UNIVERSITY OF THE PELOPONNESE & NCSR “DEMOKRITOS”
MSC PROGRAMME IN DATA SCIENCE

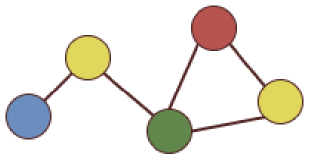
Representational learning on biological data

A study on polypharmacy side-effects
and graph embeddings

by

Symeon Panagiotoglou

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science



Supervisor: Anastasia Krithara
Dr.

Co-supervisor: Bougiatiotis Konstantinos
PhD Candidate

Athens, September 2022

Representational learning on biological graph data

Symeon Panagiotoglou

MSc. Thesis, MSc. Programme in Data Science

University of the Peloponnese & NCSR “Democritos”, September 2022

UNIVERSITY OF THE PELOPONNESE & NCSR “DEMOCRITOS” MSC PROGRAMME IN DATA SCIENCE

Declaration of authorship

1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where stated otherwise by reference or acknowledgment, the work presented is entirely my own.
2) I confirm that this thesis presented for the degree of Bachelor of Science in Informatics and Telecommunications, has
    I. Been composed entirely by myself
    II. Been solely the result of my own work
    III. Not been submitted for any other degree or professional qualification
3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Signature)

…………………………
Symeon Panagiotoglou

Athens, September 2022

Acknowledgments

I would like to thank my supervisor Anastasia Krithara for the opportunity to work on this thesis and special thanks to my co-supervisor Konstantinos Bougiatiotis for the time he spent and the guidance he offered me throughout this project. I would also like to thank the people that through their taxation made it possible for me to complete this programme tuition-free.

Περίληψη

Τα τελευταία χρόνια, οι γράφοι και τα νευρωνικά δίκτυα γράφων έχουν προσελκύσει το ενδιαφέρον στο χώρο της μηχανικής μάθησης και ειδικότερα σε βιολογικές εφαρμογές. Χρησιμοποιώντας ως αφετηρία το Decagon, ένα νευρωνικό δίκτυο γράφων, μελετάμε το πρόβλημα της πρόβλεψης παρενεργειών που προκύπτουν από την ταυτόχρονη χρήση πολλαπλών φαρμάκων, υλοποιώντας μια σειρά από baseline μοντέλα με σκοπό να εντοπίσουμε την κύρια πηγή της επιτυχίας του μοντέλου. Στη συνέχεια επικεντρωνόμαστε σε ενα υποσύνολο των αρχικών δεδομένων που αφορούν τις πιο σπάνιες παρενέργειες και δοκιμάζουμε γνωστά μοντέλα από τον χώρο των graph embeddings. Επιπλέον, εξετάζουμε αν η κανονικοποίηση των διανυσμάτων με βάση μια λογική tf-idf βελτιώνει την απόδοση. Τέλος, παραθέτουμε μια σειρά από patterns που προκύπτουν από τη χρήση του AnyBURL, ενός rule based μοντέλου στα δεδομένα μας.

Abstract

In recent years graphs, graph neural networks and graph embedding techniques are getting more attention in the area of machine learning in general, with biological applications being a major drive. Using Decagon, a graph neural network that predicts polypharmacy side-effects, as our starting point, we ımplement a number of baseline models in order to identify the aspects that play the bigger part in predicting side-effects among pairs of drugs. Later, we focus on a subset of the initial dataset containing only the rarest side-effects and experiment with well known models from the graph embeddings area. We examine whether a normalization of the feature vectors in a tf-idf fashion helps a message passing network improve its performance. Finally, we use AnyBURL, a rule based model, to identify patterns in our data.

Table of contents

List of tables

List of figures

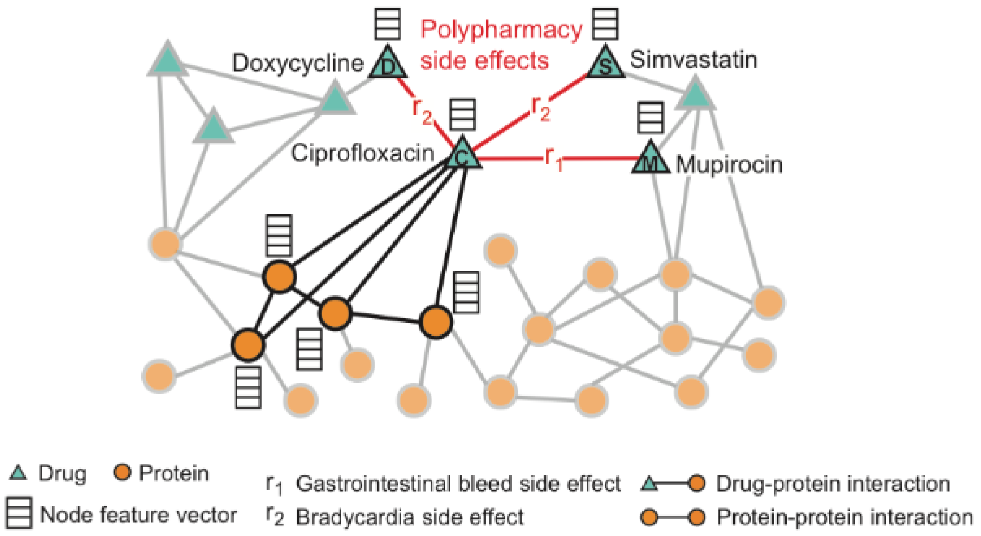
List of Abbreviations

se	side-effect
mono	mono pharmacy
poly	polypharmacy
d	drug
p	protein
r	relation

1. Introduction

Polypharmacy is a term describing a situation where a patient receives more than one medication at the same time, a situation quite common especially in cases of complex diseases or co-existing conditions. However, a major consequence of polypharmacy is a much higher risk of adverse side-effects for the patient, due to drug – drug interactions. The possible combinations of all the known drugs are just too many and too expensive to test in clinical trials while these complex relationships may be rare and hard to observe. (Bansal et al., 2014) Finding a way to predict such adverse side-effects would be of great importance for the science of pharmacology, pharmaceutical companies and most importantly, patients' well being.

The project in hand is inspired by a recent paper on modeling polypharmacy side-effects with graph convolutional networks (Zitnik et al., 2018). Previous approaches were limited in predicting a single total score/probability of drug interaction for a given pair of drugs (Trouillon et al., 2016). This could be an indication of the severity of side-effects that could appear but without having any clue about what these side-effects would be in particular. Inspired by GCNs and autoencoders, Decagon is a graph convolutional neural network for multi-relational link prediction in multimodal networks, something that allows for predicting specific side-effects for each pair of drugs.



Polypharmacy side effects
Doxycycline
D
Simvastatin
S
Ciprofloxacin
C
Mupirocin
M
r2
r2
r1
Drug
Protein
Node feature vector
r1 Gastrointestinal bleed side effect
r2 Bradycardia side effect
Drug-protein interaction
Protein-protein interaction

Figure 1.1 An illustration of the full Decagon graph taken from the original paper (Zitnik et al., 2018). The graph contains drugs and proteins as entities connected by polypharmacy side-effects (d - d), interaction (p - p), target (d- p) which are represented as edges. Individual drug side-effects are used as feature vectors for the drugs.

The idea behind Decagon is to combine information from multiple sources and combine it through an end to end encoder – decoder learning process based on neural networks. So they construct a heterogeneous graph with proteins and drugs as entities. It includes protein–protein interactions, drug–protein interactions and polypharmacy side-effects, which are represented as drug–drug interactions, where each side-effect is an edge of a different type. Every drug node also has a feature vector corresponding to its side-effects as extra information. The resulting multi-relational graph can be seen in Figure 1.1.

1.1 Overview of this work

Starting this project there were a number of aspects in the polypharmacy problem that seem interesting to investigate. First, we wanted to identify whether the structure of the network or the additional featural information was the main contribution for a successful prediction model. Regardless of the answer on the above question, experimenting with additional information regarding polypharmacy problem, not included in the original dataset, that could be incorporated in the form of features or networks would be of interest[1]. Another idea was to test whether Decagon's performance could be improved by replacing its neural network components with other more recent types of neural networks[2].

Since the original dataset is quite big and requires a lot of resources, focusing on a subpart of the original data so that we could proceed more in depth was also an idea. We also believe there is margin in improving the data fed on the model in order to improve the performance, since, as an example, all side effect data are binary without any information about the intensity or the frequency of a side-effect associated with each drug or pair of drugs.

Difficulties on reproducibility, literature review and implementation limitations reformed and eventually shaped the scope of the project in hand as described in the next paragraphs.

To identify whether network structure or featural information is more influential in prediction of polypharmacy side-effects we implemented a number of baseline models each one incorporating different kinds of information. We were able to identify that for the particular problem, structural information is more important than features containing individual side-effects information.

Literature review and data exploration led us to focus on a subgraph of the original data containing the rarest side-effects, on which we tried a number of well known models from the area of graph embeddings and AnyBURL, a recently proposed rule based model to solve this multi-relational link prediction task. AnyBURL, a light model in terms of training time and computational requirements, outperformed the graph embedding models. It also provides some sort of explainability through the rules it learns. Geometric models like TransE and RotatE

[1] For example chemical footprint of the drugs or drug - disease network
[2] Specifically we had graph attention networks in mind

proved to be fast and good baselines while RESCAL and RGCN were inferior both in performance and training time.

Continuing in the direction of identifying the relative importance of structure against featural information, we used models that can handle features and compared their performance with randomly initialized feature vectors as opposed to features containing individual side-effects information. This additional information actually proved beneficial for the models. Taking a step further, we also tried a tf-idf like way of normalizing our featural information as an attempt to improve performance by providing more quality data. We did not observe any improvement on the results by doing this.

Finally, we explore the performance of the models per side-effect and have a look at the patterns discovered from the rule based model.

In the following chapter we have a brief overview of the related work on graphs and graph embeddings as well as a few notes on the aspects of polypharmacy side-effects and previous works on the issue.

In the third chapter we present an exploratory analysis of the dataset that we use both to give an illustrative and clear image of the data to the reader, but also gain insights on the data that helped us shape the direction of this project.

In the fourth chapter we present the methodology in detail. The models we tried, the ideas we experimented on and our drives.

In the fifth chapter we present the experimental setup and the specifics of our implementation. We extend on the way we handled this link prediction task on the different experiments, the procedure we followed, but also we discuss things that worked and things that did not.

In the sixth chapter we present the results of the different experiments and comment on the patterns that were identified from the rule based model.

Finally, In the seventh chapter we dedicate some space for retrospective view, contributions of this project and thoughts on future work.

2. Related Work

2.1 Graphs

Graphs are data structures able to model a set of objects (nodes) and their relationships (edges). Graphs are the natural method of representation in areas where relation amongst objects of interest is of high importance, like social networks, biological systems, knowledge graphs, recommendation systems, citation networks etc.

There are multiple ways to categorize graphs based on different aspects. Graphs can be undirected or directed. A typical example of an undirected edge is a friendship edge between two facebook users while an example of directed edge is a follow edge in tweeter. Graphs may contain a single type of nodes (homogeneous) or more (heterogeneous). A graph can contain a single type of edges or more in which case it is called multi-relational. Additionally graphs can contain auxiliary information like labels and features for nodes and/or for edges.

A special example of multi-relational and heterogeneous graphs are knowledge graphs or knowledge bases. They represent knowledge in the form of triplets (subject, relation, object) where subjects and objects are entities represented by nodes in the graph and relation are edges that connect them. Typical examples of triples in a knowledge base could be (Athens, is capital of Greece), (MSc Data Science, takes place in Athens) etc. While these relations are asymmetrical there are also symmetric relations like (Hydrogen, interacts with, Oxygen).

2.1.1 Graph tasks

Node classification is one of the most common tasks in a graph. Most of the time, it is a form of semi-supervised learning, where labels are only available for a small proportion of nodes, with the goal being to label the full graph based only on this small initial set. Classifying documents, web pages, or individuals in a social network into different categories are some examples.

Link prediction has important applications in a wide range of subjects and tasks, from recommendation systems to biological interaction graphs and from knowledge graphs to social media. The goal is to predict missing edges, or edges that are likely to form in the future. Predicting missing friendship links in social networks, affinities between users and movies, or missing relationships in noisy, incomplete knowledge graphs are typical examples. As in our case, predicting links in knowledge graphs is a method for recommending new directions for lab research.

A special case of link prediction is knowledge graph completion. Real world knowledge graphs tend to be large, complex and noisy. And of course, incomplete. There is limited value in creating such a base by hand, with strict rules or from a single source. In most of the cases it is just a way to put some order in vast amounts of information, from various sources. So having such a graph, the next, natural step, is to try to predict missing links. Triplets that should (or could) be in the graph but are currently missing. This could either improve the completeness of the graph (and if not trivial links) improve our knowledge about the system we study, or even act as a recommendation tool that would suggest movies likely to entertain the user, friends to connect with in a social platform or even possible directions for further survey in a scientific field.

Community detection or clustering is another common task that has countless applications from computational biology to marketing (related products, client segmentation etc). In contrast with classification, here the labels are not known in advance and we rely on the graph data to give a good way to categorize data based on similarity.

Node embeddings can also be useful for **visualization** purposes, when combined with techniques such as t-SNE or principal components analysis (PCA) in order to generate 2D visualizations of graphs.

Graph classification is another task where entities are represented as whole graphs. Classifying graphs that correspond to different molecules is the most prominent application domain.

2.1.2 Representation learning - Graph embeddings

Graphs, aside from being useful as structured knowledge repositories, are also growing popular in modern machine learning. Classifying the role of a protein in a biological graph, recommending a new connection in social media or repositioning an existing drug for a new disease are typical examples.

Traditional machine learning algorithms require data to be in a more or less tabular form with every instance described by a set of features. Therefore It is not feasible to pass a graph structure into a, for example, logistic regression or SVM model as is. We first need to transform the components of the graphs into vectors that will represent them meaningfully.

The first attempts to extract structural information in the form of features were based on summary statistics and hand feature engineering (Liben-Nowell & Kleinberg, 2007). It is obvious that these methods are very limited in a sense that the process is expensive and specific for each task but also the features are static and fixed throughout the learning process.

The idea behind representation learning approaches is to learn a mapping that embeds nodes, or entire graphs, as points in a low-dimensional vector space. The goal is to optimize this mapping so that geometric relationships in the embedding space reflect the structure of the original graph. Feature engineering is not a preprocessing step anymore, but a machine learning task itself where the features need to be learned. In other words, graph embeddings are low-dimensional vectors that capture the essence of a graph's nodes (or edges or subgraphs) and can be used as features in a consequent machine learning task.

To evaluate an embedding we can either use a distance metric (or basically any kind of metric, like reconstruction error) in the vector space or use the performance in a specific downstream task. We would like similar nodes to be closer together. However, defining what we mean by “similar” is also a matter of choice. Depending on the task and the specifics of the graph we can consider two nodes to be similar when they share a lot of common neighbors or when they have a similar structural role in the graph.

First attempts to learn embeddings were based on **matrix factorization** approaches, using the adjacency matrix of the graph, Laplacian eigenmaps and inner product methods (like GraRep (Cao et al., 2015) , and HOPE (Ou et al., 2016)) The basic idea is that the dot product of a pair of embeddings should reflect the similarity (as defined) of those two nodes in the graph, for example if they are connected or not.

Other methods like DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) used the concept of **random walks**. Their key innovation is optimizing the node embeddings so that nodes have similar embeddings if they tend to co-occur on short random walks over the graph. While Deepwalk uses simple random walks, node2vec introduces two random walk hyperparameters that bias the random walk The one controls the likelihood of the walk immediately revisiting a node, while the other controls the likelihood of the walk revisiting a node’s one-hop neighborhood.   By introducing these hyperparameters, node2vec is able to smoothly interpolate between walks that are more akin to breadth-first or depth-first search, capturing the right blend of neighborhood similarity and structural role.

Models described so far have a number of limitations. No parameters are shared between nodes in the encoder. Shallow embedding also fails to leverage node attributes during encoding. Shallow embedding methods are inherently transductive. They can only generate embeddings for nodes that were present during the training phase, and they cannot generate embeddings for previously unseen nodes. (Zhou et al., 2020)

A third and most recent approach is representation learning through **neural networks**. Inspired by CNN’s performance and interpretability, GNNs took off recently. Key ingredients of CNNs are local connection, shared weights and use of multiple layers. We can see how these things relate on graphs and why GNNs are a generalization of CNN to solve their limitations to only operate on regular Euclidean data like images or text to be applied on grids.

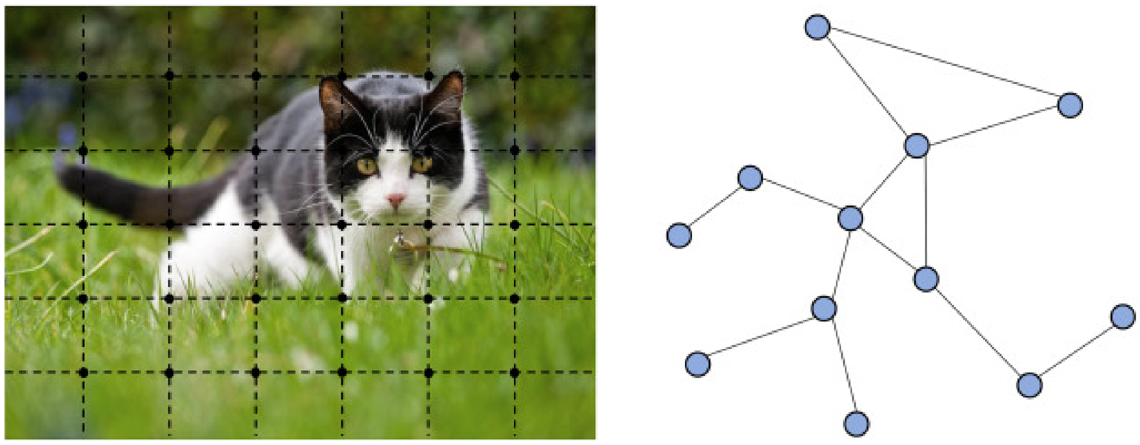


Figure 2.1 CNNs are operating on text or images which can be seen as grids on the euclidean space (left). These structures can be considered as a special instance of graphs that in the general case live on non-euclidean spaces (right) In that base GNNs are generalization of the CNNs. Image from (Zhou et al., 2020)

The first paper that introduced a graph neural network was (Scarselli et al., 2009). After that there are a number of GNN models proposed in the course of the last few years. Apart from the individual models there have been proposed several unifying frameworks aiming to integrate different models under a general umbrella. Most famous framework is the MPNN, i.e. message passing neural networks proposed by (Gilmer et al., 2017).

The model consists of the message passing phase and the read out phase. These phases can have different settings and thus express a number of different models. In the message passing phase each node starts with a feature vector. For each node we aggregate all neighboring nodes' vectors (message) with itself and update its vector. We repeat this process a number of times. Finally, In the read out phase we can create a final feature vector that represents the whole graph. A notable example of message passing neural networks are GCN (Kipf & Welling, 2016) and R-GCN (Schlichtkrull et al., 2018), its counterpart for multi-relational graphs.

2.1.2.1 Knowledge graph models taxonomy

If we follow the taxonomy of knowledge graph models in a multi-relational setting proposed in (Rossi et al., 2021) we can group them into three major categories, tensor decomposition models, geometric models and deep learning models.

Tensor decomposition models consider the KG as a 3D adjacency matrix which is partly observable due to incompleteness of the graph. The tensor is decomposed into a combination of low-dimensional vectors that are used as representations of entities and relations. The embeddings are learned as usual by optimizing the scoring function for all training triplets. These models tend to employ few or no shared parameters at all, something that makes them easy to train but without the benefits that come from knowledge transfer. Typical models of this type are DistMult (Yang et al., 2014), Rescal (Nickel et al., 2011) and TuckER (Balazevic et al., 2019)

Geometric models interpret relations as geometric transformations in the latent space. Given a triplet, the head embedding undergoes a spatial transformation τ that uses the values of the relation embedding as parameters. We measure the plausibility of a triplet by the distance between the vectors after the transformation based on a distance function δ.

Shared parameters are usually not used in these models and they can be further divided into three subcategories. Pure translational models like TransE (Bordes et al., 2013) where the relation vector is just added to the head vector, and we expect to land in a position close to the tail vector. Roto-translational models like RotatE (Sun et al., 2019) that instead of or additionally to the pure translations use rotation-like transformations to the embedding vectors. Finally, models like CrossE (Zhang et al., 2019) may use more than one embedding for each element of the graph, something that yields of course a larger number of parameters to learn.

Deep learning models like ConvE (Dettmers et al., 2018) use neural networks. A great variety of layers have been proposed, applying quite different operations to the data. Depending on the task, different types might be more appropriate, for example convolutional layers, that learn convolution kernels, or recurrent layers, that handle sequential inputs in a recursive fashion. In these models KG embeddings are usually learned jointly with the parameters of the layers and these shared parameters make these models more expressive, but also potentially heavier, harder to train, and more prone to overfitting.

2.2 Polypharmacy side-effects prediction

2.2.1 First attempts

Polypharmacy treatment - treatment of a patient with a combination of drugs- is a powerful tool in our medical arsenal when it comes to diseases that are caused by complex processes and single drug treatments are inefficient. However, it poses a higher risk of side-effects due to the interaction of the drugs (Tatonetti et al., 2012). Beside the obvious effect on the patients' wellbeing it also has a noticeable economical impact (Kantor et al., 2015).

Identifying polypharmacy side-effects through clinical trials for all the possible drug pairs (let alone all higher order combinations) is challenging and expensive[3] (Bansal et al., 2014). In the past, computational methods have been developed to help with this issue based on various concepts, like (Lewis et al., 2015) and (Chen et al., 2016) just to name a few. All these approaches have in common is that they predict a single score for each drug pair that represents the possibility or/and strength of the interaction, giving no further clue for the type or the specific side-effects that may arise. Decagon (Zitnik et al., 2018) comes to fill this gap and predict specific side-effects for any given pair of drugs[4].

[3] For any n drugs there are ~n^2 durg pairs that should be put into test.
[4] More details about this were given in the first chapter

2.2.2 Decagon inspired approaches

Due to the complexity of the model and the relationships in polypharmacy the way Decagon is implemented has a really high cost both in space and in time efficiency. In (Xu et al., 2020) a different approach is used based on the same BioSNAP Decagon datasets. Although they use the same end-to-end training approach and the same preprocessing as in the original paper, instead of building a single multi-relational and heterogeneous graph here the authors take a gradual approach.

First the model learns the embeddings for the proteins based on the PPI network. Then they propagate such embeddings to the drug – drug graph via the protein – drug graph. Learn the final drug embeddings and predict side-effects in the drug – drug graph. The drug representation is produced by combining protein embedding and other available drug information.

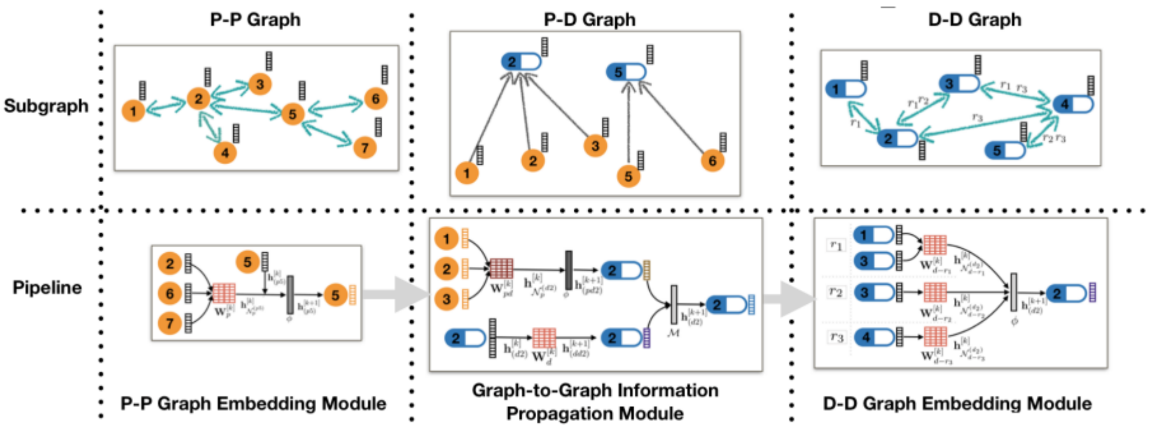

P-P Graph
P-D Graph
D-D Graph
Subgraph
Pipeline
P-P Graph Embedding Module
Graph-to-Graph Information Propagation Module
D-D Graph Embedding Module

Figure 2.2 Information propagation in TIP encoder (Xu et al., 2020)

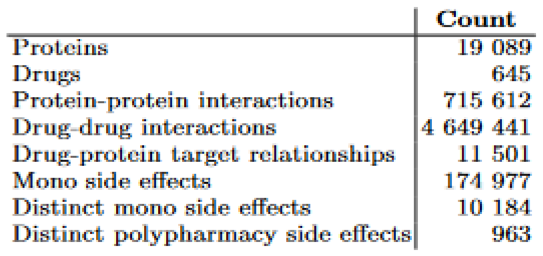
We can still consider this process as a combination of an encoder and a decoder., where encoder is a sequence of different message passing neural networks. A GCN module that learns p-p embeddings, a graph-to-graph information propagation module consisting of a single layer message passing network and a linear transformation followed by an activation function, and a d-d graph embedding module that is actually an R-GCN encoder with a basis-decomposition regularization.

Another approach was presented in (Malone et al., 2018). The authors use the same datasets and preprocessing but use a quite different approach, based on rule based features and KBLRN (Garcia-Duran & Niepert, 2017), an end to end knowledge graph embeddings learning framework.

3. Data

3.1 Overview of the dataset

The Decagon full network comprises of four separate datasets. A PPI network, a mono pharmacy drug side-effect, a polypharmacy drug side-effect and drug - protein targets. The datasets are fully described in the original paper. Here we just summarize some basic aspects of them.



	Count
Proteins	19 089
Drugs	645
Protein-protein interactions	715 612
Drug-drug interactions	4 649 441
Drug-protein target relationships	11 501
Mono side effects	174 977
Distinct mono side effects	10 184
Distinct polypharmacy side effects	963

Table 3.1 A summary of Decagon full dataset specifications. Taken from (Malone et al., 2018)

The human protein–protein interaction (PPI) network as created by (Chatr-Aryamontri et al., 2015) and (Menche et al., 2015) and extended by (Rolland et al., 2014) and (Szklarczyk et al., 2017) contains physical interactions experimentally documented in humans, such as metabolic enzyme-coupled interactions and signaling interactions. The network is unweighted and undirected with 19 085 proteins and 719 402 physical interactions. The dataset consists of triples in the form *protein - interacts_with - protein*.

Drug-targeted proteins were taken from the STITCH (Search Tool for InTeractions of CHemicals) database (Szklarczyk et al., 2016). Only interactions between drugs and target proteins that had been experimentally verified were considered. The dataset consists of triples in the form *drug - targets - protein*.

Regarding the mono side-effect data, the SIDER (side-effect Resource) (Kuhn et al., 2016) was used which was obtained by mining adverse events from the drug label text, integrated with the OFFSIDES database. The OFFSIDES database (Tatonetti et al., 2012) was generated using adverse event reporting systems that collect reports from doctors, patients and drug companies. The dataset consists of triples in the form *drug - causes - side_effect.*

For polypharmacy side-effects, the TWOSIDES was used, which details 1318 side-effects types across over 63K drug combinations, which are greater than expected given the effects of either drug in the combination individually. Like OFFSIDES, TWOSIDES was generated from adverse event reporting systems (Tatonetti et al., 2012).

The final network after linking entity vocabularies used by different databases has 645 drug and approximately 19 thousands protein nodes connected by approximately 715 thousand protein – protein, 4.5 million drug–drug and 19 thousand drug – protein edges. Side-effect synonyms were eliminated and one side-effect vocabulary was used to construct all datasets. The aggregated statistics of the dataset are shown in Table 3.1

3.2 Distribution of entities and edges

3.2.1 Protein - protein interaction dataset

The Decagon protein - protein interaction dataset contains 19,085 proteins and 719,402 known and experimentally documented physical interactions among them in the human body. As shown in Figure 3.1, the distribution of the edges per node follows the power-law distribution, with most of the nodes having a few or a few dozens of neighbors, some of them having more and only in rare cases we see some having as much as three or four hundreds of interactions.

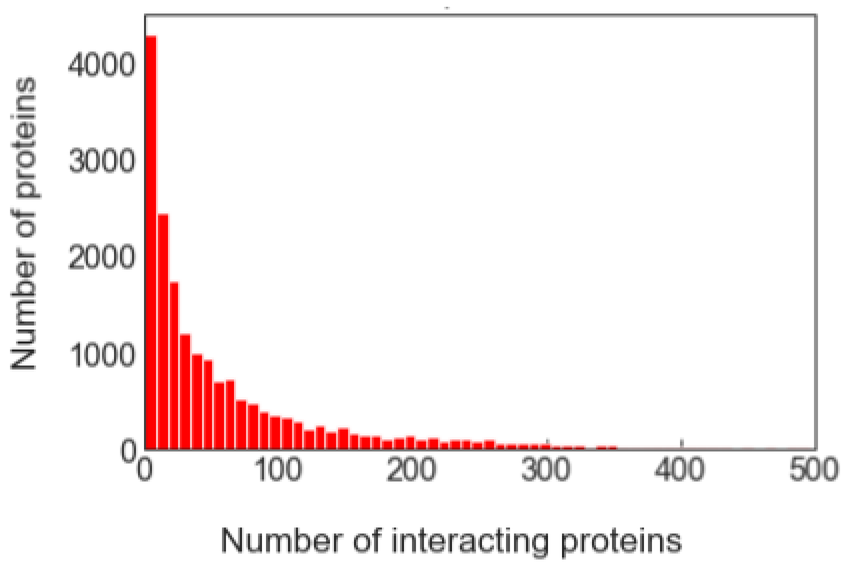


Figure 3.1 Distribution of protein interactions with other proteins. Most of them are interacting with a few dozens of other proteins, while a few are interacting with more than 200 proteins.

3.2.2 Drug - protein interaction dataset

Regarding the amount of drugs that actually target some protein, only 284 of the 645 do target at least one (~44%) and we can see the distribution in Figure 3.2. Here we have a bimodal distribution with almost half of them targeting less than ten proteins and a non-negligible number of drugs targeting 150-200 proteins.

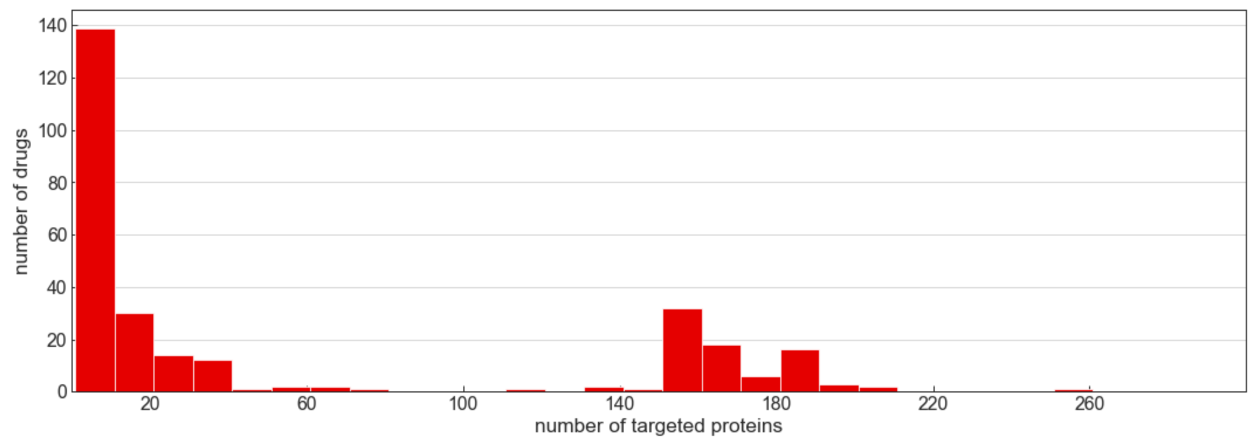


Figure 3.2 Distribution of drugs targeting proteins. We can see drugs being separated into two main groups. Drugs that target a few proteins up to 40 and drugs that target 150 to 200 distinct proteins. From the histogram we completely removed drugs that do not target specific proteins.

The total number of drug - protein edges is small compared to the total number of edges in the dataset. Most drugs don't even have protein targets, but even for those that have multiple connections, this information could be buried under the volume of polypharmacy edges, especially for the drugs in the first mode of the distribution having a few targets. This is one of our motivations for focusing on the rare side-effects subgraph, as we'll discuss later.

3.2.3 Individual drugs side-effects

The dataset that contains the individual drug side-effects has also some interesting aspects. The distribution roughly resembles a binomial with most of the drugs being associated with a few hundred mono side-effects (Figure 3.3). Plotting the inverse diagram (Figure 3.4) in order to explore the frequency of appearance of each side-effect, we note three things. The first is that almost half of the mono side-effects included in the dataset (approximately five thousand) are associated with five or less drugs. Another thing is that there are some of them that are associated with more than a hundred drugs (we remind here that we examine a total of 645 drugs) and a few that reach up to 200-300 associations.

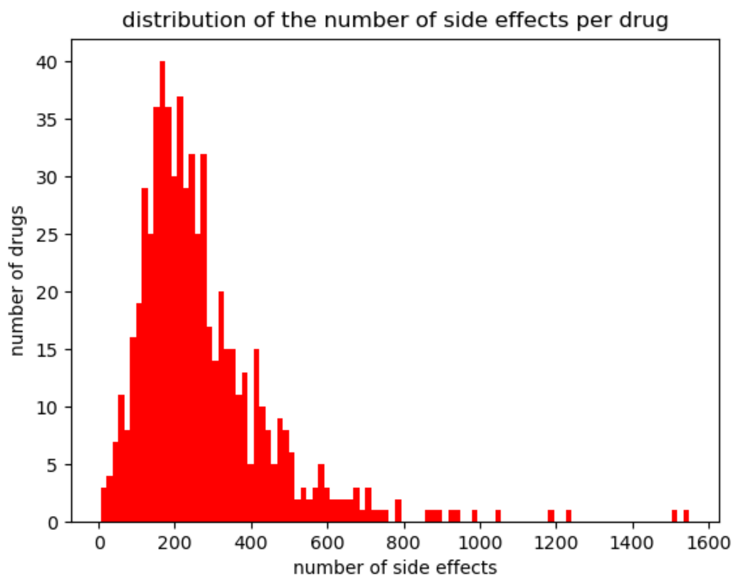


Figure 3.3 Distribution of individual side-effects per drug. The number of individual side-effects each drug is associated with varies widely. In addition to the side effects extracted from drug label text, the information is integrated with information from adverse event reporting systems.

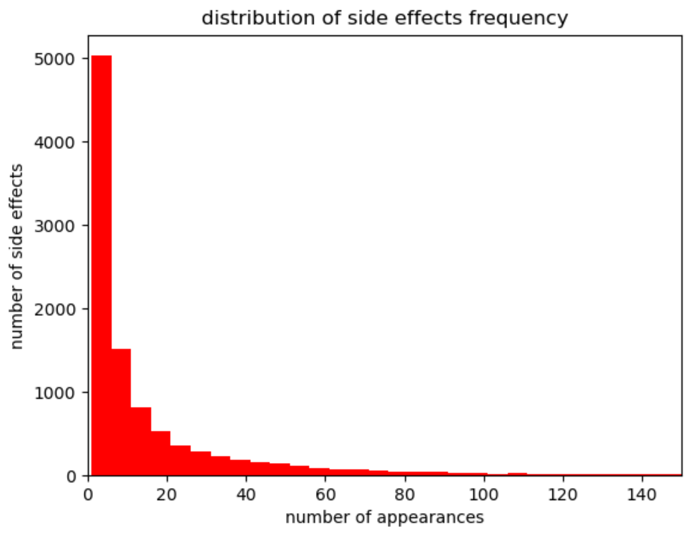


Figure 3.4 There are many side-effects that are only associated with a handful of drugs. Almost half of the total 10000 side-effects are only associated with less than five drugs. Frequent side-effects are associated with hundreds of drugs (this diagram is trimmed from the right excluding these extreme cases)

The frequency of appearance of the individual side-effects is of some importance since they are used as features for the drugs in order to improve performance. The side-effect datasets are partly based on adverse event reporting systems. The other, main, source for the mono side-effect dataset is the text descriptions that come with the drugs. What this means is that the datasets are possibly dirty and incomplete. Lack of any quantitative measure about frequency of a drug - side-effect association or its severity, leaves plenty room for improvement on the quality of the dataset.

Another consideration is that the side-effects that are included in the polypharmacy side effects are removed from the mono side-effect dataset as a conservative approach from the authors of Decagon in an attempt to avoid any leakage and reliably assess the model performance, leading to many of the expected common symptoms to be missing from the mono side-effects dataset. We will come back to this but for now we just include Table 3.3 with some more detailed stats about mono side-effect dataset distribution and Table 3.4 with the most frequent side-effects in the dataset.

side-effects	**Drugs associated**
32	> 200
116	> 150
289	> 100
5026	<= 5
3175	<= 2
2032	= 1

Table 3.2 Statistics on the frequency of individual side-effects. Around two thousand side-effects are only associated with one drug. Almost half are associated with five or less drugs while on the other hand, some are associated with almost one third of the drugs

side-effect	appearances
general physical health deterioration	301
hypoaesthesia	279
mental status changes	278
tooth extraction	276
emotional distress	275
alanine aminotransferase increased	273
condition aggravated	270
pollakiuria	267
staphylococcal infection	266
blood creatinine increased	263
spinal osteoarthritis	263
bone disorder	263
dysgeusia	257
anhedonia	249
osteopenia	243
congenital mitral valve incompetence	242
impaired healing	238
congenital tricuspid valve incompetence	238
disorder of globe	231
blood alkaline phosphatase increased	231

Table 3.3 Most common side-effects in our dataset. Some of those that we might expect to see here are missing since any side-effect that appears on the polypharmacy dataset is removed from this one, for information leakage issues. See (Zitnik et al., 2018) for more details.

3.2.3 Polypharmacy side-effects

After removing rarely documented side-effects with less than five hundred appearances in the original data[5], the polypharmacy side-effect dataset contains 645 drugs, 963 polypharmacy side-effects and approximately 4.5 million triples (drug – side-effect - drug) in total. The first interesting thing is that from the 645 x 645 = 416025 possible drug pairs, we have at least one documented polypharmacy side-effect for 63472 of them in our dataset (approximately 15% of total). Along with all these pairs that we know not a single side-effect that appears when taken together, It comes as no surprise that for many of the drug pairs we have multiple documented side-effects. The exact distribution can be seen in the figure 3.3.

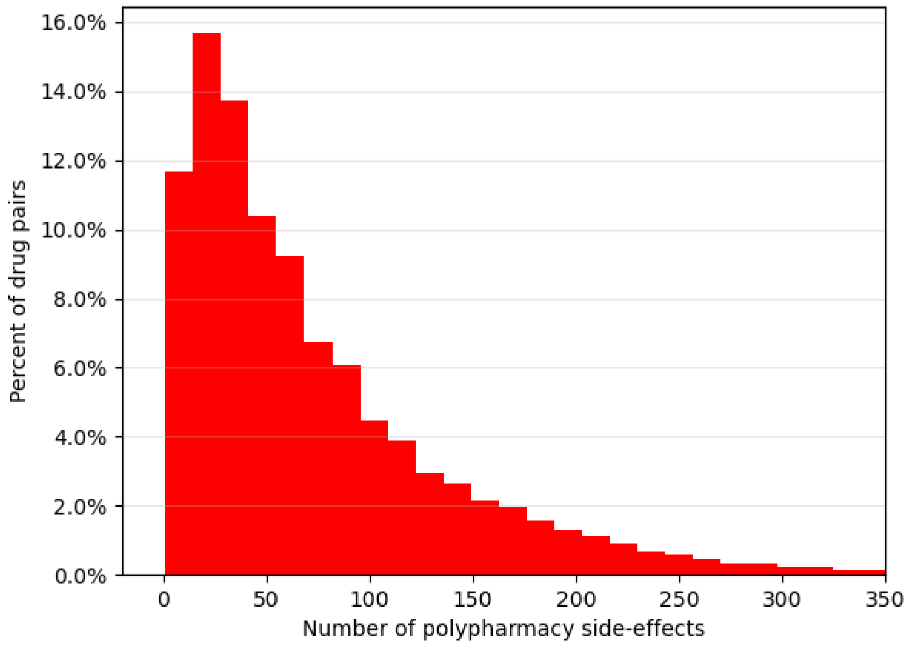


Figure 3.5 Of all the possible drug pairs, around fifteen percent manifest at least one known polypharmacy side-effect additionally to the individual side-effects of each drug. Some of them are associated with hundreds of additional side-effects. In this graph we only include pairs with at least one known polypharmacy side-effect.

Regarding the polypharmacy side-effects, obviously they also don’t appear with the same frequency in our dataset. In the list with the most frequent ones (Table 3.2), there are usual suspects like arterial pressure decrease, pain, fatigue, difficulty in breathing and nausea. In the next section we will examine how these common side-effects affect the prediction task.

[5] Same as the preprocessing that is followed in the decagon paper

Side Effect	Frequency
arterial pressure NOS decreased	28568
anaemia	27006
Difficulty breathing	26037
nausea	25190
neumonia	24430
Fatigue	24260
Pain	23894
diarrhea	23848
asthenia	23515
emesis	23043
edema extremities	21981
body temperature increased	21806

Worst performing side effects	AUPRC
Bleeding	0.679
Increased body temp.	0.680
Emesis	0.693
Renal disorder	0.694
Leucopenia	0.695
Diarrhea	0.705
Icterus	0.707
Nausea	0.711
Itch	0.712
Anaemia	0.712

Table 3.4 (left) Top polypharmacy side-effects that appear among pairs of drugs. (right) The worst performing side-effects in Decagon. We observe many common side-effects existing on both lists.

3.3 Rare side-effects subgraph

3.3.1 Why rare side-effects

Predicting a specific side-effect does not always bring the same added value. Since exhaustive clinical research is expensive and practically impossible, the motivation of predicting polypharmacy side-effects through machine learning methods is to identify good candidate side-effects and/or drug pairs in order to help clinical trials focus on these candidates so that resources can be saved or be better distributed and the research can proceed further and faster (Tatonetti et al., 2012). Rare side-effects are more difficult to spot and hence of more special interest in this context.

The original dataset contains more than a thousand polypharmacy side-effects. The most frequent of them appear in almost half of the drug pairs, as we've seen. This can lead to two separate issues. According to the original paper, the worst predicted side-effects tend to be common side-effects and/or have non molecular origins (see Table 3.4). These side-effects, due to their commonality, are easy to be associated with a large number of drug pairs without us being in a position to model them based on our data which mainly carry pharmacogenomic information.

In addition, due to their numbers, they can act as a kind of noise that lessens the general performance of the model and make modeling of the rare relations of the graph more difficult as rare side-effects are less represented in the full graph. Protein related edges could also be benefited from removing common side effects as they are also few compared to polypharmacy edges and focusing on rare side-effects will make them more dense in the graph..

3.3.2 Subgraph characteristics

After creating some subsets of the original combo dataset by setting different thresholds on side-effects to keep, we examined the graph statistics on Table 3.5. We wanted to experiment in a smaller graph that at the same time retained the qualitative characteristics of the full graph. Limited to the rarest 50 side-effects we can keep more than 80% of the drugs but there are a lot of nodes with one or few edges, unlike the initial graph. By increasing the threshold to 300 rarest side-effects, the graph has ten times more edges (while 10 times less than the full dataset) includes more than 95% of the drugs and the pairs of drugs with only one side-effect are sparse, meaning that the graph is strongly multirelational again.

# of side-effects to keep	**drugs**	**edges**	**Unique drug pairs**
50	559	27,815	15,563
100	598	62,202	24,602
200	626	152,565	36,442
300	631	283,585	45,596
963 (full)	645	4,649,441	63,472

Table 3.5 Polypharmacy dataset stats depending on the number of side-effects to keep.

We decided this was a balanced trade off among similarity to the original graph and size reduction, so we used the mini graph with 50 side-effects for initial testing and tuning and later proceeded to experiment on the bigger subgraph of 300 hundred rarest side-effects on which we report our results. On Figure 3.6 we see a comparison of side-effect frequency distribution. On the rare subgraph all side-effects are more balanced in terms of frequency, unlike the original graph where some side-effects (types of relations) were dominant and possibly make modeling of the less numerous side-effects challenging.

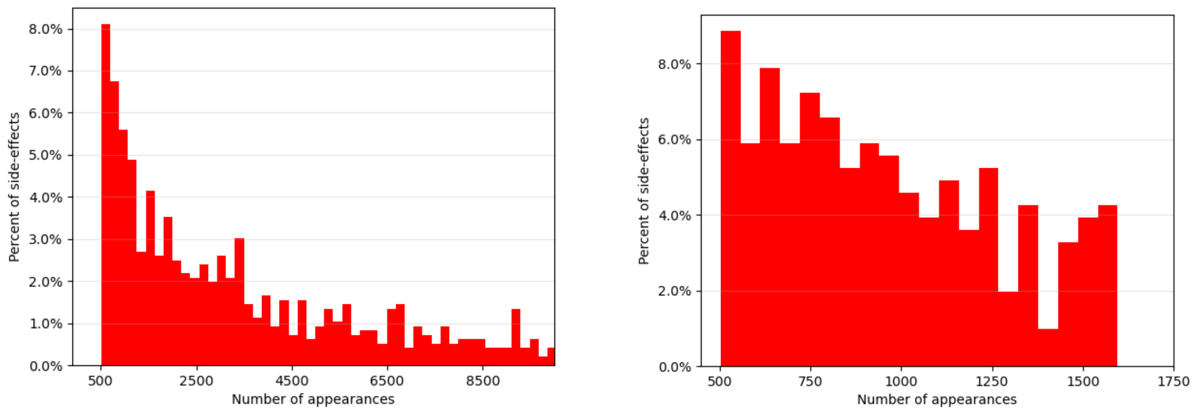


Figure 3.6 Frequency of appearance of polypharmacy side-effects on the full dataset (left), on the rare side-effects subset (right). On the full dataset we have side-effects that dominate the dataset merely by their number with some associated with more than 10 K drug pairs. On the rare subset, the situation is much more balanced.

4. Methodology

Our experimentation is focusing on trying different models and identifying behaviors, interesting points and patterns that could give some insight about the Decagon dataset or the models themselves. The experiments we performed can be divided into two main categories.

There are many ways to deal with a link prediction task in a multi-relational setting. What we choose has a major impact on the process that we follow during the training but most importantly in the evaluation. We will shortly introduce two ways that are of interest for our project. In both scenarios, we split the dataset triplets into train and test sets and we create negative samples (if not already included in the data) for the train set to allow the model to learn. Things are getting more interesting in the valuation phase.

For the first scenario the creation of negative samples in the test set is required so that the model is expected to predict (for each relation) whether each specific edge is positive or negative. This setting is often used in single-relational graphs and has many similarities with a binary classification problem and it is no surprise that similar metrics are used to evaluate the performance. AUROC curve, precision-recall, f1-score, and even accuracy can be used in this case.

In multi-relational graphs, a different setting is usually the way to go. For every triplet in the test set, the model is evaluated in its ability to assign high probability on the correct tail, given the head and the relation[6]. The task can be described as (h, r, ?). The performance is typically evaluated with some rank based metric. For every triplet in the test set, the model assigns a score for every possible tail, trying to assign the correct one a high score so that it is ranked in the highest place. In this case there is no need for negative samples in the test set.

Coming back to our project, the first half (section 4.1) includes experiments that were implemented and performed in the full dataset in a way that the results would be comparable with the results from the original Decagon paper. That means that we treated the link prediction problem the first way we described, like the authors did, and we used the same metrics to evaluate the models.

For the rest of the experiments we used a trimmed down version of the dataset containing only the rarest side-effects.The motivation for this choice is presented in the previous chapter. Since the results would by default not be comparable with the previous results, we chose to also switch the task to the second scenario we described above which is more typical for multi-relational graphs.

The main advantage of the second approach is the lack of need for negative samples in the test set. The choice of the test set can already have an impact on the performance of the model.

[6] The task can be the other way around, predicting the head given the relation and the tail. In our case all the triplets are symmetric so we don't have to worry about this. There is an easy to follow explanation on the variations of this task on the Pykeen documentation.
https://pykeen.readthedocs.io/en/stable/tutorial/understanding_evaluation.html

Unlike a dataset with independent observations, removing edges from a graph can have more complicated ramifications on the training. The creation of the negative samples, being a problem with high degrees of freedom, it introduces variance on the evaluation process that consequently leads to reliability and reproducibility issues as described in (Yang et al., 2015)

4.1 Baseline approaches on full dataset

The main idea here is that instead of running and expanding an effective and novel, but really heavy and costly model like Decagon or TIP, we could try various simpler and lighter models to see how they perform. Starting from simple and gradually growing complex we would try to identify what aspect of the problem has the greater impact in the prediction performance. We began with a model focusing only on the network information and gradually added more information in the form of feature vectors. The last model was designed to be like one of the baseline models described in the original paper (referred to as ‘concatenated features’).

In all three models, entities were depicted as n-dimensional vectors that were eventually concatenated or merged to form the representation of each drug pair. What differentiates each model is the information reflected on these feature vectors and the way it is constructed as depicted in Figure 4.1. These representations are fed to *n* individual classifiers, one for every relation, who predict whether the particular pair of drugs should be associated with the respective side-effect. For the training and evaluation we used negative samples that were created by uniformly replacing the tail of each triplet with another entity.

4.1.1 Network only

For the first model we focus on the network and add no extra feature information, so each drug is represented by its one-hot encoded index. For each pair of drugs we merge the representations of each drug so we end up with a vector of *number_of_drugs* length, with two aces indicating the two drugs of the pair.

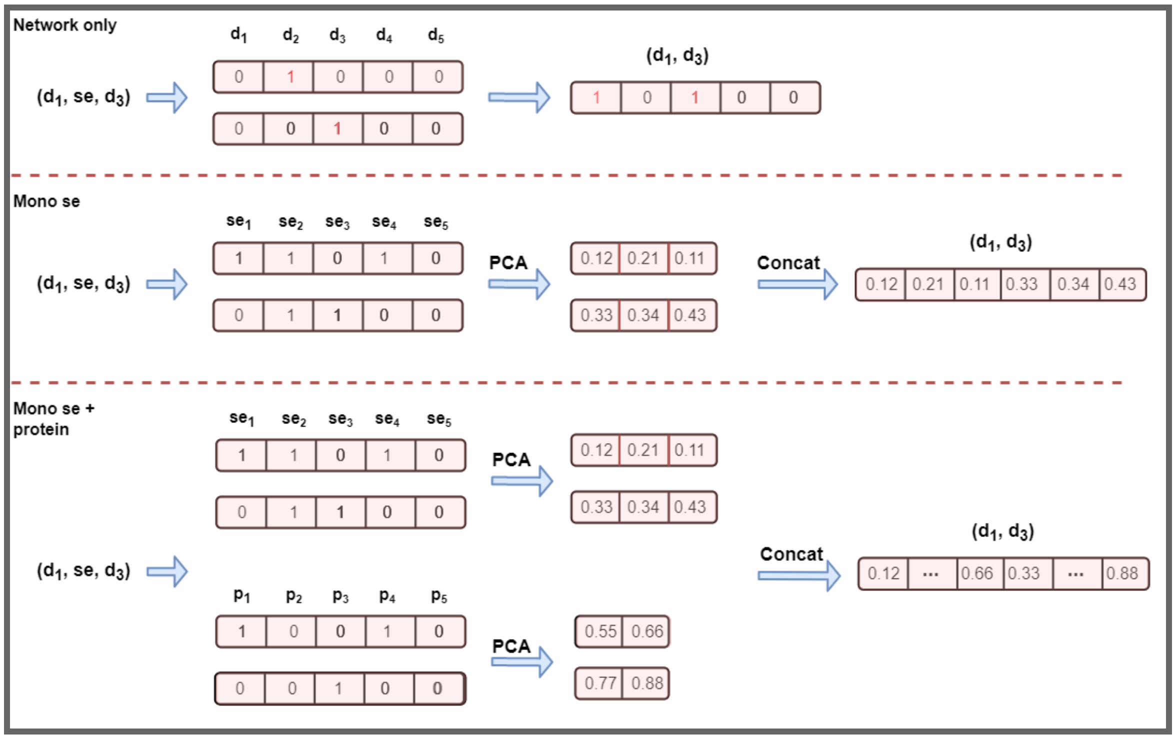


Figure 4.1 Construction of drug pairs representations in each model. Afterwards, these representations are passed to n individual classifiers, one for each polypharmacy side-effect to predict the probability of each association.

4.1.2 Drug mono side-effects as features

In this model, each drug is represented by a multi-hot-encoded feature vector of the side-effects it is known to cause when taken on its own. Due to the great dimensionality of the vectors (around 10K) we had to apply PCA on the input. The PCA representations of each drug were concatenated to represent the drug pair.

4.1.3 Include target protein information

This model is like an improvement on the previous one, as we include protein related information in the feature vectors in addition to the mono side-effects. Once again we applied PCA to the protein related features (independently) and concatenated them with mono side-effect features.

This model was supposedly identical to a baseline model from the original paper[7]. However, the performance of the model was not close to the one on the original study, as we will see in the results section. We tried to identify the root of this divergence, by adjusting a number of

[7] mentioned as "concatenated features" model

parameters, especially aspects that were not clearly specified in the paper. We tried different splits in the dataset and different ways to implement the negative sampling, even different ways of transforming our data through pca. Another consideration is whether the classifiers are independent or if there is some transfer of knowledge among them. Our tries were not enough to spot the root for this divergence. We tried to come in touch with the authors, more than once, for extra specifications and clarifications, without, unfortunately, getting any response.

4.1.4 Confine to drugs with protein interactions

On (Malone et al., 2018) it is suggested that limiting the dataset to contain only drugs that are interacting with proteins will make protein related information more relevant and dense and hence improve the performance of the model. Upon this ground we trained the previous model on this limited version of the dataset, to see if it actually yields better results.

4.2 Experiments on the rare side-effects subgraph

As we saw in the third chapter there is some interest in limiting our research in the rare side-effects graph. This way we gain more insight into side-effects that are hard to identify clinically, through making rare side-effect information more dense in the data.

4.2.1 Graph embedding models

Our dataset can be seen as a multi-relational, heterogeneous graph or even a pretty particular knowledge graph, where our entities are proteins and drugs and the relations are mainly side-effects that can arise from the combinations of drugs. So we wanted to experiment with some well-known models for graph embeddings.

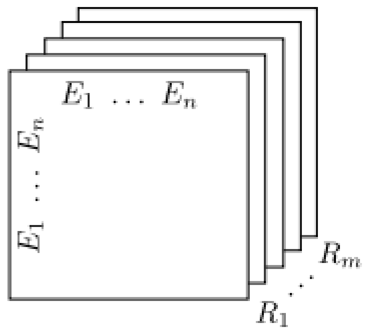


Figure 4.2 Generalizing the notion of the adjacency matrix of a graph, a multi-relational graph can be represented as a three dimensional tensor. Every slice of the tensor is the adjacency matrix among entities E for the specific relation R (Nickel et al., 2011)

Rescal (Nickel et al., 2011) is a tensor factorization method. Tensor factorization models treat graphs as three-way vectors that they decompose into a combination of low-dimensional vectors that are eventually used as feature vectors for the entities and relations. (see Figure 4.2) The learned embeddings should be able to generalize, and associate high scores to unseen valid triplets in the graph.

In multi-relational settings, beside direct correlations among entities or relations there can also exist across interconnections of different entities and relations. Detecting these types of correlations can be of great importance depending on the task. Rescal employs the following rank-r factorization where each slice of the multi-relational graph is factorized as

$$\mathcal{X}_k \approx AR_kA^T, \text{ for } k = 1, \dots, m$$

where A is an *n x r* matrix that contains the latent-component representation of the entities in the domain and R_k is an asymmetric r x r matrix that models the interactions of the latent components in the k-th predicate.

Something to keep in mind and differentiates this method from other factorization methods is that it has the same representation for the entities whether they appear as the head or the tail of a triplet.

In **TransE** (Bordes et al., 2013), relationships are represented (for the first time) as translations in the embedding space, requiring that If a triplet (h, r, t) holds, then adding the embedding of the relation to the embedding of the head should yield a vector that lies close to the vector of the tail. While the main motivation behind this model was capturing hierarchical relationships, inversion and composition patterns in the form of translations, and despite its simplicity, this model proved to be powerful on most kinds of relationships. One-to-many and many-to-one as well as symmetric and transitive relations are the weak spot of the model.

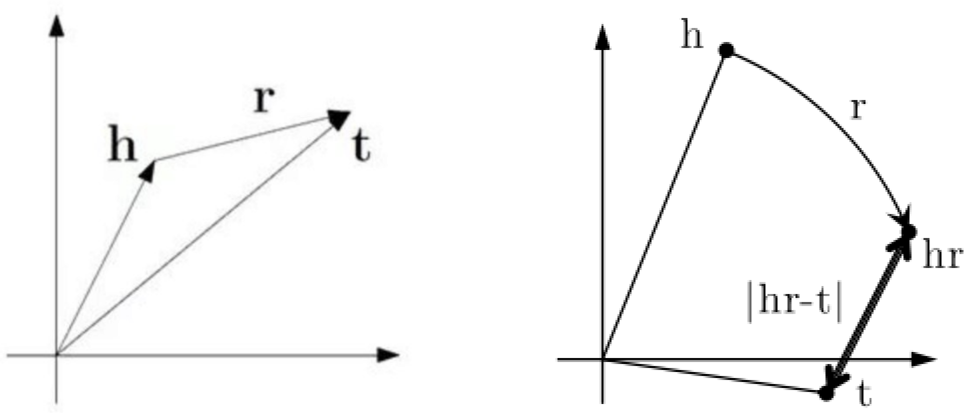


Figure 4.3 (left)Visual representation of the idea behind TransE. Adding head and relation vectors must yield a vector close to the vector of the tail of the triplet (right) Visual representation of the RotatE concept. Relations are represented as rotation in the complex panel. When performed on the head should yield a vector close to the tail. The distance of the two vectors is the quantity the model tries to minimize.

The optimization is carried out by stochastic gradient descent in mini batches. The embeddings are randomly initialized - the embedding vector is the same whether the entity appears as the head or as the tail of a triplet. A small set of triplets is sampled from the training set, and serve as the training triplets of the minibatch. For each such triplet, we then sample a single corrupted triplet, with either the head or tail replaced by a random entity. The parameters are then updated by taking a gradient step with constant learning rate.

RotatE model (Sun et al., 2019) defines each relation as a rotation from the source entity to the target entity in the complex vector space. In order to understand the reason behind RotatE as an improvement to TransE we should first briefly examine some types of relations. With the words of the authors "..some relations are symmetric (e.g., marriage) while others are antisymmetric (e.g., filiation); some relations are the inverse of other relations (e.g., hypernym and hyponym); and some relations may be composed by others (e.g., my mother's husband is my father)".

Modeling these types of relations is crucial in predicting missing links.RotatE was proposed as the first model to infer all the above patterns. We have seen for example that symmetric and antisymmetric relations are the weak spots of TransE.

Given a triplet (h, r, t), RotatE expects that the embeddings will satisfy

$$\mathbf{t} = \mathbf{h} \circ \mathbf{r}, \quad \text{where } |r_i| = 1$$

where ◦ denotes the Hadamard (element-wise) product (see Figure 4.3).

RGCN (Schlichtkrull et al., 2018) comes as a generalization of GCN (Kipf & Welling, 2016), to be able to handle multi-relational and heterogeneous graphs. GCN is in turn a generalization of convolutional neural networks in the notion that the data can have a variable number of neighbors instead of being fixed on a grid like in the case of an image. The basic steps to update the embedding of a node in a message passing network is to collect the information from the neighbors, aggregate it and pass it through a neural network layer. The formula for RGCN is the following

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

where σ is the activation function, N is the neighbors for the node i in the relation r and c is a regularization term, for example the number of neighbors in the case we chose average as the aggregation function.

Compared to GCN, we have one weight matrix for each relation, so that we don't aggregate "apples with oranges". This of course multiplies the number of trainable parameters, something that may be a concern especially for graphs with a large number of relations like in our dataset. The authors of RGCN propose ways to address this issue. The Decagon encoder is built based on this model.

4.2.2 AnyBURL

AnyBURL is a bottom-up, rule based model proposed by (Meilicke et al., 2020) as an alternative to the representational techniques which are, so far, dominant in the areas of link prediction and graph completion as opposed to symbolic ones. A bottom-up approach is based on the idea that a data point is a representation of a specific rule that can be generalized to capture a comprehensive subset of all data points. The rule does not need to be certain, only to include some positives and usually also negative examples
Along with competitive performance to the current state of the art, it offers low computational resources, fast training and evaluation time and it also provides, to an extent, explainable results coming from the rules mined.

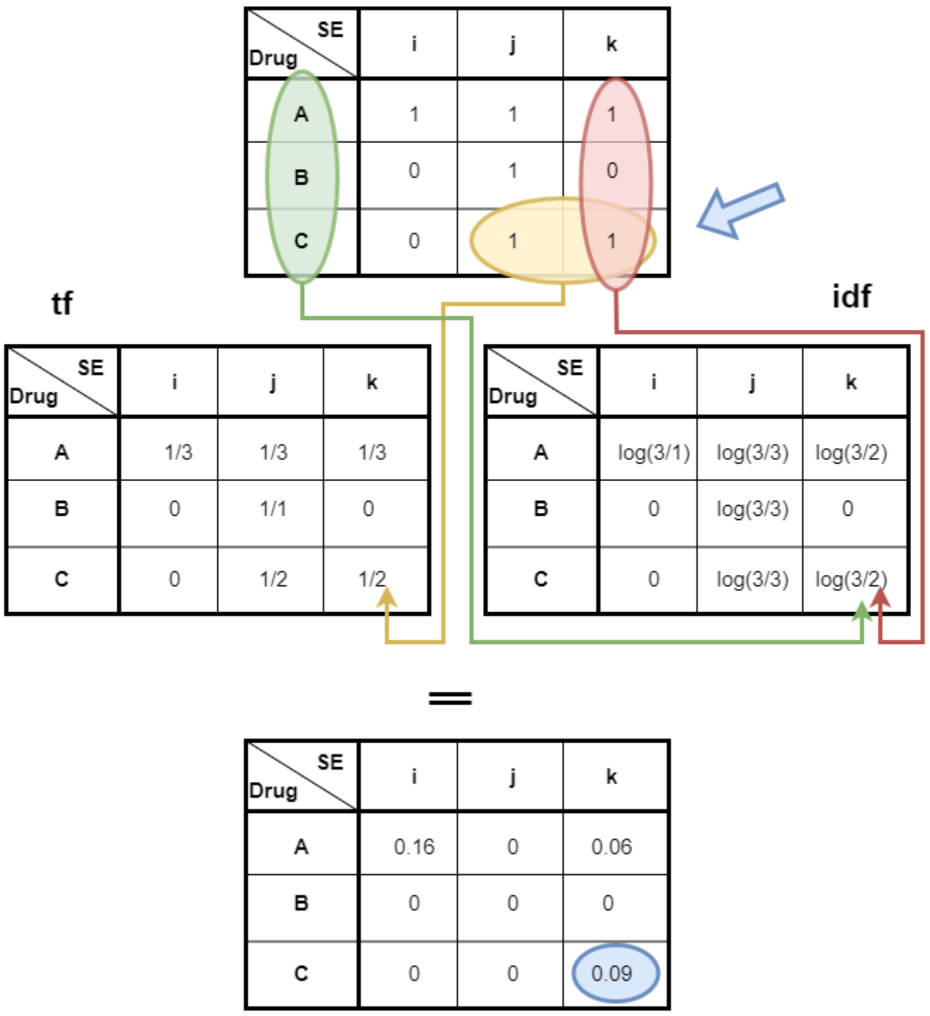
4.3 Mono side-effect feature weighting

After some time of exploration and familiarizing with the dataset there was one particular aspect of it that striked us. There is no quantitative measure of whether a side-effect is severe or mild and how frequent it is as well. Both severity and frequency are major aspects for assessing a side-effect impact (Villars et al., 2007). In both side-effect datasets we examine, the existence of a side-effect is depicted as a binary yes or no, although, especially the mono side-effects, datasets come partly from drug label text that normally contains this kind of information. This leaves plenty of room for improvement in the quality of the data fed on the model.

Regarding the polypharmacy side-effects, having a model predict the existence of each one separately is already a major improvement over the previous approaches that were basically limited on the severity of the side-effects (without specifying what they could be) from a drug combination. However the mono side-effects are used as complementary features so we thought that it could be interesting to find a way to scale them, to weigh the importance of each one.

If we recall the stats from the exploration of the datasets, there were a couple of thousand side-effects that were only associated with a single drug, and a few more thousand, associated with a handful of drugs. We believe that such extremely rare features are not beneficial for the training of a model and can only contribute to overfitting. In a similar concept there are few side-effects associated with more than a third of the drugs. These could also act as noise and deteriorate the training of a model.

Taking inspiration from the NLP area, we thought that a tf - idf like calibration (Luhn, 1957) (Jones, 1972) would make perfect sense in this context. We just need to think of drugs as documents and side-effects as words and this concept will become apparent. Common side-effects that appear in a large number of drugs should have lesser importance (inverse document frequency) than side-effects that are bound to just a few cases. And also, side-effects in drugs with few associated side-effects in total are more representative of the drug than they would be in a drug with a lot of side-effects (term frequency). A visual representation of how this concept was implemented is presented in Figure 4.4.



Drug \ SE	i	j	k
A	1	1	1
B	0	1	0
C	0	1	1

Drug \ SE	i	j	k
A	1/3	1/3	1/3
B	0	1/1	0
C	0	1/2	1/2

Drug \ SE	i	j	k
A	log(3/1)	log(3/3)	log(3/2)
B	0	log(3/3)	0
C	0	log(3/3)	log(3/2)

=

Drug \ SE	i	j	k
A	0.16	0	0.06
B	0	0	0
C	0	0	0.09

Figure 4.4 A visual explanation of the tf-idf normalization process. The tf factor normalizes the importance of each side-effect for each drug (horizontal normalization). Unlike words in documents, side-effects can only appear once in our dataset and hence the relative importance is just inverse to the total side-effects associated with each drug. Idf factor normalizes vertically, with side-effects associated with few drugs being important and on the other side, side-effects associated with each and every drug bear no power to discriminate among them and are zeroed by the process.

Let’s explain a bit further. Term frequency as an absolute number cannot really apply in our case since every side-effect can only appear once in our dataset. However, relative term frequency can. The total number of side-effects each drug is associated with, can act as a regularization parameter. In a sense, a side-effect, let’s say nausea, in a drug associated with a total of five side-effects is more representative of the drug compared to nausea as a side-effect of a drug associated with one hundred side-effects. Let’s give an example from the area of text in order to be more clear. Imagine a document containing only the phrase “King is dead”. And now think of another long length document referring to matrices, in which the word “king” also appears once (king size). Obviously ‘king’ is more important, more representative of the first document, compared to the second one.

Inverse document frequency is easier to understand. Common words like “the”, “and”, “is” are less useful in search queries. They are present in more or less every document and have little, if any, representative importance about each document. In our case, these common words would be common side-effects like “pain” or “nausea”. As opposed to more specific terms/side-effects like “hair loss” or “increased blood viscosity”.

5. Experimental Setup

5.1 Initial idea

Our initial idea for this project was to achieve a reimplementation of the Decagon model in Pytorch framework (the original is in TensorFlow) and after that, further experiment in two basic axes. The first was to replace components of the original model with different ones, for example, try different types of neural networks as encoders and decoders in the model. The second idea was to incorporate extra information in the model so that it could capture more aspects of the problem in the form of extra features for the graph entities on top of the existing or/and add extra graph networks on top of the original dataset. Some examples of extra features that were considered and searched upon in the literature were textual information about the drugs and molecular structure of drugs and proteins (Vilar et al., 2012) (Li et al., 2017). An example of an additional network that was considered was drug - disease network as in (Xuan et al., 2019).

Although not clearly stated in the original paper, Decagon is a really heavy model to train. According to (Xu et al., 2020) Decagon requires more than 28GB peak GPU memory usage and approximately 2.5 hours of training per epoch. These heavy requirements in conjunction with the difficulty in reimplementing such a complex model made us abandon the initial idea as not feasible in the context of a postgraduate thesis. However, this initial phase of the project was not a waste of time as parts of the code were later reused and the familiarity that was earned through this process on libraries and the domain in general turned out really helpful later on.

5.2 Reproduce TIP results

TIP (Xu et al., 2020) is an alternative model proposed as a lighter and more effective alternative to Decagon. With this model they achieve a 7 percent extra accuracy in terms of AUPRC and an astonishing 83x faster training and evaluation performance. We tried to reproduce the results of TIP, hoping that later we could experiment further on improving the model. Although the code is open and available on github, we had difficulties running it and reproducing the results. We tried various python environments and set ups. We also tried to dive deep into the code and overcome what seemed to be a bug caused by version conflicts in python packages by ourselves, without success.We also asked for the help of the authors and while eventually got an answer from them, this was not able to resolve the problem. [8]

[8] see https://github.com/NYXFLOWER/TIP/issues/15

5.3 Baseline approaches

In these experiments we use simple baseline models in the full polypharmacy dataset consisting of triplets in the form *(d,se,d)* and we treat the link prediction task the way the authors on the original do for comparability reasons, as described in chapter 4. Drugs are represented as feature vectors that are concatenated to form the representation of each pair and one independent classifier is trained for each polypharmacy side-effect.

We split the dataset into 80% training and 20% test (approximately 50K training and 12.5K test drug pairs). We stratified our samples so that each label appears in equal percentages in train and test sets. To train and evaluate the models we created negative samples. To achieve this, for every triplet (h, r, t) we created a corrupted one (h, r, t') where the new tail was chosen uniformly at random among all the entities, making sure that the negative triplet does not already exist in our dataset as positive.

We chose logistic regression as our base classifier due to its simplicity, performance efficiency and the lack of crucial hyperparameters that make it ideal for a baseline. Later we tried a number of different simple classifiers like naive bayes and decision trees.

For each model what is different is the information reflected and the dimension of the feature vectors.

For the first model we have an embedding of length 645 (number of drugs) with two aces indicating the two drugs of the pair. For the second model we use PCA representations of the mono side effects of each drug that we eventually concatenate in a vector with size 600. With an initial length of 10,084 side effects, we decided on an embedding of size 300 that keeps approximately 80% of variance. For the third model we additionally construct a PCA representation of the protein targets of each drug. With 50 features we are able to capture 95% of the variance. We concatenate these vectors for a total embedding with size 700.

5.4 Knowledge graph embeddings on rare side-effects subgraph

In these experiments we focus on the rarest side-effects of the dataset. After creating some subsets of the original combo dataset by setting different thresholds on side-effects to keep, we examined the graph statistics. We wanted to experiment in a smaller graph that at the same time retained the characteristics of the whole. Limited to the rarest 50 side-effects we can keep more than 80% of the drugs but there are a lot of nodes with one or few edges, unlike the initial graph. By increasing the threshold to 300 rarest side-effects, the graph has ten times more edges (while 10 times less than the full dataset) includes more than 95% of the drugs and the pairs of drugs with only one side-effect are sparse, meaning that the graph is strongly multirelational again.

We decided this was a balanced trade off among similarity to the original graph and size reduction, so we only used the mini graph for initial testing and tuning and later proceeded to experiment on the bigger subgraph of 300 hundred rarest side-effects on which we report our results.

On this trimmed down dataset we trained a number graph embedding models and evaluated their performance in the typical link prediction task, where for every pair (h,r) we want our model to assign high probability to the correct tail. We tried Rescal as an example of a tensor decomposition model, TransE and and RotatE as translational and roto-translational models, RGCN as a deep learning model and AnyBURL a rule based model.

5.4.1 Features reduction and normalization

Regarding the RGCN model we trained and evaluated three variations of it. Taking advantage of its ability to handle additional features we created a model with randomly initialized vectors, one with vectors capturing the information about individual side-effects of each drug, and one model where this information has been normalized in a tf-idf way, as described in the fourth chapter.

By comparing the performance of the model under these three variations we wanted to test whether the additional info coming from individual side-effects adds value to the model (compared to the random initialization of the features) and whether normalizing the features have any further positive effect.

5.5 Technical aspects and hyperparameter optimization

Due to the concept of this project, which was rather an exploratory and qualitative research of different methodologies in a particular dataset and not a proposal for a new methodology, hyperparameter tuning in order to achieve the best possible result was not our main focus. For most of the experiments and models, only a basic, manual optimization took place on the most basic parameters of each model, most of the time, the embedding size and the number of layers.

After some initial, manual experimentation in order to get an insight on the learning behavior, for the graph embedding models on the rare subgraph, we used an early stopper with a frequency of 5 epochs, patience 3[9] and threshold 0.002 in the hits@50 metric. For AnyBURL we manually set the duration it would require to train in 1000 seconds.

All the experiments were performed in a personal laptop running Windows 10 on 8GB RAM. Python is the main language used across the project with some scripts (mainly for

[9] a model achieving its best performance on the 100th epoch would stop on epoch 115

preprocessing purposes)written in R. For the graph embedding models we used the implementations provided by Pykeen.

5.6 Performance Metrics

Decagon authors use three metrics for reporting on the performance of their model. AUROC, AUPRC and AP@50. For reasons of comparability we also use the same metrics on the full dataset experiments[10].

These metrics have different strengths and weaknesses allowing them to identify different aspects of the performance of a model. AUROC and AUPRC have many common characteristics. The thing that distinguishes them the most is the ability of AUPRC to handle imbalanced datasets, with a rare positive class and a majority of less important negative samples. In such settings AUROC tends to flatter the model, while AUPRC is a more meaningful metric based on the concepts of precision and recall.

Average precision at k is a bit different as a concept as it is a measure of how accurate the model is in its top guesses. So we take the top k guesses of our model ordered (those with possibilities closest to one) and match them with their actual value. The more true positives and the higher they are in our list the biggest is the score. The exact formula is the following

$$AP@k = \frac{1}{N(k)} \sum_{i=1}^{k} \frac{TPseen(i)}{i}$$

where N(k) is the min(k, actual positives)[11].

In our case, since every side-effect appears at least 500 times and we keep 10% as a test, every side-effect appears at least 50 times in the test set. Possibly that was the reason decagon authors chose 50 as k in the first place.

For the experiments on the rare side-effects subgraph we use hits@k a typical metric for the link prediction task approached this way. Given a pair of entity-relation, the hits@k metric evaluates the ability of the model to assign high enough probability to the correct missing entity so that it is listed among the first k most probable results.

Generally, when rank based metrics are used and like in our case a pair of head-relation can have multiple correct tails, there must be some extra consideration. For example let us consider

[10] For the different tasks dealt with in our experiments, please refer to section 4.

[11] For an insightful explanation look here https://medium.com/@misty.mok/how-mean-average-precision-at-k-map-k-can-be-more-useful-than-other-evaluation-metrics-6881e0ee21a9

the case when a (h,r) pair has five alternative true tails. Then even if the model perfectly assigns the best probabilities to these five entities, the fifth will have a lower rank. In the extreme where we use a metric like hits@3 we evaluate the model as failed for the specific prediction, something not fair at all. As proposed in (Bordes et al., 2013) and implemented by Pykeen library we exclude other true triplets from the evaluation[12].

5.7 Negative sampling for training

The dataset that runs through this project is a multi-relational graph that contains only the polypharmacy side-effects we are aware of and does not contain any information about combinations of drugs not causing a specific side-effect. All the components of our system use supervised machine learning methods to fit classifiers, which can compute the probability of a triple being true. So we need a strategy to generate negative samples for our classifiers. There are different methods one can consider.

Skipping OWA and CWA (open and closed world assumptions) that are clearly poor choices to train a model (Nickel et al., 2016) , we followed sLCWA (stochastic local closed world assumption) to train our model. For symmetrical relations like the ones in our case, this approach converges to LCWA (Dong et al., 2014). Refer to Figure 5.1 for an insightful explanation.

In the original paper, although they don't name it explicitly, they also use the sLCWA method and create one negative sample for each positive edge randomly chosen from all the possible triplets by replacing the tail with another entity. We followed a similar procedure where we replaced the tail of each triplet with a different entity, uniformly at random, making sure that the produced triplet is not corresponding to any existing positive one.

[12] For more specifics on the pykeen implementations of this feature check https://pykeen.readthedocs.io/en/stable/tutorial/understanding_evaluation.html

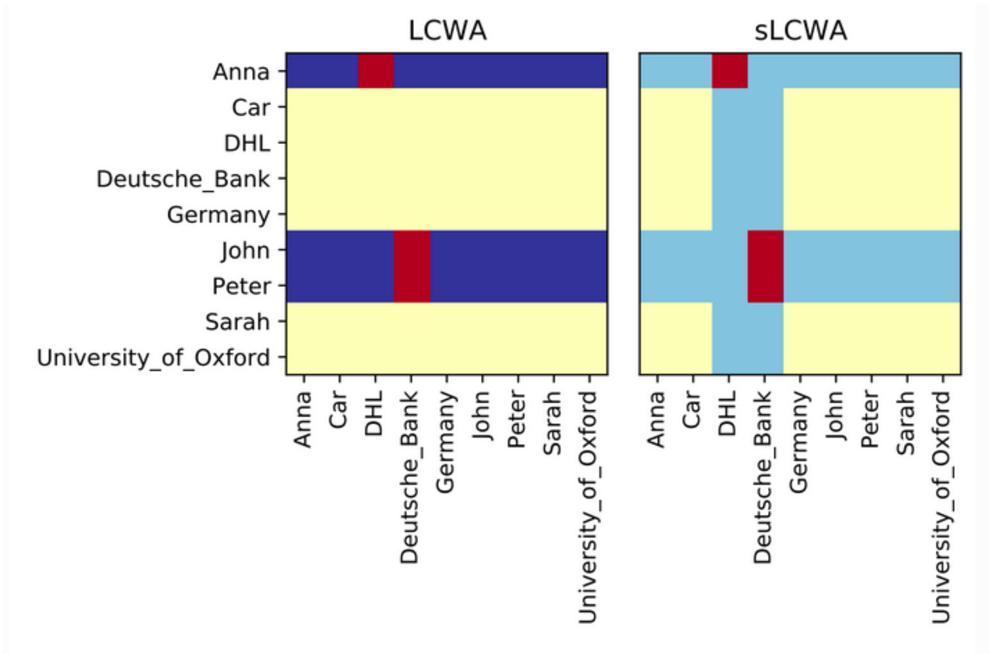


Figure 5.1 An illustrative example for the difference among the local closed world and stochastic local closed world assumption when it comes to picking negative samples for a dataset. In case of symmetrical relations both approaches converge.(Ali et al., 2021)

6. Results

6.1 Results on full dataset

In table 6.1 we present the results of the baseline models tried on the whole graph. The models were tested with a number of simple base classifiers and here we report the best performance. The values are average performance for all the different relations.

The main take away from this series of experiments is that the best performing model is the one that uses solely network structure information, without including any extra featural information, a clear indication that the structure of the polypharmacy graph is of more importance in this problem compared to the featural information coming from individual drugs' side-effects and protein related information. Perhaps the benefit from the extra information in the form of features is not enough to counterbalance the added complexity that it brings into the model.

Limiting the dataset on drugs that interact with proteins was a way of increasing the relative importance of the protein related information as proposed by (Malone et al., 2018). However, we did not observe a noticeable difference in our model.

model	AUROC	AUPRC	AP@50	# features
Structure	**0.884**	**0.861**	**0.838**	645
Side-effects	0.861	0.847	0.836	600
Side-effects + protein info	0.857	0.839	0.831	700
Side-effects + protein info Limit to protein targeting drugs	0.859	0.836	0.832	700
Decagon * (from original work)	<u>0.872</u>	<u>0.832</u>	<u>0.803</u>	--
Concatenated features * (from original work)	<u>0.793</u>	<u>0.764</u>	<u>0.712</u>	--

Table 6.1 Baseline results on the full dataset. Decagon performance is included for completeness, but should not be considered comparable to the rest of the results.

In table 6.1, along with our results we present the performance of the Decagon and one of its baselines that it is equivalent with our third baseline, the one containing featural information about side-effects and proteins. Seemingly, our simple baseline models perform better than a complex neural network, let alone the respective baseline. However, we should not consider these results comparable by default. Evaluation of link prediction can be quite tricky, especially the way the decagon's authors dealt with this task. As mentioned in the literature and discussed in the methodology chapter, even small differentiation in the specifics of the implementation, splits and negative sampling technique may have a disproportionate effect on the results. Failing to come in touch with the authors in order to clarify these specifics and reproduce their results reliably, we can only assume that the divergence is mainly due to the factors we just mentioned.[13]

6.2 Results on rare side-effects subgraph

In the following experiments we limited to the subgraph containing only the rarest side-effects but also changed the evaluation to be the typical evaluation of link prediction in a multi-relational setting as explained in the fourth chapter.

On Table 6.2 we present the results on the rare side-effects subgraph. Rescal was two orders of magnitude slower than TransE, the fastest of the graph embedding models, and also had inferior performance. RGCN performance was also low in comparison to the geometrical models and approximately an order of magnitude slower to train per epoch than TransE. TransE and RotatE both achieved high performance with RotatE performing slightly better but TransE being significantly faster. AnyBURL seems to yield the best performance of all and also in the fastest time[14].

Performance in the two different metrics captures different aspects of the model. Hits@50 is a quite forgiving metric indicative of the general performance of a model while hits@10 is more meaningful in an actual setting where we want to have reliable predictions[15]. We can see AnyBURL outperform the other models in both metrics but significantly more in the hits@10 making its superiority more evident.

With the experiments we presented in the previous section we saw that structural information was the most important element in predicting polypharmacy side-effects. Using RGCN, a much more elaborate model, we will be able to go further and identify whether extra features add any value to the models and whether improving the quality of this data will make any difference.

Comparing the performance of RGCN under the three different settings we can actually see an improvement on the performance of the model when featural information is included, meaning

[13] We focus on the negative sampling method since it is essentially a way of drastically subsampling the dataset and hence can have a major effect on the evaluation of a model (Yang et al., 2015).

[14] The amount of time to spend on training and evaluating is a hyperparameter for this model. 1000 seconds is a typical value that the authors also use in their paper

[15] We remind here the discussion in 5.6 section for the way we excluded other positive triplets from the candidate's set so that the correct candidate can always be ranked first no matter if a head-relation pair has many correct tails.

* Decagon and baseline performance is included for completeness, but should not be considered comparable to the rest of the results.

that the individual side-effect information is actually adding value on the model. On the other hand, trying to improve the quality of this extra information by normalizing the features does not seem to add any extra value. At least that seems to be the case for the way we implemented it. Perhaps due to the way message passing models work,with consecutive aggregations of neighboring vectors, these small adjustments on the initial vectors were not that important to show.

Once more we need to to point out that these results are not comparable to the ones on the full graph, and not merely from the fact that they are experiments on a different dataset. The actual task is slightly different in each case as described in the fourth chapter.

Algorithm	**Hits@50**	**Hits@10**	**Epochs**	**Embedding size**
TransE	0.689	0.245	140	300
Rescal	0.599	0.204	60	300
RotatE	0.708	0.291	60	500
AnyBURL	**0.737**	**0.427**	1000 (sec)	--
RGCN	0.621	0.226	120	500
RGCN (mono se features)	0.628	0.232	120	500
RGCN (normalized features)	0.626	0.229	120	500
Random baseline	0.079	0.016	--	--
Dummy predicting top 50 drugs	0.354	0.096	--	--

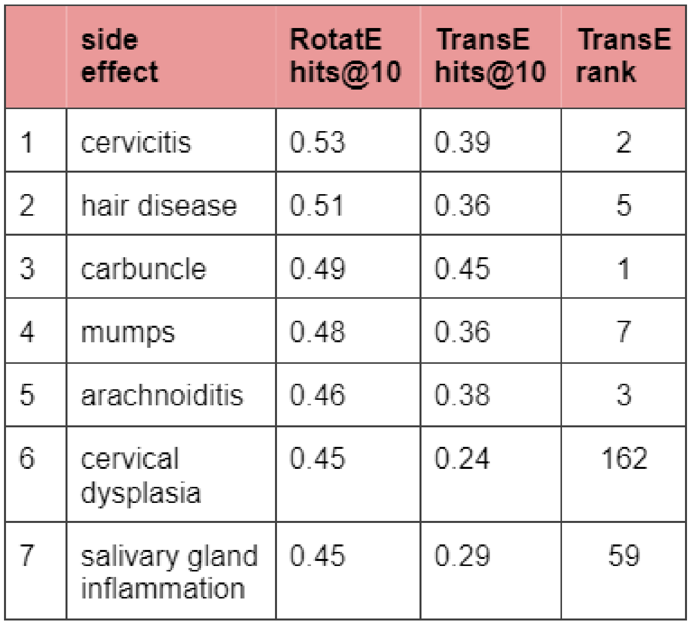
Table 6.2 Results on the subgraph containing 300 rarest side-effects. Geometrical models seem to perform best with RotatE being the best, followed closely from TransE.

6.3 Discussion

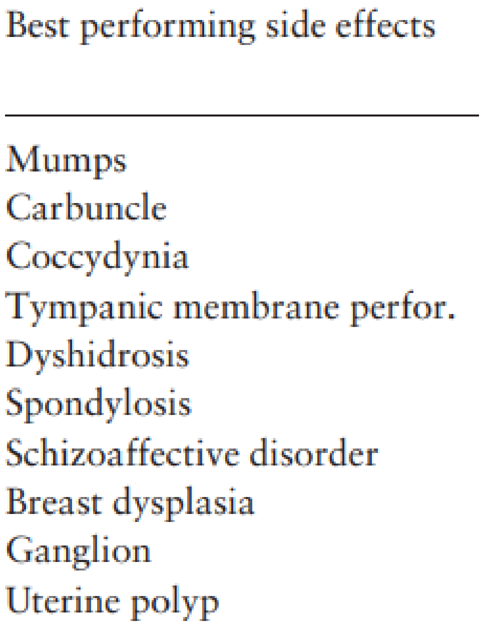
6.3.1 Performance per side-effect

We have already seen in the third chapter that some of the most common side-effects are among the worst performers in Decagon. Conversely, all the best performers are included in our rare side-effect subgraph. Unfortunately we are not able to compare our results with the ones

from Decagon in order to decide on whether pruning common side-effects from the graph helped to predict the rare ones with better accuracy.
However as a sanity check we present on Table 6.3 the best performers for TransE and RotatE in terms of hits@10. We can see many of the best performers in Decagon to also make the top ranks here both in RotatE and TransE.



	side effect	RotatE hits@10	TransE hits@10	TransE rank
1	cervicitis	0.53	0.39	2
2	hair disease	0.51	0.36	5
3	carbuncle	0.49	0.45	1
4	mumps	0.48	0.36	7
5	arachnoiditis	0.46	0.38	3
6	cervical dysplasia	0.45	0.24	162
7	salivary gland inflammation	0.45	0.29	59



Best performing side effects
Mumps
Carbuncle
Coccydynia
Tympanic membrane perfor.
Dyshidrosis
Spondylosis
Schizoaffective disorder
Breast dysplasia
Ganglion
Uterine polyp

Table 6.3 Side effects with the best performance in terms of hits@10 for RotatE compared to the respective ranking in TransE.(left) Best performing side-effects in Decagon (right). Many of the best performers overlap in all three models.

6.3.2 Patterns from AnyBURL

Apart from general performance in terms of hits@k and other ranking metrics, AnyBURL can identify specific patterns that appear in the data and compliment the results with a note of explainability as to why for example a side-effect is to be expected among a pair of drugs.. Some of the most recurring patterns are the ones in Figure 6.1. Obviously these patterns do not express certainty. They are accompanied with their respective head, body and head over body ratio, where the body is the absolute frequency of the left side in the dataset and head is the number of times this rule holds true.

$$se_1(X, Y) \Rightarrow se_2(X, Y)$$

$$se_1(A, X) \Rightarrow se_2(A, X)$$

$$se_1(A, X) \Rightarrow se_1(B, X)$$

$$se_1(A, X) \Rightarrow se_2(B, X)$$

$$se(A, X) \ \& \ se(A,Y) \Rightarrow se(X, Y)$$

$$se_1(B, A) \ \& \ se_2(A, X) \ \& \ se_3(B, Y) \Rightarrow se_4(X, Y)$$

Figure 6.1 The most recurring patterns identified by AnyBURL. The model is able to capture simple correlation patterns as well as more complex patterns.

If we try to express these patterns in words, the first rule is a simple correlation among side-effects. The second one is denoting a correlation among two side-effects in the presence of a drug. The third rule implies that two drugs have similar behavior when it comes to a specific side-effect.The fourth rule is similar to the previous one, only now whenever one drug causes one side-effect the other causes a different one. The fifth rule resembles the transitive property where if two drugs are associated with a side-effect when taken with a specific drug then they will be associated with the same side-effect when taken together. As you may imagine, the sixth pattern is a challenge to put in words and we will not attempt it. In reality we included this specific pattern as a placeholder for the more complex patterns that the model is able to identify.

Perhaps it is easier to understand the above rules if we name drug A to be paracetamol, drug B omeprazole, se_1 will be nausea and se_2 will be emesis.
In that case, the second rule denotes that if you take paracetamol and another drug and get nausea, you will probably also get emesis.
The third rule implies that if you used paracetamol with a drug and got nausea, you will probably get nausea if you take omeprazole with that same drug as well. And so on for the rest of them.

7. Conclusion

This project revolves around the problem of predicting polypharmacy side-effects based on the Decagon dataset. The dataset contains information about individual drugs, as well as polypharmacy, side-effects and protein related interactions of drugs and proteins.

The initial idea was to reimplement the Decagon model and experiment with graph neural networks in the specific issue as well as incorporate extra information on the dataset. However, difficulties on reproducibility, literature review and implementation limitations reformed the scope of the project. The above difficulties in conjunction with our exploratory analysis led us to move in two basic directions.

On one hand, we tried a number of baseline models on the full dataset. In these models every pair of drugs is represented by a feature vector that, for each model, utilizes different sources of information. We report that the model that solely contains the structural information achieves the best performance implying that extra information in the form of features regarding mono pharmacy side-effects and protein interaction information is of secondary importance, while network information regarding polypharmacy side-effects is the main driver for the success in the specific task.

On the other hand, we focused on a part of the graph, containing only the rarest polypharmacy side-effects. The main reasons for this choice were three. First, the special interest rare-side-effects carry due to their sparsity. Secondly, the information included in the dataset is not well suited for common side-effects with no molecular basis. As reported in the original paper many of the more common side-effects are amongst the worst performers. And thirdly, simply by their number, common-side-effects dominate the dataset acting like a noise for the other relations of the graph that end up really sparse.

On this subgraph we tried some representative models for graph embedding but also a rule based model. Rule based model AnyBURL achieved the best performance among them followed by RotatE and TransE. Aside from achieving the best performance, AnyBURL was also really fast to train but also, since it is rule based, its results come with the extra benefit of explainability. As presented in the results section, there are useful patterns that arise from the data and could be really useful for a medical oriented researcher helping in gaining more insight and building trust on the results as opposed to a mere number, suggesting probability for an association, without any extra explanation.

On the same subgraph we also trained three versions of RGCN. One with randomly initialized vectors, one with mono pharmacy side-effects as features and one where the features had also been normalized in a tf-idf way in an attempt to increase the quality of the data fed on the model. We report an improvement when side-effects features were taken into account, denoting that drugs' individual side effects bare useful information for predicting polypharmacy side-effects. Normalizing the vectors, on the other hand, does not seem to add any value.

7.1 Future work

There are things that we would like to try on another iteration of this project. We note these ideas here as possible future work.

Following what we tried with the weighting of each side-effect, we believe that an improved dataset that would quantify the frequency and the severity of each mono and/or poly side-effects could be a possible way to improve the predictive power and the performance of the candidate models.

From the literature and the small experimentation with AnyBURL we saw that rule based models lift off a lot of weight and complexity, thus training much lighter and competitive models. We would like to further explore the potential of these models. Rule based models are also interpretable, which is an important virtue for a model especially in this particular area of interest.

Meta classifiers would also be an interesting idea. For example we could use the predictions of our baseline models, that trained independent classifiers for each side-effect, as input to a meta classifier, so effectively, we would transfer knowledge from all the different side-effects and see how these models perform.

Having trained different models in the same splits and evaluated per relation it would be a good opportunity to take a step further and explore the reason for different models capturing some side-effects better than other models do. Perhaps with the contribution of a rule based model like AnyBURL we could gain some explainability over this issue and eventually some insight into the dataset but also the models themselves.

Finally, adding more data is always something tempting, so we could combine more information in the graph by including extra datasets, like drug - disease data or chemical fingerprints of the drugs.

References

Ali, M., Berrendorf, M., Hoyt, C., Vermue, L., Galkin, M., Sharifzadeh, S., Fischer, A., Tresp, V., & Lehmann, J. (2021). Bringing Light Into the Dark: A Large-scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework. https://doi.org/10.48550/arXiv.2006.13365

Balazevic, I., Allen, C., & Hospedales, T. M. (2019). TuckER: Tensor Factorization for Knowledge Graph Completion. https://doi.org/10.18653/v1/D19-1522

Bansal, M., Karan, C., & Yang, J. (2014). A community computational challenge to predict the activity of pairs of compounds. *Nat Biotechnol*.

Bordes, A., Usunier, N., & Garcia-Duran, A. (2013). Translating Embeddings for Modeling Multi-relational Data.

Cao, S., Lu, W., & Xu, Q. (2015). GraRep: Learning Graph Representations with Global Structural Information. *CIKM '15: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. https://doi.org/10.1145/2806416.2806512

Chatr-Aryamontri, A., Breitkreutz, B., & Oughtred, R. (2015). The BioGRID interaction database: 2015 update. *Nucleic Acids Research*, *43*(D!).

Chen, X., Ren, B., Chen, M., Zhang, L., & Yan, G. (2016). NLLSS: Predicting Synergistic Drug Combinations Based on Semi-supervised Learning. https://doi.org/10.1371/journal.pcbi.1004975

Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D Knowledge Graph Embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v32i1.11573

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., & Zhang, W. (2014). Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge

Fusion. *KDD '14: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. https://doi.org/10.1145/2623330.2623623

Garcia-Duran, A., & Niepert, M. (2017). KBLRN : End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. https://doi.org/10.48550/arXiv.1709.04676

Gilmer, J., Schoenholz, S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry. *Proceedings of the 34th International Conference on Machine Learning*.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939754

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*.

Kantor, E., Rehm, C., Haas, J., Chan, A., & Giovannucci, E. (2015). Trends in Prescription Drug Use Among Adults in the United States From 1999-2012. *JAMA*. doi:10.1001/jama.2015.13766

Kipf, T., & Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. https://doi.org/10.48550/arXiv.1609.02907

Kuhn, M., Letunic, I., Jensen, L., & Bork, P. (2016). The SIDER database of drugs and side effects. *Nucleic Acids Research*, *44*(D1).

Lewis, R., Guha, R., Korcsmaros, T., & Bender, A. (2015). Synergy Maps: exploring compound combinations using network-based visualization. *Journal of Cheminformatics volume*.

Li, X., Xu, Y., Cui, H., Huang, T., Wang, D., Lian, B., Li, W., Qin, G., Chen, L., & Xie, L. (2017). Prediction of synergistic anti-cancer drug combinations based on drug target network and drug induced gene expression profiles. *Artificial Intelligence in Medicine*, *83*. https://doi.org/10.1016/j.artmed.2017.05.008

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*. https://doi.org/10.1002/asi.20591

Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information.

Malone, B., Garcıa-Duran, A., & Niepert, M. (2018). Knowledge Graph Completion to Predict Polypharmacy Side Effects.

Meilicke, C., Chekol, M. W., Fink, M., & Stuckenschmidt, H. (2020). Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. https://doi.org/10.48550/arXiv.2004.04412

Menche, J., Sharma, A., & Kitsak, M. (2015). Uncovering disease-disease relationships through the incomplete interactome. *Science*, *347*(6224).

Nickel, M., Murphy, K., Tresp, V., & Gabrilovich, E. (2016). A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, *104*(1). 10.1109/JPROC.2015.2483592

Nickel, M., Tresp, V., & Kriegel, H. (2011). A Three-Way Model for Collective Learning on Multi-Relational Data.

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric Transitivity Preserving Graph Embedding. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939751

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: online learning of social representations. *KDD '14: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. https://doi.org/10.1145/2623330.2623732

Rolland, T., Tasan, M., & Charloteaux, B. (2014). A Proteome-Scale Map of the Human Interactome Network.

Rossi, A., Firmani, D., Matinata, A., Merialdo, P., & Barbosa, D. (2021). Knowledge Graph Embedding for Link Prediction: A Comparative Analysis.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, *20*(1). DOI: 10.1109/TNN.2008.2005605

Schlichtkrull, M., Kipf, T., Bloem, P., Berg, R., Titov, I., & Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. *The Semantic Web. ESWC*, *10843*. https://doi.org/10.1007/978-3-319-93417-4_38

Schlichtkrull, M., Kipf, T., Bloem, P., Berg, R., Titov, I., & Welling, M. (2018). Modeling Relational Data with Graph Convolutional Networks. https://doi.org/10.1007/978-3-319-93417-4_38

Sun, Z., Deng, Z., Nie, J., & Tang, J. (2019). RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.

Szklarczyk, D., Morris, J., & Cook, H. (2017). The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible.

Szklarczyk, D., Santos, A., & Mering, C. (2016). STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data. *Nucleic Acids Research*, *44*(D1).

Tatonetti, N., Ye, P., & Daneshyou, R. (2012). Data-Driven Prediction of Drug Effects and Interactions.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., & Bouchard, G. (2016). Complex Embeddings for Simple Link Prediction.

Vilar, S., Harpaz, R., Uriarte, E., Santana, L., Rabadan, R., & Friedman, C. (2012). Drug—drug interaction through molecular structure similarity analysis. *Journal of the American Medical Informatics Association*, *19*. https://doi.org/10.1136/amiajnl-2012-000935

Villars, P., Dodd, M., & West, C. (2007). Differences in the Prevalence and Severity of Side Effects Based on Type of Analgesic Prescription in Patients with Chronic Cancer Pain.

Xu, H., Sang, S., & Lu, H. (2020). Tri-graph Information Propagation for Polypharmacy Side Effect Prediction. https://doi.org/10.48550/arXiv.2001.10516

Xuan, P., Cao, Y., Zhang, T., Wang, X., Pan, S., & Shen, T. (2019). Drug repositioning through integration of prior knowledge and projections of drugs and diseases. *Bioinformatics*, *35*. https://doi.org/10.1093/bioinformatics/btz182

Yang, B., Yih, W., He, X., & Deng, L. (2014). Embedding Entities and Relations for Learning and Inference in Knowledge Bases. https://arxiv.org/abs/1412.6575

Yang, Y., Lichtenwalter, R. N., & Chawla, N. V. (2015). Evaluating link prediction methods. *Knowledge and Information Systems*, *45*. https://doi.org/10.1007/s10115-014-0789-0

Zhang, W., Paudel, B., Zhang, W., Bernstein, A., & Chen, H. (2019). Interaction Embeddings for Prediction and Explanation in Knowledge Graphs. *WSDM '19: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. https://doi.org/10.1145/3289600.3291014

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*. https://doi.org/10.1016/j.aiopen.2021.01.001

Zitnik, M., Agrawal, M., & Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, *34*.