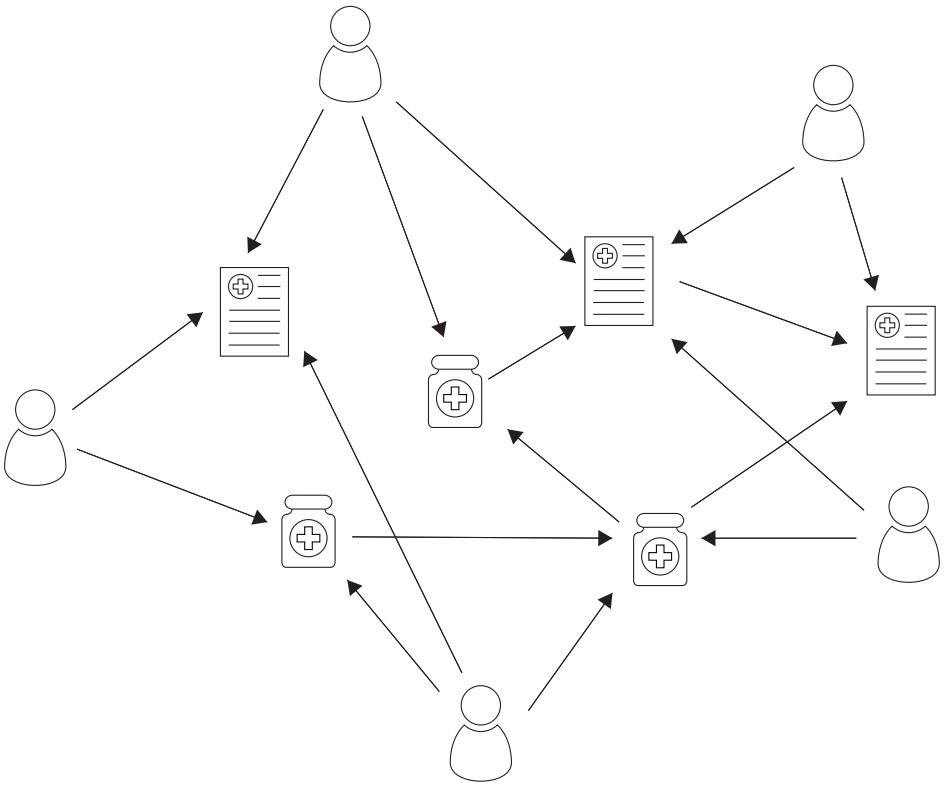
CHALMERS
UNIVERSITY OF TECHNOLOGY



Patient outcome prediction using knowledge graph representation learning

Evaluating translational knowledge graph embedding methods for patient re-admission classification

Master's thesis in Engineering Mathematics and Computational Science

Adnan Fazlinovic & Trilokinath Modi

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Patient outcome prediction using knowledge graph representation learning

Evaluating translational knowledge graph embedding methods for patient re-admission classification

Adnan Fazlinovic
Trilokinath Modi

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2021

Patient outcome prediction using knowledge graph representation learning
Evaluating translational knowledge graph embedding methods for patient re-admission classification
Adnan Fazlinovic
Trilokinath Modi

Supervisor: Magnus Kjellberg, Sahlgrenska University Hospital, digital R&DI
Supervisor: Torbjörn Lundh, Department of Mathematical Sciences
Examiner: Torbjörn Lundh, Department of Mathematical Sciences

Master's Thesis 2021
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Examplary knowledge graph using 5 patients, 3 diagnoses and 3 prescriptions. Links indicates a connection between two entities.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Patient outcome prediction using knowledge graph representation learning
Evaluating translational knowledge graph embedding methods for patient re-admission classification

Adnan Fazlinovic
Trilokinath Modi
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The project focuses on using knowledge graphs in a healthcare setting, classifying patient re-admissions. Knowledge graphs are a type of heterogeneous network consisting of entities and relations. Knowledge graph embedding method aims to generate lower-dimensional latent vector representation of these entities and relations while preserving their relational properties.

The data consists of patient admission details along with their underlying diagnoses, prescriptions consumed and procedures performed. To exploit the true nature of knowledge graphs, more information to the patient graph is added by combining various biomedical databases to obtain a richer set of relationships.

Cleaning patient records and converting the information in more standardized form, as well as gathering information and create a knowledge graph structure in the form of triplets are conducted. The generation of latent vector representations of the entities and relations are done with various embedding methods, where the final phase is to classify patient re-admissions.

The methods investigated achieves to represent entities and relations in latent vector form when evaluating the embeddings based on the proposed loss functions. However, the embeddings generated doesn't supply enough information that can accurately predict the patient readmission status in an extended down-stream fashion. The potential problems could be either of not enough features that explains the variability, not enough rich information regarding the different data sources used, or the effect of class imbalance. A stratified test subset was created from the same excerpt of training data to quantify the results.

Keywords: healthcare, EHR, biomedical ontologies, representation learning, knowledge graphs, embeddings.

Acknowledgements

This thesis work is dedicated to our friends and families. Thank you for all the love and support shown during this period.

We would also like to thank Magnus Kjellberg, our supervisor at Sahlgrenska University Hospital's digital R&DI section, and our academic supervisor and examinor Torbjörn Lundh, for their support and guidance during this master's thesis, as well as giving us the opportunity to conduct this thesis work.

Adnan Fazlinovic, Trilokinath Modi, Gothenburg, June 2021

Contents

Acronyms

ANN Artificial Neural Network.
ATC Anatomical Therapeutic Chemical.

BioSNAP Stanford Biomedical Network Dataset Collection.

CMED Cardiac Medical - for non-surgical cardiac related admissions.
CSURG Cardiac Surgery - for surgical cardiac admissions.
CUI Concept Unique Identifier.
CWA Closed World Assumption.

DAG Directed Acyclic Graph.
DENT Dental - for dental/jaw related admissions.
DO Disease Ontology.
DOID Disease Ontology Identifier.

EHR Electronic Health Record.
ENT Ear, nose, and throat - conditions primarily affecting these areas.

GNN Graph Neural Network.
GU Genitourinary - reproductive organs/urinary system.
GYN Gynecological - female reproductive systems and breasts.

h@k Hits at K.
HIPAA Health Insurance Portability and Accountability Act.
hpt head entity per tail entity.

i.i.d. Independent and Identically Distributed.
ICD International Classification of Diseases.
ICU Intensive Care Unit.

LOS Length of Stay.
LP Link Prediction.
LR Learning Rate.

MED Medical - general service for internal medicine.
MeSH Medical Subject Headings.
MIMIC-III Medical Information Mart for Intensive Care - III.
MRR Mean Reciprocal Rank.
MTHICD9 Metathesaurus International Classification of Diseases.

NMED Neurologic Medical - non-surgical, relating to the brain.
NSURG Neurologic Surgical - surgical, relating to the brain.

OBO Open Biological and Biomedical Ontology.
OBS Obstetrics - conerned with childbirth and the care of women giving birth.
OMED Orthopaedic medicine - non-surgical, relating to musculoskeletal system.
ORTHO Orthopaedic - surgical, relating to the musculoskeletal system.
OWA Open World Assumption.
OWL Web Ontology Language.

PCA Principal Component Analysis.
PSURG Plastic - restortation/reconstruction of the human body (including cosmetic or aesthetic).
PSYCH Psychiatric - mental disorders relating to mood, behaviour, cognition, or perceptions.

RDFS Resource Description Framework Schema.
ReLU Rectified Linear Unit.
RRF Rich Release Format.

SAB Abbreviated Source Name.
SD Standard Deviation.
SNAP Stanford Biomedical Network Dataset.
SNOMED Systematized Nomenclature of Medicine.
SNOMED-CT Systematized Nomenclature of Medicine - Clinical Terms.
SURG Surgical - general surgical service not classified elsewhere.

TC Triplet Classification.
tph tail entity per head entity.
TransD Translation via Dynamic Mapping Matrix.
TransE Translations in the Embedding Space.
TransH Translation on Hyperplanes.
TransR Translation in the Corresponding Relation Space.
TRAUM Trauma - injury or damage caused by physical harm from an external source.
TSNE t-distribution Stochastic Neighbour embedding.
TSURG Thoracic Surgical - surgery on the thorax, located between the neck and the abdomen.

UMLS Unified Medical Language System.

VSURG Vascular Surgical - surgery relating to the circulatory system.

WHO World Health Organization.

1
Introduction

In this chapter we will give a brief introduction to the background of this thesis work and the topics we wish to investigate. We will continue to present related work, and further on the aim and limitations of this project. The research questions we wish to answer will be stated before lastly presenting ethical considerations regarding this project, as well as a thesis outline.

1.1 Background

Medical records contain large amounts of information, either as structured data or as free text. Apart from the operational usage of medical records, e.g. by healthcare practitioners, the available data can be used to build machine learning models to support clinicians and practitioners. The models can be trained to predict patient outcomes, e.g. remaining time in hospital [42], in-hospital mortality [33], drug recommendation systems [31] etc. Further on, there is active research based on the usage of graph-based machine learning in Intensive Care Unit (ICU) [34], expanding the methods used for various patient outcome predictions. The ICUs are often in demand by critical patients and need detailed planning in resource allocation. A well-trained model can inherently help the staff to pre-plan and facilitate resources beforehand, as explained by Matlakala et al [37] as well as Holland et al [25], together with details about the challenges faced in resource allocation in .

Patient outcome predictions have been extensively studied using conventional classifiers (e.g. Support Vector Machines, Artificial Neural Networks, Naïve Bayes Method and Random Forest) [55] [12]. These methods typically use tabular format data, and can quickly scale when dealing with biomedical data, thus imposing challenges when there are many features and corresponding categories to consider. The idea that modelling patients along with medical concepts in a graph-structure can yield good predictive ability by capturing the semantic relationships between different entities and relations.

Ultimately, representing large amounts of information is a challenging task. A knowledge graph is a concept where entities, e.g. objects, events and concepts, are stored in a graph-structured data model or topology. The entities act as nodes, whereas the relations between these entities act as directed edges in the graph structure. A visual representation of this idea is presented in figure 1.1. Google uses the concept of knowledge graphs for their info-box when using their platform [47], and similarly

Amazon within their product recommendation system [19]. Patients described by their medical records can be represented by such a network, as any event a patient undergoes is represented as a fact. We can expect that patients with similar ailments will have a large share of mutual nodes, and with the addition of including domain-specific knowledge within the knowledge graph can abridge the gap between the events in a meaningful way.

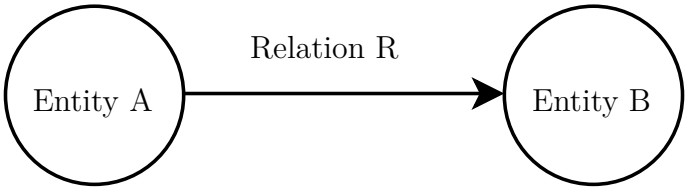


Figure 1.1: Two entities and their relation. Entity A acts as the head entity and entity B acts as the tail entity, implied by the direction of relation R. Note that a relation is generally not reversible - thus a directed relation is introduced.

A distinctive advantage with the usage of knowledge graphs is the possibility to embed the entities and relations by knowledge graph embedding methods. An embedding is a concept of representing a (usually) high-dimensional vector as a continuous vector of lower dimensionality. Knowledge graph embedding methods aim to compute these vectors to preserve the association between connecting entities and the corresponding relations, while capturing the latent properties of semantics in the knowledge graphs [11]. These embeddings can then be explored to make predictions in a down-stream fashion. By using knowledge graphs as a modelling method for healthcare domains, the ability to model latent representations between different information sources is presented which is difficult for other data models.

1.2 Related work

Healthcare offers one of the most interesting, promising, but arduous field for deep learning algorithms. There is a continuous increase in the machine learning algorithms applied in healthcare [45]. Among many machine learning techniques, knowledge graphs prove to be helpful in medicine with tasks like drug discovery and target gene-disease prediction, to name a few. Nevertheless, entities in the field of medicine cannot be formulated in relational tables easily, whereas such entities are often tangled with other entities via different types of relations. In the simplest of manner, a graph structure fulfils the visualization of such data. In the following few paragraphs, a discussion is made on how researchers face difficulty creating these graph structures, followed by how researchers use knowledge graphs in medicine research and how researchers aim to use knowledge graphs in healthcare, i.e., more patient-centred research.

The analysis regarding life-science linked open data conducted by Kamdar et al shows that the researchers endure severe technical challenges to integrate heterogeneous data from multiple sources [30]. Some of these sources can be the vast amount

of biomedical data and knowledge, such as medical records [29], biomedical publications [4] etc. They conclude by demonstrating the discrepancy in the quality of life science linked open data sources, and discuss various challenges such as lack of mappings between similar entities, varying notations and lack of reuse of standard vocabularies. A recently (February 2021) published research paper by Bonner et al [14] reviews various biomedical data sets with a central idea of the data set applicability to knowledge graphs. Additionally, they aim to guide the machine learning researchers regarding multiple biomedical terminologies and introduce high-quality resources. Lastly, they list out a set of challenges, where the need of better metadata, incorporation of features and data uncertainty are some of the challenges they addressed.

Various problems related to patient outcome prediction are addressed using machine learning techniques and classical statistical models. Song et al explores and compares machine learning techniques against statistical learning technique to classify patient health risk [49]. Dai et al aims to predict patient hospitalization using tree-based algorithms under supervised learning problem [18], whereas Beaulieu-Jones et al aims to classify disease sub-types and improve genotype-phenotype association studies using denoising autoencoders for dimensionality reduction and semisupervised learning for disease sub-type classification [10].

Wu et al presents a comprehensive survey on Graph Neural Network (GNN), where their evaluation proposes a new taxonomy to divide the evolving field of graph neural networks [56]. Additionally, they aim to provide a hands-on guide for understanding and using the different deep learning approaches for various fields and discuss the possible applications of GNN across various domains[1]. Ali et al performs a detailed analysis and large scale evaluation on various knowledge graph embedding techniques [8]. This article holds an important role in the completion of the thesis work. Gong et al uses two bipartite graphs, namely patient-disease and patient-medicine, to create a heterogeneous graph structure [24], and further successfully uses the learned embeddings to recommend safe medicine to patients. Another example is the work by Rotmensch et al [44], where researchers aim to create a health knowledge graph from electronic health records.

A funding of approximately £300 000 for a span of 3 years (2018-2021) is made for the research project described in [2]. The project aims to develop a knowledge graph, expand it with other related biomedical entities and make predictions on anonymized electronic health record data. Based on the extracted cohort of patients, classifiers are trained to predict outcome relating to clinical variables and risk associated. The research findings from this project are not yet published nor found by the thesis workers as of 25 May 2021. However, this research outline was used as motivation for our similar thesis outline.

[1]GNN and knowledge graphs are not the same but fall under a parent branch of graph networks. Nevertheless, a branch of GNN deals with heterogeneous graph structures that are not studied as a part of this thesis.

1.3 Aim

The thesis aims to describe the construction of a patient-centered knowledge graph based on different data-sources, and further on representing the patient information in latent vector spaces using knowledge graph embedding methods and classifying patient re-admission statuses. The evaluation of the patient embeddings is done via link prediction, whereas classifying the patient re-admissions is done by two methods, namely triplet classification and in a supervised manner using artificial neural networks. The artificial neural networks will evaluate both the embeddings generated in the link prediction approach, as well as two baseline-data sets which exclusively describes the patient-information.

1.4 Limitations

Studies on re-admission classification have shown promising results by applying natural language processing techniques on patients clinical notes [26]. Similarly, studies have shown that the use of time-series dependent patient features are also good predictors of patient outcome predictions [42]. Since these types of information are rather difficult and non-trivial to place in a knowledge graph, we have limited our scope of using static patient variables, and have thus disregarded time-dependent features such as vital-signs and clinical notes written at the end of a patients admission. This can be a significant limitation as patients health condition are generally monitored by vital signs, and clinical notes often describe a patient's health-behaviour quite well.

Another limitation is the amount of external data sources which are investigated. Specifically, we look into the direct relations between drugs and diseases, the indirect relations between drugs and diseases via genes and hierarchies within diseases, procedures and drugs. In the biomedical domain, these association types are merely a tiny fraction. In reality, there are plenty of associations that could trigger or affect any relation between the entities considered [3].

An assumption that could be considered a limitation is that we assume every fact mentioned in all the considered databases to be correct. Many databases are algorithmically generated, and studies have shown they may not be fully accurate for this reason [51]. However, this is not investigated in the thesis, and is thus imposed as a limitation.

1.5 Research questions

Since we wish to evaluate and compare a set of different classification algorithms and their frameworks based on data sets of different types, the project aims to answer the following research questions posed by the authors:

- How well do the models correctly classify the re-admitted patients within 30, 90 and 365 days, as well as patients who are not re-admitted?
  Medical practitioners are often aware of patient's readmission risk, however, having an prediction on the timeline could help plan resources better. Therefore, different timeline of readmission posed to be an interesting research question. The duration $\{30, 90, 365\}$ days for readmission offers a near uniform distribution in readmission status in the dataset with $\approx 5.7\%, 3.8\%, 5.5\%$ respectively. For an effective training of model this often becomes important to consider.
- How does the knowledge graph embedding method's abilities to predict readmission compare to each other?
  The Sahlgrenska University Hospital aimed to explore the possibility of using knolwdge graphs in patient centric data. Hence, attempting different knowledge graph embedding methods becomes invaluable.
- Does the included information gathered from other data sources improve the classification algorithm's ability to predict re-admissions? As explained in section 3.1, a single dataset generally may not necessarily contain important information about how patient conditions are interlinked. Hence, different datasets becomes essential to model. This thesis work aims to open the possibility for Sahlgrenska to further expand the external source data on top of the considered datasources as mentioned in section 3.1.1.

1.6 Ethical considerations

Machine learning algorithms needs large amounts of data in order to learn different behaviours and generalize over unseen data, but the core principle of privacy states "collect as little data as possible" [9]. This dichotomy is difficult to reconcile and stands as a challenge to privacy. Further on, patient information is a highly sensitive information, and sharing the data may lead to some extent in jeopardized autonomy, whereas, concealing the information might lead to compromise in the objective. There have been incidents in the past where medical practitioners have used there position to sell patient identification [41]. This can capped under serious ethical security breach. Moreover, a survey conducted found that approximately 73% of physicians text other physicians about work-related issues [7].

The Health Insurance Portability and Accountability Act (HIPAA) aims to protect the healthcare information. It regulates strict audits in timely manner to ensure preserving ethical issues at right order. The data set used in this thesis requires to complete 9 mandatory certificates explaining best practices to ensure that healthcare data is not misused. The data set has de-identified all sensitive patient-oriented information, such as time-shifting and removal and cleaning of patient information.

1.7 Thesis outline

The report is further on divided into four major parts - theory, methods, results and discussion, with the addition of a last chapter which present the conclusion of the thesis. Chapter 2 describes the theory behind the concepts used in this project, i.e. the concepts regarding knowledge graphs and the theory behind the representation learning of knowledge graph embedding methods. Chapter 3 describes how the different data-sources have been combined and derived, as well as the functionality behind the methods used. Chapter 4 describes the results from our preliminary investigations, and are further on discussed and commented in chapter 5, together with future possibilities. Chapter 6 states our final thoughts and conclusions of the thesis.

The link to thesis work programs that were used is `https://github.com/trilokimodi/Thesis.git`[2].

[2]The GitHub repository is private. Please send a request email to any thesis worker to avail permissions.

2
Theory

This section is devoted to giving the readers a brief background of some of the techniques and methods used - both in this project and in machine learning in general. It is assumed that the reader is familiar with the field before reading this paper. We will describe the key concepts related to knowledge graphs, knowledge graph embedding methods, the different classification methods used and metrices chosen to evaluate these.

2.1 Knowledge graphs

A knowledge graph is a directed, multi-relational labelled graph consisting of entities and relations acting as nodes and edges. The *entities* act as 'real world' objects, events and concepts, and the connection (relation) describes how the different entities are connected. This knowledge is stored in a graph-structured data model or topology based on available information from various sources. This way of representing knowledge is intuitive and generic when different associations connect different entities. More formally, a knowledge graph is a subset of the Cartesian product of a set of entities and a set of relations, expressed as:

$$\mathcal{K} \subseteq \mathbf{K} = \mathcal{E} \times \mathcal{R} \times \mathcal{E} \tag{2.1}$$

where $\mathcal{K}$ is the set set of observed facts, $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is set of relations. Each member of set of observed facts is called a "triplet" and is denoted as (h, r, t) where $h, t \in \mathcal{E}$ represents a triplet's respective head and tail entities, and $r \in \mathcal{R}$ represents the relation between the head and tail entities. The direction is from head entity to tail entity and is decisive of how the relation is defined. An exemplary knowledge graph can be shown in figure 2.1.

In addition to triplets presented in a knowledge graph, different assumptions can be considered regarding triplets *not* presented in the knowledge graph - whether or not there's a plausibility that a unobserved triplet can be true despite not being presented in the knowledge graph. The Closed World Assumption (CWA) states that triplets not presented in a knowledge graph are considered to be false, whereas triplets not presented in a knowledge graph are unknown to be true or false under the Open World Assumption (OWA) [40]. Depending on the type of problem that is investigated, either one of the setting is used. For example, in the example knowledge graph, the triplet $P2, hasDisease, D2$ is considered to be unknown under the OWA, whereas it's considered to be false in the CWA.

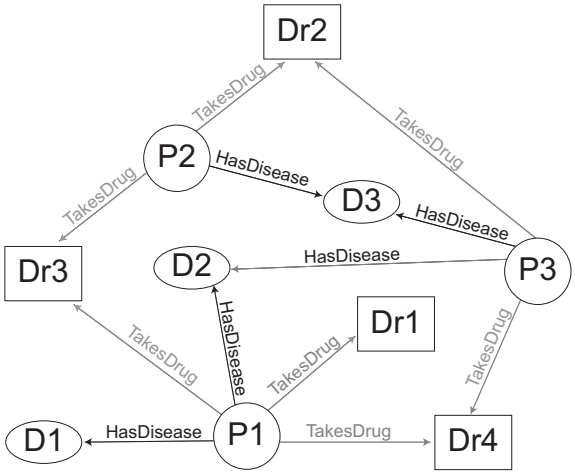


Figure 2.1: Exemplary knowledge graph of three patients. Circles represent patients, rectangles represent drugs and ellipses represents diagnoses. Different link colours denotes different relation-types.

2.2 Knowledge graph embedding methods

In this section, bold characters are used to denote embedded vector, a relation $r \in \mathcal{R}$, entity $e \in \mathcal{E}$ can be considered true unless otherwise stated. As a part of this thesis, only translation based embedding methods have been investigated. This implies that the general form for the energy function will always be `head embedding + relation embedding` $\approx$ `tail embedding`. The specific embedding method investigated will influence the construction of the energy function by varying different forms of `head, tail or relation` embeddings.

Knowledge graph embedding methods are used to learn latent vector representations of the entities $e \in \mathcal{E}$ and relations $r \in \mathcal{R}$ in a knowledge graph $\mathcal{K}$ that preserves its structural properties. The quality of the embedded vectors preserving their structural property depends on the choice of embedding method and data itself. Each of the embedding method hosts an interaction function $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ responsible to provide a real-valued measure on the plausibility of a triplet, i.e. a higher score generally reflects a higher plausibility for a triplet being true. Note that the interaction functions are model-dependent and the scores can't always be interpreted as probabilities directly, but rather needs to be evaluated for each model.

Different models have the ability to capture different relation cardinalities, i.e. $1:1$, $1:N$, $N:1$ and $N:M$ connections, which then naturally imposes different complexities on the models. Knowledge graphs are in general created with OWA i.e. the relations associated to each entity is not limited to the existing facts within the graph structure. Furthermore, a model with enough training parameters could easily over-generalize the true facts. To avoid this problem, a set of false facts is needed while training, which is further explained in subsection 2.3. All the discussed embedding methods follows the same algorithm-schema as described in algorithm 1.

Algorithm 1: Translational knowledge graph embedding algorithm.

Input: Training set $\mathcal{S} = \{(h, r, t)\}$, $h, t \in \mathcal{E}$, $r \in \mathcal{R}$ and margin γ
Initialize: Appropriate entity and relation embeddings
loop
  Appropriate normalization (method-dependent)
  $\mathcal{S}_{\text{batch}} \leftarrow$ sample($\mathcal{S}$,b) // Sample mini-batch of size b from training set $\mathcal{S}$
  $\mathcal{T}_{\text{batch}} \leftarrow \emptyset$ // Initialize the set of pairs of triplets
  for $(h, r, t) \in \mathcal{S}_{\text{batch}}$:
    $(h', r, t') \leftarrow$ sample($\mathcal{S}'_{(h,r,t)}$) // Sample a corrupted triplet
    $\mathcal{T}_{\text{batch}} = \mathcal{T}_{\text{batch}} \cup \{((h, r, t), (h', r, t'))\}$
  end for
  Update embeddings w.r.t. method-dependent energy function
  $\sum_{((h,r,t),(h',r,t')) \in \mathcal{T}_{\text{batch}}} \nabla \left[\gamma + d(\mathbf{h}, \mathbf{r}, \mathbf{t}) - d(\mathbf{h}', \mathbf{r}, \mathbf{t}')\right]_+$
end loop

The initialization mentioned in the algorithm 1 is a modified Xavier/Glorot initialization, which is a commonly used initialization schema within various fields of machine learning in order to avoid gradients from exploding or vanishing in the back-propagation step [6] [22]. The embedding vectors are initialized as described in equation 2.2.

$$x_i \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \tag{2.2}$$

where x_i is a scalar in the embedded vector and n denotes the embedding size. Note that in some models, entity and relation embeddings can have different embedding sizes. Further on, the entities and relations undergoes normalization after every epoch, depending on the model selected. This is to make sure that the constraints each method imposes are fulfilled, which are described in their own sub-section.

Lastly, the embeddings are updates using stochastic gradient descent with the Adam stochastic optimizer. Overall, the algorithm proceeds to sample a batch from training set triplets, where for each training triplet in the batch, the algorithm corrupts one triplet, computes the loss for the batch and proceeds to update the embeddings by taking a gradient step. The following sub-sections will in detail describe the different embedding methods and their methodologies. In particular, a description of their energy function changes will be provided.

2.2.1 TransE

Translations in the Embedding Space (TransE) [16] is a knowledge graph embedding method seeking to learn low-dimensional embeddings for entities and relations in the knowledge graph by considering the relations as *translations* in the constructed embedding space, thus trying to model the head and tail entities $h, t \in \mathcal{E}$ in a presented triple $(h, r, t) \in \mathcal{K}$ so that the tail embedding should lay close to the head embedding, with the addition of the relation vector $r \in \mathcal{R}$, i.e. $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. This simple but rather powerful model relies on a reduced number of tune-able parameters and

creates only one vector representation for each entity and relation. Note that the embedded vector for an entity is the same whether it's presented as a head or tail entity, and that the embedding sizes for entities and relations must be the same. A graphical illustration of the methodology is shown in figure 2.2. The energy function for TransE is defined as following:

$$d(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_{L_1/L_2} \tag{2.3}$$

This energy function returns a non-negative value for a true triplet and conversely should return a higher non-negative value for a false triplet. In order to learn these embeddings, wee seek the minimization of a margin-based ranking criterion over the training set, which is defined as:

$$\mathcal{L} = \sum_{(h,r,t)\in\mathcal{S}} \sum_{(h',r,t')\in\mathcal{S}'_{(h,r,t)}} [\gamma + d(\mathbf{h}, \mathbf{r}, \mathbf{t}) - d(\mathbf{h}', \mathbf{r}, \mathbf{t}')]_+ \tag{2.4}$$

The notation of $[x]_+$ means the positive part of x, and γ is treated as a margin hyper-parameter in the model. The model can be evaluated with both L_1 and L_2 norm. The set $\mathcal{S}'_{(h,r,t)}$ is the set of corrupted triplets, which is composed of training triplets where either the head or tail entity has been corrupted, i.e. replace by another entity randomly. Note that both entities are not corrupted at the same time, only one at the time. We define this set as following:

$$\mathcal{S}'_{(h,r,t)} = \{(h', r, t)|h' \in \mathcal{E}\} \cup \{(h, r, t')|t' \in \mathcal{E}\} \tag{2.5}$$

The loss function encourages the model to discriminate between true triplets in our knowledge graph and false ones constructed in the training phase. The optimization also imposes the constraint that the L_2-norm of the entity embeddings is unity. This is to avoid the trivial minimization of the loss function by inflating the norm of the entity embeddings.

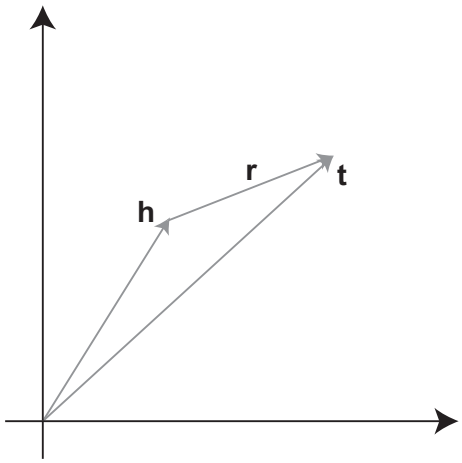


Figure 2.2: An illustration for the working of TransE methodology. The head embedding $\mathbf{h}$ adds up with relation embedding $\mathbf{r}$ to produce tail embedding t. All $\mathbf{h}, \mathbf{r}, \mathbf{t}$ lie in the same space $\mathbb{R}^n$.

2.2.2 TransH

Translation on Hyperplanes (TransH) [21] aims to project the head and tail entity embeddings $\mathbf{h}$, $\mathbf{t}$ of a true fact on a hyper-plane, whereas the projections are connected by a translation vector $\mathbf{d}_r \in \mathbb{R}^n$ laying on this hyper-plane. The role of the relation embedding $\mathbf{r}$ in the true fact comes in constructing the hyper-plane and the translation vector $\mathbf{d}_r$. Specifically, a relation is characterized by two vectors, a norm vector $\mathbf{w}_r \in \mathbb{R}^n$ of the hyper-plane and a translation vector $\mathbf{d}_r$. Note, n is a tuneable hyper-parameter. With an independent hyper-plane for each unique relation, the model tackles the difficulties of modelling $1:1$, $1:N$, $N:1$ and $N:M$ relation-cardinalities in a more suitable way. Figure 2.3 illustrates the projections of entities on a relation hyper-plane, together with its translational vector. The projections embeddings are denoted as $\mathbf{h}_\perp$ for head entity $\mathbf{h}$ and $\mathbf{t}_\perp$ for tail entity $\mathbf{t}$. Within each hyper-plane, the objective is to obtain $\mathbf{h}_\perp + \mathbf{d}_r \approx \mathbf{t}_\perp$ for each true triplet. The energy function for a true triplet $(h, t, r) \in \mathcal{K}$ is expressed as $d(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \|\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp\|_2^2$. By introducing the constraint $\|\mathbf{w}_r\|_2 = 1$, we obtain that:

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r \tag{2.6}$$

$$\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r \tag{2.7}$$

Thus, we can re-write the energy function as:

$$d(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \|\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp\|_2^2 = \left\| \left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r\right) + \mathbf{d}_r - \left(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r\right) \right\|_2^2 \tag{2.8}$$

Again, the energy function returns a non-negative scalar which is lower for true triplets and higher for false triplets.

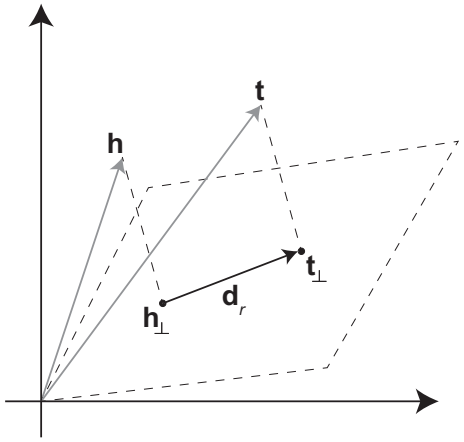


Figure 2.3: An illustration of TransH methodology. The original embeddings $\mathbf{h}$, $\mathbf{t}$ are projected in a relation hyperplane $\mathbf{r}$. The projected embeddings are connected by a relation translation vector $\mathbf{d}_r$. All $\mathbf{h}, \mathbf{r}, \mathbf{t}$ lies in the same space $\mathbb{R}^n$.

In order to learn the embeddings of the entities and relations, the same margin-based ranking criterion is used as in the TransE-method, i.e. equation 2.4, as well as the defined set of corrupted triplets defined in equation 2.5. As mentioned in the

algorithm 1, head and tail entities and relation normal vectors are normalized after each epoch. The rest of the training details follows as mentioned at the end of the section 2.2.

2.2.3 TransR

The previously two mentioned embedding methods are based on the assumption that both entity and relation embedding are presented in the same vector space $\mathbb{R}^n$. Translation in the corresponding relation space (*TransR*) [35] is an embedding method that overcomes this assumption by modelling the entities and relations in separate embedding spaces, i.e. an entity embedding space and one unique relation embedding space for each presented relation. This is done since each entity may have several aspects, and relations may focus on different aspects of the entities. Hence, TransR tries to model the facts in multiple relation spaces to achieve relation specific entity spaces. A graphical illustration is shown in figure 2.4.

The intuition behind TransR is that for each triplet $(h, r, t) \in \mathcal{K}$, the head and tail entity embeddings $\mathbf{h}, \mathbf{r} \in \mathbb{R}^n$ are first projected onto the r-relational space using the relation-specific projection-matrix $\mathbf{M}_r \in \mathbb{R}^{k \times d}$, and obtains the projected head and tail entity embeddings $\mathbf{h}_r = \mathbf{M}_r\mathbf{h}$, $\mathbf{t}_r = \mathbf{M}_r\mathbf{t}$. We wish that these are projected in a way so that $\mathbf{h}_t + \mathbf{r} \approx \mathbf{t}_r$ for the specific relation r and its embedding $\mathbf{r} \in \mathbb{R}^k$. This gives the embedding model the possibility to model head and tail entities that hold a specific relation lay close to each other in that specific relation-space, and lay far from each other in another relation space where it's not a true triplet. The energy function is then computed as:

$$d(\mathbf{h}_r, \mathbf{r}, \mathbf{t}_r) = \|\mathbf{h}_t + \mathbf{r} - \mathbf{t}_r\|_2^2 \tag{2.9}$$

The energy function returns low-valued scalars for true triplets and higher-valued scalar for false triplets. Constraint are set upon the model's embedding vector and matrices, more specifically $\forall h, t \in \mathcal{E}$, $\forall r \in \mathcal{R}$, $\|\mathbf{h}\|_2 \leq 1$, $\|\mathbf{t}\|_2 \leq 1$, $\|\mathbf{r}\|_2 \leq 1$, $\|\mathbf{h}\mathbf{M}_r\|_2 = 1$ and $\|\mathbf{t}\mathbf{M}_r\|_2 = 1$. These constraints are met after each epoch during training.

The embeddings for this model is learnt by the same margin-based ranking criterion used in the TransE-model, i.e. equation 2.4, with margin γ as a hyper-parameter, and the set of corrupted triplets defined in equation 2.5.

2.2.4 TransD

Translation via dynamic mapping matrix (*TransD*) [27] is an extension to TransR. In case of TransR, a single mapping matrix is used, whereas in case of TransD the idea is to project head and tail entities differently as they are generally of different attributes. Two embedding vectors are constructed for each head and tail entities $h, t \in \mathcal{E}$ and relation $r \in \mathcal{R}$, i.e. $\mathbf{h}, \mathbf{h}_p, \mathbf{t}, \mathbf{t}_p \in \mathbb{R}^n$ $\forall h, t \in \mathcal{E}$, $\mathbf{r}, \mathbf{r}_p \in \mathbb{R}^k$ $\forall r \in \mathcal{R}$. The first embedding vector explains the entity or relation and the second defines

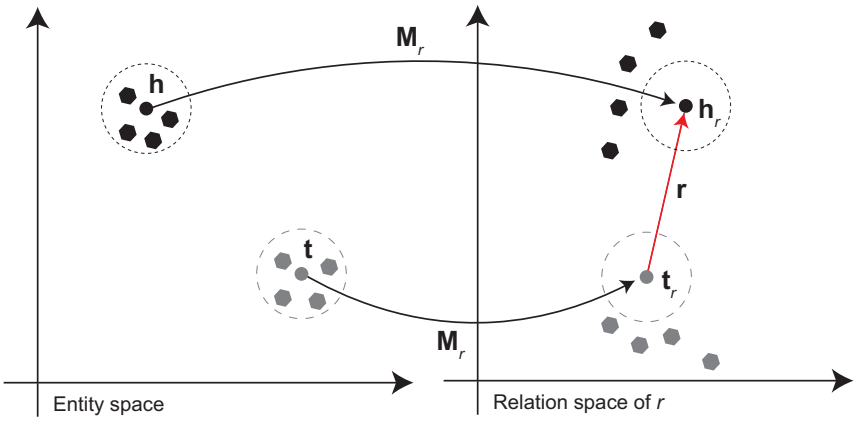


Figure 2.4: An illustration for the TransR methodology. The head and tail embeddings **h**, **t** share the same space $\mathbb{R}^n$ and are projected to a relation **r** specific space $\mathbb{R}^m$ using a single projections matrix.

how to project an entity to relation space. Matrices $\mathbf{M}_{r,h}$, $\mathbf{M}_{r,t} \in \mathbb{R}^{k \times n}$ are the two mapping matrices used for projections. These matrices are defined as:

$$\mathbf{M}_{r,h} = \mathbf{r}_p \mathbf{h}_p^\top + \mathbf{I}^{k \times n} \tag{2.10}$$

$$\mathbf{M}_{t,h} = \mathbf{r}_p \mathbf{t}_p^\top + \mathbf{I}^{k \times n} \tag{2.11}$$

Here, $\mathbf{I}^{k \times n}$ is identity matrix. The projection matrices are thus determined by both the entity and relation. The second set of embedding vectors are then projected onto the relation space by the newly constructed projection matrices as following:

$$\mathbf{h}_\perp = \mathbf{M}_{r,h} \mathbf{h} \tag{2.12}$$

$$\mathbf{t}_\perp = \mathbf{M}_{t,h} \mathbf{t} \tag{2.13}$$

The energy function for a observed triple $(h, r, t) \in \mathcal{K}$ is once again defined as:

$$d(\mathbf{h}, \mathbf{r}, \mathbf{t}) = \|\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp\|_2^2 \tag{2.14}$$

This energy function returns a low-valued scalar for true triplets and a higher-valued scalar for false triplets. Further constraints are set upon the model - the head and tail entity embeddings should be smaller or equal to unity, as well as their corresponding projected vectors. This applies to the relations as well, i.e. $\forall h, t \in \mathcal{E}$, $\|\mathbf{h}\|_2 \leq 1, \|\mathbf{t}\|_2 \leq 1, \|\mathbf{h}_\perp\|_2 \leq 1, \|\mathbf{t}_\perp\|_2 \leq 1$, $\forall r \in \mathcal{R}$, $\|\mathbf{r}\|_2 \leq 1$. These constraints are met in each epoch of the training.
Once again margin-based ranking criterion is used i.e. equation 2.4, where the margin γ is a hyper-parameter, and the set of corrupted triplets defined in equation 2.5.

2.3 Negative sampling

The negative facts are generated by assuming stochastic local closed world assumption. Under stochastic local closed world assumption, it is assumed that

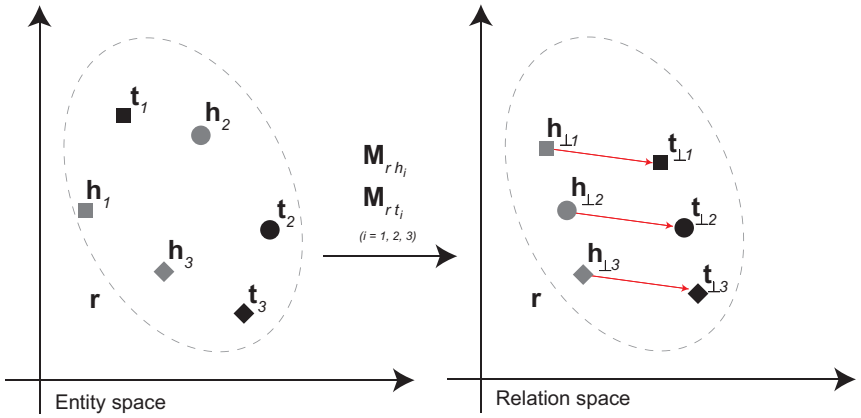


Figure 2.5: An illustration for the TransD methodology. The head and tail embeddings **h**, **t** share the same space $\mathbb{R}^n$ and are projected to a relation **r** specific space $\mathbb{R}^m$ using 2 projection matrices individually for head and tail entities respectively.

for any given true fact $(h_i, r_i, t_i) \in \mathcal{K}$, a false fact is any triplet from the set of $\{(h_j, r_i, t_i), (h_i, r_j, t_i), (h_i, r_i, t_j) \notin \mathcal{K}$. The negative sample can be generated from the set by either corrupting the head or tail entity. The choice of selecting head or tail entity is done by two sampling methods, namely uniform negative sampling or Bernoulli negative sampling.

Under uniform negative sampling, head or tail entity is randomly selected with uniform distribution and later an entity $e \in \mathcal{E}$ is uniformly sampled and selected as the corrupted head or tail entity. This negative sampling technique was used by the authors of the TransE embedding method.[16]

Under Bernoulli negative sampling, the probability of corrupting head or tail depends on the type of relation in the fact. If the relation is of $1 : N$ (one to many) then a higher probability is assigned to corrupt head h, and vice-versa for the case of $N : 1$ (many to one). In general, in order to make it more computationally efficient, for each relation $r \in \mathcal{R}$, the average number of tail entity per head entity (tph) and average number of hpt is computed. The probability of selecting head entity h or tail entity t to corrupt is then given by the Bernoulli parameter $\frac{tph}{tph+hpt}$ and $\frac{hpt}{tph+hpt}$ respectively. The Bernoulli negative sampling technique was introduced by the authors of the TransH embedding mehtod[21].

2.4 Evaluation

The training of the knowledge graph embedding methods involves an evaluation phase based on a constructed validation graph. The evaluation can involve either triplet classification or link prediction on the validation graph, depending on the embedding scenario. These are presented in the subsection below.

2.4.1 Triplet classification

According to the definition present in [53], triplet classification is defined as the task of finding if the given triplet (h, r, t) is true or false. The given triplet is scored using the scoring function of the specific embedding method at use. The triplet is considered true if the score is larger than a given threshold value t. In this thesis, the triplet classification has been modified.

The modified definition of triplet classification is, given a set of heads h and a specific relation r, the triplet is considered to be true if it obtains the maximum score among the set of tails t. The details of the modified triplet classification will be further discussed in method section 3.4.3.2.

2.4.2 Link Prediction

The performance of knowledge graph embedding methods are usually evaluated in the setting of link prediction,using the newly constructed embedding vectors. A link prediction task is defined as predicting either head h or tail t given a entity-relation pair (r, t) or (h, r) respectively. The prediction is generally done by a ranking function, and the ranks are decided by the score of the triplet. The link prediction algorithm is described in the algorithm 2[1].

Algorithm 2: Link prediction algorithm

for each triplet (h, r, t) in validation or test set:
  corrupt h
    Score the triplet $(h^{'}, r, t) \quad \forall \quad h^{'} \in \mathcal{E} - \{h\}$
    Sort the scores
    Find the rank of true triplet (h, r, t) among sorted scores
    Store the rank
  corrupt t
    Score the triplet $(h, r, t^{'}) \quad \forall \quad t^{'} \in \mathcal{E} - \{t\}$
    Sort the scores
    Find the rank of true triplet (h, r, t) among sorted scores
    Store the rank
end loop
Use a rank measurement technique to compute the overall link prediction performance

2.4.3 Rank measurement technique

Mean Reciprocal Rank (MRR) can be defined as the mean over reciprocal individual ranks, defined as in equation 2.15. The MRR-value lies in range $[0, 1]$, and indicating that a higher mean reciprocal rank is yielding a better predictive

[1]The exact algorithm is more efficient.

ability. However, MRR is not uniform because of the reciprocal rank definition. For example, MRR of 0.5 would imply a prediction within the top two entities, and MRR of 0.1 would imply a prediction within the top 10 entities, and a MRR of 0.01 would be within the top 100.

$$MRR = \frac{1}{|\mathcal{K}_{valid}|} \sum_{(h,r,t)\in\mathcal{K}_{valid}} \frac{1}{rank(h,r,t)} \tag{2.15}$$

Hits at K (h@k) - Hits at K is the fraction of test triples that are ranked under top K to the total number of triplets in the test set. Its definition is given in equation 2.16.

$$H@K = \frac{\sum \delta_i}{|\mathcal{K}_{valid}|} \tag{2.16}$$

where $\delta_i = 1$ if $rank(i) < K, \quad \forall \quad i = (h,r,t) \in \mathcal{K}_{valid}$.

During the validation phase using the described link prediction task, the MRR-metric will be used as an early stopping criteria, and h@k will be parallely computed to draw inferences.

2.4.4 Other evaluation metrics

When evaluating the performance of classification problems, the metrics used needs to reflect the strengths and weaknesses of the learning algorithm. A way to present the results is with the terms *true positives*, *true negatives*, *false positives* and *false negatives*. In a binary setting, where only two classes are to be predicted, this is a very comprehensible concept - either it belongs to one class (true) or the other (false). However, in the setting of multiclass classification with C classes, one needs to evaluate each individual class against all others together, i.e. if class C_0 is considered and a sample's true target is of class C_0, then a *true positive* if considered if the sample's predicted target is of class C_0, and a *false negative* if it's one of the remaining classes $\{C_1, \ldots, C_{|C|-1}\}$. In the same way, if a sample has a corresponding true target that is *not* C_0, and the prediction is of class C_0, then this is considered a false positive, and a *true negative* is considered if the predicted target is one of the remaining classes $\{C_1, \ldots, C_{|C|-1}\}$. Note that this doesn't the difference between classes others than C_0, only that it's actually different from C_0. In other words, metric-scores are calculated for each class, regarding the remaining classes as negative. It is evaluated as a binary classification problem for each class individually. After this, a general metric-score can be calculated by a weighted average over each classes metric score to obtain an average performance value. Each class has then obtain their corresponding set of true positives (TP), false negatives (FN), false positives (FP) and true negatives (TN). From this, the following metrics can

be calculated for each class as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.18}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.19}$$

$$\text{F1-score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2.20}$$

Accuracy is a rather straight forward metric which measures the proportion of correct predictions among all predictions. However, one limitation with this is when the distribution among the classes is imbalances, i.e. the number of test-pairs are not equal among the different classes. If 95% of the sample-pairs belong to class C_0, then classifying everything as C_0 and ignoring the remaining classes will yield a good accuracy, but miss the information given from the other classes. To calculate the proportion of predicted positives that is truly positive, then *precision* is used for this evaluation. This gives a more nuanced evaluation than just looking at the number of correct predictions. Another metric is *recall*, which tells us about the proportion of true positives that are actually predicted correctly. This metric is useful when trying to capture as many positives as possible. *F1-score* is the harmonic mean between precision and recall and is heavily used when the classes are either imbalanced or when predicting true positives is important on all classes. If either of the former metrics has a low value, it will be reflected in the F1-score, thus a high score here implies high scores for both precision and recall.

When calculating these metrices for several classes in the same classification task, the *macro-score* is an average of the metric-score over each class, defined as:

$$\text{Macro-metric} = \frac{\sum_{c=1}^{|C|} \text{Metric-score}_c}{|C|} \tag{2.21}$$

The highlight of this thesis is to use knowledge graph embedding methods and important theories related to it has been discussed. Nevertheless, other important concepts like deep learning, details on evaluation metrics, supervised learning and gradient based optimization has been explained and presented in the appendix section B.

3
Methods

This chapter describes the preliminary experiential setup and our working pipeline. The project can be divided into four phases, involving data cleaning, knowledge gathering, knowledge graph embedding and model fitting, as seen in figure 3.1. We will describe the different phases in detail, including the different data sources used, different data splits, and how we train the models. We will lastly present our proposed model training scheme as well as the metrics which evaluates these.

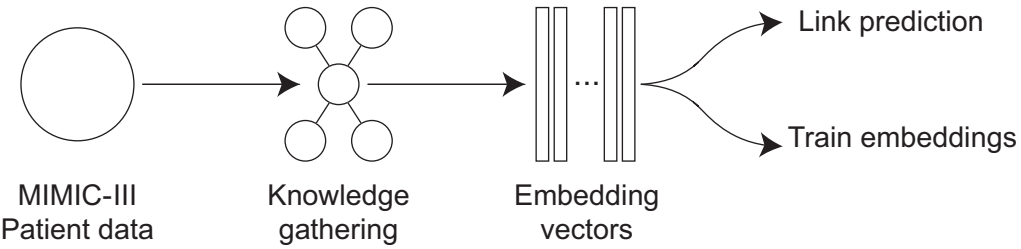


Figure 3.1: Different phases of the project. Information regarding patients is extracted from MIMIC-III, further on expanded with various data sources, before constructing and evaluating different algorithms.

3.1 Data

We wish to explore if the incorporation of domain-specific information can be utilized to yield better predictive performances than patient data alone. For the patient-central data, we have looked at the Medical Information Mart for Intensive Care - III (MIMIC-III) database version 1.4 [28]. MIMIC-III alone is an excellent data source to model patient specific models. However, it doesn't contain information on the science of diseases, drugs, and how these are related to each other. For example, MIMIC-III will present information about patient's diseases, prescriptions and procedures, but it doesn't contain information on which drug is used to cure what disease or which set of drugs targets the genes affected by a single or set of diseases. Connecting non-patient entities gives additional flexibility to the knowledge graph embedding methods on bringing related entities closer to each other. To accomplish the addition of edges involving non-patient entities, knowledge from other biological ontologies were incorporated. For data related to diseases, the Stanford Biomedical Network Dataset (SNAP) has been investigated [36], as well as Disease Ontology (DO) [46] and Unified Medical Language System (UMLS) [13][38]. For information

pertaining to drugs, DrugBank 5.0 [54] has been investigated, as well as SNAP. For data related to genes, the SNAP database was examined.

3.1.1 Biomedical Ontologies

An ontology is a collection of terms designed to represent knowledge in a specific domain. Biomedical ontologies are vast and complex, and is therefore very difficult to design a single logical terminology that can cover all biomedical information. The tangled interdependence of the working nature of diseases, organs, drugs, genes, proteins, and their interactions makes it inherently difficult to represent information in a hierarchical or structural way. Hence most of the biomedical ontologies represent knowledge in Resource Description Framework Schema (RDFS) or Web Ontology Language (OWL). Both of these languages are designed for web formats and have separate browsers to view and manipulate defined ontologies. These web format databases contain biomedical information interlinked in a Directed Acyclic Graph (DAG) structure with nodes as one of the entity in category and edges representing a relation between two entities.

3.1.2 MIMIC-III

This is a large, single-center database containing information regarding de-indentified patients admitted to critical care units admitted to the Beth Israel Deaconess Medical Center in Boston, Massachusetts. The data includes vital signs, medication, measurements obtained in laboratories, observation and chart notes written by caretakers, procedure and diagnoses codes, survival data, patients information and more. The relational database consists of 26 tables, which are linked by identifiers, which are further explained in appendix A. The database is frequently used within both the academia and industrial research fields. It consists of data corresponding to $58,976$ unique hospital admissions for a total of $46,520$ unique patients between the years 2001 and 2012. The data has been modified so that patients are de-identified according to the HIPAA standards, by structured data cleaning and date shifting. This process demands removal of identifying elements, such as patient name, phone number, address and dates. Furthermore, dates are consistently shifted with a positive random offset for each patient, so that seasonal, weekly and daily time intervals are preserved. Patient's birth of date has been shifted if their true age is over 89 years to comply with HIPAA regulations, and thus appear with ages of over 300 years in the database. Protected health information, such as diagnostic reports and physician notes, is removed from free text fields using a well-evaluated de-identification system. As the database contains sensitive and detailed information regarding the clinical care of patients, it must be treated with care and respect. In order to obtain the database, one must send a formal request for access via the MIMIC-website, where a recognized course in protecting human research participants needs to be completed. Furthermore, one must sign a data use agreement, which states appropriate data usage and security standards, where it also forbids efforts to identify patients.

3.1.3 DrugBank

Drugbank [54] is a comprehensive cheminformatics database, containing rich information about drugs and drug-related information, such as drug function, formulation, mechanism, metabolism, interactions and more. It is estimated that there are 16820 unique drugbank identifiers for drugs and 4422 unique drugbank identifiers for salt compounds, classified among type categories as approved, experimental, investigational, illicit, etc.

3.1.4 Stanford Biomedical Network Dataset

The Stanford Biomedical Network Dataset Collection (BioSNAP) [36] can be considered as an ensemble of bipartite graphs.[1] The graphs are stored as tsv-files, with each row representing a edge connecting two nodes. The nodes-types can be among diseases, drugs, genes, proteins etc. The term bipartite is used to emphasize that each of the tsv-file contains a maximum of 2 entity types, implying all rows in each tsv-file are of same relation type. It is therefore straight-forward to make a heterogeneous graph with many interesting features by using the BioSNAP graph database. These graph structures are not linked data, and hence, unlike many other biomedical ontologies, it can be difficult to find additional information about any selected entity solely from SNAP.

3.1.5 Disease Ontology

The human ()[46] is designed to abridge the gap between different biomedical ontologies through disease concepts. The other disease concepts cross-referred in are Medical Subject Headings (MeSH), , Systematized Nomenclature of Medicine (SNOMED), etc. In total, there are about $9,069$ unique Disease Ontology Identifier (DOID). It also aims to provide linked reference to other ontologies, textual information about the disease and grouping of closely related diseases in hierarchical fashion. The human disease ontology is a part of Open Biological and Biomedical Ontology (OBO) foundry. OBO is a community driven mission to develop a family of inter-operable ontologies[2][48].

International Classification of Disease - ICD-9 is the ninth revision of the International Classification of Diseases (ICD) [1] accepted throughout the world for indexing of hospital records, morbidity and mortality statistics. ICD-code is designed for healthcare classification system and includes naunced classification of variety of signs, symptoms, anomalies, external causes, injuries etc. The ICD-codes are revised periodically, where the ongoing version in healthcare is ICD10. The version used in MIMIC-III contains patients disease information in the form of ICD9-codes and hence only ICD9-codes are considered as this thesis area of interest in DO.

Medical Subject Heading - Medical Subject Headings (MeSH) [43] is one of

[1] This statement has not been verified. It is stated to motivate the general understanding of BioSNAP database.

[2] `http://www.obofoundry.org/`

the most widely used biomedical ontologies in the research domain. Hence, a large number of biomedical databases like UMLS (section 3.1.6), SNAP (section 3.1.4), DO (section 3.1.5) etc. includes MeSH-terms in their cross-references. Moreover, articles in PubMed have MeSH-terms attached to describe which biological entity is referred to. This project considers the MeSH concepts only, and not the entire MeSH ontology.

Systematized Nomenclature of Medicine - Unlike ICD, Systematized Nomenclature of Medicine - Clinical Terms (SNOMED-CT) [5] is a nomenclature designed for clinical terms. SNOMED-CT provides the core terminology for electronic health record and aims to improve the patient care, and includes approximately $311,000$ unique concepts. SNOMED-CT is used by standards setting organizations and in various health care applications, and is updated twice a year.

3.1.6 Unified Medical Language System

Unified Medical Language System (UMLS) [13] [38] is a large repository of interrelated biomedical concepts developed by U.S. National Library of Medicine. The repository is coined as Metathesaurus in UMLS website and publications. Unlike many other biomedical ontologies which lacks focus on the clinical aspects, UMLS aims to tackle this by involving a sub-domain with general terminologies such as SNOMED-CT and ICD. Some of the concepts in UMLS can be linked to external sources. Ontologies from some of the non-English language are also covered by methathesaurus.

3.1.7 Other terminologies

There is an abundance of various other terminologies within biomedical databases depending upon entity type, nature of database and applicability of the entity. Some of the entities that are used in this project are described in previous subsections, and some more are described in this section. The terminologies not used within this project are neglected.

Genes [52] - Based on genetic information, human diseases can be classified as monogenic, chromosomal, or multifactorial. Alteration in a single gene leads to monogenic disease, whereas alteration in chromosome leads to chromosomal. The mutlifactorial diseases are the complex diseases and is caused due to variation of many genes. Finding associations between disease and genetic variation is a part of genetic research. Some of the disease-drug relations are created through common gene i.e. disease affecting gene X and drug targeting gene X.

Targets - A target is a protein encoded by genes that are targeted by drugs. These targets are sometimes the carrier of drug or is responsible for the activation of drug.

3.2 Data Set Creation

This section will describe the different tables and data sources which the data sets are built upon. The process of the patient cohort selection will be described together with their chosen features and corresponding statistics. How different data sources are related and the work-flow regarding the actual act of connecting of these sources will be presented as well. Finally, the different data sets created will be presented - what information they include as well as which particular data set will be used in which re-admission classification pipeline.

3.2.1 Admission details

This subsection describes the different features regarding a unique patient's admission. Their admission details are given in ADMISSIONS and PATIENTS tables. These tables are centered around patients and their admission and the data tables consists of 58976 unique admission events for a total of 46520 unique patients. Other tables used in order to obtain patient features are ICUSTAYS, which describes a patient's visit to the ICU-departments, as well as SERVICES, which describes the different services a patient has been admitted under during their stay.

The different counts of re-admissions for all patients is presented in table 3.1. Features are selected based on the patient's first admission. This is due to the difficulty of modelling patients with several admissions within the same knowledge graph - distinguishing between different feature sets between different admissions. Also, observations in a prior admission can influence the response in a later admission.

Table 3.1: The table states the distribution of number of admissions different patients had in the entire MIMIC-III dataset.

Number of admissions	Number of patients	Percentage
One	38983	83.80%
Two	5160	11.09%
Three	1342	2.88%
Four	508	1.10%
More than four	527	1.13%

The Length of Stay (LOS) for each admission can be derived based on the available information in the table - with variables describing the date and time for both admission and discharge of a patient's visit. We can see in subtable (a) in table 3.2 that certain patients has a negative LOS, which usually are patients that act as organ donors[3]. These corresponds to 98 patients which have expired before their arrival and are thus removed from the data set. Subtable (b) in table 3.2 shows the distribution of LOS for the remaining patients. Since we need to create an entity-type representing the length of stay, we bin it within certain intervals. Making

[3] `https://mimic.mit.edu/iii/mimictables/admissions/`

weekly-long bins, we end up with the representation described in table 3.3.

Table 3.2: The table states the basic statistics of Length of stay considering the MIMIC-III patients. Table (a) represents patients considering all patients whereas (b) represents the set of patients that were selected for modelling.

(a) All patients.

Count	46520
Mean	10.131
Std	12.708
Min	-0.945
25%	3.715
50%	6.357
75%	11.718
Max	294.660

(b) Filtered patients.

Count	46422
Mean	10.151
Std	12.713
Min	0.004
25%	3.729
50%	6.369
75%	11.734
Max	294.660

Table 3.3: The table states the binned length of stay bin distributions. The binning becomes essential as the current implementation of Knowledge graph doesnot support numeric features.

Bin interval	**Number of patients**	**Percentage**
Less than one week	25159	54.20%
One to two weeks	12297	26.49%
Two to three weeks	4221	9.09%
More than three weeks	4745	10.22%

Since we also wish to predict the re-admissions for patients, keeping patients who die within their first (and only) admission is rather ambiguous and would skew the distribution of patients that are actually re-admitted and not re-admitted. For this reason, these 4319 patients are removed from the data set, resulting in 42103 unique patients.

All time-stamps presented in the MIMIC-III data set have been shifted in order to comply with the de-identification regarding HIPAA-regulations. However, the time-stamps are still intact in relation to each other, i.e. we can calculate the age for the patients by taking the difference between their date-of-birth and admission time. We remove patients that are infants, because these generally lack the descriptive features we wish to explore in this project. This reduces the number of patients to 34296. It should also be noted that MIMIC-III doesn't include pediatric patients, and that the lowest age for the patients is currently 16 years. We group patients within the following age-bins as presented in table 3.4.

Since we wish to predict readmission within the ranges of 30, 90 and 365 days, keeping patients who die within these ranges after their first discharge will bias the

Table 3.4: The table states the binning strategy and distribution for different age group of the patients in MIMIC-III dataset.

Age interval	Number of patients	Percentage
[16,40)	3868	12.70%
[40,60)	9355	30.71%
[60,80)	12336	40.49%
[80,∞)	4906	16.10%

predictions since they have no re-admission because of expiring. After removing these patients, we are left with 30466 patients. We also find that one patient has a second admission before their first discharge. This may be due to an error in the recording system, so this patient is removed from the data set. The resulting data set consists of 30465 unique patients and their respective first admission. We can now look into the different rates of re-admission for the patients after their first discharge. Taking the difference between their second admissions date and first discharge date, we obtain after how many days patients are re-admitted (if re-admitted at all). We obtain the following distribution presented in table 3.5.

Table 3.5: The table states the distribution of the response variable readmission status. The remaining 25899 belongs to no readmission category.

Within days	Number of patients	Percentage
[0, 30)	1741	5.71%
[30, 90)	1152	3.78%
[90, 365)	1673	5.49%

Within each patients first admission, they are admitted to the ICU different number of times, as well as different period of times. We can extract from the ICU-table how many times each patient has been admitted to the ICU, as well its accumulated time spent at the ICU. We can bin these time-interval in day-ranges, i.e. *less than one day*, *one to two days*, up to *six to seven days*, and from there we bin the higher day counts into the same bin *more than seven days*. We obtain the following distributions presented in table 3.6.
The ETHNICITY-table describes this demographic information regarding the patient in our cohort. Looking into this table, we can state that the distribution of ethnicities is very imbalanced among the 41 different categories. The full description of ethnicities can be found in appendix A with its corresponding mapping schema. We narrow down the groups to the following ethnicities presented in table 3.7.
The RELIGION-table described this demographic information which patients leave upon arrival at the admission. Looking into the different religions regarding the patients in our cohort selection, we have 19 different categories ranging from 7 to 15734 patients in each category. The full distribution and its corresponding mapping schema can be found in appendix A. In order to deal with class imbalance, we narrow this distribution down to the following religion categories presented in table 3.8.

Table 3.6: The table (a) states the ICU specific binned length of stay distribution, whereas the table (b) gives simple statistics about the feature ICU LOS.

(a) Binned ICU-LOS.

Time (days)	Count	Percentage
[0,1)	4879	16.02%
[1,2)	9878	32.42%
[2,3)	5281	17.33%
[3,4)	2945	9.67%
[4,5)	1736	5.70%
[5,6)	1100	3.61%
[6,7)	780	2.56%
[7,∞)	3866	12.69%

(b) Accumulated ICU-LOS.

Count	30465
Mean	3.938
Std	6.108
Min	0.000
25%	1.164
50%	2.048
75%	3.952
Max	260.709

Table 3.7: The table states the distribution of ethnicity of the patients in MIMIC-III dataset.

Ethnicity	Number of patients	Percentage
White	21760	71.43%
Unknown/not specified	3515	11.54%
African American	2445	8.03%
Hispanic American	1117	3.67%
Other	899	2.95%
Asian American	729	2.39%

Table 3.8: The table states the distribution of religion of the patients in MIMIC-III dataset.

Religion	Number of patients	Percentage
Catholic	10859	35.64%
Unknown/Not specified	10580	34.74%
Protestant	4143	13.60%
Jewish	2496	8.19%
Other	2387	7.83%

Looking into the different admission locations in the ADMISSION-table, most of the patients are admitted to the emergency room. However, quite many are also admitted under physicians referral and clinical referral, i.e. transfer of care for a patient from a physician/clinic to another by request. Other admission locations are referring to transfers, either from a hospital, a skilled nursing facility, other healthcare facility, or within the same facility. These are grouped together under the category *transfer*. We end up with the following distribution presented in table 3.9. The original distribution and its corresponding mapping to deal with class imbalance can be found in appendix A.

Each patient is discharged after their visit to the hospital, which is presented in

Table 3.9: The table states the first admission location within the hospital of the patients.

Admission location	Number of patients	Percentage
Emergency room admission	12481	40.97%
Physician referral	6788	22.28%
Transfer	5643	18.52%
Clinical referral	5553	18.23%

table 3.10. Some locations have quite few data points so these have been grouped together to *other*. The full distribution and its corresponding mapping schema is also presented in appendix A.

Table 3.10: The table states the discharge location for patients after their first admission to the hospital.

Discharge location	Number of patients	Percentage
Home	10257	33.67%
Home health care facility	8855	29.07%
Skilled nursing facility	4391	14.41%
Rehab	4302	14.12%
Other	1524	5.00%
Long term care hospital	1136	3.72%

Checking into the different admission types in the ADMISSION-table, the categories describe how each patient is admitted. We see that only a few admissions fall under *URGENT* $(908, 2.98\%)$, which is somewhat related to the category *EMERGENCY* $(23729, 77.89\%)$. We place these two together under the name *EMERGENCY* and obtain that 24637 (80.87%) are admitted under emergency whereas 5828 (19.13%) have an elective admission.

Checking into the insurances for the different patients, we have the following categories: Medicare $(14650, 48.08\%)$, Private $(11636, 38.19\%)$, MedicAid $(2688, 8.82\%)$, Government $(1051, 3.45\%)$ and Self-Pay $(440, 1.44\%)$. We grouped Self-Pay together with Private because these are in some sense similar, obtaining the final altered class Private $(12076, 39.64\%)$.

Checking into marital status, we observe many missing values for this specific feature. These are imputed to the class *Unknown/Other*. We narrow down the classes because of imbalance and obtain the following distribution: *married* (14965, 49.12%), *single* (7935, 26.05 %), *widowed* (3743, 12.29%), *divorced* (1882, 6.18%), *separated* (316, 1.04%), *unknown* (207, 0.68%) and *life partner* (0.04%). We group together *life partner*, *separated* and *unknown* together under the common category *unknown/other* and obtain the following distribution presented in table 3.11.

Table 3.11: The table states the distribution of marital statuses for the patients.

Marital status	Number of patients	Percentage
Married	14965	49.12%
Single	7935	26.05%
Widowed	3743	12.29%
Unknown/Other	1940	6.39%
Divorced	1882	6.18%

The services-table provides information the service a patient was admitted under. A patient may be located at a specific ICU-unit, the patient may not be cared for by the staff at the specific unit. This can be due to many reasons, as for example bed shortage within the different units. There are 20 unique services in the MIMIC-III data set, and 18 unique in our data set. Patients can be transferred to several services during the same stay. The following distribution of patients admission to different services is presented in table 3.12.

Table 3.12: The table states the frequencies of the services the patients were admitted under. The table contains the abbrevations of the service type and their full forms are listed in the Acronyms.

Service	Counts
CMED	6097
CSURG	6610
DENT	5
ENT	175
GU	335
GYN	255
MED	11889
NMED	1686
NSURG	2659
OBS	109
OMED	775
ORTHO	790
PSURG	207
PSYCH	1
SURG	3294
TRAUM	2529
TSURG	811
VSURG	794

3.2.2 Diagnoses

In MIMIC-III, the information regarding patient's diseases is provided in a table titled DIAGNOSES_ICD. All the diagnoses information from the patient's first admission is stored as features. A total of 6984 unique disease codes was found from MIMIC-III dataset. After the cohort selection, a total of 6031 unique disease codes are present, and in total 321250 occurring diagnoses. What is noteworthy is that 6 patients in our data set do not have a diagnosis set for their first admission. For this reason, they are removed from the data set and we are left with 30459 unique patients and their respective first admission. The 5 most frequently occurring diagnoses in our cohort selection are presented in table 3.13 together with their respective counts.

Table 3.13: The table states the most frequently occurring diagnoses in cohort selection. The information in the table is not used in the modelling, rather it is for informational purpose.

Diagnosis	**Number of patients**
Unspecified essential hypertension	13390
Coronary atherosclerosis of native coronary artery	8590
Atrial fibrillation	6741
Congestive heart failure, unspecified	5958
Other and unspecified hyperlipidemia	5411

The diseases are further grouped by ICD9-hierarchies using the information available at Wikipedia[4]. This is a tree-based grouping combining similar diagnoses within each branch, where new branches and leaf-nodes pair can be constructed for a more specific description of diagnoses. The frequency distribution of MIMIC-III diseases under each hierarchy is presented in table 3.14.

[4] `https://en.wikipedia.org/wiki/List_of_ICD-9_codes`

Table 3.14: The table states the higher grouping of ICD9 disease codes and their frequencies in the MIMIC-III dataset. The higher order groupings is directly taken based on the ICD9 standards as mentioned in section 3.2.2.

ICD9 Code	Higher Group	Counts
001-139	Infectious	8850
140-239	Neoplasms	5903
240-279	Endocrine	38314
280-289	Blood	11848
290-319	Mental	15169
320-389	Nervous	11747
390-459	Circulatory	78555
460-519	Respiratory	21557
520-579	Digestive	20158
580-629	Genitourinary	15117
630-679	Pregnancy	639
680-709	Skin	3983
710-739	Muscular	7355
740-759	Congenital	1292
760-779	Perinatal	3397
780-799	Symptoms	11805
800-999	Injury	27592
E,V codes	External	37969

3.2.3 Procedures

Much like the diagnoses, a patient's underlying procedure information is provided in a table titled PROCEDURES_ICD. There are 2009 unique procedural codes presented in the MIMIC-III data set, and after the cohort selection there are 1843 unique procedural codes present. Note that some patients do not have any procedures performed on them. However, they are left in the data set. In total, there are 132851 performed procedures in the data set. The 5 most frequently occurring procedures in this data set is shown in table 3.15. Like diseases, the procedures can be grouped by hierarchies using Wikipedia information[5] and is shown in table 3.16. Similar to diagnoses hierarchies, it groups procedures related into branches, where more specific procedures can be described further down in the tree-structure.

3.2.4 Prescriptions

Similar to diagnoses, patient's prescriptions information is retrieved from the table PRESCRIPTIONS in MIMIC-III. This lists the different drugs that a patient has been prescribed by a physician during their admission. The prescriptions listed in MIMIC-III is noisy mainly due to presence of quantified drugs, drug synonym,

[5] `https://en.wikipedia.org/wiki/ICD-9-CM_Volume_3`

Table 3.15: The table states the most frequently occurring procedures in cohort selection. This table is not used for modelling purpose, but it is only stated for informational purpose.

Procedure	**Number of patients**
Venous Catheterization, Not Elsewhere Classified	6860
Extracorporeal Circulation Auxiliary To Open Heart Surgery	6001
Insertion Of Endotracheal Tube	4190
Coronary Arteriography Using Two Catheters	4063
Continuous Invasive Mechanical Ventilation For Less Than 96 Consecutive Hours	4057

Table 3.16: The table states the higher grouping of ICD9 procedures codes and their frequencies in the MIMIC-III dataset. The higher order groupings is directly taken based on the ICD9 standards as mentioned in section 3.2.3.

ICD9 Code	**Higher group**	**Counts**
00-00	Interventions	0
01-05	Nervous	0
06-07	Endocrine	0
08-16	Eye	2,197
18-20	Ear	591
21-29	Nose/mouth	1,566
30-34	Respiratory	12,594
35-39	Cardiovascular	44,165
40-41	Hemic/lymphatic	1,332
42-54	Digestive	14,493
55-59	Urinary	724
60-71	General	1,845
72-75	Obstetrical	467
46-84	Muscoloskeletal	7,317
85-86	Integumentary	2,643
87-99	Misc	42,917

mixture of drugs, abbreviated drug names, brand names of drugs etc. A quantified drug can be defined as name of the drug followed by a quantity measure. Table 3.17 states the rules used to map drugs present in the MIMIC-III data set to DrugBank indices. Table 3.18 illustrates some more examples under each of these rules.

The proportion of each rule used to convert a drug presented in the MIMIC-III data set to a DrugBank index is shown in the pie-chart figure 3.2. A total of are 1046 unique drugs after cleaning are found in MIMIC-III data set, and after the cohort selection there are 981 unique drugs present with a total of 796705 co-occurring prescriptions.

Table 3.17: The table states the rules used to map MIMIC-III drugs with DrugBank drugs. Since MIMIC-III drugs are noisy, a many same drugs repeat in different forms. The rules are used to map them to common nomenclature i.e. drugbank drugs.

Rule	**Description**
Exact Match	MIMIC-III drug matches exactly with the drugbank drug name
Synonym Match	A substring of MIMIC-III drug matches a synonym of any drug in drugbank
Product Match	A substring of MIMIC-III drug matches a listed product variant of any drug in drugbank
Substring Match	A substring of MIMIC-III drug matches a substring of drugbank drug name, synonym or product
Hardcoded rules	A sustring of MIMIC-III matches exactly a rule

Table 3.18: The table shows some sample drugs under each rule as defined in table 3.17. The column MIMIC-III drug contains the noisy form of the drug and the column DrugBank drug is the standard nomenclature for the respective drug.

Rule	**MIMIC-III Drug**	**Drugbank Drug**
Exact Match	Prednisone	prednisone
	Tiotropium Bromide	tiotropium bromide
	Cefpodoxime Proxetil	cefpodoxime proxetil
	Memantine	memantine
Synonym Match	DHEA	prasterone
	alcohol dehydrated	ethanol
	neo*po*vitamin e tpgs 26.7 unit/ml	tocofersolan
Product Match	Maxalt	rizatriptan
	ear wax drops	carbamide peroxide
	levsin	hyoscyamine
Substring Match	fluticasone-salmeterol (100/50)	fluticasone propionate
	diltia	diltiazem
	cilo	cilostazol
Hardcoded rule	heparin (preserv. free), heparin flush (10 units/ml), heparin flush midline (100 units/ml), heparin flush port (10units/ml)	heparin
	ns (glass bottle) ns (mini bag plus) sodium chloride 3% (hypertonic)	sodium chloride

Each prescription has a hierarchical classification system grouping in the form of

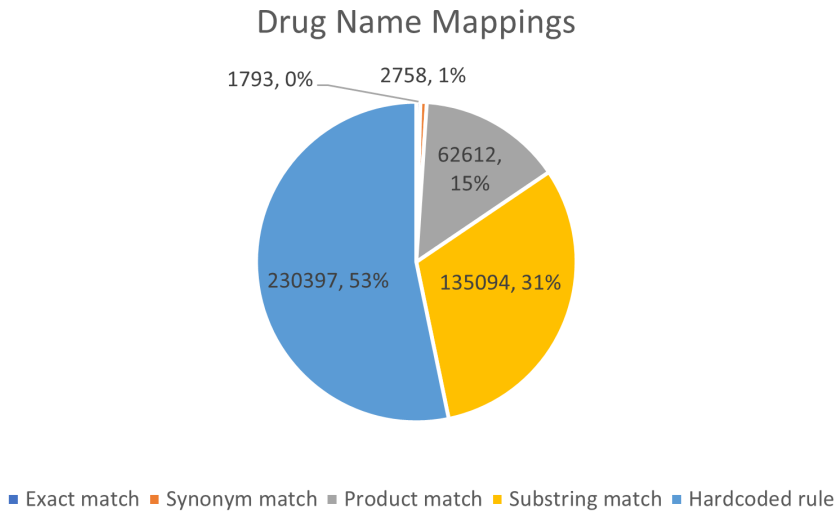


Figure 3.2: Pie chart describing the drug names mapping frequencies under each rule described in table 3.17.

Anatomical Therapeutic Chemical (ATC)-groupings[6]. This code is assigned to a prescription according to the organ or system it works on and how it works, and is furthermore maintained by the World Health Organization (WHO). The first character of an ATC-code presents the anatomical main group and consists of 14 groups. We extract this ATC-level from DrugBank for all presprictions in our cohort selection. Note that not all prescriptions had an ATC-code available, so these are mapped to ATC-level *X*. The distribution of ATC-grouped prescriptions is presented in figure 3.3.

3.2.5 Final MIMIC-III data sets

We have obtained the information wanted in order to construct the baseline-data sets. These are called *one-hot* and *numeric*. The one-hot data set represents all presented information encoded as one-hot embeddings, meaning that classes that occur within a feature are encoded as a 1 if a patient are within that specific class, and with a 0 else. We obtain 30459 samples with the feature dimensionality of 8922. The numeric data represents similar information but gathered differently - instead of binning the numerical values of features *age*, *LOS* and $ICU - LOS$, we keep these as numeric values but being centered and scaled. Also, since we have grouped diagnoses, procedures and prescriptions in higher order groups, one patient may now have several occurrences within each hierarchical group. This data set has 30459 samples and the feature dimensionality of 100. Further on, the cohort selection information extracted from the MIMIC-III data set will also be used within the creation of knowledge graphs, which is described in section 3.3.

[6] `https://www.ema.europa.eu/en/glossary/atc-code`

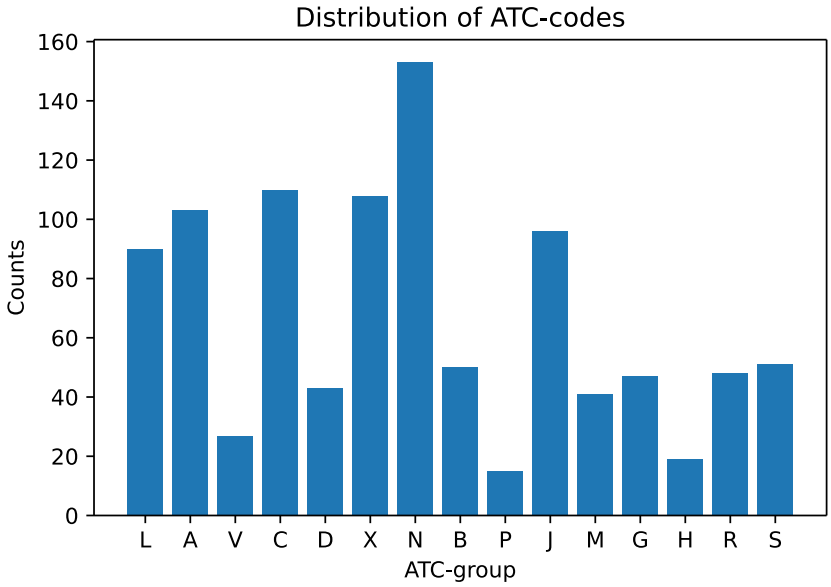


Figure 3.3: The figures illustrates the distribution of ATC-codes for prescriptions in our cohort. The ATC code is a standardized higher order code for grouping of drugs.

3.3 Knowledge Gathering

The overall MIMIC-III data gathering and cleaning pipeline is shown in figure 3.4. Patient information is collected from MIMIC-III and external links are generated using other data sources. The details of knowledge gathering is described in the current section.

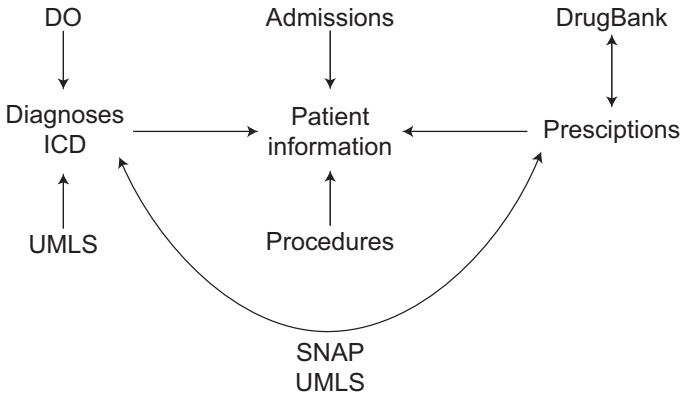


Figure 3.4: A illustration of how patients MIMIC-III data is linked with other data sources.

Patient nodes obtained from MIMIC-III is rich in information because of the edges between patient and their information. Finding the relation between diseases and drugs using other biomedical databases is the goal of the current phase in the

pipeline.

3.3.1 Diagnoses knowledge gathering

To accomplish the goal of connecting disesaes with drugs, they need to be transformed to MeSH ontology. To achieve this, various disease databases such as DO, UMLS are used. First, the available mappings between ICD9 and SNOMED-CT is extracted from the UMLS ICD9 to SNOMED mapping file `ICD9CM_SNOMED_MAP_1TO1_202012.txt`[7]. The raw structure of the information extracted from this file is stated in table 3.19. The disease ontology is then used to extract MeSH unique identifier.

Table 3.19: The table illustrates the information collected from UMLS ICD9 SNOMED map file. The column named "Column" is the name of the column in UMLS map file and a map between these columns leads to the desired ICD, SNOMED-CT mapping.

Column	Description	Example
ICD_CODE	ICD9 code	427.31
SNOMED_CID	SNOMED code	49436004

The DO[8] contains external reference `xrefs` to various biomedical disease ontologies regarding diseases. These external reference contains unique identifier codes of the underlying disease belonging to other biomedical ontologies. A sample representation is shown in the code block 3.1[9]. If the external refernce contains MeSH along with ICD9 or SNOMED-CT i.e. anyone among { MeSH, SNOMED-CT}, {MeSH, ICD9}, {MeSH, SNOMED-CT, ICD9} then the respective pair is stored in the format of key-value pair with corresponding ICD9 code as key. If {SNOMED-CT, MeSH} was found then UMLS cross reference is used to get the ICD9 code before making a key-value pair.

Using Disease Ontology and UMLS ICD9 - SNOMED-CT conversion, approximately 650 MIMIC-III ICD9 codes were mapped to MeSH codes. 650 is only one-eleventh of the total number of ICD codes in MIMIC, hence, the information contained among all ICD9 codes is still sparse. This motivates to explore more databases.

The UMLS MRCONSO.RRF[10] file is used to expand the equivalence links between ICD9 to MeSH links. The MRCONSO file is a RRF file and can be read as a pandas dataframe with delimeter '|'. The raw structure of the information used from CONSO file is stated in table 3.20.

[7] `https://www.nlm.nih.gov/research/umls/mapping_projects/icd9cm_to_snomedct.html`

[8] `https://disease-ontology.org/downloads/`

[9] This representation contains only the details that is essential to this project. Many of the intermediate key-value pairs is removed for illustration purposes.

[10] The UMLS knowledge resource metathesauraus can be downloaded from `https://www.nlm.nih.gov/research/umls/licensedcontent/umlsknowledgesources.html`

Listing 3.1: An excerpt from the DO. The excerpt contains the information this is used to extract the required diseases mapping.

```
"graphs" : [{
    "nodes" : [{
        "id" : "http://purl.obolibrary.org/obo/DOID_8717",
        "meta" : {"xrefs" : [
                    {"val" : "MESH:D003668"},
                    {"val" : "SNOMEDCT_US_2020_09_01:28103007"},
                    {"val" : "ICD9CM:707.0"}]
                  }
                 }]
            }]
```

Table 3.20: The table states the information collected from UMLS CONSO file. The column named "Column" are the column names in the CONSO file. These columns are used to obtain the desired ICD and MeSH links.

Column	Description	Example
CUI	Unique identifier for concept	C0000005
SAB	Abbreviated source name	MSH
CODE	Source asserted identifier	D012711

In reference to table 3.20, the column SAB in CONSO file contains abbreviations of various biomedical ontologies in metathesaurus format. For example, MSH is used as an abbreviation for MeSH, ICD9CM for ICD9 diseases and MTHICD9 as the metathesaurus entry terms. The CODE (refer table 3.20) for MTHICD9 is same as ICD9 code. The MTHICD9 was created to provide additional synonyms for ICD9 codes by the UMLS. Although, there are many more such abbreviations in the CONSO file, but only MSH, MTHICD9 and ICD9CM is used to generate equivalence links between ICD9 and MeSH. Nevertheless, the amount of such equivalence links can be further expanded by connecting various other database which is not covered as a part of this project scope.

Table 3.21 provides some examples of the extracted information. The ICD9 codes were then mapped to MeSH code based on common CUI codes. From the table 3.21, it can be inferred that ICD9 789.0 is equivalent to MeSH D003085 and ICD9 634 is equivalent to MeSH D000022 due to common CUI. Similarly, other ICD9-codes to MeSH-codes equivalences were generated and stored as key-value pair for appropriate use.

So far the MIMIC-III diseases and prescriptions are transformed to MeSH diseases and DrugBank indices, but no relation is developed connecting diseases with drugs. To achieve this, SNAP is used. Specifically, the file DCh-Miner[11] contain infor-

[11] https://snap.stanford.edu/biodata/datasets/10004/10004-DCh-Miner.html

Table 3.21: The table provides examples from UMLS Conso file to illustrate how the mapping is made. The column CUI is unique identifier used by UMLS CONSO. The desired mappings are then the result of "SAB" column afer inner joining the CUI

CUI	SAB	CODE
C0000729	MTHICD9	789.0
C0000729	MSH	D003085
C0000786	ICD9CM	634
C0000786	MSH	D000022

Table 3.22: The table illustrates the SNAP disease-drug database structure. The column titled "Column" is the column names in SNAP database column names. The mapping obtained are between disease and drug.

Column	Description	Example
Disease (MESH)	MeSH id of the disease	MESH:D005923
Chemical	Drugbank id for drug	DB00564

mation on drug-disease relationships. The drug-disease associations are useful to enhance knowledge about which drugs treats which diseases. The raw structure of information used is shown in table 3.22.

The example in the table 3.22 is an example of association between the shown disease and drug as they lie in the same row. Since these disease-drug association is available between MeSH and DrugBank ontology, it was necessary to transform ICD9 codes to MeSH ontology and drug names to DrugBank unique id's. The drug-disease associations was then limited to drugs and diseases available within MIMIC-III database.

Drugs and disease can be further associated based on gene and target information. The ChG-Miner and DG-Miner are the chemical-gene (target) and disease gene interaction network respectively offered by SNAP. The network ChG-Miner contain associations between DrugBank indices and the targets (genes) which interacts with the respective drug. Alongside, the DG-Miner contains MeSH disease associations with Genes. This leads to disease-drug relation based on the mutual gene shared between the two. Table 3.23 and table 3.24 illustrates the raw structure of these SNAP databases. Referring to the examples, it can be deduced that disease MESH:D009377 is associated with drug DB01834 due to mutual gene P09211. These gene based drug-disease associations are also limited to the disease and drugs within MIMIC-III database.

Drugs interact with other drugs and can cause adverse drug reactions. Drugbank database contains drug-drug interactions. Hence, drug-drug associations is generated for drugs within MIMIC-III database.

Table 3.23: The table illustrates the SNAP disease-gene database structure similar to table 3.22.

Column	Description	Example
Disease (MESH)	MeSH id of the disease	MESH:D009377
Gene	Gene	P09211

Table 3.24: The table illustrates the SNAP drug-gene database structure similar to table 3.22.

Column	Description	Example
Drug	DrugBank ID for the drug	DB01834
Gene	Gene	P09211

3.3.2 Relations

In the previous sections, various association between entities is discussed. A summary of all edge types is described in table 3.25 and table 3.26. The edges with connections involving diseases, drugs embeds additional information by containing higher group of the respective entity. This allows in reducing the number of nodes in the graph structure and it is expected that large amount of information can be embedded in small embedding dimension due to large number of relations. Some relations like `ATCgroup, d_ICD Group, p_pICD Group, Gene` are set of relations with set size $\{15, 18, 13, 1003\}$ respectively. Having more and more such relations can be benefecial for translational based embedding methods. Further, the relation-type in table 3.26 is not used as triplets in case of link prediction problem. A stratified split of this relation is used in triplet classification problem. Both link prediction and triplet classification are explained in further sections.

3.4 Experimental setup

The key aspect of the experimental setup is to obtain a numerical vector embedding for the entities and relations involved in the graph. The entities can then either be trained for readmission prediction or can be used directly to predict patient readmission status depending on the architecture of the network. Two methods under investigation, i.e. Link Prediction (LP) and TC, are employed to tackle the problem. These methods depend on whether any knowledge of re-admission is present in the graph structure. Hence, an overview of the data splits is detailed in section 3.4.2.

3.4.1 Software details

All code for this project has been written in *Python* version 3.7.10. The training of knowledge graph embedding is done in *TorchKGE* version 0.16.25 which is built solely on PyTorch, version 1.8.1 with CUDA-version 10.1. The model's training

Table 3.25: The table illustrates the Relations types and one example for each relation type. Each row under the columns "Head entity", "Relation" and "Tail entity" illustrates one type of fact. An example is shown in the column "Example".

Head entity	**Relation**	**Tail entity**	**Example**
Patient ID	isGender	Gender	p249, isGender, Female
Patient ID	inAgeRange	Age bin	p249, inAgeRange, age_bin_3
Patient ID	hasEthnicity	Ethnicity	p249, hasEthnicity, WHITE
Patient ID	hasReligion	Religion	p249, hasReligion, CATHOLIC
Patient ID	toAdmLoc	Admission location	p249, toAdmLoc, EmergencyRoomAdmit
Patient ID	toDischLoc	Discharge location	p249, toDischLoc, SNF
Patient ID	hasAdmType	Admission type	p249, hasAdmType, EMERGENCY
Patient ID	hasService	Service type	p4, hasService, MED
Patient ID	hasLOS	Length of stay	p249, hasLOS, one to two weeks
Patient ID	hasInsurance	Insurance type	p249, hasInsurance, Medicare
Patient ID	maritalStatus	Marital status	p249, maritalStatus, DIVORCED
Patient ID	inICURange	ICU stay bin	p249, inICURange, 5 days
Patient ID	ATCgroup	Drugbank drug ID	p249, A, DB00030
Patient ID	d_ICD Group	Disease ICD id	p4, d_infectious, ICD_042
Patient ID	p_pICD Group	Procedure ICD id	p4, p_respiratory, pICD_3323
Drugbank drug id	CURES	Disease ICD id	DB01069, CURES, ICD_4280
Disease ICD id	Gene	Drugbank drug id	DB00898, A5X5Y0, ICD_9779

and early stoppage handling is done using the PyTorch Ignite high level wrapper functionalities, version 0.4.4.

Table 3.26: Relations type for readmission status. There are 4 possible readmission status - r30, r90, r365 and rNo where r30 implies readmission within 30 days, r90 implies re-admission within 90 days, r365 implies re-admission within 365 days and rNo implies no re-admission. Note that if a patient qualifies for r30, it is considered disqualified for r90.

Head entity	**Relation**	**Tail entity**	**Example**
Patient ID	Readmission_Within	Readmission Type	p249, Readmission _Within, rNo

3.4.2 Data splits

Figure 3.5 illustrates the splits made on the graph structures as well as on re-admission statuses for the patients in the cohort selection. In figure 3.5, the set of triplets imply all the facts pertaining to the relations defined in table 3.25. It can be noted that the table doesn't contain relation type "readmission_type". It is dealt separately to avoid data leakage between the training and the testing sets. In the figure 3.5, the edge `convert to KG` is used to convert the set of triplets to a TorchKGE knowledge graph data structure.

The primary rule of splitting in graph structures is, for every entity or relation in test graph there should be at least one fact in the training graph. Abiding with this rule, a $0.8 - 0.2$ split is done giving rise to a training and a validation knowledge graph. As illustrated in the figure 3.5, the complete set of triplet knowledge graph(Triplet KG + Readmission KG's) is used in triplet classification problem where as the the knowledge graph splits(Train and Test KG) is used in link prediction problem. The link prediction problem then use readmission status in down stream tasks i.e. after embeddings is generated.

The re-admission statuses undergoes stratified splits in ratio 0.8:0.1:0.1 resulting in train, validation and test splits respectively. Under the link prediction task, these splits are used as sample-pairs together with generated patient embeddings for a final feed forward neural network classification task. However, in case of the triplet classification problem, the splits are converted to triplets followed by TorchKGE graph structure resulting in train, validation and test (readmission) graphs.

3.4.3 Training Overview

Figure 3.6 illustrates the high level flow of a triplet knowledge graph to readmission classification using link prediction (section 2.4.2) and triplet classification (section 2.4.1) methods.

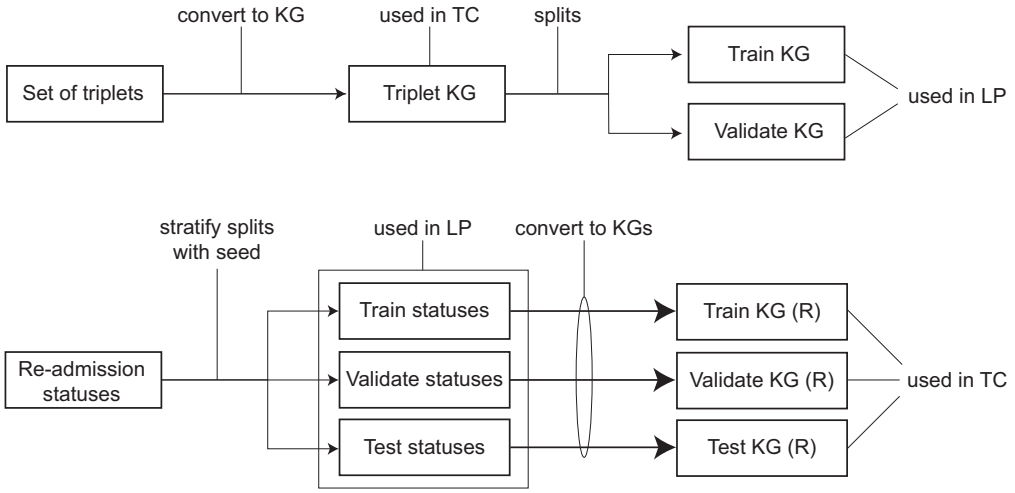


Figure 3.5: The different type of splits and data used for triplet classification and link prediction. In this flowchart, KG is knowledge graph, TC is triplet classification and LP is link prediction.

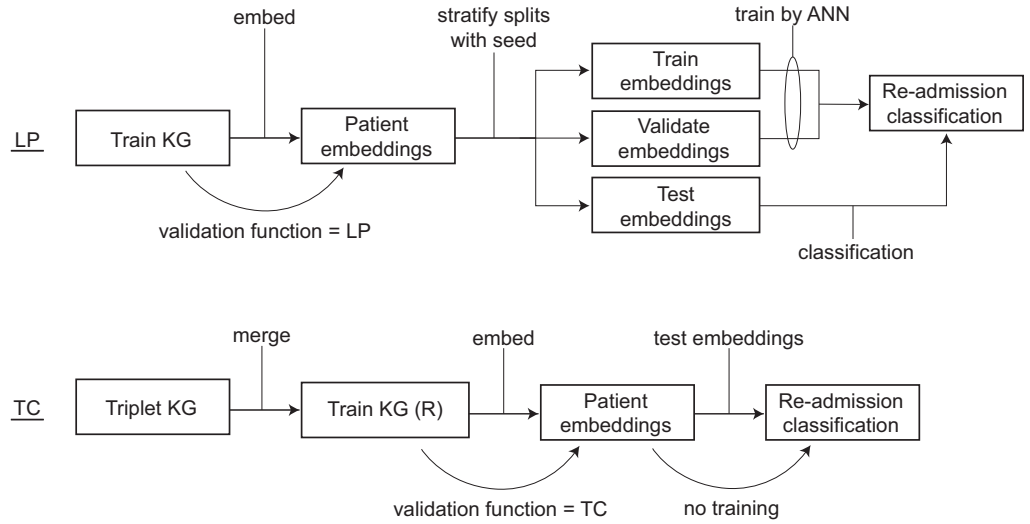


Figure 3.6: A flowchart representing the checkpoints involved in triplet classification and link prediction. In this flowchart, (R) implies re-admission.

3.4.3.1 Link prediction training

The set of triples $\mathcal{K}$ is converted to a Torch-KGE knowledge graph structure with a train-validation split of $0.8 - 0.2$. The split is done ensuring that for each entity or relation in validation set, at least one true fact exists in train set. The training is then done on the training graph structure and validating against the validation graph structure. The validation function used is defined below. Once the training ends, the patient embeddings are fed to the input layer of the neural network architecture described in subsection 3.4.5. The re-admission classification task is then made on the test set to evaluate the performance of the model.

Validation function - The amount of training is controlled by the early stoppage criteria evaluated by the validation function. Mean reciprocal rank, defined in section 2.4.3, is used as the main metric for link prediction validation function. The ranking is done based on the value of the negative energy function. The rank will be higher if the negative energy of the true triplet is higher as compared to false ones. The energy of the triplet is dependent on embedding method being used. The expression for energy measure is given in sections 2.2.1, 2.2.2, 2.2.3, 2.2.4. The rank evaluation algorithm is mentioned in 2.

3.4.3.2 Triplet classification training

The set of triples $\mathcal{K}$ is converted to TorchKGE knowledge graph and is merged with the readmission train knowledge graph. The training is then done on this merged graph structure with validation against readmission validation knowledge graph. The validation function used is defined below. After training, the patient embeddings are used to test readmission status on the readmission test knowledge graph.

Validation function - F1 score is used as the evaluation metric for triple classification validation function. The F1 scores is computed based on true and predicted readmission status for all patients. The readmission status is computed by the negative energy function of the embedding method. Expressions for energy function is given in sections 2.2.1, 2.2.2, 2.2.3, 2.2.4. The readmission status that obtains the highest negative energy is the readmission predicted by the model.

3.4.4 Training details

Figure 3.7 illustrates how each triplet undergoes the training process. The edge "convert to KG" in the figure 3.5 implies converting string formatted triplets to indices, where each entity and relation is the initialized. These indexed triplets are then divided into bathes using TorchKGE's inbuilt DataLoader utility. The training begins by dividing the triplets into batches. The batch size is tune-able but due to computational efficiency it was preset to TorchKGE's default batch size of 32768. Under each batch, a negative sample is generated for each triplet. The negative sample is generated using Bernoulli negative sampler 2.3. A forward pass is applied and the loss is computed. Each forward pass will compute the energy for positive triplet and negative triplet. The loss for each triplet is computed as the sum of margin and their negative energy difference. The total batch loss is propagated backward and the entity vectors are updated as per learning rate. The process is repeated for many epochs i.e. until the validation function allows.

Depending on the problem setting, i.e. link prediction or triple classification, the validation is conducted after every e epochs, where $e = 25$ if the problem is link prediction and $e = 1$ if the problem is triple classification. The model performance is evaluated based on how well the model can complete facts in case of link prediction and amplitude of the macro F1-score for triplet classification. A patience of 50 epochs is used in the case of the link prediction task to let the model validate at least two additional time before actually implementing early stoppage of the train-

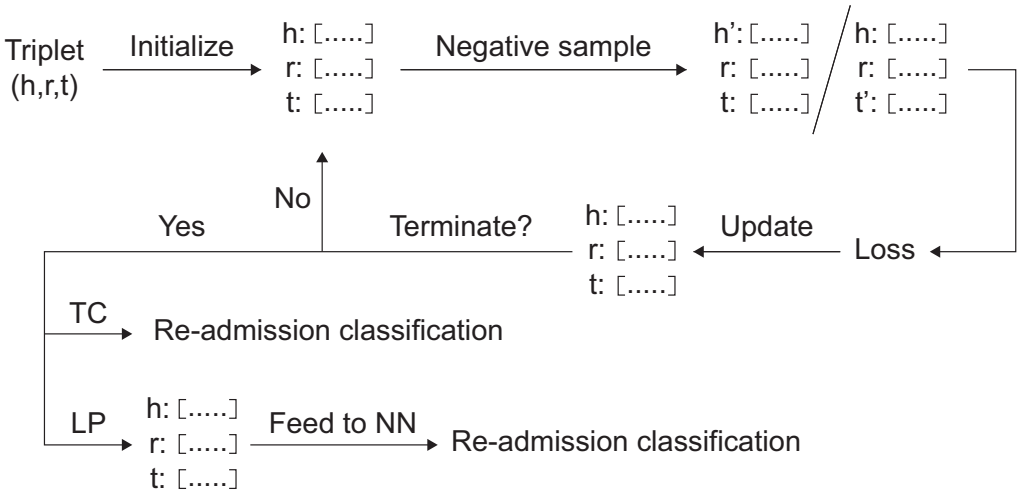


Figure 3.7: Training of each triplet. The [...] implies a numeric vector, $h^{'}$ implies a negative head entity and $t^{'}$ implies a negative tail entity.

ing, whereas a patience of 10 epochs is used in case of triple classification to let the model validate at least 10 times before enforcing the early stoppage whenever a downturn in the metric is observed.

Overall, the model requires 3 hyper-parameter tuning involving embedding dimension, learning rate and margin value for margin loss. Alongside these hyper-parameters, the embedding methodology is explored. It was observed that $FBK15$[15] database is one of the standard data set used by the creators of these embedding method. The data set contains similar number of relations, 60% of total entities as compared to the knowledge graph created in this project. This motivates to the tuning of hyper-parameter in ranges that were used in original research papers. A grid search of model's validation scores is done on these set of hyper-parameters to obtain the model that performs best under each embedding strategy. Table 3.27 states the hyper-parameter selection under each embedding method for triplet classification and table 3.28 states the same for link prediction. The table is a subset of hyper-parameter from the original papers and the subset of hyper-parameters is taken due to computational challenges.

3.4.5 Network architectures

We explore different network architectures for the artificial neural networks in this project. We also conduct exploration of different parameter setting regarding the network itself, as well as different hyper-parameters for the training setting. A early-stoppage criteria is implemented as well in order to avoid overfitting in the training phase, as well as a weighted cross entropy loss function in the training phase's back-propagation algorithm.

A shallow fully-connected feed-forward artificial neural network is created which

Table 3.27: The table states the set of hyper-parameters tried for triplet classification. Among the stated hyperparameters, all possible combinations have been attempted.

Embedding method	Parameter	Set
TransE	Dimension	20, 50
	Learning rate	0.001, 0.01
	Margin	1, 2, 10
	Dissimilarity	L1, L2
TransD	Dimension	20, 50
	Learning rate	0.0001, 0.001, 0.01
	Margin	1, 2, 5
TransH	Dimension	20, 50
	Learning rate	0.001, 0.005, 0.01
	Margin	0.5, 1, 2
TransR	Dimension	20, 50
	Learning rate	0.0001, 0.001, 0.01, 0.1
	Margin	1, 2, 4

Table 3.28: The table states the set of hyper-parameters attempted for link prediction. The motivation for choosing this set comes from the best scoring hyperparameter in the original paper for a dataset of similar size.

Embedding method	Parameter	Set
TransE	Dimension	50
	Learning Rate	0.01 (L1), 0.001 (L2)
	Margin	1
	Dissimilariy	L1, L2
Trans H	Dimension	50
	Learning Rate	0.005
	Margin	0.5
Trans R	Dimension	50
	Learning Rate	0.0001
	Margin	2
Trans D	Dimension	50
	Learning Rate	0.001
	Margin	1

consists of an input layer with the same number of features that each training-pair has, and four output neurons which represent the different possibilities: readmission with 30 days, readmission within 90 days, readmission within 365 days and no readmission. The sub-sequential layer is a hidden layer, with varying number of neurons. These are altered when conducting the hyper-parameter tuning in order to obtain a set of results with varying quality. Between the input and hidden layer, a ReLU-activation function is placed, as well as a dropout-layer with probability equal to 0.1. The last layer outputs a vector of 4 scalars representing the signal strength

for each patient outcome. We choose the neuron and corresponding target with the highest value which will be the predicted target for the sample.

Another deeper fully-connected feed-forward artificial neural network is created with two hidden layers. The remaining architecture is the same - one input layer consisting of the same number of neurons as the sample dimensions, and 4 output neurons. The neurons in the two intermediate layers are altered as a part of the hyper-parameter tuning. It uses the same activation function and dropout-layers as the previously described network, placed both between the input layer and first hidden layer, as well as between the first hidden layer and the second hidden layer.

Note that no activation function is explicitly placed on the output layer - this is since the criterion given to the network architecture is implemented in PyTorch so that the output signal is appropriately transformed before calculating the loss[12].

A stratified training-validation-testing split in the ratio of 0.8:0.1:0.1 is done to maintain the ratio of readmission status in all data splits. The training data is further divided into batches during training with the batch-size of 128. After each training-epoch, the validation set is used to evaluate if early stoppage is of need. If the validation loss is not decreased within 15 epochs, the training is terminated and the network architecture's parameters from the best validation-epoch is obtained for further prediction. From there, the testing data is used as an independent set of samples to evaluate the performance of the network.

The hyper-parameters explored in the simple-net is by altering the number of neurons in the hidden layer by $\{64, 128, 256\}$, and altering the learning rate by $\{0.001, 0.005, 0.01\}$. The deep-net explores the hyper-parameter settings: the values $\{0.001, 0.005, 0.01\}$ are tried for learning rate, for neurons in hidden layer 1 and 2 and the values $\{64, 128, 256\}$ are tried for each individual layer. The parameters β_1 and β_2, ε and the weight-decay values are set to the default values for the Adam-optimizer.

[12] `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`

4
Results

This section will present the preliminary results from our experiments conducted regarding re-admission classification on our patient cohort selection. For each data set, we will describe the different techniques and methods used for classification as well as a short description regarding evaluation and our choices to proceed. Lastly, we will present the performance results for the different models, comparing the models in their ability to capture patients re-admission statuses.

4.1 Baseline models

This section describes the re-admission classification results when feeding the patient data sets alone to the feed-forward artificial neural network. We try out different setting for hyper-parameters on both the simple-net as well as the deep-net, and present the performance evaluation with respect to the two data sets, i.e. one-hot and numeric, with the metrics described in section 2.4.4.

4.1.1 One-hot data set

The results from data set *one-hot* evaluated on the simple-net can be found in table 4.1. The results from data set *one-hot* evaluated on the deep-net can be found in table 4.2. The best obtainable macro F1-score (31.161%) using the simple-net on the one-hot data set was with the setting of 64 neurons in the hidden layer and with a learning rate of 0.005. The best obtainable macro F1-score (31.303%) using the deep-net on the one-hot data was with the setting of 128 neurons in the first hidden layer, 128 neurons in the second hidden layer and with a learning rate of 0.005.

Table 4.1: Simple-net on one-hot data. This table contains top 3 results based on the macro F1-1score. All hyper-parameter combinations is listed in table C.1. Note that HL1 implies hidden layer 1.

Parameters HL1/LR	**Macro Precision Mean (SD)**	**Macro Recall Mean (SD)**	**Macro F1 Score Mean (SD)**
64/0.005	31.194 (0.689)	33.856 (1.421)	31.161 (0.875)
128/0.005	31.044 (0.891)	32.95 (1.661)	30.865 (1.076)
256/0.01	30.664 (0.913)	32.886 (1.493)	30.494 (1.07)

Table 4.2: Deep-net on one-hot data. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations is listed in table C.2. Note that HL1 implies hidden layer 1 and HL2 implies hidden layer 2.

Parameters HL1/HL2/LR	**Macro precision Mean (SD)**	**Macro recall Mean (SD)**	**Macro F1-score Mean (SD)**
128/128/0.005	30.966 (1.054)	33.323 (1.369)	31.303 (1.208)
256/64/0.01	31.323 (1.134)	33.822 (1.696)	31.248 (0.973)
128/256/0.001	30.922 (0.49)	33.396 (1.595)	31.18 (0.619)
64/64/0.001	31.102 (0.469)	33.673 (1.234)	31.019 (0.5)

4.1.2 Numeric data

The results from data set *numeric* evaluated on the simple-net can be found in table 4.3. The results from data set *numeric* evaluated on the deep-net can be found in table 4.4. The best obtainable macro F1-score (30.368%) using the simple-net on the numeric data set was with the setting of 256 neurons in the hidden layer and with a learning rate of 0.001. The best obtainable macro F1-score (30.795%) using the deep-net on the numeric data set was with the setting of 256 neurons in the first hidden layer, 64 neurons in the second hidden layer and with a learning rate of 0.001.

Table 4.3: Simple-net on numeric data. This table contains the top 3 results based on the macro F1-score. All hyper-parameter combinations is listed in table C.3. Note that HL1 implies hidden layer 1.

Parameters HL1/LR	**Macro precision Mean (SD)**	**Macro recall Mean (SD)**	**Macro F1-score Mean (SD)**
256/0.001	30.275 (0.51)	34.377 (0.895)	30.368 (0.612)
128/0.001	30.775 (0.516)	36.005 (0.822)	30.0 (0.75)
256/0.005	30.286 (0.937)	34.48 (1.658)	29.662 (1.342)

Table 4.4: Deep-net on numeric data. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations is listed in table C.4. Note that HL1 implies hidden layer 1 and HL2 implies hidden layer 2.

Parameters HL1/HL2/LR	**Macro precision Mean (SD)**	**Macro recall Mean (SD)**	**Macro F1-score Mean (SD)**
256/64/0.001	30.763 (0.677)	35.367 (0.974)	30.795 (0.85)
256/256/0.001	30.361 (0.373)	34.588 (1.153)	30.598 (0.372)
256/128/0.001	30.193 (0.638)	33.975 (1.271)	30.348 (0.7)
128/128/0.01	30.719 (0.74)	35.805 (0.827)	30.21 (0.775)

4.2 Triplet Classification

This section describes the re-admission classification results when the employed technique is triplet classification. This model evaluates on the triplet-data set, created from all databases mentioned in table 3.25 and table 3.26. Different combination of hyper-parameters descried in table 3.27 have been tried. The highest obtaining macro F1-scores and its hyper-parameter setting under each embedding method is presented in table 4.5. Figure 4.1 represents the mean and standard deviation over 10 iterations for each hyper-parameter model under each evaluated embedding method. From the figure it can be observed that the general trend of larger standard deviation comes in by increasing the learning rate. For lower learning rates, the model performance is close to mean in case of all embedding methods except for some exceptions in case of TransD. Table 4.6 states the hyper-parameters that obtained the highest mean macro F1-scores on test dataset over 10 iterations. Interestingly, TransD performs best on average and second best in case of overall highest, but doesn't perform better than baseline model.

Table 4.5: Hyper-parameter under each method and their macro F1-score on validation and test data, sorted on macro F1-score based on the test set. These hyper-parameters are the highest values under each embedding method.

Embedding method	**Dimension**	**Learning rate**	**Margin**	**Macro F1 Validation**	**Macro F1 Test**
TransH	20	0.001	2	0.2460	**0.2574**
TransD	50	0.001	2	0.2436	0.2553
TransE(L2)	20	0.01	1	0.2567	0.2543
TransR	50	0.01	2	**0.2647**	0.2524
TransE(L1)	50	0.01	2	0.2559	0.2470

Table 4.6: Hyper-parameter and their resulting test F1 scores based on highest average mean F1 score on 10 iterations. The overall best is highlighted in bold.

Embeddinhg Method	**Dimension**	**Learning Rate**	**Margin**	**Test F1 Score Mean(Std)**
TransE-L1	50	0.01	2	0.2469(0.0073)
TransE-L2	50	0.01	1	0.2449(0.0059)
TransH	20	0.001	2	0.2418(0.0110)
TransR	50	0.001	2	0.2405(0.0073)
TransD	50	0.0001	1	**0.2499(0.0079)**

Based on the macro F1-scores obtained from the test set, the globally best embedding method is TransH with hyper-parameter selection `{Dimension, learning rate, margin}` $= 20, 0.001, 2$. The maximum achieved macro F1-score is 0.2574.

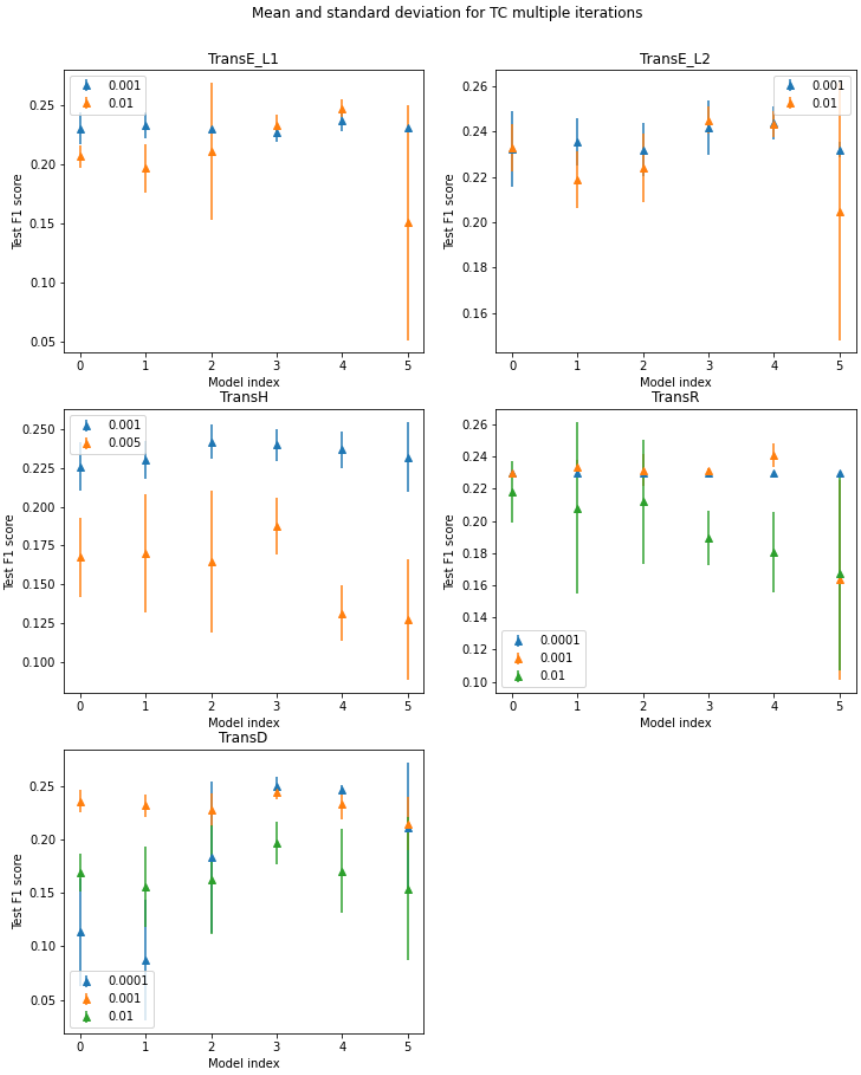
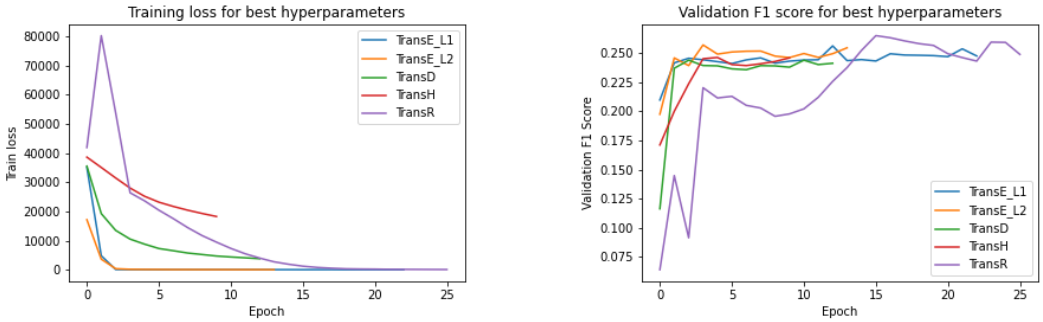


Figure 4.1: Mean and standard deviation for each model. In general, it can be observed that the standard deviations is small in case of smaller learning rate with an exception in case of TransD. The effect of smaller SD due to smaller LR can be explained by batching. A smaller learning rate forces the model to learn slow and thereby giving the facts of lower in frequency more opportunity to appear in different batches and there by having a significant contribution in backpropagation of loss. However, in case of large LR, the affect of facts with large in number dominates and the model falls in the traps of local best.

The training loss and validation macro F1-score for the hyper-parameters mentioned in table 4.5 is shown in figure 4.2. A general observation is that the training loss



(a) Training loss **(b)** Validation F1 macro score

Figure 4.2: (a)(b) Training loss and validation F1-score for the models described in table 4.5 respectively. It can be observed that knowledge graph embedding method along with learning rate have a significant impact on the number of training iterations. Further, all the methods tend to achieve similar F1-scores. A general trend is that the increase in model complexity results in decrease in number of training iterations. For example, TransD being most complex have 2^{nd} least training epochs and the simplest TransE L1 method has the 2^{nd} highest training iterations. The models complexity is generally increased with increase in the embedding spaces. TransR is one of the complex models that has high training iterations.

goes significantly low in very few epochs, but also depends on learning rate. The validation MRR score doesn't improve beyond 24%, on average for validation readmission triplets.

The macro F1-score on the test set for different hyper-parameter models is illustrated in figure 4.3. In general, all the translation models achieves competitive maximum macro F1-score on the test set. Based on their top result, no model outperform other model significantly. Nevertheless, it is observed that TransE models are not badly affected by changing hyper-parameters with drastic exceptions. It is also observed that the models doesn't have a steady learning for higher learning rates, they tend to generalize on a local affect rather than learning for global objective.

4.3 Link prediction

This section describes the re-admission classification results when the employed technique is link prediction followed by a feed-forward artificial neural network. The model trains in two phases, i.e. using the triplets defined in table 3.25. The resulting patient embeddings is then re-trained for the re-admission classification by the network architectures described in section 3.4.5. Table 4.7 states the test dataset MRR and hits @ 5 scores for different embedding methods. It can be seen that the TransE embedding method performs best for link prediction on test set. The method achieves MRR of 0.60 which could be inferred as predictions within top

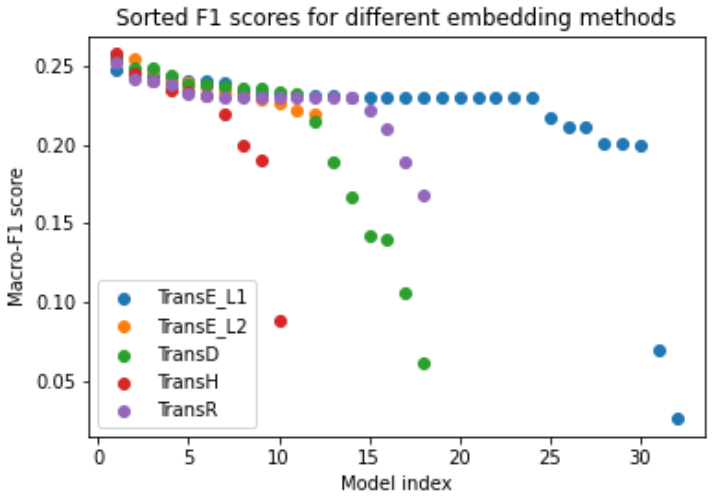


Figure 4.3: Macro F1-scores on test data for different embedding methods. All the methods obtain similar best F1 scores. The adverse change in F1 scores motivates the importance of hyperparameter tuning. The simple models like TransE L1 and TransE L2 are not much affected by different hyperparameters except $1-2$ outliers. TransR is one of the complex models which produces stable F1 scores.

2 on an average.

Figure 4.4 illustrates the logarithm of training loss during the link prediction training (phase -1). The hyper-parameters for these curves is stated in table 3.28. It can be observed that the network reaches a steady state.

Table 4.7: Test MRR and H@5 scores for the hyperparameters mentioned in table 4.4. The highest achieved MRR is highlighted.

Method	**Embedding Dimension**	**Learning Rate**	**Margin**	**Test MRR**	**Test H@5**
TransE L1	50	0.01	1	0.3920	0.4761
TransE L2	50	0.001	1	**0.6023**	0.6555
TransH	50	0.005	0.5	0.4254	0.5099
TransR	50	0.0001	2	0.1823	0.2378
TransD	50	0.001	1	0.2509	0.3926

Figure 4.5(a)(b) illustrates the validation mean reciprocal rank and hits @ 5. It can be observed from either plot that the different embedding methods tend towards steady state, with TransE L2 model outperforming the others significantly.

Table 4.9 describes the macro F1-score for the TransE embedding method with *L1* dissimilarity. The tables states different configurations/hyper-parameters of feed-forward artificial neural networks tried to obtain best fitting for predicting readmission risk. Similarly, other embedding methods results are stated in following

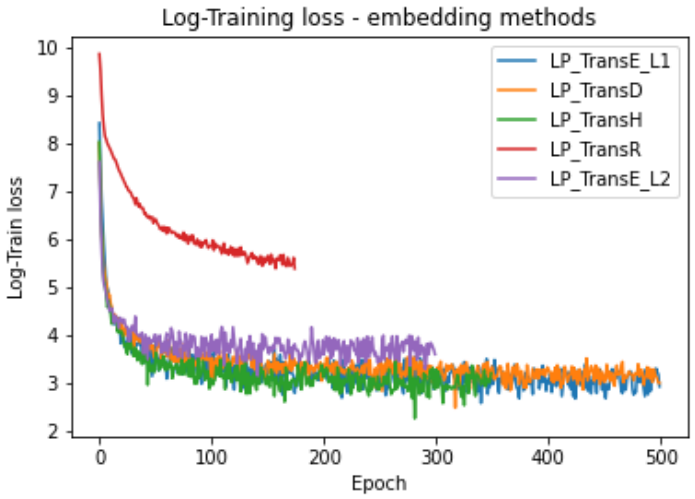
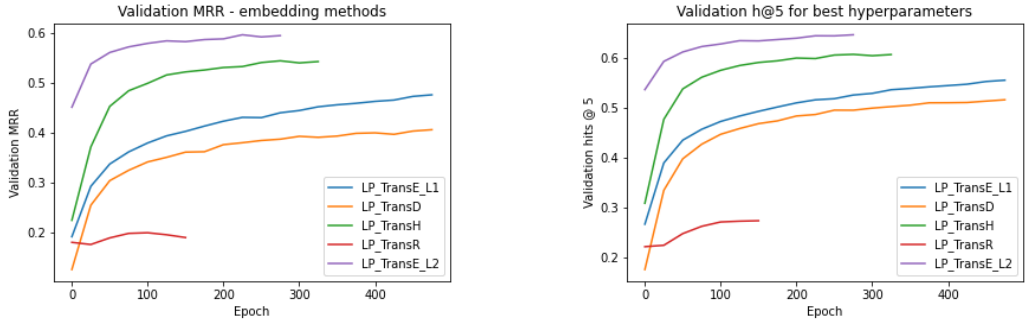


Figure 4.4: Logarithm of training loss for different embedding method for link prediction training. Most of the models obtains steady state training loss after ≈ 150 iterations. This is mainly due to model learning the facts with large frequencies much quicker than the facts that occur in smaller in number. The lowest training loss is not significantly different in all the models except TransR. Surprisingly, TransR model stops to train with much higher training loss. TransR is highly complex model with both head and tail entities being projected to different relation space to obtain embeddings that can capture different aspects of entities. Large batches might result in loss of this property resulting in a poor training.



(a) Validation MRR **(b)** Validation hits @5

Figure 4.5: (a)(b) Validation MRR and hits @5 respectively evaluated on validation knowledge graph. The simpler models achieved the best results i.e. TransE and TransH. The dataset contains varied relations types, but, many entity types falls under $1:1$ category. Only the disease and drugs are of type $M:N$ relations. This leads to the explanation of simple models outperforming other complex one's as they are known to capture simple relations very well.

tables: for TransE using L2-dissimilarity, we refer to 4.10 when the embeddings are fed to the simple-net and to 4.11 when feeding the embedding to the deep-net. For TransH, we refer to 4.12 when the embeddings are fed to the simple-net, and to 4.13

when feeding the embeddings to the deep-net. For TransR, we refer to 4.14 when the embeddings are fed to the simple-net, and to 4.15 when feeding the embeddings to the deep-net. For TransD, we refer to 4.16 when the embeddings are fed to the simple-net and to 4.17 when they're fed to the deep-net. It can be observed that the TransD deep-net model slightly outperforms other embeddings, but not significantly.

Table 4.8: Best L1-TransE-embeddings fed to simple-net. This table contains the top 3 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.5.

Parameters HL1/LR	**Macro Precision Mean (SD)**	**Macro Recall Mean (SD)**	**Macro F1 Score Mean (SD)**
64/0.01	26.738 (0.822)	28.02 (0.986)	24.352 (1.69)
256/0.005	26.169 (0.344)	27.682 (0.862)	23.969 (0.656)
256/0.01	26.172 (0.369)	27.458 (0.852)	23.766 (0.733)

Table 4.9: Best L1-TransE-embeddings fed to deep-net. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.6

Parameters HL1/HL2/LR	**Macro precision Mean (SD)**	**Macro recall Mean (SD)**	**Macro F1-score Mean (SD)**
64/128/0.005	27.037 (1.137)	28.511 (0.863)	26.558 (1.128)
64/64/0.01	27.759 (2.223)	27.435 (0.511)	26.435 (0.687)
64/128/0.01	27.273 (1.294)	28.063 (0.88)	26.054 (0.77)
64/256/0.01	27.089 (1.71)	27.797 (1.219)	26.023 (0.596)

Table 4.10: Best L2-TransE-embeddings fed to simple-net. This table contains the top 3 results based on the macro F1 score. All hyper-parameter combinations are listed in the table C.7.

Parameters HL1/LR	**Macro Precision Mean (SD)**	**Macro Recall Mean (SD)**	**Macro F1 Score Mean (SD)**
64/0.01	29.044 (0.413)	31.425 (0.482)	27.391 (0.602)
64/0.005	28.71 (0.259)	32.24 (0.535)	26.913 (0.686)
128/0.01	28.715 (0.61)	31.472 (0.999)	26.819 (0.719)

Table 4.11: Best L2-TransE-embedding fed to deep-net. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.8.

Parameters HL1/HL2/LR	Macro precision Mean (SD)	Macro recall Mean (SD)	Macro F1-score Mean (SD)
64/128/0.001	28.916 (0.458)	32.099 (1.011)	27.61 (0.831)
256/256/0.005	29.68 (1.857)	31.812 (1.079)	27.592 (0.884)
64/64/0.001	28.653 (0.391)	31.647 (0.815)	27.305 (0.841)
128/64/0.005	29.506 (0.84)	31.363 (0.834)	27.287 (1.009)

Table 4.12: Best TransH-embedding fed to simple-net. This table contains the top 3 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.9.

Parameters HL1/LR	Macro Precision Mean (SD)	Macro Recall Mean (SD)	Macro F1 Score Mean (SD)
64/0.005	27.928 (0.605)	29.933 (1.172)	26.265 (0.605)
256/0.005	27.516 (0.395)	29.903 (1.019)	25.948 (0.486)
128/0.005	28.03 (0.843)	30.479 (1.351)	25.897 (0.817)

Table 4.13: Best TransH-embedding fed to deep-net. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.10.

Parameters HL1/HL2/LR	Macro precision Mean (SD)	Macro recall Mean (SD)	Macro F1-score Mean (SD)
256/128/0.005	28.337 (0.468)	31.283 (0.938)	26.865 (1.019)
64/64/0.005	27.853 (0.531)	30.123 (1.201)	26.78 (0.6)
64/128/0.005	27.919 (0.637)	29.746 (0.977)	26.698 (1.039)
128/256/0.001	28.037 (0.329)	30.897 (1.141)	26.554 (0.609)

Table 4.14: Best TransR-embedding fed to simple-net. This table contains the top 3 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.11.

Parameters HL1/LR	Macro Precision Mean (SD)	Macro Recall Mean (SD)	Macro F1 Score Mean (SD)
64/0.01	26.387 (0.319)	27.317 (0.853)	24.32 (0.82)
64/0.005	26.435 (0.528)	27.611 (1.33)	24.005 (1.139)
256/0.005	25.855 (0.362)	26.255 (0.827)	23.68 (0.555)

4.4 Computational Time

All the computations related to triplet classification and link prediction where done using the “Tesla V100-SXM2-16GB“ which is a powerful GPU provided by Google for colab pro users.

Table 4.15: Best TransR-embeddings fed to deep-net. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.12.

Parameters HL1/HL2/LR	Macro precision Mean (SD)	Macro recall Mean (SD)	Macro F1-score Mean (SD)
64/64/0.005	27.157 (0.519)	28.443 (1.01)	26.065 (0.861)
64/128/0.001	27.0 (0.353)	28.723 (0.908)	25.34 (0.676)
64/64/0.001	26.823 (0.332)	28.392 (0.984)	25.129 (0.953)
64/256/0.001	26.887 (0.722)	28.286 (0.97)	25.048 (1.408)

Table 4.16: Best TransD-embedding fed to simple-net. This table contains the top 3 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.13.

Parameters HL1/LR	Macro Precision Mean (SD)	Macro Recall Mean (SD)	Macro F1 Score Mean (SD)
64/0.005	29.06 (0.54)	33.032 (1.087)	27.896 (0.829)
64/0.01	28.885 (0.653)	32.371 (1.002)	27.392 (0.748)
128/0.01	28.72 (0.582)	32.19 (1.102)	27.193 (0.866)

Table 4.17: Best TransD-embeddings fed to deep-net. This table contains the top 4 results based on the macro F1-score. All hyper-parameter combinations are listed in the table C.14.

Parameters HL1/HL2/LR	Macro precision Mean (SD)	Macro recall Mean (SD)	Macro F1-score Mean (SD)
64/64/0.005	30.255 (0.94)	33.262 (0.745)	28.785 (0.625)
128/64/0.005	29.604 (0.774)	32.807 (0.657)	28.541 (0.651)
64/128/0.001	29.398 (0.545)	33.725 (1.057)	28.41 (0.912)
256/64/0.005	29.555 (1.252)	32.692 (0.657)	28.395 (0.748)

Triplet Classification Computational Times

The validation in case of triplet classification involves computing dissimilarity value for the readmission status for all patients in test sets. Further, F1 score is computed based on the prediction obtained through dissimilarity value. The plot for computational time is shown in figure 4.6.

Link Prediction Computational Times

The validation in case of link prediction involves computing dissimilarity of all possible entities and rank them. The true entity's rank then contributes in computing mean reciprocal rank. Hence, the link prediction validation times is much higher as compared to triplet classification. Table 4.18 states the training times for link prediction. Figure 4.7 illustrates the mean and SD of the validation times for each of these methods.

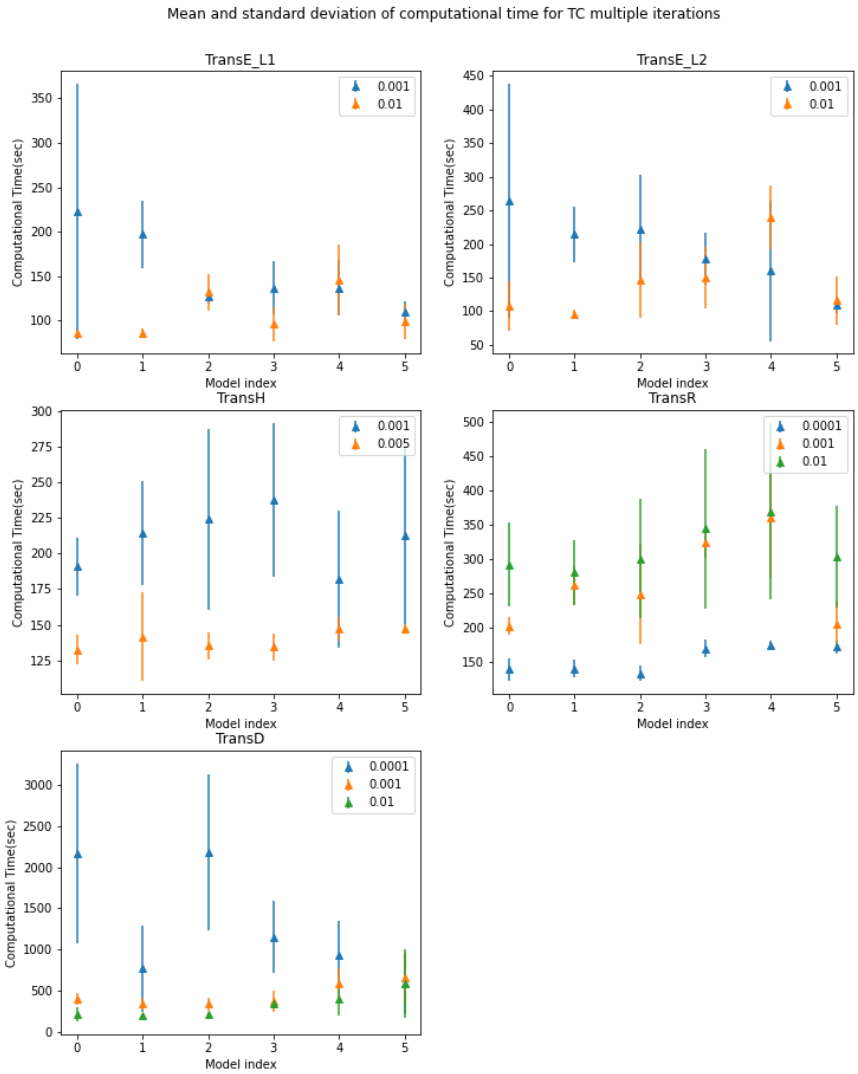


Figure 4.6: The figure illustrates the computational times for triplet classification. The general trend is higher computational time with increasing model complexity. And also, as expected the smaller learning rates in general took longer training times, except for the TransR method. Some of the highest computational time has been observed in case of TransD, but it is expected as TransD is complex model. One explanation for the anomaly in case of TransR is the large batch size. The large batch size may lead to model change the parameters abruptly after each batch process. Ultimately, the model fluctuates in terms of F1 scores and takes longer to reach early stoppage criteria. This can also be advocated with large standard deviations in the same TransR method for LR = 0.01. Note that, 10 iterations were considered to obtain the mean and standard deviation in each case.

Table 4.18: The table illustrates computational times for link prediction embedding methods. The validation contributes to greater than 95% of the training time. Due to this validation is only done after every 25 epochs. Hence, it is interesting to compare the average validaiton time taken in one validation iteration as compared to total training time. The reason bein that the total training time depends on learning rate which is not same in all 5 cases. From the table, it can be noticed that the average validation time increases with increase in model complexity which is expected.

Method	**Total training time**	**Avg val time per epoch**	**Total Epochs**	**Validaiton Epochs**
TransE L2	19537.083	1001.343	475	19
TransE L1	8208.924	1134.779	175	7
TransH	5342.791	1690.469	75	3
Trans R	6986.845	2216.474	75	3
TransD	19354.325	2018.863	200	8

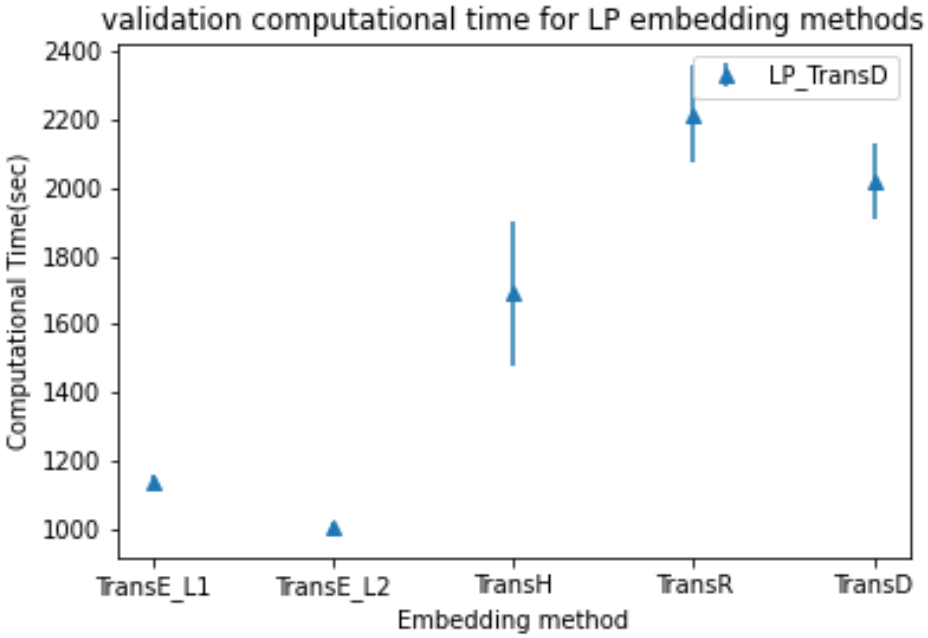


Figure 4.7: The figure illustrates the mean and standard deviation of the computational times of validation function in case of link prediciton embedding methods. It can be observed that the simple embedding methods have small SD, as compared to complex ones. The figure also illustrates the increasing validation time with increase in the model complexity. The methods in x-axis are sorted based on increasing complexity of the method.

5
Discussion

This section will present the discussions of the results from the previous section, as well as the methods used in this project. We will also try to answer why the results turned out as they did, as well as reason why we observe the particular performance and behaviour of the models. We will discuss possible limiting factors, as well as suggest what future work could be done on the topics, before finishing with the contributions made with this project.

5.1 Discussion of re-admission classification tasks

As seen in section 4.1, 4.2 and 4.3, it can be argued that none of the employed technique significantly outperforms the baseline model when evaluated on the one-hot data set. Nevertheless, the best performing models in either case are competitive in terms of the re-admission predicting capability. Below, we make an analysis to discover the probable reasons for poor model performance.

Cosine Similarity
Cosine similarity measures the cosine of angle between two vectors to produce a measure that can be interpreted as a similarity measure. The cosine similarity is generally used in positive spaces and lies in $[0, 1]$. Here a value at 0 implies very dissimilar as the angle between the vectors are almost perpendicular. On the other hand, a value of 1 implies almost parallel vectors. In a one hot encoded data, the vectors are boolean making the cosine similarity a good measure to compute the similarity between them. We use the patient's one-hot encoded information to compute pairwise cosine similarity index within each of the readmission status. The similarity measure can provide insights on how similar are two any two randomly picked patients within same readmission status.

Figure 5.1, 5.2 and 5.3 illustrates the cosine similarity measure by considering all patients pairwise within each readmission class. Table 5.1 states the mean μ and standard deviation σ of the pairwise cosine similarity within each readmission status group. The mean being less than 0.25 in all three cases and standard deviation less than 0.1 clearly indicates that most of the pairwise comparisons ended up being highly dissimilar to each other. Hence, it can be said that, it is very difficult for the model to learn a pattern that can be applied to classify patients re-admission statuses. Moreover, features being categorical adds additional degree to the difficulty for the model to generalize.

Table 5.1: The table states the cosine similarity's mean and standard deviation within each readmission status. The mean and standard deviation is computed by considering all possible pairwise patients within each of the stated readmission status.

Readmission Status	**Mean**	**SD**
Re-admission within 30 days	0.249	0.09
Re-admission within 90 days	0.241	0.08
Re-admission within 365 days	0.237	0.09

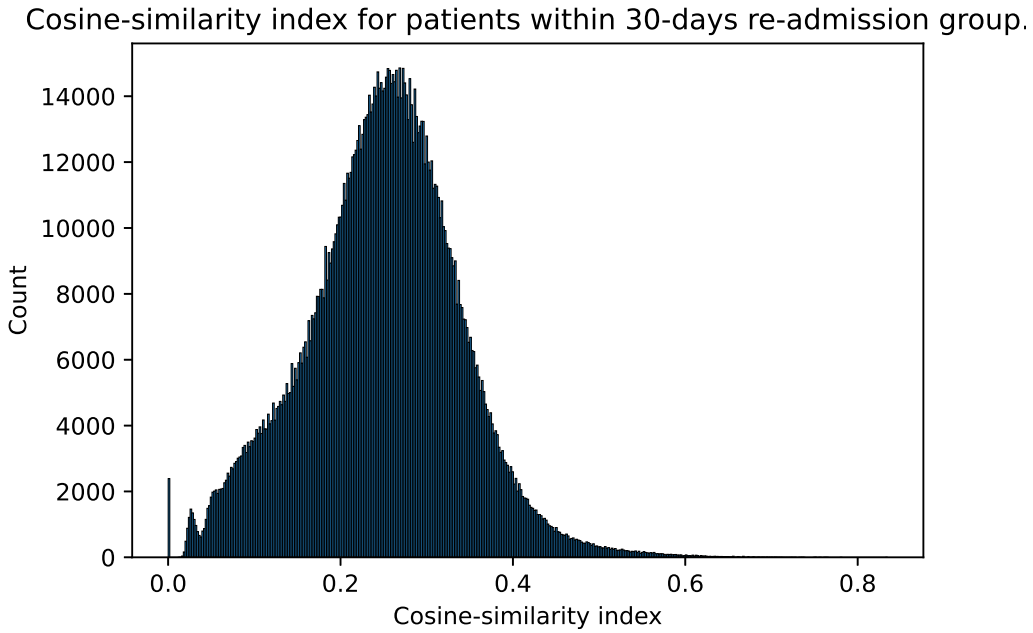


Figure 5.1: The figure illustrates the cosine similarity index for all pairwise-comparisons within patient with re-admission within 30 days. It can be observed that most of the measurements are centred around 0.25 indicating a low similarity score.

The embeddings obtained are latent vector representations of high dimensional sparse vectors. But still, it is difficult to visualize the embeddings. TSNE with perplexity = $\sqrt{\text{Embedded feature dimension}}$, and PCA are the two techniques employed to visualize the spatial positioning of these embeddings. Both Principal Component Analysis (PCA) and t-distribution Stochastic Neighbour embedding (TSNE) are generally used to represent high dimensional data in lower dimensions. It can be observed from figures 5.4, 5.5, 5.6, 5.7 and 5.8, that irrespective of embedding

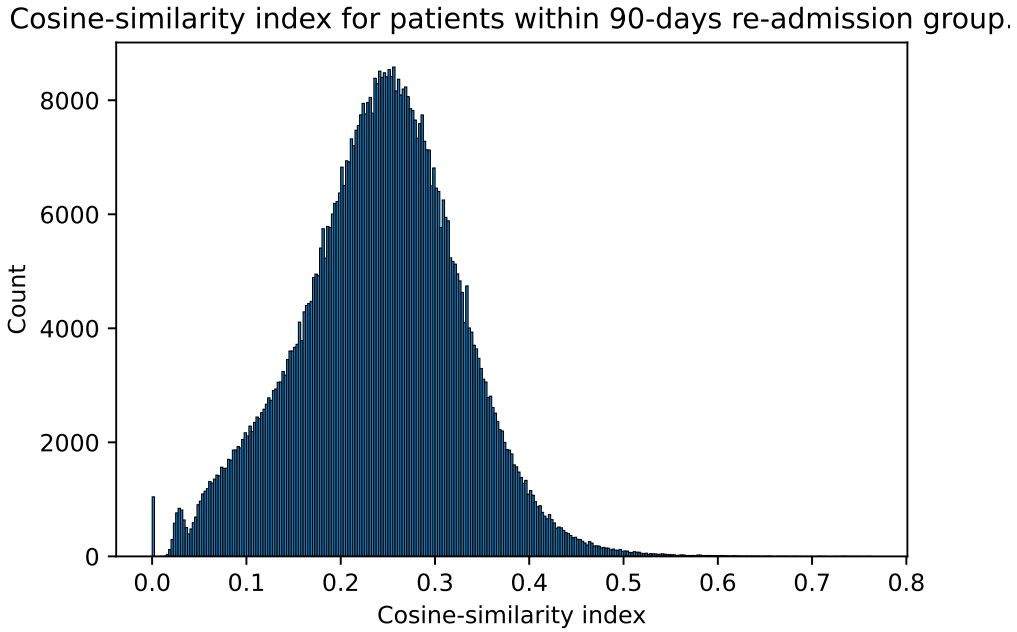


Figure 5.2: The figure illustrates the cosine similarity index for all pairwise-comparisons within patient with re-admission within 90 days. It can be observed that most of the measurements are centred around 0.25 indicating a low similarity score.

method, constructing a decision boundary that can distinguish different readmission classes appears to be extremely difficult. It can be note worthy to state that the variance captured by top 3 principal component amounted to just 18% indicating that these illustrations doesn't necessarily capture the true spread of embeddings. As seen in the figures, the embeddings do not cluster the patients accordingly to the re-admission statuses, but rather places them in a single blob with no distinct pattern recognizable. An optimal embedding would be presenting visually separable clusters, where each re-admission status would be distinct from the other. This would have then enabled the possibility for better predictive ability for the models investigated in the project.

In relation to predicting missing links i.e. link prediction - from table 4.7, we could infer that some models do perform very well in test graph structures for predicting missing links. TransE with L2-norm achieves better performance for link prediction tasks than the other evaluated methods. This can be motivated by the fact that our knowledge graph contains large number of relations which favours translational models, especially TransE favours lower `facts:relations` ratio. Nevertheless, there are other relation-cardinalities present as well. TransH which is aimed on capturing these $M : N$ relations gives results close to TransE, i.e. the predicted entities ending up in top 2 on an average. In general, it is observed that simple embedding methods have performed better than more complex ones for link prediction.

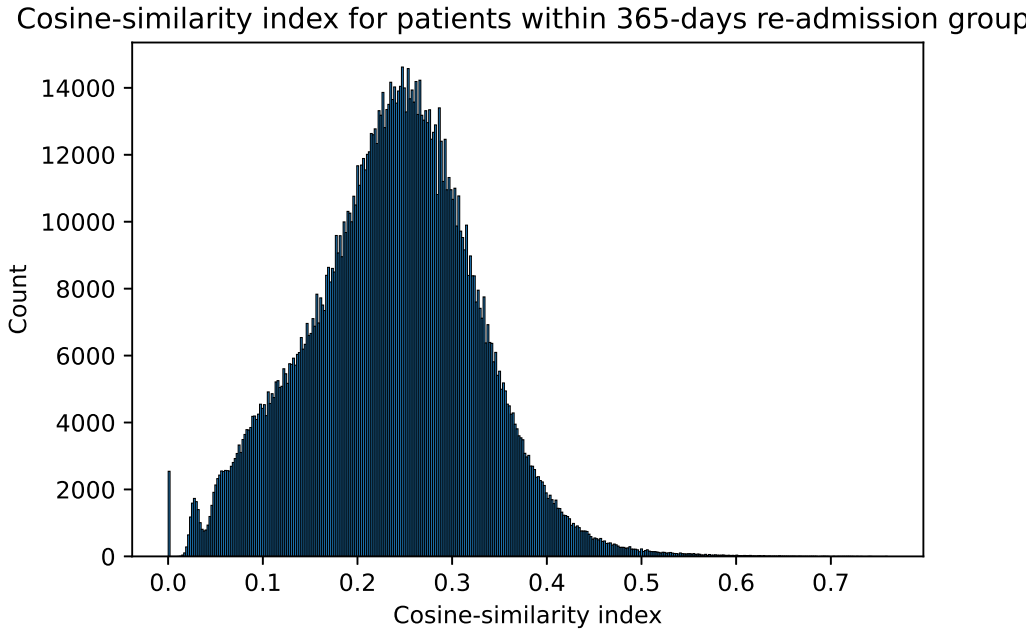


Figure 5.3: The figure illustrates the cosine similarity index for all pairwise-comparisons within patient with re-admission within 365 days. It can be observed that most of the measurements are centred around 0.25 indicating a low similarity score.

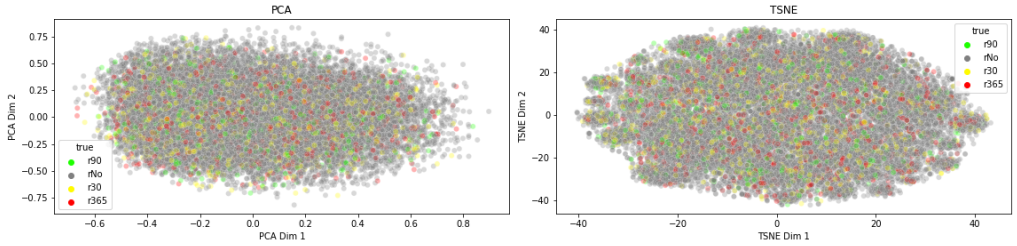


Figure 5.4: The figure illustrates PCA and TSNE for TransE embedding method with L1 dissimilarity. It can be observed that both PCA and TSNE, the spread of the readmission statuses is very wide. This implies creating a decision boundary capturing general properties is difficult.

5.2 Limiting factors

In this section, we will briefly describe what limiting factors could have been preventing the models to perform better. We'll try to give a qualitative explanation as well as reasoning behind this behaviour.

5.2.1 Data insufficiency

A possible reason for not obtaining greater results can be because of the insufficient quality of the data used. All data sources investigated comes with their own terms

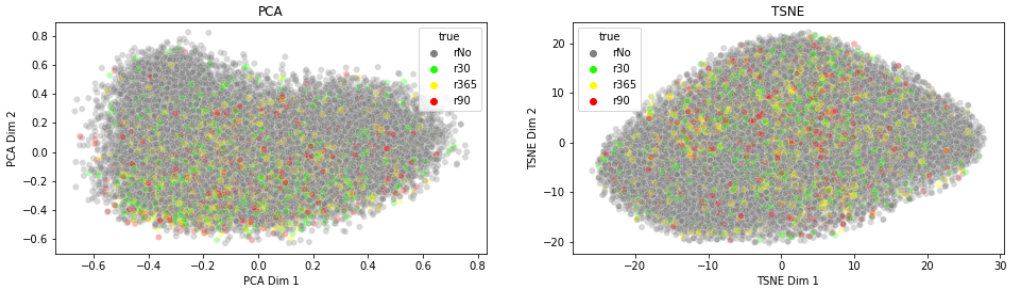


Figure 5.5: The figure illustrates PCA and TSNE for TransE embedding method with L2 dissimilarity. A large mass of data points within all readmission status appear to get closer, but distinguishing them with other status is still not significant.

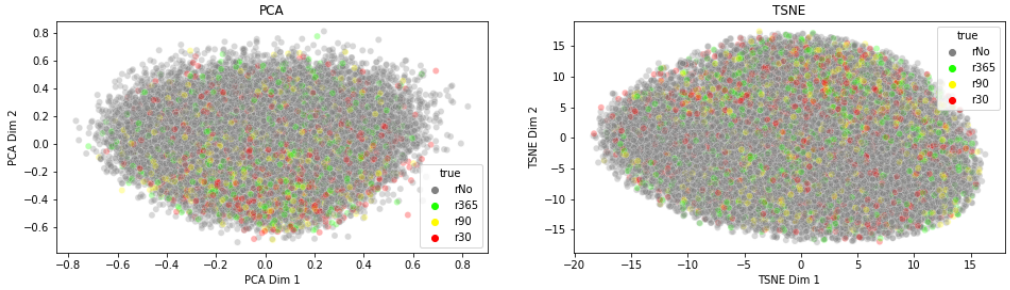


Figure 5.6: The figure illustrates PCA and TSNE for TransH embedding method. The concentration of readmission status data points is relatively similar as compared to TransE L2.

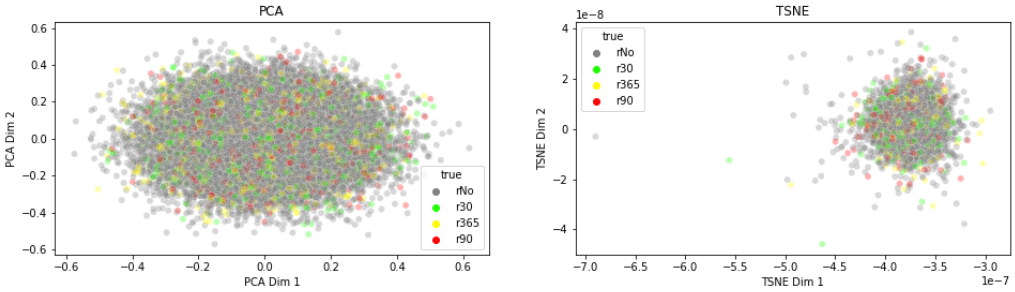


Figure 5.7: The figure illustrates PCA and TSNE for TransR embedding method. The spread of different readmission status is very wide, similar to TransE L1 method.

of noise, which can lead to an aggregated noise level which worsens the models performance to correctly classify re-admissions. Since no available data set of this sort was found, the time-consuming task of creating our own data set was necessary. A positive attribute that came with this was the freedom with selection of data points that could be chosen, as well as having full insight into how the cohort is selected. However, it also imposed with certain difficulties. There's no recognized baseline-methodology that could be implemented and therefore had to be created. Whether or not this baseline-methodology is appropriate or not is unknown until investigated.

In order to make the numerical values fit into the framework of knowledge graph

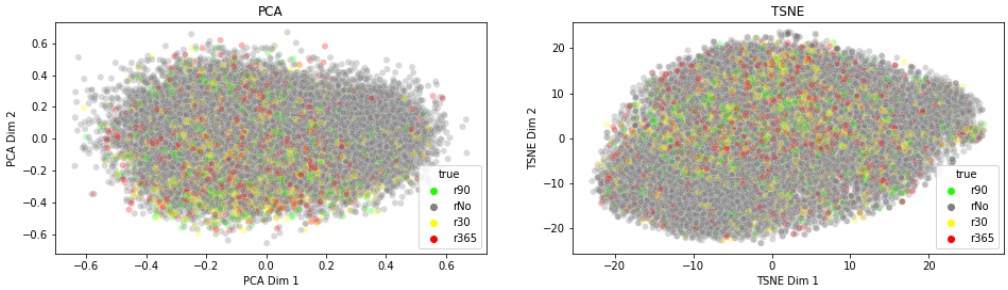


Figure 5.8: The figure illustrates PCA and TSNE for TransD embedding method. The concentration of data points is higher, but again is not sufficient enough to distinguish within each readmission status and the overlay with no readmission is quite high.

embedding methods, they needed to be binned in order to be represented as an entity. Binning these values is not straight-forward and can be conducted in many different ways. Many of our binning were done ad-hoc and could possibly be an source of data insufficiency. Trying different approaches to bin these in different intervals and counts could perhaps yield a better predictive ability to the models investigated. However, due to lack of time, this was not investigated further in the project. Moreover, binning causes a loss in structural property of numbers i.e. for example binning age loses the characteristic of numeric age.

Furthermore, many of the investigated features that was more naturally appropriate for the construction of a knowledge graph came with rather heavy class-imbalances. Most features had several classes with quite few observations. The selection of classes within features and their mappings were constructed ad-hoc, resulting in many classes called *other* or *unknown*, which can also possibly be a source of data insufficiency. A more stringent methodology regarding this task was however not implemented due to lack of time.

5.2.2 Data-linkage

All data sources investigated comes with their own terms of error and noise, and greater risk for noisy observations are presented when trying to link these together. Some publicly available data sources are algorithmically generated and are prone to error by some degree[14]. Moreover, the disease classification *ICD* for healthcare is designed to record appropriate statistics and billing, but the research publications of treating disease relates to either a part of a disease or sometimes relates to multiple diseases too making it inherently difficult to give equal weightage to all facts. Different concentrations of drugs have different effects on the body, but concentrations were ignored in this thesis to obtain facts linked with standard nomenclatures. This can also lead to significant loss of information, and in general requires detailed or expertise support as different drugs have different values to varying levels of concentrations. Lastly, the number of drug-disease connections found covered only a fraction of diseases. Problems related to data-linkage is common in this field of

research[17].

5.3 Future work

In this section, we will present some topics that were encountered that can be further investigated in future work for benefiting the model's regarding re-admission classification ability. These topics was not investigated in this project due to lack of time, but will rather be presented here with the motivation that they can be fully utilized in future work.

5.3.1 Customized loss-function

We observe in section 4.1 and 4.3 that the embeddings generated from TransD gave comparable results with the one-hot data classification when feeding them through the artificial neural network. However, we want to add weights to each facts in an algorithmic way. Adding weights to the knowledge graph embeddings would reduce the bias created by relations that exist in much larger frequency. This may potentially increase the training times largely, but one can expect that it will lead to a significant improvement in the performance of the model. The initial thoughts to incorporate weights comes with the weighted loss function, where depending on the relation or fact a weight factor can be multiplied to loss before back-propagation.

5.3.2 Expanding knowledge-graph

There is a quite extensive selection of different biomedical data sources and ontologies that could be further investigated and included in the knowledge graph. For example, including drug-target interactions, drugs and their side-effects, protein-interaction and such could possibly improve the embeddings produced by the different methods. However, this leads to an drastic increase in complexity regarding the knowledge graph embedding models, which then imposes a natural down-side.

Also, including time-series variables could possibly improve the predictive ability of the classification algorithms. However, these needs to be carefully investigated - binning numerical values is difficult, and representing time-series data as entities is even more problematic. An idea could be to bin these based on the mean-value for a given time-range. However, this would then impose the question of choosing such a time-range, as well as deciding how to select the cohort depending on this, which cut-off value should be chosen and how to deal with patients with several time-series measurements of the same type within the same admission.

5.4 Contributions

We believe that the thesis has connected different research questions and addressed at least one method to answer those research questions. Some examples of this thesis

has contributed to research are listed below.

- The thesis had made an attempt to connect healthcare data with research based biomedical ontologies.
- The thesis has explored the recently developed embedding methods for structured and relational data.
- The thesis has implemented a down-stream pipleline connecting the creation of multi-informational knowledge graphs to embeddings methods further on to classification algorithms to classify re-admission statuses.

6
Conclusion

The rapid increase of electronic health records have opened up for the possibility for patient outcome predictions, utilizing recorded information from patient's admissions to classify re-admissions within given time-intervals. The incorporation of domain-specific information in a knowledge-graph based structure can be further down-streamed for this particular type of classification problem.

We have found that this type of setting yields results that are comparable to the proposed baseline-methodology. Despite imposing some difficulties in modeling this type of structure, there are room for improvements so that this setting can ultimately out-perform the baseline-methodology.

Since the work in the thesis obtained results that is not superior to the baseline-methodology, it successfully created a pipeline to address the challenge. The thesis has also opened doors to investigate similar problems.

Nevertheless, the employed techniques presents an interesting set of theories. For example, the embeddings trained using link prediction can be used for a multitude of prediction or other type of problems. For example, a well trained link prediction problem can help in predicting alternate drugs for a patient based on dissimilarity score.

Modeling patient health information is guided by complex principles and finding the features that achieve state of the art results is inherently difficult. With this thesis we could conclude that the features selected to model patient's re-admission status needs more information and possibly incorporation of patient vital signs.

Bibliography

[1] International classification of diseases : [9th] ninth revision, basic tabulation list with alphabetic index. https://apps.who.int/iris/handle/10665/39473. Accessed: 2021-06-07.

[2] Using knowledge graph learning to predict and explain patient outcomes in electronic health records. https://gtr.ukri.org/project/DB58560B-6C53-45EF-A5EA-052E4438318E. Accessed: 2021-05-25.

[3] Overview of the pharmgkb. https://www.pharmgkb.org/page/overview. Accessed: 2021-06-07.

[4] Us national libraries of medicine. pubmed. https://www.ncbi.nlm.nih.gov/pubmed/. Accessed: 2021-05-25.

[5] Unified medical language system. about snomed-ct. https://www.nlm.nih.gov/research/umls/sourcereleasedocs/2009AA/SNOMEDCT/index.html. Accessed: 2021-05-26.

[6] Stanford cs320. https://cs230.stanford.edu/section/4/. Accessed: 2021-05-25.

[7] G. AH. Hhs steps up hipaa audits. https://www.ncbi.nlm.nih.gov/pubmed/. J AHIMA. 2011 Oct;82(10):58-9. PMID: 22029216.

[8] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, and J. Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework, 2020.

[9] I. Bartoletti. Ai in healthcare: Ethical and privacy challenges. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 7–10. Springer, 2019.

[10] B. K. Beaulieu-Jones and C. S. Greene. Semi-supervised learning of the electronic health record for phenotype stratification. *Journal of Biomedical Informatics*, 64:168–178, 2016. ISSN 1532-0464. doi:https://doi.org/10.1016/j.jbi.2016.10.007.

[11] F. Bianchi, G. Rossiello, L. Costabello, M. Palmonari, and P. Minervini. Knowledge graph embeddings and explainable ai. *ArXiv*, abs/2004.14843, 2020.

[12] J. Bleich, B. Cole, A. Kapelner, C. Baillie, R. Gupta, A. Hanish, E. Calgua, C. Umscheid, and R. Berk. Using random forests with asymmetric costs to predict hospital readmissions, 03 2021.

[13] O. Bodenreider. The unified medical language system (umls): Integrating biomedical terminology. *Nucleic acids research*, 32:D267–70, 02 2004. doi:10.1093/nar/gkh061.

[14] S. Bonner, I. P. Barrett, C. Ye, R. Swiers, O. Engkvist, A. Bender, C. T. Hoyt, and W. Hamilton. A review of biomedical datasets relating to drug discovery: A knowledge graph perspective, 2021.

[15] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[16] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[17] C. Bradley, L. Penberthy, K. Devers, and D. Holden. Health services research and data linkages: Issues, methods, and directions for the future. *Health services research*, 45:1468–88, 10 2010. doi:10.1111/j.1475-6773.2010.01142.x.

[18] W. Dai, T. S. Brisimi, W. G. Adams, T. Mela, V. Saligrama, and I. C. Paschalidis. Prediction of hospitalization due to heart diseases by supervised learning methods. *International Journal of Medical Informatics*, 84(3):189–197, 2015. ISSN 1386-5056. doi:https://doi.org/10.1016/j.ijmedinf.2014.10.002.

[19] X. L. Dong. Building product graphs automatically. `https://www.amazon.science/blog/building-product-graphs-automatically`. Accessed: 2021-06-07.

[20] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

[21] J. Feng. Knowledge graph embedding by translating on hyperplanes. 06 2014.

[22] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, 01 2010.

[23] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15, 01 2010.

[24] F. Gong, M. Wang, H. Wang, S. Wang, and M. Liu. Smr: Medical knowledge graph embedding for safe medicine recommendation, 2020.

[25] D. Holland, L. Rhudy, C. Vanderboom, and K. Bowles. Feasibility of discharge planning in intensive care units: a pilot study. *American journal of critical care : an official publication, American Association of Critical-Care Nurses*, 21 4: e94–e101, 2012.

[26] K. Huang, J. Altosaar, and R. Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission, 2020.

[27] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL*, 2015.

[28] A. Johnson, T. Pollard, L. Shen, L.-w. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Celi, and R. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 05 2016. doi:10.1038/sdata.2016.35.

[29] A. Johnson, T. Pollard, L. Shen, L.-w. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Celi, and R. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 05 2016. doi:10.1038/sdata.2016.35.

[30] M. Kamdar and M. Musen. An empirical meta-analysis of the life sciences linked open data on the web. *Scientific Data*, 8, 01 2021. doi:10.1038/s41597-021-00797-y.

[31] M. R. Karim, M. Cochez, J. B. Jares, M. Uddin, O. Beyan, and S. Decker. Drug-drug interaction prediction based on knowledge graph embeddings and convolutional-lstm network. BCB '19, page 113–123, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366663. doi:10.1145/3307339.3342161.

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[33] G. Kong, K. Lin, and Y. Hu. Using machine learning methods to predict in-hospital mortality of sepsis patients in the icu. *BMC medical informatics and decision making*, 20:251, 10 2020. doi:10.1186/s12911-020-01271-2.

[34] Q. Li, L. Li, J. Zhong, and L. F. Huang. Real-time sepsis severity prediction on knowledge graph deep learning networks for the intensive care unit. *Journal of Visual Communication and Image Representation*, 72:102901, 2020. ISSN 1047-3203. doi:https://doi.org/10.1016/j.jvcir.2020.102901.

[35] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[36] S. M. Marinka Zitnik, Rok Sosič and J. Leskovec. BioSNAP Datasets: Stanford biomedical network dataset collection. `http://snap.stanford.edu/biodata`, 08 2018.

[37] M. Matlakala, M. Bezuidenhout, and A. Botha. Challenges encountered by critical care unit managers in the large intensive care units. *Curationis*, 37: E1–7, 02 2014. doi:10.4102/curationis.v37i1.1146.

[38] B. (MD). Metathesaurus - rich release format (rrf). `https://www.ncbi.nlm.nih.gov/books/NBK9685/`. Accessed: 2021-06-07.

[39] B. Mehlig. Machine learning with neural networks, 2021.

[40] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, Jan 2016. ISSN 1558-2256. doi:10.1109/jproc.2015.2483592.

[41] F. Ozair, J. Nayer, A. Sharma, and P. Aggarwal. Ethical issues in electronic health records: A general overview. *Perspectives in clinical research*, 6:73–6, 04 2015. doi:10.4103/2229-3485.153997.

[42] E. Rocheteau, C. Tong, P. Veličković, N. Lane, and P. Liò. Predicting patient outcomes with graph representation learning, 2021.

[43] F. B. Rogers. Medical subject headings. *Bulletin of the Medical Library Association*, 51(1):114–116, 1963.

[44] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag. Learning a health knowledge graph from electronic medical records. *Scientific Reports*, 7, 12 2017. doi:10.1038/s41598-017-05778-z.

[45] J. Schaefer, M. Lehne, J. Schepers, F. Prasser, and S. Thun. The use of machine learning in rare diseases: a scoping review. *Orphanet Journal of Rare Diseases*, 15, 12 2020. doi:10.1186/s13023-020-01424-6.

[46] L. Schriml, E. Mitraka, J. Munro, B. Tauber, M. Schor, L. Nickle, V. Felix, L. Jeng, C. Bearer, R. Lichenstein, K. Bisordi, N. Campion, B. Hyman, D. Kurland, C. Oates, S. Kibbey, P. Sreekumar, C. Le, M. Giglio, and C. Greene. Human disease ontology 2018 update: classification, content and workflow expansion. *Nucleic acids research*, 47, 11 2018. doi:10.1093/nar/gky1032.

[47] A. Signhal. Introducing the kg: things, not strings. `https://blog.google/products/search/introducing-knowledge-graph-things-not`, 2012.

[48] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. Goldberg, K. Eilbeck, A. Ireland, C. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. Scheuermann, N. Shah, P. Whetzel, and S. Lewis. The obo foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25:1251–5, 12 2007. doi:10.1038/nbt1346.

[49] X. Song, A. Mitnitski, J. Cox, and K. Rockwood. Comparison of machine learning techniques with classical statistical models in predicting health outcomes. *Studies in health technology and informatics*, 107:736–40, 02 2004. doi:10.3233/978-1-60750-949-3-736.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014. ISSN 1532-4435.

[51] J.-F. Timsit, J. Aboab, and J.-J. Parienti. Is research from databases reliable? yes. *Intensive Care Medicine*, 45, 12 2018. doi:10.1007/s00134-018-5436-x.

[52] A. Vieira. About genes. https://www.nature.com/scitable/topic/genes-and-disease-17/. Accessed: 2021-05-25.

[53] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph and text jointly embedding. pages 1591–1601, 01 2014. doi:10.3115/v1/D14-1167.

[54] D. Wishart, Y. Djoumbou, A. C. Guo, E. Lo, A. Marcu, J. Grant, T. Sajed, D. Johnson, C. Li, Z. Sayeeda, N. Assempour, I. Iynkkaran, Y. Liu, A. Maciejewski, N. Gale, A. Wilson, L. Chin, R. Cummings, D. Le, and M. Wilson. Drugbank 5.0: A major update to the drugbank database for 2018. *Nucleic acids research*, 46, 11 2017. doi:10.1093/nar/gkx1037.

[55] P. Wolff, M. Graña, S. Ríos, and M. Yarza. Machine learning readmission risk modeling: A pediatric case study. *BioMed Research International*, 2019:1–9, 04 2019. doi:10.1155/2019/8532892.

[56] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi:10.1109/TNNLS.2020.2978386.

[57] M. D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-5701.

[58] Z. Zhang and M. R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels, 2018.

A

Appendix Tables

This section contains information on some specific features from MIMIC-III dataset in particularly the one that have been used in the thesis work. Later, one table containing information about all MIMIC-III tables to give the reader an information on what more can be explored from the dataset.

Table A.1: The table provides extended count details of marital status present in the MIMIC-III dataset.

Martial status	**Counts**
MARRIED	14965
SINGLE	7935
WIDOWED	3743
DIVORCED	1882
UNKNOWN/OTHER	1613
SEPARATED	316
LIFE PARTNER	11

Table A.2: Table (a) provides counts of different admission type where as table (b) provides counts on discharge types.

Admission location	**Count**
EMERGENCY ROOM ADMIT	12477
PHYS REFERRAL /NORMAL DELI	6788
CLINIC REFERRAL /PREMATURE	5553
TRANSFER FROM HOSP/EXTRAM	5541
TRANSFER FROM SKILLED NUR	86
TRANSFER FROM OTHER HEALT	14
** INFO NOT AVAILABLE **	3
TRSF WITHIN THIS FACILITY	2
HMO REFERRAL /SICK	1

Discharge location	**Counts**
HOME	10219
HOME HEALTH CARE	8855
SNF	4391
REHAB/DISTINCT PART HOSP	4302
LONG TERM CARE HOSPITAL	1136
DISC-TRAN CANCER /CHLDRN H	519
DISCH-TRAN TO PSYCH HOSP	372
SHORT TERM HOSPITAL	288
LEFT AGAINST MEDICAL ADVI	192
HOSPICE-HOME	54
OTHER FACILITY	44
HOME WITH HOME IV PROVIDR	38
ICF	27
HOSPICE-MEDICAL FACILITY	19
DISC-TRAN TO FEDERAL HC	9

Table A.3: The table provides extended details of different transfer types present in the MIMIC-III dataset.

Original	**Mapped**
TRANSFER FROM HOSP/EXTRAM	TRANSFERRED
TRANSFER FROM SKILLED NUR	TRANSFERRED
TRANSFER FROM OTHER HEALT	TRANSFERRED
** INFO NOT AVAILABLE **	EMERGENCY ROOM ADMIT
TRSF WITHIN THIS FACILITY	TRANSFERRED
HMO REFERRAL /SICK	EMERGENCY ROOM ADMIT
DISC-TRAN CANCER/CHLDRN H	OTHER
DISC-TRAN TO FEDERAL HC	OTHER
DISCH-TRAN TO PSYCH HOSP	OTHER
HOME WITH HOME IV PROVIDR	HOME
HOSPICE-HOME	OTHER
HOSPICE-MEDICAL FACILITY	OTHER
ICF	OTHER
LEFT AGAINST MEDICAL ADVI	OTHER
OTHER FACILITY	OTHER
SHORT TERM HOSPITAL	OTHER
SNF-MEDICAID ONLY CERTIF	SNF

Table A.4: The table provides extended count and details of patient religion present in the MIMIC-III dataset.

Religion	**Count**
CATHOLIC	10859
NOT SPECIFIED	6820
PROTESTANT QUAKER	3,668
UNOBTAINABLE	3466
JEWISH	2488
OTHER	1489
EPISCOPALIAN	416
GREEK ORTHODOX	237
CHRISTIAN SCIENTIST	219
BUDDHIST	122
MUSLIM	100
JEHOVAH'S WITNESS	68
UNITARIAN -UNIVERSALIST	65
HINDU	49
7TH DAY ADVENTIST	39
ROMANIAN EAST. ORTH	38
BAPTIST	14
HEBREW	8
METHODIST	6

Original	**Mapped**
7TH DAY ADVENTIST	PROTESTANT
BAPTIST	PROTESTANT
BUDDHIST	OTHER
CHRISTIAN SCIENTIST	OTHER
EPISCOPALIAN	PROTESTANT
GREEK ORTHODOX	OTHER
HEBREW	JEWISH
HINDU	OTHER
JEHOVAH'S WITNESS	OTHER
MUSLIM	OTHER
METHODIST	PROTESTANT
NOT SPECIFIED	UNKNOWN
PROTESTANT QUAKER	PROTESTANT
ROMANIAN EAST. ORTH	OTHER
UNITARIAN-UNIVERSALIST	OTHER
UNOBTAINABLE	UNKNWON

Table A.5: The table provides extended count details of patient ethnicity present in the MIMIC-III dataset.

Ethnicity	**Count**
WHITE	21570
UNKNOWN/NOT SPECIFIED	2661
BLACK/AFRICAN AMERICAN	2246
HISPANIC OR LATINO	861
OTHER	730
UNABLE TO OBTAIN	530
ASIAN	448
PATIENT DECLINED TO ANSWER	324
ASIAN - CHINESE	158
HISPANIC/LATINO - PUERTO RICAN	126
BLACK/CAPE VERDEAN	118
WHITE - RUSSIAN	81
MULTI RACE ETHNICITY	74
BLACK/HAITIAN	59
WHITE - OTHER EUROPEAN	55
HISPANIC/LATINO - DOMINICAN	53
ASIAN - ASIAN INDIAN	51
WHITE - BRAZILIAN	36
PORTUGUESE	31
ASIAN - VIETNAMESE	29
MIDDLE EASTERN	23
HISPANIC/LATINO - GUATEMALAN	23
BLACK/AFRICAN	22
WHITE - EASTERN EUROPEAN	18
AMERICAN INDIAN/ALASKA NATIVE	17
HISPANIC/LATINO - CUBAN	14
HISPANIC/LATINO - SALVADORAN	13
ASIAN - FILIPINO	12
ASIAN - OTHER	9
HISPANIC/LATINO - MEXICAN	9
ASIAN - KOREAN	9
HISPANIC/LATINO - COLOMBIAN	8
NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER	8
CARIBBEAN ISLAND	7
SOUTH AMERICAN	7
HISPANIC/LATINO - CENTRAL AMERICAN (OTHER)	7
ASIAN - CAMBODIAN	6
ASIAN - JAPANESE	4
HISPANIC/LATINO - HONDURAN	3
ASIAN - THAI	3
AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE	2

Table A.6: The table provides extended details of ethnicity present in the MIMIC-III dataset.

Original	**Mapped**
PATIENT DECLINED TO ANSWER	UNKNOWN/NOT SPECIFIED
HISPANIC OR LATINO	HISPANIC
HISPANIC/LATINO - GUATEMALAN	HISPANIC
ASIAN - CHINESE	ASIAN
HISPANIC/LATINO - PUERTO RICAN	HISPANIC
ASIAN - ASIAN INDIAN	ASIAN
ASIAN - VIETNAMESE	ASIAN
MULTI RACE ETHNICITY	OTHER
HISPANIC/LATINO - DOMINICAN	HISPANIC
AMERICAN INDIAN/ALASKA NATIVE	OTHER
WHITE - RUSSIAN	WHITE
BLACK/AFRICAN	BLACK/AFRICAN AMERICAN
HISPANIC/LATINO - SALVADORAN	HISPANIC
UNABLE TO OBTAIN	UNKNOWN/NOT SPECIFIED
BLACK/CAPE VERDEAN	BLACK/AFRICAN AMERICAN
BLACK/HAITIAN	BLACK/AFRICAN AMERICAN
WHITE - OTHER EUROPEAN	WHITE
ASIAN - CAMBODIAN	ASIAN
WHITE - EASTERN EUROPEA	WHITE
ASIAN - FILIPINO	ASIAN
PORTUGUESE	OTHER
CARIBBEAN ISLAND	OTHER
SOUTH AMERICAN	OTHER
ASIAN - KOREAN	ASIAN
HISPANIC/LATINO - COLOMBIAN	HISPANIC
WHITE - BRAZILIAN	WHITE
HISPANIC/LATINO - CENTRAL AMERICAN (OTHER)	HISPANIC
ASIAN - THAI	ASIAN
HISPANIC/LATINO - HONDURAN	HISPANIC
NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER	OTHER
HISPANIC/LATINO - CUBAN	HISPANIC
MIDDLE EASTER	OTHER
ASIAN - OTHER	ASIAN
HISPANIC/LATINO - MEXICAN	HISPANIC
AMERICAN INDIAN/ALASKA NATIVE FEDERALLY RECOGNIZED TRIBE	OTHER
ASIAN - JAPANESE	ASIAN

Table A.7: A complete overview of the MIMIC-III dataset

Table name	**Description**
ADMISSIONS	Define a patient's hospital admission, defines HADM_ID.
CALLOUT	Provides information when a patient was READY for discharge from the ICU, and when the patient was actually discharged from the ICU.
CAREGIVERS	Defines the role of caregivers.
CHARTEVENTS	Contains all charted data for all patients.
CPTEVENTS	Contains current procedural terminology (CPT) codes, which facilitate billing for procedures performed on patients.
D_CPT	High-level definitions for current procedural terminology (CPT) codes.
D_ICD_DIAGNOSES	Definition table for ICD diagnoses.
D_ICD_PROCEDURES	Definition table for ICD procedures.
D_ITEMS	Definition table for all items in the ICU databases.
D_LABITEMS	Definition table for all laboratory measurements.
DATETIMEVENTS	Contains all date formatted data.
DIAGNOSES_ICD	Contains ICD diagnoses for patients, most notably ICD-9 diagnoses.
DRGCODES	Contains diagnosis related groups (DRG) codes for patients.
ICUSTAYS	Defines each ICUSTAY_ID in the database, i.e. defines a single ICU stay.
INPUTEVENTS_CV	CareVue ICU databases. Input data for patients.
INPUTEVENTS_MV	Metavision ICU databases. Input data for patients.
LABEVENTS	Contains all laboratory measurements for a given patient, including out patient data.
MICROBIOLOGYEVENTS	Contains microbiology information, including cultures acquired and associated sensitivities.
NOTEEVENTS	Contains all notes for patients.
OUTPUTEVENTS	Output data for patients.
PATIENTS	Defines each SUBJECT_ID in the database, i.e. defines a single patient.
PRESCRIPTIONS	Contains medication related order entries, i.e. prescriptions.
PROCEDUREEVENTS_MV	Contains procedures for patients.
PROCEDURES_ICD	Contains ICD procedures for patients, most notably ICD-9 procedures.
SERVICES	Lists services that a patient was admitted/transferred under.
TRANSFERS	Physical locations for patients throughout their hospital stay.

B

Appendix Theory

B.1 Deep learning

Deep learning has became an invaluable part of machine learning. The most common ones rely on methods based on Artificial Neural Network (ANN). The neural networks aims to mimic the networks of neurons in the mammalian brain, thereby, mimicking the mode of communication occuring in these biological systems [39]. In case of mammalian brain, the neurons are connected to synapses which transmits signals to immediate neighbouring neurons. This motivates the construction ANN, where a set of neurons are densely connected to another set of neurons. Depending on the activation function i.e. the work done on the signal allows these neural networks to capture complex and hidden structures also known as patterns. Generally, a training dataset is used by the model to learn these patterns and are later tested against an unseen test dataset. The structural configurations of these neurons leads to different nomenclature for neural networks. For example, a feed forward neural network will have fully connected network, whereas a recurrent network will contain connections within the same layer along with forward connections.

Each synapse (neuron-to-neuron) connection has a corresponding weight and bias associated to it, where the weights control the strength of the signal processed to the sub-sequential layer, i.e. controls how much influence an neuron will have to the next connections of neurons. The biases works as offsets and are not directly connected to the synapses, but rather have their own connections to each neuron in each layer, and are therefore not influenced by the previous layers. Each neuron connected to a set of neurons in a previous layer has an associated activation function, which defines an output for the set of input signals to that specific neuron. Figure B.1 illustrates the basic concept of weights and biases. Conceptually, a neurons value is determined by the set of connected neurons values from a previous layer and their corresponding weights, and the bias connected to the neuron of interest. The input is calculated as a weighted sum of its connected neurons and corresponding weights, with the addition of bias value also known as Hebb's rule. This value is then fed through an activation function which yields another value as the output. A common choice of activation function is the Rectified Linear Unit (ReLU), which is an non-linear function described as:

$$\mathrm{ReLU}(x) = [x]_+ = \max(0, x) \tag{B.1}$$

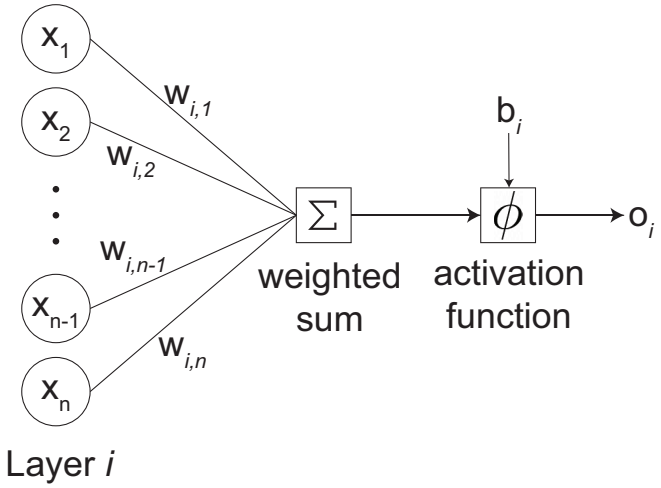


Figure B.1: Graphic example of a calculated output from an neuron in layer i with n input neurons.

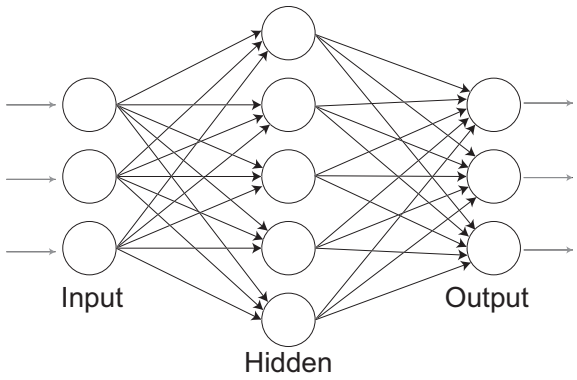


Figure B.2: Graphic example of a feed-forward artificial neural network with three input neurons, one hidden layer with 5 neurons, and an output layer with three neurons.

The equation B.1 for ReLU function yields an output signal only if the input signal is positive, otherwise it returns 0. This leads to one of the benefits of ReLU activation function i.e. it naturally implements a dropout-schema for neurons which is usually used to prevent overfitting. Another benefit is its efficient computations and it results in fewer vanishing gradient problems that other commonly used activation-functions [23]. Neural networks can easily fall into the trap of overfitting on training data. One way to avoid some overfitting is by having dropout. A dropout excludes or ignore units randomly with a probability and their corresponding connections [50].

Feed-Forward Neural Network

A feed-forward neural network is a type of artificial neural networks where the connections between neurons do not form a circle or loop, and the transmitted information is fed forward in its architecture. The signal is transmitted from neurons of

one entire layer to the neurons of its subsequent layer. No information is exchanged within the neurons of same layer. The information moves in one direction - starting in the input layer, moving on to the hidden layers and ending up in the output layer giving rise to the name feed forward. A neural network is called as deep network when it's architecture contains one or more hidden layers. The weights connecting these layers are initialized using Xavier initialization. An exemplary feed-forward artificial neural network is presented in figure B.2, showing one input layer, one hidden layers and one output layer.

A deep learning problem statement

The problem that the deep learning is trying to solve can be stated as finding an approximate function $\mathcal{F}$ that satisfies equation B.2. The candidates that approximately satisfy the equation are called as hypothesis space. In general, the hypothesis space should map input points $\mathbf{x}$ to target points $\mathbf{y}$ for each $(\mathbf{x}, \mathbf{y}) \in (\mathcal{X}, \mathcal{Y})$ respectively in a corresponding fashion.

$$\mathcal{F} : \mathbf{x} \rightarrow \mathbf{y}, \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y} \tag{B.2}$$

The parameterized function that satisfy the hypothesis space can be termed as $\mathcal{F}_\theta$, where θ is the set of parameters that makes the function satisfy the hypothesis. In neural networks, the θ is generally weights and biases. Hence, neural networks aims to obtain the parameters θ that best approximate $\mathcal{F}_\theta(\mathbf{x}) \approx \mathcal{F}(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$. Generalising the problem, one can state that there is a joint probability distribution $P(\mathbf{x}, \mathbf{y})$ over $\mathcal{X}$ and $\mathcal{Y}$. The training set contain Independent and Identically Distributed (i.i.d.) samples from this joint distribution. The network learns by constantly predicting the response vectors and using the information of how deviated is the predicted response from the true response. Therefore, for each given pair in the training set, the function maps the input point to a predicted target point, i.e. $\mathcal{F}_\theta(\mathbf{x}) = \hat{\mathbf{y}}$. A associated non-negative real-valued loss function measures the deviance between the predicted target vector of the function $\mathcal{F}_\theta$ and the true target vector, i.e. $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, where $\hat{\mathbf{y}}$ denotes the predicted target vector. The risk that is then associated with the function $\mathcal{F}$ is expressed as shown in equation B.3.

$$R(\mathcal{F}) := \mathbb{E}\left[\mathcal{L}(\,\mathcal{F}(\mathbf{x}), \mathbf{y}\,)\right] = \int_{(\mathbf{x},\mathbf{y})\in(\mathcal{X},\mathcal{Y})} \mathcal{L}\left(\mathcal{F}(\mathbf{x}), \mathbf{y}\right) dP(\mathbf{x}, \mathbf{y}) \tag{B.3}$$

The risk is just the expectation of the loss-function. Ultimately, we wish to find the optimal function $\mathcal{F}^*$ from the hypothesis space which minimizes the risk:

$$\mathcal{F}^* = \arg\min_{\mathcal{F}} R(\mathcal{F}(\mathbf{x})) \tag{B.4}$$

Since this joint probability distribution is not known, it's not generally possible to compute the risk. However, an approximation called empirical risk can be computed by averaging the loss over the training sample pairs, and thus selecting the function which minimizes the empirical loss over the training sample pairs as shown in equation B.5. The choice of loss function is generally problem-dependent.

$$\hat{\mathcal{F}} = \arg\min_{\mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left(\mathcal{L}(\mathcal{F}(\mathbf{x}_i), \mathbf{y}_i)\right) \tag{B.5}$$

For an ANN, the principle is that the network architecture $\hat{\mathcal{F}}$ is initiated with many free parameters (weights and biases) such that it can find a suitable imitation of the unknown function $\mathcal{F}$. For each training pair $(\mathbf{x}_i, \mathbf{y}_i)$, we can make a prediction of the target vector based on the input vector $\hat{\mathbf{y}}_i = \hat{\mathcal{F}}(\mathbf{x}_i)$ and calculate the loss for that prediction using a loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ which measures the difference between the target vector and the predicted target vector. The parameters of the network architecture can then be updated with respect to the loss captured by the loss function, so that the prediction loss is lowered in the following iteration:

$$\hat{\mathcal{F}}_{t+1} \leftarrow \hat{\mathcal{F}}_t + B\left(\hat{\mathcal{F}}_t, \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})\right) \tag{B.6}$$

B denotes a back propagation which updates the parameters of the network architecture so that it lowers the proceeding prediction error measured by the loss function. This is generally achieved by gradient descent algorithms like stochastic gradient descent, refer section B.1.2.

B.1.1 Supervised learning

This section builds on the details mentioned at the end of the deep learning section B.1. Infact, the problem described in section B.1 is a typical supervised learning problem i.e. when the response variables are known.

Supervised learning is one type of machine learning problems which learns a function $\hat{\mathcal{F}}$ that maps an input points $\mathbf{x}$ to a target point $\mathbf{y}$ based on input-target-pairs $\{\mathbf{x}, \mathbf{y}\}$. Let, a set of training input-target-pairs of the form $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ is given. Here, $\mathbf{x}_i$ represents the i-th observation's feature vector and $\mathbf{y}_i$ represents its corresponding target vector. The learning algorithm seeks to find the mapping function that maps input feature vectors belonging to an input domain $\mathcal{X}$ to a target domain $\mathcal{Y}$.

One important goal of machine learning algorithms is to generalize the training. In the context of supervised learning problem, one could refer to this as bias-variance trade-off problem i.e. if the model fit too much on the training data, it has high variance and it is overfitting. Contrary to this, if the model fit too little to the training data then it has high bias and is underfitted. Figure B.3 illustrates model fits. Therefore, mandatory train, validation and test splits are required to control the amount of fitting is done on the training data.

B.1.2 Back propagation using Stochastic Gradient Descent

Back propagation is a method frequently used to train, i.e. update the parameters of a feed forward artificial neural network with respect to a loss function. The idea is to adjust the parameters of the network architecture in order to correctly classify each training-pair, or to minimize the error between the target vector and the predicted target vector. The loss function is parameterized with smoothness

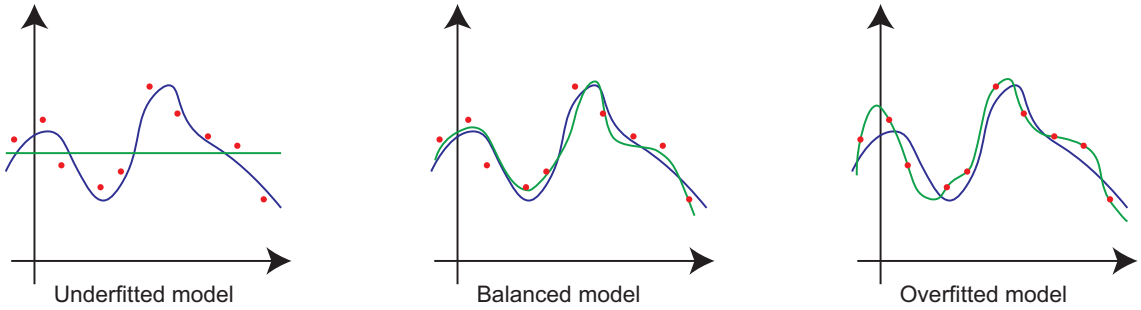


Figure B.3: Graphic example of a underfitted, balanced and overfitted model. Red dots are data points with noise, blue curve is true function and green curve is fitted model.

properties (differentiable or sub-differentiable), meaning that the gradient can be calculated with respect to the network parameters. The stochastic nature arises in the calculation of the gradient of the loss - it's an approximation of the actual gradient which is calculated on the whole training set with a subset of it called mini-batch. This has two benefits - it reduces the complexity burden and achieves faster training (but with trade-off with slower convergence) and also has the possibility of escaping local minimas. This is since the weights and biases are adapted to take a step in the steepest descent direction of the gradient computed with respect to the loss. The sampling of the mini-batches gives the ability to move in an (globally) non-optimal path, meaning not moving in the direct steepest descending direction, and avoid getting stuck in local optimums. The updated network parameters using steepest gradient descent algorithm can be described in the iterative scheme (we denote the parameters of the network architecture as θ for simplicity):

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla \mathcal{L}(\theta_t | (\mathbf{x}, \mathbf{y})) \tag{B.7}$$

where η is a tunable parameter called *learning rate*, which determines how much we move in the direction of the steepest descent, and the loss is calculated based on a training-pair $(\mathbf{x}, \mathbf{y})$, $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$. This methods calculates the gradient for each training-pair, which quickly scales in time complexity if the number of samples is large, and also if the dimensionality of θ is large. Furthermore, it is greedy in the sense that it always follows the most optimal descent in a local fashion, making it prone to land in local minimums. We wish to approximate equation B.7 with the following:

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\theta_t | (\mathbf{x}_i, \mathbf{y}_i)) \tag{B.8}$$

where $\{(\mathbf{x}_1, \mathbf{y}_i), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ is a random subset of training-pairs of size N. Since the mini-batches are differently sampled each time and is therefore a random variable, it introduces the possibility to move in non-optimal paths, leaving local minimums and explore the search space better in order to find the global minimum.

However, when dealing with imbalanced training pairs, i.e. that we have a discrepancy between the sizes of the different class sizes, each batch needs to be weighted

so that the expectation of each class occurring is roughly the same. By this, the network learns to identify patterns in different classes, and do not neglect the minority class. This introduces the possibility for the learning algorithm to learn from all classes, despite their imbalance in size.

B.1.3 Adam for stochastic optimization

In the previous sub-section, the stochastic gradient descent method using mini-batches was presented. However, it uses a fixed learning rate which is a difficult hyper-parameter to tune, and doesn't change during the training phase. Different methods have been proposed that deals with dynamic learning rates, which can make to stochastic optimization more effective. These include ADAGRAD, which adapts the learning rate to the parameters based on frequently occurring features [20], and ADADELTA [57], which is an extension of the formerly mentioned adaptive stochastic optimizer. Adam, short for Adaptive Moment Estimation [32] is another stochastic optimizer that computes adaptive learning rates for the parameters, using the first-order gradients and require little memory overhead. It computes adaptive learning rates for each individual parameter from estimates of the first and second moments of the gradients. By calculating the averages of the gradient m_t and squared gradient v_t, which acts as the first and second moment of the gradients with respect to the objective function $\mathcal{L}$, it updates the exponential moving averages of m_t and v_t with respect to time as following:

$$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1)\nabla\mathcal{L}(\theta_{t-1}) \tag{B.9}$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\nabla(\mathcal{L}(\theta_{t-1}))^2 \tag{B.10}$$

These moving averages are initialized to the zero vector, and is thus biased towards 0 in initial time-steps and for hyper-parameter values β_1 and β_2 close to unity. Therefore, the bias corrected estimates of the moments are calculated as:

$$\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t} \tag{B.11}$$

$$\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t} \tag{B.12}$$

The time variable t is also updated since the hyper-parameters are time-dependent with respect to the power function, changing for each time-step. This is then used for updating the parameters θ as:

$$\theta_{t+1} \leftarrow \theta_t - \eta\frac{1}{\sqrt{\hat{v}_t}+\varepsilon}\hat{m}_t \tag{B.13}$$

There is now 4 hyper-parameters present - η which acts as a time-step, β_1 and β_2 which acts as the relative exponential decay rate for the first and second moments of the gradients with respect to the objective function. The hyper-parameter ϵ is set to a small number to avoid division-by-zero.

B.1.4 Loss functions

The loss function plays an important role in the setting of deep learning and needs to be chosen wisely for the particular task. Its main purpose is to give an indication of measurement for how well the training is performed on the learning algorithm, as well as updating the learning algorithm in the direction of the steepest descending gradient so that the performance is increased. The loss function, also frequently called *energy function*, gives us a measure on how well the predicted target vector is mimicking the actual target vector.

Cross entropy loss [58], or log-loss, is used to measure a models prediction where the output is presented as a vector of probabilities between 0 and 1 of belonging to a set of different classes, and the true class that sample belongs to. Since the network's outputs are scalars that don't necessarily represent a probability distribution of a sample $\mathbf{x}$ belonging to one of C classes, it needs to be transformed so that the values in the output layer are non-negative and sum to unity. This is done via the Softmax-function, which is described as following:

$$\text{Softmax}(\mathbf{o})_i = \frac{\exp(o_i)}{\sum_{j \in C} \exp(o_j)} \tag{B.14}$$

This function maps each scalar in the vector $\mathbf{o}$, which is referring to the output vector from a neural network, to a scalar in the range $(0, 1)$ where the sum of all elements in the vector sums to 1. This can then directly be considered a probability distribution of belonging to a class $\{0, \ldots, |C| - 1\}$. The cross entropy-loss is then calculated as following:

$$\mathcal{L}_{CE}(\mathbf{o}, \mathbf{y}) = -\sum_{j \in C} y_j \log(\text{Softmax}(\mathbf{o})_j) \tag{B.15}$$

Here, $\mathbf{y}$ represents a one-hot encoding of the targets, i.e. a vector of size equal to the number of classes, with value 1 at the corresponding index of the target class and 0 elsewhere. In the setting where a learning algorithm needs to classify target-pairs in an imbalanced distribution, i.e. the distribution of class occurrences is skewed and some classes occurs more frequently than others, a weighted variant of the cross entropy loss-function is applied:

$$\mathcal{L}_{wCE}(\mathbf{o}, \mathbf{y}) = -\sum_{j \in C} w_j y_j \log(\text{Softmax}(\mathbf{o})_j) \tag{B.16}$$

The weight vector $\mathbf{w}$ denotes the weights of each class. This weight is set to be the inverse class frequency, normalized so that the sum of the weight-vector is unity, so classes with few observations will have a higher weights than classes with many observations. This penalizes predictions where the target is of a minority class more than the majority class, so that the loss function accounts for class imbalances.

C

Appendix Results

This section contains the detailed hyperparameter results for the models trained.

Table C.1: The table states the results obtained on different hyperparameters for simple-net on one-hot data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	31.10 (0.5)	33.86 (0.9)	30.45 (0.6)
128/0.001	30.92 (0.5)	32.78 (0.9)	30.25 (0.5)
256/0.001	30.63 (0.7)	31.99 (1.2)	29.98 (0.8)
64/0.005	31.19 (0.7)	33.86 (1.4)	31.16 (0.9)
128/0.005	31.04 (0.8)	32.95 (1.7)	30.87 (1.1)
256/0.005	30.41 (0.9)	31.44 (1.4)	30.21 (1.0)
64/0.01	30.54 (0.6)	33.53 (1.2)	30.15 (1.1)
128/0.01	30.52 (0.7)	33.38 (1.9)	30.29 (1.2)
256/0.01	30.66 (0.9)	32.89 (1.5)	30.49 (1.1)

Table C.2: The table states the results obtained on different hyperparameters for deep-net on one-hot data.

Parameters HL1/HL2/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	31.10 (0.5)	33.67 (1.2)	31.02 (0.5)
128/64/0.001	30.85 (0.5)	33.09 (1.3)	30.83 (0.7)
256/64/0.001	30.41 (0.5)	32.03 (1.3)	30.40 (0.7)
64/128/0.001	31.03 (0.8)	33.58 (1.3)	30.92 (0.9)
128/128/0.001	30.45 (0.7)	32.60 (1.7)	30.62 (0.8)
256/128/0.001	30.33 (0.6)	32.48 (1.9)	30.55 (0.8)
64/256/0.001	30.63 (0.4)	32.85 (1.5)	30.53 (0.5)
128/256/0.001	30.92 (0.5)	33.37 (1.6)	31.18 (0.6)
256/256/0.001	30.30 (0.7)	31.24 (1.5)	30.15 (0.8)
64/64/0.005	30.80 (1.1)	33.57 (1.6)	30.72 (1.2)
128/64/0.005	30.62 (0.7)	33.49 (1.2)	30.76 (1.1)
256/64/0.005	30.53 (0.6)	32.54 (1.5)	30.73 (0.7)
64/128/0.005	30.95 (0.7)	33.65 (0.8)	30.49 (0.8)
128/128/0.005	30.97 (1.1)	33.32 (1.4)	31.30 (1.2)
256/128/0.005	30.10 (0.8)	32.51 (1.5)	30.26 (1.0)
64/256/0.005	31.33 (1.2)	33.96 (0.7)	30.90 (0.9)
128/256/0.005	30.52 (1.1)	32.79 (1.3)	30.58 (1.4)
256/256/0.005	30.43 (0.9)	32.43 (1.4)	30.43 (0.9)
64/64/0.01	30.69 (0.6)	34.24 (1.2)	29.68 (0.9)
128/64/0.01	30.87 (1.2)	32.35 (1.2)	30.38 (1.5)
256/64/0.01	31.32 (1.1)	33.82 (1.7)	31.25 (0.9)
64/128/0.01	31.26 (1.9)	33.67 (0.9)	29.25 (1.4)
128/128/0.01	30.43 (0.5)	32.76 (1.5)	30.45 (0.6)
256/128/0.01	30.64 (1.1)	32.32 (1.3)	30.37 (0.8)
64/256/0.01	29.75 (1.3)	33.74 (1.7)	28.68 (1.7)
128/256/0.01	31.25 (0.9)	33.89 (1.0)	29.80 (1.6)
256/256/0.01	30.00 (0.7)	31.96 (1.3)	29.61 (0.9)

Table C.3: The table states the results obtained on different hyperparameters for simple-net on numeric data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	30.62 (0.5)	36.12 (0.8)	29.59 (0.7)
128/0.001	30.78 (0.5)	36.01 (0.8)	30.00 (0.8)
256/0.001	30.28 (0.5)	34.37 (0.9)	30.36 (0.6)
64/0.005	30.29 (0.8)	34.87 (0.6)	29.43 (1.0)
128/0.005	30.64 (0.9)	34.99 (1.1)	29.53 (0.9)
256/0.005	30.29 (0.9)	34.48 (1.6)	29.66 (1.3)
64/0.01	29.47 (0.8)	34.19 (1.1)	27.80 (0.8)
128/0.01	29.54 (0.8)	33.89 (1.2)	27.63 (1.0)
256/0.01	30.12 (1.2)	33.49 (1.5)	27.42 (1.1)

Table C.4: The table states the results obtained on different hyperparameters for deep-net on numeric data.

Parameters HL1/HL2/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	30.59 (0.5)	35.51 (0.7)	29.56 (0.7)
128/64/0.001	30.36 (0.5)	35.40 (1.1)	29.89 (0.7)
256/64/0.001	30.76 (0.6)	35.36 (0.9)	30.79 (0.8)
64/128/0.001	30.77 (0.6)	35.93 (1.1)	29.82 (0.6)
128/128/0.001	30.12 (0.2)	34.70 (0.6)	29.58 (0.3)
256/128/0.001	30.19 (0.6)	33.97 (1.2)	30.34 (0.7)
64/256/0.001	30.89 (1.3)	35.51 (1.2)	29.72 (1.1)
128/256/0.001	30.09 (0.5)	34.52 (0.9)	29.76 (0.6)
256/256/0.001	30.36 (0.3)	34.58 (1.1)	30.59 (0.3)
64/64/0.005	30.82 (2.4)	34.78 (0.5)	28.96 (0.7)
128/64/0.005	30.29 (1.0)	34.59 (1.3)	29.04 (1.6)
256/64/0.005	29.93 (0.6)	34.60 (0.7)	28.91 (1.1)
64/128/0.005	29.94 (1.1)	34.47 (0.9)	28.84 (0.9)
128/128/0.005	30.08 (0.9)	34.61 (1.0)	28.68 (1.0)
256/128/0.005	29.92 (0.9)	33.95 (1.3)	28.98 (0.8)
64/256/0.005	29.42 (1.1)	33.75 (1.0)	28.31 (0.8)
128/256/0.005	29.65 (0.9)	33.87 (0.9)	28.23 (0.7)
256/256/0.005	29.75 (0.7)	34.27 (1.0)	28.70 (1.0)
64/64/0.01	29.97 (0.9)	33.24 (0.8)	28.21 (0.6)
128/64/0.01	29.83 (1.1)	33.99 (1.3)	27.74 (0.9)
256/64/0.01	29.49 (1.9)	34.16 (1.7)	28.06 (2.2)
64/128/0.01	28.33 (2.1)	32.30 (1.5)	26.53 (1.2)
128/128/0.01	30.71 (0.7)	35.80 (0.8)	30.21 (0.7)
256/128/0.01	30.51 (1.0)	34.12 (1.3)	28.36 (1.3)
64/256/0.01	27.13 (1.6)	31.62 (1.9)	25.83 (0.9)
128/256/0.01	29.27 (1.1)	32.87 (1.3)	27.30 (1.4)
256/256/0.01	29.89 (1.4)	32.93 (0.9)	27.72 (1.0)

Table C.5: The table states the results obtained on different hyperparameters for simple-net on link prediction TransE-L1 embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	26.45 (0.2)	28.77 (0.8)	21.84 (1.0)
128/0.001	26.33 (0.2)	28.55 (0.9)	21.67 (0.5)
256/0.001	26.45 (0.2)	28.89 (0.5)	23.09 (0.2)
64/0.005	26.41 (0.9)	27.68 (1.0)	23.30 (1.9)
128/0.005	26.04 (0.3)	27.54 (0.7)	22.87 (0.7)
256/0.005	26.16 (0.3)	27.68 (0.8)	23.96 (0.6)
64/0.01	26.73 (0.8)	28.02 (0.9)	24.35 (1.6)
128/0.01	26.13 (0.5)	27.38 (1.2)	23.02 (1.0)
256/0.01	26.17 (0.3)	27.45 (0.8)	23.76 (0.7)

Table C.6: The table states the results obtained on different hyperparameters for deep-net on link prediction TransE-L1 embedded data.

Parameters HL1/HL2/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	26.68 (0.5)	28.53 (1.1)	24.64 (2.4)
128/64/0.001	26.29 (0.4)	27.99 (0.7)	23.04 (1.6)
256/64/0.001	26.32 (0.5)	28.26 (1.4)	23.05 (0.8)
64/128/0.001	26.59 (0.6)	28.06 (0.7)	24.32 (1.8)
128/128/0.001	26.43 (0.8)	28.04 (0.7)	23.12 (1.8)
256/128/0.001	26.47 (0.4)	28.27 (1.1)	23.75 (0.9)
64/256/0.001	26.36 (0.4)	27.95 (1.1)	23.16 (0.9)
128/256/0.001	26.17 (0.3)	27.77 (0.8)	22.92 (0.7)
256/256/0.001	26.28 (0.3)	28.00 (0.9)	23.78 (0.8)
64/64/0.005	27.01 (1.3)	28.15 (1.5)	24.99 (2.7)
128/64/0.005	27.26 (1.3)	28.66 (0.9)	25.70 (1.6)
256/64/0.005	26.84 (0.7)	28.10 (1.0)	25.33 (1.7)
64/128/0.005	27.03 (1.1)	28.51 (0.8)	26.55 (1.1)
128/128/0.005	26.31 (0.8)	27.85 (0.9)	24.41 (1.7)
256/128/0.005	26.19 (0.4)	27.48 (0.7)	24.10 (1.3)
64/256/0.005	27.55 (1.1)	28.66 (0.7)	25.79 (1.2)
128/256/0.005	26.52 (0.9)	27.60 (1.5)	24.10 (1.7)
256/256/0.005	26.76 (1.1)	27.71 (1.0)	25.18 (1.3)
64/64/0.01	27.76 (2.2)	27.43 (0.5)	26.43 (0.6)
128/64/0.01	25.83 (1.5)	27.68 (1.5)	24.97 (1.8)
256/64/0.01	26.07 (1.2)	27.17 (1.2)	22.52 (3.3)
64/128/0.01	27.27 (1.3)	28.06 (0.8)	26.05 (0.7)
128/128/0.01	26.54 (0.3)	28.42 (0.8)	23.80 (1.0)
256/128/0.01	25.77 (1.5)	27.11 (1.4)	23.46 (1.8)
64/256/0.01	27.08 (1.7)	27.79 (1.2)	26.02 (0.5)
128/256/0.01	26.39 (1.0)	27.14 (1.1)	25.27 (1.3)
256/256/0.01	26.16 (0.6)	26.49 (0.8)	24.25 (1.5)

Table C.7: The table states the results obtained on different hyperparameters for simple-net on link prediction TransE-L2 embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	28.39 (0.7)	30.98 (1.1)	26.74 (0.7)
128/0.001	28.05 (0.2)	30.84 (0.8)	25.49 (0.4)
256/0.001	27.93 (0.2)	30.51 (0.6)	25.87 (0.3)
64/0.005	28.71 (0.2)	32.24 (0.5)	26.91 (0.7)
128/0.005	28.29 (0.4)	30.99 (0.9)	26.50 (0.5)
256/0.005	28.17 (0.6)	30.81 (1.1)	26.79 (0.5)
64/0.01	29.04 (0.4)	31.42 (0.4)	27.39 (0.6)
128/0.01	28.71 (0.6)	31.47 (0.9)	26.81 (0.7)
256/0.01	28.55 (0.6)	31.25 (0.9)	26.80 (0.8)

Table C.8: The table states the results obtained on different hyperparameters for deep-net on link prediction TransE-L2 embedded data.

Parameters HL1/HL2/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	28.65 (0.3)	31.64 (0.8)	27.30 (0.8)
128/64/0.001	28.33 (0.4)	30.88 (0.9)	26.67 (0.7)
256/64/0.001	28.18 (0.4)	30.64 (0.8)	26.62 (0.6)
64/128/0.001	28.91 (0.4)	32.09 (1.0)	27.61 (0.8)
128/128/0.001	28.58 (0.4)	31.35 (1.1)	26.91 (0.6)
256/128/0.001	28.42 (0.4)	31.03 (1.0)	26.83 (0.5)
64/256/0.001	28.56 (0.5)	31.21 (1.1)	27.10 (0.8)
128/256/0.001	28.71 (0.5)	31.17 (0.9)	27.00 (0.7)
256/256/0.001	28.64 (0.4)	31.21 (0.6)	27.08 (0.5)
64/64/0.005	29.23 (1.2)	31.45 (0.9)	27.16 (0.9)
128/64/0.005	29.50 (0.8)	31.36 (0.8)	27.28 (1.0)
256/64/0.005	29.11 (1.0)	30.92 (1.0)	26.95 (0.8)
64/128/0.005	29.29 (1.3)	31.31 (0.6)	27.14 (0.6)
128/128/0.005	29.01 (0.7)	31.44 (1.1)	27.15 (1.1)
256/128/0.005	28.86 (0.5)	31.05 (0.9)	26.91 (0.9)
64/256/0.005	28.88 (1.1)	30.78 (0.8)	26.54 (1.0)
128/256/0.005	28.78 (0.7)	31.28 (1.1)	26.83 (1.3)
256/256/0.005	29.68 (1.8)	31.81 (1.0)	27.59 (0.8)
64/64/0.01	28.38 (0.9)	30.25 (1.2)	26.13 (1.2)
128/64/0.01	28.33 (1.2)	30.43 (0.6)	26.16 (1.2)
256/64/0.01	28.54 (1.6)	30.59 (1.0)	26.62 (1.2)
64/128/0.01	28.35 (0.8)	30.47 (0.9)	26.60 (1.3)
128/128/0.01	28.58 (0.4)	31.30 (0.7)	27.11 (0.6)
256/128/0.01	28.55 (0.8)	31.01 (0.9)	27.21 (1.2)
64/256/0.01	27.95 (0.7)	30.37 (1.0)	25.95 (0.9)
128/256/0.01	27.86 (0.5)	29.85 (0.6)	26.28 (1.1)
256/256/0.01	28.33 (1.1)	30.36 (1.3)	26.78 (0.9)

Table C.9: The table states the results obtained on different hyperparameters for simple-net on link prediction TransH embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	27.61 (0.2)	30.10 (0.9)	25.47 (1.1)
128/0.001	27.76 (0.4)	31.07 (1.5)	24.78 (0.7)
256/0.001	27.29 (0.4)	29.81 (1.2)	24.71 (0.7)
64/0.005	27.93 (0.6)	29.93 (1.1)	26.26 (0.6)
128/0.005	28.03 (0.8)	30.47 (1.3)	25.89 (0.8)
256/0.005	27.51 (0.3)	29.90 (1.0)	25.95 (0.4)
64/0.01	27.67 (0.6)	29.78 (0.9)	25.73 (0.6)
128/0.01	27.85 (0.4)	30.81 (1.0)	25.76 (0.7)
256/0.01	27.37 (0.4)	29.67 (0.9)	25.53 (0.4)

Table C.10: The table states the results obtained on different hyperparameters for deep-net on link prediction TransH embedded data.

Parameters HL1/HL2/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	27.73 (0.8)	29.58 (1.2)	26.27 (1.1)
128/64/0.001	27.88 (0.3)	30.60 (0.8)	26.30 (0.7)
256/64/0.001	27.85 (0.4)	30.86 (0.8)	26.13 (0.6)
64/128/0.001	27.61 (0.4)	29.75 (1.1)	26.30 (0.5)
128/128/0.001	27.74 (0.3)	30.54 (0.8)	25.71 (0.7)
256/128/0.001	28.01 (0.5)	31.01 (0.9)	26.22 (0.7)
64/256/0.001	27.74 (0.4)	30.22 (1.0)	26.26 (0.5)
128/256/0.001	28.03 (0.3)	30.89 (1.1)	26.55 (0.6)
256/256/0.001	27.98 (0.4)	31.14 (1.1)	26.29 (0.7)
64/64/0.005	27.85 (0.5)	30.12 (1.2)	26.78 (0.6)
128/64/0.005	27.51 (0.6)	29.81 (1.1)	26.14 (1.0)
256/64/0.005	28.03 (0.5)	30.51 (1.3)	26.43 (0.7)
64/128/0.005	27.91 (0.6)	29.74 (0.9)	26.69 (1.0)
128/128/0.005	27.76 (0.3)	30.35 (0.8)	26.11 (0.6)
256/128/0.005	28.33 (0.4)	31.28 (0.9)	26.86 (1.0)
64/256/0.005	27.78 (0.4)	30.34 (1.0)	26.31 (0.6)
128/256/0.005	27.69 (0.6)	30.08 (1.3)	25.71 (1.1)
256/256/0.005	27.91 (0.6)	30.61 (1.2)	26.12 (0.8)
64/64/0.01	27.73 (0.8)	29.88 (1.4)	26.18 (1.7)
128/64/0.01	27.82 (0.6)	30.07 (1.3)	26.13 (1.1)
256/64/0.01	27.98 (0.3)	30.61 (0.6)	26.43 (0.8)
64/128/0.01	27.65 (0.4)	30.00 (0.8)	25.91 (1.1)
128/128/0.01	27.98 (0.3)	30.89 (0.91)	26.28 (0.6)
256/128/0.01	27.15 (0.4)	28.75 (0.9)	25.28 (1.0)
64/256/0.01	27.58 (0.6)	29.67 (0.8)	25.89 (0.7)
128/256/0.01	27.59 (1.3)	29.41 (1.3)	25.27 (1.1)
256/256/0.01	27.94 (0.8)	30.24 (1.1)	26.01 (1.0)

Table C.11: The table states the results obtained on different hyperparameters for simple-net on link prediction TransR embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	27.06 (0.3)	29.34 (1.2)	22.99 (0.9)
128/0.001	26.16 (0.3)	26.76 (0.8)	21.63 (0.6)
256/0.001	25.92 (0.3)	26.04 (1.0)	22.31 (0.6)
64/0.005	26.43 (0.5)	27.61 (1.3)	24.00 (1.1)
128/0.005	26.07 (0.5)	26.74 (1.3)	23.32 (0.8)
256/0.005	25.85 (0.3)	26.25 (0.8)	23.68 (0.5)
64/0.01	26.38 (0.3)	27.31 (0.8)	24.32 (0.8)
128/0.01	26.28 (0.6)	27.13 (1.4)	23.63 (1.0)
256/0.01	25.87 (0.3)	26.21 (0.7)	23.64 (0.4)

Table C.12: The table states the results obtained on different hyperparameters for deep-net on link prediction TransR embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	26.82 (0.3)	28.39 (0.9)	25.12 (0.9)
128/64/0.001	26.84 (0.3)	28.51 (0.7)	24.27 (1.1)
256/64/0.001	26.25 (0.4)	26.85 (0.9)	23.09 (0.7)
64/128/0.001	27.00 (0.3)	28.72 (0.9)	25.34 (0.6)
128/128/0.001	26.46 (0.3)	27.68 (0.8)	23.26 (0.6)
256/128/0.001	26.24 (0.6)	27.12 (1.4)	23.40 (1.0)
64/256/0.001	26.89 (0.7)	28.28 (0.9)	25.04 (1.4)
128/256/0.001	26.46 (0.4)	27.35 (0.6)	23.85 (1.2)
256/256/0.001	26.05 (0.4)	26.68 (1.2)	23.41 (0.6)
64/64/0.005	27.15 (0.5)	28.44 (1.0)	26.06 (0.8)
128/64/0.005	26.55 (0.6)	27.65 (0.9)	24.91 (1.1)
256/64/0.005	26.57 (0.3)	27.56 (0.4)	24.27 (1.1)
64/128/0.005	27.19 (1.1)	27.67 (0.9)	24.96 (1.3)
128/128/0.005	26.69 (0.4)	27.68 (1.0)	24.60 (1.2)
256/128/0.005	26.45 (0.5)	27.58 (1.2)	23.91 (1.0)
64/256/0.005	26.47 (0.5)	27.56 (0.8)	24.19 (1.0)
128/256/0.005	26.57 (0.5)	27.72 (1.1)	24.25 (1.5)
256/256/0.005	26.17 (0.5)	26.73 (1.3)	23.60 (1.2)
64/64/0.01	26.64 (0.6)	27.60 (1.2)	24.01 (1.5)
128/64/0.01	26.57 (0.4)	27.43 (0.8)	23.60 (1.3)
256/64/0.01	26.59 (1.3)	26.96 (1.2)	23.41 (1.5)
64/128/0.01	26.97 (0.6)	28.03 (0.9)	24.84 (1.3)
128/128/0.01	26.66 (0.6)	27.91 (1.4)	23.86 (1.6)
256/128/0.01	26.05 (0.4)	26.52 (0.8)	22.06 (1.1)
64/256/0.01	26.61 (0.6)	27.30 (1.2)	24.20 (1.6)
128/256/0.01	26.41 (0.6)	27.08 (1.0)	23.42 (1.4)
256/256/0.01	26.63 (0.8)	27.34 (1.4)	23.86 (1.8)

Table C.13: The table states the results obtained on different hyperparameters for simple-net on link prediction TransD embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/0.001	29.09 (0.3)	33.71 (0.9)	26.79 (0.5)
128/0.001	28.86 (0.4)	32.96 (1.1)	26.78 (0.6)
256/0.001	28.44 (0.2)	31.65 (0.6)	26.78 (0.3)
64/0.005	29.06 (0.5)	33.03 (1.0)	27.89 (0.8)
128/0.005	28.57 (0.4)	31.71 (0.9)	27.13 (0.6)
256/0.005	28.37 (0.6)	31.21 (1.1)	26.96 (0.8)
64/0.01	28.88 (0.6)	32.37 (1.0)	27.39 (0.7)
128/0.01	28.72 (0.5)	32.19 (1.1)	27.19 (0.8)
256/0.01	28.50 (0.5)	31.53 (1.1)	26.91 (0.8)

Table C.14: The table states the results obtained on different hyperparameters for deep-net on link prediction TransD embedded data.

Parameters HL1/LR	**Macro precision [%] Mean (SD)**	**Macro recall [%] Mean (SD)**	**Macro F1-score [%] Mean (SD)**
64/64/0.001	29.21 (0.7)	33.22 (1.6)	28.17 (1.0)
128/64/0.001	28.96 (0.4)	32.62 (0.8)	27.98 (0.6)
256/64/0.001	28.90 (0.3)	32.37 (0.6)	28.08 (0.5)
64/128/0.001	29.39 (0.5)	33.72 (1.0)	28.41 (0.9)
128/128/0.001	28.91 (0.6)	32.59 (1.2)	27.66 (0.7)
256/128/0.001	28.75 (0.4)	32.23 (0.6)	27.53 (0.5)
64/256/0.001	29.42 (0.4)	33.52 (0.8)	28.32 (0.8)
128/256/0.001	29.20 (0.6)	32.68 (1.2)	27.91 (0.9)
256/256/0.001	29.03 (0.4)	32.55 (1.3)	27.79 (0.8)
64/64/0.005	30.25 (0.9)	33.26 (0.7)	28.78 (0.6)
128/64/0.005	29.60 (0.7)	32.80 (0.6)	28.54 (0.6)
256/64/0.005	29.55 (1.2)	32.69 (0.6)	28.39 (0.7)
64/128/0.005	29.36 (0.8)	32.84 (0.7)	28.06 (0.5)
128/128/0.005	30.10 (1.2)	33.08 (0.8)	28.39 (0.8)
256/128/0.005	29.55 (0.8)	32.40 (1.0)	27.76 (0.7)
64/256/0.005	28.60 (0.7)	32.30 (0.7)	27.46 (0.5)
128/256/0.005	30.19 (1.4)	32.81 (0.7)	27.65 (0.5)
256/256/0.005	29.10 (1.1)	32.66 (1.3)	27.73 (0.9)
64/64/0.01	29.85 (1.6)	32.46 (1.0)	28.11 (0.8)
128/64/0.01	32.17 (7.3)	32.96 (1.2)	28.16 (0.8)
256/64/0.01	29.18 (0.9)	32.45 (0.7)	27.68 (1.3)
64/128/0.01	29.50 (1.3)	32.10 (1.5)	27.87 (0.6)
128/128/0.01	28.83 (0.3)	32.48 (0.8)	27.60 (0.5)
256/128/0.01	28.66 (0.9)	32.20 (1.1)	27.71 (1.4)
64/256/0.01	28.47 (0.9)	32.14 (0.6)	27.14 (0.8)
128/256/0.01	29.08 (0.7)	31.66 (0.7)	27.53 (1.5)
256/256/0.01	28.50 (0.4)	31.93 (0.8)	27.90 (0.5)

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY