Master 2 Bio-informatique
Université de Rennes - UFR SVE
Parcours IBI

Inference of biological interactions on large heterogenous graph networks with machine learning

---

Antoine TOFFANO

Encadrant : Christophe Héligon

Institut Génétique et Développement de Rennes (IGDR)
UMR 6290 CNRS - Équipe CeDRE - Cell Division Reverse Engineering
2 Avenue du Professeur Léon Bernard, 35000 Rennes

Juin 2023

Master bioinformatique

UNIVERSITÉ DE RENNES 1

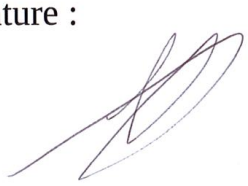
ENGAGEMENT DE NON PLAGIAT

Je, soussigné(e) ..Antoine TOFFANO.....
étudiant(e) en...M2 Bioinformatique.....
déclare être pleinement informé que le plagiat de documents ou d'une partie de document publiés sur toute forme de support, y comprit l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour la rédaction de ce document.

Date : 8 JUIN 2023

Signature :



Document à compléter de manière manuscrite et à insérer obligatoirement en première page du rapport de stage.

Acknowledgements

I would like to express my gratitude to all people that supported me during the development of this thesis, starting with my supervisor Christophe Héligon, whose guiding hand helped me not spread myself too thin throughout my work. Thank you for the great supervision, kindness and for putting into question my perspective on the project, always for the better.

Thank you to all of the CeDRE's team, especially Youssef for his valuable advice on all things related to machine learning. Thank you for taking the time to brainstorm alongside me, the sessions really were a boon to the project.
Thank you to Sylvain for being the beating heart of the team, and while i would not acknowledge it to his face, i really enjoyed being in the same team. The lunch plays were a delight.
Thanks to Pauline and Meline for sharing the burdens of being an CeDRE intern and thanks to Mathis, Louis and Christophe for helping throughout the internship.
Thanks to Hélène and Jacques for their feedback on my work and presentations, and especially Jacques for allowing me the chance of being part of his team.

Thanks to the BIS2 group for the chance to talk bioinformatics, and thanks to the lunch-play group whose Friday sessions were the highlight of my week.

A particular thanks to Enora and Victor for being co-interns at the institute and being overall good friends during this time.

Lastly, we thank Olivier Dameron for his initial input at the start of the project on data integrtion issues, which despite being done before this internship still has had an impact on this thesis.

This work was granted access to the HPC resources of IDRIS under the allocation 2023-AD010314082 made by GENCI.

Table des matières

1 Introduction

The study of the relationship between genes and phenotypes is of great importance in the field of biology, as it unravels the mechanisms by which genetic variations contribute to the development, functioning, and diversity of organisms, advancing our knowledge of genetics, evolution, and human health. However, deriving those relations from the ever-growing amount of biological data and knowledge is becoming increasingly challenging for individuals. This complex and heterogeneous data can be modelized in computer-actionable graphs using frameworks like Semantic Web technologies. This internship aims at taking advantage of multiple data types as a whole to discover new relations between biological entities, like genes, phenotypes or diseases. We hypothesize that by combining heterogeneous data graphs and machine learning we can generate a tool resilient to both sparse and noisy data.

1.1 Background

1.1.1 Knowledge Graphs and Link Prediction

Knowledge graphs (KGs) have become a popular and effective means of organizing and representing large-scale, heterogeneous data 1,2. They are a type of graph database that represents entities as nodes and relationships between entities as edges. The basic building blocks of KGs are called 'triples', which are made of three parts : a 'head' node, a 'relation' link and a 'tail' node. This is similar to the 'subject-predicate-object' in Semantic Web. In fact, data represented in a Semantic Web format are KGs. Heterogeneous graphs in general have been used to represent data in various domains such as healthcare, recommendation systems, and social media, to name a few 3. In the context of biology, KGs have been used to represent biological entities such as genes, proteins, and diseases, as well as their relationships and interactions.

One of the key challenges in managing and analyzing knowledge graphs is link prediction (LP) 1,4, which involves predicting missing or potential relationships between entities in the graph. LP enables the discovery of new knowledge and insights that are not explicitly represented in the graph. For example, in the context of biological KGs, LP can be used to predict potential drug targets or disease associations based on known relationships between genes, proteins, and diseases. Therefore, LP is an essential task in KG analysis, and there is a growing interest in both developing and evaluating methods for link prediction in knowledge graphs.

LP, however, is a difficult task due to the intrinsic nature of KGs. KGs are usually both large and complex, with multiple entities and relation types. This is particularly true for biological KGs. As such, developing efficient LP methods requires the integration of multiple techniques, such as graph embedding methods, machine learning and even natural language processing.

1.1.2 Introduction to the data : C. elegans and Wormbase Database

Caenorhabditis elegans (C. elegans) is a species of nematode worm that is widely used as a model organism in biological research. C. elegans has a relatively simple yet complete anatomy, a short lifespan, and a well-defined nervous system, making it an ideal model

organism for studying genetics, development, and behavior.

The WormBase database 5 is a comprehensive database of genomic and biological information about C. elegans, as well as related nematode species. It provides a wealth of information, including genomic sequences, gene expression patterns, protein interactions, and phenotypic data. WormBase is a community-driven effort on an international level, funded by public institutions including CalTech, The Wellcome Trust Sanger Institute and EBI among others. WormBase is a founding member of the Alliance of Genome Resources Project.

Previous work in the lab done by Christophe Héligon consisted of generating a knowledge graph from the raw Wormbase data and to integrate it as RDF in a SPARQL endpoint. RDF (Resource Description Framework) is a data model for representing knowledge in the form of subject-predicate-object triples. A SPARQL endpoint is a web-based interface that allows users to query and retrieve data stored in RDF format using the SPARQL query language.

This C. elegans Knowledge Graph (CeKG) by itself is a valuable resource for researchers studying this model organism. It contains a variety of nodes representing genes, proteins, phenotypes, diseases, and links representing how they interact with each other. The CeKG is linked with different types of relationships, such as gene-phenotype, gene-gene, and gene-disease interactions. The data is represented in what is called a 'reified' format, meaning that a given relation between a gene and a phenotype is represented as a node linked to both the gene node and the phenotype node (Figure 1.). This intermediary node will be called an 'Annotation' node throughout this thesis.

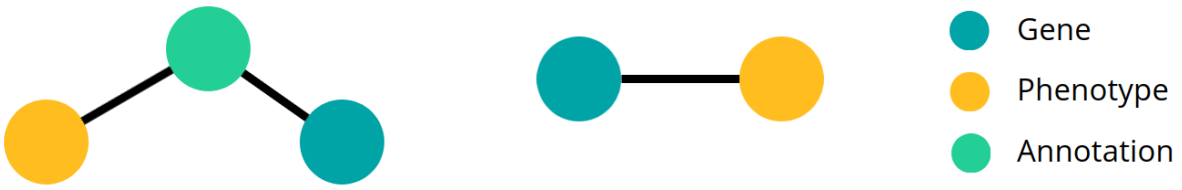


FIGURE 1 – *The reification of RDF data. The data can be represented using reification (left) by connecting information through an annotation node. This is useful to represent metadata about a triple, allowing for example to add the source article or the type of disease associated with this specific interaction. While more expressive, this structure is however more topologically complex than a non-reified structure (right).*
An overview of the general structure of the CeKG can be found in Figure 2.

1.2 Problem Statement and Objectives

The complexity and size of the CeKG make predicting links between biological entities and concepts a challenging real-life task. However, it also presents an opportunity to develop an efficient method and offer useful insights to biologists, saving them time and money in experiment prioritization, especially when it comes to the study of gene-phenotype relationships.

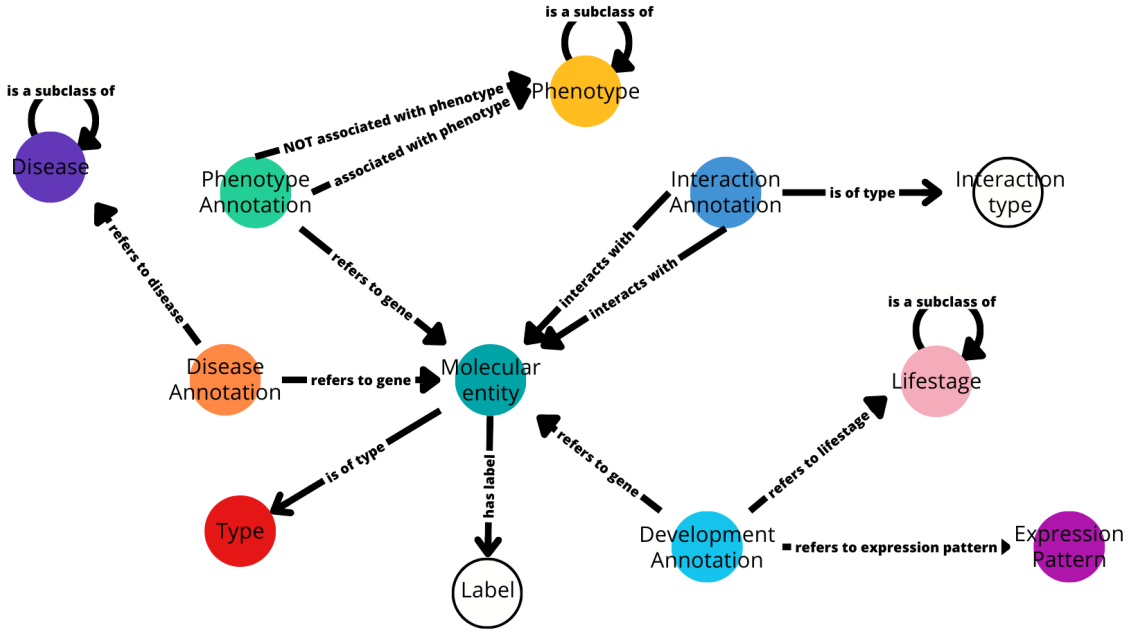


FIGURE 2 – *Overview of the CeKG data structure, or data model, selected in our experiments* (called 'Base' further down the line). Each type of annotation is represented as a colored node which points to different properties of this annotation.
Example : Molecular entities -including genes- (turquoise) are hubs connected to every annotation type which provide information (data and metadata) about that entity.

Therefore, this internship aimed to implement and develop graph embedding-based link prediction methods to predict relationships between biological entities in the CeKG.

The end goal of this project is to ultimately have a set of tools for biologists to predict interactions and a modulable workflow that allows bioinformatician users to train models to predict whether a relation exists between two biological entities, and do so in a manner that is resilient to both noise and lack of information in the graph.

2 Literature Review - Graph Embedding

Graph embedding is a technique that allows for the representation of graph data (i.e. nodes and relations) in a vector space (also known as latent space), making it easier for computers to analyze and process the data by representing it numerically. The process of embedding a graph consists of finding a mathematical function that maps each node in the graph to a point in the vector space, while preserving the information carried by the node (its properties and relations).
In this section, we review some of the popular graph embedding techniques that have been used for the LP task that we will implement.

2.1 Translational Methods

Translational methods are a class of graph embedding techniques that aim to capture the properties of a graph by modeling the interactions between nodes as translations in an embedding space. These methods include TransE 6 and TorusE 7, among others.

TransE is the most popular model due to it being historically the first to introduce a concept on which most methods are based. The idea behind TransE is to represent nodes as a vector in the embedding space and the relationship between them as a translation from the head entity to the tail entity (Figure 3.).

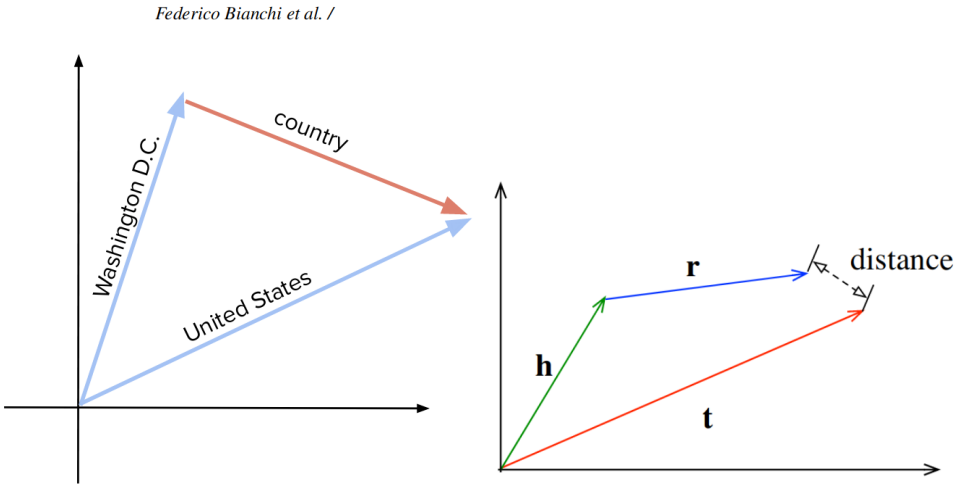


FIGURE 3 – *Example of how TransE represents entities and their relationships in vector space.*

Node and relation embeddings are computed iteratively at each step and evaluated using a scoring function. The driving idea of this score function is that the sum of the head node vector *Washington D.C.* with the relation vector here country should ideally be similar to the vector representation of the tail node United States ($h + r \approx t$). As such, the algorithm tries to both minimize that score function for real, existing triples, called 'true' triples (which amount to reducing the distance between the two parts of the equation in the embedding space) and maximize it for false, not existing triples, called 'corrupted' triples.

Figure from 2 Text adapted from 4.

However, TransE forces the embeddings to be on a sphere in the latent space. This representation means that embeddings are warped (Figure 4a.) 7. TorusE is a method that tries to fix this regularization problem by changing the embedding space from a sphere to a torus (Figure 4b.).

2.2 Bilinear Methods

Bilinear models often tend to use a multiplicative approach and to represent the relationships as matrices in the vector space. Popular methods include RESCAL 8, HolE 9, and DistMult 10, among others.

RESCAL, for example, represents each relation as a full rank matrix and the interaction between the elements as products between entity vectors and the full rank matrix

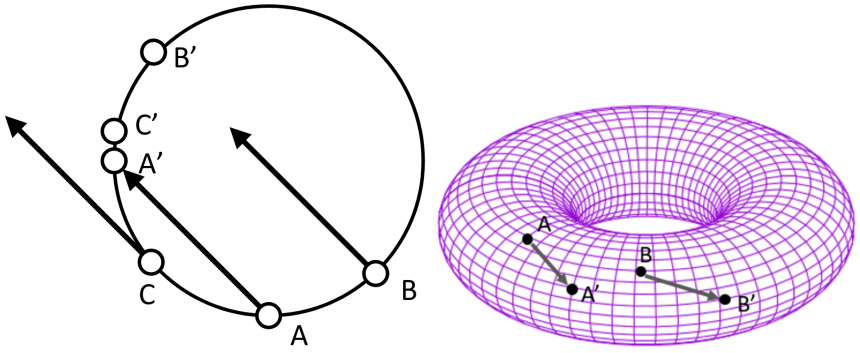


FIGURE 4 – *(a). 2D representation of embeddings obtained by TransE.* The spherical representation means that in most cases, embeddings are unable to fulfill the previously mentioned principle h + r ≈ t. In this figure, only the triple (A, r, A') holds true with the principle, while (B, r, B') and (C, r, C') do not. *(b). Representation of embeddings on a* 2-dimensional torus obtained by TorusE. Due to the geometric nature of a torus, the scoring function is constrained, as it 'loops' over the torus. Embeddings of the triples (A, r, A') and (B, r, B') are illustrated. Note that [A'] - [A] and [B'] - [B] are similar on the torus.
Figures from 7. Text adapted from the same source.

(Figure 5.). This matrix, because of its high expressivity, has a tendency to result in overfitting due to the large number of parameters.

In ComplEx 11, each entity and relation are represented by two vectors : a real-valued vector Re(e) and an imaginary-valued vector Im(e) (Figure 6). This allows ComplEx to capture symmetric (two way) and antisymmetric (one way) relationships (Figure 7.) without adding numerous parameters, lowering the resulting model's potential to overfit. ComplEx was considered as of 2020 to be one of the best performing models in KG embedding 4.

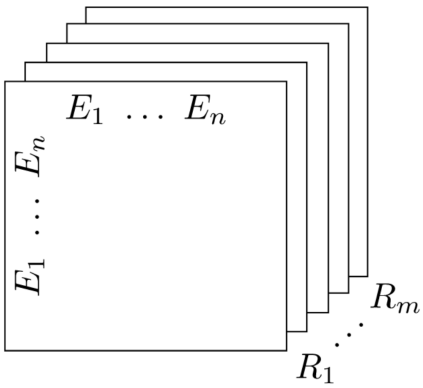


FIGURE 5 – *Tensor model for relational data in RESCAL.* E1 · · · En denote the entities, while R1 · · · Rm denotes the relations in the domain.
Figure taken from

8.

2.3 Deep Learning Methods

Deep learning methods have been increasingly used for graph embedding due to their

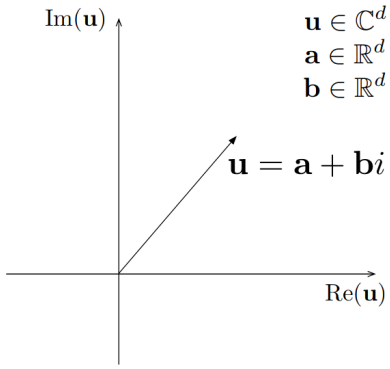


FIGURE 6 – *Complex vector space. An entity u, whether a relation or a node, is made up of both a vector* a in real space Re(u) and a vector bi in complex space Im(u).

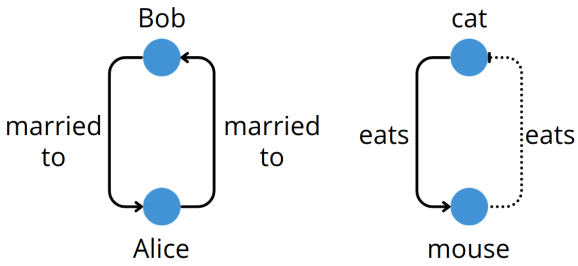


FIGURE 7 – *Symmetrical and asymmetrical relations. A symmetrical relation goes both ways. If Bob is married to Alice, then Alice is also married to Bob. However some relations are asymmetrical ; While cats eat mice, the opposite is not true (or you have a serious mouse problem).*

ability to capture complex patterns and hierarchical representations in data. These methods include (among others) graph convolutional neural networks, which uses convolutions to learn node embeddings by aggregating information from the node's neighbors. ConvKB 12 is one such method, which works by applying convolution operations to triple embeddings in order to learn its semantic representation (Figure 8.).

Recently, with the fast progress of language models, researchers have tried to leverage those models' capabilities to integrate the intrinsic information contained in textual descriptions of nodes and relations to improve embeddings. One such model, Relphormer, uses BERT (a large language model) 13 to encode the hierarchical relations between entities in the knowledge graph. The method outperforms other previously mentioned models on both link and relation prediction tasks 14 on non-biological datasets.

2.4 Random Walk with Restart

Interestingly, graph embedding in biology is mostly done through Random Walks with Restart (RWR). RWR is a graph-based algorithm that models the probability distribution of a random walker moving from one node to another, with a restart probability at a specific node. The probabilities of ending the walk on different nodes of the graph is then used as the embedding of the starting node.
This method has the advantage of being able to add others nodes to the graph without necessarily requiring to update all other embeddings in the graph.
However, it requires a lot of computing time and only takes into account graph topology.

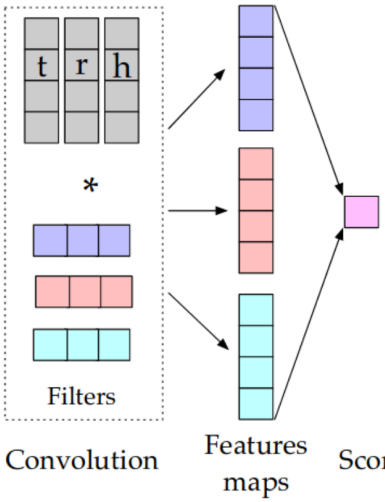


FIGURE 8 – *Convolutional process involved in ConvKB. The embeddings of the triple are represented as a three by four matrix on which is applied a series of convolutional filters. The concatenated feature maps are then computed with a weight vector, resulting in a score for the likelihood of this triple existing.* Figure from 2.

To take into account heterogeneous topological information, methods have emerged like MultiVerse 15 and PhenoGeneRanker 16 17, but at the price of adding additional hyper-parameters.
Due to the quadratic computing time required to run a RWR algorithm ($O(n^2)$) 15 and the high number of nodes contained in our dataset (~ 600 000), we decided not to implement this type of embedding method during this internship, and instead go with the previously mentioned machine learning methods, most of them having a linear time complexity ($O(n)$). Comparing the performance of RWR with the presented machine learning methods is however planned.

3 Methodology

The end goal of this project is to ultimately have a workflow that allows the user to predict whether a relation exists between two biological entities. The usual approach when it comes to such a scenario, i.e. predicting a positive or negative output, is to build a binary classifier. However, a classifier needs a numerical input found in the form of a node's embedding.

Following this thought, our pipeline is divided into two main parts. First, we train what is called a graph embedding model (EM) that can generate a numerical representation of the nodes and relations of our graph (See III.B - Graph Embedding).
Then, we train a classifier that determines from two embeddings whether a relation exists between the two corresponding nodes (See III.C - Link prediction classifiers). Once all models have been trained, we can predict from two input nodes whether a link exists between them.

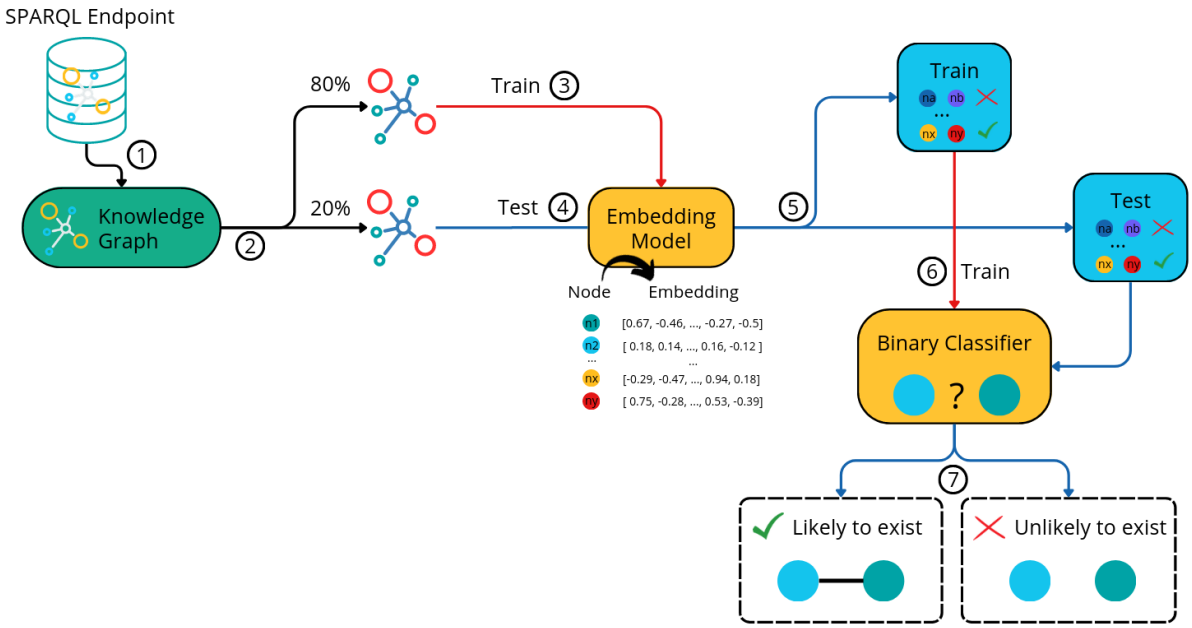


FIGURE 9 – *Overview of the methodology.* 1. Data selection through SPARQL queries. 2. Dataset splitting. 3. Training of the graph EM. 4. Testing of the graph EM. 5. Transformation of the nodes of the test dataset by the EM into their corresponding embeddings. 6. Training of a binary classifier to predict whether a link exists or not. 7. Inference of link existence on the leftover test set.

3.1 Data Preprocessing

In this section, we will talk about the data itself, and the processing steps it undergoes before the following experiments. This corresponds to Figure 9, steps 1-2.

The CeKG used in this study is stored in a SPARQL endpoint (Figure 9-1.), which provides a platform for querying the graph. This approach has been chosen due to its respect of FAIR principles of bioinformatics, as the original graph can be stored in its untampered format, and users can still modify the graph structure through SPARQL queries. Reproducing results then becomes as simple as sharing the SPARQL queries used to preprocess the graph.

A python script has been developed to interact with the SPARQL endpoint through SPARQL queries. The query results are stored by the same script as a triple (n3) file.

3.1.1 Data representation

Representing graph data of a knowledge base in a reified format introduces several challenges compared to a non-reified format. It increases graph topological complexity, making navigation and interpretation of the graph more intricate due to the addition of intermediary nodes. From a machine learning perspective, because reification adds steps to interpret the graph, it reduces semantic simplicity which affects model development and comprehension. It also requires more memory, computational resources, and time for processing.

However, the reified structure enhances expressivity by allowing representation of complex relationships and contextual information. It improves provenance tracking by

capturing metadata about the origin and source of statements. Fine-grained modeling of statements enables a detailed representation of knowledge, accommodating additional metadata like annotations, certainty, or expression values.

Considering the default reified format of the CeKG data, we initially chose this structure. However, we later explored the impact of a non-reified format on algorithm performance.

While choosing the right data structure is an important part of the work, selecting which kind of data is used is as important, if not more.

3.1.2 Data selection

When dealing with large-scale knowledge graphs like the CeKG, not all data is necessarily relevant for link prediction. In fact, some types of data may introduce noise in the form of unwanted nodes into the graph, which can hinder model performance. For example, information about the date at which a datapoint was added, or the author who added the datapoint are unlikely to be useful for our task.

To address this issue, we manually reviewed the graph data types and structure and from this manually identified the most informative and relevant data types in the CeKG. In addition, we sought the input of biologists to both help in the selection process and suggest any additional data type that may have information worth adding despite the potential noise they introduce.

The features initially included in the data are physical interactions between genes (including through proteins), relations between molecular entities (genes, proteins, non-coding RNAs, etc, ...) and phenotypes or disease (including human orthologs), expression patterns of molecular entities and node labels. The graph is further structured using ontologies for phenotypes, life stage, and diseases (because a phenotype can be a subclass of another).

Some information has been excluded, such as expression values, as we consider the amount of data (~5 millions triples) for this data type alone large enough to obfuscate the informational content of the other data types. Also, the expression data available in Wormbase may not be relevant to the biological question of CeDRE team as it is. In total, the amount of data selected for the experiments amounts to ~1.41 million triples. See a graphical representation of the bias in Figure 10a.

Some other data types are left out in the first tests (ie. GO annotations and ontology) despite being considered potentially meaningful because of their complexity in both relation types and amount of data. This is done in order to first determine the performance of each model type and good hyperparameters for each on a smaller, less complex dataset. The processes of both data representation and data selection are handled exclusively through SPARQL queries, making the workflow fully reproducible by simply sharing the query used. This also facilitates further experiments, as the user only needs to modify the SPARQL queries to include or exclude specific data types.

3.1.3 Dataset splitting

Dataset splitting is a crucial step in evaluating the performance of link prediction

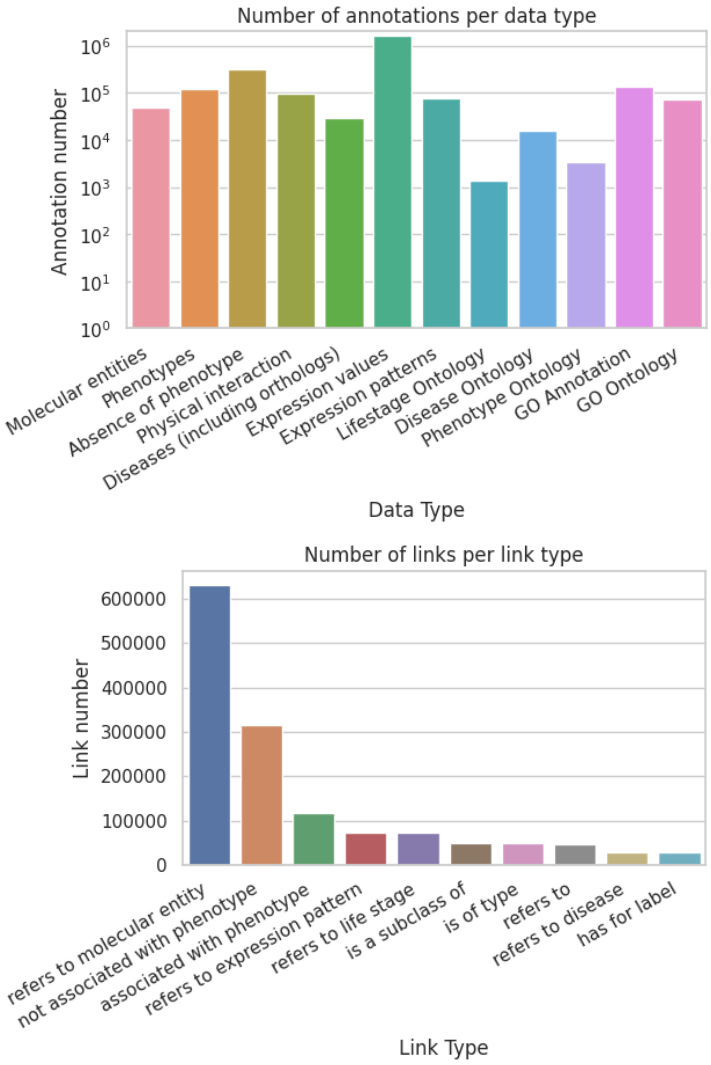


FIGURE 10 – (a). Number of annotations per data type in the raw knowledge graph in log scale. This is calculated over the full graph, for all annotations. (b). Number of links per link type in the cleaned knowledge graph. The most linked data types are the molecular entities followed by the phenotypes. In total, ten link types exist in our initial dataset.

methods. In this study, two methods of dataset splitting were considered.
The first is a random split where certain relations are omitted from the train set to constitute the test set, and the second is a time-based split. This second method depends on the WormBase version, where new links in recent versions are used as the test set. Data added iteratively in previous WormBase updates are not equally distributed between types however. Currently new genes are rarely found, while new interactions are found every day. It is also worth noting that going further in the WormBase release means adapting the formatting script, as the data structure sometimes changes. These are the main reasons we have gone with the random split approach.

There is one bias to keep in mind when it comes to splitting data. An embedding can only be computed for a node if this node exists somewhere in the training set, even if it is a single instance. During the split, all nodes are distributed between sets according to this rule.

Practically, this implies that nodes with only a single connection are absent from the validation set. As such one affected type of node, containing the textual description of a node (i.e. its label) is not present in the other sets. This is not expected to have a meaningful impact on results, as these nodes are few in numbers (See Figure 10b, Link type 'has for label').

At the end of the preprocessing step, we are left with two datasets ; One for training the EM, the other to test it. Those datasets are in a reified structure, with only certain types of data manually selected.

3.2 Graph Embedding

In this section, we will give an overview of the methodology used to embed the nodes from the preprocessed CeKG, which are the next steps in the pipeline (Figure 9, steps 3-4).

3.2.1 Embedding methods

Our approach consists of training multiple popular types of graph embedding methods to get an overview of how different methods perform on our dataset. We implemented two 'translational' algorithms, namely TransE and TorusE, two 'bilinear' algorithms, namely ComplEx and RESCAL, and one 'deep learning' method, ConvKB. These methods are widely used and have shown promising results in various applications 6–8,11, although often outside the biological field. As such, benchmarking their ability to generate high quality embeddings on our dataset is the first step in this work.

In addition to the above methods, we also tried to implement the language model-based method Relphormer 14, to compare this approach to the more popular ones. Relphormer is a recent method that uses BERT 13 (a large language model) to encode the entities in the KG using their textual label. While implementing such a method adds a whole new information layer to the KG, actual implementation during this internship was layered with bugs and hard-to-meet technical requirements (namely, GPU VRAM). Due to time constraints, the implementation could not be done in this timeframe, but still holds a lot of potential.

Each EM is trained on 80% of the dataset, and tested on 10% (Figure 9-2).

For a comprehensive understanding of the performance of each method on the C. elegans dataset, a grid search has been conducted to optimize hyperparameters. The range and selection of hyperparameters are derived from both the original method's publication and preliminary experiments (Not shown). The hyperparameter search typically involves parameters such as epochs, batch size, distance calculation function, and loss function. See Appendix - Preliminary experiments for additional details on experiments done on each method.

The selection of these graph embedding methods provides a diverse set of techniques to capture different aspects of the C. elegans KG. The evaluation of these methods will be discussed in the Discussion section.

3.2.2 Evaluation metrics

In order to assess the performance of the embedding methods, we assess the quality of the embeddings by using evaluation metrics that measure the ability of the models to predict link types between entities in the knowledge graph. The core idea is that if the model is able to reliably predict the right type of link between two nodes, then it has done a good job of encapsulating the node's information content into embeddings.

This relation prediction task involves producing a likelihood score for each relation type between a head and a tail entity using their embeddings, and these scores are ranked to produce a ranked list of possible relations.

We use the Hit@k metric, which measures the proportion of instances where the correct relation is ranked in the top k positions to evaluate the performance of our EM. In our experiments, we mainly use k=1, or Hit@1, to evaluate the ability of the models to make accurate predictions.

By using these evaluation metrics, we can compare the performance of the different embedding methods and determine which methods are most effective for accurately representing the data in the CeKG.

However, predicting the type of a link between two nodes is not enough : we need to predict whether that link exists in the first place.
Indeed, a very large majority of potential links in the graph do not actually exist. This means that we need to develop a method that can also predict the presence or absence of a link between two nodes. This is a crucial step in the link prediction pipeline.

3.3 Link prediction classifiers

The binary classifier uses the concatenated embeddings of two nodes to determine the likelihood of a link existing between those nodes or nodes. The overview of the process is detailed in (Figure 9, steps 5-7.).

Since a binary classifier requires exemples for both outcomes, negative examples were generated through triple corrupting, which consists of 'corrupting' a triple by replacing either its head or tail by an unconnected node. We used a Bernouilli sampler to perform this operation (See Appendix - Bernoulli Negative Sampler). We settled on an equal amount of positive and negative examples. Practically, this means that our dataset is made of about ~440 000 data points in the first implementations of our dataset.

The training set of the classifier includes the testing set of the EM. Since there is no relation between those two models, reusing a set is not a problem.
The stratified k-fold cross-validation approach is a technique used to evaluate the performance of a model. It involves dividing the dataset into k subsets or folds, each conserving the distribution of the original data. A set of folds are used as the validation set and the remaining folds are used for training. This process is repeated k times, and the performance metrics are averaged to provide a more robust estimate of the model's performance.
Here, we use k=10 folds, and use 8 folds for training while 2 are held out for testing.
Practically, this can be seen as 10 different models trained on 10 different 0.8/0.2 train/test splits.
Since there are multiple ways to do binary classification, we decided to train different

types of classification models using the python pipeline PyCaret (See Material - Software). We will use the Accuracy metric to rank the different models. Accuracy is a simple metric, basically defined as the correctly classified data points divided by the total amount of data points.

However, we must keep in mind the ultimate goal of the project, which is not to discriminate between true and false links but to find links where there is none in the graph but should be.
This means that the metric of interest is not Accuracy alone. In fact, we are interested in both maximizing Accuracy while minimizing the amount of false positives, where a link is wrongfully predicted. This is because our false positives are the list of predicted candidate links, and reducing it as much as possible filters the bad candidates that do not actually exist from the reliable ones. The Precision metric focuses on minimizing false positives and is calculated as the ratio of true positives to the sum of true positives and false positives.
By comparing the performance of different classifiers, we should be able to identify the right architecture to predict biological relations from our graph embeddings.

3.4 Material

3.4.1 Hardware

Models were trained on GPU units hosted on High Performance Computing clusters. Debugging and initial exploration were done on the CeDRE's team servers, on a variety of GPU units including a NVIDIA GeForce RTX 2080 Ti, Tesla P100-PCIE-16GB and a Tesla V100-PCIE-32GB.
Production and hyperparameter tuning were done on the JeanZay HPC cluster. Unless specified otherwise, all reported metrics are obtained this way using Tesla V100-16GBs.

3.4.2 Software

All software used are from open-source public repositories, in order to have a FAIR (Findable, Accessible, Interoperable and Reproducible) approach.

Environment management is done through miniconda. Environment configurations and developed software are available in the github repository for easy replication.

The raw data is stored on CeDRE team's server and accessible through Rennes University Medical School subnetwork at the following SPARQL endpoint : cedre-16a.med.univ-rennes1.fr :3030/
Communication with the endpoint is implemented with the SPARQLWrapper python package.

As knowledge graph embedding is not a new field, several repositories have been dedicated to facilitating research in this domain. Multiple pipelines dedicated to KG research are available online, and we selected one pipeline based on the following criteria : Whether the pipeline handles all functions relevant for our purpose, whether it supports the models we wanted to implement and last but not least if it has a fleshed-out documentation. We used the TorchKGE pipeline 18, as it natively supports all methods presented in this work, with the exception of Relphormer. A few tweaks were however needed to

have certain functions work with our experiments (See Appendix - Relation prediction on TorusE).

Classification models were trained using the PyCaret package. Pycaret is a package that works as a pipeline for training, tuning, and comparing different machine learning models for binary and multi-classification problems.

4 Results

4.1 Evaluation of Graph Embedding Methods

To evaluate the performance of the different graph embedding methods on the CeKG, we used a relation prediction evaluation pipeline (See Methodology section), where we train our model to create embeddings of nodes and relations, and measure the ability of the methods to predict missing link types. Here we report the results using the Hit@1 metric (See Methodology - B.2.).

All methods have been trained on a range of hyperparameters depending on the method.

4.1.1 Hyperparameter Search

A grid search is a widely used method in machine learning to identify optimal hyperparameters for models. It entails systematically exploring a predetermined set of hyperparameter combinations and assessing the performance of each combination using a selected evaluation metric. This technique helps in determining the best configuration for the model by exhaustively searching through the hyperparameter space.

Figure 11 shows the embedding quality of the different EMs depending on the following hyperparameters : batch size ranging from 64 to 10 000 and epoch from 5 to 100.

Batch size denotes the number of training examples processed in a single iteration before the model's parameters are updated.

An epoch is a complete pass through the entire training dataset, corresponding to all batches of data being seen.

Other hyperparameters are fixed. The embedding size is fixed to 50, and the margin value to 1. Preliminary tests (Not shown) have shown that increasing embedding size does not seem to improve performance by a significant margin. Doing so not only increases the training time while making the model more susceptible to overfitting. For all methods but ConvKB, the loss function is the margin loss. For ConvKB, the number of filters is set to 10 and the loss is binary cross entropy.

We observe that most of the graph embedding methods perform well on a relation prediction task, indicating that they are able to capture meaningful information about the CeKG. Among the methods, ComplEx and ConvKB perform the best in terms of Hit@1 in this grid search. TransE and RESCAL perform relatively poorly compared to the other methods. Note that since we use the Hit@1 metric and that the graph used here has a total of 10 unique relations, a completely random method would attain a score of 0.1, while a perfect method would attain a score of 1.0.

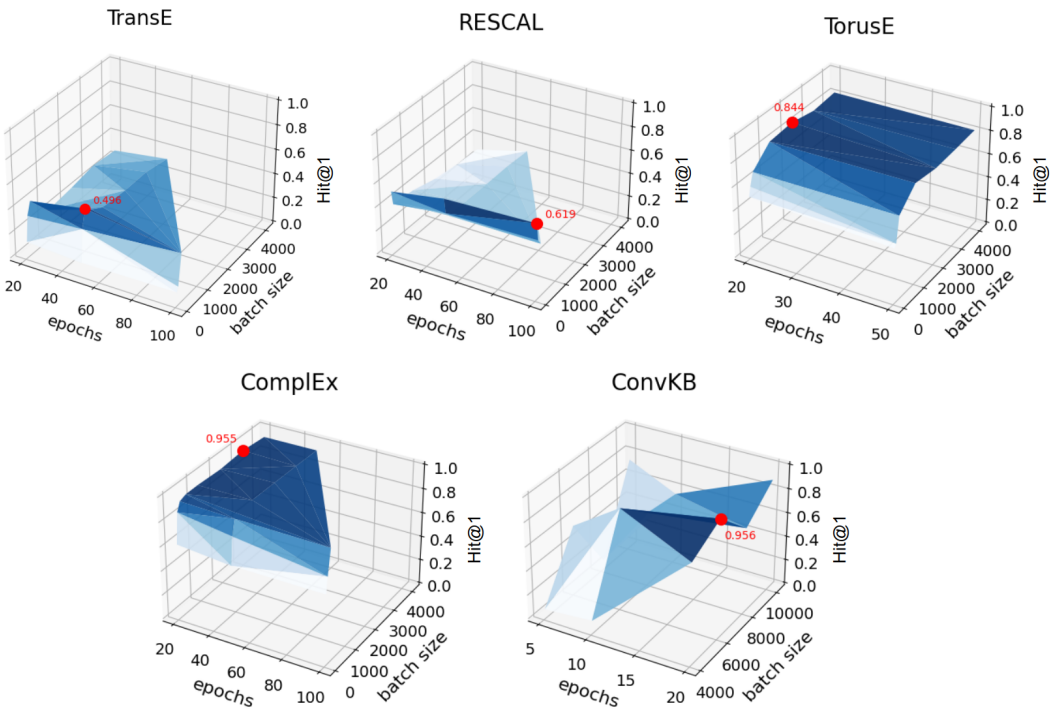


FIGURE 11 – *Performance of the different graph embedding methods on the CeKG in terms of the Hits@1 metric.* The x-axis represents the number of training epochs, the y-axis represents the batch size and the z-axis the Hit@1 score. The best performing model in each case is marked by a red dot.

Due to both the poor performance of RESCAL and its high training time, reaching up to nine hours in the worst case (due to its quadratic time complexity), we removed this method from the pool for further experiments.

Some methods seem relatively resistant to variations in batch size and epochs, such as TorusE and ComplEx, while others are more subject to those variations, especially ConvKB. This is also the case when varying other hyperparameters for this method, such as loss and number of filters (See Appendix - Hyperparameter search).

4.1.2 Method Stability

To account for the variability of machine learning methods, reproducing the obtained performance is necessary in order to make sure that our models behave as expected. For each method, we set the hyperparameters to those giving the best performance in the grid search, and we then evaluate their stability by training multiple models for each method.

ComplEx and TorusE are in line with their expected performance, while TransE performs overall slightly below. ConvKB, the best performing model on the initial grid search experiment, is surprisingly variable in its performance. This is consistent with the way the grid search does not seem to stabilize in any particular location.

4.1.3 Impact of data selection and structure

Including certain data types from the dataset can potentially improve model perfor-

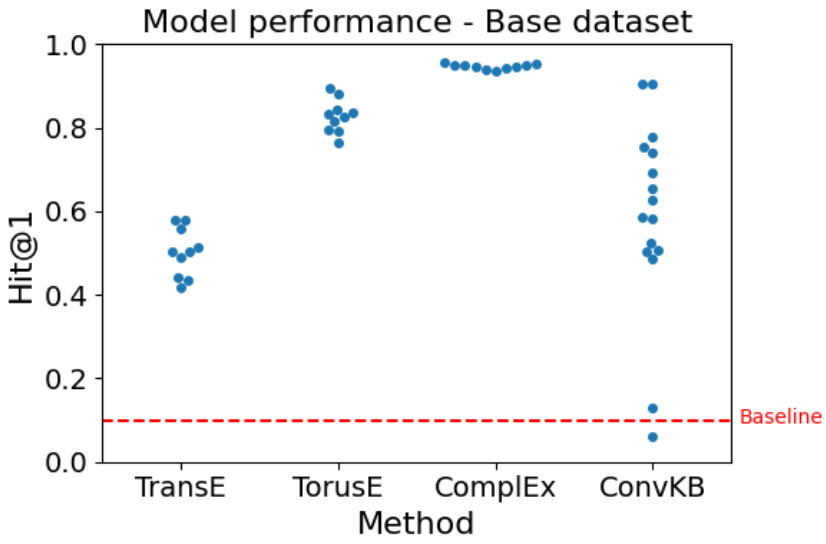


FIGURE 12 – *Reproduction of model performance on the base dataset.* A baseline corresponding to random performance (0.1) is indicated as a red line.

mance if the data is relevant to the task. However, this additional data may introduce noise in the form of irrelevant, hard to grasp or outright wrong information that could negatively impact the model's ability to learn meaningful patterns. Furthermore, modifying the data structure from a reified format can impact performance in the same way.

To explore those effects, we ran a few experiments in order to evaluate :

— Whether adding GO annotations and ontology increases embedding quality.

— Whether removing negative associations with phenotypes (i.e. experiments where a molecular entity was not found linked to a phenotype) is more noise than useful information, as in our data model, phenotype links and negative phenotype links are topologically similar.

— Whether reducing the distance between biological entities and concepts by changing the graph model to not follow the reification formalism increases embedding quality (This heavily impacts the graph topology).

Adding GO information does not seem to improve performance (Figure 13). However, we must keep a bias in mind : more link types means that ranking link types becomes harder.

Interestingly, removing negative phenotype associations improves the performance of TorusE, while reducing ComplEx's.

Changing the graph structure to a non reified format has a contrastive effect depending on the method : TorusE benefits from this format while TransE has worse results. ComplEx seems unaffected. We cannot say for sure concerning ConvKB, but it seems to lower the performance.

As such we wondered what kind of information do the models use to represent the graph as embeddings. We hope to explain the content of the embeddings by looking at whether they are affected more by topology, underlying node nature or something else by using a dimension reduction method : t-distributed stochastic neighbor embedding (t-SNE).

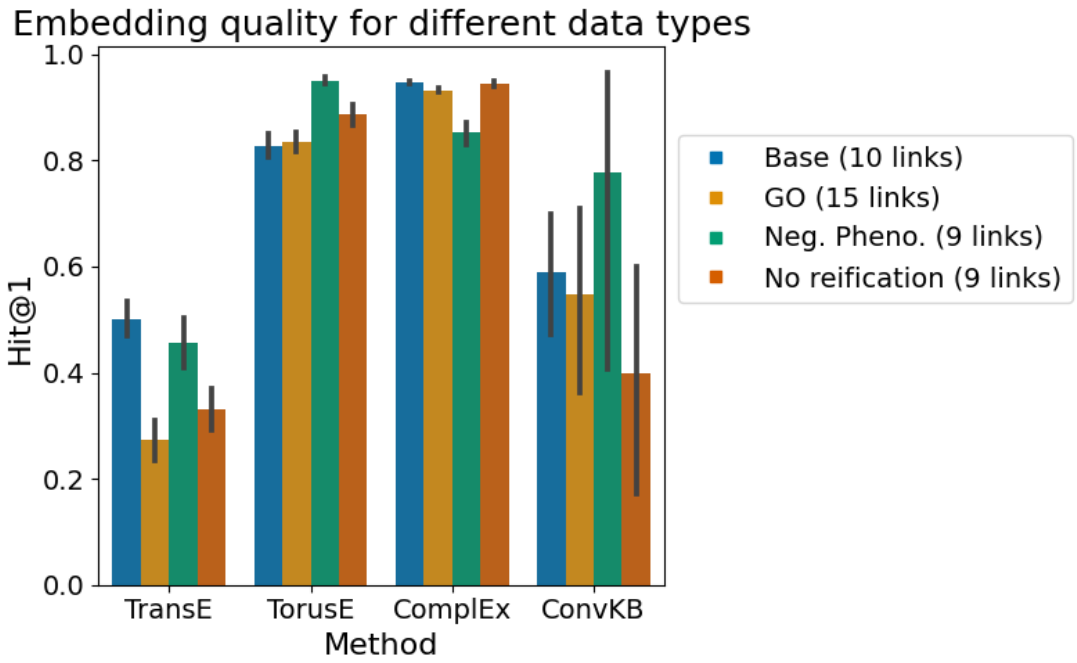


FIGURE 13 – *Data cleaning impact on model performance.* The dataset used in the previous parts, 'Base', is denoted in blue. The results when adding GO annotation and ontology are in yellow while removing negative phenotype associations is in green. Changing the graph structure from reified to non-reified is in brown. As adding information or changing the structure modifies the number of links, no baseline is shown.

4.2 Model explainability through embeddings projection

We reduce the dimension and project the embeddings obtained from the EMs using a t-SNE and see whether there are interesting patterns emerging in the data. This also serves as a control to check that nodes connected only a single time are present only in the training set.

These figures show that the train and test set have the same overall distribution, and that the node embeddings are well-defined for high connectivity nodes especially (Example : green nodes with 3 or 4 connexions), which consist of nearly all nodes that are not annotations (Figure 13a). Even then, some annotations are well-defined and form separated groups in Figure 13b, even among the same type (ex : phenotype annotation, in yellow). Interestingly, correlations can be observed on the left-side of Figure 13b between annotations from phenotypes (yellow) and development annotations (blue), suggesting associations between their underlying biological entity. We can also see in Figure 13a that nodes with a connectivity of 1 (red) are absent from the test set, as expected. While the repartition of node embeddings from Figure 13b seems to be inconsistent in the group containing labels (green) between the train and test set, this is due to the fact that displayed label nodes in the train set hides the underlying molecular entities (red) seen at this place in the test set.

Overall we obtain satisfactory results for two methods, ComplEx and TorusE, whose performance seem to plateau around the same range of values, indicating that the embedding quality may have reached a cap. Either way, we then proceed to make use of

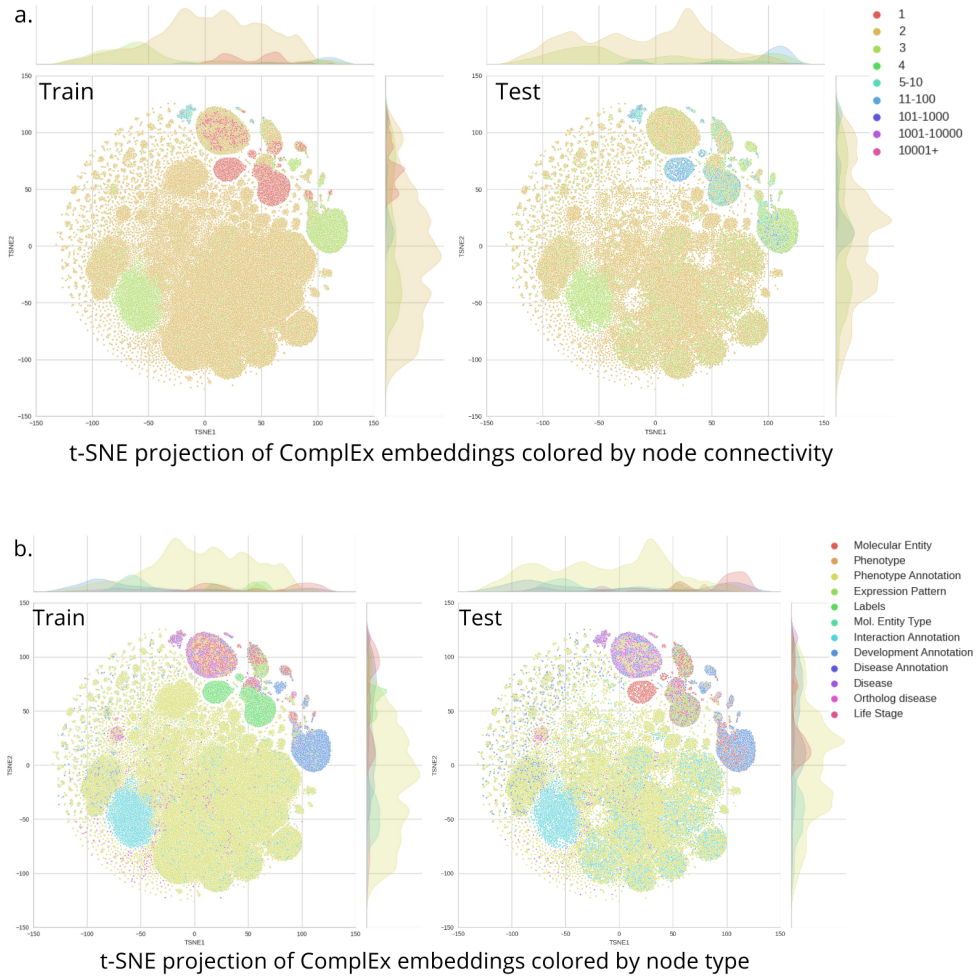


FIGURE 14 – *t-SNE projection of ComplEx node embeddings.* There are five times the number of nodes in the train set (left) compared to the test set (right), according to the 0.8 split ratio. This distribution is reflected on the density plots on the top and right of each figure. Due to the high density of points, datapoints can be hidden by those on top (See labels (green) in (c)). (a). Separation according to node connectivity. (b) Separation according to the nature of the node.

these embeddings to train a classifier to predict the existence of relation between nodes.

4.3 Comparison of Link Prediction Classifiers

For the following experiments, we used the best performing EM of each category to generate an embedding of all nodes from the test set, then concatenated the embeddings for the head and tail nodes of each triple, both true and false. From this dataset, we trained 11 types of classifiers using a k-fold approach, with k=10 folds and a train/test ratio of 0.8. See Methodology - Link prediction classifiers for a full breakdown.

We report here both previously mentioned metrics : Accuracy which measures the overall correctness of the model's predictions and Precision which minimizes false positives.

Ensemble methods built upon decision trees seem to work the best for our purposes,

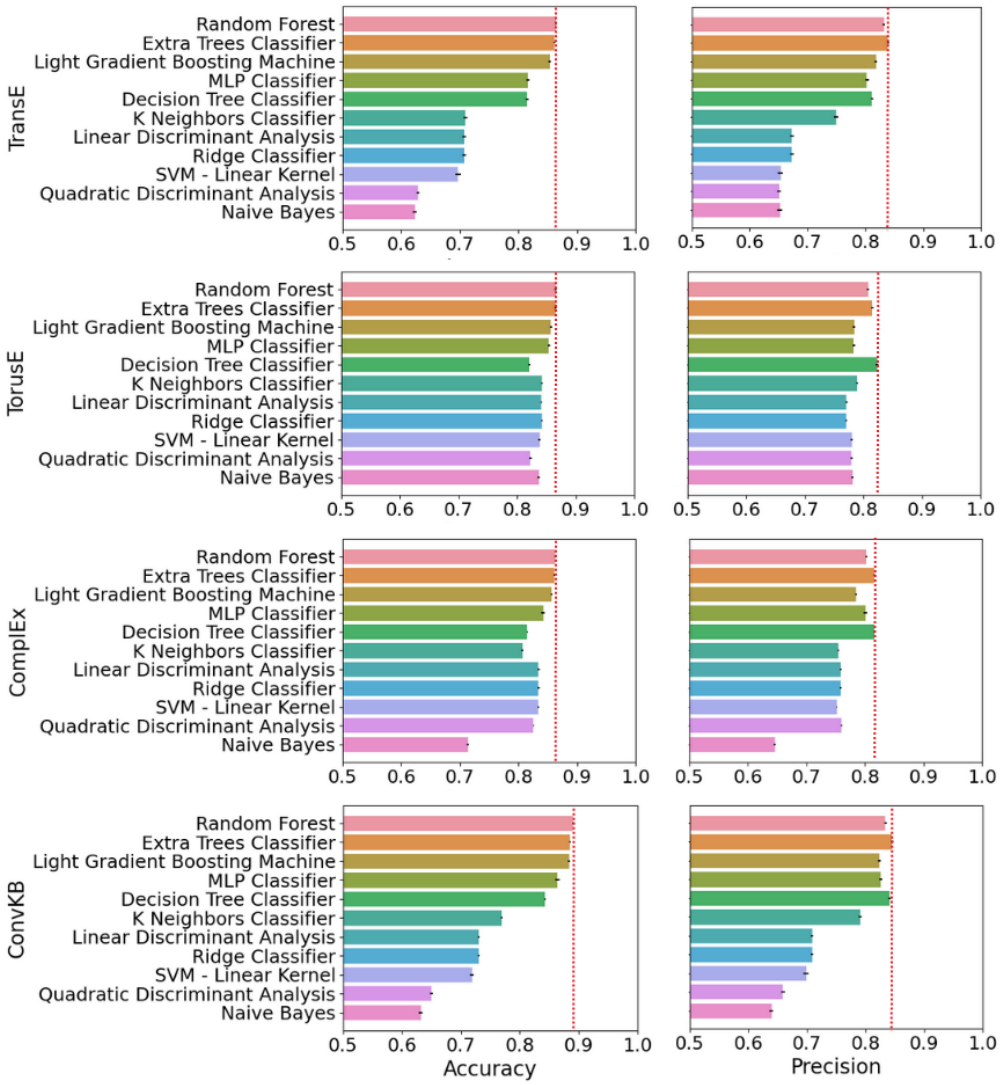


FIGURE 15 – *Accuracy and Precision of binary classifiers.* n=10 for each classifier type. Note that a random classifier would reach 0.5 in both metrics.

and seem to reach the same overall Accuracy (>0.85) except for ConvKB (near 0.9), while some decision trees models, especially Extra Trees, reach a high Precision (0.82˜0.85). The best overall model for our purposes seems to be the Extra Trees classifier, although it is closely followed by the Random Forest and Decision Tree classifiers.

Surprisingly, classifiers using TransE embeddings perform as well as classifiers using other embedding methods, despite the low Hit@1 obtained during EM relation prediction evaluation.

To better illustrate the practical implications of our findings, below is reported the confusion matrix obtained from the embeddings of a ComplEx algorithm using an Extra Trees classifier.

When looking at a confusion matrix, the usual goal is to maximize Accuracy, meaning the ratio of well classified samples. However, for our use case of building a list of

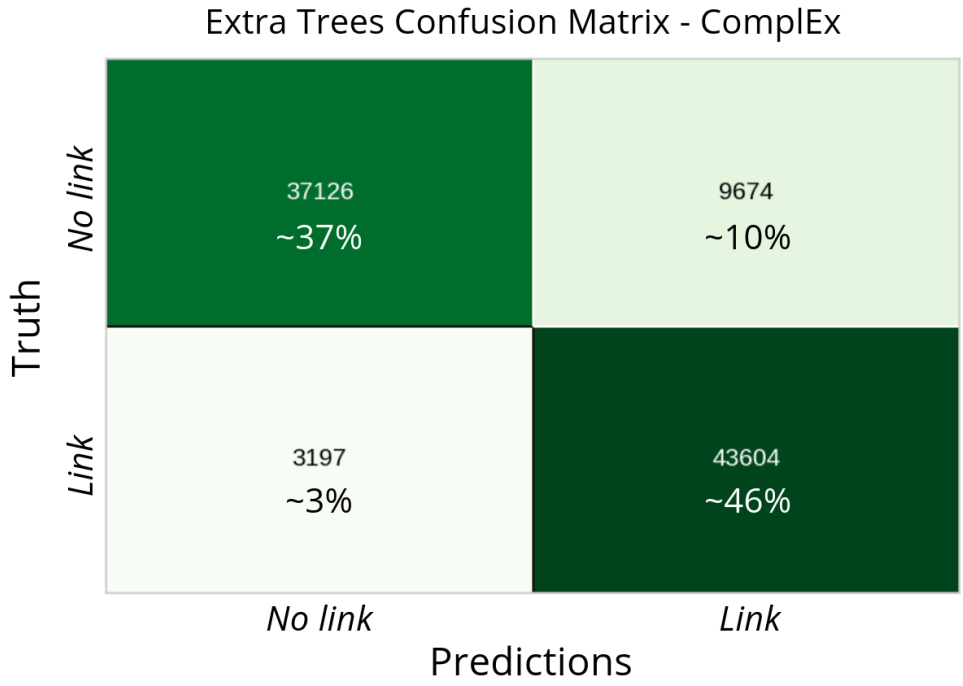


FIGURE 16 – *Confusion matrix of an Extra Trees classifier using ComplEx embeddings.*

candidate links we are also interested in the false positives, where the classifier infers a link that does not exist in the 'ground truth', since samples in this category are all potential candidates. In this example, approximately 10% of the nodes are classified as false positives.
False positives in the context of our project are a combination of both errors made by the model and silenced information, which is information that exists but is 'silenced' in the graph since there is no link to indicate it.
On the flipside, false negatives represent both actual links that are missed by the model plus erroneous information in the graph. Indeed, while false positives are silenced information, false negatives can be thought of as model error plus 'noise', which is erroneous information that has been put into the graph but does not truly exist.
We finally decided to see how changing the graph structure to a non-reified format changes the ability of the classifier to perform. Below are reported the corresponding performances.

Surprisingly, classifiers trained on the non-reified data structure show a lower performance of about 0.1 Accuracy and 0.05 Precision despite a reduced distance between node types of interest (including genes and phenotypes). Intermediary annotation nodes connecting biological entities are removed in this format, meaning there is less noise in this graph. However this removal also comes at the price of changing the topology of the graph.

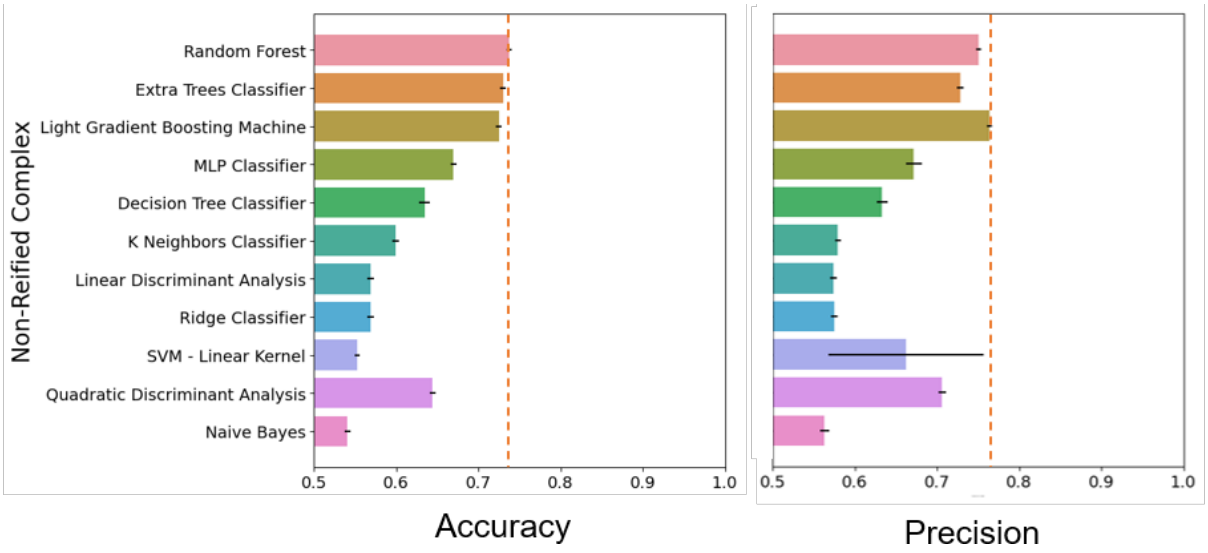


FIGURE 17 – *Performance of a ComplEx embedding model on a non-reified data structure.*

5 Discussion

5.1 Embedding Models

5.1.1 Hyperparameter Search

In the hyperparameter search phase, we evaluated the performance of different graph embedding methods on the CeKG. The models were trained with varying epochs and batch size, and the quality of the embeddings was measured using the Hits@1 metric. The results showed that most of the methods performed well in predicting link types, indicating their ability to capture meaningful information about the knowledge graph.

ComplEx and ConvKB emerged as the top-performing models in terms of Hit@1, while TransE and RESCAL exhibited comparatively poor performance. The poor results of RESCAL in encoding the KG may be attributed to the method operating under a Closed-World Assumption 2, meaning that it assumes that all unobserved triples are false, despite most graphs (and especially biological KGs) being incomplete. Because of both its relatively poor results and its high training time, RESCAL was excluded from further experiments.

The hyperparameter search provided insights into the relative performance of different methods, guiding us in selecting the best hyperparameters for subsequent evaluations.

5.1.2 Model stability

The stability of the selected graph embedding methods was assessed by training multiple models for each method, using the hyperparameters that yielded the best performance during the grid search.

ComplEx and TorusE exhibited consistent and stable performance, ranging between 0.93-0.95 and 0.76-0.83 respectively across the n=10 replications. This underlines their robustness and ability to consistently capture meaningful information from the CeKG.

TorusE seems particularly affected by changes to the topological aspect of the data, as shows the results when removing negative phenotypes and adopting a non-reified structure.

ConvKB showed significant variability in its performance. A simple explanation could be that the grid is too restricted for this particular method. Since ConvKB is a method whose weights are initialized from a trained TransE method, one alternative explanation for this inconsistent behavior could be that the underlying TransE model used to set the initial weights has a varying performance. However, when trying with two TransE models with a gap in performance of about 0.2 in terms of Hit@1, the results were not conclusive (Not Shown).
This variability could be instead attributed to the model's sensitivity to random initialization or the presence of local optima in the optimization landscape.

This highlights the importance of considering the reliability and consistency of the graph embedding methods when applying them to real-world applications.

5.2 Model explainability through embedding projection

In order to explain what information the embeddings are made of in order to understand which graph features are used by the embedding methods, we looked at the 2D projection of ComplEx embeddings using a t-SNE. The embeddings are well-separated according to both connectivity and underlying type. While both parameters are not totally independent since certain types often have a fixed connectivity, this suggests that the learned embeddings are representative of their associated nodes. We can see that for Annotations, their separation -while distinct- can sometimes become blurry, as seen with Phenotype Annotations (Fig14.b - yellow dots). This is in line with the fact that there is little informational content included in annotation nodes. Another explanation could be that two dimensions are not enough to represent some clusters in a visible way.

One interesting fact is that some phenotype annotations cluster with development associations, suggesting a link between their underlying biological entities. This leads to an interesting approach to predict links, which would consist of directly calculating the distance between two embeddings instead of using a machine learning approach. This approach would have the advantage of using less computational power.

Looking at the way embeddings cluster is a good way to see what features are taken into account by models. The effect of both changing methods and modifying the input data would most likely result in a change in how the embeddings are displayed, allowing to derive further insights into how each model works with the data at hand.

5.3 Impact of data selection and structure

The impact of data selection on model performance was investigated by adding GO annotations and ontology to the dataset in one experiment, and removing negative associations with phenotypes in a second experiment. The results show that adding GO annotations did not improve performance, suggesting at first glance that this type of data may not provide significant additional information and could even introduce noise

to the graph.

However, adding GO annotations increases the number of link types in the graph from 10 to 15. Since the Hit@1 metric first ranks the possible links depending on their likelihood, adding more link types makes the task harder, meaning that conserving the same performance may actually be an improvement to the representation of knowledge, rather than noise.

It is difficult to evaluate how much GO information contributes to the representation and in fact it is a possibility that the information is noise, but the models are good enough not to take it into account while keeping performance the same. This leads us to rethink our approach in evaluating the performance of the EMs. Indeed changing the number of link types in the graph leads to models whose performance can not be directly compared.

To get a consistent performance evaluation independant from variations in link number, we could replace the relation prediction approach by a node prediction approach. Instead of asking the model which type of link connects two nodes, we only give it the head node and the relation type. Then, we ask it to determine which is the most likely tail node by ranking all nodes. While this circumvents the problem of adding link types, this method does also have its drawbacks. Adding more nodes to the data makes it hard to directly compare models trained on different amounts of nodes. This also requires more computation time, as there are far more nodes than link types to rank in the graph. The literature also supports this direction : focusing on different measures for the evaluation of knowledge graph embeddings and intensive studies of how hyperparameters are selected are key to satisfactory model performance 4.

Interestingly, removing negative phenotype associations had contrasting effects on different models. TorusE demonstrated improved performance, indicating that the presence of negative associations might provide useful information for this method. Since the change of the graph topology to a non reified format increases the performance of TorusE as well, an hypothesis concerning this method is that it is sensible to changes in the graph structure because it may be paying more attention to topological features.
In contrast, ComplEx's performance decreased when negative phenotype associations were removed, suggesting that these associations may contain valuable topological information used by ComplEx that are not computed by TorusE. This shows that different models are able to capture different aspects of the informational content of the graph.

One way to leverage this would be to use not only a single embedding per node, but two embeddings from two different models for a single node during the classification task. While this would require more computation power, as this virtually doubles the amount of features, the added information could increase the accuracy of the binary classifier. The computation time required for training the classifier could also be reduced by first applying dimension reduction methods such as a PCA if the computation time is too high.

5.4 Comparison of Link Prediction Classifiers

The comparison of different types of binary classifiers demonstrates that ensemble

methods built upon decision trees yield the highest Accuracy and Precision among the various classifiers tested. Among the classifiers of this type, Extra Trees perform the overall best, reaching an Accuracy above 0.85 in most cases and a Precision ranging between 0.81 and 0.89. This result suggests that these ensemble methods effectively capture the complex patterns and interactions present in the embeddings generated by the EMs. The consistent Accuracy across different types of ensemble algorithms indicates that the specific choice of ensemble method may not significantly affect this metric. Precision however seems more affected by the type of classifier.

In the binary classifier context, ensemble classifiers using TransE embeddings reach an unexpectedly high Accuracy, matching that of classifiers using embeddings from other EMs. This is very unexpected, as TransE consistently scored lower than other EMs during embedding quality evaluation. This implies that the evaluation method (Relation Prediction with Hit@1) used to score EM performance may be inaccurate, or subject to an unknown bias. Instead of evaluating EM performance on a relation prediction task, we could instead evaluate it on a node prediction task, where the start node and relation are fixed, and the model is tasked to predict the end node. While computationally more expensive than the alternative (there are far more nodes than relations), this could be a better alternative to evaluate EMs.

The Accuracy of models trained on a non-reified graph structure shows a far lower Accuracy (about ~0.1 less), despite the simpler graph structure. This is most likely because of the very high connectivity of each node in this structure, which makes it hard for the model to learn meaningful topological patterns.

5.4.1 What is a good link classifier ?

Usually, in a binary classification task, a perfect Accuracy of 1.0 determines what is an ideal model. When looking at the confusion matrix of such a model, false positives and negatives amount to zero because it represents the error made by the model.

However, the ideal model in this project is rather counterintuitive. We do not want to have zero false positive and zero false negative, because while that would mean our model does not make mistakes, we have to keep in mind that the 'truth' represented by the graph is erroneous (noise) and incomplete (silence). This means that the ideal model actually makes mistakes in comparison to the actual Truth. As such there would be no way to predict 'silenced' interactions that exist in truth but are not in the graph.

Instead we want to find a percentage of false positives and false negatives that match the amount of incompleteness (silence) and errors (noise) in our graph, respectively. The idea behind this is that if we manage to reach this balance, then the false positives are very likely to be actual information missing from the graph rather than model errors while false negatives are likely to be noisy erroneous information that should be removed from the graph rather than model error.
What the percentage of noise and silence in the graph is, however, is not known. We may very well end up reaching a point where it becomes more interesting to assess the uncertainty of the data in our graph, rather than improving the performance of our models as a better, more accurate model can in fact be worse at modeling the true underlying

data.

5.5 Perspectives

Future work lies in two main directions at the moment. First is looking into other EM evaluation methods, namely the node prediction task previously mentioned, but also other metrics that can consider the overall ranking of predictions and not just the first, most likely one (Hit@1). A combination of both a node and relation prediction evaluation method could even be considered, to make sure that the evaluation takes into account both node and relation comprehension by the model.

The second direction would be to refine the input data by engineering the data types included to optimize our signal-to-noise ratio. This could be through investigating in depth each data type or adding new data types. Specifically, the initially excluded expression values in the form of gene coexpression determined with weighted correlation network analysis (WGCNA) could be a meaningful addition.

Further down the line, it would be worth trying methods able to assign weights on different relations as the current CeKG is made of only directed, unweighted links. A straightforward example of the underlying value of this improvement would be in how coexpressed genes are not all correlated with the same strength.
Other improvements could be made in estimating the noise and silence in the data, as discussed before, but also adding other data modalities, which could include more exotic information. This could be structural information on proteins, textual description embeddings or even embeddings of images.
Once a decent model has been established, portability could easily be done on other species, provided the required data is available.

6 Conclusion

In this master's thesis, we have developed a methodology for predicting biological relations in a CeKG by leveraging graph machine learning techniques. By combining heterogeneous data graphs and machine learning, we aimed to develop a tool which discovers new relations between biological entities to shed light on the underlying biological mechanisms linking genes and phenotypes in a manner that is resilient to both erroneous data in the graph (noise) and incompleteness of the data (silence).
This has implications for gene prioritization in functional knock-out RNAi experiments, potentially saving time and resources for biologists. Our approach involved three main steps. From the raw data available on a SPARQL endpoint, we first established a preprocessing pipeline using SPARQL queries to select parts of the raw data graph.

Then, in a second step we trained a graph EM to generate numerical representations of the informational content of nodes and relations in the CeKG. This allowed us to capture meaningful information from the complex and heterogeneous data, enabling the model to learn the underlying patterns and relationships.

We evaluated five graph embedding methods chosen from the literature according to performance, popularity and ease-of-use criteria before identifying the top-performing

models based on a relation prediction task performance.

In a third and last step, we trained a binary classifier using the embeddings generated by the graph EM. This classifier was designed to predict whether a relation exists between two chosen nodes, specifically focusing on gene-phenotype associations. We compared different types of classifiers and found that ensemble methods built upon decision trees yielded the highest Accuracies and Precisions, indicating their effectiveness in capturing the complex patterns and interactions present in the embeddings.

Our results demonstrated the potential of combining graph machine learning with heterogeneous data to both predict new biological relations in the CeKG (identifying silenced information), but also predict which data in the graph is erroneous (noise identification). To be certain of the validity of our relation predictions however, the only way is to verify them through biological experimentations.

This thesis contributes to the field of biology and bioinformatics by providing a framework for uncovering new biological relations, more specifically gene-phenotype associations, in the CeKG. As shown in this work, the combination of graph machine learning and heterogeneous data holds great potential for accelerating biological research. Our findings pave the way for future studies and applications in understanding the genetic basis of phenotypes and advancing biological research in silico.

7 Bibliography

1. Rossi, A., Firmani, D., Matinata, A., Merialdo, P. & Barbosa, D. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. ACM Trans. Knowl. Discov. Data 15, 1–49 (2021).

2. Ji, S., Pan, S., Cambria, E., Marttinen, P. & Yu, P. S. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. IEEE Trans. Neural Netw. Learn. Syst. 33, 494–514 (2022).

3. Wang, X. et al. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. IEEE Trans. Big Data 1–1 (2022) doi:10.1109/TBDATA.2022.3177455.

4. Bianchi, F., Rossiello, G., Costabello, L., Palmonari, M. & Minervini, P. Knowledge Graph Embeddings and Explainable AI. Preprint at https://doi.org/10.3233/SSW200011 (2020).

5. Davis, P. et al. WormBase in 2022—data, processes, and tools for analyzing Caenorhabditis elegans. Genetics 220, iyac003 (2022).

6. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J. & Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. in Advances in Neural Information Processing Systems vol. 26 (Curran Associates, Inc., 2013).

7. Ebisu, T. & Ichise, R. TorusE: Knowledge Graph Embedding on a Lie Group. Preprint at https://doi.org/10.48550/arXiv.1711.05435 (2017).

8. Nickel, M., Tresp, V. & Kriegel, H.-P. A three-way model for collective learning on multi-relational data. in Proceedings of the 28th International Conference on International Conference on Machine Learning 809–816 (Omnipress, 2011).

9. Nickel, M., Rosasco, L. & Poggio, T. Holographic Embeddings of Knowledge Graphs. Preprint at http://arxiv.org/abs/1510.04935 (2015).

10. Yang, B., Yih, W., He, X., Gao, J. & Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. Preprint at http://arxiv.org/abs/1412.6575 (2015).

11. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É. & Bouchard, G. Complex Embeddings for Simple Link Prediction. Preprint at http://arxiv.org/abs/1606.06357 (2016).

12. Nguyen, D. Q., Nguyen, T. D., Nguyen, D. Q. & Phung, D. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) 327–333 (2018). doi:10.18653/v1/N18-2053.

13. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Preprint at http://arxiv.org/abs/1810.04805 (2019).

14. Bi, Z. et al. Relphormer: Relational Graph Transformer for Knowledge Graph Representations. Preprint at http://arxiv.org/abs/2205.10852 (2022).

15. Pio-Lopez, L., Valdeolivas, A., Tichit, L., Remy, É. & Baudot, A. MultiVERSE: a multiplex and multiplex-heterogeneous network embedding approach. Sci. Rep. 11, 8794 (2021).

16. Dursun, C., Shimoyama, N., Shimoyama, M., Schläppi, M. & Bozdag, S. PhenoGeneRanker: A Tool for Gene Prioritization Using Complete Multiplex Heterogeneous Networks. in Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics 279–288 (ACM, 2019).

doi:10.1145/3307339.3342155.

17. Dursun, C., Kwitek, A. E. & Bozdag, S. PhenoGeneRanker: Gene and Phenotype Prioritization Using Multiplex Heterogeneous Networks. IEEE/ACM Trans. Comput. Biol. Bioinform. 19, 2950–2962 (2022).

18. Boschin, A. TorchKGE: Knowledge Graph Embedding in Python and PyTorch. Preprint at https://doi.org/10.48550/arXiv.2009.02963 (2020).

19. Wang, Z., Zhang, J., Feng, J. & Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. Proc. AAAI Conf. Artif. Intell. 28, (2014).

Annexes

While all relevant information is presented at a high-level in this thesis, all results can be found in their respective .CSV on the project's github.

7.1 Preliminary experiments on Embedding Models

For each method, the following hyperparameters were sampled at different points in addition to those presented in the corpus of the thesis. Usually only one or two variations were tested for each metric.

Table 1 – Embedding Methods and Parameters

Method	Embedding Size	Dissimilarity	Margin	Learning Rate
TransE	50 - 100 - 200	L1, L2	1	0.0001, 0.001
RESCAL	50 - 100	L1, L2	1	0.001, 0.0001
TorusE	50 - 100 - 200	torus_L1, torus_L2	1	0.001, 0.0001
ComplEx	50 - 100 - 200	L1, L2	0.5 - 1	0.001, 0.0001
ConvKB	50	L1, L2	1	0.01, 0.001, 0.0001

7.2 Preliminary experiments on Embedding Models

ConvKB being a convolutional neural network method, other relevant hyperparameters exist, like the number of convolutionnal filters. To get a better idea of how convKB behaves, the grid search for this method was extended to include two other hyperparameters : the loss functions margin and binary cross entropy as well as the number of convolution filters (Supplementary material - Figure 18.)

7.2.1 Bernoulli negative sampler

The Bernoulli negative sampler, introduced in 19, is a method that involves replacing either the head or tail entity in a triple with another entity chosen randomly. This choice of replacement is determined based on probabilities that consider the profiles of the relations involved in the triple. For a more comprehensive understanding of the approach, refer to the original paper.

7.2.2 Relation prediction on TorusE

Due to a problem in TorchKGE's pipeline, evaluation of the TorusE model does not work, as a parameter that is not defined in TorusE is called. An easy fix for this is to replace in torchkge/models/translation.py the line 765 :

```
candidates = candidates.expand(b_size, self.n_rel, self.rel_emb_dim)
```

by

```
candidates = candidates.expand(b_size, self.n_rel, self.emb_dim)
```

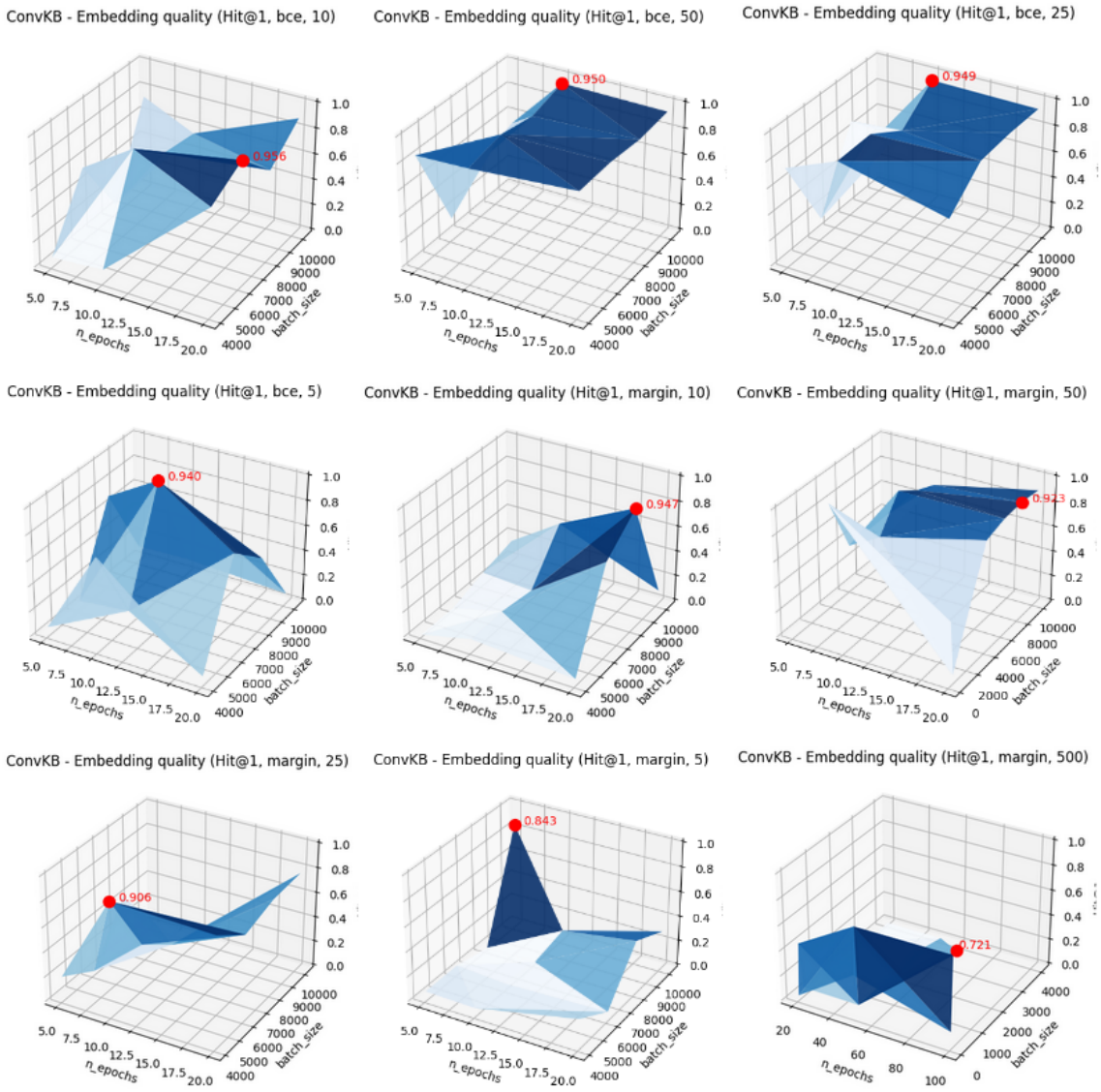


FIGURE 18 – *Grid search on ConvKB hyperparameters : Number of convolutional filters and loss function.*

ABSTRACT The study of the relationship between genes and phenotypes is of great importance in the field of biology. However, deriving those relations from the ever-growing amount of biological data and knowledge is becoming increasingly challenging for individuals. This complex and heterogeneous data can be modelized in computer-actionable graphs using frameworks like Semantic Web technologies. The present work aims at taking advantage of multiple data types as a whole to discover new relations between biological entities, like genes, phenotypes or diseases. We hypothesize that by combining heterogeneous data graphs and machine learning we can generate a tool resilient to both sparse and noisy data. In this context, the C. elegans model organism provides a rich source of data, with its well-studied genetics and vast array of available experimental data. Indeed, the WormBase database contains a wealth of information on C. elegans genes and phenotypes, including interactions between genes, diseases, and gene expression patterns that can be represented as a large heterogeneous knowledge graph. Our approach to analyzing this data is through the use of graph machine learning methods. We use the rich variety of data points present in the heterogeneous knowledge graph derived from the WormBase database to train both foundational and state-of-the-art machine learning algorithms to infer missing links in the knowledge graph in order to predict new gene-phenotype associations. This is a two-steps methodology. We first train a graph embedding model that can generate numerical representation of the nodes and relations. In a second step, we train a classifier that uses those embeddings to determine whether a relation exists between two chosen nodes. Our work has the potential to shed light on the underlying biological mechanisms linking genes and phenotypes, and to contribute to the acceleration of research not only in C. elegans, but a potentially wide range of organisms.

KEY WORDS : Heterogeneous Graph - Machine Learning - Link Prediction - Embedding