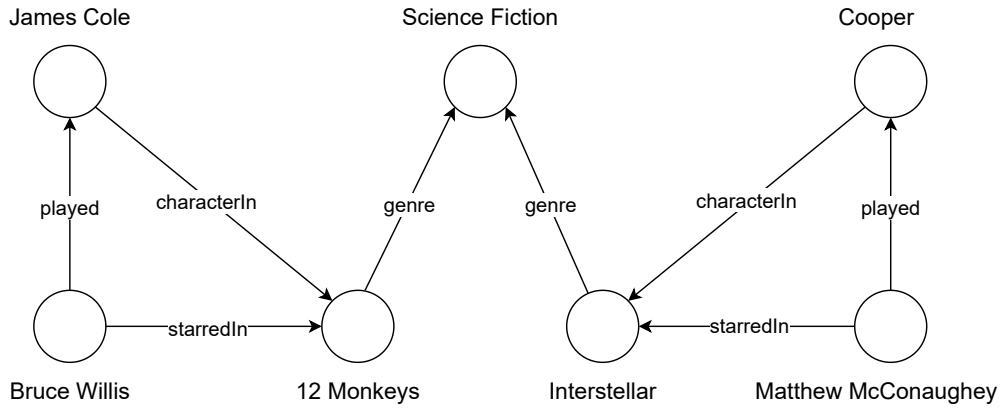

Figure 3.9 Film industry, represented by a knowledge graph. Constructed graph contains characters, actors, movies, and genres as entity types. In addition, starred-in, played, character-in, and genre are shown as the relation types on this graph.

Knowledge graphs follow the open-world assumption, which means that the absence of a relation does not automatically imply that the relation is false. It is simply accepted as unknown. As an example in Figure 3.9, there is no relation between Bruce Willis and Interstellar. In this case, the relation between Bruce Willis and Interstellar is accepted as unknown. There are four types of relation patterns that are very important in knowledge graphs. These patterns are symmetry, antisymmetry, inversion, and composition. Formal definitions for these patterns are as follows (Hamilton, 2020):

Definition 1. A relation is **symmetric** if $\forall(x, y)$

$$r(x, y) \Rightarrow r(y, x)$$

Definition 2. A relation is **antisymmetric** if $\forall(x, y)$

$$r(x, y) \Rightarrow \neg r(y, x)$$

Definition 3. A relation r_1 is **inverse** to relation r_2 if $\forall(x, y)$

$$r_2(x, y) \Rightarrow r_1(y, x)$$

Definition 4. A relation r_1 is **composed** of relation r_2 and relation r_3 if $\forall(x, y, z)$

$$r_2(x, y) \wedge r_3(y, z) \Rightarrow r_1(x, z)$$

In all definitions, r_i represents relation and x, y, z represents nodes.

3.6.1 Knowledge Graph Representation Learning

Knowledge graphs are known to be high dimensional and sparse. Nodes and edges can be represented using an adjacency matrix. However, this matrix would be sparse. Several methods have been proposed to generate meaningful dense vectors, called as *knowledge graph embeddings*, to represent nodes and edges. There exist certain techniques (Bordes et al., 2013), (Yang et al., 2014), (Trouillon et al., 2016) to extract embedding vector from knowledge graphs and both contains three major steps:

- Step 1: Initially, nodes and edges are mapped randomly to low dimensional continuous vector spaces by maintaining the structure of the knowledge graph.
- Step 2: For each triple in the graph, a set of negative triples are generated synthetically to improve the quality of the embedding vectors by preventing overfitting (Zhang et al., 2019). Then, a *scoring function* $f(h,r,t)$ is applied to each true and negative triples. The output of this function is the likelihood of a triple to exist in knowledge graph. A higher score typically indicates that the triple exists in the graph. Scores for the negative triples are expected to be low.
- Step 3: Use a $lossfunction$ to calculate to total loss. Then, solve an optimization problem to maximize the likelihood of observed triples by updating embeddings of the nodes and edges.

There exist certain methods to generate embeddings from knowledge graphs.

3.6.1.1 TransE

TransE (Bordes et al., 2013) is a translational distance model which uses a distance based scoring function. In this method, nodes and edges are represented in the same dimensional space, R^d. Assume that there are two nodes h (stands for *head*), t (stands for *tail*) which are connected with an edge r (stands for *relation*). The triple (h,r,t) indicates that node h and node t are connected by an edge r with low error, i.e. $h+r \approx t$. It can be visualized in Figure 3.10.

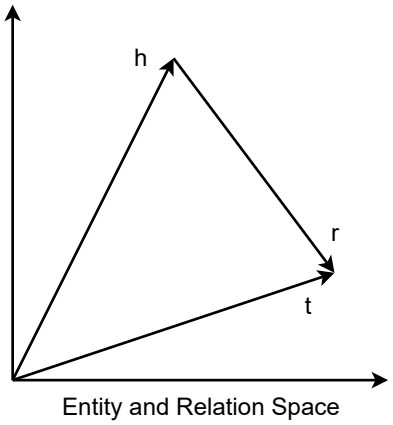

Figure 3.10 Visualization of TransE where h and t indicates two different entities and r represents a relation. For visualization, entites and relations are shown in 2 dimensional space.

The scoring function is defined as the negative distance between $h+r$ and t.

$$f(h,t) = -||h+r-t||$$

If the triple (h,r,t) exists, then the score should be high. On the other, if this triple does not exist in the graph structure, i.e. negative sample, the score should be low.

Relations in TransE satisfy antisymmetry, inversion and compostion pattern requirements. However it does not support symmetric relations since $f(h,t) \neq f(t,h)$.

TransE is very simple and it is commonly used as a strong baseline. However, it has weakness in representing 1 to N, N to 1 and N to N relations. The weakness in 1 to N can be explained by an example. Assume that (h,r,t_i) exists for $i = 1,2,...,p$. In this case, this method will force $h+r \approx t_i$ for all values of i. Weakness in N to 1 and N to N relations can be explained by a similar way. This disadvantage has overcome by introducing relation-specific hyperplanes in TransH (Wang et al.,

2014).

3.6.1.2 RESCAL

Unlike TransE, RESCAL (Nickel et al., 2011) is not a translational distance model, it is a semantic matching model. Its' scoring function is not distance based, but instead it is similarity based. It is the first method that uses semantic matching. RESCAL represents nodes by a vector in R^d, and relations as a matrix in R^{dxd} to model pairwise interaction between nodes. This idea is visualized in Figure 3.11.

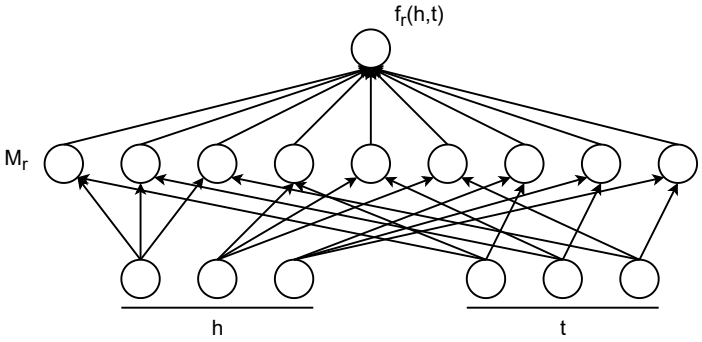

Figure 3.11 Visualization of RESCAL. For visualization purposes, d is selected as 3. h and t indicates entities in 3 dimensional space and M_r represents a relation matrix with 3 rows and 3 columns.

Score is calculated for every triple by the following scoring function:

$$f(h,t) = h^T * M_R * t = \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} [M_r]_{ij} * [h]_i * [t]_i$$

where nodes $h, t \in R^d$ and relation $r \in R^{dxd}$.

RESCAL can handle symmetric, antisymmetric, inversion and composition patterns. But due to matrix usage, RESCAL requires $\mathcal{O}(d^2)$ parameters per relation.

3.6.1.3 DistMult

Similar to RESCAL, DistMult (Yang et al., 2014) also uses semantic matching model. In DistMult, nodes h and t are still represented as a vector, however re-

lation r is represented by a diagonal matrix M_r, where the diagonal entries describe the relation r. This approach reduces the number of parameters from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. This idea is visualized in Figure 3.12.

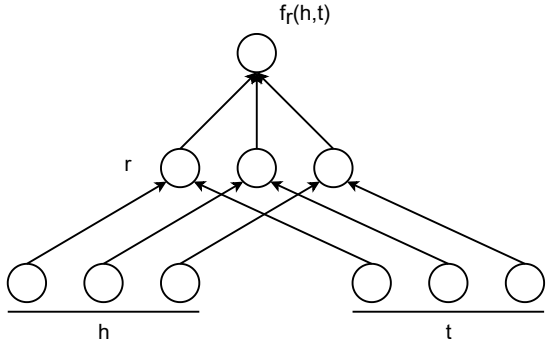

Figure 3.12 Visualization of DistMult. For visualization purposes, d is selected as 3. Similar to RESCAL, h and t indicates entities in 3 dimensional space and r represents a diagonal relation matrix with 3 rows and 3 columns.

TransE uses additive interaction whereas DistMult uses multiplicative interaction in its scoring function:

$$f(h,t) = h^T * diag(r) * t = \sum_{i=0}^{d-1} [h]_i * [r]_i * [t]_i$$

Using a diagonal matrix reduces the complexity. However, it introduces a disadvantage. The disadvantage of DistMult is that although it can model symmetric relations, it cannot handle asymmetric relations since $f(h,t) = f(t,h)$. Because of that, it is not applicable in general knowledge graphs. Relations in DistMult only satisfy the symmetry pattern.

3.6.1.4 ComplEx

ComplEx (Trouillon et al., 2016) can be interpreted as an extension DistMult. DistMult uses real space whereas ComplEx uses complex space to handle asymmetric relations which cannot be handled by DistMult. Nodes h and t and edge r are lie in a complex space C^d. Similar to DistMult, diagonal matrix is used to capture the symmetric relations where the asymmetric relations are handled by the imaginary part. Scoring function is defined as

$$f(h,t) = Re(h^T * diag(r) * \hat{t}) = Re(\sum_{i=0}^{d-1} [h]_i * [r]_i * [\hat{t}]_i)$$

Re stands for the real part of the complex value and $\hat{t}$ is the conjugate of t. Relations in ComplEx satisfy symmetry, antisymmetry and inversion patterns but not the composition pattern.

4. DATASET DESCRIPTION

In this chapter, we share the evolution of electronic health records. In addition, we explain the publicly available electronic health record dataset that we used in the experiments, called MIMIC-III.

4.1 Electronic Health Records

The systematic collection of human health data started in the late 1800sto track infectious diseases such as cholera, smallpox, and tuberculosis (Birkhead et al., 2015). This information was used to quarantine patients to prevent the further spread of diseases. Over the years, the domain and purposes of recording health information changed. By 1930s, the recording of sexually transmitted diseases (Daskalakis, 2017), vaccine-preventable diseases, and chronic diseases (Klompas et al., 2017) had been widely adopted. With the advancement of technology in healthcare systems, laboratory results and medical images became part of medical records, kept by healthcare providers.

Until the mid-1990s, most of the health information of patients was stored in paper (Evans, 2016). Using paper record to store a patient's health information have some inherent disadvantages. Locating and retrieving information is much slower and paper records entail large physical space for storage. Further disadvantages include vulnerability against unauthorized access and accidental data loss, physical deterioration of the documents and the complications in data back-up. Finally, paper record keeping greatly limits information sharing among different healthcare providers. When paper records are shared among different institutions different encoding systems would complicate the interpretation of the records.

With the development of technology, patients' health information is now stored

electronically. This electronic storage of a patient's health information is called an Electronic Health Record. In the United States, only 9.4% of hospitals were using basic EHR in 2008. With the support of HITECH, the adoption of at least basic EHR has risen to 83.8% by 2015 (Charles et al., 2013). Hospitals in the United States adopt one of the 3 versions of EHRs systems: Basic EHR, Basic EHR with Clinical Notes, and Comprehensive EHR . A comparison of these different versions can be found in Table 4.1.

Table 4.1 Comparison of EHR types in Unites States. (Charles et al., 2013)

Required EHR Functions	Basic EHR	Basic EHR with Clinical Notes	Comprehensive EHR
Electronic Clinical Information			
Patient demographics	*	*	*
Physician notes		*	*
Nursing assessments		*	*
Problem lists	*	*	*
Medication lists	*	*	*
Discharge summaries	*	*	*
Advance directives			*
Computerized Provider Order Entry			
Lab reports			*
Radiology tests			*
Medications	*	*	*
Consultation requests			*
Nursing orders			*
Results Management			
View lab reports	*	*	*
View radiology reports	*	*	*
View radiology images			*
View diagnostic test results	*	*	*
View diagnostic test images			*
View consultant report			*

Demographic information, previous and current diagnoses, procedures, laboratory

results, previously used medications, medical images, physician and therapy notes are stored in databases. In order to manage different types of medical information, a common terminology is used for different medical events. For instance, RxNorm vocabulary is mostly used for medications, Logical Observation Identifiers Names and Codes (LOINC) for laboratory test, Current Procedural Terminology (CPT) for procedures, and International Statistical Classification of Diseases and Related Health Problems (ICD) for diagnoses (Shickel et al., 2017). Example codes for these vocabularies can be found in Table 4.2.

Table 4.2 Examples for vocabularies used in EHRs for diagnosis, medications, laboratory results and procedures. (Shickel et al., 2017)

Vocabulary	**Examples**
ICD	**R19.1:** Abnormal bowel sounds **J12.0:** Adenoviral pneumonia
CPT	**78075:** Adrenal imaging, cortex and/or medulla **1004494:** Tenodesis at wrist
LOINC	**78803-4:** Barbital [Mass/mass] in Hair **2579-1:** Lutropin [Moles/volume] in Serum or Plasma
RxNorm	**1370581:** 2-sulfolaurate **1729527:** 5 ML torsemide 10 MG/ML Injection

Not all healthcare providers use the same vocabulary for the same medical event. To overcome this problem, the United Medical Language System (UMLS) and Systemized Nomenclature of Medicine - Clinical Terms (SNOMED CT) provide a mapping for different vocabularies. These mappings enabled data sharing across healthcare providers.

One challenge with EHR data is that it contains heterogeneous data. For example, laboratory results are represented as numerical values; date objects represent admission dates; diagnoses and procedures are represented as codes which makes them categorical data, and doctor notes are represented by free-text form. In addition, each record is ordered chronologically, which makes it difficult to apply certain techniques to analyze data and develop models.

The increasing volume of health-related data has fueled the development of machine learning applications in the healthcare ecosystem. Some of these applications are used directly by healthcare providers. To support researchers in developing applications for the healthcare ecosystem, some of the EHR datasets have been made publicly available. In the next section, we will detail the MIMIC- III database, which that is used in this work.

4.2 MIMIC - III

In this study, we use the Medical Information Mart for Intensive Care (MIMIC-III) EHR database (Johnson et al., 2016). It is the successor of the MIMIC-II database (Saeed et al., 2011), created by the Laboratory of Computational Physiology of The Massachusetts Institute for Technology (MIT). The aim of this database is to provide access to well-curated clinical information for data analysis. This dataset contains information on more than 46,000 patients in the ICU at Beth Israel Deaconess Medical Center Boston. Moreover, it holds the records of 56,000 ICU admissions since some patients would be admitted to ICU multiple times. The date of records begins in June 2001 and ends in October 2012. This database can be loaded into MySQL, Oracle, and PostgreSQL.

MIMIC-III Database contains demographic information, laboratory results, medication usage information, observations, diagnosis, procedures, lengths of stay, survival information, doctor notes, etc. In order to protect patient privacy, date entries were randomly shifted with varying offsets. Two critical care information systems were in use during data recording: Philips CareVue Clinical Information System and iMDsoft MetaVision ICU. Table 4.3 lists sample statistics of the patients who are 16 years old or above.

Table 4.3 Sample statistics of the MIMIC-III patients who are aged 16 years and above. CCU stands for Coronary Care Unit; CSRU stands for Cardiac Surgery Recovery Unit; MICU stands for Medical Intensive Care Unit; SICU stands for Surgical Intensive Care Unit; TSICU stands for Trauma Surgical Intensive Care Unit. (Johnson et al., 2016)

Critical Care Unit	CCU	CSRU	MICU	SICU	TSICU	Total
Unique patients	5,674 (14.7%)	8,091 (20.9%)	13,649 (35.4%)	6,372 (16.5%)	4,811 (12.5%)	38,597 (100%)
Unique Hospital admissions	7,258 (14.6%)	9,156 (18.4%)	19,770 (39.7%)	8,110 (16.3%)	5,491 (11.0%)	49,785 (100%)
Unique ICU stays	7,726 (14.5%)	9,854 (18.4%)	21,087 (39.5%)	8,891 (16.6%)	5,865 (11.0%)	53,423 (100%)
Median age	70.1	67.6	64.9	63.6	59.9	65.8
Gender, male	4,203 (57.9%)	6,000 (65.5%)	10,193 (51.6%)	4,251 (52.4%)	3,336 (60.7%)	27,983 (55.9%)
Median ICU length of stay	2.2	2.2	2.1	2.3	2.1	2.1
Median hospital length of stay	5.8	7.4	6.4	7.9	7.4	6.9
In ICU mortality	685 (8.9%)	353 (3.6%)	2,222 (10.5%)	813 (9.1%)	492 (8.4%)	4,565 (8.5%)
In hospital mortality	817 (11.3%)	424 (4.6%)	2,859 (14.5%)	1,020 (12.6%)	628 (11.4%)	5,748 (11.5%)

In total, MIMIC-III contains 26 tables in a relational database. Columns which

have the 'ID' suffix are treated as identifiers and used to link different tables. As an example, SUBJECT_ID represents a unique patient, HADM_ID represents a unique hospital admission and ICUSTAY_ID represents a unique intensive care unit stay. There are some tables which have the prefix 'D_'. These tables are treated as dictionary tables and provide definitions for the identifiers. For instance, DIAGNOSES_ICD contains ICD-9 codes of the diagnosis for the patients. Descriptions for all ICD-9 codes that are used in DIAGNOSES_ICD can be found in D_ICD_DIAGNOSES table.

Names of the each 26 tables and a single sentence explanation is shared in Table 4.4.

Table 4.4 Names and short details of the each 26 tables in MIMIC-III database, sorted alphabetically. (Johnson et al., 2016)

Table	Detail
ADMISSIONS	Contains unique hospital admissions, identified by HADM_ID.
CALLOUT	Details about discharging from ICU.
CAREGIVERS	Details about each caregiver.
CHARTEVENTS	All electronic chart related data is stored.
CPTEVENTS	CPT codes for applied procedures, used for billing.
D_CPT	Describes CPT codes.
D_ICD_DIAGNOSES	Describes ICD-9 codes which are used for diagnosis.
D_ICD_PROCEDURES	Describes ICD-9 codes which are used for applied procedures.
D_ITEMS	Describes local codes which are used to record vital signs.
D_LABITEMS	Describes local codes which are used in lab events.
DATETIMEEVENTS	Contains all events which are based on date and time.
DIAGNOSES_ICD	Diagnosis each patients, recorded by using ICD-9 codes.
DRGCODES	Contains diagnosis related groups (DRG) codes for patients.
ICUSTAYS	Contains unique ICU stays, identified by ICUSTAY_ID.
INPUTEVENTS_CV	Intake events of the patients, recorded by CareVue system.
INPUTEVENTS_MV	Intake events of the patients, recorded by MetaVision system.
OUTPUTEVENTS	Output events of the patients.
LABEVENTS	Resulst of the lab measurements.
MICROBIOLOGYEVENTS	Contains microbiology information.
NOTEEVENTS	Caregiver notes about the patient.
PATIENTS	Details about unique patients, identified by SUBJECT_ID.
PRESCRIPTIONS	Medication usage information during ICU stay.
PROCEDUREEVENTS_MV	Details about applied procedures, available in MetaVision system.
PROCEDURES_ICD	All applied procedures for each patient, recorded by using ICD-9 codes.
SERVICES	Set of services which a patient is transferred to.
TRANSFERS	Details about transfers from one service to another.

In the each following subsection, you can find information about the important tables in MIMIC-III.

4.2.1 Diagnoses

Diagnoses for each patient are stored in *DIAGNOSES_ICD* table. Diagnoses are represented by International Classification of Diseases Version 9 (ICD-9) codes. Descriptions of ICD-9 codes can be found in *D_ICD_DIAGNOSES* table. Since a patient can have multiple diagnoses, each diagnoses was ordered by priority where the order have an impact on treatment. In total, table contains 651.047 diagnoses with 6.984 of which are unique entries. On average, a patient has approximately 14 diagnoses.

ICD-9 codes for diagnoses are used for billing purposes and recorded at the end of the hospital stay. There have been previous studies which have used this table as an input in predictive tasks, but we opted not to since it can easily leak information. Counts of the most frequent 50 diagnoses were shared in Figure 4.1.

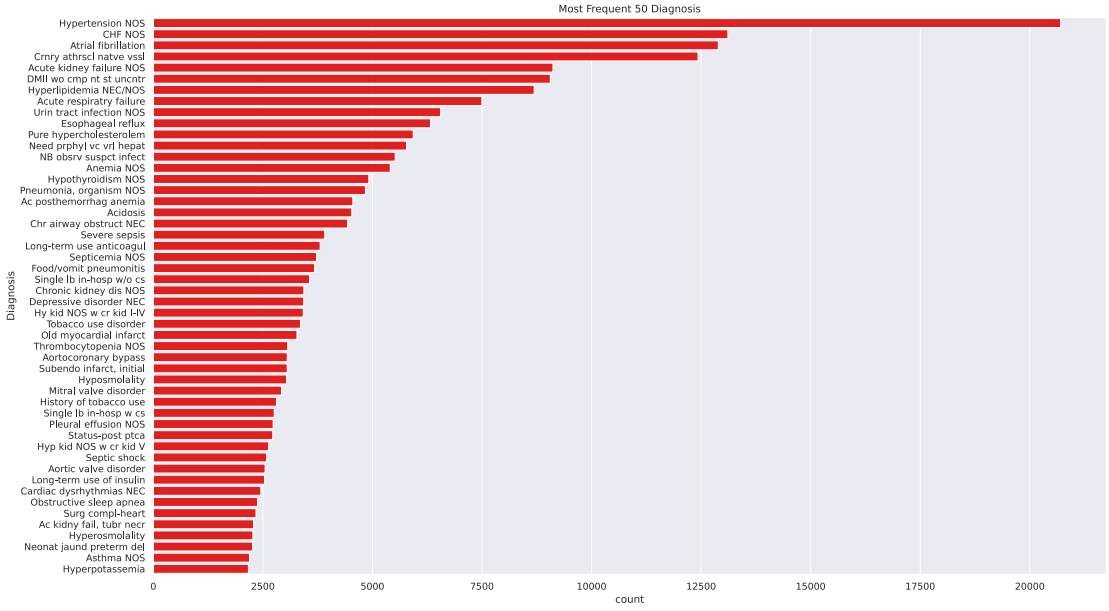

Figure 4.1 Most frequent 50 diagnosis in MIMIC-III dataset. Most common three diagnosis are hypertension, chronic heart failure, atrial fibrillation.

4.2.2 Procedures

Applied procedures for each patient are stored in *PROCEDURES_ICD* table. Like diagnoses, procedures are represented by ICD-9 codes. Descriptions of ICD-9 codes for procedures can be found in *D_ICD_PROCEDURES* table. Since multiple procedures can be applied to a single patient, each procedure are ordered by priority where the order have an impact on treatment. In total, the table contains 247.846 applied procedures, 2.009 of which are unique entries. On average, a patient had undergone approximately 6 applied procedures.

Counts of the most frequent 50 procedures are shared in Figure 4.2.

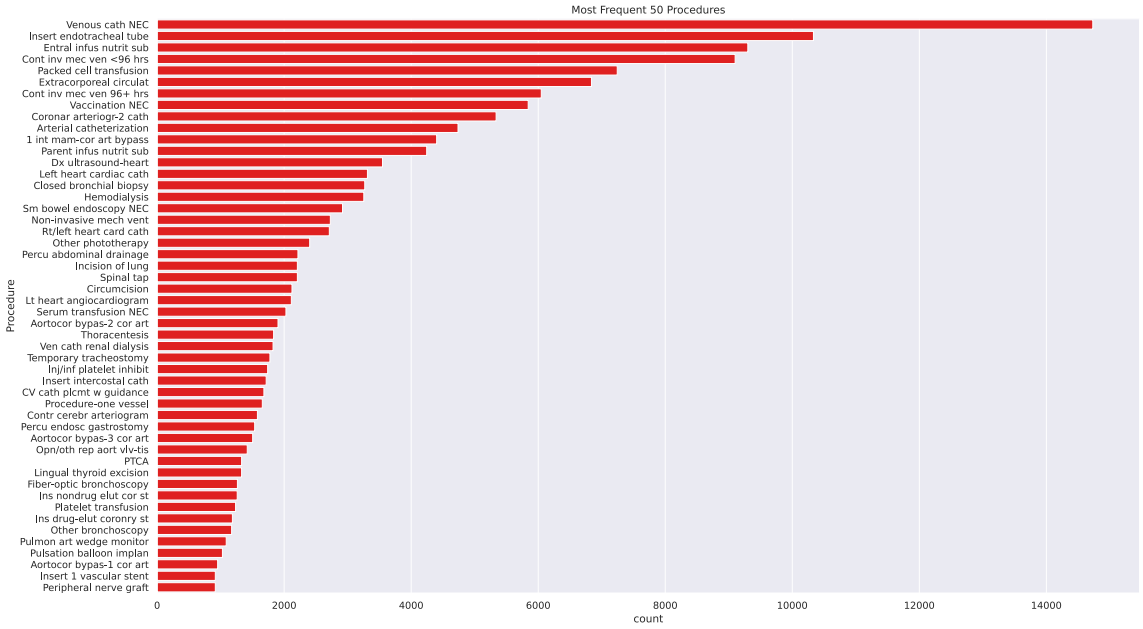

Figure 4.2 The most frequent 50 procedures in the MIMIC-III dataset. The most common three procedures are placing catheter into a large vein, inserting endotracheal tube and applying enteral nutrition.

4.2.3 Prescriptions

All the medication related information for a given patient is stored in *PRESCRIPTIONS* table. Name and type of the each prescribed drug is stored

in this table. In total, this table contains 4.156.450 prescribed drugs, 4.525 of which are unique entries. On average, 105 drugs had been prescribed to a patient.

Counts of the most frequent 50 drugs are shown in Figure 4.3.

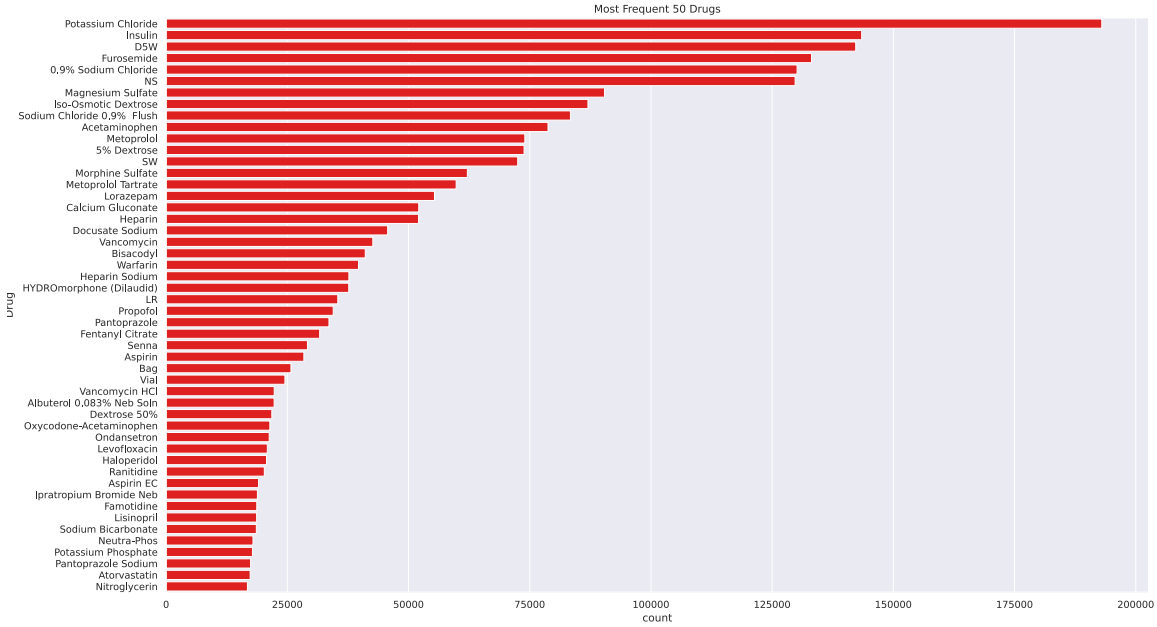

Figure 4.3 Most frequent 50 drugs in MIMIC-III dataset. Most common three drugs are potassium chloride, insulin, Dextrose 5%.

4.2.4 Lab Events

All the laboratory measurements for a given patient is stored in *LABEVENTS* table. The following is the procedure for obtaining a lab measurement: initially, a caregiver obtains a fluid from the patient's body (e.g. blood from an arterial line, urine from a catheter, etc). The fluid is then bar coded to link it to the patient and timestamped to keep track of when it was obtained. Within 4-12 hours, the lab analyzes the data and gives a result. In total, the table contains 27.854.055 lab events, 575 of which are unique. On average, 602 lab events pertains to a patient.

Counts of the most frequent 50 lab events are shared in Figure 4.4.

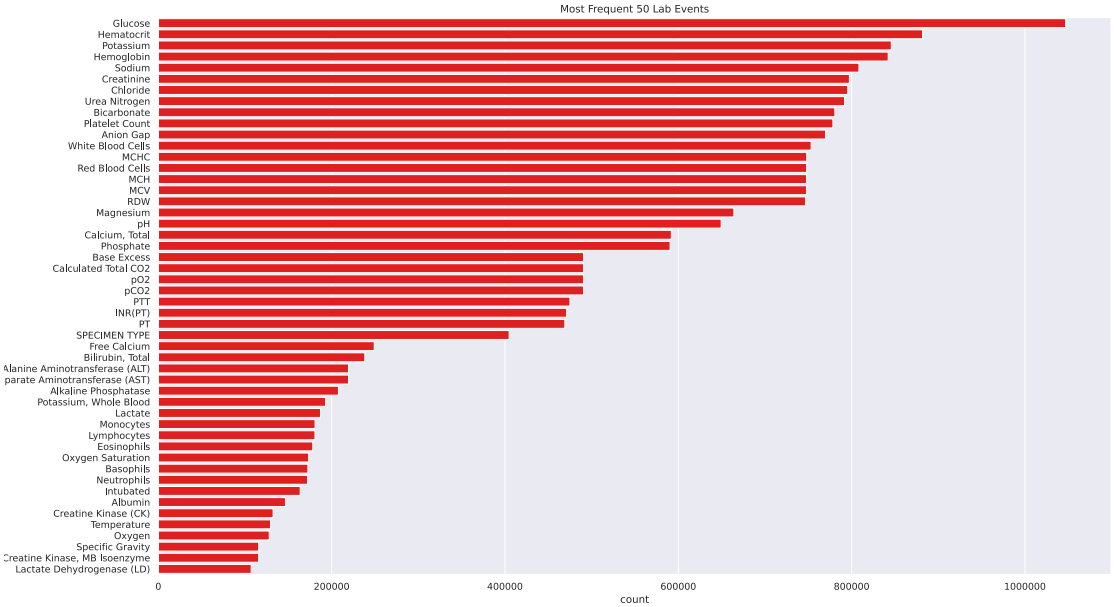

Figure 4.4 Most frequent 50 lab events in MIMIC-III dataset. The most common three lab events are glucose measurement, hematocrit measurement, and potassium measurement.

4.2.5 Input Events

Any fluids that have been given to the patient, such as oral or tube feedings, or intravenous solutions carrying drugs, are considered as inputs. Since MIMIC-III dataset uses Philips CareVue and iMDSoft Metavision as ICU information system, all the input events are stored in two different tables, $INPUTEVENTS\_CV$ and $INPUTEVENTS\_MV$. Since the observations are not duplicated across tables, the results from two tables can be merged. In total, 21.146.926 input events are recorded, 3.160 of which are unique. On average, 490 input events had been recorded for a patient.

Counts of the most frequent 50 input events are displayed in Figure 4.5.

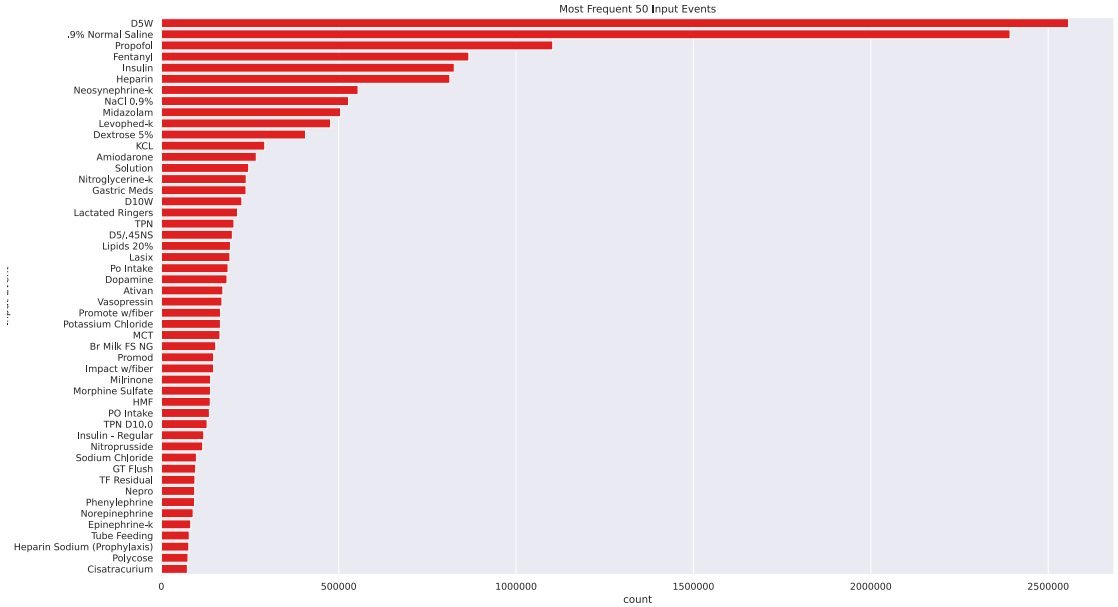

Figure 4.5 Most frequent 50 input events in MIMIC-III dataset. Most common three input events are Dextrose 5%, sodium chloride injection and propofol.

5. METHODOLOGY

In this chapter, we present the details of the experimental setups, and the representation of the patient data in the prediction models.

5.1 Cohort Selection

We use the same cohort selection criteria as in MIMIC-Extract (Wang et al., 2020). The selection criteria to include in the cohort are as follows:

- We select patients older than 15 years old.(This is needed due to substantial differences between adult and pediatric physiology)
- The ICU stay of the patients should be at least 30 hours.
- If a patient has multiple ICU admissions, we use only the first visit.

Applying the criteria to MIMIC-III datasets results in a set of $23,944$ patients. To be able to compare the performance results fairly with Wang et al. (2020), we generate the same training, validation and test folds by running the data split code with the same random seed. 70% of the patients are used in training, 10% for validation and 20% for testing.

5.2 Knowledge Graph Representation of MIMIC-III Dataset

To represent the rich medical information in EHRs, we use a knowledge graph. The graph contains patient nodes and nodes that represent the different entities in the

medical records. In Section 5.2.1 we describe the information that goes into the graph centered on a single patient.

Figure 5.1 Patient nodes.

5.2.1 Patient Demographic Information

The demographic information is included in the knowledge graph as follows:

- *Gender* represents the genotypical sex of the patient. It can take *Male* and *Female* as value. If the patient is male, his patient node is connected with the Male node, with an edge type *GENDER_IS*. Otherwise, it is connected to the Node that represent *Female*.
- To represent the *Age* of the patient, we use the following age buckets: *Under 30*, *Over 30 and Under 50*, *Over 50 and Under 70* and *Over 70*. The *Patient* node is linked to the corresponding age node with the edge type *AGE_IS*.
- *Ethnicity* attribute in MIMIC-III take values *Black*, *White*, *Asian*, *Hispanic* and *Other*. The *Patient* node is connected with the corresponding *Ethnicity* type with the edge *ETHNICITY_IS*.

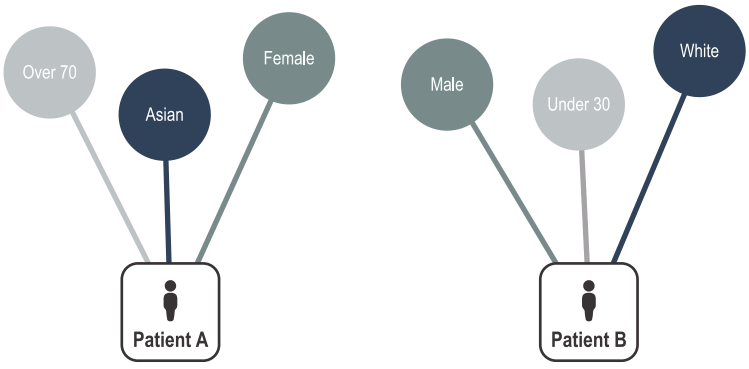

Figure 5.2 Patient and demographic information related nodes.

5.2.2 Hospital Stay Related Information

The patient hospital stay related information include the following:

- *First Care Unit* is the section of the ICU where the patient was first cared for. It can take the following possible value: *MICU* (Medical Intensive Care Unit), *CSRU* (Cardiac Surgery Recovery Unit), *SICU* (Surgical Intensive Care Unit), *CCU* (Coronary Care Unit), *TSICU* (Trauma Surgical Intensive Care Unit). Each *Patient* node and *First Care Unit* node is connected with by edge with the type of *FIRSTCARE_UNIT_IS*.
- *Admission Type* describes the type of the admission. *Emergency* and *Urgent* describes unplanned medical care. *Elective* describes a previously planned hospital admission. We connect the *Patient* node with its corresponding *Admission Type* with the *ADMISSION_TYPE_IS*.
- *Insurance Type* describes the type of the insurance of the patient during admission. Possible values for this information are *Medicare*, *Private*, *Medicaid*, *Government* and *Self Pay*. Each *Patient* node and *Insurance Type* node is connected by *INSURANCE_IS* type.

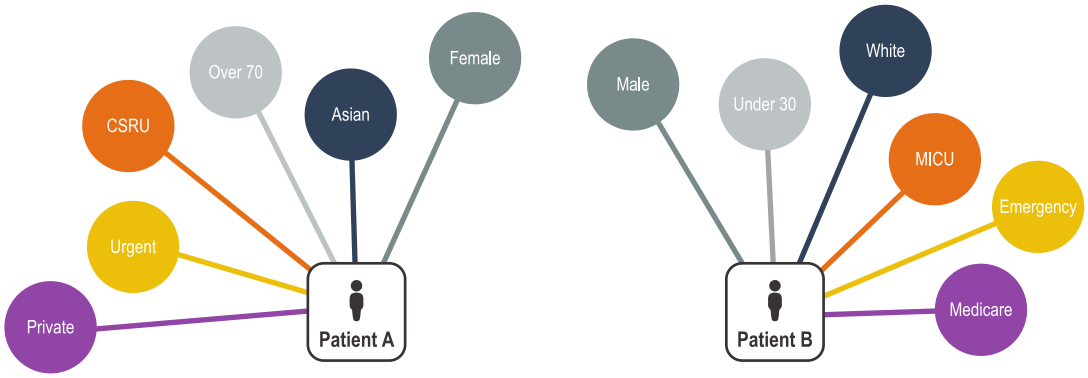

Figure 5.3 Patient, demographic information and hospital stay related information nodes.

5.2.3 In ICU Related Information

This type of information is recorded during ICU stay. We use only the information recorded in the first 24 hours of the ICU stay.

Input Event describes any fluids which have been administered to the patient. For instance, tube or oral feedings or intravenous solutions which contain medications. In total, there are 854 *Input Event* nodes. Each *Patient* node and *Input Event* node is connected by an edge with *TAKES_INPUT* type.

Lab Event describes information respecting laboratory based measurements. Each event of the laboratory measurement is flagged if the observed result was abnormal. This work only considers the events which have abnormal results. Some examples are *Hemoglobin* and *Hematocrit*. In total, there are 215 *Lab Event* nodes. Each *Patient* node and *Lab Event* node is connected by an edge of the *IS_ABNORMAL* type.

Prescription includes the prescribed medicines. Some examples are *Clindamycin* and *Simvastatin*. In total, there are 1970 *Prescription* nodes. Each *Patient* node and *Prescription* node is connected by *PRESCRIBED_WITH* edge type.

Procedure describes the administered procedures for the patient. Some examples are *Chest X-Ray* and *EKG*. In total, there are 112 *Procedure* nodes. Each *Patient* node and *Procedure* node is connected by an edge of the *RECEIVES_PROCEDURE* type.

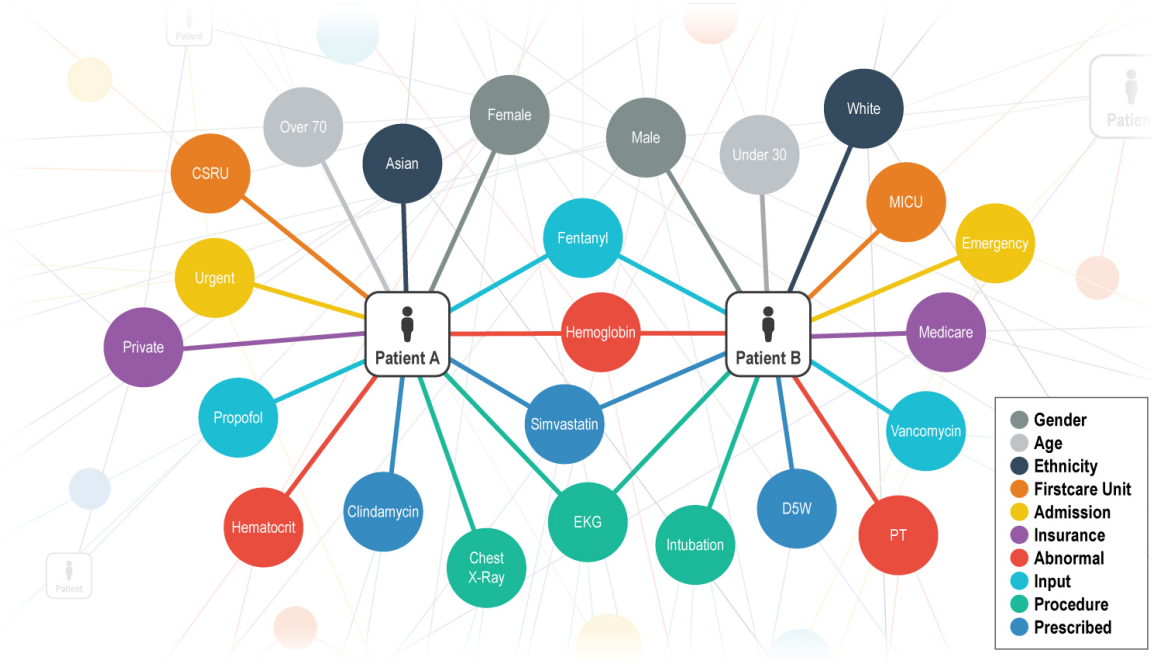

Figure 5.4 Patient Knowledge Graph, generated on MIMIC-III dataset. For visualization, knowledge graph was limited to two patients.

Figure 5.4 contains the visualization of the graph for two patients. In total, the knowledge graph contains 27.119 nodes and 1.161.686 edges. Number of node types

in the knowledge graph is 11 and Number of edge types in the knowledge graph is 10.

5.3 Target Data

Target data represents the target label which we would like to predict using the patient embedding vectors. In this thesis, we focus on the following binary classification tasks:

- In - ICU Mortality: Predicts whether the patient will die in the intensive care unit.
- In - Hospital Mortality: Predicts whether the patient will die in the hospital ward after discharge from the intensive care unit.
- Length of Stay Over 3 Days: Predicts whether the patient will stay in the intensive care unit longer than three days.
- Length of Stay Over 7 Days: Predicts whether the patient will stay in the intensive care unit longer than seven days.

After a patient is admitted to ICU, only the data generated in the first 24 hour of the ICU stay will be used. This time period is called the *observation window*. Predictions are not made immediately at the end of the observation window. If a patient dies in the 25^{th} hour of the ICU stay, this means that the patient's vital signs got worse before the observation window has ended. This situation may cause information leakage in the prediction models. In order to prevent this type of leakage, we use a 6 hour gap at the end of the observation window. We make predictions only after the 30^{th} hour. We call this window the *prediction window* as in Wang et al. (2020) . The prediction windows ends when the patient dies or is discharged from the hospital. We make predictions at the beginning of the prediction window. Figure 5.5 illustrates the observation and prediction windows.

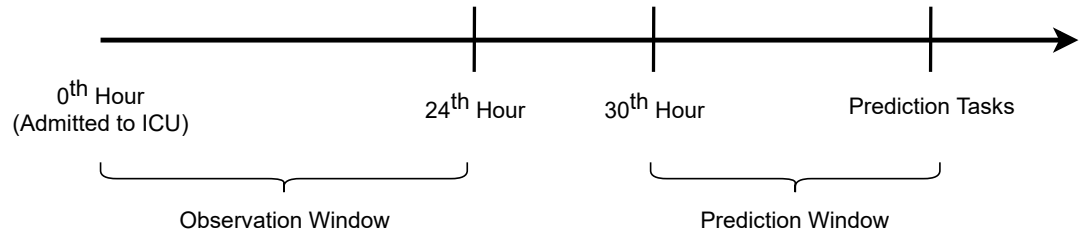

Figure 5.5 Observation window and prediction window.

5.4 Handling Class Imbalance

In the majority of the prediction tasks, the output labels were not distributed equally. In Table 5.1 presents the positive class ratios for each task.

Prediction Task	Positive Class Ratio (%)
In - ICU Mortality	7
In - Hospital Mortality	10
Length of Stay Over 3 Days	43
Length of Stay Over 7 Days	8

Table 5.1 Rate of positive class in prediction tasks.

Since there are few examples in the minority class, learning minority class examples will be challenging for naive application of a predictive model. In order to solve this challenge, we resort to cost sensitive learning (Elkan, 2001). We assign higher weight to samples with minority class label.

5.5 Evaluation Criteria

Due to data imbalance in the majority of the prediction tasks, accuracy score is not the right choice to evaluate the performance. In this thesis, we used AUROC (Bradley, 1997) and AUPRC (Davis and Goadrich, 2006) scores.

5.5.1 Area Under Receiver Operating Characteristic Curve (AUROC)

ROC (receiver operating characteristic curve) evaluates the performance of the binary classification on the positive (minority) class. X axis represents the false positive rate (ratio of the number of false positives divided by the sum of the false positives and the true negatives) and the Y axis represents the true positive rate (ratio of the number of true positives divided by the sum of the true positives and false negatives). In the best possible model, the value of the true positive rate is close to 1 and the value of the false positive rate is close to 0.

ROC uses the predicted class probabilities. After predicting class probabilities, a threshold value is used to classify samples to negative or positive class. By default, if the predicted positive class probability for a sample is greater than 0.5, than that sample will be labeled as positive. Then, the true positive and false positive rates are calculated for that threshold value. This approach will be done for many different threshold values. At the end, all the calculated true positive rates and false positive rates are plotted as a curve. This curve is called as ROC and AUROC is the area under the ROC.

5.5.2 Area Under Precision-Recall Curves (AUPRC)

PR curve (precision - recall curve) evaluates the performance of the binary classification on the positive (minority) class. X axis represents the recall (ratio of the number of true positives divided by the sum of the true positives and the false negatives) and the Y axis represents the precision (ratio of the number of true positives divided by the sum of the true positives and false positives). In the best possible model, both the value of the recall and the precision is close to 1.

The procedure for plotting PR curve is very similar to plotting a ROC curve. Instead of the true positive rate and the false positive rate, the precision and recall values will be plotted for every threshold value. While evaluating a binary classifier on imbalance dataset (Saito and Rehmsmeier, 2015), PR curve is more informative than ROC curve. To represent performance of each classifier with a value, the area under PR curve is used, which is called the AUPRC score.

5.6 Environment

In this thesis, Python 3.6 was used to carry out computations on Ubuntu 16.04.4 as the operating system with a Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, GeForce GTX 1080 Ti GPU and 256GB RAM. The CPU contains 10 cores and 20 threads where size of the L1 cache is 32KB, L2 cache is 256KB and L3 cache is 25MB.

6. EXPERIMENTS & RESULTS

The process of building the knowledge graph and learning node embeddings is explained in the previous section. This section presents the experiments conducted and the obtained results with these methods. There are several experimental setups. In all setups, embedding vectors were trained with the following knowledge graph embedding methods: TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), and ComplEx (Trouillon et al., 2016). After training the knowledge graph embeddings, the embedding vectors that correspond to the patient nodes are retrieved, and in the ensuing step, patients' embedding vectors are fed into a feed-forward neural network with three hidden layers.

6.1 Using Only Knowledge Graph Embeddings

We first aim to find out which knowledge graph embedding method works the best. To this end, in this experiment, we evaluate the success of the knowledge graph embeddings obtained with the methods TransE, DistMult, and ComplEx. Figure 6.1 visualizes this experimental setup.

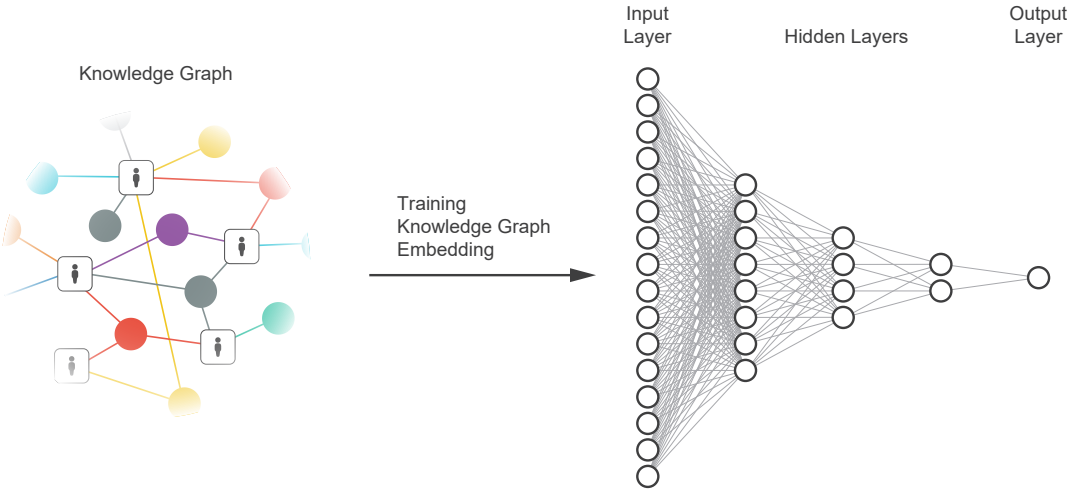

Figure 6.1 The feed-forward neural network architecture with 3 hidden layers. Only knowledge graph embeddings of the patients are used as input to the neural network.

We set the number of neurons in each hidden layer as half of the number of neurons in the previous layer. For instance, if the size of the embedding vector is 200, then the first hidden layer contains 100 neurons, the second hidden layer contains 50 neurons, and the last hidden layer contains 25 neurons. In each hidden layer, *ReLU* activation function is used. Output layer contains a single neuron with *Sigmoid* activation function. Since each task is a binary classification problem, we use the *binary cross entropy* loss. The feed-forward neural network is then trained for 4000 epochs with an early stopping approach.

We tune the hyperparameters on the validation dataset. The following hyperparameters were tuned:

- **Learning rate**: The amount that updates weight in feed-forward neural network.
- **Embedding size**: The size of the generated embedding vector.
- **Number of epochs**: The number of iterations in the training loop to generate embedding vectors.
- **Number of negative triples**: The number of negative triples to generate for each true triple.

We first tune the learning rate of the feed-forward neural network. The following values are tried: 0.01, 0.001, 0.0001 and 0.00001. In this tuning step, we set the embedding size to 200, the number of epochs during the training phase of embedding vectors to 250, and the number of negative samples for each true triple to 2.

Table 6.1 AUPR scores obtained on the validation dataset for different learning rates for each method and task are listed.

Method	**Learning Rate**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	0.01	6.50% ±1.30	11.10% ±4.73%	45.33 ±4.59	7.58% ±0.26
	0.001	**10.50% ±5.14**	**17.39% ±5.31**	48.32% ±5.62	**9.35% ±3.58**
	0.0001	9.95% ±6.44	15.15% ±8.52%	**48.74% ±6.32**	8.95% ±2.86
	0.00001	7.49% ±1.15	10.34% ±0.91%	45.06 ±0.79	7.79% ±0.55
ComplEx	0.01	34.32% ±5.26	34.85% ±3.84	56.60 ±2.26	14.81% ±2.27
	0.001	**36.95% ±2.97**	35.62% ±3.81	57.14% ±1.10	14.31% ±1.76
	0.0001	36.07% ±3.99	**37.15% ±3.97**	57.95% ±1.25	15.77% ±2.66
	0.00001	35.83% ±3.54	36.63% ±3.70	**63.38% ±1.44**	**20.84% ±2.07**
DistMult	0.01	33.21% ±2.74	31.83% ±2.61	55.65% ±2.47	14.41% ±1.96
	0.001	36.50% ±4.99	34.40% ±2.76	57.16% ±1.39	14.57% ±2.40
	0.0001	**39.62% ±4.96**	37.03% ±2.99	58.33% ±1.33	16.01% ±2.51
	0.00001	39.54% ±6.87	**39.57% ±3.74**	**62.07% ±2.03**	**19.64% ±1.63**

Table 6.1 lists the learning rate that achieves the highest AUPR score for each method and task. In general, the lower learning rates achieve higher performance.

In the second step of hyperparameter tuning, we tune the embedding size. For each method and task, we evaluated the following embedding sizes: 100, 200, and 400. The optimal learning rate, which was found in Table 6.1, will be used in the rest of the hyperparameter tuning for each method and task.

Table 6.2 AUPR scores obtained over the validation set for different embedding sizes for each method and task.

Method	**Size**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	100	9.99% ±3.49	15.40% ±0.33	46.54% ±2.66	7.67% ±0.44
	200	11.18% ±3.52	17.07% ±0.98	**49.53% ±6.57**	8.77% ±3.03
	400	**17.16% ±1.09**	**22.92% ±1.76**	48.41% ±1.18	**10.34% ±4.03**
ComplEx	100	31.09% ±3.56	**39.01% ±3.28**	53.99% ±19.42	13.84% ±10.3
	200	36.16% ±2.90	36.04% ±3.64	**63.02% ±0.89**	**20.62% ±2.32**
	400	**38.88% ±3.81**	38.69% ±5.09	61.84% ±2.94	19.09% ±3.90
DistMult	100	39.48% ±1.49	26.13% ±17.66	53.52% ±17.64	13.11% ±11.15
	200	**39.95% ±4.07**	37.89% ±4.59	**63.42% ±1.38**	**19.59% ±1.34**
	400	39.55% ±4.86	**43.97% ±2.69**	62.53% ±2.22	18.97% ±3.1

In Table 6.2, you can find the embedding size that achieves the highest AUPR score for each method and task. In general, 200 and 400 embedding size achieves higher performance.

In the third step of hyperparameter tuning, the number of epochs will be tuned. For each method and task, the following number of epochs will be used: 100, 250, and 500. Optimal learning rate and embedding size, which were found in Table 6.1 and Table 6.2, will be used for each method and task.

Table 6.3 Results are AUPR score and obtained by using validation dataset for different number of epochs for each method and task.

Method	**Epoch**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	100	14.85% ±1.26	21.47% ±1.66	47.63% ±6.16	10.61% ±3.23
	250	**17.16% ±1.09**	**22.92% ±1.76**	**49.53% ±6.57**	10.34% ±4.03
	500	13.10% ±1.29	18.13% ±1.11	48.19% ±6.09	**11.12% ±3.90**
ComplEx	100	**39.03% ±2.84**	**41.24% ±1.58**	62.72% ±0.78	19.9% ±1.36
	250	38.88% ±3.81	39.01% ±3.28	**63.02% ±0.89**	**20.62% ±2.32**
	500	38.78% ±4.24	39.35% ±3.49	62.76% ±0.88	18.66% ±7.90
DistMult	100	**40.06% ±2.89**	**45.78% ±2.70**	63.11% ±0.83	**19.01% ±1.12**
	250	39.95% ±4.07	43.97% ±2.69	**63.42% ±1.38**	18.36% ±6.63
	500	38.73% ±4.28	42.77% ±3.84	63.26% ±1.26	18.06% ±6.78

In Table 6.3, you can find the number of the epoch that achieves the highest AUPR score for each method and task. In general, 100 and 2500 epoch achieves higher performance. Embedding vectors which were trained on 500 epochs, tend to overfit the input graph.

In the last step of hyperparameter tuning, the number of negative triples will be tuned. For each method and task, the following number of negative triples will be used: 2, 5, and 10. Optimal learning rate, embedding size and number of epoch, which were found in Table 6.1, Table 6.2 and Table 6.3, will be used for each method and task.

Table 6.4 Following results are AUPR score and obtained by using validation dataset for different number of negative triples for each method and task.

Method	**Negative Triples**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	2	**17.16% ±1.09**	**22.92% ±1.76**	49.53% ±6.57	11.12% ±3.90
	5	16.12% ±1.59	21.65% ±2.53	49.07% ±8.99	**11.67% ±3.10**
	10	15.82% ±1.17	19.56% ±1.31	**50.06% ±6.31**	11.31% ±4.10
ComplEx	2	**39.03% ±2.84**	**41.24% ±1.58**	**63.02% ±0.89**	**20.62% ±2.32**
	5	35.31% ±2.97	39.06% ±1.82	62.82% ±1.26	20.19% ±2.15
	10	37.62% ±1.67	38.80% ±2.66	61.95% ±1.06	20.27% ±1.43
DistMult	2	**40.06% ±2.89**	**45.78% ±2.7**	**63.42% ±1.38**	19.01% ±1.12
	5	37.08% ±2.59	43.93% ±2.56	62.82% ±1.22	19.57% ±1.05
	10	37.96% ±3.13	42.88% ±2.31	62.43% ±1.84	**19.64% ±1.68**

In Table 6.4, higher performance was achieved when the number of negative triples was 2. The increasing number of negative triples tends to underfit the input graph.

Lastly, optimal hyperparameters were used to evaluate the performance of the test dataset. Train and validation datasets were concatenated and used for training.

Table 6.5 Results of each method and task on test dataset. Train and validation sets were concatenated and used as input for training.

Method	**Tasks**			
	In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	16.62% ±0.92	22.10% ±0.83	53.91% ±0.77	12.33% ±0.75
ComplEx	**48.34% ±0.81**	46.74% ±1.61	65.04% ±0.32	17.46% ±0.58
DistMult	46.84% ±1.32	**49.84% ±0.92**	**65.25% ±0.26**	**18.43% ±0.57**

In Table 6.5 except *In - ICU Mortality* task, higher AUPR scores were obtained by using *DistMult* as the embedding generation method. Embedding vectors generated with *TransE* had poor performance in all tasks.

6.2 Integrating Measurements & Vital Signs

Due to the nature of the knowledge graphs, variables with continuous values cannot be represented as nodes in the graph. Vital signs and measurements are examples of this type of variable. They contain precious information about the patients' health conditions. There exist several works (Harutyunyan et al., 2019), (Purushotham et al., 2018), (Hyland et al., 2020), (Wang et al., 2020) that uses measurements and vitals signs of patients to increase the performance of their predictor models. In this thesis, two different sets of measurements were used.

Firstly, 16 measurements were selected. These measurements were a subset of the measurements which were used in Harutyunyan et al. (2019). In the rest of this thesis, this set of measurements will be called *Set A*. Measurement names and details in *Set A* is shared in Table 6.6.

Table 6.6 Names and short descriptions of the measurements in Set A.

Measurement	Notes
Diastolic blood pressure	Recorded hourly, highly available for all patients.
Fraction inspired oxygen	Recorded in every 1 to 3 hour, not available for most patient.
Glascow coma scale eye opening	Recorded in every 4 hour, not available for most patient.
Glascow coma scale motor response	Recorded in every 4 hour, not available for most patient.
Glascow coma scale verbal response	Recorded in every 4 hour, not available for most patient.
Glascow coma scale total	Recorded in every 4 hour, not available for most patient.
Glucose	Cannot find pattern for recording, but highly available for most of the patient.
Heart Rate	Recorded hourly, highly available for all patients.
Height	Recorded once, not available for most patients.
Mean blood pressure	Recorded hourly, highly available for all patients.
Oxygen saturation	Recorded hourly, highly available for all patients.
Respiratory rate	Recorded hourly, highly available for all patients.
Systolic blood pressure	Recorded hourly, highly available for all patients.
Temperature	Recorded in every 3 to 4 hour, available for most patients.
Weight	Recorded once, available for most patients.
pH	Recorded in every 5 hours, available for some patients.

Secondly, 102 measurements were selected. The name of these measurements is shared in Appendix as Table A. These measurements were a subset of the measure-

ments which were used in Wang et al. (2020). In the rest of this thesis, this set of measurements will be called *Set B*.

6.2.1 Extracting Statistical Features

When a patient stays in ICU, patients' vital signs and measurements are recorded in an interval. As a result, these records create time series. However, in MIMIC-III, most of the vital signs are recorded irregularly. Thus, time series become non-evenly spaced. There exist certain ways to handle non-evenly spaced time series, and the first way is to extract statistical features from time series. For each measurement in Table 6.6 and Table A, following statistical features are generated:

- Minimum
- Maximum
- Median
- Standard Deviation
- Frequency

In the process of statistical feature generation, measurements were scaled with *Robust Scaler*. In addition, only the first 24 hours of a patients' data were used. In the architecture, generated feature vector was concatenated to the patient embedding vector. Obtained input vector was fed into a feed-forward neural network to perform prediction tasks.

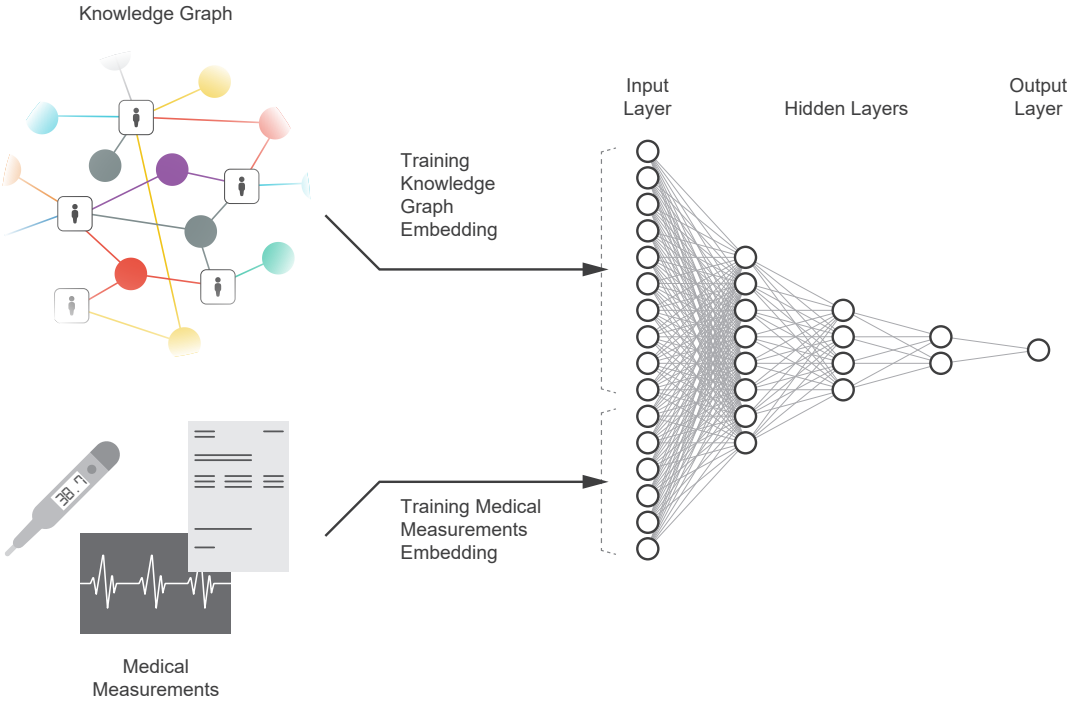

Figure 6.2 Feedforward neural network architecture. Knowledge graph embeddings were concatenated with statistical feature vector and used as input to feed-forward neural network

Since optimal values for embedding size, number of negative triples and number of epoch to generate knowledge graph embeddings were found in Section 6.1, hyper-parameter tuning will be only applied to the learning rate of the feed-forward part. Similar to Section 6.1, following values will be used: 0.01, 0.001, 0.0001 and 0.00001.

Table 6.7 Results of the concatenation of embedding vectors and statistical features for *Set A*. Following results are AUPR score and obtained by using validation dataset for different learning rates for each method and task.

Method	**Learning Rate**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	0.01	**42.84% ±2.96**	**42.22% ±3.36**	58.81% ±2.20	**18.18% ±3.03**
	0.001	36.12% ±2.99	35.66% ±3.87	58.71% ±1.53	14.08% ±1.97
	0.0001	39.82% ±4.15	39.89% ±2.57	**62.75% ±2.13**	14.08% ±1.97
	0.00001	38.3% ±2.28	41.40% ±1.35	59.64% ±6.79	15.05% ±1.61
ComplEx	0.01	47.92% ±4.61	42.80% ±3.19	57.53% ±2.70	17.19% ±2.90
	0.001	42.72% ±3.28	41.98% ±3.73	60.22% ±1.23	16.44% ±2.34
	0.0001	43.95% ±3.65	**46.49% ±2.87**	61.22% ±1.11	16.15% ±1.74
	0.00001	**48.27% ±2.55**	36.55% ±22.55	**63.70% ±1.14**	**19.96% ±6.14**
DistMult	0.01	42.08% ±5.36	44.80% ±6.00	57.03% ±3.24	17.10% ±2.03
	0.001	39.56% ±3.82	44.85% ±3.27	59.83% ±2.09	15.85% ±1.13
	0.0001	**42.99% ±4.73**	45.13% ±2.21	61.14% ±1.56	16.62% ±1.76
	0.00001	40.89% ±4.69	**49.24% ±1.72**	**63.53% ±1.00**	**19.34% ±5.62**

In Table 6.7, you can find the learning rate that achieves the highest AUPR score for each method and task. In general, a lower learning rate achieves higher performance on $DistMult$ and $ComplEx$. However, in $TransE$, higher learning rates had higher AUPR scores. In the next step, optimal learning rate values will be used for each method and task on the test dataset.

Table 6.8 Results of the concatenation of embedding vectors and statistical features for Table 6.6. Following results are AUPR score and obtained by using test dataset by using optimal hyperparameter values for each method and task.

Method	**Tasks**			
	In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	49.48% ±2.17	51.65% ±2.12	65.61% ±0.60	18.99% ±1.25
ComplEx	**58.26% ±1.85**	59.77% ±2.39	65.88% ±1.48	19.49% ±0.71
DistMult	57.13% ±3.88	**61.81% ±2.88**	**66.36% ±0.94**	**19.88% ±1.09**

As results are shared in Table 6.8, in comparison to Section 6.1, performance in all method and task had significant gain. Similar to previous chapter, in each task,

TransE had the lowest AUPR score. And, except *In - ICU Mortality* task, *DistMult* outperforms *ComplEx*.

Similar to measurements in Table 6.6, learning rate will be tuned for the measurements in Table A by trying the same set of values: 0.01, 0.001, 0.0001 and 0.00001.

Table 6.9 Results of the concatenation of embedding vectors and statistical features for Table A. Following results are AUPR score and obtained by using validation dataset for different learning rates for each method and task.

Method	**Size**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	0.01	35.57% ±14.38	35.19 ±20.07	55.25 ±5.37	16.65% ±3.62
	0.001	37.55% ±4.04	38.53% ±3.74	57.81% ±2.00	14.33% ±1.44
	0.0001	36.16% ±3.07	38.79% ±2.99	59.67% ±1.07	13.66% ±1.66
	0.0001	**37.56% ±1.89**	**44.04% ±1.56**	**63.18% ±0.95**	**18.01% ±3.15**
ComplEx	0.01	33.99% ±21.66	**44.39% ±7.35**	57.47% ±1.28	16.05% ±6.16
	0.001	39.56% ±4.70	40.54% ±3.28	60.39% ±1.53	15.94% ±1.59
	0.0001	41.37% ±2.81	41.89% ±2.91	61.85% ±0.85	15.87% ±1.94
	0.0001	**42.61% ±3.74**	42.95% ±4.03	**64.04% ±1.56**	**19.30% ±1.55**
DistMult	0.01	**40.02% ±8.72**	43.74% ±12.41	57.36% ±2.89	16.93% ±5.2
	0.001	38.19% ±6.25	43.86% ±2.14	60.15% ±1.69	15.66% ±2.48
	0.0001	39.99% ±3.42	44.40% ±2.85	61.02% ±1.17	15.95% ±1.15
	0.0001	39.51% ±5.13	**48.16% ±2.22**	**64.31% ±1.29**	**19.01% ±2.56**

In Table 6.9, you can find the learning rate that achieves the highest AUPR score for each method and task. Unlike the measurements in Table 6.6, smaller learning rates had higher AUPR score in *TransE*. Similar to measurements in Table 6.6, smaller learning rate achieves higher performance on *DistMult* and *ComplEx* in general. In the next step, optimal learning rate values will be used for each method and task on the test dataset.

Table 6.10 Results of the concatenation of embedding vectors and statistical features for *Set B*. Following results are AUPR score and obtained by using test dataset by using optimal hyperparameter values for each method and task.

Method	**Tasks**			
	In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	47.62% ±1.76	52.48% ±1.71	64.86% ±0.64	19.34% ±0.77
ComplEx	**51.72% ±1.42**	52.90% ±3.14	65.82% ±0.84	**19.70% ±0.83**
DistMult	46.38% ±7.75	**56.78% ±1.64**	**66.05% ±0.88**	19.66% ±0.77

As results are shared in Table 6.10, in all tasks, using measurements in Table 6.6 achieved higher AUPR score. Since measurements in Table A contain the high number of measurements, the size of the input vector of the feed-forward part becomes large. In neural networks, using more features does not always lead to higher performance. It may create a complex model that causes to overfit. Thus, using measurements in Table A affected performance negatively.

6.2.2 Separate Network for Measurements & Vital Signs

In Section 6.2.1, the patient embedding vector was concatenated with the statistical feature vector before feeding into the neural network. In this section, both the embedding vector and statistical feature vector will be fed into a separate neural network, and then outputs will be concatenated to be fed into several hidden layers.

In the first experiment of this section, embedding vectors and statistical features will be fed into a separate network with 1 layer. Outputs of networks will be concatenated and fed into two more layers. Lastly, outputs will be fed into a single neuron in the output layer. This architecture can be visualized in Figure 6.3.

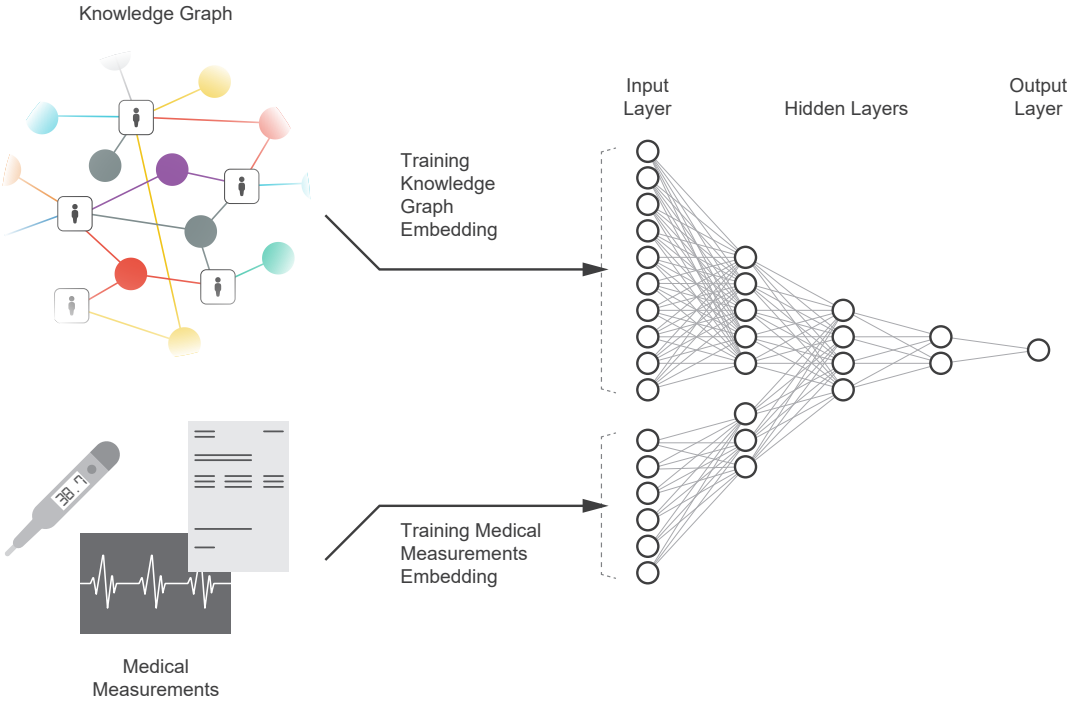

Figure 6.3 Feedforward neural network diagram. Embedding vectors and statistical features are feed into separate layers, then outputs were concatenated and feed into hidden layers.

Since optimal values for embedding size, the number of epochs, and number of negative triples were found in the first section of this chapter, and hyperparameter tuning will only be applied to the learning rate. Similar to the previous section, the following values will be used: 0.01, 0.001, 0.0001 and 0.00001.

Table 6.11 Results of feeding embedding vectors and statistical features into separate networks for 1 layer. Outputs were concatenated and fed into 2 more layers. Following results are AUPR score and obtained by using validation dataset for different learning rates for each method and task.

Method	**Size**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	0.01	40.08% ±3.58	**47.04% ±3.18**	**64.18% ±1.34**	**20.06% ±1.82**
	0.001	**42.14% ±2.29**	44.02% ±1.67	63.34% ±0.96	18.85% ±0.90
	0.0001	41.03% ±2.87	43.27% ±2.93	62.40% ±0.91	19.06% ±0.90
	0.00001	39.31% ±3.53	42.06% ±3.26	61.17% ±1.36	19.08% ±1.2
ComplEx	0.01	49.76% ±3.36	**49.48% ±2.15**	**66.55% ±1.18**	**22.14% ±2.15**
	0.001	51.72% ±3.46	49.4% ±2.72	66.14% ±0.80	22.07% ±1.76
	0.0001	**52.85% ±2.67**	47.98% ±2.21	65.22% ±1.09	22.05% ±1.78
	0.00001	52.50% ±3.87	46.54% ±1.85	64.46% ±1.30	21.79% ±1.88
DistMult	0.01	47.0% ±3.07	50.61% ±3.17	**66.61% ±1.17**	21.43% ±1.35
	0.001	47.44% ±2.53	51.70% ±1.37	65.92% ±0.78	21.49% ±2.07
	0.0001	**47.57% ±2.72**	**51.98% ±2.00**	65.32% ±1.02	**21.71% ±1.62**
	0.00001	47.04% ±3.18	51.95% ±2.35	64.46% ±1.11	21.47% ±1.49

In the Table 6.11, you can find the learning rate that achieves the highest AUPR score for each method and task. In general, a higher learning rate achieves higher performance on *Transe* and *ComplEx*. However, in *DistMult*, generally, lower learning rates had higher AUPR scores. In the next step, optimal learning rates will be used for each method and task on the test dataset.

Table 6.12 Results of feeding embedding vectors and statistical features into separate networks for 1 layer. Following results are AUPR score and obtained by using test dataset by using optimal hyperparameter values for each method and task.

Method	**Tasks**			
	In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	50.00% ±4.67	50.63% ±4.06	59.06% ±1.73	19.01% ±1.26
ComplEx	55.90% ±0.89	**59.06% ±1.73**	**67.42% ±1.80**	19.14% ±0.80
DistMult	**56.30% ±1.05**	58.74% ±1.62	66.96% ±1.23	**19.85% ±1.32**

As results on test set are shared in Table 6.12, in each task, *TransE* had the lowest AUPR score. *ComplEx* had highest AUPR scores in *LoS > 3 Days* and *In - Hos-*

pital Mortality tasks. By contrast, *DistMult* had highest AUPR scores in *In - ICU Mortality* and *LoS > 7 Days* tasks.

In the second experiment of this section, embedding vectors and statistical features will be fed into separate networks with 2 layers. Outputs of networks will be concatenated and fed into 1 more layer. Lastly, outputs will be fed into a single neuron in the output layer.

Since optimal values for embedding size, the number of epochs and the number of negative triples was found in Section 6.1, hyperparameter tuning will only be applied to the learning rate. Similar to previous experiments, the following values will be used: 0.01, 0.001, 0.0001, and 0.00001.

Table 6.13 Results of feeding embedding vectors and statistical features into separate networks for 2 layers. Outputs were concatenated and fed into 1 more layer. Following results are AUPR score and obtained by using validation dataset for different learning rates for each method and task.

Method	**Size**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	0.01	39.21% ±4.32	42.87% ±2.42	**63.03% ±1.52**	19.60% ±1.00
	0.001	**39.47% ±3.05**	**43.82% ±3.29**	62.81% ±1.16	**19.64% ±1.69**
	0.0001	37.58% ±2.15	42.56% ±2.38	61.87% ±0.97	19.41% ±1.39
	0.00001	37.11% ±2.80	40.50% ±1.79	60.36% ±1.19	17.13% ±5.95
ComplEx	0.01	**50.75% ±2.53**	48.36% ±4.05	**65.42% ±1.06**	**22.25% ±1.36**
	0.001	50.34% ±2.91	**49.53% ±2.61**	65.03% ±0.97	21.53% ±1.49
	0.0001	48.63% ±2.46	49.07% ±5.08	64.46% ±1.41	21.08% ±1.62
	0.00001	48.57% ±2.17	47.93% ±6.4	64.24% ±1.46	20.80% ±1.90
DistMult	0.01	48.57% ±2.17	50.00% ±2.14	**65.9% ±1.19**	20.65% ±1.22
	0.001	**49.00% ±2.15**	**50.76% ±2.4**	65.27% ±0.31	20.70% ±1.72
	0.0001	47.56% ±2.13	49.20% ±1.94	64.60% ±0.81	**20.84% ±1.57**
	0.00001	46.80% ±2.67	48.51% ±1.97	64.12% ±0.85	20.59% ±1.70

In Table 6.13, you can find the learning rate that achieves the highest AUPR score for each method and task. Similarly to previous experiment, higher learning rate achieves higher performance on *TransE* and *ComplEx*. Except *LoS > 7 Days* task, *DistMult* also performed better with higher learning rates. In the next step, optimal learning rates will be used for each method and task on the test dataset.

Table 6.14 Results of feeding embedding vectors and statistical features into separate networks for 2 layers. Following results are AUPR score and obtained by using test dataset by using optimal hyperparameter values for each method and task.

Method	**Tasks**			
	In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	49.08% ±0.97	50.71% ±2.97	64.76% ±1.63	**18.94% ±1.05**
ComplEx	**55.36% ±3.94**	**57.02% ±2.70**	**66.22% ±0.99**	18.58% ±1.17
DistMult	53.80% ±2.64	55.85% ±2.86	66.02% ±0.94	18.47% ±0.79

As results are shared in Table 6.14, except *LoS > 7 Days* task, *ComplEx* achieved higher AUPR score. Surprisingly, *TransE* outperformed *ComplEx* and *DistMult* in *LoS > 7 Days* task. Consequently, in comparison to using 2 layers as a separate network, using 1 layer as a separate network had higher AUPR scores in all tasks.

6.2.3 Time Series Implementation

As stated in Section 6.2.1, measurements and vital signs were recorded irregularly. Thus treating them as time series becomes harder. A possible solution to represent irregularly recorded data as artificially regular data is dividing data into chunks. In this experiment, patients' measurements and vital signs were divided into hourly chunks. Since only the first 24 hours of a patients' data was used, data is divided into 24 chunks. For each measurement and vital sign, each chunk was represented by the median value of the records in that chunk. If a chunk does not have any records, then the median value of the rest of the chunks will be used to represent that chunk.

Since measurements in Table 6.6 is used, each chunk was represented by a vector with a length of 16. In total, the measurements of each patient was represented by a two-dimensional array. The length of the first dimension is 24, which is the number of chunks. The length of the second dimension is 16, which is the number of measurements in each chunk.

In this subsection, three different setups were used. In all setups, measurements and embedding vectors were fed into separate networks. Embedding vectors were fed into a network with 1 layer. Conv 1D, LSTM and Transformer were used in the separate network for measurements to learn temporal representations. Outputs

were concatenated and fed into 2 more layers.

In the first setup, time-series data was fed into Conv 1D. For hyperparameter tuning, the number of filters, size of each filter and learning rate were tuned by using values in Table 6.15.

Table 6.15 Hyperparameter tuning for Conv 1D. This process was divided into two steps. In the first step, for each task and method, *filters* and *kernel_size* hyperparameters were tuned at the same time by using grid search. During this step, *learning_rate* was fixed at 0.001. In the second step, optimal values for *filters* and *kernel_size* were used to tune *learning_rate*. All the experiments were applied on validation dataset.

Hyperparameter	Search Space	Description
filters	[1, 3, 5]	Number of filters
kernel_size	[3, 5, 7]	Size of the kernel
learning_rate	[0.01, 0.001, 0.0001, 0.00001]	Learning rate

In the second setup, time series data was feed into LSTM. For hyperparameter tuning, number of units and learning rate were tuned by using values in Table 6.16.

Table 6.16 Hyperparameter tuning for LSTM. This process was divided into two steps. In the first step, for each task and method, *units* hyperparameter was tuned. During this step, *learning_rate* was fixed at 0.001. In the second step, optimal value for *units* were used to tune *learning_rate*. All the experiments were applied on validation dataset.

Hyperparameter	Search Space	Description
units	[8, 16, 32]	Dimension of inner cell
learning_rate	[0.01, 0.001, 0.0001, 0.00001]	Learning rate

In the third setup, time series data was feed into Transformer. For hyperparameter tuning, number of multi-attention heads, size of each attention head, number of encoder blocks and learning rate were tuned by using values in Table 6.17.

Table 6.17 Hyperparameter tuning for Transformer. This process was divided into three steps. In the first step, for each task and method, *num_heads* and *head_size* hyperparameters were tuned at the same time by using grid search. During this step, *num_encoder_blocks* was fixed at 2 and *learning_rate* was fixed at 0.001. In the second step, optimal values for *num_heads* and *head_size* were used to tune *num_encoder_blocks*. In the third step, optimal values for *num_heads*, *head_size* and *num_encoder_blocks* were used to tune *learning_rate*. All the experiments were applied on validation dataset.

Hyperparameter	Search Space	Description
num_heads	[2, 4, 6]	Number of attention heads
head_size	[32, 64, 128]	Size of each attention head
num_encoder_blocks	[2, 3, 4]	Number of encoder blocks
learning_rate	[0.01, 0.001, 0.0001, 0.00001]	Learning rate

In Table 6.18, you can find results on the test dataset of these three setups for each method and task pair by using optimal values for each hyperparameter.

Table 6.18 Results of feeding embedding vectors and time series measurements into separate networks for 1 layer. Performance of using Conv 1D, LSTM and Transformer for measurements were compared. Following results are AUPR score and obtained on test dataset by using optimal hyperparameter values for each method and task.

Method	**Implementation**	**Tasks**			
		In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
TransE	Conv 1D	48.54% ±2.21	50.35% ±1.99	63.09% ±0.82	16.17% ±0.46
	LSTM	46.54% ±2.00	49.87% ±1.76	63.43% ±1.88	17.09% ±1.03
	Transformer	44.14% ±2.17	46.07% ±4.14	61.91% ±0.76	15.90% ±0.91
ComplEx	Conv 1D	**57.11% ±3.33**	56.62% ±2.18	66.32% ±0.82	17.78% ±0.82
	LSTM	53.88% ±5.44	58.5% ±2.17	**66.57% ±1.04**	18.82% ±0.95
	Transformer	55.33% ±2.56	55.52% ±2.34	66.30% ±0.84	18.31% ±0.90
DistMult	Conv 1D	56.94% ±4.10	**60.38% ±3.65**	66.15% ±0.79	18.09% ±1.30
	LSTM	56.59% ±1.53	59.84% ±1.20	66.50% ±0.75	18.88% ±0.72
	Transformer	56.09% ±1.60	59.59% ±1.56	66.32% ±0.28	**19.11% ±0.45**

Table 6.18 shows that results are similar to previous experiments. *TransE* had lowest AUPR scores in all tasks. For *In - ICU Mortality* and *LoS > 3 Days* tasks, *ComplEx* had higher performance by using Conv 1D and LSTM respectively. On the other hand, *DistMult* had better performance on *In - Hospital Mortality* and *LoS > 7 Days* tasks by using Conv 1D and Transformer respectively.

6.3 Comparison with MIMIC-Extract

In Section 6.1 and Section 6.2, a certain number of different setups were used. From all experiments, for each method and task, the highest AUPR score is shared in Table 6.19. In addition, AUPR scores for all tasks in MIMIC-Extract are also shared in Table 6.19.

Table 6.19 Comparison of AUPR scores with MIMIC - Extract. For each method and task, highest AUPR score is shared. Results are generated on test dataset.

Method	**Tasks**			
	In - ICU Mortality	In - Hospital Mortality	LoS > 3 Days	LoS > 7 Days
MIMIC - Extract	50.90	53.20	**68.5**	19.50
TransE	50.00 ±4.67	52.48 ±1.71	65.61 ±0.60	19.34 ±0.77
ComplEx	**58.26 ±1.85**	59.77 ±2.39	67.42 ±1.80	19.70 ±0.83
DistMult	57.13 ±3.88	**61.81 ±2.88**	66.96 ±1.23	**19.88 ±1.09**

As shared in Table 6.19, except *LoS > 3 Days* task, proposed solution achieved superior performance.

7. CONCLUSION

In this work, we explore representing patients with embedding vectors learned in EHRs knowledge graphs. These learned vectors are non-sparse constructs by design that encode useful patient medical information. We explore the usefullness of this representation in downstream prediction tasks at ICU. These tasks include the mortality prediction and length of stay predictions. These tasks have highly imbalanced class labels. Therefore, to deal with the imbalance we resort to cost sensitive learning. The data contain features across a time window andwe ensure that features that can leak information about future are excluded. For example to prevent data leakage, the information on diagnoses of a particular stay is not included as the diagnoses codes in the dataset were recorded at the end of the hospital stay.

In our work we employed and compared three different knowledge graph embedding methods. *TransE* consistently underperformed in comparison to *DistMult* and *ComplEx* tools in the ICU prediction tasks. On the other hand, *DistMult* and *ComplEx* performances were similar to each other.

In our models the demographic and categorical features are represented in the knowledge graph, but the patient measurements and vital signs are treated separately. We derive features that represent signals over a period of time, as explained in Section 6.2.1. Extracted feature vectors are either used directly or fed to a feed-forward neural network to obtain more complex features learned on them (Section 6.2.2). We also explored treating the measurements data as time series in Section 6.2.3. Since most of the measurements were recorded irregularly, measurements were aggregated into hourly buckets to have denser representations. Except for the *LoS > 3 Days* task, representing these measurements with simple aggregation features yielded the highest performance. For *LoS > 3 Days* task, the set-up proposed in Section 6.2.2 resulted with the highest performance. Treating measurements as time series in Section 6.2.3 did not turn out to be beneficial and it actually reduced the model performance. Highest AUPR scores for each method and task were compared with MIMIC-Extract in Section 6.3. The proposed knowledge based representation of patients yielded superior performance in three of the four different classification

tasks.

Measurements and vital signs contain highly valuable information, thus using these information have showed a significant change in performance. We observed that increasing the number of measurements does not always translate to performance improvement. Incorporating more measurements also prolongs the training times and increases model complexity. Using separate network for measurements and embeddings in Section 6.2.2 did not improve the performance either. Treating measurements as time series in Section 6.2.3 did not improve the performance since measurements were recorded sporadically. In addition, some of the measurements were recorded sparsely in time.

7.1 Limitations and Future Work

The presented approach has some limitations. Not all the information in the MIMIC-III data have been used during the generation of knowledge graph. Currently, we ignore the doctor and nurse notes, which may contain valuable information about the patient. These notes can be used to extract information, and the information can be represented as nodes. In addition, caregivers can be represented as nodes in a graph.

Secondly, due to nature of knowledge graphs, numerical values of the measurements and vitals signs are not included in knowledge graphs. It is not possible to assign feature vectors to nodes in a knowledge graph, thus some valuable information have been inevitably left-out. This situation has posed a similar limitation in representing the prescribed medicine. The duration and the dosage information for the medicine could not be represented in the graph. One way to incorporate this information could be adapting a graph neural network (GNN) approach. Since the graph in this thesis is a heterogeneous graph and contains different type of nodes, heterogeneous based GNN's can be used to extract embedding vectors.

Due to the basic construction of knowledge graphs, if a graph update occurs, (if nodes are added or removed), the whole graph should be retrained. In our case, when a new patient arrives to ICU, whole graph should be trained from scratch to learn the embedding vector of the new patient. In the recent years, methods to learn embeddings for new nodes without training from scratch have been proposed. Wewer et al. (2021) uses neighbor information to initialize embeddings for new nodes. By

fixing the embeddings for old nodes, only the new node embeddings are optimized. Such incremental training approaches could be useful to deploy the system in real life.

Finally, the training steps for node embedding and prediction task are conducted separately. These parts can be combined in an end-to-end training fashion. The hyperparameter tuning on an end-to-end architecture may improve the performance which we leave as possible future work.

BIBLIOGRAPHY

AC06953431, A. (2008). *Health informatics-Electronic health record communication-Part 1: Reference model*. ISO.

Aliyu, I., Kana, A., and Aliyu, S. (2020). Development of knowledge graph for university courses management. *International Journal of Education and Management Engineering*, 10(2):1.

Beaulieu-Jones, B. K., Greene, C. S., et al. (2016). Semi-supervised learning of the electronic health record for phenotype stratification. *Journal of biomedical informatics*, 64:168–178.

Birkhead, G. S., Klompas, M., and Shah, N. R. (2015). Uses of electronic health records for public health surveillance to advance public health. *Annual review of public health*, 36:345–359.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Bonomi, S. (2016). The electronic health record: a comparison of some european countries. In *Information and Communication Technologies in Organizations and Society*, pages 33–50. Springer.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, pages 1–9.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Brown, S.-A. (2016). Patient similarity: emerging concepts in systems and precision medicine. *Frontiers in physiology*, 7:561.

Caballero Barajas, K. L. and Akella, R. (2015). Dynamically modeling patient's health state from electronic medical records: A time series approach. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78.

Cao, Y., Wang, X., He, X., Hu, Z., and Chua, T.-S. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Charles, D., Gabriel, M., and Furukawa, M. F. (2013). Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2014. *ONC data brief*, 9:1–9.

Cheng, Y., Wang, F., Zhang, P., and Hu, J. (2016). Risk prediction with electronic health records: A deep learning approach. In *Proceedings of the 2016 SIAM international conference on data mining*, pages 432–440. SIAM.

Daskalakis, D. C. (2017). The electronic health record and patient portals in hiv medicine: pushing the boundaries of current ethics and stigma. *Cambridge Quarterly of Healthcare Ethics*, 26(2):332–336.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

Dušek, O. and Jurčíček, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

Esteban, C., Schmidt, D., Krompaß, D., and Tresp, V. (2015). Predicting sequences of clinical events by using a personalized temporal latent embedding model. In *2015 International Conference on Healthcare Informatics*, pages 130–139. IEEE.

Evans, R. S. (2016). Electronic health records: then, now, and in the future. *Yearbook of medical informatics*, 25(S 01):S48–S61.

Ferrão, J. C., Oliveira, M. D., Gartner, D., Janela, F., and Martins, H. M. (2021). Leveraging electronic health record data to inform hospital resource management. *Health Care Management Science*, 24(4):716–741.

Fries, J. A. (2016). Brundlefly at semeval-2016 task 12: Recurrent neural networks vs. joint inference for clinical temporal information extraction. *arXiv preprint arXiv:1606.01433*.

Gentimis, T., Ala'J, A., Durante, A., Cook, K., and Steele, R. (2017). Predicting hospital length of stay using neural networks on mimic iii data. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 1194–1201. IEEE.

Ghassemi, M., Pimentel, M., Naumann, T., Brennan, T., Clifton, D., Szolovits, P., and Feng, M. (2015). A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. (2020). Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Gupta, M., Phan, T.-L. T., Bunnell, T., and Beheshti, R. (2019). Obesity prediction with ehr data: A deep learning approach with interpretable elements. *arXiv preprint arXiv:1912.02655*.

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159.

Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., and Galstyan, A. (2019). Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):1–18.

Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.

Hillestad, R., Bigelow, J., Bower, A., Girosi, F., Meili, R., Scoville, R., and Taylor, R. (2005). Can electronic medical record systems transform health care? potential health benefits, savings, and costs. *Health affairs*, 24(5):1103–1117.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Hyland, S. L., Faltys, M., Hüser, M., Lyu, X., Gumbsch, T., Esteban, C., Bock, C., Horn, M., Moor, M., Rieck, B., et al. (2020). Early prediction of circulatory failure in the intensive care unit using machine learning. *Nature medicine*, 26(3):364–373.

Jagannatha, A. N. and Yu, H. (2016a). Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2016, page 473. NIH Public Access.

Jagannatha, A. N. and Yu, H. (2016b). Structured prediction models for rnn based sequence labeling in clinical text. In *Proceedings of the conference on empirical methods in natural language processing. conference on empirical methods in natural language processing*, volume 2016, page 856. NIH Public Access.

Jensen, P. B., Jensen, L. J., and Brunak, S. (2012). Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405.

Johnson, A. E., Pollard, T. J., Shen, L., Li-Wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Klompas, M., Cocoros, N. M., Menchaca, J. T., Erani, D., Hafer, E., Herrick, B., Josephson, M., Lee, M., Payne Weiss, M. D., Zambarano, B., et al. (2017). State and local chronic disease surveillance using electronic health record systems. *American journal of public health*, 107(9):1406–1412.

Kuss, O. (2002). Global goodness-of-fit tests in logistic regression with sparse data. *Statistics in medicine*, 21(24):3789–3801.

Lee, D., Jiang, X., and Yu, H. (2020). Harmonized representation learning on dynamic ehr graphs. *Journal of biomedical informatics*, 106:103426.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

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

Liu, J., Zhang, Z., and Razavian, N. (2018a). Deep ehr: Chronic disease prediction using medical notes. In *Machine Learning for Healthcare Conference*, pages 440–464. PMLR.

Liu, Y., Ge, T., Mathews, K. S., Ji, H., and McGuinness, D. L. (2018b). Exploiting task-oriented resources to learn word embeddings for clinical abbreviation expansion. *arXiv preprint arXiv:1804.04225*.

Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Madsen, L. B. (2014). *Data-driven healthcare: how analytics and BI are transforming the industry*. John Wiley & Sons.

McLoughlin, I. P., Garrety, K., and Wilson, R. (2017). *The digitalization of healthcare: Electronic records and the disruption of moral orders*. Oxford University Press.

Mehrabi, S., Sohn, S., Li, D., Pankratz, J. J., Therneau, T., Sauver, J. L. S., Liu, H., and Palakal, M. (2015). Temporal pattern and association discovery of diagnosis codes using deep learning. In *2015 International conference on healthcare informatics*, pages 408–416. IEEE.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Moor, M., Rieck, B., Horn, M., Jutzeler, C. R., and Borgwardt, K. (2021). Early prediction of sepsis in the icu using machine learning: a systematic review. *Frontiers in medicine*, 8:348.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Icml*.

Noel, S., Harley, E., Tam, K. H., Limiero, M., and Share, M. (2016). Cygraph: graph-based analytics and visualization for cybersecurity. In *Handbook of Statistics*, volume 35, pages 117–167. Elsevier.

Oh, S.-H., Lee, S. J., Noh, J., and Mo, J. (2021). Optimal treatment recommendations for diabetes patients using the markov decision process along with the south korean electronic health records. *Scientific reports*, 11(1):1–10.

Pham, T., Tran, T., Phung, D., and Venkatesh, S. (2016). Deepcare: A deep dynamic memory model for predictive medicine. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 30–41. Springer.

Pollard, T. J., Johnson, A. E., Raffa, J. D., Celi, L. A., Mark, R. G., and Badawi, O. (2018). The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):1–13.

Purushotham, S., Meng, C., Che, Z., and Liu, Y. (2018). Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics*, 83:112–134.

Rocheteau, E., Tong, C., Veličković, P., Lane, N., and Liò, P. (2021). Predicting patient outcomes with graph representation learning. *arXiv preprint arXiv:2101.03940*.

Rojas, J. C., Carey, K. A., Edelson, D. P., Venable, L. R., Howell, M. D., and Churpek, M. M. (2018). Predicting intensive care unit readmission with machine learning using electronic health record data. *Annals of the American Thoracic Society*, 15(7):846–853.

Saeed, M., Villarroel, M., Reisner, A. T., Clifford, G., Lehman, L.-W., Moody, G., Heldt, T., Kyaw, T. H., Moody, B., and Mark, R. G. (2011). Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952.

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432.

Shickel, B., Tighe, P. J., Bihorac, A., and Rashidi, P. (2017). Deep ehr: a survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604.

Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C., and Ré, C. (2015). Incremental knowledge base construction using deepdive. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, volume 8, page 1310. NIH Public Access.

Si, Y., Du, J., Li, Z., Jiang, X., Miller, T., Wang, F., Zheng, W. J., and Roberts, K. (2021). Deep representation learning of patient data from electronic health records (ehr): A systematic review. *Journal of Biomedical Informatics*, 115:103671.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016). Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080. PMLR.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., and Chao, L. S. (2019). Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787*.

Wang, S., McDermott, M. B., Chauhan, G., Ghassemi, M., Hughes, M. C., and Naumann, T. (2020). Mimic-extract: A data extraction, preprocessing, and representation pipeline for mimic-iii. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, pages 222–235.

Wang, Y. and Tian, F. (2016). Recurrent residual learning for sequence classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 938–943.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Wewer, C., Lemmerich, F., and Cochez, M. (2021). Updating embeddings for dynamic knowledge graphs. *arXiv preprint arXiv:2109.10896*.

Wu, L. Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., and Weston, J. (2018). Starspace: Embed all the things! In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wu, T., Wang, Y., Wang, Y., Zhao, E., Yuan, Y., and Yang, Z. (2019). Representation learning of ehr data via graph-based medical entity embedding. *arXiv preprint arXiv:1910.02574*.

Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases.

Zhang, S., Liu, L., Li, H., Xiao, Z., and Cui, L. (2016). Mtpgraph: A data-driven approach to predict medical risk based on temporal profile graph. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 1174–1181. IEEE.

Zhang, Y., Yao, Q., Shao, Y., and Chen, L. (2019). Nscaching: simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 614–625. IEEE.

Zhu, W. and Razavian, N. (2021). Variationally regularized graph-based representation learning for electronic health records. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 1–13.

APPENDIX A

Measurement
Alanine aminotransferase
Albumin
Albumin ascites
Albumin pleural
Albumin urine
Alkaline phosphate
Anion gap
Asparate aminotransferase
Basophils
Bicarbonate
Bilirubin
Blood urea nitrogen
Calcium
Calcium ionized
Calcium urine
Cardiac index
Cardiac output fick
Cardiac output thermodilution
Central venous pressure
Chloride
Chloride urine
Cholesterol
Cholesterol hdl
Cholesterol ldl
Co2
Co2 (etco2, pco2, etc.)
Creatinine
Creatinine ascites
Creatinine pleural
Creatinine urine
Diastolic blood pressure
Eosinophils
Fibrinogen
Fraction inspired oxygen
Fraction inspired oxygen set
Glascow coma scale total

Glucose
Heart rate
Height
Hematocrit
Hemoglobin
Lactate
Lactate dehydrogenase
Lactate dehydrogenase pleural
Lactic acid
Lymphocytes
Lymphocytes ascites
Lymphocytes atypica
Lymphocytes body fluid
Lymphocytes percent
Lymphocytes pleural
Magnesium
Mean blood pressure
Mean corpuscular hemoglobin
Mean corpuscular hemoglobin concentration
Mean corpuscular volume
Monocytes
Monocytes csl
Neutrophils
Oxygen saturation
Partial pressure of carbon dioxide
Partial pressure of oxygen
Partial thromboplastin time
Peak inspiratory pressure
Ph
Ph urine
Phosphate
Phosphorous
Plateau pressure
Platelets
Positive end expiratory pressure
Positive end expiratory pressure set
Post void residual

Potassium
Potassium serum
Prothrombin time inr
Prothrombin time pt
Pulmonary artery pressure mean
Pulmonary artery pressure systolic
Pulmonary capillary wedge pressure
Red blood cell count
Red blood cell count ascites
Red blood cell count csf
Red blood cell count pleural
Red blood cell count urine
Respiratory rate
Respiratory rate set
Sodium
Systemic vascular resistance
Systolic blood pressure
Temperature
Tidal volume observed
Tidal volume set
Tidal volume spontaneous
Total protein
Total protein urine
Troponin i
Troponin t
Venous pvo2
Weight
White blood cell count
White blood cell count urine