Computational Molecular Medicine
TUM School of Computation, Information and Technology
Technical University of Munich

Link Prediction in Biological Knowledge Graphs

Course: Machine Learning for Regulatory Genomics
Project Report – Group C

Hui Cheng, Simon Koch, Johannes Spies, Lisa Schmierer

Examiner:
Prof. Dr. Julien Gagneur

Supervisor:
Emy Yue Hu, Sophie Xhonneux

Submitted:
Munich, 16.07.2023

Contents

1 Introduction and Background

This work tries to improve the understanding of regulatory mechanisms in cells involving long non-coding RNAs (lncRNA). We investigate how lncRNA, messenger RNA (mRNA), and other molecules interact with each other, what functions they serve, what roles they play in gene regulation, and other cellular processes. In order to do so, we perform link prediction in biomedical knowledge graphs. We use a path-based reasoning graph neural network (GNN) building on NBFnet [29] and evaluate it against other state-of-the-art knowledge graph embedding methods, such as TransE and RotatE. The resulting predictions could potentially be used to explain aspects of known regulatory mechanisms, explore unknown ones, and design novel experiments.

1.1 Biological Background

Genetic research distinguishes the transcriptome into exons and introns. Exons are spliced and part of mature RNAs translated to proteins. Whereas exons only make up ~1.5% of the human genome, the majority consists of non-coding regions [10]. One of their main purposes is gene regulation. I.e. they are jointly responsible to regulate gene expression, which determines what proteins are present in different cells and in which quantity.

This work focuses on the regulatory interactions between lncRNA and other molecules. LncRNAs are RNAs longer than 200 nucleotides and are not translated into proteins as the name suggest. Nevertheless, they may contain short open reading frames (ORFs). According to the current state of research, lncRNAs have lower expression levels than mRNA, are more abundant in the nucleus compared to other types of RNA and their sequence is poorly conserved [27, 24].

LncRNAs serve multiple purposes. For example, they can serve as potential biomarkers, they regulate key functions of cells, e.g. proliferation, differentiation, and are crucial for gene expression: They regulate DNA methylation, histone modifications at chromatin level and are involved with transcriptional regulation.

Many of the regulation targets in processes mediated by lncRNA are mRNAs. In contrast to lncRNA, mRNAs are coding. They are transcribed from DNA and serve as the blueprint for proteins. In a process called translation, the ribosomes read the three-base long codons from the mRNA and recruit the corresponding amino acids. The concatenation of these yields a protein [10].

1.2 Graphs

Graphs are mathematical objects that consists of a set of nodes V and a set of edges E; $G = \{V, E\}$. An edge connects two nodes with each other. Edges can be directed or undirected. In contrast to directed edges, undirected edges express symmetric relationships, i.e. relations that hold in both directions.

One important characteristic of graphs is their connectivity. A graph is fully connected if it contains a path between every pair of nodes. It is *sparse* if the number of connections is low relative to the possible number of connections. *Connected components* (CC) of graphs are subsets of nodes that are connected to each other through multi-hop paths. In directed graphs, weakly connected components require just unidirectional links between nodes. Throughout this work we only consider weakly connected components

on directed graphs and refer to them simply as connected components for ease of notation.

Real world graphs very often follow the power law. This does especially hold for social and biological networks. The power law suggests, that the majority of nodes in a graph have relatively low degrees and only very few nodes have an extremely high degree. This results in a degree distribution with a distinct right sided tail [13].

1.3 Knowledge Graphs

Knowledge graphs (KG) are a way to represent entities and the relationship between them. They are directed labeled graphs in which each label indicates the semantic meaning of the relationships between entities. As such, they are commonly used to model real-world information in a structured, and machine-readable way. They are key to semantic web projects but also used in multiple other domains. They include knowledge management and data governance, information retrieval, as well as question answering, and semantic parsing. Famous examples include Wikidata [3], an openly accessible and collaboratively managed source for open data, or Google KG [2] –a knowledge base that feeds the information boxes showing next to a google search result.

Mathematically speaking, a knowledge graph is a subset of the cross product $N \times L \times N$, where N is a set of nodes and L is a set of Labels. Hence, each of these triplets describes a relationship between nodes.

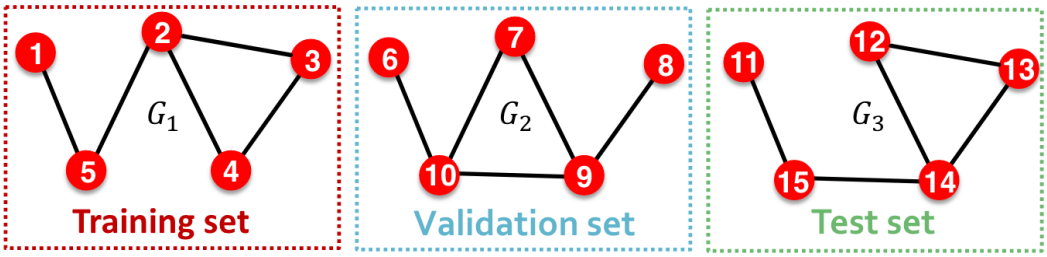
KGs can be heterogeneous or homogeneous in both their relation and node types. In a graph with homogeneous relations all triplets share the same type of relation, i.e. each edge has the same label. Graphs with heterogeneous relations on the other hand, can model different kind of relationships. Similar holds in regard to node types: KGs can either consider every node to be of the same type (homogeneous) or different types (heterogeneous).

1.4 Data Splitting on Knowledge Graphs

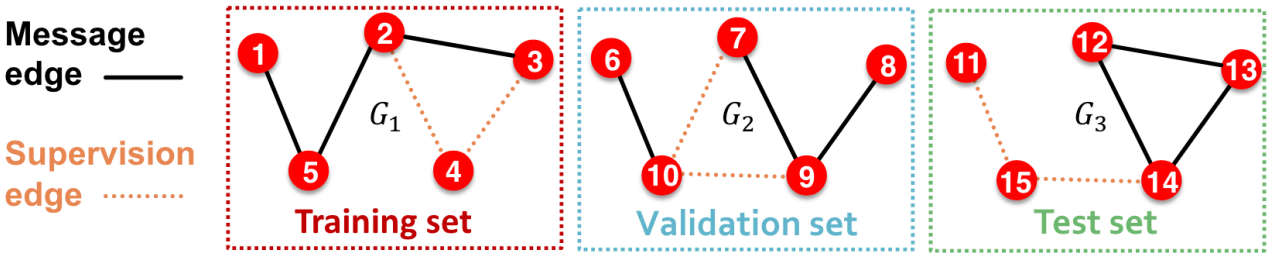
A typical assumption in machine learning (ML) is that the data is independent and identically distributed (i.i.d.). This assumption motivates the split into train, validation and test data. Machine learning on graphs is special in this regard, because the independence assumption does not hold for nodes. In a graph every node depends on its neighbors and — as these depend on their neighbors recursively — on the full graph. Therefore, by removing data from the training set (i.e. splitting) not only the removed data point becomes unobserved, but information is lost for all other data points in the training set as well [16].

Additionally, graph data allows for two different approaches: inductive and transductive learning. While inductive learning corresponds to supervised learning, transductive learning is a special case of semi-supervised learning. In transductive learning not all, but only a subset of the data is labeled. The objective is to predict the labels of the unlabeled subset. However, the unlabeled data is already available during training [13].

The inductive setting for knowledge graph completion (KGC) tasks is depicted in fig. 1.1a. It corresponds to the case where multiple graphs with disjoint node sets are available. One of the graphs is used as the training set, one as the validation set, and one as the test set. The machine learning model is supposed to generalize to completely unseen graphs [16]. In order to provide supervision, each of the graphs is split a second time. The edges of each graph are divided into message-passing edges that are used as input for the machine learning model and supervision edges, which the model tries to predict. Figure 1.1b illustrates the supervision split for each of the three sets. With slight variations inductive learning is



(a) Split into three sets.



(b) Split into message passing and supervision edges.

Figure 1.1 Data split for link prediction tasks in the inductive setting. (a): Three disjoint graphs form the training, validation and test set. The model is supposed to generalize to completely unseen graphs. (b): Each of the sets is split into message passing and supervision edges. Message-passing edges are fed as input, and supervision edges are predicted. Predictions are used for computing loss and performance metrics. [16]

also applicable, if only one or many graphs are available. In the former case, the graph is just split into three disconnected components. In the latter, multiple graphs are combined into one by treating them as disconnected components of the same graph.

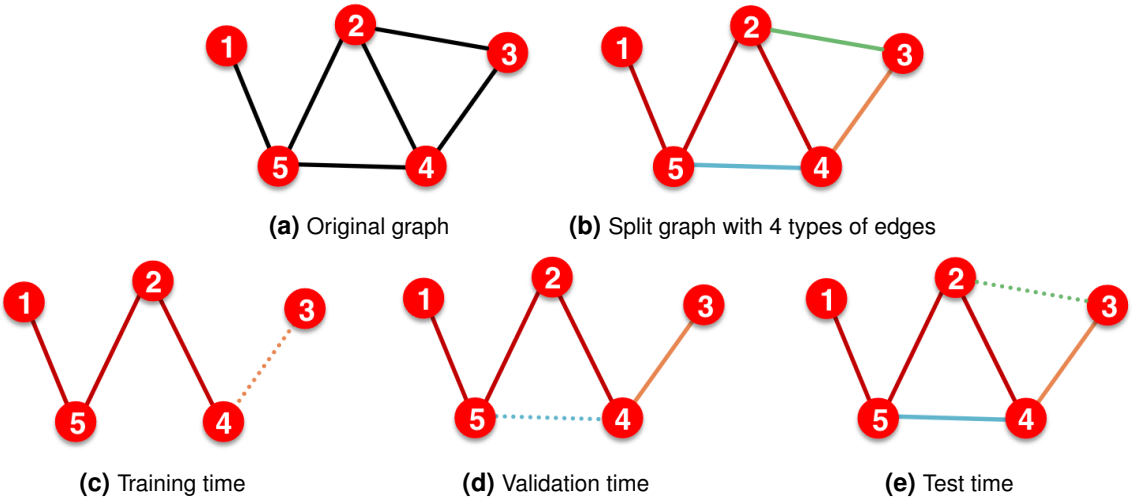
While it is possible to set up link prediction tasks with the inductive approach, the transductive setting is more common. In this case, all sets come from the same graph. As depicted in fig. 1.2 the edges of the full graph are split into four disjoint sets. At training time the training message passing edges are used as inputs and the model tries to predict the training supervision edges. At validation time the training message passing edges and the training supervision edges are combined and form the validation message passing set. Prediction and supervision are performed on the validation supervision set. At test time validation message passing and supervision edges are combined again into one message passing network and the performance is evaluated on the test supervision edges [16].

An additional challenge arises, if — as in our case — the graph structure is the main training signal, e.g. because no or few node features are given. In this case, the model cannot distinguish between nodes of degree zero and must treat all of them the same. As this is likely to produce undesirable results, all splits in common benchmark data sets such as WN18RR [6] or FB15k-237 [23] contain only nodes of degree one or greater, i.e. all connected components have at least size two (see table 3.3).

Small connected components and unseen but connected nodes are likely to result in sub-optimal learning as well. In both cases, the model has no or very little information about the corresponding part of the data set and, therefore, no chance to learn from it. However, as can be seen in table 3.3, common benchmark data sets do not take this into account.

All of the described problems are rooted in the fact that the i.i.d. assumption does not hold for graphs. Due to this, multiple design decisions have to be made regarding the data set:

- Should the training set contain disconnected nodes?



(a) Original graph **(b)** Split graph with 4 types of edges

(c) Training time **(d)** Validation time **(e)** Test time

Figure 1.2 Data split for link prediction tasks in the transductive setting: The edges of the original graph (a) are split into four disjoint sets (b): training message passing edges (red), training supervision edges (orange), validation supervision edges (blue), test supervision edges (green). At training time (c) message passing edges are fed as input, supervision edges are predicted. Predictions are used for computing loss and performance metrics. At validation time (d) the training message passing and supervision edges are combined and form the validation message passing set. At test time (e) the validation supervision edges are included in the message passing network as well. [16]

- Should the validation and test sets contain unseen but connected nodes?
- Should the data set make further guarantees on the connectedness of the graph?

We answer the first two questions with no, as we do not aim to generalize to unseen nodes and our methods rely on paths, i.e. cannot handle completely disconnected nodes well (see chapter 2). As described in section 3.3 we split the data such, that the training message passing network contains all nodes and all of them have degree one or greater. Guaranteeing connectedness is non-trivial, potentially computationally expensive, and — for sparse graphs like LncTarD — partially predetermines the resulting data splits. Due to this and as common benchmark data sets do not make any guarantees there, we forgo it as well.

2 Methods

A wide variety of approaches for knowledge graph completion is available today. Typically they define a score function that assigns a likelihood to each given link between two nodes. We explored geometric embedding models and models based on path-based reasoning. In the following, we give an overview of the methods we used for training and evaluating.

2.1 Geometric KGC Embedding Models

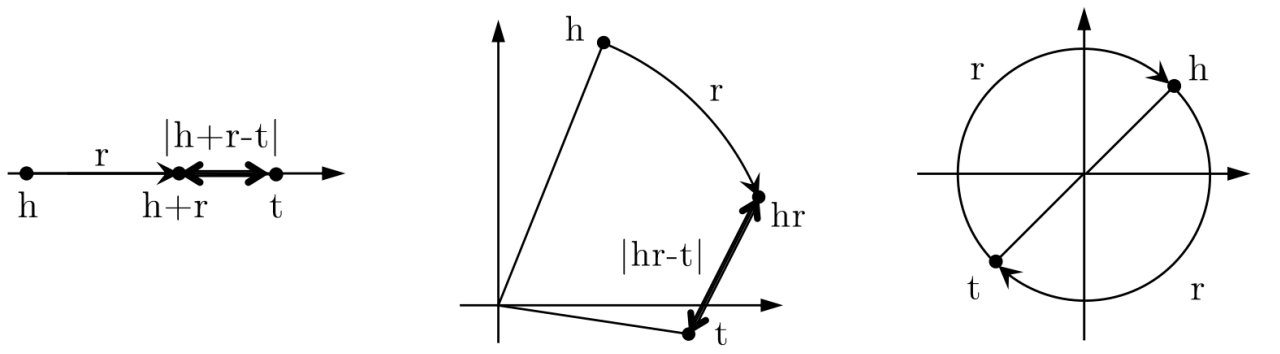
Geometric embedding models define a vector space and learn representations for nodes and relations in this space. Nodes are interpreted as points in space, and relations as geometric transformations. The transformation of a node embedding by the relation is supposed to lead to the embedding of the connected nodes. [17].

TransE [6] is a translation-based model which represents entities and relations in a k-dimensional vector space, i.e., $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^k$, and makes embeddings follow the principle $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$. The L_1 or the L_2-norm — common distance metrics for vectors — are used as dissimilarity measures. The score function is $d_{\boldsymbol{r}}(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|$ using L_1-norm.

RotatE [21] also represents the entities and relations as d-dimensional vectors, but instead of the Euclidean space, it puts the entities and relations into the complex vector space and defines each relation as a rotation from the source entity to the target entity, i.e., $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{C}^k$, the score function is $d_{\boldsymbol{r}}(\boldsymbol{h}, \boldsymbol{t}) = \|\boldsymbol{h} \circ \boldsymbol{r} - \boldsymbol{t}\|$, where $\circ$ denotes the Hadamard (element-wise) product, the modulus $|r_i| = 1$, i.e., the vector's length is one in the complex plane. In contrast to TransE, RotatE is capable of modeling symmetric relations.

2.2 Path-based reasoning with NBFNet

Traditional link prediction approaches leverage heuristics on the paths between a pair of nodes. If paths are explicitly used, the interactions of nodes that lead to a prediction can be directly examined. How-



(a) TransE models r as translation in real line.

(b) RotatE models relation r as rotation in complex plane.

(c) RotatE models symmetric relation r in complex plane.

Figure 2.1 Illustrations of TransE and RotatE with only 1 dimension of embedding [21]

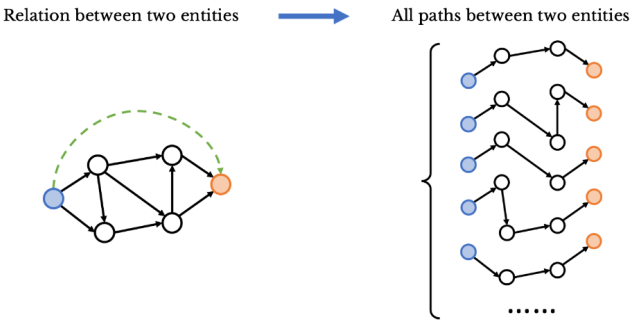


Figure 2.2 Illustration of path-based reasoning.

ever, traditional path-based approaches are limited in their capacity [29]. By making existing path-based methods more flexible and parametrizable, NBFNet is capable of using ML to improve existing reasoning capabilities. This is achieved by exchanging fixed operations of algorithms by learnable counterparts, which leads to an increase in capacity. NBFNet formulates the learning of embeddings as an instance of the so-called generalized Bellman-Ford algorithm, dubbed neural Bellman-Ford. The necessary capacity improvement is achieved by replacing the *boundary condition*, *multiplication* and *summation* operations by learned *indicator*, *message* and *summation* operations [29].

A key observation used prior to NBFNet is the observation that different graph-based methods — like the Bellman-Ford algorithm — can be expressed with a shared framework. Changing the algorithm means changing one of three functions mentioned earlier. As a result, these functions can also be exchanged by parametrizable functions that can be learned. In general, every possible path has to be considered when reasoning about the relationship between two nodes. Usually, this is infeasible because of the exploding number of paths in large graphs, which is illustrated in fig. 2.2. The generalized Bellman-Ford algorithm provides a scalable iterative method, that can be used to prune the number of paths that has to be investigated. The iteration steps to obtain a pair representation $\mathbf{h}_q(u, v)$ between nodes u and v for a given query relation q can be expressed using the following two formulas:

$$\mathbf{h}_q^{(0)}(u, v) \leftarrow \mathbb{1}_q(u = v), \tag{2.1}$$

$$\mathbf{h}_q^{(t)}(u, v) \leftarrow \left(\bigoplus_{(x,r,v)\in\mathcal{E}(v)} \mathbf{h}_q^{(t-1)}(u, x) \otimes \mathbf{w}(x, r, v) \right) \oplus \mathbf{h}_q^{(0)}(u, v), \tag{2.2}$$

where $\mathbb{1}_q(u = v)$ is an indicator function that is one iff $u = v$ and zero otherwise. $\mathbf{w}(x, r, v)$ is an edge embedding connecting nodes x and v via a relation r. $\mathcal{E}(v)$ contains all edges ending in node v. Note that the precise indicator function $\mathbb{1}_q(u = v)$, the aggregation method $\oplus$, and message passing step $\otimes$ are exactly the operations that are relaxed by the use of neural networks. Once a pair representation has been obtained, a classification head on top of neural Bellman-Ford is used to predict probabilities from the embedding $\mathbf{h}_q(u, v)$ from a head node u to a tail node v via the relation q. This can be either tail prediction, where $p(v|u, q)$ is sought, or head prediction, expressed as $p(u|v, q^{-1})$. A huge advantage of path-based reasoning is its great interpretability. For a human, it is easy to see how a model made its decision, when looking at the sequence of nodes that yielded a certain result.

2.3 Measuring performance on link prediction tasks

The output of a knowledge graph completion (KGC) model is typically a score measuring the likelihood of a specific link — i.e. a triplet of head entity, relation and tail entity — to exist. The link prediction task corresponds to a binary classification into *link* and *no link*. One could choose an acceptance threshold on

the score and use typical binary classification metrics for evaluating performance. However, in the KGC setting only positive samples are known. Negative samples must be drawn from the set of unobserved links under the assumption, that the majority of possible links are negative samples [14]. Due to this and the high class imbalance that follows from it, performance on link prediction tasks is typically evaluated by rank based metrics. Three common ones are *mean reciprocal rank* (MRR), *hits@k* (sometimes also called *top K predictive rate*) and *mean rank* (MR) [14]. The metrics are defined as

$$\text{hits@}k = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} \mathbb{I}[r \leq k]$$

$$\text{MR} = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} r$$

$$\text{MRR} = \frac{1}{|\mathcal{I}|} \sum_{r \in \mathcal{I}} r^{-1} = \left(\frac{\mathcal{I}}{\sum_{r \in \mathcal{I}} r^{-1}} \right)^{-1}$$

where $\mathcal{I}$ is the set of all ranks for the true triplets. The ranks are typically computed by the following procedure: For each testing triplet (e^l, r, e^r) the entity e^l is removed. Then the score of (e, r, e^r) for all $e \in D_e$ is computed, where D_e is the set of entities. The resulting list is sorted and all true test, training, and validation triplets except the one in question are removed [6]. If multiple triplets achieve the same score their order is randomly chosen [22]. The rank of the triplet with the true entity e^l is recorded. The same procedure is repeated for e^r. This yields a list of ranks on which the metrics defined above are computed [7].

The metrics as well as the negative sampling procedures are not undisputed [14, 26]. Hits@k is heavily influenced by the chosen threshold, MR is not comparable for data sets of different sizes and susceptible to outliers and there is discussion about theoretical concerns regarding MRR [9, 18]. Moreover, statistical tests for evaluating the significance of performance differences between models are missing from previous studies on KGE models for link prediction [14].

Despite these issues hits@k, MR and MRR remain the most commonly used metrics for performance evaluation on link prediction tasks [14], which is why we decided to use them in the following to report our models' performances.

3 Data Analysis and Preparation

For our analysis, we used the LncTarD 2.0, an updated comprehensive database, as our main datasource, [28].[1] From LncTarD 2.0 we could naturally extract a knowledge graph where nodes are regulators or target genes and the relations are the interaction between them. In total, there are over 8000 interactions, i.e., triplets, present in the dataset. In the following, we provide a more in-depth overview of the data and show results from an analysis conducted prior to training any models.

3.1 Missing type annotations

The LncTarD dataset contains inconsistencies regarding the node types. Therefore, we refrained from using LncTarD's node type annotations and switched to another source for type annotation. The Gencode project [8] comes with extensive annotations of many human genes. Thus, we used the `gen_type` annotation from the Gencode project for all nodes where an annotation is available. For those, where Gencode has no annotation, we used a dummy marker to declare that the node type is unknown.
In order to ensure that there is reasonable coverage of type annotations, we analyzed the number of triplets where at least one of the involved nodes (head/tail) has no node type annotations. Our results are summarized in table 3.1. For every type (head/tail) and for every split, the number of nodes where Gencode does not supply a type annotation is a single-digit percentage number. Therefore, we considered the Gencode type annotations to be sufficient for our analysis.

Split & Type	# of nodes without Gencode type
Train heads	4.33% (1181/27269)
Train tails	4.34% (1179/27192)
Validation heads	3.45% (776/22496)
Validation tails	3.55% (802/22621)
Test heads	3.44% (774/22482)
Test tails	3.53% (797/22607)

Table 3.1 Unknown node type in LncTarD triplets. Numbers in parentheses are absolute amounts, followed by the total number of triplets.

3.2 Type distribution

First, we inspected the distribution of individual node types ("TargetType" and "RegulatorType") and the types of relations ("SearchregulatoryMechanism"). The results are shown in fig. 3.1. We observe that there are mostly lncRNAs acting as regulators, e.g. head nodes. This is expected given that we are dealing with a dataset specifically tailored towards lncRNAs. However, this probably does not reflect the abundance of RNAs in a real environment. Therefore we consider this distribution as a dataset-specific bias. Furthermore, most tail nodes are protein-coding genes. This is not unexpected when taking the biological function into account, i.e. the function of lncRNAs is to change the behavior of other mechanisms. Note that also TFs and other RNA types are targeted quite often. For the relation types, we observe that some like "chromatin looping" or "interact with mRNA" occurs not very often, whereas most interactions are

[1] We downloaded the dataset from `https://lnctard.bio-database.com/`.

of type “ceRNA or sponge” or “expression association”. In general, the occurrences of different relations are unbalanced.

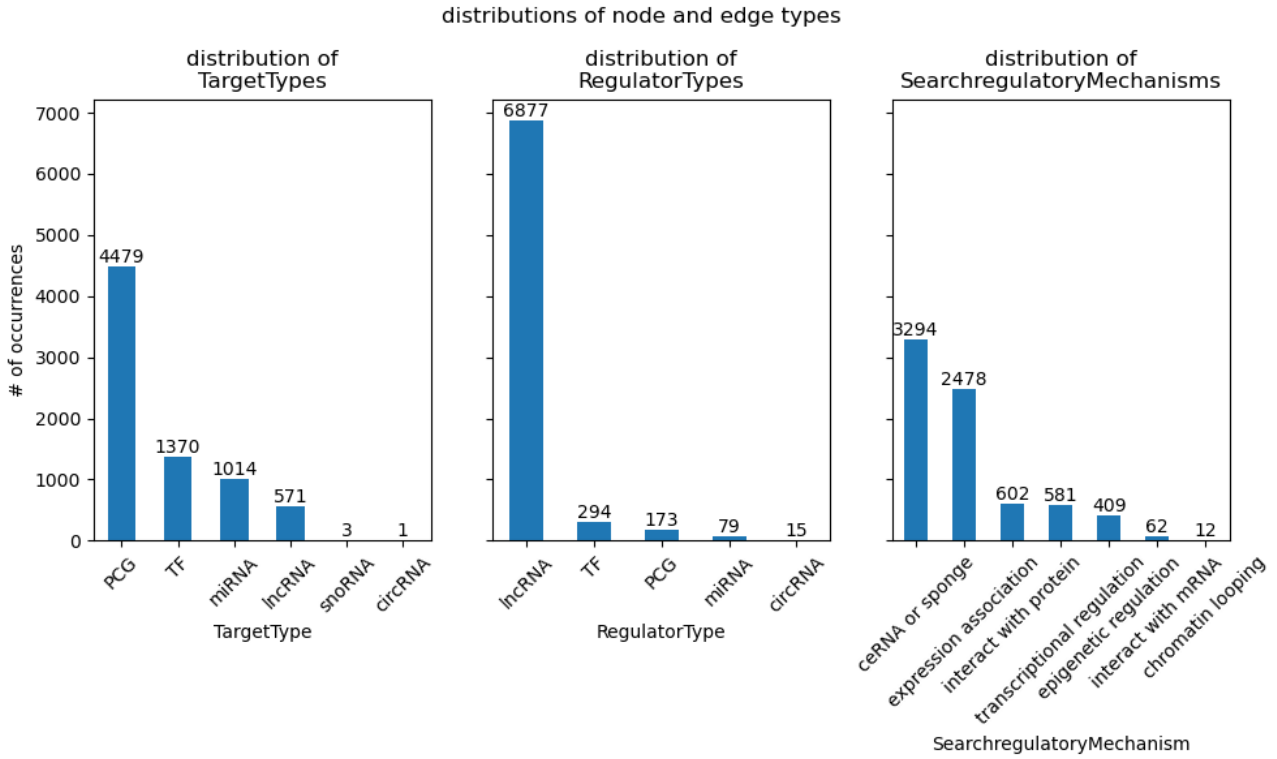


Figure 3.1 Distribution of target, regulator, and relation types present in LncTarD 2.0. Starting from the leftmost column, they are grouped into the different available types for targets, regulators, and search regulatory mechanisms; acting as tail nodes, head nodes, and relation types respectively.

3.3 Splits

For training our models, we first created splits and obtained a train, a validation, and a test data set (80/10/10). As introduced in section 1.4, we further split the train set into a message passing and a supervision set (80/20, yielding a 64/16/10/10 split of the entire data set). We distinguish between two modes, with fact graph, and without fact graph. When a fact graph is used, edges from an external graph containing PPI data serve as the message-passing graph, whereas the train set consists of all triplets in the (original) message-passing and supervision subsets. Note that in this case the size of the message passing, and therefore of the whole dataset, depends on the size of the fact graph. When no fact graph is used, the train set is split into message passing and supervision. The total amounts of triplets for the two modes investigated can be seen in table 3.2. In the following, we give a brief description of the tools we looked into for split generation.

Split	Without fact graph	With fact graph
Message passing	64% (4572)	n.a.
Supervision	16% (1143)	80% (5715)
Validation	20% (714)	20% (714)
Test	20% (715)	20% (715)
Total	100% (7144)	100% (7144 + fact graph)

Table 3.2 Triplet splitting into different subsets.

DeepSNAP [1]: This tool allows for splitting according to the inductive as well as the transductive setting and follows the procedures described in section 1.4. However, it does not ensure that nodes of degree one or more are present in all splits. When using DeepSNAP for creating our splits, we obtained ~700 nodes (~17% of the data set) of degree zero in the training set. As the LncTarD data set is rather small anyway we cannot afford to filter out all of these nodes.

Data set	Split	# of CCs	Size of largest CC	Size of smallest CC	# of nodes	# of edges
FB15k-237	train	5	14496	2	14505	272115
	val	5	14504	2	14513	289650
	test	6	14529	2	14541	310116
WN18RR	train	46	40442	2	40559	86835
	val	30	40679	2	40757	89869
	test	13	40917	2	40943	93003
LncTarD	train	295	3256	2	3908	5950
	val	268	3322	2	3908	6694
	test	249	3367	2	3908	7438

Table 3.3 Statistics on connected components (CC) for common benchmark data sets.

PyKEEN [4]: The library provides two alternative splitting procedures. The `CleanupSplitter` splits the triplets randomly first and afterwards cleans up the data sets by moving triplets to the training set until it contains at least one triplet for every node. The `CoverageSplitter` first moves triplets to the training set greedily and then splits the remaining triplets. In both cases it is ensured that all nodes are contained in the training set with at least one triplet. Note that this is the library we used ultimately. We created our four splits with the `CoverageSplitter`.

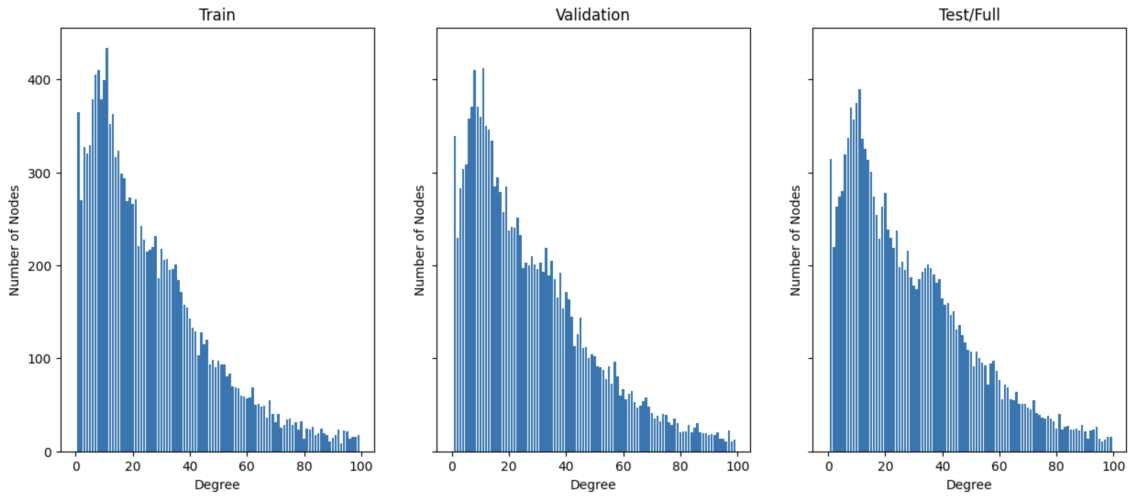
Table 3.3 compares the resulting splits to those of FB15k-237 and WN18RR. The training set is a subset of the validation set and the validation set is a subset of the test set. The size of a CC corresponds to the number of nodes in the CC. Both other benchmark data sets are much larger and denser than LncTarD. The sparsity of LncTarD results in a substantially higher number of connected components. However, the majority of nodes are contained in the largest connected component and despite the different number of connected components, the node degree distribution of the largest connected component matches the degree distribution in FB15k-237 and WN18RR. In fig. 3.2 we show these degree distributions for the three datasets. For every combination of data set and split, we observe — despite being very different in absolute numbers — a clustering towards low degrees in every figure. I.e. the node degrees follow a power law distribution. Note that none of the data sets contains nodes of degree zero. Therefore we find the NBFNet approach applicable to the LncTarD dataset as well.

3.4 Fact graph and data leakage

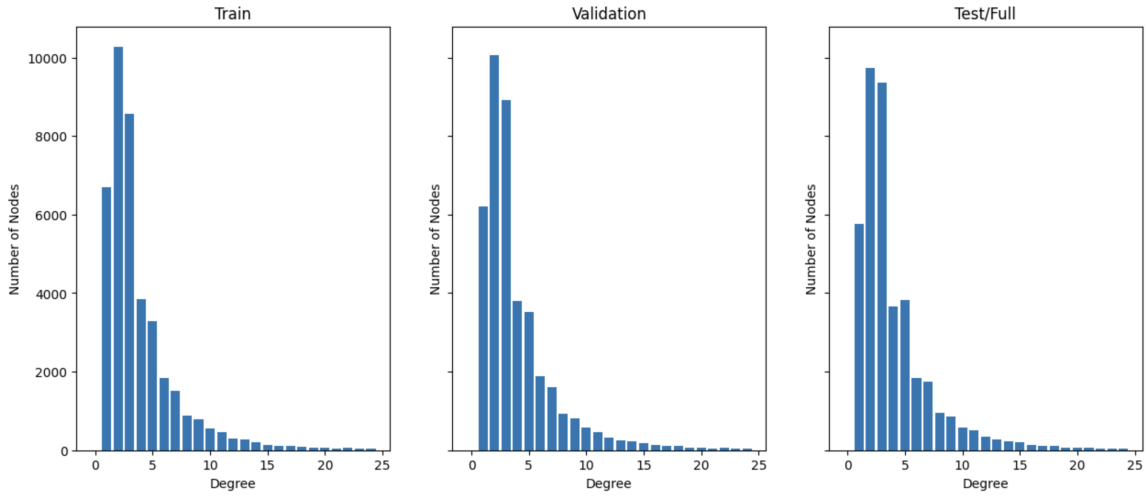
As introduced before, our dataset consists of triplets. In our experiments, we first investigated the impact of using a fact graph for message passing, as described in section 3.3. We picked the Pathway Commons [25] biological knowledge graph as a fact graph since it contains many pathways and interactions.[2] After some initial investigations, which will be presented in the following, we decided to use the graph as-is without modifications.

ML is usually conducted to create a model that generalizes on a task, not a specific dataset. Therefore, most training pipelines are laid out such that they use sets that contain distinct samples, meaning

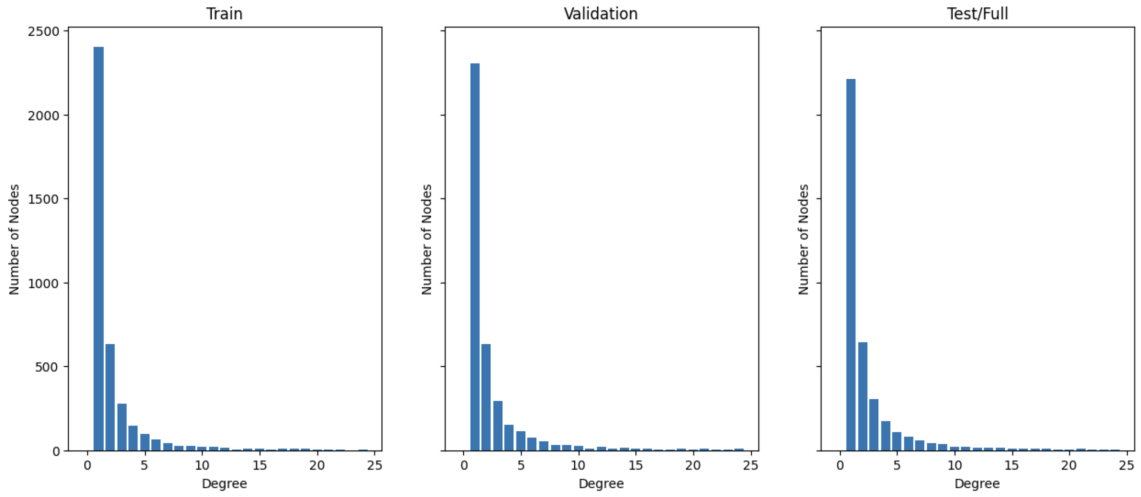
[2] In this paper, we refer to the dataset and the respective graph as PathwayCommons, PPI graph, or fact graph in a synonymous way.



(a) FB15k-237



(b) WN18RR



(c) LncTarD

Figure 3.2 Number of nodes with a specific degree (up to 100 or 25 respectively) for each data set and every split.

that samples are identically distributed and independent, but present in only one of the sets. When performance is determined on samples that were already present in training, the model does not need to generalize. Therefore, such sharing (or leakage) has to be mitigated when evaluating a model's performance.

In our case, the LncTarD and PathwayCommons datasets have been created by independent authors. Thus, it might be the case that some information is shared across the datset. This is inapplicable for our use case because we need to use the PathwayCommons dataset as a fact graph, and the fact graph is shared among the training, validation, and testing phases. If the fact graph contains a link between nodes, that are also connected in the test set, information is shared between the fact graph and training graph. That way data can be leaked into the testing phase. As a result, we investigated the number of connections that are shared among the fact graph and each of the splits. We looked at all triplets in the fact graph and all triplets in the respective split, and calculated the fraction of triplets that are present in both sets. The results of this investigation can be found in table 3.4. Note that there is some triplet overlap between the fact graph and some of the splits. However, because the leakage is below one-digit percentage numbers, we consider the overlap to be negligible. For our investigations, we used the fact graph without any modifications.

Split	Leakage
Train	0.44%
Validation	0.00%
Test	0.46%

Table 3.4 Triplet leakage between each split and fact graph.

4 Results

In this section, we provide a detailed report of the experiments conducted on the LncTarD dataset we prepared in the last chapter and a comparison of the different models.

4.1 Experimental Setups

4.1.1 Environment

Most of our experiments, as well as training and evaluation of each model, have been performed on a server container using 12 vCPUs Intel Xeon Platinum 8255C at 2.50GHz, 40GB RAM, and one GPU NVIDIA RTX 3080 with 10GB VRAM. The operating system is Ubuntu 20.04 with Python 3.10.12, CUDA 11.8, and PyTorch 2.0.0. Only the attempt for NBFNet trained with an external fact graph of protein-protein interaction (PPI) dataset has been tried out on the other server container with 14 vCPUs on an AMD Epyc platform, 100 GB RAM, and one GPU NVIDIA RTX A6000 with 48 GB VRAM.

4.1.2 Geometric Models

With the significant attention that knowledge graph embeddings have received, many software libraries have been developed for training and evaluating the models in this domain. In this project, we used PyKeen [4], an open-source Python library for knowledge graph embeddings, to train and validate the geometric models TransE and RotatE on the inductive splits of the LncTarD dataset with a ratio of 80:10:10 for train/valid/test. We chose $k = 512$ as our target embedding dimension and adjusted and tuned the batch size, the learning rate, the number of epochs, and the number of negative samples while using PyKeen default values for the remaining parameters. The batch size was of range $\{32, 64, 128, 256, 512, 1024\}$. For each positive sample, the number of negative samples was chosen among $\{16, 32, 64, 128\}$. Generally, smaller learning rates require more training epochs [19]. In this experiment when we use a learning rate of 0.001, the number of training epochs was 200, and when the learning rate was reduced to 0.0001, the number of training epochs increased accordingly to 300.

The optimal configuration for TransE: number of negative samples per positive sample was 128, batch size was 128, learning rate was 0.001, and training epochs was 200. The optimal configuration for RotatE: number of negative samples per positive sample was 128, batch size was 128, the learning rate was 0.0001, and training epochs were 300. As a result, RotatE outperforms TransE. The evaluation statistics are displayed in table 4.1.

4.1.3 Path-Based Models

The implementations of the path-based model were built on the official implementation of NBFNet[1] with 6 layers and each layer with 32 hidden units [29]. For all our NBFNet-related experiments, we set the following hyper-parameters fixed: the Optimizer was Adam with the learning rate of 0.005; the number of negative samples was set to 32; the batch size and epoch numbers for training are 32 and 10.

[1] Available at `https://github.com/DeepGraphLearning/NBFNet`.

Class	Model	MR	MRR	Hits@1	Hits@3	Hits@10
Geometric	TransE	488.388	0.03944	0.00304	0.04103	0.09347
Models	RotatE	398.976	0.10523	0.04483	0.10334	0.23176
Path-	NBFNet	214.18	0.13890	0.06839	0.13754	0.30015
Based	NBFNet+Fact Graph	111.524	0.18563	0.10334	0.18693	0.37766
Models	NBFNet+Fact Graph+Entity Types	110.177	0.18604	0.09726	0.19605	0.37614

Table 4.1 Knowledge graph completion results

Head	Relation	Prediction Node	Probability
MALAT1	transcriptional regulation	PTEN	0.80096
MALAT1	transcriptional regulation	MYC	0.74737
MALAT1	transcriptional regulation	KLF2	0.70795

Table 4.2 Top-3 novel prediction given MALAT1 as head and transcriptional regulation as relation

NBFNet To reproduce the NBFNet's competency of knowledge graph completion on the LncTarD dataset, we created new dataset class `LncTarD`[2] for loading the data splits we have prepared. We adapted the configuration file, and trained and evaluated the model. Compared to the traditional geometric models, the path-based graph neural network has a dramatic performance improvement.

NBFNet + Fact Graph Based on the original NBFNet our supervisor Emy Yue Hu came up with the idea of the fact graph and implemented it with a new task `KnowledgeGraphCompletionBiomed`[3] for NBFNet requiring a 4-splits dataset where train set is further divided into message passing and supervision parts. We correspondingly created a new dataset class `LncTarD2`. We also tried masking out the node type information during training by modifying the parameter `heterogeneous_negative` in the configuration file to determine the contribution of the node types to the model's performance. The results in table 4.1 show that using the fact graph can greatly improve the performance of the NBFNet. Moreover, we tried to use an external PPI dataset containing 1,199,579 triples as our fact graph to further improve the model's performance. Unfortunately, the poor RTX 3080 server is not capable of training this big model, so we rented another server with a larger VRAM, but it could still not handle the training for 10 full epochs. After roughly 4 epochs, the performance on the validation set reaches an MRR of 0.179483.

4.2 Novel Predictions

To test the ability of the methods of predicting new links, we did tail predictions for all genes in LncTarD using our best-performing model given lncRNA MALAT1 (metastasis-associated lung adenocarcinoma transcript 1) as head, transcriptional regulation as relation and calculated the conditional probability $p($ prediction node | MALAT1, transcriptional regulation$)$ for all of the prediction nodes. This test indicates whether a gene is potentially transcriptional regulated by the lncRNA MALAT1. A novel prediction is made when there is no direct edge between the head node and the prediction node in the knowledge graph constructed by LncTarD. The top-3 novel predictions with the highest conditional probability are shown in table 4.2. PTEN, MYC, and KLF2 are all protein-coding genes. Studies have already proven that the expression of MYC is directly enhanced by MALAT1 [15], and the rest two genes PTEN and KLF2 are also consequentially affected by MALAT1 [12][20].

Based on the direct regulation of MATAL1 on MYC, we proceeded to delve into the relations between them and attempted to interpret the model's predictions. In the tail prediction, if the condition probability

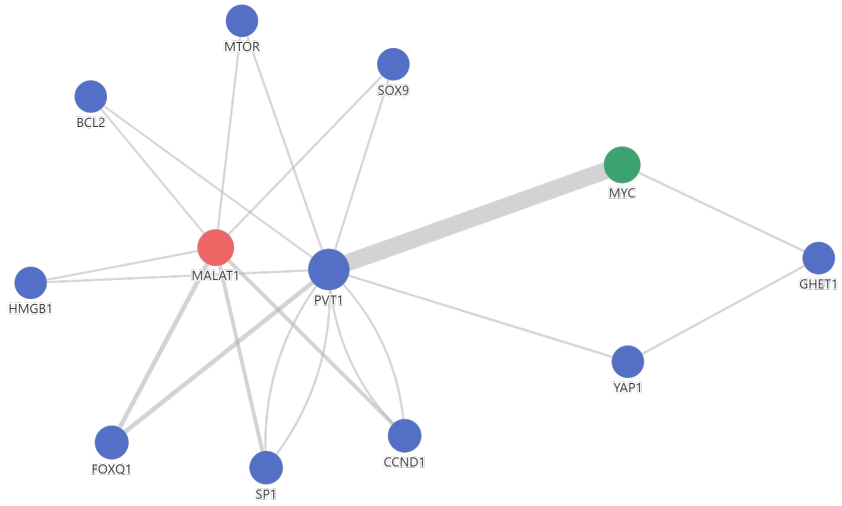
[2]Available at `https://github.com/neftlon/NBFNet`
[3]Avaliable at `https://github.com/emyyue/NBFNet`

Head	Relation	Prediction Node	Probability
MALAT1	expression association	MYC	0.80400
MALAT1	transcriptional regulation	MYC	0.74737
MALAT1	chromatin looping	MYC	0.73230
MALAT1	interact with protein	MYC	0.68971
MALAT1	interact with mRNA	MYC	0.68373
MALAT1	ceRNA or sponge	MYC	0.66323
MALAT1	epigenetic regulation	MYC	0.62232

Table 4.3 Probabilities $P($ MYC | MALAT1, r$)$ for all types of relation r between MALAT1 and MYC.

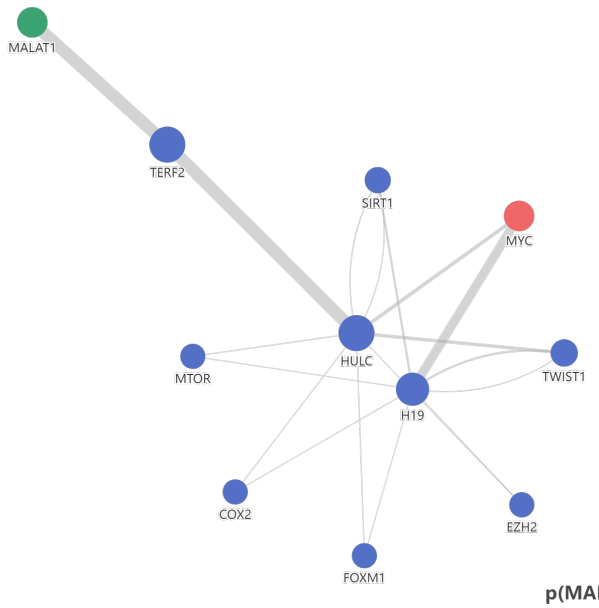
$p(t|h,r)$ is greater than 0.5, we could assume that there tends to be a relation r between head node h and tail node t. We calculated the probabilities $P($ MYC | MALAT1, r$)$ for all types of relation r between MALAT1 and MYC, and all of them are substantially greater than 0.5. Therefore, the hypothesis that MYC is regulated by MALAT1 can be supported. However, the exact type of relation could not be decided just based on the output given by the model. As shown in table 4.3, even though the relation of expression association, possesses the highest value, we cannot simply say that this relation exists for MALAT1 and MYC, because these probabilities are conditioned on the relation. Indeed, the regulatory relation between MALAT1 and MYC is more transcriptionally oriented than expression association. To determine the exact type, further calculations or mechanisms are needed on top of the model outputs.

For interpreting the model, we visualized the paths and their weights between MALAT1 and MYC given the relation of transcriptional regulation as shown in fig. 4.1. In the tail prediction which calculated the conditional probability $p($ MYC | MALAT1, transcriptional regulation$)$, the most important fact for prediction is that MYC is regulated by PVT1. Besides MYC, the PVT1-regulated genes CCDN1, FOXQ1, BCL2, MTOR, SOX9, and HMGB1 are also affected by MALAT1. Thus we could assume that MYC is also regulated by MALAT1. Alternatively, in the head prediction which counts the conditional probability $p($ MALAT1 | MYC, transcriptional regulation$^{-1})$. We can infer that TERF2 is transcriptionally regulated by HULC and MALAT1, whereas MYC is also regulated by HULC, so it may also be regulated by MALAT1.



p(MYC | MALAT1, transcriptional regulation (4))
ranking = 45

(a) Tail prediction given MALAT1 as head and transcriptional regulation as relation



p(MALAT1 | MYC, transcriptional regulation (4)^(-1))
ranking = 13

(b) Head prediction given MYC as tail and the inverse relation of transcriptional regulation

Figure 4.1 Visualization of the paths and their weights between MALAT1 and MYC

5 Discussion and Outlook

Overall the results of the experiments are in line with our expectations. In the experiments with geometric modeling, since gene regulatory relations in biological knowledge graphs are usually N-to-N, RotatE tends to have better performance than TransE, as mentioned in [21]. Our different attempts based on the original NBFNet also progressively improve the model performance to a level where it has the ability to make biologically meaningful predictions. However, the model performances are still largely limited by both the inaccuracy and incompleteness of the data set as well as by time and resource constraints.

One problem is the inconsistent type annotations in our dataset. As described in section 3.1 we were able to obtain many of them by introducing external data sets. As a result, the share of triplets that contain a node with an unknown node type in any of the data splits is reasonably low (~4%). However, the overall share of nodes with unknown node types in the graph is as high as 12.8%. This is especially problematic, as some types of well-studied entities such as MALAT1 are still missing. These represent hubs in our data set. Therefore, missing information for these has an increased effect on a large number of nodes. Acquiring the true node types based on other databases and literature research might help to increase performance further.

Additionally, the relations in our dataset are incomplete and ambiguous. On the one hand, it contains only a subset of known interactions. For example, MALAT1 has other confirmed regulatory effects such as acting as competing endogenous RNA (ceRNA) for miR-205-3p [11], but miR-205-3P is not included in our dataset. On the other hand, for some nodes, there are multiple edges having the same direction between them. As shown in fig. 4.1a CCND1 is regulated by PVT1 with relation types of "expression association" and "ceRNA or sponge" at the same time. Trimming and completing the dataset will likely not only help improve the accuracy but also the interpretability of the model.

Furthermore, the small size and sparsity of our dataset is a constraint in itself. The methods we used were developed and benchmarked on much larger and denser datasets as shown in 3.3. Here it could also be helpful to explore other datasets with the potential to join them into one bigger graph.

Resources were a major concern as well as we were mainly limited to Google Colab. We could not afford to perform extensive hyperparameter tuning and limited it to the absolutely necessary. As NBFNet requires large amounts of memory for training with external fact graphs, we were not able to train it in this setting for a substantial amount of epochs. Visualization of the results required even more memory and wasn't feasible at all with our resources. Using a smaller PPI-graph might reduce training time significantly and would allow for more fine-tuning. It is worth investigating how to best balance training time and the size of the PPI-factgraph, i.e. the amount of information provided by the PPI-KG. Due to the resource limitations, the performance reached in our experiments should be regarded as a proof-of-concept. It would be interesting to repeat the experiments we conducted with more resources and thorough hyperparameter tuning.

An entirely different direction to approach the problem of RNA interaction is sequence-based methods [5]. These take RNA sequences of heads and tails as input and predict interaction based on this without considering relations between further entities. A combination of both approaches, e.g. by introducing sequence embeddings as node features could improve performance. This could also allow for leveraging much bigger data sets by pre-training the sequence embeddings on RNA data not related to regulatory processes.

Another open issue is to determine the type of novel predictions as mentioned in section 4.2. NBFNet gives only the likelihood that a node is a tail given a head node and a relation type, i.e., $p(t|h,r)$, rather than the probability for relations given specific head and tail, i.e., $p(r|h,t)$. We discussed two potential ways to circumvent this issue, but due to time constraints, we have not implemented these approaches. The probabilistic solution is to compute $p(t|h)$ and $p(r|h)$, then according to Bayes' theorem the probability of relations given head and tail can be computed as :

$$p(r|h,t) = \frac{p(t|h,r)p(r|h)}{p(t|h)}$$

The deep learning solution is to add a simple MLP network after NBFNet, which takes the 7-dimensional vectors that result from NBFNet's prediction of all the relationships for the given head and tail as input, and the output can be either a 1- or 7-dimensional vector. The 1-dimensional output tells whether there is a link between the head and the tail. The 7-dimensional output then represents the probabilities for each relation type.

Bibliography

[1] DeepSNAP.

[2] Google.

[3] Wikidata.

[4] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. PyKEEN 1.0: a python library for training and evaluating knowledge graph embeddings. 22(1):82:3723–82:3728.

[5] Iñaki Amatria-Barral, Jorge González-Domínguez, and Juan Touriño. pRIblast: A highly efficient parallel application for comprehensive lncRNA–RNA interaction prediction. 138:270–279.

[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

[7] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. 25(1):301–306. Number: 1.

[8] Adam Frankish, Mark Diekhans, Irwin Jungreis, Julien Lagarde, Jane E Loveland, Jonathan M Mudge, Cristina Sisu, James C Wright, Joel Armstrong, If Barnes, Andrew Berry, Alexandra Bignell, Carles Boix, Silvia Carbonell Sala, Fiona Cunningham, Tomás Di Domenico, Sarah Donaldson, Ian T Fiddes, Carlos García Girón, Jose Manuel Gonzalez, Tiago Grego, Matthew Hardy, Thibaut Hourlier, Kevin L Howe, Toby Hunt, Osagie G Izuogu, Rory Johnson, Fergal J Martin, Laura Martínez, Shamika Mohanan, Paul Muir, Fabio C P Navarro, Anne Parker, Baikang Pei, Fernando Pozo, Ferriol Calvet Riera, Magali Ruffier, Bianca M Schmitt, Eloise Stapleton, Marie-Marthe Suner, Irina Sycheva, Barbara Uszczynska-Ratajczak, Maxim Y Wolf, Jinuri Xu, Yucheng T Yang, Andrew Yates, Daniel Zerbino, Yan Zhang, Jyoti S Choudhary, Mark Gerstein, Roderic Guigó, Tim J P Hubbard, Manolis Kellis, Benedict Paten, Michael L Tress, and Paul Flicek. GENCODE 2021. 49:D916–D923.

[9] Norbert Fuhr. Some common mistakes in IR evaluation, and how they can be avoided. 51(3):32–41.

[10] Julien Gagneur. TUM IN2393: Machine learning for regulatory genomics.

[11] Na Gao, Yueheng Li, Jing Li, Zhengfan Gao, Zhenzhen Yang, Yong Li, Hongtao Liu, and Tianli Fan. Long non-coding RNAs: The regulatory mechanisms, research strategies, and future directions in cancers. 10.

[12] Qi Gao and Yanfeng Wang. Long noncoding RNA MALAT1 regulates apoptosis in ischemic stroke by sponging miR-205-3p and modulating PTEN expression. 12(6):2738–2748.

[13] Stephan Günnemann. TUM IN2323: Machine learning for graphs and sequential data.

[14] Charles Tapley Hoyt, Max Berrendorf, Mikhail Galkin, Volker Tresp, and Benjamin M. Gyori. A unified framework for rank-based evaluation metrics for link prediction in knowledge graphs.

[15] Alessia Iaiza, Claudia Tito, Zaira Ianniello, Federica Ganci, Valentina Laquintana, Enzo Gallo, Andrea Sacconi, Silvia Masciarelli, Luciana De Angelis, Sara Aversa, Daniele Diso, Marco Anile, Vincenzo Petrozza, Francesco Facciolo, Enrico Melis, Edoardo Pescarmona, Federico Venuta, Mirella Marino,

Giovanni Blandino, Giulia Fontemaggi, Alessandro Fatica, and Francesco Fazi. METTL3-dependent MALAT1 delocalization drives c-myc induction in thymic epithelial tumors. 13(1):173.

[16] Jure Leskovec. Stanford CS224w: Machine learning with graphs - YouTube.

[17] Andrea Rossi, Donatella Firmani, Antonio Matinata, Paolo Merialdo, and Denilson Barbosa. Knowledge graph embedding for link prediction: A comparative analysis. 15(2):1–49.

[18] Tetsuya Sakai. On fuhr's guideline for IR evaluation. 54(1):12:1–12:8.

[19] Sadaf Shafi and Assif Assad. Exploring the relationship between learning rate, batch size, and epochs in deep learning: An experimental study. In Manoj Thakur, Samar Agnihotri, Bharat Singh Rajpurohit, Millie Pant, Kusum Deep, and Atulya K. Nagar, editors, *Soft Computing for Problem Solving*, Lecture Notes in Networks and Systems, pages 201–209. Springer Nature.

[20] Kou-Gi Shyu, Bao-Wei Wang, Wei-Jen Fang, Chun-Ming Pan, and Chiu-Mei Lin. Hyperbaric oxygen-induced long non-coding RNA MALAT1 exosomes suppress MicroRNA-92a expression in a rat model of acute myocardial infarction. 24(22):12945–12954.

[21] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space.

[22] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods.

[23] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66. Association for Computational Linguistics.

[24] Wenlun Wang, Lu Min, Xinyuan Qiu, Xiaomin Wu, Chuanyang Liu, Jiaxin Ma, Dongyi Zhang, and Lingyun Zhu. Biological function of long non-coding RNA (LncRNA) xist. 9.

[25] Jeffrey V Wong, Max Franz, Metin Can Siper, Dylan Fong, Funda Durupinar, Christian Dallago, Augustin Luna, John Giorgi, Igor Rodchenkov, Özgün Babur, John A Bachman, Benjamin M Gyori, Emek Demir, Gary D Bader, and Chris Sander. Author-sourced capture of pathway knowledge in computable form using biofactoid. 10:e68292. Publisher: eLife Sciences Publications, Ltd.

[26] Yang Yang, Ryan N. Lichtenwalter, and Nitesh V. Chawla. Evaluating link prediction methods. 45(3):751–782.

[27] Xiaopei Zhang, Wei Wang, Weidong Zhu, Jie Dong, Yingying Cheng, Zujun Yin, and Fafu Shen. Mechanisms and functions of long non-coding RNAs at multiple regulatory levels. 20(22):5573.

[28] Hongying Zhao, Xiangzhe Yin, Haotian Xu, Kailai Liu, Wangyang Liu, Lixia Wang, Caiyu Zhang, Lin Bo, Xicheng Lan, Shihua Lin, Ke Feng, Shangwei Ning, Yunpeng Zhang, and Li Wang. LncTarD 2.0: an updated comprehensive database for experimentally-supported functional lncRNA-target regulations in human diseases. 51:D199–D207.

[29] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction.