Authors: Ana Jimenez[1], María José Merino[1], Juan Parras[1], Santiago Zazo[1]
Affiliations: information processing and telecommunications center, universidad politécnica de madrid, etsi telecomunicación, madrid spain[1]
Orcid ids: 0009-0003-8679-4680[1], 0009-0002-6177-1071[1], 0000-0002-7028-3179[1], 0000-0001-9073-7927[1]
Contact e-mail: ana.jimenezb@upm.es
Journal: DrugRxiv
Review statement: The manuscript is currently under review and should be treated with discretion.
Links to data: https://github.com/hetio/hetionet
Funder(s): HORIZON EUROPE Health
Manuscript submission date: 10 October 2023
Keywords: Drug Repurposing, Heterogeneous Knowledge Graphs, Knowledge Graph Completion, Interpretability, Hetionet, Rule-based link prediction

EXPLAINABLE DRUG REPURPOSING VIA PATH-BASED KNOWLEDGE GRAPH COMPLETION

Ana Jiménez
Information Processing and Telecommunications Center
Universidad Politécnica de Madrid, ETSI Telecomunicación
Madrid, Spain
ana.jimenezb@upm.es

María José Merino
Information Processing and Telecommunications Center
Universidad Politécnica de Madrid, ETSI Telecomunicación
Madrid, Spain

Juan Parras
Information Processing and Telecommunications Center
Universidad Politécnica de Madrid, ETSI Telecomunicación
Madrid, Spain

Santiago Zazo
Information Processing and Telecommunications Center
Universidad Politécnica de Madrid, ETSI Telecomunicación
Madrid, Spain

September 26, 2023

ABSTRACT

Drug repurposing aims to find new therapeutic applications for existing drugs in the pharmaceutical market, leading to significant savings in time and cost. The use of artificial intelligence and knowledge graphs to propose repurposing candidates facilitates the process, as large amounts of data can be processed. However, it is important to pay attention to the explainability needed to validate the predictions. We propose a general architecture to understand several explainable methods for graph completion based on knowledge graphs and design our own architecture for drug repurposing. We present XG4Repo (eXplainable Graphs for Repurposing), a framework that takes advantage of the connectivity of any biomedical knowledge graph to link compounds to the diseases they can treat. Our method allows methapaths of different types and lengths, which are automatically generated and optimised based on data. XG4Repo focusses on providing meaningful explanations to the predictions, which are based on paths from compounds to diseases. These paths include nodes such as genes, pathways, side effects, or anatomies, so they provide information about the targets and other characteristics of the biomedical mechanism that link compounds and diseases. Paths make predictions interpretable for experts who can validate them and use them in further research on drug repurposing. We also describe three use cases where we analyse new uses for Epirubicin, Paclitaxel, and Predinisone and present the paths that support the predictions.

1 Introduction

Drug discovery is a time-consuming and high-cost process that involves several stages to obtain the approval of the authorities of a new drug. It takes 10-15 years and requires between $500 million and $2 billion. Moreover, approximately 90% of drugs fail in the early stages of development and toxicity testing and even among drugs that pass these steps, most fail due to side effects or adverse problems [1], [2].

Due to these limitations, the identification of new applications for existing drugs is a time-effective and cost-effective alternative, which is called drug repurposing. It allows the increase in treatment options for existing diseases and provides faster treatment for emerging diseases [2], [3].

The increasing amount of data available in recent years has led to the use of machine learning approaches for drug repurposing. This research focusses on drug repurposing based on biological knowledge graph datasets. Knowledge graphs are a set of nodes and edges that represent the relations between nodes. In the case of biological knowledge graphs, the nodes can be of different types, such as genes, compounds, side effects, diseases, etc. The goal of drug repurposing based on knowledge graphs is to discover links of type “treats" between entities of type compounds and diseases.

1.1 Related work

Drug repositioning is a complex task that includes several approaches. In the state-of-the-art drug repurposing based on knowledge graphs, the most extended methods are based on embeddings, which map the nodes to a low-dimensional representation that summarises their graph location and the structure of their neighbourhood. Models such as those developed in [4]–[9] are based on embeddings.

In [4], a model based on attention called MT-DTI is developed to identify drug target interactions using binding affinity scores. In [5], CoV-KGE, a deep learning model is used to generate a biomedical network and then applied embedding methods to find drug target interactions for COVID-19. In [6], the authors use embeddings, but also diffusion networks and proximity-based algorithms for drug repurposing.

These methods provide predictions, but do not include any information on why or how the prediction was made. In addition, most of these methods cannot capture multistep relations. Interpretability is very important in this field, so several models include different methods to explain the predictions. Most of them are based on paths that relate drugs and diseases through the nodes of the graph. These paths provide biological explanations for why the compound can treat a certain disease. They also use metapaths, which are sequences of types of nodes and relations, to obtain paths.

Some methods use a small set of metapaths selected by an expert to obtain paths used for prediction [10]–[12]. The predictions are always based on the metapaths that are known to be useful. Other methods do the opposite, evaluating every possible metapath [13], [14] that connects the compound to the disease. In the first case, the model is based on expert knowledge rather than data, and the second approach is computationally expensive.

Other methods use tools for graph analysis such as [15]. They built NeDRex, a platform for identifying subgraphs that represent the mechanisms of action of diseases that include genes and proteins, called disease modules. Then, disease modules are used to predict a list of drug candidates to treat a certain disease. They generated disease modules using network-based medical algorithms. However, the entities of the network and the relations between them are limited.

In [16], a framework for drug repurposing named Torchdrug is developed that includes several important tasks for drug discovery, such as biomedical knowledge graph reasoning. In this field, they provided benchmarks for embedding-based models for Hetionet. However, they evaluate the complete graph and do not focus on repurposing, which is what we do in this work.

Other researches develop hybrid architectures that make predictions based on embeddings or other non-interpretable methods and apply some technique to provide explanations [17]–[20]. In [11], KGML-xDTD is developed to predict repurposing candidates using embedding methods and random forest. Then, the model includes an actor-critic reinforcement learning approach to find paths between the drugs and the diseases that could explain the predictions. The agent was guided by demonstration paths, which are paths that can explain why a drug treats a disease. These paths were previously selected by experts.

MINERVA [21] is a reinforcement learning agent used for general link prediction. PoLo [22] is a modification of MINERVA which integrates the use of predefined rules for the task of drug repurposing.

1.2 Our contribution: XG4Repo

In this work we address the drug repurposing problem using knowledge graphs, and we propose XG4Repo (eXplainable Graphs for Repurposing), which achieves good performance as well as high quality explanations. We focus on the interpretability and limitations of the models found in the literature to design our framework. Our main contributions are:

- Present a general architecture for knowledge graph drug repurposing and show how several algorithms fit this description. These models follow different approaches, but we show that they all share similar principles.
- Design XG4Repo, our own drug repurposing strategy that focusses on interpretability. Our approach combines and optimises state-of-the-art algorithms for graph completion and presents the results in natural language so they can be easily understood for humans. We provide a ready-to-use framework to propose candidates for repurposing. Our proposal is able to find high-quality paths with an adjustable computational cost, and works with any heterogeneous graph. Our method allows methapaths of different types and lengths, which are automatically generated and optimised based on data.
- We validate our approach by presenting three use cases in which we show that the predictions are interpretable and reliable, in line with the state-of-the-art in current clinical studies.

2 Methods

2.1 Knowledge graph drug repurposing background

Graphs are collections of objects (nodes) and the set of interactions (edges) between pairs of these objects [23]. Knowledge graphs ($\mathcal{G}$) are a particular type of multirelational graph where the information is defined by a set of existing triples, including a head node (h), a tail node (t^*) and a relation (r) that links them:

$$(h, r, t^*) \in \mathcal{G} \tag{1}$$

Drug repurposing on knowledge graphs can be seen as a task of link prediction, where we ask the graph which diseases a certain compound treats. We can understand the problem as a query that has to be solved by the graph. The query is composed of a “compound" c as the head, and the relation “treats". The answer to this query is a disease d that can be treated with the compound, which is the tail of the triple.

$$(c, treats, d^*) \in \mathcal{G} \tag{2}$$

The problem can be formulated in terms of the probability of success of the compound over the disease d, where the objective is that the answer equals the tail of the triple $d = d^*$, and which is conditioned on the existing graph:

$$p(c, treats, d = d^* \mid \mathcal{G}) = p(d \mid \mathcal{G}, (c, treats)) = p(d \mid \mathcal{G}, q) \tag{3}$$

where $q = (c, treats)$ is the query.

2.2 Path-based drug repurposing

Path-based drug repurposing leverages the connectivity of the graph to predict the disease that can be treated with a certain drug and also to provide a biological explanation of the prediction. These methods provide paths, which are sequences of nodes and relations that start and the head node of the query, in this case the compound, and follow different relations and nodes to arrive at the candidate disease. Another important concept is the metapath, which is a sequence of types of nodes and types of relations. For example, in the path (Epirubicin $\xrightarrow{\text{upregulates}}$ Gene EGF $\xrightarrow{\text{regulates}}$ Gene BRAF $\xrightarrow{\text{is associated to}}$ Breast cancer) is a particularisation of the metapath (Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{regulates}}$ Gene $\xrightarrow{\text{is associated to}}$ Disease). Metapaths are also called rules in certain contexts.

To generate paths, a strategy is needed. This strategy can be represented by a policy μ, which indicates the node that should follow the current node on the path. Another method of obtaining paths is the use of rules z or metapaths that applied to the graph generate paths. This strategy conditionally characterises the probability of a cadidate disease as:

$$p(d \mid \mathcal{G}, q, \mu) \tag{4}$$

in the case of policies, and

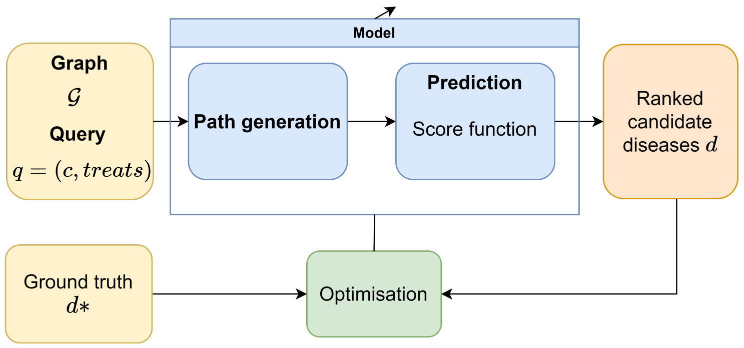


Figure 1: General architecture to train path-based drug repurposing models. The information is presented in the form of a knowledge graph composed of triples (in yellow). The input of the model is the graph and the query that has to be solved. The model (in blue) generates paths between compounds and different diseases. A score is computed to assess the quality of the paths and diseases that they propose. The model is optimised (green block) so that the ground truth diseases have the highest score.

$$p(d \mid \mathcal{G}, q, z) \tag{5}$$

in the case of rules.

The main reason to use paths is that predictions can be interpreted by healthcare professionals, which is necessary to validate candidate diseases for further research. Paths provide information about the side effects, targets, or anatomies involved in the biological mechanism.

We propose an architecture that generalises several methods for path-based graph completion, and we show the relation between them. We also present a mathematical formulation in Supplementary Data that unifies these models to understand them as a particularisation of the architecture described in this work.

The objective of the model is to predict diseases that can be treated by a certain compound using paths to connect the compound to the disease. Figure 1 shows the training process where the model is optimised to generate high-quality paths between the heads and tails of the queries. In the drug repurposing case, given a compound, the model generates paths that end in a set of candidate diseases. A score function evaluates the quality of the proposed diseases. The model learns to give high scores to diseases that are known to be treated with the compound. Therefore, other diseases that have high scores are good candidates for repurposing.

Taking into account the concepts of path generator and score function, the probability of the candidate for repurposing can be parameterised by θ and ω.

$$p(d \mid \mathcal{G}, q) \rightarrow p_{\omega,\theta}(d \mid \mathcal{G}, q) \tag{6}$$

where the path generation process is parameterised with θ and the score function with ω.

This expression can be decomposed into two processes: path generation and reasoning prediction. The objective of the path generator is to obtain the paths from components to diseases, and the reasoning predictor uses those paths to answer queries. As explained before, paths can be generated using policies or rules.

$$p_{\omega,\theta}(d \mid \mathcal{G}, q) = \sum_{\mu} p_{\omega}(d \mid \mathcal{G}, q, \mu) p_{\theta}(\mu \mid \mathcal{G}, q) \tag{7}$$

$$p_{\omega,\theta}(d \mid \mathcal{G}, q) = \sum_{z} p_{\omega}(d \mid \mathcal{G}, q, z) p_{\theta}(z \mid \mathcal{G}, q) \tag{8}$$

Several models fit this description with minor modifications. There are different approaches, some models are based on rules, while others use reinforcement learing techniques. These approaches fit into the general model that we propose because they are based on similar ideas. Further details on this formulation can be found in Supplementary Data.

2.2.1 Fixed path generator

There are several ways to generate paths, and the simplest is to use a fixed path generator. We can generate paths using a fixed generator following different principles, for example, random walks. This is the approach followed in AnyBURL [24]. AnyBURL is a bottom-up technique for efficiently learning logical rules from large knowledge graphs inspired by

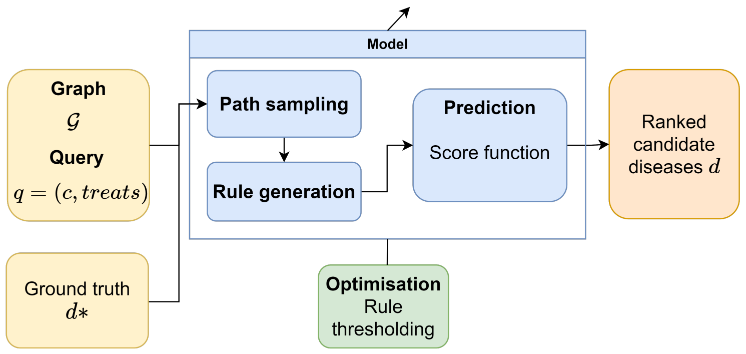


Figure 2: Description of AnyBURL based on our architecture. In this case, the input of the model is the whole graph (training set) including the ground truth. Paths are sampled based on random walk, and they are used to generate rules using a bottom-up approach. Then the confidence of the rule is computed, so only the rules with a confidence higher than a threshold are used for prediction. Rules are applied to the graph to obtain predictions which are ranked using the confidence of the rule.

classic bottom-up rule learning approaches. AnyBURL learns as many rules as possible by sampling random paths over a predetermined time interval. Then, each rule is evaluated according to the rate of correct positive predictions among all inferred predictions to obtain the confidence of the rule. The particularisation of the rules in the graph given a query generates paths between the compound and the candidate diseases. This is done in the path generator block in Figure 1.

Several rules generate the same candidate, so an aggregation of the score of each rule is required to find the final score of a candidate. This corresponds to the prediction block in Figure 1. There are three different approaches to determine the score of each candidate: Maximum score and Noisy-OR originally proposed with AnyBURL in [25], and Non-redundant Noisy-OR proposed as a framework called SAFRAN [26].

The optimisation in this case is very simple as the only task is to keep the rules that have a high enough confidence so that they can provide good predictions.

2.2.2 Reinforcement learning based path generator

The next step is to use path generators that can be updated based on data to learn the best way to traverse the graph to make predictions. Some methods use reinforcement learning to model the trajectory on the graph as a Markov Decision Process. Starting from the head node, the agent learns to walk to the tail node, choosing intermediate nodes step by step, taking into account the path history. Paths are generated based on policy μ, which is the strategy to traverse the graph to make good predictions.

This approach is followed in MINERVA [21] and its variants [22], [27]. In the drug repurposing context, the environment is the graph, and the possible actions are all the links the agent can choose from a certain node to the next. The objective of the agent is to move from a compound node to a disease node which is linked through the relations “treats". The state includes all nodes and relations travelled through to the current node, so the next action depends on the whole path. Moreover, it is necessary to define a reward function $R\left(\pi^n \mid q\right)$ that indicates whether the path π^n provides good predictions or not.

In this case, as shown in Figure 12, the main element is a policy generator that is trained to maximise long-term reward. Paths are sampled from the generator to connect the compound to the candidate diseases. The prediction and calculation of the score function are included in this block, because the score is directly related to the policy followed to generate the path as shown in the Supplementary Data.

The long-term reward over the policy is defined as:

$$\mathbb{E}_{\pi^n \sim \mu_\theta}\left[R\left(\pi^n \mid q\right)\right] \tag{9}$$

where the policy generator is parameterised with θ and π^n represent the paths sampled from policy μ_θ. After some manipulation found in the Supplementary material, the function that needs to be optimised is the following:

$$\frac{1}{N}\sum_{n=1}^{N} R\left(\pi^n \mid q\right) \log p_\theta(\pi^n \mid \mathcal{G}, q) \tag{10}$$

which averages the paths π^n sampled from policy μ_θ weighted by the reward $R\left(\pi^n \mid q\right)$ of the path. N is the number of paths.

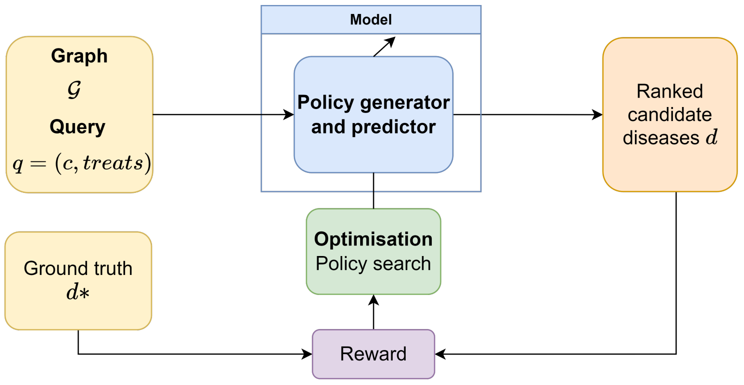


Figure 3: Description of MINERVA based on our architecture. The core of the algorithm is the policy generator, which is trained to obtain the best policy through the reward using policy search. Paths are sampled from the generator to obtain candidate diseases which are ranked according to the path that proposes them.

PoLo [22] is a modification of MINERVA that focusses on drug repurposing. The model includes a term in the reward related to how similar the path is to a set of manually crafted metapaths considered reliable for repurposing. This model relies on the existence of expert knowledge to improve the results of MINERVA.

2.2.3 Path generator using variational inference

Reasoning based on reinforcement learning has the problem that the action space is large and the reward is sparse, as few paths lead to the correct answer and a positive reward. For that reason, there are models that use rules as latent variables to make predictions. These rules (z) allow for the interpretability of the results and support the predictions. RNNLogic [28] follows this approach.

The model includes a rule generator and a reasoning predictor that apply the rules to propose candidate answers for the query, as shown in Figure 4. The rule generator returns a set of logic rules conditioned on the query, which are given to the reasoning predictor for query answering. The reasoning predictor computes the likelihood of the answer conditioned on the logic rules and the existing knowledge graph $\mathcal{G}$, $p_\omega(d \mid \mathcal{G}, q, z)$. At each training iteration, a few logic rules are sampled from the generator, which are fed into the reasoning predictor to try these rules for prediction. The distribution $p(d \mid \mathcal{G}, q)$ can be calculated according to Equation 27 as:

$$p_{w,\theta}(d \mid \mathcal{G}, q) = \sum_z p_w(d \mid \mathcal{G}, q, z) p_\theta(z \mid q) \tag{11}$$

which is the objective function that has to be optimised by the whole model. This task is divided as the generator and predictor use different optimisation algorithms, but both contribute to a common goal.

The generator $p_\theta(z \mid q)$ is updated using expectation maximisation (EM), and the optimisation of the predictor $p_w(d \mid \mathcal{G}, q, z)$ is based on maximum likelihood principles (MLE). Given the graph and the query, a set of rules is sampled and then used for the prediction $\hat{z} \sim p_\theta(z \mid q)$. Based on the results of the predictions, a score is calculated for each rule $\mathcal{H}(\hat{z}_i)$. It includes information from both the generation and prediction processes, so it is possible to know which are the high-quality rules for prediction $\hat{z}_I$.

To optimise the generator $p_\theta(z \mid q)$, a set of high-quality rules z_I is selected according to $\mathcal{H}(\hat{z}_I)$. For each data instance, the set of rules $\hat{z}_I$ is treated as part of the training data, and the generator is updated by maximising the logarithmic likelihood of $\hat{z}_I$. Moreover, $\mathcal{H}(\hat{z}_I)$ has information on the quality of the rules, so it can also be included in the generator optimisation in the form of weights of each rule:

$$\mathcal{H}(\hat{z}_I) \log p_\theta\left(\hat{z}_I \mid q\right) = \sum_{z_i \in \hat{z}_i} \mathcal{H}(\hat{z}_I) p_\theta\left(\hat{z}_i \mid q\right) \tag{12}$$

The function to be maximised is the average of the rules weighted by the score of the rules.

Rules generate paths that end in candidate diseases which are ranked according to a score computed based on trainable parameters related to the importance of rules and paths. The score measures the reliability of the predictions and can be used in the drug repurposing case study to evaluate and interpret candidates for repurposing.

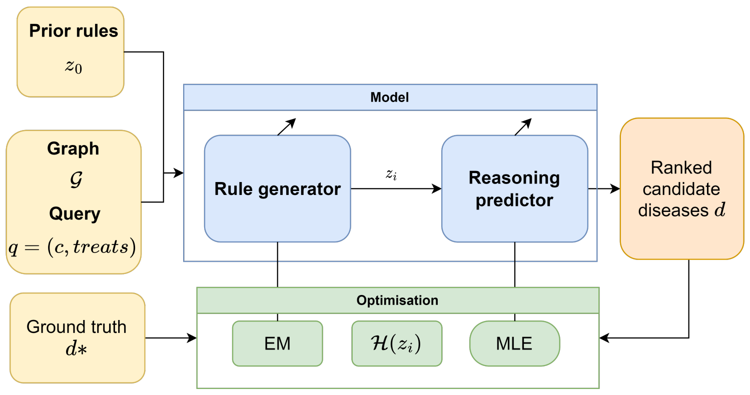


Figure 4: Description of RNNLogic based on our architecture. In addition to the graph and the query, there is another input which is a set of prior rules to initialize the generator. The model consists of a rule generator and a reasoning predictor. A set of rules is sampled and used for prediction. During training, the predictor is updated using maximum likelihood estimation (MLE). Combining information of the generation and the prediction, a score for each rule $\mathcal{H}(z_i)$ is computed and it is using during the training of the generator which is based on expectation maximisation.

2.3 XG4Repo

We have developed XG4Repo, a ready-to-use framework for computational drug repurposing using knowledge graphs. This framework is capable of predicting candidate diseases for repurposing and providing informative explanations to help a human expert in the research of new treatments.

Our proposal is a particularisation of the described architecture that combines state-of-the-art methods for graph completion and optimises them for drug repurposing. In Figure 11 we see the architecture of XG4Repo. The core of the framework is RNNLogic, because it provides informative rules and achieves good results. To initialise the generator, we have used the rule miner in AnyBURL, as the generated rules are good for prediction tasks. These rules need to be processed to be readable by the generator and filtered to remove those that are not general enough.

We have adapted the graph completion task to repurposing. In conventional graph completion, the model is trained to predict queries that include every type of relation. In drug repurposing, we are only interested in the relation “treats", so the model is specifically optimised to find diseases that can be treated by compounds. The training set only includes triples of “compound treats disease" but the whole graph can be traversed to find paths that include nodes and relations of any kind. Reducing the training set reduces computational time and resources, which is very interesting in the case of drug repurposing. As we see in Figure 5, “compound treats disease" (CtD) triples are differentiated from the rest of the graph.

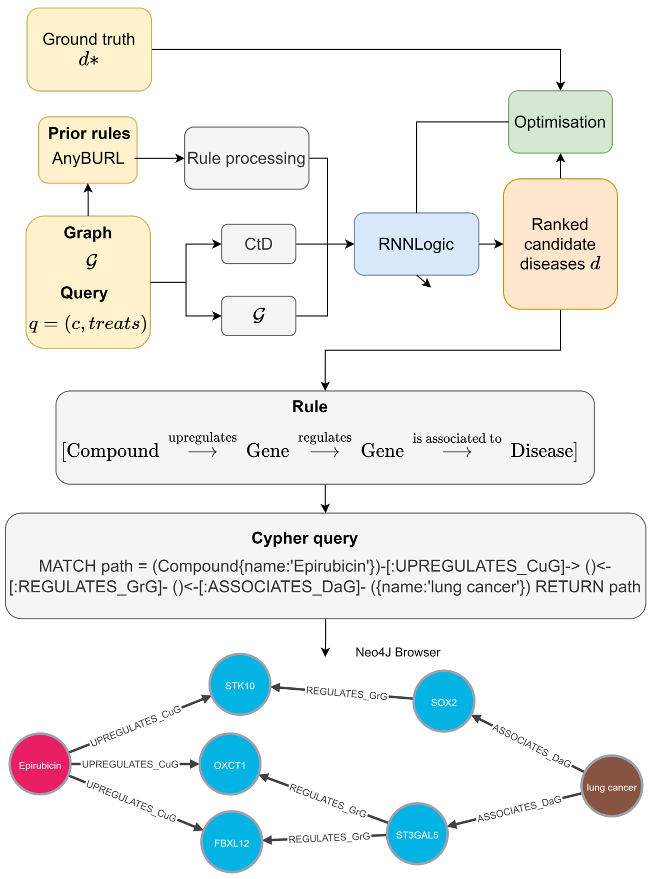
A key element in our design is a module for the interpretability of the results, where we can look for the predictions, the rules that support them, how important these rules are, and the paths that connect the compound to the disease. This is useful for those experts interested in the repurposing task, as they get predictions and explanations in natural language. Moreover, the code generates Cypher queries to obtain the paths generated by any specific rule on Hetionet. This is more efficient than storing every path generated by the rules. The code of XG4Repo is ready-to-use and available in XG4Repo.

2.4 Data

Hetionet [13] was developed within the Rephetio project with the aim of creating a knowledge graph suitable for different tasks related to drug repurposing, and is publicly available.

Among the 2,250,197 triplets that make up the knowledge graph, only 755 correspond to “compound treats disease". An 80-10-10% split was applied to divide the data set for training, testing and validation, respectively, obtaining the triplets. Of the 755 triplets of “compound treats disease", 598 are used to train the model, 82 for testing, and the remaining 75 triplets are used for validation.

The graph has been augmented with inverse relations, which, for each triplet, go from tail to head (t, r^{-1}, h). This adds flexibility to the rules and allows more connections between the nodes.



$[\text{Compound} \xrightarrow{\text{upregulates}} \text{Gene} \xrightarrow{\text{regulates}} \text{Gene} \xrightarrow{\text{is associated to}} \text{Disease}]$

```
MATCH path = (Compound{name:'Epirubicin'})-[:UPREGULATES_CuG]-> ()<-
[:REGULATES_GrG]- ()<-[:ASSOCIATES_DaG]- ({name:'lung cancer'}) RETURN path
```

Figure 5: Description of XG4Repo architecture. The first step of the process is to generate a set of prior rules using AnyBURL rule miner. These rules are processed and used as priors in the generators. The model is trained using only the triples “compound treats disease" so the computational complexity is reduced. Once the predictions are made, the rules and corresponding scores are stored in natural language, so they can be easily understood. Moreover, our framework can generate Cypher queries to obtain the paths in Hetionet given the rules. This adds interpretability to the predictions without adding extra storage requirements.

2.5 Evaluation metrics

Once a model has been trained, it has to be evaluated. The output of the model is a score for each possible answer for the test. During the test, we check that the disease of the triple being evaluated (d^*) receives a high score. Candidate diseases are ordered by decreasing score and the rank is defined as the position of the disease of the ground truth (d^*) in the list of candidates.

Based on the rank, several metrics are computed, which aggregate in a single number the performance of the model [29]. In this work, we have evaluated the models using mean reciprocal rank (MRR), Hits@1, Hits@3 and Hits@10.

The metrics calculated in this research are filtered as described in [29]. Moreover, binomial proportion confidence intervals are applied to compare the performance of the models as the size of the test set does not have enough samples to use the Gaussian approximation. It provides an interval estimate of a success probability p when only the number of experiments n and the number of successes n_s are known.

3 Results

We have trained several path-based graph completion models for Hetionet. The models being compared are XG4Repo, which represents our approach, MINERVA as a representation of reinforcement learning-based methods, and AnyBURL-based methods. For AnyBURL, we test three prediction strategies: Maximum score, Noisy-OR and SAFRAN. In the case of MINERVA and XG4Repo, we have trained only “drug treats disease" triples. For models based on AnyBURL,

Table 1: Comparison of the test results on “compound treats disease" on Hetionet using explainable methods. We include the results reported by PoLo [22] for comparison.

Method	MRR	Hits@1	Hits@3	Hits@10
PoLo	0.402	0.314	0.428	0.609
PoLo (pruned)	0.430	0.337	0.47	0.641
AnyBURL(maxscore)	0.520	0.390	0.573	0.817
AnyBURL(noisy-OR)	0.511	0.366	0.598	0.805
SAFRAN	0.563	0.439	0.598	0.793
MINERVA	0.359	0.244	0.378	0.622
XG4Repo	**0.612**	**0.488**	**0.671**	**0.890**

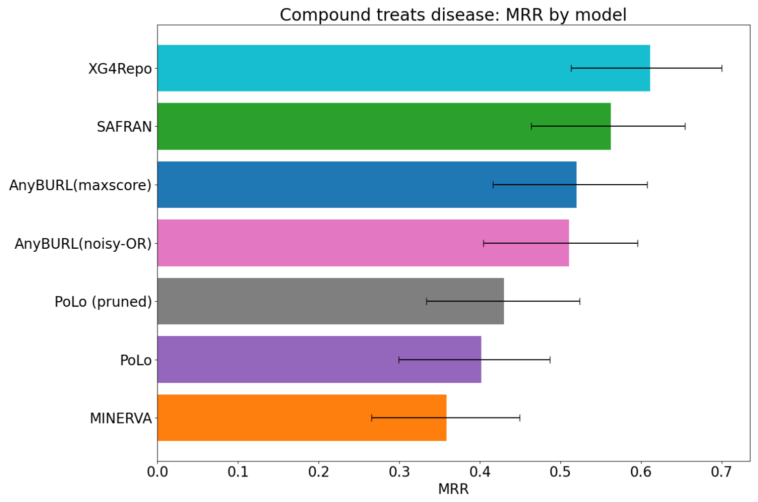


Figure 6: Comparison of the MRR of different models for “compound treats disease" in Hetionet. The confidence intervals at 90% are included. Rule-based models work better than reinforcement learning. Due to the small test set, confidence intervals for rule-based models overlap, so it is not possible to clearly identify the best performing one.

we have trained over every relation and then filtered test triples for evaluation, so test samples are the same in all cases. In Table 1, we present test metrics for models when the path length is set to three in all cases for comparison. For XG4Repo, 100 rules have been sampled from the generator.

We include the results reported in PoLo [22], because as far as we know, this is the only work that provides results for “compound treats disease" on Hetionet. We see that XG4Repo obtains better metrics under the same experimental conditions.

In Figure 6, we show the MRR of each model including 90% confidence interval. As the test set only includes 82 triples, the confidence intervals are large, so for most models, they overlap. Confidence intervals can be expected to narrow in larger knowledge graphs. For that reason, we propose XG4Repo as a promising tool to propose repurposing candidates and to provide meaningful explanations about the predictions. We can see that XG4Repo is clearly better than models based on reinforcement learning models, as it can generate more variate paths that lead to different candidates. We show the performance of the model and the interpretability of the predictions using three use cases of repurposing.

3.1 Use cases

In this section, we present three use cases of repurposing using the framework we have developed. The goal is to obtain diseases that can be treated with Epirubicin, Paclitaxel, and Prednisone using the methods explained previously. We include the predictions of AnyBURL-based models and MINERVA to support consistency in the predictions of our framework, as in most cases different models propose the same candidates.

The rules provided in this section are generated by XG4Repo and have a length of three steps. Paths provide explanations for the predictions and include the score by which the rule contributes to the prediction.

Table 2: Top 10 diseases predicted by each model for the query “Epirubicin treats disease". Each disease is predicted in a different position (rank) for different models. Diseases are ordered by XG4Repo results. Notice that in MINERVA, several paths can lead to the same node in different realisations. In bold, those diseases that are in the test set and, therefore, are true answers of the query.

Disease	AnyBURL maxscore	AnyBURL noisy-OR	SAFRAN	MINERVA	XG4Repo
Breast cancer	1	1	3	8, 9	1
Lung cancer [30]	2	2	2		2
Sarcoma	3	6	6		3
Kidney cancer	5	4	1		4
Muscle cancer [31]	4	5	5		5
Bone cancer	6	7	4		6
Melanoma [32]					7
Lymphatic system cancer [33]			8		8
Germ cell cancer [34]	9		10		9
Coronary artery disease					10
Hypertension		3			
Colon cancer [35]	7	10	7		
Multiple sclerosis	8	9	9		
Brain cancer [36]	10				
Asthma		8			
Epilepsy				1, 2, 5, 6, 10	
Osteoporosis				3	
Atopic dermatitis				4	

We also include some references to show that there are research and clinical trials that use drugs to treat diseases that have been proposed by the model. This shows that our framework can be a useful tool for healthcare professionals, as it is capable of handling larger amounts of data and coming to the same conclusions as them. It is an interesting first approximation for the processing of large datasets that has to be validated by further research.

3.1.1 Epirubicin

Epirubicin is a chemotherapy drug that is used to treat various types of cancer. Epirubicin treats 14 types of cancer according to Hetionet. The test set includes breast cancer, bone cancer, sarcoma, and uterine cancer, and the rest of the diseases are used for training.

In Table 2, we show the test results for different models and include the position of the disease in the prediction (rank). We see that the triples in the test set (in bold), those that we know to be true, are proposed as the first candidates for repurposing for every model except for MINERVA.

- Epirubicin treats breast cancer

All the models propose breast cancer as a candidate disease to be treated by Epirubicin. We already know that this prediction is true, as it is in the test set. The models can effectively identify those diseases that could improve with the use of the drug. In Table 3, we see the most important rules for this prediction according to XG4Repo and a path length of 3. These metapaths are expressive and include useful information about the targets of the disease. Most of them match the metapaths found relevant in [13]. The score of the rule shows the importance of the rule for prediction.

We do know that the relation treats exists between Epirubicin and breast cancer and have presented the rules that show the mechanisms that explain it. Moreover, we can see the paths to identify the nodes that relate the query and the prediction. In Figure 7 we see some paths that follow the rule: [Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is expressed by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]. We also provide Cypher queries to explore all these paths in Neo4J browser along with the code.

- Epirubicin treats lung cancer

Most models predict that lung cancer can be treated with Epirubicin in second position. This disease is not included in Hetionet, so it is a candidate. DrugBank [30] is an online free-to-access database that contains information on drugs and drug targets. In [30], DrugBank includes Epirubicin as treatment for Non-Small Cell Lung Carcinoma and Small Cell Lung Cancer (SCLC). Then, the models has been able to predict a treatment for a disease that healthcare community has

Table 3: Top rules for Epirubicin treats breast cancer and the corresponding scores.

Score	Rule
1,793	[Compound $\xrightarrow{\text{causes}}$ Side effect $\xrightarrow{\text{is caused by}}$ Compound $\xrightarrow{\text{treats}}$ Disease]
1,007	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is expressed by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]
636	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{regulates}}$ Gene $\xrightarrow{\text{is associated to}}$ Disease]
412	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is upregulated by}}$ Compound $\xrightarrow{\text{treats}}$ Disease]
378	[Compound $\xrightarrow{\text{treats}}$ Disease $\xrightarrow{\text{associates}}$ Gene $\xrightarrow{\text{is associated to}}$ Disease]
202	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is downregulated by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]
200	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is upregulated by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]

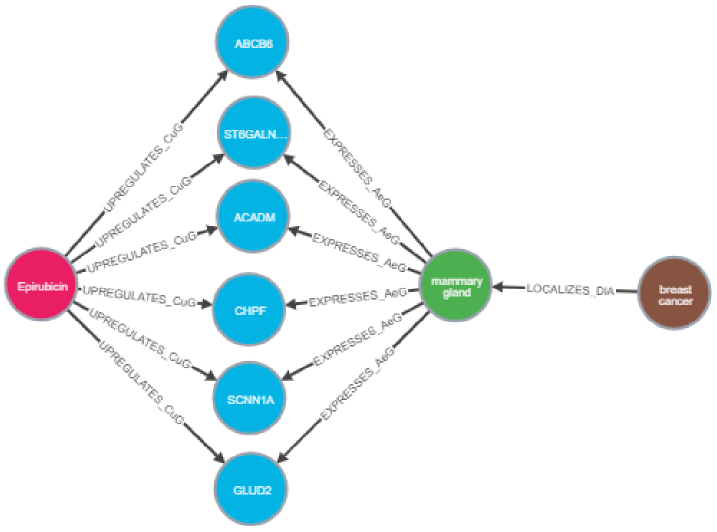


Figure 7: Set of paths that represent the triple Epirubicin treats breast cancer following the metapath [Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is expressed by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]. The number of nodes has been limited to facilitate visualization.

Table 4: Top rules for Epirubicin treats lung cancer and the corresponding scores.

Score	Rule
1,208	[Compound $\xrightarrow{\text{causes}}$ Side effect $\xrightarrow{\text{is caused by}}$ Compound $\xrightarrow{\text{treats}}$ Disease]
713	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is expressed by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]
340	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is upregulated by}}$ Compound $\xrightarrow{\text{treats}}$ Disease]
302	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{regulates}}$ Gene $\xrightarrow{\text{is associated to}}$ Disease]
259	[Compound $\xrightarrow{\text{treats}}$ Disease $\xrightarrow{\text{associates}}$ Gene $\xrightarrow{\text{is associated to}}$ Disease]
152	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is upregulated by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]
137	[Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is downregulated by}}$ Anatomy $\xrightarrow{\text{is localized to}}$ Disease]

accepted, even though it is not included in the knowledge graph used for training. We also include the most important rules for this prediction Table 4 and some paths in Figure 8.

Furthermore, actual applications of existing drugs are also published in [30], so from there we identified that muscle cancer, colon cancer, and germ cell cancer are being treated with Epirubicin. With respect to muscle cancer, it is specified in DrugBank as soft tissue sarcoma [31]. In addition, Epirubicin has actually been proven to be evaluated for colorectal cancer [35], being colon cancer predicted by AnyBURL-based methods. Similarly, the germ cell cancer

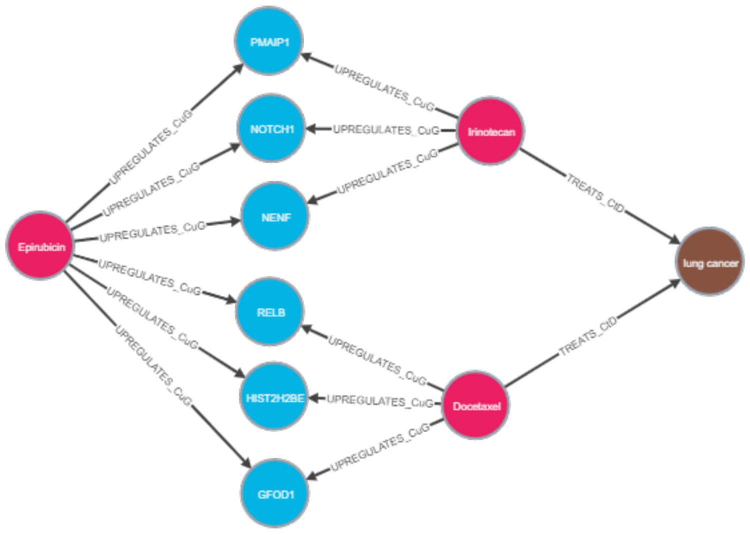


Figure 8: Set of paths that represent the triple “Epirubicin treats lung cancer" following the metapath [Compound $\xrightarrow{\text{upregulates}}$ Gene $\xrightarrow{\text{is upregulated by}}$ Compound $\xrightarrow{\text{treats}}$ Disease]. The number of nodes has been limited to facilitate visualization.

inferred by AnyBURL maximum score, SAFRAN and XG4Repo, is the general name of a type of cancer that develops mainly in the ovary or testicle, being the ovarian cancer actually treated with Epirubicin [34].

Moreover, in [30] the finalised and active clinical trials can be found. For lymphatic system cancer also known as lymphoma, which is proposed as a candidate by SAFRAN and XG4Repo, different studies have been carried out focussing mainly on determining its effectiveness in combination with other drugs. For example, a recent study published in November 2022 [33] aims to evaluate the efficacy and safety of Camrelizumab combined with Epirubicin, Vincristine and Dacarbazine to treat patients with advanced classical Hodgkin’s lymphoma. They obtain an Objective Response Rate (ORR) of 100%, which means that 100% of the study patients had a partial and complete response within the study period.

Researchers are also studying the use of Epirubicin to treat melanoma [32] in combination with other drugs.

For the rest of the diseases, current evidence of treatment has not been found in the literature. However, since these methods inferred these diseases, they could be potential candidates for diseases that could be treated with Epirubicin, providing a starting point for research.

3.1.2 Paclitaxel

Paclitaxel is a taxoid chemotherapeutic agent used as first-line and subsequent therapy for the treatment of advanced carcinoma of the ovary [37]. As shown in Table 5, all models except MINERVA have been able to predict ovarian cancer as a disease to be treated. In particular, XG4Repo proposes ovarian cancer as the first candidate.

In DrugBank [30], we found that urinary bladder cancer, pancreatic cancer, testicular cancer, melanoma, head and neck cancer, and sarcoma are being treated with Paclitaxel, in combination with other drugs. Moreover, we have found in [30] clinical trials that study the treatment of hematologic cancer [40], stomach cancer [38], prostate cancer [39], psoriasis [43], esophageal cancer [41] and colon cancer [44] with Paclitaxel. Paclitaxel has also been associated with the treatment of pulmonar hypertension [42].

For the rest of the diseases, no evidence of treatment or clinical trials has been found yet.

3.1.3 Prednisone

Prednisone is a corticosteroid used to treat inflammation or immune-mediated reactions and to treat endocrine or neoplastic diseases [30]. Ulcerative colitis, hematologic cancer, atopic dermatitis, and chronic obstructive pulmonary disease can be treated with Prednisone and are included in the test set. All models have been able to predict ulcerative colitis as a candidate and most of them hematologic cancer. XG4Repo has been able to identify chronic obstructive pulmonary disease as a candidate.

Table 5: Top 10 diseases predicted by each model for the query “Paclitaxel treats disease". Each disease is predicted in a different position (rank) for different models. Diseases are ordered by XG4Repo results. Notice that in MINERVA, several paths can lead to the same node on different realisations. In bold, those diseases that are in the test set and therefore are true answers of the query.

Disease	AnyBURL maxscore	AnyBURL noisy-OR	SAFRAN	MINERVA	XG4Repo
Ovarian cancer [37]	1	1	1		1
Pancreatic cancer [30]	3	3	4		2
Melanoma [30]	2	2	5		3
Stomach cancer [38]	4	4	3		4
Prostate cancer [39]	5	5	2		5
Hematologic cancer [40]		7	7	1, 2, 3, 8, 9, 10	6
Head and neck cancer [30]	6	6	6		7
Esophageal cancer [41]					8
Urinary bladder cancer [30]	9	9			9
Testicular cancer [30]	7		10		10
Hypertension [42]		8		6, 7	
Psoriasis [43]			8		
Colon cancer [44]	8	10			
Epilepsy				4	
Sarcoma	10				
Osteoporosis				5	

Table 6: Top 10 diseases predicted by each model for the query “Prednisone treats disease". Each disease is predicted in a different position (rank) for different models. Diseases are ordered by XG4Repo results. Notice that in MINERVA, several paths can lead to the same node on different realisations. In bold, those diseases that are in the test set and therefore are true answers of the query.

Disease	AnyBURL maxscore	AnyBURL noisy-OR	SAFRAN	MINERVA	XG4Repo
Ulcerative colitis	2	1	1	1, 3, 4, 6, 7, 8, 9, 10	1
Atopic dermatitis					2
Allergic rhinitis [30]	1	2	2	5	3
Chronic obstructive pulmonary disease					4
Hematologic cancer	3	3	3		5
Amyotrophic Lateral Sclerosis [45]					6
Leprosy [46]					7
Bone cancer					8
Malaria					9
Primary biliary cholangitis					10
Osteoporosis [47]	10	7	7		
Lung cancer [48]		8	10		
Breast cancer [49]	4	5	4		
Hypertension [50]	8	4	5		
Coronary artery disease [51]	9	6	6		
Dilated cardiomyopathy			8		
Kidney cancer			9		
Epilepsy [52]				2	
Colon cancer	6				
Urinary bladder cancer [52]	7	9			
Testicular cancer [53]	5	10			

Several models predict osteoporosis as a candidate; however, osteoporosis is a side effect of Prednisone [47]. As found in [13], it is possible that metapaths find contraindications to the diseases, so it is always necessary to study the predictions before starting clinical trials.

We have found clinical trials using Prednisone for lung cancer [48], breast cancer [49], testicular germ cell cancer [53], epilepsy in children [52], leprosy [46] and amyotrophic lateral sclerosis [45].

In [30], they propose Prednisone as a treatment for allergic rhinitis. In the case of hypertension, there are studies that relate the impact of Prednisone on this disease, but mainly in a negative way [50]. In [13], they already found that some of the predictions made were contraindications to the disease, as the included relations are too general. The relation of Prednisone and coronary artery disease has also been studied [51]. The use of Prednisone to treat coronary artery disease has been studied in the past [54], although it has not been used in general patients. This shows that our tool can make proposals similar to those made by a healthcare professional. Some studies also propose Prednisone to treat urinary bladder cancer [52].

We can perform this analysis with any other drug present in the graph and obtain the rules and paths that support these predictions.

4 Conclusion

In this work, we propose a general architecture to unify the process of graph-completion methods using paths. These methods are explainable, which is particularly relevant in the drug repurposing context. Moreover, they have good performance, which makes them trustworthy. We have analysed how some methods proposed in the literature fit our architecture and show that they are different approaches to complete the same stages of the process.

We have designed XG4Repo, a framework for drug repurposing using knowledge graphs that predict diseases that can be treated with a given compound. Along with the prediction, the model provides the rules that support the prediction and the importance of the rule. This step is necessary so that researchers can validate the prediction through the biological mechanism of action.

The results are presented for Hetionet, but the model can be trained on different knowledge graphs that include examples “compound treats disease". Using other knowledge graphs could lead to different but relevant predictions.

We have included three use cases to show that the model is able to propose candidates similar to those proposed by humans. This is important because the objective of these tools is not to replace research, but to analyse large quantities of data in a short amount of time. Therefore, it is possible to accelerate the first stages of drug repurposing.

Key points

- We propose a general architecture for explainable knowledge graph completion methods and describe several algorithms in these terms.
- We design XG4Repo, a ready-to-use framework for drug repurposing. The framework focusses on interpretability for the use of the predictions in further research.
- We present three use cases to show how the framework is used and the informative power of the rules that support the predictions.

5 Competing interests

No competing interest is declared.

6 Data availability

The dataset used in this project is Hetionet [13], which was developed within the Rephetio project and is publicly available. The framework developed in this work is available in XG4Repo.

7 Author contributions statement

A.J., M.J.M., S.Z. and J.P. conceived this study. A.J, J.P. and S.Z. designed the general architecture and M.J.M. developed the code, designed and performed the experiments, and analysed the results. A.J. and M.J.M. wrote the original manuscript. All authors read and approved the final manuscript.

8 Author biographies

Ana Jiménez received her Bachelor’s degree, MSC in Signal Processing for Big Data and MSC in Telecommunications Engineering from the Universidad Politecnica de Madrid (UPM). Her research interests are artificial intelligence applied to biomedical field.

María José Merino received her Bachelor’s degree from Universidad de Jaén, and MSC in Signal Processing for Big Data and MSC in Telecommunications Engineering from Universidad Politécnica de Madrid (UPM). She is working as Data scientist with a primary focus on reinforcement learning, deep learning, and Natural Language Processing (NLP).

Juan Parras received his B.S in Telecommunications Engineering from Universidad de Jaén, and MSC and PhD in Telecommunications Engineering from Universidad Politécnica de Madrid (UPM). He is currently an Assistant Professor at UPM and his research interests include deep learning techniques applied to the health field, as well as deep reinforcement learning and game theory.

Santiago Zazo is a professor at UPM leading a research team focused on the application of AI and deep learning to health for medical prognosis, diagnosis, and decision support systems.

9 Funding

This project is funded by the European Union under grant agreement No. 101057619. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them. This work was also partly supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract No. 22.00115.

A Mathematical framework formulation

We present a unified framework to understand several methods for graph completion based on path reasoning. Graphs are collections of objects (nodes) and the set of interactions (edges) between pairs of these objects [23]. Knowledge graphs ($\mathcal{G}$) are a particular type of multirelational graph where the information is defined by a set of existing triples, including a head node (h), a tail node (t^*) and a relation (r) that links them:

$$(h, r, t^*) \in \mathcal{G} \tag{13}$$

Drug repurposing on knowledge graphs can be seen as a task of link prediction, where we ask the graph which diseases a certain compound treats. We can understand the problem as a query that has to be solved by the graph. The query is composed of a compound (c) as the head, and the relation treats, $q = (c, treats)$. The answer to this query is a disease (d) that can be treated with the compound, which is the tail of the triple.

$$(c, treats, d^*) \in \mathcal{G} \tag{14}$$

The problem can be formulated in terms of the probability of success of the compound over the disease, where the objective is that the answer equals the tail of the triple $d = d^*$, and which is conditioned on the existing graph:

$$p(c, treats, d = d^* \mid \mathcal{G}) = p(d \mid \mathcal{G}, (c, treats)) = p(d \mid \mathcal{G}, q) \tag{15}$$

It is interesting to characterise a related conditional probability that depends on an additional variable μ that represents the selected set of policies under consideration, thus the set of rules or strategy that we follow to traverse the graph following paths toward a certain disease.

$$p(c, treats, d = d^* \mid \mathcal{G}, \mu) = p(d \mid \mathcal{G}, (c, treats), \mu) = p(d \mid \mathcal{G}, q, \mu) \tag{16}$$

The probability of a certain candidate disease d can be formulated as the softmax of a score function f_ω, which represents the good match between the expected response of the query and the candidate d:

$$p_\omega(d \mid \mathcal{G}, q, \mu) = \frac{\exp f_\omega(d; q, \mathcal{G}, \mu)}{\sum_{d' \in \mathcal{D}} \exp f_\omega(d'; q, \mathcal{G}, \mu)} \tag{17}$$

where $\mathcal{D}$ is the set of final nodes (diseases) that are reached when we run a set of trajectories in the graph following a certain policy μ.

The score is computed over the set of candidate answers and measures the plausibility of the node being the correct answer. It is a function of the candidate node being evaluated, the query, the graph, and the policy. The score function can be defined as:

$$f_\omega(d; q, \mathcal{G}, \mu) = \mathrm{AGG}(\{\psi_\omega(\mu_i)\, \mathrm{AGG}(\{\phi_\omega(d; \pi_i^n, q, \mathcal{G}, \mu_i)\}_{\pi_i^n})\}_{\mu_i \in \mu}) \tag{18}$$

and it is parameterised by ω. π_i^n represents paths generated according to the policy μ_i and AGG represents an aggregation operation such as a sum or a maximum. The first aggregation evaluates a set of policies, and the second aggregation evaluates the set of paths or trajectories generated by each policy. The functions ϕ_ω and ψ_ω can take different forms depending on the model and represent the importance of the path and the policy in the prediction. The quality of the policy μ_i is represented through ψ_ω, which is a trainable parameter that weights the importance of different policies. In models in which rules are used, ψ_ω measures the weight of the rule. The quality of the generated trajectories or paths is represented in the term ϕ_ω, which should have high values if the evaluated node (the final node of the trajectory) is the answer and low values otherwise.

A.1 Path generator

A policy generator is an element that generates policies. The policy is the strategy we use to traverse the graph. This generation process will be modelled as a random generator as follows:

$$p_\theta(\mu \mid \mathcal{G}, q) \tag{19}$$

whose distribution is parameterised by θ, where μ has to be understood as a set of policies.

$$\mu = \{\mu_i\}_{i=1}^{M} = [\begin{array}{cccc} \mu_1 & \mu_2 & \cdots & \mu_M \end{array}] \tag{20}$$

Sampling any of these policies, which are themselves random variables, we can obtain trajectories:

$$\{\mu_i\} \sim p_\theta(\mu \mid \mathcal{G}, q) \to \{\pi_i^n\} \sim p_\theta(\pi \mid \mu_i) \tag{21}$$

where the trajectory π_i^n defines a sequence of nodes and the corresponding relations between them. An example of a path that proposes Epirubicin as a treatment for lung cancer is:

$$[\text{Epirubicine} \overset{\text{Upregulates}}{\longrightarrow} \text{Gene STK10} \overset{\text{Regulates}}{\longrightarrow} \text{Gene SOX2} \overset{\text{is associated with}}{\longrightarrow} \text{Lung cancer}]$$

In this project, we will also work with the concept of rule, which is another strategy to obtain paths or trajectories to traverse the graph in order to answer the query.

A rule is defined by the head and body following the structure [55]:

$$\text{head} \Rightarrow \text{body}$$

and the triplets take the form of:

$$\text{initial node type} \overset{\text{relation}}{\longrightarrow} \text{final node type}$$

The body of the rule gives a possible explanation for the relation represented by the head. Formally, the rules have the form of:

$$\text{Node type 0} \xrightarrow{\text{r}} \text{Node type L} \Rightarrow [\text{Node type 0} \xrightarrow{r_1} \text{Node type 1} \xrightarrow{r_2} \text{Node type 2} \longrightarrow \cdots \xrightarrow{r_L} \text{Node type L}]$$

where L is the length of the rule, node types represent the type of entity in the graph, for example, compound or disease, and r_l represents the relations or edges such as “treats” or “palliates”. An important property of these rules is composition, as it allows generating paths along the graph that can explain the head and make it more meaningful. Node type 0 and L, which are linked through the relation r, are also related through the sequence of relations r_1 to r_L.

For the objective of knowledge graph completion, missing triples are the head of the rule, and the body is used to find alternative paths between nodes that are plausible and equivalent to the relation in the head. In the drug repurposing case, the query is always “compound treats disease”.

A rule generator provides the sequence that defines the entity types and the relations that create the compositional effect. Defining by letter z the rules as latent variables, the rule generator will be represented by:

$$p_\theta(z \mid \mathcal{G}, q) \tag{22}$$

where each rule z_i is, in fact, an ordered sequence of edge types or relations (starting from e_0):

$$z_i = [r_1, r_2, ...r_L] \tag{23}$$

where the relation also defines the type of nodes, as a relation can only exist between specific node types, for example, the relation “treats” can only exist between a compound and a disease. Each realisation of z represents a different sequence z_i that can be translated into a trajectory.

$$z = \{z_i\}_{i=1}^{M} = [\begin{array}{cccc} z_1 & z_2 & \cdots & z_M \end{array}] \tag{24}$$

$$\{z_i\} \sim p_\theta(z \mid G, q) \to \{\pi_i^n\} \sim p_\theta(\pi \mid z_i) \tag{25}$$

As the rule is defined by the relation, the same rule z_i generates multiple trajectories that differ at the nodes. Applying the rule:

$$\text{Compound} \xrightarrow{\text{treats}} \text{Disease} \Rightarrow [\text{Compound} \xrightarrow{\text{causes}} \text{Side effect} \xrightarrow{\text{is caused by}} \text{Compound} \xrightarrow{\text{treats}} \text{Disease}]$$

the query can be solved because a compound "Lapatinib" causes nausea like "Captopri" which treats hypertension or because "Lapatinib" causes insomnia like "Bepridil" which treats hypertension. The same rule generates two trajectories. Furthermore, the same generator can generate another rule:

$$\text{Compound} \xrightarrow{\text{treats}} \text{Disease} \Rightarrow [\text{Compound} \xrightarrow{\text{downregulates}} \text{Gene} \xrightarrow{\text{is associated to}} \text{Disease} \xrightarrow{\text{associates}} \text{Gene} \xrightarrow{\text{is associated to}} \text{Disease}]$$

which also relates compounds with the diseases they treat.

In the same way, the policy represents sequences of nodes and relations in the form of trajectories or paths.

Taking into account the concepts of policy generator and score function, the probability of the candidate for repurposing can be parameterised by θ and ω.

$$p(d \mid \mathcal{G}, q) \to p_{\omega,\theta}(d \mid \mathcal{G}, q) \tag{26}$$

This expression can be decomposed into two processes: path generation and reasoning prediction. The objective of the path generator (parameterised by θ) is to obtain the policy/rules that models the problem, and the reasoning predictor (parameterised by ω) uses those paths to answer queries.

$$p_{\omega,\theta}(d \mid \mathcal{G}, q) = \sum_{\mu} p_\omega(d \mid \mathcal{G}, q, \mu) p_\theta(\mu \mid \mathcal{G}, q) \tag{27}$$

$$p_{\omega,\theta}(d \mid \mathcal{G}, q) = \sum_{z} p_\omega(d \mid \mathcal{G}, q, z) p_\theta(z \mid \mathcal{G}, q) \tag{28}$$

The objective of the graph completion task, which is to predict correct answers, is now modelled by a path generator that has to be optimised to provide the best trajectories and a reasoning predictor that gives the likelihood of the answer d conditioned on a latent set of policies μ or rules z, the query q, and the knowledge graph. Then, the problem can be structured in two steps:

- To evaluate the predictor using some rules/paths for training.
- To generate (good) paths. The method of obtaining policies differentiates most algorithms.

We present in Figure 9 the general scheme that the algorithms described are going to follow to solve the problem of answering queries from the graph.

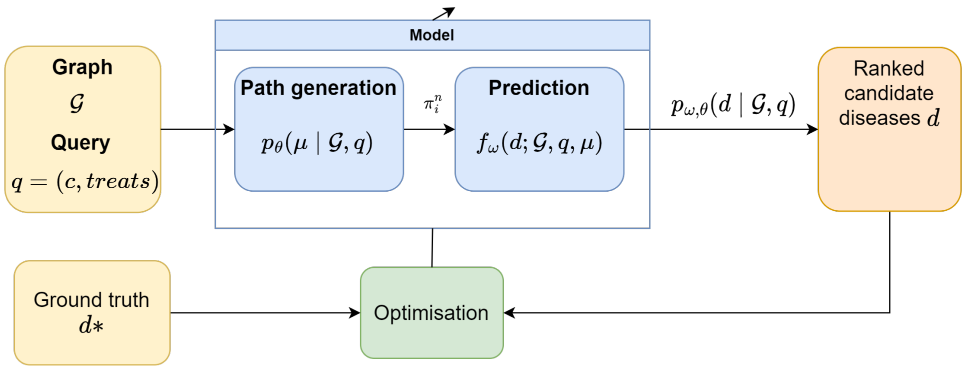

Graph
$\mathcal{G}$
Query
$q = (c, treats)$
Model
Path generation
$p_\theta(\mu \mid \mathcal{G}, q)$
π_i^n
Prediction
$f_\omega(d; \mathcal{G}, q, \mu)$
$p_{\omega,\theta}(d \mid \mathcal{G}, q)$
Ranked candidate diseases d
Ground truth
$d*$
Optimisation

Figure 9: Representation of the graph completion process using path reasoning. Both steps of the process can be distinguished: the reasoning predictor based on the score function and the policy generator used to generate trajectories or paths for the prediction. The representation is the same in the case rules z are used instead of policies μ.

A.2 Fixed policy

There are several ways to generate paths and the most simple one is to assume that the generator is fixed. We can generate paths following different principles, for example random walks, Breadth First Search (BFS) or Depth First Search (DFS). Using these fixed methods, we can obtain paths to traverse the graph from a starting compound node to a final disease node.

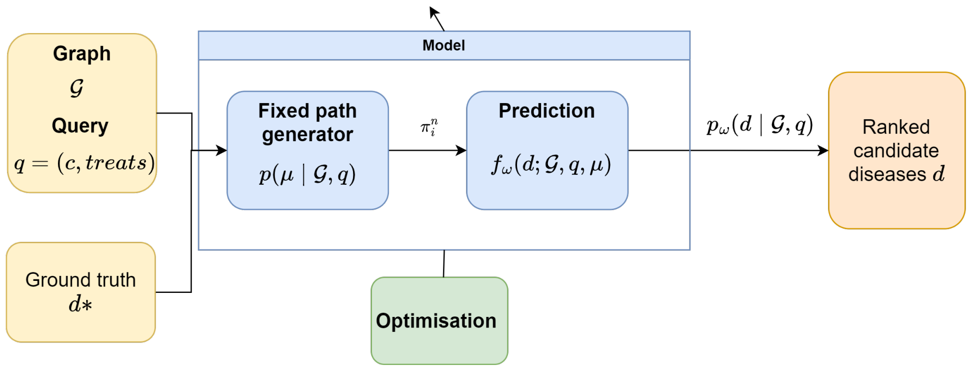

Graph
$\mathcal{G}$
Query
$q = (c, treats)$
Ground truth
$d*$
Model
Fixed path generator
$p(\mu \mid \mathcal{G}, q)$
π_i^n
Prediction
$f_\omega(d; \mathcal{G}, q, \mu)$
$p_\omega(d \mid \mathcal{G}, q)$
Ranked candidate diseases d
Optimisation

Figure 10: Architecture of the path based model for drug repurposing considering fixed path generation. The strategy to obtain the paths is always the same and independent of the prediction results. We represent the expression of policy μ for simplicity, but rule generators also fit this description.

The paths π_i^n obtained with the generator propose a set of candidate diseases that have to be ranked according to the scoring function $f_\omega(d; \mathcal{G}, q, \mu)$. With that score function, we can obtain the probability that a certain compound treats the candidate disease. The parameters of the score function can be optimised to maximise the probability of true treatments.

AnyBURL [24], and therefore SAFRAN [26], are models that use this workflow as shown in Figure 11 to generate rules. In this case, the fixed path generator consists of a path sampler and a rule generator. Rules are generalised from paths sampled using random walks. Only rules with a score higher than a threshold are kept. The rules are then used for prediction, using different score functions based on the confidence of the rules [25], [26].

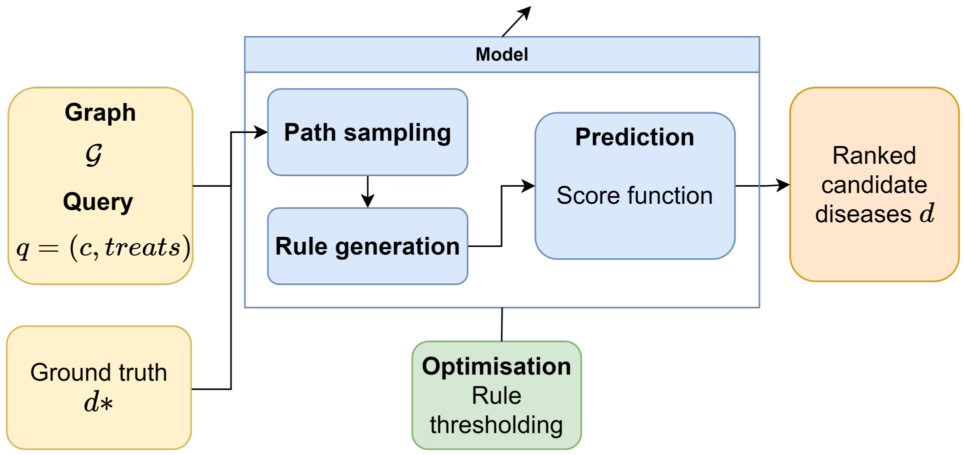


Figure 11: Description of AnyBURL based on our architecture. In this case, the input of the model is the whole graph (training set) including the ground truth. Paths are sampled based on random walk, and they are used to generate rules using a bottom-up approach. Then the confidence of the rule is computed, so only the rules with a confidence higher than a threshold are used for prediction. Rules are applied to the graph to obtain predictions that are ranked using the confidence of the rule.

A.3 Reinforcement learning

The next step is to use generators that can be updated to find the best paths between the compounds and the corresponding diseases. Some methods use reinforcement learning to model the trajectory on the graph as a Markov Decision Process. The objective of the reinforcement learning problem is to find the optimal policy to answer the queries. Starting from the head node, the agent learns to walk to the tail node, choosing intermediate nodes step by step, taking into account the path history.

In the drug repurposing context, the environment is the graph, and the possible actions are all the links the agent can choose from a certain node to the next. The objective of the agent is to move from a compound node to a disease node that is linked through the relations "treats". The state includes all nodes and relationships travelled through to the current node, so the next action depends on the whole path. For that reason, it has to be modelled with a structure with memory such as a recurrent neural network.

The trajectory π can be modeled as:

$$p_\theta(\pi \mid \mathcal{G}, q) = p_\theta(a_1, a_2, ..., a_L \mid \mathcal{G}, q) = \prod_{l=1}^{L} p_\theta(a_{S_l} \mid S_l) \tag{29}$$

where S_l is the state and a_{S_l} is the action taken at state S_l. The state includes information about the query, the initial entity (head of the query), the relation, and the previous actions or relations selected. The graph is also represented in the state since the choice of actions represents a path through the graph. The selected action a_{S_l} depends on the history states S_l up to step l, which is the current state.

Reinforcement learning methods also require the definition of the reward function that indicates whether the actions taken are good or not. In the most simple case,

$$\begin{cases} R\left(S_L \mid q\right) = 1 & \text{if } d = d^* \\ R\left(S_L \mid q\right) = 0 & \text{if } d \neq d^* \end{cases} \tag{30}$$

where d is the final node at step L, so the reward is positive if the agent reaches the final target at step L and zero if not. Gradient-based optimisation techniques are applicable if we have N trajectories for policy μ_θ (considering that here we have a single policy, as we are working on-policy).

For models that are optimised using policy search, the objective function of the policy gradient algorithm is the expected return of the policy.

$$\mathbb{E}_{\pi_\theta^n \sim \mu_\theta}\left[R\left(S_L \mid q\right)\right] \tag{31}$$

A widely used estimator for this objective function is the following one, based on trajectories sampled from the policy:

$$\mathbb{E}_{\pi_\theta^n \sim \mu_\theta}\left[R\left(S_L \mid q\right)\right] \approx \frac{1}{N}\sum_{n=1}^{N} R\left(S_L^n \mid q\right) p_\theta(\pi_\theta^n \mid \mathcal{G}, q) = \frac{1}{N}\sum_{n=1}^{N} R\left(S_L^n \mid q\right) \prod_{l=1}^{L} p_\theta(a_{S_l^n} \mid S_l^n) \tag{32}$$

Replacing the probability by the logarithm of the probability, the objective function that needs to be maximised is the following:

$$\frac{1}{N}\sum_{n=1}^{N} R\left(S_L^n \mid q\right) \sum_{l=1}^{L} \log p_\theta(a_{S_l^n} \mid S_l^n) \tag{33}$$

This is the objective function for one query q, to train the model, we average over the whole training set.

Policy search is an algorithm whose objective focuses on maximising the reward to obtain the parameters of the policy generator. We are working on policy, so only one policy is taken into account; therefore, the score function has to be particularised for that case.

$$f_\theta(d; q, \mathcal{G}, \mu_\theta) = \sum_{\pi_\theta^k} \phi_\theta\left(d; \pi_\theta^k, q, \mathcal{G}, \mu_\theta\right) = \sum_{\pi_\theta^k} \sum_{l=1}^{L} \phi_{\theta,l}\left(a_{S_l^n}; \pi_\theta^k, q, \mathcal{G}, \mu_\theta\right) \tag{34}$$

where $k \in \mathbb{K}$ represents all the trajectories that have d as the final node. We use parameters θ as the predictor is part of the policy generator, the predictions are used to generate paths choosing based on the probability step by step. The probability of each of these nodes can be derived from Equation 33, where in this equation (Equation 33) the final nodes of the N trajectories are taken into account $d \in \mathcal{D}$:

$$p_\theta(d \mid \mathcal{G}, q, \mu_\theta) = \frac{1}{K}\sum_{\pi_\theta^k} R(S_L^k \mid q) \sum_{l=1}^{L} \log p_\theta(a_{S_l^k} \mid S_l^k) \tag{35}$$

and related to Equation 34 since the probability is the softmax of the score function as represented in Equation 17.

The algorithm can be summarised in the following scheme:

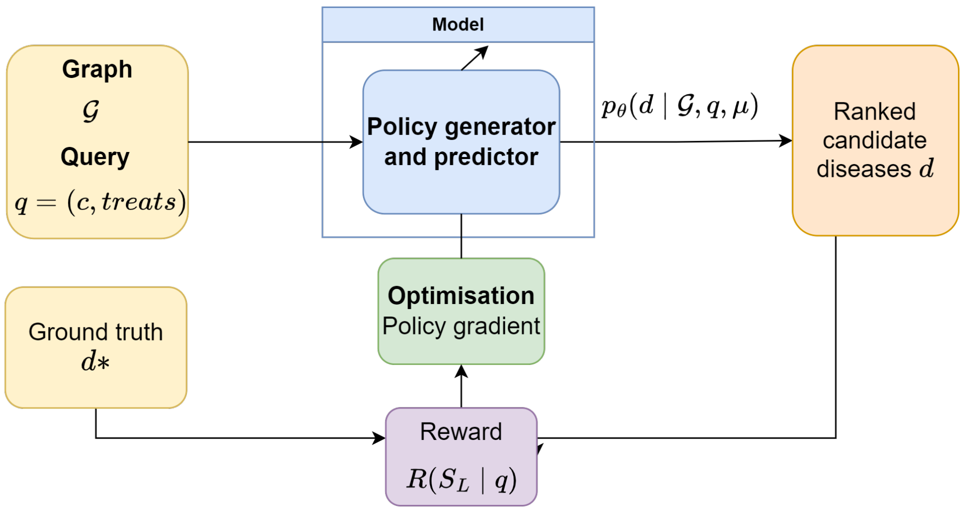


Figure 12: Description reinforcement learning model for path reasoning, in particular, MINERVA [21]. The core of the algorithm is the policy generator, which is trained to obtain the best policy through the reward using policy search. Paths are sampled from the generator to obtain candidate diseases which are ranked according to the path that proposes them.

The algorithm presents two main blocks: the policy generator combined with reasoning predictor and the optimisation block based on policy gradient. When the current policy is sampled μ_θ, the trajectory is generated conditioned on the query and the graph, and a final node is reached. In this case, the reasoning predictor is included in the policy generator, as the action or relation selected at each step depends on the score function and the result of the predictor. Once the agent reaches the final node, the reward is observed and used to optimise the policy using policy gradient and the objective function presented in Equation 32 and Equation 33.

MINERVA [21] is a reinforcement learning agent optimised using this policy search formulation, in particular, REINFORCE. Other models have included modifications in MINERVA, such as PoLo [22], which includes more terms in the reward, so paths that follow metapaths that are known to be useful have a higher reward, or DIVINE [27], which adds generative adversarial reasoner to the process.

A.4 Rule based path reasoning

Reasoning based on reinforcement learning has the problem that the action space is large and the reward is sparse, as few paths lead to the correct answer and a positive reward. There are other strategies, such as those that use trainable rule generators to obtain paths for prediction.

Drug repurposing based on rules consists of a rule generator and a reasoning predictor with logic rules, which are independent and trained simultaneously to improve each other [28]. The reasoning predictor uses the logic rules provided by the rule generator to answer queries, providing an effective reward to train the rule generator, which helps significantly reduce the search space.

Based on a query $q = (c, treats, d^*)$, the probability of the answer conditioned on the existing knowledge graph $p(d = d^* \mid G, q) = p(d \mid \mathcal{G}, q)$ is modelled using a set of logic rules z, treated as a latent variable that must be inferred.

$$p_{\omega,\theta}(d \mid \mathcal{G}, q) = \sum_z p_\omega(d \mid \mathcal{G}, q, z) p_\theta(z \mid \mathcal{G}, q) \tag{36}$$

The rule generator defines a prior distribution on the logic rules for each query, $p_\theta(z \mid q)$, which is parameterised by a recurrent neural network. The probability of the rule generator that generates rules of length L can be defined as:

$$\begin{aligned} p_\theta(z \mid \mathcal{G}, q) &= p_\theta(r_1, r_2, ...r_L \mid \mathcal{G}, q) = \\ &= p_\theta(r_1 \mid e_0) p_\theta(r_2 \mid e_0, r_1)...p_\theta(r_L \mid e_0, r_1, r_2, ...r_L) = \\ &= \prod_{l=1}^{L} p_\theta(r_l \mid S_l) \end{aligned} \tag{37}$$

where r_i is the relation at step i, and S_l is defined as the state and includes information about the query, the initial entity (head of the query), the relation, and the previous relations selected. The graph is also represented in the state, since the choice of actions represents a path through the graph. This rule generator is equivalent to the policy generator defined previously.

The reasoning predictor computes the likelihood of the answer conditioned on the logic rules and the existing knowledge graph $\mathcal{G}$, $p_\omega(d \mid \mathcal{G}, q, z)$. At each training iteration, a few logic rules are sampled from the generator, which are fed into the reasoning predictor to test these rules for prediction. The distribution $p(d \mid \mathcal{G}, q)$ can be calculated according to **??** as:

$$p_{w,\theta}(d \mid \mathcal{G}, q) = \sum_z p_w(d \mid \mathcal{G}, q, z) p_\theta(z \mid q) = \mathbb{E}_{p_\theta(z|q)} \left[p_w(d \mid \mathcal{G}, q, z) \right] \tag{38}$$

which is the objective function that has to be optimised by the whole model. This task is divided as the generator and predictor use different optimisation algorithms, but both contribute to a common goal.

Sampling the rule generator, we obtain the rules z_i which define the metapaths. The particularisation of the rule on the graph gives what is called the path, the sequence of nodes and relations, obtained applying the rule to a certain query. This path is what we consider to be a trajectory π_i^n.

RNNLogic [28] implements this architecture. The model includes a rule generator and a reasoning predictor that applies the rules to propose candidate predictions. The rule generator uses a recurrent neural network to generate a set of logic rules conditioned on the query, which are given to the reasoning predictor for query answering.

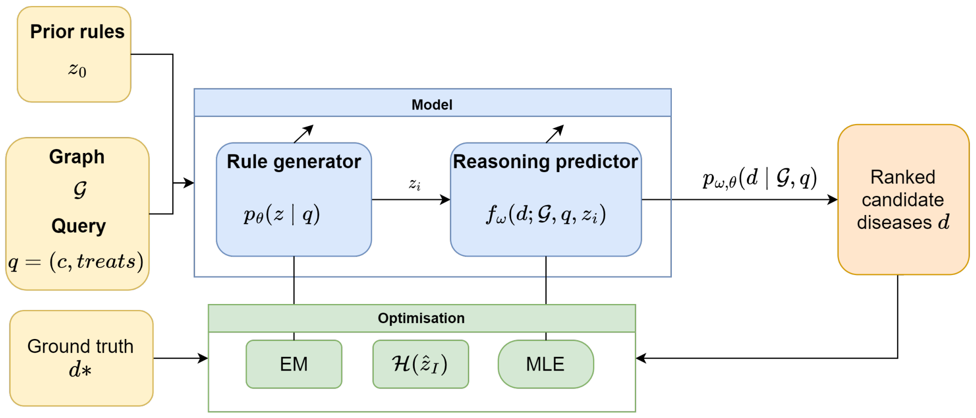


Figure 13: Process followed by RNNLogic for the graph completion task. Two main blocks, rule generator and reasoning predictor enhance each other to predict the correct answers for the queries. In addition to the graph and the query, there is another input which is a set of prior rules to initialize the generator. The model consists of a rule generator and a reasoning predictor. A set of rules is sampled and used for prediction. During training, the predictor is updated using maximum likelihood estimation (MLE). Combining information of the generation and the prediction, a score for each rule $\mathcal{H}(z_i)$ is computed and it is using during the training of the generator which is based on expectation maximisation.

The optimisation process is based on the EM algorithm for the rule generator and the maximum likelihood estimation principles (MLE) for the reasoning predictor, as shown in Figure 13. In each iteration, a set of rules z_i is sampled from the generator and used to update the predictor. Based on the result of the predictor, a set of high-quality rules is identified via posterior inference, taking into account the prior of the generator and the likelihood of the reasoning predictor.

The criteria for selecting the set of high-quality rules is the posterior distribution of each subset of logic rules. The authors use an approximation of the posterior probability $\mathcal{H}(\hat{z}_I)$, as the exact calculation is nontrivial because of its intractable partition function. Intuitively, the score $\mathcal{H}(\hat{z}_I)$ represents the quality of the rule taking into account the generator and the predictor. The scores should be high for rules that lead to true predictions according to the predictor. Moreover, the score should consider the prior probability of the rule given by the generator.

This set of high-quality rules is used to update the generator. This model requires a set of pre-computed rules and their corresponding prior scores to start the process. Different models can be used to obtain the initial set of rules based on the training data from the graph, for example AnyBURL [24].

The objective is to jointly train the rule generator and the reasoning predictor to maximise the logarithmic likelihood of the training data conditioned on the graph and the query.

$$\log p_{w,\theta}(d \mid \mathcal{G}, q) = \log \mathbb{E}_{p_\theta(z|q)} \left[p_w(d \mid \mathcal{G}, q, z) \right] \tag{39}$$

where the generator is parameterised by θ and the predictor by w. The graph is represented by $\mathcal{G}$, the query by q and the set of rules by z.

As there is an expectation operation related to the generator, a sample $\hat{z} \sim p_\theta(z \mid q)$ is used to approximate the objective function for each training instance as:

$$\log \mathbb{E}_{p_\theta(z|q)} \left[p_w(d \mid \mathcal{G}, q, z) \right] \approx \log p_w(d \mid \mathcal{G}, q, \hat{z}) \tag{40}$$

which represents the function that has to be maximised.

The set of rules generated $\hat{z}$ can be used by the predictor to find paths in the graph $\mathcal{G}$ leading to different candidate answers. Each candidate answer has a score:

$$f_\omega(d; q, \mathcal{G}, z) = \sum_{z_i \in \hat{z}} \psi_\omega\left(z_i\right) \sum_{\pi_i^n} \phi_\omega\left(e, \pi_i^n, \mathcal{G}, q, z_i\right) \tag{41}$$

where ψ_ω is the score of the rule z_i, which is a trainable parameter, and ϕ_ω is the score of the path π_i^n, taking into account the final node d, the trajectory through the graph, and the query. The path score should be high if the final node is the correct answer node and low otherwise. Once we have the score for each candidate answer, we can further define the probability that the answer d of query q is correct using a softmax function as in Equation 17. Then, the parameters ω are updated to maximise the log-likelihood of the correct answer.

To optimise the generator $p_\theta(z \mid q)$, a set of high-quality rules $\hat{z}_I$ is selected according to $\mathcal{H}(\hat{z}_I)$. For each data instance, the set of rules $\hat{z}_I$ is treated as part of the training data, and the generator is updated by maximising the logarithmic likelihood of $\hat{z}_I$. Moreover, $\mathcal{H}(\hat{z}_I)$ has information on the quality of the rules, so it can also be included in the generator optimisation in the form of weights of each rule:

$$\mathcal{H}(\hat{z}_I) \log p_\theta\left(\hat{z}_I \mid q\right) = \sum_{z_i \in \hat{z}_I} \mathcal{H}(\hat{z}_I) \sum_{l=1}^{L} \log p_\theta(r_{l^i} \mid S_l) \tag{42}$$

where r_{l^i} is the relation at step l of the rule i.

The optimisation of the rule generator in the rule-based architecture has strong connections with policy search algorithm used in reinforcement learning. The objective function of the rule generator is:

$$\sum_{z_i \in \hat{z}_I} \mathcal{H}(\hat{z}_I) \sum_{l=1}^{L} \log p_\theta(r_{l^i} \mid S_l) \tag{43}$$

considering one training sample and a set of high quality sampled rules $\hat{z}_I$. These rules generate a set of trajectories in the same way as sampled paths in reinforcement learning. The objective function in this case is defined as:

$$\sum_{n=1}^{N} R\left(S_L^N \mid q\right) \sum_{l=1}^{L} \log p_\theta(a_{S_l^n} \mid S_l^n) \tag{44}$$

so both functions are equivalent if we consider the posterior score $\mathcal{H}(\hat{z}_I)$ of rule-based as a reward. Reinforcement learning has the limitation that the reward is always set to 1 if the correct answer is reached and to 0 otherwise. The rule-based score includes more information as it combines the prior score of the rule with the result of the prediction.

Regarding the trajectories, there are also some differences. In reinforcement learning, L trajectories are generated in the form of multiple trials of the same query and the same policy. In rule-based models, the trajectories are generated by a set of rules $\hat{z}_I$. These rules are a subset of the rules used for prediction to select the best rules for optimisation. Both approaches coincide if the score function of rule-based models (which behaves as a reward) is set to 1 for the top rules selected from the posterior and to 0 otherwise.

References

[1] V. Parvathaneni, N. S. Kulkarni, A. Muth, and V. Gupta, "Drug repurposing: A promising tool to accelerate the drug discovery process," *Drug discovery today*, vol. 24, no. 10, pp. 2076–2085, 2019.

[2] N. Saberian, A. Peyvandipour, M. Donato, S. Ansari, and S. Draghici, "A new computational drug repurposing method using established disease–drug pair knowledge," *Bioinformatics*, vol. 35, no. 19, pp. 3672–3678, 2019.

[3] M. Danishuddin and A. U. Khan, "Structure based virtual screening to discover putative drug candidates: Necessary considerations and successful case studies," *Methods*, vol. 71, pp. 135–145, 2015.

[4] B. R. Beck, B. Shin, Y. Choi, S. Park, and K. Kang, "Predicting commercially available antiviral drugs that may act on the novel coronavirus (sars-cov-2) through a drug-target interaction deep learning model," *Computational and structural biotechnology journal*, vol. 18, pp. 784–790, 2020.

[5] X. Zeng, X. Song, T. Ma, *et al.*, "Repurpose open data to discover therapeutics for covid-19 using deep learning," *Journal of proteome research*, vol. 19, no. 11, pp. 4624–4636, 2020.

[6] D. Morselli Gysi, Í. Do Valle, M. Zitnik, *et al.*, "Network medicine framework for identifying drug-repurposing opportunities for covid-19," *Proceedings of the National Academy of Sciences*, vol. 118, no. 19, e2025581118, 2021.

[7] A. K. Gogineni, "Analysis of drug repurposing knowledge graphs for covid-19," *arXiv preprint arXiv:2212.03911*, 2022.

[8] K. Yang, Y. Yang, S. Fan, *et al.*, "Dronet: Effectiveness-driven drug repositioning framework using network embedding and ranking learning," *Briefings in Bioinformatics*, vol. 24, no. 1, bbac518, 2023.

[9] P. Xuan, Y. Ye, T. Zhang, L. Zhao, and C. Sun, "Convolutional neural network and bidirectional long short-term memory-based method for predicting drug–disease associations," *Cells*, vol. 8, no. 7, p. 705, 2019.

[10] Y. Zhu, C. Che, B. Jin, N. Zhang, C. Su, and F. Wang, "Knowledge-driven drug repurposing using a comprehensive drug knowledge graph," *Health Informatics Journal*, vol. 26, no. 4, pp. 2737–2750, 2020.

[11] C. Ma, Z. Zhou, H. Liu, and D. Koslicki, "Predicting drug repurposing candidates and their mechanisms from a biomedical knowledge graph," *BioRxiv*, pp. 2022–11, 2022.

[12] A. Daowd, S. Abidi, and S. S. R. Abidi, "A knowledge graph completion method applied to literature-based discovery for predicting missing links targeting cancer drug repurposing," in *Artificial Intelligence in Medicine: 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 14–17, 2022, Proceedings*, Springer, 2022, pp. 24–34.

[13] D. S. Himmelstein, A. Lizee, C. Hessler, *et al.*, "Systematic integration of biomedical knowledge prioritizes drugs for repurposing," *eLife*, vol. 6, pp. 1–35, 2017. DOI: `https://doi.org/10.7554/eLife.26726.001`.

[14] D. Domingo-Fernández, Y. Gadiya, A. Patel, *et al.*, "Causal reasoning over knowledge graphs leveraging drug-perturbed and disease-specific transcriptomic signatures for drug discovery," *PLoS computational biology*, vol. 18, no. 2, e1009909, 2022.

[15] S. Sadegh, J. Skelton, E. Anastasi, *et al.*, "Network medicine for disease module identification and drug repurposing with the nedrex platform," *Nature Communications*, vol. 12, no. 1, p. 6848, 2021.

[16] Z. Zhu, C. Shi, Z. Zhang, *et al.*, "Torchdrug: A powerful and flexible machine learning platform for drug discovery," *arXiv preprint arXiv:2202.08320*, 2022.

[17] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen, "Interaction embeddings for prediction and explanation in knowledge graphs," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 96–104.

[18] Z. Gao, P. Ding, and R. Xu, "Kg-predict: A knowledge graph computational framework for drug repurposing," *Journal of biomedical informatics*, vol. 132, p. 104 133, 2022.

[19] M. A. Thafar, R. S. Olayan, H. Ashoor, *et al.*, "Dtigems+: Drug–target interaction prediction using graph embedding, graph mining, and similarity-based techniques," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–17, 2020.

[20] O. Gurbuz, G. Alanis-Lobato, S. Picart-Armada, *et al.*, "Knowledge graphs for indication expansion: An explainable target-disease prediction method," *Frontiers in genetics*, vol. 13, 2022.

[21] R. Das, S. Dhuliawala, M. Zaheer, *et al.*, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *arXiv preprint arXiv:1711.05851*, 2017.

[22] Y. Liu, M. Hildebrandt, M. Joblin, M. Ringsquandl, and V. Tresp, "Integrating logical rules into neural multi-hop reasoning for drug repurposing," *arXiv preprint arXiv:2007.05292*, 2020.

[23] W. L. Hamilton, *Graph Representation Learning*. Springer International Publishing, 2020.

[24] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt, “Anytime bottom-up rule learning for knowledge graph completion,” *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2019-August, pp. 3137–3143, 2019.

[25] C. Meilicke, M. W. Chekol, M. Fink, and H. Stuckenschmidt, “Reinforced anytime bottom up rule learning for knowledge graph completion,” *arXiv preprint arXiv:2004.04412*, 2020.

[26] S. Ott, C. Meilicke, and M. Samwald, “Safran: An interpretable, rule-based link prediction method outperforming embedding models,” *arXiv preprint arXiv:2109.08002*, 2021.

[27] R. Li and X. Cheng, “Divine: A generative adversarial imitation learning framework for knowledge graph reasoning,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2642–2651.

[28] M. Qu, J. Chen, L.-P. Xhonneux, Y. Bengio, and J. Tang, “Rnnlogic: Learning logic rules for reasoning on knowledge graphs,” in *International Conference on Learning Representations*, 2021.

[29] M. Ali, M. Berrendorf, C. T. Hoyt, *et al.*, “PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings,” *Journal of Machine Learning Research*, vol. 22, no. 82, pp. 1–6, 2021. [Online]. Available: `http://jmlr.org/papers/v22/20-825.html`.

[30] D. S. Wishart, C. Knox, A. C. Guo, *et al.*, “Drugbank: A knowledgebase for drugs, drug actions and drug targets,” *Nucleic acids research*, vol. 36, no. suppl_1, pp. D901–D906, 2008.

[31] A. Gronchi, S. Frustaci, M. Mercuri, *et al.*, “Localized, high-risk soft tissue sarcomas (sts) of the extremities and trunk wall in adults: Three versus five cycles of full-dose anthracyclin and ifosfamide adjuvant chemotherapy: A phase iii randomized trial from the italian sarcoma group (isg) and spanish sarcoma group (geis).,” *Journal of Clinical Oncology*, vol. 28, no. 15_suppl, pp. 10 003–10 003, 2010.

[32] J. Chen, Y. Liu, Q. Sun, B. Wang, N. Li, and X. Chen, “Cyr61 suppresses growth of human malignant melanoma,” *Oncology Reports*, vol. 36, no. 5, pp. 2697–2704, 2016.

[33] S. Zhao, Y. Liu, Z. Yao, *et al.*, “Phase ii clinical trial of camrelizumab combined with avd (epirubicin, vincristine and dacarbazine) in the first-line treatment for patients with advanced classical hodgkin’s lymphoma,” *Blood*, vol. 140, no. Supplement 1, pp. 6579–6580, 2022.

[34] UNICANCER, *Combination Chemotherapy Plus Peripheral Stem Cell Transplantation in Treating Patients With Germ Cell Tumors*, ClinicalTrials.gov Identifier: NCT00003852, Retrieved from `https://clinicaltrials.gov/ct2/show/{NCT00003852}`, 2016.

[35] University of Pisa, *Xenotransplantation of Primary Cancer Samples in Zebrafish Embryos (xenoZ)*, ClinicalTrials.gov Identifier: NCT03668418, Retrieved from `https://clinicaltrials.gov/ct2/show/{NCT03668418}`, 2018.

[36] D. Kong, W. Hong, M. Yu, Y. Li, Y. Zheng, and X. Ying, “Multifunctional targeting liposomes of epirubicin plus resveratrol improved therapeutic effect on brain gliomas,” *International Journal of Nanomedicine*, pp. 1087–1110, 2022.

[37] N. C. Kampan, M. T. Madondo, O. M. McNally, M. Quinn, and M. Plebanski, “Paclitaxel and its evolving role in the management of ovarian cancer,” *BioMed research international*, vol. 2015, 2015.

[38] P. Katsaounis, A. Kotsakis, N. Kentepozidis, *et al.*, “Nab-paclitaxel as second-line treatment in advanced gastric cancer: A multicenter phase ii study of the hellenic oncology research group,” *Annals of gastroenterology*, vol. 31, no. 1, p. 65, 2018.

[39] S. A. Rosenthal, D. Hunt, A. O. Sartor, *et al.*, “A phase 3 trial of 2 years of androgen suppression and radiation therapy with or without adjuvant chemotherapy for high-risk prostate cancer: Final results of radiation therapy oncology group phase 3 randomized trial nrg oncology rtog 9902,” *International Journal of Radiation Oncology* Biology* Physics*, vol. 93, no. 2, pp. 294–302, 2015.

[40] OHSU Knight Cancer Institute, *Serial Measurements of Molecular and Architectural Responses to Therapy (SMMART) PRIME Trial*, ClinicalTrials.gov Identifier: NCT03878524, Retrieved from `https://clinicaltrials.gov/ct2/show/NCT03878524`, 2023.

[41] H. P. Safran, K. Winter, D. H. Ilson, *et al.*, “Trastuzumab with trimodality treatment for oesophageal adenocarcinoma with her2 overexpression (nrg oncology/rtog 1010): A multicentre, randomised, phase 3 trial,” *The Lancet Oncology*, vol. 23, no. 2, pp. 259–269, 2022.

[42] W. Feng, J. Wang, X. Yan, *et al.*, “Paclitaxel alleviates monocrotaline-induced pulmonary arterial hypertension via inhibition of foxo1-mediated autophagy,” *Naunyn-Schmiedeberg’s Archives of Pharmacology*, vol. 392, pp. 605–613, 2019.

[43] A. Ehrlich, S. Booher, Y. Becerra, *et al.*, “Micellar paclitaxel improves severe psoriasis in a prospective phase ii pilot study,” *Journal of the American Academy of Dermatology*, vol. 50, no. 4, pp. 533–540, 2004.

[44] R. Xu, N. Sato, K. Yanai, *et al.*, “Enhancement of paclitaxel-induced apoptosis by inhibition of mitogen-activated protein kinase pathway in colon cancer cells,” *Anticancer research*, vol. 29, no. 1, pp. 261–270, 2009.

[45] C. N. Fournier, D. Schoenfeld, J. D. Berry, *et al.*, “An open label study of a novel immunosuppression intervention for the treatment of amyotrophic lateral sclerosis,” *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*, vol. 19, no. 3-4, pp. 242–249, 2018.

[46] M. R. Jardim, X. Illarramendi, O. J. Nascimento, *et al.*, “Pure neural leprosy: Steroids prevent neuropathy progression,” *Arquivos de neuro-psiquiatria*, vol. 65, pp. 969–973, 2007.

[47] S. K. Shah and G. T. Gecys, “Prednisone-induced osteoporosis: An overlooked and undertreated adverse effect,” *Journal of Osteopathic Medicine*, vol. 106, no. 11, pp. 653–657, 2006.

[48] Memorial Sloan Kettering Cancer Center, *Study to Evaluate the Efficacy and Safety of Nintedanib (BIBF 1120) + Prednisone Taper in Patients With Radiation Pneumonitis*, ClinicalTrials.gov Identifier: NCT02496585, Retrieved from `https://clinicaltrials.gov/ct2/show/{NCT02496585}`, 2022.

[49] Janssen Research & Development, *A Study That Provides Long-term Safety Follow-up and Examines Long-term Exposure to Abiraterone Acetate*, ClinicalTrials.gov Identifier: NCT01517802, Retrieved from `https://clinicaltrials.gov/ct2/show/{NCT01517802}`, 2022.

[50] E. M. Hamed, A. R. Ibrahim, M. H. Meabed, *et al.*, “The outcomes and adverse drug patterns of immunomodulators and thrombopoietin receptor agonists in primary immune thrombocytopenia egyptian patients with hemorrhage comorbidity,” *Pharmaceuticals*, vol. 16, no. 6, p. 868, 2023.

[51] B. F. Uretsky, S. Murali, P. S. Reddy, *et al.*, “Development of coronary artery disease in cardiac transplant patients receiving immunosuppressive therapy with cyclosporine and prednisone.,” *Circulation*, vol. 76, no. 4, pp. 827–834, 1987.

[52] H. Verhelst, P. Boon, G. Buyse, *et al.*, “Steroids in intractable childhood epilepsy: Clinical experience and review of the literature,” *Seizure*, vol. 14, no. 6, pp. 412–421, 2005.

[53] Fred Hutchinson Cancer Center, *Alemtuzumab and Glucocorticoids in Treating Newly Diagnosed Acute Graft-Versus-Host Disease in Patients Who Have Undergone a Donor Stem Cell Transplant*, ClinicalTrials.gov Identifier: NCT00410657, Retrieved from `https://clinicaltrials.gov/ct2/show/{NCT00410657}`, 2010.

[54] J. E. Parrillo, R. E. Cunnion, S. E. Epstein, *et al.*, “A prospective, randomized, controlled trial of prednisone for dilated cardiomyopathy,” *New England Journal of Medicine*, vol. 321, no. 16, pp. 1061–1068, 1989.

[55] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, “A Survey on Knowledge Graphs: Representation, Acquisition, and Applications,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, Feb. 2022.