Universiteit Leiden

Master Computer Science

Automated drug repurposing workflow for rare diseases

Name:	Carmen Reep
Student ID:	s3006182
Date:	16/08/2022
Specialisation:	Bioinformatics
1st supervisor:	Dr. Eleni Mina
2nd supervisor:	Dr. Núria Queralt Rosinach
3rd supervisor:	Dr. Katy Wolstencroft

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Drug development is a time-consuming and costly process. It is therefore not economically viable for pharmaceutical companies to develop entirely new drugs for diseases with low prevalence. As a result, fewer than 6% of all rare diseases have an approved treatment option. A cost efficient and faster alternative to novel drug development would be to redevelop existing compounds for use in different diseases. This is referred to as drug repurposing. In this study, we developed an automated drug repurposing workflow that uses network and graph-based analysis methods together with machine learning to produce a ranked list of candidate compounds. We created a web app that runs a drug repurposing process after a user inputs a rare disease and symptom of interest. The app could be used for lab researchers to find potential candidate compounds to alleviate symptoms of a rare disease. This research demonstrates each step of the workflow with the use-case Huntington's disease, which is a rare neurodegenerative disorder of the central nervous system caused by an elongated CAG repeat on the Huntingtin gene. To evaluate the predictions made by the machine learning algorithm in the workflow, we manually explored the three top-ranked drug predictions for Huntington's disease. All three drugs are supported by evidence as plausible candidates.

Acknowledgements

I would first like to thank my supervisors Dr. Eleni Mina and Dr. Núria Queralt Rosinach for their guidance and mentoring throughout the project. Our weekly Teams meetings were very helpful, from discussing my thesis progression to exchanging knowledge about hobbies and houseplants.
I also wish to thank my LIACS supervisor, Dr. Katherine Wolstencroft for the opportunity to do this project and her feedback on my first draft version of the report.
Furthermore, I would like to express my gratitude to the BioSemantics group of the LUMC (especially Pablo, Karolis, Daphne, Deniz, Annika and Rajaram) for giving me feedback, asking critical questions, and answering all my questions.
Lastly, I would like to thank my dad for helping me figure out how to turn my workflow into a Flask app so that I could demonstrate this repurposing workflow within a rudimentary but working web application.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rare diseases

Rare diseases are low-prevalent disorders caused by pathogenic mutations or particular harmful environmental effects (e.g. air pollution, pesticides or metals) that can be chronically debilitating or life-threatening [1]. Currently, there are more than 7000 rare diseases that together affect 10% of the European and American populations [2]. Of all these diseases, fewer than 6% have an approved treatment option [2]. The majority of patients with a rare disease have a drastically decreased quality of life due to stress, anxiety, physical impairment, or pain. This illustrates the need for drug development for rare diseases. Novel drug development however is a time-consuming and costly process. Developing a new drug typically requires 10 to 15 years before it enters the market and can cost as much as US$2.5 billion [2]. Especially for rare diseases that affect only a small number of individuals, developing an entirely new drug is unlikely to provide a return of investment for pharmaceutical companies [1].

1.2 Drug repurposing

A cost efficient and faster way to provide drugs for rare diseases is via drug repurposing. Drug repurposing is the process of redeveloping a compound for use in a different disease [2] based on the knowledge that drugs target particular pathways and disease mechanisms that may be shared by multiple diseases [3]. It is faster than developing a new drug because preclinical testing and safety assessment will have already been done. It is cheaper due to substantial savings in costs of preclinical phases (costs of bringing a repurposed drug to the market is estimated to be only US$300 million, which is about ten times cheaper than developing a new drug [4]). Besides these two big advantages, repurposed drugs also have a lower risk of failure, as for these drugs safety assessments have been performed and the toxicity levels have been defined. Additionally, drug repurposing can help to reveal new targets and pathways that can be further exploited [4].

There are two main drug repurposing approaches: computational approaches and experimental approaches [4]. Experimental approaches use techniques such as affinity chromatography and mass spectrometry to identify binding partners for drugs, or phenotypic screening to identify compounds that show disease-relevant effects in model systems. Computational approaches involve systematic analysis of any type of data. Current computational approaches can be divided into molecular, clinical, and biophysical methods [5]. Molecular-based drug repurposing techniques aim to predict drugs that may reverse disease-gene signatures by comparing these disease-gene expression signatures with drug-gene expression signatures post-drug treatment. Clinical drug repurposing methods leverage large-scale health data such as the electronic medical record (EMR) and health surveys, making them well suited for precision medicine [5]. Biophysical methods include structural, ligand-based, and molecular docking methods and are useful for diseases with

known targets. Biophysical methods are computationally efficient and leverage the biochemical properties of drugs to achieve drug-gene interaction predictions, often referred to as drug-target interaction (DTI) prediction. This approach is based on the hypothesis that chemicals with similar structure have shared biological activity [4].

To help realise the full potential of drug repurposing, Pushpakom et al. (2018) put forward some recommendations based on current drug repurposing challenges. One technological challenge is data access and integration. The landscape of biomedical information resources is heterogeneous and broad, resulting in the informatics community to often face the challenge of integrating data across many knowledge resources [6]. Thus, there is need for better integrative platforms for data analysis to realise computational drug repurposing. This way, analyses can be more refined and put in a more user-friendly format for 'non-experts' [4].

1.3 BioKnowledge reviewer library

Queralt-Rosinach *et al.* [6] built a platform that creates structured review articles for a specific disease by mining and integrating data from different online databases. This way, the process of knowledge integration is accelerated and, as the data mined is a vast amount of knowledge known to date, the research performed using this data is efficient. The platform is referred to as the BioKnowledge reviewer library. The structured review article created with the BioKnowledge reviewer is represented as a knowledge graph, which is a semantic representation of relational information, where concepts are represented as nodes and relations between these concepts are represented as edges [6]. Knowledge graphs are computationally accessible and efficiently processable, which allows for application of graph and machine learning algorithms.

1.4 Research question

The research question addressed in this thesis is: **'Can the BioKnowledge reviewer data mining and integration approach together with the biophysical drug repurposing approach be used to create an automated drug repurposing workflow that finds novel drug-target interactions to alleviate symptoms of a specific rare disease?'**.

In this project, we built a drug repurposing workflow that first mines and integrates data from online data resources using the BioKnowledge reviewer, resulting in a knowledge graph containing different biological entities and relations. Then graph and network-based analysis methods together with machine learning methods are used to predict novel drug-target interactions, where the targets are the genes that contribute to the disease phenotype of interest. With this network-based approach, the drug-gene interaction prediction is formulated as a link prediction problem, as can be seen in Figure 1.1.

We turned this automated data mining workflow into a web app, which takes as input a specific rare disease and symptom for which drugs are sought. The idea is that this web app can be used by lab researchers as a tool to find drug candidates that can be tested experimentally.
The drug-repurposing workflow generalises to any (rare) disease, but as a proof-of-concept, we use the rare disease of Huntington's to demonstrate the working of each step of the workflow.

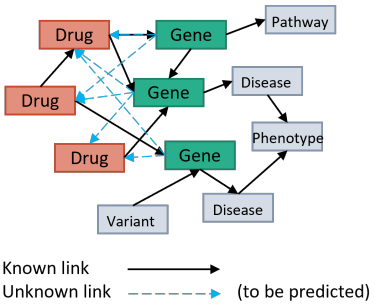


Figure 1.1: *An illustration of the network-based approach to find novel drug-gene interactions (the dashed blue arrows).*

1.5 Huntington's disease

Huntington's disease (HD) is a rare neurodegenerative disorder of the central nervous system characterized by dementia, choreatic movements and behavioural and psychiatric disturbances [7]. It has a prevalence of 1/10,000-1/20,000 and the mean age of onset is 30-50 years. The disease is caused by an elongated CAG repeat on the short arm of chromosome 4p16.3 in the Huntingtin gene, where a longer CAG repeat results in earlier onset of disease. Although the gene that causes HD was found almost 30 years ago, the disease still lacks effective treatment. There are some symptomatic treatments available but because their effects are limited, there is a constant need for better, modifying drugs to treat symptoms of the disease [7].

1.6 Overview

We discuss each step of our drug repurposing workflow in Chapter 2. Chapter 3 showcases this workflow for the use-case Huntington's disease with symptom chorea, and shows the web app prototype. In Chapter 4, we discuss the limitations and improvements of our workflow. The last chapter, Chapter 5, contains some concluding words.

Chapter 2

Methods

Figure 2.1 provides an overview of the workflow we created to find candidate compounds for a symptom of a specific rare disease. This workflow involves four main steps:

1. Creation of a knowledge graph by selecting a subnetwork from the Monarch Initiative, adding drugs using the DGIdb database, creating drug-drug edges based on drug compound structure similarity, followed by transformation of this final graph into an RDF graph.
2. Embedding of the graph using RDF2Vec, which creates random walks over the graph and inputs these as sentences in a Word2Vec algorithm. For each entity in the graph, a feature vector is created.
3. Creation of edge representations by fusing drug and gene feature vectors and sampling from these representations to obtain prediction and training data.
4. Training of a supervised machine learning model (XGBoost) using the training data, evaluating using repeated stratified 10-fold cross validation and using the best found model to predict the unknown drug-gene interactions of the prediction data.

In this chapter, we discuss each step of this workflow in detail, together with how we ranked and validated the final predictions and how the web app was created. The source code is available at `https://git.lumc.nl/catreep/drugrepurposing`.

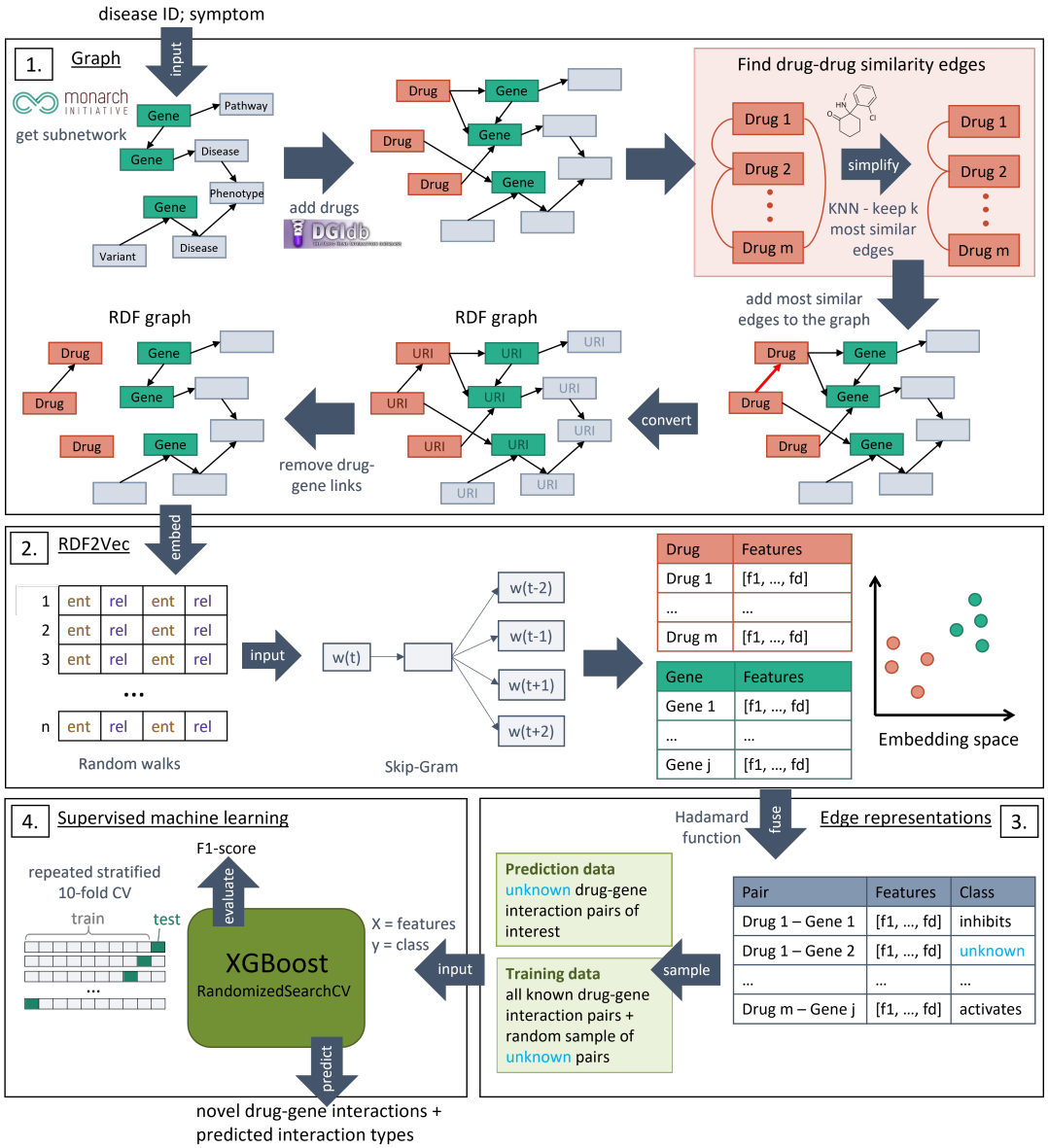


Figure 2.1: *The drug repurposing workflow. (1.) It takes as input a disease and symptom of interest, then constructs the knowledge graph using Monarch and DGIdb, adds drug-drug similarity edges based on SMILES compound structure, turns the graph into an RDF graph and removes drug-gene links ready for embedding. (2.) It then applies RDF2Vec, which creates random walks over the graph for each entity, trains a skip-gram model and outputs one feature vector for each entity in the graph. (3.) Next, it generates edge representations for each drug-gene pair and turns this into prediction and training data. (4.) Then it trains an XGBoost classification model using the prediction data, finds the best model by hyperparameter tuning using randomized search, evaluates using repeated stratified 10-fold CV and uses the best found model to predict the interactions of interest.*

2.1 Graph construction

2.1.1 Monarch

In the first step of creating the knowledge graph, the BioKnowledge reviewer library (https://github.com/SuLab/bioknowledge-reviewer) is used to obtain human and animal biological data and metadata from Monarch. The Monarch Initiative is a collaborative, open science effort that aims to semantically integrate genotype–phenotype data from many sources and species in order to support disease modelling, precision medicine, and mechanistic exploration [8]. The BioKnowledge reviewer library retrieves data and metadata using the Monarch Biolink API (https://api.monarchinitiative.org/api/), version 1.1.14. After providing a list of seed nodes, in this case a disease and symptom identifier, the first layer of neighbours and relations from each seed are obtained from Monarch. After this, ortholog phenotype nodes are obtained by first finding the orthologs for all genes that are connected to the two input seeds, then using Monarch to find all ortholog phenotypes for these ortholog genes. Ortholog phenotypes can help to increase the connectivity around the seed nodes, since the conserved biology of animal models can help to explain the pathology in humans [6]. The seed nodes together with their neighbour nodes and ortholog-phenotype nodes are used to obtain the final Monarch network. This final network is used to build nodes and edges, after which these are saved in a specific format.
The edges are formatted as triples: subject, predicate, object. Each triple contains additional information: a reference URI, reference supporting text and reference date, a property label, property description and property URI. Nodes have an identifier (ID), semantic group, URI, label, name, synonyms and description. The IDs are from different ontologies maintained or used by Monarch. For example, the phenotypic abnormalities 'weight loss' has ID 'HP:0001824' which is from the Human Phenotype Ontology (HPO). The semantic group specifies which biomedical entity category the node belongs to, for example genotype, gene or disease. A URI is a Uniform Resource Identifier, which is a persistent string of characters used to identify a resource on the internet. In this case, the URI is a link that directs the user to a web page that provides a more detailed description about the biomedical identifier, semantics of the relation, or the research study where the relation is extracted from [6]. The advantage of URIs is that if the resource is modified, the identifier will remain the same [9].

The BioKnowledge reviewer code uses the API to extract only the IDs and labels of subjects, objects and relations, after which the semantic categories and URIs are derived from the prefixes of the IDs using a manually created dictionary, the so-called 'prefix2semantic' dictionary. This way of adding semantic groups and URIs has some downsides. Firstly, if Monarch adds a new source or species, the prefixes of these new IDs are not yet in the prefix2semantic dictionary, which means these should be added manually. The second downside is that some prefixes translate to different semantic groups. For example, an entity with ID prefix 'FlyBase' could be a gene (e.g. FlyBase:FBgn0027655) or a variant (e.g. FlyBase:FBal0209708) and an entity with ID prefix 'MGI' could be a genotype (e.g. MGI:5441507) or a gene (e.g. MGI:96904). To avoid many incorrect semantic categories and URIs and to avoid the need for manually updating the prefix2semantic dictionary, we adjusted the code such that the URIs and semantic categories are also automatically extracted from the Monarch Biolink API.

2.1.2 DGIdb

After obtaining the Monarch graph, the next step is to find drugs to include in the graph. For this, the Drug-Gene Interaction Database (DGIdb) is used. DGIdb is a web resource that provides information on drug-gene interactions and druggable genes from publications, databases, and other web-based sources [10]. The drug-gene information is obtained using an API (https://dgidb.org/api/) version v2, with all genes in the Monarch graph as seeds. For this,

the Monarch genes, which are all from different resources, need to be normalised to Entrez. This is done by using BioThings `MyGene.info` API, which is accessed with the Python wrapper *biothings_client* version v0.2.6 [11]. Then for each gene, a list of drugs (ID, name) that interact with the gene, together with the type of interaction and interaction source is obtained. The drug identifiers are all from ChEMBL Database or the Wikidata knowledge base, e.g. 'chembl:CHEMBL212579' for 'BENZAMIL' and 'wikidata:Q798309' for 'BCG VACCINE'. There are many different interaction types. For better prediction, we used the direction of the interaction types instead of the interaction types themselves. There are two interaction directions: 'inhibits', where the drug decreases the biological activity or expression of a gene target, and 'activates', where the drug increases the biological activity or expression of a gene target [10]. Because there were some relations without direction, we introduced a third category: 'regulates'. Table 2.1 shows each interaction direction category with all interaction types that belong to that category.

We turned these three interaction groups into URIs, using the OBO Relations Ontology (RO) `https://www.ebi.ac.uk/ols/ontologies/ro`, version 2022-05-23. RO is a collection of relations intended for standardisation across ontologies in the OBO Foundry. It incorporates upper-level relations as well as biology-specific relationship types [12]. Using the OBO Relations Ontology, the URI for 'inhibits' becomes `http://purl.obolibrary.org/obo/RO_0002408`, for 'activates' `http://purl.obolibrary.org/obo/RO_0002406`, and for 'regulates' `http://purl.obolibrary.org/obo/RO_0011002`.

The final DGIdb graph is saved as two files: a nodes file and an edges file, with the same format as the two Monarch files.

category	ID	interaction types
inhibits	RO:0002408	antagonist, antibody, antisense oligonucleotide, blocker, cleavage, inhibitor, inhibitory allosteric modulator, inverse agonist, negative modulator, partial antagonist, suppressor
activates	RO:0002406	activator, agonist, chaperone, cofactor, inducer, partial agonist, positive modulator, stimulator, vaccine
regulates	RO:0011002	NA, None, n/a, other/unknown, adduct, allosteric modulator, binder, ligand, modulator, multitarget, potentiator, product of, substrate

Table 2.1: *The three drug-gene interaction categories, with all interaction types that belong to each category. Source:* `https://www.dgidb.org/interaction_types`.

2.1.3 Drug-drug similarity

The hypothesis for the biophysical drug repurposing approach used in this thesis is that structurally similar molecules share similar targets. This is why we needed a way to define structurally similar drugs. In our workflow, the Tanimoto coefficient is used to measure the similarities between drugs. Tanimoto coefficient is a widely used similarity measure for comparing molecular fingerprints [13]. Molecular fingerprints are bit-string representations indicating whether a feature is present (set as 1) or absent (set as 0) in the molecule. The higher the Tanimoto coefficient, the more similar the two compounds are [13]. Given molecules A and B, where N_A is the number of features (set as 1) in molecule A, N_B is the number of features (set as 1) in molecule B, and N_{AB} is the number of features set as 1 in both molecules A and B, the Tanimoto coefficient is calculated as follows [13]:

$$\tau = \frac{N_{AB}}{N_A + N_B - N_{AB}}$$

Thus, in order to find the Tanimoto coefficients between all drugs in the knowledge graph, the drugs need to be converted into molecular fingerprints. This is done in a few steps.
First, for each drug, the SMILES chemical structure notation is found using the BioThings

`MyGene.info` API accessed with *biothings_client* version v0.2.6 [11]. SMILES stands for Simplified Molecular Input Line Entry System (SMILES), which is a widely used 1D representation for molecular structures [14]. This specification describes molecular structures in the form of ASCII strings, where symbols such as C, N, O are used for atoms and =, # for bonds. See Figure 2.2 for an example of a SMILES structure notation.

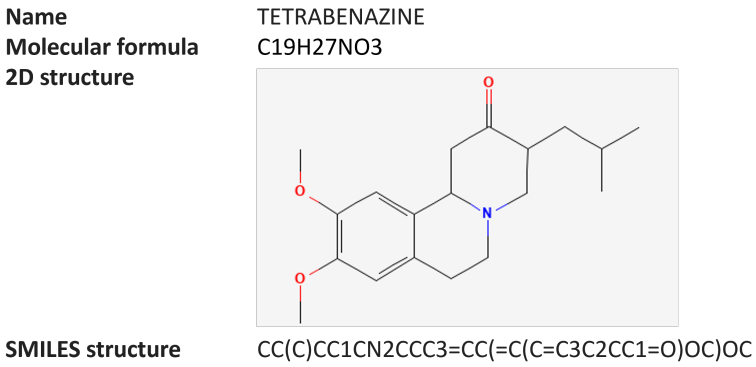


Figure 2.2: *An example of a SMILES structure notation [15]. Tetrabenazine is a known drug to treat involuntary choreatic movements in HD patients [1].*

After obtaining the SMILES chemical structure for each drug, the SMILES structures are converted into RDKit molecule objects using the *RDKit* Python package version 2022.3.2 with the *Chem* module [16]. RDKit is an open source toolkit for cheminformatics. The RDKit molecule objects are then turned into Morgan fingerprints using the *GetMorganFingerprintAsBitVect()* function of the *AllChem* RDKit module. Morgan fingerprints are widely used molecular fingerprints as they are the best performing fingerprints in small molecule target prediction benchmarks and virtual screening [17]. They recognise the presence of specific circular substructures around each atom in a molecule, which are predictive of the biological activities of small organic molecules [17].
Once a fingerprint is obtained for each drug in the knowledge graph, the Tanimoto coefficients are calculated using the *BulkTanimotoSimilarity()* function of the *DataStruct* module from *RDKit* (version 2022.3.2). This function takes one query fingerprint and a list of target fingerprints as input and returns a list of Tanimoto coefficients, one for each target fingerprint. So for each molecule, the Tanimoto coefficient score is calculated for every other molecule in the dataset and stored in a symmetric matrix of pairwise similarities.

Now for every possible pair of drugs in the DGIdb graph, the Tanimoto coefficient is calculated, resulting in a huge number of weighted edges, especially when dealing with a large amount of drugs. For example, as explained in more detail in Chapter 3.1, the graph created with the seeds Huntington's disease and chorea has 1352 drugs, which would result in 1,827,904 similarity edges (for comparison, the graph has 4,444 edges not including similarity edges). So to avoid a large, complex network, the majority of edges are removed. This is done based on a method by Thafar et al. [18], where all similarity scores are ranked in descending order and only the top-10 most similar drugs are kept, as per the k-nearest neighbours algorithm. For an efficient calculation of the top ten most similar drugs for each drug, the matrix of Tanimoto scores are sorted with the indices of the k largest similarities in front and all indices of similarities smaller than the k-th element moved to the back using numpy.argpartition() [19] with k = 10. Then for each drug, the edges to the first ten drugs are kept and all other edges are removed.
To all these obtained similarity edges the label 'similar to' is assigned, with ID 'CHEMINF:000481' and URI '`http://semanticscience.org/resource/CHEMINF_000481`'.

The most similar drug-drug edges are saved in the same format as the Monarch and DGIdb edges files.

2.1.4 RDF graph

Using the BioKnowledge reviewer library, the files from Monarch and DGIdb containing nodes are combined into one final nodes file, and the edges files from Monarch, DGIdb and the drug-drug similarity are combined into one final edges file. The final nodes and edges files together form the knowledge graph. This final graph is turned into a Resource Description Framework (RDF) graph. RDF is a powerful data model for representing information on the Web and is a World Wide Web Consortium (W3C) recommendation to be used as a semantic web technology [20]. For the RDF graph, besides every entity, the semantic groups these entities belong to (gene, drug, etc.) also need to be represented as URIs. Because Monarch did not provide identifiers for the semantic groups, we manually mapped the semantic group labels to terms in ontologies and used their URIs. For this, we used the Ontology Lookup Service (OLS) (**http://www.ebi.ac.uk/ols**), which is a repository for biomedical ontologies that aims to provide access to the latest ontology versions. For best practice, we used the minor number of ontologies as possible. We achieved this using only one ontology to represent the semantic types in the RDF graph, which is the Semanticscience Integrated Ontology (SIO) (`http://semanticscience.org/ontology/sio.owl`) version 1.53. SIO is a public ontology that facilitates biomedical knowledge discovery and is commonly used by the bioinformatics community [21]. It offers classes and relations to describe objects, processes and their attributes in the biomedical domain [21]. After we found IDs for each semantic label, we obtained URIs using the URI resolution service *identifiers.org* (`https://registry.identifiers.org/registry/sio`, accessed June 2022). Identifiers.org URIs are stable, perennial and globally unique which can be used to annotate a wide range of entities or concepts from a variety of fields, including proteins, publications and ontological terms [22].

An RDF graph consists of subject-predicate-object triples. Whereas subjects and predicates are always URIs, objects can be described using either a URI or a literal [23], see Figure 2.3. Literals are used to identify values such as strings, numbers, and dates.

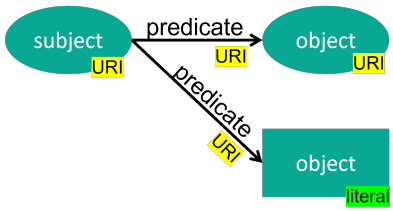


Figure 2.3: *Example of two RDF triples with the same subject, where one object of the triple is a literal (string, number, date), and the other object is a resource (URI).*

The BioKnowledge reviewer library did not support transformation of the graph into RDF. In our workflow, the RDF graph is created using the RDFLib Python package version 6.1.1 [24] and stored in Turtle format, which stands for 'Terse RDF Triple Language' [25]. Figure 2.4 describes two entities in the RDF graph in Turtle format. Each subject can have multiple predicates, where the ';' symbol is used to separate these predicates [25]. The end of a predicate-object list for one subject is indicated with '.'. Literals are enclosed between quotes '""', as can be seen in Figure 2.4, the label of each entity is a literal. URIs can be written as absolute URIs (between '< >') or specified with a prefix label together with an identifier, which is also referred to as a CURIE (a compact URI). Prefixes are defined at the start of the Turtle document and referred to as namespaces. The example shows *ns2:RO_0003301*, which is equivalent to <*http://purl.obolibrary.org/obo/RO_0003301*>.

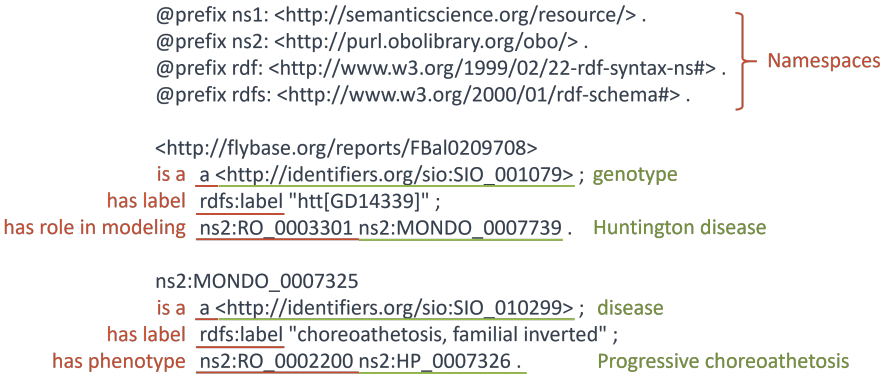


Figure 2.4: *Example of two entities in the RDF turtle file.*

2.2 Embedding

The next step after creating the knowledge graph is to embed the graph so that it can be used as input to train a Machine Learning model. Before the embedding, all known drug-gene interactions are removed from the graph. This is done because keeping these edges would create bias on the prediction task, as the embedding vectors of the drugs would be very similar to the embedding vectors of the genes these drugs directly interact with. It is therefore better to see the graph as two separate graphs, one 'drug' graph and one 'gene' graph, where the gene graph is essentially the graph obtained from Monarch. Then after the embedding, the drug vectors are fused with the gene vectors to obtain drug-gene edges, which are used for training the machine learning model, as explained in more detail in section 2.3.

Celebi *et al.*[26] evaluated knowledge graph embedding approaches for drug-drug interaction prediction, concluding that RDF2Vec with Skip-Gram generally surpasses other embedding methods, as it had the best performance values (AUC, AUPR and F1-score). This is why in this workflow, RDF2Vec is used for the graph embedding.
RDF2Vec is a methodology that adapts the language modelling approach of Word2vec to RDF graph embeddings [26]. Word2vec is a computationally-efficient two-layer neural network model for learning word embeddings from text [27]. Word2vec takes sentences (i.e. sequence of words) as input and trains a neural network to represent each word as a vector of numerical values in a latent feature space, assuming that words that occur closer together in word sequences are more statistically dependent [27]. In RDF2Vec, sequences of entities and relations are created with random walks over the graph. These entity-relationship sequences are seen as the sentences to input in the Word2vec model. To learn one embedding for each entity in the graph, the Skip-Gram model is used, which is a language model consisting of a neural network with one hidden layer [18]. It takes as input one target entity (one-hot encoded) and tries to predict context entities. After training the model, all entities are projected into a two-dimensional feature space [27]. Semantically and syntactically closer entities will appear closer in the feature space [26]. For the prediction task used in this thesis, only the vectors for drugs and genes are of interest, so only these vectors are selected for further computation. If two genes have many of similar entities close to them in the sequences of random walks, these two genes will end up having similar embeddings. In other words, the gene vectors are created by taking into account all entities in the graph.

For transforming the RDF graph to vectors, the Python function *RDF2VecTransformer* from the *rdf2vec* module of the package *pyrdf2vec* version 0.2.3 is used [28]. The maximum depth of one walk is set to 4 and for each entity in the graph, the maximum number of walks is set to 10, which is copied from the 'Getting Started' example of *readthedocs* version 8.3.5 of the pyrdf2vec

package (https://pyrdf2vec.readthedocs.io/en/latest/readme.html).

2.3 Edge representations

With RDF2Vec, it is possible to obtain edge embeddings. In this workflow, we did not create edge embeddings using RDF2Vec. Instead, we created node embeddings and fused these into edge embeddings. We did this because besides edge representations of known drug-gene interactions, we also need edge representations of unknown drug-gene interactions (the negative samples). The unknown interaction embeddings cannot be obtained with RDF2Vec, because these edges simply do not exist. For consistency, we created edge representation for known and unknown drug-gene interaction in the same way: by fusing node embeddings.

2.3.1 Fusion

The edge representations of known and unknown drug-gene interactions are created by fusing the drug embeddings with the gene embeddings, resulting in one embedding for every possible drug-gene combination. Fusion is done with the Hadamard function, which, according to Grover and Leskovec [29], is the best performing operator for learning edge features for link prediction using node2vec. Node2vec is a similar approach to RDF2vec, but has the downside that it reads graphs as unlabelled graphs, while knowledge graphs are labelled by nature, i.e. they contain different types of edges [30]. The Hadamard equation is as follows:

$$[f(d) \odot f(g)] = f(d) * f(g),$$

where $\odot$ represents element wise matrix multiplication, d is a drug node and g is a gene node, $f(d)$ is the feature vector of drug d and $f(g)$ is the feature vector of gene g.

For computational efficiency, all drug embeddings are turned into one array, after which each gene embedding is multiplied with all drug embeddings at once using this drug embedding array. For each drug-gene fused embedding the class is added, which is inhibits, activates or regulates for the known edges. For the edges that are not present in the graph, the label 'unknown' is assigned.

2.3.2 Training/prediction data

After obtaining one embedding for each possible drug-gene combination, the next step is to select the embedding vectors to train with (training data) and to find the embeddings of the drug-gene interactions of interest to predict the class for (prediction data). For this, first all fused embeddings are split into two groups: known interaction (regulates, inhibits or activates) and unknown interactions. Then all genes that contribute to the disease phenotype of interest are found (defined as the 'genes of interest'). Using these genes of interest, the group of unknown interactions is also split into two groups: unknown interactions that **do not** include these genes of interest and unknown interactions that **do** include these genes of interest, of which the latter group is the prediction data.

For the training data, positive and negative samples are needed because a supervised machine learning model will be used to predict drug-gene interaction classes [26]. The positive samples are all known interactions (regulates, inhibits, or activates). The negative samples can be obtained from the unknown interactions that are not the prediction data. Taking all these negative samples would result in data imbalance. As can be read in Section 3.2, for the graph created with the seeds Huntington's disease and chorea, 1753 of the created edge representations are known (positive samples) and 382,215 edge representations are unknown, where 269,323 of these unknown edges form the prediction data, leaving 112,892 edges to form the negative samples. This imbalanced data would influence the performance metric of our machine learning model [26]. There exists several techniques to deal with this known issue. One common way is to rebalance imbalanced data artificially, which is referred to as "upsampling" when cases are replicated from the minority,

and "downsampling" when cases from the majority are ignored [31]. In this thesis, we chose to account for the data imbalance issue by downsampling the negative samples. Negative samples are randomly selected from the unknown interactions that are not in the prediction data, with sample size equivalent to the class in the positive set with the largest amount of interactions (regulates, inhibits, or activates). In the case of Huntington's disease with symptom chorea, the 1753 positive samples are divided in 1301 regulates edges, 391 inhibits edges, 61 activates edges, which meant that 1301 negative samples were randomly selected.

2.4 Supervised machine learning

Thafar *et al.* [18] found that the XGBoost classifier is the best classifier for drug-target interaction prediction for datasets with more than 100 drugs and genes [18]. This is why we used the XGBoost machine learning library in this thesis. XGBoost, which stands for 'eXtreme Gradient Boosting', provides high performing, scalable tree boosting algorithms. It uses parallel tree boosting, which makes it is more than ten times faster than other machine learning or deep learning models [32]. The *XGBClassifier()* function of the Python package *xgboost* [33] version 1.3.3 is used for implementation of the model. The learning objective is set to 'multi:softmax', which enables XGBoost to do multiclass classification using the softmax objective.

Even though not all unknown interactions are used as the negative samples, but a sample is taken with size equivalent to the largest class of the positive set (regulates, inhibits, or activates), there is still some class imbalance. In the case of Huntington's disease with chorea, the negative class has 1301 cases as does the class 'regulates', but 'inhibits' and 'activates' classes have fewer cases. To deal with this class imbalance, sample weights are computed using the *compute_sample_weight()* function from *sklearn* [34] version 1.1.1, which are then used for training the model. This provides some bias towards the minority classes while training the model.

The model's hyperparameters are optimised using a randomized search on the search space, as shown in Table 2.2. With a randomized search, a fixed number of parameter settings is sampled from the specified distributions. For this, we used the *RandomizedSearchCV()* function from *model_selection* module of the *sklearn* Python package [34] version 1.1.1. We set the number of parameter setting combinations to try out (n_iter) to 20. The higher the number of n_iter the better, which is why we set n_iter higher than the default setting of 10, but keeping the computation time in mind, we did not set n_iter too high.

The model is evaluated using the repeated stratified k-fold cross validation technique together with the F1-score metric. With k-fold cross validation, the training data is partitioned into equal sized k-subsets, in which one subset is used as a test set and the remaining data is used to train the model for a fold [26]. Stratified cross-validation ensures that each subset is an appropriate representative of the original data in terms of class balance, rather than randomly splitting the data into k subsets. Repeated cross-validation means that the cross-validation process is repeated a number of times, which leads to a better evaluation [35]. For repeated stratified k-fold cross validation, we used the function *RepeatedStratifiedKFold()* from *model_selection* module of the *sklearn* Python package [34] version 1.1.1. The number of subsets for the k-fold cross validation is set to 10, which is a popular-used value for k in machine learning. The number of repeats is set to 5.

For each iteration, the F1 score is calculated for each class X. The F1 score can be interpreted as the harmonic mean of precision and recall, where precision is the percentage of instances that are correctly classified as X, and recall is the percentage of instances that are actually X but are predicted by the classifier to be Y [35]. The equations below show the formulas for precision, recall

and F1 score, where TP is the true positive, FP false positive, and FN false negative.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The F1 scores for each class are averaged, weighted by the support (number of true instances for each label) of each class. Weighting by support accounts for class imbalance.
The final F1 score is the average over all k iterations and number of repeats.

parameter	description	search space
min_child_weight	Minimum sum of instance weight (hessian) needed in a child.	2, 3, 5, 8, 13, 20, 30
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree.	0, 0.2, 0.5, 0.8, 1.2, 1.6, 2.5, 4, 6
reg_alpha	L2 regularization term on weights.	0, 0.5, 1, 3, 5, 10
reg_lambda	L1 regularization term on weights.	0, 0.5, 1, 3, 5, 10
subsample	Subsample ratio of the training instances.	uniform(0.5, 1)
colsample_bytree	Subsample ratio of columns when constructing each tree.	uniform(0.2, 1)
max_depth	Maximum depth of a tree.	4, 6, 8, 10, 12, 14, 16
n_estimators	Number of boosting rounds.	35, 45, 50, 70, 80, 90, 100
learning_rate	Step size shrinkage used after each boosting step to prevent overfitting.	uniform(0, 0.3)

Table 2.2: *The hyperparameter search space of the XGBoost classifier, with descriptions of each parameter. Source: https://xgboost.readthedocs.io/en/stable/parameter.html. uniform(a,b) indicates a uniform distribution on (a,b).*

The best hyperparameters are used to build the final model. The final model is used to predict the classes of the unknown drug-gene interactions of interest. The confidence of each prediction is obtained using the *predict_proba()* function of *xgboost* version 1.3.3, which returns the probability of an interaction belonging to its predicted class.

2.5 Ranking & validation

2.5.1 Drug ranking

Now for every gene that is associated with the symptom of interest, an interaction type and score is found for every drug in the dataset, given that the drug had no known interaction with the gene. The next step is to remove all drug-gene interactions that are predicted to have class 'unknown', which is the negative class (meaning no interaction). Then, all unknown drug-gene interactions that are predicted to be positive (inhibits, activates, or regulates) are left, which is the goal of this thesis. However, this is most likely a very long list of predicted positive interactions. Thus, there is need for a way to rank this long drug-gene interaction list.

Ranking is performed in three steps. First, the confidence score of each prediction is used to select only the best predicted interactions. For this, all scores less than 0.9 are dropped. Then the drug-gene interactions are sorted by grouping on the drugs and looking at the number of interactions each drug has. Our hypothesis is that if a drug has a positive interaction with many genes

that cause a specific symptom, that drug will better alleviate the symptom than a drug that has a positive interaction with only a few genes that causes the symptom. So now drugs that have more interactions are ranked higher. The next step is to rank drugs that interact with the same amount of genes. This is done by ranking these drugs by the sum of confidences of prediction. Take for example drug A and drug B, with both drugs having the same amount of positive interactions with score higher than 0.9. If the sum of the prediction confidences of all interactions of drug A is higher than the sum of the prediction confidences of all interactions of drug B, drug A will be ranked higher than drug B.

2.5.2 Prediction validation

Our model is a black box model that does not provide the justification underlying each prediction. For lab researchers, experimentally testing drug candidates is challenging and costly [36]. It is therefore of importance to them to have some evidence underlying our predictions. This is why we manually explored the potential evidence of the top three ranked predictions for Huntington's disease with chorea. Thafar *et al.* [18] used a similar approach to look for novel drug-gene interactions as we did in this thesis. They validated their novel predictions manually by using scientific literature and biomedical databases, including ChEMBL, PubChem and DrugBank, to see whether the predictions already exist. They managed to confirm 21 out of 25 of their highest ranked predictions. However, the data used for their predictions was last updated in 2008, which resulted in many 'novel' predicted drug-gene interactions that actually already exist. In this thesis, data is extracted from online databases and knowledge graphs, which are frequently updated, ensuring that novel drug-gene predictions have never previously been proposed. This is why instead of looking at whether the predicted interactions already exist, we manually explored the top three ranked drugs by looking at the whether these drugs are used in other neurodegenerative disorders.

2.6 Web app

A web app is created as a user-friendly interface to the drug repurposing workflow to be used by researchers with no technical expertise. The user selects a disease and a symptom, which triggers the running of the whole process as described above, resulting in the final list of ranked predictions. This app is created with the Python web framework Flask [37] version 1.1.2. Flask is based on the Werkzeug WSGI (Web Server Gateway Interface) toolkit and the Jinja2 template engine [38]. The Werkzeug toolkit implements requests, response objects, and utility functions. The Jinja2 template engine enables the passing of Python variables into HTML templates using only a few lines. For example, the way the final list of ranked predictions (saved as dataframe: df_predictions) can be turned into an HTML template with title 'Predictions' as follows:

```
<html>
    <head>
        Predictions
    </head>
    <body>
        {df_predictions}
    </body>
</html>
```

A detailed description of the function of the web app can be found in Appendix A.

Chapter 3

Results

In this chapter, we show the numbers and figures of each step of the workflow for the use-case Huntington's disease with symptom chorea (from now on referred to as 'HD chorea'), for which the graph was created on 2022-06-27. Also, we show the created web app with all its functionalities.

3.1 Graph construction

The HD chorea heterogeneous network constructed using the workflow has seven entities, with many underlying relations. In total, there are 2189 nodes and 17467 edges in the HD chorea graph. Figure 3.1 shows the RDF data model of this graph. In order to turn the network into an RDF graph, these entity groups needed to be turned into URIs, for which identifiers were found manually using the SIO ontology (as described in Section 2.1.4). Table 3.1 provides each entity group, their SIO identifiers and descriptions, together with the amount of nodes in each group, sorted in descending order.

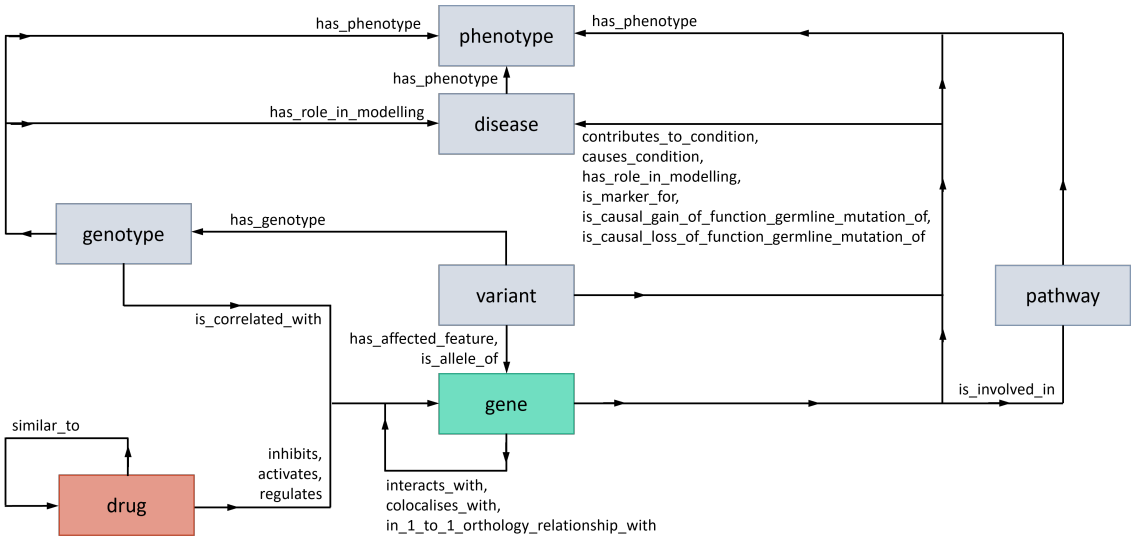


Figure 3.1: *Overview of all entities and relations in the HD chorea KG.*

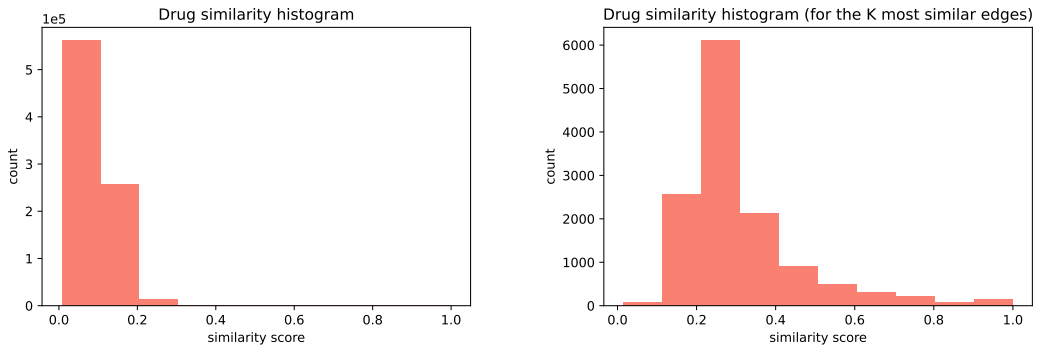
Table 3.2 provides the identifiers for each relation in the HD chorea graph, together with their description and the amount of edges in each relation group, again sorted in descending order. All relation identifiers, except for the drug relations, were automatically extracted from Monarch. The identifiers for 'inhibits', 'activates', 'regulates' and 'similar to' were manually added (see Sections 2.1.2 and 2.1.3). For every drug, edges were formed with the top 10 most similar drugs, which

entity	identifier	description	count
drug	SIO:010038	A drug is a chemical substance that contains one or more active ingredients that regulate one or more biological processes.	1352
gene	SIO:010035	A gene is part of a nucleic acid that contains all the necessary elements to encode a functional transcript.	284
disease	SIO:010299	A disease is the outward manifestation of one or more disorders.	194
genotype	SIO:001079	A genotype is a functional specification of a biological entity in terms of its genetic composition (or lack thereof).	127
variant	SIO:001381	A genomic sequence variant is part of a nucleic acid which is compositionally different than another reference genomic part.	106
phenotype	SIO:010056	A phenotype is an observable characteristic of an individual.	71
pathway	SIO:001107	A pathway is an effective specification that outlines a set of actions that forms a way to achieve an objective.	49

Table 3.1: *All entity groups in the HD chorea KG graph with their SIO identifiers and description (source: Semanticscience Integrated Ontology (SIO) [21]). Column 'count' shows the number of nodes in each entity group.*

means that there should be 10 outgoing edges for each drug node. However, as can be seen, the amount of 'similar to' edges is not exactly 10 times the amount of drugs in the graph. This can be explained by the fact that all Wikidata drug IDs (48 in total) and a few CHEMBL IDs could not be turned into SMILES structures using *biothings_client*, resulting in no formed similarity edges between these drugs.

The distribution of the similarity scores calculated using the Tanimoto coefficient on the SMILES compound structures can be seen in Figure 3.2. Figure 3.2a shows the similarity distribution of all possible drug-drug combinations. As can be seen, most have a similarity score between 0 and 0.2, which indicates most drugs are not that similar. Figure 3.2b shows the similarity distribution for the 10 most similar edges for each drug. Here, it can be seen that a large amount of the least similar edges were removed, revealing the truly similar edges.



(a) Distribution of the similarity scores for all drug-drug edges.

(b) Distribution of the similarity scores for the k-most similar edges.

Figure 3.2: *Distribution of the similarity scores based on the drug SMILES compound structure, calculated using the Tanimoto coefficient.*

relation	identifier	description	count
similar to	CHEMINF:000481	Connects a molecular entity that is deemed similar to another according to some algorithm.	13023
regulates	RO:0011002	The entity x has an activity that regulates an activity of the entity y.	1301
has phenotype	RO:0002200	A relationship that holds between a biological entity and a phenotype. Here a phenotype is construed broadly as any kind of quality of an organism part, a collection of these qualities, or a change in quality or qualities. The subject of this relationship can be an organism, a genomic entity such as a gene or genotype, or a condition such as a disease.	1016
interacts with	RO:0002434	A relationship that holds between two entities in which the processes executed by the two entities are causally connected.	900
inhibits	RO:0002408	Directly negatively regulates.	391
causes condition	RO:0003303	A relationship between an entity (e.g. a genotype, genetic variation, chemical, or environmental exposure) and a condition (a phenotype or disease), where the entity has some causal role for the condition.	212
contributes to condition	RO:0003304	A relationship between an entity (e.g. a genotype, genetic variation, chemical, or environmental exposure) and a condition (a phenotype or disease), where the entity has some contributing role that influences the condition.	107
has genotype	GENO:0000222	A relationship that holds between a biological entity and some level of genetic variation present in its genome.	106
has role in modelling	RO:0003301	A relation between a biological, experimental, or computational artefact and an entity it is used to study, in virtue of its replicating or approximating features of the studied entity.	103
correlated with	RO:0002610	A relationship that holds between two entities, where the entities exhibit a statistical dependence relationship. The entities may be statistical variables, or they may be other kinds of entities such as diseases, chemical entities or processes.	72
activates	RO:0002406	Directly positively regulates.	61
involved in	RO:0002331	x is involved in y if and only if x enables some process y', and y' is part of y.	
enables	RO:0002327	catalyses.	49
colocalises with	RO:0002325	x colocalises with y if and only if x is transiently or peripherally associated with y.	38
is allele of	GENO:0000408	A relation linking an instance of a variable feature (aka an allele) to a genomic location/locus it occupies. This is typically a gene locus, but a feature may be an allele of other types of named loci such as QTLs, or alleles of some unnamed locus of arbitrary size.	39
has affected feature	GENO:0000418	A relation that holds between an instance of a genetic variation and a genomic feature (typically a gene class) that is affected in its sequence or expression.	22
is causal loss of function germline mutation of	RO:0004012	Relates a gene to a condition, such that a mutation in this gene in a germ cell impairs the function of the corresponding product and that is sufficient to produce the condition and that can be passed on to offspring.	15
in 1 to 1 orthology relationship with	RO:HOM0000020	Orthology that involves two genes that did not experience any duplication after the speciation event that created them.	10
is marker for	RO:0002607	x is marker for y if the presence or occurrence of y is correlated with the presence or occurrence of x, and the observation of x is used to infer the presence or occurrence of y. Note that this does not imply that x and y are in a direct causal relationship, as it may be the case that there is a third entity z that stands in a direct causal relationship with x and y.	1
is causal gain of function germline mutation of	RO:0004011	Relates a gene to a condition, such that a mutation in this gene in a germ cell provides a new function of the corresponding product and that is sufficient to produce the condition and that can be passed on to offspring.	1

Table 3.2: *All relations in the HD chorea KG with their identifiers and description (sources: OBO Relations Ontology [39]; GENO ontology [40]). Column 'count' shows the number of edges in each relation group.*

3.2 Edge representations

RDF2Vec created one feature vector for every drug and gene node in the graph, which resulted in 1636 vectors. These drug and gene vectors were then fused into one vector for each drug-gene pair (the edge representations). Because there are 1352 drugs and 284 genes in the graph, 383,968 edge representations have been created, of which 1753 are known interactions (1301 regulates, 391 inhibits, 61 activates). This means 382,215 interactions received the label 'unknown'.

3.2.1 Prediction data

The graph contains 200 genes that are associated with the symptom chorea, which are the genes of interest. There are 1077 known drug-gene edges with these 200 genes, which are the known interactions of interest, meaning there are $200 \times 1352 - 1077 = 269,323$ unknown interactions of interest (the prediction data).

3.2.2 Training data

The training data consists of all known interactions (the positive classes) and has as negative class a random sample of unknown interactions with size equivalent to the largest positive class.

For the negative samples, first the unknown interactions that are in the prediction data were removed, resulting in 112,892 unknown interactions that are **not** in the prediction data. The class in the positive set with the largest amount of interactions is 'regulates' with 1301 interactions. This means that from the unknown edges that are not the prediction data, 1301 edges were randomly selected to be the negative samples for the training data.

So the training data has 1301 regulates edges, 391 inhibits edges, 61 activates edges, and 1301 random unknown edges.

3.3 Supervised machine learning

The best hyperparameters found using randomized search on the search space provided (see Section 2.4) can be seen in Table 3.3. The F1 score of the XGBoost model with these hyperparameters is 0.867. This model was used to predict the unknown interactions of interest in the prediction dataset.

parameter	best
min_child_weight	5
gamma	0.5
reg_alpha	0.5
reg_lambda	3
colsample_bytree	0.8053
max_depth	10
n_estimators	50
learning_rate	0.1258

Table 3.3: *The best found hyperparameters for the XGBoost model.*

3.4 Ranking & validation

3.4.1 Drug ranking

After removing the drug-gene edges that were predicted to have no interaction, 74,323 interactions were left to have a positive prediction. The distribution of the confidence score of each positive prediction (the probability of an interaction belonging to its predicted class) can be seen in Figure 3.3.

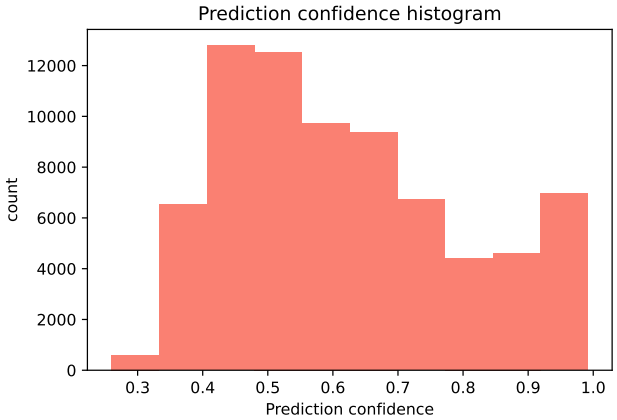


Figure 3.3: *The distribution of the confidence scores obtained with the predict_proba() function of xgboost.*

To shorten the long list of positive predictions, only the predicted interactions with a score higher than 0.9 were kept, resulting in a list of 8173 interactions. These best predicted interactions were then grouped by drug, where drugs with more interactions are ranked higher than drugs with fewer interactions. Drugs with the same amount of interactions are ranked according to the sum of prediction confidences. Table 3.4 gives the two highest ranked drugs and for both drugs shows all genes the drugs have positive predicted interactions with. Both drugs were predicted to interact with eleven genes that contribute to the phenotype chorea. Table 3.5 gives the URI links and labels of the top-10 highest ranked drugs.

3.4.2 Prediction validation

In order to support the predictions made using the drug repurposing approach in this thesis, we explored the top three ranked drugs by looking at whether these drugs are used in other neurodegenerative disorders.

The top ranked drug to reduce chorea in HD patients is CHEMBL29097. According to its compound report card (which was obtained by clicking on the URI link), its ChEMBL synonym is MK-886. MK-886 is an inhibitor of 5-lipoxygenase-activating protein activity. It has been found that 10 microM MK-886 can abolish the biosynthetic production of cysteinyl leukotrienes (CysLTs), which is suggested to be involved in brain inflammation and neurological diseases [41]. We did not find a relation between MK-886 and chorea specifically.

The second-highest ranked drug is baicalein. Baicalein is a flavonoid isolated from the traditional Chinese medicinal herbal Scutellaria baicalensis Georgi. Although we did not find any relation to chorea, we did find relations between baicalein and other neurodegerative disorders. Baicalein has known anti-inflammatory and neuroprotective efficacy in neurodegenerative disease models [42]. Rui *et al.* [42] studied the effects of baicalein on inflammasome-induced neuroinflammation during Parkinson's disease (a neurodegenerative disorder) and found that baicalein can reverse

drug ID	drug label	gene ID	gene label	interaction type	confidence
chembl:CHEMBL8260	CHEMBL29097	HGNC:10555	ATXN2	regulates	0.990
		HGNC:10596	SCN8A	inhibits	0.963
		HGNC:4572	GRIA2	inhibits	0.934
		HGNC:29259	TAOK1	regulates	0.955
		HGNC:1461	CAMK2B	regulates	0.935
		HGNC:4235	GFAP (human)	regulates	0.935
		HGNC:713	ARSA	regulates	0.969
		HGNC:4076	GABRA2	activates	0.908
		HGNC:4580	GRIK2	inhibits	0.974
		HGNC:11005	SLC2A1	regulates	0.914
		HGNC:30035	PIK3R5	inhibits	0.981
chembl:CHEMBL8260	BAICALEIN	HGNC:10555	ATXN2	regulates	0.988
		HGNC:10596	SCN8	inhibits	0.979
		HGNC:4572	GRIA2	inhibits	0.946
		HGNC:29259	TAOK1	regulates	0.963
		HGNC:4584	GRIN1	inhibits	0.911
		HGNC:1461	CAMK2B	regulates	0.918
		HGNC:4235	GFAP (human)	regulates	0.905
		HGNC:713	ARSA	regulates	0.960
		HGNC:4580	GRIK2	inhibits	0.971
		HGNC:2295	CP (human)	regulates	0.915
		HGNC:30035	PIK3R5	inhibits	0.984

Table 3.4: *The two highest ranked drugs with the genes they interact with, the interaction types and prediction confidence.*

drug URI	drug label
`https://identifiers.org/chembl:CHEMBL29097`	CHEMBL29097
`https://identifiers.org/chembl:CHEMBL8260`	BAICALEIN
`https://identifiers.org/chembl:CHEMBL221137`	EMBELIN
`https://identifiers.org/chembl:CHEMBL267345`	AMPHOTERICIN B
`https://identifiers.org/chembl:CHEMBL308688`	5,7-DIMETHOXYISOFLAVONE
`https://identifiers.org/chembl:CHEMBL2110660`	IGMESINE
`https://identifiers.org/chembl:CHEMBL275809`	FR-122047
`https://identifiers.org/chembl:CHEMBL161343`	ARACHIDONOYL GLYCINE
`https://identifiers.org/chembl:CHEMBL585`	TRIAMTERENE
`https://identifiers.org/chembl:CHEMBL1269845`	CHEMBL1269845

Table 3.5: *The top ten ranked drugs for HD chorea.*

MPTP-induced neuroinflammation in mice by suppressing the NLRP3/caspase-1/GSDMD pathway. Zhou *et al.* [43] showed that baicalein significantly improves the biochemical and histopathological condition of Alzheimer's disease (another neurodegenerative disorder) in rats, concluding that baicalein improves behavioural dysfunction induced by Alzheimer's disease. Additionally, a very recent study (June 2022) found that baicalein possibly affects the progression of Alzheimer's disease by regulating the expression of PTGS2 [44].

The last drug that was explored is third-ranked drug embelin. Embelin is a natural product found in plants that possesses interesting biological and pharmacological properties. It is suggested that embelin's long alkyl C10 tail may be useful for cell membrane insertion, which stimulates the cytoprotection in microglia and the antioxidant defence system. It is therefore seen as an interesting compound able to decrease the damage associated with metabolic and neurodegenerative diseases [45].

3.5 Web app

In this section, the working of the web app is demonstrated using screenshots of the app. Figure 3.4 shows the home page of the app. There are three different sections: one for selecting an existing disease graph built for drug repurposing (top left), one for creating a new disease graph (bottom right), and one with all previous predictions (top right).
The existing disease graph section shows all previously created disease graphs, indicated with the disease name and the creation date of the graph (in between round brackets). This section states that if the disease of interest is not in the list, or a newer version is wanted, a new graph should be created using the input box at the bottom of the page.

The section where a new graph can be created asks the user to insert the phenotype OMIM number of the disease of interest, with three examples of numbers to insert.

The section 'previous predictions' shows all predictions previously made with this app. Each prediction is indicated with the disease name, the date of creation of this disease graph, and the symptom for which predictions are made.

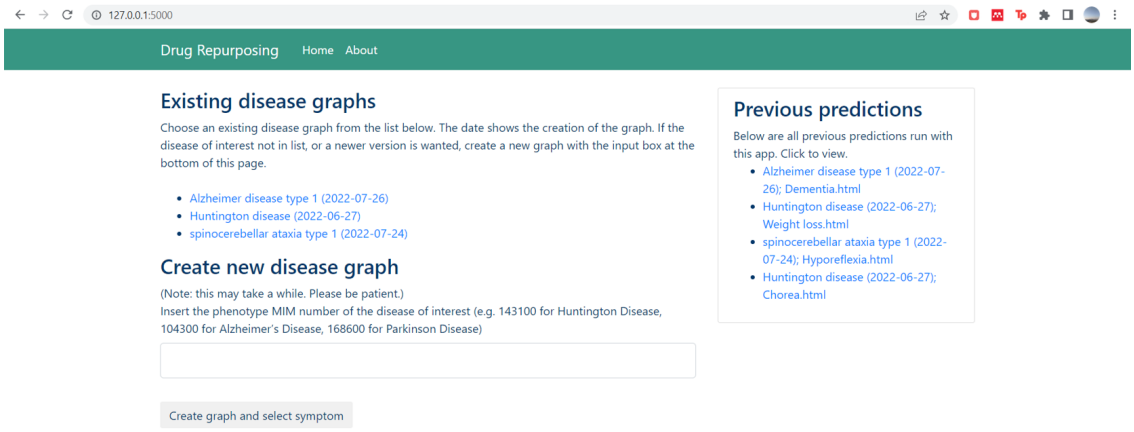


Figure 3.4: *The home page of the drug repurposing web app. It has three sections: one for choosing an existing disease graph, one for creating a new graph, and one for selecting a previous prediction.*

After selecting an existing disease graph or creating a new disease graph, a new page is displayed, as can be seen in Figure 3.5. This page states the name of the disease graph that has been selected or created (in this case the Huntington's disease graph created on the 27th of June 2022) and asks the user to choose a symptom of interest. These symptoms are retrieved using the selected or created disease graph. It gives a warning that after selecting a symptom, a new graph is created and a machine learning model is trained, which takes some time. This page also shows all previous predictions for the user to explore.

After the user selects a symptom of interest (in this case chorea), the page shown in Figure 3.6 is displayed. It states the disease and symptom of interest, the date of creation of the disease graph, and the ranked drug-gene interaction predictions with interaction type and prediction confidence. Each drug/gene label is a clickable link, dereferencing the URI to the description of the gene/drug. This way, the user can easily find more information about a specific drug and gene. This page also includes all previous predictions of the app.

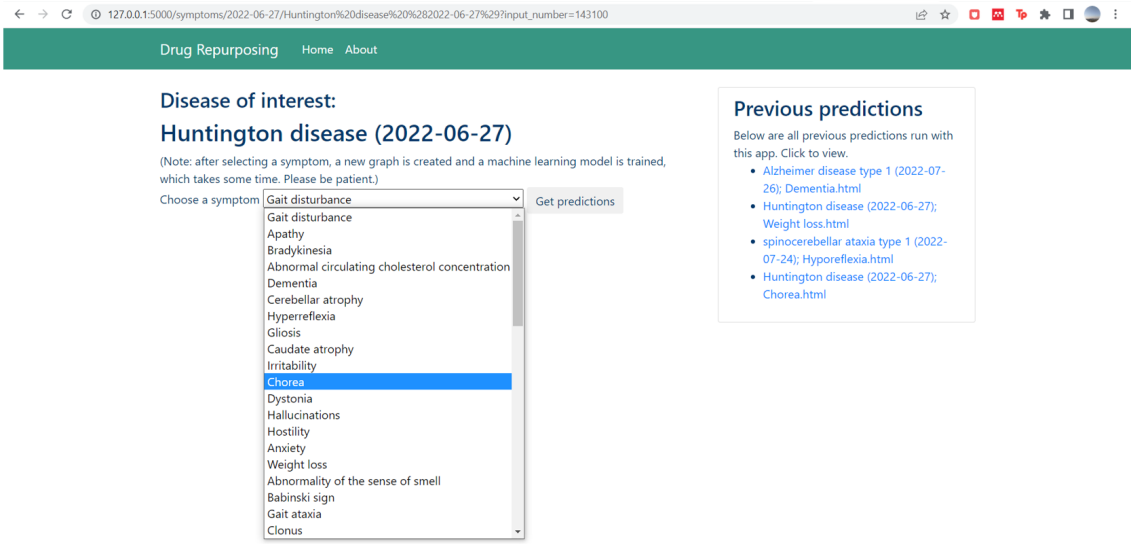


Figure 3.5: *The page of the drug repurposing web app the user lands on after selecting an existing disease graph or inserting a MIM number to create a new disease graph. This page provides the user with a list of symptoms of the selected disease and asks the user to select a symptom of interest. There is also the option to view a previously run prediction.*

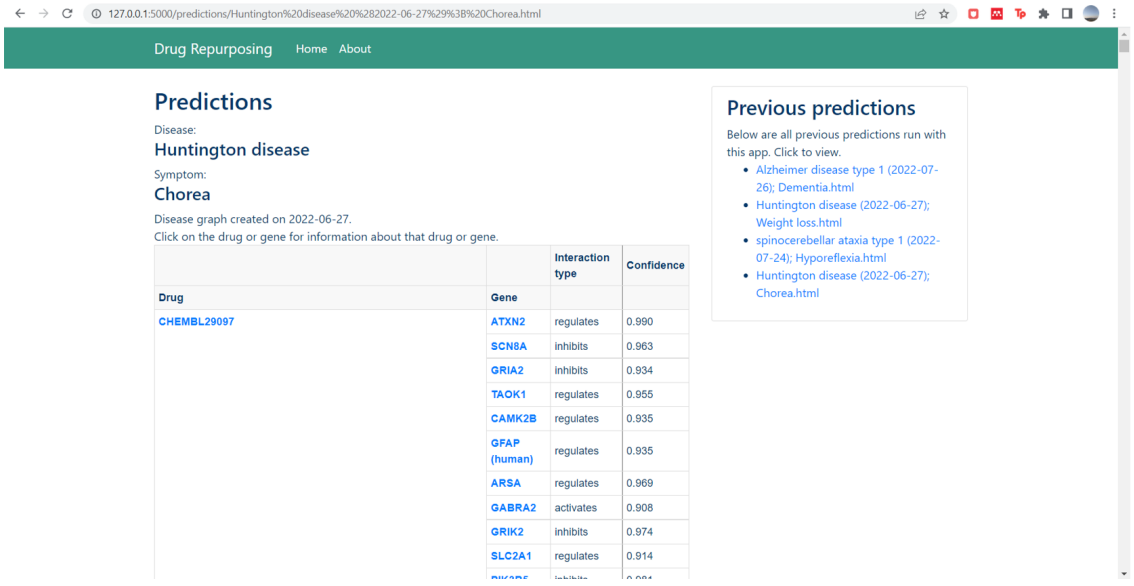


Figure 3.6: *This page appears when the user selects a symptom of interest (after some running time). It shows a list of drugs with all genes they will likely interact with, where the genes are associated with the symptom of interest.*

Chapter 4

Discussion

In this chapter, we mention our achievements, discuss potential improvements, and suggest some future work.

4.1 Achievements

The aim of this thesis was to find novel drug-target interactions to alleviate symptoms of a specific rare disease using the Bioknowledge reviewer data mining and integration approach together with the biophysical drug repurposing approach. We managed to create an automated drug repurposing workflow that uses network and graph-based analysis methods together with machine learning to produce a ranked list of candidate compounds. We created a web app as a user-friendly interface to the drug repurposing workflow to be used by researchers with no technical expertise.
With our approach, we built the first drug repurposing web application that creates a heterogeneous network using an integrative platform and performs network- and machine learning methods to predict candidate compounds.
We managed to run the workflow and find drug candidates for Huntington's Disease with symptom chorea, for which the top three found drugs could all be linked to a neurodegenerative disease. The app can also easily be run for other diseases. We tested the app for two additional diseases: Alzheimer disease type 1 with symptom dementia and spinocerebellar ataxia type 1 (SCA1) with symptom hyporeflexia, which provided a new list of candidate compounds, but have not been validated yet.

4.2 Improvements

Although using the method as it is already gives results that could be interesting to test experimentally, a few improvements of the workflow could be made.

First, because the drug-gene interactions are predicted using supervised learning for classification, negative samples are needed. Similarly to most drug-gene interaction prediction studies that use classification, the approach discussed in this thesis uses unknown drug-gene pairs as negative samples. However, among these negative samples, there may be positive samples that are just not yet known, so the performance of a machine learning model based on such a method may be biased [46]. Better would be to select realistic negative interactions, for example by using the ChEMBL database to collect experimentally validated negative interactions, as Amiri Souri *et al.* [47] stated in their publication. However, this is a challenging task due to the lack of experimentally validated negative interactions.

Secondly, as machine learning methods only perform well if the datasets used are of good quality [5], more effort should be put into the creation of a good knowledge graph. One way to improve the graph is by specifying more input seeds for Monarch. In the current drug repurposing workflow, Monarch takes as input only two nodes, a disease seed and a symptom seed. It would be better to include seeds like genes that are associated with the disease and/or metabolites. Queralt-Rosinach *et al.* [6] built a knowledge graph focused on NGLY1 Deficiency, for which they used nine seeds: the disease, seven genes, and one metabolite. Including more seeds requires domain expert knowledge about the disease, which the lab researchers using the app would probably have. Besides more genes and metabolites, it would also be beneficial to include related diseases and/or related symptoms as seeds. However, it should be kept in mind that the more seeds, the more computation time needed for each step of the drug repurposing workflow (creating the graph, adding drugs, finding drug similarity edges, turning the graph into an RDF graph, creating a machine learning model).

Thirdly, the biophysical drug repurposing approach is used in this thesis, which assumes that chemicals with similar structure have shared biological activity. However, this is not always the case: there could be errors in the chemical structures or physiological effects that exist beyond the structure of the chemical [4]. It would therefore be good to include more information about each drug in the KG like side effects or links between drugs and diseases. In that case, the drug embeddings would not be solely based on chemical structure similarity.

The final point to discuss is that a lot of parameter settings for different methods used in the drug repurposing workflow have been chosen based on similar methods done by others. Given more time, it would be better to perform a systematic optimisation of the hyperparameters. For example, for creating the drug similarity edges, our workflow finds the top k most similar edges for each drug in the graph using the k-nearest neighbour algorithm, for which we set k=10. As mentioned by Thafar et al. (2021), choosing this value for k is not an easy task [18]. If k is set too large, the process becomes computationally expensive, if k is set too small, some useful information could be left out which could affect the performance. It would be best to conduct some experiments to determine the optimal value for k.
For RDF2Vec, the parameters for the random walk could be adjusted to obtain better results (max_depth and num_walks). According to Huang *et al.* [48], longer walk lengths improve the embedding quality, so better would be to set the max_depth to, for example, 20 instead of current setting of 4.
For the final drug list, we only kept the predicted interactions with a confidence score higher than 0.9. For better predictions, a higher minimal confidence score could be chosen, but the higher the minimal score, the fewer interactions remain. It would be a possibility to add a new feature to the web app, where the user decides how confident the predictions should be.

4.3 Future work

4.3.1 FAIR

This project was carried out with FAIR principles in mind. FAIR stands for findable, accessible, interoperable, and reusable. The knowledge graph is currently in the form of an RDF graph, which, as mentioned before, is a standard model for data interchange on the Web [26]. Because the FAIR principles state that data resources, tools, vocabularies and infrastructures should assist discovery and reuse by third-parties through the Web [49], an RDF graph could be referred to as a FAIR knowledge graph. Even though our workflow is containerised and accessible (`https://git.lumc.nl/catreep/drugrepurposing`), each graph created with our drug repurposing approach is not yet stored in a place where it is findable, accessible, interoperable and reusable. The RDF graphs should be made accessible on the web for people to upload to their own triple stores, or properly hosted so that they can be queried. It would also be a good addition to the workflow to

store the RDF graph in a database like Neo4j, which enables storage, management, and mining of structured graph data [6]. The nodes and edges graph csv files created in this thesis are in the correct format for the Python file *neo4jlib.py* of the BioKnowledge reviewer library, which sets up a Neo4j server instance and loads the review knowledge graph into Neo4j [6]. Then in the Flask web app, having the option to go to the created graph in Neo4j for viewing, querying and mining would make this work FAIRer.

4.3.2 Web app validation

The web app is currently not validated. The app should be tested by different users to ensure that all necessary information is provided and properly formatted in order to successfully run the drug repurposing workflow.

Chapter 5

Conclusion

In this thesis, we addressed the research question: **'Can the BioKnowledge reviewer data mining and integration approach together with the biophysical drug repurposing approach be used to create an automated drug repurposing workflow that finds novel drug-target interactions to alleviate symptoms of a specific rare disease?'**. We managed to develop an automated data mining drug repurposing workflow that uses the BioKnowledge reviewer library to extract information from the platform Monarch, adds drugs using the DGIdb database, finds similarity edges between drugs based on their compound structure, turns the resulting combined knowledge graph into an RDF graph and uses RDF2Vec to input this graph into an XGBoost supervised machine learning model. The best hyperparameters for XGBoost are found and used in the final model to classify unknown drug-gene interactions of interest to be inhibiting, activating, regulating, or not interacting. The final predicted interactions are grouped on drugs, then ranked based on the number of positive gene interactions each drug has (with a high prediction confidence).
We turned this whole drug repurposing workflow into a web app that asks the user for a disease and symptom of interest, after which it produces the ranked list of drugs. The intention is that lab researchers could use the app to find potential candidate compounds to test experimentally.

Our drug repurposing workflow can be improved, for example by letting the user add more input seeds for Monarch, adjusting some parameters that could lead to higher predictions, adding more information about each drug, and making the method more FAIR by storing the knowledge graph in a database like Neo4j, which also enables the user to view and query the graph.
However, using the method as it is already gives results that could be interesting to test experimentally. For Huntington's Disease with symptom chorea, the top three found drugs are CHEMBL29097, BAICALEIN, and EMBELIN, which could all three be linked to a neurodegenerative disease in some way.

Bibliography

[1] S. Shah, M. M. Dooms, S. Amaral-Garcia *et al.*, "Current drug repurposing strategies for rare neurodegenerative disorders," *Frontiers in Pharmacology*, vol. 12, Dec. 2021. [Online]. Available: https://doi.org/10.3389/fphar.2021.768023.

[2] H. I. Roessler, N. V. Knoers, M. M. van Haelst *et al.*, "Drug repurposing for rare diseases," *Trends in Pharmacological Sciences*, vol. 42, no. 4, pp. 255–267, Apr. 2021. [Online]. Available: https://doi.org/10.1016/j.tips.2021.01.003.

[3] T. B. Malas, W. J. Vlietstra, R. Kudrin *et al.*, "Drug prioritization using the semantic properties of a knowledge graph," *Scientific Reports*, vol. 9, no. 1, Apr. 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-42806-6.

[4] S. Pushpakom, F. Iorio, P. A. Eyers *et al.*, "Drug repurposing: Progress, challenges and recommendations," *Nature Reviews Drug Discovery*, vol. 18, no. 1, pp. 41–58, Oct. 2018. [Online]. Available: https://doi.org/10.1038/nrd.2018.168.

[5] M. D. Paranjpe, A. Taubes and M. Sirota, "Insights into computational drug repurposing for neurodegenerative disease," *Trends in Pharmacological Sciences*, vol. 40, no. 8, pp. 565–576, Aug. 2019. [Online]. Available: https://doi.org/10.1016/j.tips.2019.06.003.

[6] N. Queralt-Rosinach, G. S. Stupp, T. S. Li *et al.*, "Structured reviews for data and knowledge-driven research," *Database*, vol. 2020, Jan. 2020. [Online]. Available: https://doi.org/10.1093/database/baaa015.

[7] R. A. Roos, "Huntington's disease: A clinical review," *Orphanet Journal of Rare Diseases*, vol. 5, no. 1, Dec. 2010. [Online]. Available: https://doi.org/10.1186/1750-1172-5-40.

[8] C. J. Mungall, J. A. McMurry, S. Köhler *et al.*, "The monarch initiative: An integrative data and analytic platform connecting phenotypes to genotypes across species," *Nucleic Acids Research*, vol. 45, no. D1, pp. D712–D722, Nov. 2016. [Online]. Available: https://doi.org/10.1093/nar/gkw1128.

[9] J. Malone, R. Stevens, S. Jupp *et al.*, "Ten simple rules for selecting a bio-ontology," *PLOS Computational Biology*, vol. 12, no. 2, S. Markel, Ed., e1004743, Feb. 2016. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1004743.

[10] S. L. Freshour, S. Kiwala, K. C. Cotto *et al.*, "Integration of the drug–gene interaction database (DGIdb 4.0) with open crowdsource efforts," *Nucleic Acids Research*, vol. 49, no. D1, pp. D1144–D1151, Nov. 2020. [Online]. Available: https://doi.org/10.1093/nar/gkaa1084.

[11] C. Wu, *Biothings$_c$lient.py*, https://github.com/biothings/biothings_client.py, 2022.

[12] J. Balhoff, *Obo-relations*, https://github.com/oborel/obo-relations, 2022.

[13] A. Kumar, "Chemical similarity methods : A tutorial review," *The Chemical Educator*, Feb. 2011.

[14] H. Öztürk, E. Ozkirimli and A. Özgür, "A comparative study of SMILES-based compound similarity functions for drug-target interaction prediction," *BMC Bioinformatics*, vol. 17, no. 1, Mar. 2016. [Online]. Available: https://doi.org/10.1186/s12859-016-0977-x.

[15] *Pubchem compound summary for cid 6018, tetrabenazine.* https://pubchem.ncbi.nlm.nih.gov/compound/Tetrabenazine, 2022.

[16] *Rdkit: Open-source cheminformatics*, http://www.rdkit.org.

[17] A. Capecchi, D. Probst and J.-L. Reymond, "One molecular fingerprint to rule them all: Drugs, biomolecules, and the metabolome," *Journal of Cheminformatics*, vol. 12, no. 1, 2020. [Online]. Available: https://doi.org/10.1186%2Fs13321-020-00445-4.

[18] M. A. Thafar, R. S. Olayan, S. Albaradei *et al.*, "DTi2vec: Drug–target interaction prediction using network embedding and ensemble learning," *Journal of Cheminformatics*, vol. 13, no. 1, Sep. 2021. [Online]. Available: https://doi.org/10.1186/s13321-021-00552-w.

[19] C. R. Harris, K. J. Millman, S. J. van der Walt *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[20] G. Klyne and J. J. Carroll. "Resource description framework (rdf): Concepts and abstract syntax," W3C. (2004), [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (visited on 15/03/2015).

[21] M. Dumontier, C. J. Baker, J. Baran *et al.*, "The semanticscience integrated ontology (SIO) for biomedical research and knowledge discovery," *Journal of Biomedical Semantics*, vol. 5, no. 1, Mar. 2014. [Online]. Available: https://doi.org/10.1186/2041-1480-5-14.

[22] S. M. Wimalaratne, N. Juty, J. Kunze *et al.*, "Uniform resolution of compact identifiers for biomedical data," *Scientific Data*, vol. 5, no. 1, May 2018. [Online]. Available: https://doi.org/10.1038/sdata.2018.29.

[23] S. Kawashima, T. Katayama, H. Hatanaka *et al.*, "NBDC RDF portal: A comprehensive repository for semantic data in life sciences," *Database*, vol. 2018, Jan. 2018. [Online]. Available: https://doi.org/10.1093/database/bay123.

[24] I. Aucamp, *Rdflib*, https://github.com/RDFLib/rdflib, 2021.

[25] *Rdf 1.1 turtle*, https://www.w3.org/TR/turtle/, 2014.

[26] R. Celebi, H. Uyar, E. Yasar *et al.*, "Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction in realistic settings," *BMC Bioinformatics*, vol. 20, no. 1, Dec. 2019. [Online]. Available: https://doi.org/10.1186/s12859-019-3284-5.

[27] P. Ristoski and H. Paulheim, "RDF2vec: RDF graph embeddings for data mining," in *Lecture Notes in Computer Science*, Springer International Publishing, 2016, pp. 498–514. [Online]. Available: https://doi.org/10.1007/978-3-319-46523-4_30.

[28] G. Vandewiele, B. Steenwinckel, T. Agozzino *et al.*, "pyRDF2Vec: Python Implementation and Extension of RDF2Vec," IDLab, 2020. [Online]. Available: https://github.com/IBCNServices/pyRDF2Vec.

[29] A. Grover and J. Leskovec, *Node2vec: Scalable feature learning for networks*, 2016. [Online]. Available: https://arxiv.org/abs/1607.00653.

[30] J. Portisch and H. Paulheim, *Putting rdf2vec in order*, 2021. eprint: arXiv:2108.05280.

[31] F. J. Provost, "Machine learning from imbalanced data sets 101," 2008.

[32] A. Paleczek, D. Grochala and A. Rydosz, "Artificial breath classification using XGBoost algorithm for diabetes detection," *Sensors*, vol. 21, no. 12, p. 4187, Jun. 2021. [Online]. Available: https://doi.org/10.3390/s21124187.

[33] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] Y. Jian, M. Pasquier, A. Sagahyroon *et al.*, "A machine learning approach to predicting diabetes complications," *Healthcare*, vol. 9, no. 12, p. 1712, Dec. 2021. [Online]. Available: https://doi.org/10.3390/healthcare9121712.

[36] A. Sertkaya, H.-H. Wong, A. Jessup *et al.*, "Key cost drivers of pharmaceutical clinical trials in the united states," en, *Clin. Trials*, vol. 13, no. 2, pp. 117–126, Apr. 2016.

[37] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[38] *What is flask python*, https://pythonbasics.org/what-is-flask-python/.

[39] C. Mungall, J. A. Overton, D. Osumi-Sutherland *et al.*, *Obo-relations: 2015-10-29 release*, 2015. [Online]. Available: https://zenodo.org/record/32899.

[40] M. Brush, N. Matentzoglu and M. Haendel, *Geno-ontology*, 2022. [Online]. Available: https://github.com/monarch-initiative/GENO-ontology.

[41] P. Ballerini, P. D. Iorio, R. Ciccarelli *et al.*, "P2ysub1/sub and cysteinyl leukotriene receptors mediate purine and cysteinyl leukotriene co-release in primary cultures of rat microglia," *International Journal of Immunopathology and Pharmacology*, vol. 18, no. 2, pp. 255–268, Apr. 2005. [Online]. Available: https://doi.org/10.1177/039463200501800208.

[42] W. Rui, S. Li, H. Xiao *et al.*, "Baicalein attenuates neuroinflammation by inhibiting NLRP3/caspase-1/GSDMD pathway in MPTP induced mice model of parkinson's disease," en, *Int. J. Neuropsychopharmacol.*, vol. 23, no. 11, pp. 762–773, 2020.

[43] L. Zhou, S. Tan, Y. long Shan *et al.*, "Baicalein improves behavioral dysfunction induced by alzheimer&amprsquos disease in rats," *Neuropsychiatric Disease and Treatment*, vol. Volume 12, pp. 3145–3152, Dec. 2016. [Online]. Available: https://doi.org/10.2147/ndt.s117469.

[44] T. Xie, Y. Pei, P. Shan *et al.*, "Identification of miRNA–mRNA pairs in the alzheimer's disease expression profile and explore the effect of miR-26a-5p/PTGS2 on amyloid- induced neurotoxicity in alzheimer's disease cell model," *Frontiers in Aging Neuroscience*, vol. 14, Jun. 2022. [Online]. Available: https://doi.org/10.3389/fnagi.2022.909222.

[45] F. Caruso, M. Rossi, S. Kaur *et al.*, "Antioxidant properties of embelin in cell culture. electrochemistry and theoretical mechanism of scavenging. potential scavenging of superoxide radical through the cell membrane," *Antioxidants*, vol. 9, no. 5, p. 382, May 2020. [Online]. Available: https://doi.org/10.3390/antiox9050382.

[46] L. Xu, X. Ru and R. Song, "Application of machine learning for drug–target interaction prediction," *Frontiers in Genetics*, vol. 12, Jun. 2021. [Online]. Available: https://doi.org/10.3389/fgene.2021.680117.

[47] E. A. Souri, R. Laddach, S. N. Karagiannis *et al.*, "Novel drug-target interactions via link prediction and network embedding," *BMC Bioinformatics*, vol. 23, no. 1, Apr. 2022. [Online]. Available: https://doi.org/10.1186/s12859-022-04650-w.

[48] K. Huang, C. Xiao, L. M. Glass *et al.*, "SkipGNN: Predicting molecular interactions with skip-graph networks," *Scientific Reports*, vol. 10, no. 1, Dec. 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-77766-9.

[49] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, Mar. 2016. [Online]. Available: https://doi.org/10.1038/sdata.2016.18.

[50] Prateek, *Basics of flask*, https://www.codementor.io/@overiq/basics-of-flask-fzvh8ueed, 2018.

[51] *Wtforms*, https://wtforms.readthedocs.io/en/3.0.x/, 2018.

Appendix A

Web app functioning

We saved the whole drug repurposing workflow as described in this thesis in five separate Python files: *Monarch.py*, *DGIdb.py*, *drugsimilarity.py*, *combine_graphs.py*, and *rdf2vec.py*. To run this whole process with Flask, we created three extra Python files: *__init__.py*, *routes.py*, and *forms.py*.

The *__init__.py* file imports the *Flask* class from the *flask* package version 1.1.2 and instantiates the Flask app. The *routes.py* file specifies the routes of the app, which are acts that bind a URL to a view function (a function that responds to a request) [50]. The *forms.py* file includes form classes for user input and uses WTForms [51] to render and validate the forms.

Each time the user makes a request (by clicking on a link or giving input via a form), the *render_template()* function is run, which generates output from a template file that is found in the 'templates' folder of the app. For a nice general style of the app, each template file extends the *layout.html* template, which provides a site header and title and a section with links of all previously run predictions.

To start the server, the *run()* method of the *Flask* object is called. It returns the URL where the server is available. When opening the URL, the home page of the app is shown, which is rendered from the *home.html* template file. On this home page, there are several options for the user. The user can choose an existing disease graph, which has been previously created using a disease seed as input for Monarch, where the date of creation of the graph is specified. If the user is interested in a disease that is not in this list of previously created graphs, or the user wants a newer version of a graph, the user can create a new disease graph by specifying the phenotype MIM number of the disease of interest (e.g. '143100' for Huntington's disease). This enables the Python *Monarch.py* file to run the function that creates a new network from Monarch using this input number as seed. Creating a new network form Monarch takes some time, so the user is asked to be patient. After running a new disease graph, the graph is saved and added to the existing disease graphs on the homepage.

After selecting an existing disease graph, or creating a new disease graph, the *symptoms.html* template is rendered. Here, all symptoms of the disease of interest are listed, and the user is asked to select one symptom of interest. When a user selects a symptom, again, the Python *Monarch.py* file is called to run the function that creates a new network from Monarch, with this time only the chosen symptom as seed. After creating this symptom Monarch graph, this graph is merged with the disease Monarch graph to create the final Monarch graph. Then *DGIdb.py*, *drugsimilarity.py*, *combine_graphs.py*, and *rdf2vec.py* are run in this order. The last Python file saves the ranked predicted drug-gene interactions as an HTML file in the *templates* folder of the app. When everything is run, this predictions HTML file is rendered and the list of ranked predictions is shown on screen. This list includes the drugs, the genes each drug interacts with, together with the interaction type and confidence. Each drug and gene is presented with its label. Clicking on a label will direct the

user to the URI link of the drug or gene, which shows all up-to-date information about that drug or gene.

On every page (home, symptom, prediction), we added the option to select a previously run prediction, to avoid long running times.

Every time the workflow is run, around 1 GB of data is created. To avoid taking up too much space on the server, all created files (CSV files, Turtle files, pickle files) are deleted after the final predictions HTML file is created and rendered. Only the predictions HTML files and the Monarch disease CSV files are kept, as these are needed when a user selects a previously run prediction or an existing disease graph.